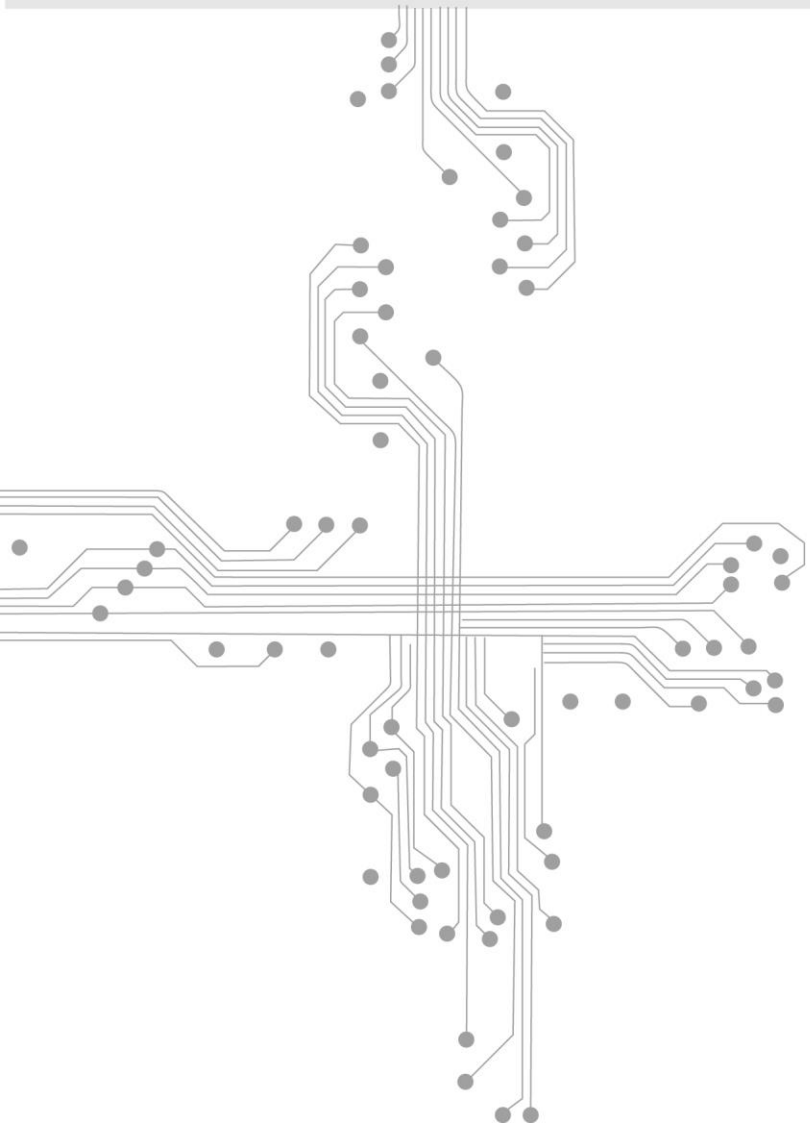
WEINVIEW

EasyBuild Pro
使用手册



目 录

第一章 关于 EasyBuilder Pro 安装

1.1 安装环境要求

软件来源:

可登录威纶通公司网站 http://www.weinview.cn 下载所有可用软件语言版本 (包括简体中文、繁体中文、英文、意大利文、韩文、西班牙文、俄罗斯文及法文版本) 及最新软件更新信息。

计算机硬件要求 (建议配置):

CPU: INTEL Pentium II 以上等级
内存: 256MB 以上
硬盘: 2.5GB 以上，最少有 500MB 以上的硬盘空间
光驱: 4 倍速以上光驱一个
显示器: 支持分辨率 1024 x 768 以上的彩色显示器
键盘和鼠标各一个
以太网接口: 工程下载 / 上传 时使用
USB 接口 2.0: 工程下载 / 上传 时使用
RS-232 COM 接口: 在线模拟时使用
打印机

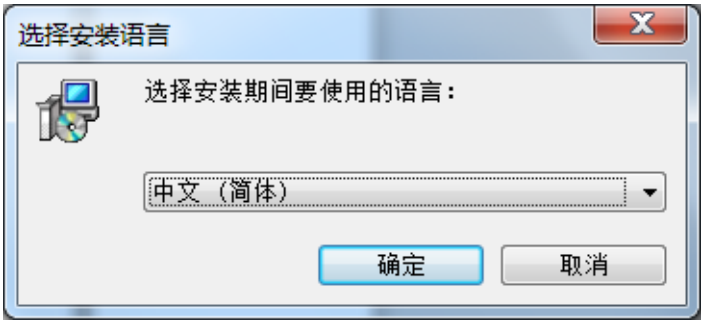
操作系统:

Windows XP / Windows Vista / Windows 7 / Windows 8 均可。

1.2 安装步骤

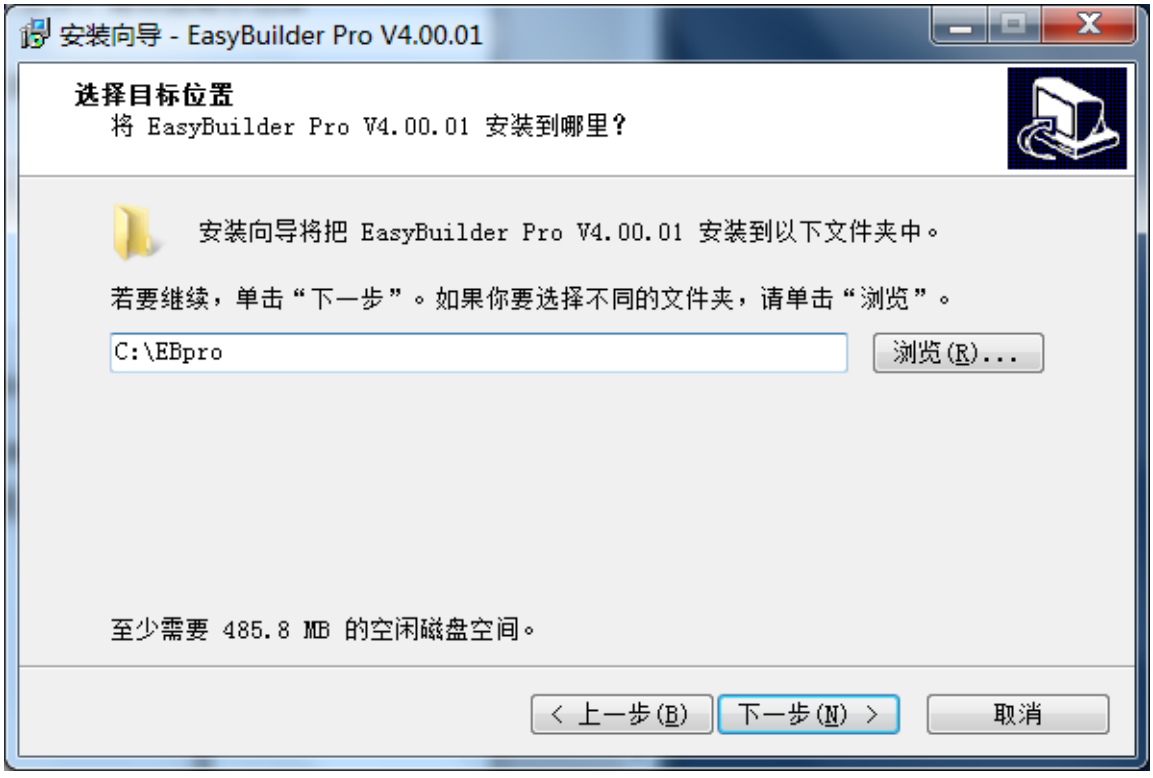
1. 将光盘放入光盘机，计算机将会自动执行安装程序，或者您手动执行光盘根目录下的 [autorun.exe] 文件，屏幕将显示安装程序如下：

2. 点选 [EasyBuilder Pro]，选择所需语言版本后，点选 [下一步]。



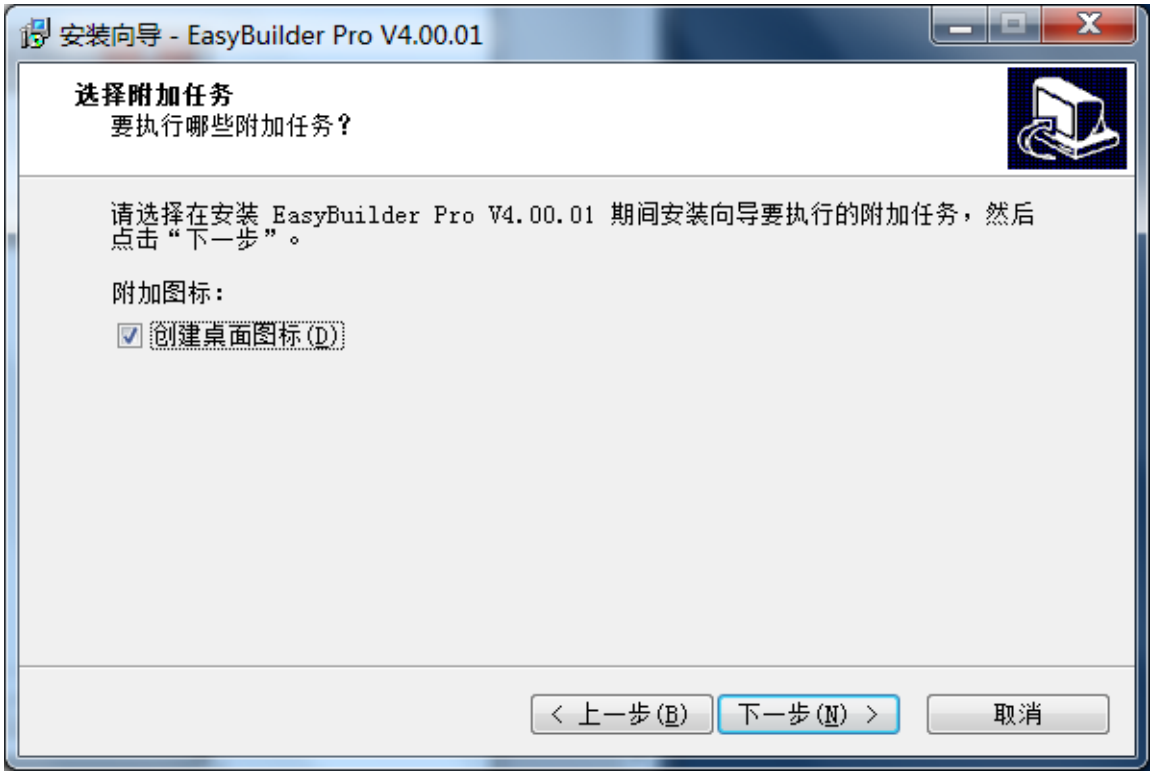
3. 系统将会询问使用者是否想要移除计算机中已安装过的 EasyBuilder Pro 版本，请根据需求选择后，点选 [下一步]。

4. 指定一个全新的文件夹给 EasyBuilder Pro 安装资料，或是直接使用系统建议的文件夹，点选 [下一步]。



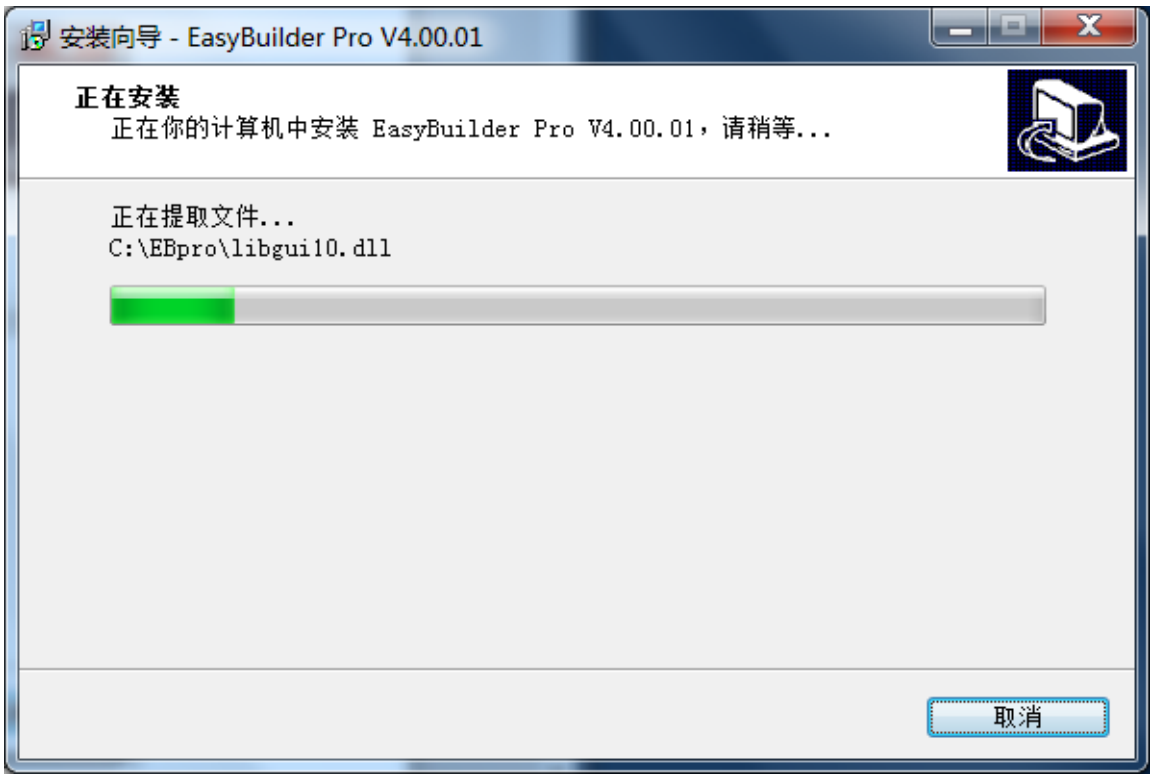
5. 系统会请使用者指定一个开始功能表文件夹用来建立程序捷径，点选 [浏览] 指定一个文件夹，或是使用程序建议的文件夹，点选 [下一步] 继续安装程序。

6. 系统会询问使用者是否要执行附加工作，例如 [建立桌面图示]，若需要请勾选，然后按 [下一步] 继续安装程序。



7. 在此阶段所有设定已完成，请检查是否无误，若有需要改选的部份，请按 [上一步]，若全部正确，请按 [安装] 开始安装程序。

8. 安装程序执行中。



9. 安装完成，请按［完成］结束安装。

10. 在［开始］»［所有程序］»［EasyBuilder Pro］目录下可看到 EasyBuilder Pro 各功能快捷方式。

软件目录下各选项的涵义如下:

项目	描述
Administrator Tool	可将［使用者账号］、［USB 安全金钥］、［e-Mail SMTP 服务器设定］、［e-Mail 联络人］四种资料储存于 U 盘并导入 HMI 之中。
EasyAccess	透过网络监测连结的触摸屏，并直接在计算机上操作触摸屏。
EasyBuilder Pro	EasyBuilder Pro 程序编辑软件。
EasyConverter	资料取样数据与事件记录文件转档工具。
EasyDiagnoser	在线监控程序通讯诊断工具。
EasyPrinter	远端屏幕打印及资料备份的服务器。
EasySimulator	执行程序模拟。
EasyWatch	可以透过 PC 监看或设定 HMI 和 PLC 内的地址数值，以达到监控的目的。
Recipe Editor	可以设定配方数据格式，开启配方数据文件及外部存储器文件。
Release Note	软件版本及相关最新信息说明。
Structure Editor	支持 AB TAG 结构，可提高元件读/写的灵活性。
Utility Manager	EasyBuilder Pro 综合管理软件。

Note

- HMI 支持使用 USB 线下载/上传程序，完成安装 EasyBuilder Pro 后，可至 [我的电脑] » [设备管理器] 确认 USB 驱动是否正确安装，若尚未安装，请手动完成安装。

第二章 关于 Utility Manager

2.1 概要

在 EasyBuilder Pro 软件安装完成后，双击电脑桌面上的 Utility Manager 快捷方式即可开启。这是 EasyBuilder Pro 软件的综合管理器，可当成独立的程序来操作。

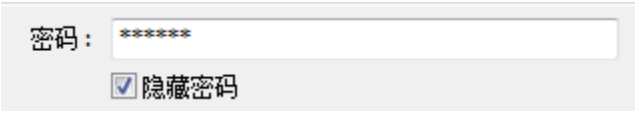
设定	描述
设计	**EasyBuilder Pro:** 启动 EasyBuilder Pro 程序编辑器。 **地址浏览程序:** 检视各个 PLC 的设备类型地址范围。 **模拟:** 在电脑上模拟工程文件的运行。
分析测试工具	**EasyDiagnoser:** 侦测 HMI 与 PLC 之间的通信状况。 详细信息请参考《33 EasyDiagnoser》。 **EasyWatch:** 可以透过 PC 监看或设定 HMI 和 PLC 内的

	地址数值。 详细信息请参考《35 EasyWatch》。 **重新启动 HMI:** 将 HMI 恢复到开机时的状态。 **穿透通讯设定:** 允许 PC 上的应用程序透过 HMI 直接连结 PLC。 详细信息请参考《29 Pass-through》。
传输	**下载:** 使用以太网下载工程文件到 HMI。 **上传:** 使用以太网上传 HMI 的工程文件到 PC。 **建立储存在 SD 卡与 U 盘中的下载资料：** 建立的资料可使用 SD 卡/U 盘下载到 HMI。cMT-SVR 系列不支持此功能。
维护	**EasyPrinter:** 启用远端备份/打印服务器。 **管理员工具:** 提供将 [用户账号]，[USB 安全密钥]，[e-mail SMTP 服务器设定]，[e-Mail 联系人] 四种资料储存于 U 盘。cMT 系列不支持此功能。 详细信息请参考《36 管理员工具》。 **EasyAccess**: 通过局域网或广域网来监控远端 HMI，详细说明可至 www.ihmi.net 查询。 **资料取样 / 事件记录档案信息**: 透过 USB 线或以太网连线查看 HMI 内的历史资料记录文件个数。cMT-SVR 系列不支持此功能。 **CloudHMI:** 连线至 cMT-SVR，并显示 cMT-SVR 预先载入的工程文件及数据资料。
数据转换	**配方数据库编辑器:** 可编辑配方资料。 **EasyConverter:** 可读取由 HMI 撷取的资料取样记录 (.dtl) 或事件记录 (.evt)，并转换成 Excel (.xls)格式。 详细信息请参考《25 EasyConverter》。 **配方资料/扩展内存编辑器:** 可建立 HMI 所使用的配方资料文件，也可开启及编辑现有的配方资料文件。 详细信息请参考《24 Recipe Editor》。
	将 Utility Manager 界面对话窗口最小化。
	关闭 Utility Manager。
	将常用的工具加入设定对话窗下方的工具列。
	执行在工具列中选择的程序。
	删除在工具列中选择的程序。

2.2 HMI 地址及密码设定

设定

当用户要用以太网或 USB 线操作 HMI 时，需正确设定操作 HMI 所需的密码，避免没有授权的用户入侵 HMI 程序。



下载

设定下载密码。若勾选 [隐藏密码]，输入密码时会以“*”符号显示。

Note

请妥善保管密码，若因忘记密码而要将 HMI 恢复为原厂设定时，将导致 HMI 内部的所有程序和资料被清除。

重新启动 HMI

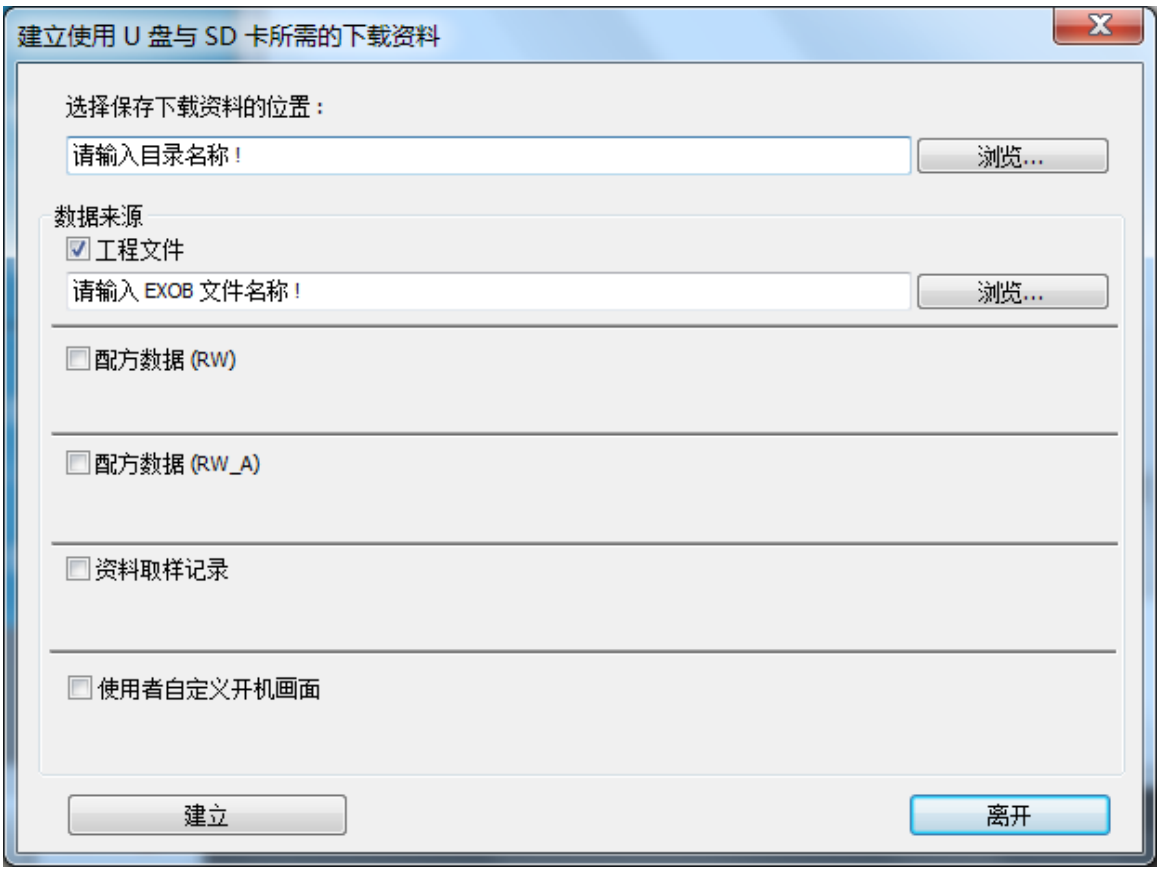
不需拔除电源即可重新启动 HMI，并恢复到一开机时的状态。若使用以太网重启 HMI 时，请设定正确的 HMI IP 地址。

资料取样 / 事件记录文件信息

设定好连结方式，即可连线至 HMI 查看内部历史资料文件个数。

2.3 编辑工具

2.3.1 建立储存在 SD 卡与 USB 中的下载资料



(1). 将 SD 卡或 U 盘插入 PC。
(2). 指定文件资料所要存放的路径位置。
(3). 指定所要建立的来源资料文件存放位置。
(4). 点选 [建立]。

所要建立的来源资料文件将写入所指定的外围设备中，让用户可以透过该设备下载工程文件至 HMI，可以不需要透过以太网或 USB 线下载工程文件。

2.3.2 透过 U 盘 / SD 卡下载程序到 HMI 的步骤

假设已经把来源资料文件建立在 U 盘里的文件夹名称 “123” *(K:\123)*

(1). 将 U 盘 (工程文件已包含在内) 接上 HMI。
(2). 弹出 [Download / Upload] 窗口，请选择 [Download]。
(3). 输入下载密码。

(4). 在 [Download Settings] 窗口，勾选 [Download project files] 以及 [Download history files]。
(5). 点击 [OK]。
(6). 在 [Pick a Directory] 窗口，选择路径：usbdisk\disk_a_1\123。
(7). 点击 [OK]。

工程文件将被自动更新。

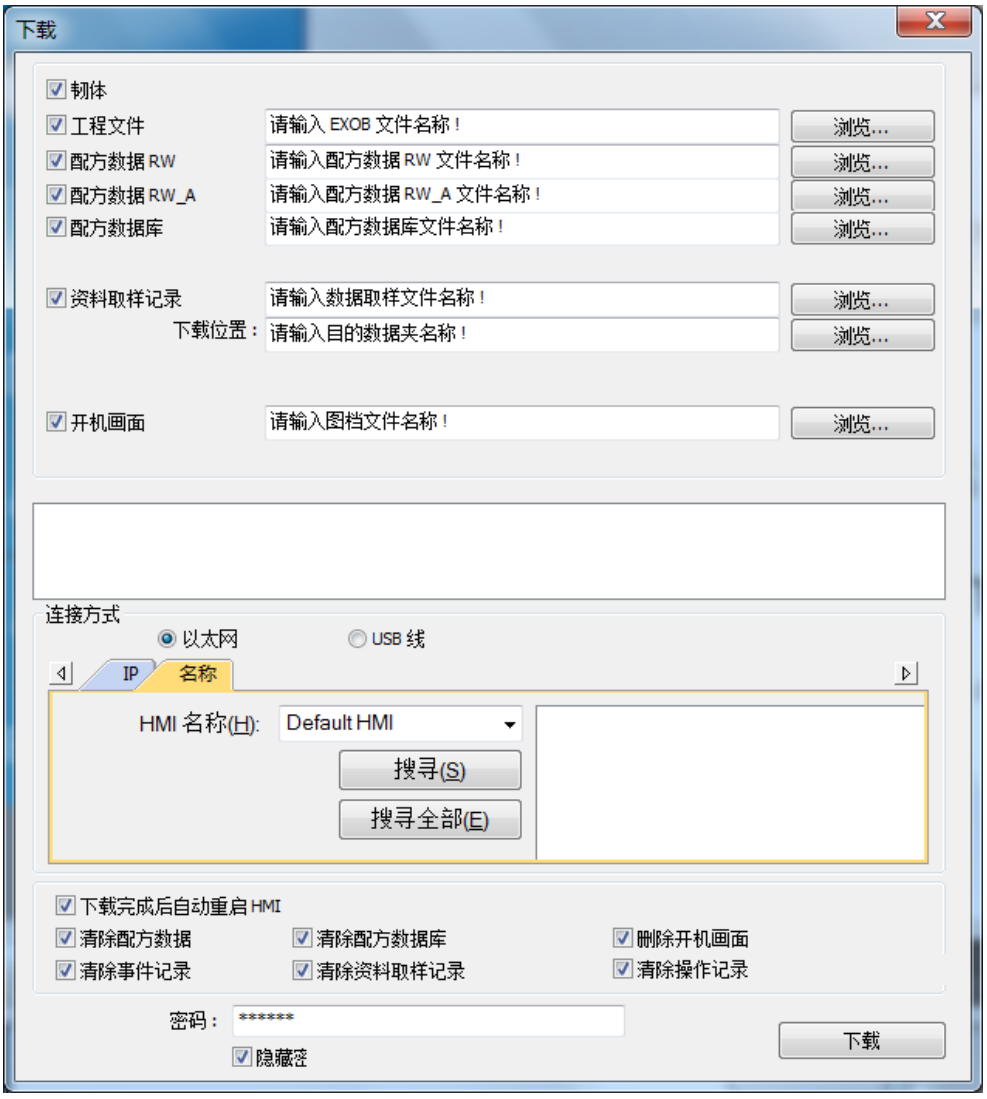
Note

若没有下载工程文件而只有下载历史资料，必须手动去重新启动 HMI 更新文件。

2.4 传输

2.4.1 下载

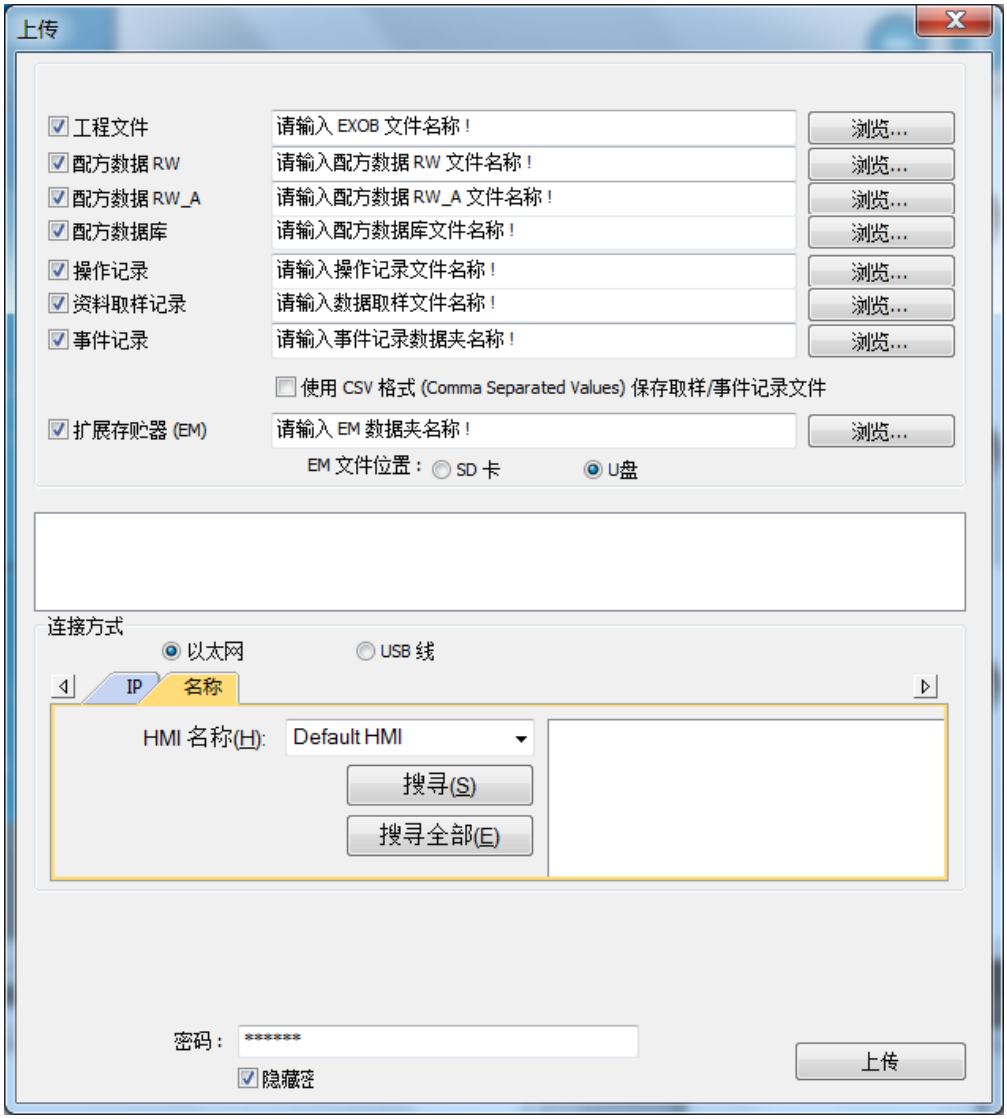
透过此功能可以使用以太网或 USB 线下载文件到 HMI 上。



设定	描述
韧体	若勾选此项，表示要更新 HMI 所有核心程序。 第一次下载文件至 HMI 时，一定要下载韧体。
工程文件	选择 .exob 格式的工程文件。
配方数据 RW / RW_A	选择 .rcp 格式的配方文件。
配方数据库	选择 .db 格式的配方数据库文件。
资料取样记录	先选择 HMI 上资料取样的文件夹名称后，再选择 .dtl 格式的资料取样文件。
开机画面	将指定的 .bmp 图片下载到 HMI，HMI 启动时，就会先显示此图片，再载入下载的程序。
下载完成后自动启动 HMI	若勾选此项，HMI 将会在下载文件成功后自动重新启动。
清除 配方数据 / 事件记录 / 配方数据库/ 资料取样记录 / 删除开机画面/ 清除操作记录	下载文件前会先清除勾选的文件。

2.4.2 上传

透过此功能可以使用以太网或 USB 在线传 HMI 的文件到 PC。上传前须先选择存放文件的路径。按下 [浏览]，指定上传文件所要存放的位置。



设定	描述
事件记录	将 HMI 的 .evt 文件上传到 PC。
扩展存储器 (EM)	将 HMI 上的 SD 卡、USB 碟内的 .emi 文件上传到 PC。

关于 [工程文件]、[RW / RW_A]、[配方数据库]、[资料取样记录] 请见本章《2.4.1 下载》。

Note

- 若将工程文件上传至 PC，由于上传回来的文件格式为 .exob 文件，用户需先反编译为 .emtp 文件，

才能透过 EasyBuilder Pro 编辑。

2.5 模拟

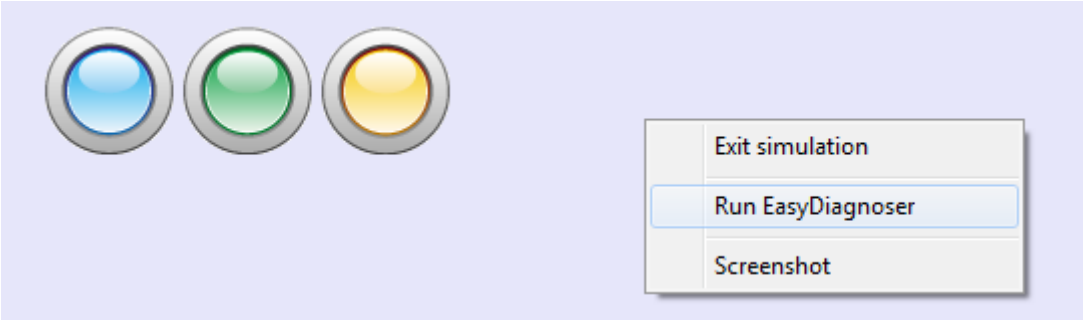
2.5.1 离线模拟和在线模拟

离线模拟 – 在 PC 上模拟工程文件的运行，不与任何装置连线。
在线模拟 – 在 PC 上模拟工程文件的运行，此时 PLC 是直接与 PC 连接。

Note

- 在 PC 上进行 [在线模拟] 时，若监控设备是接在本地 PC 上的 PLC，监控时间会有 10 分钟的限制。

执行在线模拟和离线模拟功能，需先选择 .exob 文件的来源位置。
在执行在线模拟/离线模拟时，点选鼠标右键后可以执行以下功能：



设定	描述
Exit simulation	关闭模拟状态。
Run EasyDiagnoser	执行 EasyDiagnoser 监看目前的通讯状态。
Screenshot	将目前的模拟画面使用图片的方式储存到安装路径下的 screenshot 文件夹。

2.6 穿透通讯

穿透通讯功能允许 PC 上的应用程序透过 HMI 直接连结 PLC，此时 HMI 所扮演的角色类似转接器。

穿透通讯功能包含 **[串行端口]** 模式与 **[以太网]** 模式。
在使用 **[以太网]** 穿透通讯功能前，请先安装虚拟串口驱动程序。

详细信息请参考《29 穿透通讯》。

第三章 建立简单的工程文件

3.1 概要

建立一个工程文件的基本步骤如下：

1. 建立新的工程文件。
2. 保存/ 编译工程文件。
3. 执行在线模拟 / 离线模拟。
4. 下载工程文件至 HMI。

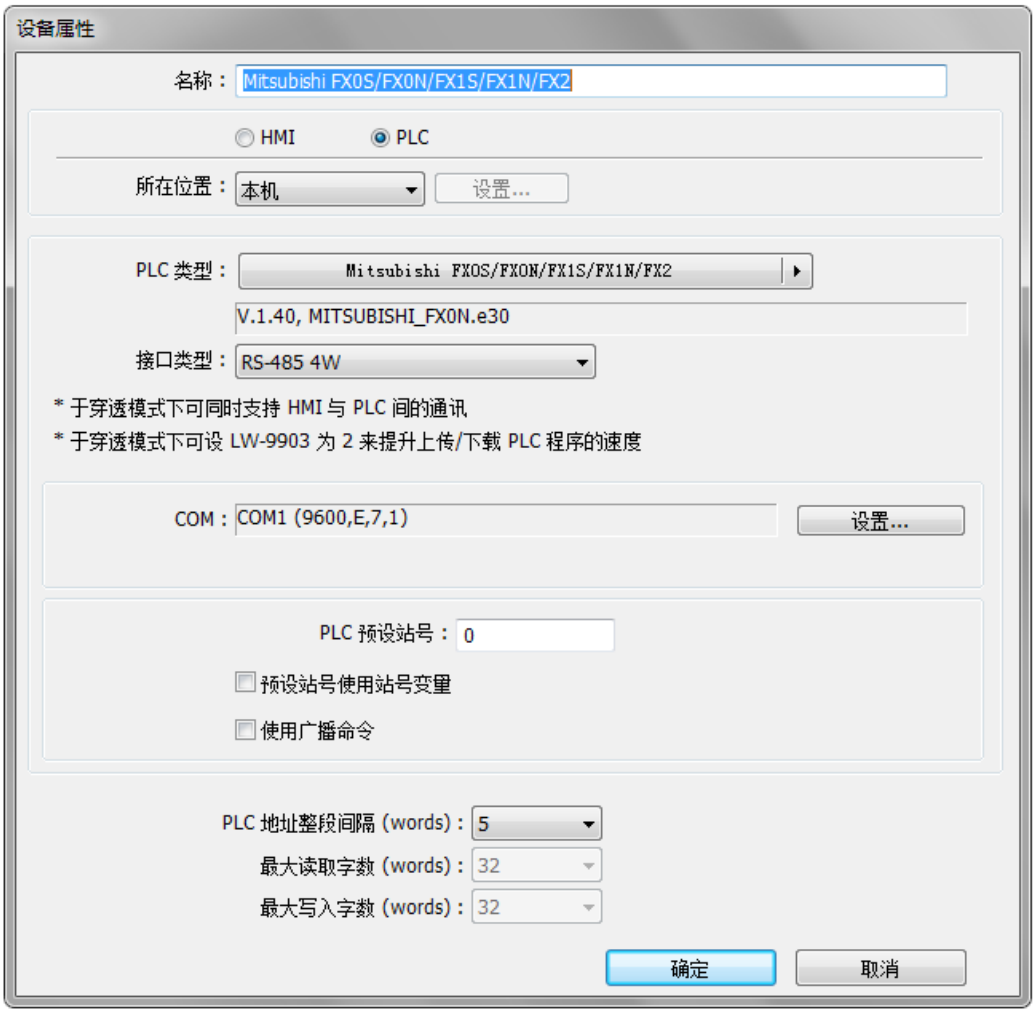
以下将说明每个步骤的设定方法。

3.2 建立新的工程文件

1. 进入 EasyBuilder Pro 并开启新工程。
2. 选择 [型号]，勾选 [使用模板]。

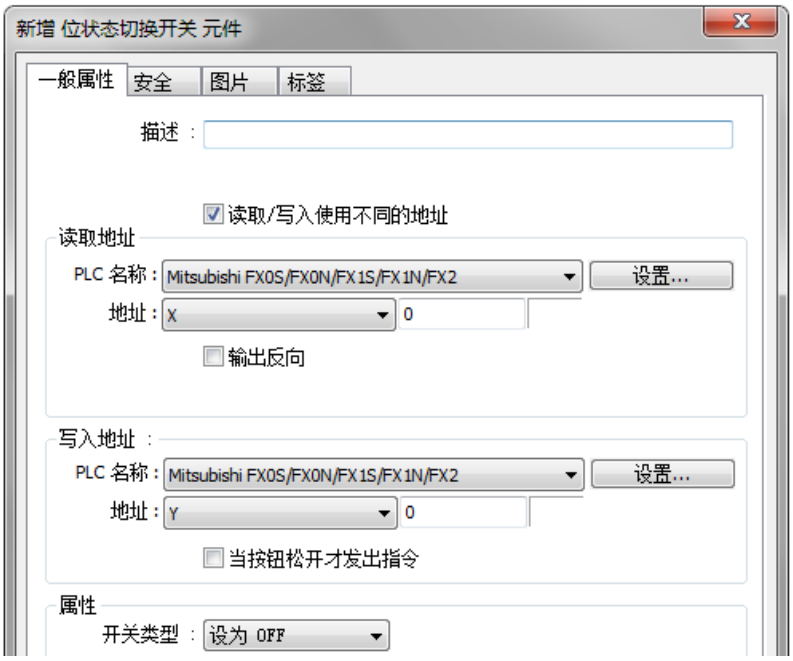
3. 点击 [新增]。

4. 设定正确参数。

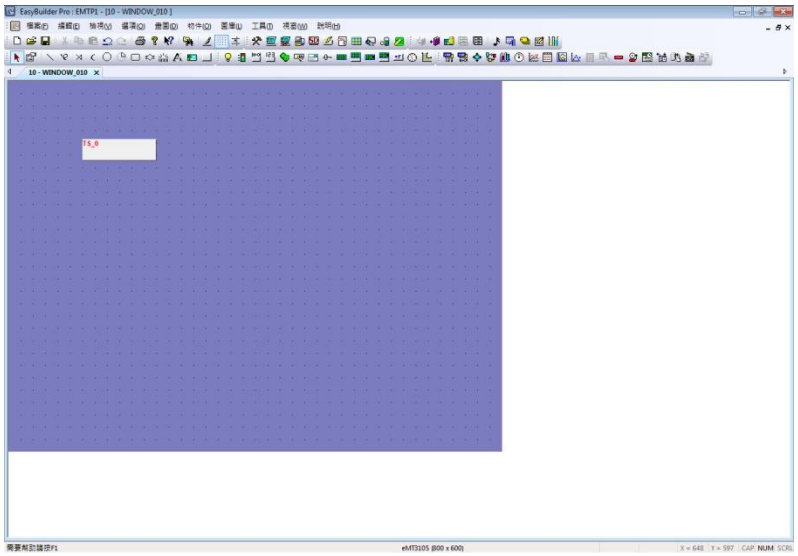


5. [设备列表] 增加了一个新的装置。

6. 建立一个新元件，以 [位元状态切换开关] 元件 为例，设定 PLC 地址。



7. 将元件放置于编辑视窗中，即完成一个简单的工程文件。



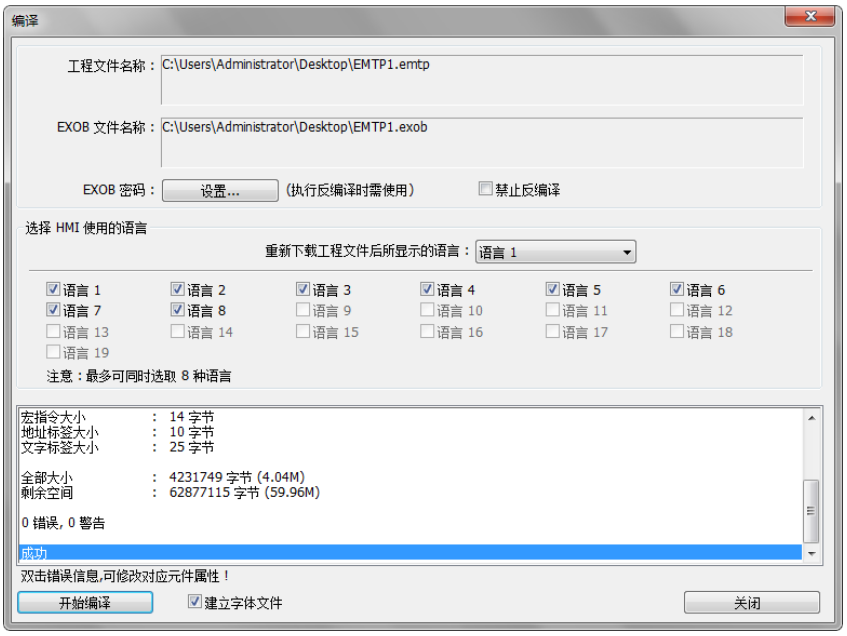
3.3 保存和编译工程文件

1. 在 EasyBuilder Pro 的工具列上，点击 [文件] » [保存文件] 存成 .emtp 文件。
2. 在 EasyBuilder Pro 的工具列上，点击 [工具] » [编译] 将文件编译为 .exob 文件。将此文件下载至 HMI，并检查工程文件可否正常运行。

Note

- cMT-SVR 保存的文件格式为 .cmtp 文件，编译的文件格式为 .cxob 文件。

3. 用户需事先于文字标签库设定多国语言后，再选择工程文件所需要的语言并下载至 HMI，最多可勾选 8 种语言。编译成功的对话框如下所示：



3.4 离线模拟和在线模拟

离线模拟：在 PC 上模拟工程文件的运行，不与任何装置连线。

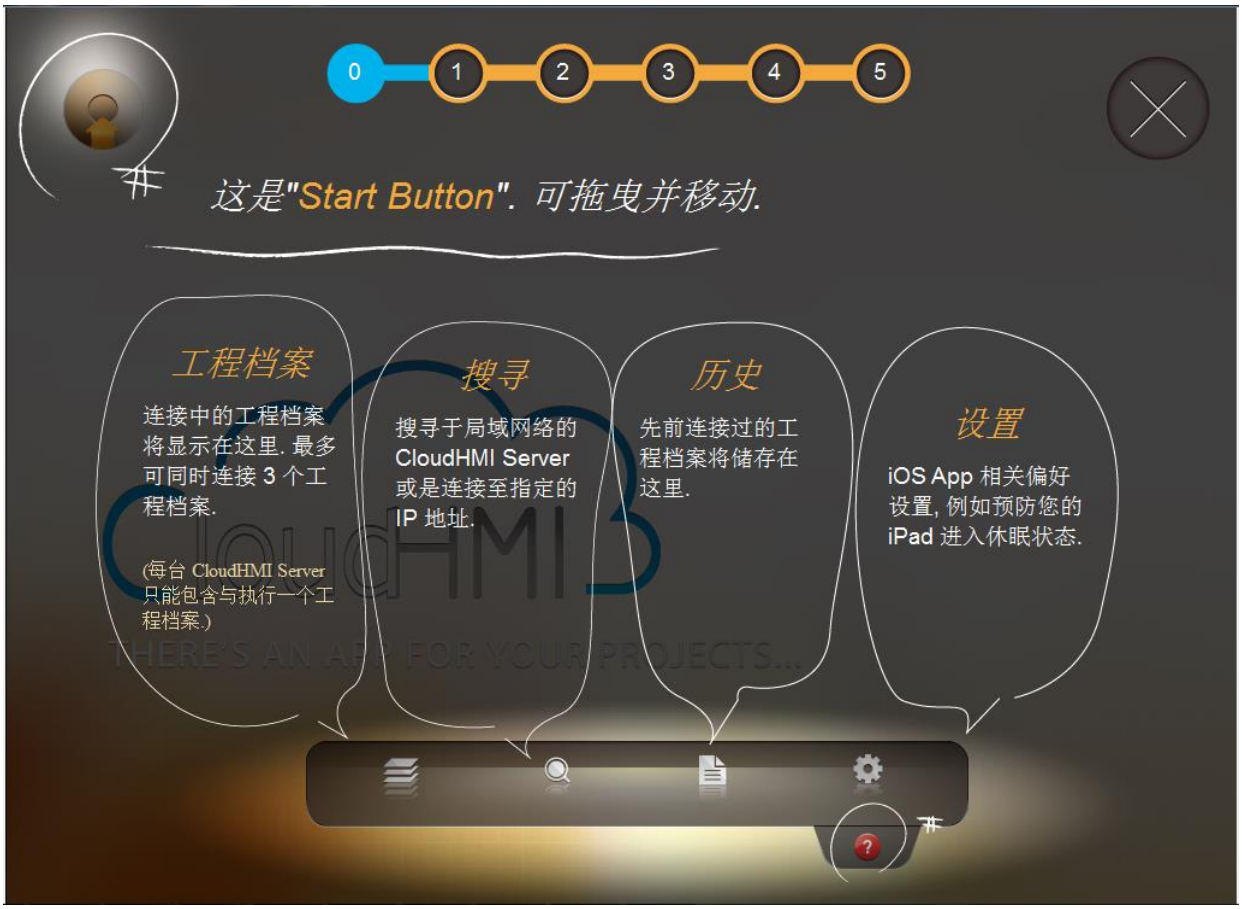
在线模拟 ：在 PC 上模拟工程文件的运行，不需将程序下载到 HMI。此时 PLC 是直接与 PC 连接，请设定正确参数。

Note

- 在 PC 上进行 [在线模拟] 时，若监控设备是接在本地 PC 上的 PLC，监控时间会有 10 分钟的限制。

3.5 CloudHMI

CloudHMI 可以利用网络连接到 CloudHMI Server (cMT-SVR)，就象是一台 iPad 执行 CloudHMI iOS 程序，欲执行此程序请到安装目录下执行 CloudHMI.exe，或是在 EasyBuilder Pro 的工具列上，点击 [工具] » [CloudHMI]。



设定	描述
Start Button	可进入程序主要操作画面并可以任意拖曳移动此按钮。
工程文件	正在运行中的工程文件会显示在此，最多可同时执行三个工程文件。(每一个 CloudHMI Server 只会执行一个工程文件)
搜索	搜寻在网络上的 CloudHMI Server，或者是连接指定的 IP 地址。
历史	显示先前曾经连接过的工程文件。
设置	应用程序设定相关的喜好，例如: 垂直模式。

3.6 下载工程文件至 HMI

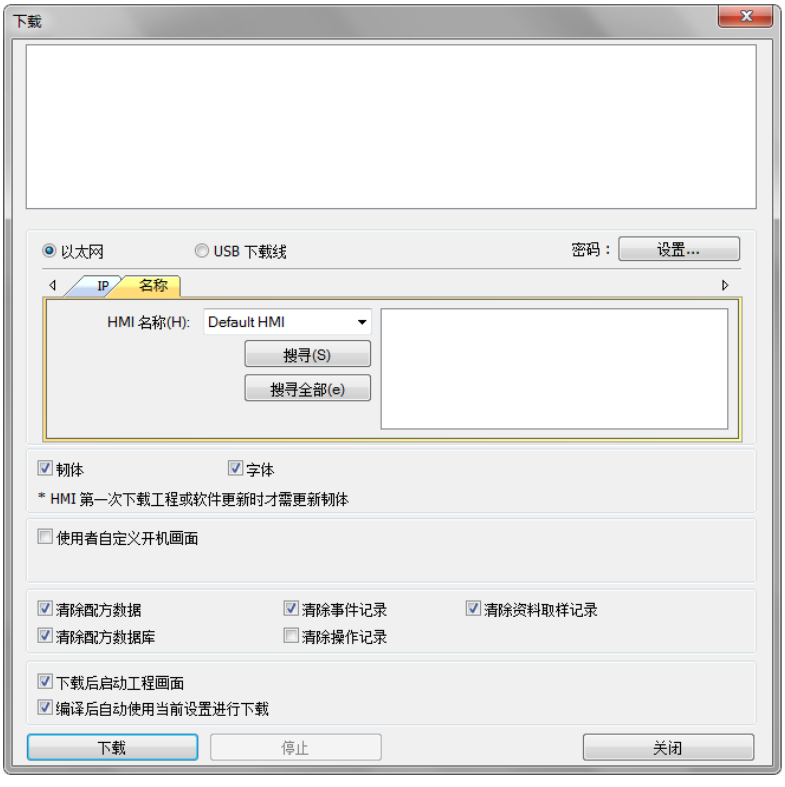
以下介绍四种下载工程文件至 HMI 的方法。

Note

- cMT-SVR 只适用 3.6.1 下载方法。

3.6.1 自 EasyBuilder Pro 设定

1. 在 EasyBuilder Pro 的工具列上，点击 [工具] » [下载]。请先确认所有设定是否正确。
2. 选择 [以太网]，设定 [密码] 并指定 [HMI IP]。



设定	描述
韧体	勾选此选项表示要更新触摸屏上的所有核心程序。第一次下载工程文件或更新 EasyBuilder Pro 版本并下载文件至 HMI 时，一定要下载此韧体。
字体	将工程文件中选用的字体下载至 HMI。
清除配方数据 / 配方数据库 / 事件记录 /资料取样记录	选项如被勾选，下载程序前会先清除机器上所选取存在的文件。
下载后启动工程画面	此选项如被勾选，下载程序完成后会自动重新启动 HMI。
编译后自动使用当前设置下载	如果勾选此项，下一次只要点选 [下载]，EasyBuilder Pro 将自动编译程序并下载到上次下载的目标 HMI，请见下方说明。

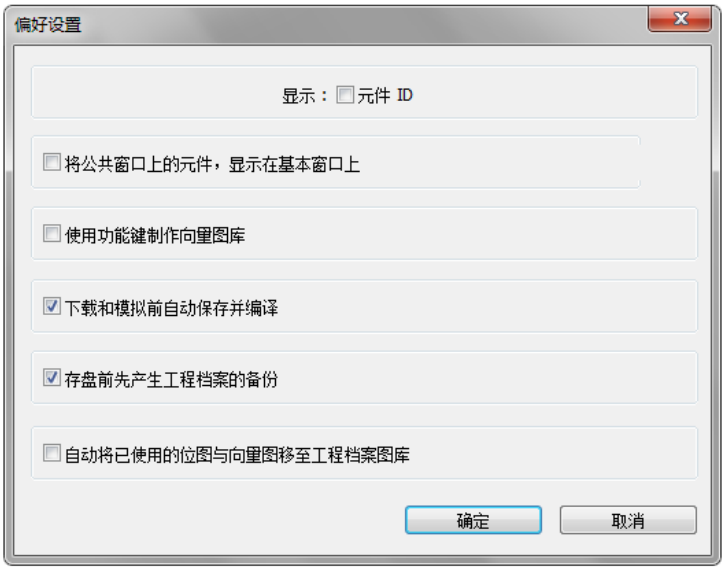
Note

■ cMT-SVR 没有 [韧体] 与 [字体] 等选项。

[编译后自动使用当前设置下载]

如果勾选此项，下一次只要点选 [下载]，EasyBuilder Pro 将自动编译程序并下载到上次下载的目标 HMI。

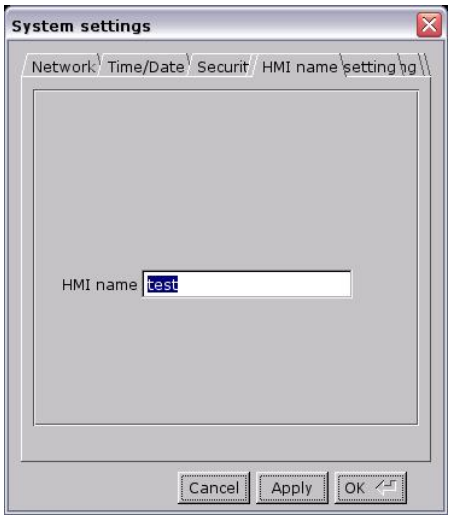
1. 在 EasyBuilder Pro 的工具列上，点击 [选项] » [偏好设定]。
2. 勾选 [下载和模拟前自动保存并编译]。



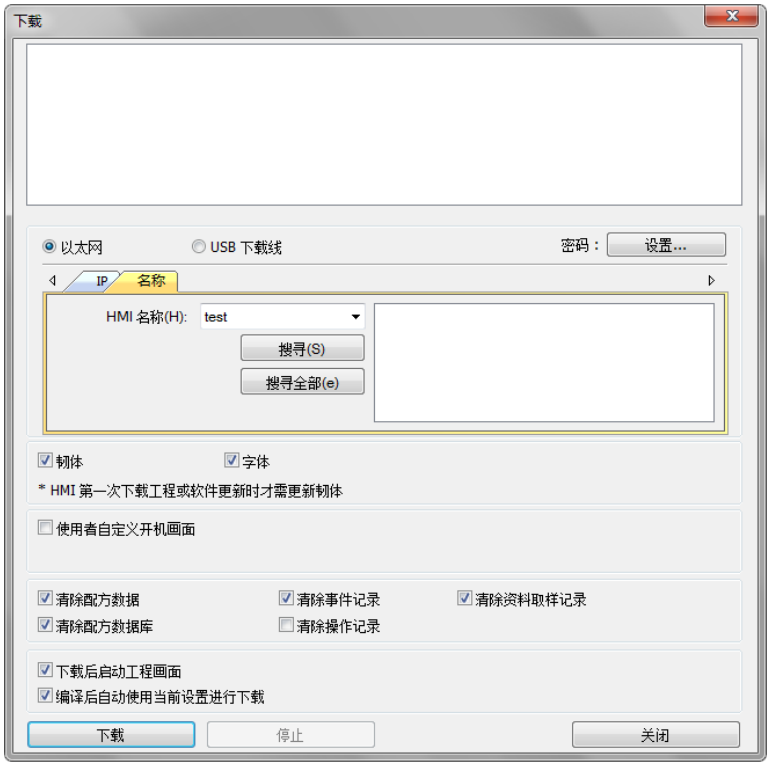
3. 在 EasyBuilder Pro 的工具列上，点选 [保存] 工程文件，再点选 [下载]。
4. 勾选对话窗上 [编译后自动使用当前设置下载]。
5. 点击 [下载]。
6. 完成以上设定后，下一次只要点选 [下载]，EasyBuilder Pro 将自动编译程序并下载到上次下载的目标 HMI。

3.6.2 使用 HMI 名称

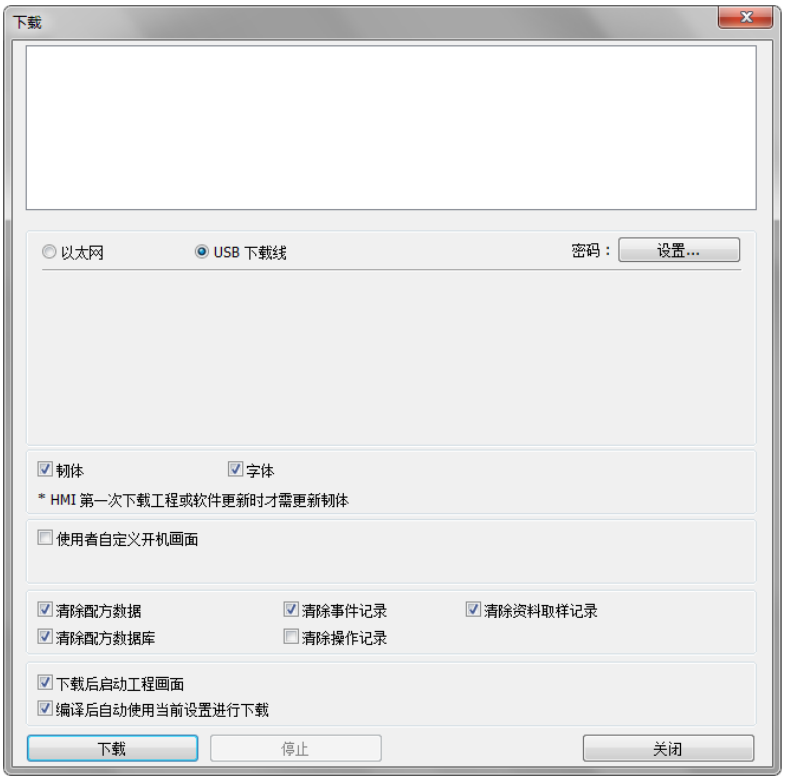
1. 在 HMI 上的 System settings 先设定 HMI name。



2. 在计算机上，选择先前设定的 HMI 名称并开始下载。若使用 [搜寻]，请在 [HMI 名称] 中输入要搜寻的特定 HMI 名称。若使用 [搜寻全部]，则搜寻同网域的所有 HMI。



3.6.3 使用 USB 下载线



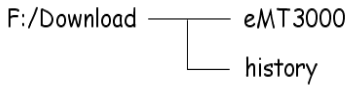
选择 USB 线下载程序，其余设定请参考《3.6.1 自 EasyBuilder Pro 设定》。使用 USB 线传输程序前，需至 [我的电脑] » [设备管理器] 确认 USB 驱动是否安装完成，若尚未被安装，请参阅 安装步骤 手动完成安装。

3.6.4 使用 U 盘 / SD 卡

以下说明使用 U 盘或 SD 卡下载工程文件的步骤。

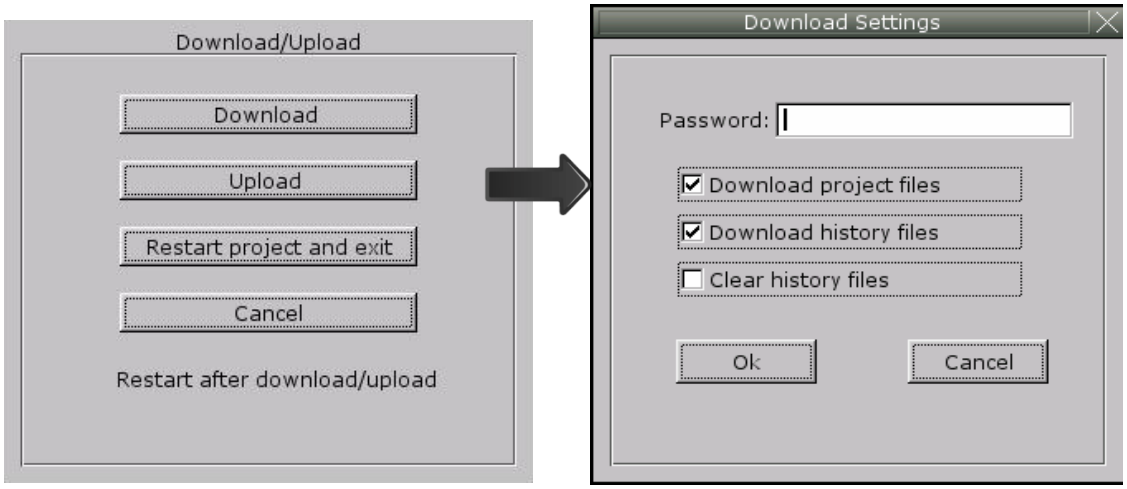
1. 使用 Utility Manager 中的 [建立使用在 U 盘与 SD 卡所需的下载资料] 先建立要下载的资料。一般分为两个目录，如果设定如下：

则下载资料的存放结构为：

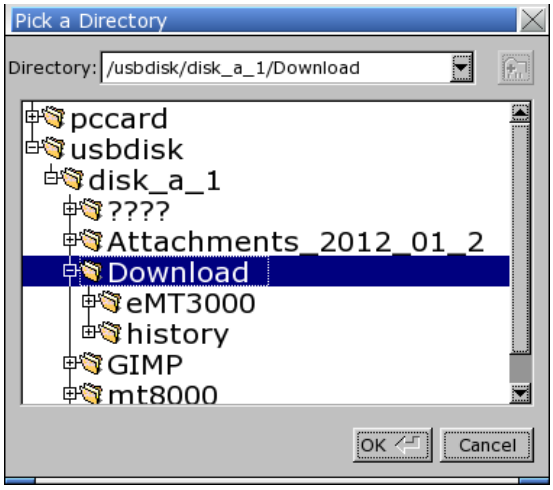


“history”目录在存放工程文件需要下载历史资料时产生。

2. 插入外围设备如 SD 卡或 U 盘至 HMI。
3. 在 HMI 上选择 [Download]，输入密码。



4. 密码确认后会显示外部装置下的目录名称。(pccard：SD 卡；usbdisk：U 盘)



5. 选择工程档存放路径，按下 [OK] 即开始下载。

Note

- 此时下载文件时必须选择存放**下载资料的最上层路径**，以上图为例，必须选择 Download，不可选择 eMT3000 或 history。

第四章 硬件设定

4.1 I/O 端口

HMI 支持的通讯接口，依不同机种而有差异，详细规格请参阅各机型的规格表。

- SD 卡插槽：SD 卡提供工程文件上传及下载，包括配方数据、事件记录与资料取样记录，亦可备份或记录历史资料。
- 串行接口：连接 PLC 或其他设备，接口规格为：RS-232 / RS-485 2W/ RS-485 4W / CAN Bus。
- 以太网接口：提供工程文件上传及下载，包括配方数据、事件记录与资料取样记录。亦可连接具网络通讯功能的设备，如 PLC、PC 等。
- USB Host：支持各种 USB 接口的设备，如鼠标键盘、U 盘、打印机、条码机等。
- USB Client：提供工程文件上传及下载，包括配方数据、事件记录与资料取样记录。

当首次操作 HMI 前，必须在 HMI 上完成以下各项系统设定，设定完成后即可使用 EasyBuilder Pro 工程软件开发工程文件。

4.2 系统设置

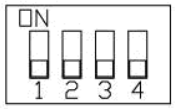
每台 HMI 背后都有一组复位按钮及指拨开关，做不同模式切换时，将可触发对应功能。若遗失 HMI 的系统设定密码时，可以藉由调整指拨开关将 HMI 恢复成出厂设置。详细设定步骤如下：

5. 将 DIP Switch 1 切至 ON，其余指拨开关保持为 OFF，然后重新启动 HMI。此时 HMI 将进入触控校正模式。
6. 在 HMI 会出现“+”光标。使用触控笔或者手指点选“+”光标的中心点【持续按住 2 秒钟左右】进行五点校正。所有十字皆被准确触控之后，“+”光标会消失。校准参数会保留在系统里。
7. 完成校正动作后，系统会询问用户是否将 HMI 的系统设定密码恢复为出厂设定，选择 [Yes]。
8. 再次确认用户是否要将 HMI 的系统设定密码恢复为出厂设置。当输入 [yes] 按下 [OK] 后，HMI 内所有的工程文件及历史资料将全部被清除。
   (出厂时的 [Local Password] 预设密码为 111111；但其他密码，包括下载与上传所使用的密码当选择恢复出厂值设定后，皆需重新输入)。

以下为各机型指拨开关设置，请参阅相关安装说明书。

eMT / iE

Dip Switch



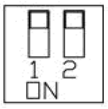
SW	SW	SW	SW	模式
ON	OFF	OFF	OFF	触控校正模式
OFF	ON	OFF	OFF	隐藏系统工具列
OFF	OFF	ON	OFF	Boot 载入模式
OFF	OFF	OFF	ON	保留
OFF	OFF	OFF	OFF	正常模式

Note

- 每台 HMI 的 Dip Switch 4 的开关位置可能有所不同。 若 Dip Switch 4 的开关出厂时已被剪掉，代表此台 HMI 的 Dip Switch 4 必须设定为 ON。若 Dip Switch 4 的开关出厂时被保留，代表此台 HMI 的 Dip Switch 4 必须设定为 OFF。

cMT-HD

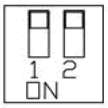
Dip Switch



SW	SW	模式
ON	ON	恢复出厂设置
ON	OFF	隐藏系统工具列
OFF	ON	Boot 载入模式
OFF	OFF	正常模式

cMT-SVR

Dip Switch



SW	SW	模式
ON	ON	恢复出厂设置
ON	OFF	恢复以太网 IP 设定
OFF	ON	Boot 载入模式
OFF	OFF	正常模式

4.3 系统工具列

启动 HMI 后可利用在屏幕下方的 [工具列] 做系统设定，一般情况下它是自动隐藏的，用户只需点击屏幕右下角的箭头图示即会弹出工具列，如图示，由左向右分别为：系统设定、系统信息、文字键盘、数字键盘。

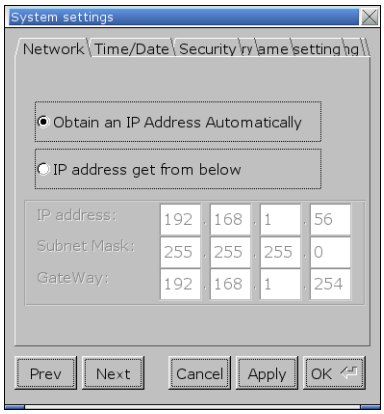
隐藏 HMI 系统设定列的方法：

- 将 Dip Switch 2 设为 ON，系统设定列会被隐藏，设为 OFF，系统设定列便可显示并被控制。用户需重启 HMI 来启用/停止这个功能。
- cMT-HD 系列则需将 Dip Switch 1 设为 ON，隐藏系统设定列。
- 另外可使用系统寄存器 [LB-9020] 来显示/隐藏系统设定列，当 [LB-9020] 设为 ON，此工具列

会被显示，当设为 OFF，则会被隐藏。

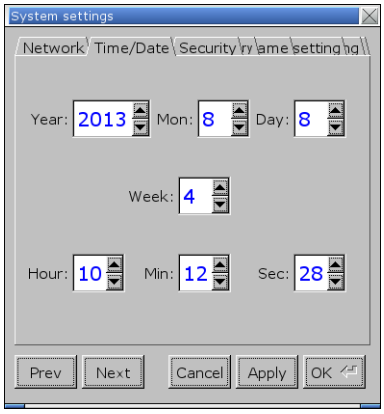
4.3.1 系统设定

设定或变更 HMI 的各项系统参数，基于安全考量必须进行密码确认。出厂时的预设密码为 111111。



Network

用“以太网”下载工程文件到 HMI 上，需正确设定操作对象 (HMI) 的 IP 地址。可选择自动取得 IP 地址或自行输入 IP 地址。



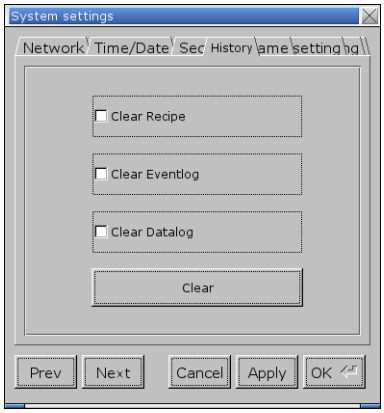
Time / Date

设定 HMI 内本地的日期、时间。

Security

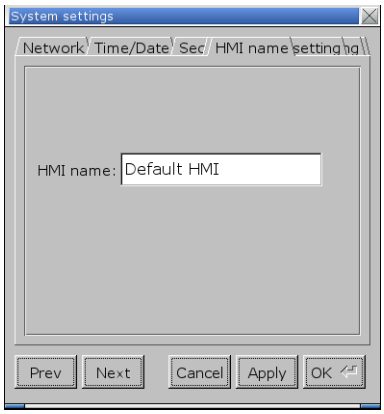
HMI 的密码防护，预设密码为 111111，请用户设定自己的密码，完成后始可使用该密码。

[进入系统设定的密码]
[上传工程文件的密码]
[下载工程文件的密码]
[上传历史记录的密码]



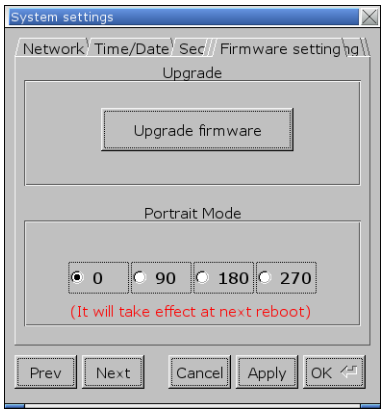
History

清除储存于 HMI 内的历史记录：
[清除配方数据]
[清除事件记录]
[清除资料取样]



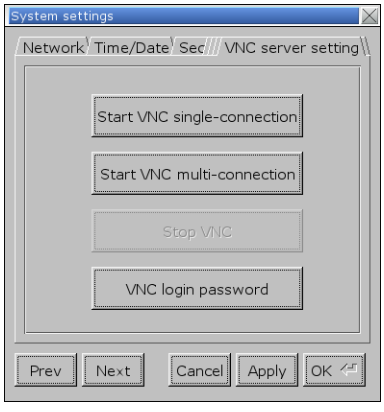
HMI name

设定 HMI 名称以便于下载/上传工程文件时辨识。



Firmware setting

更新系统韧体及调整显示模式。当调整了显示模式后，需将 HMI 重启才可生效。

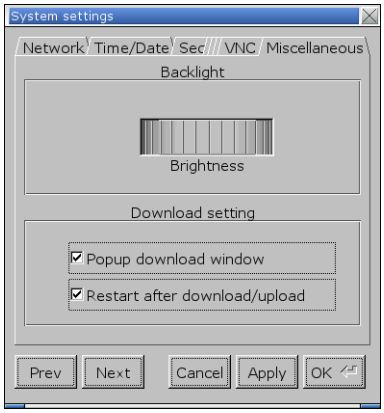


VNC server

启用后，可使用 VNC 透过以太网监控远端 HMI。

[Start VNC Single-connection]
允许一台 VNC Client 装置连接。

[Start VNC multi-connection]
允许多台 VNC Client 装置连接。
同时连接越多 VNC Client 装置可能影响 HMI 的通讯速度。
设定步骤请见如下说明。



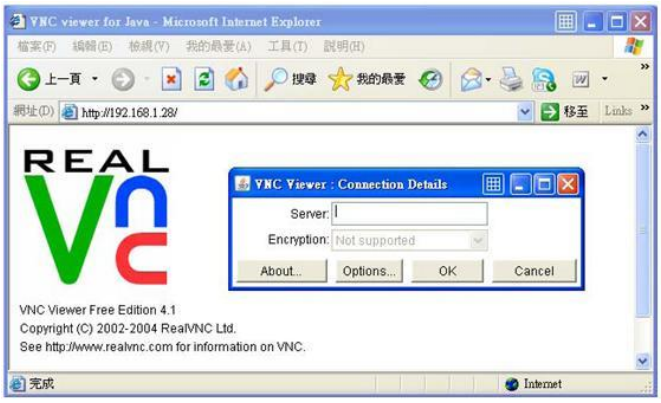
Miscellaneous
旋钮可调整 LCD 画面亮度。

[Popup download window]
启用后，接上 U 盘或 SD 卡后，显示上传/下载选项对话框。

[Restart after download/upload]
启用后，当工程文件被上传或下载后会重新启动 HMI。

以下为设定 VNC Server 的步骤。

1. 开启 HMI 的 VNC server 并设置登入密码。
2. 安装 Java IE 或 VNC viewer 到 PC。
3. 安装 Java IE 后可透过 IE 浏览器输入远端 HMI 的 IP 地址。
   或者透过 VNC viewer 输入远端 HMI 的 IP 地址和密码。

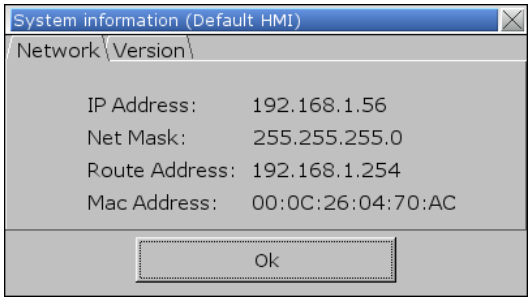


Note

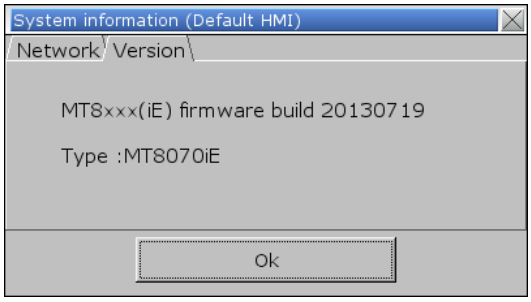
- 当 VNC server 持续一小时没有操作，系统将自动登出。
- cMT-HD 系列不提供 VNC server 功能。

4.3.2 系统信息

Network：显示网络信息，包含 HMI 的 IP 地址等。



Version：显示 HMI 系统版本及机型信息。



第五章 系统参数设置

5.1 概要

在 EasyBuilder Pro 窗口 [编辑] 下拉菜单的 [系统参数设置] 后，可以弹出 [系统参数设置] 对话框。系统参数设置的各页如下所示，下面将说明各页的内容。

eMT、iE、cMT-HD 系列

cMT-SVR 系列

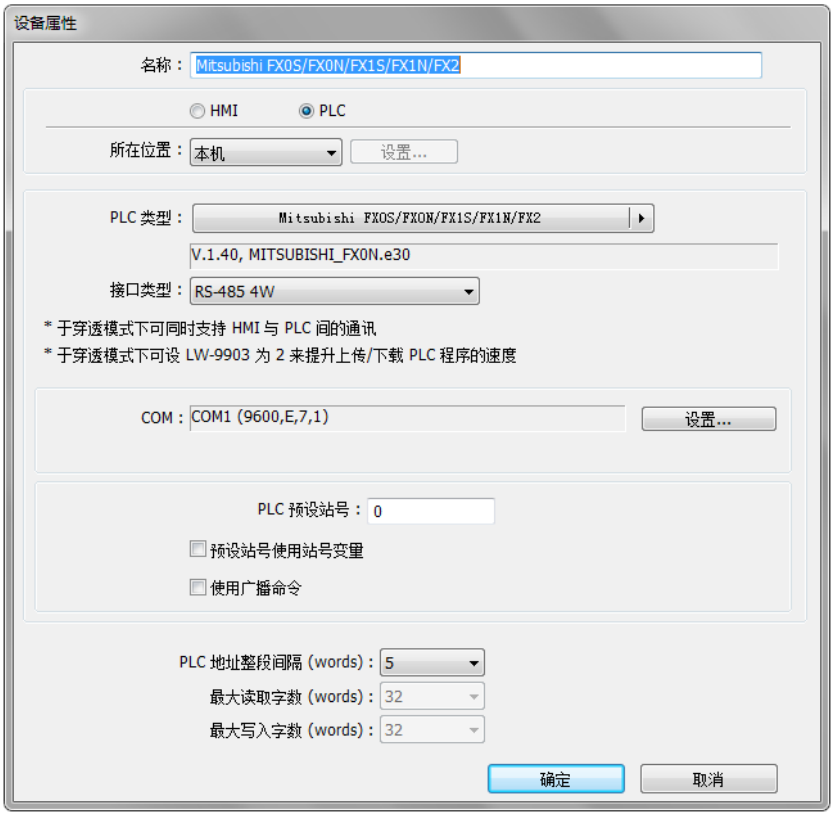
5.2 设备列表

用来设置与 HMI 连接的各项设备的属性，这些设备包括各种 PLC、远端的 HMI 和 PC。当开启新的 EMTP 文件后，会预设一个“Local HMI”设备，被用来识别本机，也可称为本地 HMI。若要修改设备内容，点击 [系统参数设置] 对话框的 [设置]，即可弹出 [设备属性] 对话框。

5.2.1 如何控制一台本机 PLC

所谓“本机 PLC”是指直接连接在本地 HMI 上的 PLC，若要控制本机 PLC 时，需先新增此种类型的设备。点击 [系统参数设置] 对话框中的 [新增]按钮后，即可弹出 [设备属性] 对话框窗口。

以 Mitsubishi FX0s/FX0n/FX1s/FX1n/FX2 为本机 PLC 为例：



设置	描述
名称	设备名称。
HMI 或 PLC	此范例的连接设备为 PLC，所以此时选择 [PLC]。
所在位置	可以选择 [本机] 或 [远端]，因为此范例 PLC 连接在本机 HMI 上，所以选择 [本机]。

PLC 类型	选择 PLC 的型号。
接口类型	PLC 所使用的接口类型，可以选择 [RS-232]、[RS-485 2W]、[RS-485 4W]、[以太网]、[USB] 以及 [CAN Bus]。 ● 接口类型如果为 [RS-232]、[RS-485 2W]、[RS-485 4W]，按下 [设备属性] 对话框中的 [设置]，可弹出[通讯端口设置] 对话框，并设置正确的通讯参数。  **超时** 通讯中断超过此项设置值 (单位为秒)，HMI 会使用 5 号窗口 作为 “PLC No Response” 为提示。 **通讯延时** HMI 在送出下一个命令给 PLC 前，会刻意先延迟此项设置值 (单位为毫秒)，再送出命令。此项设置值会降低 HMI 与 PLC 间的通讯效率，因此若无特殊需求，设置为 “0” 即可。 注意：若使用的 PLC 为 Siemens S7-200 系列，则不能忽略此项设置值，建议将 [通讯延时] 设置为 “5”，[ACK 讯号延时] 设置为 “30”。 ● 接口类型如果为 [以太网]，点击 [设备属性] 对话框中的 [IP 地址设置] 后可以弹出 [IP 地址设置] 对话框，用户必须正确设置 PLC 的 IP 地址与 TCP 端口号。

- 接口类型如果为 [USB]，则不需再做进一步的设置，请检查 [设备属性] 上的各设置值是否正确。
- 接口类型如果为 [CAN Bus]，请参照 《PLC 连接手册》 中关于 “CANopen” 的说明，并导入 eds 文件。

PLC 预设站号

PLC 所使用的预设地址站号。当地址内容不包括站号信息时，将使用此项设置值做为 PLC 的站号。

另外，也可将 PLC 站号信息直接设置在地址内容中，此时所使用的地址格式为 ABC#DEFGH

其中 ABC 表示 PLC 所使用的站号，此时 ABC 必须大于等于 0，且小于等于 255。DEFGH 用来指定 PLC 的地址，两个数据之间以 “#” 做为区隔。如下图所示，显示此时将读取 PLC 站号为 1 的 PLC 的 T-20 地址的内容。

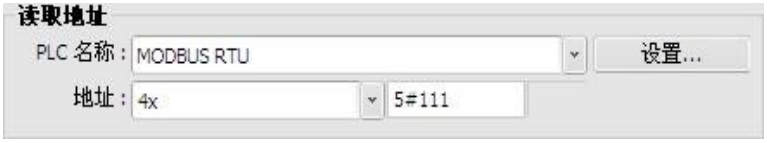
预设站号使用站号变量

设备列表中 PLC 属性的 PLC 预设站号可以使用站号变量。利用 LW-10000~LW-10015 (var 0 ~ var 15) 来设置站号变量。当 PLC 预设站号选择使用站号变量时，若在 PLC 地址中未指定所使用的站号，则站号一律由预设站号所指定的站号变量来决定。

例如 PLC 预设站号已选择使用 var3：

以下使用几个例子说明。

- 所操作的 PLC 站号为 5



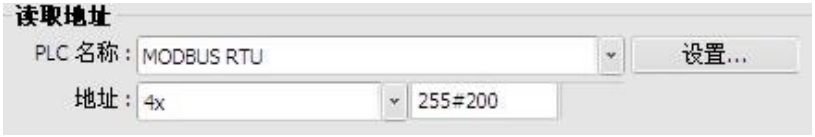
- 所操作的 PLC 站号由 var7 (LW-10007) 来决定

- PLC 的地址为“111”，此时不指定 PLC 站号，但因为已使用预设站号 var3，所以所操作的 PLC 站号由 var3 (LW-10003) 来决定。

使用广播命令

当勾选 [使用广播命令] 后，根据所使用的 PLC 的广播站号，填入到 [广播命令所使用的站号]。当人机使用广播站号发送命令时，PLC 将只接收命令而不回复人机。

如下图所示：



假设广播站号为 255 ，当 HMI 发送命令至 255#200 这个地址时，所有的 PLC 会接收这个命令但不回复 HMI 命令。

有支持广播命令的 PLC 才适用此功能。

PLC 地址段间隔 (words)	不同读取命令的读取地址间距若小于此项设置值，这些命令可以合并为同一个命令。此项设置值如果为“0”，将取消命令合并功能。 举例来说，若此项设置值为“5”，当分别从 LW-3 读取 1 个 word 与从 LW-6 读取 2 个 word 的数据 (即 LW-6 与 LW-7 的内容)时，因 LW-3 与 LW-6 的地址差距小于 5 ，此时可以将两个命令合并为 1 个命令，合并后的命令内容为从 LW-3 开始连续读取 5 个 word 的数据(读取 LW-3 ~ LW-7)。需注意，可以被合并的命令读取数据大小将不会大于 [最大读取字数 (words)]。
最大读取字数 (words)	一次可以从设备读取数据的最大量，单位为 word。
最大写入字数 (words)	一次可以写入到设备的数据最大量，单位为 word。

完成上述的各项设置后，在 [设备列表] 中可以发现新增了一个名称为 “本机 PLC 1”的设备。

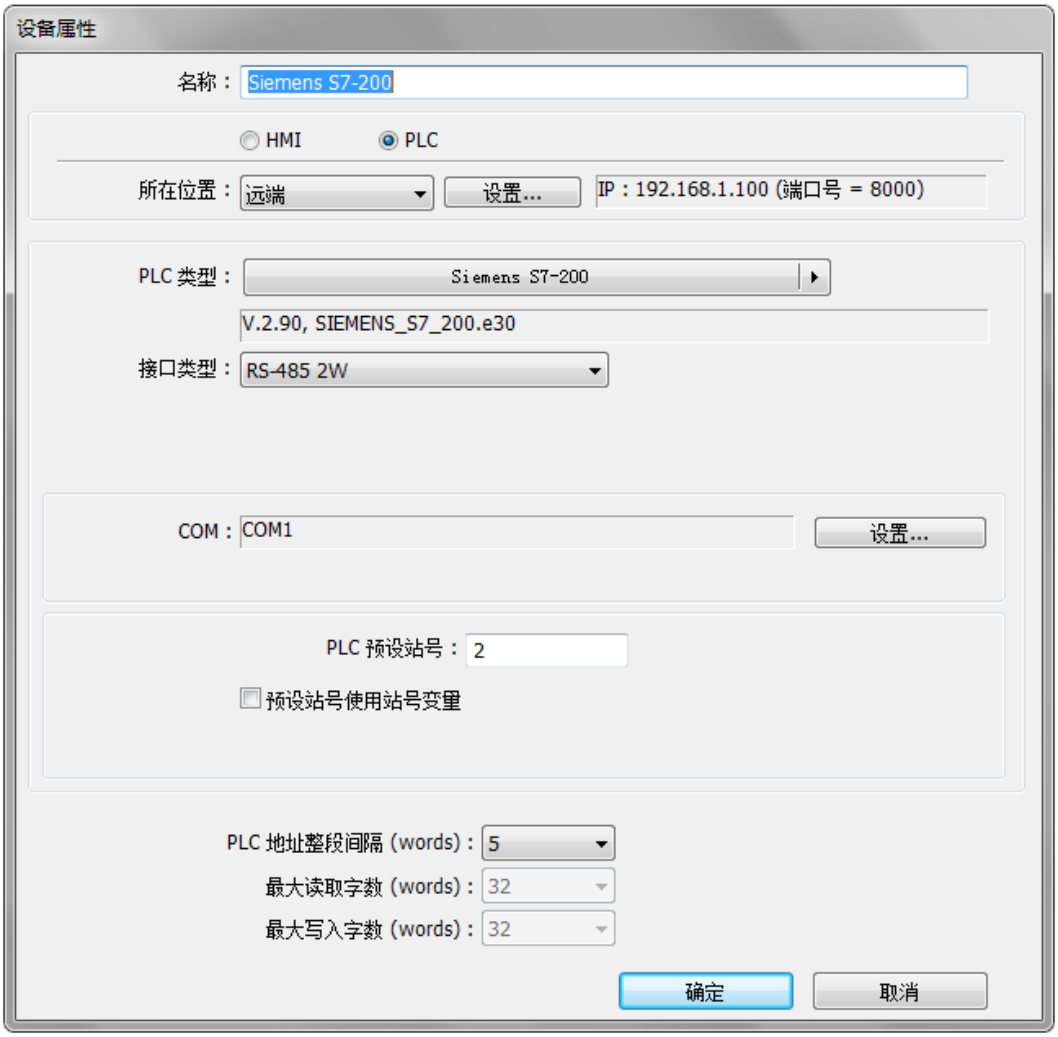
Note

使用 cMT-SVR 时，可以在 [系统参数设置] 对话框中，选择“本机 HMI”，按下 [新增 PLC] 按钮，便会在“本机 HMI”之下，新增一个“本机 PLC 1”，如下图所示。

5.2.2 如何控制一台远端 PLC

所谓“远端 PLC” 是指直接连接在远端 HMI 的 PLC。若要控制远端 PLC 需先增加此种类型的设备。在点击[系统参数设置] 对话框的 [新增]，即可在弹出的 [设备属性] 对话框设置各项属性。

以远端 Siemens S7-200 PLC 为例，如下图所示：



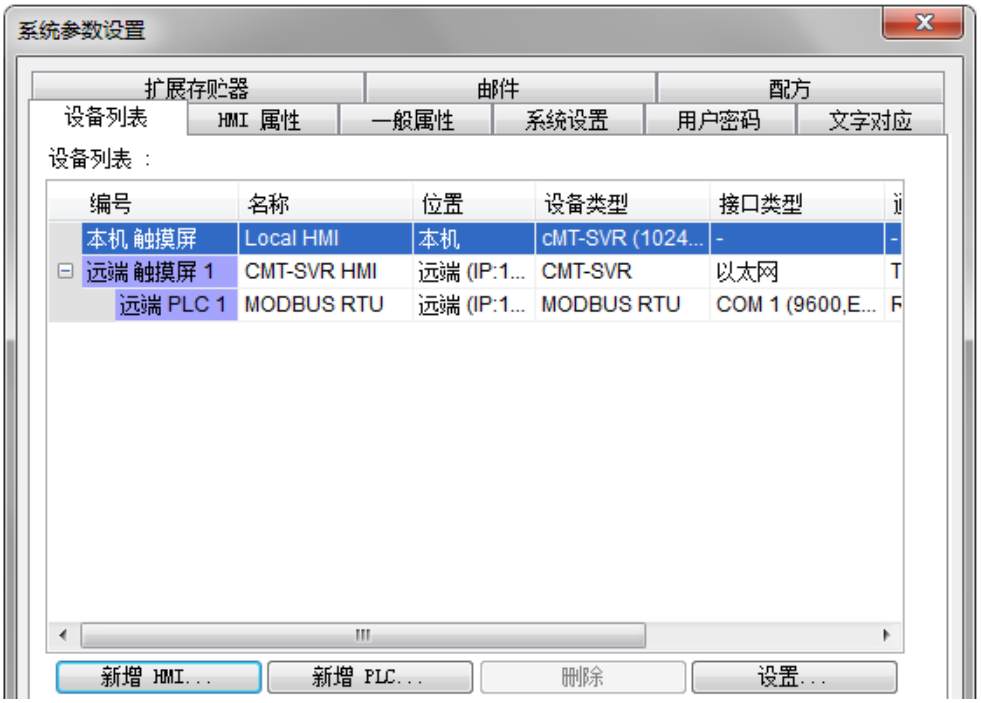
设置	描述
HMI 或 PLC	连接设备为 PLC，所以此时选择 [PLC]。
所在位置	可以选择 [本机] 或 [远端]。因 PLC 连接在远端 HMI 上，所以此时选择 [远端]，并且设置远端 HMI 的 IP 地址及连接端口。请在 [设备属性] 对话框按下 [所在位置] 旁的 [设置]。 

PLC 种类	选择远端 PLC 的型号。
PLC 接口	远端 PLC 所使用的接口类型，若远端 PLC 使用串行端口时，接口需选择 [RS-232]，[RS-485 2W]，[RS485 4W] 任一种。
COM	远端 PLC 连接远端 HMI 时所使用的串行端口。此项内容必须正确设置。
PLC 预设站号	远端 PLC 所使用的站号。

完成上述的各项设置后，在 [设备清单] 中可以发现新增了一个名称为 “远端 PLC 1” 的设备。

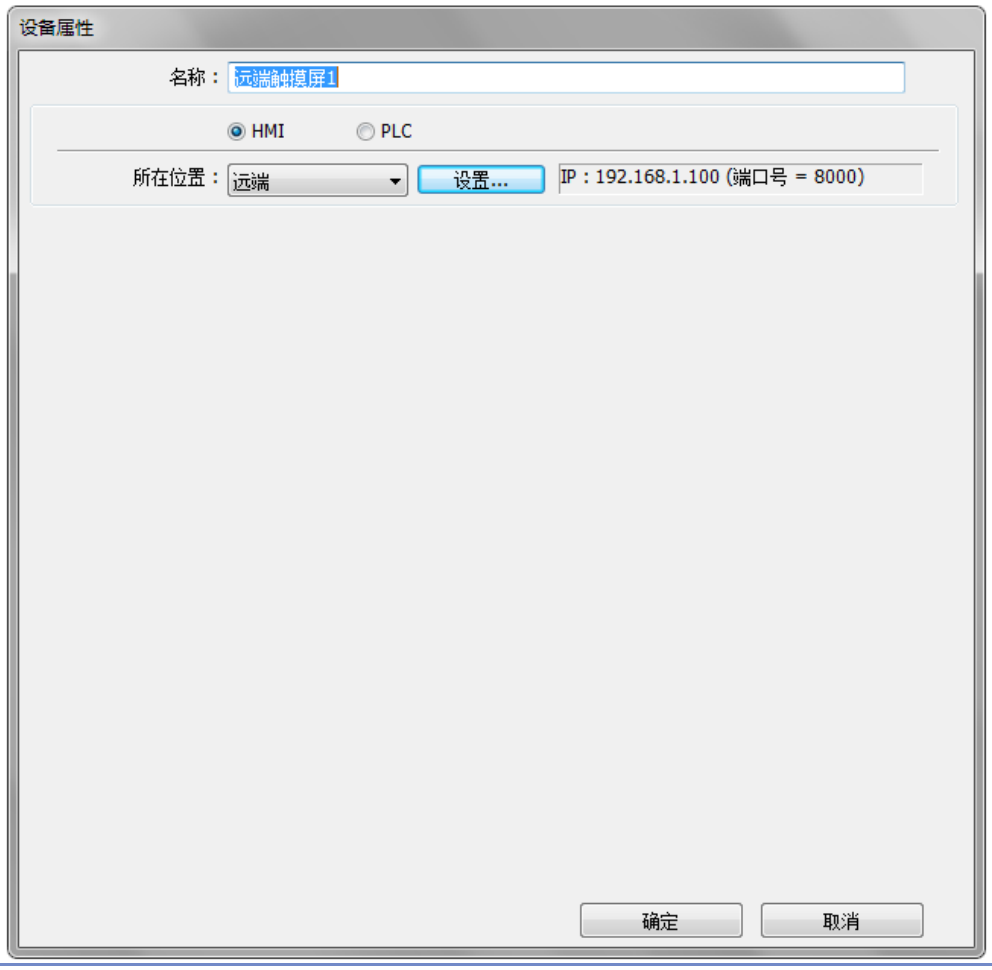
Note

使用 cMT-SVR 时，可以在 [系统参数设置] 对话框中，选择已建立的 “远端 HMI 1”，按下 [新增 PLC] 按钮，便会在 “远端 HMI 1” 之下，新增一个 “远端 PLC 1”，如下图所示。



5.2.3 如何控制一台远端 HMI

所谓“远端 HMI” 是指非本地 HMI 的所有其它 HMI，PC 也被视为远端 HMI 的一种。若要控制远端 HMI 需先新增此种类型的设备。在点击 [系统参数设置] 对话框的 [新增]，在弹出的[设备属性] 对话框设置各项属性即可。



设置	描述
HMI 或 PLC	连接设备为 HMI，所以此时选择 [HMI]。
所在位置	可以选择 [本机] 或 [远端]，因为使用远端 HMI，此时选择 [远端]，并且必须设置远端 HMI 的 IP 地址及连接端口。请在 [设备属性] 对话框按下 [设置]。

完成上述的各项设置后，在 [设备清单] 即新增了一个为“远端触摸屏 1”的设备。

Note

使用 cMT-SVR 时，可以在 [系统参数设置] 对话框中，按下 [新增 HMI] 按钮，便会新增一个“远端触摸屏 1”，如下图所示。

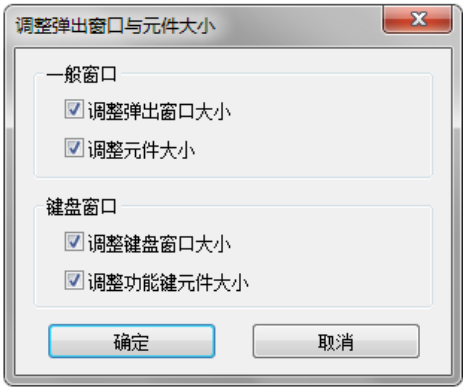
5.3 HMI 属性

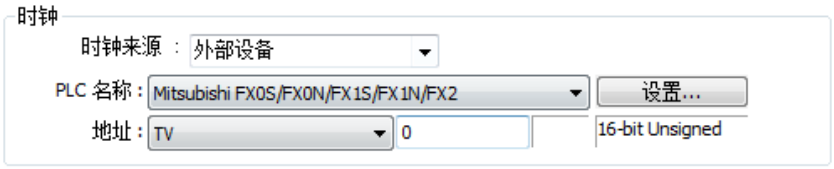
[HMI 属性] 设置页用来设置 [HMI 型号]、[时钟来源]、[打印机]、以及 [卷动轴] 宽度。

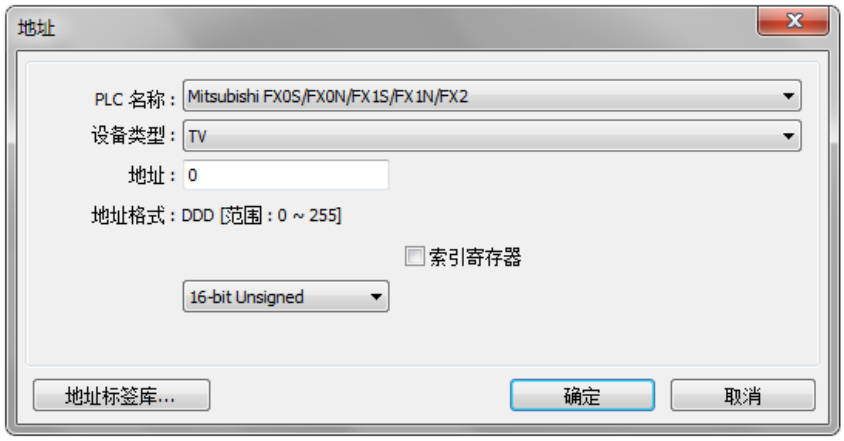
Note

使用 cMT-SVR 时，只需设置 [HMI 型号]、[HMI 站号]、与 [端口号]。

设置	描述
HMI 型号	请选择要使用的 HMI 型号。 当用户更换其它的 HMI 型号并按确认键时，会弹出 [调整弹出窗口与元件大小] 对话框如下图所示。按下确认后即完成变更 HMI 型号的设置。



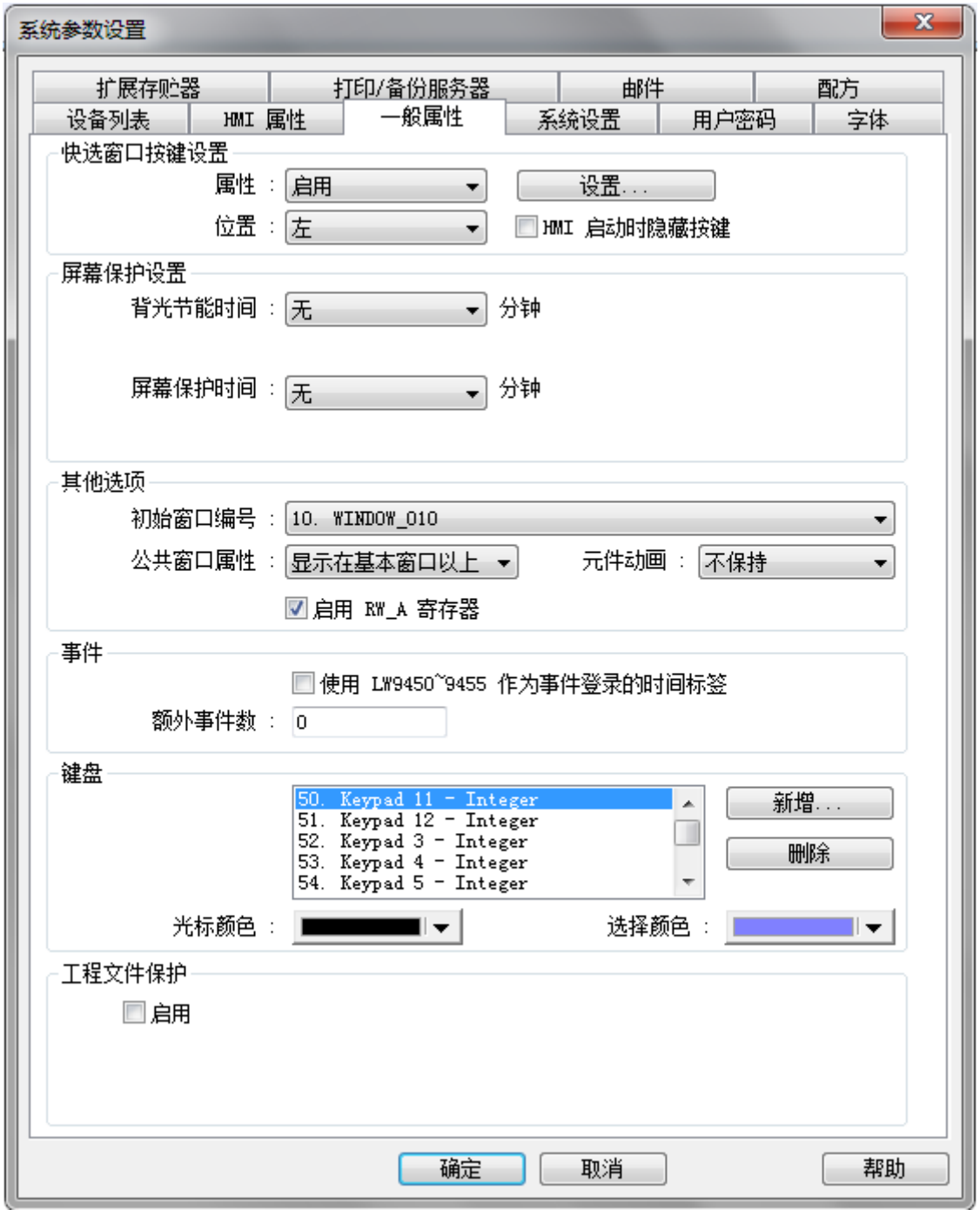
HMI 站号	选择 HMI 所使用的站号，若无特殊目的，采用默认值即可。
端口号	设置 HMI 所使用的通讯端口号，此号码也用于 MODBUS 服务器，若无特殊目的，采用默认值即可。
时钟	**时钟来源** 设置时间信号的来源，可用在 [资料取样]，[事件登录] 等需时间记录的元件上。 ● 当选择 [触摸屏实时时钟] 时，表示时间信号来自 HMI 上内含的时钟。 ● 当选择 [外部设备] 时，表示时间信号来自外部设备，此时需正确设置时间信号的来源地址。以下图所示的设置为例，表示时间来自 “本机 PLC” 的 TV 地址，此时地址 TV-0 开始的连续 6 个 word 内分别存放下列信息： TV-0 -> 秒 (限制范围： 0~59) TV-1 -> 分 (限制范围： 0~59) TV-2 -> 时 (限制范围： 0~23) TV-3 -> 日 (限制范围： 1~31) TV-4 -> 月 (限制范围： 1~12) TV-5 -> 年 (限制范围： 1970~2037) 



打印机	**型号** 显示目前支持的打印机类型，其中 HP PCL Series 需使用 USB 接口连接，其它类型打印机需使用串行端口连接。 详细信息请参考《第二十三章 HMI 支持的打印机型号》。 使用串行端口连接的打印机需正确设置串行端口的通讯参数。当打印机的型号为 [SP-M, D, E,F] 时，需设置 [每行点数]，此设置值不可超过打印机每行可以打印的点数，否则将造成错误的打印结果。
卷动轴	可设置卷动轴的尺寸。当元件的尺寸不足以显示其内容时，卷动轴将显示于该元件上。

5.4 一般属性

[一般属性] 设置页用来设置与画面操作有关的各项属性。



设置	描述
快选窗口按键设置	设置快选窗口（3 号窗口）的各项属性，若要使用快选窗口前，需先建立 3 号窗口。

	属性 选择是否使用快选窗口，在选择 [启用] 后，可以点击 [设置] 功能，设置快选窗口按键的各项属性，包括标示于快选窗口按键上的颜色与文字。 **位置** 选择快选窗口按钮的出现位置，选择 [左] 则快选窗口按钮出现在画面的左下角；选择 [右] 则快选窗口按钮出现在画面的右下角。
屏幕保护设置	**背光节能时间** 当未碰触屏幕的持续时间等于此设置值时，将关闭背光灯，设置的时间单位为分钟。关闭背光灯后只需碰触到屏幕，即可重开背光灯。当设置值选择 [无] 时，HMI 将不使用背光节能的功能。 **屏幕保护时间** 当未碰触屏幕的持续时间等于此设置值时，将自动切换到 [屏幕保护使用窗口] 所指定的窗口，设置的时间单位为分钟。当设置值选择 [无] 时，HMI 将不使用屏幕保护的功能。 **屏幕保护使用窗口** 指定屏幕保护功能执行时要切换的页面。
其它选项	**初始窗口编号** 选择 HMI 开机后的起始页面。 **公共窗口属性** 公共窗口 (4 号窗口) 内的元件会出现在每个基本窗口中，此选项可设置公共窗口内的元件，是出现在基本窗口原来元件的上层或下层。 **元件动画** 如果选择 [保持] 模式，则 HMI 在运作时，[动画] 与 [移动图形] 元件将显示在其它类型元件的上方，与元件的建立顺序无关。如果选择 [不保持] 模式，则元件的显示顺序依照元件建立的次序先后，先建立者先显示。 **启用 RW_A 寄存器** 可勾选是否启用配方资料 RW_A。在启用 RW_A 后，元件才可以操作 RW_A 的内容。RW_A 的大小为 64K。
事件	**额外事件数** 系统预设的事件记录总数为 1000 笔，若要增加记录的笔数，可以更改此设置值，目前设置的上限为 10000。
键盘	显示键盘存在的页面，这些页面代表在使用 [数值输入] 与 [字符输入] 元件时，可以选择的键盘类型，最多可以新增到 32 个键盘。用户要建立自订的键盘时，需先在已存在的页面规划好要使用的键盘，并使用 [新增] 功能选择这些页面并加入到列表中即可。

	详细信息请参考《第十二章 键盘的设计与使用》。 **光标颜色** 设置使用 [数值输入] 或 [字符输入] 元件时，输入字段中出现的光标颜色。
工程文件保护	用户的工程文件可被限定在特定的 HMI 上执行。 详细信息请参考《第三十章 工程文件保护功能》。

Note

- cMT-SVR 系列不支持 [快选窗口按键设置] 与 [键盘] 等相关设置。

5.5 系统设置

[系统设置] 控制 EasyBuilder Pro 中各种功能。

以上有些功能从系统寄存器复制而来，例如：[隐藏系统设置工具条 (LB-9020)]、[隐藏鼠标光标 (LB-9018)]、[取消声音导出功能 (LB-9019)]、[禁止远端 HMI 连接 (LB-9044)] 以及 [取消上传功能 (LB-9033)]，用户可以选择系统寄存器来使用这些功能。

要选择系统寄存器，用户可在新增元件时，于设置页勾选 [系统寄存器] 并选择 [设备类型]。

要检视所有的系统寄存器，可于 EasyBuilder Pro 点选 [图库] » [地址标签库] » [系统寄存器]。

设置	描述
重新下载工程文件后所显示的语言	选择在重新下载工程文件并启动 HMI 时所显示的语言。
开机后使用初始化宏指令	指定当 HMI 开机后所执行的宏指令。
自动注销	当用户没有操作 HMI 的时间等于此设置值，人机中有设置安全等级的元件将无法使用，需要再次输入用户 ID 及密码才能操作元件。
隐藏系统设置工具条	将 HMI 右下角系统设置列隐藏。
隐藏鼠标光标	将 HMI 上鼠标光标隐藏。
取消声音导出功能	将 HMI 声音关闭。
禁止远端 HMI 连接	将远端 HMI 连接功能关闭，远端 HMI 将无法操作本机 HMI。
取消上传功能(重新启动 HMI 后生效)(或设 LB9033 为 ON)	将 HMI 上传工程文件功能关闭，勾选此选项并下载工程文件后，必须重启 HMI 才会生效。
禁止密码远端读取操作(或设 LB9053 为 ON)	禁止远端 HMI 读取本机 HMI 的密码。
禁止密码远端写入操作(或设 LB9054 为 ON)	禁止远端 HMI 写入本机 HMI 的密码。
当与 PLC 通讯失败时，显示断线图标在相关的元件上	选择当与 PLC 通讯失败时，是否在相关元件上显示断线图标  当使用此项功能且元件无法与 PLC 通讯时，断线图标会显示在元件右下角。 
VNC 服务器	设置登入 VNC 服务器所使用的密码。
LW 保护 **RW 保护**	如果用户勾选 [禁止 LW/RW 远端写入]，并在 [LW/RW 范围] 内设置所要保护的范围，此范围内的数值将不能透过远端 HMI 调整。
Easy Access 服务器	透过这项技术，用户可以透过网络监测网络上连结的 HMI，并直接在电脑上操作，就像直接将 HMI 拿在手上一样。 特别的是，Easy Access 不需传输更新的图片，只需传输实时资料。这使得传输更快更有效率。详细的信息请参阅《Easy Access》说明。

Note

- cMT-SVR 系列不支持 VNC 服务器相关设置。

5.6 用户密码

系统参数中的[用户密码] 设置页用来设置用户的密码以及可操控的元件类别。共有两种认证模式：一般模式和进阶安全模式。

详细信息请参考《第十章 元件安全防护》。

5.6.1 一般模式

可设置 12 组用户密码。密码须为非负整数。

系统运行时，用户在成功输入密码后，系统会依照用户的设置内容决定用户可以操作的元件类别。在工程文件中，元件的类别被区分为 [类别 A] 至 [类别 F] 共 6 种。

类别属于 [无] 的元件，开放给所有用户使用。

当 “用户 3”的设置内容如上时，则此用户只被允许使用类别属于 “无”与 A、B、C 的元件。

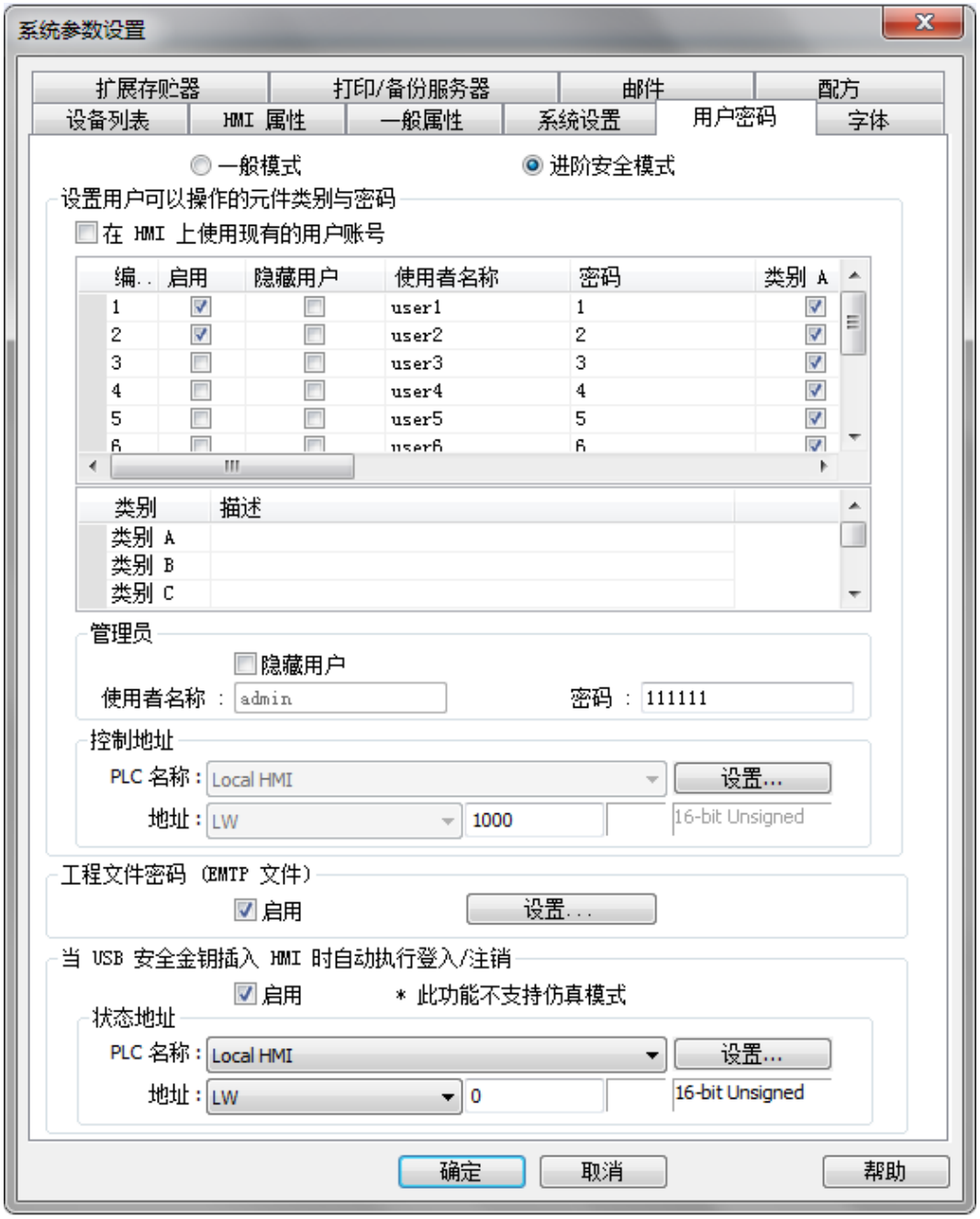
用户可以设置 [工程文件密码 (EMTP 文件)]。若有设此密码，当用户想要编辑 .emtp 文件时，必须输入此密码才能打开编辑。

勾选 [启用] 再按下 [设置] 就可以设置密码。

完成设置后，在编辑工程文件前，将要求用户输入密码，输入正确才能进入工程文件。

5.6.2 进阶安全模式

进阶安全模式可规划的用户数为 11 组，另外提供 [管理员] 使用模式，此管理员有最大使用权，任何元件的安全等级皆可操作。不同的用户密码可由英文字母或数字所组成，并可规划每个用户可操作的元件类别分为 [类别 A] 至 [类别 L] 等共 12 个类别。



设置	描述
设置用户可以操作的元件类别与密码	勾选 [在 HMI 上使用现有的用户账号] 时，会依照 HMI 上设置的用户账号，来辨别是否操作元件。
管理员	为内定管理员账号，不可被删除，且权限全开不得修改权限。 进阶安全模式可搭配 [项目选单] 元件来显示账号名称和权限。

	若勾选 [隐藏用户] 则账号名称和权限等数据不会显示在 [项目选单] 元件上。
控制地址	进阶安全模式提供一组 [控制地址] 供用户登入和管理账号。
当 USB 安全金钥插入 HMI 时自动执行登入/注销	勾选后可使用 USB 安全金钥自动登入/注销，而登入/注销的状态会自动写入指定的 [状态地址]。当 USB 插入时，将执行自动登入，当 USB 拔出时，将自动执行注销。状态地址的数值意义为： 0x00: 无动作 0x01: 登入成功 0x04: 登入失败 0x08: 注销成功 0x10: 注销失败

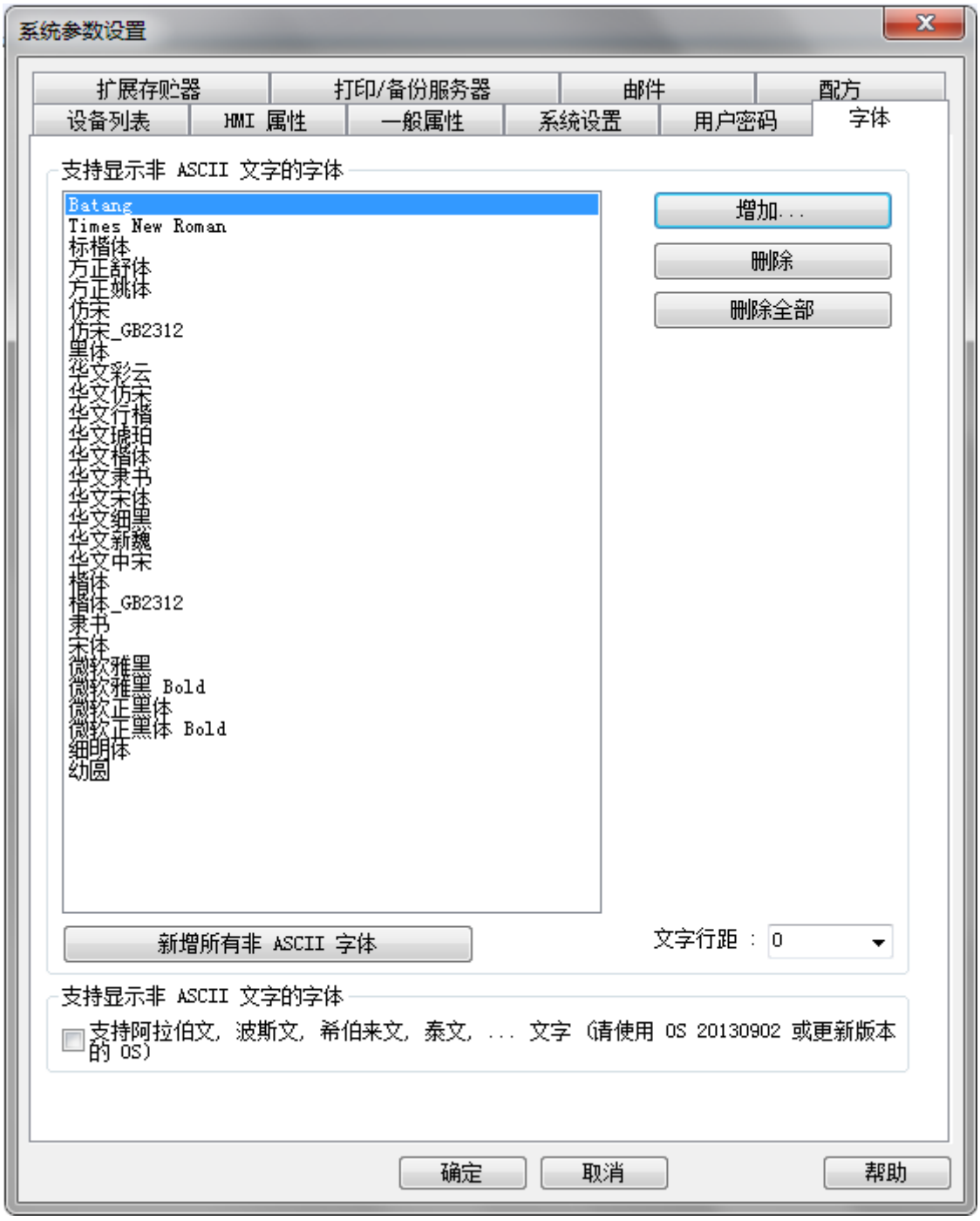
Note

cMT-SVR 仅支持 [进阶安全模式]，但不支持 [当 USB 安全金钥插入 HMI 时自动执行登入/注销] 相关设置。

5.7 字体或文字对应

5.7.1 eMT、iE、cMT-HD 系列

[字体] 设置页用来设置非 ASCII 文字所使用的字体。



在此项目表列出的字体为非 ASCII 的文字所使用的字体。当用户使用非 ASCII 的文字或双字符文字 (例如：简体中文字、繁体中文字、日文、韩文等)，且非表中的字体时，EasyBuilder Pro 会自动为这种文字选用项目表中的字体。

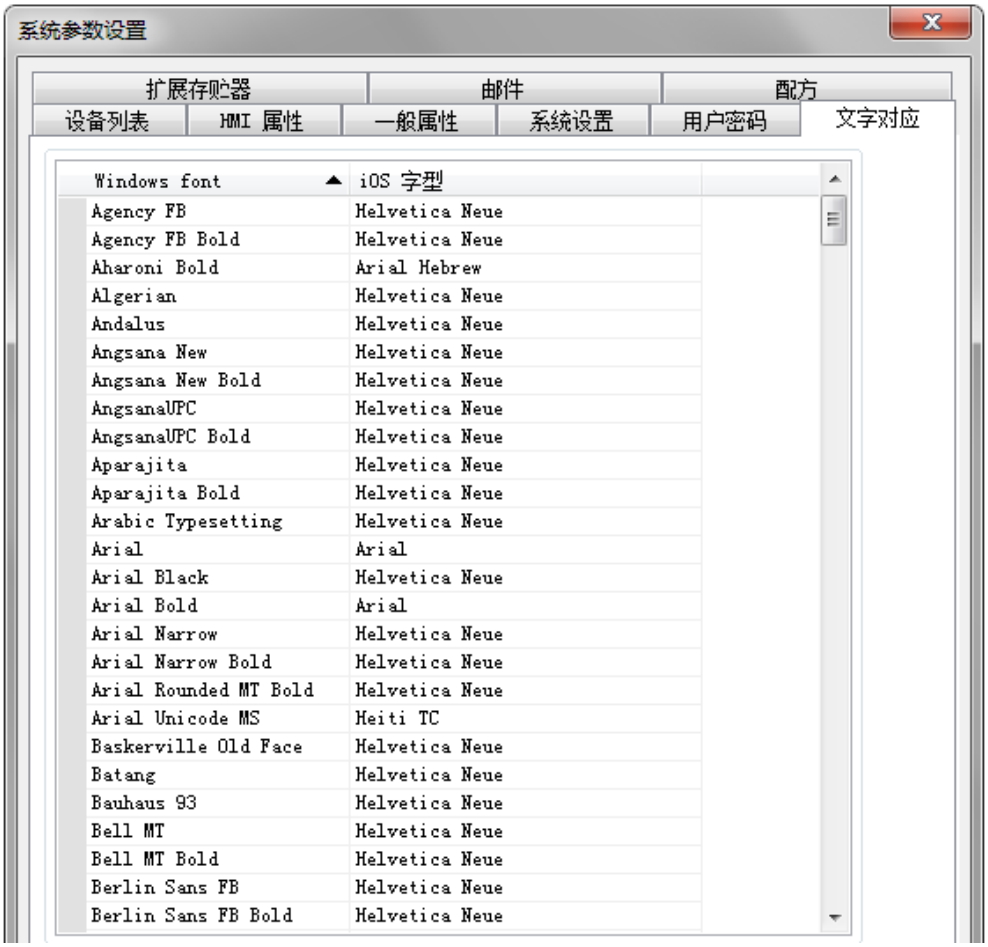
按下 [新增所有非 ASCII 字体] 可以将 WINDOWS 的非 ASCII 文字字体，新增到本项目表中。

用户也可以在 [文字行距] 调整行与行之间的间距。

勾选 [支持阿拉伯文、波斯文、希伯来文、泰文，...文字] 将可正确显示这些文字。

5.7.2 cMT-SVR 系列

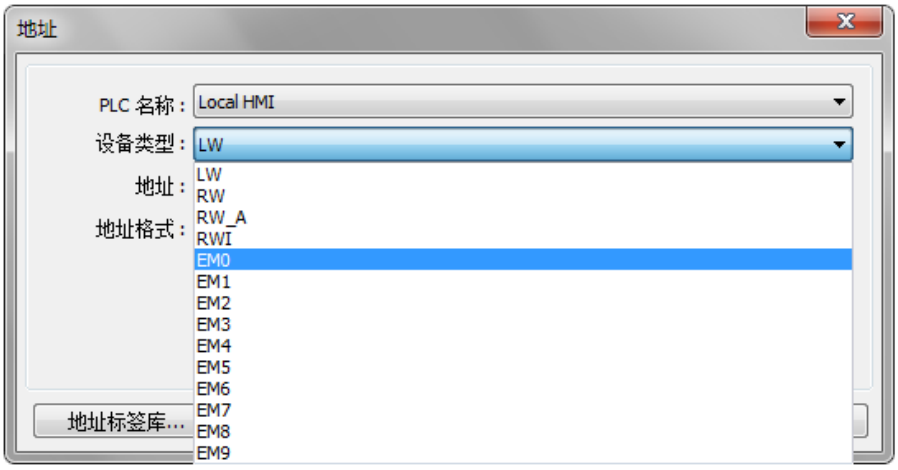
[文字对应] 设置页中列出在 WINDOWS 中使用的字体，所对应在 iPad 上可以显示的字体。



5.8 扩展内存

[扩展内存] 设置页用来设置扩展内存的位置。

扩展内存包含 EM0 ~ EM9，使用方式与其它 HMI 上的设备类型相似(类似使用 LW 或 RW 地址类型)，用户只需在[设备类型] 中指定使用 EM0~EM9 即可，每个扩展内存最多可以存放 2G word 的资料。



扩展内存中的数据使用文件的形式存放在 [SD 卡]、[U 盘] 上，[EM0] ~ [EM9] 所使用的文件名称分别为 em0.emi ~ em9.emi，用户可以使用 RecipeEditor.exe 开启这些文件并编辑扩展内存中的数据。

扩展内存中的数据不会因 HMI 断电而消失，也就是下次开机后，扩展内存中的数据会恢复为关机前的状态，与配方数据 (RW、RW _A) 类似。较特别的是用户可以指定扩展内存的位置，可以选择 [SD 卡]、[U 盘]。

当做为扩展内存的设备不存在时，若读取扩展内存中的数据，内容将一律为“0”；当作为扩展内存的设备不存在时，若要将数据写到扩展内存中，HMI 将会显示“PLC no response”反应。

HMI 在机器不需断电的情况下，可以随时插上或移除外接设备。用户可以利用这项特性，更新或撷取扩展内存中的数据。

5.9 打印/备份服务器

此设置页用来设置 MT 远端打印 / 备份服务器的相关数据。

设置	描述
导出设置	**方向** 设置文字或图片要 以[水平] 或 [垂直] 的方式输出。 **打印大小** 设置要照 [原始尺寸] 或 [配合打印机边界] 打印。

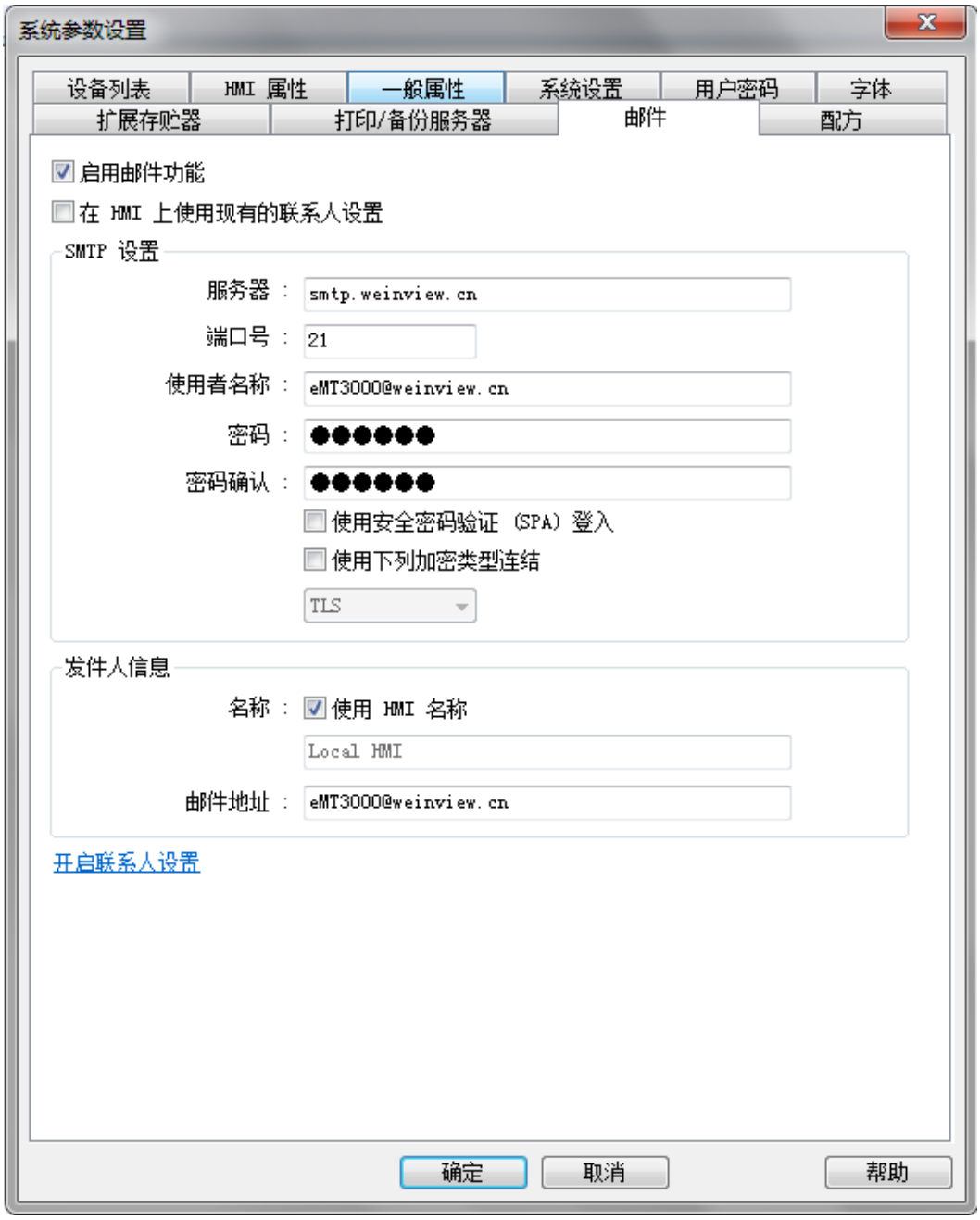
	边界 设置纸张上下左右的边界宽度。
通讯设置	**IP 地址** 透过网络指定打印机的 IP 地址。 **连接端口、用户名称、密码** 指定连接打印机的信息。 端口号可设置为 1~65535。 用户名称或密码最长可为 12 个字符。

详细信息请参考《第二十六章　EasyPrinter》。

5.10 邮件

[邮件] 设置页用来设置 e-Mail 的相关参数。

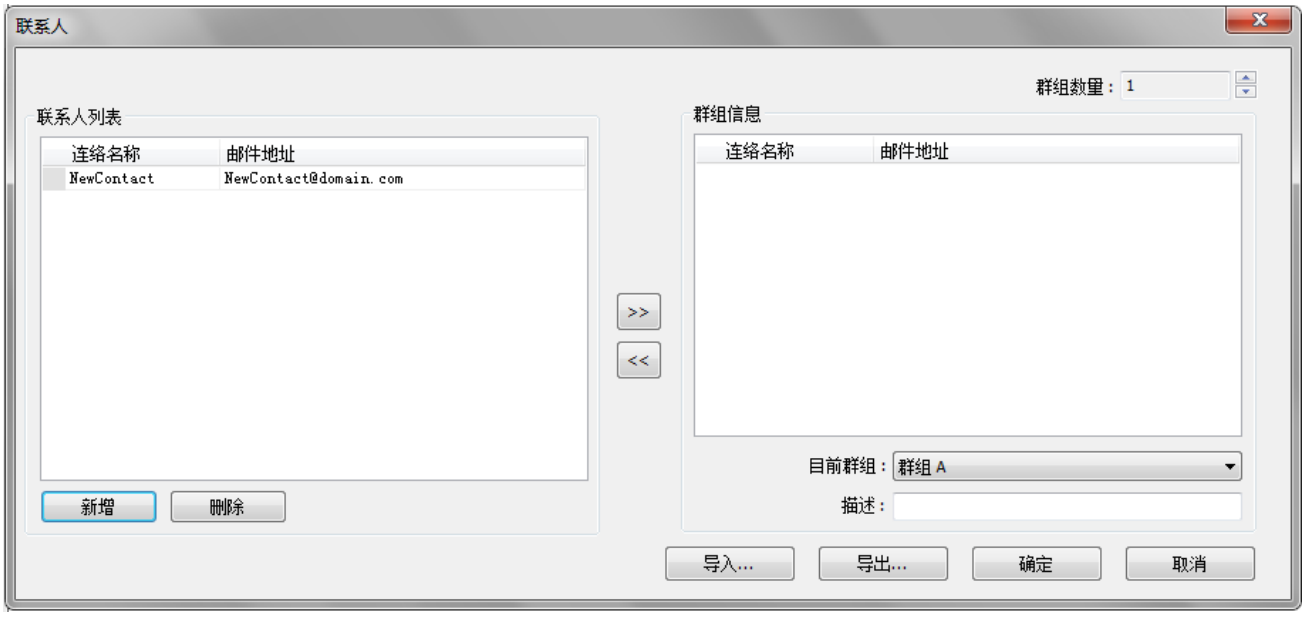
若勾选 [在 HMI 上使用现有的联系人设置]，会依照 HMI 上的联系人设置为主。



设置	描述
SMTP 设置	**服务器：** 设置邮件服务器。 **端口号：** 设置端口号。 **使用者名称：** 设置电子邮件地址。

	密码：设置邮件密码。 **确认密码：**确认设置的邮件密码。 **使用安全密码验证(SPA)登入：**邮件是否需要输入安全密码才能登入。 **使用下列加密类型连结：** 邮件发出时，选择是否需要加密联机。(TLS、SSL)
发件人信息	**名称** 可勾选使用 HMI 名称或自行输入发件人名称。 **邮件地址** 设置电子邮件地址。

按下 [开启联络人设置] 后：



设置	描述
联系人列表	新增或删除联系人清单。
群组信息	将多个联系人设置成群组。 **群组数量** 设置群组数量，群组名称依数量，依序命名为群组 A ~ P，最多共可设置 16 个群组。 **目前群组** 显示目前清单中的联系人所属群组。 **描述** 用户可以输入关于该群组的描述。

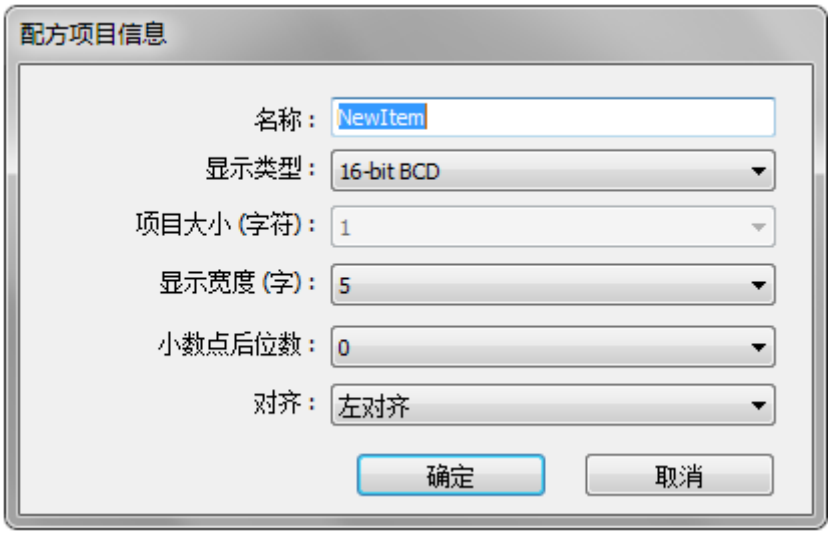
详细信息请参考《第七章 事件登录》。

5.11 配方

[配方] 设置页用来设置配方的相关数据。

设置	描述
配方列表	新增配方或删除配方。
新增	新增一笔新项目。
设置	各项目可以自行选择和修改，请见下方说明。
删除	选择要删除的项目，即可进行删除。

按下 [设置] 后：



设置	描述
名称	输入配方项目名称。
显示类型	设置项目的数据类型。
项目大小 (字符)	设置项目大小，单位为字符。
显示宽度 (字)	设置项目的显示字数。
小数点后位数	设置数据显示时的小数点后位数。
对齐	设置显示时的对齐方式，可选择左对齐、置中对齐、右对齐。

详细信息请参考《第二十四章 Recipe Editor》。

第六章 窗口

6.1 概要

窗口是 EasyBuilder Pro 工程软件里最重要的元素之一。所有需要显示在触摸屏上的各种元件、图形、文字等必须通过窗口才能呈现。EasyBuilder Pro 内建 1997 个窗口，其范围为窗口 3 至窗口 1999。

6.2 窗口类型

依照功能与使用方式的不同，可将窗口分为下列四种类型：

- 基本窗口
- 快选窗口
- 公共窗口
- 系统信息窗口

6.2.1 基本窗口

基本窗口是最常被使用的窗口类型，除了可当作主画面的用途之外，也被用在：

- 底层画面，可提供其它窗口作为背景画面。
- 键盘窗口。
- 功能键元件所选用的弹出窗口。
- 间接窗口与直接窗口元件所使用的弹出窗口。
- 屏幕保护窗口画面。

Note

- 由于基本窗口的尺寸大小必须与触摸屏显示屏幕相同，所以其分辨率设置也必须与所使用的触摸屏分辨率一致。

6.2.2 快选窗口

3 号窗口为预设的系统快选窗口，它可以与基本窗口同时存在，一般用来放置常用的工作按钮，位置为左下角或右下角。要使用快选窗口除了需先建立 3 号窗口外，还需设置快选窗口按钮的各项属性，快选窗口按钮的各项设置在 [系统参数设置] » [一般属性] 页中。除了可以使用快选窗口按钮来切换快选窗口的显示与隐藏之外，系统寄存器也提供以下地址来控制快选窗口：

[LB-9013] 快选窗口控制 [隐藏 (ON)/显示 (OFF)]

[LB-9014] 快选按键控制 [隐藏 (ON)/显示 (OFF)]

[LB-9015] 快选窗口/按键控制 [隐藏 (ON)/显示 (OFF)]

Note

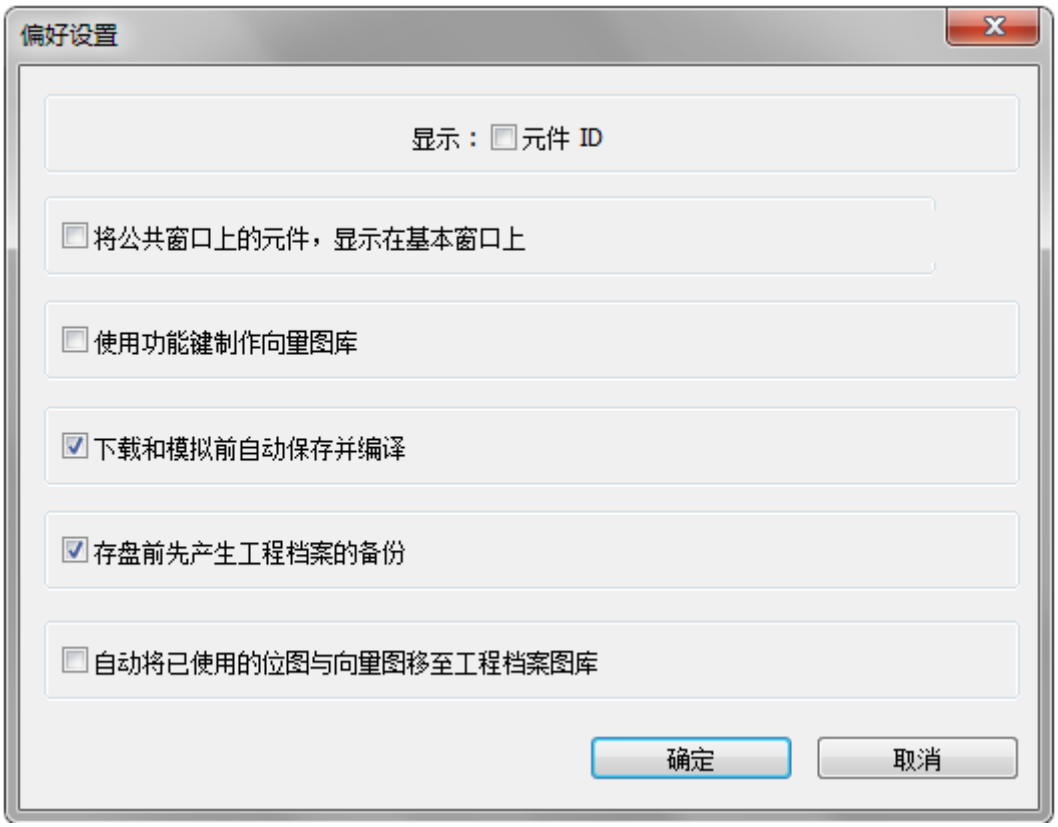
- cMT-SVR 机型不支持快选窗口功能。

6.2.3 公共窗口

4 号窗口为预设的公共窗口，此窗口中的元件也会出现在其它基本窗口中，但不包含弹出窗口，因此通常会将各窗口共享的元件放置在公共窗口中。

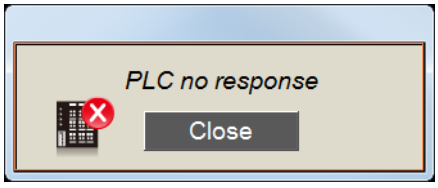
触摸屏上的程序运行时，可以使用功能键元件的 [切换公共窗口] 模式，在线更改公共窗口的来源。

在 [选项菜单] » [偏好设置] 可设置当编辑程序时，公共窗口上的元件是否会被显示于基本窗口。有了此预览功能，可避免当编辑程序时，将基本窗口的元件重叠到公共窗口的元件。



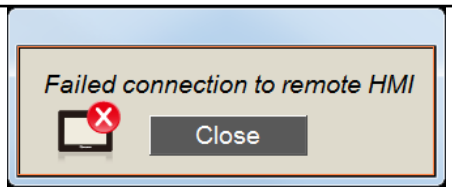
6.2.4 系统信息窗口

5、6、7、8 号窗口为预设的系统提示信息窗口：



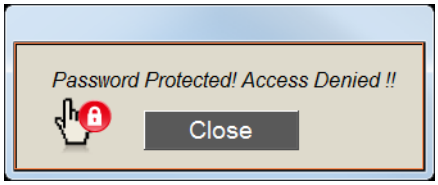
5 号窗口为 PLC Response 窗口

当触摸屏与 PLC 通讯中断时，系统将自动弹出 PLC no response 的警告窗口。可使用系统寄存器所提供的相关地址来禁止弹出此窗口。



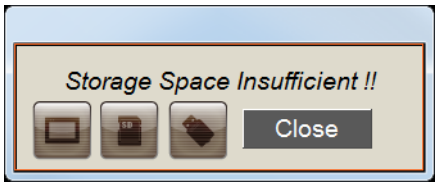
6 号窗口为 HMI Connection 窗口

当本地触摸屏无法连接到远端的 HMI 时，系统将自动弹出 Failed conncetion to remote HMI 的警告窗口。



7 号窗口为 Password Restriction 窗口

当用户无权限操作某元件时，可依设置决定是否弹出 Password Protected! Access Denied! 的警告窗口。



8 号窗口为 Storage Space Insufficient 窗口

当 HMI 内存、U 盘或 SD 卡上的可用空间不足以储存新的数据时，系统将自动弹出 Storage Space Insufficient 的警告窗口。(当系统检测到存储空间剩 4 MB 以下)

下列的系统寄存器可检视 HMI、U 盘或 SD 卡上目前可用的储存空间：

[LW-9072] HMI 目前的可用空间 (K bytes)

[LW-9074] SD 卡目前的可用空间 (K bytes)

[LW-9076] U 盘目前的可用空间 (K bytes)

并可利用下列的系统寄存器检视储存装置空间是否足够，若系统检测到内存剩 4 MB 以下，会将相关地址设为 ON：

[LB-9035] HMI 可用空间不足警示 (当状态为 ON)

[LB-9036] SD 卡可用空间不足警示 (当状态为 ON)

[LB-9037] U 盘可用空间不足警示 (当状态为 ON)

详细信息请参考《第二十二章 地址寄存器》。

窗口 5 ~ 窗口 8 的内容提示，可以根据实际需要来修改，以方便操作人员明白和识别故障信息。

Note

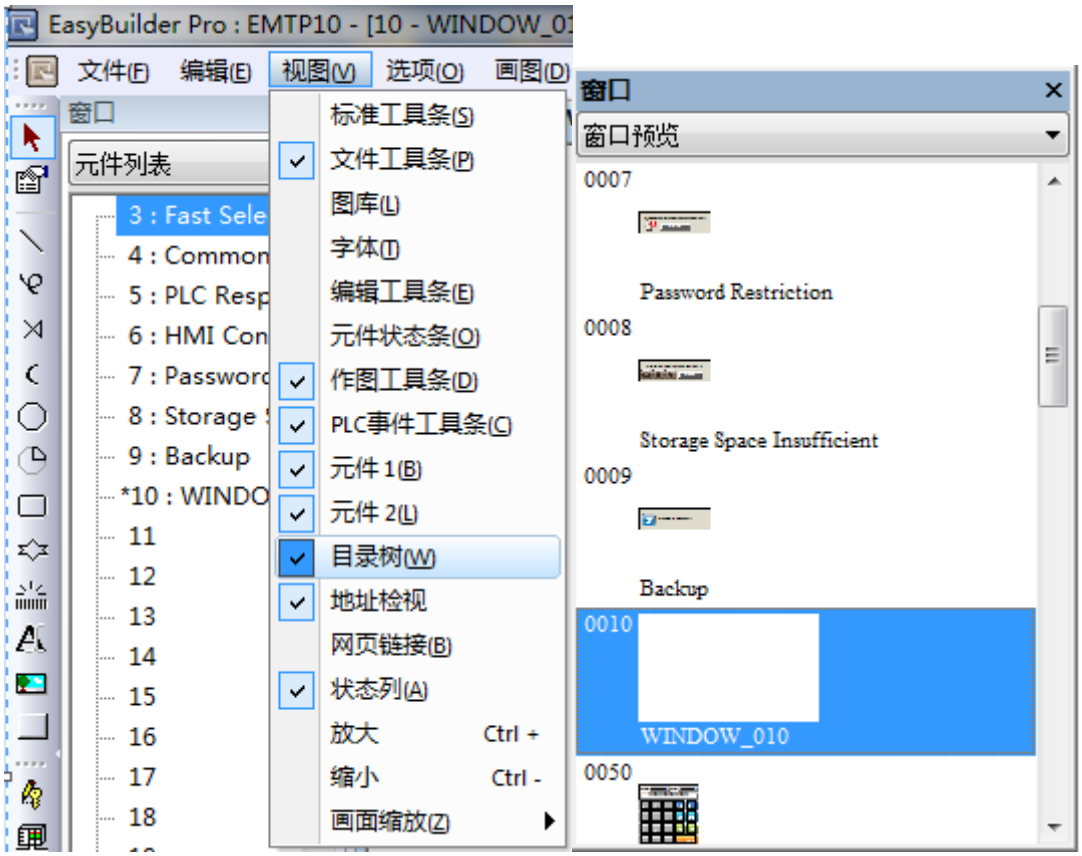
- 最多可同时开启 16 个弹出窗口，包含系统信息窗口、直接窗口和间接窗口。
- 系统不允许在一个基本窗口上使用 2 个直接(或间接) 窗口弹出同一个窗口。
- 窗口 3 ~ 窗口 9 为系统内部使用，窗口 10 ~ 1999 为用户可任意编辑操作窗口。
- cMT-SVR 机型同时只能开启 1 个弹出窗口。

6.3 窗口的建立、设置与删除

可通过 [检视菜单] » [目录树] 查看已建立的窗口。

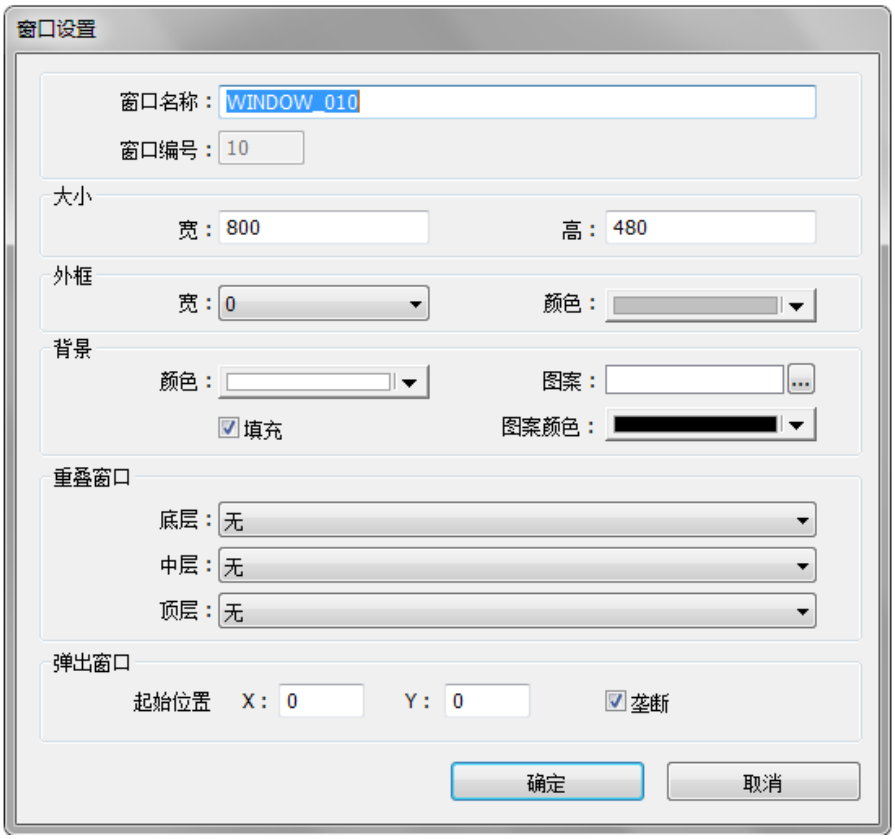
[元件列表] 显示窗口编号，已定义的窗口名称将被显示。目前已打开编辑的窗口编号前会有个 (*) 符号，按下窗口编号旁的 (+) 号可看到窗口含有哪些元件，包含元件 ID、地址与描述。

[窗口预览] 用整体窗口外型的小图来预览窗口。



6.3.1 窗口的建立与设置

在目录树 [元件列表] 模式下，选择欲建立的窗口编号后按下鼠标右键选择 [新增]。



设置	描述
窗口名称	输入的名称将出现在窗口的控制条，也会显示在目录树中。
窗口编号	由 3 ~ 1999。
大小	设置窗口的大小，一般基本窗口的分辨率与所选用的 HMI 的分辨率一样。
重叠窗口	重叠窗口功能可视为另一个额外的公共窗口，在设计程序时，同一元件可能被放置于许多的窗口中，但不是所有窗口时，便可使用重叠窗口。 每一个基本窗口最多可选择三个窗口作为背景，从 [底层] 开始到 [顶层] 结束，这些背景窗口内的元件将在基本窗口中依序出现的。
弹出窗口	基本窗口也可当作弹出窗口，[X] 与 [Y] 用来设置基本窗口在画面弹出的坐标位置。坐标原点为画面的左上角。
垄断	若勾选，此窗口的显示将独占所有的操作权。例如，当一个设置垄断的窗口弹出，所有弹出窗口与背景窗口的操作将完全冻结，直到垄断的窗口被关闭才可操作其它窗口。当基本窗口作为键盘窗口时，自动具有此属性。

Note

- 在重叠窗口中的元件无法从显示它们的基本窗口上编辑，若要编辑重叠窗口的元件，均需打开该窗口编辑。
- 若基本窗口所使用的重叠窗口编号与欲弹出的窗口编号相同，将无法弹出该窗口。

■ 当基本窗口与弹出窗口皆使用相同的重叠窗口时，弹出的窗口将无法显示重叠窗口上的元件。

或是在 [窗口] 选项 » [窗口列表] 选择 [新增]，并选择欲建立的窗口类型。

呼叫 [窗口设置] 对话窗有以下方式：

- 在目录树选择窗口编号，按下鼠标右键选择 [设置]。
- 在 [窗口菜单] » [窗口列表] 选择要设置的窗口后点击 [设置]。
- 在该窗口中，未选择任何元件时按下鼠标右键选择 [属性]。

6.3.2 窗口的开启、关闭或删除

开启已存在的窗口有以下方式：

- 双击目录树元件列表上的窗口编号。
- 在目录树元件列表上选择要打开的窗口后，按下鼠标右键选择 [打开]。
- 在 [窗口菜单] » [窗口列表] 选择要打开的窗口后选择 [开启]。

关闭或删除现有的窗口有以下方式：

- 在目录树元件列表上选择欲关闭或删除的窗口后，按下鼠标右键选择 [关闭] 或 [删除]。
- 在 [窗口菜单] » [窗口列表] 选择欲删除的窗口后选择 [删除]。

若要删除某一个窗口，必须先将其关闭才可以删除。

第七章 事件登录

7.1 概要

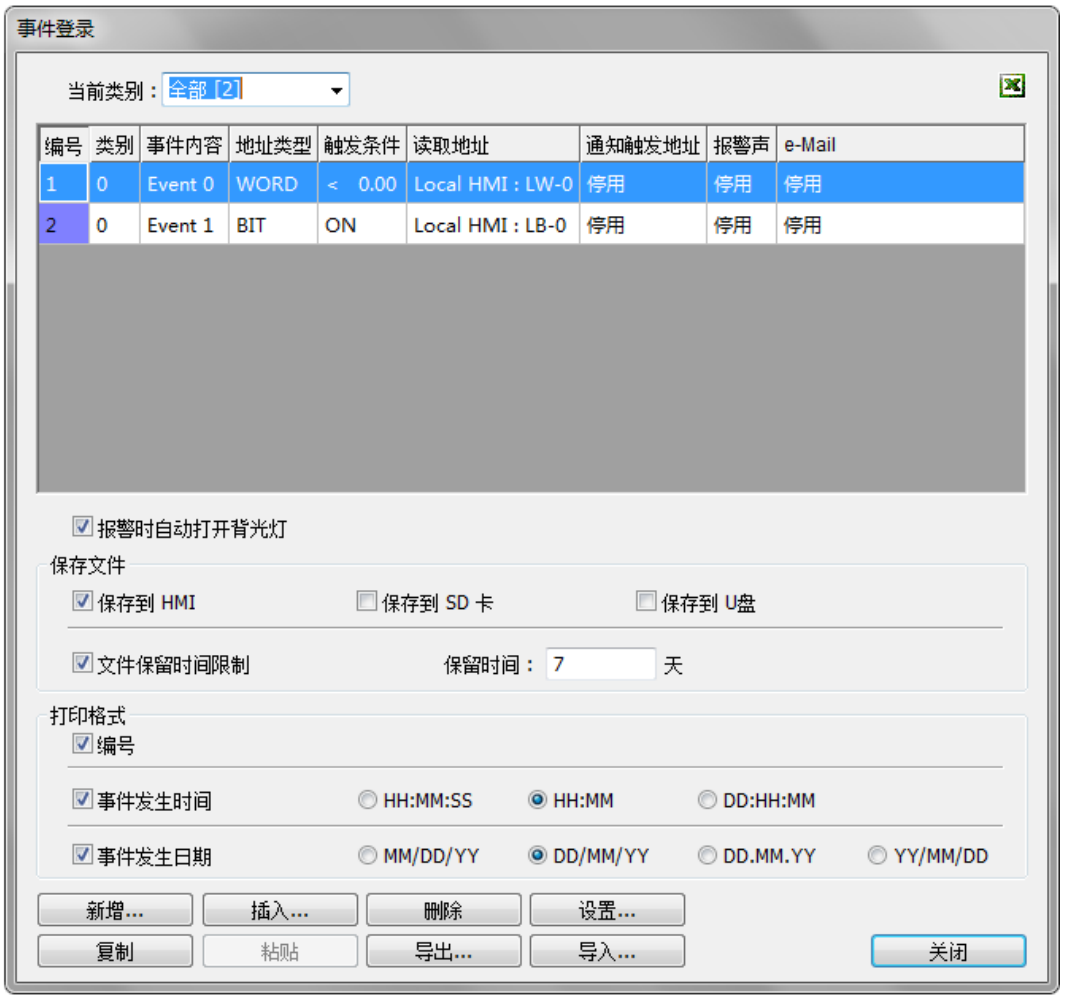
使用事件登录的基本程序如下：

1. 定义事件触发条件与内容。
2. 根据条件触发事件。
3. 可将事件记录储存到指定位置。
4. 可使用元件检视事件的完整处理周期。

7.2 事件登录管理

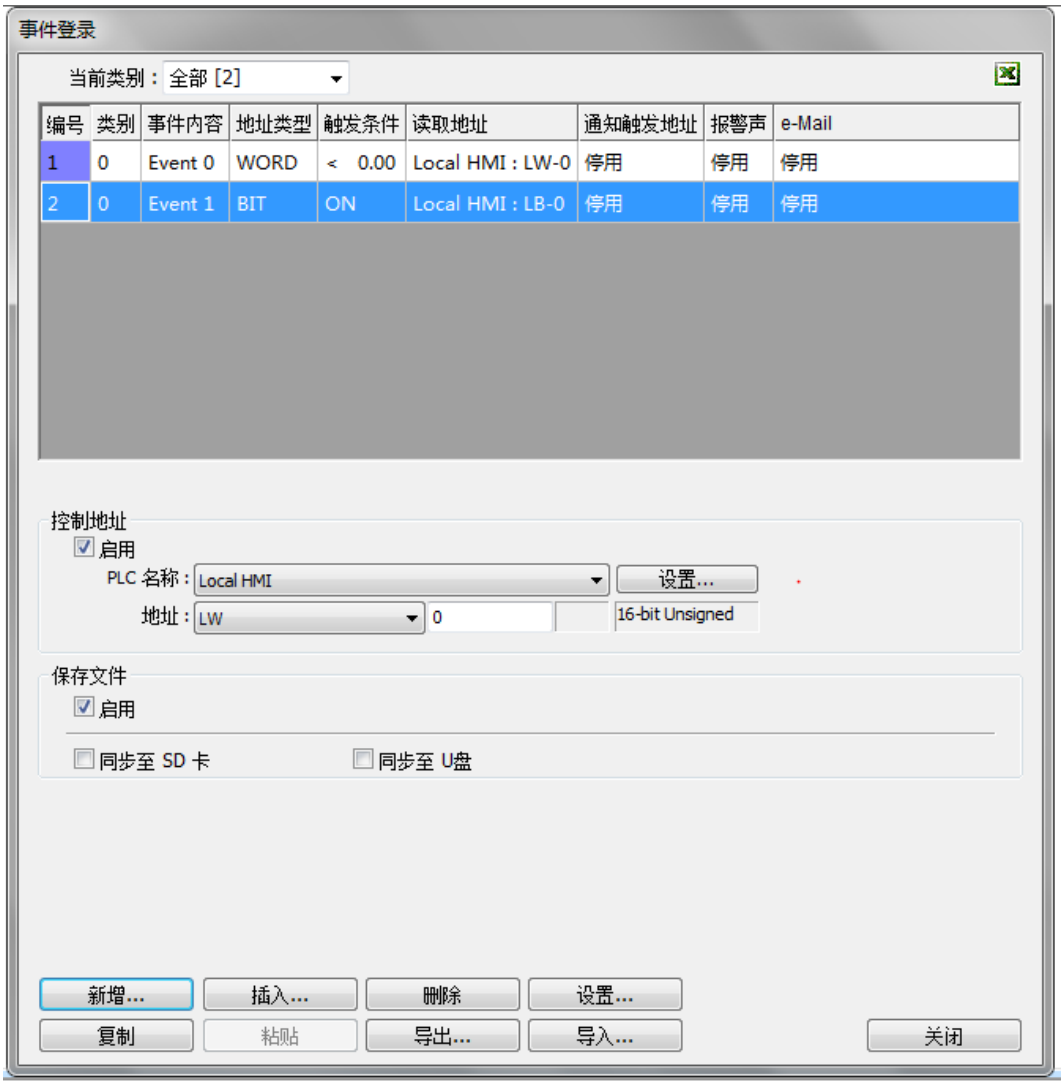
透过报警条  、报警显示  、事件显示  等元件，可以了解事件从发生 → 等待处理 → 警报解除的时间。首先需定义事件的内容。

7.2.1 eMT、iE、cMT-HD 系列



设置	描述
当前类别	提供事件分类功能，将事件分成 0 ~ 255 个类别，可选择一个类别来登陆或显示事件数据。[ ] 中显示目前的事件类别。
保存文件	指定事件记录文件的储存位置，但若使用在线或离线模拟功能时，文件一律存放在安装目录下的 HMI_memory / SD_card / U 盘文件夹内。 **文件保留时间限制** 此项设置值用来设置事件记录文件被保留的天数。若 [保留时间] 设置为两天，也就是系统将只保留昨天与前天的事件记录，这个时间范围之前的文件将自动被删除，用来避免储存空间被耗尽。
打印格式	需先在 [系统参数设置] » [HMI 属性] 选择打印机型号，才允许设置当事件触发时，欲打印的格式。

7.2.2 cMT-SVR 系列



设置	描述
目前类别	提供事件分类功能，将事件分成 0 ~ 255 个类别，可选择一个类别来登陆或显示事件数据。[ ] 中显示目前的事件类别。
控制地址	主要控制事件记录文件存放方式，分为系统自动存放和用户手动控制。 **系统自动存放** 若没有勾选 [控制地址]，系统会自动将事件记录文件储存到 HMI_memory，可存放 10000 笔数据，超过 10000 笔时，会将前面的 1000 笔数据删除。 若勾选 [控制地址]，会依照 [保存文件] 是否有启用，来决定存放到 HMI_memory、SD 卡或是 U 盘。 **用户手动控制** 若勾选 [控制地址] 并启用 [保存文件]，用户可以在控制地址输入数值，来同步或清除装置中的事件记录文件。

	当写入特定数值于控制地址时，可触发相关命令。 当数值设为 1 时，清除 cMT-SVR 上的资料。 当数值设为 2 时，将数据同步到外接装置。 当数值设为 3 时，先将数据同步到外接装置后，再清除 cMT-SVR 上的数据。 若用户没有手动在控制地址上输入以上数值，系统仍会自动将事件记录文件放在 HMI_memory 里。
保存文件	事件记录文件的储存位置。可选择同步至 SD 卡或 U 盘。

Note

- 若用户需将 SD 卡或 U 盘拔除时，可以先使用控制地址将事件记录同步。

7.2.3 Excel 编辑

点击事件登录窗口右上角的 Excel 小图示，可直接开启 Excel 表格做为编辑时的范例参考。
此范例为安装目录下的 EventLogExample.xls Excel 文件，范例文件中有设计好的验证机制和下拉式选单。

	A	B	C	D	E	F	G	H	I	J	K
1	Category	Priority level	Address type	PLC name	Device type	System tag	User-defined tag	Address	Index	Data Format	Enable
2	0	Middle	Word	Local HMI	LW	False	False	100	null	32-bit Signed	True
3	1	Low	Bit	Local HMI	LB-9009	True	False	9009	IDX 5	16-bit BCD	lse
4										16-bit BCD	
5										32-bit BCD	
6										16-bit Unsigned	
7										16-bit Singed	
										32-bit Unsigned	
										32-bit Signed	
										32-bit Float	

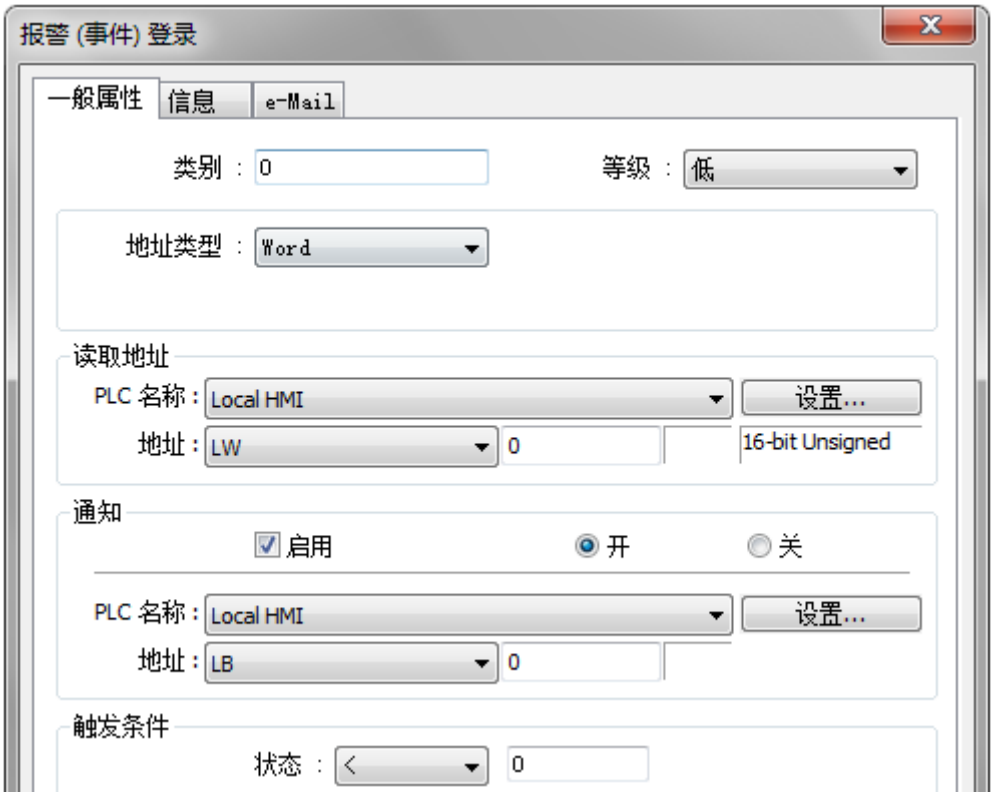
Note

- [System tag] 与 [User-defined tag] 请勿同时设为 true，否则系统会将其视为 [System tag]，并将 [User-defined tag] 视为 false。若在 [Device type] 所输入的数据为 [User-defined tag]，请将 [System tag] 设为 false。
- 若在 Excel 表中将 [User-defined tag] 设为 true，但若系统将 [Device type] 与系统中用户自订的 tag 做比对，却找不到合适的 tag，则系统会自动将事件登录中的 [User-defined tag] 设为 false。
- [Color] 格式为 R:G:B，各介于 0 ~ 255 之间的整数。
- 在导入文字卷标库或声音库前，请确认系统中已存在对应名称。

7.3 建立一个新的事件记录

7.3.1 一般属性设置

点击 [事件登录] 窗口的 [新增]，将出现 [一般属性] 对话框。



设置	描述
类别	选择事件类别，从 0 ~ 255。
等级	当已发生事件的数目等于系统允许的最大值数目 (默认值为 1000) 时，重要程度较低的事件将从事件记录中被删除，并加入新发生的事件。
读取地址	系统利用读取此地址所获得的数据，来检查事件是否满足触发条件。
通知	若勾选，系统会在事件发生时，将指定寄存器状态设为 [开] 或 [关]。
触发条件	当选择 Bit 时，事件登录将侦测一个位地址的状态。 当选择 Word 时，事件登录将侦测一个字符地址的值是否等于、大于或小于一个特定数值。请见以下范例 1、范例 2。

范例 1

上面的设置内容表示

当 [读取地址] 中的数据大于等于 29 (= 30 – 1)

或小于等于 31 (= 30 + 1) 时，事件将被触发。也就是事件被触发的条件为：

29 ≤ [读取地址] 中的数据 ≤ 31

事件被触发后，当 [读取地址] 中的数据大于 32 (= 30 + 2) 或小于 28 (= 30 – 2) 时，系统将恢复为正常状态。也就是系统恢复为正常状态的条件为：

[读取地址] 中的数据 < 28 或 [读取地址] 中的数据 > 32

范例 2

上面的设置内容表示

当 [读取地址] 中的数据小于 29 (= 30 – 1)

或大于 31 (= 30 + 1) 时，事件将被触发。也就是事件被触发的条件为：

[读取地址] 中的数据 < 29 或 [读取地址] 中的数据 > 31

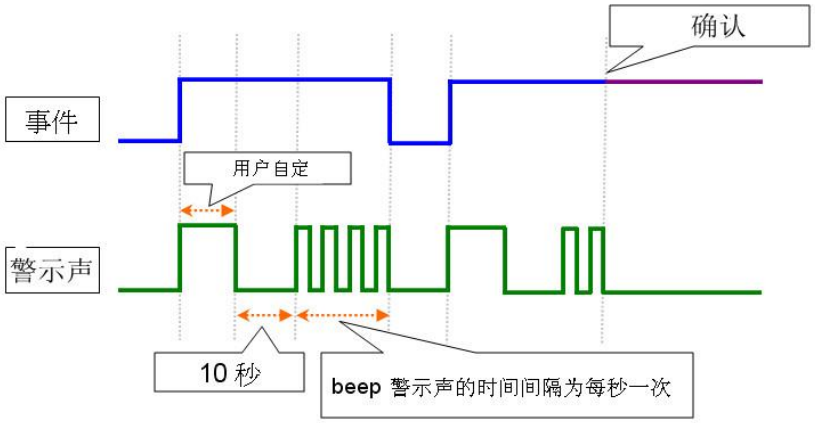
事件被触发后，当 [读取地址] 中的数据大于等于 28 (= 30 – 2) 或小于等于 32 (= 30 + 2) 时，系统将恢复为正常状态。也就是系统恢复为正常状态的条件为：

28 ≤ [读取地址] 中的数据 ≤ 32

7.3.2 信息设置

设置	描述
内容	事件记录在 [报警条]、[报警显示] 与 [事件显示] 元件中显示的信息内容。并可在内容中使用 [监看地址] WATCH1 至 WATCH8 来引用数据或见以下范例 3、范例 4。
字体 / 颜色	每一个事件可以单独设置字型与颜色，用于 [报警条]、[报警显示] 与 [事件显示] 中的字型颜色。
事件确认时写入	当 [事件显示] 与 [报警显示] 元件中的事件项目被确认时，会将此数值写入该元件指定的 [写入地址]。

报警声	若勾选 [启动报警声]，当事件发生时会播放指定的声音，并且可以选择持续发出警示声响，直到该事件被确认或恢复正常时，才会停止发出声音。 若勾选使用持续警示声响时，可以选择在警报被触发后，延迟所指定的时间后才开始连续发出声音。



监看地址	用户可以点选 [语法] 来编辑并显示当事件触发时，监看地址所设置寄存器内的数值。最多可同时监看八个地址。

范例 3

可以在显示内容中包含事件被触发当时 LW 地址中的数据。
使用格式为：**%#d** (% -> 起始, # -> 地址, d -> 结束)
假设触发时 LW-20 中的数值为 13：
设置为 “High Temperature = %20d”，则会显示为 “High Temperature = 13”。

范例 4

可以在显示内容中包含事件被触发当时，特定 PLC 地址类型中的数据，此特定地址类型与事件登录的 [读取地址] 需为相同地址类型，假设选择 MODBUS RTU 4x 地址类型。
使用格式为：$#d ($ -> 起始, # -> 地址, d -> 结束)
假设触发时 MODBUS RTU 4x-15 中的数值为 42：
设置为 “High Temperature = $15d”，则显示为 “High Temperature = 42”。

7.3.3 e-Mail 设置

[事件登录] 另一页为 [e-Mail] 设置页。请先在 [系统参数设置] » [e-Mail] 启用此功能。

设置	描述
收件者	可分别在收件者、副本、密件副本字段，选择收件者。
主旨	可输入信件显示的主题内容。
讯息	可输入信件显示的开头与结尾的内容。
附件	若勾选，即可选择屏幕截图当作附件。

第八章 资料取样

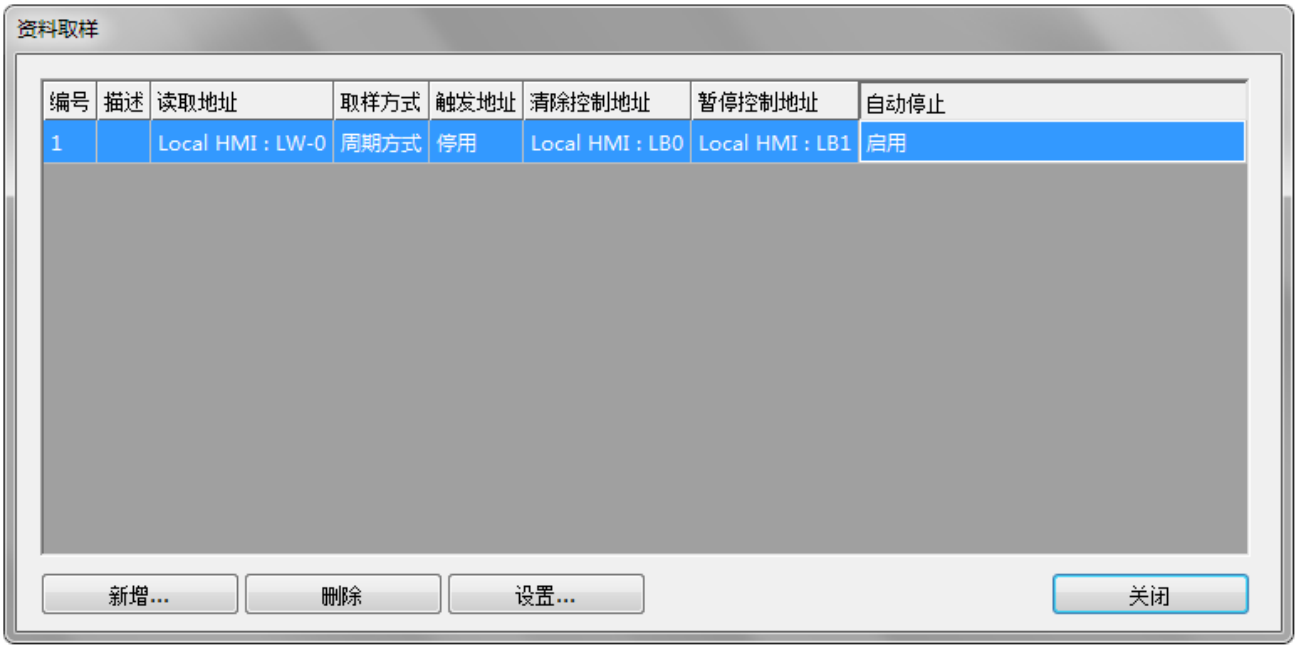
8.1 概要

定义「资料取样」的取样方式，例如：取样时间，取样地址，及字符长度后，可将已获得的取样资料储存到指定的位置，如 HMI 内存、SD 卡或 U 盘。资料取样可搭配使用趋势图或历史数据显示元件检视资料取样记录的内容。

8.2 资料取样记录管理

新增一个资料取样，请依照下列步骤：

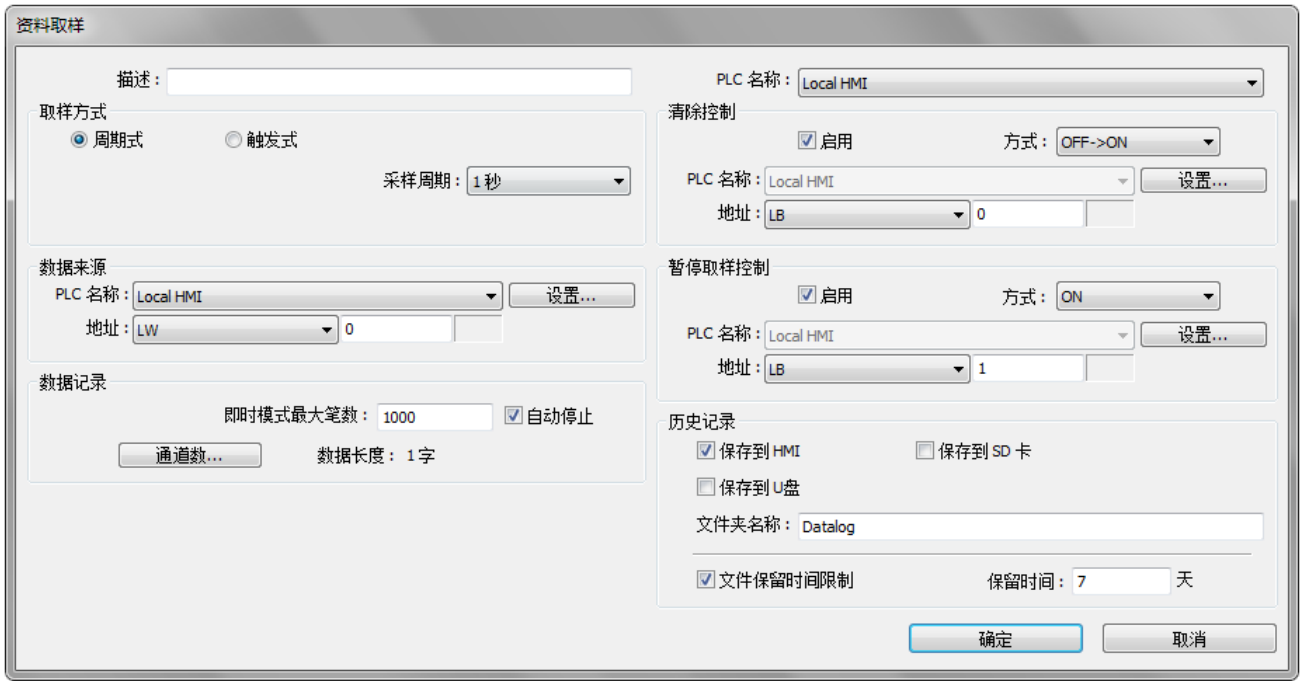
1. 点选菜单 [元件]，再点击 [资料取样]。
2. 点击 [新增] 开始相关设置，如下图所示：



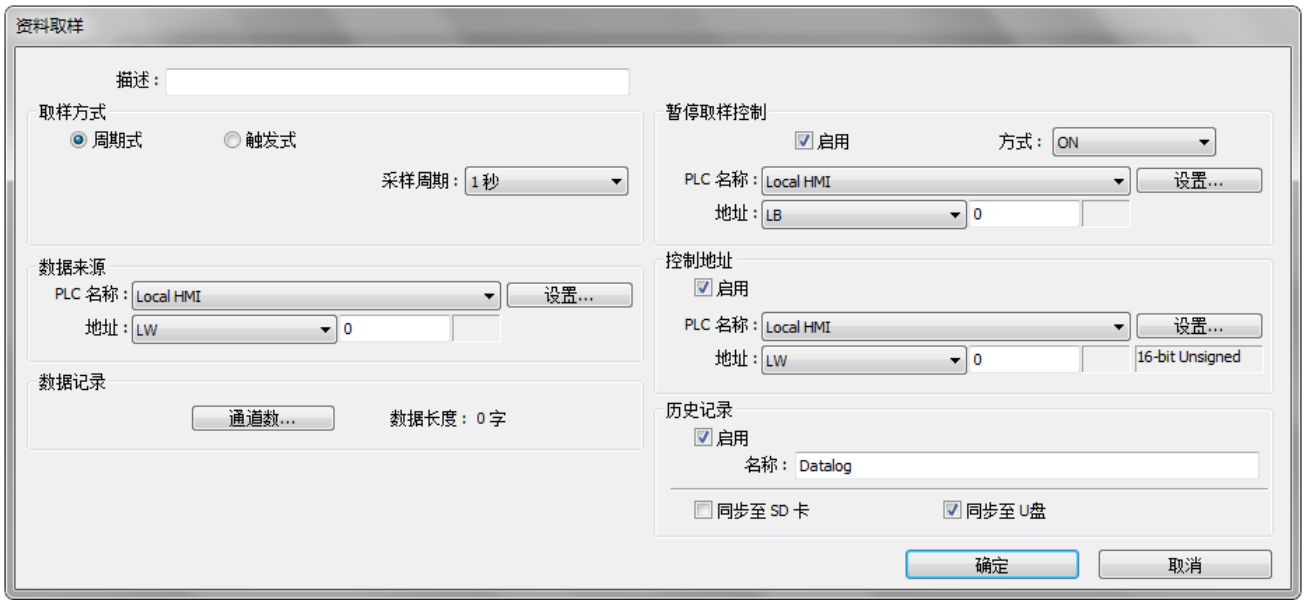
8.3 新增一个资料取样

以下介绍如何设置一笔资料取样元件：

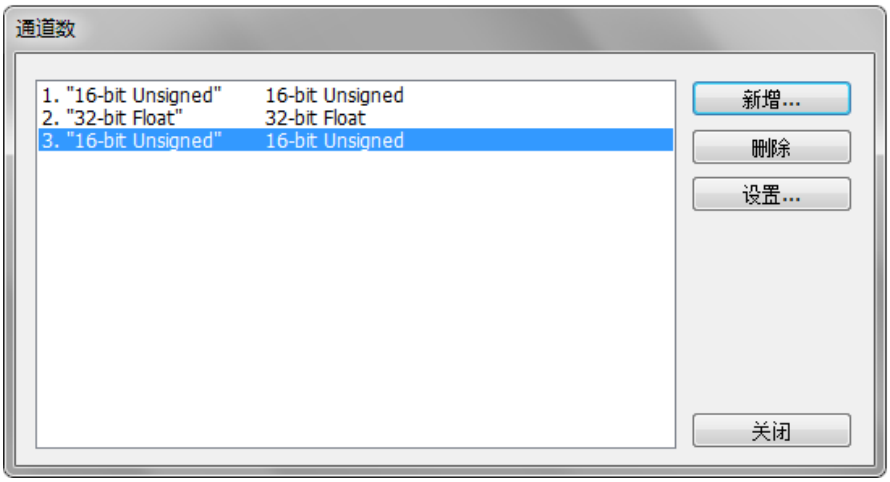
eMT、iE、cMT-HD 系列



cMT-SVR 系列



设置	描述
取样模式	**周期式** 用固定的时间频率进行资料取样，[采样周期] 可设置范围从 0.1 秒至 120 分钟。 **触发式** 利用一个特定位地址的状态，来触发取样动作。 **[模式]** 可为：

	[OFF -> ON] 当指定地址的状态从 OFF 变为 ON，会触发资料取样。 [ON -> OFF] 当指定地址的状态从 ON 变为 OFF，会触发资料取样。 [OFF <-> ON] 只要指定地址的状态改变，就会触发资料取样。
数据来源	选择一个设备地址作为资料取样的来源。
数据记录	一个资料取样项目一天最多可以记录的资料取样笔数为 86400，即一天 24 小时，每秒取样一次。若 [采样周期] 设为 “0.1 秒”，一天最多仍为 86400 笔。 **数据格式** 可设置读取多个不同格式的连续地址的数据。假设有三笔数据，地址及格式分别为 LW-0 (16-bit Unsigned)、LW-1 (32-bit Float)、LW-3 (16-bit Unsigned)，则需建立数据格式如下图所示：  **自动停止** 搭配不同元件，[自动停止] 产生的效用就不同。 请见《8.3.1 自动停止选项范例》。
清除控制	当指定位地址的状态由 [OFF -> ON] 或 [ON -> OFF] 时，将清除在趋势图 [即时模式] 下已取样的资料，取样资料的数目也会被归零，但不影响已存成文件中的历史取样资料。
暂停控制	当指定位地址的状态被触发时，将暂停取样动作，直到指定地址的状态恢复。
控制地址 (cMT-SVR 系列)	对控制地址输入指令数值时，即对历史数据发送特定的命令。 命令 1：清除 HMI 里的资料。 命令 2：将数据同步到外接储存装置。 命令 3：先将数据同步到外接储存装置，接着清除 HMI 里的数据。
历史记录	● **eMT、iE、cMT-HD 系列** **保存到 HMI** 将取样资料储存在 HMI 里。资料必需要到达 4kb，才会被储存；若少于 4kb，可以使用系统寄存器 [LB-9034] 来强迫储存。

保存到 SD 卡 / U 盘

将资料取样储存到指定的外部装置中。

文件夹名称

设置取样文件夹的名称，必须全部由 ASCII 字符所组成。

储存的方式为：[保存位置] \ [取样文件夹名称] \ yyyymmdd.dtl

文件储存是按日期记录至指定文件名的数据夹内。

文件保存时间限制

此项设置值用来决定资料取样记录文件被保留的时间。

- **cMT-SVR 系列**

历史数据储存的路径，可选择 U 盘或 SD 卡。当 cMT-SVR 的资料取样至 10000 笔时，会自动将资料取样储存至指定的外接储存装置，并删除最早的 1000 笔数据。

详细规则请见《8.4 外接储存装置同步 CloudHMI 数据》。

Note

- 一笔资料取样可能包含超过一项以上的数据，资料取样动作可以同时撷取不同类型的数据。您可以自行定义一笔取样资料的内容。例如：您总共定义了三笔数据，总长度为 4 words，即每次的取样动作，系统会从指定的数据来源地址撷取长度为 4 words 的数据，作为一笔取样资料的内容。
- 假设 [文件保存时间限制] 设置保留时间为两天，也就是说系统将仅保留昨天与前天的资料取样记录文件，超过这个时间范围的文件将自动被删除，以避免储存空间被耗尽。
- 在 PC 使用仿真功能时，资料取样文件一律储存在安装目录下的「保存位置」下的 datalog 子目录。路径为：C:\EBpro\ [保存位置] \datalog。此时若要改变资料取样的格式，须先删除安装目录下的旧资料取样记录，以避免系统误读旧记录。

8.3.1 自动停止选项范例

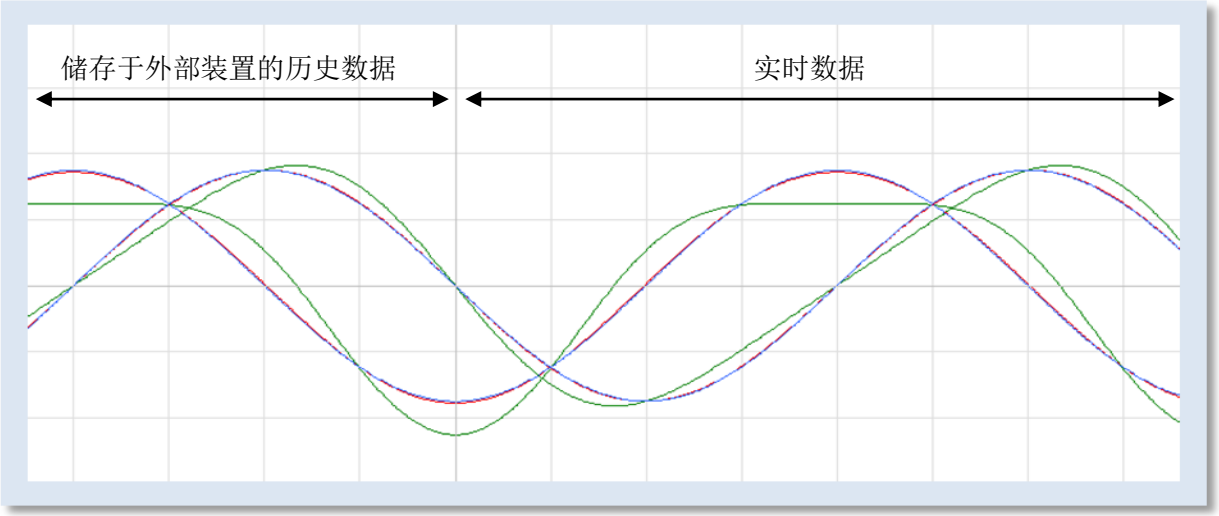
搭配不同元件，[自动停止] 产生的效用就不同，如下表：(假设最大资料设为 n)

搭配元件	未勾选 [自动停址]	勾选 [自动停止]
趋势图-实时模式	将删除较旧的取样数据，并显示刚获得最新的 n 笔资料在趋势图上。请参考下列图解。	至第 n 笔数据后停止动作。
趋势图-历史模式	数据持续被取样，并显示所有历史数据在趋势图上。	至第 n 笔数据后停止动作。
历史数据显示	数据持续被取样，并显示所有历史数据在历史数据显示上。	至第 n 笔数据后停止动作。
资料取样	持续取样新记录。	至第 n 笔资料后停止取样。

范例：设置数据长度数为 10 个，当第 11 个数据产生时，在未勾选 [自动停止] 的状况下，最旧的数据记录将会被删除，并增加最新的记录，如下图所示。

Record Number	Data	Not selecting [Auto. stop]
1	101	102
2	102	103
3	103	104
4	104	105
5	105	106
6	106	107
7	107	108
8	108	109
9	109	110
10	110	111
11	111	

8.4 外接储存装置同步 CloudHMI 数据



以往资料取样的数据在 [趋势图] 元件上显示时，必须设置是实时模式或是历史模式，两者模式不可共存于同一个趋势图元件上。cMT-SVR 系列将历史模式与实时模式结合，达到无缝连接，让所有资料取样可在同一个 [趋势图] 元件或 [历史数据显示] 元件上显示，画面也会自动的更新。另外可透过外接储存装置将数据与之同步。

同步规则：

1. 每当取样的资料达至 10000 笔时，人机便会自动将数据写入至外接储存装置，接着删除 HMI 里最早的 1000 笔数据。
2. 若将外接储存装置从 HMI 上移除，之后在新产生的取样的资料不超过 9000 笔时就插回 HMI，那在外接储存装置移除的期间，取样资料仍然会保存在 HMI 里不会被清除。若在外部装置移除的

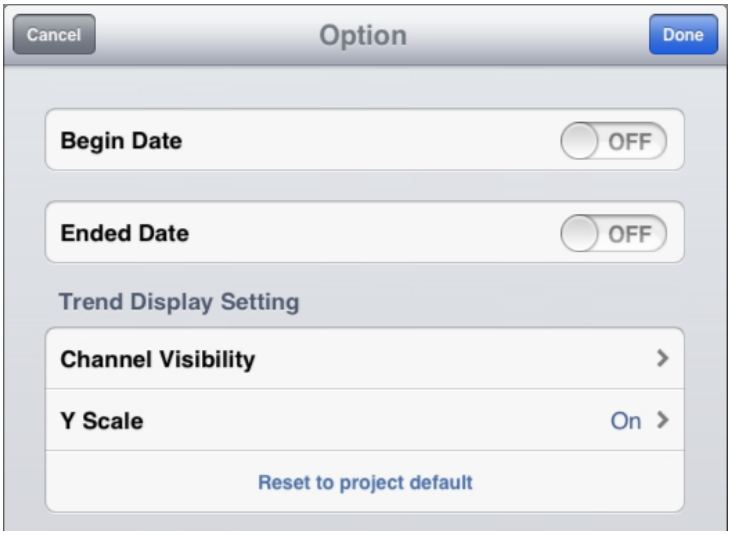
期间，取样的数据已经超过 9000 笔，那较旧的数据就会被删除，即使之后将外接储存装置插回也无法同步到被删除的数据。

3. 若外接储存装置内已经有取样数据，则每次同步时，新的取样数据将会附加在原有的数据。

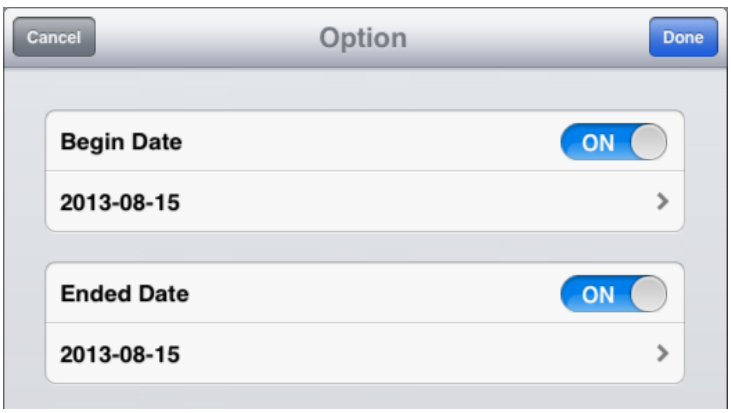
8.5 查看 CloudHMI 特定日期的历史资料

若想查看历史的资料，请参照以下步骤 （以趋势图为例）：

1. 点击趋势图右上方的  按钮。
2. 跳出趋势图设置对话窗。



3. 指定欲查询的 Begin Date (开始日期) 及 Ended Date (结束日期)。



4. 按下 [Done] 完成设置。

第九章 元件一般属性

9.1 概要

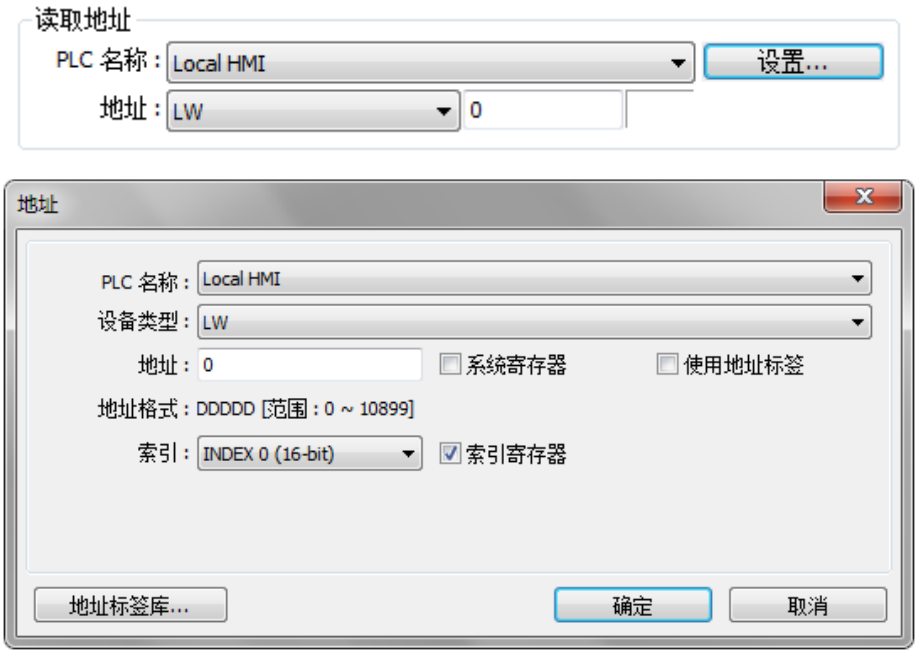
一般建立一个元件的步骤如下：

1. 选择 PLC 设备并设置读写地址。
2. 使用向量图库或图片库。
3. 设置标签内容。
4. 调整轮廓。

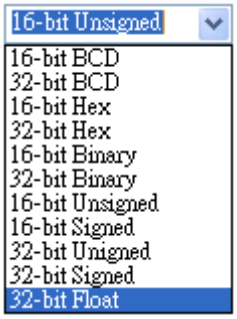
本章将说明元件一般属性的内容。

9.2 选择 PLC 并设置读写地址

某些元件的使用需选择要操作的 PLC 元件。[PLC 名称] 用来表示要控制的 PLC，这些 PLC 名称来自 [系统参数] 中 [设备列表] 的内容。



设置	描述
PLC 名称	选择 PLC 的型号。
设备类型	选择地址类型，当 PLC 型号不同时，将出现不同的地址类型。
地址	设置读写的地址。
系统寄存器	地址标签包含 [系统寄存器] 与 [用户自定义地址标签]。此项目用来选

	择是否使用 [地址标签库]。系统寄存器为系统保留作为特殊用途的地址，分为 bit 地址系统寄存器与 word 地址系统寄存器 (LB 或 LW)。在选择使用 [系统寄存器] 后，除了 [设备类型] 将显示系统寄存器的内容之外， [地址] 将显示目前所选用的系统寄存器。
索引寄存器	选择是否使用 [索引寄存器]，有些元件需正确选择数据类型，尤其是在使用地址标签库时。EasyBuilder Pro 支持下列的数据型态： 

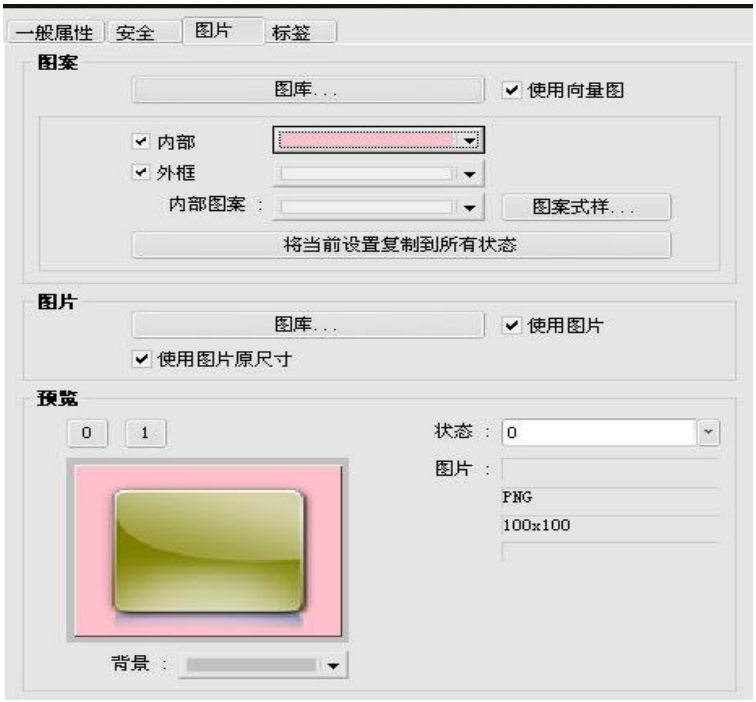
关于系统寄存器的详细信息请参考《第二十二章 地址寄存器》。

关于索引寄存器的详细信息请参考《第十一章 索引寄存器》。

关于地址标签库的详细信息请参考《第十六章 地址标签库》。

9.3 使用向量图库与图片库

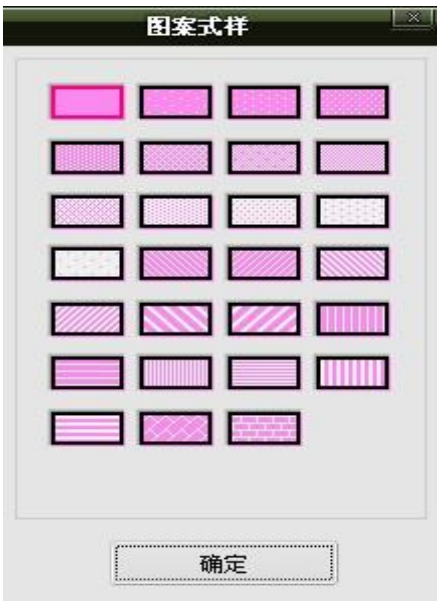
某些元件可以使用向量图库与图片库的图形，增加元件的视觉效果。向量图库与图片库的使用在元件属性页中的 [图片] 页中设置，如下图所示。



设置	描述
图库	勾选 [使用向量图] 并从 [图库] 中选择图形样式，此项目请参考后面的说明。
内部	选择是否使用图案的内底，按下颜色设置钮后所出现的 [颜色] 对话窗，可来设置内底的颜色。用户也可以设置 [自定义颜色]，按下 [新增自定颜色] 后 EasyBuilder Pro 会记住用户设置的 [自定义颜色]。 
外框	选择是否使用图案的外框，按下颜色设置钮后所出现的 [颜色] 对话窗，可用来设置外框的颜色。
内部图案	用来设置内底填充的图案式样与颜色。

图案式样

按下设置钮后，可用来选择填充式样。



把设置复制到所有状态	将目前状态各项属性设置到其它状态。

9.3.1 向量图管理

按下 [图库] 按钮后可以得到下面的 [向量图库] 对话窗，由对话窗中可看出目前选择的样式会使用黄色背景来加以标示。

上图显示向量图库中某一样式的信息，这些信息的意义如下：

Shape1　　表示此向量图的名称

状态数 ：2　　此向量图的状态个数

元件：4　　表示此向量图被 4 个元件所使用

此向量图的状态 0 与状态 1 皆只具备内部，不具备外框。在完成各项设置并按下确定键后，元件将使用目前所选择的样式。

9.3.2 图片管理

勾选 [使用图片]，在按下 [图库] 按钮后可以得到下面的 [图片管理] 对话窗并从中选择图片。

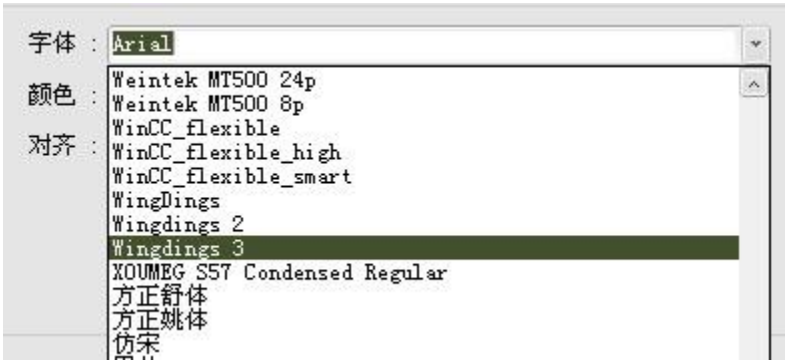
关于向量图库与图片库详细信息请参考《第十四章 向量图库与图形库的建立》。

9.4 设置标签内容

设置	描述
使用文字标签	勾选此选项元件才允许使用文字标签。
使用文字标签库	勾选此选项表示文字内容将来自文字标签库。
使用 bitmap 字体	勾选此选项，文字将使用 bitmap 图形显示。
文字标签库	检视文字库的内容 详细信息请参考《第十五章 文字标签库与多国语言使用》。
字体	选择文字所使用的字型。EasyBuilder Pro 支持 WINDOWS 的

true-font 字型。

cMT-SVR 机型选择字型后若有括号，内容代表此项目下载到 iPad 上实际所显示的字型。



颜色	选择文字的颜色。
尺寸	选择文字的大小。
对齐	**左对齐** **置中对齐** **右对齐** 111 111 111 222222 222222 222222 333333333 333333333 333333333
闪动	选择文字闪烁方式，可选择不闪烁 [无]，或闪烁时间间隔为 [1 秒] 或 [0.5 秒] 的闪烁方式。
斜体	使用斜体字体。 *Italic Label*
底线	文字加上底线。 Underline Label
走马灯	**移动方向** 设置走马灯的效果并选择文字的移动方向，可选择 [不移动]、[向左]、[向右]、[向上]、[向下]。 **持续移动** 未勾选此选项，则文字需在全部消失后才出现后续的文字，如下图。 Alarm rm 有勾选此选项，则文字会连续出现，如下图。 rm Alarm A **速度** 选择文字的移动速度。
内容	设置文字内容。如使用 **[文字标签库]**，此项内容将来自文字标签库。

编辑时文字位置连动	勾选此选项时，移动某个状态的文字将连带移动其它状态的文字。
复制文字到所有的状态	将目前的文字内容复制到其它所有的状态。

9.5 轮廓调整

将元件放置于编辑窗口后，再点击元件可利用**[轮廓]** 设置页来调整元件的外型大小。

设置	描述
位置	**图钉** 锁定设置，勾选此选项后将无法改变元件的位置与大小。 **[X]、[Y]** 此坐标为元件放置于编辑窗口中的位置。
尺寸	设置元件的 [宽度] 和 [高度]。

第十章 用户密码与元件安全防护

10.1 概要

在 EasyBuilder Pro 设置用户的安全等级与密码，共有两种模式：

- 一般模式
- 进阶安全模式

后面的章节将详细介绍两种模式的设置方式。

元件安全防护须完成以下两项设置：

1. 用户密码与可操作元件类别设置
2. 元件操作安全防护

一个元件只能属于一个安全等级，或将安全等级设置为“无”，使得任何人皆可操作该元件。

10.2 用户密码与可操作元件类别设置

在 [编辑] » [系统参数设置] » [用户密码] 设置页可设置相关参数，分为两种模式：一般模式和进阶安全模式。

10.2.1 一般模式

一般模式可设置最多 12 个用户，各设置不同的用户密码，密码需为非负整数，并规划每个用户可操作的元件类别分为“A ~ F”等共 6 个类别。

HMI 运作时，用户在成功输入密码后，系统会依照设置内容决定用户可以操作的元件类别。如下图，“用户 1”只被允许操作元件类别为“A、C”。

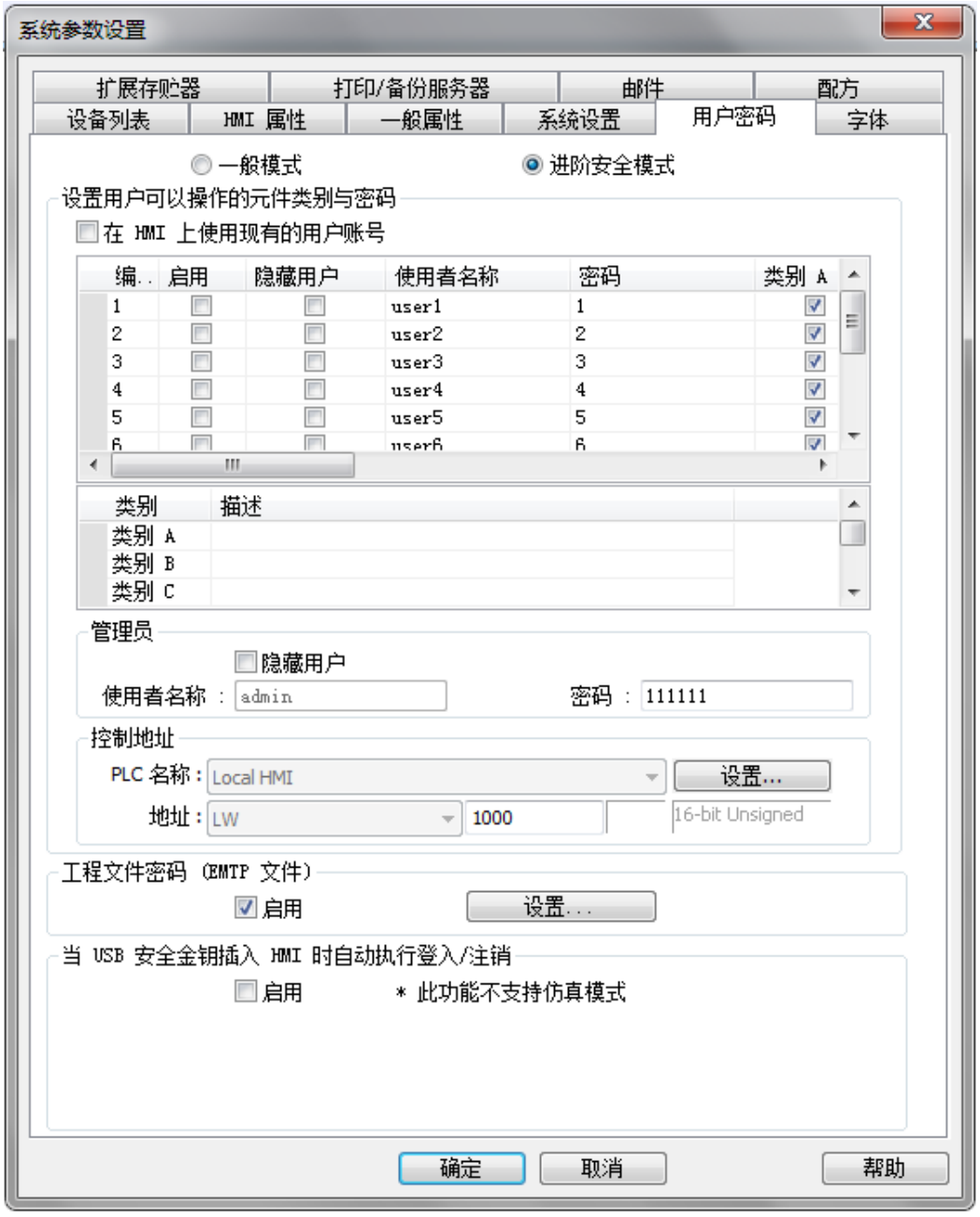
Note

- cMT-SVR 机型不支持用户密码的一般模式。

10.2.2 进阶安全模式

进阶安全模式可设置的用户数为 11 组，另外提供 [管理员] 使用模式，此管理员有最大使用权限，任何元件的安全等级皆可操作。不同的用户密码可由英文或数字所组成，并可设置每个用户可操作的元件类别分为 “A ~ L” 等共 12 个类别。(使用管理员工具，可设置最多 127 组用户。此功能介绍于《10.4 进阶安全模式之应用》)

进阶安全模式提供一组 [控制地址] 机制，供用户登入和管理账号，请参考《10.3 进阶安全模式之控制地址》的说明。或者可使用 USB 安全金钥自动登入，当插入的设备含有 USB 安全金钥时，将自动登入指定的账号，请参考《10.4.3 使用 USB 安全金钥登入 / 注销 》的说明。



10.3 进阶安全模式之控制地址

控制地址可用于登陆和管理账号，此控制地址的字符地址来源只能为 Local HMI 的 LW 地址，并使用连续 20 个地址作为相关管理功能。使用 [控制地址] 执行登陆时，需选择 [用户名称] 或 [用户索引] 其中一种方式登陆。

[用户名称] 及 [密码] 需先至 [系统参数设置] » [用户密码] » [进阶安全模式] 设置。

10.3.1 控制地址使用说明

当控制地址设为 LW-n 时，n 为任一数字，则将使用以下地址：

控制地址	标签名称	描述
LW-n (占 1 个字符)	命令	控制各项操作命令 (例如: 登陆，注销，新增/修改/删除账号...等等)。
LW-n + 1 (占 1 个字符)	命令执行结果	显示执行命令的结果。
LW-n + 2 (占 1 个字符)	用户索引	账号索引 (配合项目选单元件使用)。
LW-n + 3 (占 1 个字符)	用户权限	权限值 (Level A = bit0, Level B = bit1...等等)。
LW-n + 4 (占 8 个字符)	用户名称	账号名称 (可为英文或数字 + '-' + '_', 大小写视为不同)。
LW-n + 12 (占 8 个字符)	密码	账号密码 (可为英文或数字 + '-' + '_', 大小写视为不同)。

当控制地址设置后，可至 [地址卷标库] » [用户定义] 查询其它相关功能的地址。

例如，当 [控制地址] 为 LW-0 时，则

LW-0 → [命令]

LW-1 → [命令执行结果]

LW-2 → [用户索引]

LW-3 → [用户权限]

LW-4 ~ LW-11 → [用户名称]

LW-12 ~ LW-20 → [密码]

Note

- 当使用 cMT-SVR 机型搭配用户密码之进阶安全模式时，控制地址的字符地址来源只能为 Local HMI 的 PLW 地址。

10.3.2 命令功能说明

当在 [命令] LW-n 输入特定 [数值] 时,-> 可操作的功能如下：

设置数值	命令	搭配地址
1	使用名称登入账号	需先定义 [用户名称] 和 [密码]，输入名称及密码后，系统会去比对是否和 [系统参数设置] » [用户密码] » [进阶安全模式] 设置相符。
2	使用索引登入账号	需先定义 [用户索引] 和 [密码]，请参考

		《10.4.4 进阶安全模式搭配项目选单元件》说明。
3	注销账号	
4	更改目前已登入账号的密码	需先定义 [用户名称] 和 [密码]，其中 [用户名称] 需填入原密码，[密码] 填入新密码。
5	新增账号	需先定义 [用户名称]，[密码] 和 [用户权限]。
6	新增临时账号	需先定义 [用户名称]，[密码]，[用户权限] 和 [用户索引]，其中 [用户索引] 用来指定有效时间 (单位:分钟)，0 表示永久有效。
7	使用名称删除现有账号	需先定义 [用户名称]。
8	使用索引删除现有账号	需先定义 [用户索引]。
9	使用名称设置现有账号的权限	需先定义 [用户名称] 和 [用户权限]。
10	使用索引设置现有账号的权限	需先定义 [用户索引] 和 [用户权限]。
11	使用名称设置现有账号的密码	需先定义 [用户名称] 和 [密码]。
12	使用索引设置现有账号的密码	需先定义 [用户索引] 和 [密码]。
13	使用名称读取现有账号的权限	需先定义 [用户名称]，若成功执行则将账号权限显示于 [用户权限]。
14	使用索引读取现有账号的权限	需先定义 [用户索引]，若成功执行则将账号权限显示于 [用户权限]。

Note

- 新增临时账号: 临时账号与一般账号的差异在于临时账号不会被保存于 Flash 里，故断电后即失效，且临时账号将在超出有效时限后自动被删除。
- 删除现有账号: 不可删除目前登入账号。
- 离线模拟/在线模拟: 皆使用程序内的账号设置，进行模拟时，用户对账号内容的修改不会保留到下一次模拟。
- admin: 被内定为管理员账号，不可被删除，且权限全开不得修改权限。
- 系统寄存器 LW-10754: 显示目前登入的用户名称。
- 系统寄存器 PLW-10754: 显示目前登入的用户名称 (只支持于 cMT-SVR 机型)。

10.3.3 结果输出说明

每当执行命令后，系统自动将执行结果输出值传送到控制地址的 LW-n + 1 地址。下列结果输出值为 16 进制数值：

结果输出值	信息
(0x001)	命令执行成功
(0x002)	错误命令
(0x004)	账户已存在 (发生于新增账号时)
(0x008)	账户不存在
(0x010)	密码错误
(0x020)	目前命令无法被执行
(0x040)	不合法的账户名称
(0x080)	密码含有无效字符
(0x100)	导入数据无效
(0x200)	超出有效期限 (当使用 USB 安全金钥登入时，有效期限可设置于管理员工具)

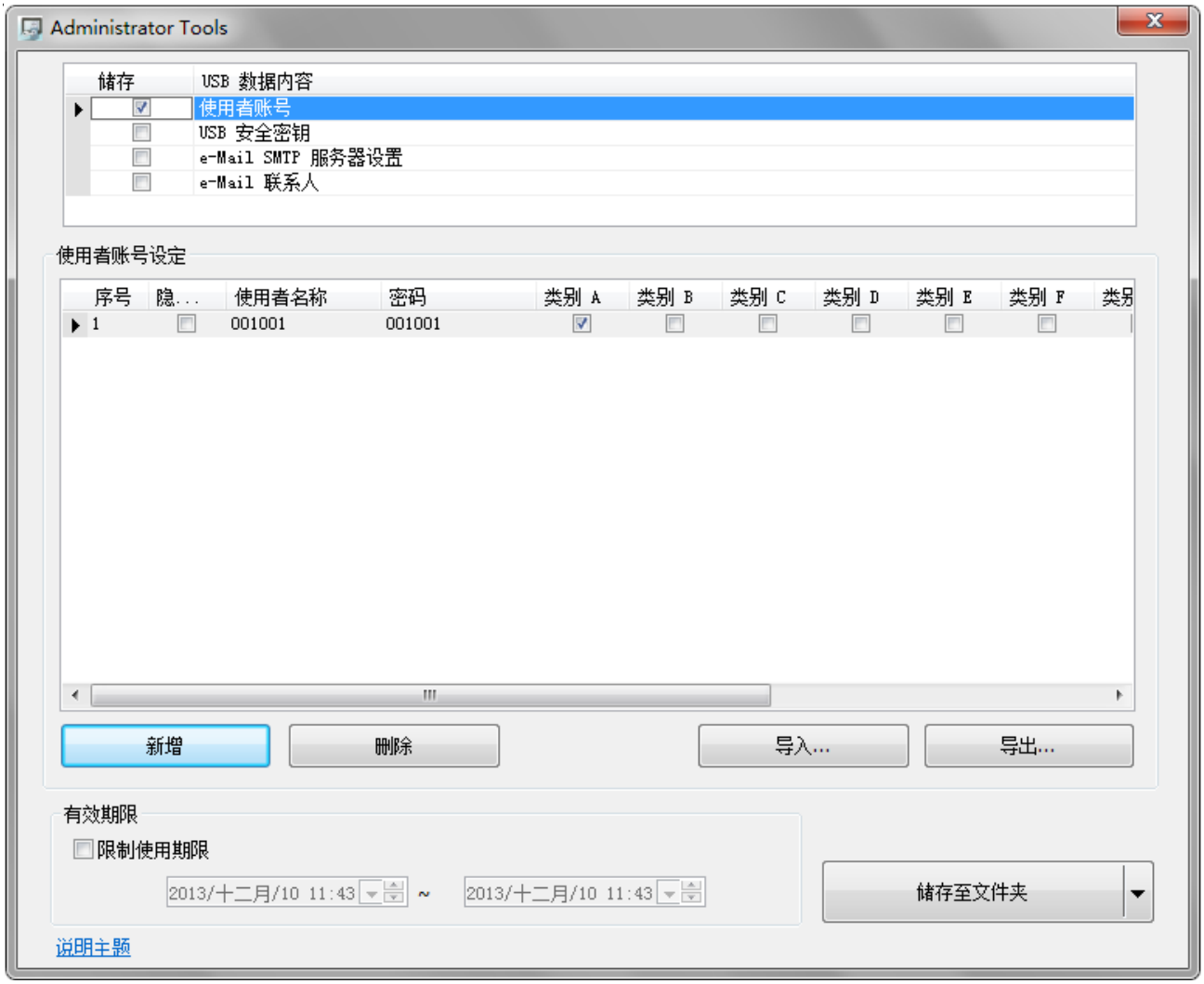
Note

- 用户可将 [命令执行结果] LW-n + 1 与输出值的对应信息预先建立于事件登录元件里，并藉由事件显示元件直接显示命令的执行结果。

10.4 进阶安全模式之应用

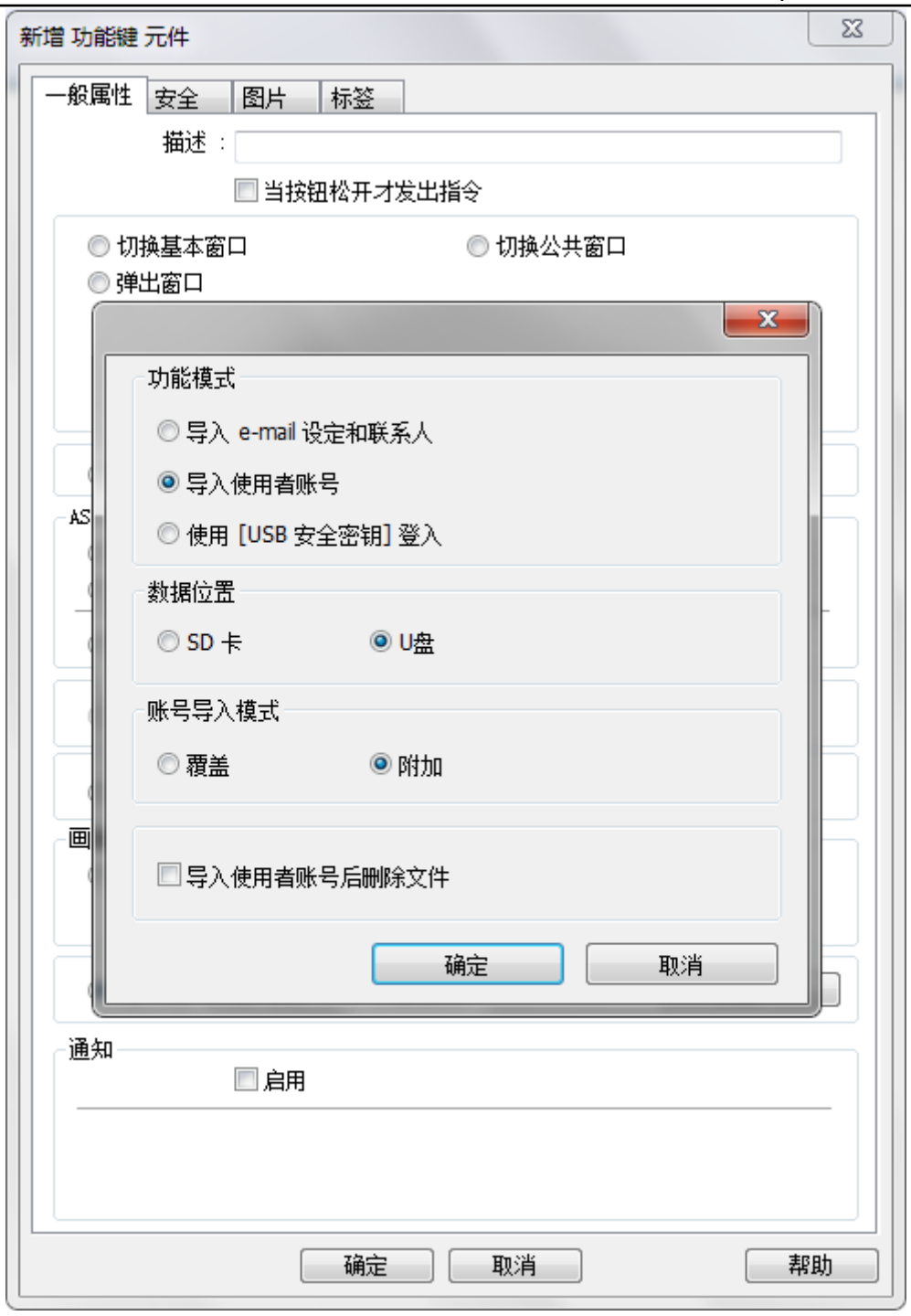
10.4.1 导入用户账号

除了 [系统参数设置] » [用户密码] 设置页可以设置用户账号之外，从 EasyBuilder Pro 安装目录下开启管理员工具后，勾选 [用户账号]，亦可进行用户账户数据的设置。使用管理员工具最多可新增 127 笔账户资料。使用画面如下所示：



关于管理员工具的详细信息请参考《第三十六章 管理员工具》。

新增账号完毕后，可储存至 U 盘或 SD 卡，并预先在程序内建立 [功能键] » [导入用户账号]，设置如下：

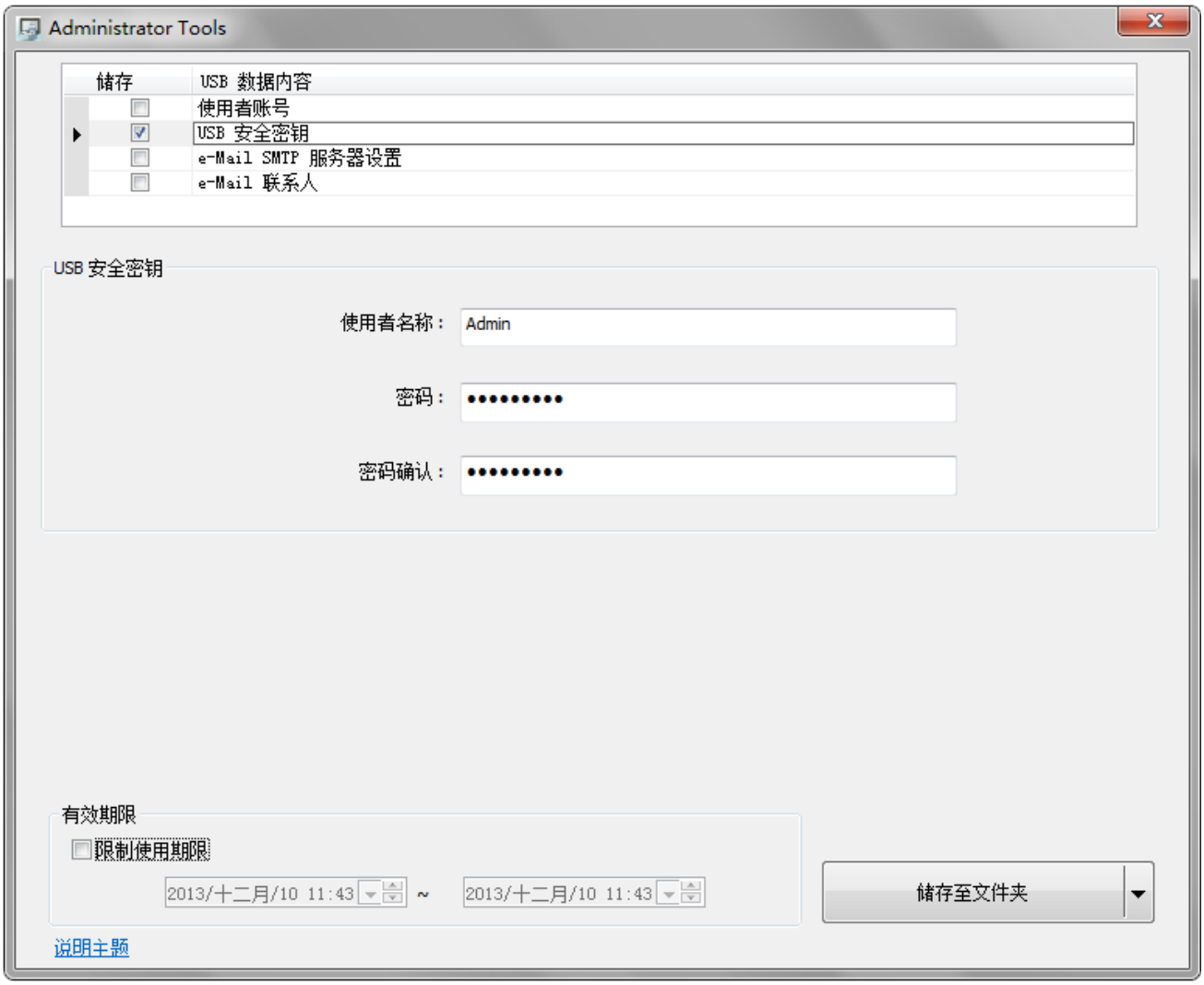


设置完成后，即可将外部设备插入 HMI，并执行此功能键来导入用户账号。

若账号导入方式选择 [覆盖]，则导入前会先清除系统中所有账户，导入完成后执行注销。若选择 [导入用户账号后删除文件]，在完成导入账号后系统将自动删除储存于外部设备的导入数据。若有在管理员工具 中指定 [有效期限]，则用户只能在指定的期限内执行导入动作。导入的账号并不会因为超过有效期限而被系统删除。

10.4.2 使用 USB 安全金钥登入

除了手动输入账号及密码进行账户登入外，用户亦可藉由一键登入来完成登入的动作，从 EasyBuilder Pro 安装目录下开启管理员工具后，勾选 [USB 安全金钥]，即可进行相关设置，透过预先设置好用户的登录信息，即可利用 USB 安全金钥直接登入账户，设置画面如下：

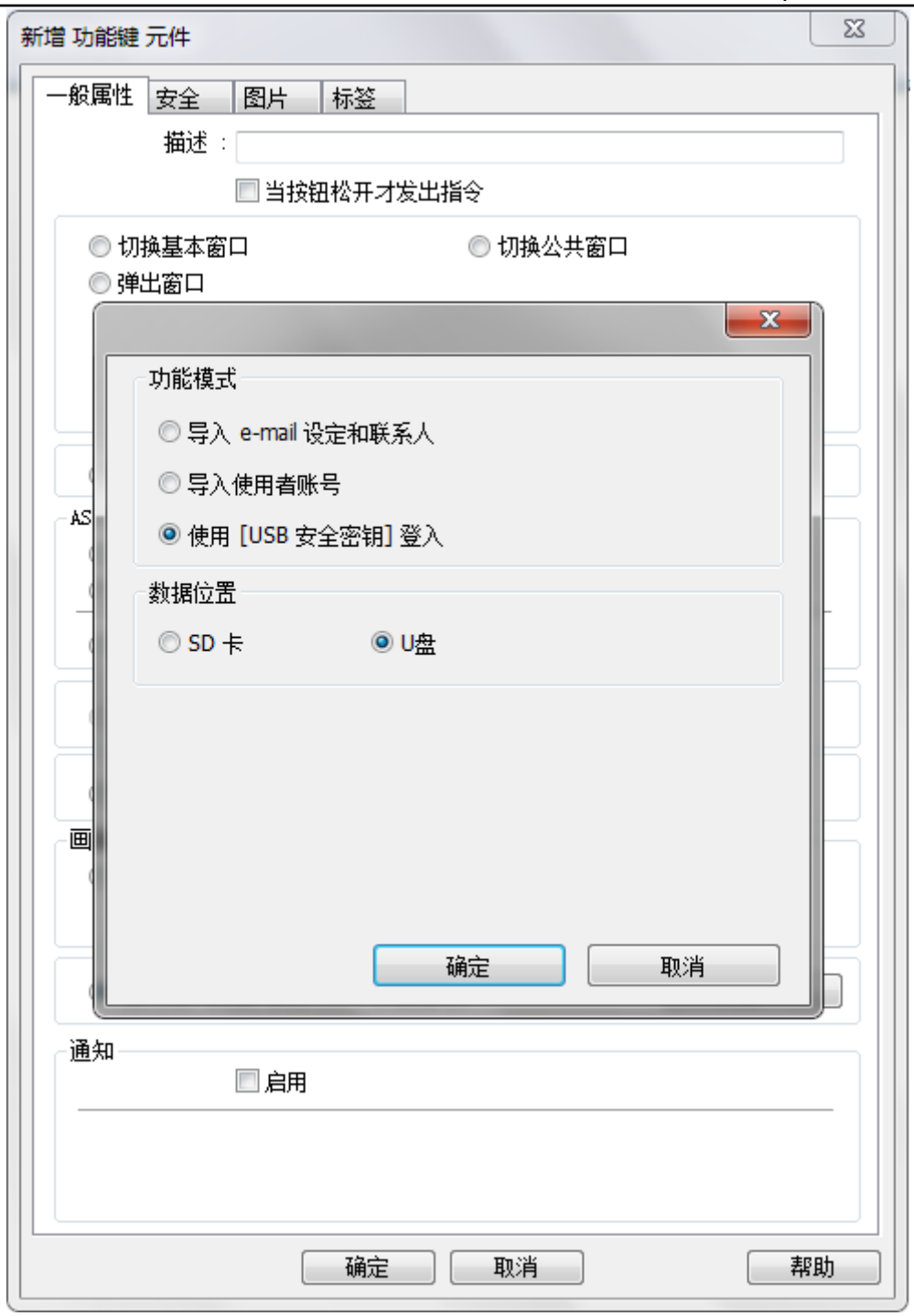


Note

- USB 安全金钥所欲设置的用户账号须为已存在于 HMI 程序上的用户账号。

关于管理员工具的详细信息请参考《第三十六章 管理员工具》。

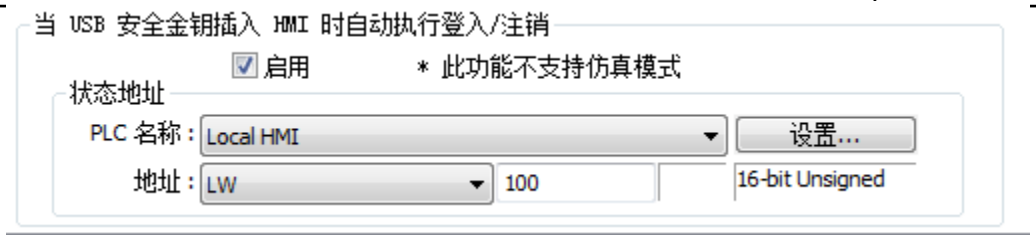
设置完 USB 安全金钥后可储存至 U 盘或 SD 卡，在程序内建立 [功能键] » [使用 [USB 安全金钥]]，设置如下：



设置完成后，即可将外部设备插入 HMI，并执行此功能键来一键登入 USB 安全金钥。若有在管理员工具 中指定 [有效期限]，则用户只能在指定的期限内执行登入动作，且系统会在金钥超出有效期限后自动注销。

10.4.3 使用 USB 安全金钥自动登入/注销

如下图，在 [系统参数设置] 勾选 [启用] 便可启用 USB 安全金钥自动登入及注销：



启用后，只要用户插入含有 USB 安全金钥的外部设备，就可以直接使用设置的账号登入。拔出装置后，账号将自动注销。登入/注销状态会自动写入至指定的 [状态地址]，其数值代表的意义为：

0x00: 无动作

0x01: 登入成功

0x04: 登入失败

0x08: 注销成功

0x10: 注销失败

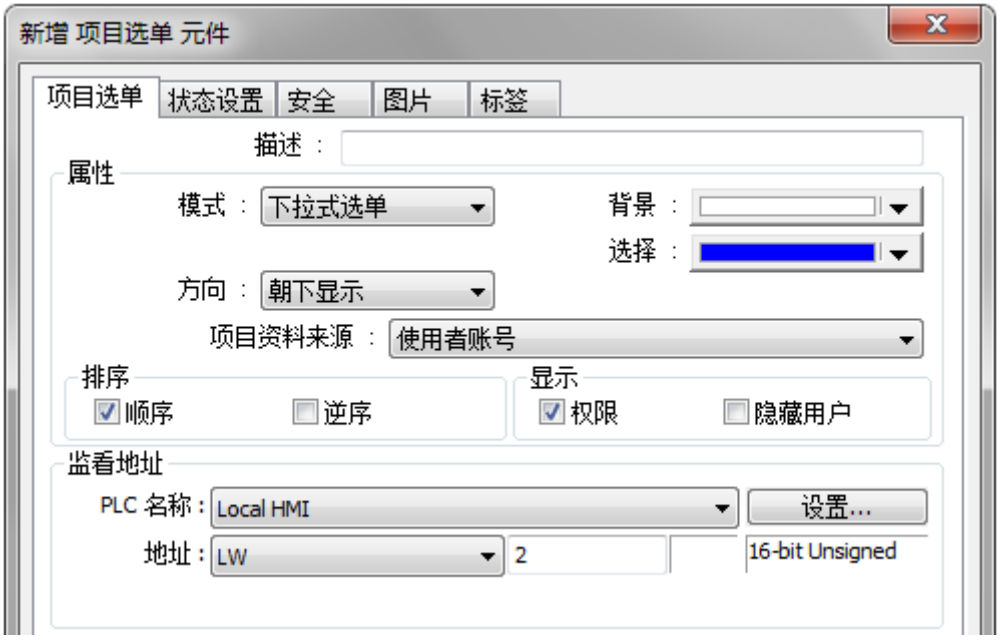
设置 USB 安全金钥的方法请参考《36 管理员工具》。

Note

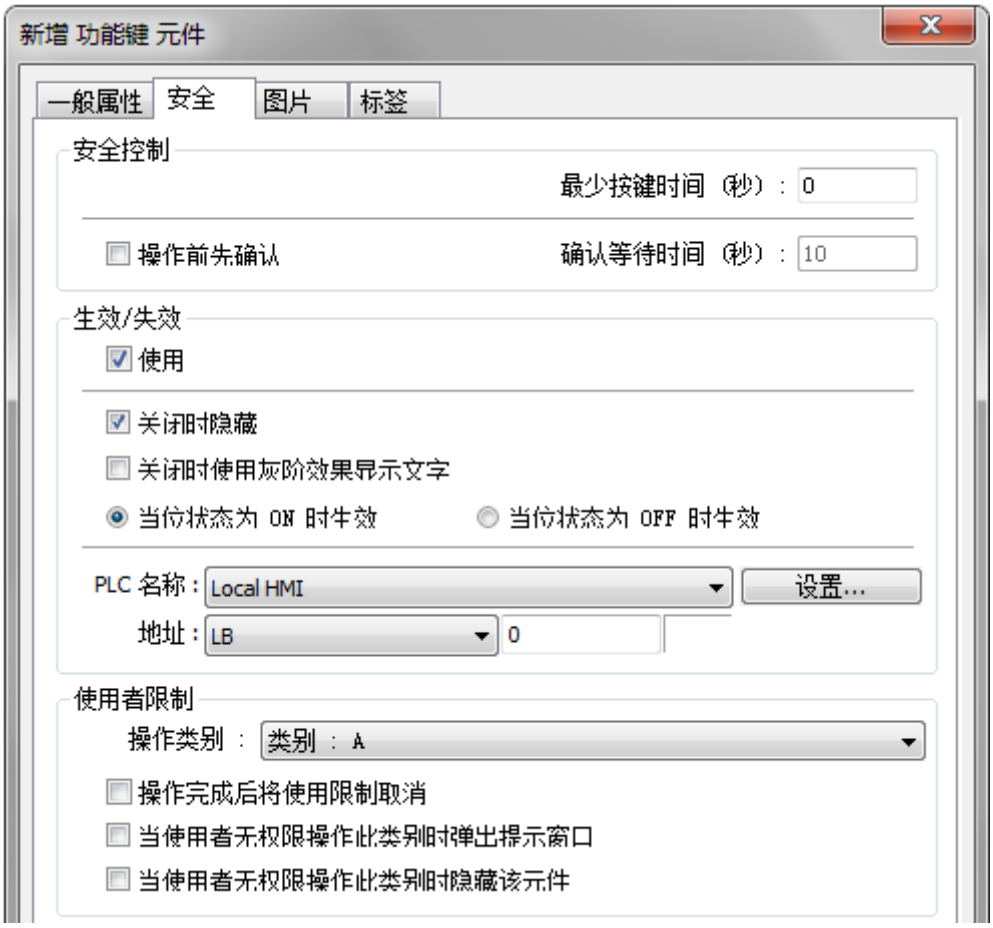
- 当启用 [当 USB 安全金钥插入 HMI 时自动执行登入/注销] 时，将无法使用 [功能键] 登入，但仍可使用控制地址登入及注销。
- 此功能无法在 PC 模拟模式下使用。
- 此功能只能使用储存在 U 盘中的 USB 安全金钥。

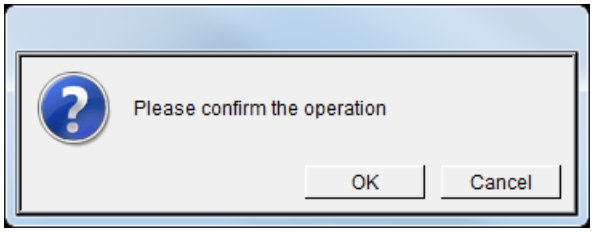
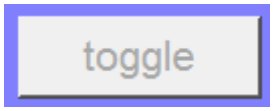
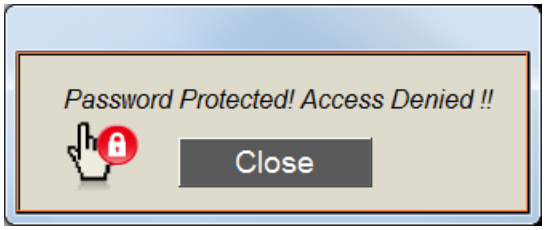
10.4.4 进阶安全模式搭配项目选单元件

进阶安全模式的控制地址 LW-n + 2 为用户索引，可搭配项目选单元件的用户账号来显示账号名称和权限。用户可以选择是否要在项目选单元件上显示账号权限和隐藏用户。隐藏用户是指在 [系统参数设置] » [用户密码] » [进阶安全模式] 下，可将用户账号名称隐藏起来，增加安全性。假设进阶安全模式的控制地址设置为 LW-0，则在此监看地址即为 LW-2。



10.5 元件操作安全防护



设置	描述
安全控制	**最少按键时间** 只有当持续按压元件时间大于此设置值才能成功启动操作。 **操作前先确认** 按压一个元件后，会出现一个确认对话窗，需点选 ok 来确定操作。如果等待确认操作的时间超过 [确认等待时间]，对话窗会自动消失并取消动作。 
开启/关闭	若勾选，此元件是否允许被操作，将决定于一个指定位地址的状态。如图，则必须在 LB-0 状态为 ON 时，才允许操作此元件。 **关闭时隐藏** 当指定的位地址于关闭状态时元件会被隐藏。 **关闭时使用灰阶效果显示文字** 元件的标签文字会在指定的位地址于关闭状态时以灰阶样式显示。 
用户限制	设置元件类别，只允许可操作此类别的用户操作。 **操作类别** “无”表示任何用户皆可操作。“管理者”表示只有 admin 账号可以操作。 **操作完成后将使用限制取消** 一旦用户的操作等级被允许操作该元件，系统便不再检查该元件的安全等级，也就是说，即使是别的用户，该元件也可被随意操作。 **当用户无权操作此类别时弹出提示窗口** 当用户操作身份不符合此元件的操作等级时，将弹出警告窗口 7 号。用户可自行设置此窗口上的提示文字。 

当用户无权操作此类别时隐藏此元件

当用户操作身份不符合此元件操作的等级时，元件会被隐藏。

10.6 元件安全防护范例

一般模式之元件安全防护的使用范例：

1. 建立一个工程文件，[系统参数设置] » [用户密码] » [一般模式] 中启用三个用户，例如：

   用户 1 = 操作元件类别 A

   用户 2 = 操作元件类别 A, B

   用户 3 = 操作元件类别 A, B, C

2. 在窗口 10 设计如下图所示:

建立两个 [数值输入] 元件：

[LW-9219] 用户编号 (1~12)，长度 = 1 word

[LW-9220] 输入用户密码，长度 = 2 words

建立一个 [数值显示] 元件：

[LW-9222] 显示目前用户的状态，格式为 16-bit Binary

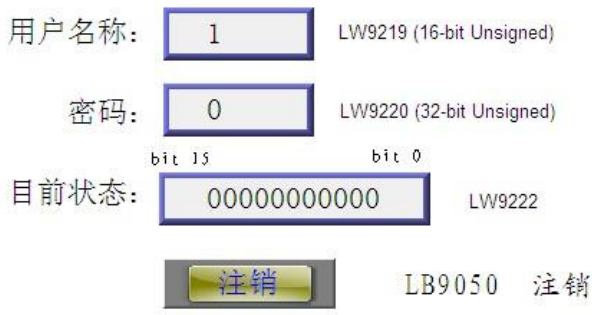
建立一个 [位状态设置] 元件：

[LB-9050] 用户注销

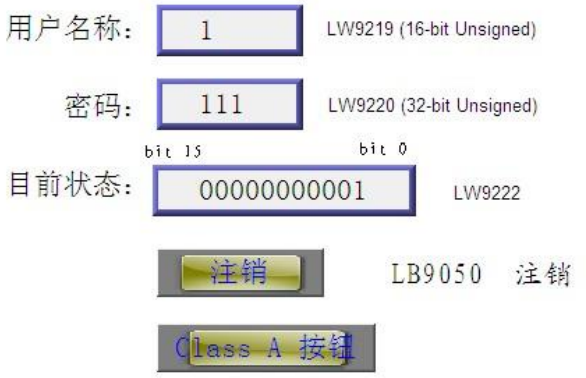
建立三个 [位状态设置] 元件：

选择不同的元件类别，但皆选择 [当用户无权操作此类别时隐藏此元件]。

3. 完成上述的各项设计并在存盘与编译后即可执行离线仿真功能，下图为离线仿真功能的起始画面。

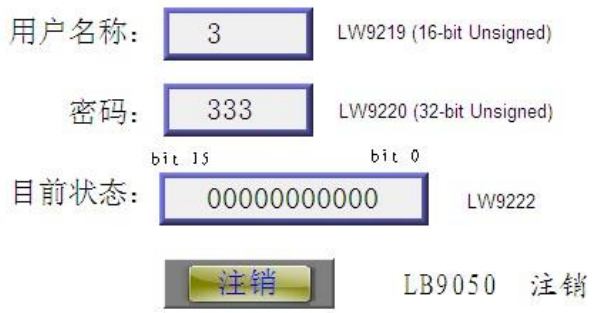


因尚未输入密码，数值元件显示“0000000000000000”，使用元件类别“无”。[Class A 按钮] ~ [Class C 按钮] 元件分别属于类别“A ~ C”并设置 [当用户无权操作此类别时隐藏此元件]，所以皆被系统所隐藏。



输入“用户 1”的密码(111)：
因设置“用户 1”允许操作类别 A 元件，所以此时 [Class A 按钮] 元件将出现并允许操作。[LW-9222] 的 bit 0 变为“1”，表示此时的用户允许使用类别“A”的元件。

输入“用户 3”的密码(333)，
因设置允许操作类别 A, B, C 元件，[LW-9222] 的 bit 0 ~ bit 2 的值皆变为“1”，表示此时的用户允许使用类别“A ~ C”的元件。



此时若按下 [注销] 强迫用户注销，可以发现系统将回复到起始状态，此时只允许操作类别为“无”的元件。

Note

- 密码输入：当密码输入错误时，[LB-9060] 的状态将被设置为 ON 状态；当密码输入成功时，[LB-9060] 的状态将自动被恢复为 OFF 状态。用户 1 至用户 12 所有用户的密码可以利用读取系统保留寄存器 [LW-9500] 至 [LW-9522]，共 24 words 的内容取得。
- 在线更改密码：当 [LB-9061] 的状态设置为 ON 时，系统将读取 [LW-9500] 至 [LW-9522] 内的数值，更新用户的密码，往后并使用这些新的密码。此时用户可操作类别的元件并不会因密码的变更而改变。

第十一章 索引寄存器

11.1 概要

索引寄存器是 EasyBuilder Pro 提供用于变换地址的寄存器。有了索引寄存器后，用户可以在不改变元件地址内容的情况下，在 HMI 上直接修改元件的读取与写入地址。EasyBuilder Pro 提供 32 组索引寄存器，分别为 16 组 16-bit 的索引寄存器和 16 组 32-bit 的索引寄存器。

LW-9200 (16bit) : 地址索引寄存器 0	LW-9230 (32bit) : 地址索引寄存器 16
LW-9201 (16bit) : 地址索引寄存器 1	LW-9232 (32bit) : 地址索引寄存器 17
LW-9202 (16bit) : 地址索引寄存器 2	LW-9234 (32bit) : 地址索引寄存器 18
LW-9203 (16bit) : 地址索引寄存器 3	LW-9236 (32bit) : 地址索引寄存器 19
LW-9204 (16bit) : 地址索引寄存器 4	LW-9238 (32bit) : 地址索引寄存器 20
LW-9205 (16bit) : 地址索引寄存器 5	LW-9240 (32bit) : 地址索引寄存器 21
LW-9206 (16bit) : 地址索引寄存器 6	LW-9242 (32bit) : 地址索引寄存器 22
LW-9207 (16bit) : 地址索引寄存器 7	LW-9244 (32bit) : 地址索引寄存器 23
LW-9208 (16bit) : 地址索引寄存器 8	LW-9246 (32bit) : 地址索引寄存器 24
LW-9209 (16bit) : 地址索引寄存器 9	LW-9248 (32bit) : 地址索引寄存器 25
LW-9210 (16bit) : 地址索引寄存器 10	LW-9250 (32bit) : 地址索引寄存器 26
LW-9211 (16bit) : 地址索引寄存器 11	LW-9252 (32bit) : 地址索引寄存器 27
LW-9212 (16bit) : 地址索引寄存器 12	LW-9254 (32bit) : 地址索引寄存器 28
LW-9213 (16bit) : 地址索引寄存器 13	LW-9256 (32bit) : 地址索引寄存器 29
LW-9214 (16bit) : 地址索引寄存器 14	LW-9258 (32bit) : 地址索引寄存器 30
LW-9215 (16bit) : 地址索引寄存器 15	LW-9260 (32bit) : 地址索引寄存器 31

16-bit 地址索引寄存器 0 至 15 的对应地址为 LW-9200 (16bit) 至 LW-9215 (16bit)，其最大偏移量为 65536 words。
32-bit 地址索引寄存器 16 至 31 的对应地址为 LW-9230 (32bit)至 LW-9260 (32bit)，其最大偏移量为 4294967296 words 。

使用 [索引寄存器] 后，所使用 [设备类型] 的地址则由下列公式决定：
“设置的常数地址 + 所选择索引寄存器中的值”

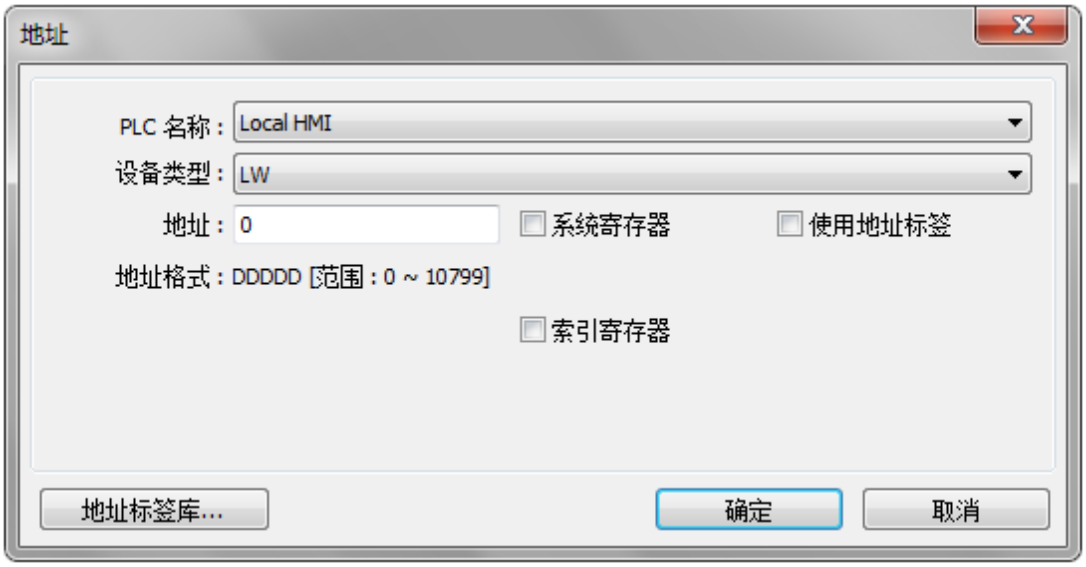
Note

- 索引寄存器可使用于所有设备的字符格式的地址寄存器。若使用于位格式的地址寄存器，则当索引寄存器中的数据每改变 1 会偏移 16 个位地址。

11.2 使用索引寄存器范例

以下为实际存取地址的计算过程：

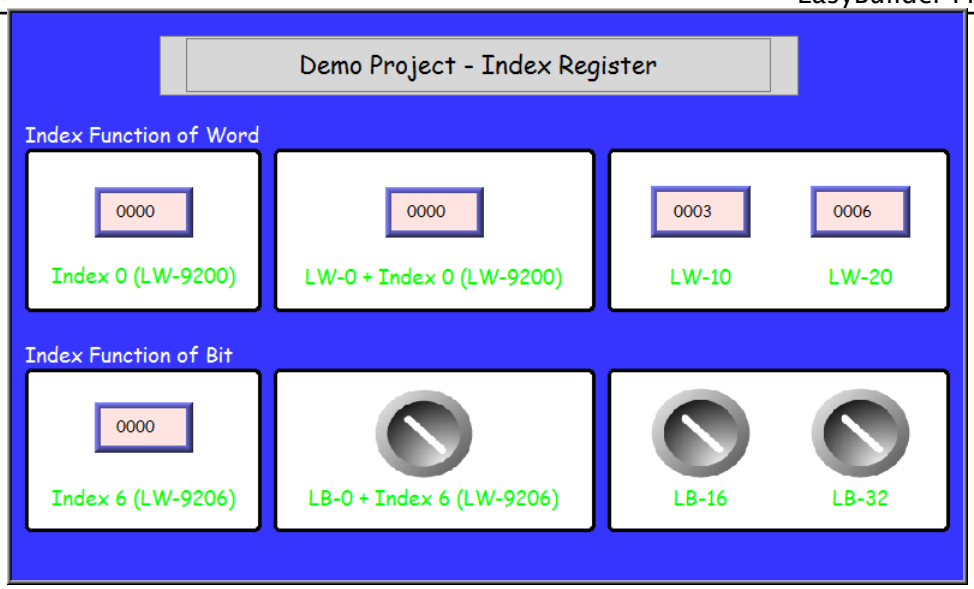
若未勾选 [索引寄存器] 并设置地址为 LW-10，系统则对此地址做读取/写入的动作。



若勾选 [索引寄存器] 并选择 [索引] 为 [INDEX0 (16bit)]，则存取地址为 [LW-(10 +地址索引寄存器 0 的值)]。

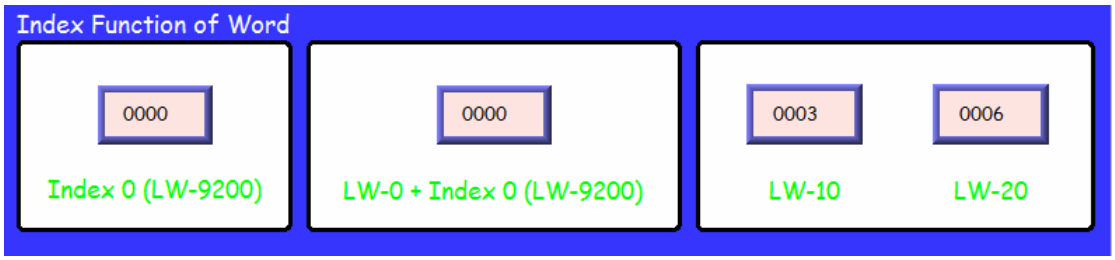
例如：[LW-9200] 地址中的数值为 “5”，根据计算公式可得出实际存取地址为 [LW-(10+5)]，即 [LW-15]。

以实际范例作进一步说明：

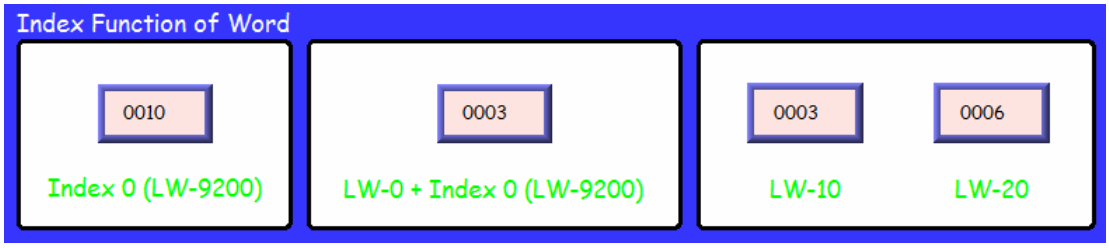


范例 1

下图表示使用索引寄存器的字符格式地址。假设 [LW-0] 的值为 “0”，[LW-10] 的值为 “3”，[LW-20] 的值为 “6”，则结果如下：



若 Index 0 (LW-9200) 地址中的数据为 “0”，则 [LW-0 + Index 0] = 读取 [LW-0] 内容。



若 Index 0 (LW-9200) 地址中的数据设为 “10”，则 [LW-0 + Index 0] = 读取 [LW-10] = “3”。

范例 2

下图表示使用索引寄存器的位格式地址。

由于 1 个字符 = 16 个位，所以索引寄存器数值改变 1 相当于偏移 16 个位。假设 [LB-16] 为 ON，而 [LB-32] 为 OFF，则结果如下：

若 Index 6 (LW-9206) 地址中的数据设为 “1”，则开关 [LB-0 + Index 6] 读取 LB-16 地址状态，也就是 ON 的状态。

若 Index 6 (LW-9206) 地址中的数据设为 “2”， 则开关 [LB-0 + Index 6] 读取 LB-32 地址状态，也就是 OFF 的状态。

Note

- 使用索引寄存器于位地址时，所设置的位地址将会以 16 个位地址为一个偏移单位。假设以 LB-0 为范例且使用索引寄存器，若是索引寄存器里的数值为 1，则 LB-16 将会动作，若是索引寄存器里的数值为 2，则 LB-32 会动作。

第十二章 键盘的设计与使用

12.1 概要

数值输入与字元输入元件都需要使用键盘做为输入工具，而数字键盘以及字符键盘均是使用功能键元件来制作的。除了使用 EasyBuilder Pro 提供的内建键盘外，您也可以自行设计键盘。键盘种类可分为：

- 弹出键盘 (可选择是否使用窗口控制条)
- 固定键盘
- UNICODE 文字键盘

Note

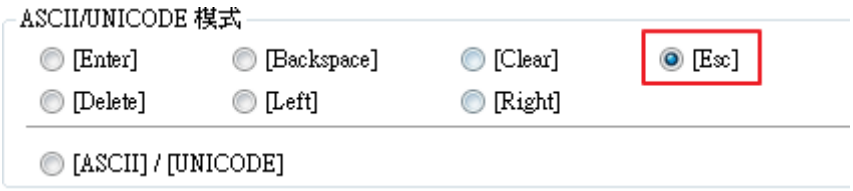
- cMT-SVR 系列采用 iPad 内建键盘，无法自行设计键盘，因此本章节皆不适用于 cMT-SVR 系列。

12.2 设计弹出键盘

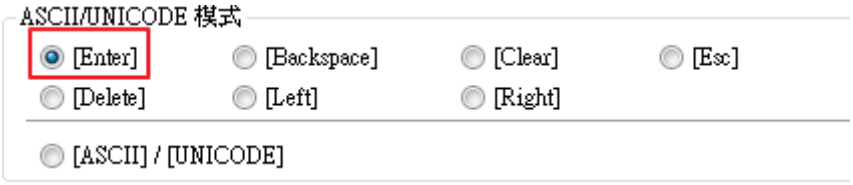
1. 先建立并开启要作为键盘的窗口，假设为窗口 200。

2. 调整窗口 200 的长度与宽度，建立各个功能键元件，并用 [ASCII/UNICODE 模式]。

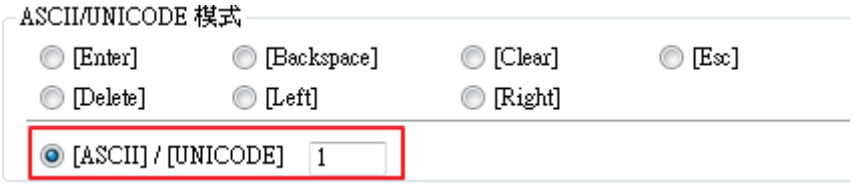
将其中一个功能键用来触发取消讯号 [ESC]。



另一个功能键用来触发输入讯号 [Enter]。



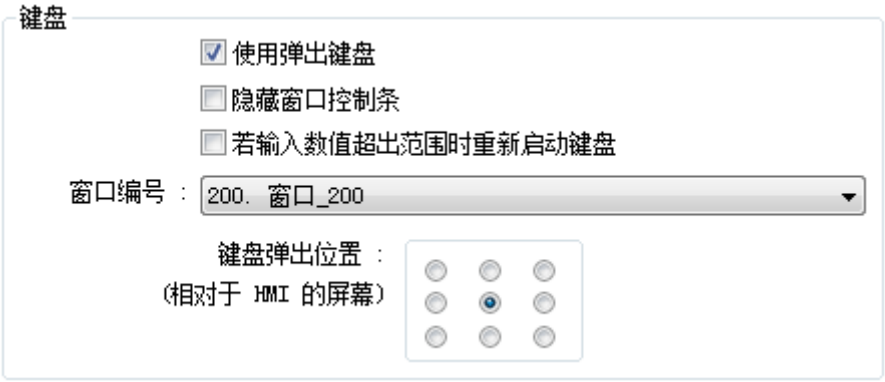
其它大部分功能键用来触发数值输入讯号，例如用来触发数值 1 的输入讯号。



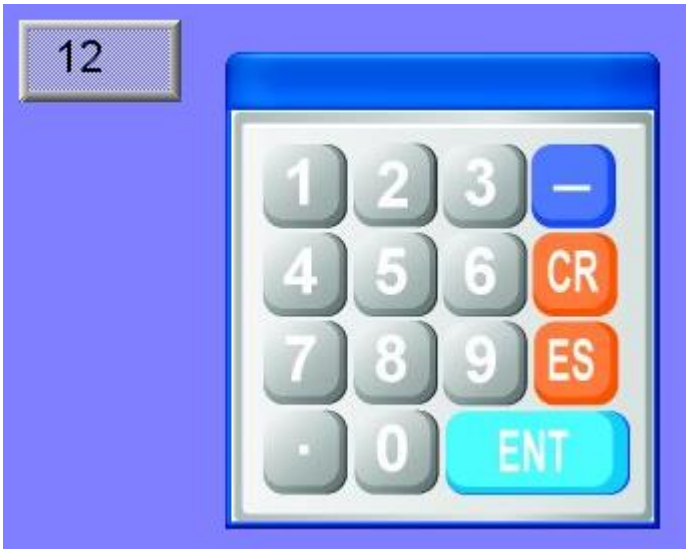
3. 最后为功能键挑选适合的图形做为图片元件，并置于所有元件最下层作为背景图案。

4. 在 [系统参数设置] » [一般属性] » [键盘] 设置中，按下 [新增] 后选择加入 [窗口 200]。最多可新增 32 个键盘窗口。

5. 在完成上述的所有步骤后，当使用数值输入与字元输入元件的设置页时，即可发现在 [数值输入] » [键盘] 设置中的 [窗口编号]，增加了 “200. Keyboard" 的选项。[键盘弹出位置] 可用来选择键盘在 HMI 的出现位置，系统将 HMI 屏幕划分为 9 个区域，如下图所示。

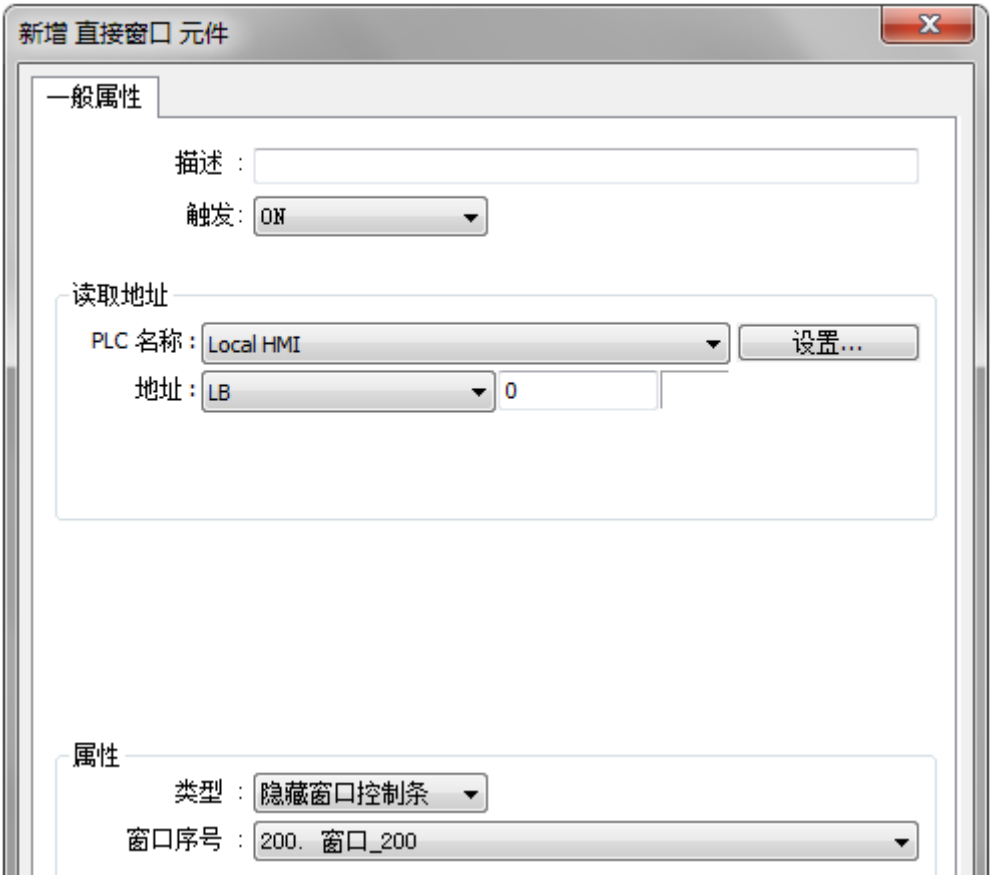


6. 在选择 [200. Keyboard] 后，当按下数值输入或字符输入元件时，将自动弹出窗口 200，按键盘上的功能键就可以输入数值。



12.3 使用直接窗口的方式设计键盘

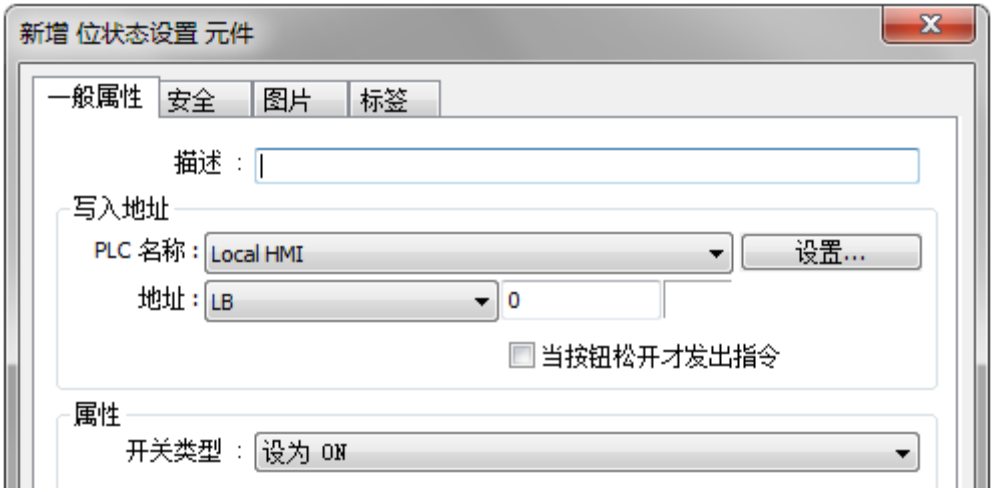
1. 新增一个直接窗口元件，设置读取地址 LB-0 来启动直接窗口。在 [属性] 内选择 [隐藏窗口控制条] 及设置键盘所在 [窗口序号]。



2. 在 [轮廓] 页签将元件尺寸设置等同键盘窗口的大小。

3. 新增数值输入元件，在数值输入页内不要勾选 [使用弹出键盘]。

4. 设置一个位状态设置元件，写入地址为 LB-0，开关类型为 [设为 ON]，并重叠在数值输入元件上。当按下数值输入元件的同时，也会将键盘窗口开启。



5. 在键盘的 [Enter] 功能键和 [ESC] 功能键上，分别放置一个位状态设置元件，写入地址为 LB-0，开关类型为 [设为 OFF]。在按下这两个键的任意一个键时，可将键盘窗口关闭。

12.4 将键盘固定在需要输入的窗口上

若不采用弹出键盘方式或是使用直接窗口来预设键盘所在位置，可采用此固定键盘方式，但这方式将无法移动或关闭键盘。

1. 新增数值输入元件，在 [数值输入] » [键盘] 属性中不要勾选 [使用弹出键盘]。
2. 使用功能键元件将键盘按键设计好后，放置于窗口上即可使用。
3. 当按下数值输入元件时，可以藉由键盘上的功能键来输入数值。

12.5 制作 UNICODE 键盘

本节说明如何使用功能键元件制作 Unicode 键盘：

1. 放置一个字符输入元件在窗口上，并勾选 [使用 UNICODE]。
2. 制作 「威」、「纶」、「科」、「技」 这四个文字输入功能键，再制作一个 [Enter] 输入功能键，即完成一个简单的文字键盘。

Note

■ 您可以将自制的键盘设置群组为「群组图片」并添加到「群组图库」中，以便于后续的调用。

第十三章 元件

13.1 位状态指示灯

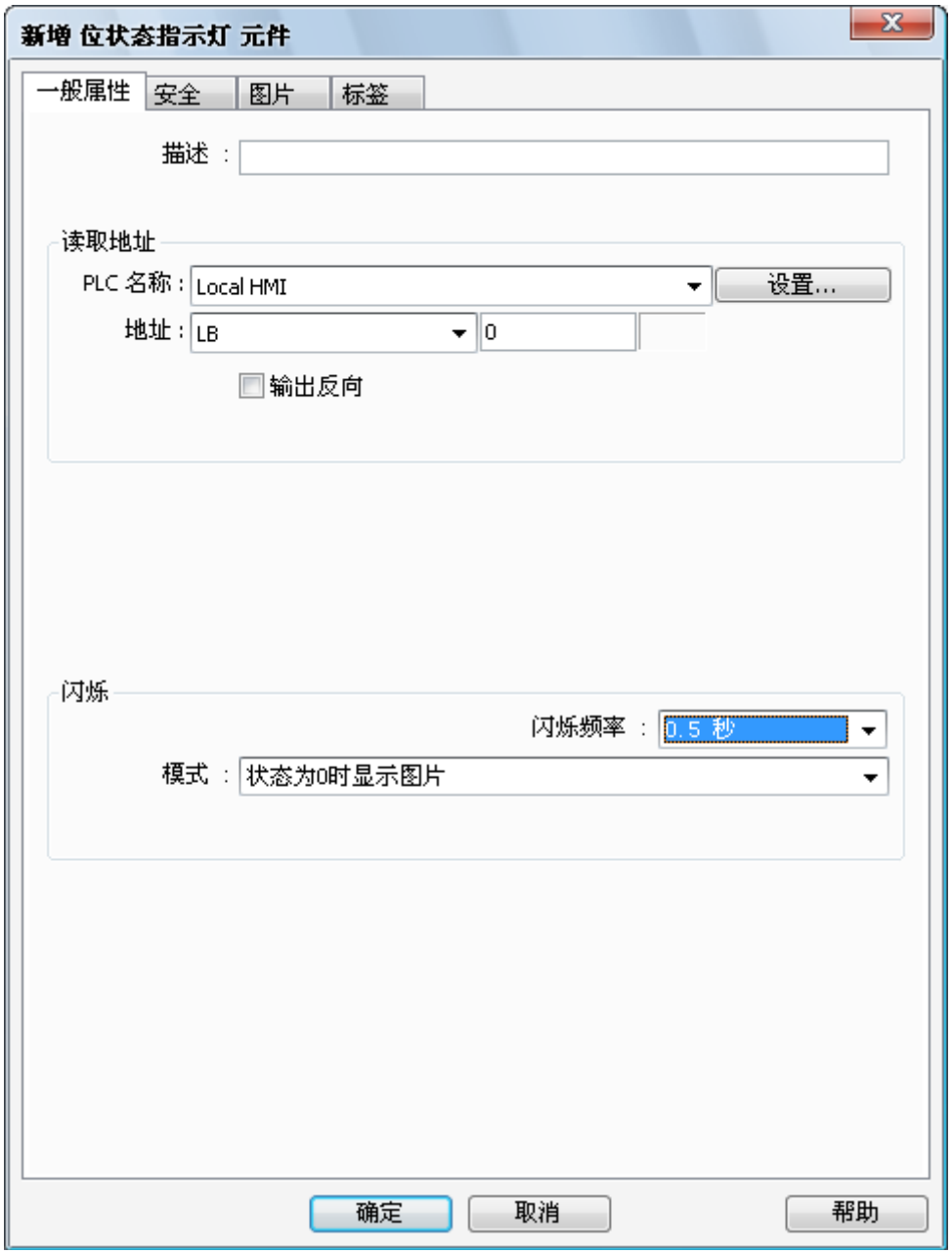
13.1.1 概要

“位状态指示灯”元件用来显示位寄存器的状态。状态 0 代表位的状态为 OFF；状态 1 代表位的状态为 ON。

13.1.2 设定

按下工作列上的“位状态指示灯”按钮后即会开启“位状态指示灯元件属性对话框”，正确设定各项属性后按下确认键，即可新增一个“位状态指示灯”元件。

一般属性设定



设定	描述
描述	用户可为此元件描述相关信息。
读取地址	点选“设置”后选择位寄存器设备类型的“PLC 名称”，”地址”，”设备类型”，”系统寄存器”，”索引寄存器”来控制位状态指示灯元件。用户也可在“一般属性”页中设定地址。 **输出反向** 可以将读取的状态作反向显示，例如位的状态实际上为 OFF，但勾选了“输出反向”后会显示为 ON。
闪烁	设定元件在位状态为 ON 或 OFF 的显示方式。

模式：

无

不闪烁。

状态为 0 时显示图片

状态为 OFF 时，图片会进行状态 0 及状态 1 交互闪烁。

状态为 1 时显示图片

状态为 ON 时，图片会进行状态 0 及状态 1 交互闪烁。

状态为 0 时闪烁

状态为 OFF 时，图形 0 会进行出现与消失交互动作。

状态为 1 时闪烁

状态为 ON 时，图形 1 会进行出现与消失交互动作。

当目前状态无相对应的图片时，不显示图片

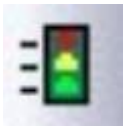
若勾选，当图片数目不足以显示全部状态时，将不显示图片。反之则显示最后一个状态。

13.2 多状态指示灯

13.2.1 概要

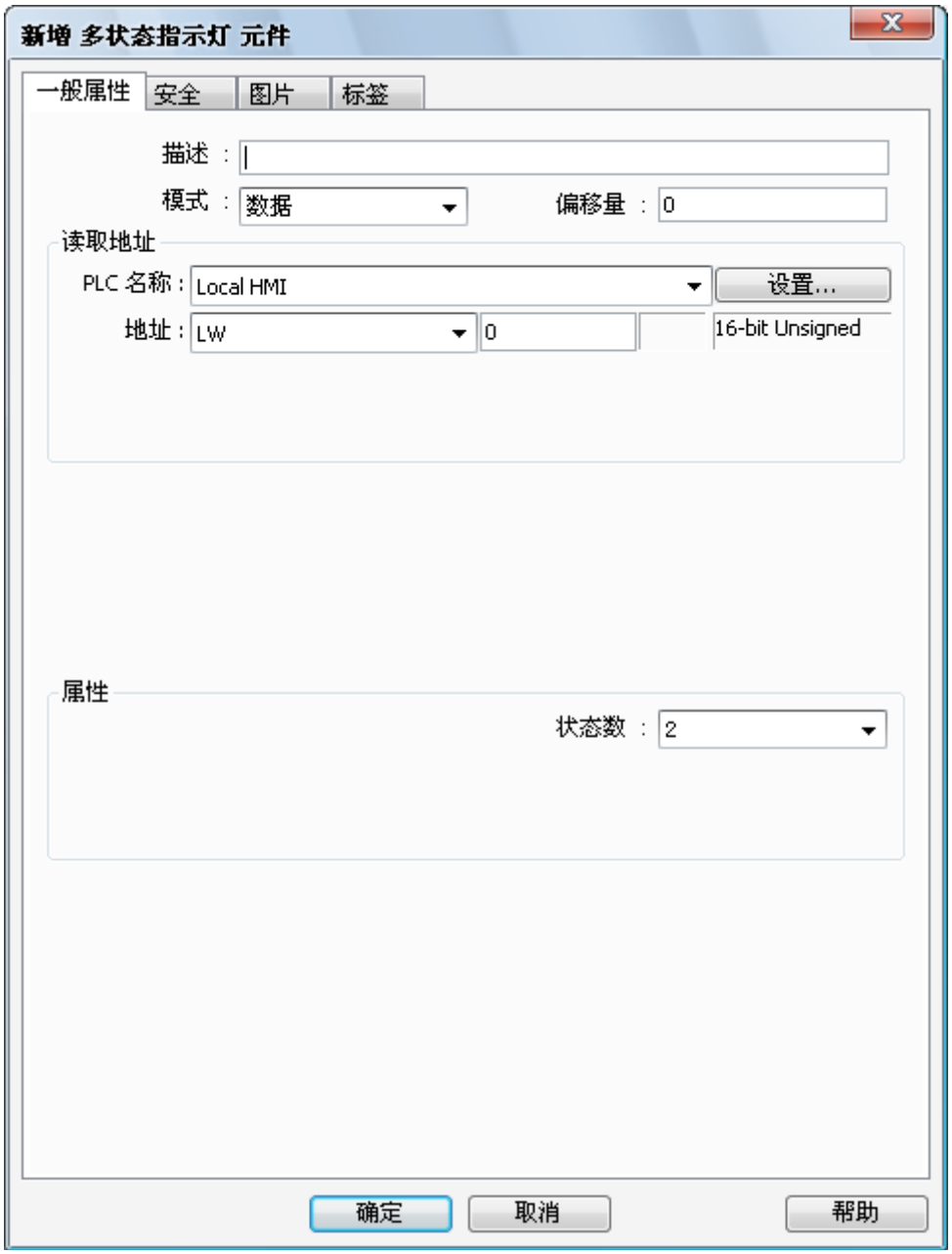
“多状态指示灯”元件利用字符寄存器内的数据，显示相对的状态与图形(最多可支持 256 种状态的显示)。当寄存器内的数值为 0 时，显示“状态 0”；当数值为 1 时，则显示“状态 1”，依此类推。

13.2 设定

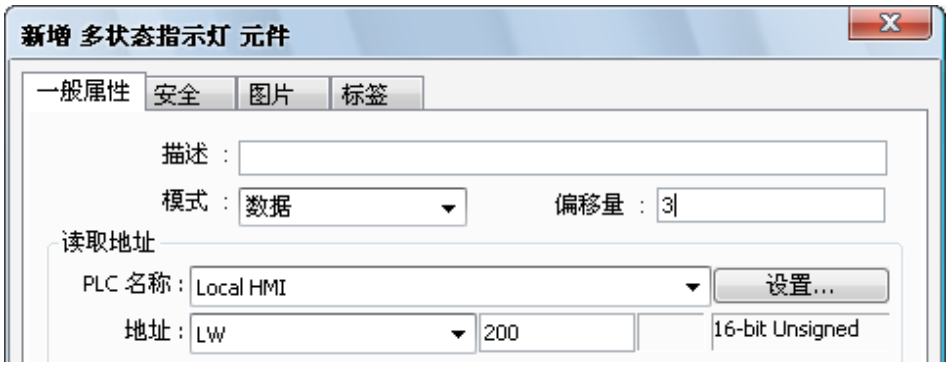


按下工作列上的“多状态指示灯”按钮后即会开启“多状态指示灯元件属性对话框”，正确设定各项属性后按下确认键，即可新增一个“多状态指示灯”元件。

一般属性设定



设定	描述
描述	用户可为此元件描述相关讯息。
模式 / 偏移量	“多状态指示灯”元件提供三种不同的数据显示模式。 **数据** 直接利用寄存器内的数据减“偏移量”的结果做为元件目前的状态。如下图所示，当写入一个数值 3 至寄存器 LW-200 时，因为有偏移量 3，所以地址 LW-200 的元件图片会显示状态 0。



LSB

此模式首先会将寄存器内的数据先转换为 2 进制，接着使用不为 0 的最低位决定元件目前的状态。以下表数据为例：

十进制	二□制	显示的状态
0	0000	全部 bit 皆为 0，则显示状态 0
1	0001	不是 0 的最低位为 bit 0，此□显□状态 1
2	0010	不是 0 的最低位为 bit 1，此时显示状□2
3	0011	不是 0 的最低位为 bit 0，此□显示状态 1
4	0100	不是 0 的最低位为 bit 2，此时显示状态 3
5	0101	不是 0 的最低位为 b□t 0，此时显示状态 1
6	0110	不是 0 的最低位□bit 1，此时显示状态□2
7	0111	不是 0 的最低位为 bit 0，此时显示状态 1
	□000	不是 0 的最低位为 bit 3，此时显示状态 4

周期转换状态

元件的状态会依照固定的频率依序变换状态。用户可以利用“频率”设定状态改变频率。

读取地址

点选“设置”后选择字符寄存器设备类型的“PLC 名称”,”地址”,”设备类型”,”系统寄存器”,”索引寄存器”来控制多状态指示灯元件。用户也可在“一般属性”页中设定地址。

属性

状态数

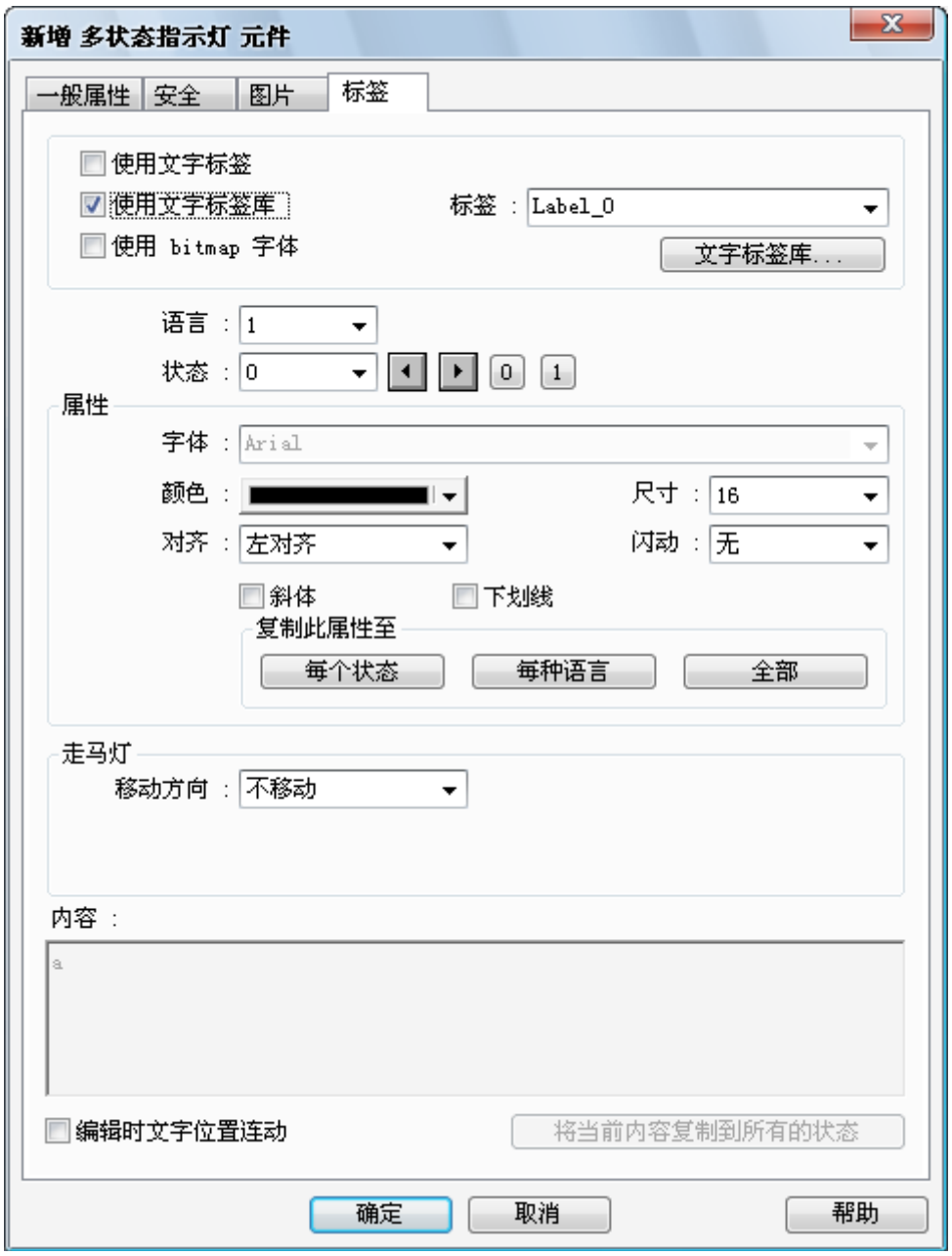
元件显示的状态数目。状态从 0 开始编号，能显示的最大状态编号为设定的“状态数”－1，当要求显示超过设定的状态数时，系统会显示最后一个状态。例如设定“状态数”为 8，则显示的状态依序为 0，1，2，…，7，若要求寄存器显示状态 8 (含) 以上的状态时，显示的图片仅显示状态 7。

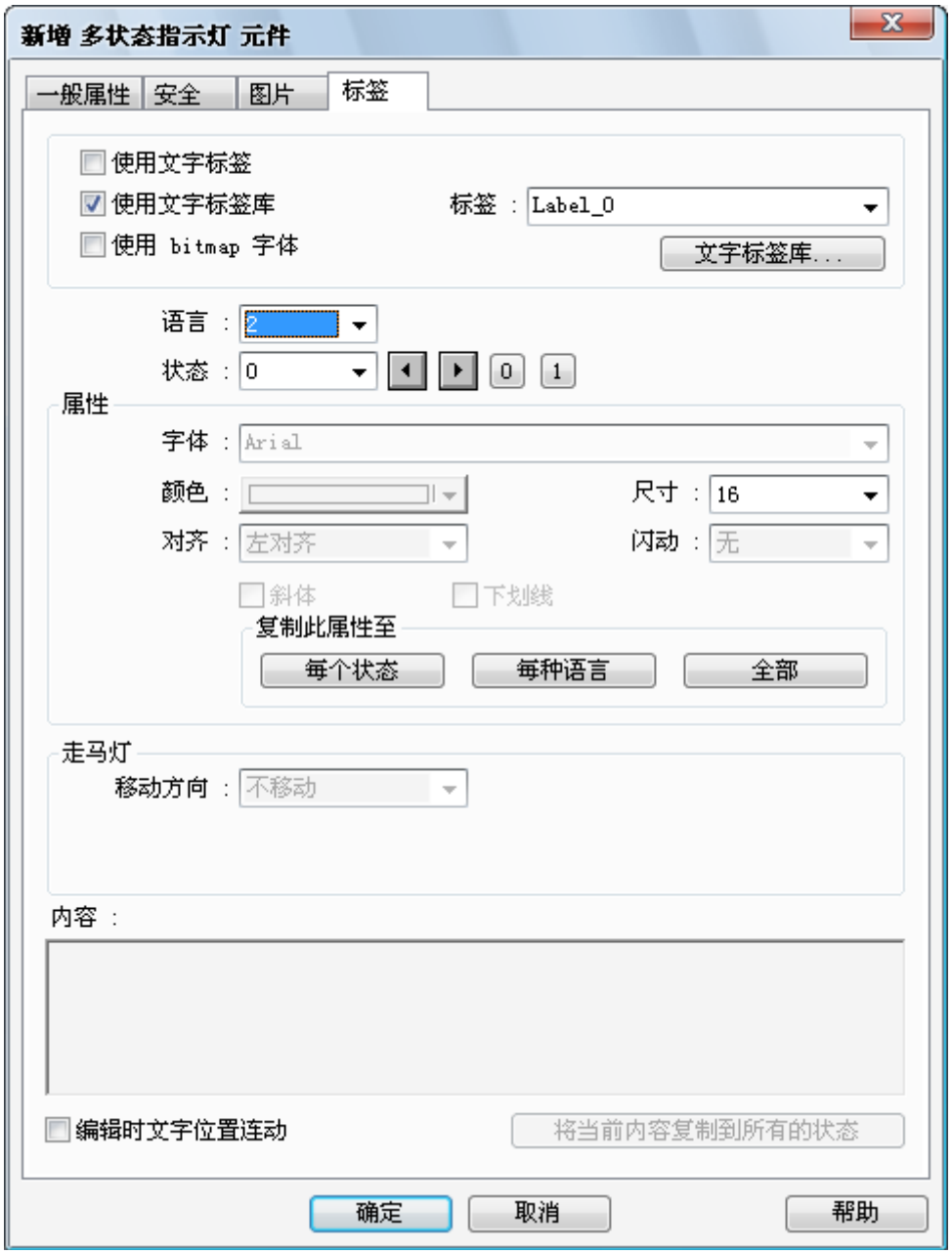
当目前状态无相对应的图片时，不显示图片

若勾选，当图片数目不足以显示全部状态时，将不显示图片。反之则显示最后一个状态。

Note

- 在标签页中，语言 1 能够改变字型相关属性设定，但语言 2~8 只能改变字的尺寸，其它属性设定皆与语言 1 相同。



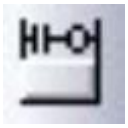

新增 多状态指示灯 元件
一般属性
安全
图片
标签
使用文字标签
使用文字标签库
使用 bitmap 字体
标签 :
Label_0
文字标签库...
语言 :
2
状态 :
0
0
1
属性
字体 :
Arial
颜色 :
尺寸 :
16
对齐 :
左对齐
闪动 :
无
斜体
下划线
复制此属性至
每个状态
每种语言
全部
走马灯
移动方向 :
不移动
内容 :
编辑时文字位置连动
将当前内容复制到所有的状态
确定
取消
帮助

13.3 位状态设定

13.3.1 概要

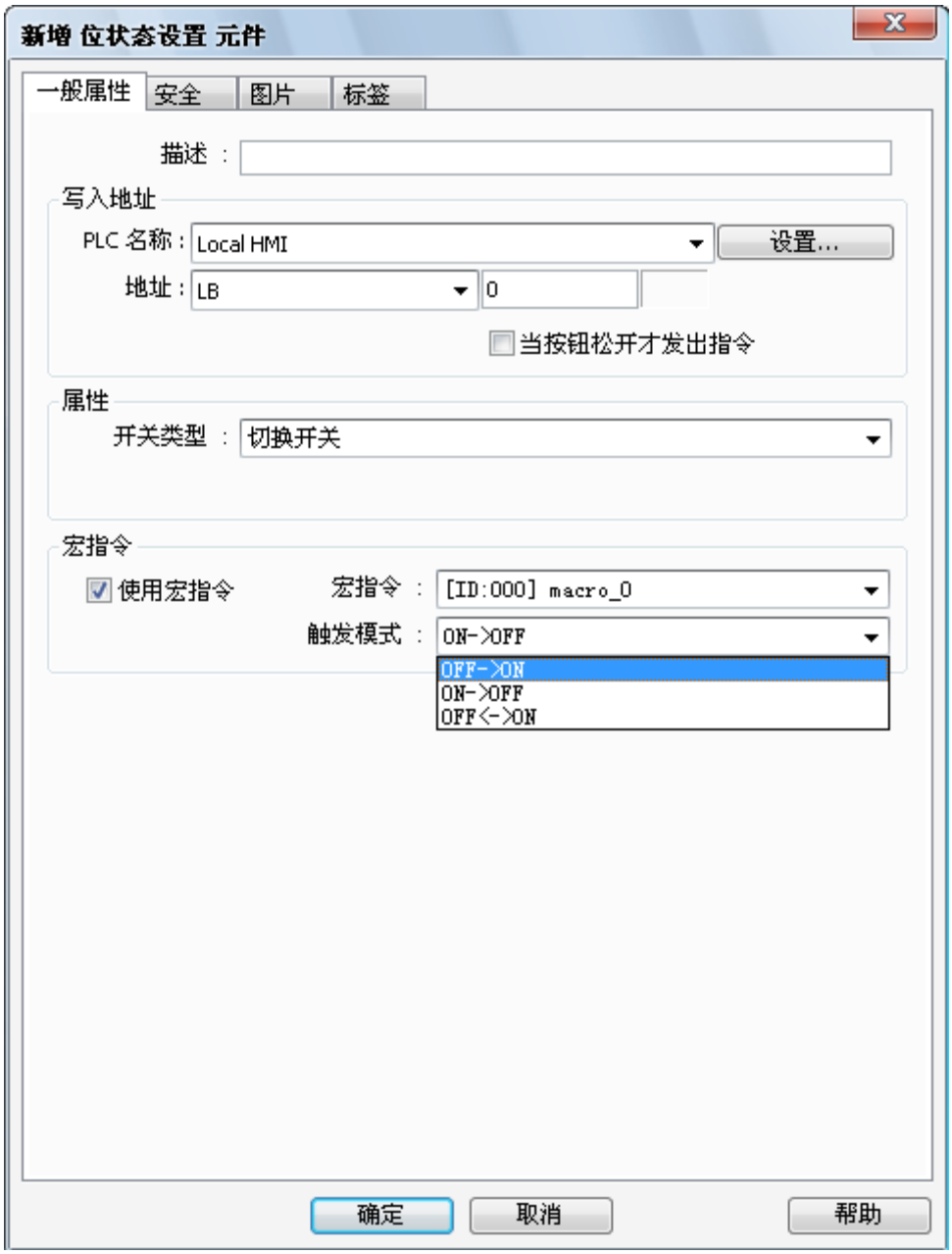
“位状态设置”用于设定位寄存器的状态。此元件提供手动操作与自动执行两种操作模式。使用手动操作模式，按压此按钮可以将寄存器的状态设定为 ON 或 OFF。
若使用自动执行模式，则在某些特定条件下会自动执行指定的动作，使用此种操作模式，即使按压此按钮也不会有任何影响。

13.3.2 设定



按下工作列上的“位状态设置” 按钮后即会开启“位状态设置” 元件属性对话框，正确设定各项属性后按下确认键，即可新增一个“位状态设置” 元件。

一般属性设定



设定	描述
写入地址	点选“设置”后选择位寄存器设备类型的“PLC 名称”，”地址”，”设备类型”，”系统寄存器”，”索引寄存器”来控制位状态设定元件。用户也可在“一般属性”页中设定地址。 **当按钮松开才发出指令** 使用此设定表示在按下元件后，必须完全松开按压动作，元件定义的操作模式才会被执行。如未使用此项设定，只要一碰触此区域，将立刻执行元件的动作。若选择使用复归型模式，将不支持此项功能。

属性

开关类型	描述
设□ON	按压此元件后，所指定寄存器的状态将□设定为 ON。
设为 OFF	按□此元件后，所指定寄□器的状态将被设定将为 OFF。
切换开关	按压此元件后，所指定寄存器的状态将被反向，即 ON→OFF, OFF→ ON。
复归型	按压此元件后，所指定寄存器的状态将先被设定为 ON，但手放开后，状态将被设定为 OFF。
周□切换开	所指定寄存器的状态将在 ON 与 OFF 间周期性切换，此模式为自动执行。可设定的周期为 0.1 秒 ~ 25.5 秒。
窗口打□时设 ON	元件所在位置的窗口被打开时，所指定寄存器的状态将自动被设定为 ON。
窗口打开时设 OFF	元件所在位置的窗口被打开时，所指定寄存器的状态将自动被设定为 OFF。
窗口关闭时设 ON	元件所在位置的窗口被关闭时，□指定寄存器的状态将自□被设定为 ON。
窗口关闭时设 OFF	元件所在位置的窗口被关闭时，所指定寄存器的状态将自动被设定为 OFF。
当背光灯开时设 ON (不支持 cMT□SVR□列)	当背光灯打开时，所指定寄存器的状态将自动被设定为 ON。
当背光灯开时设 OFF (不支持 cMT-SVR 系列)	当背光灯打开时，所指定寄存器的状态将自动被设定为 OFF。
当背光灯关时设 ON (不支持 cMT-SVR 系列)	当背光灯关闭时，所指定□存器的状态将自动被设定为 ON。
当背光灯关时设 OFF (不支持 cMT-SVR 系列)	当背光灯关闭时，所指定寄存器的□态将自动被设定为 OFF。

宏指令

“位状态设置” 元件可以搭配执行宏命令。选择此项功能前需先建立宏命令。

如何建立宏命令请参考《18 宏指令说明》

触发模式

当元件的操作模式，选择“切换开关”时，设定执行宏命令的条件，可以选择状态由 OFF 变为 ON 或由 ON 变为 OFF 时，才执行宏命令，也可选择状态改变时(OFF<->ON)，即执行宏命令。

13.4 多状态设定

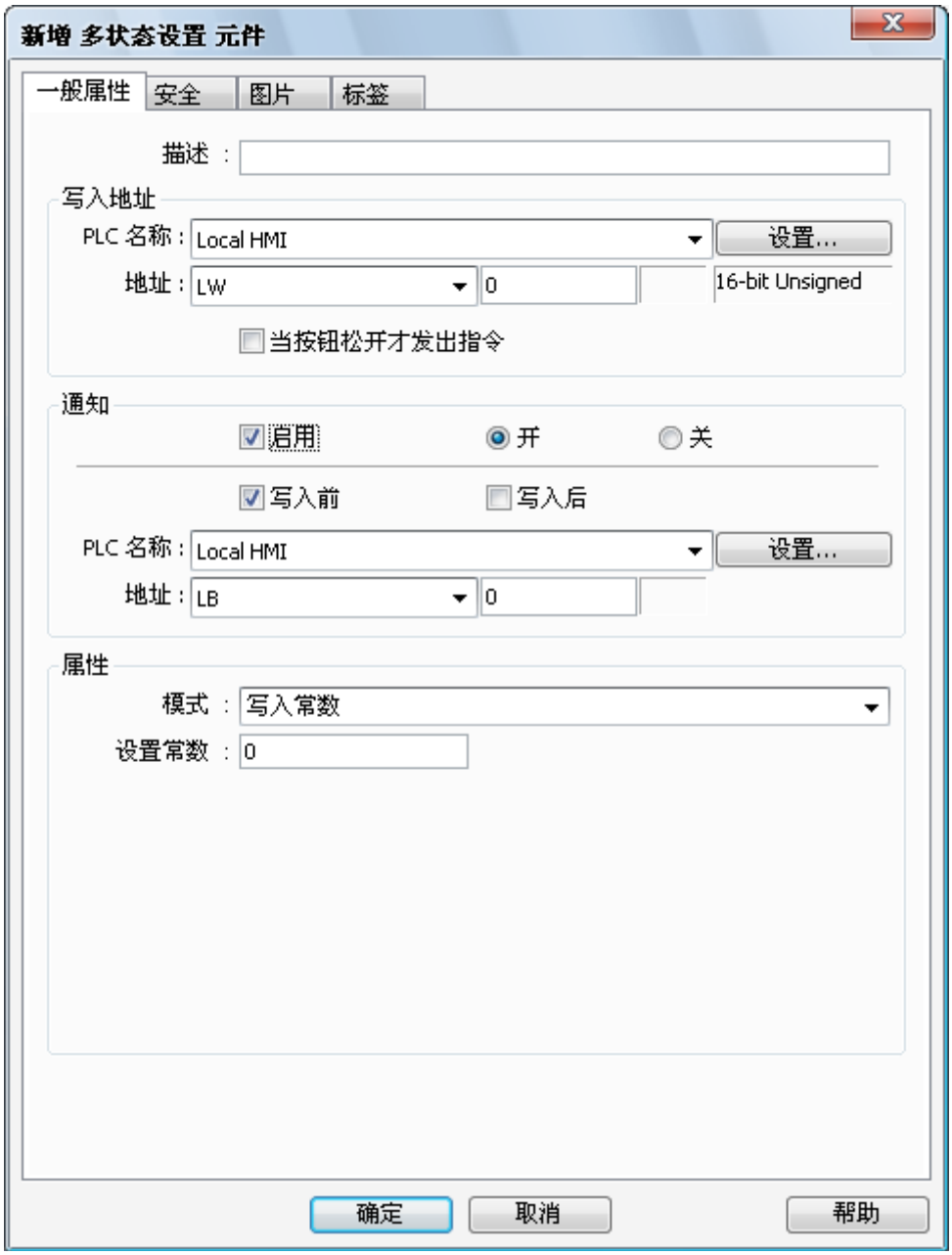
13.4.1 概要

“多状态设置”用于设定字符寄存器的数据。此元件提供手动操作与自动执行两种操作模式。使用手动操作模式，按压此按钮可以设定寄存器内的数据。
若使用自动执行模式，则在某些特定条件下会自动执行指定的动作，使用此种操作模式，即使按压此按钮也不会有任何影响。

13.4.2 设定

按下工作列上的“多状态设置”按钮后即会开启“多状态设置”元件属性对话框，正确设定各项属性后按下确认键，即可新增一个“多状态设置”元件。

一般属性设定


新增 多状态设置 元件
一般属性
安全
图片
标签
描述 :
写入地址
PLC 名称 :
Local HMI
设置...
地址 :
LW
0
16-bit Unsigned
当按钮松开才发出指令
通知
启用
开
关
写入前
写入后
PLC 名称 :
Local HMI
设置...
地址 :
LB
0
属性
模式 :
写入常数
设置常数 :
0
确定
取消
帮助

设定	描述
写入地址	点选“设置”后选择字符寄存器设备类型的”PLC 名称”，”地址”，”设备类型”，”系统寄存器”，”索引寄存器”来控制多状态设定元件。用户也可在“一般属性”页中设定地址。 **当按钮松开才发出指令** 使用此项设定表示在按压此按钮后，必须完全离开此区域才会执行元件定义的动作。如未使用此项设定，则只要一按压此钮，将立刻执行元件定义的动作。
通知	使用此项设定，则在使用手动操作模式时，在完成动作后可以连带设定此项目所指定寄存器的状态，使用“开”与“关”选择要设定的状态。 **“写入前” / “写入后”** 在写入动作前 / 后设定所指定寄存器的状态。
属性	**模式** 选择元件的动作模式。可以选择的模式请见以下范例 2。 **动态限制** 上下限可由指定寄存器设定，请见以下范例 1。

范例 1

上下限可由指定寄存器设定。当写入地址为 LW-n，则上/下限会根据以下的规则自动被设定为：

地址格式	16-bit	32-bit
动态限制地址	LW-n	LW-n
下限	LW-n	LW-n
上限	LW-n+1	LW-n+2

以下表为例，当“寄存器地址”为 LW-100 时，则上/下限的地址会自动被设定为：

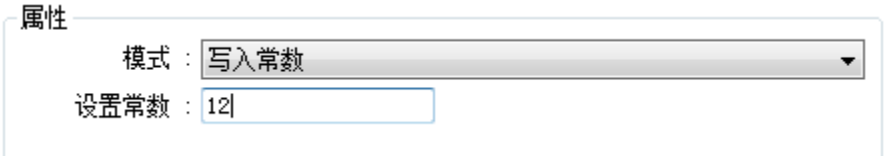
地址格式	16-bit	32-bit
动态限制地址	LW-100	LW-100
下限	LW-100	LW-100
上限	LW-101	LW-102

范例 2

可以选择模式如下：

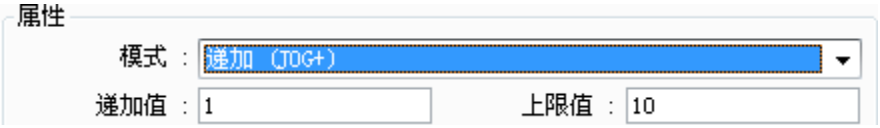
- 写入常数

设置常数功能。每按压一次元件，”设置常数”中的设定值将写至指定的寄存器中。常数的类型可为 16-bit BCD、32-bit BCD、…、32-bit float 等。以下图为例，当按压此按钮后，会将数值 12 写入指定的寄存器中。



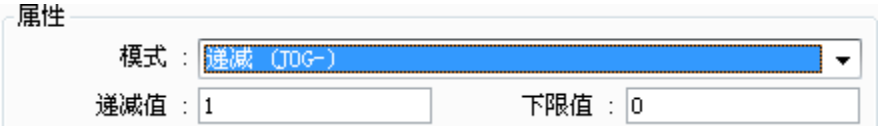
- 递加 (JOG+)

加值功能。每按压一次元件，所指定寄存器内的数据将加上“递加值”中设定的增量值，但增值的结果将不超过“上限值”中的设定值。以下图为例，每按压一次此按钮后，会将指定的寄存器中的数值+1，直至抵达上限值 10。



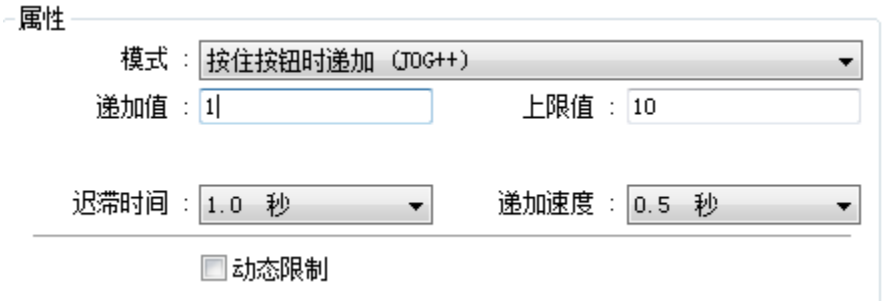
- 递减 (JOG-)

减值功能。每按压一次元件，所指定寄存器内的数据将减去“递减值”中设定的减量值，但是减值的结果不会低于“下限值”中的设定值。以下图为例，每按压一次此按钮后，会将指定的寄存器中的数值-1，直至抵达下限值 0。



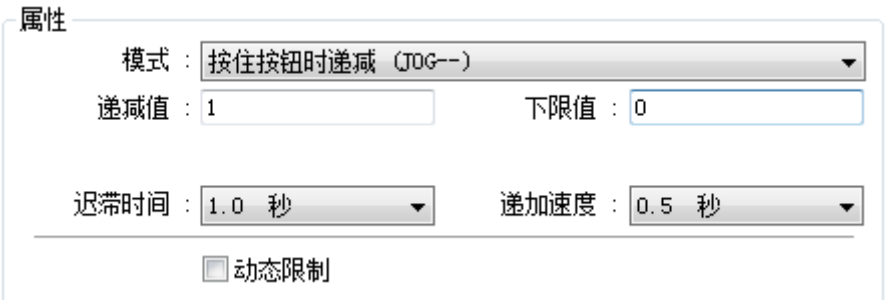
- 按住按钮时递加 (JOG++)

按住按钮时递加功能。若按压元件超过“迟滞时间”设定时间，则所指定寄存器内的数据将以“递加速度”所设定的速度，每次增加“递加值”中设定的增量值，但增量的结果将不超过“上限值”中的设定值。以下图为例，每按压一次此按钮后，会将指定的寄存器中的数值+1，若按住此按钮的时间超过 1.0 秒后，会以每 0.5 秒的速度持续+1 直到抵达上限值 10。



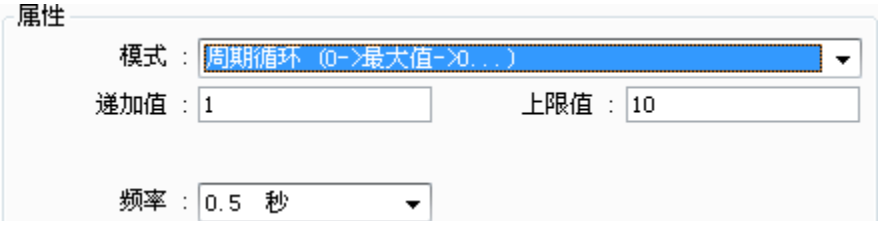
- 按住按钮时递减 (JOG--)

按住按钮时递减功能。若按压元件超过“迟滞时间”的设定时间，则所指定寄存器内的数据将以“递加速度”所设定的速度，每次减少“递减值”中设定的减量值，但减值的结果不会低于“下限值”中的设定值。以下图为例，每按压一次此按钮后，会将指定的寄存器中的数值-1，若按住此按钮的时间超过 1.0 秒后，会以每 0.5 秒的速度持续-1 到抵达下限值 0。



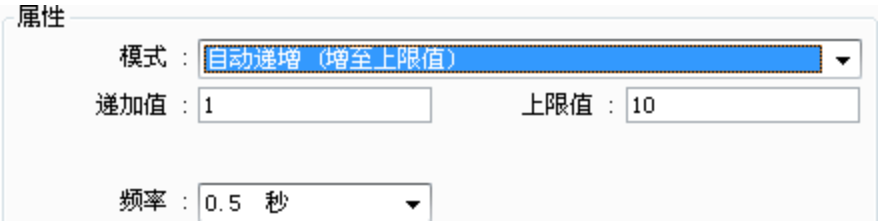
- 周期循环 (0->最大值->0)

周期性递加功能。"多状态设置"元件会使用"频率"设定的周期与"递加值"中设定的增量值，自动增量所指定寄存器内的数据，但增量的结果将不超过"上限值"中的设定值。以下图为例，系统会自动将指定的寄存器中的数值以每 0.5 秒的频率+1，直到抵达上限值 10，接着数值会返回 0 再重新持续+1。



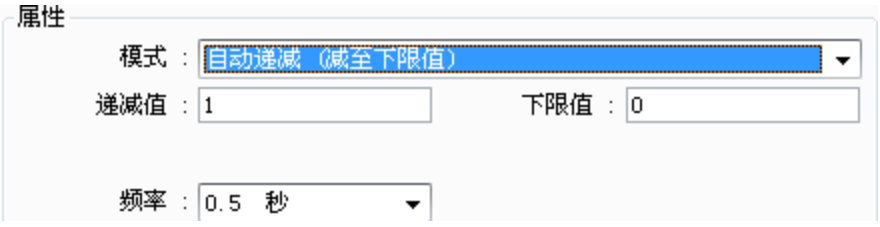
- 自动递增 (增至上限值)

周期性递增功能。"多状态设置"元件会使用"频率"设定的周期，自动将所指定寄存器内的数据加上"递加值"中设定的增量值，当结果等于"上限值"时自动停止。以下图为例，系统会自动将指定的寄存器中的数值以每 0.5 秒的频率+1，直到抵达上限值 10 后停止。



- 自动递减 (减至下限值)

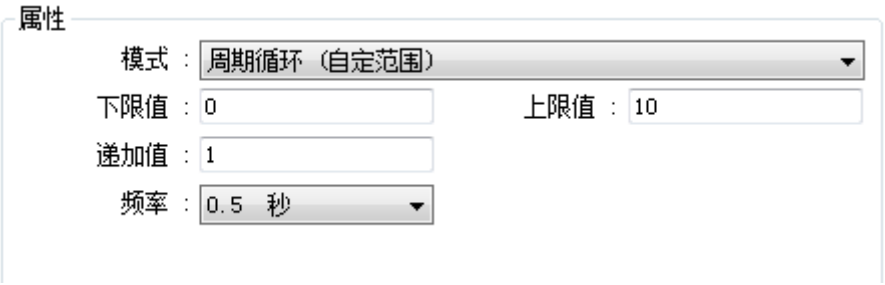
周期性递减功能。"多状态设置"元件会使用"频率"设定的周期，自动将所指定寄存器内的数据减去"递减值"中设定的减量值，当结果等于"下限值"时自动停止。以下图为例，系统会自动将指定的寄存器中的数值以每 0.5 秒的频率-1，直到抵达下限值 10 后停止。



- 周期循环 (自定范围)

周期性循环功能。"多状态设置"元件会使用"频率"设定的周期，每次将所指定寄存器内的数据加上"递加值"中的设定值，直到寄存器内的数据等于"上限值"；接着使用相同的周期，将寄存器内的数据减去"递加值"中的设定值，直到寄存器内的数据等于"下限值"。以下图为例，系统会自动将指定的寄存器中的数值

以每 0.5 秒的频率+1，直到抵达上限值 10，接着再以相同的频率-1 直到等于下限值 0，如此周而复始的执行不停止。

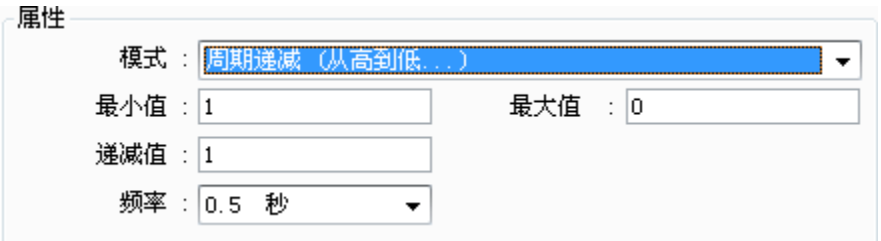


- 周期递加（从低到高）

步进功能。”多状态设置”元件会使用“频率”设定的周期，每次将所指定寄存器内的数据加上“递加值”中的设定值，直到寄存器内的数据等于“最大值”，接着会将寄存器内的数据复归为“最小值”，并重复先前的动作，让数据一直保持动态变化。以下图为例，系统会自动将指定的寄存器中的数值以每 0.5 秒的频率+1，直到抵达上限值 10，接着再返回下限值 0 重新递增，如此周而复始的执行不停止。

- 周期递减（从高到低）

步退功能。”多状态设置”元件会使用“频率”设定的周期，每次将所指定寄存器内的数据减去“递减值”中的设定值，直到寄存器内的数据等于“最小值”，接着会将寄存器内的数据复归为“最大值”，并重复先前的动作，让数据一直保持动态变化。以下图为例，系统会自动将指定的寄存器中的数值以每 0.5 秒的频率-1，直到抵达下限值 0，接着再返回上限值 10 重新递减，如此周而复始的执行不停止。



- 窗口打开时设定 / 窗口关闭时设定

开启 / 关闭元件所在位置的窗口时，会将“设定常数”中的设定值自动写至指定的寄存器中。若”设定常数”设为 5，当该页的窗口被开启 / 关闭时，系统会自动将数值 5 写入寄存器中。

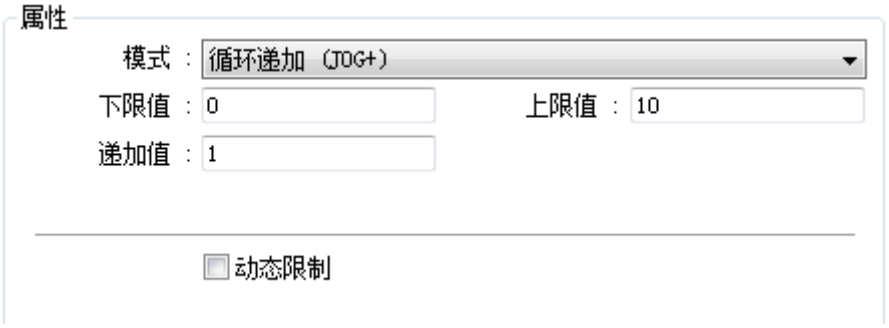
- 当背光灯开时设定 /当背光灯关时设定 (不支持 cMT-SVR 系列)

当背光灯原处在关闭 / 开启状态，若恢复为相反状态时，会将“设定常数”中的设定值自动写至指定的寄存器中。若“设定常数”设为 5，当该页的背光灯状态改变时，系统会自动将数值 5 写入寄存器中。

- 循环递加 (Cyclic JOG+)

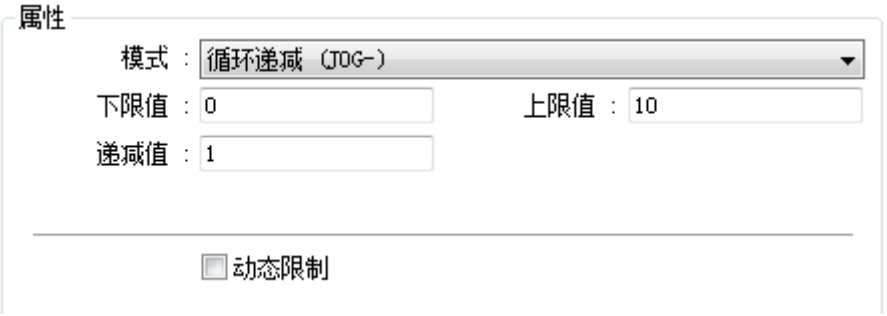
加值功能。每按压一次元件，所指定寄存器内的数据将加上“递加值”中设定的增量值。当增量值达到上限时，会复归回下限再重新递增。以下图为例，每按压一次此元件后，会将指定的寄存器中的数值+1，当抵

达上限值 10 后会自动复归回 0 再递增执行。



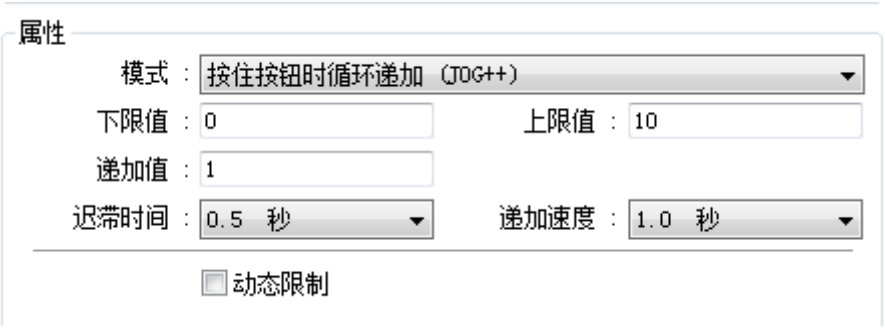
- 循环递减 (Cyclic JOG-)

减值功能。每按压一次元件，所指定寄存器内的数据将减去“递减值”中设定的减量值。当减量值达到下限时，会复归回上限再重新递减。以下图为例，每按压一次此元件后，会将指定的寄存器中的数值-1，当抵达下限值 0 后会自动复归回 10 再递减执行。



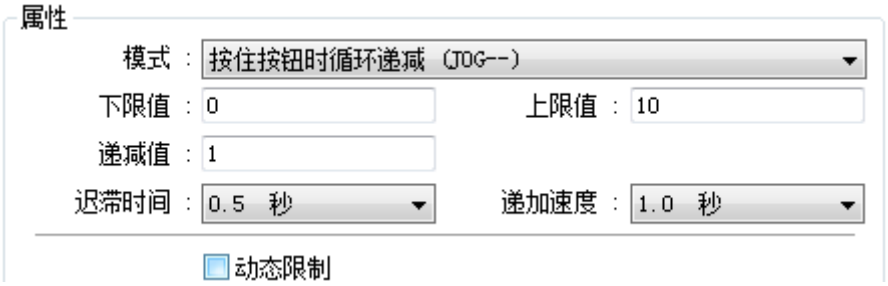
- 按住按钮时循环递加 (JOG++)

持续递加功能。当按住按钮的时间超过 “迟滞时间” 时，此元件会根据 “递加速度” 的设定将指定寄存器的数据持续的递加至上限值，之后会复归回下限值。以下图为例，每按压一次此按钮后，会将指定的寄存器中的数值+1，若按住此按钮的时间超过 0.5 秒后，会以每 0.1 秒的速度持续+1 直到抵达上限值 10，接着会复归回 0 再递增执行。



- 按住按钮时循环递减 (JOG- -)

持续递减功能。当按住按钮的时间超过 “迟滞时间” 时，此元件会根据 “递减速度” 的设定将指定寄存器的数据递减至下限值，之后会复归回上限值。以下图为例，每按压一次此按钮后，会将指定的寄存器中的数值-1，若按住此按钮的时间超过 0.5 秒后，会以每 0.1 秒的速度持续-1 直到抵达下限值 0，接着会复归回 10 再递增执行。


属性
模式 :
按住按钮时循环递减 (JOG--)
下限值 :
0
上限值 :
10
递减值 :
1
迟滞时间 :
0.5 秒
递加速度 :
1.0 秒
动态限制

13.5 功能键

13.5.1 概要

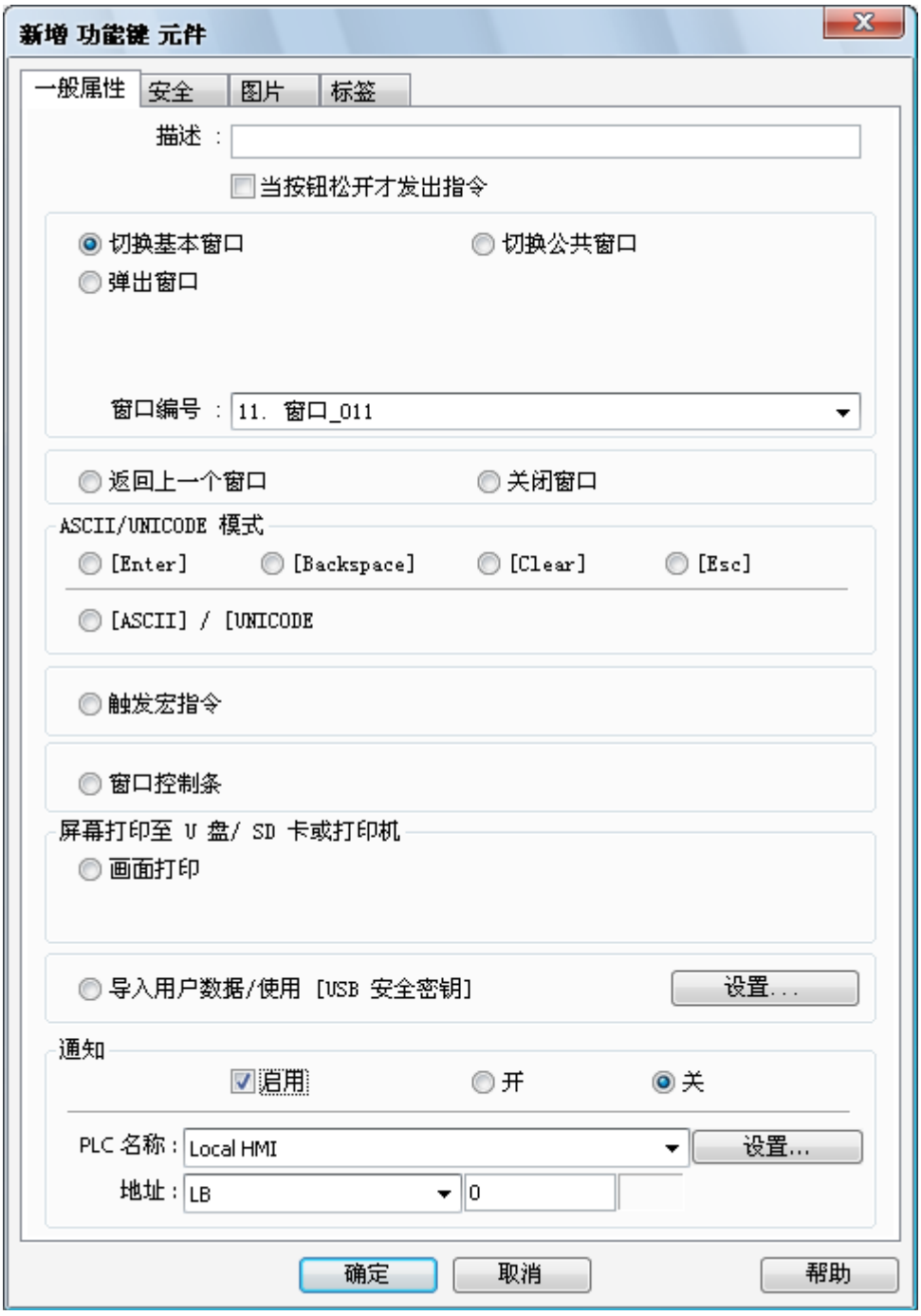
“功能键”元件提供窗口切换、键盘制作、宏执行及画面打印等功能，同时也可用于设定 USB 安全密钥。

13.5.2 设定

按下工作列上的“功能键”按钮后即会开启“功能键”元件属性对话框，正确设定各项属性后按下确认键，即可新增一个“功能键”元件。

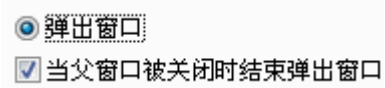
一般属性设定

cMT-SVR 系列



设定	描述
松开按键时触发该指令	使用此选项表示必须在释放按压元件的动作后，选择的动作才会被执行。若未选择，则在碰触元件后，将立刻执行选择的动作。
窗口切换	**切换基本窗口：**切换基本窗口。 **切换公共窗口：**切换公用窗口。

弹出窗口：呼叫其它窗口。此时呼叫出的窗口必定在基本窗口的上面。使用此功能可以选择是否使用“当父窗口被关闭时结束弹出窗口”，参考下图。选择此属性则呼叫出的窗口会在发生换页动作时自动消失，否则用户必须自行在被呼叫出的窗口上设计“关闭窗口”功能键来关闭此窗口。



返回上一个窗口：返回前一页基本窗口。例如当由 ”窗口 10” 切换到 ”窗口 20” 时，使用此功能可以再返回”窗口 10”。此功能只对基本窗口有效。

关闭窗口：关闭在基本窗口上被呼叫出的窗口，包括讯息窗口。

ASCII/ UNICODE 模式

用来作为键盘的输入讯号，主要用在“数值输入”与“字符输入”元件需要使用键盘来输入数字或文字的场合。

Enter: 与键盘的输入（Enter）动作相同。

Backspace: 与键盘的后退删除（Backspace）动作相同。

Clear: 清除寄存器中已输入的数据。

Esc: 与使用“关闭窗口”功能相同，可用来关闭弹跳出的键盘窗口。

Delete: 与键盘的删除（Delete）动作相同，可将光标右方的一个字符删除。

Left: 与键盘的←动作相同，可将光标向左移动一个字符。

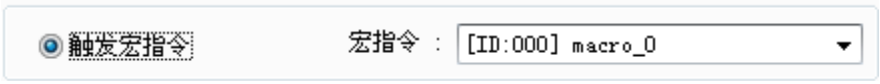
Right: 与键盘的→动作相同，可将光标向右移动一个字符。

ASCII/UNICODE: 设定键盘的输入字符。

触发宏指令

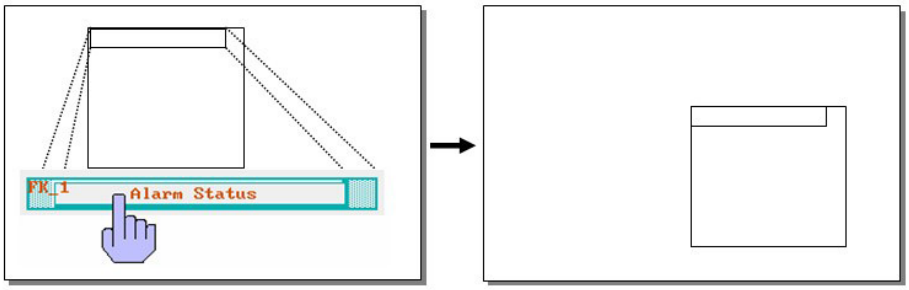
选择此项功能，将执行指定的宏命令，选择此项功能前需先建立宏命令。

如何建立宏命令请参考《18 宏指令说明》



窗口控制条

当弹出的窗口**无**窗口控制条时，若需要移动窗口，则先点一下此元件，在移动的目的地再点一下，则窗口就会被移动到指定的位置。



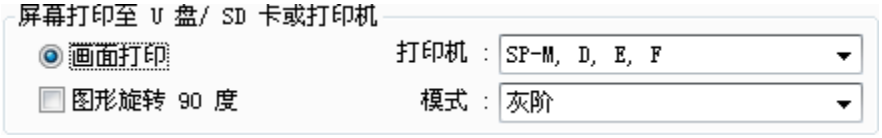
首先点击窗口控制条

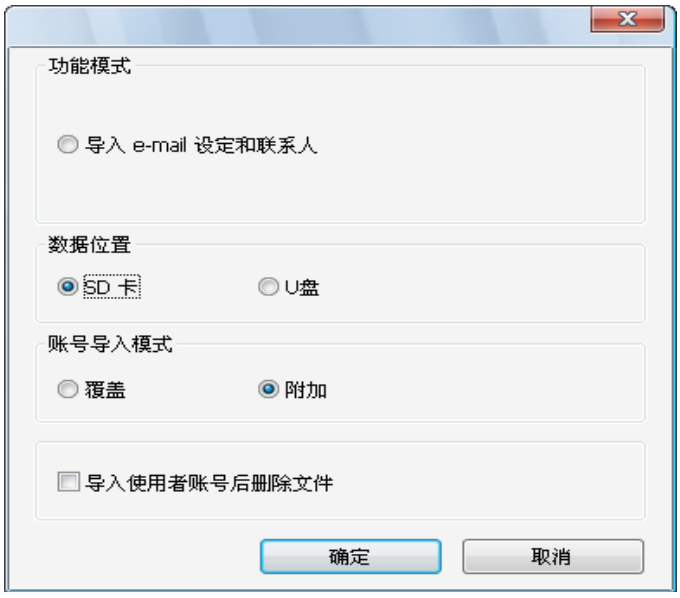
接着在要移动的位置触控一下，该窗口就会移动到该位置了

画面打印至 U 盘, SD 卡或打印机

此项功能用来打印当前的画面。要选择此项功能前需先在“系统参数设置”»“HMI 属性”中选择所使用的打印机类型。使用单色打印机时，勾选“灰阶效果”可以提升画面的辨识度，但也会影响文字的显示效果，因此如果是强调文

字的打印效果，并不需要使用灰阶功能。



画面打印 (cMT-SVR 系列)	输出目前的画面至 iPad 的 Photos。
Acknowledge all events (alarms) (cMT-SVR 系列)	确认所有事件。
导入用户数据/使用“USB 安全密钥”	用来导入进阶安全的用户账号或 e-mail 的联络人。也可设定为使用 USB 密钥登入。  **数据位置：**提供从 U 盘、SD 卡读取两个选项。 **账号导入方式：**选择“覆盖”，HMI 内将只保存此次导入的账号数据，若是选择“附加”，HMI 内账号数据将保留，并加入此次导入的新账号数据。 **导入用户账号后删除文件：**将 USB 内的用户账号导入后即删除来源数据，可确保数据不泄漏。
通知	使用此项设定，则在完成动作后可以连带设定此项目所指定寄存器的状态，使用“开”与“关”选择要设定的状态。

Note

- 当导入 e-mail 的联络人时，仅会以“覆盖”的方式导入，因此已存在的联络人皆会先被删除后才导入新的联络人。

详细信息请参考《6 窗口》、《12 键盘的设计与使用》、《36 管理员工具》。

13.6 位状态切换开关

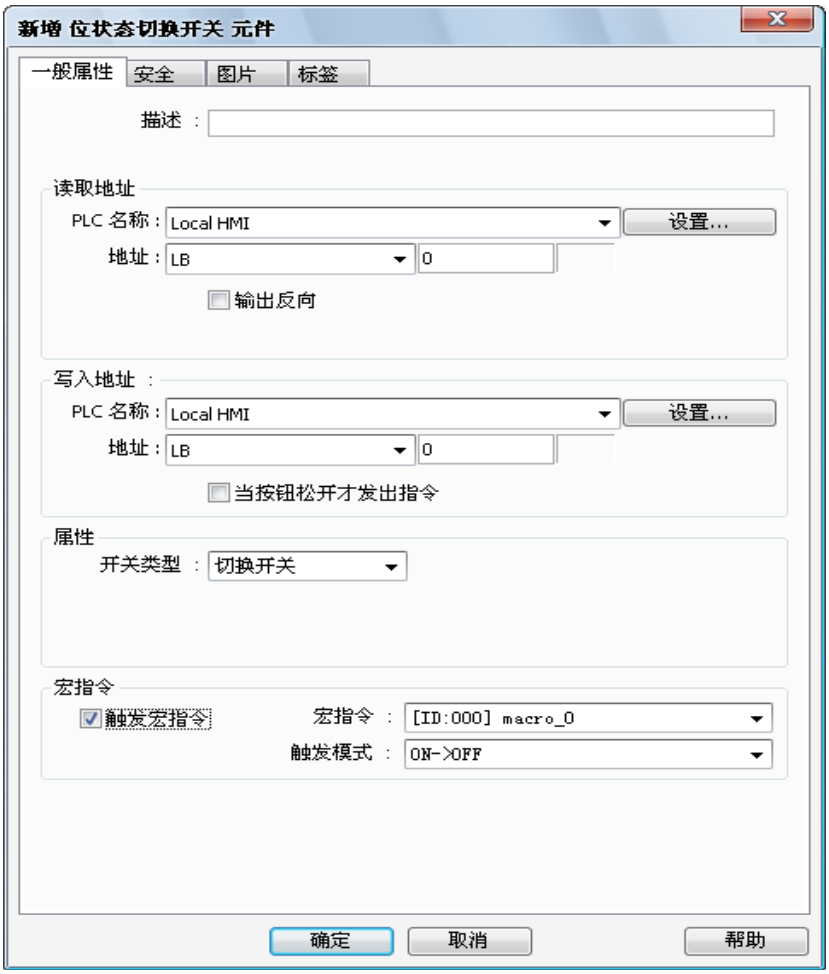
13.6.1 概要

“位状态切换开关”为”位状态指示灯”元件与”位状态设置”元件的组合。此元件除了可以用来显示寄存器的状态外，也可以利用这个元件在窗口上定义一个碰触区域，按压此区域可以设定所指定寄存器的状态为 ON 或 OFF。

13.6.2 设定

按下工作列上的“位状态切换开关”按钮后即会开启“位状态切换开元件属性对话框”，正确设定各项属性后按下确认键，即可新增一个“位状态切换开关”元件。

一般属性设定



<table>
<tr><th>设定</th><th>描述</th></tr>
<tr><td>读取地址</td><td>点选“设置”后选择位寄存器设备类型的“PLC 名称”，”地址”，”设备类型”，”系统寄存器”，”索引寄存器”来控制显示位状态的切换开关状态元件。用户也可在“一般属性”页中设定地址。
输出反向
可以将读取的状态作反向显示，例如位的状态实际上为 OFF，但勾选了“输出反向”后会显示为 ON。</td></tr>
<tr><td>写入地址</td><td>点选“设置”后选择位寄存器设备类型的“PLC 名称”，”地址”，”设备类型”，”系统寄存器”，”索引寄存器”来控制位状态切换开关元件。用户也可在“一般属性”页中设定地址。此寄存器可以与“读取地址”所指定的寄存器相同亦或不同。
当按钮松开才发出指令
使用此设定表示在按下元件后，必须完全松开按压动作，元件定义的操作模式才会被执行。如未使用此项设定，只要一碰触此区域，将立刻执行元件的动作。若选择使用复归型模式，将不支持此项功能。</td></tr>
<tr><td>属性</td><td>
<table>
<tr><th>开关类</th><th>描述</th></tr>
<tr><td>设为 ON</td><td>按压此元件后，所指定寄存器的状态将被设定为 ON。</td></tr>
<tr><td>设为 OFF</td><td>按压此元件后，所指定寄存器的状态将被设定为 OFF。</td></tr>
<tr><td>切换开关</td><td>按压此元件后，所指定寄存器的状态将被反相。当状态为 ON 时，会被显示为 OFF。当□态为 OFF 时，则会被显示为 ON。</td></tr>
<tr><td>复归型</td><td>按压此元件后，所指定寄存器的状态将先被设定为 O□ 但手放开后，状态将被设定为 OFF。</td></tr>
</table>
</td></tr>
<tr><td>宏指令</td><td>“位状态切换开关”元件可以搭配执行宏指令。选择此项功能前需先建立宏指令。

如何建立宏命令请参考《18 宏指令说明》</td></tr>
</table>

13.7 多状态切换开关

13.7.1 概要

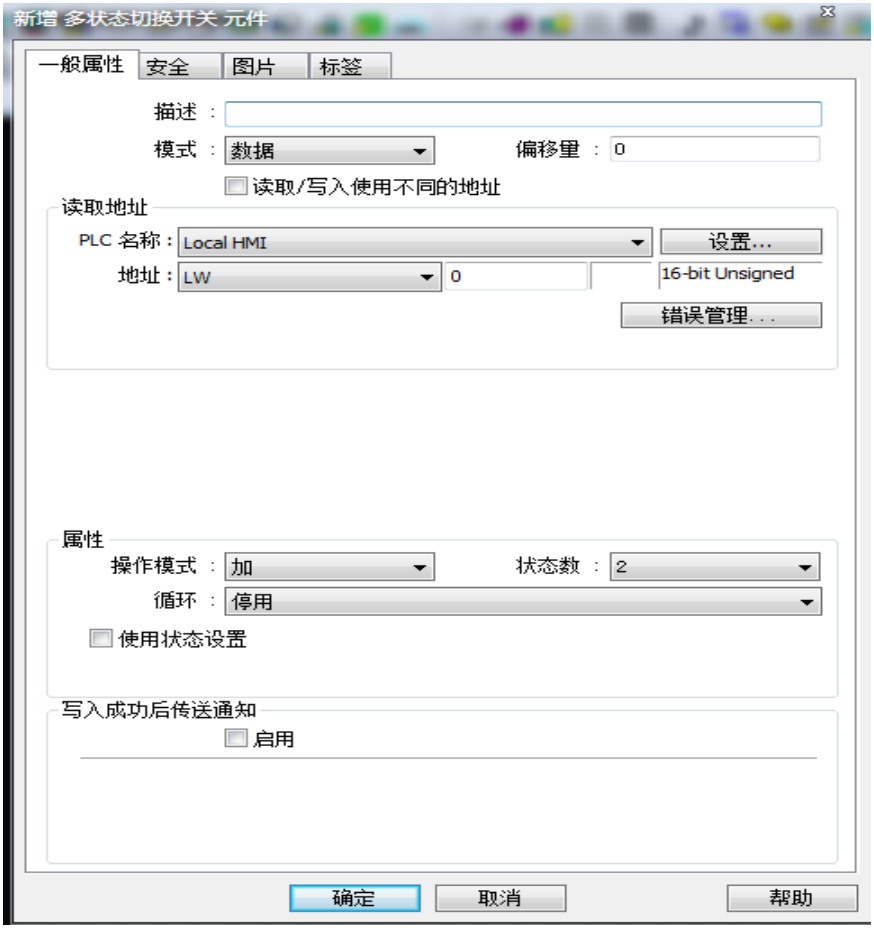
“多状态切换开关”元件为“多状态指示灯”元件与“多状态设置”元件的组合。此元件除了可以利用寄存器内的数据显示不同的状态外，也可以利用这个元件在窗口上定义一个碰触区域，按压此区域可以设定所指定寄存器内的数据。

13.7.2 设定

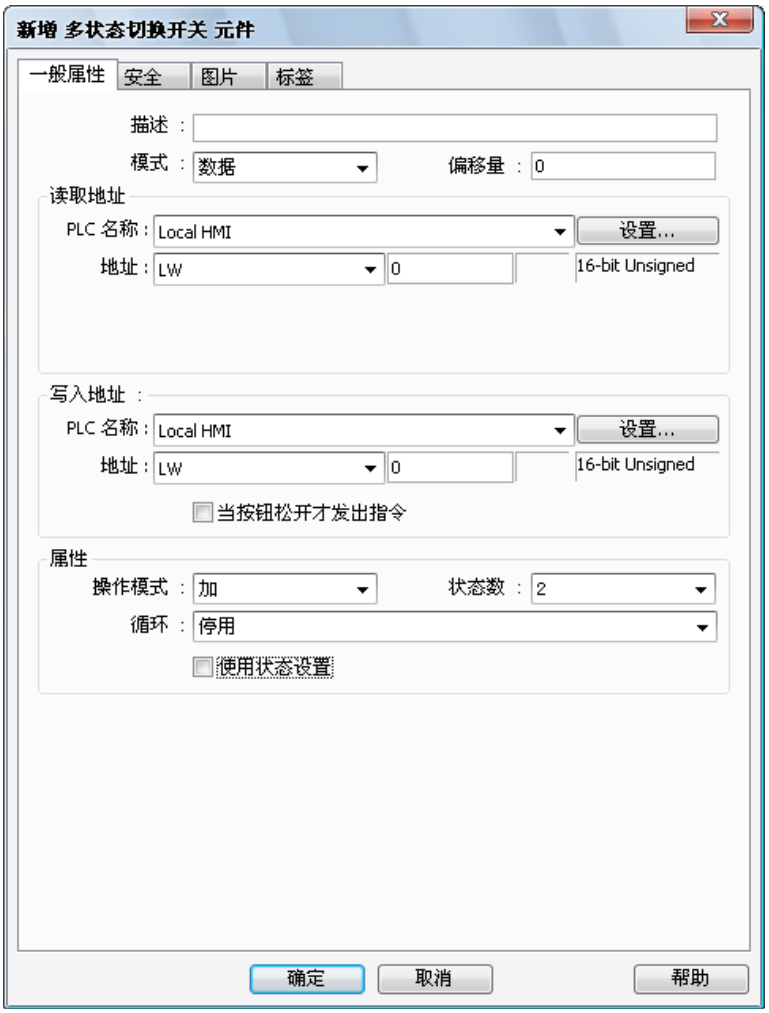
按下工作列上的“多状态切换开关”按钮后即会开启“多状态切换开关”元件属性对话框，正确设定各项属性后按下确认键，即可新增一个“多状态切换开关”元件。

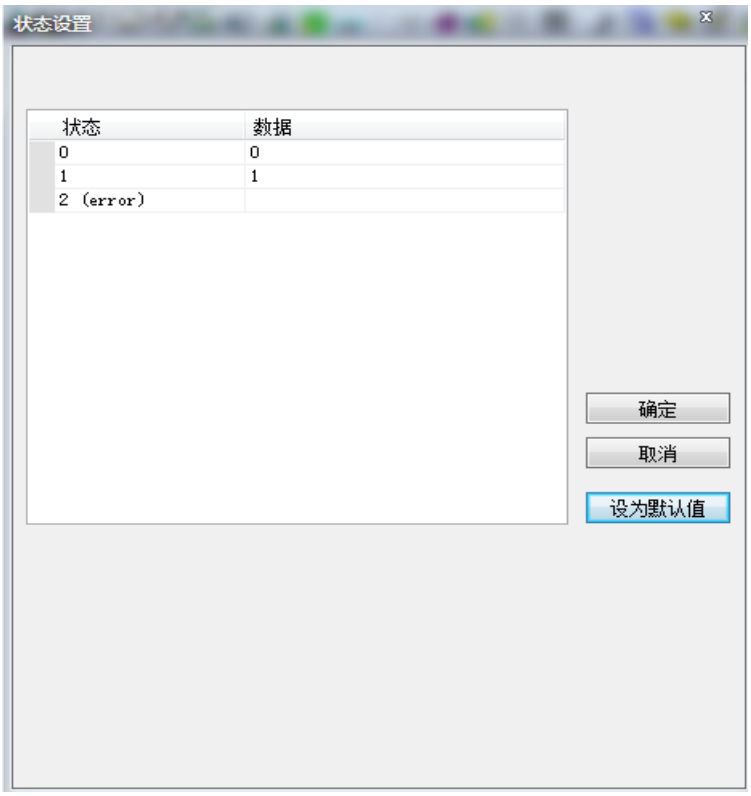
一般属性设定

cMT-SVR 系列



eMT、iE、cMT-HD 系列



设定	描述
模式 / 偏移量	提供三种不同的数据显示模式： 数据、LSB。 详细信息请参考《13.2 多状态指示灯》。
读取地址	点选“设置”后选择字符寄存器设备类型的“PLC 名称”,”地址”,”设备类型”,”系统寄存器”, ”索引寄存器”来控制显示多状态切换开关状态元件。用户也可在“一般属性”页中设定地址。
写入地址	点选“设置”后选择字符寄存器设备类型的“PLC 名称”,”地址”,”设备类型”,”系统寄存器”, ”索引寄存器”来控制多状态切换开关元件。用户也可在“一般属性”页中设定地址。此寄存器可以与“读取地址”所指定的寄存器相同亦或不同。 **当按钮松开才发出指令** 使用此项设定表示在按压此按钮后，必须完全离开此区域才会执行元件定义的动作。如未使用此项设定，则只要一按压此钮，将立刻执行元件定义的动作。
属性	**模式** 选择元件的操作方式。可以选择的模式请见以下范例 1。 **使用状态设定** 用户可修改状态对应的数值，亦可使用当有不合法的数值输入时的动作状态和通知指定位切换状态。 

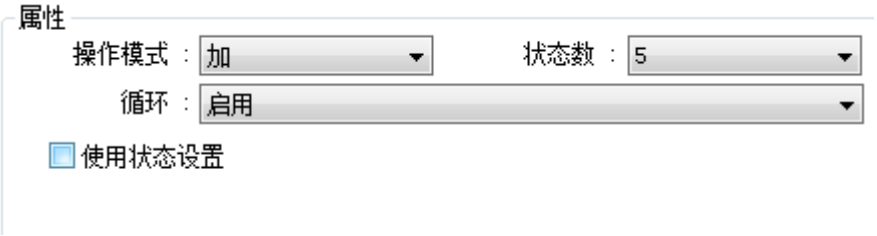
	保持目前状态 若输入超出范围的数值，多状态切换开关会保持目前状态。 **跳至错误状态** 若输入超出范围的数值，多状态切换开关会跳到错误状态。 **错误通知** 当输入无效的数值时，可以自动设定所指定地址的状态。
写入成功后传送通知	当写入 PLC 的动作成功后，将指定位寄存器的状态设为开 / 关。
Error handling (cMT-SVR 系列)	当有不合法的数值输入时的动作状态和通知指定位切换状态。用途与“使用状态设置”雷同，但可不必设置各状态对应的数据。

范例 1

可以选择模式如下：

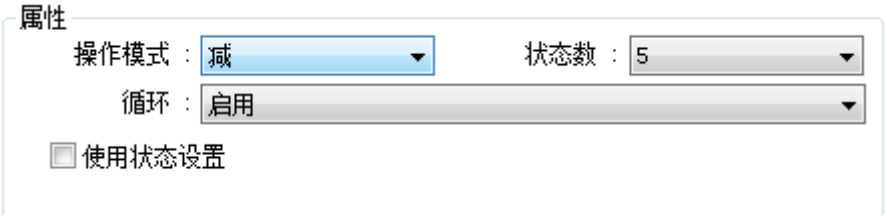
- 加 (JOG+)

递加功能。每按压一次元件，所指定寄存器内的数据+1，但增值的结果将不超过“状态数”。若 “启用” 循环，则抵达最大状态后会复归回最低状态 0。以下图为例，若操作模式选择 “加”，状态数为 5 且 “启用” 循环，则每按压一次此元件，状态会从状态 0 会往上+1 直至状态 4 (“状态数” -1)，然后复归回状态 0 重新递加。



- 减 (JOG-)

递减功能。每按压一次元件，所指定寄存器内的数据-1 直至 0。若 “启用” 循环，则抵达最大状态后会复归回最高状态。以下图为例，若操作模式选择 “减”，状态数为 5 且 “启用” 循环，则每按压一次此元件，会往下-1 直至状态 0，然后复归回最高状态 4 (“状态数” -1) 重新递减。



13.8 滑动开关

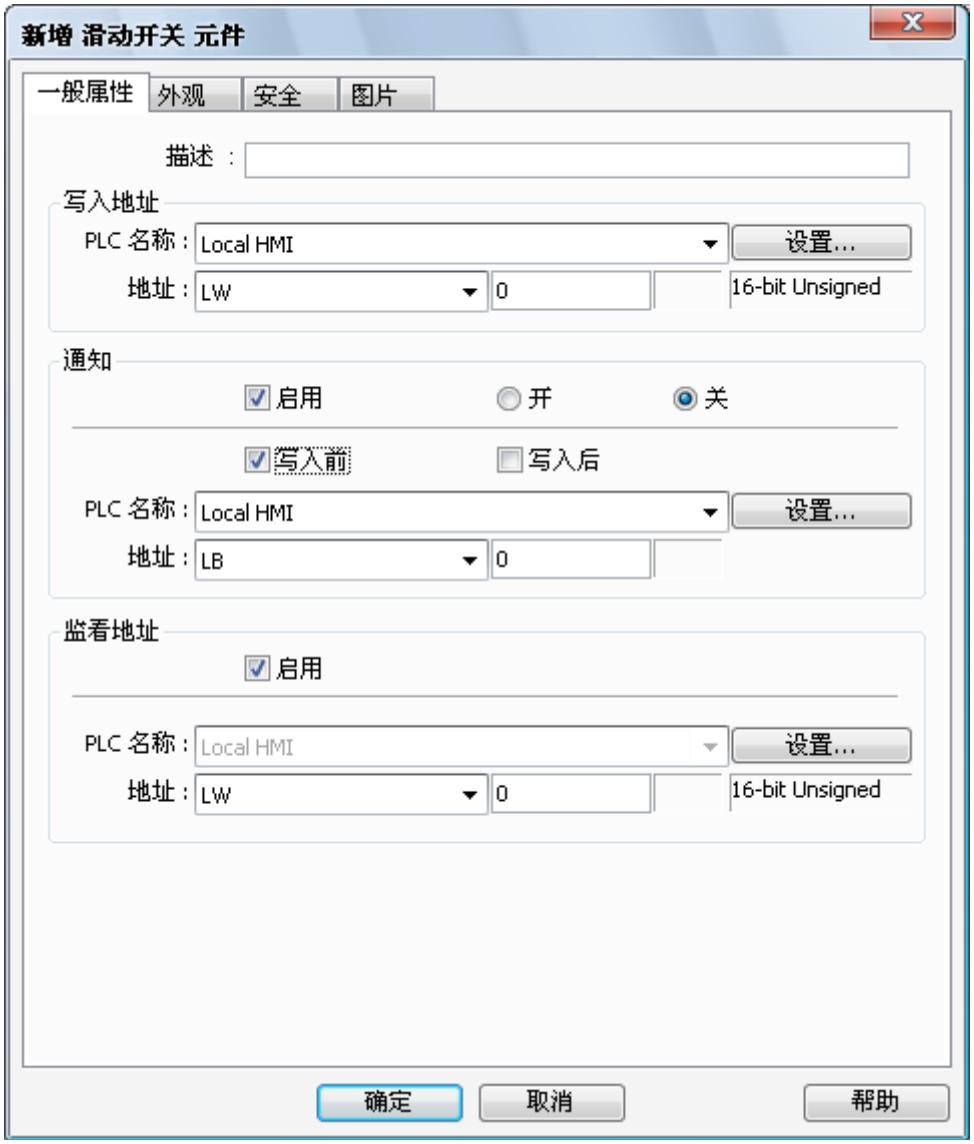
13.8.1 概要

“滑动开关” 元件是用来建立一个滑块区域显示数值或借由拖曳滑轨改变指定寄存器内的数值。

13.8.2 设定

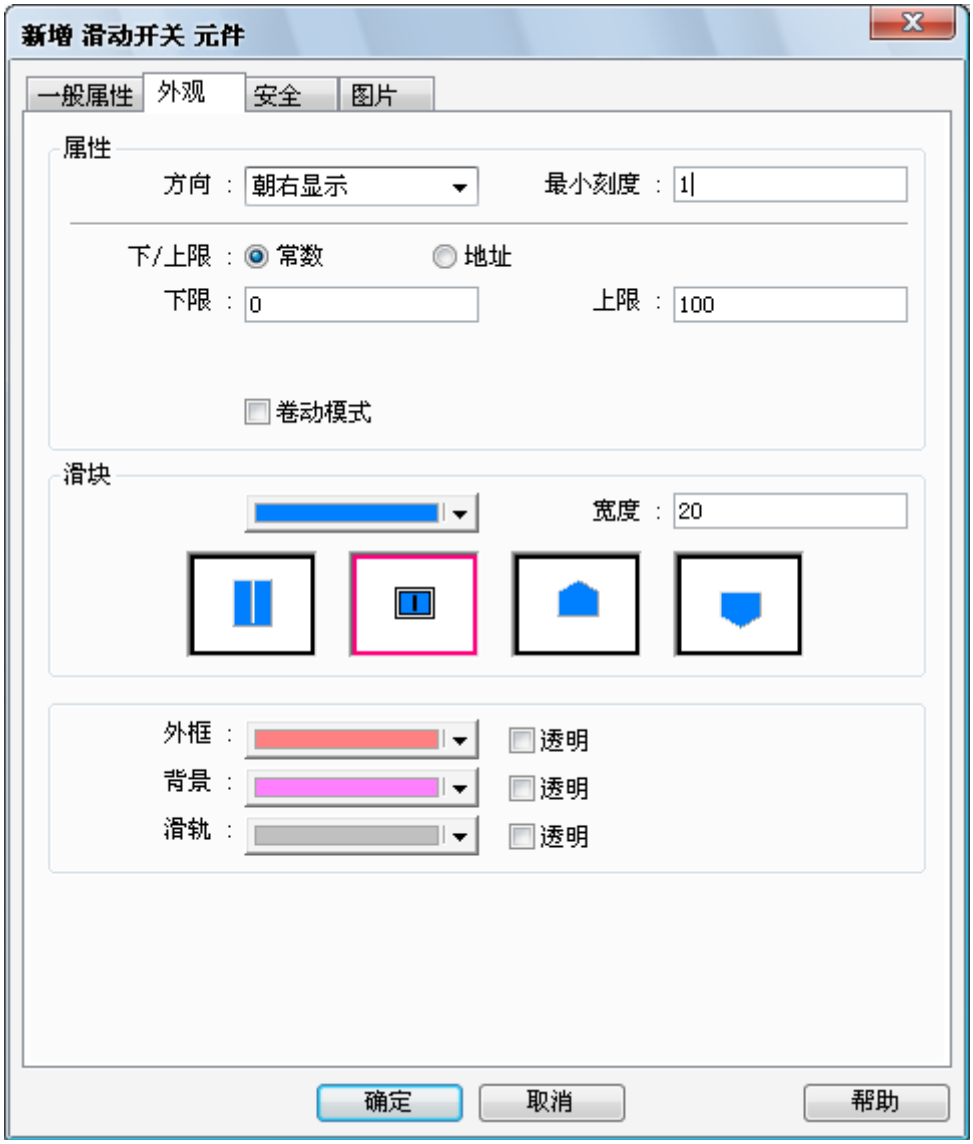
按下工作列上的“滑动开关”按钮后即会开启“滑动开关元件属性对话框”，正确设定各项属性后按下确认键，即可新增一个“滑动开关”元件。

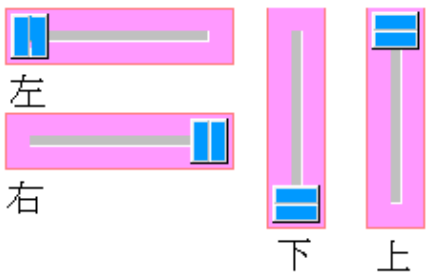
一般属性设定

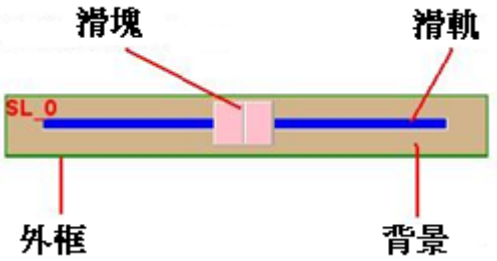


设定	描述
写入地址	点选“设置”后选择字符寄存器设备类型的“PLC 名称”,”地址”,”设备类型”,”系统寄存器”,”索引寄存器”来控制滑动开关元件。用户也可在“一般属性”页中设定地址。
通知	使用此项设定，则在使用手动操作模式时，在完成动作后可以连带设定此项目所指定寄存器的状态，使用“开”与“关”选择要设定的状态。 点选**“设置”**后选择位寄存器设备类型的“PLC 名称”，”地址”，”设备类型”，”系统寄存器”，”索引寄存器”来控制通知位项目。用户也可在“一般属性”页中设定地址。 **“写入前”/“写入后”** 在写入动作前 / 后设定所指定寄存器的状态。
监看地址	在滑块被拖曳时，可以实时显示当前写入地址的设定值。

外观设定



设定	描述
属性	**方向** 滑动开关元件可以四个方向来显示 (朝右显示，朝上显示，朝左显示，朝下显示)。  **最小刻度**

	依照所填入之最小刻度值来显示。例如：设“最小刻度”为 10，数值显示为每一次都是依据 10 的刻度来跳动。 **常数** 可直接设定字符寄存器的上下限常数值。例如：设“下限”为 5 和“上限”为 100，则设定的数值范围为 5 ~ 100。 **地址** 上下限可由指定寄存器设定，请见以下范例 1。 **卷动模式** 不同于 “最小刻度” 拖拉滑动开关改变数值，只要轻触一下 “滑动开关” 物件时，数值会被递增/递减，增减的多寡会根据 “卷动值” 的设定。
滑块	**使用图片 (只支持 cMT-SVR 系列)** 若勾选，可从图片库挑选一图片作为滑块的显示图标。 系统共有四种预设滑块样式可供选择，也可调整滑块宽度。 **颜色** 用来选择滑块元件的外框，背景和滑轨颜色。 

范例 1

上下限可由指定寄存器设定。当写入地址为 LW-n，则上/下限会根据以下的规则自动被设定为：

地址格式	16-bit	32-bit
地址	LW-n	LW-n
下限	LW-n	LW-n
上限	LW-n+1	LW-n+2

以下表为例，当“寄存器地址”为 LW-100 时，则上/下限的地址会自动被设定为：

地址格式	16-bit	32-bit
地址	LW-100	LW-100
下限	LW-100	LW-100
上限	LW-101	LW-102

13.9 数值输入与数值显示元件

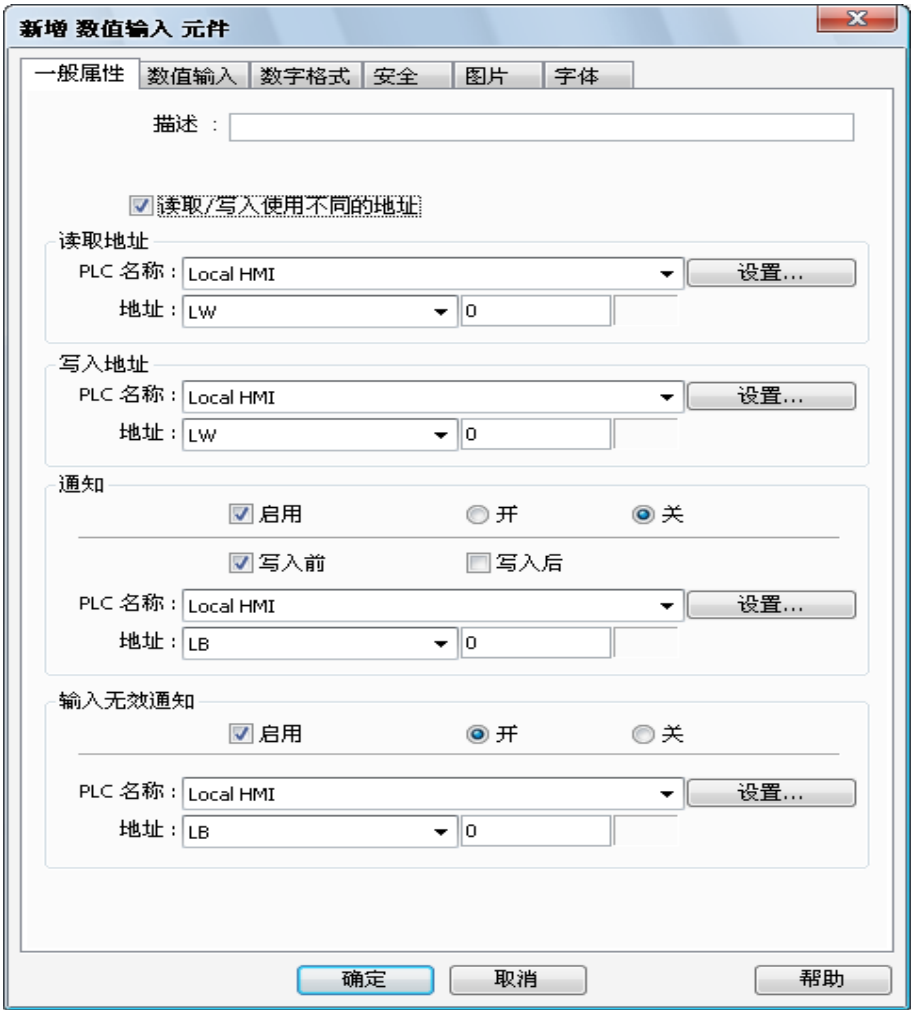
13.9.1 概要

“数值输入”与“数值显示”元件皆可以用来显示所指定字符寄存器内的数值，其中“数值输入”元件可用键盘来输入数值，更改寄存器内的数据。

13.9.2 设定

按下工作列上的“数值输入/数值显示”按钮后即会开启“数值输入/数值显示元件属性对话框”，正确设定各项属性后按下确认键，即可新增一个“数值输入/数值显示”元件。“数值输入”元件比 “数值显示”元件多了写入的功能。

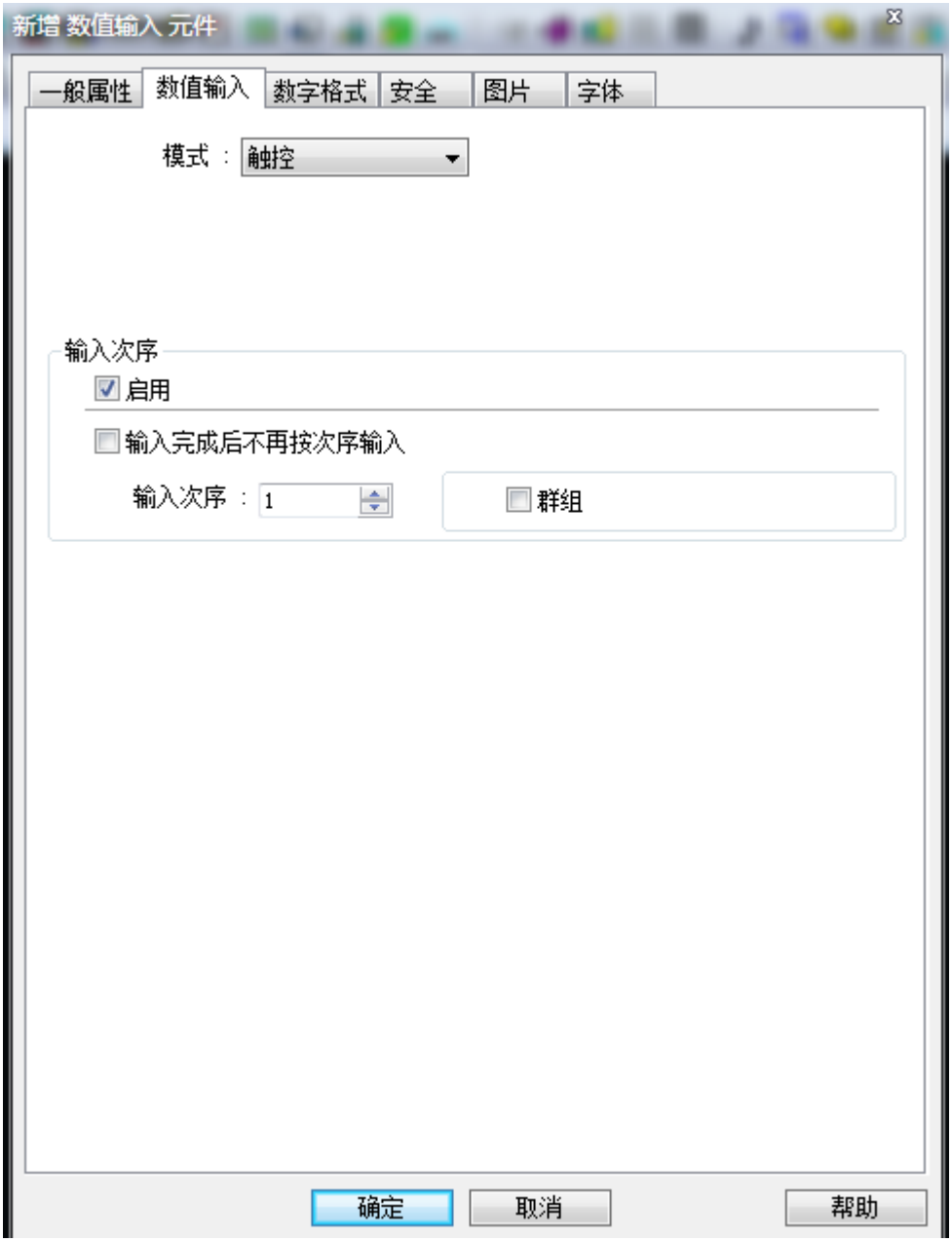
一般属性设定

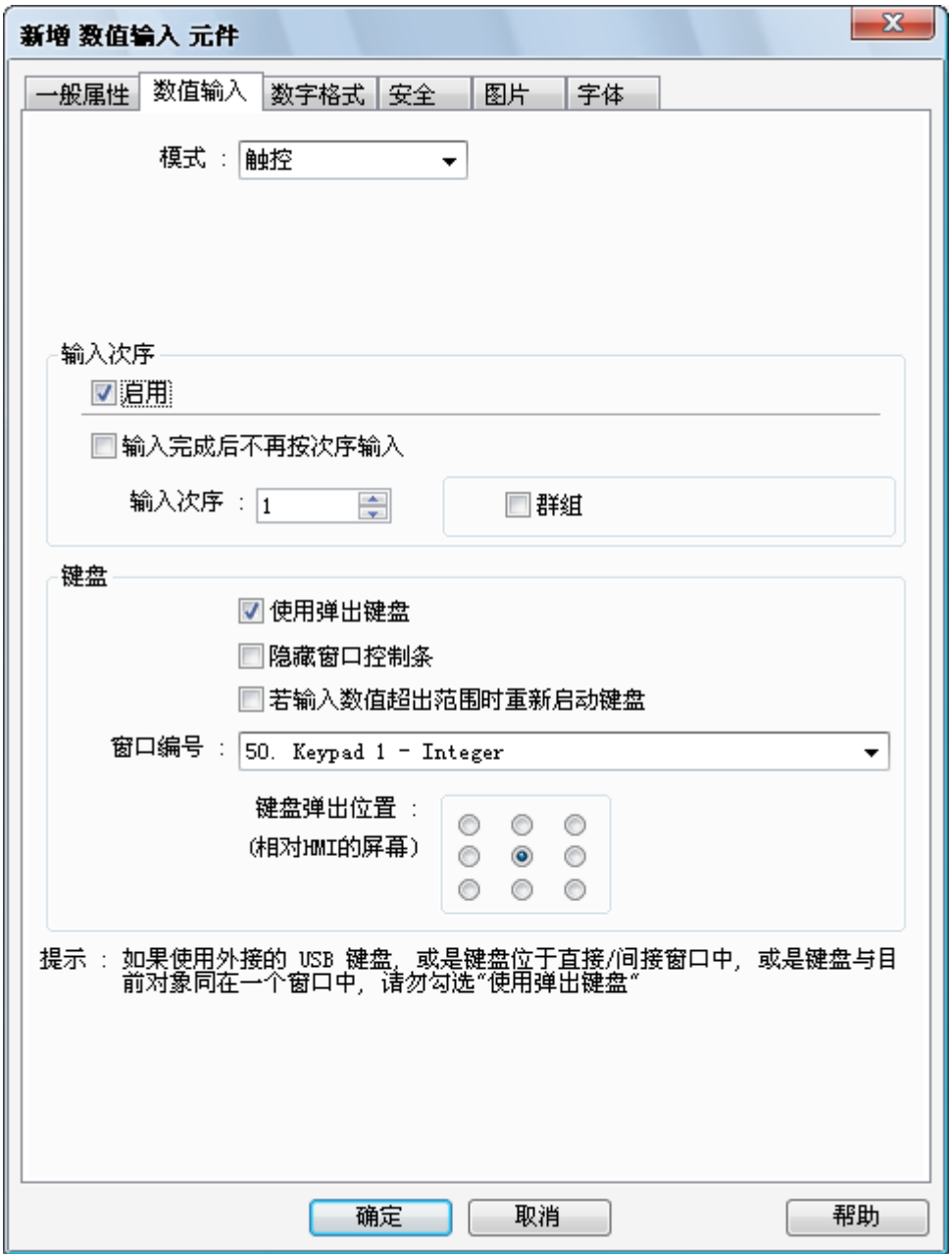


设定	描述
读取/写入使用不同地址	用户可以分开设定数据的读取地址与写入地址。
读取地址	点选“设置”后选择字符寄存器设备类型的“PLC 名称”,”地址”,”设备类型”,”系统寄存器”,”索引寄存器”来显示数值。用户亦可选用“地址标签库”里设定好的地址标签。
写入地址	选择字符相关的“PLC 名称”,”设备类型”,”地址”作为数值写入目标。
通知	使用此项设定，则在完成动作之前 / 之后可以连带设定此项目所指定寄存器的状态，使用“开”与”关”选择要设定的状态。 **写入前/写入后** 在写入动作前 / 后设定所指定寄存器的状态。
输入无效通知	当输入无效的数值时，设定指定寄存器的状态“开 / 关”。

数值输入设定

cMT-SVR 系列





设定	描述
模式	**触控** 用户通过触控元件来启动输入程序。 **位控制** 用户通过指定位寄存器的开或关来启动及结束输入程序。
允许输入位地址	指定控制输入启动及结束的位寄存器地址。输入顺序必须遵照“输入次序”的设定，输入时须搭配外接 USB 键盘，不可使用屏幕触控键盘。

	cMT-SVR 则需使用 iPad 的内建键盘。
输入次序	设定输入次序及输入次序群组达成多个输入元件连续输入。 使用准则： ● 输入次序范围: 1 ~ 511。群组范围: 1 ~ 15。 ● 若无勾选“群组”时，输入次序群组为 0。 ● 系统只寻找同一个输入次序群组中的输入元件。 ● 越小的输入次序数值代表输入顺序排在越前面，反之则越后面。 ● 若多个输入元件有同样的输入次序群组及输入次序，则较下层的输入元件将优先输入。
键盘 (适用于 eMT、iE、cMT-HD 系列)	**使用弹出键盘** 勾选：指定键盘窗口及弹出位置。当启动输入时，系统将在指定位置弹出键盘窗口，并于结束输入时关闭。 不勾选：启动输入时系统将不会弹出键盘窗口，用户必须以下列方法进行输入动作： ● 自行在窗口中设计键盘。 ● 使用外接键盘。 **隐藏窗口控制条** 数值 / 字符输入键盘可选择不使用窗口控制条。 **若输入数值超出范围时重新启动键盘** 当使用输入元件时，若输入的数值超出设定范围时系统将自动重新启动键盘。

Note

cMT-SVR 采用 iPad 的内建键盘输入。

若欲于当前窗口内嵌键盘，请参考《第十二章:键盘设计与应用》。

范例 1

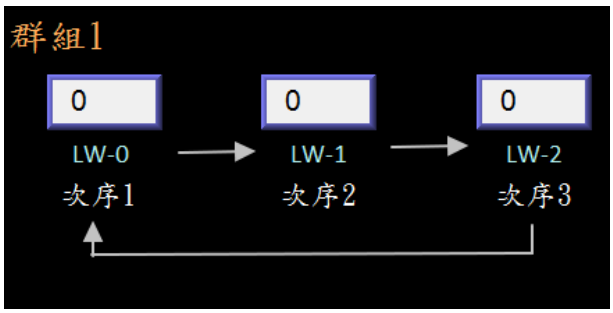
设计群组式的数值输入元件。
通过设定输入次序及输入次序群组达成多个输入元件连续输入。当完成目前元件输入动作后，系统将自动跳至下一个同一群组的输入元件继续输入。

1. 建立三个数值输入元件，皆使用“输入次序”，次序分别为次序 1、次序 2、及次序 3，并设定为“群组 1”。则输入顺序如下图：

LW-0

LW-1

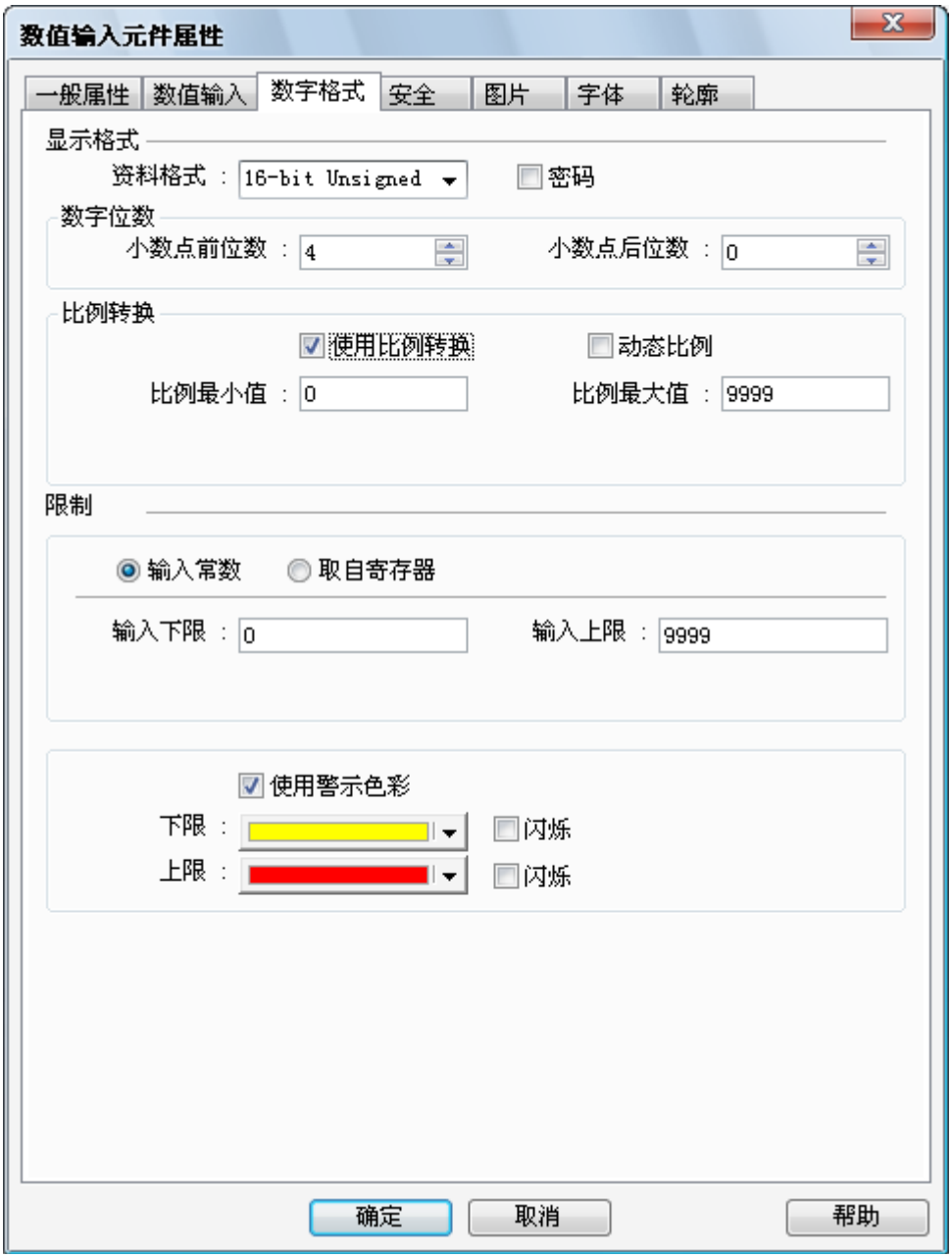
LW-2



2. 若在输入完最后一个数值输入元件后，即停止跳出键盘结束所有的输入动作，则勾选“输入完成后不再按次序输入”即可。

数字格式设定

cMT-SVR 系列



设定	描述
显示格式	**数据格式** 支持二进码十进数 (BCD)、二进制 (Binary)、十进制带/不带符号整数 (Signed/Unsigned)、十六进制 (Hex)、浮点数 (Float) 的读取与写入。使用 16-bit 格式时，会占用寄存器一个字符，使用 32-bit 格式时，则是占用两个字符。 **密码** 数值显示时将使用 ”*” 符号代替所有数字。
数字位数	**小数点前位数**

	小数点前的显示位数。 **小数点后位数** 小数点后的显示位数。
比例转换	**内插法** 所显示的数据是利用寄存器中的原始数据经过换算后所获得。 选择此项功能必须设定“比例最小值”、”比例最大值”与“限制”项目中的“输入下限”、”输入上限”。请见以下范例 2。 动态比例： 比例转换的上下限可由指定寄存器设定。请见以下范例 3。 **宏副函式 (不适用于 cMT-SVR 系列)** 元件的读取/写入数据将经过宏运算而获得。 读取转换：元件的数值将经过宏运算后显示。 写入转换：写入元件的数值将经过宏运算后回传。 使用此功能，请见 《13.9.2.1 数值元件使用宏副函式规则》。
其它选项 (适用于 cMT-SVR 系列)	**显示上下限值** 勾选此选项，在输入数值时，元件旁会显示该元件地址的上下限值。 **显示前一数值** 勾选此选项，在输入数值时，元件旁会显示该元件地址更改前的数值。 
限制	用来设定输入数值上下限的来源，并可设定警示颜色与警示效果。 **输入常数** 选择输入数值的上下限分别来自“PLC 下限”与“PLC 上限”中的设定值。若输入值不在上下限定义的范围内，将无法更改寄存器内的数值。 **取自寄存器** 上下限可由指定寄存器设定。请见以下范例 4。
使用警示色彩	**下限** 当寄存器内的数值小于下限值时，元件会使用此项颜色显示数值。 **上限** 当寄存器内的数值大于上限值时，元件会使用此项颜色显示数值。 **闪烁** 当寄存器内的数值小于下限值或大于上限值时，元件会使用闪烁的效果加以警示。

数值元件使用宏副函式规则

- 必须有回传值且只能有一个参数。

例如：

sub char test (short a)　// (正确)

sub test (char a)　// (错误，没有回传值)

sub char test (char a, char b)　// (错误，有两个参数)

- 数值元件的数据格式须对应到特定的参数类型。

如下表所示：

宏参数类型	数值元件的数据格式
short	16-bit Signed
Int	32-bit Signed
unsigned short	16-bit BCD, 16-bit HEX, 16-bit Binary, 16-bit Unsigned
unsigned int	32-bit BCD, 32-bit HEX, 32-bit Binary, 32-bit Unsigned
float	32-bit Float

假设一数值元件的数据格式为 16-bit Unsigned 时，只能选择参数类型为 unsigned short 的宏副函式，例如：

sub char test(unsigned short a)　// (正确)

sub char test(char a)　// (错误)

- 仅可存取本地触摸屏的地址。

例如：

GetData(var, ""ocal HMI"" LB, 0, 1) // (正确)

GetData(var, ""ODBUS RTU"" 0x, 0, 1) // (错误)

- 无法呼叫下列函数：

ASYNC_TRIG_MACRO, SYNC_TRIG_MACRO, DELAY, FindDataSamplingDate, FindDataSamplingIndex, FindEventLogDate, FindEventLogIndex, INPORT, INPORT2, OUTPORT, PURGE, TRACE

- 无法使用下列循环语句：

For-Next, While-Wend

范例 2

比例转换使用“内插法”公式如下：

$$新数值 = 比例最小值 + (原数值 - PLC下限) \times \frac{比例最大值 - 比例最小值}{PLC上限 - PLC下限}$$

以下图的设定为例，当原始数据是 15 时，则经过换算得到的数值为 40。

范例 3

比例转换使用“内插法”时，比例最小值/比例最大值可由指定寄存器设定。当写入地址为 LW-n，则比例最小值/比例最大值会根据以下的规则自动被设定为：

地址格式	16-bit	32-bit
动态比例地址	LW-n	LW-n
比例最小值	LW-n	LW-n
比例最大值	LW-n+1	LW-n+2

以下表为例，当“动态限制地址”为 LW-100 时，则比例最小值/比例最大值地址会自动被设定为：

地址格式	16-bit	32-bit
动态比例地址	LW-100	LW-100
比例最小值	LW-100	LW-100
比例最大值	LW-101	LW-102

范例 4

上下限可由指定寄存器设定。当写入地址为 LW-n，则上/下限会根据以下的规则自动被设定为：

地址格式	16-bit	32-bit
寄存器地址	LW-n	LW-n
下限	LW-n	LW-n
上限	LW-n+1	LW-n+2

以下表为例，当“寄存器地址”为 LW-100 时，则上/下限的地址会自动被设定为：

地址格式	16-bit	32-bit
寄存器地址	LW-100	LW-100
下限	LW-100	LW-100
上限	LW-101	LW-102

范例 5

此范例说明“数值输入”元件如何使用“宏副函式”执行比例转换。

假设有两个函式库分别如下：一个执行“读取转换”，另一个执行“写入转换”。

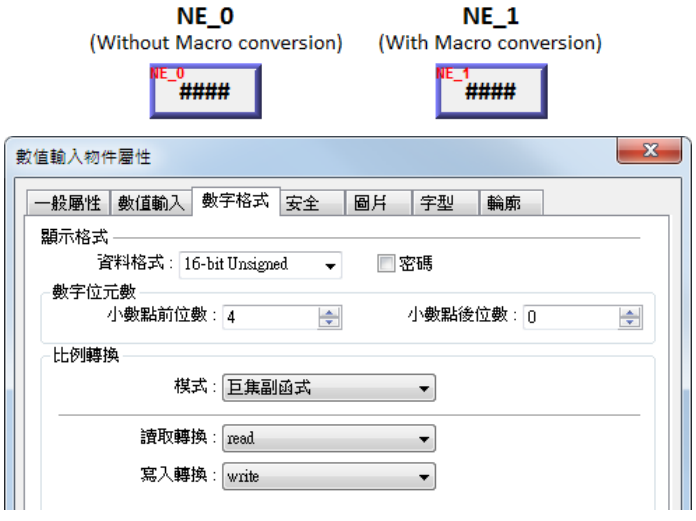
Read Conversion

```
sub short read(unsigned short a)

short b
b = a + 10
return b

end sub
```

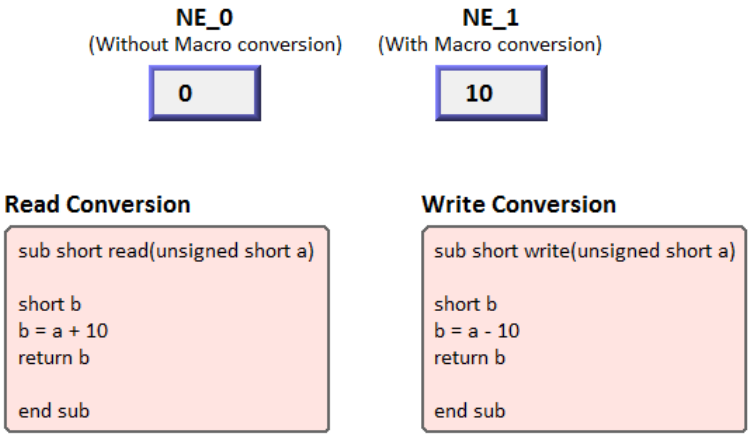
Write Conversion

```
sub short write(unsigned short a)

short b
b = a - 10
return b

end sub
```

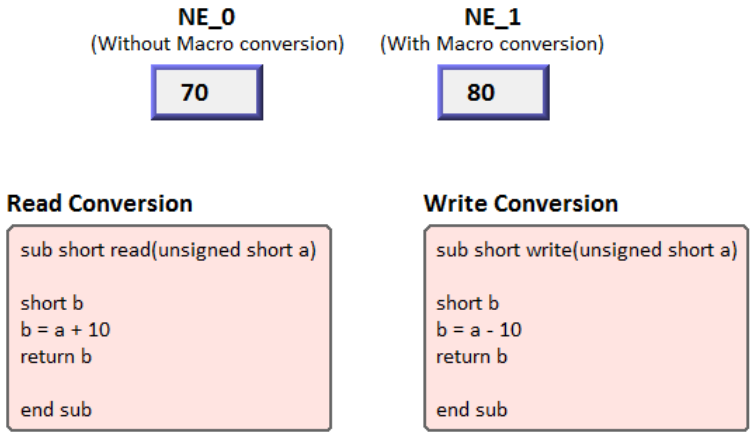
1. 建立两个控制地址相同的“数值输入”元件 NE_0 及 NE_1，令 NE_1 使用“宏副函式”执行读取/写入比例转换。



2. 当在 NE_0 写入 0 时， NE_1 会执行“读取转换”换算为 10。



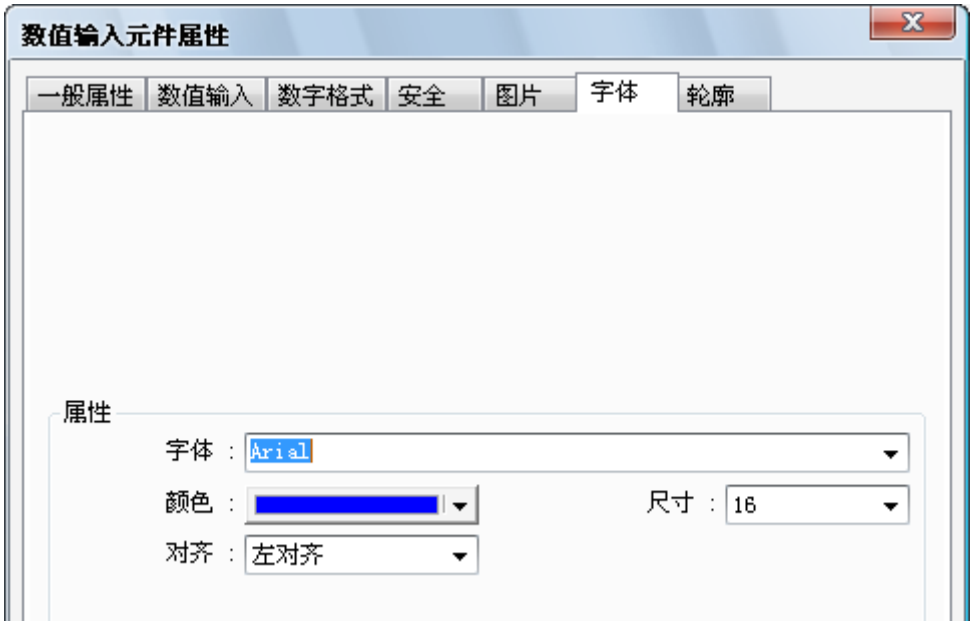
当在 NE_1 输入 80 时，会执行“写入转换”换算后为 70，故 NE_0 显示为 70。

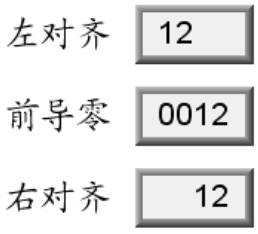


Note

- 当一数值输入元件同时使用“读取转换”及“写入转换”时，若在此元件输入一数值，则元件显示的数值会先执行“写入转换”，再根据转换算后的数值执行“读取转换”。若在此范例中，”写入转换”设定为 b = a – 20，则于 NE_1 写入 80 后，会先执行“写入转换”回传数值 60，再执行“读取转换”显示数值 70。

字型设定



设定	描述
颜色	当数值在上下限的范围内时，使用此项颜色显示。
对齐	**左对齐：**数值靠左显示。 **置中对齐：**数值置中显示。 **右对齐：**数值靠右显示。 **前导零：**数值不满设定的位数前会补零。 
尺寸	设定字号。

13.10 字符输入与字符显示

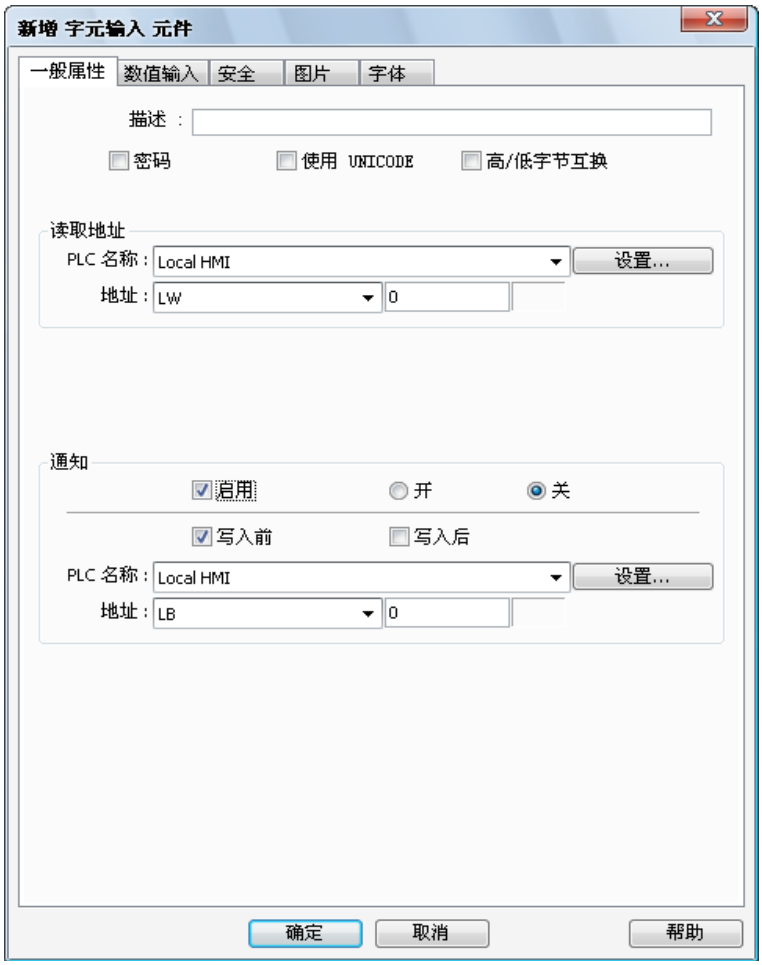
13.10.1 概要

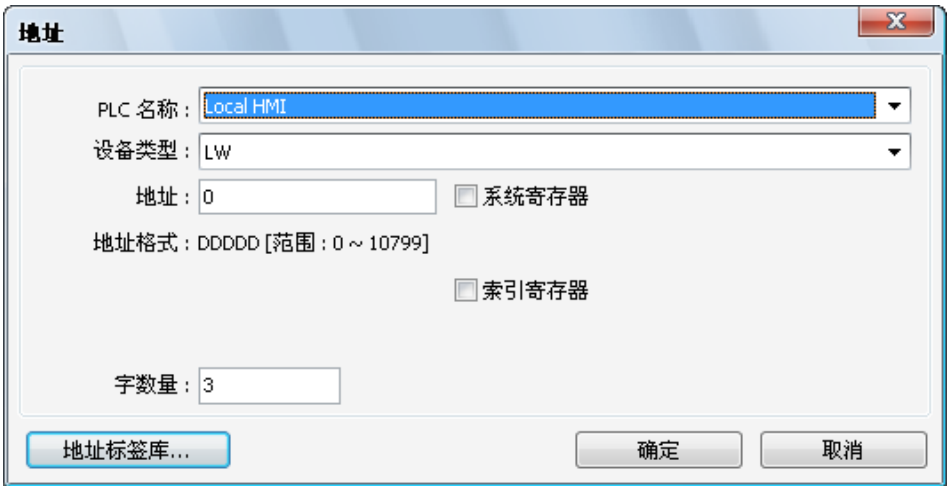
“字符输入”与“字符显示”元件使用 ASCII 编码的方式显示所指定寄存器中的数据，其中“字符输入”元件可用键盘来输入数值，更改寄存器内的数据。

13.10.1 设定

按下工作列上的“字符输入/字符显示”按钮后即会开启“字符输入/字符显示元件属性对话框”，正确设定各项属性后按下确认键，即可新增一个“字符输入/字符显示”元件。“字符输入”元件比“字符显示”元件多了“通知”。

一般属性设定

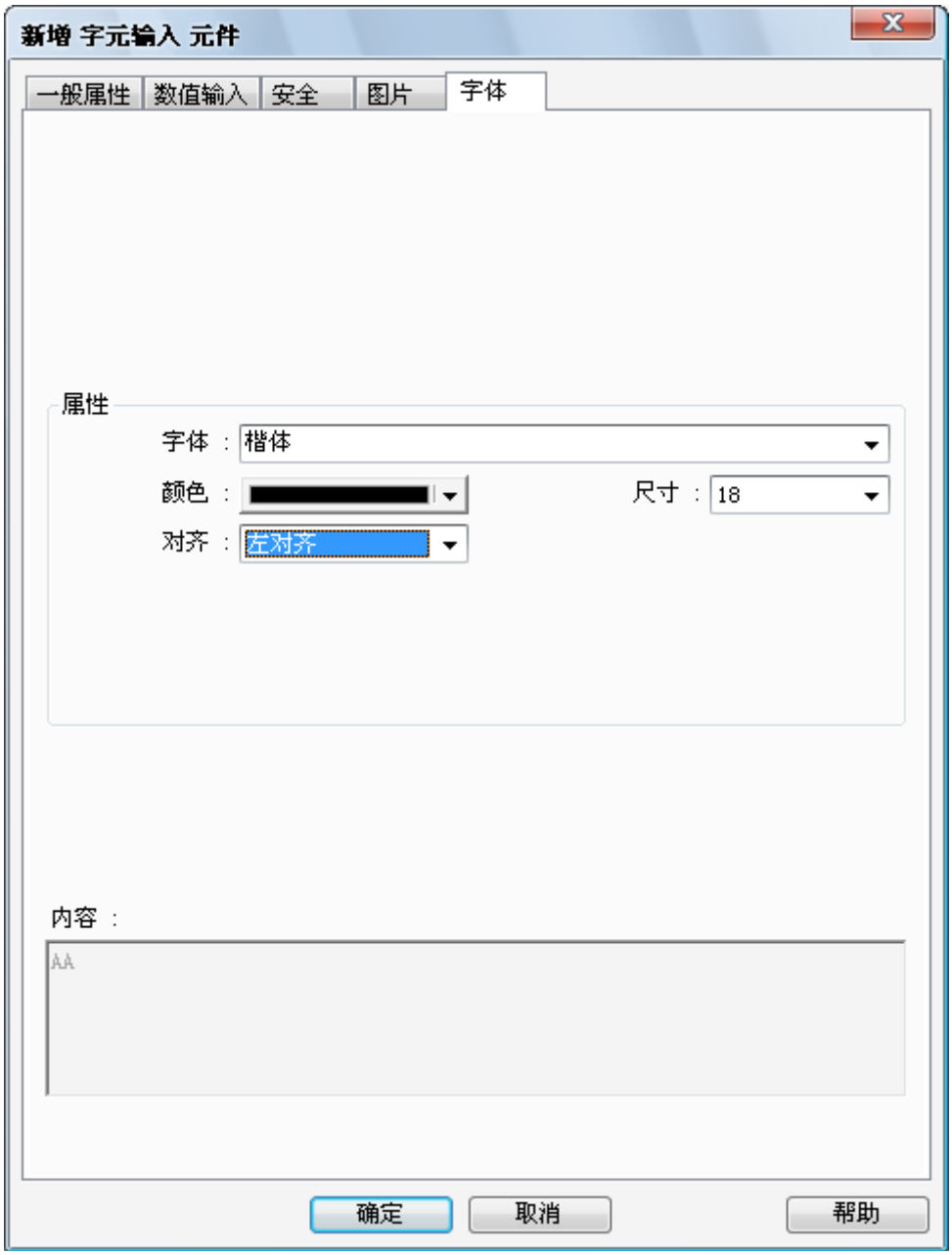


设定	描述
密码	字符显示时将使用“*”符号代替所有字符。
使用 UNICODE	可显示 UNICODE 格式的数据。否则系统会显示字符成 ASCII 格式。此功能可搭配功能键的“ASCII/UNICODE”。
高字节/低字节互换	正常情况下，ASCII code 的显示顺序为“高字节”+“低字节”。勾选此功能后，则显示顺序改为“低字节”+“高字节”。  高低位元不互换　　高低位元互换
读取地址	点选“设置”后，可以选择字符寄存器设备类型的“PLC 名称”，”地址”，”设备类型”，”系统寄存器”，”索引寄存器”来显示字符。用户亦可选用“地址标签库”里设定好的地址标签，也可在“一般属性”页中直接设定 PLC 名称、寄存器名称及地址。  **字符数量** 选择文字最多可显示的数据长度，单位为 word。

Note

- 使用 UNICODE 时，一个 UNICODE 文字等于一个字符 (word)；而使用 ASCII 时，一个 ASCII 文字等于一个字节 (byte)，所以一个字符 (word) 可以有两个 ASCII 文字。(1 个字符 (word) 等于 2 个字节 (byte))

字型设定



设定	描述
属性	可设定文字显示时所使用的字型、字号与颜色，另外也包括文字对齐的方式。 **对齐** 左对齐：文字靠左显示 置中对齐：文字置中显示 右对齐：文字靠右显示

13.11 间接窗口

13.11.1 概要

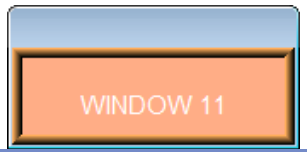
“间接窗口”元件为使用字符寄存器控制指定编号的窗口的弹出及关闭。弹出窗口的显示范围有两种方式，第一种是先在窗口上定义一个显示区域，在此显示区域内显示弹出窗口的内容。所显示的弹出窗口的长度与高度不会大于此显示区域；第二种是使用“自动调整窗口尺寸”功能，启用此功能后不需事先定义弹出窗口的区域，系统会自动根据对应的弹出窗口尺寸调整其显示区域。欲关闭弹出窗口只需将控制的字符寄存器的内容设定为 0 即可。”直接窗口”与“间接窗口”的分别在于直接窗口是利用位状态进行窗口控制，而间接窗口则是利用字符数值进行窗口控制。

13.11.2 设定

按下工作列上的“间接窗口”按钮后即会开启“间接窗口”元件属性对话框，正确设定各项属性后按下确认键，即可新增一个“间接窗口”元件。

一般属性设定

cMT-SVR 系列

设定	描述
读取地址	点选“设置”后选择字符寄存器设备类型的“PLC 名称”,”地址”,”设备类型”,”系统寄存器”，”索引寄存器”来控制窗口弹出。用户也可在“一般属性”页中设定地址。
属性	**类型** 设定弹出窗口样式。支持两种样式： ● 隐藏窗口控制条 弹出的子窗口不包含窗口控制条，无法拖曳移动窗口。  ● 显示窗口控制条 弹出的子窗口包含窗口控制条，它的窗口位置可藉由控制条任意被拖曳。 
使用窗口编号偏移量	弹出的窗口的编号会等于寄存器中的数据加上偏移量。如：控制字符寄存器中

	的数据为 20，偏移量为 5，则会弹出窗口编号 25。
自动调整窗口尺寸	系统会根据弹出的窗口的尺寸调整其显示范围，并自动调整相对位置。 **(相对于元件显示区域)** 弹出的窗口的位置基准点 (对应于间接窗口元件)。如：若设定右下角为基准点，则任一窗口弹出时，会以自身的右下角对准此基准点。若设定左上角为基准点，则任一窗口弹出时，会以自身的左上角对准此基准点，依此类推。请见以下范例 1。

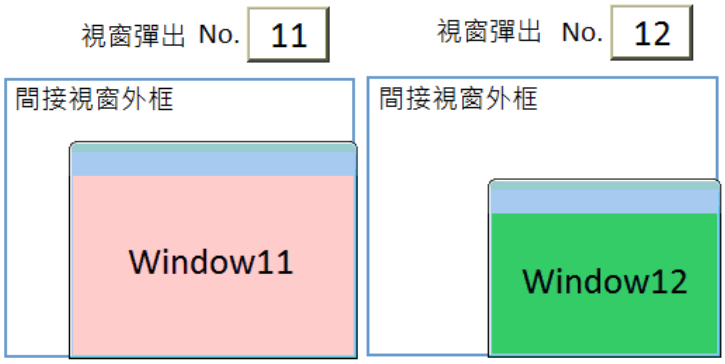
范例 1

若有两个弹出窗口编号 11 及编号 12，使用字符寄存器地址 LW-0 做控制，勾选 “自动调整窗口尺寸” 功能并设定右下角为弹出基准点。

1. 建立一个“间接窗口”元件，读取地址为 LW-0，勾选“自动调整窗口尺寸”。
2. 在“间接窗口”元件上调整好弹出窗口的显示区域。

3. 将数值 11 写入 LW-0，则弹出窗口编号 11。

4. 将数值 12 写入 LW-0，则弹出窗口编号 12。
5. 将数值 0 写入 LW-0，则关闭弹出窗口。



要关闭弹出窗口除了可以对控制字符寄存器写入数值 0 之外，也可在弹出窗口上设计一个“功能键”元件，选择“关闭窗口”模式，在按下此元件后即可关闭弹出窗口。

Note

- 在程序运作时最多可同时显示 24 个窗口。
- 系统不允许在一个基本窗口上使用 2 个直接 (或间接) 窗口弹出同一个窗口。
- 如果弹出的窗口有“垄断”属性，当窗口弹出后，背景窗口的操作将完全暂停，直到垄断的窗口被关闭才可操作其它窗口。

13.12 直接窗口

13.12.1 概要

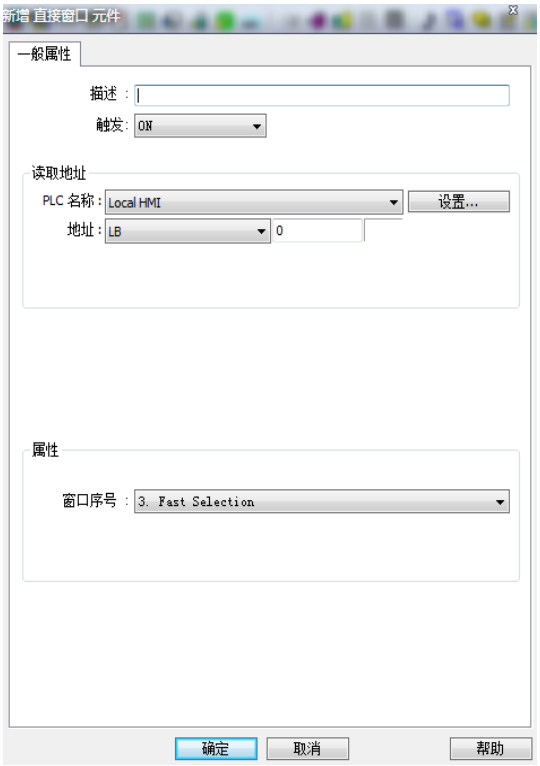
“直接窗口”元件是用位寄存器去控制弹出窗口的开启及关闭。首先，在窗口上定义一个显示区域，当所指定的位寄存器的状态改变时，将在此显示区域内显示此窗口的内容。所显示窗口的长度与高度不会大于此显示区域。将控制此弹出窗口的位寄存器状态恢复即可关闭此弹出窗口。
“直接窗口”与“间接窗口”的分别为直接窗口是由位寄存器做控制且须先设定弹出的窗口编号，而间接窗口则是由字符寄存器做控制并根据寄存器内的数据弹出对应的窗口。

13.12.2 设定

按下工作列上的“直接窗口”按钮后即会开启“直接窗口”元件属性对话框，正确设定各项属性后按下确认键，即可新增一个“直接窗口”元件。

一般属性设定

cMT-SVR 系列



eMT、iE、cMT-HD 系列

设定	描述
读取地址	点选“设置”后选择位寄存器设备类型的”PLC 名称”，”地址”，”设备类型”，”系统寄存器”，”索引寄存器”来控制窗口弹出。用户也可在“一般属性”页中设定地址。
属性	**类型** 设定弹出窗口样式。支持两种样式，”隐藏窗口控制条”和“显示窗口控制条”。 **窗口序号** 设定要弹出的窗口号码。

范例 1

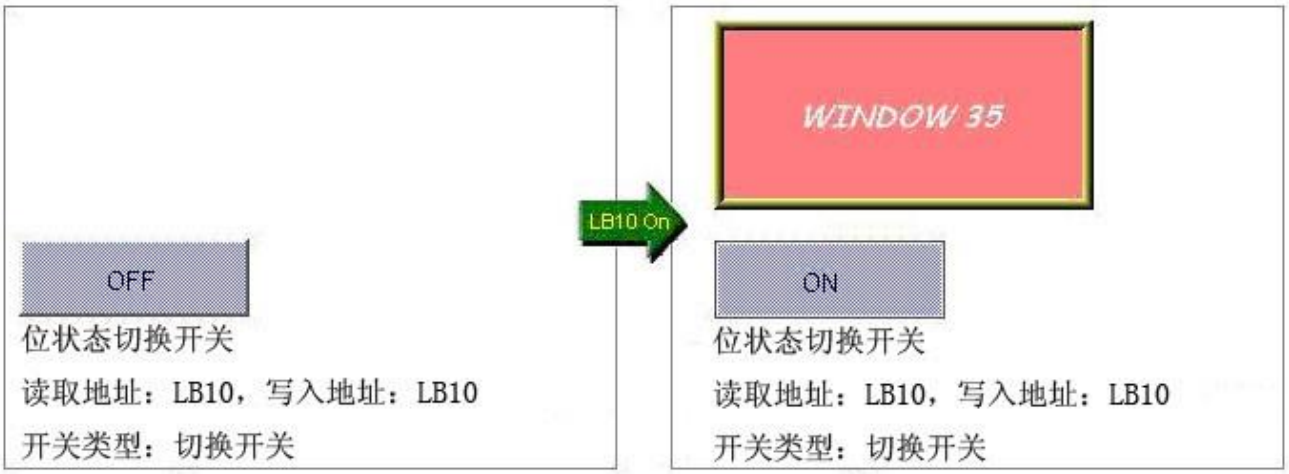
现在使用一个简单的例子说明“直接窗口”的使用方式，下图为“直接窗口”元件的设定内容，此时使用 LB-10 来决定是否显示”窗口 35”。

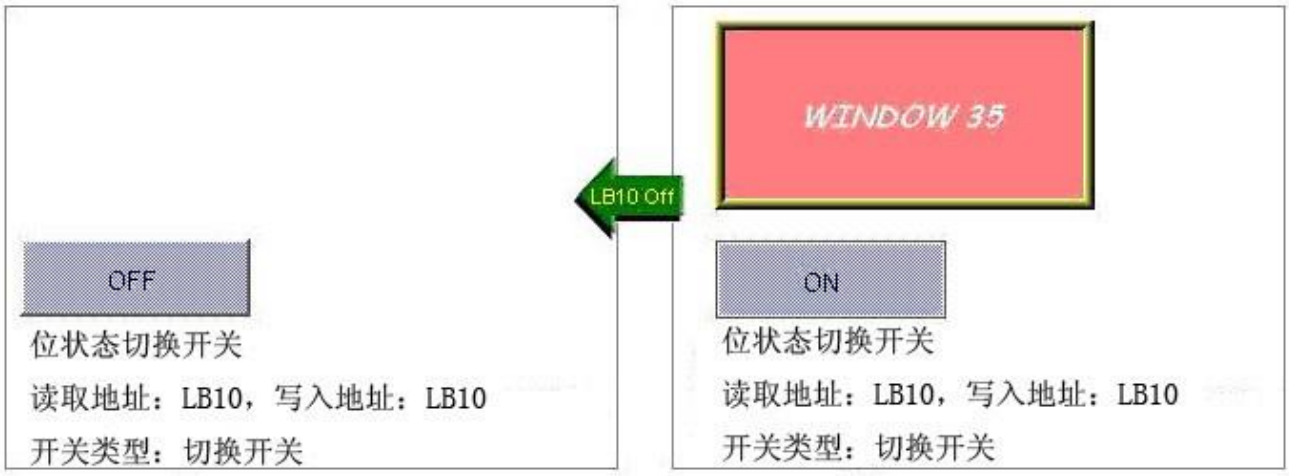
位状态切换开关

读取地址：LB10，写入地址：LB10

开关类型：切换开关

当 LB-10 状态为 ON 时，”窗口 35”将出现；当 LB-10 状态为 OFF 时，”窗口 35”将消失。参考下图。





Note

最多可同时开启 24 个弹出窗口，包含系统讯息窗口、直接窗口和间接窗口。

- 系统不允许在一个基本窗口上使用 2 个直接(或间接) 窗口弹出同一个窗口。
- 如果弹出的窗口有“垄断”属性，当窗口弹出后，背景窗口的操作将完全暂停，直到垄断的窗口被关闭才可操作其它窗口。

13.13 移动图形

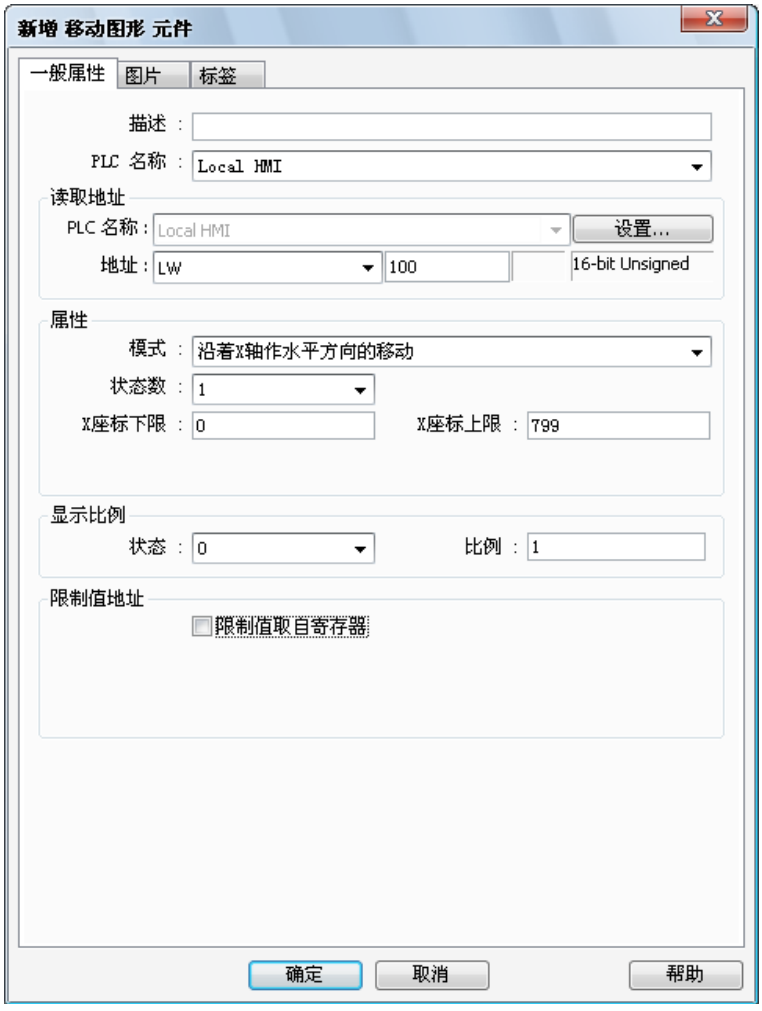
13.13.1 概要

“移动图形”元件可定义元件的状态和移动距离。元件会根据读取地址及连续的寄存器内的数据，改变元件的状态与元件的移动距离。

13.13.2 设定

按下工作列上的“移动图形”按钮后即会开启“移动图形”元件属性对话框，正确设定各项属性后按下确认键，即可新增一个“移动图形”元件。

一般属性设定



设定	描述
读取地址	点选“设置”后选择字符寄存器设备类型的”PLC 名称”，”地址”，”设备类型”，”系统寄存器”，”索引寄存器”来做为控制移动图形状态和移动位置的目标。用户也可在“一般属性”页中设定地址。
属性	选择元件的移动方式及移动的范围，请见下方《13.13.2.1 属性模式说明》。
显示比例	元件各个状态的图形在显示时，可以分开设定缩放比例，参考下图。 
限制值地址	利用“限制值地址”内的数据动态调整元件的显示区域，请见范例 1。

范例 1

利用“限制值地址”内的数据动态调整元件的显示区域。当限制值地址为 LW-n，则 X 轴与 Y 轴的上下限会根据以下的规则自动被设定为：

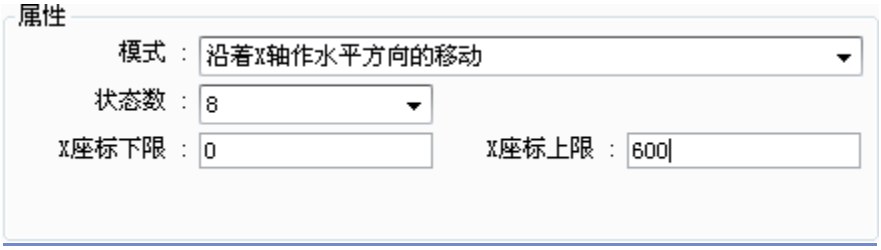
数据格式	16-bit	32-bit
X 轴坐标下限	LW-n	LW-n
X 轴坐标上限	LW-n+1	LW-n+2
Y 轴坐标下限	LW-n+2	LW-n+4
Y 轴坐标上限	LW-n+3	LW-n+6

属性模式说明

(以下假设读取地址使用 LW-n)

●沿着 X 轴作水平方向的移动

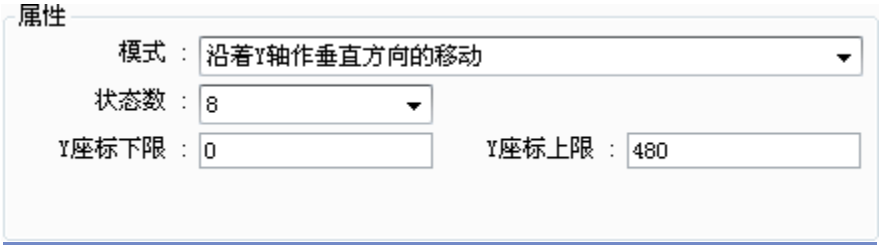
只允许元件沿着 X 轴作水平方向的移动。移动范围由“X 轴坐标下限”与“X 轴坐标上限”来决定。



数据格式	16-bit	32-bit
控制元件状态地址	LW-n	LW-n
控制元件 X 轴移动距离地址	LW-n+1	LW-n+2

●沿着 Y 轴作垂直方向的移动

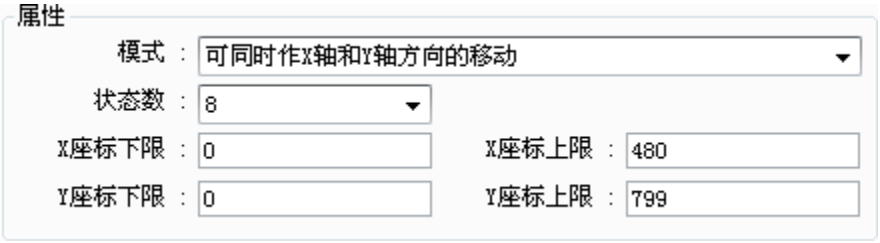
只允许元件沿着 Y 轴作垂直方向的移动。移动范围由 “Y 轴坐标下限” 与 “Y 轴坐标上限” 来决定。



数据格式	16-bit	32-bit
控制元件状态地址	LW-n	LW-n
控制元件 Y 轴移动距离地址	LW-n+1	LW-n+2

●可同时作 X 轴与 Y 轴方向的移动

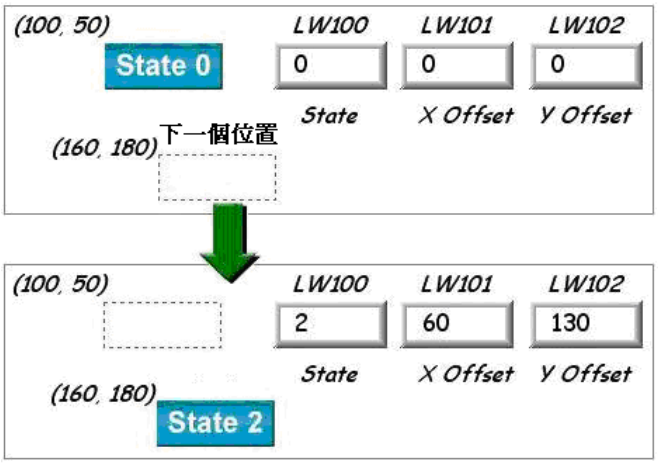
允许元件沿着 X 轴与 Y 轴移动。移动范围由 “X 轴坐标下限”、”X 轴坐标上限” 与 “Y 轴坐标下限”、”Y 轴坐标上限” 来决定。



数据格式	16-bit	32-bit
控制元件状态地址	LW-n	LW-n
控制元件 X 轴移动距离地址	LW-n+1	LW-n+2
控制元件 Y 轴移动距离地址	LW-n+2	LW-n+4

举例来说，若寄存器为 LW-100，且变量类型使用”16-bit Unsigned”，则 LW-100 存放元件的状态，LW-101 存放 X 轴方向的移动距离，LW-102 存放 Y 轴方向的移动距离。

以下图为例，元件的地址为 LW-100 且起始地址为(100，50)，假使现在要移动元件至(160，180)且显示状态 2 的图形，则 LW-100 需设定为 2，LW-101 = 160-100 = 60，”LW102” = 180-50 = 130



●沿着 X 轴按比例作水平方向的移动

只允许元件沿着 X 轴、按比例作水平方向的移动。

公式：$位移距離 = (讀取位址數據 - 輸入下限) \times \frac{比例上限 - 比例下限}{輸入上限 - 輸入下限}$

数据格式	16-bit	32-bit
控制元件状态地址	LW-n	LW-n
控制元件 X 轴移动距离地址	LW-n+1	LW-n+2

●沿着 Y 轴按比例作垂直方向的移动

只允许元件沿着 Y 轴、按比例作垂直方向的移动。公式与“沿着 X 轴按比例作水平方向的移动”相同。

数据格式	16-bit	32-bit
控制元件状态地址	LW-n	LW-n
控制元件 Y 轴移动距离地址	LW-n+1	LW-n+2

●沿着 X 轴按反比例作水平方向的移动

此项功能与“沿着 X 轴按比例作水平方向的移动”相同，但移动方向相反。

●沿着 Y 轴按反比例作水平方向的移动

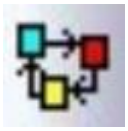
此项功能与“沿着 Y 轴按比例作垂直方向的移动”相同，但移动方向相反。

13.14 动画

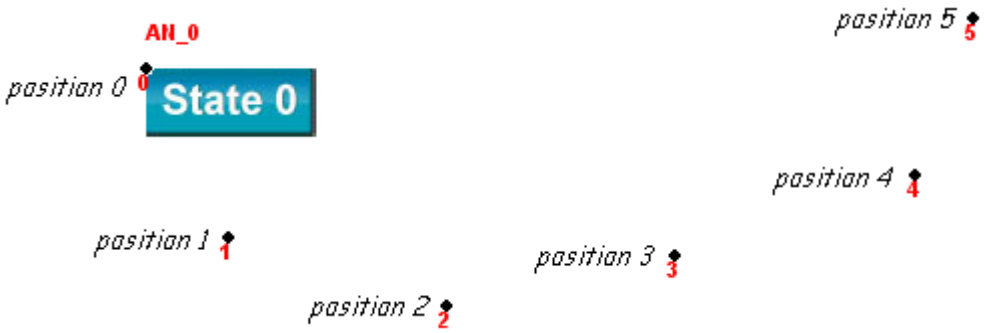
13.14.1 概要

用户可以预先定义“动画”元件的移动轨迹，并利用更改寄存器内的数据，控制元件的状态与元件在移动轨迹上的位置。系统将使用连续两个寄存器内的数据来控制动画元件，第一个寄存器为控制元件的状态，第二个为控制元件的位置。

14.14.2 设定

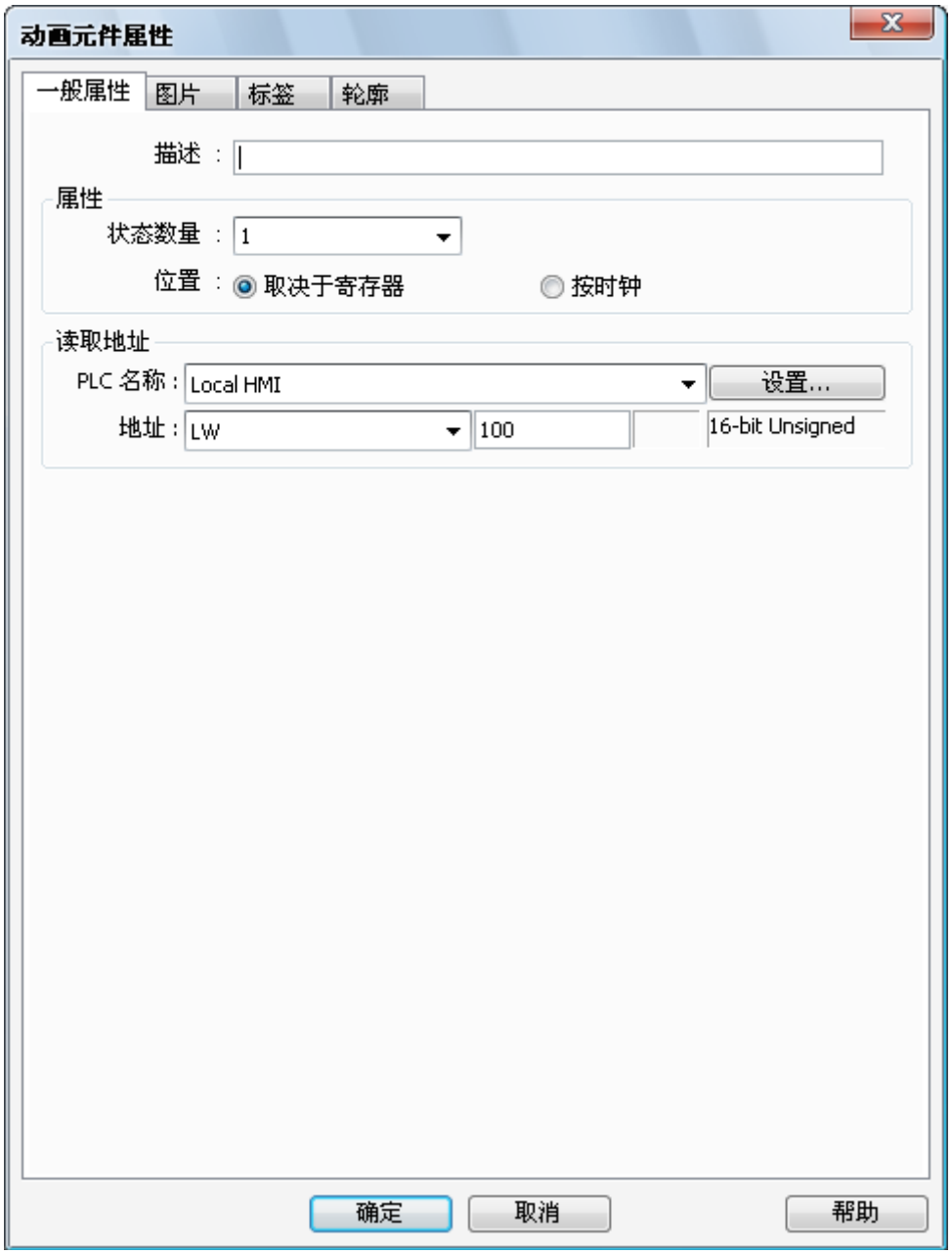


按下工作列上的“动画”按钮。首先，设定元件的移动路径。使用鼠标在编辑画面上点击左键一一指定每个移动位置，然后点击右键即会开启“动画”元件属性对话框，正确设定各项属性后按下确认键，即可新增一个“动画”元件。

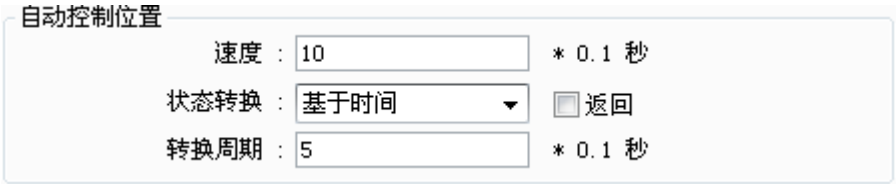


要更改元件的属性，可以使用鼠标左键双击元件所在位置，利用出现的“动画”元件属性对话框，即可更改元件的各项属性。

一般属性设定



设定	描述
属性	**状态数量** 设定元件状态数目。 **取决于寄存器** 由寄存器数据控制元件的状态和位置，此时必须正确设定元件状态与位置的读取地址。请见以下范例 1。 **按时钟** 元件自动改变状态与显示位置，”自动控制位置”项目用来设定状态与显示位置改变方式。



速度：位置改变的速度，单位为 0.1 秒。例如设定为 10，则元件每隔 1 秒钟变换一个位置。

状态转换：状态改变的方式，可以选择“基于位置”与“基于时间”。选择“基于位置”表示位置改变，状态也随着改变。若选择“基于时间”，表示位置使用固定的频率自动变换，变换频率在“转换周期”中设定。

返回：假设元件有 4 个位置，分别为 position 0、position 1、position 2、position 3。若未选择此项设定，当移动到最后一个位置(position 3)后，元件将移动到初始位置 position 0，再重复原来位置改变方式，移动位置整理顺序如下。

position 0→ position 1→ position 2→ position 3→ position 0→ position 1→ position 2…

若选择此项设定，当移动到最后一个位置 (position 3) 后，元件将使用反向的移动方式，移动到初始位置 position 0，再重复原来位置改变方式，移动位置整理顺序如下。

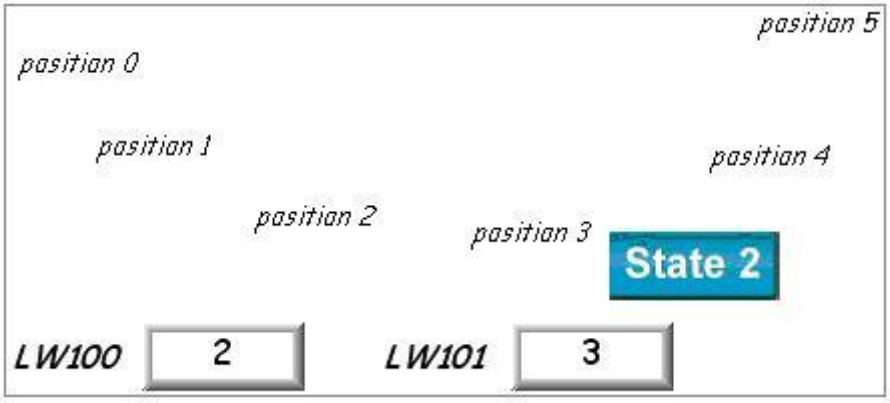
position 0 → position 1→ position 2→ position 3→ position 2→ position 1→ position 0…

范例 1

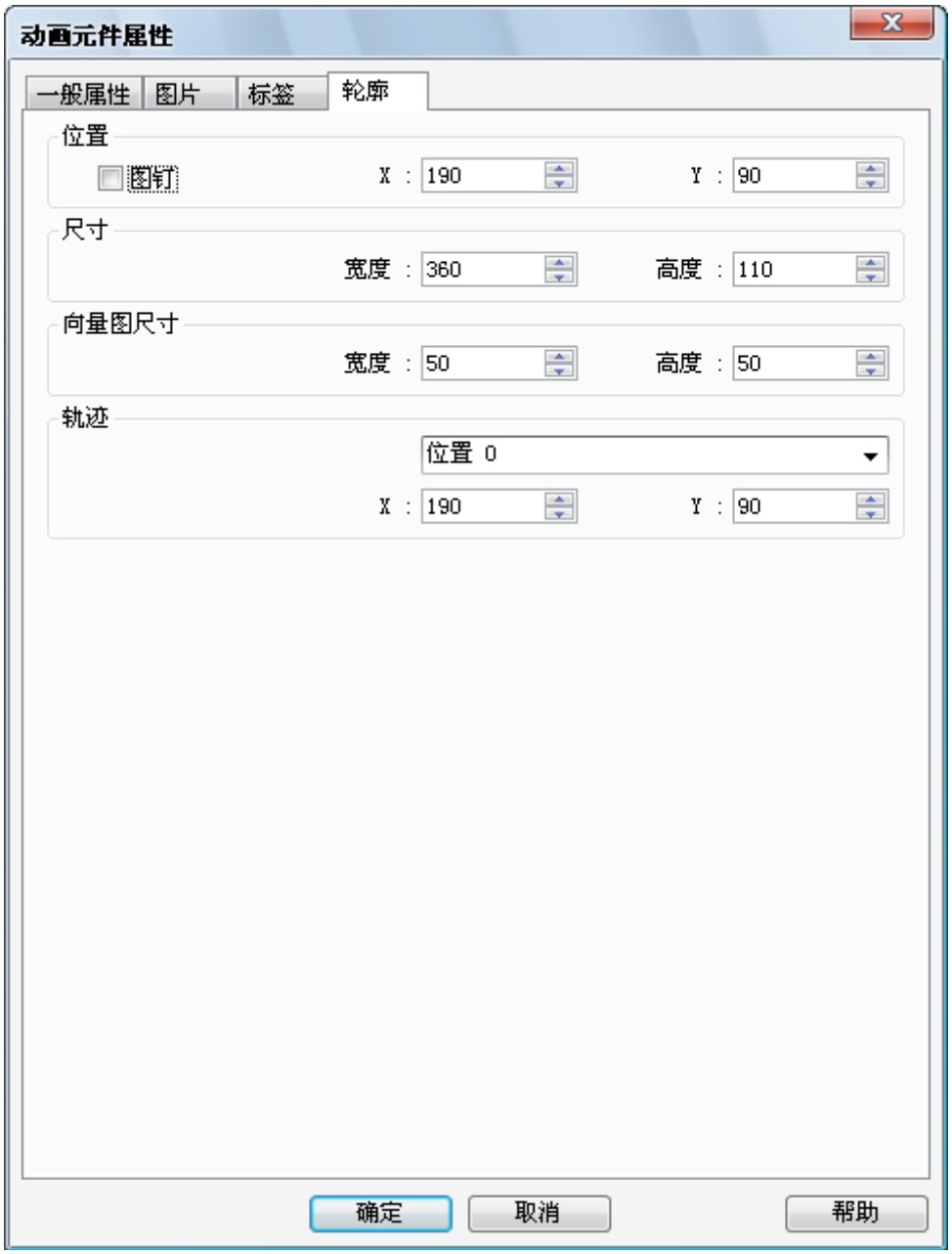
如果元件的状态与位置由寄存器中的数据决定，就必须正确设定元件状态与位置的读取地址。读取地址格式如下表。

数据格式	16-bit	32-bit
控制元件状态地址	LW-n	LW-n
控制元件位置地址	LW-n+1	LW-n+2

举例来说，若寄存器为 LW-100，且格式使用”16-bit Unsigned”，则 LW-100 存放元件的状态，LW-101 存放元件的显示位置。以下图为例，LW-100 = 2，LW-101 = 3，所以元件显示状态 2，并出现在位置 3。


position 5
position 0
position 1
position 4
position 2
position 3
State 2
LW100
2
LW101
3

轮廓设定


动画元件属性
一般属性
图片
标签
轮廓
位置
图钉
X : 190
Y : 90
尺寸
宽度 : 360
高度 : 110
向量图尺寸
宽度 : 50
高度 : 50
轨迹
位置 0
X : 190
Y : 90
确定
取消
帮助

设定	描述
向量图尺寸	设定元件所显示图形的大小。
轨迹	设定移动轨迹上各点的位置。

Note

- 因为一个动画元件可设定多个不同的图片，因此无法使用“使用原尺寸”将所有的图片还原成原始尺寸。

13.15 棒图

13.15.1 概要

“棒图”元件使用百分比例与棒图的方式，显示寄存器中的数据。

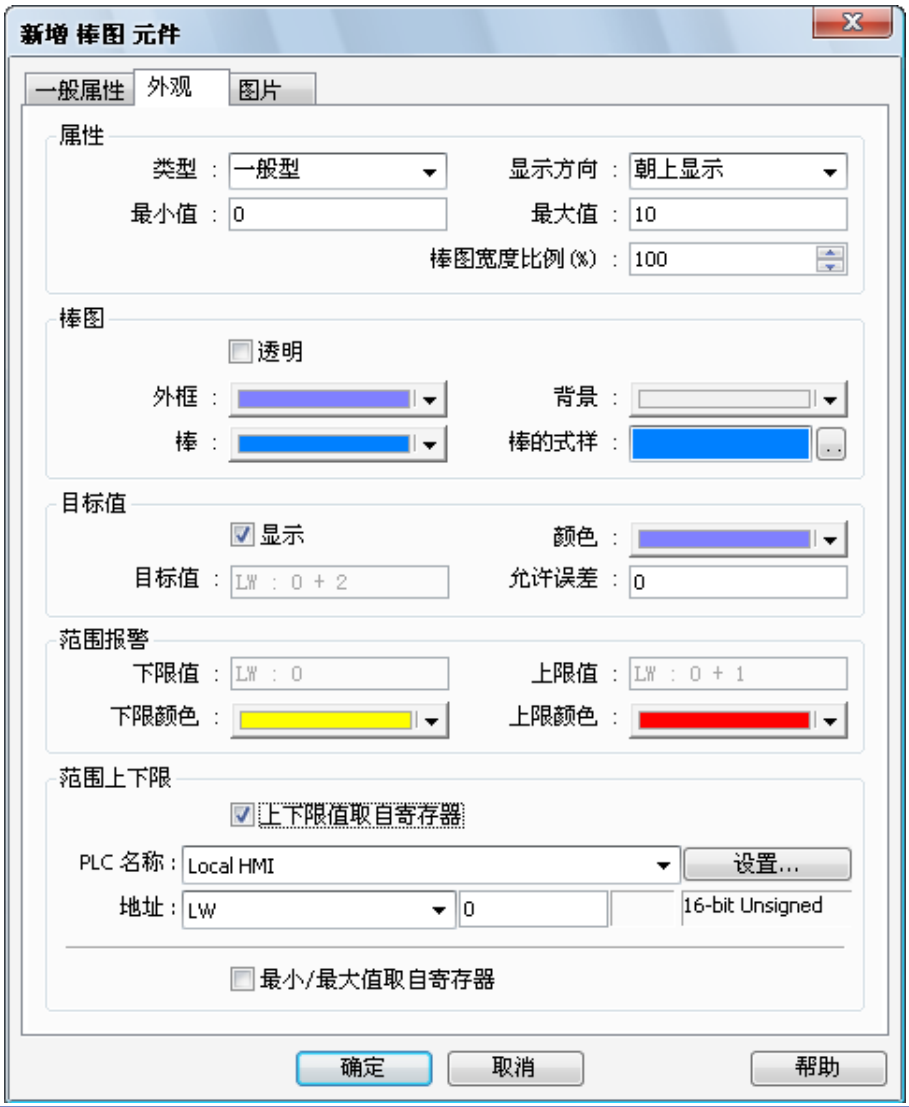
13.15.2 设定

按下工作列上的“棒图”按钮后即会开启“棒图”元件属性对话框，正确设定各项属性后按下确认键，即可新增一个“棒图”物件。

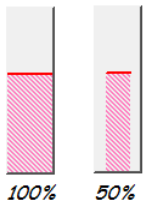
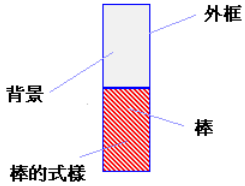
一般属性设定

设定	描述
读取地址	点选“设置”后选择字符寄存器设备类型的”PLC 名称”，”地址”，”设备类型”，”系统寄存器”，”索引寄存器”来做为棒图显示的数据依据。

外观设定



设定	描述
属性	**型式** 选择“一般型”与“偏差型”。当选择“偏差型”时，需设定原点位置。 **显示方向** 选择棒图的显示方向，可以选择“朝上显示”、”朝下显示”、”朝右显示”、”朝左显示”。 **最小值、最大值** 棒图填充的百分比可以利用公式换算而得，请见以下范例 1。

	棒宽度比例(%) 设定棒图的显示宽度与元件宽度间的百分比率。以下图标显示不同比例，100%以及 50%。 
棒图	设定棒图外框、背景颜色与填充区域的样式与颜色，参考下图。 
目标值	当寄存器内的数据符合条件时，填充区域的颜色可以变更为此项目所定义的颜色。请见以下范例 2。
范围警报	当数据大于“上限值”时，填充区域的颜色可以变更为“上限颜色”所定义的颜色；若当数据小于“下限值”时，填充区域的颜色可以变更为“下限颜色”所定义的颜色。
范围上下限	当选择“上下限值取自寄存器”，”范围报警项目”中所使用的“下限值”、”上限值”与“目标值项目”中的“目标值”皆读取自指定的寄存器。指定之寄存器会显示在输入的字段之中。请见以下范例 3。

范例 1

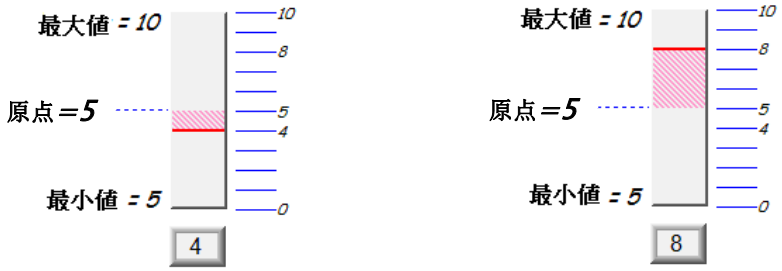
棒图填充的百分比可以利用下列的公式换算而得：

显示区域百分比=(寄存器数据-最小值)/(最大值-最小值)×100%

若（寄存器数据 -“原点位置”）大于 0，则棒图将由“原点位置”的位置往上填充；若（寄存器数据 -“原点位置”）小于 0，则棒图将由“原点位置”的位置往下填充。

下图显示在“原点位置”设定为 5，”最大值”为 10，”最小值”为 0 并使用不同数据时，棒图的填充情形。

当读取数值为 4　　当读取数值为 8

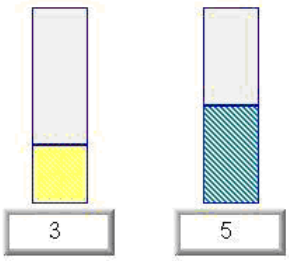


范例 2

当寄存器内的数据符合下列的条件时，填充区域的颜色可以变更为此项目所定义的颜色。

“目标值” - “允许误差” ≤ 寄存器内的数据 ≤ “目标值” + “允许误差”

参考下图，此时 “目标值” = 5，”误差值” = 1，则寄存器的值大于或等于 5-1=4，且小于或等于 5 + 1= 6，填充区域的部分将改变为“目标值颜色”。



目标值=5

允许误差=1

范例 3

范围报警的上下限可由指定寄存器设定。当写入地址为 LW-n，则上/下限及相关数值会根据以下的规则自动设定：(假设“最小/最大值取自寄存器”已勾选)

数据格式	16-bit	32-bit
警报下限值	LW-n	LW-n
警报上限值	LW-n+1	LW-n+2
目标值	LW-n+2	LW-n+4
最小值	LW-n+3	LW-n+6
最大值	LW-n+4	LW-n+8
原点位置	LW-n+5	LW-n+10

13.16 表针

13.16.1 概要

“表针”元件会使用仪表图的方式，指示目前寄存器中的数据。

13.16.2 设定

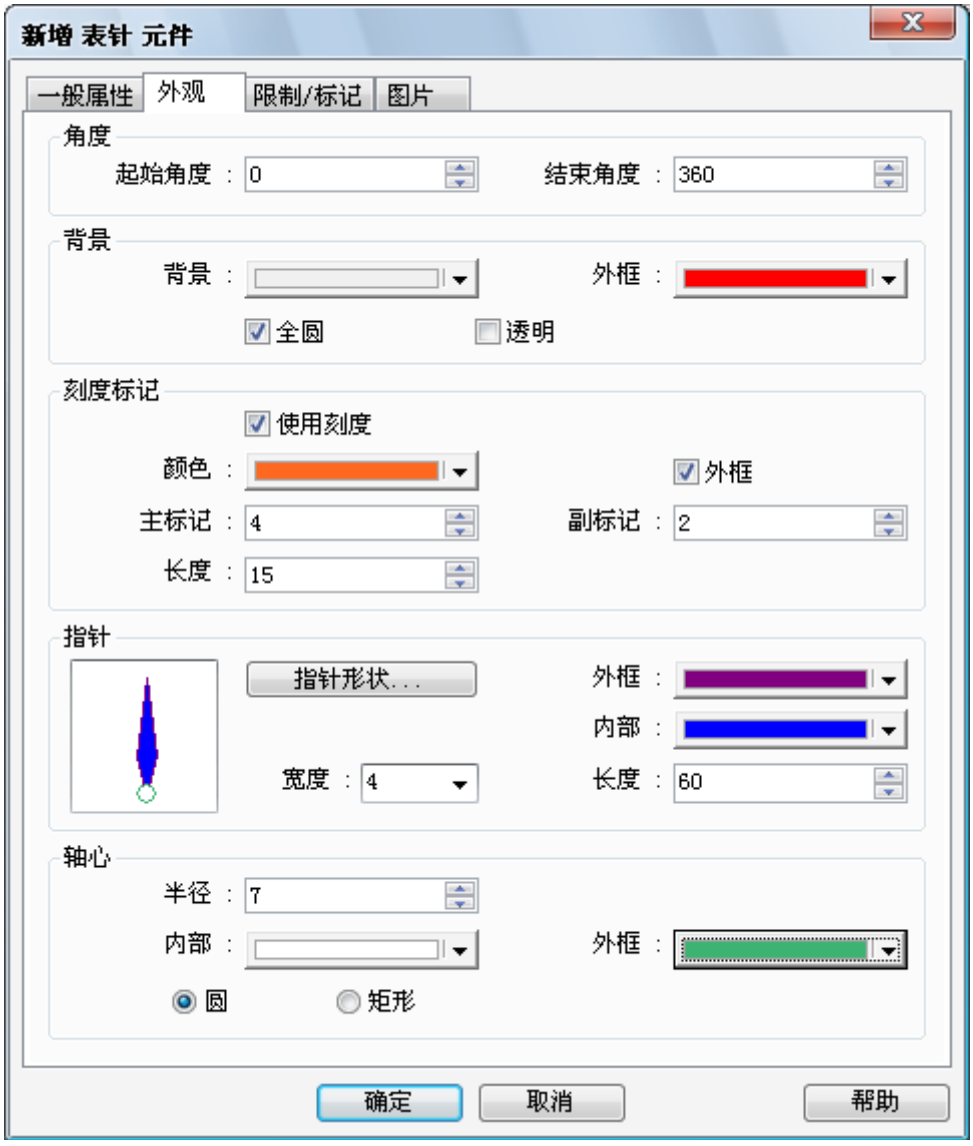
按下工作列上的“表针”按钮后即会开启“表针”元件属性对话框，正确设定各项属性后按下确认键，即可新增一个“表针”物件。

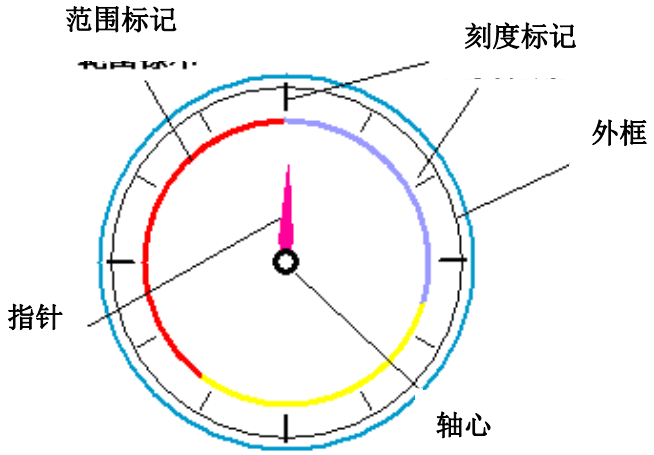
eMT、iE、cMT-HD 系列

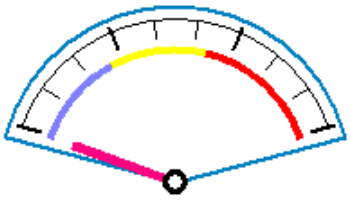
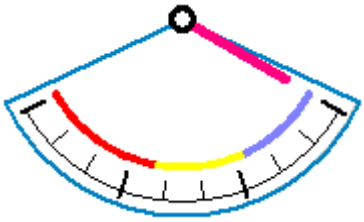
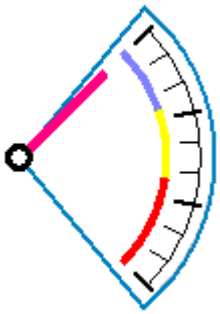
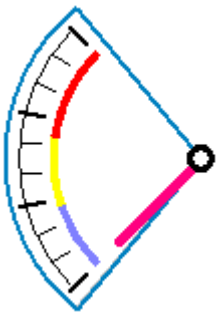
一般属性设定

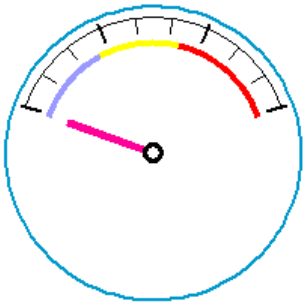
设定	描述
读取地址	点选“设置”后选择字符寄存器设备类型的”PLC 名称”，”地址”，”设备类型”，”系统寄存器”，”索引寄存器”来做为表针显示的数据依据。用户也可在“一般属性”页中设定地址。

外观设定

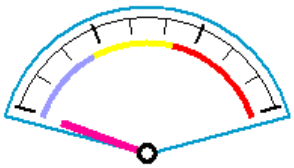




设定	描述
角度	表针元件以图形中点上方为起始点，表示为 0 度或 360 度。向左为逆时针，向右为顺时针。角度可设定范围皆为 0~360 度。不同的设定值所显示的结果，可参考下面的几种不同的设定。 ●“起始角度” = 290。 “结束角度” = 70。  ●“起始角度” = 120。 “结束角度” = 240。  ●“起始角度” = 40。 “结束角度” = 140。  ●“起始角度” = 225。 “结束角度” = 315。 
背景	设定元件的背景与圆周的颜色。 **全圆** 当选择“全圆”时，表针元件将显示整个圆形，反之则显示被定义的角度范围。



使用全圓



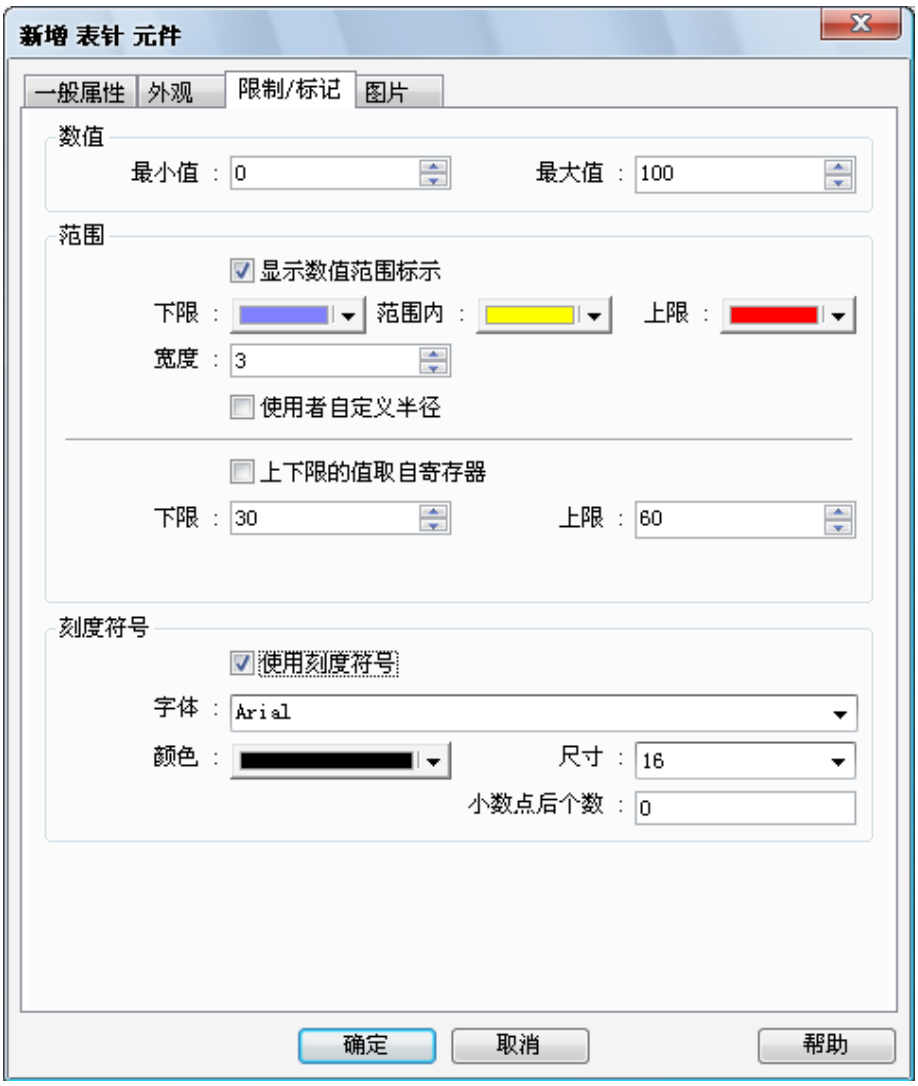
非使用全圓

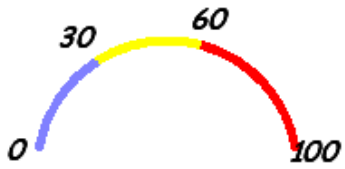
透明

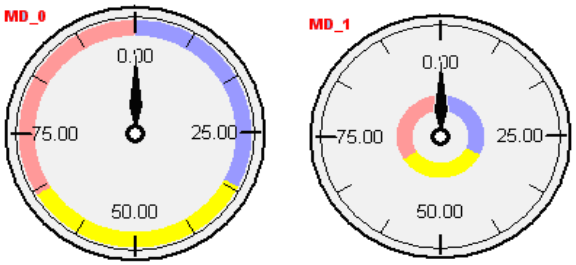
当选择“透明”时，表针元件将不会显示背景与外框颜色。

刻度标记	设定标记的数量与颜色。
指针	设定指针的形状，长宽度和颜色。
轴心	设定轴心的样式与颜色。

限制 / 标记设定



设定	描述
数值	设定元件所要显示的数值范围。请见以下范例 1。
范围	设定上、下限值及指示的颜色与宽度。  **用户自定半径** 数值范围显示色彩离圆心的距离。 例如，设为 80：　　设为 30：



上下限取自寄存器

上下限可由指定寄存器设定。请见以下范例 2。

刻度符号	设定是否使用刻度符号于表针上。

范例 1

设定元件所要显示的数值范围。指针的指示角度计算：

$$\text{角度 (度)} = \frac{\text{读取数据-最小值}}{\text{最大值 - 最小值}} \times (\text{结束角度-起始角度})$$

假设读取的数据为 30，起始角度 0°，结束角度 360°，最小值 0，最大值 100，则指针的指示角度为：

$$\text{角度 (度)} = \frac{30-0}{100-0} \times (360-0) = 108\,(\text{度})$$

范例 2

上下限可由指定寄存器设定。当写入地址为 LW-n，则上/下限会根据以下的规则自动被设定为：

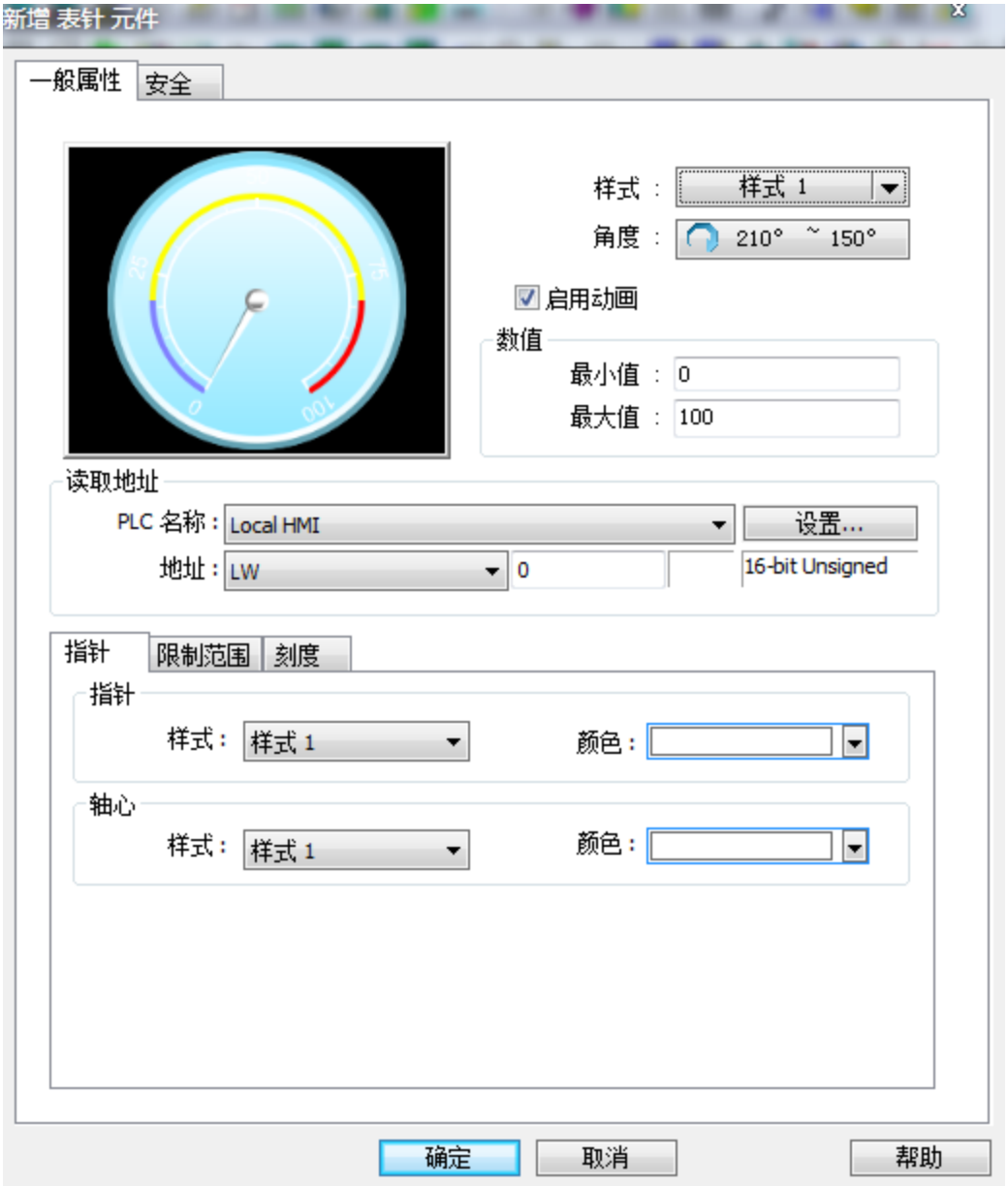
地址格式	16-bit	32-bit
下限	LW-n	LW-n
上限	LW-n+1	LW-n+2

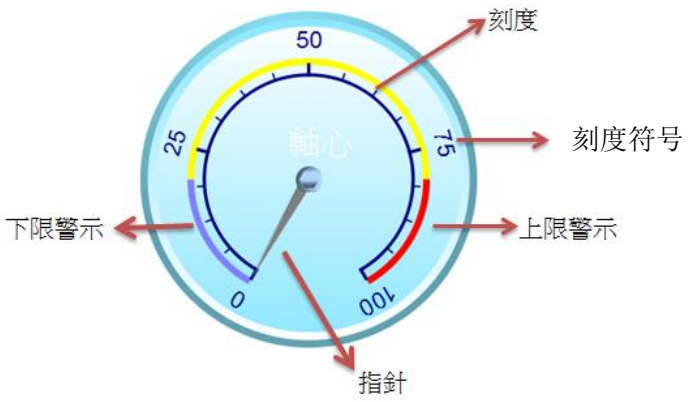
以下表为例，当“寄存器地址”为 LW-100 时，则上/下限的地址会自动被设定为：

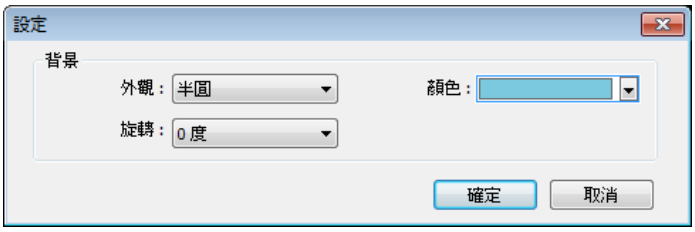
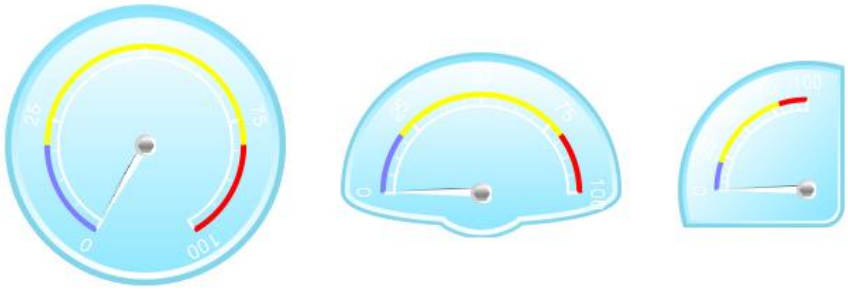
地址格式	16-bit	32-bit
下限	LW-100	LW-100
上限	LW-101	LW-102

cMT-SVR 系列

一般属性设定





设定	描述
样式	分为“自订”、”样式 1”、”样式 2”。点选文字可设定样式属性。选择“自订”时，需自订表针的各种属性，包含指针、轴心、背景图片等。
设定	点选“样式”的文字可设定表针样式。  **外观** 表针的范围。下图是样式 1 的全圆、半圆及四分之一圆的显示样式。  下图是样式 2 的全圆、半圆及四分之一圆的显示样式。  **旋转** 表针的背景图片会依照角度顺时针旋转指定的度数。 **颜色** 表针图片的背景色。
角度	调整指针的刻度范围。表针元件以图形中点上方为起始点，表示为 0 度或 360 度，向右为顺时针，向左为逆时针。 **全圆** 勾选后，表针会从“起始角度”的角度作为原点，根据“顺时针 / 逆时针”方向画一个全圆。数据的最小值及最大值于“数值”字段中的“最小值”及”最大值”设置。

启用动画	设定表针移动时是否有滑动式移动至指定位置。若不勾选则表针在数据变换后会直接跳到指定位置。
数值	设定表针的上下限。
读取地址	表针显示的数据来源地址。
指针	设定指针及轴心的样式。若采用“自订”模式，指针的指向方向必须为朝上才可正确显示。
限制范围	设定上下限警示的颜色。 **上下限取自寄存器** 上下限警示的范围由指定寄存器设定。请见上面范例 2。
刻度	设定刻度的标记间隔及色彩。

13.17 趋势图

13.17.1 概要

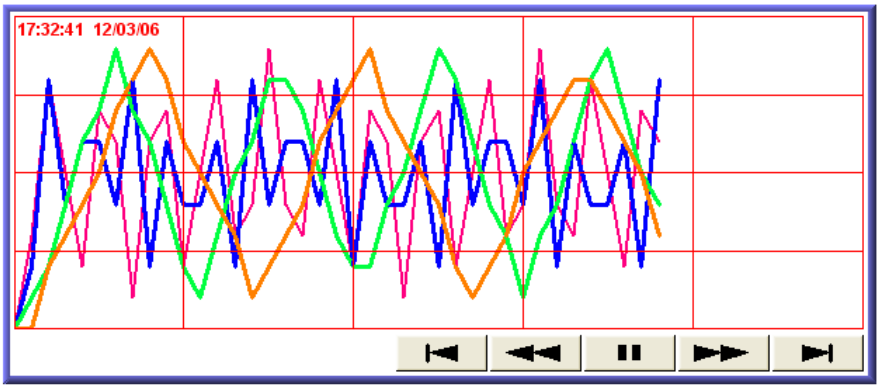
“趋势图”元件会使用连续的线段描绘“资料取样”中的数据，以利资料分析。

13.17.2 设定

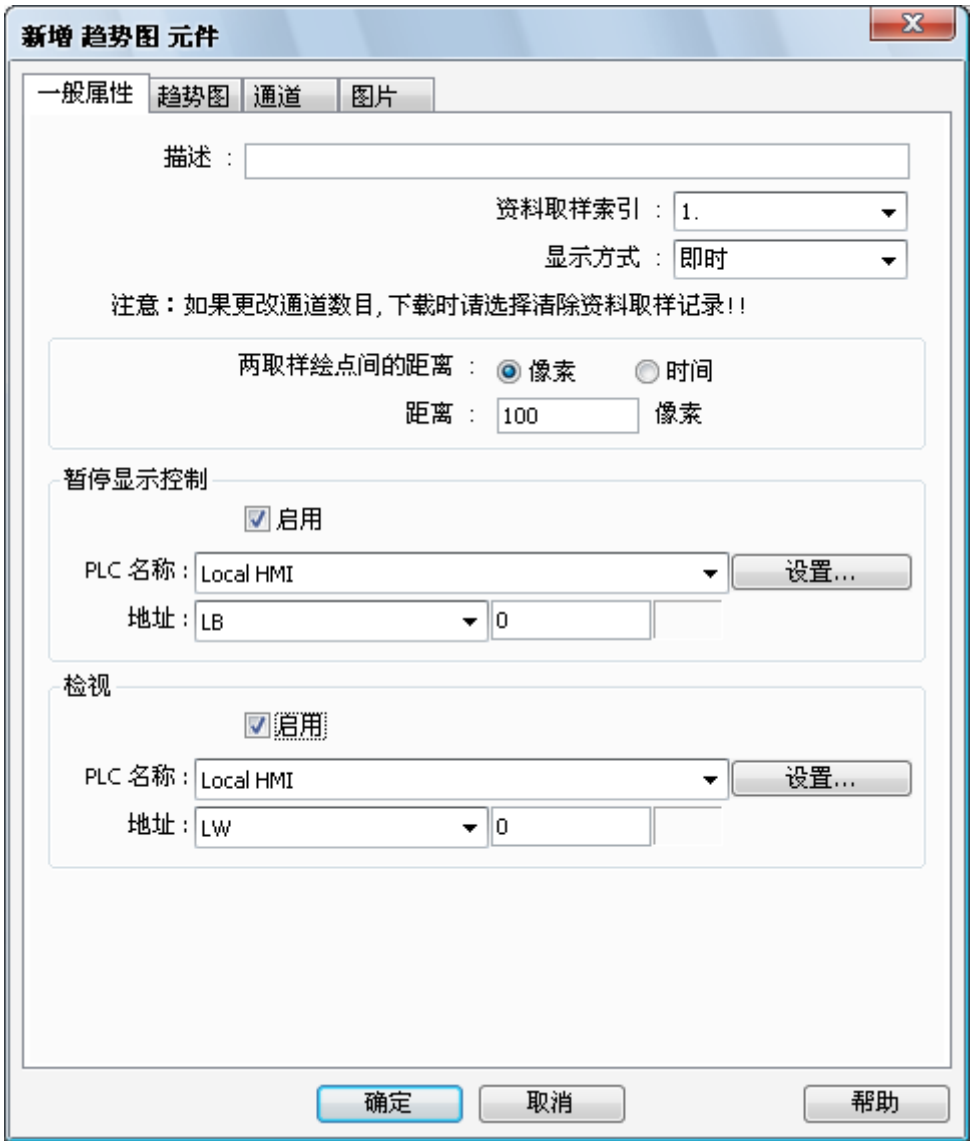
按下工具列上的“趋势图”按钮后即会出现“趋势图”元件属性对话框，正确设定各项属性后按下确认键，即可新增一个“趋势图”物件。

eMT、iE、cMT-HD 系列

一般属性设定



按钮	描述
	显示最初的取样资料。
	显示往前一个间隔的取样资料。
	暂停画面自动卷动功能。当新的取样资料产生时，画面不卷动，也不显示超出画面范围的最新取样资料。
	开启画面自动卷动功能。当新的取样资料产生时，画面会卷动来显示最新的取样数据。
	显示往后一个间隔的取样资料。
	显示最新的取样数据。



设定	描述
资料取样元件索引	选择“资料取样”元件作为绘图所需的数据来源。
资料来源	选择数据来源的形式，可以选择“实时”或“历史”。 **实时** 可显示来自“资料取样”元件从触摸屏开机后，固定笔数的取样数据。取样数据的显示数量于“资料取样”元件的“最大数据 (实时模式)”中设定。当超过此设定的数量，则较旧的数据会从画面上删除。若需显示他日或较旧的资料，需使用“历史”模式。 可以利用“暂停控制”功能暂停元件画面更新的动作，但仅指暂停画面刷新，并不会暂停“资料取样”元件的取样动作。 **历史** 历史记录来自“资料取样”元件使用日期来分类并储存的取样数据。使用“历

史”模式可以利用“资料取样元件索引”选定要显示的历史记录，并利用“历史控制”选择不同日期的历史记录。若检视的内容为当天的历史记录，将会每隔 10 秒钟自动更新检视的内容。

HMI 会将取样数据的历史记录文件依时间先后排序，以日期最新的文件为记录 0 (一般是今日已存盘的取样数据)，日期次新的文件为记录 1，其余记录依此类推。

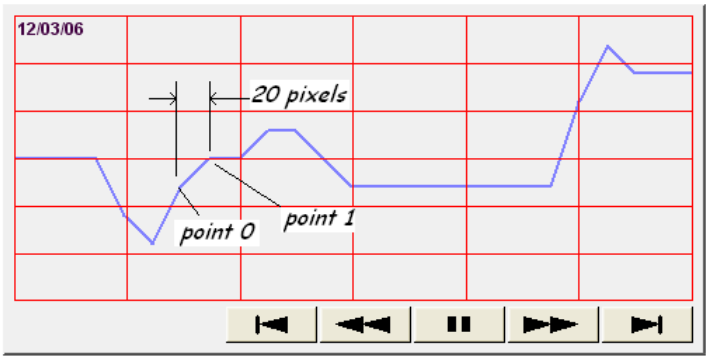
在“历史控制”中所指定寄存器中的数据如果为 0，”趋势图”元件将显示记录 0 的数据；寄存器中的数据如果为 1，将显示记录 1 的数据，也就是说寄存器中的数据如果为 n，将显示记录 n 的数据。

范例：若历史控制寄存器为 LW-0，假使目前的“资料取样”元件已储存的取样数据文件依时间先后分别为 pressure_20140420.dtl、pressure_20140423.dtl、pressure_20140427.dtl、pressure_20140503.dtl，共 4 笔文件，并且今日时间为 2014/5/3，则依照 LW-0 中的数据内容，”趋势图”所显示的取样数据文件整理如下：

LW-0 之数值	显示的历史资料取样文件
0	pressure_20140503.dtl
1	pressure_20140427.dtl
2	pressure_20140423.dtl
3	pressure_20140420.dtl

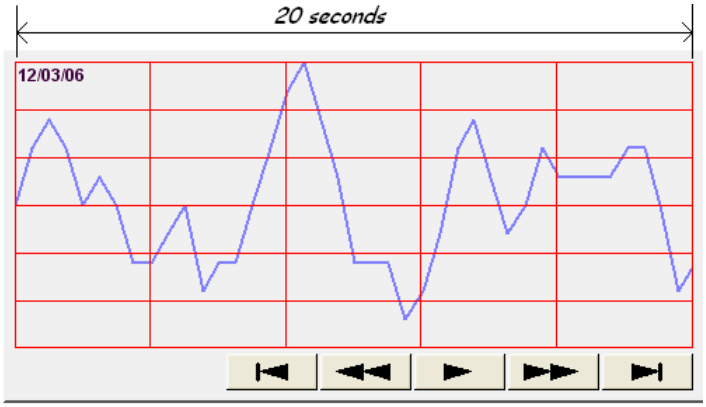
像素

设定两取样绘点间的距离，如下所示。



时间

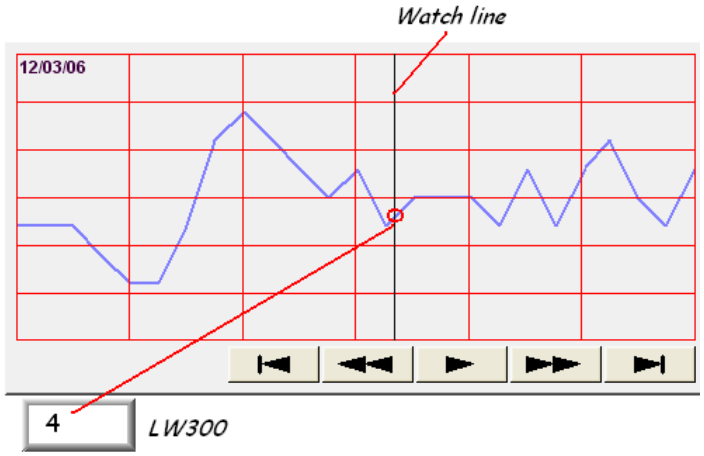
设定 X 轴表示的时间范围，如下所示。



可在“趋势图”页面的“网格”项目启用“时间刻度”功能。

检视

启用后，用户碰触“趋势图”元件会产生一垂直检视线，并将检视线上的取样数据输出到指定的地址，再显示于 “数值显示” 元件，如下所示。



“检视”功能也可以输出多个通道的取样数据，系统会依照“资料取样”元件中所定义的取样数据数据格式，依序将标记所在位置的取样数据，从“检视”功能所定义的起始地址依序写入。例如“资料取样”元件的取样数据包含数个数据，格式皆不同，假设此时 LW-300 为“检视”功能所定义的寄存器，则检视线所标记的取样数据的输出位置如下所示。

寄存器	通道	数据格式
LW-300	0	16-bit Unsigned (1 word)
LW-301	1	32-bit Unsigned (2 words)
LW-303	2	32-bit float (2 words)
LW-305	3	16-bit Signed (1 word)

时间标签输出

若启用，系统将会以第一个取样点的取样时间作为时间原点并开始计数，并将最新取样点之累计秒数输出至“时间标签输出地址 + 2”。

当点选元件上的曲线时，可将触碰处最接近的取样点之累计秒数输出至“时间标签输出地址”。

若触发资料取样元件的“清除控制地址”，除了可以清除目前的取样数据，也

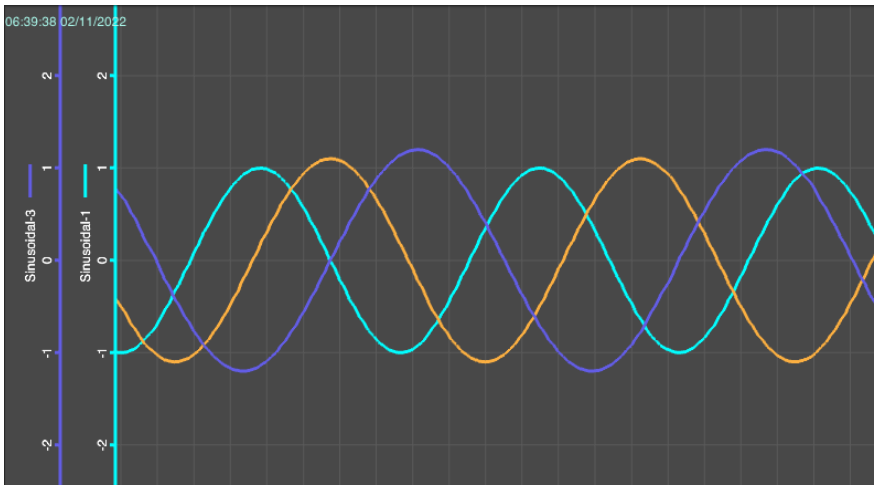
可以重置取样时间原点。

注意:“时间标签输出地址”与“时间标签输出地址 + 2”皆须为 32-bit 格式。”时间标签输出地址 + 2”只适用于实时模式，而“时间标签输出地址”适用于实时模式及历史模式。

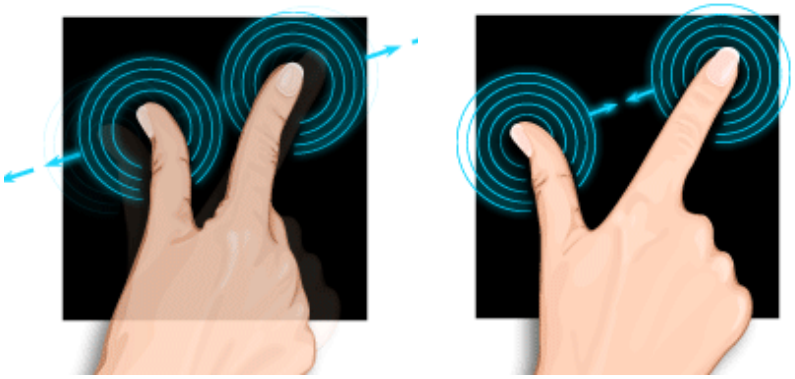
当趋势图页面的“相对时间模式”被勾选时，才可启用此功能。

cMT-SVR 系列

一般属性设定

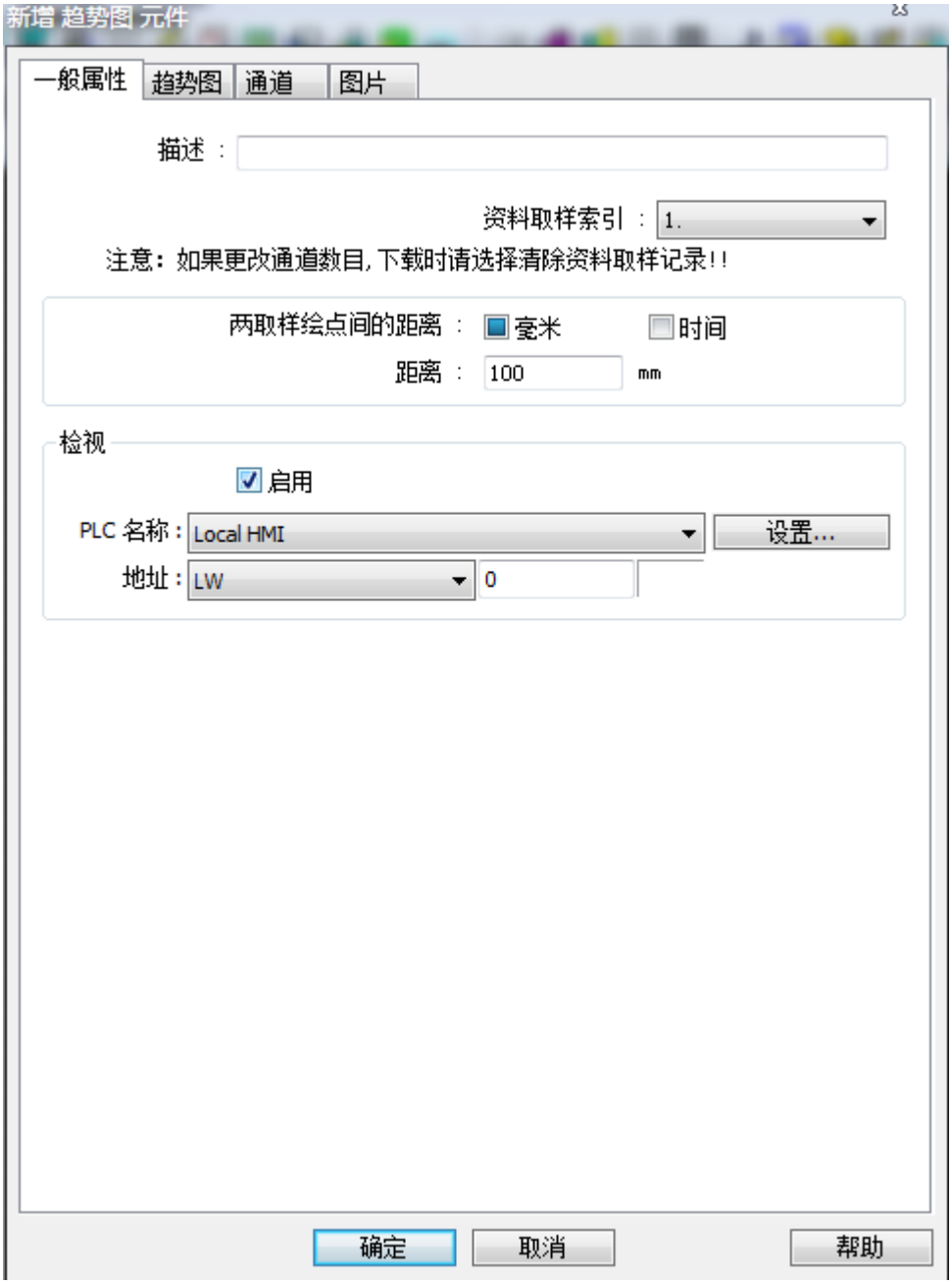


cMT-SVR 系列的趋势图结合了“实时”模式和“历史”模式，在元件上往左侧拖曳即可查看历史数据，往右侧拖曳可查看最新的取样资料，并可用双指拖曳来放大或缩小趋势图。



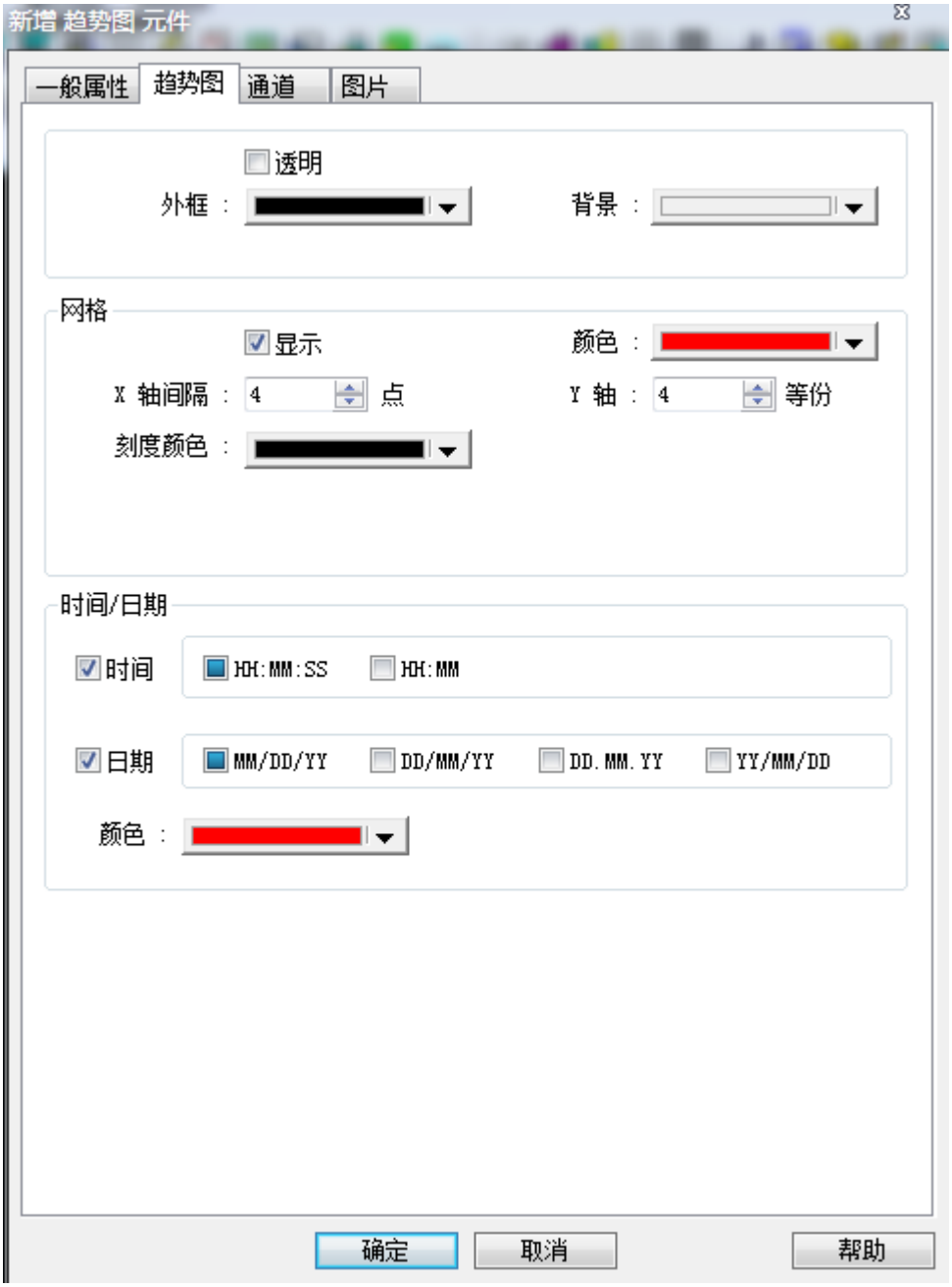
放大趋势图　　缩小趋势图

cMT-SVR 系列资料取样储存机制的详细信息请参考《8 资料取样》。



设定	描述
资料取样元件索引	选择“资料取样”元件作为绘图所需的数据来源。
毫米	与 eMT、iE、cMT-HD 系列相同。
时间	与 eMT、iE、cMT-HD 系列相同。
检视	与 eMT、iE、cMT-HD 系列相同。

趋势图设定



设定	描述
外框 / 背景	设定元件的外框与背景颜色。
使用画面卷动控制按钮	启用 / 取消 画面卷动控制按钮，如下所示。
网格	设定网格线的数目与颜色。会根据“一般属性”设定页的“两取样绘点间的距离”或是“X 轴表示时间范围”设定不同而有差异。系统会利用这些设定，自动计算垂直网格线的数目。

X 轴间隔

设定网格线垂直线的数目。

- 依照“两取样绘点间的距离”：

选择每两条垂直网格线间所包含的取样点数目。

- 依照“X 轴表示时间范围”：

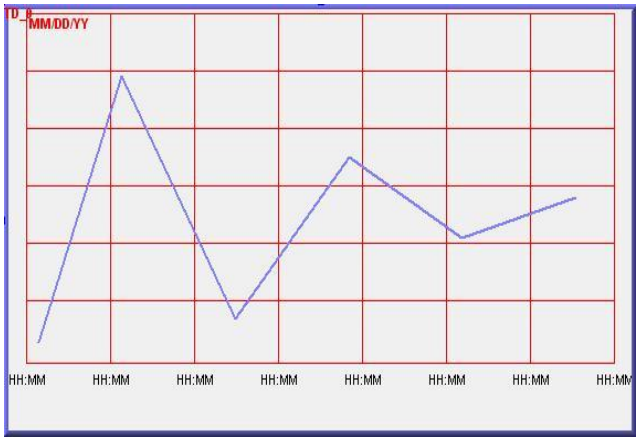
选择每两条垂直网格线间所显示的时间间隔。

Y 轴间隔

设定网格线水平线的数目。

时间刻度

选择“显示”来显示时间刻度于趋势图的 X 轴，如下所示。



格式

选择时间刻度格式为 HH:MM 或 HH:MM:SS。

字型 / 颜色 / 尺寸

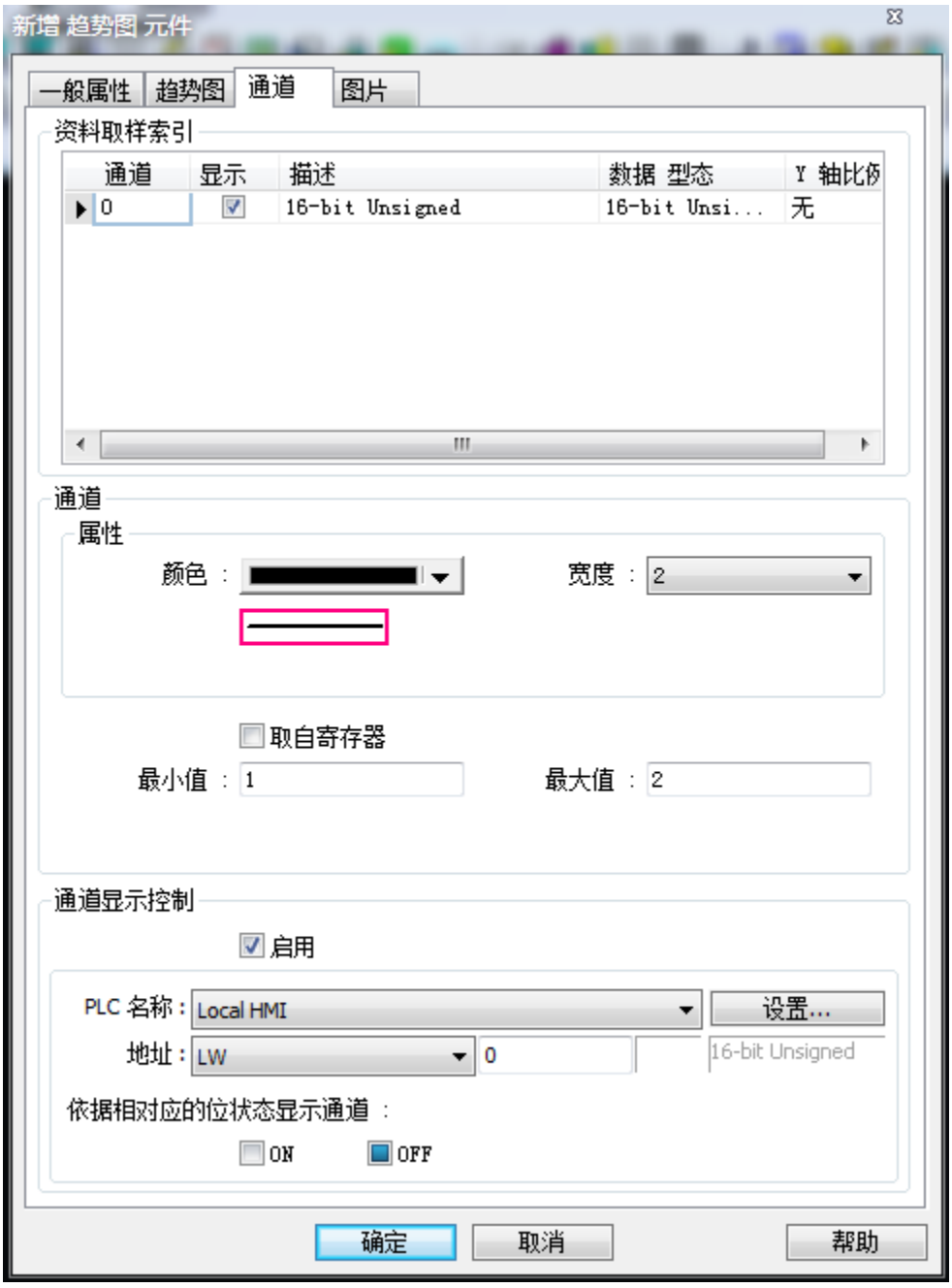
选择时间刻度显示文字之字型、字体颜色、字体尺寸。

字体尺寸的默认值为 8。

时间 / 日期

最新的取样数据的时间信息将标示于元件的左上角，此项目用来设定时间 / 日期的显示格式与颜色。

通道设定



设定	描述
Y 轴比例	适用于 cMT-SVR 系列，请见《13.17.2.3 Y 轴比例说明》。
通道	设定各条曲线的颜色，宽度，与样式。最多可同时支持 64 个通道。 **取自寄存器** ● 不勾选“取自寄存器” “最小值”与“最大值”用来设定各曲线所描绘的取样数据的最小值与最大值。也就是说如果存在某一曲线所描绘的取样数据最小值为 50，最大值为 100，则“最小值”与“最大值”需设定为 50 与 100，如此所有的取样数据才会完

全被描绘在元件中。

- 勾选“取自寄存器”

上下限可由指定寄存器设定。当写入地址为 LW-n，则上/下限会根据以下的规则自动被设定为：

地址格式	16-bit	32-bit
下限	LW-n	LW-n
上限	LW-n+1	LW-n+2

以下表为例，当“寄存器地址”为 LW-100 时，则上/下限的地址会自动被设定为：

地址格式	16-bit	32-bit
下限	LW□100	LW-100
上限	L□101	LW-102

此设定常用于放大与缩小趋势图 (不适用于 cMT-SVR 系列)。请见以下范例 1。

通道显示控制

若勾选“启用”，则此地址中的各个位将会被用来控制各个通道的显示与否。Bit-1 控制通道 1，Bit-2 控制通道 2，依此类推。举例来说，建立 5 个通道，并设定通道控制的地址为 LW-0，则各个通道会被以下地址控制：

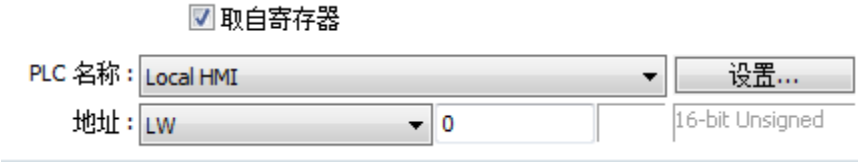
通道	控制地址	位状态	是否显示
1	LW_bit-000	OF	YES
2	LW_bit-001	ON	NO
3	LW_bit-0□2	ON	NO
4	LW_bi□003	OFF	YES
5	LW_bit-004	OFF	YES

若通道选择不被显示，则被取消的通道不会占用到通道控制地址。假设以上表为例：总共有 5 个通道，但第 3 个通道未勾选显示于趋势图上，所以最多会有 4 个通道同时显示于趋势图上，控制通道的地址只会使用 LW_bit-000~003。

范例 1

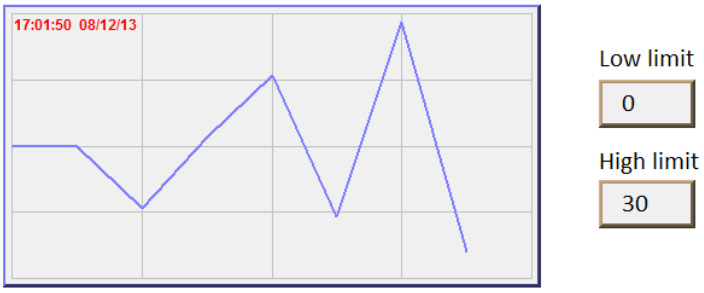
此范例说明如何缩放趋势图。此范例不适用于 cMT-SVR 系列。

用户需在通道的上限/下限设定字段勾选 **“取自寄存器”** 以实现此功能。当最大/最小值取自寄存器设定为 LW-n，则 LW-n 控制最小值，LW-n+1 控制最大值。

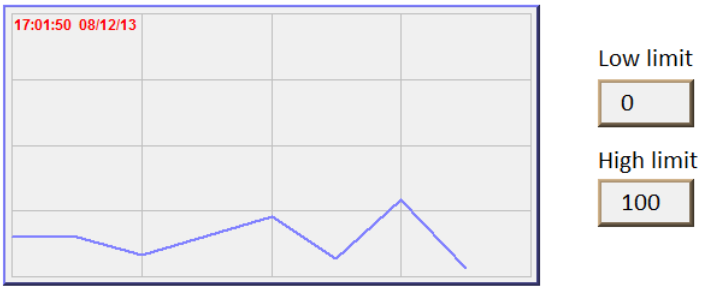


设定最大/最小值取自寄存器 LW-0，建立两个“数值输入”元件控制最小值及最大值，地址分别为 LW-0 和 LW-1。

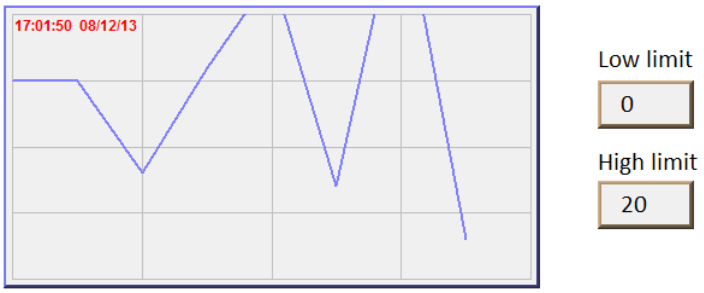
当有一组数据的大小皆介于 0 至 30，则在控制最小值地址输入 0，控制最大值地址输入 30，趋势图呈现如下所示。



若要缩小趋势图，则可以在最大值输入较大的数据。例如：在控制最小值地址输入 0，控制最大值地址输入 100，趋势图呈现如下所示。



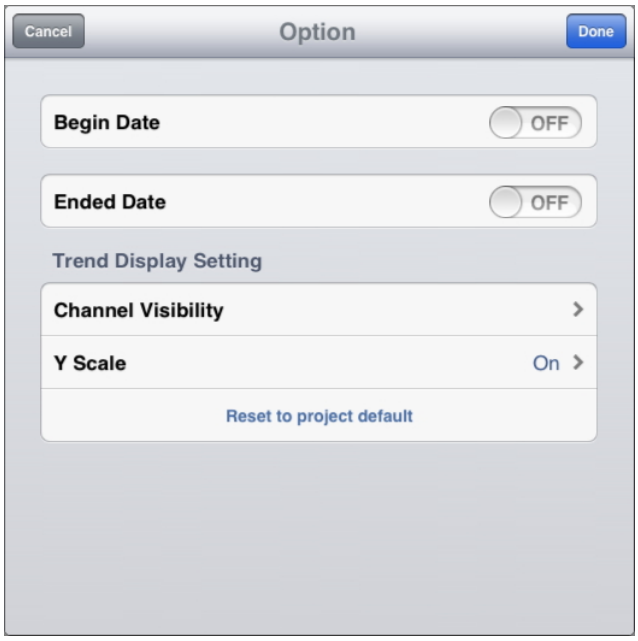
若要放大趋势图，则可以在最大值输入较小的数据。例如：在控制最小值地址输入 0，控制最大值地址输入 20，趋势图呈现如下所示。



Y 轴比例说明

此功能仅适用于 cMT-SVR 系列。元件上的 Y 轴会显示该通道的通道刻度。当“趋势图”设定页的网格选择“显示”时，才可使用此功能。Y 轴比例也可在 iPad 上调整，请参考以下步骤。

1. 点选“趋势图”物件右上方的  按钮。
2. 点选“Trend Display Setting”下的 Y Scale。



3. 选择要显示 Y Scale 的通道。

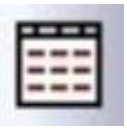
13.18 历史数据显示

13.18.1 概要

“历史数据显示”元件用来显示已经储存的取样数据数据，跟趋势图不同的是“历史数据显示”元件使用表列的方式直接显示这些数据的内容。若检视的内容为当天的历史记录，将会每隔 10 秒钟自动更新检视的内容。历史数据范例表，如下图所示。

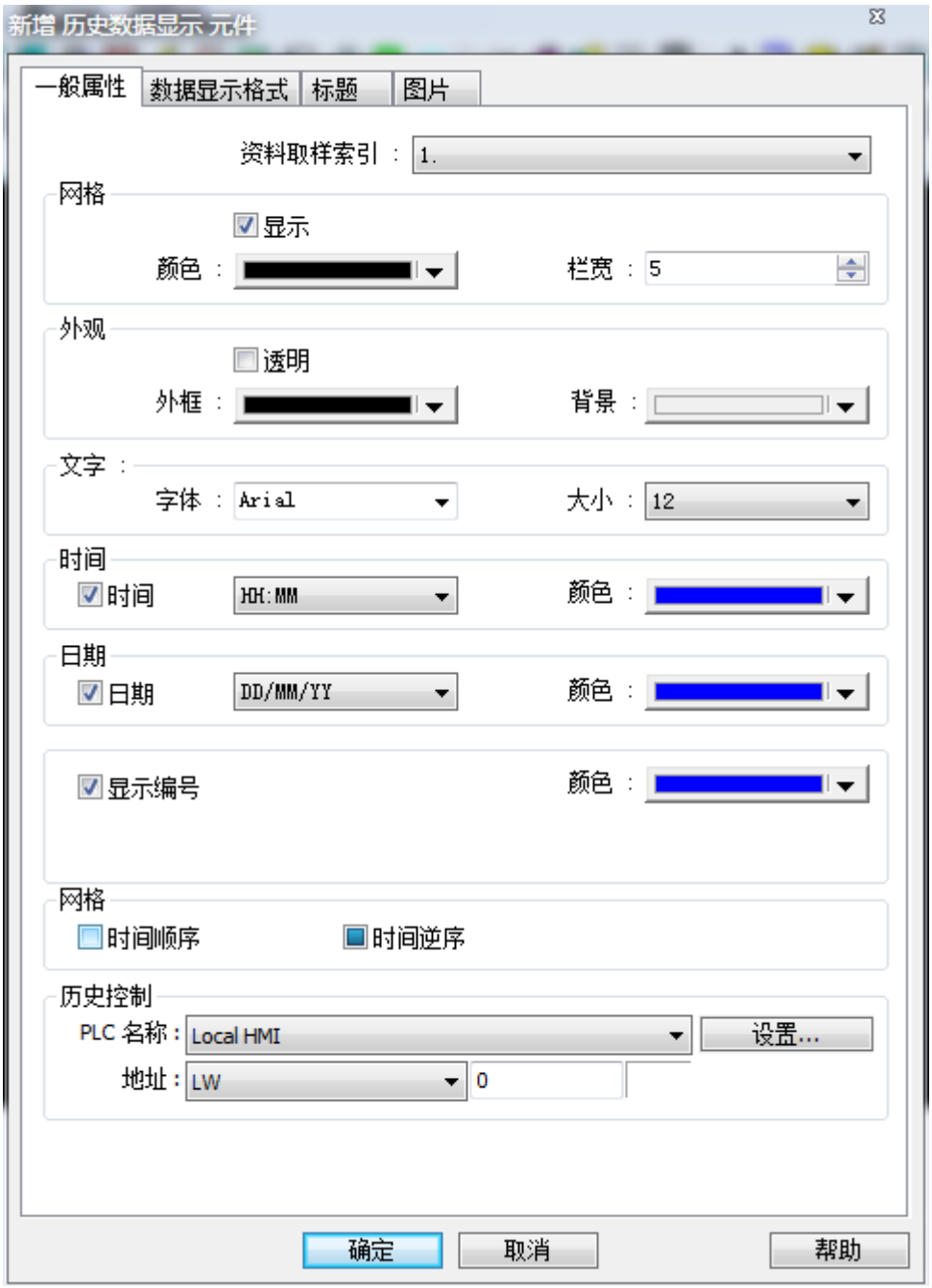
编号	时间	日期	ch.0	ch.1	ch.2
13	15:24	05/05/14	0	0	0
12	15:24	05/05/14	0	0	0
11	15:24	05/05/14	0	0	0
10	15:24	05/05/14	0	0	0
9	15:24	05/05/14	0	0	0
8	15:24	05/05/14	0	0	0
7	15:24	05/05/14	0	0	0
6	15:24	05/05/14	0	0	0
5	15:24	05/05/14	0	0	0
4	15:24	05/05/14	0	0	0
3	15:24	05/05/14	0	0	0
2	15:24	05/05/14	0	0	0
1	15:24	05/05/14	0	0	0

13.18.2 设定

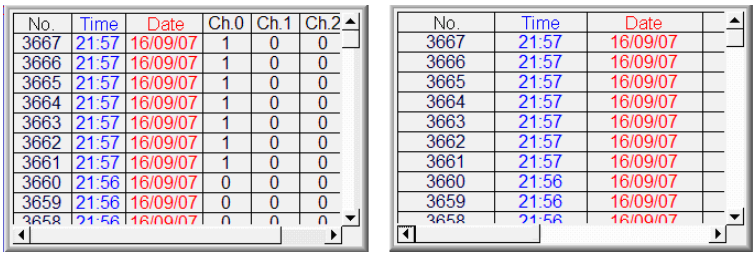


按下工作列上的“历史数据显示”按钮后即会出现“历史数据显示” 元件属性对话框，正确设定各项属性后按下确认键，即可新增一个“历史数据显示”元件。

一般属性设定



设定	描述
资料取样元件索引	选择“资料取样”元件作为所需的数据来源。
网格线	选择元件是否使用网格线区分每个字段。 **颜色** 设定网格线所使用的颜色。 **字段距离** 此项设定值用来调整各字段间的距离，下图为使用不同“字段距离”设定时的显示情形。



No.	Time	Date	Ch.0	Ch.1	Ch.2
3667	21:57	16/09/07	1	0	0
3666	21:57	16/09/07	1	0	0
3665	21:57	16/09/07	1	0	0
3664	21:57	16/09/07	1	0	0
3663	21:57	16/09/07	1	0	0
3662	21:57	16/09/07	1	0	0
3661	21:57	16/09/07	1	0	0
3660	21:56	16/09/07	0	0	0
3659	21:56	16/09/07	0	0	0

No.	Time	Date
3667	21:57	16/09/07
3666	21:57	16/09/07
3665	21:57	16/09/07
3664	21:57	16/09/07
3663	21:57	16/09/07
3662	21:57	16/09/07
3661	21:57	16/09/07
3660	21:56	16/09/07
3659	21:56	16/09/07

外观	设定元件的外框与背景颜色，若勾选“透明”表示不使用外框与背景颜色。
时间 / 日期	用来选择是否显示数据的取样时间与日期，并决定时间与日期的显示格式。 **按时间顺序** 先显示取样时间较早的资料。 **按时间逆序** 先显示取样时间较晚的资料。
历史数据控制 (eMT、iE、cMT-HD 系列)	系统会将取样数据的历史记录文件依时间先后排序，日期最新的文件为记录 0 (一般是今日已存盘的取样数据)，日期次新的文件为记录 1，其余记录依此类推。”历史数据控制”项目则用来指定要显示的记录。

Note

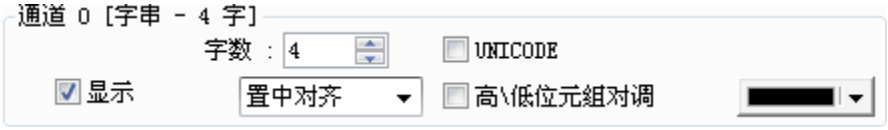
使用 cMT-SVR 系列时，请直接于 iPad 画面上点选历史数据显示元件右上角的漏斗图标来指定日期并显示数据。

数据显示格式设定

设定	描述
通道	最多可同时显示 64 个通道。由此设定页可得知目前选取的“资料取样”元件一次取样的数据长度、各通道的数据格式、显示于此元件上的通道编号。如上图：“资料取样”元件执行一次将存取读取 4 个数据 (通道 0~通道 3)，各通道的数据格式皆为 16-bit Unsigned。由于只勾选通道 0 及通道 3，历史数据显示元件显示的数据方式如下图所示。

编号	时间	日期	ch.0	ch.3
27	15:29	05/05/14	0	0
26	15:29	05/05/14	0	0
25	15:29	05/05/14	0	0
24	15:29	05/05/14	0	0
23	15:29	05/05/14	0	0
22	15:29	05/05/14	0	0
21	15:29	05/05/14	0	0
20	15:29	05/05/14	0	0
19	15:29	05/05/14	0	0
18	15:29	05/05/14	0	0

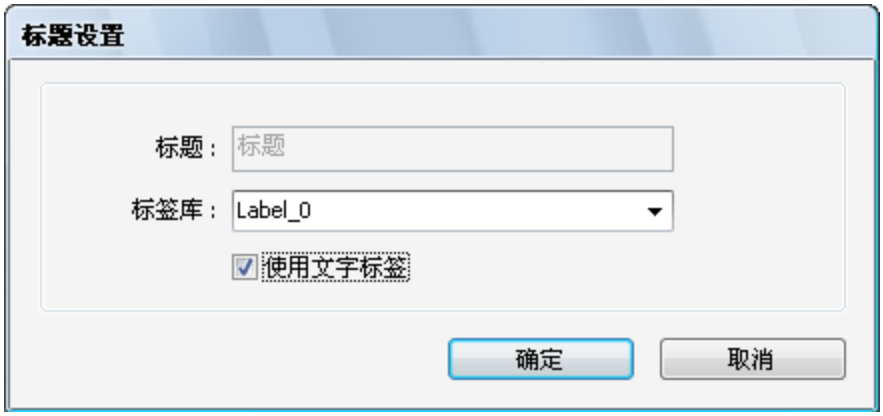
当使用历史数据显示元件显示“字符串”格式时，可以选择：



- 使用“UNICODE”模式显示。
- 将数据的高字节与低字节数据互换后，再加以显示。

标题设定

设定	描述
使用标题	选择是否使用标题。 
标题背景	**透明** 勾选“透明”表示不使用标题文字的背景色。

	颜色 设定标题文字的背景色。
设定	设定标题的文字。 标题也可以使用文字标签库，显示多国语言，只需使用“设置”。在出现设定对话框后，选择使用文字标签库即可。 

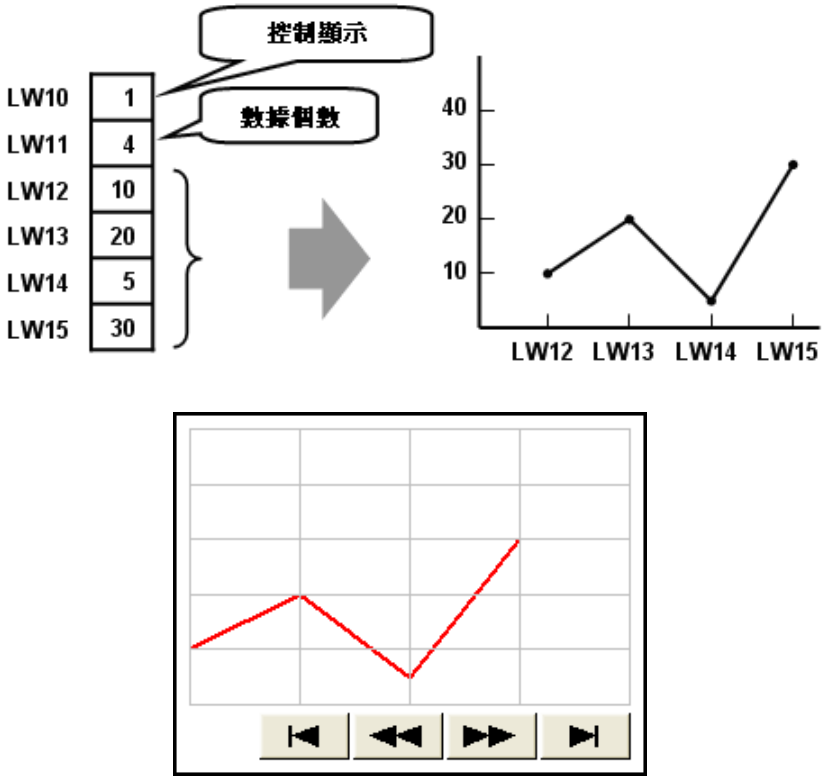
Note

- 当改变过数据格式于资料取样后，在执行离线模式时，不可再更改资料取样，如果需要改变，请将 C:\EasyBuilder \HMI_memory\datalog 内记录历史数据显示的文件删除，再进行离线模拟的动作。

13.19 数据群组显示

13.19.1 概要

一个数据群组(或区块)是指一组连续地址中的数据，X 轴代表地址，Y 轴代表数据，如下图为使用数据群组显示元件显示单一数据群组 LW-12~LW-15 中的数据。数据群组显示元件亦可同时显示多个数据群组的内容，例如同时显示 LW-12~LW-15 与 RW-12~RW-15 两个数据群组，用户可通过此方式来观察及比较各寄存器中的数据。

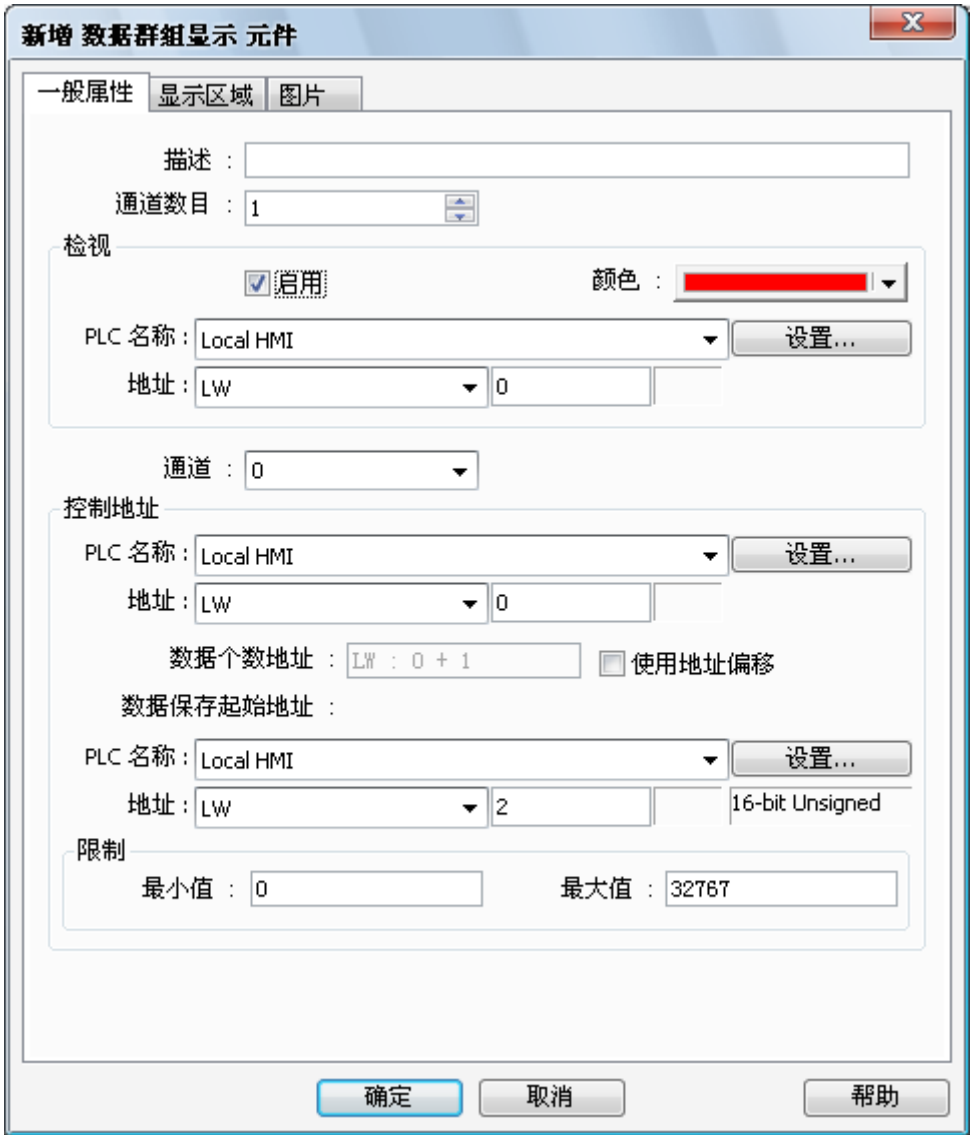


实际执行结果

13.19.2 设定

按下工作列上的“数据群组显示”按钮后即会开启“数据群组显示”元件属性对话框，正确设定各项属性后按下确认键，即可新增一个“数据群组显示”元件。

一般属性设定



设定	描述
描述	用户可为此元件描述相关讯息。
通道数目	设定元件的通道数目。每个通道表示一组群组数据。最多可同时支持 12 组。
检视	若启用，可显示垂直检视线上的数据索引和数值，请见以下范例 1。
通道	选择一个通道来设定控制地址的相关数据群组属性。
控制地址	选择数据群组的控制地址及数据来源。 当控制地址设定为 LW-n 时，输入特定数值到 LW-n 来控制图形的显示及清除。当执行完命令后，系统会将控制地址重设为 0。 输入 “0”：无动作 (默认值) 输入 “1”：绘图 输入 “2”：清除 输入 “3”：重新绘图

	数据个数地址 当控制地址设定为 LW-n 时，写入特定数值至 LW-n+1 做为存放群组的数据数量，最多支持 1024 个。 **数据储存起始地址** 若启用“使用地址偏移”，”数据储存偏移地址”定义为 LW-n+2。 若选择 16-bit 格式，每个起始数据的地址间隔为 1。 例如：起始地址 + 1，起始地址 + 2，等等。 若选择 32-bit 格式，每个起始数据的地址间隔为 2。 例如：起始地址 + 2，起始地址 + 4，等等。 关于控制地址各项设定，请见以下范例 2~5。
限制	用来设定所显示图形之最大值与最小值。

Note

- 重复绘图上限次数为 32 除以通道数。

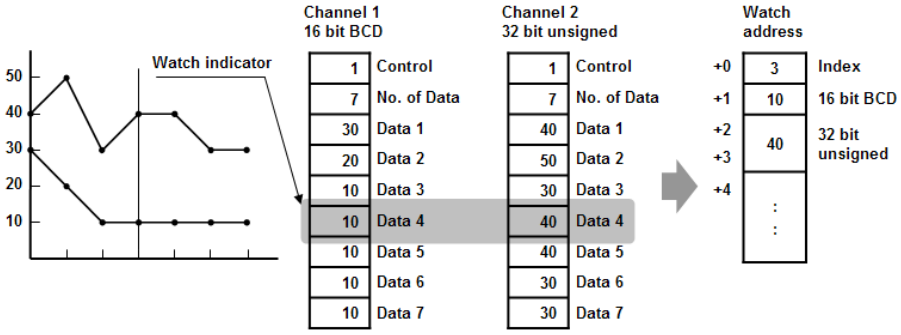
范例 1

数据检视功能

启用 “检视” 功能，则当触控此元件上任一点时，会显示一条垂直的检视线，曲线与该检视线交会形成的点所相对应的数值将会写入至指定的寄存器中。

数据格式	数据索引	通道 1 数值	通道 2 数值
16-bit	地址	地址 + 1	地址 + 2
32-bit	地址	地址 + 2	地址 + 4

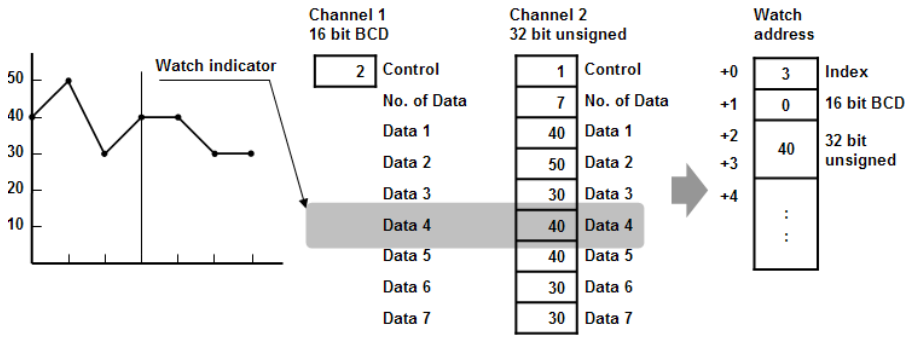
若设定检视地址为 LW-n，则在 LW-n 写入数值代表从各通道欲呼叫的索引编号 (从 0 计算)，如下图：



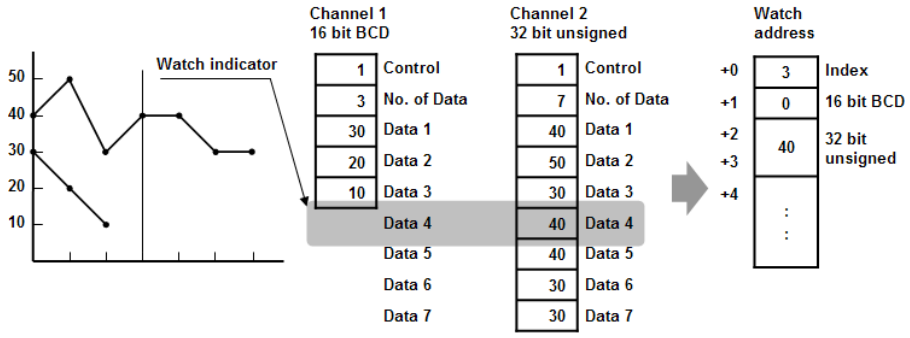
Note

- 索引编号为 16 位无符号数整数。若指定之寄存器为 32 位时，只有较低的 16 位可作用，并忽略使用较高的 16 位。
- 当检视的通道无数据时，则会以 0 代替。

  EX: 仅有通道二的数据，通道一无数据，LW-n+1 显示为 0。



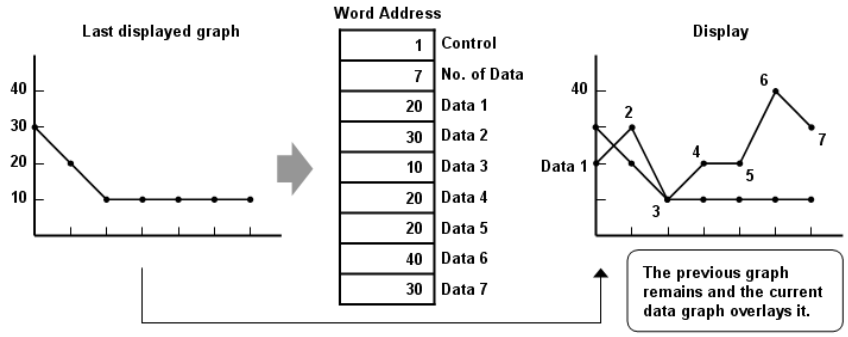
EX: 仅有通道二的数据，通道一无数据，LW-n+1 显示为 0。



范例 2

如何显示数据群组的内容

1. 在“数据个数地址”输入欲显示的数据笔数，也就是“控制地址 +1”。
2. 在“数据储存起始地址”依序填入数据内容。
3. 在“控制地址”输入”1”；此时 HMI 将以折线图画出目前寄存器的内容 (并保留先前图形)。
4. HMI 在完成前项动作后将对”控制地址” 写入”0”。



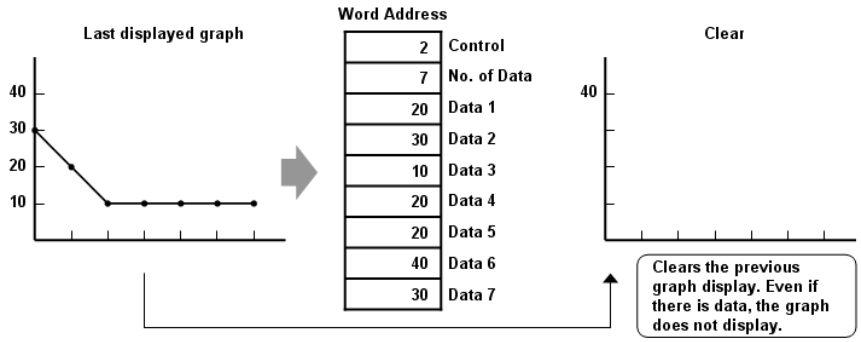
Note

- 在上述动作 3 和 4 之间，请勿更改“控制地址”、”数据个数地址” 及“数据储存起始地址” 内容，否则可能产生非预期结果。

范例 3

如何清除已显示的图形

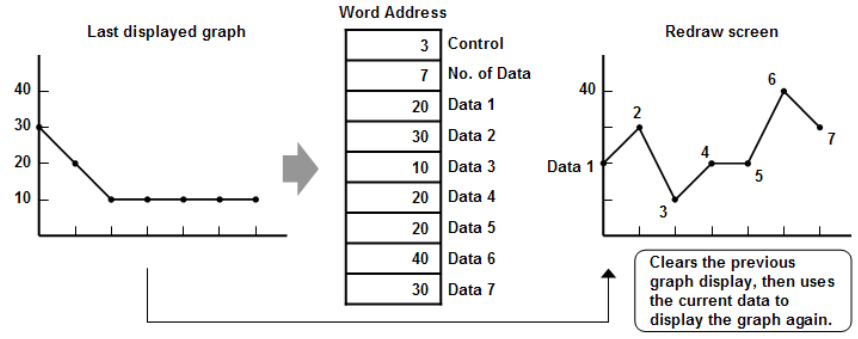
1. 在“控制地址”输入”2”；将清除先前所画之线图。
2. HMI 在完成前项动作后将于“控制地址”写入”0”。



范例 4

清除已显示的图形并显示新数据的图形

1. 在“数据个数地址”输入欲显示的数据笔数，也就是“控制地址 +1”。
2. 在“数据储存起始地址”依序填入数据内容。
3. 在“控制地址”输入”3”；此时 HMI 会先将先前的折线图清除，再画出目前地址内的内容。
4. HMI 在完成前项动作后将于“控制地址”写入”0”。



范例 5

地址偏移模式

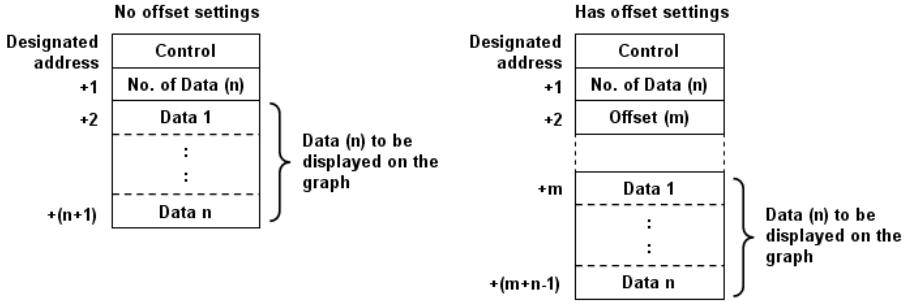
启用后，则各个通道的“控制地址”、”数据个数地址”、”数据储存偏移地址”会使用连续的地址。

例如有 3 个通道(通道 0 至通道 2)，且“控制地址”分别为 LW-0、LW-100 和 LW-200，则各个通道的“控制地址”、”数据个数地址”、”数据储存偏移地址”如下：(下表使用 3 个通道数，格式皆为 16-bit Unsigned 且数据储存偏移地址内的数值设为 m)

项目	通道 0	通道 1	通道 2

控制地址	LW-0	LW-100	LW-200
数据个数地址	LW-1	LW-101	LW-201
数据储存偏移地址	LW-2 (=m)	LW-102 (=m)	LW-202 (=m)
资料 1	LW-0+m	LW-100+m	LW-200+m
资料 2	LW-1+m	LW-101+m	LW-201+m
…	…	…	…

下图左侧代表未使用“偏移模式”的读取方式，右侧则是使用地址偏移模式的读取方式。

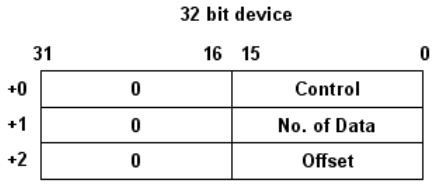


Note

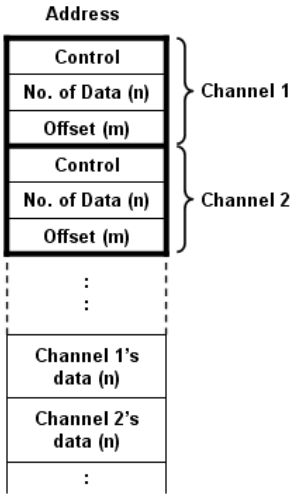
- 当“控制地址”设定为 LW-n 时，”数据个数地址”及“数据储存偏移地址”会根据以下规则设定：

数据格式	16-bit	32-bit
控制地址	LW-n	LW-n
数据个数地址	LW-n+1	LW-n+2
数据储存偏移地址	LW-n+2	LW-n+4

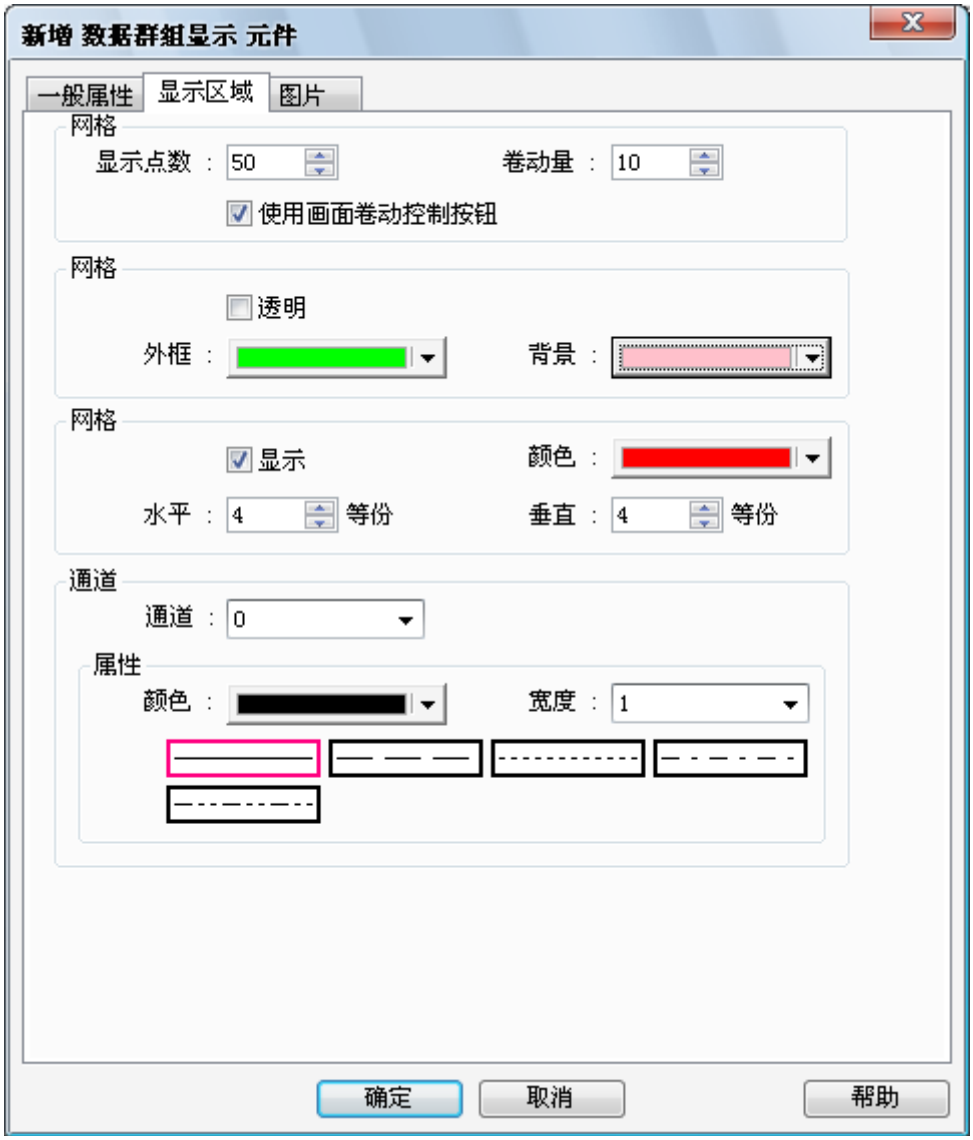
- 当“控制地址”为 32 位时，只有较低的 16 位产生作用，请将较高的 16 位内容设为 0。



- 系统会在“控制地址”的数据不为 0 时读取“数据个数地址”及“数据储存偏移地址”的内容。
- 当使用了两个同类型寄存器以上的通道，则启用“使用地址偏移”并使用连续的地址做控制地址可减少系统读取数据的时间。如下图。当使用 16 位格式时，设定通道 1 的控制地址为 LW-n，通道 2 的控制地址为 LW-n+3，依此类推。


Address
Control
No. of Data (n)
Offset (m)
Channel 1
Control
No. of Data (n)
Offset (m)
Channel 2
Channel 1's data (n)
Channel 2's data (n)

显示区域设定


新增 数据群组显示 元件
一般属性
显示区域
图片
网格
显示点数 : 50
卷动量 : 10
使用画面卷动控制按钮
网格
透明
外框 :
背景 :
网格
显示
颜色 :
水平 : 4 等份
垂直 : 4 等份
通道
通道 : 0
属性
颜色 :
宽度 : 1
确定
取消
帮助

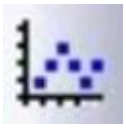
设定	描述
网格线	**显示点数** 设定图形一页所能显示最大数据笔数。 **卷动量** 左右卷动的数据笔数。 **使用画面卷动控制按钮** 按下  后，画面将显示往前或往后一个点的资料。 按下  后，画面将显示最初或最后的资料。
外观	元件的边缘线颜色及背景颜色。 **透明** 若勾选透明，则元件就不会有背景颜色，也不会出现“颜色”的选项。
网格	元件上分隔水平及垂直区块的网格线。
通道	设定各数据群组图形之线条颜色、粗细及样式。

13.20 XY 曲线图

13.20.1 概要

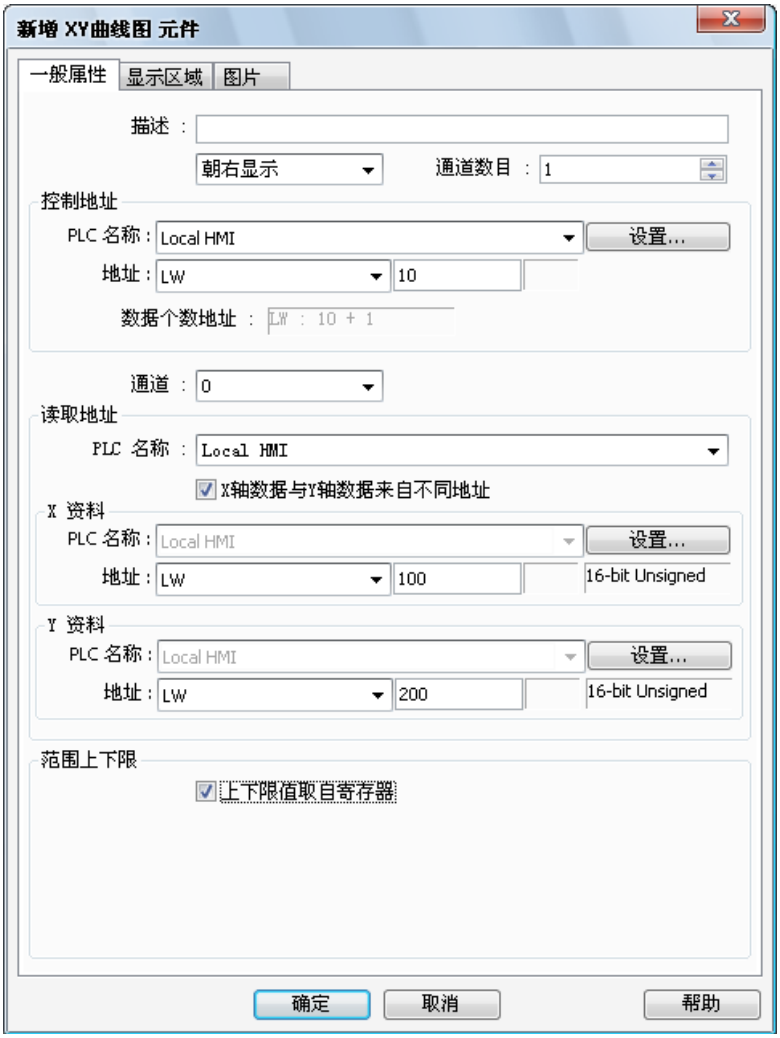
“XY 曲线图”元件用来显示二维坐标的 XY 数据点，每个数据包含 X 值和 Y 值，皆从寄存器中读取。同时可显示最多 16 组曲线。此功能可让用户观察及分析各寄存器中的数据。负数亦可使用。

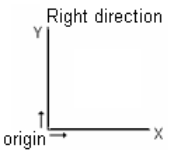
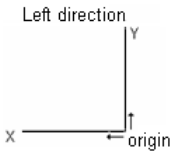
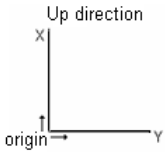
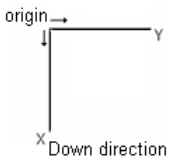
13.20.2 设定



按下工作列上的“XY 曲线图”按钮，随即出现元件属性对话框。

一般属性设定



设定	描述
方向	XY 轴方向可选择“朝右”、”朝左”、”朝上”或“朝下”显示，如下图： 右:  左:  上:  下: 
通道数目	用来设定欲观察的通道数据笔数。
控制地址	用来控制 XY 曲线图的显示或清除，当控制地址设定为 LW-n，则对 LW-n 写入不同的数值代表不同的命令。同时，LW-n+1 会被用来调整显示的数据个数。当 HMI 完成指定的动作后会将“控制地址”之值设为 0。 **数据个数地址** 此地址是用来储存数据显示的数量。每个通道可以有高达 1023 个 XY 数据。
通道	指定一个通道并设定读取的相关属性。
读取地址	**PLC 名称** 选择读取的数据来源装置。 存取寄存器数据时，须同时考虑是否启用 **“X 轴数据和 Y 轴数据来自不同地址”**和 **“上下限值取自寄存器”**。请见以下范例 1。
范围上下限	● 未勾选时： 上限 / 下限为常数。上下限是用于计算 X，Y 轴的刻度百分比，请见以下范例 2。 ● 勾选时： 用户可改变上下限来达到缩放效果。请见以下范例 3。

控制地址	数值	结果
LW-	1	显示目前图形 (保留已绘制的图形)
	2	清除图形
	3	清除所有图形，并重新绘制图形
LW-n+1	1-1023	显示的数据个数

范例 1

存取寄存器数据时，须同时考虑是否启用 **“X 轴数据和 Y 轴数据来自不同地址”** 和 **“上下限值取自寄存器”**。以下以实例说明各情况 (假设皆使用 16-bit 寄存器)：

- 假设**停用** “X 轴数据和 Y 轴数据来自不同地址”，当 “读取地址” 设为 LW-0 时：

	启用 “上下限值取自寄存器”		**停用** “上下限值取自寄存器”	
	X 资料	**Y 资料**	**X 资料**	**Y 资料**
下限	LW-n	LW-n+2	常数	常数
上限	LW-n+1	LW-n+3	常数	常数
第一笔数据	LW-n+4	LW-n+5	LW-n+0	LW-n+1
第二笔数据	LW-n+6	LW-n+7	LW-n+2	LW-n+3
第三笔数据	LW-n+8	LW-n+9	LW-n+4	LW-n+5
第四笔数据	LW-n+10	LW-n+11	LW-n+6	LW-n+7

- 假设**启用** “X 轴数据和 Y 轴数据来自不同地址”，当 “X 数据” 为 LW-m，”Y 数据” 为 LW-n：

	启用 “上下限值取自寄存器”		**停用** “上下限值取自寄存器”	
	X 资料	**Y 资料**	**X 资料**	**Y 资料**
下限	LW-m+0	LW-n+0	常数	常数
上限	LW-m+1	LW-n+1	常数	常数
第一笔数据	LW-m+2	LW-n+2	LW-m+0	LW-n+0
第二笔数据	LW-m+3	LW-n+3	LW-m+1	LW-n+1
第三笔数据	LW-m+4	LW-n+4	LW-m+2	LW-n+2
第四笔数据	LW-m+5	LW-n+5	LW-m+3	LW-n+3

范例 2

当 **“上下限值取自寄存器”** 未勾选时，上限 / 下限为常数。上下限用于计算 X，Y 轴的刻度百分比：

$$\text{刻度百分比 (\%)} = \frac{\text{寄存器数据-下限}}{\text{上限 - 下限}}$$

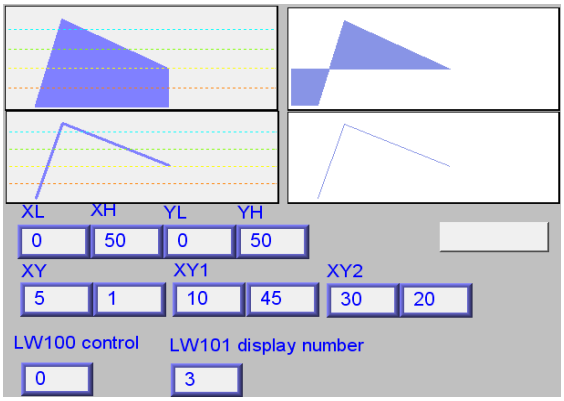
假设寄存器为 LW- n 且**停用** “X 轴数据和 Y 轴数据来自不同地址”，则上限 / 下限的数据来源会根据以下方式设定：

数据格式	16-bit	32-bit
X 轴下限	LW-n	LW-n
X 轴上限	LW-n+1	LW-n+2
Y 轴下限	LW-n+2	LW-n+4
Y 轴上限	LW-n+3	LW-n+6

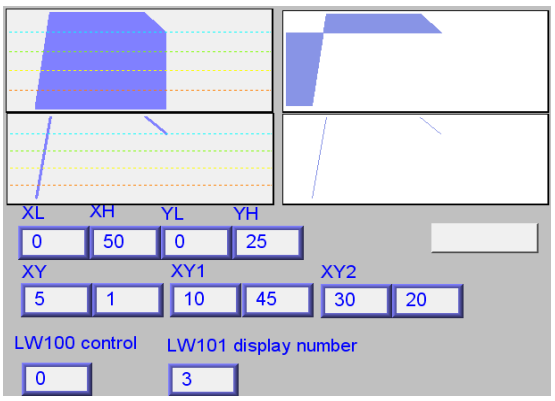
范例 3

若勾选 **“上下限值取自寄存器”**，用户可改变上下限来达到缩放效果。

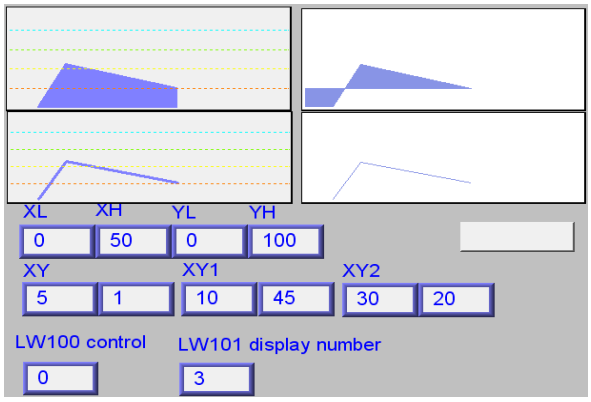
如下范例，XL=X 下限，XH=X 上限，YL=Y 下限，YH=Y 上限，XY，XY1，XY2 为三个 XY 数据。此时改变 Y 轴的上限，即可观察缩放效果。效果如下：



原图



改变 Y 轴上限为 25 (放大效果)



改变 Y 轴上限为 100 (缩小效果)

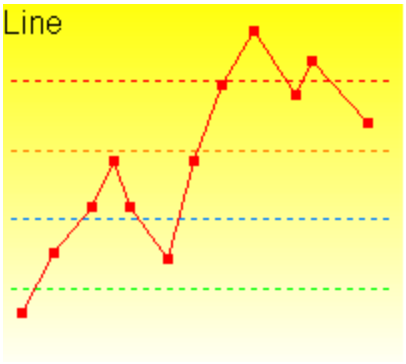
详细信息请参考《13.17 趋势图》。

Note

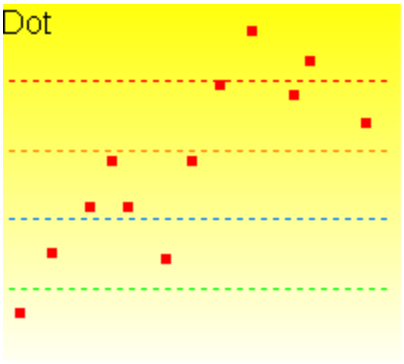
- cMT-SVR 系列可直接用手指拖曳画面进行图形的缩放，因此不适用此方法。
- X 和 Y 数据可使用不同格式，例如 X 数据使用 16-bit unsigned 而 Y 数据使用 32-bit signed，此时需特别留意地址的设置。
- 当 PLC 是 Tag PLC 时，例如 AB tag PLC，则 X 和 Y 一定要使用相同的地址格式。若选择不同的格式会出现警示讯息。

显示区域设定

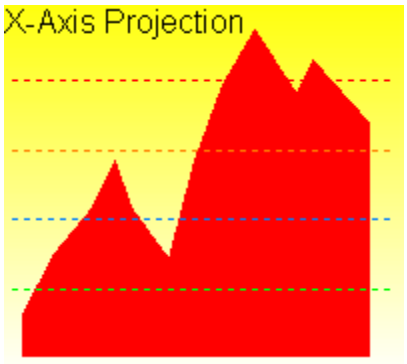
设定	描述
外观	勾选“透明”时背景为透明，无勾选则依照所选择的色彩来表现外框及背景。
曲线	可在此设定通道所要显示的属性。
样式	设定屏幕以线，点，对 X 轴投影或对 Y 轴投影显示。 其对应的示意图如下：



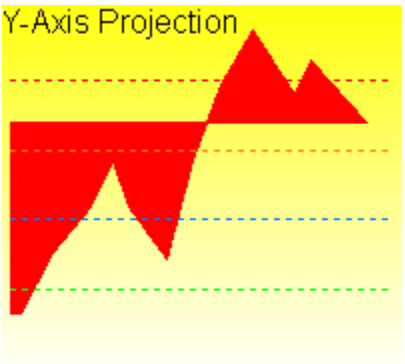
线



点



对 X 轴投影



对 Y 轴 影

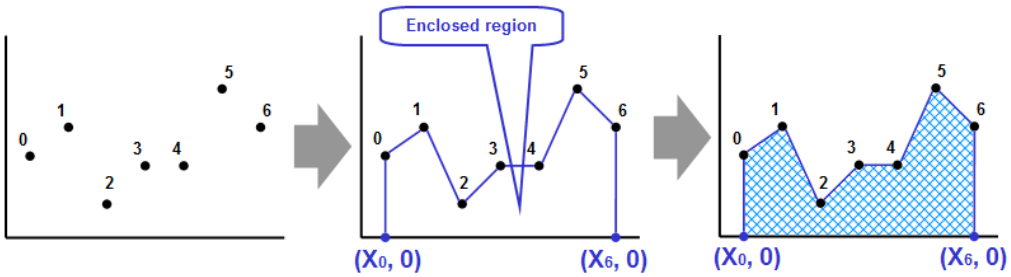
请见以下范例 4。

参考线	最多可画四条参考线在曲线图上，用户可以自行选择线条的色彩及参考的数值，并且依据所设定数值来显示在屏幕上。若勾选上“下限值取自寄存器”，则需设定一个参考线之读取地址。

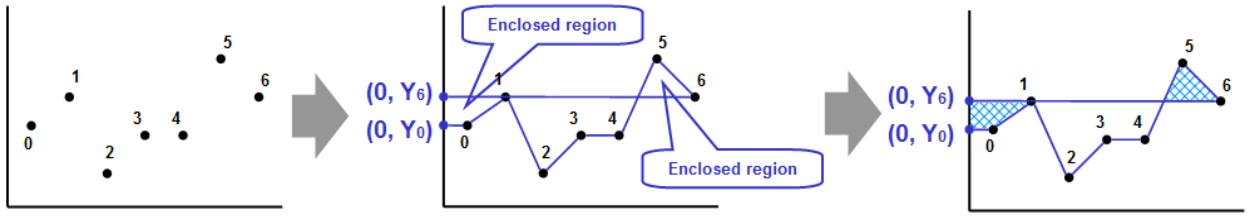
范例 4

下图中的曲线由 7 个点构成，由 P0 到 P6。系统绘制“对 X 轴投影”方式如以下步骤：

1. 自动计算出二个投影的点： X 轴 – (X_0, 0) 和 (X_6, 0)
2. 依照点出现的顺序，连结所有的点： (X_0, 0)，P0… P6，(X_6, 0) 并且最后连结到第一个点 (X_0, 0)
3. 填满封闭区域，结果如下：



同样的“对 Y 轴投影”可得：



Note

- XY Plot 最多可以重复画 32 次。计算方式如下：
  1 个通道可以重复画 32 次，若是有 2 个通道，则只能重复画 16 次。用 32 除以通道数可得最多重复画的次数。

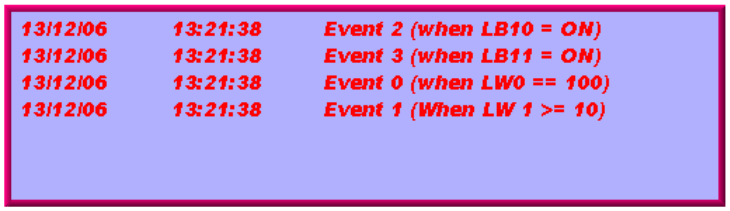
13.21 报警条与报警显示

13.21.1 概要

“报警条”与“报警显示”元件可以用来显示已被定义在“事件登录”中，且系统目前状态满足触发条件的事件，此时这些事件也被称为警示。”报警条”与“报警显示”元件将利用事件被触发的时间先后，依序显示这些警示，其中“报警条”元件使用单行跑马灯型式呈现警示内容；”报警显示”元件则可同时显示多行警示内容。下图显示不同元件对警示的表示方式。

有关事件登录详细信息请参考《7 事件登录》。

“报警条”元件，单行显示多个事件



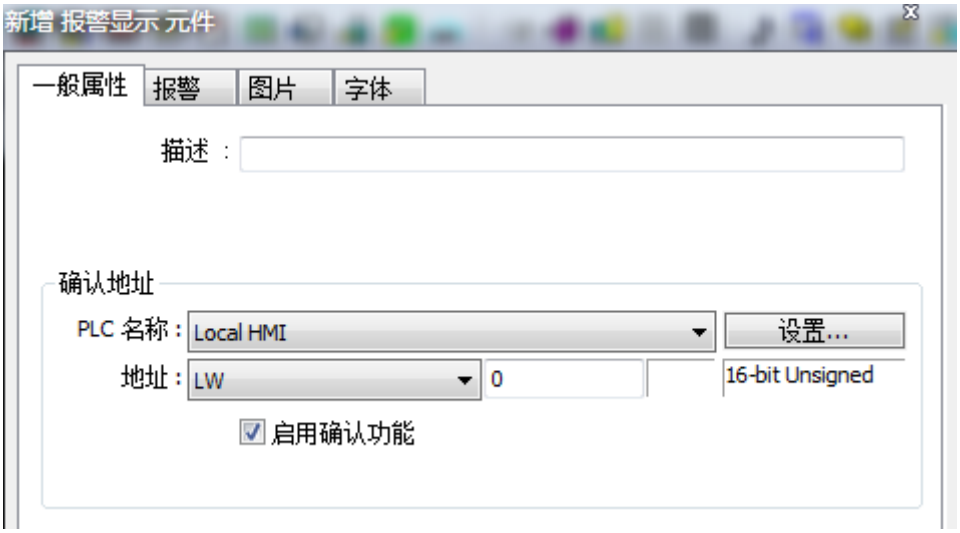
“报警显示”元件，可显示多行

13.21.2 设定

按下工具列上的“报警条”按钮后，即会出现元件属性对话框；相同方式，按下工具列上的“报警显示”按钮后，即会出现元件属性对话框，设定各项属性后按下确定键，即可新增一个元件。

一般属性设定

“报警条”和“报警显示”的设定唯一不同的是“报警显示”可以设定“确认地址”，如下图：



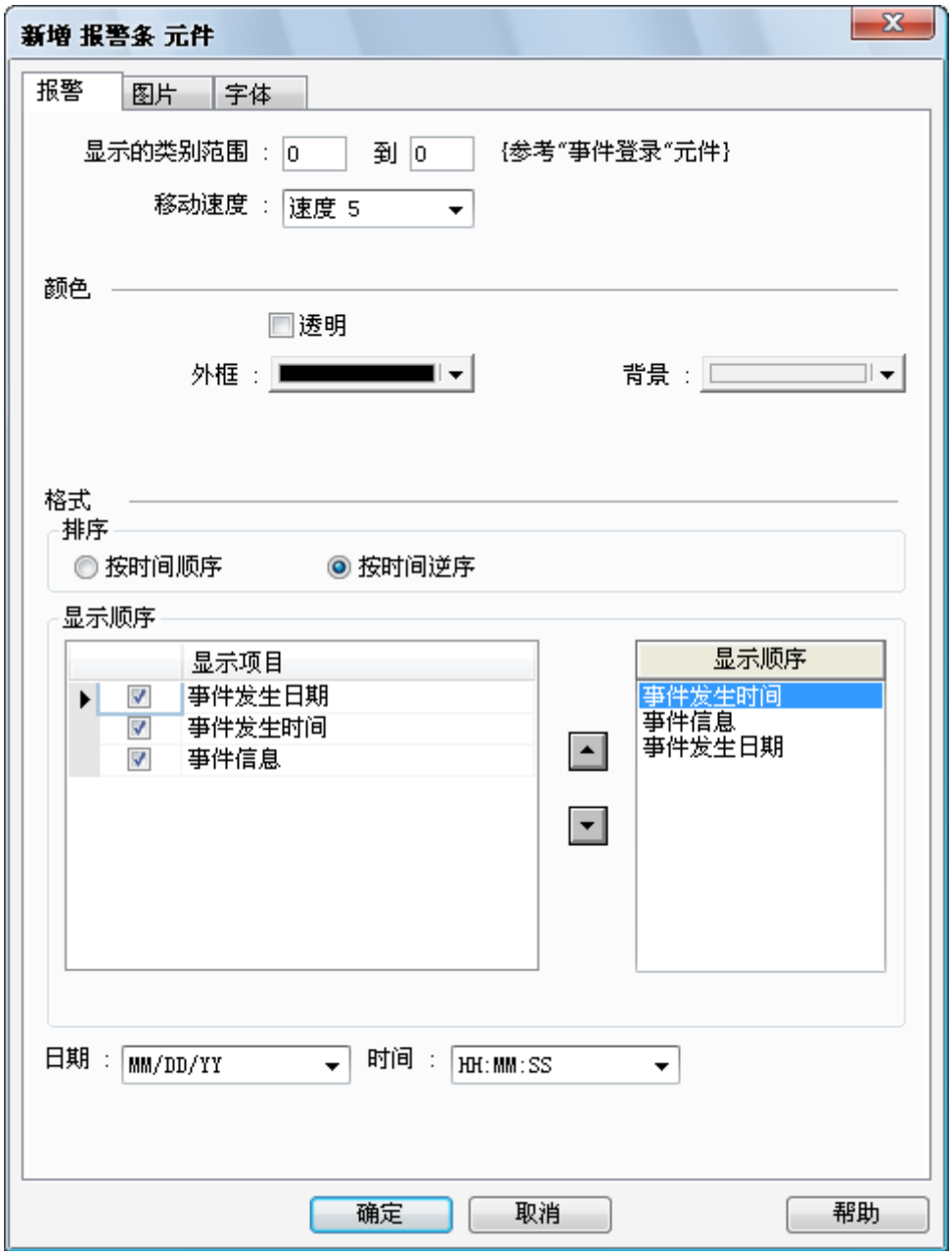
若点选“启用确认功能”将会把在“事件登录”中所指定的“事件确认时写入报警显示/事件显示元件”之值写入“确认地址”。

Note

使用 cMT-SVR 系列时，当手指触压屏幕且不移动位置时会被视为确认当笔事件。若手指触压屏幕且滑动则视为拖曳滚动条。

以下是“报警条”和“报警显示”之共同设定：

报警设定



设定	描述
显示的类别范围	被触发事件的“类别”需符合此处设定的显示范围才会被显示(事件的类别在“事件登录”中设定)。例如当“报警条”元件的“类别”被设定为 2~4，则仅有“类别”为 2、3 或 4 的事件才会被显示在“报警条”元件中。详细说明请参考《7 事件登录》。。
移动速度	仅“报警条”可设定。设定“报警条”元件中所显示文字的移动速度。
格式	**按时间顺序** 较晚发生的警示被排列在后(或在下)。

按时间逆序

较晚发生的警示被排列在前(或在上)。

显示顺序

用户可勾选想要显示的信息以及设定显示顺序。

日期

选择显示事件发生日期的格式，共有以下 4 种模式。

MM/DD/YY、DD/MM/YY、DD.MM.YY、YY/MM/DD

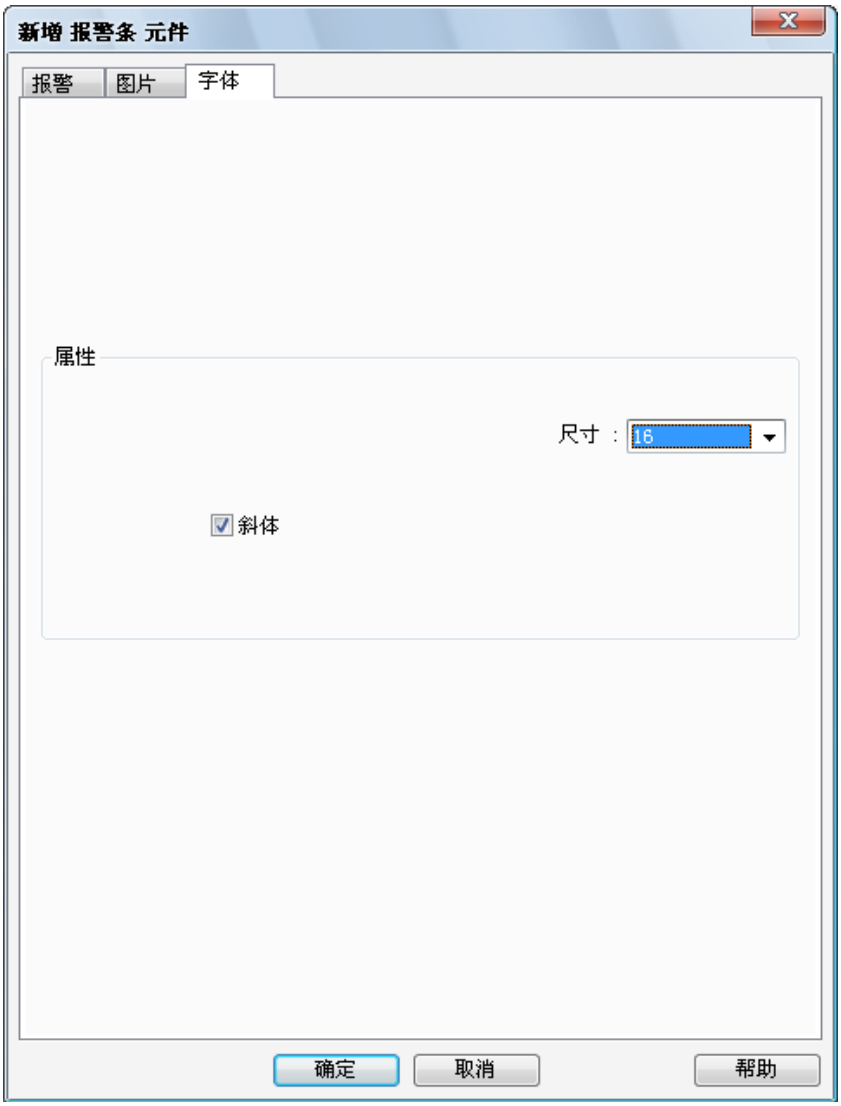
时间

选择显示事件发生时间的格式，共有以下 4 种模式。

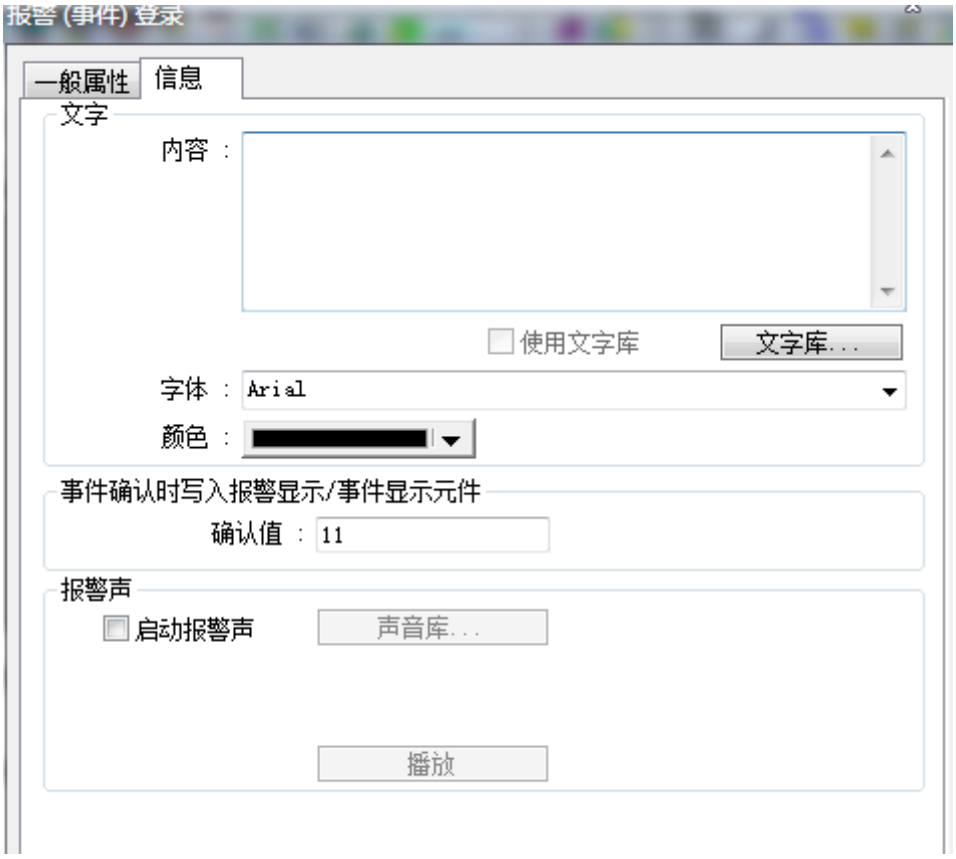
HH:MM:SS、HH:MM、DD:HH:MM、HH

字型设定

设定元件文字之尺寸与斜体效果。



而“报警条”和“报警显示”中事件所显示的讯息内容、字型与颜色是根据“事件登录”中的设定，如下图：

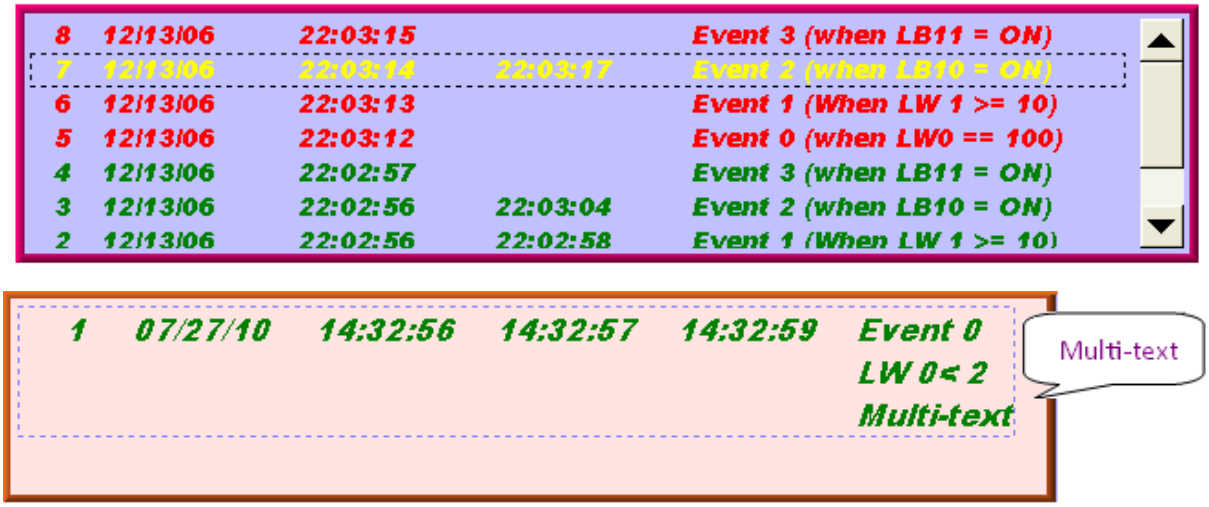


13.22 事件显示

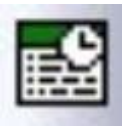
13.22.1 概要

“事件显示”元件可以用来显示已被定义在“事件登录”中，且曾经满足触发条件的事件。”事件显示”元件将根据事件被触发的时间顺序，依序显示这些事件。

“事件显示”元件可以显示事件发生日期、事件发生时间、事件确认时间、恢复正常时间、以及事件讯息内容。讯息内容可以用多行的方式显示。



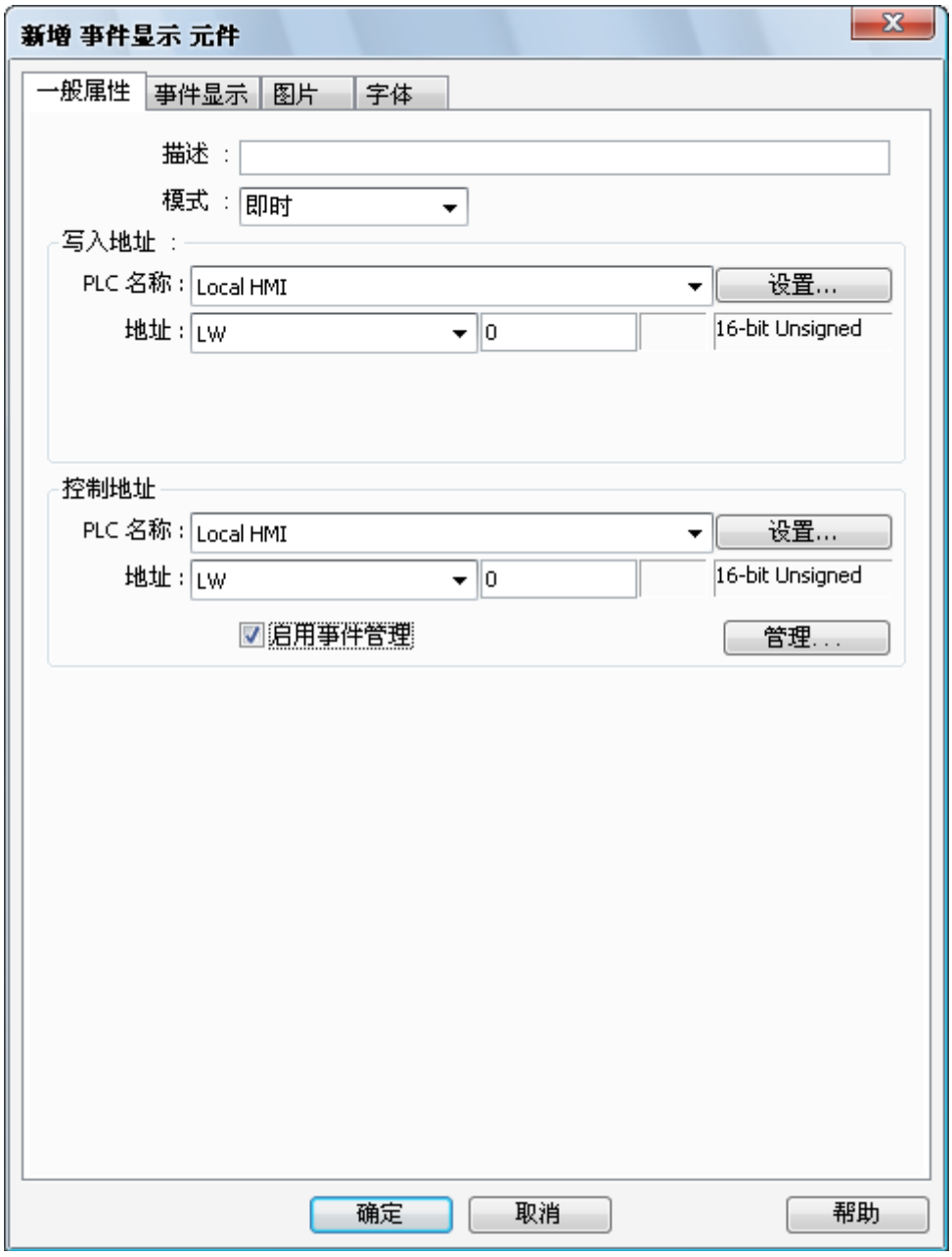
13.22.2 设定



按下工具列上的“事件显示”按钮后，即会出现“事件显示”元件属性对话框，正确设定各项属性后按下确定键，即可新增一个“事件显示”元件。

一般属性设定

eMT、iE、cMT-HD 系列机型

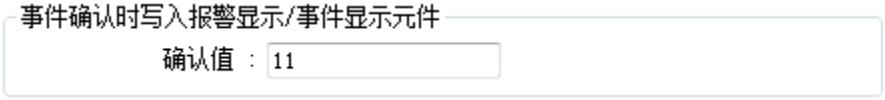


设定	描述
模式	选择事件来源的形式，可以选择“实时”或“历史”。 ● 实时 将会显示所有自开机以来被触发的事件内容。 ● 历史 系统会读取内存内的事件文件，显示数据于元件中。若检视的内容为当天的历史记录，将会每隔 10 秒钟自动更新检视的内容。

确认地址

选择“实时”模式时可设定此地址。

当事件被确认时，在“事件登录”»“讯息”页设定的“事件确认时写入报警显示/事件显示元件”中的数值会被输出到“事件显示元件”的“确认地址”。详细信息请参考《7 事件登录》



历史数据控制

选择“历史”模式时可设定此地址。

- 无勾选“启用读取多个历史数据”

历史模式仅可显示单一天的历史事件记录。当历史数据控制地址为 LW-n 时，于 LW-n 输入特定数值可显示对应的历史数据。

系统通过索引来选择历史记录 .evt 档：

输入 0 则显示最近一日的历史资料

输入 1 显示第二近期的一笔历史资料

输入 2 显示第三近期的一笔历史资料

以此类推。

假设历史数据控制地址是 LW-100，而目前有以下四笔历史资料：EL_20140420.evt 、 EL_20140423.evt 、 EL_20140427.evt 、 EL_20140503.evt。当输入数值到 LW-100 时，所显示的历史数据如下表：

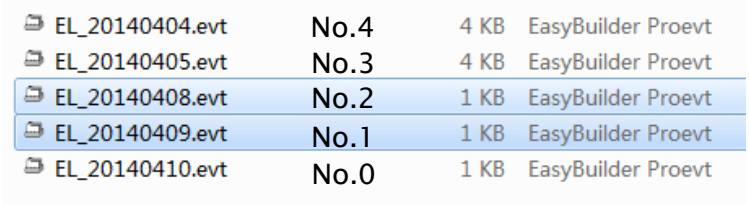
LW-100 数值	相对应的历史资料
0	EL_20140503.ev
1	EL_2□140427.evt
2	EL_20140423.evt
3	EL_□0140420.evt

- 勾选“启用读取多个历史数据”

勾选后，可于一个事件显示元件上显示多天的历史事件资料。使用时，会占用两个连续地址。假设“历史数据控制”地址为 LW-n，则 LW-n 与 LW-n+1 分别控制欲显示的起始数据索引及范围。

选择“天数”：

历史数据显示范围由 LW-n 标记开始算起，往前推算 LW-n+1 天的资料。假设 LW-n 输入数值为“1”，LW-n+1 输入数值为“3”，则表示显示的历史数据范围由 20140409 开始，到往前推算三天 (包括 20140409)，所以应该显示 20140407~20140409 范围内的资料。但是历史数据库中 20140407 数据不存在，所以显示的历史数据为 20140409 与 20140408 的资料。



选择“最后历史数据索引”：

历史数据显示范围由 LW-n 标记开始算起，往前推算 LW-n+1 笔的资料。假设 LW-n 输入数值为“1”，LW-n+1 输入数值为“3”，显示的历史数据为下图中 No. 1、No. 2、No. 3 的历史资料。若设定的显示范围超过数据的数目，则 LW-n+1 无作用，仅会显示 LW-n 的数据。

系统最多可显示 4MB 历史数据，超出部份系统将略过。

以下为显示数据过大的例子。

5 个历史资料，每个 0.5 MB → 最多可显示: 8 x 0.5MB

5 个历史资料，每个 1 MB → 最多可显示: 4 x 1MB

5 个历史资料，每个 1.5 MB → 最多可显示: 2 x 1.5MB + 1 x 1MB *(部分)*

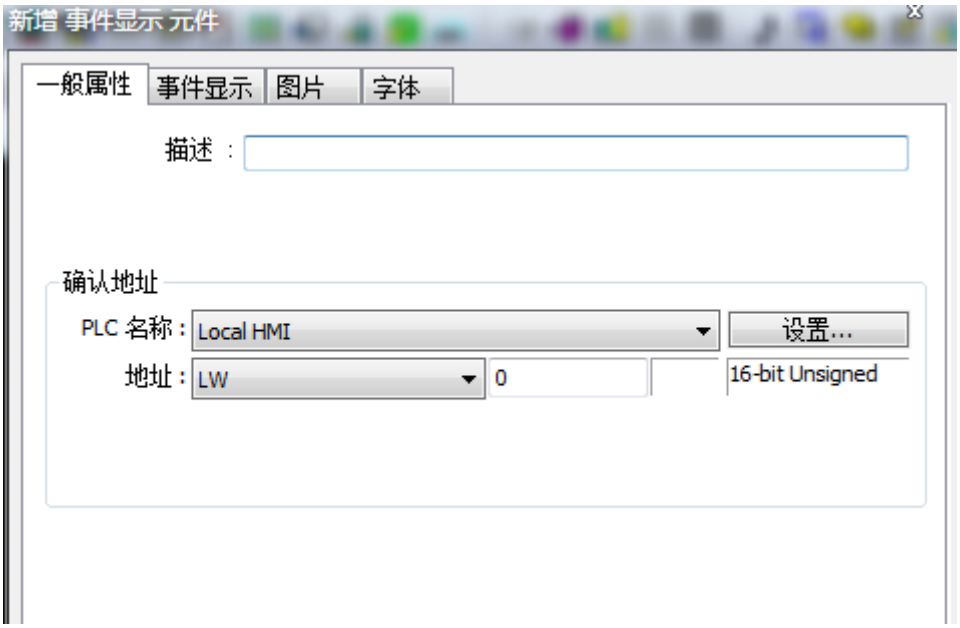
控制地址

启用事件管理

启用后，将特定的数值写入寄存器 LW-n 及 LW-n+1 可对 「事件显示」元件给予不同的命令，命令条件及内容如下表：

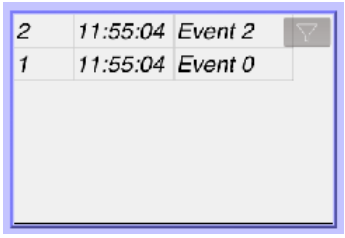
地址	数值	命令内容
LW-n	0	显示所有事件
	1	隐藏「已确认」事件
	2	隐藏「已恢复」 件
	3	隐藏「已确认」或「已恢复」 件
	4	隐藏「已确认」与「已恢复」事件
LW-n+1	1	删除选择的单一事件

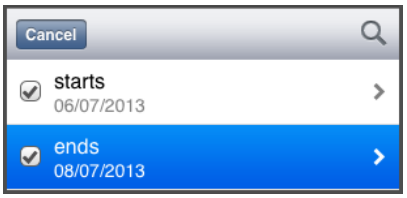
cMT-SVR 系列机型



cMT-SVR 中的事件显示元件，会显示所有已发生的事件并实时更新。

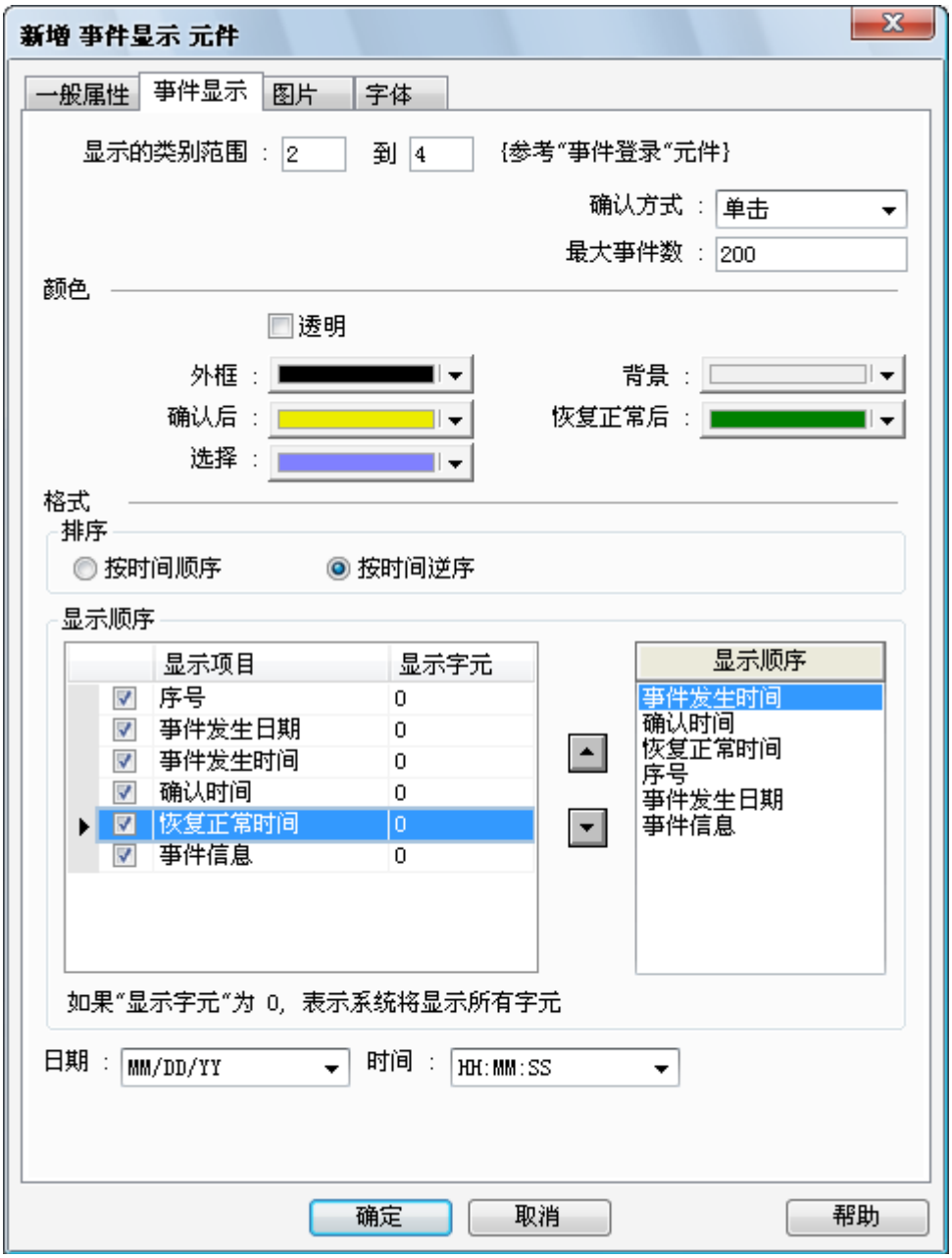
点选元件右上方的漏斗图标可以设定开始和结束的日期。如都未勾选，则会显示所有事件。



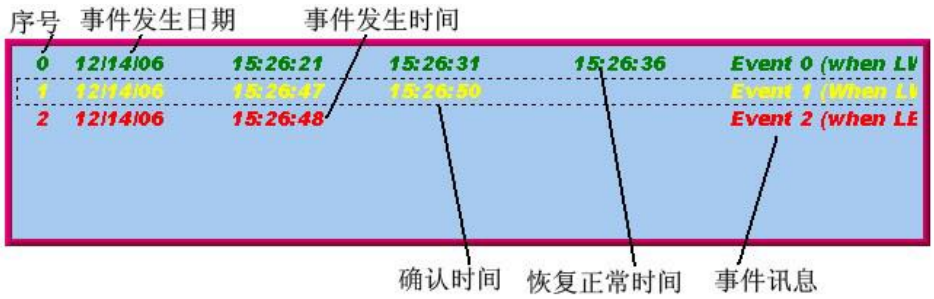


事件显示设定

cMT-SVR 系列



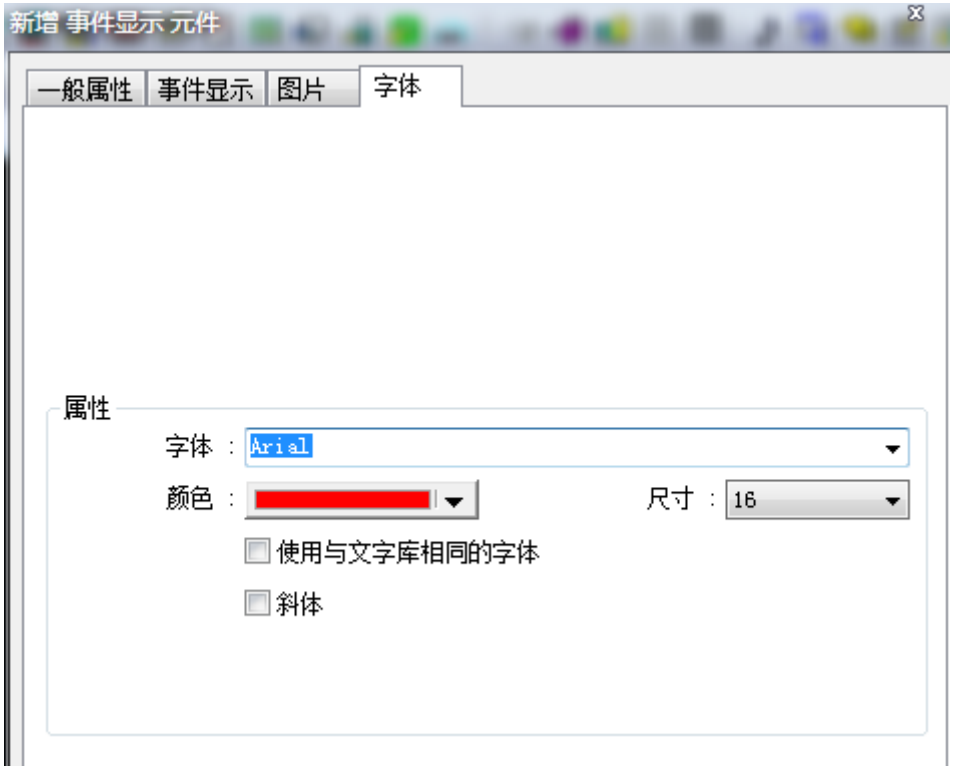
设定	描述
显示的类别范围	事件的“类别”需满足此项设定范围才会被显示 (事件的“类别”在“事件登录”中设定)。假设“显示的类别范围”被设定为 2 到 4，则仅有“类别”2 或 3 或 4 的事件，才会被显示在“事件显示”元件中。详细信息请参考《7 事件登录》中有关“类别”的解释。
确认方式	可选择“单击”或“双击”的确认方式。当事件发生时，用户可依设定方式来做确认的动作。此处所谓“确认”的动作是指用户对于已发生并显示在“事件显示”元件上的事件，此时除了会将该事件的显示颜色转变为“确认”的颜色之外，也会将此事件预先设定的输出值，写至“确认地址”所设定的地址上。

	当输出地址为 LW-100，且事件确认时的写入值为 31，则当用户使用“确认”的动作时，LW-100 中的数据将被设定为 31，利用此项功能搭配“间接窗口”元件，可以在不同事件发生时弹跳出不同的窗口并说明事件的内容。
最大事件数	元件所能显示事件的最大数目。当元件所显示的事件已等于所设定的最大数目时，新发生的事件将取代已发生的事件中的第一笔资料。
颜色	设定事件在确认后、恢复正常后、以及被点选时等状态下的显示颜色。系统将绘出虚线显示刚被点选的事件。 
格式	 **按时间顺序** 较晚发生的事件被排列在后 (或在下)。 **按时间逆序** 较晚发生的事件被排列在前 (或在上)。 **显示顺序** 用户可勾选想要显示的信息以及设定显示顺序。 **日期** 选择显示事件发生日期的格式，共有以下 4 种模式。 MM/DD/YY、DD/MM/YY、DD.MM.YY、YY/MM/DD **时间** 选择显示事件发生时间的格式，共有以下 4 种模式。 HH:MM:SS、HH:MM、DD:HH:MM、HH

字型

实时模式： 可以设定斜体及尺寸。字型将根据“事件登录”中的设定显示

历史模式： 可以设定斜体及尺寸，字型及颜色，或是勾选“使用文字标签库中相同的字型”。



13.23 触发式数据传输

13.23.1 概要

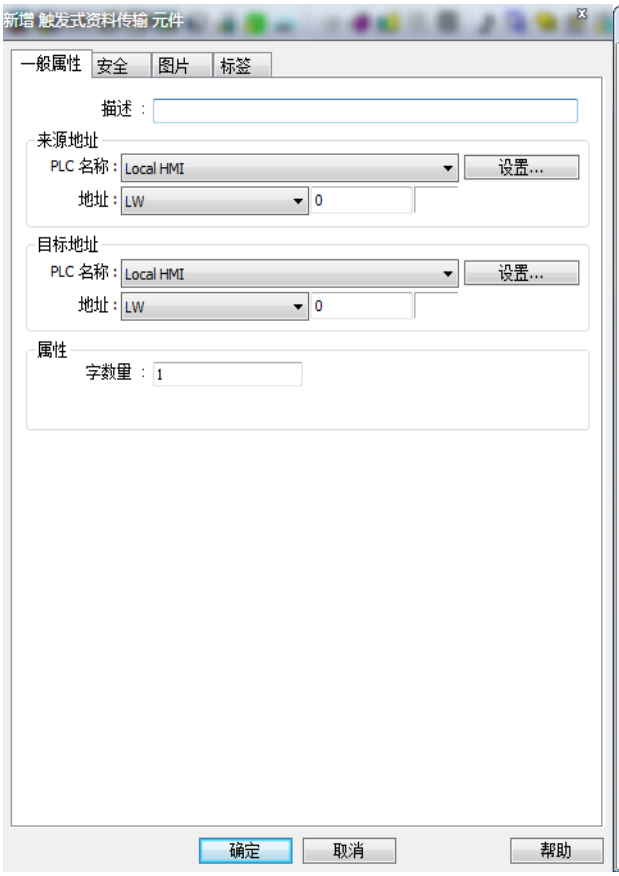
“触发式数据传输”元件可以将指定地址中的数据传送到其它地址中。可以使用手动按钮的方式启用数据传送，也可以利用特定地址的状态改变，来触发数据传输的动作。
若使用 cMT-SVR 系列则无模式选择，预设即为手动模式。

13.23.2 设定

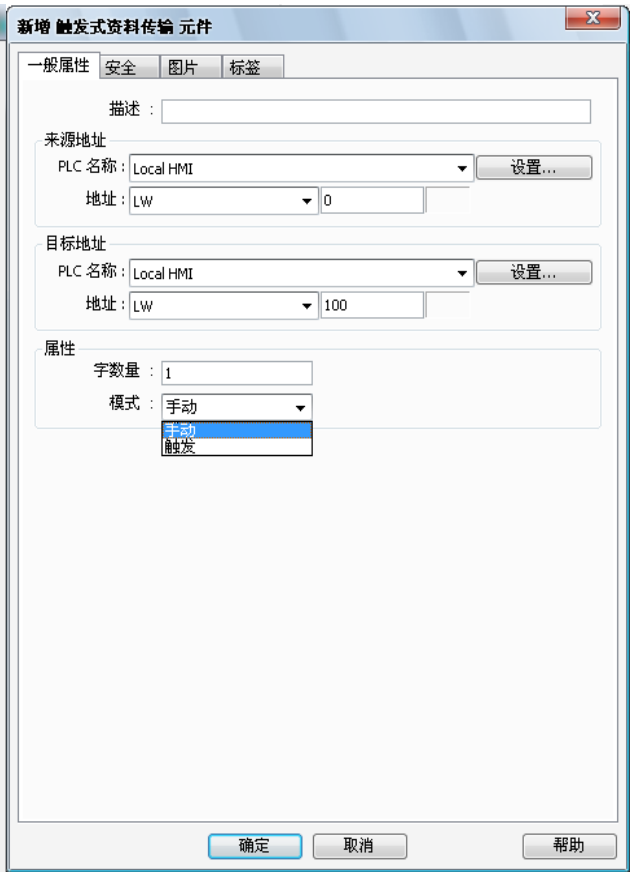
按下工具列上的“触发式数据传输”按钮后，即会出现“触发式数据传输”元件属性对话框，正确设定各项属性后按下确认键，即可新增一个元件。

一般属性设定

cMT-SVR 系列



eMT、iE、cMT-HD 系列



设定	描述
来源地址	设定被传送数据的来源地址。
目标地址	设定数据传送的目的地址。
属性	**字符数量** 数据的传送数量，单位为字符。 **模式** **手动模式** 需使用手动的方式按下数据传输元件，才会进行数据传送的动作。 **触发模式** 利用所指定寄存器状态的改变来触发数据传送的动作，利用“触发模式”选择需要的触发方式，这些触发方式包括： 可以选择状态由 OFF 变为 ON 或由 ON 变为 OFF 时，执行数据传输动作。也可选择状态改变时 (OFF<->ON)，即执行数据传输动作。
触发地址	触发模式所使用的寄存器地址在“触发地址”中设定。

Note

使用位触发的数据传输元件时，必须将控制位寄存器置于同一窗口中，触发模式的数据传输才会启动。若将触发式数据传输元件置于公用窗口，则当在任一窗口中，控制位寄存器的指定状态改变时，即可启动触发模式数据传输。

13.24 备份

13.24.1 概要

利用“备份”元件 可以将配方数据(RW，RW_A)、事件记录、配方数据库、指定的资料取样记录及操作记录复制到扩充装置（SD 卡或 U 盘)，并可以指定备份的时间范围或格式。例如事件记录原来储存在 SD 卡，此时可以在不需关机的情形下插上 U 盘，并利用“备份”元件复制一份相同的数据到 U 盘，并在不需关机的情形下，直接拔取 U 盘，这些数据即可移至 PC 做进一步的分析。当备份动作进行中时，”LB-9039”的状态将维持在 ON。另外搭配“邮件”的设定，可以通过邮件功能将数据以夹带方式通过邮件发送至指定的收件人信箱。

13.24.2 设定

若使用 eMT, iE, cMT-HD 系列，按下工作列上的“备份”按钮后即会出现“备份”元件属性对话框，正确设定各项属性后按下确认键，即可新增一个“备份”物件。

若使用 cMT-SVR 系列，按下工作列上的“备份”按钮后即会出现备份元件管理对话框；按下 “新增” 按键，并利用出现的 “备份” 元件属性对话框正确设定元件的各项属性，最后按下确定键即可新增一个“备份” 物件。

一般属性设定

cMT-SVR 系列

eMT、iE、cMT-HD 系列

设定	描述
来源	**“RW”、”RW_A”、”配方数据库”、”事件记录”、”资料取样记录”** 这些选项用来选择要复制的文件来源，当选择文件来源为“资料取样记录”时，需使用“资料取样元件索引”选择要复制的资料取样记录。
备份位置	设定来源文件的备份位置。 **SD 卡 / U 盘** 备份至 SD 卡或 U 盘。外部装置需事先连接在触摸屏上。 若使用 cMT-SVR 系列，SD 卡和 U 盘只能备份 “RW”、”RW_A” 和 “配方数

据库”。

远程打印/备份服务器（仅 eMT, iE, cMT-HD 系列）

备份至远程备份服务器。若使用此选项，用户需先在“系统参数设置”»“打印/备份服务器”中设定。

详细信息请参考《26 EasyPrinter》。

邮件

将备份数据以邮件寄出。

若使用此选项，用户需先在“系统参数设置”»“邮件”中设定，再至“备份”元件属性 »“e-Mail”设定收件者、主题、信息等邮件设定。

储存格式

选择想要储存的备份文件格式。

eMT, iE, cMT-HD 系列：

- HMI 事件记录文件(.evt) / HMI 数据记录文件(.dtl)
- Comma Separated Values (.csv)

当备份事件记录文件为 CSV 格式时，可以在 EXCEL 中发现 data fields。

	A	B	C	D	E
1	Event	Category	Date	Time	Message
2	0	1	2013/7/4	16:12:11	Event A
3	2	1	2013/7/4	16:12:12	Event A
4	0	0	2013/7/4	16:12:33	Event B
5	2	0	2013/7/4	16:12:36	Event B
6	0	0	2013/7/4	16:12:37	Event B
7	1	0	2013/7/4	16:12:37	Event B
8	2	0	2013/7/4	16:12:39	Event B
9	0	0	2013/7/4	16:12:40	Event B

0: 事件触发

1: 事件确认

2: 事件恢复正常

HMI 事件记录文件 (.evt) 和触摸屏数据记录文件 (.dtl) 都可以利用 EasyConverter 轻松转成 .xls 或 .csv 格式。

- SQLite 数据库文件（.db)

cMT-SVR 系列：

- SQLite 数据库文件（.db)
- Comma Separated Values (.csv)

范围

几天内

设定储存天数。例如，起始时间设定为“昨天”并选择 2 天。表示储存昨天及前天的数据。选择全部，可储存最多 90 天的数据。

触发

(eMT, iE, cMT-HD 系列)

模式

选择元件的执行方式。

手动

用户只需按压元件，即可执行文件复制动作。

触发 (位)

当指定的寄存器状态改变符合触发条件时，元件将执行文件复制动作。

可以选择状态由 OFF 变为 ON 或由 ON 变为 OFF 时，执行文件复制动作，也可选择状态改变时(OFF<->ON)，即执行文件复制动作。

触发 (字符)

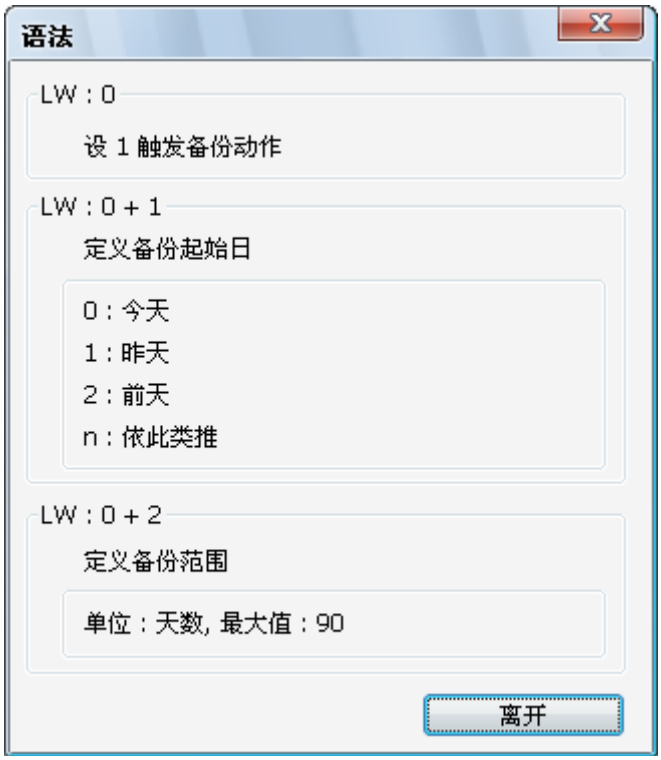
可以通过“触发地址”设定所需备份数据的时间范围。

“触发地址”的用法如下(假使目前触发地址被设定为 LW-0)：

LW-0：当此地址所指定寄存器中的数据由 0 变为 1，将触发备份动作。

LW-1：此地址所指定寄存器中的数据用来指定备份的起始时间。

LW-2：此地址所指定寄存器中的数据用来指定备份的天数 (最多 90 天)。



触发地址 (cMT-SVR 系列)	当指定的寄存器状态设为 ON 时，元件将执行文件复制动作。备份完成后，该寄存器将被设为 OFF。

Note

请注意所有的数据必须要被储存在任意一内存之中 (如：HMI memory 或 U 盘或 SD 卡)，否则无法使用备份功能。

- 单次备份最大天数为 90 天。(不包含 cMT-SVR 系列)

关于 cMT-SVR 系列同步资料取样记录及事件记录至外部装置的方式请参考 《7 事件登录》、《8 资料取样》。

13.25 媒体播放器

13.25.1 概要

第一次使用媒体播放器时，必须要使用 Ethernet 下载工程文件到触摸屏。EasyBuilder Pro 会自动安装媒体播放器的驱动。媒体播放器功能不只是播放影片，另外也提供额外操作功能例如搜寻，放大缩小，音量调整等作用。以动态影片来指示作业与维修保养，建立更容易了解且任何现场作业人员都能进行维修保养的环境。

13.25.2 设定

按下工作列上的“媒体播放器”按钮后即会开启“媒体播放器”元件属性对话框，正确设定各项属性后按下确认键，即可新增一个“媒体播放器”元件。

一般属性设定

设定	描述
控制地址	● 启用时 用户可针对“媒体播放器”进行控制并且得到播放信息。必须指定一地址用来控制元件行为。 ● 未启用时 无法手动控制影片的播放状态。在窗口开启时，系统会自动播放影片。 **命令**(控制地址 + 0) 控制“媒体播放器”的动作模式。 **参数 1** (控制地址 + 1) 相对于特定命令所传入的参数 1。 **参数 2** (控制地址 + 2) 相对于特定命令所传入的参数 2。 **状态** (控制地址 + 3) 记录文件状态、播放情况及错误代码。 **文件索引**(控制地址 + 4)

	播放的文件位于指定目录下的索引（以文件名排序，建议以数字为起始文件名)。 **开始时间** (控制地址 + 5) 影片开始时间（秒)。(通常为 0) **结束时间** (控制地址 + 6) 影片结束时间（秒)。(影片的时间长度) **更新影片播放时间** 启用时，每隔“更新周期”(秒）会将影片已播放时间写入“播放时间”寄存器中。 **更新周期** “播放时间”的更新周期，范围由 1 至 60 秒。 **播放时间** (控制地址 + 7) 影片已播放时间（秒) (介于“开始时间”与“结束时间”之间)。
外部装置	选择播放 SD / USB 里的文件。 **文件夹名称** 影片文件放置的文件夹名称。文件必须被放置于文件夹中且文件夹只能为一层，多层文件夹将不会被接受（例如指定”文件夹名称”为”example\ex”将会出现错误)。 “文件夹名称”不可空白，必须为英文或数字，且全部由 ASCII 字符所组成。
属性	**自动重复** 当所有的影片播放结束，会自动跳回第一个影片从头播放。 例如：Video 1 > Video 2 > Video 1 > Video 2 **背景** 指定元件背景颜色。

Note

预设的寄存器格式为 16 位无正负号；当指定之寄存器为 32 位时，只有较低的 16 位产生作用，并请将较高的 16 位固定为 0。

控制命令

以下说明各种控制命令的设定。

- **播放索引文件**

“命令” = 1
“参数 1” = 文件索引
“参数 2” = 忽略（应设为 0)

Note

- 文件以档名排序。
- 假如找不到文件，则将“状态”的位 8 设为 ON。

- 若需中途切换影片，请先将正在播放的影片停止。

- **播放上一个文件**

“命令” = 2
“参数 1” = 忽略(应设为 0)
“参数 2” = 忽略(应设为 0)

Note

- 若“文件索引”为 0，则输入命令 2 会从头播放原文件。
- 假如找不到文件，则将“状态”的位 8 会被设为 ON，代表命令错误。

- **播放下一个文件**

“命令” = 3
“参数 1” = 忽略(应设为 0)
“参数 2” = 忽略(应设为 0)

Note

- 如果找不到文件，则播放索引值 0 的文件。
- 假如找不到文件，则将“状态”的位 8 会被设为 ON，代表命令错误。

- **暂停/播放 切换**

“命令” = 4
“参数 1” = 忽略(应设为 0)
“参数 2” = 忽略(应设为 0)

- **停止播放并关闭文件**

“命令” = 5
“参数 1” = 忽略 (应设为 0)
“参数 2” = 忽略 (应设为 0)

- **从指定位置开始播放**

“命令” = 6
“参数 1” = 目标时间 (以秒为单位)
“参数 2” = 忽略 (应设为 0)

Note

- 参数 1 (目标时间) 应小于结束时间，若超出结束时间则由结束时间前 1 秒开始播放。

- **往后跳跃 (秒)**

“命令” = 7
“参数 1” = 目标时间 (以秒为单位)
“参数 2” = 忽略 (应设为 0)

Note

- 从目前时间往后跳跃“参数 1”指定秒数后开始播放。若系统目前为暂停播放影片状态，则跳跃动作会在开始播放后才会进行。
- 若播放时间超过结束时间，则由结束时间前 1 秒开始播放。

- **往前跳跃 (秒)**

“命令” = 8
“参数 1” = 目标时间 (以秒为单位)
“参数 2” = 忽略 (应设为 0)

Note

- 从目前时间往前跳跃“参数 1”指定秒数后开始播放。若系统目前为暂停播放影片状态，则跳跃动作会在开始播放后才会进行。
- 若播放时间少于开始时间，系统会从头播放影片。

- **设定音量**

“命令” = 9
“参数 1” = 音量 (0 ~ 128)
“参数 2” = 忽略(应设为 0)

Note

- 预设为最大音量 (128)。

- **设定影像放大倍率**

“命令” = 10
“参数 1” = 影像大小 (0 ~ 16)
“参数 2” = 忽略(应设为 0)

Note

- “参数 1 = 0”：配合元件大小。
- “参数 1 = 1 ~ 16”：放大倍率范围 25% ~ 400%，设定 1 为放大 25%，2 为 50%，3 为 75% 以此类推。

- **状态(控制地址 + 3)**

当 HMI 正在播放影片，则“文件开启位”位 00 及“文件播放位”位 01 将被同时设为 ON (0 → 1)。相反地，若找不到文件或输入的命令不正确，则“错误命令位”位 08 将会被设为 ON (0 → 1)。若在播放过程中发现文件格式错误或任何磁盘 I/O 错误(例：USB 磁盘被拔除)，则 “文件错误位” 位 09 将被设为 ON(0→1)。

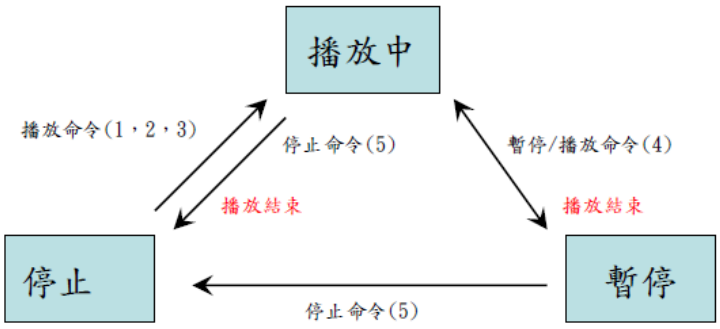
15	09	08		02	01	00	位元
保留 (0 固定)	0	0			0	0	

位元 00： 档案开启位(0：未开启档案；1：已开启档案)
位元 01： 档案播放位(0：未播放档案；1：档案播放中)
位元 08： 错误命令位(0：正确命令格式；1：错误命令或参数)
位元 09： 档案错误位(0：档案格式及读取正确；1：档案格式或读取错误)

Note

- 参考下图“媒体播放器”之状态转换图可知：

“停止”时状态 = 0, “暂停”时状态 = 1, “播放”时状态 = 3



- 请通过设定“命令”，”参数 1”及“参数 2”来操作元件，并将其它地址视为只读。

预览设定

用户可利用预览功能来检查 HMI 是否支持欲播放的影片格式。

设定	描述
前进<< / 后退 >>	往前或往后快转 (以 1 分钟为单位)。
播放 / 暂停	可选择影片开始播放或暂停播放。
停止	停止播放影片并关闭文件。若需测试另一影片，必须先停止播放目前影片。
载入	选择要预览的影片。

Note

- 用户需注意 HMI 上同一时间只能有一个影片档被开启。
- 假如用户没有启用“控制地址”且没有设定“自动重复”，则指定目录下的第一个文件播完一遍后，系统会自行将影片关闭。
- 当没有启用“控制地址”时，元件生成后自动到指定目录下找寻第一个文件 (以檔名排序)开始播放。
- 当影片可以使用媒体播放器的预览功能，表示触摸屏支持此影片格式并且可以播放。若是在触摸屏上有播放质量不佳的状况请调整影片的分辨率。
- 支持文件格式有: mpeg4，xvid，flv...等等。

13.26 数据传输 (背景)

13.26.1 概要

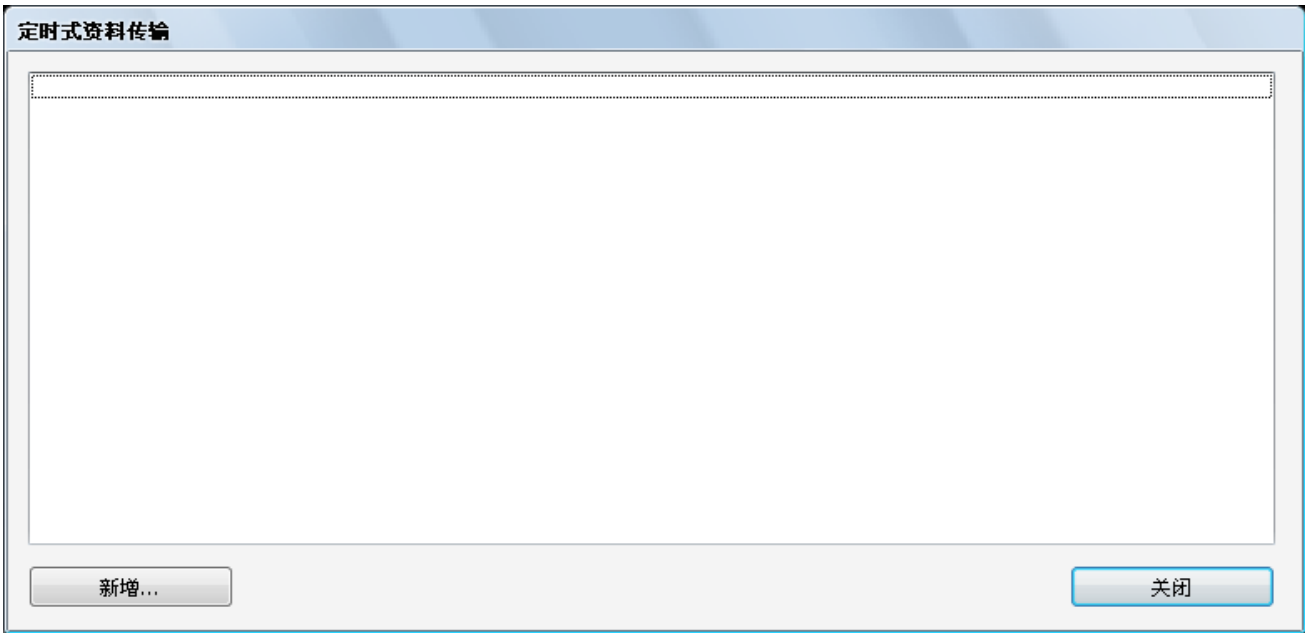
“数据传输 (背景)”元件与“触发式数据传输”元件雷同，皆用来将指定地址中的数据传送到其它地址中。与“触发式数据传输”元件不同的是“数据传输 (背景)”元件使用固定的频率、自动执行数据传送的工作，并且可以传送使用位为计数单位的数据。

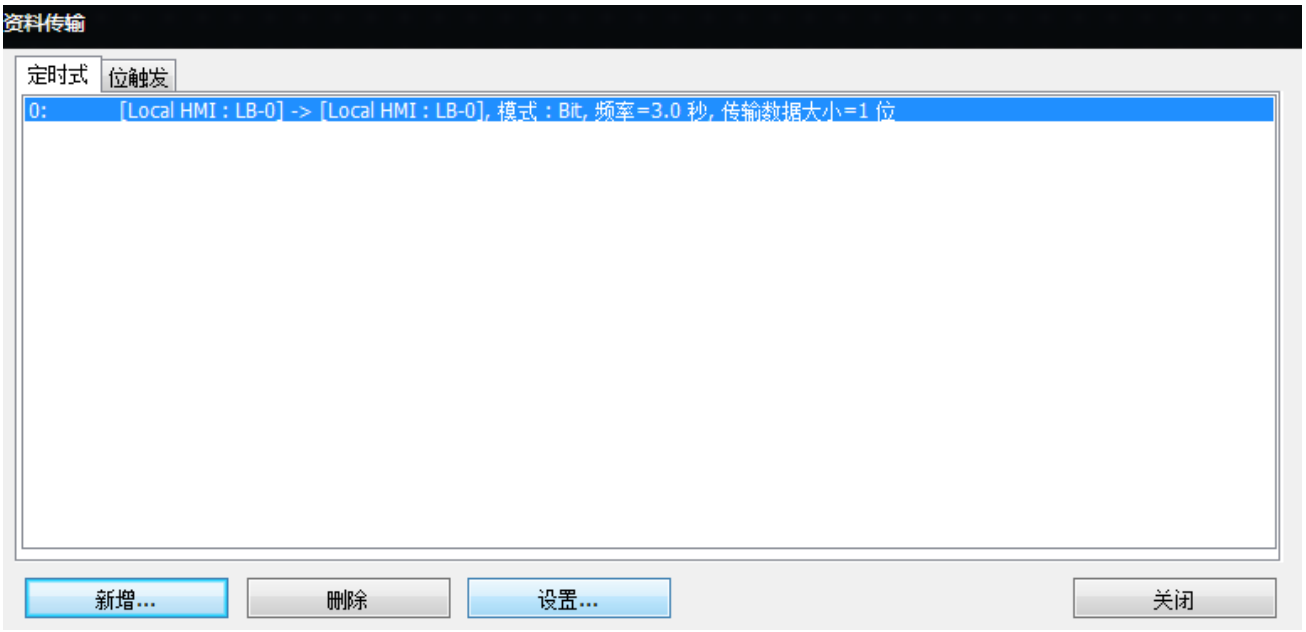
若使用 cMT-SVR 系列，则分为“定时式”及“位触发”，皆为系统背景执行。”定时式”触发与上述相同。”位触发”则是当特定位的状态改变时，就会触发数据传输的动作。关于 cMT-SVR 系列的位触发请参考《13.26.2.2 位触发数据传输》。

13.26.2 设定

按下工具列上的“数据传输 (背景)”按钮打开“数据传输” 管理对话框。按下“新增”按钮，正确设定各项属性后，即可新增一个元件。管理对话框中可看出此元件的内容。

eMT、iE、cMT-HD 系列

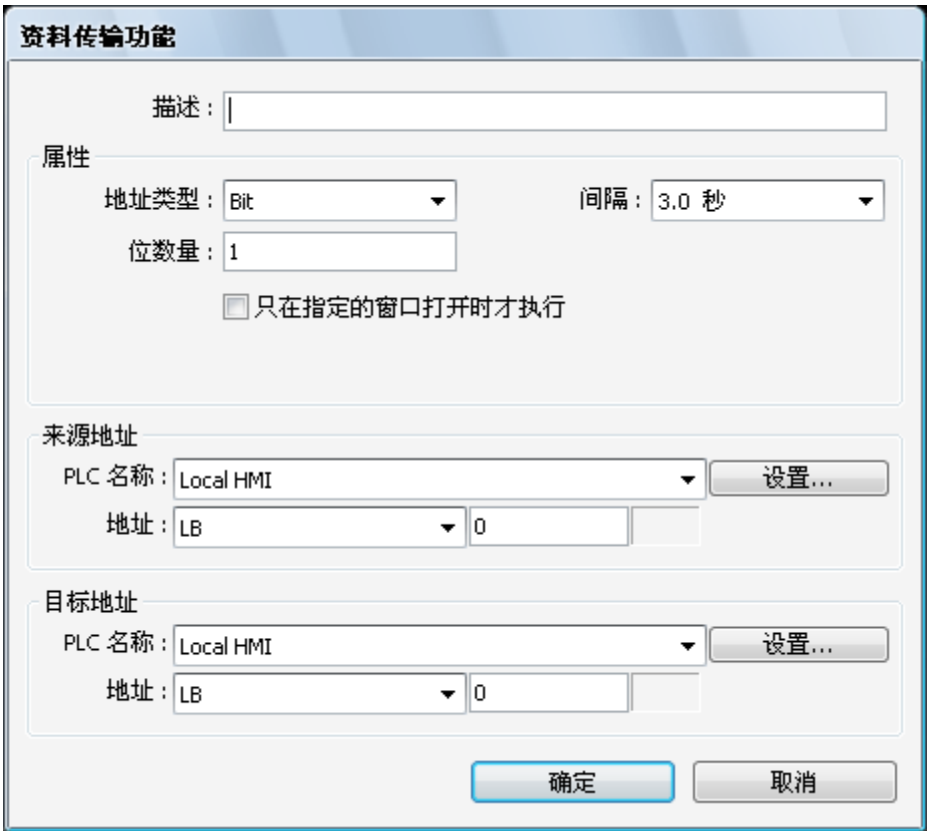




定时式数据传输

一般属性设定

开启“定时式”页面，按下“新增”则出现“定时式数据传输”对话框。

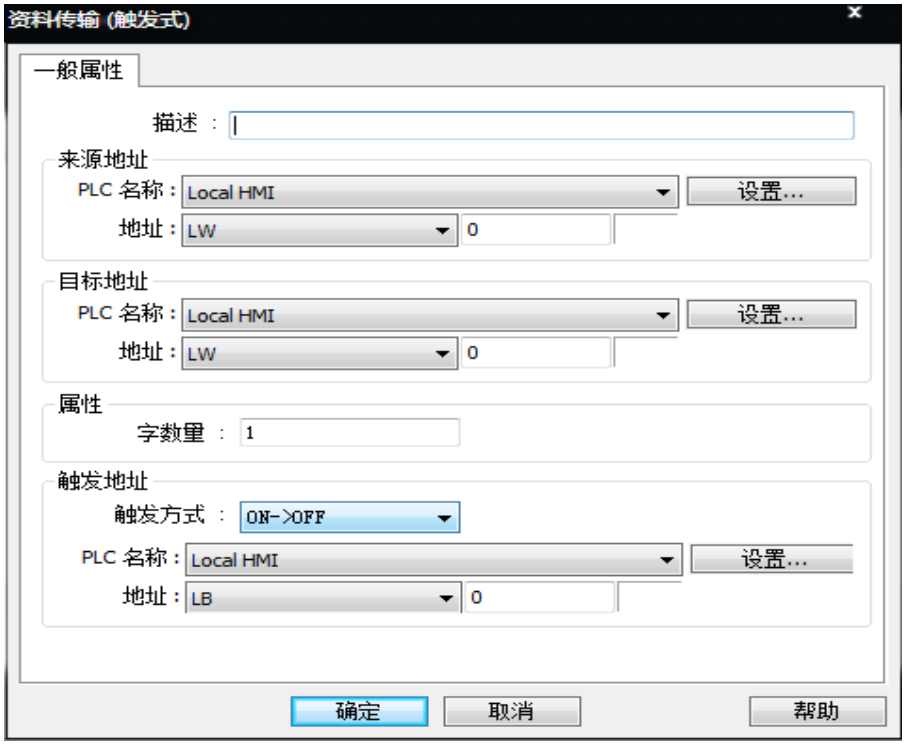


设定	描述
属性	**地址类型** 选择被传送数据的类型，可以选择“Bit”或“Word”的数据。 **位数量 / 字符数量** 当“地址类型”选择“Bit”类型时，数据传送单位为 bit，使用“位数量”设定传送数量。 当“地址类型”选择“Word”类型时，数据传送单位为 word，使用“字符数量”设定传送数量。 **间隔** 数据传送频率，例如选择 3 秒，则每隔 3 秒，将传送数据到指定的地址中。 较小的间隔或是大量的数据传输可能会导致系统执行速度变慢，建议用户拉长传送的间隔或是一次传送小量的数据，以避免系统执行速度变慢。 当需要设定短时间的传输时，请注意设定间隔的时间要大于数据传输的时间。例如：单次数据传输操作需要 2 秒，则间隔时间须设置超过 2 秒。
来源地址	设定数据传送的来源地址。
目标地址	设定数据传送的目标地址。

位触发数据传输

一般属性设定

开启“位触发”页面，按下“新增”则出现“数据传输 (触发式)”对话框。



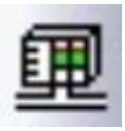
设定	描述
来源地址	设定数据传送的来源地址。
目标地址	设定数据传送的目标地址。
字符数量	数据的传送数量，单位为字符。
触发地址	设定执行触发的寄存器地址及触发模式。 **触发模式** 当指定的状态改变时，执行数据传输。可以选择状态由 ON 变为 OFF 或由 OFF 变为 ON 时，才执行数据传输，也可选择状态改变时(ON<->OFF)，即执行数据传输。

13.27 PLC 控制

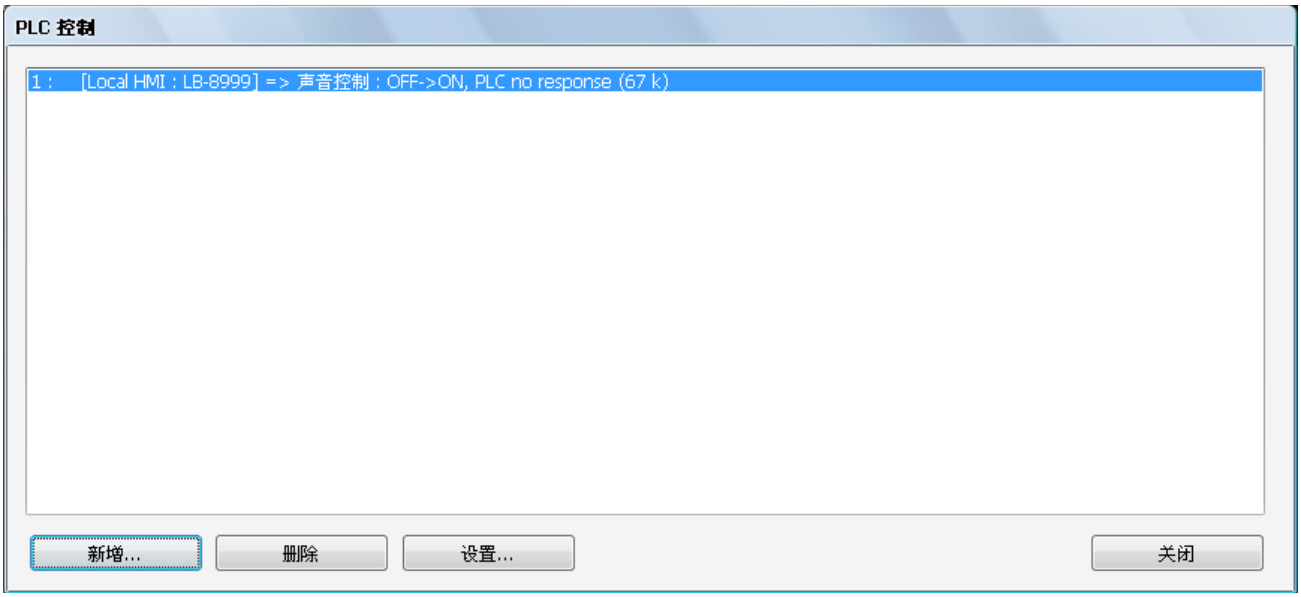
13.27.1 概要

当相应的控制命令被触发时，”PLC 控制”元件能启动某个特定的动作。

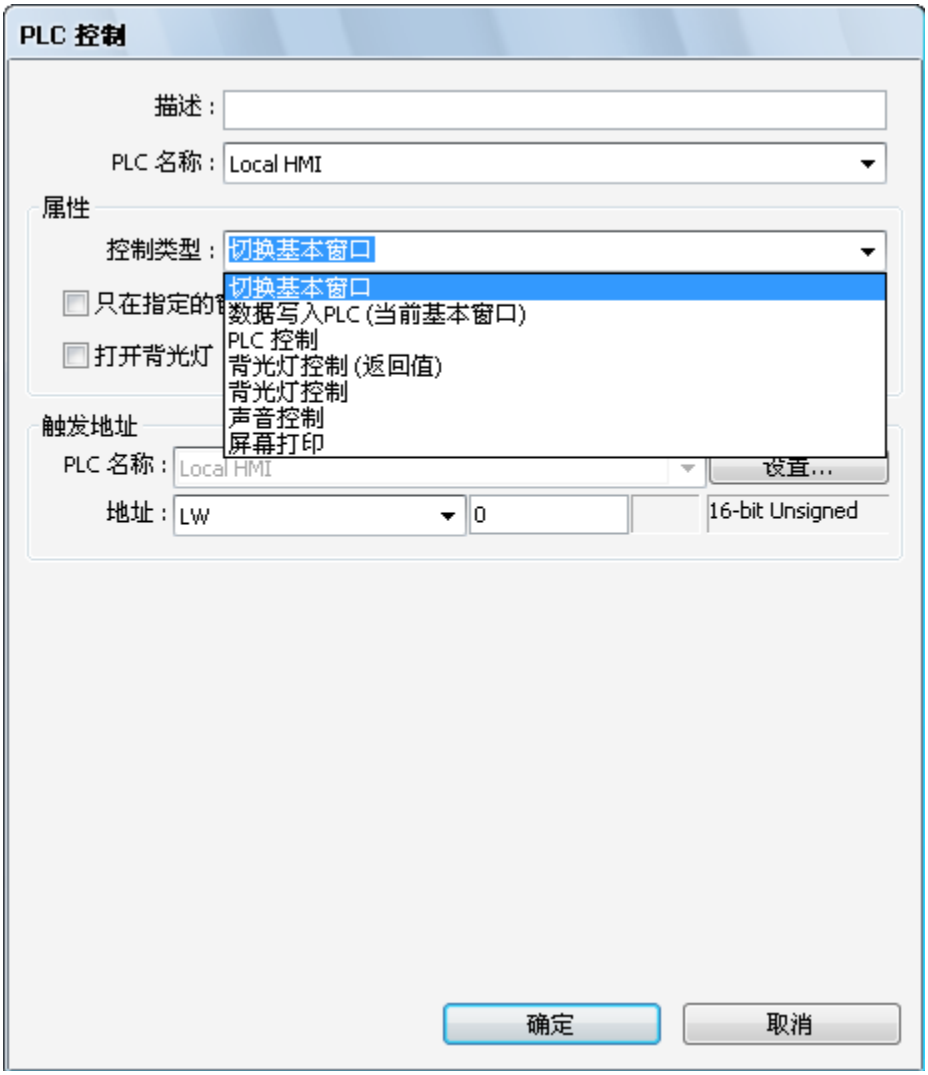
13.27.2 设定



按下工具列上的“PLC 控制”按钮后即会出现“PLC 控制”元件管理对话框，接着可按下“新增”按键，并利用出现的“PLC 控制”元件设定对话框正确设定元件的各项属性，最后按下确定键即可新增一个“PLC 控制”元件。



下图为按下“新增”按键后所出现的设定对话框。请见下面《13.27.2.1 控制类型说明》。

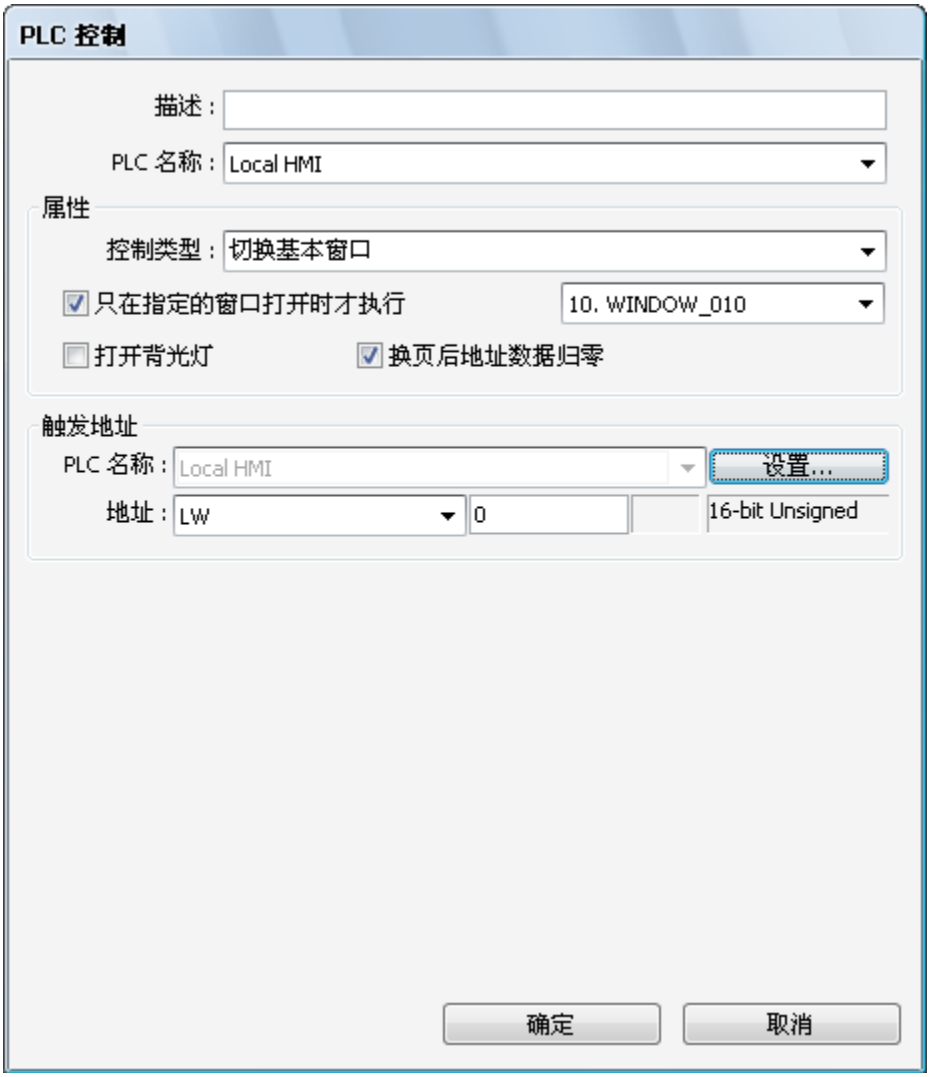


Note

- cMT-SVR 系列不支持“PLC 控制”、”背光灯控制”等选项。

控制类型说明

- 切换基本窗口



设定	描述
只在指定的窗口被开启时才执行	此切换窗口的功能将只在指定的窗口内才有作用。
打开背光灯	若启用此选项，当背光灯为关闭状态时，切换窗口成功后会自动开启背光灯。cMT-SVR 无此选项。
换页后地址数据归零	若启用此选项，切换窗口成功后会将触发地址中的数据归零。
使用窗口编号偏移	当勾选此选项时，”触发地址”所指定寄存器中的数据，再加上“窗口编号偏移”，才是实际切换的目的窗口号码。例如：当触发地址为 LW-0，偏移量为 -10，当 LW 0 的数值为 25 时，窗口将切换到窗口编号 15 (数值 25 + 偏移量-10)。偏移量范围为 -1024 至 1024。如勾选“使用窗口编号偏移”，则

不能选择“换页后地址数据归零”。

Note

- 当“LB－9017”的状态被设定为 ON 时，切换后的窗口编号将不再写至特定的地址中。
- 切换基本窗口的功能。当“触发地址”中的数据改变，且改变后的数据为一个有效的窗口编号时，将关闭目前的窗口并切换至“触发地址”中数据所指定的窗口，并将此时切换后的窗口编号写至“触发地址 ＋ 1 (16bit)”或是“触发地址 ＋ 2 (32bit)”中。

例如：目前的窗口编号为 10，触发地址为 LW-0：

当 LW-0 中的数据由其它数据改变为 11 时，EasyBuilder Pro 除了会将基本窗口切换到窗口 11 之外，也会将 LW-1 中的数据更改为 11。

当切换窗口成功时，切换后的窗口编号的写入地址与“触发地址”中设定的读取地址、变量类型皆有关系，如下表所示：

数据类型	目的窗口编号读取地址 (触发地址)	切换后窗口编号的写入地址
16-bit BCD	地址	地址 ＋ 1
32-bit BCD	地址	地址 ＋ 2
16-bit Unsigned	地址	地址 ＋ 1
16-bit Signed	地址	地址 ＋ 1
32-bit Unsigned	地址	地址 ＋ 2
32-bit Signed	地址	地址 ＋ 2

- **数据写入 PLC (当前基本窗口)**

当切换基本窗口时，会将基本窗口的编号写至“触发地址”中。

- **PLC 控制 (eMT, iE, cMT-HD)**

此项功能提供用户利用寄存器中的数据，控制 PLC 与 HMI 之间的数据传输，数据传输方向包含四种类型，参考下表的内容：

数据传输类型	数据传输方向
1	PLC 寄存器中的数据 → HMI 上的 RW 寄存器。
2	PLC 寄存器中的数据 → HMI 上的 LW 寄存器
3	HMI 上的 RW 配方资料 → PLC 上的寄存器
4	HMI 上的 LW 寄存器 → PLC 上的寄存器

使用此项功能时，由“触发地址”所设定的地址连续四个寄存器中的数据，决定数据传输类型、数据传送数量、数据来源地址与数据传送目的地址等。下表表示各寄存器中数据所表示的意义：

地址	用途	说明
“触发地址”	存放数据传输类型，并决定数据传输的方向。	用来决定数据传输类型，如上表所述，共有四种类型。当寄存器被写入新的数据时，HMI 即执行相应的传输，传输完

		成后会将寄存器中的数据设为 0。
“触发地址” + 1	存放欲传输数据大小	单位为字符(word)。
“触发地址” + 2	存放传输过程中数据来源的地址偏移量	传输的数据来源的起始地址为: **“触发地址” + 4 + 地址偏移量** 以 OMRON PLC 为例，如果此时设定的“触发地址”为 DM-100，而在寄存器“触发地址” +2 也就是 DM-102 中的数据为“5”，则传输的数据来源的起始地址为 DM-109，其中 109 = (100 + 4) + 5。
“触发地址” + 3	存放传输过程中配方数据寄存器 (RW) 或者本地数据寄存器 (LW) 的起始地址	以 OMRON PLC 为例，如果此时设定的“触发地址”为 DM-100，而在寄存器“触发地址” + 3 也就是 DM-103 中的数据为“100”，则传输过程中操作的 RW 或 LW 的起始地址为 RW-100 或 LW-100。

范例 1

假使现在需要使用“PLC 控制”的功能，将 OMRON PLC 中从 DM-100 起始的 16 words 的数据，传输到 HMI 配方内存 RW-200 开始的地址中，设定方法如下：

1. 首先，假设用 DM-10 起始的四个数据寄存器来控制传输。先建立一个“PLC 控制”元件，选择类型为“PLC 控制”，读取地址设定为 DM-10。
2. 确定操作数据的大小和地址的偏移量。
   将 DM-11 设定为 16，表示传输数据的大小为 16 words；将 DM-12 设定为 86，表示数据的来源地址为 DM-100 (100 = 10 + 4 + 86)；将 DM-13 设定值为 200，表示目标地址为 RW-200。
3. 最后，依照数据传输的方向，设定传输类型。
   将 DM-10 设定为 1，表示将传输 PLC 寄存器中的数据到 HMI 上的 RW 寄存器中。
   如果设定 DM-10 值为 3，则传输方向相反。

- **背光灯控制 (返回值) (eMT, iE, cMT-HD)**

当“触发地址”的状态由 OFF 变为 ON 时，HMI 将打开/关闭背光灯，此时也会将“触发地址”的状态设定为 OFF。背光灯关闭时，用户只需碰触屏幕，背光灯即会再度打开。

- **背光灯控制 (eMT, iE, cMT-HD)**

当“触发地址”的状态由 OFF 变为 ON 时，HMI 将打开/关闭背光灯。但因不具备 "返回值" (write back) 功能，此时并不会将“触发地址”的状态设定为 OFF。

- **声音控制**

当“触发地址”的状态改变符合触发条件时，将播放预先指定的声音文件。

可以选择状态由 OFF 变为 ON 或由 ON 变为 OFF 时，播放声音文件。也可选择状态改变时 (OFF<->ON)，即播放声音文件。

- **执行宏指令**

文件中若有编辑完成的宏指令，此选项便会出现。

当“触发地址”的状态改变符合触发条件时，将执行指定的宏指令。

可以选择状态由 OFF 变为 ON 或由 ON 变为 OFF 时，执行宏指令。也可选择状态改变时 (OFF<->ON)，即执行宏指令。或是当状态为 ON 时即执行：只需状态维持在 ON，即可持续执行指定的宏指令 (最快为每 0.5 秒执行一次)。

- **屏幕打印**

当“触发地址”的状态改变符合触发条件时，将打印指定的画面。

可以选择状态由 OFF 变为 ON 或由 ON 变为 OFF 时，打印指定的画面。也可选择状态改变时 (OFF<->ON)，即打印指定的画面。

可以选择要打印的画面，共有三种指定方式：

打印目前显示的窗口

将打印目前显示的画面。

所打印的窗口由 PLC 控制

将利用指定的地址读取数据，此数据即为窗口的号码。如果此窗口存在，则打印这个窗口的内容。

指定被打印的窗口

直接指定要打印的窗口。

若未设定打印机，可选择使用 SD 卡或 U 盘。打印机可在“系统参数设置”的“HMI 属性”页面中设定。

Note

- cMT-SVR 无“打印机”设定。窗口打印会将图片存在 iPad 的图片文件夹中。
- 当指定被打印的窗口不是当前窗口时，系统提供背景打印。
- 指定背景窗口时，该窗口为“直接窗口”或“间接窗口”将不会被打印。

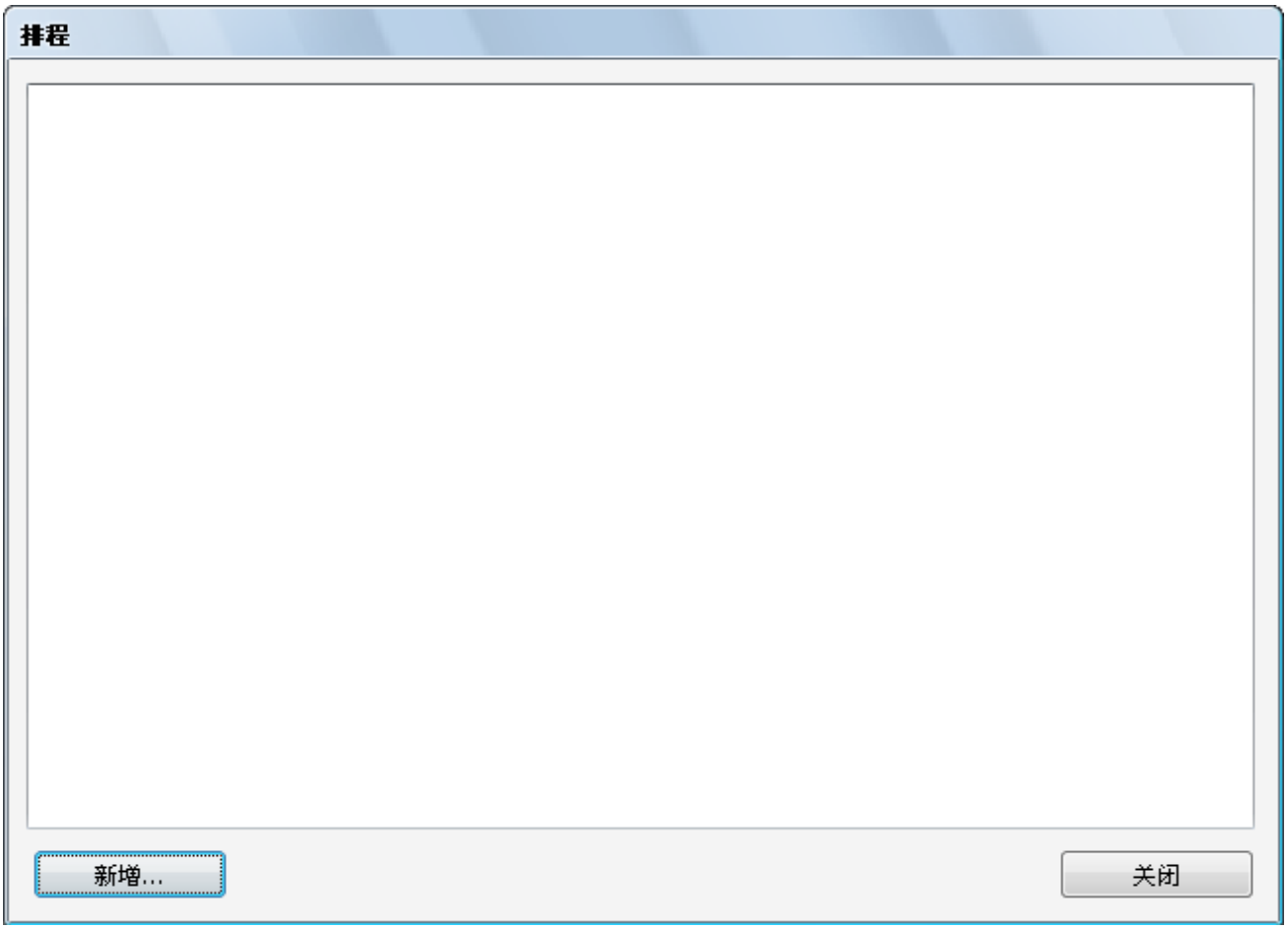
13.28 排程

13.28.1 概要

“排程”可用来设定时刻表，将位设为 ON / OFF 或在字符地址写入数值，适合用来规划一周内的例行程序。

13.28.2 设定

按下工作列上的“排程”按钮后即会出现对话框，按下 “新增” 键，即可进入排程的设定页。

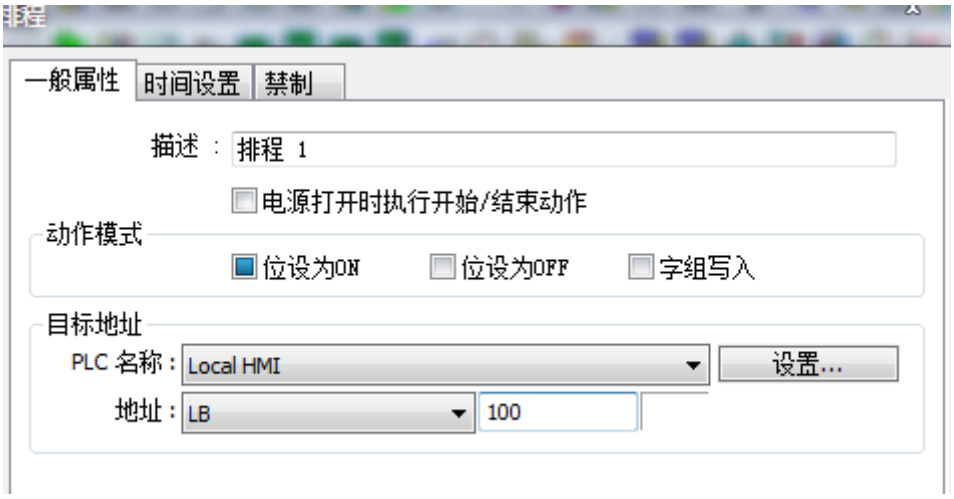


先介绍两个范例再详细说明各项功能：

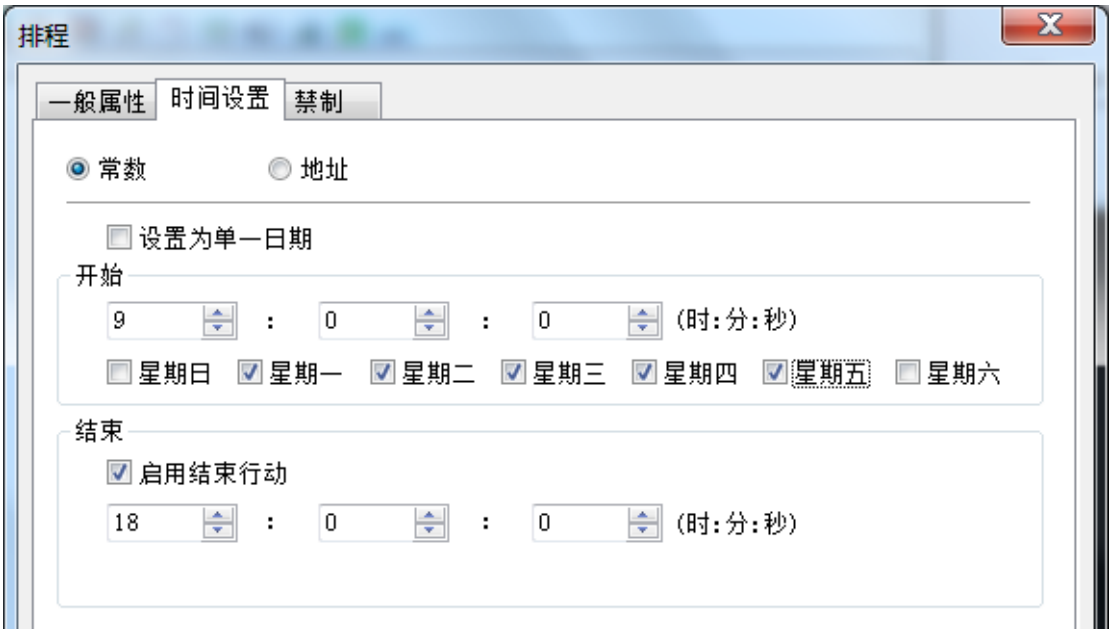
范例 1

马达（地址：LB – 100）从星期一一直运转到星期五，时间由每天上午 9 点到下午 6 点。设定程序为在起始时间（早上 9 点）将地址 LB-100 设为 ON，在结束时间（下午 6 点）将地址 LB-100 设为 OFF。

1. 按下工作列上的“排程”按钮后即会出现对话框，按下“新增”键，即可进入排程的设定页。
2. 选择“一般属性”页面，选定“行动模式”为“位设为 ON”，并设定“目标地址”为 LB-100。



3. 选择“时间设置”页面，接着选择“常数”。



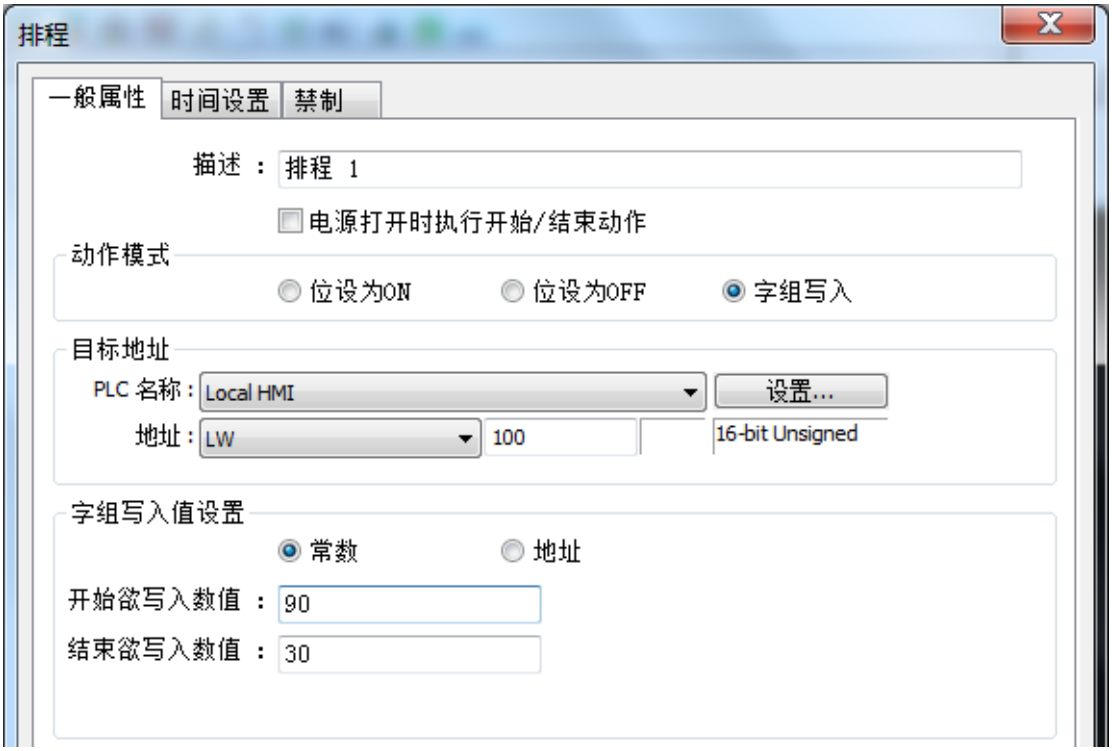
4. 设定“开始”。将时间设为 9 点 0 分 0 秒，接着勾选星期一到星期五，不勾选“设定为单一日期”。
5. 设定“结束”。勾选“启用结束行动”，将结束时间设定为 18 点 0 分 0 秒。
6. 按下“确定”键后，即可看到排程的日程表。

范例 2

从星期一到星期五，在起始时间 8 点把温度设定值 90 度写入字符地址 LW-100，此时系统进入运转模式。在结束时间 17 点把温度设定值 30 度写入字符地址 LW-100，此时系统进入等待模式。

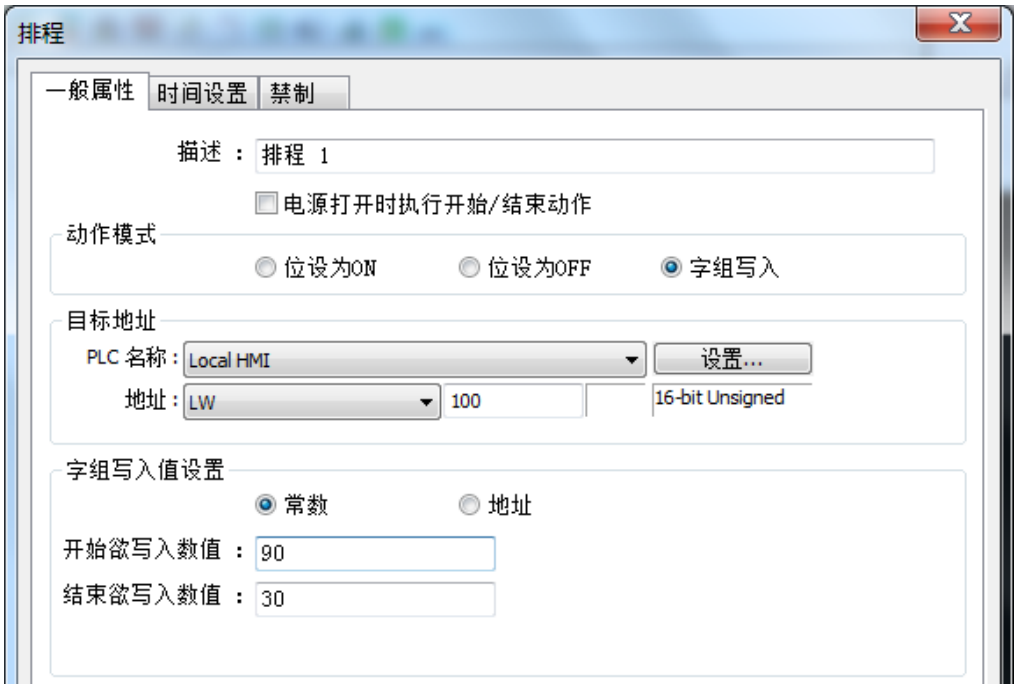
1. 按下工作列上的“排程”按钮后即会出现对话框，按下“新增”键，即可进入排程的设定页。
2. 选择“一般属性”页面，选定“行动模式”为“字组写入”。设定“目标地址”为 LW-100。

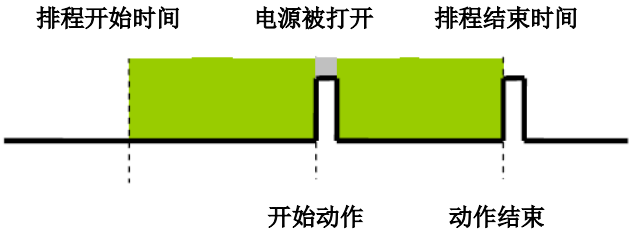
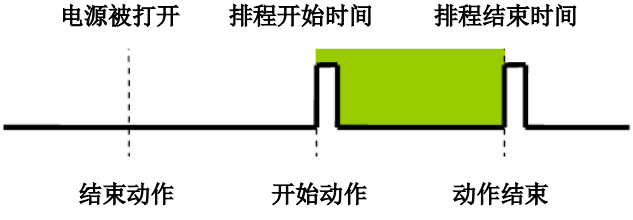
3. 选择“常数”，设定“开始欲写入数值”为 90。



4. 选择“时间设置”页面，接着选择“常数”。
5. 设定“开始”。将时间设为 8 点 0 分 0 秒，接着勾选星期一到星期五，不勾选“设定为单一日期”。
6. 设定“结束”。勾选“启用结束行动”，将结束时间设定为 17 点 0 分 0 秒。
7. 选择“一般属性”页面，设定“结束欲写入数值”为 30。
8. 按下“确定”键后，即可看到排程的日程表。

一般属性设定



设定	描述
电源开启时执行开启 / 结束动作	当电源打开时，执行已设定的动作。 ● 启用时 假如 HMI 的电源在排程区间内被打开，则开始动作会被执行。 假如 HMI 的电源在排程区间外被打开，则结束动作会被执行。 **在排程区间内**  **在排程区间外**  ● 停用时

假如电源打开时晚于排程开始时间，则开始动作不会自动执行。然而结束动作依然会执行。假如结束动作未被设定，由于无法正确地判定排程区间，结束动作将不会被执行。

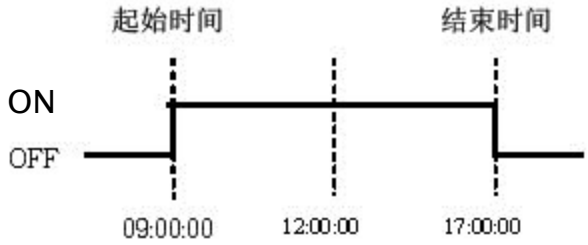
行动模式

选择在设定的时间要操作的类型。

位设为 ON

在排程开始时，将指定位地址的状态设为 ON；在排程结束时，将指定位地址的状态设为 OFF。

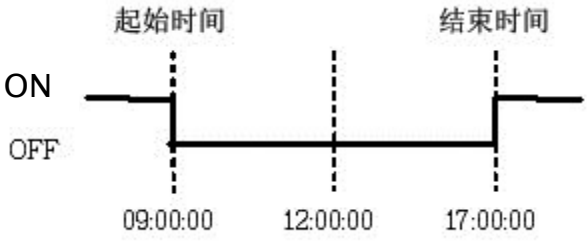
例如：起始时间：09:00:00 结束时间：17:00:00



位设为 OFF

在排程开始时，将指定位地址的状态设为 OFF；在排程结束时，将指定位地址的状态设为 ON。

例如：起始时间：09:00:00 结束时间：17:00:00



字组写入

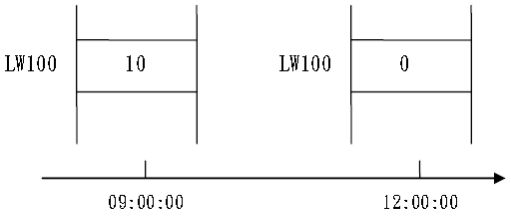
在排程开始时，将“开始欲写入数值”写入指定字符地址；在排程结束时，将“结束欲写入数值”写入指定字符地址。用户可以直接输入常数，或是用“地址”模式设定数值。若使用“地址”模式，则“控制地址”内的数值为开始欲写入的数值，”控制地址 + 1”内的数值为结束欲写入的数值。

例如：字组写入值设定地址：LW-100

起始时间：09:00:00 结束时间：17:00:00

使用常数：开始欲写入数值：10 结束欲写入数值：0

使用地址：若控制地址设定为 LW-n，则在 LW-n 内输入 10，在 LW-(n+1) 内输入 0。

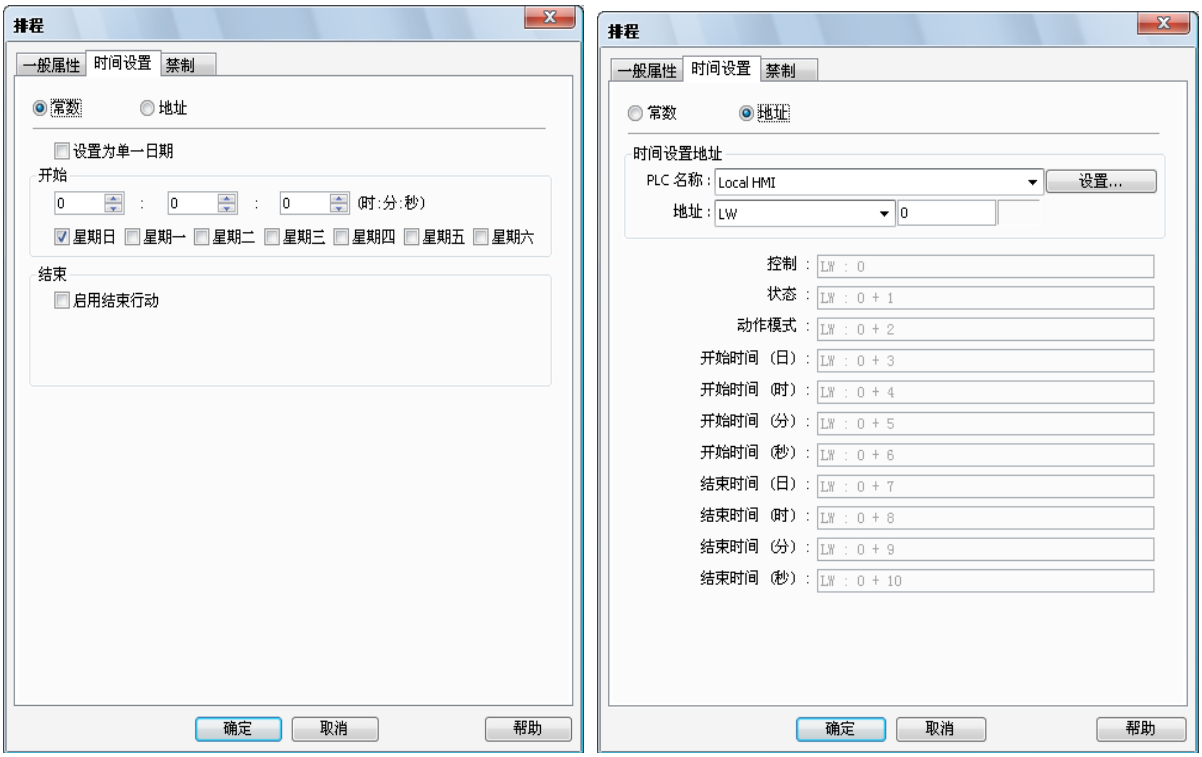


Note

■ 必须在“时间设置”页面中勾选“启用结束动作”才能使用“结束欲写入数值”。

时间设定

选择设定起始时间和结束时间的方法。”常数”可指定一个固定的时间和日期，而“地址”将指定特定地址存储时间和日期的信息。

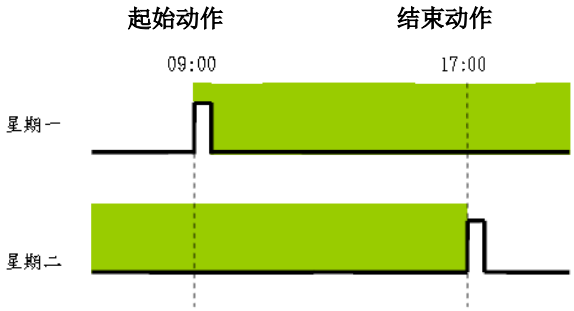


● 常数

“设定为单一日期”

启用时

动作可在一周内指定的日期及时间被执行。当启用后，则必须设定所有日期有相同的动作开始时间及结束时间。

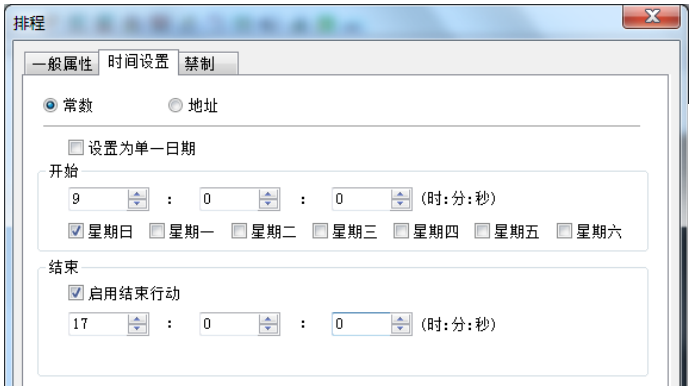


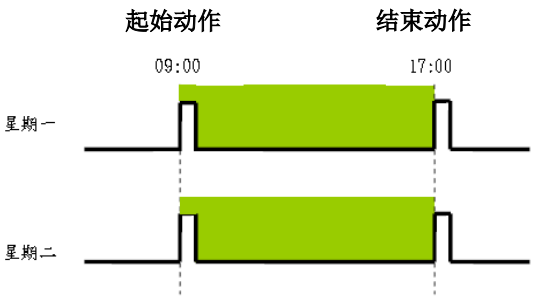
Note

- 必须输入起始时间和结束时间。
- 不能在起始时间和结束时间字段里输入一模一样的时间和日期。

停用时

排程时间必须被限定在一天之内 (起始时间和结束时间必须在 24 小时内)。





Note

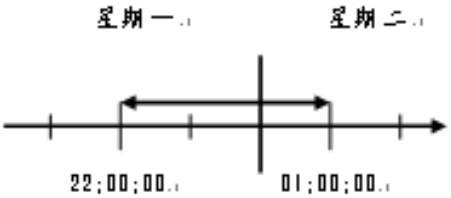
- 不能在起始时间和结束时间字段里输入一模一样的时间和日期。
- 此种时间排程只适用于一天之内的排程，因此如果所键入的结束时间早于起始时间，则结束动作将会等到下一天才会执行。

例如：

起始日期：星期一

起始时间：22:00:00

结束时间：01:00:00



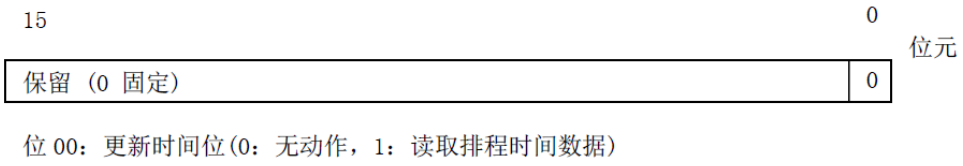
- 地址

开始时间、结束时间、执行命令、执行结果皆由指定地址控制。用户只需定义 “时间设定地址”，其余的 11 个控制字符会自动产生并列示出来。图上之数据长度皆以 16 位为例；当指定之寄存器为 32 位时，只有较低的 16 位产生作用，并请将较高的 16 位固定为 0。

以下说明各地址之使用说明：

控制 (时间设定地址 + 0)

当“更新时间位” (见下图) 被侦测为 ON (0→1) 时，则读出“模式”、“开始时间”和“结束时间”。



位 00：更新时间位 (0：无动作，1：读取排程时间数据)

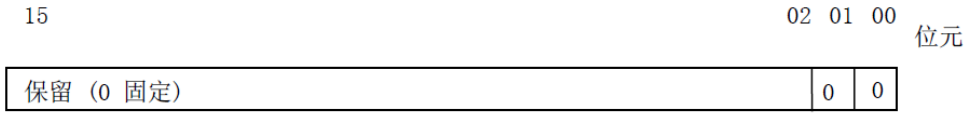
Note

- HMI 并不会定期地读取时间设定地址的“模式” (地址 + 2) 到“结束时间(秒)” (地址 + 10)里的数据。所以，当排程时间数据改变时，请务必把“控制”中的“更新时间位”设为 ON (0→1)。

状态(时间设定地址 + 1)

在“控制”中的时间数据读取完成之后，HMI 将会把“时间读取完成位”设为 ON (0→1)。

同样地，若输入的时间数据不正确，”错误通知位”将会同时被设为 ON (0→1)。

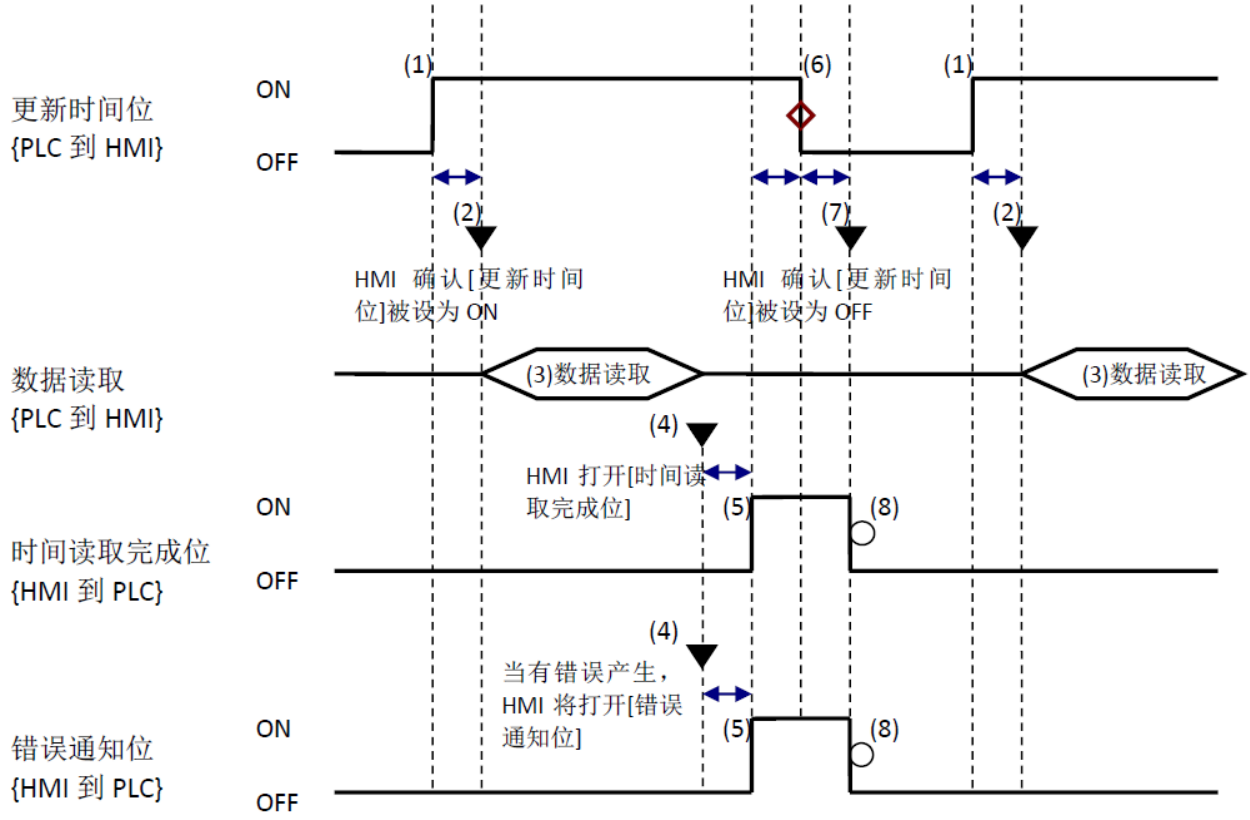


位 00：时间读取完成位 (0：还没开始或是正在读取时间数据；1：时间数据读取完成)

位 01：错误通知位 (0：时间数据被正确更新；1：时间数据中包含错误)

Note

- 一旦发现“时间读取完成位”被触发，请务必把“控制”中的“更新时间位”设为 OFF(1→0)。一旦这个位被设为 OFF(1→0)，则“状态”中的“时间读取完成位”及“错误通知位”将同时被设为 OFF(1→0)。



模式(时间设定地址 + 2)

启用或停用“结束时间动作设置”和“单一日期指定模式”。不管“结束时间动作设置”的状态如何，所有的时间数据（“时间设定地址”中的 11 个字组地址）都会被读取。

15	02	01	00	位元
保留 (0 固定)		0	0	

位 00：结束时间动作设定(0：停用；1：启用)

位 01：单一日期指定模式(0：停用；1：启用)

Note

- 若“结束时间动作设置”输入 0(停用)，仍会读取结束时间数据但忽略其内容。
- 若“单一日期指定模式”输入 1(启用)，请确认是否已输入开始及结束时间信息。假如有 2 个以上的开始/结束日期位被同时设为 ON，则会产生错误。

开始 / 结束日期 (开始日期：时间设定地址 + 3；结束日期：时间设定地址 + 7)

指定触发开始/结束动作的日期。

15 ... 07	06	05	04	03	02	01	00	位元
保留 (0 固定)	Sat	Fri	Thu	Wen	Tue	Mon	Sun	

位 00：星期日(0：无；1：指定)
位 01：星期一(0：无；1：指定)
位 02：星期二(0：无；1：指定)
位 03：星期三(0：无；1：指定)
位 04：星期四(0：无；1：指定)
位 05：星期五(0：无；1：指定)
位 06：星期六(0：无；1：指定)

开始/结束时间 (开始时间：时间设定地址 + 4 到 + 6；结束时间：时间设定地址 + 8 到 + 10)

时：0 – 23 分：0 – 59 秒：0 – 59

假如所指定的值超出上面的范围，将会产生错误。

Note

- 用户所输入的时间数据应为 16 位无正负号 (unsigned) 格式，系统不接受 BCD 格式的时间数据。
- 结束时间取决于 “模式”(地址+2) 设定。同样地，”结束时间动作设置” (位 00)有效与否取决于 “单一日期指定模式” (位 01)的使用。

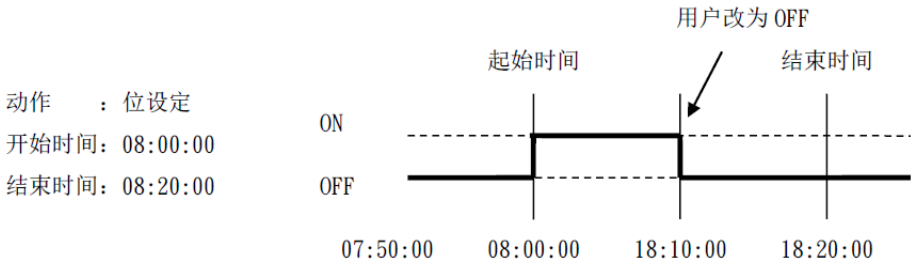
单一日期指定模式	使用	不使用	
结束时间动作设定	使用	使用	不使用

禁制

启用时，在执行开始动作前触摸屏将读取此位状态，若为 ON，则略过此次开始及结束动作(若存在)；反之则正常执行设定动作。

Note

- 最多可注册 32 个“排程”物件。
- 时间排程的特性为一次动作。当开始时间到达时，特定的设备地址只会被写入一次，这个写入的动作将不会重复。



- “开始/结束写入数值”和“禁制位”只会在执行开始动作前读取一次。所以当开始动作执行后，就算再去改变“禁制位”状态或“结束写入数值”都无法改变结束动作的执行与否及写入数值。另外，为了读取“开始/结束写入数值”和“禁制位”数据，起始动作可能因数据通讯而有少许延迟。

- 当用户改变 HMI 的系统时间，系统将会重新确认排程中起始与结束时间的范围。假如编辑的元件位于新范围中，则开始动作会被执行。假如结束动作未被设定，系统无法确认新范围，则这个动作将不会被执行。
- 当相同的起始和结束时间出现在多个排程元件中，将依其编号由小到大顺序被处理。
- 当“时间设置”指定为“地址”，系统将会定期去读取“控制”地址，时间长短视系统忙碌程度而定。
- 当“时间设置”指定为“地址”，且指定开始时间和结束时间超过时间合法的范围，则设定的时间可能不会正确地运作。注意：不能使用 BCD 当成输入值。
- 当“时间设置”指定为“地址”，排程元件直到第一次成功更新时间数据，才开始运作。

13.29 项目选单

13.29.1 概要

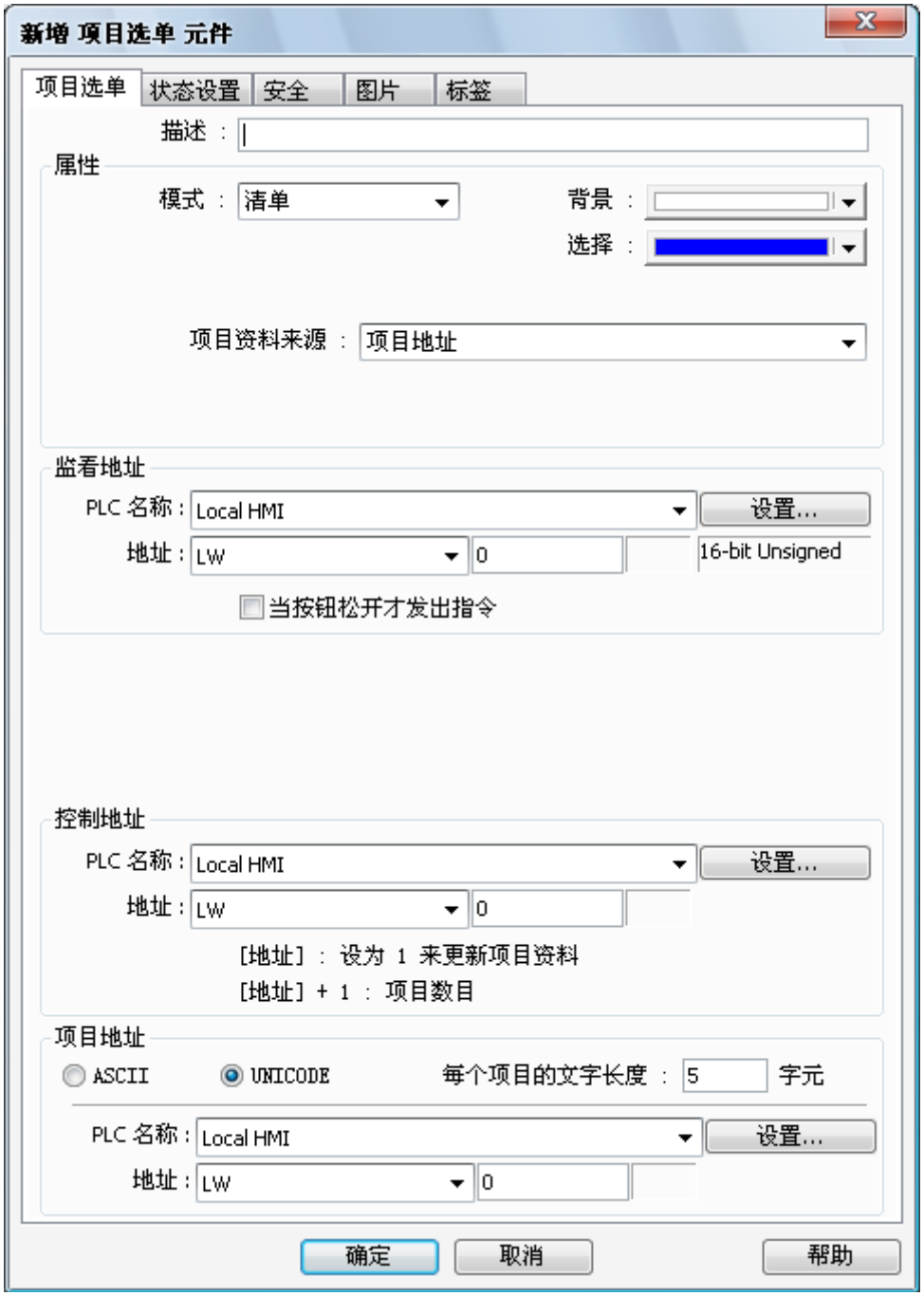
“项目选单”元件可以显示多样项目成一列表，用户可以藉此检视并选择。一旦用户选择了某一项目，相对应的项目数据将被写入到字符寄存器。

“项目选单”有两种显示模式：”清单”和“下拉式选单”。清单可以完整显示所有的项目，并把目前所选择的项目标示出来。此外，下拉式选单在一般情况下只显示目前所选择之项目。但是当用户点选下拉式选单时，系统则会列出所有完整项目(类似于清单的显示法) 如下所示：

13.29.2 设定

按下工作列上的“项目选单”按钮后即会开启“项目选单”元件属性对话框，正确设定各项属性后按下确认键，即可新增一个“项目选单”元件。

项目选单设定



设定	描述
属性	**模式：**可选择“清单”或“下拉式选单”。 **项目数：**设定需使用的项目数。每个项目代表一个状态，当某一项目被选择时，会将相对应的数据写入“监看地址”中。 **背景：**设定背景的颜色。 **选择：**设定选择的项目的背景颜色。 **项目数据来源：**项目选单显示的内容，共有四种模式：预设、历史数据日期、

	项目地址模式、用户账号。请见《13.29.2.1 项目数据来源说明》。
监看地址	系统会将已选择项目的相对应数据写入“监看地址”中。 **当按钮松开才发出指令** 若启用，当按钮松开时才会将指定的数据写入“监看地址”中。
写入成功后传送通知	当写入 PLC 的动作成功后，将指定位寄存器的状态设为开 / 关。

Note

- cMT-SVR 系列的项目数据来源没有“历史数据日期”，及“当按钮松开才发出指令”选项。

项目数据来源说明

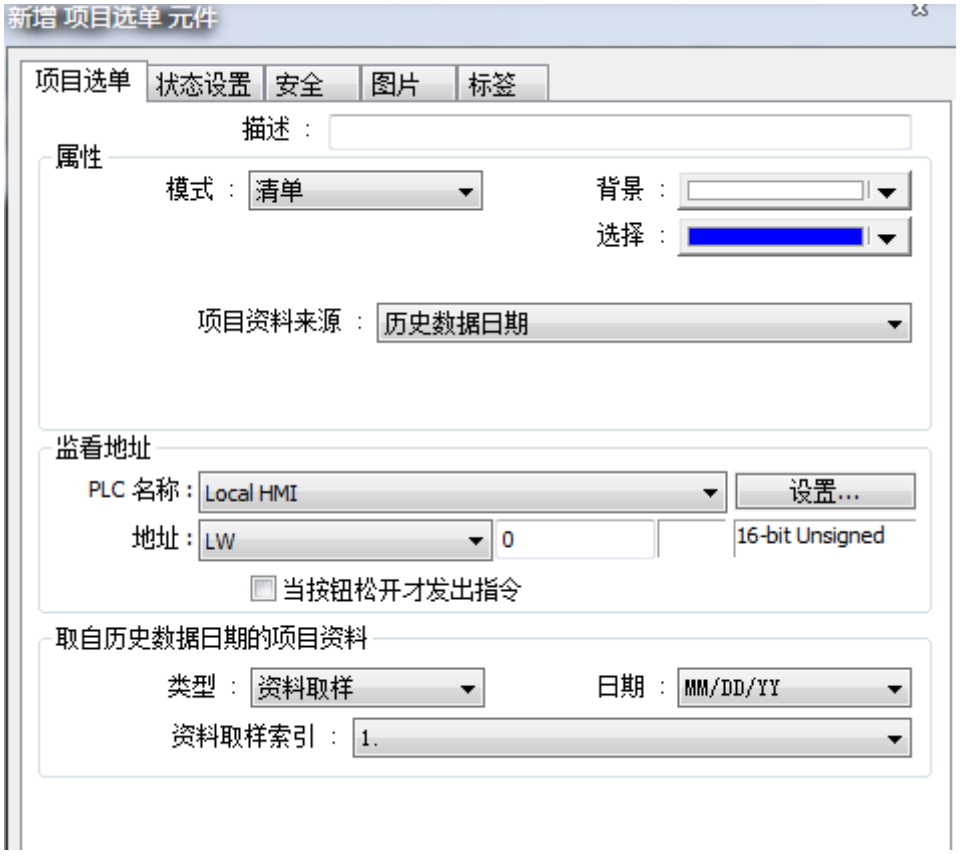
- **预设**

显示的选项由用户在“状态设置”分页中手动输入。

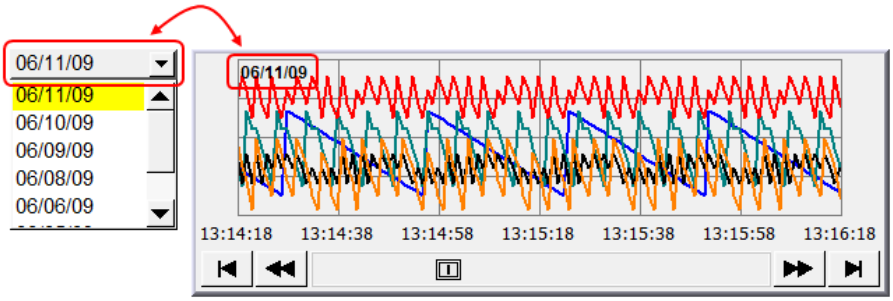
可调整欲使用的“项目数”。每一个项目表示一个状态并会显示在列表上，且相对应的数值可被写入至“监看地址”。

- **历史数据日期**

此功能 cMT-SVR 系列不支持。



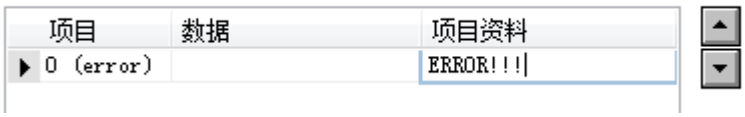
可与历史数据显示元件搭配使用，例如“趋势图”、”历史数据显示”、”事件显示”元件。当在项目选单上选择一日期后，历史数据显示元件会显示其对应日期的数据。显示方式如下图所示：



设定	描述
类型	可选择“事件登录”或是“资料取样”。
日期	共有 8 种日期模式可选择，YYYY 代表四位数年份 (例如 2012)、YY 代表二位数年份 (例如 12)、MM 代表月份、DD 代表日期。
资料取样索引	若“类型”选择“资料取样”，需在“资料取样索引”设定欲显示的资料取样元件。一般来说，选择与之搭配的趋势图的历史模式或历史数据元件相同即可。

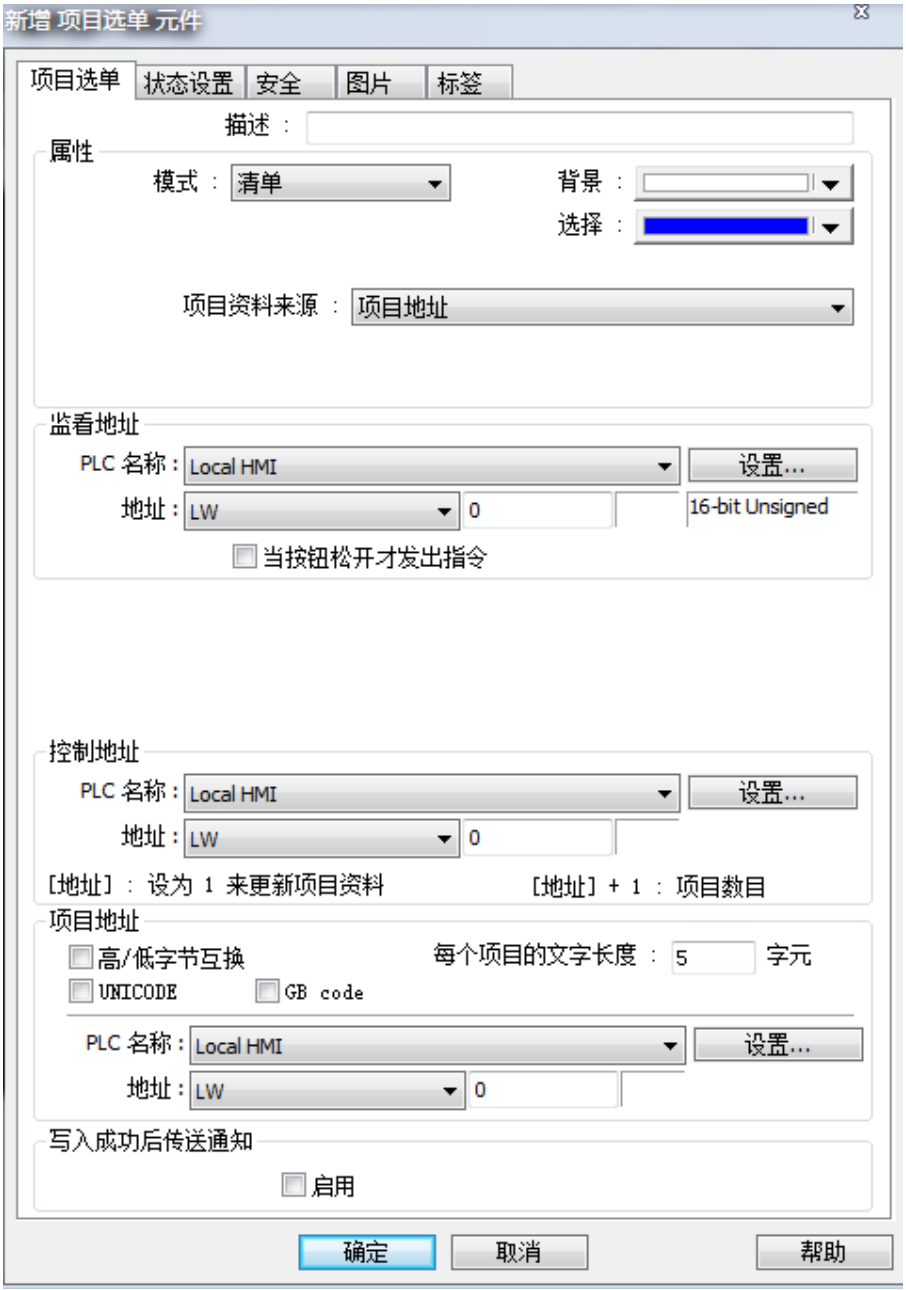
Note

- 当选用历史数据来源(历史数据日期)，由于系统将自动读取历史文件并产生日期数据，故状态设定中用户不需再填写。
- 用户可以在状态设定中设定当项目选单进入错误状态时，项目选单将显示的数据。



- **项目地址模式**

“项目地址”模式可加载“项目地址”的文字并将其文字显示于项目选单上。当选择“项目地址”模式后，下方会出现“控制地址”和“项目地址”，如下图所示：



设定	描述
控制地址	**“地址”**：若将此地址所指定的寄存器中数据设定为 1，将更新元件所显示的项目为“项目地址”的内容。更新完成后寄存器中的数据会被恢复为 0。 **“地址”+ 1**：此地址中的数据用来设定项目的个数。
项目地址	设定用来存放项目内容的起始地址。 **UNICODE** 选项的内容使用 UNICODE 文字，例如中文字。

	每个项目的文字长度 每一个项目的文字长度，单位是字符。

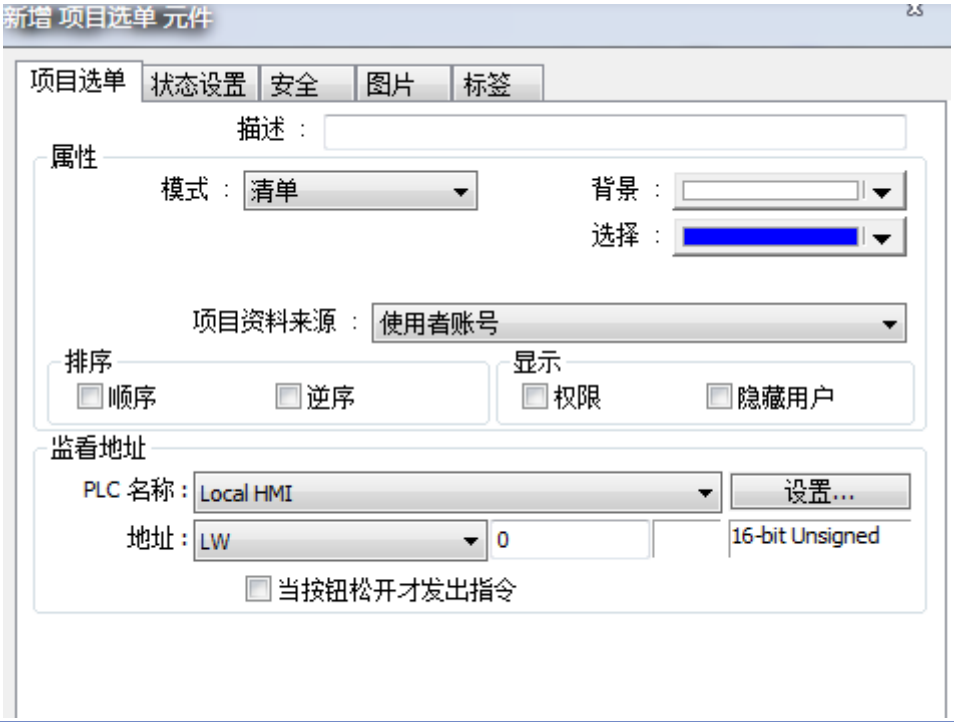
Note

- 使用到的 UNICODE 文字必须先用“文字”元件输入好，EasyBuilder Pro 才会预先编译所需要的字型文件，在下载时一并存放到 HMI 上，如此才能正确显示 UNICODE 文字。
- (项目的个数) x “每个项目的文字长度”不得超过 1024。

- 若选择“项目地址”模式，系统将自动取消“状态设置”页面。

● **用户账号**

当启用“进阶安全模式”后，则数据来源将有“用户账号”选项。此时的项目选单会显示用户的名称。



设定	描述
排序	决定用户账号的排序方式。
显示	若勾选“权限”，将显示各个用户的使用权限。若勾选“隐藏用户”，将显示已隐藏的用户数据。

Note

- 用户索引的地址为“进阶安全模式” » “用户密码” » “进阶安全模式”中的“控制地址+2 (LW-n + 2)”。

状态设定

此设定页显示所有状态的项目、文字和数值，如果要改变项目数，请点选“项目选单”»“属性”»“项目数”。

设定	描述
项目	系统会列出目前所有使用的项目，每一个项目表示一个状态并且会显示在列表。此栏为只读。
数据	用户可为每个项目设定数值，但须注意： 读取监看地址：如果系统侦测到“监看地址”的数值有改变，元件会根据其数

	值并选择第一个吻合的项目。如果没有项目吻合，将跳至错误状态；如果已经设定错误通知位，该位会被触发。 写入监看地址：当用户选择某项目，系统将数据写入至“监看地址”。
项目数据	用户可为每个项目设定显示文字，项目选单元件将显示所有项目的文字在列表上供用户检视和选择。
错误状态	错误状态的文字只能应用于“下拉式选单”模式，”清单”模式无法使用错误状态文字。 在错误状态发生时，”清单”模式将不会选取任一个项目来表示错误状态，而“下拉式选单”模式则会显示错误状态的文字。 例如，当“项目数”设 8 时，项目编号 8 即为错误状态 (因为第 1 个项目是编号 0)。
设为默认值	将所有项目数据还原为默认值，例如：将项目 0 的数据还原成 0，项目 1 的数据还原为 1…等等。
错误通知	当项目选单侦测到不合法的数据写入时，会触发指定寄存器的状态“开”或“关”以示警告。此外，用户亦可搭配其它元件，例如“事件登录”、”报警条”、”弹出窗口”来提示错误。

13.30 定时器

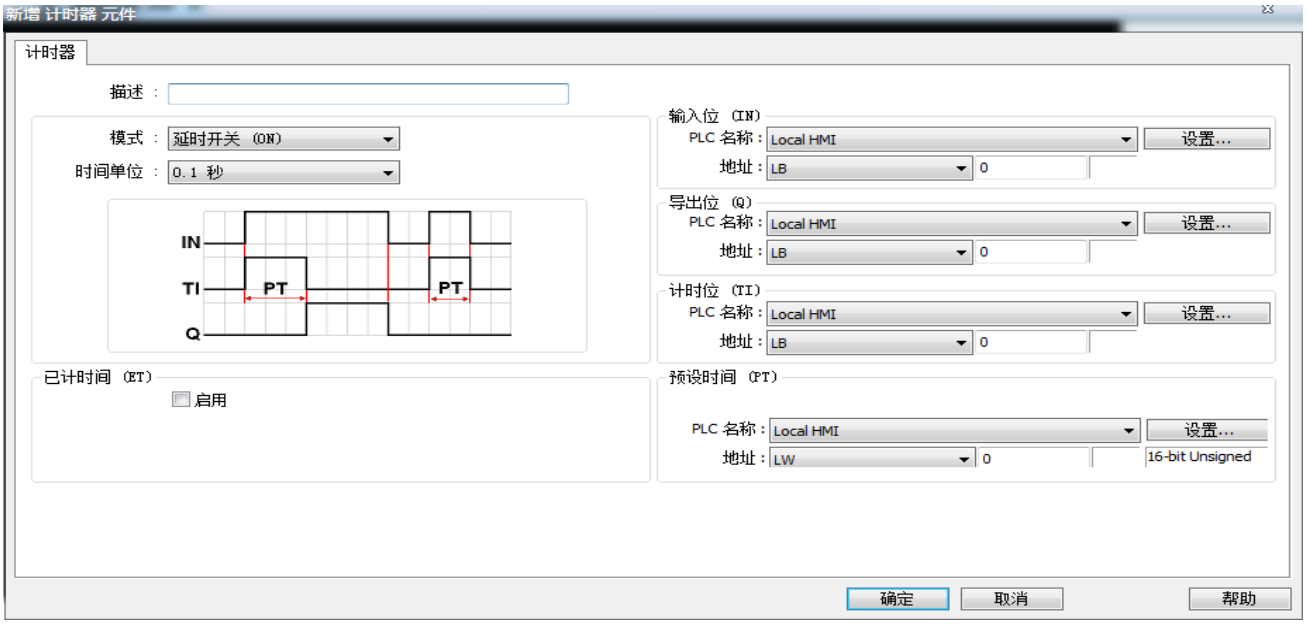
13.30.1 概要

定时器相当于一计时开关，可做为延时开关、脉冲开关及累加式延时开关，其包含下列六项变量：

定时器变数	变量类型	描述
输入位 (IN)	位变量	定时器的总开关
测量位 (TI)	位变量	计时开始时设 ON
输出位 (Q)	位变量	计时结束后启动相关设定
预设时间 (PT)	字符变量	设定定时器时间数值
已计时间 (ET)	字符变量	显示定时器目前已计时间
重置位 (R)	位变量	将目前定时器已计时间 (ET) 归零

13.30.2 设定

点选“定时器”图示，其“定时器”元件属性对话框显示如下。

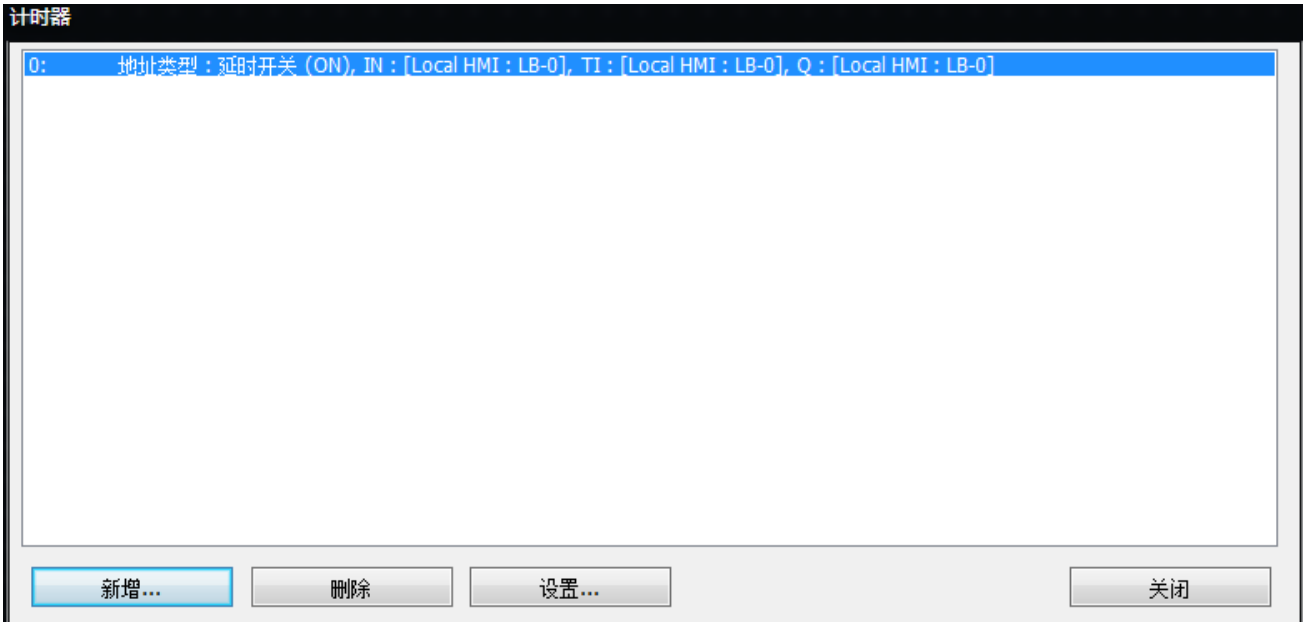


Note

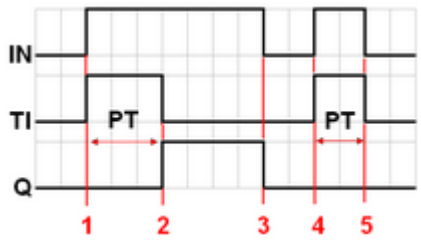
- “使用常数设定预设时间”仅适用于 cMT-SVR 系列。

若使用 cMT-SVR 系列，点选“定时器”图示，会先开启定时器管理窗口。

点选“新增”可建立“定时器”元件。

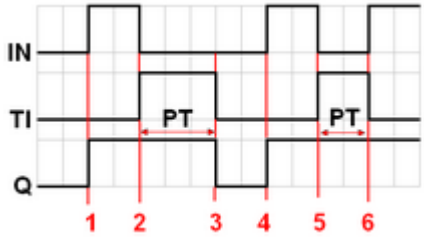


- **延时开关 (ON)**

电位图	寄存器
	**输入位 (IN)**：定时器的总开关。 **测量位 (TI)**：计时开始设 ON。 **输出位 (Q)**：计时结束后设 ON。 **预设时间 (PT)**：设定定时器时间数值。 **已计时间 (ET)**：显示定时器目前已计时间。

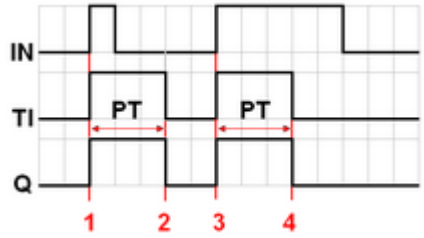
说明 (参照上图)
时段 1：输入位 IN 设 ON 时，测量位 TI 开启，已计时间 ET 开始计数，输出位 Q 保持 OFF。 **时段 2**：当已计时间 ET 等于预设时间 PT 时，测量位 TI 被关闭，同时输出位 Q 被开启。 **时段 3**：输入位 IN 设 OFF 时，输出位 Q 被关闭，同时已计时间 ET 归零。 **时段 4**：输入位 IN 设 ON 时，测量位 TI 被开启，已计时间 ET 开始计数，输出位 Q 保持 OFF。 **时段 5**：在已计时间 ET 到达预设时间 PT 之前，将输入位 IN 设为 OFF，测量位 TI 将被关闭，同时已计时间 ET 归零。因为 ET 仍小于 PT，输出位 Q 保持在 OFF。

- 延时开关 (OFF)

电位图	寄存器
	**输入位 (IN)：** 定时器的总开关。 **测量位 (TI)：**计时开始设 ON。 **输出位 (Q)：**计时结束后设 OFF。 **预设时间 (PT)：**设定定时器时间数值。 **已计时间 (ET)：**显示定时器目前已计时间。

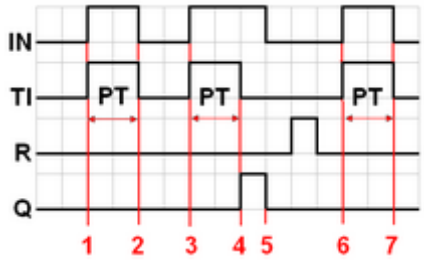
说明 (参照上图)
时段 1：输入位 IN 设 ON 时，测量位 TI 保持 OFF，输出位 Q 被开启，已计时间 ET 归零。 **时段 2：**输入位 IN 设 OFF 时，测量位 TI 被开启，输出位 Q 保持 ON，已计时间 ET 开始计数。 **时段 3：**当已计时间 ET 等于预设时间 PT 时，输出位 Q 和测量位 TI 被关闭。 **时段 4：**输入位 IN 设 ON 时，测量位 TI 保持 OFF，输出位 Q 被开启，已计时间归零。 **时段 5：**当输入位 IN 设 OFF 时，测量位 TI 被开启，输出位 Q 保持 ON，已计时间 ET 开始计数。 **时段 6：**当在已计时间 ET 到达预设时间 PT 的数值前设输入位 IN 为 ON，测量位 TI 被关闭，同时输出位 Q 保持 ON，已计时间 ET 归零。

- 脉冲启动开关

电位图	寄存器
	**输入位 (IN)：**定时器的总开关。 **测量位 (TI)：**计时开始设 ON。 **输出位 (Q)：**计时开始设 ON；计时结束后设 OFF。 **预设时间 (PT)：**设定定时器时间数值。 **已计时间 (ET)：**显示定时器目前已计时间。

说明 (参照上图)
时段 1：当输入位 IN 设 ON 时，测量位 TI 和输出位 Q 同时被开启，已计时间 ET 开始计数。 **时段 2：**当已计时间 ET 等于预设时间 PT 时，输出位 Q 和测量位 TI 同时被关闭。(因为在计数同时已先将输入位 IN 设 OFF，所以已计时间 ET 将被自动归零。) **时段 3：**当输入位 IN 设 ON 时，测量位 TI 和输出位 Q 同时被开启，已计时间 ET 开始计数。 **时段 4：**当已计时间 ET 等于预设时间 PT 时，输出位 Q 和测量位 TI 同时被关闭。

- **累加式延时开关 (ON)**

电位图	寄存器
	**输入位 (IN)：**定时器的总开关。 **测量位 (TI)：**计时开始时设 ON。 **输出位 (Q)：**计时结束后设 ON。 **预设时间 (PT)：**设定定时器时间数值。 **已计时间 (ET)：**显示定时器目前已计时间。 **重置位 (R)：**将目前已计时间 (ET) 归零。

说明 (参照上图)

时段 1：当输入位 IN 设 ON 时，测量位 TI 被开启，已计时间 ET 开始计数，输出位 Q 保持 OFF。

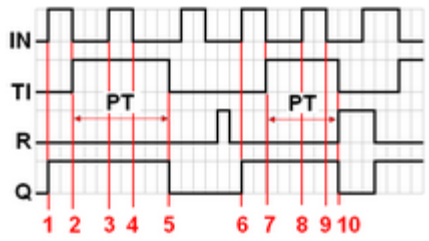
时段 2：当输入位 IN 设 OFF 时，如果已计时间 ET 未到达预设时间 PT，测量位 TI 被关闭，同时输出位 Q 保持 OFF。已计时间 ET 保持现在的状态数值。

时段 3：当输入位 IN 再度设 ON 时，测量位 TI 被开启，同时已计时间 ET 再次由刚刚保持的状态数值开始计数，同时输出位 Q 保持 OFF。

时段 4：当已计时间 ET 等于预设时间 PT 时，测量位 TI 被关闭，同时输出位 Q 被开启。

时段 5：设输入位 IN 为 OFF，同时输出位 Q 也被关闭。(此时设重置位 ON 使已计时间 ET 归零后再设为 OFF。)

- **累加式延时开关 (OFF)**

电位图	寄存器
	**输入位 (IN)：**定时器的总开关。 **测量位 (TI)：**计时开始时设 ON。 **输出位 (Q)：**计时结束后设 OFF。 **预设时间 (PT)：**设定定时器时间数值。 **已计时间 (ET)：**显示定时器目前已计时间。 **重置位 (R)：**将目前已计时间 (ET) 归零。

说明 (参照上图)

时段 1：当输入位 IN 设 ON 时，测量位 TI 保持 OFF，同时输出位 Q 被开启。

时段 2：当输入位 IN 设 OFF 时，测量位 TI 被开启，同时输出位 Q 保持 ON。已计时间 ET 开始计数。

时段 3：当输入位 IN 再度设 ON 时，测量位 TI 和输出位 Q 保持 ON，同时已计时间 ET 暂停计数。

时段 4：当输入位 IN 再度设 OFF 时，已计时间 ET 再次由刚刚保持的状态数值开始计数。

时段 5：当已计时间 ET 等于预设时间 PT 时，测量位 TI 和输出位 Q 同时被关闭。(此时设重置位 ON 使已计时间 ET 归零后再设为 OFF。)

13.31 影像输入

13.31 概要

HMI 提供影像输入功能，用户加装监视镜头后，通过监视镜头即可实时监看现场状况，也能将画面记录到储存装置并且在计算机上做分析。

此功能可应用在各个层面，除了可监看现场状况外，也能应用在行车装置或是大楼监控。

在硬件上，触摸屏提供 2 个影像输入通道，用户可自由切换所欲监控的画面，而且影像撷取功能不受暂停播放控制之影响，所撷取的图片仍是外部影像输入之实时画面。

此功能支持 NTSC 及 PAL 二种影像格式。

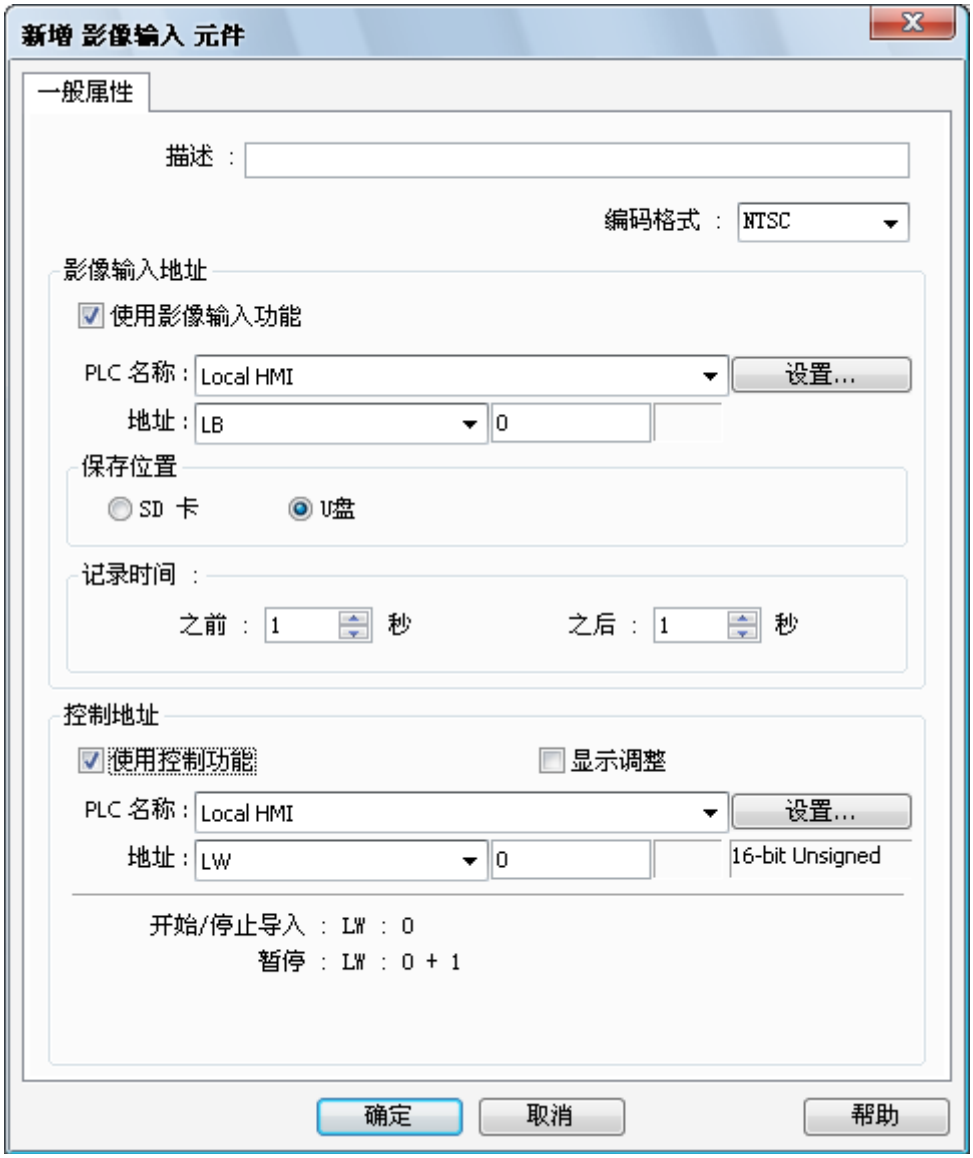
Note

- 仅 eMT3120A / eMT3150A 支持此功能。

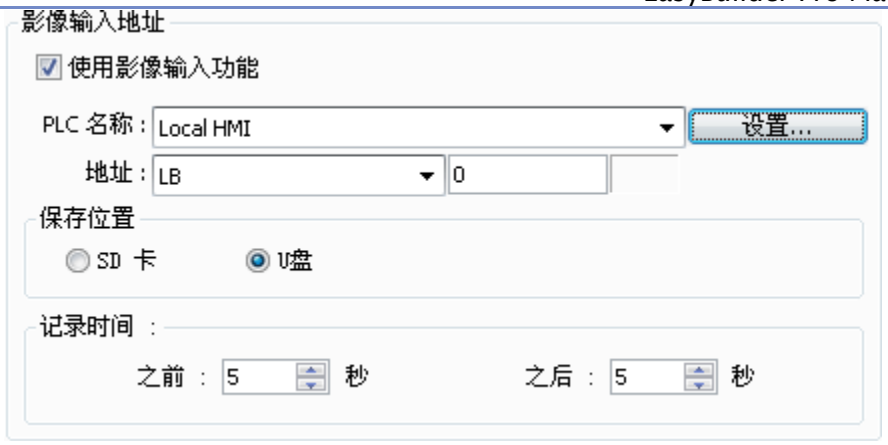
13.31.2 设定

按下工作列上的“影像输入”按钮后即会开启“影像输入”元件属性对话框，正确设定各项属性后按下确认键，即可新增一个“影像输入”元件。

一般属性设定



设定	描述
输入通道	可选择使用影像输入通道 1 或影像输入通道 2。
编码格式	可选择 NTSC 或 PAL 讯号。
影像撷取地址	勾选“使用影像撷取功能”来设定撷取输入影像画面之功能。



影像撷取地址

触发系统撷取图片的控制地址。

储存空间

选择图片储存媒体为 SD 卡或 U 盘。输入通道 1 的影像撷取将储存在外部装置的 VIP1 文件夹中，输入通道 2 则储存在 VIP2 文件夹。

记录时间

设定撷取画面之时间范围。

- 最大撷取范围为“影像撷取地址”触发时之前后 10 秒。
- 系统每秒撷图一次。
- 图片文件命名规则：

“影像撷取地址”触发前后： YYYYMMDDhhmmss.jpg

“影像撷取地址”触发当下：YYYYMMDDhhmmss@.jpg

以上图为例，设定记录时间为前后 5 秒，当“影像撷取地址”的状态由 OFF 转为 ON 时，系统将从触发时间点起算，每秒 1 张，撷取前后 5 秒之输入画面，共 11 张图。

控制地址

使用控制功能

启用后，将特定的数值写入“控制地址”及连续的寄存器可控制影像输出。假设控制地址设定为 LW-n (n 为任意地址)，将特定数值写入指定地址的执行命令如下表所示。

地址	数值	命令内容
LW-n	0	停止播放影像
	1	开启输入通道 1 的影像并显示于屏幕上
	2	开启输入通道 2 的影像并显示于屏幕上
	3	开启输入通道 1 的影像但不显示 (仍可执行影像撷取功能)
	4	开启输入通道 2 的影像但不显示 (仍

		可执行影像撷取功能)
LW-n+1	1	暂停/继续播放影像
LW-□+2	1~100	对比调整
L□n+3	1~100	亮度调整

- 在变更“控制地址 (LW-n)”的值后，系统将保留变更后的值。
- 在变更“控制地址 + 1 (LW-n+1)”的值后，系统将在执行对应命令结束后将其清除为 0。
- 若不启用“使用控制功能”，系统将自动播放“输入通道”指定之影像输入。
- 勾选“显示调整”后，才可调整对比及亮度。

Note

- “影像输入”元件仅可使用于 eMT 系列中的 eMT3120A / eMT3150A。
- 系统中任何时间点只能有一组影像输入通道开启影像输入。
- 影像撷取功能不受暂停播放控制之影响，所撷取的图片仍是外部影像输入之实时画面。
- 推荐的格式类型和分辨率：

	1:1	50%
NTSC	720 x 480	360 x 240
PAL	720 x 576	360 x 288

13.32 系统讯息

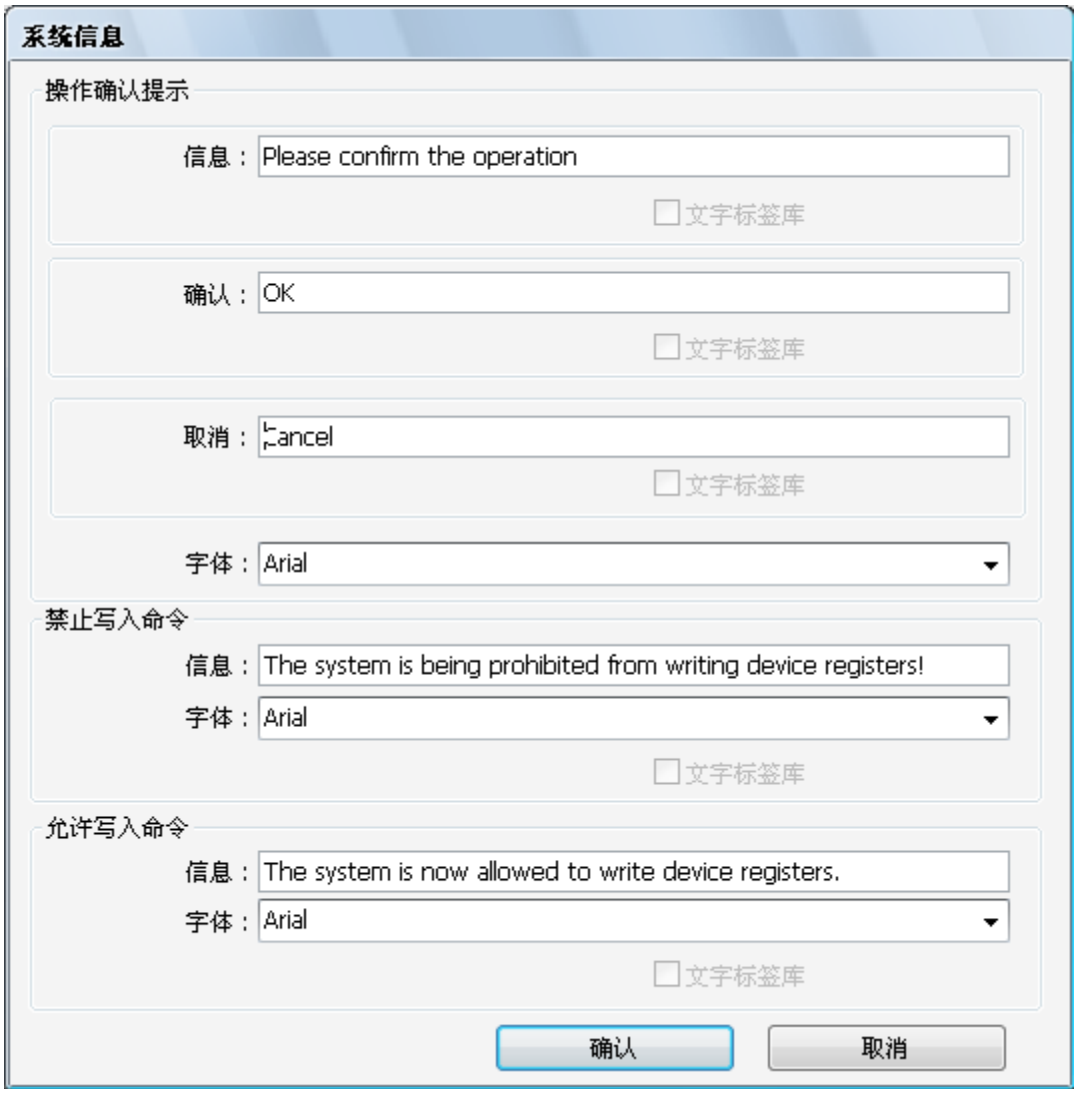
13.32.1 概要

当元件选择被使用前需弹出确认窗口或是否可使用远程登入时，系统会先弹出讯息窗口：“操作确认提示”、”禁止写入命令”、”允许写入命令”的讯息，此三项讯息的内容可在“系统讯息”中编辑。

13.32.2 设定

按下工作列上的“系统讯息”按钮 后即会开启“系统讯息”对话框，可以设定系统讯息。

系统讯息设定



设定	描述
窗口尺寸	选择提示的窗口和字体尺寸。
操作确认提示	要操作受保护的元件时，显示讯息向用户确认这项操作。您可以设定“操作确认提示”中的讯息与“确认”、”取消”两个按钮上的文字。“确认”与“取消”两个按钮上的文字，需使用相同的字型。另外，只有在“讯息”选择使用文字标签库时，”确认”与“取消”两个按钮才允许使用文字标签库。
禁止写入命令	当系统寄存器 LB-9196 (本地触摸屏只支持检视功能) 设为 ON 时，显示此讯息。
允许写入命令	当系统寄存器 LB-9196 (本地触摸屏只支持检视功能) 设为 OFF 时，显示此讯息。

Note

- cMT-SVR 系列不支持调整提示窗口尺寸及使用 LB-9196 设定远程控制功能。

13.33 配方检视

13.33.1 概要

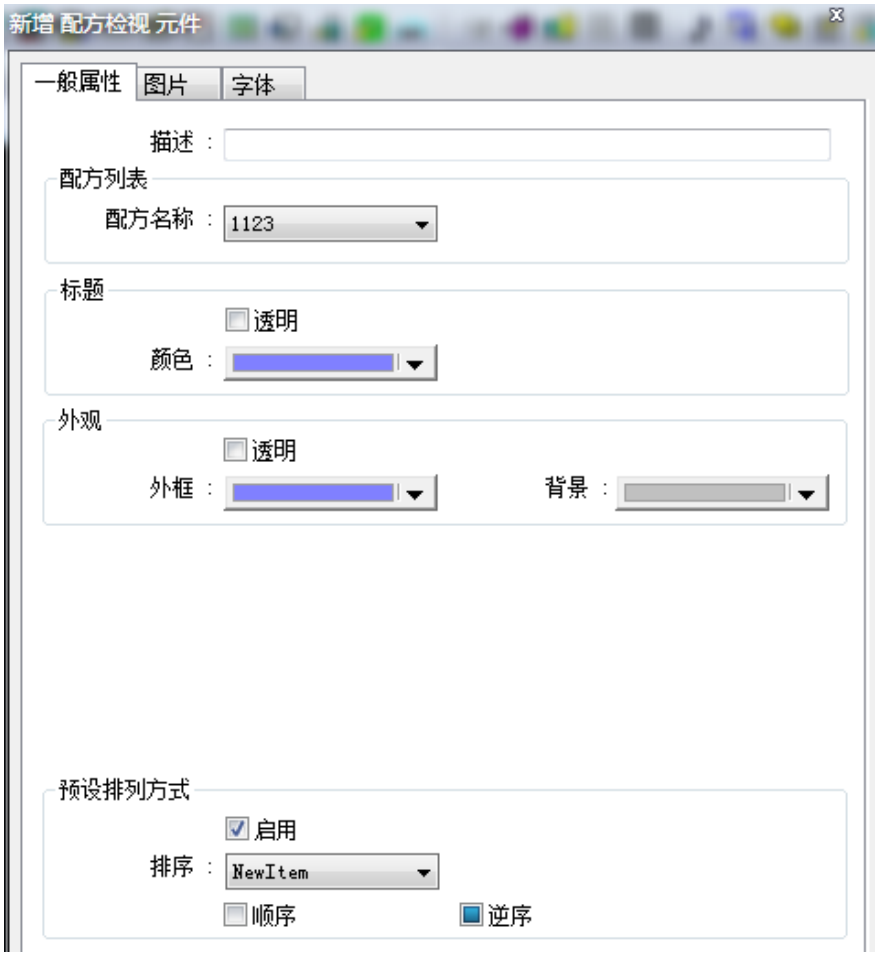
“配方检视”可用来检视特定的配方数据，用户可在“配方检视”上观察到该笔配方的所有项目及数值。

13.33.2 设定

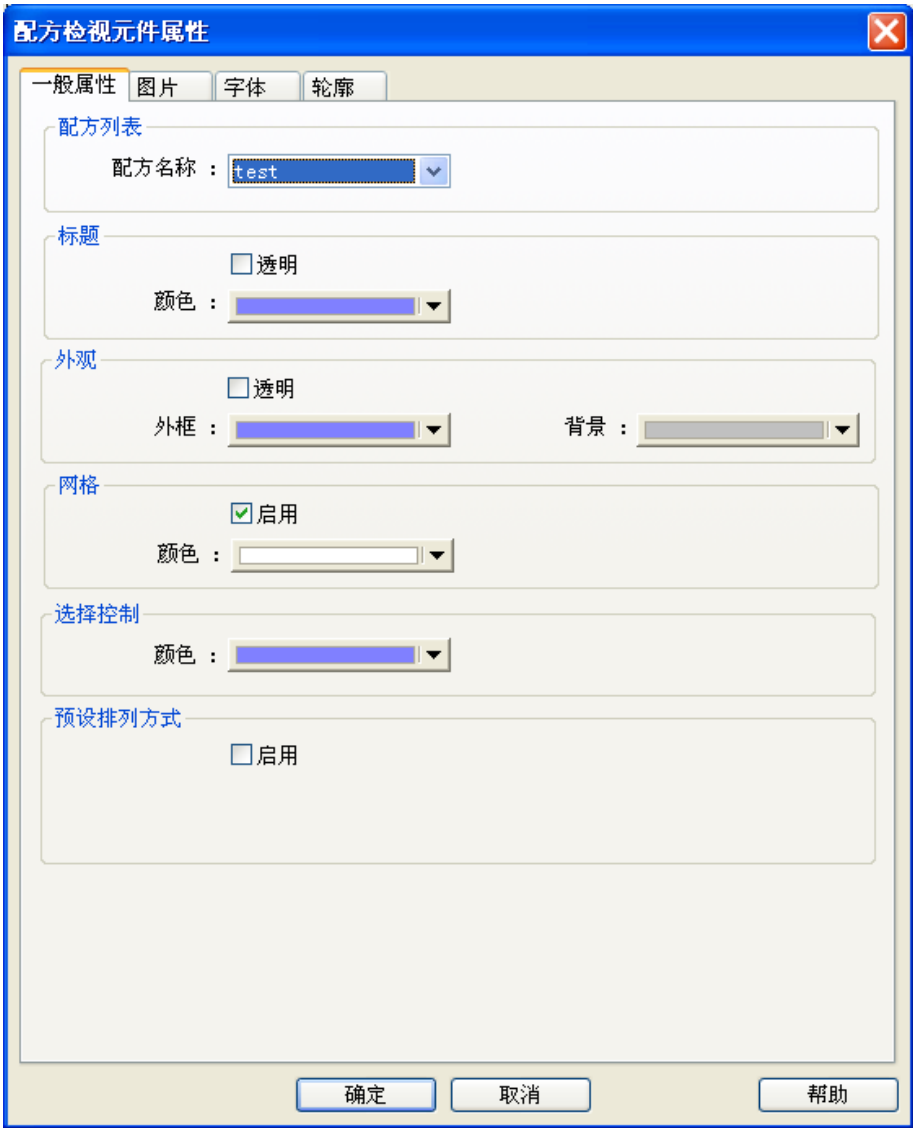
按下工作列上的“配方检视”按钮后即会出现“配方检视”元件属性对话框，正确设定各项属性后按下确认键，即可新增一个“配方检视”物件。

一般属性设定

cMT-SVR 系列



eMT、iE、cMT-HD 系列



“配方检视”各部份的名称请参考下图：

设定	描述
配方列表	选择欲检视的配方名称，可点选下拉式选单寻找其它配方。
标题	每个项目的标题，根据“系统参数设置”»“配方”中的设定。 **透明** 若勾选，则标题就不会有背景颜色，并且不会出现“颜色”的选项。
外观	元件的边缘线颜色及背景颜色。 **透明** 若勾选，则元件就不会有背景颜色，并且不会出现“颜色”的选项。
网格 (cMT-SVR 不适用)	区分配方每个数据的间隔线。 **透明** 若勾选，则网格就不会有颜色，并且不会出现“颜色”的选项。
选择控制 (cMT-SVR 不适用)	当点选到特定一行的数据时，所显示的颜色。
预设排列方式	设定“配方检视”的排序方式。提供“顺序”与“逆序”两种排序方式。

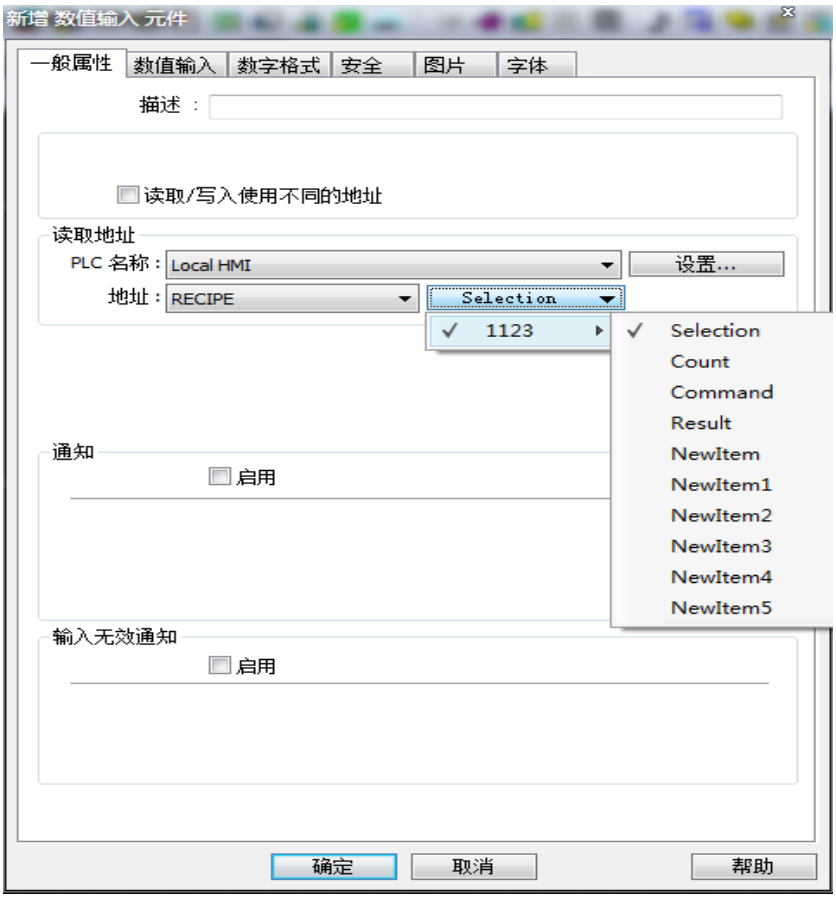
Note

- 可使用四个系统寄存器来查询/更新/增加/删除配方数据库的内容：

Selection

目前所选择的配方编号。编号为从 0 开始计算，因此若点选第一笔，则 Selection 的数值会显示 0，依此类推。

当“Selection”的值改变时，相对应的寄存器也会跟着改变，例如下图中的 No., Timer_1, Timer_2。



Count

目前配方中的资料笔数。

Command

输入特定的数值可对选取的配方数据下执行命令。(数字为执行的命令数值)

输入 ”1” 将新的配方数据新增到最后

输入 “2” 将更新目前选择的配方资料

输入 “3” 将删除目前选择的配方资料

输入 “4” 将删除所有配方数据

Result

可监看命令的执行结果。(数字为执行命令后的结果数值)

数值 ”1” 代表命令成功执行

数值 “2” 代表该笔配方不存在

数值 “4” 代表未知的命令

数值 “8” 代表配方已达上限(10000 笔)，无法新增

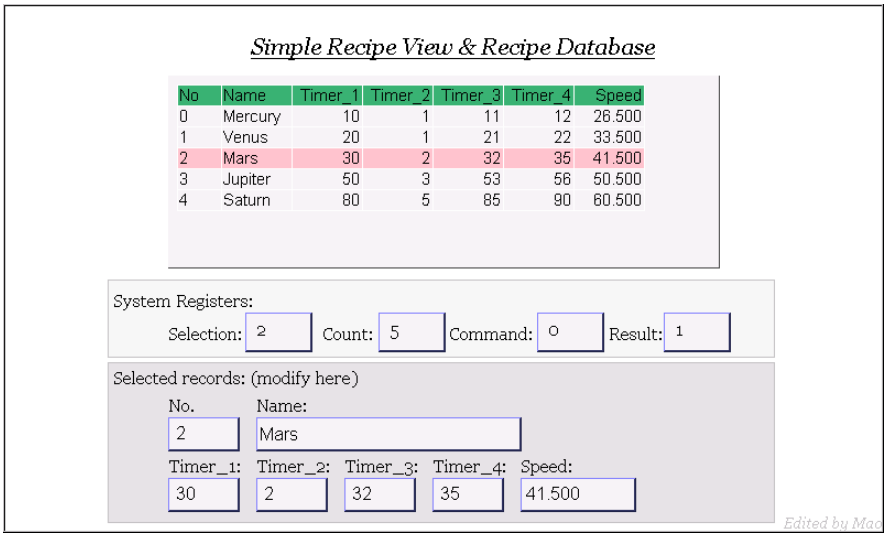
使用此功能需要至 “系统参数设置” » “配方” 建立配方数据，请参考《5 系统参数设定》。

配方记录的建立请参考《24 Recipe Editor》。

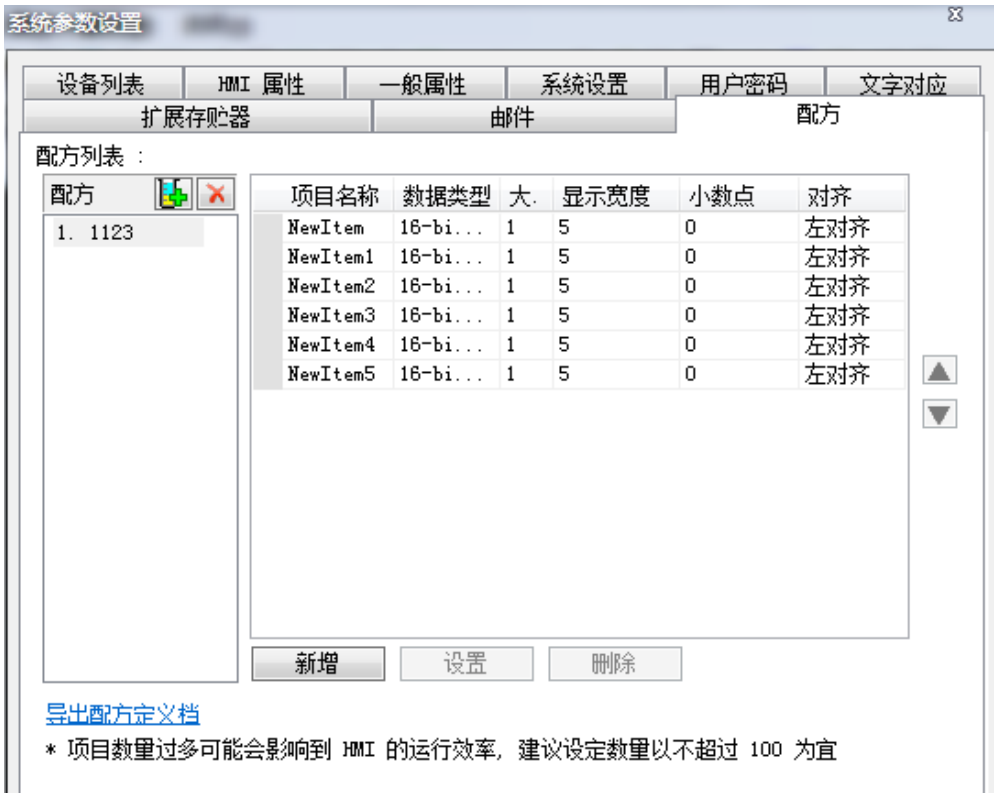
范例 1

以下示范“配方检视”和配方数据库的简易使用方法。

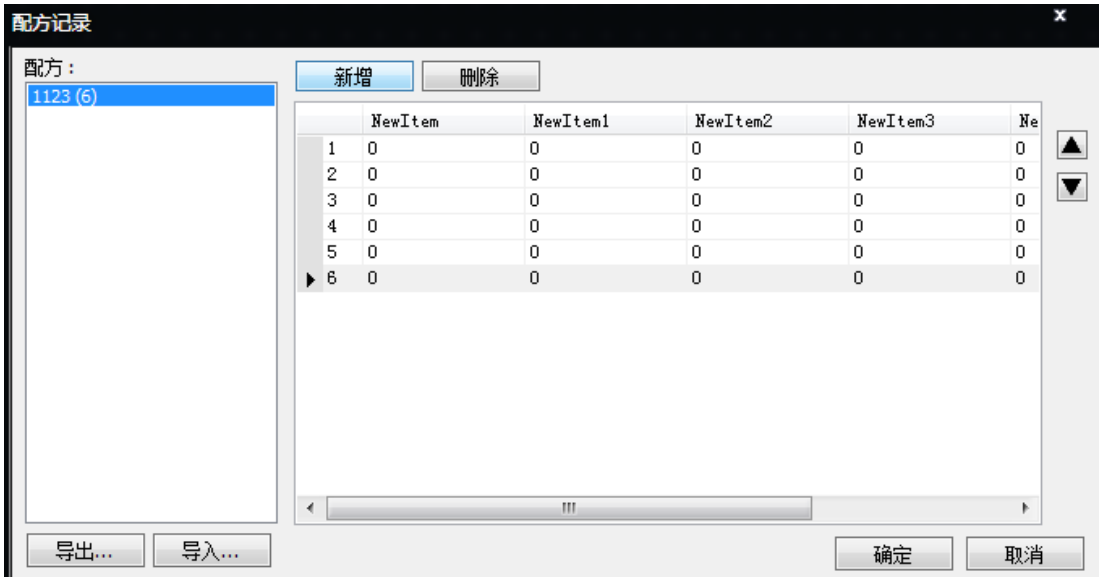
在范例中会建立配方数据库，并且使用“配方检视”来检视和选择配方数据。当您点选了“配方检视”中的任一笔记录时，”Selection”及对应的寄存器数值都会改变。最后，可以使用“Command”来编辑修改配方数据库。



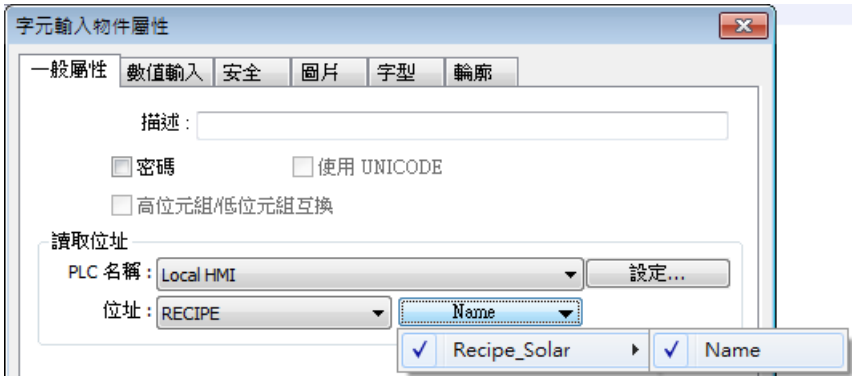
1. 建立一个配方，如下图：



2. 使用“配方记录”增加任意数笔的数据，或如下图：



3. 新增一个“配方检视”元件并使用刚建立的配方数据库。
4. 建立四个使用“Selection”、”Count”、”Command”、”Result”的“数值输入”元件。
5. 建立 No, Name, Timer_1, Timer_2, ..., Timer_4, Speed 项目对应的元件。例如使用“字符输入”并设定地址为“Recipe”中的“Name”，如下图所示：



6. 如此便完成设计。
7. 画面中已选取 Mars，并且对应的 Timer_1, Timer_2, ..., Speed 亦更新至对应的元件。配方记录中共有 5 笔记录，所以 Count 显示为 5。可以试着选择“配方检视”的其它笔记录，对应的 Name, Timer_1, ... 等信息也会同时更新。
8. 亦可试着操作：

- 新增：

在 Command 中输入 1，将目前输入的数据加至配方数据库成为一笔新的记录。

- 更新：

在 Command 中输入 2，将“数值输入”及“字符输入”的数据更新至配方数据库。

- 删除：

在 Command 中输入 3 即可删除目前选择的记录。

- 排序不同的项目

点选“标题”即可对不同的项目排序。

13.34 流动块

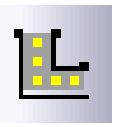
13.34.1 概要

“流动块”元件可表示导管内的滑块移动或运输线的情况。不同于以往使用“移动图形”元件绘制流动图形时，需自行丈量及确认两点之间的位置是否齐整，流动块的每一段区块必为精准的水平或垂直线段且流动间隔固定。

以下列举“流动块”元件的特点：

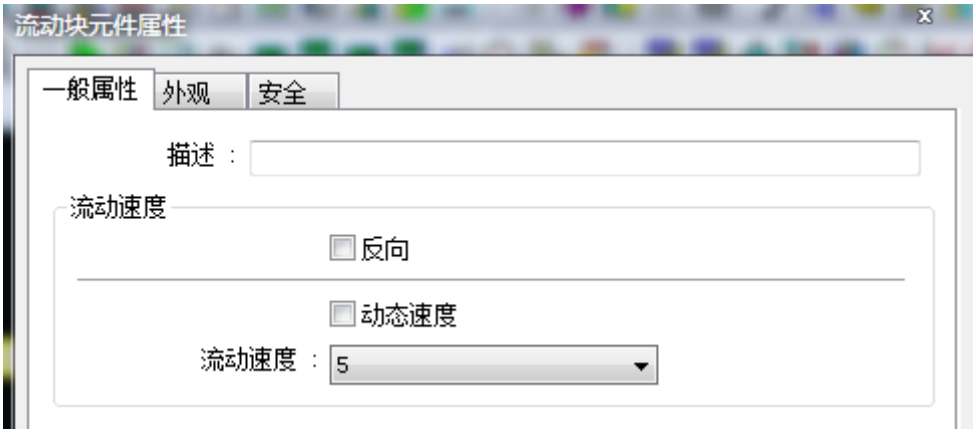
- 每个线段必为垂直或水平的直线，且流动间隔固定。
- 支持动态调整流速及方向 (流速及方向可用指定寄存器调整)。
- 可使用安全机制。利用指定位的状态作为流动块显示与否的依据。

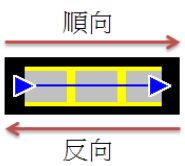
13.34.2 设定



请直接点击“流动块”图标建立此元件，或点选工具列上的“元件”»“流动块”新增此物件。

一般属性设定

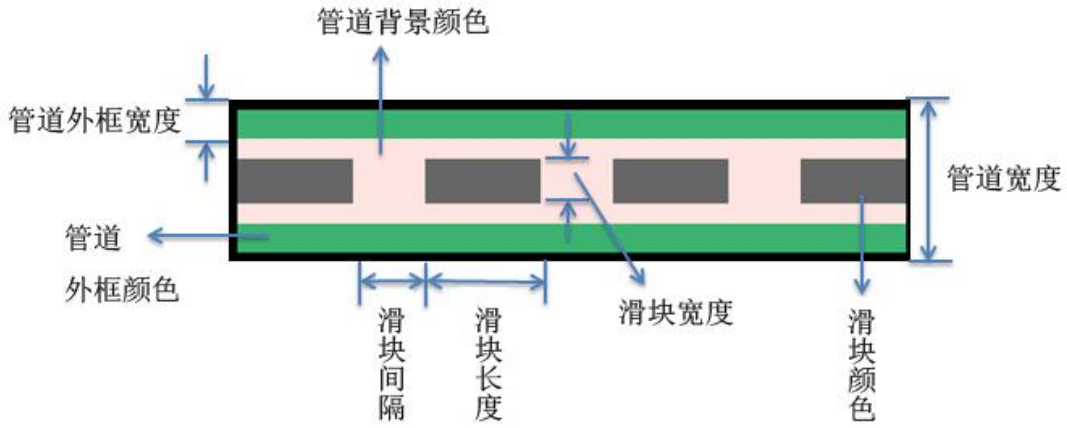


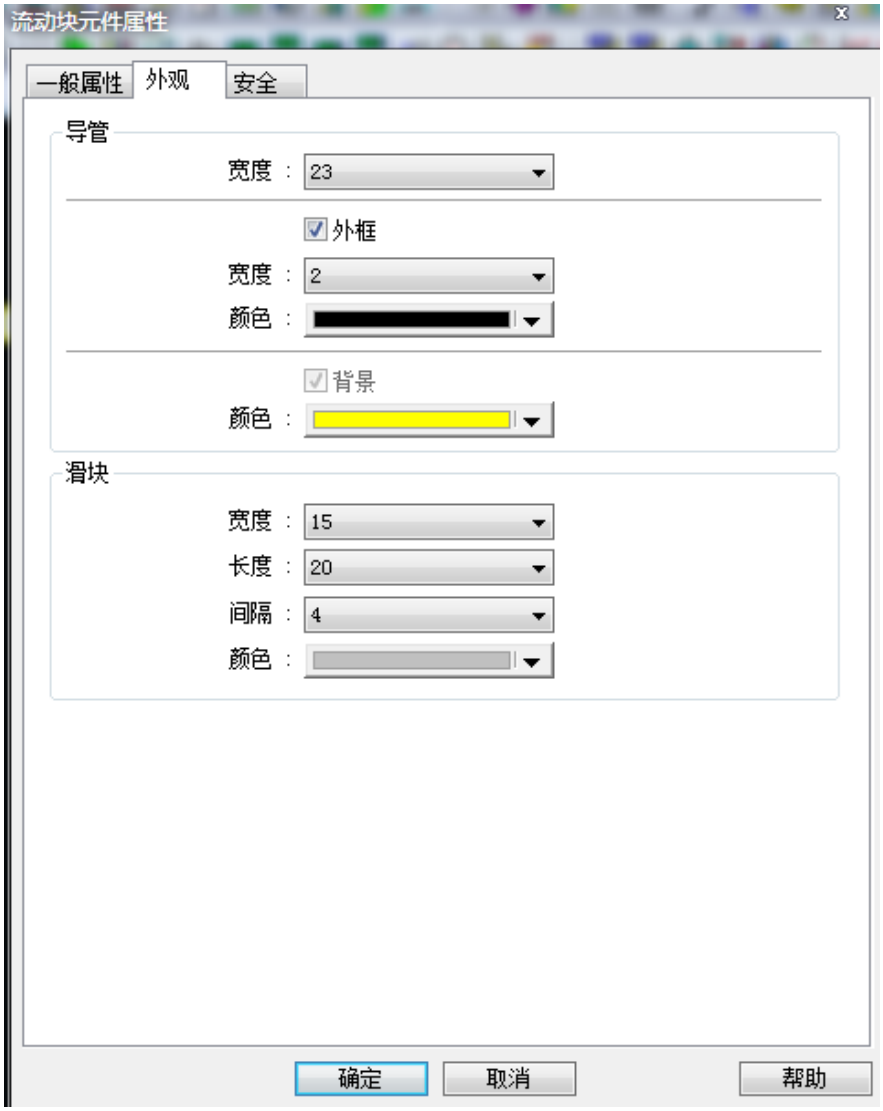
设定	描述
反向	流动块的流动方向会依照建立时的顺序方向移动 (蓝色箭头方向)。勾选反向后，流动的方向将以反向移动。 

动态速度	**读取地址** 流动块的速度及方向可用指定寄存器调整。使用范围为 -25 ~ 25。当输入负值时代表流动反向。 **设定** 显示目前选择的地址及使用格式。并可在此进阶使用“系统寄存器”、”索引寄存器”、”地址标签库” 来设定地址。
流动速度	分为 25 个级速，数值越大，流速越快。若未使用“动态速度”时，则流动速度可设定的范围仅为正向 0 ~ 25。

外观设定

设定流动块元件的外观，各部位的名称请参考下图：





设定	描述
导管	流动块外围的属性设定。可设定外围的背景颜色、边界颜色及宽度。当勾选“外框”时，则必定要使用背景。
滑块	流动块内部的属性设定。可设定流动块的长、宽、间隔及颜色。

Note

- 若在“一般属性”设定页中同时勾选”反向”及“动态速度”，则当在动态速度的控制地址中输入负值时，流动块流动方向将为顺向流动。

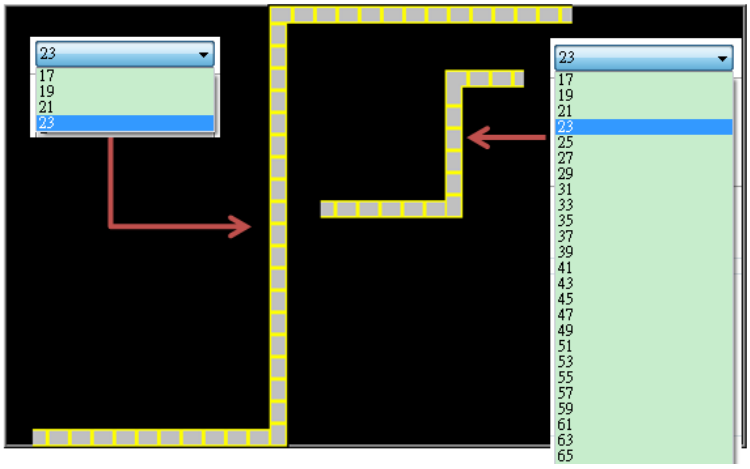
- 绘制流动块元件时，为避免转折处重迭面积过大，在转折处会有最小的规划量。

如图 34.1 的十字下方的图形。图 34.2 的每一线段皆为使用最小规划量绘制出的图形。

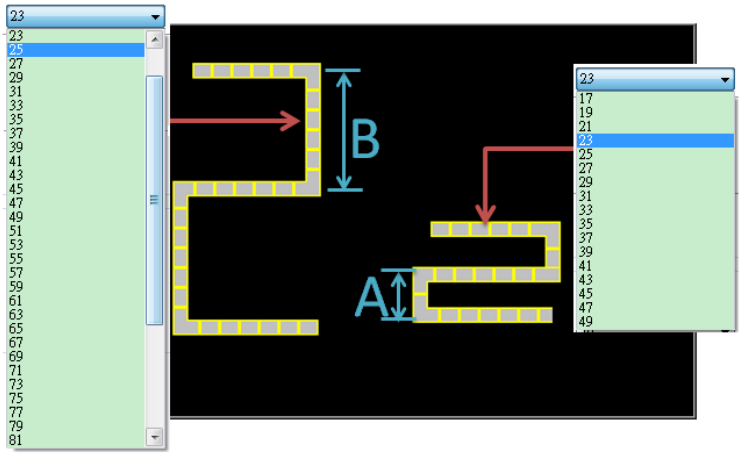
(圖 34.1)　(圖 34.2)

流动块可调整的长宽高参数会根据已绘制的图形间隔尺寸及窗口尺寸改变。

如下图，当流动块整体尺寸越大，则为避免参数调整后会超出窗口尺寸，可设定的长宽参数则越少。反之当整体尺寸越小，可设定的长宽参数则越多。



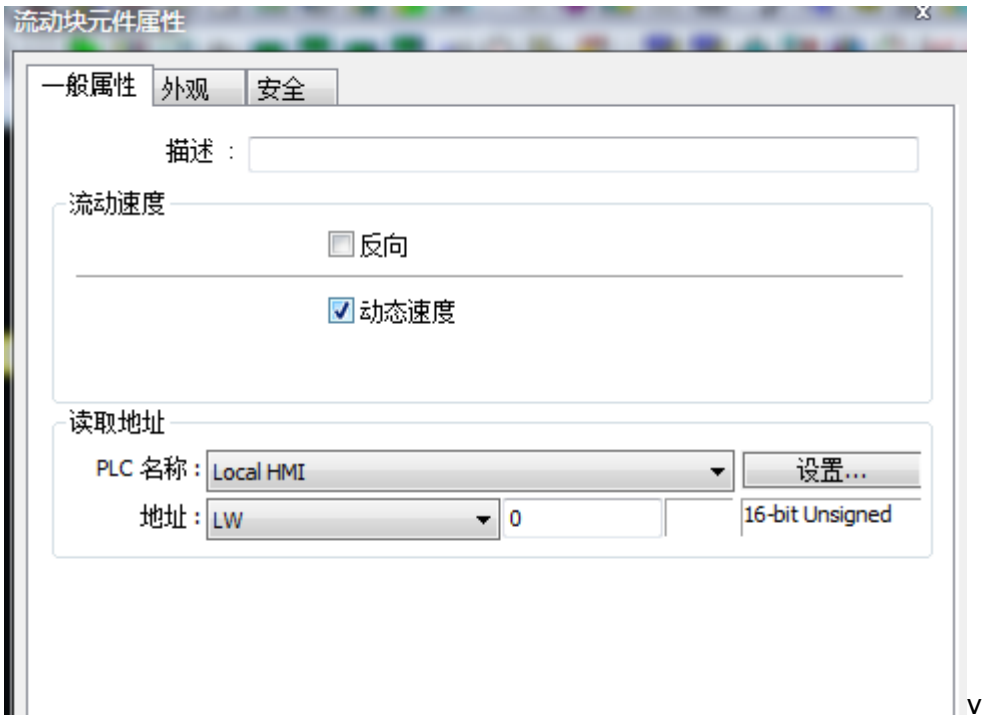
如下图，当流动块转折处的间隔越小，则为了避免参数调整后，会致使流动块上边与下边触及 (如图 34.4 A 区段)，可调整参数会越少；反之当流动块转折处间隔越大 (如图 34.4 B)，可调参数越多。



范例 1

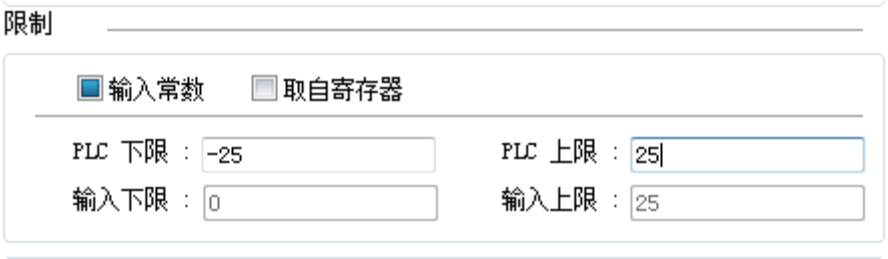
以下范例展示如何藉由“动态速度”调整流动块的速度及方向。

1. 建立一个流动块元件，并使用动态速度，地址设为 LW-0，格式为 16-bit Signed。



v

2. 建立一个数值输入元件，地址设为 LW-0。上限为 25，下限为 -25，格式为 16-bit Signed。



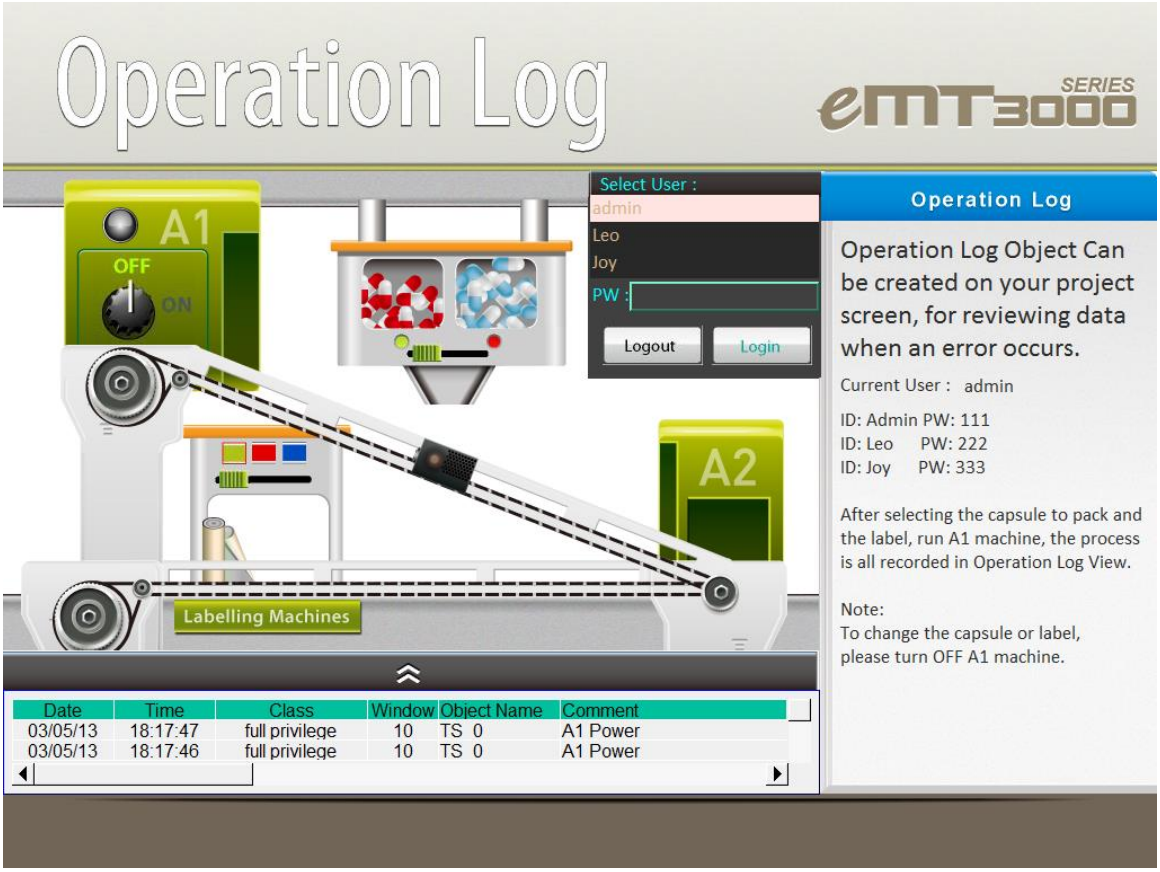
3. 执行模拟或下载至触摸屏验证操作。当在 LW-0 输入正值时，流动方向为顺向，且数值越大，流动速度越快。当输入负值时，流动为反向，且数值越小，流动速度越快。若输入值为 0，则停止流动。

13.35 操作记录

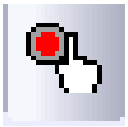
13.35.1 操作记录设定

概要

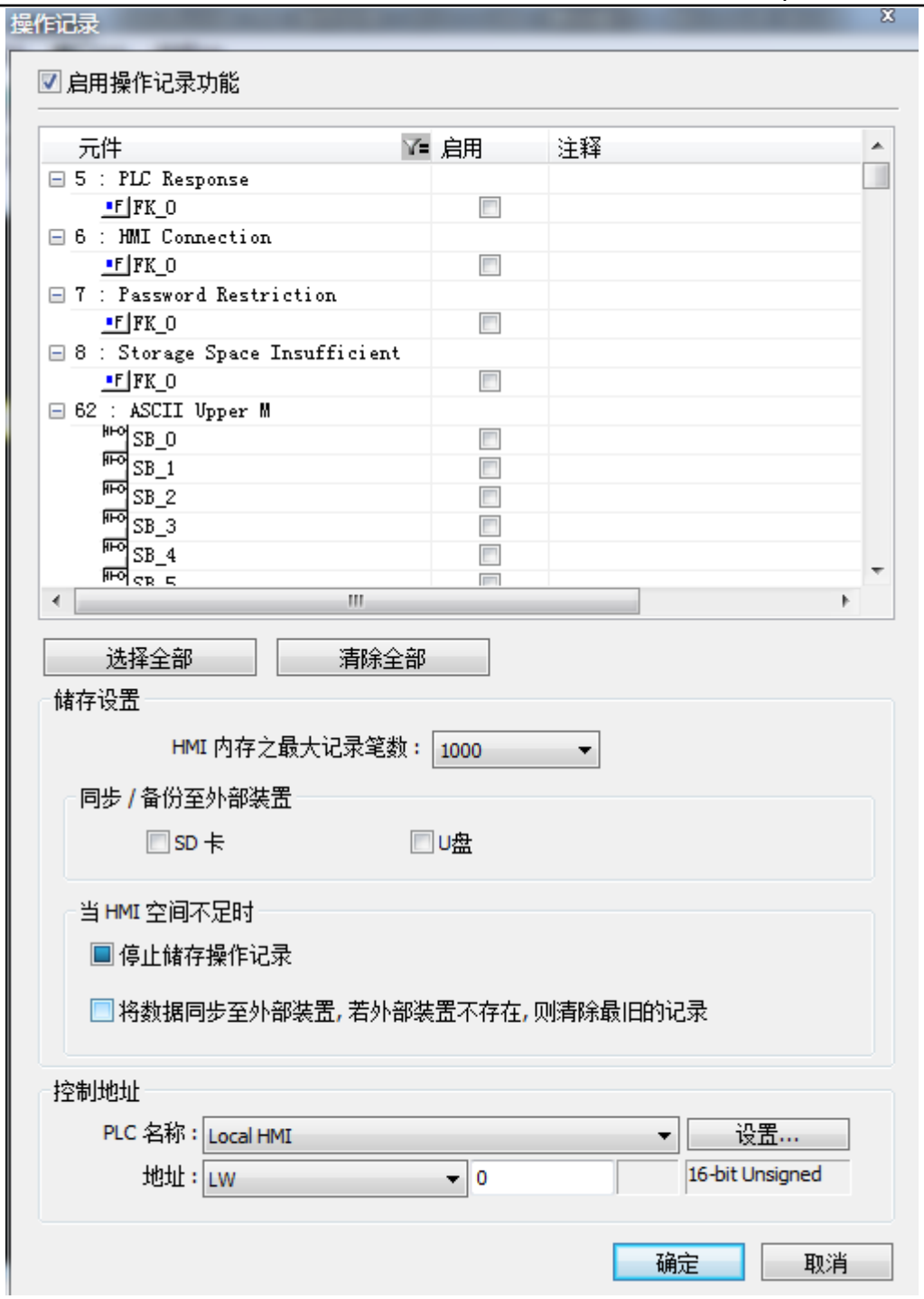
通过操作记录的功能，能够实时将用户所操作的步骤显示出来，当出现异常状况时，可用来分析操作记录，从备份出来的报表可以得知操作过程是否符合作业流程，进一步能针对有问题的部份进行检讨及修正。



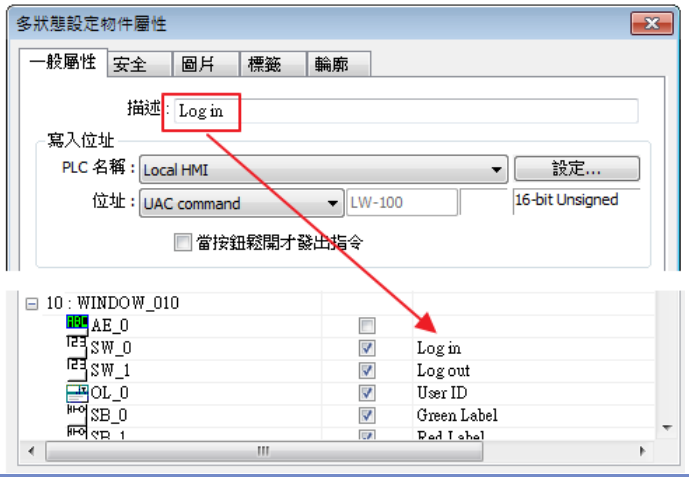
设定



设定需要被记录的写入功能元件。点选“元件”，指向“操作记录”，点选“操作记录设置”并勾选“启用操作记录功能”。



设定	描述
物件	启用操作记录功能后，系统会依照窗口的编号自动列出所有具写入功能的元件。 点选 图标会列出所有具写入功能的元件，用户可通过筛选过滤掉不需记录的元件，仅显示需记录的元件于元件列表。
启用	选择元件是否要被记录。

注释	元件的描述。如下图所示。 
选择全部	选择记录所有元件。 若有使用“筛选”功能，则“选择全部”只会全选显示于列表上的元件。
清除全部	清除所有已勾选的元件。 若有使用“筛选”功能，则“清除全部”只会取消显示于列表上的元件。
储存设定	设定操作记录的数据储存方式。 **HMI内存之最大记录笔数** 设定HMI内存可储存的最大数据笔数。 **同步 / 备份至外部装置** 选择将数据备份至SD卡或U盘。 **当HMI空间不足时** 当触摸屏内的储存空间不足时，选择“停止储存操作记录”以保留较早的操作记录数据或选择“将数据同步至外部装置”。若选择储存至外部的装置，当装置不存在时，则触摸屏会自动清除HMI内存内最旧的记录。
控制地址	输入特定的数值可对选取的操作记录下执行命令并回传命令执行结果。 假设控制地址为LW-n (n为任意地址)，则显示执行结果的地址为LW-n+1。 控制地址 (LW-n)： (1)：清除所有记录 (2)：复制数据到U盘 (3)：复制数据到SD卡 (4)：复制数据到U盘并清除触摸屏内的操作记录 (5)：复制数据到SD卡并清除触摸屏内的操作记录 命令执行结果 (LW-n+1)： (0)：处理中 (1)：执行成功 (2)：装置不存在 (3)：操作记录不存在

(4)：无法判读的错误

Note

- 操作记录仅记录可被手动触发的元件，被动的元件不会被记录，如：定时式数据传输元件。
- 若使用离线 / 在线模拟时，操作记录会被存放于 EasyBuilder 安装文件夹下的 HMI_memory\operationlog\operationlog.db
- 若使用位设定元件触发宏时，则操作记录会记录两笔数据，位的触发及宏的触发。

13.35.2 操作记录检视

概要

“操作记录检视”元件可用来检视操作记录。

设定

使用此元件前，需先至“操作记录设置”设定须检视的元件。点选“元件”»“操作记录”»“操作记录检视”。

一般属性设定


新增 操作记录检视 元件
一般属性
标题
图片
描述 :
标题
透明
颜色 :
外观
透明
外框 :
背景 :
网格
显示
选择控制
颜色 :
字体
颜色 :
名称 :
Arial
尺寸 :
12
确定
取消
帮助


[标题]
[外观]
外框及背景
[选择控制]
[网格]
Date
Time
Class
Window
03/04/13 15:15:07 full privilege 10
03/04/13 15:15:07 full privilege 10
03/04/13 15:15:06 full privilege 10
03/04/13 15:15:00 full privilege 10
03/04/13 15:14:55 full privilege 10
03/04/13 15:14:55 full privilege 10
02/26/13 18:50:21 full privilege 10
02/26/13 18:50:18 full privilege 10
02/26/13 18:50:15 full privilege 10
02/26/13 18:50:09 full privilege 10
02/26/13 18:50:07 full privilege 10

设定	描述
标题	设定标题字段的颜色。 **透明** 若勾选透明，则标题字段就不会有颜色，也没有“颜色”的选项。
外观	设定元件的外框线颜色及背景颜色。 **透明** 若勾选透明，则元件就不会有外观颜色，也没有“颜色”的选项。
网格	设定区分每个行与列的间隔线颜色。 **显示** 若勾选，则显示网格，若不勾选，则不显示网格。
选择控制	当点选到特定的一列资料时，该字段所显示的颜色。
字型	“操作记录检视”元件上文字的字型、尺寸、颜色。

标签设定

设定	描述
标题	显示于“操作记录检视”元件上的标题。
排序	设定操作记录排序方式。
显示顺序	设定当操作记录产生时，项目信息的显示顺序。若“显示字符”设定为 0，系统将显示所有字符。
日期/时间	设定“操作记录检视”元件上日期及时间的格式。

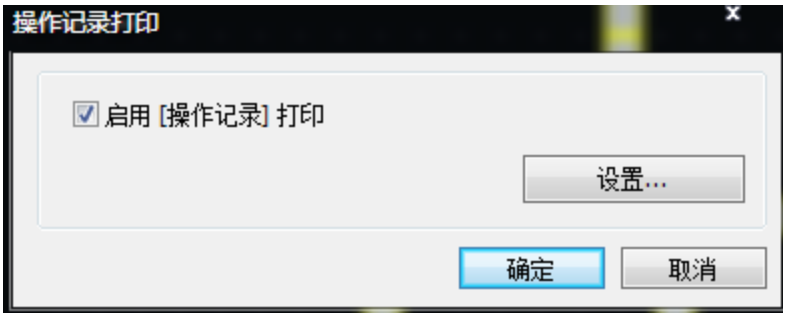
13.35.3 操作记录打印

概要

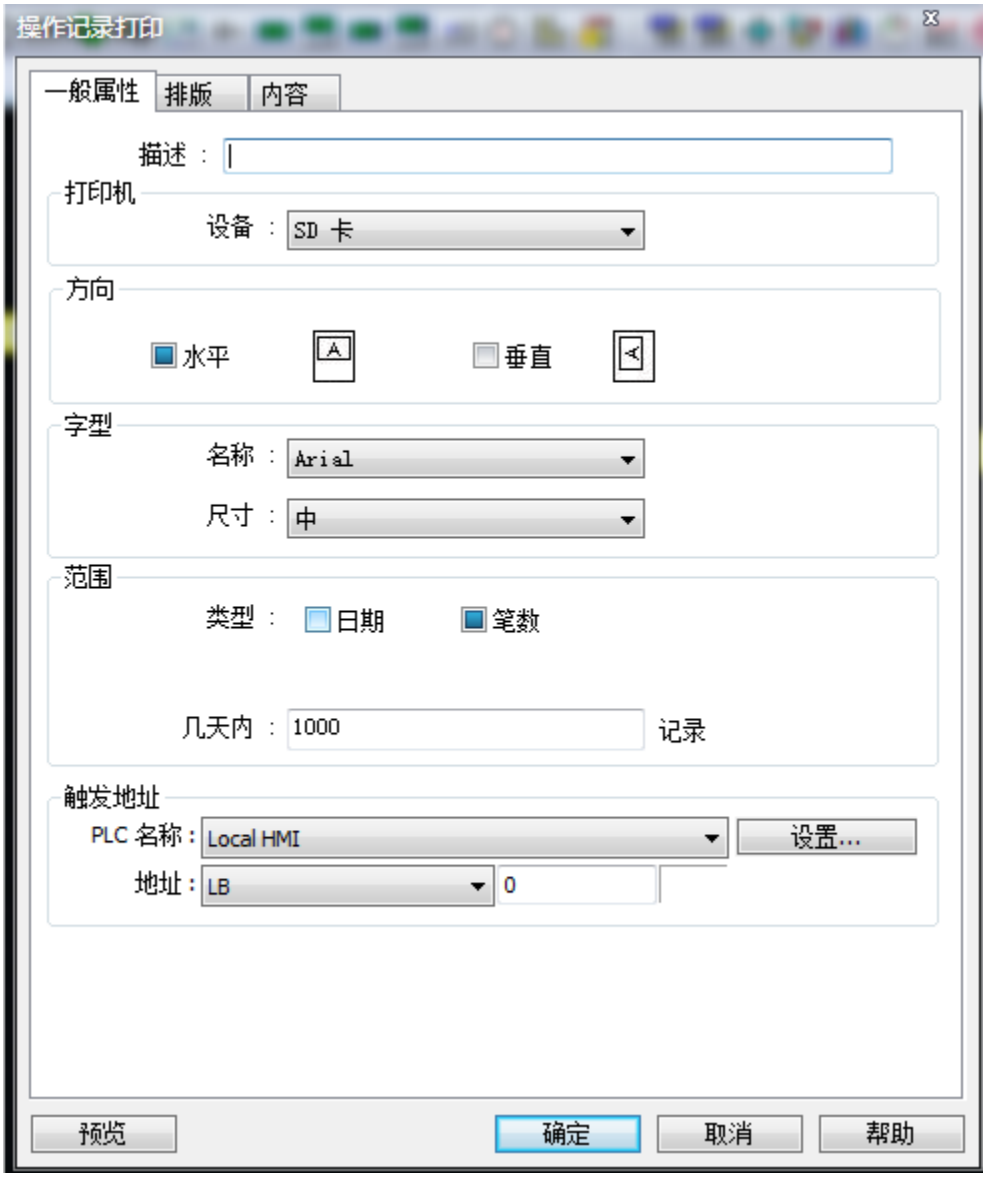
“操作记录打印”可将操作记录的内容转换成报表的形式输出至外接储存装置或打印机。当使用外接储存装置时，报表会输出成 JPEG 格式。使用此功能前，需先启用“操作记录设置”。

操作

勾选“启用“操作记录”打印”后，点选“设置”进入打印设定。



一般属性设定



设定	描述
打印机	选择输出“操作记录”文件时的外部储存装置。当选择输出于打印机时，打印机必须为 A4 格式。若选择储存于外部储存装置时，系统会产生一个 **operationlogsheet** 文件夹， 将操作记录的报表转成 JPEG 图文件，以 "打印的日期_打印序号" 的方式命名并储存于此文件夹。 例如：2013/05/08 打印的第一张图档，则 JPEG 档名为 130508_0000，依此类推。
方向	选择“操作记录”文件输出时的方向。
字型	选择“操作记录”文件输出时的字型及尺寸。当字型设定为大中小时的对应尺寸如下表：

尺寸	内文	标题
大	16点	20点
中	12点	16点
小	8点	12点

范围	选择“操作记录”文件输出时的数据范围。 **日期** 使用日期决定输出的数据范围时，会从“起始时间”往前推算“几天内”做输出报表。最大为 30 天。 **笔数** 使用笔数决定输出的总数时，最大为 10000 笔记录。
触发地址	设定控制“操作记录打印”的寄存器地址来源。当寄存器被设定成 ON 时会触发输出，在输出工作完成后，寄存器会自动重置成 OFF 状态。
预览	检视欲输出的画面。

排版设定

排版设定页各部分的位置分配如右图。

设定	描述
标题	设定欲显示标题的内容，标题只可为一行。 **显示于所有页面** 勾选后，则标题内容将显示于每一页，反之，只显示于第一页。
页首	设定欲显示页首的内容，页首内容最多支持打印五行文字。 **显示于所有页面** 勾选后，则页首内容将显示于每一页，反之，只显示于第一页。
页尾	设定欲显示页尾的内容，页尾内容最多支持打印五行文字。 **显示于所有页面** 勾选后，则页尾内容将显示于每一页，反之，只显示于最后一页。
日期/时间	勾选后，打印时的日期/时间会显示于每一页右下角，反之，则不显示。
页码	每一页都会显示。

内容设定

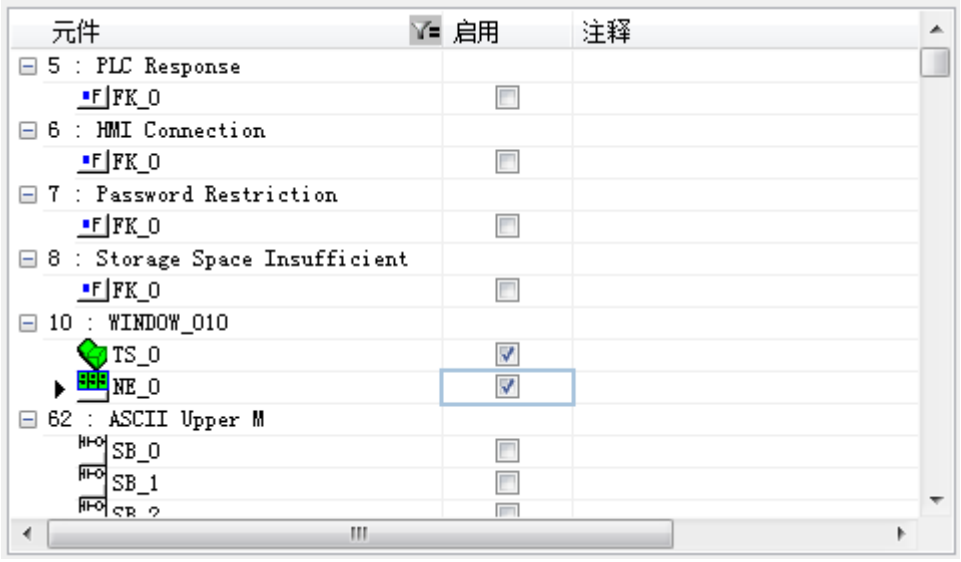
设定	描述
标题列表	设定标题名称的显示文字。
排序	**按时间顺序** 操作记录将由旧到新依序显示，最近的操作记录显示于底部。 **按时间逆序** 操作记录将由新到旧依序显示，最近的操作记录显示于顶部。
日期/时间	设定时间信息的显示格式。

范例讲解

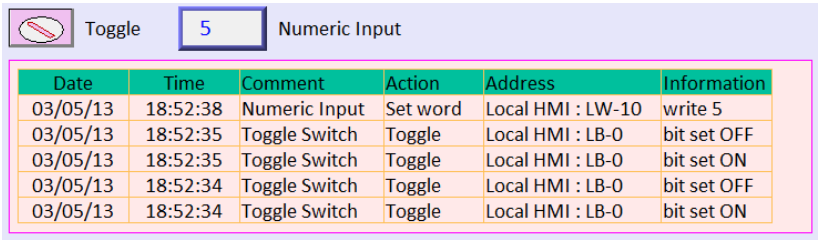
范例 1

本范例讲解如何建立一个操作记录。

1. 建立一个“位状态切换开关”元件和“数值输入”元件于窗口 10。
2. 开启“操作记录设置”元件，启用窗口 10 下的“位状态切换开关”元件和“数值输入”元件。



3. 建立一个“操作记录检视”元件。设定好各项属性后关闭。
4. 执行离线模拟，触发“位状态切换开关”元件和“数值输入”元件，操作记录会显示于“操作记录检视”元件上。

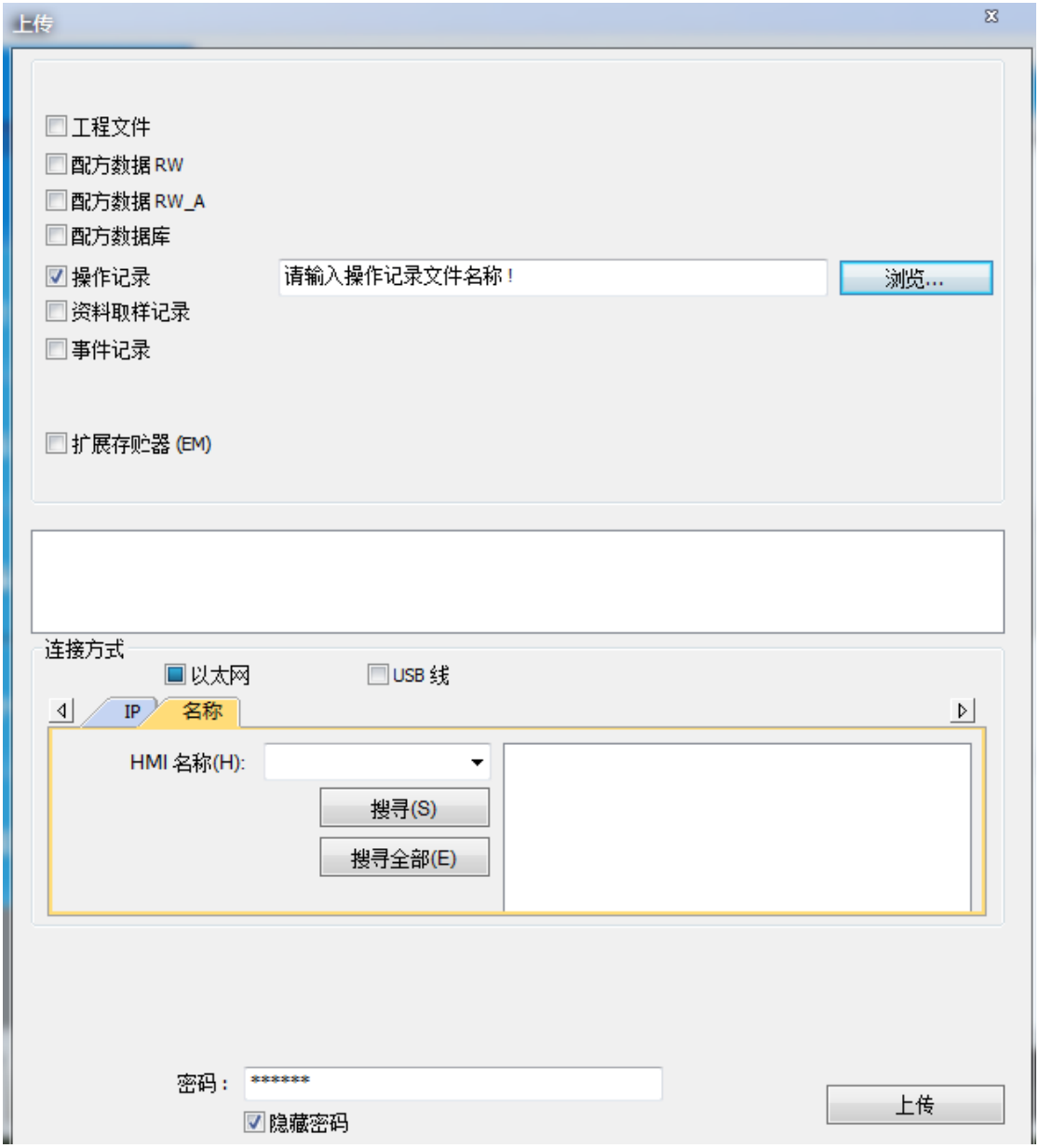


范例 2

操作记录文件可以通过 Utility Manager 上传到计算机或是利用“备份”元件将文件通过邮件传送至信箱。

- 使用 Utility Manager 上传

1. 开启 Utility Manager，点选“上传”。
2. 勾选“操作记录”，输入文件名及触摸屏的 IP，点选“上传”。



- 使用邮件传送到信箱

1. 点选“系统参数设置”»“邮件”设定邮件的服务器及收/寄件者邮件地址。
2. 建立“备份”元件，来源设定为“操作记录”，备份位置设定为“邮件”。

邮件服务器设定方法请参考《5 系统参数设定》。

13.36 复合式多功能按钮

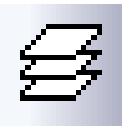
13.36.1 概要

复合式多功能按钮元件可以执行多重指令。以往同一区块若要执行多个指令时，必须将元件迭加在同一位置，且执行时会根据迭加的顺序执行指令。用户必须根据指令执行的顺序迭加元件，规划程序时需花费较多时间测试执行的顺序。复合式多功能按钮可以直接让用户设定多重指令的执行，并调整其顺序。

以下列举复合式多功能按钮元件的特点：

- 可执行多重指令。
- 自行调整多重指令的执行顺序。
- 可使用位或字符作其元件显示的状态。

13.36.2 设定



按下工作列上的“复合式多功能按钮”按钮后即会开启“复合式多功能按钮”元件属性对话框，正确设定各项属性后按下确认键，即可新增一个“复合式多功能按钮”元件。

一般属性设定

设定	描述
指示灯	元件显示的状态模式，可选择是否使用多状态显示。 **无** 不使用多种状态。 **位状态指示灯** 读取位寄存器的数据来显示状态。”输出反向”可以将读取的状态作反向显示，例如位的状态实际上为 OFF，但勾选了“输出反向”后会显示为 ON。 **多状态指示灯** 读取字符寄存器的数据来显示状态。 “状态数”为元件显示的状态数目。状态从 0 开始编号，能显示的最大状态编号为设定的“状态数”– 1，当要求显示超过设定的状态数时，系统会显示最后一个状态。例如设定 “状态数” 为 8，则显示的状态依序为 0，1，2，…，7，若要求寄存器显示状态 8 以上的状态时，显示的图片仅显示状态 7。
动作	设定执行的动作，可选择使用“延迟”、”位状态设置”、”多状态设置”、“切换基本窗口”。一个复合式多功能按钮最多可执行 8 个指令。

改变选取的指令的执行顺序。

 复制  贴上  删除

对选取的指令进行复制、贴上、或删除的动作。

第十四章 向量图库与图片库的建立

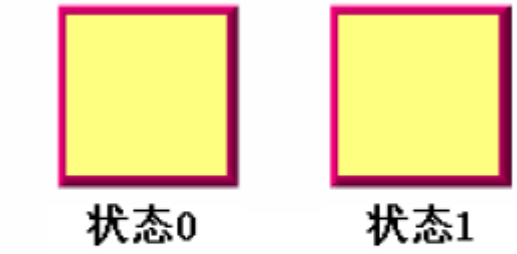
14.1 概要

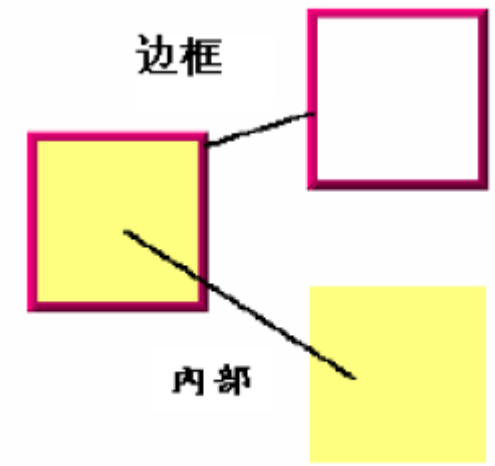
EasyBuilder Pro 提供向量图库与图片库的使用。在图库管理上提供“工程文件”及“图库”二种模式。”工程文件”的图片被储存在 .emtp 的工程文件内，”图库”的图片则会被储存在 EasyBuilder Pro 图库目录或是用户自订的路径下。图片可以增加元件在视觉上的效果，每个向量图或图片最多可包含 256 个状态。下文将说明如何建立向量图库与图片库。

向量图库与图片库的使用请参考《9 元件一般属性》。

14.2 向量图库的建立

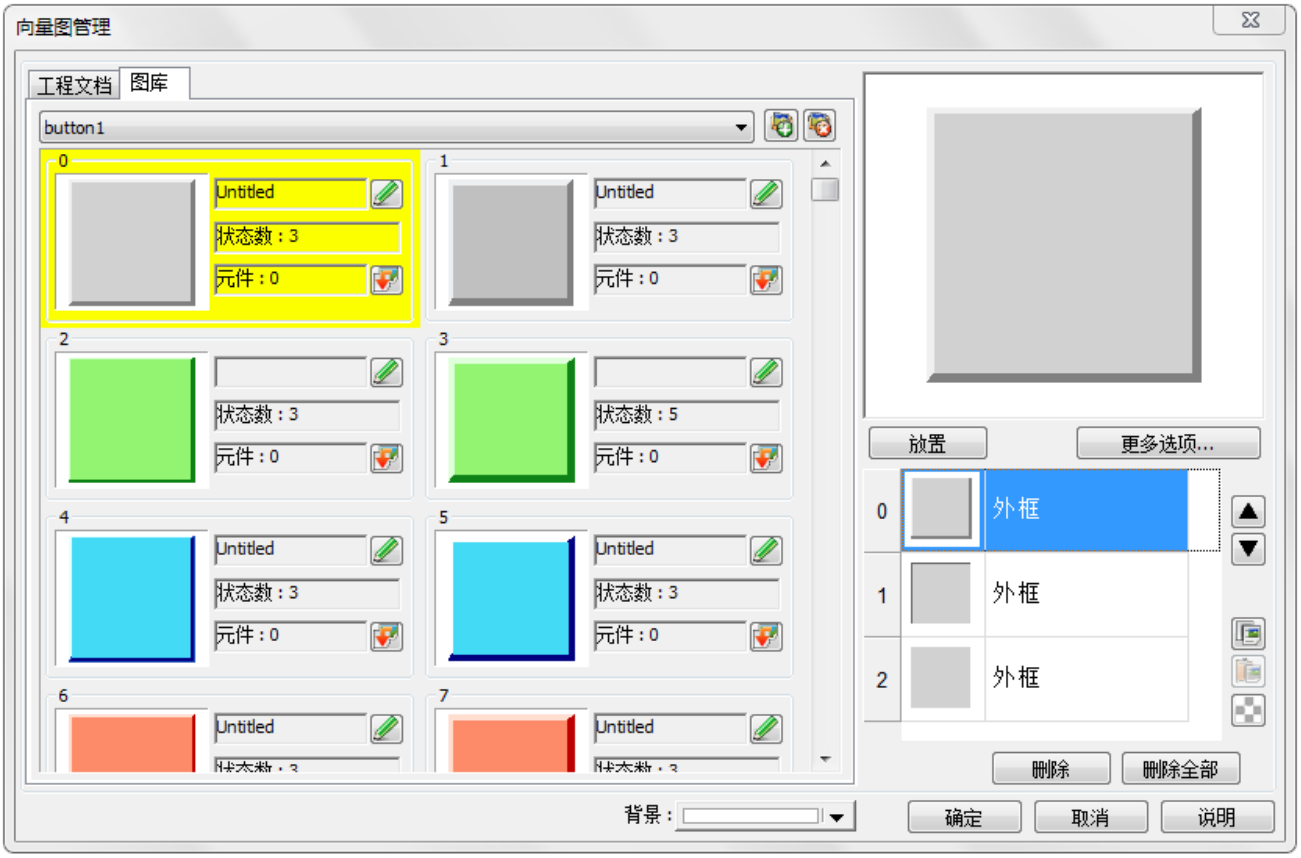
向量图是由直线、曲线、多边形等绘图组件所构成的图形；一个完整的向量图可能具有一个以上的状态，每个状态都可包含两个部分：外框与内部，如下图所示。





14.2.1 向量图管理

元件可以单独选择使用向量图的外框及内部或者同时使用。在按下工具列的“调用向量图库”按钮，即可进入“向量图管理”对话框，如下图所示。

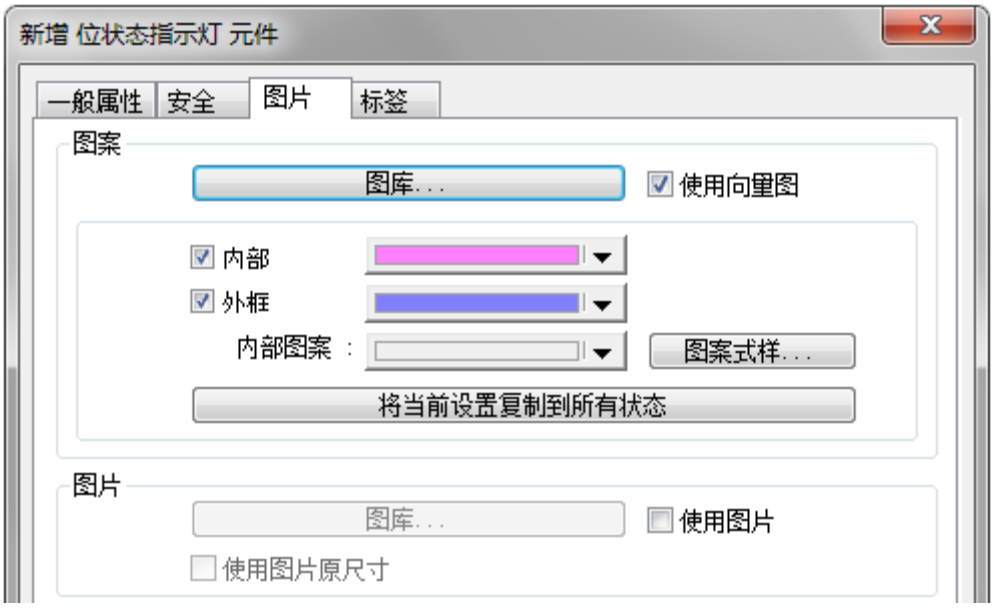


设定	描述
工程文件	在此页面编辑的向量图，将储存于 .emtp 工程文件中，最多可新增 1000 个向量图。
图库	在此页面编辑的向量图，将储存于计算机图库目录下，但并不会储存于 .emtp 工程文件中。
新增图库	新增已存在的 .plb 向量图库文件。若要新增一个全新的图库，则输入不存在的档名后按“开启”，将会建立一个空白的图库文件，最多可新增 40 个图库。
删除图库	删除目前的图库。
复制至工程文件	将此向量图复制到“工程文件”中。仅支持非系统向量图库的图形复制。以下 4 种系统向量图库不支持复制功能 System

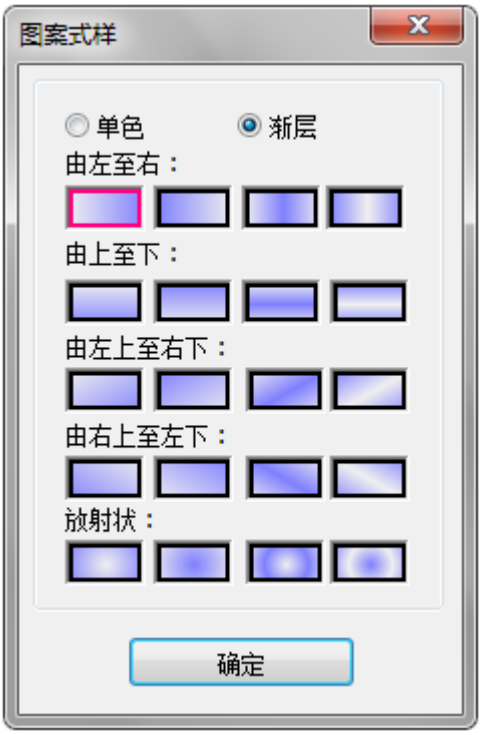
	Frame/System Button / System Lamp / System Pipe。
背景	选择向量图的背景颜色，此颜色只会在“向量图管理”中显示，实际图片并不会有此背景颜色。
放置	将所选择的向量图放置到项目编辑画面中。此功能提供给非系统图库使用。
更多选项	可以设定“内部”、”外框”以及“图案”的颜色及样式。
▲ ▼	将此向量图形往前 / 往后移一个状态。
复制	复制此向量图形。
贴上	贴上已复制的向量图形。
插入透明状态	在此向量图形后插入一个空白状态。
删除	删除此向量图形的某一个状态。
删除全部	删除所选择的向量图形的全部状态。
确定	确定储存此次编辑结果。
取消	取消此次编辑结果。
说明	开启说明文件。

Note

- 向量图库中可选择“内部”，”外框”的色彩，以及“图案式样”的功能只开放给 System Frame/System Button 图库。



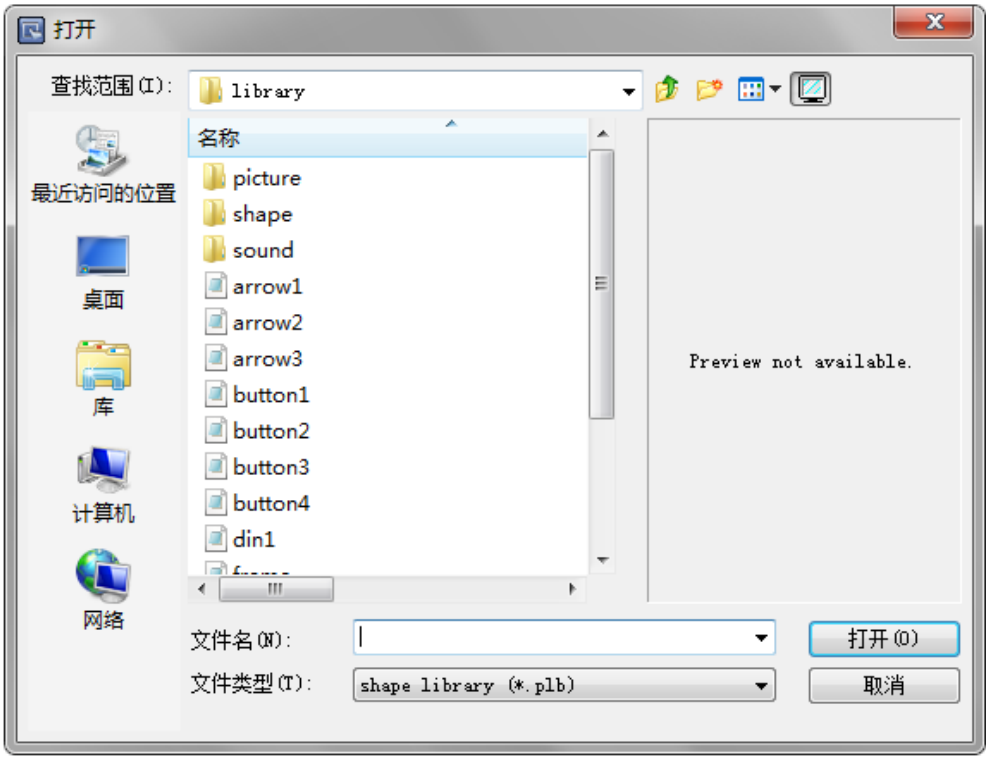
■ cMT-SVR 系列的“图案式样”功能支持渐层样式，如下图所示：



14.2.2 建立向量图库的步骤

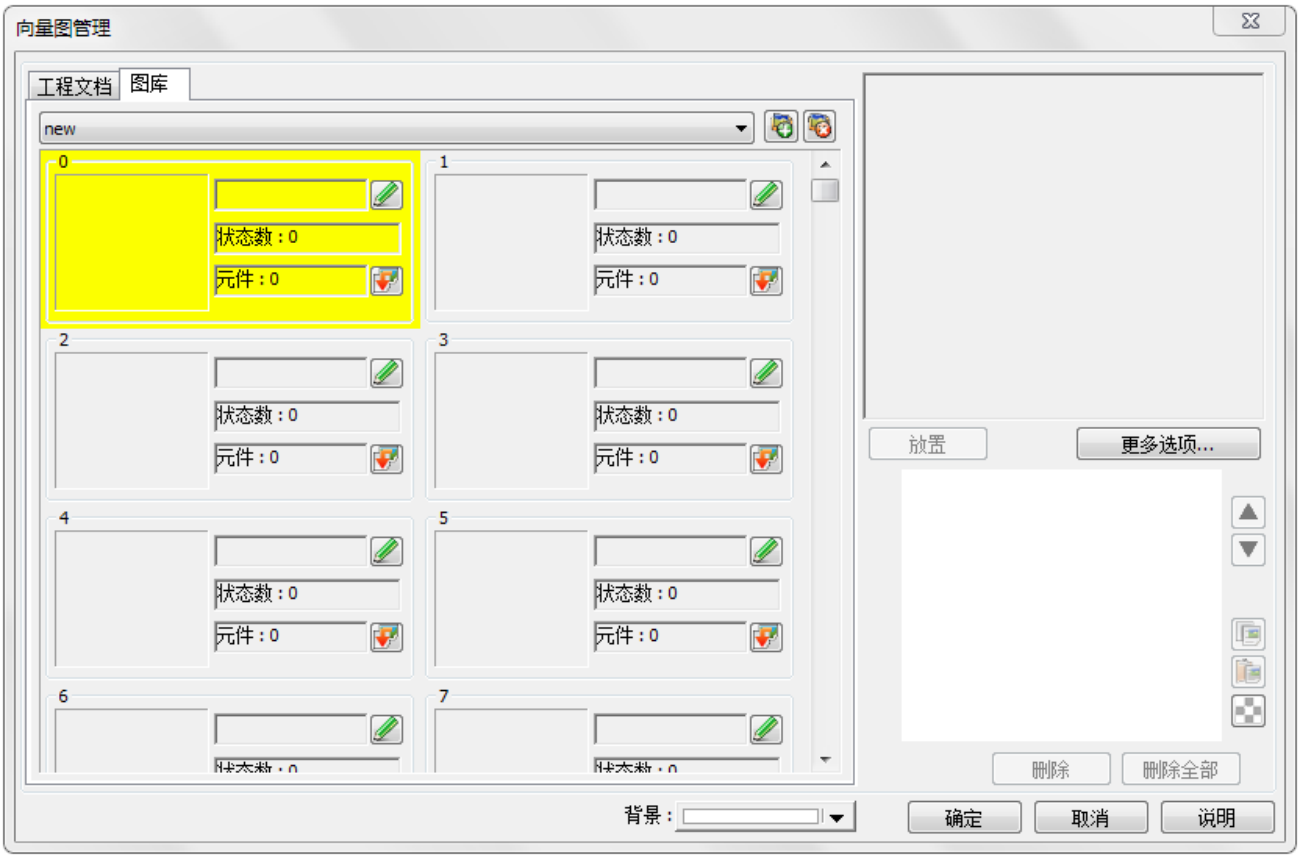
下面说明如何建立一个新的向量图库，并在此图库中加入一个具有两个状态的向量图。

1. 按下“新增图库”后，在对话框中输入新的向量图库名称。

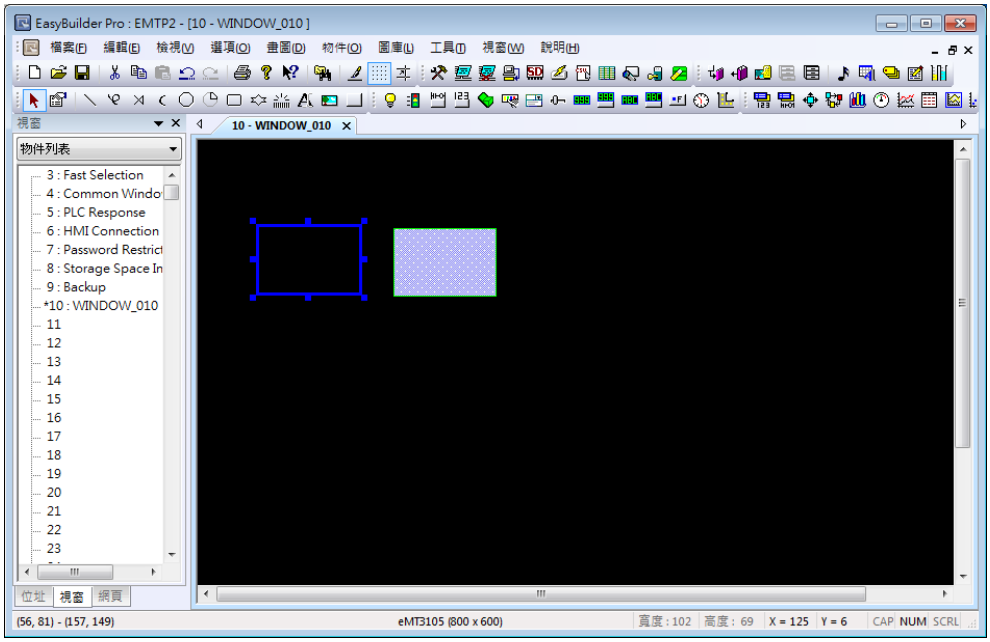


2. 点选“开启”后，会弹出寻问是否要建立文件的对话框，接着按“是”进行下一步。

3. 此时可以发现“向量图管理”中增加一个新的向量图库“new_lib”，且此新的向量图库中并未包含任何向量图，如下图所示。

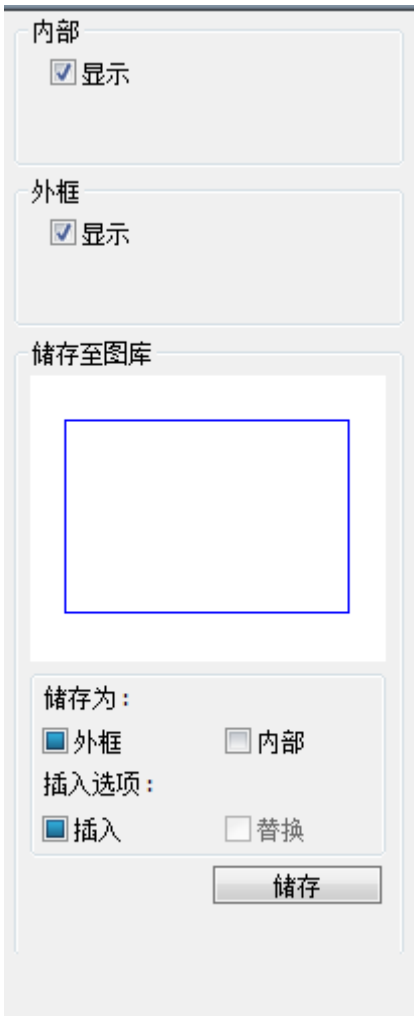


4. 对特定向量图，加入一个状态。首先使用绘图工具在窗口上绘出需要的外框与内部图形，并圈选要加到向量图库的外框图形。



5. 接着在圈选图形组件后，按下工具列上的“储存至向量图库”按钮，选择已建立的“new_lib”图库，接着选择要储存的向量图编号，被选择的向量图会显示黄色的背景。

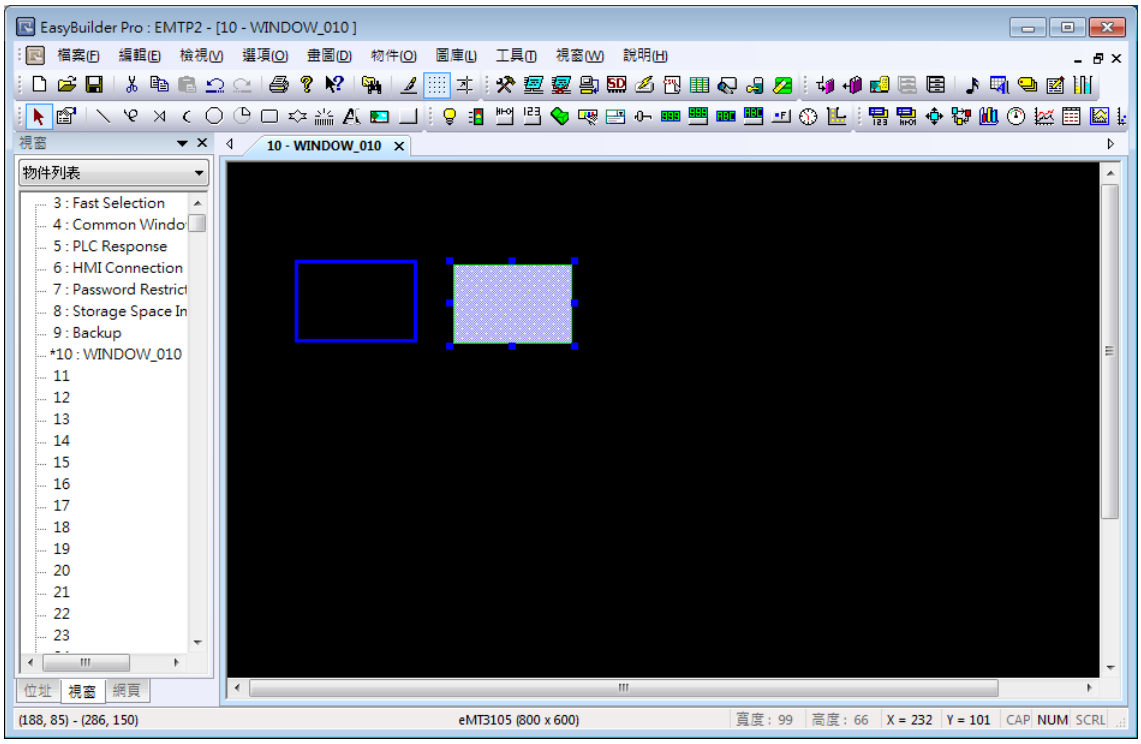
6. 接下来设定将此向量图形储存为此状态的“外框”，插入选项选择“插入”并点选“储存”。



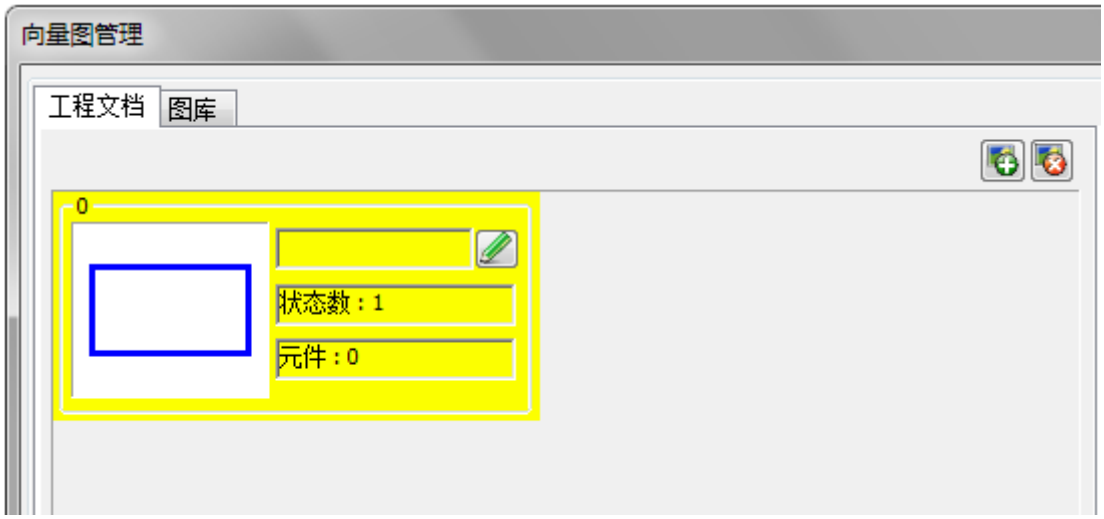
设定	描述
内部	设定是否显示向量图形内部。
外框	设定是否显示向量图形外框。
储存至图库	**储存为外框** 将此图形储存为向量图外框。 **储存为内部** 将此图形储存为向量图内部。 **插入** 选择插入一个新的向量图状态。 **替换** 选择替换此向量图状态。
储存	确定储存以上设定。

7. 由下图可以看到此向量图已新增了一个状态，它的外框已被定义完成。

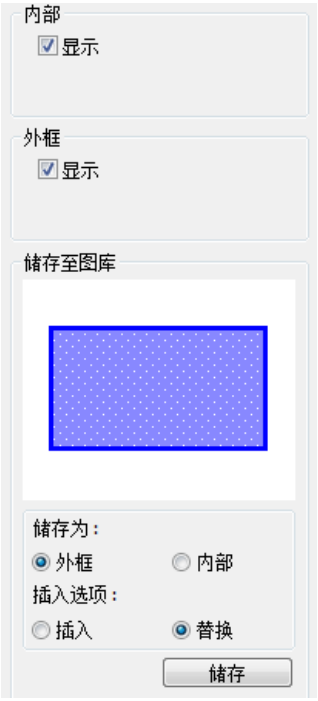
8. 设定内部图形。在窗口上圈选已绘好的内部图形。



9. 按下工具列上的“储存至向量图库”按钮，选择已建立的“new_lib”图库，接着选择储存与外框相同的向量图编号。

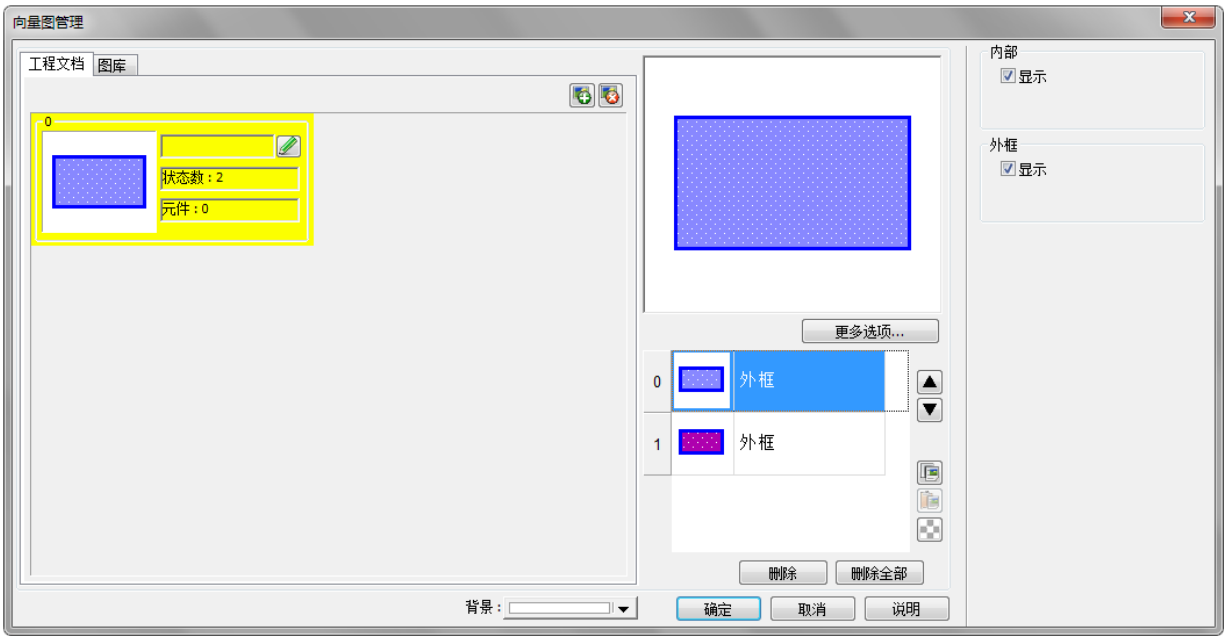


10. 设定将此向量图形储存为此状态的“内部”，插入选项选择“替换”，点选“储存”。



11. 一个状态的向量图形可以只有“内部”或是“外框”，也可以同时存在。由下图可看到状态 0 的向量图形已具备“内部”与“外框”的设定。按下“确定”后，状态 0 的图形即建立完成。

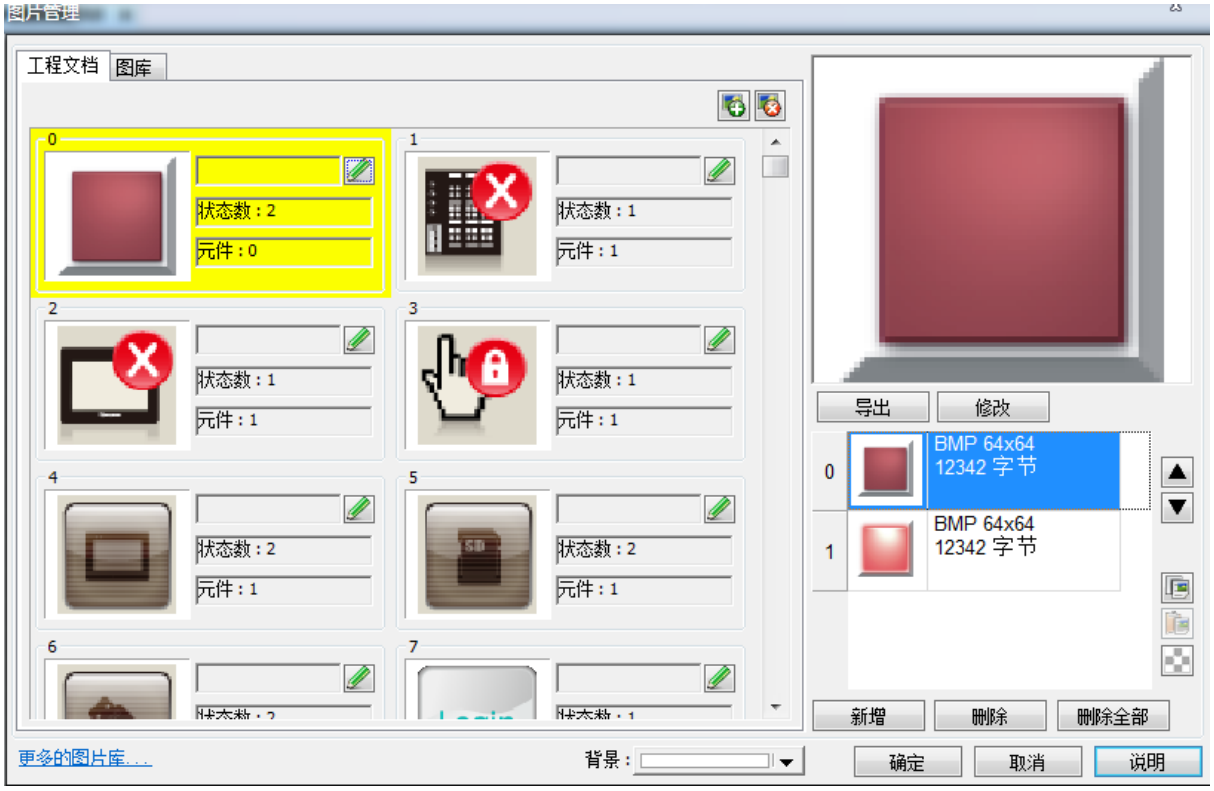
12. 使用与状态 0 相同的方式，插入一个向量图形做为状态 1，如下图所示，此向量图编号已建立包含两个状态的向量图形，最后按下“确定”即完成设定。



14.3 图片库的建立

14.3.1 图片管理

按下工具列上的“调用图片库”工作按钮后将出现“图片管理”对话框，如下图所示。



设定	描述
工程文件	在此页面编辑的图片，将储存于 .emtp 工程文件中。最多可新增 1000 个图片。
图库	在此页面编辑的图片，将储存于计算机图库目录下，不会储存于 .emtp 工程文件中。
新增图库	新增已存在的 .flb 图库文件，若要新增一个全新的图库，则输入不存在的档名后按“开启”，将会建立一个空白的图库文件，最多可新增 40 个图库。

删除图库	删除目前的图库。
复制至工程文件	将此图片复制到“工程文件”中。
背景	选择图片的背景颜色，此颜色只会在“图片管理”中显示，实际图片并不会有此背景颜色。
更多的图片库	连结至 Weintek 官方网站的图片库下载区，登入账号后即可下载更多的图片库。
导出	将此图片导出。
修改	修改此图片设定。
	将此图片往前 / 往后移一个状态。
复制	复制此图片。
贴上	贴上已复制的图片。
插入透明状态	在此图片后插入一个空白状态。
新增	新增一个图片。
删除	删除此图片。
删除全部	删除全部的图片。
确定	确定储存此次编辑结果。
取消	取消此次编辑结果。
说明	开启说明文件。

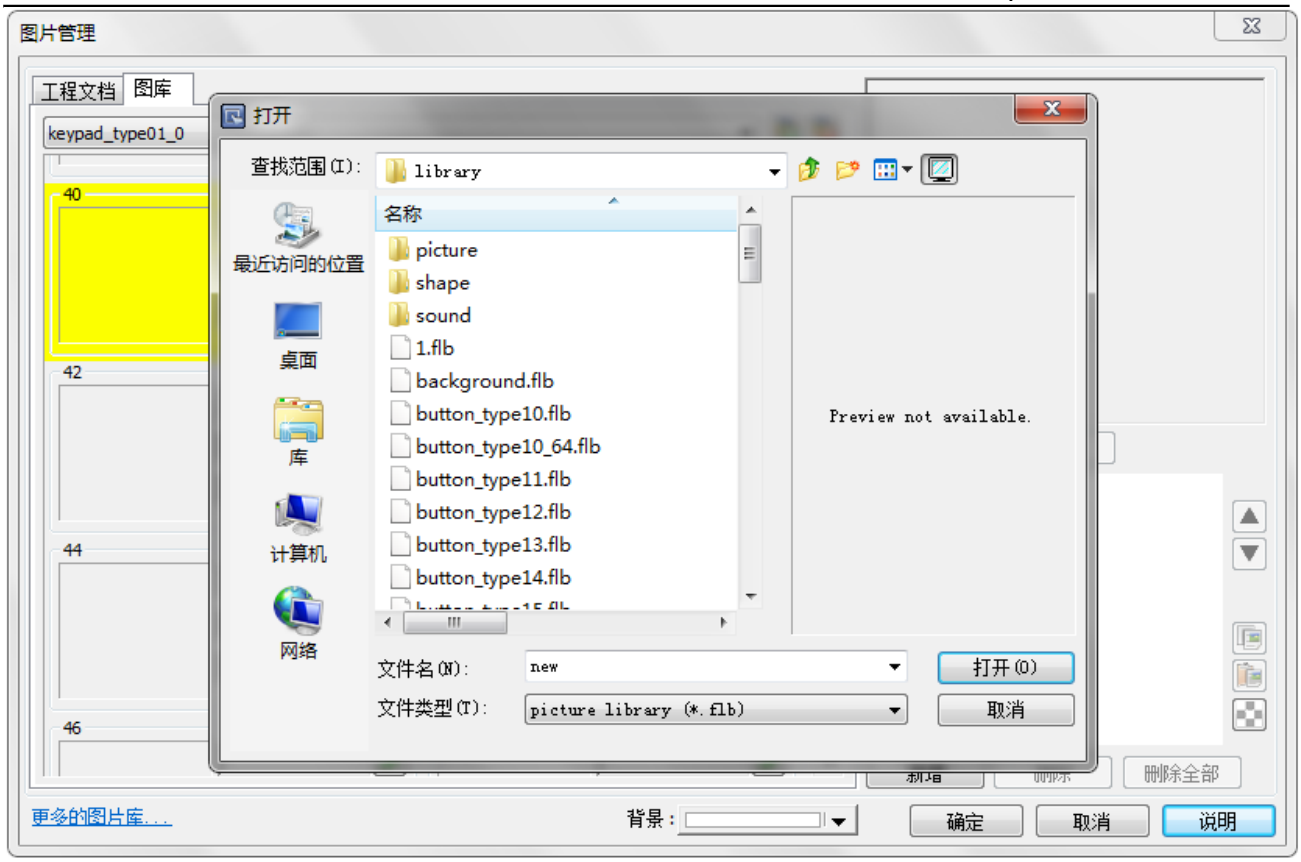
Note

- 图片库支持的图片格式为 .bmp、.jpg、.gif、.dpd、.svg 和 .png。在图片库新增 .gif 格式的图档时，如果图档属于动画类型的文件，用户可以设定动画的播放次数。如下图所示

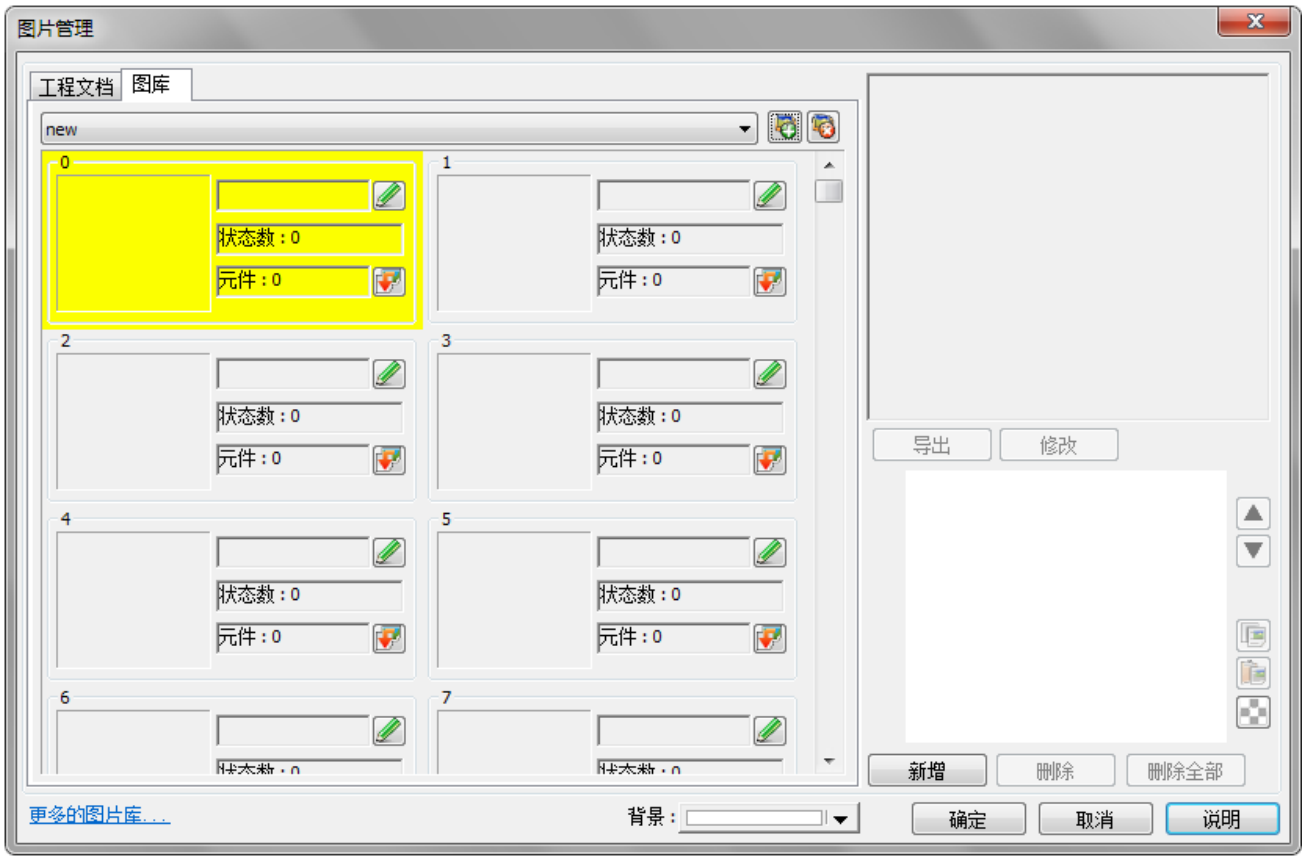
14.3.2 建立图片库的步骤

下面说明如何新建立一个新的图片库，并在此图库中加入一个具有两个状态的图片。

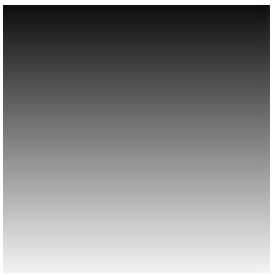
1. 按下“新增图库”后，在对话框中输入新的图片库名称。



2. 点选“打开”后，会弹出寻问是否要建立文件的对话框，接着按“是”进行下一步。
3. 此时可以发现“图片管理”对话框中增加一个新的图库“new_lib”，且此新的图库中并未包含任何图片，如下图所示。



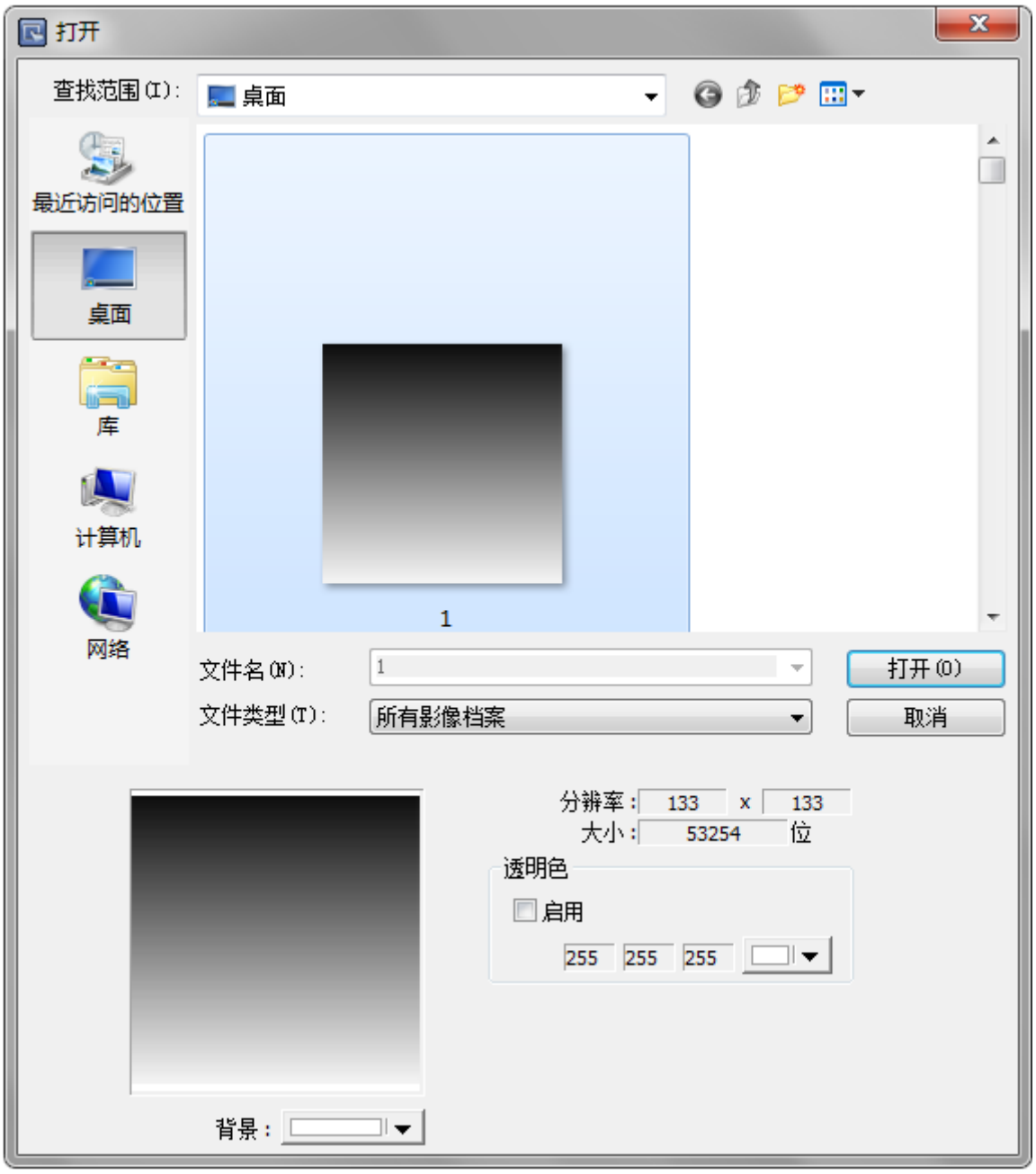
4. 假设要将下面两张图片分别用来表示状态 0 与状态 1。



5. 首先选择已建立的“new_lib”图库，接着选择图片接下来要储存的图片编号，被选择的图片编号会显示黄色的背景。

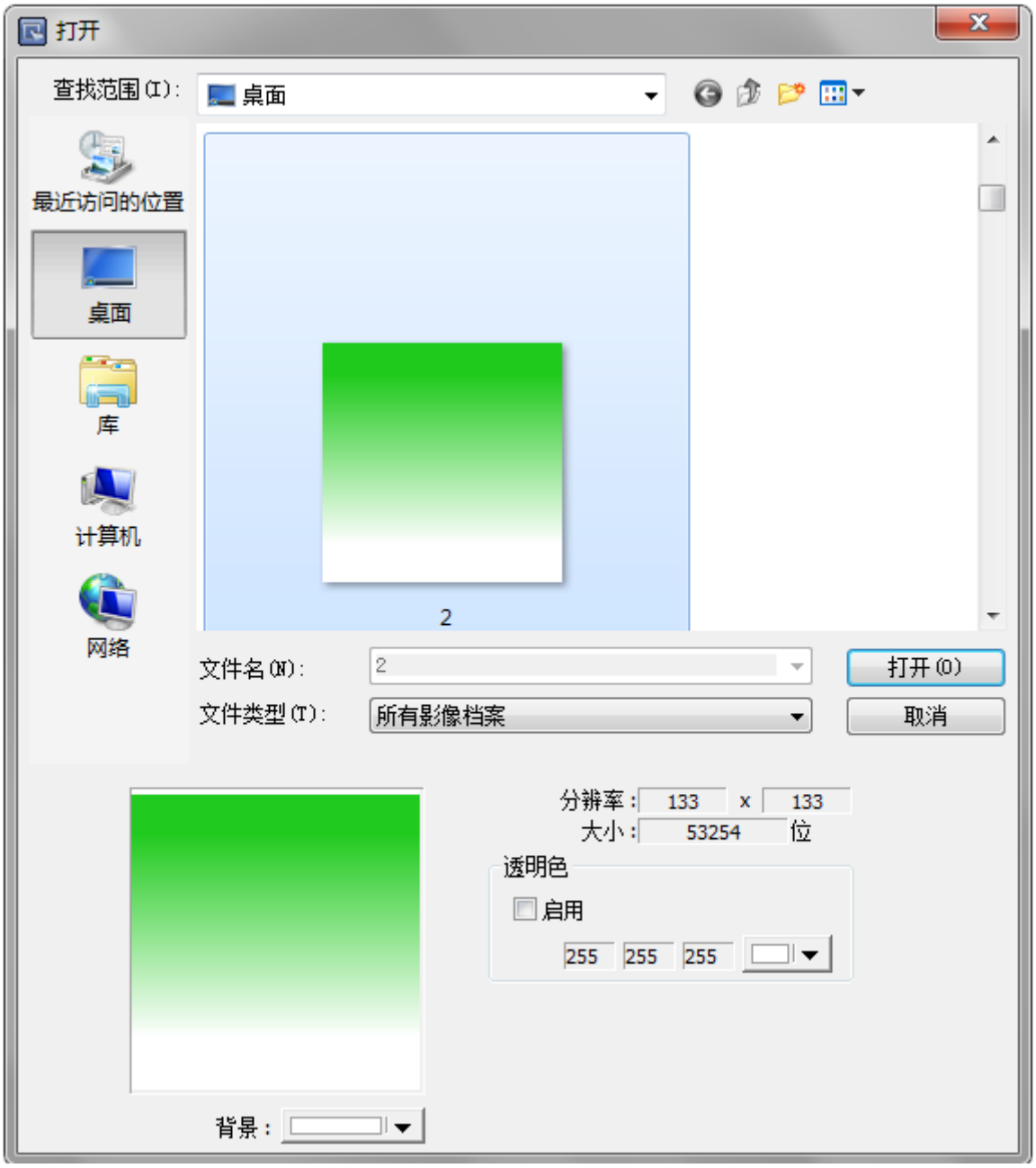
6. 按“新增”后，选择状态 0 的图片文件来源。

7. 此时出现下图的对话框时，若勾选“启用”来使用透明色，并设定 RGB (121, 121, 121)，会将此颜色区域设定为透明色，或是使用鼠标点击想要作为透明色的位置，系统将自动判断 RGB 的数值。



8. 要挑选透明色首先需先勾选“启用”，接着使用鼠标点击欲作为透明区域的位置，此时 会自动显示作为透明色的 RGB 值，实际显示的图片如上。

9. 此时已完成状态 0 的图片设定，接着“新增”状态 1 的图片，步骤与状态 0 相同。在“图片管理”对话框中新加入的图片为编号 0，由图片信息中也可看出此图片为 bitmap 形式，且包含两个状态，如下图所示。



10. 在完成上述的各项动作后，按下“确定”即可建立一个完整的图片。

Note

- 仅 .bmp、.dpd 和 .jpg 文件可使用透明色。

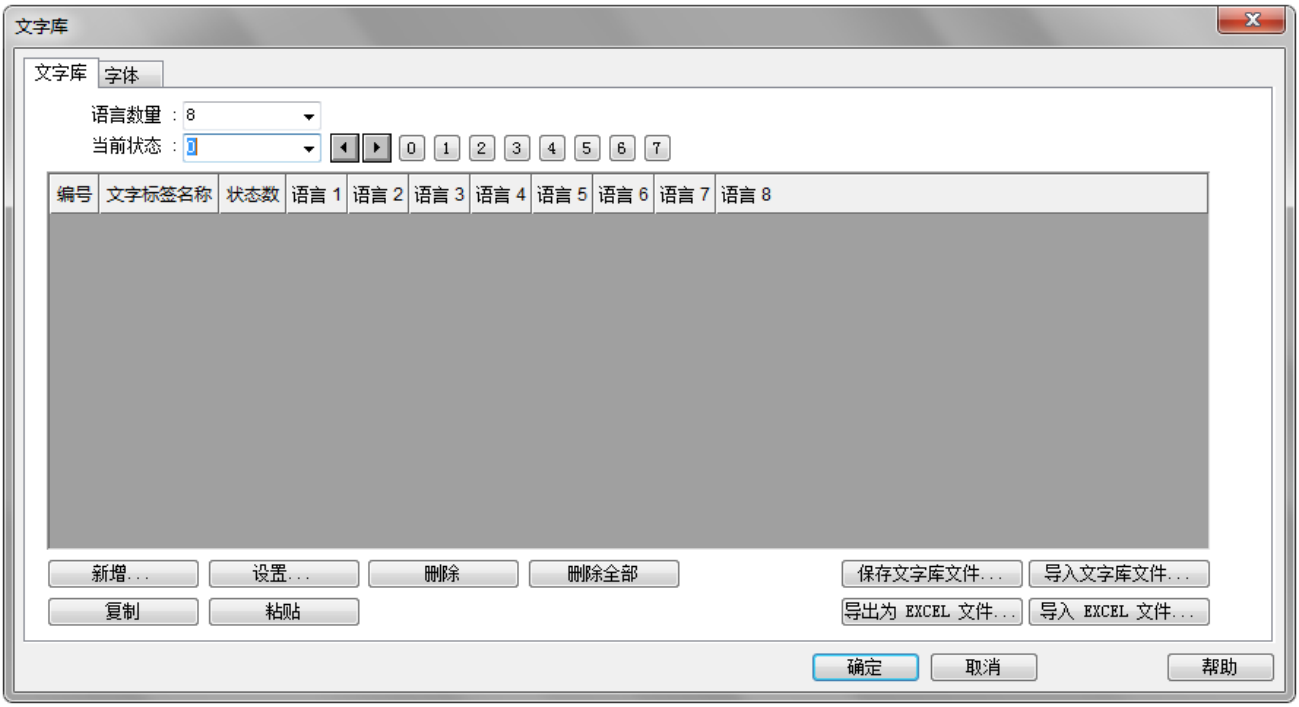
第十五章 文字标签库与多国语言使用

15.1 概要

当需要在工程文件中使用多国语言时，可以先建立文字标签库后，再从中选择需要的标签。系统在运作时会依照语言模式设定，在工程文件中显示所选择语言模式相对应的文字。EasyBuilder Pro 同时支持 8 种不同语言的文字显示。本章节将说明如何建立文字标签库并使用多国语言。

15.2 文字标签库管理

按下工具列上的“图库”»“文字库”即可进入“文字标签库”对话框如下图所示。



设定	描述
语言数量	设定本工程文件所使用的语言数量。
目前状态	一个文字标签最多可拥有 256 个状态 (0 ~ 255)。 状态数量将受“语言数量”使用的限制，若使用 1 ~ 3 个语言，每个语言最多可有 256 个状态，若使用 4 个以上的语言数量，则可用 768 去除以语言数量后，即可得到最多状态数。

	例如: 语言数量为 24，则 768/24 = 32 (状态数)。
新增	新增一笔文字标签。
设定	设定所选文字标签内容。
储存文字库文件	储存所有文字标签为 .lbl 格式文件。
导入文字库文件	将现存文字标签 .lbl 文件导入文字标签库。
导出为 EXCEL 文件	储存所有文字标签为 .csv 或 .xls 格式文件。
导入 EXCEL 文件	将现存文字标签 .csv 或 .xls 文件导入文字标签库。

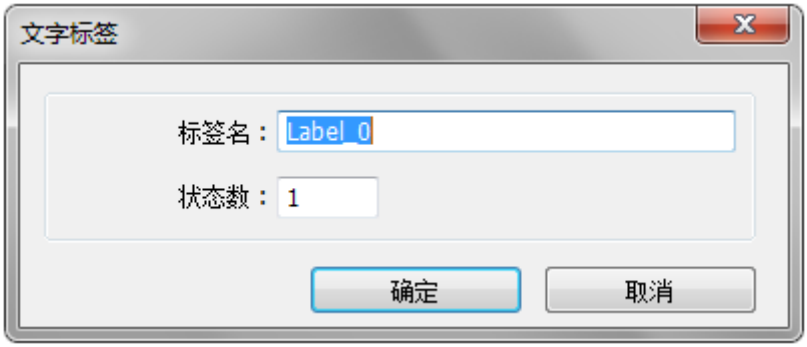
Note

- 导入或导出 Excel 文件均不支持 Unicode。

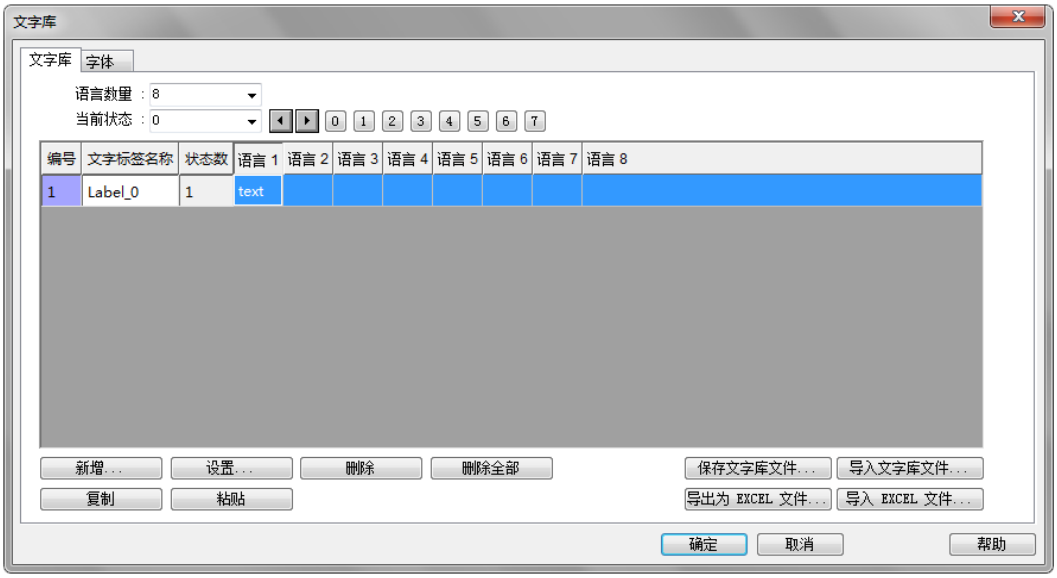
15.3 文字标签库的建立

请依照以下步骤建立文字标签库。

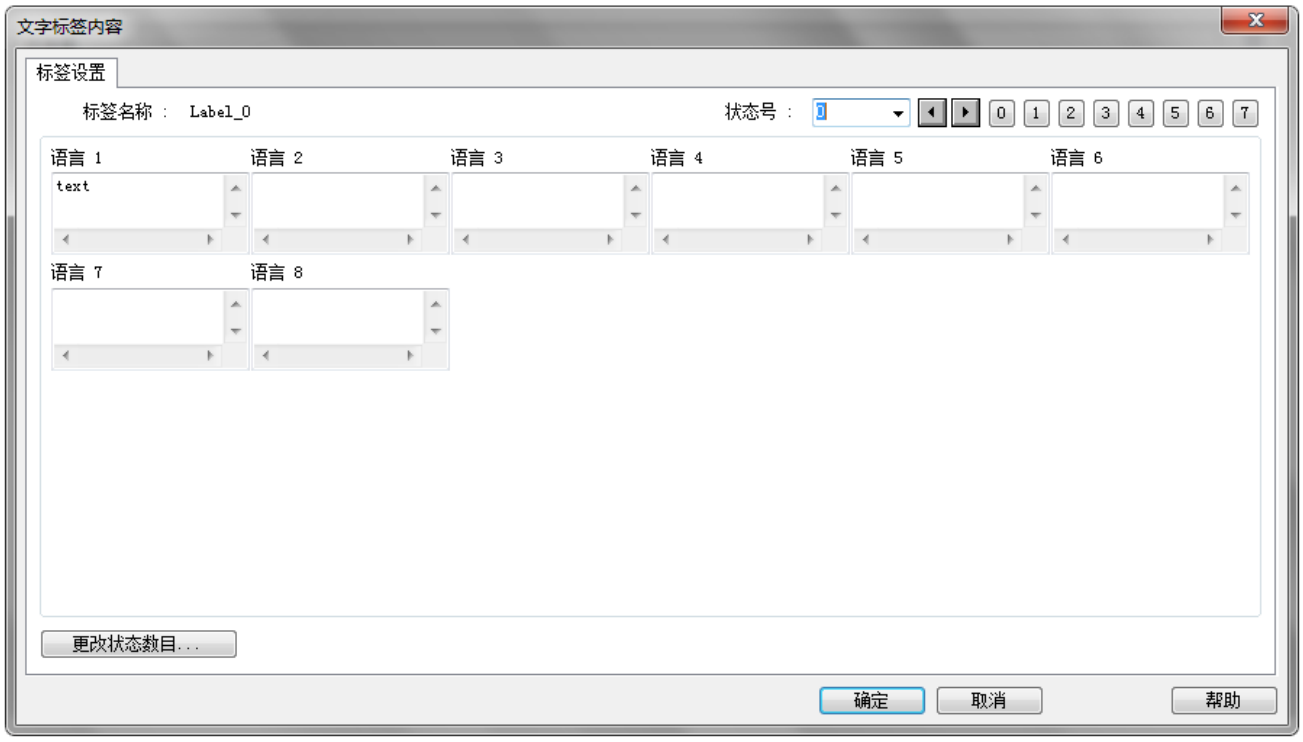
1. 开启“文字标签库”»“新增”。定义文字标签的名称并设定文字标签所要表现的状态数。



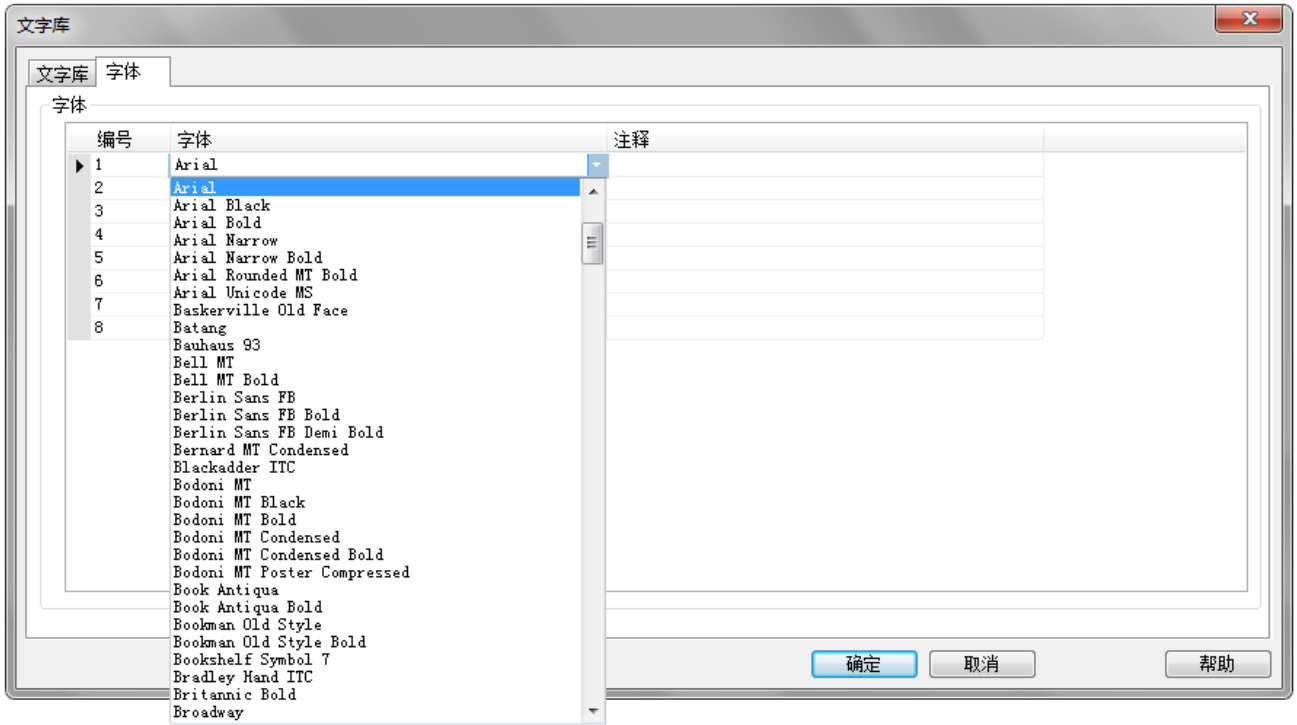
2. 按下“确定”会显示一个新的空白标签，选择该标签后按“设置”即可编辑标签内容。



3. 设定相关语言的内容。



4. 在“文字标签库”»“字体”页面中可以设定目前已经存在的标签所包含的语言字型，不同语言可选择不同的字型，如有需要，亦可为每种字型写下注释。



15.4 文字标签库的使用

当文字标签库存在已定义完成的标签，在元件的“标签”页面中可以勾选“使用文字标签库”，在“标签”的下拉选单会列示出已定义的文字标签。

选择标签后，可以发现“内容”中的文字来源所显示为文字标签的内容，所使用的字体为文字标签库的设定内容。请注意，语言 2 ~ 24 的文字属性设定只有文字尺寸是可被单独设定的，其余属性设定，包含文字颜色、对齐和闪动等，均与语言 1 相同。

15.5 多国语言的使用

当元件的文字内容要求表现出多国语言的效果时，除了使用文字标签外，也需搭配系统寄存器“LW-9134”的使用。

“LW-9134”的有效可设定值范围为 0 ~ 7，不同的数据对应到需显示的语言，由于 EasyBuilder Pro 可以设定 24 种语言，但是在触摸屏上最多能够显示 24 种语言的其中 8 种。当编译下载的文件没有勾选全部语言时，”LW-9134”使用方式将有改变。

例如用户建立 5 种语言：

1: 英文 2: 繁体中文 3: 简体中文 4: 法文 5: 韩文

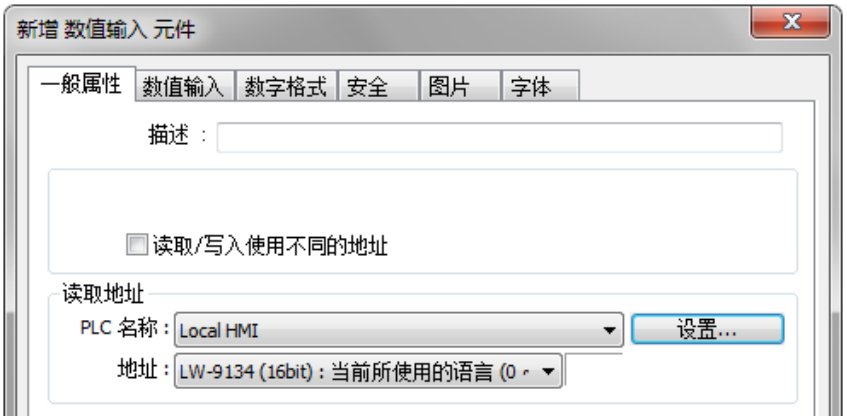
而编译时只勾选语言 1, 3, 5 则”LW-9134”对应数值为：

0: 英文 1: 简体中文 2: 韩文

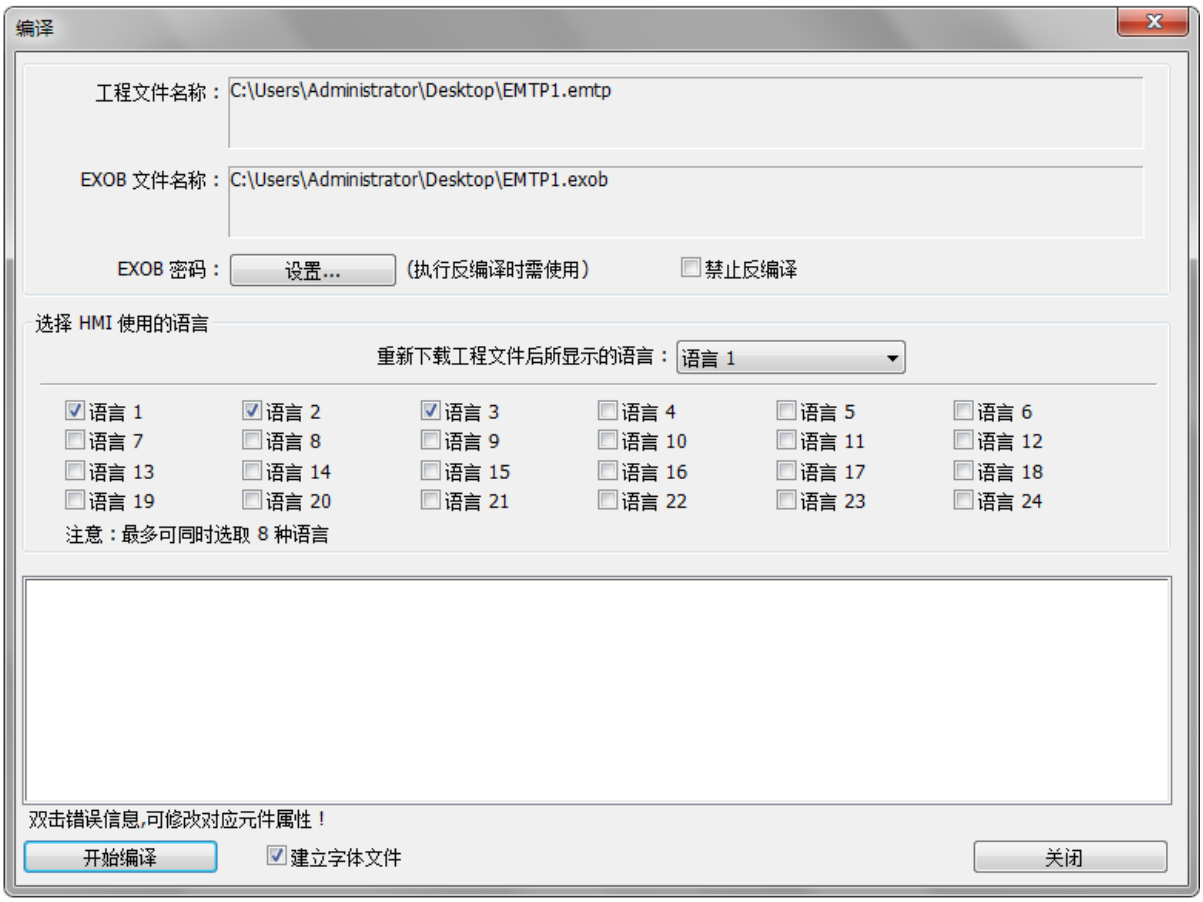
请依照下列步骤使用多国语言。

1. 先建立一个“文字元件”，勾选“使用文字标签库”。

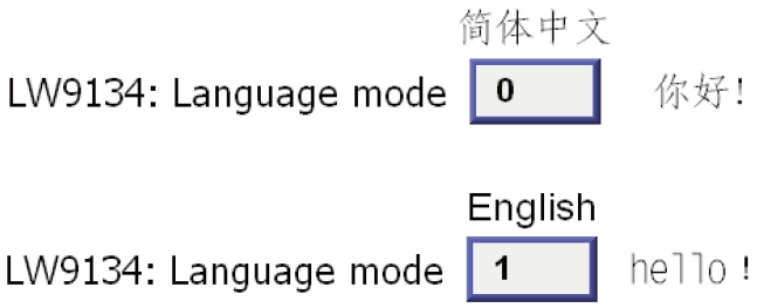
2. 再建立一个“数值输入元件”，地址设为系统寄存器“LW-9134”。



3. 编译时勾选需要并已定义的语言。



4. 模拟如下，当更改“LW-9134”的内容时，即可变换文字元件所显示的内容。



Note

- 最多同时可以选择 8 种语言下载至 HMI。
- 若 HMI 型号选择 cMT 时，”LW-9134” 是指系统语言模式，是修改 cMT 上的语言，”PLW-9134” 是修改 iPad 上的语言。

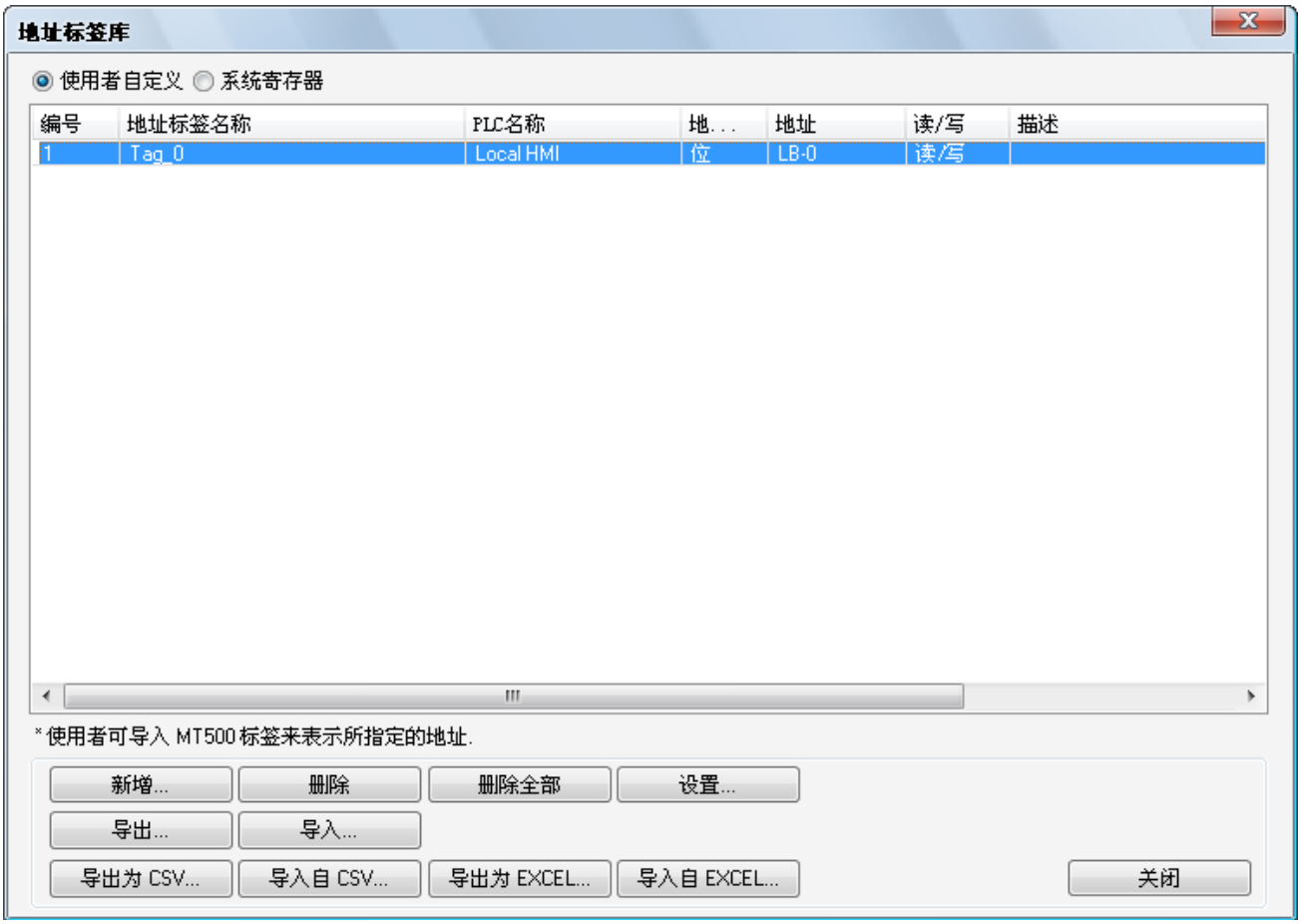
第十六章 地址标签库的建立与使用

16.1 概要

一般来说，建议用户在程序设计开始时，先将常用的地址定义在地址标签库中，除了可以省去繁复的地址输入外，也可以增加元件地址信息的可读性。

16.2 地址标签库的建立

按下工具列上的“图库”»“地址标签库”即可进入“地址标签库”对话框如下图所示。



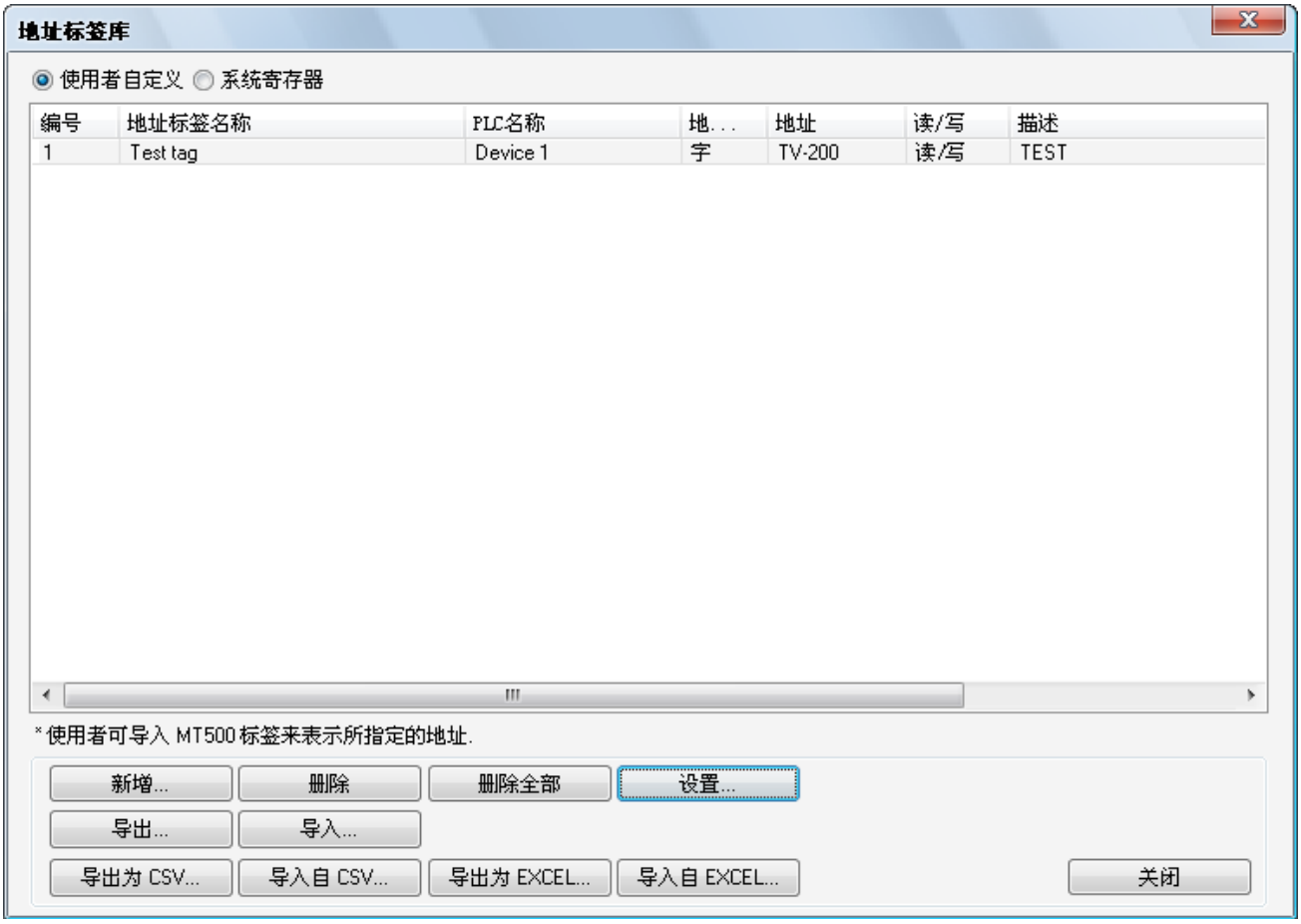
设定	描述
用户定义	显示用户自订标签。
系统寄存器	显示系统保留地址标签，系统寄存器不能被删除或更改。

新增	如何增加一个地址标签，请见下页说明。
设定	更改所选地址标签的内容。
输出	使用 .tgl 格式将地址标签库的所有内容输出并储存。
导入	将已存在且为 .tgl 格式的地址标签文件导入到目前的工程文件。
输出为 CSV	使用 .csv 格式将地址标签库的所有内容输出并储存。
导入自 CSV	将已存在且为 .csv 格式的地址标签文件导入到目前的工程文件。
输出为 EXCEL	使用 .xls 格式将地址标签库的所有内容输出并储存。
导入自 EXCEL	将已存在且为 .xls 格式的地址标签文件导入到目前的工程文件。

1. 按下“新增”后，即可设定相关属性。

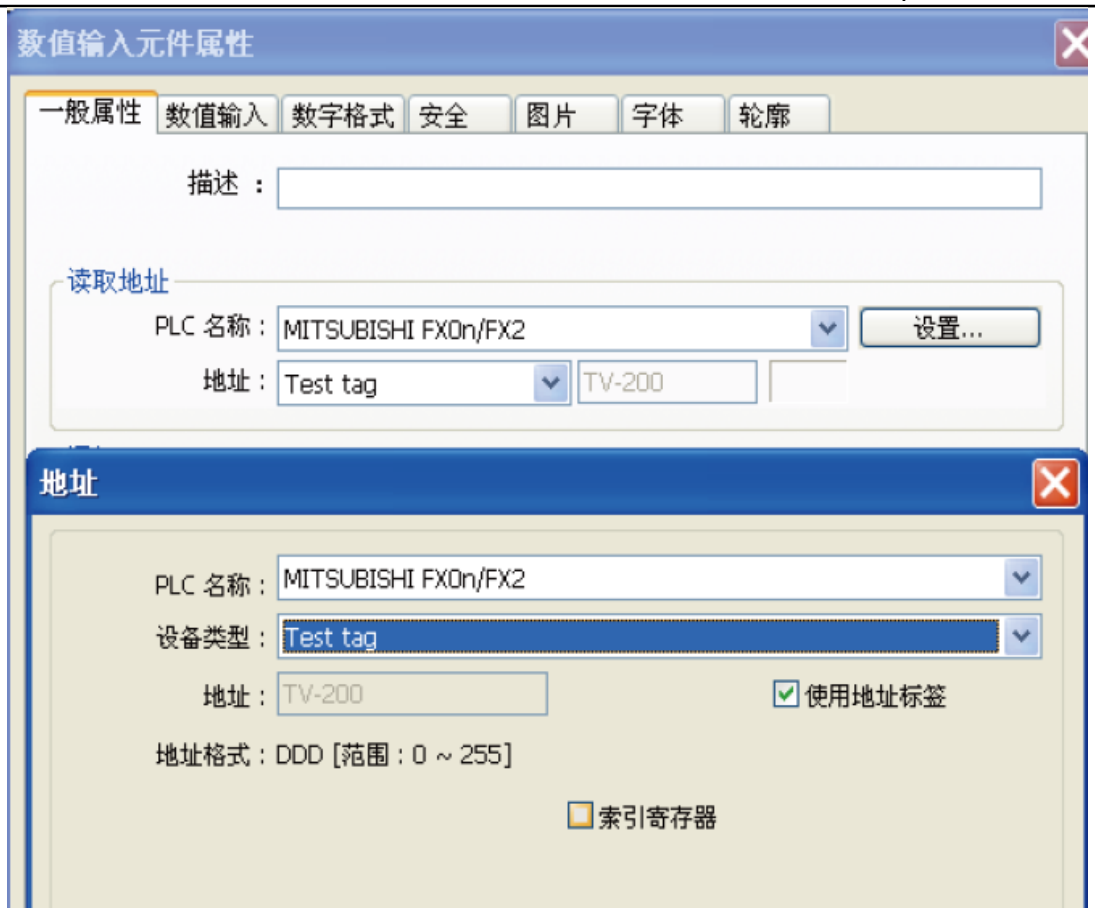
设定	描述
描述	为各个标签输入可读信息。
标签名称	设定标签的名称。
PLC 名称	可设定的内容来自”系统参数设置” »“设备清单”页面。
地址类型	可选择“位”或“字符”类型。
设备类型	可选择类型与“PLC 名称”、”地址类型”的设定有关。
地址	设定标签地址。

2. 按下“确定”后，可在“用户定义”地址标签中发现一笔新增的地址标签。



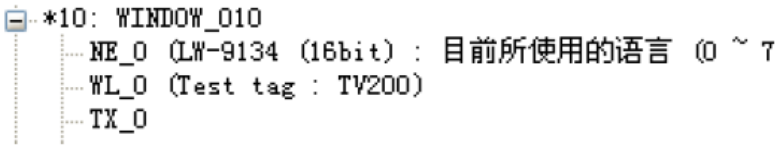
16.3 地址标签库的使用

1. 设定地址标签库的相关定义。
2. 新增元件，于地址属性选择相关“PLC 名称”。
3. 点选“设置”可弹出地址设定窗口。
4. 勾选“使用地址标签”。



5. 在“设备类型”选用自订标签。

6. 设定完成后，可在元件信息树形图检视所使用的地址标签名称。



第十七章 配方数据传送

17.1 概要

所谓配方数据是指存在 RW 与 RW_A 地址上的数据，读写这些地址的方式与读写一般字符地址的方式并无不同，配方数据的特性在于关机后这些数据将保存在触摸屏闪存上，重新开机后 RW 与 RW_A 地址上的数据将维持前一次记录的内容。

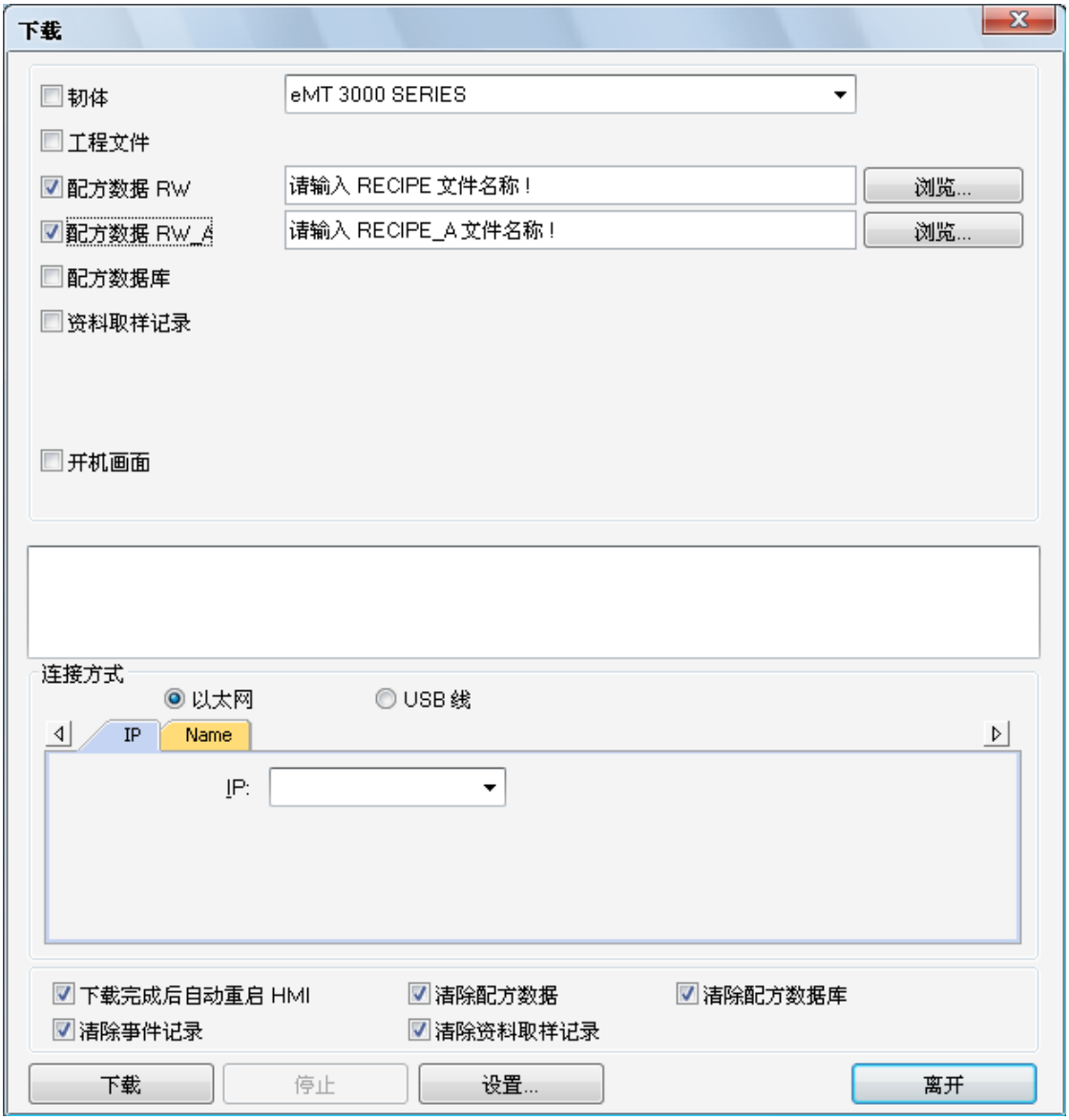
RW 地址可储存的配方数据大小为 512 K words，而 RW_A 地址则为 64 K words，用户可以通过 SD 卡、U 盘、USB 线或以太网更新配方数据，并利用这些数据更新 PLC 上的数据，同样地也可以上传配方数据至计算机；此外，用户亦可以将 PLC 上的数据保存在配方数据中。下文将针对配方资料的各种操作做说明。

17.2 使用以太网或 USB 线更新配方数据

1. 通过 Utility Manager 选择“下载”功能。
2. 勾选“RW”与“RW_A”后选择要下载的文件来源。
3. 下载成功后重新启动 HMI，即可更新 RW 与 RW_A 的内容。

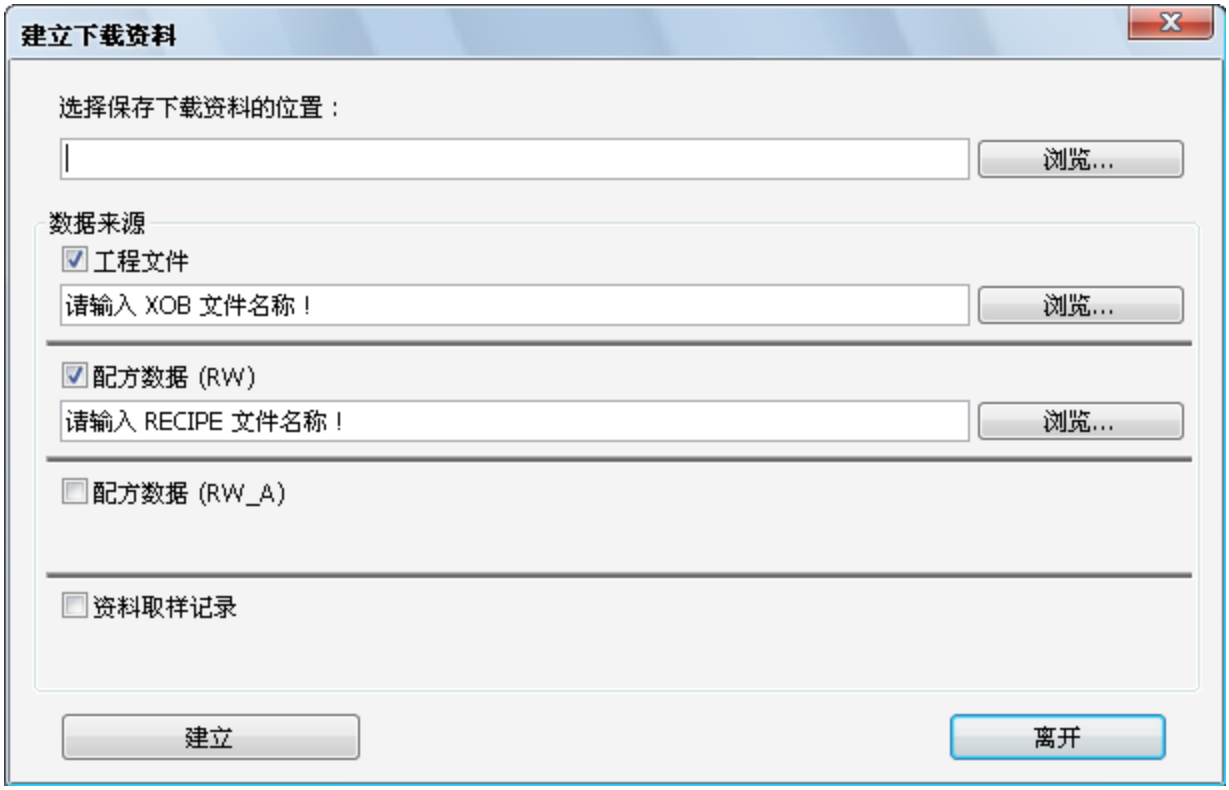
若勾选“下载完成后自动启动 HMI”，可免去手动重启触摸屏的步骤。

若勾选“清除配方数据”，在进行任何下载动作前，系统会先将“RW”与“RW_A”上的数据内容全部清除。



17.3 使用 SD 卡或 U 盘更新配方数据

1. 在 Utility Manager 点选“建立使用 U 盘或 SD 卡所需的下载资料”。
2. 将 SD 卡或 U 盘插入计算机。
3. 点选“浏览”指定文件数据所要存放的路径名称。
4. 点选“建立”，EasyBuilder Pro 会自动把所要建立的来源数据文件写入到 SD 卡或 U 盘里。

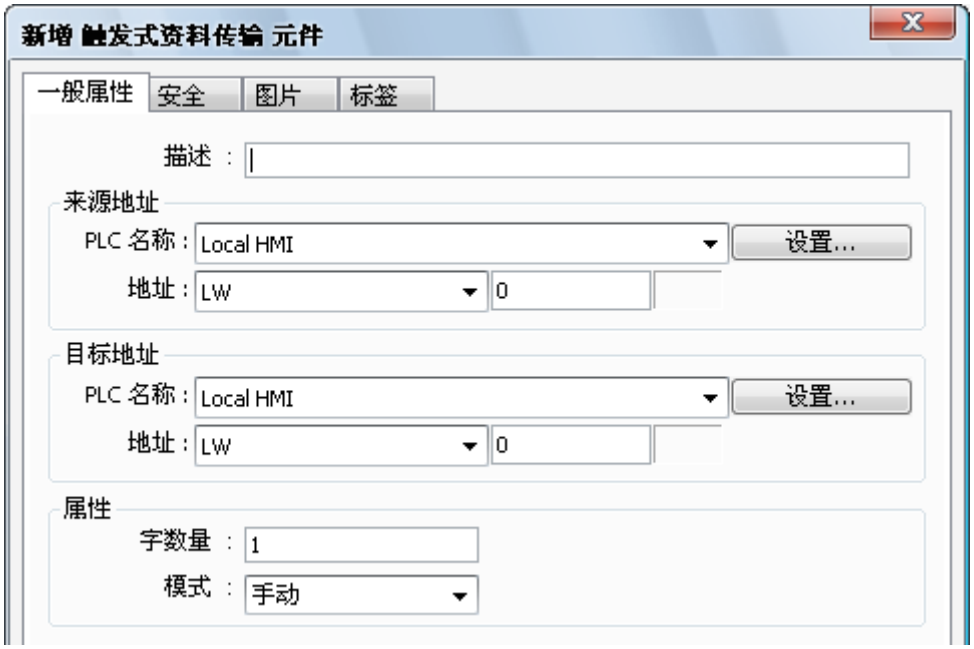


Note

- 建立完成后将可发现两个新文件夹，分别是 history 和 emt3000。emt3000 是存放工程文件的文件夹，而 history 则是存放配方数据及资料取样/事件记录的文件夹。

17.4 配方数据传输

可以通过“触发式数据传输”元件，将配方数据传送到特定地址；也可以将特定地址的数据保存在“RW”与“RW_A”中。



设定	描述
来源地址	设定数据传送的来源地址。
目标地址	设定数据传送的目标地址。
属性	设定自来源地址传送到目标地址的数据之字符个数。

17.5 配方数据储存

为了延长触摸屏上的闪存使用寿命，系统以每隔 1 分钟的时间间隔将配方数据保存在触摸屏上，且为了避免配方数据在两次储存动作间因关机而造成数据的流失，EasyBuilder Pro 提供系统寄存器 “LB-9029: 强迫储存配方数据到 HMI”，只需对其送出 ON 的讯号， 系统即会执行一次配方数据储存动作。另外如果对 “LB-9028: 重置配方数据” 送出 ON 的讯号，则会将所有的配方数据清除。

第十八章 宏指令说明

18.1 概要

宏指令提供了应用程序之外所需的附加功能。在触摸屏触摸屏界面运行时，宏指令可以自动的执行这些命令。它可以担负执行譬如复杂的运算、字符串处理，和用户与工程之间的交流等功能。本章主要介绍宏指令的语法、如何使用和编辑方法等功能。希望通过本章的说明，能够使各位能够快速的掌握 EasyBuilder Pro 软件提供的强大的宏指令功能。

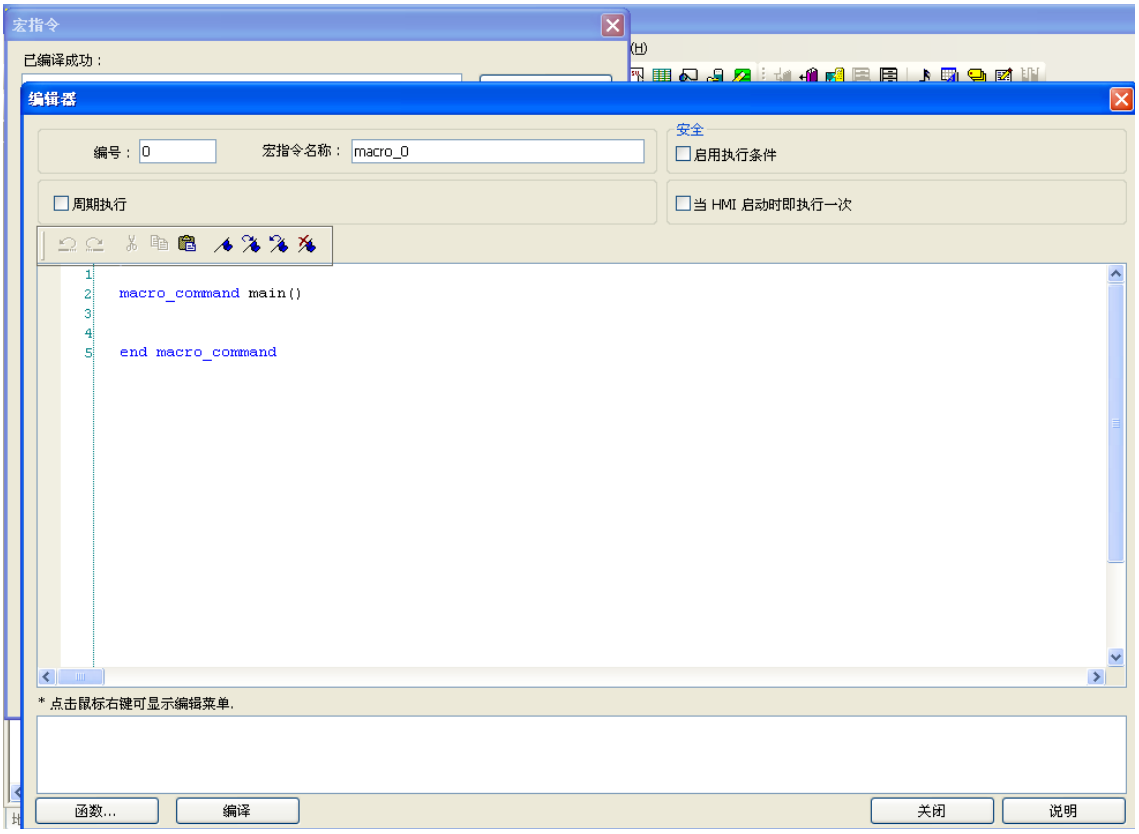
18.2 宏指令编辑器功能使用说明

宏指令编辑器提供下列新功能：

- 显示行号
- 复原 (Undo) / 重复 (Redo)
- 剪下 (Cut) / 复制 (Copy) / 贴上 (Paste)
- 全选 (Select All)
- 建立 / 取消书签 (Toggle Bookmark) / 上一个书签 (Previous Bookmark) / 下一个书签 (Next Bookmark) / 清除全部书签 (Clear All Bookmarks)
- 程序代码折迭 (Toggle All Outlining)
- 安全 -> 启用执行条件
- 周期执行
- 当触摸屏启动时即执行一次

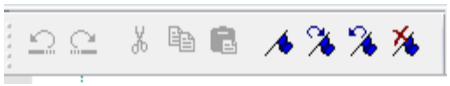
以下将详细描述如何使用各项功能。

1. 打开宏指令编辑器，可以看到编辑区左边将自动显示行号。

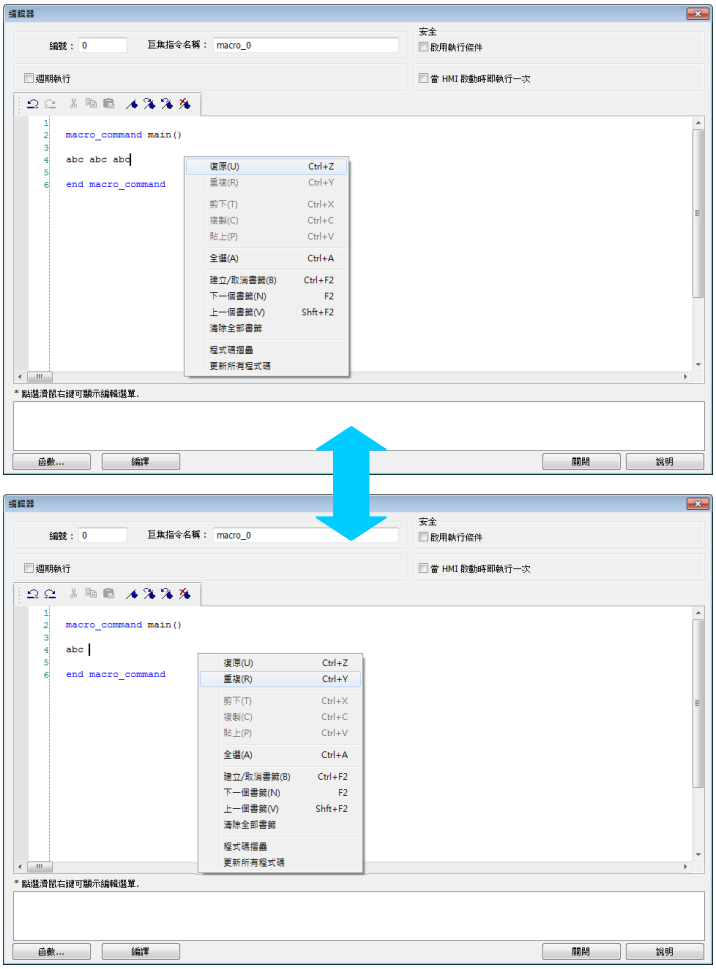


2. 编辑区中按鼠标右键，呼叫出右键选单如下图。目前的状态无法使用的功能将显示灰色。例如必须在编辑区中选取一段文字才会开启复制功能，因此未选取任何文字的状态下，复制功能暂不开启。提供快速键，如选单内所提示。

3. 编辑区上方有工具列，提供“复原”、”重复”、”剪下”、”复制”、”贴上”、”建立/取消书签”、”下一个书签”、”上一个书签”、”清除全部书签”等按钮方便快速选取。

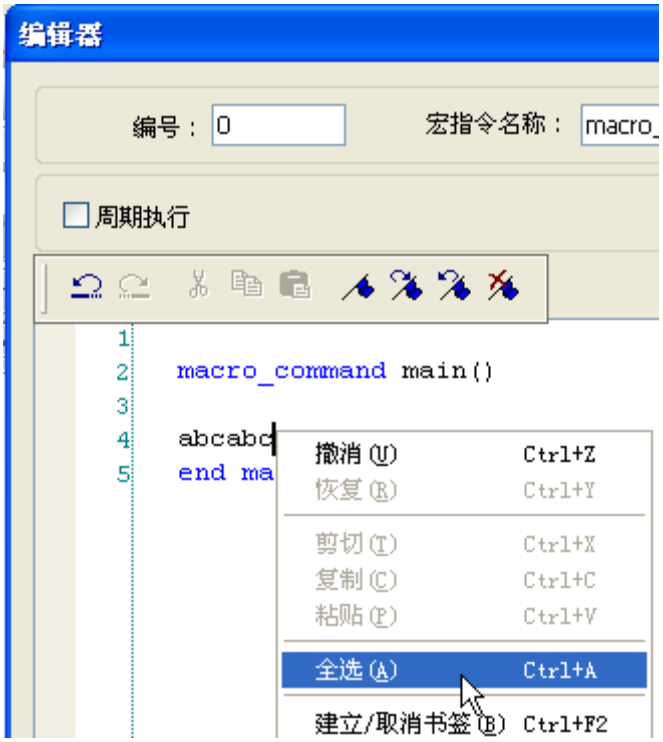


4. 改变编辑区内容将开启“复原”功能，用户执行复原后可用“重复”复原。用户可从右键选单或是利用热键 (Undo：Ctrl+Z，Redo：Ctrl+Y) 执行此功能。



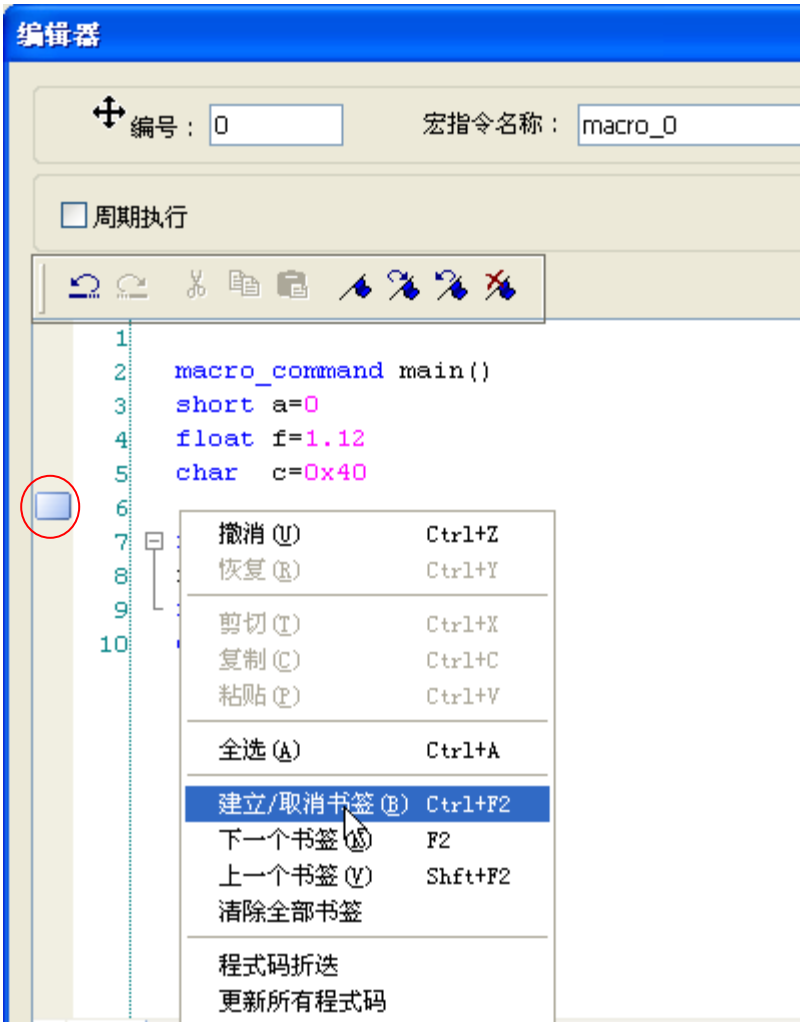
5. 在编辑区选取一段文字后可进行“剪下”、”复制”，之后可用“贴上”将选取的文字贴上。

6. 选择“全选”可选取编辑区全部内容。

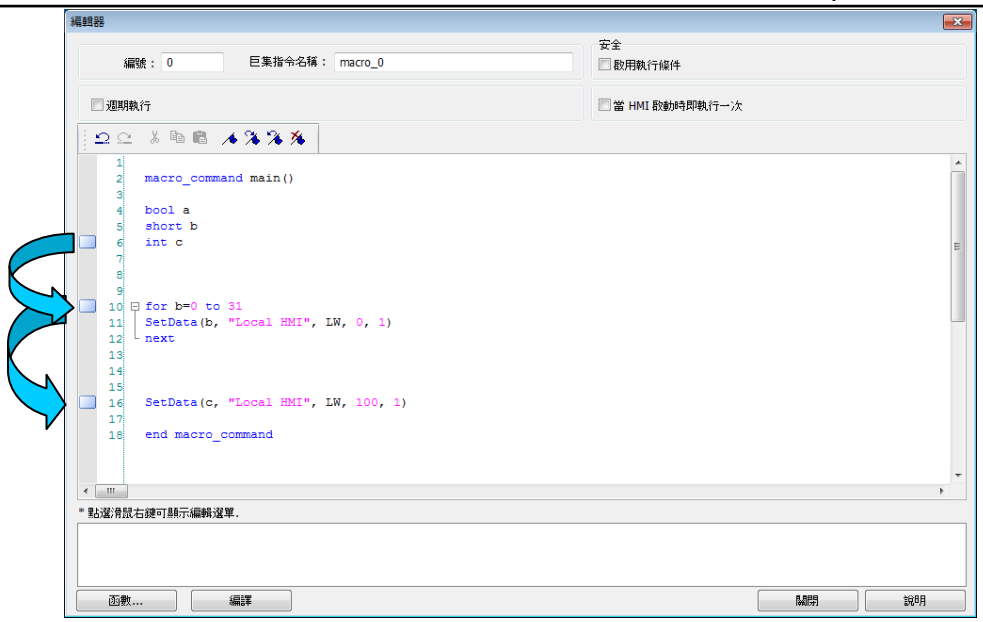


7. 当程序代码很长的时候，为方便用户阅读，提供了书签功能。下面说明如何使用此功能。

- 将光标移至编辑区中想要插入书签的位置，按右键，选择“建立 / 取消书签”。编辑区左边将会看到一个代表书签的蓝色小方块。



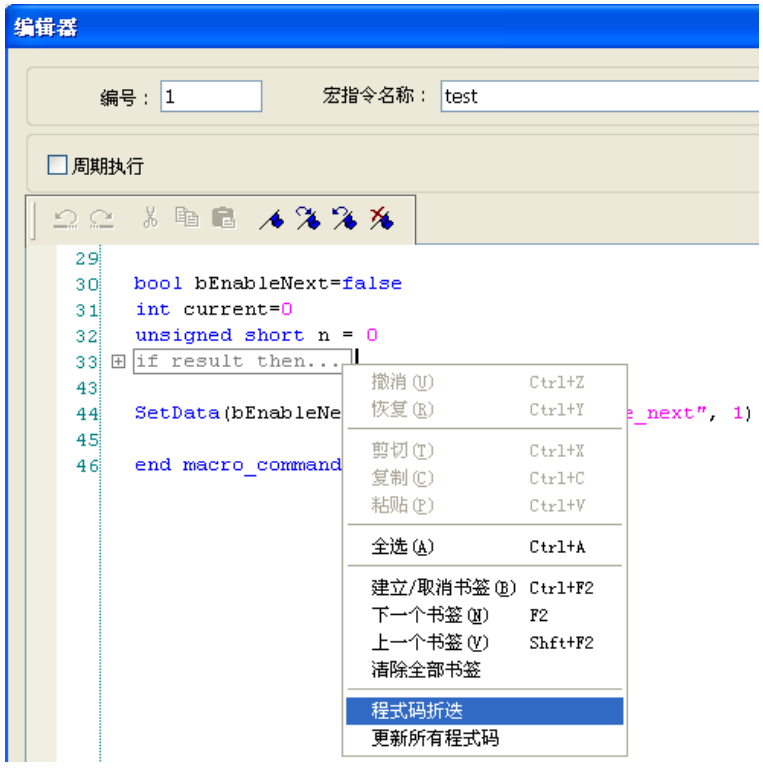
- 若光标所在位置已存在书签，选择“建立 / 取消书签”可将其关闭，反之，选择“建立 / 取消书签”可将其开启。
- 右键选择“下一个书签”光标将会移至下一个书签所在位置。选择“上一个书签”光标将会移至上一个书签所在位置。

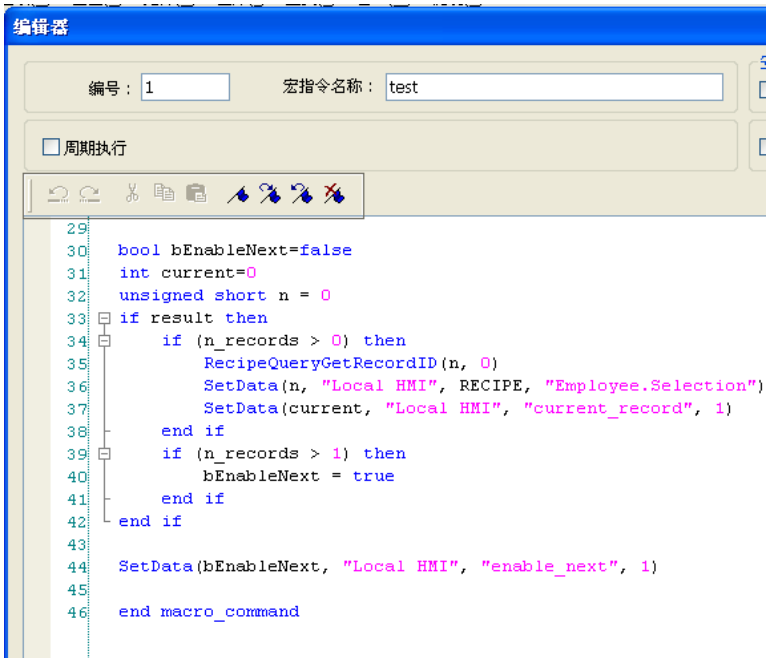


- 选择“清除全部书签”将关闭所有书签。

8. 宏指令编辑器提供程序代码折迭功能，方便用户浏览程序代码。所谓程序代码折迭，是指编辑器可将属于同一区块的程序代码隐藏起来，被隐藏起来的程序代码在编辑区里会显示成。编辑 ⊞ ... 区左侧会显示树形图，用户可按下⊟隐藏程序区块，按下⊞展开程序区块。如下图所示：

9. 右键选择“程序代码折迭”可展开所有程序代码区块。



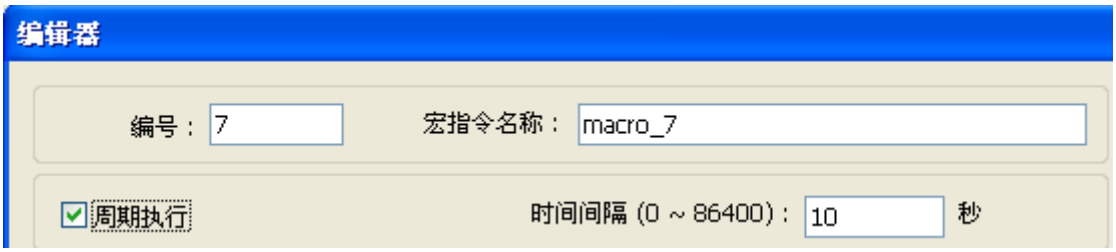


10. 有时候程序代码区块可能会误判。这种误判起因于编辑器没有办法区分当前输入的关键词是否存在于注释中。例如下图。用户可以从右键选单选择“更新所有程序代码”来更正这个错误。

11. 包围在特定关键词内的程序代码称为一程序代码区块。内定的程序代码区块如下列：

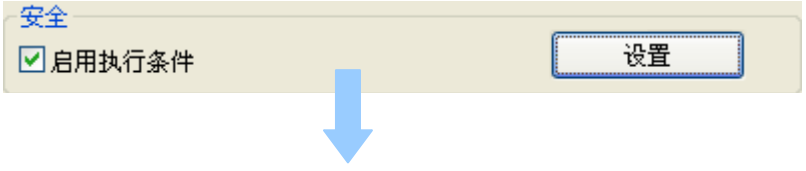
- 子函数： sub – end sub
- 循环语句：
  i. for – next
  ii. while – wend
- 逻辑运算语句：
  i. if – end if
- 多重判断语句： select case – end select

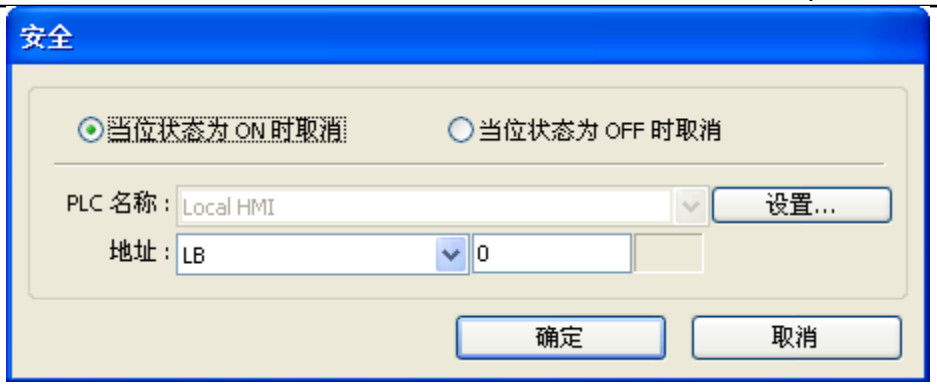
12. 用户勾选“周期执行”时，会周期性的触发此宏。



13. 用户勾选“安全”-> “启用执行条件”-> “设置”后,可以进行安全设定：

- 当位状态 ON 时取消:当位状态 ON 时禁止执行此宏。
- 当位状态 OFF 时取消: 当位状态 OFF 时禁止执行此宏。





14. 用户勾选“当 HMI 启动时即执行一次”时，在触摸屏启动时会自动执行宏一次。

18.3 宏指令的结构

宏指令是由各种语句组成的。这些语句包含常数、变量和各种运算符号。这些语句放置在特定的顺序位置以便执行后达到一个希望的执行结果。

宏指令的结构一般为以下格式：

```
全局变量声明                          ----------可选

Sub Function Block Declarations(子函数声明) ----------可选
    局部变数声明
End Sub(结束子函数)

macro_command main()    [主函数]          ----------必须
    局部变数声明
    [各式语句]
end macro_command       [结束主函数]      ----------必须
```

一个宏指令必须有一个且只有一个主函数，用来开始宏指令的执行。格式为：

macro_command 函数名称()

end macro_command

变量声明必须放在宏指令语句的前面，否则如果语句放置在变量声明的前面，将会造成宏指令无法编译通过。
局部变量一般用在宏指令主函数或者自定义的子函数中。它的合法性只在指定的函数中有效。
全局变量一般是定义在所有宏指令函数的前面，且它在整个宏指令中均具有有效性。当局部变量和全局变量被定义为相同的名称时，只有局部变量有效。

下面就是一个简单的宏指令，其中就包含了变量声明和函数呼叫。双斜线 "//" 代表程序批注，在它后面的文字不会被执行。

```
macro_command main()
    short pressure = 10                                    //   局部变数声明
    SetData(pressure, "Allen-Bradley DF1", N7, 0, 1)       //   函数呼叫
end macro_command
```

18.4 宏指令的语法

18.4.1 常数和变数

常数

常数是一个可以被各式语句直接使用的固定的资料。有如下格式：

常数类型	使用说明	举例
十进制整数		345, -234, 0, 23456
十六进制数	必须以 0x 开头	0x3b, 0xffff, 0x237
字符型	字符必须使用单引号，字符串使用双引号	‘a’, "data", "函数名称"
布尔型		true, false

下面即为一个简单的常数使用的范例。

```
macro_command main()
    short A, B   //   声明 A 和 B 为短整型变数
    A = 1234
    B = 0x12     //   1234 和 0x12 即为常数
end macro_command
```

变数

变量是一个代表着各种资料的名称。在宏指令中，这些资料可以随着宏指令语句执行的结果改变而改变。

变量的命名规则

- 必须以英文字母开头
- 变量名称长度不超过 32 个字符
- 系统保留寄存器名称不能作为变量名称。

下面为 8 种不同的变量类型，前 5 种为有号数值类型，后 3 种为无号数值类型：

变量类型	描述	范围
bool 布尔型	1 bit (一个位)	0, 1
char 字符型	8 bits (一个字节)	+127 ~ -128
short 短整型	16 bits (一个字符)	+32767 ~ -32768
int 双整型	32 bits (双字符)	+2147483647 ~ -2147483648
float 浮点型	32 bits (双字符)	
unsigned char 字符型	8 bits (一个字节)	0 到 255
unsigned short 短整型	16 bits (一个字符)	0 到 65535
unsigned int 双整型	32 bits (双字符)	0 到 4,294,967,295

变数声明

变量必须在使用前声明。所以，在宏指令，所有的变量都必须在语句使用前都被声明完成。声明变量时，先定义变量的类型，后面再跟着变量名称。

如下范例:

```
int          a
short        b, switch
float        pressure
unsigned short    c
```

数组声明

宏指令支持一维数组 (下标从 0 开始)。声明数组变量时，先定义数组变量的类型，变量名称，接着就是该数组变量的个数，变量个数必须放置在“”符号中。数组变量的长度为 1 ~ 4096。一个宏指令中最多只支持 4096 个变量。

如下范例:

```
int          a[10]
short        b[20], switch[30]
float        pressure[15]
```

数组的下标最小为 0，最大下标为(数组的长度-1)

如下范例:

```
    char data[100]          //   数组变量的长度是 100
```

所以: 最小的数组为 “data”0””，最大的数组为 “data”99””，即 100 - 1 = 99。

变量和数组初始化

有两种方法可以让变量初始化：

- 使用语句中的赋值语句 (=)

如下范例:

```
int a
float b[3]
a = 10
b[0] = 1
```

- 声明变量时直接赋值

```
char a = '5', b = 9
```

数组变量的声明是一个特殊的情况。一个完整的数组被初始化时，可以在数组变量声明时，将数据放置在波形括号**"{}"**里面，各数据使用逗号分开。

如下所示:

```
float data[4] = {11, 22, 33, 44}   // 这样 data[0] = 11, data[1] = 22....
```

18.4.2 运算符号

运算符通常被用来指定数据是如何被操作和运算，如下：(在任何一个语句中，运算符左边的变量结果均依据运算符右边的条件而获得。)

运算符号	描述	举例
=	赋值运算符号	pressure = 10

数学运算符号	描述	举例
+	加	A = B + C
-	减	A = B - C
*	乘	A = B * C
/	除	A = B / C
%	求余 (返回剩余数)	A = B % 5

比较运算符号	描述	举例
<	小于	if A < 10 then B = 5
<=	小于或者等于	if A <= 10 then B = 5
>	大于	if A > 10 then B = 5
>=	大于或者等于	if A >= 10 then B = 5
==	等于	if A == 10 then B = 5
<>	不等于	if A <> 10 then B = 5

逻辑运算符号	描述	举例
And	与	if A < 10 and B > 5 then C = 10
Or	或	if A >= 10 or B > 5 then C = 10
Xor	异或	if A xor 256 then B = 5
Not	非	if not A then B = 5

移位元和位运算符号通常被用来操作字符型变量、短整型变量和双整型变量的位。在一个语句中，这些运算符号的优先权是在从该语句的左边到右边依此执行的。即在语句中左边位置的优先执行，依次从左到右执行。

移位运算符号	描述	举例
<<	往左移动指定的位数	A = B << 8
>>	往右移动指定的位数	A = B >> 8

位运算符号	描述	举例
&	位与运算	A = B & 0xf
|	位或运算	A = B | C
^	位异或运算	A = B ^ C
~	位取反运算	A = ~B

所有运算符号的优先权

上述所有运算符号的优先权从高到低详细如下所述：

1. 位于圆括号里面的运算符号最优先
2. 数学运算符号
3. 移位和位运算符号
4. 比较运算符号
5. 逻辑运算符号
6. 赋值运算符号

关键词

下面的关键词为宏指令保留使用。这些均不能用来作为变量名称、数组名或者函数名称等。

+, -, *, /, %, >=, >, <=, <, <>, ==, and, or, xor, not, <<, >>,=, &, |, ^, ~

exit, macro_command, for, to, down, step, next, return, bool, short, int, char, float, void, if, then, else, break, continue, set, sub, end, while, wend, true, false

SQRT, CUBERT, LOG, LOG10, SIN, COS, TAN, COT, SEC, CSC, ASIN, ACOS, ATAN, BIN2BCD, BCD2BIN, DEC2ASCII, FLOAT2ASCII, HEX2ASCII, ASCII2DEC, ASCII2FLOAT, ASCII2HEX, FILL, RAND, DELAY, SWAPB, SWAPW, LOBYTE, HIBYTE, LOWORD, HIWORD, GETBIT, SETBITON, SETBITOFF, INVBIT, ADDSUM, XORSUM, CRC, INPORT, OUTPORT, POW, GetError, GetData, GetDataEx,

SetData, SetDataEx, SetRTS, GetCTS, Beep, SYNC_TRIG_MACRO, ASYNC_TRIG_MACRO, TRACE, FindDataSamplingDate, FindDataSamplingIndex, FindEventLogDate, FindEventLogIndex
StringGet, StringGetEx, StringSet, StringSetEx, StringCopy, StringMid, StringDecAsc2Bin, StringBin2DecAsc, StringDecAsc2Float, StringFloat2DecAsc, StringHexAsc2Bin, StringBin2HexAsc, StringLength, StringCat, StringCompare, StringCompareNoCase, StringFind, StringReverseFind, StringFindOneOf, StringIncluding, StringExcluding, StringToUpper, StringToLower, StringToReverse, StringTrimLeft, StringTrimRight, StringInsert。

18.5 语句

18.5.1 定义语句

这个定义语句包含了变量和数组的声明。正式的格式如下：
定义一个变量的名称为"名称"且类型为"类型"。
举例:

```
int A       //定义了变量 A 为双整型格式
```

定义一个数组变量为"名称"，大小为"数组长度"且类型为"类型"时。
举例:

```
int B[10]  //定义了一维数组变量 B 的长度为 10，类型为双整型
```

18.5.2 赋值语句

赋值语句使用赋值运算符号将赋值运算符号右边表达式运算的结果放置到运算符号左边的变量中。一个表达式是由变量、常数和各种运算符号组成，执行后产生一个新的数据。
举例:

```
A = 2           //这样变量 A 就被赋值为 2
```

18.5.3 逻辑运算语句

逻辑运算语句是根据逻辑（布尔）表达式的结果来执行相应的动作。它的语句如下所示：
单行格式

```
If <Condition> then
[Statements]
else
[Statements]
end if
```

举例:

```
if a = = 2 then
        b = 1
else
        b = 2
end if
```

区块格式

```
If <Condition> then
[Statements]
else if <Condition-n> then
  [Statements]
else
[Statements]
end if
```

举例:

```
if a == 2 then
        b = 1
else if a = = 3 then
        b = 2
else
        b = 3
end if
```

语法描述

if	必须用在该语句的开始部分。
<Condition>	必要条件。这是一个控制语句。当 <Condition> 为 0 时，即为 “FALES”，(条件为假)；当 <Condition> 为非 0 时，即为“True”(条件为真)。
then	当 <Condition> 执行为 “TRUE” (真) 时，必须放置在需要执行的语句之前。
[Statements]	在区块形式中是可选择的参数，在单行形式中，且没有 else 子句时，为必要参数，该语句在 <Condition> 为真时执行。
else if	可选，一条或多条语句，在相对应的 <Condition - n> 为 true 时执行。
<Condition-n>	可选，解释同 Condition
else	可选，在上述 Condition 和 Condition - n 都不为 true 时执行。
end if	必须。在一个 if-then 语句中使用这个来结束 if-then 语句。

18.5.4 多重判断语句

Select-case 可用来处理多重判断的叙述，其功能类似 if-else 语句。根据所指定变量的值，分别对应到符合该值的 case，并执行 case 下面的叙述，直到遇到 break 叙述时，才跳到结束符号 end select 处。语法结构如下：

没有预设 case 的形式：

```
Select Case   [variable]
Case   [value]
      [Statements]
      break
end Select
```

举例:

```
Select Case A
      Case 1
            b=1
      break
end Select
```

有预设 case 的形式：

```
Select Case   [variable]
Case   [value]
      [Statements]
      break
      Case else
      [Statements]
      break
end Select
```

举例:

```
Select Case A
      Case 1
            b=1
      break
      Case else
            b=0
      break
```

```
end Select
```

多个不同 case 对应到相同区块：

```
Select Case  [variable]
Case  [value1]
      [Statements]
Case  [value2]
      [Statements]
     break

end Select
```

举例:

```
Select Case A
    Case 1
    Case 2
        b=2
    Case 3
        b=3
    break
end Select
```

语法描述

Select Case	必须用在该语句的开始部分。
“variable”	必要条件。此变量将会与每一个 case 做比较。
Case else	可选。代表预设 case。当 “variable” 的值不符合任何一个 case 时，将会执行此叙述下面的区块。在没有预设 case 的情况，当 “variable” 的值不符合任何一个 case 时，将不会做任何动作而直接跳出 select 控制结构。
break	可选。跳到某一个 case 下面执行时，将一句一句执行 case 语句下面的叙述直到遇到 break 命令才结束，并跳到 end select 叙述。当 case 叙述下面没有任何 break 命令时，流程将不断往下执行，直到遇到 end select 叙述，才结束并跳出 select 控制结构。
end Select	select-case 语句的结束标志。

18.5.5 循环语句

循环语句依据循环条件来反复的执行一个任务。循环语句有两种表达方式。

for next 语句

For-next 语句通常用来执行次数固定的循环任务。一个变量用作为任务执行次数的计数器和结束循环任务

执行的条件。这个变量为固定执行的次数。语法结构如下：

```
for  [Conunter] = <StartValue> to <EndValue> [step <StepValue>]
   [Statements]
next  [Counter]
```

或者

```
for  [Conunter] = <StartValue> down <EndValue> [step <StepValue>]
   [Statements]
next  [Counter]
```

举例:

```
for a = 0 to 10 step 2
    b = a
next a
```

语法描述

for	必须用在该语句的开始部分。
[Counter]	必要，循环计数器的数值变量，该变量的结果用来计数循环的次数。
<StartValue>	必要，Counter 的初值。
to/down	必要。用来决定步长是递增还是递减。 “to” 以 <StepValue> 为步长递增 <Counter> “down” 以 <StepValue> 为步长递减 <Counter>
<EndValue>	必要，Counter 的终值、测试点。当 <Connter> 大于该值时，宏指令将结束这个循环任务。
step	可选，指定 <Step Value> 的步长，指定为 1 以外的数值。
[StepValue]	可选，Counter 的步长，只能是数值，如果没有指定，则预设为 1。
[Statements]	可选，for 和 next 之间的语句区块，该语句区块将执行所指定的次数。
next	必须的。
[Counter]	可选。

while-wend 语句

While-wend 语句是用来执行不确定次数的循环任务。设置一个变量用来判断结束循环的条件。当条件为“True”时，该语句将一直循环执行直到条件变为 “False”。语法结构如下：

```
while <Condition>
    [Statements]
wend
```

举例:

```
while a < 10
    a = a + 10
wend
```

语法描述

while	必须用在该语句的开始部分。
continue	必要条件。 这是一个控制语句。当为“True”时，开始执行循环命令，当为“False”时，结束执行循环命令。
return [value]	当判断为“TRUE”时，继续执行循环命令
wend	While-wend 语句的结束标志。

其它控制命令

break	用在 for-next 和 while-wend 语句中。当遇到此语句时，立即跳到语句的结束部分。
continue	用在 for-next 和 while-wend 语句中。当遇到此语句时，立即结束当前循环命令而开始执行下一个循环命令。
return	可用在自订 function 的回传值叙述。写在主函数里面时，用来强制跳出主函数。

18.6 子函数

使用子函数可以有效的减少循环命令的代码，子函数必须在使用前被定义，且可以使用任何变量和语句类型。在主函数中，将子函数的参数放置在子函数名称后面的圆括号中，即可调用子函数。子函数被执行后，将执行后的结果返回到主函数需要的赋值语句或者条件中。定义子函数时，不一定要有返回值，且参数部分可以为空。在主函数中调用子函数时，调用方式应符合其定义。语法结构如下：

有返回值的子函数语法：

```
sub type <函數名稱>  [(parameters)]
    Local variable declarations
    [Statements]
    [return  [value]]
end sub
```

举例:

```
sub int Add(int x, int y)
        int result
        result = x +y
```

```
            return result
        end sub

macro_command main()
            int   a = 10, b = 20, sum
            sum = Add(a, b)
        end macro_command
```

或:

```
    sub int Add()
            int result, x=10, y=20
            result = x +y
            return result
        end sub

        macro_command main()
            int   sum
            sum = Add()
        end macro_command
```

没有返回值的子函数语法:

```
sub <函数名称>   [(parameters)]
    Local variable declarations
    [Statements]
end sub
```

举例:

```
    sub Add(int x, int y)
            int result
            result = x +y
        end sub

        macro_command main()
            int   a = 10, b = 20
            Add(a, b)
        end macro_command
```

或:

```
    sub Add()
```

```
        int result, x=10, y=20
        result = x +y
    end sub

    macro_command main()
        Add()
    end macro_command
```

语法描述

sub	必须用在该子函数的开始部分。
type	可选。用来定义子函数执行后返回的数据类型。子函数也可以不回传任何值。
(parameters)	可选。这些参数保留了从主函数传入的数值。这些被传入的参数必须使用与在参数变量声明的类型一致。 举例： sub int MyFunction(int x, int y). x 和 y 必须为从主函数中传过来的双整型数据格式的数据。调用此子函数的语句格式大致为这样： ret = MyFunction(456, pressure)，其中 pressure 需为双整型数据格式方符合子函数参数变量的声明。 请注意调用语句的参数部分可以是常数也可以是变量。当执行这个子函数后，一个双整型数据将会返回给变量“ret”。
Local variable declaration	除了被传递的参数之外，子函数中使用的变量必须事先声明。在上面的“举例”中，X 和 Y 就是子函数可以使用的变量。全局变量也可以用在子函数中。
[Statements]	需要执行的语句。
[return [value]]	可选。用来将执行的结果返回给调用语句。这个结果可以是一个常数或者变量。返回后同时也结束了子函数的执行。子函数也可以不回传任何值，但是当 type 部分有定义时，则必须加上此 return 叙述。
end sub	必须的。用来结束子函数。

18.7 内置函数功能

EasyBuilder Pro 软件宏指令中本身提供了一些内建的函数用来从 PLC 获取数据和传输数据到 PLC、数据处理和数学运算等。

18.7.1 数学运算函数

函数名称	SQRT
语法	SQRT(source, result)
描述	开平方根。数据来源 source 可以是常数或者变量，但是存放结果的 result 必须为变量。数据来源必须为一个正数。

举例	macro_command main() float source, result SQRT(15, result) source = 9.0 SQRT(source, result)// 执行后 result = 3.0 end macro_command

函数名称	CUBERT
语法	CUBERT (source, result)
描述	开三次方根。数据来源 source 可以是常数或者变量，但是存放结果的 result 必须为变量。数据来源必须为一个正数。

举例	macro_command main()
	float source, result
	
	CUBERT (27, result) // 执行后 result = 3.0
	
	source = 27.0
	CUBERT (source, result) // 执行后 result = 3.0
	
	end macro_command

函数名称	POW
语法	POW (source1, source2, result)
描述	计算 source1 的某次方 (source2) 。数据来源 source1 和 source2 可以是常数或者变量，但是存放结果的 result 必须为变量。数据来源必须为一个正数。
举例	macro_command main() float y, result y = 0.5 POW (25, y, result) // 执行后 result = 5 end macro_command

函数名称	SIN
语法	SIN(source, result)
描述	三角函数的正弦计算。数据来源 source 可以是常数或者变量，但是存放结果的 result 必须为变量。
举例	macro_command main() float source, result SIN(90, result) // result is 1 source = 30

	SIN(source, result) // result is 0.5 end macro_command

函数名称	COS
语法	COS(source, result)
描述	三角函数的余弦计算。数据来源 source 可以是常数或者变量，但是存放结果的 result 必须为变量。

举例	macro_command main()
	float source, result
	
	COS(90, result) // result is 0
	
	source = 60
	GetData(source, "Local HMI", LW, 0, 1)
	COS(source, result) // result is 0.5
	
	end macro_command

函数名称	TAN
语法	TAN(source, result)
描述	三角函数的正切计算。数据来源 source 可以是常数或者变量，但是存放结果的 result 必须为变量。

举例	macro_command main()
	float source, result
	
	TAN(45, result) // result is 1
	
	source = 60
	TAN(source, result) // result is 1.732
	
	end macro_command

函数名称	COT
语法	COT(source, result)
描述	三角函数的余切计算。数据来源 source 可以是常数或者变量，但是存放结果的 result 必须为变量。
举例	macro_command main() float source, result COT(45, result) // result is 1 source = 60

	COT(source, result) // result is 0.5774 end macro_command

函数名称	SEC
语法	SEC(source, result)
描述	反三角函数中反正割计算。数据来源 source 可以是常数或者变量，但是存放结果的 result 必须为变量。

举例	macro_command main()
	float source, result
	
	SEC(45, result) // result is 1.414
	
	source = 60
	SEC(source, result) // if source is 60, result is 2
	
	end macro_command

函数名称	CSC
语法	CSC(source, result)
描述	反三角函数中反余割计算。数据来源 source 可以是常数或者变量，但是存放结果的 result 必须为变量。

举例	macro_command main()
	float source, result
	
	CSC(45, result) // result is 1.414
	
	source = 30
	CSC(source, result) // result is 2
	
	end macro_command

函数名称	ASIN
语法	ASIN(source, result)
描述	反三角函数中反正弦计算。数据来源 source 可以是常数或者变量，但是存放结果的 result 必须为变量。

举例	macro_command main()
	float source, result
	
	ASIN(0.8660, result) // result is 60
	
	source = 0.5
	ASIN(source, result) // result is 30
	
	end macro_command

函数名称	ACOS
语法	ACOS(source, result)
描述	反三角函数中反余弦计算。数据来源 source 可以是常数或者变量，但是存放结果的 result 必须为变量。

举例	macro_command main()
	float source, result
	
	ACOS(0.8660, result) // result is 30
	
	source = 0.5
	ACOS(source, result) // result is 60
	
	end macro_command

函数名称	ATAN
语法	ATAN(source, result)
描述	反三角函数中反正切计算。数据来源 source 可以是常数或者变量，但是存放结果的 result 必须为变量。

举例	macro_command main()
	float source, result
	
	ATAN(1, result) // result is 45
	
	source = 1.732
	ATAN(source, result) // result is 60
	
	end macro_command

函数名称	LOG
语法	LOG (source, result)
描述	从取得 source 的自然对数，存入 result 变量。 source 可为变数或常数。 存放结果的 result 必须为变量。
举例	macro_command main() float source=100, result LOG (source, result) // result 约等于 4.6052 end macro_command

函数名称	LOG10
语法	LOG10 (source, result)
描述	从取得 source 的以 10 为基底的对数，存入 result 变数。 Source 可为变数或常数。 存放结果的 result 必须为变量。
举例	macro_command main() float source=100, result LOG10 (source, result) // result 等于 2 end macro_command

函数名称	RAND
语法	RAND(result)
描述	产生一个随机数 存放结果的 result 必须为变量。
举例	macro_command main() short result RAND (result) // result is not a fixed value when executes macro every time end macro_command

18.7.2 数据转换函数

函数名称	BIN2BCD
语法	BIN2BCD(source, result)
描述	将 BIN 格式的数据 (source) 转换为 BCD 格式的数据 (result)。数据来源 source 可以是常数或者变量，但是存放结果的 result 必须为变量。

举例	macro_command main()
	
	short source, result
	
	BIN2BCD(1234, result) // result is 0x1234
	
	source = 5678
	BIN2BCD(source, result) // result is 0x5678
	
	end macro_command

函数名称	BCD2BIN
语法	BCD2BIN(source, result)
描述	将 BCD 格式的数据 (source) 转换为 BIN 格式的数据 (result)。数据来源 source 可以是常数或者变量，但是存放结果的 result 必须为变量。

举例	macro_command main()
	
	short source, result
	
	BCD2BIN(0x1234, result) // result is 1234
	
	source = 0x5678
	BCD2BIN(source, result) // result is 5678
	
	end macro_command

函数名称	DEC2ASCII
语法	DEC2ASCII(source, result[start], len)
描述	将十进制的数据 (source) 转换为 ASCII 格式的数据，并存放在一个一维数组 (result) 中。len 表示这个转换后的字符串的长度，同时这个长度也取决于存放结果的一维数组的数据格式。例如：如果 resul t 一维数组的格式为“char” (字符型，长度为一个字节)，则长度为 “字节数*len” 。如果 result 一维数组的格式为 “short” (短整型数据，2 个字节)，则长度为 “word*len”。依此类推。 转换后的第一个字符放在 result”start” 中，第二个字符放在 result”start+1” 中，最后一个字符放在 result”start+(len-1)”中。 source 和 len 可以是常数或者变量，单数 result 必须为变量。start 必须为常数。

举例	macro_command main()
	short source
	char result1[4]
	short result2[4]
	
	source = 5678
	DEC2ASCII(source, result1[0], 4)
	// result1[0] is '5', result1[1] is '6', result1[2] is '7', result1[3] is '8'
	// the length of the string (result1) is 4 bytes(= 1 * 4)
	
	DEC2ASCII(source, result2[0], 4)
	// result2[0] is '5', result2[1] is '6', result2[2] is '7', result2[3] is '8'
	// the length of the string (result2) is 8 bytes(= 2 * 4)
	
	end macro_command

函数名称	HEX2ASCII
语法	HEX2ASCII(source, result”start”, len)
描述	十六进制格式数据 (source) 转换为 ASCII 格式的数据，并将结果存放在一个一维数组 (result) 中。len 表示这个转换后的字符串的长度，同时这个长度也取决于存放结果的一维数组的数据格式。例如：如果 result 一维数组的格式为“char” (字符型，长度为一个字节)，则长度为 “字节数*len” 。如果 result 一维数组的格式为 “short” (短整型数据，2 个字节)，则长度为 “word*len” 。依此类推。 source 和 len 可以是常数或者变量，单数 result 必须为变量。start 必须为常数。
举例	macro_command main() short source char result[4] source = 0x5678 HEX2ASCII(source, result[0], 4) // result[0] is '5', result[1] is '6', result[2] is '7', result[3] is '8' end macro_command

函数名称	FLOAT2ASCII
语法	FLOAT2ASCII (source, result[start], len)
描述	浮点数格式数据 (source) 转换为 ASCII 格式的数据，并将结果存放在一个一维数组 (result) 中。len 表示这个转换后的字符串的长度，同时这个长度也取决于存放结果的一维数组的数据格式。例如：如果 result 一维数组的格式为 “char” (字符型，长度为一个字节)，则长度为 “字节数*len” 。如果 result 一维数组的格式为 “short” (短整型数据，2 个字节)，则长度为 “word*len” 。依此类推。 source 和 len 可以是常数或者变量，单数 result 必须为变量。start 必须为常数。
举例	macro_command main() float source char result[4] source = 56.8 FLOAT2ASCII (source, result[0], 4) // result[0] is '5', result[1] is '6', result[2] is '.', result[3] is '8' end macro_command

函数名称	ASCII2DEC
语法	ASCII2DEC(source[start], result, len)
描述	将字符型 ASCII 数据 (source) 转换为十进制格式的数据，并存放在 result 变数中。ASCII 的长度即为 len，第一个字符的位置即为 source[start] 的数据。 source 和 len 可以是常数或者变量，单数 result 必须为变量。start 必须为常数。
举例	macro_command main() char source[4] short result source[0] = '5' source[1] = '6' source[2] = '7' source[3] = '8' ASCII2DEC(source[0], result, 4) // result is 5678 end macro_command

函数名称	ASCII2HEX
语法	ASCII2HEX (source[start], result, len)
描述	将 ASCII 字符型数据 (source) 转换为十六进制的数据，并存放在 result 变数中。字符的长度即为 len 的数据。第一个字符存放在 source[start] 中。 source 和 len 可以是常数或者变量，单数 result 必须为变量。start 必须为常数。
举例	macro_command main() char source[4] short result source[0] = '5' source[1] = '6' source[2] = '7' source[3] = '8' ASCII2HEX(source[0], result, 4) // result is 0x5678 end macro_command

函数名称	ASCII2FLOAT
语法	ASCII2FLOAT (source[start], result, len)
描述	将字符型 ASCII 数据 (source) 转换为浮点数格式的数据，并存放在 result 变数中。ASCII 的长度即为 len，第一个字符的位置为 source[start] 的数据。 source 和 len 可以是常数或者变量，单数 result 必须为变量。Start 必须为常数。
举例	macro_command main() char source[4] float result source[0] = '5' source[1] = '6' source[2] = '.' source[3] = '8' ASCII2FLOAT(source[0], result, 4) // result is 56.8 end macro_command

18.7.3 数据操作函数

函数名称	FILL
语法	FILL(source[start], preset, count)
描述	依序将默认值 (preset) 放置到一维数组 source[start] 开始的数组中，放置的数据个数由 count 决定。 source 和 start 必须为变量，preset 可以为一个常数或者变量。

举例	macro_command main()
	char result[4]
	char preset
	
	FILL(result[0], 0x30, 4)
	// result[0] is 0x30, result[1] is 0x30, , result[2] is 0x30, , result[3] is 0x30
	
	preset = 0x31
	FILL(result[0], preset, 2) // result[0] is 0x31, result[1] is 0x31
	
	end macro_command

函数名称	SWAPB
语法	SWAPB(source, result)
描述	将一个 16 位字的高低字节颠倒，并将结果存放在 result 变量中。 source 可以是常数或者是变量，单数 result 必须为变量。

举例	macro_command main()
	short source, result
	
	SWAPB(0x5678, result) // result is 0x7856
	
	source = 0x123
	SWAPB(source, result) // result is 0x2301
	
	end macro_command

函数名称	SWAPW
语法	SWAPW(source, result)
描述	将一个 32 位双整型数据的高位字符和低位字符颠倒，并将结果存放在 result 变量中。source 可以是常数或者变量，但是 result 必须为变量。

举例	macro_command main()
	int source, result
	
	SWAPW(0x12345678, result) // result is 0x56781234
	
	source = 0x12345
	SWAPW(source, result) // result is 0x23450001

	end macro_command

函数名称	LOBYTE
语法	LOBYTE(source, result)
描述	获取一个 16 位数据的低字节，并且放置在 result 变量中。 source 可以是常数或者变量，但是 result 必须为变量。

举例	macro_command main()
	short source, result
	
	LOBYTE(0x1234, result) // result is 0x34
	
	source = 0x123
	LOBYTE(source, result) // result is 0x23
	
	end macro_command

函数名称	HIBYTE
语法	HIBYTE(source, result)
描述	获取一个 16 位数据的高字节，并且放置在 result 变量中。 source 可以是常数或者变量，但是 result 必须为变量。

举例	macro_command main()
	short source, result
	
	HIBYTE(0x1234, result) // result is 0x12
	
	source = 0x123
	HIBYTE(source, result) // result is 0x01
	
	end macro_command

函数名称	LOWORD
语法	LOWORD(source, result)
描述	获取一个 32 位数据的低位字符，并将结果放置在 result 变量中。 source 可以是常数或者变量，但是 result 必须为变量。

举例	macro_command main()
	int source, result
	
	LOWORD(0x12345678, result) // result is 0x5678
	
	source = 0x12345
	LOWORD(source, result) // result is 0x2345
	
	end macro_command

函数名称	HIWORD
语法	HIWORD(source, result)
描述	获取一个 32 位数据的高位字符，并将结果放置在 result 变量中。 source 可以是常数或者变量，但是 result 必须为变量。

举例	macro_command main()
	int source, result
	
	HIWORD(0x12345678, result) // result is 0x1234
	
	source = 0x12345
	HIWORD(source, result) // result is 0x0001
	
	end macro_command

18.7.4 位状态转换

函数名称	GETBIT
语法	GETBIT(source, result, bit_pos)
描述	获取数据或者变量 (source) 指定的位的状态，并将结果放置在 result 变量中。 result 的数据将为 1 或者 0。 source 和 bit_pos 可以是常数或者变量，但是 result 必须为变量。

举例	macro_command main()
	int source, result
	short bit_pos
	
	GETBIT(9, result, 3) // result is 1
	
	source = 4
	bit_pos = 2
	GETBIT(source, result, bit_pos) // result is 1
	
	end macro_command

函数名称	SETBITON
语法	SETBITON(source, result, bit_pos)
描述	将数据或者变量 (source) 指定的位地址设置为 1，并将改变后的数据存放在 result 变量中。 source 和 bit-pos 可以是常数或者变量，但是 result 必须为变量。

举例	macro_command main()
	int source, result
	short bit_pos
	
	SETBITON(1, result, 3) // result is 9
	
	source = 0
	bit_pos = 2
	SETBITON (source, result, bit_pos) // result is 4
	
	end macro_command

函数名称	SETBITOFF
语法	SETBITOFF(source, result, bit_pos)
描述	将数据或者变量 (source) 指定的位地址设置为 0，并将改变后的数据存放在 result 变量中。 source 和 bit-pos 可以是常数或者变量，但是 result 必须为变量。

举例	macro_command main()
	int source, result
	short bit_pos
	
	SETBITOFF(9, result, 3) // result is 1
	
	source = 4
	bit_pos = 2
	SETBITOFF(source, result, bit_pos) // result is 0
	
	end macro_command

函数名称	INVBIT
语法	INVBIT(source, result, bit_pos)
描述	将数据或者变量 (source) 指定的位地址状态相反，并将改变后的数据存放在 result 变量中。 source 和 bit-pos 可以是常数或者变量，但是 result 必须为变量。

举例	macro_command main()
	int source, result
	short bit_pos
	
	INVBIT(4, result, 1) // result = 6
	
	source = 6
	bit_pos = 1
	INVBIT(source, result, bit_pos) // result = 4
	
	end macro_command

18.7.5 通讯有关的函数

函数名称	DELAY
语法	DELAY(time)
描述	让宏指令暂停执行，持续的时间至少是指定的这个时间。时间的单位为毫秒。 time 可以是常数或者变量。

举例	macro_command main()
	int time = 500
	
	DELAY(100) // delay 100 ms
	DELAY(time) // delay 500 ms
	
	end macro_command

函数名称	ADDSUM
语法	ADDSUM(source[start], result, data_count)
描述	将 source[start] 到 source[start+data-count-1] 的所有一维数组的数据累加起来，以获得 checksum (校验和)，并将结果存放在 result 变量中。 result 必须为变量，data_count 是进行累加的资料的个数，可以是常数或者是变量。
举例	macro_command main() char data[5] short checksum data[0] = 0x1 data[1] = 0x2 data[2] = 0x3 data[3] = 0x4 data[4] = 0x5 ADDSUM(data[0], checksum, 5) // checksum is 0xf end macro_command

函数名称	XORSUM
语法	XORSUM(source[start], result, data_count)
描述	将 source[start] 到 source[start+data-count-1] 的所有一维数组的数据进行异或运算，以获得 checksum (校验和)，并将结果存放在 result 变量中。 result 必须为变量，data_count 是进行异或计算的数据的个数，可以是常数或者是变量。
举例	macro_command main() char data[5] = {0x1, 0x2, 0x3, 0x4, 0x5} short checksum XORSUM(data[0], checksum, 5) // checksum is 0x1 end macro_command

函数名称	CRC
语法	CRC(source[start], result, data_count)
描述	将 source[start] 到 source[start+data-count-1] 的所有一维数组的数据进 16-bit CRC 计算，以获得 checksum (校验和)，并将结果存放在 result 变量中。 result 必须为变量，data_count 是进行计算的资料的个数，可以是常数或者是变量。
举例	macro_command main() char data[] = {0x1, 0x2, 0x3, 0x4, 0x5} short 16bit_CRC CRC(data[0], 16bit_CRC, 5) // 16bit_CRC is 0xbb2a end macro_command

函数名称	OUTPORT
语法	OUTPORT(source[start], device_函数名称, data_count)
描述	将放置在从 source[start] 到 source[start+count-1] 的所有数据通过串行端口或者以太网口传送给 PLC 或者控制器中。 device_函数名称是在 “设备列表” 中定义的 “PLC 名称”，而这个 device 必须选择为 “Free Protocol” 这个 PLC 类型。 Data_count 是发送数据的个数，可以是常数或变量。

举例	要使用 OUTPORT 函数，必须要在 PLC 类型中选择 “Free Protocol”，如下图所示。
	 
	
	这里的 device_函数名称即为 “MODBUD RTU Device”。端口的属性也是依据在这个 “系统参数” 中的设定，譬如，在此设定为 (9600，E，8，1…)
	
	下面是一个范例程序，使用自由协议，以 MODBUS RTU 的协议格式，将单个寄存器设置为 ON。
	
	macro_command main()
	
	char command[32]
	short address, checksum
	
	FILL(command[0], 0, 32) // 初始化命令
	
	Command[0] = 0x1 // 站号
	Command[1] = 0x5 // 功能码 : 写单个位
	
	address = 0
	HIBYTE(address, command[2])
	LOBYTE(address, command[3])
	
	Command[4] = 0xff // 使该bit设置为ON
	Command[5] = 0
	
	CRC(command[0], checksum, 6)

	LOBYTE(checksum, command[6])
	HIBYTE(checksum, command[7])
	
	// 将命令通过串行埠送出去
	OUTPORT(command[0], "MODBUS RTU Device", 8)
	
	end macro_command

函数名称	INPORT
语法	INPORT(read_data[start], device_函数名称, read_count, return_value)
描述	从串行端口或者以太网口读数据到触摸屏界面上。这些资料保存在 read-data[start]~read-data[start+read-count-1] 这个一维数组中。同样的，device_函数名称见上面的说明，在此不再详述。 read-count 是设定的需要读取的命令的位组长度，它可以是一个常数或变量。如果这个函数能够成功的从 PLC 或者控制器中读取到数据，则return_value 的值为 1，否则就为 0。

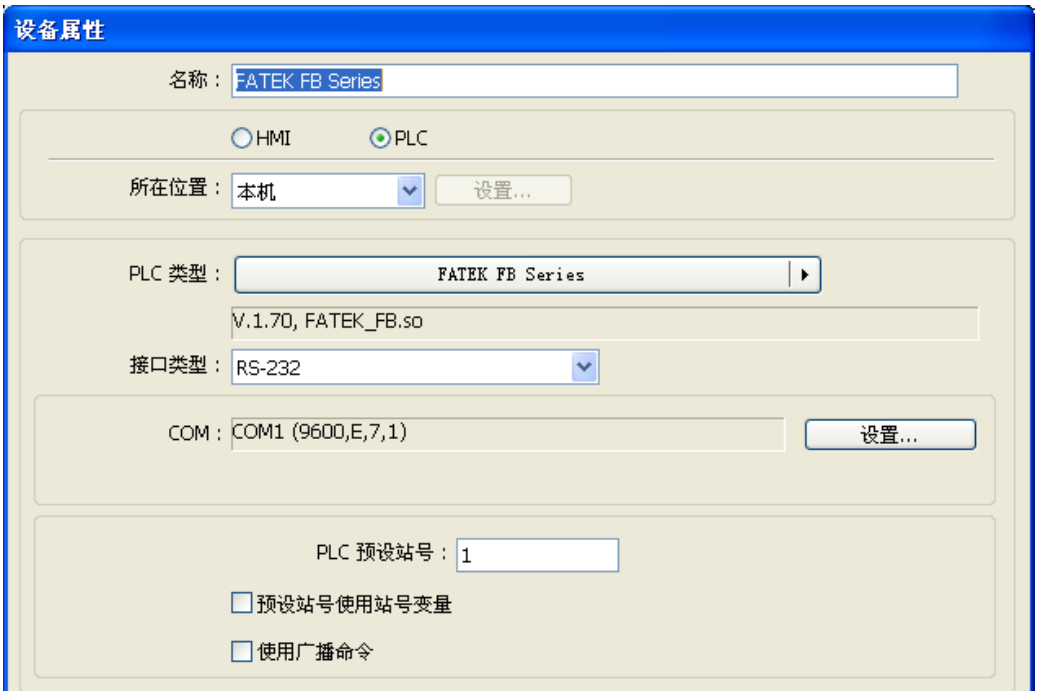
举例	下面就是一个使用 INPORT 函数读取一个 MODBUS 设备保持寄存器数据的范例。
	
	// 读取保持寄存器数据
	macro_command main()
	char command[32], response[32]
	short address, checksum
	short read_no, return_value, read_data[2]
	
	FILL(command[0], 0, 32) // 命令初始化
	FILL(response[0], 0, 32)
	
	Command[0] = 0x1 // 站号
	Command[1] = 0x3 // 功能码：读取保持寄存器
	
	address = 0
	HIBYTE(address, command[2])
	LOBYTE(address, command[3])
	
	read_no = 2 // read 2 words (4x_1 and 4x_2)
	HIBYTE(read_no, command[4])
	LOBYTE(read_no, command[5])
	
	CRC(command[0], checksum, 6)
	
	LOBYTE(checksum, command[6])
	HIBYTE(checksum, command[7])
	
	// 使用 OUTPORT 函数将命令送出去
	OUTPORT(command[0], "MODBUS RTU Device", 8)
	
	// 使用 INPORT 函数读取返回的命令

	INPORT(response[0], "MODBUS RTU Device", 9, return_value)
	
	if return_value > 0 then
	  read_data[0] = response[4] + (response[3] << 8) // data in 4x_1
	  read_data[1] = response[6] + (response[5] << 8) // data in 4x_2
	
	  SetData(read_data[0], "Local HMI", LW, 100, 2)
	end if
	
	end macro_command

函数名称	INPORT2
语法	INPORT2(response[start], device_name, receive_len, wait_time)
描述	从串行端口或者以太网口读数据到触摸屏界面上。这些数据保存在 response 这个一维数组中。device_name 的说明与 OUTPORT 相同，在此不再详述。 receive_len 存放所接收到的数据位组长度，必须为变量。receive_len 的最大长度将不会超过 response 数组宣告的大小。wait_time 表示等待时间（单位是 millisecond），可以是一个常数或变量。当数据读取完毕后，若在指定的等待时间内未再收到任何数据，此函数便结束执行并回传结果。

举例	macro_command main()
	
	short wResponse[6], receive_len, wait_time=20
	
	INPORT2(wResponse[0], "Free Protocol", receive_len, wait_time)
	// wait_time 单位 ：millisecond
	
	if receive_len > 0 then
	  SetData(wResponse[0], "Local HMI", LW, 0, 6)
	  // 把响应的数据写入LW0
	end if
	
	end macro_command

函数名称	GetData
语法	GetData(read_data[start], device_函数名称, device_type, address_offset, data_count) or GetData(read_data, device_函数名称, device_type, address_offset, 1)
描述	获取 PLC 的资料。数据是存储在 read_data[start]~read_data[start+data_count-1] 这些一维数组变量中。 data_count 是设定的读取数据的个数。一般来说，read_data 是一个一维数组，但是如果 data_count 是 1，read_data 可以是一个一维数组，也可以是一个普通的变量。下面是两种从 PLC 中读取一个字的方法。 macro_command main() short read_data_1[2], read_data_2 GetData(read_data_1[0], “FATEK FB Series”, RT, 5, 1) GetData(read_data_2, “FATEK FB Series”, RT, 5, 1) end macro_command 此处的 device_函数名称，即为在 “系统参数” 中建立 PLC 类型时，设定的“PLC 名称”。在此，PLC 名称被设定为 “FATEK FB Series”，如下图所示。  device_type 是设备类型和 PLC 中数据的编码方式。例如：如果 device_type 是 LW_BIN，那么读取的设备类型为 LW，数据编码方式为 BIN。如果使用 BIN 编码方式，“_BIN” 可以忽略。 如果 device_type 是 LW_BCD，表示设备类型 LW，数据的编码方式为 BCD 格式。 address_offset 是 PLC 中的地址偏移量。 例如，GetData(read_data_1[0], “FATEK FB Series”, RT, 5, 1) 代表读取的设备地址偏移量为 5。 如果 address_offset 使用格式为 “N#AAAAA”，N 表示 PLC 的站号，AAAAA 表示地址偏移量。此情况一般使用在同一个串行埠上连接有多台 PLC 或者控制器的情况下。例如：GetData(read_data_1[0], “FATEK FB Series”, RT, 2#5, 1) 表示读取站号为 2 的 PLC 的数据。如果 GetData() 使用 “系统参数 / 设备列表” 中设定的默认的站号，在此可以不填这个站号。



从 PLC 中读取的资料个数，根据 read_data 变量的类型和 data_count 的值来决定的。如下表所示：

read_data的类型	data_count的值	读取16位数据的个数
char (8-bit)	1	1
char (8-bit)	2	1
boo□(8-b□t)	1	1
b□ol (8-bit)	2	1
short (16-bit)	1	1
short (16-bit)	2	2
int (32-bit)	1	2
int (32-bit)	2	4
float (32-bit)	1	2
float (32-bit)	2	4

当 Getdata() 函数读取 32 位的数据类型 (int 或者float型) 时，此函数会自动的转换这个数据。例如：

```
macro_command main()
    float f
    GetData(f, "MODBUS", 6x, 2, 1)  // f 中将会是浮点型的数据

end macro_command
```

举例	macro_command main()
	bool a
	bool b[30]
	short c
	short d[50]
	int e
	int f[10]
	double g[10]
	
	// 读取 LB2 的状态到变量 a 中
	GetData(a, “Local HMI”, LB, 2, 1)
	
	// 读取 LB0~LB29共 30 个状态，到变量 b[0] ~ b[29] 中
	GetData(b[0], “Local HMI”, LB, 0, 30)
	
	// 读取 LW-2 的数据到变量 c 中
	GetData(c, “Local HMI”, LW, 2, 1)
	
	// 读取 LW-0 ~ LW-49 共 50 个字到变数 d[0] 到 d[49] 中
	GetData(d[0], “Local HMI”, LW, 0, 50)
	
	// 读取两个字 LW-6 ~ LW-7 到变数 e 中
	// 注意此时变量 e 的类型为 int
	GetData(e, “Local HMI”, LW, 6, 1)
	
	// 读取 LW-0 ~ LW-19 共 20 个字到变数 f[0] ~ f[9] 中 (共 10 个 int 型变量)，数组 f[10] 的变量类型定义为 int。
	// 注意一个 int 资料将占据 2 个字符
	GetData(f[0], “Local HMI”, LW, 0, 10)
	
	// 读取 LW-2 ~ LW-3 共 2 个字到变数 f 中
	GetData(f, “Local HMI”, LW, 2, 1)
	
	end macro_command

函数名称	GetDataEx
语法	GetDataEx (read_data[start], device_函数名称, device_type, address_offset, data_count) or GetDataEx (read_data, device_函数名称, device_type, address_offset, 1)
描述	获取 PLC 的数据，不等待 PLC 响应，径自往下执行。 read_data、device_函数名称、device_type、address_offset 和 data_count的说明和 GetData 相同。
举例	macro_command main() bool a bool b[30] short c

```
short d[50]
int e
int f[10]
double g[10]

// 读取 LB-2 的状态到变量 a 中
GetDataEx (a, “Local HMI”, LB, 2, 1)

// 读取 LB-0 ~ LB-29 共 30 个状态，到变量 b[0] ~ b[29] 中
GetDataEx (b[0], “Local HMI”, LB, 0, 30)

// 读取 LW-2 的数据到变量 c 中
GetDataEx (c, “Local HMI”, LW, 2, 1)

// 读取 LW-0 ~ LW-49 共 50 个字到变数 d[0] 到 d[49 ]中
GetDataEx (d[0], “Local HMI”, LW, 0, 50)

// 读取两个字 LW-6 ~ LW-7 到变数 e 中
// 注意此时变量 e 的类型为 int
GetDataEx (e, “Local HMI”, LW, 6, 1)

// 读取 LW-0 ~ LW-19 共 20 个字到变数 f[0] ~ f[9] 中，数组 f[10] 的变量类型定义为 int。
// 注意一个 int 资料将占据 2 个字符
GetDataEx (f[0], “Local HMI”, LW, 0, 10)

// 读取 LW-2 ~ LW-3 共 2 个字到变数 f 中
GetDataEx (f, “Local HMI”, LW, 2, 1)

end macro_command
```

函数名称	SetData
语法	SetData(send_data[start], device_函数名称, device_type, address_offset, data_count) or SetData(send_data, device_函数名称, device_type, address_offset, 1)
描述	将数据写到 PLC 中。资料保存在 send_data[start]~send_data[start+data_count-1] 中。 data_count 是写入到 PLC 中资料的个数。一般来说，send_data 是一个数组。但是如果 data_count 是 1，send_data 可以是一个数组也可以是一个普通的变量。下面是写一个数据到 PLC 中的方法。 macro_command main() short send_data_1[2] = { 5, 6}, send_data_2 = 5 SetData(send_data_1[0], “FATEK FB Series”, RT, 5, 1) SetData(send_data_2, “FATEK FB Series”, RT, 5, 1)

```
end macro_command
```

device_函数名称详见上面的说明，在此不在说明。
device_type 是设备类型和 PLC 中数据的编码方式。例如：如果 device_type 是 LW_BIN，那么读取的设备类型为 LW，数据编码方式为 BIN。如果使用 BIN 编码方式，“_BIN”可以忽略。
如果 device_type 是 LW_BCD，表示设备类型 LW，数据的编码方式为 BCD 格式。
address_offset 是 PLC 中的地址偏移量。
例如，SetData (read_data_1[0], “FATEK FB Series”, RT, 5, 1）代表读取的设备地址偏移量为 5。
如果 address_offset 使用格式为 “N#AAAAA”，N 表示 PLC 的站号，AAAAA 表示地址偏移量。此情况一般使用在同一个串行埠上连接有多台 PLC 或者控制器的情况下。例如：SetData(send_data_1[0], “FATEK FB Series”, RT, 2#5, 1）表示设定站号为 2 的 PLC 的数据。如果 SetData()使用“系统参数 / 设备列表”中设定的默认的站号，在此可以不填这个站号。

设定到 PLC 的数据个数，根据 sead_data 变量的类型和 data_count 的值来决定的。如下表所示：

sead_data 的类型	data_count 的值	设定 16 位数据的个数
char (8-bit)	1	1
char (8-bit)	2	1
bool (8-bit)	1	1
bool (8-bit)	2	1
short (16-bit)	1	1
short (16-bit)	2	2
int (32-bit)	1	2
int (32-bit)	2	4
float (32-bit)	1	2
float (32-bit)	2	4

当 Setdata() 函数写入 32 位的数据类型 (int 或者 float 型) 到 PLC 时，此函数会自动的转换这个数据。例如：

```
macro_command main()
float f = 2.6
SetData(f, "MODBUS", 6x, 2, 1) // 在此将会设定一个浮点数到 PLC 中
end macro_command
```

举例

```
macro_command main()
int i
bool a = true
bool b[30]
short c = false
short d[50]
int e = 5
```

```
int f[10]

for i = 0 to 29
b[i] = true
next i

for i = 0 to 49
d[i] = i * 2
next i

for i = 0 to 9
f [i] = i * 3
next i

// 变量 a 的数值设定到 LB2 中
SetData(a, “Local HMI”, LB, 2, 1)


// 设定 LB0 ~ LB29 共 30 个位的状态
SetData(b[0], “Local HMI”, LB, 0, 30)

// 将变量 c 的值设定到 LW-2 中
SetData(c, “Local HMI”, LW, 2, 1)

// 设定 LW-0 ~ LW-49 共 50 个数据
SetData(d[0], “Local HMI”, LW, 0, 50)

// 将变量 e 的值写入到 LW-6 ~ LW-7 两个寄存器中，注意变量 e 的类型为int。
SetData(e, “Local HMI”, LW, 6, 1)

// 设定 LW-0 ~ LW-19 共 20 个字的数据
// 10 个双整型数据等于 20 个 16 位整型数 (1个字符)，因一个 int 数据将占据 2 个字符
SetData(f[0], “Local HMI”, LW, 0, 10)

end macro_command
```

函数名称	SetDataEx
语法	SetDataEx (send_data[start], device_函数名称, device_type, address_offset, data_count) or SetDataEx (send_data, device_函数名称, device_type, address_offset, 1)
描述	将数据写到 PLC 中，不等待 PLC 回应，径自往下执行。 send_data、device_函数名称、device_type、address_offset和data_count的说明和 SetData 相同。
举例	macro_command main() int i

```
bool a = true
bool b[30]
short c = false
short d[50]
int e = 5
int f[10]

for i = 0 to 29
b[i] = true
next i

for i = 0 to 49
d[i] = i * 2
next i

for i = 0 to 9
f [i] = i * 3
next i

// 将变量 a 的数值设定到 LB2 中
SetDataEx(a, “Local HMI”, LB, 2, 1)


// 设定 LB0 ~ LB29 共 30 个位的状态
SetDataEx (b[0], “Local HMI”, LB, 0, 30)

// 将变量 c 的值设定到 LW-2 中
SetDataEx (c, “Local HMI”, LW, 2, 1)

// 设定 LW-0 ~ LW-49 共 50 个数据
SetDataEx (d[0], “Local HMI”, LW, 0, 50)

// 将变量 e 的值写入到 LW-6 ~ LW-7 两个寄存器中，注意变量 e 的类型为int。
SetDataEx (e, “Local HMI”, LW, 6, 1)

// 设定 LW-0 ~ LW-19 共 20 个字的数据
// 10 个双整型数据等于 20 个 16 位整型数 (1个字符)，因一个 int 数据将占据 2 个字符
SetDataEx (f[0], “Local HMI”, LW, 0, 10)

end macro_command
```

函数名称	GetError
语法	GetError(err)
描述	取得错误码。

举例	macro_command main()
	short err
	char byData[10]
	
	GetDataEx(byData[0], “MODBUS RTU”, 4x, 1, 10) // 读取 10 byte数据
	
	// 当错误码 (err) 为 0，表示 GetDataEx 成功执行
	GetErr(err) // 读取错误码，存入变量err
	
	end macro_command

函数名称	PURGE
语法	PURGE (com_port)
描述	com_port 表示串行端口编号，支持 COM1 ~ COM3。此参数可以为变量或常数。 可利用这个指令来清空 COM port 的输出入缓冲区。
举例	macro_command main() int com_port=3 PURGE (com_port) PURGE (1) end macro_command

函数名称	SetRTS
语法	SetRTS(com_port, source)
描述	拉高或拉低 RS-232 之 RTS 讯号。 com_port 表示串行端口编号，支持 COM1。此参数可以为变量或常数。source参数也可以为变量或常数。 当传入的 source 参数值大于 0 时，将拉高 RTS 电位，当 source 参数值等于 0 时，将拉低 RTS 电位。

举例	macro_command main()
	char com_port=1
	char value=1
	
	SetRTS(com_port, value) // value > 0，拉高 COM1 之 RTS 电位
	
	SetRTS(1, 0) // 拉低 COM1 之 RTS 电位
	
	end macro_command

函数名称	GetCTS
语法	GetCTS(com_port, result)
描述	侦测 RS-232 之 CTS 讯号。 com_port 表示串行端口编号，支持 COM1。此参数可以为变量或常数。result参数必须为变量。 此函数可侦测 CTS 电位值，当 CTS 在高电位时，result 将写入 1，否则写入 0。

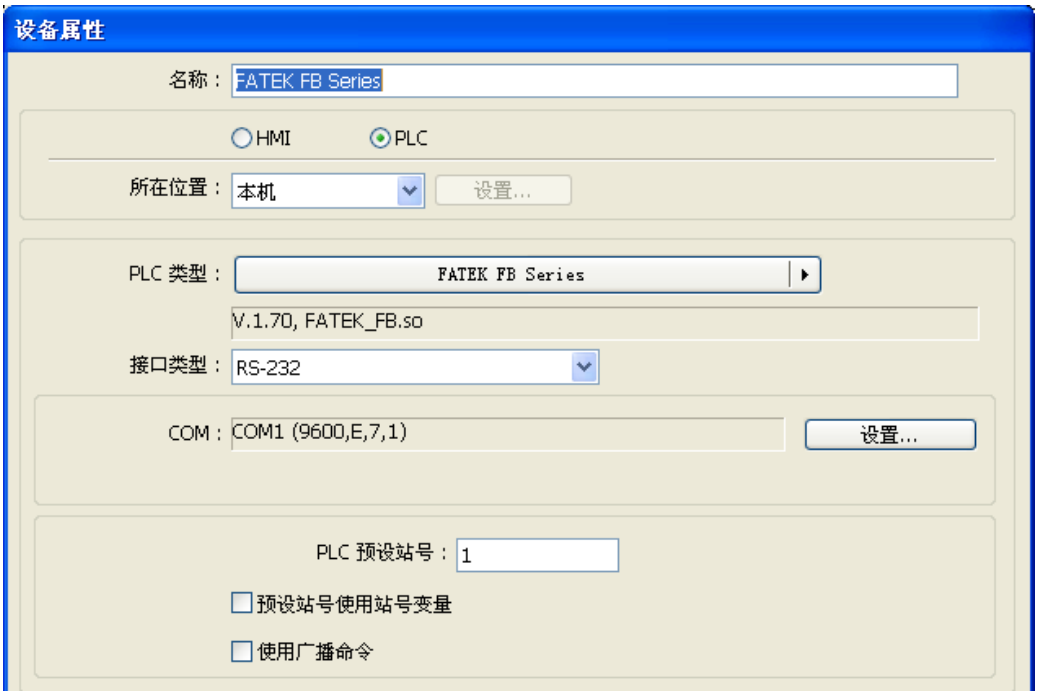
举例	macro_command main()
	char com_port=1
	char result
	
	GetCTS(com_port, result) // 侦测 COM1 之 CTS 电位值
	
	GetCTS (1, result) // 侦测 COM1 之 CTS 电位值
	
	end macro_command

18.7.6 字符串处理函数

函数名称	StringGet
语法	StringGet(read_data[start], device_函数名称, device_type, address_offset, data_count)
描述	获取 PLC 的资料。字符串的数据类型为字符数组，是存储在 read_data[start]~read_data[start+data_count-1] 这些一维数组变量中。read_data 必须为一维字符数组。 data_count 是设定的读取字符的个数，可以是常数也可以是变量。 此处的 device_函数名称，即为在“系统参数”中建立 PLC 类型时，设定的“PLC 名称”。在此，PLC 名称被设定为 “FATEK FB Series”，如下图所示。  device_type 是设备类型和 PLC 中数据的编码方式。例如：如果 device_type 是 LW_BIN，那么读取的设备类型为 LW，数据编码方式为 BIN。如果使用 BIN 编码方式，“_BIN”可以忽略。 如果 device_type 是 LW_BCD，表示设备类型 LW，数据的编码方式为 BCD 格式。address_offset 是 PLC 中的地址偏移量。 例如，StringGet(read_data_1[0], “FATEK FB Series”, RT, 5, 1）代表读取的设备地

址偏移量为 5。

如果 address_offset 使用格式为 “N#AAAAA”，N 表示 PLC 的站号，AAAAA 表示地址偏移量。此情况一般使用在同一个串行埠上连接有多台 PLC 或者控制器的情况下。例如：StringGet(read_data_1”0”, “FATEK FB Series”, RT, 2#5, 1）表示读取站号为 2 的 PLC 的数据。如果 StringGet()使用 “系统参数 / 设备列表” 中设定的默认的站号，在此可以不填这个站号。



从 PLC 中读取的数据个数，由 data_count 的值来决定，因 read_data 变量仅接受 char 数组类型。如下表所示：

read_data 的类型	data_count 的值	读取 16 位数据的个数
char (8-bit)	1	1
char (8-bit)	2	1

因为一个 WORD (16 位) 等于 2 个 ASCII 字符的长度，当设备类型长度为 WORD 时，根据上表，读取 2 个 ASCII 字符实际上是读 1 个 WORD 的数据。

举例

```
macro_command main()
char str1[20]

// 读取 LW-0 ~ LW-9 共 10 个 WORD 到变数 str1[0] 到 str1[19] 中
// 因 1 WORD 可存放 2 个 ASCII 字符，欲读取 20 个 ASCII 字符
// 实际上共读取了 10 个WORD
StringGet(str1[0], “Local HMI”, LW, 0, 20)

end macro_command
```

函数名称	StringGetEx
语法	StringGetEx (read_data[start], device_函数名称, device_type, address_offset, data_count)
描述	获取 PLC 的数据，不等待 PLC 响应，径自往下执行。 read_data、device_函数名称、device_type、address_offset 和 data_count 的说明和 GetData 相同。

举例	macro_command main()
	char str1[20]
	short test=0
	
	// 当 MODBUS 设备未回应，test = 1将照常执行
	StringGetEx(str1[0], "MODBUS RTU", 4x, 0, 20)
	test = 1
	
	// 当 MODBUS 设备未响应，test = 2 将不会被执行，直到得到响应
	StringGet(str1[0], "MODBUS RTU", 4x, 0, 20)
	test = 2
	
	end macro_command

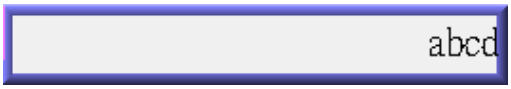
函数名称	StringSet
语法	StringSet (send_data[start], device_函数名称, device_type, address_offset, data_count)
描述	将数据写到 PLC 中。字符串数据保存在 send_data[start]~send_data[start+data_count-1] 中，send_data 必须为一维字符数组类型。 data_count 是写入到 PLC 中字符数据的个数，可以是常数也可以是变量。 device_函数名称详见上面的说明，在此不再说明。 device_type 是设备类型和 PLC 中数据的编码方式。例如：如果 device_type 是 LW_BIN，那么读取的设备类型为 LW，数据编码方式为 BIN。如果使用 BIN 编码方式，“_BIN”可以忽略。 如果 device_type 是 LW_BCD，表示设备类型 LW，数据的编码方式为 BCD 格式。 address_offset 是 PLC 中的地址偏移量。 例如，StringSet(read_data_1[0], “FATEK FB Series”, RT, 5, 1) 代表读取的设备地址偏移量为 5。 如果 address_offset 使用格式为 “N#AAAAA”，N 表示 PLC 的站号，AAAAA 表示地址偏移量。此情况一般使用在同一个串行埠上连接有多台 PLC 或者控制器的情况下。例如：StringSet(send_data_1[0], “FATEK FB Series”, RT, 2#5, 1) 表示设定站号为 2 的 PLC 的数据。如果 StringSet() 使用“系统参数 / 设备列表”中设定的默认的站号，在此可以不填这个站号。 设定到 PLC 的资料个数，根据 data_count 的值来决定，因 send_data 变量仅接受 char 数组类型。如下表所示：

sead_data 的类型	data_count 的值	设定 16 位数据的个数
char (8-bit)	1	1
char (8-bit)	2	1

因为一个 WORD (16 位) 等于 2 个 ASCII 字符的长度，当设备类型长度为 WORD 时，根据上表，写入 2 个 ASCII 字符实际上是写 1 个 WORD 的数据。宏指令会以先写 low byte 再写 hight byte 的顺序，依序将 ASCII 字符写入。
使用“字符显示”元件显示数据时，data_count 必须填入 2 的倍数才能正确显示。例如：

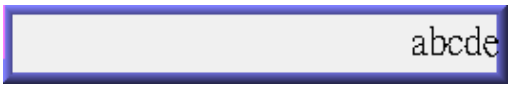
```
macro_command main()
char src1[10]="abcde"
StringSet(src1[0], "Local HMI", LW, 0, 5)
end macro_command
```

“字符显示”元件显示如下：



当 data_count 为一个大于或等于字符串长度的偶数时，可以完整显示整个字符串：

```
macro_command main()
char src1[10]="abcde"
StringSet(src1[0], "Local HMI", LW, 0, 6)
end macro_command
```



举例

```
macro_command main()
char str1[10]= “abcde”

// 字符串 str1 写入 LW-0 ~ LW-2 共三个 WORD
// 即使 data_count 为 10，写到第 3 个 WORD 时发现字符串已结束，
// 便不再写入后面的数据
StringSet(str1[0], "Local HMI", LW, 0, 10)

end macro_command
```

函数名称	StringSetEx
语法	StringSetEx(send_data[start], device_函数名称, device_type, address_offset,

	data_count)
描述	将数据写到 PLC 中，不等待 PLC 回应，径自往下执行。 send_data、device_函数名称、device_type、address_offset 和 data_count的说明和 StringSet 相同。

举例	macro_command main()
	char str1[20]= “abcde”
	short test=0
	
	// 当 MODBUS 设备未回应，test = 1 将照常执行
	StringSetEx(str1[0], "MODBUS RTU", 4x, 0, 20)
	test = 1
	
	// 当 MODBUS 设备未响应，test = 2 将不会被执行，直到得到响应
	StringSet(str1[0], "MODBUS RTU", 4x, 0, 20)
	test = 2
	
	end macro_command

函数名称	StringCopy
语法	success = StringCopy ([source], destination[start]) 或 success = StringCopy (source[start], destination[start])
描述	利用此函数进行字符串的复制。此函数可以将传入的静态字符串 (以双引号”” 括起来的字符串)，或是存放在字符数组中的字符串复制到目的 buffer。 来源字符串 sourc e可以为静态字符串 (如 “source”)或是一维字符数组变量 (如source[start])。 Destination[start] 必须为一维字符数组变量。 复制完毕会回传一 bool 类型的值给 success 字段。当复制成功，success等于 true，否则等于 false。当来源字符串的长度大于目的 buffer 的大小时，将不做任何处理，并回传 false 到 success 字段。 success 字段可写可不写。

举例	macro_command main()
	char src1[5] = "abcde"
	char dest1[5]
	bool success1
	success1 = StringCopy(src1[0], dest1[0])
	// success1=true，dest1为 “abcde”
	char dest2[5]
	bool success2
	success2 = StringCopy("12345", dest2[0])
	// success2 = true，dest2为 “12345”
	char src3[10] = "abcdefghij"
	char dest3[5]
	bool success3
	success3 = StringCopy(src3[0], dest3[0])
	// success3 = false，dest3内容不变

	char src4[10] = "abcdefghij" char dest4[5] bool success4 success4 = StringCopy(src4[5], dest4[0]) // success4=true，dest4为 “fghij” end macro_command

函数名称	StringDecAsc2Bin
语法	success = StringDecAsc2Bin(source[start], destination) 或 success = StringDecAsc2Bin(“source” , destination)
描述	此函数将十进制字符串转换成整数。 来源字符串 source 可以为静态字符串 (如 “source”)或是一维字符数组变量 (如source[start])。 destination 必须为一变量，用以存放转换后的整数值。 执行完毕会回传一 bool 类型的值给 success 字段。当转换成功，success等于 true，否则等于 false。 来源字符串必需为十进制字符串，若其包含 ‘0’ ~ ‘9’ 以外的字符，函数将会回传false。 success 字段可写可不写。

举例	macro_command main()
	char src1[5] = "12345"
	int result1
	bool success1
	success1 = StringDecAsc2Bin(src1[0], result1)
	// success1 = true，result1 为 “12345”
	
	char result2
	bool success2
	success2 = StringDecAsc2Bin("32768", result2)
	// success2 = true，但结果超出 result2 所能表达的范围
	
	char src3[2] = "4b"
	char result3
	bool success3
	success3 = StringDecAsc2Bin (src3[0], result3)
	// success3 = false，因 src3 包含 ‘0’ ~ ‘9’ 以外的字符
	
	end macro_command

函数名称	StringBin2DecAsc
语法	success = StringBin2DecAsc (source, destination[start])
描述	此函数将整数转换成十进制字符串。 来源 source 可以为常数或变量。 destination” start” 必须为一维字符数组变量，用以存放转换后的十进制字符串。 执行完毕会回传一 bool 类型的值给 success 字段。当转换成功，success等于 true，否则等于 false。 若转换后的十进制字符串长度大于目标数组的大小，函数将会回传 false。 success 字段可写可不写。

举例	macro_command main()
	int src1 = 2147483647
	char dest1[20]
	bool success1
	success1 = StringBin2DecAsc(src1, dest1[0])
	// success1 = true，dest1为 “2147483647”
	
	short src2 = 0x3c
	char dest2[20]
	bool success2
	success2 = StringBin2DecAsc(src2, dest2[0])
	// success2 = true，dest2为 “60”
	
	int src3 = 2147483647
	char dest3[5]
	bool success3
	success3 = StringBin2DecAsc(src3, dest3[0])
	// success3 = false，dest3 内容不变
	
	end macro_command

函数名称	StringDecAsc2Float
语法	success = StringDecAsc2Float (source[start], destination) 或 success = StringDecAsc2Float (“source” , destination)
描述	此函数将十进制字符串转换成浮点数。 来源字符串 source 可以为静态字符串 (如 “source”)或是一维字符数组变量 (如 source[start])。 destination 必须为一变量，用以存放转换后的浮点数值。 执行完毕会回传一 bool 类型的值给 success 字段。当转换成功，success等于 true，否则等于 false。 来源字符串必需为十进制字符串，若其包含 ‘0’ ~ ‘9’ 或 ‘.’ 以外的字符，函数将会回传 false。 success 字段可写可不写。

举例	macro_command main()
	char src1[10] = "12.345"
	float result1
	bool success1
	success1 = StringDecAsc2Float(src1[0], result1)
	// success1 = true，result1 为 “12.345”
	
	float result2
	bool success2
	success2 = StringDecAsc2Float("1.234567890", result2)
	// success2 = true，但结果超出 result2 所能表达的范围，可能丧失精确度
	
	char src3[2] = "4b"
	float result3
	bool success3
	success3 = StringDecAsc2Float(src3[0], result3)
	// success3 = false，因 src3 包含 ‘0’ ~ ‘9’ 或 ‘.’ 以外的字符
	
	end macro_command

函数名称	StringFloat2DecAsc
语法	success = StringFloat2DecAsc(source, destination[start])
描述	此函数将浮点数转换成十进制字符串。 来源 source 可以为常数或变量。 Destination[start] 必须为一维字符数组变量，用以存放转换后的十进制字符串。 执行完毕会回传一 bool 类型的值给 success 字段。当转换成功，success等于 true，否则等于 false。 若转换后的十进制字符串长度大于目标数组的大小，函数将会回传 false。 success 字段可写可不写。

举例	macro_command main()
	float src1 = 1.2345
	char dest1[20]
	bool success1
	success1 = StringFloat2DecAsc(src1, dest1[0])
	// success1 = true，dest1为 “1.2345”
	
	float src2 = 1.23456789
	char dest2 [20]
	bool success2
	success2 = StringFloat2DecAsc(src2, dest2 [0])
	// success2 = true，但可能丧失精确度
	
	float src3 = 1.2345
	char dest3[5]
	bool success3
	success3 = StringFloat2DecAsc(src3, dest3 [0])
	// success3 = false，dest3 内容不变
	
	end macro_command

函数名称	StringHexAsc2Bin
语法	success = StringHexAsc2Bin (source[start], destination) 或 success = StringHexAsc2Bin (“source” , destination)
描述	此函数将十六进制字符串转换成整数。 来源字符串 source 可以为静态字符串 (如 “source”)或是一维字符数组变量 (如 source” start”)。 destination 必须为一变量，用以存放转换后的整数值。 执行完毕会回传一 bool 类型的值给 success 字段。当转换成功，success等于 true，否则等于 false。 来源字符串必需为十六进制字符串，若其包含 ‘0’ ~ ‘9’ 或 ‘a’ ~ ‘f’ 或 ‘A’ ~ ‘F’ 以外的字符，函数将会回传 false。 success 字段可写可不写。

举例	macro_command main()
	char src1[5]="0x3c"
	int result1
	bool success1
	success1 = StringHexAsc2Bin(src1[0], result1)
	// success1 = true，result1 为 3c
	
	short result2
	bool success2
	success2 = StringDecAsc2Bin("1a2b3c4d", result2)
	// success2 = true，但结果超出 result2 所能表达的范围，result2 = 3c4d
	
	char src3[2] = "4g"
	char result3
	bool success3
	success3 = StringDecAsc2Bin (src3[0], result3)
	// success3=false，因 src3包含 ‘0’ ~ ‘9’ 或 ‘a’ ~ ‘f’ 或 ‘A’ ~ ‘F’ 以外的字符
	
	end macro_command

函数名称	StringBin2HexAsc
语法	success = StringBin2HexAsc (source, destination[start])
描述	此函数将整数转换成十六进制字符串。 来源source可以为常数或变量。 Destination[start] 必须为一维字符数组变量，用以存放转换后的十六进制字符串。 执行完毕会回传一 bool 类型的值给 success 字段。当转换成功，success等于 true，否则等于 false。 若转换后的十六进制字符串长度大于目标数组的大小，函数将会回传 false。 success 字段可写可不写。

举例	macro_command main()
	int src1 = 20
	char dest1[20]
	bool success1
	success1 = StringBin2HexAsc(src1, dest1[0])
	// success1 = true，dest1 为 “14”
	
	short src2 = 0x3c
	char dest2[20]
	bool success2
	success2 = StringBin2HexAsc(src2, dest2[0])
	// success2 = true，dest2 为 “3c”
	
	int src3 = 0x1a2b3c4d
	char dest3[6]
	bool success3
	success3 = StringBin2HexAsc(src3, dest3[0])
	// success3 = false，dest3 内容不变
	
	end macro_command

函数名称	StringMid
语法	success = StringMid (source[start], count, destination[start]) 或 success = StringMid (“string” , start, count, destination[start])
描述	利用此函数可将一个字符串中的某一段子字符串提取出来。 来源字符串 source 可以为静态字符串 (如 “source”)或是一维字符数组变量 (如 source[start])。当来源字符串为字符数组变量时，由数组下标决定子字符串起始位置。当来源字符串为静态字符串时，由第二个参数 start 决定子字符串起始位置。 count 决定要提取的子字符串长度。 Destination[start] 必须为一维字符数组变量，用以存放提取出来的子字符串。 执行完毕会回传一 bool 类型的值给 success 字段。当执行成功，success等于 true，否则等于 false。 若提取出来的子字符串长度大于目标数组的大小，函数将会回传false。 success 字段可写可不写。

举例	macro_command main()
	char src1[20] = "abcdefghijklmnopqrst"
	char dest1[20]
	bool success1
	success1 = StringMid(src1[5], 6, dest1[0])
	// success1 = true，dest1 为 “fghijk”
	
	char src2[20] = "abcdefghijklmnopqrst"
	char dest2[5]
	bool success2
	success2 = StringMid(src2[5], 6, dest2[0])
	// success2 = false，dest2 内容不变
	
	char dest3[20] = "12345678901234567890"
	bool success3
	success3 = StringMid("abcdefghijklmnopqrst", 5, 5, dest3[15])
	// success3 = true，dest3 = “123456789012345fghij”
	
	end macro_command

函数名称	StringLength
语法	length = StringLength (source[start]) 或 length = StringLength ([source])
描述	取得字符串的长度。 来源字符串 source 可以为静态字符串 (如 “source”) 或是一维字符数组变量 (如 source[start])。 函数回传值代表来源字符串的长度。

举例	macro_command main()
	char src1[20] = "abcde"
	int length1
	length1= StringLength(src1[0])
	// length1 = 5
	
	char src2[20] = { 'a', 'b' , 'c' , 'd' , 'e' }
	int length2
	length2= StringLength(src2[0])
	// length2 = 20
	
	char src3[20] = "abcdefghij"
	int length3
	length3 = StringLength(src3 [2])
	// length3 = 8
	
	end macro_command

函数名称	StringCat
语法	success = StringCat (source[start], destination[start]) 或 success = StringCat (“source” , destination[start])
描述	利用此函数将来源字符串衔接于目标字符串之后。此函数执行成功后，目标字符串将等于两字符串衔接后的结果。 来源字符串source可以为静态字符串 (如 “source”) 或是一维字符数组变量(如source[start])。 Destination[start]必须为一维字符数组变量。 执行完毕会回传一 bool 类型的值给 success 字段。当执行成功，success等于true，否则等于 false。当两字符串衔接后的长度超过目标数组的长度时，目标字符串将保留衔接以前的内容，不做任何改变，并回传 false。

举例	macro_command main()
	char src1[20] = "abcdefghij"
	char dest1[20] = "1234567890"
	bool success1
	success1= StringCat(src1[0], dest1[0])
	// success1 = true，dest1 = "123456790abcdefghij"
	
	char dest2 [10] = "1234567890"
	bool success2
	success2= StringCat("abcde", dest2 “0”)
	// success2 = false，dest2 内容不变
	
	char src3[20] = "abcdefghij"
	char dest3[20]
	bool success3
	success3 = StringCat(src3[0], dest3[15])
	// success3 = false，dest3内容不变
	
	end macro_command

函数名称	StringCompare
语法	ret = StringCompare (str1[start], str2[start]) ret = StringCompare (“string1”, str2[start]) ret = StringCompare (str1[start], “string2”) ret = StringCompare (“string1”, “string2”)
描述	比较两字符串的内容是否相等。此函数将大小写视为不同。 两个传入的字符串皆可以为静态字符串 (如 “source”)或是一维字符数组变量 (如 source[start])。 执行完毕会回传一 bool类型的值给 ret 字段。若两字符串相等，ret 为 true，否则为 false。

举例	macro_command main()
	char a1[20] = "abcde"
	char b1[20] = "ABCDE"
	bool ret1
	ret1 = StringCompare(a1[0], b1[0])
	// ret1 = false
	
	char a2[20] = "abcde"
	char b2[20] = "abcde"
	bool ret2
	ret2 = StringCompare(a2[0], b2[0])
	// ret2 = true
	
	char a3 [20] = "abcde"
	char b3[20] = "abcdefg"
	bool ret3
	ret3 = StringCompare(a3”0”, b3”0”)
	// ret3 = false
	
	end macro_command

函数名称	StringCompareNoCase
语法	ret = StringCompareNoCase(str1[start], str2[start]) ret = StringCompareNoCase(“string1”, str2[start]) ret = StringCompareNoCase(str1[start], “string2”) ret = StringCompareNoCase(“string1”, “string2”)
描述	比较两字符串的内容是否相等。此函数将大小写视为相同。 两个传入的字符串皆可以为静态字符串 (如 “source”) 或是一维字符数组变量 (如 source[start])。 执行完毕会回传一 bool 类型的值给 ret 字段。若两字符串相等，ret 为 true，否则为 false。

举例	macro_command main()
	char a1[20] = "abcde"
	char b1[20] = "ABCDE"
	bool ret1
	ret1 = StringCompareNoCase(a1[0], b1[0])
	// ret1 = true
	
	char a2[20] = "abcde"
	char b2[20] = "abcde"
	bool ret2
	ret2 = StringCompareNoCase(a2[0], b2[0])
	// ret2 = true
	
	char a3 [20] = "abcde"
	char b3[20] = "abcdefg"
	bool ret3
	ret3 = StringCompareNoCase(a3[0], b3[0])
	// ret3 = false
	
	end macro_command

函数名称	StringFind
语法	position = StringFind (source[start], target[start]) position = StringFind (“source”, target[start]) position = StringFind (source[start], “target”) position = StringFind (“source”, “target”)
描述	寻找某一字符串 (target) 在另一个字符串 (source) 里第一次出现的位置。 两个传入的字符串皆可以为静态字符串 (如 “source”) 或是一维字符数组变量 (如 source[start])。 执行完毕会回传target字符串在 source 字符串里出现的位置。Source 字符串由 0 开始递增为字符编索引值。若 source 字符串存在一个子字符串，其所包含的字符与排列顺序跟 target 字符串完全相等，则函数会回传此子字符串开头的索引值，若没有找到，则回传 -1。

举例	macro_command main()
	char src1[20] = "abcde"
	char target1[20] = "cd"
	bool pos1
	pos1 = StringFind(src1[0], target1[0])
	// pos1 = 2
	
	char target2[20] = "ce"
	bool pos2
	pos2 = StringFind("abcde", target2” 0”)
	// pos2 = -1
	
	char src3[20] = "abcde"
	bool pos3
	pos3 = StringFind(src3[3], "cd")
	// pos3 = -1
	
	end macro_command

函数名称	StringReverseFind
语法	position = StringReverseFind (source[start], target[start]) position = StringReverseFind (“source” , target[start]) position = StringReverseFind (source[start], “target”) position = StringReverseFind (“source” , “target “)
描述	寻找某一字符串 (target) 在另一个字符串 (source) 里最后一次出现的位置。 两个传入的字符串皆可以为静态字符串 (如 “source”) 或是一维字符数组变量 (如 source[start])。 执行完毕会回传 target 字符串在 source 字符串里最后一次出现的位置。source 字符串由 0 开始递增为字符编索引值。若 source 字符串存在一个子字符串，其所包含的字符与排列顺序跟 target 字符串完全相等，则函数会回传此子字符串开头的索引值，若没有找到，则回传 -1。若 source 字符串中存在多个与 target字符串相等的子字符串，则回传最后一个出现的子字符串的位置。

举例	macro_command main()
	char src1[20] = "abcdeabcde"
	char target1[20] = "cd"
	bool pos1
	pos1 = StringReverseFind(src1[0], target1[0])
	// pos1 = 7
	
	char target2[20] = "ce"
	bool pos2
	pos2 = StringReverseFind("abcdeabcde" , target2[0])
	// pos2 = -1
	
	char src3[20] = "abcdeabcde"
	bool pos3
	pos3 = StringReverseFind(src3[6], "ab")
	// pos3 = -1
	
	end macro_command

函数名称	StringFindOneOf
语法	position = StringFindOneOf (source[start], target[start]) position = StringFindOneOf (“source”, target[start]) position = StringFindOneOf (source[start], “target”) position = StringFindOneOf (“source”, “target”)
描述	寻找 target 字符串中任一个字符在 source 字符串中第一次出现的位置。 两个传入的字符串皆可以为静态字符串 (如 “source”) 或是一维字符数组变量 (如 source[start])。 执行完毕会回传 target 字符串中任一个字符在 source 字符串里第一次出现的位置，即 source 字符串中为该字符编的索引值 (由 0 开始)。若没有找到，则回传 -1。

举例	macro_command main()
	char src1[20] = "abcdeabcde"
	char target1[20] = "sdf"
	bool pos1
	pos1 = StringFindOneOf(src1[0], target1[0])
	// pos1 = 3
	
	char src2[20] = "abcdeabcde"
	bool pos2
	pos2 = StringFindOneOf(src2[1], "agi")
	// pos2 = 4
	
	char target3 [20] = "bus"
	bool pos3
	pos3 = StringFindOneOf("abcdeabcde", target3” 1”)
	// pos3 = -1
	
	end macro_command

函数名称	StringIncluding
语法	success = StringIncluding (source[start], set[start], destination[start]) success = StringIncluding (“source” , set[start], destination[start]) success = StringIncluding (source[start], “set” , destination[start]) success = StringIncluding (“source” , “set” , destination[start])
描述	提取 source 字符串中某个以索引值 0 的字符 (第一个字符) 开头的子字符串，而且此子字符串的每个字符都能在 set 字符串中找到相同的字符。此函数将会从 source 字符串的第一个字符开始寻找，直到找到不存在于 set 字符串中的字符为止。 source 字符串与 set 字符串皆可以为静态字符串 (如 “source”) 或是一维字符数组变量 (如 source[start])。 执行完毕会回传一 bool 类型的值给 success 字段。当执行成功，success等于 true，否则等于 false。当提取出来的字符串长度大于目标数组的大小，将会回传 false。

举例	macro_command main()
	char src1[20] = "cabbageabc"
	char set1[20] = "abc"
	char dest1[20]
	bool success1
	success1 = SDtringIncluding(src1[0], set1[0], dest1[0])
	// success1 = true， dest1 = “cabba”
	
	char src2[20] = "gecabba"
	char dest2[20]
	bool success2
	success2 = StringIncluding(src2[0], "abc", dest2[0])
	// success2 = true， dest2 = “ ”
	
	char set3[20] = "abc"
	char dest3[4]
	bool success3
	success3 = StringIncluding("cabbage", set3[0], dest3[0])
	// success3 = false， dest3内容不变
	
	end macro_command

函数名称	StringExcluding
语法	success = StringExcluding (source[start], set[start], destination[start]) success = StringExcluding (“source”, set[start], destination[start]) success = StringExcluding (source[start], “set”, destination[start]) success = StringExcluding (“source”, “set”, destination[start])
描述	提取 source 字符串中某个以索引值 0 的字符 (第一个字符) 开头的子字符串，而且此子字符串的每个字符必不存在于set字符串中。此函数将会从 source字符串的第一个字符开始寻找，直到找到存在于 set 字符串中的字符为止。 source 字符串与 set 字符串皆可以为静态字符串 (如 “source”) 或是一维字符数组变量 (如 source[start])。 执行完毕会回传一 bool 类型的值给 success 字段。当执行成功，success等于 true，否则等于 false。当提取出来的字符串长度大于目标数组的大小，将会回传 false。

举例	macro_command main()
	char src1[20] = "cabbageabc"
	char set1[20] = "ge"
	char dest1[20]
	bool success1
	success1 = StringExcluding(src1[0], set1[0], dest1[0])
	// success1 = true， dest1 = "cabba"
	
	char src2[20] = "cabbage"
	char dest2[20]
	bool success2
	success2 = StringExcluding(src2[0], "abc", dest2[0])
	// success2 = true， dest2 =" "
	
	char set3[20] = "ge"
	char dest3[4]
	bool success3
	success3 = StringExcluding("cabbage", set3[0], dest3[0])
	// success3 = false， dest3 内容不变
	
	end macro_command

函数名称	StringToUpper
语法	success = StringToUpper (source[start], destination[start]) success = StringToUpper (“source” , destination[start])
描述	将 source 字符串的字符全部转换成大写，并写入 destination 字符串。 source 字符串可以为静态字符串 (如 “source”) 或是一维字符数组变量 (如 source[start])。 执行完毕会回传一 bool 类型的值给 success 字段。当执行成功，success等于 true，否则等于 false。source 字符串长度大于目标数组的大小，将会回传 false。

举例	macro_command main()
	char src1[20] = "aBcDe"
	char dest1[20]
	bool success1
	success1 = StringToUpper(src1[0], dest1[0])
	// success1 = true，dest1 = “ABCDE”
	
	char dest2[4]
	bool success2
	success2 = StringToUpper("aBcDe", dest2[0])
	// success2 = false，dest2 内容不变
	
	end macro_command

函数名称	StringToLower
语法	success = StringToLower (source[start], destination[start]) success = StringToLower (“source” , destination[start])
描述	将 source 字符串的字符全部转换成小写，并写入 destination 字符串。 source 字符串可以为静态字符串 (如 “source”)或是一维字符数组变量 (如 source[start])。 执行完毕会回传一 bool 类型的值给 success 字段。当执行成功，success等于 true，否则等于 false。source 字符串长度大于目标数组的大小，将会回传 false。

举例	macro_command main()
	char src1[20] = "aBcDe"
	char dest1[20]
	bool success1
	success1 = StringToUpper(src1[0], dest1[0])
	// success1 = true，dest1 = “abcde”
	
	char dest2[4]
	bool success2
	success2 = StringToUpper("aBcDe", dest2[0])
	// success2 = false，dest2 内容不变
	
	end macro_command

函数名称	StringToReverse
语法	success = StringToReverse (source[start], destination[start]) success = StringToReverse (“source” , destination[start])
描述	将 source 字符串反转，并写入 destination 字符串。 source 字符串可以为静态字符串 (如 “source”) 或是一维字符数组变量 (如 source[start])。 执行完毕会回传一 bool 类型的值给 success 字段。当执行成功，success等于 true，否则等于 false。source 字符串长度大于目标数组的大小，将会回传 false。

举例	macro_command main()
	char src1[20] = "abcde"
	char dest1[20]
	bool success1
	success1 = StringToUpper(src1[0], dest1[0])
	// success1 = true， dest1 = “edcba”
	
	char dest2[4]
	bool success2
	success2 = StringToUpper("abcde", dest2[0])
	// success2 = false， dest2 内容不变
	
	end macro_command

函数名称	StringTrimLeft
语法	success = StringTrimLeft (source[start], set[start], destination[start]) success = StringTrimLeft (“source”, set[start], destination[start]) success = StringTrimLeft (source[start], “set”, destination[start]) success = StringTrimLeft (“source”, “set”, destination[start])
描述	从 source 字符串的第一个字符开始往后寻找，若找到与 set 字符串相同的字符便裁剪掉该字符，直到遇到不存在于 set 字符串中的字符为止。 source 字符串与 set 字符串皆可以为静态字符串 (如 “source”) 或是一维字符数组变量 (如 source[start])。 执行完毕会回传一 bool 类型的值给 success 字段。当执行成功，success等于 true，否则等于 false。当裁剪完后的字符串长度大于目标数组的大小，将会回传 false。

举例	macro_command main()
	char src1[20] = "# *a*#bc"
	char set1[20] = "# *"
	char dest1[20]
	bool success1
	success1 = StringTrimLeft (src1[0], set1[0], dest1[0])
	// success1 = true， dest1 = "a*#bc"
	
	char set2[20] = {'#', ' ', '*'}
	char dest2[4]
	success2 = StringTrimLeft ("# *a*#bc", set2[0], dest2[0])
	// success2 = false， dest2 内容不变
	
	char src3[20] = "abc *#"
	char dest3[20]
	bool success3
	success3 = StringTrimLeft (src3[0], "# *", dest3[0])
	// success3 = true， dest3 = "abc *#"
	
	end macro_command

函数名称	StringTrimRight
语法	success = StringTrimRight (source[start], set[start], destination[start]) success = StringTrimRight (“source”, set[start], destination[start]) success = StringTrimRight (source[start], “set”, destination[start]) success = StringTrimRight (“source”, “set”, destination[start])
描述	从 source 字符串的最后一个字符开始往前寻找，若找到与 set 字符串相同的字符便裁剪掉该字符，直到遇到不存在于 set 字符串中的字符为止。 source 字符串与 set 字符串皆可以为静态字符串 (如 “source”) 或是一维字符数组变量 (如 source[start])。 执行完毕会回传一 bool 类型的值给 success 字段。当执行成功，success等于 true，否则等于 false。当裁剪完后的字符串长度大于目标数组的大小，将会回传 false。

举例	macro_command main()
	char src1[20] = "# *a*#bc# * "
	char set1[20] = "# *"
	char dest1[20]
	bool success1
	success1 = StringTrimRight(src1[0], set1[0], dest1[0])
	// success1 = true，dest1= “# *a*#bc”
	
	char set2[20] = {'#', ' ', '*'}
	char dest2[20]
	success2 = StringTrimRight("# *a*#bc", set2[0], dest2[0])
	// success2 = true，dest2 = “# *a*#bc”
	
	char src3[20] = "ab**c *#"
	char dest3[4]
	bool success3
	success3 = StringTrimRight(src3[0], "# *", dest3[0])
	// success3 = false，dest3 内容不变
	
	end macro_command

函数名称	StringInsert
语法	success = StringInsert (pos, insert[start], destination[start]) success = StringInsert (pos, “insert”, destination[start]) success = StringInsert (pos, insert[start], length, destination[start]) success = StringInsert (pos, “insert”, length, destination[start])
描述	将 insert 字符串插入到目标字符串中的特定位置，插入位置由 pos 所指定。Insert 字符串可以为静态字符串 (如 “insert”) 或是一维字符数组变量 (如 insert[start])。 用户亦可以在 length 字段指定 insert 字符串的长度。 执行完毕会回传一 bool 类型的值给 success 字段。当执行成功，success等于 true，否则等于 false。当插入完成后的字符串长度大于目标数组的大小，将会回传 false。

举例	macro_command main()
	
	char str1[20] = "but the question is"
	char str2[10] = ", that is"
	char dest[40] = "to be or not to be"
	bool success
	
	success = StringInsert(18, str1[3], 13, dest[0])
	// success = true, dest = "to be or not to be the question"
	
	success = StringInsert(18, str2[0], dest[0])
	// success=true, dest="to be or not to be, that is the question"
	
	success = StringInsert(0, "Hamlet:", dest” 0”)
	// success = false, dest 内容不变
	
	end macro_command

18.7.7 配方数据函数

函数名称	RecipeGetData
语法	RecipeGetData (destination, recipe_address, record_ID)
描述	取得配方中的资料。所取得的数据存放在destination且必须为变量。recipe_address由配方名称与项目名称组成："recipe_name.item_name"。record_ID指定欲取得配方中的第几笔数据，也就是数据列编号。

举例	macro_command main()
	int data = 0
	char str[20]
	int recordID
	bool result
	
	recordID = 0
	result = RecipeGetData(data, "TypeA.item_weight", recordID)
	// 从配方 "TypeA" 中取得第0笔，且item name为 "item_weight" 的资料
	
	recordID = 1
	result = RecipeGetData(str[0], "TypeB.item_name", recordID)
	// 从配方 "TypeB" 中取得第1笔，且 item name为 "item_name" 的资料
	
	end macro_command

函数名称	RecipeQuery
语法	RecipeQuery (SQL command, destination)
描述	通过SQL语句，对配方中的数据进行查询。查询结果的数据笔数存放在destination，必须为变量。SQL command 的部份可为静态文字或传入字符数组，例如： RecipeQuery(“SELECT * FROM TypeA” , destination) 或 RecipeQuery(sql[0], destination) SQL 语句必须以 “SELECT * FROM” 开头，后面接配方名称及查询条件。

举例	macro_command main()
	
	int total_row = 0
	char sql[100] = "SELECT * FROM TypeB"
	bool result
	
	result = RecipeQuery("SELECT * FROM TypeA", total_row)
	// 查询配方 "TypeA"。查询结果的数据笔数存放在 total_row.
	
	result = RecipeQuery(sql[0], total_row)
	// 查询配方 "TypeB"。查询结果的数据笔数存放在 total_row.
	
	end macro_command

函数名称	RecipeQueryGetData
语法	RecipeQueryGetData (destination, recipe_address, result_row_no)
描述	取得 RecipeQuery 的查询结果。呼叫此函数前必须先呼叫 RecipeQuery，且 recipe_address 中需指定与 RecipeQuery 相同的配方名称。 result_row_no 指定欲取得查询结果中的第几笔数据。

举例	macro_command main()
	
	int data = 0
	int total_row = 0
	int row_number = 0
	bool result_query
	bool result_data
	
	result_query = RecipeQuery("SELECT * FROM TypeA", total_row)
	// 查询配方 "TypeA"。查询结果的数据笔数存放在 total_row.
	
	if (result_query) then
	for row_number = 0 to total_row - 1
	result_data = RecipeQueryGetData(data, "TypeA.item_weight", row_number)
	next row_number
	end if
	
	end macro_command

函数名称	RecipeQueryGetRecordID
语法	RecipeQueryGetRecordID (destination, result_row_no)
描述	取得 RecipeQuery 查询结果的数据列编号。呼叫此函数前必须先呼叫 RecipeQuery。 result_row_no 指定欲取得查询结果中的第几笔数据的数据列编号，并将资料列编号写入 destination。

举例	macro_command main()
	
	int recorded = 0
	int total_row = 0
	int row_number = 0
	bool result_query
	bool result_id
	
	result_query = RecipeQuery("SELECT * FROM TypeA", total_row)
	// 查询配方 "TypeA"。查询结果的数据笔数存放在 total_row.
	
	if (result_query) then
	for row_number = 0 to total_row - 1
	result_id = RecipeQueryGetRecordID(recordID, row_number)
	next row_number

	end if end macro_command

函数名称	RecipeSetData
语法	RecipeSetData(*source*, *recipe_address*, *record_ID*)
描述	将数据写入配方数据库中。如果成功将传回 True，否则将传回 False。 recipe_address 由配方名称与项目名称组成："recipe_name.item_name"。 record_ID 指定欲修改配方中的第几笔数据，也就是数据列编号。

举例	macro_command main()
	
	int data=99
	char str[20]="abc"
	int recordID
	bool result
	
	recordID = 0
	result = RecipeSetData(data, "TypeA.item_weight", recordID)
	// set data to recipe "TypeA", where item name is "item_weight" and the record ID is 0.
	
	recordID = 1
	result = RecipeSetData(str[0], "TypeB.item_name", recordID)
	// set data to recipe "TypeB", where item name is "item_name" and the record ID is 1.
	
	end macro_command

18.7.8 其它函数

函数名称	Beep
语法	Beep ()
描述	发出系统警示音。 此函数可发出 800 赫兹，30 毫秒的系统警示音。
举例	macro_command main() Beep() end macro_command

函数名称	SYNC_TRIG_MACRO
语法	SYNC_TRIG_MACRO(macro_id)
描述	一个执行中的宏指令可以使用此函数利用同步的方式触发执行其它的宏指令，使用 macro_id 指定被触发的宏指令 “编号” 。

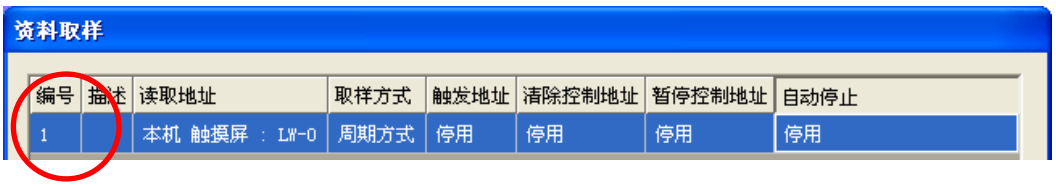
	使用此函数的宏指令将暂停执行，直到被触发的宏指令执行完成才会继续执行后续的命令。 macro_id 可以是常数或使用变量来表示。
举例	macro_command main() char ON = 1, OFF = 0 SetData(ON, “Local HMI”, LB, 0, 1) SYNC_TRIG_MACRO(5) // call a macro (its ID is 5) SetData(OFF, “Local HMI”, LB, 0, 1) end macro_command

函数名称	ASYNC_TRIG_MACRO
语法	ASYNC_TRIG_MACRO(macro_id)
描述	一个执行中的宏指令可以使用此函数，利用异步的方式触发执行其它宏指令，使用 macro_id 指定被触发的宏指令 “编号”。 宏指偷在使用此函数后将立即执行后续的命令，不需等待被触发的宏指令执行完成。 macro_id 可以是常数或使用变量来表示。
举例	macro_command main() char ON = 1, OFF = 0 SetData(ON, “Local HMI”, LB, 0, 1) ASYNC_TRIG_MACRO(5) // call a macro (its ID is 5) SetData(OFF, “Local HMI”, LB, 0, 1) end macro_command

函数名称	TRACE
语法	TRACE(format, argument)
描述	一个执行中的 宏指令可以使用此函数，监视变量内容的变化，并打印字符串，以协助除错。用户应开启 EasyDiagnoser 观看此函数的输出结果。 当 TRACE 函数抓取一个%开头的特殊字符，将同时从 argument 抓取一个参数做格式化后输出。 Format 代表打印格式，支持 % 开头的特殊字符。特殊字符格式如下，其中方括号内的字段为可选，粗体字字段为必需： %” flags” “width” “.precision” type 每个字段的意义如下所述： flags (可选)： -

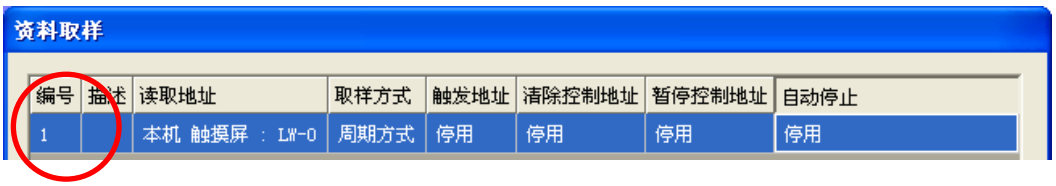
	+ width (可选)： 十进制正整数，指定应预留的字符宽度，不足部份补空格符。 precision (可选)： 十进制正整数，指定精确度，以及输出字符数。 type： C 或 c ： 以字符方式输出 d ： 以 signed 十进制整数输出 i ： 以 signed 十进制整数输出 o ： 以 unsigned 八进位整数输出 u ： 以 unsigned 十进制整数输出 X 或 x ： 以 unsigned 十六进制整数输出 E 或 e ： 以科学表示法输出。格式为” - “d.dddd e “sign” ddd。其中字段d是十进制数，字段dddd是一至多个十进制数，字段ddd必须是三个十进制数。sign是+或-。 f ： 以单倍精确度浮点数输出。格式为” - “dddd.dddd。 *Format* 字符串最长支持 256 个字符，多出的字符将被忽略。 *Argument* 部份可写可不写。但一个特殊字符应搭配一个变量。

举例	macro_command main()
	char c1 = ‘a’
	short s1 = 32767
	float f1 = 1.234567
	
	TRACE(“The results are”) // 输出：The results are
	TRACE(“c1 = %c, s1 = %d, f1 = %f” , c1, s1, f1)
	// 输出：c1 = a, s1 = 32767, f1 = 1.234567
	
	end macro_command

函数名称	FindDataSamplingDate
语法	return_value = FindDataSamplingDate (data_log_number, index, year, month, day) or FindDataSamplingDate (data_log_number, index, year, month, day)
描述	利用此函数查询资料取样文件的日期。根据所输入的资料取样编号 (data_log_number) 与资料取样文件索引 (index) 可查询该资料取样的日期，依照年、月、日的顺序写入 year、month、day 的字段中。  资料取样文件的储存方式为：” 保存位置” \ “取样文件夹名称” \ yyyymmdd.dtl。而资料取样文件索引 (index) 指的是取样文件夹下面所有资料取样文件依文件名

	排序后的顺位值 (从0开始)。Index 值越小者，日期越新。例如假设某取样文件夹下面有四个资料取样文件： 20101210.dtl 20101230.dtl 20110110.dtl 20110111.dtl 则index依序为： 20101210.dtl -> index 为 3 20101230.dtl -> index 为 2 20110110.dtl -> index 为 1 20110111.dtl -> index 为 0 当成功搜寻到所欲查询的资料取样文件时，将回传 1 至 return_value，否则将回传 0。 data_log_number 与 index 可以为常数或是变量，但 year、month、day与 return_value 必须为变量。 return_value 为可选。

举例	macro_command main()
	short data_log_number = 1, index = 2, year, month, day
	short success
	
	// 若存在一资料取样文件 20101230.dtl，其资料取样编号为 1，文件索引为 2
	// 则success == 1，year == 2010，month == 12，day == 30
	success = FindDataSamplingDate(data_log_number, index, year, month, day)
	
	end macro_command

函数名称	FindDataSamplingIndex
语法	return_value = FindDataSamplingIndex (data_log_number, year, month, day, index) or FindDataSamplingIndex (data_log_number, year, month, day, index)
描述	利用此函数查询资料取样文件的文件索引值。根据所输入的资料取样编号 (data_log_number) 与日期可查询该资料取样文件的文件索引，并将其写入 index。year、month、day 三个字段依序代表年、月、日，其输入格式为 YYYY (年)、MM (月)、DD (日)。  资料取样文件的储存方式为：”保存位置” \ “取样文件夹名称” \ yyyymmdd.dtl。而资料取样文件索引 (index) 指的是取样文件夹下面所有资料取样文件依文件名排序后的顺位值 (从 0 开始)。Index 值越小者，日期越新。例如假设某取样文件夹下面有四个资料取样文件： 20101210.dtl 20101230.dtl 20110110.dtl 20110111.dtl 则 index 依序为： 20101210.dtl -> index 为 3 20101230.dtl -> index 为 2 20110110.dtl -> index 为 1 20110111.dtl -> index 为 0 当成功搜寻到所欲查询的资料取样文件时，将回传 1 至 return_value，否则将回传 0。 data_log_number 与 year、month、day 可以为常数或是变量，但 index 与 return_value 必须为变量。 return_value 为可选。

举例	macro_command main()
	short data_log_number = 1, year = 2010, month = 12, day = 10, index
	short success
	
	// 若存在一资料取样文件 20101210.dtl，其资料取样编号为 1，文件索引为 2
	// 则success == 1，index == 2
	success = FindDataSamplingIndex (data_log_number, year, month, day, index)
	
	end macro_command

函数名称	FindEventLogDate
语法	return_value = FindEventLogDate (index, year, month, day) or FindEventLogDate (index, year, month, day)
描述	利用此函数查询事件登录文件的日期。根据所输入的事件登录文件索引 (index)可查询该事件登录的日期，依照年、月、日的顺序写入 year、month、day的字段中。事件登录文件索引 (index) 指的是指定的保存位置 (HMI、SD 卡或 USB)下事件登录文件依档名排序后的顺位值 (从 0 开始)。Index 值越小者，日期越新。例如假设有四个事件登录档： EL_20101210.evt EL_20101230.evt EL_20110110.evt EL_20110111.evt 则index依序为： EL_20101210.evt -> index为3 EL_20101230.evt -> index为2 EL_20110110.evt -> index为1 EL_20110111.dtl -> index为0 当成功搜寻到所欲查询的事件登录档时，将回传 1 至return_value，否则将回传 0。 Index 可以为常数或是变量，但 year、month、day与return_value 必须为变量。return_value 为可选。

举例	macro_command main()
	short index = 1, year, month, day
	short success
	
	// 若存在一事件登录档 EL_20101230.evt，文件索引为 1
	// 则 success == 1，year == 2010，month == 12，day == 30
	success = FindEventLogDate (index, year, month, day)
	
	end macro_command

函数名称	FindEventLogIndex
语法	return_value = FindEventLogIndex (year, month, day, index) or FindEventLogIndex (year, month, day, index)
描述	利用此函数查询事件登录文件的文件索引值。根据所输入的日期可查询该事件登录文件的文件索引，并将其写入 index。year、month、day 三个字段依序代表年、月、日，其输入格式为 YYYY (年)、MM (月)、DD (日)。 事件登录文件索引 (index) 指的是指定的保存位置 (HMI、SD 卡或 USB)下事件登录文件依档名排序后的顺位值 (从 0 开始)。Index 值越小者，日期越新。例如假设有四个事件登录档： EL_20101210.evt EL_20101230.evt EL_20110110.evt EL_20110111.evt 则 index 依序为： EL_20101210.evt -> index 为 3 EL_20101230.evt -> index 为 2 EL_20110110.evt -> index 为 1 EL_20110111.dtl -> index 为 0 当成功搜寻到所欲查询的事件登录档时，将回传 1 至 return_value，否则将回传 0。 year、month、day 可以为常数或是变量，但 index 与 return_value 必须为变量。return_value 为可选。

举例	macro_command main()
	short year = 2010, month = 12, day = 10, index
	short success
	
	// 若存在一事件登录档 EL_20101210.evt，文件索引为 2
	// 则success == 1，index == 2
	success = FindEventLogIndex (year, month, day, index)
	
	end macro_command

18.8 怎样建立和执行宏指令

18.8.1 怎样建立一个宏指令

按照以下步骤以建立一个宏指令。

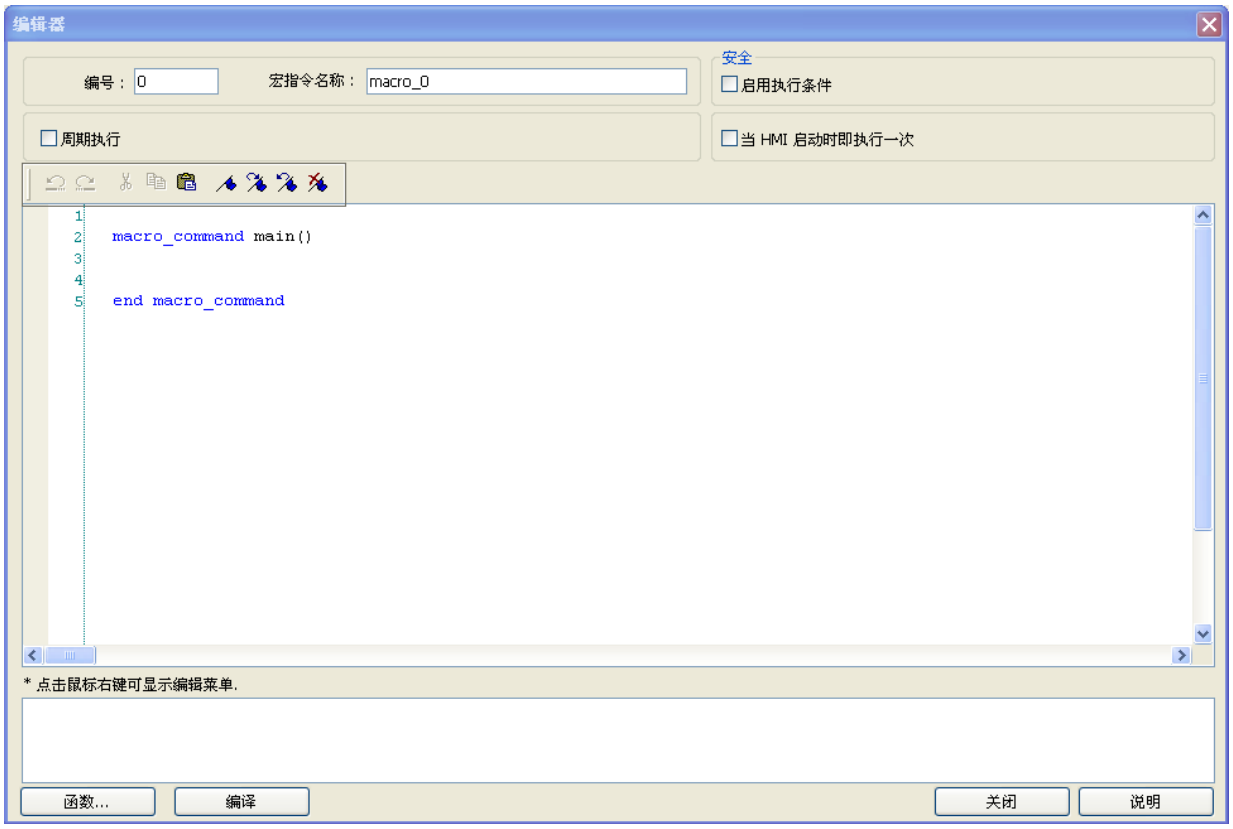
1. 单击 EasyBuilder Pro 软件工具栏上的宏指令图示 打开宏指令管理对话框。

在宏指令管理对话框中，已经编译成功的宏指令会出现在“已编译成功”列表上，未完成编译的会出现在“未完成编译”列表中。下面是宏指令管理对话框中各按键的功能描述。

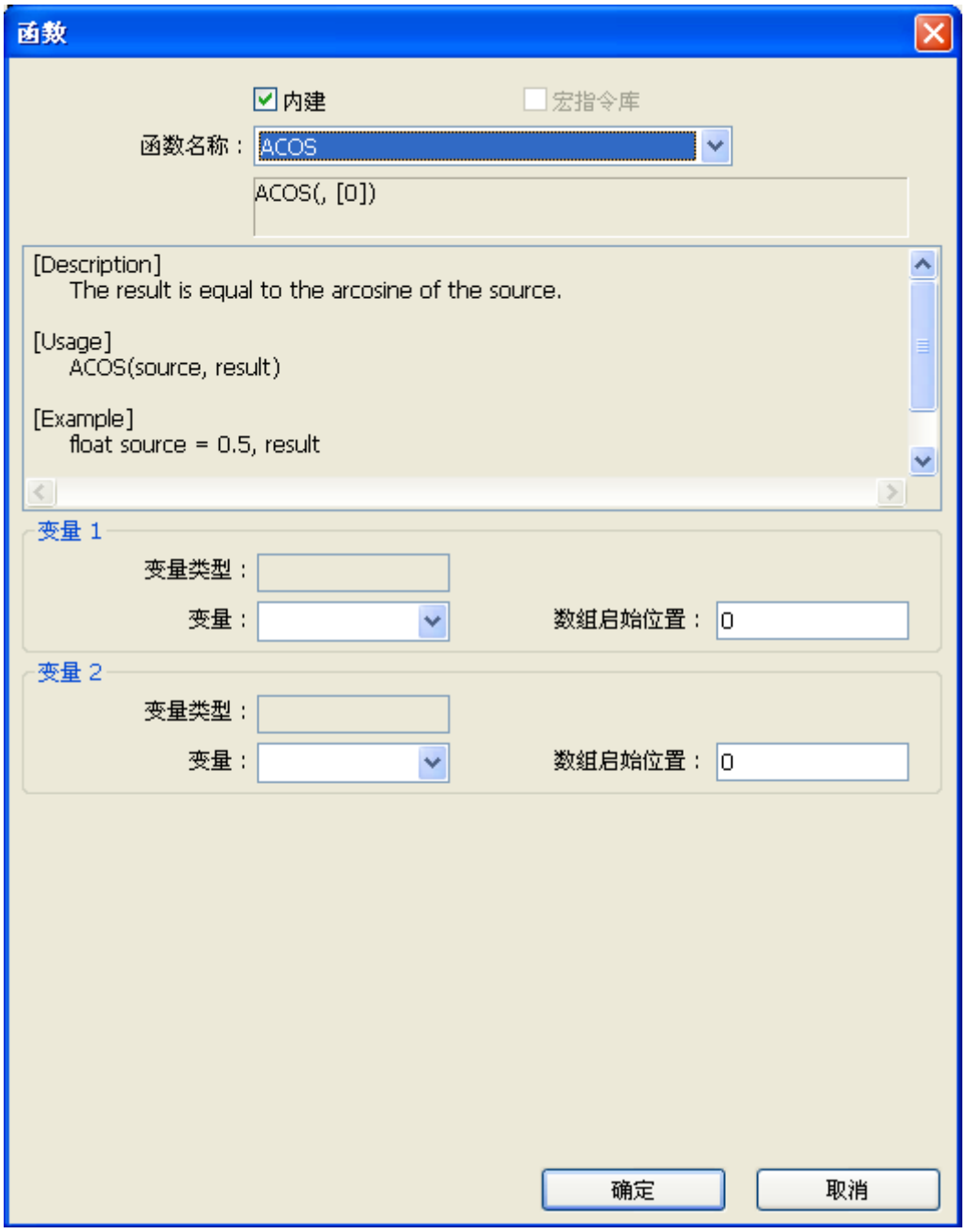
设定	描述
新增	新增一个宏指令，并开启新建的宏指令编辑区。
删除	删除选择的宏指令。
编辑	打开宏指令编辑区，并开启选择的宏指令。
复制	复制选择的宏指令。
贴上	将刚刚选择需要复制的宏指令，贴至“已编译完成”列表区，并产生一个

	新的宏指令名称。
确认	确认此次所编辑的所有宏指令后离开此宏指令管理对话框，必须按此键才会储存　此次编辑内容。
取消	取消此次所编辑的所有宏指令，离开宏指令管理对话框。
宏指令库	进入宏函数库管理对话框。

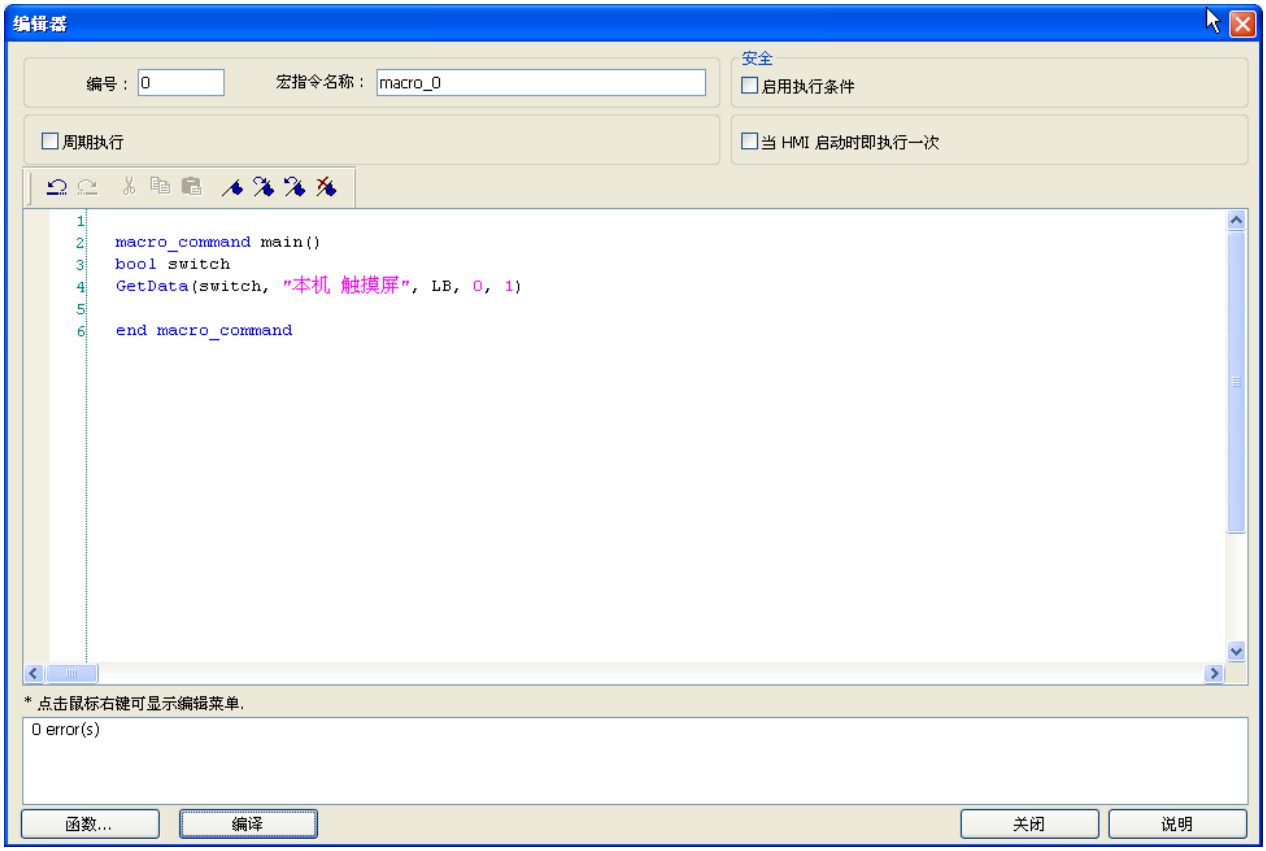
2. 按下“新增”按钮，打开一个新增的宏指令编辑区。每一个宏指令都有一个唯一的编号，定义在“编号”这个位置。在“宏指令名称”这个栏目中也必须输入宏指令的名称，否则编译将无法通过。



3. 设计属于您的宏指令程序。如果要使用内建的函数，譬如 Setdata() 或者 Getdata() 等函数，单击“函数”按键开启函数列表对话框，选择需要的函数，并设定必要的参数。



4. 编辑完成一个新建的宏指令程序后，单击“编译”按键，对该宏指令进行编译工作。



5. 如果没有错误，单击“关闭”按键，这样在“已编译成功”区会发现新增了一个“test”这个名称的宏指令。

18.8.2 执行宏指令

执行宏指令有多种不同的方法，下面分别说明。

- 使用“PLC 控制”元件

1. 打开“PLC 控制”元件，并设定属性为“执行宏指令”。
2. 选择需要执行的宏指令名称。选择一个位作为触发宏指令并设定触发宏指令的条件。在条件满足时，该宏指令将会被重复执行。为了每次只让宏指令执行一次，设计时需在宏指令将该触发位复位。
3. 使用一个“位状态设置” 元件或者“位状态切换开关”元件作为这个位的控制开关。

- 使用“位状态设置” 元件或者“位状态切换开关”元件

1. 在“位状态设置” 元件或者“位状态切换开关”元件的一般属性页中，勾选“使用宏指令”。
2. 选择要执行的宏指令。当这个元件被执行时，选择的宏指令就会被执行一次。

- 使用“功能键”

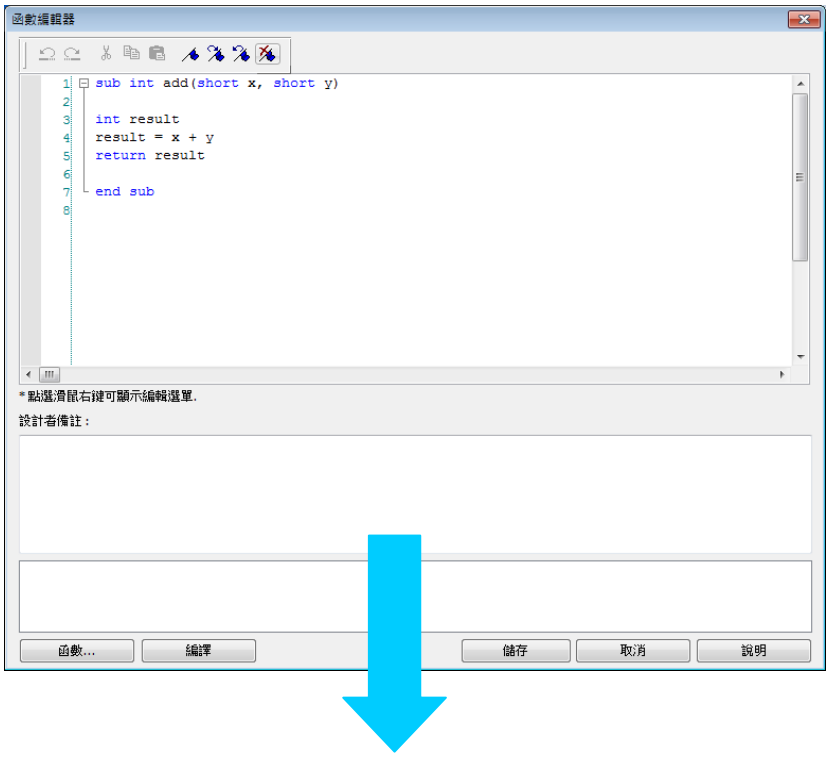
1. 在“功能键”的一般属性页对话框中，勾选“触发宏指令”。
2. 选择需要执行的宏指令。每按一下这个功能键时，选择的宏指令就会被执行一次。

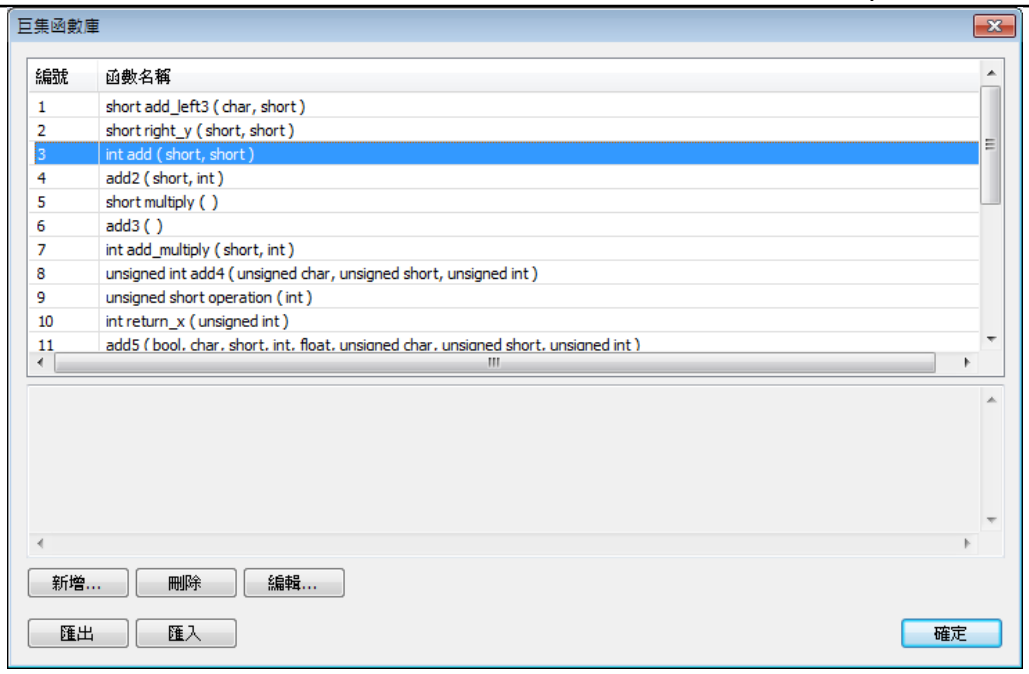
- 使用“宏指令编辑器”设定条件

1. “周期执行”：可以设定每间隔几秒自动执行宏指令一次。
2. “当 HMI 启动时即执行一次”：当触摸屏重新上电或重新启动时会执行此宏指令一次。

19.9 用户自定义函数功能

在使用宏编辑器时，为了减少定义函数的时间，用户可从内建的函数库搜寻所需的函数。然而，当用户编辑宏时，若有某些特定的函数常常使用却无法从内建函数库搜寻到时，就可以自行定义所需的函数并储存起来。当下次需要再定义相同的函数时，可由“宏指令库”呼叫出已储存的函数，方便函数编辑。另外，”宏指令库”也大幅提升了用户自订函数之可移植性。建立函数前可先查看现有内建函数或在线函数库是否有现成函数可使用。





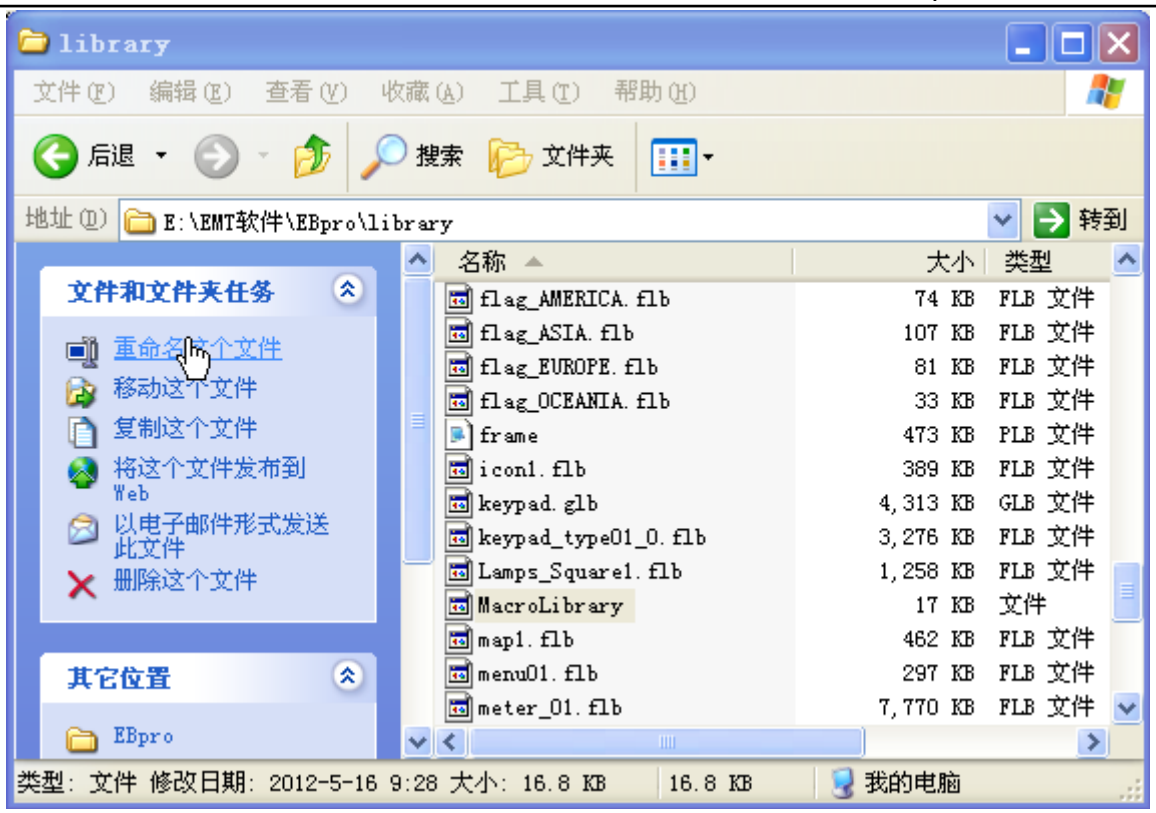
18.9.1 导入函数库文件

HMI 编辑软件开启一个工程文件时，会自动去读入一个预设函数库文件，并加载函数信息。当开启一项目有呼叫到用户定义函数时必须先加载相关的 .mlb 文件。

1. 预设函数库文件名：MacroLibrary (没有扩展名)
2. 函数库路径：HMI 编辑软件的安装目录的 \library 文件夹下面。
3. \library 文件夹下面可以看到两种函数库文件：

   没有扩展名： MacroLibrary，为预设函数库，触摸屏编辑软件指定读入此文件。

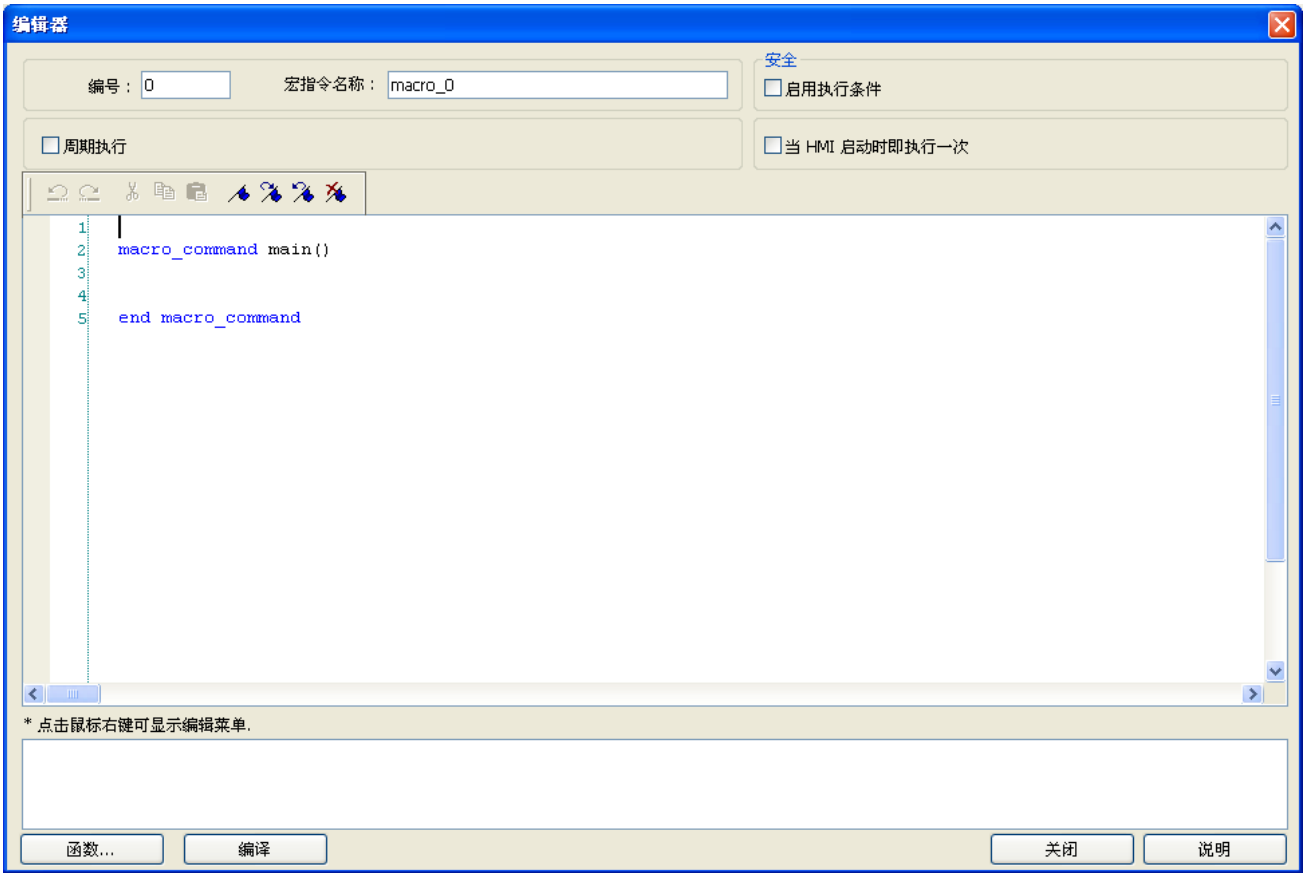
   有扩展名 (.mlb)：例如 math.mlb，用户导入/导出时会去读 / 写的文件，具可移植性，欲使用时再从文件夹呼叫出文件即可。
4. EasyBuilder Pro 开启时，只会去加载预设函数库中的函数，用户若需要用到 .mlb 文件内的函数时，必须自行将其导入。



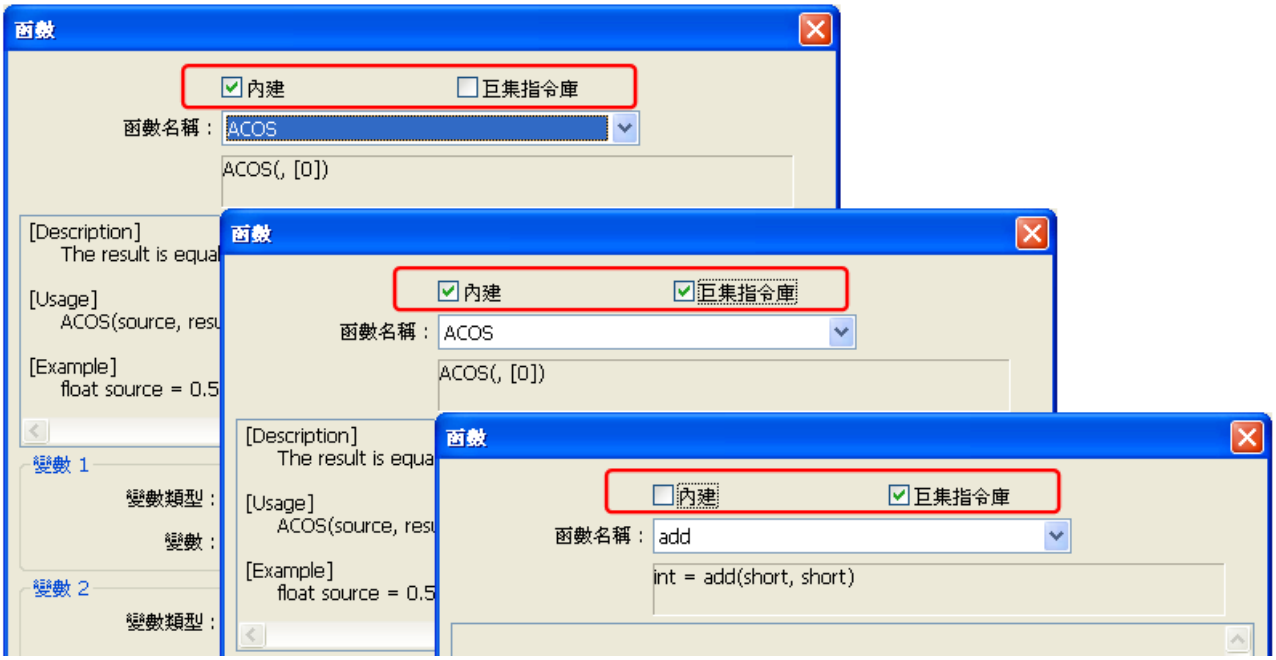
18.9.2 如何使用宏函数库

1. 在宏编辑器中直接呼叫函数。

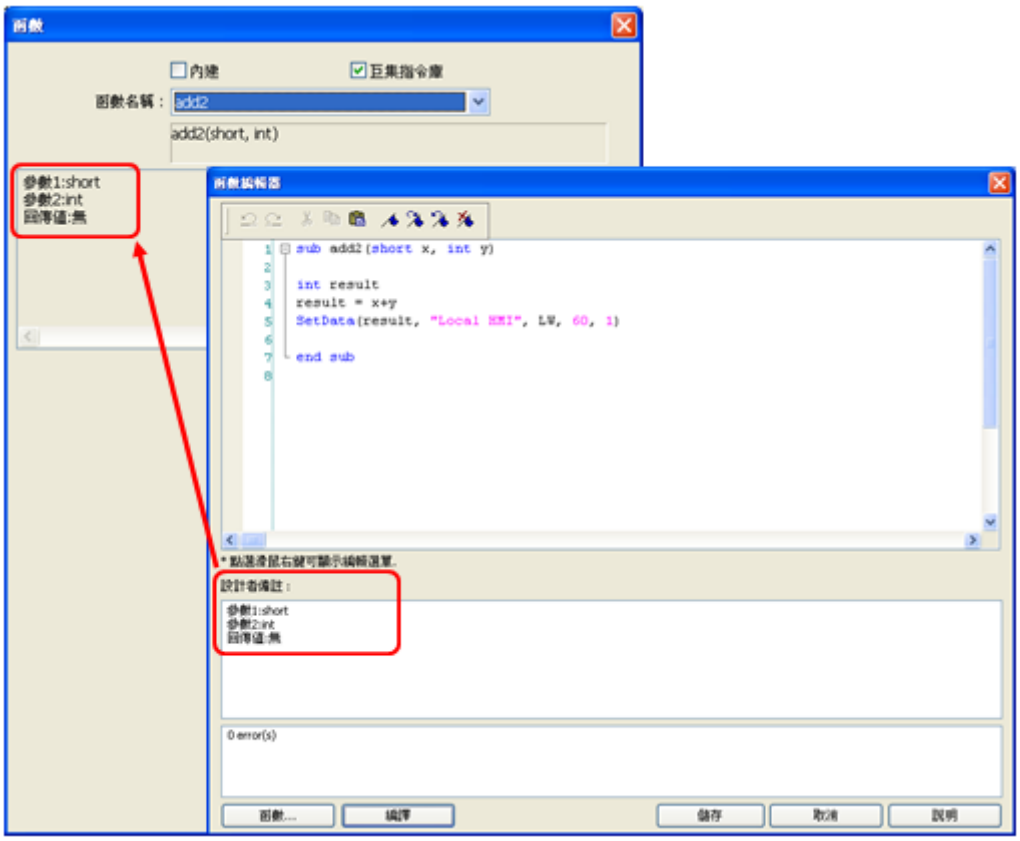
2. 按下宏编辑器左下角的“函数”按钮开启函数对话框。



3. 须至少勾选一项“宏指令库”或“内建”，并选择欲使用的函数。



4. 显示函数说明文字，即是用户在函数编辑器中编辑的说明文字。



5. 选取欲使用的函数，将函数库中的“变量类型”修改为指定的变量名称。

```
1
2   macro_command main()
3
4   short a
5   int b,result
6
7   add2(short, int)
8
9   end macro_command
```

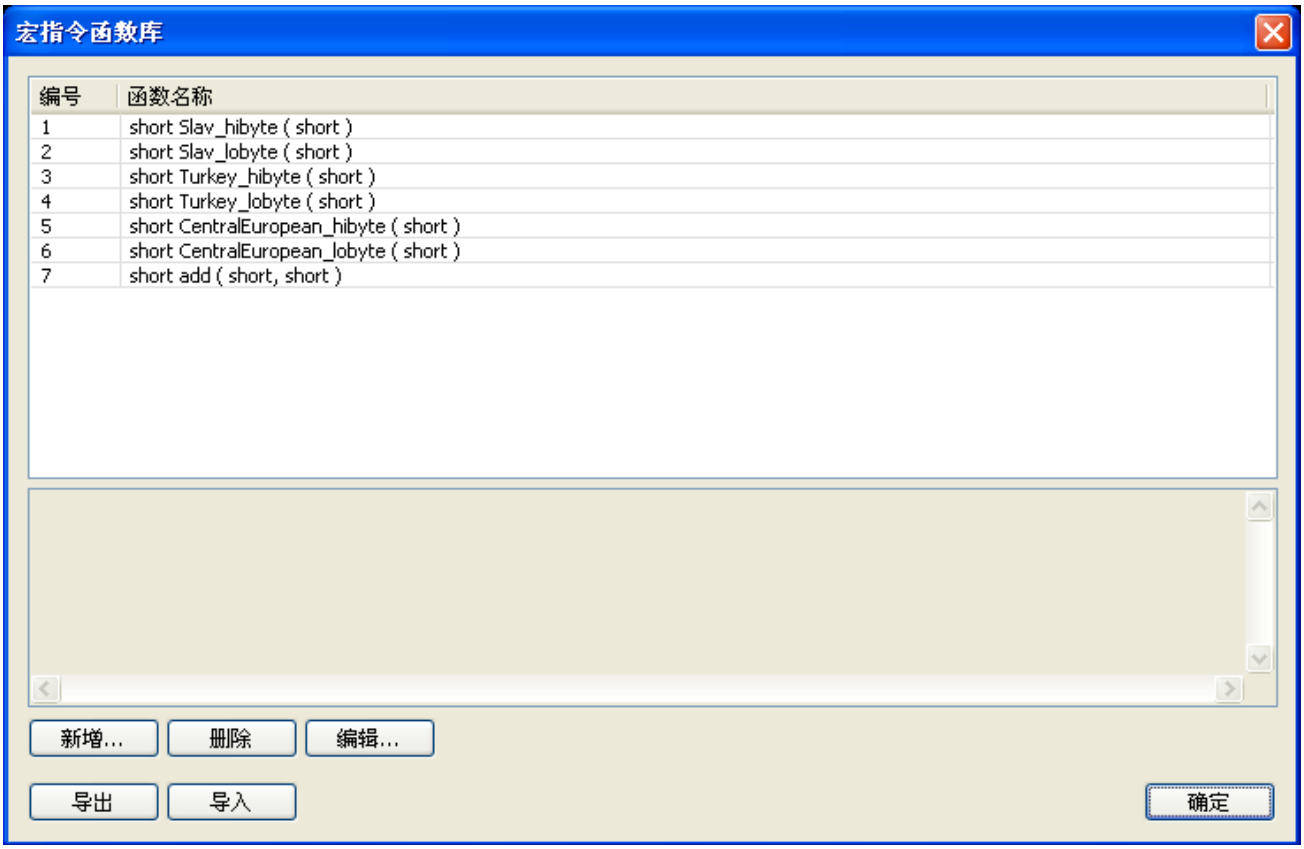
```
1
2   macro_command main()
3
4   short a
5   int b,result
6
7   result = add2(a, b)
8
9   end macro_command
```

6. 用户根据以上的步骤，即可完成使用自订函数的功能，有效节省了重复定义相同的函数。

18.9.3 函数库管理接口

1. 开启宏管理对话框，按下右下角“宏指令库”按钮，进入函数库管理对话框接口。

2. 函数库管理对话框中有函数列表。此列表列出工程文件开启时，触摸屏编辑软件会从预设函数库加载所有函数。



3. 函数列表中每一行的格式如下：

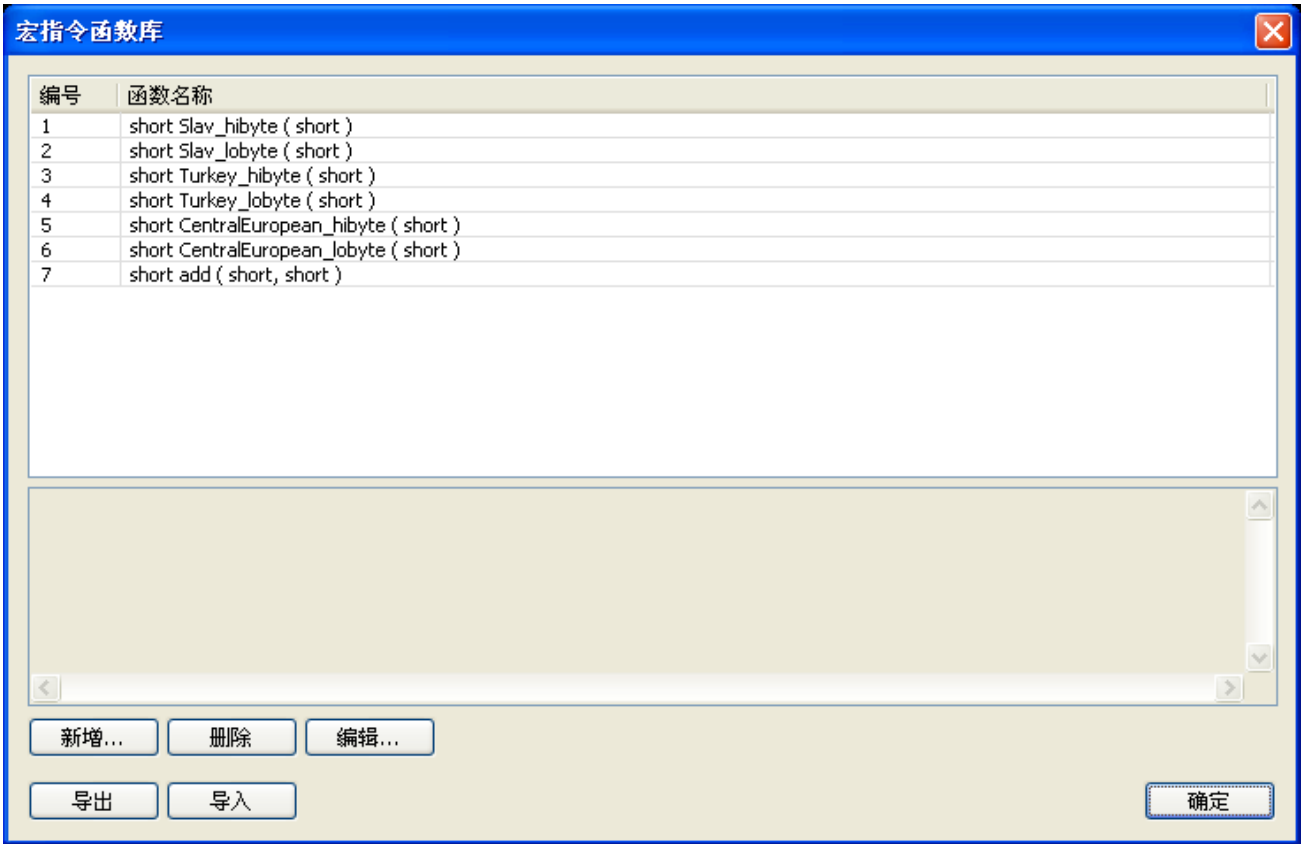
return_type function_name (*parameter_type1*, ..., *parameter_typeN*)

return_type 表示回传值类型，若无回传值则此字段省略；function_name 表示函数名称。*parameter_typeN* 表示第 N 个参数类型，若此函数不接受任何参数，此字段省略。

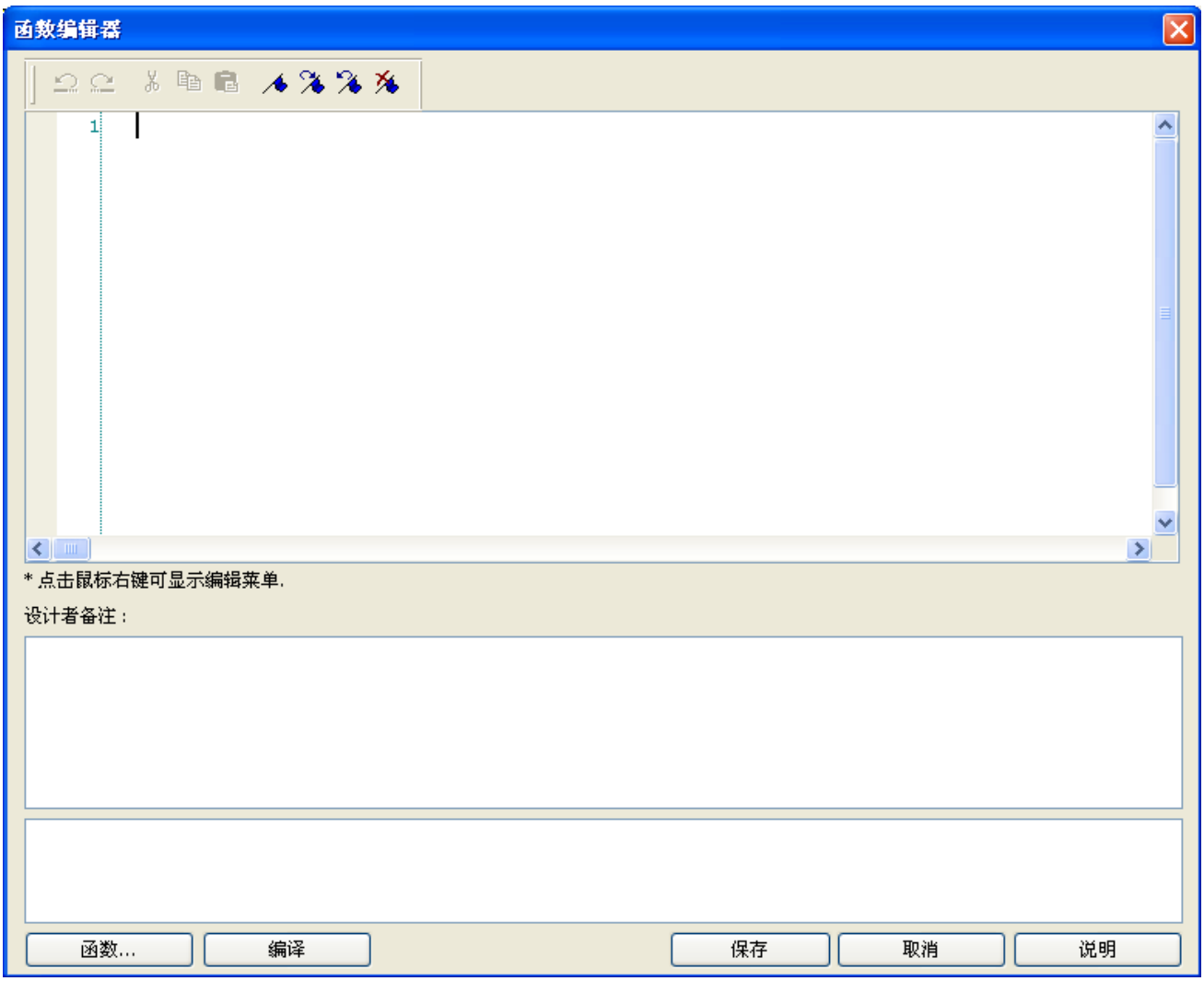
```
1  sub int ADD(int a, int b)
2    int ret
3    ret = a+b
4    return ret
5  end sub
6
```

新建函数

1. 按下“新增”进入函数编辑器。

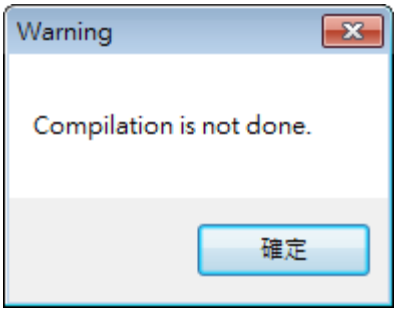


2. 在函数编辑区编辑函数，接着按“编译”再按“储存”。

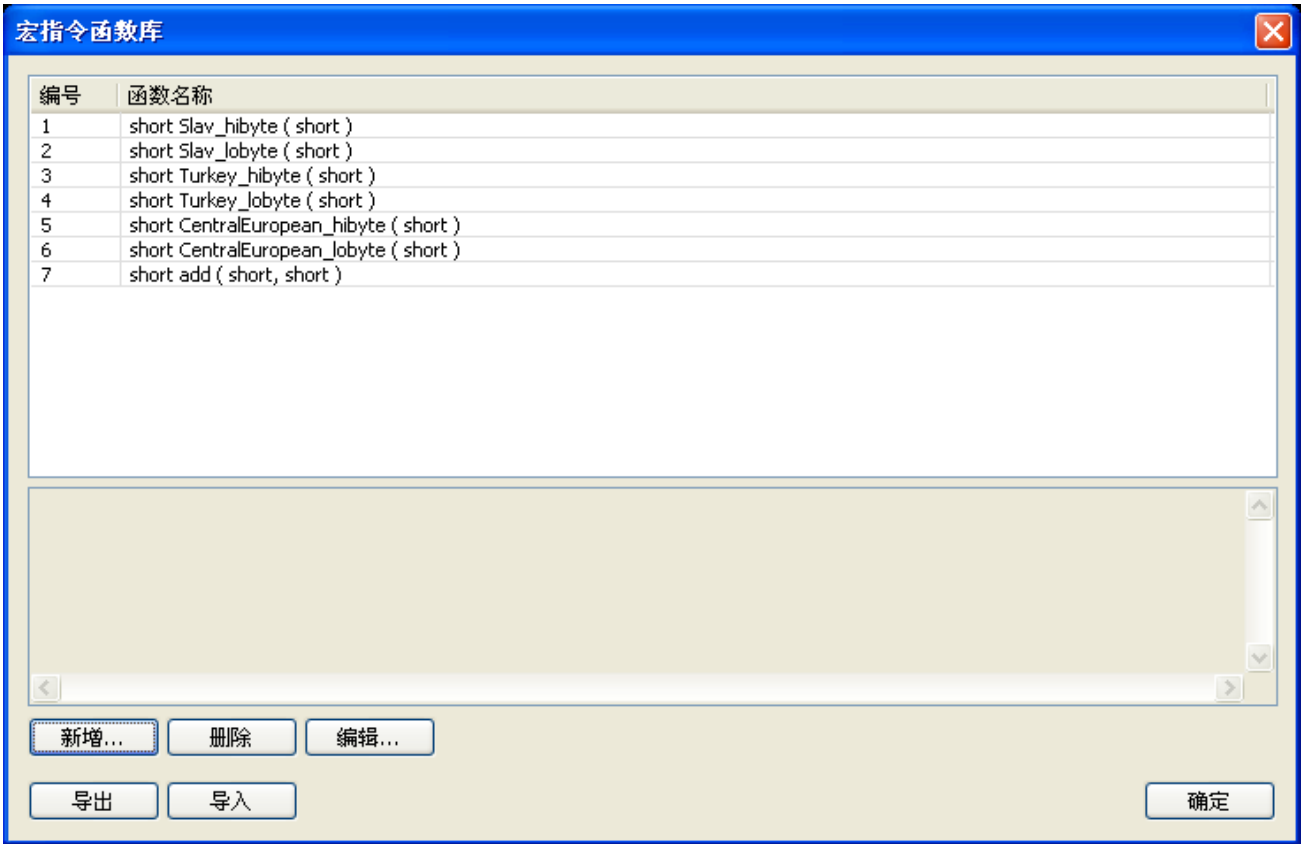


3. 用户可以在函数说明编辑区中编辑说明文字，说明此函数的规格、使用方法、作者声明等等。

4. 函数编辑完成后，必须通过编译，才可以按下“储存”写入函数库。否则会出现下图弹出窗口，警告用户此函数未完成编译。



5. 成功加入函数库。

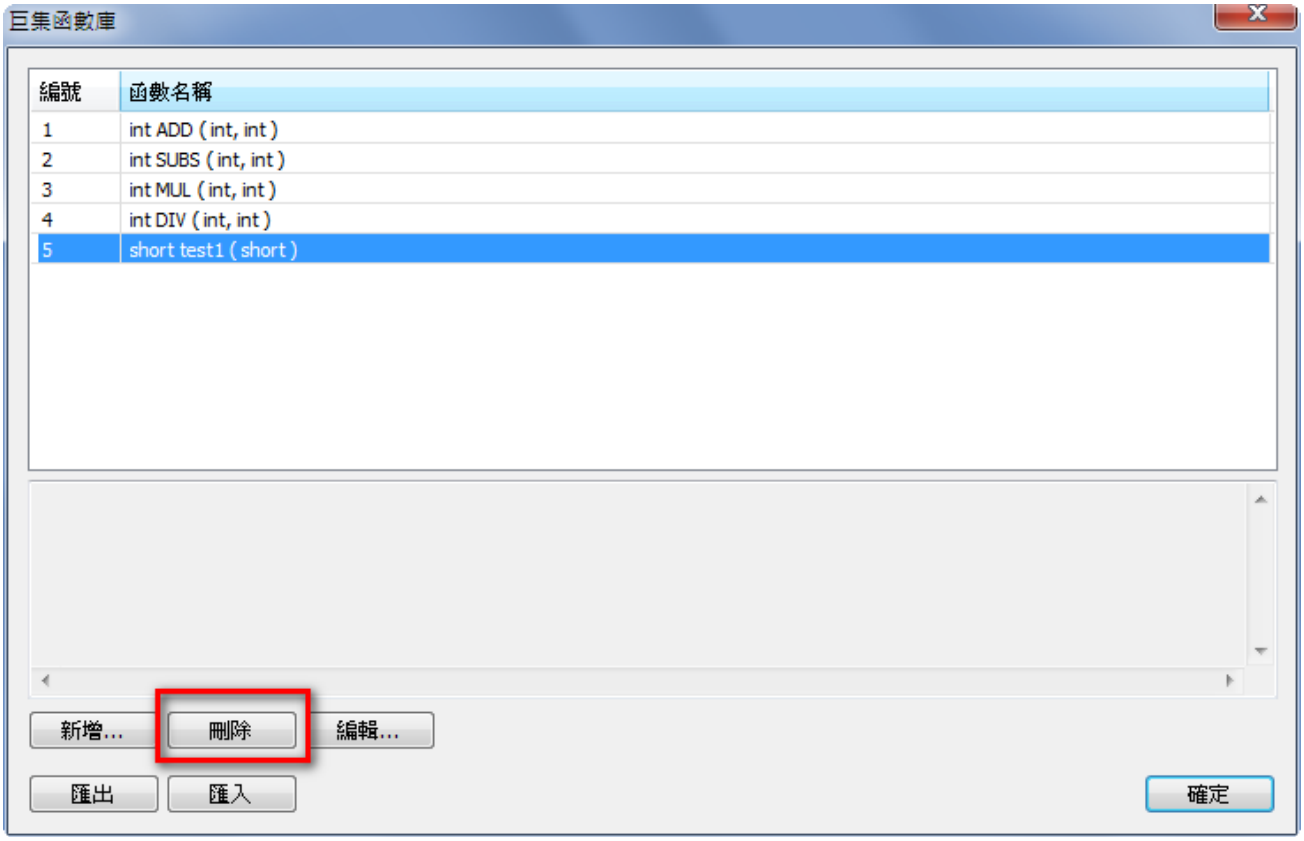


Note

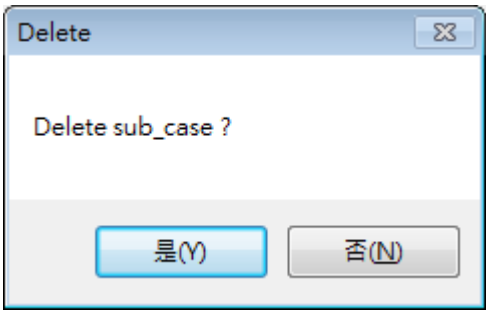
- 函数中可宣告的数据类型总数为 4096 个字节。
- 函数名称必须为英数字符，且不可为数字开头。

删除函数

1. 在函数列表中选定要删除的函数，按下“删除”按钮，即可删除该函数。

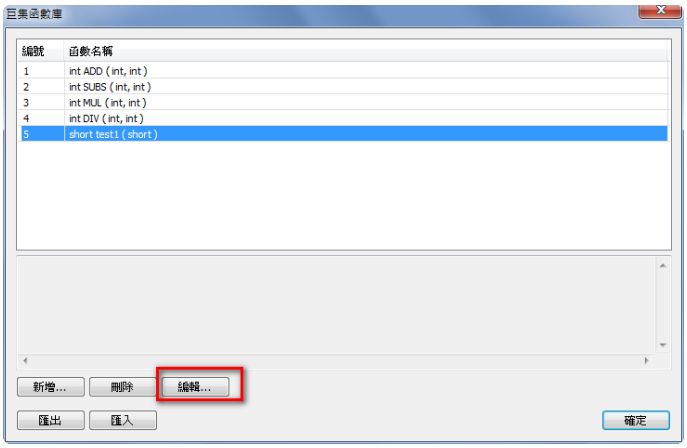


2. 按下“是”确认删除，按下“否”取消删除。按下“是”删除 MAX_SHORT 此函数。

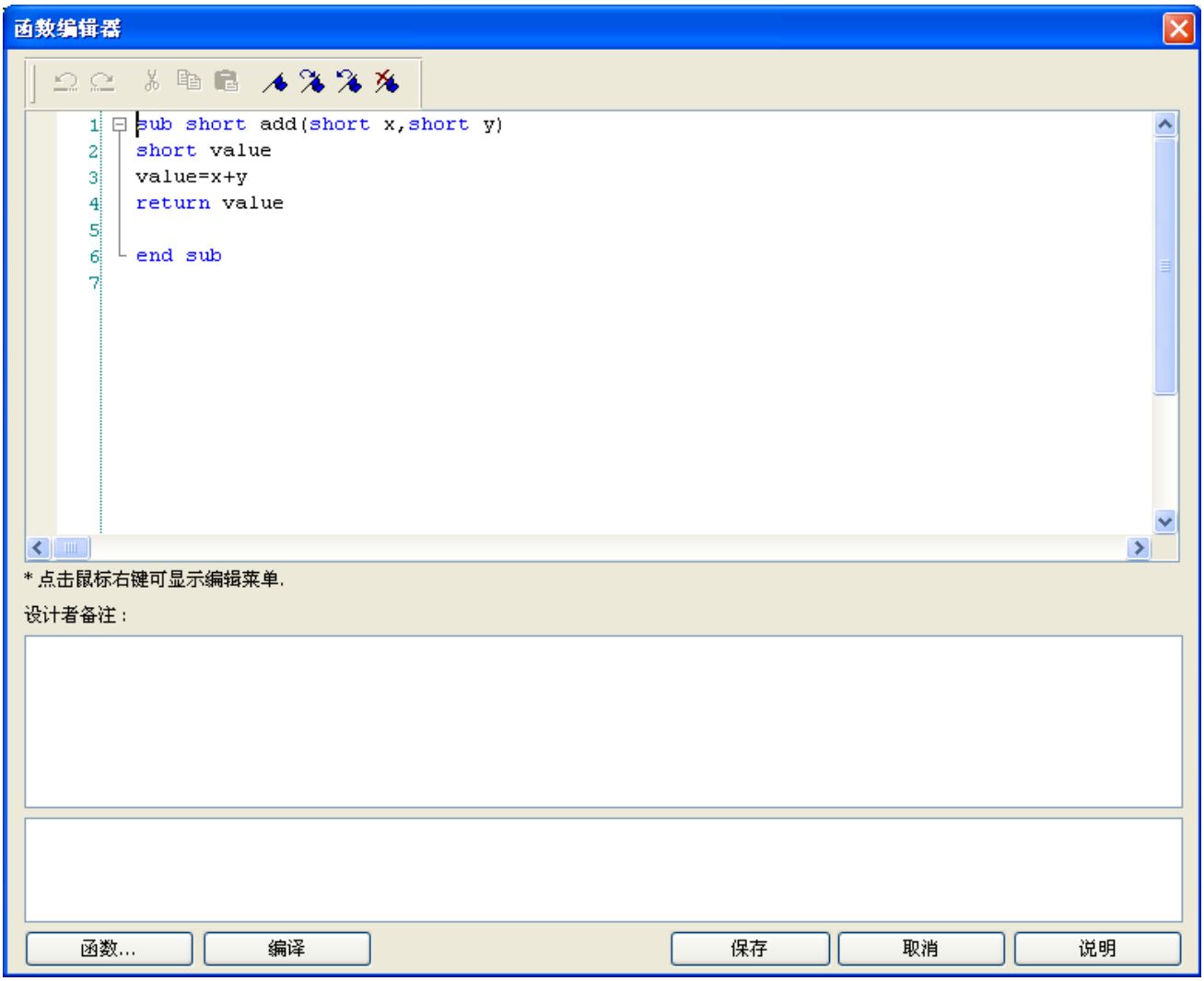


修改函数

1. 用户可以修改函数库中的函数。
2. 选定要修改的函数，按下“编辑”按钮，进入函数编辑器。



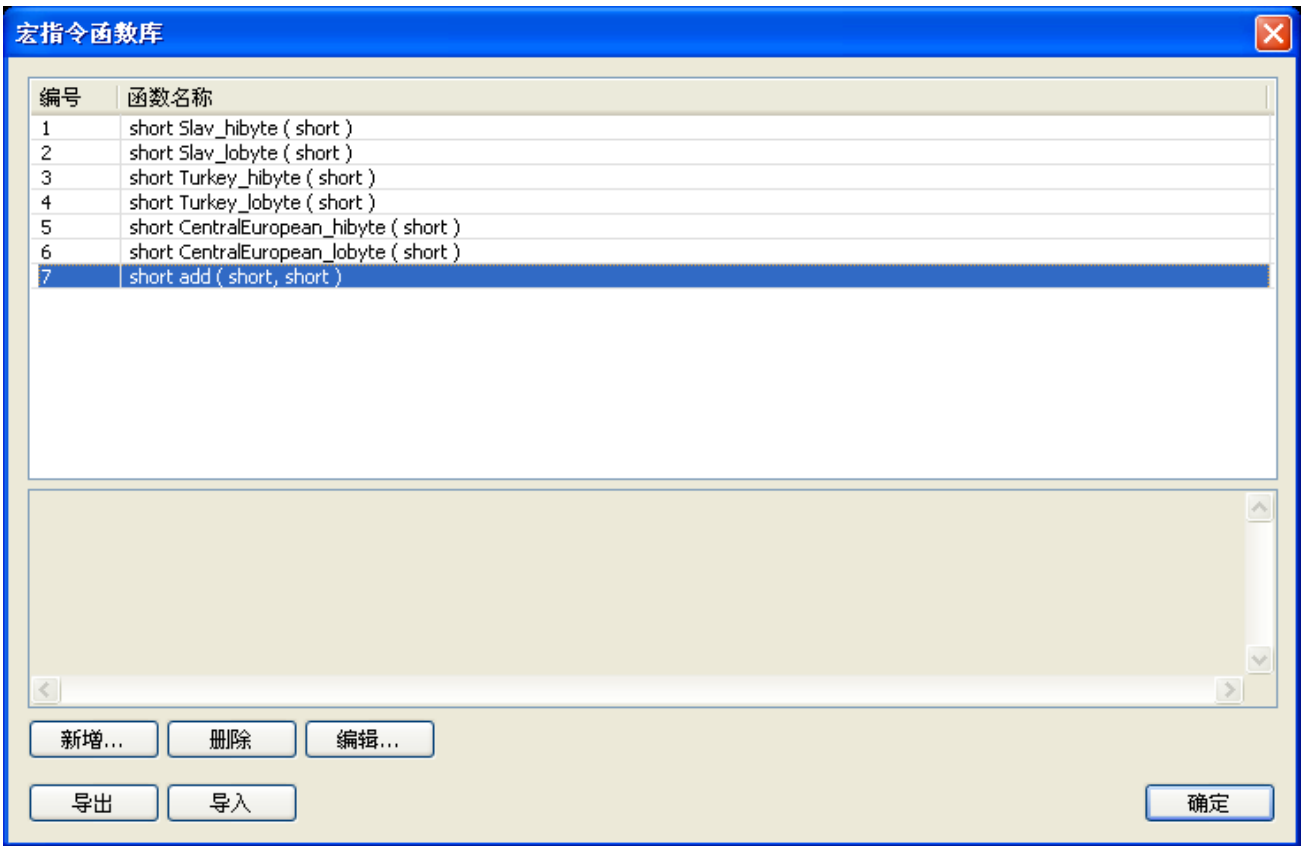
3. 也可直接在该函数上双点鼠标左键，进入函数编辑器。



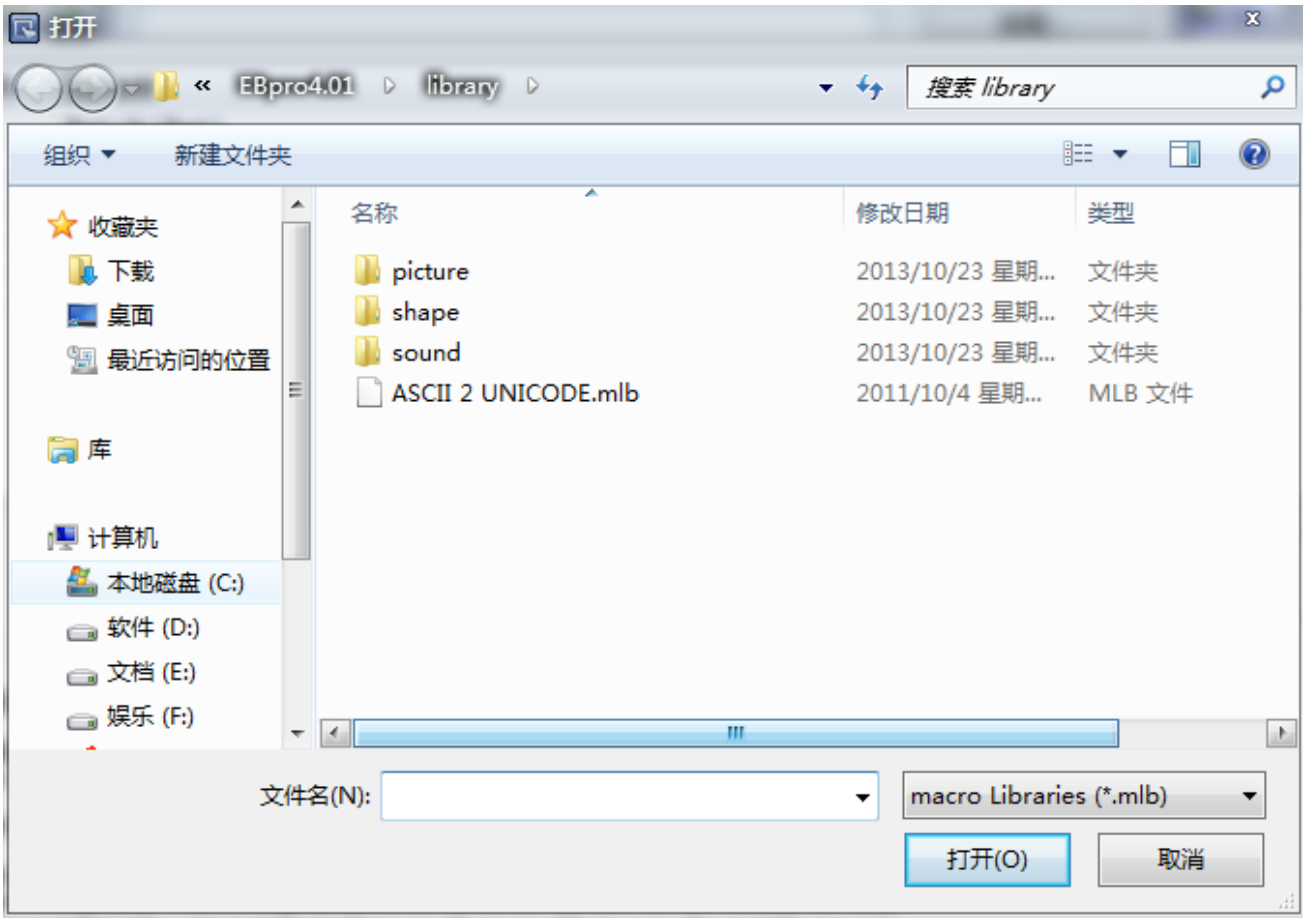
4. 修改完成后，一样要先通过编译，才可以储存离开。

导入函数

1. 用户可以从外部的 .mlb 文件将函数导入。


宏指令函数库
编号
函数名称
1 short Slav_hibyte (short)
2 short Slav_lobyte (short)
3 short Turkey_hibyte (short)
4 short Turkey_lobyte (short)
5 short CentralEuropean_hibyte (short)
6 short CentralEuropean_lobyte (short)
7 short add (short, short)
新增...
删除
编辑...
导出
导入
确定

2. 假设欲加入函数库 math.mlb，此函数库内含有一函数 test1。按下”开启旧文件”按钮。



3. 导入函数时，若文件库中已存在相同名称的函数，会出现下面的弹出窗口。

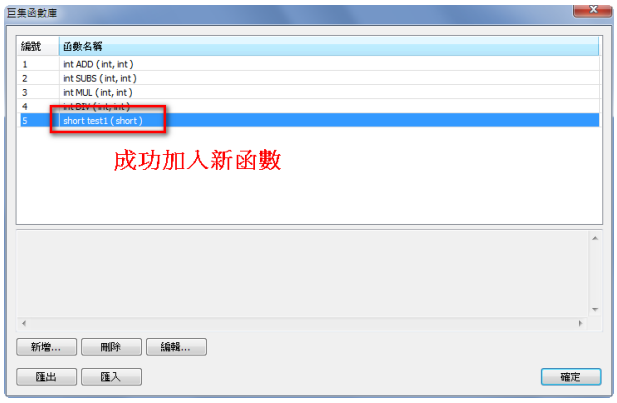
“确定”：用外部导入的函数覆盖函数库中现有的同名函数。

“否”：放弃外部导入此同名函数。

“全部确定”：全部用外部导入的同名函数覆盖所有同名函数。

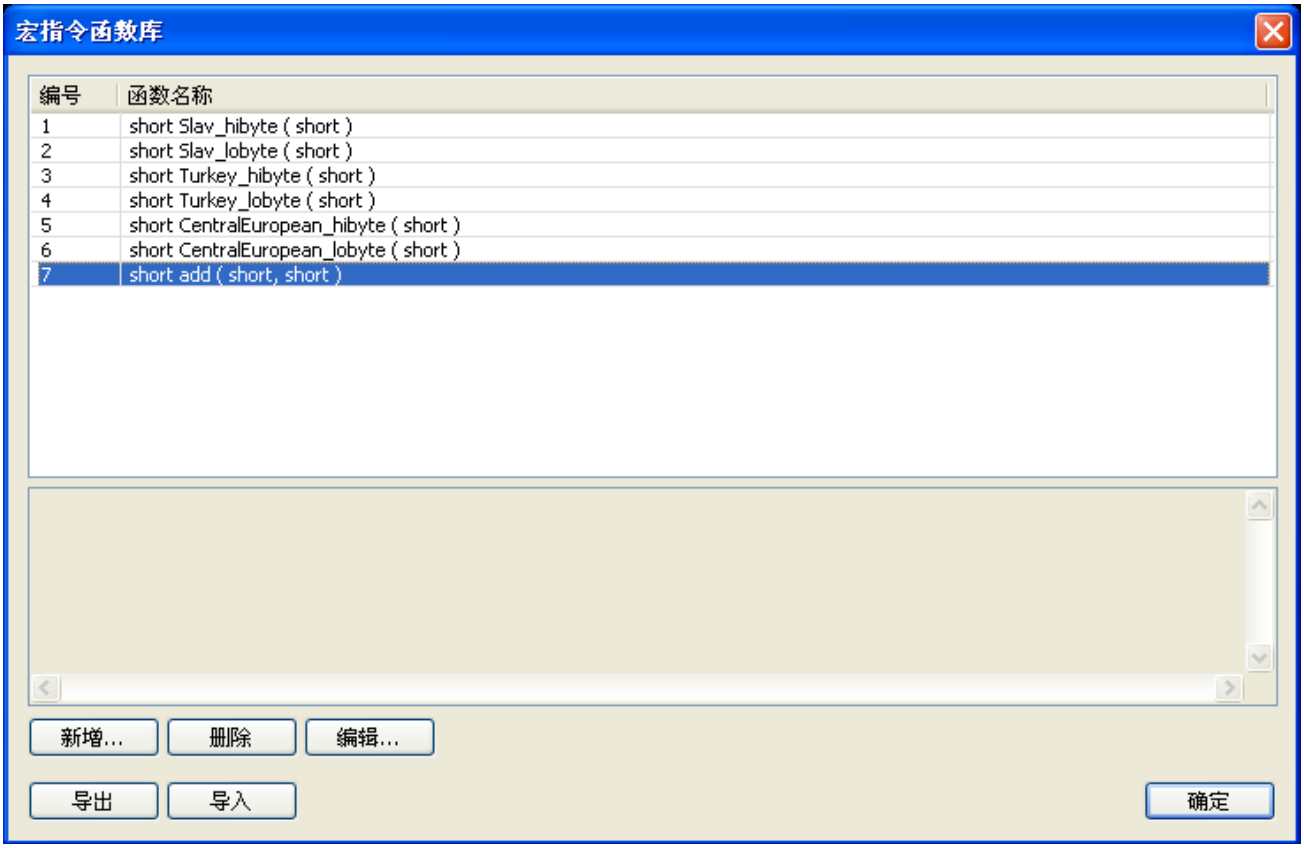
“全否”：全部放弃覆盖导入同名的函数。

4. 完成导入后，导入的函数都已写入到预设函数库中，因此用户可以将 math.mlb 文件删除，而 test1 仍会存在函数库中，即使重新开启触摸屏编辑软件亦然。

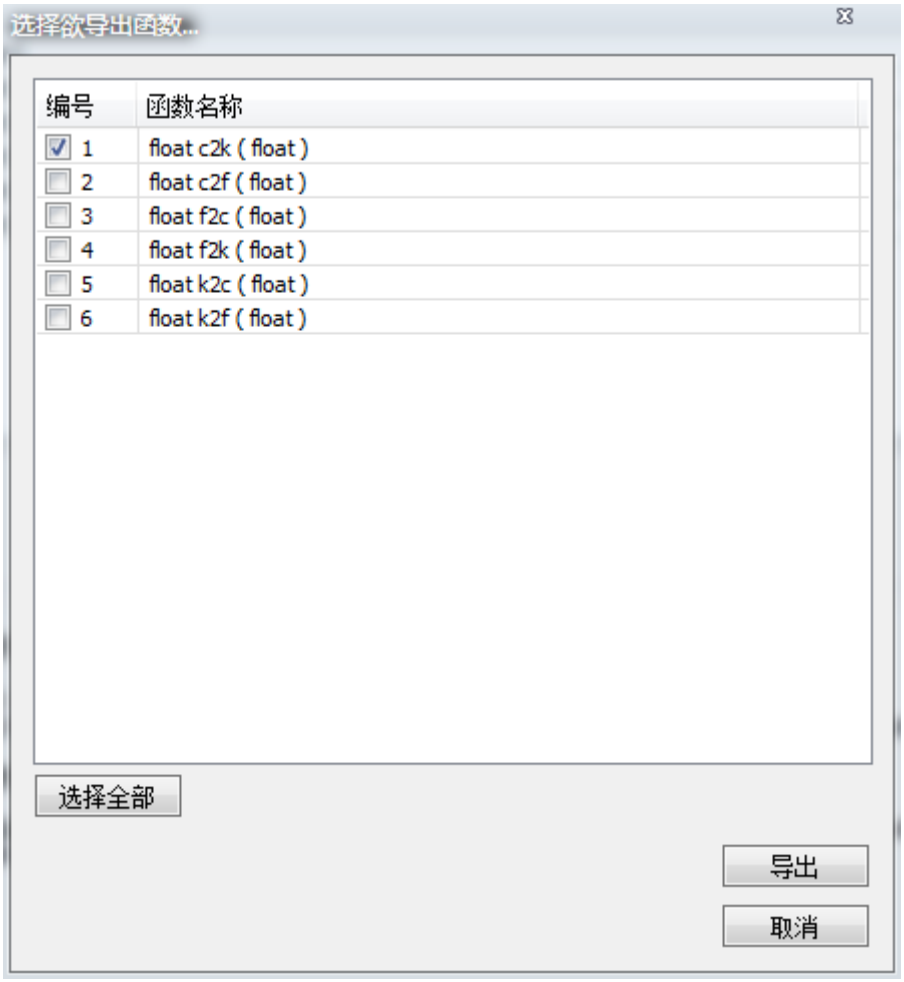


导出函数

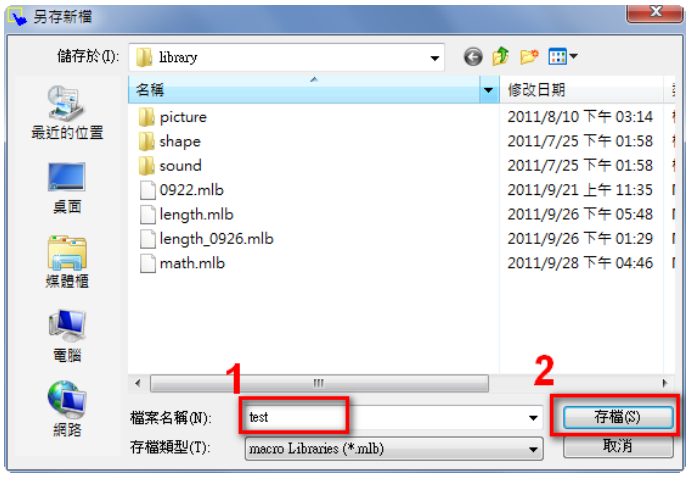
1. 用户可以将函数库中的函数导出到 .mlb 檔。按下“导出”按钮。



2. 选择要导出的函数，然后按“导出”。



3. 导出的目录下出现一个 math.mlb 的函数库文件，其中包含 ADD、SUBS、MUL、DIV 这四支函数。
4. 导出的 .mlb 文件可以携带到别的计算机上，只要用户开启触摸屏编辑软件并完成导入动作后，即可使用此文件内所提供的所有函数。



18.10 使用宏指令时的注意事项

1. 储存局部变量的空间是 4KB，所以各种不同变量类型的最大数组大小为如下：

   char a[4096]
   bool b[4096]
   short c[2048]
   int d[1024]
   float e[1024]

2. 一个 EasyBuider Pro 工程中最多包含 255 个宏指令。
3. 宏指令有可能造成触摸屏当机，可能的原因为：

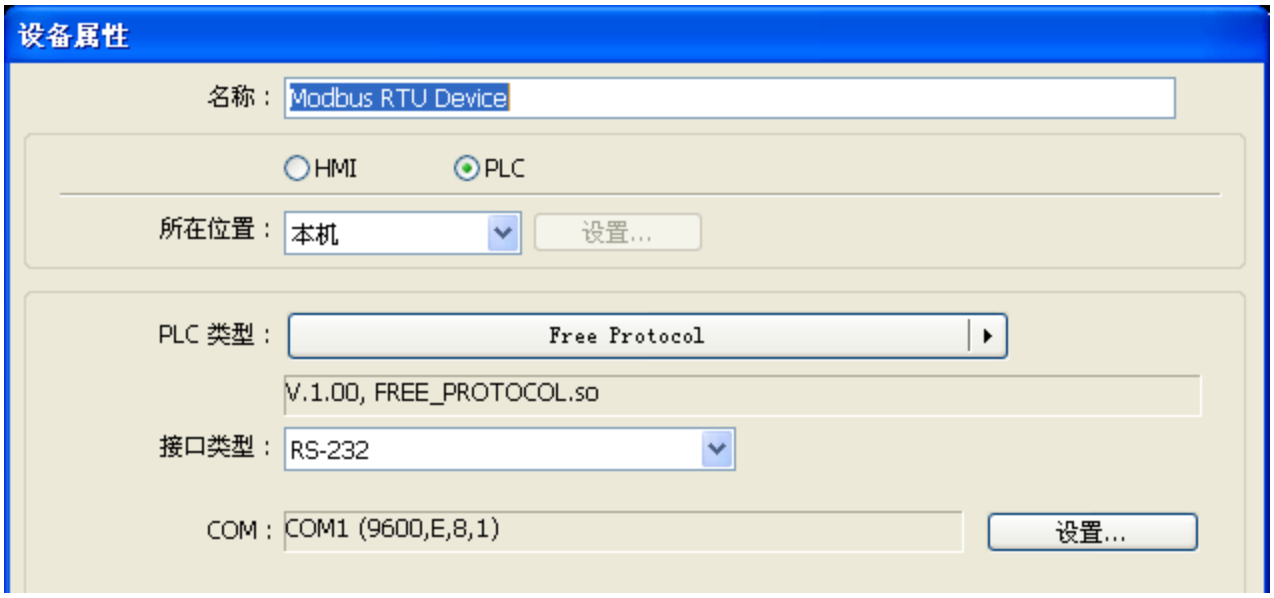
- 宏指令中执行了一个死循环命令
- 数组的大小超过了宏指令的变量容量

4. PLC 的通讯速度可能影响宏指令的执行速度。相对的，使用过多的宏指令，可能会造成与 PLC 的通讯速度变慢。

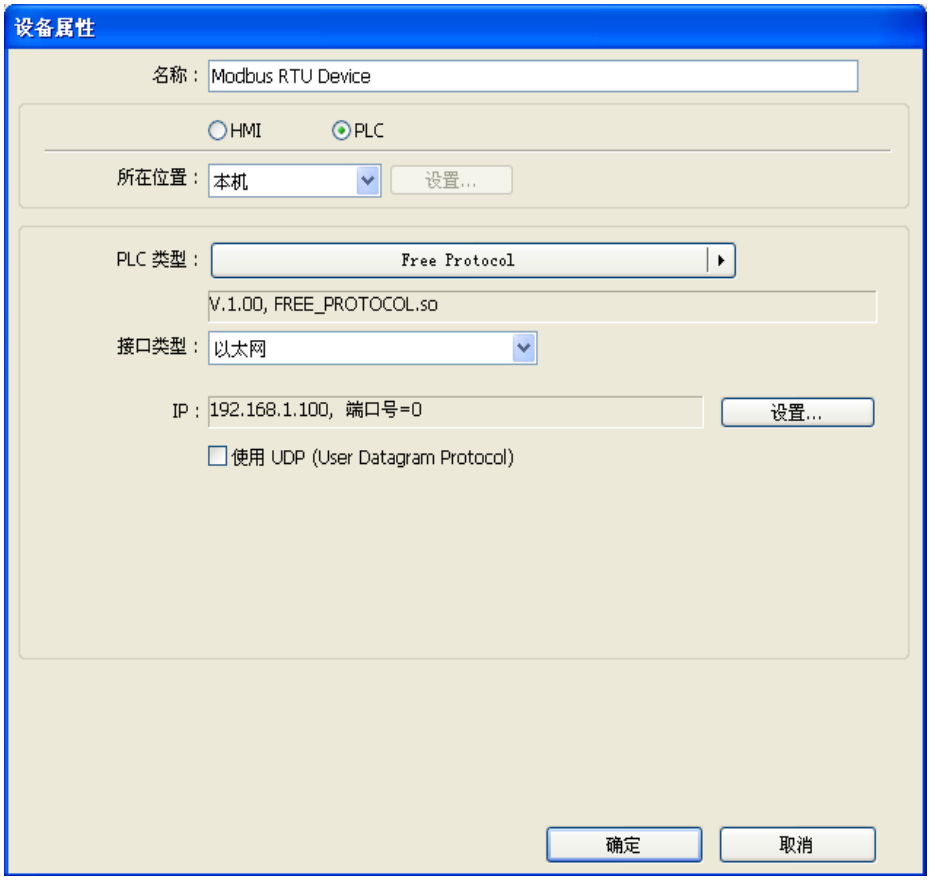
18.11 使用自由协议去控制一个设备

当 EasyBuilder Pro 没有内建与某一个设备通讯的驱动程序时，用户也可以使用宏指令中的 OUTPORT 和 INPORT 函数来实现与该设备的通讯。使用 OUTPORT 和 INPORT 函数发送和接收的数据，必须遵行该设备的通讯协议。下面的范例程序说明了如何使用这两个函数来控制一个 MODBUS RTU 设备。

1. 首先，在系统参数/设备列表中建立一个新的设备。这个新建的 “PLC 类型” 设置为 “Free Protocol”，”PLC 名称” 设置为 “MODBUS RTU Device”，如下图所示。



2. 这里的设备连接使用 RS232，如果连接一个 MODBUS TCP / IP 设备，这个接口类型必须设定为“以太网”，同时必须设定正确的 IP 地址和连接端口号，如下图所示。



假设触摸屏触摸屏界面将要读取设备中的 4x_1 和 4x_2 两个数据寄存器。首先，使用 OUTPORT 函数发送读命令给这个设备，OUTPORT 函数的写法为：

OUTPORT(command[start], device_函数名称, cmd_count)

因为“MODBUS RTU Device”是一个 MODBUS RTU 设备，读数据的命令必须遵行 MODBUS RTU 协议的命令规则。所以必须使用 0x03 这个命令去读取 4x_1 和 4x_2 这两个数据寄存器的数据。下图说明了读命令的格式。(省略了设备的站号和最后的两个 CRC 字节)

Request

Function code	1 Byte	0x03
Starting Address	2 Bytes	0x0000 to 0xFFFF
Quantity of Registers	2 Bytes	1 to 125 (0x7D)

Response

Function code	1 Byte	0x03
Byte count	1 Byte	2 x N*
Register value	N* x 2 Bytes	

*N = Quantity of Registers

Error

Error code	1 Byte	0x83
Exception code	1 Byte	01 or 02 or 03 or 04

根据这个 MODBUS RTU 协议，发送命令的内容如下所示：（总共 8 个字节）

Command[0]: station number (BYTE 0) 站号

Command[1]: function code (BYTE 1) 功能码

Command[2]: high byte of starting address (BYTE 2) 起始地址高位

Command[3]: low byte of starting address (BYTE 3) 起始地址低位

Command[4]: high byte of quantity of registers (BYTE 4) 数据寄存器的高位

Command[5]: low byte of quantity of registers (BYTE 5) 数据寄存器的低位

Command[6]: low byte of 16-bit CRC (BYTE 6) CRC 的低字节

Command[7]: high byte of 16-bit CRC (BYTE 7) CRC 的高字节

所以读数据的命令宏指令程序设计如下：

```
char command[32]
short address, checksum

FILL(command[0], 0, 32)   //   initialize command[0]~command[31] to 0

Command[0] = 0x1   //   station number
Command[1] = 0x3   //   read holding registers (function code   is 0x3)

address =   //   starting address (4x_1) is 0
HIBYTE(address, command[2])
LOBYTE(address, command[3])

read_no = 2   //   the total words of rading is 2 words
HIBYTE(read_no, command[4])
LOBYTE(read_no, command[5])
```

最后，使用 OUTPORT 函数将这个读命令发送给这个 MODBUS RTU 设备。

```
OUTPORT(command[0], “MODBUS RTU Device”, 8)   //   发送命令
```

发送完这个命令后，使用 INPORT 函数读取该 MODBUS RTU 设备返回的命令。根据 MODBUS RTU 协议，这个回复的命令内容为如下 (总共 9 个字节)：

Command[0]: station number (BYTE 0) 站号
Command[1]: function code (BYTE 1) 功能码
Command[2]: byte count (BYTE 2) 位组长度
Command[3]: high byte of 4x_1 (BYTE 3) 第一个数据的高字节
Command[4]: low byte of 4x_1 (BYTE 4) 第一个数据的低字节
Command[5]: high byte of 4x_2 (BYTE 5) 第二个数据的高字节
Command[6]: high byte of 4x_2 (BYTE 6) 第二个数据的低字节
Command[7]: low byte of 16-bit CRC (BYTE 7) CRC 的低字节
Command[8]: high byte of 16-bit CRC (BYTE 8) CRC 的高字节

此时，INPORT 函数的语句如下：

```
INPORT(response[0], “MODBUS RTU Device”, 9, return_value)  //  读取回复命令
```

函数中，实际读取到的位组长度存放在变量 return_value (变量类型为字节) 中。如果 return_value 的数据为 0，则表示使用 INPORT 读取命令失败。

根据 MODBUS RTU 协议，如果命令回复的正确，则 response[1] 必须为 0x03。当读取到正确的命令后，计算出 4x_1 和 4x_2 这两个寄存器的值，并将这两个数据送到触摸屏界面的 LW-100 和 LW-101 寄存器中。

```
If (return_value) >0 and response[1] == 0x3) then
read_data[0] = response[4] + (response[3] << 8) //  计算 4x_1 的资料
read_data[1] = response[6] + (response[5] << 8) //  计算 4x_2 的资料
SetData(read_data[0], “Local HMI”, LW, 100, 2)  //  将资料存至 HMI 上
endif
```

完整的宏指令程序如下：

```
// Read Holding Registers
macro_command main()
char command[32], response[32]
short address, checksum
short read_no, return_value, read_data”2], i
FILL(command[0], 0, 32)//   initialize command[0]~command[31] to 0
FILL(response[0], 0, 32)
Command[0] = 0x1//   station number
Command[1] = 0x3//   read holding registers (function code is 0x3)
address = 0
address =   0//   starting address (4x_1) is 0
HIBYTE(address, command[2])
LOBYTE(address, command[3])
read_no = 2/   the total words of reading is 2 words
HIBYTE(read_no, command[4])
LOBYTE(read_no, command[5])
CRC(command[0], checksum, 6)//   calculate 16-bit CRC
LOBYTE(checksum, command[6])
HIBYTE(checksum, command[7])
OUTPORT(command[0], "MODBUS RTU Device", 8 )//   send request
INPORT(response[0], "MODBUS RTU Device", 9, return_value)//   read response
if (return_value > 0 and response[1] == 0x3) then
   read_data[0] = response[4] + (response[3] << 8)//   4x_1
   read_data[1] = response[6] + (response[5] << 8)//   4x_2

   SetData(read_data[0], "Local HMI", LW, 100, 2)
end if

end macro_command
```

下面的举例说明如何使用自由协议设定 MODBUS RTU 设备中 0x_1 的状态。这个是使用 MODBUS RTU 协议中的“写单个寄存器”的功能码“0x05”来实现的。

Request

Function code	1 Byte	**0x05**
Output Address	2 Bytes	0x0000 to 0xFFFF
Output Value	2 Bytes	0x0000 or 0xFF00

Response

Function code	1 Byte	**0x05**
Output Address	2 Bytes	0x0000 to 0xFFFF
Output Value	2 Bytes	0x0000 or 0xFF00

Error

Error code	1 Byte	**0x85**
Exception code	1 Byte	01 or 02 or 03 or 04

完整的宏指令程序如下：

```
//   Write Single Coil (ON)
macro_command main()

char command[32], response[32]
short address, checksum
short i, return_value

FILL(command[0], 0, 32)//   initialize command[0]~ command[31] to 0
FILL(response[0], 0, 32)

Command[0] = 0x1//   station number
Command[1] = 0x5//   function code : write single coil

address = 0
HIBYTE(address, command[2])
LOBYTE(address, command[3])

Command[4] = 0xff//   force 0x_1 on
Command[5] = 0

CRC(command[0], checksum, 6)

LOBYTE(checksum, command[6])
HIBYTE(checksum, command[7])

OUTPORT(command[0], "MODBUS RTU Device", 8)//   send request
INPORT(response[0], "MODBUS RTU Device", 8, return_value)//   read response

end macro_command
```

18.12 编译错误提示信息

- 错误信息格式：

error C# : error 描述

(# 是错误信息编号)

举例：error C37：undeclared identifier：i

当编译后提示有错误信息时，这个错误的描述内容可以参考错误信息编号。

- Error 描述

(C1) 语法 error：“标识符”

出现这个信息时，有许多种可能：

举例:

```
macro_command main()
char i, 123xyz    // 不支持这个变量类型
end macro_command
```

(C2) 'identifier' used without having been initialized (使用的标识符没有初始化)

宏指令必须定义声明的数组变量的大小

举例:

```
macro_command main()
char i
int g[i]    // i 必须为一个数值常数
end macro_command
```

(C3) redefinition error : 'identifier' (标识符被重复定义)

函数名称和变量名称在有效范围内，必须是唯一的。

举例:

```
macro_command main()
int g[10]，g    // 重复定义错误
end macro_command
```

(C4) function 函数名称 error : 'identifier' (函数名称定义错误)

保留的关键词和常数，不能被定义为函数名称

举例：

```
sub int if()  // 函数名称定义错误
```

(C5) parentheses have not come in pairs (圆括号没有成对的出现)

语句中缺少了 "(" or ")"

举例：

```
macro_command main  )  // 缺少了 "("
```

(C6) illegal expression without matching 'if' (if 语句中没有合法的表达式)

也就是在 if 语句中缺少表达式

(C7) illegal expression (no 'then') without matching 'if' (if 语句中缺少了 then)

也就是 if 和 then 没有成对

(C8) illegal expression (no 'end if') (if 语句中缺少了 enf if)

缺少了 "end if"

(C9) illegal 'end if' without matching 'if' (end if 语句前缺少了 if)

End if 语句前缺少了 if 语句

(C10) illegal 'else' (不合法的 else 语句)

这个 "if" 语句的格式为：

if "逻辑表达式" then

" else "if "逻辑表达式" then " "

end if

任何与以上格式不符合的语句，在编译时就会错误。

(C17) illegal expression (no 'for') without matching 'next' (没有与 next 相配的 for 语句)

"for" 语句错误：在 "next" 前，缺少了 "for" 语句

(C18) illegal variable type (not integer or char) (不合法的变量类型)

变量类型定义错误，此处应为整数类型或字符类型变量

(C19) variable type error

缺少赋值语句

(C20) must be keyword 'to' or 'down' (缺少了关键词 "to" 或者 "down")

缺少了关键词 "to" 或者 "down"

(C21) illegal expression (no 'next') (非法的表达式，缺少了 "next")

"for" 语句的格式为：

for "变量" = "初始值" to "结束值" "step"

next "变数"

任何与上述格式不符合的语句，编译时会错误。

(C22) 'wend' statement contains no 'while'

循环缺少“while”关键词，”wend”前面应有“while”关键词

(C23) illegal expression without matching 'wend'

缺少“wend”关键词

“while”语句的格式为：

while “逻辑表达式”

wend

任何不符合上述语法的，在编译时会错误。

(C24) 语法 error : 'break'

不合法的 ”break”语句。break 语句只能在 for 循环、while 循环选择结构中使用。

(C25) 语法 error : 'continue'

不合法的“continue”语句。continue 语句只会在“for”或者“while”语句中出现。

(C26) 语法 error

表达式不正确

(C27) 语法 error

表达式中缺少了一个运算符号可能会造成这个编译错误信息。

举例 ：

```
macro_command main( )
int a, b
for a = 0 to 2
b = 4 + xyz     // 不合法之处：xyz 变量没有被定义
next a
end macro_command
```

(C28) must be 'macro_command'

此处应该为“macro_command”

(C29) must be key word 'sub'

子函数的定义格式为：

sub "data type" function_函数名称 (...)

.....

end sub

举例:

sub int pow (int exp)

.....

end sub

任何不符合上述语法结构的，在编译时会错误。

(C30) number of parameters is incorrect

参数个数不对。

(C31) parameter type is incorrect

参数数据类型不相配。调用函数时，参数必须在数据类型、个数上一一对应才能通过编译，否则编译时将出现此项错误讯息。

(C32) variable is incorrect

变量类型不正确。当变量被当成参数传递给一个函数时，变量的数据类型应与函数所宣告的类型相同，否则将出现此项错误讯息。

(C33) function 函数名称 : undeclared function

没有定义的函数名称

(C34) expected constant expression

不合法的数组下标表达形式

(C35) invalid array declaration

不合法的数组定义

(C36) array index error

不合法的数组下标

(C37) undeclared identifier : i 'identifier'

使用没有定义的变量。只能使用已经定义的变量和函数，否则编译时将出现此项错误讯息。

(C38) un-supported PLC data address

通讯函数 GetData(…)、SetData(…)的参数中有包含 PLC 地址类型信息，当 PLC 地址类型不是此种 PLC 支持的地址类型时，编译时将出现此项错误信息。

(C39) 'idenifier' must be integer, char or constant

数组的格式为：

声明： array_函数名称"constant" (constant is the size of the array)

使用： array_函数名称"integer, character or constant"

任何不符合上述规则的数组表达式，编译时将会错误

(C40) execution 语法 should not exist before variable declaration or constant definition

变量定义语句的前面不能有执行语句

举例 ：

```
macro_command main(  )
int a, b
for a = 0 To 2
b = 4 + a
int h , k   // 定义变量语句在此处是错误的，在一个函数内定义变量语句的前面
不能有执行语句，例如 b = 4 + a
next a
end macro_command
```

(C41) float variables cannot be contained in shift calculation

移位运算中，操作数不能为浮点数。

(C42) function must return a value

函数应有返回值

(C43) function should not return a value

函数不应有返回值

(C44) float variables cannot be contained in calculation

运算中不能有 float 型数据

(C45) PLC address error

PLC 地址错误

(C46) array size overflow (max. 4k)

一维数组的大小超过 4k

(C47) macro command entry function is not only one

宏指令程序入口只能有一个

(C48) macro command entry function must be only one

宏指令入口函数不是唯一。宏指令的入口函数只能有一个，形式为：

macro_command function_函数名称( )

end macro_command

(C49) an extended addressee's station number must be between 0 and 255

在宏指令中，扩展地址内的站号大小只能从 0 到 255

举例 ：

SetData(bits[0] , "PLC 1", LB , 300#123, 100)

// illegal : 300#123 意思是站号为 300，但是最大值是 255

(C50) an invalid PLC 函数名称

在宏指令中，PLC 的名称并未定义在系统参数的设备列表中

(C51) macro command do not control a remote device

宏指令只能控制本机连接的设备

举例 ：

SetData(bits[0] , "PLC 1" , LB , 300#123, 100)

"PLC1"连接在远程的触摸屏上，所以它不能被执行。

18.13 宏指令范例程序

- for 循环，各种表达式 (算术，移位元，逻辑，关系表达式)

```
macro_command main()
  int a[10], b[10], i

  b[0] = (400 + 400 << 2) / 401
  b[1] = 22 *2 - 30 % 7
  b[2] = 111 >> 2
  b[3] = 403 > 9 + 3 >= 9 + 3 < 4 + 3 <= 8 + 8 == 8
  b[4] = not 8 + 1 and 2 + 1 or 0 + 1 xor 2
  b[5] = 405 and 3 and not 0
  b[6] = 8 & 4 + 4 & 4 + 8 | 4 + 8 ^ 4
  b[7] = 6 - (~4)
  b[8] = 0x11
  b[9] = 409

  for i = 0 to 4 step 1
    if (a[0] == 400) then
      GetData(a[0],"Device 1", 4x, 0,9)
      GetData(b[0],"Device 1", 4x, 11,10)
    end If
  next i
end macro_command
```

- while, if, break 语句

```
macro_command main()
  int b[10], i
  i = 5
  while i == 5 - 20 % 3
    GetData(b[1], "Device 1", 4x, 11, 1)

    if b[1] == 100 then
      break
    end if
  wend
end macro_command
```

- 全局变量和子函数调用

```
char g
sub int fun(int j, int k)
    int y

    SetData(j, “Local HMI”, LB, 14, 1)
    GetData(y, “Local HMI”, LB, 15, 1)
    g = y

    return y
end Sub

macro_command main()
    int a, b, i

    a = 2
    b = 3
    i = fun(a, b)
    SetData(i, “Local HMI”, LB, 16, 1)
end macro_command
```

- if 结构语句

```
macro_command main()
    int k[10], j

    for j = 0 to 10
      k[j] = j
    next j

    if k[0] == 0 then
      SetData(k[1], “Device 1”, 4x, 0, 1)
    end if

     if k[0] == 0 then
      SetData(k[1], “Device 1”, 4x, 0, 1)
    else
      SetData(k[2], “Device 1”, 4x, 0, 1)
    end if
```

```
    if k[0] == 0 then
      SetData(k[1], “Device 1”, 4x, 1, 1)
    else if k[2] == 1 then
      SetData(k[3], “Device 1”, 4x, 2, 1)
    end If

    if k[0] == 0 then
      SetData(k[1], “Device 1”, 4x, 3, 1)
    else if k[2] == 2 then
      SetData(k[3], “Device 1”, 4x, 4, 1)
    else
      SetData(k[4], “Device 1”, 4x, 5, 1)
    end If
end macro_command
```

- while 和 wend 结构语句

```
macro_command main()
      char i = 0
      int a[13], b[14], c = 4848

      b[0] = 13

      while b[0]
        a[i] = 20 + i * 10

        if a[i] == 120 then
          c =200
          break
      end if

      i = i + 1
    wend

    SetData(c, “Device 1”, 4x, 2, 1)
end macro_command
```

- break 和 continue 语句结构

```
macro_command main()
    char i = 0
    int a[13], b[14], c = 4848
    b[0] = 13

    while b[0]
        a[i] = 20 + i * 10

        if a[i] == 120 then
        c =200
          i = i + 1
          continue
        end if
        i = i + 1

      if c == 200 then
      SetData(c, “Device 1”, 4x, 2, 1)
      break
      end if
    wend
end macro_command
```

- 数组结构

```
macro_command main()
    int a[25], b[25], i

    b[0] = 13

    for i = 0 to b”0” step 1
      a[i] = 20 + i * 10
    next i

    SetData(a[0], “Device 1”, 4x, 0, 13)
end macro_command
```

18.14 宏指令 TRACE 函数

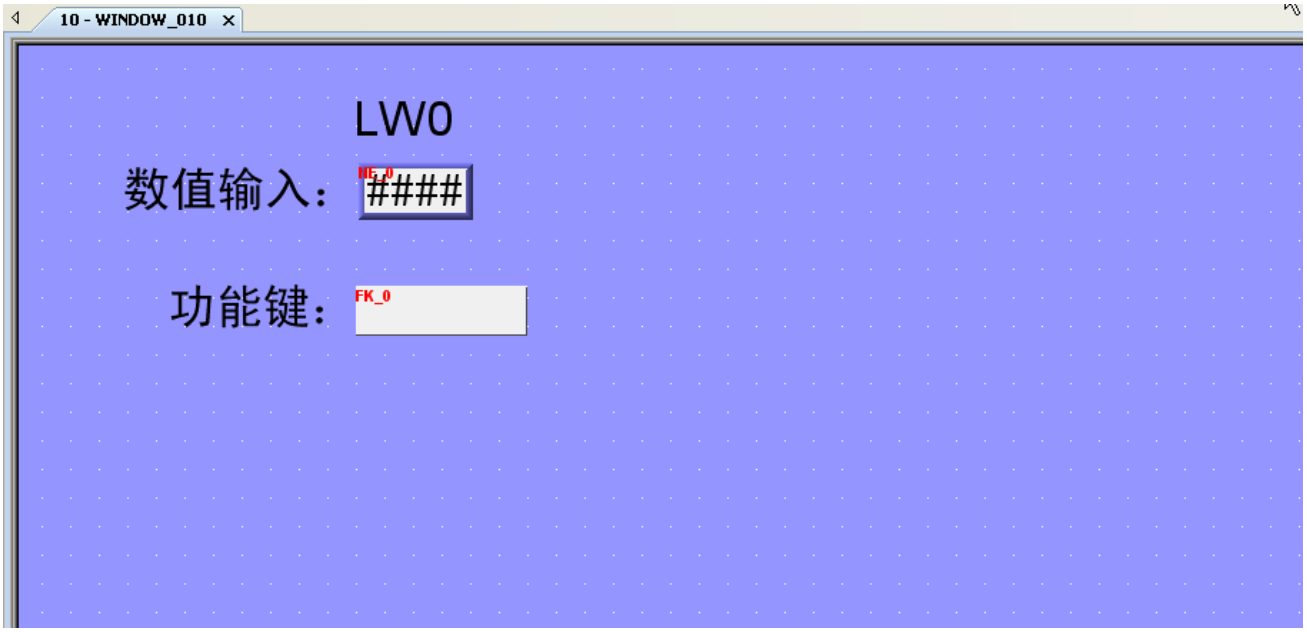
宏指令的 TRACE 函数，搭配使用 EasyDiagnoser，可用来检视所使用变量目前的内容。

下面示范如何在宏指令中利用 TRACE 命令。

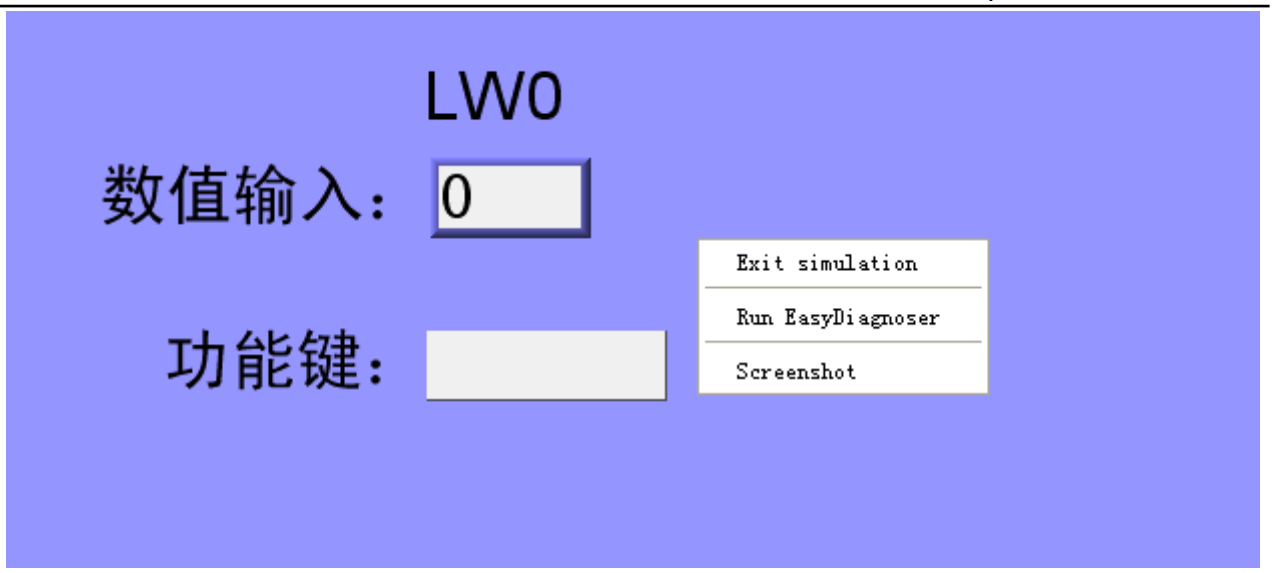
1. 首先，请在工程文件中新增 macro_0，并在 macro_0 的内容中加入 TRACE("LW = %d", a)，"%d" 表示使用 10 进制显示 LW 目前的数值。macro_0 的内容如下：

```
1
2   macro_command main()
3
4   short a
5   GetData(a, "Local HMI", LW, 0, 1)
6   a=a+1
7   SetData(a, "Local HMI", LW, 0, 1)
8   TRACE ("LW0 = %d" , a)
9
10  end macro_command
```

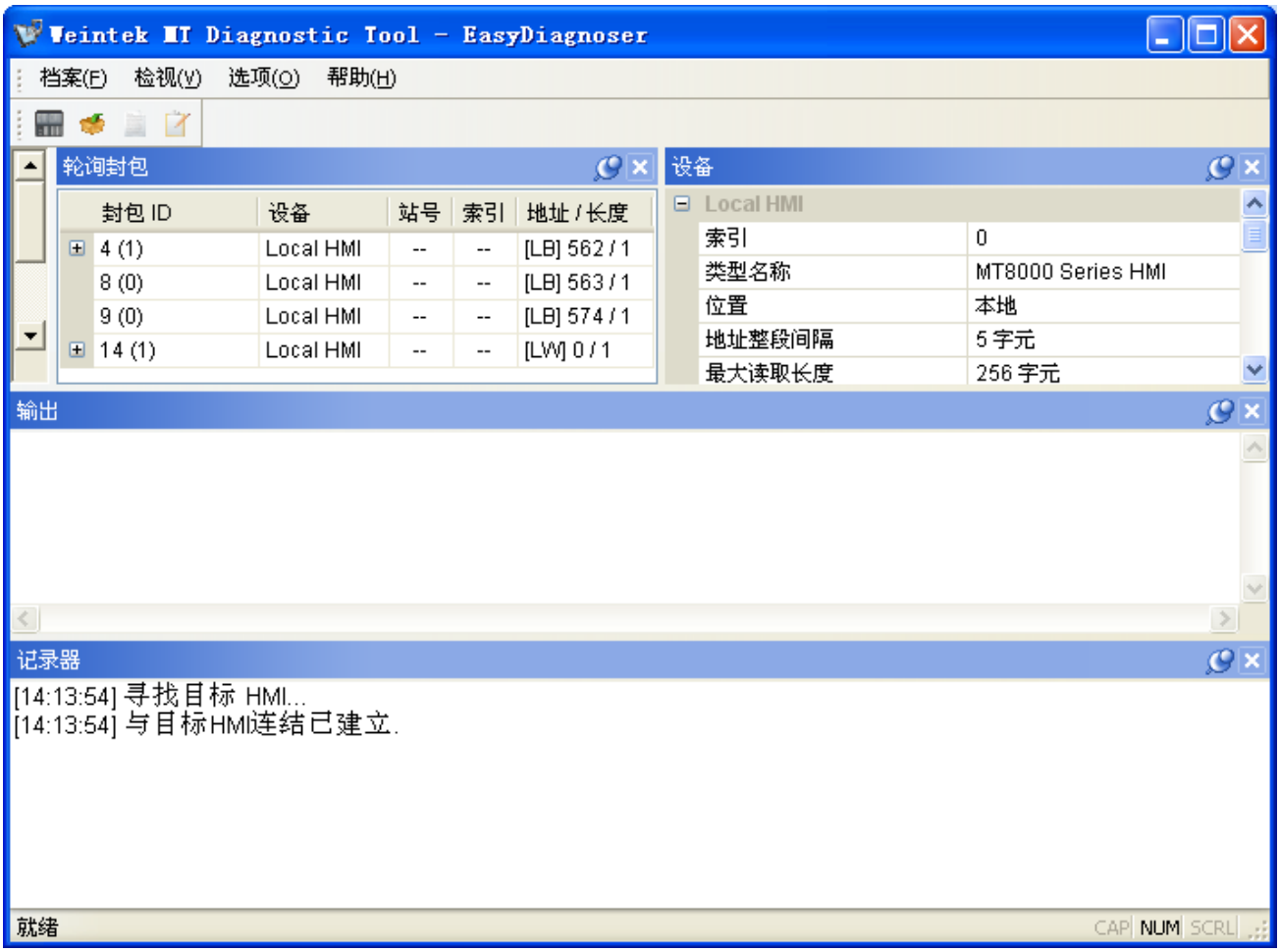
2. 接着在工程文件的第 10 页分别加上“数值显示”与“功能键”元件，”功能键”元件用来执行 macro_0。



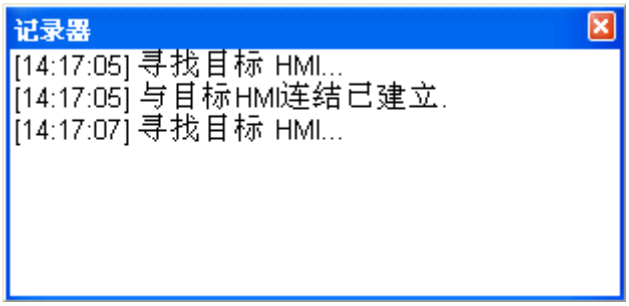
3. 最后编译已完成的工程文件并执行离线或在线模拟。
4. 在 PC 上进行仿真功能时，点击鼠标右键后选择选单上的 "Run EasyDiagnoser"。



5. 此时即会出现 EasyDiagnoser 的画面，”记录器”窗口用来显示 EasyDiagnoser 是否可以连接上需要监视的 HMI，”输出”窗口用来显示 TRACE 的执行结果，下图表示 EasyDiagnoser 已成功连接上 HMI。

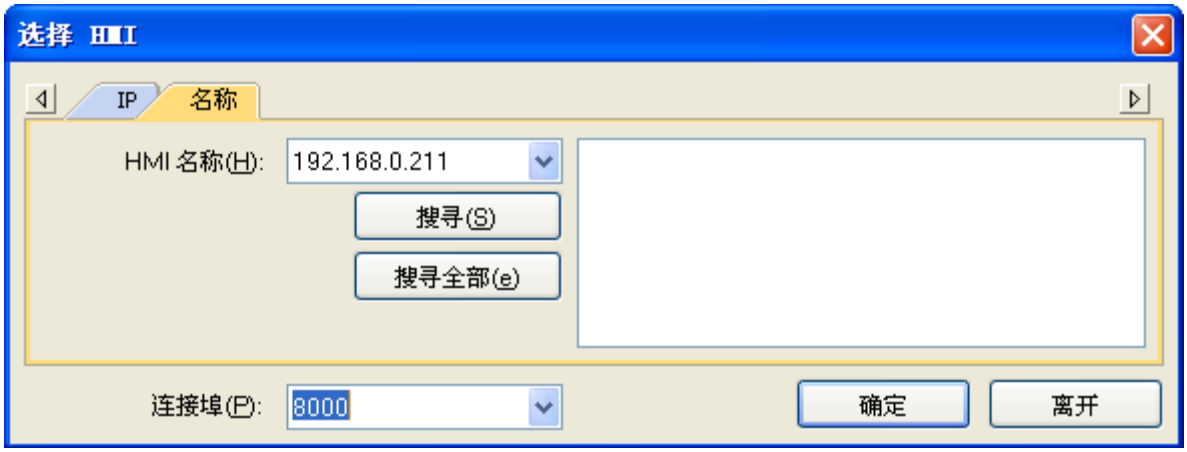


若未成功连接上 HMI，”记录器”的窗口会显示下面的内容：



6. 未联机成功的可能原因是电脑 未成功执行仿真功能；另一个原因是在电脑 执行仿真功能的工程文件所使用的“连接埠”不正确 (可能已被系统占用)，此时请更改工程文件的“连接埠”(请参考下图)，再次编译重新执行仿真功能即可。

7. 开启 EasyDiagnoser 时，应设定与工程文件相同的连接端口。



从所设定的“连接埠”算起连续 3 个 port 将保留给触摸屏做通讯用。例如，上图中设定“连接埠”为 8005，则 8005、8006、8007 三个 port 将被保留。因此在电脑 上模拟时，应确定这些被保留的 port 未被其它程序所占用。

TRACE 命令语法如下：

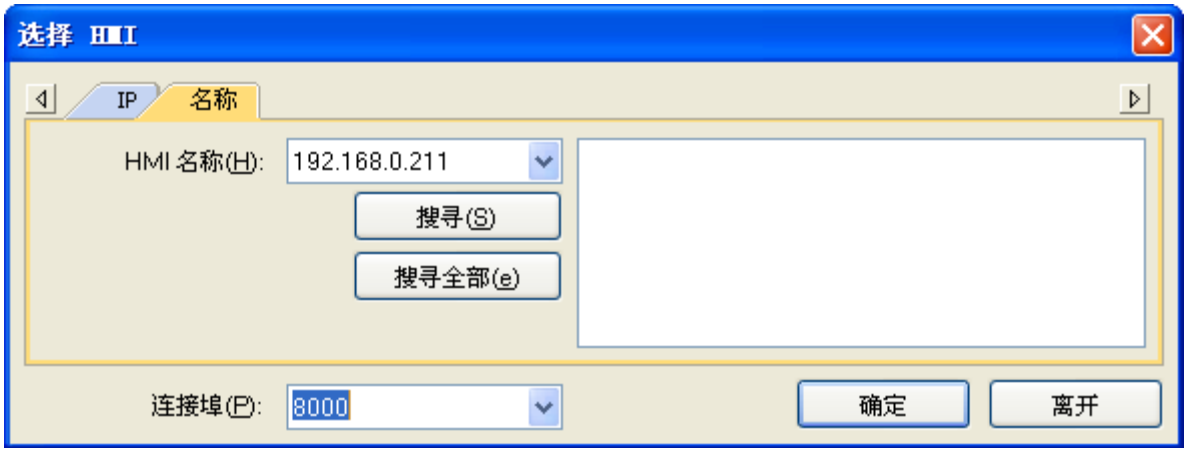
函数名称	TRACE
语法	TRACE(format, argument)
描述	一个执行中的宏指令可以使用此函数，监视变量内容的变化，并打印字符串，以协助除错。用户应开启 EasyDiagnoser 观看此函数的输出结果。 当 TRACE 函数抓取一个 % 开头的特殊字符，将同时从 argument 抓取一个参数

	做格式化后输出。 format 代表打印格式，支持 % 开头的特殊字符。特殊字符格式如下，其中方括号内的字段为可选，粗体字字段为必需： %[flags] [width] [.precision] type 每个字段的意义如下所述： flags (可选)： - + width (可选)： 十进制正整数，指定应预留的字符宽度，不足部份补空格符。 precision (可选)： 十进制正整数，指定精确度，以及输出字符数。 type： C 或 c ： 以字符方式输出 d ： 以 signed 十进制整数输出 i ： 以 signed 十进制整数输出 o ： 以 unsigned 八进位整数输出 u ： 以 unsigned 十进制整数输出 X 或 x ： 以 unsigned 十六进制整数输出 E 或 e ： 以科学表示法输出 f ： 以单倍精确度浮点数输出 *format* 字符串最长支持 256 个字符，多出的字符将被忽略。 *argument* 部份可写可不写。但一个特殊字符应搭配一个变量。

举例	macro_command main()
	char c1 = ‘a’
	short s1 = 32767
	float f1 = 1.234567
	
	TRACE(“The results are”) // 输出：The results are
	TRACE(“c1 = %c, s1 = %d, f1 = %f”, c1, s1, f1)
	// 输出： c1 = a, s1 = 32767, f1 = 1.234567
	
	end macro_command

8. 新增 LB-9059 -关闭宏指令 TRACE 功能 (当状态为 ON)。当设定为 ON 时，TRACE 将不会把数据输出到 EasyDiagnoser。

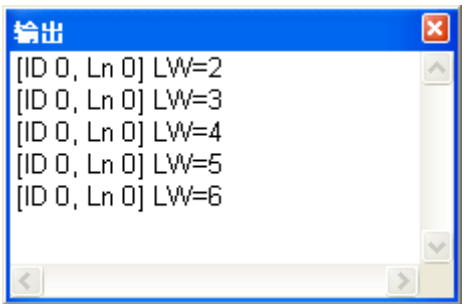
9. 也可以直接利用 Utility Manager 执行 EasyDiagnoser.exe，Utility Manager 将会显示网络上目前存在的 HMI，此时只要选择要监看通讯状态的 HMI 即可。请注意 Project Port 的部份应设定与工程文件所使用的“连接埠”相同。



10. 将工程文件下载到触摸屏实际操作。当 EasyDiagnoser 无法连接上需要监视的触摸屏时，一般可能的原因是触摸屏未上电。或是“连接埠”不正确，可能发生 EasyDiagnoser 不断连上触摸屏又

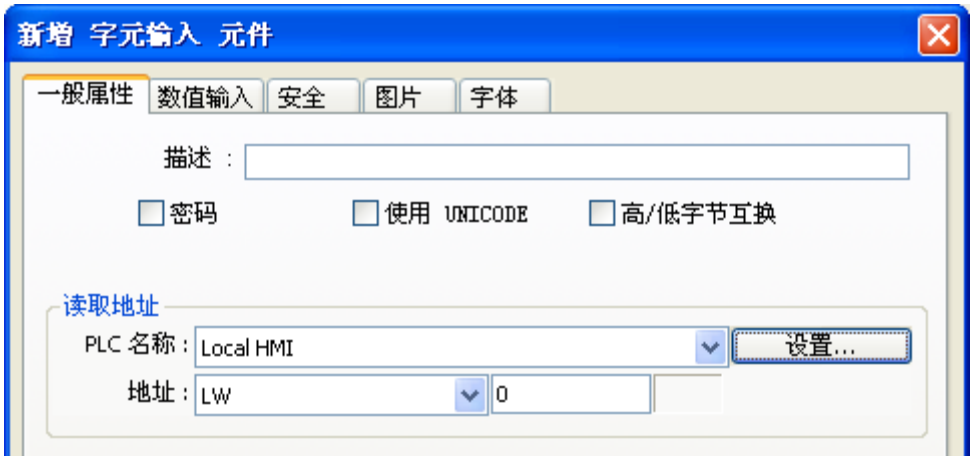
断线的情况。请确定 EasyDiagnoser 应设定与工程文件相同的 Port No.，更改方式如前面所述。

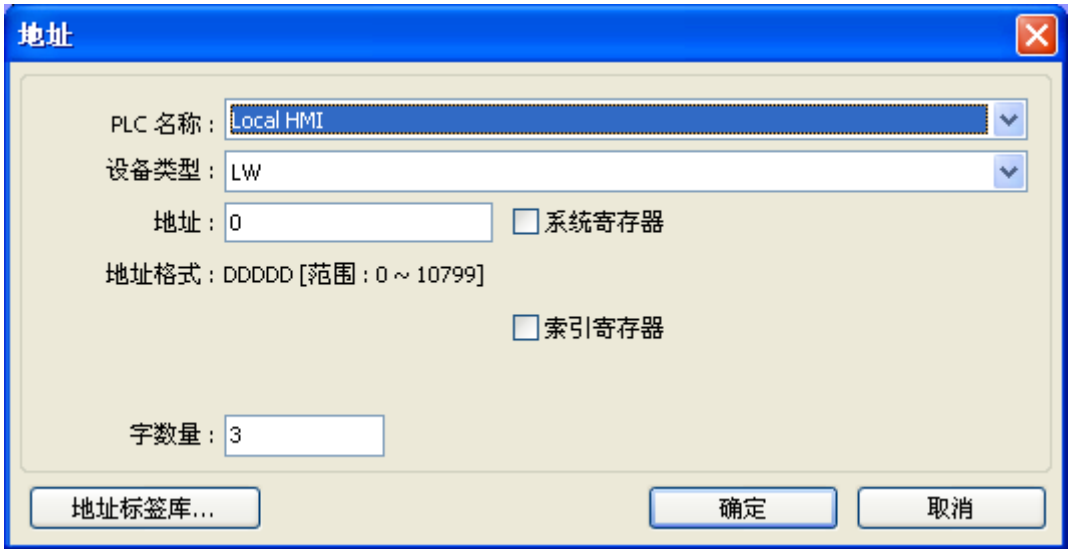
11. 当 EasyDiagnoser 成功与触摸屏连结后，只要执行 macro_0，即可发现“输出”窗口显示目前 TRACE 的执行结果。



18.15 字符串处理函式使用方法

宏指令提供字符串处理函式，令用户可以很方便的对字符串进行各种操作。所谓字符串，是由一连串的 ASCII 字符组成，每一个 ASCII 字符占用 1 字节 (byte)，在 16 位寄存器中是以低字节优先方式储存。举例来说，我们利用一个“文字输入”元件在 LW-0 ~ LW-2 的位置，写入一条字符串 ”abcdef”，设定如下：





在此“文字输入”元件输入 ”abcdef”：

此字符串在 LW-0 ~ LW-2 中的存放方式如下图所示 (LB 代表低字节，HB 代表高字节)：

	HB	LB
LW0	'B'	'A'
LW1	'D'	'C'
LW2	'F'	'E'
LW3		
LW4		
LW5		

由于“文字输入”元件显示的数据长度单位为 word，而每个 ASCII 字符长度为一个 byte，所以每次最少会显示两个字符，此部份在“文字输入”元件的章节有详细的说明。因此上面的例子中，设定“文字输入”元件的字符数量为 3，表示最多可输入 / 显示 6 个 ASCII 字符。

目前提供的字符串处理函式，其功能整理如下表：

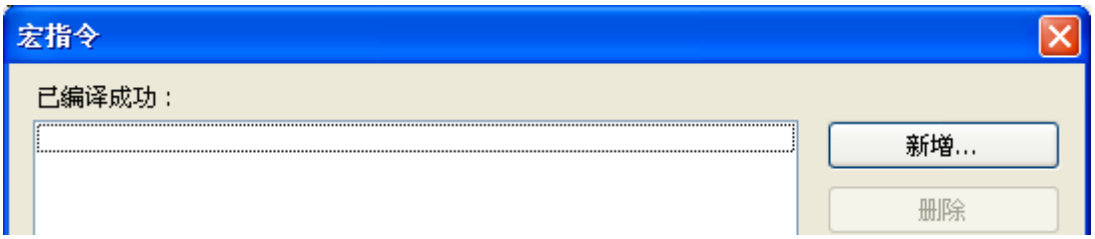
函数名称	描述
StringGet	获取 PLC 的字符串数据。
StringGetEx	获取 PLC 的字符串数据，不等待 PLC 响应，径自往下执行。
StringSet	将字符串数据写到 PLC 中。
StringSetEx	将字符串数据写到 PLC 中，不等待 PLC 响应，径自往下执行。
StringCopy	利用此函数进行字符串的复制。
StringMid	利用此函数可将一个字符串中的某一段子字符串提取出来。
StringDecAsc2Bin	此函数将十进制字符串转换成整数。
StringBin2DecAsc	此函数将整数转换成十进制字符串。
StringDecAsc2Float	此函数将十进制字符串转换成浮点数。
StringFloat2DecAsc	此函数将浮点数转换成十进制字符串。

StringHexAsc2Bin	此函数将十六进制字符串转换成整数。
StringBin2HexAsc	此函数将整数转换成十六进制字符串。
StringLength	取得字符串的长度。
StringCat	连接两字符串。
StringCompare	比较两字符串的内容是否相等。大小写视为不同。
StringCompareNoCase	比较两字符串的内容是否相等。大小写视为相同。
StringFind	在某个字符串中寻找另一个字符串。
StringReverseFind	在某个字符串中寻找另一个字符串，从后面开始找。
StringFindOneOf	寻找某字符串中任一个字符在另一个字符串中第一次出现的位置。
StringIncluding	从某字符串第一个字符开始提取包含特定字符的一段字符串。
StringExcluding	从某字符串第一个字符开始提取不包含特定字符的一段字符串。
StringToUpper	字符全部转换成大写。
StringToLower	字符全部转换成小写。
StringToReverse	字符串反转。
StringTrimLeft	裁剪字符串开头的特定字符。
StringTrimRight	裁剪字符串尾端的特定字符。
StringInsert	插入字符串到另一字符串中的特定位置。

上表中所有字符串处理函数的详细规格与使用范例请参考内置函数功能的章节。下面将举例说明如何从新增一个宏指令的方法开始，一直到执行模拟为止，带您一步步建立一个可执行的工程文件，以实际体会字符串处理函数强大的功能。

1. 以下说明如何从寄存器读入与写入一个字符串。

请新增 1 个宏指令：



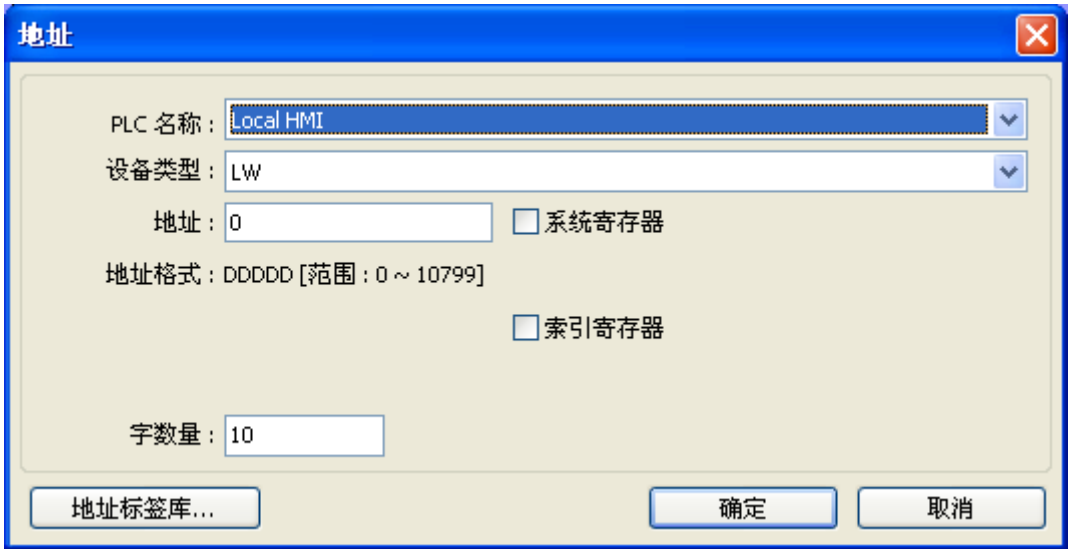
在宏指令编辑器编辑以下内容：

```
1
2   macro_command main()
3
4   char str[20]
5
6   StringGet(str[0], "Local HMI", LW, 0, 20)
7   StringSet(str[0], "Local HMI", LW, 50, 20)
8
9   end macro_command
```

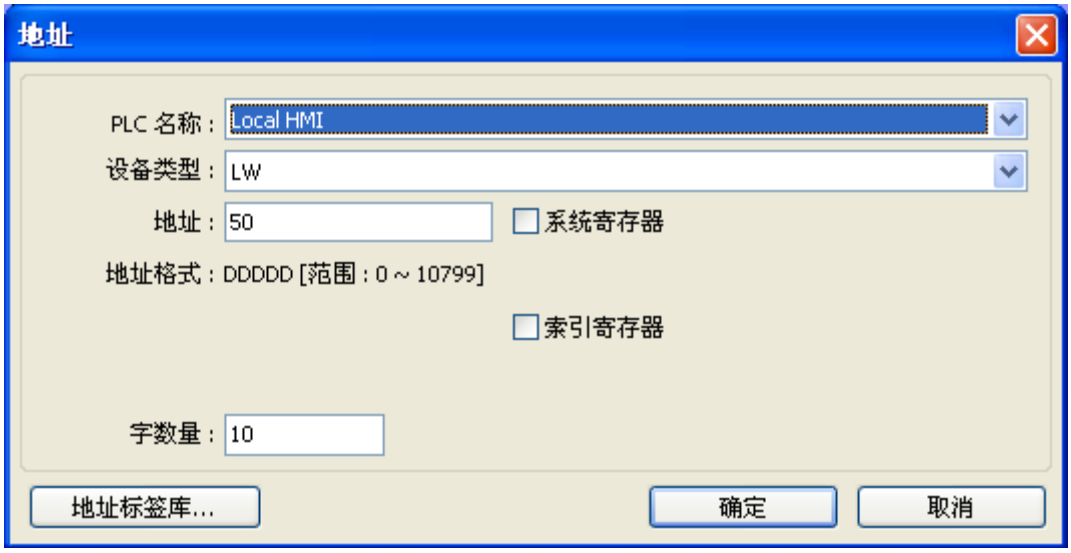
此宏指令的目的是利用 StringGet 函数从寄存器中读入一条字符串放入字符数组 str 中，并利用 StringSet 输出 str 的内容。

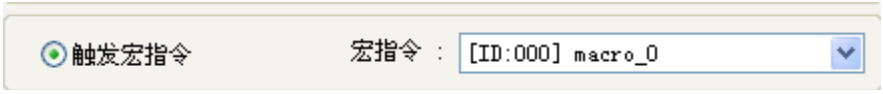
接着在工程文件的第 10 页分别加上“文字输入”与“功能键”元件，元件的设定内容请参考下图，”功能键”元件用来执行 macro_0。

“文字输入”元件



“功能键”元件

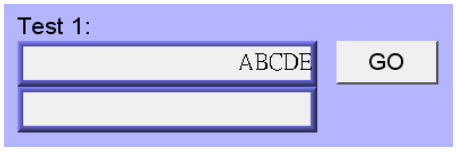




最后编译 已完成的工程文件并执行离线 (off-line) 或在线 (on-line) 模拟，并在模拟画面按照以下步骤操作：

Step 1. 输入字符串

Step 2. 按下“GO”按钮



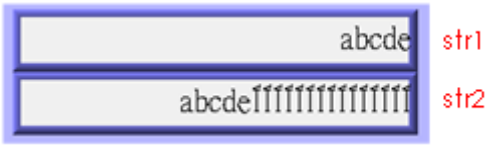
Step 3. 显示结果

2. 字符串之初始化。

在宏指令编辑器编辑以下内容：

```
1
2   macro_command main()
3
4   char str1[20]="abcde"
5   char str2[20]={'a','b','c','d','e'}
6
7   StringSet(str1[0], "Local HMI", LW, 0, 20)
8   StringSet(str2[0], "Local HMI", LW, 50, 20)
9
10  end macro_command
```

用双引号（“ ”）刮起来的内容视为一个字符串。str1 是以字符串方式初始化，而 str2 是以字符数组方式初始化。



以字符串方式初始化时宏编译器会在字符串结尾后面加上结束符号 '\0' 代表字符串结束。利用 StringSet 将字符串写入寄存器时遇到结束符号便会停止继续写入数组后面的内容。即使 data count 被设为大于字符串长度的数值，显示字符串时只会显示结束符号之前的内容。

以字符数组方式初始化时数组内容并不会被视为一个字符串，因此也不会加上结束符号 '\0'。写入寄存器的字符数会依据 data count 所设定的数值决定。

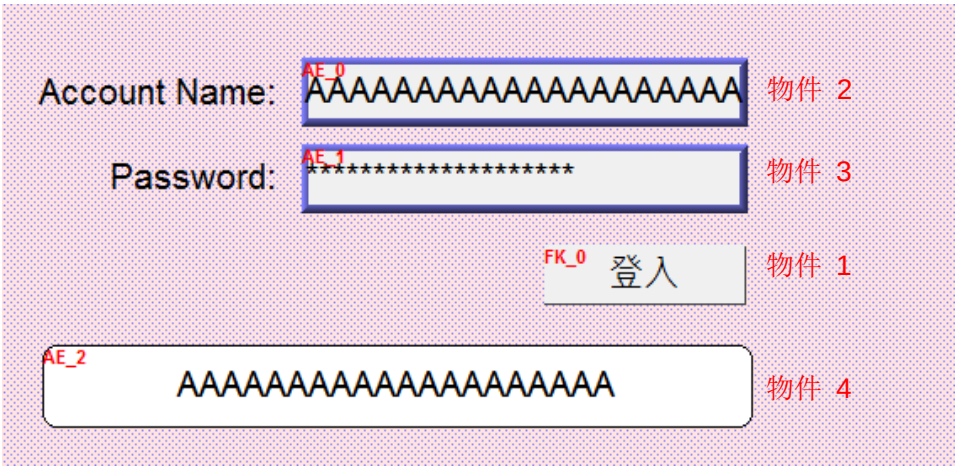
3. 一个简单的账号密码登入页面。

在宏指令编辑器编辑以下内容(宏指令 [ID:001] macro_1)：

```
1
2   macro_command main()
3
4   char name[20]="admin"
5   char password[20]="123456"
6   char name_input[20], password_input[20]
7   char message_success[40]="Success! Access Accepted."
8   char message_fail[40]="Fail! Access Denied."
9   char message_clear[40]
10  bool name_match=false, password_match=false
11
12  StringGet(name_input[0], "Local HMI", LW, 0, 20)
13  StringGet(password_input[0], "Local HMI", LW, 50, 20)
14
15  name_match = StringCompare(name_input[0], name[0])
16  password_match = StringCompare(password_input[0], password[0])
17
18  FILL(message_clear[0], 0x20, 40) //FILL with white space
19  StringSet(message_clear[0], "Local HMI", LW, 100, 40)
20
21  if (name_match==true and password_match==true) then
22      StringSet(message_success[0], "Local HMI", LW, 100, 40)
23  else
24      StringSet(message_fail[0], "Local HMI", LW, 100, 40)
25  end if
26
27  end macro_command
```

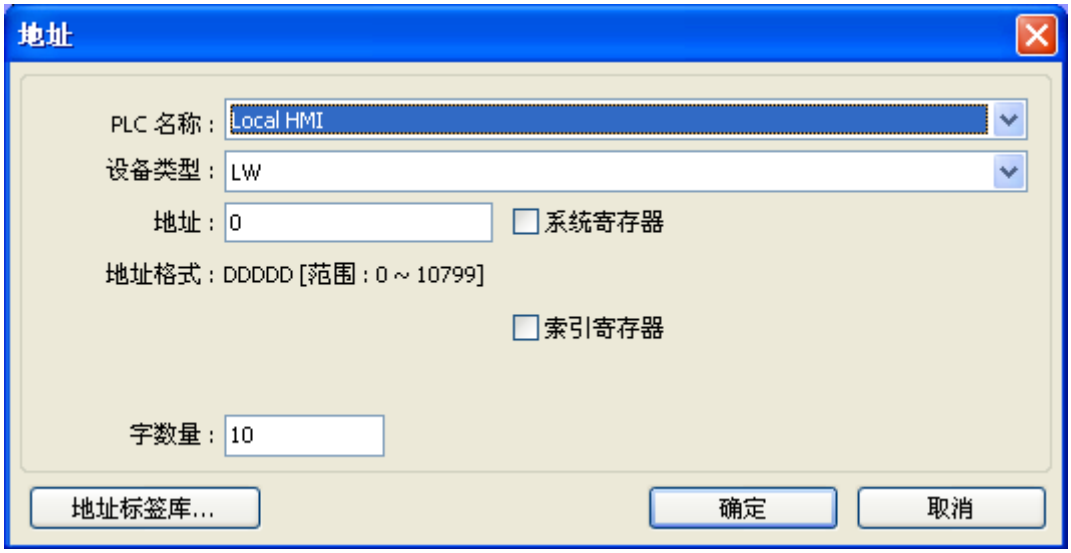
此宏将利用 “StringGet” 取得用户输入的账号密码的字符串，分别放入 name_input 与 password_input 两个数组中。接着利用 “StringCompare” 比对账号密码，若账号相符，则 name_match 设为 true；若密码相符，则 password_match 设为 true。最后检查 name_match 与 password_match 的值，若同时为 true，表示登入成功，并印出 ”Success! Access Accepted.” 字符串。若其中之一不相符，则印出 ”Fail! Access Denied.” 字符串。

接着在工程文件的第 10 页分别加上 “文字输入” 与 “功能键” 元件，元件的设定内容请参考下图，”功能键” 元件用来执行 macro_1。

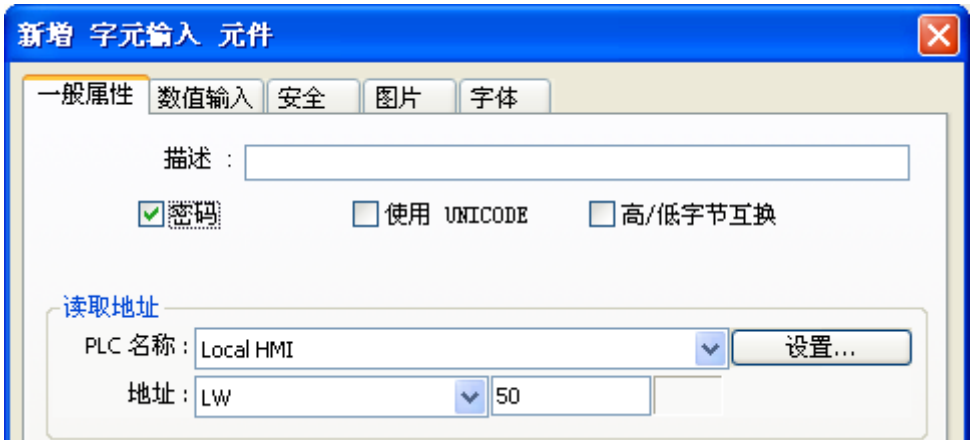


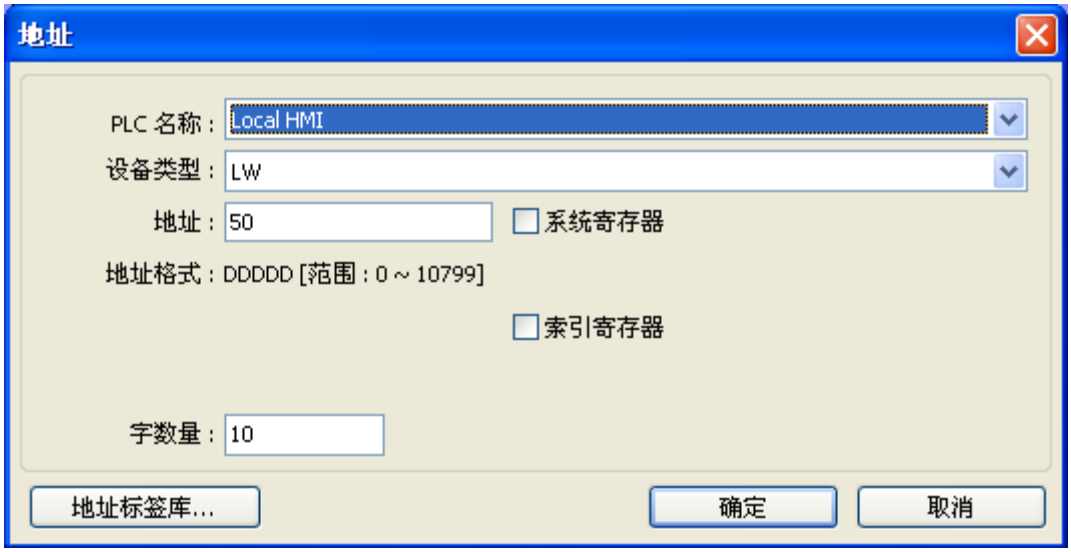
物件 1：”功能键”元件 ，点选“触发宏指令”并选择宏指令 [ID:001] macro_1。

物件 2：”文字输入”元件

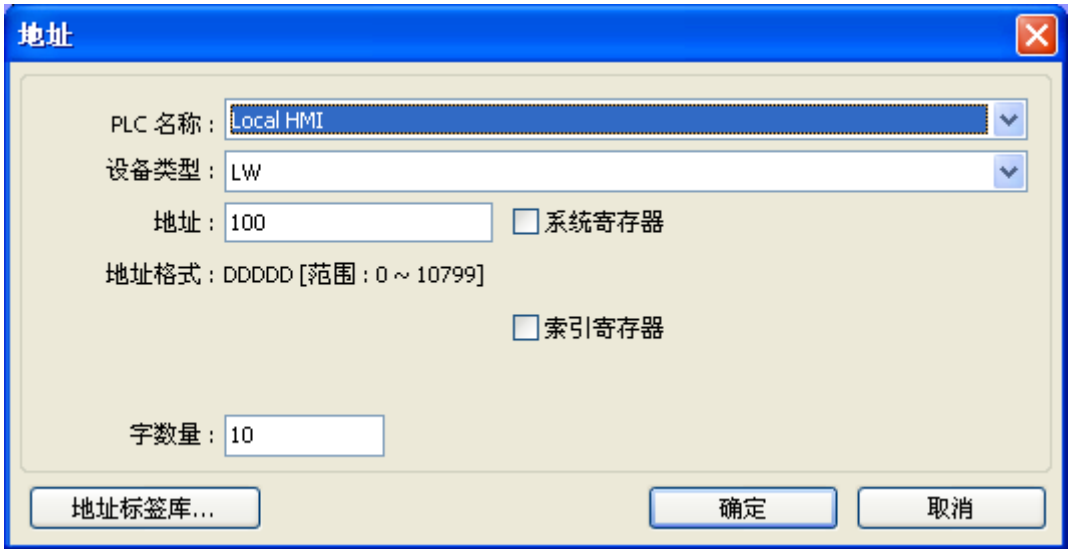


物件 3：”文字输入”元件



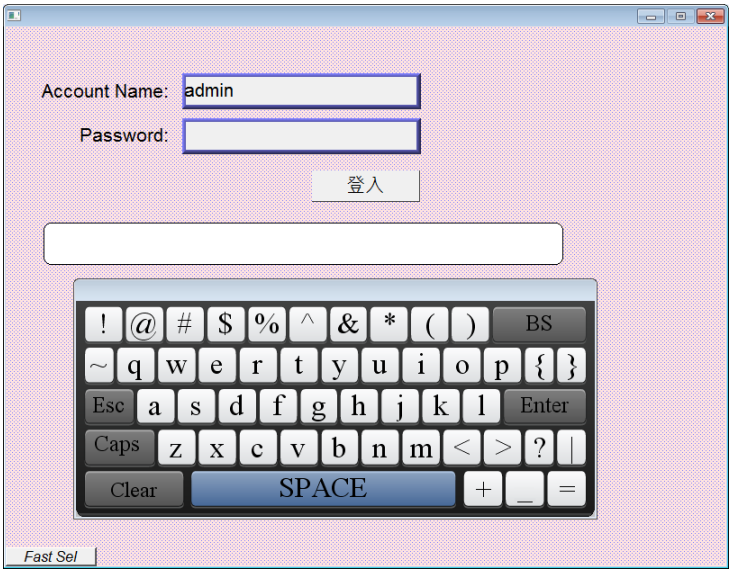


物件 4：”文字显示” 元件

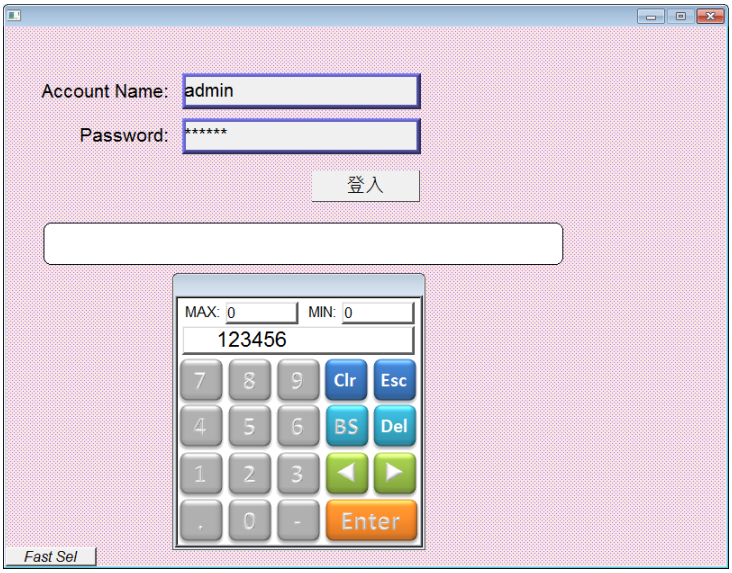


最后编译 已完成的工程文件并执行离线 (off-line) 或在线(on-line) 模拟，并在模拟画面按照以下步骤操作：

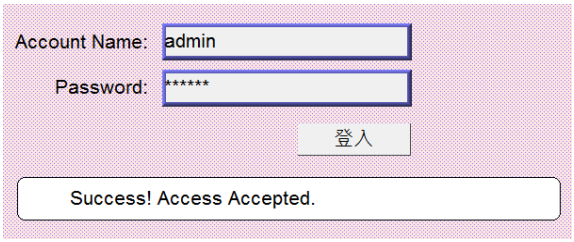
1. 输入账号。



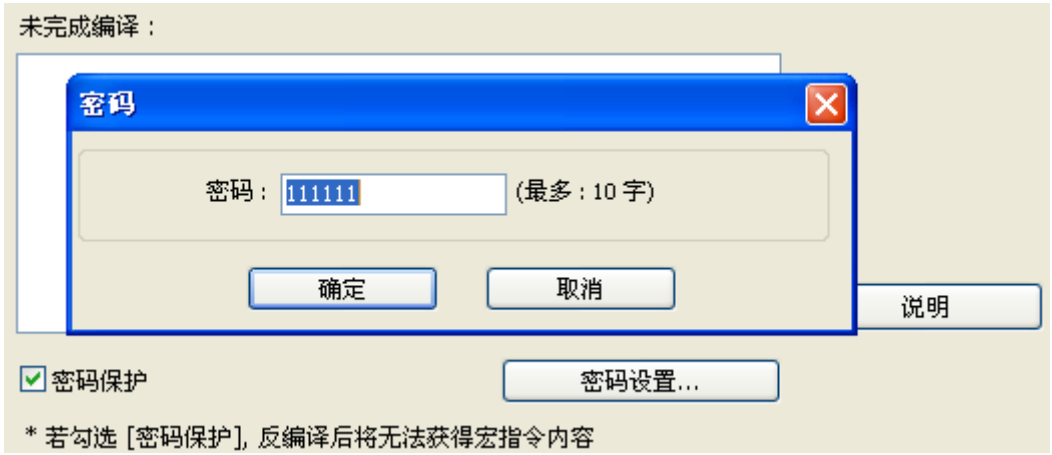
2. 输入密码并按下登入键。



3. 显示登入成功或登入失败。



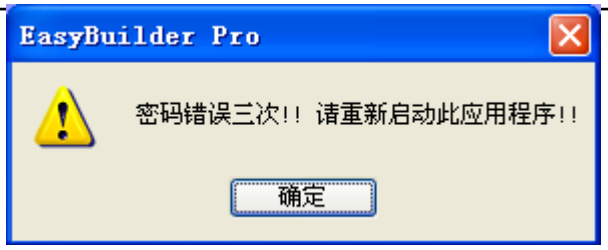
18.16 宏指令密码保护



在宏指令编辑器窗口中提供“密码保护”选项，当勾选“密码保护”后按下“密码设置”按钮可以设定密码，密码最多不可超过 10 个字符（只支持 ASCII 文字，并区分大小写，例如可以输入“a$#*hFds”）。

当设定宏指令“密码保护”功能后，用户欲开启宏指令编辑器窗口时，需先输入正确的密码。

当输入不正确的密码三次，需重新启动 EasyBuilder Pro，才能重新输入密码。



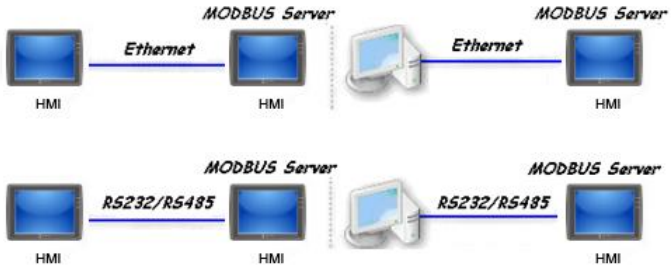
Note

- 当开启宏指令密码保护功能后，反编译 EXOB 文件将无法回复宏指令的内容。

第十九章 如何将触摸屏设定成 MODBUS 设备

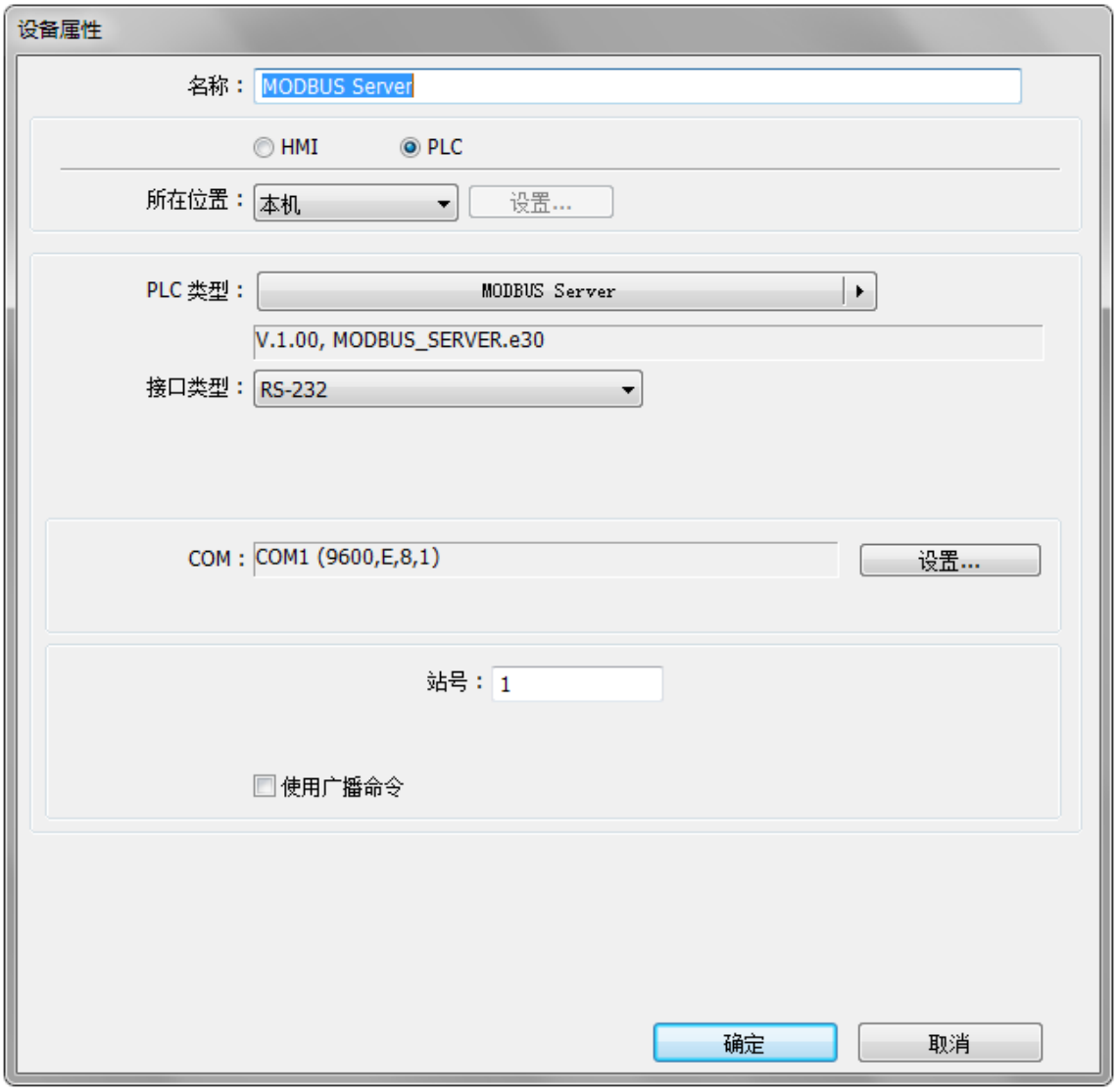
19.1 概要

将触摸屏设定成 MODBUS 设备后，通过 MODBUS 协议即可擦写触摸屏上的数据。
下图显示触摸屏被设定成 MODBUS 设备 (又称为 MODBUS Server)，HMI、PC 或其它设备只需使用 MODBUS 协议，通过 Ethernet 或 RS-232 / RS-485 接口，即可擦写触摸屏上的数据。



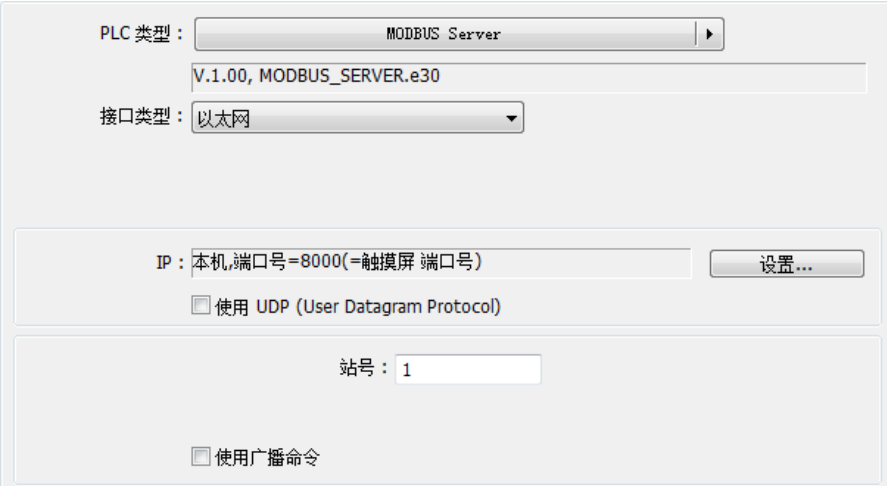
19.2 建立一个 MODBUS Server 设备

1. 要将触摸屏设定为 MODBUS 设备，首先需在“设备清单”中增加一个新的设备，此时 PLC 类型需选择 MODBUS Server，可以选择的“PLC 接口”如下图所示。

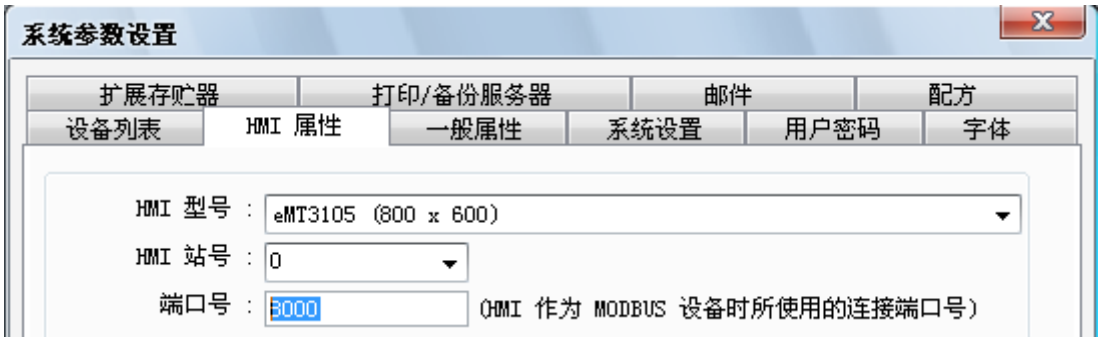


2. 当 PLC 接口选择“RS-232”或“RS-485”时，需选择使用的“COM”(COM 1 ~ COM 3)，并设定正确的通讯参数。如下图，此时 MODBUS Server 的“站号”设定为 1。

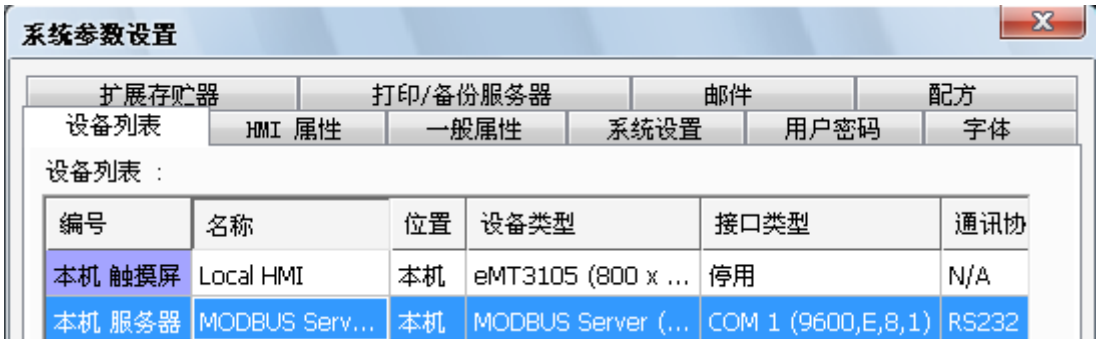
当 PLC 接口选择“以太网”时，需设定“连接端口号”。



因 MODBUS Server 与触摸屏须使用相同的“连接端口”，若要更改 MODBUS Server 的连接端口，需在“HMI 属性”页面中修改。



3. 在按下确定键后，即可在“设备清单”中发现一个新的设备：MODBUS Server，此时即完成 MODBUS 设备的设定，在完成 .emtp 文件的编译并将获得的 .exob 文件下载到触摸屏后，即可通过 MODBUS 协议读写触摸屏上的数据。



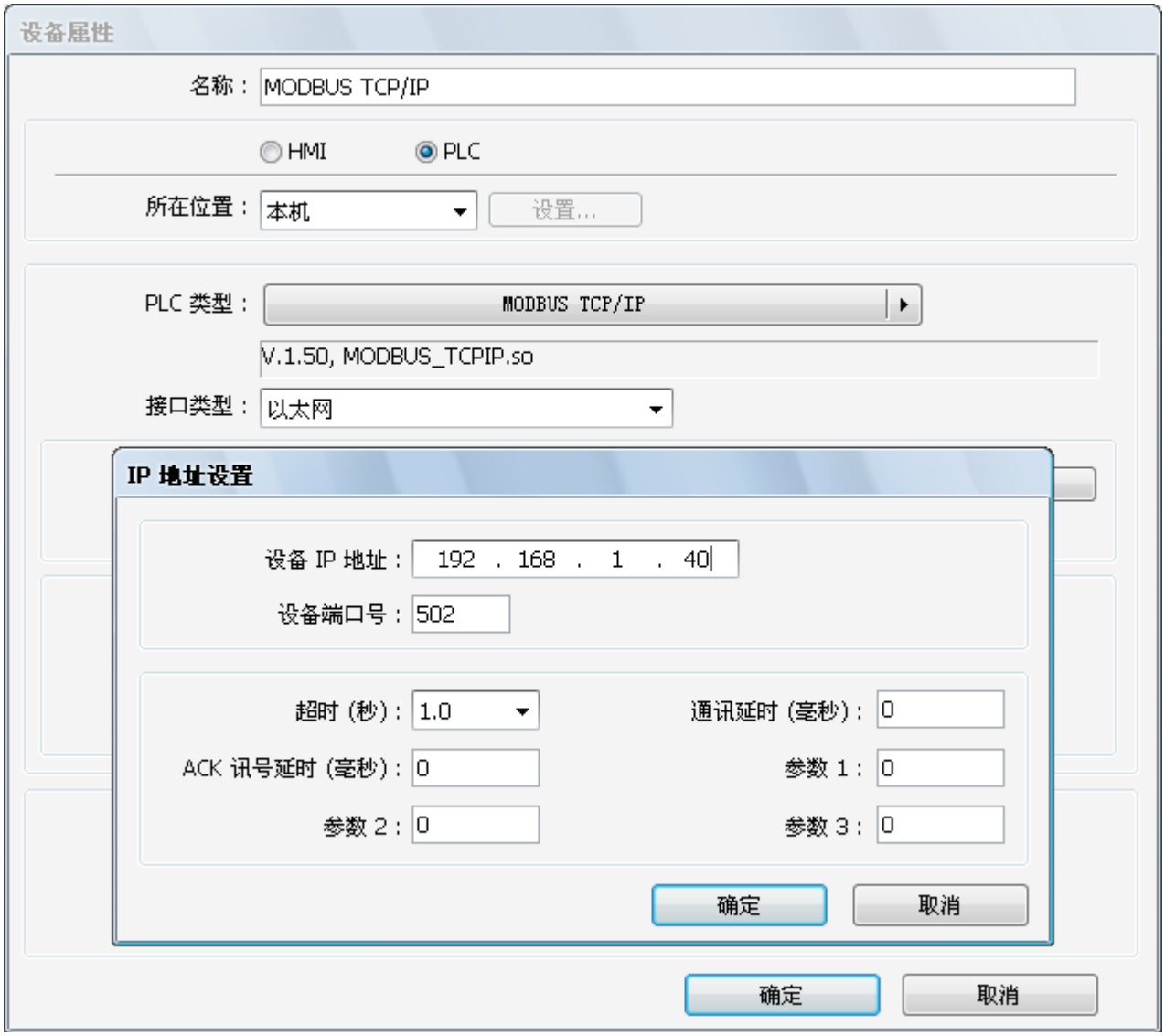
Note

- cMT-SVR 在 PLC 接口选择“以太网”时，连接端口可自行输入。

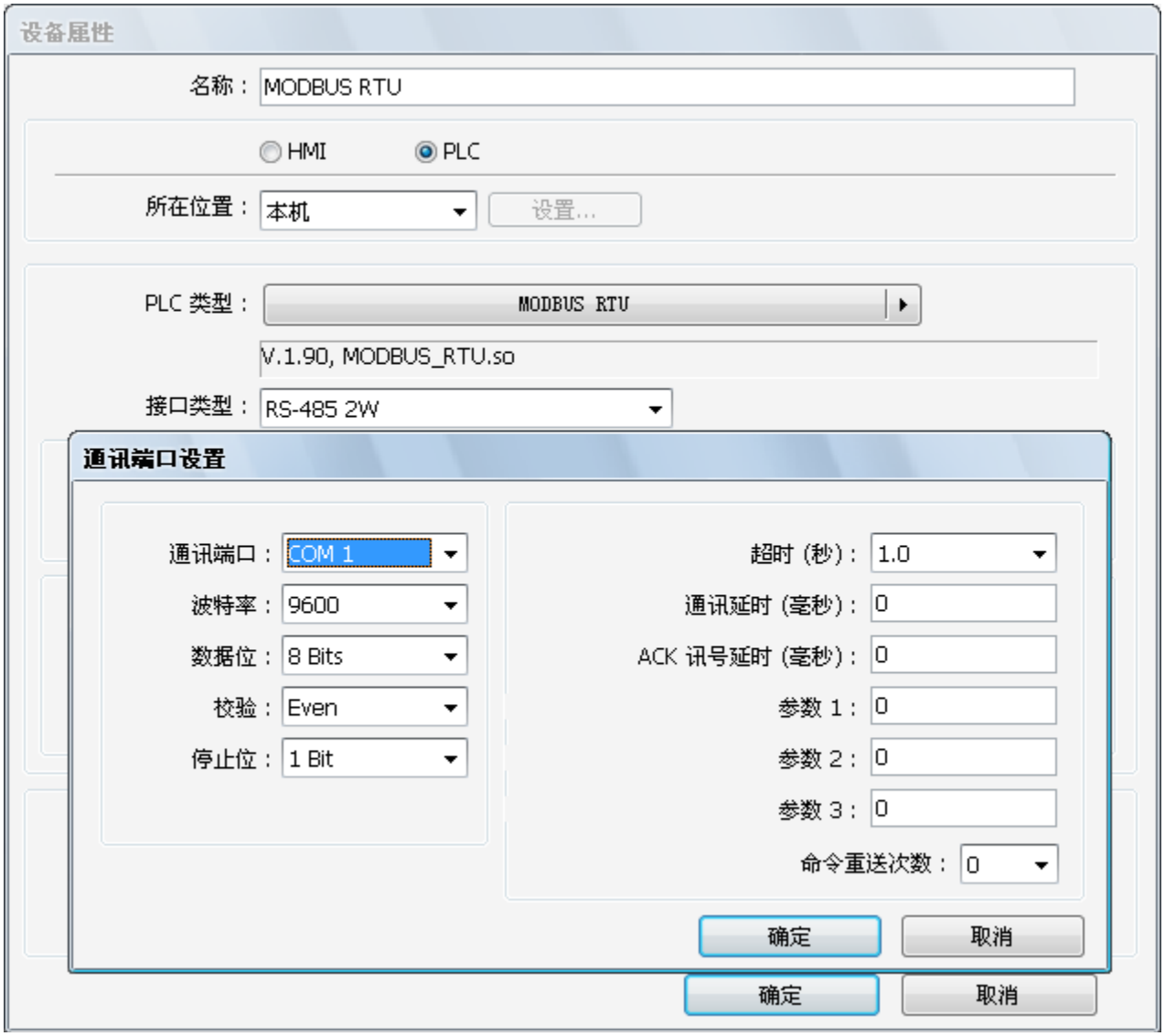
19.3 读写一个 MODBUS Server 设备

两台触摸屏可以通过设定成 MODBUS Client (主机端) 和 Server (从机端) 相互通讯。

1. 在 Client 端的设备清单中，需增加一个新的设备。若 Client 端使用“以太网”接口，则“PLC 类型”需挑选 MODBUS TCP/IP，并正确设定“IP 地址”(即 server 端所在位置的 IP)、”连接端口”与“站号”。

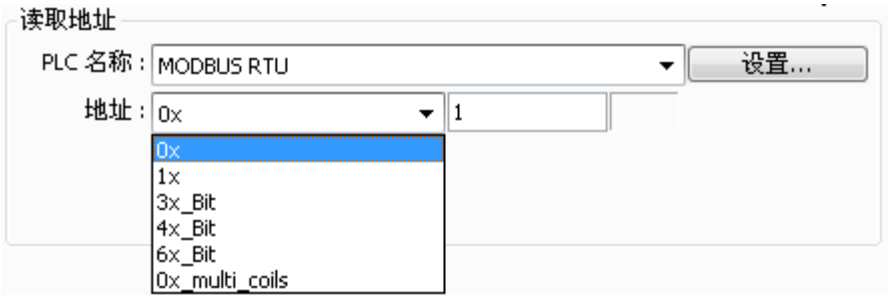


若 Client 端要使用“RS-232”或“RS-485”界面。则“PLC 类型”需挑选 MODBUS RTU，并正确设定各项通讯参数。



2. 完成各项设定并按下确定键后，即可在"设备清单"中发现一个新的设备 "MODBUS RTU"。

3. 开启各个元件的设定页，在“PLC 名称”选择 MODBUS RTU 后，即可设定 MODBUS 设备的各项读写地址。



此时因被读写的设备（Server 端）为 HMI，所以实际读写的位置的对应关系如下：

读写 0x/1x (1 ~ 12096)	对应到 读写 LB (0 ~ 12095)
读写 3x/4x/5x (1 ~ 9999)	对应到 读写 LW (0 ~ 9998)
读写 3x/4x/5x (10000 ~ 65535)	对应到 读写 RW (0 ~ 55535)

19.4 在线更改 MODBUS Server 站号

EasyBuilder Pro 提供下列系统寄存器，让用户可以在线更改 MODBUS Server 所使用的站号。

LW-9541	MODBUS/ASCII server 站号 (COM 1)
LW-9542	MODBUS/ASCII server 站号 (COM 2)
LW-9543	MODBUS/ASCII server 站号 (COM 3)
LW-9544	MODBUS/ASCII server 站号 (Ethernet)

19.5 关于 MODBUS 各地址的说明

EasyBuilder Pro 中 MODBUS 协议的设备类型为 0x，1x，3x，4x，5x，6x，还有 3x_bit，4x_bit 等，下面将分别说明这些设备类型在 MODBUS 协议中支持哪些功能码。

0x	是个可读可写的设备类型，相当于操作 PLC 的输出点。该设备类型读位状态的时候，发出的功能码为 01H，写位状态的时候发出的功能码为 05H。写多个位寄存器时，发出的功能码为 0fH。
1x	是个只读的设备类型，相当于读 PLC 的输入点。读位状态的时候发出的功能码为 02H。
3x	是个只读的设备类型，相当于读 PLC 的只读数据寄存器。读数据的时候，发出的功能码为 04H。
4x	是个可读可写的设备类型，相当于操作 PLC 的数据寄存器。当读数据的时候，发出的功能码是 03H，当写数据的时候发出的功能码是 10H。

5x	该设备类型与 4x 的设备类型属性是一样的。即发出读写的功能码完全一样。不同之处在于，当为双字符时，若 32_bit unsigned 格式的数据，使用 5x 和 4x 两种设备类型分别读取数据时，高字符和低字符的位置是颠倒的。若使用 4x 设备类型读到的数据是 0x1234，那么使用 5x 设备类型读取的数据是 0x3412。
6x	是一个可读可写的设备类型，读数据的时候发出的功能码也是 03H，与 4x 不同之处在于写数据的时候，发出的功能码为 06H，即写单个寄存器的数据。
3x_bit	该设备类型支持的功能码与 3x 设备类型完全一致，不同之处是 3x 是读数据，而 3x_bit 是读数据中的某一个 bit 的状态。
4x_bit	该设备类型支持的功能码与 4x 设备类型完全一致，不同之处是 4x 是读数据，而 4x_bit 是读数据中的某一个 bit 的状态。

更多信息请参考《37 MODBUS TCP/IP 网关功能》。

第二十章 如何使用条形码扫描仪

20.1 概要

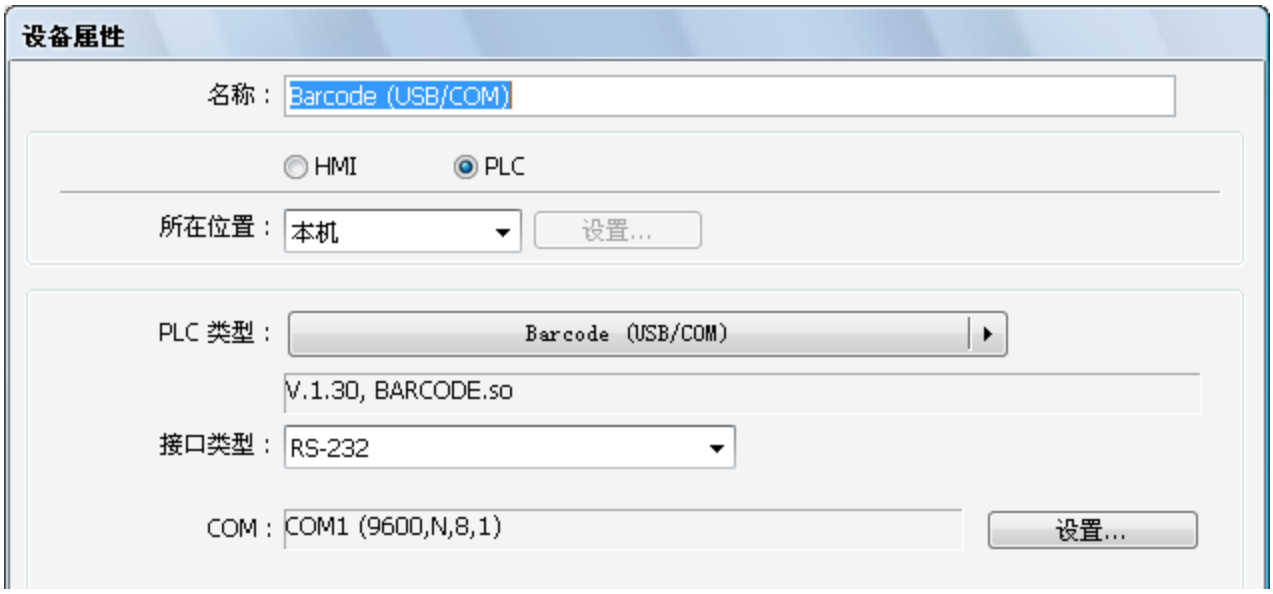
HMI 支持通过下列通讯端口连接条形码扫描仪：

- USB
- COM Port

欲连接条形码扫描仪，请先在设备清单中增加一个新设备。

20.2 连接条形码扫描仪的步骤

1. 在“编辑菜单”»“系统参数设置” »“设备清单”页面中增加一个设备。



2. 按下“设置”按钮并完成“条形码扫描仪 / 键盘设置”。

设定	描述
超时	当勾选“键盘”时，可设定键盘数据输入之时间范围，系统将于开始输入数据时计时。
通讯端口 **传输速率** **数据位** **校验** **停止位**	当选用 COM 接口时，须正确设定条形码扫描仪的通讯参数；若选用 USB 接口，则无须设定通讯参数。

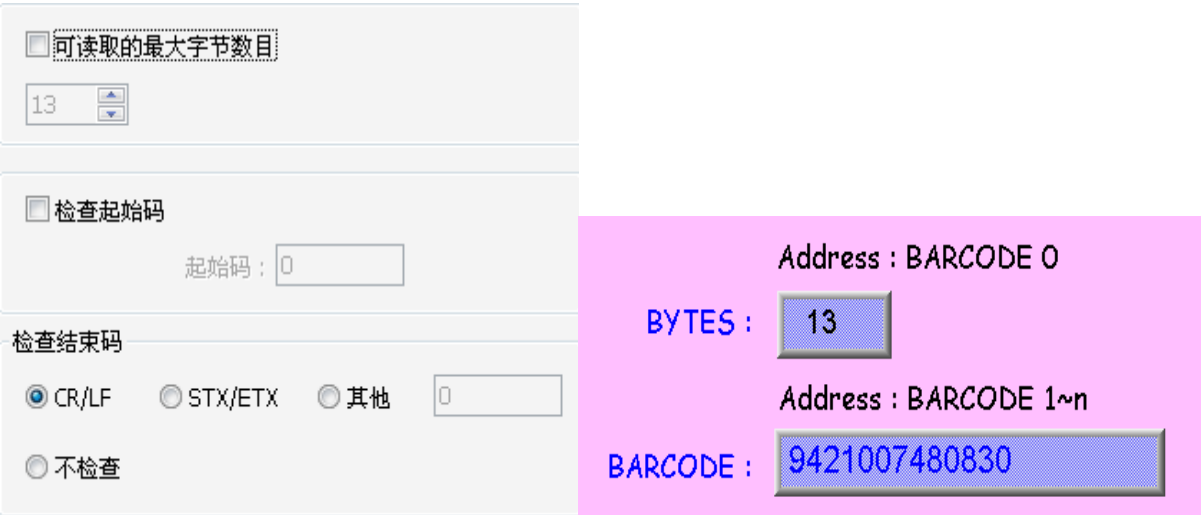
可读取的最大 byte 数目	若勾选，则可以限制条形码扫描仪读取的 byte 数目，以避免设备读取过多的数据。此项设定值范围为 10 ~ 512。 例如: 设定“可读取的最大 byte 数目”为 10，若条形码扫描仪读取到的数据为 12 个 byte 0x34 0x39 0x31 0x32 0x30 0x30 0x34 0x37 0x30 0x38 0x33 0x38 但因为限制读取数目为 10 个 byte，所以实际读取成功的数据为 0x34 0x39 0x31 0x32 0x30 0x30 0x34 0x37 0x30 0x38
检查起始码	若勾选，则条形码扫描仪所读取到的第一个数据必须与起始码相同，系统才会将读取的数据视为是合法的输入，否则将会忽略读取的数据。 起始码并不会被存放在条形码扫描仪所对应的地址中。 例如起始码为 255 (0xff)，且读取到的数据为 *0xff* 0x34 0x39 0x31 0x32 0x30 0x30 0x34 0x37 则实际存放在条形码扫描仪对应的地址中的数据为 0x34 0x39 0x31 0x32 0x30 0x30 0x34 0x37
结束码设定	结束码用来标示数据的结尾，当读取到结束码时，表示读取到一笔完整的数据。
CR/LF	0x0a 或 0x0d 皆为结束码。
STX/ETX	0x02 或 0x03 皆为结束码。
其它	由用户自订数据的结束码。
不检查	若选择此项设定，HMI 会将全部读取到的数据存放至条形码扫描仪对应的地址中。

完成以上各项设定后，即可在“设备清单”中发现一个新的条形码扫描仪设备。
此时在元件的设定属性页中的“PLC 名称”即可选择条形码扫描仪，并可使用相关的地址类型。

地址类型	地址名称	描述
位	FLAG	**FLAG 0:** 指示数据是否读取完成。在读取到数据时，系统会自动将 FLAG 先设定为 OFF，待读取成功后再设定为 ON。
	RESET	**RESET 0:** 当设为 ON 时，可清除 BARCODE 和 RESULT 内的数据。
字符	BARCODE	**BARCODE 0:** 记录目前读取到的 byte 数目。 **BARCODE 1 ~ n:** 存放设备读取的数据。
	RESULT	**RESULT 0:** 指示 BARCODE 的读取结果。各项数据的表示意义如下: 0x00: 等待读取 BARCODE。 0x01: 读取 BARCODE 成功。 0x02: BARCODE 格式错误。 0x03: 在启用“可读取的最大 byte 数目”时，所读取的数据长度超过所设定的大小。 0x04: 在启用“检查起始码”时，所读取的数据不符合设定值。 0x05: 在启用“结束码”时，所读取的数据不符合设定值。

范例 1

假设目前条形码扫描仪的设定如下图，且读取到的条形码为 9421007480830，图中的数值显示元件 (BYTES) 的地址为 BARCODE 0，字符显示物件 (BARCODE) 的地址为 BARCODE 1 ~ n。



此时条形码扫描仪设备对应的地址所存放的数据如下：

条形码扫描仪对应地址	数据
BARCODE 0	13 bytes (十进制) 但实际存入地址中的数据为 14 bytes = 7 words 也就是当读取 byte 数目为奇数时，系统会自动加上一个 byte 的数据 (0x00)
BARCODE 1	3439 (HEX)
BARCODE 2	3132 (HEX)
BARCODE 3	3030 (HEX)
BARCODE 4	3437 (HEX)
BARCODE 5	3038 (HEX)
BARCODE 6	3338 (HEX)
BARCODE 7	0030 (HEX)

Note

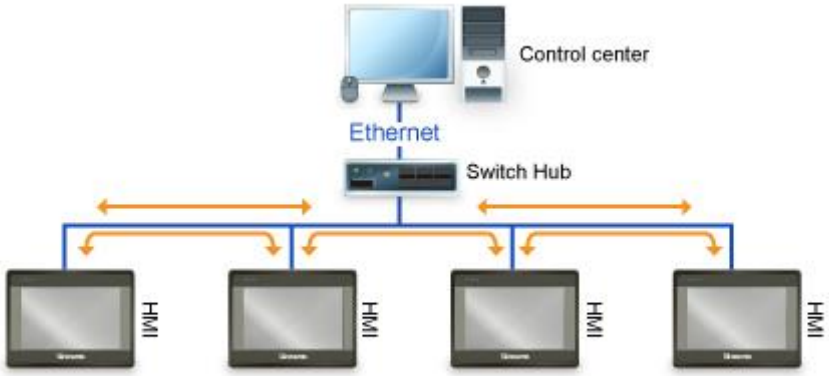
- 每台触摸屏只支持連接一台 USB 接口的条形码扫描仪设备。当工程文件的设备列表中包含 USB 条形码扫描仪设备时，系统寄存器 LB-9064 “启用 USB 条形码扫描仪设备 (键盘功能关闭) (当状态为 ON)” 将自动被设定为 ON。若此时需恢復 USB 键盘的功能并暂停使用 USB 条形码扫描仪，可以将 LB-9064 设定为 OFF。

第二十一章 以太网通讯与多台触摸屏联机

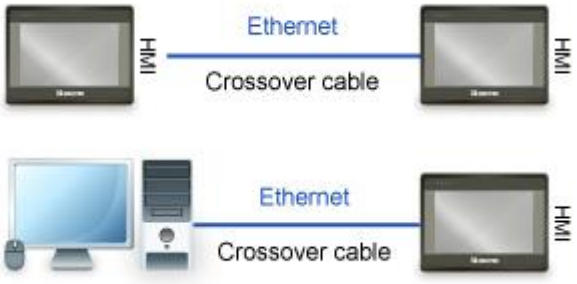
21.1 概要

以太网联机的方式分为两种：

- 使用 RJ45 平行网络线与集线器。



- 使用 RJ45 跳接网络线，不需使用集线器，但只限使用在一对一联机的情况下 (HMI 对 HMI，或电脑 对 HMI)。



通过以太网联机，系统提供了下列三种数据交换的方式：

- HMI 与触摸屏间的通讯。
- PC 与触摸屏间的通讯。
- 控制连接在其它触摸屏上的 PLC。

21.2 HMI 与触摸屏间的通讯

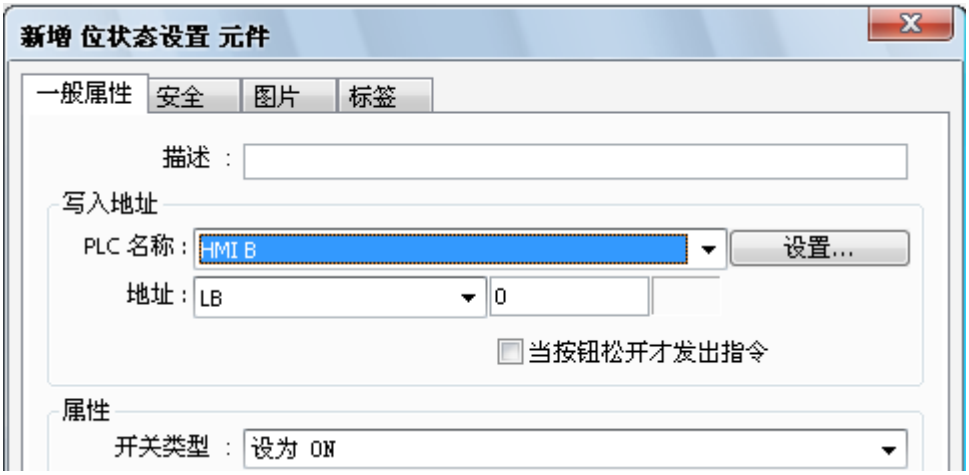
触摸屏之间通讯可在“系统参数设置” 中新增一个远程 HMI 设备即可。

以两台触摸屏的通讯为例 (HMI A 与触摸屏 B)，假设触摸屏 A 欲使用位状态设定元件控制触摸屏 B 的

“LB-0”地址的内容，则触摸屏 A 工程文件的设定步骤如下。

1. 设定各台触摸屏的 IP 地址，假设触摸屏 A: 192.168.1.1，HMI B: 192.168.1.2。
2. 自”系统参数设置” »“设备列表”，新增一台远程 HMI，即为触摸屏 B (IP: 192.168.1.2)。

3. 设定一个位状态设定元件，在“PLC 名称”中选择“HMI B”，即可控制远程触摸屏的地址。

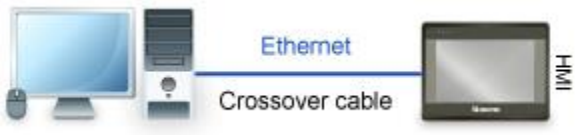


Note

- 一台 HMI 最多可同时处理来自 64 个不同触摸屏的访问要求。
- 一台 cMT-SVR 最多可同时处理来自 32 个不同触摸屏的访问要求。

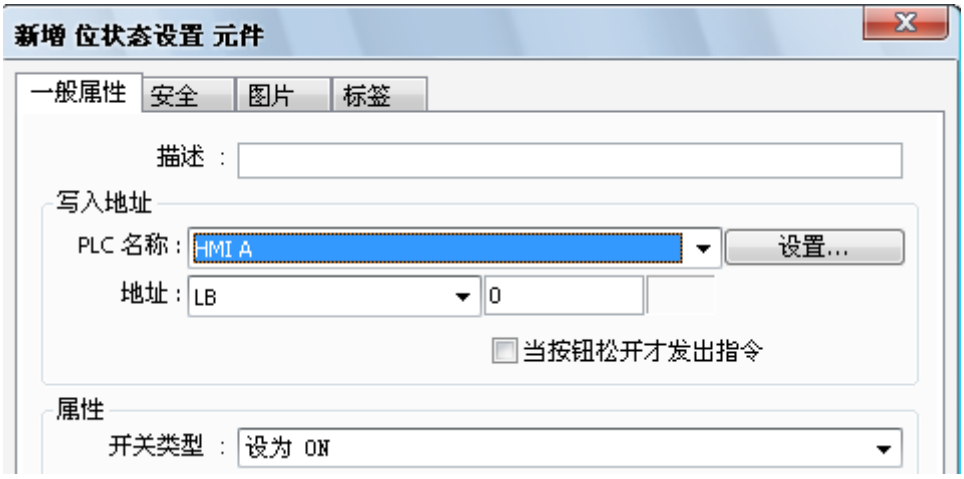
21.3 PC 与触摸屏间的通讯

通过在线模拟功能，PC 可以藉由以太网撷取触摸屏上的数据，并保存在电脑 上。假设电脑 欲通讯的设备为两台触摸屏(HMI A 与触摸屏 B)，则电脑 端所使用工程文件的设定步骤如下。



1. 设定各台触摸屏的 IP 地址，假设触摸屏 A: 192.168.1.1，HMI B: 192.168.1.2。
2. 自"系统参数设置" »"设备列表"，新增两台远程 HMI，分别为触摸屏 A (IP:192.168.1.1)，与触摸屏 B (IP:192.168.1.2)。

3. 设定一个位状态设定元件，在"PLC 名称"中选择"HMI A"，即可控制远程触摸屏 A 的地址。同样的方式也可用于 HMI B。



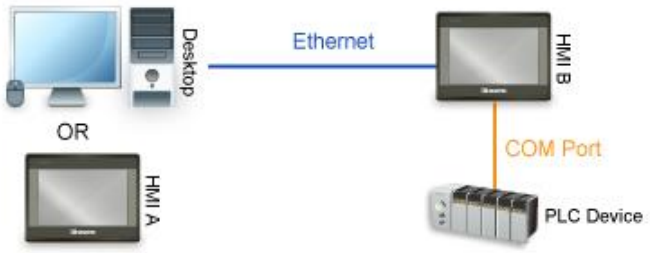
Note

- 一台电脑 最多可同时控制 64 台远程 HMI。
- 如上面的例子，HMI 也允许操作电脑 上的数据，此时只需将电脑 视为另一台触摸屏即可，也就是必须在触摸屏 A /触摸屏 B 使用的工程文件中新增一台远程 HMI，并将此远程触摸屏的 IP 地址指向

电脑。

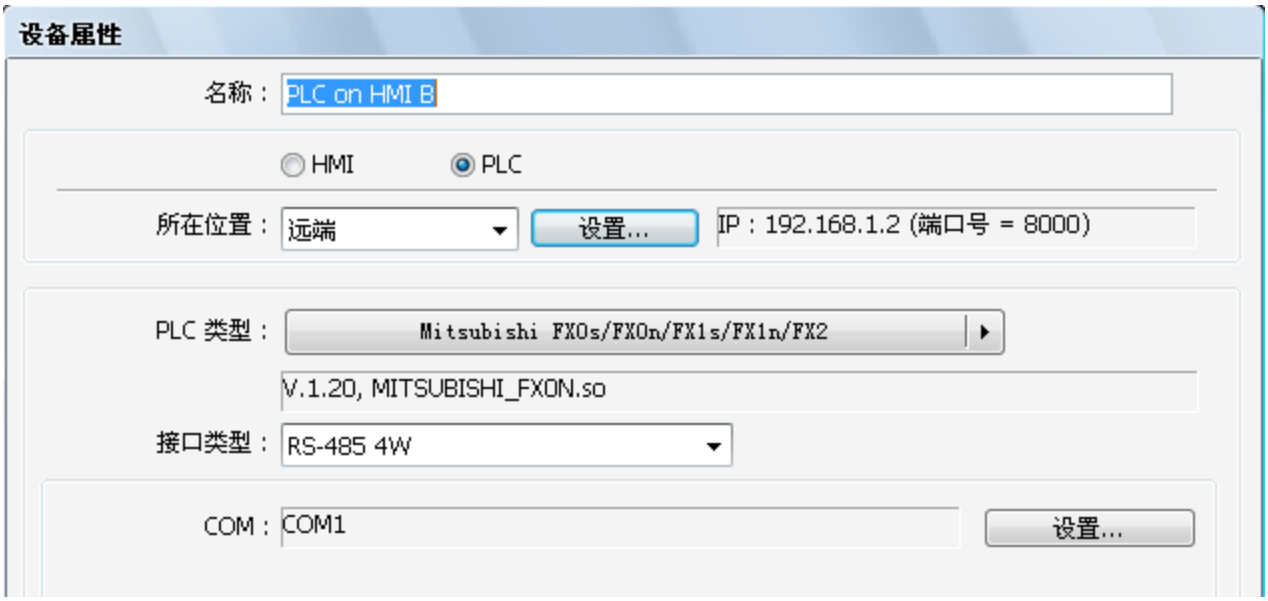
21.4 控制连接在其它触摸屏上的 PLC

通过以太网联机，PC 或触摸屏可以操作连接在其它触摸屏上的远程 PLC。假设现在有一台 PLC 连接到触摸屏 B 的 COM 1，当电脑 或触摸屏 A 欲读取此台 PLC 上的数据，则电脑 端或触摸屏 A 上所使用工程文件设定步骤如下。

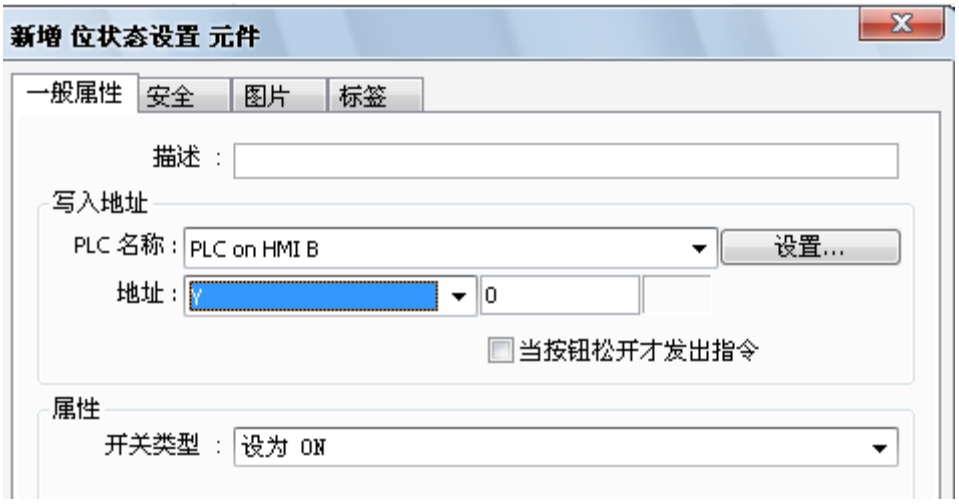


21.4.1 eMT / cMT-HD 系列的设定方法

1. 设定触摸屏 B 的 IP 地址，假设触摸屏 B: 192.168.1.2。
2. “系统参数设置” »“设备列表”，新增一台远程 PLC，将名称设为 “PLC on 触摸屏 B” 并正确设定 PLC 的相关通讯参数。因此台 PLC 是连接在远程触摸屏 B 上，所以将远程 IP 地址指向触摸屏 B (IP: 192.168.1.2)。

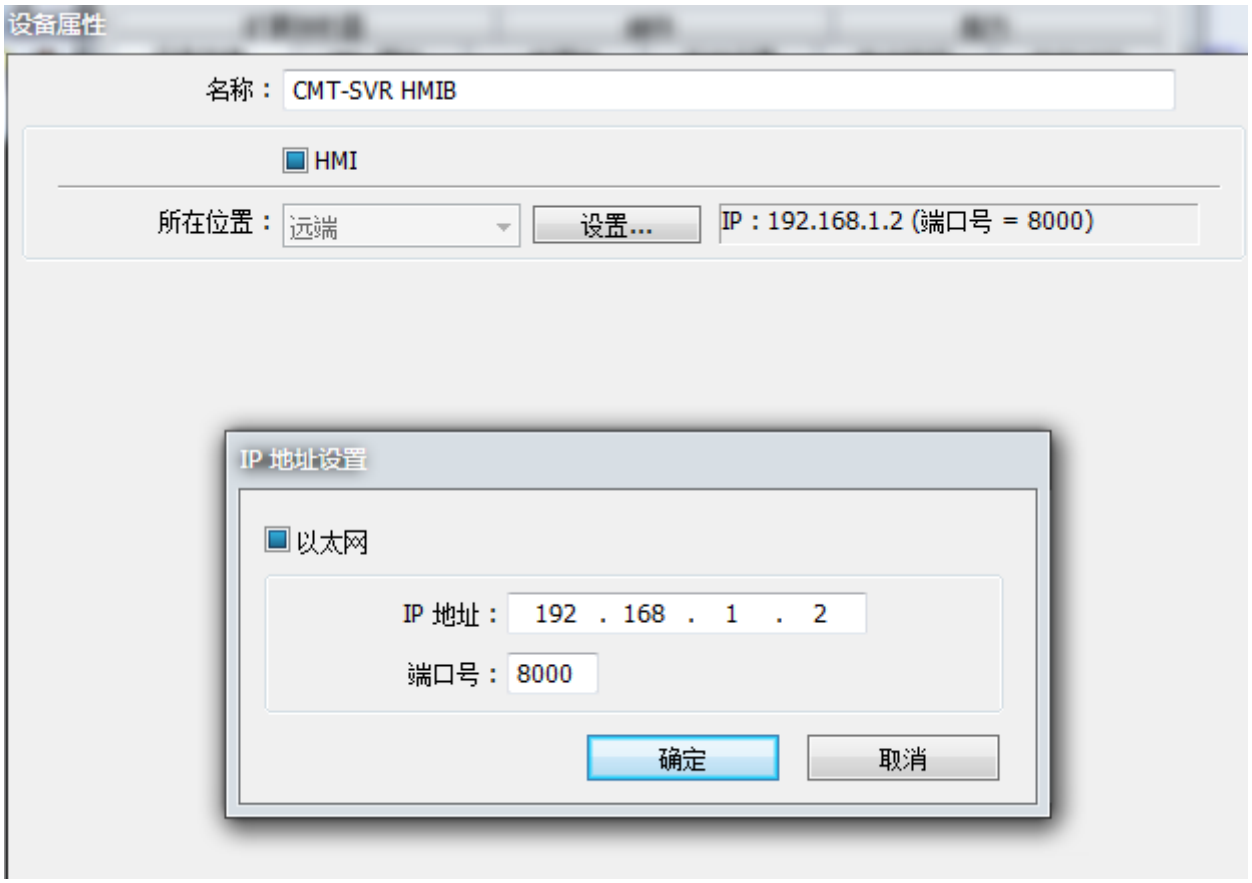


3. 设定一个位状态设定元件，在 “PLC 名称” 中选择 “PLC on 触摸屏 B”，即可控制远程触摸屏 B 上的 PLC。

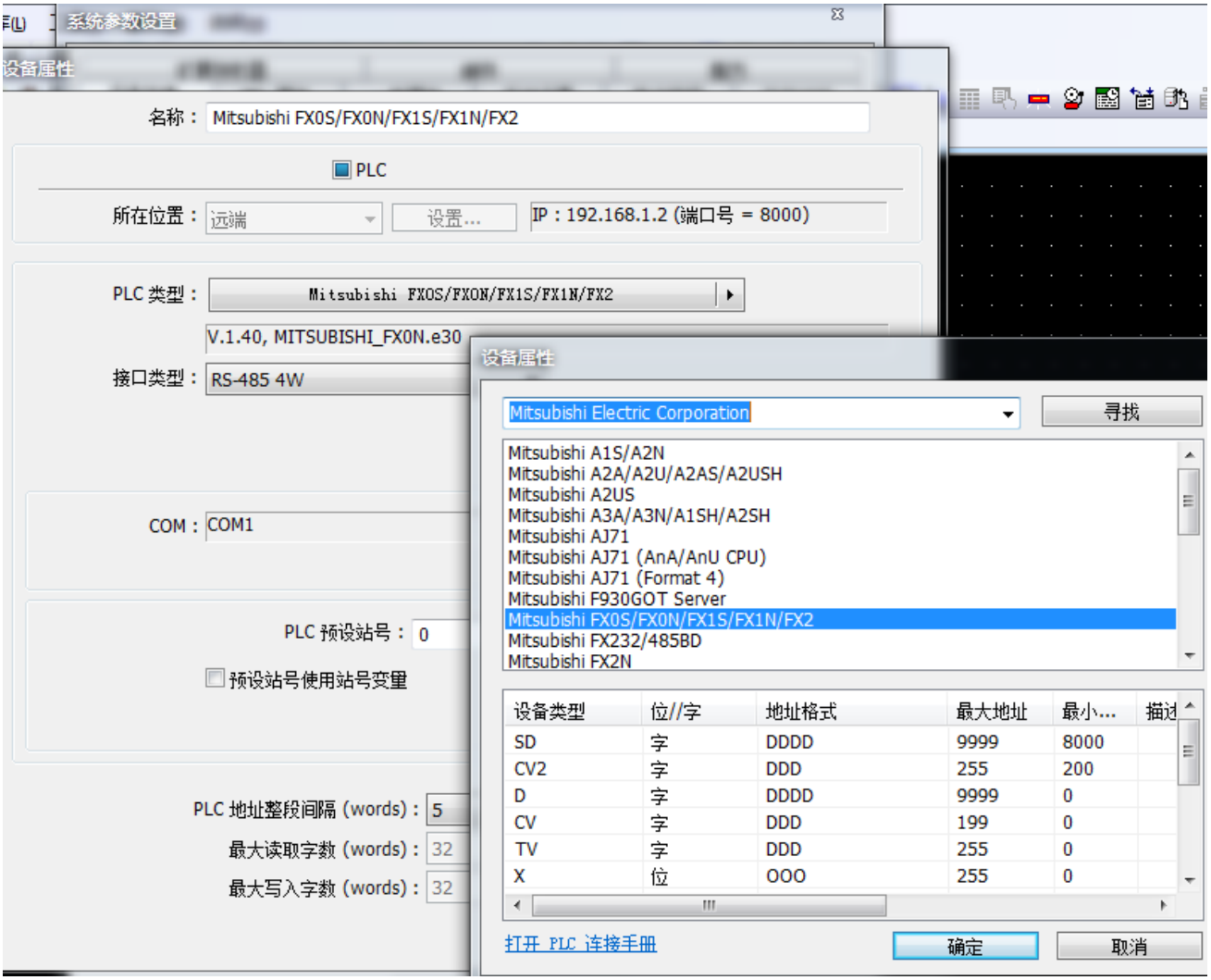


21.4.2 cMT-SVR 系列的设定方法

1. 设定触摸屏 B 的 IP 地址，假设触摸屏 B: 192.168.1.2。
2. 自“系统参数设置” 中，”新增 HMI”并设定 HMI B 的 IP 地址，假设触摸屏 B: 192.168.1.2。

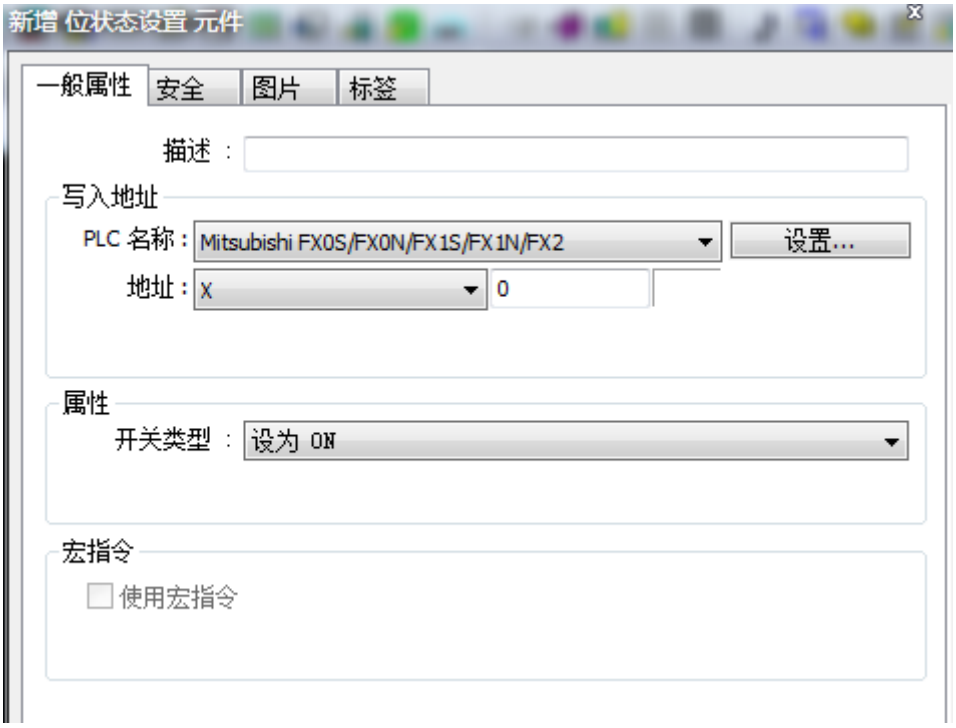


3. 在 HMI B 底下点选 “新增 PLC”，新增一台远程 PLC，将名称设为 “PLC on 触摸屏 B” 并正确设定 PLC 的相关通讯参数。



4. 建立完成后，可以看到一台远程的 PLC 设定被建立在远端的 HMI 下面，本机 HMI 代表的是 HMI A，远端 HMI 1 是 HMI B，远端 PLC 1 则是 HMI B 所连接的 PLC。

5. 设定一个位状态设定元件，在“PLC 名称”中选择“PLC on 触摸屏 B”，即可控制远端触摸屏 B 上的 PLC。



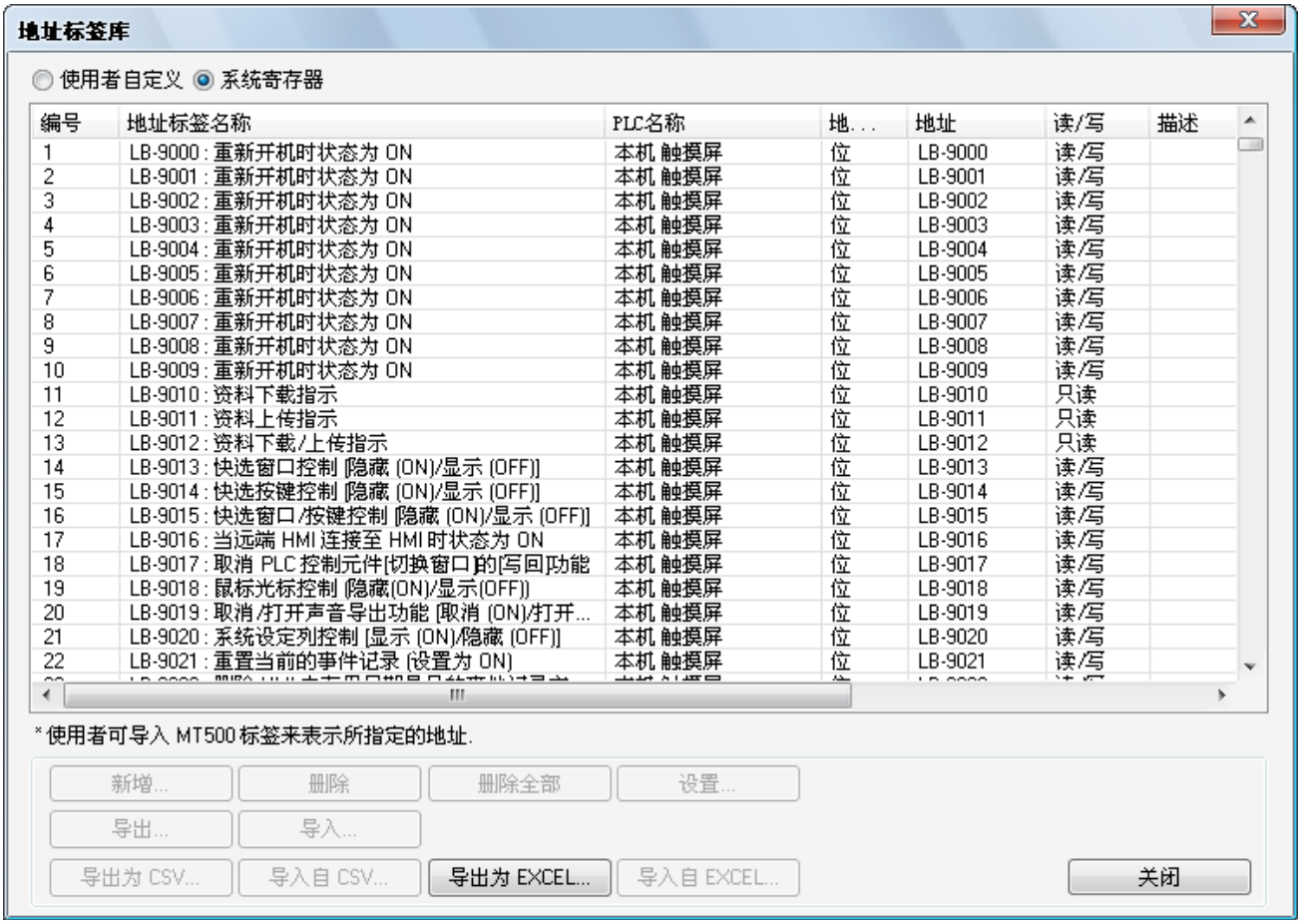
Note

- cMT-SVR 系列的远端 HMI 限制只能为 cMT-SVR 系列。故无法与 eMT/cMT-HD 系列上的 PLC 做通讯。

第二十二章 系统寄存器

22.1 概要

EasyBuilder Pro 工程软件保留了一些位地址和字地址的寄存器供给系统使用，这些系统保留寄存器分别有不同的功用，而我们将系统保留寄存器地址分类如下。



22.2 本机触摸屏内存地址范围

22.2.1 位地址

寄存器	设备类型	范围	格式
本机位地址	LB	0 ~ 12095	DDDDD
本机字地址取位地址	LW_BIT	0 ~ 1079915	DDDDDdd DDDDD: 地址 dd: 位地址 (00 ~ 15)
配方寄存器的位地址索引偏移量	RBI	0 ~ 65535f	DDDDDh DDDDD: 地址 h: 位地址 (0 ~ f) 通过 LW-9000 来当作索引寄存器，并对应到 RW_Bit
配方寄存器 RW 的位地址	RW_Bit	0 ~ 524287f	DDDDDh DDDDD: 地址 h: 位地址 (0 ~ f)
配方寄存器 RW_A 的位地址	RW_A_Bit	0 ~ 65535f	DDDDDh DDDDD: 地址 h: 位地址 (0 ~ f)

22.2.2 字地址

寄存器	设备类型	范围	格式
本机字地址	LW	0 ~ 10799	DDDDD
配方寄存器 RW	RW	0 ~ 524287	DDDDDD
配方寄存器的字地址索引偏移量	RWI	0 ~ 65535	DDDDD 通过 LW-9000 来当作索引寄存器，并对应到 RW
配方寄存器 RW_A	RW_A	0 ~ 65535	DDDDD
扩展内存寄存器	EM0 ~ EM9	0 ~ 1073741823	DDDDDDDDDD

22.3 HMI 时间

地址	描述	读 / 写 / 控制		
		本地 HMI	宏	远程 HMI
LB-11958	时间设定错误 (当状态为 ON) *注 3	读	读	读
LW-9010	(16bit-BCD)：本地时间 (秒)	读/写	读/控制	读/控制
LW-9011	(16bit-BCD)：本地时间 (分)	读/写	读/控制	读/控制
LW-9012	(16bit-BCD)：本地时间 (时)	读/写	读/控制	读/控制
LW-9013	(16bit-BCD)：本地时间 (日)	读/写	读/控制	读/控制
LW-9014	(16bit-BCD)：本地时间 (月)	读/写	读/控制	读/控制
LW-9015	(16bit-BCD)：本地时间 (年)	读/写	读/控制	读/控制
LW-9016	(16bit-BCD)：本地时间 (星期)	读	读	读
LW-9017	(16bit)：本地时间 (秒)	读/写	读/控制	读/控制
LW-9018	(16bit)：本地时间 (分)	读/写	读/控制	读/控制
LW-9019	(16bit)：本地时间 (时)	读/写	读/控制	读/控制
LW-9020	(16bit)：本地时间 (日)	读/写	读/控制	读/控制
LW-9021	(16bit)：本地时间 (月)	读/写	读/控制	读/控制
LW-9022	(16bit)：本地时间 (年) *注 1	读/写	读/控制	读/控制
LW-9023	(16bit)：本地时间 (星期) *注 2	读	读	读
LW-9030	(32bit)：系统时间 (单位 ：0.1 秒)	读	读	读
LW-9048	(16bit)：时间 (0：AM, 1：PM)	读/写	读/控制	读/控制
LW-9049	(16bit)：本地时间 (12 小时制)	读/写	读/控制	读/控制

Note

1. 数值范围为 2000 ~ 2037。
2. 数值范围为 1 ~ 7，即为星期一 ~ 星期日。
3. 当通过 LW-9010 ~ LW-9023 更新触摸屏时间时，系统将比对 RTC 内的时间，以确认时间是否修改成功。若不成功，系统会将时间回复为设定前的时间，并将 LB-11958 设定为 ON。若使用 LW-9010 ~ LW-9023 于电脑 模拟时修改时间，将无法作用。

22.4 用户名称及密码

地址	描述	读 / 写 / 控制		
		本地 HMI	宏	远程 HMI
LB-9050	用户注销	写	控制	控制
LB-9060	密码输入错误指示	读	读	读
LB-9061	更新密码 (设定为 ON)	写	控制	控制
LW-9082	(16bit) : 自动注销时间 (单位 : 分钟, 0 : 取消此功能)	读/写	读/控制	读/控制
LW-9219	(16bit) : 用户编号 (1 ~ 12)	读/写	读/控制	读/控制
LW-9220	(32bit) : 用户密码	读/写	读/控制	读/控制
LW-9222	(16bit) : 目前用户可使用的元件类别 (bit 0:A, bit 1:B,bit 2:C, ...)	读	读	读
LW-9500	(32bit) : 用户 1 的密码	读/写	读/控制	读/控制
LW-9502	(32bit) : 用户 2 的密码	读/写	读/控制	读/控制
LW-9504	(32bit) : 用户 3 的密码	读/写	读/控制	读/控制
LW-9506	(32bit) : 用户 4 的密码	读/写	读/控制	读/控制
LW-9508	(32bit) : 用户 5 的密码	读/写	读/控制	读/控制
LW-9510	(32bit) : 用户 6 的密码	读/写	读/控制	读/控制
LW-9512	(32bit) : 用户 7 的密码	读/写	读/控制	读/控制
LW-9514	(32bit) : 用户 8 的密码	读/写	读/控制	读/控制
LW-9516	(32bit) : 用户 9 的密码	读/写	读/控制	读/控制
LW-9518	(32bit) : 用户 10 的密码	读/写	读/控制	读/控制
LW-9520	(32bit) : 用户 11 的密码	读/写	读/控制	读/控制
LW-9522	(32bit) : 用户 12 的密码	读/写	读/控制	读/控制
LW-10754	(8 words) : 目前登入的用户名称 *注 1	读	读	读

Note

1. 只支持于 “用户密码” » “进阶安全模式”。

22.5 资料取样

地址	描述	读 / 写 / 控制		
		本地 HMI	宏	远程 HMI
LB-9025	删除触摸屏内存里日期最早的资料取样文件 (设定为 ON)	写	控制	控制
LB-9026	删除触摸屏内存里全部资料取样文件 (设定为 ON)	写	控制	控制
LB-9027	更新触摸屏内存里资料取样统计信息 (设定为 ON)	写	控制	控制
LB-9034	强迫储存事件记录与取样数据至 HMI, U 盘, SD 卡 (设定为 ON)	写	控制	控制
LB-11949	删除 SD 卡里日期最早的资料取样文件 (设定为 ON)	写	控制	控制
LB-11950	删除 SD 卡里全部资料取样文件 (设定为 ON)	写	控制	控制
LB-11951	更新 SD 卡里资料取样统计信息 (设定为 ON)	写	控制	控制
LB-11952	删除 U 盘里日期最早的资料取样文件 (设定为 ON)	写	控制	控制
LB-11953	删除 U 盘里全部资料取样文件 (设定为 ON)	写	控制	控制
LB-11954	更新 U 盘里资料取样统计信息 (设定为 ON)	写	控制	控制
LW-9063	(16bit) :触摸屏内存里存在的资料取样文件的数目	读	读	读
LW-9064	(32bit) :触摸屏内存里存在的资料取样文件的大小	读	读	读
LW-10489	(16bit) : SD 卡里存在的资料取样文件的数目	读	读	读
LW-10490	(32bit) : SD 卡里存在的资料取样文件的大小	读	读	读
LW-10492	(16bit) : U 盘里存在的资料取样文件的数目	读	读	读
LW-10493	(32bit) : U 盘里存在的资料取样文件的大小	读	读	读

Note

1. 删除或更新资料取样的相关寄存器，在电脑模拟仿真时皆无作用。

22.6 事件登录

地址	描述	读 / 写 / 控制		
		本地 HMI	宏	远程 HMI
LB-9021	重置目前的事件记录 (设定为 ON)	写	控制	控制
LB-9022	删除触摸屏内存里日期最早的事件记录文件 (设定为 ON)	写	控制	控制
LB-9023	删除触摸屏内存里全部事件记录文件 (设定为 ON)	写	控制	控制
LB-9024	更新触摸屏内存里事件记录统计信息 (设定为 ON)	写	控制	控制
LB-9034	强迫储存事件记录与取样数据至 HMI, U 盘, SD 卡 (设定为 ON)	写	控制	控制
LB-9042	确认全部事件 (设定为 ON)	写	控制	控制
LB-9043	存在未确认的事件 (当状态为 ON)	读	读	读
LB-11940	删除 SD 卡里日期最早的事件记录文件 (设定为 ON)	写	控制	控制
LB-11941	删除 SD 卡里全部事件记录文件 (设定为 ON)	写	控制	控制
LB-11942	更新 SD 卡里事件记录统计信息 (设定为 ON)	写	控制	控制
LB-11943	删除 U 盘里日期最早的事件记录文件 (设定为 ON)	写	控制	控制
LB-11944	删除 U 盘里全部事件记录文件 (设定为 ON)	写	控制	控制
LB-11945	更新 U 盘里事件记录统计信息 (设定为 ON)	写	控制	控制
LW-9060	(16bit) :触摸屏内存里存在的事件记录文件的数目	读	读	读
LW-9061	(32bit) :触摸屏内存里存在的事件记录文件的大小	读	读	读
LW-9450	(16bit) : 事件登录的时间标签 – 秒 *注 1	读/写	读/控制	读/控制
LW-9451	(16bit) : 事件登录的时间标签 – 分 *注 1	读/写	读/控制	读/控制
LW-9452	(16bit) : 事件登录的时间标签 – 时 *注 1	读/写	读/控制	读/控制
LW-9453	(16bit) : 事件登录的时间标签 – 日 *注 1	读/写	读/控制	读/控制
LW-9454	(16bit) : 事件登录的时间标签 – 月 *注 1	读/写	读/控制	读/控制
LW-9455	(16bit) : 事件登录的时间标签 – 年 *注 1	读/写	读/控制	读/控制
LW-10480	(16bit) : SD 卡里存在的事件记录文件的数目	读	读	读
LW-10481	(32bit) : SD 卡里存在的事件记录文件的大小	读	读	读
LW-10483	(16bit) : U 盘里存在的事件记录文件的数目	读	读	读
LW-10484	(32bit) : U 盘里存在的事件记录文件的大小	读	读	读

Note

1. 若欲使用 LW-9450 ~ LW-9455 作为事件登录的时间来源标签，请先在“系统参数设置” »“一般属性”页面设定相关属性。
2. 删除或更新事件记录的相关寄存器，在电脑模拟仿真时皆无作用。

22.7 HMI 硬件操作

地址	描述	读 / 写 / 控制		
		本地 HMI	宏	远程 HMI
LB-9018	鼠标光标控制“隐藏 (ON)/显示 (OFF)”	读/写	读/控制	读/控制
LB-9019	声音输出功能“取消 (ON)/开启 (OFF)”	读/写	读/控制	读/控制
LB-9020	系统设定列控制“显示 (ON)/隐藏 (OFF)”	读/写	读/控制	读/控制
LB-9033	HMI 上传功能“取消 (ON)/开启 (OFF)” *注 1	读/写	读/控制	读
LB-9040	背光灯调亮 (设定为 ON) *注 2	写	控制	控制
LB-9041	背光灯调暗 (设定为 ON) *注 2	写	控制	控制
LB-9047	重新启动触摸屏 (设定为 ON，并当 LB-9048 状态为 ON 时)	写	控制	控制
LB-9048	重启机制保护	读/写	读/控制	读/控制
LB-9062	开启硬件设定对话框 (设定为 ON)	写	控制	控制
LB-9063	当插上 U 盘时，弹出下载窗口控制“隐藏 (ON)/显示 (OFF)”	读/写	读/控制	读/控制
LB-11959	LED 指示灯控制 *注 4	读/写	读/控制	读/控制
LW-9008	(32bit-float)：电池电压 *注 3	读	读	读
LW-9025	(16bit)：CPU 使用率	读	读	读
LW-9026	(16bit)：OS 版本 (年)	读	读	读
LW-9027	(16bit)：OS 版本 (月)	读	读	读
LW-9028	(16bit)：OS 版本 (日)	读	读	读
LW-9040	(16bit)：背光灯目前亮度值 *注 2	读	读	读
LW-9080	(16bit)：背光节能时间 (单位 ：分钟)	读/写	读/控制	读/控制
LW-9081	(16bit)：屏幕保护时间 (单位 ：分钟)	读/写	读/控制	读/控制

Note

1. 当变更设定时，需重启触摸屏以更新设定值。
2. LW-9040 可搭配 LB-9040 ~ LB-9041 来调整背光亮度，亮度值为 0 ~ 31。
3. 建议当 LW-9008 的电池电压值少于 2.80V 时，可更换电池。

4. 可用于辨识 cMT-HD 与 cMT-SVR 的设备。当有多台 cMT-HD 或 cMT-SVR 时，可触发此地址让 LED 灯的状态闪烁。

22.8 本地触摸屏网络信息

地址	描述	读 / 写 / 控制		
		本地 HMI	宏	远程 HMI
LB-12094	更新以太网 1 设定 (IP, 屏蔽, 网关) (设 ON)	读/写	读/控制	读/控制
LB-12095	更新以太网 2 设定 (IP, 屏蔽, 网关) (设 ON)	读/写	读/控制	读/控制
LW-9125	(16bit) :触摸屏以太网 1 所使用的网关 0 (只在触摸屏上有效)	读/写	读/控制	读/控制
LW-9126	(16bit) :触摸屏以太网 1 所使用的网关 1 (只在触摸屏上有效)	读/写	读/控制	读/控制
LW-9127	(16bit) :触摸屏以太网 1 所使用的网关 2 (只在触摸屏上有效)	读/写	读/控制	读/控制
LW-9128	(16bit) :触摸屏以太网 1 所使用的网关 3 (只在触摸屏 上有效)	读/写	读/控制	读/控制
LW-9129	(16bit) :触摸屏以太网 1 所使用的 IP 0 (只在触摸屏上有效)	读/写	读/控制	读/控制
LW-9130	(16bit) :触摸屏以太网 1 所使用的 IP 1 (只在触摸屏上有效)	读/写	读/控制	读/控制
LW-9131	(16bit) :触摸屏以太网 1 所使用的 IP 2 (只在触摸屏上有效)	读/写	读/控制	读/控制
LW-9132	(16bit) :触摸屏以太网 1 所使用的 IP 3 (只在触摸屏上有效)	读/写	读/控制	读/控制
LW-9133	(16bit) ： 以太网所使用的连接端口 (只在触摸屏上有效)	读	读	读
LW-9135	(16bit) ： 以太网 1 实体地址 (MAC) 0	读	读	读
LW-9136	(16bit) ： 以太网 1 实体地址 (MAC) 1	读	读	读
LW-9137	(16bit) ： 以太网 1 实体地址 (MAC) 2	读	读	读
LW-9138	(16bit) ： 以太网 1 实体地址 (MAC) 3	读	读	读
LW-9139	(16bit) ： 以太网 1 实体地址 (MAC) 4	读	读	读
LW-9140	(16bit) ： 以太网 1 实体地址 (MAC) 5	读	读	读
LW-10750	(16bit) :触摸屏以太网 1 所使用的屏蔽 0 (只在触摸屏上有效)	读/写	读/控制	读/控制

LW-10751	(16bit) :触摸屏以太网 1 所使用的屏蔽 1 (只在触摸屏上有效)	读/写	读/控制	读/控制
LW-10752	(16bit) :触摸屏以太网 1 所使用的屏蔽 2 (只在触摸屏上有效)	读/写	读/控制	读/控制
LW-10753	(16bit) :触摸屏以太网 1 所使用的屏蔽 3 (只在触摸屏上有效)	读/写	读/控制	读/控制
LW-10786	(16bit) :触摸屏以太网 2 所使用的 IP 0 (只在触摸屏上有效)	读	读	读
LW-10787	(16bit) :触摸屏以太网 2 所使用的 IP 1 (只在触摸屏上有效)	读	读	读
LW-10788	(16bit) :触摸屏以太网 2 所使用的 IP 2 (只在触摸屏上有效)	读	读	读
LW-10789	(16bit) :触摸屏以太网 2 所使用的 IP 3 (只在触摸屏上有效)	读	读	读
LW-10790	(16bit) :触摸屏以太网 2 所使用的屏蔽 0 (只在触摸屏上有效)	读	读	读
LW-10791	(16bit) :触摸屏以太网 2 所使用的屏蔽 1 (只在触摸屏上有效)	读	读	读
LW-10792	(16bit) :触摸屏以太网 2 所使用的屏蔽 2 (只在触摸屏上有效)	读	读	读
LW-10793	(16bit) :触摸屏以太网 2 所使用的屏蔽 3 (只在触摸屏上有效)	读	读	读
LW-10794	(16bit) :触摸屏以太网 2 所使用的网关 0 (只在触摸屏上有效)	读	读	读
LW-10795	(16bit) :触摸屏以太网 2 所使用的网关 1 (只在触摸屏上有效)	读	读	读
LW-10796	(16bit) :触摸屏以太网 2 所使用的网关 2 (只在触摸屏上有效)	读	读	读
LW-10797	(16bit) :触摸屏以太网 2 所使用的网关 3 (只在触摸屏 上有效)	读	读	读
LW-10798	(16bit) : 以太网 2 实体地址 (MAC) 0	读	读	读
LW-10799	(16bit) : 以太网 2 实体地址 (MAC) 1	读	读	读
LW-10800	(16bit) : 以太网 2 实体地址 (MAC) 2	读	读	读
LW-10801	(16bit) : 以太网 2 实体地址 (MAC) 3	读	读	读
LW-10802	(16bit) : 以太网 2 实体地址 (MAC) 4	读	读	读
LW-10803	(16bit) : 以太网 2 实体地址 (MAC) 5	读	读	读
LW-10804	(16bit) : 以太网 1 网域名称系统 (DNS) 服务器	读	读	读

	IP0			
LW-10805	(16bit) : 以太网 1 网域名称系统 (DNS) 服务器 IP1	读	读	读
LW-10806	(16bit) : 以太网 1 网域名称系统 (DNS) 服务器 IP2	读	读	读
LW-10807	(16bit) : 以太网 1 网域名称系统 (DNS) 服务器 IP3	读	读	读
LW-10808	(16bit) : 以太网 2 网域名称系统 (DNS) 服务器 IP0	读	读	读
LW-10809	(16bit) : 以太网 2 网域名称系统 (DNS) 服务器 IP1	读	读	读
LW-10810	(16bit) : 以太网 2 网域名称系统 (DNS) 服务器 IP2	读	读	读
LW-10811	(16bit) : 以太网 2 网域名称系统 (DNS) 服务器 IP3	读	读	读

Note

1. 以太网 2 的相关寄存器只适用于 cMT-SVR 机型。

22.9 配方及扩展内存

地址	描述	读 / 写 / 控制		
		本地 HMI	宏	远程 HMI
LB-9028	重置配方数据 (设定为 ON)	写	控制	控制
LB-9029	强迫储存配方数据到触摸屏(设定为 ON)	写	控制	控制
LB-9460	EM0 的储存设备 (SD 卡) 不存在 (当状态为 ON)	读	读	读
LB-9461	EM1 的储存设备 (SD 卡) 不存在 (当状态为 ON)	读	读	读
LB-9462	EM2 的储存设备 (SD 卡) 不存在 (当状态为 ON)	读	读	读
LB-9463	EM3 的储存设备 (SD 卡) 不存在 (当状态为 ON)	读	读	读
LB-9464	EM4 的储存设备 (SD 卡) 不存在 (当状态为 ON)	读	读	读
LB-9465	EM5 的储存设备 (SD 卡) 不存在 (当状态为 ON)	读	读	读
LB-9466	EM6 的储存设备 (SD 卡) 不存在 (当状态为 ON)	读	读	读
LB-9467	EM7 的储存设备 (SD 卡) 不存在 (当状态为 ON)	读	读	读
LB-9468	EM8 的储存设备 (SD 卡) 不存在 (当状态为 ON)	读	读	读

LB-9469	EM9 的储存设备 (SD 卡) 不存在 (当状态为 ON)	读	读	读
LB-9470	EM0 的储存设备 (U 盘) 不存在 (当状态为 ON)	读	读	读
LB-9471	EM1 的储存设备 (U 盘) 不存在 (当状态为 ON)	读	读	读
LB-9472	EM2 的储存设备 (U 盘) 不存在 (当状态为 ON)	读	读	读
LB-9473	EM3 的储存设备 (U 盘) 不存在 (当状态为 ON)	读	读	读
LB-9474	EM4 的储存设备 (U 盘) 不存在 (当状态为 ON)	读	读	读
LB-9475	EM5 的储存设备 (U 盘) 不存在 (当状态为 ON)	读	读	读
LB-9476	EM6 的储存设备 (U 盘) 不存在 (当状态为 ON)	读	读	读
LB-9477	EM7 的储存设备 (U 盘) 不存在 (当状态为 ON)	读	读	读
LB-9478	EM8 的储存设备 (U 盘) 不存在 (当状态为 ON)	读	读	读
LB-9479	EM9 的储存设备 (U 盘) 不存在 (当状态为 ON)	读	读	读

22.10 内存储存空间管理

地址	描述	读 / 写 / 控制		
		本地 HMI	宏	远程 HMI
LB-9035	HMI 可用空间不足警示 (当状态为 ON)	读	读	读
LB-9036	SD 卡可用空间不足警示 (当状态为 ON)	读	读	读
LB-9037	U 盘可用空间不足警示 (当状态为 ON)	读	读	读
LB-12048	U 盘状态 (当状态为 ON 时表示存在)	读	读	读
LW-9070	(16bit) : 可用空间警示下限 (Mega bytes)	读	读	读
LW-9071	(16bit) : 系统保留的可用空间 (Mega bytes)	读	读	读
LW-9072	(32bit) :触摸屏目前的可用空间 (K bytes)	读	读	读
LW-9074	(32bit) : SD 卡目前的可用空间 (K bytes)	读	读	读
LW-9076	(32bit) : U 盘目前的可用空间 (K bytes)	读	读	读

22.11 触碰位置

地址	描述	读 / 写 / 控制		
		本地 HMI	宏	远程 HMI
LW-9041	(16bit) : 触控状态 (位 0 on = 正在触碰屏幕)	读	读	读
LW-9042	(16bit) : 触碰时, X 的位置	读	读	读
LW-9043	(16bit) : 触碰时, Y 的位置	读	读	读

LW-9044	(16bit)：离开时, X 的位置	读	读	读
LW-9045	(16bit)：离开时, Y 的位置	读	读	读

22.12 站号变数

地址	描述	读 / 写 / 控制		
		本地 HMI	宏	远程 HMI
LW-10000	(16bit) : var0 – 站号变量 (语法 : var0#地址)	读/写	读/控制	读/控制
LW-10001	(16bit) : var1 – 站号变量 (语法 : var1#地址)	读/写	读/控制	读/控制
LW-10002	(16bit) : var2 – 站号变量 (语法 : var2#地址)	读/写	读/控制	读/控制
LW-10003	(16bit) : var3 – 站号变量 (语法 : var3#地址)	读/写	读/控制	读/控制
LW-10004	(16bit) : var4 – 站号变量 (语法 : var4#地址)	读/写	读/控制	读/控制
LW-10005	(16bit) : var5 – 站号变量 (语法 : var5#地址)	读/写	读/控制	读/控制
LW-10006	(16bit) : var6 – 站号变量 (语法 : var6#地址)	读/写	读/控制	读/控制
LW-10007	(16bit) : var7 – 站号变量 (语法 : var7#地址)	读/写	读/控制	读/控制
LW-10008	(16bit) : var8 – 站号变量 (语法 : var8#地址)	读/写	读/控制	读/控制
LW-10009	(16bit) : var9 – 站号变量 (语法 : var9#地址)	读/写	读/控制	读/控制
LW-10010	(16bit) : var10 – 站号变量 (语法 : var10#地址)	读/写	读/控制	读/控制
LW-10011	(16bit) : var11 – 站号变量 (语法 : var11#地址)	读/写	读/控制	读/控制
LW-10012	(16bit) : var12 – 站号变量 (语法 : var12#地址)	读/写	读/控制	读/控制
LW-10013	(16bit) : var13 – 站号变量 (语法 : var13#地址)	读/写	读/控制	读/控制
LW-10014	(16bit) : var14 – 站号变量 (语法 : var14#地址)	读/写	读/控制	读/控制
LW-10015	(16bit) : var15 – 站号变量 (语法 : var15#地址)	读/写	读/控制	读/控制

22.13 索引寄存器

地址	描述	读 / 写 / 控制		
		本地 HMI	宏	远程 HMI
LW-9200	(16bit)：地址索引寄存器 0	读/写	读/控制	读/控制
LW-9201	(16bit)：地址索引寄存器 1	读/写	读/控制	读/控制
LW-9202	(16bit)：地址索引寄存器 2	读/写	读/控制	读/控制
LW-9203	(16bit)：地址索引寄存器 3	读/写	读/控制	读/控制
LW-9204	(16bit)：地址索引寄存器 4	读/写	读/控制	读/控制
LW-9205	(16bit)：地址索引寄存器 5	读/写	读/控制	读/控制
LW-9206	(16bit)：地址索引寄存器 6	读/写	读/控制	读/控制
LW-9207	(16bit)：地址索引寄存器 7	读/写	读/控制	读/控制
LW-9208	(16bit)：地址索引寄存器 8	读/写	读/控制	读/控制
LW-9209	(16bit)：地址索引寄存器 9	读/写	读/控制	读/控制
LW-9210	(16bit)：地址索引寄存器 10	读/写	读/控制	读/控制
LW-9211	(16bit)：地址索引寄存器 11	读/写	读/控制	读/控制
LW-9212	(16bit)：地址索引寄存器 12	读/写	读/控制	读/控制
LW-9213	(16bit)：地址索引寄存器 13	读/写	读/控制	读/控制
LW-9214	(16bit)：地址索引寄存器 14	读/写	读/控制	读/控制
LW-9215	(16bit)：地址索引寄存器 15	读/写	读/控制	读/控制
LW-9230	(32bit)：地址索引寄存器 16	读/写	读/控制	读/控制
LW-9232	(32bit)：地址索引寄存器 17	读/写	读/控制	读/控制
LW-9234	(32bit)：地址索引寄存器 18	读/写	读/控制	读/控制
LW-9236	(32bit)：地址索引寄存器 19	读/写	读/控制	读/控制
LW-9238	(32bit)：地址索引寄存器 20	读/写	读/控制	读/控制
LW-9240	(32bit)：地址索引寄存器 21	读/写	读/控制	读/控制
LW-9242	(32bit)：地址索引寄存器 22	读/写	读/控制	读/控制
LW-9244	(32bit)：地址索引寄存器 23	读/写	读/控制	读/控制
LW-9246	(32bit)：地址索引寄存器 24	读/写	读/控制	读/控制
LW-9248	(32bit)：地址索引寄存器 25	读/写	读/控制	读/控制
LW-9250	(32bit)：地址索引寄存器 26	读/写	读/控制	读/控制
LW-9252	(32bit)：地址索引寄存器 27	读/写	读/控制	读/控制
LW-9254	(32bit)：地址索引寄存器 28	读/写	读/控制	读/控制
LW-9256	(32bit)：地址索引寄存器 29	读/写	读/控制	读/控制

LW-9258	(32bit)：地址索引寄存器 30	读/写	读/控制	读/控制
LW-9260	(32bit)：地址索引寄存器 31	读/写	读/控制	读/控制

22.14 工程文件信息

地址	描述	读 / 写 / 控制		
		本地 HMI	宏	远程 HMI
LW-9100	(16bit)：工程文件的名称 (16 字符)	读	读	读
LW-9116	(32bit)：工程文件的大小 (单位 ：byte)	读	读	读
LW-9118	(32bit)：工程文件的大小 (单位 ：K bytes)	读	读	读
LW-9120	(32bit)：工程文件所使用的编译器版本	读	读	读
LW-9122	(16bit)：工程文件编译时间 (年)	读	读	读
LW-9123	(16bit)：工程文件编译时间 (月)	读	读	读
LW-9124	(16bit)：工程文件编译时间 (日)	读	读	读

22.15 MODBUS Server 通讯

地址	描述	读 / 写 / 控制		
		本地 HMI	宏	远程 HMI
LB-9055	MODBUS server (COM 1) 接收到合法的命令 (当状态为 ON)	读	读	读
LB-9056	MODBUS server (COM 2) 接收到合法的命令 (当状态为 ON)	读	读	读
LB-9057	MODBUS server (COM 3) 接收到合法的命令 (当状态为 ON)	读	读	读
LB-9058	MODBUS server (以太网) 接收到合法的命令 (当状态为 ON)	读	读	读
LW-9270	(16bit)：请求的功能码 – MODBUS server (COM 1)	读	读	读
LW-9271	(16bit)：请求的开始地址 – MODBUS server (COM 1)	读	读	读
LW-9272	(16bit)：请求的地址数目 – MODBUS server (COM 1)	读	读	读
LW-9275	(16bit)：请求的功能码 – MODBUS server (COM 2)	读	读	读

LW-9276	(16bit)：请求的开始地址 – MODBUS server (COM 2)	读	读	读
LW-9277	(16bit)：请求的地址数目 – MODBUS server (COM 2)	读	读	读
LW-9280	(16bit)：请求的功能码 – MODBUS server (COM 3)	读	读	读
LW-9281	(16bit)：请求的开始地址 – MODBUS server (COM 3)	读	读	读
LW-9282	(16bit)：请求的地址数目 – MODBUS server (COM 3)	读	读	读
LW-9285	(16bit)：请求的功能码 – MODBUS server (以太网)	读	读	读
LW-9286	(16bit)：请求的开始地址 – MODBUS server (以太网)	读	读	读
LW-9287	(16bit)：请求的地址数目 – MODBUS server (以太网)	读	读	读
LW-9288	(16bit)：最后通讯错误码 – MODBUS server (以太网)	读	读	读
LW-9541	(16bit)：MODBUS/ASCII server 站号 (COM 1)	读/写	读/控制	读/控制
LW-9542	(16bit)：MODBUS/ASCII server 站号 (COM 2)	读/写	读/控制	读/控制
LW-9543	(16bit)：MODBUS/ASCII server 站号 (COM 3)	读/写	读/控制	读/控制
LW-9544	(16bit)：MODBUS/ASCII server 站号 (以太网)	读/写	读/控制	读/控制
LW-9570	(32bit)：已接收的数据 (bytes) (COM 1 MODBUS server)	读	读	读
LW-9572	(32bit)：已接收的数据 (bytes) (COM 2 MODBUS server)	读	读	读
LW-9574	(32bit)：已接收的数据 (bytes) (COM 3 MODBUS server)	读	读	读
LW-9576	(32bit)：已接收的数据 (bytes) (以太网 MODBUS server)	读	读	读

22.16 通讯参数设定

地址	描述	读 / 写 / 控制		
		本地 HMI	宏	远程 HMI
LB-9030	更新 COM 1 通讯参数 (设定为 ON)	读/写	读/控制	读/控制
LB-9031	更新 COM 2 通讯参数 (设定为 ON)	读/写	读/控制	读/控制
LB-9032	更新 COM 3 通讯参数 (设定为 ON)	读/写	读/控制	读/控制
LB-9065	停用/启用 COM 1 广播站号	读/写	读/控制	读/控制
LB-9066	停用/启用 COM 2 广播站号	读/写	读/控制	读/控制
LB-9067	停用/启用 COM 3 广播站号	读/写	读/控制	读/控制
LW-9550	(16bit) : COM 1 模式 (0:RS232,1:RS485 2W,2:RS485 4W)	读/写	读/控制	读/控制
LW-9551	(16bit) : COM 1 波特率 (7:1200,8:2400,0:4800,1:9600,10:14400,2:19200, 11:28800,3:38400,4:57600,..)	读/写	读/控制	读/控制
LW-9552	(16bit) : COM 1 数据位 (7 : 7 bits, 8 : 8 bits)	读/写	读/控制	读/控制
LW-9553	(16bit) : COM 1 校验 (0:none, 1:even, 2:odd, 3:mark, 4:space)	读/写	读/控制	读/控制
LW-9554	(16bit) : COM 1 停止位 (1 : 1 bit, 2 : 2 bits)	读/写	读/控制	读/控制
LW-9555	(16bit) : COM 2 模式 (0:RS232,1:RS485 2W,2:RS485 4W)	读/写	读/控制	读/控制
LW-9556	(16bit) : COM 2 波特率 (7:1200,8:2400,0:4800,1:9600,10:14400,2:19200, 11:28800,3:38400,4:57600,..)	读/写	读/控制	读/控制
LW-9557	(16bit) : COM 2 数据位 (7 : 7 bits, 8 : 8 bits)	读/写	读/控制	读/控制
LW-9558	(16bit) : COM 2 校验 (0:none, 1:even, 2:odd, 3:mark, 4:space)	读/写	读/控制	读/控制
LW-9559	(16bit) : COM 2 停止位 (1 : 1 bit, 2 : 2 bits)	读/写	读/控制	读/控制
LW-9560	(16bit) : COM 3 模式 (0:RS232, 1:RS485 2W)	读/写	读/控制	读/控制
LW-9561	(16bit) : COM 3 波特率 (7:1200,8:2400,0:4800,1:9600,10:14400,2:19200,	读/写	读/控制	读/控制

	11:28800,3:38400,4:57600,..)			
LW-9562	(16bit) : COM 3 数据位 (7 : 7 bits, 8 : 8 bits)	读/写	读/控制	读/控制
LW-9563	(16bit) : COM 3 校验 (0:none, 1:even, 2:odd, 3:mark, 4:space)	读/写	读/控制	读/控制
LW-9564	(16bit) : COM 3 停止位 (1 : 1 bit, 2 : 2 bits)	读/写	读/控制	读/控制
LW-9565	(16bit) : COM 1 广播站号	读/写	读/控制	读/控制
LW-9566	(16bit) : COM 2 广播站号	读/写	读/控制	读/控制
LW-9567	(16bit) : COM 3 广播站号	读/写	读/控制	读/控制
LW-10500	(16bit) : PLC 1 超时 (单位 : 100ms)	读/写	读/控制	读/控制
LW-10501	(16bit) : PLC 1 通讯延时 (单位 : ms)	读/写	读/控制	读/控制
LW-10502	(16bit) : PLC 1 ACK 讯号延时 (单位 : ms)	读/写	读/控制	读/控制
LW-10503	(16bit) : PLC 1 参数 1	读/写	读/控制	读/控制
LW-10504	(16bit) : PLC 1 参数 2	读/写	读/控制	读/控制
LW-10505	(16bit) : PLC 2 超时 (单位 : 100ms)	读/写	读/控制	读/控制
LW-10506	(16bit) : PLC 2 通讯延时 (单位 : ms)	读/写	读/控制	读/控制
LW-10507	(16bit) : PLC 2 ACK 讯号延时 (单位 : ms)	读/写	读/控制	读/控制
LW-10508	(16bit) : PLC 2 参数 1	读/写	读/控制	读/控制
LW-10509	(16bit) : PLC 2 参数 2	读/写	读/控制	读/控制
LW-10510	(16bit) : PLC 3 超时 (单位 : 100ms)	读/写	读/控制	读/控制
LW-10511	(16bit) : PLC 3 通讯延时 (单位 : ms)	读/写	读/控制	读/控制
LW-10512	(16bit) : PLC 3 ACK 讯号延时 (单位 : ms)	读/写	读/控制	读/控制
LW-10513	(16bit) : PLC 3 参数 1	读/写	读/控制	读/控制
LW-10514	(16bit) : PLC 3 参数 2	读/写	读/控制	读/控制
LW-10515	(16bit) : PLC 4 超时 (单位 : 100ms)	读/写	读/控制	读/控制
LW-10516	(16bit) : PLC 4 通讯延时 (单位 : ms)	读/写	读/控制	读/控制
LW-10517	(16bit) : PLC 4 ACK 讯号延时 (单位 : ms) (SIEMENS S7/400 连接类型)	读/写	读/控制	读/控制
LW-10518	(16bit) : PLC 4 参数 1 (SIEMENS S7/400 机座)	读/写	读/控制	读/控制
LW-10519	(16bit) : PLC 4 参数 2 (SIEMENS S7/400 CPU 插槽)	读/写	读/控制	读/控制
LW-10520	(16bit) : PLC 5 超时 (单位 : 100ms)	读/写	读/控制	读/控制
LW-10521	(16bit) : PLC 5 通讯延时 (单位 : ms)	读/写	读/控制	读/控制
LW-10522	(16bit) : PLC 5 ACK 讯号延时 (单位 : ms) (SIEMENS S7/400 连接类型)	读/写	读/控制	读/控制
LW-10523	(16bit) : PLC 5 参数 1 (SIEMENS S7/400 机座)	读/写	读/控制	读/控制
LW-10524	(16bit) : PLC 5 参数 2 (SIEMENS S7/400 CPU 插	读/写	读/控制	读/控制

	槽)			
LW-10525	(16bit) : PLC 6 超时 (单位 : 100ms)	读/写	读/控制	读/控制
LW-10526	(16bit) : PLC 6 通讯延时 (单位 : ms)	读/写	读/控制	读/控制
LW-10527	(16bit) : PLC 6 ACK 讯号延时 (单位 : ms) (SIEMENS S7/400 连接类型)	读/写	读/控制	读/控制
LW-10528	(16bit) : PLC 6 参数 1 (SIEMENS S7/400 机座)	读/写	读/控制	读/控制
LW-10529	(16bit) : PLC 6 参数 2 (SIEMENS S7/400 CPU 插槽)	读/写	读/控制	读/控制
LW-10530	(16bit) : PLC 7 超时 (单位 : 100ms)	读/写	读/控制	读/控制
LW-10531	(16bit) : PLC 7 通讯延时 (单位 : ms)	读/写	读/控制	读/控制
LW-10532	(16bit) : PLC 7 ACK 讯号延时 (单位 : ms) (SIEMENS S7/400 连接类型)	读/写	读/控制	读/控制
LW-10533	(16bit) : PLC 7 参数 1 (SIEMENS S7/400 机座)	读/写	读/控制	读/控制
LW-10534	(16bit) : PLC 7 参数 2 (SIEMENS S7/400 CPU 插槽)	读/写	读/控制	读/控制
LW-10535	(16bit) : PLC 8 超时 (单位 : 100ms)	读/写	读/控制	读/控制
LW-10536	(16bit) : PLC 8 通讯延时 (单位 : ms)	读/写	读/控制	读/控制
LW-10537	(16bit) : PLC 8 ACK 讯号延时 (单位 : ms) (SIEMENS S7/400 连接类型)	读/写	读/控制	读/控制
LW-10538	(16bit) : PLC 8 参数 1 (SIEMENS S7/400 机座)	读/写	读/控制	读/控制
LW-10539	(16bit) : PLC 8 参数 2 (SIEMENS S7/400 CPU 插槽)	读/写	读/控制	读/控制

22.17 与 PLC (COM) 通讯状态

地址	描述	读 / 写 / 控制		
		本地 HMI	宏	远程 HMI
LB-9150	自动连结 PLC 1 (COM 1) (当状态为 ON)	读/写	读/控制	读/控制
LB-9151	自动连结 PLC 2 (COM 2) (当状态为 ON)	读/写	读/控制	读/控制
LB-9152	自动连结 PLC 3 (COM 3) (当状态为 ON)	读/写	读/控制	读/控制
LB-9200	与 PLC 1 的通讯状态 (站号 0, COM 1), 设 ON 重连一次 *注 1	读/写	读/控制	读/控制
LB-9201	与 PLC 1 的通讯状态 (站号 1, COM 1), 设 ON 重连一次	读/写	读/控制	读/控制
LB-9202	与 PLC 1 的通讯状态 (站号 2, COM 1), 设 ON	读/写	读/控制	读/控制

	重连一次			
LB-9203	与 PLC 1 的通讯状态 (站号 3, COM 1), 设 ON 重连一次	读/写	读/控制	读/控制
LB-9204	与 PLC 1 的通讯状态 (站号 4, COM 1), 设 ON 重连一次	读/写	读/控制	读/控制
LB-9205	与 PLC 1 的通讯状态 (站号 5, COM 1), 设 ON 重连一次	读/写	读/控制	读/控制
LB-9206	与 PLC 1 的通讯状态 (站号 6, COM 1), 设 ON 重连一次	读/写	读/控制	读/控制
LB-9207	与 PLC 1 的通讯状态 (站号 7, COM 1), 设 ON 重连一次	读/写	读/控制	读/控制
LB-9500	与 PLC 2 的通讯状态 (站号 0, COM 2), 设 ON 重连一次 *注 2	读/写	读/控制	读/控制
LB-9501	与 PLC 2 的通讯状态 (站号 1, COM 2), 设 ON 重连一次	读/写	读/控制	读/控制
LB-9502	与 PLC 2 的通讯状态 (站号 2, COM 2), 设 ON 重连一次	读/写	读/控制	读/控制
LB-9503	与 PLC 2 的通讯状态 (站号 3, COM 2), 设 ON 重连一次	读/写	读/控制	读/控制
LB-9504	与 PLC 2 的通讯状态 (站号 4, COM 2), 设 ON 重连一次	读/写	读/控制	读/控制
LB-9505	与 PLC 2 的通讯状态 (站号 5, COM 2), 设 ON 重连一次	读/写	读/控制	读/控制
LB-9506	与 PLC 2 的通讯状态 (站号 6, COM 2), 设 ON 重连一次	读/写	读/控制	读/控制
LB-9507	与 PLC 2 的通讯状态 (站号 7, COM 2), 设 ON 重连一次	读/写	读/控制	读/控制
LB-9800	与 PLC 3 的通讯状态 (站号 0, COM 3), 设 ON 重连一次 *注 3	读/写	读/控制	读/控制
LB-9801	与 PLC 3 的通讯状态 (站号 1, COM 3), 设 ON 重连一次	读/写	读/控制	读/控制
LB-9802	与 PLC 3 的通讯状态 (站号 2, COM 3), 设 ON 重连一次	读/写	读/控制	读/控制
LB-9803	与 PLC 3 的通讯状态 (站号 3, COM 3), 设 ON 重连一次	读/写	读/控制	读/控制
LB-9804	与 PLC 3 的通讯状态 (站号 4, COM 3), 设 ON 重连一次	读/写	读/控制	读/控制

LB-9805	与 PLC 3 的通讯状态 (站号 5, COM 3), 设 ON 重连一次	读/写	读/控制	读/控制
LB-9806	与 PLC 3 的通讯状态 (站号 6, COM 3), 设 ON 重连一次	读/写	读/控制	读/控制
LB-9807	与 PLC 3 的通讯状态 (站号 7, COM 3), 设 ON 重连一次	读/写	读/控制	读/控制
LB-12030	COM 1 开启状态指示 (OFF：正常, ON：开启失败) *注 4	读	读	读
LB-12031	COM 2 开启状态指示 (OFF：正常, ON：开启失败)	读	读	读
LB-12032	COM 3 开启状态指示 (OFF：正常, ON：开启失败)	读	读	读
LB-12033	COM 4 开启状态指示 (OFF：正常, ON：开启失败)	读	读	读
LB-12034	COM 5 开启状态指示 (OFF：正常, ON：开启失败)	读	读	读
LB-12035	COM 6 开启状态指示 (OFF：正常, ON：开启失败)	读	读	读
LB-12036	COM 7 开启状态指示 (OFF：正常, ON：开启失败)	读	读	读
LB-12037	COM 8 开启状态指示 (OFF：正常, ON：开启失败)	读	读	读
LB-12038	COM 9 开启状态指示 (OFF：正常, ON：开启失败)	读	读	读

Note

1. LB-9200 COM 1 通讯状态的相关寄存器，可延伸使用到 LB-9455 (站号 255, COM 1)。
2. LB-9500 COM 2 通讯状态的相关寄存器，可延伸使用到 LB-9755 (站号 255, COM 2)。
3. LB-9800 COM 3 通讯状态的相关寄存器，可延伸使用到 LB-10055 (站号 255, COM 3)。
4. COM 的开启状态指示可使用于在电脑模拟仿真时，查看 COM 是否被其它程序占用。

22.18 与 PLC (以太网) 通讯状态

地址	描述	读 / 写 / 控制		
		本地 HMI	宏	远程 HMI
LB-9153	自动连结 PLC 4 (以太网) (当状态为 ON) *注 1	读/写	读/控制	读/控制
LB-9154	自动连结 PLC 5 (以太网) (当状态为 ON)	读/写	读/控制	读/控制
LB-9155	自动连结 PLC 6 (以太网) (当状态为 ON)	读/写	读/控制	读/控制
LB-9156	自动连结 PLC 7 (以太网) (当状态为 ON)	读/写	读/控制	读/控制
LB-9157	自动连结 PLC 8 (以太网) (当状态为 ON)	读/写	读/控制	读/控制
LB-9158	自动连结 PLC 9 (以太网) (当状态为 ON)	读/写	读/控制	读/控制
LB-10070	当在线更改 PLC 4 (以太网) 的 IP 或系统参数时, 设 ON 重新连结 PLC *注 2	读/写	读/控制	读/控制

LB-10071	当在线更改 PLC 5 (以太网) 的 IP 或系统参数时，设 ON 重新连结 PLC	读/写	读/控制	读/控制
LB-10072	当在线更改 PLC 6 (以太网) 的 IP 或系统参数时，设 ON 重新连结 PLC	读/写	读/控制	读/控制
LB-10073	当在线更改 PLC 7 (以太网) 的 IP 或系统参数时，设 ON 重新连结 PLC	读/写	读/控制	读/控制
LB-10074	当在线更改 PLC 8 (以太网) 的 IP 或系统参数时，设 ON 重新连结 PLC	读/写	读/控制	读/控制
LB-10075	当在线更改 PLC 9 (以太网) 的 IP 或系统参数时，设 ON 重新连结 PLC	读/写	读/控制	读/控制
LB-10100	与 PLC 4 的通讯状态 (以太网)，设 ON 重连一次 *注 3	读/写	读/控制	读/控制
LB-10400	与 PLC 5 的通讯状态 (以太网)，设 ON 重连一次	读/写	读/控制	读/控制
LB-10700	与 PLC 6 的通讯状态 (以太网)，设 ON 重连一次	读/写	读/控制	读/控制
LB-11000	与 PLC 7 的通讯状态 (以太网)，设 ON 重连一次	读/写	读/控制	读/控制
LB-11300	与 PLC 8 的通讯状态 (以太网)，设 ON 重连一次	读/写	读/控制	读/控制
LB-11600	与 PLC 9 的通讯状态 (以太网)，设 ON 重连一次	读/写	读/控制	读/控制
LB-11900	与 PLC 10 的通讯状态 (以太网)，设 ON 重连一次	读/写	读/控制	读/控制
LB-11901	与 PLC 11 的通讯状态 (以太网)，设 ON 重连一次	读/写	读/控制	读/控制
LB-11902	与 PLC 12 的通讯状态 (以太网)，设 ON 重连一次	读/写	读/控制	读/控制
LB-11903	与 PLC 13 的通讯状态 (以太网)，设 ON 重连一次	读/写	读/控制	读/控制
LB-11904	与 PLC 14 的通讯状态 (以太网)，设 ON 重连一次	读/写	读/控制	读/控制
LB-11905	与 PLC 15 的通讯状态 (以太网)，设 ON 重连一次	读/写	读/控制	读/控制
LB-11906	与 PLC 16 的通讯状态 (以太网)，设 ON 重连一次	读/写	读/控制	读/控制
LW-9600	(16bit) : PLC 4 的 IP0 (IP 地址 = IP0:IP1:IP2:IP3) *注 4	读/写	读/控制	读/控制
LW-9601	(16bit) : PLC 4 的 IP1 (IP 地址 = IP0:IP1:IP2:IP3)	读/写	读/控制	读/控制
LW-9602	(16bit) : PLC 4 的 IP2 (IP 地址 = IP0:IP1:IP2:IP3)	读/写	读/控制	读/控制
LW-9603	(16bit) : PLC 4 的 IP3 (IP 地址 = IP0:IP1:IP2:IP3)	读/写	读/控制	读/控制
LW-9604	(16bit) : PLC 4 的连接埠	读/写	读/控制	读/控制
LW-9605	(16bit) : PLC 5 的 IP0 (IP 地址 = IP0:IP1:IP2:IP3)	读/写	读/控制	读/控制
LW-9606	(16bit) : PLC 5 的 IP1 (IP 地址 =	读/写	读/控制	读/控制

	IP0:IP1:IP2:IP3)			
LW-9607	(16bit) : PLC 5 的 IP2 (IP 地址 = IP0:IP1:IP2:IP3)	读/写	读/控制	读/控制
LW-9608	(16bit) : PLC 5 的 IP3 (IP 地址 = IP0:IP1:IP2:IP3)	读/写	读/控制	读/控制
LW-9609	(16bit) : PLC 5 的连接埠	读/写	读/控制	读/控制
LW-9610	(16bit) : PLC 6 的 IP0 (IP 地址 = IP0:IP1:IP2:IP3)	读/写	读/控制	读/控制
LW-9611	(16bit) : PLC 6 的 IP1 (IP 地址 = IP0:IP1:IP2:IP3)	读/写	读/控制	读/控制
LW-9612	(16bit) : PLC 6 的 IP2 (IP 地址 = IP0:IP1:IP2:IP3)	读/写	读/控制	读/控制
LW-9613	(16bit) : PLC 6 的 IP3 (IP 地址 = IP0:IP1:IP2:IP3)	读/写	读/控制	读/控制
LW-9614	(16bit) : PLC 6 的连接埠	读/写	读/控制	读/控制
LW-9615	(16bit) : PLC 7 的 IP0 (IP 地址 = IP0:IP1:IP2:IP3)	读/写	读/控制	读/控制
LW-9616	(16bit) : PLC 7 的 IP1 (IP 地址 = IP0:IP1:IP2:IP3)	读/写	读/控制	读/控制
LW-9617	(16bit) : PLC 7 的 IP2 (IP 地址 = IP0:IP1:IP2:IP3)	读/写	读/控制	读/控制
LW-9618	(16bit) : PLC 7 的 IP3 (IP 地址 = IP0:IP1:IP2:IP3)	读/写	读/控制	读/控制
LW-9619	(16bit) : PLC 7 的连接埠	读/写	读/控制	读/控制
LW-9620	(16bit) : PLC 8 的 IP0 (IP 地址 = IP0:IP1:IP2:IP3)	读/写	读/控制	读/控制
LW-9621	(16bit) : PLC 8 的 IP1 (IP 地址 = IP0:IP1:IP2:IP3)	读/写	读/控制	读/控制
LW-9622	(16bit) : PLC 8 的 IP2 (IP 地址 = IP0:IP1:IP2:IP3)	读/写	读/控制	读/控制
LW-9623	(16bit) : PLC 8 的 IP3 (IP 地址 = IP0:IP1:IP2:IP3)	读/写	读/控制	读/控制
LW-9624	(16bit) : PLC 8 的连接埠	读/写	读/控制	读/控制
LW-9625	(16bit) : PLC 9 的 IP0 (IP 地址 = IP0:IP1:IP2:IP3)	读/写	读/控制	读/控制
LW-9626	(16bit) : PLC 9 的 IP1 (IP 地址 = IP0:IP1:IP2:IP3)	读/写	读/控制	读/控制

LW-9627	(16bit) : PLC 9 的 IP2 (IP 地址 = IP0:IP1:IP2:IP3)	读/写	读/控制	读/控制
LW-9628	(16bit) : PLC 9 的 IP3 (IP 地址 = IP0:IP1:IP2:IP3)	读/写	读/控制	读/控制
LW-9629	(16bit) : PLC 9 的连接埠	读/写	读/控制	读/控制

Note

1. LB-9153 自动连结 PLC 4 (以太网) 的相关寄存器，可延伸使用到 LB-9189 PLC 40。
2. LB-10070 在线重连 PLC 4 (以太网) 的相关寄存器，可延伸使用到 LB-10081 PLC 15。
3. LB-10100 PLC 4 (以太网) 通讯状态的相关寄存器，可延伸使用到 LB-11939 PLC 49。
4. LW-9600 PLC 4 (以太网) 地址设定的相关寄存器，可延伸使用到 LW-9769 PLC 37。

22.19 与 PLC (USB) 通讯状态

地址	描述	读 / 写 / 控制		
		本地 HMI	宏	远程 HMI
LB-9190	自动连结 PLC (USB) (当状态为 ON)	读/写	读/控制	读/控制
LB-9191	与 PLC 的通讯状态 (USB)，设 ON 重连一次	读/写	读/控制	读/控制

22.20 与 PLC (CAN Bus) 通讯状态

地址	描述	读 / 写 / 控制		
		本地 HMI	宏	远程 HMI
LB-12080	自动连结 PLC (CAN Bus) (当状态为 ON)	读/写	读/控制	读/控制
LB-12081	与 PLC 的通讯状态 (CAN Bus)，设 ON 重连一次	读/写	读/控制	读/控制
LB-12100	暂停 CAN Bus 设备 1 的通讯 (当状态为 ON) *注 1	读/写	读/控制	读/控制
LB-12101	暂停 CAN Bus 设备 2 的通讯 (当状态为 ON)	读/写	读/控制	读/控制
LB-12102	暂停 CAN Bus 设备 3 的通讯 (当状态为 ON)	读/写	读/控制	读/控制
LB-12103	暂停 CAN Bus 设备 4 的通讯 (当状态为 ON)	读/写	读/控制	读/控制
LB-12104	暂停 CAN Bus 设备 5 的通讯 (当状态为 ON)	读/写	读/控制	读/控制
LB-12105	暂停 CAN Bus 设备 6 的通讯 (当状态为 ON)	读/写	读/控制	读/控制
LB-12106	暂停 CAN Bus 设备 7 的通讯 (当状态为 ON)	读/写	读/控制	读/控制
LB-12107	暂停 CAN Bus 设备 8 的通讯 (当状态为 ON)	读/写	读/控制	读/控制
LB-12108	暂停 CAN Bus 设备 9 的通讯 (当状态为 ON)	读/写	读/控制	读/控制

LB-12109	暂停 CAN Bus 设备 10 的通讯 (当状态为 ON)	读/写	读/控制	读/控制

Note

1. LB-12100 暂停 CAN Bus 设备的通讯的相关寄存器，可延伸使用到 LB-12355 CAN Bus 设备 256。

22.21 与远程触摸屏通讯状态

地址	描述	读 / 写 / 控制		
		本地 HMI	宏	远程 HMI
LB-9068	自动连结远程触摸屏 1 (当状态为 ON) *注 1	读/写	读/控制	读/控制
LB-9069	自动连结远程触摸屏 2 (当状态为 ON)	读/写	读/控制	读/控制
LB-9070	自动连结远程触摸屏 3 (当状态为 ON)	读/写	读/控制	读/控制
LB-9071	自动连结远程触摸屏 4 (当状态为 ON)	读/写	读/控制	读/控制
LB-9072	自动连结远程触摸屏 5 (当状态为 ON)	读/写	读/控制	读/控制
LB-9073	自动连结远程触摸屏 6 (当状态为 ON)	读/写	读/控制	读/控制
LB-9074	自动连结远程触摸屏 7 (当状态为 ON)	读/写	读/控制	读/控制
LB-9075	自动连结远程触摸屏 8 (当状态为 ON)	读/写	读/控制	读/控制
LB-9100	与远程触摸屏 1 的通讯状态，设 ON 重连一次 *注 2	读/写	读/控制	读/控制
LB-9101	与远程触摸屏 2 的通讯状态，设 ON 重连一次	读/写	读/控制	读/控制
LB-9102	与远程触摸屏 3 的通讯状态，设 ON 重连一次	读/写	读/控制	读/控制
LB-9103	与远程触摸屏 4 的通讯状态，设 ON 重连一次	读/写	读/控制	读/控制
LB-9104	与远程触摸屏 5 的通讯状态，设 ON 重连一次	读/写	读/控制	读/控制
LB-9105	与远程触摸屏 6 的通讯状态，设 ON 重连一次	读/写	读/控制	读/控制
LB-9106	与远程触摸屏 7 的通讯状态，设 ON 重连一次	读/写	读/控制	读/控制
LB-9107	与远程触摸屏 8 的通讯状态，设 ON 重连一次	读/写	读/控制	读/控制
LB-9149	当在线更改远程触摸屏的 IP 时，设 ON 重新连结远程 HMI	读/写	读/控制	读/控制
LW-9800	(16bit) : 远程触摸屏 1 的 IP0 (IP 地址 = IP0:IP1:IP2:IP3) *注 3	读/写	读/控制	读/控制
LW-9801	(16bit) : 远程触摸屏 1 的 IP1 (IP 地址 = IP0:IP1:IP2:IP3)	读/写	读/控制	读/控制
LW-9802	(16bit) : 远程触摸屏 1 的 IP2 (IP 地址 = IP0:IP1:IP2:IP3)	读/写	读/控制	读/控制

LW-9803	(16bit)：远程触摸屏 1 的 IP3 (IP 地址 = IP0:IP1:IP2:IP3)	读/写	读/控制	读/控制
LW-9804	(16bit)：远程触摸屏 1 的连接埠	读/写	读/控制	读/控制
LW-9805	(16bit)：远程触摸屏 2 的 IP0 (IP 地址 = IP0:IP1:IP2:IP3)	读/写	读/控制	读/控制
LW-9806	(16bit)：远程触摸屏 2 的 IP1 (IP 地址 = IP0:IP1:IP2:IP3)	读/写	读/控制	读/控制
LW-9807	(16bit)：远程触摸屏 2 的 IP2 (IP 地址 = IP0:IP1:IP2:IP3)	读/写	读/控制	读/控制
LW-9808	(16bit)：远程触摸屏 2 的 IP3 (IP 地址 = IP0:IP1:IP2:IP3)	读/写	读/控制	读/控制
LW-9809	(16bit)：远程触摸屏 2 的连接埠	读/写	读/控制	读/控制
LW-9810	(16bit)：远程触摸屏 3 的 IP0 (IP 地址 = IP0:IP1:IP2:IP3)	读/写	读/控制	读/控制
LW-9811	(16bit)：远程触摸屏 3 的 IP1 (IP 地址 = IP0:IP1:IP2:IP3)	读/写	读/控制	读/控制
LW-9812	(16bit)：远程触摸屏 3 的 IP2 (IP 地址 = IP0:IP1:IP2:IP3)	读/写	读/控制	读/控制
LW-9813	(16bit)：远程触摸屏 3 的 IP3 (IP 地址 = IP0:IP1:IP2:IP3)	读/写	读/控制	读/控制
LW-9814	(16bit)：远程触摸屏 3 的连接埠	读/写	读/控制	读/控制
LW-9815	(16bit)：远程触摸屏 4 的 IP0 (IP 地址 = IP0:IP1:IP2:IP3)	读/写	读/控制	读/控制
LW-9816	(16bit)：远程触摸屏 4 的 IP1 (IP 地址 = IP0:IP1:IP2:IP3)	读/写	读/控制	读/控制
LW-9817	(16bit)：远程触摸屏 4 的 IP2 (IP 地址 = IP0:IP1:IP2:IP3)	读/写	读/控制	读/控制
LW-9818	(16bit)：远程触摸屏 4 的 IP3 (IP 地址 = IP0:IP1:IP2:IP3)	读/写	读/控制	读/控制
LW-9819	(16bit)：远程触摸屏 4 的连接埠	读/写	读/控制	读/控制
LW-9820	(16bit)：远程触摸屏 5 的 IP0 (IP 地址 = IP0:IP1:IP2:IP3)	读/写	读/控制	读/控制
LW-9821	(16bit)：远程触摸屏的 IP1 (IP 地址 = IP0:IP1:IP2:IP3)	读/写	读/控制	读/控制
LW-9822	(16bit)：远程触摸屏 5 的 IP2 (IP 地址 = IP0:IP1:IP2:IP3)	读/写	读/控制	读/控制
LW-9823	(16bit)：远程触摸屏 5 的 IP3 (IP 地址 =	读/写	读/控制	读/控制

	IP0:IP1:IP2:IP3)			
LW-9824	(16bit)：远程触摸屏 5 的连接埠	读/写	读/控制	读/控制
LW-9825	(16bit)：远程触摸屏 6 的 IP0 (IP 地址 = IP0:IP1:IP2:IP3)	读/写	读/控制	读/控制
LW-9826	(16bit)：远程触摸屏 6 的 IP1 (IP 地址 = IP0:IP1:IP2:IP3)	读/写	读/控制	读/控制
LW-9827	(16bit)：远程触摸屏 6 的 IP2 (IP 地址 = IP0:IP1:IP2:IP3)	读/写	读/控制	读/控制
LW-9828	(16bit)：远程触摸屏 6 的 IP3 (IP 地址 = IP0:IP1:IP2:IP3)	读/写	读/控制	读/控制
LW-9829	(16bit)：远程触摸屏 6 的连接埠	读/写	读/控制	读/控制
LW-9830	(16bit)：远程触摸屏 7 的 IP0 (IP 地址 = IP0:IP1:IP2:IP3)	读/写	读/控制	读/控制
LW-9831	(16bit)：远程触摸屏 7 的 IP1 (IP 地址 = IP0:IP1:IP2:IP3)	读/写	读/控制	读/控制
LW-9832	(16bit)：远程触摸屏 7 的 IP2 (IP 地址 = IP0:IP1:IP2:IP3)	读/写	读/控制	读/控制
LW-9833	(16bit)：远程触摸屏 7 的 IP3 (IP 地址 = IP0:IP1:IP2:IP3)	读/写	读/控制	读/控制
LW-9834	(16bit)：远程触摸屏 7 的连接埠	读/写	读/控制	读/控制
LW-9835	(16bit)：远程触摸屏 8 的 IP0 (IP 地址 = IP0:IP1:IP2:IP3)	读/写	读/控制	读/控制
LW-9836	(16bit)：远程触摸屏 8 的 IP1 (IP 地址 = IP0:IP1:IP2:IP3)	读/写	读/控制	读/控制
LW-9837	(16bit)：远程触摸屏 8 的 IP2 (IP 地址 = IP0:IP1:IP2:IP3)	读/写	读/控制	读/控制
LW-9838	(16bit)：远程触摸屏 8 的 IP3 (IP 地址 = IP0:IP1:IP2:IP3)	读/写	读/控制	读/控制
LW-9839	(16bit)：远程触摸屏 8 的连接埠	读/写	读/控制	读/控制
LW-9905	(16bit)：远程触摸屏 21 的 IP0 (IP 地址 = IP0:IP1:IP2:IP3) *注 4	读/写	读/控制	读/控制
LW-9906	(16bit)：远程触摸屏 21 的 IP1 (IP 地址 = IP0:IP1:IP2:IP3)	读/写	读/控制	读/控制
LW-9907	(16bit)：远程触摸屏 21 的 IP2 (IP 地址 = IP0:IP1:IP2:IP3)	读/写	读/控制	读/控制
LW-9908	(16bit)：远程触摸屏 21 的 IP3 (IP 地址 =	读/写	读/控制	读/控制

	IP0:IP1:IP2:IP3)			
LW-9909	(16bit)：远程触摸屏 21 的连接埠	读/写	读/控制	读/控制
LW-9910	(16bit)：远程触摸屏 22 的 IP0 (IP 地址 = IP0:IP1:IP2:IP3)	读/写	读/控制	读/控制
LW-9911	(16bit)：远程触摸屏 22 的 IP1 (IP 地址 = IP0:IP1:IP2:IP3)	读/写	读/控制	读/控制
LW-9912	(16bit)：远程触摸屏 22 的 IP2 (IP 地址 = IP0:IP1:IP2:IP3)	读/写	读/控制	读/控制
LW-9913	(16bit)：远程触摸屏 22 的 IP3 (IP 地址 = IP0:IP1:IP2:IP3)	读/写	读/控制	读/控制
LW-9914	(16bit)：远程触摸屏 22 的连接埠	读/写	读/控制	读/控制
LW-9915	(16bit)：远程触摸屏 23 的 IP0 (IP 地址 = IP0:IP1:IP2:IP3)	读/写	读/控制	读/控制
LW-9916	(16bit)：远程触摸屏 23 的 IP1 (IP 地址 = IP0:IP1:IP2:IP3)	读/写	读/控制	读/控制
LW-9917	(16bit)：远程触摸屏 23 的 IP2 (IP 地址 = IP0:IP1:IP2:IP3)	读/写	读/控制	读/控制
LW-9918	(16bit)：远程触摸屏 23 的 IP3 (IP 地址 = IP0:IP1:IP2:IP3)	读/写	读/控制	读/控制
LW-9919	(16bit)：远程触摸屏 23 的连接埠	读/写	读/控制	读/控制
LW-9920	(16bit)：远程触摸屏 24 的 IP0 (IP 地址 = IP0:IP1:IP2:IP3)	读/写	读/控制	读/控制
LW-9921	(16bit)：远程触摸屏 24 的 IP1 (IP 地址 = IP0:IP1:IP2:IP3)	读/写	读/控制	读/控制
LW-9922	(16bit)：远程触摸屏 24 的 IP2 (IP 地址 = IP0:IP1:IP2:IP3)	读/写	读/控制	读/控制
LW-9923	(16bit)：远程触摸屏 24 的 IP3 (IP 地址 = IP0:IP1:IP2:IP3)	读/写	读/控制	读/控制
LW-9924	(16bit)：远程触摸屏 24 的连接埠	读/写	读/控制	读/控制
LW-9925	(16bit)：远程触摸屏 25 的 IP0 (IP 地址 = IP0:IP1:IP2:IP3)	读/写	读/控制	读/控制
LW-9926	(16bit)：远程触摸屏 25 的 IP1 (IP 地址 = IP0:IP1:IP2:IP3)	读/写	读/控制	读/控制
LW-9927	(16bit)：远程触摸屏 25 的 IP2 (IP 地址 = IP0:IP1:IP2:IP3)	读/写	读/控制	读/控制
LW-9928	(16bit)：远程触摸屏 25 的 IP3 (IP 地址 = IP0:IP1:IP2:IP3)	读/写	读/控制	读/控制

LW-9929	(16bit)：远程触摸屏 25 的连接埠	读/写	读/控制	读/控制
LW-9930	(16bit)：远程触摸屏 26 的 IP0 (IP 地址 = IP0:IP1:IP2:IP3)	读/写	读/控制	读/控制
LW-9931	(16bit)：远程触摸屏 26 的 IP1 (IP 地址 = IP0:IP1:IP2:IP3)	读/写	读/控制	读/控制
LW-9932	(16bit)：远程触摸屏 26 的 IP2 (IP 地址 = IP0:IP1:IP2:IP3)	读/写	读/控制	读/控制
LW-9933	(16bit)：远程触摸屏 26 的 IP3 (IP 地址 = IP0:IP1:IP2:IP3)	读/写	读/控制	读/控制
LW-9934	(16bit)：远程触摸屏 26 的连接埠	读/写	读/控制	读/控制
LW-9935	(16bit)：远程触摸屏 27 的 IP0 (IP 地址 = IP0:IP1:IP2:IP3)	读/写	读/控制	读/控制
LW-9936	(16bit)：远程触摸屏 27 的 IP1 (IP 地址 = IP0:IP1:IP2:IP3)	读/写	读/控制	读/控制
LW-9937	(16bit)：远程触摸屏 27 的 IP2 (IP 地址 = IP0:IP1:IP2:IP3)	读/写	读/控制	读/控制
LW-9938	(16bit)：远程触摸屏 27 的 IP3 (IP 地址 = IP0:IP1:IP2:IP3)	读/写	读/控制	读/控制
LW-9939	(16bit)：远程触摸屏 27 的连接埠	读/写	读/控制	读/控制
LW-9940	(16bit)：远程触摸屏 28 的 IP0 (IP 地址 = IP0:IP1:IP2:IP3)	读/写	读/控制	读/控制
LW-9941	(16bit)：远程触摸屏 28 的 IP1 (IP 地址 = IP0:IP1:IP2:IP3)	读/写	读/控制	读/控制
LW-9942	(16bit)：远程触摸屏 28 的 IP2 (IP 地址 = IP0:IP1:IP2:IP3)	读/写	读/控制	读/控制
LW-9943	(16bit)：远程触摸屏 28 的 IP3 (IP 地址 = IP0:IP1:IP2:IP3)	读/写	读/控制	读/控制
LW-9944	(16bit)：远程触摸屏 28 的连接埠	读/写	读/控制	读/控制
LW-9945	(16bit)：远程触摸屏 29 的 IP0 (IP 地址 = IP0:IP1:IP2:IP3)	读/写	读/控制	读/控制
LW-9946	(16bit)：远程触摸屏 29 的 IP1 (IP 地址 = IP0:IP1:IP2:IP3)	读/写	读/控制	读/控制
LW-9947	(16bit)：远程触摸屏 29 的 IP2 (IP 地址 = IP0:IP1:IP2:IP3)	读/写	读/控制	读/控制
LW-9948	(16bit)：远程触摸屏 29 的 IP3 (IP 地址 = IP0:IP1:IP2:IP3)	读/写	读/控制	读/控制
LW-9949	(16bit)：远程触摸屏 29 的连接埠	读/写	读/控制	读/控制

LW-9950	(16bit)：远程触摸屏 30 的 IP0 (IP 地址 = IP0:IP1:IP2:IP3)	读/写	读/控制	读/控制
LW-9951	(16bit)：远程触摸屏 30 的 IP1 (IP 地址 = IP0:IP1:IP2:IP3)	读/写	读/控制	读/控制
LW-9952	(16bit)：远程触摸屏 30 的 IP2 (IP 地址 = IP0:IP1:IP2:IP3)	读/写	读/控制	读/控制
LW-9953	(16bit)：远程触摸屏 30 的 IP3 (IP 地址 = IP0:IP1:IP2:IP3)	读/写	读/控制	读/控制
LW-9954	(16bit)：远程触摸屏 30 的连接埠	读/写	读/控制	读/控制
LW-9955	(16bit)：远程触摸屏 31 的 IP0 (IP 地址 = IP0:IP1:IP2:IP3)	读/写	读/控制	读/控制
LW-9956	(16bit)：远程触摸屏 31 的 IP1 (IP 地址 = IP0:IP1:IP2:IP3)	读/写	读/控制	读/控制
LW-9957	(16bit)：远程触摸屏 31 的 IP2 (IP 地址 = IP0:IP1:IP2:IP3)	读/写	读/控制	读/控制
LW-9958	(16bit)：远程触摸屏 31 的 IP3 (IP 地址 = IP0:IP1:IP2:IP3)	读/写	读/控制	读/控制
LW-9959	(16bit)：远程触摸屏 31 的连接埠	读/写	读/控制	读/控制
LW-9960	(16bit)：远程触摸屏 32 的 IP0 (IP 地址 = IP0:IP1:IP2:IP3)	读/写	读/控制	读/控制
LW-9961	(16bit)：远程触摸屏 32 的 IP1 (IP 地址 = IP0:IP1:IP2:IP3)	读/写	读/控制	读/控制
LW-9962	(16bit)：远程触摸屏 32 的 IP2 (IP 地址 = IP0:IP1:IP2:IP3)	读/写	读/控制	读/控制
LW-9963	(16bit)：远程触摸屏 32 的 IP3 (IP 地址 = IP0:IP1:IP2:IP3)	读/写	读/控制	读/控制
LW-9964	(16bit)：远程触摸屏 32 的连接埠	读/写	读/控制	读/控制

Note

1. LB-9068 自动连结远程触摸屏的相关寄存器，可延伸使用到 LB-9099 远程触摸屏 32。
2. LB-9100 远程触摸屏通讯状态的相关寄存器，可延伸使用到 LB-9148 远程触摸屏 49。
3. LW-9800 远程触摸屏 1 地址设定的相关寄存器，可延伸使用到 LW-9899 远程触摸屏 20。
4. LW-9905 远程触摸屏 21 地址设定的相关寄存器，可延伸使用到 LW-9999 远程触摸屏 39。

22.22 与远程 PLC 通讯状态

地址	描述	读 / 写 / 控制		
		本地 HMI	宏	远程 HMI
LW-10050	(16bit)：连接远程 PLC 1 的触摸屏的 IP0 (IP 地址 = IP0:IP1:IP2:IP3) *注 1	读/写	读/控制	读/控制
LW-10051	(16bit)：连接远程 PLC 1 的触摸屏的 IP1 (IP 地址 = IP0:IP1:IP2:IP3)	读/写	读/控制	读/控制
LW-10052	(16bit)：连接远程 PLC 1 的触摸屏的 IP2 (IP 地址 = IP0:IP1:IP2:IP3)	读/写	读/控制	读/控制
LW-10053	(16bit)：连接远程 PLC 1 的触摸屏的 IP3 (IP 地址 = IP0:IP1:IP2:IP3)	读/写	读/控制	读/控制
LW-10054	(16bit)：连接远程 PLC 1 的触摸屏的连接埠	读/写	读/控制	读/控制
LW-10055	(16bit)：连接远程 PLC 2 的触摸屏的 IP0 (IP 地址 = IP0:IP1:IP2:IP3)	读/写	读/控制	读/控制
LW-10056	(16bit)：连接远程 PLC 2 的触摸屏的 IP1 (IP 地址 = IP0:IP1:IP2:IP3)	读/写	读/控制	读/控制
LW-10057	(16bit)：连接远程 PLC 2 的触摸屏的 IP2 (IP 地址 = IP0:IP1:IP2:IP3)	读/写	读/控制	读/控制
LW-10058	(16bit)：连接远程 PLC 2 的触摸屏的 IP3 (IP 地址 = IP0:IP1:IP2:IP3)	读/写	读/控制	读/控制
LW-10059	(16bit)：连接远程 PLC 2 的触摸屏的连接埠	读/写	读/控制	读/控制
LW-10060	(16bit)：连接远程 PLC 3 的触摸屏的 IP0 (IP 地址 = IP0:IP1:IP2:IP3)	读/写	读/控制	读/控制
LW-10061	(16bit)：连接远程 PLC 3 的触摸屏的 IP1 (IP 地址 = IP0:IP1:IP2:IP3)	读/写	读/控制	读/控制
LW-10062	(16bit)：连接远程 PLC 3 的触摸屏的 IP2 (IP 地址 = IP0:IP1:IP2:IP3)	读/写	读/控制	读/控制
LW-10063	(16bit)：连接远程 PLC 3 的触摸屏的 IP3 (IP 地址 = IP0:IP1:IP2:IP3)	读/写	读/控制	读/控制
LW-10064	(16bit)：连接远程 PLC 3 的触摸屏的连接埠	读/写	读/控制	读/控制
LW-10065	(16bit)：连接远程 PLC 4 的触摸屏的 IP0 (IP 地址 = IP0:IP1:IP2:IP3)	读/写	读/控制	读/控制
LW-10066	(16bit)：连接远程 PLC 4 的触摸屏的 IP1 (IP 地址 = IP0:IP1:IP2:IP3)	读/写	读/控制	读/控制

LW-10067	(16bit)：连接远程 PLC 4 的触摸屏的 IP2 (IP 地址 = IP0:IP1:IP2:IP3)	读/写	读/控制	读/控制
LW-10068	(16bit)：连接远程 PLC 4 的触摸屏的 IP3 (IP 地址 = IP0:IP1:IP2:IP3)	读/写	读/控制	读/控制
LW-10069	(16bit)：连接远程 PLC 4 的触摸屏的连接埠	读/写	读/控制	读/控制
LW-10300	(16bit)：远程 PLC 1 的 IP0 (IP 地址 = IP0:IP1:IP2:IP3)	读/写	读/控制	读/控制
LW-10301	(16bit)：远程 PLC 1 的 IP1 (IP 地址 = IP0:IP1:IP2:IP3)	读/写	读/控制	读/控制
LW-10302	(16bit)：远程 PLC 1 的 IP2 (IP 地址 = IP0:IP1:IP2:IP3)	读/写	读/控制	读/控制
LW-10303	(16bit)：远程 PLC 1 的 IP3 (IP 地址 = IP0:IP1:IP2:IP3)	读/写	读/控制	读/控制
LW-10304	(16bit)：远程 PLC 1 的连接埠	读/写	读/控制	读/控制
LW-10305	(16bit)：远程 PLC 2 的 IP0 (IP 地址 = IP0:IP1:IP2:IP3)	读/写	读/控制	读/控制
LW-10306	(16bit)：远程 PLC 2 的 IP1 (IP 地址 = IP0:IP1:IP2:IP3)	读/写	读/控制	读/控制
LW-10307	(16bit)：远程 PLC 2 的 IP2 (IP 地址 = IP0:IP1:IP2:IP3)	读/写	读/控制	读/控制
LW-10308	(16bit)：远程 PLC 2 的 IP3 (IP 地址 = IP0:IP1:IP2:IP3)	读/写	读/控制	读/控制
LW-10309	(16bit)：远程 PLC 2 的连接埠	读/写	读/控制	读/控制
LW-10310	(16bit)：远程 PLC 3 的 IP0 (IP 地址 = IP0:IP1:IP2:IP3)	读/写	读/控制	读/控制
LW-10311	(16bit)：远程 PLC 3 的 IP1 (IP 地址 = IP0:IP1:IP2:IP3)	读/写	读/控制	读/控制
LW-10312	(16bit)：远程 PLC 3 的 IP2 (IP 地址 = IP0:IP1:IP2:IP3)	读/写	读/控制	读/控制
LW-10313	(16bit)：远程 PLC 3 的 IP3 (IP 地址 = IP0:IP1:IP2:IP3)	读/写	读/控制	读/控制
LW-10314	(16bit)：远程 PLC 3 的连接埠	读/写	读/控制	读/控制
LW-10315	(16bit)：远程 PLC 4 的 IP0 (IP 地址 = IP0:IP1:IP2:IP3)	读/写	读/控制	读/控制
LW-10316	(16bit)：远程 PLC 4 的 IP1 (IP 地址 = IP0:IP1:IP2:IP3)	读/写	读/控制	读/控制
LW-10317	(16bit)：远程 PLC 4 的 IP2 (IP 地址 =	读/写	读/控制	读/控制

	IP0:IP1:IP2:IP3)			
LW-10318	(16bit) : 远程 PLC 4 的 IP3 (IP 地址 = IP0:IP1:IP2:IP3)	读/写	读/控制	读/控制
LW-10319	(16bit) : 远程 PLC 4 的连接埠	读/写	读/控制	读/控制

Note

1. LW-10050 连接远程 PLC 的触摸屏地址的相关寄存器，可延伸使用到 LW-10299 连接远程 PLC 50 的 HMI。

22.23 通讯错误讯息及未处理命令数

地址	描述	读 / 写 / 控制		
		本地 HMI	宏	远程 HMI
LW-9350	(16bit) : 本机尚未处理的命令数目	读	读	读
LW-9351	(16bit) : PLC 1 (COM 1) 尚未处理的命令数目 *注 1	读	读	读
LW-9352	(16bit) : PLC 2 (COM 2) 尚未处理的命令数目	读	读	读
LW-9353	(16bit) : PLC 3 (COM 3) 尚未处理的命令数目	读	读	读
LW-9354	(16bit) : PLC 4 (以太网) 尚未处理的命令数目	读	读	读
LW-9355	(16bit) : PLC 5 (以太网) 尚未处理的命令数目	读	读	读
LW-9356	(16bit) : PLC 6 (以太网) 尚未处理的命令数目	读	读	读
LW-9357	(16bit) : PLC 7 (以太网) 尚未处理的命令数目	读	读	读
LW-9390	(16bit) : PLC (USB) 尚未处理的命令数目	读	读	读
LW-9392	(16bit) : PLC (CAN Bus) 尚未处理的命令数目	读	读	读
LW-9400	(16bit) : 与 PLC 1 通讯错误时产生的错误讯息 *注 2	读	读	读
LW-9401	(16bit) : 与 PLC 2 通讯错误时产生的错误讯息	读	读	读
LW-9402	(16bit) : 与 PLC 3 通讯错误时产生的错误讯息	读	读	读
LW-9403	(16bit) : 与 PLC 4 通讯错误时产生的错误讯息	读	读	读
LW-9404	(16bit) : 与 PLC 5 通讯错误时产生的错误讯息	读	读	读
LW-9405	(16bit) : 与 PLC 6 通讯错误时产生的错误讯息	读	读	读
LW-9406	(16bit) : 与 PLC 7 通讯错误时产生的错误讯息	读	读	读
LW-9407	(16bit) : 与 PLC 8 通讯错误时产生的错误讯息	读	读	读
LW-9490	(16bit) : 与 PLC (USB) 通讯错误时产生的错误讯息	读	读	读

Note

1. LW-9351 PLC 尚未处理的命令数目的相关寄存器，可延伸使用到 LW-9389 PLC 39。
2. LW-9400 与 PLC 通讯错误时产生的错误讯息的相关寄存器，可延伸使用到 LW-9449 PLC 50。

22.24 其它功能项目

地址	描述	读 / 写 / 控制		
		本地 HMI	宏	远程 HMI
LB-9000~ LB-9009	重新开机时状态为 ON	读/写	读/控制	读/控制
LB-9010	资料下载指示	读	读	读
LB-9011	资料上传指示	读	读	读
LB-9012	资料下载/上传指示	读	读	读
LB-9016	远程触摸屏连接至本机触摸屏(当状态为 ON)	读	读	读
LB-9017	取消 PLC 控制元件”切换窗口”的”写回”功能	读/写	读/控制	读/控制
LB-9039	文件备份动作状态 (备份中状态为 ON)	读	读	读
LB-9045	memory-map 通讯失败 (当状态为 ON)	读	读	读
LB-9049	看门狗功能 “开启 (ON)/取消 (OFF)” *注 1	读/写	读/控制	读/控制
LB-9059	关闭宏指令 TRACE 功能 (当状态为 ON) *注 2	读/写	读/控制	读/控制
LB-9064	启用 USB 条形码扫瞄器设备 (键盘功能关闭) (当状态为 ON) *注 3	读/写	读/控制	读
LW-9006	(16bit)：连接到本机的远程触摸屏数目	读	读	读
LW-9024	(16bit)：memory link 系统寄存器	读/写	读/控制	读/控制
LW-9032	(8 words)：备份历史记录到 SD 卡, U 盘的文件夹名称 *注 5	读/写	读/控制	读/控制
LW-9050	(16bit)：目前显示的基本窗口编号	读	读	读
LW-9134	(16bit)：目前所使用的语言 (0 ~ 7) *注 4	读/写	读/控制	读/控制
LW-9141	(16bit) :触摸屏站号	读/写	读/控制	读/控制
LW-9216	(16bit)：导入邮件数据的结果	读	读	读
LW-9300	(16bit)：连接在本机的 PLC 1 所使用的驱动程序编号	读	读	读
LW-9301	(16bit)：连接在本机的 PLC 2 所使用的驱动程序编号	读	读	读
LW-9302	(16bit)：连接在本机的 PLC 3 所使用的驱动程序编	读	读	读

	号			
LW-9303	(16bit)：连接在本机的 PLC 4 所使用的驱动程序编号	读	读	读
LW-9900	(16bit) :触摸屏工作模式 (0：正常模式, 1~3：测试模式 (使用 COM 1~COM 3)	读/写	读/控制	读/控制
LW-10884	(16 words) :触摸屏名称	读/写	读/控制	读/控制

Note

1. 若启用 LB-9049 看门狗功能，当触摸屏发生不可预期的通讯错误(失败)时，系统将在 10 秒后自动重启 HMI。
2. 当元件的文字内容要求表现出多国语言的效果时，除了需使用文字标签外，也需搭配系统保留寄存器 LW-9134 的使用。LW-9134 的有效可设定值范围为 0 ~ 7，此字地址数值的映像方式将与下载至触摸屏的语言种类有关。当编译下载的文件没有勾选全部语言时，LW-9134 使用方式将有所改变。例如: 用户在文字标签库建立了 5 种语言，分别是语言 1 (繁体中文)，语言 2 (简体中文)，语言 3 (英文)，语言 4 (法文)，语言 5 (日文)。若用户只下载语言 1、语言 3、语言 5，此时 LW-9134 里的数值对应的语言种类为 0 → 语言 1 (繁体中文)，1 → 语言 3 (英文)，2 → 语言 5 (日文)。通过项目选单元件搭配 LW-9134 来切换语言范例如下:
3. 系统将以触摸屏名称作为预设的备份资料夹名称。

22.25 远程打印/备份服务器

地址	描述	读 / 写 / 控制		
		本地 HMI	宏	远程 HMI
LB-10069	当在线更改远程打印/备份服务器的 IP 时，设 ON 时重新连结远程打印/备份服务器	读/写	读/控制	读/控制
LB-12040	远程打印/备份服务器断线警示 (当状态为 ON)	读	读	读
LW-9770	(16bit) ：远程打印/备份服务器的 IP0 (IP0:IP1:IP2:IP3)	读/写	读/控制	读/控制
LW-9771	(16bit) ：远程打印/备份服务器的 IP1 (IP0:IP1:IP2:IP3)	读/写	读/控制	读/控制
LW-9772	(16bit) ：远程打印/备份服务器的 IP2 (IP0:IP1:IP2:IP3)	读/写	读/控制	读/控制
LW-9773	(16bit) ：远程打印/备份服务器的 IP3 (IP0:IP1:IP2:IP3)	读/写	读/控制	读/控制
LW-9774	(6 words)：登入远程打印/备份服务器所需的用户名	读/写	读/控制	读/控制

	称 *注 1			
LW-9780	(6 words)：登入远程打印/备份服务器所需的密码 *注 1	读/写	读/控制	读/控制

Note

1. 若欲使用 LW-9774 及 LW-9780 更改设定，必须重新启动触摸屏此变更才有效。

22.26 EasyAccess

地址	描述	读 / 写 / 控制		
		本地 HMI	宏	远程 HMI
LB-9051	与 EasyAccess 服务器断线 (设 OFF)/联机 (设 ON)	读/写	读/控制	读/控制
LB-9052	与 EasyAccess 服务器联机状态 (当联机中状态为 ON)	读	读	读
LB-9196	本地触摸屏只支持检视功能 (当状态为 ON)	读/写	读/控制	读/控制
LB-9197	只允许远程触摸屏使用检视功能 (当状态为 ON)	读/写	读/控制	读/控制

关于 EasyAccess 的更多详情，请参阅网址 http://www.ihmi.net/。

22.27 穿透通讯设定

地址	描述	读 / 写 / 控制		
		本地 HMI	宏	远程 HMI
LW-9901	(16bit)：穿透通讯数据来源串行端口口 (1~3：COM 1~COM 3)	读/写	读/控制	读/控制
LW-9902	(16bit)：穿透通讯数据目标串行端口口 (1~3：COM 1~COM 3)	读/写	读/控制	读/控制
LW-9903	(16bit)：穿透通讯控制 (0：正常, 1：暂停, 2：执行穿透功能时，停止触摸屏与 PLC 间的通讯)	读/写	读/控制	读/控制
LW-9904	(16bit)：穿透服务器连接埠 (2000~2100)	读/写	读/控制	读/控制

22.28 禁止弹出 PLC No Response 窗口

地址	描述	读 / 写 / 控制		
		本地 HMI	宏	远程 HMI
LB-9192	禁止弹出 PLC (USB) 的 "PLC No Response" 窗口 (当状态为 ON)	读/写	读/控制	读/控制
LB-11960	禁止弹出 PLC 1 的 "PLC No Response" 窗口 (当状态为 ON) *注 1	读/写	读/控制	读/控制
LB-11961	禁止弹出 PLC 2 的 "PLC No Response" 窗口 (当状态为 ON)	读/写	读/控制	读/控制
LB-11962	禁止弹出 PLC 3 的 "PLC No Response" 窗口 (当状态为 ON)	读/写	读/控制	读/控制
LB-11963	禁止弹出 PLC 4 的 "PLC No Response" 窗口 (当状态为 ON)	读/写	读/控制	读/控制
LB-11964	禁止弹出 PLC 5 的 "PLC No Response" 窗口 (当状态为 ON)	读/写	读/控制	读/控制
LB-11965	禁止弹出 PLC 6 的 "PLC No Response" 窗口 (当状态为 ON)	读/写	读/控制	读/控制
LB-11966	禁止弹出 PLC 7 的 "PLC No Response" 窗口 (当状态为 ON)	读/写	读/控制	读/控制
LB-11967	禁止弹出 PLC 8 的 "PLC No Response" 窗口 (当状态为 ON)	读/写	读/控制	读/控制
LB-12082	禁止弹出 CAN Bus 设备的 "PLC No Response" 窗口 (当状态为 ON)	读/写	读/控制	读/控制

Note

1. LB-11960 禁止弹出 "PLC No Response" 窗口的相关寄存器，可延伸使用到 LB-12026 PLC 67。

22.29 触摸屏与工程文件识别码

地址	描述	读 / 写 / 控制		
		本地 HMI	宏	远程 HMI
LB-9046	工程文件识别码与触摸屏识别码不同 (当状态为 ON)	读	读	读
LW-9046	(32bit)：触摸屏识别码 *注 1	读/写	读/控制	读

Note

1. 若欲使用 LW-9046 更改触摸屏识别码设定，必须重新启动触摸屏此变更才有效。

22.30 快选窗口控制

地址	描述	读 / 写 / 控制		
		本地 HMI	宏	远程 HMI
LB-9013	快选窗口控制“隐藏 (ON)/显示 (OFF)”	读/写	读/控制	读/控制
LB-9014	快选按键控制“隐藏 (ON)/显示 (OFF)”	读/写	读/控制	读/控制
LB-9015	快选窗口/按键控制“隐藏 (ON)/显示 (OFF)”	读/写	读/控制	读/控制

22.31 输入元件功能

地址	描述	读 / 写 / 控制		
		本地 HMI	宏	远程 HMI
LW-9002	(32bit-float)：数值输入元件允许输入的上限值	读	读	读
LW-9004	(32bit-float)：数值输入元件允许输入的下限值	读	读	读
LW-9052	(32bit-float)：数值输入元件的前一次输入值	读	读	读
LW-9150	(32 words)：显示目前键盘所输入的数据 (ASCII)	读	读	读
LW-9540	(16bit)：键盘大小写切换	读/写	读/控制	读/控制

22.32 本地/远程操作限制

地址	描述	读 / 写 / 控制		
		本地 HMI	宏	远程 HMI
LB-9044	禁止远程控制 (当状态为 ON)	读/写	读/控制	读/控制
LB-9053	禁止密码远程读取操作 (当状态为 ON)	读/写	读/控制	读/控制
LB-9054	禁止密码远程写入操作 (当状态为 ON)	读/写	读/控制	读/控制
LB-9196	本地触摸屏只支持检视功能 (当状态为 ON)	读/写	读/控制	读/控制
LB-9197	只允许远程触摸屏使用检视功能 (当状态为 ON)	读/写	读/控制	读/控制
LB-9198	禁止本地触摸屏触发宏 (当状态为 ON)	读/写	读/控制	读/控制
LB-9199	禁止远程触摸屏触发宏 (当状态为 ON)	读/写	读/控制	读/控制

22.33 VNC 控制

地址	描述	读 / 写 / 控制		
		本地 HMI	宏	远程 HMI
LB-12089	VNC 不须密码即可登入 (当状态为 ON) *注 1	读/写	读/控制	读/控制
LB-12090	VNC client 连接至触摸屏(当状态为 ON) (请使用 OS 20120621 或更新版本的 OS)	读	读	读
LB-12091	当 VNC client 连接至触摸屏时取消自动注销功能 (当状态为 ON) (请使用 OS 20120621 或更新版本的 OS)	读/写	读/控制	读/控制
LB-12092	VNC 功能“开启 (ON)/取消 (OFF)”	读/写	读/控制	读/控制
LB-12093	VNC 模式 (OFF：单台联机, ON：多台联机) *注 1	读/写	读/控制	读/控制
LW-9530	(8 words)：VNC 服务器密码	读/写	读/控制	读/控制

Note

1. 若欲更改 VNC 设定模式，必须先关闭 VNC Server 后再开启此变更才有效。

22.34 cMT-SVR 相关寄存器

地址	描述	读 / 写 / 控制		
		本地 HMI	宏	远程 HMI
PLW-9041	(16bit)：触控状态 (位 0 on = 正在触碰屏幕)	读	读	读
PLW-9042	(16bit)：触碰时, X 的位置	读	读	读
PLW-9043	(16bit)：触碰时, Y 的位置	读	读	读
PLW-9044	(16bit)：离开时, X 的位置	读	读	读
PLW-9045	(16bit)：离开时, Y 的位置	读	读	读
PLW-9050	(16bit)：目前显示的基本窗口编号	读	读	读
PLW-9052	(32bit-float)：数值输入元件的前一次输入值	读	读	读
PLW-9134	(16bit)：目前所使用的语言 (0 ~ 7)	读/写	读/控制	读/控制
PLW-9222	(16bit)：目前用户可使用的元件类别 (bit 0:A, bit 1:B,bit 2:C, ...)	读	读	读
PLW-10754	(8 words)：目前登入的用户名称	读	读	读

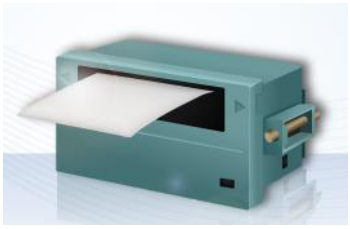
Note

1. LW 与 PLW 不同处在于 LW 是指触摸屏本机上的地址，而 PLW 则是指 iPad 上操作这些功能的地址。因每台 cMT-SVR 可供多台 iPad 连接，因此以上这些功能的系统寄存器将由 iPad 个别处理。

第二十三章 HMI 支持的打印机类型

23.1 支持的打印机类型

HMI 可支持的打印机驱动大致分为以下几种。

打印机类型	描述
● SP-M, D, E, F 	此驱动使用 EPSON ESC 串口微型打印机协议。使用串行端口连接，请配合打印机调整通讯参数。”每行点数” 必须正确设定，且不得超过打印机默认值: 100 pixels：1610 型号之打印机 ； 220 pixels：2407, 4004 型号之打印机。 SP-E1610SK (纸张宽度 45mm), SP-E400-4S (纸张宽度 57.5mm) 建议中国地区以外的用户使用此种 SP 打印机型。 北京迅普: http://www.siupo.com
● EPSON ESC/P2 系列 	使用串行端口连接，请配合打印机调整通讯参数。此驱动使用 EPSON ESC/P2 打印机通讯协议。 针式打印机: LQ-300, LQ-300+, LQ-300K+ (RS-232), LQ-300+II (RS-232) 喷墨式打印机: Stylus Photo 750 激光打印机: EPL-5800
● HP 电脑 L 系列(USB) 	使用 USB 埠连接，适用于所有支持 HP 电脑 L 5 level 3 通讯协议，以及使用 USB 埠的 HP 打印机。 •电脑 L 5 在 1990 年 3 月发布于 HP Laser Jet III。新增功能: Intellifont 字体缩放，outline 字型与 HP-GL/2 (向量) 图形。 •电脑 L 5e (PCL 5 进阶版) 在 1992 年 10 月发布于 P Laser Jet 4。新增功能: 打印机与 PC 双向通讯及 Windows 字型。 关于触摸屏支持的 HP 电脑 L 系列，请参考本章《23.2 查询支持的打印机型号》。

- **Axiohm A630**

法国爱克胜微型打印机，使用串行端口连接，请配合打印机调整通讯参数。

- **SPRT**

使用串行端口连接，请配合打印机调整通讯参数。”每行点数”必须正确设定，且不得超过打印机默认值“100”。
支援型号:
SP-DN40SH：针式打印机
SP-RMDIII40SH：热感式打印机

- **EPSON TM-L90**

使用串行端口连接，请配合打印机调整通讯参数。”每行点数”必须正确设定，且不得超过打印机默认值 “576”。

- **EPSON TM-T70**

使用串行端口连接，请配合打印机调整通讯参数。”每行点数”必须正确设定，且不得超过打印机默认值 “576”。 可选择纸张切割模式，分为“不切”或“半切”。

- **BRIGHTEK WH-A19**

支援型号：
A92R10-00E72A：后缀为 72 的是 16 位打印机，A 表示宽电压 5~9V，此款与 A6 16 点阵相同。

- **BRIGHTEK WH-E19**

使用串行端口连接，请配合打印机调整通讯参数。

- **BRIGHTEK WH-22 (炜煌)**

支援型号：
E22R10-00E725: 与 A7 16 点阵相同，A7 型号为 A72R90-31E72A
E221R90-00E11740GA: 此打印机为 485 接口，需要使用 232 转 485 模块。

- **BRIGHTEK WH-C1/C2**

使用串行端口连接，请配合打印机调整通讯参数。可选择纸张切割模式，分为“不切”、”半切”或“全切”。

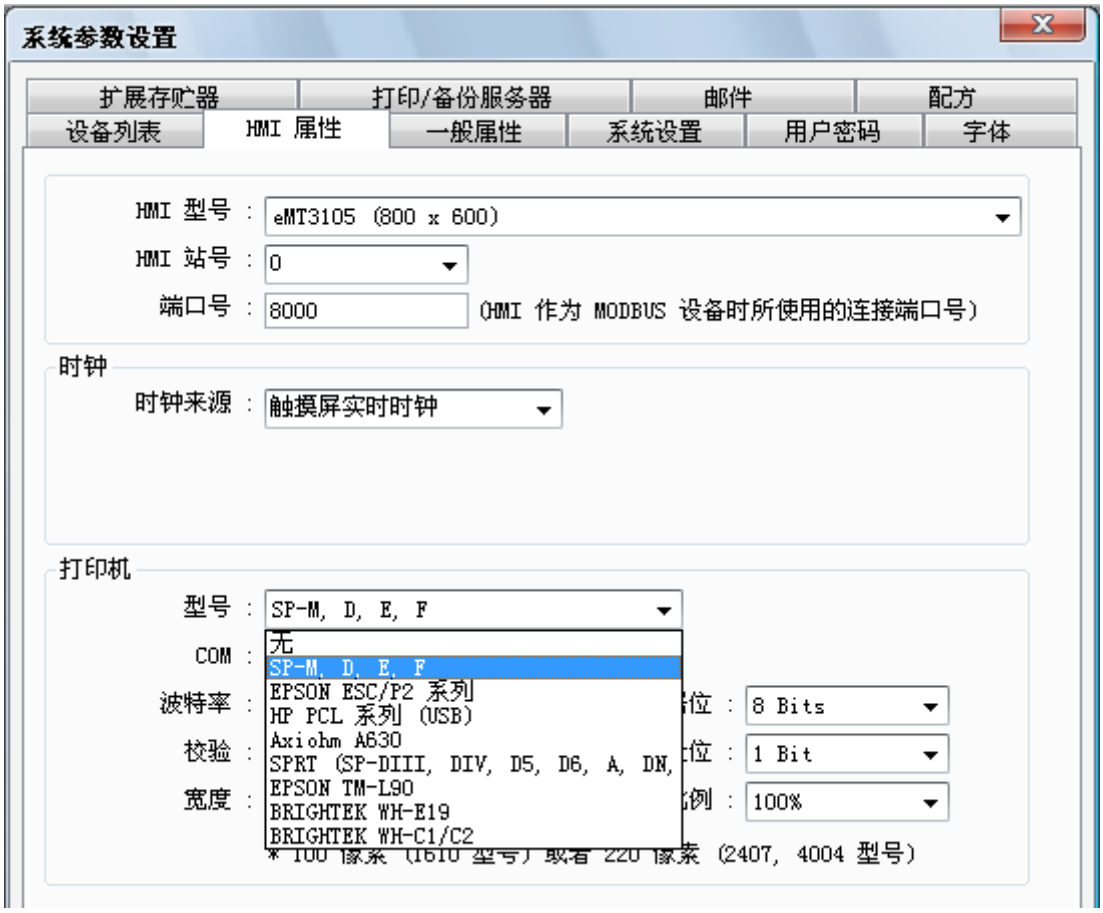
- **远程打印服务器**

HMI 可通过以太网连接在电脑上的打印机，并使用 EasyPrinter 来执行打印动作。由于 EasyPrinter 在 MS Windows 系统下运行，因此支持市面上大部分的打印机。

23.2 如何新增一台打印机设备并触发打印

1. 新增打印机设备。

- 从“系统参数设置” »“HMI 属性”页面选择欲连接的打印机型号，并正确设定相关参数。



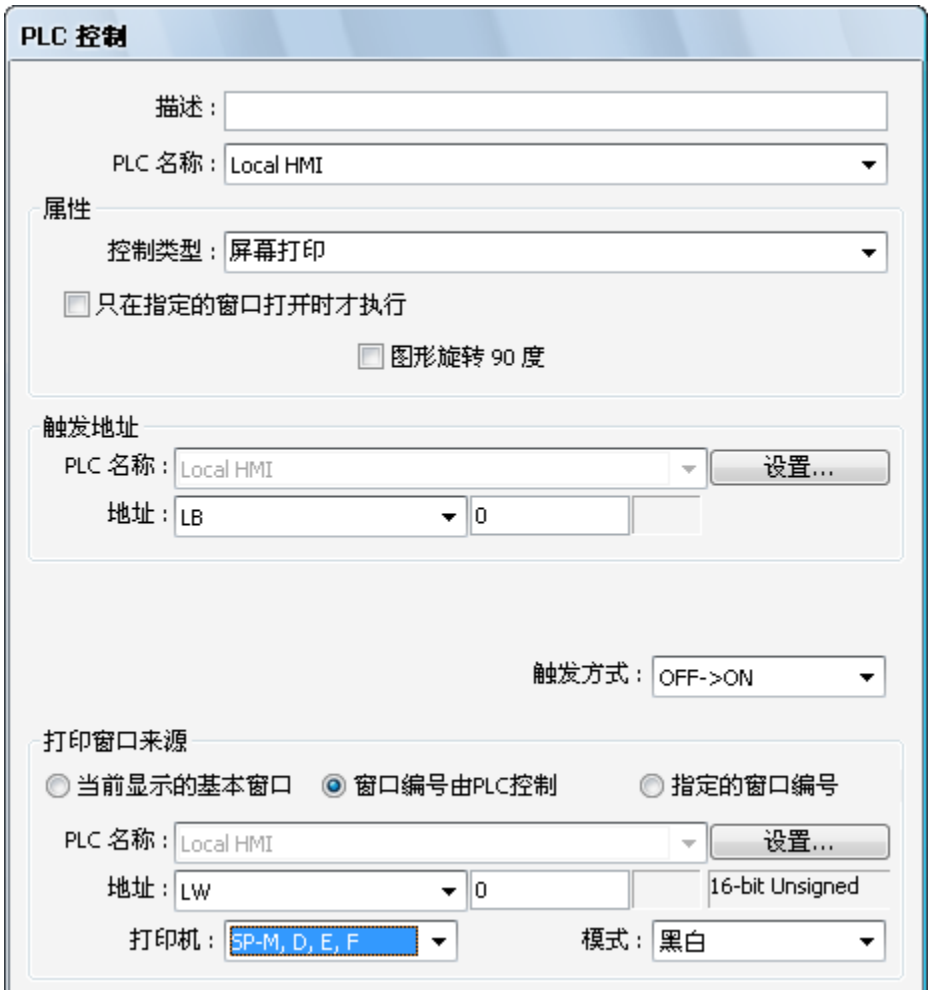
- 若欲连接至远程打印服务器，则需启用“系统参数设置” »“打印 / 备份服务器”页面，并正确设定相关参数。

2. 触发打印。

- 用户可通过“功能键”元件来直接触发打印功能。


新增 功能键 元件
一般属性
安全
图片
标签
描述 :
当按钮松开才发出指令
切换基本窗口
切换公共窗口
弹出窗口
返回上一个窗口
关闭窗口
ASCII/UNICODE 方式
[Enter]
[Backspace]
[Clear]
[Esc]
[ASCII] / [UNICODE
触发宏指令
窗口控制条
屏幕打印至 U 盘/ SD 卡或打印机
画面打印
打印机 :
SP-M, D, E, F
图形旋转 90 度
模式 :
灰阶
导入用户数据/使用 [USB 安全金钥]
通知
启用
确定
取消
帮助

- 或者，使用“PLC 控制”元件的“屏幕打印”，借由预先定义好的位地址来触发打印功能。



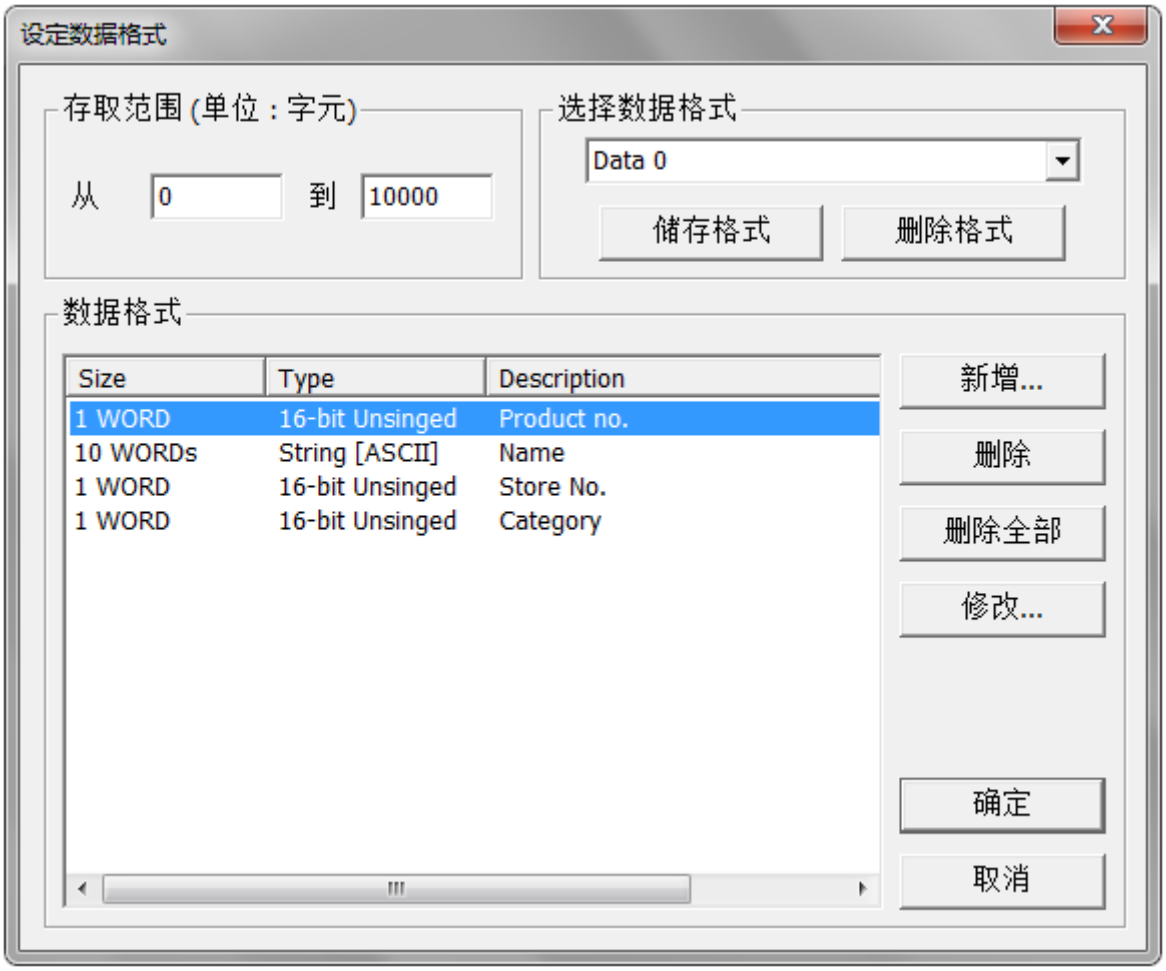
第二十四章 配方编辑器 Recipe Editor

24.1 概要

Recipe Editor 可用来建立触摸屏所使用的配方数据文件，也可开启及编辑现有的配方数据文件。此外，EasyBuilder Pro 提供另一个编辑配方数据文件的工具 – 配方记录，此功能需先在 EasyBuilder Pro “系统参数设置” »“配方”定义配方，再使用“配方检视元件”来显示配方内容，以下将介绍此二种编辑器的使用方法。

24.2 配方数据 / 扩展内存编辑器设定

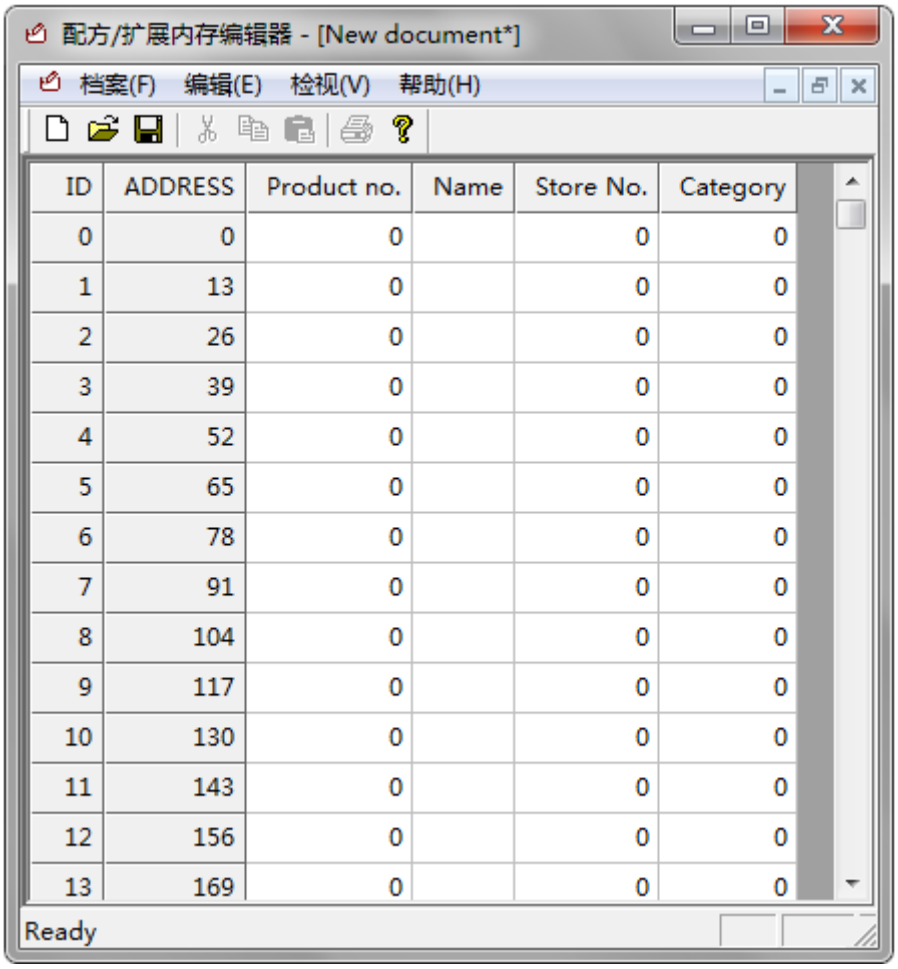
1. 从 Utility Manager 点击“配方数据/扩展内存编辑器”。
2. 要新增新的 .rcp 或是 .emi 文件，请点选“文件” » “开新文件”。
3. 设定存取地址范围与数据格式。



设定	描述
存取范围	填入起始地址和结束地址，以字符为单位。
选择数据格式	定义完成的数据格式可储存，并于下次需要时加载。范本将存成“dataEX.fmt”文件并存放在 EasyBuilder Pro 安装目录下。
数据格式	可在数据格式区域中编辑新的数据格式。

4. 点击“新增”后，弹出数据类型编辑窗口如下，请在“描述”字段输入数据类型的名称，并选择数据格式。若选择“String”，需输入字符长度及设定格式类型为“ASCII”或“Unicode”。

5. 数据格式定义完成后，点选“确定”即可编辑配方数据。



ID	ADDRESS	Product no.	Name	Store No.	Category
0	0	0		0	0
1	13	0		0	0
2	26	0		0	0
3	39	0		0	0
4	52	0		0	0
5	65	0		0	0
6	78	0		0	0
7	91	0		0	0
8	104	0		0	0
9	117	0		0	0
10	130	0		0	0
11	143	0		0	0
12	156	0		0	0
13	169	0		0	0

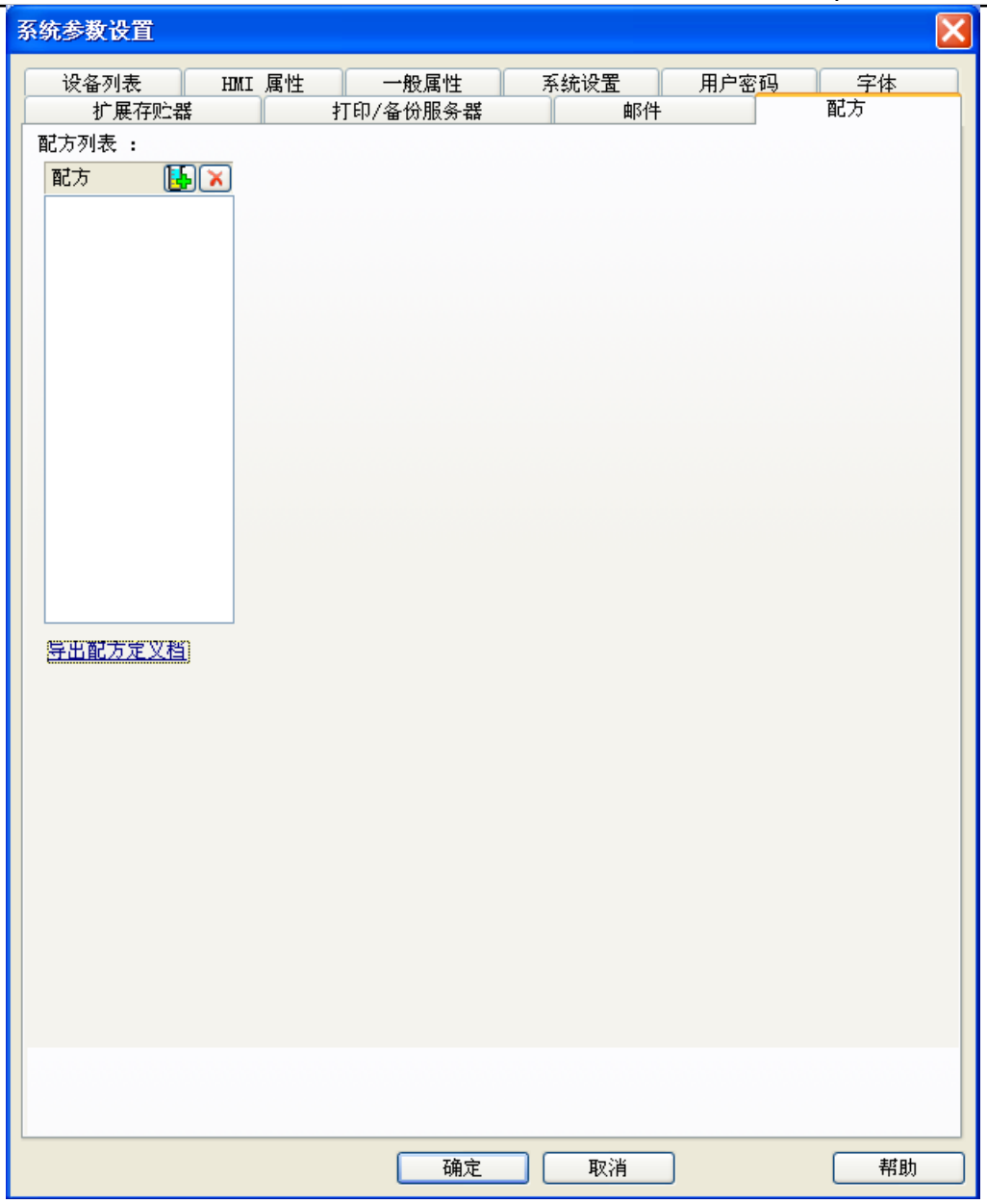
此范例的数据格式长度为 13 个字符，因此可将每 13 个字符长度视为一组配方来使用。如上，
第一组的“Product no.”为 address 0，”Name”地址为 address 1 ~ 10，”Store No.”为 address 11 ，”Category”为 address 12;
第二组的“Product no.”为 address 13，”Name”地址为 address 14 ~ 23，”Store No.”为 address 24 ，”Category”为 address 25…依此类推。

Note

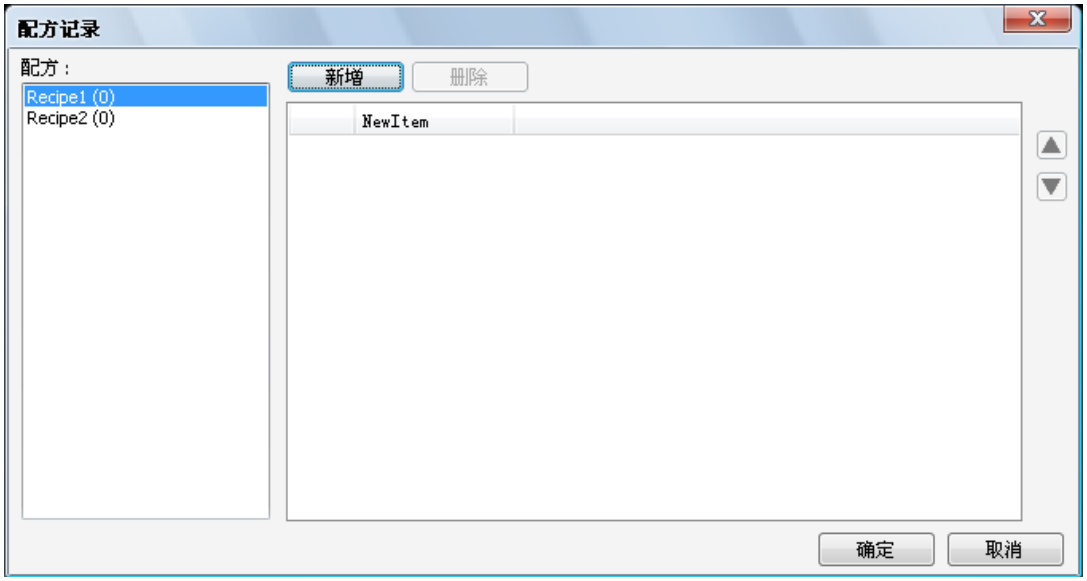
- 编辑完成的配方数据可储存成 .rcp、.emi 或 .csv 文件格式。储存完成的 .rcp 文件可藉由 Utility Manager 或外部设备 (U 盘或 SD 卡) 下载到 HMI，而 .emi 则可直接存放至外部设备并插入至触摸屏读取，即为扩展地址 EM。

24.3 配方记录的设定

1. 在使用配方记录之前，需要先设定 EasyBuilder Pro 的“系统参数设置” »“配方”，详细的设定方式可参考《第五章系统参数设定》章节。

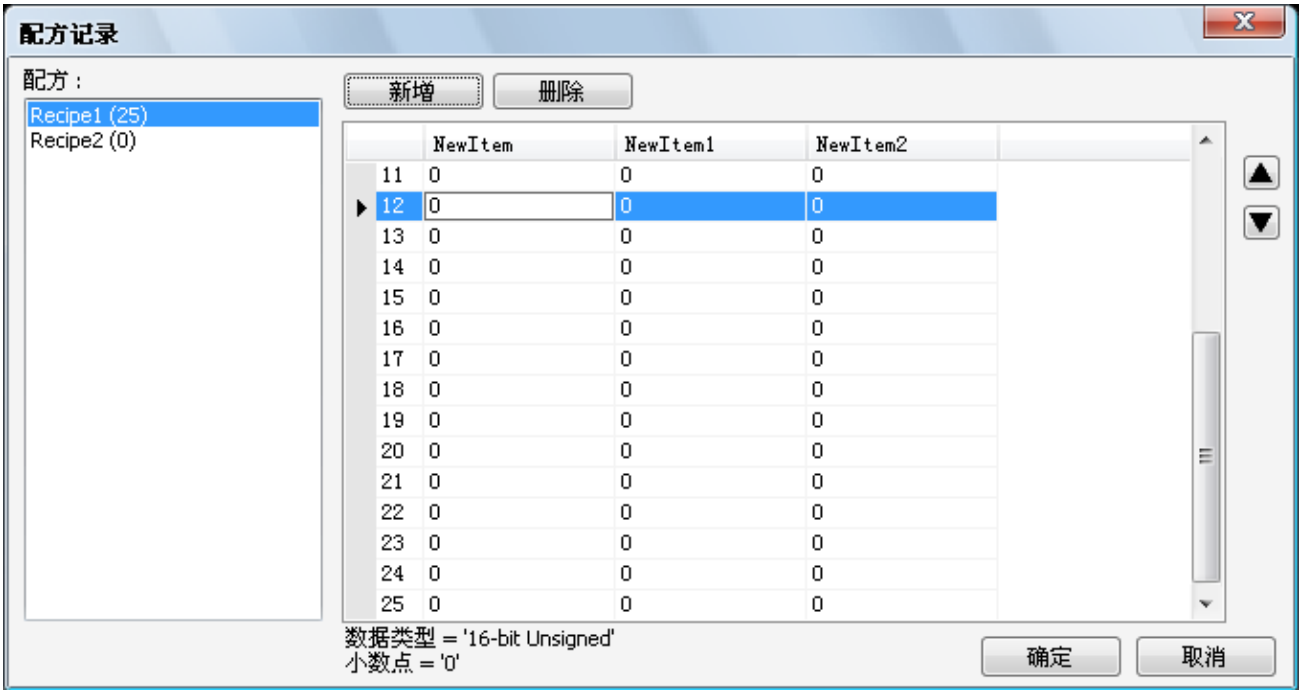

系统参数设置
设备列表
HMI 属性
一般属性
系统设置
用户密码
字体
扩展存贮器
打印/备份服务器
邮件
配方
配方列表 ：
配方
导出配方定义档
确定
取消
帮助

2. 在设定完系统参数设定的配方后，开启“图库”»“配方记录”。于下图中的左边有 Recipe1 和 Recipe2，以及右边上方的三个配方项目，名称皆是读取系统参数设定的配方而来的，接下来可分别针对这二个配方名称的格式来定义配方的数据。



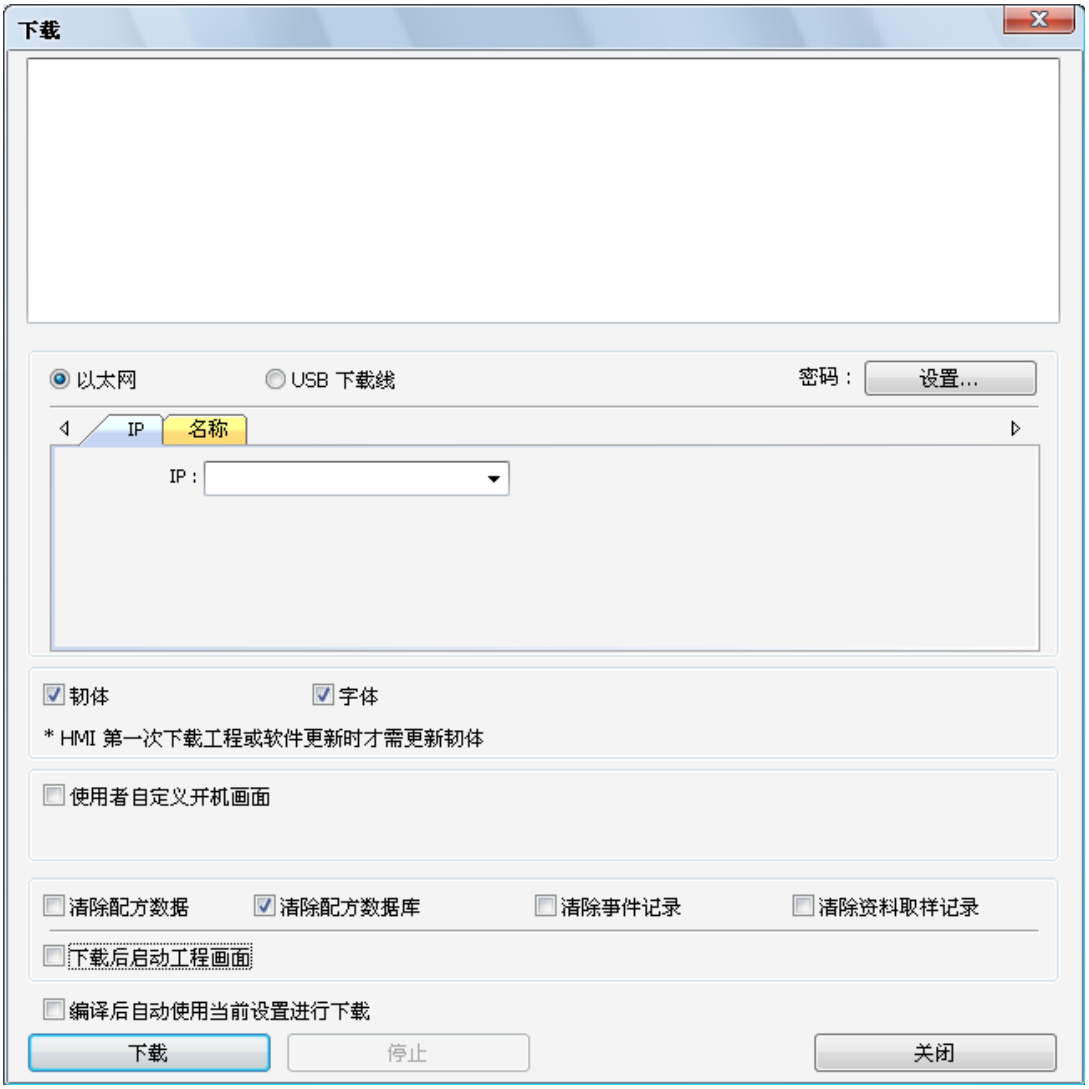
设定	描述
配方	配方列表，此内容可显示系统参数设定的配方，其括号内的数字可显示出有几笔配方资料。
新增	依照各项目的格式定义，可开始编辑配方内容。
删除	可删除所编辑配方的内容。
上下键	利用上下键可移动至想要编辑的配方内容。

3. 依照各项目的格式定义，按下“新增”按钮后，可编辑配方项目的内容，每一个项目的格式在点选字段后会显示在下方，用户可依项目的格式填入所需的内容，并按下“确定”来完成储存的动作。


配方记录
配方：
Recipe1 (25)
Recipe2 (0)
新增
删除
NewItem
NewItem1
NewItem2
11 0 0 0
12 0 0 0
13 0 0 0
14 0 0 0
15 0 0 0
16 0 0 0
17 0 0 0
18 0 0 0
19 0 0 0
20 0 0 0
21 0 0 0
22 0 0 0
23 0 0 0
24 0 0 0
25 0 0 0
数据类型 = '16-bit Unsigned'
小数点 = '0'
确定
取消

Note

- 每一个配方最多可以新增 10000 笔记录。
- 配方数据在编译后会储存于 .exob 文件内并且被下载到 HMI，此配方数据无法共享在其它工程文件。若是在下载工程文件后，再次使用配方记录修改配方内容且需下载到 HMI，务必勾选“清除配方数据库”，若无勾选，则触摸屏还是会保持原本旧的配方数据库内容。



第二十五章 EasyConverter

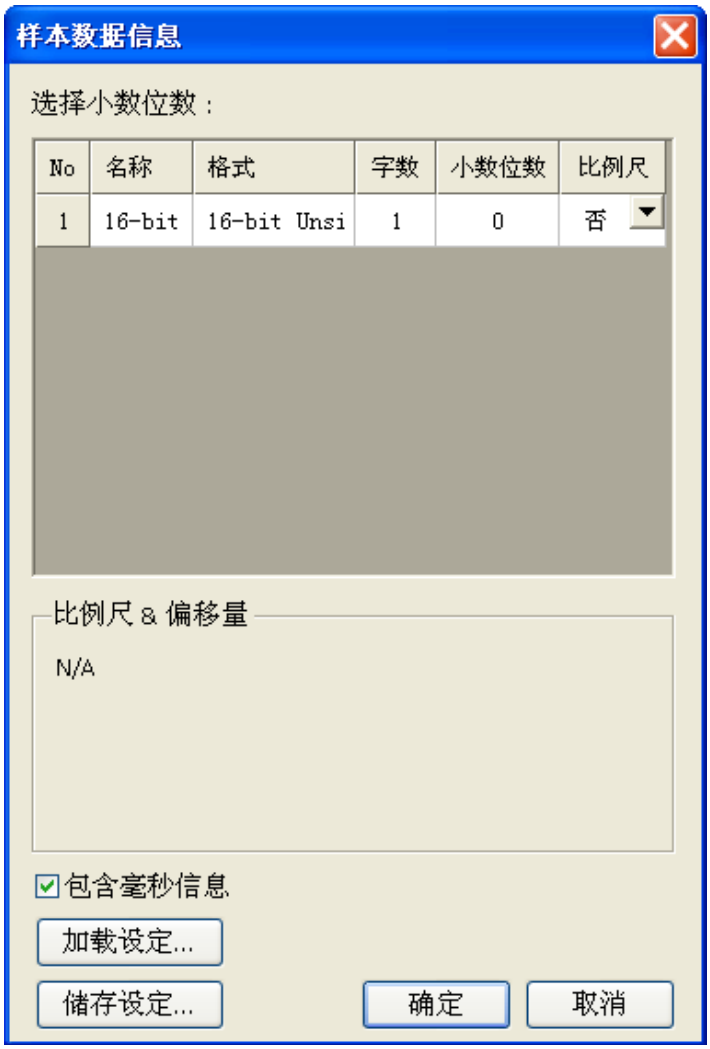
25.1 概要

EasyConverter 可读取由触摸屏保存的资料取样记录 (.dtl) 或事件记录 (.evt)，并转换成 Excel (.xls)格式。

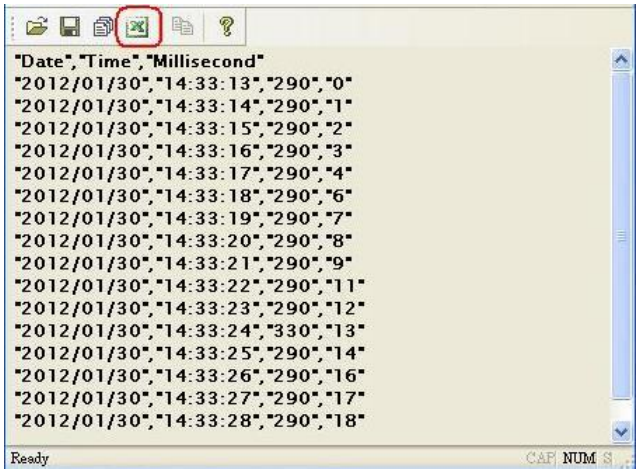
- 从 Utility Manager 点击 EasyConverter。
- 从 EasyBuilder Pro 工具选单下点击“事件记录/资料取样记录转换程序”。

25.2 将 DTL 或 EVT 文件输出至 Excel 的步骤

1. 当开启资料取样 (.dtl) 文件后，会弹出设定窗口如下。



2. 按下“确定”后，再按下“导出至 Microsoft Excel”即可。



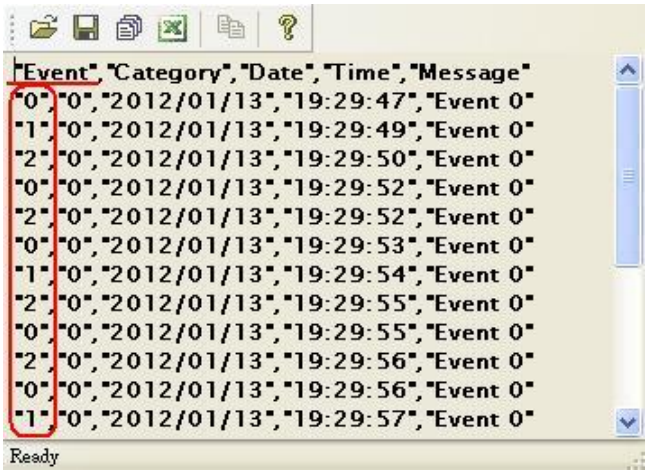
3. Excel 文件呈现如下。

	A	B	C	D	E
1	Date	Time	Millisecond	sample	
2	2012/1/30	14:33:13	290	0	
3	2012/1/30	14:33:14	290	1	
4	2012/1/30	14:33:15	290	2	
5	2012/1/30	14:33:16	290	3	
6	2012/1/30	14:33:17	290	4	
7	2012/1/30	14:33:18	290	6	
8	2012/1/30	14:33:19	290	7	
9	2012/1/30	14:33:20	290	8	
10	2012/1/30	14:33:21	290	9	
11	2012/1/30	14:33:22	290	11	
12	2012/1/30	14:33:23	290	12	
13	2012/1/30	14:33:24	330	13	
14	2012/1/30	14:33:25	290	14	
15	2012/1/30	14:33:26	290	16	
16	2012/1/30	14:33:27	290	17	
17	2012/1/30	14:33:28	290	18	
18					
19					
20					

20120130

当开启事件记录（.evt）档后，可以发现“Event”字段如下:

0 → 表示事件触发时 ；1 → 表示事件确认时 ；2 → 表示事件恢复正常时。



Note

- 当双击 .dtl 或 .evt 文件时，可自动产生 Excel 文件。但若 .dtl 的字符串为 UNICODE 模式，则须开启 EasyConverter 手动转换。

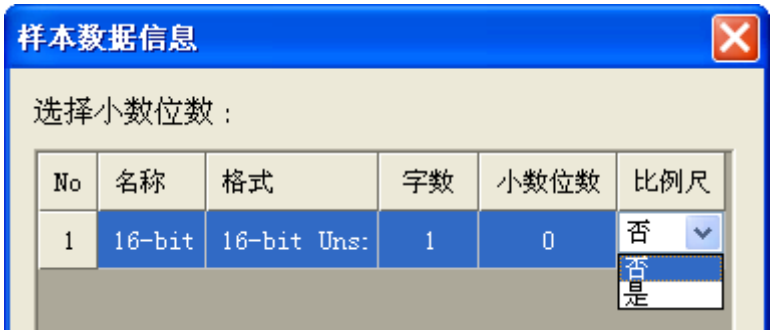
25.3 比例转换功能

比例转换功能使用方式如下:

新数值 = “(数值 + A) x B” + C

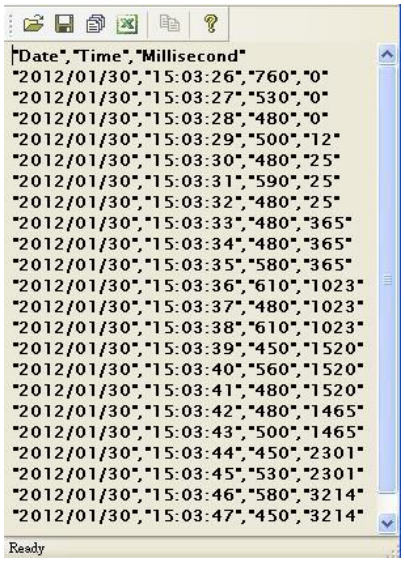
用户可以在 A、B 和 C 设定数值。

A 为数值下限 ; B 为 “(比例最大值) – (比例最小值) / (数值上限) – (数值下限)” ; C 为比例最小值。

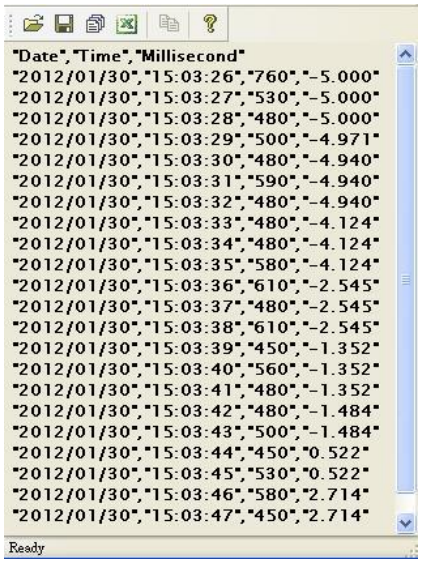


如下，有一电压数据，其格式是 16-bit unsigned，电压数值是介于 0 ~ 4096，若要将其电压数值转换成伏特，介于 -5V ~ +5V 之间。新数值 = “(数值 + 0) x 0.0024” + (-5):

比例转换前



比例转换后

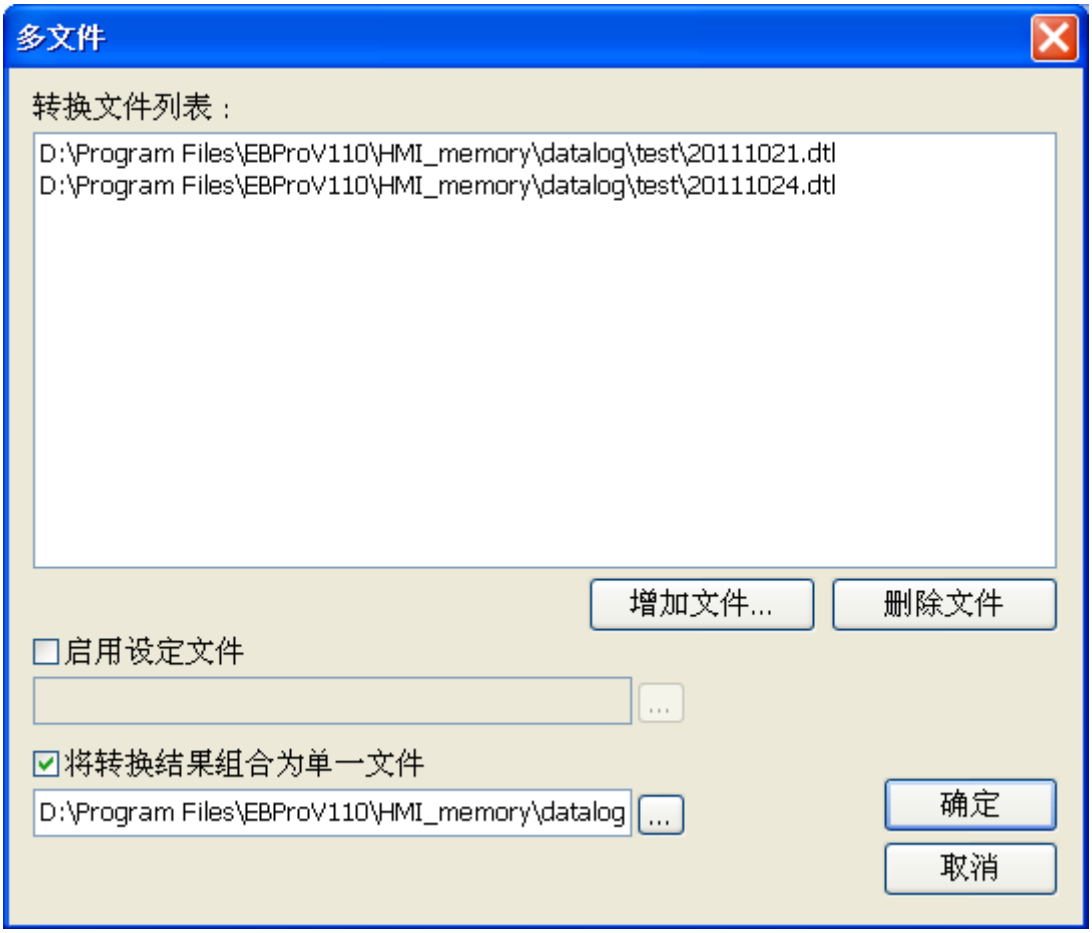


Note

- 以上的数据设定可以储存成模板，并于下次使用时可以直接加载设定。模板设定的附文件名为 .lgs。
- 设定完比例转换的各项数值之后，点选 “储存设置”；并于新的样本数据信息，点选 “加载设置” 即可加载先前模板。

25.4 使用多文件转换的步骤

1. 点选“文件”»“多文件”»“增加文件”加入多个文件合并为一个 Excel (.xls)。
2. 点选“将转换结果组合为单一文件”，文件将被输出为 Excel (.xls) 并一个日期文件分一页面。如不点选此选项而按下确定，文件将个别被输出至 Excel。



3. Excel 文件呈现如下。

	A	B	C	D
1	Date	Time	Millisecon	bit Unsign
2	#########	14:10:38	810	0
3	#########	14:10:39	810	0
4	#########	14:10:40	810	0
5	#########	14:10:41	810	0
6	#########	14:10:42	890	0
7	#########	14:10:43	930	1
8	#########	14:10:44	840	1
9	#########	14:10:45	810	1
10	#########	14:10:46	810	1
11	#########	14:10:47	810	1

20111021 20111024

用户也可以加载已储存的设定文件至合并档：

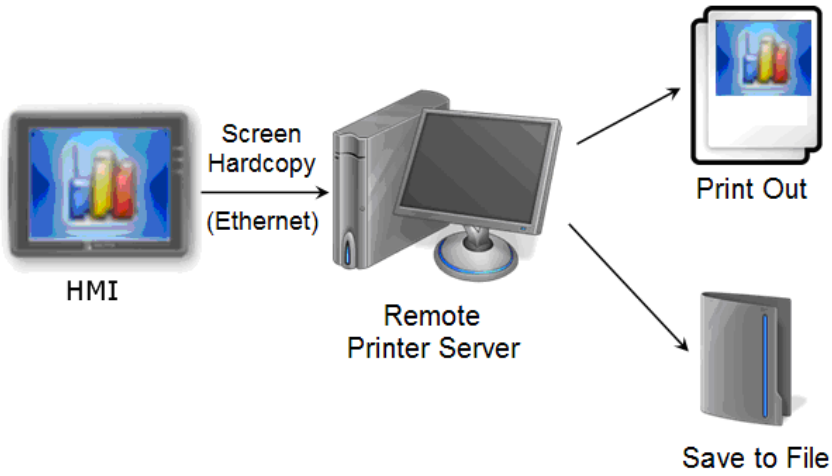
1. 点选“启用设定文件”和“将转换结果组合为单一文件”。

选择所要合并的文件名称后点选“确定”。

第二十六章 EasyPrinter

26.1 概要

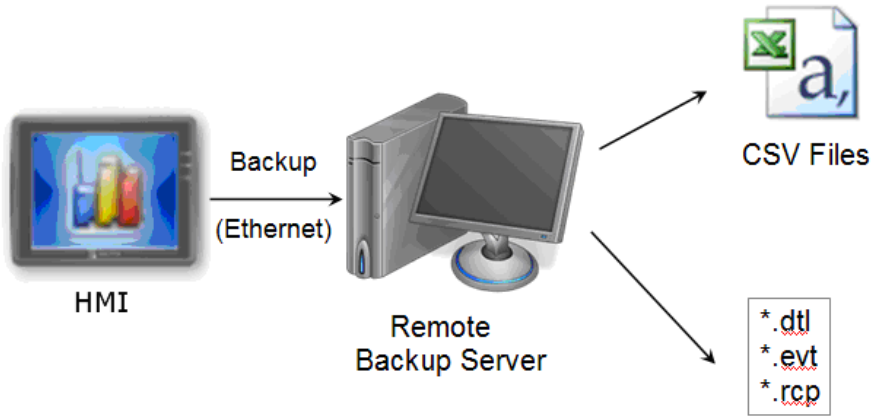
EasyPrinter 是属于 Win 32 的应用程序，因此只能在 MS Windows 2000 / XP / Vista / 7 / 8 等系统下运行。此功能让 HMI 可以通过以太网，输出屏幕撷取画面并打印于远程计算机。



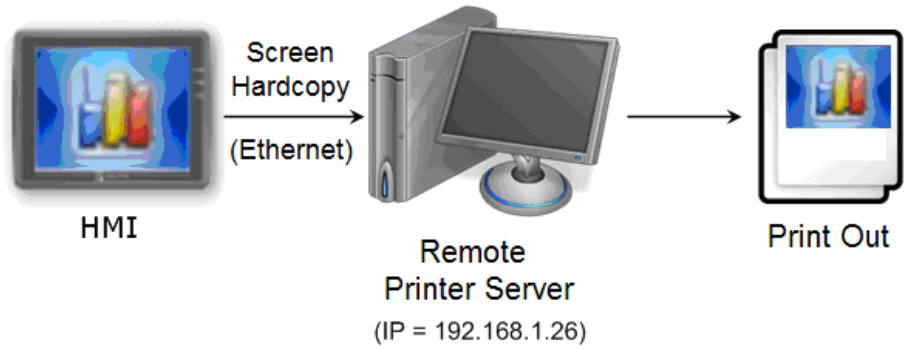
以下为使用 EasyPrinter 的优点：

- EasyPrinter 提供两种屏幕打印输出模式："输出至"及"储存至"。用户可使用其中一种或两个都使用。
- 由于 Easy Printer 在 MS Windows 系统下运行，因此可支持市面上大部分的打印机。
- 此功能下多台 HMI 可以共享一台实体打印机，用户不需为每台 HMI 各准备一台打印机。

另外，EasyPrinter 可以当做是一台备份服务器。用户可使用 HMI 上的备份元件，通过以太网，将取样数据与事件记录等历史文件备份至远程 PC。请见下方说明：



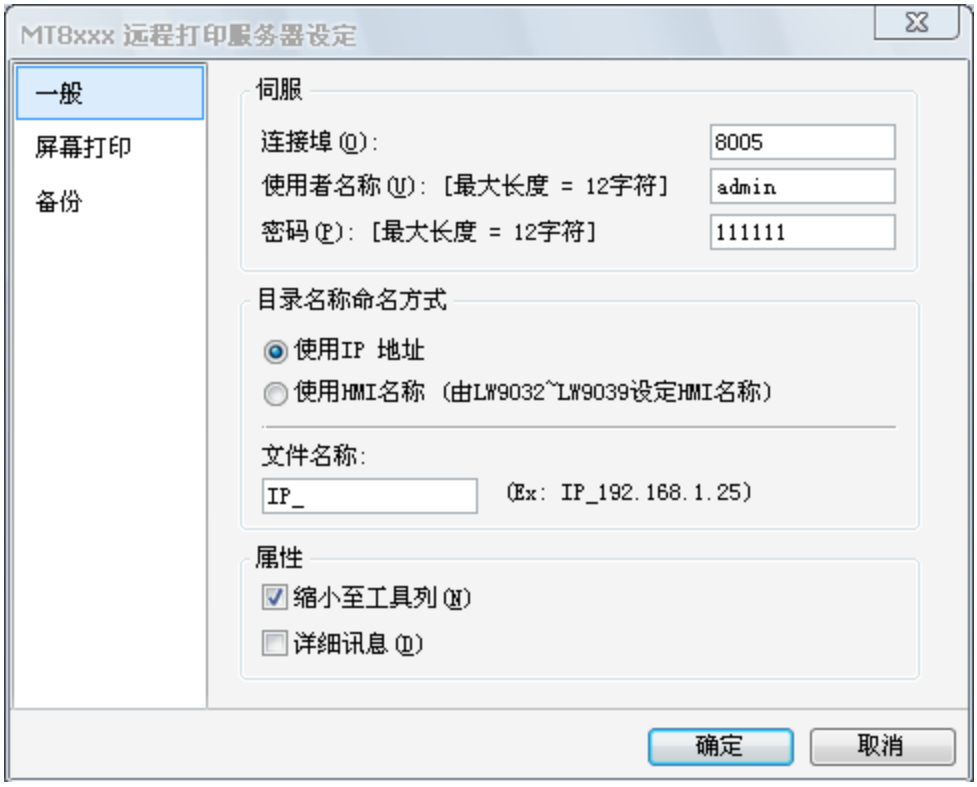
26.2 使用 EasyPrinter 为打印服务器



用户可用“功能键”元件来操作屏幕打印。这些屏幕撷取画面会通过以太网被传送至远程打印机服务器，然后被打印出来。

26.2.1 EasyPrinter 设定程序

在 EasyPrinter 设定页下点选“选项”»“设置”会出现下面的对话框：



1. 点选左列“一般”
2. 在“伺服”，设定“连接埠”为“8005”，”用户名称”为“admin”，”密码”为“111111”。(以上皆为默认值)。
3. 在“目录名称命名方式”，点选“使用 IP 地址”并在“文件名称”填入“IP_”。

4. 在“属性”，选择“缩小至工具列”。

接着设定输出位置。

1. 点选左列“屏幕打印”
2. 在“输出”，点选“输出至”并选择一台打印机做为屏幕打印的输出设备。(注意：用户只能选择自己的系统中存在的打印机，上列打印机仅为参考。)
3. 按下“确定”确认使用以上设定。
4. 在 EasyPrinter 设定页下点选“文件”»“允许输出”，EasyPrinter 会将这些打印指令输出。

26.2.2 EasyBuilder Pro 设定程序

在 EasyBuilder Pro 设定 EasyPrinter 的设定程序：

1. 开启 EasyBuilder Pro，并开启新项目或已使用之项目。
2. 在“编辑”»“系统参数设置” »“打印/备份服务器”中，勾选“使用远程打印/备份服务器”。

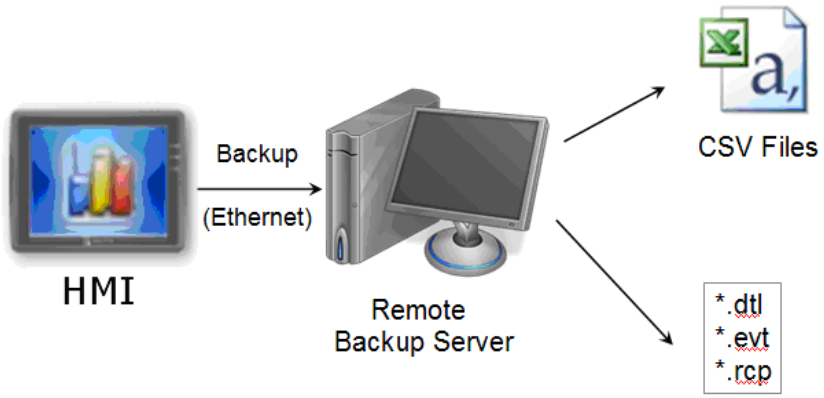
3. 在“输出设置”，指定适当的边界， (在此例中上下左右边界皆设为 15mm)。
4. 在“通讯设置”，输入打印服务器“IP 地址”，同 EasyPrinter 的设定。指定“连接埠”号“8005”，”用户名称”为“admin”，”密码”为“111111”。
5. 按下“确定”。
6. 接着在菜单“元件”»“开关”选择 “功能键”并在元件设定页中点选“画面打印”并将“打印机”设定至“MT 远程打印/备份服务器”。

7. 将“功能键”元件置于“公共窗口”(四号窗口)，用户便可以随时开始屏幕打印
8. “编译”及“下载”工程文件至 HMI，按下已设定的“功能键”元件，开始打印。

Note

- 用户亦可通过“PLC 控制”元件来达成屏幕打印
- 报警数据无法通过 EasyPrinter 打印。
- EasyPrinter 只能通过以太网与 HMI 通讯，请确认所使用 HMI 网络是否设定正确。

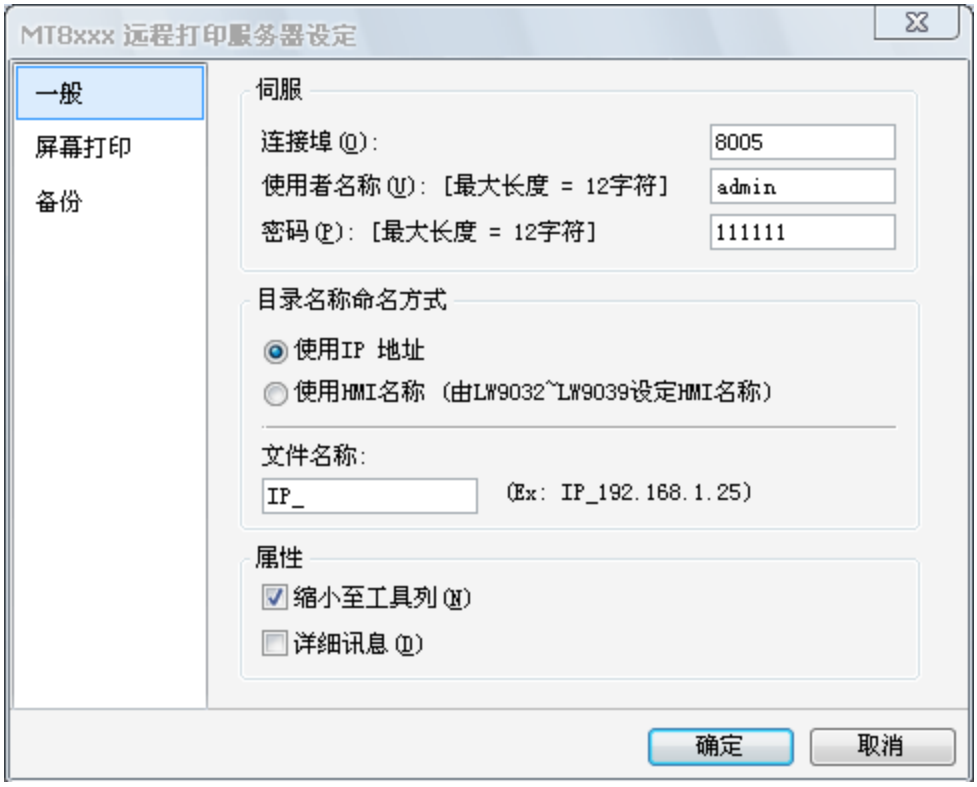
26.3 使用 EasyPrinter 为备份服务器



用户可使用“备份”元件，将历史数据，例如取样数据与事件记录等上传至远程备份服务器。

26.3.1 EasyPrinter 设定程序

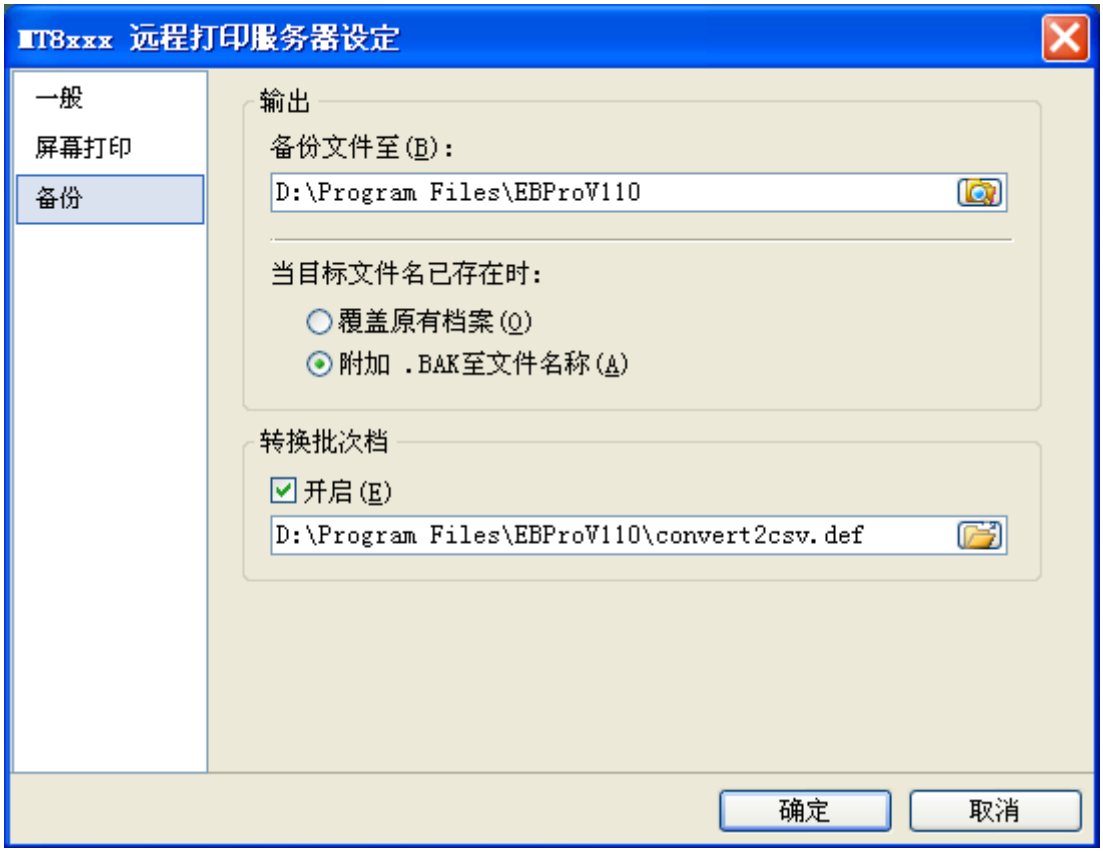
在 EasyPrinter 设定页下点选“选项”»“设置”会出现下面的对话框：



1. 在左列选择“一般”
2. 在“服务器”中，指定“连接埠”为“8005”，”用户名称”为“admin”，”密码”为“111111”。(以上皆为默认值)。
3. 在“目录名称命名方式”，点选”使用 IP 地址”并指定“IP_”为”文件名称”。

4. 在“属性”，选择”缩小至工具列”。

接着设定备份位置。



1. 点选左列“备份”
2. 在“输出”，点击 来浏览及选择历史数据的储存路径。
3. 按下“确定”确认使用以上设定。
4. 在 EasyPrinter 设定页下点选“文件”»“允许输出”，EasyPrinter 会将备份数据储存在方才所选的路径。

26.3.2 EasyBuilder Pro 设定程序

接着在 EasyBuilder Pro 工程文件中加入相关设定：

1. 开启 EasyBuilder Pro，并开启新项目或已使用之项目。
2. 在“编辑”»“系统参数设置” »“打印/备份服务器”中，勾选“使用远程打印/备份服务器”。

3. 在"通讯设置"，输入打印服务器"IP 地址" 同 EasyPrinter 的设定，指定"连接埠"号 "8005"，"用户名称" 为 "admin"，"密码" 为"111111"。 (以上皆为默认值)。
4. 按下 "确定"。

接下来添加备份功能到窗口。

1. 接着在菜单 "元件" 选择 "备份" 会出现下面的对话框：

2. 在“来源”，选择“事件记录”(或照需求选择“RW”、”RW_A”)。
3. 在“备份位置”，选择“远程打印/备份服务器”
4. 在“范围”，选择“今天”和“全部”(或照实际需求改变)。
5. 在“触发”，选择“手动”
6. 按下“确定”
7. 将“备份”元件置于窗口中，例如“4:共享窗口”，用户便可随时执行备份。
8. “编译”及“下载”工程文件至 HMI，按下前面设定的“备份”元件，开始备份历史数据。

Note

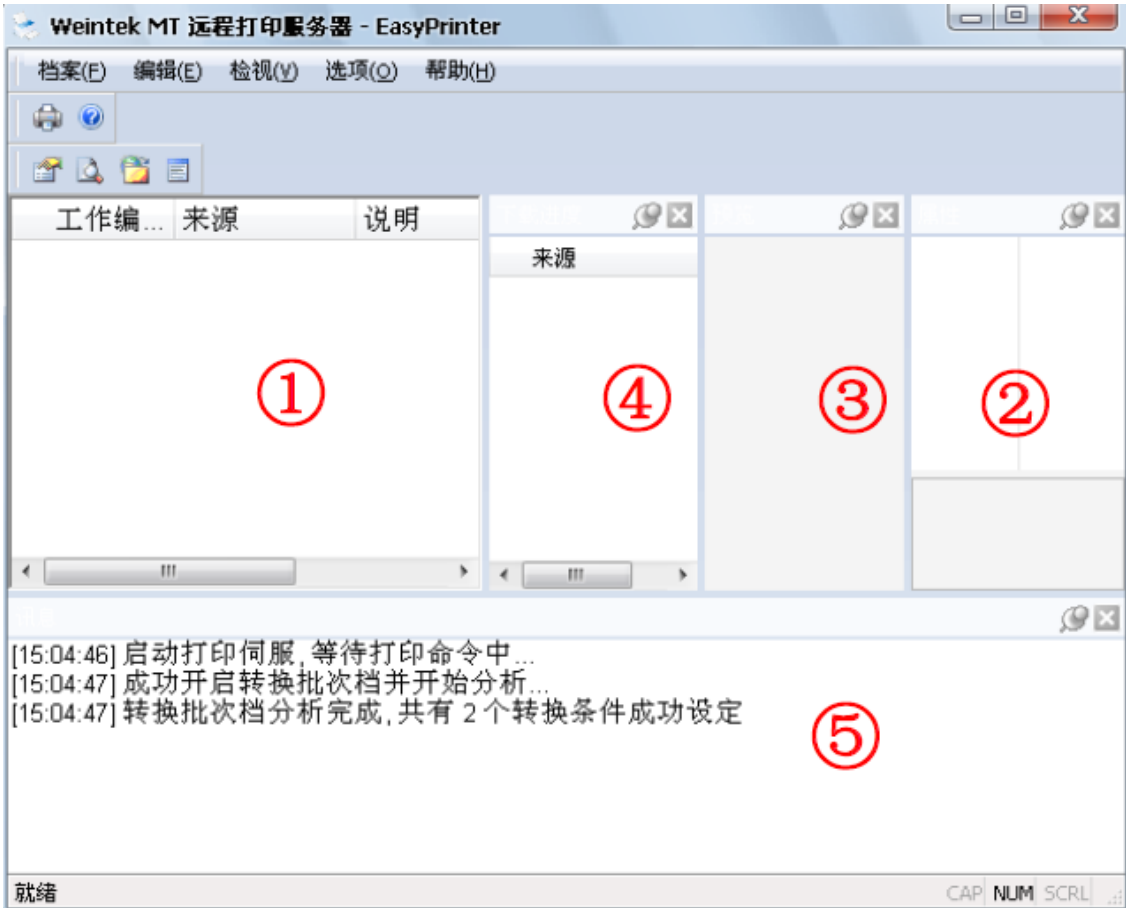
- “备份”元件亦可用位地址触发。
- 用户可以放置一个“排程”元件，在一周的最后一天转为 ON，用以触发“备份”元件自动备份所有历史数据。

26.4 EasyPrinter 操作说明

以下介绍 EasyPrinter 窗口接口和操作说明。

26.4.1 窗口

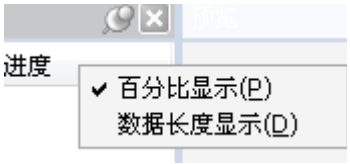
EasyPrinter 窗口可切分为五个区域，如下图。



区域	名称	描述
1	**工作列表**	此視窗顯示所有工作，包括螢幕列印與備份工作。
2	**下载进度窗口**	此視窗顯示新進工作之下載進度。
3	**预览窗口**	此視窗顯示工作列表中所選螢幕列印之預覽影像。
4	**属性窗口**	此視窗顯示工作列表中所選工作之資訊。
5	**讯息窗口**	此視窗顯示工作執行中的時間與訊息，例如密碼錯誤等等。

26.4.2 选单项目

以下说明其它 EasyPrinter 功能。

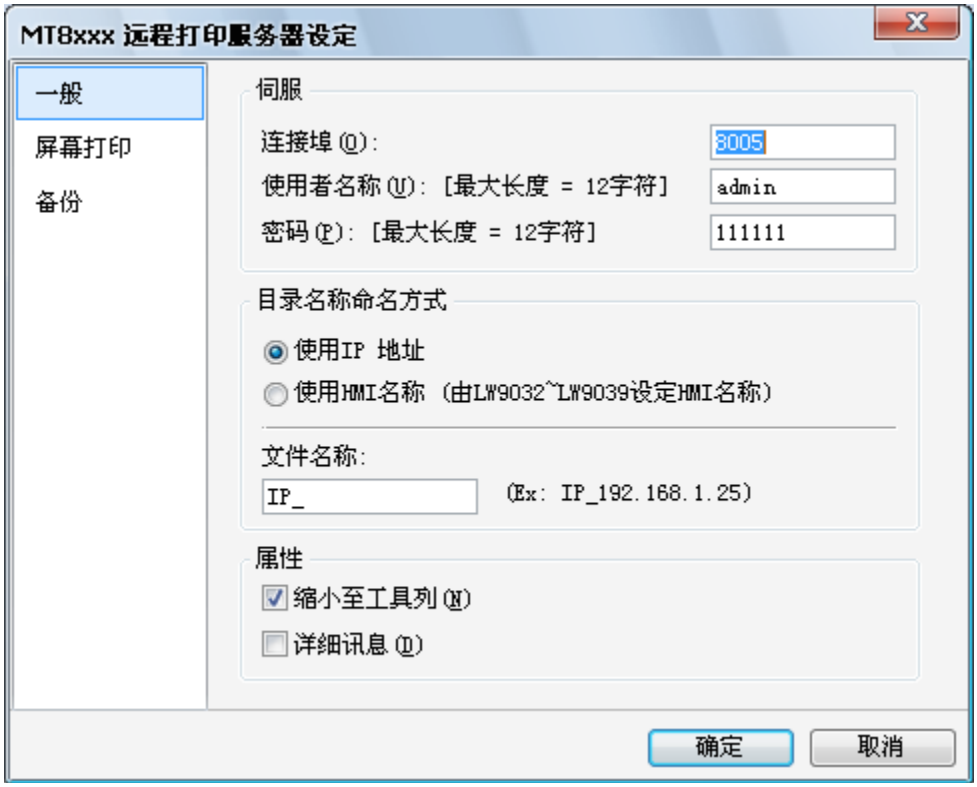
选单项目	描述
文件	**允许输出** 已选：EasyPrinter 执行工作 未选：EasyPrinter 将工作保留于内存中
编辑	**编辑** 编辑“屏幕打印”。用户可以在此自由设定方向、打印大小、边界。 **删除** 永久删除所选工作。 **选择全部** 选择“工作列表”上的所有工作。
检视	**属性窗口** 显示或隐藏属性窗口。 **预览窗口** 显示或隐藏预览窗口。 **下载进度窗口** 用户可以选择下载进度的显示方式。在窗口中，可点击**“进度”**选择依百分比显示或数据长度显示。  **讯息窗口** EasyPrinter 可以保留 10,000 笔讯息窗口中的讯息，当新讯息产生时，最旧的讯息会被删除。
选项	下页详细说明各项设定和意义。

Note

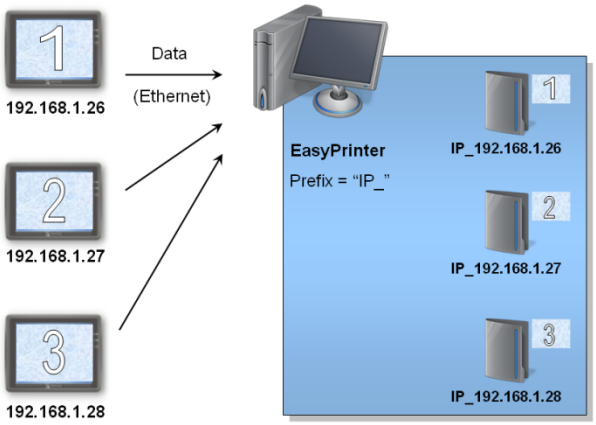
- EasyPrinter 只能将最多至 128MB 的工作资料保留于内存中，若内存已满，所有新进工作会被拒绝。用户可以选择”允许输出”直接执行，或是删除部分工作來清出记忆空间给新进工作。
- 备份工作不可编辑。
- 只有在选择工作后才能”编辑”。
- 选择至少一样工作方可”删除”。

以下详细说明“选项”»“设置”的各项设定和意义。

- 一般页面



设定	描述
伺服	**连接埠** 设定以太网接口让 HMI 连结，范围 1~65535，8005 是默认值。 **用户名称 / 密码** 设定用户名称及密码，让有授权的 HMI 方能使用 EasyPrinter 功能。
目录名称命名方式	EasyPrinter 使用不同的文件夹储存来自不同 HMI 的各种文件(屏幕打印位图文件、备份档等等)。有两种方式可以命名这些文件夹： **使用 IP 地址** 在使用所设 IP 地址的触摸屏送出指令后，EasyPrinter 会为文件夹命名为文件名称+IP 地址。请见下图。



使用 HMI 名称

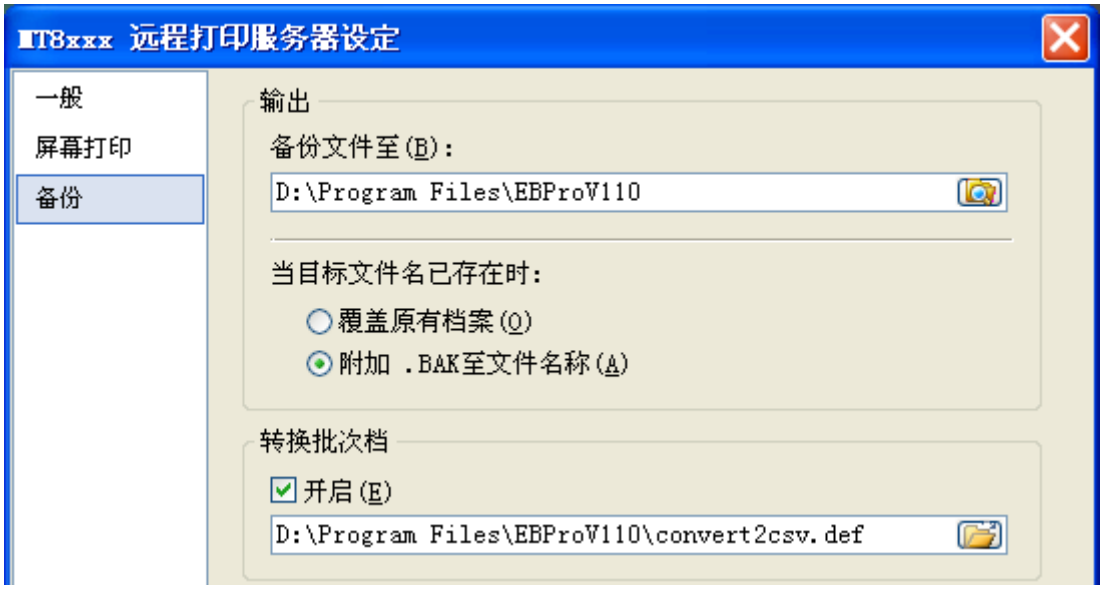
EasyPrinter 使用送出指令的 HMI 之名称来为文件夹命名:"文件名称"+"HMI 名称"。

属性	**缩小至工具列** 勾选此项，EasyPrinter 会被缩小并置于工具列中，用户只要双击工具列上的图像，即可打开 EasyPrinter。 **详细讯息** 勾选此项，讯息窗口中会显示更详细的事件讯息。

- 屏幕打印：

设定	描述
输出	**输出至** 选择此项告知 EasyPrinter 将屏幕打印结果通过特定打印机打印出来。 **储存至** 选择此项告知 EasyPrinter 将屏幕打印结果转成位图档，并储存于指定路径。用户可以在以下路径找到位图文件： "用户指定路径" \ "HMI 文件夹" \ yymmdd_hhmm.bmp 举例来说，当一个屏幕打印指示发生于 17:35:00 2009 年 1 月 12 日，位图档将被命名为"090112_1735.bmp"。如果同一分钟有另一个位图档产生，新图档将被命名为"090112_1735_01.bmp"，以此类推。

- 备份页面



设定	描述
输出	EasyPrinter 将备份文件储存于特定路径下。 事件记录历史数据路径： “用户指定路径” \ “HMI 文件夹” \ “事件记录” \ EL_yyyymmdd.evt 资料取样历史数据路径： “用户指定路径” \ “HMI 文件夹” \ “取样数据记录” \ “资料取样元件的文件名称” \ yyyymmdd.dtl 配方数据备份路径: “用户指定路径” \ “HMI 文件夹” \ “配方数据” \ recipe.rcp 或 recipe_a.rcp
转换批次档	勾选“开启”指定自动将上传之历史文件转档成.csv 或.xls (Excel) 档之转换批次档。请参考下一小节。

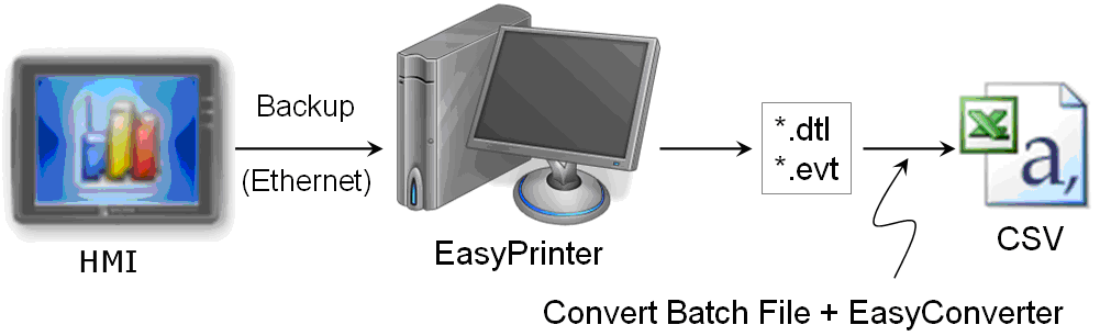
Note

- 用户可以用寄存器 LW-9032 至 LW-9039 来指定 HMI 名称。
- 若尚未设定 HMI 名称，EasyPrinter 会使用 IP 地址来命名文件夹。

26.5 转换批次档

EasyPrinter 提供一个转文件设备，可将上传之资料取样与事件记录等两种历史文件自动存成 .csv 档。用户若需要此功能，须先准备一个转换批次档，告知 EasyPrinter 如何转换历史文件。

如下所示，此转文件功能实际是由 EasyConverter 执行，EasyPrinter 只是遵照转换批次档的标准来用正确的参数启动 EasyConverter 达成转档指示。



Note

- EasyConverter 是另一个 Win32 应用程序，可将历史数据转换成.csv 或 Excel 的 .xls 等文件。用户可在 EasyBuilder Pro 下载路径中找到这个程序。
- 用户若需使用此功能，须先确定 EasyPrinter 及 EasyConverter 皆被放置于相同路径下。

26.5.1 转换批次文件默认值

以下为预设转换批次档。

程序代码 1 预设转换批次档 (convert2csv.def)

```
1: "dtl", "EasyConverter /c $(路径名称)"
2: "evt", "EasyConverter /c $(路径名称)"
```

文件文字会以两行呈现，每行含有两个参数，用逗号分隔，形成对应特定类型文件(资料取样与事件记录)的处理标准。第一个参数显示该文件类型的扩展名，第二个参数显示操作模式所需执行的命令。“$(路径名称)”是关键词，告诉 EasyPrinter 需用转档的备份文件名称来取代之。例如，资料取样文件名称为 20090112.dtl，已被上传及储存，EasyPrinter 会输出以下指令于命令窗口。

```
1: EasyConverter /c 20090112.dtl
```

如此一个名称为 20090112.csv 的文件即被建立。

因此，转换批次档的预设标准如下:

1. 转换所有资料取样历史文件(.dtl)为 .csv 檔。
2. 转换所有事件记录历史文件(.evt)为.csv 檔。

Note

- 事实上，第二个参数中的“$(路径名称)”代表文件的完整路径名称，在前面的例子中，EasyPrinter 以下面名称取代：
  “用户指定路径”\”HMI 文件夹”\”数据记录”\”资料取样元件文件名称”\20090112.dtl
- EasyPrinter 以一行文件文字为单位来解读转换批次档，也就是说，一行文字形成一个标准。
- 任两个参数皆须以逗号分隔。
- 任一参数皆须以双引号标示。

- 同一参数中切勿放逗号。

详细信息请参考《25 EasyConverter》。

26.5.2 特定标准

请参考以下特定标准：

- 可针对特定 HMI 上传的文件使用特别的操作，例如程序代码 2。
- 可用 HMI 的名称来辨别该 HMI，例如程序代码 3。
- 或是针对不同的资料取样历史记录需要不同的操作方式，例如程序代码 4。
  (只适用在 “资料取样” 文件，且名称为"Voltage"时。)
  第三个参数(“*”)表示接受来自任何 HMI 的资料取样当中符合标准者。
  用户可以转换第三个参数为"192.168.1.26"，"192.168.1.*"，HMI 名称等等，用以减少目标 HMI 范围。

程序代码 2 针对 HMI IP = 192.168.1.26 的特别定义标准

```
1: "dtl", "EasyConverter /c $(Pathname)", "192.168.1.26"
```

程序代码 3 针对 HMI 名称为 Weintek_01 的特别定义标准

```
1: "dtl", "EasyConverter /c $(Pathname)", "Weintek_01"
```

程序代码 4 针对资料取样元件文件名称 = Voltage 的特别定义标准

```
1: "dtl", "EasyConverter /s Voltage.lgs $(Pathname)", "*", "Voltage"
```

26.5.3 转换批次档格式

以下列出标准格式与各参数之说明。

文件类型	指令(行)	HMI IP/名称	条件 1	条件 2

- 文件类型
  这个参数特定出此标准所针对的上传文件类型之扩展名(e.g. “.dtl” 为资料取样历史文件，“.evt”为事件记录历史文件)
- 指令(行)
  当上传文件符合标准时，EasyPrinter 送至命令窗口的确切指令。
- HMI IP/名称
  此参数指定这个标准所针对的 HMI。
- 条件 1
  如果文件类型是“.dtl”， 此参数将这个标准所针对之”资料取样”元件的文件夹名称指定出来。对其它文件类型无效。
- 条件 2
  未使用 (保留为以后使用)

26.5.4 执行顺序

EasyPrinter 于每个文件上传后，由下往上验证各标准。一旦符合标准，就会停止验证，并开始处理下一个文件。因此，**用户可将较广泛的标准放在下方，而将较明确的标准置于上方**。例如，以下为批次档内容：

"dtl", "EasyConverter /c $(路径名称)"
"evt", "EasyConverter /c $(路径名称)"
"dtl", "EasyConverter /c $(路径名称)", "192.168.1.26"
"dtl", "EasyConverter /c $(路径名称)", "my_HMI_01"
"dtl", "EasyConverter /c $(路径名称)", "my_HMI_02"
"dtl", "EasyConverter /s Voltage.lgs $(路径名称)", "*", "Voltage”

其正确的顺序为 (由最后一行往上执行):
"dtl", "EasyConverter /s Voltage.lgs $(路径名称)", "*", “Voltage"
"dtl", "EasyConverter /c $(路径名称)", "my_HMI_02"
"dtl", "EasyConverter /c $(路径名称)", "my_HMI_01"
"dtl", "EasyConverter /c $(路径名称)", "192.168.1.26"
"dtl", "EasyConverter /c $(路径名称)"
"evt", "EasyConverter /c $(路径名称)"

第二十七章 EasySimulator

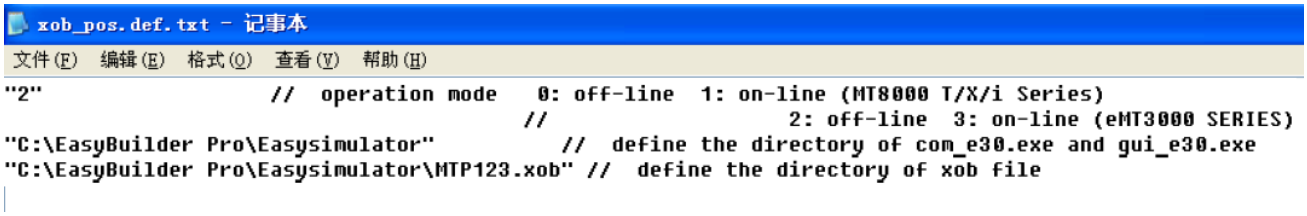
27.1 概要

EasySimulator 功能允许用户可以不用安装整个 EasyBuilder Pro 软件，即可执行在线/离线模拟，但用户须先备好所需的相关文件。请依照下面步骤完成设定。

27.2 设定 EasySimulator 的步骤

1. 准备相关文件：
- [driver] → [win32]
- com_e30.exe
- EasySimulator.exe
- gui_e30.exe
- sqlite3.dll
- xob_pos.def

2. 使用文字编辑工具开启 xob_pos.def (例如：记事本)，并正确设定相对内容。



行数	描述
1	“2” 执行离线模拟；“3” 执行在线模拟。
2	指出相关文件的存放路径。 (例如：com_e30.exe, gui_e30.exe, EasySimulator.exe…等等)。
3	指出 .exob 文件的存放路径。

3. 双击 EasySimulator.exe 即可开始执行模拟。
4. 联机 / 离线模拟结果将显示在屏幕上。

Note

- 以上相关文件皆可在 EasyBuilder Pro 安装目录文件夹里找到，所以用户必须在某部计算机上安装 EasyBuilder Pro 软件后，再将这些文件复制到目标计算机上。
- 若执行 EasySimulator.exe 后没有反应，请再次确认相关文件路径是否定义正确。

- 若弹出“Failed to open project file：No such file or directory.”的窗口，表示 .exob 文件路径错误，请再次确认是否定义正确。

第二十八章 使用串行端口实现一机多屏功能 (主从模式)

28.1 概要

使用串行端口实现一机多屏是指：触摸屏通过串行端口连接远端的触摸屏，并读取连接在远端触摸屏上 PLC 的数据，参考下图。

上图显示 PLC 连接在 HMI 1 上，HMI 1 与触摸屏 2 使用串行端口直接连接，HMI 2 可以通过触摸屏 1 读取 PLC 上的数据。

下面说明如何使用 EasyBuilder Pro 设定触摸屏 1 与 HMI 2 所使用的工程文件，实现一机多屏功能。

28.2 设定主机所使用的工程文件内容

下图为触摸屏 1 所使用工程文件“系统参数”中“设备清单”的内容。

1. 因为触摸屏 1 的 COM 1 连接 PLC ，所以设备清单中需存在“本机 PLC1“，并设定正确的 PLC 通讯参数。假设此时所连接的 PLC 为 FATEK FB Series。
2. 因 HMI1 的 COM 3 用来接收来自 HMI2 的命令，所以必须建立“Master-Slave Server”类型的设

备，用来设定 COM 3 的属性。

由上图可以发现 COM 3 的通讯参数为“115200,E,8,1”，并使用 RS-232 接口。此项参数并不限定需与 PLC 的通讯参数相同，但限制数据位必须为 8。另外，尽可能设定为较快的通讯速度，这样触摸屏 2 可以较有效率读取到 PLC 的数据。

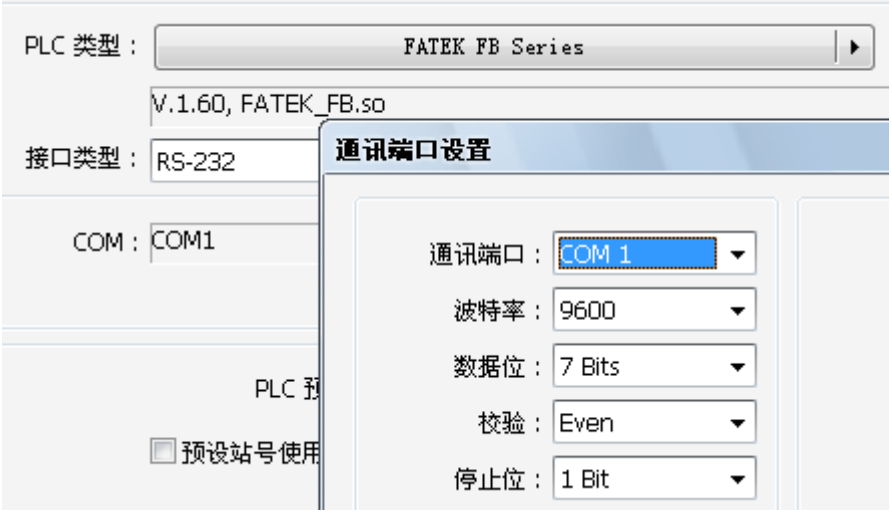
28.3 设定从机所使用的工程文件内容

下图为触摸屏 2 所使用工程文件“系统参数”中“设备清单”的内容。

因为触摸屏 2 所读取的 PLC 连接在触摸屏 1 上，所以触摸屏 2 将 PLC 视为远程 PLC，因此在设备清单中需存在“*远程 PLC 1“，此时所连接的 PLC 为 FATEK FB Series。下文说明如何建立”*远程 PLC 1“。

1. 在设备清单中建立一个新的设备，”PLC 类型”请选择“FATEK FB Series”，“PLC 预设站号”需与 PLC 所使用的站号相同。

2. 设定正确的通讯参数。此时触摸屏 2 的 COM 1 是与触摸屏 1 的 COM 3 相互连接，并不是与 PLC 直接连接，因此必须忽略 PLC 的通讯参数，而应让触摸屏 2 的 COM 1 与触摸屏 1 的 COM 3 所使用的接口与通讯参数相同。因为触摸屏 1 的 COM 3 使用 RS-232 接口，通讯参数为“115200,E,8,1”，所以触摸屏 2 的 COM 1 也需依此参数设定，参考下图。



3. 因为触摸屏 2 视 PLC 为远程 PLC ，所以需选择“所在位置”为“远程”。并选择使用“串行端口”的方式连接远程触摸屏(即触摸屏 1)。

4. 完成上述的各项步骤后，在“设备清单”中可以发现新增一项设备：“*远程 PLC 1“。此设备名称包含 '*' 符号，用来表示即使名称中包含“远程”，但实际上仍由本机的串行端口发送命令与接收回复，所以与 PLC 的连接状态只需检视本机的系统保留地址即可；也就是“*远程 PLC 1”、”*远程 PLC 2”、”*远程 PLC 3” 与 “本机 PLC 1”、”本机 PLC 2”、”本机 PLC 3”使用相同的系统保留地址。这些系统保留地址包含，如下表所示：

地址	描述
LB-9150	状态为 ON 时，若与连接在 COM 1 的 PLC 断线，系统将自动联机。 状态为 OFF 时，忽略与此 PLC 的断线状态。
LB-9151	状态为 ON 时，若与连接在 COM 2 的 PLC 断线，系统将自动联机。 状态为 OFF 时，忽略与此 PLC 的断线状态。
LB-9152	状态为 ON 时，若与连接在 COM 3 的 PLC 断线，系统将自动联机。 状态为 OFF 时，忽略与此 PLC 的断线状态。
LB-9200~ LB-9455	这些寄存器用来指示与连接在 COM 1 的 PLC 间的联机状态。LB-9200 指示与站号为 0 的 PLC 的联机状态，LB-9201 指示与站号为 1 的 PLC 的联机状态，依此类推。 状态为 ON 表示目前联机正常。 状态为 OFF 表示目前与 PLC 为断线状态，此时可以将此状态重设为 ON，系统将尝试与 PLC 再联机一次。
LB-9500~ LB-9755	这些寄存器用来指示与连接在 COM 2 的 PLC 间的联机状态。LB-9500 指示与站号为 0 的 PLC 的联机状态，LB-9501 指示与站号为 1 的 PLC 的联机状态，依此类推。 状态为 ON 表示目前联机正常。 状态为 OFF 表示目前与 PLC 为断线状态，此时可以将此状态重设为 ON，系统将尝试与 PLC 再联机一次。
LB-9800~ LB-10055	这些寄存器用来指示与连接在 COM 3 的 PLC 间的联机状态。LB-9800 指示与站号为 0 的 PLC 的联机状态，LB-9801 指示与站号为 1 的 PLC 的联机状态，依此类推。 状态为 ON 表示目前联机正常。 状态为 OFF 表示目前与 PLC 为断线状态，此时可以将此状态重设为 ON，系统将尝试与 PLC 再联机一次。

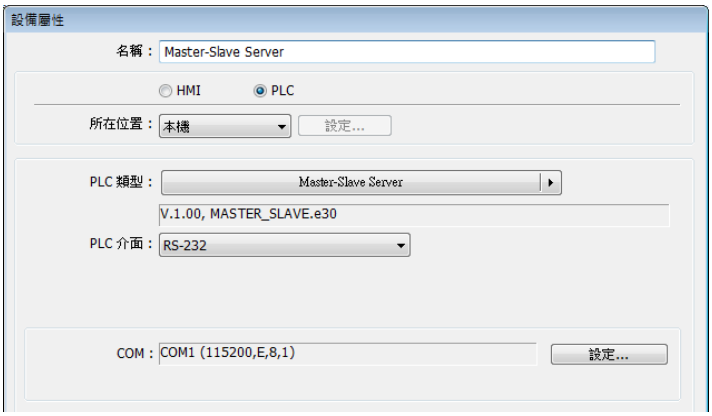
28.4 如何连结从机的 MT500 工程文件

目的是让 MT500 可以使用 EasyBuilder 的主从通讯协议使得 eMT3000 系列的 Local Data 和 MT500 所连接的 PLC Data 可以交换信息。

28.4.1 EasyBuilder Pro 设定

1. 选取“Master-Slave Server”驱动并点击“设置”。若有连接 PLC ，则依照原本设定方式即可。

2. 选取 RS-232 并点击“设置”。



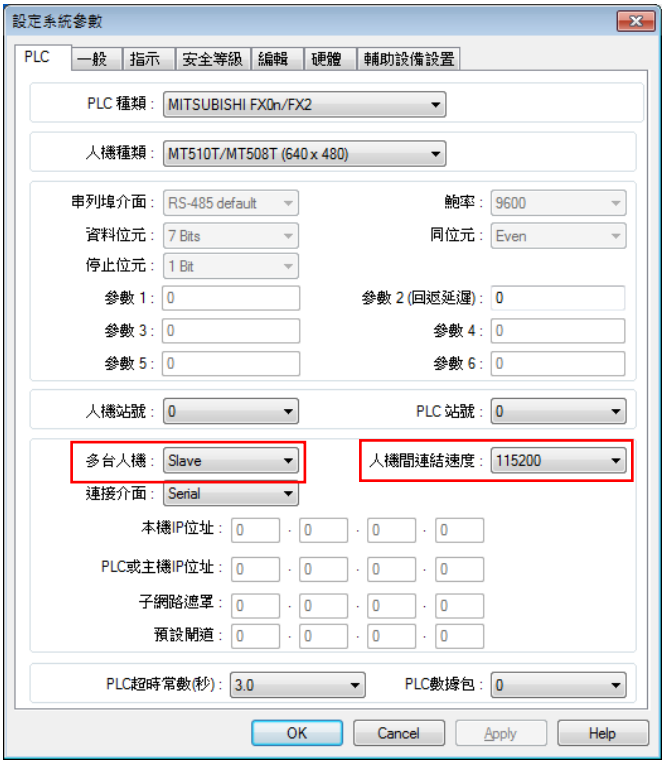
3. 参数 1 要填入 MT500 PLC ID No. (请参考 MT500 设定)。

28.4.2 EB500 设定

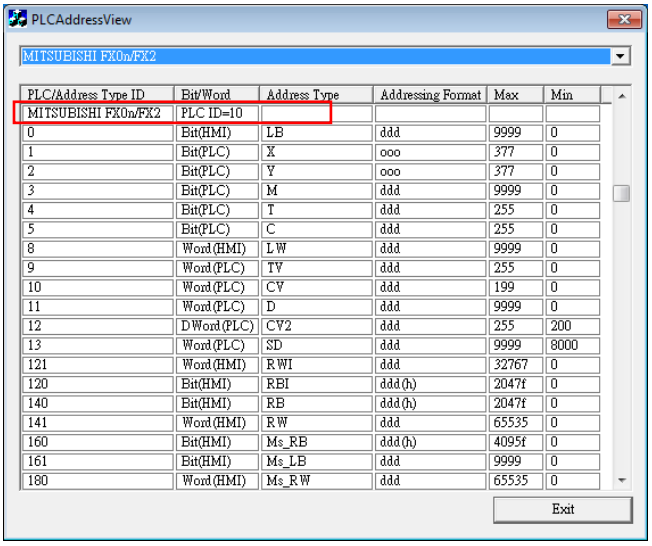
1. 在 EB500 的系统参数设定内设定多台触摸屏: Slave，触摸屏间连结速度: 115200。

Note

- Baud rate 的设定在 EB500 及 EasyBuilder Pro 要相同。



2. 双击 PLC Address View.exe 来查询 PLC 的 ID No. 并填入至 EasyBuilder 的参数 1。



3. HMI 之间使用串行端口 RS-232 连接后即可通讯。

Note

- 因为 MT500 参数设定一定会有某台 PLC 被选定，因此即使只要读写 eMT3000 的 Local Data，仍需要将 MT500 参数设定所选中的 PLC ID 填入到 EasyBuilder 的参数 1。
- 以下方式，主从模式不适用。当 MT500 选用 S7-200、S7-300 驱动时，因 MT500 将高低位互换，而导致 MT500 读取 eMT3000 的 Local Data 时，发生错误。

下表比较 MT500 与 eMT3000 的地址。

设备地址	MT500	eMT3000	范围
位	Ms_RB	RW_Bit	dddd: 0~4095 (h): 0~f
位	Ms_LB	LB	dddd: 0~9999
字符	Ms_RW	RW	ddddd: 0~65535

第二十九章 穿透通讯功能

29.1 概要

穿透通讯功能允许电脑上的应用程序通过触摸屏直接控制 PLC；此时 HMI 的功能类似转接器。
穿透通讯功能包含以下两种模式：

- 以太网
- 串行端口

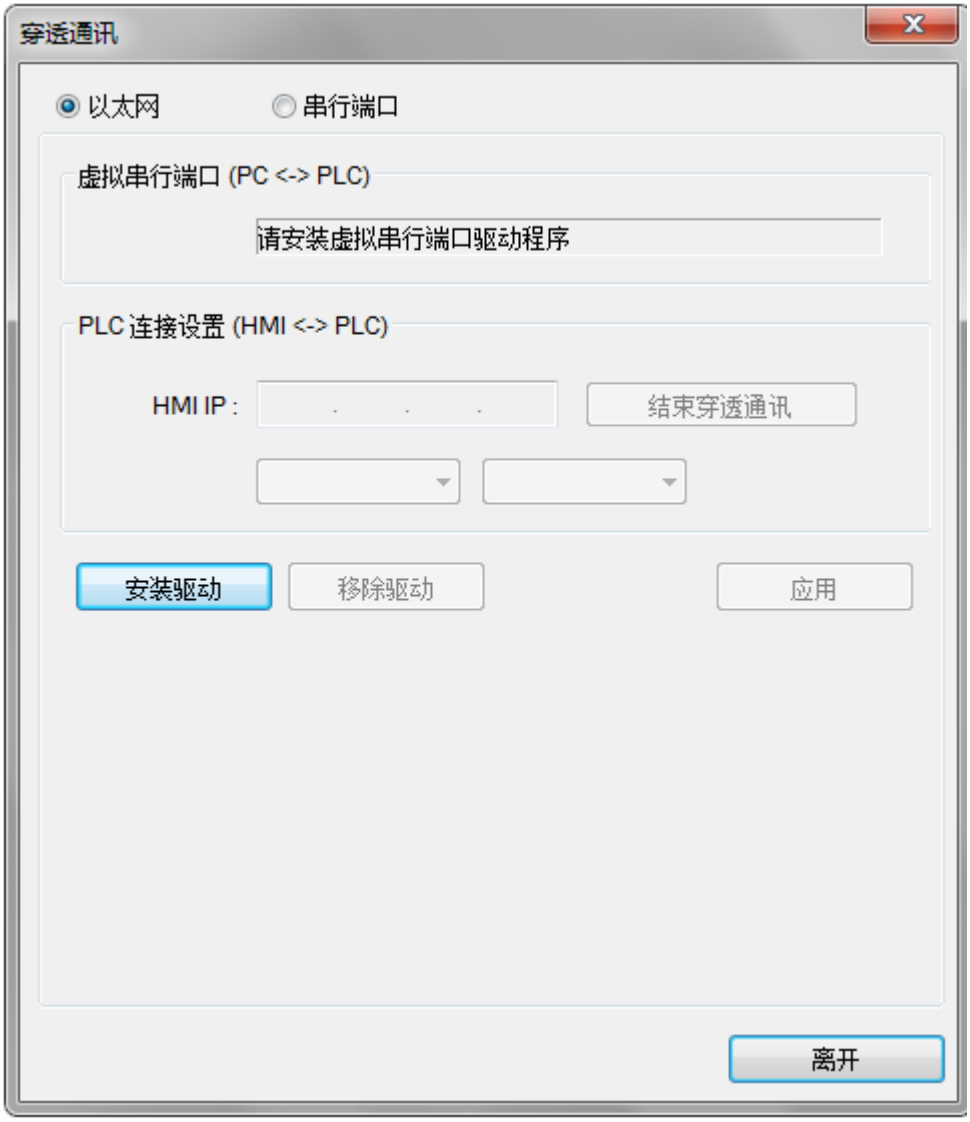
点击 Utility Manager 的“穿透通讯设置” 按钮，即可检视这两种模式的设定内容。

29.2 以太网模式

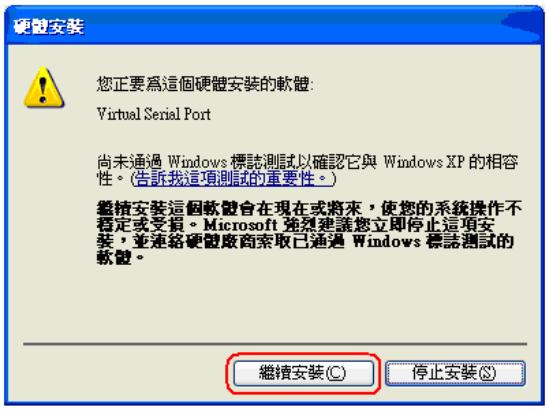
29.2.1 安装虚拟串行端口驱动程序的步骤

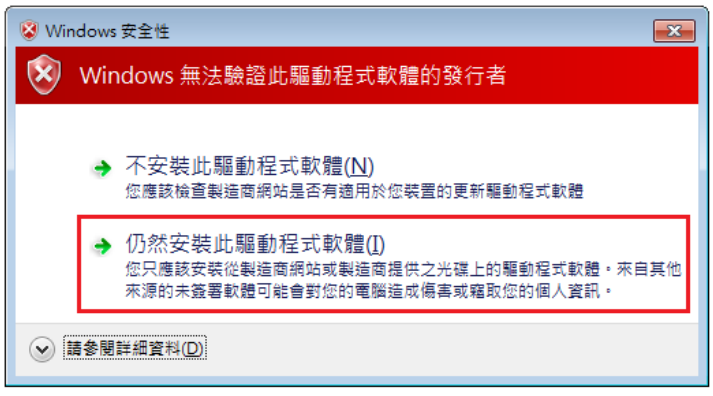
在使用“以太网”穿透通讯功能前，要先安装虚拟串行端口驱动程序。

1. 开启 Utility Manager 检视目前驱动程序的安装状态，若画面显示“请安装虚拟串行端口驱动程序”，请点击“安装驱动”按钮。



2. 在安装驱动程序的过程中可能被要求确认安装，请选择“继续安装”。

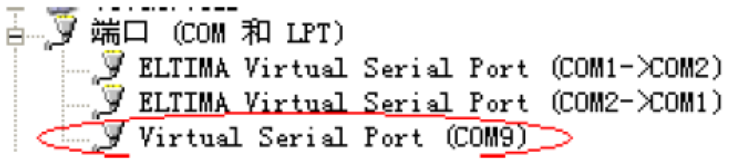




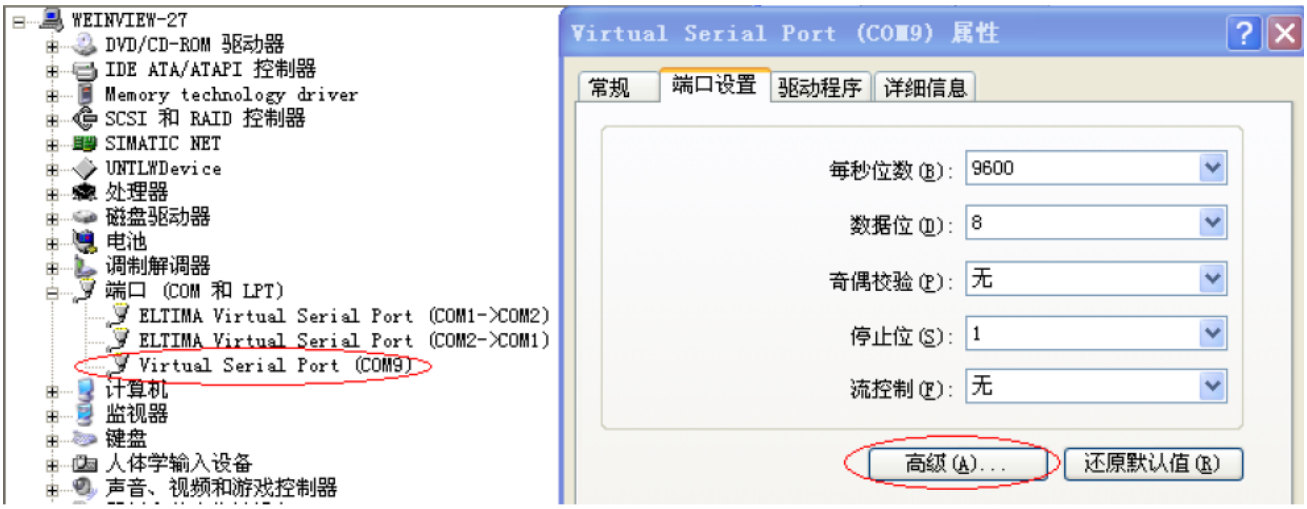
3. 在完成安装程序后，原先显示“请安装虚拟串行端口驱动程序”的位置将显示目前所使用的虚拟串行端口号。

29.2.2 更改虚拟串行端口的步骤

1. 在“设备管理员”的内容中可发现已安装完成“Virtual Serial Port”。



2. 若要更改虚拟串行端口号，只需进入“Virtual Serial Port”的内容，并选择“连接埠设置” 下的“进阶”，即可更改虚拟串行端口号。

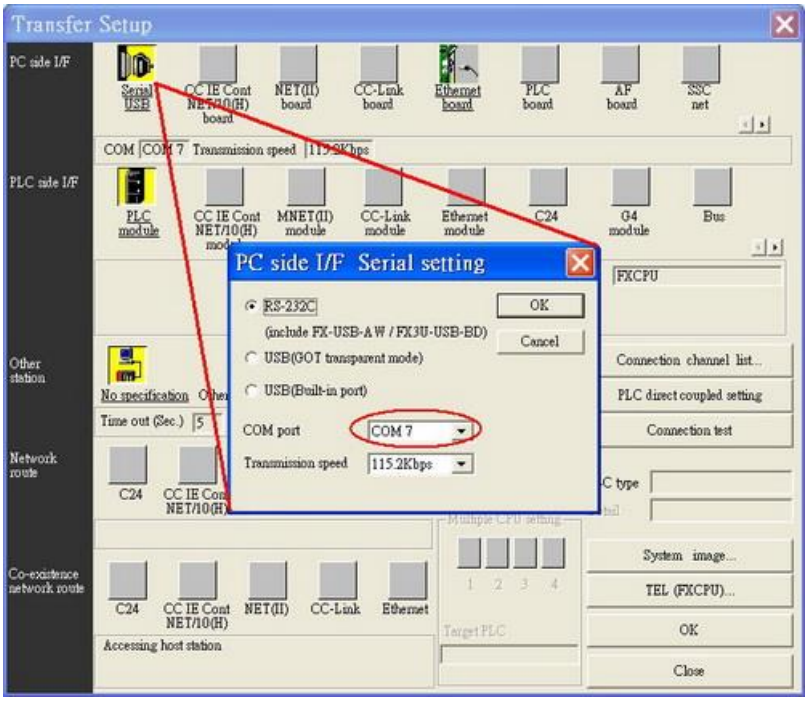


29.2.3 以太网模式设定

在安装完成虚拟串行端口驱动程序后，只需依照下面步骤即可使用网络穿透通讯功能。

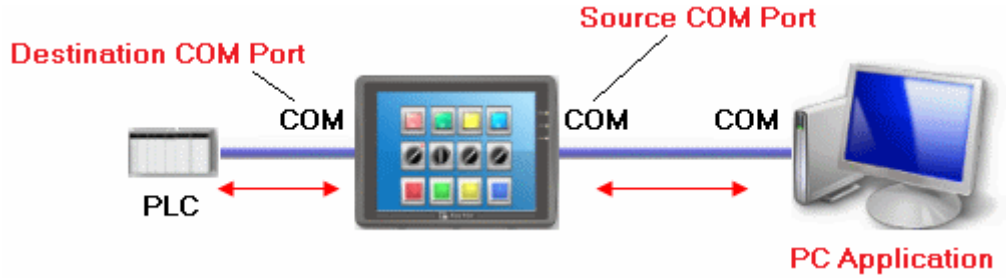
1. 请设定连接 PLC 的触摸屏 IP 地址，下图显示目前触摸屏 IP 地址为 192.168.1.206。
2. 指定触摸屏连接 PLC 的串行端口与属性，下图显示此时使用 COM 2 RS-232 通讯模式。
3. 完成所有设定后需按下“应用”按钮，所有属性才会生效。

4. 在执行电脑 上的应用程序时，所使用的串行端口号需指向虚拟串行端口号。以 Mitsubishi 的应用程序为例，若此时的虚拟串行端口为 COM 7，则在“PC side I/F Serial setting”窗口中的“COM port”需选择 COM 7。



5. 完成上述各项设定后，用户在执行电脑 上的 PLC 应用程序时，HMI 会自动切换为穿透通讯模式 (此时将暂停触摸屏与 PLC 间的通讯)，此时可以将应用程序视为直接使用虚拟串行端口控制 PLC。在关闭应用程序时，HMI 也会自动关闭穿透通讯模式。

29.3 串行端口模式



“数据来源串口”是指与电脑连接的触摸屏端口。

“数据目标串口”是指与 PLC 连接的触摸屏端口。

在使用“串行端口”穿透通讯功能前，需先正确设定这两个串口的属性。

29.3.1 串行端口设定

启用“串行端口”穿透通讯功能的方式有两种。

- 使用 Utility Manager
- 使用系统寄存器

LW-9901 设定数据来源串口 (1 ~ 3: COM 1 ~ COM 3)

LW-9902 设定数据目标串口 (1 ~ 3: COM 1 ~ COM 3)

29.3.2 使用 Utility Manager

1. 在 Utility Manager 按下“穿透通讯”按钮并设定通讯参数如下图所示。

设定	描述
HMI IP	需指定触摸屏的 IP 地址。
读取触摸屏通讯参数设定	读取触摸屏上数据来源串口与数据目标串口的各项设定值。在按下“读取触摸屏通讯参数设置” 按钮后，所有通讯参数将被更新。
数据来源 COM 端口(PC->HMI) /数据目标 COM 端口(HMI->PLC)	显示与设定数据来源串口与数据目标串口的通讯参数。当点选“开始穿透通讯”，将根据“数据来源 COM 端口”、”数据目标 COM 端口”中所设定的内容，执行穿透通讯功能。
传输速率 / 数据位 / 校验 / 停止位	通常“数据来源 COM 端口”与“数据目标 COM 端口”中的这些设定需相同。”数据来源 COM 端口”因为是连接到电脑，通讯模式通常选择为 “RS-232” 即可；”数据目标 COM 端口”因为接到 PLC，通讯模式需依照 PLC 的通讯设定，可选择： “RS-232”、“RS-485 2W”或 “RS-485 4W”。

Note

- 若不需要穿透通讯功能，需点选“结束穿透通讯”来关闭穿透通讯功能，此时触摸屏才会重新开启和 PLC 的通讯。

共有三种模式可显示目前触摸屏的工作模式。

模式	描述
未知	在未读取触摸屏的设定值前，所显示的触摸屏工作状态。
正常模式	在读取触摸屏的设定值后，显示的状态。触摸屏处在正常通讯状态，不接受来自数据来源串口的任何数据。
穿透模式	HMI 目前正处在穿透模式状态。此时电脑 上的应用程序可以通过数据来源串口直接控制连接在数据目标串口上的 PLC。

29.3.3 使用系统寄存器

另一种启动触摸屏穿透通讯功能的方式为直接更改系统寄存器 LW-9901 (数据来源串口) 与 LW-9902 (数据目标串口) 中的数据内容。当 LW-9901 与 LW-9902 中的数据符合下列条件时，HMI 将自动启动穿透通讯功能。

- LW-9901 与 LW-9902 中的数据需为 1 ~ 3 (1 ~ 3 分别表示 COM 1 ~ COM 3)。
- LW-9901 与 LW-9902 中的数据不可相同。

如有需要更改各串口的通讯参数，只需更改各参数相对应的系统寄存器中的数据，并对”LB-9030：更新 COM 1 通讯参数”、”LB-9031：更新 COM 2 通讯参数”、”LB-9032：更新 COM 3 通讯参数” 送出 ON 的讯号即可。

Note

- 若要关闭触摸屏的穿透通讯功能，只需将 LW-9901 与 LW-9902 中的数据更改为 0。

29.4 穿透通讯控制

一般来说，开启穿透模式时，HMI 会关闭与 PLC 间的通讯，直到穿透模式结束。然而，特定的 PLC 驱动程序于穿透模式下可支持 HMI- PLC,电脑-HMI 同时通讯。

要查询支持的 PLC 驱动，请参阅《PLC 连结手册》的相关章节。

同时通讯功能可利用系统寄存器 LW-9903 进行控制。

LW-9903 数值	描述
0 (预设)	正常模式。执行穿透功能时，HMI- PLC,电脑-HMI 可以同时通讯。
2	执行穿透功能时，将停止 HMI 与 PLC 间的通讯。

Note

因串行端口通讯速度受限，可设定 LW-9903 为 2 关闭此功能来加快上传/下载 PLC 程序的速度。

第三十章 工程文件保护功能

30.1 概要

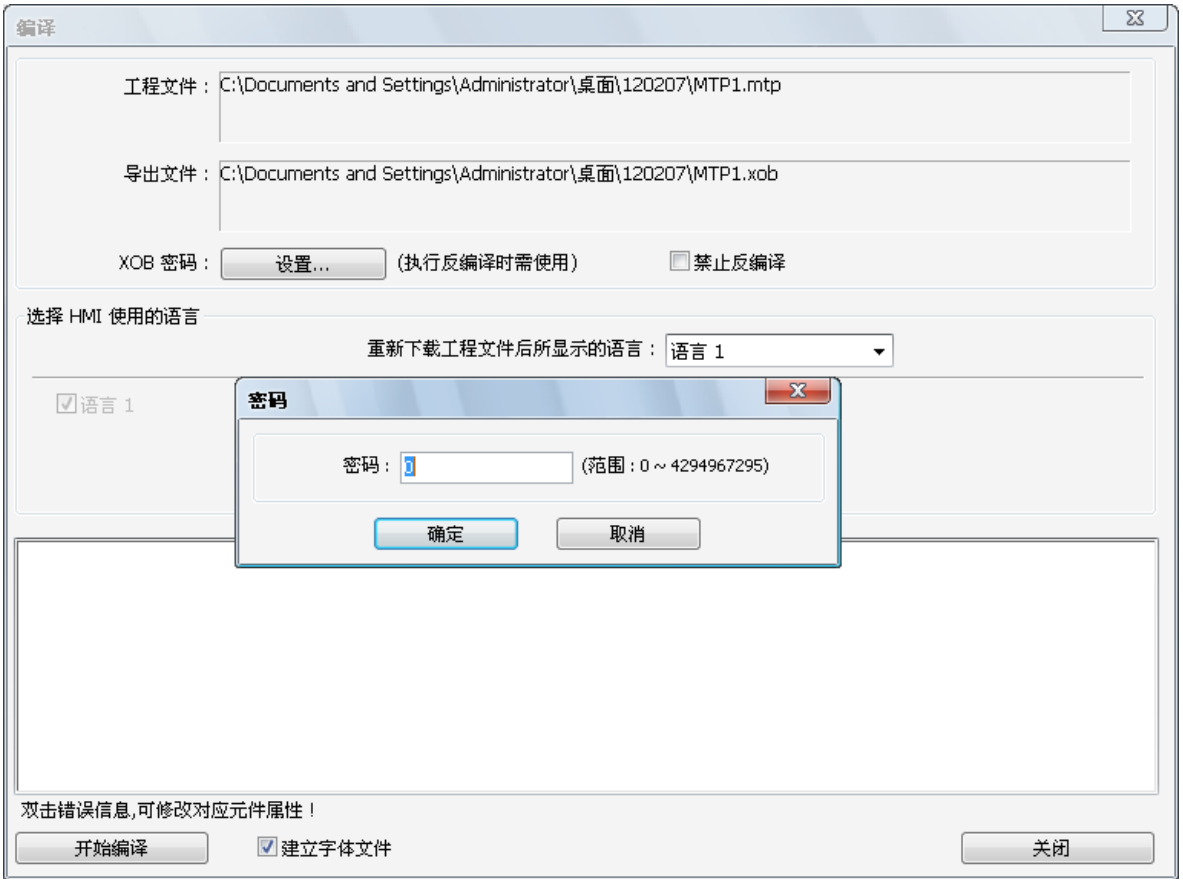
EasyBuilder Pro 提供许多工程文件的安全加密功能，完整保护工程文件。

Note

- 以下介绍的安全加密功能，若不慎忘记密码时，因已由用户自行加密，所以原厂也无法解开，请妥善保管密码。

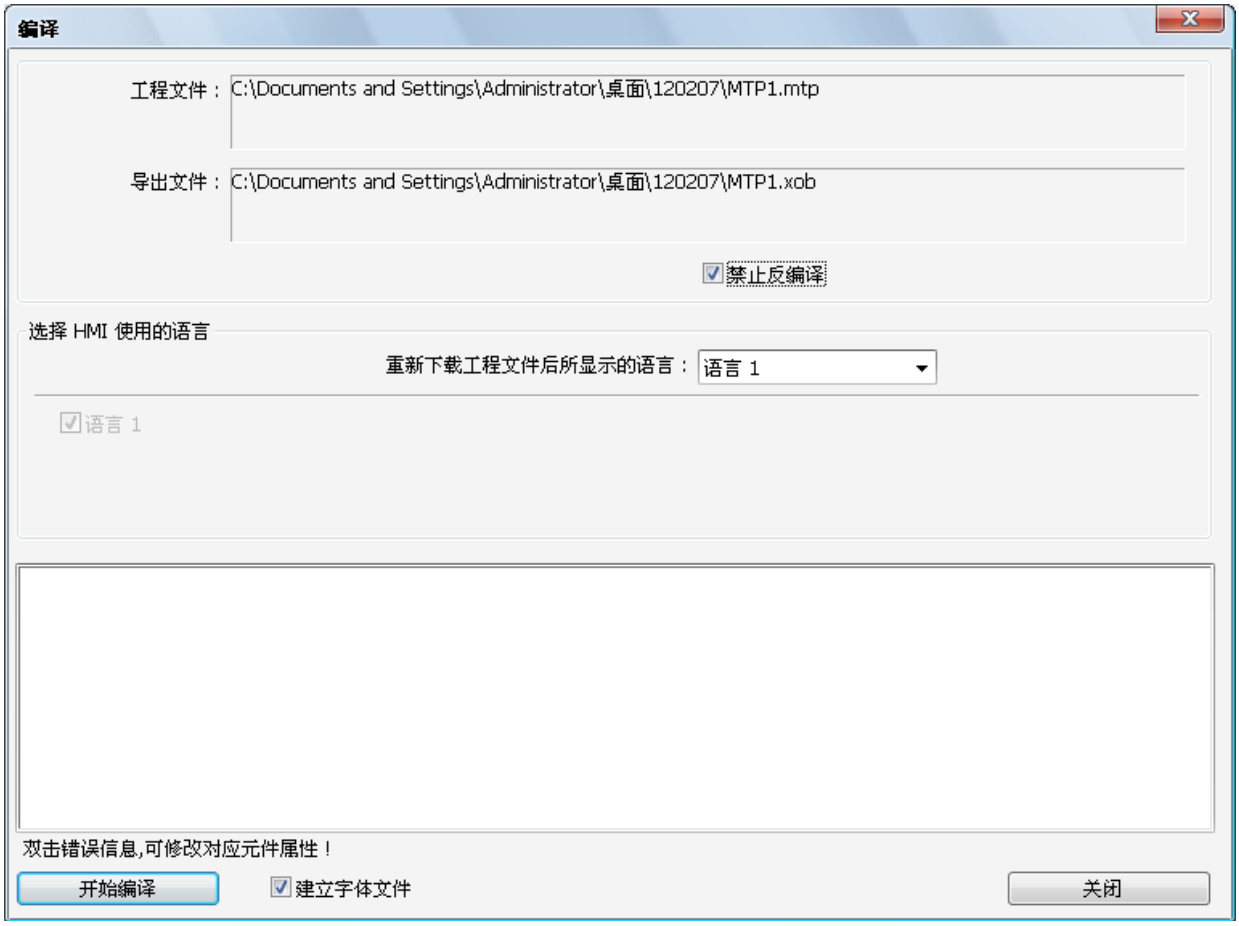
30.2 EXOB 密码

在编辑完成工程文件 (.emtp) 后，可通过系统提供的编译功能，将 .emtp 工程文件编译为下载至触摸屏所需的 .exob 文件。用户可在编译窗口设定 “EXOB 密码”(范围: 0 ~ 4294967295)，若之后要将此 .exob 反编译成 .emtp 工程文件，必需输入此密码才能完成反编译。若反编译时密码输入错误三次，请重新启动 EasyBuilder Pro。



30.3 禁止反编译

在编辑完成工程文件 (.emtp) 后，可通过系统提供的编译功能，将 .emtp 工程文件编译为下载至触摸屏所需的 .exob 文件。用户可在编译窗口设定“禁止反编译”，系统将忽略“EXOB 密码”的设定，且此 .exob 文件将无法反编译回 .emtp 工程文件。

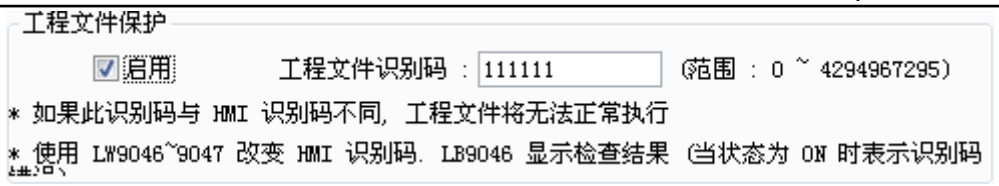


30.4 禁止 EXOB 文件上传功能

EasyBuilder Pro 提供系统寄存器 LB-9033 作为禁止 EXOB 文件上传的功能。当此地址被设定为 ON 时，即上传后得到的 .exob 文件大小将为 0 字节而无法反编译。此外，当变更任何设定时，必须重新启动触摸屏以更新设定值。

30.5 工程文件识别码

用户的工程文件可以被限制只能在特定的触摸屏上执行，下图为“系统参数设置”»“一般属性”页面中“工程文件保护”的设定画面。



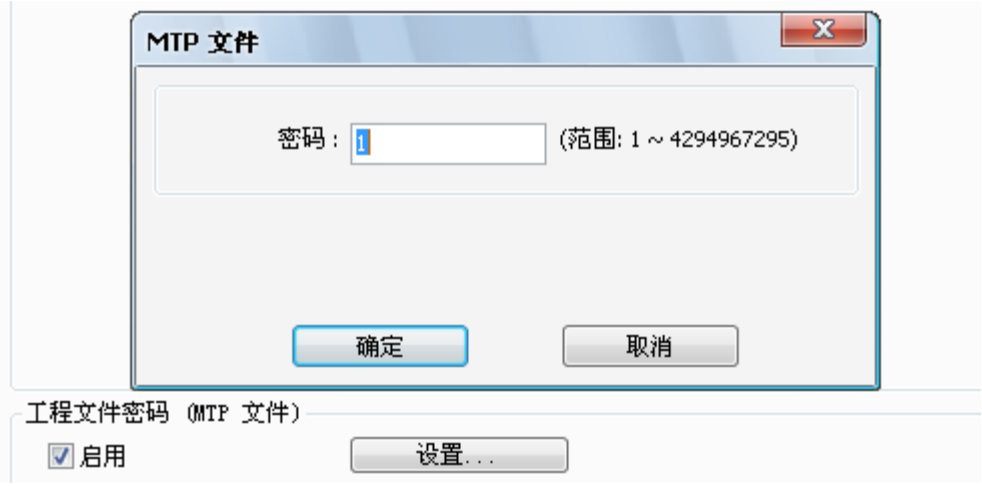
当启用“工程文件保护”功能时，可设定“工程文件识别码”(范围: 0 ~ 4294901750)，且需搭配系统寄存器 LW-9046 及 LW-9047 (32-bit) 设定触摸屏的“触摸屏识别码”，其数据无法被读取或远程写入。而编译后所得到的 .exob 文件只能在“触摸屏识别码”与“工程文件识别码”相同的触摸屏上执行。若“触摸屏识别码”与“工程文件识别码”不相同时，LB-9046 的状态将被设定为 ON。当变更“触摸屏识别码”设定时，必须重新启动触摸屏以更新设定值。

Note

- 若“触摸屏识别码”与“工程文件识别码”不相同时，HMI 与 PLC 之间将无法通讯。

30.6 EMTP 密码

在编辑完成工程文件 (.emtp) 后，可以设定保护 .emtp 工程文件的密码，请至“系统参数设置” »“用户密码”页面设定此功能 (范围: 1 ~ 4294967295)。启用此功能后，每次欲开启该 .emtp 工程文件时，须先输入正确的密码后才能开启并编辑。



Note

- 在使用 “窗口复制” 功能时，若欲复制的來源工程文件设有 EMTP 密码保护时，须先输入正确的密码后，系统才能执行窗口复制功能。

第三十一章 Memory Map 通讯协议

31.1 概要

Memory Map 通讯协议类似于 IBM 3764R 通讯协议，使用于对应内存数据的变化量较少的场合，且专为两台设备交换数据的通讯协议。Memory Map 通讯协议的特征是两台设备必须一方为 master，另一方为 slave。在一般情况下，master 和 slave 并没有建立通讯，只有当某一方所指定的内存数据变化时，通讯才建立。当双方数据一致后，通讯则马上中断。所以通讯的目的是保持两台设备(master 和 slave)之间相对应的一块相同大小内存数据的一致性。

其中 master 和 slave 中对应的内存需和 MW(MB)内存具有相同的性质和大小。HMI 中的 MW(MB)大小皆为 10,000 字。

MB 和 MW 都是指向相同的寄存器区块，例如，MB0~MBf 对应到 MW0 的各位，MB10~MB1f 对应到 MW1，如下表所示：

设备名称	格式	范围
MB	DDDDh	DDDD:0~4095 h:0~f(hex)
MW	DDDD	DDDD:0~9999

31.2 接脚设定

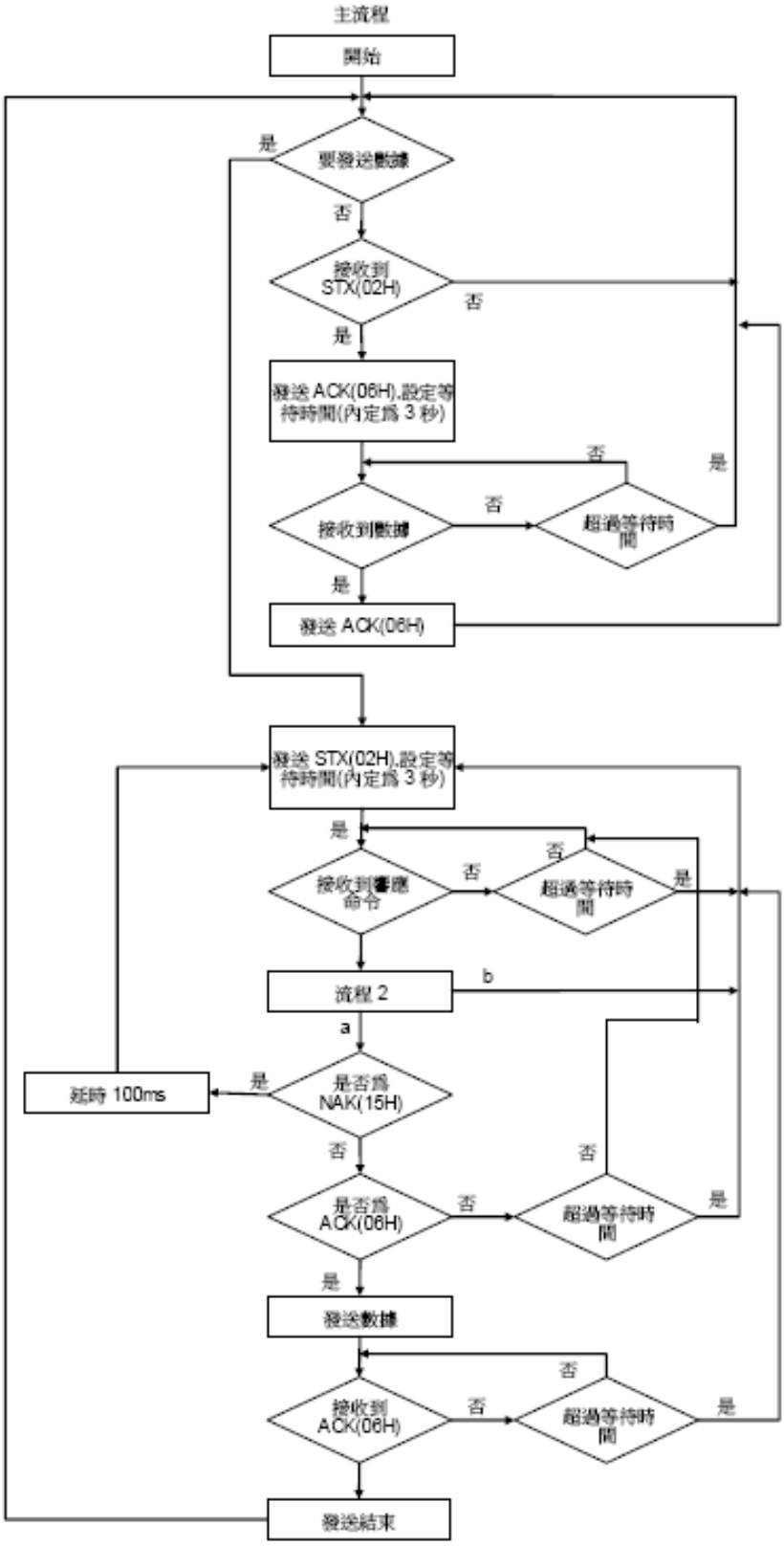
使用 Memory Map 通讯协议时，master 和 slave 必须使用相同的通讯参数。其接线方式如下：

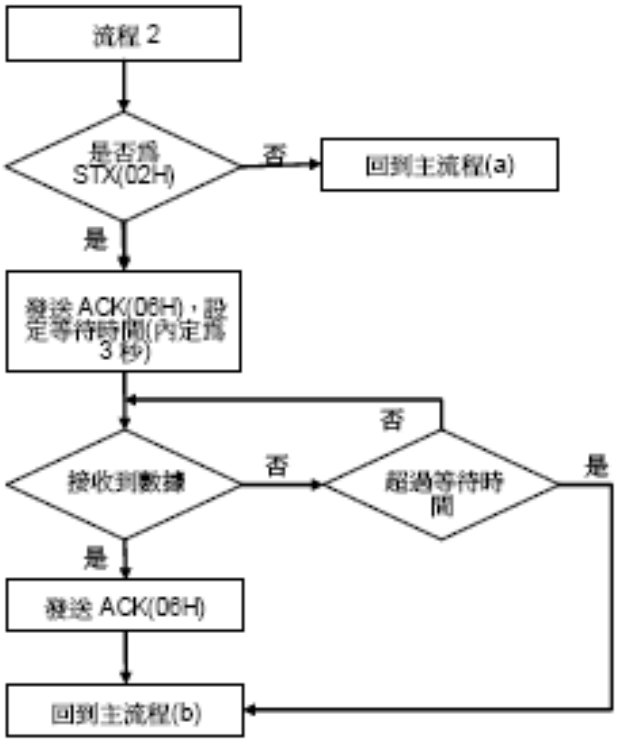
(#表示由具体 PLC 或控制器决定)

界面	RS-232	
设备	Master	Slave
对应脚位	TX(#)	RX(#)
	RX(#)	TX(#)
	GND(#)	GND(#)

界面	RS-485 (4W)	
设备	Master	Slave
对应脚位	TX+(#)	RX+(#)
	TX-(#)	RX-(#)
	RX+(#)	TX+(#)
	RX-(#)	TX-(#)
	GND(#)	GND(#)

31.3 通讯流程图


主流程
開始
是
要發送數據
否
接收到
STX(02H)
否
是
發送 ACK(06H),設定等
待時間(內定爲 3 秒)
否
是
接收到數據
否
超過等待時
間
是
發送 ACK(06H)
發送 STX(02H),設定等
待時間(內定爲 3 秒)
是
否
接收到響應
命令
否
超過等待時
間
是
流程 2
b
a
延時 100ms
是
是否爲
NAK(15H)
否
否
是否爲
ACK(06H)
否
超過等待時
間
是
是
發送數據
否
接收到
ACK(06H)
否
超過等待時
間
是
發送結束



Note

- 流程 2 对 slave 有效，对 master 无效。
- STX 为通讯请求信号，ACK 为响应请求信号，NAK 为忙碌信号。

31.4 通讯数据格式

通讯数据格式可分为两种，MB 指令和 MW 指令。

对 MB 操作的指令格式，请参见下表：

MB 指令

偏移量 (字节)	格式	描述
0	0x02	对 MB 操作的标志
1 2	0x## 0x##	地址(低字节) 位地址(高字节) 如果是 MB-18，则 1*16+2=18，为 0x12，0x00
3	0x00 (or 0x01)	表示所指定 MB 地址的数据内容(因为是 Bit 类型，只能是 0 或 1)

4, 5	0x10, 0x03	结束标志
6	0x##	总和检查码；从偏移量 0 到 5 之字节进行 XOR 运算

对 MW 操作的指令格式，请参见下表：

MW 指令

偏移量 (字节)	格式	描述
0	0x01	对 MW 操作的标志
1 2	0x## 0x##	地址(低字节) 位地址(高字节) 如果地址数据中包含一个 0x10，则在 0x10 后再插入一个 0x10，地址表示多出一个字节，命令格式相应的向后推移一个字节，例如地址为 0x10，0x04，则变为 0x10，0x10，0x04
3	0x##	传送的字节数(由于对字操作，字节数一定为偶数)，如果字节数为 0x10，则在 0x10 后再插入一个 0x10 命令格式相应的向后推移一个字节
4 to 4+n-1	0x##(L),0x##(H) 0x##(L),0x##(H) …	为 1，2 字节所对应地址为起始地址的数据，其中 n 为数据的字节数，如果数据中有 0x10，则在 0x10 后再插入一个 0x10，而”传送字节数”不变，n 则为 n+1，以次类推
4+n, 4+n+1	0x10 0x03	结束标志
4+n+2	0x##	总和检查码；对前面所有字节 xor

31.4.1 通讯范例

范例 1

假设 master 将 MW-3 的内容设为 0x0a，因为数据更动，master 立刻会和 slave 建立通讯，而 slave 接收到数据后把它对应的 MW-3 的内容更新为 0x0a。其对应过程为：

1. master 发送 STX(0x02h)。
2. slave 接收到 master 发送的 STX(0x02h)后，发送返回命令 ACK(0x06h)。
3. master 接收到 slave 的返回命令 ACK(0x06h)。
4. master 发送资料 0x01，0x03，0x00，0x02，0x0a，0x00，0x10，0x03，0x19，如下表所示：

偏移量 (字节)	格式	描述
0	0x01	对 MW 操作的标志
1 2	0x03 0x00	地址(低字节) 位地址(高字节)
3	0x02	传送的字节数(MW3 为两个字节)
4, 5	0x0a, 0x00	MW-3 的内容为 0x0a，0x00
6, 7	0x10, 0x03	结束标志
8	0x19	总和检查码，0x01^0x03^0x00^0x02^0x0a^0x00^0x10^0x03=0x19

5. slave 收到 master 发送的数据后，发送返回命令 ACK(0x06h)。
6. master 接收到 slave 的返回命令 ACK(0x06h)。

通讯完成，master 把更改的 MW 的地址和内容传送给了 slave，slave 再更改 MW 的数据，使得 master 和 slave 对应节点地址内容保持一致。

范例 2

如果地址和数据中包括 0x10 的情况，请注意观察数据格式的变化。

我们假设 slave 把 MW-10 的内容设为 0x10，根据这个协议，slave 立刻会和 master 建立通讯，从而使得 master 接收到数据后把它对应的 MW-10 的内容置为 0x10。过程为：

1. slave 发送 STX(0x02h)。
2. 主方接收到 slave 发送的 STX(0x02h)后，发送返回命令 ACK(0x06h)。
3. slave 接收到 master 的返回命令 ACK(0x06h)。
4. slave 发送资料 0x01，0x10，0x10，0x00，0x02，0x10，0x10，0x00，0x10，0x03，0x10 如下表所示：

偏移量 (字节)	格式	描述
0	0x01	对 MW 操作的标志
1 2 3	0x10 0x10 0x00	地址(低字节) 插入一个 0x10 字节 位地址(高字节)
4	0x02	传送的字节数(MW-10 为两个字节)
5 6 7	0x10 0x10 0x00	MW-10 的低字节内容为 0x10 插入一个 0x10 字节 高字节内容为 0x00

8	0x10	结束标志
9	0x03	
10	0x10	总和检查码，0x01^0x10^0x10^0x00^0x02^0x10^0x10^0x00^0x10^0x03=0x10

5. master 收到 slave 发送的数据后，发送返回命令 ACK(0x06h)。
6. slave 接收到 master 的返回命令 ACK(0x06h)。

slave 把更改的 MW 的地址和内容传送给了 master，master 再更改 MW 的数据，使得 slave 和 master 对应节点地址内容保持一致。

31.5 实作范例

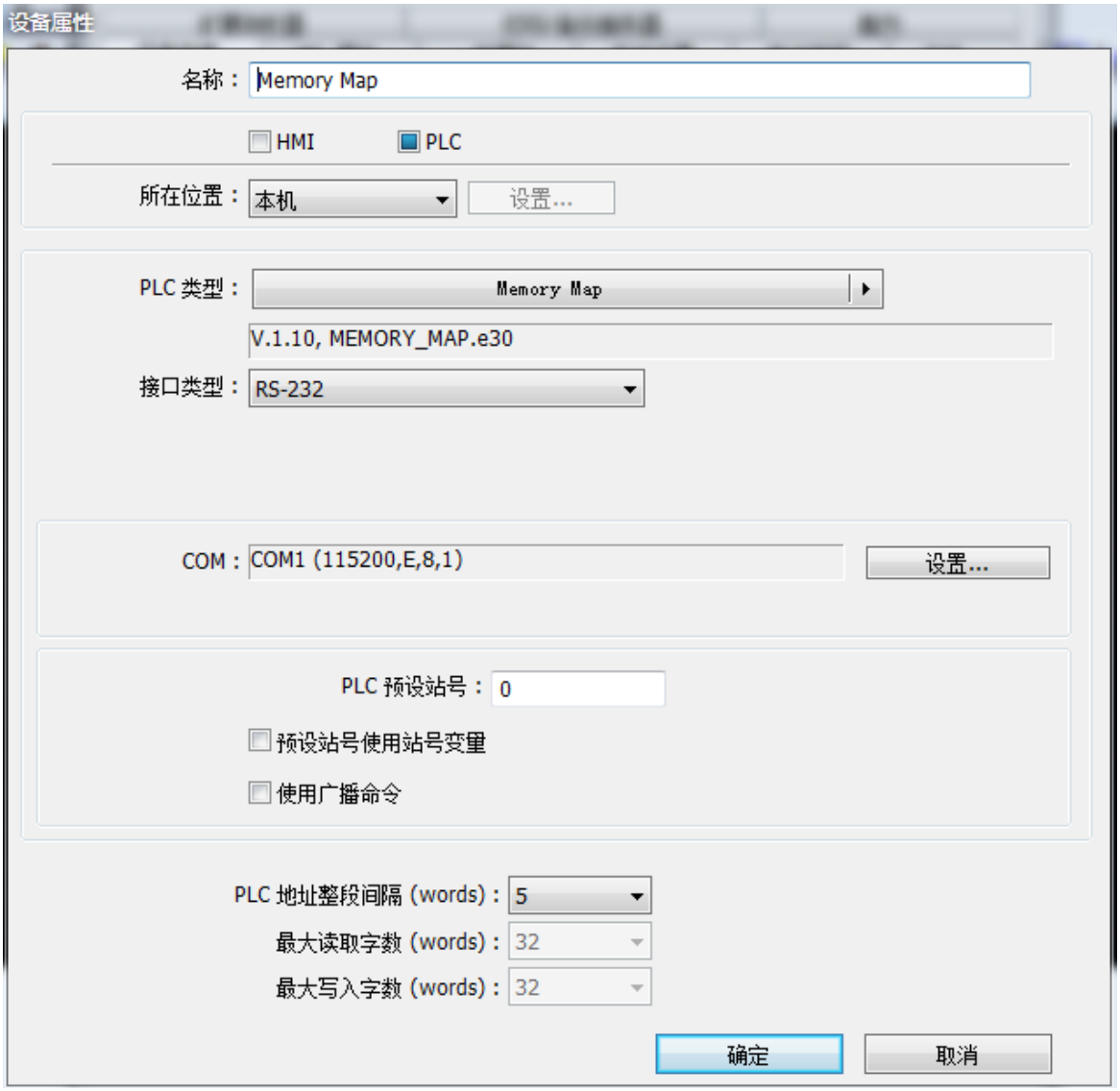
以下将示范两台 HMI 如何使用 Memory Map 通讯协议连接。

Note

- 若是两台的型号不同，请分别建立不同的工程文件，或者是在完成第一台 HMI 的设定后，直接更改“编辑”»“系统参数设置” »“HMI 属性”为第二台 HMI 的型号，重新编译工程文件再下载至第二台 HMI。

31.5.1 新增 Memory Map 的步骤

1. 开启 EasyBuilder Pro，选择“开新文件”，设定 HMI 型号后，照以下步骤：
2. 点选菜单列“编辑”，并点击“系统参数设置”，接着选择“设备清单”页面并点选“新增...”加入新的设备。
3. “名称”填入“Memory Map”，并点选“PLC”，设定“所在位置”为“本机”。
4. “PLC 类型”选择“Memory Map”，并在“PLC 接口”选择“RS-232”。


设备属性
名称：
Memory Map
HMI
PLC
所在位置：
本机
设置...
PLC 类型：
Memory Map
V.1.10, MEMORY_MAP.e30
接口类型：
RS-232
COM：
COM1 (115200,E,8,1)
设置...
PLC 预设站号：
0
预设站号使用站号变量
使用广播命令
PLC 地址整段间隔 (words)：
5
最大读取字数 (words)：
32
最大写入字数 (words)：
32
确定
取消

5. 点击“设置”，设定如下：

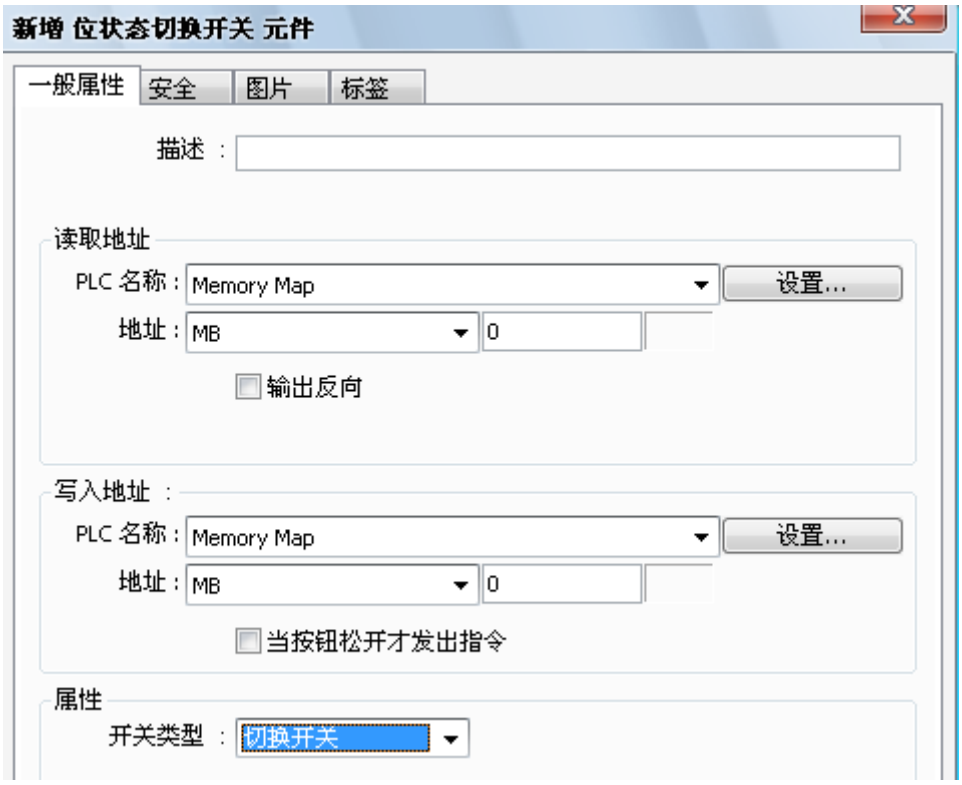
6. 完成通讯端口设定后按下“确定”。

Note

- MT500 系列有分为 MemoryMap_Master 和 MemoryMap_Slave，详情请参考相关手册。
- eMT 3000 系列和 MT8000 系列选择 Memory Map 即可。
- “数据位”必须为 8 Bits。
- 两台 HMI 的所有其它设置必须一致。

31.5.2 元件设定

接着在窗口 10 上增加 2 个元件，「位状态切换开关」和「多状态切换开关」：
新增位状态切换开关，如下图所示：



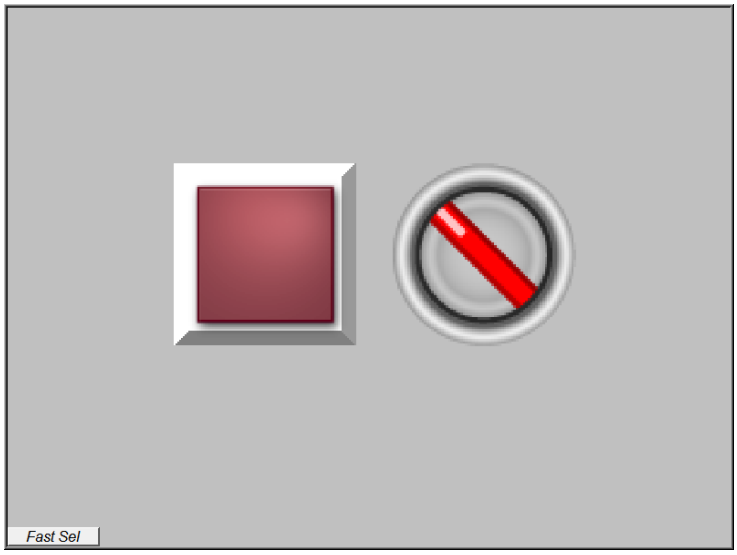
1. 读取和写入地址的“PLC 名称”选择“Memory Map”。
2. 地址选择 MB-0。
3. “开关类型”选择“切换开关”。(可选择适合的图片或标签以供辨识)

新增多状态切换开关，如下图所示：

1. 读取和写入地址的“PLC 名称”选择“Memory Map”。
2. 地址选择 MW-1。
3. “循环”选择“启用”。(可选择适合的图片或标签以供辨识)

31.5.3 执行结果

将工程文件编译并下载至 HMI，再将工程文件下载至第二台 HMI。
完成画面可参考下图：



按一下任意一个按钮，对应另一台触控屏幕的该按钮也将跟着动作，它们的状态将始终保持一致。

一台 HMI 和任一台控制器之间的通讯其方式与上述类似，其根本原理是 2 台设备的相同寄存器的数据要保持一致。

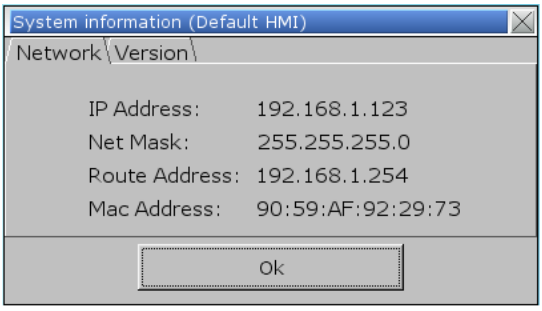
第三十二章 FTP 服务器的运用

32.1 概要

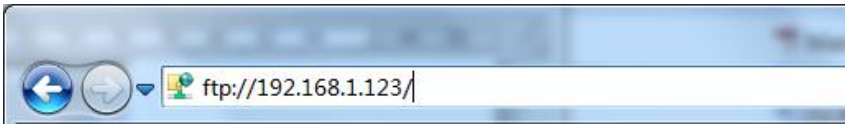
除了使用 SD 卡、U 盘或 EasyPrinter 服务器将历史数据由触摸屏备份至电脑 之外，也能利用 FTP 服务器达成这个目标。当工程文件下载到触摸屏后，可通过 FTP 服务器进行历史数据备份、配方数据备份或更新配方数据的动作，但是无法删除存在 FTP 服务器内的文件。

32.2 登入 FTP 服务器的步骤

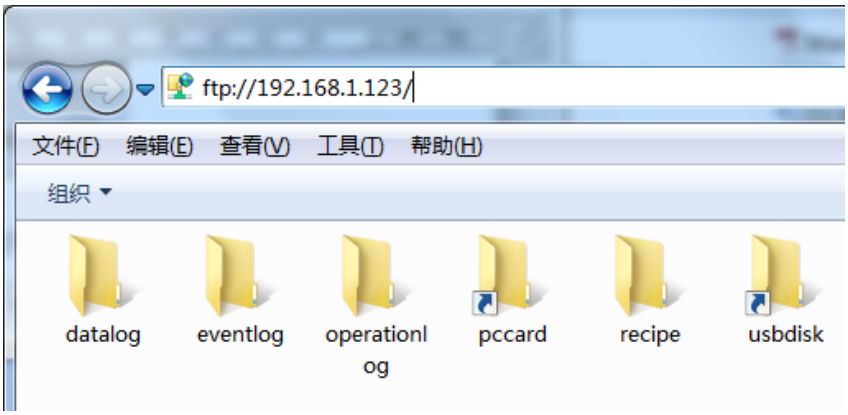
1. 在登入 FTP 服务器之前请先确认触摸屏的 IP 地址。



2. 在“我的计算机”网址列输入触摸屏的 IP 地址 ftp://192.168.1.123（范例）然后登入账号 uploadhis，密码为触摸屏的 history upload password（若没有更改过密码，预设密码为 111111)。或是直接输入 ftp://uploadhis:111111@192.168.1.123/



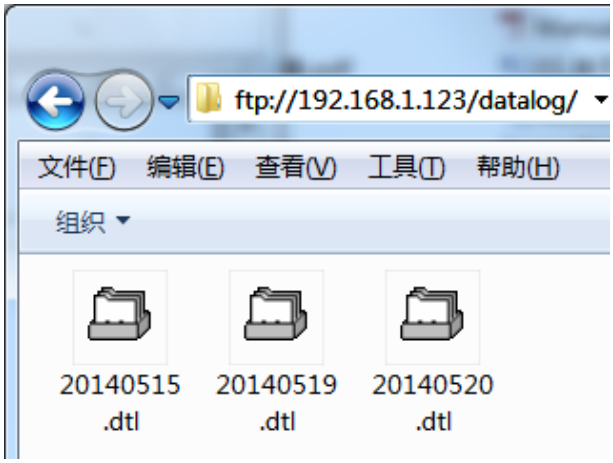
3. 输入 IP 地址后，网址列会显示为 ftp://192.168.1.123/，并且可以看到下图所示的文件夹。



32.3 备份历史数据及更新配方数据

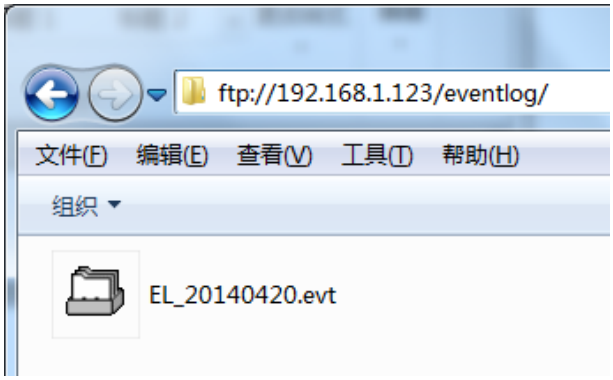
- 备份资料取样记录

1. 点选 datalog 文件夹后，可看到 datalog 在 EasyBuilder Pro 设定的文件夹文件名。
2. 点选档名后即可看到 datalog 的文件。
3. 使用复制及贴上的功能，将资料取样的记录保存在电脑 上。



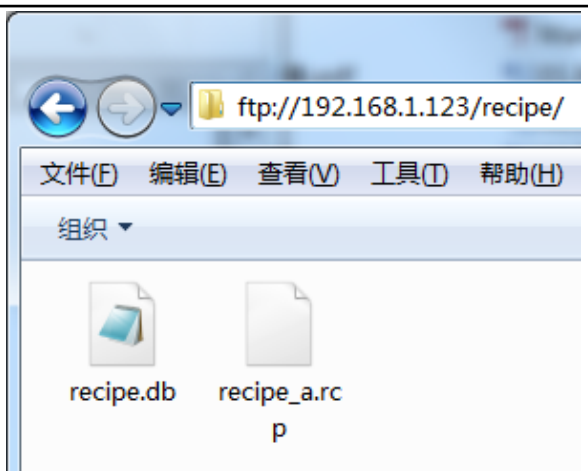
- 备份警报及事件记录

1. 点选 eventlog 文件夹后，即可看到事件的记录文件。
2. 可使用复制及贴上的功能将事件记录保存在电脑 上。



- 备份及更新配方数据

1. 点选 recipe 文件夹后，即可看到配方数据的文件。
2. 可使用复制及贴上的功能将配方数据保存在电脑 上。



Note

- 因为配方数据每分钟会自动储存一次，若要更新 recipe.rcp 或 recipe_a.rcp，请务必于 1 分钟内将触摸屏重启，否则新配方资料将被原有旧配方覆盖过去。

用户也可使用系统寄存器 LB-9047 (重新启动 HMI) 及 LB-9048 (重启机制保护) 将触摸屏重启。若是使用 LB-9047 及 LB-9048，需先将 LB-9048 设 ON 后，再将 LB-9047 设 ON，即可重启 HMI。

第三十三章 EasyDiagnoser

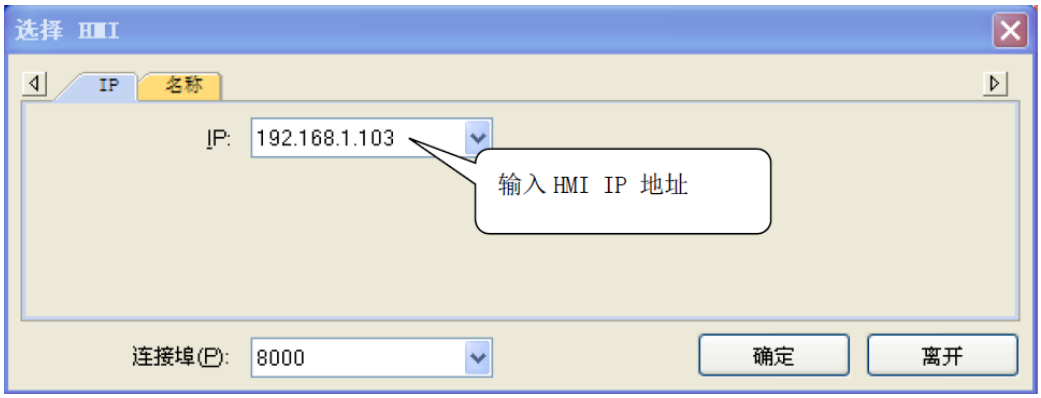
33.1 概要

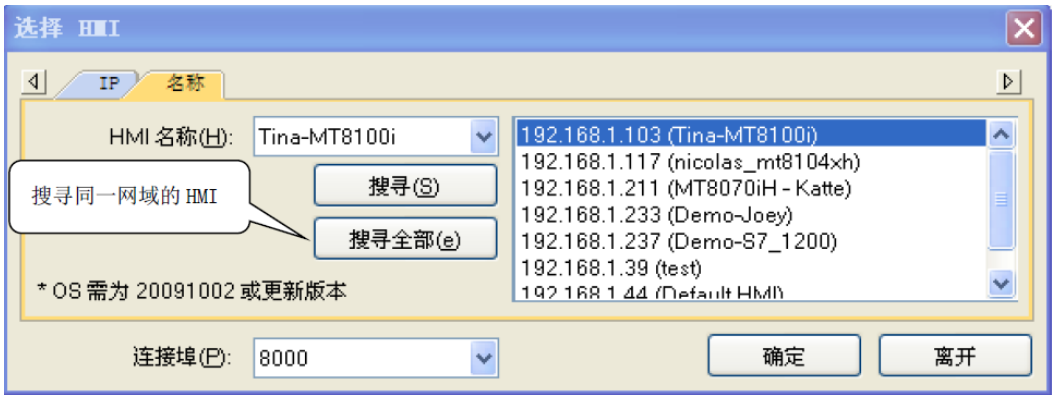
EasyDiagnoser 是用来侦测触摸屏与 PLC 之间通讯是否正常的工具。

33.2 设定步骤

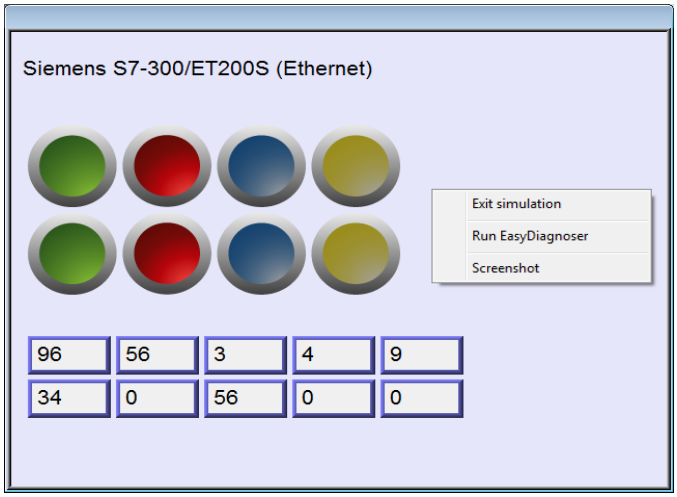
以下为设定 EasyDiagnoser 的步骤：

1. 开启 Utility Manager 并且点选 EasyDiagnoser。
2. 设定欲进行通讯之触摸屏的 IP 地址，可选择自行输入 IP 地址或使用 “Search all” 功能，并输入 “Project Port”。

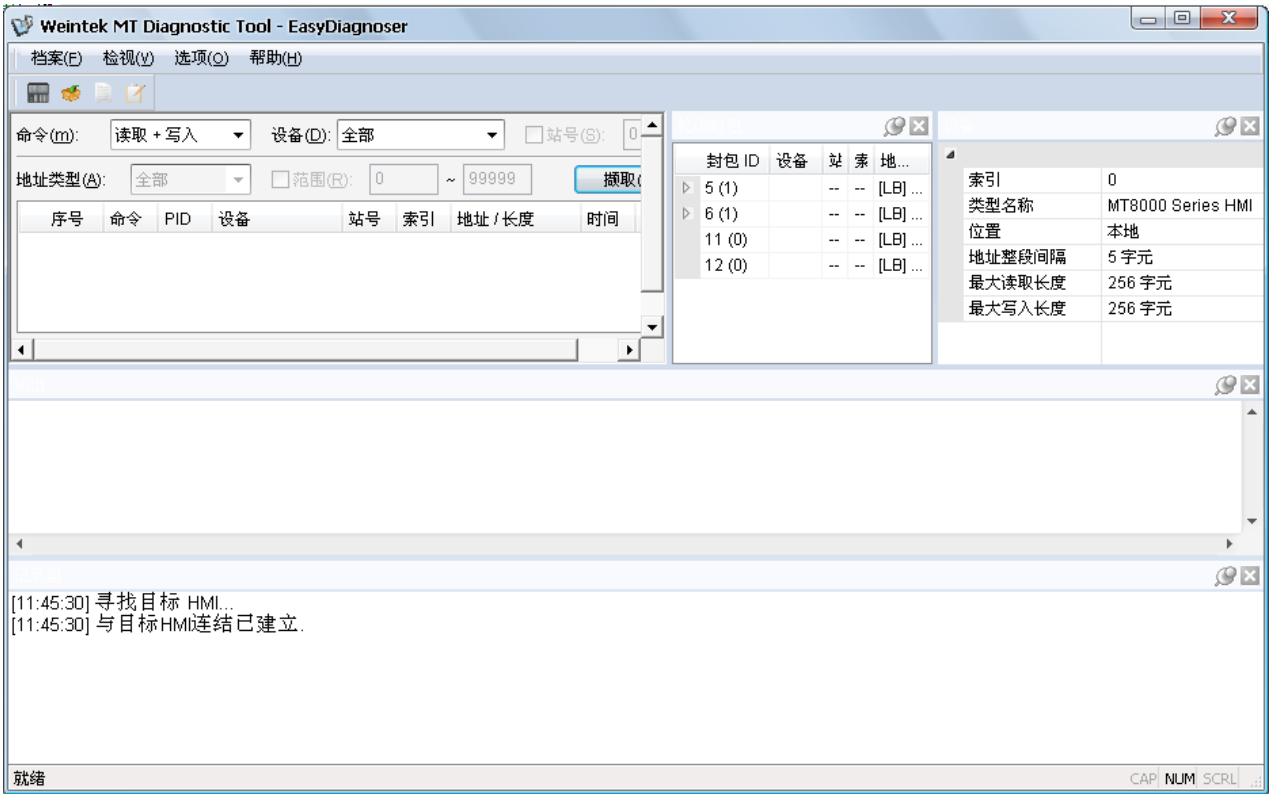




另外在 EasyBuilder Pro 执行 On Line Simulation 时，按下右键可选择 “Run EasyDiagnoser” 进入 EasyDiagnoser。



3. 完成以上设定之后，按下 “OK”，EasyDiagnoser 操作画面如下图所示：



33.3 EasyDiagnoser 设定

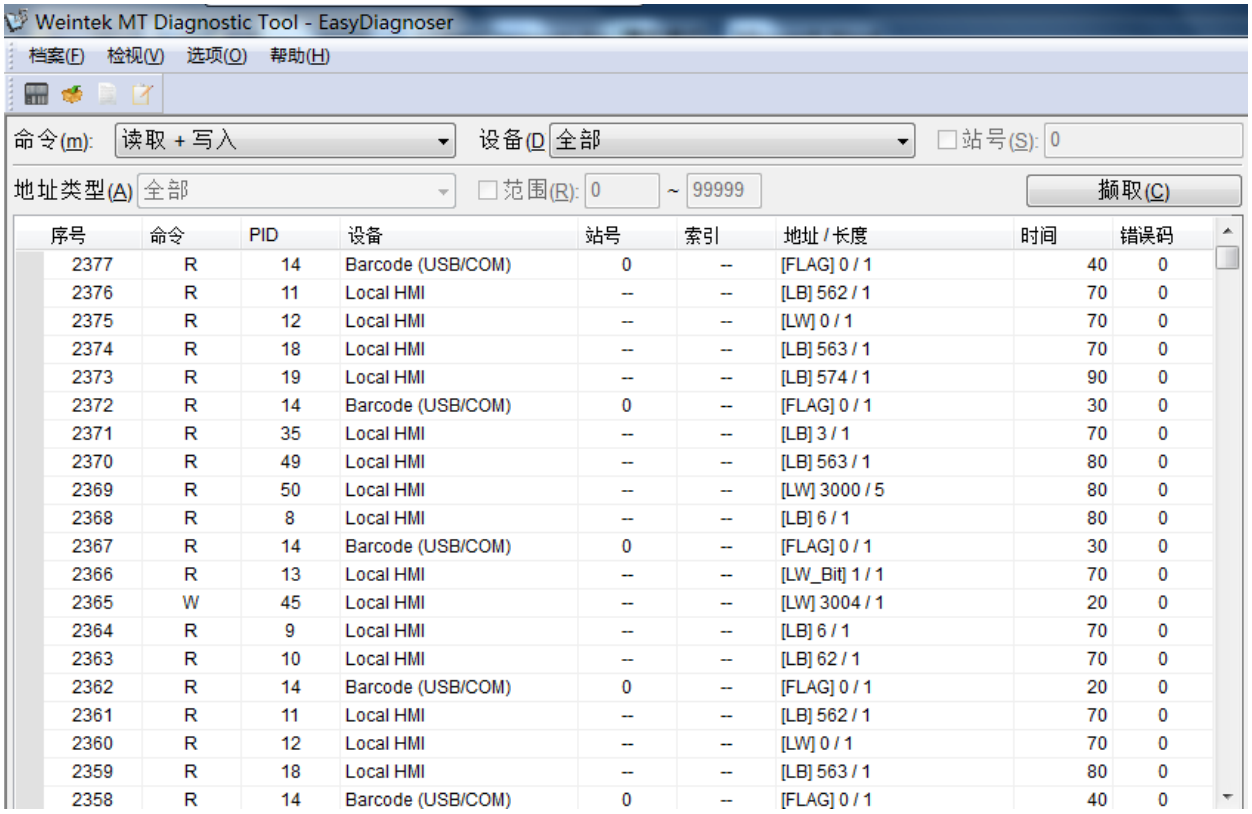
33.3.1 主要选单

项目	描述
文件	**另存新档** 可将撷取下来的通讯数据，储存成 .xls 文件，并可于 Excel

	开启。 **离开** 离开目前文件。
检视	**设备列表**可显示设备列表窗口。 **封包列表**可显示封包列表窗口。 **讯息窗口**可显示讯息窗口窗口。 **输出窗口**可显示输出窗口窗口。
选项	**工具栏** 显示“设备列表”、”封包列表”、”讯息窗口”、”输出窗口”之工具栏。 **状态列** 在 EasyDiagnoser 窗口最底下，显示 CAP ， NUM 或 SCRL 的信息。 **更新封包列表** 显示目前触摸屏页面的封包。 **显示组件 ID** 显示触摸屏上组件的 ID。  **清除通讯记录** 清除所有通讯中所记录的信息。
帮助	显示 EasyDiagnoser 版本信息。

33.3.2 通讯记录区

用户可以观察触摸屏和 PLC 之间的通讯。



序号	命令	PID	设备	站号	索引	地址 / 长度	时间	错误码
2377	R	14	Barcode (USB/COM)	0	--	[FLAG] 0 / 1	40	0
2376	R	11	Local HMI	--	--	[LB] 562 / 1	70	0
2375	R	12	Local HMI	--	--	[LW] 0 / 1	70	0
2374	R	18	Local HMI	--	--	[LB] 563 / 1	70	0
2373	R	19	Local HMI	--	--	[LB] 574 / 1	90	0
2372	R	14	Barcode (USB/COM)	0	--	[FLAG] 0 / 1	30	0
2371	R	35	Local HMI	--	--	[LB] 3 / 1	70	0
2370	R	49	Local HMI	--	--	[LB] 563 / 1	80	0
2369	R	50	Local HMI	--	--	[LW] 3000 / 5	80	0
2368	R	8	Local HMI	--	--	[LB] 6 / 1	80	0
2367	R	14	Barcode (USB/COM)	0	--	[FLAG] 0 / 1	30	0
2366	R	13	Local HMI	--	--	[LW_Bit] 1 / 1	70	0
2365	W	45	Local HMI	--	--	[LW] 3004 / 1	20	0
2364	R	9	Local HMI	--	--	[LB] 6 / 1	70	0
2363	R	10	Local HMI	--	--	[LB] 62 / 1	70	0
2362	R	14	Barcode (USB/COM)	0	--	[FLAG] 0 / 1	20	0
2361	R	11	Local HMI	--	--	[LB] 562 / 1	70	0
2360	R	12	Local HMI	--	--	[LW] 0 / 1	70	0
2359	R	18	Local HMI	--	--	[LB] 563 / 1	80	0
2358	R	14	Barcode (USB/COM)	0	--	[FLAG] 0 / 1	40	0

项目	描述
命令	**读取 + 写入** 显示读和写的命令在通讯记录区。 **读取** 只显示读的命令在通讯记录区。 **写入** 只显示写的命令在通讯记录区。
设备	**全部** 显示本地触摸屏和 PLC 的信息。 ● 如果设定“命令 : 读取 + 写入”，在通讯记录区会显示本地触摸屏和 PLC 的读和写的信息。 ● 如果“命令 : 读取”，在通讯记录区会显示本地触摸屏和 PLC 的读的信息。 ● 如果“命令 : 写入”，在通讯记录区会显示本地触摸屏和 PLC 的写的信息。 **Local HMI** 显示本地触摸屏的信息。

- 如果设“命令 : 读取 + 写入”，在通讯记录区会显示本地触摸屏的读和写的信息。
- 如果“命令 : 读取”，在通讯记录区会显示本地触摸屏的读的信息。
- 如果“命令 : 写入”，在通讯记录区会显示本地 HMI 的写的信息 PLC。

PLC

显示 PLC 的信息。

- 如果设“命令 : 读取 + 写入”，在通讯记录区通讯记录区会显示 PLC 的读和写的信息。
- 如果“命令 : 读取”，在通讯记录区通讯记录区会显示 PLC 的读的信息。
- 如果“命令 : 写入”，在通讯记录区通讯记录区会显示 PLC 的写的信息。

站号	选择想显示的 PLC 之站号 (当“设备”选择 All 时无法使用使功能)。
地址类型	用户可以选择全部或是其中的设备地址类型显示在屏幕上 (当“设备”选择 All 时无法使用使功能)。
范围	设定要撷取的地址范围 (当“设备”选择 All 时无法使用使功能)。
撷取	点选”撷取”钮开始或停止撷取通讯信息。
错误码	请参考本章《33.4 错误代码》。

33.3.3 轮询封包

项目	描述
封包 ID	封包的 ID 编号，可由通讯记录区看出那一个封包 ID 的组件有问题。
设备	显示触摸屏和 PLC 型号。
站号	显示 PLC 站号。
索引	显示组件所使用的索引寄存器编号。
地址/长度	显示设备类型地址及封包内的 word 长度。

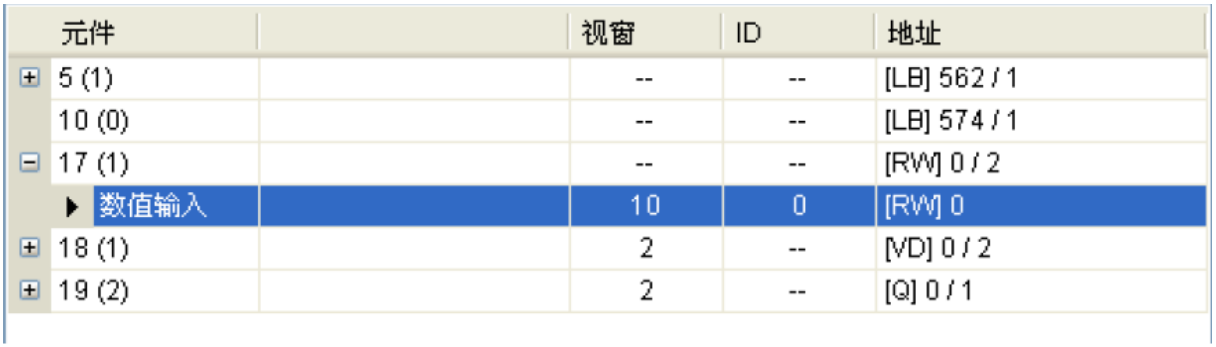
项目	描述
物件	封包 ID 内的物件。
窗口	组件在程序中的所在窗口。
ID	组件的 ID 号码。
地址	组件地址。

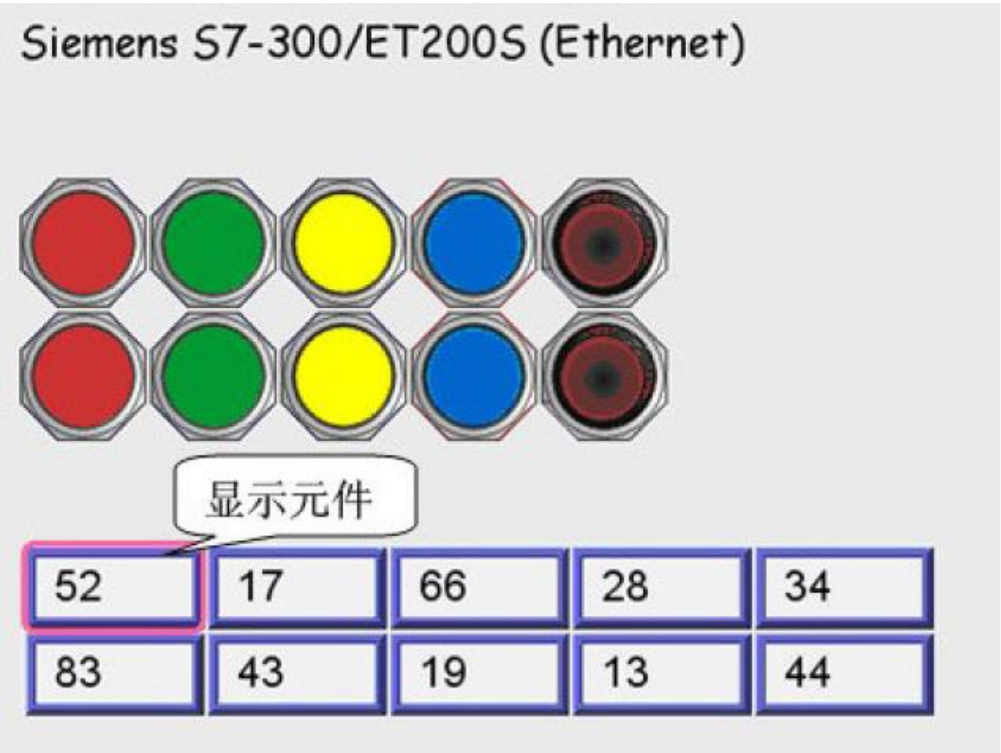
Note

■ 点选封包 ID 后，第 3 栏会显示设备的站号：

■ 双击封包 ID 之后点选组件，可显示组件所在的位置。

例如， 选择数值输入 同时“窗口”显示 10，表示此组件在程序中的第 10 窗口中，同时此组件在触摸屏上会被粉红色的框框标示出来，如下图所示：





33.3.4 设备

设备窗口显示触摸屏及 PLC 的相关讯息。

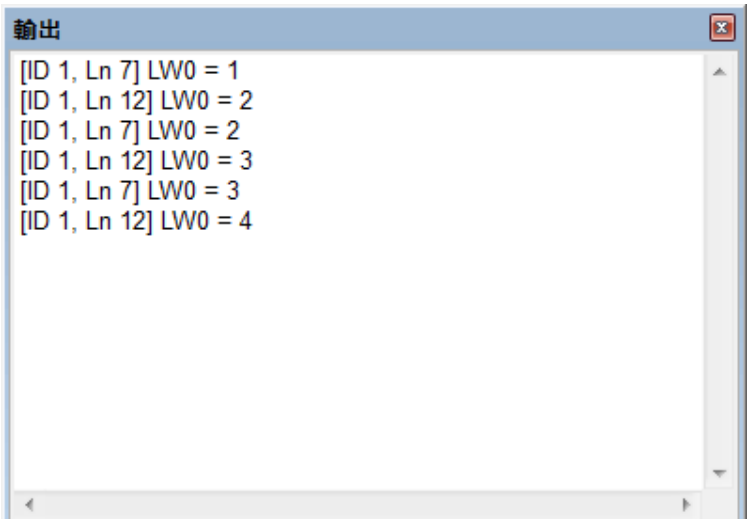
33.3.5 输出 (Macro debug)

搭配使用 Macro 所提供的 Trace 函数，即可侦测 Macro 执行的状态。

以下图为例，”ID 1,Ln 7” 及 “ID 1,Ln 12”：

ID 1 表示 Macro 的名称。

Ln 7 及 Ln 12 表示显示在 Macro 中的第 7 行及第 12 行资料。



详细信息请参考《18 宏指令》。

33.4 错误代码

在通讯记录区可从错误代码找出错误原因。请参考下列错误代码。

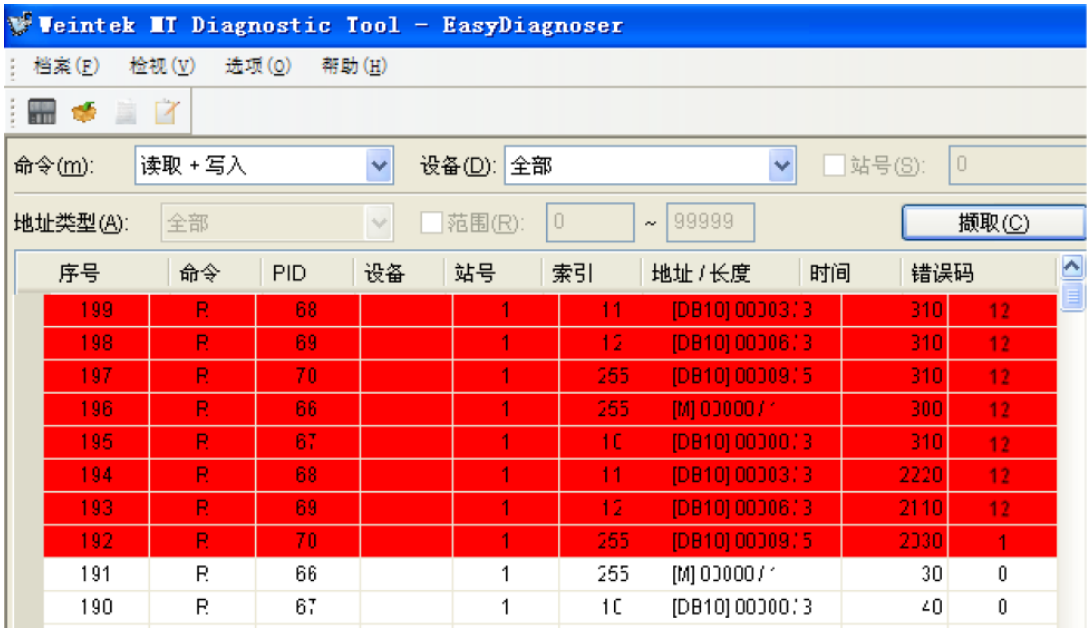
0 : Normal

1 : Time out

2 : Fail Error

12 : Ignore

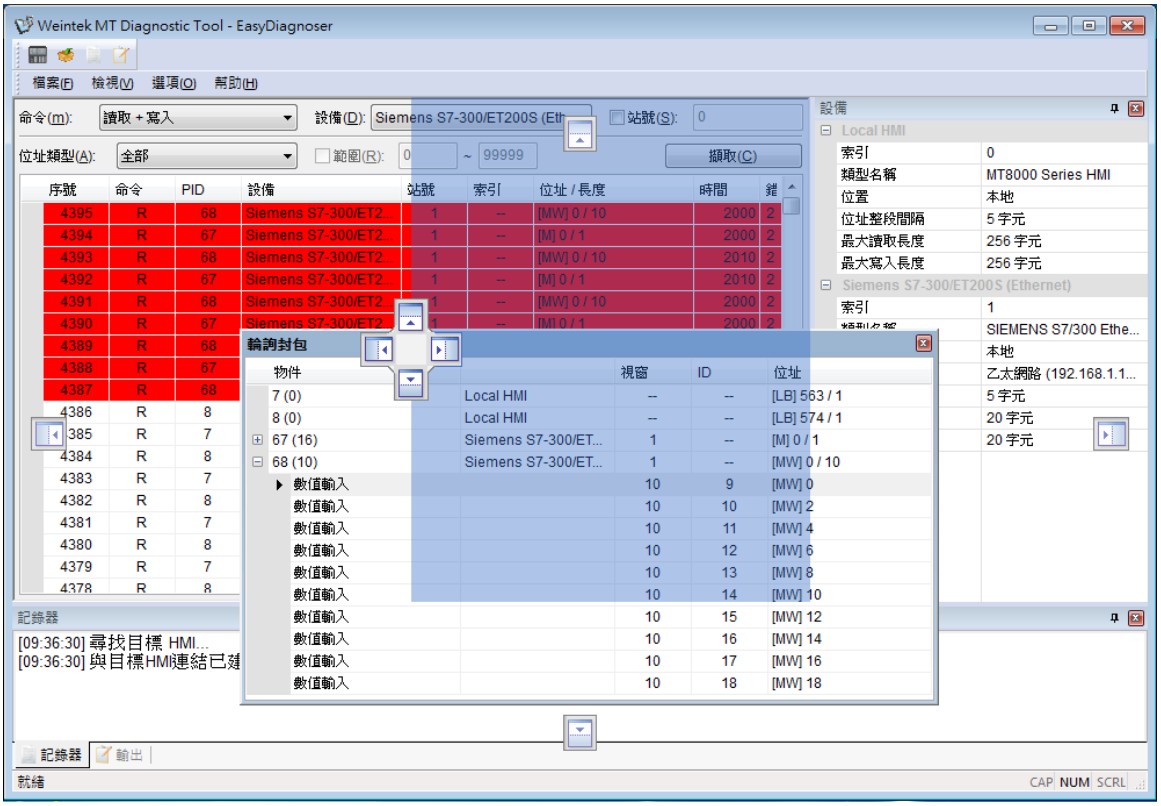
当错误发生时，错误的讯息会标示成红色，如下图：



错误代码是 1 是由于 PLC 与触摸屏断了通讯，如果错误代码是 12，表示出现 **“PLC No Response”** 讯息窗口。

33.5 窗口调整

用户可以使用拖曳功能及显示在编辑画面上的多个定点图标来放置窗口到您喜爱的位置。



第三十四章 Rockwell EtherNet/IP Free Tag Names

34.1 概要

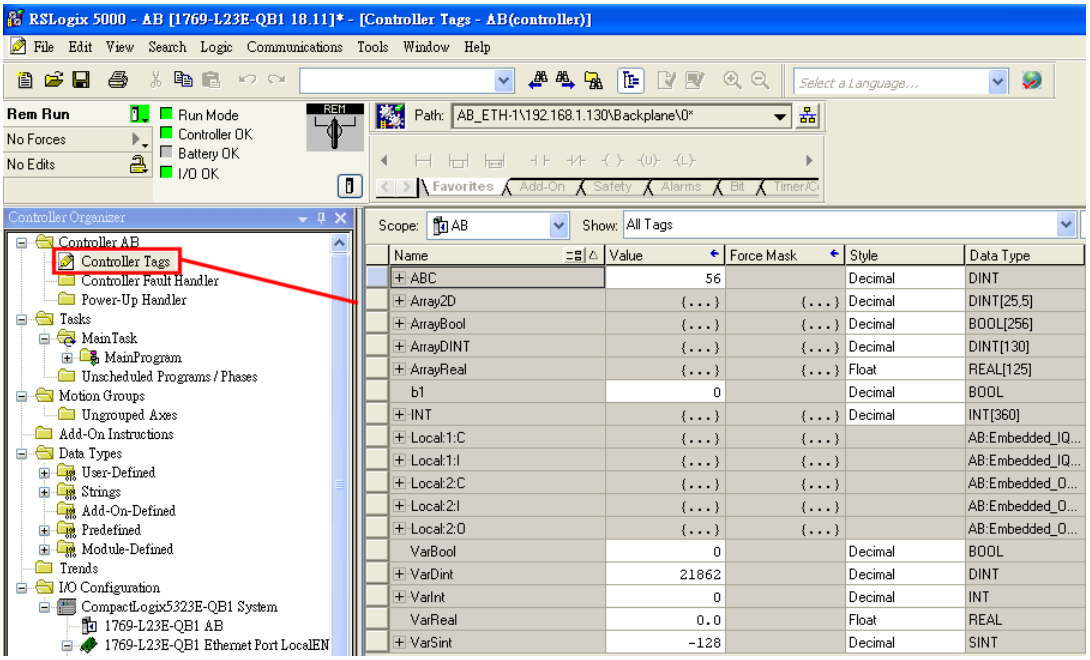
使用 Rockwell EtherNet/IP-Tag (CompactLogix/ControlLogix) 的驱动时可以将用户于 RSLogix5000 编辑的 tag 导出成 .csv 文件，再开启 EasyBuilder Pro 设定驱动后直接导入 .csv 文件，以取得 tag 的信息，但是 User-Defined, Predefined 和 Module-Defined 的结构并不会被导出。

此时可利用 EasyBuilder Pro 的 Structure Editor 工具来输入和编辑 User-Defined, Predefined 和 Module-Defined 中的数据。

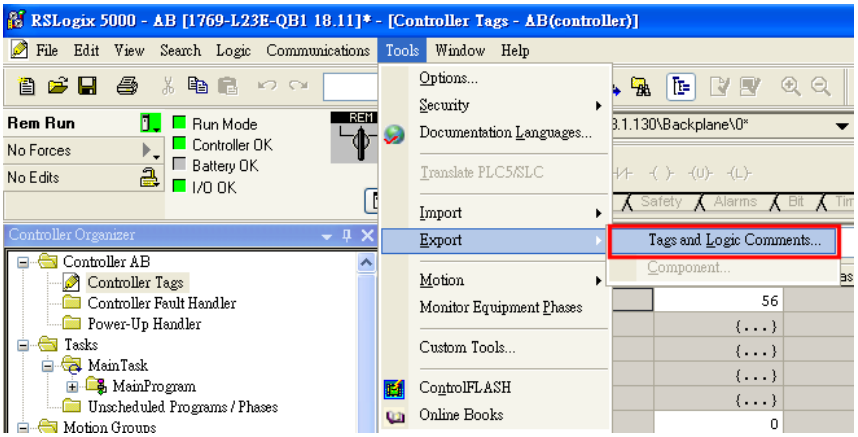
	A	B	C	D	E	F	
7	TYPE	SCOPE	NAME	DESCRIPT	DATATYPE	SPECIFIER	ATTRIBUTES
8	TAG		Local:1:C		AB:Embedded_IQ16F:C:0		
9	TAG		Local:1:I		AB:Embedded_IQ16F:I:0		
10	TAG		Local:2:C		AB:Embedded_OB16:C:0		
11	TAG		Local:2:I		AB:Embedded_OB16:I:0		
12	TAG		Local:2:O		AB:Embedded_OB16:O:0		
13	TAG		Array2D		DINT[25,5]		(RADIX := Decimal, Cons
14	TAG		ArrayBool		BOOL[256]		(RADIX := Decimal, Cons
15	TAG		ArrayDINT		DINT[130]		(RADIX := Decimal, Cons
16	TAG		ArrayReal		REAL[125]		(RADIX := Float, Constant
17	TAG		B001		INT[15]		(RADIX := Decimal, PLC
18	TAG		b003		INT[255]		(RADIX := Decimal, PLC
19	TAG		b1		BOOL		(RADIX := Decimal, Cons

34.2 导入用户自订 AB Tag CSV 档至 EasyBuilder Pro

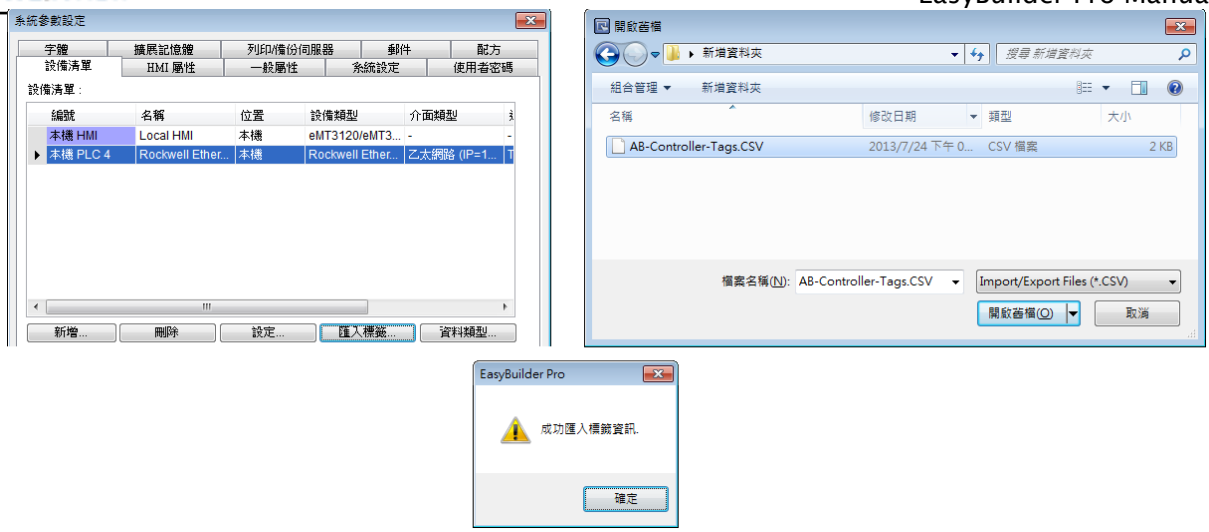
1. 在 RSLogix5000 建立 Tags。



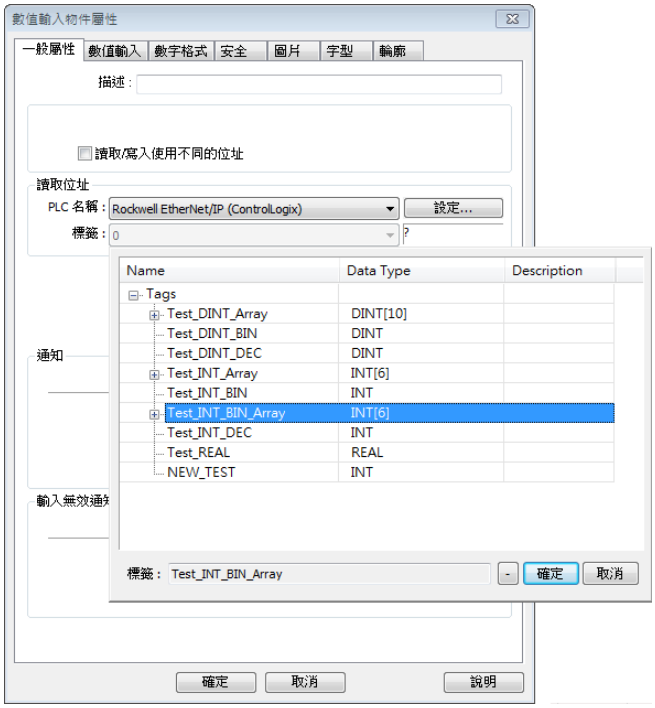
2. 导出 Tags 成 .csv 檔。



3. 在 EasyBuilder Pro 建立 Rockwell EtherNet/IP-Tag (CompactLogix/ControlLogix) 驱动。输入 PLC 的 IP 地址并点选“导入标签”。



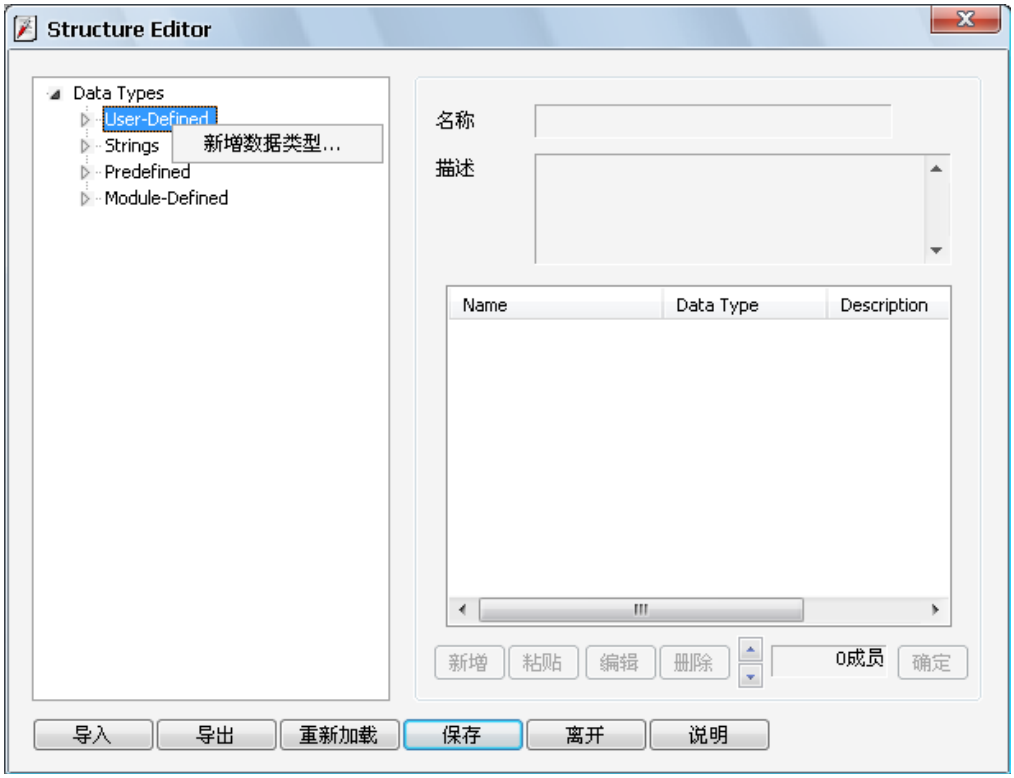
4. 在元件窗口中选择 PLC，并选择 controller tag 即可。



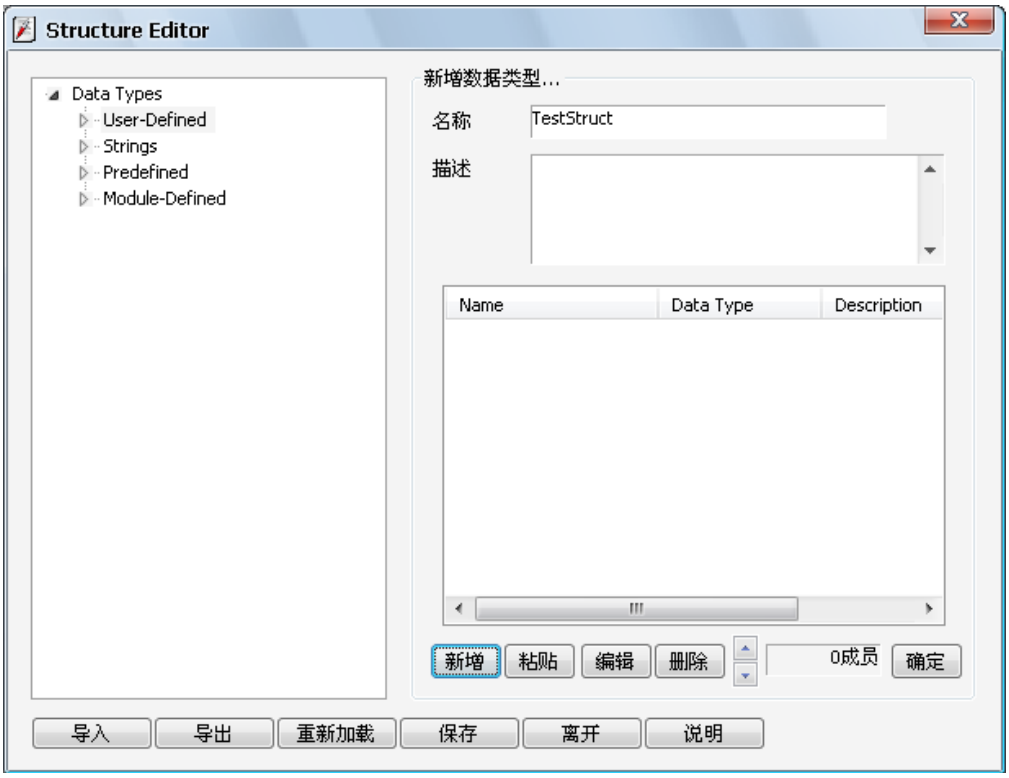
34.3 新增数据类型的步骤

Structure Editor 工具放置于 EasyBuider Pro 安装后的文件夹内。点击 Structure Editor.exe 后会出现如下图的编辑窗口。

1. 在所属的类型上点击鼠标右键（通常为 User-Defined），点击选单的"新增数据类型"即可开始编辑。

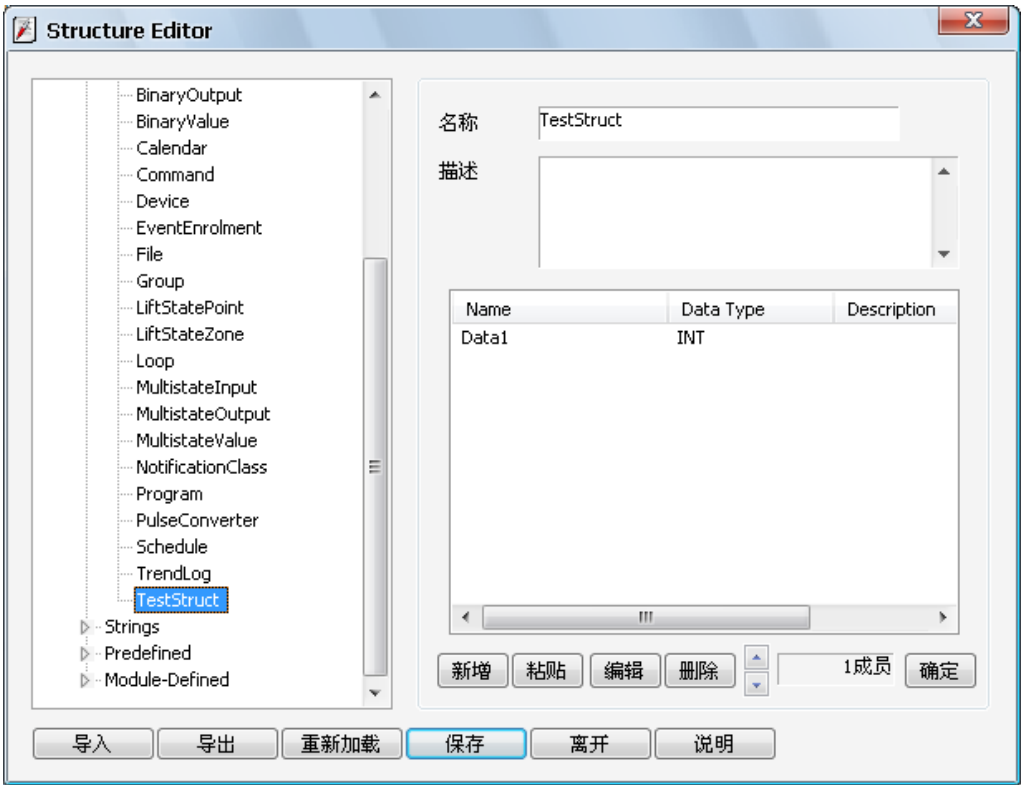


2. 输入类型名称后，"描述"可略过。新增资料成员，点击"新增"按钮。



3. 输入数据的名称与类型后按下“确定”离开。

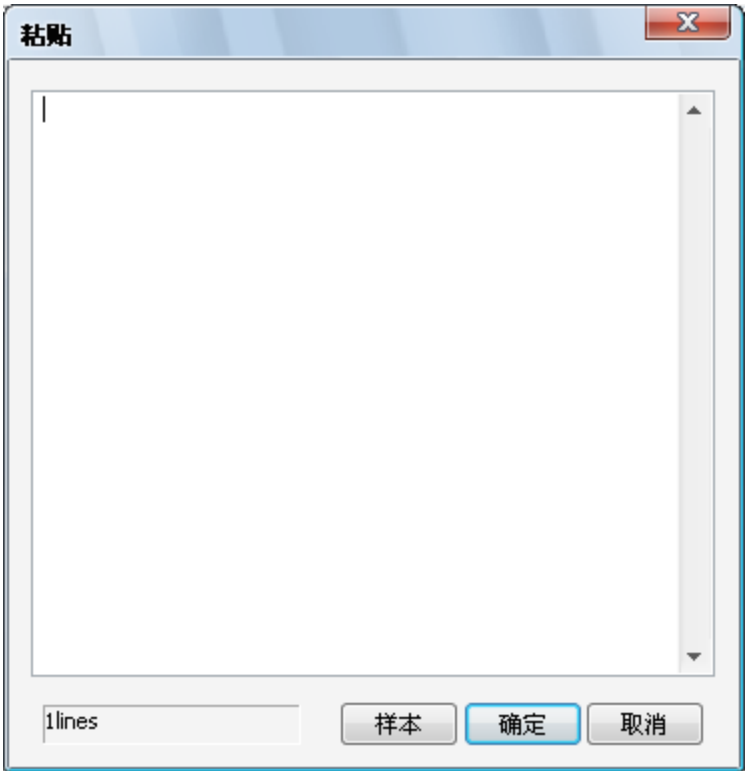
4. 增加完所有的数据成员后按下”确定”键，此时左边的类别表即会出现刚建立的数据类型。



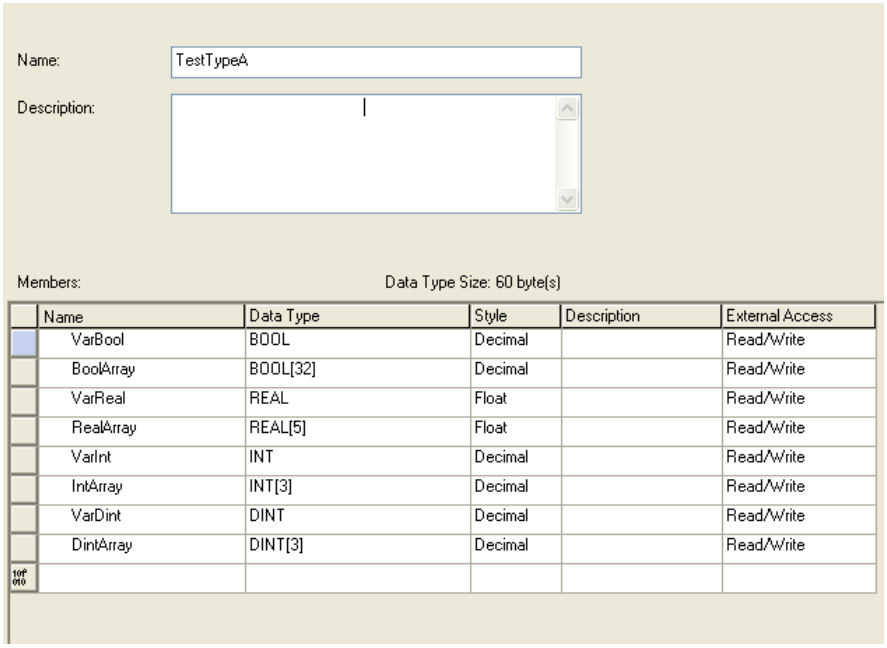
5. 修改数据类型名称与描述后需按下“确定”才进行更动。

34.4 执行粘贴功能的步骤

1. 在新增数据成员时，使用此功能可一次新增多笔数据，方式为在主画面按下“粘贴”按钮。

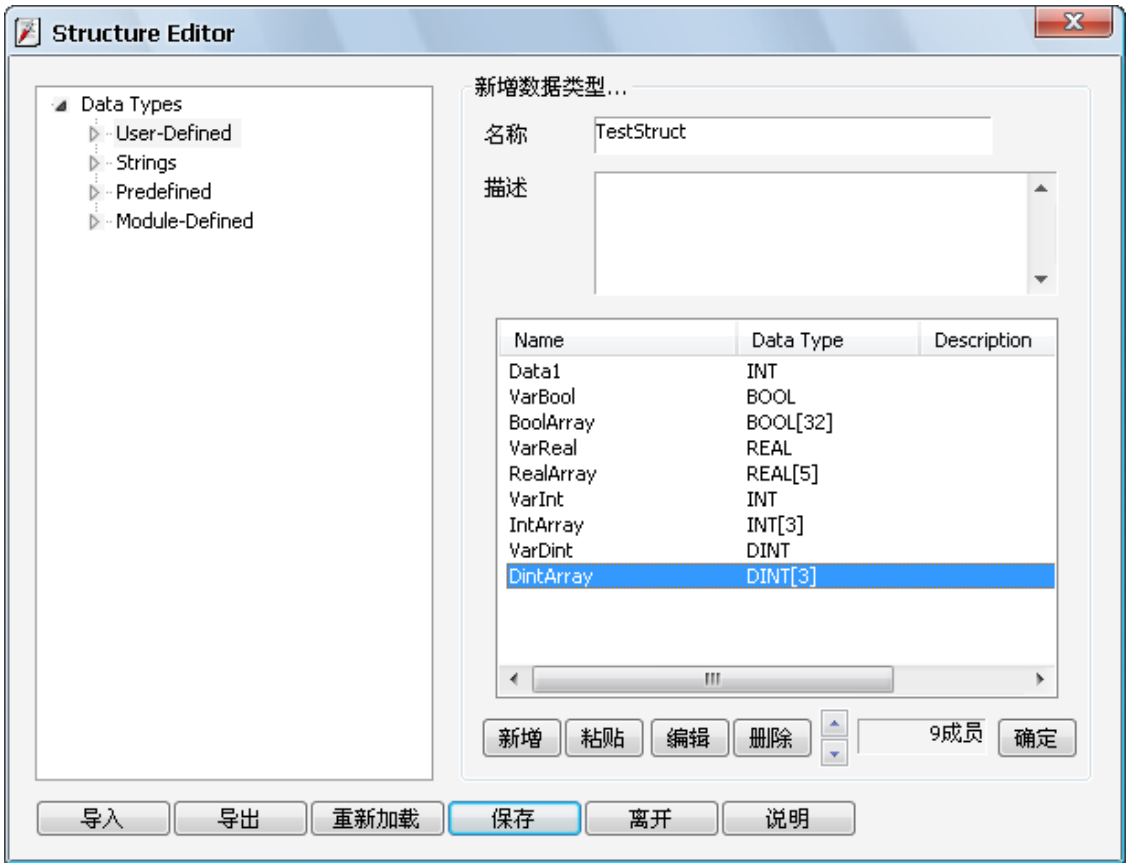


2. 编辑方式每行先输入数据名称后接一空格或 tab，然后输入数据类型，可按“样本”按钮参考；建议从 RSLogix 5000 中直接复制黏贴以避免输入错误。



3. 上图为一在 RSLogix 中自定义的类型，使用鼠标将 Name 与 Data Type 选取并进行复制，然后粘贴到编辑画面上，如下图所示。

4. 此时按下“确定”完成操作回到主画面，即可看到已成功新增多笔资料。



34.5 其它功能

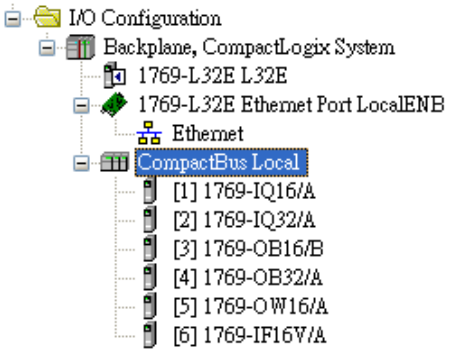
- 修改数据
  直接双击主画面中需要修改的数据，或点击该数据后按下“编辑”按钮。
- 删除数据
  选取要删除的数据按下“删除”按钮；若要删除所有数据则压住键盘上的 Delete 键并点击主画面上的“删除”按钮。
- 删除数据类型
  在主画面左边的类型列表选取要删除的类型，按下键盘上的 Delete 按键，即可删除该类型。
- 导入预设
  可以导入预设文件开始重新编辑。
- 导出预设
  可将编辑好的数据导出，以便在其它文件中使用。

34.6 模块预设结构

模块预设结构 Module-Defined

这里示范如何建立一个模块的预设结构。

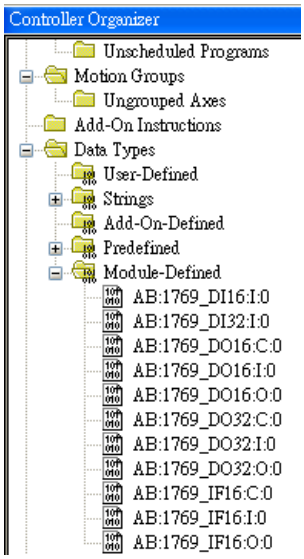
在 RSLogix5000 的 I/O Configuration 里设定了 I/O 的模块。



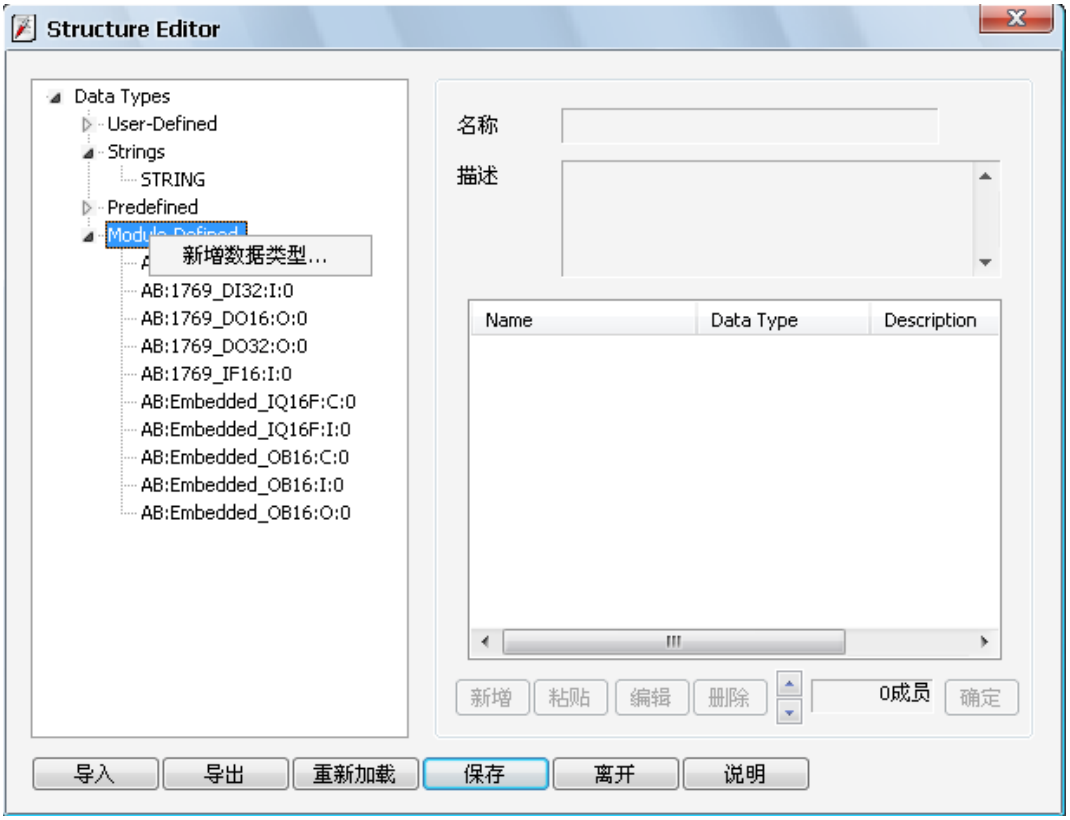
这些模块的 Tag 在输出到 .csv 文件时并不会把结构列出来，所以我们必须帮它建立。

	A	B	C	D	E	F	G	H
7	TYPE	SCOPE	NAME	DESCRIPT	DATATYPE	SPECIFIER	ATTRIBUTES	
8	TAG		Local:1:I		AB:1769_DI16:I:0			
9	TAG		Local:2:I		AB:1769_DI32:I:0			
10	TAG		Local:3:C		AB:1769_DO16:C:0			
11	TAG		Local:3:I		AB:1769_DO16:I:0			
12	TAG		Local:3:O		AB:1769_DO16:O:0			
13	TAG		Local:4:C		AB:1769_DO32:C:0			
14	TAG		Local:4:I		AB:1769_DO32:I:0			
15	TAG		Local:4:O		AB:1769_DO32:O:0			
16	TAG		Local:5:C		AB:1769_DO16:C:0			
17	TAG		Local:5:I		AB:1769_DO16:I:0			
18	TAG		Local:5:O		AB:1769_DO16:O:0			
19	TAG		Local:6:C		AB:1769_IF16:C:0			
20	TAG		Local:6:I		AB:1769_IF16:I:0			
21	TAG		Local:6:O		AB:1769_IF16:O:0			

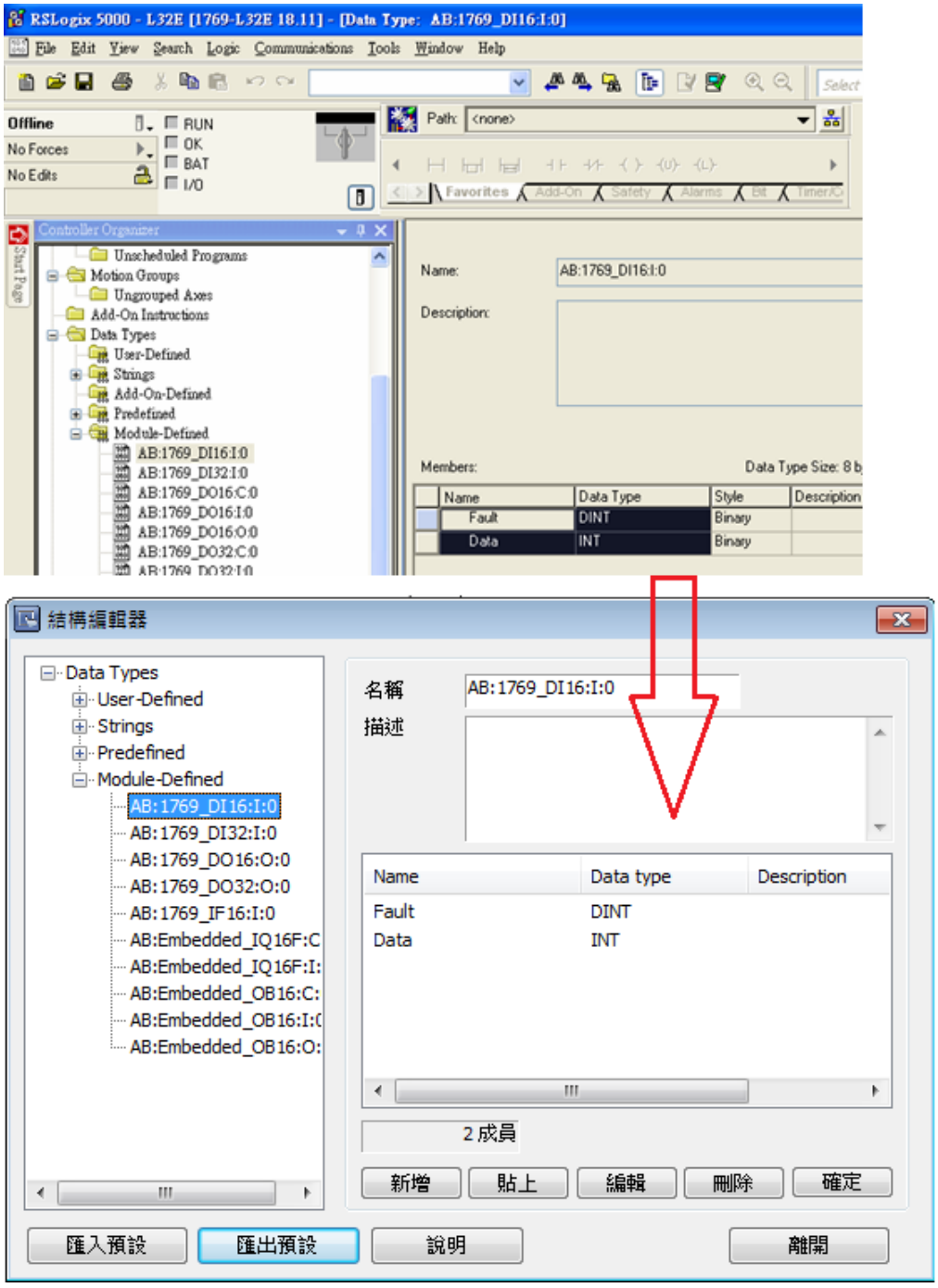
1. 在 RSLogix5000 的 "Controller Organizer"»"Data Types"»"Module-Defined" 对模块的 DataType 双击鼠标左键，就会弹出对话框显示模块的 Data Type 的成员。复制成员里的 Name 和 Data Type。



2. 在 Structure Editor 对着 “Module-Defined” 按下鼠标右键，点选 “New Data Type”。在 New Data Type 的 Name 写入 Module-Defined Name。



3. 按下“Paste”按钮，在对话框中按键盘上的 Ctrl+V 把 Name 和 Data Type 贴上。



4. 点选 Data 再按下“编辑”按钮，因为模块的数据可以用 bit 来操作，所以这里要勾选“二进制存取”，按下“确定”回到 Structure Editor 。

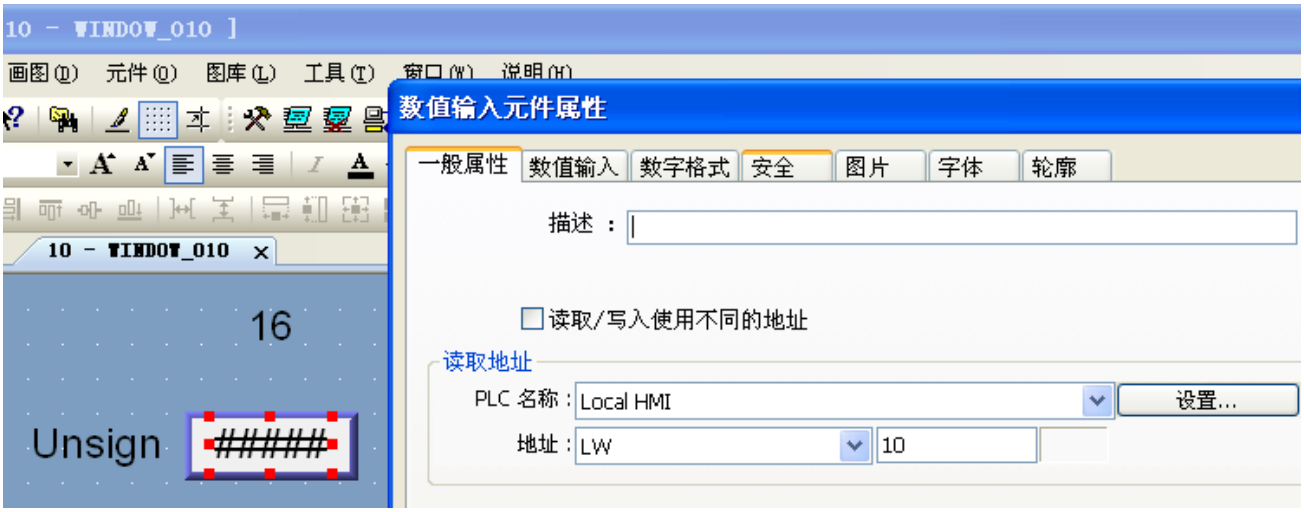
5. 按下“确定”完成设定。

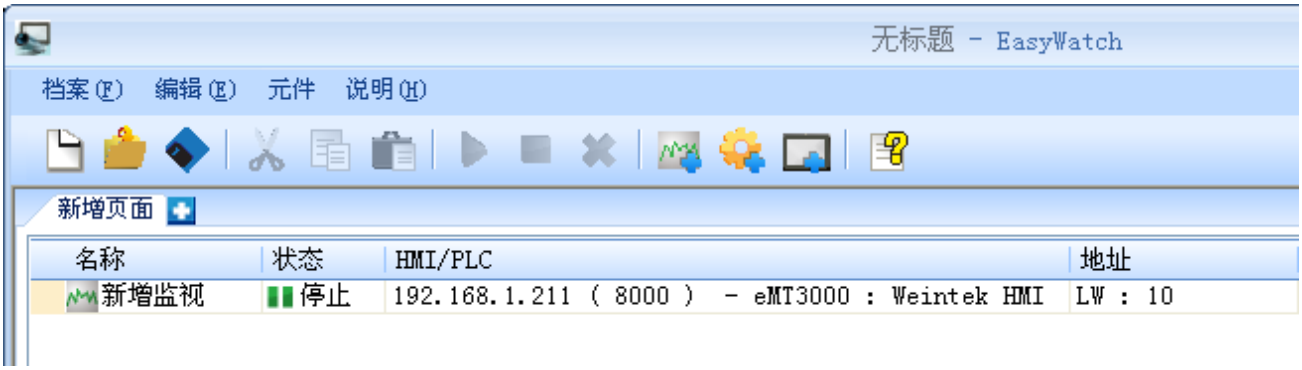
第三十五章 EasyWatch

35.1 概要

EasyWatch 可以通过电脑监看或设定触摸屏和 PLC 内的地址数值，同时也可以进行宏指令的呼叫，更方便用户进行除错及远程监控使用。

以下范例说明，用户如何通过 EasyWatch 查看设定值及数据的正确性。在 EasyBuilder Pro 新增“数值输入”元件，地址设定成 LW-10，再到 EasyWatch 新增相同地址，当执行监视时，状态为已连结，且显示正确数值，表示已联机，即可开始监看。





Note

当系统寄存器“LB-9044 (禁止远程控制)”或“系统参数设置”“»“系统设置”“»“禁止远程触摸屏连接”被设定时，将无法使用 EasyWatch 功能监控。

35.2 设定

35.2.1 基本功能

设定	描述
文件	**新增：**开启一个新的 EasyWatch 文件。 **开启：**开启已编辑的 EasyWatch 文件。 **储存文件：**储存 EasyWatch 文件设定。 **另存新檔：**可将 EasyWatch 文件设定，储存成 .ewt 格式。 **离开：**关闭 EasyWatch。
编辑	**剪下：**剪下选取的元件至剪贴簿中。 **复制：**复制选取的元件至剪贴簿中。 **贴上：**贴上剪贴簿中的元件。
元件	**新增元件：**可新增监视元件或宏元件。 **删除元件：**选择欲删除的元件，系统会弹出讯息，确认是否删除。 **修改元件：**选择欲修改的元件，即可修改内容。 **HMI 管理器：**对触摸屏进行新增、修改、移除的管理。 **执行：**选择欲执行的元件，即可执行此元件。 **停止：**选择执行中的元件，即可停止此元件。
说明	**说明主题：**提供基本功能的操作方法，供用户参考。 **关于 EasyWatch：**显示此版本信息。

35.2.2 快速工具

设定	描述
 **开启新增**	开启一个新的 EasyWatch 文件。
 **开启旧档**	开启已编辑的 EasyWatch 文件。
 **储存文件**	储存 EasyWatch 文件设定。
 **剪切**	剪切选取的元件至剪切板上。

 **复制**	复制选取的元件至剪切板中。
 **粘贴**	粘贴剪切板中的元件。
 **执行**	选择欲执行的元件，即可执行元件。
 **停止**	选择执行中的元件，即可停止元件。
 **删除元件**	选择欲删除的元件，即可删除元件。
 **监视物件**	新增监视物件。
 **宏元件**	新增宏元件。
 **触摸屏管理器**	对触摸屏进行新增、修改、移除的管理。
 **说明主题**	提供基本功能的操作方法，供用户参考。

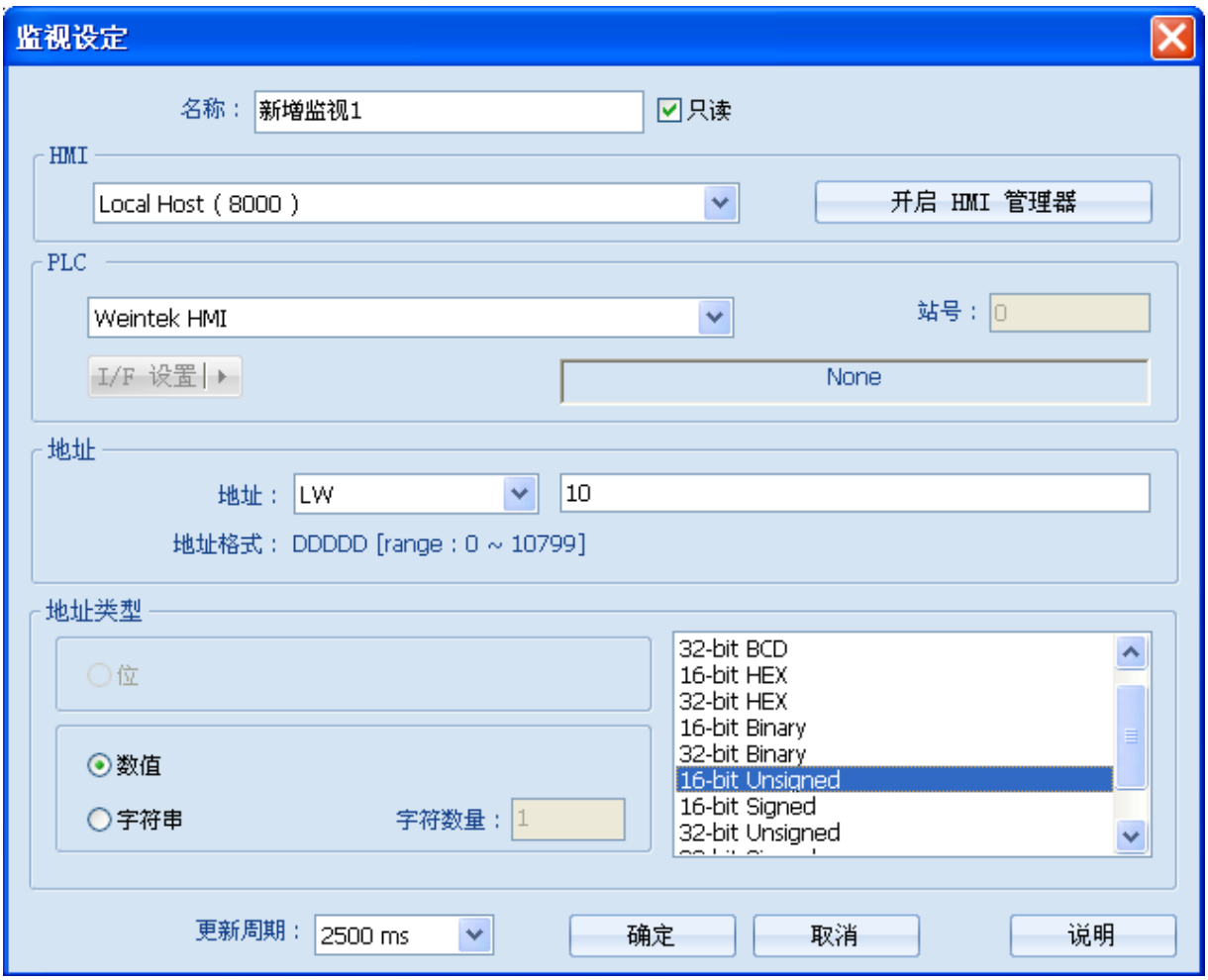
35.3 监视元件设定

35.3.1 新增监视元件

系统提供两种方式新增元件：

- 在基本工具列选择 “元件” » “新增元件” » “新增监视元件”。
- 在快速工具列选择新增监视元件图标。

35.3.2 监视元件设定

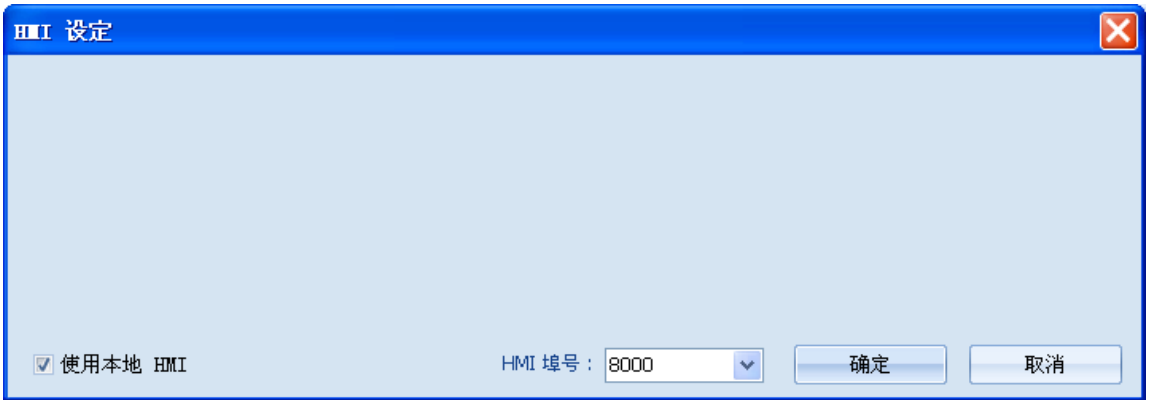


设定	描述
名称	对元件命名，名称不可重复。 **只读：**若元件设为只读时，将不能设定该地址的数值。
HMI	选择欲监视的 HMI。
PLC	设定欲监视地址所属 PLC 的类型、站号及联机方式。
地址	设定欲监视地址的类型及地址。
地址类型	将依照地址类型列出可以选择的显示方式，执行时会依照显示方式来解析并显示该地址。
更新周期	设定监视元件的更新周期，若同时运行过多的元件将导致误差与延迟。

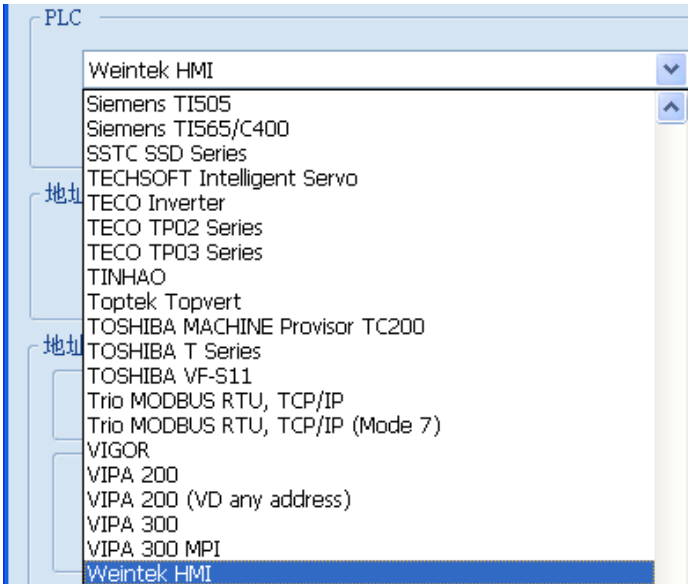
35.3.3 新增监视元件的步骤

1. 选择欲操作的 HMI，若触摸屏不存在，点选“开启触摸屏管理器”»“新增”，可通过网络搜寻 HMI，选择“确定”即可完成新增动作。

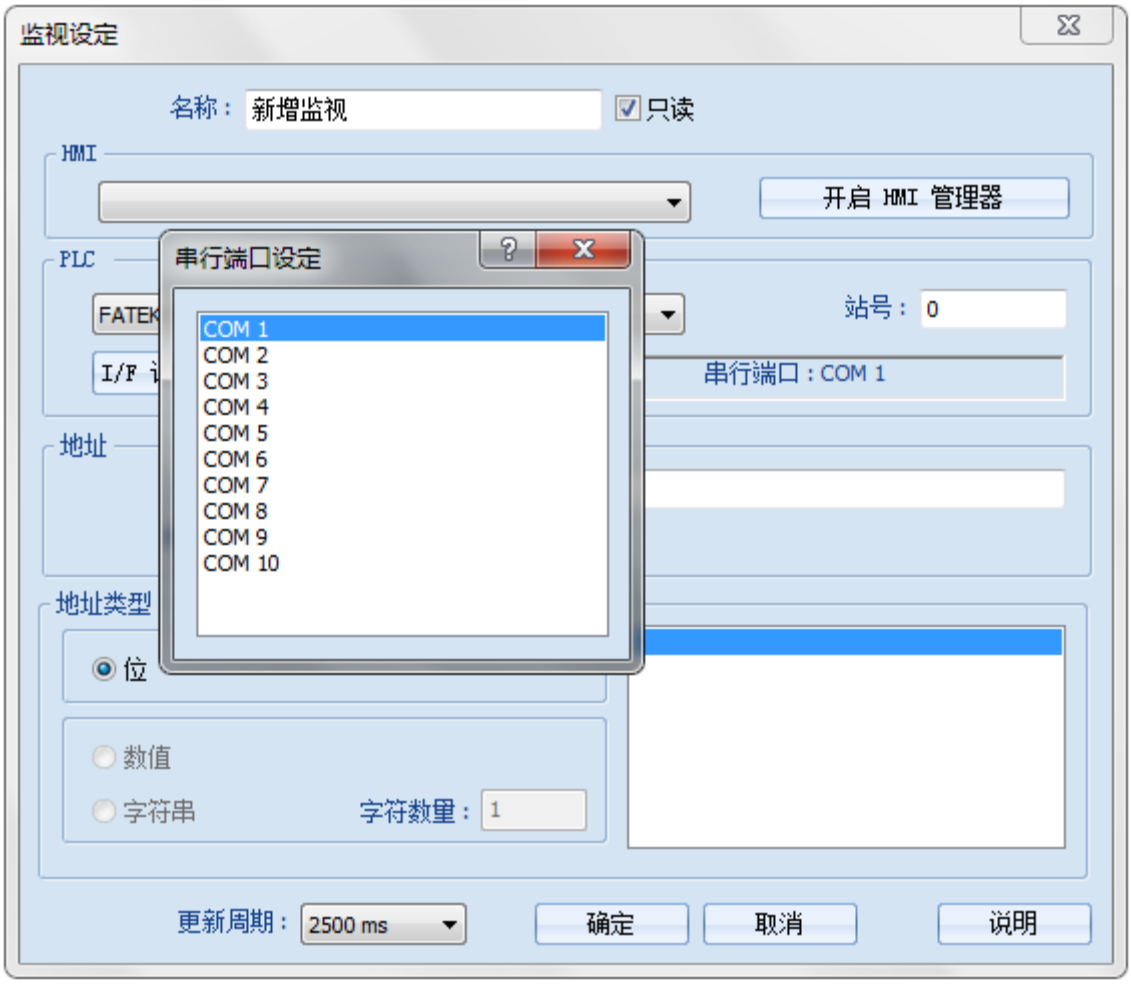
也可勾选“使用本地 HMI”，将以电脑上模拟的程序设为监视设备。



2. 选择直接操作触摸屏或 PLC。若选择 HMI，即可直接对本机触摸屏操作。

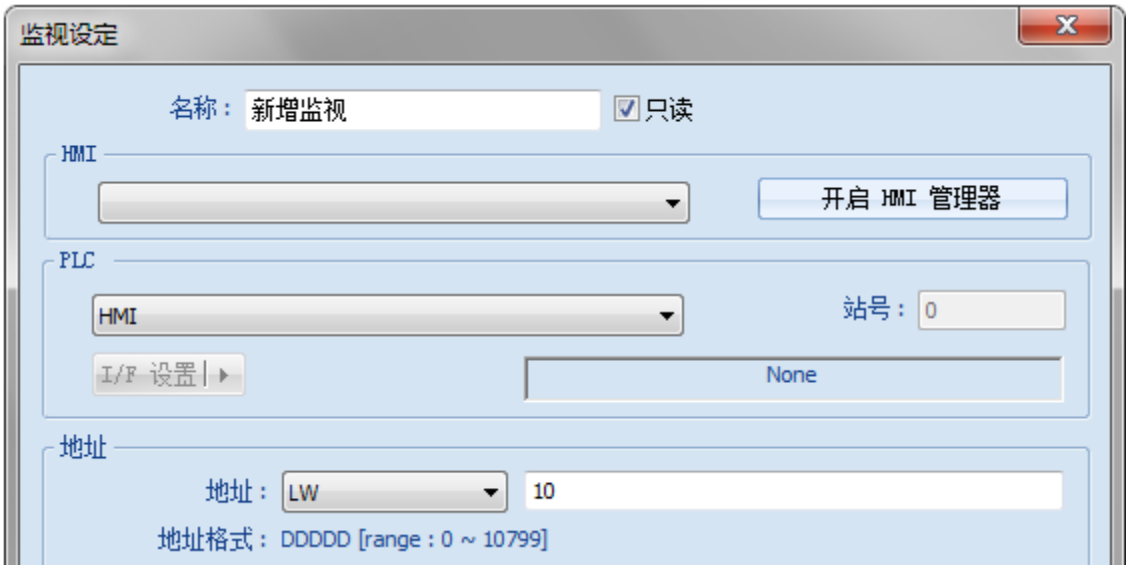


若选择 PLC 时，PLC 连结方式 (I/F 设定) 可使用“串行端口”。



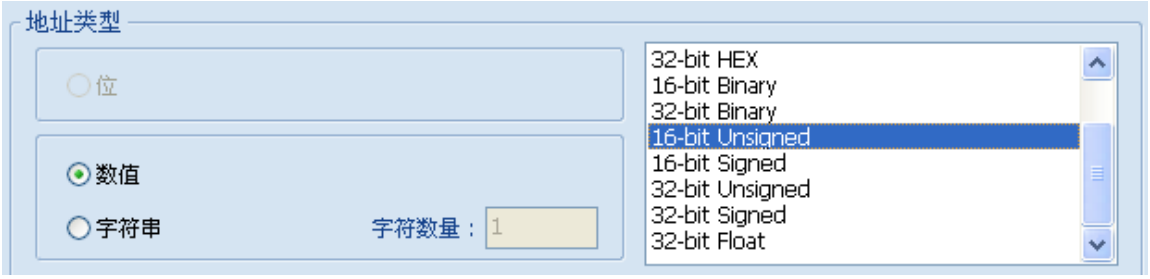
也可以选择“以太网”并设定 IP 地址。

3. 设定欲监控的地址及类型。



4. 当选用字符类型时，可设定该地址为数值或字符串。

“数值”：可选择欲监视的地址的数据格式。



“字符串”：可设定 ANSI、UNICODE、高/低反向三种格式的数据。并可由“字符数量”设定欲读取的字符数量。

5. 设定监视元件的更新周期。可设定范围从 500ms 至 5000ms。

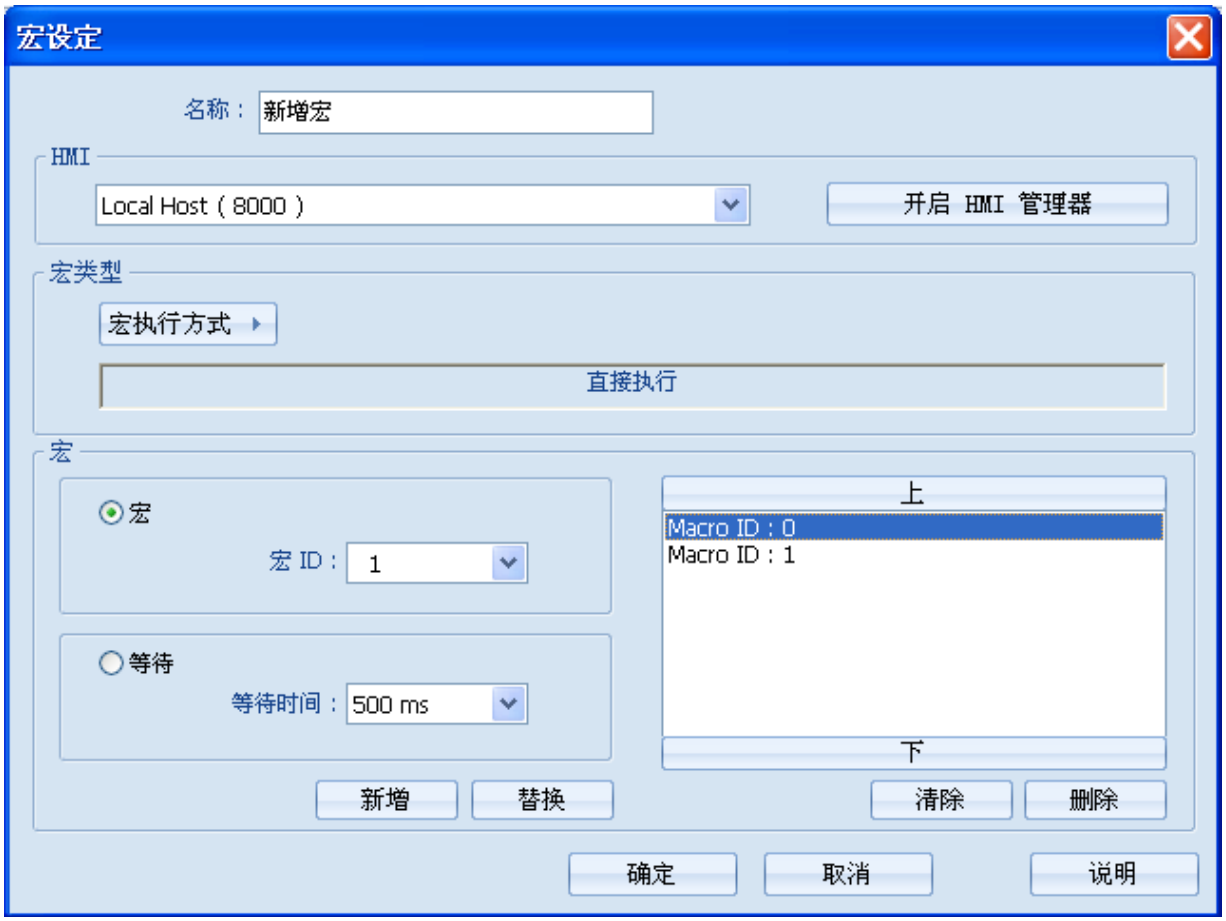
35.4 宏元件设定

35.4.1 新增宏元件

系统提供两种方式新增元件：

- 在基本工具列选择“元件”»“新增元件”»“新增宏元件”。
- 在快速工具列选择新增宏元件图标。

35.4.2 宏元件设定

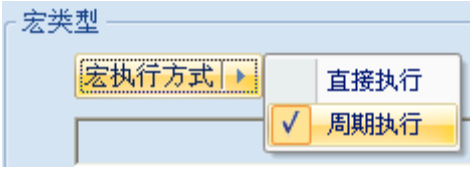


设定	描述
名称	对元件命名，名称不可重复。
HMI	选择欲监视的触摸屏设备。
宏类型	宏执行的方式可分为直接呼叫和周期呼叫。
宏	每个宏元件可执行数个宏指令，且宏与宏执行之间可设定间隔时间。

35.4.3 新增宏设定

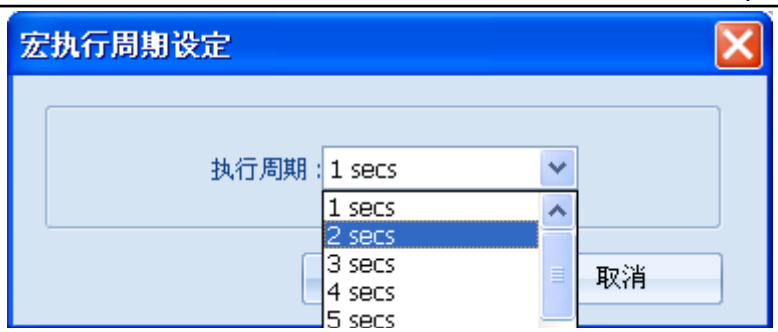
1. 选择 HMI，可参照本章节 《35.3.3 新增监视设定》。
2. 选择宏执行方式，可选择直接执行或周期执行。

“直接执行”：宏指令会直接执行，并执行一次。



“周期执行”：可设定宏指令执行的周期时间。

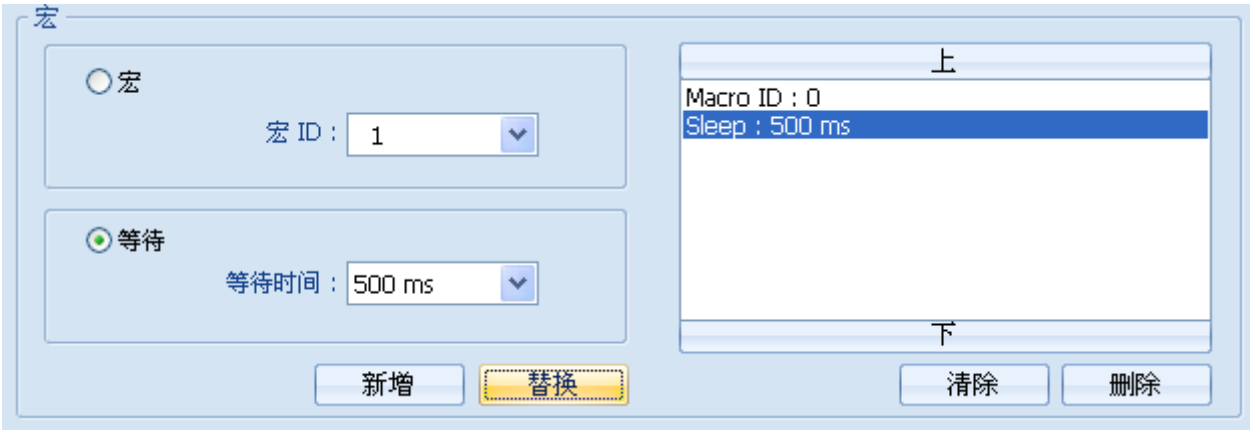
假设在“执行周期”设定为 5 秒，当执行完所有宏指令，将于 5 秒后重新执行此宏元件。



3. 设定宏，包含执行“宏”与“等待”时间。

“宏”：选择要执行的宏 ID，点击“新增”即可新增到宏列表中。

“等待”：选择等待时间，在执行完一个宏指令后，等待所设定的时间，再执行下一个宏指令。点选“新增”或“替换”可新增或取代宏列表中所选取的宏。



35.5 HMI 设定

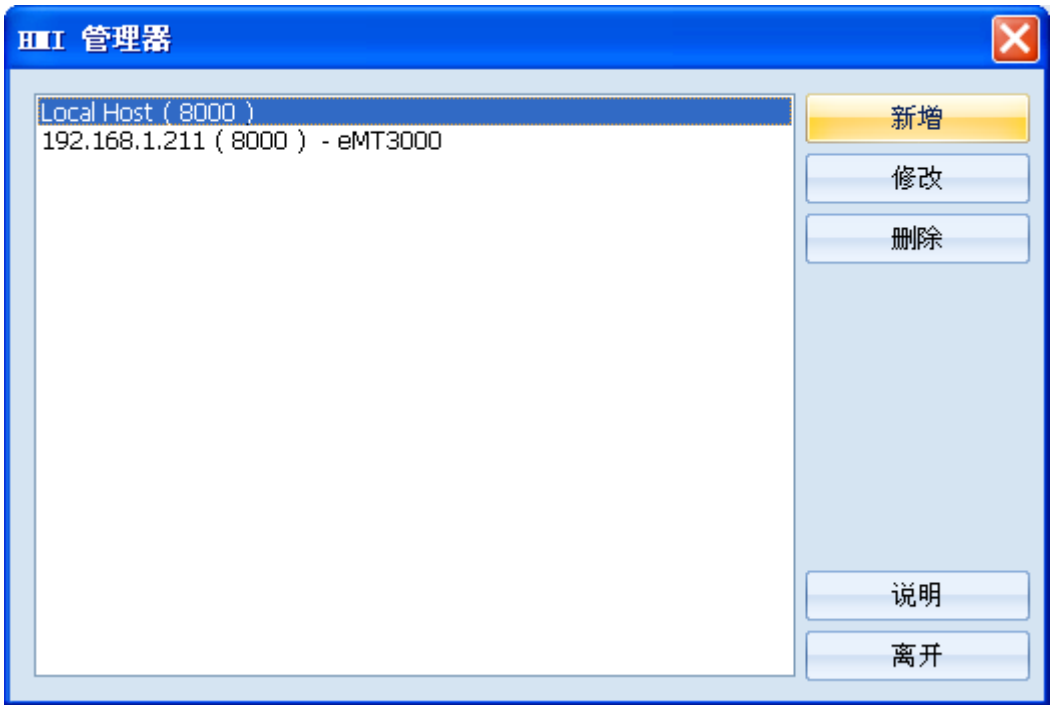
35.5.1 开启触摸屏设定

系统提供两种方式新增元件：

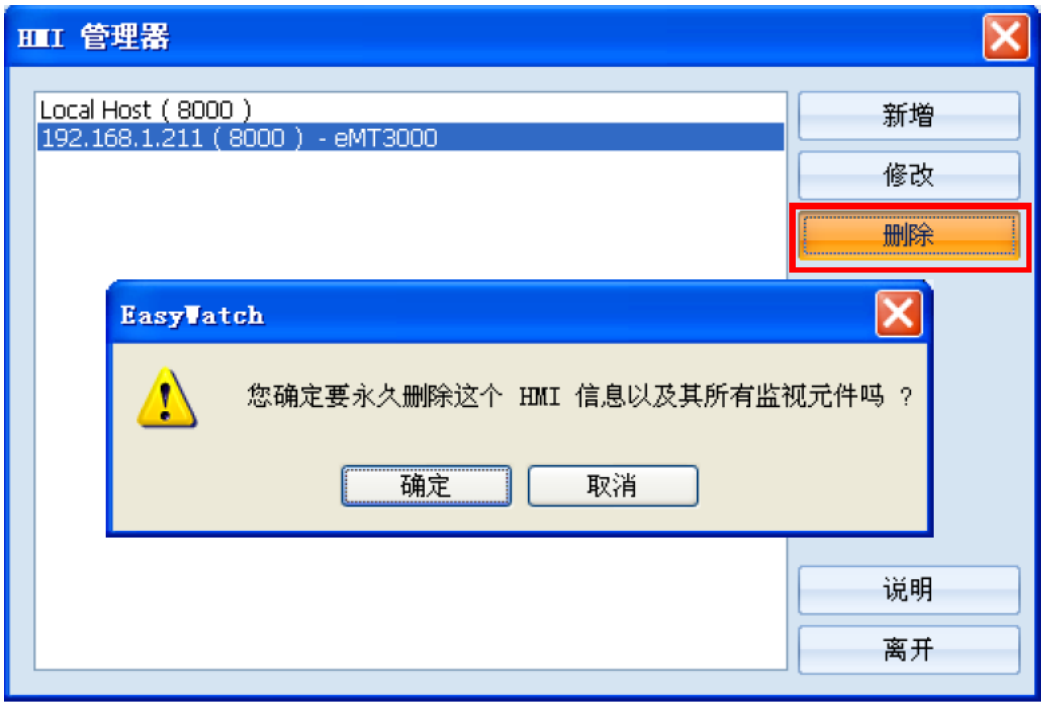
- 在基本工具列选择“元件”»“HMI 管理器”。
- 在快速工具列选择触摸屏管理器图示。

35.5.2 HMI 管理器

系统提供两种方式开启触摸屏设定：

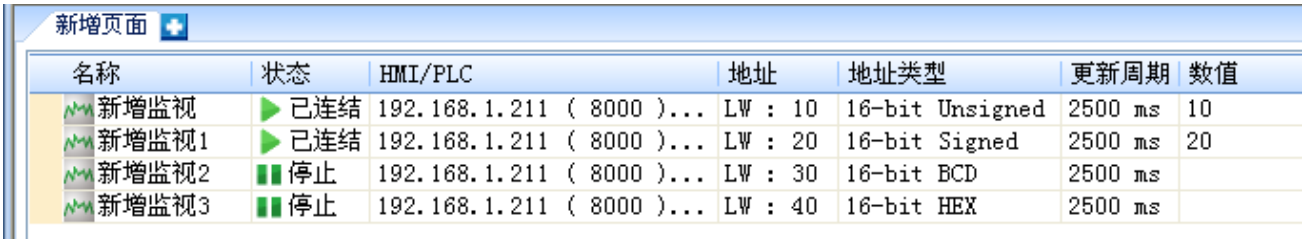


设定	描述
新增	选择 HMI，可参照本章节 《35.3.3 新增监视元件的步骤》。.
修改	选择欲修改触摸屏项目即可修改。
删除	选择欲移除触摸屏项目，确认后即可移除。



35.6 元件显示列表

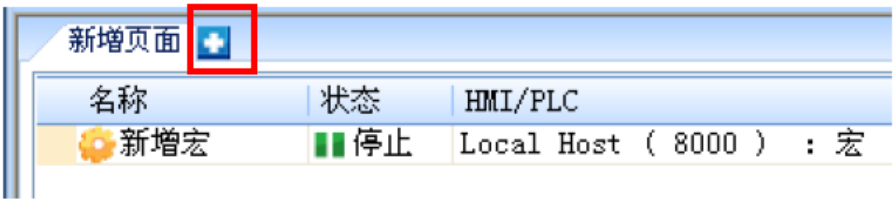
35.6.1 元件显示字段



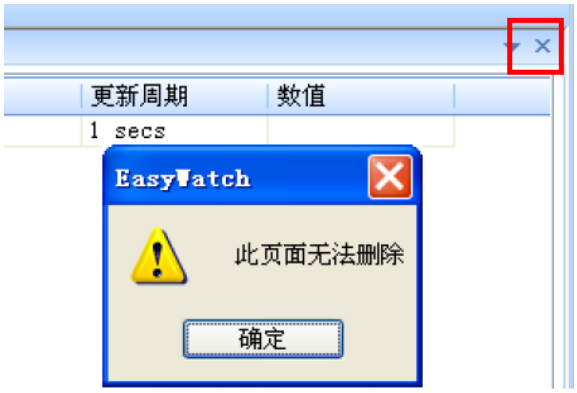
设定	描述
名称	显示元件名称，并且通过图形显示，可以方便用户识别元件的种类。.
状态	显示目前元件的执行状态，分别有“连结中”、”已连结”或是“停止”，同时也可显示错误讯息。若该触摸屏不在在线或是连接埠号输入错误，会显示“无法发现 HMI”的讯息；若为监视元件且地址设定有误，则显示“地址错误”的讯息。
HMI/PLC	显示目前元件所操作 HMI/PLC 的相关信息。
地址 **地址类型**	若为监视元件，将显示其地址相关设定数据。
更新周期	设定监视元件的更新周期。
数值	若为监视元件，且状态为已连结时，将显示目前触摸屏上该地址的数值。当监视元件不具只读属性时，也可通过修改该字段来设定监视地址的内容。若为宏元件，并且类型为“直接执行”，在数值字段上会显示按钮，点击后可直接执行该宏指令。

35.6.2 页面设定

- 新增页面：点选下面图示新增页面。

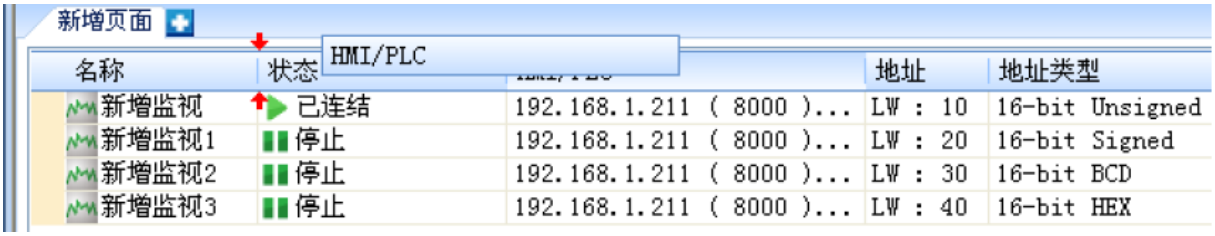


- 删除页面：点选下面图示删除页面。



- 重新命名：在页面的名称上双击鼠标左键后，即可重新命名。

- 变换字段顺序：元件显示字段顺序可依用户喜好自行排列选择。

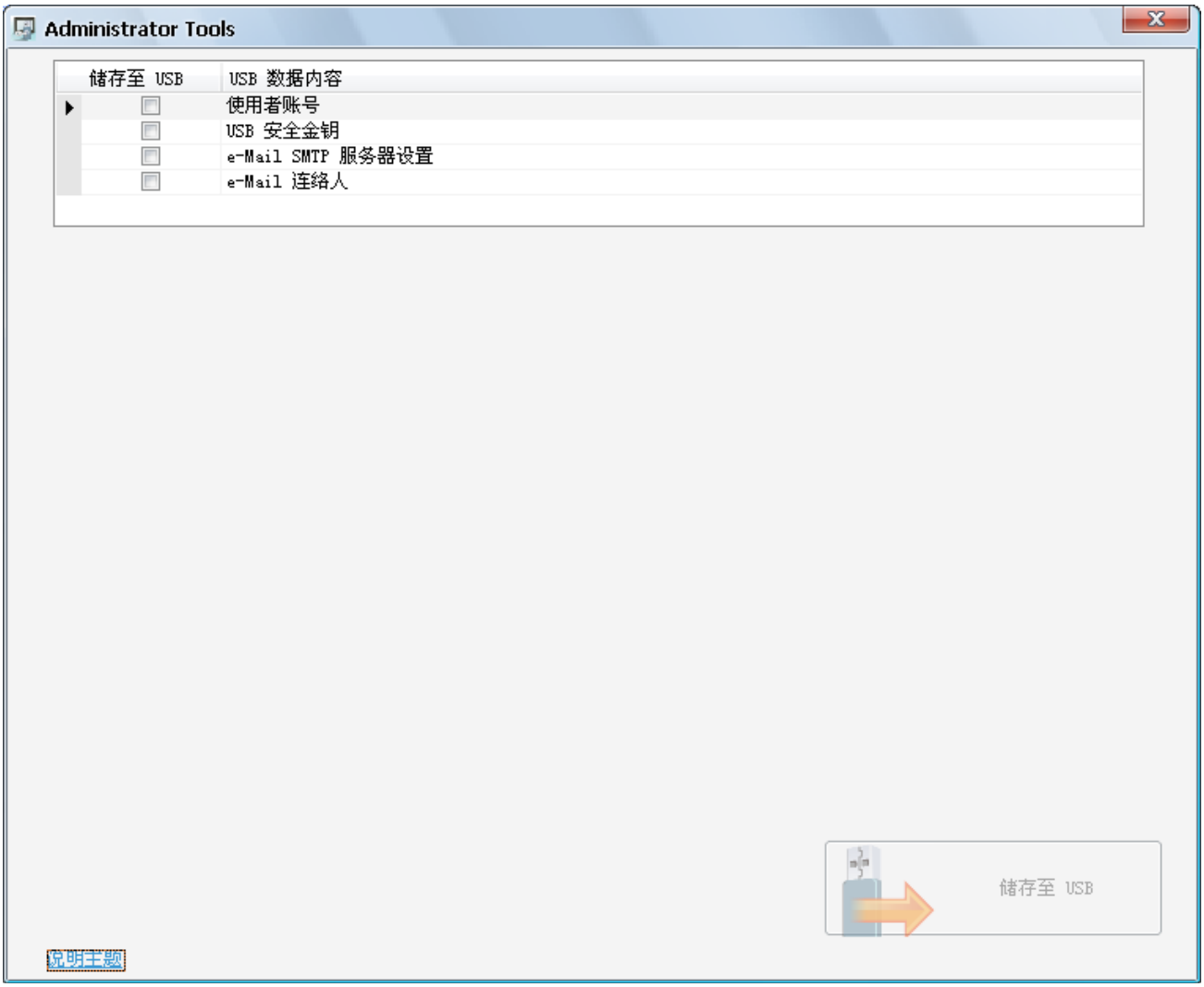


第三十六章 管理员工具

36.1 概要

管理员工具提供将“用户账号”，”USB 安全金钥”，”e-mail SMTP 服务器设置”，”e-Mail 连络人”四种数据储存于 U 盘，并配合 EasyBuilder Pro 用户账户以及 e-Mail 的功能，利用功能键的导入用户数据/使用”USB 安全金钥”，可将所建立的数据导入 HMI，大幅增加数据的可移植性与便利性。

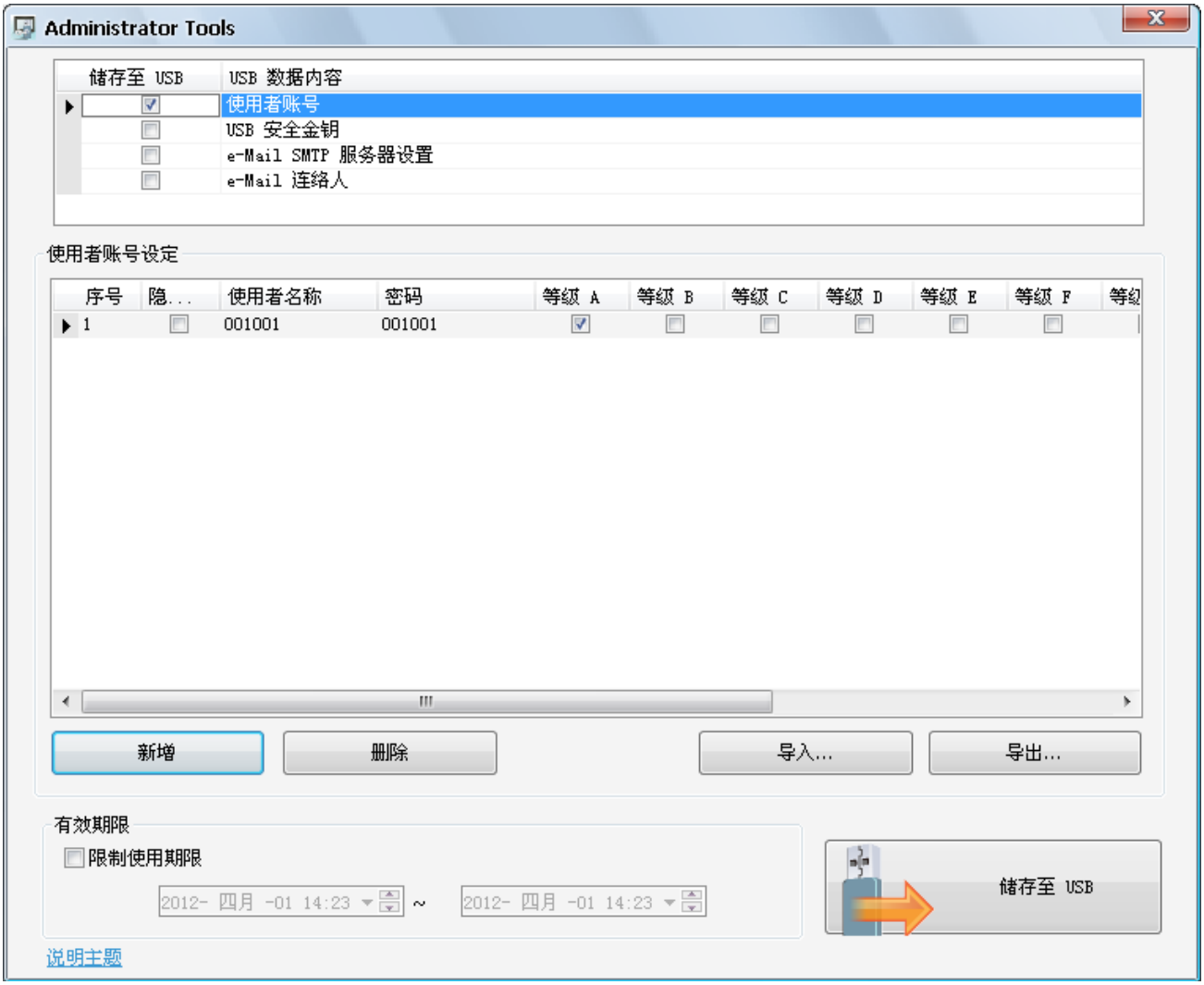
开启管理员工具软件后，在储存复选框中打勾，即可开启该项的编辑功能，后续章节将逐一介绍各个功能的设定。



36.2 用户账号

36.2.1 用户账号介绍

勾选用户账号的复选框，即可进行用户账户数据的设定。



设定	描述
隐藏用户	选择此账户是否为隐藏用户。
使用者名称	使用者名称。*Note 1
密码	用户密码。*Note 1
类别 A ~ L	用户权限。
新增	新增一笔账户资料。*Note 2
删除	删除一笔账户数据。
导入	导入用户账户数据。

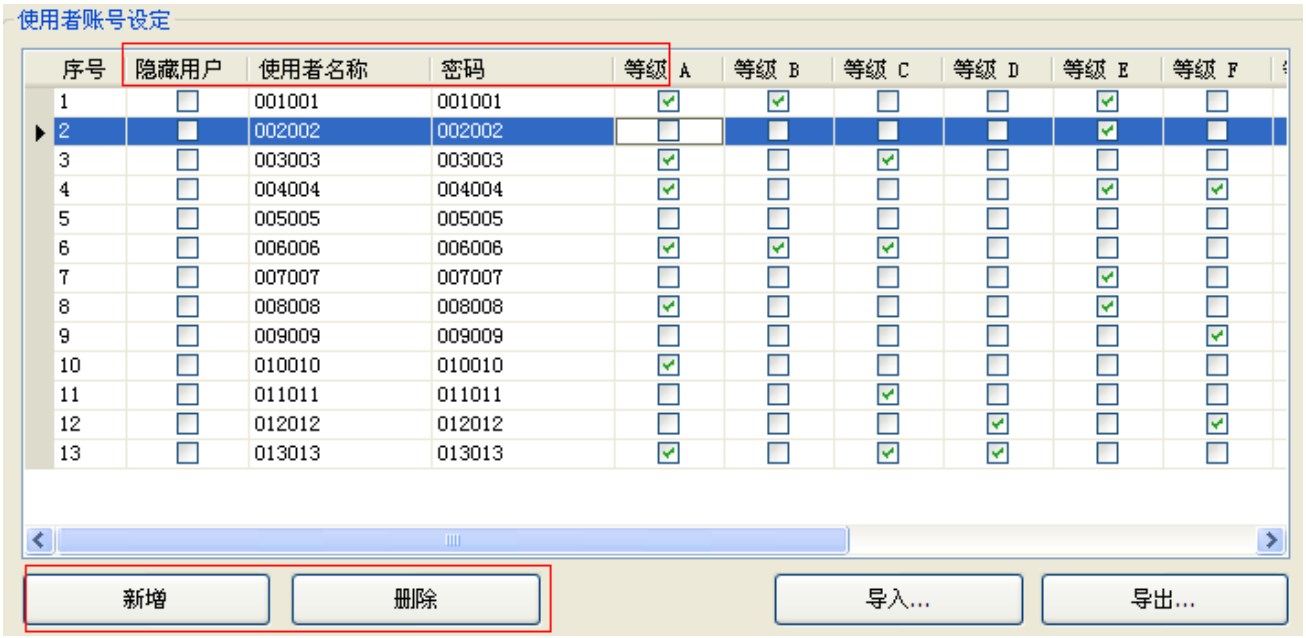
导出	导出用户账户数据。
有效期限	没有勾选“限制使用期限”，将数据导入 HMI 后，数据是永久有效。 若勾选”限制使用期限”并且设定有效期限后，数据需要在有效期限内导入 HMI，若是过了有效期限才导入 HMI，则数据无法被导入，请重新使用此工具制作一份新的数据。
储存至 USB	将数据储存至 USB。若要储存至目录，可点选“▼”后选择目录。 

Note

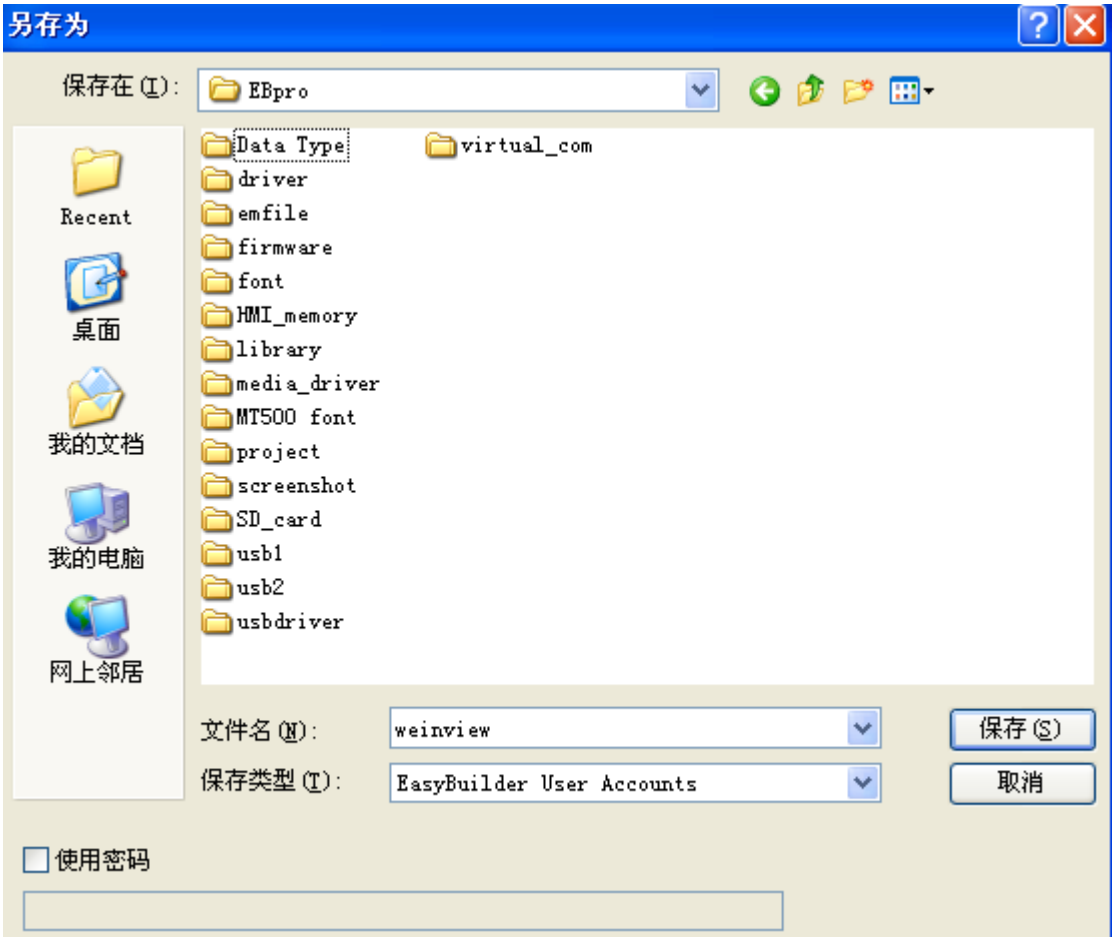
1. 可以是字母，数字符号，“-”，“_”所构成，大小写视为不同。
2. 最多可新增 127 笔账户资料。

36.2.2 用户账号设定

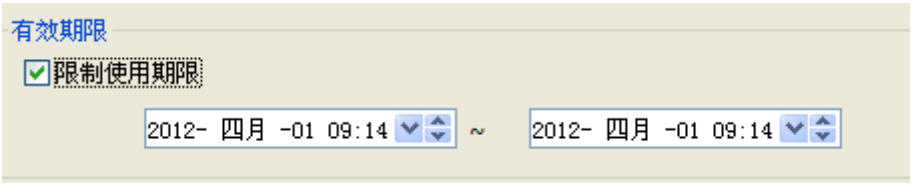
1. 点选“新增”可新增一笔用户数据，选点“删除”即可删除该笔账户数据，勾选“隐藏用户”则该账号成为隐藏用户，在“用户名称”输入用户名称，“密码”输入用户密码，并勾选该账号可使用的权限“类别 A”~“类别 L”。



2. 将账户数据建立完成之后，点选“导出”可将数据导出备份，左下角可选择是否勾选“使用密码”对此数据做保护，日后若需要重新建立或修改只要再点选“导入”并输入保护密码，即可导入该数据。



3. 若勾选了“有效期限”»“限制有效期限”则代表限制只有在此范围时间内，才允许 U 盘将用户账户数据做导入触摸屏的动作，若没有勾选则代表没有限制范围，即任何时间皆可以做导入触摸屏的动作。



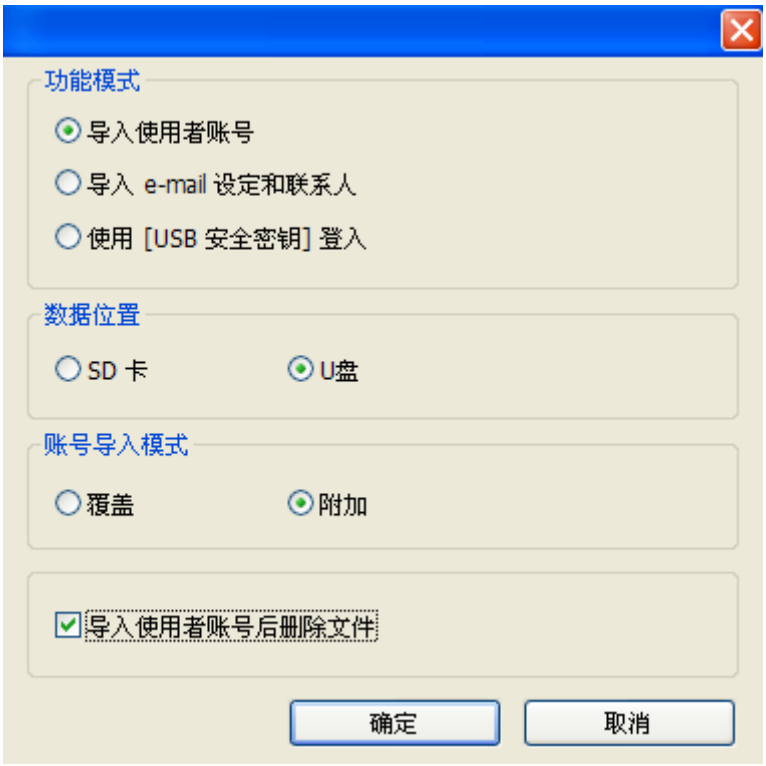
4. 全部都设定完成后，点选“储存至 USB”，选择要储存的 U 盘位置，点选“建立”，建立成功会出现“建立成功!”的讯息。

36.2.3 使用 EasyBuilder Pro 导入账户

利用 EasyBuilder Pro 所建立的”功能键”元件，在动态操作面板触发此元件时，方可执行此导入的动作，以下介绍功能键建立的步骤：

1. 在 EasyBuilder Pro 选择”功能键”元件，选择”导入用户数据 / 使用 “USB 安全金钥”，接着按 “设置”。

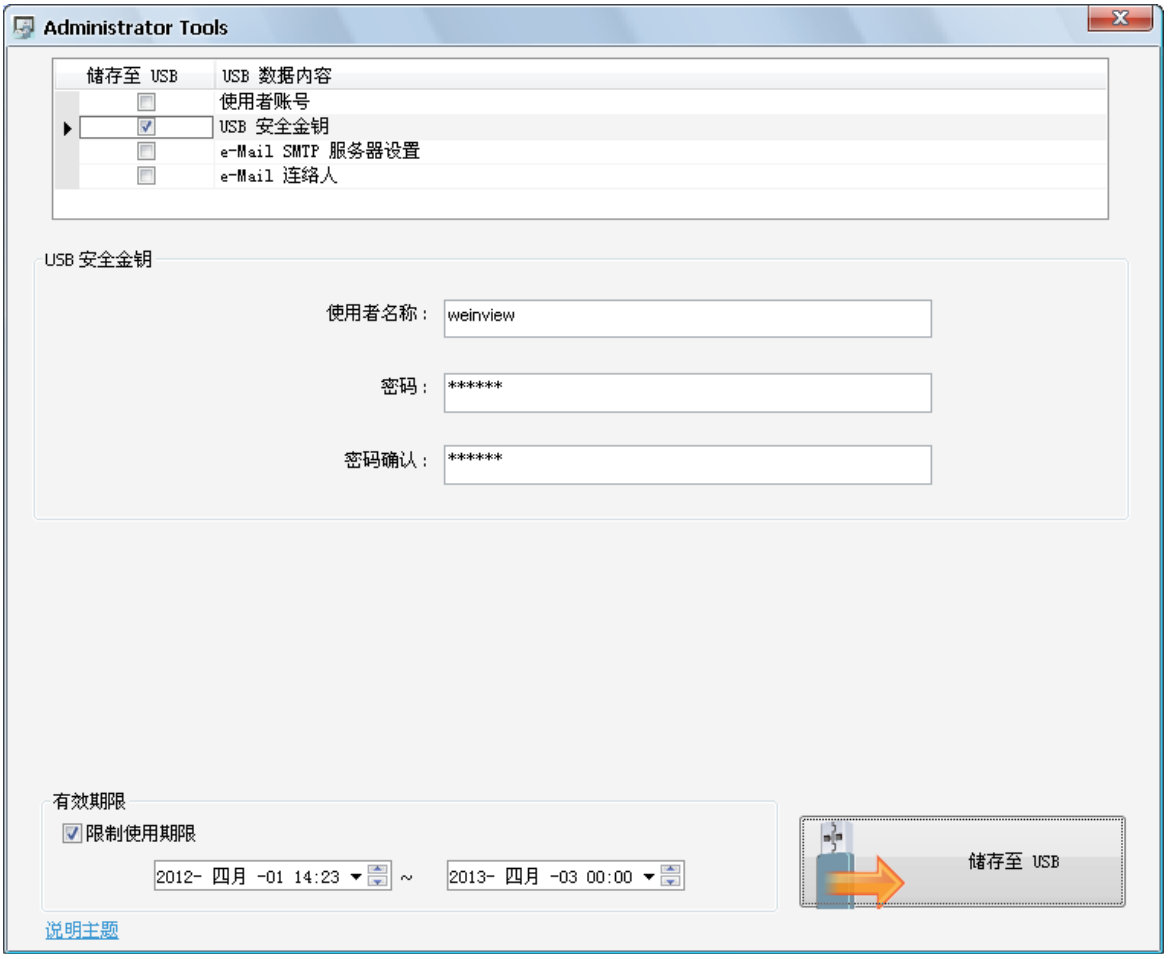
2. 在 “功能模式” 选择 “导入用户账号”，”数据位置” 依用户欲导入的设备决定，”账号导入方式” 选择 “覆盖”，触摸屏内将只保存此次导入的账号数据，若是选择 “附加”，触摸屏内账号数据将保留，并加入此次导入的新账号数据；勾选 “导入用户账号后删除文件” 在导入账号后，将删除来源数据文件，接着按 “确定” 即完成设定。


功能模式
导入使用者账号
导入 e-mail 设定和联系人
使用 [USB 安全密钥] 登入
数据位置
SD 卡
U盘
账号导入模式
覆盖
附加
导入使用者账号后删除文件
确定
取消

36.3 USB 安全金钥

36.3.1 USB 安全金钥介绍

勾选“USB 安全金钥”复选框即可进行用户安全金钥的设定，预先设定好用户的登录信息，即可利用用户安全金钥设定直接登入账户，设定画面如下所示：



设定	描述
用户名称	用户名称。*Note 1
密码	用户密码。*Note 1
密码确认	确认用户密码。
有效期限	在此有效期限内可以在触摸屏上使用安全金钥登入，若没有勾选则代表无限制。
储存至 USB	将数据储存至 USB。

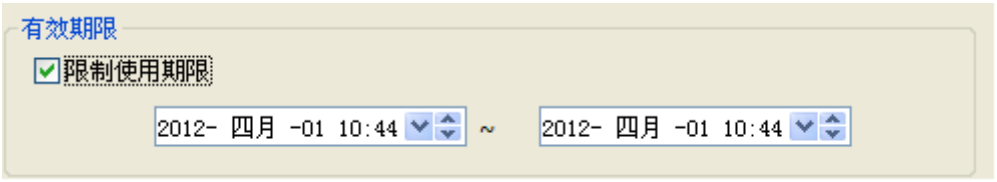
Note

1. 可以是字母，数字符号，“-”，“_”所构成，大小写视为不同。

36.3.2 USB 安全金钥设定

1. 在“用户名称”输入用户的账户名称，”密码”输入用户的密码，”密码确认”重复输入密码确认。

2. 若勾选了“有效期限”»“限制有效期限”则代表限制只有在此范围时间内，才允许用户用此 USB 安全金钥登入功能，若没有勾选则代表没有限制，任何时间皆可以执行此功能。



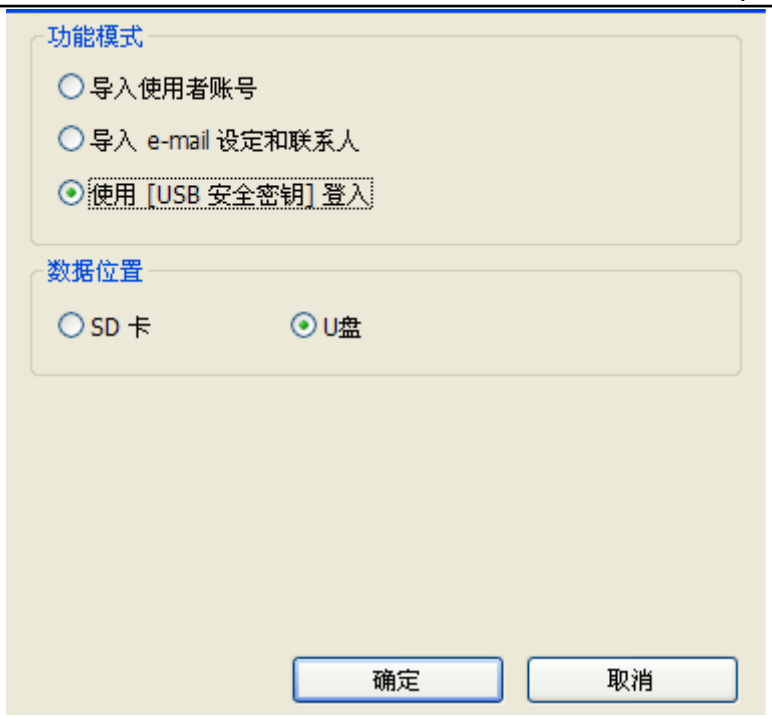
3. 全部都设定完成后，点选“储存至 USB”，选择要储存的 U 盘位置，点选“建立”，建立成功会出现“建立成功!”的讯息。

36.3.3 使用 EasyBuilder Pro 设定 USB 安全金钥

利用 EasyBuilder Pro 所建立的“功能键”元件，在动态操作面板触发此元件时，方可执行 USB 安全金钥登入的动作，以下介绍“功能键”建立的步骤。

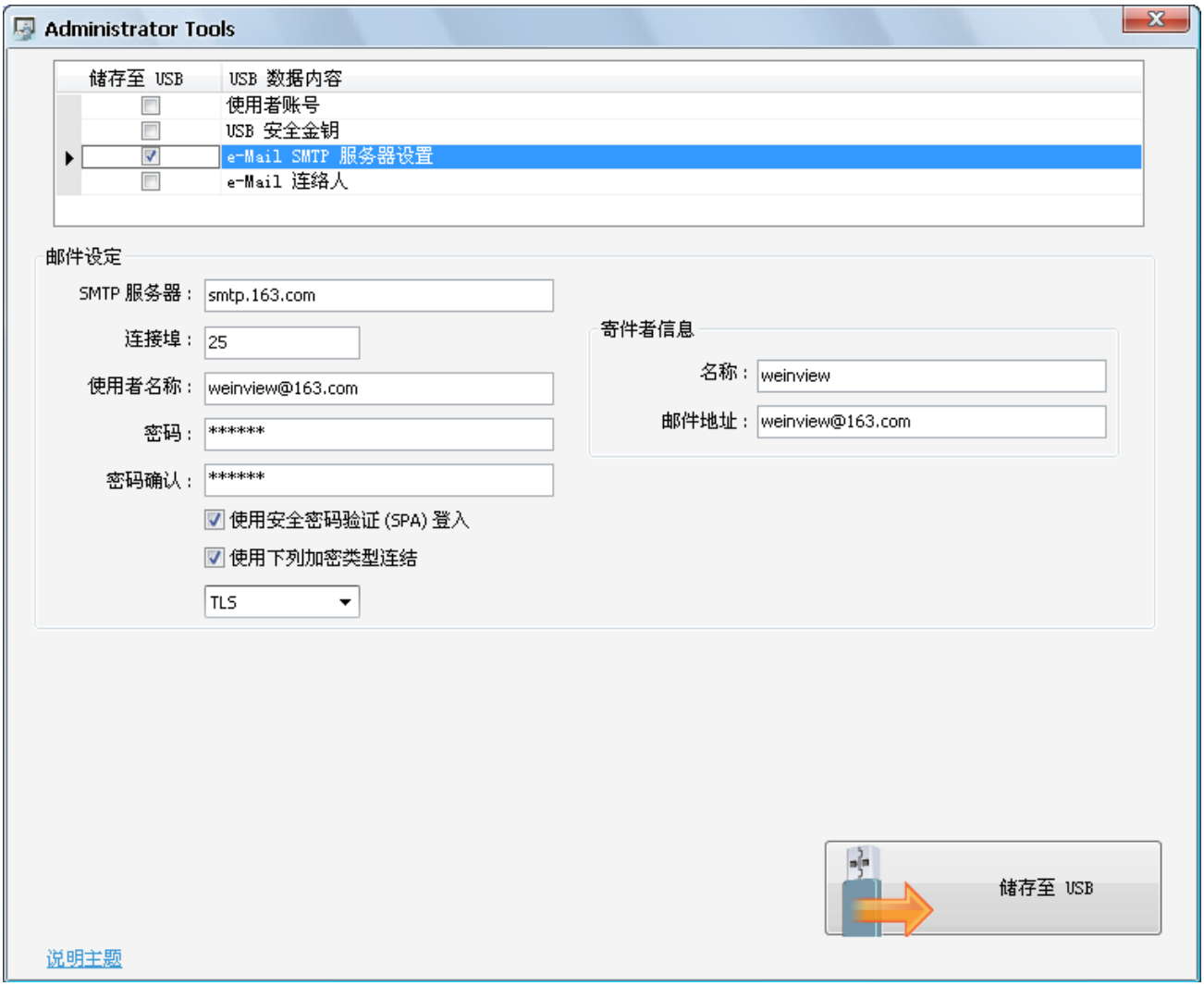
1. 在 EasyBuilder Pro 选择“功能键”的元件，选择“导入用户数据/使用”USB 安全金钥””，接着按“设置”。

2. 在“功能模式”选择“使用“USB 安全金钥登入””，”数据位置”依安全金钥文件存放的设备决定，按“确定”即完成设定。


功能模式
导入使用者账号
导入 e-mail 设定和联系人
使用 [USB 安全密钥] 登入
数据位置
SD 卡
U盘
确定
取消

36.4 e-Mail SMTP 服务器

勾选“e-Mail SMTP 服务器设置“ 的复选框，即可进行外寄邮件服务器的设定。



邮件设定	描述
SMTP 服务器	外寄邮件服务器。
连接埠	外寄邮件服务器连接端口号码。
用户名称	用户邮件账号名称。
密码	用户邮件账号密码。
密码确认	确认用户邮件账号密码。
寄件者设定	**描述**
名称	收件时显示寄件者的名称。
邮件地址	收件时显示寄件者的邮寄地址。
储存至 USB	将数据储存至 USB。

36.4.1 e-Mail SMTP 服务器设定

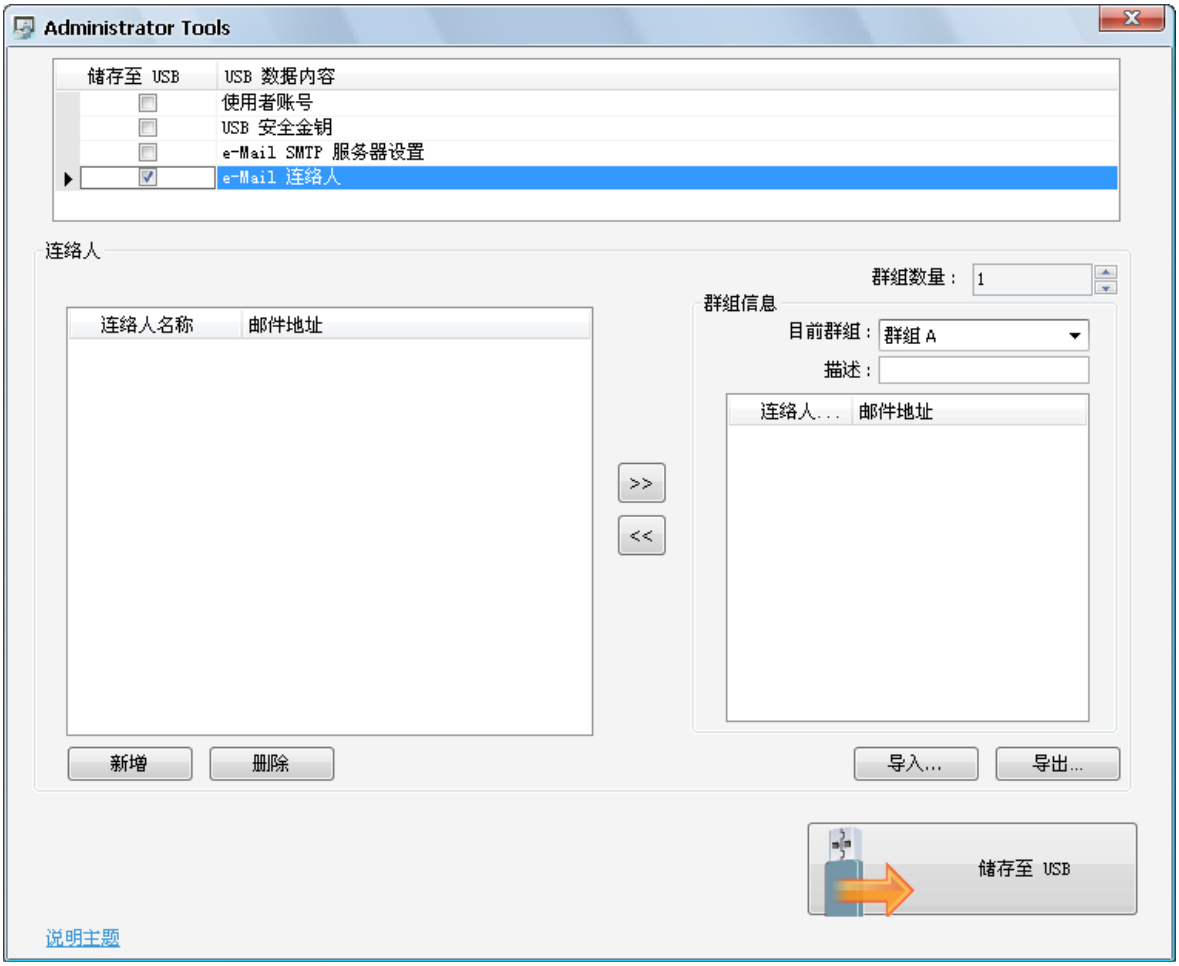
1. 以下依照 e-mail SMTP 设定内容。

2. 全部都设定完成后，点选“储存至 USB”，选择要储存的 U 盘位置，点选“建立”，则会出现“建立成功!”的讯息。

36.5 e-Mail 联络人

36.5.1 e-Mail 联络人介绍

勾选“e-Mail 连络人”的复选框，即可设定邮件连络人。



设定	描述
新增	新增一笔联络人数据。 *Note 1
删除	移除一笔联络人数据。
群组数量	群组数目。*Note 2
目前群组	目前群组名称。 *Note 3
描述	群组描述。
导入	导入联络人信息。
导出	导出联络人信息。
储存至 USB	将数据储存至 USB。

Note

1. 最多可新增 256 笔联络人数据
2. 最多可新增 16 个群组 (群组 A ~ 群组 P)
3. 群组 A ~ 群组 P ，当群组数量为 1 时，只会存在群组 A ，增加至 2 时，则会同时存在群组 A 与群组 B ，以此类推。

36.5.2 e-Mail 联络人设定

1. 按“新增”新增完成所有邮件通知元件。
2. 将连络人加入到群组 A 之中，可以看到被加入到此群组的连络人显示的颜色是红色。

3. 设定群组数量，即可新增群组，此时会新增一个群组 B 。重复步骤 a 与步骤 b 将联络人加入群组之中。

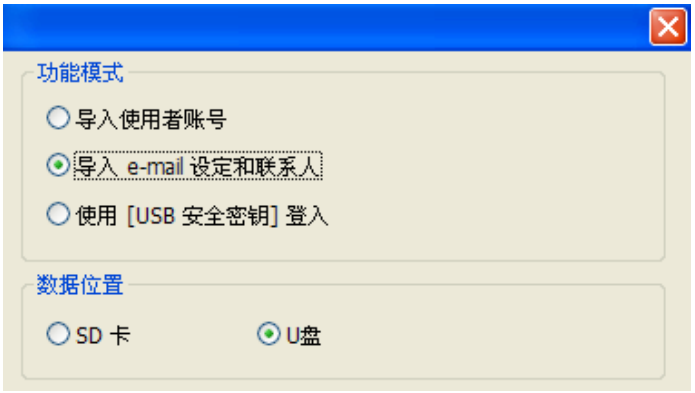
4. 将 e-Mail 连络人数据建立完成之后，点选“导出”可将数据导出备份，日后若需要重新建立或修改只要再点选“导入”导入该数据即可。

5. 全部都设定完成后，点选“储存至 USB”，选择要储存的 U 盘位置，点选“建立”，建立成功会出现“建立成功!”的讯息。

36.5.3 使用 EasyBuilder Pro 导入 e-Mail 设定和连络人

利用 EasyBuilder Pro 所建立的“功能键”元件，在动态操作面板触发此元件时，方可执行此导入的动作，以下介绍“功能键”建立的步骤。

1. 在 EasyBuilder Pro 选择“功能键”的元件，选择“导入用户数据/使用”USB 安全金钥”，接着按“设置”。
2. 在“功能模式”选择“导入 e-mail 设定和连络人”，”数据位置”依用户欲导入的设备决定，接着按“确定”即完成设定。



第三十七章 MODBUS TCP/IP 网关功能

37.1 概要

以往若要使用 SCADA *(Supervisory Control and Data Acquisition)* 软件去存取与触摸屏连接的 PLC 数据时，需通过数据传输先将 PLC 数据传送至触摸屏的本地地址，再于电脑 上使用 MODBUS TCP/IP 通讯协议去读取触摸屏的本地地址将 PLC 数据取回。

现在用户可以通过 EasyBuilder 提供的 MODBUS TCP/IP 网关功能，将 MODBUS 与 PLC 的地址预先设定对应后，即可以直接利用 MODBUS TCP/IP 通讯协议存取 PLC 上的数据。

37.2 如何建立一个地址对应表

新增一个地址对应表，请依照下列步骤：

1. 于“系统参数设定 \ 设备清单”新增欲监控的 PLC 设备。(以 FATEK FB Series 为例)
2. 新增一个 MODBUS Server (以太网)，并启用“MODBUS TCP/IP 网关”，如下图：

3. 点选“地址对应表”按钮后，会显示预设的对应表，用户可以依需求修改并新增其它对应表。

4. 假设，SCADA 需存取 FATEK FB Series PLC 的 D0 寄存器开始的连续 50 个地址，设定如下：

(1) 设定欲对应的寄存器类型，此范例为“字符”。

(2) 设定欲对应的寄存器之存取模式，此范例为“读/写”。

(3) 设定欲对应的 MODBUS 起始地址，此范例为“4x1”。

(4) 设定欲对应的 PLC 起始地址，此范例为“D0”。

(5) 设定欲对应地址的范围大小，此范例为“50”。

(6) 选择是否要高/低字节或高/低字符组转换。

对应表	描述	MODBUS 地址		PLC 名称	PLC 对应地址	长度	读/写
1	Access D0 ~ D49	4x-1	<==>	FATEK FB Series	D-0	50 字符	读/写

上图的设定内容说明 MODBUS Server 4x1 ~ 4x50 地址对应到 FATEK FB Series PLC 的 D0 ~ D49 地址。

5. 完成以上设定后，SCADA 只需利用 MODBUS TCP/IP 协议，发送读/写 4x1 ~ 4x50 地址的命令，即可以直接存取 FATEK FB Series PLC 的 D0 ~ D49 地址。

37.3 地址对应设定须知

- “MODBUS TCP/IP 网关”功能不支持使用 UDP。
- 只支持使用 MODBUS Server (以太网) 接口。
- 系统提供寄存器 LW-9288，可用来指示此功能数据传送是否正常。

各错误码表示如下：

数值	定义
0	正常
1	读取或写入未定义在地址对应表中的寄存器
2	读取或写入的地址范围超出单一地址对应表所定义的数据长度 (或是读取/写入跨表格的寄存器)
3	命令格式未遵循 MODBUS TCP/IP 通讯协议
4	修改只允许读取的寄存器
5	读取只允许写入的寄存器
6	在设定的时间内无法得到 PLC 的正确响应
7	使用了 MODBUS Server 不支持的功能码

- 各个对应表间定义的寄存器之地址范围不可重复。
- 启用“MODBUS TCP/IP 网关”功能后，EasyBuilder 将取消 MODBUS Server 与触摸屏地址间原有的对应关系，包含：

  (1) 0x, 1x 对应到 LB

  (2) 3x, 4x 对应到 LW, RW

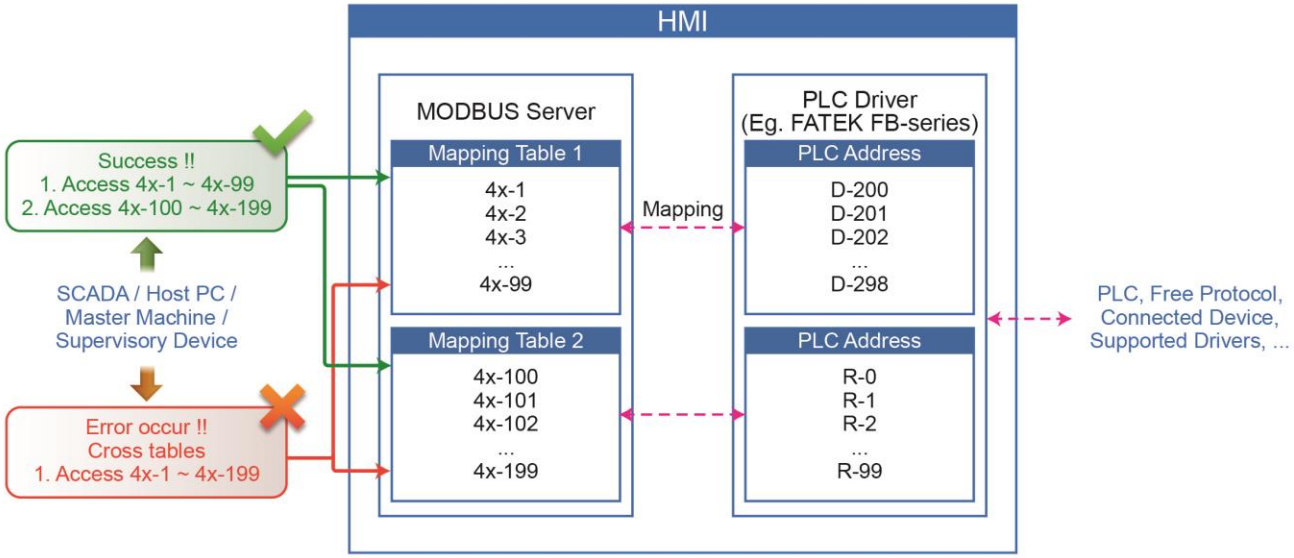
  因此如需通过 0x, 1x, 3x, 4x 的命令来存取 LB 或 LW 的数据，仍需先将地址对应关系重新设定于“地址对应表”中，可参考下列设定内容。

对应表	描述	MODBUS 地址		PLC 名称	PLC 对应地址	长度	读/写
1	0x <==> LB	0x-1	<==>	Local HMI	LB-0	12400 位	读/写
2	1x <==> LB	1x-1	<==	Local HMI	LB-0	12400 位	只读
3	3x <==> LW	3x-1	<==	Local HMI	LW-0	9999 字符	只读
4	4x <==> LW	4x-1	<==>	Local HMI	LW-0	9999 字符	读/写
5	3x <==> RW	3x-10000	<==	Local HMI	RW-0	55536 字符	只读
6	4x <==> RW	4x-10000	<==>	Local HMI	RW-0	55536 字符	读/写

- SCADA 一次只能读取/写入一个对应表内的寄存器，即无法使用同一个 MODBUS 命令存取不同表格中的寄存器。

对应表	描述	MODBUS 地址		PLC 名称	PLC 对应地址	长度	读/写
1	Access D200 ~ D298	4x-1	<==>	FATEK FB Series	D-200	99 字符	读/写
2	Access R0 ~ R99	4x-100	<==>	FATEK FB Series	R-0	100 字符	读/写

以上图为例，于“对应表 1”设定 MODBUS 4x1 对应到 D200 地址，长度为 99；于“对应表 2”设定 MODBUS 4x100 对应到 R0 地址，长度为 100，若此时 SCADA 发出一道命令要一次读取 4x 1 ~ 4x199 长度为 199 的地址，因已经跨表格存取，此命令将不被触摸屏接受，应将命令分为两道分别存取 4x1 ~ 4x 99 和 4x100 ~ 4x199。如下图：



WEINVIEW

WEINVIEW

400-836-2000

CALL CENTER 技术支持专线

周一至周五 9：:0-17:30(法定节假日除外）

威纶通科技有限公司

www.weinview.cn

[深圳]

地址:深圳市南山区南海大道和登良路交汇处恒裕中心B座十层

电话:0755-26456333

传真:0755-86036588

[天津]

地址:天津市新技术开发区华苑产业园榕苑路2号海益国际3号楼604

电话:022-23853058

[成都]

地址:成都市武侯区人民南路四段12号华宇蓉国府一栋一单元414

电话:028-85516173

传真:028-85516173

[苏州]

地址:苏州市工业园区万盛街28号邮政7楼

电话:0512-69001690

传真:0512-62990145

[武汉]

地址:武汉市武昌区中南路7号中商广场写字楼B座B3301室

电话:027-87269732

传真:027-87269733

[青岛]

地址:青岛市城阳区正阳中路192号国贸大厦729室

电话:0532-55693209

传真:0532-55693210