

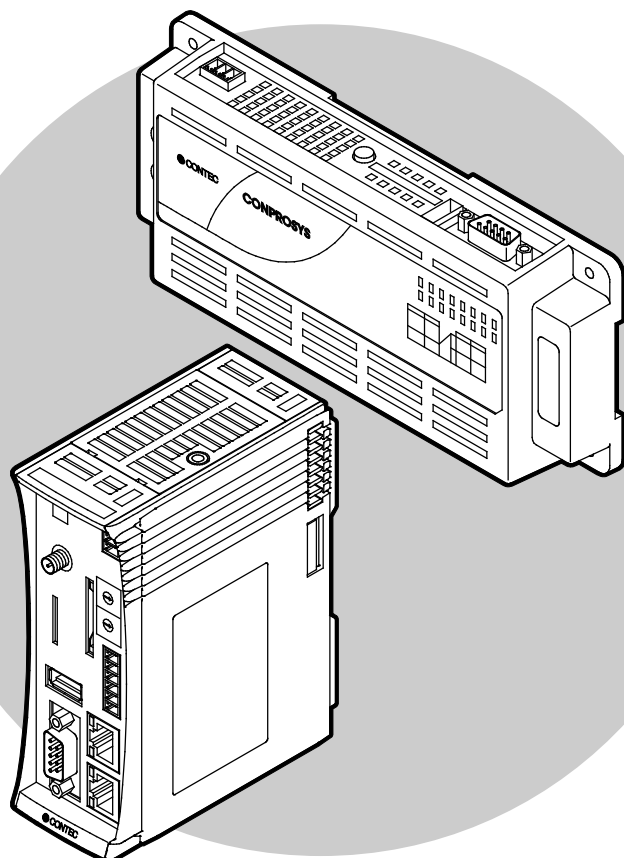
参考使用说明

PAC 系列

紧凑型/堆栈型

目录

介绍	6
安全使用须知	14
电脑网络设定	18
关于CONPROSYS WEB Setting.....	24
安装CODESYS	45
基本编程	52
设置通信方法	68
监控编辑	136
问题	151
规格	154
附录	159



目录

介绍 6

- 1. 产品概要7
 - 1. PAC系列的功能7
- 2. 使用说明构成9
- 3. 产品设定流程10
- 4. 相关使用说明介绍.....11
- 5. 在线帮助12
- 6. 固件确认13

安全使用须知 14

- 1. 注意符号说明15
- 2. 安装注意事项16
- 3. 安全相关注意事项.....17
 - 1. 安全隐患.....17
 - 2. 安全对策事例.....17

电脑网络设定 18

- 1. 连接电脑19
- 2. 电脑网络设定20
 - 1. 电脑网络设定顺序 (Windows 10时)20
- 3. 确认是否已通信.....23

关于 CONPROSYS WEB Setting 24

- 1. CONPROSYS WEB Setting概要.....25
 - 1. 支持的Web浏览器.....25
 - 2. CONPROSYS WEB Setting的启动25
 - 3. CONPROSYS WEB Setting的基本操作26
- 2. 菜单选项功能一览.....27
- 3. 功能的详细说明.....28
 - 1. 网络设置.....28
 - 2. 用户/密码设置30
 - 3. 时间设定.....31
 - 4. 数据发送设定32
 - 5. 设置备份.....33
 - 6. 设置还原.....34
 - 7. 设置保存.....35
 - 8. 设置初始化.....35
 - 9. 系统信息.....36
 - 10. 重新启动/关闭39
 - 11. PLC程序保存40
 - 12. 固件更新41
 - 13. Editor42

目录

14. Viewer	43
15. 保存页面	44

安装 CODESYS..... 45

1. 安装CODESYS	46
2. 安装CODESYS包	47
3. 连接电脑和CONPROSYS	48
4. 固件版本升级	50

基本编程..... 52

1. CODESYS各部分名称	53
1. 设备窗口，设备配置窗口	53
2. ST编辑器窗口	53
2. ST语言格式和描述方法	54
3. 基本编程顺序	56
1. 新建项目的生成	56
2. 从CODESYS连接到控制器.....	57
3. 添加 I / O 模块	58
4. I / O 变量的定义.....	59
5. 程序创建和构建	60
6. 程序的下载和执行	61
7. PLC程序的ROM保存	62
8. 删除ROM保存的PLC程序	62
4. 产品接口定义	63
1. 参数	63
2. I / O映射	64
3. 串口	67
4. OPC UA符号设置	67

设置通信方法..... 68

1. 控制器与PC的串行通信	70
1. 串行通信的准备	70
2. 串行通信库的设置	70
3. 创建和执行发送程序	71
4. 接收程序的制作和执行	73
2. 用作EtherCAT主站.....	76
1. 设备的准备	76
2. 添加从设备	76
3. 从设备设置.....	78
4. 创建和构建程序	79
5. 下载并执行程序	80
3. 用作Modbus TCP Master	81
1. 设备的准备	81
2. 添加从设备	81
3. Modbus TCP Master的设置	84

目录

4. 创建和构建程序	86
5. 下载并执行程序	87
4. 用作Modbus TCP Slave	88
1. 设备的准备	88
2. 进行从属设置	89
3. 创建和构建程序	93
4. 下载并执行程序	94
5. 设置OPC UA Server	97
1. 设备的准备	97
2. CODESYS的设定	97
3. OPC UA Client的设定	99
6. 计数器输入	101
1. 基本操作顺序	101
2. 事件操作顺序	102
3. 计数值设定操作顺序	103
4. 示例程序	104
7. 访问文件	109
1. 设置文件访问库	109
2. 功能一览	109
3. 数据类型	110
4. 功能说明	111
5. 示例程序	117
8. 将数据发送到云服务	125
1. 数据发送库的设置	125
2. 功能一览	125
3. 数据类型	126
4. 功能说明	128
5. 示例程序	132

监控编辑..... 136

1. 使用CONPROSYS HMI	137
1. CODESYS的设定	137
2. 创建和构建程序	137
3. 使用HMI Editor创建画面	139
4. HMI Viewer的执行	142
5. 可与IEC数据类型使用的部件	143
2. CONPROSYS HMI的概要	144
1. 可以使用CONPROSYS HMI做什么	144
2. 操作区域	145
3. 创建监控画面	145
4. 创建监控画面的基本操作	146
3. 显示部件一览	150

问题..... 151

1. 发生故障时	152
1. 整体	152

目录

规格 154

- 1. 硬件规格 155
 - 1. CPS-PC341EC-1-9201的规格 155
 - 2. CPS-PC341MB-ADSC1-9201的规格 155
 - 3. CPS-PCS341EC-DS1-1201的规格 156
 - 4. CPS-PCS341MB-DS1-1201的规格 156
- 2. CONPROSYS HMI 规格 157

附录 159

- 1. 发送数据的格式 160

介绍

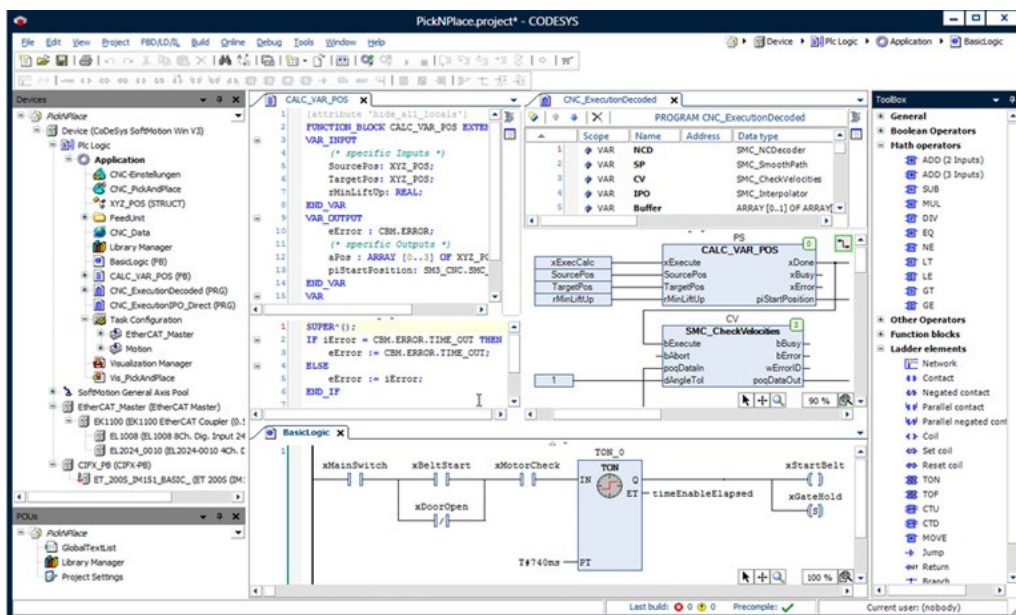
在了解本产品的产品概要，该使用说明的整体构成以及本产品相关的各种使用说明等内容之前，请先阅读本章内容。

1. 产品概要

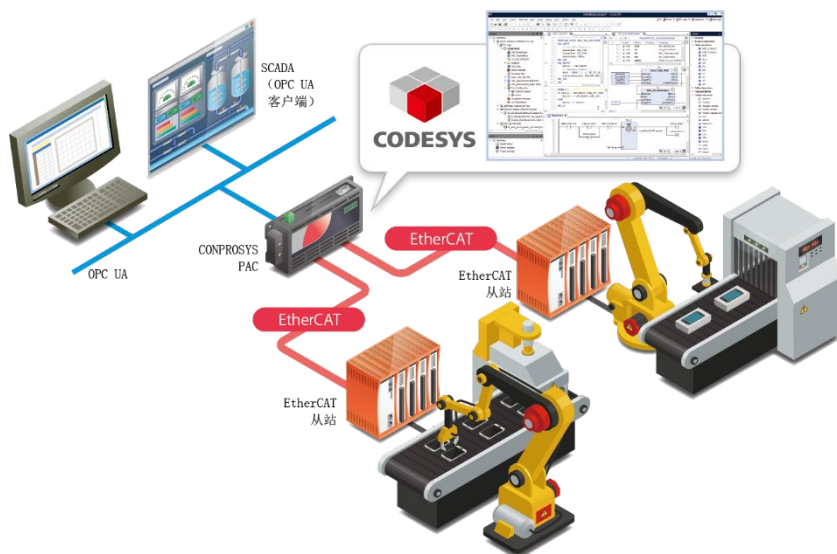
PAC系列搭载了软件PLC“CODESYS”功能。

1. PAC系列的功能

◆ 国际标准 IEC 61131-3 标准 “CODESYS” 编程对应



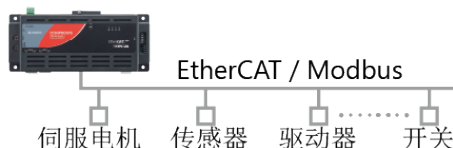
- 集成所有PLC编程，现场总线设置等的开发环境“CODESYS” 免费提供。
- 依据国际标准IEC 61131-3标准“CODESYS”的开放通信，可以选择不依赖特定通信标准的I / O和驱动器等外围设备。
- 对应包括标准ST为首的 LD、FB、SFC、IL、CFC共计6种语言。也支持IEC 61131-3第三版定义对象的编程。



◆ 可直接用变量使用现场总线I / O.

支持EtherCAT / Modbus现场总线

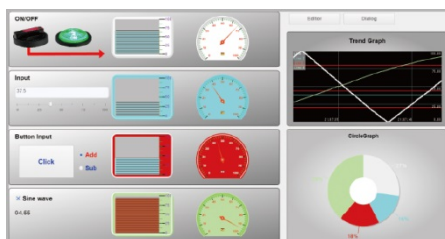
- 内置开放现场网络EtherCAT/Modbus的主机功能。
与内置I / O和堆栈I / O一样，现场总线I / O可以直接分配给CODESYS集成开发环境中的变量。



◆ 使用Web浏览器轻松监控信息

Web监控功能

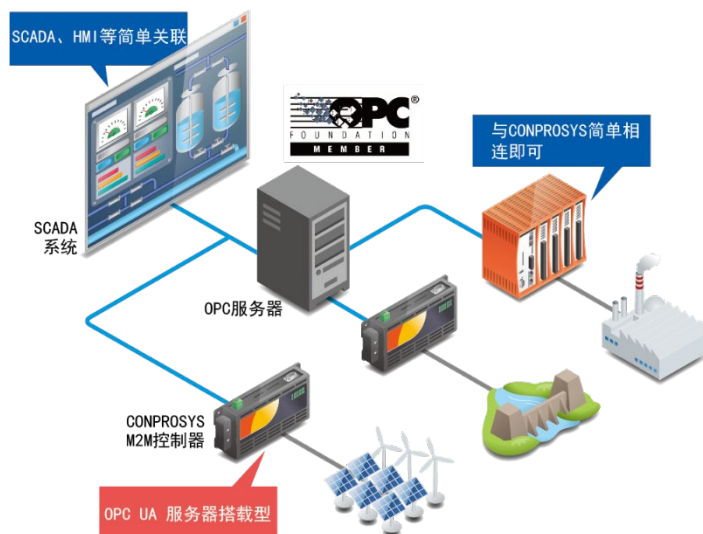
- 控制器主机内置CONPROSYS HMI。可以自由创建监视器屏幕。
- 即使没有单独的服务器，也可以使用Web服务器功能轻松显示设备信息。



◆ 与SCADA / MES · ERP系统进行稳定的数据交换

OPC UA服务器功能（支持SCADA / MES / ERP链接）

控制器主机内置OPC UA服务器功能。可与SCADA软件和MES · ERP系统进行安全稳定的数据交换。



2. 使用说明构成

使用说明由以下构成。

使用手册章	记载内容
介绍	介绍本产品相关的使用手册。 请根据需要参考相关的使用手册
安全使用须知	安全注意事项
电脑网络设定	电脑的设置方法 在使用本产品前需要对电脑的网络进行设置。 本章对设置方法进行说明。
关于CONPROSYS WEB Setting	CONPROSYS WEB Setting的说明 本章对CONPROSYS WEB Setting上的各种功能设置及操作方法进行说明。
安装CODESYS	CODESYS的详细使用方法 讲解使用codesys来连接、设定功能等详细用途。
基本编程	
设置通信方法	
监控编辑	
问题	设置完成却不运行或有疑问时的解决对策。
规格	讲解关于硬件规格和conprosys hmi的规格。
附录	记录各种数据格式的式样。

3. 产品设定流程

使用本产品的一般流程如下。



根据您使用本产品的环境和系统类型，使用的流程可能会有所不同。

4. 相关使用说明介绍

本产品相关的使用说明构成如下。

请与本书一起活用。

◆ 请务必阅读

名称	用途	内容	入手先
产品介绍(或商品介绍)	本产品拆封后请务必阅读。	对使用本产品前确认附件及注意事项进行说明。	製品に同梱(印刷物)
设置手册	请在安装本产品时阅读。	对安装所需要准备的东西以及连接, 设置方式进行说明。	 请从本公司网站下载(PDF)
参考手册(硬件篇)	使用本产品时请阅读。	对本产品的功能, 设定等相关硬件进行说明。	 请从本公司网站下载(PDF)
参考手册(软件篇)	设定『CONPROSYS WEB Setting』时请阅读。	对『CONPROSYS WEB Setting』的各种设定方法进行说明。	 请从本公司网站下载(PDF)

◆ 各种使用手册的下载

请点击以下URL下载各种使用手册。

下载

<https://www.contec.com/cn/download/>

5. 在线帮助

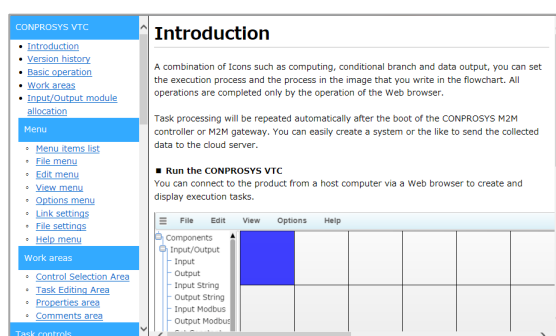
组建计算或控制等处理任务的『CONPROSYS VTC』或用于操作和编辑监视画面的『CONPROSYS HMI』的详细信息可通过在线帮助查看。

请在必要时利用。

◆ CONPROSYS VTC (Visual Task Control)

在线帮助

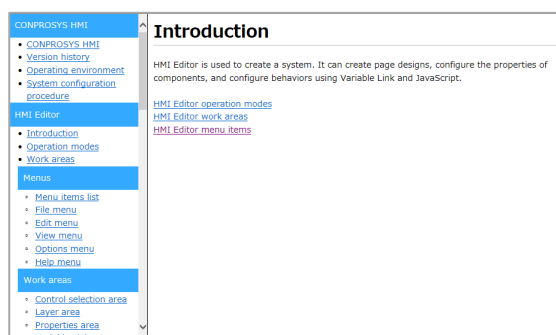
<http://data.conprosys.com/help/task/V1/zh/>



◆ CONPROSYS HMI (Human Machine Interface)

在线帮助

<http://data.conprosys.com/help/hmi/V1/zh/>



6. 固件确认

在使用前请在本公司网站确认固件的版本，请使用最新版本固件。
通过更新固件修复异常确保产品运行稳定。

下载

<https://www.contec.com/cn/download/>

※ 固件更新方法请参考『固件更新(P41)』。

安全使用须知

为确保安全使用本产品，对注意事项进行说明。请务必在使用本产品前阅读。

1. 注意符号说明

为避免人身事故或机器损坏，本手册通过下列符号提供安全相关信息。

请充分理解其内容后安全操作机器。

 危险	如果无视这个标志，采取错误的操作，极有可能造成人员的伤亡。
 警告	如果无视这个标志，采取错误的操作，可能会造成人员的伤亡。
 注意	如果无视这个标志，采取错误的操作，仅可能会造成人员的受伤及物品损坏。

2. 安装注意事项

注意

- 如本产品发生功能追加，品质提升等情况时，将不进行规格变更通知。即便持续使用本公司产品也请务必阅读并确认本公司网站上的使用手册内容。
- 请勿改造本产品。
对于改造过的产品，本公司概不负责。
- 对于以本产品的运用为由申请赔偿的，不管何种情况，本公司均不负责，敬请谅解。

3. 安全相关注意事项

考虑到链接网络时存在安全隐患，请参考安全对策事例，正确设定本机以及相关网络机器。

1. 安全隐患

- 通过来自外网的不良侵入导致系统的停止，数据缺损，信息被窃取，Malicious Software *1。
- 侵入后借助被侵入的机器，向外部网络发起攻击。（从受害者变为加害者）
- 与外部网络链接时所伴随的无意识信息泄露。
- 由不存在的原因或结果导致的经济损失，损害赔偿负担，信用丧失，机会损失等，可看作事故的二次被害。

※1: Malicious Software: 恶意程序。擅自进行用户不希望的操作的程序

2. 安全对策事例

- 更改初期密码。（密码设定方法请参考产品说明书/使用手册）
- 请设定强度高的密码。

密码包含半角英文小写，大写及数字等

- 定期更改密码。
- 停止（无效化）不需要的网络服务以及不需要的功能。
- 关于网络连接机器，限制网络接入端。※2
- 关于网络连接机器，限制网络解放端口。※2
- 使用专用网络或VPN*3等局域网构建网络。

※2: 设定方法请咨询网络机器的厂家。

※3: VPN(Virtual Private Network): 通过采用认证及加密的通信线路，防止第三者入侵的安全网络。

非法连接的手段及安全漏洞日新月异，并没有可以完美防范的手段。连接网络，在理解会经常伴随着危险的同时，强烈建议时常入手新的信息，实施安全对策。

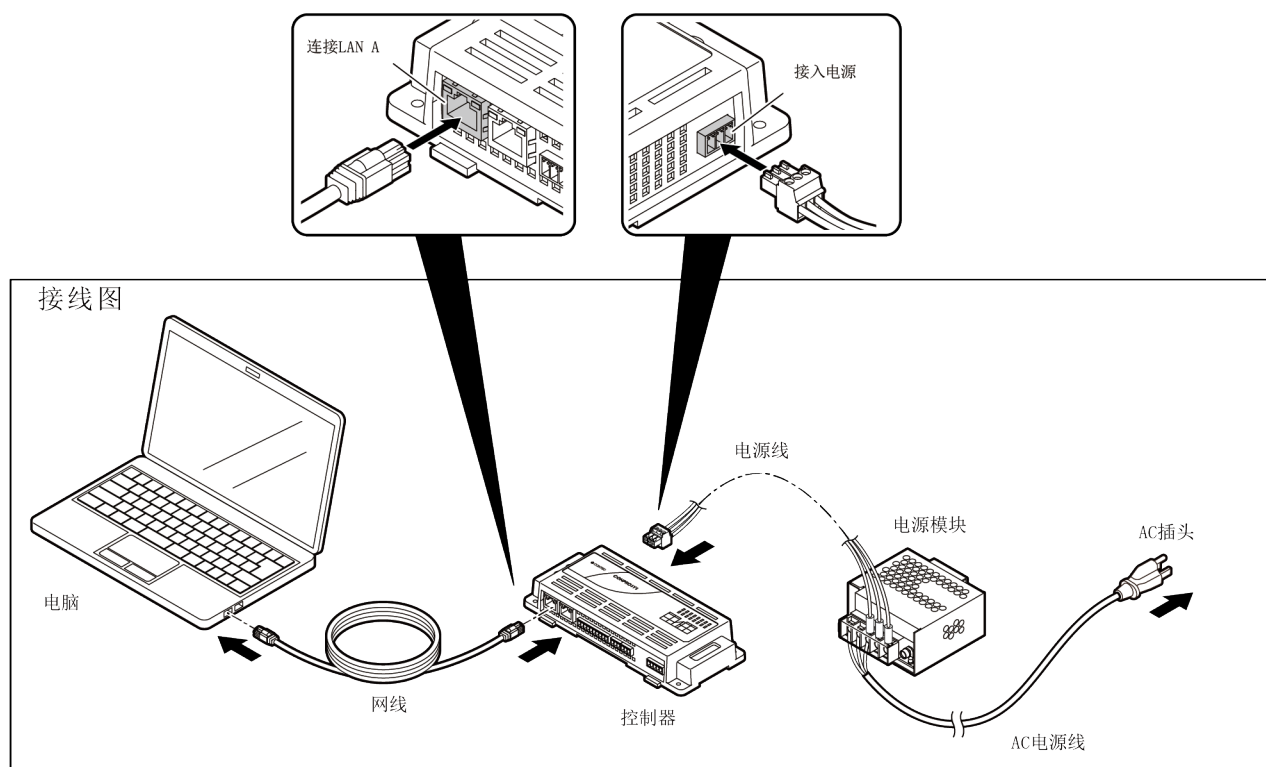
电脑网络设定

本章对本产品连接电脑的方式，网络设定以及通信确认方法进行说明。

1. 连接电脑

对本产品进行各种设置前，需要确立本产品和电脑之间已通信并对网络进行设置。
首先，将本产品与电脑连接。

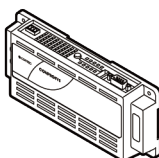
- 1 请参考下图将电脑，控制器以及电源连接起来。
※ 电源线的制作方法请参考『参考使用说明(硬件篇)』。



- 2 将电源模块与AC适配器连接，开启电脑。
※ 从电源模块与AN适配器连接开始到控制器启动结束需要花费一些时间。(大概1 - 2分钟)

2. 电脑网络设定

请参考一下设定顺序，将电脑的IP地址设置如下。

电脑端	控制器端
	
IP地址 10 . 1 . 1 . 200	IP地址 10 . 1 . 1 . 101 *
子网掩码 255 . 0 . 0 . 0	子网掩码 255 . 0 . 0 . 0 *

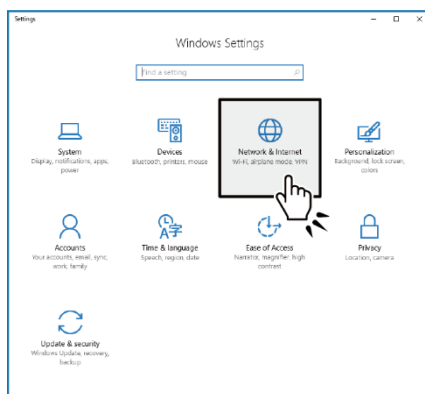
* 出厂设置

IP地址中加粗的部分（**200**和**101**）需要设定为不重复的数字。

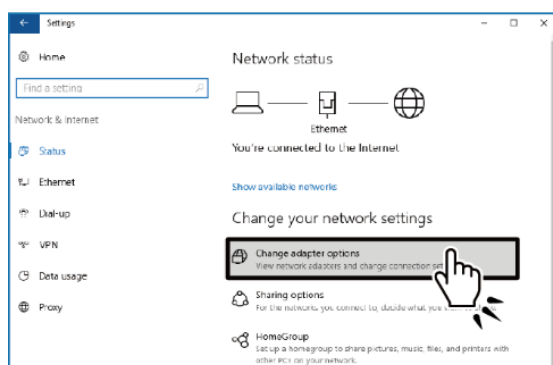
※ 如果电脑有设定代理，请更改为不使用。

1. 电脑网络设定顺序 (Windows 10时)

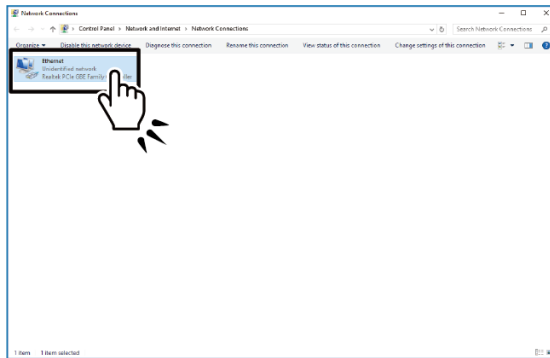
1 在 [Windows设置] 画面中点击 [网络和Internet]。



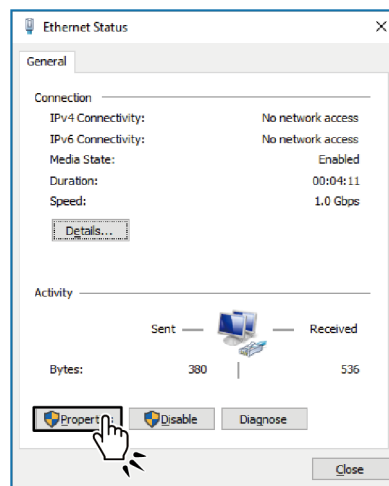
2 点击 [更改网络设置] 中的 [更改适配器选项]。



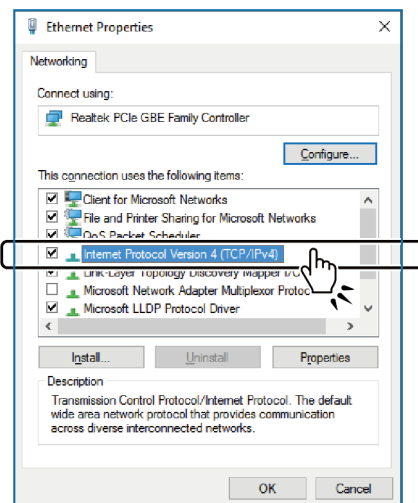
3 在显示的下一个屏幕上双击[本地连接]。



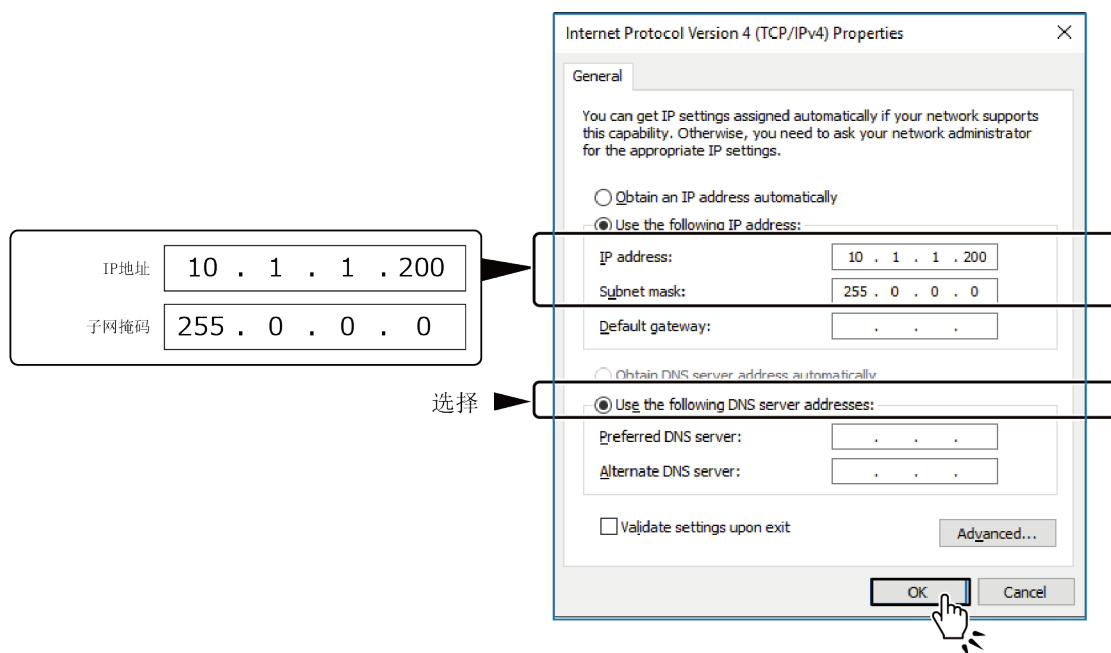
4 [点击[本地连接状态]对话框的[属性]。]



5 双击[本地连接属性]对话框中的[Internet协议版本4(TCP/IPv4)]。]



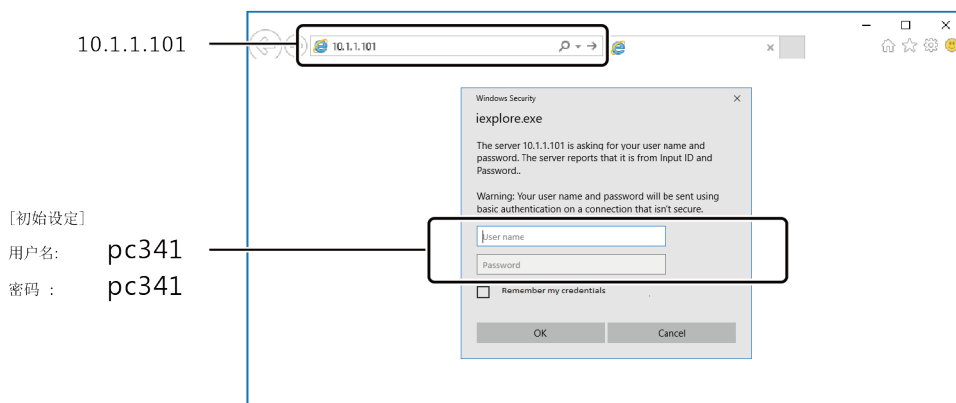
6 在 [Internet协议版本4(TCP/IPv4)属性] 对话框中设定如下IP地址及子网掩码。



7 点击 [确认] → [确认] → [关闭] 关闭对话框，网络设定完成。

3. 确认是否已通信

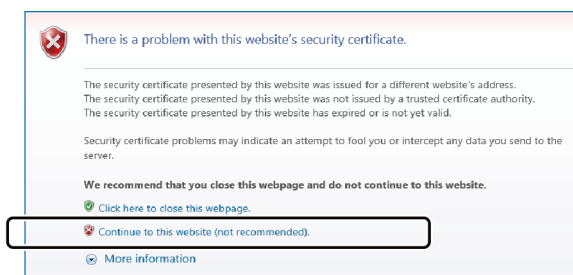
- 1 在电脑上启动Web浏览器，并在地址栏输入控制器的IP地址(10.1.1.101)然后回车。
当出现输入用户名和密码的画面时，输入用户名和密码并点击 [OK]。



※ 本产品支持的Web浏览器请参考『支持的Web浏览器 (P25)』。

※ 实际运用时，请在CONPROSYS WEB Setting [维护] - [用户/密码] 中修改用户名和密码。

※ 输入IP地址点击回车后，出现安全证书确认画面时请选择 [继续浏览]。



- 2 当出现CONPROSYS WEB Setting画面时，表示电脑和控制器已连通。

关于CONPROSYS WEB Setting

对本产品的系统及各种设置进行说明。

1. CONPROSYS WEB Setting概要

本产品的设定及各种设置在『CONPROSYS WEB Setting』中进行。

CONPROSYS WEB Setting可在Web浏览器上显示并简单设置。

1. 支持的Web浏览器

CONPROSYS WEB Setting支持如下Web浏览器。

支持的Web浏览器	支持版本
Microsoft Internet Explorer	Ver. 11以上
Google Chrome	Ver. 52以上
Mozilla Firefox	Ver. 55以上

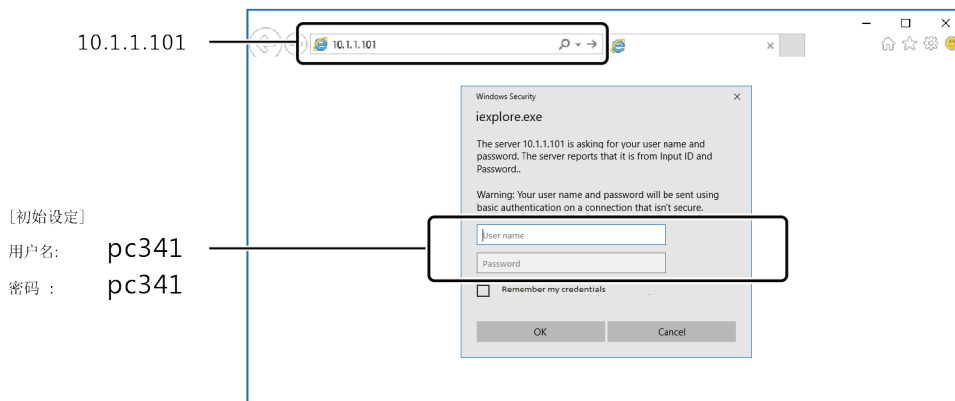
※ 非支持的Web浏览器可能会发生运行不良的情况。请务必使用以上支持的浏览器。

2. CONPROSYS WEB Setting的启动

在连接本产品的电脑上启动Web浏览器，并在地址栏输入IP地址(10. 1. 1. 101)后点击回车。

当出现输入用户名和密码的画面时，输入用户名和密码并点击 [OK]。

※ 本产品的连接方法请参考『安装说明手册』。

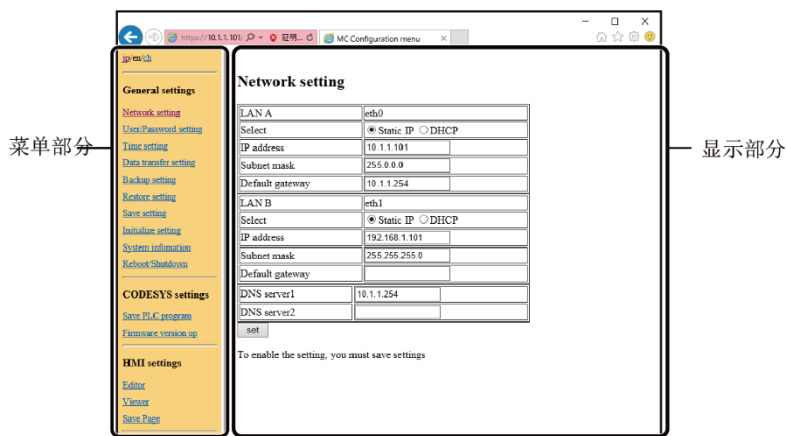


※ 输入IP地址点击回车后，出现安全证书确认画面时请选择 [继续浏览]。

3. CONPROSYS WEB Setting的基本操作

◆ CONPROSYS WEB Setting画面构成

CONPROSYS WEB Setting界面下，点击画面左侧『菜单部分』中的选项后，该选项的内容竟会显示在画面右侧的『显示、设置部分』并可进行设置。



2. 菜单选项功能一览

菜单选项名称	功能	详细信息
一般设定		
网络设置	进行IP地址等网络设置。	P28
用户/密码设置	设置用户/密码，使用Web浏览器登录本产品。	P30
时间设定	设置NTP服务器以获取日期和时间。	P31
数据传输设置	设置测量数据的目标服务器。	P32
设置备份	备份监控屏幕，任务程序，各种设置。	P33
配置还原	根据备份文件，恢复监控屏幕，任务程序，各种设置。	P34
保存设置	将各设置内容保存在ROM中。	P35
设置初始化	将所有设置恢复为出厂默认设置。	P35
系统信息	显示本产品的信息。	P36
重启/关机	重启/关闭本产品。	P39
CODESYS设置		
PLC程序保存	将PLC程序保存在ROM中。	P40
固件更新	更新固件。	P41
HMI设置		
Editor	显示监控画面的编辑画面（CONPROSYS HMI）。	P42
Viewer	显示监控画面（CONPROSYS HMI）。	P43
保存页面	保存编辑器在ROM中创建的屏幕内容（CONPROSYS. HMI）。	P44

3. 功能的详细说明

1. 网络设置

设置本产品的IP地址并检查网络上的通信。

※ 分别为LAN A和LAN B设置网络设置。



Network setting

LAN A	eth0
Select	☒ Static IP ☐ DHCP
IP address	10.1.1.101
Subnet mask	255.0.0.0
Default gateway	10.1.1.254
LAN B	eth1
Select	☒ Static IP ☐ DHCP
IP address	192.168.1.101
Subnet mask	255.255.255.0
Default gateway	192.168.1.254
DNS server1	10.1.1.254
DNS server2	
set	

To enable the setting, you must save settings

◆ Select

选择IP地址设置方法。

[静态IP地址]: 设置IP地址, 子网掩码等。

[DHCP获取]: 使用DHCP客户端自动获取IP地址。

【默认值】: 静态IP地址

◆ IP address

设置IP地址。

选择[静态IP地址]时有效。

【默认值】: 10.1.1.101

◆ Subnet mask

设置子网掩码。

选择[静态IP地址]时有效。

【默认值】: 255.0.0.0

◆ Default gateway

设置默认网关的IP地址。

选择[静态IP地址]时有效。

不进行设定时, 请不要输入。

【默认值】: 10.1.1.254

◆ DNS server1、DNS server2

设置DNS服务器的IP地址。

选择[静态IP地址]时有效。

不进行设定时, 请不要输入。

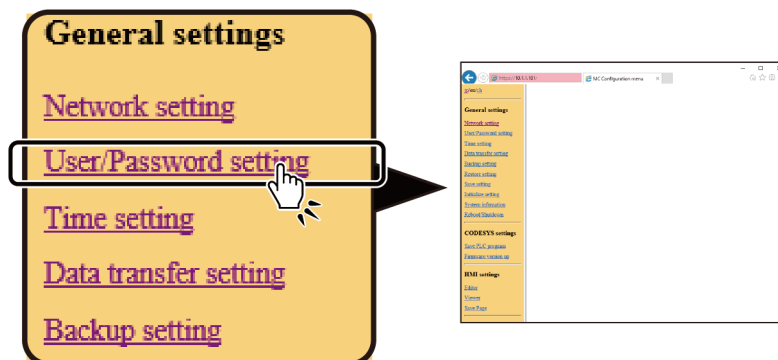
【默认值】: DNS server1 : 10.1.1.254

DNS server2 : (没有输入)

2. 用户/密码设置

设置用户和密码以便使用Web浏览器登录。

请更改出厂密码后使用。



User/Password setting

user name	
passwd	
add	add

user name	
pc341	del

To enable the setting, you must save settings

用户/密码设置

使用Web浏览器登录CONPROSYS WEB设置屏幕的用户可以添加和删除。

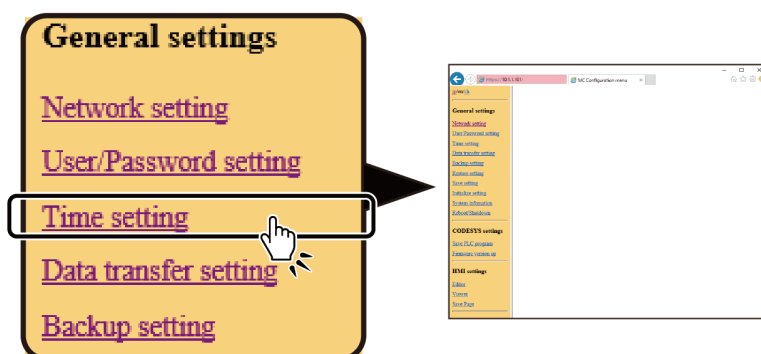
用户和密码最多可输入31个字母数字字符。

重启本产品后，对设置的更改将生效。

为了增强安全性，请在操作期间添加与默认值不同的用户并删除默认用户。

3. 时间设定

进行时间设置，设置NTP服务器获取日期和时间。



Time sync setting

Current date and time	1970-01-01 09:19:57	reload	write
Synchronization server	ntp.nict.jp		
Synchronization time	02:20		
Timezone	UTC+09(JST) ▼		
set now			

To enable the setting, you must save settings

Change date and time

yyyy-mm-dd hh:mm:ss	1970	-	01	-	01	09	:	19	:	57
set										

◆ 时间同步设置

当前日期和时间

显示当前日期和时间。

单击[reload]以更新同步服务器的时间。

单击[write]将当前时间保存在ROM中。

同步服务器

使用NTP时，请设置NTP服务器的地址。

【设置范围】: FQDN或IP地址

【默认值】: ntp.nict.jp

实施同步时间

本产品启动后每天2:20更新（同步）同步服务器的时间。（不是设置项目）

时区

使用本产品设置世界标准时间与区域标准时间之间的时差。

【默认值】: UTC+09 (JST-9)

◆ 设置日期和时间

手动设置时间时，通过在表格中输入日期和时间，时间来更新日期。

【设定范围】：1970年1月1日0时0分0秒 - 2038年1月19日3时14分7秒

【默认值】：1970年1月1日0时0分0秒

4. 数据发送设定

设定测量数据的发送目的地服务器。

关于发送数据的格式，请参照“发送数据的格式（P160）”。



Data transfer setting

Data transfer URL

set

To enable the setting, you must save settings

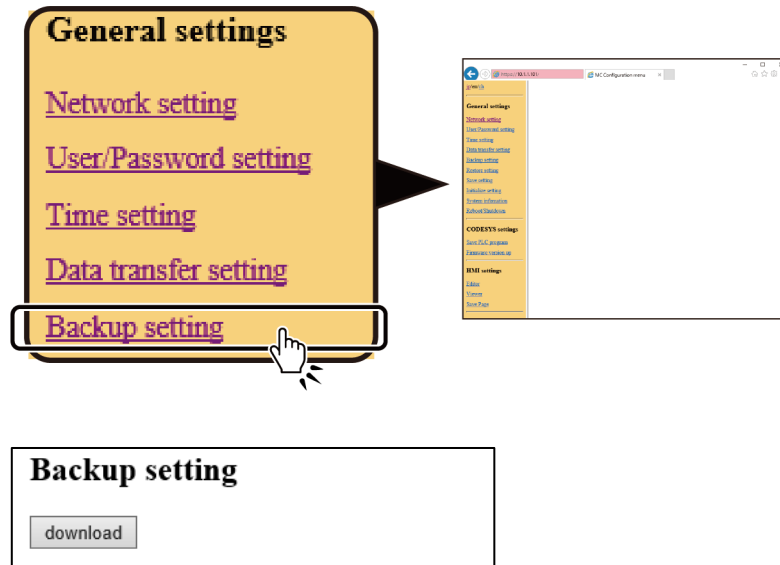
◆ 数据发送目的地URL

输入要发送的数据的服务器的URL。

【默认值】：无设置

5. 设置备份

备份监控画面、任务程序、各种设定。

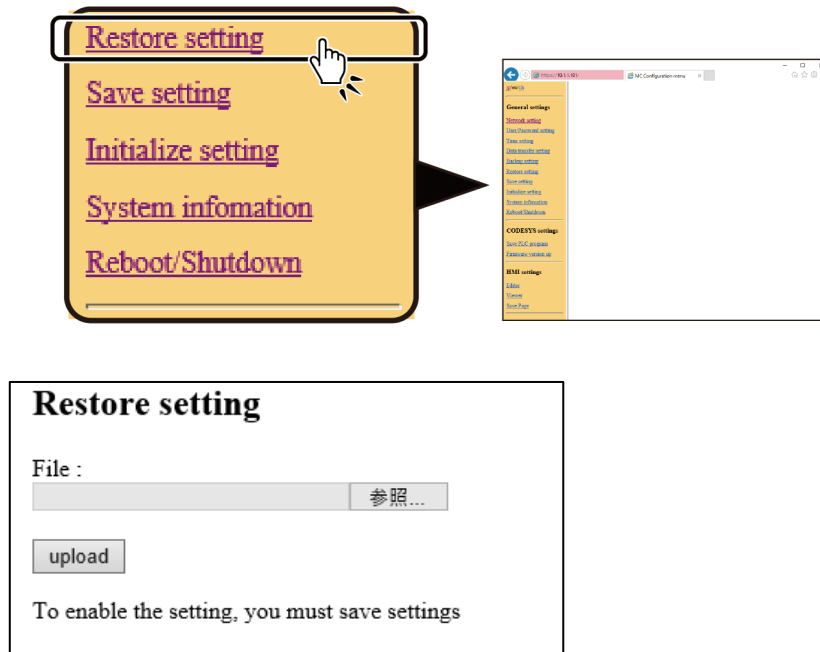


使用[download]来保存各种设定的备份。

由于默认文件名是config.dat，因此请使用任意文件名保存。

6. 设置还原

根据备份文件，恢复监控画面，任务程序，各种设置。



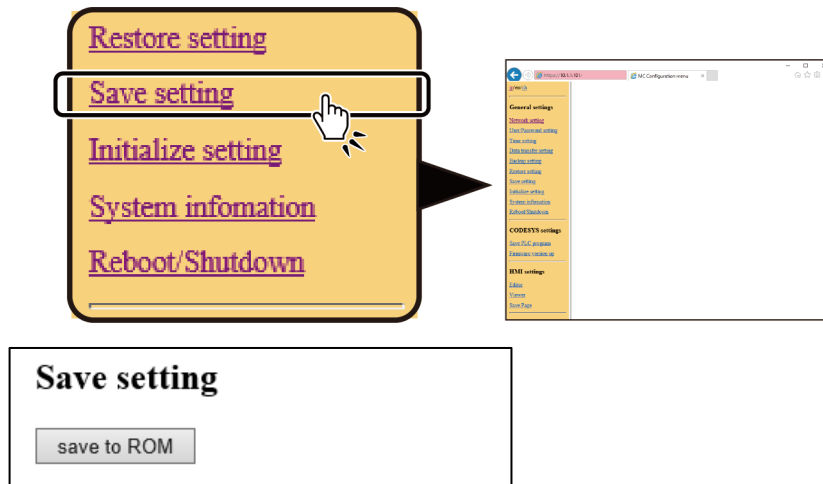
从[浏览]中选择保存在配置备份中的备份文件，然后单击[upload]。

恢复的各种设定可以在各自的设定画面中确认，请有效设置[设定保存]和[再启动]。

7. 设置保存

将CONPROSYS WEB Setting设置的内容保存到ROM中。

点击[save to ROM]，PWR的LED会闪烁，并开始保存。



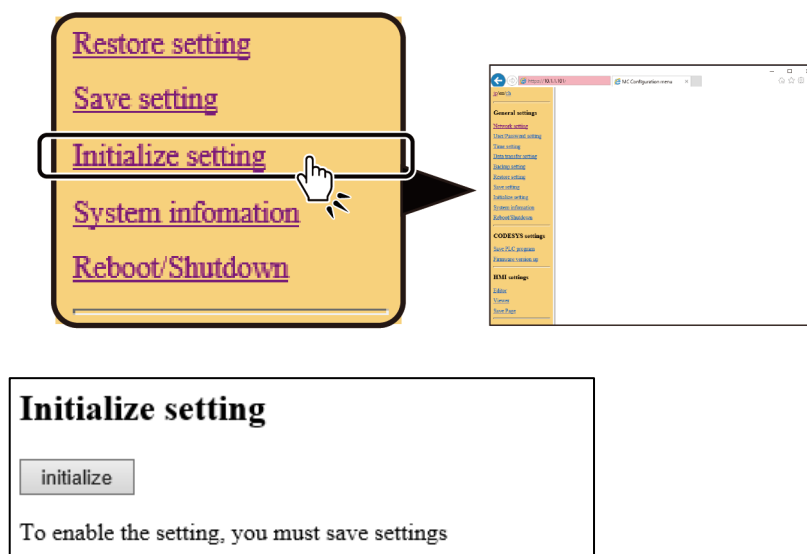
⚠ 注意

保存设置时，在主机上的PWR LED完成闪烁之前，请勿关闭电源。（所需时间：约5秒）

如果在不保存设置的情况下重新启动或关闭电源，则设置的内容将恢复为原始设置。

8. 设置初始化

将本产品重置为出厂默认设置。

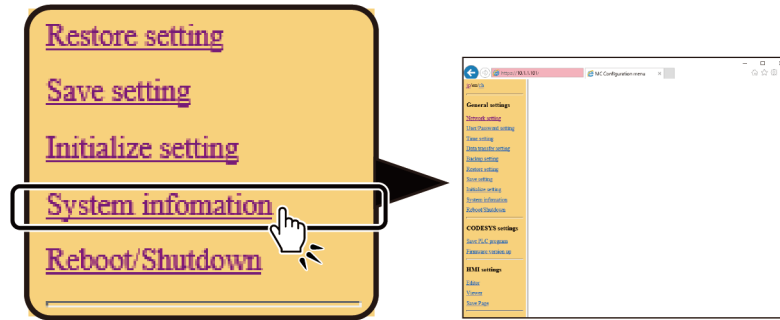


单击[initialize]，进行初始化设定。

要启用初始化的设定，请进行[设定保存]和[再启动]。

9. 系统信息

显示本产品的信息。



显示以下画面。

System infomation		
Version	1.3.0	
Serial number	XXXXXXXXXXXX	
ID	XXXX-XXXX-XXXX-XXXX-XXXX-XXXX-XXXX-XXXX	
MAC address	LAN A	XX:XX:XX:XX:XX:XX
	LAN B	XX:XX:XX:XX:XX:XX
Runtime version	1.2.0.0	
Driver version	1.2.0.0	
Battery power	Yes	
Server comm log	View	
Detail	View	
license	View	

◆ 版本

显示固件版本。

◆ 序列号

显示本产品的序列号。

◆ ID

显示在云服务器CDS或CDS 2上注册所需的ID。

◆ MAC地址

显示有线LAN的MAC地址。

◆ 运行版本

显示运行时版本。

◆ 驱动程序版本

显示驱动程序的版本。

◆ 电池余量

显示有无电池剩余量。

◆ 服务器连接日志

显示与服务器的通信日志。

项目	说明
Web服务器通信日志	显示与在数据发送目的地URL中指定的服务器的最新通信日志。
Azure log	显示Azure IoT Hub的最终发送日志。
NTP服务器通信日志	显示与指定的NTP服务器的最新通信日志。
邮件发送日志	显示与所指定的SMTP服务器的最新通信日志。

Web server comm log

Log not found

NTP server comm log

```
Error resolving ntp.nict.jp: Name or service not known (-2)
12 Jan 02:20:20 ntpdate[515]: Can't find host ntp.nict.jp: Name or service not kno
12 Jan 02:20:20 ntpdate[515]: no servers can be used, exiting
```

◆ 详细信息

显示系统信息的详细信息。

uptime

```
02:26:49 up 1:07, 0 users, load average: 0.00, 0.01, 0.04
```

free

```
Mem:          total        used        free        shared        buffers
-/+ buffers:  513172      197008      316164           0          236
Swap:           0           0           0
```

df

```
Filesystem      1K-blocks      Used Available Use% Mounted on
/dev/root        63461        30738      29447    51% /
devtmpfs         256584           0      256584     0% /dev
none             256584          16      256568     0% /var
none             256584       14884      241700     6% /tmp
none             256584           0      256584     0% /dev
/dev/mtdblock5   18688       12872       5816    69% /mnt/mtd
/dev/mtdblock6   32768         536      32232     2% /mnt/mtd2
tmpfs            256584       35020     221564    14% /home
none             256584       14884      241700     6% /home/opt/httpd/httpd
```

ps aux

```
PID  USER  TIME  COMMAND
1  root   0:06  init
2  root   0:00  [kthreadd]
3  root   0:00  [ksoftirqd/0]
4  root   0:00  [kworker/0:0]
6  root   0:00  [khelper]
7  root   0:00  [kdevtmpfs]
8  root   0:00  [netns]
9  root   0:00  [sync_supers]
10 root   0:00  [bdi-default]
11 root   0:00  [kblockd]
```

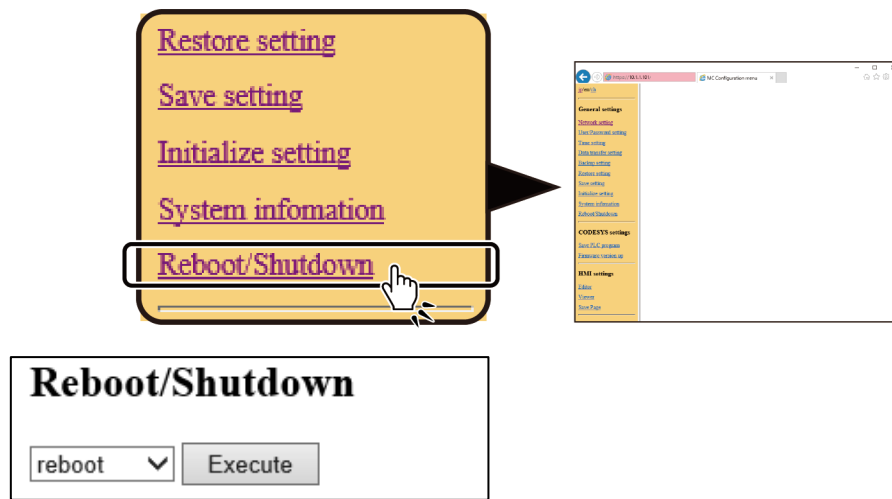
◆ license

单击[显示]，将显示使用软件的许可信息。

使用软件需要许可。

10. 重新启动/关闭

本产品的重新启动或关闭电源。



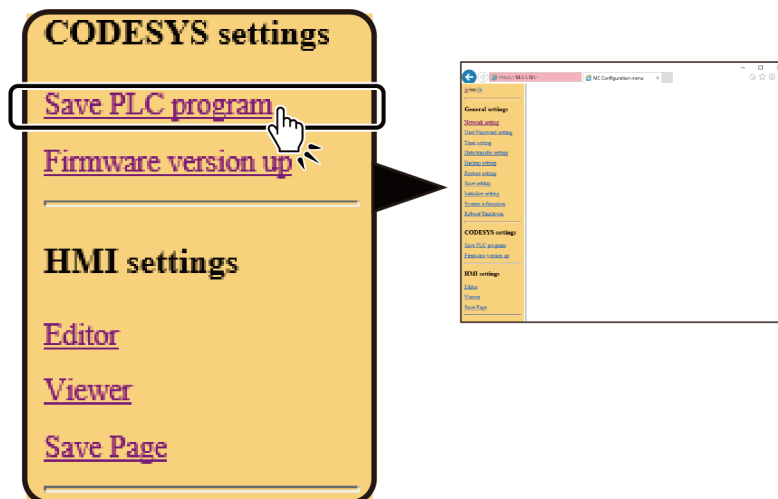
选择重启或关机并单击[start]时，LED开始闪烁并重新启动或关闭电源。

11. PLC程序保存

将PLC程序保存在ROM区域中。

单击[写入]按钮时，ST1和ST2 LED将闪烁并开始存储。

保存完成后闪烁停止。



⚠ 注意

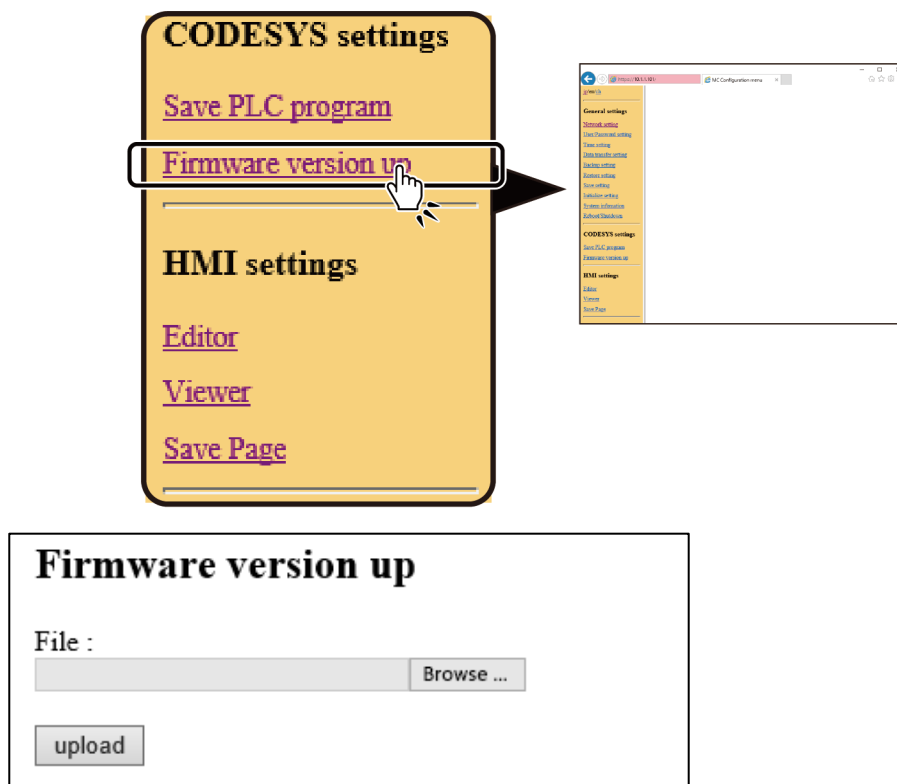
保存设置时，主机LED的ST1和ST2闪烁。

请不要在此期间关闭电源。（所需时间：约5秒）

如果在不保存设置的情况下重新启动或关闭电源，则设置的内容将恢复为原始设置。

12. 固件更新

进行固件的更新。



固件更新文件可以从我们的网站下载。

下载

<https://www.contec.com/cn/download/>

从[浏览]按钮中选择下载的文件，然后单击[upload]按钮。

※ 由于下载的文件是以zip格式压缩的，因此将其解压缩并选择扩展名为bin的文件。

⚠ 注意

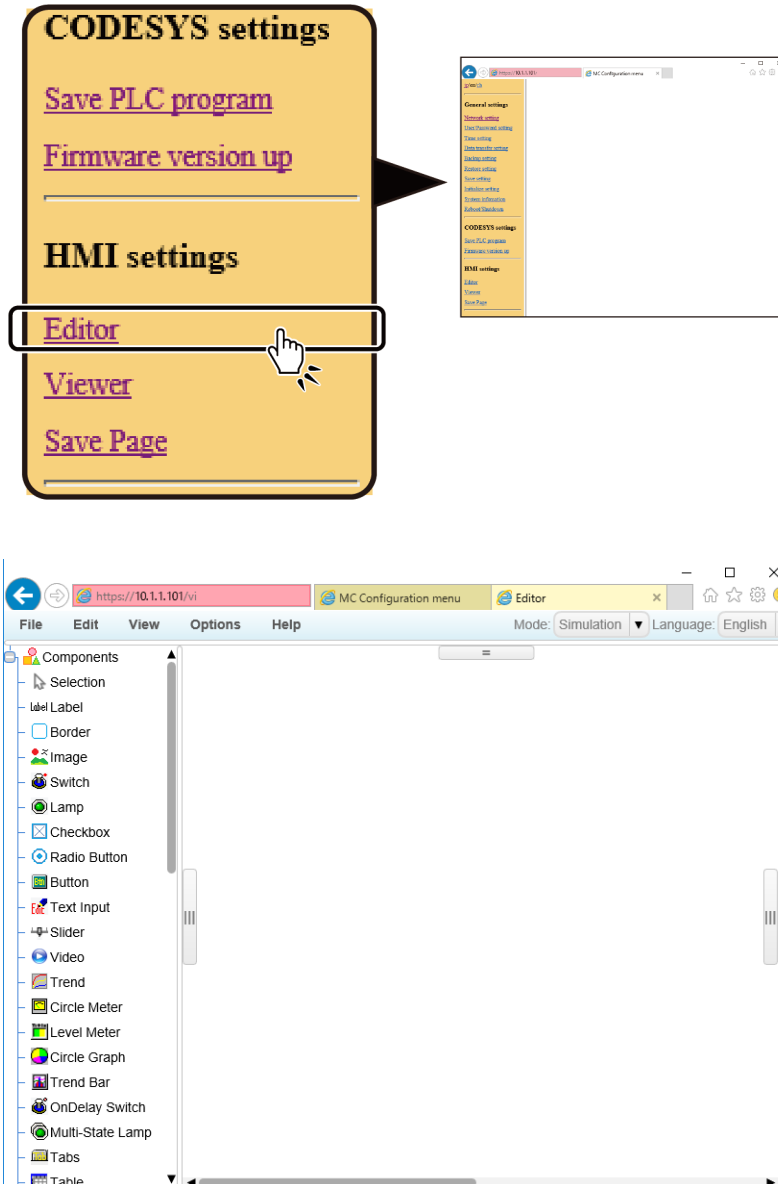
固件更新时，主机LED的ST1和ST2将闪烁。

期间请不要关闭电源。 数据将被损坏，无法启动。

13. Editor

显示监控画面的制作/变更的画面。

显示项目并可以简单地制作自己的监控画面。

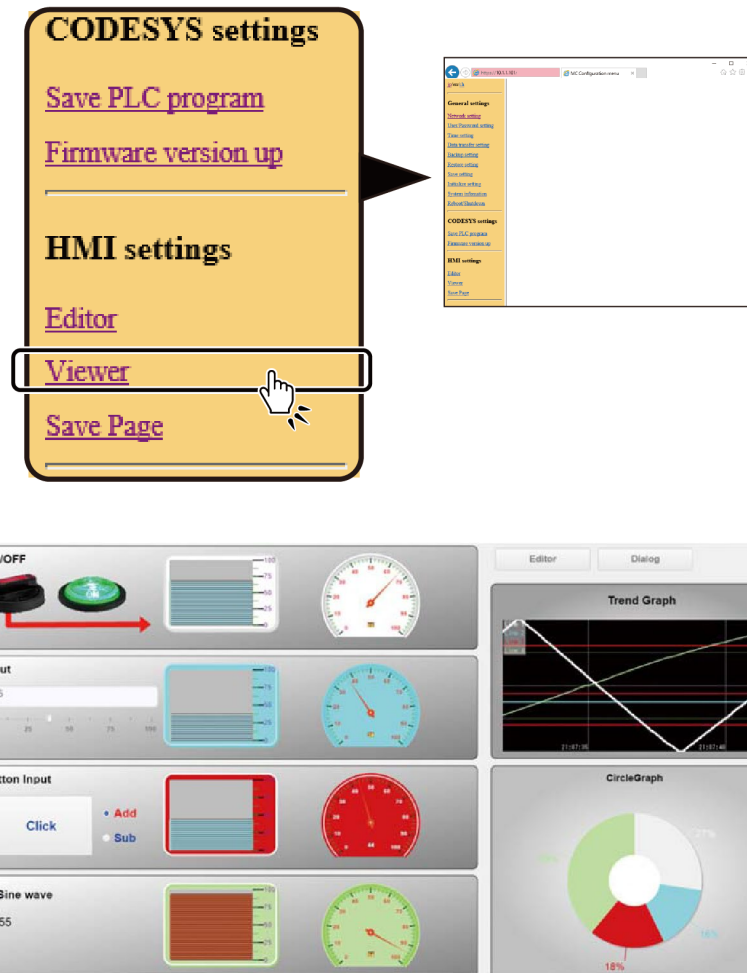


※ 有关监视编辑画面的详细信息，请参阅“**监控编辑（P136）**”。

14. Viewer

显示监控画面。

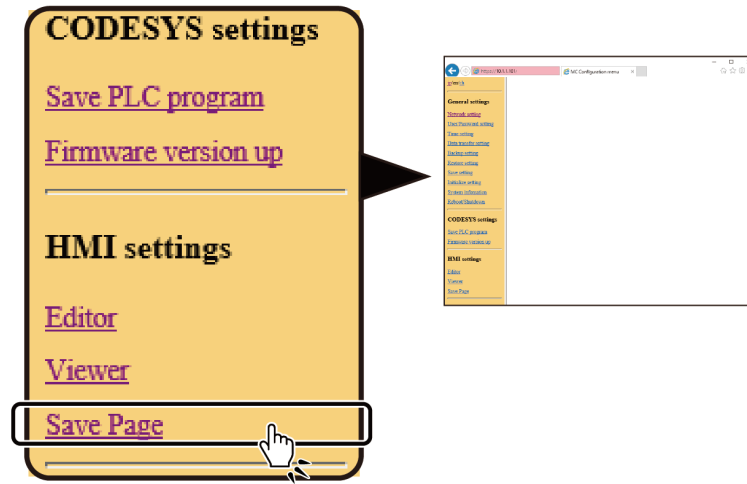
监控画面可以监控输入的信号状态。



※ 有关监视编辑画面的详细信息，请参阅“**监控编辑**（P136）”。

15. 保存页面

将Editor创建的页面内容保存在ROM区域中。



安装CODESYS

CODESYS安装顺序的说明。

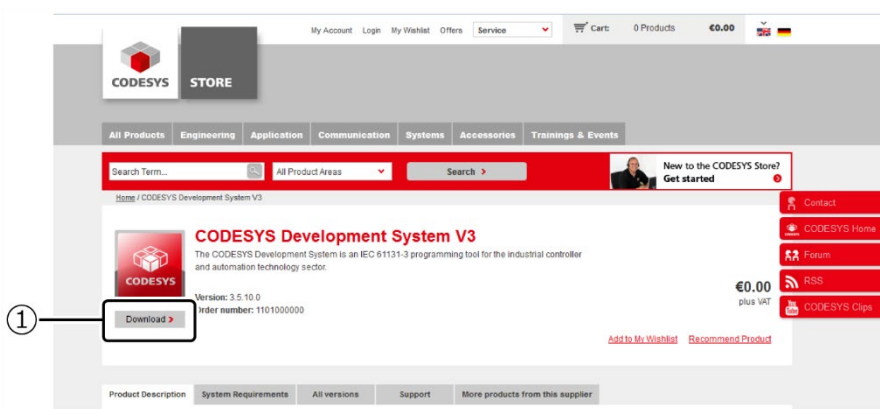
1. 安装CODESYS

PAC系列使用符合国际标准IEC 61131-3的集成开发环境“CODESYS”，无论平台如何，都可以开发PLC和HMI。
为了使用CODESYS，有必要在个人计算机上安装CODESYS开发环境和“COPRESYS COPRESYS包”。

- 1 如何获得CODESYS的开发环境
从“CODESYS Store”下载CODESYS开发环境。
- 2 使用Web浏览器访问CODESYS Store，然后单击①[Download]按钮。
CODESYS会不时更新，所以请下载最新版本。

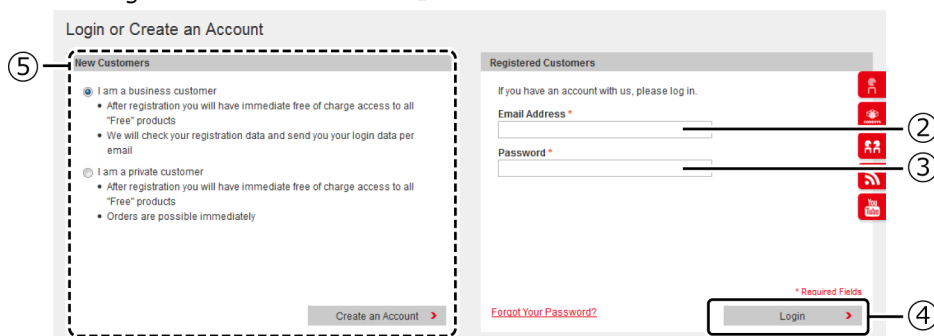
下载

<http://store.codesys.com/codesys.html>



- 3 单击[Download]按钮后，将显示“Login or Create an Account”屏幕。
如果有CODESYS Store帐户，请在“Registered Customers”中输入②“Email Address”和③“Password”，然后单击④“Login”按钮。
对于新用户，请在用户注册⑤“New Customers”后登录。

▼『Login or Create an Account』画面



- 4 在“Registered Customers”中输入“Email Address”和“Password”，然后登录下载中心。登录后，单击[Download]开始下载。
- 5 解压缩下载的文件，运行安装文件（扩展名为.exe的文件）并安装CODESYS。

2. 安装CODESYS包

要将本产品与CODESYS一起使用，必须为CONPROSYS安装CODESYS包。

- 1 从我们网站上的下载菜单中选择[Driver • Software]。 请按产品名称或类别（PAC系统）缩小范围，单击[搜索]，然后单击此产品的驱动程序。

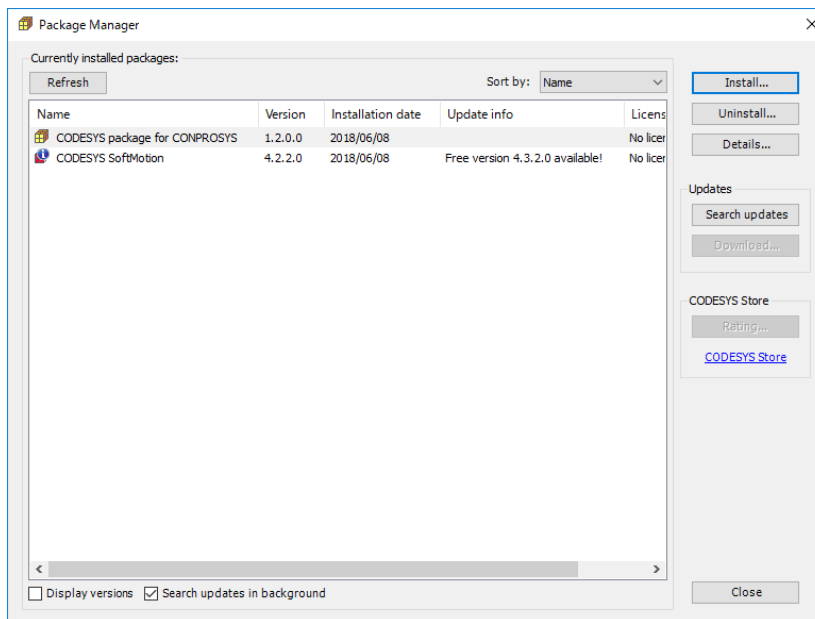
下载

<https://www.contec.com/cn/>

- 2 下载CODESYS包。

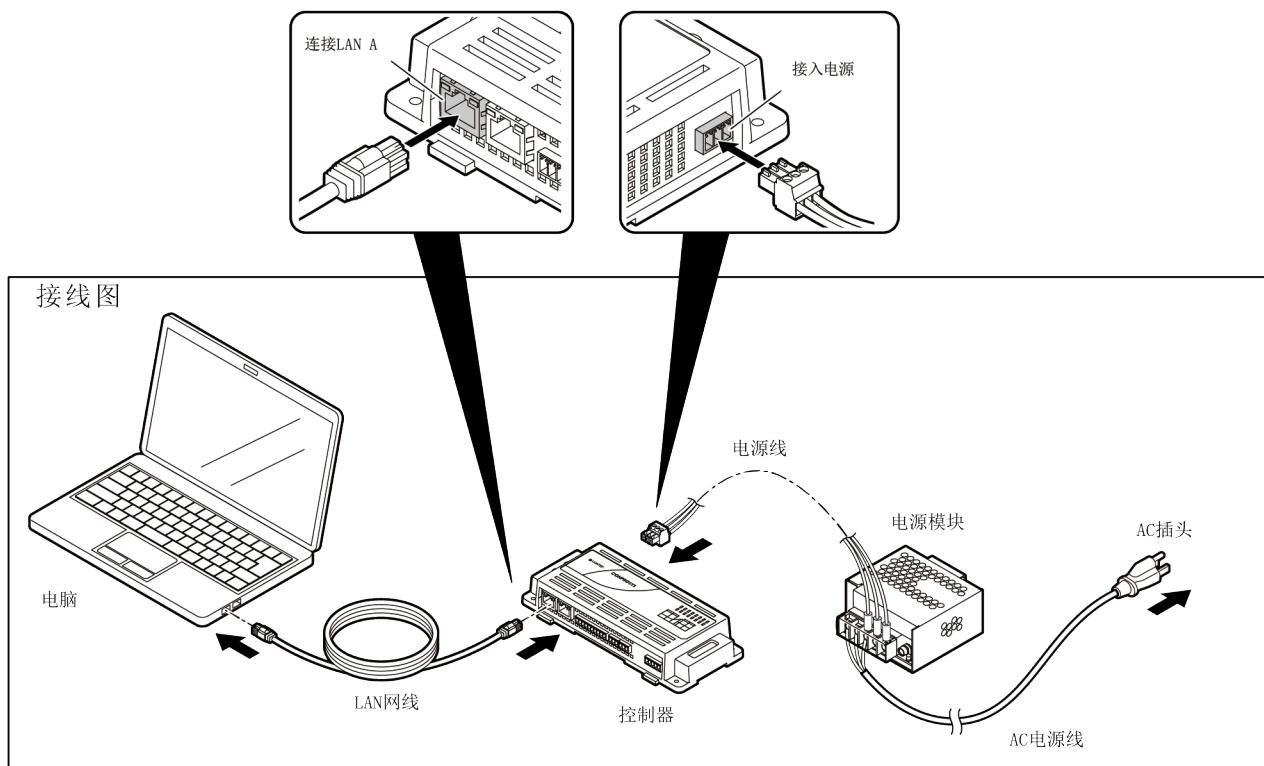
- 3 执行CODESYS并从菜单的[Tool]中选择[Package Manager ...]。

- 4 单击[安装...]，解压缩下载的文件并指定CODESYS包文件。
文件名: CONTEC CONPROSYS PAC Series.package

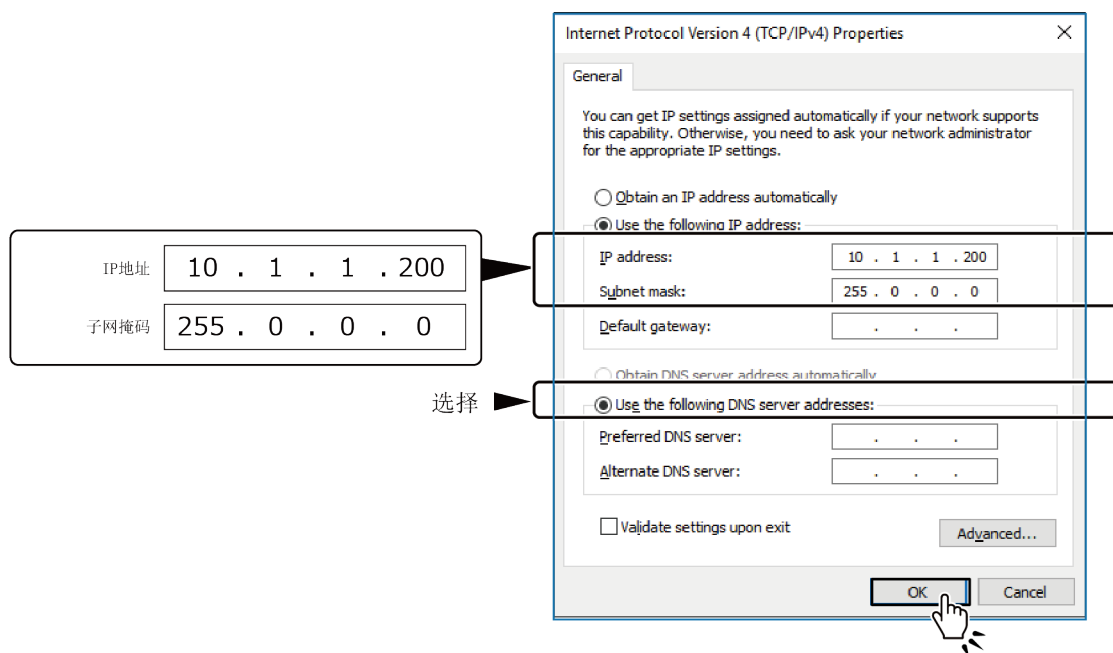


3. 连接电脑和CONPROSYS

- 1 使用以太网电缆将CODESYS开发环境的PC LAN端口与本产品的LAN A端口连接。
※ 请使用本产品的LAN A端口连接个人计算机。LAN B用于EtherCAT通信和Modbus TCP从站连接。



- 2 打开本产品的电源。
- 3 更改PC的网络设置。
本产品的LAN A端口，出厂设置的IP地址为“10.1.1.101”，子网掩码为“255.0.0.0”。
如果LAN A的IP地址为“10.1.1.101”，请将PC的IP地址设置为“10.1.1.200”。将子网掩码设置为“255.0.0.0”。



※ 对于PLC控制器，必须设置与网络上的其他设备不重叠的IP地址。 可以通过Web浏览器设置PLC控制器的IP地址。

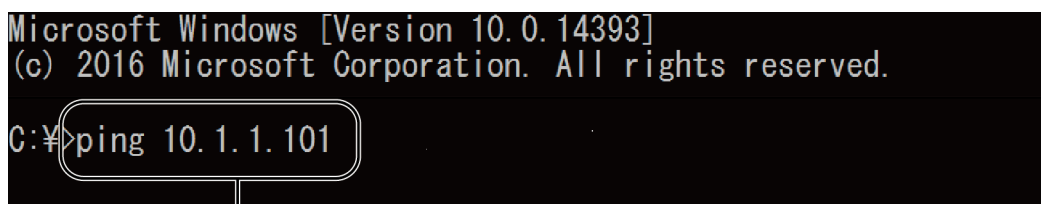
※ 有关IP地址设置的详细信息，请参阅“**电脑网络设定（P18）**”。

- 4** 请使用ping命令等，确认电脑与PLC控制器连接。
打开命令程序，对IP地址“0.1.1.101”执行ping命令。

格式

ping设备的IP地址

如有响应，则可以确认它已连接。



输入ping 10.1.1.101并按[Enter]键。

4. 固件版本升级

升级CODESYS相关固件版本。

- 1 从公司网站上的下载菜单中选择[Driver • Software]。
按产品名称或类别（PAC系统）过滤，单击[搜索]，单击此产品的驱动程序。

下载

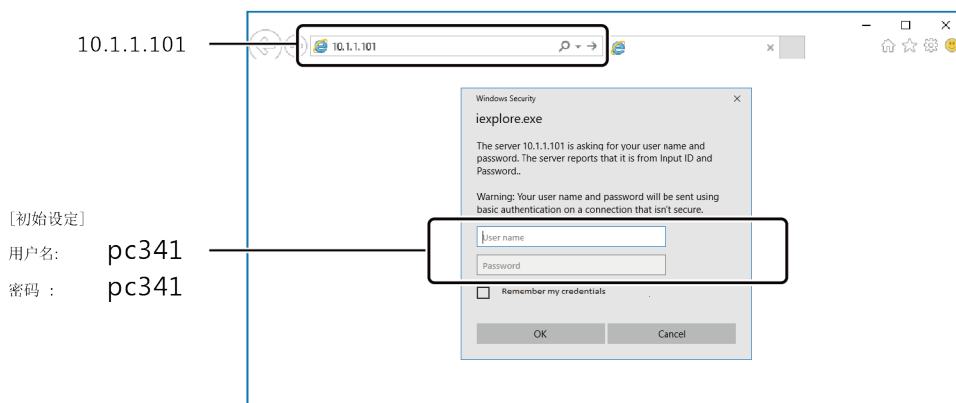
<https://www.contec.com/cn/>

- 2 下载固件。
(由于下载的文件是以zip格式压缩的，因此将其解压缩并选择扩展名bin的文件)

- 3 『连接电脑和CONPROSYS (P48)』的顺序，连接电脑和本产品。

- 4 启动Web浏览器，输入地址“http : / / / 0.1.1.101 / (PLC控制器的IP地址) ”。
※ 也可以访问“https://10.1.1.101/”。 如果显示“证书错误”屏幕，请选择“继续浏览此站点”。

- 5 用户名为“pc341”，密码输入“pc341”并登录。
可以在“一般設定”的“系统信息”屏幕上检查固件版本。



- 6 在“CODESYS设置”中选择[固件更新]，单击“选择文件”，然后打开解压缩的固件。

7 单击[upload]按钮开始更新固件。 更新固件时，主机上的ST1和ST2 LED重复闪烁。

注意

固件更新时，主机LED的ST1和ST2将闪烁。

期间请不要关闭电源。 数据将被损坏，无法启动。

8 更新完成后，将自动重启，请在“系统信息”屏幕上确认版本已更新。

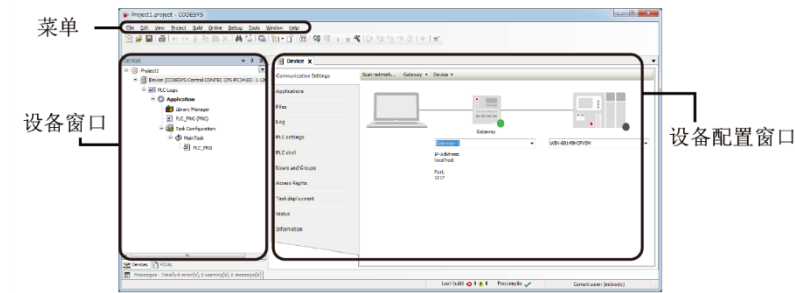
基本编程

在CODESYS的编程中，说明了用于控制本产品的基本程序。

1. CODESYS各部分名称

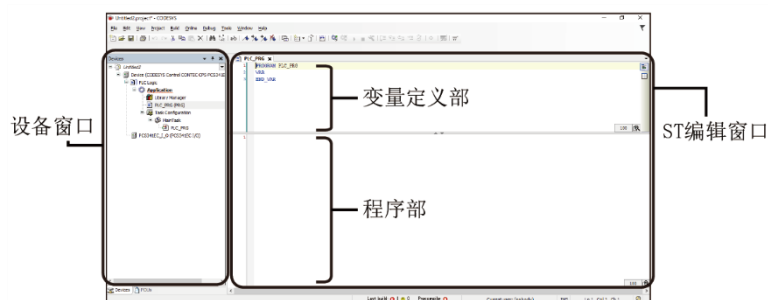
CODESYS的基本画面，在下图的窗口中显示。

1. 设备窗口，设备配置窗口



※ 双击设备窗口中的“设备 (CODESYS CONTEC CONTEC CPS - PCXXXXXXXXX)”图标时，将打开设备配置窗口。

2. ST编辑器窗口



※ 双击设备窗口中的“PLC_PRG (PRG)”图标时，将打开ST编辑器窗口。

2. ST语言格式和描述方法

本书的编程例子使用了ST语言。

显示了ST语言的基本格式，例如赋值公式，条件式，注释方法。

◆ 注释

为程序添加注释。

```
/* 注释的内容 */  
(* 注释的内容*)  
// 注释的内容
```

- 要注释多行，请用“/ * * /”或“(* *)”括起来。
- 一行中的注释开头带有“//”前缀。

◆ 赋值公式

为变量赋值。

```
D00 := 1;
```

- “变量名称”:= “value”;为变量赋值。
- 在行尾需要“;”。

◆ 定义类型

定义变量的数据类型。

```
StartFlag : BOOL := TRUE;  
Error : BOOL;  
TestString : STRING := 'Test String!';
```

- 第一行：将要赋值给变量“StartFlag”的值指定为布尔值（True / False）并指定True。
- 第2行：分配给变量“Error”的值作为Boolearn值。
- 第3行：将要分配给变量“TestString”的值指定为字符串，并替换为“TestString!”。

◆ 条件式(IF – THEN)

判定条件来执行程序/回避。

```
IF StartFlag THEN  
  (例行程序)  
END_IF
```

- 如果变量“StartFlag”的值为“True”，则执行END_IF包含的例程。

◆ 条件式(CASE – OF)

判定条件来执行程序/回避。

```
CASE State OF  
  0:  
    (例行程序1)  
  1:  
    (例行程序2)  
END_CASE
```

- 如果变量“State”的值为“0”，则执行例程1。
- 如果变量“State”的值为“1”，则执行例程2。

※ 有关ST语言的信息，请参阅“PLCopen”站点，该站点是为PLC编程传播国际标准IEC 61131-3的第三方机构。

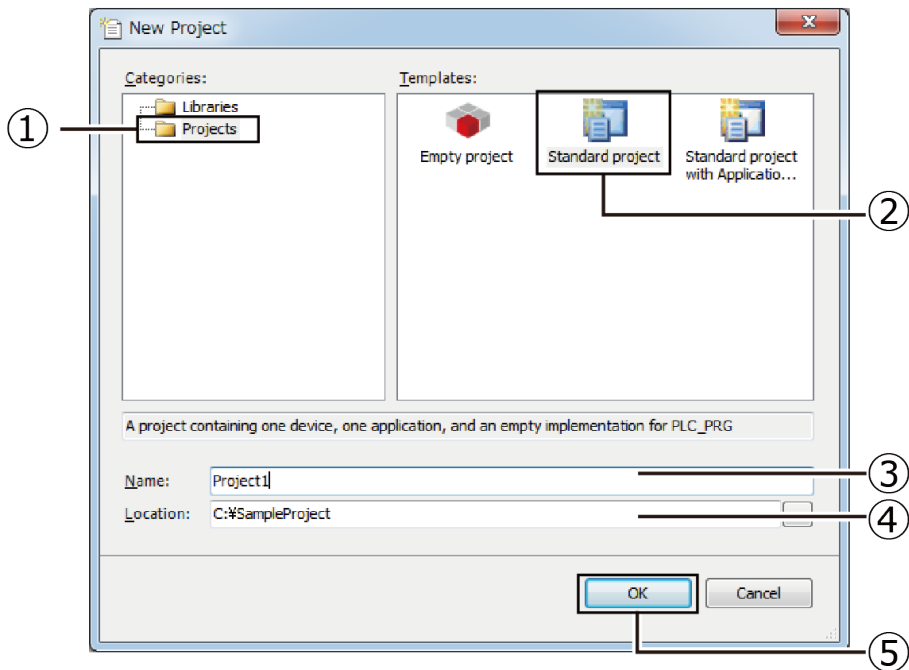
PLCopen

<https://www.plcopen-japan.jp/>

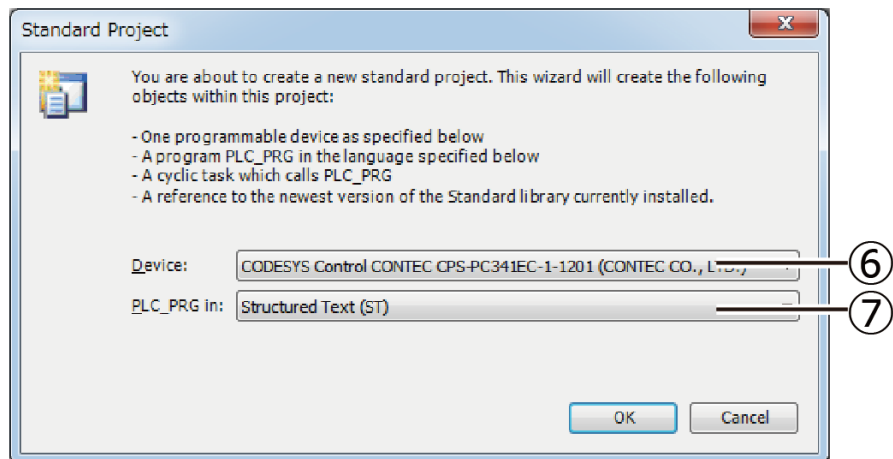
3. 基本编程顺序

1. 新建项目的生成

- 1 启动CODESYS，从[文件]菜单中选择[新项目]。
- 2 在“新建项目”对话框中，从类别中选择①“项目”，从模板中选择②“Standard project”。指定项目的③“名称”和④“位置”（保存目的地）⑤单击[确定]。

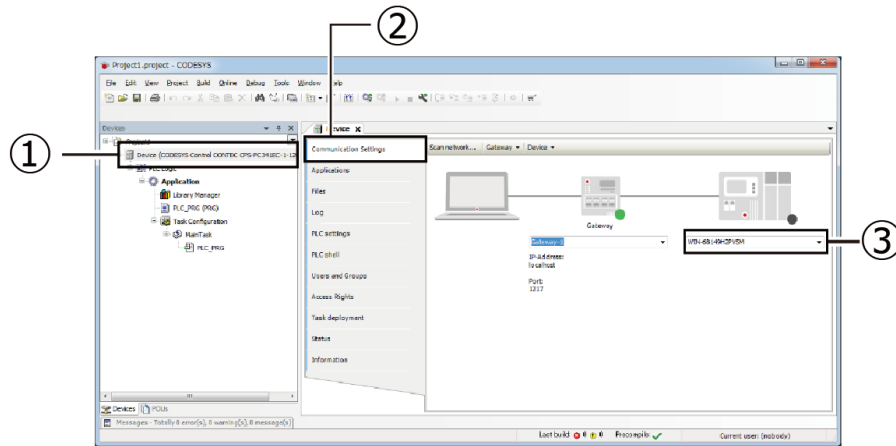


- 3 在“标准项目”对话框中，在⑥“设备”栏中选择使用的控制器。设备标记为“CODESYS CONTEC CONTEC CPS - PCXXXXXXXXXX”。
⑦对于“PLC_PRG的语言”，选择Structured Text (ST)。

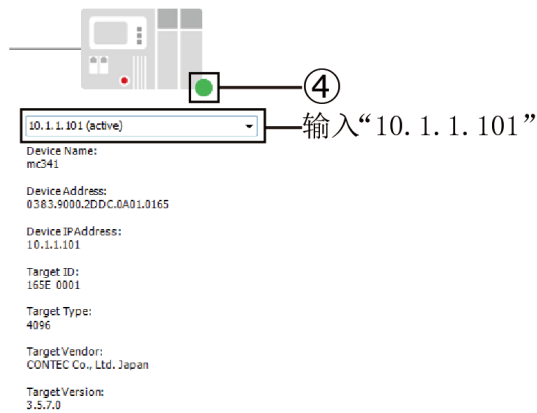


2. 从CODESYS连接到控制器

- 1 双击设备窗口中的① “Device (CODESYS Control CONTEC CPSPCXXXXXXXXXX)” 图标，打开 “设备配置窗口”，然后单击② “通信设置” 选项。
- 2 ③在 “组合框” 中输入 “10.1.1.101”，然后按键盘上的[Enter]。



- 3 建立与控制器的连接后，控制器图像的黑色圆圈变为绿色。

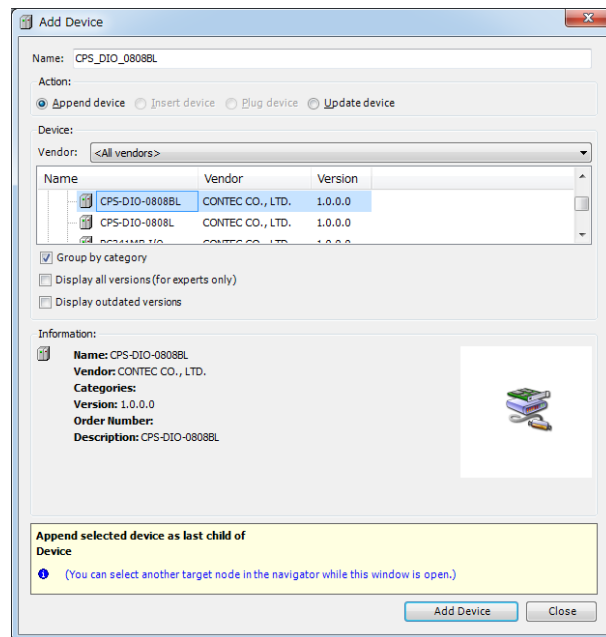


3. 添加 I / O 模块

I / O 模块可以自由组合添加到堆栈型控制器中。

※ 由于控制器中安装的 I / O 功能已在 CODESYS 设备窗口控制器的树形层次结构中注册（显示为“PCS341XX I / O”），因此不需要执行以下步骤。

- 1 右键单击 CODESYS 设备窗口中的控制器，然后选择[添加设备]。
- 2 在“添加设备”对话框中，选择要添加的 I / O 模块，然后单击[添加设备]按钮。 I / O 模块在“其他”树层次结构中注册。



4. I / O变量的定义

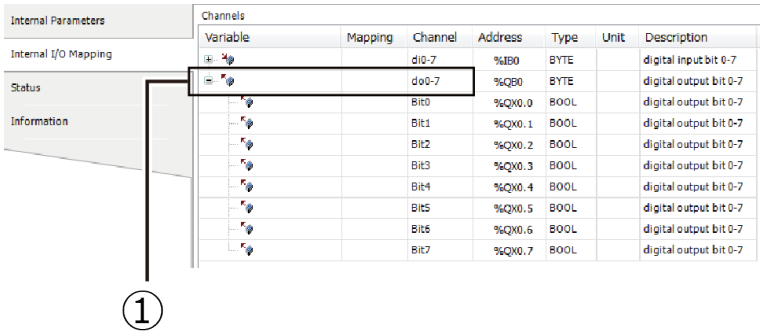
为了在CODESYS中进行I / O操作，将I / O模块的输入/输出通道定义为变量，请输入I / O映射或参数内的变量。

以下是定义I / O变量以将变量“D00”分配给CPS-DIO-0808BL的D0端口的位0的过程。 安装在其他I / O模块和控制器中的I / O定义也可以在此过程中执行。

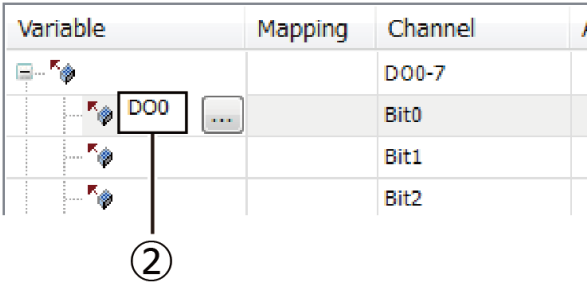
1 在“添加I / O模块（P58）”过程中添加CPS-DIO-0808BL，然后双击设备窗口中的图标。
※ 安装在控制器中的I / O功能在CODESYS设备窗口的控制器的树形层次中注册（显示为“PCS341XX I / 0”）。

2 显示配置窗口，选择Intnal I / O映射。

3 ①用“channel”打开树“do 0 - 7”。



4 ②双击“通道”“位0”行的“变量”列的空白部分，然后输入“D0 0”以定义变量。

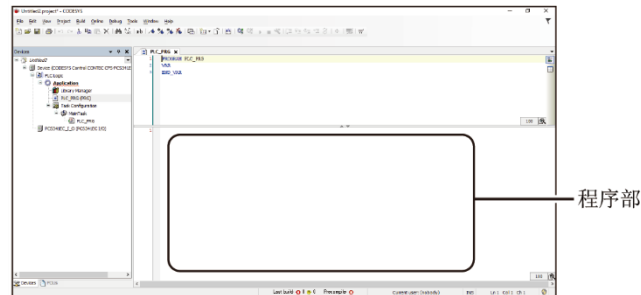


5. 程序创建和构建

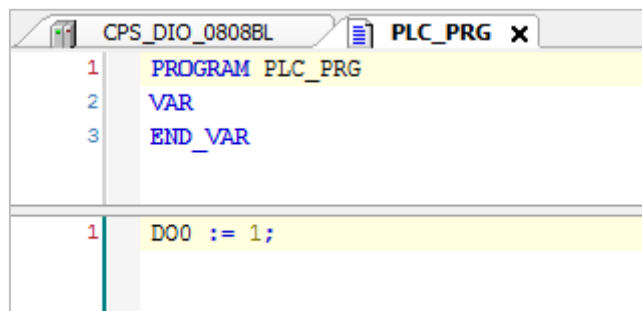
使用“**I / O变量的定义（P59）**”中定义的I / O变量“D00”创建PLC程序，并从数字输出位0输出ON。 编程使用ST语言。

1 双击设备窗口中的[PLC_PRG（PRG）]图标以显示“ST编辑器窗口”。

2 在“ST Editor窗口”的“程序部”中写下以下源代码。



```
D00 := 1;
```



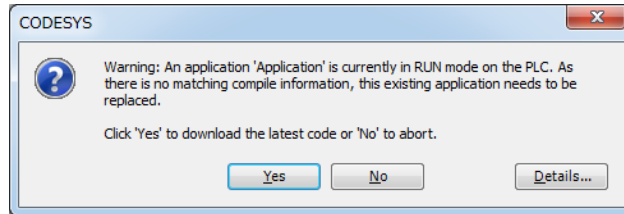
由于变量D0 0被定义为CPS-DI0-0808BL的输出位0，它从CPS-DI0-0808BL的数字输出位0输出ON信号。

3 从CODESYS的[Build]菜单执行[Rebuild]，检查构建过程是否正常完成。
如果发生错误，可能原因是由于不正确设置I / O定义，没有升级固件，或者库管理器中可能存在未知库。

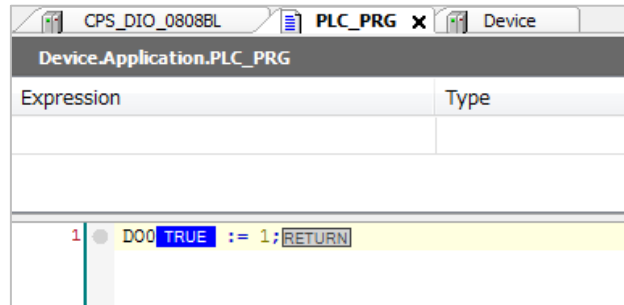
6. 程序的下载和执行

1 使用“程序创建和构建 (P60)”完成构建后，从CODESYS在线菜单登录。

2 将显示下载确认对话框。单击[是]。



3 执行[Debug]菜单中的[Run]时，程序操作开始。
检查CPS-DIO-0808BL前面板上的D0位0 LED是否亮起。
在ST编辑器窗口中，当前值“TRUE”显示在“D0 0”变量旁边。



7. PLC程序的ROM保存

由于PLC程序下载到RAM区域，因此在关闭控制器电源时会被删除。为避免这种情况，请将程序保存在ROM区域中。

1 在与控制器连接的PC上启动Web浏览器，在地址输入部分输入“http://10.1.1.101/”（PLC控制器的IP地址）。

※有关控制器连接的详细信息，请参阅“**电脑网络设定（P18）**”。

※也可以访问“https://10.1.1.101/”。如果显示“证书错误”画面，请选择“继续浏览此站点”。

2 用户名为“pc341”，密码输入“pc341”并登录。

3 在“CODESYS设置”中选择“保存PLC程序”，然后单击“保存到ROM”。在写入完成之前，主机上的ST1和ST2 LED重复闪烁。

⚠ 注意

保存设置时，主机LED的ST1和ST2会闪烁。

请不要在此期间关闭电源。（所需时间：约5分钟）

如果在不保存设置的情况下重新启动或关闭电源，则设置的内容将恢复为原始设置。

8. 删除ROM保存的PLC程序

要删除保存在ROM区域中的PLC程序，请按照以下步骤操作。

1 在CODESYS开发环境中登录控制器。

※有关登录的详细信息，请参阅“**从CODESYS连接到控制器（P57）**”。

2 从CODESYS菜单执行[在线] - [复位（PLC初始化）]。

3 按照“**PLC程序的ROM保存（P62）**”的相同步骤将PLC程序保存到ROM中。
重新启动控制器后，将删除PLC程序。

4. 产品接口定义

参数，I / O映射，串口各种输入/输出接口，如何在CODESYS上定义。

1. 参数

控制器操作开始时，该参数仅需设置一次。在CODESYS中会显示在[Internal Parameters]中。

产品型号	参数名称	CODESYS中的标记	含义
CPS-PC341EC-1-9201	无	—	—
CPS-PC341MB-ADSC1-9201	数字过滤器	di filter	选择数字输入滤波器值 (* 1) 默认设置 : not used
CPS-PCS341EC-DS1-1201 CPS-PCS341MB-DS1-1201	通用输入/输出方向	GPI00 - GPI03	从[Input] [Output]中选择输入/输出方向 默认设置 : Input
CPS-DIO-0808L CPS-DIO-0808BL CPS-DIO-0808RL CPS-DI-16L CPS-DI-16RL	数字滤波器	di filter	选择数字输入过滤器的值 (* 1) 默认设置: not used
	内置电源	internal power	选择内置电源的ON / OFF (CPS-DIO-0808BL) 默认设置 : ON
CPS-SSI-4P	结线方式	wire0-3	结线方式从[3-wire] [4-wire]中选择 默认设置: 4-wire

*1 数字输入过滤器的值

输入过滤器值	
not used	1.024 msec
0.25 μsec	2.048 msec
0.5 μsec	4.096 msec
1.0 μsec	8.192 msec
2.0 μsec	16.384 msec
4.0 μsec	32.768 msec
8.0 μsec	65.536 msec
16.0 μsec	131.072 msec
32.0 μsec	
64.0 μsec	
128.0 μsec	
256.0 μsec	
512.0 μsec	

2. I / O映射

I / O映射是在每个循环时间重复执行的项目。在CODESYS中，它显示在[Internal I/O Mapping]中。

产品型号	I / O映射名称	CODESYS中的标记	含义
CPS-PC341EC-1-9201	电池状态	battery	电池状态 0=没电池余量 1=有电池剩余
CPS-PC341MB-ADSC1-9201	数字输出	do0-1	数字输出位0 - 1
	数字输入	di0-3	数字输入位0 - 3
	计数器输入0	cnt0	计数器输入通道0 范围: 0 - 16, 777, 215
	计数器输入1	cnt1	计数器输入通道1 范围: 0 - 16, 777, 215
	模拟输入0	ai0	模拟输入通道0 范围: 0 - 4, 095
	模拟输入1	ai1	模拟输入通道1 范围: 0 - 4, 095
	清除计数器	cnt clear	清除计数器值命令 Bit0 = 0N清除计数器0 Bit1 = 0N清除计数器1
	电池状态	battery	电池状态 0=没电池余量 1=有电池剩余
CPS-PCS341EC-DS1-1201 CPS-PCS341MB-DS1-1201	电池状态	battery	电池状态 0=没电池余量 1=有电池剩余
	数字输出※	digital outputs (GPIO0..GPIO3)	数字输出位0 - 3
	数字输入※	digital inputs (GPIO0..GPIO3)	数字输入位0 - 3
CPS-DIO-0808L CPS-DIO-0808BL CPS-DIO-0808RL	数字输出	do0-7	数字输出位0 - 7
	数字输入	di0-7	数字输入位0 - 7
CPS-AI-1608LI CPS-AI-1608ALI	模拟输入	ai0-7	模拟输入通道0 - 7 范围: 0 - 65, 535
CPS-AO-1604LI CPS-AO-1604VLI	模拟输出	ao0-3	模拟输出通道0 - 3 范围: 0 - 65, 535
CPS-RRY-4PCC	数字输出	do0-3	数字输出位0 - 3
CPS-DI-16L CPS-DI-16RL	数字输入	di0 0-7 di1 0-7	数字输入0 位0 - 7 数字输入1 位0 - 7
CPS-DO-16L CPS-DO-16RL	数字输出	do0 0-7 do1 0-7	数字输出0 位0 - 7 数字输出1 位0 - 7
CPS-SSI-4P	温度输入	temp0-3	温度输入通道0 - 3 范围: -200 - 800、断开连接时-999
	SSI数据输入(*1)	data0-3	SSI数据输入通道0 - 3

CPS-CNT-3202I	计数器输入	cnt_val0-1	计数器输入通道0 - 1
	状态输入	status0-1	状态输入通道0 - 1
	控制标志输出	out_ctrl0-1	控制标志输出通道0 - 1 位0=OFF停止、ON开始 位1=计数值设置，从OFF变为ON
	控制标志输入	in_ctrl0-1	控制标志输入通道0 - 1
	事件输入	event0-1	事件输入通道0 - 1 位0= ON与计数比较值一致
	事件重置输出	event_reset0-1	事件复位输出通道0 - 1 通过将每个位从OFF更改为ON来重置事件
	Z相模式输出	zmode0-1	Z相模式输出通道0 - 1 0=未使用、1=下一次、2=每次
	Z相逻辑输出	zlogic0-1	Z相逻辑输出通道0-1 0=正逻辑、1=负逻辑
	计算方向输出	dir0-1	计数方向输出通道0 - 1 0=倒数、1=上升计数
	相数输出(*2)	phase0-1	相数输出通道0 - 1 0=单相、1=2相、2=门控
	倍频输出(*2)	multi0-1	倍频输出0 - 1 0=1倍频、1=2倍频、2=4倍频
	清除输出 (* 2)	clr0-1	清除输出通道0 - 1 0=异步清除、1=同步清除
	数字滤波器输出 (* 3)	filter0-1	数字滤波器输出通道0 - 1
	单次脉冲宽度 (* 4) 输出	pulse0-1	单触发脉冲宽度输出通道0 - 1
	计数值输出	cnt_set0-1	计数值输出通道0 - 1
	计数比较值输出	cnt_match0-1	计数比较值输出通道0 - 1

※在通用输入/输出（GPIO）中，为每个输入/输出定义4位I / O映射，但实际可用的输入/输出由通用输入/输出方向参数确定。

*1 SSI数据格式

	开始地址								开始地址 + 1								开始地址+2								开始地址+3												
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0					
	Status								符号	MSB LSB																											
	传感器异常	Reserved						有效数据	4096°C 1°C ↓ ↓ 1/1024°C ↓																												
1°C		0	*	*	*	*	*		*	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				
-1 °C	0	*	*	*	*	*	*	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0					
-1/102°C	0	*	*	*	*	*	*	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1					
-999°C (断线时)	1	*	*	*	*	*	*	*	1	1	1	1	0	0	0	0	0	1	1	0	0	1	*	*	*	*	*	*	*	*	*	*					

※: 規定値無し

*2 相数 / 倍频 / 清除

可设定的脉冲信号类型（操作模式）
2相输入、同步清除、1 倍频模式
2相输入、同步清除、2 倍频模式
2相输入、同步清除、4 倍频模式
2相输入、非同步清除、1 倍频模式
2相输入、非同步清除、2 倍频模式
2相输入、非同步清除、4 倍频模式
单相输入、非同步清除、1 倍频模式
带门控的单相输入、非同步清除、1 倍频模式
带门控的单相输入、非同步清除、2 倍频模式

*3 数字滤波器

设定值	值
0h	0.1 μ sec
1h	6.52 μ sec
2h	25.7 μ sec
3h	32.1 μ sec
4h	204.9 μ sec
5h	211.3 μ sec
6h	230.5 μ sec
7h	236.9 μ sec
8h	819.3 μ sec
9h	825.7 μ sec
Ah	844.9 μ sec
Bh	851.3 μ sec
Ch	1024.1 μ sec
Dh	1030.5 μ sec
Eh	1049.7 μ sec
Fh	1056.1 μ sec

*4 一次脉冲宽度

脉冲宽度设定数据 $\times 409.6$ = 脉冲宽度 (μ sec)

3. 串口

产品型号	产品符号	CODESYS上的端口号
CPS-PC341MB-ADSC1-9201	COM A	端口1
	COM B	端口2
CPS-PCS341EC-DS1-1201 CPS-PCS341MB-DS1-1201	COM A	端口1
CPS-COM-1PC CPS-COM-1PD CPS-COM-2PC CPS-COM-2PD	1通道类型COM A. 2通道类型COMA, COM B.	串行通信模块的端口号按照从最靠近控制器的控制器的顺序分配为端口2, 端口3,。 (端口1用于控制器主机COM)

4. OPC UA符号设置

通过在CODESYS的IEC程序内使用的变量进行符号注册，实现OPC UA通信，或者在CONPROSYS HMIA进行监测。
可以从CODESYS自由设置符号，但以下符号设置为出厂默认值，并在打开电源后自动作为OPC UA服务器运行。

产品型号	I/O名	OPC UA符号名称	访问, 数据类型, 范围
CPS-PC341EC-1-9201	无	—	—
CPS-PC341MB-ADSC1-9201	数字输出位0	D00	Read/Write、BIT、0 or 1
	数字输出位1	D01	
	数字输入位0	DI0	Read、BIT、0 or 1
	数字输入位1	DI1	
	数字输入位2	DI2	
	数字输入位3	DI3	
	模拟输入0	AI0	Read、DWORD、0 - 4095
	模拟输入1	AI1	
CPS-PCS341EC-DS1-1201、 CPS-PCS341MB-DS1-1201	计数器输入0	CNT0	Read、DWORD、 0 - 16777215
	计数器输入1	CNT1	
	计数器清零0	CNTCLR0	Read/Write、BIT、0 or 1
	计数器清零1	CNTCLR1	
	数字输入位0	DI0	Read、BIT、0 or 1
	数字输入位1	DI1	
	数字输入位2	DI2	
	数字输入位3	DI3	

设置通信方法

说明PAC系列的通信方法和设置。

下表是可以与本章的各功能对应・连接的产品一览。

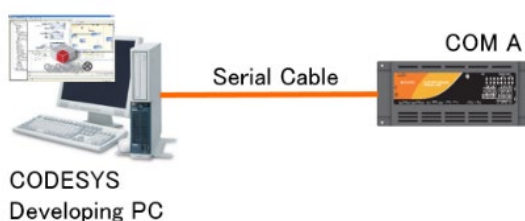
接线设定	对应的产品	可以作为Slave连接的产品
用控制器和pc进行串口通讯	CPS-PC341MB-ADSC1-9201 CPS-PCS341EC-DS1-1201 CPS-PCS341MB-DS1-1201	—
将本产品用作EtherCAT主站	CPS-PC341EC-1-9201 CPS-PCS341EC-DS1-1201	CPS-ECS341-1-011 其他公司制造商的Slave对应设备
将本产品用作Modbus TCP Master	CPS-PC341MB-ADSC1-9201 CPS-PCS341MB-DS1-1201	CPS-MC341-ADSC1-111等本公司对应的产品 其他公司制造商的Slave对应设备
将本产品用作Modbus TCP Slave	CPS-PC341MB-ADSC1-9201 CPS-PCS341MB-DS1-1201	—

1. 控制器与PC的串行通信

对应的产品	可以作为Slave连接的产品
CPS-PC341MB-ADSC1-9201 CPS-PCS341EC-DS1-1201 CPS-PCS341MB-DS1-1201	—

1. 串行通信的准备

用串行电缆连接控制器和PC，从控制器发送到PC的数据，从PC上接收到控制器的数据。使用控制器主机的COM端口A作为COM端口。

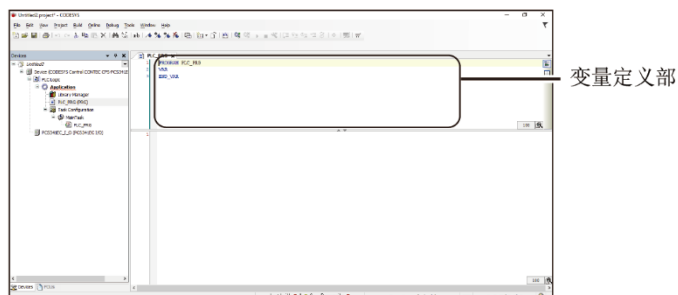


2. 串行通信库的设置

- 1 创建一个新的CODESYS项目。
IEC程序语言选择ST。
※ 有关如何创建项目的详细信息，请参阅“**新建项目的生成（P56）**”。
- 2 在设备窗口中双击[Library Manager]。
- 3 单击“Add Library”，在“Use Cases”树层次结构中选择“Serial Communication”，然后单击“OK”。

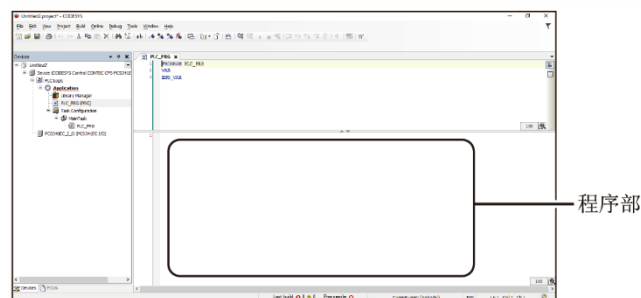
3. 创建和执行发送程序

- 1 双击设备窗口中的[PLC_PRG (PRG)]图标。
将显示“ST编辑器窗口”。
- 2 在“ST编辑器窗口”的“变量定义部”的VAR和END_VAR之间写下以下源代码。



```
StartFlag : BOOL := TRUE;  
State : INT := 0;  
ComlParams : ARRAY [1..7] OF COM.PARAMETER;  
ComlOpen : COM.Open;  
ComlClose : COM.Close;  
ComlWrite : COM.Write;  
Error : BOOL;  
TestString : STRING := 'Test String!';
```

- 3 在“ST Editor窗口”的“程序部”中编写以下源代码。



```
IF StartFlag THEN
CASE State OF
0:
  Com1Params[1].udiParameterId := COM.CAA_Parameter_Constants.udiPort;
  Com1Params[1].udiValue := 3;
  Com1Params[2].udiParameterId := COM.CAA_Parameter_Constants.udiBaudrate;
  Com1Params[2].udiValue := 9600;
  Com1Params[3].udiParameterId :=
    INT_TO_UDINT(COM.CAA_Parameter_Constants.udiParity);
  Com1Params[3].udiValue := INT_TO_UDINT(COM.PARITY.NONE);
  Com1Params[4].udiParameterId := COM.CAA_Parameter_Constants.udiStopBits;
  Com1Params[4].udiValue := INT_TO_UDINT(COM.STOPBIT.ONESTOPBIT);
  Com1Params[5].udiParameterId := COM.CAA_Parameter_Constants.udiTimeout;
  Com1Params[5].udiValue := 0;
  Com1Params[6].udiParameterId := COM.CAA_Parameter_Constants.udiByteSize;
  Com1Params[6].udiValue := 8;
  Com1Params[7].udiParameterId := COM.CAA_Parameter_Constants.udiBinary;
  Com1Params[7].udiValue := 0;
  Com1Open(xExecute:= TRUE, usiListLength:= SIZEOF(Com1Params) /
    SIZEOF(COM.PARAMETER), pParameterList:= ADR(Com1Params));
  IF Com1Open.xError THEN
    Error := TRUE;
    State := 1000;
  END_IF
  IF Com1Open.xDone THEN
    State := 1;
  END_IF
1:
  Com1Write(xExecute := TRUE, hCom:= Com1Open.hCom, pBuffer:=
    ADR(TestString), szSize:= SIZEOF(TestString));
  IF Com1Write.xError THEN
    Error := TRUE;
    State := 1000;
  END_IF
  IF Com1Write.xDone THEN
    State := 2;
  END_IF
2:
  Com1Close(xExecute := TRUE, hCom:= Com1Open.hCom);
  IF Com1Close.xError THEN
    Error := TRUE;
  END_IF
  IF Com1Close.xDone OR Com1Close.xError THEN
    State := 1000;
  END_IF
1000:
  StartFlag := FALSE;
END_CASE
END_IF
```

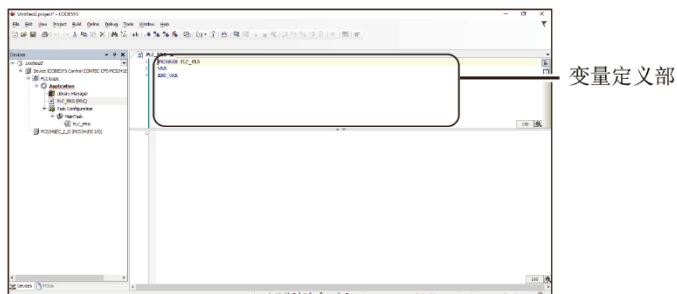
4 从CODESYS的[Build]菜单执行[Rebuild]，检查构建过程是否正常完成。

◆ 字符列发送

- 1 从CODESYS的[在线]菜单执行[登录]。
- 2 将显示下载确认对话框。单击[是]。
- 3 将终端连接到PC端的串行端口并等待接收。
- 4 在CODESYS的[Debug]菜单上执行[Run]，程序操作将启动。在PC侧终端屏幕上显示字符串“Test String! ”。

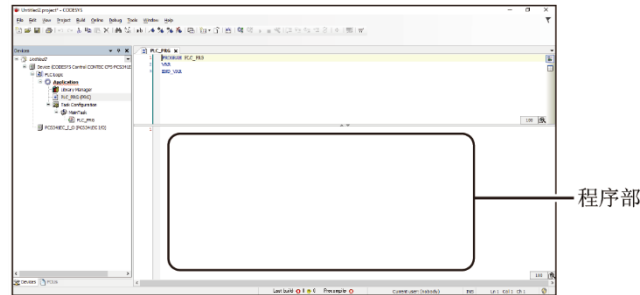
4. 接收程序的制作和执行

- 1 为接收程序创建一个新的CODESYS项目。
IEC程序语言选择ST。
※有关如何创建项目的详细信息，请参阅“新建项目的生成（P56）”。
- 2 在设备窗口中双击[PLC_PRG（PRG）]图标。
将显示“ST编辑器窗口”。
- 3 在“ST编辑器窗口”的“变量定义部”的VAR和END_VAR之间编写以下源代码。



```
StartFlag : BOOL := TRUE;  
State : INT := 0;  
Com1Params : ARRAY [1..7] OF COM. PARAMETER;  
Com1Open : COM. Open;  
Com1Close : COM. Close;  
Com1Read : COM. Read;  
Error : BOOL;  
TestString : STRING(25);
```

4 在“ST Editor窗口”的“程序部”中编写以下源代码。

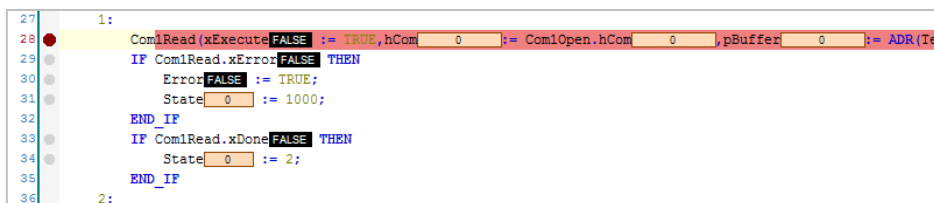


```
IF StartFlag THEN
CASE State OF
  0:
    Com1Params[1].udiParameterId := COM.CAA_Parameter_Constants.udiPort;
    Com1Params[1].udiValue := 1;
    Com1Params[2].udiParameterId := COM.CAA_Parameter_Constants.udiBaudrate;
    Com1Params[2].udiValue := 9600;
    Com1Params[3].udiParameterId := COM.CAA_Parameter_Constants.udiParity;
    Com1Params[3].udiValue := INT_TO_UDINT(COM.PARITY.NONE);
    Com1Params[4].udiParameterId := COM.CAA_Parameter_Constants.udiStopBits;
    Com1Params[4].udiValue := INT_TO_UDINT(COM.STOPBIT.ONESTOPBIT);
    Com1Params[5].udiParameterId := COM.CAA_Parameter_Constants.udiTimeout;
    Com1Params[5].udiValue := 0;
    Com1Params[6].udiParameterId := COM.CAA_Parameter_Constants.udiByteSize;
    Com1Params[6].udiValue := 8;
    Com1Params[7].udiParameterId := COM.CAA_Parameter_Constants.udiBinary;
    Com1Params[7].udiValue:= 0;
    Com1Open(xExecute:= TRUE, usiListLength:= SIZEOF(Com1Params) /
              SIZEOF(COM.PARAMETER), pParameterList:= ADR(Com1Params));
    IF Com1Open.xError THEN
      Error := TRUE;
      State := 1000;
    END_IF
    IF Com1Open.xDone THEN
      State := 1;
    END_IF
  1:
    Com1Read(xExecute := TRUE, hCom:= Com1Open.hCom, pBuffer:=
              ADR(TestString), szBuffer:= SIZEOF(TestString));
    IF Com1Read.xError THEN
      Error := TRUE;
      State := 1000;
    END_IF
    IF Com1Read.xDone THEN
      State := 2;
    END_IF
  2:
    Com1Close(xExecute := TRUE, hCom:= Com1Open.hCom);
    IF Com1Close.xError THEN
      Error := TRUE;
    END_IF
    IF Com1Close.xDone OR Com1Close.xError THEN
      State := 1000;
```

```
END_IF
1000:
    StartFlag := FALSE;
END_CASE
END_IF
```

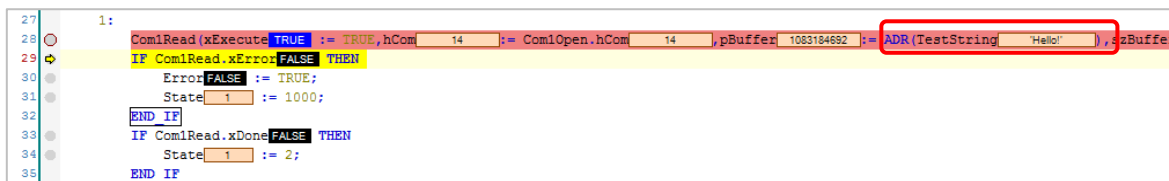
◆ 字符列发送

- 1 从CODESYS的[在线]菜单执行[登录]。
- 2 将显示下载确认对话框。单击[是]。
- 3 终端连接到PC端的串口并等待发送。
- 4 在程序源代码的接收功能部分插入断点。在调用Com1Read函数的行上，选择[Debug] - [New Breakpoint]。



```
27 1:
28 Com1Read(xExecute FALSE := TRUE, hCom 0 := Com1Open.hCom 0, pBuffer 0 := ADR(Te
29 IF Com1Read.xError FALSE THEN
30 Error FALSE := TRUE;
31 State 0 := 1000;
32 END_IF
33 IF Com1Read.xDone FALSE THEN
34 State 0 := 2;
35 END_IF
36 2:
```

- 5 执行[Debug]菜单上的[Run]开始程序操作。该程序在断点位置暂停。
- 6 从PC终端输入发送字符串。
- 7 在CODESYS端，按F10键并执行程序一步。可以确认从PC终端发送的字符串存储在TestString变量中。



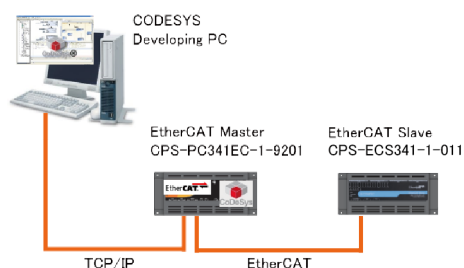
```
27 1:
28 Com1Read(xExecute TRUE := TRUE, hCom 14 := Com1Open.hCom 14, pBuffer 1083184682 := ADR(TestString 'Hello!'), szBuffer
29 IF Com1Read.xError FALSE THEN
30 Error FALSE := TRUE;
31 State 1 := 1000;
32 END_IF
33 IF Com1Read.xDone FALSE THEN
34 State 1 := 2;
35 END_IF
```

2. 用作EtherCAT主站

对应的产品	可以作为Slave连接的产品
CPS-PC341EC-1-9201 CPS-PCS341EC-DS1-1201	CPS-ECS341-1-011 其他公司制造商的Slave对应设备

1. 设备的准备

作为EtherCAT设备，说明使用本公司的CPS-ECS341-1-011的CODESYS编程示例。



- 1 使用以太网电缆将CPS-PC341EC-1-9201的LAN A连接到个人计算机的LAN端口。
- 2 使用以太网电缆连接CPS-PC341EC-1-9201的LAN B和CPS-ECS 341-1-011的IN端口。

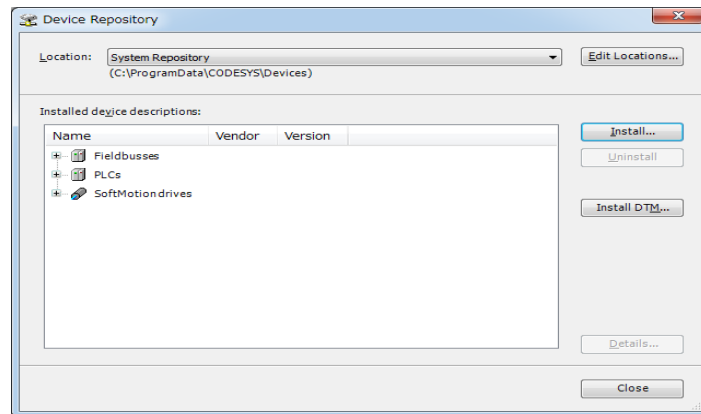
2. 添加从设备

要使用EtherCAT从控制器，必须注册ESI（EtherCAT Slave Information）文件。

使用我们公司的从控制器时，不需要注册1到3，因为ESI文件包含在CODESYS包中。要使用其他公司制造的从属设备，请执行注册程序1至3。

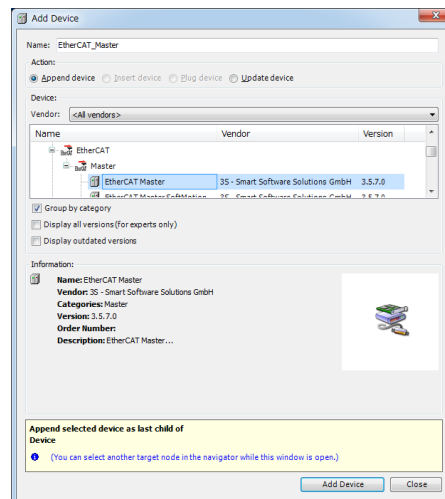
- 1 创建一个新的CODESYS项目。
选择设备连接的EtherCAT型号的名称，IEC程序语言选择ST。
※ 有关如何创建项目的详细信息，请参阅“**新建项目的生成（P56）**”。
- 2 从使用ESI文件的设备制造商下载。
- 3 从CODESYS [Tool]菜单中选择[Device Repository]。

- 4 单击[安装]并指定ESI文件“CONTEC_CPS-ECS341-111_v100”（对于第三方设备，获取的ESI文件）。



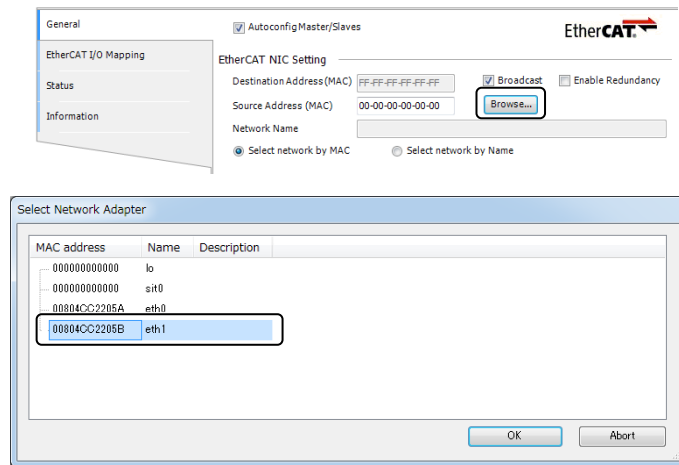
- 5 在CODESYS设备窗口中右键单击[设备（CODESYS Control CONTEC CPS-PC341EC-1-9201）]图标，然后选择[添加设备]。

- 6 在“添加设备”对话框中，选择[EtherCAT主站]，然后单击[添加设备]按钮。[EtherCAT Master]在[Fieldbus - EtherCAT - Master]的树层次结构中注册。



- 7 右键单击添加到设备窗口的[EtherCAT_Master]图标，然后选择[添加设备]。
- 8 在添加设备对话框中，选择[CPS-ECS341-011]或要使用的设备名称，然后单击[添加设备]按钮。“CPS-ECS341-1011”在“现场总线 - EtherCAT - Slave - CONTEC Co., Ltd. - CONPROSYS EtherCAT”的树状层次中注册。
- 9 双击设备窗口中的[EtherCAT_Master (EtherCAT Master)]图标以打开设置窗口。

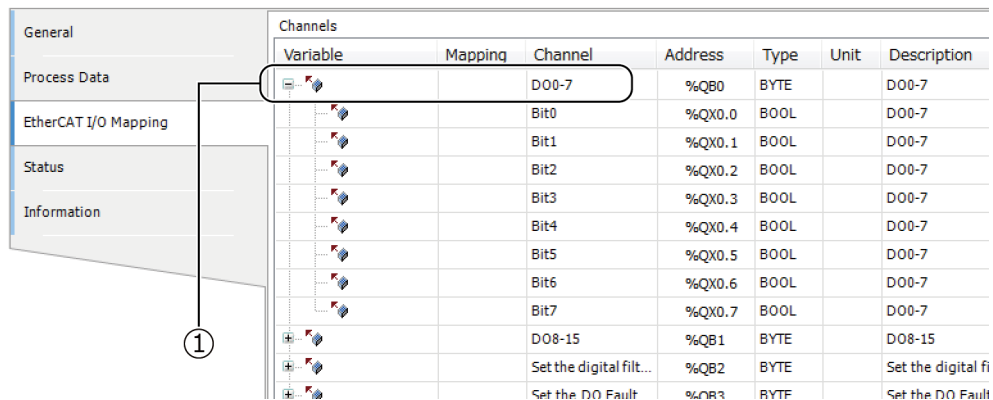
10 单击[常规]选项上的[Browse]按钮，然后选择[eth1]。



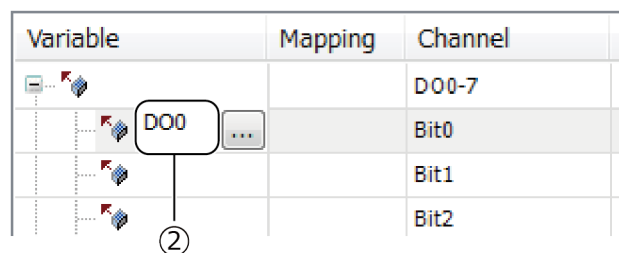
3. 从设备设置

描述EtherCAT从设备的设置，以连续打ON/OFF CPS-DIO-0808L的输出端口2位0。

- 1 双击设备窗口中的[CPS-DIO-0808L]图标。
- 2 显示CPS-DIO-0808L的配置窗口时，选择[EtherCAT I / O Mapping]。
- 3 ①在“通道”列中打开“D0 0 - 7”行树。



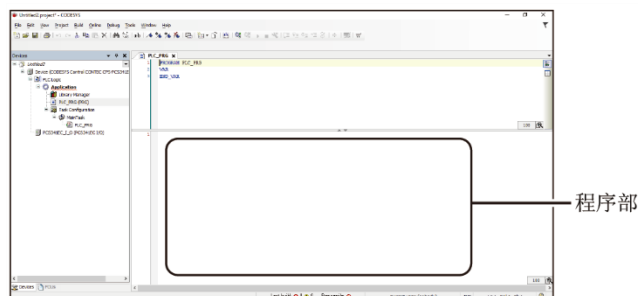
- 4 “通道”列为“位0”双击②“变量”列的空白部分以定义“D00”变量。



4. 创建和构建程序

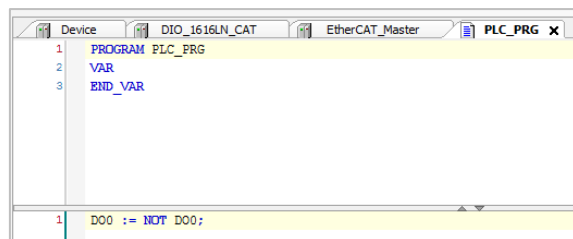
程序1和程序2是用于连续打开/关闭CPS-DIO-0808L的输出端口2位0的程序，程序3和4是用于目视检查LED闪烁的周期时间设置。

- 1 双击设备窗口中的[PLC_PRG (PRG)]图标。
将显示“ST编辑器窗口”。
- 2 在“ST Editor窗口”的“程序部”中写下以下源代码。

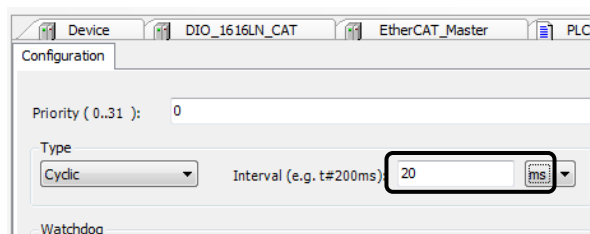


```
D00 := NOT D00;
```

上述程序在每个周期重复地将“1”和“0”设置为“D0 0”变量。



- 3 双击设备窗口中的[MainTask]图标。
- 4 将“周期”从“4000 μ s”更改为“20 ms”。

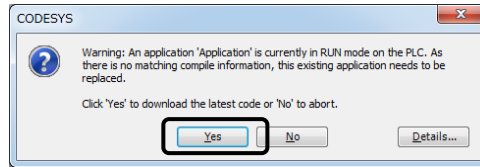


- 5 从CODESYS的[Build]菜单执行[Rebuild]，检查构建过程是否正常完成。

5. 下载并执行程序

1 从CODESYS的[在线]菜单执行[登录]。

2 将显示下载确认对话框。单击[是]。



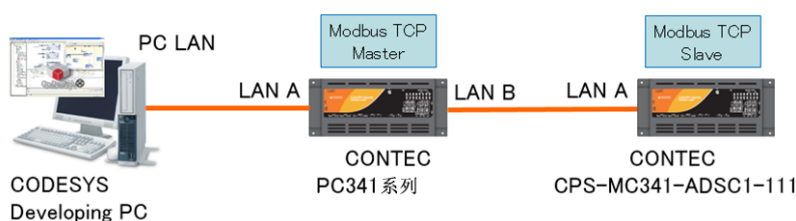
3 执行[Debug]菜单中的[Run]时，EtherCAT操作开始。
通过CPS-DIO-0808L的位0 LED闪烁可以确认。

3. 用作Modbus TCP Master

对应的产品	可以作为Slave连接的设备
CPS-PC341MB-ADSC1-9201 CPS-PCS341MB-DS1-1201	CPS-MC341-ADSC1-111等本公司对应的产品 其他公司制造商的Slave对应设备

1. 设备的准备

将控制器作为Modbus TCP主设备运行，并使用Modbus TCP从设备执行数据输入/输出。在本例中，我们将CPS-PC341MB-ADSC1-9201用作Modbus TCP主站，将CONTEC CPS-MC341-ADSC1-111用作Modbus TCP从站。

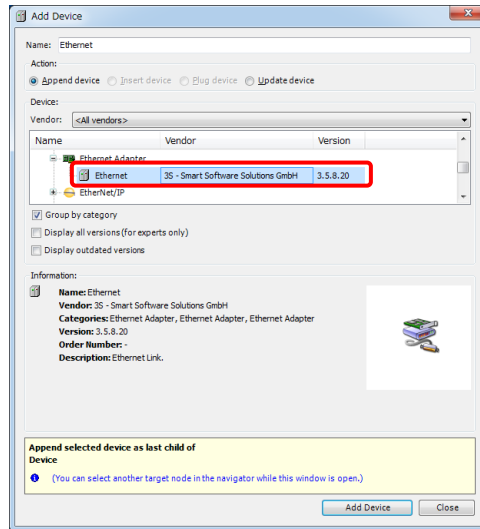


- 1 使用CODESYS开发环境将控制器的LAN A连接到计算机的LAN端口。
- 2 将控制器的LAN B连接到CONTEC CPS-MC341-ADSC1-111。

2. 添加从设备

- 1 创建一个新的CODESYS项目。
选择设备要连接的Modbus型号的名称，IEC程序语言选择ST。
※ 有关如何创建项目的详细信息，请参阅“**新建项目的生成 (P56)**”。
- 2 在设备窗口中右键单击控制器[设备 (CODESYS CONTEC CONTEC CPS - PCXXXXXXXXXX)]，然后选择[添加设备]。

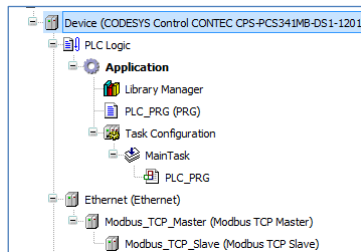
- 3 在Add Device对话框中，选择[Fieldbus - Ethernet Adapter - Ethernet]并单击[Add Device]按钮。



- 4 在设备窗口中右键单击[Ethernet]，然后选择[添加设备]。

- 5 在“添加设备”对话框中，添加[现场总线 - Modbus - Modbus TCP主站 - Modbus TCP主站]。

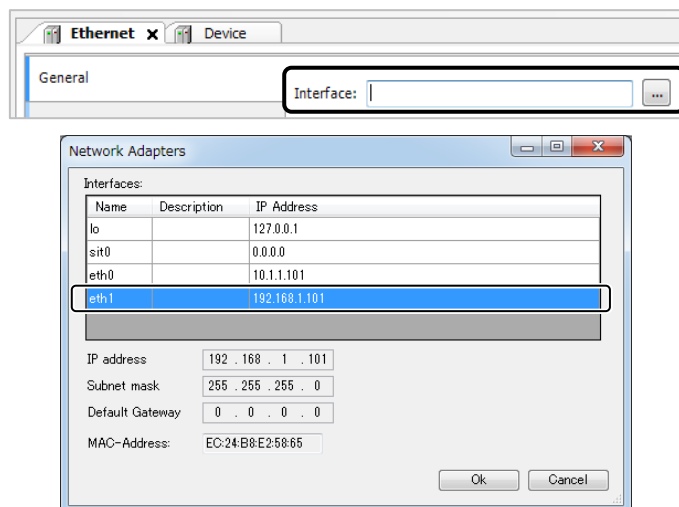
- 6 在设备窗口中右键单击Modbus TCP Master，选择添加设备，然后添加[现场总线 - Modbus - Modbus TCP从站 - Modbus TCP从站]。



- 7 从设备窗口双击[Ethernet]图标，打开设备配置窗口。

- 8 打开“常规”选项。

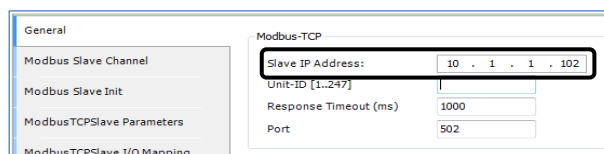
- 9** 单击“界面”文本框旁边的按钮，然后从列表中选择[eth1]。eth1将是LAN B的内部名称。



- 10** 在设备窗口中，双击[Modbus_TCP_Slave]图标以打开设备配置窗口。

- 11** 打开[Modbus主站设置]选项。

- 12** 在从属设备上配置的IP地址输入“从属IP地址”。



3. Modbus TCP Master的设置

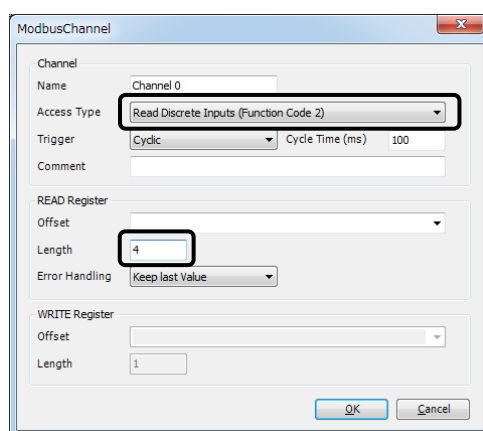
在这里，我们创建一个执行2位数字输出和4位数字输入的示例。

要使用的从设备CPS-MC341-ADSC1-111的功能代码定义如下。

名称/功能代码	地址	数据
Read Coils(代码1)	0	数字输出位0
	1	数字输出位1
Read Discrete Inputs(代码2)	0	数字输入位0
	1	数字输入位1
	2	数字输入位2
	3	数字输入位3
Read Input Registers(代码4)	0	模拟输入通道0(16位)
	1	模拟输入通道1(16位)
	2	计数器输入通道0上位16位
	3	计数器输入通道0下位16位
	4	计数器输入通道1上位16位
	5	计数器输入通道1下位16位
Write Single Coil(代码5)	0	数字输出位0
	1	数字输出位1

※ 作为样品制作的2位数字输出，Writsingle Coil1 (Code 5)，4位的数字输入使用Raddree Inputs (代码2)。

- 1 在设备窗口中，双击[Modbus_TCP_Slave]图标以打开设备配置窗口。
- 2 打开[Modbus Slave Channel]选项。
- 3 单击[Add Channel]按钮，从访问类型中选择[Read Discrete Inputs (功能代码2)]，在Rader寄存器长度输入“4”后单击[OK]。



4 单击[Add Channel]按钮，从访问类型中选择[Write Single Coil（功能代码5）]，在Write寄存器偏移中输入“0”并单击[OK]。

5 单击[Add Channel]按钮，从访问类型中再次选择[Write Single Coil]，在Write寄存器偏移中输入“1”并单击[OK]。

Name	Access Type	Trigger	READ Offset	Length	Error Handling	WRITE Offset
Channel 0	Read Discrete Inputs (Function Code 02)	Cyclic, t#100ms	16#0004	1	Keep last Value	
Channel 1	Write Single Coil (Function Code 05)	Cyclic, t#100ms				16#0000
Channel 2	Write Single Coil (Function Code 05)	Cyclic, t#100ms				16#0001

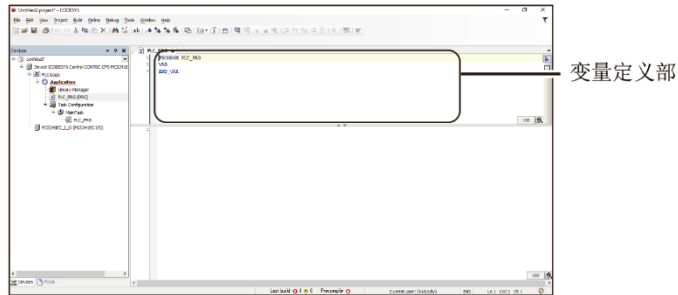
6 单击 [Modbus TCP Slave I/O Mapping]。

7 将“DI 0”，“DOBIT 0”，“DOBIT 1”分配给Channel0[0]、Channel1[0]、Channel2[1]。

Variable	Mapping	Channel	Address	Type	Unit	Description
		Channel 0	%IB0	ARRAY [0..0] OF BYTE		Read Discrete Inputs
DI0		Channel 0[0]	%IB0	BYTE		Read Discrete Inputs
DOBIT0		Channel 1[0]	%QB0	BYTE		Write Single Coil
DOBIT1		Channel 2[0]	%QB1	BYTE		Write Single Coil

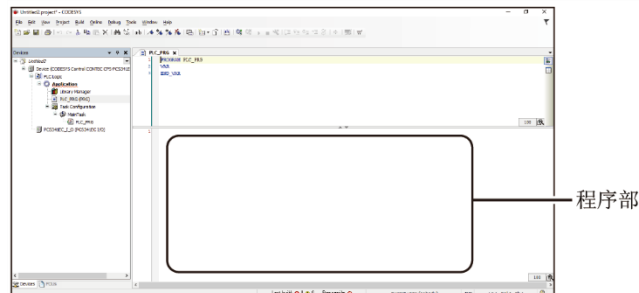
4. 创建和构建程序

- 1 双击设备窗口中的[PLC_PRG (PRG)]图标。
将显示“ST编辑器窗口”。
- 2 在“ST编辑器窗口”的“变量定义部”的VAR和END_VAR之间编写以下源代码。



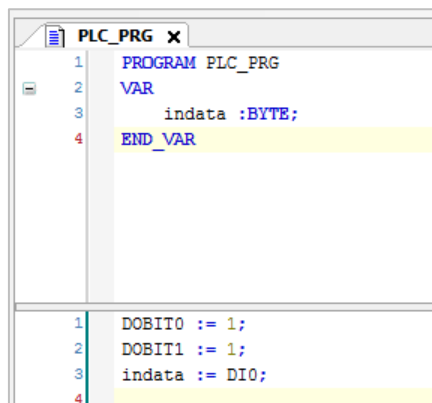
```
indata :BYTE
```

- 3 在“ST Editor窗口”的“程序部”中编写以下源代码。



```
DOBIT0 := 1;  
DOBIT1 := 1;  
indata := DI0;
```

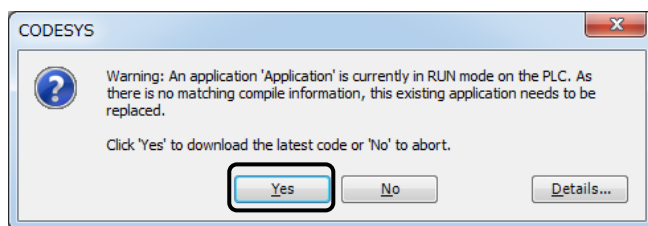
上述的程序是，在CPS-PC341MB-ADSC1-9201的数字输出位0，向1输出ON信号，从数字输入位0输入信号。



- 4 从CODESYS的[Build]菜单执行[Rebuild]，检查构建过程是否正常完成。

5. 下载并执行程序

- 1 从CODESYS的[在线]菜单执行[登录]。
- 2 将显示下载确认对话框。单击[是]。



- 3 如果在[Debug]菜单上运行[Run]，程序操作将开始。 可以通过在CPS-PC341MB-ADSC1-9201的前面板上D0位0和11的LED亮起来确认。

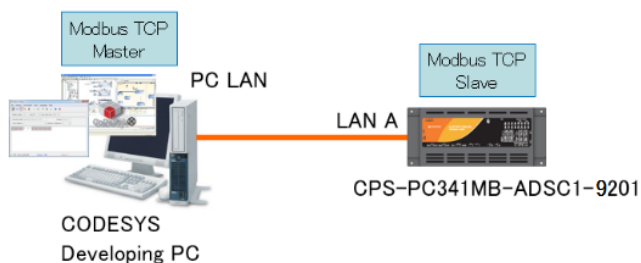
4. 用作Modbus TCP Slave

对应的产品	要连接的设备
CPS-PC341MB-ADSC1-9201 CPS-PCS341MB-DS1-1201	—

1. 设备的准备

控制器作为Modbus TCP Slave设备操作，与来自Modbus TCP Master的数据输入对应。在本例中，我们将使用CPS-PC341MB-ADSC1-9201作为Modbus TCP Slave解释CODESYS编程示例，其中CODESYS开发环境安装为Modbus TCP Master。

使用QModMaster作为Modbus TCP Master软件。



- 1 将Modbus TCP Master软件QModMaster安装在电脑上。
QModMaster是免费软件，可以从下面下载。

下载

<http://sourceforge.net/projects/qmodmaster/>

- 2 用以太网电缆CPS-PC341MB-ADSC1-9201的LAN A与CODESYS开发环境电脑的LAN端口连接。

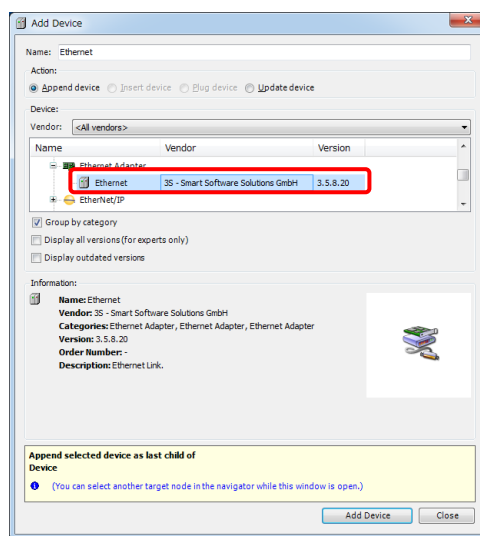
2. 进行从属设置

- 1 创建一个新的CODESYS项目。
IEC程序语言选择ST。

※ 有关如何创建项目的详细信息，请参阅“新建项目的生成（P56）”。

- 2 右键单击设备窗口的控制器[设备（CODESYS CONTROL CONTEC CPS - PCXXXXXXXXXX）]并选择[添加设备]。

- 3 在Add Device对话框中，选择[Fieldbus - Ethernet Adapter - Ethernet]并单击[Add Device]按钮。
※ EthernetAdapter将被添加。



- 4 在设备窗口中右键单击[Ethernet]，然后选择[添加设备]。

- 5 在添加设备对话框中，添加[现场总线 - Modbus - Modbus TCP从站设备 - ModbusTCP Slave Device]。
※ 请在Ethernet Adapter设备下添加Modbus TCP Slave设备。

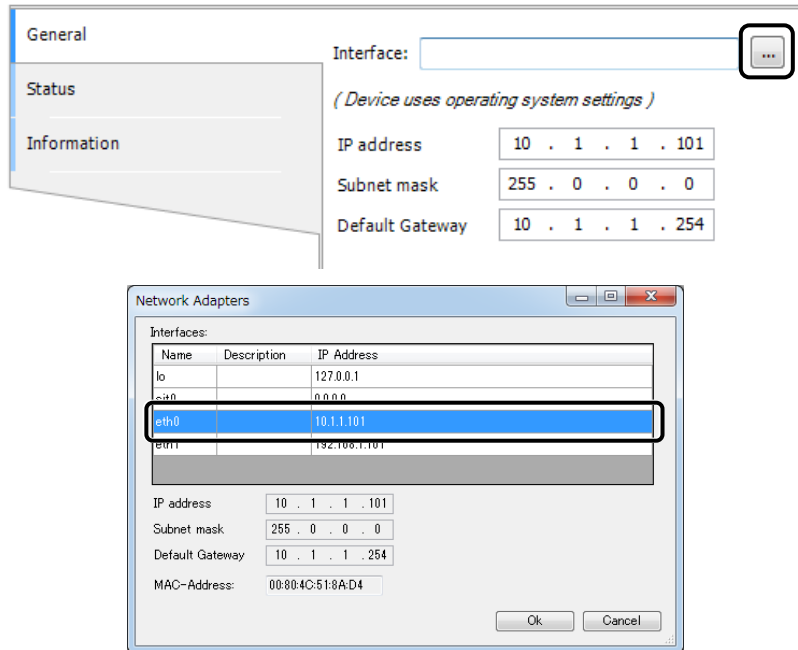
- 6 从设备窗口，点击[Ethernet]图标打开设备配置窗口。

- 7 打开“常规”选项。

8 单击[Interface]文本框旁边的按钮，然后从列表中选择[eth0]。

※对Ethernet Adapter设置IP地址。

※eth0是LAN A的内部名称。



9 双击设备窗口中的[PC341MB_I_0]图标以打开设备配置窗口。

10 单击 [Internal I/O Mapping]。

11 在“设备”窗口中，双击 [Ethernet] 图标以打开“设备配置”窗口。

12 如下定义变量。

变量（输入值）		通道	地址
		do0-1	
D00		Bit0	%QX20.0
D01		Bit1	%QX20.1
		Di0-3	
DI0		Bit0	%IX20.0
DI1		Bit1	%IX20.1
DI2		Bit2	%IX20.2
DI3		Bit3	%IX20.3
CNT0		cnt0	%ID6
CNT1		cnt1	%ID7
AI0		ai0	%ID8
AI1		ai1	%ID9

13 双击设备窗口中的[ModbusTCP_Slave_Device]图标以打开设备配置窗口。

14 单击 [Modbus TCP Slave Device I/O Mapping]。

15 如下定义变量。

变量（输入值）	通道	地址
	Input	%IW1
	Input[0]	%IW1
mD00	Bit0	%IX2. 0
mD01	Bit1	%IX2. 1
	:	
	Bit15	%IX3. 7
	Input[1]	%IW2
	:	
	Input[9]	%IW10
	Output	%QW1
	Output[0]	%QW1
mDI0	Bit0	%QX0. 0
mDI1	Bit1	%QX0. 1
mDI2	Bit2	%QX0. 2
mDI3	Bit3	%QX0. 3
	:	
	Bit15	%QX1. 7
mAI0	Output[1]	%QW2
MAI1	Output[2]	%QW3
mCNT0_L	Output[3]	%QW4
mCNT0_H	Output[4]	%QW5
mCNT1_L	Output[5]	%QW6
mCNT1_H	Output[6]	%QW7
	:	
	Output[9]	%QW10

※ I / O映射设置，Modbus Slave的I / O映射定义已完成。 通过在Modbus Slave的I / O映射中定义变量，可以与Modbus Master进行数据通信。

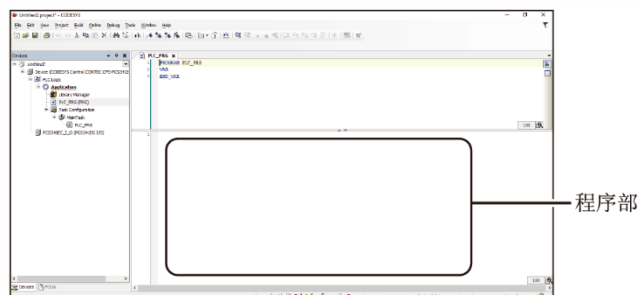
※ 定义在Inputs映射区域输出变量，在Outputs映射区域输入变量，是为了Inputs映射区域从Master到Slave的输入，Outputs映射区域从Slave到Master的输出。

※ 计数器数据是24位不适合Modbus的WORD区域，所以将下位16位存储到mCNT0_L，将上位8位存储到mCNT0_H。

Internal Parameters		Find	Filter	Show all				
Internal I/O Mapping		Variable	Mapping	Channel	Address	Type	Unit	Description
Status	Information	mDO0		Bit0	%QB20	BYTE		digital output bit 0-1
		mDO1		Bit1	%QX20.0	BOOL		digital output bit 0-1
				Bit2	%QX20.1	BOOL		digital output bit 0-1
				Bit3	%QX20.2	BOOL		digital output bit 0-1
					%QX20.3	BOOL		digital output bit 0-1
					%QX20.4	BOOL		digital output bit 0-1
				Bit5	%QX20.5	BOOL		digital output bit 0-1
				Bit6	%QX20.6	BOOL		digital output bit 0-1
				Bit7	%QX20.7	BOOL		digital output bit 0-1
				di0-3	%IB20	BYTE		digital input bit 0-3
		mDI0		Bit0	%IX20.0	BOOL		digital input bit 0-3
		mDI1		Bit1	%IX20.1	BOOL		digital input bit 0-3
		mDI2		Bit2	%IX20.2	BOOL		digital input bit 0-3
		mDI3		Bit3	%IX20.3	BOOL		digital input bit 0-3
					%IX20.4	BOOL		digital input bit 0-3
				Bit5	%IX20.5	BOOL		digital input bit 0-3
				Bit6	%IX20.6	BOOL		digital input bit 0-3
				Bit7	%IX20.7	BOOL		digital input bit 0-3
		mCNT0		cnt0	%ID6	DWORD		counter channel 0
		mCNT1		cnt1	%ID7	DWORD		counter channel 1
		mAI0		ai0	%ID8	DWORD		analog input channel 0
		mAI1		ai1	%ID9	DWORD		analog input channel 1
				cnt0	%ID6	DWORD		counter channel 0
				cnt1	%ID7	DWORD		counter channel 1
				ai0	%ID8	DWORD		analog input channel 0
				ai1	%ID9	DWORD		analog input channel 1
				battery	%ID10	BYTE		battery status

3. 创建和构建程序

- 1 双击设备窗口中的[PLC_PRG (PRG)]图标。
将显示“ST编辑器窗口”。
- 2 在“ST Editor窗口”的“程序部”中写下以下源代码。



```
mDI0 := DI0;  
mDI1 := DI1;  
mDI2 := DI2;  
mDI3 := DI3;  
mAIO := DWORD_TO_WORD(AIO);  
mAI1 := DWORD_TO_WORD(AI1);  
mCNT0_L := DWORD_TO_WORD(CNT0 AND 16#0000FFFF);  
mCNT0_H := DWORD_TO_WORD(SHR(CNT0, 16) AND 16#0000FFFF);  
  
mCNT1_L := DWORD_TO_WORD(CNT1 AND 16#0000FFFF);  
mCNT1_H := DWORD_TO_WORD(SHR(CNT1, 16) AND 16#0000FFFF);  
DO0 := mDO0;  
DO1 := mDO1
```

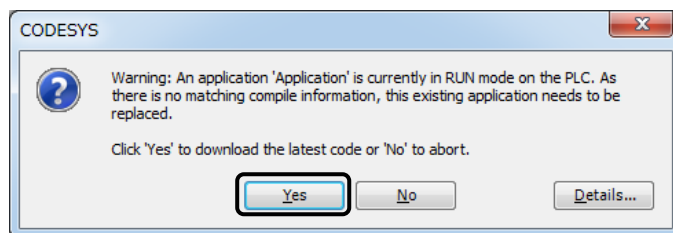
※ 将数字输入，模拟输入，计数器输入数据从I / O变量复制到IEC程序中的Modbus变量。 在数字输出数据的情况下，内容从Modbus变量复制到I / O变量。

4. 下载并执行程序

在步骤1到10中，显示了数字输出位0中输出1的示例，并且在步骤11和12中，解释了输入计数器输入0数据的示例。

1 从CODESYS的[在线]菜单执行[登录]。

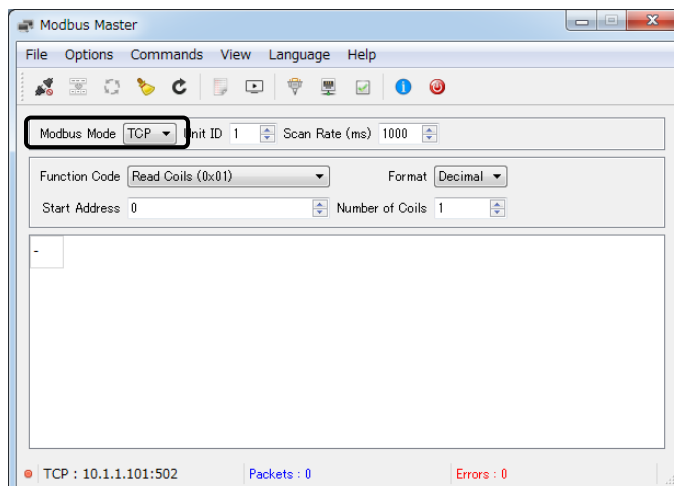
2 将显示下载确认对话框。单击[是]。



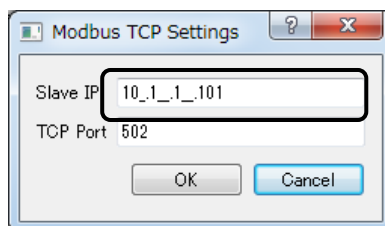
3 从[Debug]菜单执行[Operation]时，程序操作开始。

4 在步骤3的状态下启动QModMaster。

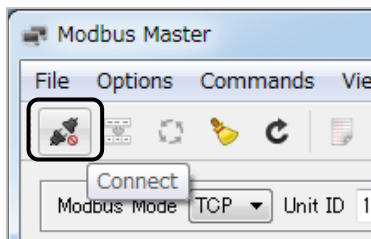
5 从“Modbus Mode”中选择“TCP”。



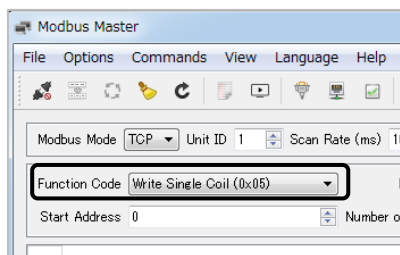
6 从“Options”菜单中选择“Modbus TCP”，然后设置从站设备的IP地址“10.1.1.101”。



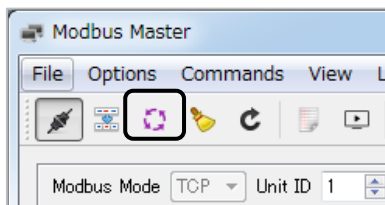
7 单击“Connect”图标。



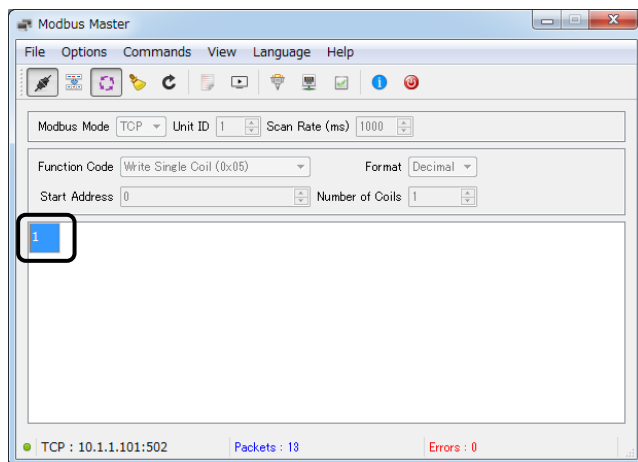
8 从“Function Code”中选择“Write Single Coil(0x05)”。



9 将“Start Address”设置为“0”，然后单击“Scan”图标。

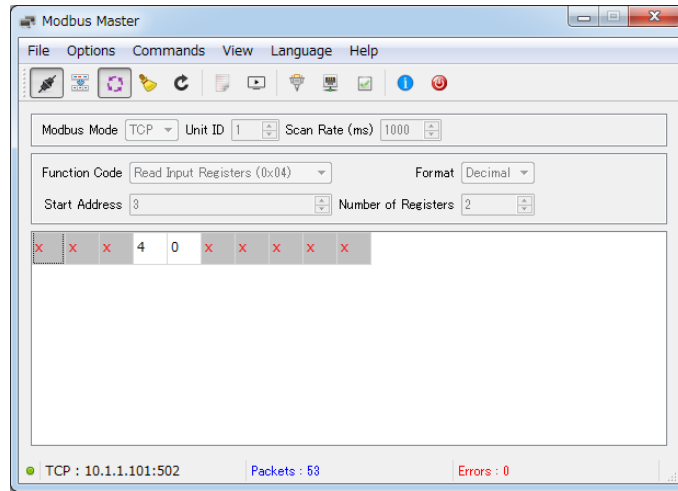


10 双击下面屏幕上“0”的部分，将其更改为“1”。 CONTEC CPS的D00 LED - PC341MB - ADSC1 - 9201 亮起。



11 从“Function Code”中选择[Read Input Registers(0x04)]。

- 12** 在“Start Address”中输入“3”，在“Number of Registers”中输入“2”，然后单击[Scan]图标。计数器0的低16位显示在左侧，高8位显示在右侧。



5. 设置OPC UA Server

CODESYS添加了OPC UA Server功能。

1. 设备的准备

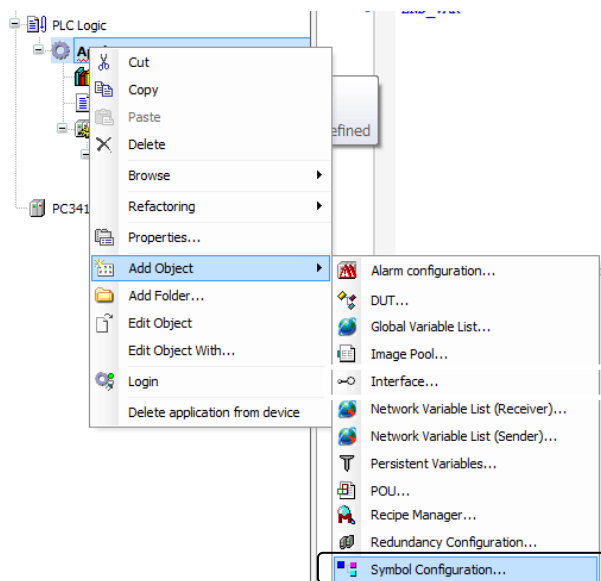
- 1 使用CODESYS打开在“基本编程顺序（P56）”中创建的项目。
- 2 由Unified Automation提供的UaExpert作为OPC UA客户端软件。
UaExpert可从以下URL免费下载。（需要用户注册）

下载

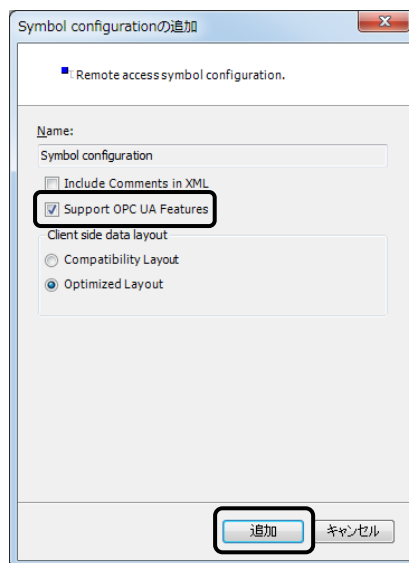
<https://www.unified-automation.com/downloads/opc-ua-clients.html>

2. CODESYS的设定

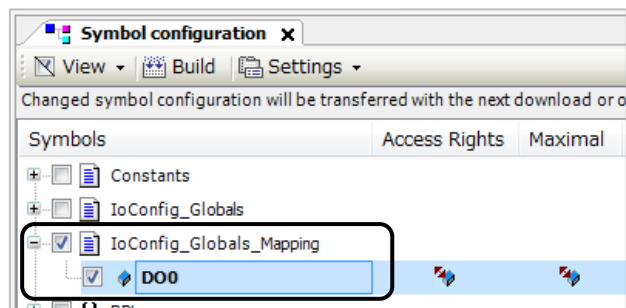
- 1 右键单击设备窗口中显示的[Application]，然后选择[添加对象] - [Symbol configuration]。



- 2 选中“添加Symbol configuration”对话框中的“Support OPC UA Features”，然后单击“添加”按钮。



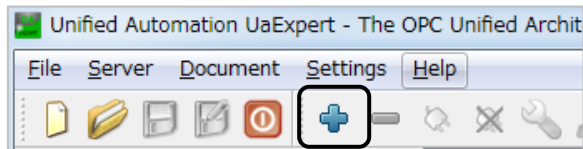
- 3 将显示“Symbol configuration”窗口，从列表中选择要用于OPC UA Server的变量。在这种情况下，检查“IoConfig_Global_Mapping”下的“DO 0”变量。



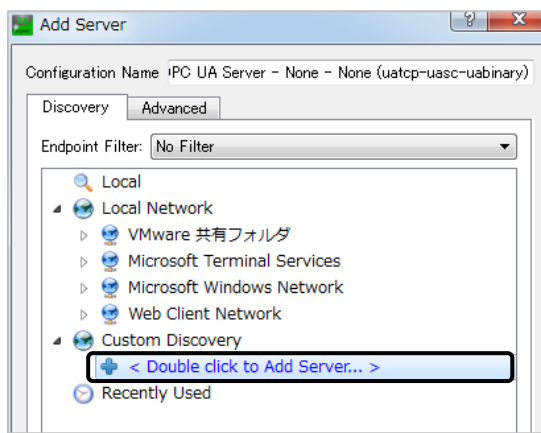
- 4 构建程序，确认没有发生错误等，登录控制器并开始操作。

3. OPC UA Clientの設定

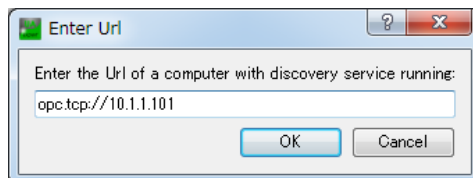
- 1 启动OPC UA客户端软件UaExpert，然后单击+图标以添加服务器。



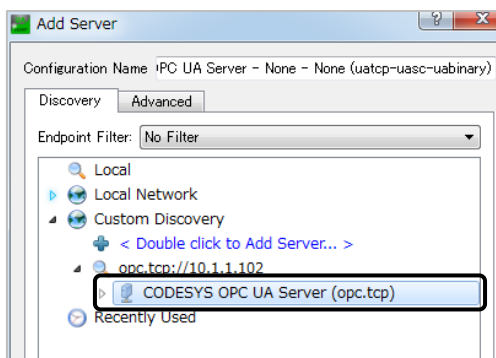
- 2 双击 [Double click to Add Server...]。



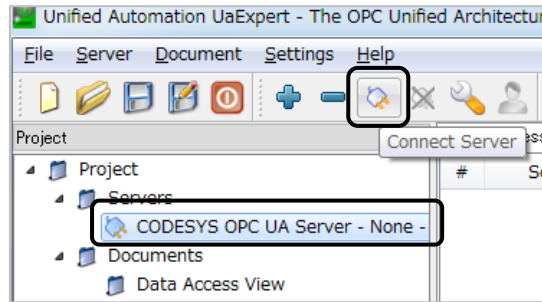
- 3 在URL中输入控制器的IP地址“opc.tcp://10.1.1.101”。



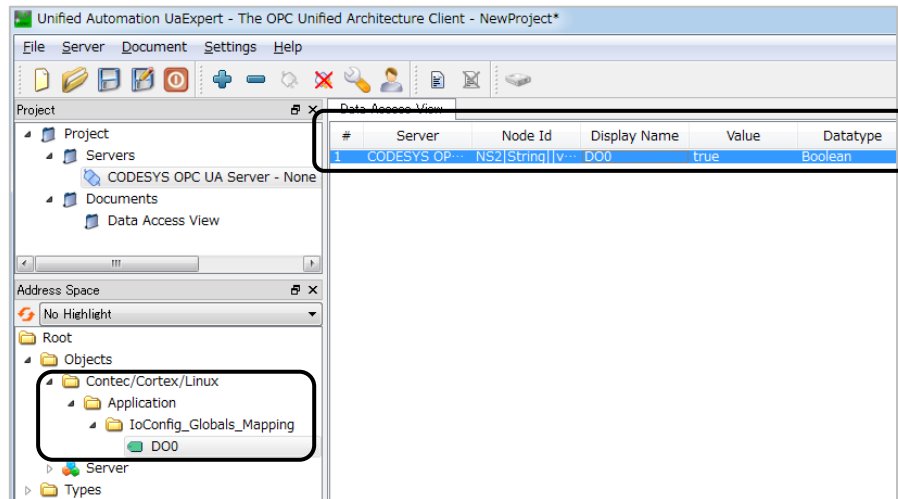
- 4 创建树时，选择[CODESYS OPC UA Server]并单击[OK]。



5 选择服务器，然后单击连接图标。



6 『Address Space』的『Root\Objects\Contec/Cortex/Linux\Application\PLC_PRG\』的树阶层内，显示了由CODESYS OPC Configurator注册的变量，将想监控的变量（这里为D00）拖动到『Data Access View』窗口。可以确认变量的状态。

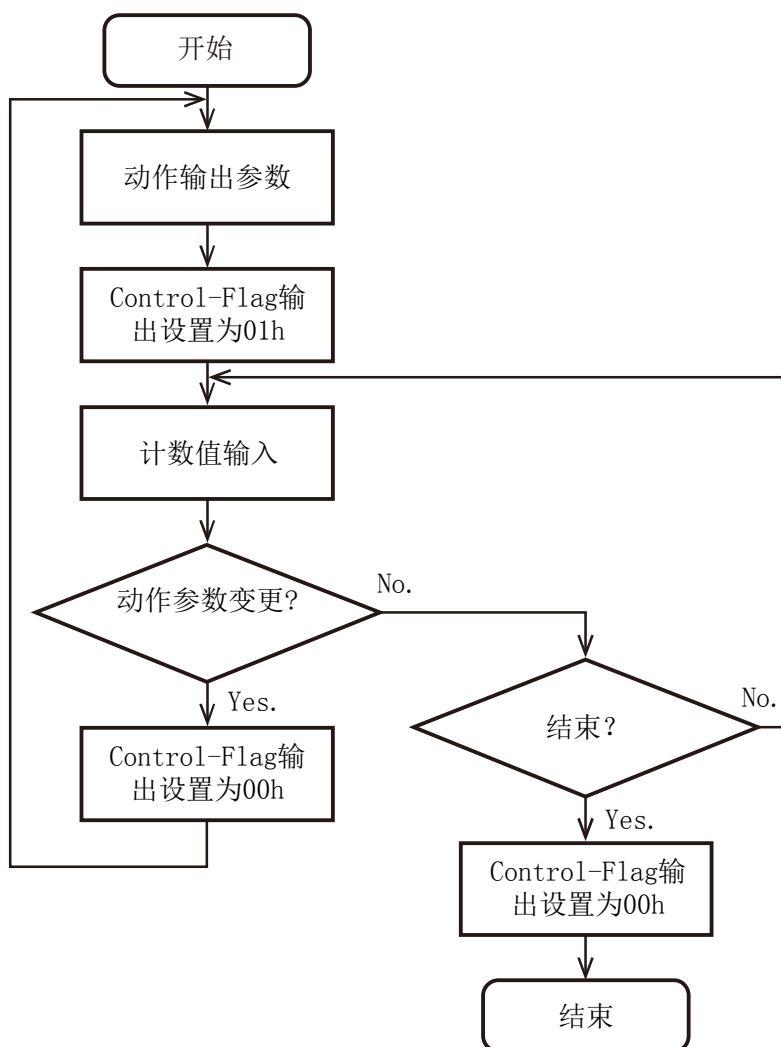


6. 计数器输入

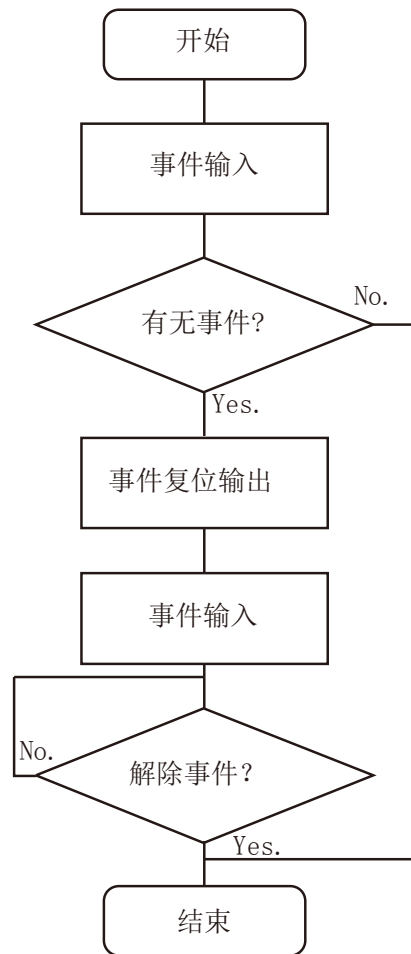
当I / O模块CPS-CNT-3202I连接到堆栈型CPU控制器时，可以使用计数器输入。 要使用计数器输入功能，需要执行以下步骤。

如果根据《基本编程顺序（P56）》来准备编程，请参考以下的动作程序进行编程。

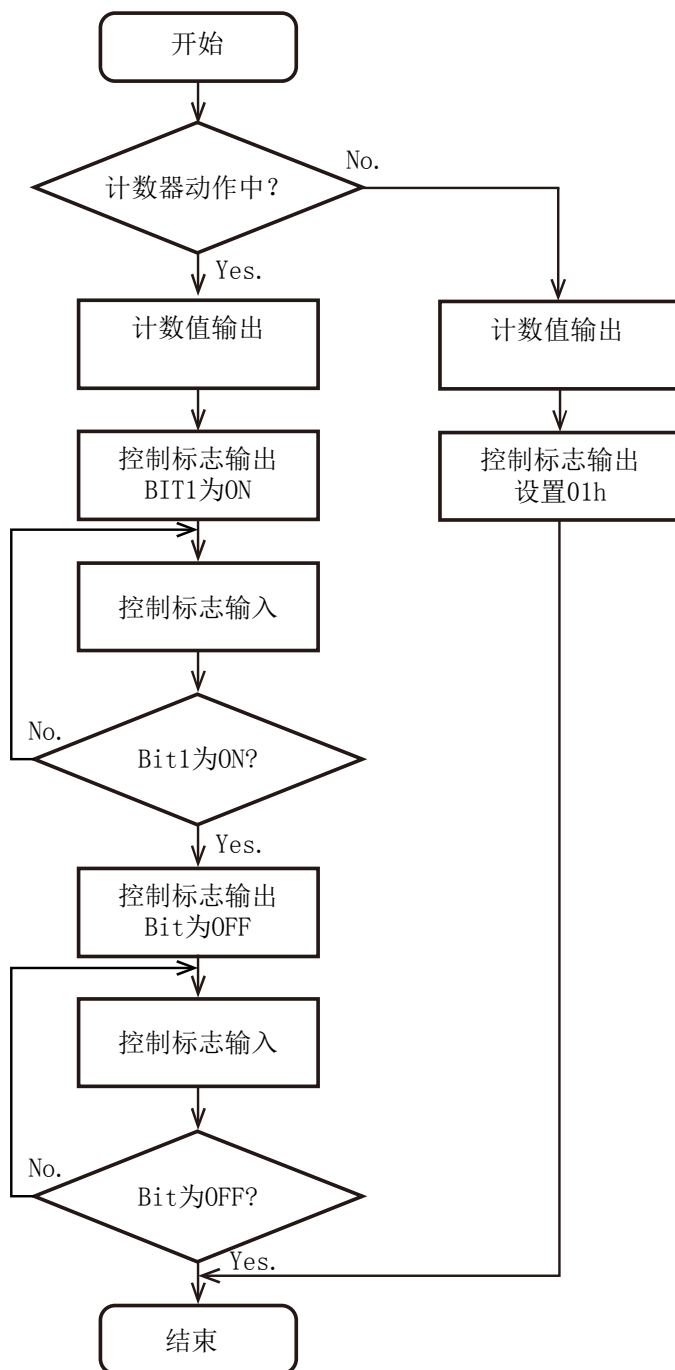
1. 基本操作顺序



2. 事件操作顺序



3. 计数值设定操作顺序



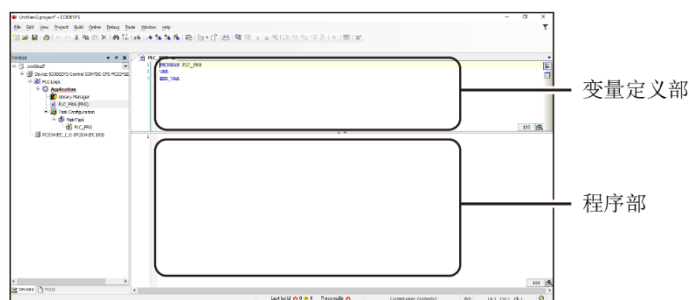
4. 示例程序

提供关于前面所示操作的示例程序。

程序标题	功能
示例1	设定CPS-CNT-3202I并进行计数器输入时
示例2	发生比较值一致的事件时
示例3	在操作中改变计数值时

以下显示示例程序的执行程序。

- 1** 双击设备窗口中的[PLC_PRG (PRG)]图标。
将显示“ST编辑器窗口”。
- 2** 在“ST编辑器窗口”的“变量定义部”的VAR和END_VAR之间写入变量定义部分源代码。
- 3** 在“ST Editor窗口”的“程序部”中写入程序部分的源代码。



◆ 示例1

设定CPS-CNT-3202I并进行计数器输入时

变量定义部

```
uiState: UINT := 0;  
dwCounter: DWORD;
```

程序部

```
CASE uiState OF  
  0:  
    CNT0_OutCtrl := 0;  
    CNT0_EventReset := 0;  
    uiState := 1;  
  1:  
    CNT0_ZMode := 0;  
    CNT0_ZLogic := 0;  
    CNT0_Dir := 0;  
    CNT0_Phase := 1;  
    CNT0_Multi := 0;  
    CNT0_Clear := 1;  
  
    CNT0_DiFilter := 0;  
    CNT0_Pulse := 0;  
    CNT0_CntSet := 0;  
    CNT0_CntMatch := 0;  
    uiState := 2;  
  2:  
    CNT0_OutCtrl := 16#01;  
    uiState := 3;  
  3:  
    dwCounter := CNT0_CntVal;  
    IF DWORD_TO_DINT(dwCounter) >= 100 THEN  
      uiState := 4;  
    END_IF  
  4:  
    CNT0_OutCtrl := 0;  
    uiState := 5;  
  5:  
END_CASE
```

◆ 示例2

发生比较值一致的事件时

变量定义部

```
uiState: UINT := 0;  
dwCounter: DWORD;  
byEvent: BYTE;
```

程序部

```
CASE uiState OF  
  0:  
    CNT0_OutCtrl := 0;  
    CNT0_EventReset := 0;  
    uiState := 1;  
  1:  
    CNT0_ZMode := 0;  
    CNT0_ZLogic := 0;  
    CNT0_Dir := 0;  
    CNT0_Phase := 1;  
    CNT0_Multi := 0;  
    CNT0_Clear := 1;  
    CNT0_DiFilter := 0;  
    CNT0_Pulse := 0;  
    CNT0_CntSet := 0;  
    CNT0_CntMatch := 100;  
    uiState := 2;  
  2:  
    CNT0_OutCtrl := 16#01;  
    uiState := 3;  
  3:  
    dwCounter := CNT0_CntVal;  
    byEvent := CNT0_Event;  
  
    IF (byEvent AND 16#01) = 16#01 THEN  
      uiState := 4;  
    END_IF  
  4:  
    CNT0_EventReset := 16#01;  
    uiState := 5;  
  5:  
    byEvent := CNT0_Event;  
    IF (byEvent AND 16#01) = 16#00 THEN  
      uiState := 6;  
    END_IF  
  6:  
    CNT0_OutCtrl := 0;  
    uiState := 7;  
  7:  
END_CASE
```

◆ 示例3

在操作中改变计数值时

变量定义部

```
uiState: UINT := 0;  
dwCounter: DWORD;  
byInCtrl: BYTE;
```

程序部

```
CASE uiState OF  
  0:  
    CNT0_OutCtrl := 0;  
    CNT0_EventReset := 0;  
    uiState := 1;  
  1:  
    CNT0_ZMode := 0;  
    CNT0_ZLogic := 0;  
    CNT0_Dir := 0;  
    CNT0_Phase := 1;  
    CNT0_Multi := 0;  
    CNT0_Clear := 1;  
    CNT0_DiFilter := 0;  
    CNT0_Pulse := 0;  
    CNT0_CntSet := 100;  
    CNT0_CntMatch := 0;  
    uiState := 2;  
  2:  
    CNT0_OutCtrl := 16#01;  
    uiState := 3;  
  3:  
    dwCounter := CNT0_CntVal;  
  
    IF DWORD_TO_DINT(dwCounter) >= 200 THEN  
      uiState := 4;  
    END_IF  
  4:  
    CNT0_CntSet := 100;  
    CNT0_OutCtrl := CNT0_OutCtrl OR 16#02;  
    uiState := 5;  
  5:  
    byInCtrl := CNT0_InCtrl;  
    dwCounter := CNT0_CntVal;  
  
    IF (byInCtrl AND 16#02) = 16#02 THEN  
      uiState := 6;  
    END_IF  
  6:  
    CNT0_OutCtrl := CNT0_OutCtrl AND 16#FD;  
    uiState := 7;
```

```
7:
  byInCtrl := CNT0_InCtrl;
  dwCounter := CNT0_CntVal;

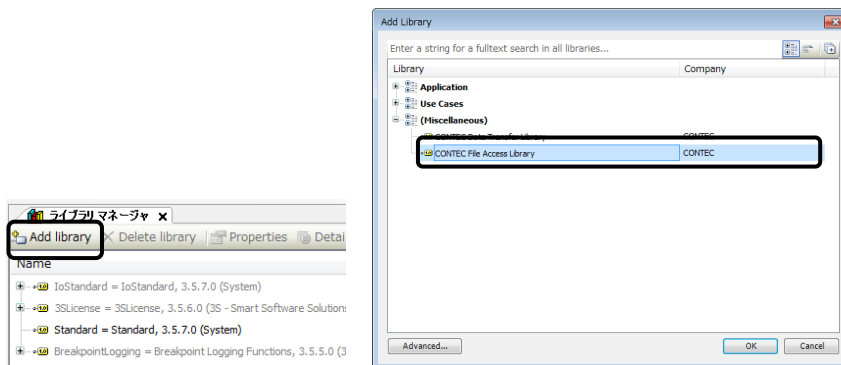
  IF (byInCtrl AND 16#02) = 16#00 THEN
    uiState := 8;
  END_IF
8:
  CNT0_OutCtrl := 0;
  uiState := 9;
9:
END_CASE
```

7. 访问文件

为与本产品连接的存储设备的文件分配访问权限。

1. 设置文件访问库

- 1 创建一个新的CODESYS项目。
IEC程序语言选择ST。
※有关如何创建项目的详细信息，请参阅“**新建项目的生成（P56）**”。
- 2 在设备窗口中双击[Library Manager]。
- 3 单击[Add Library]，在（Miscellaneous）树的层次结构中选择[CONTEC File Access Library]，然后单击[OK]。



2. 功能一览

功能名	功能
CFA_FileAccess	指定区域的访问控制
CFA_FileOpen	文件打开
CFA_FileClose	文件关闭
CFA_FileCloseAll	文件全部关闭
CFA_FileRead	读取文件
CFA_FileWrite	文件写入
CFA_FileSeek	文件指向移动
CFA_FileGetLine	从文件中读取一行的字符串
CFA_FilePutLine	对文件的字符串写入
CFA_FileDelete	删除文件
CFA_StringSeparate	字符串分解

3. 数据类型

◆ ERROR

名称	值	含义
NO_ERROR	0	正常结束
INVALID_PARAM	1	参数不正确。
INTERNAL_ERROR	2	资源不足。
INVALID_HANDLE	3	处理编号不正确。
NOT_EXIST	4	文件不存在。
EXIST	5	文件已存在。
READ_ONLY_FS	6	在装入的专用设备上执行文件写入。
NOT_MOUNT	7	未挂载。
GENERAL_ERROR	8	一般错误。

◆ AREATYPE

名称	值	含义
ROM	0	指定ROM区域。
RAM	1	指定RAM区域。
SD	2	指定SD卡。
USB	3	指定USB设备。

◆ SEEKWHENCE

名称	值	含义
CUR	0	文件指向的当前位置。
END	1	文件结束。
SET	2	文件的开头。

4. 功能说明

◆ CFA_FileAccess

功能

控制指定区域的访问状态。

输入值

名称	类型	含义
eArea	AREATYPE	指定要访问的区域类型。 由于RAM区域始终可读写，因此无需通过此功能进行控制。
bMount	BOOL	指定文件系统的安装状态。 访问的区域是USB设备的情况是必要的。 TRUE : 挂载在文件系统上。 FALSE: 卸载文件系统。
bWriteEnable	BOOL	指定写入启用状态。 TRUE : 允许写入。 FALSE: 禁止写入。

输出值

名称	类型	含义
eError	ERROR	错误代码。

◆ CFA_FileOpen

功能

在指定区域中打开文件。

输入值

名称	类型	含义
eArea	AREATYPE	指定要访问的区域类型。
sFileName	STRING (256)	指定文件名。
sFileMode	STRING (8)	指定文件的开放模式。 “r” : 以只读方式打开。 “w” : 开放写入。 “a” : 追加写入用。 “r+” : 读写用。 “w+” : 读写用。 “a+” : 读取和添加写入。

输入值

名称	类型	含义
eError	ERROR	错误代码。
hFile	DWORD	文件句柄号。

◆ CFA_FileClose

功能

关闭文件。

输入值

名称	类型	含义
hFile	DWORD	指定文件的句柄号。

输出值

名称	类型	含义
eError	ERROR	错误代码。

◆ CFA_FileCloseAll

功能

关闭所有打开的文件。

输入值

无

输出值

名称	类型	含义
eError	ERROR	错误代码。

◆ CFA_FileRead

功能

从文件中读取数据。

输入值

名称	类型	含义
hFile	DWORD	指定文件的句柄号。
pBuffer	POINTOR TO BYTE	指定所读取的数据的区域的地址。
szBuffer	WORD	指定存储数据的区域的大小。

出力值

名前	型	意味
eError	ERROR	错误代码。
szSize	WORD	读取的数据量。

◆ CFA_FileWrite

功能

将数据写入文件。

输入值

名前	型	意味
hFile	DWORD	指定文件的句柄号。
pBuffer	POINTOR TO BYTE	指定要写入的数据区的地址。
szSize	WORD	指定要写入的数据的数量。

输出值

名称	类型	含义
eError	ERROR	错误代码。

◆ CFA_FileSeek

功能

将文件指针移动到指定的位置。

输入值

名称	类型	含义
hFile	DWORD	指定文件的句柄号。
diOffset	DINT	指定从参考位置移动的字节数
eWhence	SEEKWHENCE	指定参考位置。

输出值

名称	类型	含义
eError	ERROR	错误代码。

◆ CFA_FileGetLine

功能

从文件中读取一行字符串。

读取的换行符将被删除。

输入值

名称	类型	含义
hFile	DWORD	指定文件的句柄号。
pszString	POINTOR TO STRING	指定存储读取字符串的区域的地址。
szSring	WORD	指定存储字符串的区域的大小。 szString - 1是最大字符数。 请指定足够的尺寸，包括换行符。

输出值

名称	类型	含义
eError	ERROR	错误代码。
szSize	WORD	导入的字符串的大小。

◆ CFA_FilePutLine

功能

将字符串写入文件。

换行符会自动添加到字符串的末尾。

输入值

名称	值	含义
hFile	DWORD	指定文件的句柄号。
pszString	POINTOR TO STRING	指定要写入的字符串的地址。

输出值

名称	值	含义
eError	ERROR	错误代码。

◆ CFA_FileDelete

功能

删除指定区域中的文件。

输入值

名称	值	含义
eArea	AREATYPE	指定要访问的区域类型。
sFileName	STRING (256)	指定文件名。

输出值

名称	值	含义
eError	ERROR	错误代码。

◆ CFA_StringSeparate

功能

将字符串分解为由分隔符分隔的字符串。

输入值

名称	值	含义
pszString	POINTER TO STRING	指定要分解的字符串的地址。
pszSeparator	POINTER TO STRING	指定分隔符组的地址。
pszLeft	POINTER TO STRING	指定存储左侧字符串的区域的地址，由分隔符分隔。
pszRight	POINTER TO STRING	指定存储分隔符右侧字符串的区域的地址。

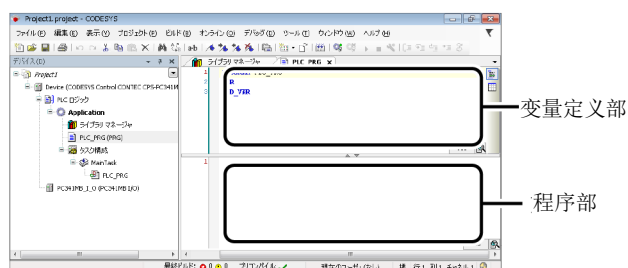
输出值

名称	值	含义
eError	ERROR	错误代码。

5. 示例程序

执行示例程序的顺序如下所示。

- 1 双击设备窗口中的[PLC_PRG (PRG)]图标。
将显示“ST编辑器窗口”。
- 2 在“ST编辑器窗口”的“变量定义部”的VAR和END_VAR之间写入变量定义部分源代码。
- 3 在“ST Editor窗口”的“程序部”中写入程序部分的源代码。



◆ 示例 1

在USB设备上写文件时

变量定义部

```
uiState: UINT := 0;
eAreaType: INT := CONTEC_File_Access_Library.AREATYPE.USB;
sFileName: CONTEC_File_Access_Library.FILENAME := 'sample1.dat';
hFile: CONTEC_File_Access_Library.HANDLE := 16#FFFFFFFF;
eError: CONTEC_File_Access_Library.ERROR;
byBuffer: ARRAY[0..255] OF BYTE;
iIndex: INT;
CFA_FileAccess: CONTEC_File_Access_Library.CFA_FileAccess;
CFA_FileOpen: CONTEC_File_Access_Library.CFA_FileOpen;
CFA_FileClose: CONTEC_File_Access_Library.CFA_FileClose;
CFA_FileWrite: CONTEC_File_Access_Library.CFA_FileWrite;
```

程序部

```
CASE uiState OF
  0:
    CFA_FileAccess.eArea := eAreaType;
    CFA_FileAccess.bMount := TRUE;
    CFA_FileAccess.bWriteEnable := TRUE;
    CFA_FileAccess(eError => eError);
    IF eError = 0 THEN
      uiState := 1;
    ELSE
      uiState := 5;
    END_IF
  1:
    CFA_FileOpen.eArea := eAreaType;
    CFA_FileOpen.sFileName := sFileName;
    CFA_FileOpen.sFileMode := 'w';
    CFA_FileOpen(hFile => hFile, eError => eError);
    IF eError = 0 THEN
      uiState := 2;
    ELSE
      uiState := 4;
    END_IF
  2:
    FOR iIndex := 0 TO 255 DO
      byBuffer[iIndex] := INT_TO_BYTE(iIndex);
    END_FOR
    CFA_FileWrite.pBuffer := ADR(byBuffer);
    CFA_FileWrite.szSize := 256;
    CFA_FileWrite(hFile := hFile, eError => eError);
    uiState := 3;
  3:
    CFA_FileClose(hFile := hFile, eError => eError);
    uiState := 4;
  4:
    CFA_FileAccess.eArea := eAreaType;
    CFA_FileAccess.bMount := FALSE;

    CFA_FileAccess.bWriteEnable := FALSE;
    CFA_FileAccess(eError => eError);
    uiState := 5;
  5:
END_CASE
```

◆ 示例2

在USB设备上读取文件时

变量定义部

```
uiState: UINT := 0;
eAreaType: INT := CONTEC_File_Access_Library.AREATYPE.USB;
sFileName: CONTEC_File_Access_Library.FILENAME := 'sample1.dat';
hFile: CONTEC_File_Access_Library.HANDLE := 16#FFFFFFFF;
eError: CONTEC_File_Access_Library.ERROR;
byBuffer: ARRAY[0..127] OF BYTE;
iIndex: INT;
szSize: WORD;
CFA_FileAccess: CONTEC_File_Access_Library.CFA_FileAccess;
CFA_FileOpen: CONTEC_File_Access_Library.CFA_FileOpen;
CFA_FileClose: CONTEC_File_Access_Library.CFA_FileClose;
CFA_FileRead: CONTEC_File_Access_Library.CFA_FileRead;
```

程序部

```
CASE uiState OF
  0:
    CFA_FileAccess.eArea := eAreaType;
    CFA_FileAccess.bMount := TRUE;
    CFA_FileAccess.bWriteEnable := FALSE;
    CFA_FileAccess(eError => eError);
    IF eError = 0 THEN
      uiState := 1;
    ELSE
      uiState := 5;
    END_IF

  1:
    CFA_FileOpen.eArea := eAreaType;
    CFA_FileOpen.sFileName := sFileName;
    CFA_FileOpen.sFileMode := 'r';
    CFA_FileOpen(hFile => hFile, eError => eError);

    IF eError = 0 THEN
      uiState := 2;
    ELSE
      uiState := 4;
    END_IF

  2:
    FOR iIndex := 0 TO 127 DO
      byBuffer[iIndex] := 0;
    END_FOR
    CFA_FileRead.pBuffer := ADR(byBuffer);
    CFA_FileRead.szBuffer := 128;
    CFA_FileRead(hFile := hFile, szSize => szSize, eError => eError);
```

```
IF eError = 0 THEN
  IF szSize = 0 THEN
    uiState := 3;
  END_IF
ELSE
  uiState := 3;
END_IF

3:
CFA_FileClose(hFile := hFile, eError => eError);
uiState := 4;

4:
CFA_FileAccess.eArea := eAreaType;
CFA_FileAccess.bMount := FALSE;
CFA_FileAccess.bWriteEnable := FALSE;
CFA_FileAccess(eError => eError);
uiState := 5;

5:
END_CASE
```

◆ 示例 3

使用文件指针移动功能获取文件大小时

变量定义部

```
uiState: UINT := 0;
eAreaType: INT := CONTEC_File_Access_Library.AREATYPE.USB;
sFileName: CONTEC_File_Access_Library.FILENAME := 'sample1.dat';
hFile: CONTEC_File_Access_Library.HANDLE := 16#FFFFFFFF;
eError: CONTEC_File_Access_Library.ERROR;
diFileSize: DINT := 0;
CFA_FileAccess: CONTEC_File_Access_Library.CFA_FileAccess;
CFA_FileOpen: CONTEC_File_Access_Library.CFA_FileOpen;
CFA_FileClose: CONTEC_File_Access_Library.CFA_FileClose;
CFA_FileSeek: CONTEC_File_Access_Library.CFA_FileSeek;
```


程序部

```
CASE uiState OF
  0:
    CFA_FileAccess.eArea := eAreaType;
    CFA_FileAccess.bMount := TRUE;
    CFA_FileAccess.bWriteEnable := FALSE;
    CFA_FileAccess(eError => eError);
    IF eError = 0 THEN
      uiState := 1;
    ELSE
      uiState := 5;
    END_IF

  1:
    CFA_FileOpen.eArea := eAreaType;
    CFA_FileOpen.sFileName := sFileName;
    CFA_FileOpen.sFileMode := 'r';
    CFA_FileOpen(hFile => hFile, eError => eError);
    IF eError = 0 THEN
      uiState := 2;
    ELSE
      uiState := 4;
    END_IF

  2:
    CFA_FileSeek.diOffset := 0;
    CFA_FileSeek.eWhence := CONTEC_File_Access_Library.SEEKWHENCE.END;
    CFA_FileSeek(hFile := hFile, diPos => diFileSize, eError => eError);
    uiState := 3;

  3:
    CFA_FileClose(hFile := hFile, eError => eError);
    uiState := 4;

  4:
    CFA_FileAccess.eArea := eAreaType;
    CFA_FileAccess.bMount := FALSE;
    CFA_FileAccess.bWriteEnable := FALSE;
    CFA_FileAccess(eError => eError);
    uiState := 5;

  5:
END_CASE
```

◆ 示例 4

删除USB设备上的文件时

变量定义部

```
uiState: UINT := 0;  
eAreaType: INT := CONTEC_File_Access_Library.AREATYPE.USB;  
sFileName: CONTEC_File_Access_Library.FILENAME := 'sample1.dat';  
eError: CONTEC_File_Access_Library.ERROR;  
CFA_FileAccess: CONTEC_File_Access_Library.CFA_FileAccess;  
CFA_FileDelete: CONTEC_File_Access_Library.CFA_FileDelete;
```

程序部

```
CASE uiState OF  
  0:  
    CFA_FileAccess.eArea := eAreaType;  
    CFA_FileAccess.bMount := TRUE;  
    CFA_FileAccess.bWriteEnable := TRUE;  
    CFA_FileAccess(eError => eError);  
    IF eError = 0 THEN  
      uiState := 1;  
    ELSE  
      uiState := 3;  
    END_IF  
  
  1:  
    CFA_FileDelete.eArea := eAreaType;  
    CFA_FileDelete.sFileName := sFileName;  
    CFA_FileDelete(eError => eError);  
    uiState := 2;  
  
  2:  
    CFA_FileAccess.eArea := eAreaType;  
    CFA_FileAccess.bMount := FALSE;  
    CFA_FileAccess.bWriteEnable := FALSE;  
    CFA_FileAccess(eError => eError);  
    uiState := 3;  
  
  3:  
END_CASE
```

◆ 示例5

为RAM中的文件逐行写入或读取字符串时

变量定义部

```
uiState: UINT := 0;
eAreaType: INT := CONTEC_File_Access_Library.AREATYPE.RAM;
sFileName: CONTEC_File_Access_Library.FILENAME := 'sample2.txt';
hFile: CONTEC_File_Access_Library.HANDLE := 16#FFFFFFFF;
eError: CONTEC_File_Access_Library.ERROR;
sStringData1: STRING(256) := 'Entry1=1234567890';
sStringData2: STRING(256) := 'Entry2=abcdefghijklmnopqrstuvwxy';
sStringData3: STRING(256);
sSeparator: STRING(8) := '=';
sStringLeft: STRING(256);
sStringRight: STRING(256);
iIndex: INT;
szSize: WORD;
CFA_FileOpen: CONTEC_File_Access_Library.CFA_FileOpen;
CFA_FileClose: CONTEC_File_Access_Library.CFA_FileClose;
CFA_FilePutLine: CONTEC_File_Access_Library.CFA_FilePutLine;
CFA_FileGetLine: CONTEC_File_Access_Library.CFA_FileGetLine;
CFA_StringSeparate: CONTEC_File_Access_Library.CFA_StringSeparate
```

程序部

```
CASE uiState OF
  0:
    CFA_FileOpen.eArea := eAreaType;
    CFA_FileOpen.sFileName := sFileName;
    CFA_FileOpen.sFileMode := 'w';
    CFA_FileOpen(hFile => hFile, eError => eError);
    IF eError = 0 THEN
      uiState := 1;
    ELSE
      uiState := 7;
    END_IF

  1:
    CFA_FilePutLine.pszString := ADR(sStringData1);
    CFA_FilePutLine(hFile := hFile, eError => eError);
    IF eError = 0 THEN
      uiState := 2;
    ELSE
      uiState := 3;
    END_IF

  2:
    CFA_FilePutLine.pszString := ADR(sStringData2);
    CFA_FilePutLine(hFile := hFile, eError => eError);
    uiState := 3;
```

```
3:
  CFA_FileClose(hFile := hFile, eError => eError);
  uiState := 4;

4:
  CFA_FileOpen.eArea := eAreaType;
  CFA_FileOpen.sFileName := sFileName;
  CFA_FileOpen.sFileMode := 'r';
  CFA_FileOpen(hFile => hFile, eError => eError);
  IF eError = 0 THEN
    uiState := 5;
  ELSE
    uiState := 7;
  END_IF

5:
  FOR iIndex := 0 TO 256 DO
    sStringData3[iIndex] := 0;
    sStringLeft[iIndex] := 0;
    sStringRight[iIndex] := 0;
  END_FOR
  CFA_FileGetLine.pszString := ADR(sStringData3);
  CFA_FileGetLine.szString := 256;
  CFA_FileGetLine(hFile := hFile, szSize => szSize, eError => eError);

  IF eError = 0 THEN
    IF szSize <> 0 THEN
      CFA_StringSeparate.pszString := ADR(sStringData3);
      CFA_StringSeparate.pszSeparator := ADR(sSeparator);
      CFA_StringSeparate.pszLeft := ADR(sStringLeft);
      CFA_StringSeparate.pszRight := ADR(sStringRight);
      CFA_StringSeparate(eError => eError);
    ELSE
      uiState := 6;
    END_IF
  ELSE
    uiState := 6;
  END_IF

6:
  CFA_FileClose(hFile := hFile, eError => eError);
  uiState := 7;

7:
  END_CASE
```

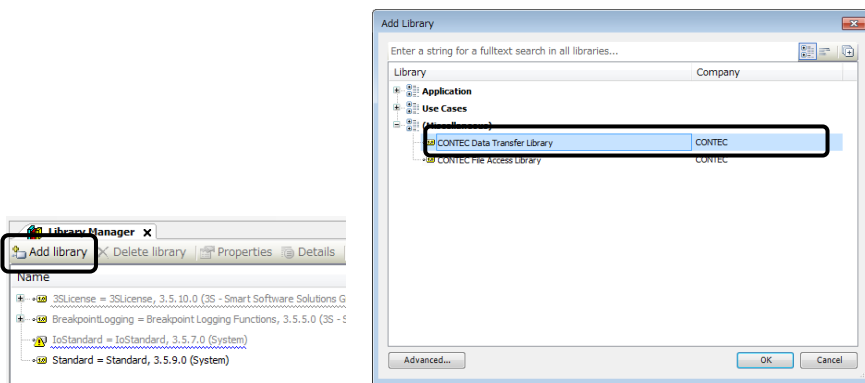
8. 将数据发送到云服务

将本产品收集的数据发送云服务的设定。

数据可以以CSV形式的方式保存。

1. 数据发送库的设置

- 1 创建一个新的CODESYS项目。
IEC程序语言选择ST。
※ 有关如何创建项目的详细信息，请参阅“新建项目的生成（P56）”。
- 2 在设备窗口中双击[Library Manager]。
- 3 单击[Add Library]，在（Miscellaneous）树的层次结构中选择[CONTEC Data Transfer Library]，然后单击[OK]。



2. 功能一览

功能名	功能
CDT_FileSetValue	数据设置为数据文件
CDT_FileDelete	删除数据文件
CDT_FileSetParameter	数据文件设置
CDT_CloudSend	云传输
CDT_CloudSetParameter	云传送用的参数组
CDT_CloudGetInformation	获取云传输信息
CDT_CloudControl	云传输的控制

3. 数据类型

◆ ERROR

名称	值	含义
NO_ERROR	0	正常结束
INVALID_PARAM	1	参数不正确。
INTERNAL_ERROR	2	资源不足。
TX_NODATA	3	数据文件不存在或数据文件中不存在数据。
TX_PENDING	4	由于在短时间内调用了云发送功能，因此未执行数据传输。
TX_ERROR	5	由于数据发送错误被返回，所以删除了数据文件。
TX_RESEND	6	因为数据发送失败，所以被设置为再发送的排队。

◆ DATATYPE

名称	值	含义
DATETIME	0	SYSTIME型的时间数据。
CRLF	1	表示一行最后的改行文字。
VAL_STRING	2	STRING型数据。
VAL_BOOL	3	BOOL型数据。存储区的大小是8位。
VAL_BYTE	4	无符号8位数据。
VAL_WORD	5	无符号16位数据。
VAL_DWORD	6	无符号32位数据。
VAL_LWORD	7	无符号64位数据。
VAL_SINT	8	带有符号的8位数据。
VAL_INT	9	带有符号的16位数据。
VAL_DINT	10	带有符号的32位数据。
VAL_LINT	11	带有符号的64位数据。
VAL_REAL	12	单精度浮点数数据。
VAL_LREAL	13	双精度浮点数数据。

◆ FILEPARAM

名称	值	含义
MAX_SIZE	0	以1K字节为单位的最大数据文件大小。
TIME_FORMAT	1	时间格式数据类型。 0 : YYYYMMDD, hhmmss, 毫秒 1 : YYYYMMDDhhmmss 2 : YYYYMMDDhhmm 3 : YYYYMMDD 4 : hhmmss 5 : 毫秒

◆ CLOUDPARAM

名称	值	含义
RETRY_TIME	0	再次发送等待之后，进行数据的再发送的时间（秒）。
RETRY_NUM	1	保持重新传输等待文件的文件数

◆ CLOUDINFO

名称	值	含义
RETRY_NUM	0	等待重传的文件数。

◆ CLOUDCTRL

名称	值	含义
RETRY_CLR	0	清除文件等待重传。

4. 功能说明

◆ CDT_FileSetValue

功能

创建要发送到服务器的数据文件。

数据以逗号分隔的CSV格式保存。

输入值

名称	类型	含义
iFileNo	INT	指定文件编号（0 - 9）。
eType	DATATYPE	指定要在数据文件中设置的数据类型。
pValue	POINTER TO BYTE	要设置的数据的地址。 时间数据设置SysTimeRtcHighResGet（）获得的值。 指定0时，在功能内获取时间并设置。 改行文字时，请指定0。

输出值

名称	类型	含义
eError	ERROR	错误代码。

◆ CDT_FileDelete

功能

删除创建中的数据文件。

输入值

名称	类型	含义
iFileNo	INT	指定文件编号（0-9）。

输出值

名称	类型	含义
eError	ERROR	错误代码。

◆ CDT_FileSetParameter

功能

设置有关数据文件的参数。

输入值

名称	类型	含义
iFileNo	INT	指定文件编号（0-9）
eParamNo	FILEPARAM	指定参数编号。 (MAX_SIZE) 以1 KB为单位指定数据文件的最大大小。 如果超出最大大小，将删除旧数据。 指定0时，没有大小限制。 数据 : WORD型 默认设置 : 0 (TIME_FORMAT) 设置时间数据的格式。 数据 : INT型 默认设置 : 1
pValue	POINTER TO BYTE	要设置的数据的地址。

输出值

名称	类型	含义
eError	ERROR	错误代码

◆ CDT_CloudSend

功能

将创建的数据文件发送到云服务。

传输的数据文件将被删除。

输入值

名称	类型	含义
iFileNo	INT	指定文件编号（0 - 9）。
xExecute	BOOL	指定执行状态。 当FALSE变为TRUE时，传输处理开始。

输出时

名称	类型	含义
xDone	BOOL	正常结束。
xBusy	BOOL	执行过程中。
xError	BOOL	错误终止。
eError	ERROR	错误代码。

◆ CDT_CloudSetParameter

功能

设置与云传输相关的参数。

输入值

名称	类型	含义
eParamNo	CLOUDPARAM	指定参数编号。 (RETRY_TIME) 重新发送等待后，设置重新发送数据的时间（秒）。 当指定0时，在执行云发送功能时发送等待重发的文件。 数据 : WORD型 默认设置 : 0 (RETRY_NUM) 设置保存重新传输等待文件的文件数。 指定0时，不保留等待重新传输的文件。 数据 : DWORD型 默认设置 : FFFFFFFFh
pValue	POINTER TO BYTE	要设置的值的地址。

输出值

名前	型	意味
eError	ERROR	错误代码。

◆ CDT_CloudGetInformation

功能

获取有关云传输的信息。

输入值

名称	类型	含义
eInfoNo	CLOUDINFO	指定信息编号。 (RETRY_NUM) 检索等待重新传输的文件数。 データ : DWORD型

输出值

名称	类型	含义
eError	ERROR	错误代码。

◆ CDT_CloudControl

功能

进行关于云传输的控制。

输入值

名称	类型	含义
eCtrlNo	CLOUDCTRL	指定控制编号。 (RETRY_CLR) 清除文件等待重传。 数据: 无
pValue	POINTER TO BYTE	存储控制信息的区域的地址。

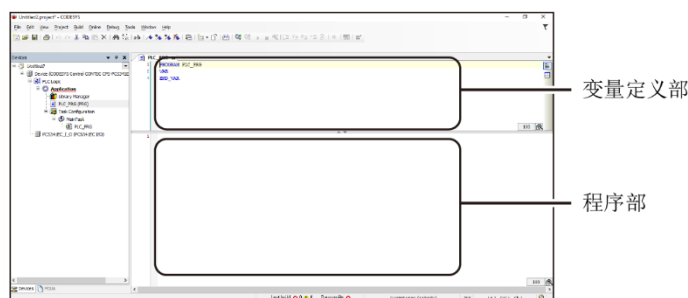
输出值

名称	类型	含义
eError	ERROR	错误代码

5. 示例程序

执行示例程序的过程如下所示。

- 1 双击设备窗口中的[PLC_PRG (PRG)]图标。
将显示“ST编辑器窗口”。
- 2 在“ST编辑器窗口”的“变量定义部”的VAR和END_VAR之间写入变量定义部分源代码。
- 3 在“ST Editor窗口”的“程序部”中写入程序部分的源代码。



◆ 示例 1

将CPS-DIO-0808RL的数字输入数据和CPS-SSI-4P的温度数据发送到云服务时

※ 此示例程序通过连接到堆栈型控制器的CPS-DIO-0808RL和CPS-SSI-4P执行。

变量定义部

```
DiData: ARRAY[0..7] OF BOOL;  
TempData: ARRAY[0..3] OF LREAL;  
dwIntervalTime: DWORD := 100;  
dwTimerCount1: DWORD;  
dwTimerCount2: DWORD;  
uiState: UINT := 0;  
uiCloudTxState: UINT;  
iFileNo: INT := 0;  
usFileSize: WORD := 8;  
iTimeFormat: INT := 1;  
dwCounter: DWORD;  
iIndex: INT;  
eError: CONTEC_Data_Transfer_Library.ERROR;  
eCloudTxError: CONTEC_Data_Transfer_Library.ERROR;  
CDT_FileSetValue: CONTEC_Data_Transfer_Library.CDT_FileSetValue;  
CDT_FileSetParameter: CONTEC_Data_Transfer_Library.CDT_FileSetParameter;  
CDT_CloudSend: CONTEC_Data_Transfer_Library.CDT_CloudSend;  
CDT_CloudSetParameter: CONTEC_Data_Transfer_Library.CDT_CloudSetParameter;  
CDT_CloudGetInformation: CONTEC_Data_Transfer_Library.CDT_CloudGetInformation;
```

程序部

```
CASE uiState OF
  0:
    CDT_FileSetParameter.eParamNo := CONTEC_Data_Transfer_Library.FILEPARAM.MAX_SIZE;
    CDT_FileSetParameter.pValue := ADR(usFileSize);
    CDT_FileSetParameter(iFileNo := iFileNo, eError => eError);

    CDT_FileSetParameter.eParamNo :=
      CONTEC_Data_Transfer_Library.FILEPARAM.TIME_FORMAT;
    CDT_FileSetParameter.pValue := ADR(iTimeFormat);
    CDT_FileSetParameter(iFileNo := iFileNo, eError => eError);
    dwTimerCount1 := 0;
    dwTimerCount2 := 0;
    wCounter := 0;
    uiCloudTxState := 0;
    uiState := 1;

  1:
    DO_BYTE := DO_BYTE + 1;
    DiData[0] := DI_BIT0;
    DiData[1] := DI_BIT1;
    DiData[2] := DI_BIT2;
    DiData[3] := DI_BIT3;
    DiData[4] := DI_BIT4;
    DiData[5] := DI_BIT5;
    DiData[6] := DI_BIT6;
    DiData[7] := DI_BIT7;
    TempData[0] := TEMP0;
    TempData[1] := TEMP1;
    TempData[2] := TEMP2;
    TempData[3] := TEMP3;

    dwTimerCount1 := dwTimerCount1 + dwIntervalTime;
    IF dwTimerCount1 >= 60000 THEN
      dwTimerCount1 := 0;

      CDT_FileSetValue.iFileNo := iFileNo;
      CDT_FileSetValue.eType := CONTEC_Data_Transfer_Library.DATATYPE.DATETIME;
      CDT_FileSetValue.pValue := 0;
      CDT_FileSetValue(eError => eError);

      dwCounter := dwCounter + 1;
      CDT_FileSetValue.eType := CONTEC_Data_Transfer_Library.DATATYPE.VAL_DWORD;
      CDT_FileSetValue.pValue := ADR(dwCounter);
      CDT_FileSetValue(eError => eError);

      CDT_FileSetValue.eType := CONTEC_Data_Transfer_Library.DATATYPE.VAL_BOOL;
      FOR iIndex := 0 TO 7 DO
        CDT_FileSetValue.pValue := ADR(DiData[iIndex]);
        CDT_FileSetValue(eError => eError);
      END_FOR

      CDT_FileSetValue.eType := CONTEC_Data_Transfer_Library.DATATYPE.VAL_LREAL;

      FOR iIndex := 0 TO 3 DO
        CDT_FileSetValue.pValue := ADR(TempData[iIndex]);
        CDT_FileSetValue(eError => eError);
```

```
END_FOR

CDT_FileSetValue.eType := CONTEC_Data_Transfer_Library.DATATYPE.CRLF;
CDT_FileSetValue.pValue := 0;
CDT_FileSetValue(eError => eError);
END_IF

dwTimerCount2 := dwTimerCount2 + dwIntervalTime;

CASE uiCloudTxState OF
0:
    IF dwTimerCount2 >= 600000 THEN
        dwTimerCount2 := 0;
        CDT_CloudSend.iFileNo := iFileNo;
        CDT_CloudSend(xExecute := FALSE);
        CDT_CloudSend(xExecute := TRUE);
        uiCloudTxState := 1;
    END_IF
1:
    CDT_CloudSend();
    IF CDT_CloudSend.xBusy = FALSE THEN
        IF CDT_CloudSend.xDone OR CDT_CloudSend.xError THEN
            eCloudTxError := CDT_CloudSend.eError;
            uiCloudTxState := 0;
        END_IF
    END_IF
END_CASE
END_CASE
```

◆ 示例 2

设置应用程序获取的时间数据时

SysTimeRtcHighResGet () 需要在[Library Manager]的[System] - [SysLibs]中添加[SysTime]。

变量定义部

```
uiState: UINT := 0;  
iFileNo: INT := 0;  
TimeStamp: SYSTIME;  
eError: CONTEC_Data_Transfer_Library.ERROR;  
CDT_FileSetValue: CONTEC_Data_Transfer_Library.CDT_FileSetValue;
```

程序部

```
CASE uiState OF  
  0:  
    SysTimeRtcHighResGet(TimeStamp);  
  
    CDT_FileSetValue.iFileNo := iFileNo;  
    CDT_FileSetValue.eType := CONTEC_Data_Transfer_Library.DATATYPE.DATETIME;  
    CDT_FileSetValue.pValue := ADR(TimeStamp);  
    CDT_FileSetValue(eError => eError);  
  
    CDT_FileSetValue.eType := CONTEC_Data_Transfer_Library.DATATYPE.CRLF;  
    CDT_FileSetValue.pValue := 0;  
    CDT_FileSetValue(eError => eError);  
    uiState := 1;  
  
  1:  
END_CASE
```

监控编辑

解释了CONPROSYS HMI (Human Machine Interface)，可以很容易地在Web浏览器上创建监控画面，并可以监控运行状态，故障监控，操作，停止等。

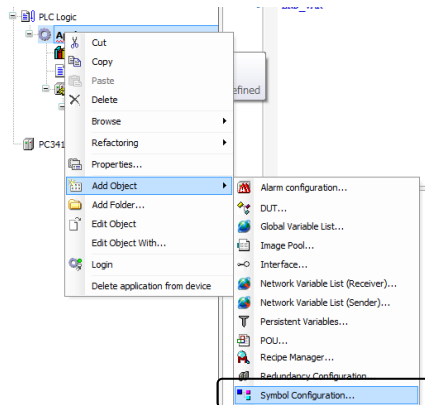
1. 使用CONPROSYS HMI

在本产品进行监测的情况下，使用附属的《CONPROSYS HMIH》，可以制作监控画面。

1. CODESYS的设置

1 利用『基本编程顺序 (P56)』中的『新建项目的生成 (P56)』和『从CODESYS连接到控制器 (P57)』程序创建CODESYS项目。

2 右键单击设备窗口中显示的[Application]，然后选择[添加对象] - [Symbol configuration]。



3 在“添加Symbol configuration”对话框中，单击“添加”按钮。

2. 创建和构建程序

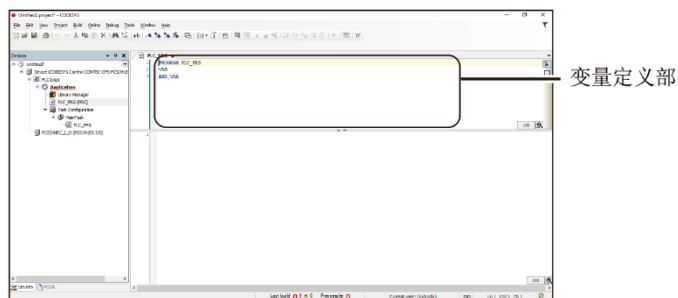
在此，作为例子，不使用硬件的I / O功能，创建只有软件变量的CODESYS程序。

Incrementer是每周期时间进行计数的变量。

用Switch变量控制动作的开始/停止，Status变量保持当前状态。

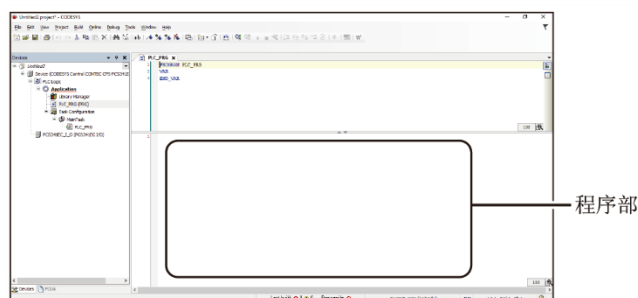
1 双击设备窗口中的[PLC_PRG (PRG)]图标。
将显示“ST编辑器窗口”。

- 2 在“ST编辑器窗口”的“变量定义部”的VAR和END_VAR之间编写以下源代码。



```
Switch: BOOL;  
Incrementer: INT;  
Status: STRING[10];
```

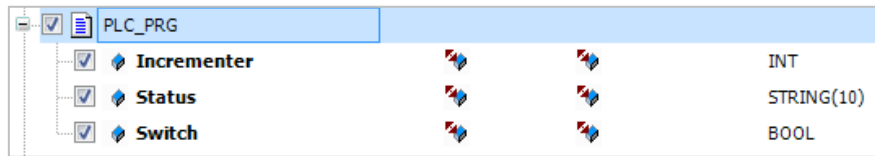
- 3 在“ST Editor窗口”的“程序部”中编写以下源代码。



```
IF Switch = TRUE THEN  
    Incrementer := Incrementer + 1;  
    IF Incrementer = 100 THEN  
        Incrementer := 0;  
    END_IF  
    Status := 'BUSY';  
ELSE  
    Status := 'STOP';  
END_IF
```

- 4 从CODESYS的[Build]菜单执行[Rebuild]，检查构建过程是否正常完成。

- 5 在设备窗口中双击[Symbol configuration]，然后检查PLC_PRG树层次结构中的“Incrementer”，“Status”，“Switch”变量。



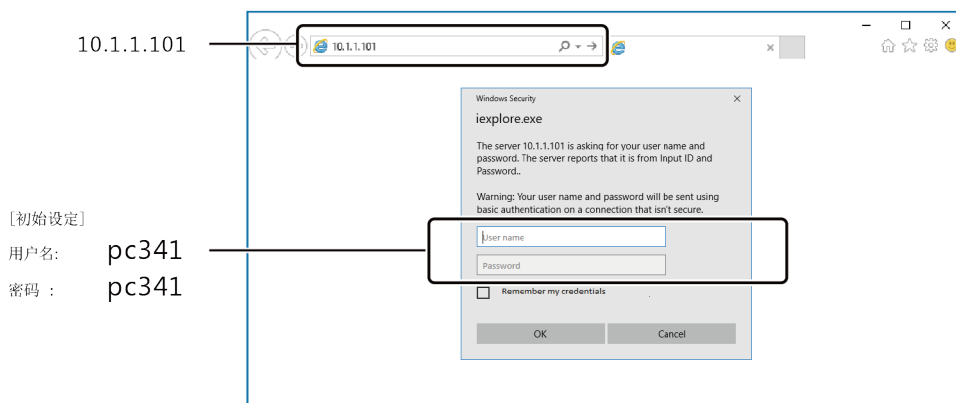
※*只有在Symbol configuration中“IoConfig_Globals_Mapping”或“PLC_PRG”的树层次结构中注册的变量才能与CONPROSYS HMI一起使用。
在其他树上注册的变量，在CONPROSYS HMI上被忽略。

- 6 从CODESYS的[在线]菜单执行[登录]。
※ 将显示下载确认对话框，单击[是]。

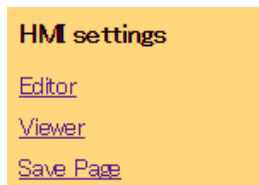
3. 使用HMI Editor创建画面

- 1 在控制器所连接的PC上启动Web浏览器，并在地址输入部分输入“http://10.1.1.101/”（PLC控制器的IP地址）。
※ 有关控制器连接的详细信息，请参阅“**电脑网络设定（P18）**”。
※ 也可以访问“https://10.1.1.101/”。如果显示“证书错误”屏幕，请选择“继续浏览此站点”。

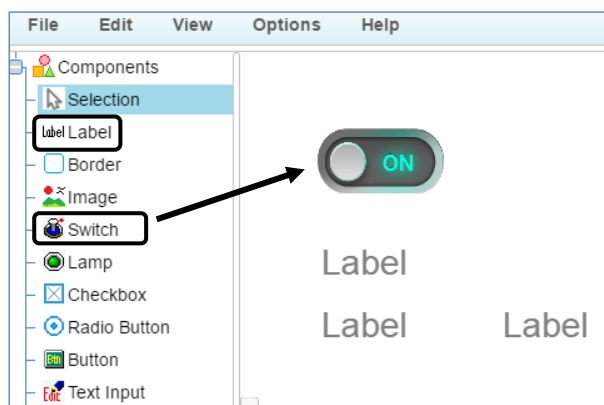
- 2 用户名为“pc341”，密码输入“pc341”并登录。



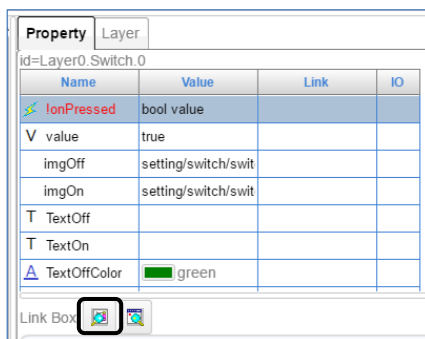
3 选择“HMI设置”的“Editor”以启动CONPROSYS HMI。



4 将“显示控制选择区域”中的1个“开关”和3个“标签”拖放到图层区域。



5 单击开关将其选中。
接下来，单击编辑器画面右侧属性中的链接设置图标。



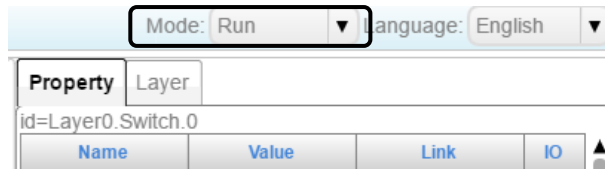
- 6** 单击设备树中的[更新]按钮，将更新CODESYS中使用的变量列表。 选择[Switch]，然后单击[OK]。
变量Switch被分配给开关部件。



- 7** 与步骤5和6一样，将Incrementer和Status变量分配给2个标签。
要在另1个标签上显示“Status”，请在属性的文本部分中输入“Status”。

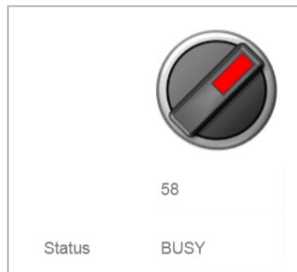
4. HMI Viewer的执行

- 1** 将画面右上角的模式设置为[执行]。



- 2** 从CODESYS [Debug]菜单中选择[Drive]。

- 3** 当放置在画布上的开关ON时，Incrementer编号将在0到100的范围内更新。 如果OFF开关，Incrementer更新将停止。
此外，Status变量中显示1（PLC正在进行中的操作），当在CODESYS侧停止操作时，该值将更改为0。



- 4** 不是在Editor中操作，而是在Web浏览器执行Viewer画面。
开始保存在[文件] - [保存页面]中制作的HMIX画面。
存储位置位于user文件夹下，存储名称为Page 1。
接下来，选择[文件] - [Viewer运行]，将打开一个新选项，并执行Viewer画面。

※ 保存的画面重新启动会丢失。请用以下两种方法来保存页面的ROM区域。

- 单击[文件] - [将设置保存到ROM]。
- 单击CONPROSYS WEB Setting的HMI设置中的[保存页面]。

5. 可与IEC数据类型使用的部件

CONPROSYS HMIA可以使用以下IE数据类型。

- BOOL
- DWORD
- INT
- STRING
- BYTE
- UDINT
- DINT
- USINT
- LWORD
- LINT
- WORD
- ULINT
- REAL
- UINT
- SINT
- LREAL

可用于各个部件的IEC数据类型

部件名称	BOOL型	STRING型	除此之外
标签	○	○	○
框架	×	×	×
图像	×	×	×
开关	○	×	×
灯	○	×	×
复选框	○	×	×
单选按钮	×	×	×
按钮	×	×	×
输入文字	○	○	○
滑块	○	×	○
视频	×	×	×
趋势	○	×	○
圆形仪表	○	×	○
液位计	○	×	○
圆形图表	○	×	○
趋势栏	○	×	○
延迟开关	○	×	×
多状态灯	○	×	○

2. CONPROSYS HMI 的概要

1. 可以使用CONPROSYS HMI做什么

排列准备的显示零件，可以监控输入的信号状态。

只需在Web浏览器上创建画面，不需要程序语言的知识 and 特别的开发环境。

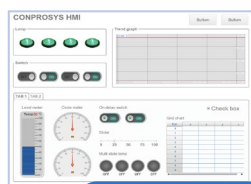
可以拖放许多准备好的显示部件。

另外，显示部件的设定和传感器等的数据的链接，可以在属性画面上设置。

多种多样的显示部件
丰富多彩的画面

编辑时也可以同时进行
数据关联和确认

只用conprosys就可以
实现可视化



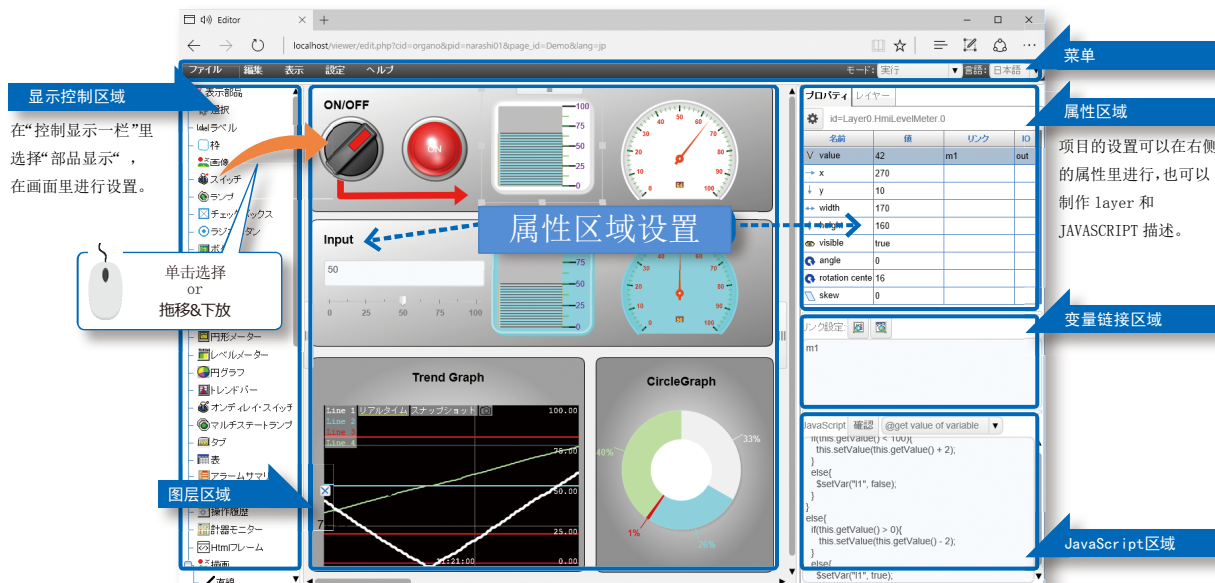
收集监控信息

信息的确认

设备的变更指示

设置、保存、执行在web上完成

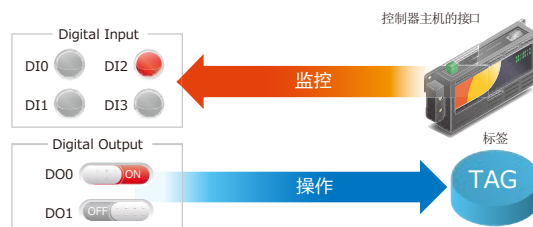
可以多平台使用



上述画面是显示器画面的印象图，实际出货时的设定为白色背景。

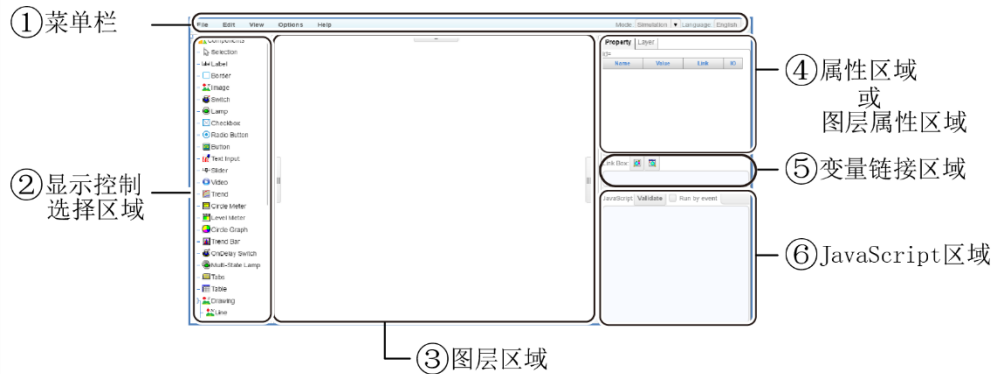
CONPROSYS VTC的数据关联

可以和使用内变量(TAG)的 conprosysVTC 进行关联，可以显示用 VTC 处理的结果，也可以把 HMI 的操作反映在 VTC 的 flow control。



2. 操作区域

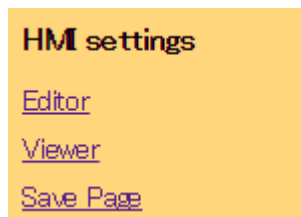
CONPROSYS HMI具有以下操作区域。



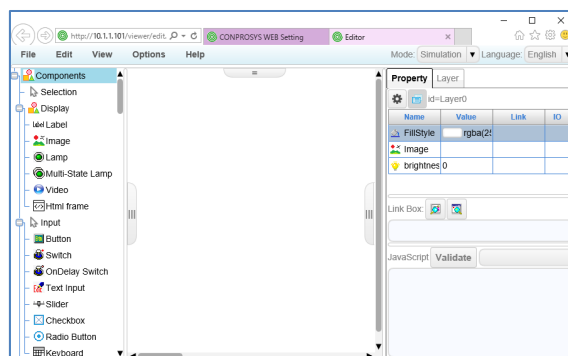
No.	名称	功能
①	菜单栏	以菜单按钮格式执行命令。
②	显示控制选择区域	可以选择要放置的显示控件。
③	图层区域	画面设计屏幕的区域。
④	属性区域	可以更改显示控件的属性变量。
	图层属性区域	可以添加，删除和更改图层属性。
⑤	变量链接区域	执行显示控制属性和变量链接。
⑥	JavaScript区域	通过JavaScript设置实现显示控制。

3. 创建监控画面

单击CONPROSYS WEB设置的“HMI设置”中的“Editor”。



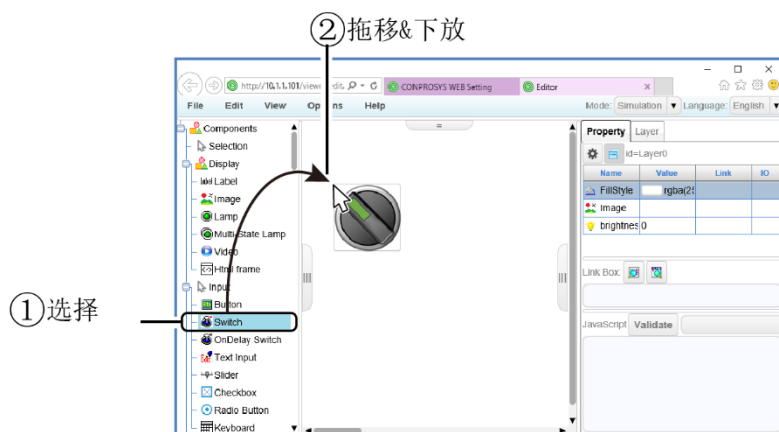
CONPROSYS HMI将启动，可以创建监控画面。



4. 创建监控画面的基本操作

◆ 放置显示部件

①从显示控制选择区域的树中选择显示部件，②然后将其拖放到图层区域。



◆ 设置属性

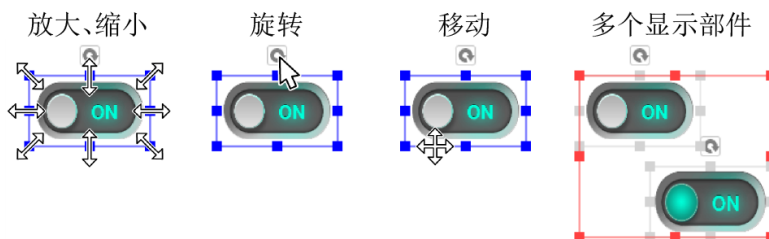
①选择配置的显示部件时，②会在属性区域显示该显示部件的属性。

在属性区域中，可以直接更改值或设置数据以与输入/输出设备和其他部件链接。



◆ 调整显示部件的位置和大小

可以通过拖动显示的框架来调整位置，大小和角度，同时左键单击所选的显示部件并选择它。
也可以同时选择多个零件并同时调整位置和大小。



◆ 复制或删除显示的部件

右键单击选中放置的显示部件，然后显示编辑菜单以复制和删除显示部件。

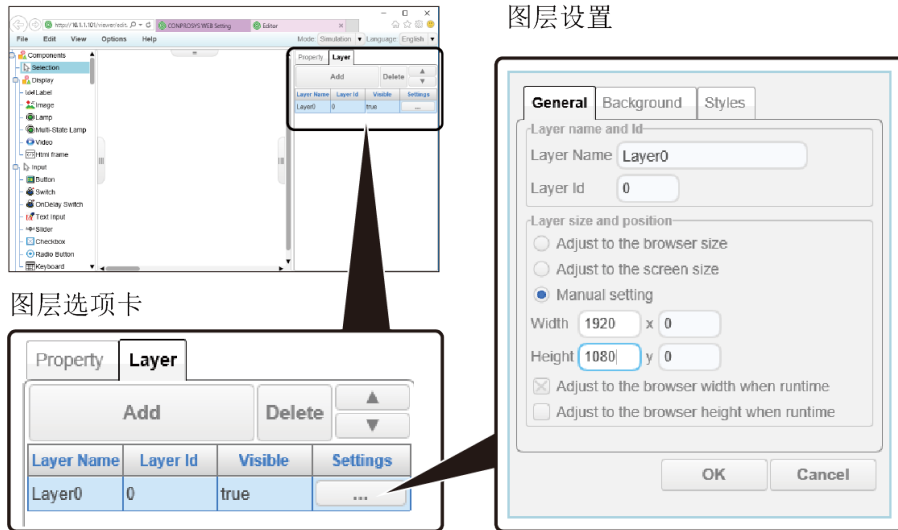


◆ 设置图层

从画面右上角选择[图层]选项时，将显示图层属性区域。

图层属性区域单击[设置]列中的[...]按钮以显示设置对话框。

可以设置监视画面的大小和背景。

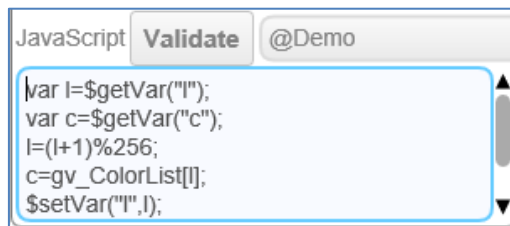


◆ 编写JavaScript

如有必要，请编写Javascript。

如果您需要自己的控制逻辑，还可以在JavaScript区域编写Javascript来进行操作。

有关各个显示部件可用的Javascript功能，请参阅“在线帮助”。

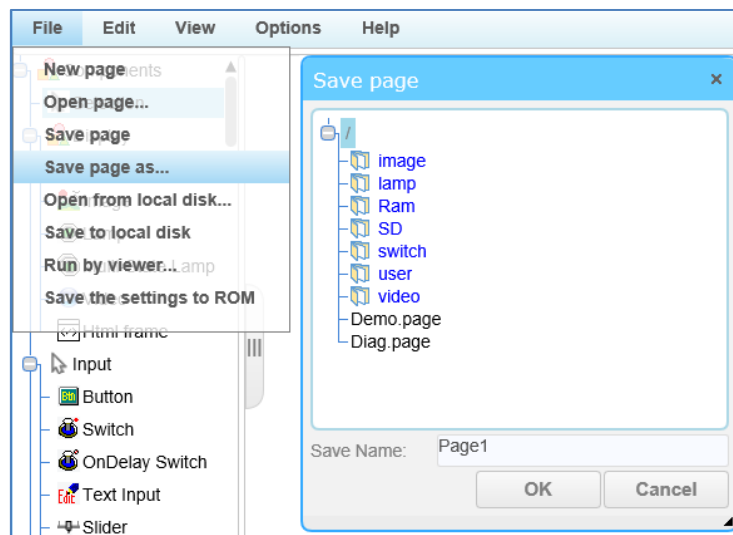


◆ 保存监控画面

创建监视画面后，可以使用文件菜单中的任意名称保存屏幕。

保存后，请在电源OFF前从菜单中执行[文件] -> [保存设置到ROM]菜单。

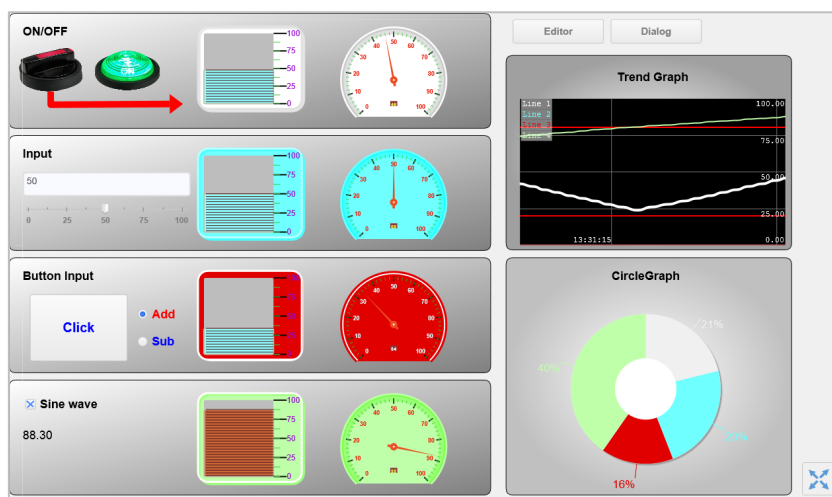
※ 如果不执行[将设置保存到ROM]，重新启动或电源OFF，创建的监视画面将返回其原始状态。



◆ 显示监控画面

可以通过单击CONPROSYS WEB Setting的[状态菜单] - [监视显示]来显示监视画面。

默认情况下，会显示保存在“user / Page1.page”中的画面。



※ 要显示具有不同名称的画面，请在Web浏览器中指定以下URL。

<http://<IP地址>:<端口号>/viewer/view.htm?pagepath=<页面文件路径>&lang=<显示语言>>

〈页面文件路径〉：指定页面的文件名。例：/user/Page1.page

〈显示语言〉：指定要显示的语言名称。例：日语是jp。语言规范可以省略。

3. 显示部件一览

可用于监视CONPROSYS HMI编辑的显示部件如下。

显示部件	名称	内容
	标签	可以显示字符串。
	框架	带标题显示的框架。
	图像	可以显示图像。
	开关	可以输出ON / OFF状态的开关。
	灯	可以显示ON / OFF状态的灯。
	复选框	可以输出ON / OFF状态并可以显示字符串的复选框。
	单选按钮	只可从多个条件下只选择一个的单选按钮。
	按钮	可以显示和单击字符串的按钮。
	输入文字	可以输入和显示字符串。
	滑块	滑块可以输出数据。
	视频	可以播放视频。
	趋势	可以显示时间序列的数据。
	圆形仪表	可以以圆形表形式显示数据。
	液位计	可以以水平表形式显示数据。
	圆形图表	可以用圆图形式显示数据。
	趋势栏	可以用趋势或条形显示数据。
	延时开关	是可以在被按下的一定时间后输出ON / OFF状态的开关。
	多状态灯	是可以显示多个不同状态值的灯。
	标签	可以创建多个标签，并切换显示。
	表格	可以用表格形式显示数据。
	Html框架	可以在框架中显示另一个Html文档。
	列表	以列表形式显示值。
	直线	可以在画面上绘制直线。
	折线	可以在画面上绘制折线（连续直线）。
	曲线	可以在画面上绘制曲线。
	四方形	可以在画面上绘制四方形。
	圆角矩形	画面上可以绘制圆角的四方形。
	多边形	在画面上可以绘制多边形。
	圆	在画面上可以绘制圆。
	管道	可以在画面上绘制管道的连续线。

有关CONPROSYS HMI操作和功能的详细信息，请参阅“在线帮助”。

在线帮助

<http://data.conprosys.com/help/hmi/V1/jp/>

问题

对故障和各问题的原因和应对方法，进行确认方法的说明。

1. 发生故障时

使用当中发生不良时，请按以下顺序进行确认。

1. 整体

◆ 确认前面的LED。

- 请确认PWR的LED是否会亮。
- 请确认ST1的LED是否闪烁。

◆ 确认网络端口的LED。

请确认前面的UTP连接器的LED。

如果网线与HUB正确连接的话，Link/Act LED会亮灯。

灯不亮的情况，请参照“硬件安装指南”。

通过网络端口进行通信的话Link/Act LED会闪烁。

◆ 使用PING命令确认主计算机是否有应答。

PING该机器的IP地址。

机器正在运行状态的话会进行应答。

例) 将机器IP设定为10.1.1.101时

```
ping 10.1.1.101<Enter>:  
Reply from 10.1.1.101: bytes=32 time<10ms TTL=255  
Reply from 10.1.1.101: bytes=32 time<10ms TTL=255  
Reply from 10.1.1.101: bytes=32 time<10ms TTL=255  
显示应答。
```

忘记已设定的IP地址时，可将SW1-2的开关设为ON（左）接通电源，恢复出厂设置（IP地址：10.1.1.101）（用户名、密码也会恢复出厂设置）。

⚠ 注意

开关恢复原状时，按ROM中保存的设置内容启动。

◆ 通过主计算机的浏览器连接的用户名、密码认证失败时

用户名、密码需要区分大小写。请确认是否大写锁定，再次输入。

忘记用户名或密码时，可将SW1-2的开关设为ON（左）接通电源，恢复出厂设定。

（IP地址也会恢复出厂设置）

⚠ 注意

设置的其他设定的其他项目也会被初始化。

◆ PING命令虽然有应答，但是与浏览器连接后出现「该页面无法显示」时

请根据以下内容对浏览器进行设置。

代理服务器的设置

请设置为「不使用代理服务器」。

拨号设置

请设定为「不使用拨号」。

◆ 无法正常启动时

请交给本公司进行维修。

规格

关于硬件和CONPROSYS HMI规格的说明。

1. 硬件规格

1. CPS-PC341EC-1-9201的规格

项目		CPS-PC341EC-1-9201
CODESYS对应功能	版本	V3.5 SP7 Patch2 以上
	语言	LD, SFC, FBD, ST, IL, CFC (IEC61131-3标准)
	现场总线	EtherCAT Master、Modbus TCP Slave
	通讯协议	OPC-UA Server
程序大小	ROM大小	1MB
	最大步数	250K 步
CPU基本性能	基本命令执行速度 (LD)	1.6ns
	应用命令执行速度 (ST)	5.8ns
	扫描时间	74 μs (20000 ステップ時)
EtherCAT性能	输入处理时间 (LD)	144ns
	输出处理时间 (ST)	138ns
	扫描时间	166 μs (64点输入, 64输出时)

2. CPS-PC341MB-ADSC1-9201的规格

项目		CPS-PC341MB-ADSC1-9201
CODESYS对应功能	版本	V3.5 SP7 Patch2 以上
	语言	LD, SFC, FBD, ST, IL, CFC (IEC61131-3标准)
	现场总线	Modbus TCP Master / Slave
	通讯协议	OPC-UA Server
程序大小	ROM大小	1MB
	最大步数	250K 步
CPU基本性能	基本命令执行速度 (LD)	1.6ns
	应用命令执行速度 (ST)	5.8ns
	扫描时间	74 μs (20000 ステップ時)

3. CPS-PCS341EC-DS1-1201的规格

项目		CPS-PCS341EC-DS1-1201
CODESYS对应功能	版本	V3.5 SP7 Patch2 以上
	语言	LD, SFC, FBD, ST, IL, CFC (IEC61131-3标准)
	现场总线	EtherCAT Master、Modbus TCP Slave
	通讯协议	OPC-UA Server
程序大小	ROM大小	1MB
	最大步数	250K 步
CPU基本性能	基本命令执行速度 (LD)	1.6ns
	应用命令执行速度 (ST)	5.8ns
	扫描时间	74 μs (20000 ステップ時)
EtherCAT性能	输入处理时间 (LD)	144ns
	输出处理时间 (ST)	138ns
	扫描时间	166 μs (64点输入, 64输出時)

4. CPS-PCS341MB-DS1-1201的规格

项目		CPS-PCS341MB-DS1-1201
CODESYS对应功能	版本	V3.5 SP7 Patch2 以上
	语言	LD, SFC, FBD, ST, IL, CFC (IEC61131-3标准)
	现场总线	Modbus TCP Master / Slave
	通讯协议	OPC-UA Server
程序大小	ROM大小	1MB
	最大步数	250K 步
CPU基本性能	基本命令执行速度 (LD)	1.6ns
	应用命令执行速度 (ST)	5.8ns
	扫描时间	74 μs (20000 ステップ時)

2. CONPROSYS HMI 规格

项目		规格
对应的IEC数据类型		BOOL BYTE USINT WORD UINT DWORD UDINT LWORD ULINT SINT INT DINT LINT REAL LREAL STRING
CONPROSYS HMI预留变量		PLC状态 项目名称: Status/PLCStatus 访问: R 数据类型: BYTE 数据范围: 0 (PLC停止中) or 1 (PLC运作中)
最大可用变量数		128
出厂设置	CPS-PC341EC-1-9201	无
	CPS-PC341MB-ADSC1-9201	数字输出位0~1 项目名称: IoVariables/D00~1 访问: R/W 数据类型: BIT 数据范围: 0 or 1 数字输入位0~3 项目名称: IoVariables/DI0~3 访问: R 数据类型: BIT 数据范围: 0 or 1 模拟输入通道0~1 项目名称: IoVariables/AI0~1 访问: R 数据类型: DWORD 数据范围: 0~4095 计数器输入通道0~1 项目名称: IoVariables/CNT0~1 访问: R 数据类型: DWORD 数据范围: 0~16777215 计数器输入通道0~1清除 项目名称: IoVariables/CNTCLR0~1 访问: R/W 数据类型: BIT 数据范围: 0 or 1 (1でクリア) PLC状态 项目名称: Status/PLCStatus 访问: R 数据类型: BYTE

项目		规格
		数据范围：0 (PLC停止中) or 1 (PLC运作中)
	CPS-PCS341EC-DS1-1201 CPS-PCS341MB-DS1-1201	数字输入位0~3 项目名称：IoVariables/DI0~3 访问：R 数据类型：BIT 数据范围：0 or 1 PLC状态 项目名称：Status/PLCStatus 访问：R 数据类型：BYTE 数据范围：0 (PLC停止中) or 1 (PLC运作中)

附录

本章针对出现的规格及产品进行补充说明。

1. 发送数据的格式

本产品与服务器通过使用http或https发送数据。

在设置的URL POST下列参数。

◆ 发送参数一览

发送内容	参数1	参数2
测量数据文件	file=data	filename=YYYYMMDDhhmmss_cdtxx.csv

◆ 从Web服务器响应

应答	内容	本产品的动作
代码: 200 X-AggregateInfo-Result: OK	正常	将发送的文件从本机中删除。
代码: 400	无效ID、认证code异常、Format异常等	删除发送失败的文件。
除此之外	其他异常	将发送失败的文件作为再次发送用的文件保存。

◆ 电文例

http request
<pre>POST /XXXXX HTTP/1.1<CR_LF> User-Agent: XXXXX<CR_LF> Host: 192.168.1.110<CR_LF> Accept: */*<CR_LF> Content-Length: 40602<CR_LF> Expect: 100-continue<CR_LF> Content-Type: multipart/form-data; boundary=-----43ac9283b67c39f1<CR_LF> Content-Disposition: form-data; name="data"; filename="201401011000.csv"<CR_LF> Content-Type: text/plain;charset=UTF-8<CR_LF> <CR_LF> [测试数据] -----43ac9283b67c39f1<CR_LF> Content-Disposition: form-data; name="err"; filename="20161219100000_cdr00.csv "<CR_LF> Content-Type: text/plain;charset=UTF-8<CR_LF> <CR_LF></pre>
http response (正常)
<pre>HTTP/1.1 200 OK<CR_LF> Server: Apache-Coyote/1.1<CR_LF> Content-Type: text/plain;charset=UTF-8<CR_LF> Content-Length: XXXX<CR_LF> Date: Wed, 01 Jan 2014 10:00:01 GMT<CR_LF> X-AggregateInfo-Result: OK<CR_LF></pre>

连接超时.....20秒

Web服务器应答超时.....60秒

◆ 发送端Web服务器

使用在CONPROSYS WEB Setting『数据发送设置』『数据发送先URL』中指定的URL。

◆ 数据文件

分类	项目	内容	备注
页眉	终端ID	X (7) 9 (6)	记录产品的序列号
	预约		未使用
数据 1 - 数据n			每分钟的测量数据项目用「,」区分
页脚	预约		未使用
	发送种类	9 (1)	0: 通常 1: 再送

◆ 数据

数据格式可以在日期和时间之后任意指定。

列	内容	说明
1	日期和时间	测量的日期和时间 (YYYYMMDDhhmm)
2	...	

修订履历

修订日	修订内容
2018年10月	初版

- 我们期待本书的内容能够做到十全十美，但如您发现有不甚明确的地方或有任何疑问，都可与购买本产品的零售店联系。

CONTEC CO., LTD. 3-9-31, Himesato, Nishiyodogawa-ku, Osaka 555-0025, Japan

<https://www.contec.com/>

No part of this document may be copied or reproduced in any form by any means without prior written consent of CONTEC CO., LTD.

PAC系列 系统设置指南

NA06448 (LYWJ971) [10312018]

October 2018 Edition