以AppWizard设计充电桩GUI项目

NuMicro® 32 位系列微控制器范例代码介绍

文件信息

应用简述	本范例代码使用 emWin AppWizard 工具，结合 M467 系列开发平台与开发资源，完成充电桩图形化界面设计。
BSP 版本	M460_Series_BSP_CMSIS_V3.00.002 emWin Software Package for M460 V3.0
开发平台	NuMaker-M467HJ V1.0

The information described in this document is the exclusive intellectual property of Nuvoton Technology Corporation and shall not be reproduced without permission from Nuvoton.

Nuvoton is providing this document only for reference purposes of NuMicro microcontroller and microprocessor based system design. Nuvoton assumes no responsibility for errors or omissions.

All data and specifications are subject to change without notice.

For additional information or questions, please contact: Nuvoton Technology Corporation.

www.nuvoton.com

1. 概述

本范例代码将示范如何使用 SEGGER emWin AppWizard 工具，搭配 NuMaker-M467HJ V1.0 开发平台，逐步完成一个充电桩项目的画面设计、基本功能演示。

1.1 原理

1.1.1 系统方块图

此范例代码直接使用新唐 GUI 开发平台 - NuMaker-M467HJ V1.0 开发板，开发时会使用到的组件包括分辨率 480x272 且屏幕大小 4.3 吋的 TFT-LCD 电容式触控面板、主控板上 NuMicro M467HJHAN 微控制器、以及容量大小 4Mbyte 的 SPI Flash。其系统方块图如图 1-2 所示。

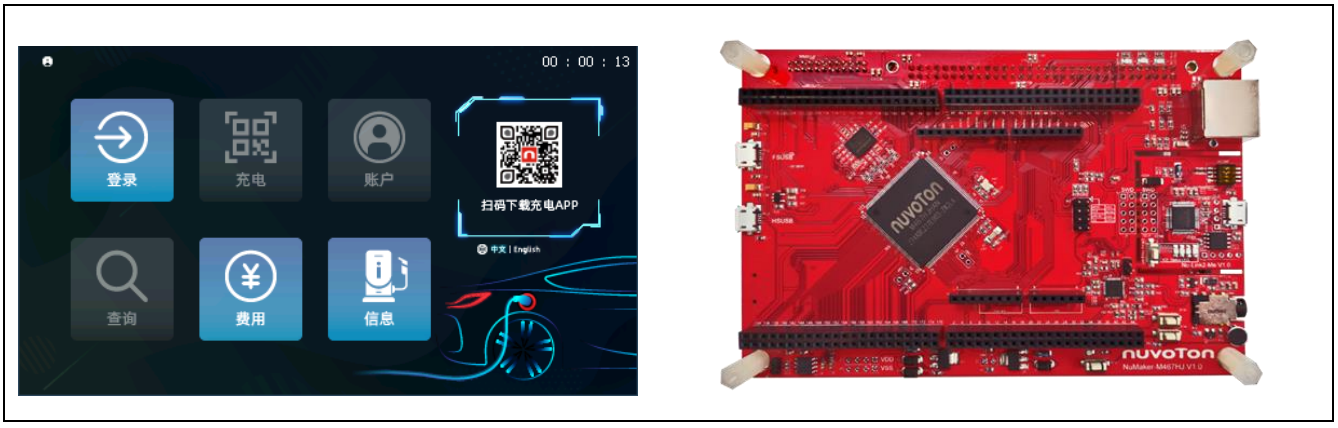


图 1-1 NuMaker-M467HJ V1.0 开发板

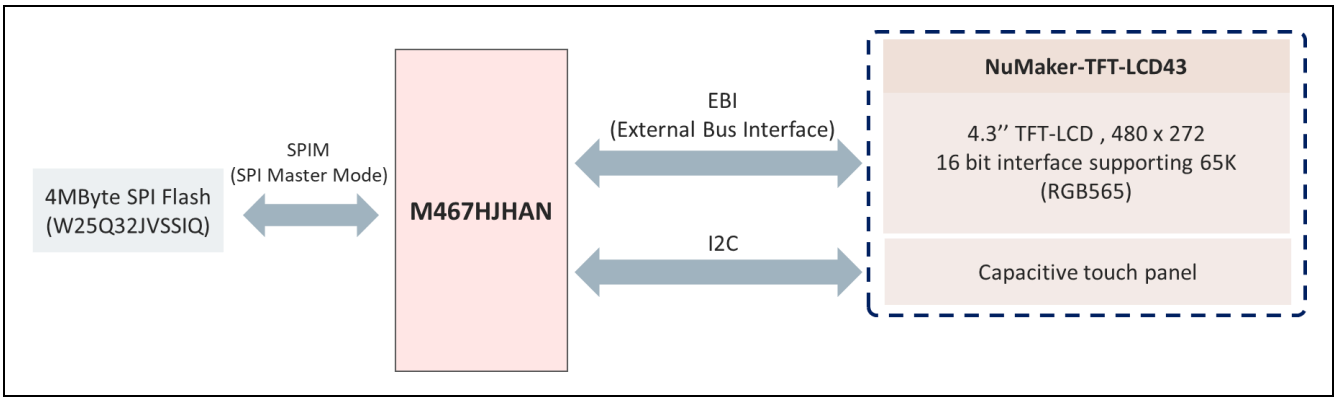
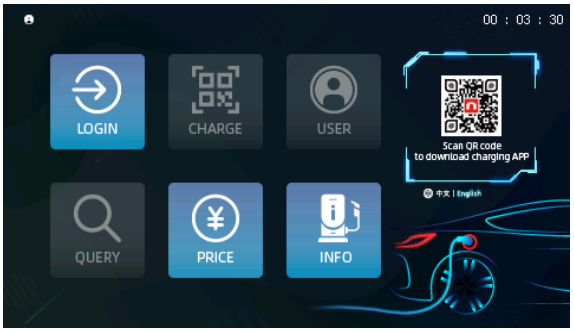
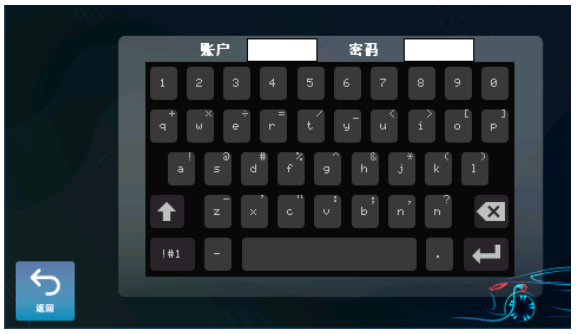
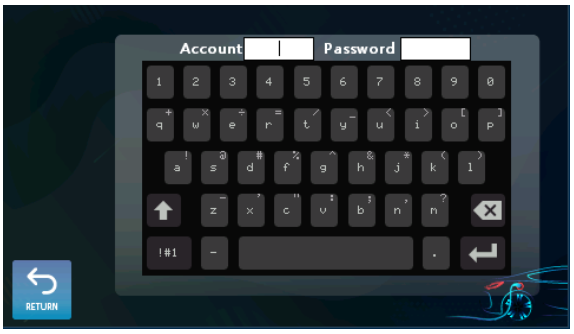
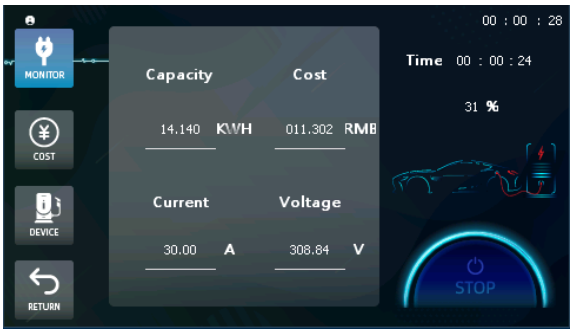
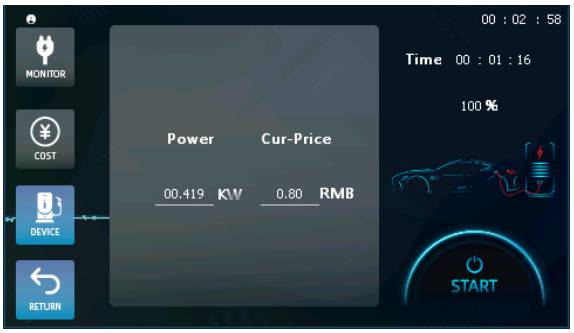
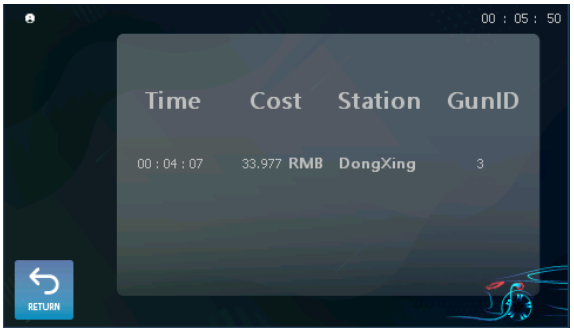


图 1-2 系统方块图

1.1.2 充电桩画面

此范例设计画面如下：

	中文画面	英文画面
屏幕一		
屏幕二		
屏幕三		

屏幕三		
屏幕三		
屏幕四		
屏幕五		

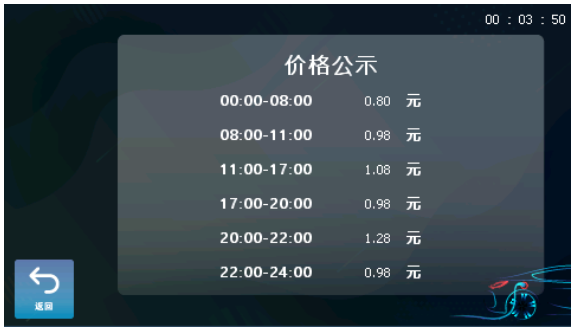
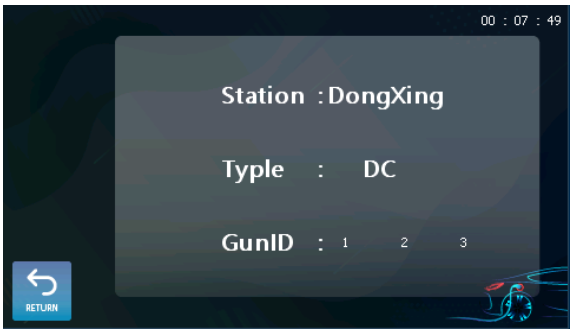
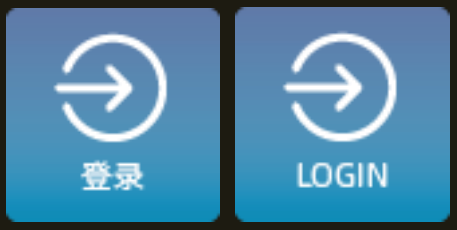
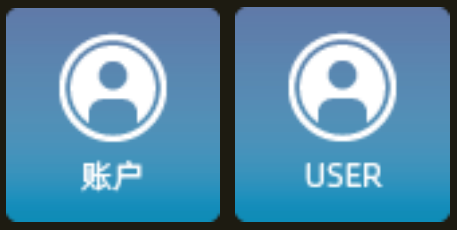
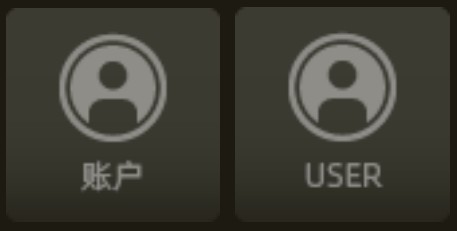
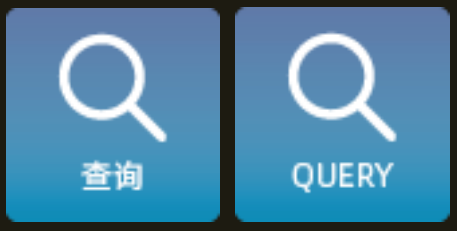
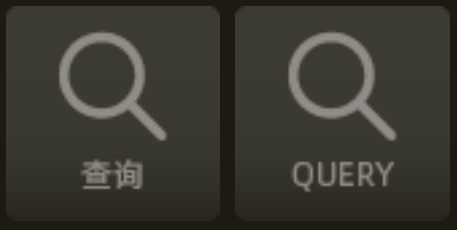
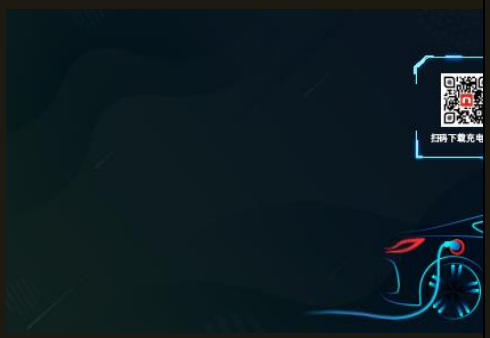
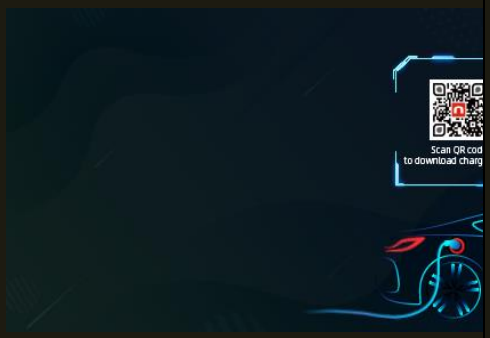
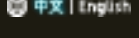
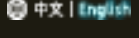
屏幕六		
屏幕七		

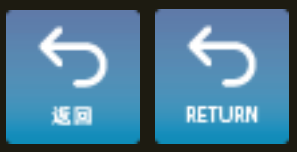
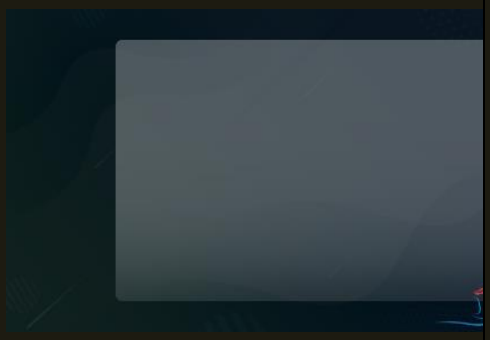
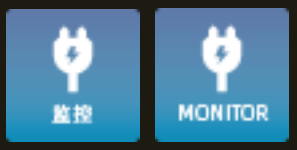
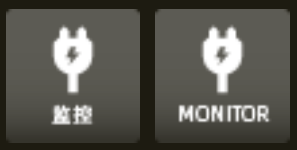
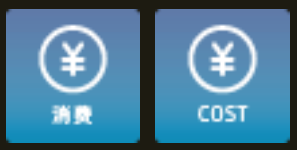
表 1-1 充电桩画面

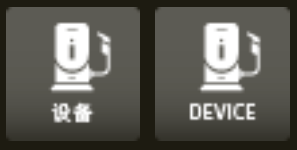
1.1.3 充电桩画面使用元素总览

ChargePile App Des	Icon Name	ChargePile App GUI	Object
登录键有效	Login_Pressed_Ini、Login_Pressed_Ini_En		Image
充电键有效	Charge_Pressed_Ini、Charge_Pressed_Ini_En		

充电键无效	Charge_Lock_Ini、 Charge_Lock_Ini_En	
账户键有效	Account_Pressed_Ini、 Account_Pressed_Ini_En	
账户键无效	Account_Lock_Ini、 Account_Lock_Ini_En	
查询键有效	Query_Pressed_Ini、 Query_Pressed_Ini_En	
查询键无效	Query_Lock_Ini、 Query_Lock_Ini_En	
价格键有效	Price_Pressed_Ini、 Price_Pressed_Ini_En	

价格键无效	Price_Ini、 Price_Ini_En	
信息键有效	Info_Pressed_Ini、 Info_Pressed_Ini_En	
信息键无效	Info_Ini、Info_Ini_En	
初始屏幕背景	Bg2、Bg2_En	 
中文	Lan_Cn	
英文	Lan_En	

用户	Head_Ini	
返回键有效	Return1、Return1_En	
返回键无效	Return0、Return0_En	
其他屏幕背景	Bg3	
监控键有效	Mon、Mon_En	
监控键无效	Mon_0、Mon_0_En	
消费键有效	Cost、Cost_En	
消费键无效	Cost_0、Cost_0_En	
设备键有效	Device、Device_En	

设备键无效	Device_0、Device_0_En	
启动充电	Start	
停止充电	Stop_Pressed	
0%电量	che0	
0%~19%电量	che1	
20%~39%电量	che2	
40%~59%电量	che3	
60%~79%电量	che4	
80%~100%电量	che5	

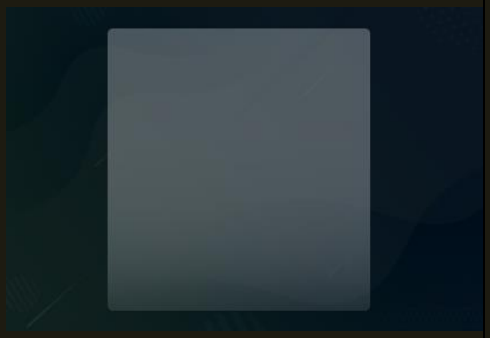
充电屏幕背景	Bg_Cha	
连接线	Connect	
短数据下划线	ExpenseLine	
长数据下划线	MonLine	

表 1-2 画面元素

1.1.4 相关资源

- SEGGER emWin 官方论坛：Segger emWin Forum
- 新唐官方论坛：Nuvoton emWin Forum
- SEGGER emWin AppWizard 工具使用手册下载：AppWizard User Manual

1.2 执行步骤

使用 AppWizard 工具完成一项 GUI 项目的执行步骤可分为五大阶段，第一阶段需准备硬件与软件下载，第二阶段则于 AppWizard 中完成对应的画面设计、object 设定，第三阶段则于 AppWizard 中完成 Interactions 设定、动画设定，第四阶段使用 AppWizard 导出代码并同时加入客制化代码，以完成更多应用功能，第五阶段则是实际烧录到板端。

以下于 1.2.1-1.2.5 就各项阶段详细说明执行步骤。

1.2.1 准备相关硬件与软件资源

1. 硬件使用 NuMaker-M467HJ V1.0，相关资源可参考 参考线路、开发板使用手册。

2. 软件资源需下载 AppWizard 软件工具。

   1) 请点击以下链接 emWin Software Package for M460 进入下载页面，点选 emWin Software Package for M460 项目即可开始下载，下载前需注册与登入新唐官网会员。

   2) 下载完成后，解压缩，开启 M460_emWin_NonOS 文件夹。

   3) 依表 1-3 路径，点击 AppWizard_V134a_630a_Install.exe 安装 AppWizard 工具，需注意若有旧的 AppWizard 版本，需将旧版本完全移除后，才能安装新版本。

M460_emWin_NonOS emWin Tool AppWizard_V134a_630a_Install.exe

表 1-3 AppWizard 工具安装文件路径

1.2.2 在 AppWizard 工具中编辑与设定 GUI

1. 新建工程，并添加项目所需屏幕。

   1) 于桌面快捷图示开启 AppWizard 工具。

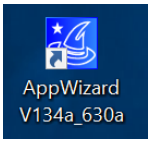


图 1-3 AppWizard 桌面快捷方式

2) 点选 “Create new project” 以创建新专案。

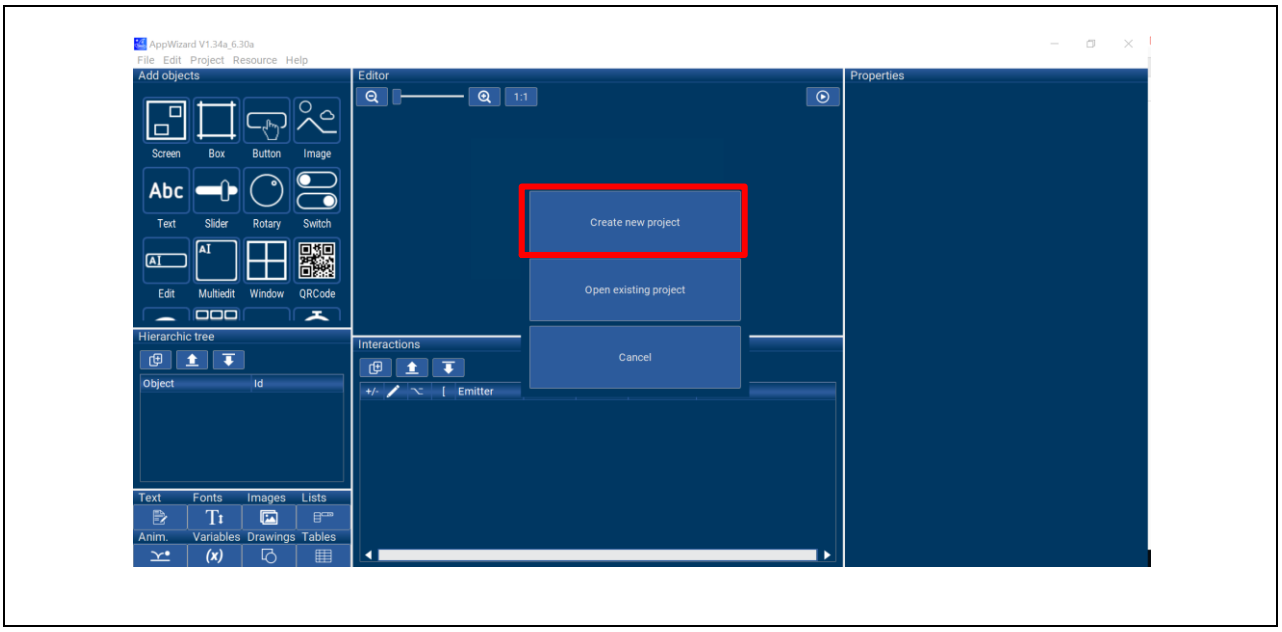


图 1-4 创建专案页面

3) 依下图设定项目路径、名称，符合使用屏幕的颜色格式：16 Bit, GUICC_M565，勾选 “Enable Multibuffering”，点选 “ok”。

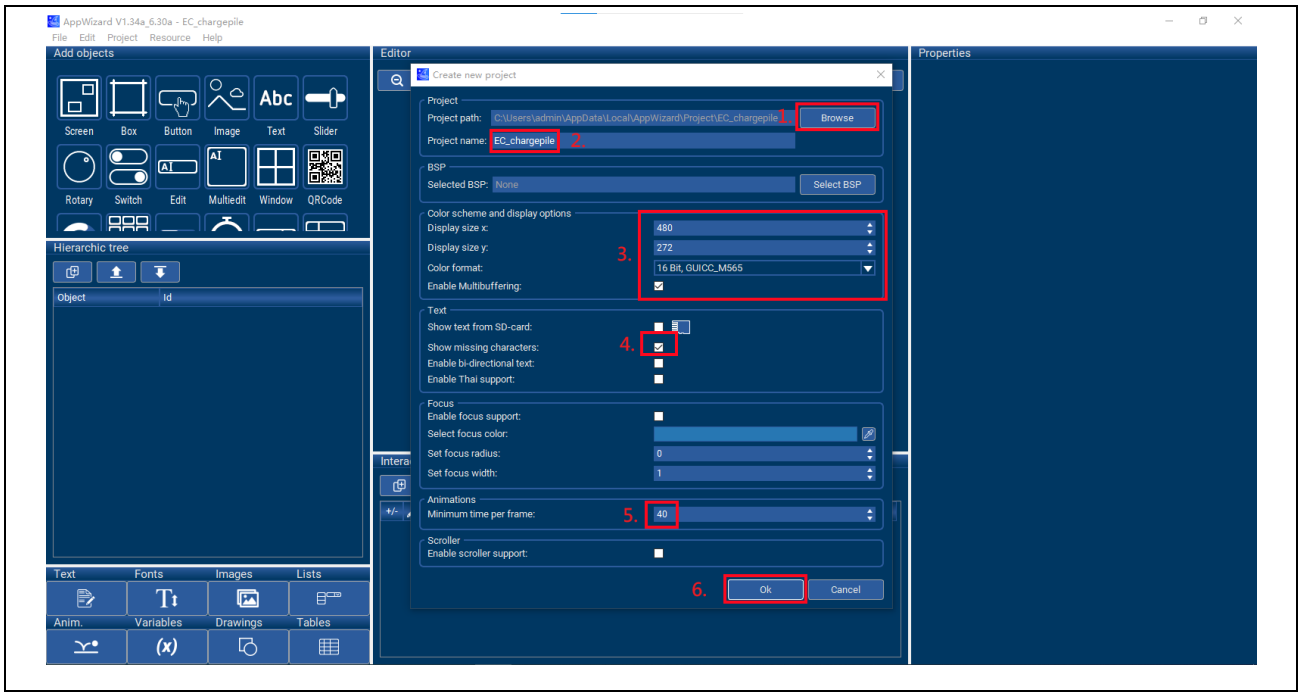


图 1-5 新专案设定项目

4) 在“Add objects”下点选七次“Screen”，以创建七个屏幕，并在“Hierarchic tree”中，为 Screen 分别命名为 ID_SCREEN_Init、ID_SCREEN_Login、ID_SCREEN_Charge、ID_SCREEN_Account、ID_SCREEN_Query、ID_SCREEN_Price、ID_SCREEN_Info 以利辨认。

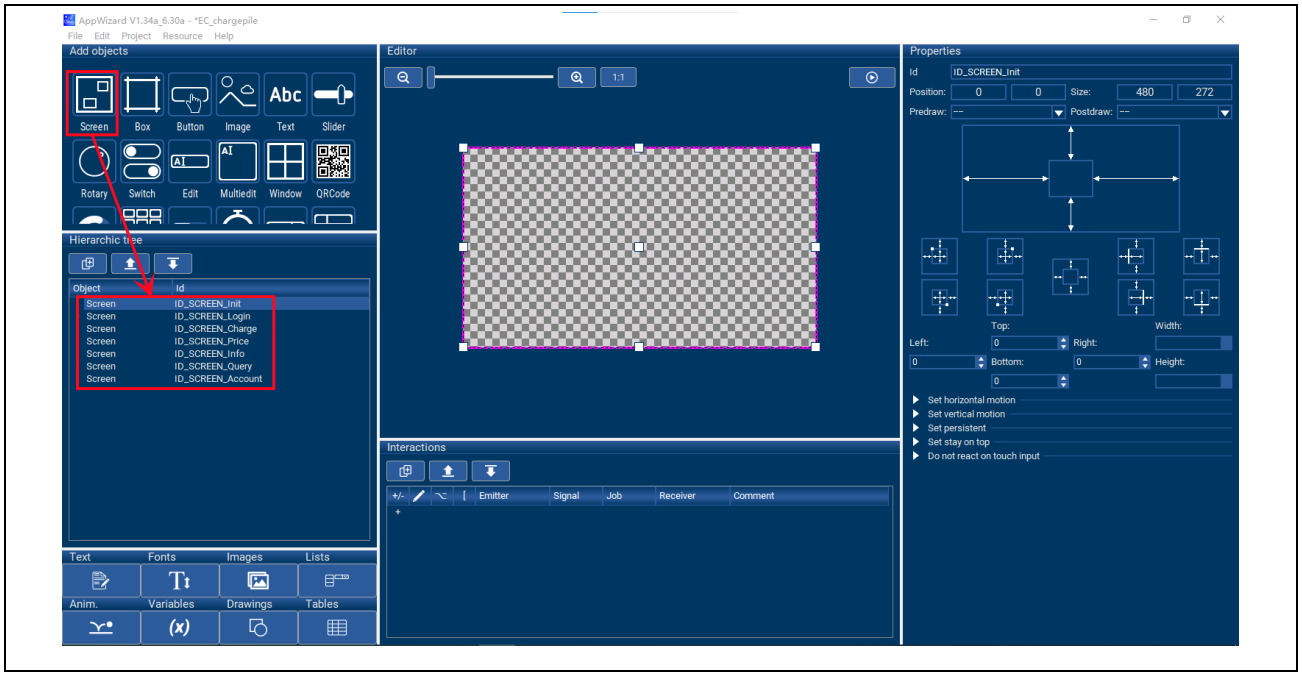


图 1-6 创建 7 个屏幕

2. Init 屏幕设计。包括背景、用户图案；用户名、实时时间显示；六个屏幕切换键及语言切换键。

   1) 点选 ID_SCREEN_Init 以编辑第一个画面，在“Add objects”下面点选“Image”，以创建一个背景，将名称改为 ID_IMAGE_Bg_Ini，然后在 Properties 中设置它的位置和尺寸属性。

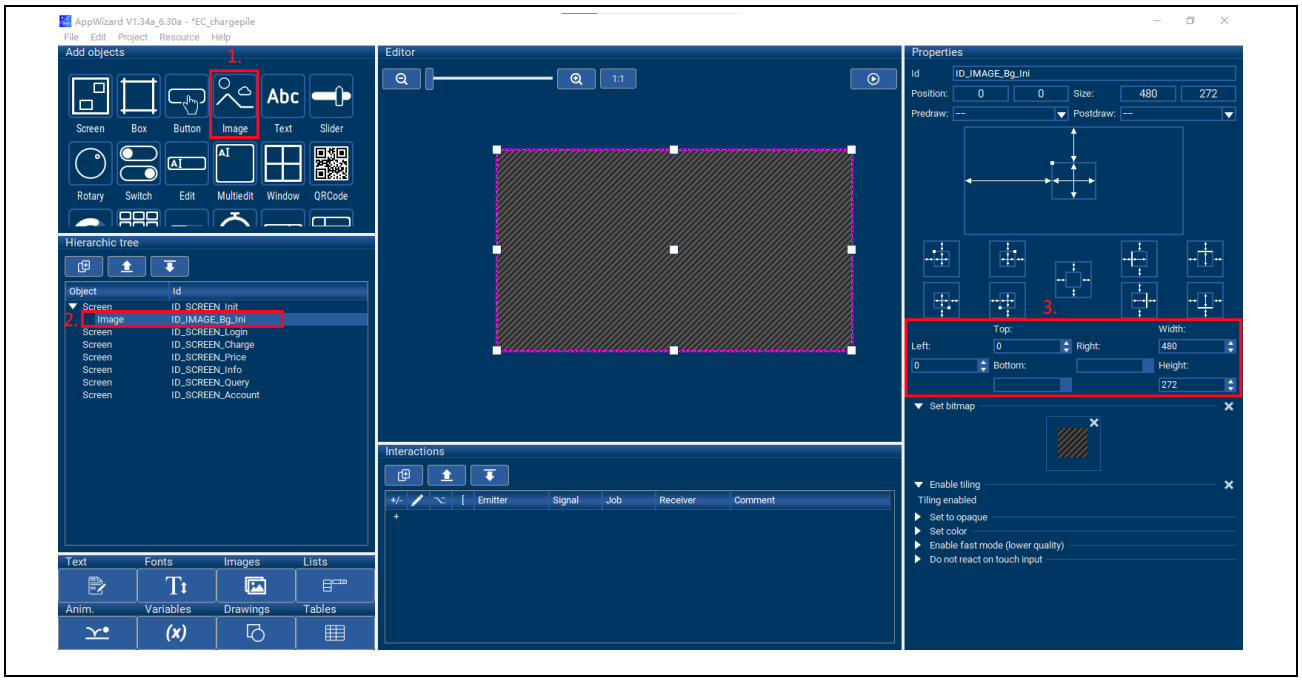


图 1-7 以 Image 创建背景及设置位置、尺寸属性

2) 在 Properties 中，为“Bg”设定 Bitmap 属性。

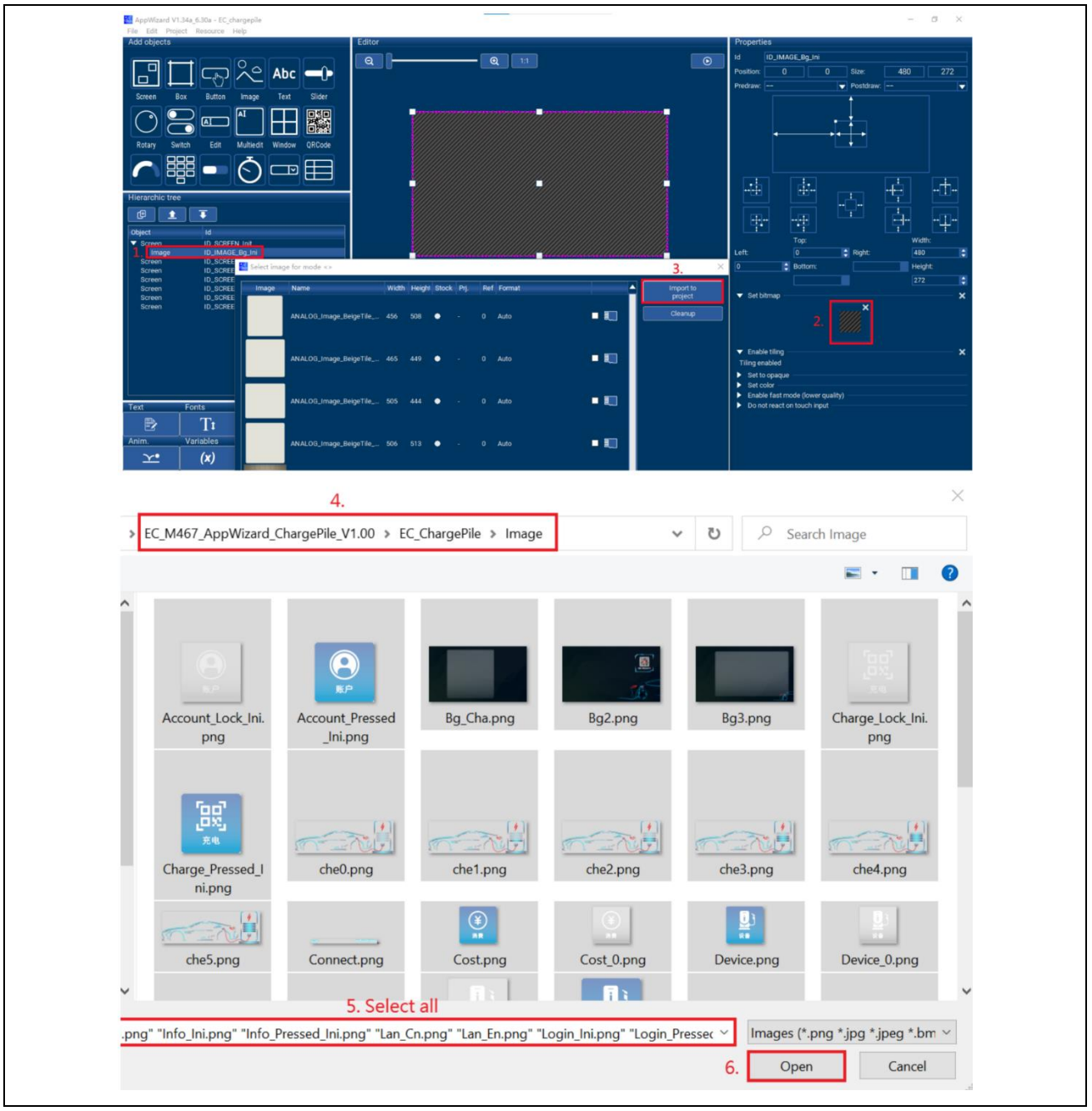


图 1-8 导入项目所需 icon、为“Bg” 设置 Bitmap 属性

3) 点选 ID_SCREEN_Init，为制作功能切换键，在 “Add objects” 下七次点选 “Button”，并于 “Hierarchic tree” 中分别重新命名为 ID_BUTTON_Login_Ini 、ID_BUTTON_Charge_Ini 、ID_BUTTON_Acc_Ini 、ID_BUTTON_Query_Ini 、ID_BUTTON_Price_Ini 、ID_BUTTON_Info_Ini 、ID_BUTTON_Lan_Ini 以利辨认，并对他们的位置、尺寸属性进行相关设置。

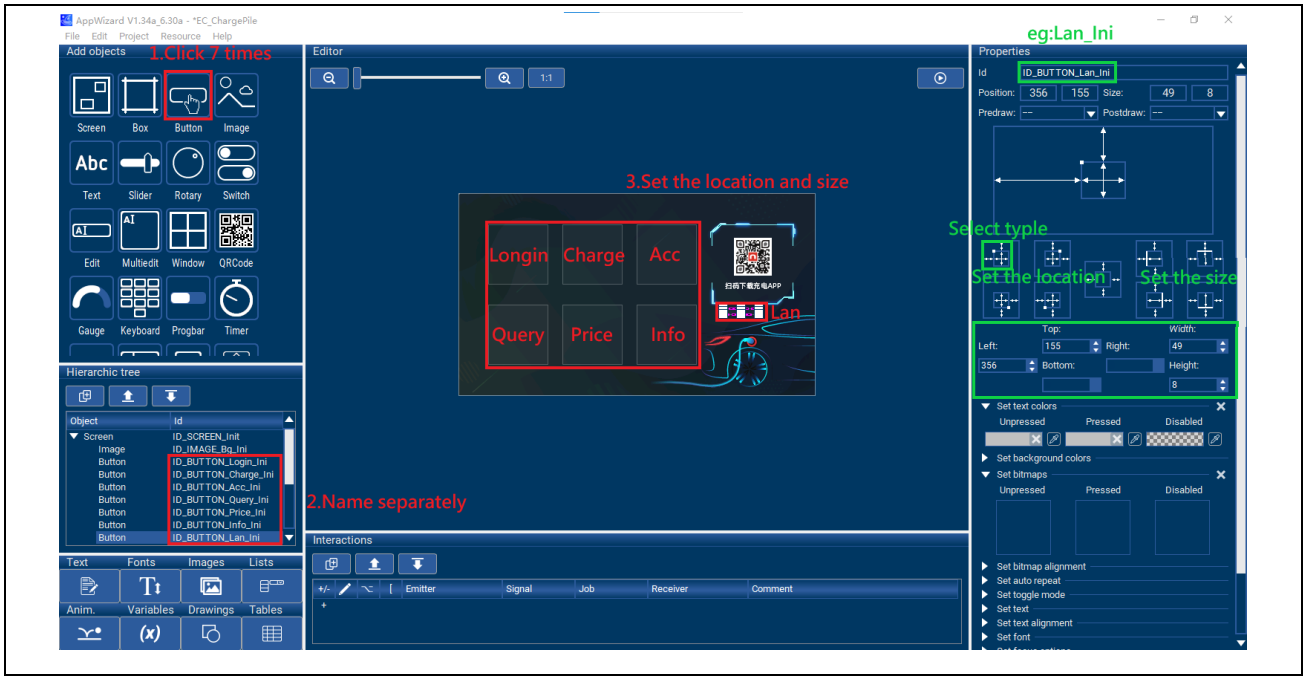


图 1-9 以 Button 创建屏幕切换键

4) 在 Properties 中，为各 Button 设定 Bitmaps 属性（参考 1.2.2 章 2 节 2 步骤，忽略其中第 3~6 子步骤）。

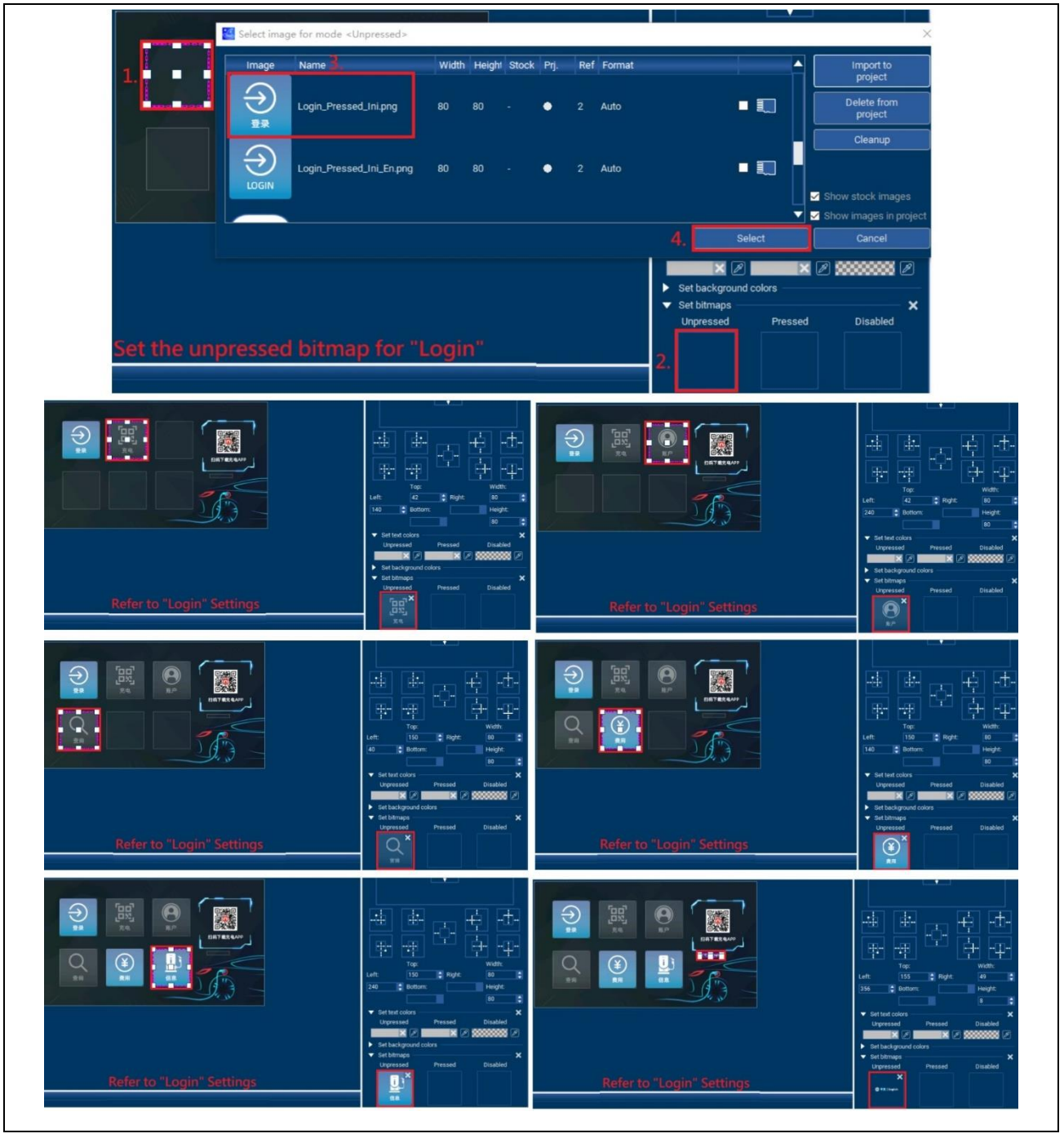
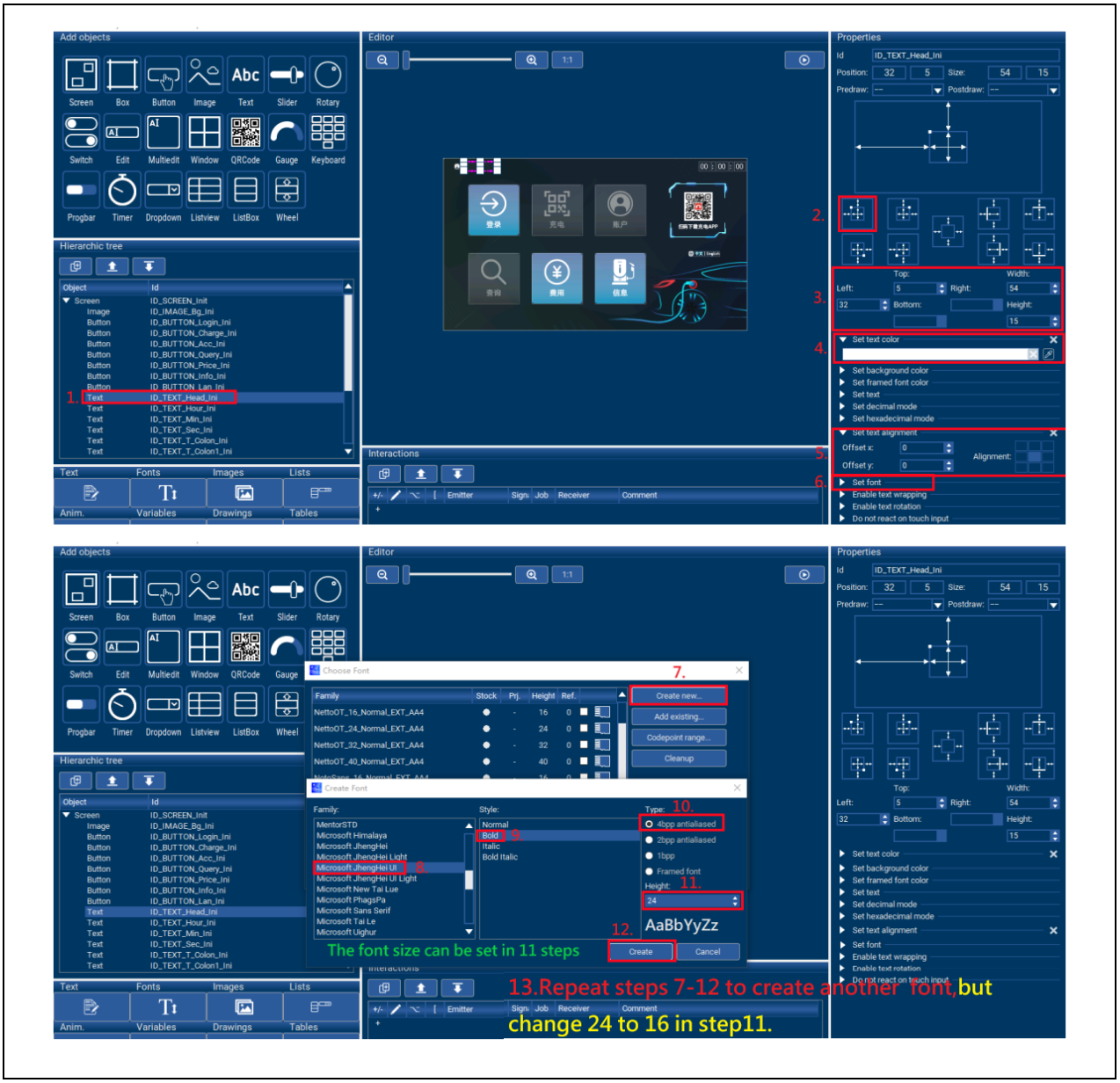


图 1-10 设定各 Button Bitmaps 属性

5) 点选 ID_SCREEN_Init，为制作显示文本，在 “Add objects” 中六次点选 “Text”，并于 “Hierarchic tree” 中分别重新命名为 ID_TEXT_Head_Ini 、ID_TEXT_Hour_Ini 、ID_TEXT_Min_Ini 、 ID_TEXT_Sec_Ini 、 ID_TEXT_T_Colon_Ini 、ID_TEXT_T_Colon1_Ini 以利辨认。

6) 设置 ID_TEXT_Head_Ini 物件的位置、尺寸、颜色、对齐模式、字体，用于显示文本。注意子步骤 7~13 是创建 2 种字体。



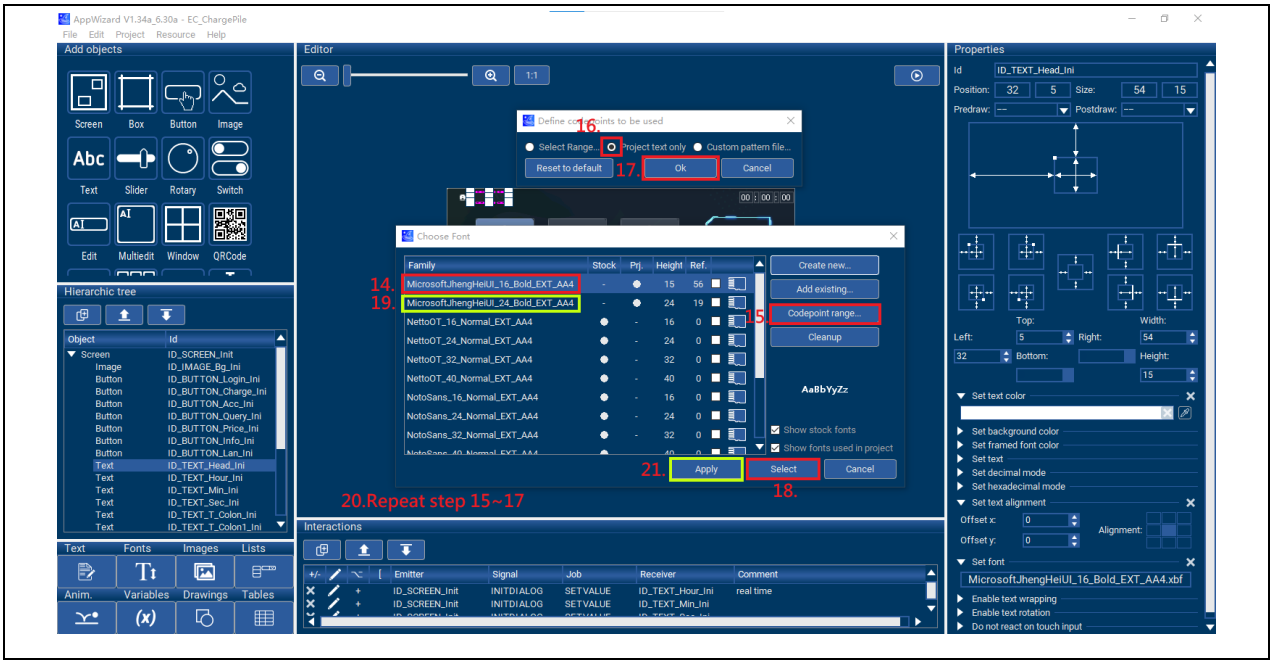


图 1-11 设置 Text 以显示文本的相关属性

7) 设置 ID_TEXT_Hour_Ini 相关属性以显示 2 位数值 (ID_TEXT_Min_Ini、ID_TEXT_Sec_Ini 的设置和 ID_TEXT_Hour_Ini 类似)

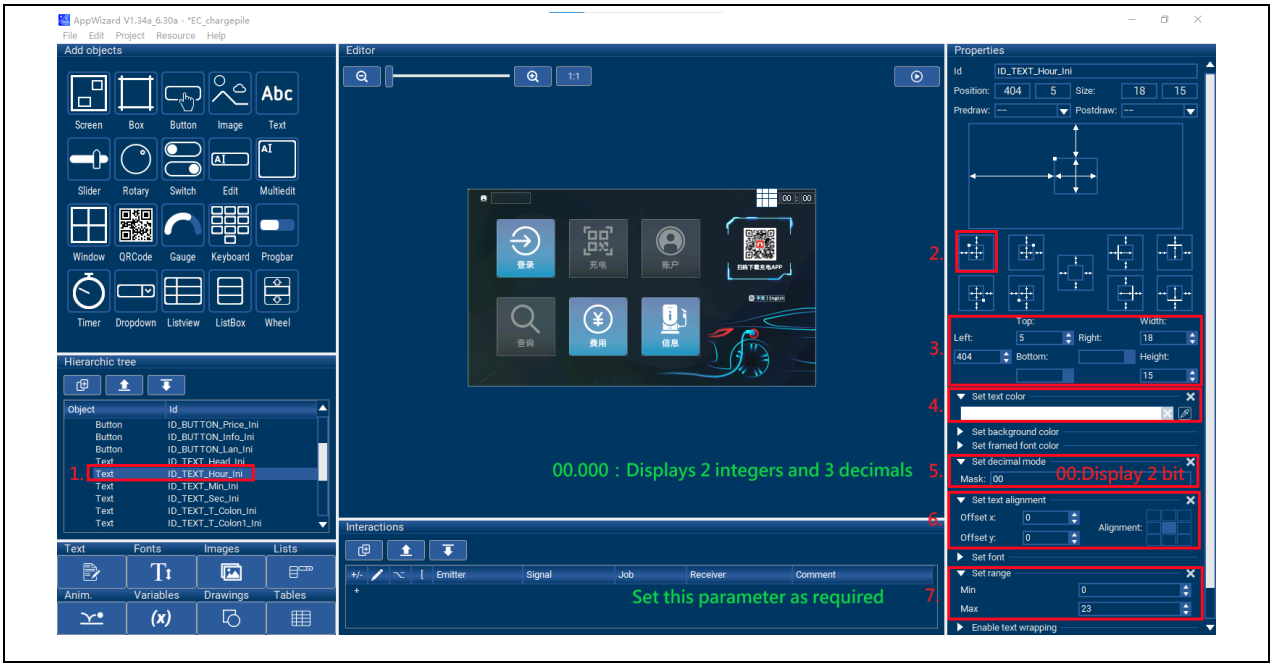


图 1-12 设置 Text 以显示数值的相关属性

8) 设置 ID_TEXT_T_Colon_Ini 以显示中英文时间分隔符(ID_TEXT_T_Colon1_Ini 的设置与 ID_TEXT_T_Colon_Ini 类似)。注意子步骤 7~12 以创建中英文 2 种语言。

图 1-13 设置 Text 以显示中英文本相关属性

9) 点选 ID_SCREEN_Init 以编辑第一个画面，于 “Add objects”中点选”Image”，以创建用户图案，将名称改为 ID_IMAGE_User_Ini，并设置位置、尺寸、Bitmaps 属性。

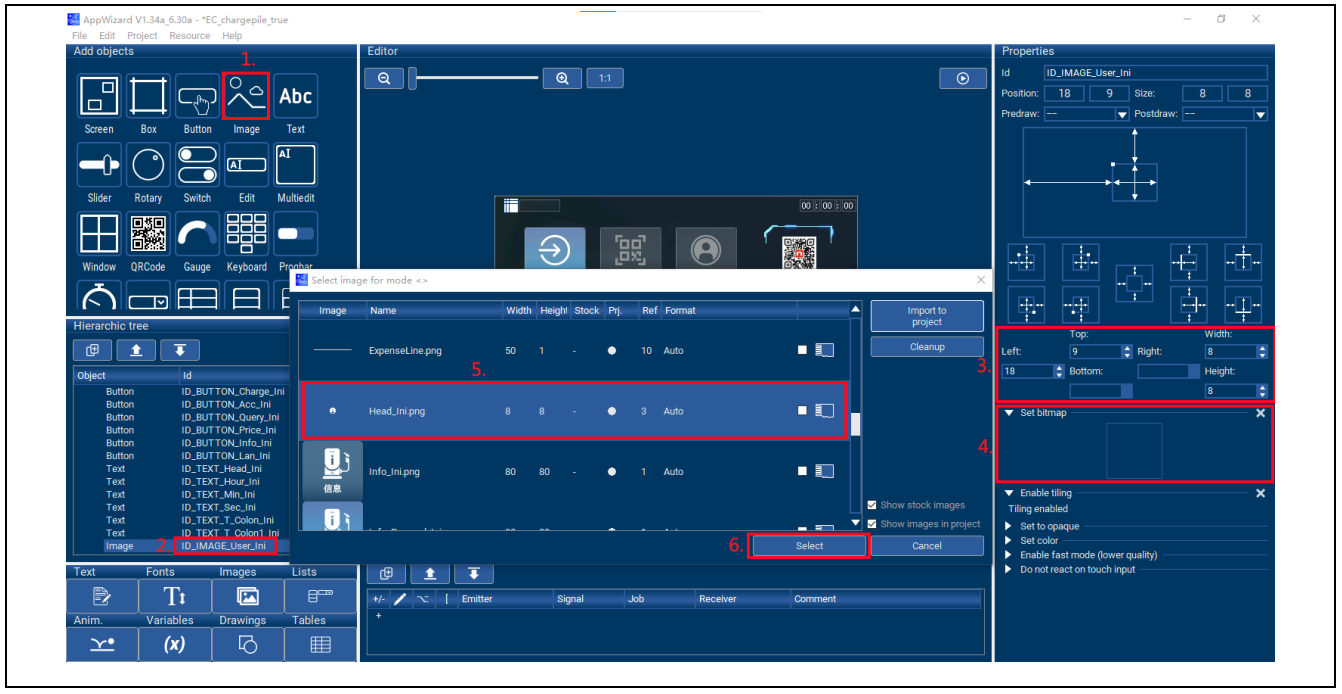


图 1-14 设置“User”位置、尺寸及 Bitmaps 属性

10) 点选 ID_SCREEN_Init 以编辑第一个画面，在 “Add objects”中点选“Timer”，以创建一个定时器，将名称改为 ID_TIMER_1S_Ini。并设置定时周期及使能自动重启模式。

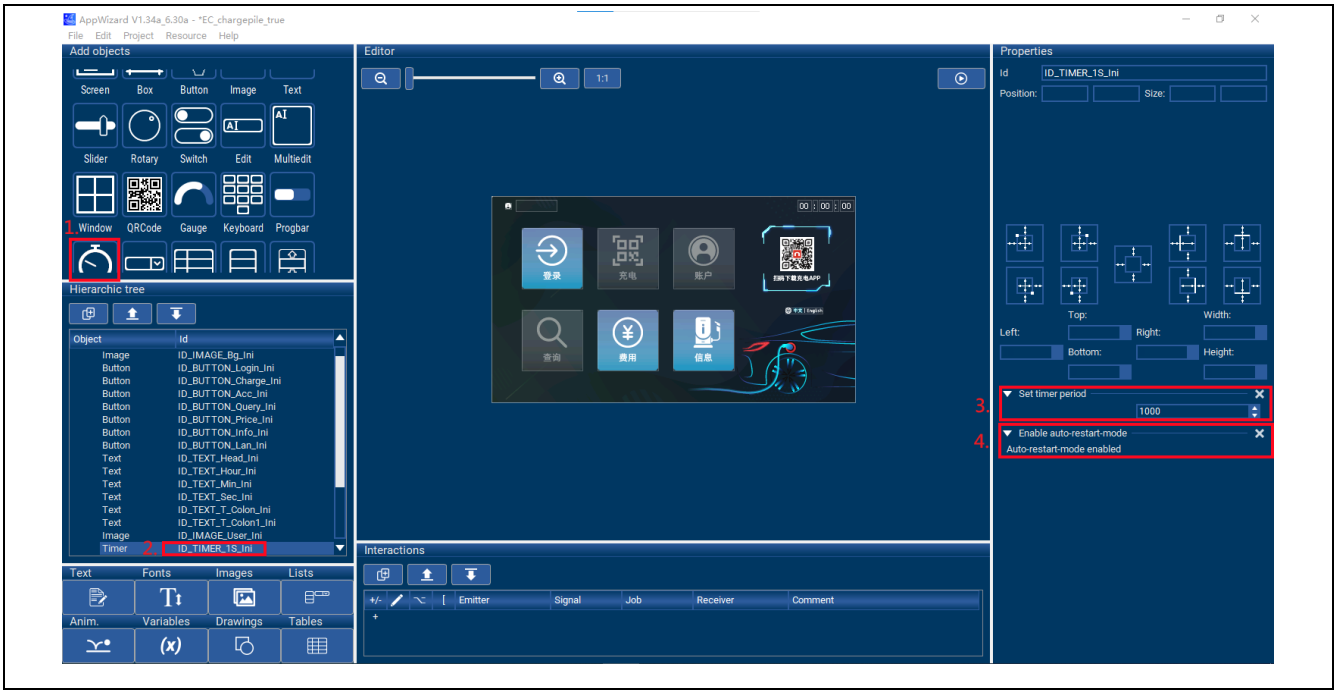


图 1-15 设置 Timer 相关属性

3. Charge 屏幕设计。包括主体部分、监控窗口、消费窗口、设计窗口。

1) Charge 屏幕主体。参考 Init 屏幕设计，添加各物件并设置他们的相关属性。包括 3 个切换键 1 个返回键及 1 个启动/停止充电键；实时时钟、充电时间、用户名显示；背景、用户图案加载；充电百分数及图形化显示。

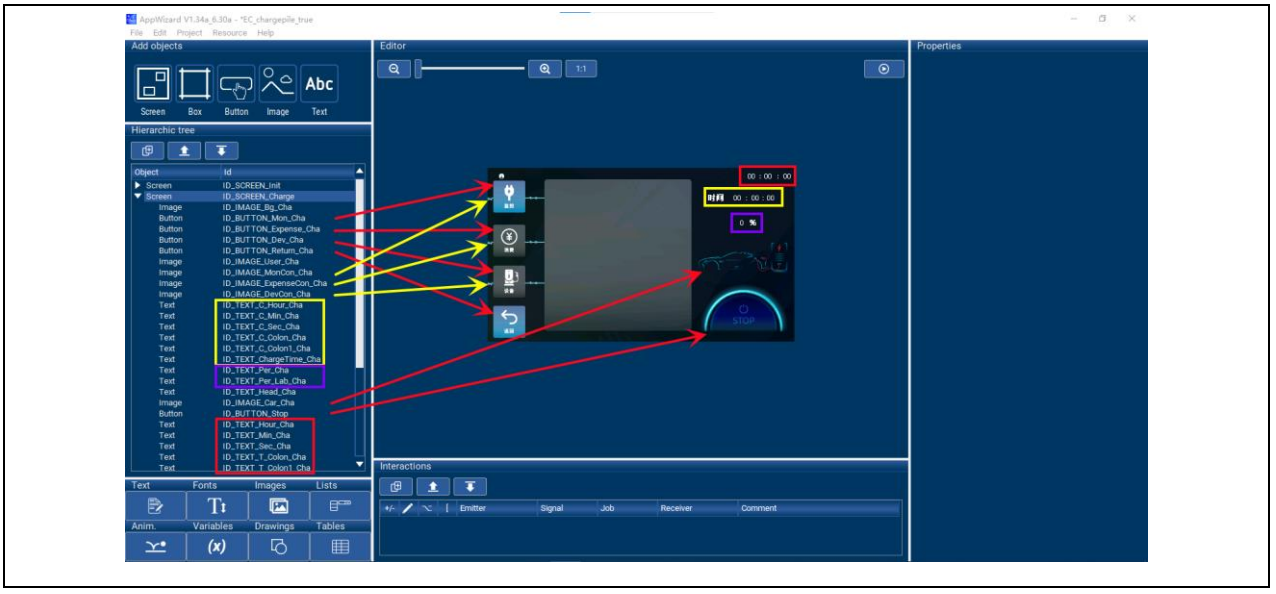


图 1-16 添加所有物件并设置它们的相关属性

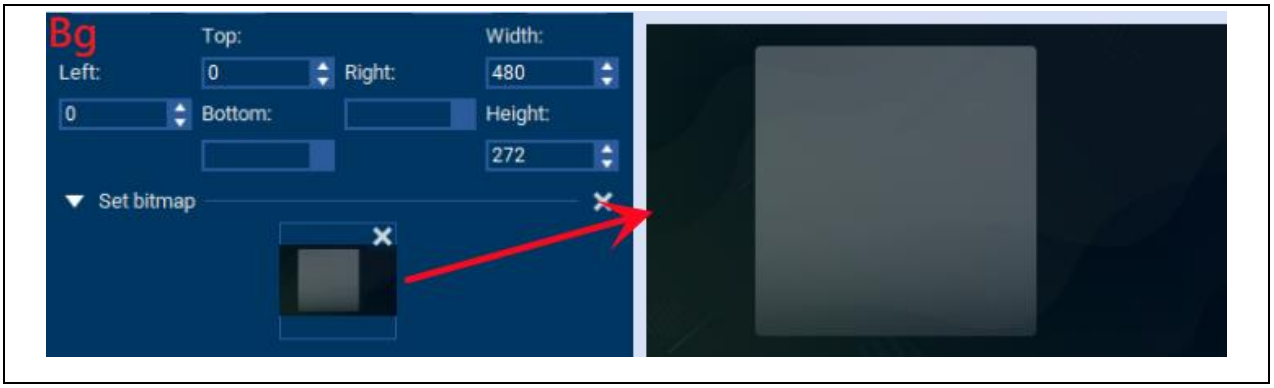


图 1-17 设置“Bg”的属性

图 1-18 设置“Mon”、“Dev”、“Expense”、“Return” 的相关属性

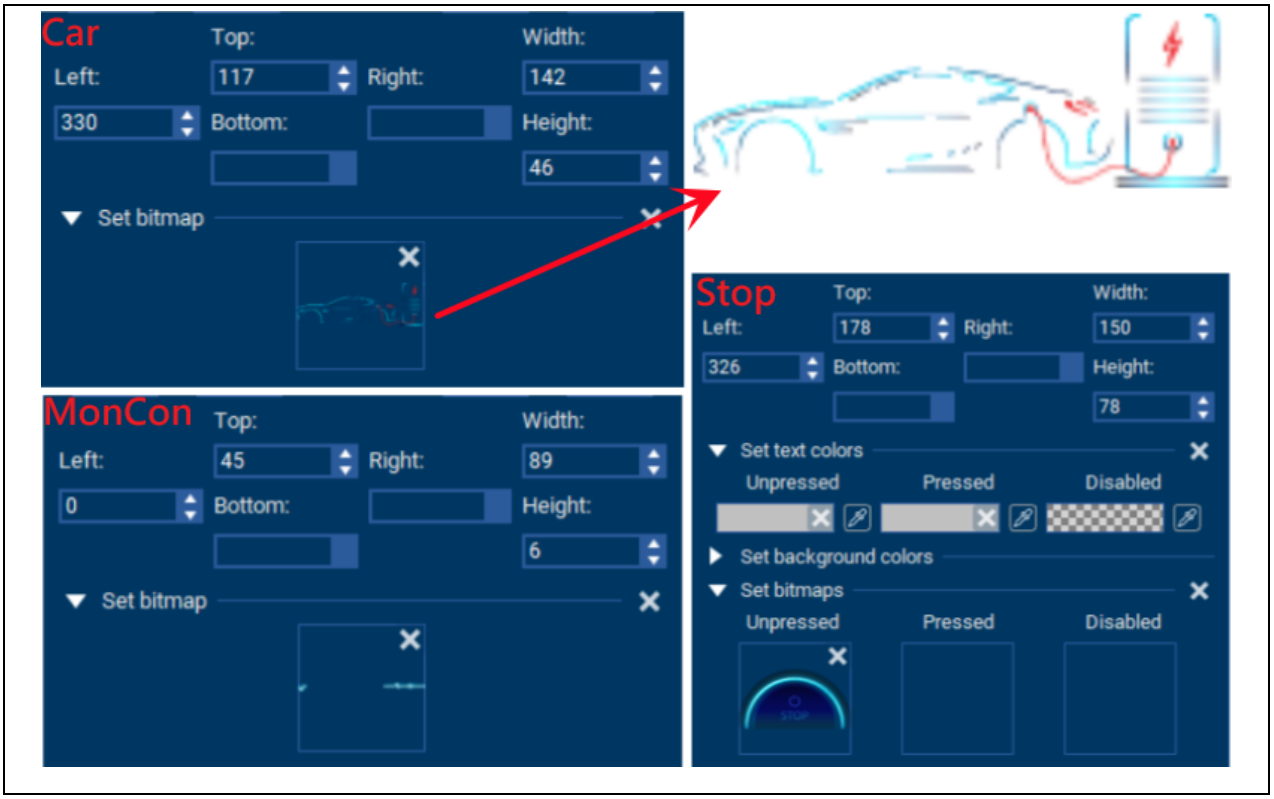


图 1-19 设置“Car”、“MonCon”、“Stop”的相关属性

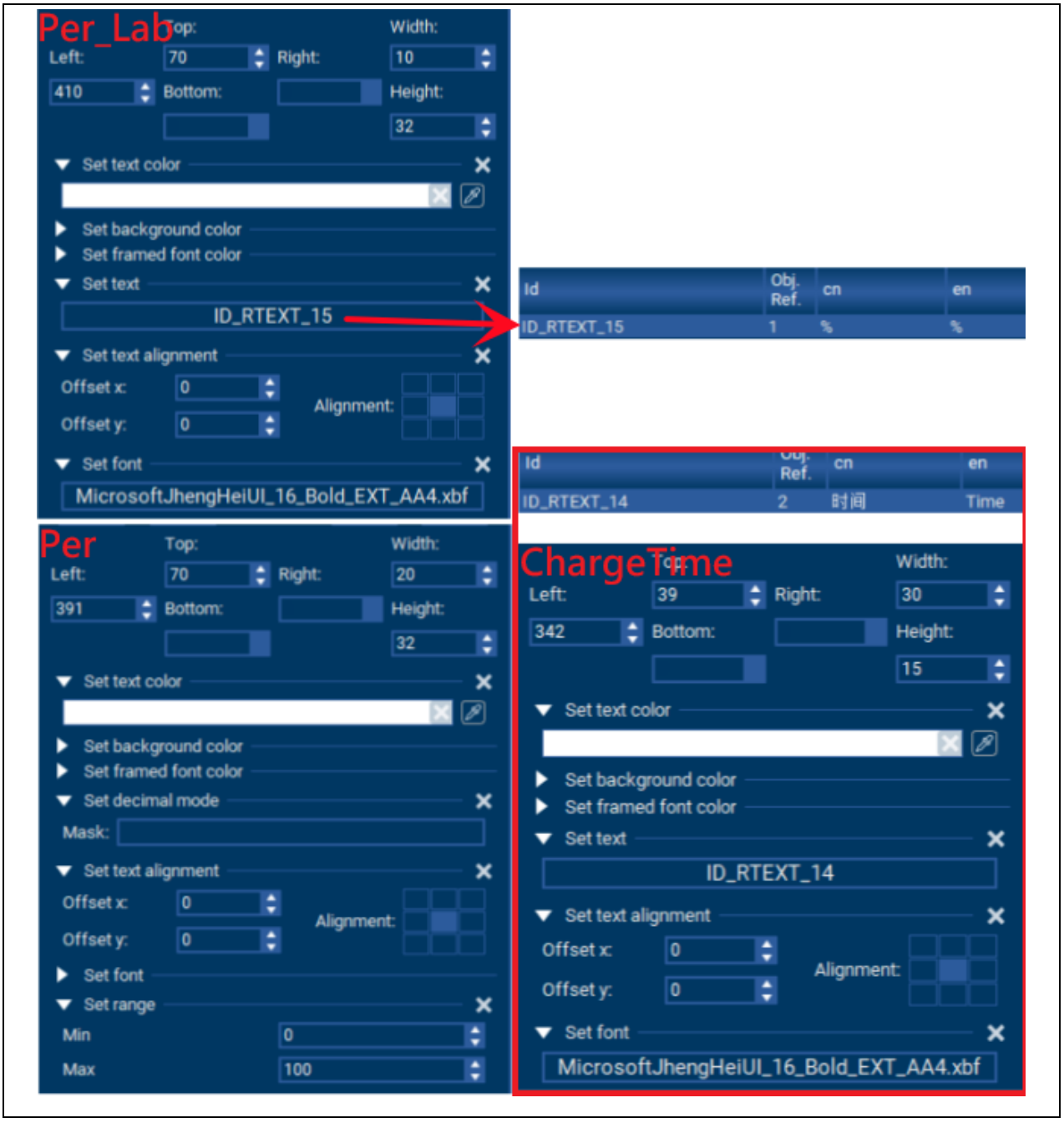


图 1-20 设置“Per_Lab”、“Per”、“ChargeTime” 相关属性

2) Charge 屏幕监控窗口。包括累计电量、累计消费显示和充电电流、充电电压显示。

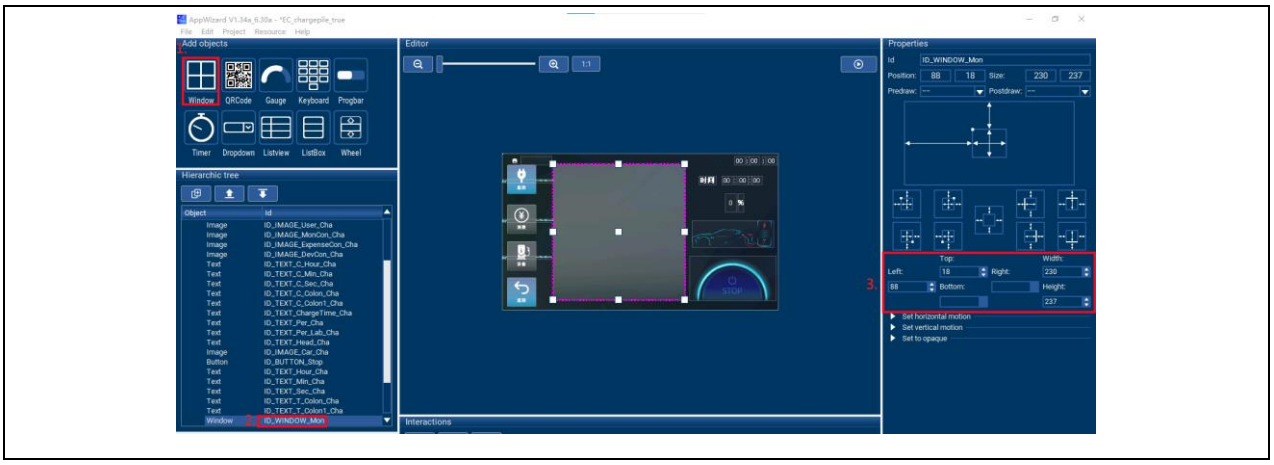


图 1-21 新建“Mon”窗口

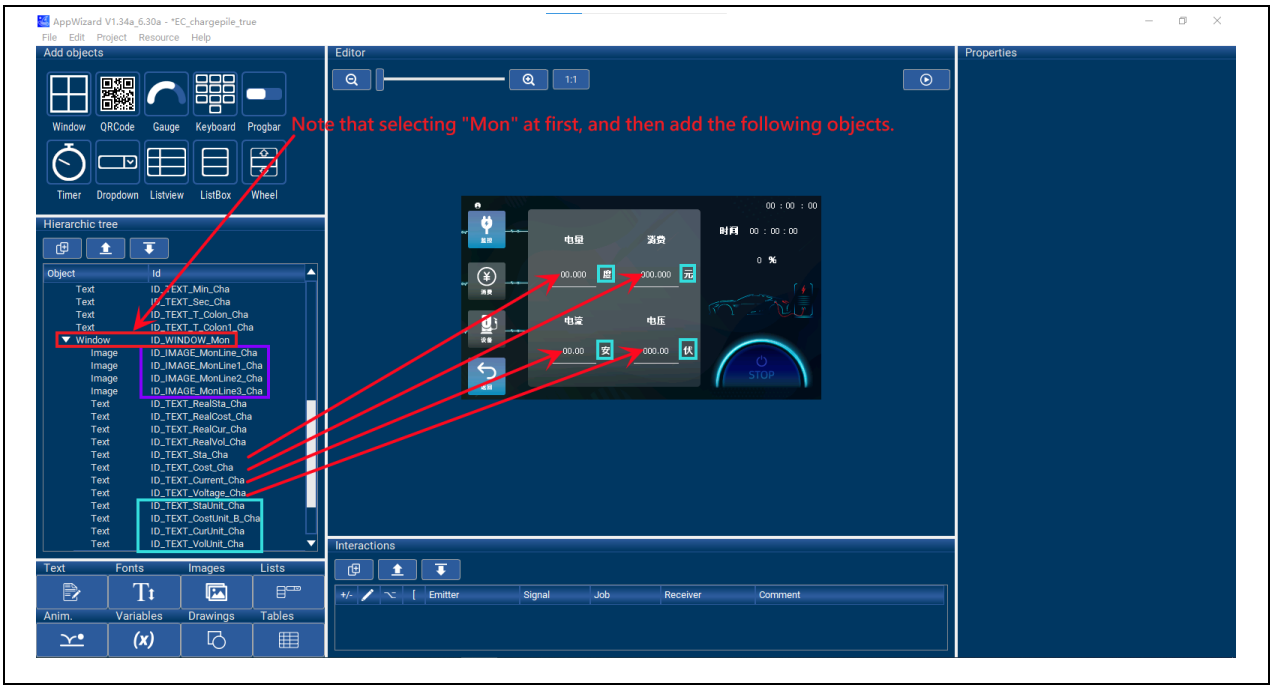


图 1-22 添加窗口所有物件并设置它们的相关属性

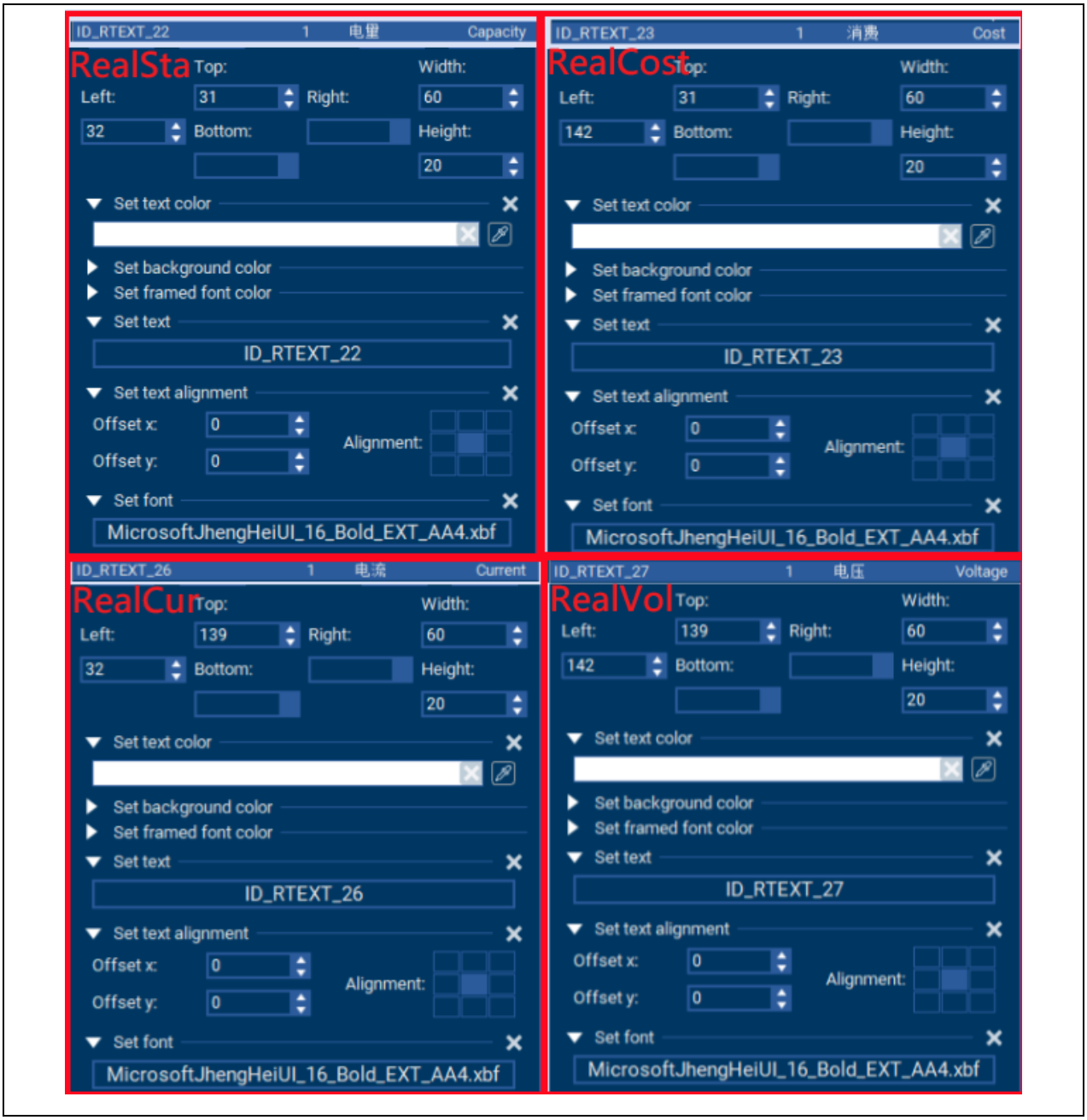


图 1-23 设置“RealSta”、 “Realcost”、 “RealCur”、 “RealVol”相关属性

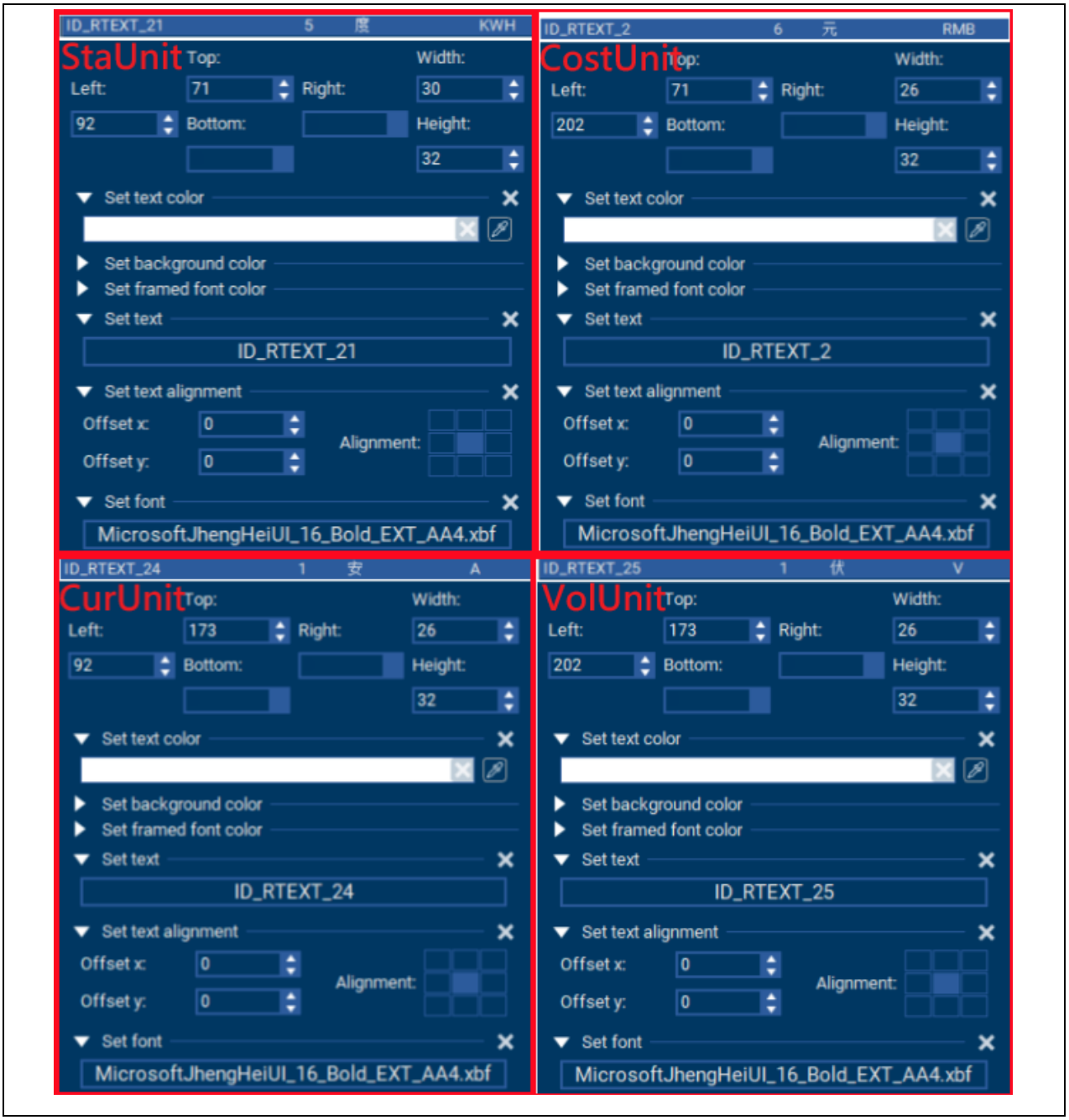


图 1-24 设置“StaUnit”、 “CostUnit”、 “CurUnit”、 “VolUnit”相关属性

图 1-25 设置“Sta”、 “Cost”、 “Current”、 “Voltage” 相关属性

3) Charge 屏幕消费窗口，参考 1.2.2 章 3 节 2 步骤。包括显示尖、峰、平、谷各时间段所消费的电量及费用。

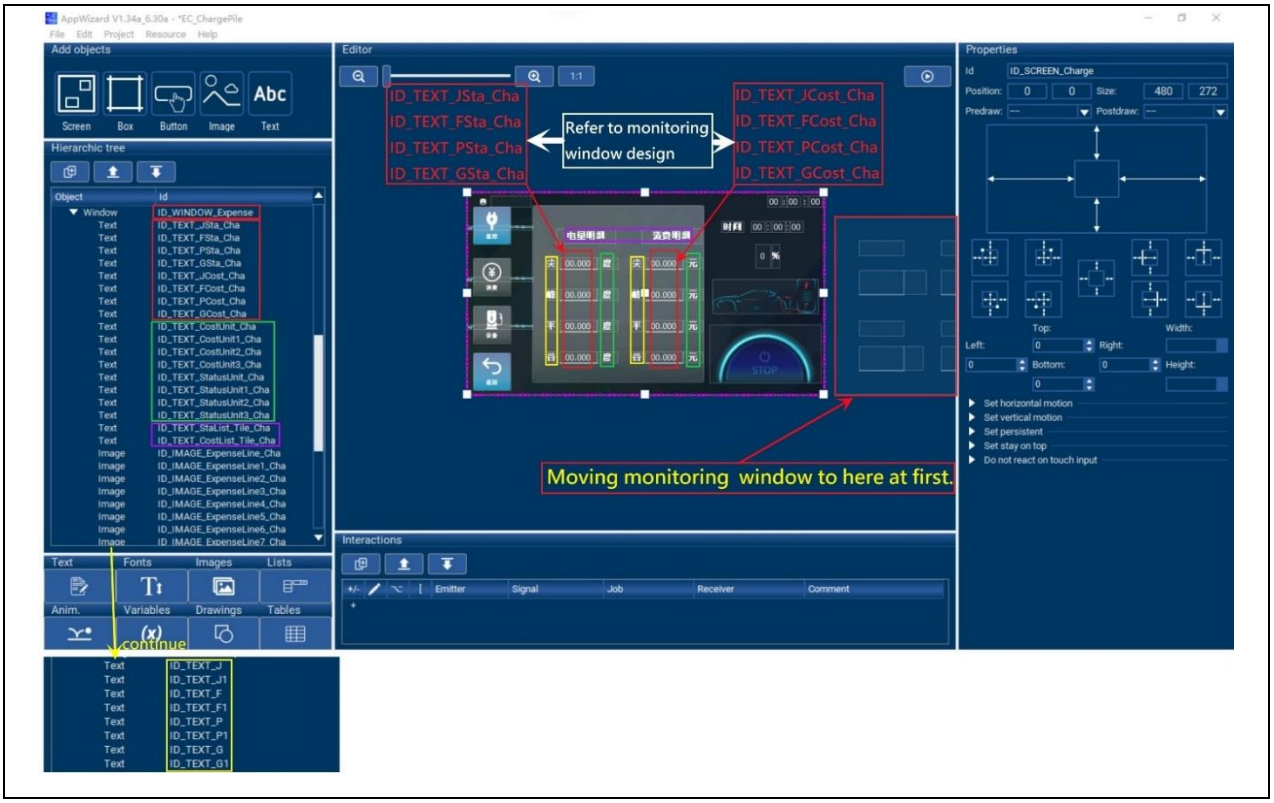


图 1-26 添加窗口所有物件并设置它们的相关属性

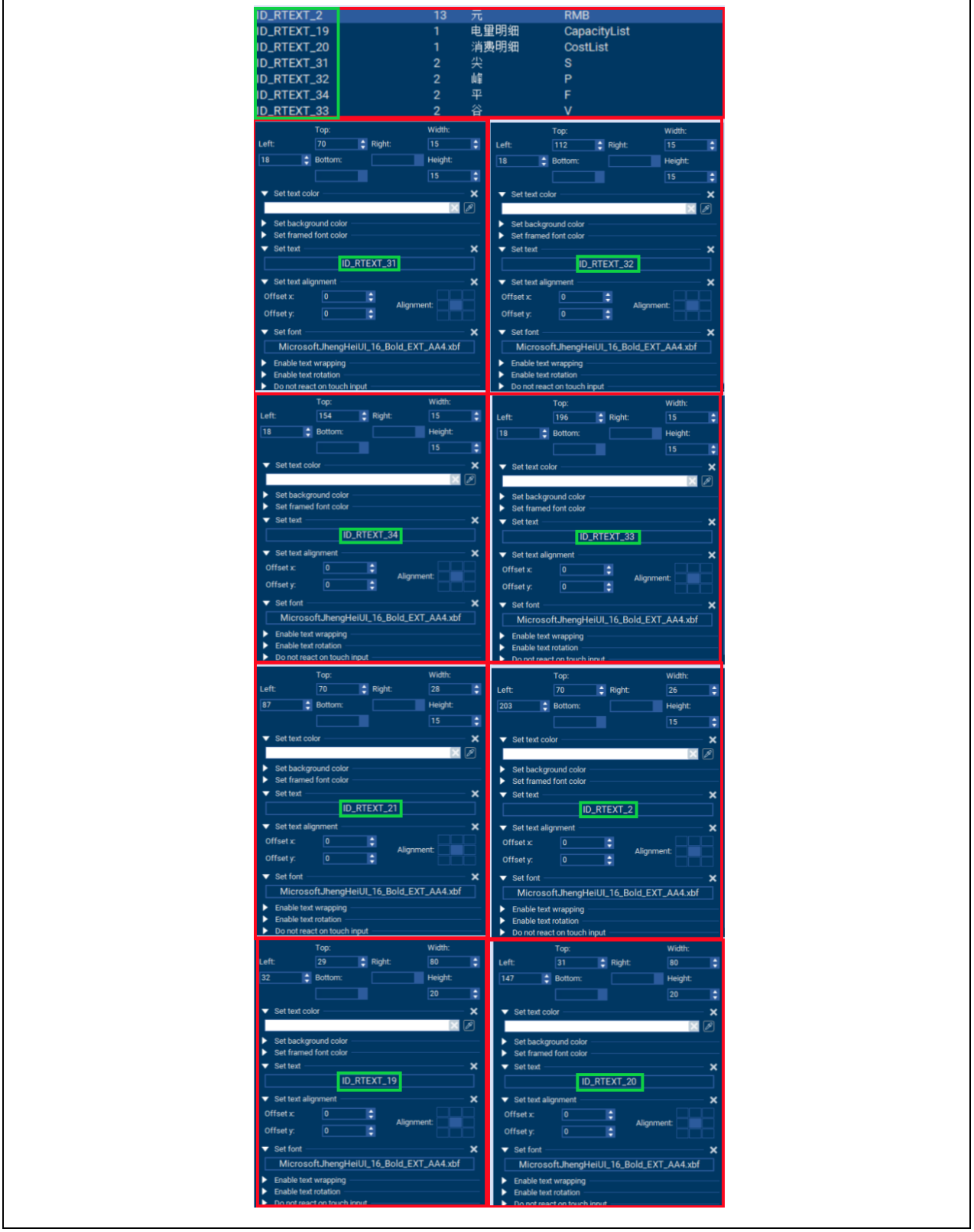


图 1-27 设置所有 Text 文本属性

4) Charge 屏幕设备窗口，参考 1.2.2 章 3 节 2 步骤。包括功率、当前价格显示。

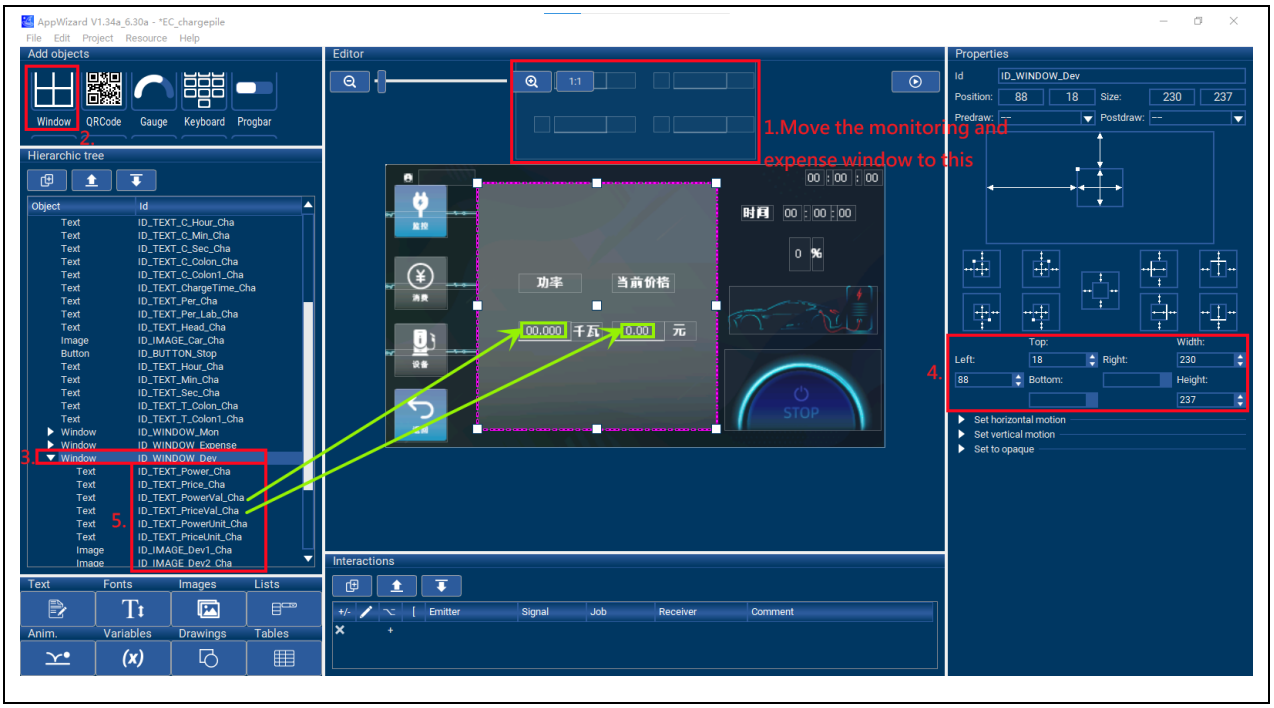


图 1-28 添加窗口所有物件并设置它们的相关属性

图 1-29 设置“Power”、 “Price”、 “PowerUnit”、 “PriceUnit”相关属性

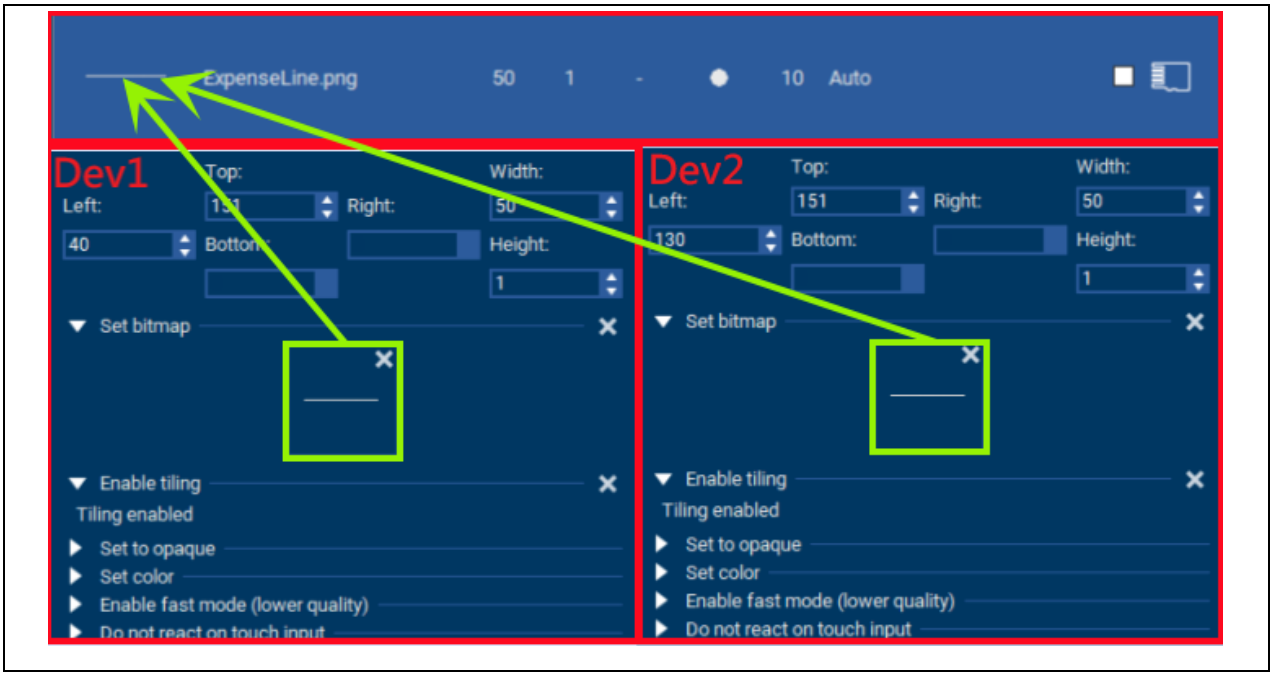


图 1-30 设置“Dev1”、 “Dev2”相关属性

5) 添加 2 个定时器。分别用于实时时钟显示和充电计时显示。

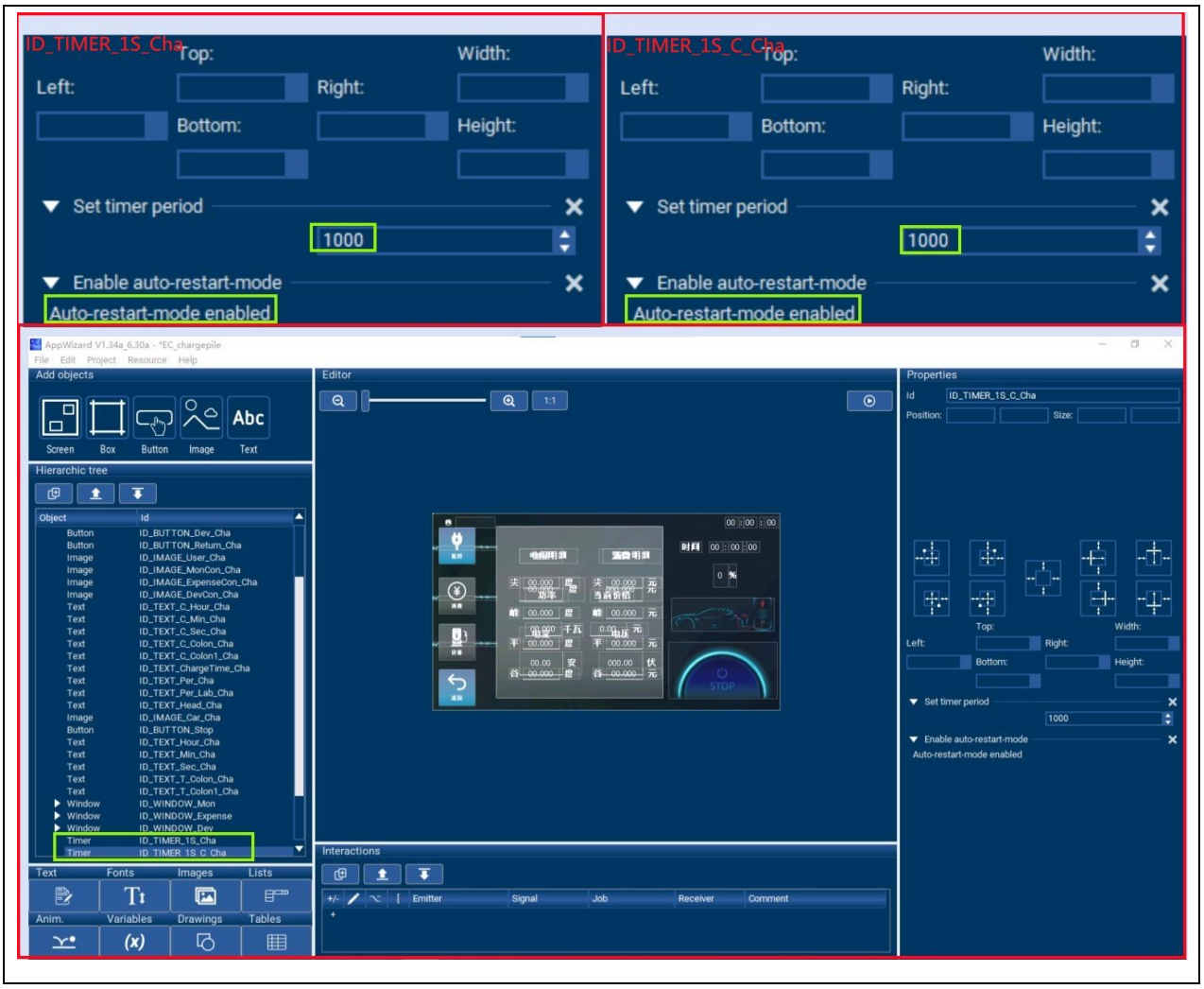


图 1-31 设置定时器相关属性

4. Price 屏幕设计。主要包括显示一天中 6 个时间段的价格。

1) 各时间段价格显示。参考 Init 屏幕设计，添加所有物件并设置它们的相关属性。

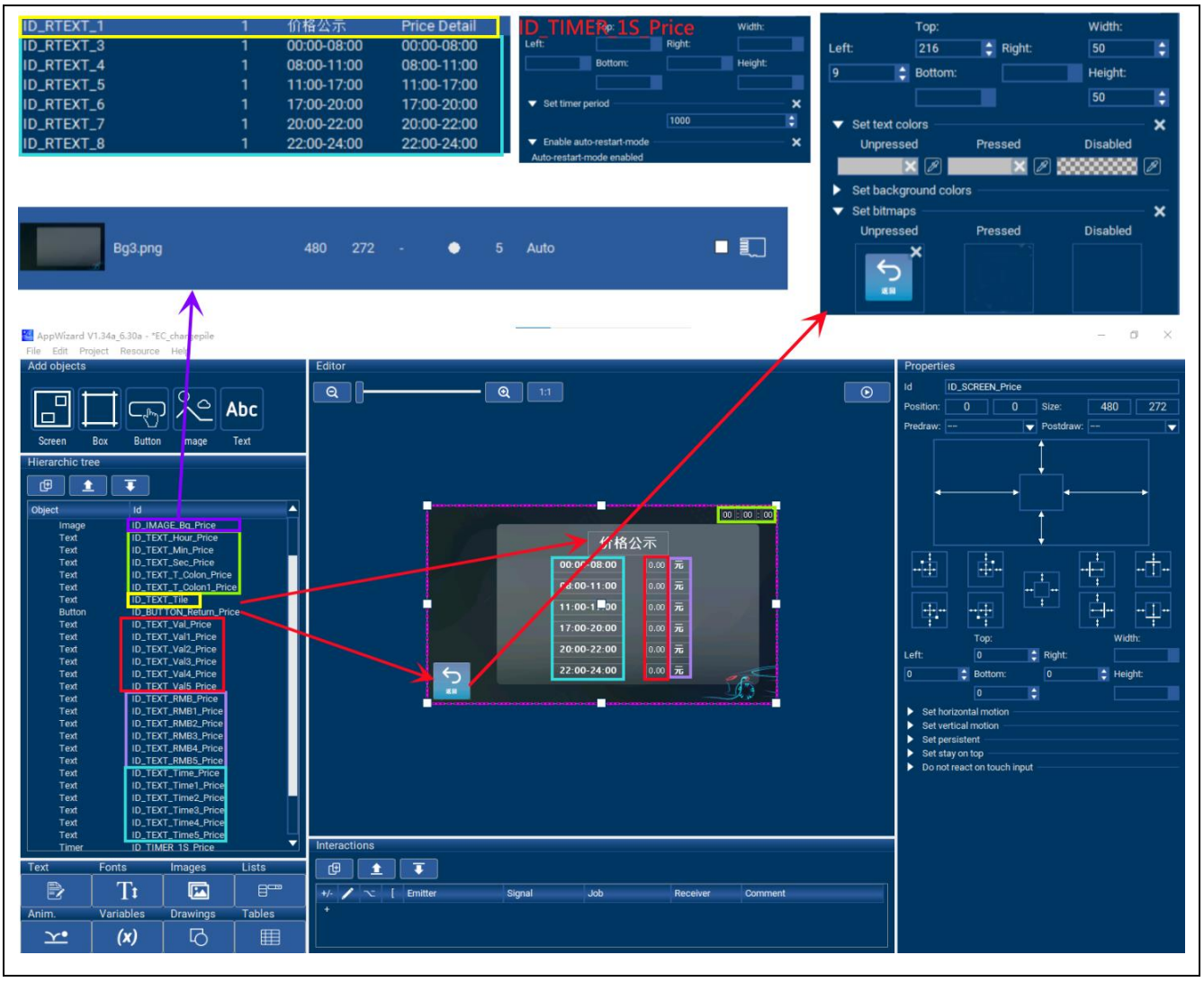


图 1-32 添加所有物件并设置他们的相关属性

5. Info 屏幕设计。主要包括显示当前充电站名称、充电类型及支持的充电枪号。

   1) 充电站、类型、枪号显示。参考 Init 屏幕设计，添加所有物件并设置它们的相关属性。

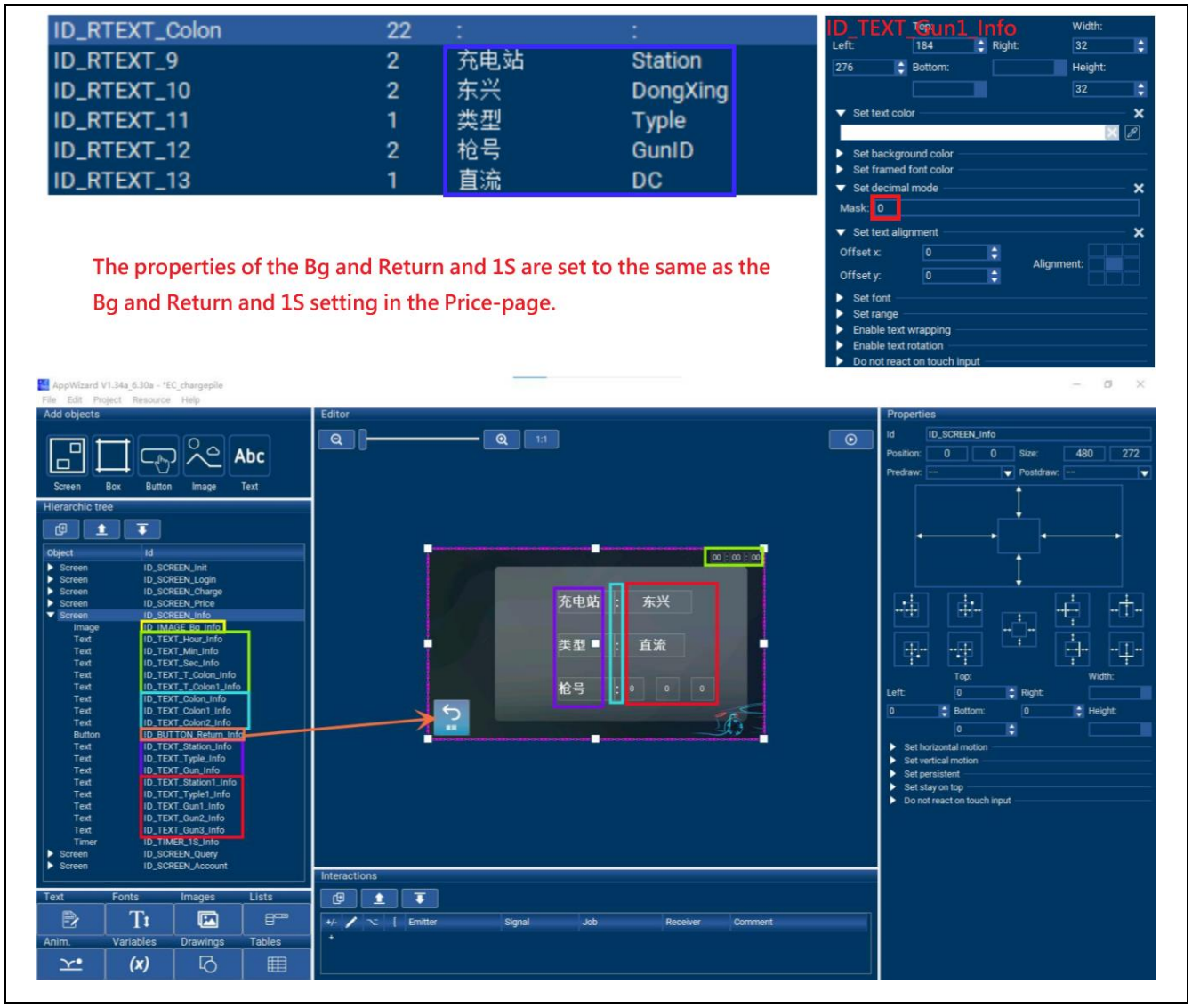


图 1-33 添加所有物件并设置它们的相关属性

6. Query 屏幕设计。主要包括最近一次充电时长、费用、充电站名称及充电枪号显示。

1) 充电信息显示。参考 Init 屏幕设计，添加所有物件并设置它们的相关属性。

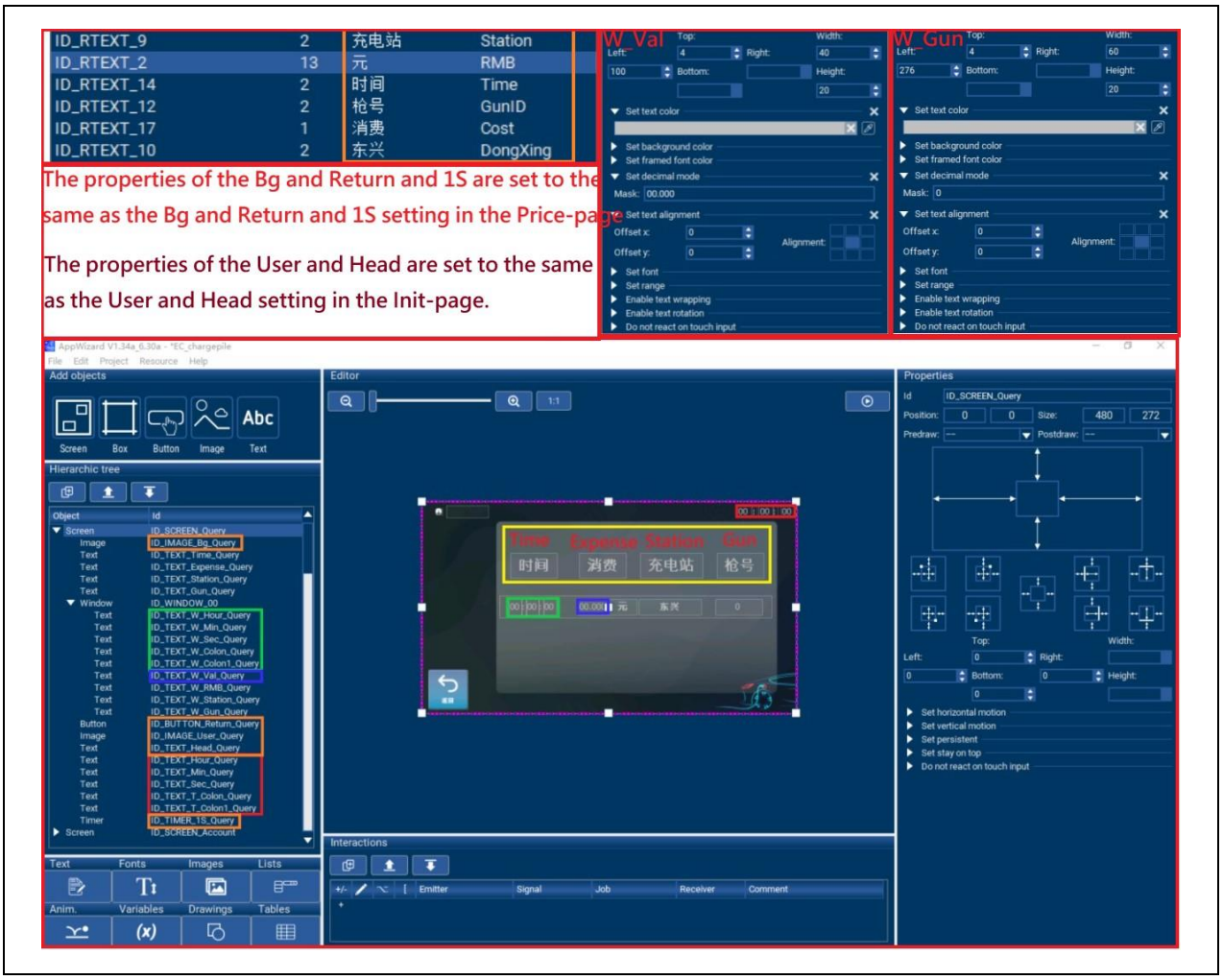


图 1-34 添加所有物件并设置它们的相关属性

7. Account 屏幕设计。主要包括显示账户姓名、余额及手机号码。

1) 账户姓名、余额、手机号码显示。参考 Init 屏幕设计，添加所有物件并设置它们的相关属性。

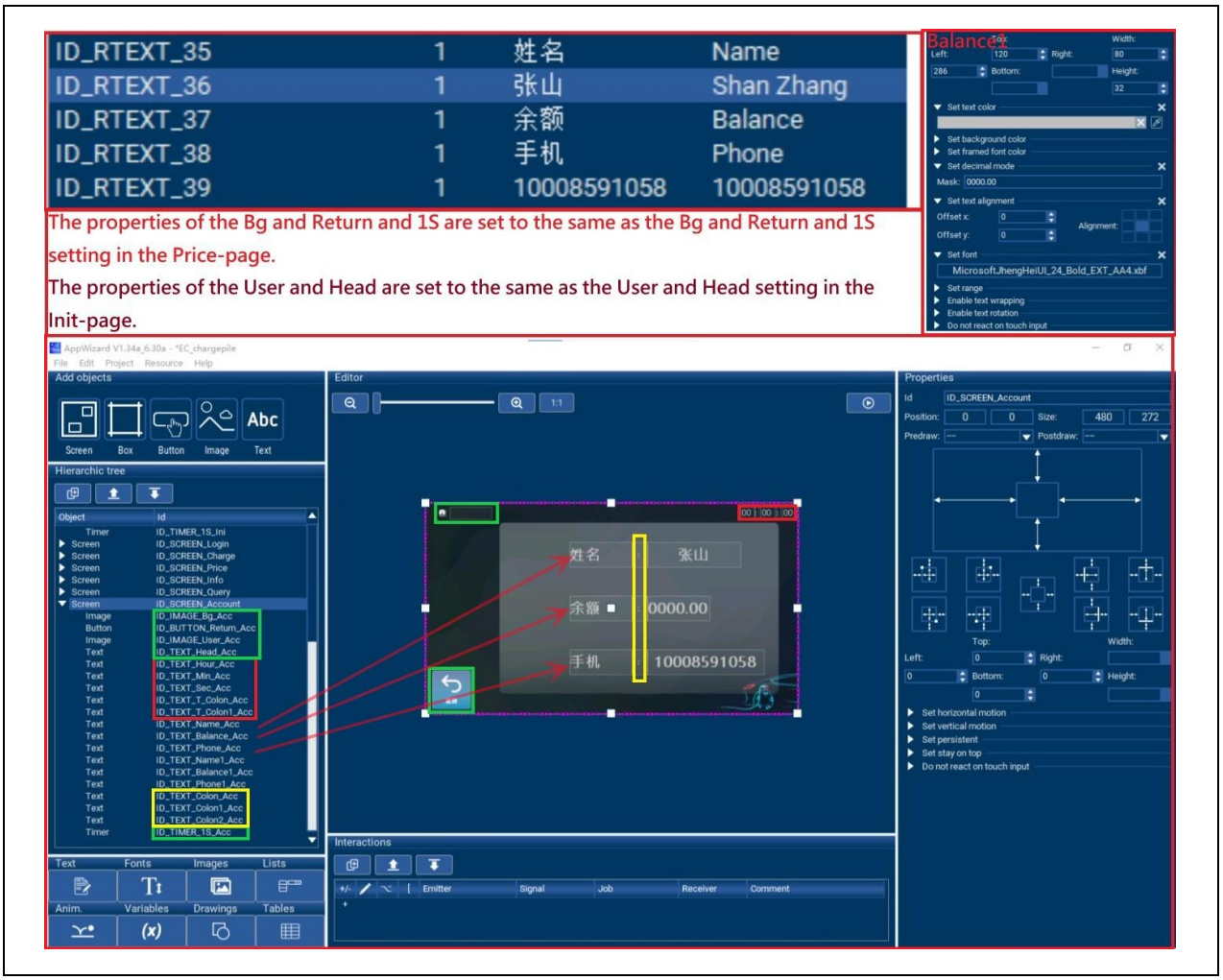


图 1-35 添加所有物件并设置它们的相关属性

8. Login 屏幕设计。主要包括接收用户名、密码输入及显示密码状态。

1) 用户名、密码输入及密码状态显示。参考 Init 屏幕设计，添加所有物件并设置它们的相关属性。

图 1-36 添加所有物件并设置它们的相关属性

1.2.3 在 AppWizard 工具中进行 Interactions 设定

1. Login 屏幕交互设计。包括按“Return”键返回 Init 屏幕；接收用户名、密码的输入；结合代码实时显示密码是否输入正确。

   1) 按“Return”键返回 Init 屏幕。

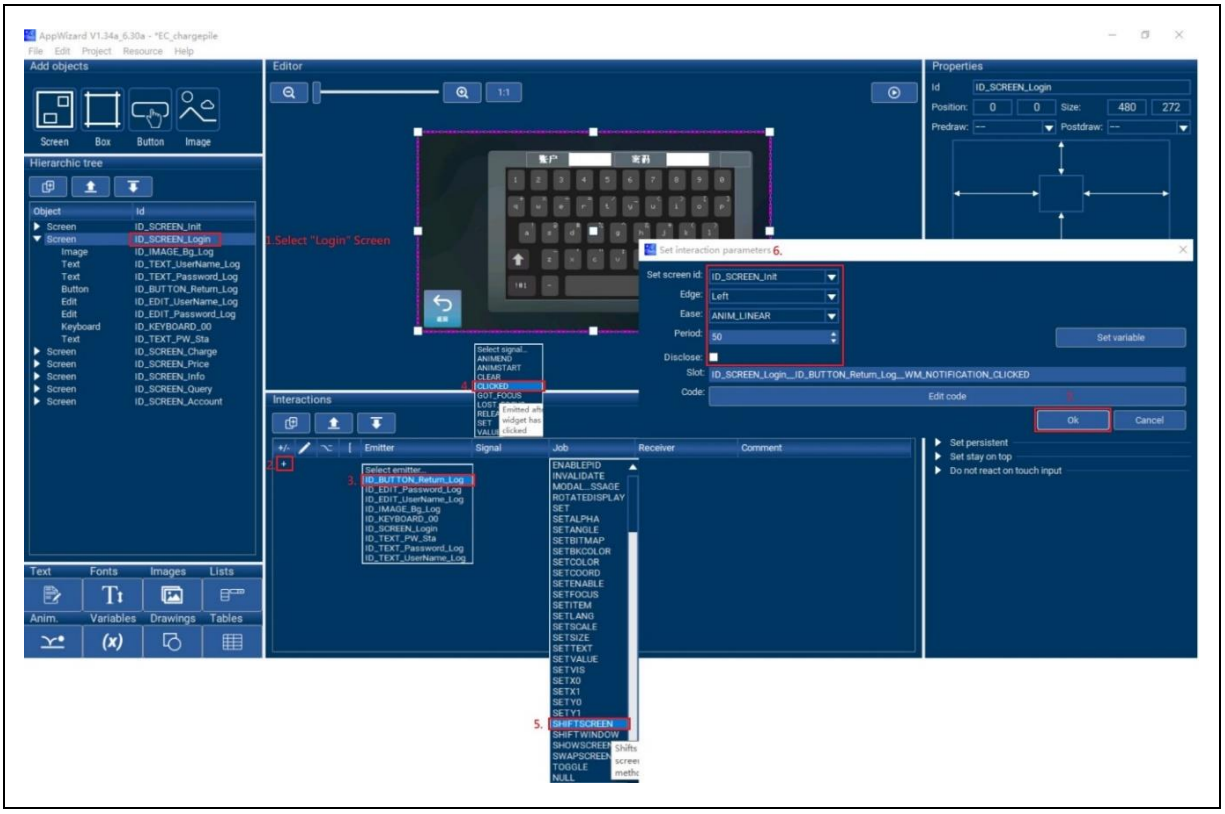


图 1-37 按“Return”键返回 Init 屏幕

2) 加载 Login 屏幕，聚焦“UserName”物件。

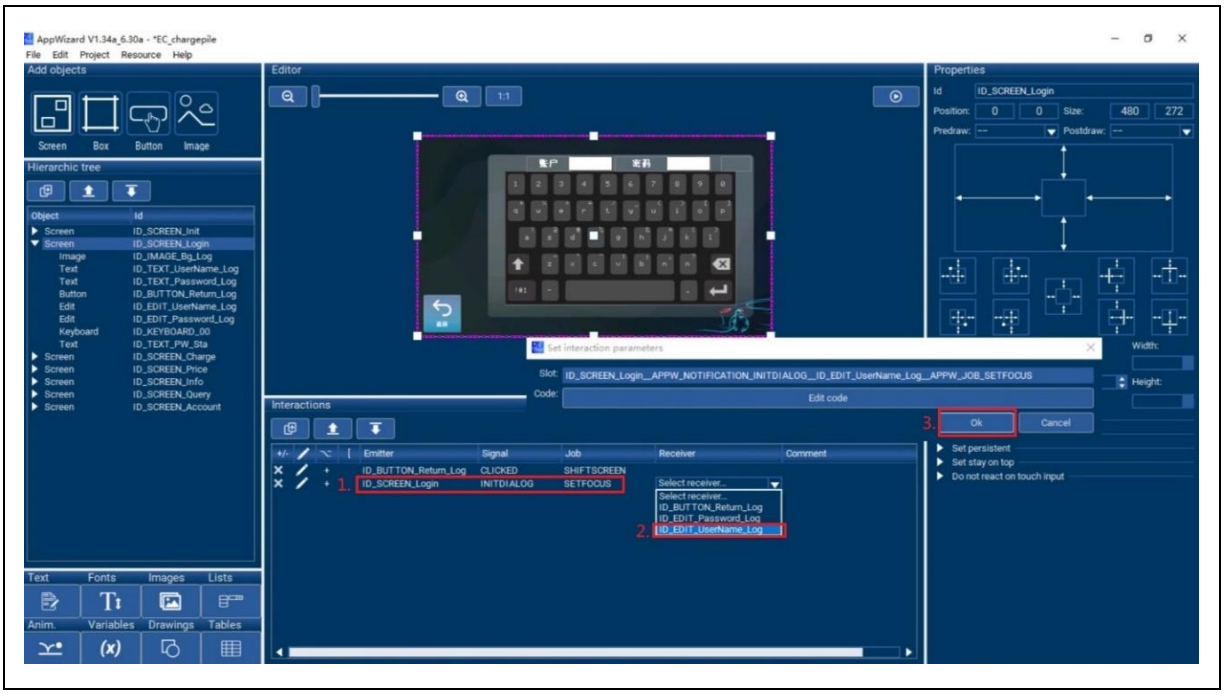


图 1-38 加载 Login 屏幕聚焦“UserName”物件

3) 加载 Login 屏幕，根据 “Lan”值的不同为 “Return”键设置不同的图案。

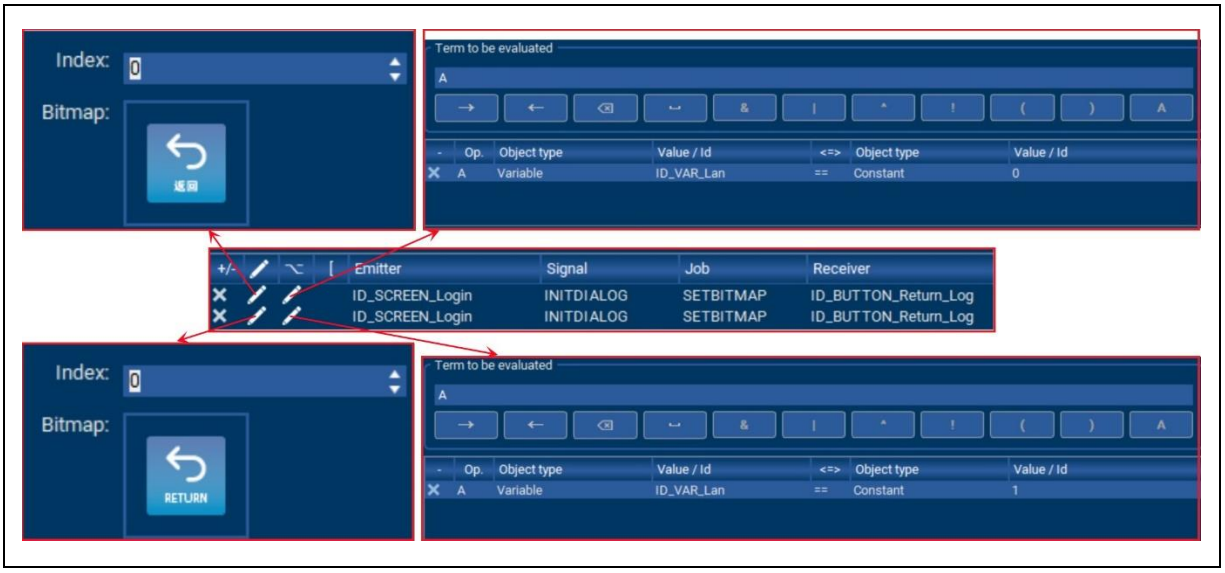


图 1-39 加载 Login 屏幕为 “Return”键设置不同的图案

4) 定义密码状态变量“PW_1”。当输入密码正确的时候由代码设置“PW_1”为 1，当输入密码错误的时候由代码设置“PW_1”为 0。默认正确密码为“ qqqqqqq”。

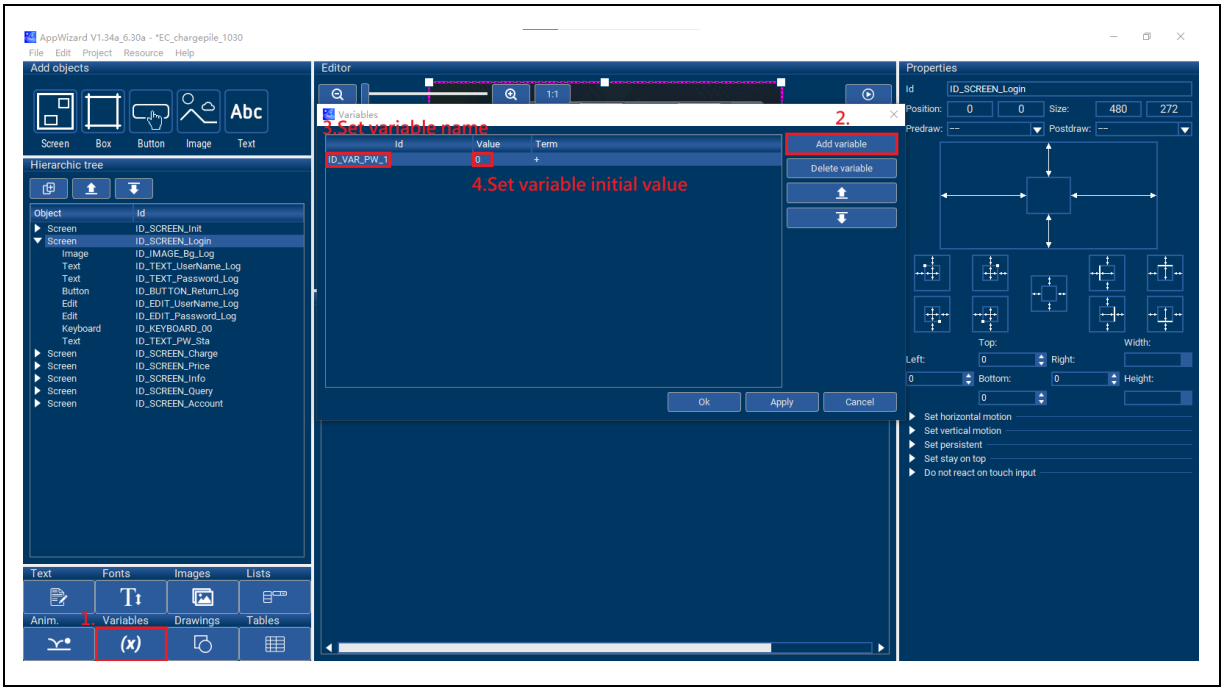


图 1-40 “PW_1” 变量定义

5) 用户名和密码处理逻辑。

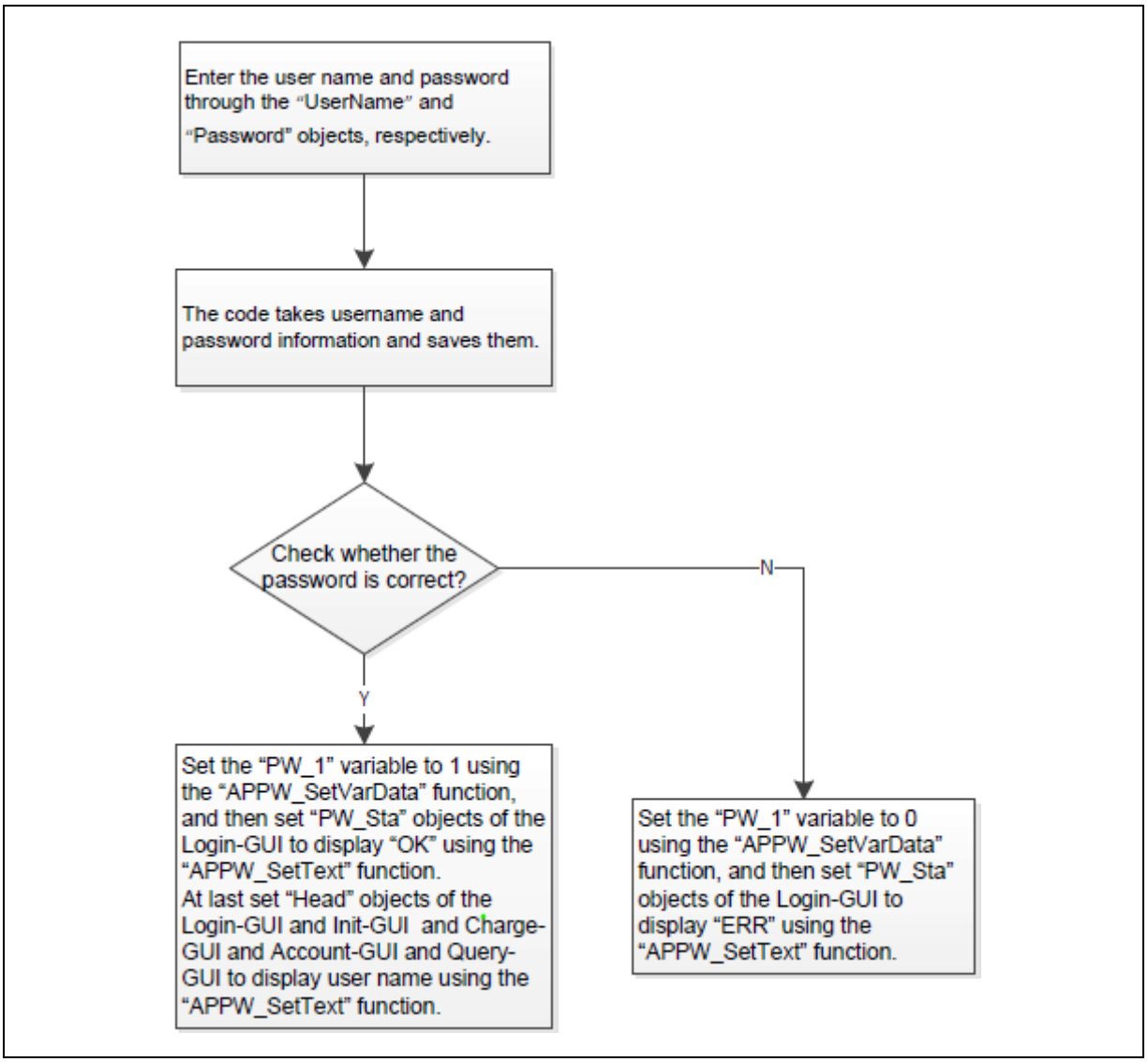


图 1-41 用户名和密码处理逻辑

2. Init 屏幕交互设计。包括实时时钟显示；结合代码进行用户名显示；按不同的键切换到不同的屏幕，按中英文切换键(“Lan”键)进行中英文切换；充电、账户、查询键图案根据“PW_1”变量的值进行不同的显示。

1) 定义实时时钟显示所需变量(“Hour”、“Min”、“Sec”、“SecCnt”)。

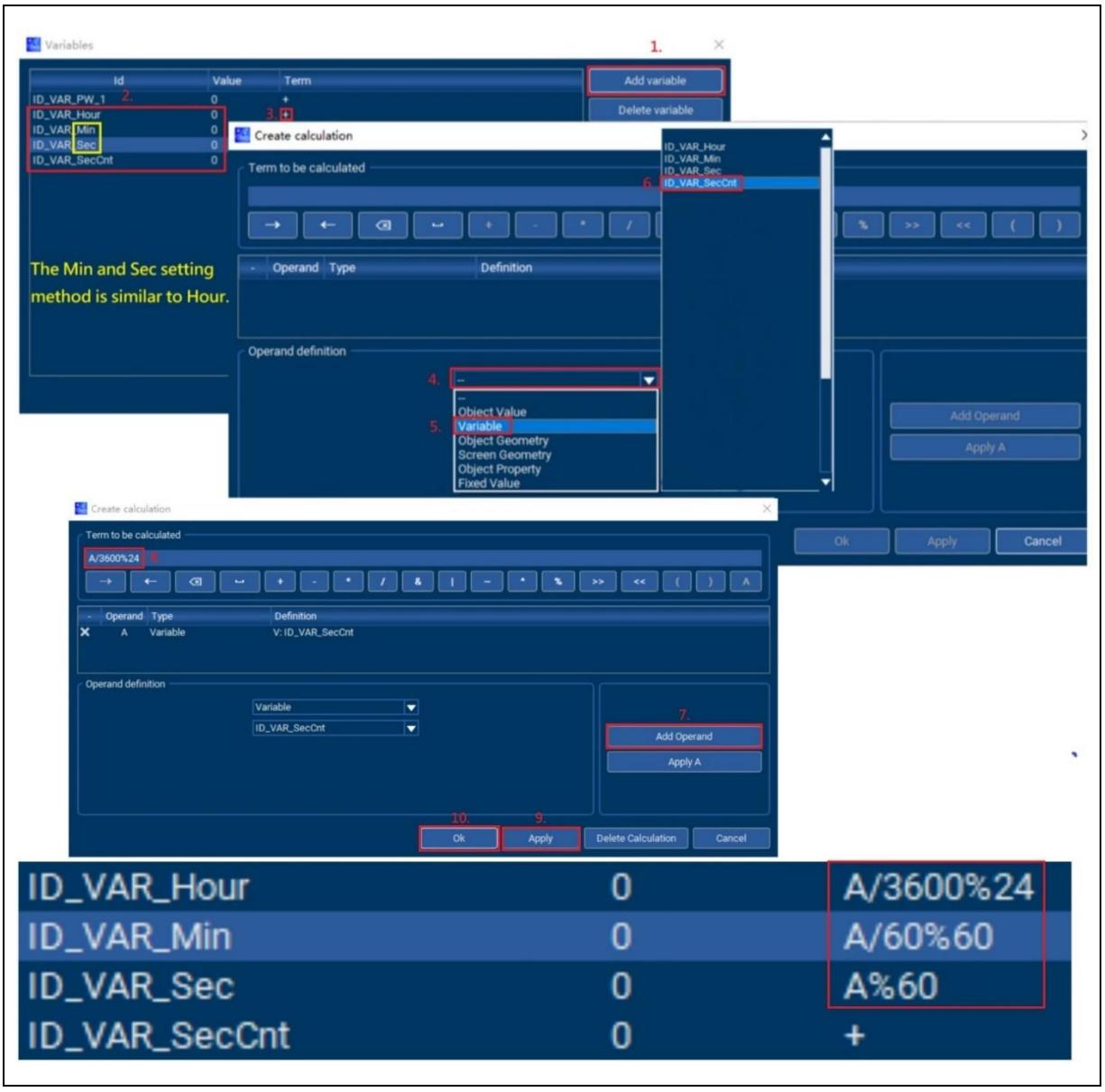


图 1-42 实时时钟相关变量定义

2) 实时时钟显示设计。上电启动从 00 时 00 分 00 秒开始计时。

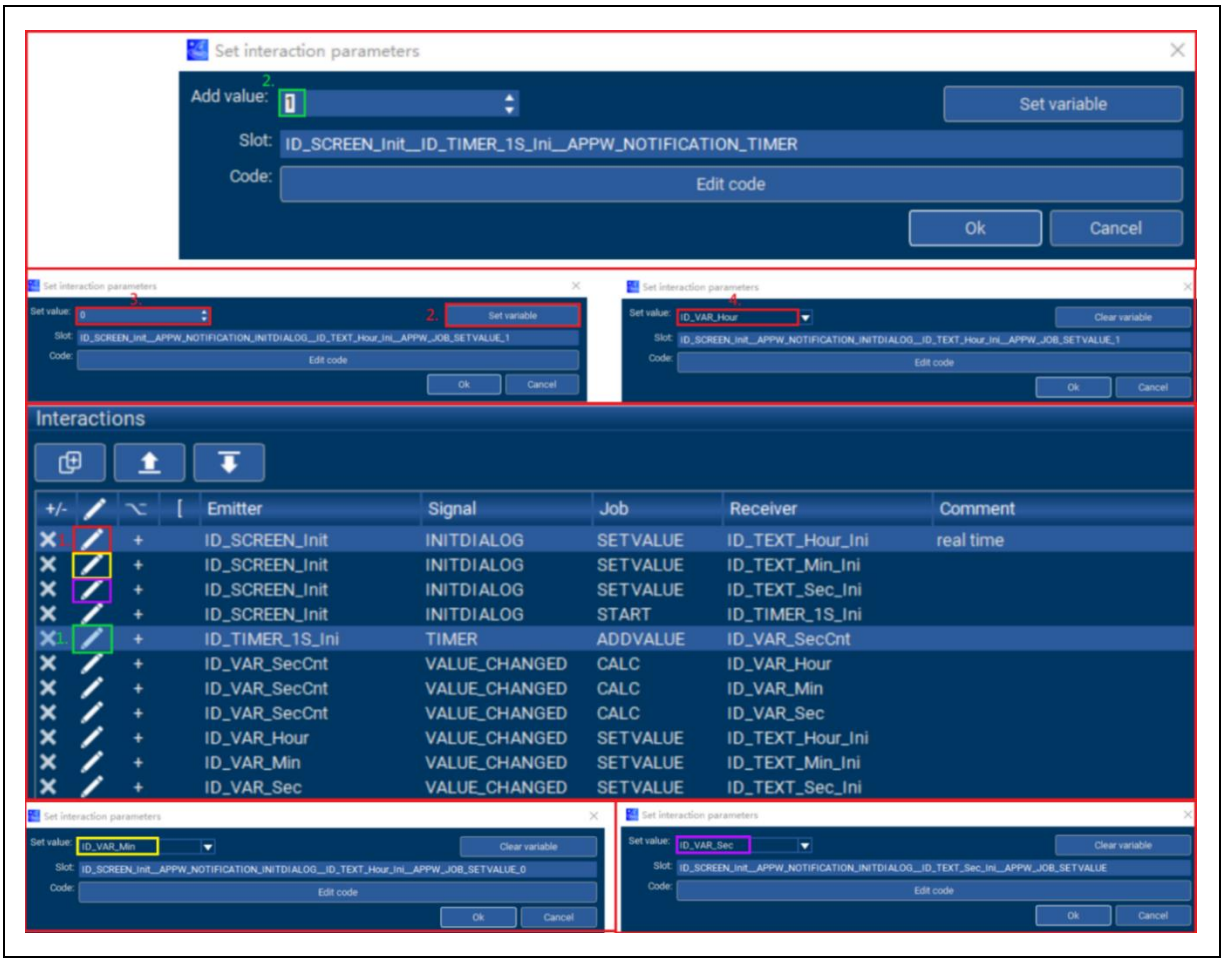


图 1-43 实时时钟显示设计

3) 定义语言变量“Lan”。为 0 表示中文，为 1 表示英文。

图 1-44 定义语言变量

4) 加载 Init 屏幕更新“Lan”键图案。

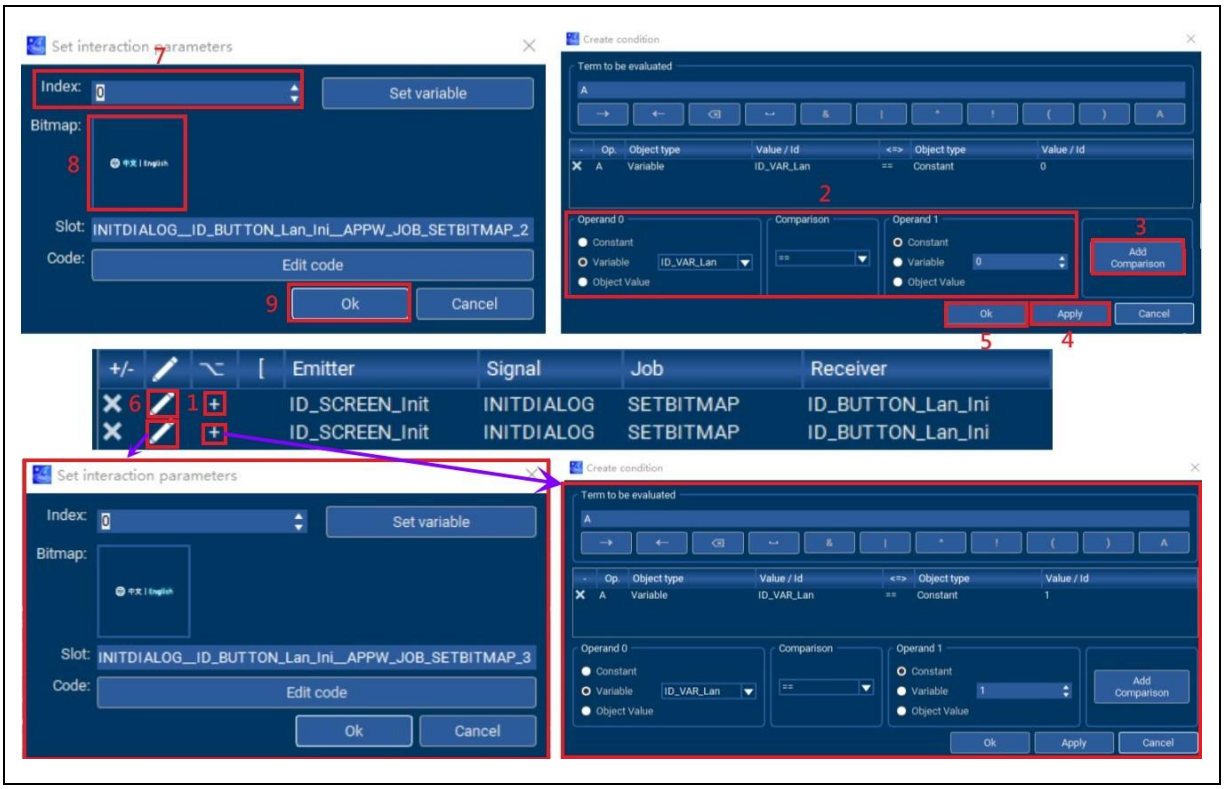


图 1-45 加载 Init 屏幕更新“Lan”键图案

5) 加载 Init 屏幕更新“Charge”键图案。

图 1-46 加载 Init 屏幕更新“Charge”键图案

6) 加载 Init 屏幕更新背景“Bg”、 “Login” 键 、“Price” 键 、“Info” 键图案，参考 1.2.3 章 2 节 4 步骤。加载 Init 屏幕更新“Acc”键 、“Query” 键 ，参考 1.2.3 章 2 节 5 步骤。

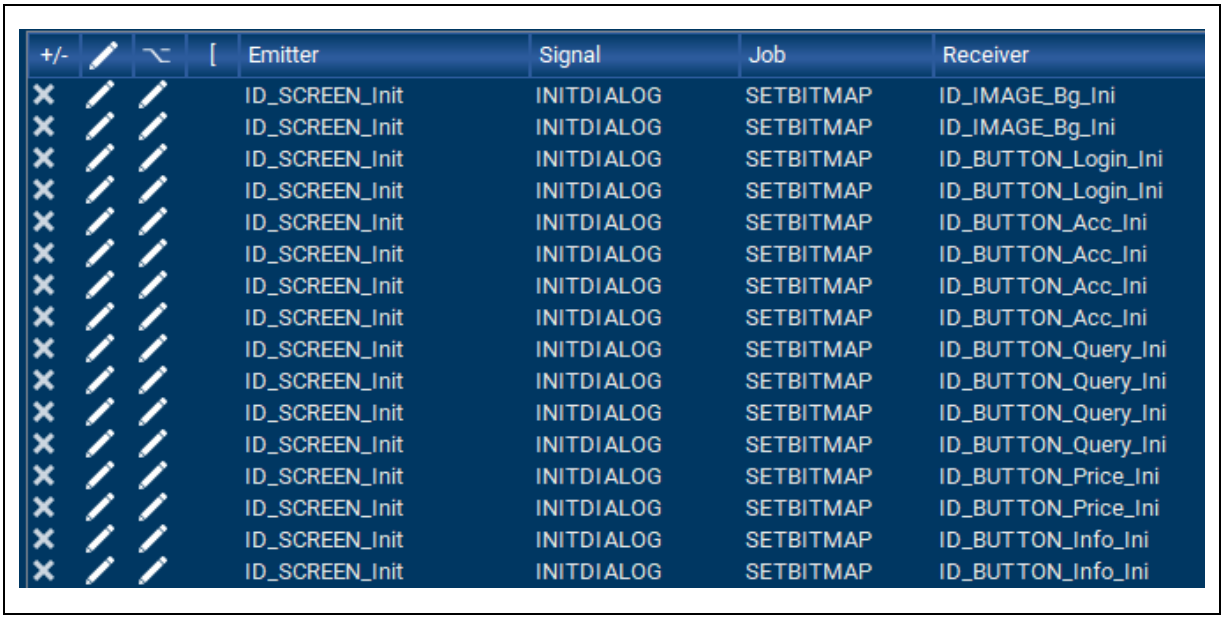


图 1-47 加载 Init 屏幕更新“Bg ”、“Login” 、“Price” 、“Info” 、“Acc” 、“Query”图案

7) 按“Lan”键改变“Lan”变量的值，值的范围是 0 或 1。

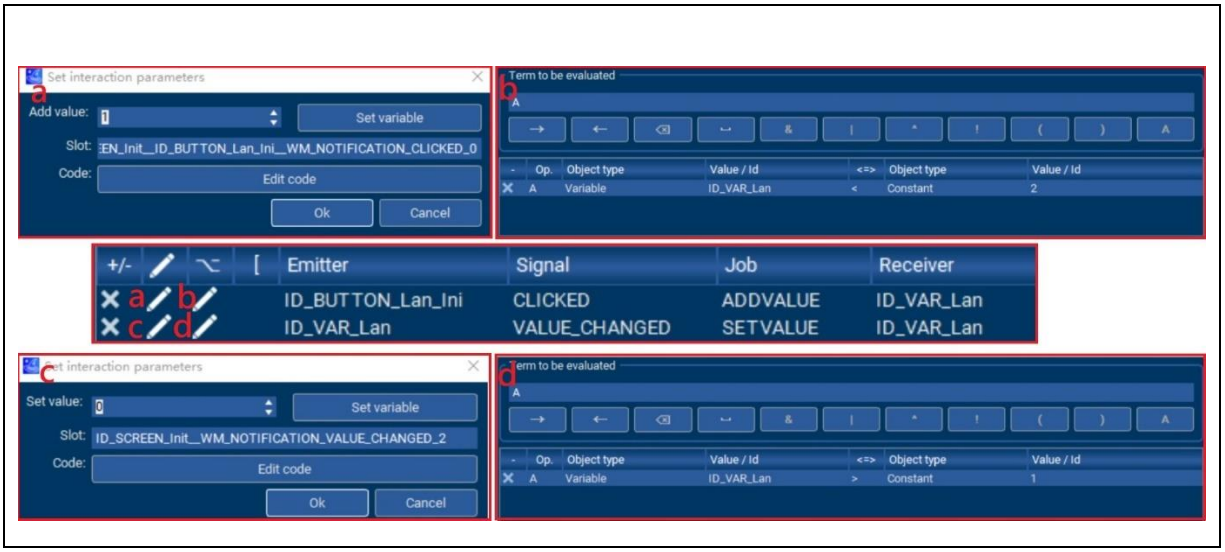


图 1-48 按“Lan”键改变“Lan”变量的值

8) 按“Lan”键实时更新“Lan”键图案。

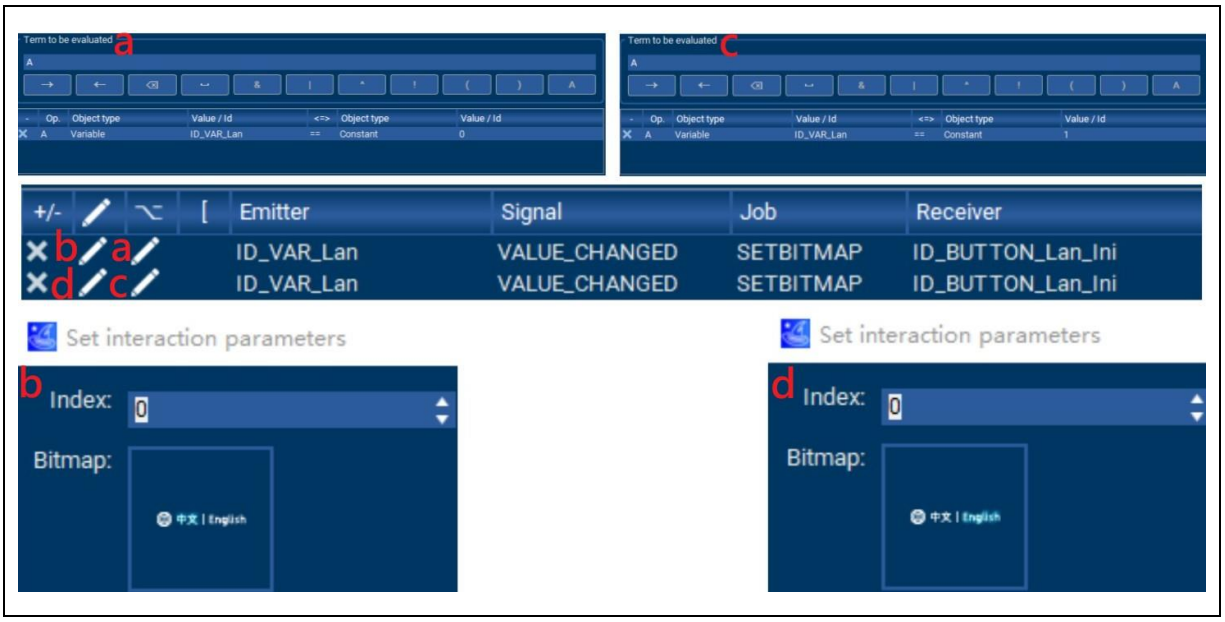


图 1-49 按“Lan”键实时更新“Lan”键图案

9) 设置“Lan”变量为 0 对应语言索引 0，“Lan”变量为 1 对应语言索引 1。

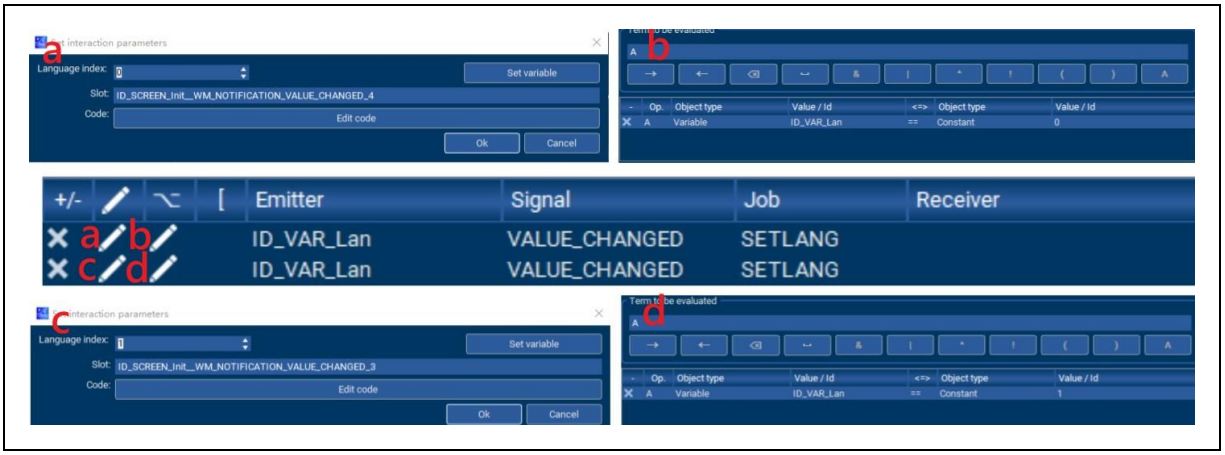


图 1-50 设置“Lan”变量与语言索引的对应关系

10) 点击“Login”、“Price”、“Info”、“Charge”、“Account”、“Query”键进行屏幕切换。

图 1-51 点击进行屏幕切换

11) “PW_1”变量变化结合 “Lan”变量的值，更新充电、账户、查询键图案。

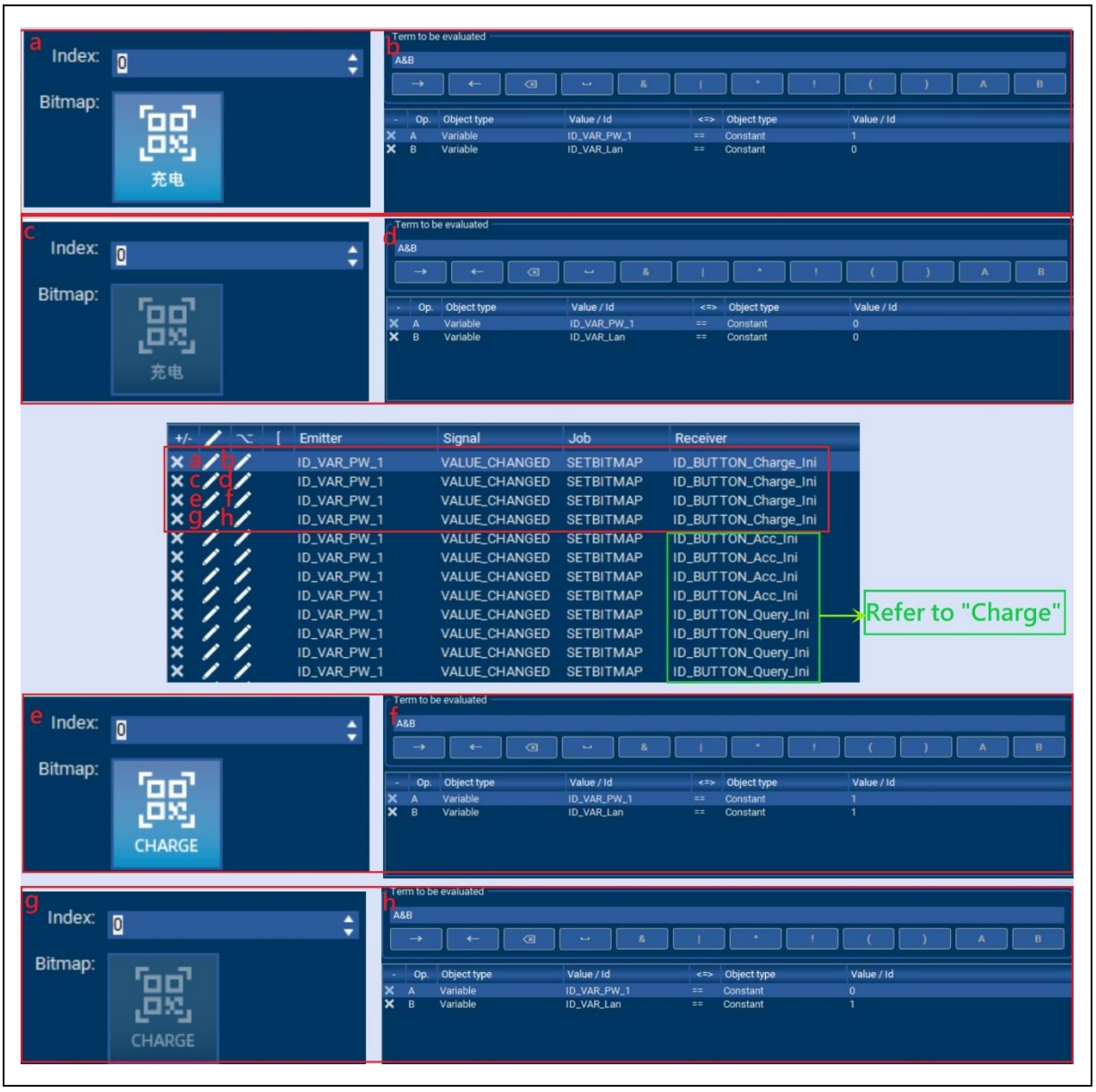


图 1-52 充电、账户、查询键图案显示

12) “Lan”变量变化结合 “PW_1”变量的值，更新登录、充电、账户、查询、价格、信息键及背景图案。

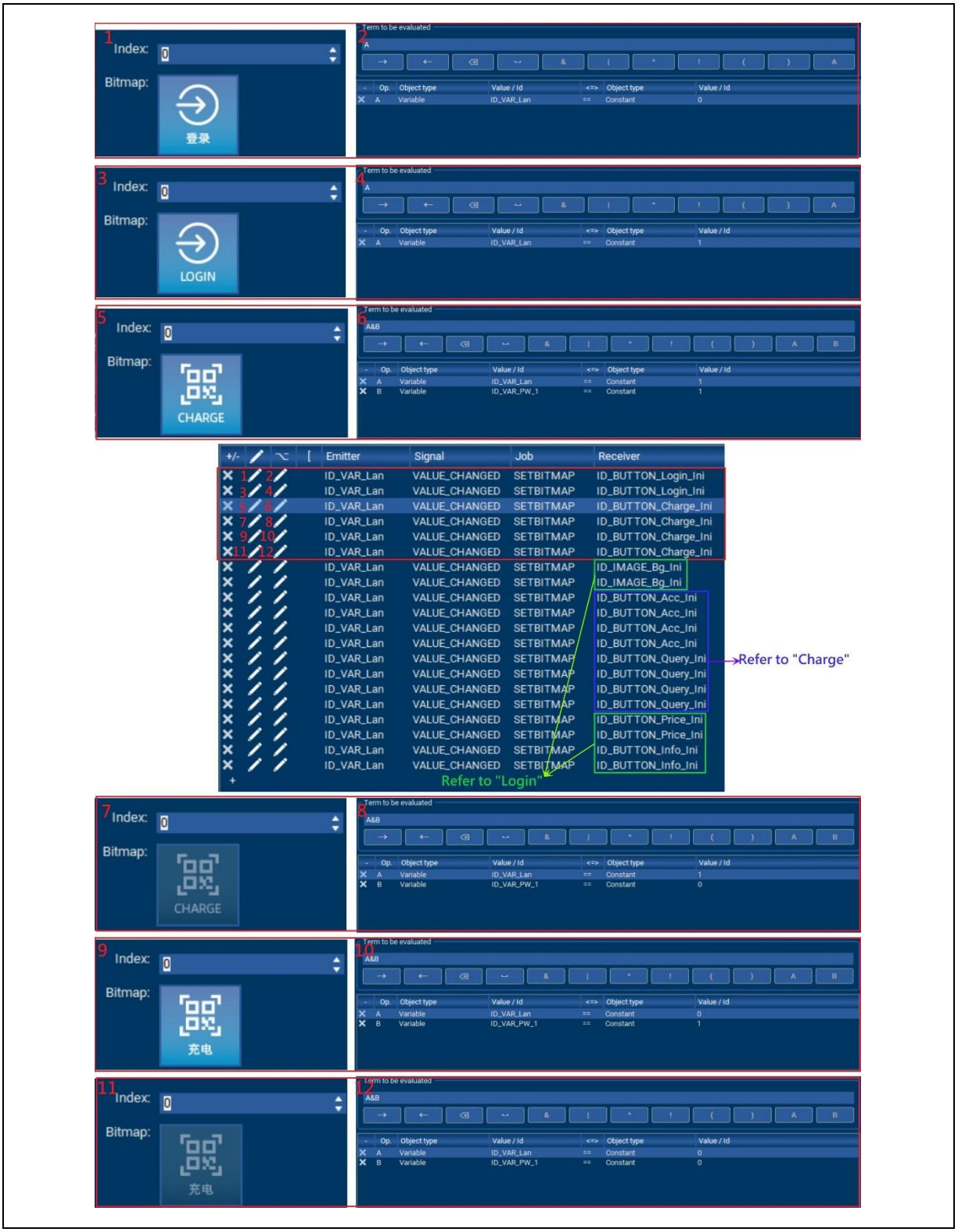


图 1-53 各按键及背景图案显示

3. Charge 屏幕交互设计。主要包括按“Return”键返回 Init 屏幕；实时时钟显示；结合代码实现用户名显示；实现显示及控制充电状态，包括充电时长显示；电压、电流、功率实时显示；当前电价显示；总充电电量及总充电费用以及尖、峰、平、谷各时间段充电电量及各时间段充电费用显示。充电百分比显示；通过按键可以启动充电或停止充电。

1) 定义充电变量“Charging”，为 0 表示停止充电，为 1 表示充电中。

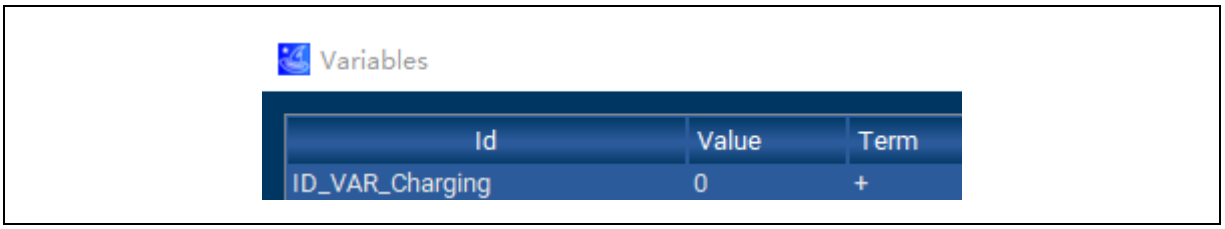


图 1-54 定义充电变量“Charging”

2) 按“Return”键返回 Init 屏幕。

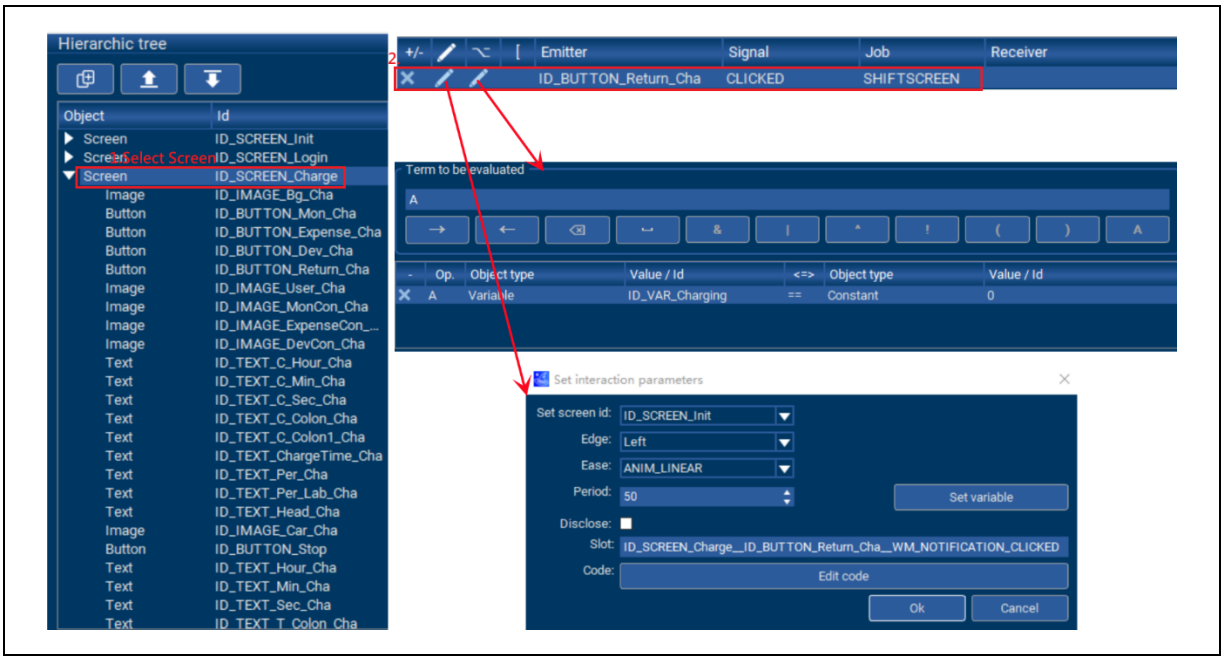
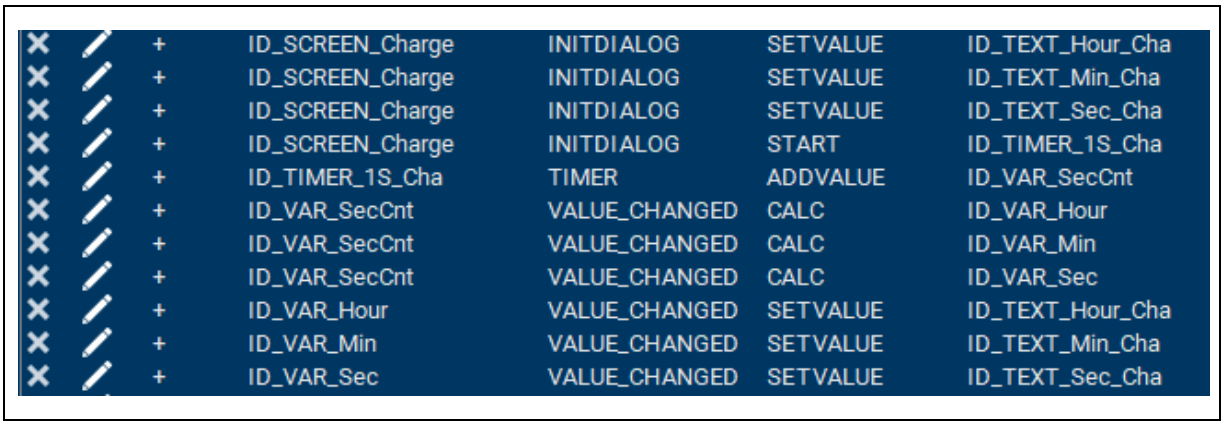


图 1-55 按“Return”键返回 Init 屏幕

3) 实时时钟显示设计。参考 1.2.3 章 2 节 2 步骤。



+	ID_SCREEN_Charge	INITDIALOG	SETVALUE	ID_TEXT_Hour_Cha
+	ID_SCREEN_Charge	INITDIALOG	SETVALUE	ID_TEXT_Min_Cha
+	ID_SCREEN_Charge	INITDIALOG	SETVALUE	ID_TEXT_Sec_Cha
+	ID_SCREEN_Charge	INITDIALOG	START	ID_TIMER_1S_Cha
+	ID_TIMER_1S_Cha	TIMER	ADDVALUE	ID_VAR_SecCnt
+	ID_VAR_SecCnt	VALUE_CHANGED	CALC	ID_VAR_Hour
+	ID_VAR_SecCnt	VALUE_CHANGED	CALC	ID_VAR_Min
+	ID_VAR_SecCnt	VALUE_CHANGED	CALC	ID_VAR_Sec
+	ID_VAR_Hour	VALUE_CHANGED	SETVALUE	ID_TEXT_Hour_Cha
+	ID_VAR_Min	VALUE_CHANGED	SETVALUE	ID_TEXT_Min_Cha
+	ID_VAR_Sec	VALUE_CHANGED	SETVALUE	ID_TEXT_Sec_Cha

图 1-56 实时时钟显示

4) 定义充电相关变量(“ConstantVolMode”、“Charge_Hour”、“Charge_Min”、“Charge_Sec”、“Charge_SecCnt”)。“ConstantVolMode”为 0 表示为恒流模式，“ConstantVolMode”为 1 表示为恒压模式。

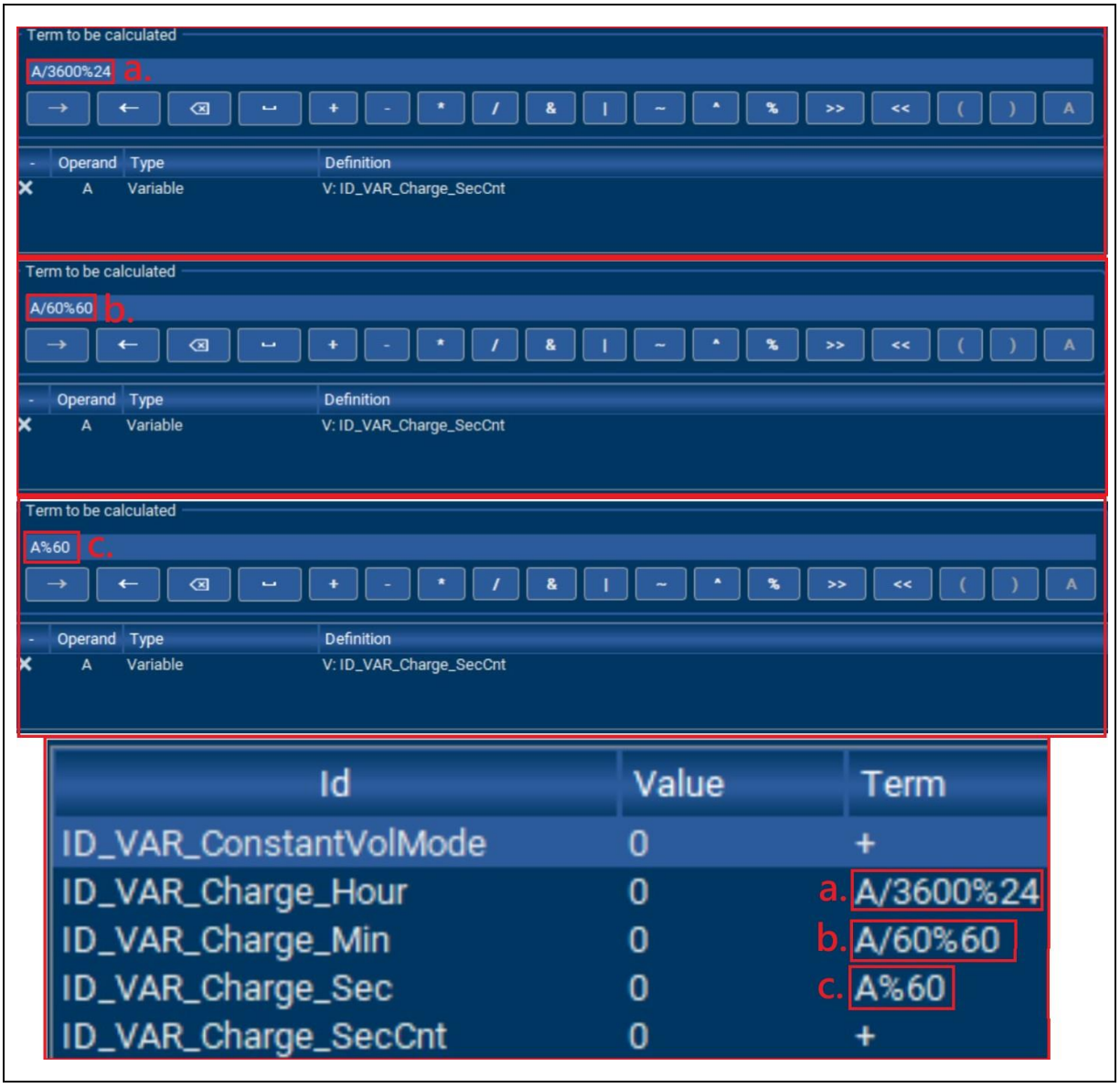


图 1-57 定义充电相关变量

5) 充电时间显示。参考 1.2.3 章 3 节 3 步骤。

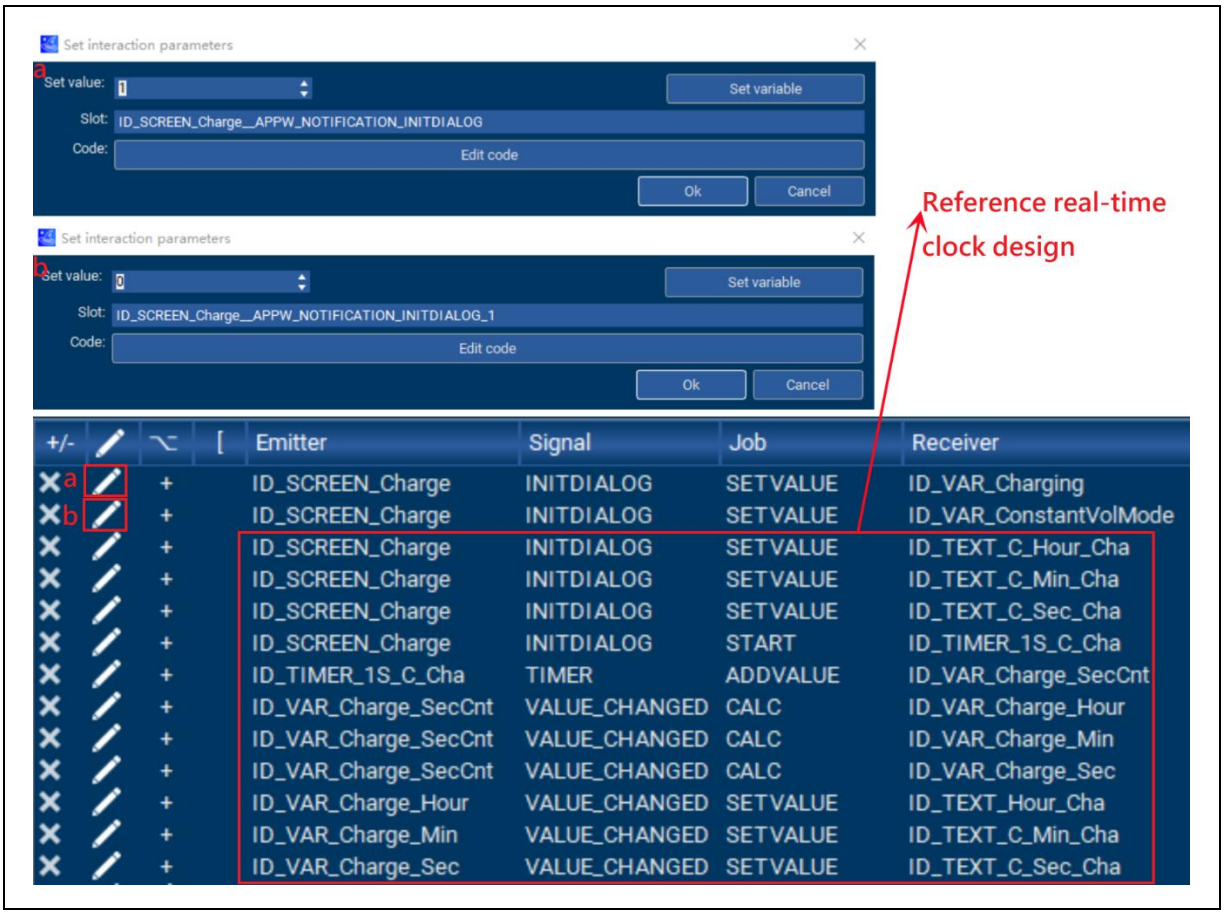


图 1-58 充电时间显示

6) 充电模式更新。“ChargeSecCnt”小于 60 为恒流模式，大于等于 60 为恒压模式。

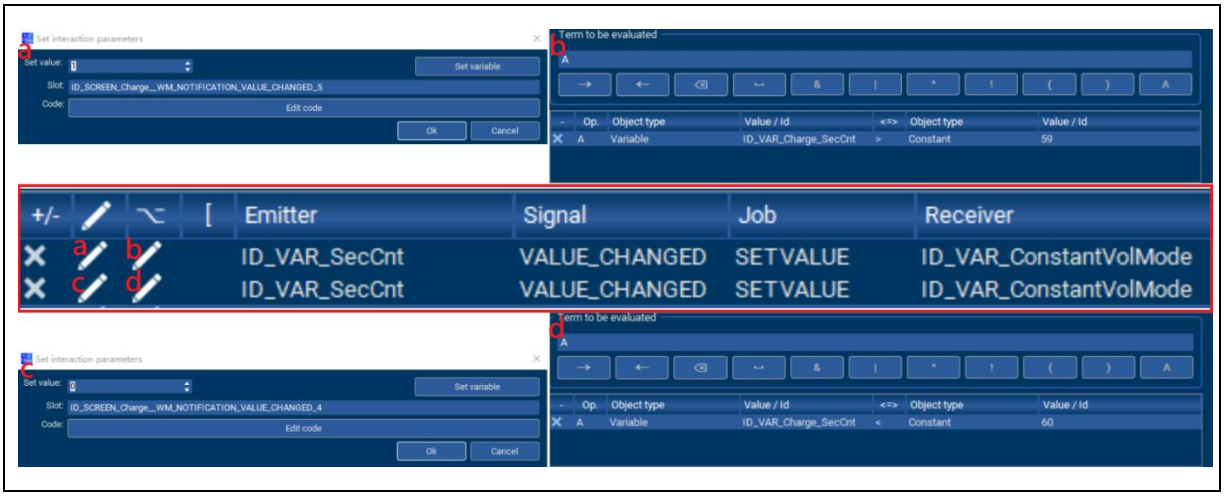


图 1-59 充电模式更新

7) 定义当前价格变量 “CurPrice”。

图 1-60 当前价格变量定义

8) 当前价格更新。为了方便计算电量及费用，假定 1 秒相当于实际 4 分钟。

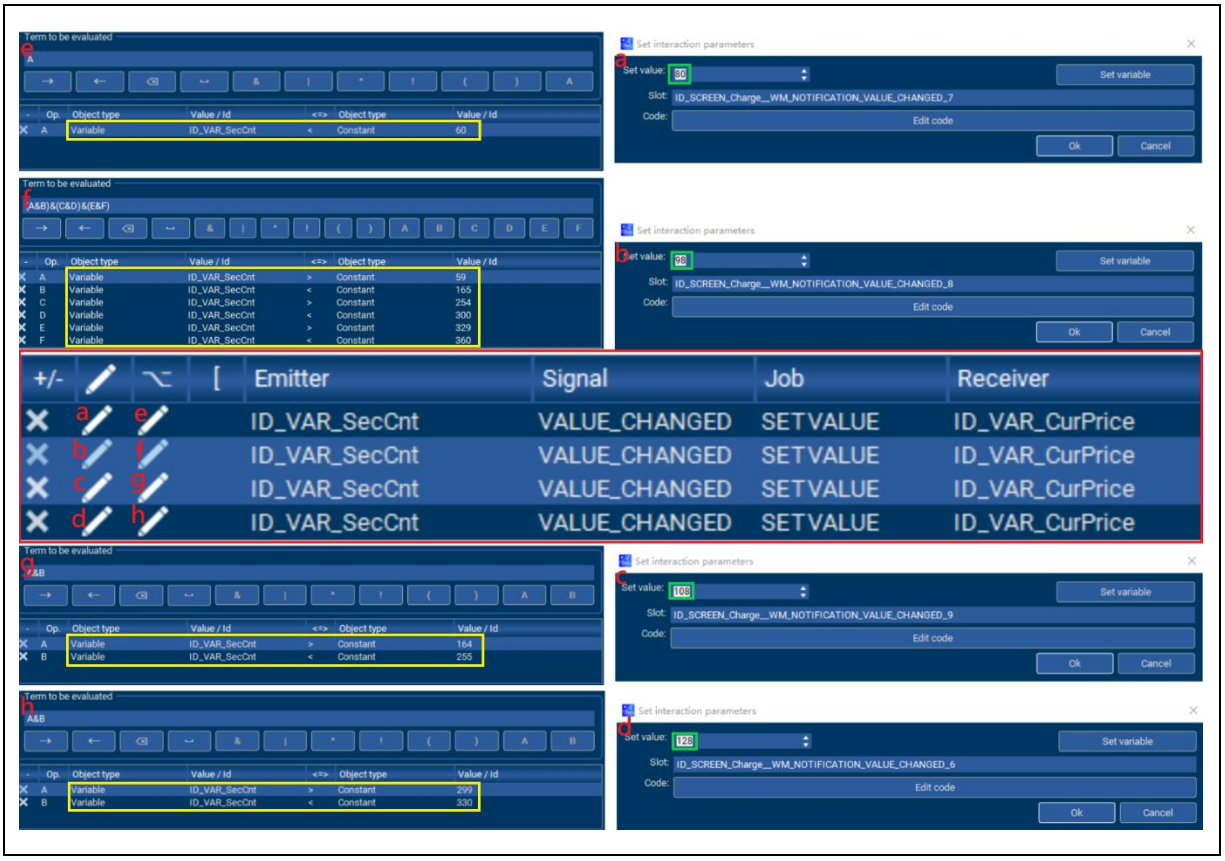


图 1-61 当前价格更新设计

9) 定义电流、电压变量(“Cur”、“Voltage”)。初值分别为 30A、281V。恒流模式每隔 4 分钟电压升高 1.16V；恒压模式每隔 4 分钟电流变小 1.8A。

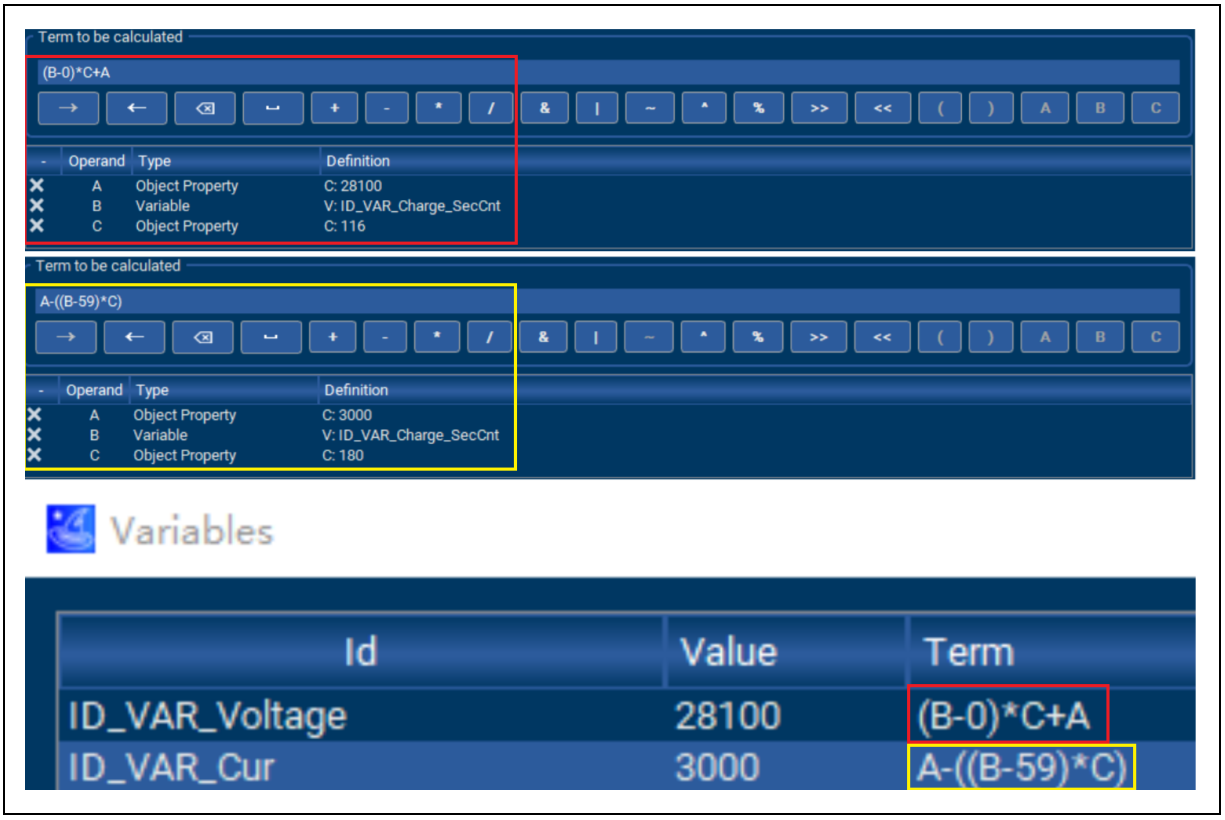


图 1-62 电流、电压变量定义

10) 电流、电压显示。

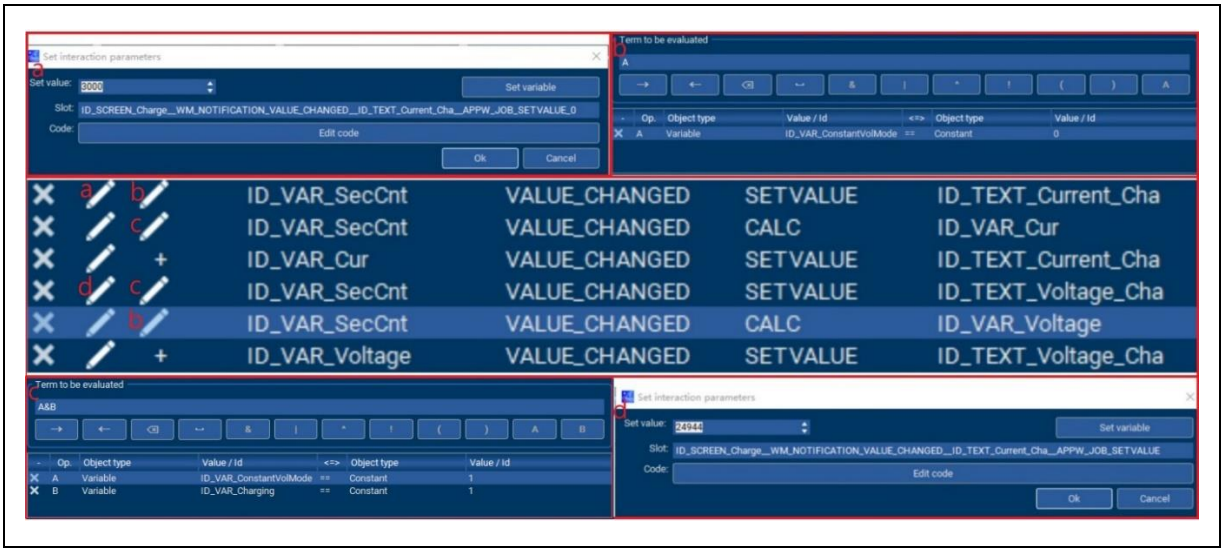


图 1-63 电流、电压显示

11) 定义实时电量(“RealCapacity”)、实时消费(“RealCost”)及累计电量(“SumCapacity”)、累计消费(“SumCost”)变量。

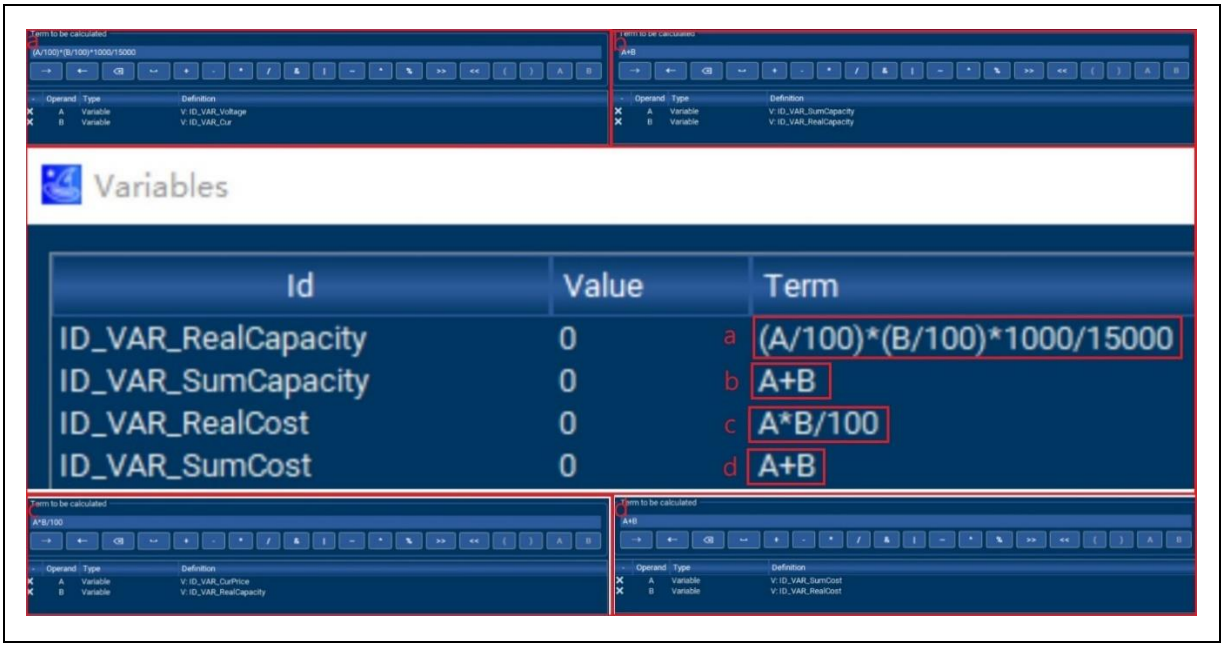


图 1-64 定义实时电量、实时消费和累计电量、累计消费

12) 计算实时电量、实时消费及累计电量、累计消费，并显示累计电量和累计消费。

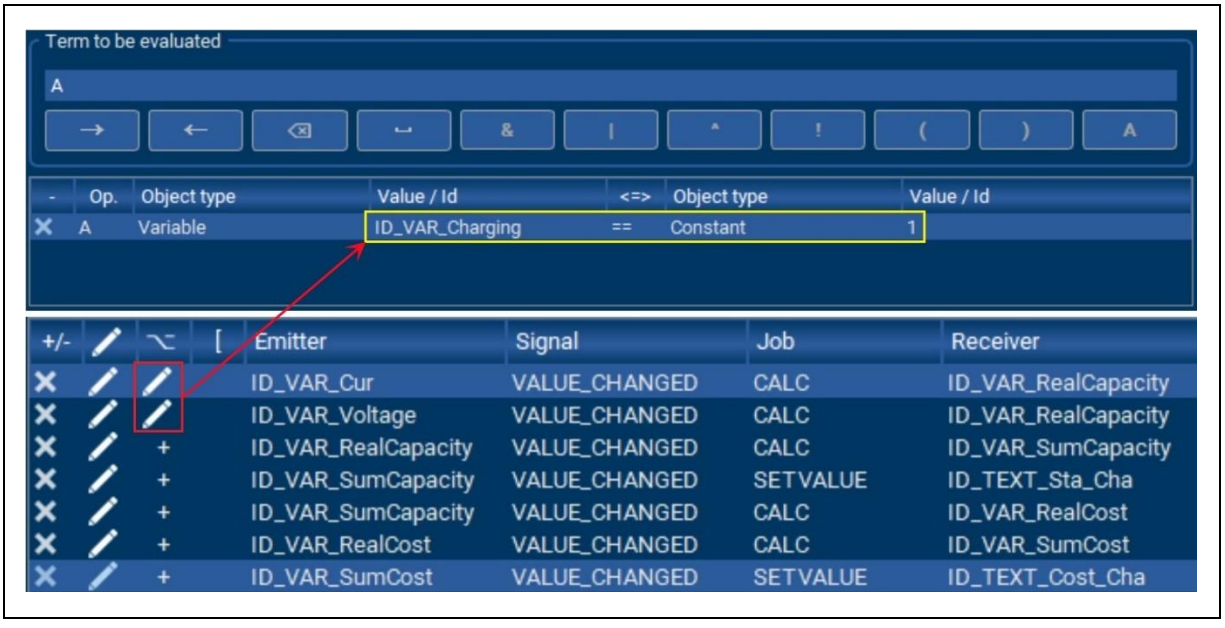


图 1-65 计算实时电量、实时消费及累计电量、累计消费

13) 停止充电，包括充满停止充电和按“Stop”键停止充电。

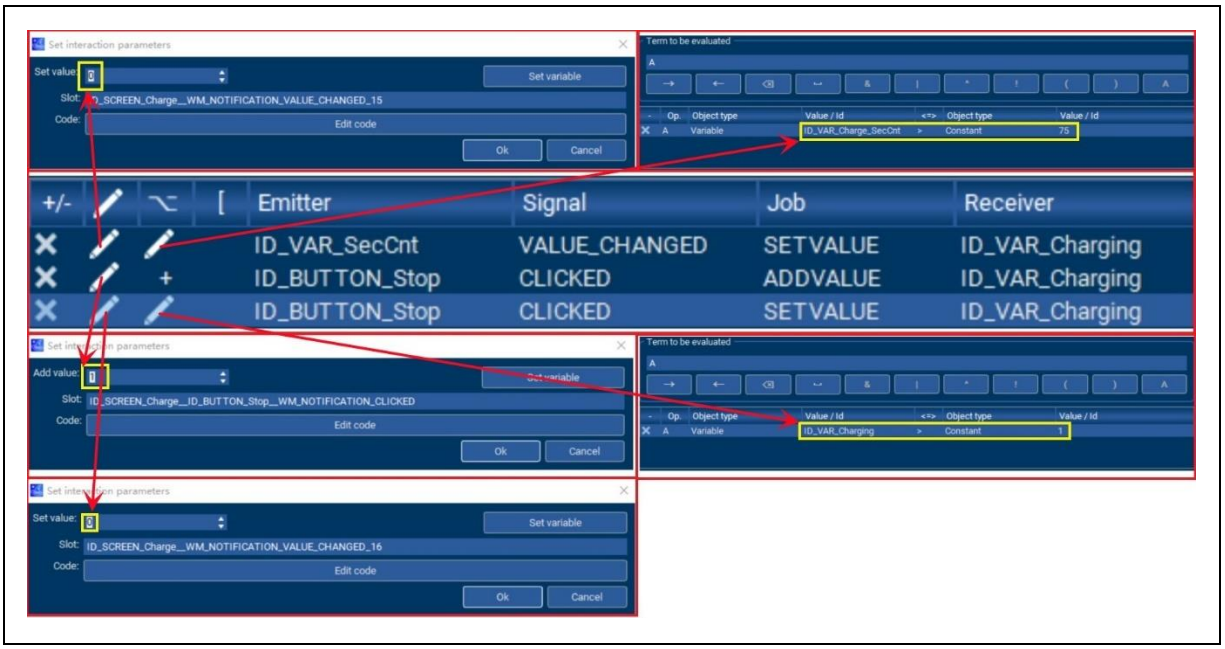


图 1-66 停止充电

14) 启动、停止充电的相关动作。

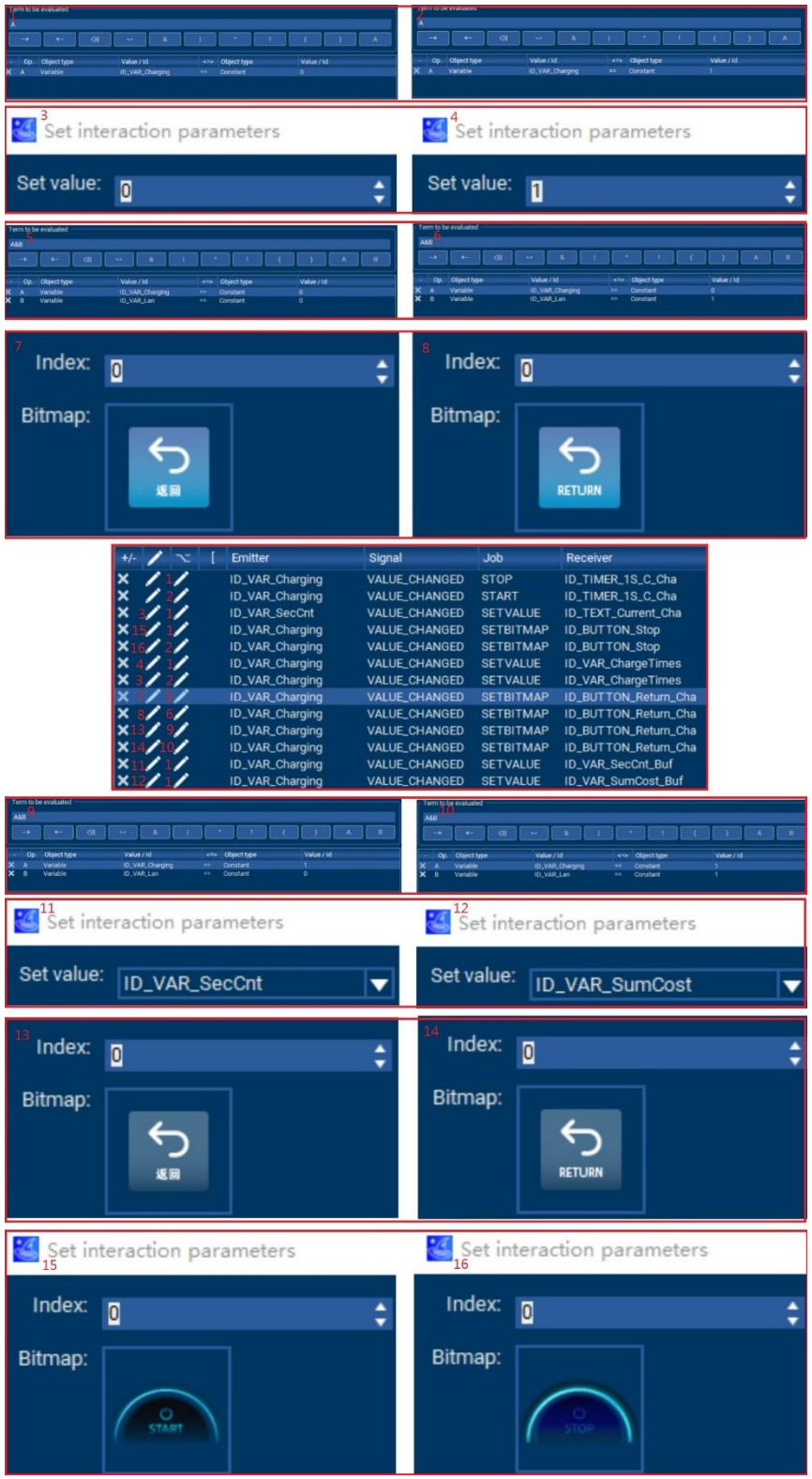


图 1-67 启动、停止充电的相关动作

15) 加载 Charge 屏幕，默认显示监控窗口，并进入充电状态。

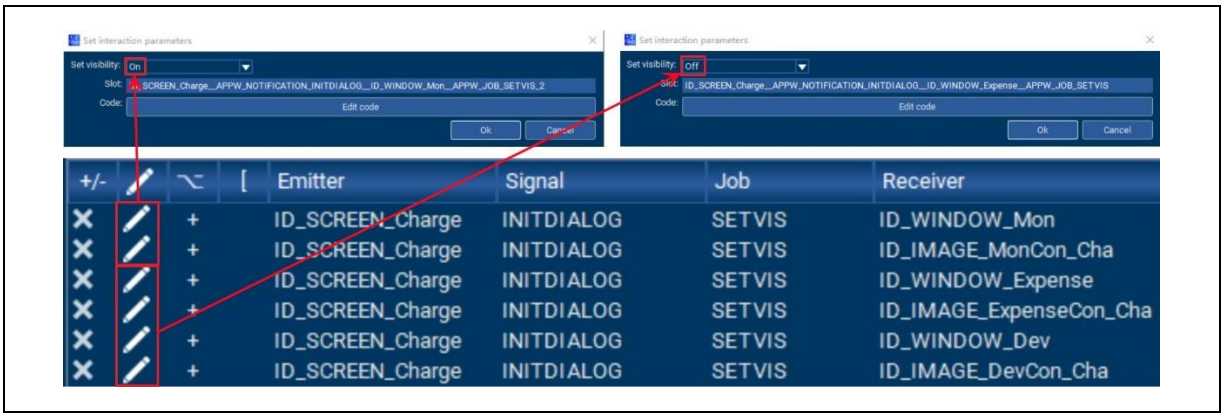


图 1-68 加载 Charge 屏幕默认显示监控窗口

16) 监控键按下显示监控窗口信息，消费、设备窗口信息不可见。

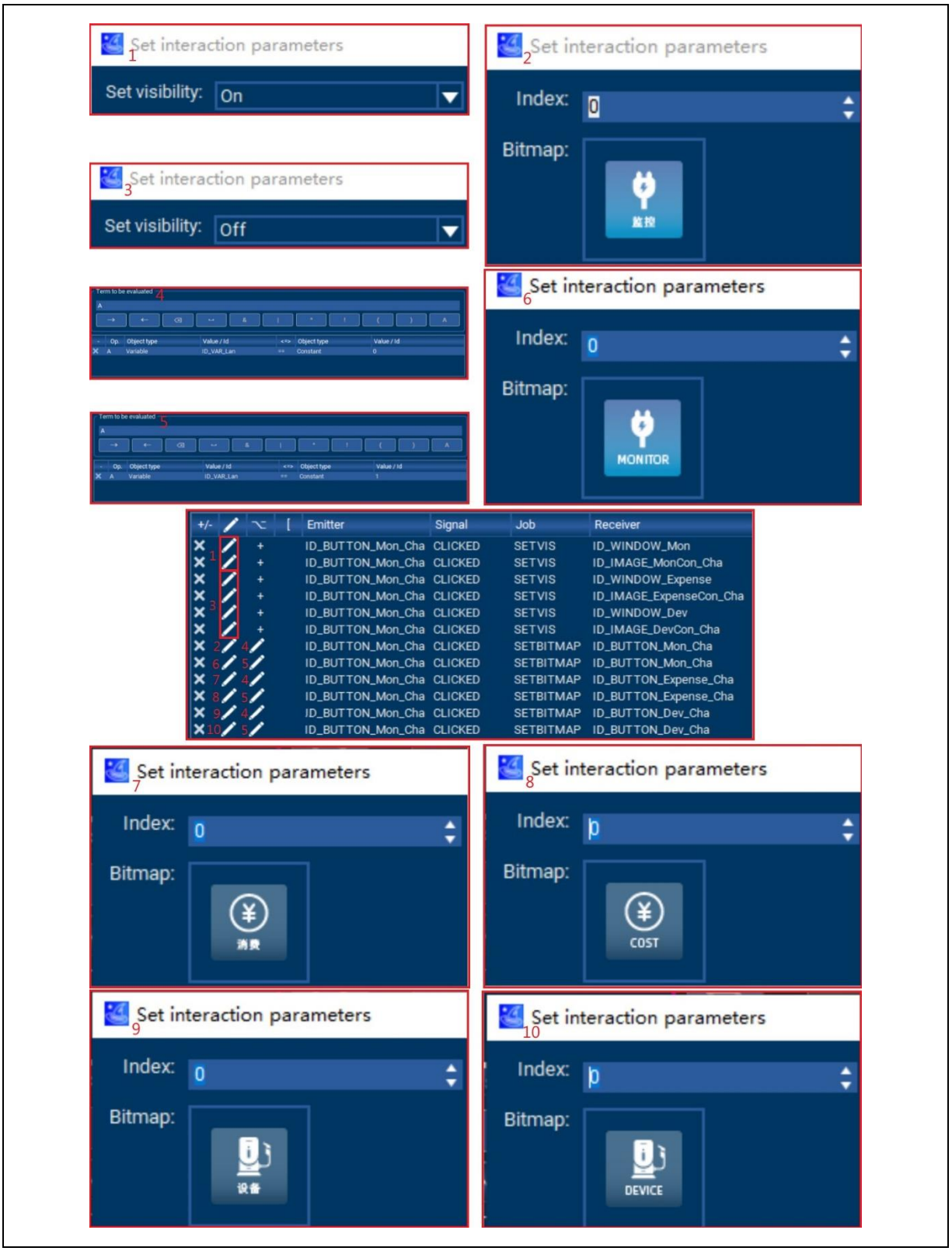
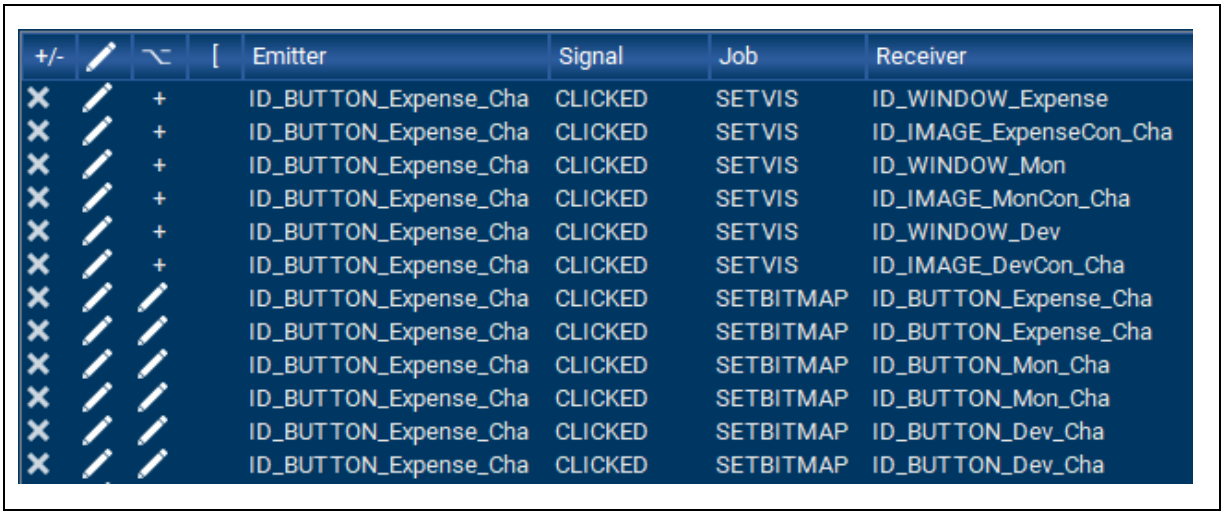


图 1-69 监控键按下

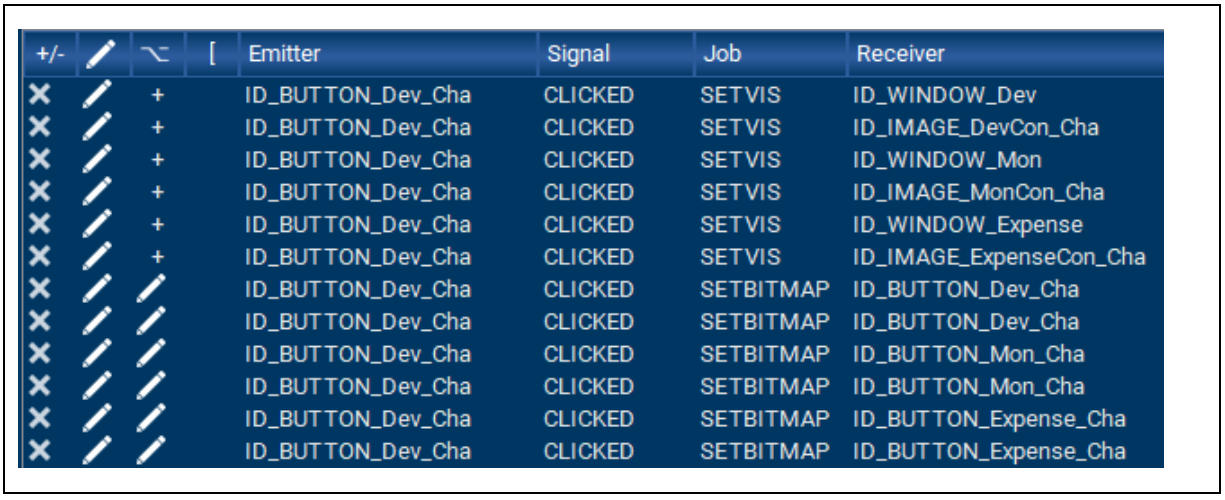
17) 消费键按下显示消费窗口信息，监控、设备窗口信息不可见。参考 1.2.2 章 3 节 16 步骤。



+/-		⌥	[	Emitter	Signal	Job	Receiver
✕		+		ID_BUTTON_Expense_Cha	CLICKED	SETVIS	ID_WINDOW_Expense
✕		+		ID_BUTTON_Expense_Cha	CLICKED	SETVIS	ID_IMAGE_ExpenseCon_Cha
✕		+		ID_BUTTON_Expense_Cha	CLICKED	SETVIS	ID_WINDOW_Mon
✕		+		ID_BUTTON_Expense_Cha	CLICKED	SETVIS	ID_IMAGE_MonCon_Cha
✕		+		ID_BUTTON_Expense_Cha	CLICKED	SETVIS	ID_WINDOW_Dev
✕		+		ID_BUTTON_Expense_Cha	CLICKED	SETVIS	ID_IMAGE_DevCon_Cha
✕				ID_BUTTON_Expense_Cha	CLICKED	SETBITMAP	ID_BUTTON_Expense_Cha
✕				ID_BUTTON_Expense_Cha	CLICKED	SETBITMAP	ID_BUTTON_Expense_Cha
✕				ID_BUTTON_Expense_Cha	CLICKED	SETBITMAP	ID_BUTTON_Mon_Cha
✕				ID_BUTTON_Expense_Cha	CLICKED	SETBITMAP	ID_BUTTON_Mon_Cha
✕				ID_BUTTON_Expense_Cha	CLICKED	SETBITMAP	ID_BUTTON_Dev_Cha
✕				ID_BUTTON_Expense_Cha	CLICKED	SETBITMAP	ID_BUTTON_Dev_Cha

图 1-70 消费键按下

18) 设备键按下显示设备窗口信息，消费、监控窗口信息不可见。参考 1.2.2 章 3 节 16 步骤。



+/-		⌥	[	Emitter	Signal	Job	Receiver
✕		+		ID_BUTTON_Dev_Cha	CLICKED	SETVIS	ID_WINDOW_Dev
✕		+		ID_BUTTON_Dev_Cha	CLICKED	SETVIS	ID_IMAGE_DevCon_Cha
✕		+		ID_BUTTON_Dev_Cha	CLICKED	SETVIS	ID_WINDOW_Mon
✕		+		ID_BUTTON_Dev_Cha	CLICKED	SETVIS	ID_IMAGE_MonCon_Cha
✕		+		ID_BUTTON_Dev_Cha	CLICKED	SETVIS	ID_WINDOW_Expense
✕		+		ID_BUTTON_Dev_Cha	CLICKED	SETVIS	ID_IMAGE_ExpenseCon_Cha
✕				ID_BUTTON_Dev_Cha	CLICKED	SETBITMAP	ID_BUTTON_Dev_Cha
✕				ID_BUTTON_Dev_Cha	CLICKED	SETBITMAP	ID_BUTTON_Dev_Cha
✕				ID_BUTTON_Dev_Cha	CLICKED	SETBITMAP	ID_BUTTON_Mon_Cha
✕				ID_BUTTON_Dev_Cha	CLICKED	SETBITMAP	ID_BUTTON_Mon_Cha
✕				ID_BUTTON_Dev_Cha	CLICKED	SETBITMAP	ID_BUTTON_Expense_Cha
✕				ID_BUTTON_Dev_Cha	CLICKED	SETBITMAP	ID_BUTTON_Expense_Cha

图 1-71 设备键按下

19) 定义充电百分比变量“ChargePer”。

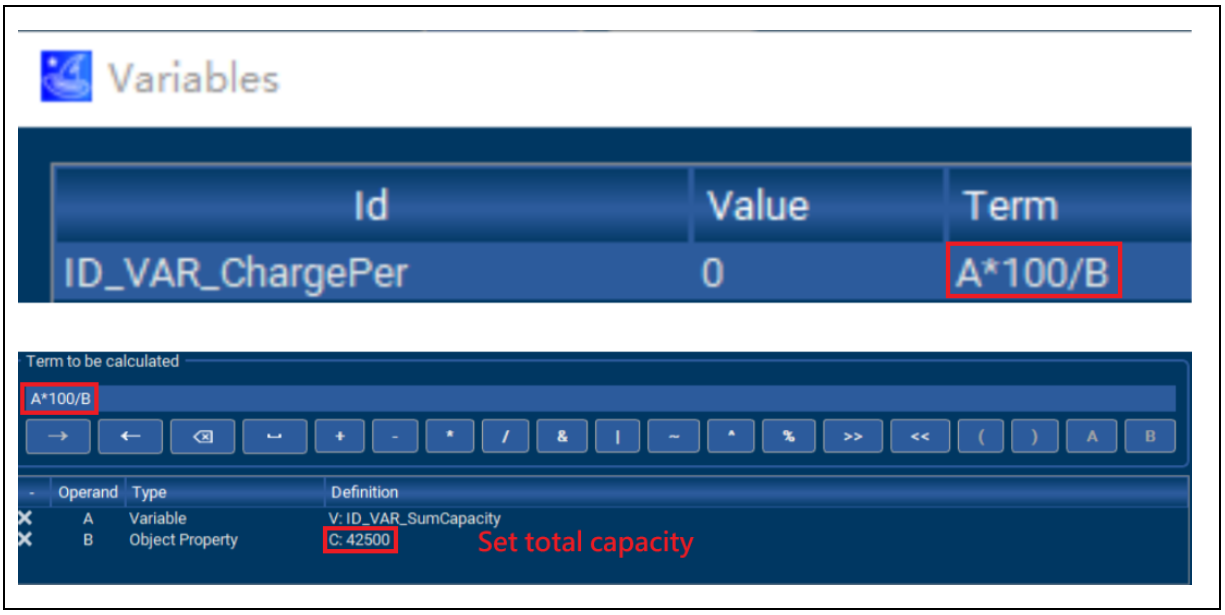


图 1-72 充电百分比变量定义

20) 充电百分比显示。0%~19%显示 1 格电量；20%~39%显示 2 格电量；40%~59%显示 3 格电量；60%~79%显示 4 格电量；80%~100%显示 5 格电量。

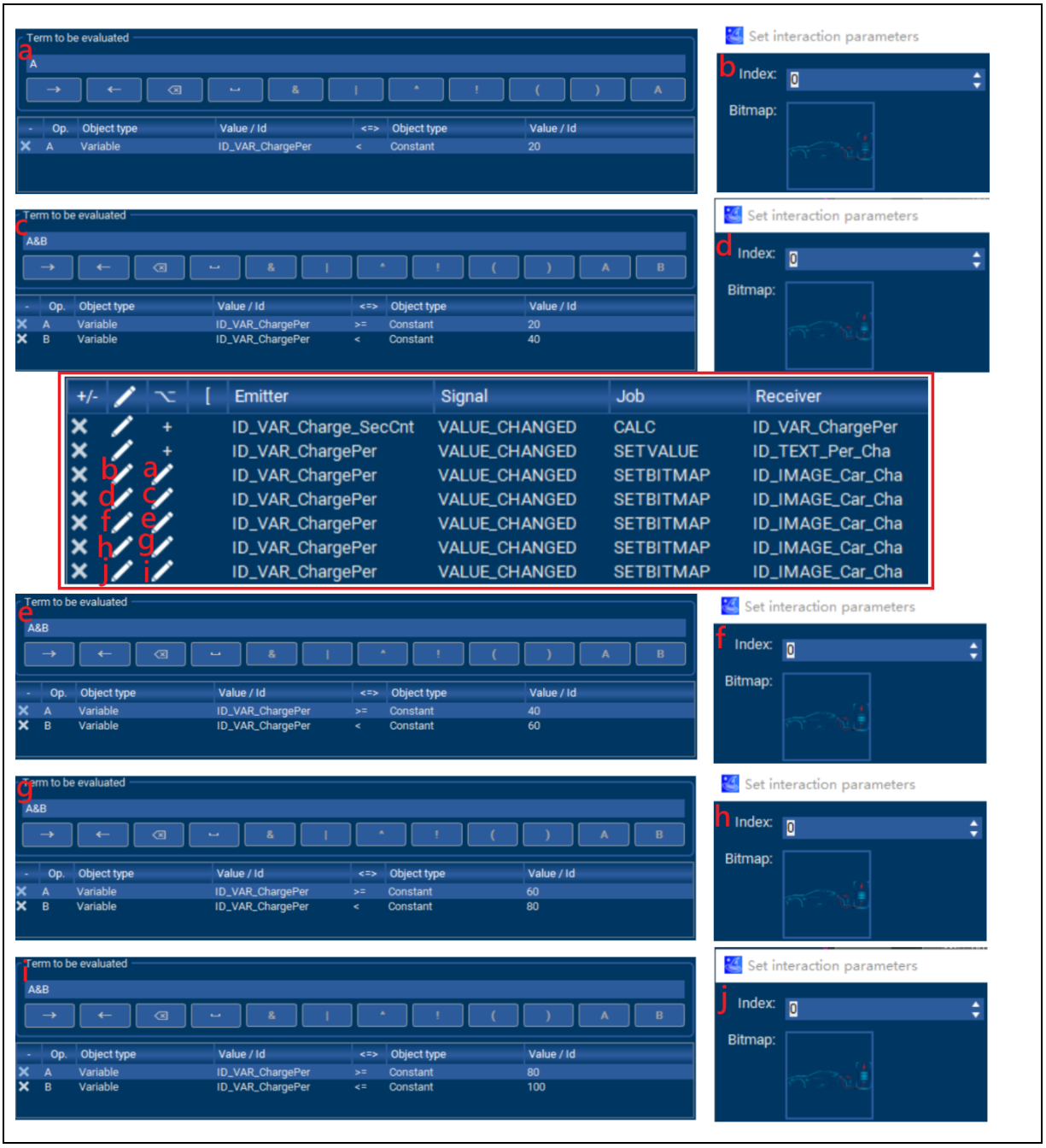


图 1-73 充电百分比显示

21) 定义尖、峰、平、谷各时间段累计电量、累计费用变量。

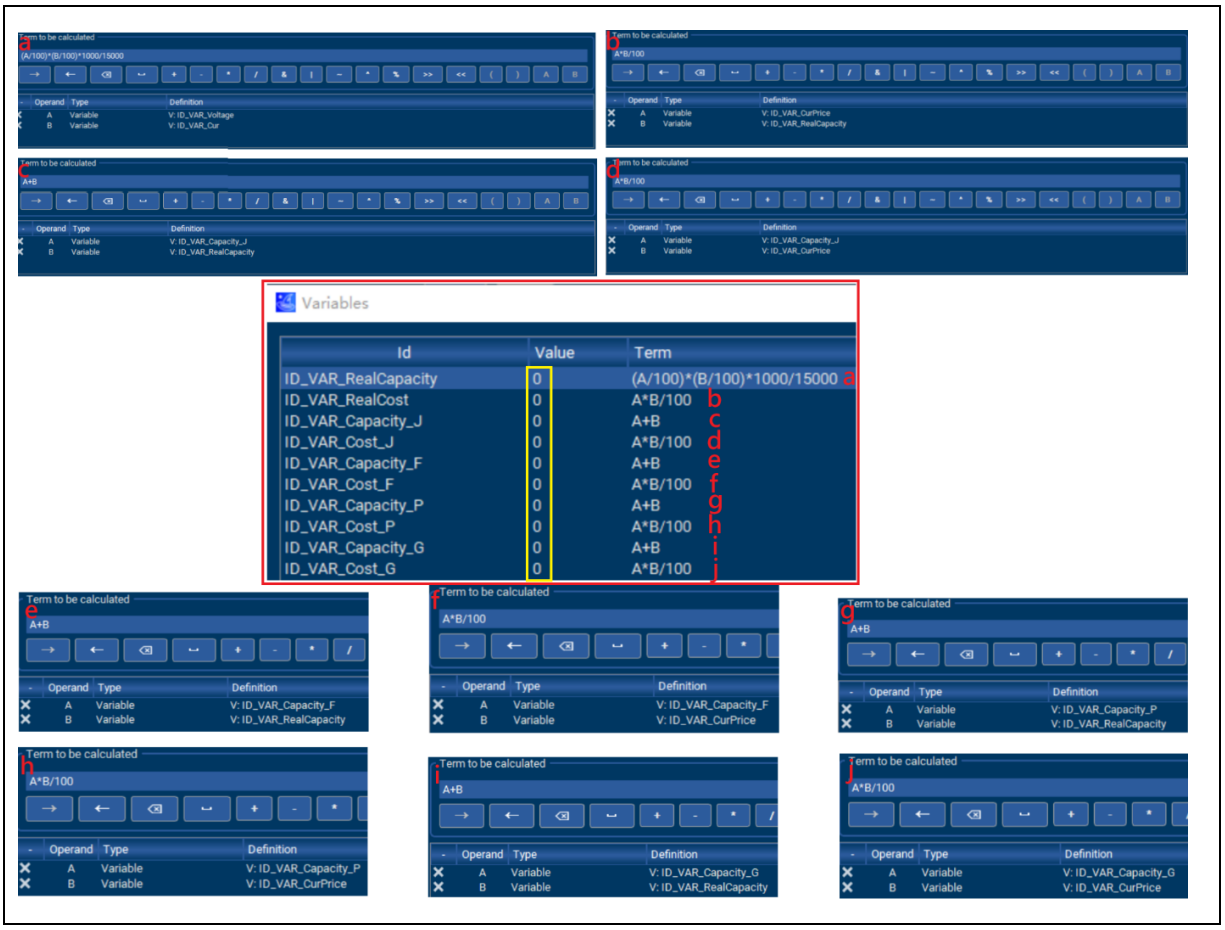


图 1-74 电量、费用变量定义

22) 尖、峰、平、谷各时间段所消费的累计电量及累计费用显示。

图 1-75 尖、峰、平、谷各时间段累计电量、累计费用显示

23) 定义功率变量。

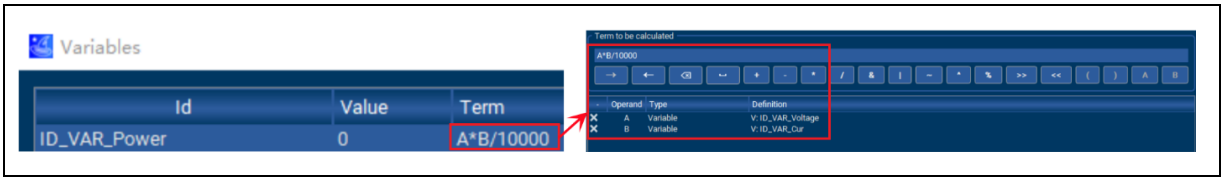


图 1-76 功率变量定义

24) 功率显示。

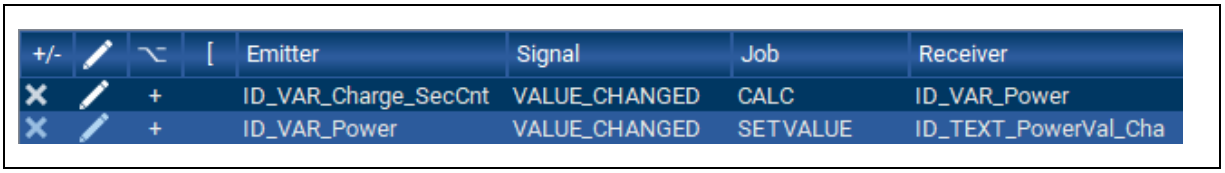


图 1-77 功率显示

25) 当前价格显示。

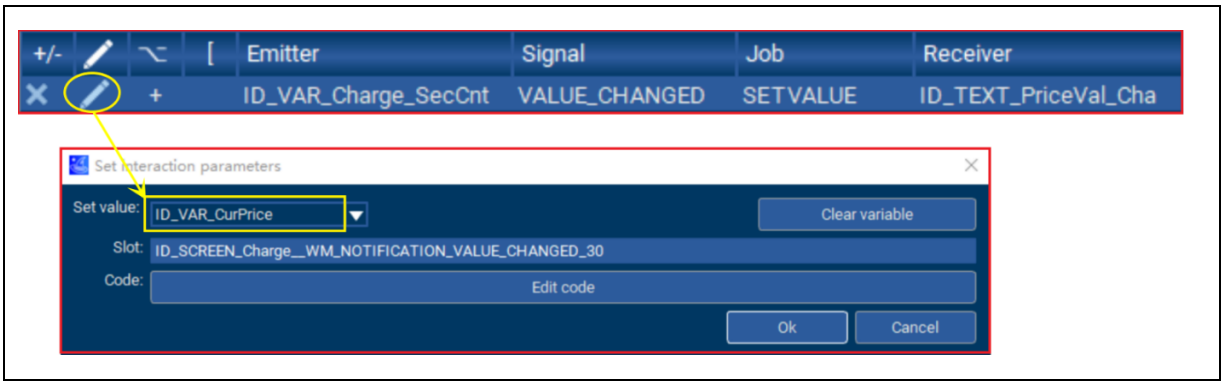


图 1-78 当前价格显示

26) 加载 Charge 屏幕，根据“Lan”变量的值更新监控、消费、设备键图案。

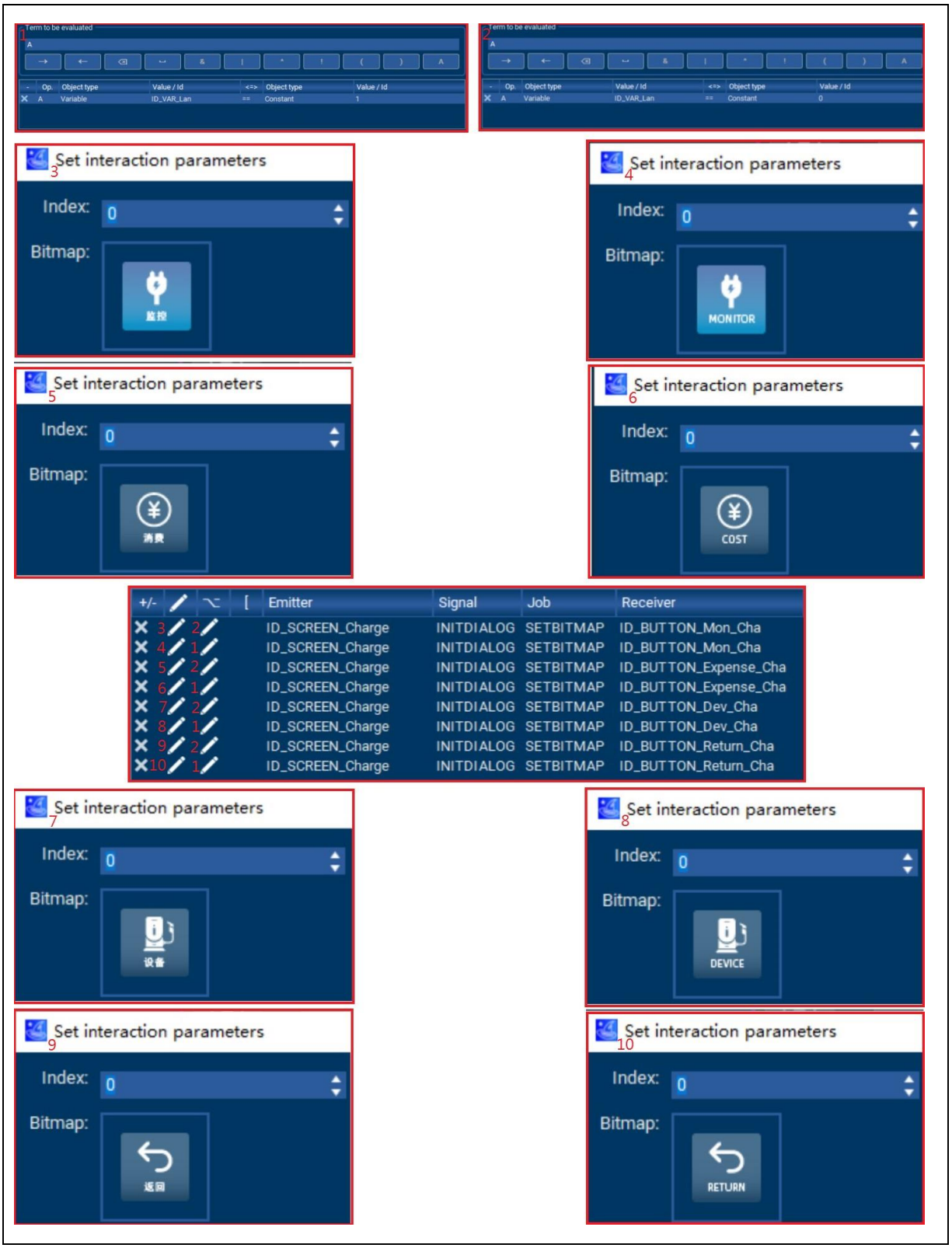


图 1-79 更新监控、消费、设备键图案

4. Price 屏幕交互设计。包括按“Return”键返回 Init 屏幕，显示实时时钟、显示各时间段的价格。

1) 按“Return”键返回 Init 屏幕。参考 1.2.3 章 1 节 1 步骤。

图 1-80 按“Return”键返回 Init 屏幕

2) 实时时钟显示。参考 1.2.3 章 2 节 2 步骤。

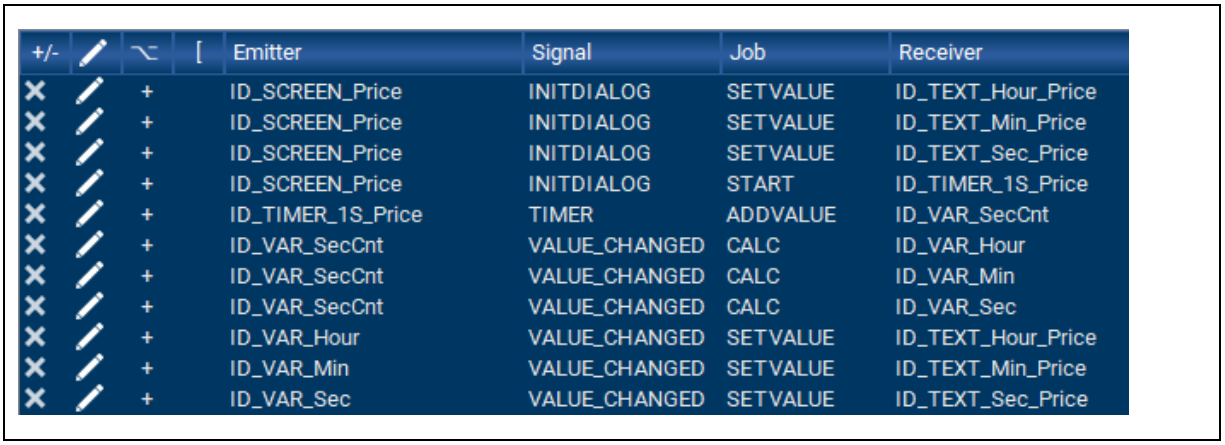


图 1-81 实时时钟显示

3) 定义尖、峰、平、谷各时间段价格变量。

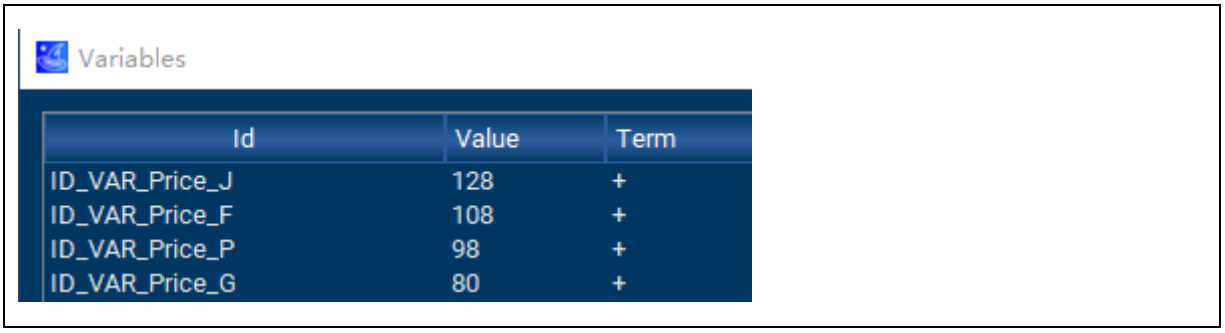


图 1-82 尖、峰、平、谷各时间段价格变量定义

4) 显示尖、峰、平、谷各时间段的价格。

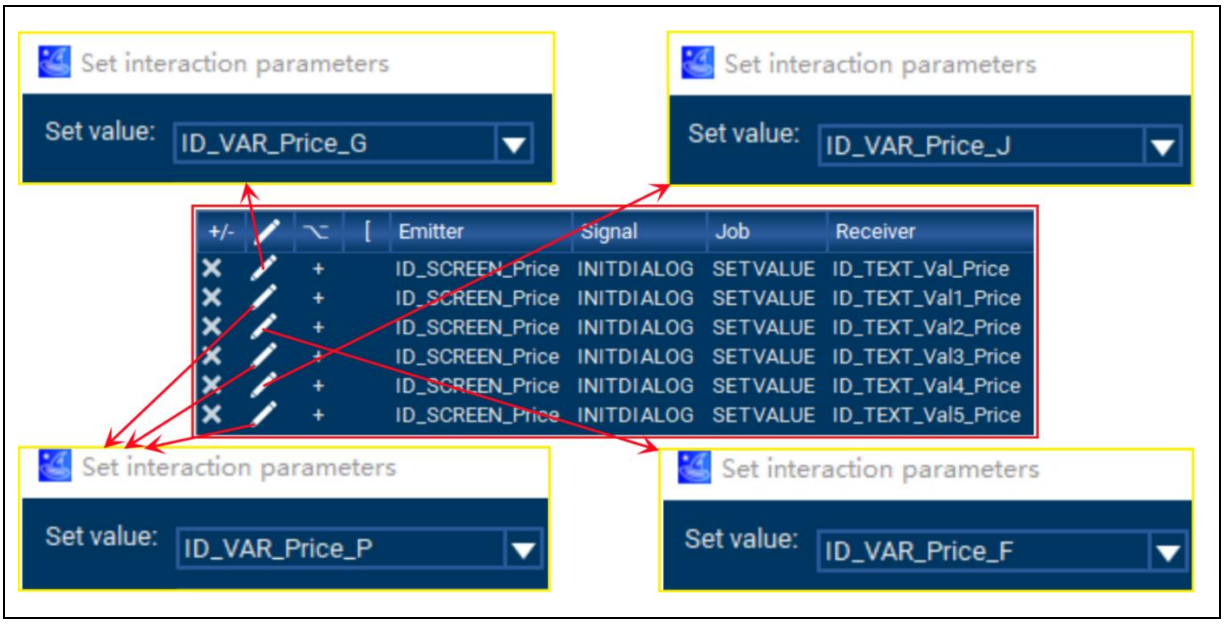


图 1-83 尖、峰、平、谷各时间段价格显示

5) 加载 Price 屏幕，根据“Lan”变量的值更新“Return”键图案。参考 1.2.3 章 1 节 3 步骤。

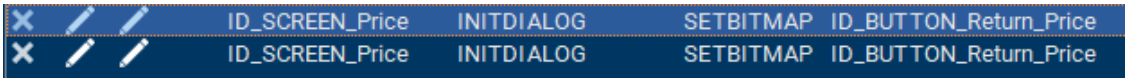


图 1-84 更新“Return”键图案

5. Info 屏幕设计。包括按“Return”键返回 Init 屏幕；显示实时时钟；显示充电站名称、支持的充电类型以及可用的充电枪号。

1) 按“Return”键返回 Init 屏幕。参考 1.2.3 章 1 节 1 步骤。

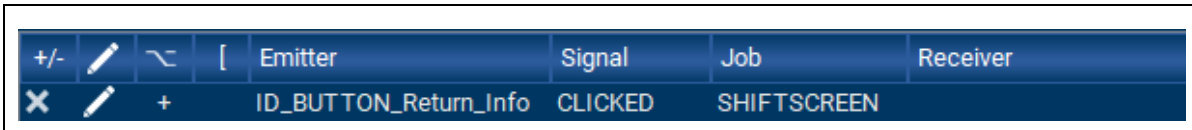
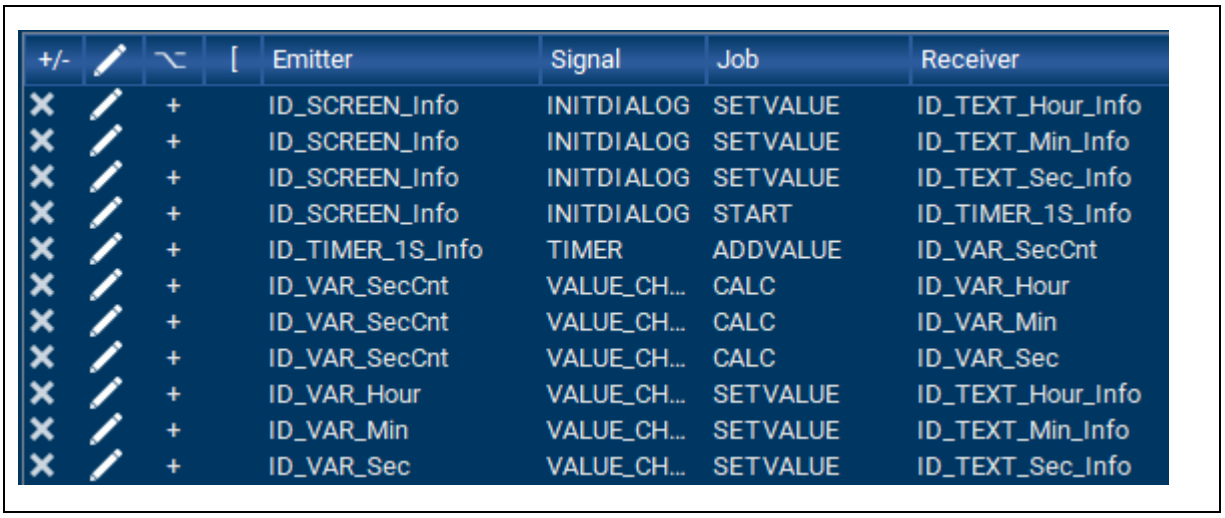


图 1-85 按“Return”键返回 Init 屏幕

2) 实时时钟显示。参考 1.2.3 章 2 节 2 步骤。



+/-	Emitter	Signal	Job	Receiver
+	ID_SCREEN_Info	INITDIALOG	SETVALUE	ID_TEXT_Hour_Info
+	ID_SCREEN_Info	INITDIALOG	SETVALUE	ID_TEXT_Min_Info
+	ID_SCREEN_Info	INITDIALOG	SETVALUE	ID_TEXT_Sec_Info
+	ID_SCREEN_Info	INITDIALOG	START	ID_TIMER_1S_Info
+	ID_TIMER_1S_Info	TIMER	ADDVALUE	ID_VAR_SecCnt
+	ID_VAR_SecCnt	VALUE_CH...	CALC	ID_VAR_Hour
+	ID_VAR_SecCnt	VALUE_CH...	CALC	ID_VAR_Min
+	ID_VAR_SecCnt	VALUE_CH...	CALC	ID_VAR_Sec
+	ID_VAR_Hour	VALUE_CH...	SETVALUE	ID_TEXT_Hour_Info
+	ID_VAR_Min	VALUE_CH...	SETVALUE	ID_TEXT_Min_Info
+	ID_VAR_Sec	VALUE_CH...	SETVALUE	ID_TEXT_Sec_Info

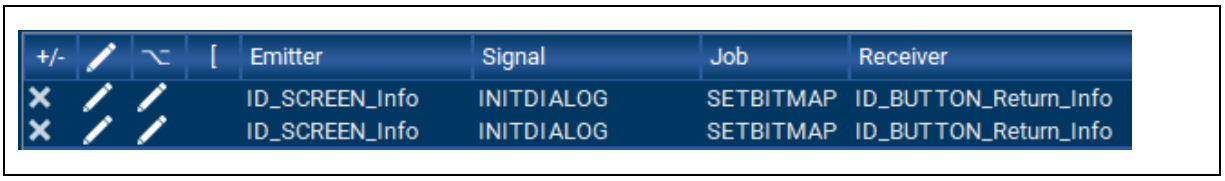
图 1-86 实时时钟显示

3) 枪号初始化。

+/-	Emitter	Signal	Job	Receiver
+	ID_SCREEN_Info	INITDIALOG	SETVALUE	ID_TEXT_Gun1_Info
+	ID_SCREEN_Info	INITDIALOG	SETVALUE	ID_TEXT_Gun2_Info
+	ID_SCREEN_Info	INITDIALOG	SETVALUE	ID_TEXT_Gun3_Info

图 1-87 枪号初始化

4) 加载 Info 屏幕，根据“Lan”变量的值更新“Return”键图案。参考 1.2.3 章 1 节 3 步骤。

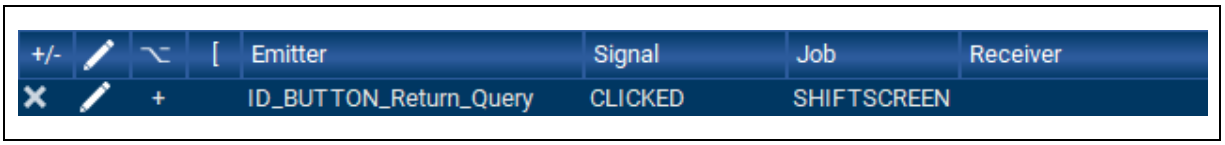


+/-	Emitter	Signal	Job	Receiver
	ID_SCREEN_Info	INITDIALOG	SETBITMAP	ID_BUTTON_Return_Info
	ID_SCREEN_Info	INITDIALOG	SETBITMAP	ID_BUTTON_Return_Info

图 1-88 加载 Info 屏幕更新”Return”键图案

6. Query 屏幕交互设计。包括按“Return”键返回 Init 屏幕；显示实时时钟，结合代码显示用户名；上电后直到充电完成才会显示最近一次的充电信息。

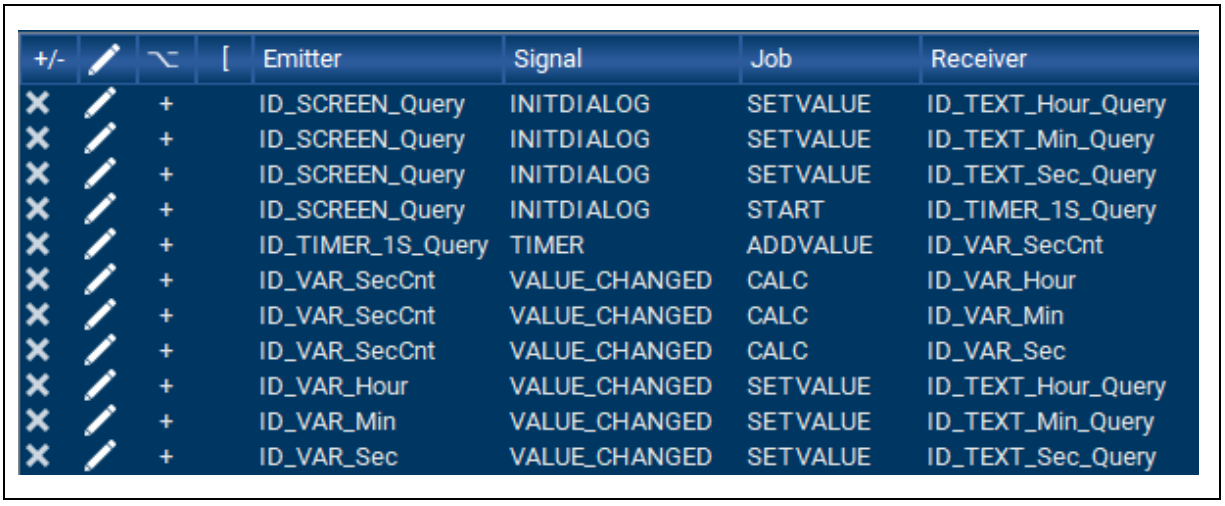
1) 按“Return”键返回 Init 屏幕。参考 1.2.3 章 1 节 1 步骤。



+/-		⌥	[	Emitter	Signal	Job	Receiver
✕	✎	+		ID_BUTTON_Return_Query	CLICKED	SHIFTSCREEN	

图 1-89 按“Return”键返回 Init 屏幕

2) 实时时钟显示。参考 1.2.3 章 2 节 2 步骤。



+/-		⌥	[	Emitter	Signal	Job	Receiver
✕	✎	+		ID_SCREEN_Query	INITDIALOG	SETVALUE	ID_TEXT_Hour_Query
✕	✎	+		ID_SCREEN_Query	INITDIALOG	SETVALUE	ID_TEXT_Min_Query
✕	✎	+		ID_SCREEN_Query	INITDIALOG	SETVALUE	ID_TEXT_Sec_Query
✕	✎	+		ID_SCREEN_Query	INITDIALOG	START	ID_TIMER_1S_Query
✕	✎	+		ID_TIMER_1S_Query	TIMER	ADDVALUE	ID_VAR_SecCnt
✕	✎	+		ID_VAR_SecCnt	VALUE_CHANGED	CALC	ID_VAR_Hour
✕	✎	+		ID_VAR_SecCnt	VALUE_CHANGED	CALC	ID_VAR_Min
✕	✎	+		ID_VAR_SecCnt	VALUE_CHANGED	CALC	ID_VAR_Sec
✕	✎	+		ID_VAR_Hour	VALUE_CHANGED	SETVALUE	ID_TEXT_Hour_Query
✕	✎	+		ID_VAR_Min	VALUE_CHANGED	SETVALUE	ID_TEXT_Min_Query
✕	✎	+		ID_VAR_Sec	VALUE_CHANGED	SETVALUE	ID_TEXT_Sec_Query

图 1-90 实时时钟显示

3) 屏幕加载窗口显示。充电完成后才显示窗口。

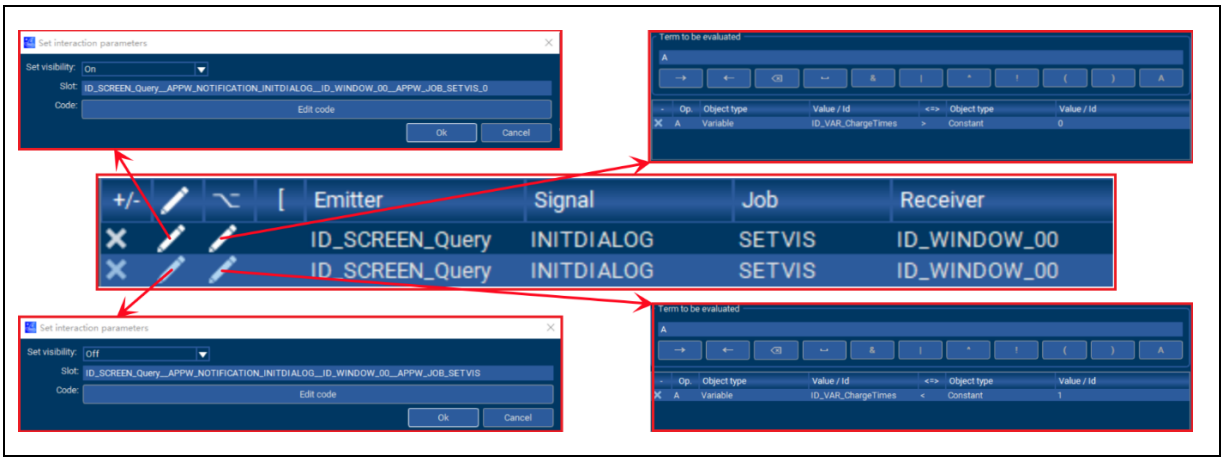


图 1-91 屏幕加载窗口显示

4) 定义时、分、秒变量及“SecCnt”备份变量、“SumCost”备份变量。

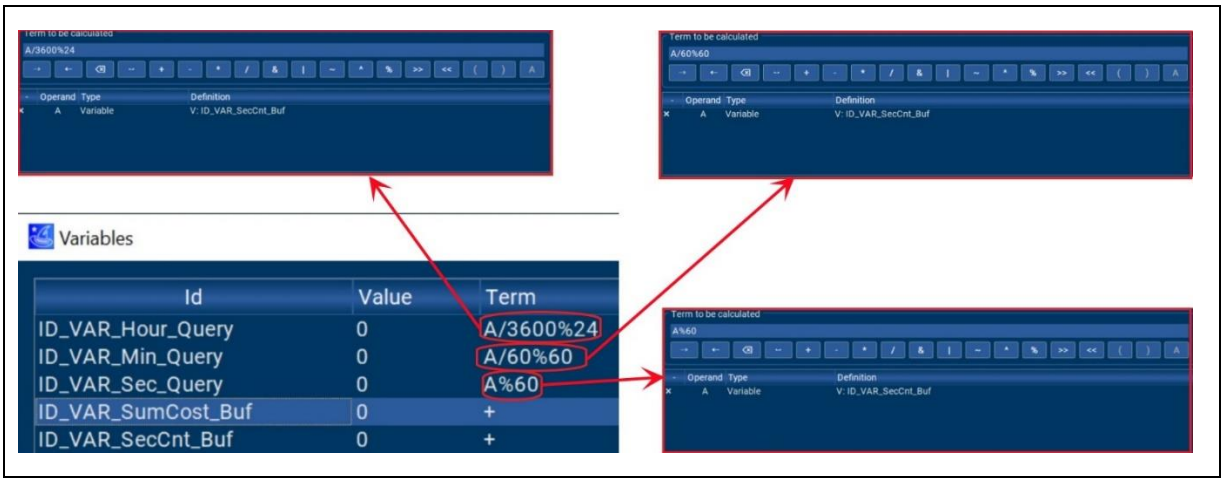


图 1-92 时、分、秒变量及“SecCnt”备份变量、“SumCost”备份变量定义

5) 显示最近一次充电信息，包括时间、费用、充电站、枪号。

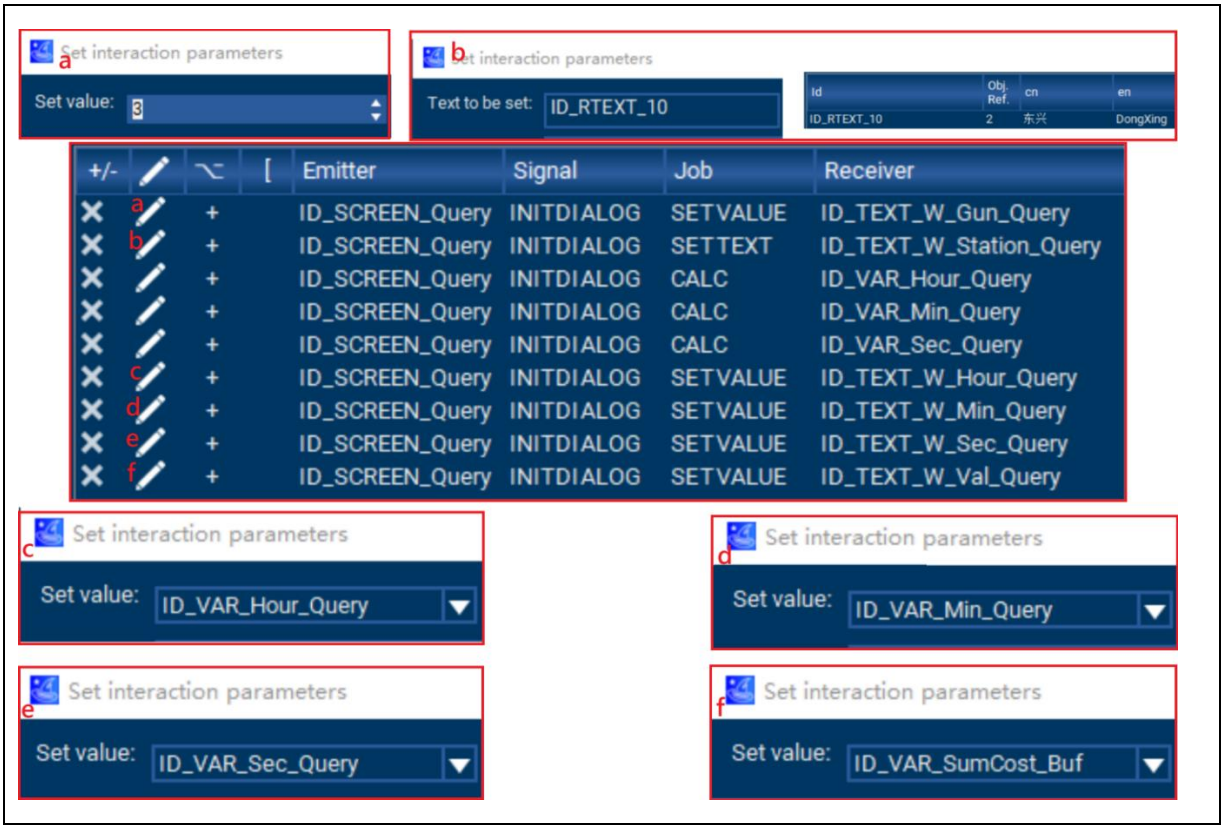


图 1-93 最近一次充电信息显示

6) 加载 Query 屏幕，根据“Lan”变量的值更新“Return”键图案。参考 1.2.3 章 1 节 3 步骤。

+/-		⌥	[	Emitter	Signal	Job	Receiver
✕				ID_SCREEN_Query	INITDIALOG	SETBITMAP	ID_BUTTON_Return_Query
✕				ID_SCREEN_Query	INITDIALOG	SETBITMAP	ID_BUTTON_Return_Query

图 1-94 加载 Query 屏幕更新“Return”键图案

7. Account 屏幕交互设计。包括按“Return”键返回 Init 屏幕；结合代码显示用户名，显示实时时钟；显示账户姓名、余额、电话号码。

1) 按“Return”键返回 Init 屏幕。参考 1.2.3 章 1 节 1 步骤。

+/-		⌥	[	Emitter	Signal	Job	Receiver
✕		+		ID_BUTTON_Return_Acc	CLICKED	SHIFTSCREEN	

图 1-95 按“Return”键返回 Init 屏幕

2) 实时时钟显示。参考 1.2.3 章 2 节 2 步骤。

+/-		⌥	[	Emitter	Signal	Job	Receiver
✕		+		ID_SCREEN_Account	INITDIALOG	SETVALUE	ID_TEXT_Hour_Acc
✕		+		ID_SCREEN_Account	INITDIALOG	SETVALUE	ID_TEXT_Min_Acc
✕		+		ID_SCREEN_Account	INITDIALOG	SETVALUE	ID_TEXT_Sec_Acc
✕		+		ID_SCREEN_Account	INITDIALOG	START	ID_TIMER_1S_Acc
✕		+		ID_TIMER_1S_Acc	TIMER	ADDVALUE	ID_VAR_SecCnt
✕		+		ID_VAR_SecCnt	VALUE_CHANGED	CALC	ID_VAR_Hour
✕		+		ID_VAR_SecCnt	VALUE_CHANGED	CALC	ID_VAR_Min
✕		+		ID_VAR_SecCnt	VALUE_CHANGED	CALC	ID_VAR_Sec
✕		+		ID_VAR_Hour	VALUE_CHANGED	SETVALUE	ID_TEXT_Hour_Acc
✕		+		ID_VAR_Min	VALUE_CHANGED	SETVALUE	ID_TEXT_Min_Acc
✕		+		ID_VAR_Sec	VALUE_CHANGED	SETVALUE	ID_TEXT_Sec_Acc

图 1-96 实时时钟显示

3) 定义余额变量。初始值为 1000。

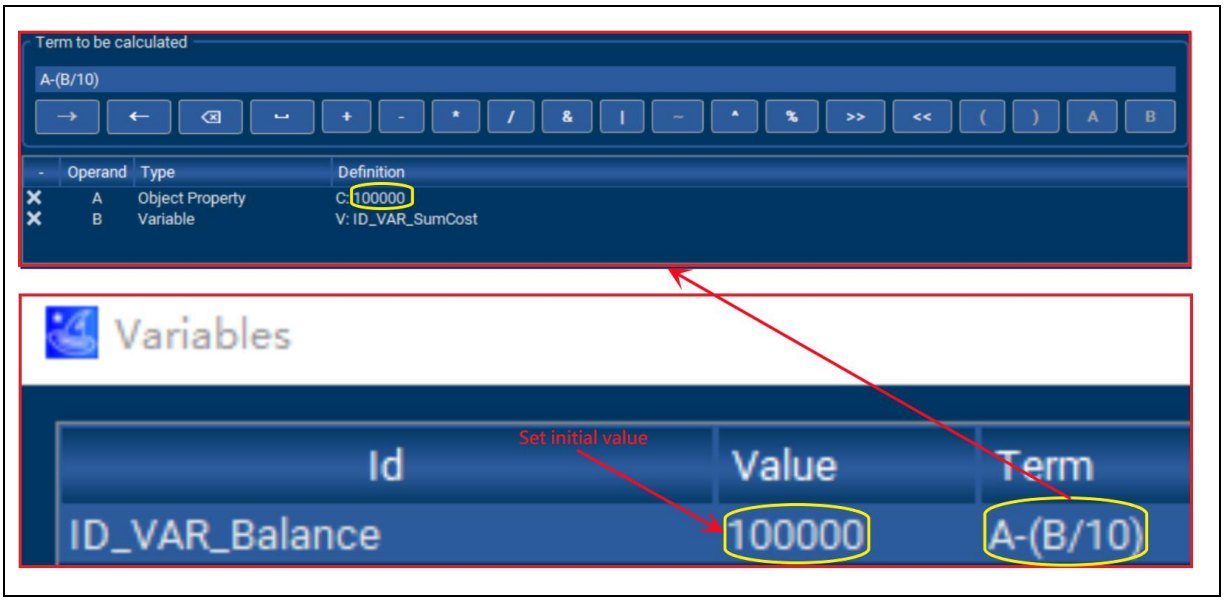


图 1-97 余额变量定义

4) 余额显示。

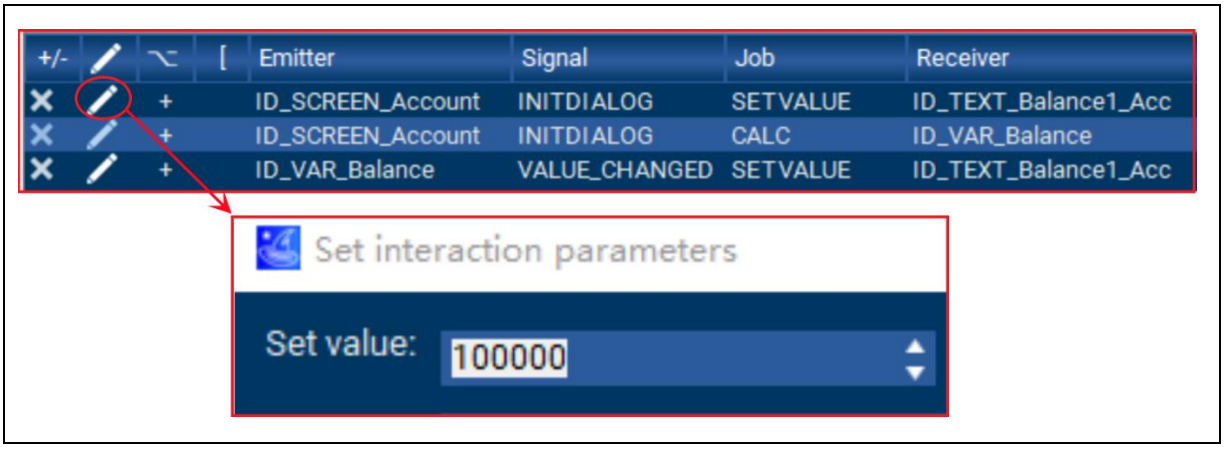


图 1-98 余额显示

5) 加载 Account 屏幕，根据 “Lan”变量的值更新“Return”键图案。参考 1.2.3 章 1 节 3 步骤。

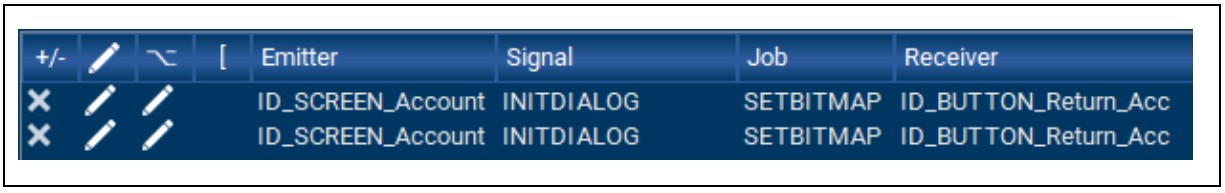


图 1-99 余额显示

1.2.4 程序代码编辑

1. 设计完成即可于 File-> Export & Save 输出程序代码，如下图。此范例所使用之项目文件名称为 EC_ChargePile.zip，档案路径可参考图 4-1，在解压缩后，可以使用 AppWizard 直接开启该项目档。

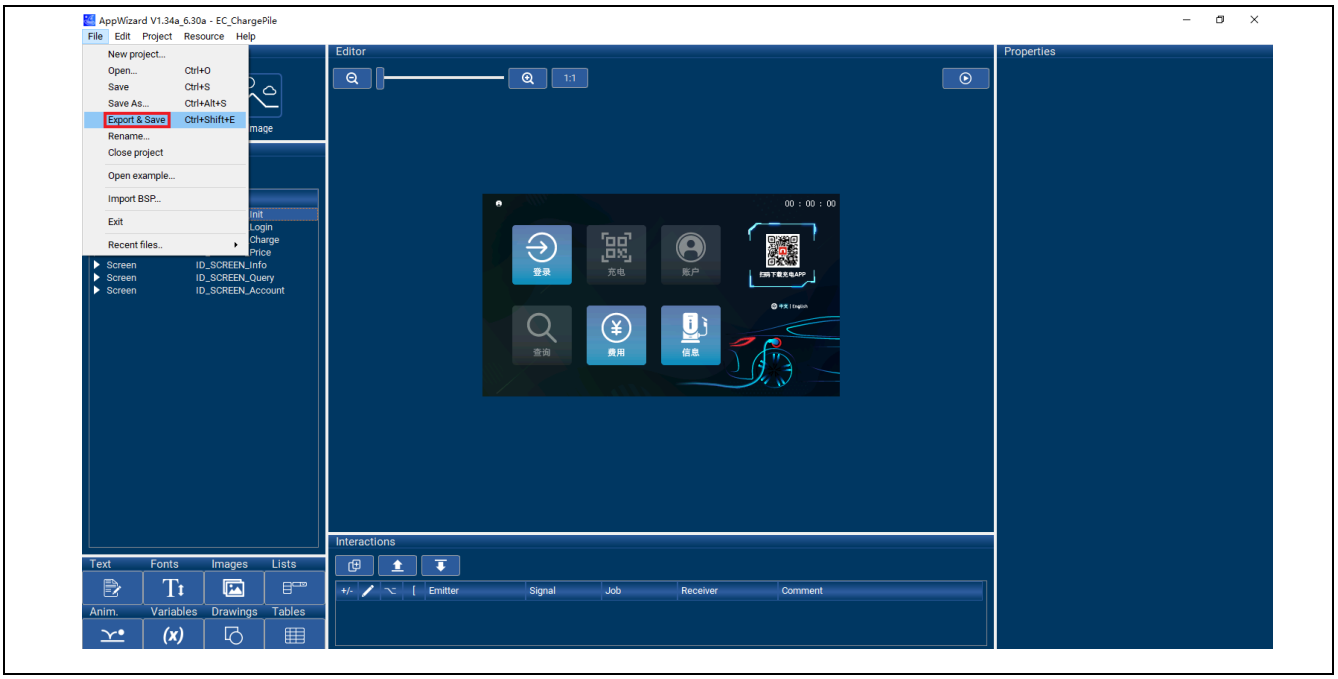


图 1-100 输出程序代码

2. 于 Project -> Open containing folder 开启 AppWizard 专案文件夹，复制 Resource 与 Source 文件夹，于路径：EC_M467_AppWizard_ChargePile_V1.00\SampleCode\ExampleCode\M467_AppWizard_ChargePile\Applicationn 中覆盖原本档案，步骤如下 2 图所示。

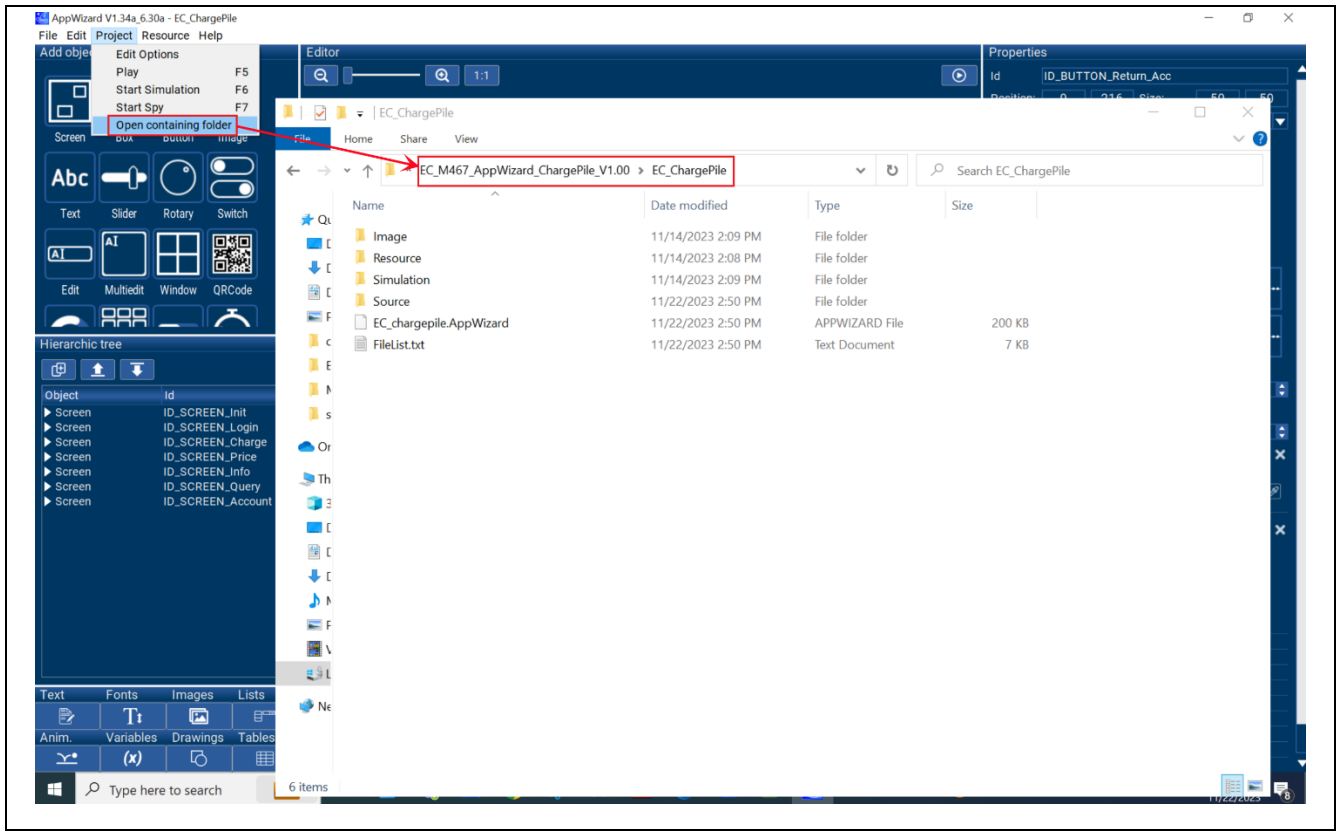


图 1-101 开启专案文件夹

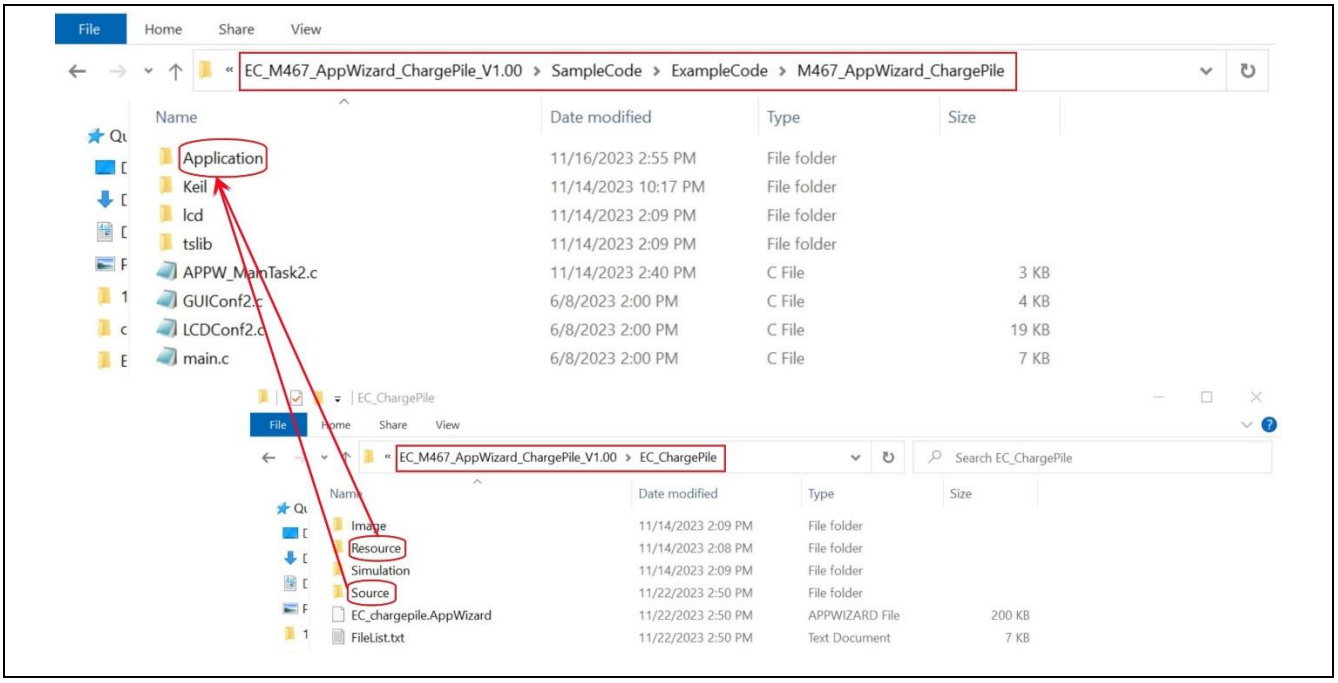


图 1-102 移植至程序代码项目中

3. 主程序代码请参考 2 代码介绍。

1.2.5 烧录运行在装置中

实际运行在装置后，仍可对整体执行效能做调整，相关优化设定如下所述

1. 于 AppWizard 使用的图片 format，需设定为符合屏幕格式，此范例设定为 “High color with alpha (565)”或 “Auto”。

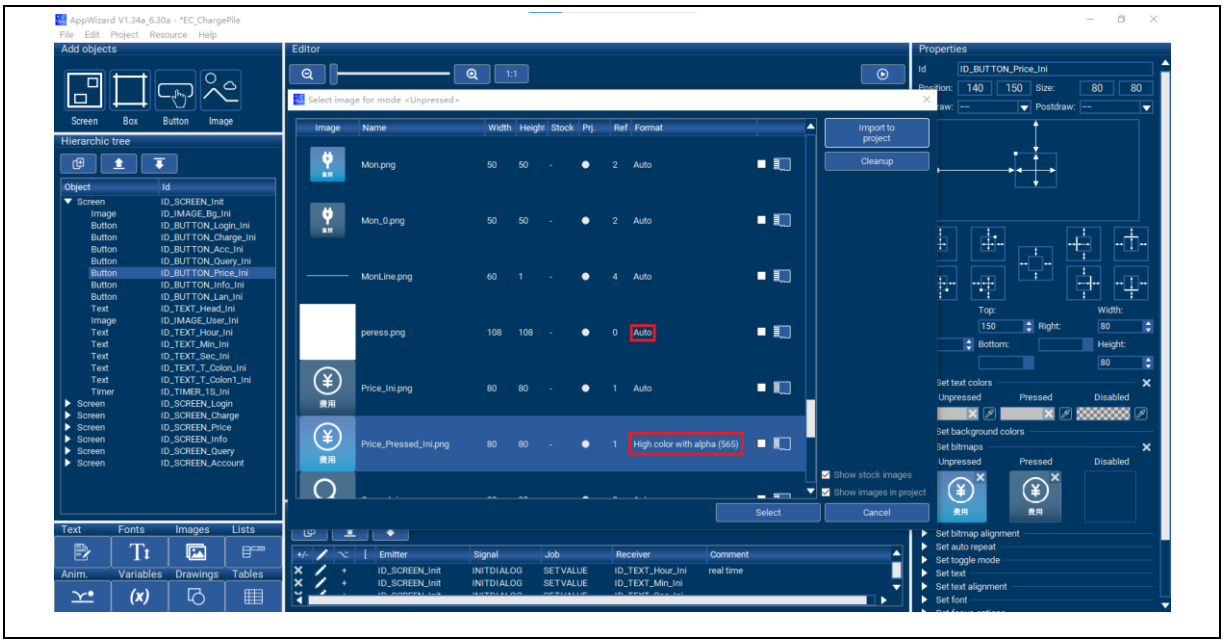


图 1-103 Image 设定

1.3 执行结果

本范例代码执行编译和刻录的结果如下图所示。

图 1-104 执行结果

2. 代码介绍

主程序由 *main.c* 进入，包括 3 个部分如下：

- 系统时钟与引脚初始化：sys_init
- 界面初始化与开启：UART、SPIM、Timer、Touch 等
- 应用主程序：MainTask2()

```
int main(void)
{
    // Initialize system clock
    sys_init();
    // Initialize debug port
    debug_port_init();
    SPIM_init();
    g_pu8FrameBufPtr  = (U8 *)((U32)g_u8FrameBuf);
    g_enable_Touch = 0;
    // Initialize timer
    tmr_init();
    // Startup CTP
    st1633i_low_level_init();
    st1633i_startup_sequence();

    printf("\r\n");
    printf("CPU @ %d Hz\r\n", SystemCoreClock);
    printf("+------------------------------------------------------------+\r\n");
    printf("|               emWin Simple Demo                            |\r\n");
    printf("+------------------------------------------------------------+\r\n");
    printf("\r\n");

    g_enable_Touch = 1;
    // Enable memory device before GUI_Init() to avoid flickering
    printf("Main Task2 -> \n");
    // Main Loop
    // MainTask2 for anti-flickering
    MainTask2();

    return 0;
}
```

其中，使用 SPIM 接口，是为了将占较大空间的图片、文本文件资源存至外部 SPI Flash，利用 SPI Synchronous Serial Interface Controller (SPI Master Mode)，在 Direct Memory Mapping Mode(DMM)的模式下，SPI Flash 可被视为 ROM Module，直接到指定地址读取 SPI Flash，并在内建的 cache 帮忙下，以及使用四线模式(quad read)，提升整体读取效率。

初始化 SPIM 的程序代码如下，需包括 SPIM 速度、SPI flash detection、开启 SPIM cache mode、开启 SPIM DMM mode、开启 fast quad read mode。相关参数设定可参考 W25Q32JVTBIQ 之说明。

```
void SPIM_init(void){
    uint8_t     idBuf[3];
    SYS_UnlockReg();                /* Unlock protected registers */
    SPIM_SET_CLOCK_DIVIDER(2);     /* Set SPIM clock as HCLK divided by 2, 200MHz/2 = 100*/

    SPIM_SET_RXCLKDLY_RDDLYSEL(0);     /* Insert 0 delay cycle. Adjust the sampling clock
of received data to latch the correct data. */
    SPIM_SET_RXCLKDLY_RDEDGE();        /* Use SPI input clock rising edge to sample
received data. */
    SPIM_SET_DCNUM(4);           /* 4 is dummy cycle for command 0xEB(fast quad read).*/

    if (SPIM_InitFlash(1) != 0)         /* Initialized SPI flash */
    {
        printf("SPIM flash initialize failed!\n");
    }

    SPIM_ReadJedecId(idBuf, sizeof (idBuf), 1);
    printf("SPIM get JEDEC ID=0x%02X, 0x%02X, 0x%02X\n", idBuf[0], idBuf[1], idBuf[2]);

    SPIM_DISABLE_CCM();
    SPIM_ENABLE_CACHE();

    if (SPIM_CIPHER_ON)
        SPIM_ENABLE_CIPHER();
    else
        SPIM_DISABLE_CIPHER();

    if (SPIM_Enable_4Bytes_Mode(USE_4_BYTES_MODE, 1) != 0)
    {
        printf("SPIM_Enable_4Bytes_Mode failed!\n");
    }

    SPIM->CTL1 |= SPIM_CTL1_CDINVAL_Msk;         // invalid cache
    SPIM_SetQuadEnable(1, 1);   //set quad mode bit
    SPIM_EnterDirectMapMode(USE_4_BYTES_MODE, CMD_DMA_FAST_QUAD_READ, 0);

}
```

除了 SPIM 初始化，亦须编辑 Scatter file，以让编译程序知道 SPI Flash 的位置与内容。于 KEIL 环境中，依照路径 Options for Target-> Linker->Scatter File-> Edit，开启.sct 档案并编辑内容如下。

```
;APROM
LOAD_ROM_1  0x0
{
    APROM.bin  0x0
    {
     startup_M460.o (RESET, +FIRST)
     *(+RO)
    }
```

```
    SRAM  0x20000000 0x80000
    {
     * (+RW, +ZI)
    }
}

;SPI flash
LOAD_ROM_2  0x100000
{
    SPIM.bin  0x100000
    {
     MicrosoftJhengHeiUI_16_Bold_EXT_AA4.o (+RO)
     MicrosoftJhengHeiUI_24_Bold_EXT_AA4.o (+RO)
     Account_Lock_Ini.o (+RO)
     Account_Pressed_Ini.o (+RO)
     Account_Lock_Ini_En.o (+RO)
     Account_Pressed_Ini_En.o (+RO)
     Bg_Cha.o (+RO)
     Bg2.o (+RO)
     Bg2_En.o (+RO)
     Bg3.o (+RO)
     Charge_Lock_Ini.o (+RO)
     Charge_Pressed_Ini.o (+RO)
     Charge_Lock_Ini_En.o (+RO)
     Charge_Pressed_Ini_En.o (+RO)
     che0.o (+RO)
     che1.o (+RO)
     che2.o (+RO)
     che3.o (+RO)
     che4.o (+RO)
     che5.o (+RO)
     Connect.o (+RO)
     Cost.o (+RO)
     Cost_0.o (+RO)
     Cost_En.o (+RO)
     Cost_0_En.o (+RO)
     Device.o (+RO)
     Device_0.o (+RO)
     Device_En.o (+RO)
     Device_0_En.o (+RO)
     ExpenseLine.o (+RO)
     Head_Ini.o (+RO)
     Info_Pressed_Ini.o (+RO)
     Info_Pressed_Ini_En.o (+RO)
     Lan_Cn.o (+RO)
     Lan_En.o (+RO)
     Login_Pressed_Ini.o (+RO)
     Login_Pressed_Ini_En.o (+RO)
     Mon.o (+RO)
     Mon_0.o (+RO)
     Mon_En.o (+RO)
     Mon_0_En.o (+RO)
     MonLine.o (+RO)
     Price_Pressed_Ini.o (+RO)
     Price_Pressed_Ini_En.o (+RO)
     Query_Lock_Ini.o (+RO)
```

```
    Query_Lock_Ini_En.o (+RO)
    Query_Pressed_Ini.o (+RO)
    Query_Pressed_Ini_En.o (+RO)
    Return0.o (+RO)
    Return1.o (+RO)
    Return0_En.o (+RO)
    Return1_En.o (+RO)
    Start.o (+RO)
    Stop_Pressed.o (+RO)
   }
}
```

程序进入应用主程序 MainTask2()后，将执行 AppWizard 相关函数，在 while 循环中会执行画图的动作，在此范例程序使用 APPW_SetVarData()函数，将值从 MCU 传出给 AppWizard 中的 Variable；使用 APPW_GetText()函数，获取 AppWizard 中的 Text 值；使用 APPW_SetText()函数，将字符从 MCU 传给 AppWizard 中的 Text 变量。通过 1.2.3 章 1 节 5 步骤所述之方法，实现从 AppWizard 中 Login 屏幕输入用户名、密码，代码对输入密码进行检查。如果密码正确，首先设置“PW_1”变量为 1，然后设置字符“OK”显示在 Login 屏幕“PW_Sta”物件上，最后设置用户名显示在 Login、Init、Charge、Account、Query 屏幕的“Head”物件上；如果密码错误，首先设置“PW_1”变量设置为 0，然后设置字符“ERR”显示在 Login 屏幕的“PW_Sta”物件。

```
void MainTask2(void)
{
    volatile int i=0;
    volatile int j=0;
    char UserName[20];
    char PW_Val[8] = {'q','q','q','q','q','q','q','q'};
    char PW_OK[3] = {'O','K',' '};
    char PW_ERR[3] = {'E','R','R'};
    char PW_Rev[9];
   // Setup configuration dependent pointers
   APPW_X_Setup();
   // Initialize AppWizard
   APPW_Init(APPW_PROJECT_PATH);
   // Create all persistent screens except initial screen
   APPW_CreatePersistentScreens();
   // Create initial screen...
   APPW_CreateRoot(APPW_INITIAL_SCREEN, WM_HBKWIN);
   // ...and keep it alive
    while (1)
    {
      // Drawing a frame
      GUI_Exec1();
      APPW_Exec();
      NVT_Draw1();
         i = APPW_GetText(ID_SCREEN_Login,ID_EDIT_UserName_Log,UserName,8);
         i = APPW_GetText(ID_SCREEN_Login,ID_EDIT_Password_Log,PW_Rev,9);

         for(i = 0;i < 8;i++)
         {
             if(PW_Rev[i] != PW_Val[i])
```

```
            {
                APPW_SetVarData(ID_VAR_PW_1, 0);
                APPW_SetText(ID_SCREEN_Login,ID_TEXT_PW_Sta,PW_ERR);
                break;
            }
        }
        if(i == 8)
        {
            APPW_SetVarData(ID_VAR_PW_1, 1);
            APPW_SetText(ID_SCREEN_Login,ID_TEXT_PW_Sta,PW_OK);
            APPW_SetText(ID_SCREEN_Init,ID_TEXT_Head_Ini,UserName);
            APPW_SetText(ID_SCREEN_Charge,ID_TEXT_Head_Cha,UserName);
            APPW_SetText(ID_SCREEN_Account,ID_TEXT_Head_Acc,UserName);
            APPW_SetText(ID_SCREEN_Query,ID_TEXT_Head_Query,UserName);
        }
    }
}
```

AppWizard 执行的画图内容，会从 ID_Screen_Init_Slots.c 中读取。在 1.2.3 章 2 节 7 步骤中，“Lan”键按下的程序代码中加入 printf("Button Click\n")，那么当 Lan 键按下，UART 会于 PC 印出”Button Click”内容。

```
/*
File        : ID_SCREEN_Init_Slots.c
Purpose     : AppWizard managed file, function content could be changed------------------
--------END-OF-HEADER-----------------------------
*/

#include "Application.h"
#include "../Generated/Resource.h"
#include "../Generated/ID_SCREEN_Init.h"
/*** Begin of user code area ***/
/*** End of user code area ***/

/*********************************************************************
*
*       Public code
*
**********************************************************************
*/
/*********************************************************************
*
*       cbID_SCREEN_Init
*/
void cbID_SCREEN_Init(WM_MESSAGE * pMsg) {
  GUI_USE_PARA(pMsg);
}
/*********************************************************************
*
*       ID_SCREEN_Init__ID_BUTTON_Lan_Ini__WM_NOTIFICATION_CLICKED_0
*/
void ID_SCREEN_Init__ID_BUTTON_Lan_Ini__WM_NOTIFICATION_CLICKED_0(APPW_ACTION_ITEM *
pAction, WM_HWIN hScreen, WM_MESSAGE * pMsg, int * pResult) {
  GUI_USE_PARA(pAction);
  GUI_USE_PARA(hScreen);
  GUI_USE_PARA(pMsg);
  GUI_USE_PARA(pResult);
  printf("Button Click\n");

}
```

3. 软件与硬件需求

3.1 软件需求

- BSP 版本
  - M460_Series_BSP_CMSIS_V3.00.002
- emWin Software Package for M460 版本
  - emWin Software Package for M460 V3.0
- IDE 版本
  - Keil uVersion 5.38.0.0
- AppWizard 版本
  - AppWizard V1.34a_6.30a

3.2 硬件需求

- 电路组件
  - NuMaker-M467HJ V1.0
- 示意图
  - 如图 3-1，将 Nu-Link2-Me 与子板之屏幕 USB connector 以 2 条 USB 线与 PC 相连

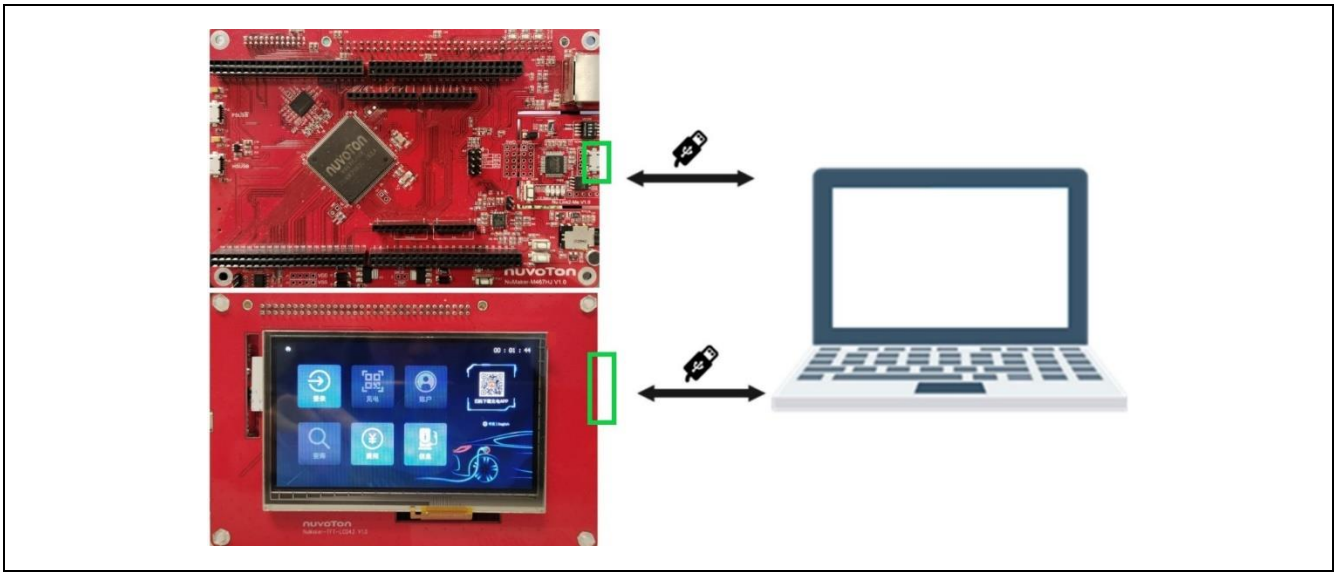


图 3-1 以 USB 线将 NuMaker-M467HJ V1.0 与 PC 相连

4. 目录信息

EC_M467_AppWizard_ChargePile_V1.00	
Library	Sample code header and source files
CMSIS	Cortex® Microcontroller Software Interface Standard (CMSIS) by Arm® Corp.
Device	CMSIS compliant device header file
StdDriver	All peripheral driver header and source files
SampleCode	
ExampleCode	
Project	Source file of example code
ThirdParty	Header files for emWin project
EC_ChargePile.zip	ChargePile sample project of AppWizard. Please open by AppWizard.exe.

图 4-1 目录资讯

5. 范例程序执行

1. 根据目录信息章节进入 ExampleCode 路径中的 KEIL 文件夹，双击 *M467_AppWizard_ChargePile.uvprojx*。
2. 进入编译模式接口
   - 编译
   - 下载代码至内存
   - 进入 / 离开除错模式
3. 进入除错模式接口
   - 执行代码

6. 修订纪录

Date	Revision	Description
2023.12.07	1.00	初始发布。

Important Notice

Nuvoton Products are neither intended nor warranted for usage in systems or equipment, any malfunction or failure of which may cause loss of human life, bodily injury or severe property damage. Such applications are deemed, "Insecure Usage".

Insecure usage includes, but is not limited to: equipment for surgical implementation, atomic energy control instruments, airplane or spaceship instruments, the control or operation of dynamic, brake or safety systems designed for vehicular use, traffic signal instruments, all types of safety devices, and other applications intended to support or sustain life.

All Insecure Usage shall be made at customer's risk, and in the event that third parties lay claims to Nuvoton as a result of customer's Insecure Usage, customer shall indemnify the damages and liabilities thus incurred by Nuvoton.

Please note that all data and specifications are subject to change without notice.
All the trademarks of products and companies mentioned in this datasheet belong to their respective owners.