EzUIH 系列模块（无触摸版）

UART 串口版本

编程手册

V6.0 – 2022.11

深圳市鑫洪泰电子科技有限公司技术资料

版权声明

更新纪录：

2015/10 V21 针对 EzUIH 系列 V10 版本带触摸模块

2017/02 V30 针对 EzUIH 系列 V30 版本带触摸模块

2017/06 V31 针对 EzUIH 系列 V31 版本（固件）带触摸模块

2017/07 V32 针对 EzUIH 系列 V32 版本（固件）带触摸模块

- 增加按钮类控件的长按键消息设置；
- 增加波形控件的步进值设置；
- 添加部分控件配置数据关联控件功能；
- 控件响应消息添加对时间、日期显示控件的操作；

2018/03 V50 针对 EzUIH 系列 V50 版本（固件）带触摸模块

- 增加滑动条控件（SliderCtrl）的配置；
- 增加进度条控件位图重绘属性选择；
- 增加进度条控件配置选项，允许配置叠加一个数值控件或字符串控件；

2018/05 V51 针对 EzUIH 系列 V51 版本（固件）带触摸模块

- 增加二维码显示控件的配置；

2018/08 V52 针对 EzUIH 系列 V52 版本（固件）带触摸模块

- 增加界面及部分控件触发的 GUI 服引擎消息数据包的发送使能开关设置；

2019/10 V53 针对 EzUIH 系列 V53 版本（固件）带触摸模块

- 增加位图按钮控件及位图动画控件的底图绘图模式设置；
- 增加字符串控件的控件关联配置；
- 增加一次对多个连续控件数值/状态写入指令说明；

2022/10 V60 针对 EzUIH(S)系列 V10 版本（固件）带无触摸模块

- 增加字符背景图层覆盖模式及字符擦除模式；
- 增加位图按钮、位图动画控件控件底图模式；
- 增加矩形绘制圆角属性；增加圆形（非实心圆）跟随绘图模式线宽设置进行绘制属性；
- 增加圆弧绘制、缺口圆绘制属性；
- 增加串口传输动态 JPG 图片文件显示功能；
- 增加模块 USB 模式锁定/解锁功能；
- 增加矢量字库支持；
- 增加动态加载区域按钮、位图按钮控件功能；
- 增加位图动画控件配置图片外置（不加载入资源文件，但要预存于模块 U 盘）属性配置；
- 增加波形控件绘制柱形图属性。

目　　录

1 EzUIH(S)系列模块简介

1.1 特点

EzUIH(S)系列彩色液晶模块为鑫洪泰研发、生产销售的智能型人机界面显示模块;无触摸版 EzUIH(S)系列模块基于 TFT 显示屏整合了自成体系的 GUI 系统，结合界面开发工具软件（EzUITool）可实现用户界面设计无代码的目标。而 EzUIH(S)系列为 EzUI 系列的高性价比版本;EzUIH(S)系列模块兼容全部 EzUI、EzUIH 模块的功能。

EzUIH(S)系列模块最大的特色为使用简便、功能丰富，用户对其使用可简可繁。用户可以使用界面开发工具软件（EzUITool）进行人机界面的设计、编辑、控件配置、响应设置等，将工具软件生成的资源文件复制到 EzUIH(S)系列模块之中，便可达成所需人机交互界面的设计制作，而无需用户单片机或其它控制器的编程控制。而为满足有特殊要求的用户，EzUIH(S)系列模块还保留有串口显示控制指令（如绘直线、矩形、圆、字符串显示等），以便于用户可以更自由的对模块显示进行直接控制。

EzUIH(S)系列模块的命名规则为“EzUIH**XXX**(S)”，最前面的字符“EzUIH”表示该模块为 EzUIH 系列模块，后面三位数字为模块的屏幕尺寸；(S)表示最新的 EzUIH(S)系列（部份模块为(SK)，表示适装外壳版）。

V10 版固件的无触摸版 EzUIH(S)系列模块（UART 串口版本）具备以下特性：

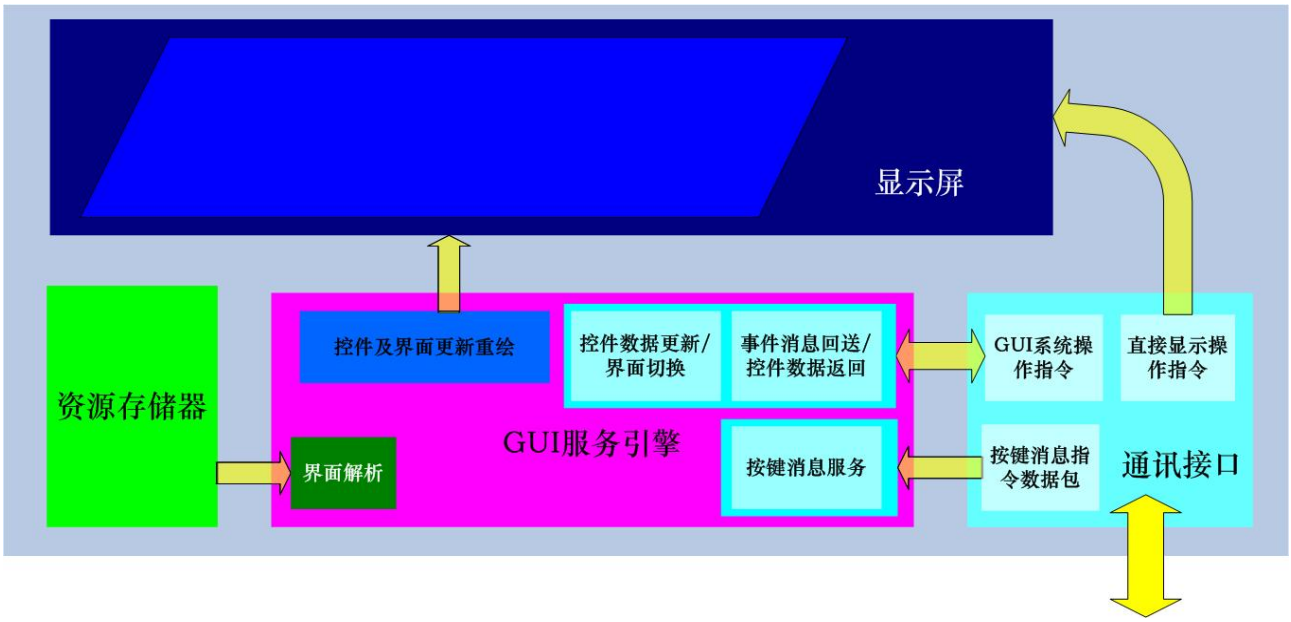
- UART 串行接口（TTL 电平/RS232 电平）方式，支持 9600~1000000bps;
- 标准版模块内置 1G 位（120M bytes 可用）大小的资源存储器;
- **选配 RTC 实时时钟模块**;
- 模块内部自带 6×10、8×16、10×20、16×32 点 ASCII 码西文字库;
- 支持基本绘图指令（绘点、直线、矩形、圆形、字符串显示、位图显示、**Jpeg 图片显示**等）;
- 资源文件支持加载点阵字库（GBK2312 二级汉字库、ASCII 西文字库等）;
- 资源文件支持加载 BMP 位图、Jpeg 图片;
- 支持矢量字库（多国语言、抗锯齿效果）;
- 支持界面切换效果设置，如透明度渐入、随机快渐入、百叶窗渐入、缓冲区快速切换;
- 支持区域按钮控件，多种属性配置，控件消息响应可配置;
- 支持位图按钮控件，多种属性配置，控件消息响应可配置;
- 支持数值控件（整数、浮点数均可），多种属性配置，支持数值输入，支持数据关联同步;
- 支持字符串控件（中英文均可），多种属性配置，支持中英文字符串输入;
- 支持下拉选择控件，控件消息响应可配置，支持数据关联同步;
- 支持波形控件，允许同一 ID 号控件内最多四条波形线，支持柱形图显示;
- 支持进度条控件，控件消息响应可配置，支持叠加显示数值/字符串控件，支持数据关联同步;
- 支持位图动画控件，控件消息响应可配置，支持数据关联同步，支持配置图片外置资源文件;

- 支持时间显示、日期显示控件；
- 支持表盘显示控件，支持数据关联同步；
- 支持滑动条控件，控件消息响应可配置，支持数据关联同步；
- 支持二维码显示控件，二维码生成版本及纠错等级自动选择，自动放大显示；

1.2 功能框图

无触摸版的 EzUIH(S)系列模块 UART 串口版本（以下简称 EzUIH 系列模块）主要由四大功能模块构成：显示模块、资源存储器、GUI 服务引擎、UART 通讯接口。

下图比较全面的描述了 EzUIH 系列模块的功能结构示意：



资源存储器：模块将在资源存储器中保存后缀为“.ers”的资源文件，资源文件中预存有图像、汉字库、数字输入控件资源、字符串输入控件资源等；并在其中保存用户设计的界面信息，以及各个界面中定义的控件、显示控制指令等；资源文件由用户使用界面开发工具软件进行编辑、生成。

GUI 服务引擎：该块服务引擎将会从资源存储器中读取资源文件，并解析文件中保存的界面信息、控件定义、显示控制指令等，进行相应的显示；引擎将从通讯接口获取用户 MCU 发来的按键消息，并根据界面以及控件的定义，进行相应的显示响应，比如完成数值输入、字符串输入、控件响应等；引擎根据界面中定义的以唯一标识 ID 号的控件进行控件管理，用户可以通过通讯接口按指定的 ID 号对指定控件进行数据读取；此外，用户也可通过通讯接口对指定 ID 号的控件进行数据更新，引擎将会根据更新的情况进行相应的显示刷新操作。

UART 通讯接口：模块对外部引出 UART 接口，用户可使用单片机或其它控制器按照指定的通讯协议对模块进行指令控制，主要分为三大类指令，一为 GUI 系统操作指令：按 ID 号对模块上定义的控件进行读取、写入数据操作；二为直接显示操作指令：可直接通过相应的指令对模块的当前显示进行操作；三为按键消息指令：由用户 MCU 发送给模块的按键消息指令，按键的键值和事件类型需按定义给予，包含 TAB（切换选择键）、Entry（确定键）、方向键、数字键等。

1.3 工作流程

使用 EzUITool 编辑设计界面：

EzUITool 是为 EzUI 系列模块（包括 EzUIH(S)、EzUIC、EzUILet 系列）而设计的一款资源整合、界面编辑、控件配置的工具软件，用户可使用该工具完成基本的显示界面设计，以及配置各个显示界面的显示内容、控件配置，以及控件之间的消息响应机制。

EzUITool 工具可以生成一个资源文件，其中包含有预置显示资源，而预置显示资源主要为用户界面所需的图片、字库、通用控件资源等构成；这些预置显示资源都是整个用户界面的各个基本材料。

用户可在 EzUITool 工具上为资源文件编辑多个显示界面，每个界面都有唯一的界面索引号，模块上电后，将会从资源文件中定义的启动显示界面开始显示。

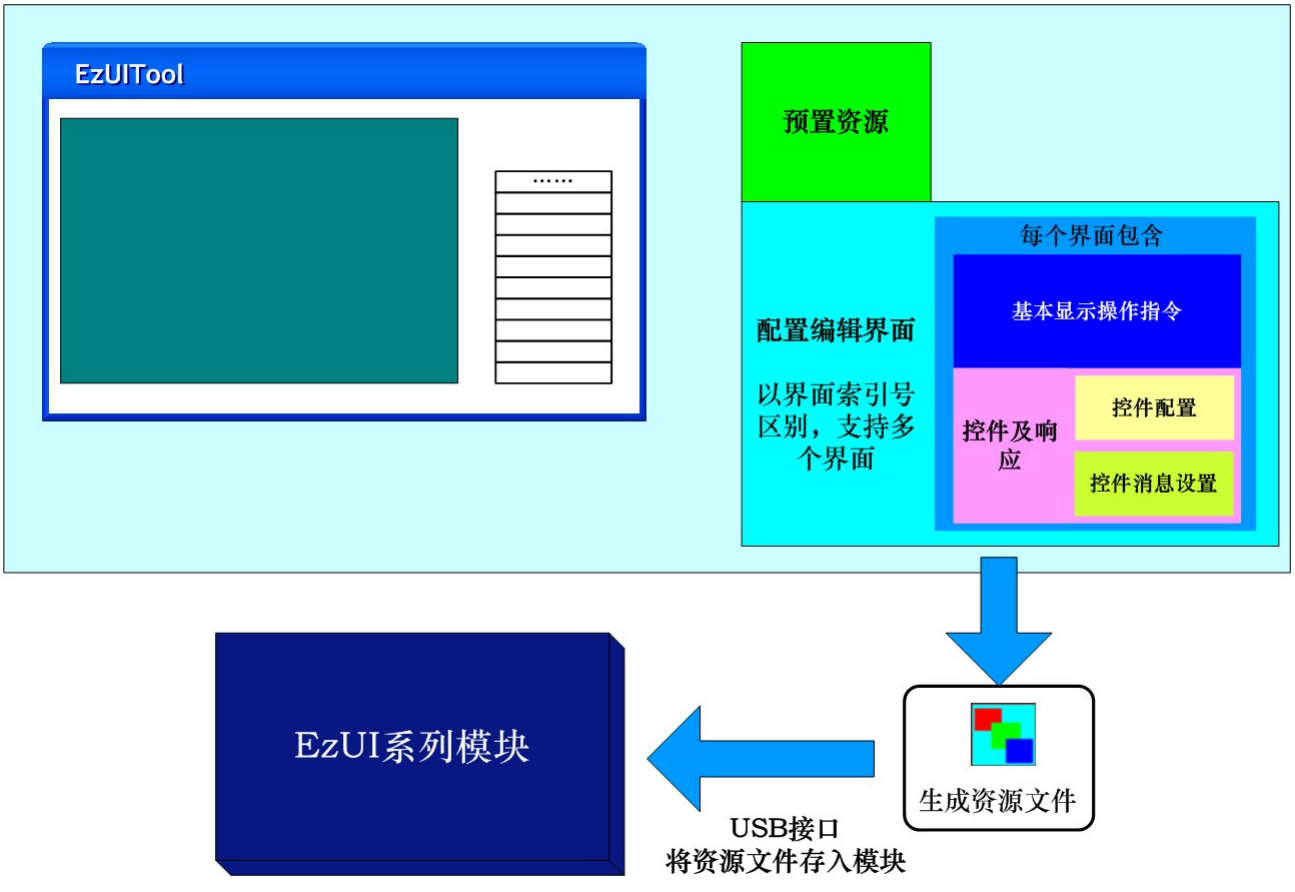
每个界面将包含有三类数据：

1、 基本显示操作指令：包括背光设置、绘图操作、字符串显示、位图显示等基本显示操作；

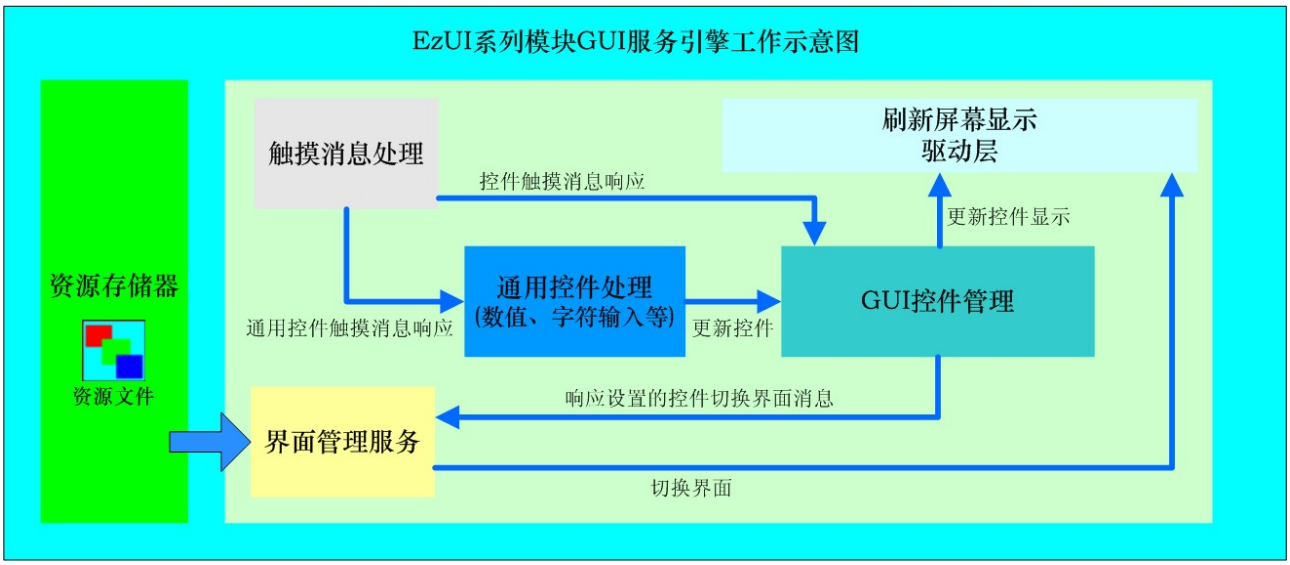
2、 控件配置：该块数据将配置所属界面的控件，在同一资源文件中的所有控件将会统一以 ID 号来管理，只有当前显示界面之中的控件会处于活动状态，即允许其显示更新以及接受触摸屏消息；

3、 控件消息设置：也即控件响应消息设置，每一条消息设置都会从属于相应的 ID 号的控件，表示该 ID 号的控件在设置相符的条件下时，可以响应相应的消息；比如可以设置某一个 ID 号控件有触摸事件发生时，切换界面或控制其它 ID 号的控件数据。

EzUITool 工具将最终生成一个资源文件，该文件可以通过 EzUI 系列模块的 USB 接口存入模块的资源存储器当中。

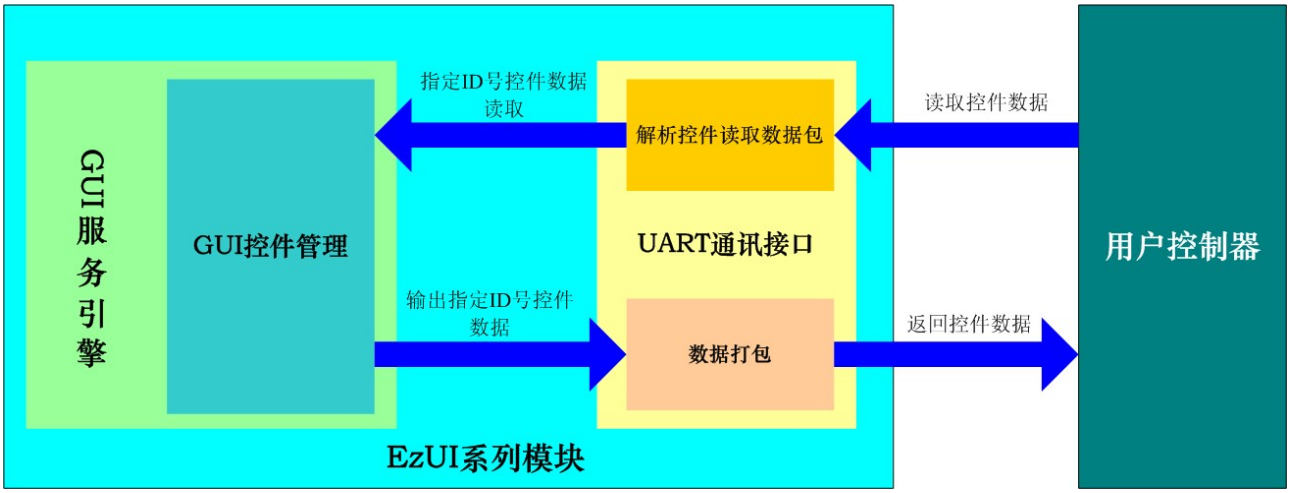


模块 GUI 服务引擎工作示意：



EzUI 系列（包括 EzUIH(S)、EzUIC、EzUILet 系列）模块的 GUI 服务引擎让模块脱离用户的控制器进行人机界面的交互；用户使用 EzUITool 工具编辑制作的资源文件存入模块的资源存储器后，模块在上电后便可开始自主工作（当然也允许用户控制进行显示干预）；显示界面的切换、更新显示，控件数据的数据输入、更新显示都可以自主的完成。

用户控制器获取控件数据：



用户控制器（单片机或其它有 UART 接口的处理器）可以通过 EzUI 系列模块的 UART 通讯接口对模块之中的控件数据、状态进行读取；只要按指定的通讯协议，指定要读取操作的控件 ID 号，便可随时读取其数据、状态。

用户更新控件数据：

用户控制器也可通过 UART 通讯接口对模块之中的指定 ID 号的控件进行数据写入更新，用户无需关心这些控件的显示刷新以及是否处于当前显示界面，模块的 GUI 服务引擎会自动进行判别处理。

2 EzUIH 系列模块资源文件

EzUIH 系列模块的显示内容以及显示界面都与资源文件紧密关联，或者说，模块上保存的资源文件基本上决定了模块显示内容以及界面。

EzUIH 系列模块的资源文件后缀为“.ers”，该文件由 EzUITool 工具软件制作生成，将会保存于 EzUIH 系列模块之中的资源存储器当中。

2.1 资源文件构成

资源文件中主要包含有预置显示资源和显示界面这两大部份数据；预置显示资源主要为用户界面所需的图片、字库、通用控件资源等构成；这些预置显示资源都是整个用户界面的各个基本材料。而显示界面的数据主要为用户所定义的各个显示界面以及显示界面当中具体的显示内容等数据。资源文件中，最多允许加载入 **400** 个资源。

资源文件当中的显示界面数据可以包含多个显示界面，每个界面都有唯一的界面索引号，模块上电后，将会默认从索引号为 0 的界面开始显示。资源文件中，最多允许 **200** 个界面。

每个界面将包含有三类数据：

- 基本显示操作指令：包括背光设置、绘图操作、字符串显示、图像显示等基本显示操作；
- 控件配置：该块数据将配置所属界面的控件，在同一资源文件中的所有控件将会统一以 ID 号来管理，只有当前显示界面之中的控件会处于活动状态，即允许其显示更新以及用户发送的键值消息进行操控；
- 控件消息设置：也即控件响应消息设置，每一条消息设置都会从属于相应的 ID 号的控件，表示该 ID 号的控件在设置相符的条件下时，可以响应相应的消息；比如可以设置某一个 ID 号控件有事件发生时，切换界面或控制其它 ID 号的控件数据。

2.2 资源文件中的显示资源

资源文件中的预置显示资源是整个用户界面的基本材料，包括：ASCII 西文字库、中文字库（GBK2312）、BMP 位图、**Jpeg 图像**。所有加载到资源文件中的显示资源项都有一个索引号（或称编号），这样用户对模块进行显示操作或者显示设置时，将通过这些资源项的索引号的操作，来达到显示图片、设置字库等目的。

2.3 资源文件中的显示界面

资源文件中可以由用户在 EzUITool 工具软件之中进行创建、编辑多个显示界面；实际上相当于预置显示界面，每个显示界面都有唯一的索引号来进行标识，这样 EzUIH 系列模块内部的 GUI 服务引擎将会以这些唯一索引号的界面进行界面的管理。

每个显示界面包含有三大块数据内容，分别是：基本显示操作指令、控件配置和控件消息设置。一个资源文件中可以预置最多 200 个界面，模块在复位后将会从资源文件中定义的启动显示界面开始显示；用户随时切换当前显示的界面，而切换界面的途径有两种，一为用户通过模块的通讯接口给模块发送界面切换指令来实现；二则是通过界面中的控件响应消息配置来实现。

任何时候，只能有一个界面处于当前显示状态，这是一条基本原则。切换界面时，EzUI 模块会先执行要显示的界面中的基本显示操作指令，如点亮背光、清屏、显示位图等操作；然后 EzUI 模块会根据该界面中所定义的控件进行控件重绘，并使能该界面所定义的控件，使它们处于活动状态（即允许显示更新、允许响应按键消息事件）。最后，EzUI 模块的 GUI 服务引擎只对当前处于显示的界面中所定义的控件进行按键消息响应服务，并根据情况响应该界面所配置的控件消息（由此来实现控件间的联动、切换界面等）。

当一个界面处于当前显示状态时，也即表示当前 GUI 服务引擎只会允许该界面中所定义的控件响应按键消息，以及显示更新；而此时其它界面中所定义的控件将不响应按键消息，以及不允许进行显示的更新；但是，所有的控件在任何时候都允许对其进行数值写入或数值读取操作。

2.4 资源文件存入模块

将资源文件存入 EzUIH 系列模块当中的资源存储器之中非常简单，只需将模块配置到 USB 模式（具体配置方法请参考具体型号的模块数据手册），然后重新对模块上电、复位，再将模块与 PC 机通过 USB 线相连，即可在 PC 机识别出一个 U 盘，将资源文件复制到该盘之中的根目录之下即可；不过需要注意，只允许有**一个资源文件**保存于模块上的资源存储器，且要求资源文件名称为**英文名**称。

3 EzUIH 系列模块工作原理

EzUIH 系列模块与资源文件相关密切，模块上显示的中文汉字以及位图资源以及控件设置、控件消息配置等都靠资源文件中的数据定义；可以将 EzUIH 系列模块看作是执行资源文件预置显示的显示平台。接下来将介绍 EzUIH 系列模块的工作原理，并介绍模块所支持的各项功能、控件特性等。

3.1 资源存储器资源文件相关

资源文件除了有前面所述的两大类数据（预置显示资源、界面数据）之外，还有几项用以模块全局设置的地方，其中一项为默认 ASCII 字库设置，另一为默认中文字库设置，这两项设置都可以在 EzUITool 工具软件中在制作资源文件时设置好，EzUIH 系列模块上电后，将会以它们作为默认设置。

3.1.1 字库管理

EzUIH 系列模块（EzUIH(S)，不包含原 EzUIH 版本）支持系统**矢量字库**以及**点阵字库**（包括 ASCII 西文字库以及中文汉字库），对于点阵字库，模块内置有几个（通常是 4 个）ASCII 西文字库，当内置的不合适用户使用时，也可以选择在资源文件中加载所制作的 ASCII 字库资源；而 EzUIH 系列模块并无内置中文字库，用户需要使用中文显示时，可在资源文件中加载所制作的中文字库资源。EzUIH 系列查块在显示字符串之前，需选择好所使用的字库（选用矢量字库或点阵字库，二者选其一）；当用户选择使用点阵字库时，可分别对 ASCII 西文字库以及中文字库进行选择配置，因为两者相互并不包含对方。

EzUIH 系列模块（EzUIH(S)，不包含原 EzUIH 版本）支持**矢量字库**；用户选用矢量字库时，字符显示将会以设定的字符号（对应字符宽高像素点数值）进行显示，可按需要随时设置任意大小的字符进行显示。矢量字库需要将矢量字库支持文件存放于模块资源存储器当中，且在资源文件（.ers）创建时选择好与矢量字库对应的语言选项。

而使用矢量字库时，比如语言为中文的矢量字库，通常会将 ASCII 码的西文字符也一起包含在内，并不需要再多作设置，只需设置好使用的字号（对应为像素点）即可。

EzUIH 系列模块内置的 ASCII 西文字库资源在使用时，其索引号（16 位 unsigned short 型数据）的最高位为 1，用以标明该字库为模块内置西文字库；如要使用内置的第 1 号 ASCII 西文字库时，设置时需指明该字库索引号为“0x8001”。而加载在资源文件中的 ASCII 字库以及中文字库，将以资源列表中的索引号为准。具体有关字库选择设置的内容，参考后续模块的普通显示操作指令的介绍。

EzUIH 系列模块所支持的点阵字库资源要求如下：

字库（也就是字模的集全）的数据采用了与一般的单色点阵 LCD 一样的数据组成方式，即字模当中的一个位代表 LCD 显示中的一个像素点，取点方式为**从左到右，自上到下**的顺序。对于这点，ASCII 码英文字库的字模和中文字库中的字模要求一样。

字模采用了以 Byte 为单位的位流结构，即当一行取点不为 8 的整数倍时，补齐数据至 8 位，无用位填零。

字库文件的命名请按照上节中的要求，否则无法正常加载。

而 ASCII 英文字库的字符个数为 256 个（实际显示的有效 ASCII 码为：32~127）。

EzUIH 系列模块支持后续为.ttf 的矢量字库，该字库文件需预存于模块 U 盘（根目录下“SysFontLib”文件夹内）当中，以待配置使用。模块在工作时，仅会使用一个矢量字库，当存入多个矢量字库时，以模块实际查询存储器时获取到的第一个字库文件为准。

3.1.2 位图资源

EzUIH 系列模块的资源文件支持 BMP 位图文件、Jpeg 图片文件，BMP 位图文件分别支持 24 位真彩色 BMP 位图、8 位色（256 色）BMP 位图、4 位色（16 色）BMP 位图以及单色 BMP 位图文件。需要注意的是，24 位真彩色 BMP 位图实际上加载到资源文件后，会将其取模，最终形成 16 位色（65536 色）的位图资源。

3.2 基本显示规则

3.2.1 图层原则

EzUIH 系列模块内部提供了两个独立的图层，分别是显示图层以及背景图层，用户并不需要操作背景图层，而背景图层也不会与显示图层有叠加显示的关系；可以这样定义：

显示图层：当前屏上的显示内容都来源于显示图层，用户对显示的操作，实际上是直接操作了显示图层的内容，模块上的屏幕显示的每一个像素点都由一个显示图层的存储单元来保存，每个单元为 16 位二进制位的数据，即 unsigned short 型的数据；数据的排列规则是 RGB565；用户可通过对应屏上显示的 X 轴和 Y 轴坐标来操作指定的像素点显示内容。

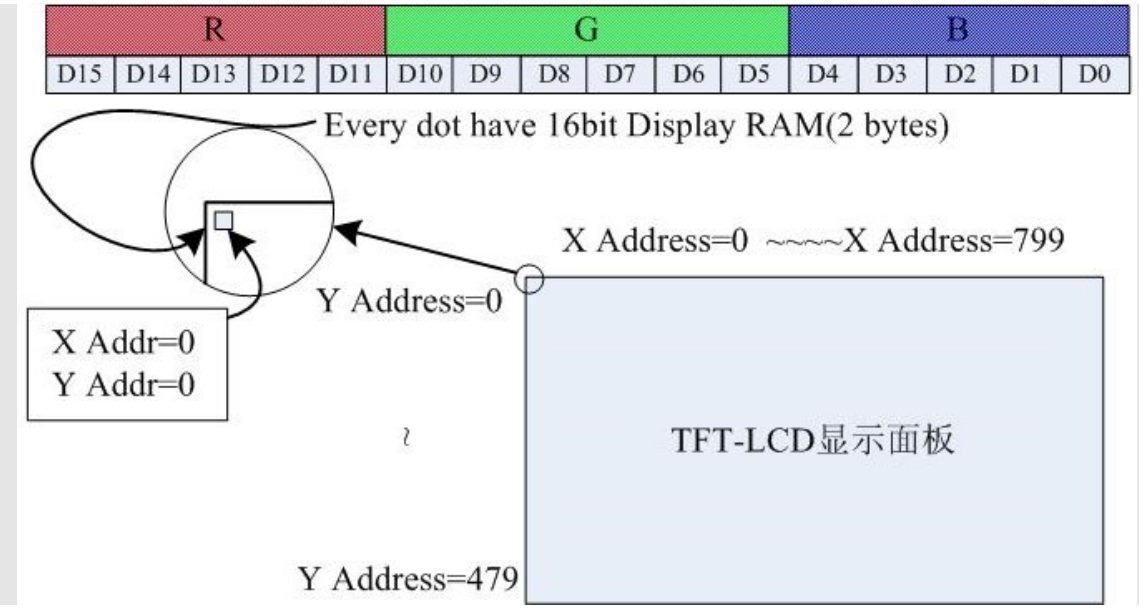
背景图层：该图层的数据结构与显示图层一样，只是该图层用于保存用户界面的背景数据，通常用户不需要操作该图层的数据；仅用于 EzUI 系列模块的界面管理及 GUI 服务引擎在处理界面刷新、弹出窗口消除等内部处理。

以 7 寸的模块“EzUIH070(S)”为例，说明图层中的数据与显示屏上的像素点的对应关系：

EzUIH070(S)模块的 7.0 英寸 TFT-LCD 显示面板上，共分布着 800×480 个像素点，而模块内部的 TFT-LCD 驱动控制芯片内置有与这些像素点对应的显示数据 RAM（简称显存）。模块中每个像素点需要 16 位的数据（即 2 字节长度）来表示该点的 RGB 颜色信息，所以模块内置的显存共有 800 × 480 × 16bit 的空间，通常我们以字节（byte）来描述其的大小。

而为了便于索引操作，模块将所有的显存地址分为 X 轴地址（X Address）和 Y 轴地址（Y Address），分别可以寻址的范围为 X Address=0~799，Y Address = 0~479，X Address 和 Y Address 交叉对应着一个显存单元（2byte）；模块的显示操作非常简便，模块封装了基本的绘图功能接口，只需要调用相应的绘图指令，指明 X 和 Y 轴地址，便可以快速的修改模块上的显示内容。

EzUIH070 模块的像素点与显存对应关系下图所示：



3.2.2 基本绘图设置及操作

EzUIH 系列模块有一个绘图色(前景色)的概念，而模块所支持的基本绘图指令/操作，如绘点、绘直线、绘矩形框、绘矩形、绘圆形框、绘圆形等，都是要以当前模块设置的绘图色来绘制的；而 EzUI 系列模块的绘图色用户可以通过发送对应的指令来设置，该指令需要一个 16 位长度的颜色参数，不同的数值代表不同的颜色。

颜色的数据与 TFT 屏上像素点的颜色数据类似，16 位长度的颜色数据排列着三基色的指数，即 R、G、B（红、绿、蓝）；分别占用 5、6、5 个二进制位。如下图所示：

比如，绘图色设置的数值为 0xF800 时，为红色；值为 0x07E0 时，为绿色；而黑色为 0x0000，白色为 0xFFFF。

而在界面当中，每个界面的普通显示指令预置那里，都可以随时对绘图色进行设置。

绘图色的设置指令/操作含有另一参数，为绘图线宽设置，该参数最大值为 8，最小值为 1，表示绘图操作时使用的线宽，该参数决定绘制直线、绘制矩形框的操作。

绘图色的设置/操作，直到下一次设置/操作之前，都不会改变。

3.2.3 字符显示设置及操作

字符显示设置涉及两类设置/操作，分别为：字库设置、字符覆盖模式设置。

字库设置：

EzUIH 系列模块在固件中预置有几种（视具体的模块而定）字号的 ASCII 字符可供用户选择，而此外 EzUIH 系列模块的资源存储器也支持 ASCII 西文字库资源及中文字库（GB2312）；统称为点阵字库。EzUIH 系列模块的点阵字库将 ASCII 西文字库和中文字库分开管理，也即显示 ASCII 字符时使用的是所设置的 ASCII 字库，而显示中文字符时，使用的是所设置的中文字库；但两种字库使用的字符色统一管理，

也即当前只能有一种字符色在作用。

EzUIH 系列模块（EzUIH(S)，不包含原 EzUIH 版本）支持**矢量字库**；用户选用矢量字库时，字符显示将会以设定的字符号（对应字符宽高像素点数值）进行显示，可按需要随时设置任意大小的字符进行显示。矢量字库需要将矢量字库支持文件存放于模块资源存储器当中，且在资源文件（.ers）创建时选择好与矢量字库对应的语言选项。

字库设置还需给定一个字符显示颜色的参数，该参数也为 16 位的数据，数据规则为 RGB565 格式。

字符覆盖模式设置：

EzUIH 系列模块提供了一个字符覆盖模式的设置，该设置/操作提供了字符显示与原屏上显示的背景叠加关系的设置；设置/操作需要有 3 个字节的数据，分别是 1 个字节的覆盖模式设置，以及 2 个字节（16 位长度）的字符覆盖色设置；覆盖模式，可以为 0、1、2 或 3，字符覆盖色为覆盖模式下字符显示的背景擦除时使用的颜色（该颜色的数据结构与前景色的数据结构类似，RGB565 的色彩数据）。

- 字符覆盖模式为 0 时，表示在显示字符时，对要显示字符的目标位置区域的当前显示内容不作擦除，仅显示该字符的字符线条；
- 字符覆盖模式为 1 时，在显示字符时将会使用指令设置的覆盖模式背景擦除色对该字符的显示区域进行擦除后才以当前设置的字符色来显示字符线条；
- 字符覆盖模式为 2 时，可视为背景图层覆盖模式，在显示字符时将会对该字符的显示区域以背景图层中相应区域的图像调入显示图层后才以当前设置的字符色来显示字符线条；
- 字符覆盖模式为 3 时，为背景图层擦除模式，在操作中将使用当前背景图层的数据擦除字符。

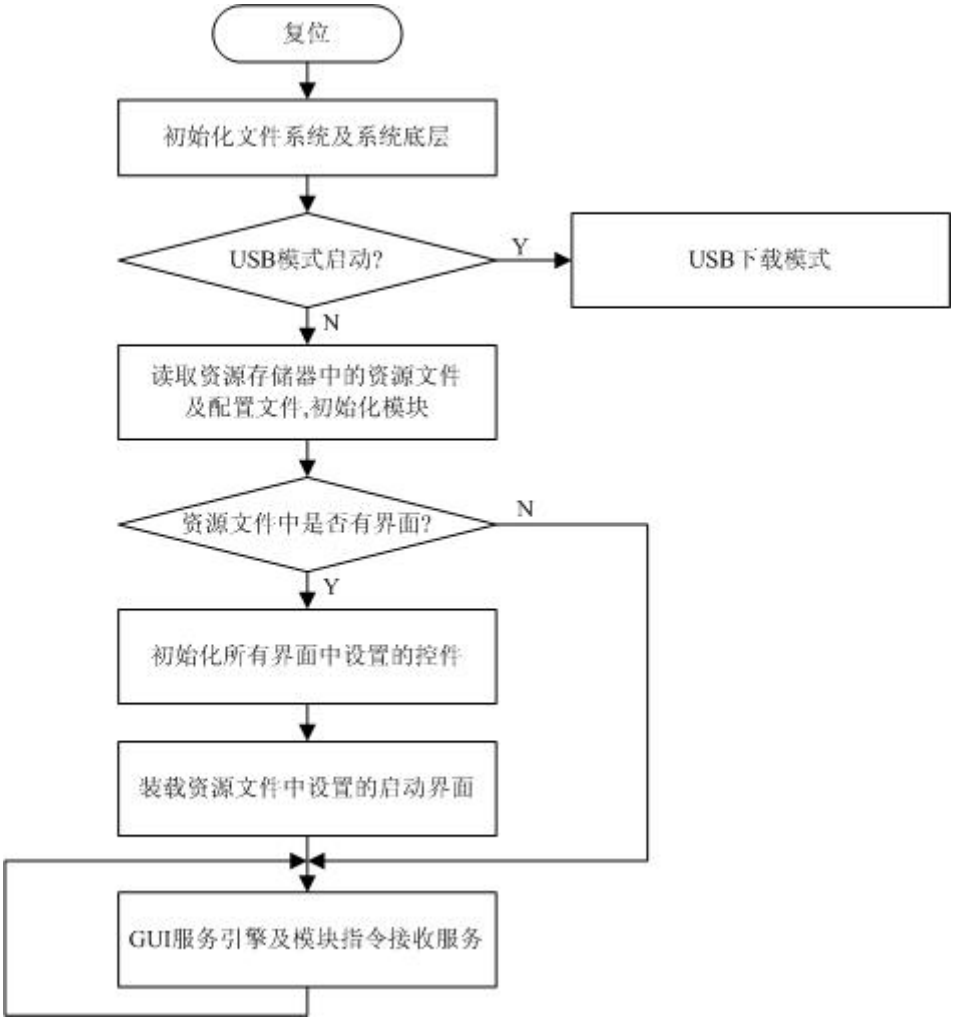
下图所示，所显示的 ASCII 码字符选择为固件自带 ASCII 字符库，序号为 3；字符色为 0xFFE0 时，而字符覆盖模式设置为 0 时的显示‘A’和‘a’；而将字符覆盖模式设置为 1，字符覆盖背景擦除色为 0x0000 时，显示字符串：“16*32”。

3.3 界面管理

EzUIH 系列模块的 GUI 服务引擎将会以界面来刷新显示以及管理活动控件；任何情况下也仅允许一个界面处于当前显示，该界面中所设置的控件将处于活动状态，表示它们可以响应触摸事件，以及更新显示。

3.3.1 系统初始化加载

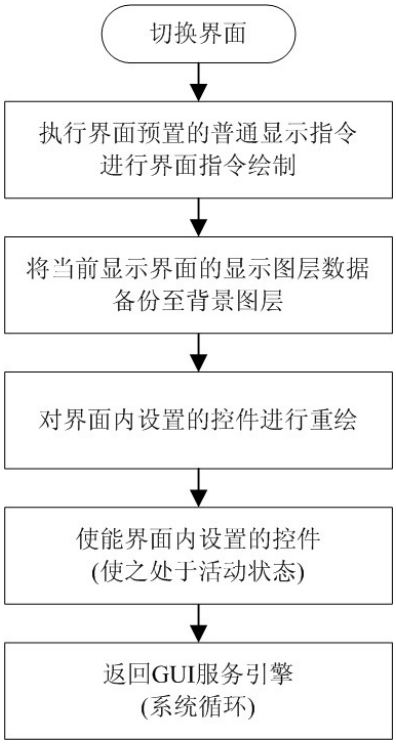
EzUIH 系列模块在上电后将会按以下流程进行系统的初始化：



- 系统复位完后后，会判断模块是否启动了 USB 模式，如启动，则进入 USB 下载模式，下载完资源文件后，用户需要将模块设置到正常显示模式（即取消 USB 模式），再重新上电、复位，才可进入正常的显示模式。
- 模块从资源存储器当中读取并分析模块配置文件，并根据配置文件的设置对模块进行基本的初始化，其中，包含有 UART 接口波特率的设置。
- 资源文件中如有界面，则模块会在读取分析完资源文件后，将所有界面中所设置的所有控件进行初始化，该操作完成后，用户才可对模块上的资源文件中设置的控件进行数据初始化、数据读取等操作。
- 资源文件中如有界面，则模块会在初始化控件完成后，将资源文件中定义的启动界面装载到当前显示，也即从该界面开始运行。

3.3.2 界面更新绘制原则

界面的切换或装载时都会按照同样的规则进行界面的装载，具体流程如下图所示：



切换界面时，EzUIH 系列模块会从要装载的界面当中获取该界面预置的普通显示指令，并按照它们的顺序进行执行显示操作；在这个过程中，界面预置的普通显示指令由 EzUITool 工具软件在制作资源时对界面进行编辑预置，包含背光设置、绘图色设置、绘直线、绘矩形框/实心矩形、字库设置、字符串显示、位图显示。

资源文件中允许存在多个界面，以及多个标准控件，界面以界面索引号为标识，而控件则以控件的 ID 号作为标识，它们都是唯一的（在同一个资源文件当中）。标准控件需要在界面当中进行设置，当装载某一个界面时，该界面内的控件将会进行重绘，即全部的更新该界面上所设置控件的显示。

当装载某一个界面时，EzUIH 系列模块将会先清除当前活动控件，然后将该界面所设置的控件设置为活动状态，也即表示，在任何一个时候，只会允许一个界面所设置的控件处于活动状态；控件处于活动状态表示该控件可以响应按键消息事件，可以更新显示。

3.3.3 控件数据关联

EzUIH 系列模块允许部份具备数值或状态的控件进行控件间的数据关联同步，在配置控件时，可以设置指定的数据关联控件 ID 号，有效设置后，所涉及的控件在模块端触摸输入数据或被 MCU 设置数据后，可自动更新当前数值或状态。

例如，ID 号为 100 的数值控件设置数据关联控件为 ID 号 101 的数值控件，而 ID 号 101 的数值控件设置数据关联控件为 ID 号 102 的进度条控件，此时 ID 号 100、101、102 的三个控件将为数据关联关系，用户无论通过触摸或通过 MCU 发送控件数值写入指令，对其中任意一个控件进行数值操作后，三个控件的数据将一起更新。

控件数据关联的功能允许处于不同界面的不同类型控件进行相互关联。

比如 A 控件设置关联 B 控件，B 控件关联 C 控件，C 控件关联 D 控件，而 E 控件关联 C 控件（注意，是 E 控件关联 C 控件），此时 A、B、C、D、E 控件均为数据关联的状态，用户在模块上触摸输入其中任意一个控件的数值或状态，或者通过串口设置其中任意一个控件的数值或状态，都会使用 A、B、C、D、E 控件同步的更新同样的数值（或状态）。

3.4 控件详述

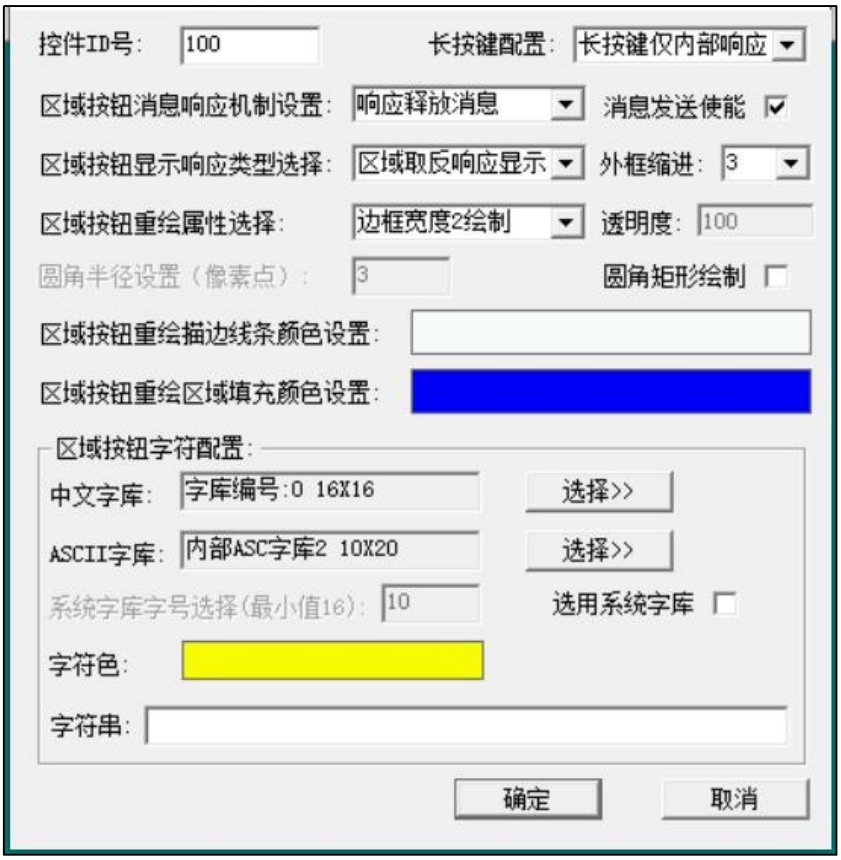
EzUIH 系列模块支持多种标准控件，在此为更好的说明这些控件的特性，将会结合 EzUITool 工具软件的界面进行详细说明。

3.4.1 区域按钮控件

区域按钮控件（TouchArea）是一个定义好区域的触摸按钮，其有多种属性可配置，并可实现多种显示、响应效果；该控件支持控件消息配置。

注：区域按钮控件在不带触摸屏的 EzUIH 系列模块需要由用户 MCU 发送来的按键消息指令来进行响应服务。

在 EzUITool 工具中，区域按钮控件的配置画面如下图所示：



控件 ID 号：控件的唯一标识，由工具自动生成，标准控件的 ID 号从 100 开始。

消息发送使能：该项设置为区域按钮控件所触发消息数据通过串口发送的使能开关，默认为勾选状态，也即控件触发消息时将会通过模块串口发送消息数据包（GUI 服务引擎消息数据包）；取消勾选后，该控件将不会再触发消息数据包的发送。

长按键配置：可以在此设置控件的长按键消息触发的机制，备选的选项如下：

- 不支持长按键：表示该控件不支持长按键功能；
- 长按键仅内部响应：表示控件长按键消息仅供模块内部系统驱动控件消息使用，不会连续触发消息数据包（通过串口）的发送；
- 长按键连续消息：表示该控件在长按键时，除了连续触发消息供模块内部系统驱动控件消息之

外，在消息发送使能勾选的前提下还会连续的发送消息数据包给用户 MCU。

消息响应机制设置：

该项的属性设置指的是在消息发送使能勾选的前提下，设置控件响应触摸事件返回消息的机制，返回对像为用户 MCU，通过 UART 接口，以特定的数据包格式回送给用户 MCU；该项配置有三项可选：

- 响应按下消息：表示控件只有当被触摸按下时，才会返回消息给用户 MCU；
- 响应释放消息：表示控件只有当触摸释放时，才会返回消息给用户 MCU；
- 响应按下及释放消息：表示控件会在触摸按下及触摸释放时都会返回消息给用户 MCU。

显示响应类型选择：

该设置为区域按钮响应按键消息事件时，显示作出相应变化的方式；可选的选项如下：

- 无响应：表示该区域按钮在被按下时，不作显示响应，即显示无任何变化；
- 外框宽度 1 取反显示：表示该区域按钮被按下时，将会在指定的外框以宽度为 1 的矩形框进行取反显示；（指定的外框的区域为所设置的区域按钮控件的区域进行指定像素点的缩进值后形成的矩形框，所指定的缩进值由“外框缩进”进行选择设置）
- 外框宽度 2~8 取反显示：表示该区域按钮被按下时，将会在指定的外框以线宽为 2~8 的值进行矩形框取反显示；
- 区域取反响应显示：表示该区域按钮被按下时，将会在指定的外框区域内进行取后显示。

外框缩进：

此处的外框缩进，指的是区域按钮控件显示响应时相对该区域按钮的实际定义区域进行缩进后的“外框”，其值可设置为 0~8。

重绘属性选择：

控件进行重绘时，会按照所设置的重绘属性进行绘制控件，区域按钮控件的重绘属性可选项如下：

- 不重绘：即控件不会进行重绘操作，相当于控件是隐藏着的；
- 边框宽度 1~8 绘制：控件将会以线宽为 1~8 的像素点，并且以所设置的控件重绘描边线条色进行绘制边框；
- 区域填充绘制：在重绘控件时，将会以所设置的控件区域填充色进行绘制控件区域；而填充时可以设置填充的颜色的透明度值，该特性使得控件可以体现出半透明的效果。
- 3D 按钮 阴影宽 1~5：在重绘控件时，将会以所设置的控件区域填充色进行绘制控件区域，并且在区域边沿呈现立体效果

透明度：选择区域填充绘制时可用，该值设置为 10~100，值越小，表示控件进行区域填充重绘时，填充的颜色块透明度越高。

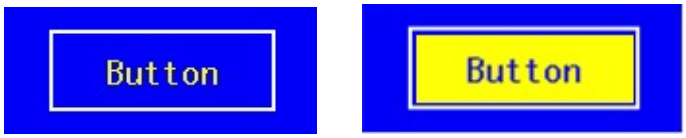
圆角绘制：选择边框绘制或区域填充绘制时可用，勾选后控件将以设置的圆角半径进行圆角矩形绘制控件外形。

区域按钮字符配置：

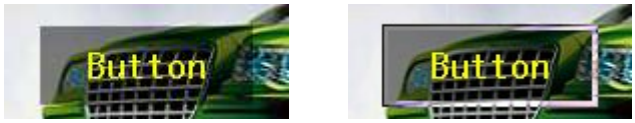
区域按钮作为一个控钮类的控件，允许用户对该控件进行设置字符串，支持使用点阵字库（资源文件中文字库以及西文字库，还有模块内置西文字库）或矢量字库（系统字库），如需要在控件内设置字符

串时，需要设置好字符串所使用的字库，包含对中文字库的选择以及 ASCII 码西文字库的选择；字符串显示的字符色也可设置。

示例：



定义一个区域按钮，重绘属性为边框宽度 2 绘制，边框描边线条色为白色（0xffff），字符串设置模块内部 ASCII 字符，字符色为黄色（0xffe0），字符串为“Button”，显示响应类型为“区域取反响应显示”，外框缩进为 3。上图左侧为控件在蓝色的背景下重绘的效果，右侧为控件按下时显示的效果。



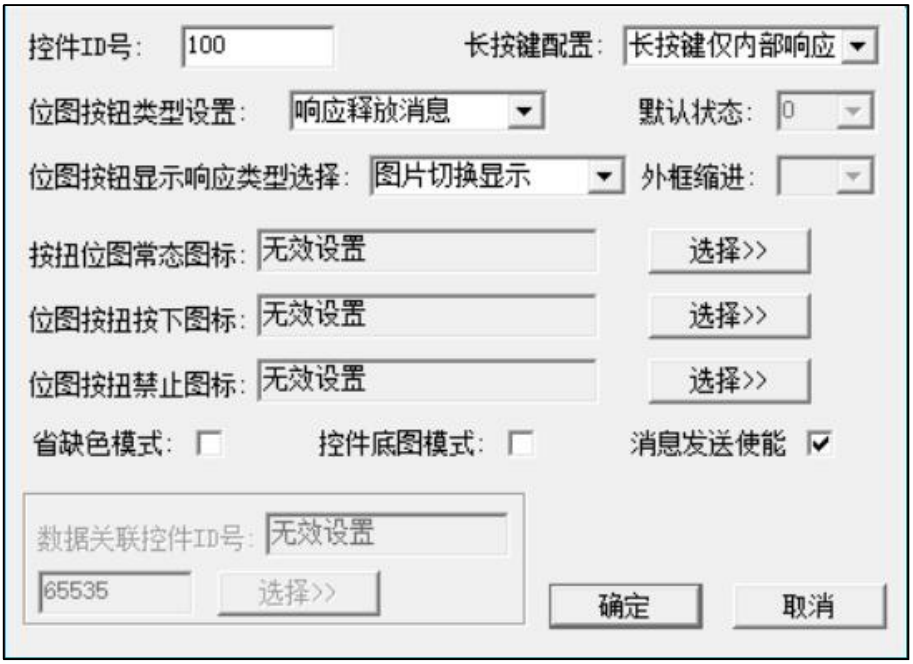
定义一个区域按钮，重绘属性为区域填充绘制，填充色为黑色（0x0000），透明度为 50，字符串设置模块内部 ASCII 字符，字符色为黄色（0xffe0），字符串为“Button”，显示响应类型为“外框宽度 3 取反”，外框缩进为 2。上图左侧为控件在一个图像背景下重绘的效果，右侧为控件按下时显示的效果。

定义一个区域按钮，重绘属性为区域填充绘制，填充色为黑色（0x0000），透明度为 70，圆角半径 8 像素点，使用矢量字库，字符色为黄色（0xffe0），字符串为“Button”，显示响应类型为“区域取反”，外框缩进为 3。上图左侧为控件在一个图像背景下重绘的效果，右侧为控件按下时显示的效果。

3.4.2 位图按钮控件

位图按钮是一个基于位图资源的按钮，其有多种属性可配置，并可实现多种显示、响应效果；该控件支持控件消息配置。

在 EzUITool 工具中，位图按钮控件的配置画面如下图所示：



消息发送使能：该项设置为位图按钮控件所触发消息数据通过串口发送的使能开关，默认为勾选状态，也即控件触发消息时将会通过模块串口发送消息数据包（GUI 服务引擎消息数据包）；取消勾选后，该控件将不会再触发消息数据包的发送。

长按键配置：

该项属性仅对 EzUIH 系列有效，可以在此设置控件的长按键消息触发的机制，备选的选项如下：

- 不支持长按键：表示该控件不支持长按键功能；
- 长按键仅内部响应：表示控件长按键消息仅供模块内部系统驱动控件消息使用，不会连续触发消息数据包的发送；
- 长按键连续消息：表示该控件在长按键时，除了连续触发消息供模块内部系统驱动控件消息之外，在消息发送使能勾选前提下还会连续的发送消息数据包给用户 MCU。

位图按钮类型设置：

该项的属性设置与区域按钮的消息响应机制设置类似，主要设置的是位图按钮控件响应按键消息事件时返回消息的机制，所不同的时位图按钮控件在这里的设置多出两项，该项配置有五项可选，需要注意在以下说明当中只有当消息发送使能勾选前提下控件才会最终触发消息数据包的串口发送：

- 响应按下消息：表示控件只有当按下时，才会通过串口发送消息给用户 MCU；
- 响应释放消息：表示控件只有当释放时，才会通过串口发送消息给用户 MCU；
- 响应按下及释放消息：表示控件会在按下及释放时都会通过串口发送消息给用户 MCU；
- 乒乓开关：表示控件为状态按钮，类似状态切换按钮，可利用该属性实现单选按钮的效果；乒乓开关的属性，控件将会在状态发生改变时通过串口发送消息给用户 MCU。
- 乒乓开关（单选）：表示控件为状态按钮，与乒乓开关类似，不同的仅是该类型设置时，对于用户的按键消息事件响应，控件仅允许从状态 0 切换到状态 1，但不允许从状态 1 切换到状态 0。

默认状态：

当控件类型设置为乒乓开关以及乒乓开关（单选）类型时，该参数可以设置，即设置控件在初始化之后的默认状态。

显示响应类型选择：

该设置为位图按钮控件响应触摸事件时，显示作出相应变化的方式；可选的选项如下：

- 位图切换显示：表示该控件区域在被按下时，将会切换显示位图按钮控件所配置的“按下图标”位图；
- 外框宽度 1~8 取反显示：表示该控件区域被按下时，将会在指定的外框以线宽为 2~8 的值进行矩形框取反显示；
- 区域取反响应显示：表示该控件区域被按下时，将会在指定的外框区域内进行取后显示。
- 截图切换显示：特性与“位图切换显示”设置类型，但表示该控件的常态图标及按下图标将由选择的位图的局部区域作为素材，区域的选择为控件相对显示屏当中的区域所对应设置的位图的局部区域。通常情况下使用与屏幕一样大小的位图素材。

常态图标：

位图按钮控件是基于位图资源的触摸按钮控件，要求至少要指定一个位图资源作为控件的常态图标，该位图资源可从加载到资源文件中的位图资源选择；控件在重绘时，将显示该位图显示。

按下图标：

位图按钮控件的按下图标设置可有可无，但如显示响应类型选择为“位图切换显示”或位图按钮控件类型选择为“乒乓开关”时，该项配置必需有，且要求按下图标的位图尺寸与常态图标一样。

禁止图标：

位图按钮控件禁止图标设置为可选设置，如配置该项图标资源时，若控件处于禁能状态，则会控件将显示该项配置所对应的图像资源。

乒乓开关：

当位图按钮控件设置为乒乓开关时，需指定控件的常态以及按下图标位图资源，此时，控件相当于一个 0 和 1 状态的开关，当状态为 0 时显示常态图标，状态为 1 时显示按下图标，而控件的状态将会在控件接受有效按键消息事件时发生切换，比如控件重绘完成后，状态为 0，此时如将控件设置为选择状态，然后用户 MCU 发送“Entry”按键消息，然后控件的状态就会切换到 1，此时显示按下图标。用户也可通过指令来给控件进行状态设置，控件将会根据实际情况进行更新显示，也可通过指令读取控件的状态。

数据关联控件 ID 号：

当位图按钮控件属性设置为乒乓开关（包括单选）时，位图按钮控件将具备数值（状态），此时允许设置该控件的数据关联控件 ID 号。

省缺色模式：

当控件配置为图片切换的属性时，如选择配置的图片资源项为位图图像，则可勾选此项；使用省缺色模式显示的位图按钮控件，模块将会获取图像的四个角的四点像素颜色值，如四点颜色相同，则会认定该颜色为省缺色，而后在显示位图时将与省缺色相同的素材点不进行显示；所以用制作位图素材时，应注意以下：

位图图像除有效区域之外的地方，应使用有别于有效区域颜色的背景色，这样在位图显示时，将会自动把有效区域之外的区域清除不显示。

控件底图模式：

位图按钮控件支持控件底图模式的选择，当勾选此项时，该控件的显示图像将可视为控件区域的背景图层内容，简单说，勾选此项后，该位图按钮控件之上可以叠加显示别的控件。

示例：

设置一个位图按钮控件，控件类型为非“乒乓开关”，显示响应类型为“图片切换显示”，而常态图标使用下图左则图标，按下图标使用下图右则图标：

控件在重绘后，将显示上图左侧的效果，当控件被按下时，将显示上图右侧效果，控件释放后，又将恢复为上图左侧效果。其间，将会根据所设置的消息返回机制，将控件按下或释放的消息返回给用户 MCU。

设置一个位图按钮控件，控件类型为“乒乓开关”，显示响应类型只能选择“图片切换显示”之外的选项，在此，选择外框宽度 2 取反显示，外框缩进为 0，而常态图标使用下图左则图标，按下图标使用下图右则图标：

则当控件重绘后，假设其状态为 0，则显示上图左侧效果，然后，此时如将控件设置为选择状态，然后用户 MCU 发送“Entry”按键消息，则在控件释放之前，控件图标外框将会以宽度为 2 的边框取反显示，当控件释放后，控件状态切换为 1，此时显示上图右侧效果。

截图切换显示示例，在此使用两张位图资源（一般要求与屏幕大小一致，也即像素点数相同），如下图所示：

首先在显示界面当中，将左侧的位图显示在界面上，然后在界面当中，选择一个按钮区域设置位图按钮控件，如下图所示：

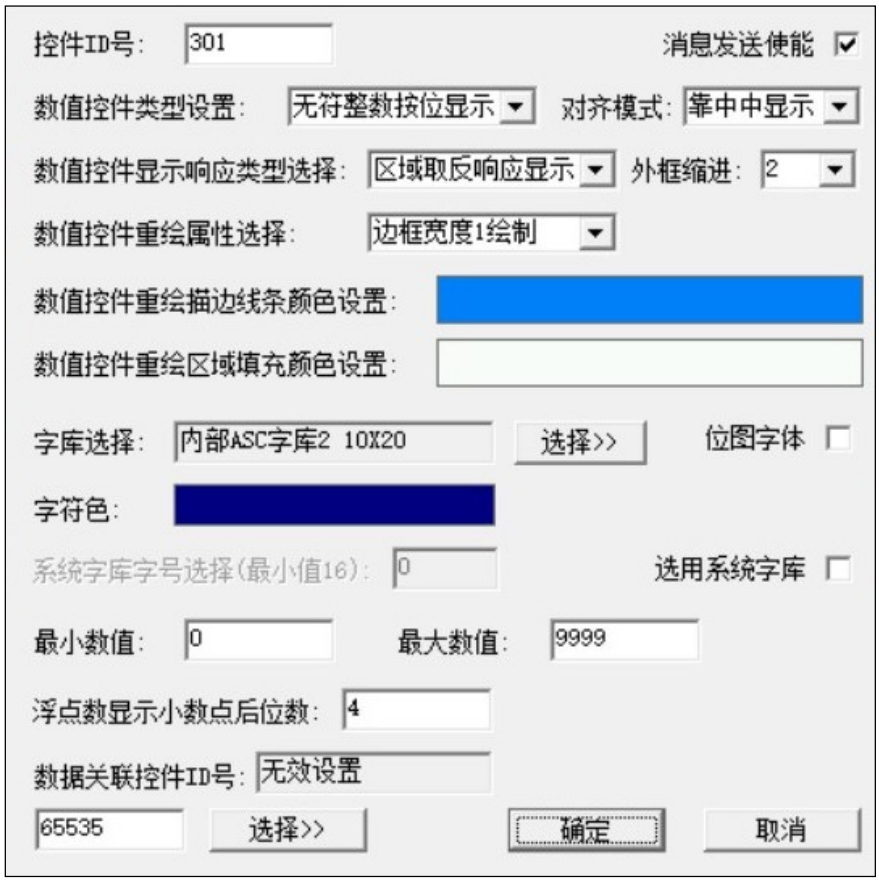
图中虚框区域也为要设置的位图按钮控件的区域，在弹出的对话框中配置控件的属性，选择控件类型为非“乒乓开关”，显示响应类型为“截图切换显示”，而常态图标使用前面图中左则图标，按下图标使用前面图中图右则图标。这样，实际上控件将会在刷新显示时，以控件所设置的区域从常态、按下图标的位图中提取出局部的图像进行显示。

下图左侧为控件常态下显示的状态，右侧为按下时显示的状态：

3.4.3 数值控件

数值控件（Number_Ctrl）是一个可输入数值也可显示用户设置数值的控件，其有多种属性可配置，并可实现多种显示、响应效果；该控件不支持控件消息配置，但允许其它控件对它进行控制。

在 EzUITool 工具中，数值控件的配置画面如下图所示：



数值控件类型设置：可将数值控件设置为无符整型数、有符整型数、浮点数、无符整型数按位显示。

对齐模式：即数值在所定义的控件区域内显示的对齐方式，可设置垂直以及水平的对齐方式，以两个方向的组合进行选择，弹出的下拉选框中可以选择到合适的对齐方式。

字库选择：数值控件在显示数值时，以 ASCII 西文字符进行显示，用户需指定控件所使用的字库，以及字符色；可选择点阵字库（模块内置西文字库或加载入资源文件的西文字库）或矢量字库（系统字库）。

位图字体：EzUIH 模块支持使用位图替代数字字符显示，当勾选此项时，则可通过按钮选择配置已加载入资源文件的图片作为字符显示的素材；作为位图字体素材的图片应在加载时遵循以下规则：

- 图片资源应按顺序从 0~9 数字图片、小数点图片、负号图片在资源文件中加载；
- 负号图片在控件选择为无符数时可选择不加载入资源文件；
- 小数点图片在控件选择为整型数时可选择不加载入资源文件；
- 如负号图片选择加载，则小数点图片必需加载入资源文件。

最小数值、最大数值：数值控件可以设置该控件允许输入的最大最小值，当输入的数值超出所设置的范围时，输入将不会生效。

浮点数显示小数点后位数：当数值控件的数值类型为浮点数时，可以设置显示小数点后的有效位数；此外当数值控件类型为“无符整型数按位显示”时，该设置可以将数值显示时按照设置的位数进行显示。

数据关联控件 ID 号：

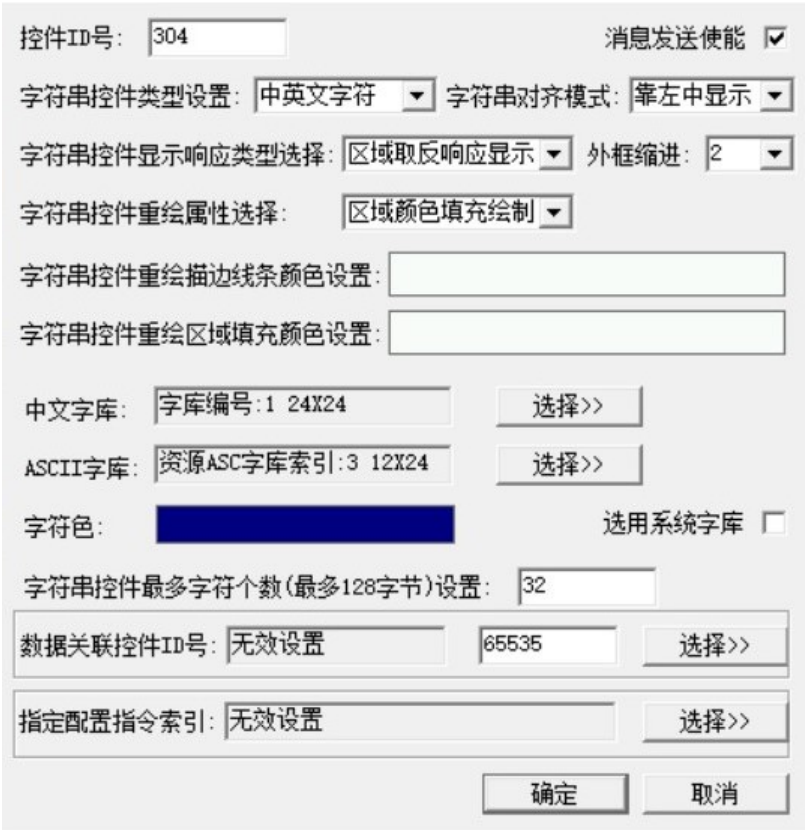
数值控件允许设置该控件的数据关联控件 ID 号。

消息发送使能：该项设置为数值控件所触发消息数据通过串口发送的使能开关，默认为勾选状态，也即控件触发消息时将会通过模块串口发送消息数据包（GUI 服务引擎消息数据包）；取消勾选后，该控件将不会再触发消息数据包的发送。

3.4.4 字符串控件

字符串控件（String_Ctrl）是一个可显示用户设置字符串的控件，其有多种属性可配置，并可实现多种显示效果；该控件不支持控件消息配置，但允许其它控件对它进行控制。

在 EzUITool 工具中，字符串控件的配置画面如下图所示：



字符串控件类型设置：可将字符串控件设置为数字符号、西文字符以及中英文字符，实际上相当于配置该控件在输入字符串时，允许输入的字符类型。

最多字符个数设置：用户可以设置该数值，限制输入时的字符个数（字节数）。

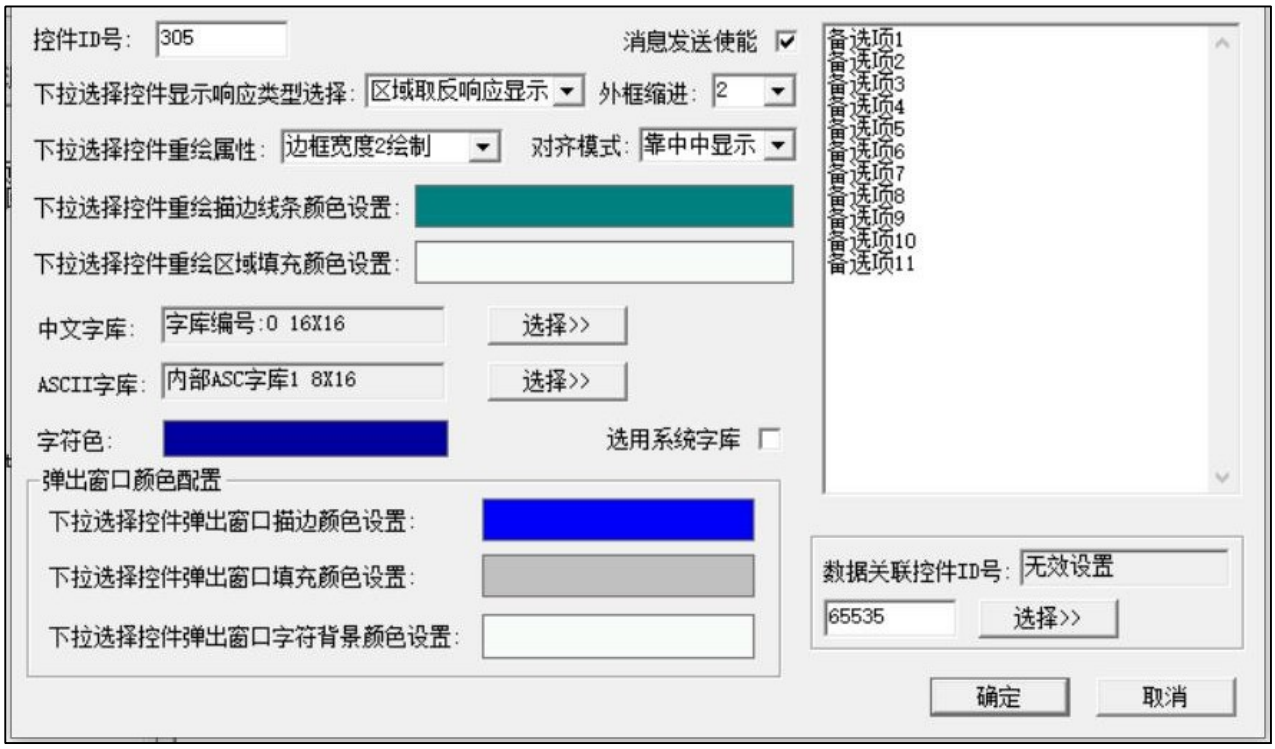
字符串控件与数值控件类似，在此不多作叙述。

3.4.5 下拉选择控件

下拉选择控件（Combox_Ctrl）为一个预置多个选项，可响应用户发送的按键消息选择预选项的控件，最多可预置 50 条预选项，每个预选项可用最长 25 个字节的字符串进行描述。

下拉选择控件有多种属性可配置，并可实现多种显示、件响应效果；该控件支持控件消息配置，也允许其它控件对它进行消息控制。

下拉选择控件在 EzUITool 工具中的配置对话框如下图所示：



下拉选择控件的大部分属性设置与前面所述的几类控件相似，在此仅对不同之处进行说明。

显示响应类型选择当中，有一个选项是“不响应（不允许输入）”，当控件选择该属性时，该控件不允许按键消息响应，即无法通过用户 MCU 发送按键消息指令对控件的选项选择；此时该控件只会显示出当前控件被选择项，被选择项可由用户用过相应的指令进行设置（控件数值设置指令）。

弹出窗口颜色配置：

无触摸版模块该处配置无意义。

设置对话框的右侧，有一个大的文本输入框，在此可以输入预置选项的文字描述，多个选项之间在输入时在计算机上“Ctrl+Enter”键进行换行，每行表示一个预选项的文字描述，每行最多支持 25 个字节的字符串。

示例：

设置一个下拉选择控件，显示响应选择为“区域取反响应显示”，外框缩进为 2，重绘属性为“区域颜色填充绘制”，文本对齐模式为“靠中中显示”，控件重绘填充色为白色，使用模块内部序号 1 的 ASCII 字库（0x8001），字符色为深蓝色；弹出窗口描边色为黑色，窗口填充色为浅灰色，窗口字符背景色为白色；而控件预置选项的字符串如下图所示：

共 7 项待选项，此时，控件在蓝色的背景上重绘后，假设当前被选项为第 0 项，则显示如下图左侧

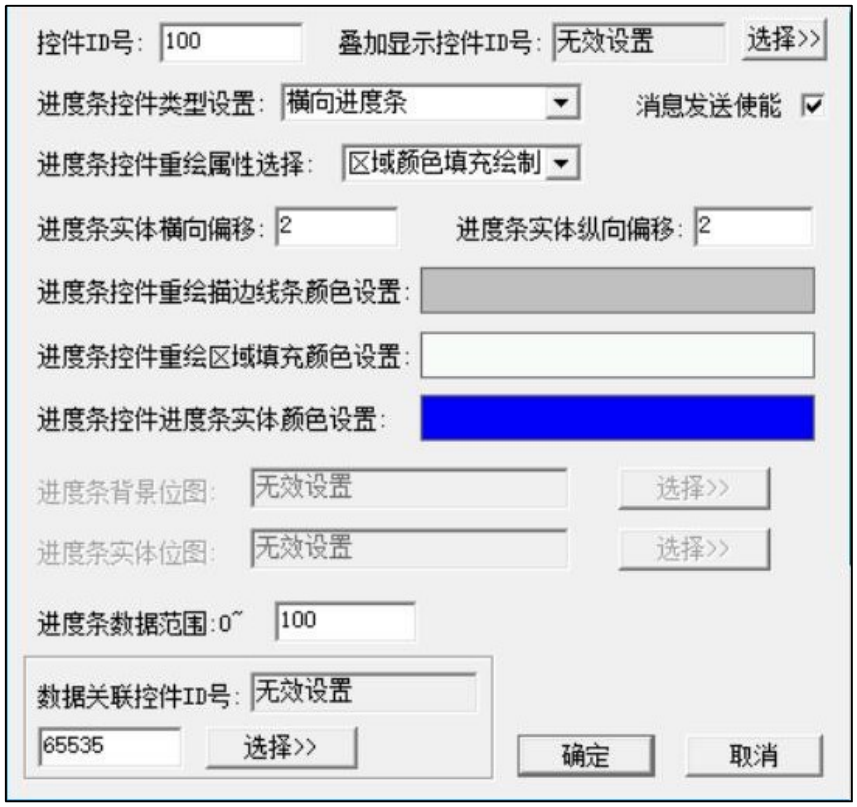
所示：

当控件的当前被选项发生改变时，如消息发送使能勾选，模块会返回消息给用户 MCU，通知用户，当前控件的被选项发生改变。

3.4.6 进度条控件

进度条控件（Process_Ctrl）为仅显示的控件，用进度条的形式将用户数据呈现。

进度条控件在 EzUITool 工具中的配置对话框如下图所示：



类型设置：可选择进度条为横向进度条或纵向进度条；横向进度条控件将从左向右进行增长，纵向进度条控件将从底至上进行增长。

重绘属性选择：

进度条控件可选择多种重绘属性，可选项如下：

- 边框线宽 1~8 绘制：控件将会以线宽为 1~8 的像素点，并且以所设置的控件重绘描边线条色进行绘制边框；
- 区域填充绘制：在重绘控件时，将会以所设置的控件区域填充色进行绘制控件区域。
- 背景图层重绘：在重绘控件时，将不对背景图层的显示进行叠加显示，并且在进度条更新显示进度条位置时，也都将在背景图层的显示内容上进行绘制进度条。
- 位图局部截图重绘：选择该属性时，用户将只需要设置“进度条背景位图”和“进条度实体位

图”，控件将会使用所选用的位图的局部区域作为控件的背景以及进度条实体的绘制素材，局部的区域由控件相对屏幕的区域决定；一般选择使用与屏幕大小一致的位图。

- 位图重绘：选择该属性时，用户将只需要设置“进度条背景位图”和“进条度实体位图”，控件将会使用所选择的位图资源项进行进度条的绘制，要求用以绘制进度条控件的位图资源项大小尺寸相同。

进度条实体横向偏移：该项设置将决定进度条相对于所设置的控件区域的横向偏移缩进量，以像素为单位。

进度条实体纵向偏移：该项设置将决定进度条相对于所设置的控件区域的纵向偏移缩进量，以像素为单位。

重绘描边线条颜色设置：该项设置可配置进度条控件的边框线条颜色。

重绘区域填充颜色设置：该项设置可配置进度条控件的整体背景填充色。

进度条实体填充颜色设置：该项设置将决定进度条实体的颜色。

进度条数据范围：该项设置将决定进度条实体在显示满值时的最大数值，控件将会以该数值作为参考，显示当前进度条实体的位置，当控件当前数值超出该设置时，进度条控件将显示满值。

叠加显示控件 ID 号：

进度条控件允许设置一个叠加在其之上显示的控件，允许设置的控件类型为数值控件或字符串控件，当设置叠加显示后，叠加在进度条控件之上显示的控件将只有字库配置、字符色、字符对齐的配置是有效的；要求叠加的数值控件或字符串控件的控件区域完全在进度条控件的区域之内。

叠加显示的数值控件或字符串控件将不可以进行输入，且当被叠加的进度条控件被控制为隐藏时，所叠加的控件也将一起隐藏。

示例：

设置一个进度条控件，横向进度条类型，重绘属性为区域填充重绘，重绘区域填充色为白色，进度条实体颜色为蓝色，进度条实体横向、纵向偏移均为 2，进度条数据范围为 100，如当前进度条控件的数值为 60 时，显示在蓝色的背景之上的进度条控件如下图效果：

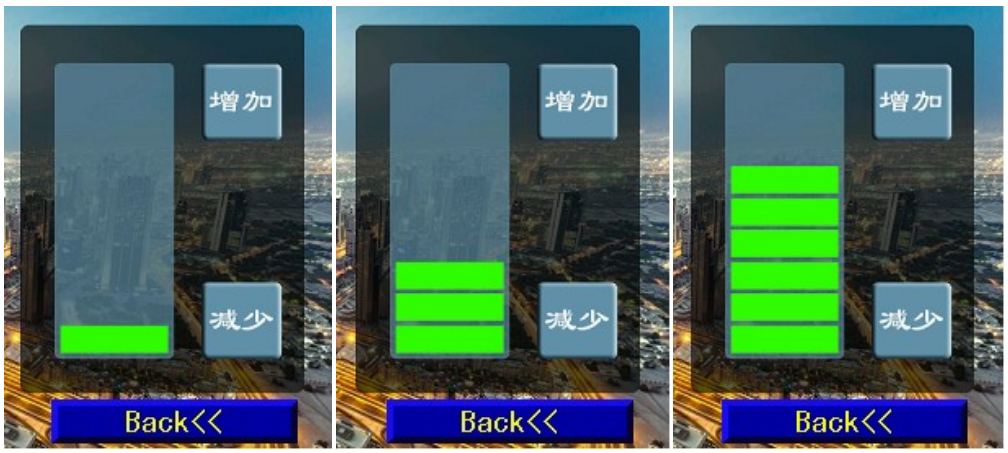
设置一个进度条控件，横向进度条类型，重绘属性为背景图层重绘，进度条实体为绿色，进度条实体横向、纵向偏移均为 1，进度条数据范围为 100，如当前进度条控件的数值为 60 时，显示在位图的背景之上的进度条控件如下图效果：

位图局部截图重绘属性的示例，使用两张位图资源（一般要求与屏幕大小一样的图片作为素材），如下两张图片：

首先在显示界面当中，将左侧的位图显示在界面上，然后在界面当中，选择一个区域设置进度条控件，如下图所示：

图中虚框区域也为要设置的进度条控件的区域，在弹出的对话框中配置控件的属性，选择控件类型为非“纵向进度条”，重绘属性为“位图局部截图重绘”，而进度条背景位图使用前面图中左则图标，进度条实体位图使用前面图中图右则图标。这样，实际上控件将会在刷新显示时，以控件所设置的区域从所设置的位图中提取出局部的图像进行显示。设置数据范围为 0~9。

下图自左到右的三个图片分别是进度条数值为 1、3、6 时的显示状态：



3.4.7 波形控件

波形控件（WaveForm_Ctrl）为仅显示的控件，用波形线的形式将用户数据呈现。同一个波形控件当中，允许最多四条波形线在其中进行更新、绘制；各条波形线均可单独设置其波形线颜色。

波形控件在 EzUITool 工具中的配置对话框如下图所示：

控件类型设置：

波形控件允许波形线的移动方向的设置，有两项可供选择：

- 波形线自左向右移动：波形更新显示时，新的数据将从控件的左侧加入，如波形已显示满控件，则将波形线向右移动；
- 波形线自右向左移动：波形更新显示时，新的数据将从控件的右侧加入，如波形已显示满控件，则将波形线向左移动；
- 柱形图显示：数据以柱形图的形式进行显示，控件的更新与波形线类似。

重绘类型设置：

该项设置有两项可选，分别是：

- 即时重绘：表示波形控件在接收到用户新加入的数据后，会即时自动刷新波形显示；
- 用户指令控制重绘：表示波形控件在接收到用户新加入的数据后，不会自动刷新波形显示，而是将数据压入缓冲区，待用户发送波形重绘指令后，波形控件才会进行波形的刷新显示。

栅格线间隔：

波形控件支持波形控件的栅格线显示，纵向与横向栅格线的设置相类似，当其值小于等于 5 时，将不显示相应的栅格线；该值的设置以像素为单位。

栅格线颜色设置：

该设置将决定波形控件的栅格线的颜色，而控件的栅格线在绘制时，线宽都为 1 像素。

波形控件波形线相关设置：

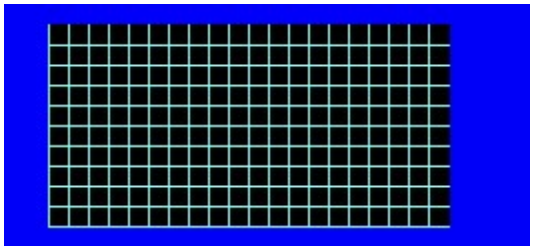
最多允许 4 条波形线，并可分别设置各波形线的颜色，系统默认波形线将会以线宽为 2 的线条进行绘制波形曲线。

波形控件的 X 轴以及 Y 轴都是以像素为单位，也即 X 轴和 Y 轴的范围，都对应实际设置控件的区域像素。

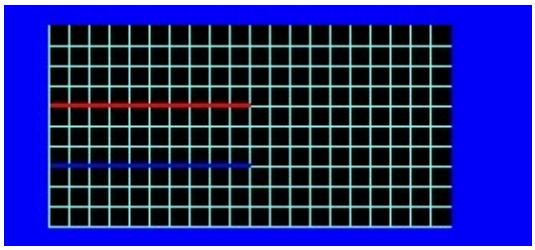
波形线 X 轴步进素像设置：波形控件默认 X 轴的步进值为 1 像素点，比如设置一个在屏上 X 轴起始坐标为 10，X 轴终点（右侧）坐标为 109 的控件，则波形控件的 X 轴范围为 100 个像素点，当步进值为置为 1 时，表示控件当中波形线在显示区域当中一共有 100 个数据点；而当设置步进值为 2 时，则控件当中波形线在显示区域当中一共有 50 个数据点。

示例：

设置一个波形控件，属性为波形线自左向右移动，X 轴高度为 100 像素点，Y 轴宽度为 200 像素点，控件背景色为黑色，栅格线间隔 X 轴及 Y 轴均为 10，栅格线颜色为青色，同时允许 2 条波形线显示，波形线 1 颜色为红色，波形线 2 颜色为蓝色；控件在蓝色的背景下重绘后，如波形控件当前暂无数据，则显示效果如下图所示：



若对波形控件进行数据加载，波形线 1 的数据加载 100 个数值，数值为 60，波形线 2 的数据加载 100 个数值，数值为 30，则波形控件重绘后，显示如下图所示：



柱形图显示演示：

当波形控件选择柱形图显示属性时，控件配置对话框将多出一些设置项，如下图所示的配置：

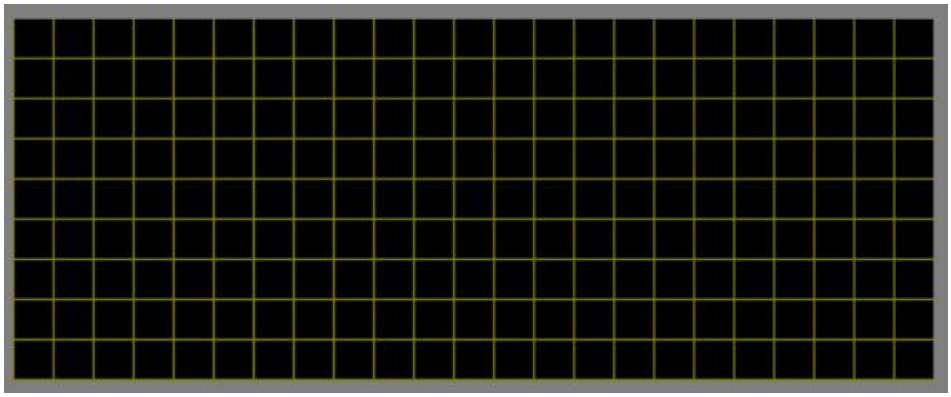
多柱形间隔设置：当控件选择的波形线数量大于 1 时，该设置将确定同一组数据的不同柱形图案的间隔像素点；在此设置为 2。

重绘方式：可选背景颜色重绘或背景图层重绘，选择背景颜色重绘时，将以前面设置的控件重绘背景颜色对控件所在区域进行背景的绘制；当选择背景图层重绘时，则前面设置的控件重绘背景颜色将无意义。

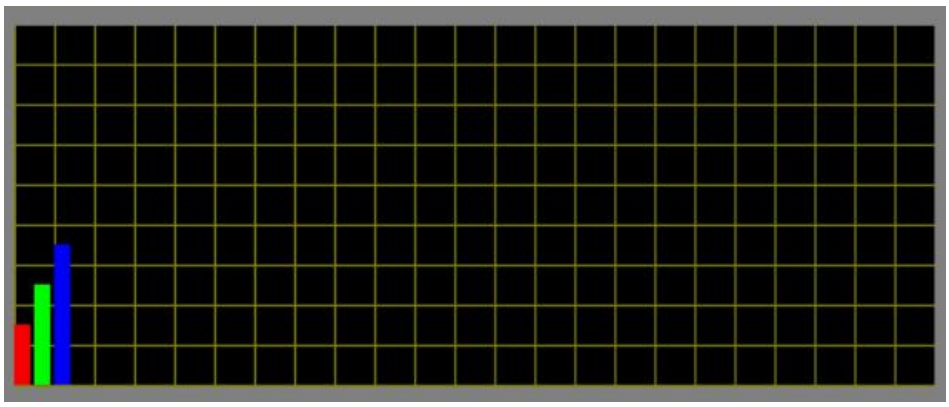
柱形图宽度：该项设置将指明柱形图显示时单条颜色柱的 X 轴宽度，在此设置为 8 像素点。

柱形 X 轴步进间隔：该设置指定两组数据的柱形图之间的间隔（两组数据同一个柱形左侧与左侧的横向坐标像素点数）；在此设置为 40 像素点。

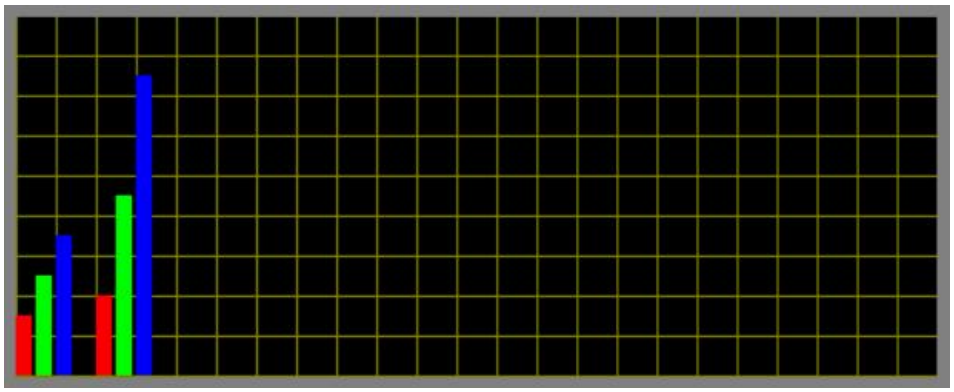
如上图配置后，当控件尚无数据设置时，显示如下图。



注意，控件中栅格线 X 轴间隔为 20 像素点。当给控件输入第一组数据（一组数据三个数值，分别为 30、50、70）时，如下图所示：



一组 3 个柱形图案，每个柱形图宽 8 像素点，柱形图之间间隔 2 像素点。当继续输入第二组数据（40，90，150），控件显示如下：

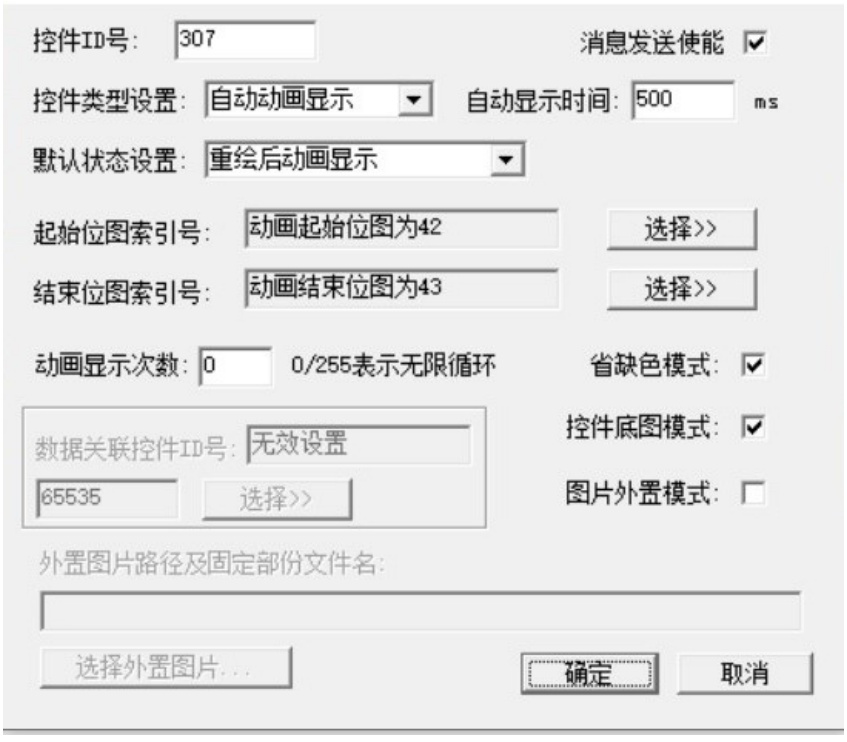


可见红色的柱形图按，两组柱形的左侧位置间隔与设置的 40 像素点一致（X 轴栅格线间隔为 20）。

3.4.8 位图动画控件

位图动画控件（Gif_Ctrl）是一个基于位图资源的动态图像显示控件，其有多种属性可配置，并可实现动画显示效果或位图状态显示效果；该控件不支持用户按键消息事件，但允许接受控件消息控制。

在 EzUITool 工具中，位图动画控件的配置画面如下图所示：



控件类型设置：

位图动画控件支持 3 种类型的设置，分别如下：

- 自动动画显示：将控件设置为该类型时，控件的状态有两种，分别是停止状态和动态显示状态；当停止状态时，控件将显示所设置的起始位图索引所指向的第一张位图；而当控件为动态显示状态时，控件将按所设置的“自动显示时间”作为位图切换的时间间隔，将“起始位图索引号”至“结束位图索引号”之间的位图依次切换显示；

- 静动态显示：将控件设置为该类型时，控件也是有两种状态，分别是静止状态和动态显示状态；当控件处于静止状态时，控件将显示所设置的“起始位图索引”把指向的第一张位图；而当控件为动态显示状态时，控件将按所设置的“自动显示时间”作为位图切换的时间间隔，将“起始位图索引号”之后的一张位图至“结束位图索引号”之间的位图依次切换显示；
- 完全指令切换：将控件设置为该类型时，控件的状态将指示控件显示位图的序号，序号的大小由“起始位图索引”至“结束位图索引”之间的位图个数决定；用户可通过指令设置控件的状态，以此切换控件所显示的位图。

自动显示时间：

该参数的设置仅在位图动画控件设置为“自动动画显示”和“静动态显示”类型时有效，表示控件在动态切换位图显示时的时间间隔，单位为毫秒（ms），允许设置的最小值为 200ms，最大值为 2000ms。

默认状态设置：

该参数设置的是控件在系统初始化之后的默认状态，其值的设置与控件的类型设置有关。

- 自动动画显示：控件将允许设置两个默认状态中的一个，分别是：重绘后动画显示以及重绘后停止显示，即表示控件在初始化之后的默认状态；当然用户也可通过指令控制控件的当前状态，即控件的动态/停止状态；
- 静动态显示：控件将允许设置两个默认状态中的一个，分别是：重绘后显示静态以及重绘后显示动态，即表示控件在初始化之后的默认状态；当然用户也可通过指令控制控件的当前状态，即控件的动态/静态状态；
- 完全指令切换：控件的默认状态将指示控件在初始化后默认显示位图的序号，序号的大小由“起始位图索引”至“结束位图索引”之间的位图个数决定；用户可通过指令设置控件的状态，以此切换控件所显示的位图。

起始位图索引(注)：

该设置将决定位图动画控件所配置的位图在资源文件中的起始索引号，要求该起始索引要小于“结束位图索引”。

结束位图索引(注)：

该设置将决定位图动画控件所配置的位图在资源文件中的结束索引号，要求该起始索引要大于“起始位图索引”。起始位图索引和结束位图索引之间的位图个数即为该控件所配置的位图个数，或称为帧数。

注：起始位图索引及结束位图索引的设置在“图片外置模式”处于取消勾选下可用。

动画显示次数：

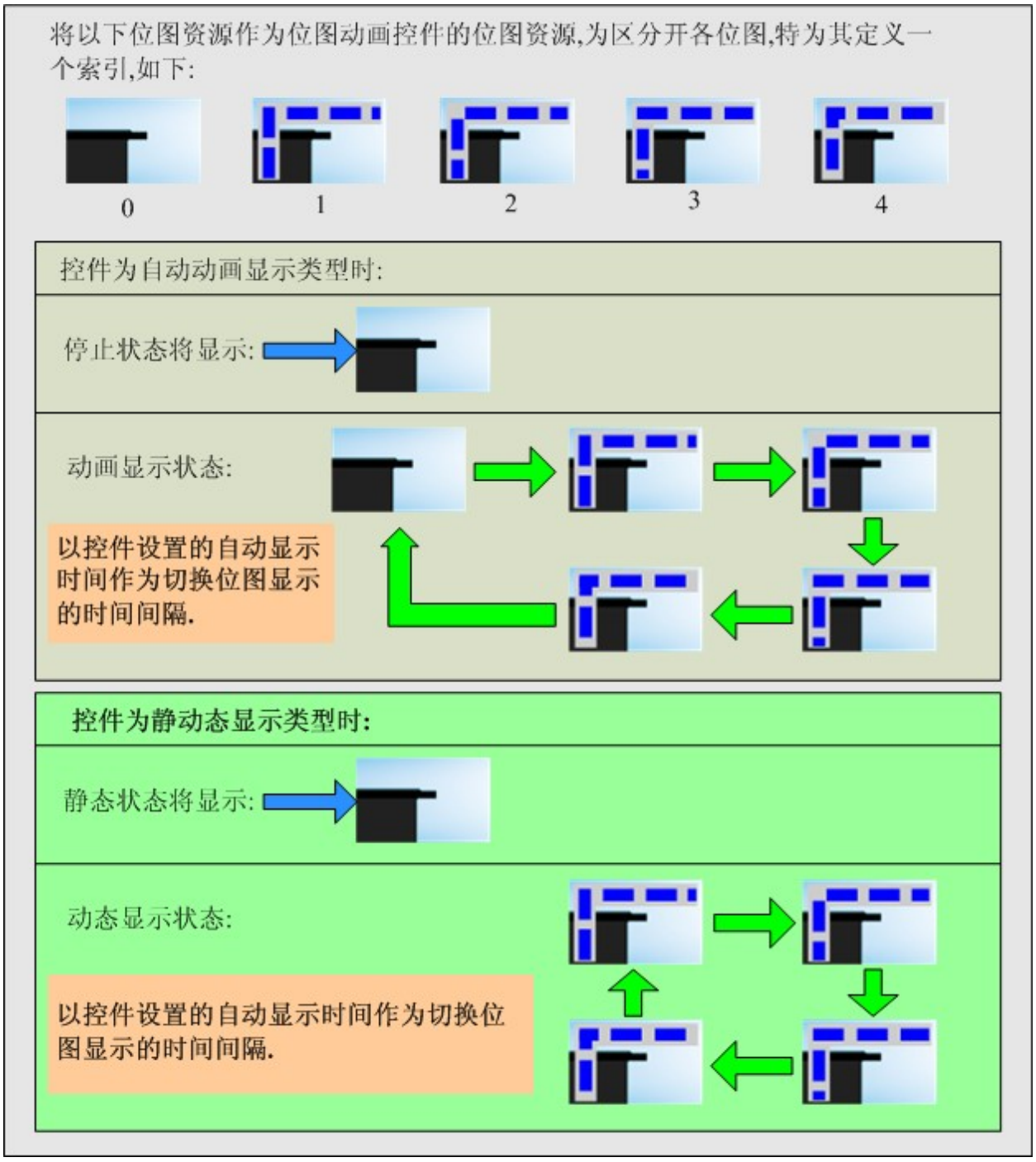
该参数的设置仅在位图动画控件设置为“自动动画显示”类型时有效，该参数设置将确定动画显示的次数，当其值为 0 或 255 时，表示无限次循环显示；而当设置为其它值时，表示动画动态显示的次数，当显示循环显示的次数达到所设置值时，位图动画控件将会在 GUI 服务引擎内部发出控件消息，可以用于驱动界面切换等操作（该项功能一般用于开机动画显示），若此时控件消息发送使能勾选，则也将触发消息数据包的串口发送。

数据关联控件 ID 号：

当位图动画控件属性设置为“自动动画”之外的选项时，允许设置该控件的数据关联控件 ID 号。

示列：

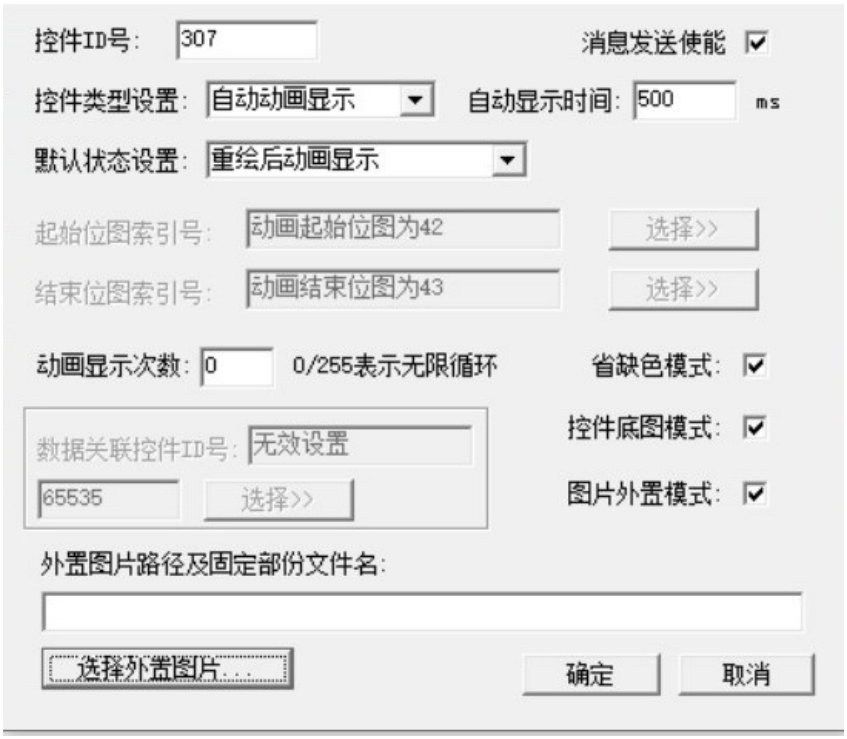
下图将说明位图动画控件的各类型设置的含意：



图片外置模式

EzUIH 模块支持位图动画控件所配置的图片预存于模块 U 盘当中，无需加载入资源文件；当用户需要使用大量图片制作数量较多的动画控件时，就不必受资源文件所支持加载的最大资源项数限制。

当用户勾选“图片外置模式”时，配置对话框如下图所示：



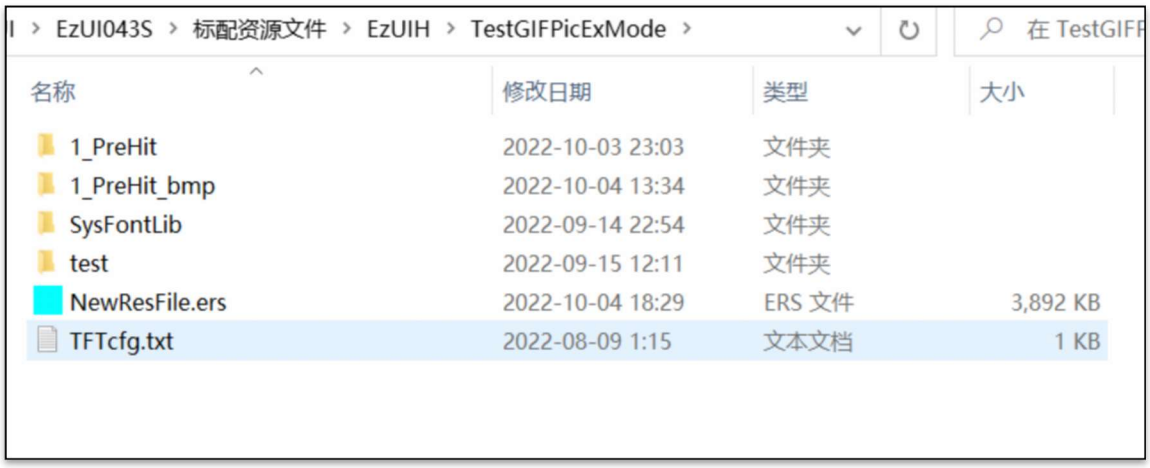
此时“起始位图索引号”及“结束位图索引号”处的配置将呈灰色不可用状态；用户需指明配置给动画控件的图片存储相对路径及配置图片的固定部份文件名。

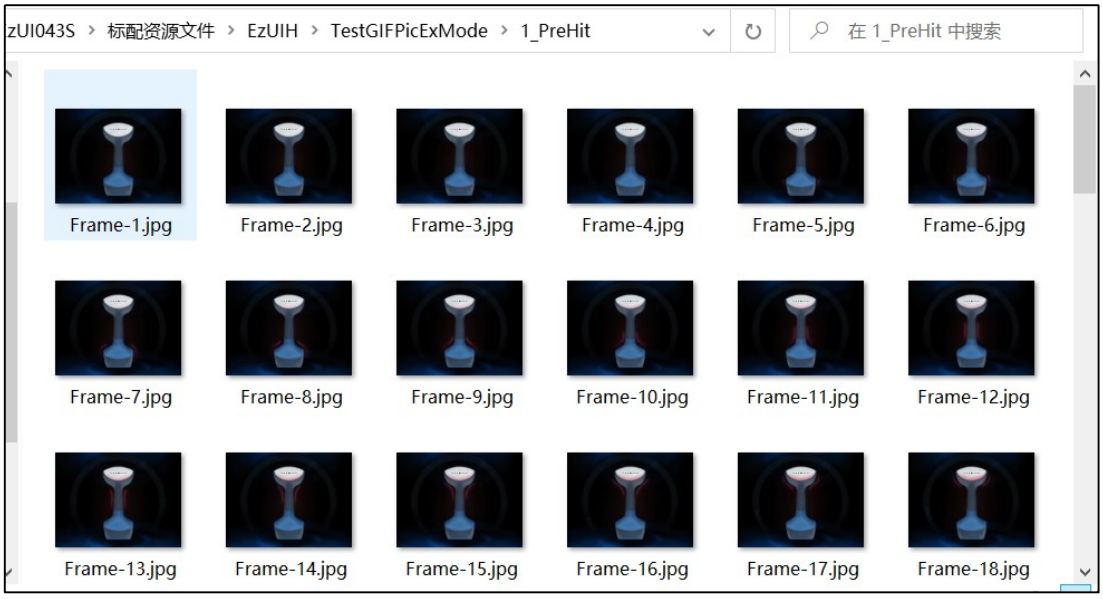
图片外置模式下给动画控件配置的图片可为 BMP 图片文件或 JPEG 图片文件，文件名需统一格式，且文件名最后需以数字作为文件名的一部份，连续多帧图片素材文件的文件名的数字部份需为连续数字；如，假设所配置的文件名固定部份为“DemoPic”，则对应于“起始图片索引号”的图片名需为“DemoPic1”，第二张图片为“DemoPic2”，第 10 张图片名为：“DemoPic10”。

图片外置模式时，动画控件允许一个控件最多配置 252 张图片作为控件的图片帧。

示例：

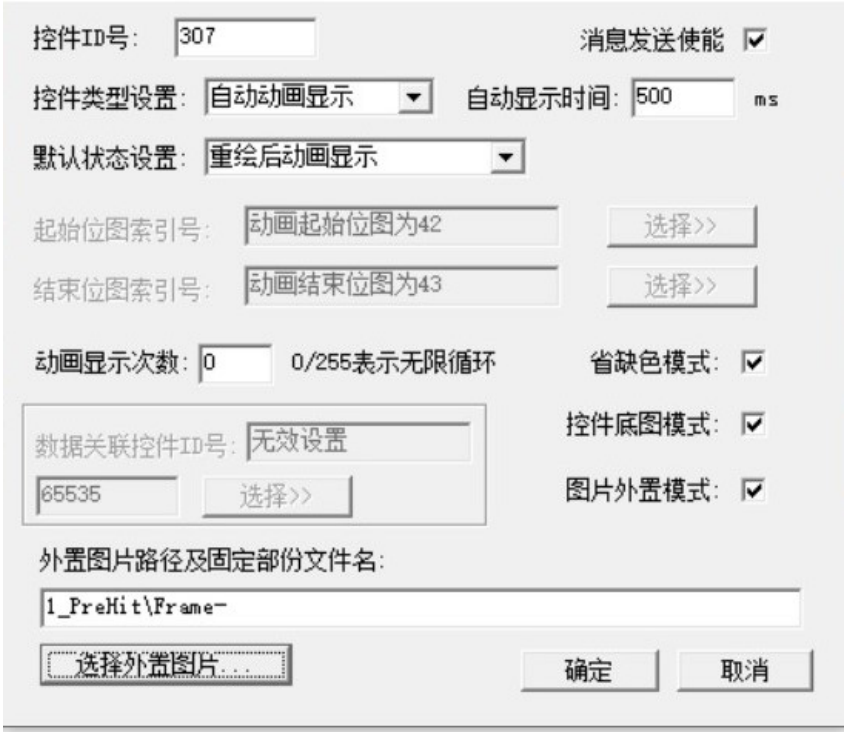
如当前模块 U 盘中，根目录下预存有多个文件夹，且文件夹中有多个符合要求的图片文件，如下图所示：





注意：示例图片中，预存外置图片的文件夹与当前编辑的资源文件（后缀.ers）为同一目录下，在编辑时最好将它们置在相似的位置，以便于工具软件进行效果呈现以及观察；当资源文件完成编辑并且要在模块上进行显示时，须将预置图片的整个文件件拷入模块 U 盘中，且保持与设置时相同的相对路径关系。

此时，点击“选择外置图片”工具按钮，在弹出的对话框中完成外置图片的选择（可选择图片帧中任意一帧），完成后如下图所示：

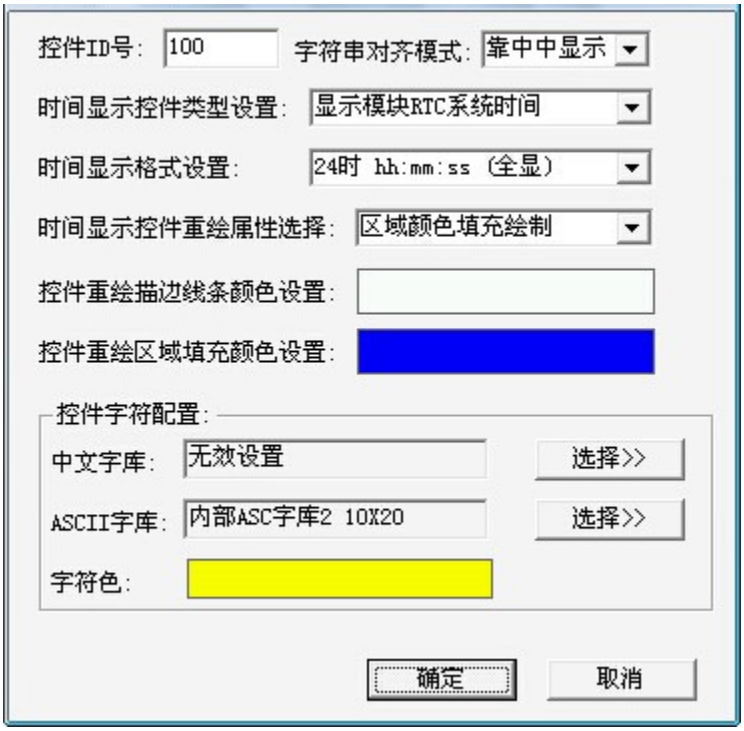


3.4.9 时间显示控件

时间显示控件（TimeDisCtrl）是一个以常见的时间显示方式显示模块内置 RTC 时钟或用户指定时间信息的控件，其有多种属性可配置，并可实现多种显示效果。

时间显示控件如要显示模块内置的 RTC 实时时钟的时间，需要模块具备内置 RTC 实时时钟功能，用户在使用时应确认所使用的模块是否具有 RTC 实时时钟功能。

在 EzUITool 工具中，时间显示控件的配置画面如下图所示：



时间显示控件的部分属性设置与字符串显示控件类似，在此不多作详述，仅介绍其独有的特性设置。

控件类型设置：可将控件设置为“显示模块 RTC 系统时间”或“显示用户指定时间”。

- 显示模块 RTC 系统时间：该类型设置表示控件从 EzUI 模块的内置 RTC 实时时钟获取当前时间进行显示；此时不允许用户通过控件数值写入指令对该控件进行更新写入操作；
- 显示用户指定时间：该类型设置表示控件显示的是由用户通过指令对控件写入的时间信息。
- 用于设置模块 RTC 系统时间：该类型设置表示控件可接受控件消息进行数据修改，并可将修改后的时间更新模块内置 RTC 实时时钟（需要控件消息进行触发）。

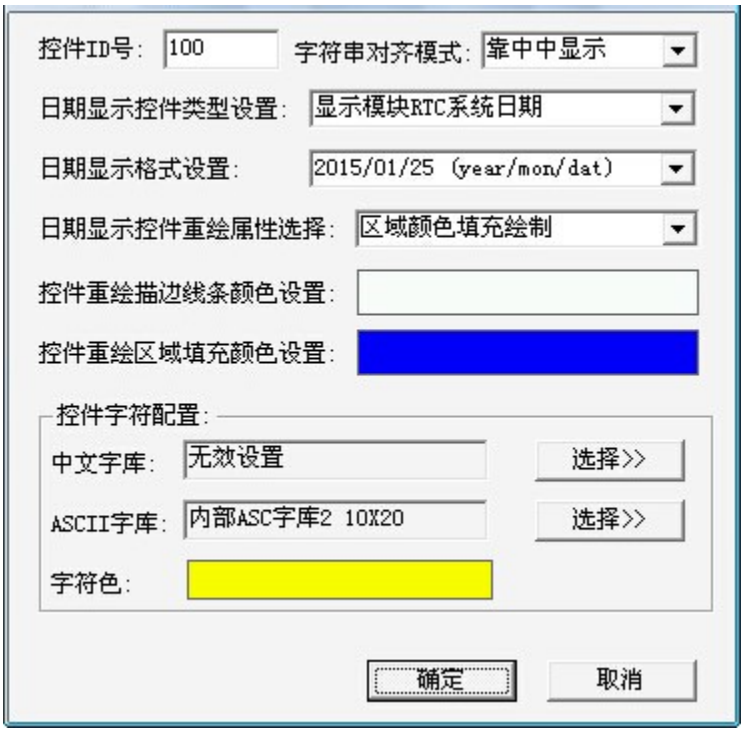
时间显示格式：该设置将决定控件在显示时间时所呈现的格式，包括 24 小时或 12 小时制显示等多种显示风格。

3.4.10 日期显示控件

日期显示控件（DateDisCtrl）是一个以常见的日期显示方式显示模块内置 RTC 日期或用户指定日期信息的控件，其有多种属性可配置，并可实现多种显示效果；该控件不支持控件消息配置，也不允许其它控件对它进行控制。

日期显示控件如要显示模块内置的 RTC 实时时钟的日期，需要模块具备内置 RTC 实时时钟功能，用户在使用时应确认所使用的模块是否具有 RTC 实时时钟功能。

在 EzUITool 工具中，日期显示控件的配置画面如下图所示：

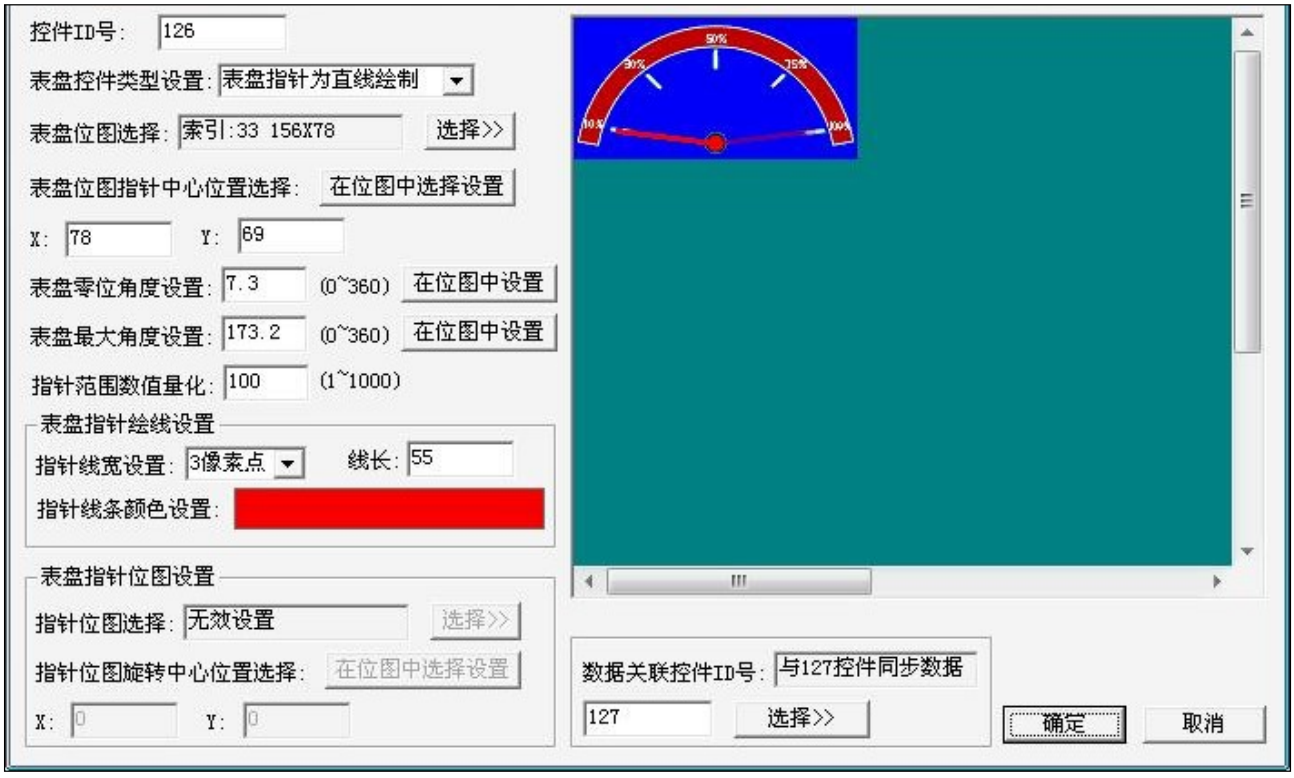


日期显示控件的属性设置与时间显示控件类似，在此不多作详述。

3.4.11 表盘显示控件

表盘显示控件是一个以表盘背景图为基础且以指针沿中心点顺时针旋转指示数值的控件，表盘背景图为位图资源，表盘指针可以设置为直线绘制或图片旋转，并可实现多种显示效果；该控件不支持控件消息配置，但允许其它控件对它进行控制。

在 EzUITool 工具中，表盘显示控件的配置画面如下图所示：



表盘控件类型设置：可将控件设置为表盘指针为直线绘制或表盘指针为位图资源；

- 表盘指针为直线绘制：该类型表示表盘显示控件在显示时，表盘的指针由用户配置的线宽、线长及绘图色决定的直线来进行绘制，上图中示例的即为表盘指针为直线绘制类型。
- 表盘指针为位图资源：该类型表示表盘显示控件在显示时，表盘的指针由用户指定的位图资源进行显示。

表盘位图选择：点击右侧按钮，可弹出位图资源选择对话框，用户可从里面选择已加载到资源文件的位图资源作为表盘的背景位图。

表盘位图指针中心位置选择：

该项设置将决定表盘显示控件的指针旋转中心点位于表盘背景位图的位置，点击右则按钮，则可在对话框右侧的显示区中用鼠标点击选择指针中心位置在表盘背景位图中的偏移位置，或者可以直接利用下方的 X 和 Y 轴设置编辑框中直接输入坐标点。

表盘零位角度设置：

表盘显示控件零位角度的设置，可直接点击右侧按钮，在对话框右侧的显示区中进行可视化操作设置，或直接在编辑框中输入零位角度的数值，数值范围为 0~360。

表盘最大角度设置：

表盘显示控件最大角度的设置，可直接点击右侧按钮，在对话框右侧的显示区中进行可视化操作设置，或直接在编辑框中输入最大角度的数值，数值范围为 0~360。

指针范围数值量化：

表盘显示控件的指针起始位置都由角度值决定，为了呈现更好的数值效果，可以设置该项数值，以便于将控件设置的角度范围进行整型数的量化，量化后，表盘控件的数值将对应角度范围的量化值。比如上图中，设置的零位角度值为 7，最大角度值为 174，量化值为 100；则表盘显示控件的角度范围为 7~174

度，则当表盘显示控件当前的值为 0 时，表示表盘显示控件的指针显示在零位（7 度）的位置上，数值每增加 1，则指针顺时针旋转 1.67 度的角度。

表盘指针绘线设置：

当表盘显示控件设置为直线绘制类型时，该区域内的设置有效。

指针线宽设置：可以选择指针绘制时，直线的线宽。

指针线长设置：可根据具体的需要设置合适的指针线长，单位为像素点。

指针线条颜色设置：可根据具体的需要，设置合适的指针颜色，点击右侧的色块，会弹出颜色选择对话框。

表盘指针位图设置：

当表盘显示控件设置为位图资源类型时，该区域内的设置有效。

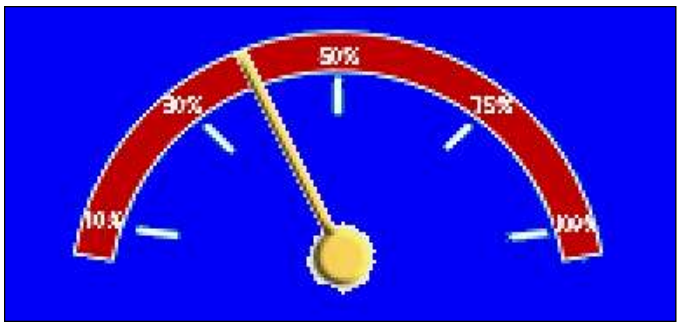
指针位图选择：点击右侧按钮，可弹出位图资源选择对话框，用户可从里面选择已加载到资源文件的位图资源作为表盘的指针位图。

用户在制作指针位图时，应注意以下几点：

- 位图图像除有效的指针线条区域之外的地方，应使用有别于指针线条颜色的背景色，这样在指针位图显示时，将会自动把指针线条之外的区域清除不显示，但背景色最好与表盘背景位图的背景颜色接近一些为好；
- 位图图像在制作时，应以指针旋转角度为 0 值时进行制作。

下图即为示例的指针位图图像：

可以看到，图像除了黄色区域那块的指针线条，其它地方都选择了有别于它的白色，这样在控件显示时，模块会自动清除边上的区域，显示效果如下图所示：



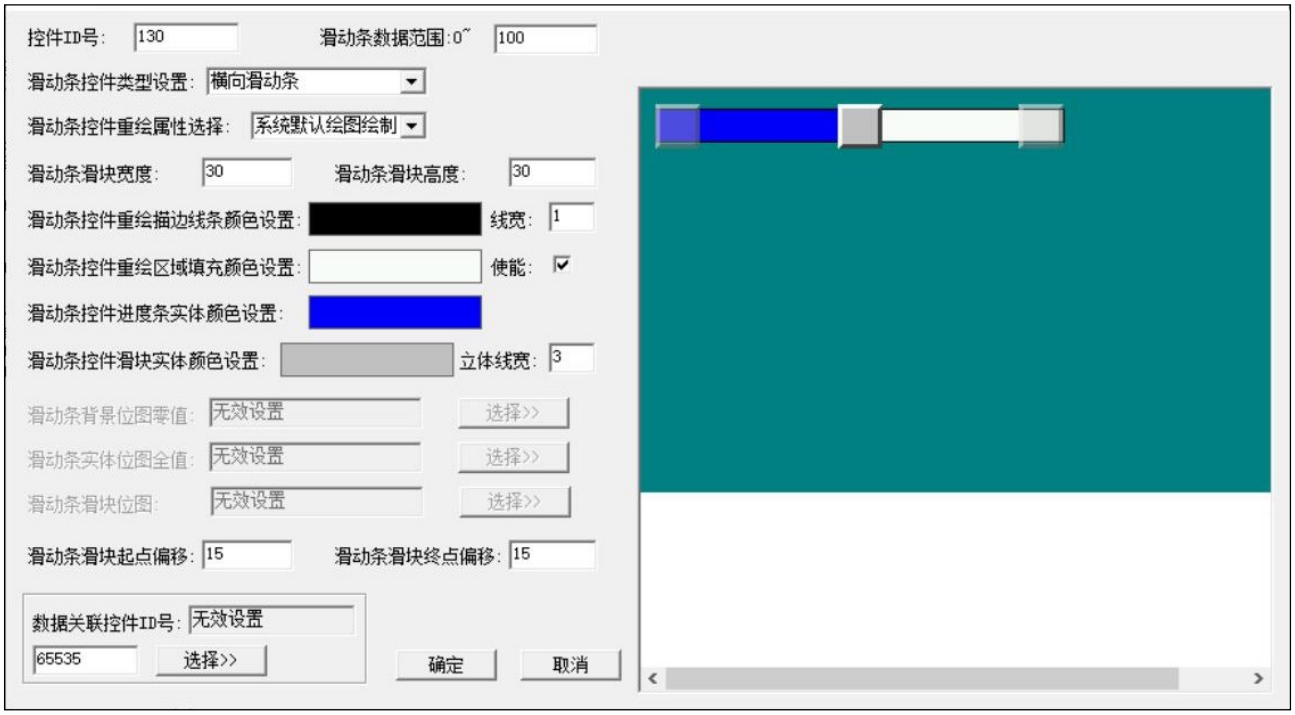
指针位图旋转中心位置选择：

可以点击右侧按钮，这样就可以在对话框的右侧显示区中利用鼠标点击选择指针位图的旋转中心点，或也可以通过下方的两个编辑框直接输入数值进行设置。

3.4.12 滑动条控件

滑动条控件（SliderCtrl）支持横向滑动或竖向滑动两种类型，可由系统绘制或由用户所设置的位图资源进行绘制；无触摸版模块该控件不响应用户按键消息。该控件不支持控件响应消息配置，但允许其它控件对它进行控制。

在 EzUITool 工具中，滑动条控件的配置画面如下图所示：



滑动条控件类型设置：可将控件设置为横向滑动条或竖向滑动条，在工具中创建新的滑动条控件时，工具将根据所选择区域的宽高比进行默认的横向竖向属性设置；

滑支条控件重绘属性选择：可将控件设置为系统绘图绘制或表位图资源绘制；

- 系统默认绘图绘制：该类型表示控件在显示时，滑动条的背景以及滑动条实体进度、滑动块均由用户配置的线宽、绘图色等决定，上图中示例的即为系统默认绘图绘制类型。
- 位图素材绘制：该类型表示控件在显示时，控件的背景、实体进度以及滑动块均由资源项里的位图资源进行绘制显示，需要注意的是，控件在显示更新位图素材时，将会提取位图的四个顶角的颜色数据，如四点颜色相同，则认为是省缺色（也即显示这些位图素材时，与省缺色相同的像素点将不进行显示）。

滑块宽度：在使用系统默认绘图进行绘制控件时，该项设置可以重新定义滑块的宽度（X 轴）大小，该项数值更新后，点击对话框其它设置区域后，将会在右侧的效果显示区更新当前配置所对应的显示效果；

滑块高度：在使用系统默认绘图进行绘制控件时，该项设置可以重新定义滑块的高度（Y 轴）大小，该项数值更新后，点击对话框其它设置区域后，将会在右侧的效果显示区更新当前配置所对应的显示效果；

重绘描边线条颜色设置：当控件重绘属性为系统默认绘图时，控件将会以右侧线宽设置的线宽值像素点，使用重绘线条颜色进行绘制控件边框背景；如描边线宽设置为 0，则表示控件不进行描边绘制，也即描边线条颜色无意义；使用鼠标点击颜色块，将会弹出颜色设置对话框进行选择设置；

重绘描边线宽：当控件重绘属性为系统默认绘图时，该项可设置 0~8 的数值，表示控件背景重绘时

的描边线宽；

重绘区域填充颜色设置：当控件重绘属性为系统默认绘图时，该项可设置控件区域的背景颜色；如右侧的“使能”钩选状态下，控件显示时将以所设置的颜色对控件区域进行颜色填充，以此为控件背景；

进度条实体颜色设置：当控件重绘属性为系统默认绘图时，该项可设置控件的进度条实体颜色，与进度条控件的类似；

滑块实体颜色设置：当控件重绘属性为系统默认绘图时，该项可设置滑块的填充颜色，

立体线宽：当控件重绘属性为系统默认绘图时，该项可设置滑块的立体效果线宽，当其值为 0 时，表示不使用立体效果；

背景位图：当控件重绘属性为位图素材绘制时，点击右侧按钮，可弹出位图资源选择对话框，用户可从里面选择已加载到资源文件的位图资源作为控件的背景位图；

实体位图：当控件重绘属性为位图素材绘制时，用户可选择已加载到资源文件的位图资源作为控件的进度条实体位图；

滑块位图：当控件重绘属性为位图素材绘制时，用户可选择已加载到资源文件的位图资源作为控件的滑块位图；

滑块起点偏移：滑动条控件的滑块在零值时与控件起始点（横向为左侧起点，竖向为下部起点）的偏移像素点设置，通常由系统默认根据控件滑块大小自动完成设置，但也允许用户自行修改；修改该项数值后，可从对话框右侧的效果示意图中看到相应的变化；

滑块终点偏移：滑动条控件的滑块在零值时与控件结束点（横向为右侧结束，竖向为上部结束）的偏移像素点设置，通常由系统默认根据控件滑块大小自动完成设置，但也允许用户自行修改；修改该项数值后，可从对话框右侧的效果示意图中看到相应的变化；

滑动条数据范围：该项设置将决定动条控件进度条实体在显示满值时的最大数值，控件将会以该数值作为参考，显示当前进度条实体的位置，当控件当前数值超出该设置时，控件将显示满值。

数据关联控件 ID 号：滑动条控件允许设置该控件的数据关联控件 ID 号。

用于制作滑动条控件的位图素材设置：

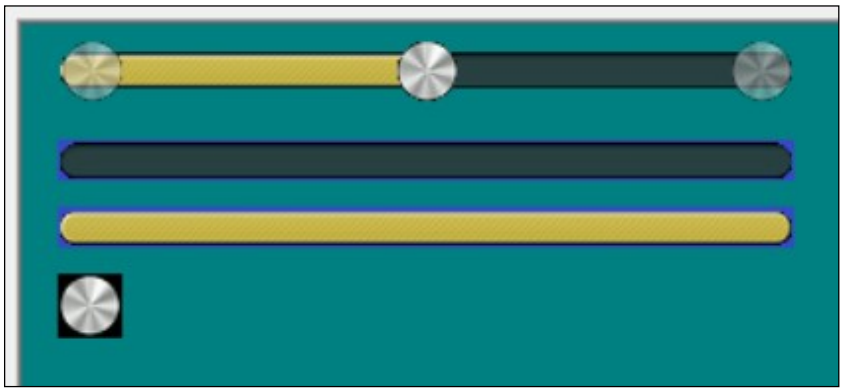
当使用位图素材来进行滑动条控件显示时，要求背景位图与实体位图的大小尺寸相同；而滑块位图不宜太小，太小则触摸设置时不太好点击或移动滑块。

控件所使用的图像素材在显示时，将会默认以省缺色的形式显示，模块将会获取图像的四个角的四点像素颜色值，如四点颜色相同，则会认定该颜色为省缺色，而后在显示位图时将与省缺色相同的素材点不进行显示；所以用制作位图素材时，应注意以下：

位图图像除有效区域之外的地方，应使用有别于有效区域颜色的背景色，这样在位图显示时，将会自动把有效区域之外的区域清除不显示，但背景色最好与表盘背景位图的背景颜色接近一些为好。

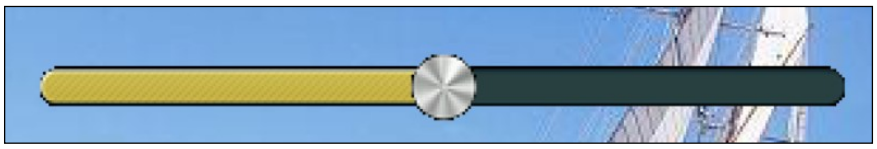
下面的三张位图为示例的位图图像，分别将会用于背景位图、实体位图和滑块位图：

下图为控件配置对话框中的显示效果：



图中两侧透明虚化显示的，则为控件滑块的零值、满值位置示意。

下图为模块在界面中的显示效果示意：



3.4.13 二维码显示控件

二维码显示控件（QRCoderDisCtrl）是一个可将用户预置或单片机即时发送字符串转换为二维码图像的显示类控件，字符串转换为二维码图像显示时，控件将自动选择合适的版本以及纠错等级，用户单片机只需要给控件发送字符串即可自动生成二维码图像并显示；生成的二维码图像将根据控件所配置的显示区域大小进行自动放大显示。

在 EzUITool 工具中，二维码显示控件的配置画面如下图所示：

控件留白区域显示设置：在创建控件时用户在显示区域内所配置的区域或许并不是正方形的，并且生成的二维码图像自动放大后，可能并不能完全填充满所配置的区域，所以在此定义了一个二维码图像显示相对所配置区域的留白区域，用户可以根据自己的需要选择合适的留白区域处理方式；

- 使用背景色填充：该选择表示控件在显示二维码图像时，留白区域使用控件所配置的背景色进行填充处理，上图中示例的即为使用背景色填充的示例。
- 不进行绘制（显示背景）：该选择表示控件在显示二维码图像时，留白区域将显示出背景画面。

背景颜色设置：设置二维码图像显示时的背景颜色，鼠标点击右侧的颜色块可弹出对话框进行设置；

前景颜色设置：设置二维码图像显示时的前景颜色；

数据共享：当控件选择为数据共享模式（即勾选上左侧的单选项）时，控件将无数据缓冲区，也无法通过指令对该控件进行字符串更新或读取，控件需要设置数据共享的目标控件 ID 号，设置好后，控件将会从所设置的目标控件获取二维码图像进行显示。数据共享模式的对立状态为自有数据模式，自有数据模式下，控件将允许设置自身的字符串数据缓冲区大小，以及可以为控件预置字符串内容。

二维码字符串缓冲区大小设置：二维码显示控件实际上将用户设置给控件的字符串转换为二维码图像进行显示，所以控件需要有一个字符串的缓冲区用于保存用户设置的字符串内容，该缓冲区可以根据需要进行设置其大小。该项设置需要控件处于自有数据模式下可配置。

预置字符串：用户可为二维码显示控件预置模块初始化后的字符串内容，如有预置的字符串，二维码显示控件将会在界面显示进，将预置的字符串转换为二维码图像进行显示，如控件当前无字符串内容，二维码显示控件将不会显示在界面当中。该项设置需要控件处于自有数据模式下可配置。

更新二维码字符串按钮：按下该按钮，可以将用户预置的二维码显示控件的字符串转换为二维码图像显示在对话框右侧的预览区域中。

数据共享控件 ID 号：用户可在控件处于共享数据模式下时，按下右侧的按钮，在弹出的对话框中选择一个模式为自有数据模式的二维码显示控件，这样，当前控件将会在界面显示更新时，从目标控件获取二维码图像进行显示。

3.5 按键消息

无触摸的 EzUI 模块是基于界面以及控件的 GUI 系统，模块的 GUI 服务引擎会替用户管理界面以及所有的控件，很多控件之间的响应、处理、显示更新、以及界面的切换等操作，不需要用户介入控制；与 EzUI 系列带触摸的模块不同，EzUI 系列不带触摸的模块需要接收用户控制器（MCU）发来的按键消息，并根据按键消息的键值对界面进行响应操作；在此将会对 EzUI 模块的 GUI 服务引擎的一些基本规则进行详细说明。

3.5.1 按键消息键值定义

EzUI 模块可以按收的按键消息包含键值和按键事件类型，下表定义了各个按键消息的键值内容：

键值	名称	允许事件类型	功能描述
0x30	GUI_SCANCODE_0 数字键盘 0 按键	0：普通按键事件	数字按键，‘0’键；可用于数值输入以及下拉选框控件选择；
0x31	GUI_SCANCODE_1 数字键盘 1 按键	0：普通按键事件	数字按键，‘1’键；可用于数值输入以及下拉选框控件选择；
0x32	GUI_SCANCODE_2 数字键盘 2 按键	0：普通按键事件	数字按键，‘2’键；可用于数值输入以及下拉选框控件选择；
0x33	GUI_SCANCODE_3 数字键盘 3 按键	0：普通按键事件	数字按键，‘3’键；可用于数值输入以及下拉选框控件选择；
0x34	GUI_SCANCODE_4 数字键盘 4 按键	0：普通按键事件	数字按键，‘4’键；可用于数值输入以及下拉选框控件选择；
0x35	GUI_SCANCODE_5 数字键盘 5 按键	0：普通按键事件	数字按键，‘5’键；可用于数值输入以及下拉选框控件选择；
0x36	GUI_SCANCODE_6 数字键盘 6 按键	0：普通按键事件	数字按键，‘6’键；可用于数值输入以及下拉选框控件选择；
0x37	GUI_SCANCODE_7 数字键盘 7 按键	0：普通按键事件	数字按键，‘7’键；可用于数值输入以及下拉选框控件选择；
0x38	GUI_SCANCODE_8 数字键盘 8 按键	0：普通按键事件	数字按键，‘8’键；可用于数值输入以及下拉选框控件选择；
0x39	GUI_SCANCODE_9 数字键盘 9 按键	0：普通按键事件	数字按键，‘9’键；可用于数值输入以及下拉选框控件选择；
0x2D	GUI_SCANCODE_DVE 数字键盘负号’-’按键	0：普通按键事件	数字按键，‘-’键；可用于数值输入
0x2E	GUI_SCANCODE_DOT 数字键盘小数点’.’按键	0：普通按键事件	数字按键，‘.’键；可用于浮点数值输入
0x0C	GUI_SCANCODE_CLR	0：普通按键事件	控制按键，清除键，用于数值输入

	控制按键，清除键		
0x0D	GUI_SCANCODE_TAB 控制按键，切换键	0：普通按键事件	控制按键，切换键，用于切换当前选择控件
0x1B	GUI_SCANCODE_ESC 控制按键，取消键	0：普通按键事件	控制按键，取消键，用于取消当前输入或当前选择
0x5A	GUI_SCANCODE_ENTR 控制按键，确定键	0：普通按键事件 1：确定键释放 2：确定键按下	控制按键，确定键，用于确定完成当前输入（数值控件或下拉选择控件）或确定按下当前选择按钮（触摸区域控件或位图按钮控件）
0x66	GUI_SCANCODE_BSP 控制按键，回删键	0：普通按键事件	控制按键，回删键，用于数值输入
0x6B	GUI_SCANCODE_LEFT 控制按键，左方向键	0：普通按键事件	控制按键，左方向键，用于数值输入
0x74	GUI_SCANCODE_RIGHT 控制按键，右方向键	0：普通按键事件	控制按键，左方向键，用于数值输入
0x71	GUI_SCANCODE_DEL 控制按键，删除键	0：普通按键事件	控制按键，删除键，用于数值输入
0x75	GUI_SCANCODE_UP 控制按键，向上键	0：普通按键事件	控制按键，向上键，用于数值输入
0x72	GUI_SCANCODE_DOWN 控制按键，向下键	0：普通按键事件	控制按键，向下键，用于数值输入

EzUI 模块还有一个可通过用户指令进行设置的状态，在此称其为“数字键盘使能/禁止”，模块上电后，默认情况下为数字键盘禁止的状态，此时用户发送按键消息来进行数值控件的数值设置时，可以只通过控制按键来完成，如左、右方向键、上、下键等，而无需用户发送 0~9 的数值按键消息，这样可以方便用户使用少量的按键即可完成数值的输入功能；而用户如通过指令来使能数字键盘后，模块的数值控件在进行数值输入时，将采取不同的方式，下面的章节将会对这些不同之处进行说明。

3.5.2 控件状态定义

从 EzUI 无触摸模块的按键消息服务的角度看，模块的 GUI 系统当中，定义了控件的三种状态，分别是：

- **空闲状态**：模块上电完成初始化显示后，所有的控件都处于空闲状态，也即无控件处于选择或设置状态；
- **选择状态**：用户可通过发送 GUI_SCANCODE_TAB 切换键消息来切换当前选择控件，或者通过指令来设置指定 ID 号的控件处于选择状态；控件处于选择状态后方可响应其它的按键消息，需要注意的是并非所有的控件都可以选择，只有按钮类（触摸区域控件和位图按钮）和可设置的数值控件、下拉选择控件可以被选择；
- **设置状态**：设置状态仅限于数值控件和下拉选择控件，当这两种控件处于选择状态时，用户发送 GUI_SCANCODE_ENTR 按键消息（普通按键事件），然后控件就会处于设置状态，此时控

件才可响应其它的按键消息。

3.5.3 控件响应按键消息

EzUI 无触摸版本的模块主要靠响应用户控制器（MCU）发送来的按键消息指令进行 GUI 服务引擎的自动服务，有一个基本原则：只有当前界面内设置的控件允许响应按键消息。GUI 服务引擎在接收到用户发来的按键消息后，会自动判别当前控件的状态，对控件进行选择，或者触发控件的按下、释放以及控件的完成设置消息，这些触发的控件消息将会通过模块的通讯端口，以 GUI 服务引擎返回数据包的形式发送煤给用户控制器。

接下来，将对每一个允许响应按键消息的控件进行详述其响应按键消息的基本原则。

区域按钮控件：

- 选择状态：当该类控件处于选择状态时，将会以外框 2 像素点的反显呈现该状态；
- 控件按下事件：当区域按钮控件处于选择状态下时，用户发送 GUI_SCANCODE_ENTR 按键消息，然后控件即发生按下事件，控件将会按用户所设置的显示响应属性进行显示刷新，若控件所设置的属性中，设置为响应按下事件且消息发送使能勾选，则控件会触发 GUI 服务引擎发送一条区域按钮控件按下消息给用户；注：如发送的 GUI_SCANCODE_ENTR 按键消息的事件类型为“普通按键事件”，则控件会在 0.3 秒后自动发生释放事件；而当发送的 GUI_SCANCODE_ENT 按键消息的事件类型为“确定键按下”则控件将会保持按下状态直到事件类型为“确定键释放”的 GUI_SCANCODE_ENTR 按键消息接收到；
- 控件释放事件：当区域按钮控件释放事件时，控件将会恢复选择状态下的显示效果，若控件所设置的属性中，设置为响应释放事件且消息发送使能勾选，则控件触发 GUI 服务引擎发送一条区域按钮控件释放消息给用户；而如果该控件配置了控件消息，则会在 GUI 服务引擎内部发生一条控件消息，将会按照该条消息的配置切换界面或对其它控件进行操作。注：如引发控件按下事件的按键消息为“普通按键事件”的 GUI_SCANCODE_ENTR，则控件会在 0.3 秒后自动发生释放事件；而当引发控件按下事件按键消息的事件类型为“确定键按下”则控件将会在接收到事件类型为“确定键释放”的 GUI_SCANCODE_ENTR 按键消息时才会引发释放事件；

位图按钮控件：

位图按钮控件要区分其类型属性，如控件设置为非“乒乓开关”类型，则按如下响应触摸消息：

- 选择状态：当该类控件处于选择状态时，将会以外框 2 像素点的反显呈现该状态；
- 控件按下事件：当位图按钮控件处于选择状态下时，用户发送 GUI_SCANCODE_ENTR 按键消息，然后控件即发生按下事件，控件将会按用户所设置的显示响应属性进行显示刷新，若控件所设置的属性中，设置为响应按下事件且消息发送使能勾选，则控件触发 GUI 服务引擎发送一条位图按钮控件按下消息给用户；注：如发送的 GUI_SCANCODE_ENTR 按键消息的事件类型为“普通按键事件”，则控件会在 0.3 秒后自动发生释放事件；而当发送的 GUI_SCANCODE_ENTR 按键消息的事件类型为“确定键按下”则控件将会保持按下状态直到事件类型为“确定键释放”的 GUI_SCANCODE_ENTR 按键消息接收到；
- 控件释放事件：当位图按钮控件释放事件时，控件将会恢复选择状态下的显示效果，若控件所设置的属性中，设置为响应释放事件且消息发送使能勾选，则控件会触发 GUI 服务引擎发送一条位图按钮控件释放消息给用户；而如果该控件配置了控件消息，则会在 GUI 服务引擎内部发生一条控件消息，将会按照该条消息的配置切换界面或对其它控件进行操作。注：如引发控件按下事件的按键消息为“普通按键事件”的 GUI_SCANCODE_ENTR，则控件会在 0.3 秒后自动发生释放事件；而当引发控件按下事件按键消息的事件类型为“确定键按下”则控件将会在接

收到事件类型为“确定键释放”的 GUI_SCANCODE_ENTR 按键消息时才会引发释放事件；

若位图按钮控件设置为“乒乓开关”类型，则按如下响应按键消息事件：

- 选择状态：当该类控件处于选择状态时，将会以外框 2 像素点的反显呈现该状态；
- 控件按下事件：当位图按钮控件处于选择状态下时，用户发送 GUI_SCANCODE_ENTR 按键消息，控件将会按用户所设置的显示响应属性进行显示刷新；注：如发送的 GUI_SCANCODE_ENTR 按键消息的事件类型为“普通按键事件”，则控件会在 0.3 秒后自动发生释放事件；而当发送的 GUI_SCANCODE_ENTR 按键消息的事件类型为“确定键按下”则控件将会保持按下状态直到事件类型为“确定键释放”的 GUI_SCANCODE_ENTR 按键消息接收到；
- 控件释放事件：当位图按钮控件释放事件时，控件将切换显示位图按钮控件设置时所设定的图标，并将位图按钮控件的当前状态进行取反（若之前为 0 则改为 1，若之前为 1 则改为 0），如消息发送使能勾选，控件会触发 GUI 服务引擎发送一条位图按钮控件状态改变消息给用户；而如果该控件配置了控件消息，则会在 GUI 服务引擎内部发生一条控件消息，将会按照该条消息的配置切换界面或对其它控件进行操作。注：如引发控件按下事件的按键消息为“普通按键事件”的 GUI_SCANCODE_ENTR，则控件会在 0.3 秒后自动发生释放事件；而当引发控件按下事件按键消息的事件类型为“确定键按下”则控件将会在接收到事件类型为“确定键释放”的 GUI_SCANCODE_ENTR 按键消息时才会引发释放事件；

数值控件：

数值控件可设置属性为不允许输入数值，此时控件将只能接受用户通过模块通讯端口发送的数据包来对控件的当前数值进行更新，有更新数值时，模块将会自动进行显示的更新。

如控件设置的属性为允许输入数值，则按如下说明进行响应：

- 选择状态：当该类控件处于选择状态时，将会以外框 2 像素点的反显呈现该状态；
- 设置状态：当数值控件处于选择状态时，用户发送 GUI_SCANCODE_ENTR 按键消息（“普通按键事件”或“确定键释放”），控件将会进入设置状态，数值控件在设置状态下的显示效果与当前模块的“数字键盘使能/禁止”设置有关，将在后面的小节中单独说明，而控件处于设置状态下时，方可响应用户发送的其它按键消息，并完成相应的数值设置操作；数值控件处于设置状态时，用户可以通过发送 GUI_SCANCODE_ENTR 按键消息来通知 GUI 服务引擎完成当前数值控件的输入，此时控件将会触发 GUI 服务引擎给用户发送一条数值控件完成输入消息；而当控件处于设置状态时，用户发送 GUI_SCANCODE_ESC 取消按键消息时，将会退出设置状态并恢复为选择状态；而当控件处于设置状态时，用户发送 GUI_SCANCODE_TAB 切换按键消息，则控件退出设置状态，并退出选择状态。

下拉选框控件：

下拉选择控件可设置属性为不允许输入选择，此时控件将只能接受用户通过模块通讯端口发送的数据包来对控件的当前选择项进行更新，有选择项更新时，模块将会自动进行显示的更新。

如控件设置的属性为允许输入选择时，则按如下说明进行响应：

- 选择状态：当该类控件处于选择状态时，将会以外框 2 像素点的反显呈现该状态；
- 设置状态：当下拉选择控件处于选择状态时，用户发送 GUI_SCANCODE_ENTR 按键消息（“普通按键事件”或“确定键释放”），控件将会进入设置状态；控件处于设置状态下时，将会把控件的区域进行取反显示进行呈现，此时控件允许用户发送 GUI_SCANCODE_UP 向上键或

GUI_SCANCODE_DOWN 向下键来进行选项的选择；当控件处于设置状态时，用户可以通过发送 GUI_SCANCODE_ENTR 按键消息来通知 GUI 服务引擎完成当前控件的输入，此时控件将会触发 GUI 服务引擎给用户发送一条下拉选择控件完成输入消息，而如果该控件配置了控件消息，则会在 GUI 服务引擎内部发生一条控件消息，将会按照该条消息的配置切换界面或对其它控件进行操作；而当控件处于设置状态时，用户发送 GUI_SCANCODE_ESC 取消按键消息时，将会退出设置状态并恢复为选择状态；而当控件处于设置状态时，用户发送 GUI_SCANCODE_TAB 切换按键消息，则控件退出设置状态，并退出选择状态。

3.5.4 控件消息响应

EzUIH 系列模块的 GUI 服务引擎允许对一些控件配置控件消息，控件消息可通过控件的事件来驱动界面的切换或者对界面内控件数据的更新，而这些消息的传递、响应操作都无需用户进行介入，可以更灵活的实现用户所需的人机交互功能。

控件消息的配置，在 EzUITool 工具中的配置对话框如下图所示：

响应控件 ID 号：表示该控件消息由哪一个控件触发产生，允许一个控件产生多条控件消息。

响应匹配条件：表示该条控件消息产生的条件，不同的控件类型触发的控件消息可选择的匹配条件不同，视具体的控件类型而定。

响应目标：该值表示控件消息产生后，对哪一个控件或界面进行操作，如是对控件进行操作，则该数值表示目标控件的 ID 号，如是对界面进行切换显示操作，则该数值为目标界面的索引号。

响应动作：

表示该条控件消息的操作类型，有如下几项待选：

- 切换界面：表示该条控件消息受触发产生后，将会切换到索引号为前面设置的“响应目标”指示的目标界面索引号；
- 对目标控件清零：表示该条控件消息受触发产生后，将会对前面设置的“响应目标”指示的目标控件进行数据/数值清零操作；
- 对目标控件加值 [1]：表示该条控件消息受触发产生后，将会对前面设置的“响应目标”指示的目标控件进行数据/数值加值操作，而加值操作时加值的大小由“响应动作参数”的数值决定；
- 对目标控件减值 [2]：表示该条控件消息受触发产生后，将会对前面设置的“响应目标”指示的目

标控件进行数据/数值减值操作，而减值操作时减值的大小由“响应动作参数”的数值决定。

- **使能目标控件：**表示该条控件消息受触发产生后，将会对前面设置的“响应目标”指示的目标控件进行使能操作，如之前目标控件为禁能且消除显示状态，则会将目标控件进行重绘显示。
- **禁能目标控件：**表示该条控件消息受触发产生后，将会对前面设置的“响应目标”指示的目标控件进行禁能操作，目标控件被禁能后，将不允许显示更新以及响应触摸消息，但仍将保持显示。
- **禁能并消除目标控件：**表示该条控件消息受触发产生后，将会对前面设置的“响应目标”指示的目标控件进行禁能操作，并将目标控件在界面中消除显示。
- **使用目标控件设置系统 RTC ：**表示该条控件消息受触发后，将会使用前面设置的“响应目标”指示的目标控件（仅限时间显示控件及日期显示控件，且要求控件类型为“用于设置模块 RTC 系统时间”）的数据对模块内部的 RTC 系统时间进行更新。

注 1：如响应目标控件为字符串控件，则响应动作为对目标控件加值时，将会使用响应参数中设置的数值对应的 ASCII 码字符加入到字符串控件当中，字符串控件的数据为字符串内容，所以加值相当于在当前字符串控件的字符串后面添加所指定的字符。

注 2：如响应目标控件为字符串控件，则响应动作为对目标控件减值时，将会在当前字符串控件的字符串内容基础之上，清除最后的一个字符然后刷新显示。

响应动作参数：

响应动作参数会根据控件消息不同的响应目标控件类型进行不同的配置，分为以下几种情况：

- 不可配置的情况：响应动作为“返回之前界面”、“对目标控件清零”、“**使用目标控件设置系统 RTC**”时，该参数不可也不需要设置；
- 配置 1 个字节动作参数：响应动作为：“对目标控件加值”、“对目标控件减值”时，且响应目标控件类型为有数据的控件（如数值控件、进度条控件、表盘控件、位图动画控件）时，可以配置 1 个字节的动作参数；
- 配置浮点数动作参数：响应动作为：“对目标控件加值”、“对目标控件减值”时，且响应目标控件类型为浮点数类型的“数值控件”时，可勾选“设置浮点数”，并设置好相应的数据即可；
- 对时间显示控件进行加减值操作：响应动作为：“对目标控件加值”、“对目标控件减值”时，且响应目标控件类型为“用于设置模块 RTC 系统时间”的“时间显示控件”时；允许设置响应动作参数值为 0~31，且可勾选要操作的时间显示控件时、分、秒选项（即表示该控件消息驱动修改的具体内容）；
- 对日期显示控件进行加减值操作：响应动作为：“对目标控件加值”、“对目标控件减值”时，且响应目标控件类型为“用于设置模块 RTC 系统日期”的“日期显示控件”时；允许设置响应动作参数值为 0~31，且可勾选要操作的时间显示控件年、月、日选项（即表示该控件消息驱动修改的具体内容）。
- 对字符串控件进行加值操作：响应动作为：“对目标控件加值”，且响应目标控件类型为“字符串控件”时，用户可以设置 1 个字节的响应参数，该设置可以直接设置 10 进制的参数数值；也以勾选“字符输入”的单选按钮，然后直接设置字符（ASCII 码），这样相当于输入该字符的 ASCII 码值。

3.5.5 按键消息触发切换控件选择示例

下面将用示例来说明 EzUIH 系列模块如何响应用户发送的 GUI_SCANCODE_TAB 切换键来进行控件选择的切换。

下例中，资源文件定义了一个界面，该界面中有四个不同的控件，ID 号分别为 100（区域按钮控件）、101（位图按钮控件）、102（数值控件）、103（下拉选择控件），下图为 EzUITool 截图：

该资源在模块上电显示初始化后，如下图所示：

如上图所示，数值控件在上电初始化后，显示当前的数值为 0，而最下方的下拉选择控件由于初始化后没有有效选项被选择，显示空白；此时用户如发送 GUI_SCANCODE_TAB 切换键消息给模块，则模块依次切换控件的选择，下面五张图片分用户发送五次 GUI_SCANCODE_TAB 切换键消息的效果：

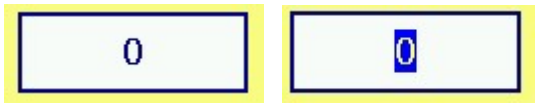
控件切换选择时，顺序将按照资源文件在定义时，这些控件在界面控件列表中的序顺进行切换选择，而当中间有不允许选择的控件，如字符串控件、进度条控件、波形控件、位图动画控件时，将会跳过这些不允许选择的控件；而在任意时候，用户均可通过发送指令的方式，来指定选择某个处于当前显示界面的控件，或者取消当前选择。

3.5.6 数值控件响应按键消息进行数值设置

当数值控件定义为允许输入数值时，则数值控件允许用户发送按键消息事进行数值的设置，本小节将介绍数值控件在“数字键盘使能/禁止”的两种前提下的数值设置效果。

“数字键盘禁止”状态下的数值设置：

在“数字键盘禁止”状态下，数值控件处于选择状态时，用户发送 GUI_SCANCODE_ENTR 按键消息，然后控件将进入设置状态，下图左侧为处于选择状态下的数值控件，右侧为刚进入设置状态下的数值控件：



图中右侧显示的数值控件内部，在’0’数据位的蓝色区块实际上是闪烁显示的，指明当前是在该位进

行数值编辑，在此将闪烁显示的区块称为当前编辑位指示光标；此时用户可通过发送 GUI_SCANCODE_UP 向上键或 GUI_SCANCODE_DOWN 向下键按键消息来进行该位数值的设置，如下图为按下一次、两次、三次的向上键时的显示效果：

在完成某一数据位的编辑输入后，用户可以发送 GUI_SCANCODE_LEFT 向左键或 GUI_SCANCODE_RIGHT 向右键按键消息来进行编辑位的切换，如数值尚未超出控件允许的最大数值，且当前编辑位处于最左或最右边的位，则会在相应的编辑位切换时扩展出一个数据位，在前面操作的基础上，用户发送一次 GUI_SCANCODE_LEFT 向左键按键消息，则显示效果如下图所示：

随后发送四次 GUI_SCANCODE_UP 向上键按键消息后，如下图所示：

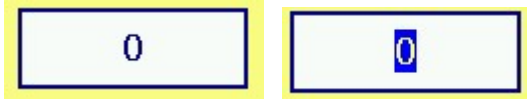
随后再发送两次 GUI_SCANCODE_RIGHT 向右键按键消息，则如下图所示：

此时再发送三次 GUI_SCANCODE_DOWN 向下键按键消息，如下图所示：

完成数值输入后，用户发送 GUI_SCANCODE_ENTR 按键消息，控件则完成输入，退出设置状态，并更新当前控件的数值显示内容。如下图所示：

“数字键盘使能”状态下的数值设置：

在“数字键盘使能”状态下，数值控件处于选择状态时，用户发送 GUI_SCANCODE_ENTR 按键消息，然后控件将进入设置状态，下图左侧为处于选择状态下的数值控件，右侧为刚进入设置状态下的数值控件：

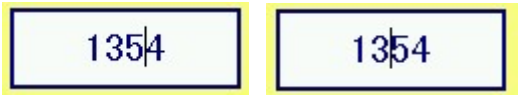


图中右侧显示的数值控件内部，在’0’数据位的蓝色区块实际上是对当前数值控件的数值进行反显，

表示如当前用户发送数字按键消息时，将会整个将原来的数值删除并插入新的数据位；在此时如用户发送 GUI_SCANCODE_1 数字键盘 1 按键消息，则显示如下图所示：

上图中，'1'数据位右侧的竖线，为光标，表示如用户再发送数字按键消息时，将从光标后方插入新的数据位；下图为用户分别依次发送 GUI_SCANCODE_3 数字键盘 3、GUI_SCANCODE_5 数字键盘 5、GUI_SCANCODE_4 数字键盘 5 后的显示新效果：

可以看到输入的数字依次从光标后插入，而用户也可发送 GUI_SCANCODE_LEFT 向左键或 GUI_SCANCODE_RIGHT 向右键修改光标位置，如下图为依次发送两次向左键的显示效果：

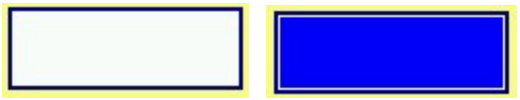


用户也可发送 GUI_SCANCODE_BSP 回删键删除光标左侧的数据位，而发送 GUI_SCANCODE_DEL 删除键将会删除光标右侧的数据位。

完成数值输入后，用户发送 GUI_SCANCODE_ENTR 按键消息，控件则完成输入，退出设置状态，并更新当前控件的数值显示内容。

3.5.7 下拉选择控件响应按键消息进行选项设置

当下拉选择控件定义为允许输入选项时，则数值控件允许用户发送按键消息事进行选项的设置。下图左侧为处于选择状态下的选择控件，而右侧的图片为用户发送 GUI_SCANCODE_ENTR 按键消息后，控件处于设置状态下的显示效果：



由于当前控件无有效选择项被选中，则控件区域当中为空；控件处于设置状态后，控件区域将取反显示。

控件处于设置状态后，用户可发送 GUI_SCANCODE_UP 向上键或 GUI_SCANCODE_DOWN 向下键按键消息来进行选项的选择，如下图为依次发送三次向下键的显示效果：

完成选择后，用户发送 GUI_SCANCODE_ENTR 按键消息，则控件退出设置状态，并更新完成设置后新的选择项显示，如下图所示：

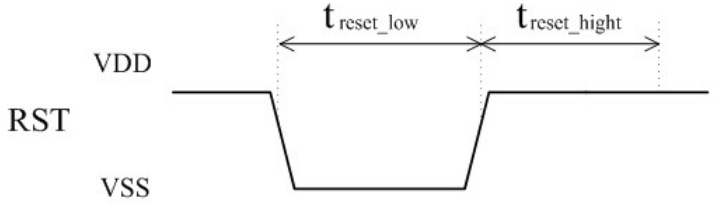
4 模块控制方法

EzUIH 系列模块 UART 串口版本提供 UART 接口(TTL 电平，请注意不要将电脑上的串口或者 USB 转串口线/模块出来的 RS232 电平的串口信号与模块直接连接！)与用户进行数据的交互，用户可对模块进行两大类操作，其一为普通显示指令操作，另一为对模块的 GUI 服务引擎进行操作（控件数据/状态读取、设置，界面操作等）；此外，如模块在正常显示模式下（即非 USB 模式下），将模块 USB 口与电脑连接，则此时模块的 UART 口将停止工作。

下面介绍 EzUIH 系列模块的控制时序，以及相关的指令以及详细控制方法。

4.1 复位操作时序

EzUIH 系列模块内部有复位电路，上电后，模块的 GUI 服务引擎需要大约 800ms 的时间启动，此期间用户不应对模块进行操作（包括通过 UART 接口给模块发送数据包）。

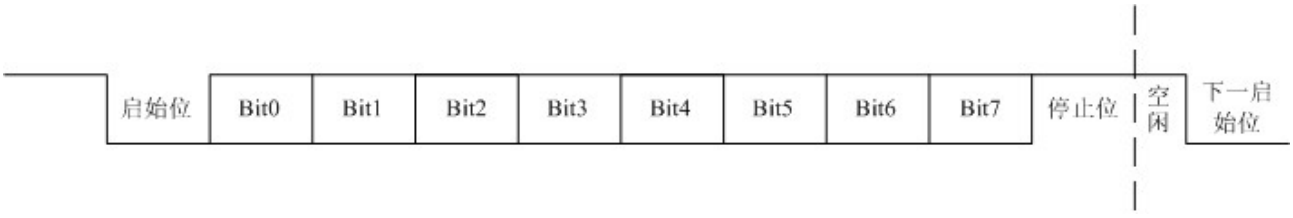


EzUIH 模块启动时间特别说明：

EzUIH 模块除了硬件本身的复位、初始化时间外，还有模块上运行的用户编辑的资源文件的启动时间；资源文件中定义的界面的具体配置将会直接影响 EzUIH 模块的实际启动时间（可以简单理解为模块上电后到显示屏亮起的过程），所以 EzUIH 模块的软硬件启动时间是不固定的。在用户使用时，强烈建议用户不要在系统上电后通过固定延时的方式来对模块进行启动等待；而是建议通过查询串口数据包的方式判断模块是否已经启动好（可读取 EzUIH 模块启动界面显示后触发的消息数据包，或者通过发送读取指令获取当前显示模块的状态）。

4.2 用户 UART 接口操作时序

EzUIH 系列模块串行 UART 接口，支持时钟频率为 1000000bps~9600bps，上电后模块的 UART 接口的默认波特率可由用户在资源存储器中的配置文件定义（参见第 5 章的介绍）；UART 采用 1 个启始位，1 个停止位，8 个数据位的时序，时序图如下：



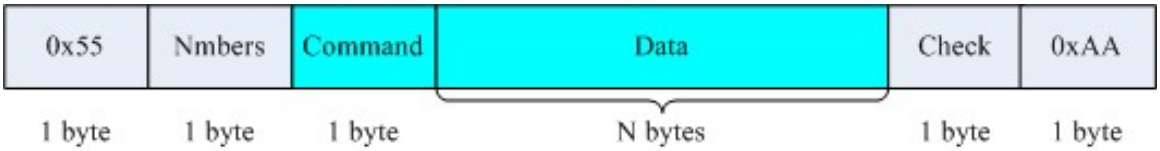
4.3 模块控制指令数据包构成

用户在通过串行接口给 EzUIH 系列模块发送控制指令、数据时，是以数据包的形式发送的，需要按

照要求来发送的；数据包的结构包含：帧头（0x55）、数据包数据个数、指令字节、指令数据（其字节数由具体的指令决定）、一个字节的和校验、帧尾（0xAA）。

指令数据包的总的字节数应小于 128 个字节。

数据包数据个数指的是在数据包当中的指令字节和指令数据的字节数的总和；而在数据包中的和校验，为指令字节和指令数据的每个字节的累加和值，但最终结果仅取低八位，即一个字节。



由上图可看出，数据包中 Numbers 即为数据包有效数据个数，即为上图中 N+1 的结果。

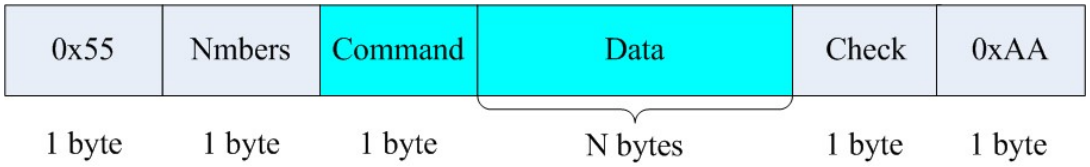
Command 为该数据包的指令标识，应按模块所支持的指令来赋予该字节有效的指令，它指明了该数据包在模块接收到之后，该执行何种操作。

数据包中，Data 的字节数由具体的指令决定，详情请参考后面对指令的介绍。

4.4 模块返回数据包构成

EzUIH 系列模块返回给用户 MCU 的数据包结构是固定的，与模块接收的数据包类似。

如下图所示：



EzUIH 系列模块返回数据包中，数据包中 Numbers 即为数据包有效数据个数，即为上图中 N+1 的结果，Command 为数据包的标识代码；数据区（Data）的含意请参考各返回数据包说明。

4.5 控制指令

EzUIH(S)系列模块“V10”版本的固件提供了多种控制指令，而针对功能的分类，又将这些控制指令分为了两类，分别为：基本显示控制指令、GUI 服务引擎接口指令。

4.5.1 基本显示控制指令

基本显示控制指令如下表介绍，数据解释针对的是该条指令在发送给 EzUIH 系列模块时，数据包中，指令数据的内容；数据个数也指的是指令数据的个数，而应与前面的数据包有效数据个数区分开，通常数据包有效个数为指令数据个数加 1。

指令码	功能	数据个数	数据解释
0x01	绘点	4	Data1：横向坐标（X 轴）位置高八位　Data2：横向坐标（X 轴）位置低八位 Data3：纵向坐标（Y 轴）位置高八位　Data4：纵向坐标（Y 轴）位置低八位
0x02	画直线	8	Data1：直线起点 X 轴坐标高八位　Data2：直线起点 X 轴坐标低八位 Data3：直线起点 Y 轴坐标高八位　Data4：直线起点 Y 轴坐标低八位

			Data5：直线终点 X 轴坐标高八位　Data6：直线终点 X 轴坐标低八位 Data7：直线终点 Y 轴坐标高八位　Data8：直线终点 Y 轴坐标低八位
0x03	画矩形框	8	Data1：左上角 X 轴坐标高八位　Data2：左上角 X 轴坐标低八位 Data3：左上角 Y 轴坐标高八位　Data4：左上角 Y 轴坐标低八位 Data5：右下角 X 轴坐标高八位　Data6：右下角 X 轴坐标低八位 Data7：右下角 Y 轴坐标高八位　Data8：右下角 Y 轴坐标低八位
0x04	画实心矩形	8	Data1：左上角 X 轴坐标高八位　Data2：左上角 X 轴坐标低八位 Data3：左上角 Y 轴坐标高八位　Data4：左上角 Y 轴坐标低八位 Data5：右下角 X 轴坐标高八位　Data6：右下角 X 轴坐标低八位 Data7：右下角 Y 轴坐标高八位　Data8：右下角 Y 轴坐标低八位
0x05	画圆框/圆弧	6/10	Data1：圆心 X 轴坐标高八位　Data2：圆心 X 轴坐标低八位 Data3：圆心 Y 轴坐标高八位　Data4：圆心 Y 轴坐标低八位 Data5：圆半径高八位　Data6：圆半径低八位 Data7(可选)：圆弧起始角度（0~3600）高八位 Data8(可选)：圆弧起始角度（0~3600）低八位 Data9(可选)：圆弧结束角度（0~3600）高八位 Data10(可选)：圆弧结束角度（0~3600）低八位 当指令数据包中无 Data7~Data10 数据时，表示绘制封密圆框。
0x06	画实心圆/缺口圆	6/10	Data1：圆心 X 轴坐标高八位　Data2：圆心 X 轴坐标低八位 Data3：圆心 Y 轴坐标高八位　Data4：圆心 Y 轴坐标低八位 Data5：圆半径高八位　Data6：圆半径低八位 Data7(可选)：缺口圆起始角度（0~3600）高八位 Data8(可选)：缺口圆起始角度（0~3600）低八位 Data9(可选)：缺口圆结束角度（0~3600）高八位 Data10(可选)：缺口圆结束角度（0~3600）低八位 当指令数据包中无 Data7~Data10 数据时，表示绘制完整实心圆形。
0x07	显示 ASCII 字符	5	Data1：字符左上角的 X 轴坐标高八位　Data2：字符左上角的 X 轴坐标低八位 Data3：字符左上角的 Y 轴坐标高八位　Data4：字符左上角的 Y 轴坐标低八位 Data5：ASCII 码
0x08	显示单个字符	5/6	Data1：字符左上角的 X 轴坐标高八位　Data2：字符左上角的 X 轴坐标低八位 Data3：字符左上角的 Y 轴坐标高八位　Data4：字符左上角的 Y 轴坐标低八位 Data5：汉字编码的高八位/ASCII 码　Data6（可选）：汉字编码的低八位/无 显示汉字字符时，需要 Data5 和 Data6 表示汉字的编码，而显示 ASCII 码字符时，只需要 Data5 即可；另，汉字的显示支持 GB2312 和 BIG5 字库。
0x09	图像显示指令	6	Data1：显示起始位置 X 轴坐标高八位　Data2：显示起始位置 X 轴坐标低八位 Data3：显示起始位置 Y 轴坐标高八位　Data4：显示起始位置 Y 轴坐标低八位 Data5：图像资源的序号高八位　Data6：图像资源的序号低八位 当图像资源序号为 0x8000 时，表示显示已发送到模块里的动态 JPEG 图片文件。
0x0A	直接数字显示	10	Data1：左上角 X 轴坐标高八位　Data2：左上角 X 轴坐标低八位 Data3：左上角 Y 轴坐标高八位　Data4：左上角 Y 轴坐标低八位 Data5~Data8：要显示的整型数据(32 位宽度)　Data9：显示十进制位数限制（2~8） Data10：显示属性（0、1 或 2，分别对应靠左、居中、靠右显示的属性）
0x0C	在指定位置开	>4	Data1：左上角 X 轴坐标高八位　Data2：左上角 X 轴坐标低八位 Data3：左上角 Y 轴坐标高八位　Data4：左上角 Y 轴坐标低八位

	始显示字符串	<=68	Data5~DataN：要显示的字符的编码数据 当显示的字符码值为 16 位长度时（如中文的 GBK 码（GB2312））按照先高字节后低字节的顺序安排，而 ASCII 字符只需要一个字节表示。
0x0D	在指定窗口显示图片资源局部图像指令	14	Data1：窗口起始位置 X 轴坐标高八位 Data2：窗口起始位置 X 轴坐标低八位 Data3：窗口起始位置 Y 轴坐标高八位 Data4：窗口起始位置 Y 轴坐标低八位 Data5：显示窗口的宽度高八位 Data6：显示窗口的宽度低八位 Data7：显示窗口的高度高八位 Data8：显示窗口的高度低八位 Data9：图片局部图像起始位置（相对图像左上角原点）X 轴坐标高八位 Data10：图片局部图像起始位置 X 轴坐标低八位 Data11：图片局部图像起始位置 Y 轴坐标高八位 Data12：图片局部图像起始位置 Y 轴坐标低八位 Data13：图片资源的序号高八位 Data14：图片资源的序号低八位 当图像资源序号为 0x8000 时，表示显示已发送到模块里的动态 JPEG 图片文件。
0x0F	区域取反显示	10	Data1：区域左上角 X 轴坐标高八位 Data2：区域左上角 X 轴坐标低八位 Data3：区域左上角 Y 轴坐标高八位 Data4：区域左上角 Y 轴坐标低八位 Data5：区域右下角 X 轴坐标高八位 Data6：区域右下角 X 轴坐标低八位 Data7：区域右下角 Y 轴坐标高八位 Data8：区域右下角 Y 轴坐标低八位 Data9：反色参考颜色高八位 Data10：反色参考颜色低八位
0x10	区域边框取反显示	10	Data1：区域左上角 X 轴坐标高八位 Data2：区域左上角 X 轴坐标低八位 Data3：区域左上角 Y 轴坐标高八位 Data4：区域左上角 Y 轴坐标低八位 Data5：区域右下角 X 轴坐标高八位 Data6：区域右下角 X 轴坐标低八位 Data7：区域右下角 Y 轴坐标高八位 Data8：区域右下角 Y 轴坐标低八位 Data9：取反显示边框的宽度
0x11	将指定区域内当前显示画面存入背景图层	8	Data1：区域左上角 X 轴坐标高八位 Data2：区域左上角 X 轴坐标低八位 Data3：区域左上角 Y 轴坐标高八位 Data4：区域左上角 Y 轴坐标低八位 Data5：区域右下角 X 轴坐标高八位 Data6：区域右下角 X 轴坐标低八位 Data7：区域右下角 Y 轴坐标高八位 Data8：区域右下角 Y 轴坐标低八位
0x30	启动动态 jpeg 文件传输	4	Data1~Data4：要传输的 jpeg 文件长度，高字节在前，低字节在后。 Jpeg 文件启动传输后，显示模块在完成指定长度的文件数据接收前，将不接收响应任何指令数据包，0x30 指令数据包发送后，模块如开辟内存成功，则通过返回数据包（指令码同为 0x30），数据包中 Data1 值为 1，如数据包 Data1 值为 0 则表示启动传输不成功。
0x80	清屏	1	Data1：清屏类型 Data1=0xAA 全屏清屏 Data1=0x55 全屏填充当前设置的绘图色
0x81	设置选择字库及字符色	4/6	Data1：要选择的字库序号高八位 Data2：要选择的字库序号低八位 Data3：字符色设置，高八位 Data4：字符色设置，低八位 Data5(可选)：设置使用系统字库（矢量字库），字库字号（像素点）值高八位 Data6(可选)：设置使用系统字库（矢量字库），字库字号（像素点）值高八位 该指令可设置选择点阵字库（包括 ASCII 字符或保存于资源文件中的中文字库）或系统字库（矢量字库），而当设置的目标为模块固件当中的内部 ASCII 字库时，需要将字库编号最高位置 1；而字符色的设置将会景响包括汉字显示在内的所有字符显示指令操作。**当选择使用系统字库时，Data5 Data6 表示设置的字号，其值与像素点对应，此时 Data1、Data2 的值应为 0x4000**。

0x84	设置绘图模式（线宽及绘图色）	3	Data0：设置绘图线宽（以像素点为单位） Data1：设置的绘图色高八位 Data2：设置的绘图色低八位 绘图线宽设置，最大线宽为 50，将影响绘直线、绘制矩形框等基本绘图操作。 绘图色的设置，将会影响绘点、直线、矩形等基本绘图操作，相当于设置前景色。
0x85	设置字符显示模式	3	Data1：bit0~bit3 字符覆盖模式（0、1、2 或 3） bit4~bit8 中西文字符混合显示模式设置（0—靠上对齐，1—居中对齐，2—靠下对齐） Data2：字符覆盖色高八位 Data3：字符覆盖色低八位 该指令仅对 ASCII 以及汉字字符显示指令起作用，不影响模块界面运行。
0x88	将当前显示屏画面存入背景图层	1	Data1：固定为 0xBB 该指令为整屏操作，将整屏显示内容转存入背景图层，背景图层的画面内容将可能会对图形界面的界面更新、控件更新产生影响，谨慎使用。
0x89	读取当前背光设置值	1	Data1：任意数据 该指令由用户控制器向模块发出后，模块收到指令数据包则会返回一个数据包给用户，数据包中包含两个字节的有效数据信息。
0x8A	背光控制指令	2	Data1：要设置的背光亮度值高八位 Data2：要设置的背光亮度值低八位 注：背光设置的范为 0~500，数值越大亮度越高。
0x8B	使能/禁止自动低功耗功能	1	Data1：0x5A 或 0xA5 值为 0x5A 表示使能，但模块自动低功耗功能的使能仍与模块 TFTcfg 中的定义有关，必需在 TFTcfg.txt 中的配置为有效使能才可真正使能；值为 0xA5 时表示关闭该功能。
0x8C	设置模块低功耗模式	1	Data1：控制参数 参数为 0xA5 时，模块将进入第一级低功耗模式，此时背光将关闭，模块可由用户设置新的背光亮度唤醒至正常工作模式；而参数为 0x5A 时，模块将进入第二极低功耗模式，此时只能由用户再次发送“0x8C”的指令唤醒，指令控制参数为 0xA5 和 0x5A 之外的其它数据。
0xD0	设置 RTC 时间	3	Data1：小时（24 小时制） Data2：分 Data3：秒
0xD1	读取 RTC 时间	1	Data1：任意数据 该指令由用户控制器向模块发出后，模块收到指令数据包则会返回一个数据包给用户，数据包中包含有小时、分钟、秒的信息。
0xD2	设置 RTC 日期	4	Data1：年 高字节数据 Data2：年 低字节数据 Data3：月 （1~12） Data3：日 年份的设置要求数据大于等于 2000，也即允许设置的是 2000 年之后的年份。
0xD3	读取 RTC 日期	1	Data1：任意数据 该指令由用户控制器向模块发出后，模块收到指令数据包则会返回一个数据包给用户，数据包中包含有年、月、日、星期的信息。
0xF8	锁定模块 USB 大容量存储器功能	11	Data1：’U’字符 ASCII 值 Data2：’O’字符 ASCII 值 Data3：’F’字符 ASCII 值 Data4~Data10：共 8 字节锁定密码数据 USB 锁定后，模块将无法进入 U 盘模式；触除锁定需 0xF9 指令，且指令所含密码数据与之前锁定时给的密码数值相同。
0xF9	解除模块 USB 大容量存储器	11	Data1：’U’字符 ASCII 值 Data2：’O’字符 ASCII 值 Data3：’N’字符 ASCII 值 Data4~Data10：共 8 字节解锁密码数据

	功能锁定		USB 锁定后，模块将无法进入 U 盘模式；触除锁定指令所含密码数据与之前锁定时给定的密码数值相同，模块才可正常解除锁定状态。

4.5.2 GUI 服务引擎接口指令

控制指令	功能	数据个数	数据解释
0xE0	控件数值/状态读取指令	2	Data1：要读取的控件 ID 号的高八位字节 Data2：要读取的控件 ID 号的低八位字节 注： 该指令将会触发模块返回数据包，数据包的数据个数将由具体要读取的控件类型及当前其数据状态决定。
0xE5	控件数值/状态写入指令	>=3	Data1：要写入的控件 ID 号的高八位字节 Data2：要写入的控件 ID 号的低八位字节 Data3~DataN：要写入的数据 注：对控件写入数值/状态，需根据控件的类型及属性来进行设置，数据包中有效数据的字节个数与控件的类型及属性相关，但需要了解的是，凡是遇到大于或等于两个字节表示的数据，均将高字节的数据放置于前，低位字节数据于后。
0xE6	控件控制指令	>=3	Data1：要写入的控件 ID 号的高八位字节 Data2：要写入的控件 ID 号的低八位字节 Data3~DataN：要写入的数据 注：对控件进行控制时使用该指令，但并非所有的控件均可由指令进行控制，具体请参考控件控制指令的介绍章节内容。
0xE8	界面切换指令	2	Data1：目标界面索引号的高八位字节 Data2：目标界面索引号的低八位字节
0xE9	读取当前活动界面索引号	1	Data1：无意义 注：该指令将触发模块返回当前处于活动状态的界面索引号。
0xEA	按键消息指令	2	Data1：键值（详见第 3 章的表格） Data2：按键事件类型（0:普通事件 1:按键释放 2 按键按下）
0xEB	模块 GUI 服务引擎消息返回数据包	7	Data1：返回消息的控件 ID 号的高八位字节 Data2：反回消息的控件 ID 号的低八位字节 Data3：返回消息的控件类型 Data4：返回消息参数 1 Data5：返回消息参数 2 Data6：返回消息参数 3 Data7：返回消息参数 4 注：当返回消息的控件类型为 0xff 时，表示返回消息不属于任何控件，表示该消息数据包为界面切换返回消息； 注：返回消息的数据包中的具体参数的定义与控件类型相关。
0xEC	设置选择控件	2	Data1：要选择的控件 ID 号的高八位字节 Data2：要选择的控件 ID 号的低八位字节 注：ID 号为 0xffff 时，表示取消当前控件选择。
0xC1	数字键盘使能	0	该指令将使能模块的数字键盘输入模式。
0xC2	数字键盘禁止	0	该指令将禁止模块的数字键盘输入模式。

4.6 基本显示控制指令详述

4.6.1 绘图前景色设置

EzUIH 系列模块有一个绘图色的概念，而模块所支持的基本绘图指令，如绘点、绘直线、绘矩形框、绘矩形、绘圆形框、绘圆形等，都是要以当前模块设置的绘图色来绘制的；而 EzUIH 系列模块的前景色需

要通过 0x84 的指令来设置，该指令需要一个 16 位长度的颜色参数，不同的数值代表不同的颜色。

颜色的数据与 TFT 屏上像素点的颜色数据类似，16 位长度的颜色数据排列着三基色的指数，即 R、G、B（红、绿、蓝）；分别占用 5、6、5 个二进制位。如下图所示：

R					G						B				
D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0

比如，绘图色设置的数值为 0xF800 时，为红色；值为 0x07E0 时，为绿色；而黑色为 0x0000，白色为 0xFFFF。

绘图色的设置指令/操作含有另一参数，为绘图线宽设置，该参数最大值为 50，最小值为 1，表示绘图操作时使用的线宽，该参数决定绘制直线、绘制矩形框的操作。

4.6.2 圆形/圆弧绘制

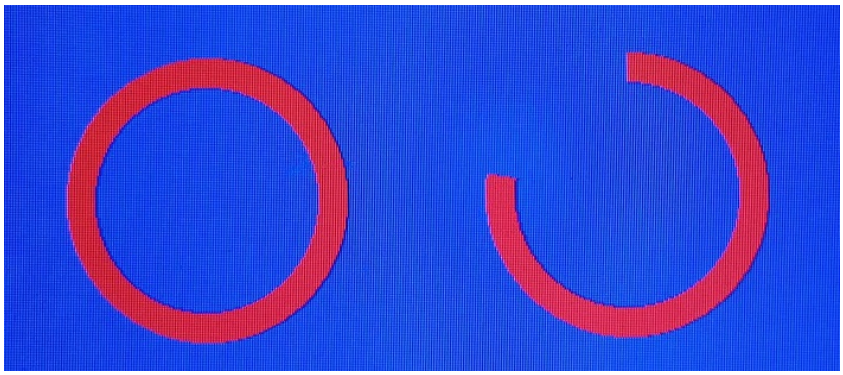
EzUIH 模块使用 0x05 指令进行圆形/圆弧绘制，当绘制圆形时，指令数据包结构如下图：

0x55	7	0x05	Data						Check	0xAA
			圆心X坐标		圆心Y坐标		圆半径			
			高字节	低字节	高字节	低字节	高字节	低字节		

当要绘制圆弧时，0x05 指令则需要多加四个字节的参数，以表明圆弧的起点及终点角度值（角度值范围：0~3600，表示 360 度），此时指令的结构如下图：

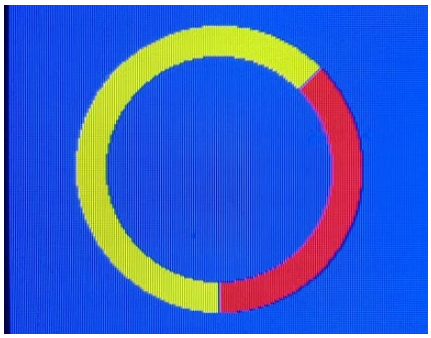
0x55	11	0x05	Data								Check	0xAA
			圆心X坐标		圆心Y坐标		圆半径		圆弧起点角度		圆弧终点角度			
			高字节	低字节	高字节	低字节	高字节	低字节	高字节	低字节	高字节	低字节		

假设当前设置的绘图色为红色，绘图线宽为 15 像素点，在显示屏 100、100 的位置为圆心，绘制半径为 60 的圆形，指令数据包内容为：0x55 0x07 0x05 0x00 0x64 0x00 0x64 0x00 0x3C 0x09 0xAA，显示的效果如图所示左侧圆形：



上图中右侧，使用 15 线宽，红色绘图色，绘制半径 60、起点弧度值 0、终点弧度值 2800 的圆弧，指令的数据：0x55 0x0B 0x05 0x01 0x2C 0x00 0x64 0x00 0x3C 0x00 0x00 0x0A 0xF0 0xCC 0xAA。

下图为红色部分使用 15 线宽，红色绘图色，半径 60，起点弧度值 450，终点弧度值 1800；黄色部分与红色圆心、半径相同，绘图线宽 15，黄色绘图色，起点弧度值 1800，终点弧度值 450。



4.6.3 字库及字符色设置

EzUIH 模块在设置字库时，可选择使用系统矢量字库或点阵字库，两者选其一；当使用矢量字库时，设置指令需指明所设置的字号大小；字号大小与像素点尺寸相关联，设置指令中使用两个字节表示字号大小。

当使用点阵字库时，则用户可选择模块在固件中预置有几种（视具体的模块而定）字号的 ASCII 字库，或选择加载在资源文件中的 ASCII 西文字库资源及中文字库（GB2312）；EzUIH 系列模块当前使用的字库将 ASCII 西文字库和中文字库分开管理，也即显示 ASCII 字符时使用的是所设置的 ASCII 字库，而显示中文字符时，使用的是所设置的中文字库；但两种字库使用的字符色统一管理，也即当前只能有一种字符色在作用。

EzUIH 系列模块的字库及字符色设置可通过 0x81 指令进行设置选择；指令内容包含有三部份或两部份：点阵字库编号两字节、字符颜色两字节、矢量字库字号两字节（该部分为指令可选内容）。

点阵字库的编号以 16 位长宽的数据表示，当使用模块内置的 ASCII 字为时，需将编号的最高位置 1，用以标明该字库为模块内置西文字库；如要使用内置的第 1 号 ASCII 西文字库时，设置时需指明该字库索引号为“0x8001”。而加载在资源文件中的 ASCII 字库以及中文字库，将以资源列表中的索引号为准。

字符色为一个 16 位长度（RGB565）的颜色数据，与绘图色的数据结构类似；设置字符色后，所有的 ASCII 字符显示、汉字字符显示都将使用该字符色，当然是在下一条设置字符色的指令执行之前。

矢量字库字号使用 16 位长度的两字节进行表示，与字符显示的实际像素点大小相对应，该部份为 0x81 指令的可选内容。

0x55	5/7	0x81	Data						Check	0xAA
			字库编号		字符色		矢量字库字号			
			高字节	低字节	高字节	低字节	高字节	低字节		

示例：设置使用内置 2 号西文字库，颜色为红色，指令数据包为：

0x55 0x05 0x81 0x80 0x02 0xF8 0x00 0x**FB** 0xAA　(0x81+0x80+0x02+0xF8+0x00=0x1**FB**)

示例：设置使用资源文件中已加载的序号为 5 的字库资源项（字库资源项可为 ASCII 西文字库或 GB2312 的中文字库），字符色使用黄色，指令数据包为：

0x55 0x05 0x81 0x00 0x05 0xFF 0xE0 0x65 0xAA

示例：设置使用系统矢量字库，字号为 50，字符色为蓝色，指令数据包为：

0x55 0x07 0x81 0x40 0x00 0x00 0x1F 0x00 0x32 0x12 0xAA　（使用矢量字库时，指令数据包中字库编码固定给值 0x4000）

4.6.4 字符覆盖模式设置

EzUIH 系列模块提供了一个字符覆盖模式的设置，该设置/操作提供了字符显示与原屏上显示的背景叠加关系的设置；设置/操作需要有 3 个字节的数据，分别是 1 个字节的覆盖模式设置，以及 2 个字节（16 位长度）的字符覆盖色设置；覆盖模式，可以为 0、1、2 或 3，字符覆盖色为覆盖模式下字符显示的背景擦除时使用的颜色（该颜色的数据结构与前景色的数据结构类似，RGB565 的色彩数据）。

字符覆盖模式为 0 时表示在显示字符时，对原字符显示位置区域的背景图像不作擦除，仅显示该字符的字符线条（字符线条的颜色由 0x81 指令设置）；而当字符覆盖模式设置为 1 时，在显示字符时将会使用指令设置的覆盖模式背景擦除色对该字符的显示区域进行擦除后才以当前设置的字符色来显示字符线条。

如下图所示，所显示的 ASCII 码字符选择为固件自带 ASCII 字符库，序号为 3；字符色为 0xFFE0 时，而字符覆盖模式设置为 0 时的显示‘A’和‘a’；而将字符覆盖模式设置为 1，字符覆盖背景擦除色为 0xFFFF 时，显示字符串：“16*32”。

0x85 为字符覆盖模式指令码，该指令设置后影响所有的有关字符显示的控制指令（不含控件）。

4.6.5 图像显示指令

EzUIH 系列模块提供了图像显示指令（0x09）

图像显示指令需要在指令数据中指明要显示的图像在资源存储器中的资源序号（用户通过串口传输的 JPEG 图像文件使用索引号 0x8000），图像显示指令需要指定显示该图像的左上角的起始坐标值，如果起始坐标值分别加上位图的宽和长的值，超出了屏幕的范围，显示将会不正常。

假设将下图放置在资源存储器的序号为 2 的资源中。

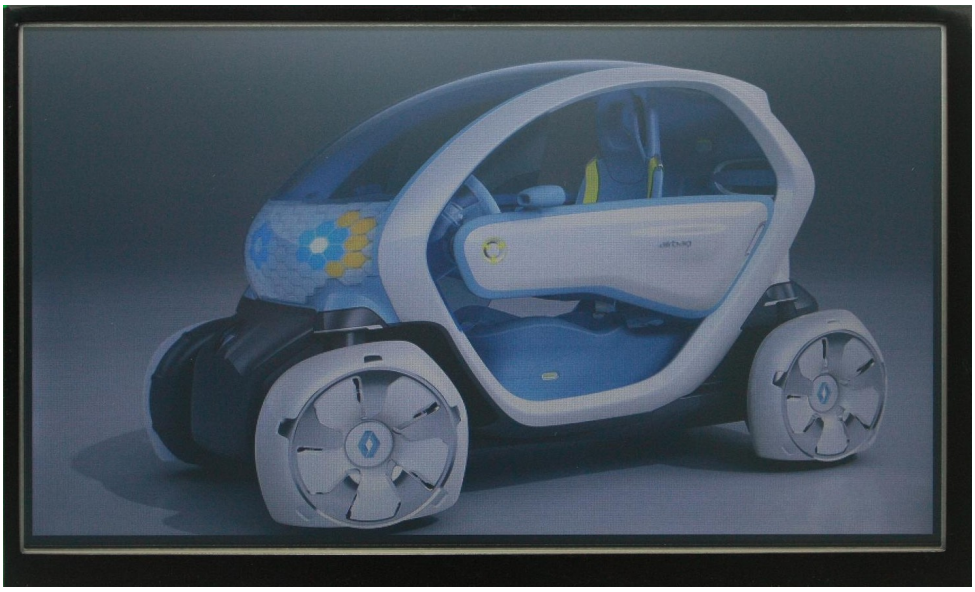
序号	类型	尺寸	偏移地址	资源大
0	汉字库	28*28	8192	915936
1	ASCII字库	24*48	925696	36864
2	真彩图	800*480	962560	768000
3	256色图	800*480	1732608	384512
4	16色图	800*480	2117632	192032
5	单色图	800*480	2310144	48004

用 0x09 的位图显示指令来显示该图时，将会在模块的屏幕上显示相应的效果。

下图是 0x09 指令的数据包示意：

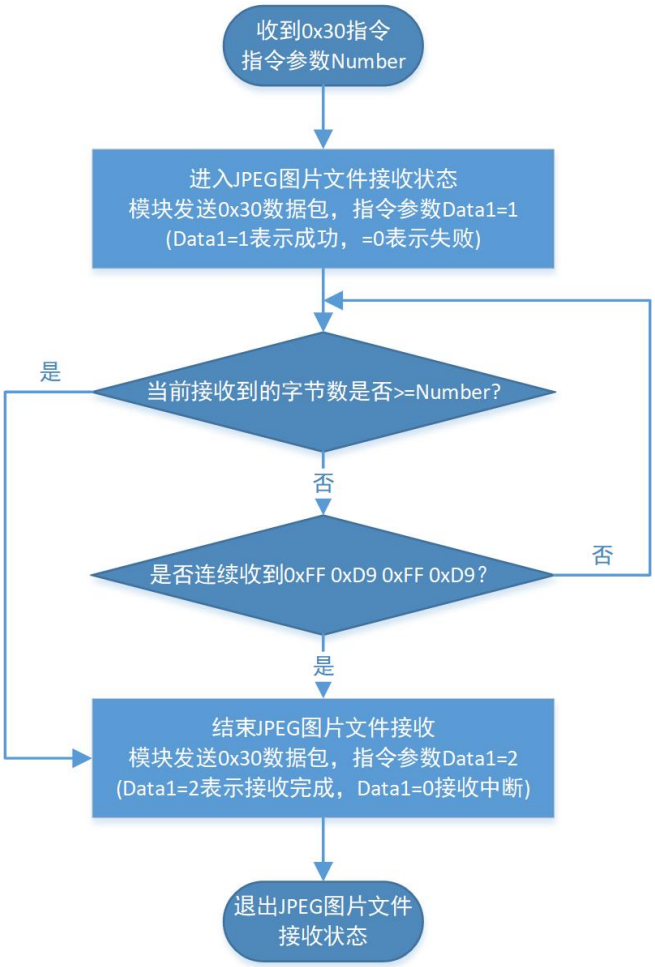
0x55	0x07	0x09	Data						0x0B	0xAA
			0x00	0x00	0x00	0x00	0x00	0x02		

显示的效果如下图所示：



4.6.6 动态 JPEG 图片接收

EzUIH 模块允许用户通过 0x30 指令向模块发送 1 个 JPEG 文件用于显示，0x30 指令较为特殊，用户需在指令数据中指明要传输的 JPEG 文件大小（字节数），当成功执行该指令后，模块将进入 JPEG 文件接收状态，此时模块在完成接收文件或中断接收之前不会接收其它的指令数据。下图为 0x30 指令执行的流程示意：



模块收到 0x30 指令后，将在模块内部开辟一个存储空间用于将要传输的 JPEG 图片文件的临时存储，如正常开辟空间，也即模块准备好了接收 JPEG 图片文件，则会通过串口发送数据包“0x55 0x02 0x30 0x01 0x31 0xAA”，如模块未能准备好空间，则表示模块未能进入 JPEG 图片文件的接收状态，此时，模块将发送消息“0x55 0x02 0x30 0x00 0x30 0xAA”。

模块收到 0x30 指令，并可准备好存储所用的空间，则模块会进入准备 JPEG 图片文件接收状态，在模块接收完指定字节数的数据或用户中断 JPEG 图片文件发送之前，模块将不会接收其它的显示操作指令（用户也不要在此期间发送除了 JPEG 图片文件外之的数据、指令）。

模块处于 JPEG 图片文件接收状态时，如果用户想提前结束当前的发送，则只需连续发送四个字节数据即可，四个字节的数据使用十六进制表示为“0xFF 0xD9 0xFF 0xD9”（0xFF 0xD9 一般为 JPEG 图片文件的结束标识）。

如模块正常接收完所指定字节数量的 JPEG 文件数据，则会在接收完成时，通过串口向用户返回数据包：“0x55 0x02 0x30 0x02 0x32 0xAA”；如接收过程中，用户发起中断传输，则模块返回数据包：“0x55 0x02 0x30 0x00 0x30 0xAA”。

4.6.7 低功耗模式设置

EzUIH 系列模块提供了两级的低功耗模式，用户可通过 0x8C 的控制指令使模块进入低功耗模式，该指令的数据包要求为：

0x55	0x02	0x8C	Data	0xE6	0xAA
			0x5A		

0x8C 指令的数据包中，Data 的数据的含意不同时，表示不同的设置指令：

- Data=0xA5：控制模块进入第一级低功耗模式，即等待模式；
- Data=0x5A：控制模块进入第二级低功耗模式，即待机模式；
- Data=其它值：控制模块恢复到正常工作模式。

EzUIH 系列模块在不同工作模式下时，将允许用户进行不同的操作，下面对它们进行详述：

- 第一级低功耗模式：该模式下，模块背光关闭，此时如用户重新设置背光亮度，或在触摸屏上进行有效的触摸事件，则模块退出该模式，恢复到正常工作模式，背光会恢复点亮；
- 第二级低功耗模式：在该模式下，模块背光关闭，此时用户只能通过低功耗模式设置指令，并将指令参数置为 0xA5 和 0x5A 之外的数据，然后模块会恢复到正常工作模式，背光会恢复点亮。

4.6.8 读取模块当前背光值返回数据包

用户控制器向模块发送 0x89 指令的数据包后，模块会返回一个数据包，数据包的结构如下图所示：

0x55	3	0x89	Data		Check	0xAA
			背光数值			
			高字节	低字节		

4.6.9 读取模块 RTC 时间返回数据包

用户控制器向模块发送 0xD1 指令的数据包后，模块会返回一个数据包，数据包的结构如下图所示：

0x55	4	0xD1	Data			Check	0xAA
			时	分	秒		
			1byte	1byte	1byte		

4.6.10 读取模块 RTC 日期返回数据包

用户控制器向模块发送 0xD3 指令的数据包后，模块会返回一个数据包，数据包的结构如下图所示：

0x55	6	0xD3	Data				Check	0xAA
			年		月	日	星期	
			高字节	低字节	1 byte	1 byte	1 byte	

注：星期的表示数值 0 为星期日，1 为星期一，2 为星期二。

4.6.11 锁定/解锁模块 USB 大容量存储器模式

为保护用户的资源文件，EzUIH 模块提供了锁定模块 U 盘功能的指令，指令码为 0xF8，指令需 11 个字节数据，前三字节固定为字符“UOF”的 ASCII 码值(0x55 0x4F 0x46)，后 8 个字节为锁定密码。

模块正角执行锁定指令后，用户将无法通过 USB 接口连接到模块 U 盘；当用户需要解锁模块 U 盘

功能时，则需要通过发送 0xF9 指令，并在指令数据包中给定与之前锁定相同的密码值，才可正确解锁。

示例：锁定模块 U 盘功能，使用密码为字符串“12345678”，则指令数据包为：

0x55 0x0C 0xF8 0x55 0x4F 0x46 0x31 0x32 0x33 0x34 0x35 0x36 0x37 0x38 0x86 0xAA

示例：用户使用密码字符串“12345678”进行模块 U 盘功能解锁：

0x55 0x0C 0xF9 0x55 0x4F 0x4E 0x31 0x32 0x33 0x34 0x35 0x36 0x37 0x38 0x8F 0xAA

模块如接收到正确的锁定/解锁 USB 大容量存储器模式指令的数据包内容，将会在显示屏当中将执行结果进行显示，以帮助用户判断当前处理结果。

4.7 GUI 服务引擎接口指令详述

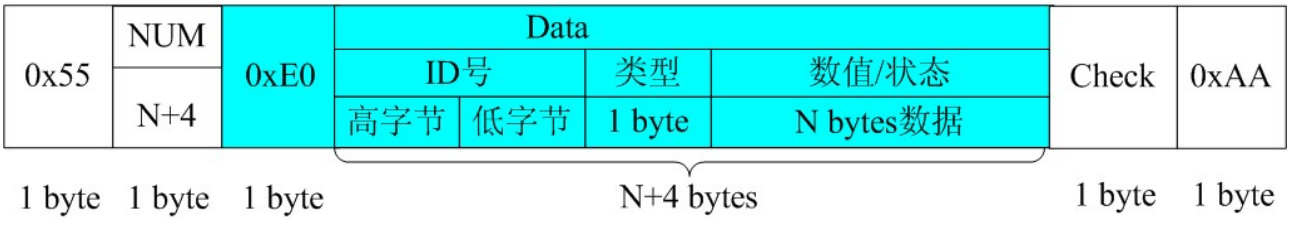
在对 GUI 服务引擎接口指令进行说明之前，需要用户了解 EzUIH 系列模块当中各类控件的类型说明，每种控件都有其指令的类型编号，如下表所述：

类型编码	控件类型说明
0x10	进度条控件(Process_Ctrl)
0x11	波形控件(WaveForm_Ctrl)
0x12	位图动画控件(Gif_Ctrl)
0x13	时间显示控件(TimeDisCtrl)
0x14	日期显示控件(DateDisCtrl)
0x15	表盘显示控件
0x16	滑动条控件
0x17	二维码显示控件
0x20	区域按钮(原触摸区域)控件(TouchArae)
0x21	位图按钮(原位图触摸按钮)控件(BitmapButton)
0x22	数值控件(Number_Edit)
0x23	字符串控件(String_Edit)
0x25	下拉选择控件(Combox_Ctrl)
0xff	GUI 服务引擎界面切换消息标识
0xfe	GUI 服务引擎触摸唤醒消息标识

4.7.1 控件数值/状态读取指令

该指令由用户发送读取指令（0xE0），指令参数中包含有要读取的控件的 ID 号（16 位的数据表示），模块会根据控件的类型返回数据给用户。

模块返回给用户 MCU 的数据包格式与前面介绍的用户 MCU 发送给 EzUIH 系列模块的数据包一样。而数据包的指令标识是 0xE0，数据包的格式如下图所示：



数据包中，0x55 为帧头，0xAA 为帧尾标识，而 NUM 为该数据包中包含数据包指令标识（0xE0）和有效数据区的数据个数，ID 号占用两个字节，类型编码占用一个字节，而控件的数值/状态的字节个数由控件的类型而定；Check 为图中淡青色区域的数据的累加和。

下面将以每种可读取数值/状态的控件作为对像，进行介绍用户对其进行数值/状态读取时，返回的数据包的具体数据结构：

进度条控件：

进度条控件的数值读取指令将会返回除 ID 号和控件类型之外的 4 个字节数据，包含有进度条控件的当前数值（或称为进度值）和进度条控件设置的数值最大值，这两个数据都使用 16 位长宽的数据表示，高字节在前，低字节在后的排列顺序。

当用户发送进度条控件数值读取指令后，模块将回送下图所示的数据包给用户：

0x55	8	0xE0	Data							Check	0xAA
			ID号		类型	当前进度值		最大进度值			
			高字节	低字节	0x10	高字节	低字节	高字节	低字节		

位图动画控件：

位图动画控件的状态读取指令将会返回除 ID 号和控件类型之外的 1 个字节数据，该字节数据标明控件当前的状态。

当用户发送位图动画控件状态读取指令后，模块将回送下图所示的数据包给用户：

0x55	5	0xE0	Data				Check	0xAA
			ID号		类型	状态		
			高字节	低字节	0x12	Status(1 byte)		

当位图动画控件的属性设置为自动动画显示类型时，状态返回 0 表示控件处于停止状态，状态返回 1 时表示控件处于动画显示状态；而当控件属性设置为静动态显示类型时，状态返回 0 表示控件处于静态显示状态，状态返回 1 时表示控件处于动态显示状态。

而当控件属性设置为完全指令切换类型时，控件所返回的状态将指示控件当前设置显示的位图的序号，序号的大小由“起始位图索引”至“结束位图索引”之间的位图个数决定。

位图按钮控件：

只有当位图按钮控件的属性类设置为乒乓开关时，位图按钮控件才可由用户发送控件数值/状态读取指令并返回控件当前的状态；所返回的数据除 ID 号和控件类型之外仅 1 字节的状态数据。

返回的数据包如下所示：

0x55	5	0xE0	Data				Check	0xAA
			ID号		类型	状态		
			高字节	低字节	0x21	Status(1 byte)		

位图按钮控件设置为乒乓开关时，控件的状态仅有两个状态，0 或 0xff。

数值控件：

数值控件的数值读取指令将会返回除 ID 号和控件类型之外的 4 个字节数据，这 4 个字节的数据按高字节在前低字节在后的顺序排列发送，其根据数值控件的类型设置而不同，如控件设置为无符整型数，则这 4 个字节的数据则按 unsigned int 型的数据进行组合；如控件设置为有符整型数，则按 signed int 型数据进行组合；而如控件设置为浮点数，则按 float 型的数据进行组合。

当用户发送位图动画控件状态读取指令后，模块将回送下图所示的数据包给用户：

0x55	8	0xE0	Data							Check	0xAA
			ID号		类型	当前数值					
			高字节	低字节	0x22	31~24bit	23~16bit	15~8bit	7~0bit		

用户在读取数值控件的数值时，应按数值控件的类型进行数据的处理，否则，将会获取到错误的结果。

字符串控件：

字符串控件的数值读取指令将会返回除控件 ID 号及控件类型之外的字符串数据，数据中第一个字节为当前字符串控件的字符数据个数（字节数），后续则是当前字符串控件的字符串数据。所返回的数据包结构如下所示：

0x55	N+5	0xE0	Data					Check	0xAA
			ID号		类型	字节数	字符串数据		
			高字节	低字节	0x23	N	N bytes		

下拉选择控件：

下拉选择控件的数值读取指令将返回除 ID 号和控件类型之外的 1 个字节数据，该字节数据指示控件当前的被选项索引号；所返回的数据包结构如下所示：

0x55	5	0xE0	Data				Check	0xAA
			ID号		类型	当前选项索引		
			高字节	低字节	0x25	Select(1 byte)		

时间显示控件：

时间显示控件可以将当前控件所显示的时间返回给用户控制器，返回的数据包格式如下图所示：

0x55	7	0xE0	Data						Check	0xAA
			ID号		类型	时	分	秒		
			高字节	低字节	0x13	1 byte	1 byte	1 byte		

注：如控件的类型设置为显示系统 RTC 时间，并且模块且有 RTC 实时时钟功能的，则读取时间显示控件的数据，可以获取到当前模块的 RTC 系统时间。

日期显示控件：

日期显示控件可以将当前控件所显示的日期信息返回给用户控制器，返回数据包格式如下图所示：

0x55	9	0xE0	Data								Check	0xAA
			ID号		类型	年		月	日	星期		
			高字节	低字节	0x14	高字节	低字节	1 byte	1 byte	1 byte		

注：如控件的类型设置为显示系统 RTC 日期，并且模块且有 RTC 实时时钟功能的，则读取日期显示控件的数据，可以获取到当前模块的 RTC 系统日期。

表盘显示控件:

表盘显示控件可以将当前控件的量化数值返回给用户控制器，当前量化数值使用两个字节表示，返回的数据包结构如下图所示：

0x55	6	0xE0	Data					Check	0xAA
			ID号		类型	高8位	低8位		
			高字节	低字节	0x15	1 byte	1 byte		

滑动条控件：

滑动条控件的数值读取指令将会返回除 ID 号和控件类型之外的 2 个字节数据，2 个字节的数据为当前量化数值，高字节在前，低字节在后的排列顺序。返回的数据包结构如下图所示：

0x55	6	0xE0	Data					Check	0xAA
			ID号		类型	当前数值			
			高字节	低字节	0x16	高字节	低字节		

二维码显示控件：

二维码显示控件的数值读取指令将会返回除控件 ID 号及控件类型之外的字符串数据，数据包中紧跟着控件类型字节之后的内容是当前控件的字符串数据。所返回的数据包结构如下所示：

0x55	N+4	0xE0	Data				Check	0xAA
			ID号		类型	字符串数据		
			高字节	低字节	0x17	N bytes		

注：设置为共享数据模式的二维码显示控件无法通过 0xE0 指令对控件字符串内容进行读取。

4.7.2 控件数值/状态写入指令

EzUIH 系列模块的 GUI 服务引擎允许用户通过指令码为 0xE5 的数据包来对控件进行数值/状态写入，0xE5 指令的数据包中要求用户指定所要写入的控件的 ID 号，而对控件要写入的数据则要根据具体的控件类型而定，0xE5 指令的数据包结构如下所示：

0x55	N+3	0xE5	Data			Check	0xAA
			ID号		数据		
			高字节	低字节	N bytes		

接下来，将以每一种允许用户进行数值/状态写入的控件作为对像，详述 0xE5 指令的操作方式。

区域按钮控件：

区域按钮控件可由用户通过指令设置当前控件的字符串，控件在接受正确的设置指令后，将会自动根据控件的活动状态以及控件的属性，进行自动的显示刷新。区域按钮控件的数值写入指令数据包结构如下所示：

0x55	N+3	0xE5	Data			Check	0xAA
			ID号		字符串数据		
			高字节	低字节	N bytes		

进度条控件：

进度条控件的数值写入指令要求数据包中除 ID 号和控件类型之外的 2 个字节数据，用于设置进度条控件的当前数值（或称为进度值），进度条控件的当前数值使用 16 位长宽的数据表示，高字节在前，低字节在后的排列顺序。

对进度条控件的数值进行写入的数据包结构如下：

0x55	5	0xE5	Data				Check	0xAA
			ID号		进度值			
			高字节	低字节	高字节	低字节		

波形控件：

波形控件的数值写入指令即是对波形控件的波形线插入新数值的操作，波形控件波形线在插入新数值后，将会根据控件当前的活动情况（是否处于当前显示界面当中）以及控件的刷新类型设置来对控件内的波形线进行自动的刷新显示。波形控件允许最多在控件当中存在 4 条波形线，所以在对波形控件的波形线插入新数值的数值写入指令操作时，需要根据控件所允许的波形线条数来确定数据包中的有效数据个数；每个波形线的新插入数据使用 16 位长度的数据表示。

对波形控件的数值写入操作指令的数据包格式如下所示：

0x55	11	0xE5	Data								Check	0xAA
			ID号		波形线1		波形线2		波形线3		波形线4			
			高字节	低字节	高字节	低字节	高字节	低字节	高字节	低字节	高字节	低字节		

实际上，如果所写入的波形控件仅设置有 1 条波形线，则用户在对其进行数值写入时，可以只写入波形线 1 的数据即可，也即上图中，“波形线 2”、“波形线 3”、“波形线 4”的数据均可不需要。

位图动画控件：

位图动画控件的状态写入指令将会将可设置目标控件的状态，位图动画控件状态写入指令的数据包示意如下：

0x55	4	0xE5	Data			Check	0xAA
			ID号		状态		
			高字节	低字节	Status(1 byte)		

位图动画控件的状态在被用户修改后，控件将会根据自身的属性以及所设置的状态进行显示的更新。

位图按钮控件：

只有当位图按钮控件的属性类设置为乒乓开关时，位图按钮控件才可由用户发送控件数值/状态写入指令以设置控件当前的状态；设置的数据包如下所示：

0x55	4	0xE5	Data			Check	0xAA
			ID号		状态		
			高字节	低字节	Status(1 byte)		

数值控件：

数值控件当中的数值可由用户使用指令进行更新，所设置更新的数值使用 4 个字节数据进行表示，这 4 个字节的数据在数据包当中按高字节在前低字节在后的顺序排列发送，其根据数值控件的类型设置而不同，如控件设置为无符整型数，则这 4 个字节的数据则按 unsigned int 型的数据进行组合；如控件设置为有符整型数，则按 signed int 型数据进行组合；而如控件设置为浮点数，则按 float 型的数据进行组合。

数值控件数值写入指令数据包的结构如下：

0x55	7	0xE5	Data						Check	0xAA
			ID号		数值					
			高字节	低字节	31~24bit	23~16bit	15~8bit	7~0bit		

字符串控件：

字符串控件可由用户通过指令设置当前控件的字符串，控件在接受正确的设置指令后，将会自动根据控件的活动状态以及控件的属性，进行自动的显示刷新。字符串控件的数值写入指令数据包结构如下所示：

0x55	N+3	0xE5	Data			Check	0xAA
			ID号		字符串数据		
			高字节	低字节	N bytes		

下拉选择控件：

下拉选择控件可由用户使用指令设置控件当前所选项，控件在收到正确的设置指令后，会自动根据控件的活动状态以及控件的属性设置，自动进行显示的更新。对下拉选框控件的数值写入指令数据包结构如下所示：

0x55	4	0xE5	Data			Check	0xAA
			ID号		设置选择项		
			高字节	低字节	Select(1 byte)		

时间显示控件：

当控件的类型设置为“显示用户指定时间”时，控件可接收并显示用户发送所要显示的时间；对时间显示控件进行时间写入的数据包格式如下图所示：

0x55	6	0xE5	Data					Check	0xAA
			ID号		时	分	秒		
			高字节	低字节	1 byte	1 byte	1 byte		

日期显示控件：

当控件的类型设置为“显示用户指定日期”时，控件可接收并显示用户发送所要显示的日期；对日期显示控件进行日期写入的数据包格式如下图所示：

0x55	7	0xE5	Data						Check	0xAA
			ID号		年		月	日		
			高字节	低字节	高字节	低字节	1 byte	1 byte		

对日期显示控件的日期数据写入时，并不需要写入星期的信息，模块会在收到数据包后自动计算出指定日期的星期信息。

表盘显示控件：

表盘显示控件可由用户使用指令设置控件当前的量化数值，控件在收到正确的设置指令后，会自动根据控件的活动状态以及控件的属性设置，自动进行显示的更新。对表盘显示控件的数值写入指令数据包结构如下所示

0x55	5	0xE5	Data				Check	0xAA
			ID号		高8位	低8位		
			高字节	低字节	1 byte	1 byte		

滑动条控件：

滑动条控件的数值写入指令要求数据包中除 ID 号和控件类型之外的 2 个字节数据，用于设置滑动条控件的当前数值（或称为进度值），数值使用 16 位长宽的数据表示，高字节在前，低字节在后的排列顺序。对滑动条控件的数值进行写入的数据包结构如下：

0x55	5	0xE5	Data				Check	0xAA
			ID号		设置数值			
			高字节	低字节	高字节	低字节		

二维码显示控件：

二维码显示控件可由用户通过指令设置当前控件的字符串，控件在接受正确的设置指令后，将会自动根据控件的活动状态以及控件的属性，进行自动的显示刷新。二维码显示控件的数值写入指令数据包结构（与字符串控件的写入相同）如下所示：

0x55	N+3	0xE5	Data			Check	0xAA
			ID号		字符串数据		
			高字节	低字节	N bytes		

注：设置为共享数据模式的二维码显示控件无法通过 0xE5 指令对控件字符串内容进行写入。

4.7.3 控件控制指令

控件控制指令在目前版本的固件当中，可对所有的控件进行禁能、使能操作，并可对不同的控件进行不同的配置更改操作（控件配置更新后下次复位将不能保存更改后的配置）。

注：波形控件的属性设置当中，可设置控件为“用户指令控制重绘”；当控件设置为该项属性时，

控件在接收到数值设置指令后（即波形线进行了有效的新数据插入），控件并不会刷新波形线的显示，只有当控件接收到刷新指令（在此，即指控件控制指令）后才会对控件的波形线进行刷新显示。

控件控制指令的数据包要求如下所示：

0x55	N+3	0xE6	Data			Check	0xAA
			ID号		参数		
			高字节	低字节	至少1字节(N bytes)		

控件控制指令数据包中至少含有 1 个字节的参数，当该参数的第 1 个字节为 0~2 的数值时，对于所有的控件都一样：

- 参数为 0 时对于所有的控件类型，均表示禁能指定 ID 号的控件；
- 参数为 1 时均为使能并恢复显示指定 ID 号的控件；
- 参数为 2 时为对指定 ID 号的控件进行禁能并消除显示。

而参数的第 1 个字节大于数值 2 时，则参数的字节数将由具体的控件类型决定，下表为针对各个类型的控件进行控件控制时，数据包参数的要求：

目标控件类型	参数功能说明
进度条控件	Byte0=0 禁能 Byte0=1 使能 Byte0=2 禁能并消除显示 Byte0=0x11 设置进度条颜色，Byte1 为颜色高八位字节，Byte2 为颜色低八位字节
波形控件	Byte0=0 禁能 Byte0=1 使能 Byte0=2 禁能并消除显示 Byte0=3 重绘波形线 Byte0=4 清除波形线及数据
位图动画控件	Byte0=0 禁能 Byte0=1 使能 Byte0=2 禁能并消除显示
时间显示控件	Byte0=0 禁能 Byte0=1 使能 Byte0=2 禁能并消除显示 Byte0=0x11 设置控件字符颜色，Byte1 为颜色高八位字节，Byte2 为颜色低八位字节 Byte0=0x12 设置控件重绘描边线条颜色，Byte1 为颜色高八位字节，Byte2 为颜色低八位字节 Byte0=0x13 设置控件重绘区域填充颜色，Byte1 为颜色高八位字节，Byte2 为颜色低八位字节
日期显示控件	Byte0=0 禁能 Byte0=1 使能 Byte0=2 禁能并消除显示 Byte0=0x11 设置控件字符颜色，Byte1 为颜色高八位字节，Byte2 为颜色低八位字节 Byte0=0x12 设置控件重绘描边线条颜色，Byte1 为颜色高八位字节，Byte2 为颜色低八位字节 Byte0=0x13 设置控件重绘区域填充颜色，Byte1 为颜色高八位字节，Byte2 为颜色低八位字节
区域按钮	Byte0=0 禁能 Byte0=1 使能 Byte0=2 禁能并消除显示 Byte0=0x10 设置控件字符串，Byte1~ByteN 为字符串内容 Byte0=0x11 设置控件字符颜色，Byte1 为颜色高八位字节，Byte2 为颜色低八位字节 Byte0=0x12 设置控件重绘描边线条颜色，Byte1 为颜色高八位字节，Byte2 为颜色低八位字节 Byte0=0x13 设置控件重绘区域填充颜色，Byte1 为颜色高八位字节，Byte2 为颜色低八位字节
位图按钮	Byte0=0 禁能 Byte0=1 使能 Byte0=2 禁能并消除显示
数值控件	Byte0=0 禁能 Byte0=1 使能 Byte0=2 禁能并消除显示 Byte0=5 设置数值控件扩展显示位数 Byte1 为要设置的显示位数 Byte0=0x11 设置数值控件字符颜色，Byte1 为颜色高八位字节，Byte2 为颜色低八位字节 Byte0=0x12 设置控件重绘描边线条颜色，Byte1 为颜色高八位字节，Byte2 为颜色低八位字节 Byte0=0x13 设置控件重绘区域填充颜色，Byte1 为颜色高八位字节，Byte2 为颜色低八位字节
字符串控件	Byte0=0 禁能 Byte0=1 使能 Byte0=2 禁能并消除显示

	Byte0=0x11 设置字符串控件颜色，Byte1 为颜色高八位字节，Byte2 为颜色低八位字节 Byte0=0x12 设置控件重绘描边线条颜色，Byte1 为颜色高八位字节，Byte2 为颜色低八位字节 Byte0=0x13 设置控件重绘区域填充颜色，Byte1 为颜色高八位字节，Byte2 为颜色低八位字节
下拉选框控件	Byte0=0 禁能 Byte0=1 使能 Byte0=2 禁能并消除显示 Byte0=0x11 设置控件字符颜色，Byte1 为颜色高八位字节，Byte2 为颜色低八位字节 Byte0=0x12 设置控件重绘描边线条颜色，Byte1 为颜色高八位字节，Byte2 为颜色低八位字节 Byte0=0x13 设置控件重绘区域填充颜色，Byte1 为颜色高八位字节，Byte2 为颜色低八位字节 Byte0=0x20 设置控件备先项字符串，Byte1 为要设置的备选项索引号（如大于当前控件备选项数目，则为添加新的备选项）Byte2~ByteN 为字符串内容
表盘控件	Byte0=0 禁能 Byte0=1 使能 Byte0=2 禁能并消除显示
滑动条控件	Byte0=0 禁能 Byte0=1 使能 Byte0=2 禁能并消除显示
二维码显示控件	Byte0=0 禁能 Byte0=1 使能 Byte0=2 禁能并消除显示

4.7.4 界面切换指令

EzUIH 系列模块的 GUI 服务引擎允许用户通过界面切换指令来控制模块切换到指定的界面显示；界面切换指令的目标界面使用 16 位长度的数据表示，而界面切换指令的数据包结构如下所示：

0x55	3	0xE8	目标界面		Check	0xAA
			高字节	低字节		

4.7.5 读取当前活动界面索引号

用户可通过读取当前活动界面索引号的指令来获知当前模块所显示的界面索引号，该指令的数据包当中，要求有一个字节的参数，但其数值并没有要求，可设为任意值，发送读取指令的数据包结构如下：

0x55	2	0xE9	参数	Check	0xAA
			(1 byte)		

读取当前活动界面索引号的指令将会使模块返回一个数据包给用户，所返回的数据包结构如下所示：

0x55	3	0xE9	当前活动界面索引号		Check	0xAA
			高字节	低字节		

当前活动界面索引号使用 16 位长度的数据表示。

4.7.6 按键消息指令

EzUI 无触摸版本的模块需要用户发送按键消息来进行 GUI 服务引擎的工作，按键消息指令数据包当中含有按键键值、按键事件类型，如下表所示：

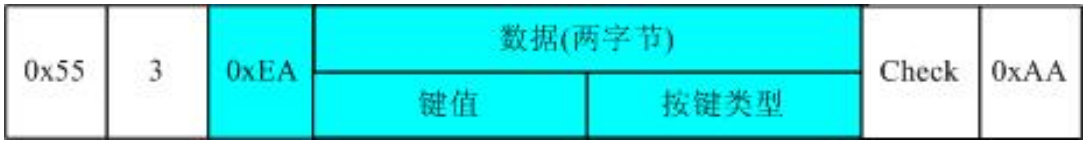
控制指令	功能	数据个数	数据解释
0xEA	按键消息指令	2	Data1：键值（详见第 3 章的表格） Data2：按键事件类型（0:普通事件 1:按键释放 2 按键按下）

键值的定义已在前面章节中进行说明，而按键事件类型较为特殊，仅有 GUI_SCANCODE_ENTR 确

定键支持该数据，其它的键值均无效，通常在发送确定键之外的按键消息时，按键事件类型均要求设置为时 0。

按键事件类型当中的“按键释放”和“按键按下”事件类型，实际上是为了实现一些用户所需要的长时按键功能，适用于按钮类的控件，这样就可以通过分别发送按下和释放事件的按键消息来实现控件的控制。

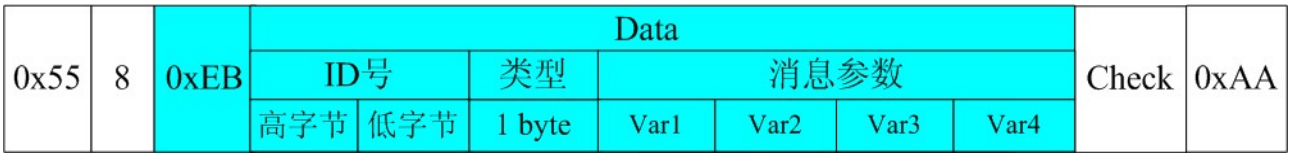
下图为按键消息的数据包结构：



4.7.7 模块 GUI 服务引擎消息返回数据包

GUI 服务引擎在控件响应事件时，如控件设置的消息发送使能勾选，将会自动返回消息给用户，以通知用户哪些控件发生了什么样的事件；此外，GUI 服务引擎还会将界面切换的事件消息返回给用户。

GUI 服务引擎的消息返回数据包的结构如下所示：



GUI 服务引擎消息返回数据包使用统一的格式，数据包中的 ID 号将标明该条消息是来源于哪一个 ID 号的控件，而类型字节则标明该控件属于哪一类控件类型（参考前面控件类型编码）；数据包中的消息参数共有 4 个字节，其含意因各控件的类型不同而不同，下面将在列表中介绍各类控件所触发返回的消息数据包。

消息来源	类型编码	消息参数说明			
		Var1	Var2	Var3	Var4
进度条控件	0x10	当进度条控件的进度值到达最大值时，则会返回消息数据包，数据包使用的消息参数为 Var1 和 Var2。 Var1：固定值为 0　　Var2：固定值为 0xff			
位图动画控件	0x12	当位图动画控件的属性设置当中，设置有效的动画循环次数时，如在当前界面中该控件已循环到达指定的次数，则会返回消息数据包，数据包中的消息参数均为 0。			
区域按钮控件	0x20	当区域按钮控件响应用户按键消息事件时，并且符合其所设置的消息响应机制，则会返回消息数据包，数据包使用消息参数为 Var1。 Var1：其值为 1 表示消息为控件释放　　其值为 2 表示消息为控件按下			
位图按钮控件 （非乒乓开关属性）	0x21	当位图按钮控件响应用户按键消息事件时，并且符合其所设置的消息响应机制，则会返回消息数据包，数据包使用消息参数为 Var1。 Var1：其值为 1 表示消息为控件释放　　其值为 2 表示消息为控件按下			
位图按钮控件 （乒乓开关属性）	0x21	当位图按钮控件属性为乒乓开关时，如控件响应用户按键消息事件的释放事件后，控件改变了自身的状态，则会返回消息数据包，数据包使用消息参数为 Var1 和 Var2。			

		Var1：其值应固定为 1　　Var2：其值表示乒乓开关的当前状态 0 或 0xff
数值控件	0x22	当数值控件的数值发生用户按键消息完成输入后，则在控件的数值发生改变时，会发送消息数据包，数据包使用消息参数 Var1、Var2、Var3 和 Var4。 Var1：数值的 31~24bit 字节数据　　Var2：数值的 23~16bit 字节数据 Var3：数值的 15~8bit 字节数据　　Var4：数值的 7~0bit 字节数据
下拉选择控件	0x25	当下拉选择控件发生用户按键消息输入后，在控件的当前选择发生了改变，则会发送消息数据包，数据包使用消息参数 Var1。 Var1：当前控件选择项索引号
GUI 服务引擎界面切换	0xff	当界面发生切换显示时，模块会发送消息数据包，以通知用户界面发生改变，数据包使用消息参数 Var1 和 Var2。 Var1：当前显示的界面索引号的高八位 Var2：当前显示的界面索引号的低八位
GUI 服务引擎触摸唤醒	0xfe	当模块处于第一级低功耗模式，即待机模式时，可由用户触摸模块的有效触摸区域，以唤醒模块恢复正常工作状态，此时模块会发送消息数据包，以通知用户发生了触摸唤醒事件，数据包并不使用消息参数。

4.7.8 波形控件一次写入多点数据

波形控件允许用户在一个数据包当中写入多条波形线的多点数据，这样可以大大提高波形线刷新显示的速度，数据包指令码仍使用 0xE5 的控件数值/状态写入指令，但要求数据包当中的有效数据个数大于 11 个字节，且要求整个数据包（包含帧头帧尾以及相关的结构字节数据）的字节个数小于 127 字节。

波形控件一次写入多点数据要求数据包当中所操作的波形线数量与实际控件所设置的波形线数量一至，否则将会发生错误的显示刷新。比如，某个波形控件设置了同时显示 2 条波形线，则使用数据包对该控件进行多点数据写入时，需要同时对控件中的 2 条波形线进行数据给定。

例如，某个波形控件设置了波形线为 1 条，需要同时给控件当中的波形线写入 N 个数据点，则数据包的结构如下：

0x55	M	0xE5	Data										Check	0xAA
			ID号		波形线1数据1		波形线1数据2		……		波形线1数据N			
			高字节	低字节	高字节	低字节	高字节	低字节			高字节	低字节		

数据包中 M 为数据包有效数据字节数，M=N*2+3。

例如某个波形控件设置的波形线为 2 条，则一次对控件写入 N 个点数据时，数据包结构如下：

0x55	M	0xE5	Data									
			ID号		波形线1数据1		波形线2数据1		波形线1数据2		波形线2数据2	
			高字节	低字节	高字节	低字节	高字节	低字节	高字节	低字节	高字节	低字节

Data					Check	0xAA
……	波形线1数据N		波形线2数据N			
	高字节	低字节	高字节	低字节		

数据包中 M 为数据包有效数据字节数，M=N*2*2+3；单控件多波形线的多点数据写入时，数据包中，波形线的数据要依顺序排好。

4.7.9 多个控件连续写入数值/状态

EzUIH 模块允许用户一次（一个数据包）对多个连续 ID 号的控件进行数值/状态写入，但对控件有如下要求：

- 一次写入的多个控件 ID 号建议为连续的；
- 一次写入的多个控件数据类型需相同，不允许字符串类型数据的控件与数值类的控件混合一次写入；
- 该指令不支持波形显示控件；
- 一次写入多个控件数据的数据包，需确保整个数据包内的有效字节个数小于 124 字节。

多个控件连续写入数据指令的指令码为 0xE1，数据包的格式要求如下图所示：

0x55	M	0xE1	Data				Check	0xAA
			类型	起始ID号		数据段		
			1 byte	高字节	低字节	N个字节		

M 为数据包内有效字节个数，M 应小于等于 124，M=N+4。

数据包当中，紧跟着指令码 0xE1 后面的一个字节，为数据类型，该字节数据的值应为 0 或 1、2、4。该字的定义决定了数据包中数据段的具体格式安排：

类型=0：数据段为字符串形式，针对字符串控件，数据段可安排多个字符串，写入每个控件的字符串之间，一字节的 0 值进行全隔；而字符串中如果有中文字符，则每个中文字符使用两个字节表示其 GB 码，高字节在前，低字节在后。

假设控 ID 号 101(十六进制表示为 0x0065)、102、103 的控件为字符串控件，分别要对这三个控件写入字符串：“Body101”、“Body102”、“Body103”的字符串时，指令数据包的内容如下：

0x55 0x1C 0xE1 0x00 0x00 0x65 0x42 0x6F 0x64 0x79 0x31 0x30 0x31 0x00 0x42 0x6F 0x64 0x79 0x31 0x30 0x32 0x00 0x42 0x6F 0x64 0x79 0x31 0x30 0x33 0x00 0xA9 0xAA

“Body101”字符串的每个字符的数据用十六进制表示为：0x42 0x6F 0x64 0x79 0x31 0x30 0x31

类型=1：数据段以字节为单位，每一个字节为对一个控件的数值写入内容。

假设控件 ID 号 101（十六进制表示为 0x0065）、102、103、104、105 的控件均为数值控件，要对这 5 个控件分别写入数值 20、30、40、50、60，则数据包的内容如下：

0x55 0x09 0xE1 0x01 0x00 0x65 0x14 0x1E 0x28 0x32 0x3C 0x0F 0xAA

类型=2：数据段以两个字节为单位向控件写入数值，每个数据高字节在前，低字节在后。

假设控件 ID 号 201（十六进制表示为 0x00C9）、202、203、204 的控件均为数值控件，要对这 4 个控件分别写入数值 1000、2000、3000、4000，则数据包的内容如下：

0x55 0x0C 0xE1 0x02 0x00 0xC9 0x03 0xE8 0x07 0xD0 0x0B 0xB8 0x0F 0xA0 0xE0 0xAA

类型=4：则数据段中，每个要写入控件的数据使用四个字节进行表示，同样也为高字节在前，低字节在后进行安排。而数据可为整型数也可为浮点数，在此不再进行单独的说明，与 0xE5 指令对数值控件的数据写入是一样的。

注：EzUI 系列模块的浮点数与单片机 C 语言中的浮点数变量定义方法相同，在此没有必要再进行详述，一般来说，可以在 C 语言中定义一个浮点数变量，然后通过指针获取到该变量的地址，从它的地址中提取出四个字节数据即可，或者写入。

4.7.10 动态创建区域按钮控件

用户 MCU 可通过发送指令码“0xBC”的指令数据包创建区域按钮控件，0xBC 指令最少需要 27 个字节的参数，下面进行说明：

数据结构大小端定义(enpoind)（1 Byte）：

值为 0 时表示数据结构中数据为小端（Little-Endian）数据，也即大于 1 个字节的数据，高字节存于高地址端，低字节存于低地址端；值为 1 时为大端（Big-Endian），即大于 1 个字节的数据时，高字节存于低地址端，低字节存于高地址端。

控件 ID 号（2 Bytes）：

模块中的所有控件都有一个唯一的控件号，用户在发送指令创建新的控件时，需指定所创建的控件的 ID 号，且要确保它的唯一性，并且符合 EzUI 模块架构的基本要求，即 ID 号大于等于 100，小于等于 65535。

所属界面索引号（2 Bytes）：

每一个控件（资源文件中定义的以及动态创建的控件）都必需从属于一个已存在的界面，可以为所创建的控件指定其所从属的界面索引号。

控件左上角 X 轴坐标（2 Bytes）：所创建的控件左上角在显示屏中的 X 轴坐标。

控件左上角 Y 轴坐标（2 Bytes）：所创建的控件左上角在显示屏中的 Y 轴坐标。

控件右下角 X 轴坐标（2 Bytes）：所创建的控件右下角在显示屏中的 X 轴坐标。

控件右下角 Y 轴坐标（2 Bytes）：所创建的控件右下角在显示屏中的 Y 轴坐标。

响应类型（1 Byte）：可选值 1、2、3；1 值表示响应触摸按下，2 值表示响应触摸释放，3 值表示响应触摸按下与释放。

显示响应（1 Byte）：

分为高四位、低四位数据进行设置；低四位可选值 1~8、15；值 1~8 表示以相应数值的像素点宽度进行外框反显；15 值，表示控件区域整体反显。高四位数值表示显示响应（触摸响应时）的显示相对控件区域的缩进值。

重绘属性（1 Byte）：

可选值 0~108；0 值表示不重绘；值 1~8 表示以相应数值的像素点的线进行区域描边重绘（以描边色设置值进行绘线）；值 9~100 表示以相应数值的百分比进行区域填充重绘（以填充色设置值进行区域填充，值小于 100 时，可实现半透明效果）；值 101~108 表示以当前数值个位数宽度进行 3D 效果显示控件（以填充色设置值进行绘制）。

描边线颜色（2 Bytes）： 以 RGB565 的颜色格式设置控件的描线色。

填充颜色（2 Bytes）： 以 RGB565 的颜色格式设置控件的控件填充色。

中文字库索引号（2 Bytes）：

控件配有字符串时，显示中文字符所使用的点阵字库的中文字库在资源文件中索引号，当不使用中文字符或者使用系统矢量字库时，该值可设置为 0xffff。

西文字库索引号（2 Bytes）：

控件配有字符串时，显示 ASCII 西文字符所使用的字库设置；当使用点阵字库时，如索引号=0x8000~0x8003 表示选用为模块内部西文字库；当使用资源文件中已加载的西文字库资源项时，该值为对应的字库的资源序号（在资源文件当中）。如使用系统矢量字库，则该值应表示为 0x4000+FontSize；FontSize 为设置的矢量字库的字号大小，此时前面配置的中文字库索引号将无意义。

字符颜色（2 Bytes）： 控件显示字符串时，所使用的字符颜色，RGB565 的颜色格式。

字符串内容（>= 1 Byte）：

控件所配置的字符串内容，该部份数据至少应有 1 个字节，当要设置的控件不含字符串内容时，该部份数据为 1 个字节，且数据为 0。

示例：

在左上角坐标处（20，80），右下角（180，130）的区域创建一个 ID 号为 200 的区域按钮控件，控件配置到界面 1 当中，MCU 数据结构为小端，响应触摸释放事件，显示响应为缩进 3，对控件区域进行取反响应，控件重绘属性为透明度 80 的区域颜色填充；外框颜色为白色（0xffff），填充颜色为蓝色（0x001f），中文字库设置无效（0xffff），西文字库选择模块内置 0x8002，字符色为黄色（0xffe0）；控件字符串内容："Test AcCtrl01"。

MCU 发往显示模块的数据包内容如下：

0x55 0x29 0xBC 0x00 0xC8 0x00 0x01 0x00 0x14 0x00 0x50 0x00 0xB4 0x00 0x82 0x00 0x02 0x3F 0x50 0xFF 0xFF 0x1F 0x00 0xFF 0xFF 0x01 0x80 0xE0 0xFF **0x54 0x65 0x73 0x74 0x20 0x41 0x63 0x43 0x74 0x72 0x6C 0x30 0x31 0x00** 0x85 0xAA

上述数据中，用蓝色字符与黑色字符将数据包中每一段数据（包括帧头等）进行区分，以便理解；红色部份是字符串内容，数据结构中（不含字符串内容），以小端表示大于 1 字节的数值，如控件 ID 号，设置为 200，十六进制表示为 0x00C8，则在发送的数据中 0xC8 0x00 进行表示。

4.7.11 动态创建位图按钮控件

用户 MCU 可通过发送指令码“0xBB”的指令数据包创建区域按钮控件，0xBB 指令最少需要 21 个字节的参数，下面进行说明：

数据结构大小端定义(enpoind)（1 Byte）：

值为 0 时表示数据结构中数据为小端（Little-Endian）数据，也即大于 1 个字节的数据，高字节存于高地址端，低字节存于低地址端；值为 1 时为大端（Big-Endian），即大于 1 个字节的数据时，高字节存于

低地址端，低字节存于高地址端。

控件 ID 号（2 Bytes）：

模块中的所有控件都有一个唯一的控件号，用户在发送指令创建新的控件时，需指定所创建的控件的 ID 号，且要确保它的唯一性，并且符合 EzUI 模块架构的基本要求，即 ID 号大于等于 100，小于等于 65535。

所属界面索引号（2 Bytes）：

每一个控件（资源文件中定义的以及动态创建的控件）都必需从属于一个已存在的界面，可以为所创建的控件指定其所从属的界面索引号。

控件左上角 X 轴坐标（2 Bytes）：所创建的控件左上角在显示屏中的 X 轴坐标。

控件左上角 Y 轴坐标（2 Bytes）：所创建的控件左上角在显示屏中的 Y 轴坐标。

控件右下角 X 轴坐标（2 Bytes）：所创建的控件右下角在显示屏中的 X 轴坐标。

控件右下角 Y 轴坐标（2 Bytes）：所创建的控件右下角在显示屏中的 Y 轴坐标。

注：位图按钮控件的实际生成时，右下角坐标的数据取决于左上角坐标定度以及所引用的图片的大小，所以在创建时右下角坐标的数值并无实际意义，但要求右下角坐标的 X Y 轴数值大于左上角的 X Y 输数值。

常态图标索引号（2 Bytes）：图片在资源文件中的索引号，如指定的图片索引号不存在于资源文件当中，则创建失败。

按下图标索引号（2 Bytes）：图片在资源文件中的索引号，如指定的图片索引号不存在于资源文件当中，则创建失败。

响应类型（1 Byte）：

可选的数值：0、1、2；0 值时，表示图片按钮控件为乒乓开关；1 值表示控件响应触摸按下事件；2 值表示控件响应触摸释放事件。

显示响应（1 Byte）：可设置的数值为 0~8、15；0 值表示控件在触摸时使用图片切换显示进行显示的响应；1~8 值表示控件使用相应线宽进行外框取反显示响应；15 值表示控件区域取反显示响应。

预设状态（1 Byte）：值 0 或 1，该项配置只有控件属性设置为乒乓开关时有效。

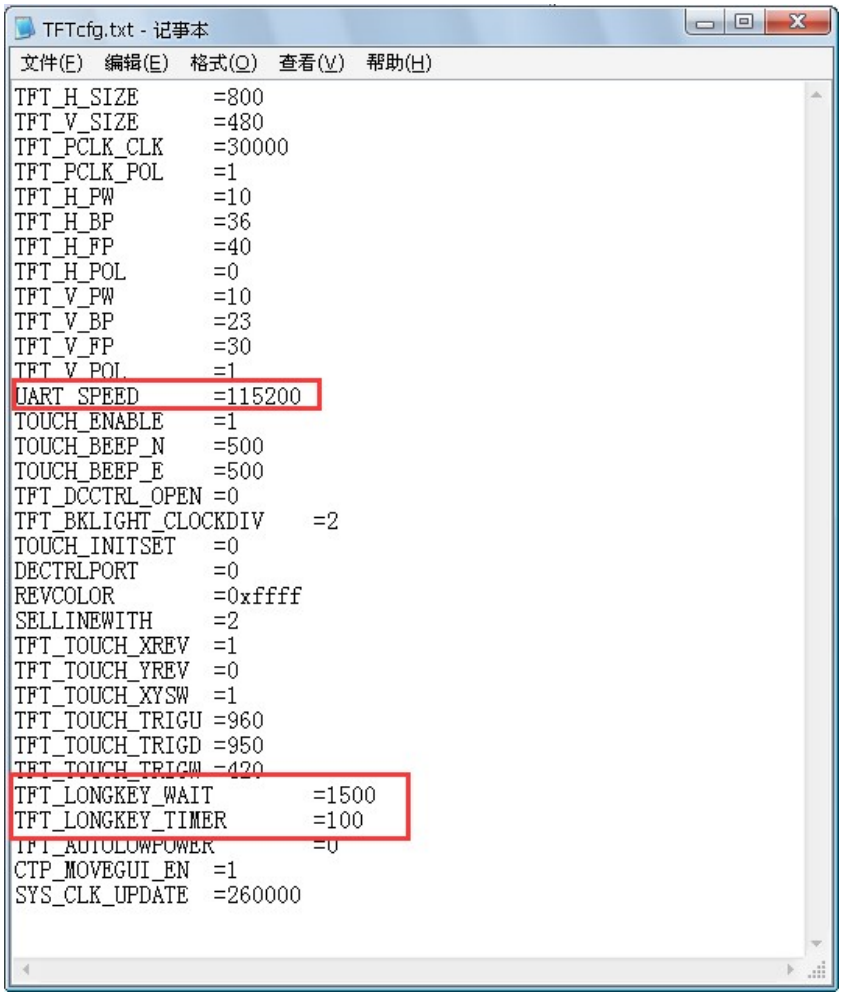
位图省缺色模式（1 Byte）：该项配置只有当控件所引用的图片为 BMP 位图时有效。

5 模块配置文件说明

EzUIH 系列模块需要在资源存储器当中预置一个配置文件，在模块连接电脑 USB 接口时，模块所被电脑认出的 U 盘当中该配置文件为一个“.TXT”文件；配置文件中有多项参数可设置，但仅有一项是对用户开放的，其它的参数请不要随意改动，否则可能会造成模块的显示不正常。

配置文件如下图所示：

配置文件打开后，如下图所示：



上图中，红框中的三项是可供用户配置的，其它内容请不要修改，以免模块无法正常显示。

UART_SPEED：该配置选项为模块上电后的 UART 波特率设置，用户可以自行修改该项配置，以适应需求。

TFT_LONGKEY_WAIT：该配置为模块长按键触发时间设置，该值小于 800 时，表示模块将不提供长按键功能，该数值以 ms 为单位。

TFT_LONGKEY_TIMER：该配置为模块长按键连续触发时间，该值也以 ms 为单位。

其它部分配置项说明：

TOUCH_ENABLE：触摸使能及类型配置，0 无触摸　1 电阻触摸屏　2 电容触摸屏

TFT_AUTOLOWPOWER：模块自动关屏时间设置，0~9999 表示关闭该功能；ms 为单位 有效数值为 10000~1200000 10S~20Min。

CTP_MOVEGUI_EN：电容触摸屏版本滑屏切换界面使能。

6 技术支持

鑫洪泰在 LCD 显示驱动方面有着丰富的经验，并通过积累总结出通用的成熟的 LCD 驱动程序架构，可方便的移植到不同的 LCD 应用上，或者移植到不同的 MCU 应用平台上。随着产品的推出，将会推出更多的完整的驱动程序，这些代码或应用开发软件用户可以放心的使用到产品当中。

而在人机界面编程方面，鑫洪泰也会给用户提供更多的参考，或者为用户订制专用的设计，使鑫洪泰的用户在得到质优价廉的产品的同时，也获得专业的技术支持。

鑫洪泰努力的目标是作为一个行业的**产品+技术的专业提供者**。

6.1 联系方式

- 深圳市鑫洪泰电子科技有限公司
- **WWW.HOTLCD.COM**
- 服务邮箱：**LCD@HOTLCD.COM**