可编程智能 TFT LCD 软件开发指南

Programmable Smart TFT LCD Software User Guide

Version 1.18

北京大器智成技术有限公司 2012

Beijing Greatal Technologies Co. Ltd.

Change History

Version	Date	Content	Owner
V1.00	20100523	Initial	QuickDevcie
V1.01	20100608	Update Hyperterminal setting	QuickDevice
V1.02	20100727	Update CTP protocol	QuickDevice
V1.03	20100920	Support JavaScript	QuickDevice
V1.04	20100929	Rewrite module description	QuickDevice
V1.05	20110121	Update company info	Greatal
V1.06	20110817	Update Designer guide	Greatal
V1.07	20110909	Update Designer Property guide	Greatal
V1.08	20110910	Introduce Designer Chinese property	Greatal
V1.09	20110923	Support beep function	Greatal
V1.10	20111020	Introduce common widget properties	Greatal
V1.11	20111031	Introduce global.js and Timer widget	Greatal
V1.12	20111120	Introduce PS-LCD emulator	Greatal
V1.13	20111202	Introduce style sheet	Greatal
V1.14	20120103	Update Designer usage	Greatal
V1.15	20120224	Introduce UserDefine protocol	Greatal
V1.16	20120308	Introduce usrRxByte() function	Greatal
V1.17	20120313	Comparison between CTP & usrdefine	Greatal
V1.18	20120329	Add more FAQs	Greatal

术 语

PS-LCD: Programmable **S**mart **LCD** 缩写，可编程智能 LCD 模组

GUI: Graphical User Interface 的缩写，图形界面

HMI: Human Machine Interface 的缩写，人机界面

Cooky: 大器智成开发的 LCD 图形界面卡（HMI Processing Module）

HC: Host Controller 的缩写，主控制器

CTP： Cooky Talk Protocol 的缩写，HC 与 PS-LCD 的通讯协议

Designer: Windows 下可视化组态编程软件，用于编辑和生成人机界面

SPF： Super Parser File 的缩写，Designer 的输出界面文件，可下载到 PS-LCD 运行

目　录

第一章 简　介

1.1 什么是 PS-LCD

可编程智能 LCD（即 Programmable Smart LCD，简称 PS-LCD）是一种包含 LCD 显示屏、LCD 控制器、触摸屏、人机界面处理系统和通讯接口于一体的智能显示模组，通过可选的通讯接口与外部控制单元（如：51 单片机、ARM、DSP、PC、PLC、总线设备等）连接，实现系统的人机交互界面。配合专用的可视化组态编辑软件，用户无需专业图形编程知识，”0”代码即可轻松设计和生成系统所需图形界面。

1.2 基于 PS-LCD 的产品硬件

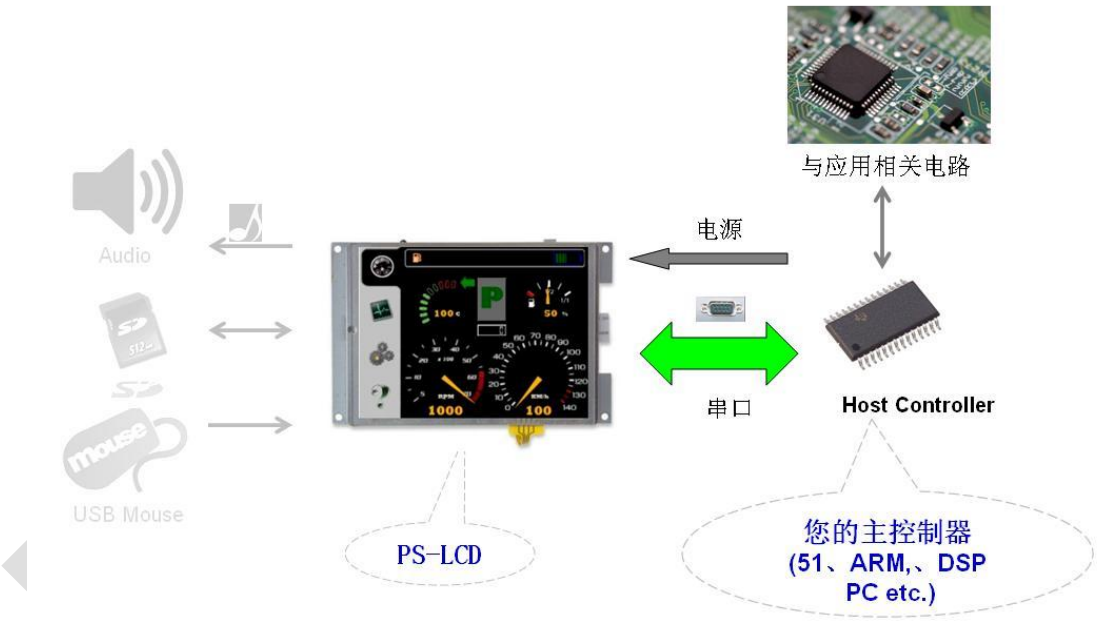


图 1-1 基于 PS-LCD 的产品硬件示意图

典型的产品硬件示意图见图 1-1，一般由三部分组成：

- PS-LCD
- 主控制器
- 相关应用电路

PS-LCD 硬件通讯接口简单，任何具有串口（或者 RS232、485、CAN 等总线接口）的控制单元都能轻松与之连接。

1.3 基于 PS-LCD 的产品软件

基于 PS-LCD 设计人机界面产品，无需编程，动几下鼠标，几分钟即可设计出如下界面效果，如同制作 PPT 一样简单、直观！目前支持十多种常用控件（如文本框、按钮、进度条、仪表盘、曲线图等），可满足大部分应用需求。

图 1-2 PS-LCD 人机界面效果

与传统模式相比，有如下优势：

- **对控制单元性能要求低**，只要有通讯接口（如三线串口）的控制单元均可与 PS-LCD 连接。
- **开发难度低**，无需图形编程知识，用可视化组态式编辑软件，“所见即所得”、“0”代码设计和生成界面；通过 PS-LCD 软件模拟器，在没有硬件情况下，可在 PC 上模拟实际界面运行效果，有效缩短界面开发周期；
- **占用控制单元资源少**，控制单元只需与 PS-LCD 交互应用相关数据即可，界面显示、触摸动作等行为均由 PS-LCD 按预设功能自动完成；
- **软件代码少**，控制单元只需三条指令或者函数与 PS-LCD 通信，快速集成界面；

第二章 开发概述

2.1 开发步骤

通过大器智成专用可视化集成开发工具 Designer 轻松完成 PS-LCD 图形界面开发。该工具包含了开发所需的界面管理、配置、编辑、生成、模拟、下载等一系列软件功能，用户无需界面编程知识，即使无任何软件开发基础的用户也可快速完成产品界面开发，最大限度降低开发难度和工作量。

PS-LCD 界面开发过程可总结为如下三步：

1) 在 Windows 上用 Designer “所见即所得”、“0”代码生成界面,；

2) 通过 USB 快速下载步骤一的输出文件 spf 到 PS-LCD，然后重启 PS-LCD，界面即可运行；

3) 界面运行时，外部控制单元通过简单协议与界面通讯，快速集成产品。

图 2-1 基于 PS-LCD 的界面开发步骤

2.2 PS-LCD 工作模式

PS-LCD 可工作在两种模式，分别是：

- **下载模式**

在该模式下，采用大器智成专用软件工具 Flex 通过 USB 与 PS-LCD 连接，即可下载界面文件 spf，如图 2-2。

图 2-2 通过 Flex 下载界面文件到 PS-LCD 示意图

进入下载模式的方法：将 PS-LCD 背面电路板上的模式跳线短接(或者按住下载按钮)，然后上电或者复位，即进入下载模式。此时，PS-LCD 底板指示灯“两快一暗”闪烁，LCD 屏上出现下载图标。

界面模式

将 PS-LCD 背面电路板上的模式跳线断开(或者松开下载按钮)，然后上电或者复位，即进入界面模式，主界面完全启动并显示大概需要 10 秒。此时，指示灯闪烁速度每秒一次。

在界面模式下，PS-LCD 主要完成：

1) 读取自定义 logo 图片并在启动过程中显示；

2) 待启动完成后，运行预先设计的人机界面；

3) 在人机界面运行过程中，外部控制单元可与人机界面通讯；

2.3 让 PS-LCD 运行我的界面

将 Designer 生成的界面文件 spf 下载到 PS-LCD，即可完成界面更新，步骤

如下：

1) 进入下载模式；

2) 启动 Flex，点击 path 按钮，选择 spf 文件（扩展名为.spf），点击 Start 按钮，下载开始；

3) 待系统提示下载完成，点击 OK；

4) 重新启动 PS-LCD，进入界面模式，人机图形界面即开始运行。

2.4 指定启动界面 logo

可任意选择 24 bit 的位图图片（图片分辨率需要小于实际 LCD 分辨率）作为系统启动 logo，方法如下：

1) 进入下载模式；

2) 启动 Flex，点击 path 按钮，选择自定义的 logo 图片（扩展名为.bmp），点击 start 按钮，下载开始；

3) 待系统提示下载完成，点击 Ok；

4) 重新启动 PS-LCD，进入界面模式，即可看到步骤 2 下载的图片生效。

第三章 设计界面

3.1 Designer 介绍

Designer 是一个 Windows 端的可视化界面集成开发环境，使用本工具可配置 PS-LCD 界面运行参数、设计界面静态外观（如外观，字体，和尺寸等），定义动态行为，“所见即所得“、“0”代码快速完成最终产品的人机界面。甚至无需 PS-LCD 硬件，通过 Designer 模拟器也可轻松完成界面效果验证。Designer 的功能区域划分如下：

图 3-1 Designer 功能区域示意图

菜单栏

文件->新建(打开) ：新建（或打开）一个项目工程；

编辑 ：编辑界面时的拷贝、粘帖、删除等命令；

工具->界面配置 ：界面运行参数设定，选中即弹出如下对话框。

桌面图片——界面切换过程中的桌面背景，不能与桌面颜色同时使用；

图 3-2 Designer 界面配置选项

桌面颜色——指定界面切换过程中显示颜色，不能与桌面图片同时使用；

桌面透明度——指定界面切换过程中显示桌面颜色的透明度，只有桌面颜色选定时有效，0 为不透明，10 为完全透明；

主窗体——指定界面的主窗体，界面启动后自动显示；

窗体尺寸——指定界面的像素尺寸，一般与实际使用的 PS-LCD 实际分辨率一致；

通讯协议——PS-LCD 与外部控制单元的通讯协议选择，可支持 CTP 和用户自定义 (UserDefine) 协议，PS-LCD 运行时只能选择其中一种作为通讯协议，默认为 CTP 协议；

通讯速率——PS-LCD 与外部控制单元的通讯速率选择，如果为串口设备，通讯格式是 8N1（8 个数据位、无校验位、一个停止位）；

触摸屏——选中该项，当界面在 PS-LCD 上运行时，按住界面空白处 5 秒以上，自动弹出触摸屏校准界面；

软键盘——选中该项，当界面在 PS-LCD 上运行时，点击文本框，软键盘自动从界面下部弹出；

调试模式——如果选中，生成的界面文件为调试(Debug)版本，界面运行过程中有任何 JS 脚本错误，自动弹出对话框提示开发人员；否则，为发

布(Release)版本，忽略一切 JS 脚本错误。一般在设计和调试界面阶段选择该选项，界面定型后应用于实际产品后，建议不选该项。

工具->全局脚本： 全局 JS 脚本，用于全局使用的变量和函数的定义。选中即弹出编辑器，可输入 JS 脚本，该脚本将在界面启动后最先运行一次，其中的所有定义变量和函数，在任何界面中可无条件访问。

工具->生成界面： 生成界面文件（扩展名为 spf，该文件可直接下载到 PS-LCD 上运行）；

工具->下载界面： 启动 Flex 专用界面文件下载工具，可通过 USB 下载 spf 界面文件到 PS-LCD；

工具->模拟器： 启动 PS-LCD 软件模拟器，并模拟界面在最终产品硬件上的运行效果和动态行为， 内嵌的串口模拟器可模拟实际串口收发数据；

选项->语言： 选择语言种类，目前支持简体中文和英文；

帮助->关于 Designer ：显示 Designer 版本信息；

帮助->检查更新 ：点击该选项，将访问大器智成软件发布页，查看最新版本软件信息；

图 3-3 PS-LCD 模拟器

工程管理区

浏览和管理界面文件。双击鼠标左键打开界面，右击鼠标，添加或者删除界面文件。

控件区

该区列出了所有控件名称，通过鼠标拖拽某一个控件到界面编辑区，界面中即生成该控件，自由调整尺寸和外观，“所见即所得”。目前支持如下控件类型，可支持控件数还在不断增加中。

- 复选框——提供某一个选项供选择时使用；
- 下拉框——提供若干种选择项，每次只能选其中一种；
- 图片——显示特定图片，通过源图片属性来指定需要显示的图片名称，支持 bmp、jpg、png、tiff 和 gif 格式图片；
- 标签——显示文本信息；
- 进度条——指示实时进度状况；
- 按钮——操作按钮，可定义成自动重复或者单次有效；
- 滑动尺——通过滑动块位置来定义输入的数值；
- 文本框——编辑和显示文本，可编辑。在 PS-LCD 中运行界面时，点击该文本框，系统自动弹出软键盘，可直接输入数字、字母、符号等信息；
- 定时器——提供定时功能。当定时时间间隔到时，可使其执行一段脚本，从而实现动画和刷新界面等效果，该控件在界面运行时不可见；
- 仪表盘——用于模拟机械仪表，仪表的背景图片通过源图片指定。更换不同的背景可设计出不同的仪表界面；
- 段式数字——模拟数码管显示数字；
- 波形图——显示曲线波形，可最多支持 4 条波形同时显示；
- 刻度尺——带刻度显示的位置指示器；

界面编辑区

用户界面的编辑区域，从控件区拖拽控件到该区域，然后用鼠标或键盘来完成控件的位置布局，可进行拖拉、重复、恢复、移动、删除、重定义尺寸等操作。选中某个控件，然后通过控件属性区来设置控件的各项属性(如背景，颜色，大小，文字、风格、边框等)和定义事件动态行为。

对象管理区

列出当前界面编辑区的所有控件对象列表，在这里可更改控件对象的名称 ID，该 ID 作为控件的唯一标识，在 PS-LCD 与主控制器通信，或者脚本编程中

使用。当在界面编辑区选中某个控件时，对象管理区的相应控件名称左侧会出现蓝色指示。

- **控件属性区**

列举出当前选中控件的所有属性，设置相关属性，可改变控件外观、风格和定义事件动作。所有控件的属性分为四个大类：

- 外观——定义控件尺寸，字体，颜色等外观特性；
- 内容——定义文字内容，图标，风格类型等；
- 边框——某些控件有该类属性，定义边框的效果；
- 行为——定义控件的动态效果和动作，其中如下属性定义了界面在 PS-LCD 上运行时的动态行为：

动作脚本(Action): 点击该属性按钮，即可在弹出的脚本编辑框中输入类 C 脚本语言（JavaScript）。当界面在 PS-LCD 上运行时，相应控件的动态属性发生变化，该脚本立即被执行，完成预设界面动作。

事件通知(Verbose):定义是否自动向外部控制单元发送事件消息。如选中，当相应控件的动态属性发生变化，PS-LCD 自动通过通讯口向外部控制单元发送事件消息（如：当界面在 PS-LCD 上运行时，用户点击按钮控件，PS-LCD 通过串口发送如下消息到外部控制单元: E+xx.clicked=1，其中 xx 为按钮控件名称）。

注意：*该属性只有选中 CTP 通讯协议时才有效*。

快速显示(Fastshow): 界面(Form)控件特有属性。如选中该属性，界面在切换过程中始终保持在内存运行，可最大限度地提高切换到该界面的速度。否则，界面在切换时需从 flash 上读入内存再运行。**注意：**如果过多界面设置该属性，有可能造成系统内存不足，界面运行速度缓慢。建议比较重要的界面才设置该属性。

3.2 设计步骤

基本步骤如下：

1) 新建 Designer 工程，定义界面分辨率、界面切换效果和主界面等；
2) 在工程中“所见即所得”生成界面，设置背景、加入/设置控件、指定图片/动画、定义事件动作等；

3) 用 LCD 模拟器验证界面效果和通讯过程，重复前面步骤直到满意为止。

4) 生成界面输出文件 spf，然后将 spf 文件通过 PS-LCD 专用软件工具 Flex 下载到 PS-LCD 验证最终界面效果。

具体设计过程参看第五章。

3.3 界面动态效果

Designer 不仅可设计界面静态外观、文字、风格类型等，而且也可通过控件的**动作脚本(action)**属性实现界面运行时动态效果。实现方法是：在 Designer 中，选中控件，然后点击控件属性区**动作脚本**按钮，在弹出的编辑框输入 JavaScript 脚本描述事件动作。

例如要实现下面动态效果：当按下界面按钮 xx，无需外部控制单元参与，让文本框 yy 内容自动刷新为”1234”。步骤是：在 Designer 中选中按钮 xx 对象，然后在属性区中点击**动作脚本**属性按钮，在弹出的编辑框中输入 JavaScript 脚本：yy.text=”1234”;，关闭编辑框即可。

注意：

- PS-LCD 仅支持 JavaScript 核心部分 ECMAScript（语法基本跟 C 一致，熟悉 C 的开发人员极容易掌握）。ECMAScript 的详细说明参看 http://www.w3school.com.cn/js/pro_js_syntax.asp 中的 ECMAScript 章节；
- 在 PS-LCD 上调试 JavaScript 脚本非常简单直观。如果脚本有问题，界面在 PS-LCD 或者模拟器上运行时，屏幕上会自动弹出错误提示框（需设置界面工程为调试版本），根据提示错误类型，脚本行号等信息，用户可轻松找到脚本错误所在，一目了然。
- 控件属性和方法可直接在 JavaScript 脚本代码中调用，具体用法参看“**PS-LCD 软件速查表**”文档。

第四章 界面通讯

4.1 PS-LCD 通讯协议

PS-LCD 作为先进的智能人机界面产品，能通过通讯接口轻松灵活地与外部控制单元实现数据交互。目前，PS-LCD 支持两种通讯协议：CTP(Cooky Talking Protocol)协议和**用户自定义**（UserDefine）协议，**在运行时只能选择其中一种**，需在 Designer 中的“界面配置”菜单中指定。两种协议的比较见下表：

	CTP 协议	**自定义协议**
数据发送 (PS-LCD 传输数据到外部控制单元)	分两种方式： 1） 在 designer 设计界面时选中该控件的“事件通知”属性，当该控件事件发生时，PS-LCD 自动发送数据消息，无需用户编写任何脚本；如：ID 为 xx 的按钮按下一次，PS-LCD 自动发送 xx.clicked=1 字符串消息； 2） 不选择“事件通知”属性，在控件的动作脚本中调用 ctpBinTx 或者 ctpAscTx 函数，实现自定义消息发送；	在界面/控件动作脚本中直接调用 usrTx 函数，发送任意自定义数据；与 CTP 协议下的发送方式 2 相同。
数据接收 (外部控制单元传输数据到 PS-LCD)	外部控制单元必须发送有效的 JS 脚本程序字符串，PS-LCD 接收后自动执行。如：外部控制单元发送 xxx.text=”test”字符串（后必须加回车字符），名字 ID 为 xxx 的控件文本自动刷性为 test。任何有效的 JS 脚本程序都能通过通讯接口以字符串方式发送给 PS-LCD 立即执行。	用户在全局脚本中先调用 usrRxSetHnd 函数注册接收函数名称，和触发字节数；然后实现接收回调函数，定义接收数据后界面的行为； 当 PS-LCD 接收到数据，自动放在系统接收缓冲区，当达到触发条件后，接收回调函数被执行，实现预定的动作行为，

4.2 CTP 协议

4.2.1 GUI 对象和属性

人机图形界面由无数控件组成，例如文本框，按钮，进度条等。把这些组成界面的控件实例称为 GUI 对象， 如名字为 myedit 的文本框。

每个 GUI 对象有唯一的名字，便于识别。另外还有一些特征参数（例如颜色，大小等），通过改变其值，可以改变对象的行为和状态，这些特征参数称为属性。GUI 对象的属性分为：

- 静态属性（Static Property 或 Design-time Property）：在 designer 设计阶段指定，运行时保持不变（例如按钮的风格和大小）。
- 动态属性 （Dynamic Property 或 Run-time Property）：既可以在 Designer 设计阶段指定，也可以在界面运行时改变其值（通过串口发送命令或者界面中嵌入的 JavaScript 脚本实现）。

4.2.2 定义

Designer 设计生成的界面，虽然没有主控制器参与能独立在 PS-LCD 上运行，但在很多应用场合，界面中的 GUI 对象需要与主控制器交互数据，界面才能与实际应用集成在一起。这种数据交互包括：

- **GUI 对象 -> 主控制器**

PS-LCD 自动检测控件对象 GUI 事件（例如按钮被点击），通过通讯接口发送事件消息告知主控制器。注意：**PS-LCD 默认不发送任何事件消息给主控制器，只有在 Designer 设计界面时，选中了相应控件的事件通知（Verbose）属性，才有此功能。**

- **主控制器 -> GUI 对象**

主控制器发送指令给 PS-LCD，控制/查询 GUI 对象状态（如文本框内容）。所有当前运行界面 GUI 对象的动态属性都能被主控制器无条件控制和查询。

CTP 协议就是为了满足上面的通讯要求而产生，主控制器通过 CTP 协议可简单直观地完成上述两种方向的数据交互，快速集成 Designer 所设计的界面。

定义：CTP 协议是实现主控制器与 PS-LCD 界面中的 GUI 对象通讯的软件协议，类似操作系统的 Shell，能够互动式地解释和执行用户系统传输的命令。该协议命令集为纯 ASCII 码，只包含三种格式，简单直观，易于理解。主控制器通过该协议与 PS-LCD 上的若干 GUI 对象通讯，能够直观快速地完成界面与实际应用的数据交互。通讯模型如图 4-1。

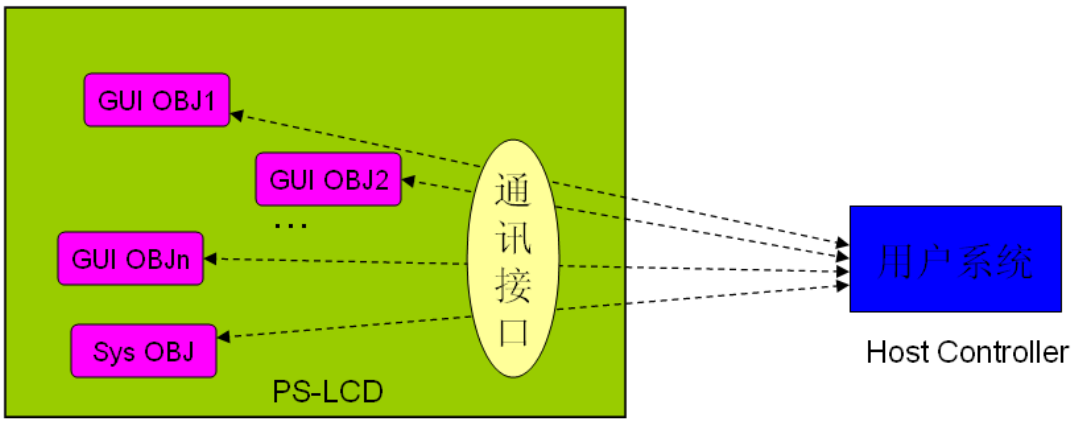


图 4-1 CTP 通讯协议模型

主控制器通过 CTP 协议可实现：

- ☑ 控制 GUI 对象的动态属性和 PS-LCD 系统工作状态；
- ☑ 查询 GUI 对象的动态属性；
- ☑ 获得 GUI 对象的事件通知；

4.2.3 格式

PS-LCD 的 CTP 通讯协议完全为 ASCII 码（除中文字符外），包括三种类型：

- **控制(Control)**

控制 GUI 对象动态属性，格式：

HC → PS-LCD	[object name].[runtime property name] = xxx<CR>	如： e1.text= “abcd\r”
PS-LCD → HC	C+<CR> C-<CR> C?<CR> C!<CR>	成功 失败 命令格式错误 GUI 对象名字错误

- **查询(Query)**

查询 GUI 对象动态属性，格式：

HC → PS-LCD	[object name].[runtime property name] ? <CR>	如： e1.text ?\r
PS-LCD → HC	Q+[object].[runtime property] = xxx <CR> Q-<CR> Q?<CR> Q!<CR>	成功，如 Q+e1.text=”abcd”\r 失败 命令格式错误 GUI 对象名字错误

- 事件(Event)

GUI 对象的某个动态属性在运行时改变，而且该 GUI 对象的**事件通知(Verbose)**属性在 Designer 设计阶段已选中，主控制器将会收到 Event 类型的消息，格式如下：

PS-LCD → HC	E+[[object name].[runtime property name] = xxx <CR>	如：HC 收到 E+e1.text=”efgh” 表示 GUI 对象 e1 的内容被用户改为”efgh”，

注：text 为文本框动态属性，e1 为文本框的名字，”abcd”为需要显示的内容；<CR>为 ASCII 码’\r’(十六进制 0x0D)

利用以上三条通讯命令，外部控制单元即可轻松掌控 GUI 对象的运行时状态，几行代码让人机界面动起来。关于详情使用说明，请参看“**PS-LCD 软件速查表**”文档。

4.2.4 主控制器软件设计

对主控制器来说，与界面相关的软件非常简单——**按三条 CTP 协议指令与 PS-LCD 通讯即可**。通常只需一个简单 Loop 循环，软件流程如图 4-2. 从图中可看出，复杂的人机界面软件转变成了简单串行口通讯程序，而通讯协议简单直观，有效降低了开发难度，提高了系统稳定性，可大大缩短产品开发周期。

由于 PS-LCD 通讯协议为纯 ASCII 码格式，在没有主控制器的情况下，将 PC 串口与 PS-LCD 连接，利用 Windows 的超级终端（程序->附件->通信->超级终端）或者串口调试助手可模拟主控制器串口，轻松直观地跟踪代码。如果采用超级终端调试，相关设置如下：

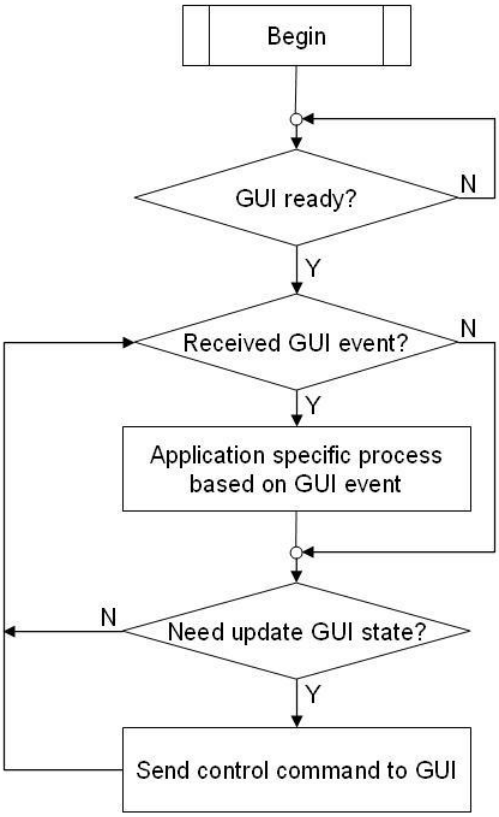
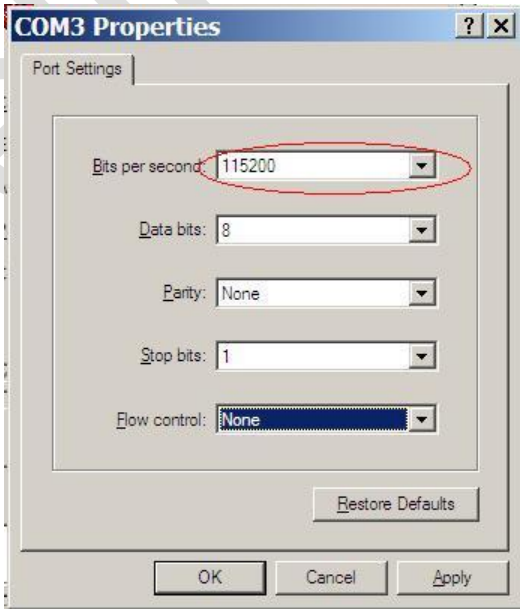
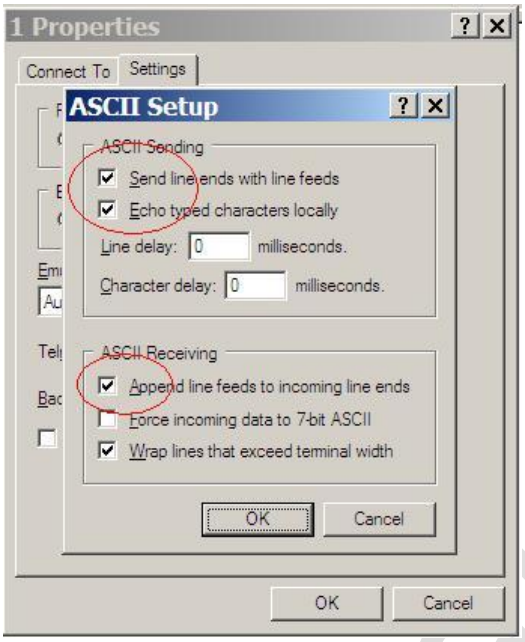


图 4-2 主控制器软件流程图

1) 设置串口，红圈内的波特率须与 Designer 中设置一致，无硬件流控制.



2) 设置 ASCII 通信方式，红圈部分必须选中。



完成设置后，点“连接”按钮。然后上电启动 PS-LCD，待 LCD 界面启动完毕后,在超级终端上将出现如下 CTP 消息：

E+sys.state=ready. （表示 PS-LCD 已经就绪，可发送接收 CTP 消息）

如果点击界面上的某个按钮控件（假设按钮名字为 b），超级终端上显示：

E+b.clicked=1

如果我们输入如下命令(敲完后回车)：

e.text=“This is a tes!”

那么你会看到 LCD 上的文本框（假设文本框的名字为 e）显示内容更新为”This is a test!”，同时超级终端上看到如下消息：

C+ （表示控制命令执行成功）

下面是一个主控制器通过 CTP 通讯协议集成界面的简单例子：

- **要求：**用文本框实时显示电压值，按钮控制系统启动/停止
- **实现：**

硬件	主控制器 + PS-LCD
软件	**PS-LCD side:** 1) 在 designer 中建立工程 demo，双击 main.ui，拖放一个文本框到界面编辑区，命名为 volt； 2) 在界面区拖放一个按钮，设文本为“启动”，控件 ID 设为 start，

不要勾选**事件通知**属性，点击属性的**动作脚本**，输入 ctpBinTx（0x55）；（**注意**：ctpBinTx 为 PS-LCD 内部函数，详细用法请参看“软件速查表”。），这样当界面运行时，按下 start 按钮，PS-LCD 会自动向外部控制单元发送一字节十六进制数 0x55。）

3) 点击菜单栏“**工具->生成界面**”，生成 demo.spf 文件；

4) 通过 Flex 下载 demo.spf 到 PS-LCD，并让 PS-LCD 进入界面模式，启动界面。

➡ **主控制器（通常为 51 或者 ARM）软件：**

1) 当收到字节值为 0x55，说明界面上按下了启动按钮，此时开始采集操作；

2) 采集电压值，假设此时为 11.22V，通过通讯接口发送字符串“volt.text=11.22”到 PS-LCD。

3) 定时重复步骤 2，可实现界面电压值的实时显示。

➡ **调试：*（用超级终端）***

假设界面并没有正常工作，可以打开超级终端（设置按上面的方法），先检查是否收到”E+sys.state=ready”消息来确认通信是否成功。在超级终端中输入 volt.text=”11.22”，查看 LCD 上是否有错误提示框弹出，然后再检查超级终端中的返回值（C+表示成功执行；C-表示执行出错）。总之，你可以快速直观地找到问题所在。

4.3 用户自定义协议

顾名思义，该协议由用户自定义实现，以便让 PS-LCD 能够灵活地与各种外部控制单元实现协议的无缝连接。理论上说，任何有公开详细通讯协议说明的外部控制单元(如：51/ARM/DSP 等主控制器、PC 电脑、PLC、现场采集控制单元等)，都能与 PS-LCD 连接通讯，实现人机界面扩展。

当在 designer 设计界面时指定 PS-LCD 通讯协议为 UserDefine，PS-LCD 只提供接收和发送数据的脚本函数接口，发送什么内容的数据，接收到数据如何处

理，由用户在脚本中自定义实现。

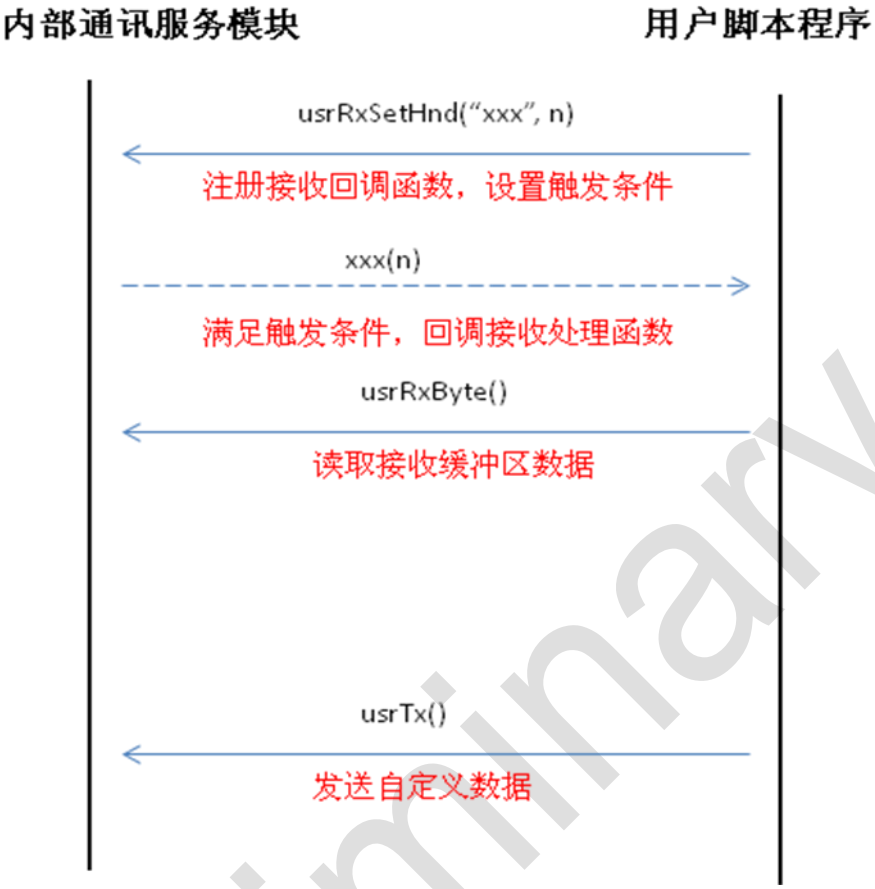


图 4-3 自定义协议通讯示意图

以下是相关的函数接口说明：

usrTx( ) —— 调用该函数，可发送任意内容数据，一个参数代表一个字节，每次调用最多支持 25 个字节。如： usrTx（0x55, 0x66）将发送十六进制 0x55 和 0x66 两字节数据；usrTx(xx.value) 将发送 ID 为 xx 的控件当前内容，假设 xx 为文本框，当前内容为 16， 那么发送的数据是 0x10 一字节数据。

usrRxSetHnd(name, size) —— 该函数用于设定接收数据的处理操作。Name 为回调处理函数的名称字符串，size 为触发接收事件的接收缓冲区字节数。如：usrRxSetHnd("xxx", 10)，设置接收回调函数名称为 xxx， 调用的触发条件：接收缓冲区数据超过 10 字节；回调函数 xxx 由用户自定义，声明应为 function xxx(size) 样式， 其中 size 为调用该函数时，可读取的数据字节数；。

usrRx( ) ——读取接收数据缓冲区所有数据，返回值为对应大小的数据字节数组，读取完毕后，接收缓冲区自动清空。**该函数只能在接收回调函数中调用**。

usrRxByte( ) ——读取接收数据缓冲区一字节数据，返回值为一字节数据，

未读取的数据仍然保留在接收缓冲区。**该函数只能在接收回调函数中调用**。

usrRxClr( ) ——清空接收缓冲区数据。

用自定义通讯协议的界面软件设计步骤：

1) 用 Designer“所见即所得”，0 代码完成界面控件布局；

2) 在需要发送数据的控件动作脚本中，调用 usrTx()函数实现数据发送；如在按钮的动作脚本中输入：usrTx(0x55)；当该按钮在运行时按下一次，PS-LCD 将向通讯口发送一字节十六进制数据 55；

3) 打开全局脚本，注册接收回调函数，输入 usrRxSetHnd("xxx", n); 其中 xxx 为接收回调函数名称， n 为触发字节数。在相关界面（一般为主界面）动作脚本中实现 xxx 函数，如：

```
function xxx(size)
{
    test.text = usrRxByte();
}
```

上面函数将在接收缓冲区数据字节数超过 n 时，自动被调用，从上面的实现可以看出，当 PS-LCD 接收到数据后，将把该字节读出，并显示在名字 ID 为 test 的控件文本中，假设该控件是一个文本框，那么外部控制单元每向 PS-LCD 发送一个字节，文本框内容自动刷新为该字节的数据显示；如： 外部控制单元向 PS-LCD 发送一字节数据 0x10， 这是可以看到界面的文本框内容自动刷新显示为 16。

4) 启动模拟器，模拟通讯的发送接收过程，验证自定义的通讯协议，重复 2 和 3 步骤直到满意为止。

5) 生成 spf 文件，将初步调试好的界面下载到 PS-LCD 中，用实际的外部控制单元与 PS-LCD 通讯，验证最终设计结果。

注意：上述步骤的通讯协议帧比较简单，主要为更好地说明自定义通讯协议的设计过程。在实际项目中，建议帧数据包中应有帧头、帧尾和校验字节位。当 PS-LCD 接收到超过帧字节数的数据后，触发执行接收回调函数，确认帧头、帧尾和校验都无误后，在调用相关脚本刷新界面控件，这样可以最大限度地保证通

讯可靠性。建议帧数据格式如下：

帧头字节	数据字节	校验字节	帧尾字节

关于自定义通讯协议函数说明，请参看“**PS-LCD 软件速查表**”；自定义通讯协议的使用实例，请参看光盘中的演示工程“**4.3-自定义通讯协议演示**”。

第五章 设计实例

本章以一个温控系统为例，详细说明基于 PS-LCD 的人机界面设计全过程(以 CTP 通讯协议为例)。

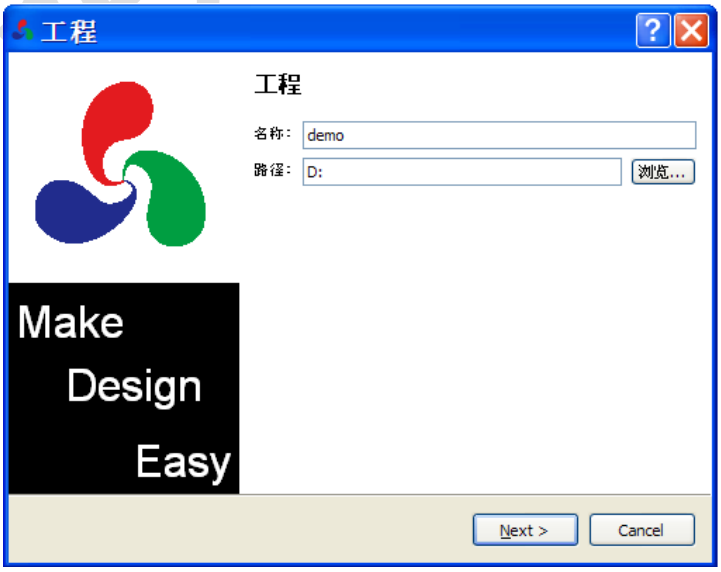
5.1 界面要求

- ☑ 用两个按钮来控制室内温度，按“+“， 温度上升，按”-”，温度下降，同时主控制器控制相关电路调节对应温度;
- ☑ 温度需要数字显示出来;
- ☑ 当温度超过 30 度，显示“过热”动画；小于 5 度，显示“过冷”动画；介于 5-30 度之间，显示“正常”动画.

5.2 实现步骤

5.2.1 编辑界面

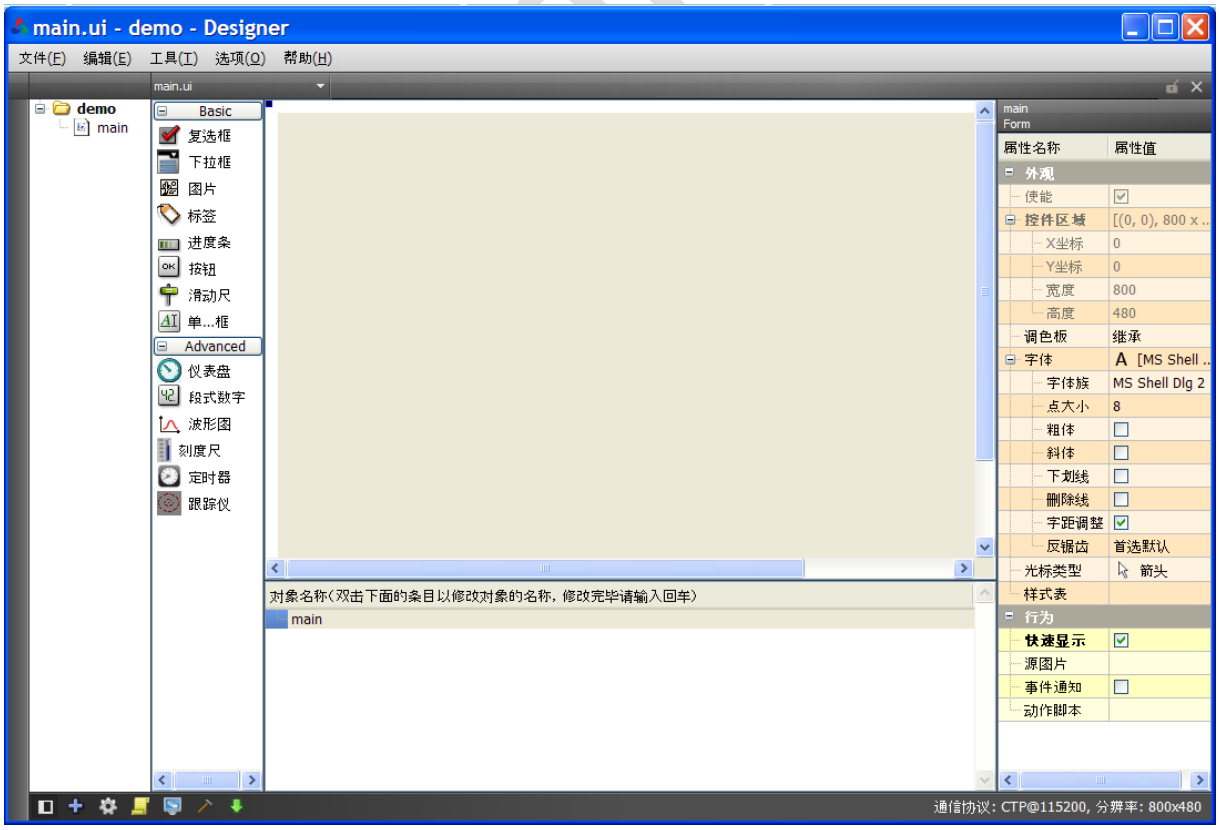
1) 打开可视化界面编辑工具 Designer，点击菜单栏中的文件->新建工程，输入新工程的名字，选择保存路径。我们这里新工程命名为 demo，路径为 D 盘， 然后点 next，然后点 finish。



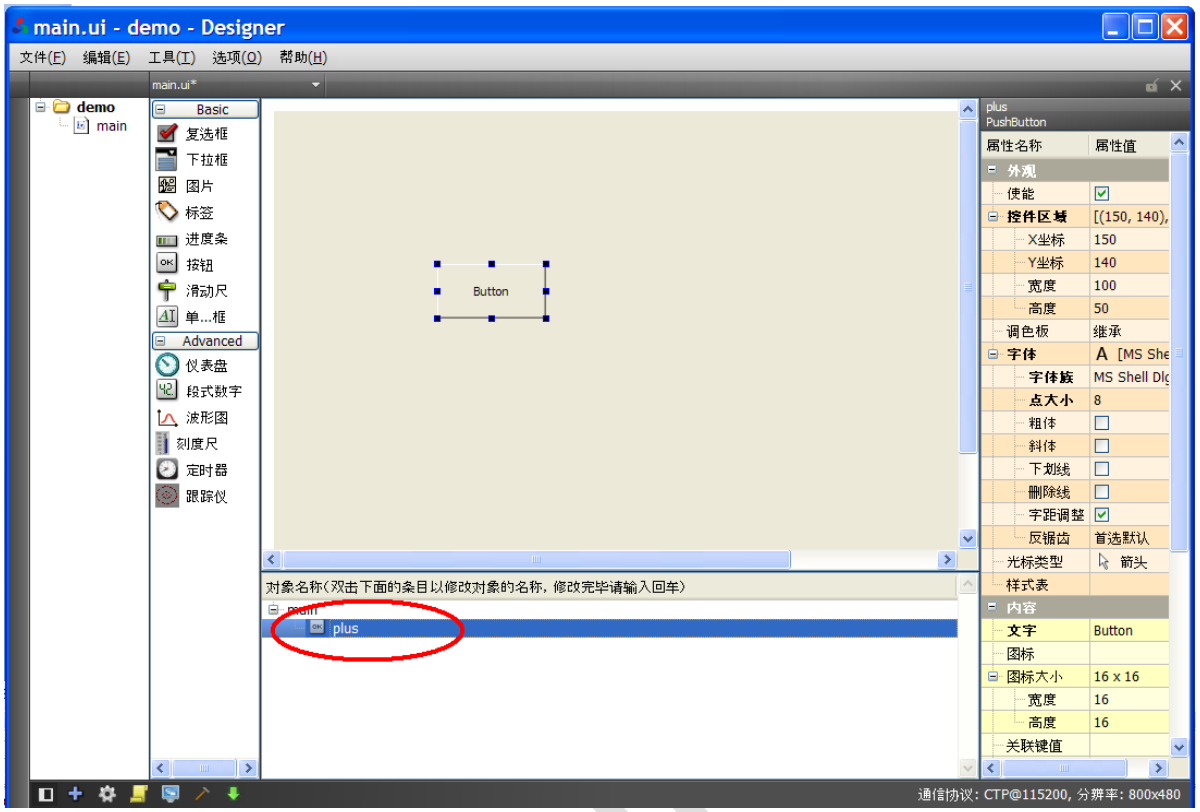
2) 当出现如下画面，采用默认主界面为 main.ui，PS-LCD 分辨率为 WVGA(800 x

480)，其他用默认选项，点确定。 工程建立完毕，可以开始编辑温控界面了。

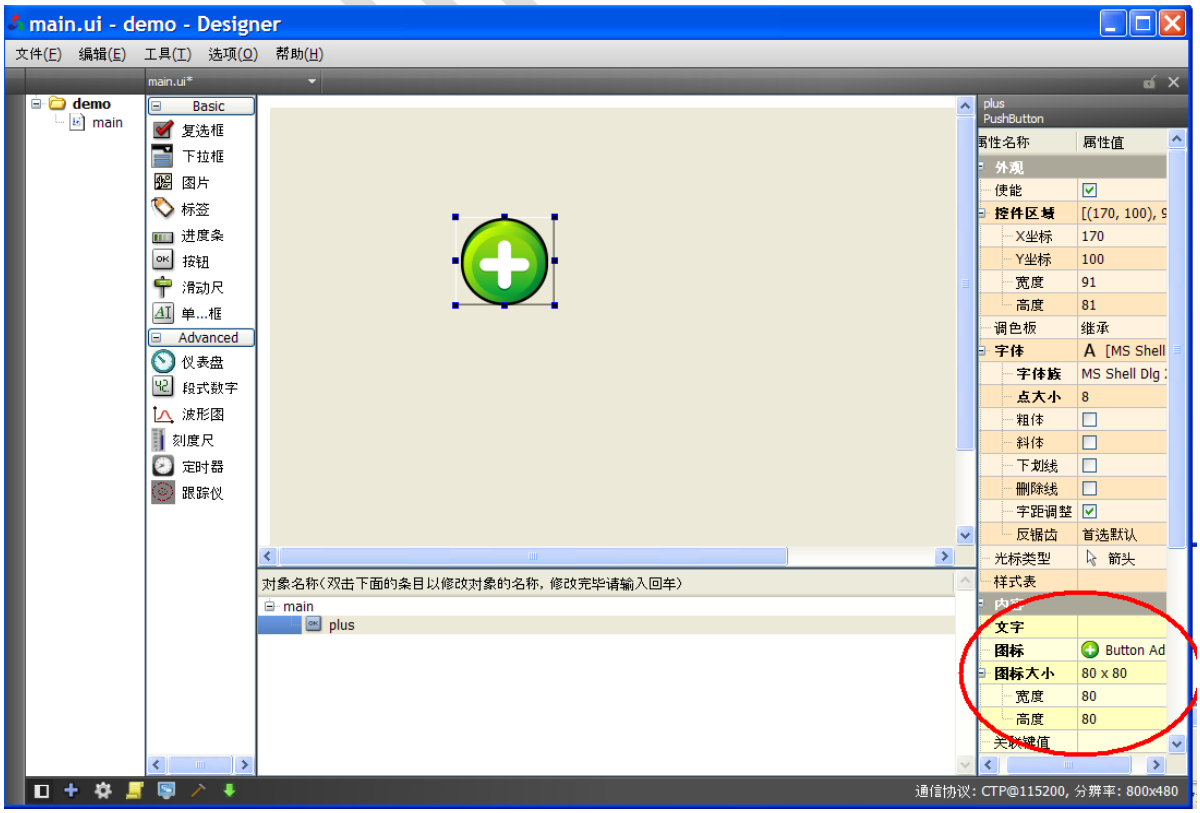
3) 当出现如下画面时，即可开始编辑 main 界面。



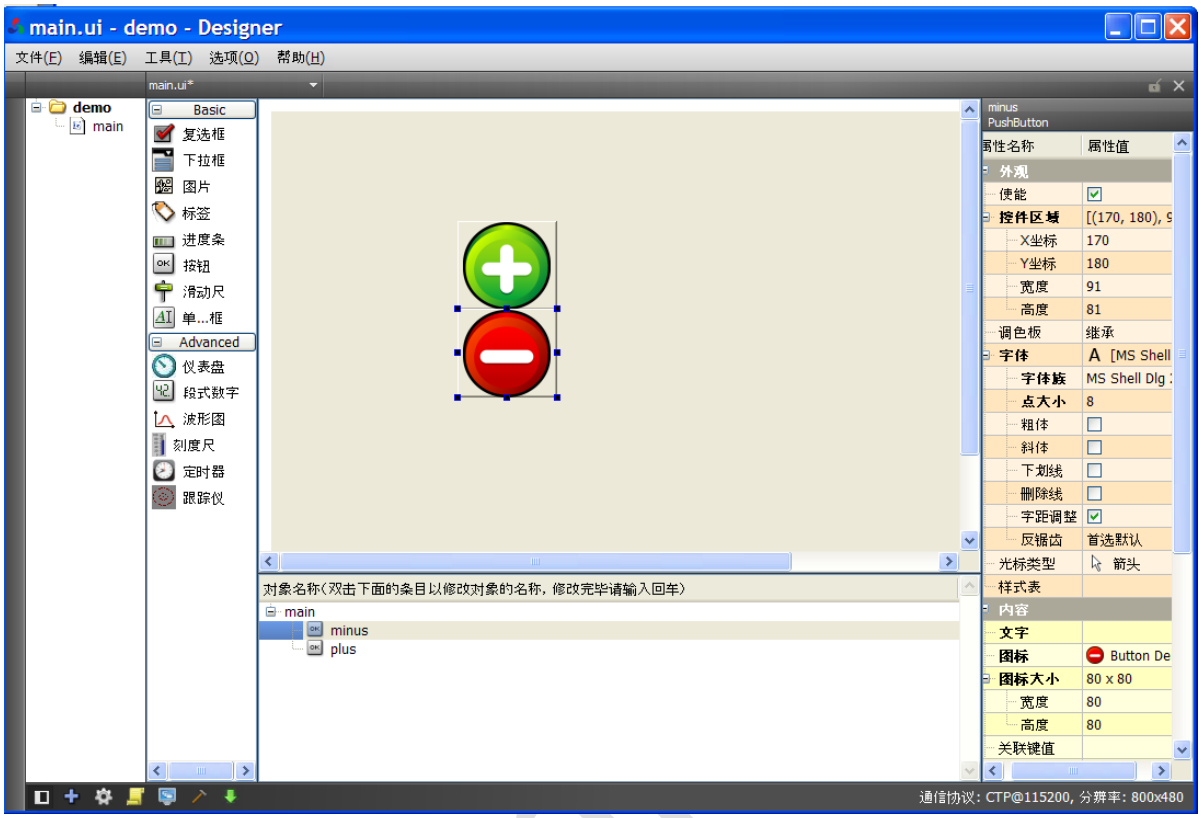
4) 用鼠标将按钮控件从控件区拖入界面编辑区，选中该按钮，然后在对象管理区中重命名为 plus。



5) 仍然选中 plus 按钮，在属性区中点击图标(icon)属性上的按钮选择图标，(在此我们选择了一个加号的 png 格式图片)，并且在图标尺寸(iconSize)属性中将图标分辨率调整为 80x80，最后调整界面编辑区中的按钮大小与图标合适。

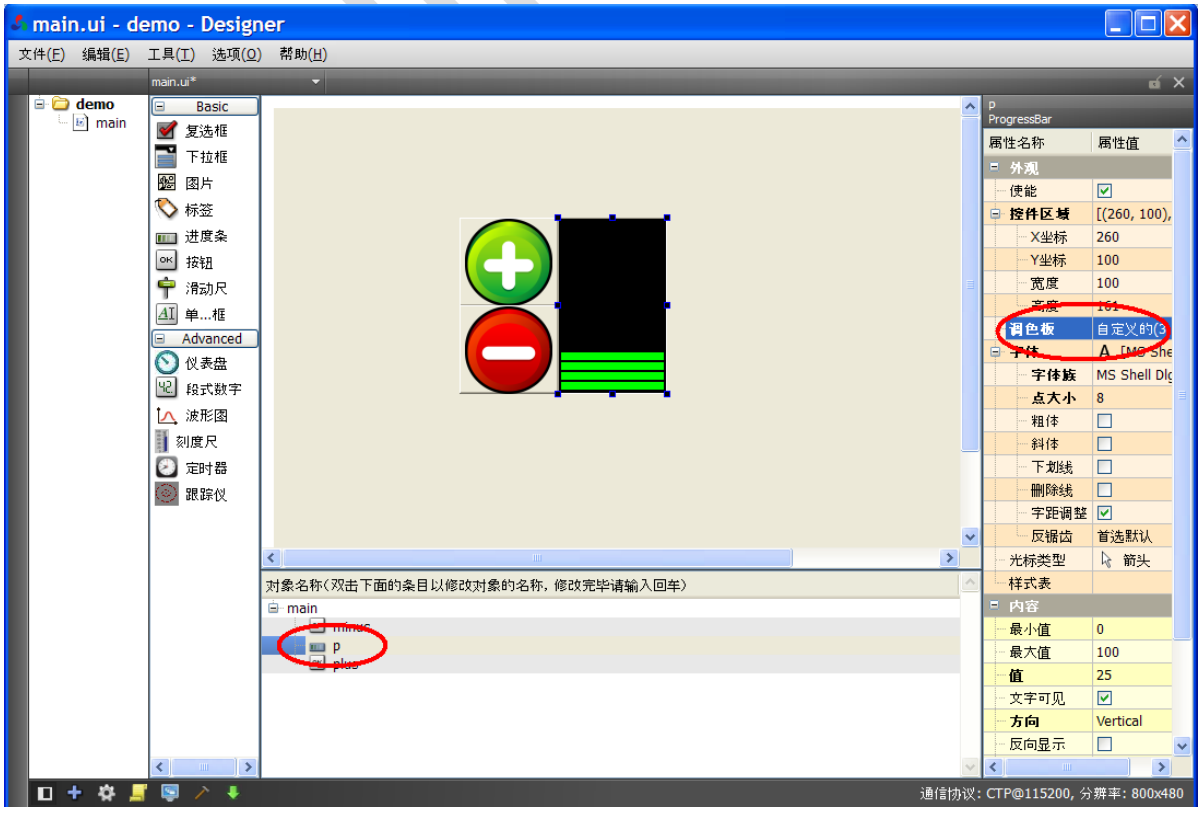


6) 按步骤 5 和 6 的方法，添加一个 minus 按钮，如下图。



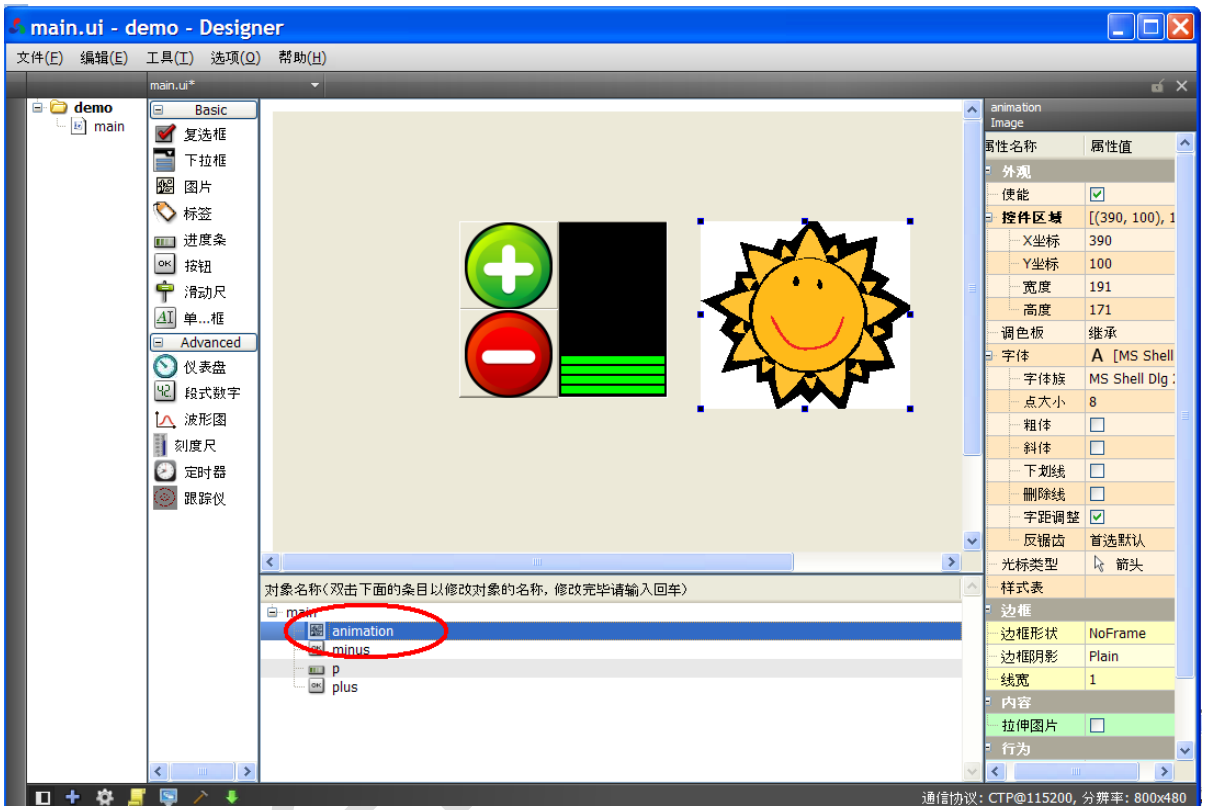
7) 从控件区拖一个进度条控件到界面编辑区，命名为 p，调整大小与按钮对齐。

在属性区通过调色板（pallete）属性改变显示风格为黑底/绿进度指示，通过



minimum/maximum/value 属性分别设置进度范围最小 0，最大 50，默认进度值为 20，通过方向(orientation)属性改变进度条方向为垂直 vertical。

8) 从控件区拖入图片控件，命名为 animation，选择源图片(source)属性中的图片为 sunlog.gif 动画文件作为默认动画。然后将 cold.gif 和 hot.gif 拷贝到 demo 工程工作目录下的 picture 目录中，以便界面运行时改变动画效果。



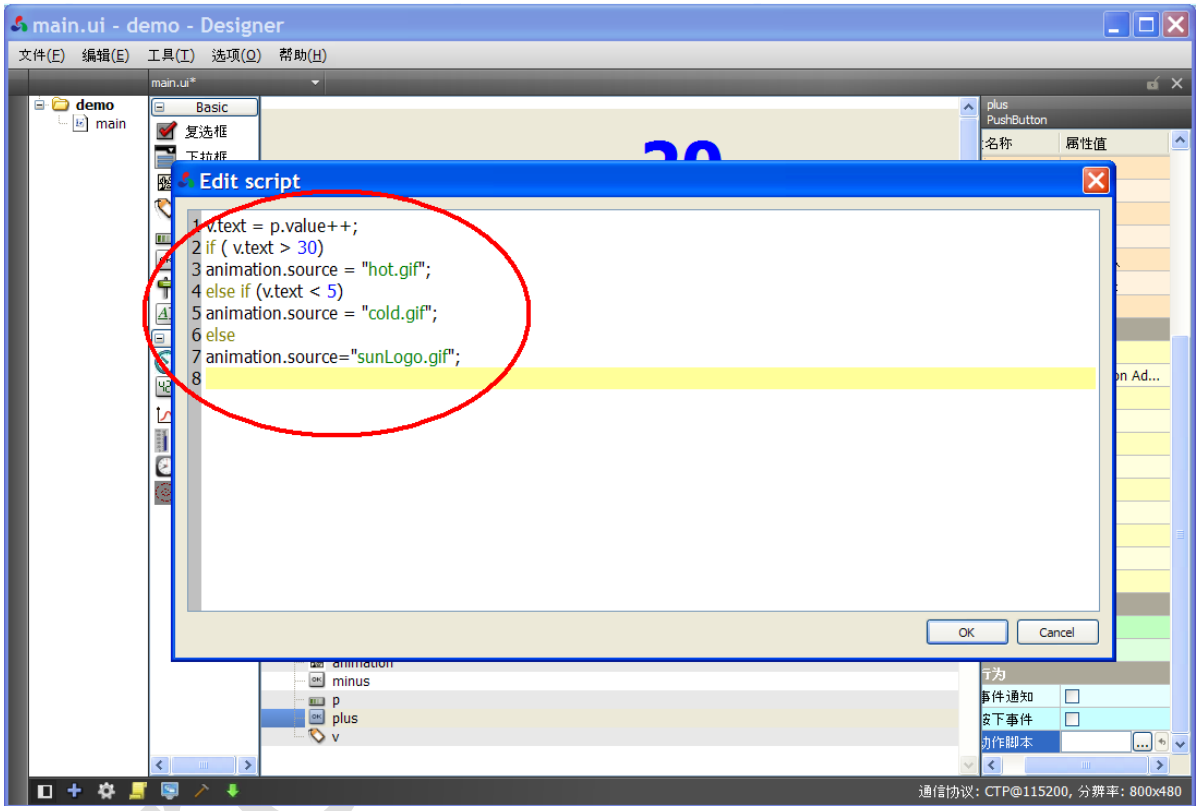
9) 从控件区分别拖入三个标签控件，分别输入文字：PS-LCD 演示——温度控制系统界面、当前温度、20。再分别通过字体(font)属性调节字体类型，大小，颜色、位置等，最后将文本内容为 20 的标签命名为 v。

10) 选中进度条对象 p，然后选中属性区中的事件通知（Verbose）属性，以便进度条 p 的值发生变化时，PS-LCD 自动发送消息通知主控制器。

11) 选中按钮 plus，点击属性区中的动作脚本(action)属性按钮，在弹出的输

入框中输入如下 JavaScript 脚本程序：

```
v.text = p.value++;
if ( v.text > 30)
    animation.source = "hot.gif";
else if (v.text < 5)
    animation.source = "cold.gif";
else
    animation.source="sunLogo.gif";
```

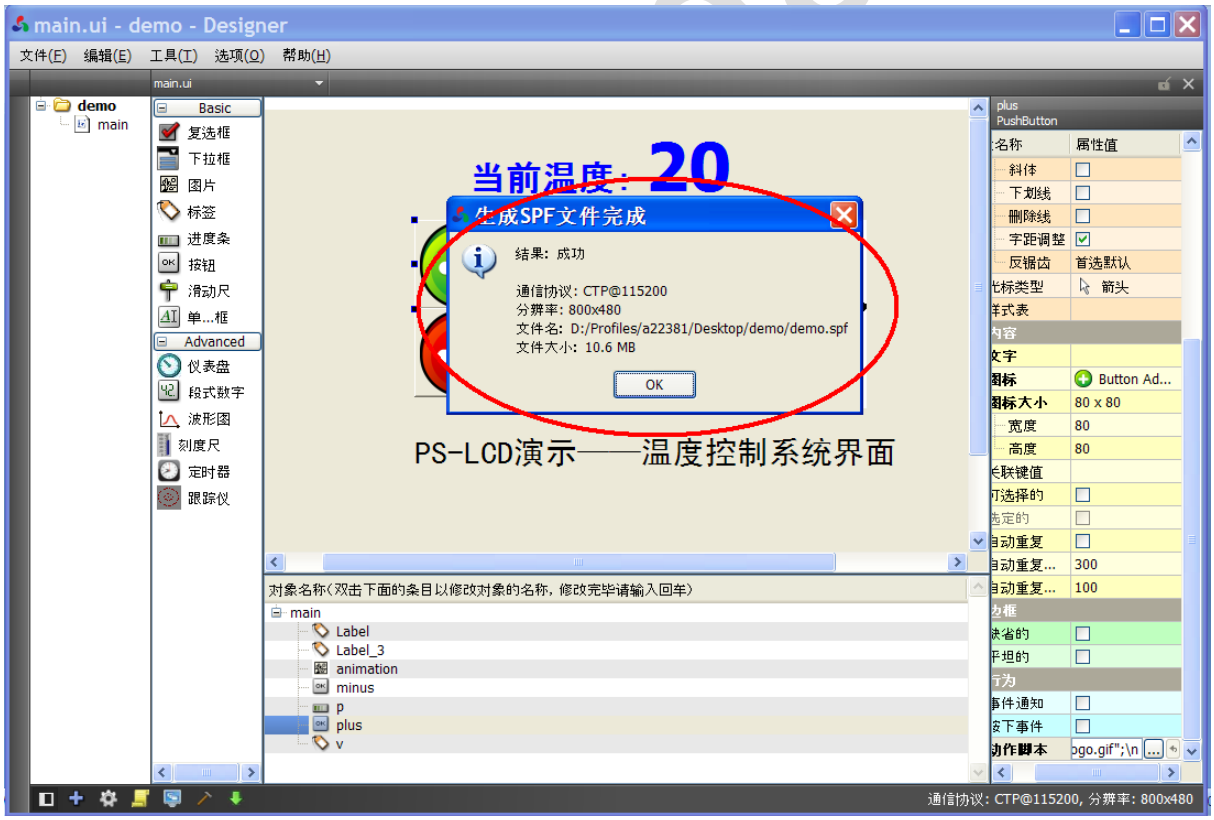


12) 同步骤 11，在按钮 minus 的**动作脚本**(action)属性输入框中输入：

```
v.text = p.value--;
if ( v.text > 30)
    animation.source = "hot.gif";
else if (v.text < 5)
    animation.source = "cold.gif";
else
    animation.source="sunLogo.gif";
```

13) 点击菜单栏“工具->模拟器”，启动 PS-LCD 软件模拟器，验证界面效果和动作是否与设计一致。

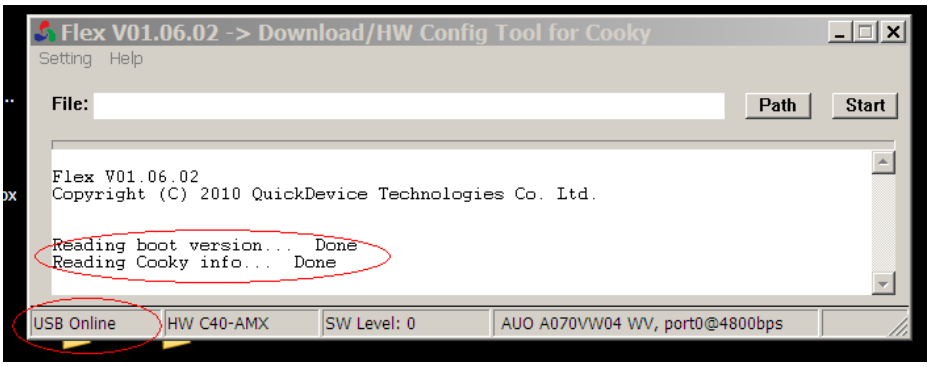
14) 上步确认无误后，点击菜单栏“工具->生成界面”选项生成 demo.spf 文件。



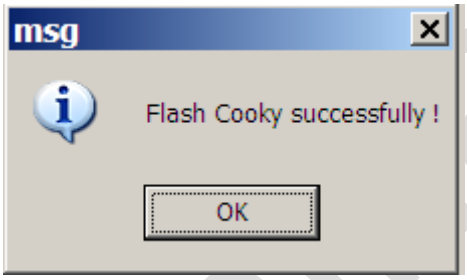
5.2.2 下载 SPF 文件

1) 用 USB 下载线连接 PS-LCD 与 PC，进入下载模式；

2) 启动 Flex 下载工具，这是会看到：USB online；



3) 点击 path 按钮，选择刚才生成的 demo.spf，点击 Start 开始下载，完成后提示下载成功：



4) 拔掉 USB 下载线，按复位按钮或者重新给 PS-LCD 上电进入界面模式（需事先去掉 PS-LCD 背面的模式短路块），待启动完毕，上面设计的 demo 工程界面即出现在 LCD 上。

5.2.3 主控制器界面软件

主控制器软件由三个文件组成：

- app.c (温控系统的 GUI 主程序)
- ctp.c (Greatal 提供的 CTP 协议源文件)
- hw.c (用户的串口驱动程序源文件)

ctp.c 由 Greatal 提供，*用户不需要做任何修改，直接与用户的应用程序编译连接即可*，这里不做介绍；hw.c 是跟特定的主控制器 HC 硬件平台相关的串口驱动，只要提供三个 API 接口（hw_init(), hw_write()和 hw_read()）供 ctp.c 调用即可，用户自行完成，我们这里也忽略。

下面重点介绍 app.c 的程序，源代码见下一页。

```
/* PS-LCD based GUI demo codes
 * Copyright (C) 2011 Beijing Greatal Technologies Co. Ltd. */
#include "ctp.h"
#include <string.h>
void main()
{
    char name[16], prop[16], value[64];
    int degree = 25;

    /* Initialize ctp port */
    ctp_init();
    /* Infinite event loop*/
    while(1) {
        if (ctp_event(name, prop, value)) {
            /* no invalid event found, do something you'd like */
            continue;
        }
        if (!strcmp(name, "p") && !strcmp(prop, "value")) {
            /* if temperature setting get changed */
            degree = atoi(value);
            /* Call hardware control routine to set temperature */
            …..
        }
}
```

当设定温度变化时处理程序

代码分析如下：

- app.c 由一个无限循环的 event loop 组成，当调用 ctp_event（）函数时，程序通过主控制器串口读取 GUI 的事件。如果没有 GUI 事件产生，该调用会立即返回（如果 hw.c 中的 hw_read()在没有有效串口数据时立即返回，适用于没有 OS 的系统）或者阻塞（如果 hw.c 中的 hw_read()在没有有效串口数据时阻塞， 适用于有 OS 的系统）。
- 如果 ctp_event（）的返回值非 0，说明有 GUI 事件产生，事件的内容存储在 name, prop 和 value（均为字符串），通过比较确定 GUI 的事件类型，然后调用相应的处理程序。
- 关于过热或者过冷的动画效果，在用 designer 编辑界面时，直接在**动作脚本** action 属性中输入 JavaScript 脚本描述即可，无需主控制器参与。
- 本例中用到了几个标准 C 语言 string 库里面的函数，它们是：strcmp（），strcat()和 itoa()，具体用法请参考相关资料。

注意：app.c 源码中的蓝色字体为在 designer 设计阶段定义的控件名字，红色字体为 ctp.c 提供的 API 函数，绿色字体为控件的动态属性名称（详情见“PS-LCD 软件速查表”文档）。

5.2.4 实际运行结果

为测试效果，本例在如下软硬件环境编译运行：

- 硬件：7 寸 PS-LCD（LCD：AUO 的 7 寸 WVGA 800 x 480 TFT LCD）
- 主控制器三线串口：采用 PC 机 RS232 串口模拟
- 编译环境：VC 下编译
- 调试环境：Windows 系统自带的“超级终端“

温控系统运行后，在实际 PS-LCD 上的界面效果如下：

图 5-1 温度正常时界面

图 5-2 按“+”键将温度调整到 31 度时界面

图 5-3 按“-”键将温度调整到 4 度时界面

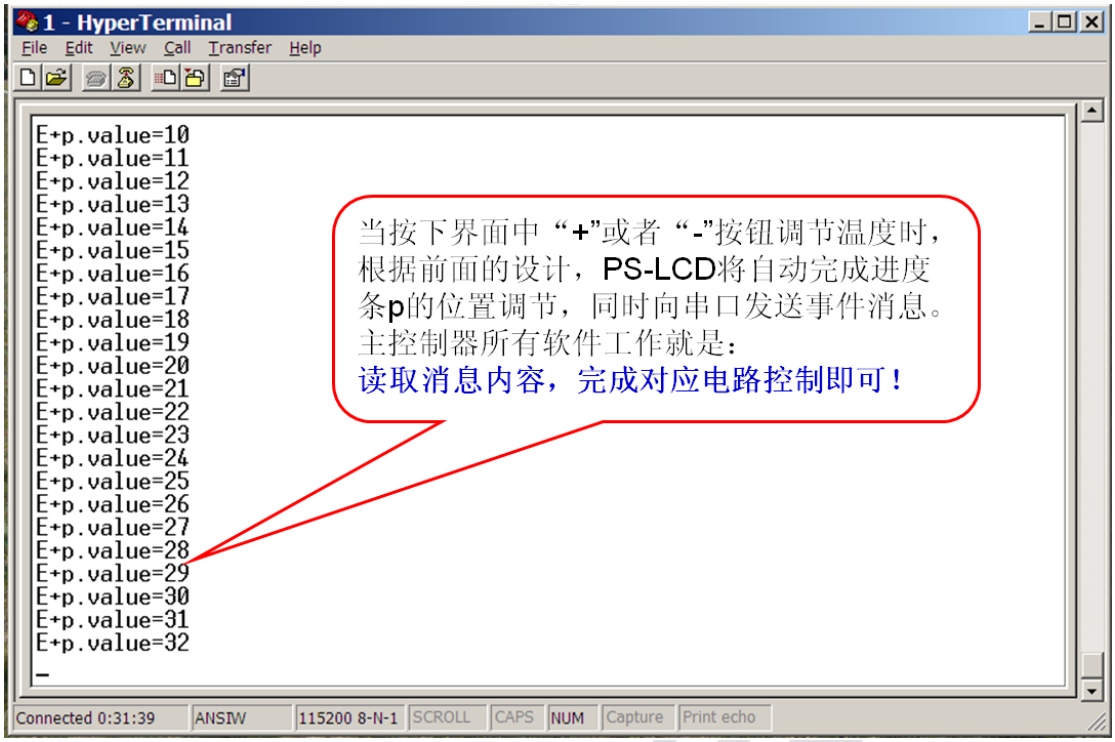


图 5-4 超级终端监控串口数据

5.3 总结

从上例可看出，为嵌入式系统增加人机界面，软件开发可以很简单：

- ☑ 调试一个串口驱动程序；
- ☑ 利用三个简单的 API（ctp_event（）， ctp_control（）， ctp_query（））根据实际需求写少量控制代码。

基于 CTP 通讯协议的人机界面程序为纯 C 代码，不依赖于任何库（除 C 语言的 string 库外），也不依赖于任何软件平台（有无操作系统均可，不需要任何 GUI 的函数库），可轻松嵌入到用户主控制器 C 代码中。

总之，**任何外部控制单元（51/ARM/DSP，PLC 等），无论性能高低，只需一个串口（或者 485、CAN、Wifi 等），都能轻松快速与 PS-LCD 连接，用极少系统资源实现强大人机界面**。

第六章 定制界面风格

通过前面章节的学习，我们已经能够生成、下载和运行自己设计的界面，具备了开发 PS-LCD 界面的基本概念和技能，完全可以胜任产品开发工作。但针对界面要求比较高场合，需更多动态效果，设计更加个性的界面样式，本章将详细阐述通过样式表(Style Sheet)来实现上述功能。

样式表（Style Sheet）是一段字符串文本，定义了如何显示各种控件的显示样式，能自如地同时改变一个或者多个控件的布局和外观。

这种改变，既可以在 Designer 设计阶段通过控件的样式表(Style Sheet)属性静态定义，也可以在界面运行时通过脚本改写该属性内容动态实现，为开发者提供了极大的灵活性。

6.1 样式表语法

样式表由两个主要的部分构成：**选择器 + 一条或多条声明**

```
selector  {declaration1; declaration2; ... declarationN }
```

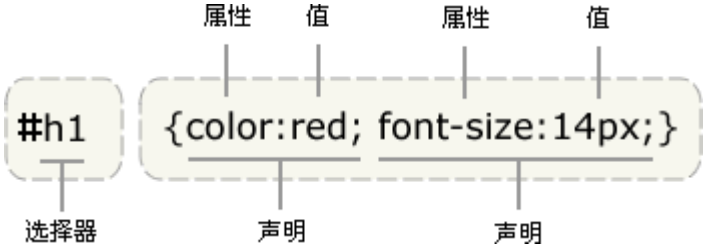
选择器通常是需要改变样式的控件名称（需在名字前加#），每条声明由一个属性和一个值组成。不同声明间被引号分开。属性（property）是您希望设置的样式属性，每个属性都有值。属性和值被冒号分开。

```
selector {property: value}
```

下面这行代码的作用是将名字为 h1 的控件内文字颜色定义为红色，同时将字体大小设置为 14 像素。在这个例子中,h1 是选择器,color 和 font-size 是属性，red 和 14px 是值。

```
#h1 {color: red; font-size:14px;}
```

下面的示意图为您展示了上面这段代码的结构：



提示：请使用花括号来包围声明。

用选择器可同时指定多个控件名字，用逗号将不同的控件名称分开，分享相同的声。。在下例中，名称为 h1 到 h6 所有控件字体颜色设置为绿色。

```
#h1,#h2,#h3,#h4,#h5,#h6 {color: green;}
```

样式表的风格，子控件从父控件那里继承。如在界面控件(form)样式表里面设置的风格，子控件（如：放在界面中的按钮控件）将继承其风格。

这里需要特别注意的是，如果针对明确的特定控件设置样式表，格式中的选择器和花括号可以忽略。如：同样完成设置 xxx 控件的样式，

在动作脚本中的写法如下：

```
xxx.styleSheet="color: green; background-color: blue;"
```

在 xxx 控件的样式表属性中输入内容如下：（如果输入的样式表语法有问题，将无法点击样式表属性对话框中的 OK 按钮）

```
color: green; background-color: blue;
```

6.2 基本样式定义

6.2.1 颜色

color 设置前景颜色，一般为控件上文本颜色。下例设置 h1 控件前景色为红：

```
#h1 {color: red}
```

使用 background-color 属性为控件设置背景色。这个属性接受任何合法的颜色值。这条规则把元素的背景设置为灰色：

```
#p {background-color: gray;}
```

如果您希望背景色从元素中的文本向外少有延伸，只需增加一些内边距：

```
#p {background-color: gray; padding: 20px;}
```

selection-color 和 selection-background-color 分别对应选中区域的前景色和背景色，定义方法与上面类似。

6.2.2 文本

通过文本属性，您可以改变文本的颜色、转换、装饰文本等等.

- text-transform

处理文本的大小写。这个属性有 3 个值：none、uppercase、lowercase。默认值 none 对文本不做任何改动，将使用原有大小写。顾名思义，uppercase 和 lowercase 将文本转换为全大写和全小写字符。作为一个属性，text-transform 可能无关紧要，不过如果您突然决定把所有 h1 元素变为大写，这个属性就很有用。不必单独地修改所有 h1 元素的内容，只需使用 text-transform 为你完成这个修改：

```
#h1 {text-transform: uppercase}
```

使用 text-transform 有两方面的好处。首先，只需写一个简单的规则来完成这个修改，而无需修改 h1 元素本身。其次，如果您以后决定将所有大小写再切换为原来的大小写，可以更容易地完成修改。

- text-align

该属性定义文本的对齐方式，只对按钮和进度条控件有用，可以有如下值：

  - top
  - bottom
  - left
  - right
  - center

- text-decoration

text-decoration 有 5 个值：none、underline、overline、line-through、blink。underline 会对元素加下划线，overline 的作用恰好相反，会在文本的顶端画一个上划线。值 line-through 则在文本中间画一个贯穿线，blink 会让文本闪烁。none 值会关闭原本应用到一个元素上的所有装饰。通常，无装饰的文本是默认外观。

6.2.3 图标

icon-size 属性设置图标大小，如下例将控件 h 的图标设为 16x16 像素大小：

```
#h {icon-size: 16px;}
```

6.2.4 字体

- font-style

font-style 属性最常用于规定斜体文本。该属性有两个值：normal - 文本正常显示、italic - 文本斜体显示。

- font-weight

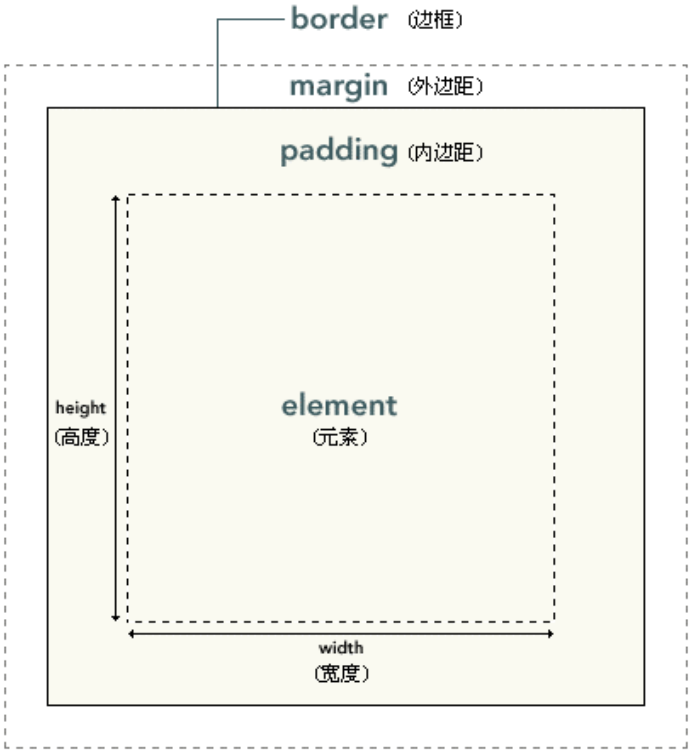
font-weight 属性设置文本的粗细。使用 bold 关键字可以将文本设置为粗体。关键字 100 ~ 900 为字体指定了 9 级加粗度。如果一个字体内置了这些加粗级别，那么这些数字就直接映射到预定义的级别，100 对应最细的字体变形，900 对应最粗的字体变形。数字 400 等价于 normal，而 700 等价于 bold。实例：

```
#p {font-weight:normal;}
```

- font-size

font-size 属性设置文本的大小，单位为像素。实例：#h1 {font-size:60px;}

6.3 框模型

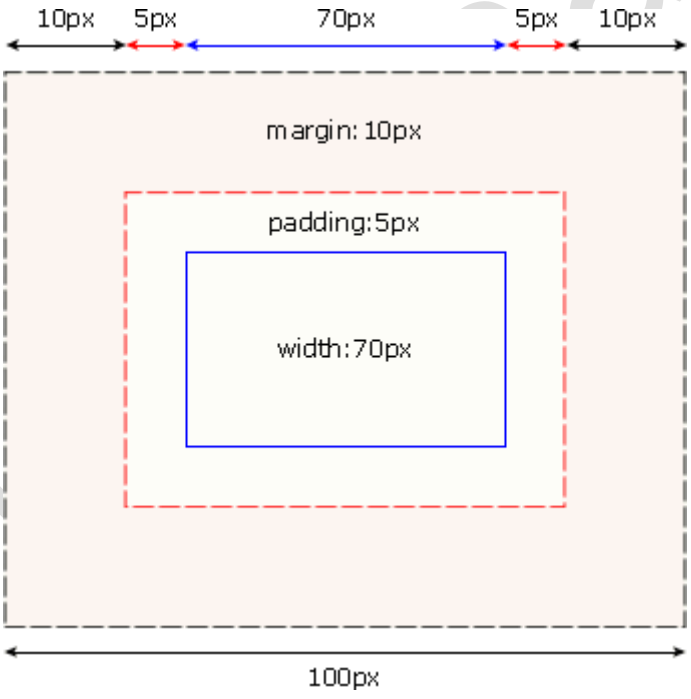


框模型(Box Model)规定了控件框处理内容、内边距、边框和外边距的方式。

控件框的最内部分是实际内容，直接包围内容的是内边距。内边距呈现了控件的背景。内边距的边缘是边框。边框以外是外边距，外边距默认是透明的，因此不会遮挡其后的任何元素。提示：背景应用于由内容和内边距组成的区域。内边距、边框和外边距都是可选的，默认值是零。内边距、边框和外边距可以应用于一个控件的所有边，也可以应用于单独的边。

在样式表中，width 和 height 指的是内容区域的宽度和高度。增加内边距、边框和外边距不会影响内容区域的尺寸，但是会增加元素框的总尺寸。假设框的每个边上有 10 个像素的外边距和 5 个像素的内边距。如果希望这个元素框达到 100 个像素，就需要将内容的宽度设置为 70 像素，见下图：

```
#box { width: 70px;   margin: 10px; padding: 5px;}
```



- element：元素。
- padding：内边距，也有资料将其翻译为填充。
- border：边框。
- margin：外边距，也有资料将其翻译为空白或空白边。

我们把 padding 和 margin 统一地称为内边距和外边距。边框内的空白是内边距，边框外的空白是外边距，很容易记吧。

6.3.1 内边距

padding 属性定义元素的内边距,值接受长度值或百分比值，但不允许使用负值。例如，如果您希望所有 h 控件的各边都有 10 像素的内边距，可这样：

```
#h {padding: 10px;}
```

还可以按照上、右、下、左的顺序分别设置各边的内边距，各边均可以使用不同的单位或百分比值：

```
#h {padding: 10px 10px 5px 2px;}
```

也通过使用下面四个单独的属性，分别设置上、右、下、左内边距：

- padding-top
- padding-right
- padding-bottom
- padding-left

下面的规则实现的效果与上面的简写规则是完全相同的：

```
#h { padding-top: 10px; padding-right: 10px; padding-bottom: 5px; padding-left: 2px; }
```

属　性	描　　述
padding	简写属性。作用是在一个声明中设置元素的所内边距属性。
padding-bottom	设置元素的下内边距。
padding-left	设置元素的左内边距。
padding-right	设置元素的右内边距。
padding-top	设置元素的上内边距。

6.3.2 边框

边框（border）是围绕控件内容和内边距的一条或多条线。border 属性允许你规定元素边框的样式、宽度和颜色。元素外边距内就是元素的的边框(border)。每个边框有 3 个方面：宽度、样式，以及颜色。

边框绘制在“元素的背景之上”。这很重要，因为有些边框是“间断的”（例如，点线边框或虚线框），元素的背景应当出现在边框的可见部分之间。边框的样式是边框最重要的一个方面，因为如果没有样式，将根本没有边框。

- border-style

border-style 属性定义了 10 个不同的样式：

值	描 述
none	定义无边框。
hidden	与 "none" 相同。不过应用于表时除外，对于表，hidden 用于解决边框冲突。
dotted	定义点状边框
dashed	定义虚线
solid	定义实线
double	定义双线。双线的宽度等于 border-width 的值。
groove	定义 3D 凹槽边框。其效果取决于 border-color 的值。
ridge	定义 3D 垄状边框。其效果取决于 border-color 的值。
inset	定义 3D 凹下效果边框。其效果取决于 border-color 的值。
outset	定义 3D 凸起效果边框。其效果取决于 border-color 的值。

例如，把一幅图片 i 的边框定义为 outset，使之看上去像是“凸起按钮”：

```
#i {border-style: outset;}
```

您可以为一个边框定义多个样式，例如：

```
#p {border-style: solid dotted dashed double;}
```

上面这条规则为名 p 的控件定义了四种边框样式：实线上边框、点线右边框、虚线下边框和一个双线左边框。我们又看到了这里的值采用了 top right bottom left 的顺序，讨论用多个值设置不同内边距时也见过这个顺序。如果您希望为

控件的某一个边设置边框样式，而不是设置所有 4 个边的边框样式，可以使用下面的单边边框样式属性：

- border-top-style
- border-right-style
- border-bottom-style
- border-left-style

因此这两种方法是等价的：

```
#p {border-style: solid solid solid none;}
#p {border-style: solid; border-left-style: none;}
```

注意：如果要使用第二种方法，必须把单边属性放在简写属性之后。因为如果把单边属性放在 border-style 之前，简写属性的值就会覆盖单边值 none。

border-width

border-width 属性为边框指定宽度。单位为像素，如 2px。所以，我们可以这样设置边框的宽度：

```
#p {border-style: solid; border-width: 5px;}
```

当然，也可以按照 top-right-bottom-left 的顺序设置元素的各边边框：

```
#p {border-style: solid; border-width: 15px 5px 15px 5px;}
```

您也可以通过下列属性分别设置边框各边的宽度：

- border-top-width
- border-right-width
- border-bottom-width
- border-left-width

因此，下面的规则与上面的例子是等价的：

```
#p {
    border-style: solid;
    border-top-width: 15px;
    border-right-width: 5px;
    border-bottom-width: 15px;
    border-left-width: 5px;
}
```

在前面的例子中，您已经看到，如果希望显示某种边框，就必须设置边框样式，比如 solid 或 outset。那么如果把 border-style 设置为 none 会出现什么情况：

```
#p {border-style: none; border-width: 50px;}
```

尽管边框的宽度是 50px，但是边框样式设置为 none。在这种情况下，不仅边框的样式没有了，其宽度也会变成 0。边框消失了，为什么呢？这是因为如果边框样式为 none，即边框根本不存在，那么边框就不可能有宽度，因此边框宽度自动设置为 0，而不论您原先定义的是什么。

记住这一点非常重要。事实上，忘记声明边框样式是一个常犯的错误。根据以下规则，所有 h1 元素都不会有任何边框，更不用说 20 像素宽了：

```
#h1 {border-width: 20px;}
```

由于 border-style 的默认值是 none，如果没有声明样式，就相当于 border-style: none。**因此，如果您希望边框出现，就必须声明一个边框样式。**

border-color

使用 border-color 属性设置边框颜色，它一次可以接受最多 4 个颜色值。可以使用任何类型的颜色值，例如可以是命名颜色，也可以是十六进制和 RGB 值：

```
#p {
  border-style: solid;
  border-color: blue rgb(25%,35%,45%) #909090 red;
  }
```

如果颜色值小于 4 个，值复制就会起作用。例如下面的规则声明了段落的上下边框是蓝色，左右边框是红色：

```
#p {
  border-style: solid;
  border-color: blue red;
}
```

默认的边框颜色是控件本身的前景色。如果没有为边框声明颜色，它将与控件的文本颜色相同。另一方面，如果控件没有任何文本，假设它是一个表格，其中只包含图像，那么该表的边框颜色就是其父控件的文本颜色（因为 color 可以继承）。这个父控件很可能是界面(form)控件。还有一些单边边框颜色属性。它们的原理与单边样式和宽度属性相同：

- border-top-color
- border-right-color
- border-bottom-color
- border-left-color

要为 h1 元素指定实线黑色边框，而右边框为实线红色，可以这样指定：

```
#h1 {
  border-style: solid;
  border-color: black;
  border-right-color: red;
  }
```

我们刚才讲过，如果边框没有样式，就没有宽度。不过有些情况下您可能希望创建一个不可见的边框。样式表引入了边框颜色值 transparent。这个值用于创建有宽度的不可见边框。请看下面的例子：

```
#a {
  border-style: solid;
  border-width: 5px;
  border-color: transparent;
  }
```

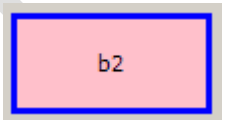
从某种意义上说，利用 transparent，使用边框就像是额外的内边距一样；此外还有一个好处，就是能在你需要的时候使其可见。这种透明边框相当于内边距，因为元素的背景会延伸到边框区域（如果有可见背景的话）。

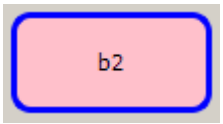
- border-radius

该属性用于设置控件的边框拐角弧度。默认的边框拐角弧度是 0，即直角，通过设置 border-radius 属性能定制出有弧度的控件，以按钮为例：

```
b2.styleSheet="border: 3px solid blue; border-radius: 0px";
b2.styleSheet="border: 3px solid blue; border-radius: 10px"
```

上面两行脚本执行完后的结果分别对应左右两幅效果图，从右图可看出 border-radius 属性发生了作用。





6.3.3 外边框

围绕在控件边框的空白区域是外边距(margin)。设置外边距会在元素外创建额外的“空白”。设置外边距的最简单的方法就是使用 margin 属性，值允许任何长度单位、百分数值甚至负值。可以设置为 auto。下面的声明在 h1 控件的各个边上设置了 2px 宽的空白：

```
#h1 {margin: 2px;}
```

下例为 h 控件的四个边分别定义了不同的外边距，单位是像素（px）：

```
#h1 {margin : 10px 0px 15px 5px;}
```

与内边距的设置相同，这些值的顺序是从上外边距（top）开始围着元素顺时针旋转的：top right bottom left。

margin 的默认值是 0，所以如果没有为 margin 声明一个值，就不会出现外边距。但是，在实际中，许多控件已经提供了预定的默认样式，外边距也不例外。当然，只要你特别作了声明，就会覆盖默认样式。

您可以使用下列任何一个属性来只设置相应的外边距，而不会直接影响所有其他外边距：

- margin-top
- margin-right
- margin-bottom
- margin-left

假设您希望把 p 元素的左外边距设置为 20px。不必使用 margin，而是可以采用以下方法：

```
#p {margin-left: 20px;}
```

另外，一个规则中可以使用多个这种单边属性，例如：

```
#h2 {
  margin-top: 20px;
  margin-right: 30px;
  margin-bottom: 30px;
  margin-left: 20px;
  }
```

当然，对于这种情况，使用 margin 更简介一些,下面语句效果跟上例一样。

```
#p {margin: 20px 30px 30px 20px;}
```

关于样式表的详细用法，可参考“**PS-LCD 软件速查表**”。

附一：常见问题

界面设计常见问题

如何为界面选择字体？

Windows 中的矢量字库均能作为界面字体类型，提升界面视觉效果。选中 Designer 属性区中的字体属性，点击右边小按钮，在弹出的选择框中选择所需字体、字号和风格。

如何让界面风格与众不同？

Designer 默认的色彩比较单一呆板，通过调色板 Palette 来改变色彩设置，让你设计的图形界面色彩风格与众不同。如：在对象面板中选中某个按钮，然后在属性面板中选择调色板属性，点击 自定义按钮，用户可以自己定义按钮的颜色特性。样式表是定制界面控件风格的一个好方法，可灵活实现各种界面样式，详情请参考第六章。

如何在界面中使用动画？

动画能让图形界面显得生动逼真，在 PS-LCD 上有两种方式实现动画效果：

- ✧ 直接使用 gif 格式图片。我们可以用 Windows 端专业的图片编辑工具如 photoshop 等生成所需 gif 图片，然后直接应用到图形界面中即可。由于 gif 仅支持 8bit 的图像， 最好使用在对动画图片颜色要求不高的场合。
- ✧ 使用 bmp，jpg，png，tiff 等格式的静态图片。例如：在界面运行时，利用定时器控件定时间隔执行脚本，如： xxx.value=1.png， xxx.value=2.png， …来实现动画。该种方式能显示 16/18bit 的高质量图片。

如果触摸屏定位不准，如何校准？

确保工具->界面配置中的触摸屏选项选中，在界面运行时，按住界面的无触摸事件区域（如：图片控件区域，或者无控件区域），并保持 5 秒以上，将自动进入校准程序。校准完毕后，校准数据自动保存在模组内，掉电不丢失。

如何在没有硬件键盘的情况下输入字母或者数字？

确保工具->界面配置中的软键盘选项选中，在界面运行时，点击文本框控件，输入软键

盘自动弹出，即可进行输入操作。

? 如何将 4x4 键盘与界面控件关联？

在设计界面阶段，请设置相关控件属性的内容->关联键值设置为需要关联的键值即可。界面在运行时，按下该键如同用手点击控件位置的触摸屏区域，如：将按钮 xxx 的关联键值设置为 1，那么按下 4x4 矩阵键盘的“1”键，按钮自动被按下。

? 如何定义供全局使用的函数？

在设计界面阶段，选中工具->全局脚本，即可打开 JS 脚本编辑器输入全局函数，该函数可供任何界面中的脚本调用，甚至可通过串口，主控制器直接远程调用， 如主控制器发送 func1(1,2) + 回车即可调用全局脚本中定义的 func1 函数。

? 我的界面很简单，但是为什么生成的 SPF 文件很大？

这极有可能是该界面选择的字体种类过多，请减少字体种类，建议一般产品界面字体少于 3 种，最好统一为一种字体，使界面简洁美观。

? 为什么在模拟器上调试好的界面，下载到 PS-LCD 运行，界面反应迟钝？

这可能是由于 PS-LCD 内存耗尽引起的。建议做如下检查和改进：

- ✧ 确认是否使用了远大于实际显示分辨率的图片，尤其 gif 动画图片。如：使用了一个 2048x1024 分辨率的 gif 动画图片，但是实际界面显示使用的是 300x150 分辨率。这将造成极大的系统内存资源浪费，造成界面运行缓慢。建议所有图片的分辨率与实际显示的分辨率相等或者相近；
- ✧ 是否过多的使用定时器。例如：在界面中定义了 5 个定时器，每个事件间隔为 50ms，某个定时器动作脚本都输入大量的刷新控件操作或者运算，这样系统将大部分资源用于处理这些频繁的定时器事件，造成响应缓慢；
- ✧ 是否将大量的界面设置为“快速显示”。设置为“快速显示”的界面和其上的控件将永远不会从内存释放，大量这样的界面可能造成系统内存消耗完毕；
- ✧ 界面中是否使用了过多的字体种类。建议一个产品上只使用一到两种字体，字体种类过多，不仅耗费系统资源，而且界面显得繁琐。

通讯常见问题

到底在我的工程中该选择哪种协议？

如果 PS-LCD 与外部控制单元通讯数据量小，而且 PS-LCD 发送到控制单元的数据较少，可使用 CTP 通讯协议，三条指令完成交互，避免制定协议的麻烦。但是，如果交互过程比较多或者外部控制器有自己专门协议，推荐使用自定义通讯协议，可最大限度降低通讯的数据量，灵活实现界面动作；

当我点击界面中的按钮，主控制器没有收到事件消息？

在 CTP 协议模式下，要让控件在状态变化时自动发送事件消息给主控制器，必须在 Designer 生成界面时，选中该界面按钮的**事件通知**属性；如果已经选中了，仍无法收到消息，请检查串口波特率是否设置正确。

主控制器如何确保发送的命令已经被 PS-LCD 正确执行？

在 CTP 通讯模式下，PS-LCD 在收到通讯命令后，会立刻执行，结束后会向主控制器返回结果。如果主控制器收到“C+”字符串，表示执行成功，可以发送下一条指令。“C-”表示执行失败，可尝试再次发送，或者检查错误原因。

在 CTP 协议模式下，如何取消 PS-LCD 命令执行结果的自动回复消息？

在 CTP 通讯模式下，PS-LCD 在收到命令并执行结束会向外部控制器返回执行结果，通过调用 **ctpSet(“reply”, 0)**函数可以取消此消息的自动回复。

PS-LCD 能自定义通讯协议吗？

PS-LCD 支持两种协议通讯模式，选择 UserDefine 通讯模式，即可实现自定义协议，这样可灵活与各种外部设备连接。

如何在 CTP 协议模式下自定义串口消息？

如果要定义某个界面事件发生时（如按下按钮），发送特定字符串/二进制编码给主控制器，可利用 **ctpBinTx()**和 **ctpAscTx()**函数自由定义，详情请参看“PS-LCD 软件速查表”中的用法。

在 CTP 通讯模式下，主控制器更新界面控件比较容易，但是 PS-LCD 自动发送过来的界面事件字符串在主控制器这边不好处理和识别，有什么改进办法吗？

可以取消控件的事件通知属性，直接在控件动作脚本中调用 **ctpBinTx** 或者 **ctpAscTx** 函数实现自定义发送数据至主控，可简化主控制器端的软件设计。

在 CTP 通讯模式下，如果主控制器想一次更新不同界面控件内容，将发送大量的类似 xxx.yyy=zzz 的 CTP 命令，有办法简化吗？

可以通过远程调用方法简化。如：主控直接发送字符串 fresh(0x55, 0x66, 0xx77); 给 PS-LCD，一次刷新三个控件。PS-LCD 上用脚本实现 fresh 函数如下：

```
Function fresh(p1, p2, p3)
{
    Xxx.value = p1;
    Yyy.text = p2;
    Zzz.text=p3;
}
```

这实际相当于，主控可远程调用 PS-LCD 上的脚本函数，传递实时参数，与在动作脚本中调用此函数实现同样的效果。