通过扩展脚本与 OMRON 温控器通讯
说明书

普洛菲斯国际贸易（上海）有限公司
技术热线：**021-6361-5008**

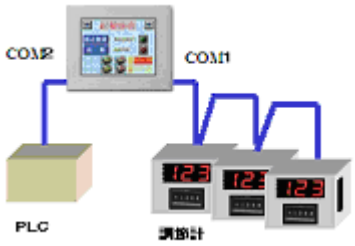
1.0版

目录

注意：在您的系统中使用本例时，在操作前请检查。

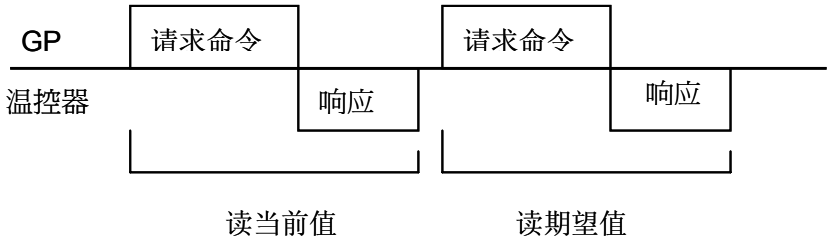
1. 概述

通过下面的步骤，您可以将 OMRON 温控器与 Modicon Modbus（SLAVE） PLC 同时连接在 GP 上，GP 作为终端站点显示信息。

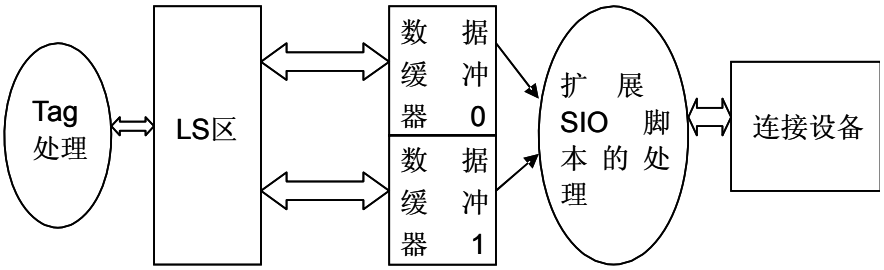


通过扩展串口脚本协议，生成与温控器通讯的驱动。与 PLC 通讯方式一样，GP 发送请求命令，然后处理温控器返回的响应数据（参考图表 1）

在 GP 系列中，通讯脚本是和画面处理分开执行的(参考图表 2)



图表 1 命令流程图



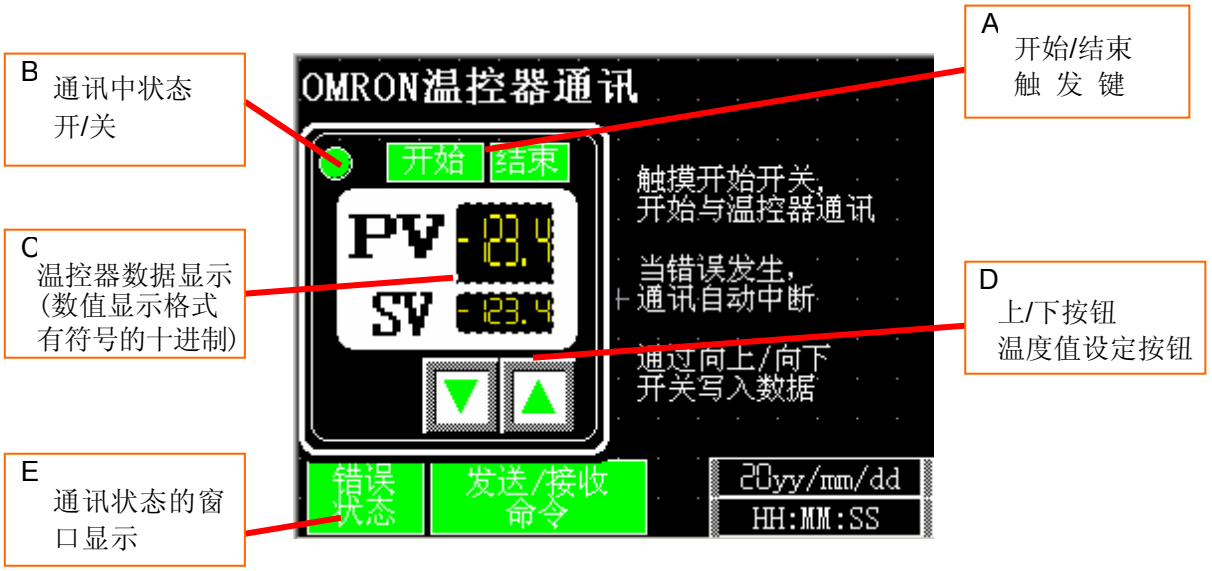
图表 2 扩展 SIO 脚本通讯略图

（注意）本示例在 GP(GLC)2000 系列上运行。

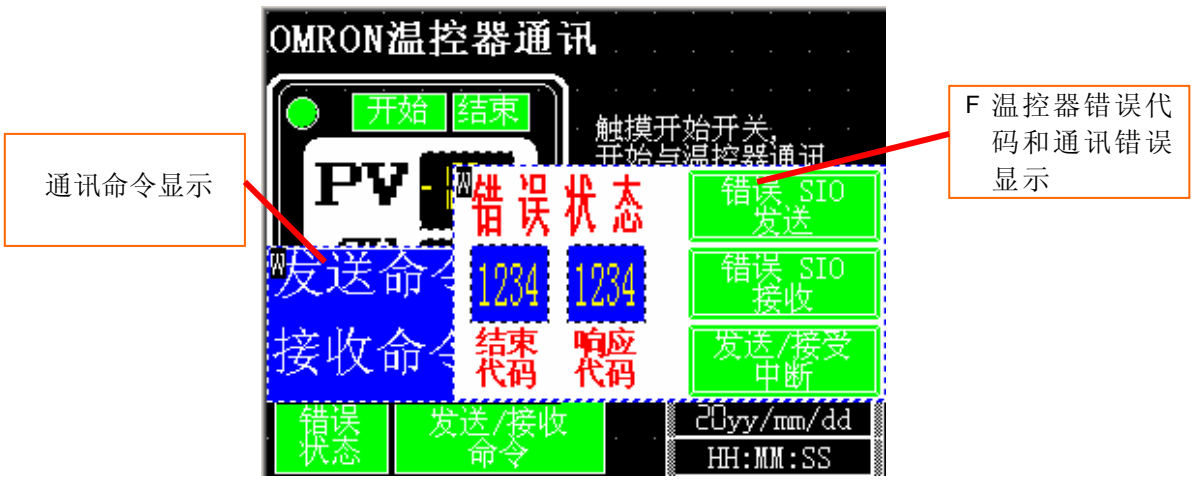
在触摸屏上设定数值时可能需 要时间。（→6. 注意事项）

2. 画面建立

〈标准画面〉



<窗口显示画面>



B1001　温控器画面

A：开始/停止按键 (位开关)

按下开始键 ，和温控器开始通讯。

按下停止键 ，和温控器终止通讯。

B：通讯状态显示灯（灯部件）

每次通讯，灯会变亮/灭。

C：显示温控器的数据(数据显示部件)

显示读取数据(当前值 ，设定值)。

D：上/下开关设置温控器数值（字开关）

可以在触摸屏上直接设置数据值。

E：窗口显示开关(位开关)

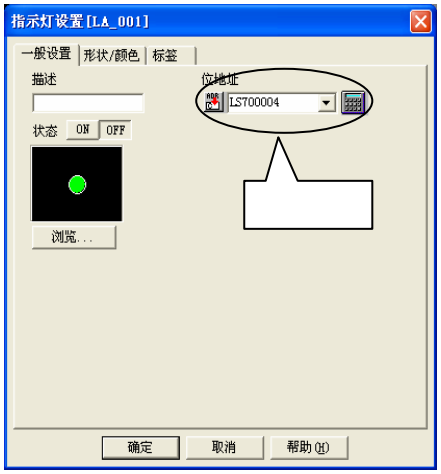
按下该开关，弹出一个窗口。

F：窗口显示(窗口显示部件)

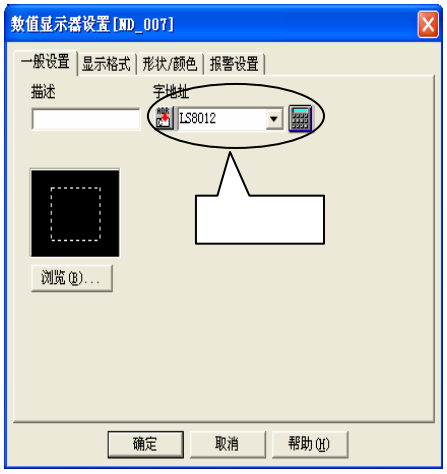
调用窗口并显示。

A：位开关

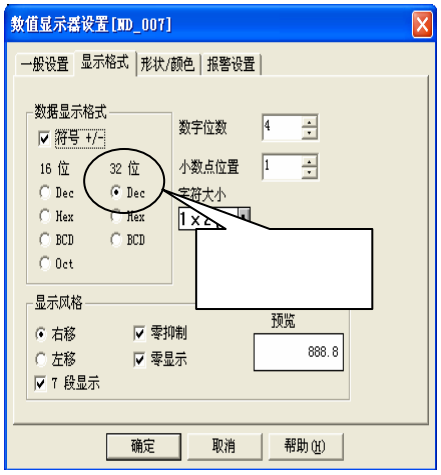
B：指示灯



C：数值显示

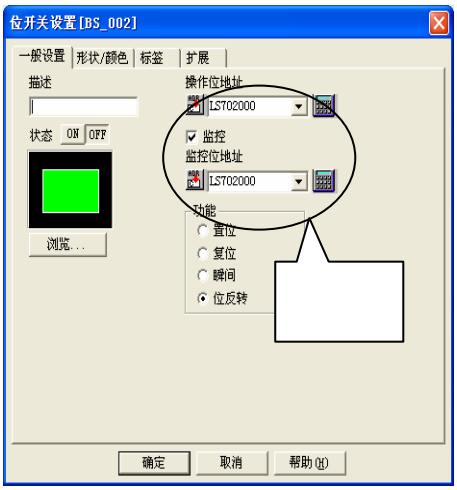


C：数值显示



D：字开关

E：位开关



F：窗口显示

3. 地址和D脚本说明

本示例运行需要使用扩展脚本，脚本说明如下。（→建立和编辑D脚本）
LS区中用到的地址如下。（→关于LS区）
(注意：通讯脚本中使用的地址只是LS 区地址)

使用的地址	详细说明
字地址	
LS7000	开始/停止
LS7001	通讯错误
LS7020	通讯错误窗口显示
LS7021	通讯命令窗口显示
LS7050	写入数据的中断标记
LS7100	BCC 操作的字节数
LS7101	BCC 操作结果存储地址
LS7110~LS7149	BCC 操作字符存储地址
LS7200	发送字节数
LS7210~LS7249	发送命令字符存储地址
LS7300	发送命令存储地址(S-tag 显示)
LS7400	接收字节数
LS7410~7449	接收命令字符存储地址
LS7500	接收命令存储地址(S-tag 显示)
LS8000~LS8002	接收数据存储地址
LS8012~LS8013	温控器当前值显示地址
LS8014~LS8015	温控器设置值显示地址
LS8100	温控器设置值写入地址
LS8110~LS8117	写入的数值经数据/字符转换后的存储地址

使用的地址	详细
位地址	
LS700000	开始位
LS700001	停止位
LS700004	通讯状态确认位
LS700100	发送错误(电缆没有连接)
LS700101	接收错误
LS700104	通讯超时
LS705000	数据写入中断位
临时地址	
t0030	发送完成标志
t0032	接收超时计数
t0033	接收数据偏移
t0035	BCC 操作循环次数
t0036	BCC 操作循环偏移
t0037	发送命令循环次数
t0038	发送命令循环偏移
t0039	发送命令循环偏移
t0040	接收字节数
t0045	接收命令循环次数
t0046	接收命令循环偏移
t0047	接收命令循环偏移

临时地址只能用于D脚本内部操作

扩展串口协议脚本的说明

```
// 结束
```

```
[c:EXT_SIO_CTRL00]=1                    // 清除发送缓冲
[c:EXT_SIO_CTRL01]=1                    // 清除接收缓冲
[c:EXT_SIO_CTRL02]=1                    // 错误清除
```

```
// 此脚本用于发送/接收从温控器读到的当前值

[t:0030]=0                              // 发送完成标志清除
Call F2_R_SV1_S                         // 调用发送功能
[t:0033]=4                              // 设置接收地址偏移
if([t:0030]==1){                        // 命令是否发送
  Call F3_RCV                           // 调用接收功能
}endif
```

```
// 此脚本用于发送/接收从温控器读到的设定值

[t:0030]=0                              // 发送完成标志清除
Call F2_R_SV1_S                         // 调用发送功能
[t:0033]=4                              // 设置接收数据的地址偏移值
if([t:0030]==1){                        // 命令是否已发送
Call F3_RCV                             // 调用接收功能
}endif
```

```
// 此脚本用于发送/接收写入到温控器的当前值
[t:0030]=0                                          // 发送完成标志清除
Call F2_W_SV1_S                                     // 调用发送功能
[t:0033]=0                                          // 设置接收数据地址偏移
if([t:0030]==1){                                    // 是否发送命令
  Call F3_RCV                                       // 调用接收功能
}endif
if(([w:LS7010]==0)and([w:LS7011]==0)){    // 温控器没有通讯错误时
  clear([b:LS705000])                               // 释放写入保留区
}endif
```

```
// BCC 操作脚本
_strlen([w:LS7100], databuf0)                       // 获取databuf0的数据长度
[t:0035]=[w:LS7100]-1                               // 减少开始位
_dlcopy([w:LS7110], databuf0, 1, [t:0035])   // 取出数据
[t:0036]=0                                          // 循环偏移
[w:LS7101]=0                                        // BCC操作初始化
loop([t:0035]){                                     // 操作时间循环
  [w:LS7101]=[w:LS7101]^[w:LS7110]#[t:0036]   // XOR操作
  [t:0036]=[t:0036]+1                               // 偏移相加
}endloop
_strset(databuf1, "")                               // databuf1初始化
_ldcopy(databuf1, [w:LS7101], 1)                    // 操作结果到databuf1
```

```
// 创建发送命令读温控器当前值
loop(){
  if([s:EXT_SIO_STAT00]==1){                    // 发送缓冲是否准备好
    break                                       // 跳出循环
  }endif
}endloop

// 生成温控器通讯命令
_strset(databuf0, "")                           // databuf0初始化
_strset(databuf0, 0x02)                         // 设置命令开始
_strcat(databuf0, "01")                         // 设置单元号码
_strcat(databuf0, "000")
_strcat(databuf0, "0101C00000000001")           // 连接到读命令
_strcat(databuf0, 0x03)                         // 连接到末尾

Call F1_BCC                                     // 调用BCC操作功能
_strcat(databuf0, databuf1)                     // 连接BCC

_strlen([w:LS7200], databuf0)                   // 获得发送字节数
memset([w:LS7210],0,50)                         // 发送命令初始化
_dlcopy([w:LS7210], databuf0, 0, [w:LS7200])    // 确认发送命令
Call Send_Byte2Word                             // 接受命令分段显示（调试）
IO_WRITE_EX([p:EXT_SIO], databuf0, [w:LS7200])  // 输出到扩展串口
[t:0030]=1                                      // 发送完成标志为ON
```

```
// 创建发送命令读温控器设定值
loop(){
  if([s:EXT_SIO_STAT00]==1){                    // 发送缓冲是否准备好
    break                                       // 跳出循环
  }endif
}endloop

// 设置温控器的通讯命令
_strset(databuf0, "")                           // databuf0初始化
_strset(databuf0, 0x02)                         // 设置命令开始
_strcat(databuf0, "01")                         // 设置号码
_strcat(databuf0, "000")
_strcat(databuf0, "0101C10003000001")           // 连接读命令
_strcat(databuf0, 0x03)                         // 连接末尾

Call F1_BCC                                     // 调用BCC操作功能
_strcat(databuf0, databuf1)                     // 连接BCC

_strlen([w:LS7200], databuf0)                   // 获得发送字节数
memset([w:LS7210],0,50)                         // 发送命令初始化
_dlcopy([w:LS7210], databuf0, 0, [w:LS7200])    // 确认发送命令
Call Send_Byte2Word                             // 接受命令分段显示（调试）
IO_WRITE_EX([p:EXT_SIO], databuf0, [w:LS7200])  // 输出到扩展串口
[t:0030]=1                                      // 发送完成标志为ON
```

```
// 创建发送命令写温控器的设定值
loop(){
  if([s:EXT_SIO_STAT00]==1){              // 发送缓冲是否准备好
    break                                 //  跳出循环
  }endif
}endloop


// 创建和温控器的通讯命令
_strset(databuf0, "")                     // databuf0初始化
_strset(databuf0, 0x02)                   // 设置开始命令
_strcat(databuf0, "01")                   // 设置号码
_strcat(databuf0, "000")                  // 设置0
_strcat(databuf0, "0102C10003000001")     // 连接写入命令

Call F2_W_Bin2ASC                         // 创建写入数据
_strset(databuf1, "")                     // databuf1初始化
_ldcopy(databuf1, [w:LS8114], 4)          // 在缓冲中存储写入数据
_strcat(databuf0, databuf1)               // 连接数据

_strset(databuf1, "")                     // databuf1初始化
_ldcopy(databuf1, [w:LS8110], 4)          // 在缓冲中存储写入数据
_strcat(databuf0, databuf1)               // 连接数据

_strcat(databuf0, 0x03)                   // 连接开始

Call F1_BCC                               // BCC操作功能处理
_strcat(databuf0, databuf1)               // 连接BCC

_strlen([w:LS7200], databuf0)             // 获得发送字节数
memset([w:LS7210],0,50)                   // 发送命令初始化
_dlcopy([w:LS7210], databuf0, 0, [w:LS7200])   // 确认发送命令
Call Send_Byte2Word                       // 接受命令分段显示（调试）
IO_WRITE_EX([p:EXT_SIO], databuf0, [w:LS7200])   // 输出到扩展串口
[t:0030]=1                                // 发送完成标志为ON
```

脚本函数：F2_W_Bin2ASC

```
// 此脚本用于 Bin 到 ASCII  字符转换

//处理第 8 位
[w:LS8120]=([w:LS8100]&0xF000)>>12              // 位标记和位移动
if(([w:LS8120]>=0x00) and ([w:LS8120]<=0x09)){  // 数值举例
  [w:LS8110]=[w:LS8120]+0x30                    // 加十六进制30
}endif
if(([w:LS8120]>=0xA) and ([w:LS8120]<=0xF)){    // 字符举例
  [w:LS8110]=[w:LS8120]+0x37                    // 加十六进制37
}endif
//处理第 7 位
[w:LS8121]=([w:LS8100]&0x0F00)>>8
if(([w:LS8121]>=0x00) and ([w:LS8121]<=0x09)){
  [w:LS8111]=[w:LS8121]+0x30
}endif
if(([w:LS8121]>=0xA) and ([w:LS8121]<=0xF)){
  [w:LS8111]=[w:LS8121]+0x37
}endif
//处理第 6 位
[w:LS8122]=([w:LS8100]&0x00F0)>>4
if(([w:LS8122]>=0x00) and ([w:LS8122]<=0x09)){
  [w:LS8112]=[w:LS8122]+0x30
}endif
if(([w:LS8122]>=0xA) and ([w:LS8122]<=0xF)){
  [w:LS8112]=[w:LS8122]+0x37
}endif
//处理第 5 位
[w:LS8123]=([w:LS8100]&0x000F)
if(([w:LS8123]>=0x00) and ([w:LS8123]<=0x09)){
  [w:LS8113]=[w:LS8123]+0x30
}endif
if(([w:LS8123]>=0xA) and ([w:LS8123]<=0xF)){
  [w:LS8113]=[w:LS8123]+0x37
}endif
```

脚本函数：F2_W_Bin2ASC（继续）

```
//处理第 4 位
[w:LS8124]=([w:LS8101]&0xF000)>>12                    // 位标记和位移动
if(([w:LS8124]>=0x00) and ([w:LS8124]<=0x09)){         // 数值举例
     [w:LS8114]=[w:LS8124]+0x30                       // 加十六进制30
 }endif
if(([w:LS8124]>=0xA) and ([w:LS8124]<=0xF)){          // 字符举例
     [w:LS8114]=[w:LS8124]+0x37                       // 加十六进制37
}endif
//处理第 3 位
 [w:LS8125]=([w:LS8101]&0x0F00)>>8
if(([w:LS8125]>=0x00) and ([w:LS8125]<=0x09)){
   [w:LS8115]=[w:LS8125]+0x30
}endif
if(([w:LS8125]>=0xA) and ([w:LS8125]<=0xF)){
   [w:LS8115]=[w:LS8125]+0x37
}endif
//处理第 2 位
 [w:LS8126]=([w:LS8101]&0x00F0)>>4
if(([w:LS8126]>=0x00) and ([w:LS8126]<=0x09)){
   [w:LS8116]=[w:LS8126]+0x30
}endif
if(([w:LS8126]>=0xA) and ([w:LS8126]<=0xF)){
   [w:LS8116]=[w:LS8126]+0x37
}endif
//处理第 1 位
 [w:LS8127]=([w:LS8101]&0x000F)
if(([w:LS8127]>=0x00) and ([w:LS8127]<=0x09)){
   [w:LS8117]=[w:LS8127]+0x30
}endif
if(([w:LS8127]>=0xA) and ([w:LS8127]<=0xF)){
   [w:LS8117]=[w:LS8127]+0x37
}endif
```

脚本函数：Send_Byte2Word

```
// 逐步发送字节数据到字地址 LS7300

[t:0037]=20 // 设定循环次数
[t:0038]=0  // 初始化
[t:0039]=0  // 初始化

loop([t:0037]){
    [w:LS7350]#[t:0039]=[w:LS7210]#[t:0038]<<8          // 偶字节上移
    [t:0038]=[t:0038]+1                                 // 增加偏移值
    [w:LS7300]#[t:0039]=[w:LS7350]#[t:0039]|[w:LS7210]#[t:0038]
                                                        // 高低位加
    [t:0038]=[t:0038]+1                                 // 增加偏移值
    [t:0039]=[t:0039]+1                                 // 增加偏移值
}endloop
```

```
// 此脚本用于反向接收发送命令

[t:0032]=10000                          // 接收超时时间设定(100ms×100)
loop(){
  if([s:EXT_SIO_STAT02]==1){            // 接收数据是否存在
    if([s:EXT_SIO_STAT03]==1){          // 接收错误是否存在
      [c:EXT_SIO_CTRL01]=1              // 接收缓冲清除
      return                            // 处理过程结束
    }else{
        _wait(3)                        // 等待接数据
       _strset(databuf2, "")            // databuf2初始化
        [t:0040]=[r:EXT_SIO_RECV]
        IO_READ_EX([p:EXT_SIO], databuf2, [t:0040])     // 获得接收数据
        _strlen([w:LS7400], databuf2)                   // 接收字节数
        memset([w:LS7410],0,20)                         // 接收地址区初始化
        _dlcopy([w:LS7410], databuf2, 0, [w:LS7400])    // 确认接收命令
        Call RCV_Byte2Word                              // 调试
// 温控器结束代码处理
        _strset(databuf3, "")                           // databuf3初始化
        _strmid(databuf3, databuf2, 5, 2)               // 从数据缓冲取出数据
        _hexasc2bin([w:LS7010], databuf3)                   // 结束代码
// 温控器响应代码处理
        _strset(databuf3, "")                           // databuf3初始化
        _strmid(databuf3, databuf2, 11, 4)                  // 从数据缓冲取出数
        据
        _hexasc2bin([w:LS7011], databuf3)                   // 响应代码
// 温控器数据处理
        _strset(databuf3, "")                           // databuf3初始化
        _strmid(databuf3, databuf2, 15, 8)                  // 从数据缓冲取出数
        据
        Call F4_DataCopy                                // 复制数据到 LS
        return
    }endif
  }endif
  if([t:0032]==0){                      // 接收是否超时
    [c:EXT_SIO_CTRL01]=1                // 接收缓冲清除
  set([b:LS700104])                     // 接收超时标志
  return                                // 功能处理结束
}endif
  [t:0032]=[t:0032]-100                 // 超时时间减少
  _wait(1)                              // 100ms等待
```

```
// 逐步接收字节数据到字地址 LS7500

[t:0055]=20              // 循环次数设定
[t:0056]=0               // 初始化
[t:0057]=0              // 初始化

loop([t:0055]){
      [w:LS7550]#[t:0057]=[w:LS7410]#[t:0056]<<8          // 偶字节到更高列
      [t:0056]=[t:0056]+1                                 // 偏移量相加
      [w:LS7500]#[t:0057]=[w:LS7550]#[t:0057]|[w:LS7410]#[t:0056]
                                                          // 高位和低位相加
      [t:0056]=[t:0056]+1                                 // 偏移量相加
      [t:0057]=[t:0057]+1                                 // 偏移量相加
}endloop
```

```
_hexasc2bin([w:LS8000], databuf3)        // 文字和数字转换
[w:LS8010]#[t:0033]=[w:LS8000]   // 逐步将当前值写入地址
[w:LS8100]=[w:LS8014]            // 复制设定值到写入地址
```

4. GP、PLC 和画面编辑软件版本

本例使用的 GP、PLC、画面编辑软件的版本号说明如下。请注意，如果您的 PLC 和触摸屏的型号与本例不同,需要对工程画面和设置作必要的调整。(→6. 使用注意)

使用的GP和PLC

GP：GP2300

PLC：Modicon Modbus（SLAVE）

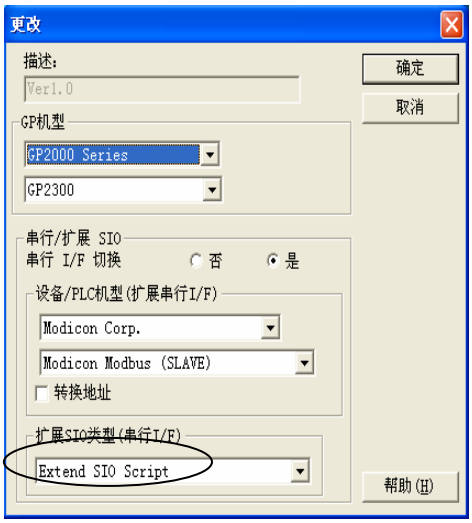
（协议：Modicon Modbus(SLAVE)）

画面编辑软件版本： GP-PRO/PBIII C-Package03（V7.23中文版）

5. 画面复制

您可以将本例直接复制到你的工程文件中。请注意在复制时，地址和画面号不能与你工程中的已有的地址和画面号重叠。（→确认地址）

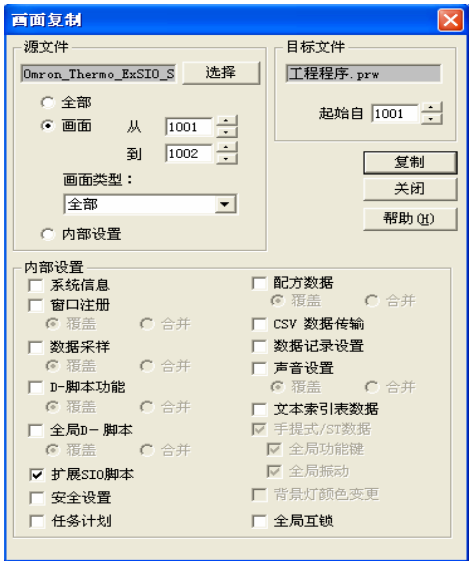
如果您使用的触摸屏型号与本例不同，需要对画面位置和设置进行适当的调整。



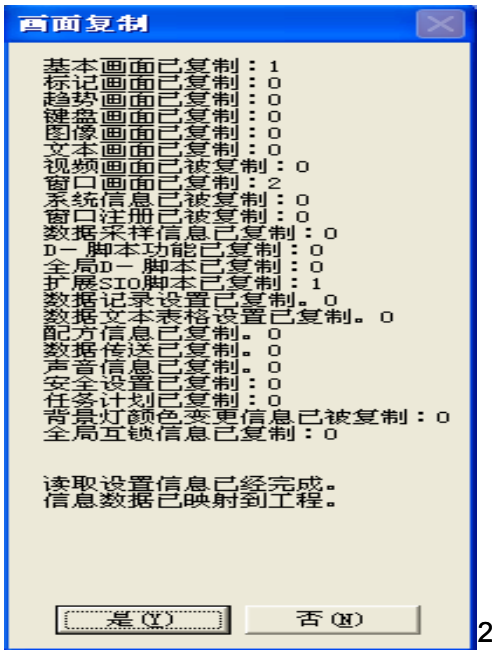
在您的工程文件的“扩展SIO型号”下选择Extend SIO Script。
如果“串口 I/F 交换”中选择“交换”则用COM1 连接温控器。

在您工程文件里的工程管理器[应用]下拉菜单中，单击[画面复制]命令。

选择本例程序的文件名并单击 [打开]。



在“画面范围”中设置开始画面 1001，结束画面 1002，画面类型为“ALL”,指定拷贝到您的工程文件后的画面编号 (在本例使用 1001)。
在系统信息中选择“扩展 SIO 脚本信息”。
设置完成,单击[复制]按钮。拷贝内容添加到您的工程文件中。



6. 注意事项

如果您的触摸屏型号与本例中使用的触摸屏型号不一致（比本例屏的尺寸大），画面设置是不同的，因此需要调整画面。可以使用本例的型号有：GP2000 系列和 GLC 系列，采用的软件是 6.23 或 6.23 以上版本。

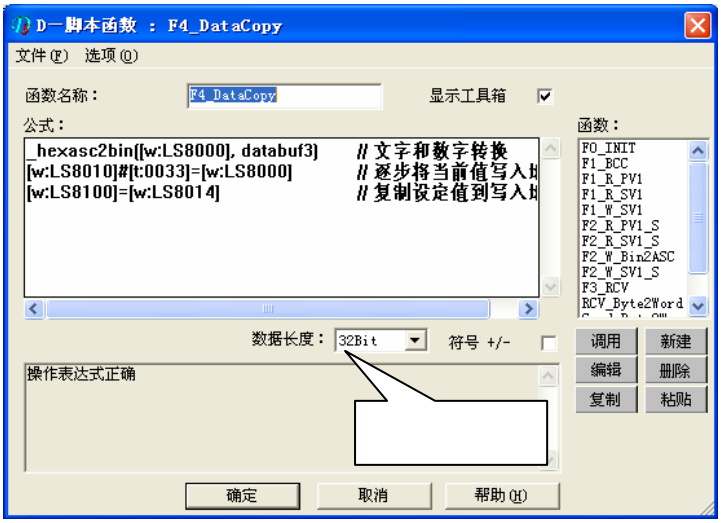
在屏上写入设定值时需要时间。

如果在 “串口 I/F 交换”
中做了与本例相同的设置，那么在COM1连接温控器，在COM2连接PLC。

本例中，您可以使用 16 位的二进制数据（0～65535）。

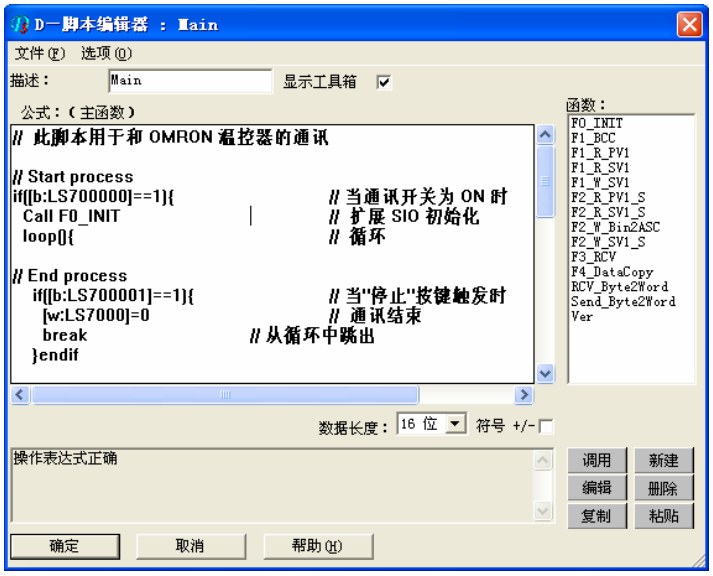
如果使用了其它格式或负数数据，您需要在 D 脚本中进行相应的设置。

示例：使用 16 位二进制负数数据



〈附录〉

在画面编辑屏幕的菜单栏里，点击[特殊]下拉菜单中的[扩展的 SIO 脚本]。



弹出D脚本编辑器窗口。通过工具栏进行地址输入和SIO操作等。 单击[确定]进行保存。

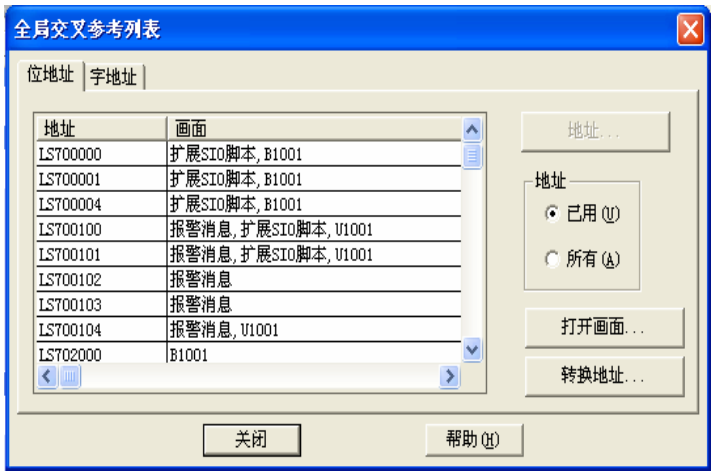
LS 区是 GP 的一个内存区域,用来控制 GP 操作。 LS 区的构成如下表：

用户区是 GP 内部使用的一块地址区域，不能分配给 PLC。这个区域用作 GP 对部件和 Tags 的内部处理，不能被 PLC 控制。

确认地址

以下部分介绍当前工程中哪些地址已被部件、D 脚本等使用，画面号也可通过同样的方法确定。

在您工程的工程管理器[应用]菜单中选择[全局交叉参考]-[列表]命令



然后会弹出“全局交叉参考列表”窗口，列表中将显示已使用的地址和画面号。双击某个地址或点击窗口右侧的[打开画面]，输入您要查找的地址，然后使用该地址的画面将自动打开。