KinCon-8000 QUICK START GUIDE

(Version 1.03)

Warranty

All products manufactured by ICPDAS Inc. are warranted against defective materials for a period of one year from the date of delivery to the original purchaser.

Warning

ICPDAS Inc. assumes no liability for damages consequent to the use of this product. ICPDAS Inc. reserves the right to change this manual at any time without notice. The information furnished by ICPDAS Inc. is believed to be accurate and reliable. However, no responsibility is assumed by ICPDAS Inc. for its use, or for any infringements of patents or other rights of third parties resulting from its use.

Copyright

Trademark

The names used for identification only maybe registered trademarks of their respective companies.

License

The user can use, modify and backup this software on a single machine. The user may not reproduce, transfer or distribute this software, or any copy, in whole or in part.

PACKAGE LIST 5

INTRODUCTION 5

ABOUT THIS MANUAL 7

PART 1: KINCON-8000 OVERVIEW 10

ORDERING INFORMATION 10
HARDWARE SPECIFICATION 11
KinCon-8x4x Front View 12
Definition of Rotary SW 12
Specifications 13
SOFTWARE TOOLS 13
Windows CE Settings 13
WinCon Utility 16
DCON Utility for I-87K Module Settings 24
VCEP 4.2 28
KINCON-8000 CONFIGURATION 29
Register 29
Configuration 31

PART 2: MULTIPROG 32

DEVELOPING A SAMPLE PROJECT 32
PHASE 1 33
CREATING A NEW PROJECT USING THE PROJECT WIZARD 33
STARTING THE PROJECT WIZARD 33
USING THE PROJECT WIZARD 34
PHASE 2 39
DEVELOPING THE LD CODE 39
INSERTING A LD NETWORK 39
DECLARING THE PROPERTIES 40
INSERTING A COUNTER USING THE EDIT WIZARD 43
INSERTING THE COUNTER 'RESET' CONTACT 45
DECLARING THE PROPERTIES OF THE COUNTER'S 46
'RESET' CONTACT 46
INSERTING A SECOND LD NETWORK AND EDITING 50
NETWORK DESCRIPTION COMMENTS 50
PHASE 3 57
COMPILING THE EXAMPLE PROJECT 57
'MAKING' THE PROJECT 57
HANDLING ERRORS AND MESSAGES 59
PHASE 4 60
DOWNLOADING THE PROJECT TO THE IO SIMULATION OR KINCON-8000 60
Download to Simulation 60
Download to KinCon-8000 61
PHASE 5 64
DEBUGGING THE PROJECT 64
DEBUG MODE 64
ONLINE EDITING 65
CROSS REFERENCE WINDOW 68
VARIABLES WATCH WINDOW 69
FORCING AND OVERWRITING 70
BREAKPOINTS 71
PHASE 6 74
PRINTING THE PROJECT DOCUMENTATION 74
SELECTING A PRINTER 74
SETTING THE PAGELAYOUT 74

PRINTING THE PROJECT *75*
PRINT PREVIEW *75*
PRINTING A SINGLE WORKSHEET *76*
USING THE I/O CONFIGURATION 77
Using Simulation *77*
Using KinCon-8000 .... *79*
CREATING AN USER DEFINED FUNCTION 82
CHANGING THE TASK CYCLE TIME 90
USING RETAIN VARIABLE 91
USING MODBUS TCP SLAVE 92
USING MODBUS RTU SLAVE 93
USING MODBUS TCP/RTU MASTER 93

PART 3: THE OPC SERVER 97

INTRODUCTION 97
ADDING AN OPC RESOURCE 98
On PC: *98*
On KinCon-8000: *99*
GENERATING THE CSV FILE 100
PREPARING AND DOWNLOADING THE PROJECT 101
WITH OPC DATA 101
USING THE OPC TEST CLIENT 103

PART 4: PROVISIT 106

PREPARING THE SAMPLE PROJECT FOR THE 106
VISUALIZATION 106
DESIGNING A VISUALIZATION PROJECT 110
CREATING A NEW VISUALIZATION PROJECT *110*
SETTING THE VISUALIZATION SCREEN PROPERTIES *111*
VISUALIZING THE 'ACTUAL_TIME' VARIABLE BY A *111*
DYNAMIC RECTANGLE *111*
VISUALIZING THE 'VISU_MOTOR_START' CONTACT BY *116*
A LIBRARY PUSH BUTTON *116*
VISUALIZING THE 'VISU_EMERGENCY_STOP' *118*
CONTACT BY A LIBRARY EMERGENCY SWITCH *118*
VISUALIZING THE VARIABLE 'PRESSED' BY A LIBRARY *120*
LCD ELEMENT *120*
VISUALIZING THE 'MOTOR' COIL BY A LIBRARY LED *121*
VISUALIZING THE RUNNING MOTOR USING A SELF *122*
DESIGNED OBJECT AND A SCRIPT *122*
SWITCHING THE VISUALIZATION TO RUNTIME 128
DOWNLOADING PROJECT TO KINCON-8000 130

APPENDIX 132

IEC PROJECT COMPONENTS IN THE 132
PROGRAMMING SYSTEM 132
PROGRAM ORGANIZATION UNITS (POUS) *132*
INSTANTIATION OF POUS AND FUNCTION BLOCKS 134
VARIABLES AND DATA TYPES 134
VARIABLE TYPES *134*
VARIABLES ADDRESSES *135*
DATA TYPES *136*
DRIVER PARAMETERS 137
For Embedded I-8K/I87K Modules *137*
For Remote I-7K/I87K Modules *141*
MODBUS ADDRESS V.S. INTERNAL ADDRESS 145

MODBUS TCP/RTU MASTER FBS......146
MB_TCPInit......*146*
MB_TCPClos......*146*
MB_TCPRCS......*147*
MB_TCPWC......*147*
MB_TCPWCS......*148*
MB_TCPRRS......*149*
MB_TCPWR......*150*
MB_TCPWRS......*150*
MB_RTUInit......*151*
MB_RTUClos......*152*
MB_RTURCS......*152*
MB_RTUWC......*153*
MB_RTUWCS......*154*
MB_RTURRS......*155*
MB_RTUWR......*156*
MB_RTUWRS......*156*
ERROR LIST AND DESCRIPTION......157
DEMO LIST......158

Package List

The package includes the following items:

- One set of KinCon8000 hardware
- One Compact Flash Memory Card for storing system files
- One KW-Software Product CD
- One ICPDAS software utility CD with Software User's Manual included

NOTE

If any of these items are missing or damaged, contact the local distributors for more information. Save the shipping materials and cartons in case you want to ship in the future.

It is recommended to read **README.TXT** first from CD\\README.TXT and to visit http://www.icpdas.com/products/PAC/kincon/indusoft_kincon.htm frequently. We will provide new library, template, and demo.

Ordering Information

Call distributor for details.

INTRODUCTION

The ICP DAS **KinCon-8045/8345/8745** is a Windows based PAC (Programmable Automation Controller). It features with Windows CE Operating System and supports KW-Software solution including ProConOS, OPC Server, and ProVisIT(RT). The integration of SoftLogic and HMI makes KinCon-8000 to be a real-time multitasking PAC which combines the feature of IPC and PLC. The description of MultiProg, ProVisIT and OPC Server is as below:

MULTIPROG is a standard programming system for IEC designed PLCs and traditional PLCs. It is based on the standard IEC 61131-3 and includes the full range of IEC features.

The programming system is based on a modern 32 bit windows technology, providing comfortable handling including zoom, drag & drop and dockable windows. The system allows the handling of IEC configuration elements, including libraries and provides a powerful debug system. With MULTIPROG all functionalities are easily accessible via the menu and it only takes you a few dialogs to get through project generation. Having finished that, you can immediately start developing your program.

The programming system consists of a PLC independent kernel for programming in the various IEC programming languages. To do so, the textual languages ST and IL as well as the graphical languages LD, FDB and SFC are provided. Each editor provides an Edit Wizard allowing keywords, statements, operators, functions and function blocks to be inserted fast and easy. The Edit Wizard can also be used for declaring data types. Specific parts adapted to the different PLCs complete the independent kernel.

The **OPC Server** was especially designed to enable the communication between any OPC Client (e.g. ProVisIT) and your PLC. It allows an OPC Client to read/write values from/to the PLC in order to visualize or control the running processes.

ProVisIT is a tool for machine visualization purposes. Using it, visualizations are created intuitively within the graphical editor which provides a large number of standard objects and dynamizations for example Size, Position, Rotation, Color changes and different Actions. The assignment and scaling of dynamizations is done easily by drag & drop.

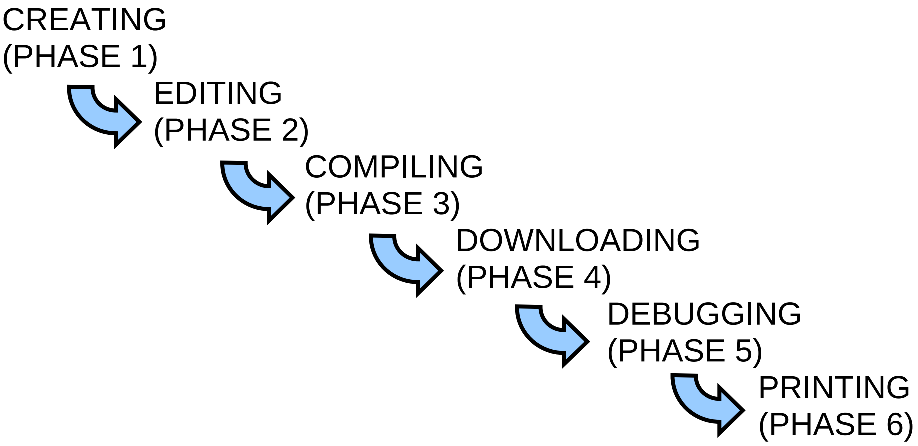
ProVisIT is able to communicate with all controls and devices which provide an OPC Server. Double data input while programming is avoided by browsing OPC variables. Using Visual Basic Script, for example variable values can be calculated and visualization objects can simply be accessed.

ABOUT THIS MANUAL

This manual is divided into four parts. The first part describes some necessary configurations including hardware setting, and software tool using. The second to fourth parts each describing one separate tool: MULTIPROG, OPC Server and ProVisIT.

Part 1: KinCon Overview
The first part lists the order information of KinCon-8000 introducing hardware specifications of KinCon-8000 and the instruction of software tools (WinCon Utility, DCON Utility and VCEP) for warm start of the second to fourth parts.

Part 2: MULTIPROG
The second part provides step by step instructions for developing, editing and running a sample Ladder Diagram (LD) program using MULTIPROG. The development of the sample project is divided into several phases as shown in the following figure:



CREATING
(PHASE 1)

EDITING
(PHASE 2)

COMPILING
(PHASE 3)

DOWNLOADING
(PHASE 4)

DEBUGGING
(PHASE 5)

PRINTING
(PHASE 6)

Each phase will be described in detail and without any gap – from project generation to project documentation.

The example application explained in this manual is a basic motor control circuit. The logic requires the operator to press the start button three times to start the motor. The motor stops running after 20 seconds.

Part 3: OPC Server
In the third manual part starting at page 62, we are going to deal with the OPC Server which enables the communication between any OPC client (the visualization in our case) and the PLC. The manual describes how to configure the server.

Part 4: ProVisIT
The fourth part of this manual starting at page 70 describes the visualization

software ProVisIT. You will find step by step descriptions proving the effectiveness by means of a sample visualization project: We are going to visualize the MULTIPROG demo project developed in part 2 of the manual.

After working through this Quick Start Guide you will be familiar
with the main features of these three highly capable tools and relative tools on KinCon.

SYSTEM REQUIREMENTS

HARDWARE REQUIREMENTS

Device	**Minimum**	**Recommended**
IBM compatible PC with Pentium processor	Pentium II 350 MHz	Pentium III 500 MHz
System RAM	64 MB	128 MB
Hard disk	200 MB free memory space	
CD ROM drive	Required	
VGA Monitor Color settings Resolution	256 colors 800 x 600	True color 1024 x 768
RS232 interface	Optional	
Mouse	Recommended for MULTIPROG Required for ProVisIT	

SOFTWARE REQUIREMENTS

- ◆ Microsoft Windows 95/98 or Windows NT 4.0 SP5 or Windows 2000 SP2 or Windows XP
- ◆ Microsoft Internet Explorer 4.02

INSTALLING THE SOFTWARE

To install the software, insert the KW-Software product CD in your CD ROM drive.

Select the corresponding entry on the product CD start page to launch the Installation Wizard which will guide you through the installation process of each tool. If the autorun feature is deactivated on your PC you can alternatively browse the CD contents and execute the setup files one after the other.

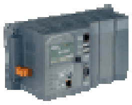
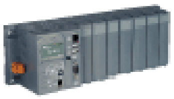
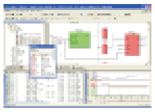
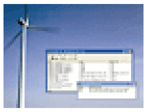
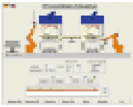
Install the tools in the following order:

- MULTIPROG
- OPC Server
- ProVisIT

Detailed information about installing and registering the tools can be found in the Installation Guide which is also available on the product CD.

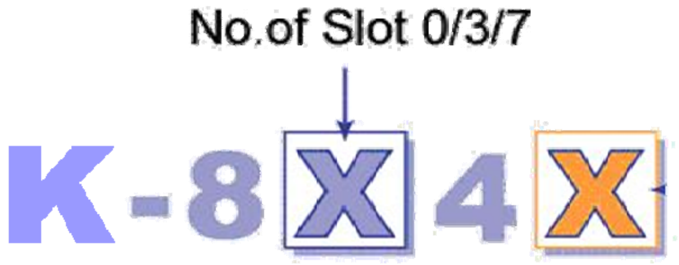
PART 1: KinCon-8000 Overview

Ordering Information

	K-8045	0-slot KinCon PAC, dual Ethernet dual USB, support KW-software
	K-8345	3-slot KinCon PAC, dual Ethernet dual USB, support KW-software
	K-8745	7-slot KinCon PAC, dual Ethernet dual USB, support KW-software
	GA-700YY-UOM / USB	7” TFT RS-232 / USB LCD Monitor w/ VGA & Touch Panel
	S-256 / S-512	256K / 512Kbytes battery backup SRAM (K-8045 doesn’t support S-256 / S-512)
	MULTIPROG 4.0	KW-Software MULTIPROG 4.0 IEC 61131 programming system, with 6 PLC languages
	OPC-Server 2.0	KW-Software OPC-Server 2.0，OPC software
	ProVisIT 2.3	KW-Software ProVisIT 2.3 machine visualization

- S-256/S-512 is needed for retain variables in KinCon-8000
- MULTIPROG is needed for softlogic programming
- MULTIPROG, OPC, and ProVisIT are all needed for softlogic programming and embedded HMI design

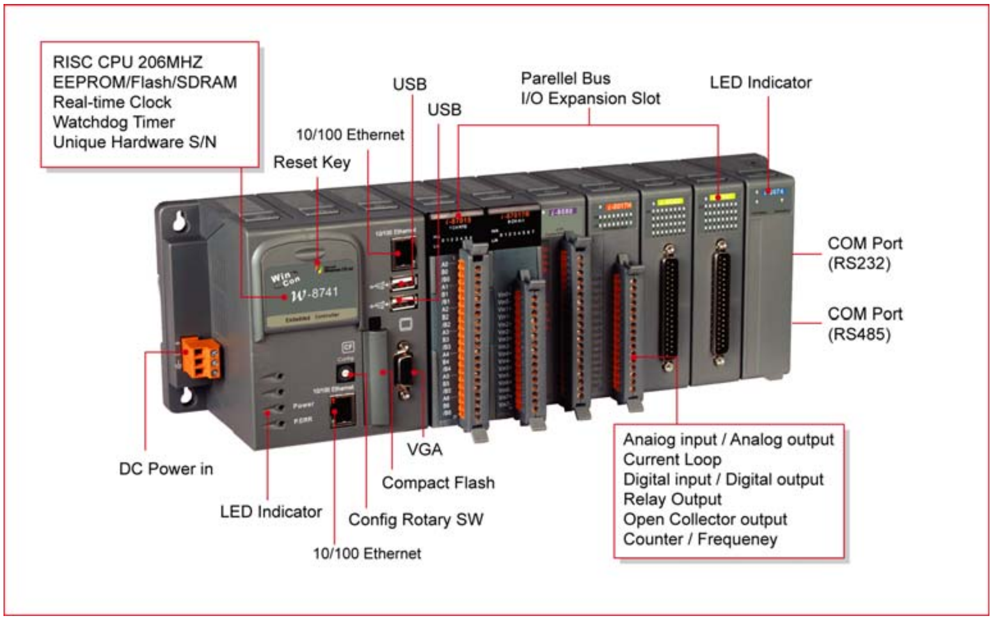
Hardware Specification



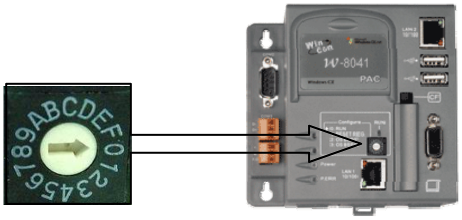
The model type of KinCon-8000 is ruled as K-8X4X, as shown in the above figure. The Second number shows the slot numbers coming with the main controller unit. Currently, we provide three types of 0,3 and 7 slots. The last number demonstrates the application platform. Number 5 is for general KW-Software solution. It will have different number for different application in the future. For more detail products specification, please refer to the following product model table.

Model	Description	CPU Speed	Embedded OS	Slot	Flash	SDRAM	Peripherals
K-8045	KW-Software Embedded Controller	206MHz	Windows CE .NET 4.1	0	32 MB	64MB	10/100BaseT Ethernet Port×2 VGA Port×1 CF Slot×1 USB×2 RS-232×1 RS-485×1 FRnet×1(Option)
K-8345				3			
K-8745				7			

KinCon-8x4x Front View



Definition of Rotary SW



Position	Mode
0	Normal Mode
1	Clear Registry
2	OS updated by PB
3~7	Reserved
8-F	User defined

Always keep rotary SW at "0" position (normal mode), except :

- To clear registry, rotate to "1"position and power on KinCon then wait for 3 sec , KinCon will recover its registry to factory (default) setting.
- If user wants to update OS image by platform builder, please rotate to "2" position.

For detail, please refer to WinCon Getting Start user manual.

Specifications

Main Control unit	
■ Intel Strong ARM CPU, 206 MHz ■ SRAM : 64M bytes ■ Flash RAM : 32M bytes ■ EEPROM : 16K bytes ■ 64-bit hardware unique serial number ■ Built-in Watchdog Timer ■ Real Time Clock	■ 1 VGA port : 320×240×16 to 1024×768×16 Default is 640×480×16 ■ 1 Compact Flash slot : CF memory card ■ Reset button ■ Power LEDs ■ USB 1.1 host x 2 ■ 10/100 Base T x 2
Cabinet	
■ COM0: Internal use ■ COM1: Serial Control for 87K Series ■ COM2: RS-232 ■ COM3: RS-485 ■ FRnet(option) ■ I/O Expansion Slot : 0 - slot for K-80X5 3 - slot for K-8**3**X5 7 - slot for K-8**7**X5	
■ Power Supply : 20W, Unregulated + 10Vdc +30Vdc	
■ Environment : Operating Temp. : -25℃ to + 75℃ Storage Temp. : -30℃ to +85℃ ■ Humidity : 5~95%	
■ Dimensions : 115.66×110×93.8(none slot) 230.25×110×93.8(3 slot) 354.26×110×93.8(7 slot)	
■ I/O module(optional) I-8000 series modules, which include DI,DO,AO,AI… I-87K series modules, which include DI,DO,AO,AI… I-7000 series modules, which include DI,DO,AO,AI… For more information please refer to relative catalog or http://www.icpdas.com	

Software Tools

Windows CE Settings

Setting Up the System Time

You can setup a new date or time in the Windows CE system by using the following steps:

1. Choose **Start → Settings → Control panel** to open the Control panel dialog.

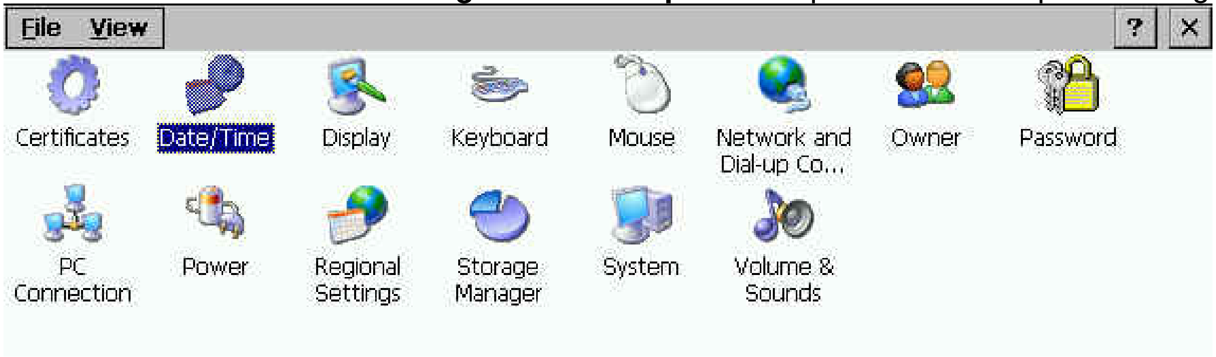
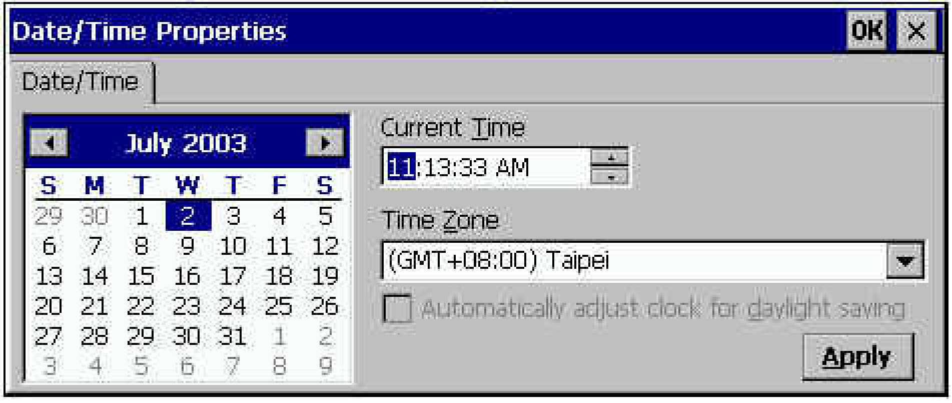


Fig. 2-1

2. Double click the Date/Time icon on the Control panel dialog.



July 2003

S	M	T	W	T	F	S
29	30	1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	31	1	2
3	4	5	6	7	8	9

Current Time: 11:13:33 AM

Time Zone: (GMT+08:00) Taipei

□ Automatically adjust clock for daylight saving

Fig. 2-2

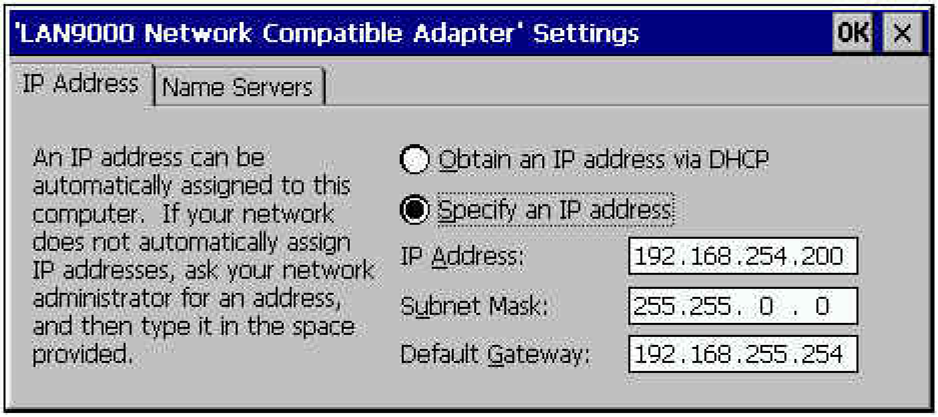
3. When the Date/Time Properties dialog displays, set the date or current Time and click the Apply button to set your system date and time.

> ✍ **Note:** If you have changed any value of the date and time. You must save the registry by means of WinCon Utility tools. For more information about WinCon Utility tools, please refer to the WinCon Utility section.

Setup the network

Generally, most users don't need to setup the network because DHCP is the default setting. However, if your network system does not contain a DHCP server, you need to configure the network setting by using the manual method. The following steps demonstrate the procedure for how to configure the network system.

1. Choose **Start → Settings → Network and Dial_up Connections** on the Windows CE desktop to open this dialog.
2. Double click the LAN90001 icon to open the “LAN9000 Network Compatible Adapter Settings” dialog.



'LAN9000 Network Compatible Adapter' Settings

An IP address can be automatically assigned to this computer. If your network does not automatically assign IP addresses, ask your network administrator for an address, and then type it in the space provided.

○ Obtain an IP address via DHCP

● Specify an IP address

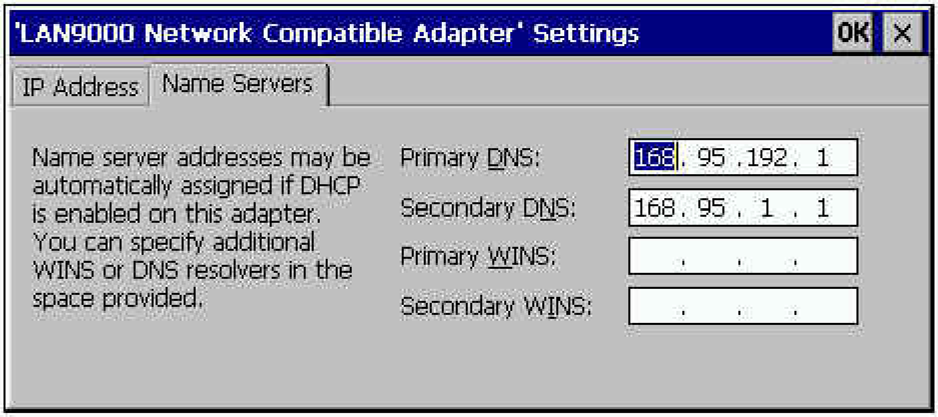
IP Address: 192.168.254.200

Subnet Mask: 255.255. 0 . 0

Default Gateway: 192.168.255.254

Fig. 2-3

3. When the "LAN9000 Network Compatible Adapter Settings" dialog displays (see figure), click (enable) the "Specify an IP address" radio button in the IP Address tab and type in the IP Address, Subnet Mask, and Default Gateway into the respective fields.
4. Choose the "Name Servers" tab and also type in the Primary DNS, Secondary DNS, Primary WINS, and Secondary WINS into the respective fields, as shown in the figure below.



'LAN9000 Network Compatible Adapter' Settings

Name server addresses may be automatically assigned if DHCP is enabled on this adapter. You can specify additional WINS or DNS resolvers in the space provided.

Primary DNS: 168. 95 .192. 1

Secondary DNS: 168. 95 . 1 . 1

Primary WINS: . . .

Secondary WINS: . . .

Fig. 2-4

5. Click OK.

Note: If you have changed any value of network configuration, you must save the registry by means of WinCon Utility tools. For more information about the WinCon Utility tool, please refer to the WinCon Utility section.

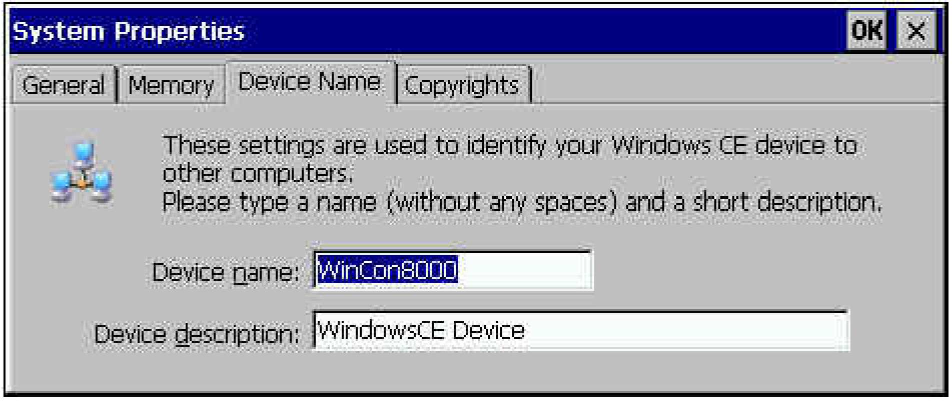
Setting up the Device Name

You can configure Wincon-8000 to have the device name of your choice. To change the device name please refer to the following steps:

1. Choose **Start → Settings → Control panel** to open the Control panel dialog.
2. Double click the System icon on the Control panel dialog to open the System

Properties.

3. When the System Properties dialog is displayed (see figure), select the Device Name tab in the dialog window.



These settings are used to identify your Windows CE device to other computers.
Please type a name (without any spaces) and a short description.

Device name: WinCon8000

Device description: WindowsCE Device

Fig. 2-5

4. Type your preferred Device Name in the Device Name box, and click OK.

> ✍ **Note**: If you have changed any information of the Device Name, you must save the registry by means of WinCon Utility tools. For more information about the WinCon Utility tool, please refer to the WinCon Utility section.

Here, we only provide some demonstrations for configuring your settings. The configuration steps and operation methods are the same as with the windows system. However, you need to keep in mind **"if you have changed any setting on KinCon-8000 embedded controller, you would need to use the WinCon utility to save the current setting into non-volatile internal memory"**. Otherwise, when you restart the system, the setting will not change.

WinCon Utility

The WinCon Utility provides many tools to save/view the system information registry and to setup the HTTP/FTP path and update non-volatile internal memory within the Wincon-8000 embedded controller. This handy utility (WinCon Utility 1.exe located in the Compact Flash/icpdas/Tools directory) should be located in the computer's Program group. Therefore, you can launch it on the computer through **Start → Programs → WinCon Utility** menu. The WinCon Utility provides many functions within the following five tabs:

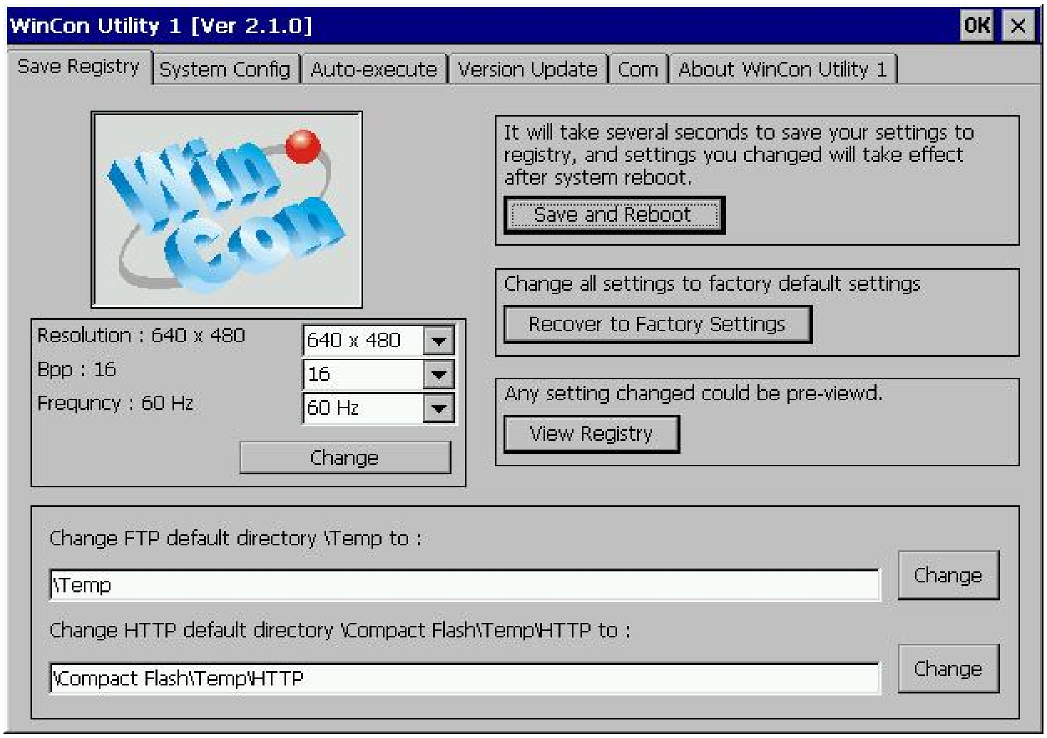
- Save Registry Tab
- System Config Tab
- Auto-execute Tab

- Version Update Tab
- Com Tab
- About WinCon Utility 1 Tab

Save Registry Tab

This tab provides functions to save/view the registry of the systems information and to setup the HTTP/FTP directory path. **It is very important to save the registry when you change any system information. Then you need to click the “Save and Reboot” button to renew the system configuration. If you do not save the current configuration into the registry, you will lose your information settings when you reboot the Wincon-8000**.

> ✍ **Note: The OS image in flash memory will crash if we push the reset or power-off buttons for WinCon-8000 whilst it was writing the registry settings to flash memory. It will take 10-15 seconds to save the registry settings. Add these notes to your user manual because it is very important!**



It will take several seconds to save your settings to registry, and settings you changed will take effect after system reboot.

Change all settings to factory default settings

Any setting changed could be pre-viewd.

Resolution : 640 x 480 — 640 x 480

Bpp : 16 — 16

Frequncy : 60 Hz — 60 Hz

Change FTP default directory \Temp to :

\Temp

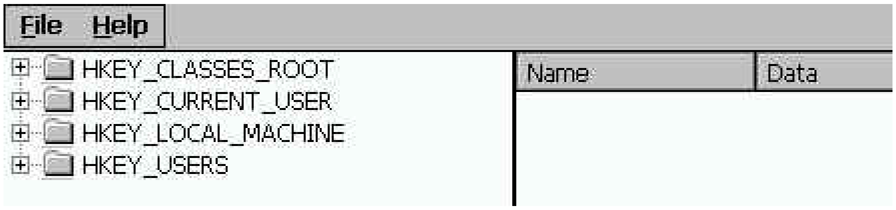
Change HTTP default directory \Compact Flash\Temp\HTTP to :

\Compact Flash\Temp\HTTP

Fig. 2-6

The Save Registry tab includes the following folders:

- **Save and Reboot button:** It will take several seconds to save your settings into registry and **non-volatile internal memory**. You must then reboot the system for the new configuration.
- **Recover to Factory Setting button:** It will take several seconds to clear your registry settings back to **Factory Setting** and Wright to **non-volatile internal memory**. You must then reboot the system for the new configuration.
- **View Registry button:** Any settings are changed in the KinCon embedded controller can be pre-viewed by using this function. It is just like the regedit function in the windows system that you are very familiar with (shown in below figure).



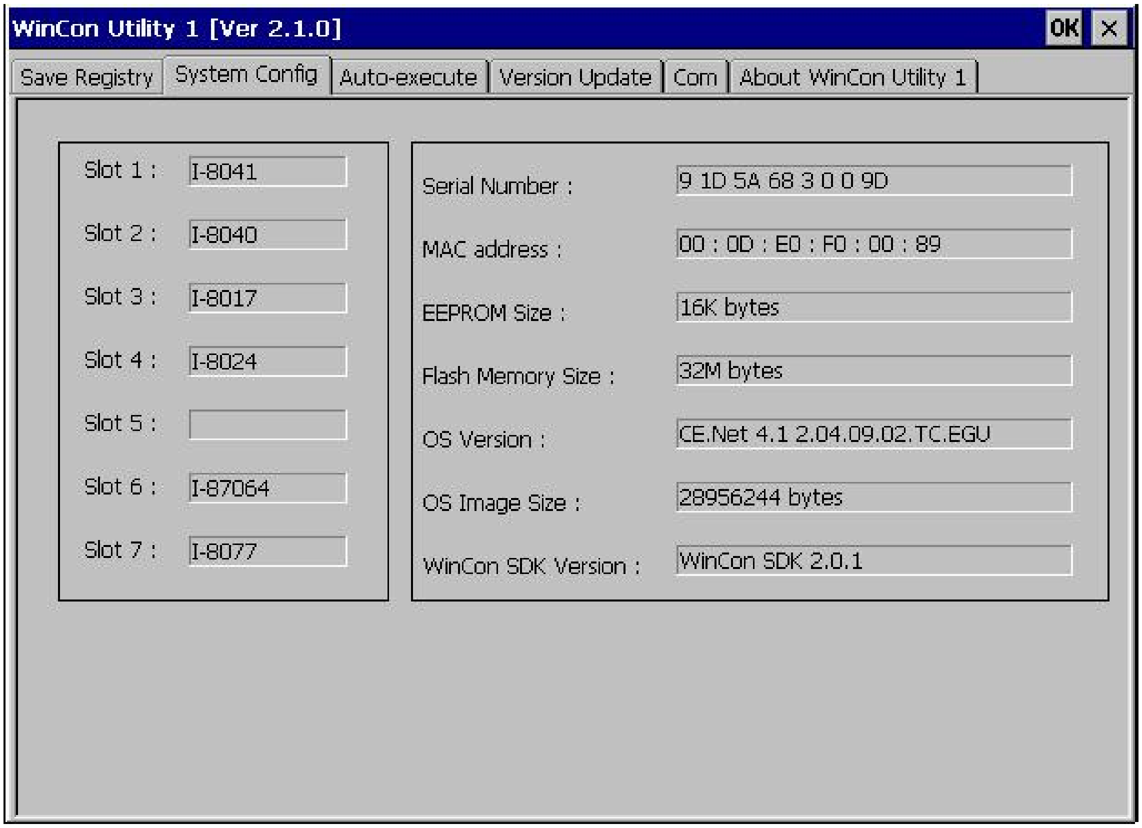
HKEY_CLASSES_ROOT
HKEY_CURRENT_USER
HKEY_LOCAL_MACHINE
HKEY_USERS

Fig. 2-7

- **Change the VGA resolution box:** You can setting the VGA Resolution to 320x240,640x480,800x600 or 1024x768, and 2,4,8,16 bits color (Bpp),the monitor reflash Frequncy for normal TFT LCD setting is 60 Hz.
- **Change FTP default directory to box:** Enter a FTP default directory path and click change button to setup the defined path to the ftp server.
- **Change HTTP default directory to box:** Enter a HTTP default directory path and click on the change button to setup the defined path for the web server.

System Config Tab

The System Config tab allows you to view the information in the KinCon-8000 embedded controller system.



Slot 1 : I-8041

Slot 2 : I-8040

Slot 3 : I-8017

Slot 4 : I-8024

Slot 5 :

Slot 6 : I-87064

Slot 7 : I-8077

Serial Number : 9 1D 5A 68 3 0 0 9D

MAC address : 00 : 0D : E0 : F0 : 00 : 89

EEPROM Size : 16K bytes

Flash Memory Size : 32M bytes

OS Version : CE.Net 4.1 2.04.09.02.TC.EGU

OS Image Size : 28956244 bytes

WinCon SDK Version : WinCon SDK 2.0.1

Fig. 2-8

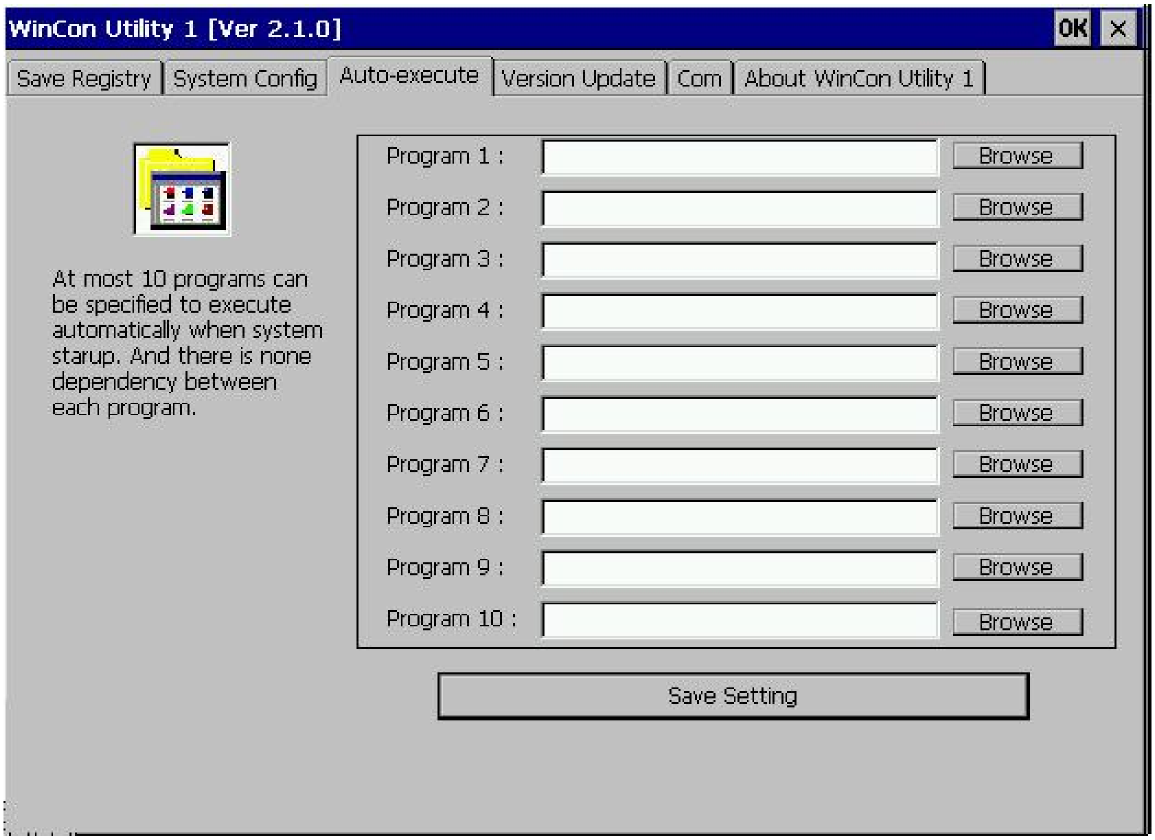
This tab includes the following folders:

- **Slot 1~7 box:** The Slot1~7 fields display the module names plugged in the KinCon-8000.
- **Serial Number box:** This field displays the serial number of the KinCon-8000.
- **MAC address:** The field displays the physical address of Ethernet port. (For K-8X4X, upper field displays 1st Ethernet MAC address, Lower field displays 2nd Ethernet MAC address)
- **EEPROM Size box:** This field displays the EEPROM size of the KinCon-8000.
- **Flash Memory Size box:** This field displays the Flash memory size of the KinCon-8000.
- **OS Version box:** This field displays the current operating system.
- **OS Image Size box:** This field displays the size of the current operating system.

- **WinCon SDK Version box:** This field displays the current WinconSDK_DLL version.

Auto-execute Tab

The Auto-execute tab, provides ten execute files, which can be run after the WinCE system has been launched on the WinCon-8000 system. You can set ten execute files through the Browse button on the tab for WinCon Utility, as shown in the below figure. Note that they are executed in order of program 1, program 2, ...



At most 10 programs can be specified to execute automatically when system starup. And there is none dependency between each program.

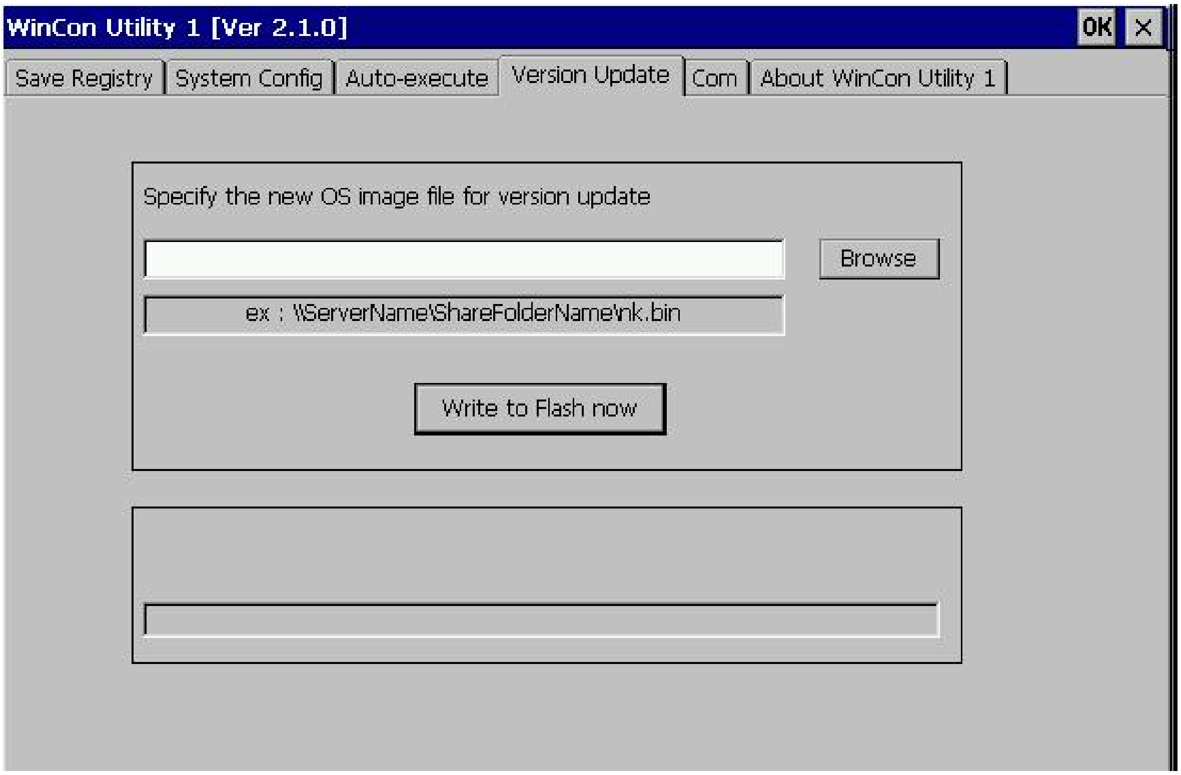
Fig. 2-9

The tab includes the following folders:

- **Program 1~10 boxes:** These files allow one to configure the auto-execute files for KinCon-8000 for when it is started up. You can choose the execute file and file directory path by means of the Browse button.
- **Save Setting button:** If you have changed the settings for the Program 1 ~ 10 field contents, you must then click the Save Setting button before closing the WinCon Utility window.

Version Update Tab

The Version Update tab provides the function to be able to update newer versions of the operating system. Users can download the OS image file from the web site: http://www.icpdas.com. You can choose the new OS image file name and directory path with the Browse button. Click the “Write to flash now” button to update the current OS version. It will take ten or more minutes to update your OS to Flash memory, and then reboot your system.



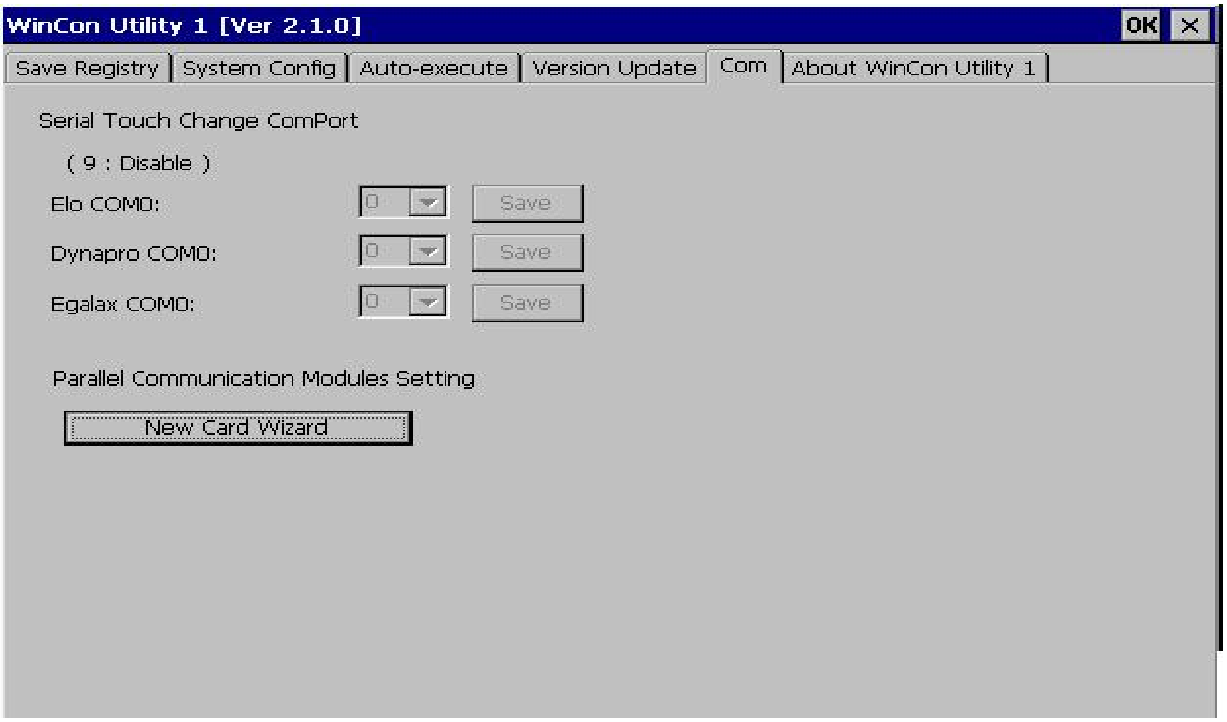
Specify the new OS image file for version update

ex : \\ServerName\ShareFolderName\nk.bin

Fig. 2-10

ComPort Tab

Fig. 2-12 KinCon-8000 show set the touch screen Com Port No, now we can support ELO,3COM Dynapro,EGALAX….,Please plug in the right Com Port No。



Elo COM0: 0 Save

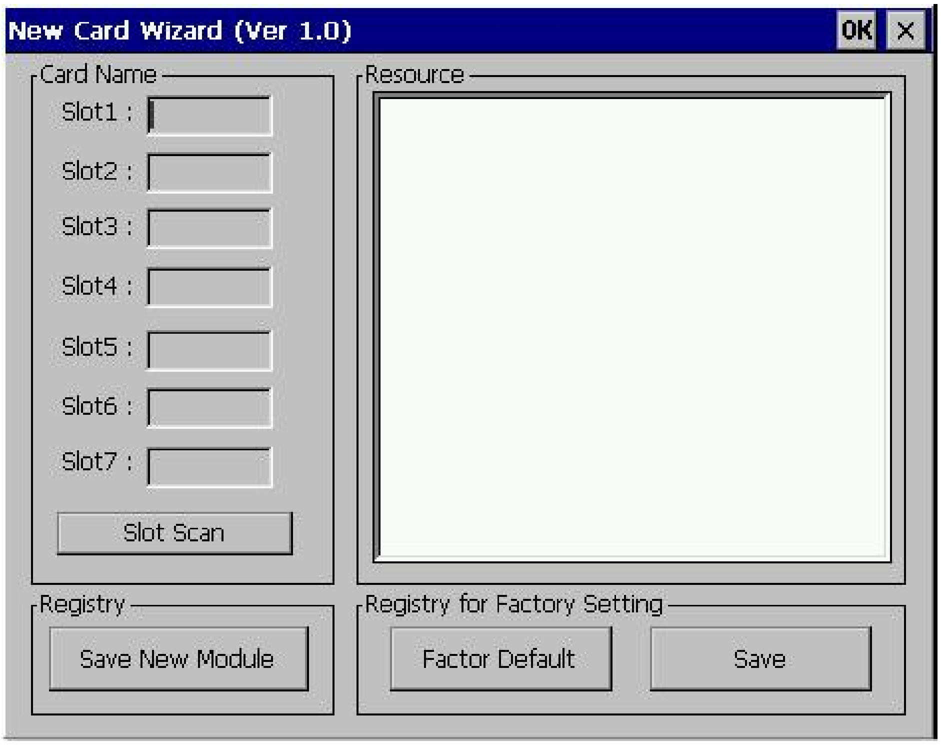
Dynapro COM0: 0 Save

Egalax COM0: 0 Save

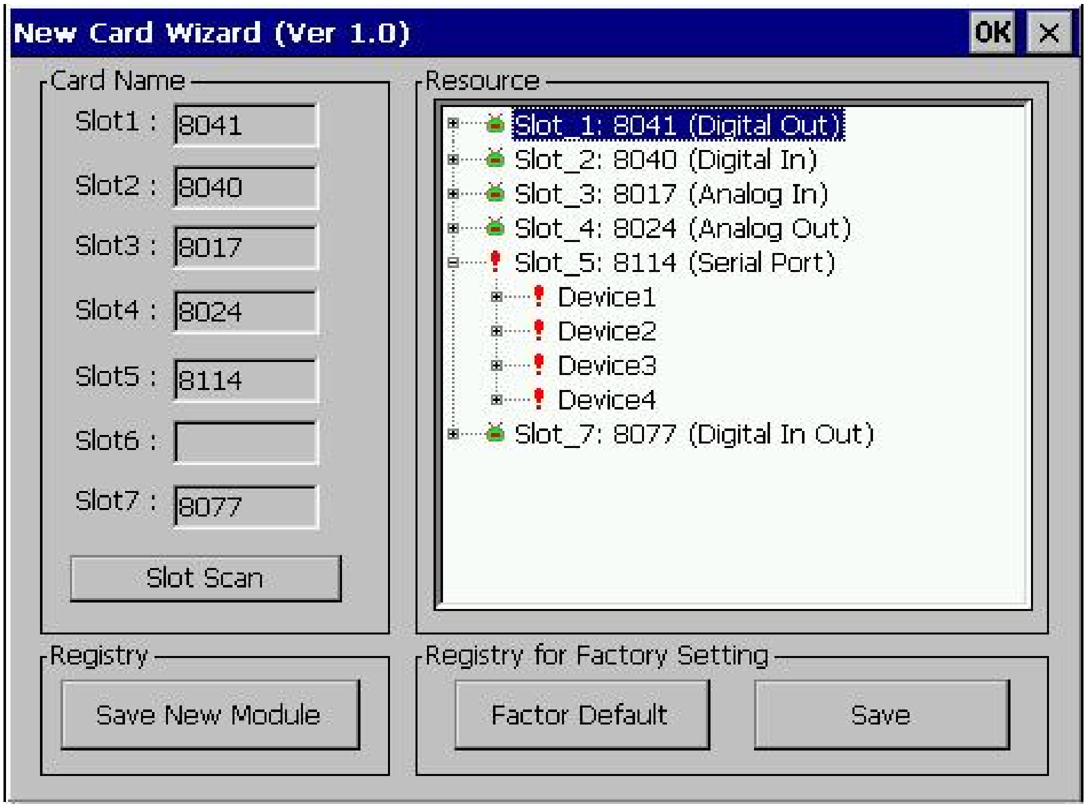
Fig. 2-11

Setting the I-81XX Serial Port

1. To click New Card Wizard button and show the New Card Wizard Window:



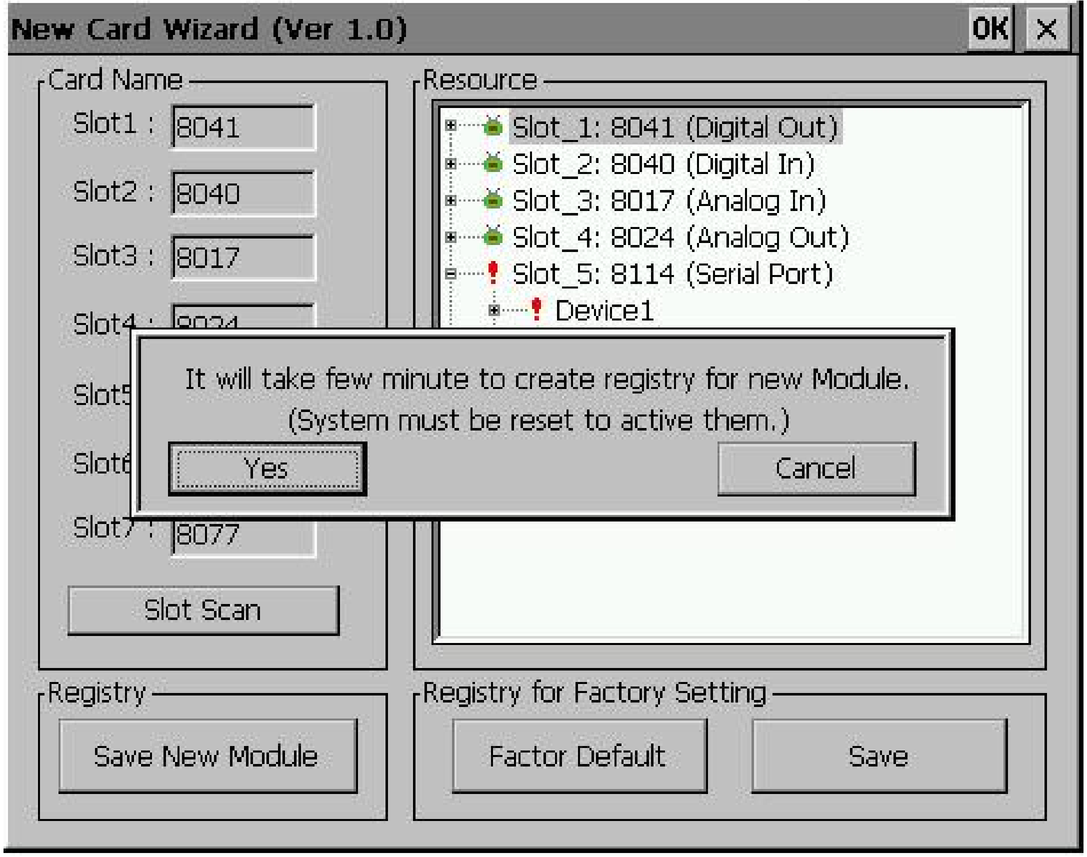
2. To click Slot Scan button and show all Cards in system:



Slot1 : 8041
Slot2 : 8040
Slot3 : 8017
Slot4 : 8024
Slot5 : 8114
Slot6 :
Slot7 : 8077

- Slot_1: 8041 (Digital Out)
- Slot_2: 8040 (Digital In)
- Slot_3: 8017 (Analog In)
- Slot_4: 8024 (Analog Out)
- Slot_5: 8114 (Serial Port)
  - Device1
  - Device2
  - Device3
  - Device4
- Slot_7: 8077 (Digital In Out)

3. To click Save New Module button and save the setting:

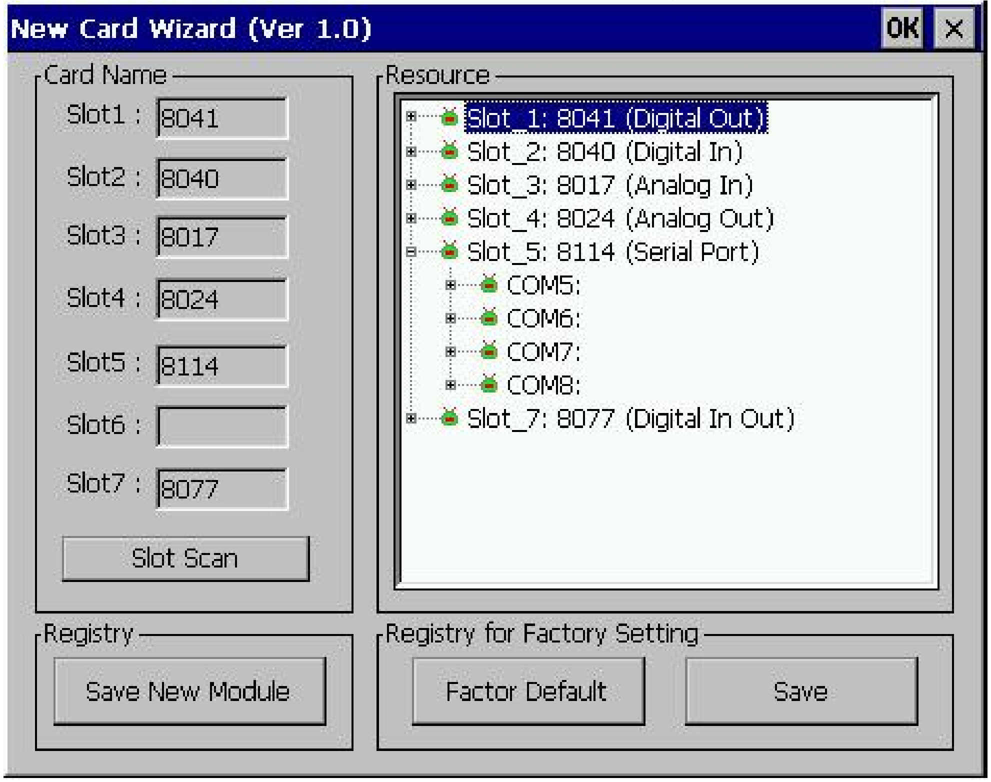


Slot1 : 8041
Slot2 : 8040
Slot3 : 8017
Slot4 : 8024
Slot5
Slot6
Slot7 : 8077

- Slot_1: 8041 (Digital Out)
- Slot_2: 8040 (Digital In)
- Slot_3: 8017 (Analog In)
- Slot_4: 8024 (Analog Out)
- Slot_5: 8114 (Serial Port)
  - Device1

It will take few minute to create registry for new Module.
(System must be reset to active them.)

4. To click Yes button and reset to finish adding Com Port.



Slot1 : 8041
Slot2 : 8040
Slot3 : 8017
Slot4 : 8024
Slot5 : 8114
Slot6 :
Slot7 : 8077

```
Slot_1: 8041 (Digital Out)
Slot_2: 8040 (Digital In)
Slot_3: 8017 (Analog In)
Slot_4: 8024 (Analog Out)
Slot_5: 8114 (Serial Port)
    COM5:
    COM6:
    COM7:
    COM8:
Slot_7: 8077 (Digital In Out)
```

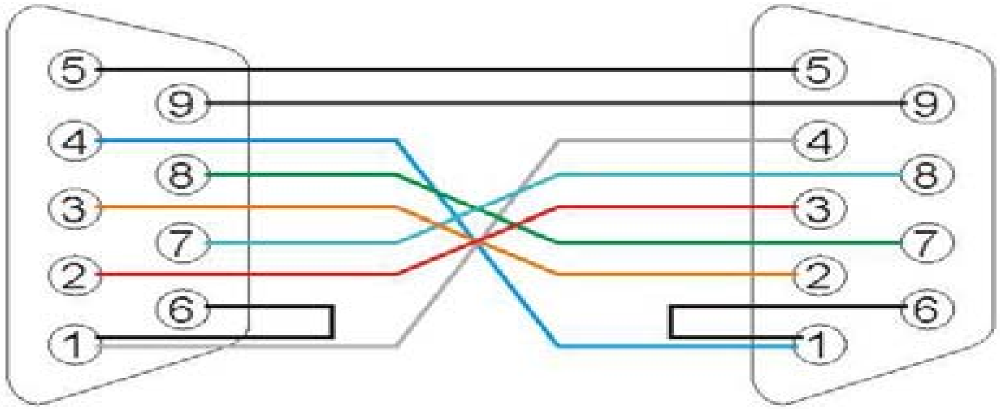
About WinCon Utility 1 Tab

This tab provides an easy function to hyperlink to the ICPDAS World Wide Web site http://www.icpdas.com. This is the best place to go for the latest developments, and support information, application stories, and product news.

DCON Utility for I-87K Module Settings

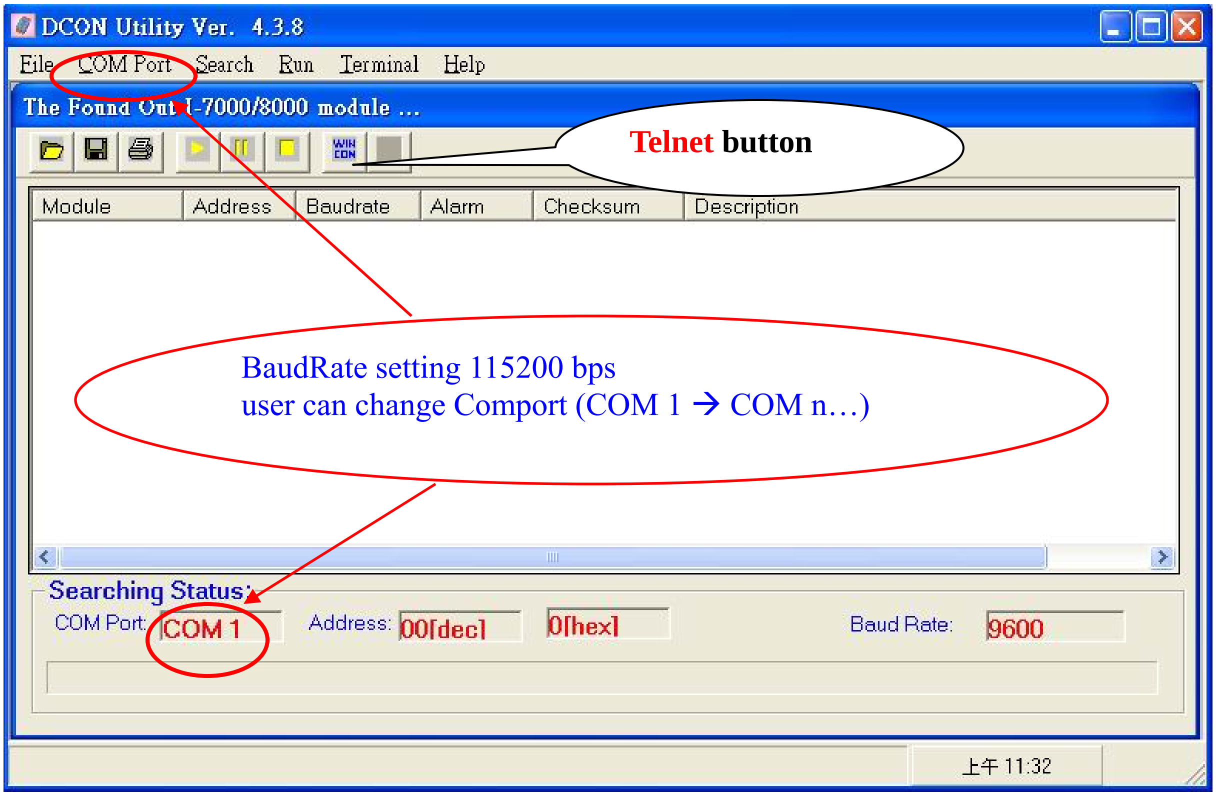
When the module of I-87K is put in the slots of KinCon-8000，it can be set up through the **DCON_Utility** of PC。The method is easy to use。 When you do it orderly，you will find it is easy to set up the parameter of the I-87K module。

1. You must prepare for a cable(CAT6) to start the DCON_CE of KinCon for PC through ethernet (telnet to use)。 To prepare one line of **Full Null Modem** and connect the **Comport** of PC and the **COM2** of WinCon-8000 (data transmission)。



DB-9-1		DB9-2	
Receive Data	**2**	**3**	**Transmit Data**
Transmit Data	**3**	**2**	**Receive Data**
Data Terminal Ready	**4**	**6、1**	**Data Set Ready 、Carrier Detect**
Signal Ground	**5**	**5**	**Signal Ground**
Data Set Ready 、Carrier Detect	**6、1**	**4**	**Data Terminal Ready**
Request to Send	**7**	**8**	**Clear to Send**
Clear to Send	**8**	**7**	**Request to Send**

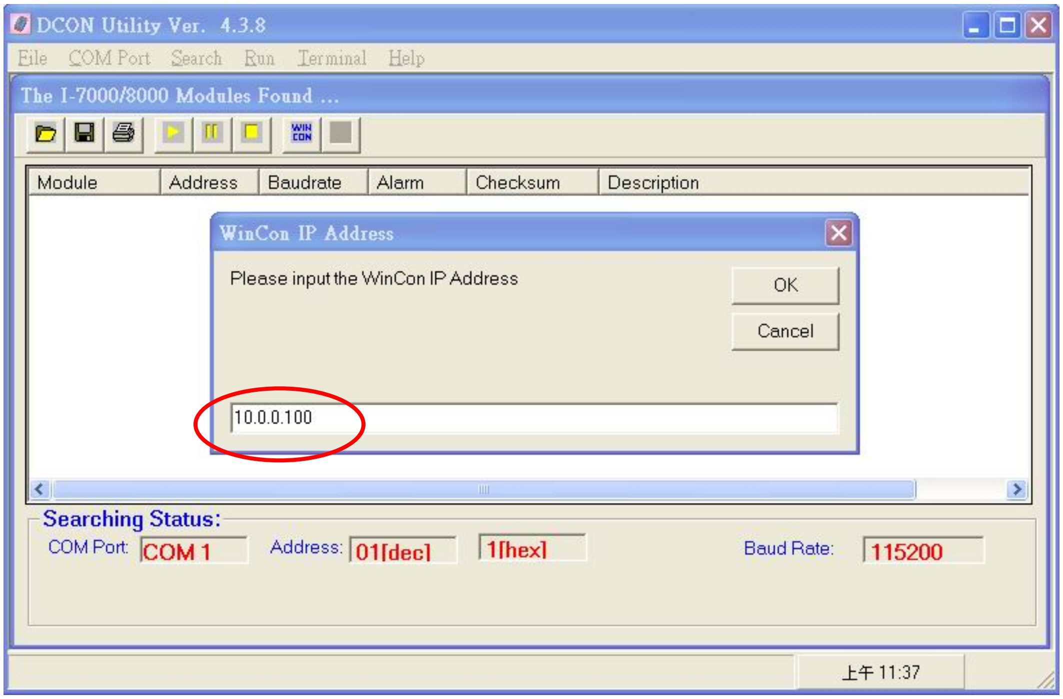
2. To confirm the DCON_CE_V200.exe in the folder of KinCon-8000 **\Compact Flash\ICPDAS\Tools**
3. To enforce the **DCON_Utility V4.3.8(later vision)** of PC and press **the connective button of Telnet** to open the **DCON_CE_V200 V2.0.0** (WinCon-8000)。



BaudRate setting 115200 bps
user can change Comport (COM 1 → COM n…)

Searching Status:
COM Port: COM 1
Address: 00[dec] 0[hex]
Baud Rate: 9600

4. To input the **address of IP** of KinCon-8000。

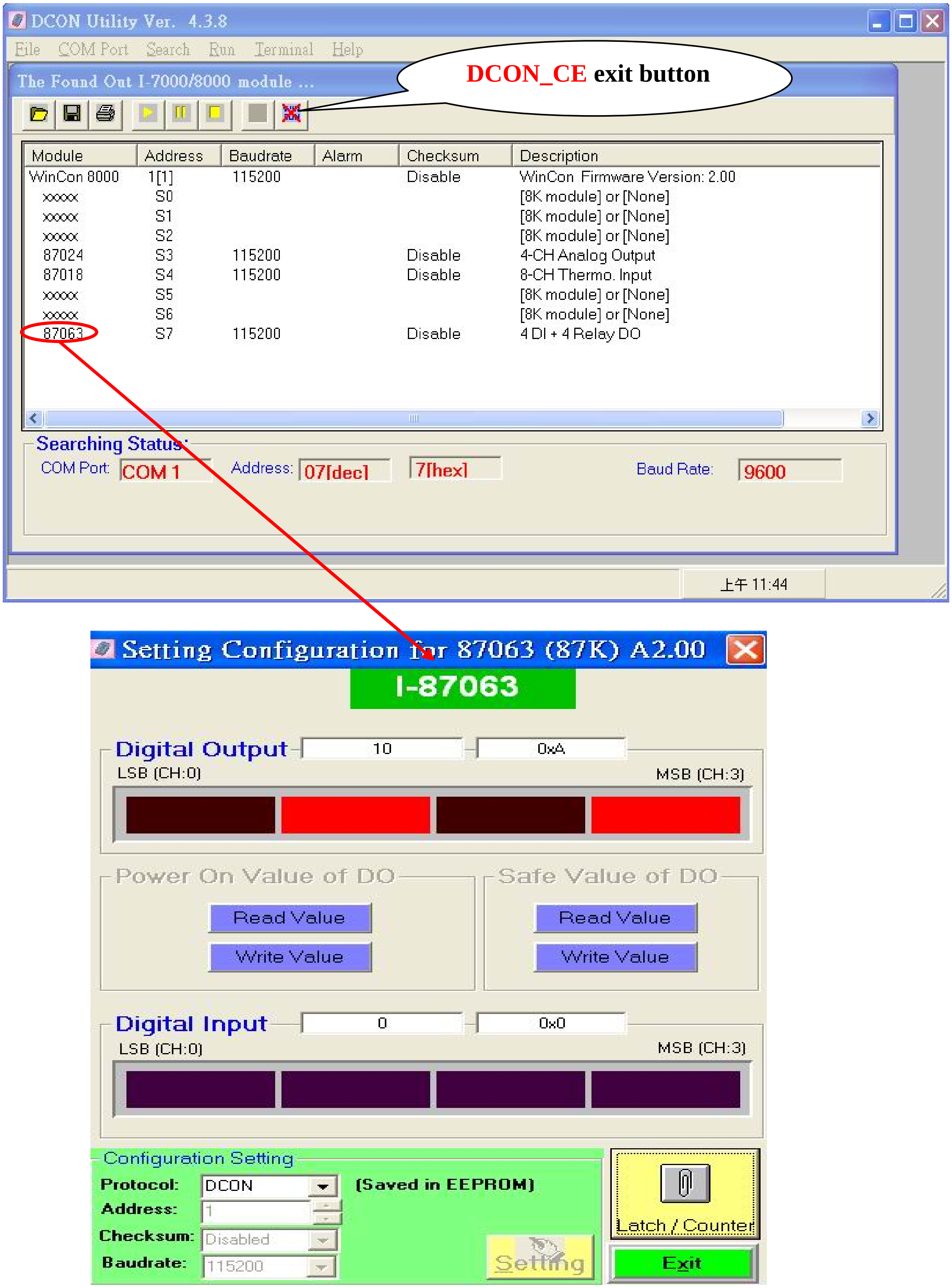


Please input the WinCon IP Address

10.0.0.100

Searching Status:
COM Port: COM 1
Address: 01[dec] 1[hex]
Baud Rate: 115200

5. After connecting the KinCon-8000，refer to the manual of the DCON_Utility.


DCON Utility Ver. 4.3.8
File COM Port Search Run Terminal Help
The Found Out I-7000/8000 module ...

DCON_CE exit button

Module	Address	Baudrate	Alarm	Checksum	Description
WinCon 8000	1[1]	115200		Disable	WinCon Firmware Version: 2.00
xxxxx	S0				[8K module] or [None]
xxxxx	S1				[8K module] or [None]
xxxxx	S2				[8K module] or [None]
87024	S3	115200		Disable	4-CH Analog Output
87018	S4	115200		Disable	8-CH Thermo. Input
xxxxx	S5				[8K module] or [None]
xxxxx	S6				[8K module] or [None]
87063	S7	115200		Disable	4 DI + 4 Relay DO

Searching Status:
COM Port: COM 1 Address: 07[dec] 7[hex] Baud Rate: 9600

上午 11:44

Setting Configuration for 87063 (87K) A2.00

I-87063

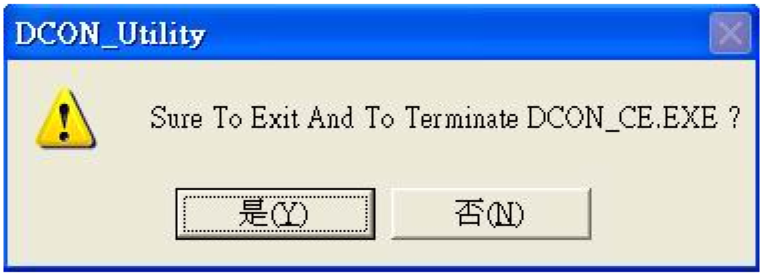
Digital Output 10 0xA
LSB (CH:0) MSB (CH:3)

Digital Input 0 0x0
LSB (CH:0) MSB (CH:3)

Configuration Setting
Protocol: DCON (Saved in EEPROM)
Address: 1
Checksum: Disabled
Baudrate: 115200

6. To press the exit button of **DCON_CE** and it can close the

DCON_CE_V200.exe of KinCon-8000。Moreover，it will exit automatically when the DCON_Utility does not work during the programs automatically scan in **thirty minutes**。

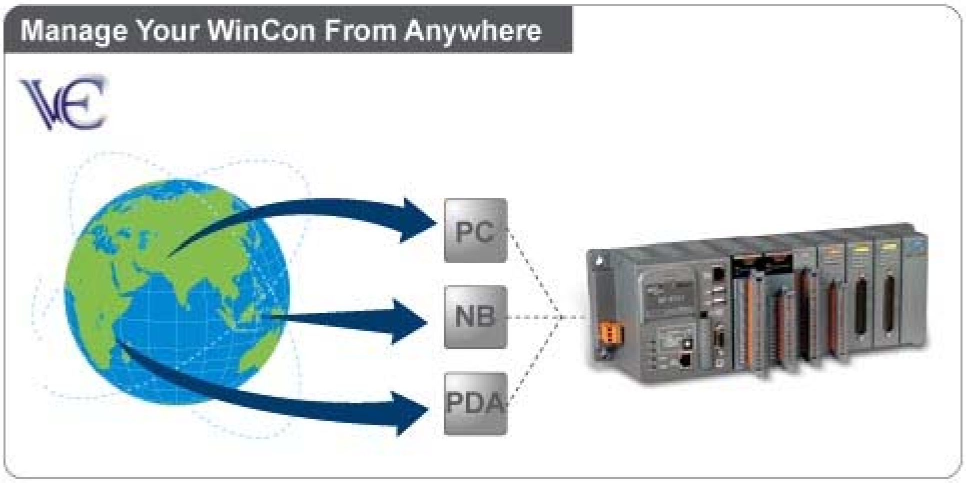

DCON_Utility

Sure To Exit And To Terminate DCON_CE.EXE ?

Please operate the methods according to the steps。If you find the abnormal situation，you can close the DCON_Utility first and then power ON/OFF KinCon-8000。

Web Download :
1) DCON_Utility (PC side) → http://www.icpdas.com/products/wincon/winconutility.htm
2) DCON_CE (WinCon) → http://www.icpdas.com/products/wincon/winconutility.htm

VCEP 4.2



ICP DAS VCEP 4.2 is designed to allow Desktop PC users to remotely manage a broad range of KinCon-8000 from a single management interface. ICP DAS VCEP 4.2 provides secure, remote access to KinCon-8000 and supports network connections over a local area network(LAN), wide area network(WAN), Internet, and direct cable connections using a serial or parallel port. ICP DAS VCEP 4.2 is composed of two main components: The server which runs on the KinCon and the client software which runs on a Desktop PC. Once a connection is established between the client and server, the client will periodically send requests for screen updates and send mouse/key click information to the server. Each video frame is inter-compressed against the previous frame and then intra-compressed with a modified LZW scheme to minimize the amount of data transmitted from server to client.

For more details, please refer to
CD\Virtual_CE_Pro\VCEP_Quick_Started_Manual.pdf

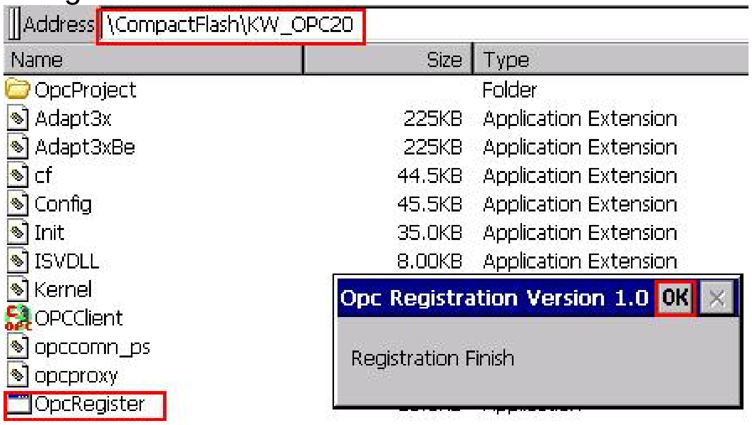
KinCon-8000 Configuration

Register

Each KinCon-8000 has demo version of OPC server and ProVisIT(RT) limited working 60 minutes continuously. If you wants to upgrade to be official version, please purchase the license from ICP DAS distributor. This section describe how to register after you purchase the license of OPC Server and ProVisIT(RT).

OPC Server:

Step1:Execute \CompactFlash\KW_OPC20\OpcRegister.exe，press **OK** when registration finish.



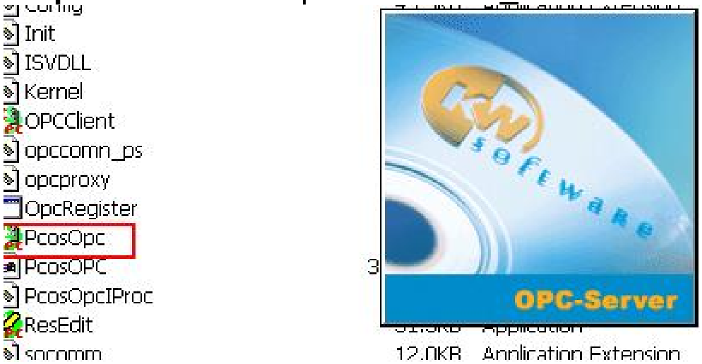
Address \CompactFlash\KW_OPC20

Name	Size	Type
OpcProject		Folder
Adapt3x	225KB	Application Extension
Adapt3xBe	225KB	Application Extension
cf	44.5KB	Application Extension
Config	45.5KB	Application Extension
Init	35.0KB	Application Extension
ISVDLL	8.00KB	Application Extension
Kernel		
OPCClient		
opccomn_ps		
opcproxy		
OpcRegister		

Opc Registration Version 1.0 OK ×

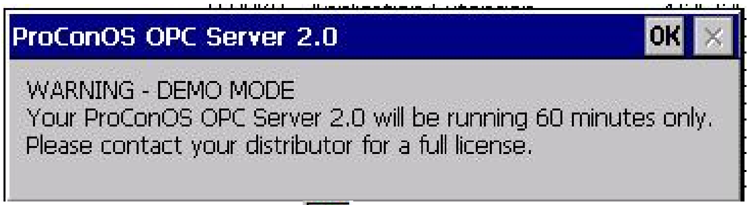
Registration Finish

Step2:Execute \CompactFlash\KW_OPC20\PcosOpc.exe



Config
Init
ISVDLL
Kernel
OPCClient
opccomn_ps
opcproxy
OpcRegister
PcosOpc
PcosOPC
PcosOpcIProc
ResEdit
sncomm 12.0KB Application Extension

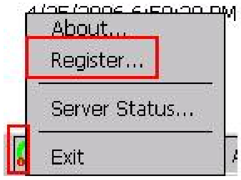
Step3:You can see the prompt '**Demo mode 60 minutes'**. If you just want to try it, please execute 'Save and Reboot' in WinCon Utility. If you have purchased license, please continue to next step.

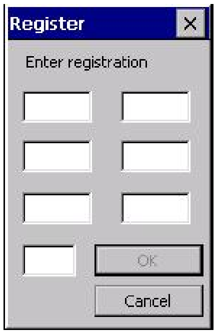


ProConOS OPC Server 2.0

WARNING - DEMO MODE
Your ProConOS OPC Server 2.0 will be running 60 minutes only.
Please contact your distributor for a full license.

Step4:Click the icon at the status bar and choose '**Register**' to enter registration number.

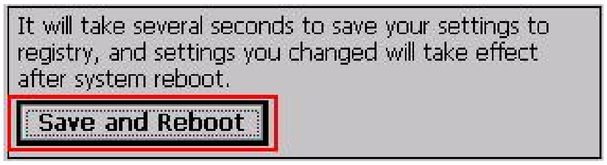




Register

Enter registration

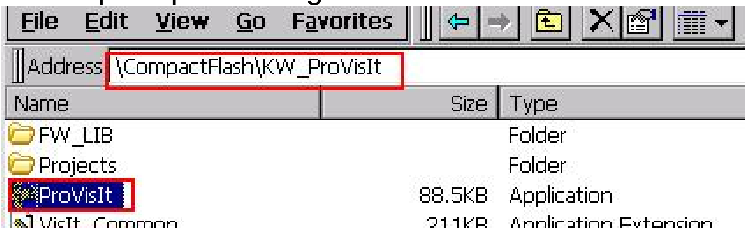
Step5:Enter your license number at the left-top of license paper. And then execute 'Save and Reboot' in WinCon Utility.



It will take several seconds to save your settings to registry, and settings you changed will take effect after system reboot.

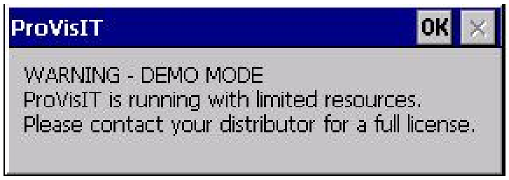
ProVisIT(RT):

Step1:Execute \CompactFlash\KW_ProVisIt\ProVisIt.exe, then you can see the prompt showing '**DEMO MODE**'.



Address \CompactFlash\KW_ProVisIt

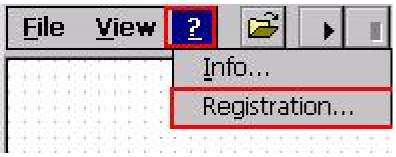
Name	Size	Type
FW_LIB		Folder
Projects		Folder
ProVisIt	88.5KB	Application
VisIt_Common	211KB	Application Extension

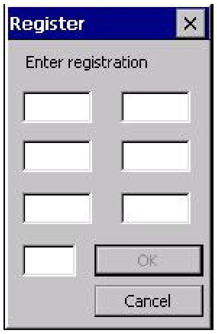


ProVisIT

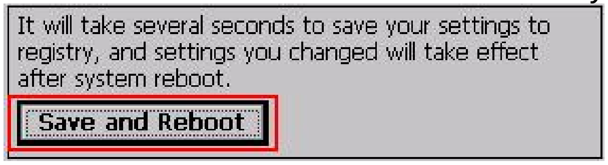
WARNING - DEMO MODE
ProVisIT is running with limited resources.
Please contact your distributor for a full license.

Step2:Execute '**?**' for Registration.





Step3: Enter your license number at the left-top of license paper. And then execute 'Save and Reboot' in WinCon Utility.


It will take several seconds to save your settings to registry, and settings you changed will take effect after system reboot.

Configuration

KinCon-8000 boot file, **\CompactFlash\KW_Pcos\KWBoot.exe**, refers to **\CompactFlash\ICPDAS\Tools\ProCos.bat** and **\CompactFlash\ICPDAS\Tools\ProVisIt.bat** to start ProConOS and ProVisIt(RT). Thus, you can change the context of ProCos.bat to configure your KinCon-8000.

Parameter	description
-S	SRAM size(1~512 KBytes)
-B	Baudrate for Modbus/RTU Slave(4800~115200)
-COM	COM port number(2~9)
-SN	Slave number(1~255)
-ST	System Ticks(2~16)，Recommended: 4

NOTE: "-ST" allows user to decide the system ticks of ProConOS. Smaller "ST" value causes higher system loading. On the contrary, smaller "S" value causes lower system loading.

For example:
General start:

ProCos.bat Context
@echo off \CompactFlash\KW_Pcos\pcwce4.exe

Start with S256 using 5 KBytes:

ProCos.bat Context
@echo off \CompactFlash\KW_Pcos\pcwce4.exe -S 5

Start with S512 using 300 KBytes:

ProCos.bat Context
@echo off \CompactFlash\KW_Pcos\pcwce4.exe -S 300

Start with Modbus/RTU Slave at baudrate:19200 COM port:2 Slave No: 1:

ProCos.bat Context
@echo off \CompactFlash\KW_Pcos\pcwce4.exe –B19200 –COM2 –SN1

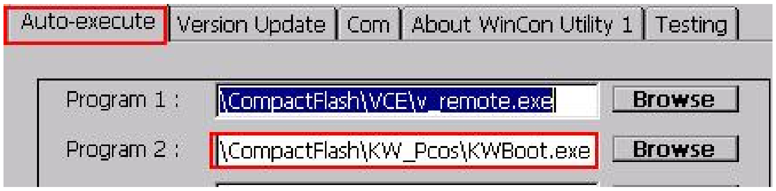
NOTE: Please double check the setting whether it conflicts with I/O configuration of remote devices. COM port can NOT share with another use.

Start with system ticks: 4ms

ProCos.bat Context
@echo off \CompactFlash\KW_Pcos\pcwce4.exe –ST 4

You can mix all parameters together to configure your KinCon-8000.

If you want to start ProConOS and ProVisIt(RT) every time you boot KinCon-8000, please add **KWBoot.exe** to '**Auto-execute**' in **WinCon Utility**.



Program 1 : \CompactFlash\VCE\v_remote.exe

Program 2 : \CompactFlash\KW_Pcos\KWBoot.exe

PART 2: MULTIPROG

DEVELOPING A SAMPLE PROJECT

Start the programming system.

We will now develop a sample project using the programming language Ladder Diagram (LD). In the first phase we create a new, empty project.

In order to get the best possible result we recommend to use the identifiers and names we use in this manual.

PHASE 1
CREATING A NEW PROJECT USING THE PROJECT WIZARD

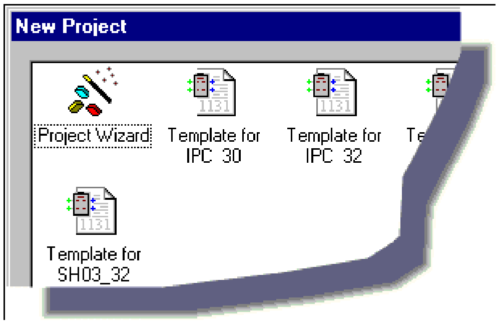
The Project Wizard guides you in 6 steps through the process of creating the new project. Here, you have to define the name and path of the project, the programming language and the type of the PLC used.

STEP 1

STARTING THE PROJECT WIZARD

Click on the 'New Project' icon:

The 'New Project' dialog appears. Double click on the 'Project Wizard' icon:

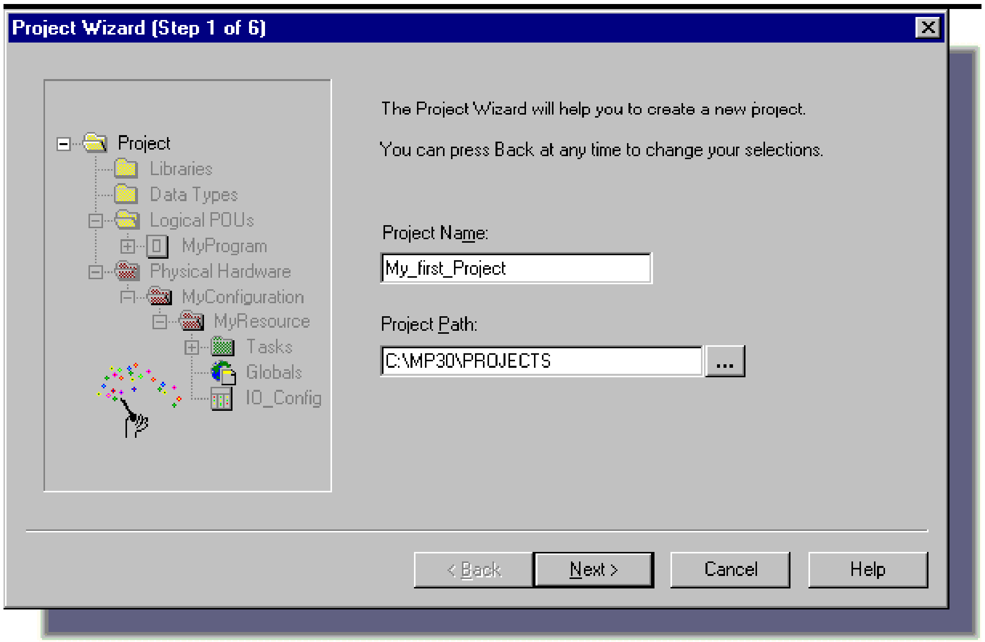


The dialog 'Project Wizard (Step 1 of 6)' appears.

STEP 2

USING THE PROJECT WIZARD

Figure 1: Dialog 'Project Wizard (Step 1 of 6)' for specifying the project name and project path



Project Wizard (Step 1 of 6)

The Project Wizard will help you to create a new project.

You can press Back at any time to change your selections.

Project Name: My_first_Project

Project Path: C:\MP30\PROJECTS

Fill in the dialog fields as follows:

a. Enter the desired project name ('My_first_Project') into the first input field as shown in the figure above. The Project Wizard will save the project to a corresponding file 'My_first_Project.mwt' and create a subfolder of the same name, in which the code bodies, variables' files, etc. will be stored.

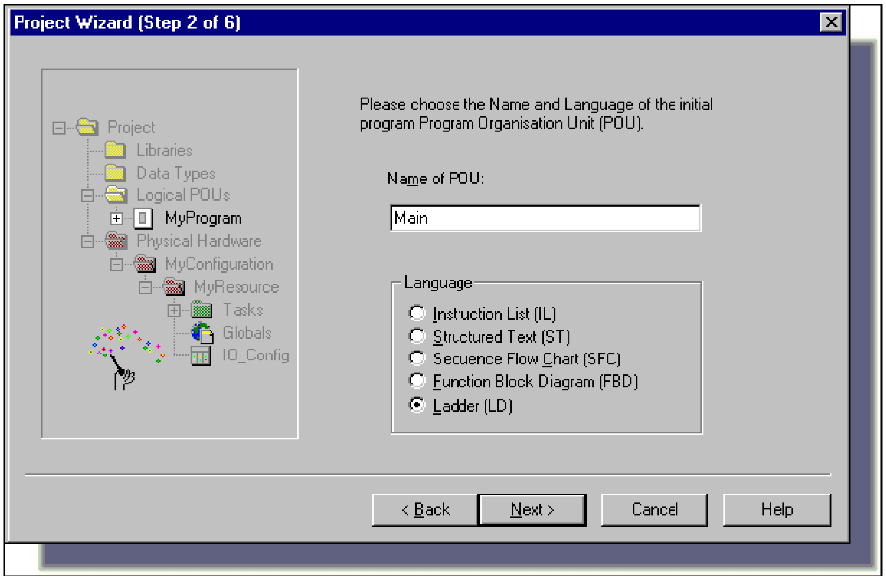
According to the rules for projects, the name and the path of the project must not contain any blanks or special characters. The maximum number of characters for the project name is 24.

The default path for the project has been entered automatically.
If you want to store your project in another path, specify this in the second input field as described below:

b. Click on the browse button:
...
The 'Select Directory' dialog appears.

c. Choose a folder for the new project.
d. Click on the button 'Next' to continue.
Next >
The dialog 'Project Wizard (Step 2 of 6)' appears.

Figure 2:
Dialog 'Project
Wizard
(Step 2 of 6)' for
entering the first
POU
and selecting the
programming
'



Please choose the Name and Language of the initial program Program Organisation Unit (POU).

Name of POU: Main

Language

○ Instruction List (IL)
○ Structured Text (ST)
○ Secuence Flow Chart (SFC)
○ Function Block Diagram (FBD)
● Ladder (LD)

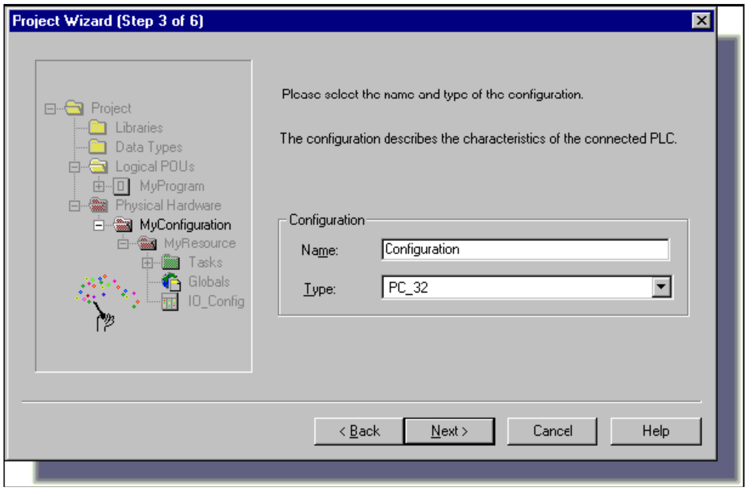
e. Enter the name for the first POU, which will be automatically inserted by the Project Wizard in the project tree when creating the project. Enter 'Main' for our example POU. Choose the language for the first program by activating the corresponding radio button.
As we want to program our sample project in the graphical language Ladder Diagram, select 'Ladder (LD)'.

f. Click 'Next' to continue.

Next >

The dialog 'Project Wizard (Step 3 of 6)' appears.

Figure 3:
Dialog 'Project
Wizard
(Step 3 of 6)' for
selecting the
configuration
name and
type



Please select the name and type of the configuration.

The configuration describes the characteristics of the connected PLC.

Configuration

Name: Configuration

Type: PC_32

g. Enter the desired configuration name into the input field. The configuration can be compared to a programmable controller system, e.g. a rack. In our example we enter 'Configuration'.

h. Select a configuration type for the project. This is necessary because the system generates PLC specific code when compiling the project.
Select the PLC type in the list box. In our example we choose 'IPC_32', so the compiler will generate Intel code for ProConOS 3.2.

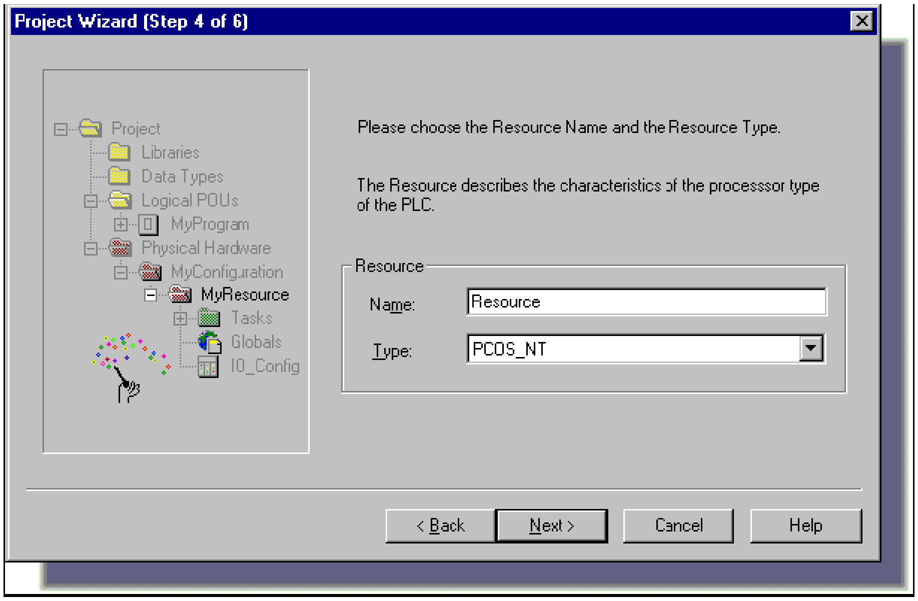
For detailed information about PLC types please refer to the corresponding PLC manual.

i. Click 'Next' to continue.

Next >

The dialog 'Project Wizard (Step 4 of 6)' appears.

Figure 4: Dialog 'Project Wizard (Step 4 of 6)' for selecting the resource name and type


Project Wizard (Step 4 of 6)

Please choose the Resource Name and the Resource Type.

The Resource describes the characteristics of the processor type of the PLC.

Resource

Name: Resource

Type: PCOS_NT

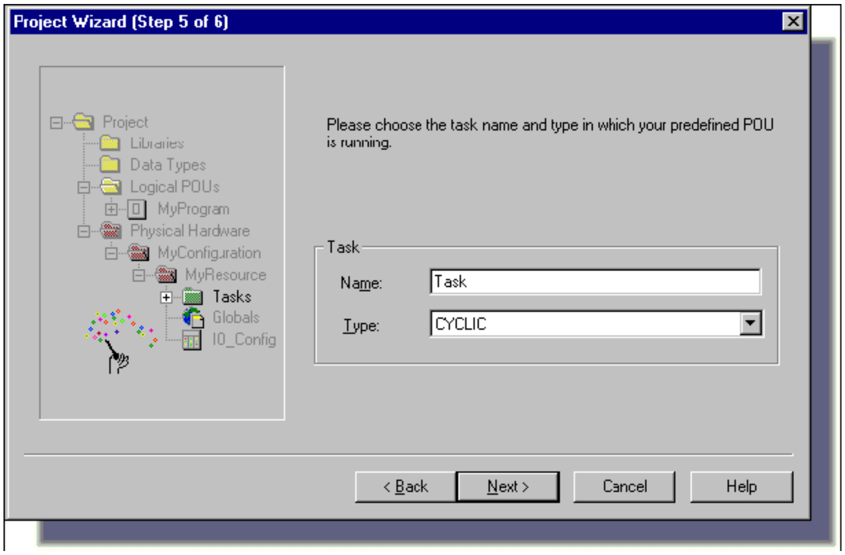
j. Enter a name for the resource (in our example 'Resource'). The resource can be compared to a CPU which can be inserted into the rack (i.e. into the configuration).
In the list box select the resource type. The list box only offers CPU types which belong to the configuration you have defined in the dialog 'Project Wizard (Step 3 of 6)' (refer to Figure 3).

k. Click 'Next' to continue.

Next >

The dialog 'Project Wizard (Step 5 of 6)' appears.

Figure 5: Dialog 'Project Wizard (Step 5 of 6)' for selecting the task name and type



Project Wizard (Step 5 of 6)

Please choose the task name and type in which your predefined POU is running.

Task

Name: Task

Type: CYCLIC

l. Enter the name of the first task into the input field (in our example 'Task').
In the list box select the task type 'CYCLIC'.

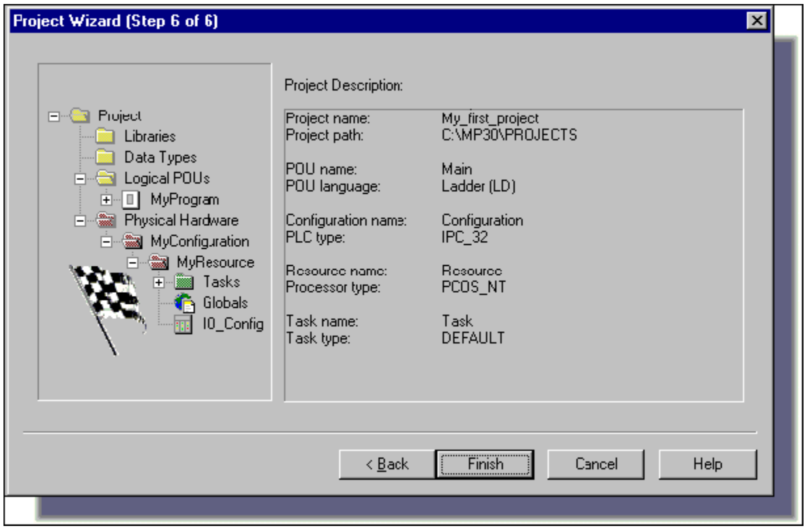
Detailed information about the different task types can be found in the online help system.

m. Click 'Next' to continue.

Next >

The dialog 'Project Wizard (Step 6 of 6)' appears.

Figure 6: Dialog 'Project Wizard (Step 6 of 6)' displaying an overview on the project settings



Project Wizard (Step 6 of 6)

Project Description:

Project name:	My_first_project
Project path:	C:\MP30\PROJECTS
POU name:	Main
POU language:	Ladder (LD)
Configuration name:	Configuration
PLC type:	IPC_32
Resource name:	Resource
Processor type:	PCOS_NT
Task name:	Task
Task type:	DEFAULT

This dialog shows the project description, which is an overview of the settings you have made in the steps 1 to 5. If an invalid name was entered the error message 'Invalid name' appears and the 'Finish' button is greyed out. If this is the case, check the spelling of the suspected identifier.

To correct an error browse to the corresponding step using the 'Back' button. Make sure that the rules for defining identifiers are followed.

n. If no error message occurs click 'Finish':

Finish

The new project will be created and inserted in the project tree window.

In the project tree window, you can see the newly generated project with its POU 'Main' in the subtree 'Logical POUs' and the configuration, resource and task in the subtree 'Physical Hardware'.

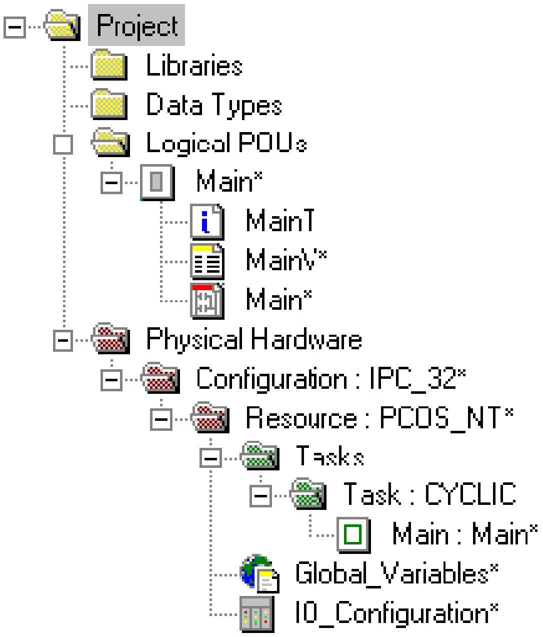


Figure 7: Automatically generated project in the project tree

The empty code body worksheet of the program 'Main' is automatically opened.

PHASE 2
DEVELOPING THE LD CODE

Now that we have created the new project we will start phase 2 and develop the project code.

To do so, we will use the programming language Ladder Diagram (LD). Having finished editing the project code, the project can be compiled, downloaded and debugged.

In the following steps we will explain how to

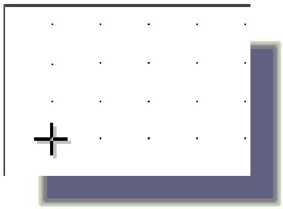
□ insert a first LD network
□ declare the properties of LD objects which are automatically inserted with the first LD network
□ insert and connect a function block in the LD code body worksheet using the Edit Wizard
□ insert and connect a contact in the LD code worksheet using the connection mode
□ declare the properties of contacts and coils
□ insert a second LD network and edit network description Comments

STEP 1

INSERTING A LD NETWORK

We want to insert the first LD network '001':

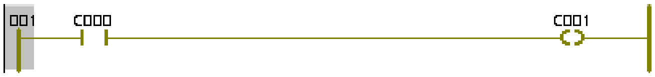
a. Click with the left mouse button into the worksheet to set an insertion mark at the position shown below. Here, the network will be inserted.



b. Click on the 'Contact network' icon to insert the LD network:

c. The network comes with a contact and a coil, and its width is set automatically as well.

Figure 8:
New LD network in the worksheet



STEP 2

DECLARING THE PROPERTIES

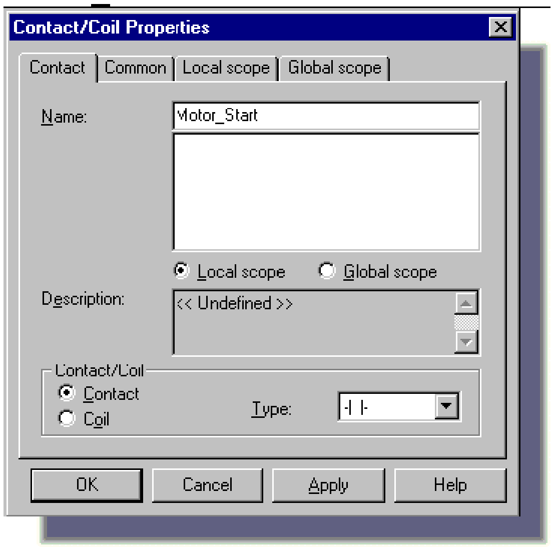
We now want to declare the properties of the LD objects which have been automatically inserted with the first LD network

a. Double click on the contact 'C000'.
As an option you can mark the contact by clicking on it and then press <Enter>.

The 'Contact/Coil Properties' dialog appears.

b. Change the variable name from its default name 'C000' to 'Motor_Start'.

Figure 9:
Dialog 'Contact/Coil Properties' for setting the contact properties



Name: Motor_Start

● Local scope ○ Global scope

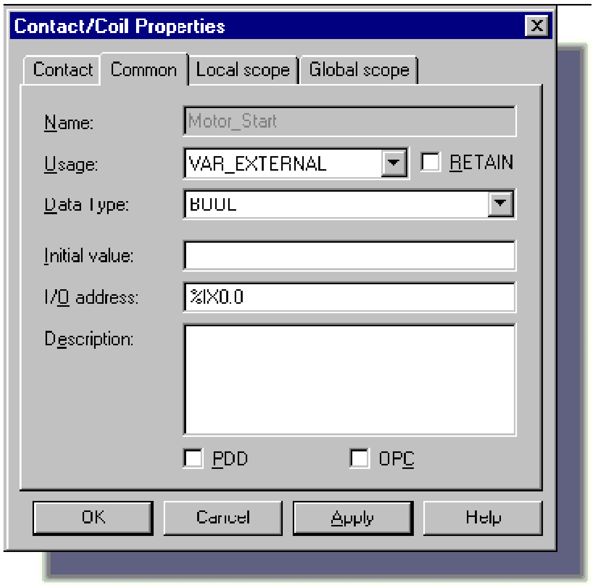
Description: << Undefined >>

Contact/Coil
● Contact
○ Coil
Type: -| |-

c. Click 'Apply' or press <Enter>. The dialog page 'Common' is opened automatically.

d. In the list box 'Usage' (see Figure 10) select 'VAR_EXTERNAL' so that the variable will be declared as global variable, meaning it can be used in each POU of the project.

e. We now want to assign the variable 'Motor_Start' to our I/O simulator, so we can test the logic on the screen. To do so, we have to assign the variable's physical PLC address in the input field 'I/O address'. Since this contact is meant to start the motor, we need to declare an input variable. Enter '%IX0.0' for the declaration of the located variable 'Motor_Start'. This is the simulator position, where '0.0' designates 'first module.first point'.



Name: Motor_Start

Usage: VAR_EXTERNAL □ RETAIN

Data Type: BOOL

Initial value:

I/O address: %IX0.0

Description:

□ PDD □ OPC

Figure 10: Dialog 'Contact/Coil Properties' for setting the contact properties

f. Open the dialog page 'Local scope' and select 'Default' from the tree.
By selecting this entry we define, that the VAR_EXTERNAL declaration of the new variable will be inserted into the variables group 'Default' of the local variables grid worksheet.

As we are declaring a global variable (usage VAR_EXTERNAL) we have to select both, a local and a global scope.

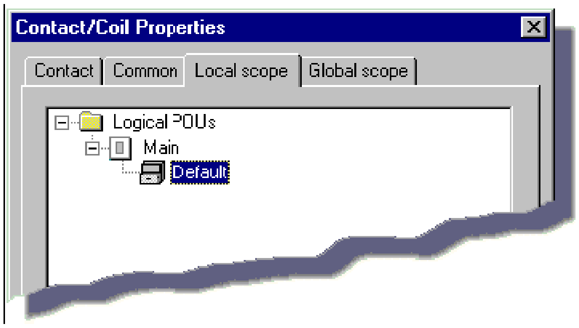


Figure 11:
Dialog 'Contact/Coil Properties' for setting the contact properties

g. Open the dialog page 'Global scope' and select 'Default' from the tree.
By selecting this entry we define, that the VAR_GLOBAL declaration of the new variable will be inserted into the variables group 'Default' of the resource's global variables grid worksheet.

As we are declaring a global variable (usage VAR_EXTERNAL) we have to select both, a local and a global scope.

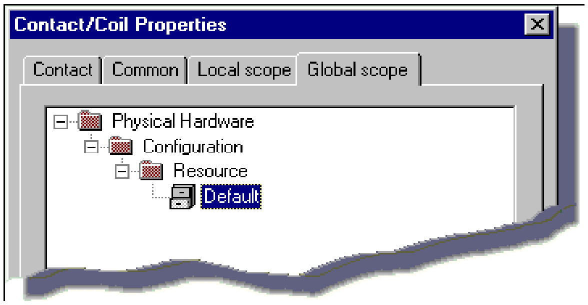


Figure 12:
Dialog 'Contact/Coil Properties' for setting the contact properties

For detailed information on the variable declaration and the function of local and global variables refer to the Appendix, section 'Variables and Data Types'.

h. Click 'OK' to confirm the 'Contact/Coil Properties' dialog.
The variable and its declaration will be inserted.

Now your screen should look as follows:

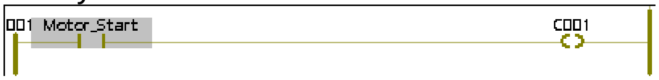


Figure 13:
LD code worksheet with global input variable 'Motor_Start'

STEP 3

INSERTING A COUNTER USING THE EDIT WIZARD

Now we want to insert and connect a counter in the LD code worksheet using the Edit Wizard.

a. As we want to insert the counter between 'Motor_Start' and 'C001', click on the line to mark it:

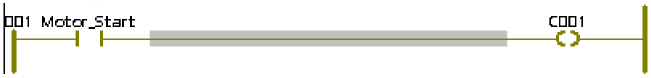
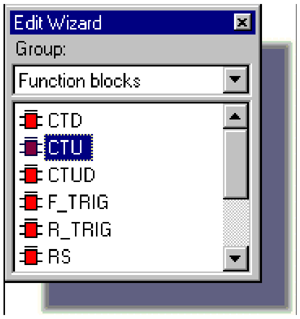


Figure 14:
Line marked to specify the position for inserting the counter

b. If the Edit Wizard is not already opened, click on the 'Edit Wizard' icon in the toolbar:

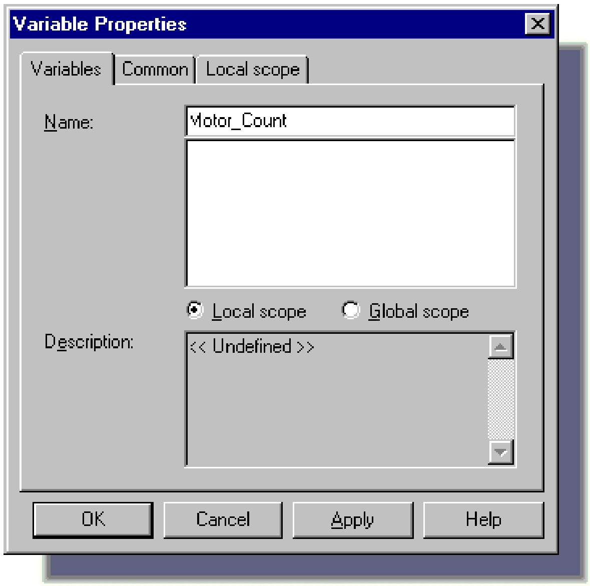
The Edit Wizard appears.

c. Open the group 'Function blocks' (if not already open). In the list of function blocks browse for the entry 'CTU' and double click on it.



The dialog 'Variable Properties' appears.

Figure 15: Dialog 'Variable Properties' for setting the counter properties



Name: Motor_Count

● Local scope ○ Global scope

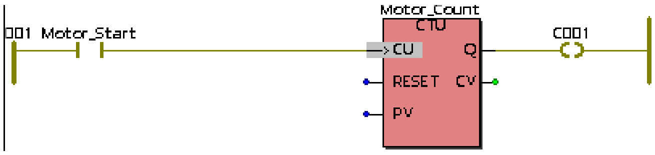
Description: << Undefined >>

d. Change the instance name from its default name to 'Motor_Count' in the 'Name' input field and click 'OK'. The 'Common' dialog page is opened automatically. Confirm the dialog with 'OK'.

e. Hide the Edit Wizard by clicking on the 'Edit Wizard' icon again:

Your worksheet should now look as follows:

Figure 16: LD code worksheet with inserted counter 'CTU'



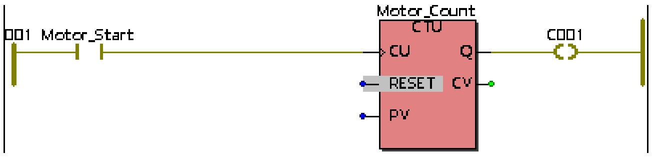
STEP 4

INSERTING THE COUNTER 'RESET' CONTACT

Now we want to define the 'Reset' input of the CTU function block and connect it to the left powerrail.

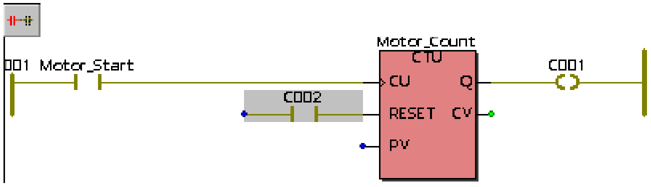
a. Click on the 'Reset' input to mark it:

Figure 17: Marked input of the counter for connecting a contact



b. Click on the 'Add contact left' icon in the toolbar to insert a contact left of the 'Reset' input.

Figure 18: Inserting a contact in the LD network

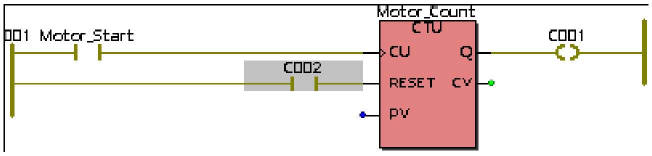


c. Click on the 'Connect Objects' icon in the toolbar:

d. Click on the contact 'C002' for the starting point of the connection line.

e. Move the mouse to the left powerrail and click once to end the line.

Figure 19: Connecting the contact using the connection mode



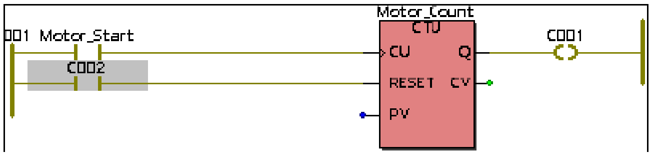
Moving the contact:

f. Switch from connection mode to mark mode by pressing <Esc> or clicking on the 'Mark' icon:

g. Click on the contact 'C002'.
Drag the contact to the left powerrail until it is positioned below 'Motor_Start'.

h. Release the mouse button to drop the contact 'C002':

Figure 20:
Moving the contact



STEP 5

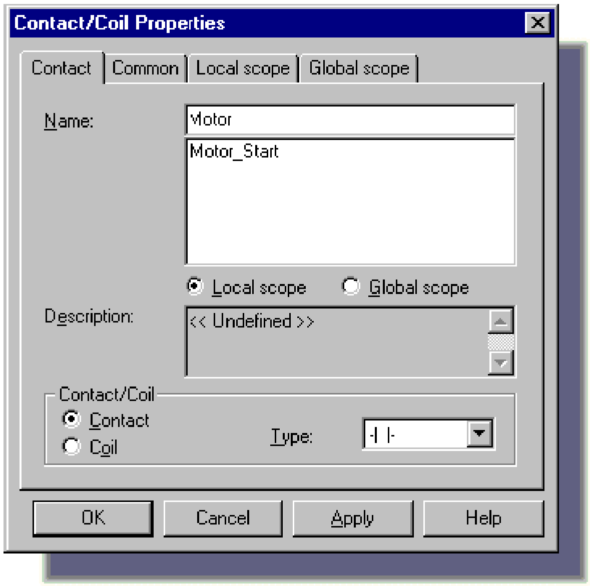
DECLARING THE PROPERTIES OF THE COUNTER'S 'RESET' CONTACT

Next we declare the properties of the reset contact 'C002'.

a. Double click on the contact 'C002' to open the 'Contact/Coil Properties' dialog.

b. Change the name from the default name 'C002' to 'Motor'.

Figure 21:
'Contact/Coil Properties' dialog for setting the contact properties



Name: Motor

Motor_Start

● Local scope ○ Global scope

Description: << Undefined >>

Contact/Coil

● Contact

○ Coil

Type: -| |-

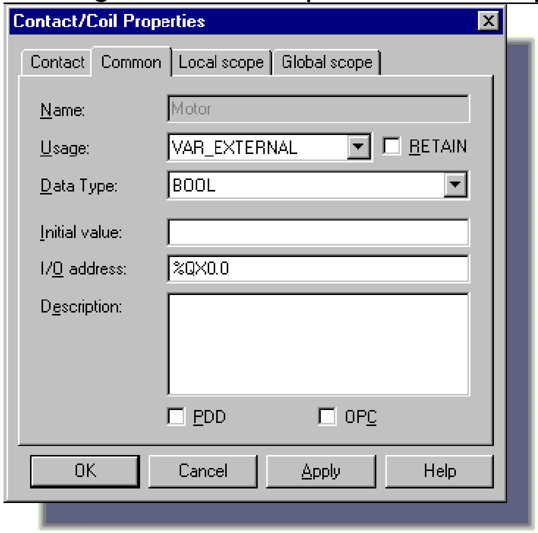
c. Click 'Apply' or press <Enter>. The dialog page 'Common' is opened automatically.

d. In the list box 'Usage' (see Figure 22) select 'VAR_EXTERNAL' so that the variable will be declared as global variable, meaning it can be used in each POU of the project.

e. We now want to assign the variable 'Motor' to our I/O simulator, so we can test the logic on the screen.

The input field 'I/O address:' specifies the physical PLC address for the variable. Enter '%QX0.0' for the declaration of the variable 'Motor', where 'Q' designates 'Output' and '0.0' signifies 'first output module.first point'.

Figure 22:
Dialog 'Contact/Coil Properties' for setting the contact properties



Name: Motor

Usage: VAR_EXTERNAL □ RETAIN

Data Type: BOOL

Initial value:

I/O address: %QX0.0

Description:

□ PDD □ OPC

Since we have already specified the local and global scope (i.e. variable group) for the first variable 'Motor_Count', it is not necessary to select a group for all further variables of the same groups.
For further information about the location and size prefixes refer to the Appendix, section 'Variables and Data Types'.

f. Click 'OK' or press <Enter> to confirm the 'Contact/Coil Properties' dialog. Now your LD worksheet should look as follows:

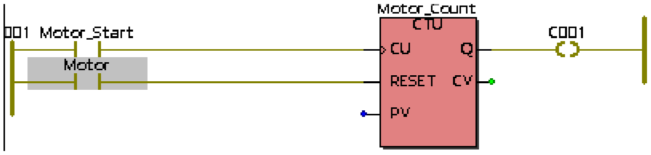


Figure 23:
Newly declared variable 'Motor' in LD code worksheet

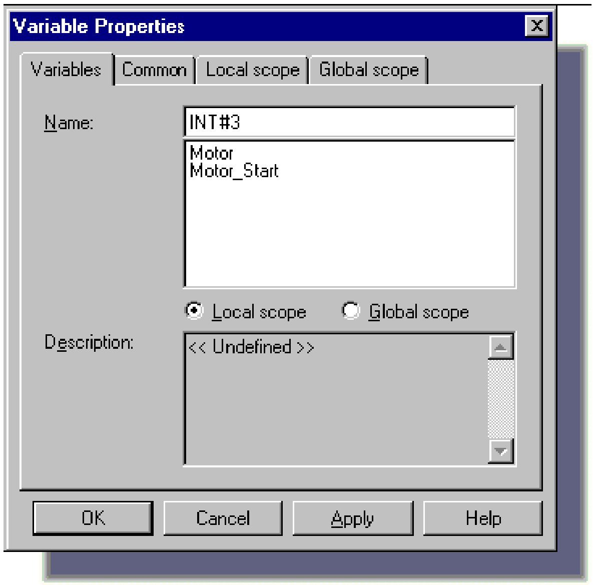
Defining the counter parameters

Defining the counter's preset value:

g. Double click on the blue connection point of the preset value 'PV'.

The 'Variable Properties' dialog appears (see Figure 24).

h. As we want the motor to start after pushing the start button three times, enter 'INT#3' as preset value in the 'Name' input field. 'INT' means 'Integer', '#' designates a constant and '3' is the actual value.



Name: INT#3

Motor
Motor_Start

● Local scope ○ Global scope

Description: << Undefined >>

Figure 24: 'Variable Properties' dialog for defining the type and name of a variable

i. Confirm the dialog with 'OK'. The integer value is directly inserted into the code body.

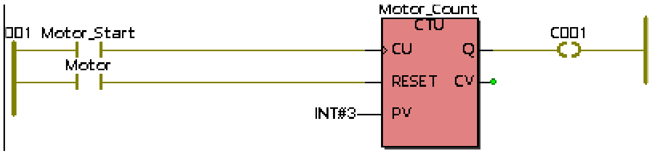


Figure 25: Newly declared constant in LD code worksheet

Configuring the counter's current value 'CV':

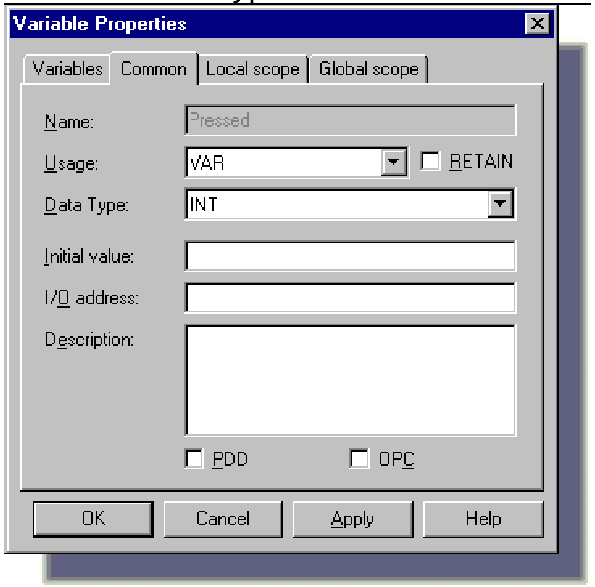
j. Double click on the green connection point of the current value output 'CV'.

The 'Variable Properties' dialog appears.

k. Enter 'Pressed' as variable name. The current value of the counter will now be stored in the variable 'Pressed'.

l. Click 'OK'. The dialog page 'Common' is opened automatically.

The current value is an integer, so select the data type 'INT' from the 'Data Type' list box.



Variable Properties

Variables | Common | Local scope | Global scope

Name: Pressed

Usage: VAR ☐ RETAIN

Data Type: INT

Initial value:

I/O address:

Description:

☐ PDD ☐ OPC

OK | Cancel | Apply | Help

Figure 26: 'Variable Properties' dialog for setting the variable properties

m. Click 'OK'. Your worksheet should look as follows:

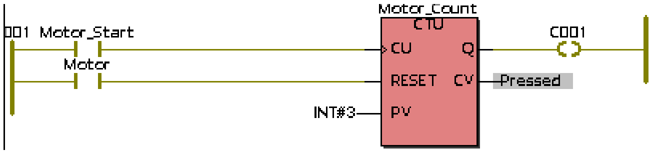
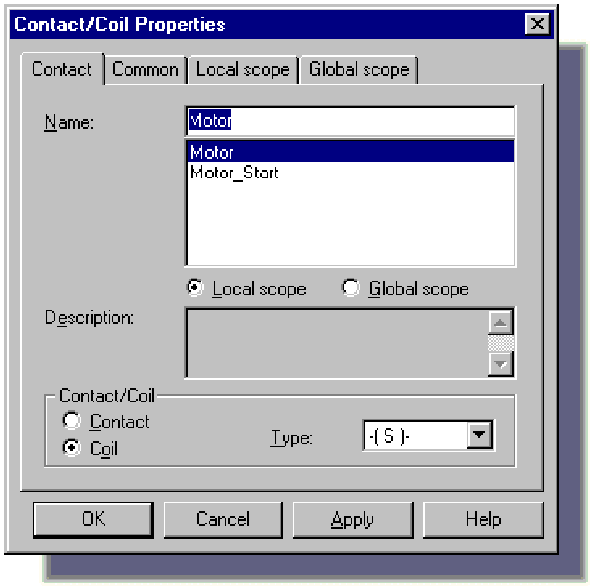


Figure 27: Newly declared variables in LD code worksheet

Configuring the coil 'C001':

n. Double click on 'C001'. The 'Contact/Coil Properties' dialog appears.

o. For this coil we select the existing variable 'Motor'. The list below the 'Name' input field contains all variables which have already been declared (local or global), depending on the activated scope radio button.
From the list select the local variable 'Motor'.



Contact/Coil Properties

Name: Motor

- Motor
- Motor_Start

● Local scope ○ Global scope

Description:

Contact/Coil

○ Contact

● Coil

Type: -(S)-

Figure 28: 'Contac/Coil Properties' dialog for selecting a variable for a coil

p. Since we want the motor to run continuously after we started it, we will use a SET coil.
From the 'Type' list box select '-(S)-' (see Figure 28).

q. Click 'OK'. Your worksheet should look as follows:

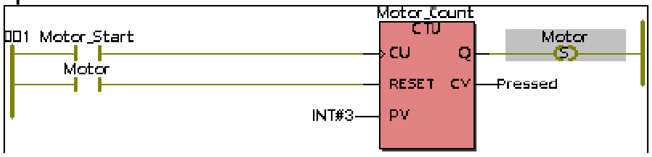


Figure 29: Inserted 'Motor' variable

INSERTING A SECOND LD NETWORK AND EDITING NETWORK DESCRIPTION COMMENTS

STEP 6

Now we want to insert a logic to stop the motor and edit the network description comments.

a. Click with the left mouse button at a suitable distance below the existing LD network, to set an insertion mark at the position shown below.

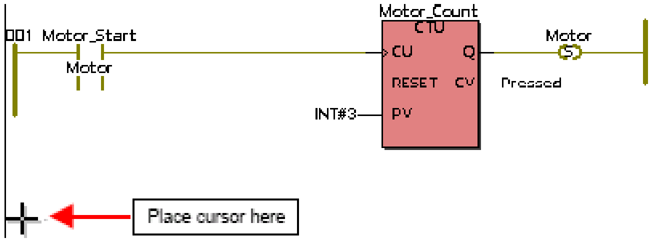


Figure 30: Insertion mark for inserting a second LD network

b. Click on the 'Contact Network' icon to insert a new LD network:

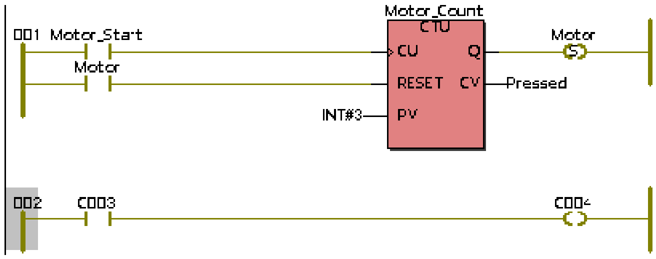
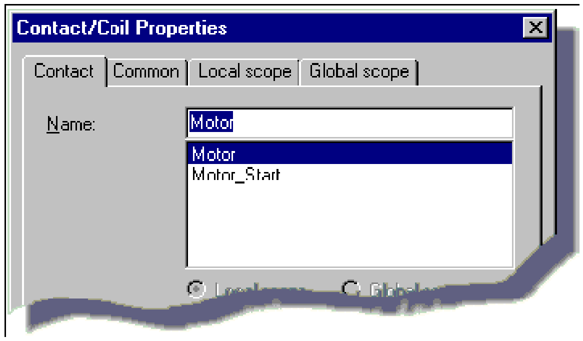


Figure 31: Second LD network inserted in the LD code body worksheet

c. Double click on the contact 'C003' to declare the contact properties.

d. In the 'Contact/Coil Properties' dialog which appears select the local variable 'Motor' from the list:



Contact/Coil Properties

Contact | Common | Local scope | Global scope

Name: Motor

Motor
Motor_Start

● Local ... ○ Global ...

Figure 32: Dialog 'Contac/Coil Properties' for setting the variable type for the contact

e. Click 'OK'. The variable 'Motor' is inserted at contact 'C003'.

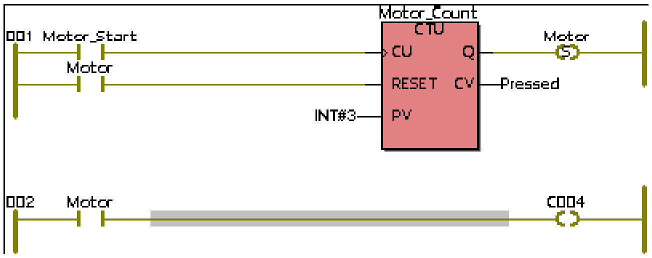
Inserting a timer:

Now we will **insert the timer** that controls how long the motor will run.

f. Open the Edit Wizard by clicking on the 'Edit Wizard' icon in the toolbar.

g. In the second LD network, mark the line between 'Motor' and 'C004' to insert and connect a function block at this position.

Figure 33: Marked line

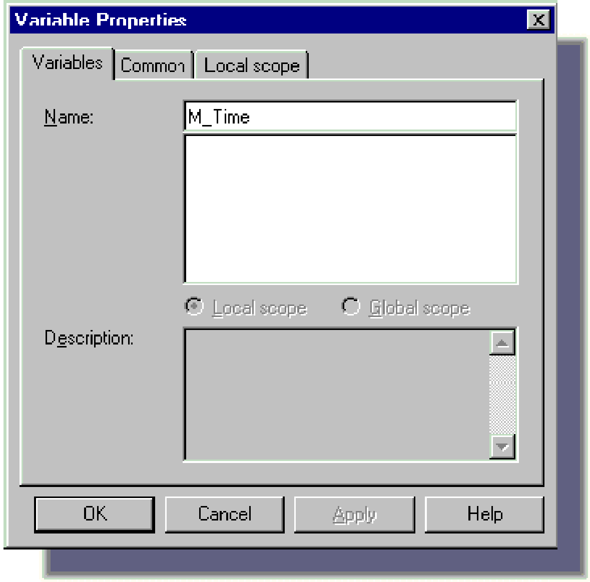


h. From the Edit Wizard's 'Group' list box select the group 'Function blocks'.

i. From the list of function blocks select the timer 'TON' ('Timer On Delay') and insert it by double clicking on it.

j. The dialog 'Variable Properties' appears. Enter 'M_Time' as instance name in the 'Name' input field:

Figure 34: Dialog 'Variable Properties' for declaring the instance name


Variable Properties

Variables | Common | Local scope

Name: M_Time

● Local scope ○ Global scope

Description:

OK | Cancel | Apply | Help

k. Click 'OK'. The 'Common' dialog page is opened automatically. Confirm the dialog with 'OK'.

Since you have marked the line before inserting the object, the function block will be inserted and connected directly at the specified position.

Your screen should look as follows:

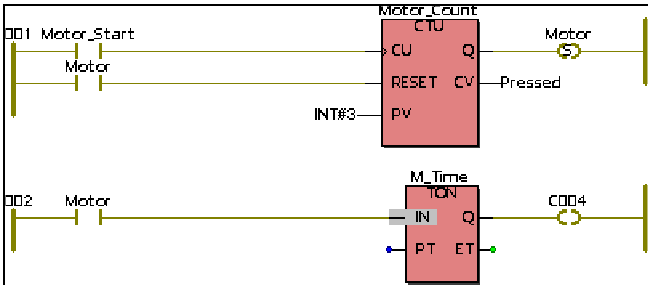


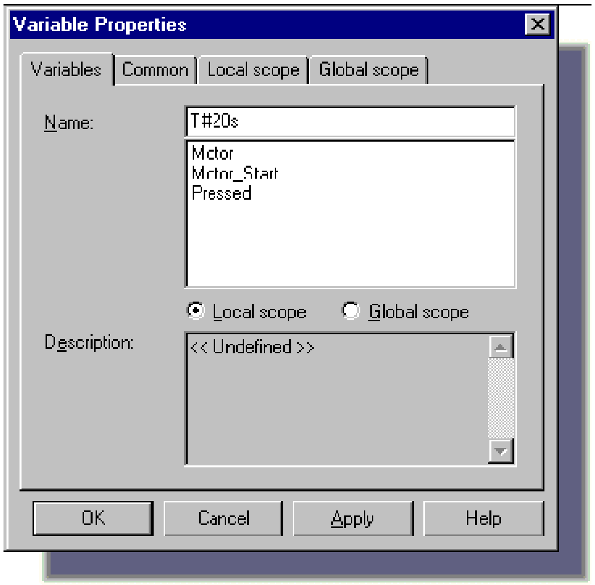
Figure 35: LD code worksheet with second LD network and function block 'TON'

l. Hide the Edit Wizard by clicking on the 'Edit Wizard' icon again:

Now we want to **determine the timer's preset time 'PT',** which will control how long the motor runs:

m. Double click on the blue connection point of input 'PT'.

The 'Variable Properties' dialog appears.



Variable Properties

Variables | Common | Local scope | Global scope

Name: T#20s

Mctor
Mctor_Start
Pressed

● Local scope ○ Global scope

Description: << Undefined >>

Figure 36: 'Variable Properties' dialog for declaring a local variable

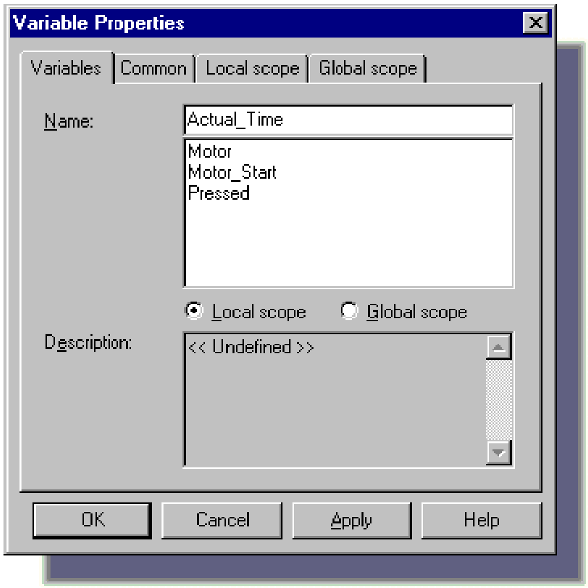
n. Enter 'T#20s' as time constant in the 'Name' input field. Here 'T' designates a time value, '#' signifies 'constant' and '20s' is the actual time value of 20 seconds (because we want the motor to run 20 seconds).

o. Click 'OK'. The constant 'T#20s' is inserted directly at the input PT (see Figure 39).

Now we are going to **define a variable to hold the elapsed time 'ET'**:

p. Double click on the green connection point of output 'ET'. The 'Variable Properties' dialog appears.

q. Enter 'Actual_Time' as name for the local variable.



Name: Actual_Time

Motor
Motor_Start
Pressed

● Local scope ○ Global scope

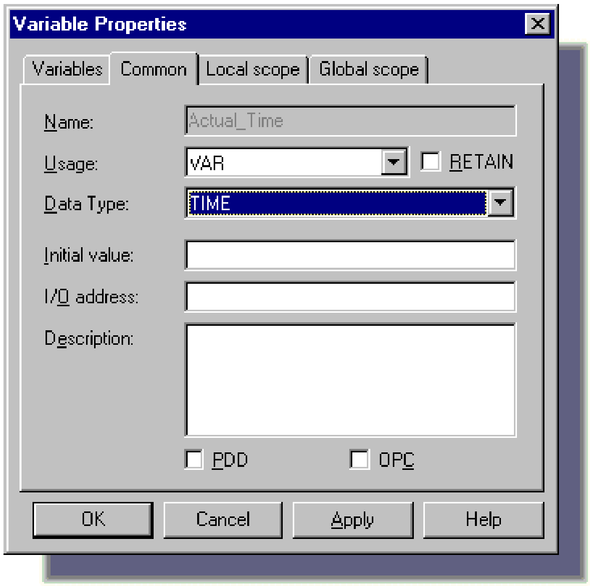
Description: << Undefined >>

Figure 37: 'Variable Properties' dialog for declaring a local variable

r. Click 'OK'. The 'Common' dialog page is opened automatically.

s. The timer output 'ET' needs a variable of the data type 'TIME'. For that reason, select the data type 'TIME' from the 'Data type' list box.

Figure 38: 'Variable Properties' dialog for declaring a local variable



Name: Actual_Time

Usage: VAR ☐ RETAIN

Data Type: TIME

Initial value:

I/O address:

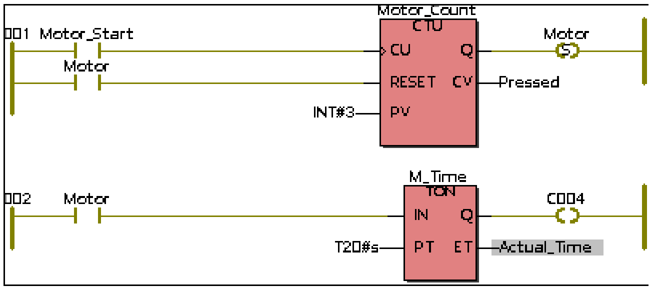
Description:

☐ PDD ☐ OPC

t. Click 'OK' to insert the newly declared variable.

Your worksheet should look as follows:

Figure 39: LD code worksheet with second LD network and function block 'TON'

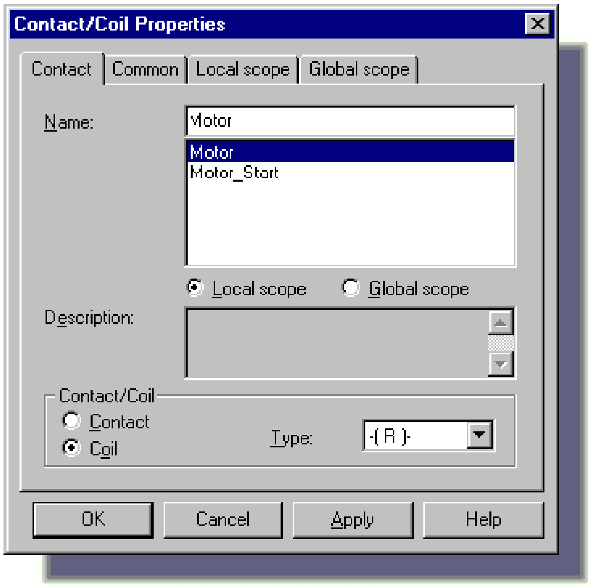


The last variable we have to declare is the one for the coil 'C004'.

u. Double click on the coil 'C004' to open the 'Contact/Coil Properties' dialog. Select the variable 'Motor' from the variables list.

v. When energized, this coil will stop the motor. Since we use a Set coil to start the motor, we need to use a Reset coil to stop it.

So set the coil type to 'RESET' by selecting the corresponding list box entry as shown in the following figure.



Contact/Coil Properties

Contact | Common | Local scope | Global scope

Name: Motor

Motor
Motor_Start

● Local scope ○ Global scope

Description:

Contact/Coil
○ Contact
● Coil

Type: -(R)-

Figure 40: 'Contact/Coil Properties' dialog for setting the coil and variable type

w. Click 'OK'. Your worksheet should look as follows:

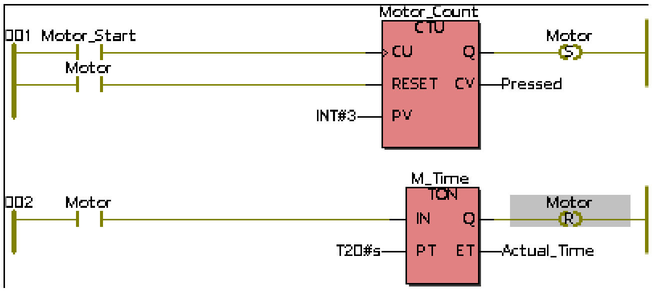


Figure 41: LD code worksheet with inserted second LD network and function block 'TON'

Finally, we will insert the network description comments.

x. Double click on the left powerrail in the LD code worksheet:

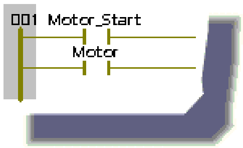


Figure 42: Marked powerrail

The 'Comment' dialog appears:

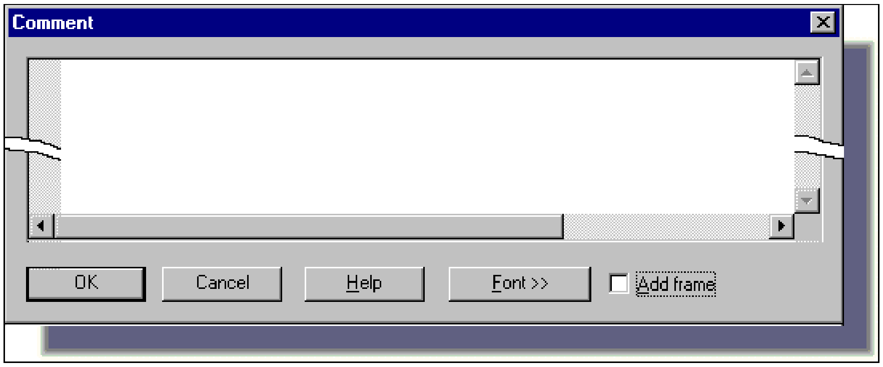
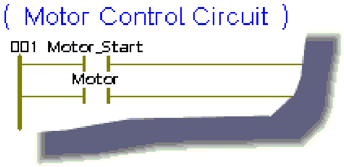


Figure 43: 'Comment' dialog for entering comments in the LD code worksheet

y. In the 'Comment' dialog type 'Motor Control Circuit'.

By clicking on the button 'Font >>', you can change the font properties. Select '**blue**' as color and the font width '**20**'.

z. Click 'OK'.



PHASE 3
COMPILING THE EXAMPLE PROJECT

Now that the editing process is finished we have to compile the project. During compilation the contents of the worksheets are translated and transformed to special code which can be executed by your PLC.

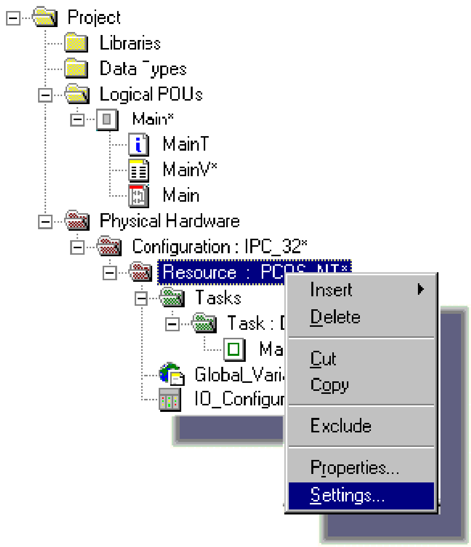
The programming system provides several possibilities for compiling. For detailed information please refer to the online help.

STEP 1

'MAKING' THE PROJECT

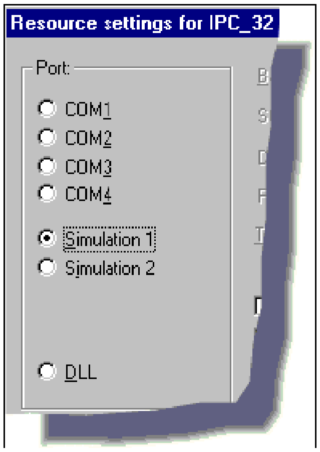
a. In our example we are working with the simulation. Make sure, that the simulation has been activated. To do so, right click in the project tree on the folder 'Resource' and select 'Settings...' from the appearing context menu:

Figure 44: Project tree with context menu for calling the resource settings



The 'Resource settings' dialog appears:

Figure 45: Dialog 'Resource settings' for setting the output device


Resource settings for IPC_32

Port:

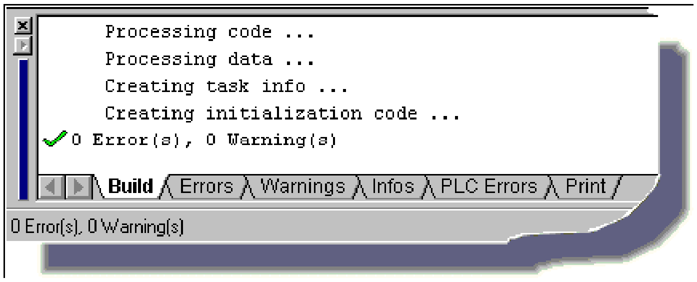
○ COM1
○ COM2
○ COM3
○ COM4
● Simulation 1
○ Simulation 2
○ DLL

b. Activate 'Simulation 1', if necessary, and close the dialog with 'OK'.

c. Click on the 'Make' icon in the toolbar:

The compilation process is displayed in the 'Build' tab of the message window. Errors and warnings detected during compilation are logged in the corresponding sheets of the message window. You can use the message window to access a particular code body worksheet by double clicking on the error message.

Figure 46: Message Window after compiling the project using the 'Make' command



```
Processing code ...
Processing data ...
Creating task info ...
Creating initialization code ...
✓0 Error(s), 0 Warning(s)
```

0 Error(s), 0 Warning(s)

STEP 2

HANDLING ERRORS AND MESSAGES

During the 'Make' process it is possible that you will detect errors or warnings.

Errors will prevent the compile process from being completed and include such issues as syntax errors or structure problems. **Warnings** indicate potential problems like a variable that is not being used. Warnings do not prevent the compilation process from being completed.

You can ignore warnings, but you must fix errors to proceed with the exercise.

□ To display the detected errors, click on the 'Errors' tab in the message window.
A list of the errors will then be displayed in the message window.

□ In order to display the list of warnings, click on the 'Warnings' tab.

□ In most cases double clicking on an error/warning will directly open the worksheet in which the programming error/the reason for the warning has occurred. The corresponding line or object is marked.
You can also mark the error and press <SHIFT> + <F1> to get the corresponding help topic with information on the cause of the error and the further steps necessary to fix it.

□ Fix all errors (if any have occurred) and re-compile the project using the 'Make' icon.

□ Only then you can download the program to the PLC.

PHASE 4

DOWNLOADING THE PROJECT TO THE IO SIMULATION OR KinCon-8000

Now, the compiled project has to be downloaded (i.e. sent) to the I/O Simulation or KinCon-8000.

The communication with the simulation or KinCon-8000 is done using the PLC control dialog, named 'Resource'.

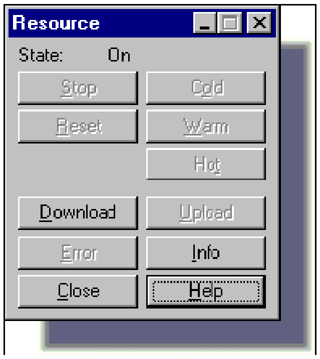
When working with several resources, different dialogs are used to download a project and control the targets.
Please refer to the online help system for detailed information.

Download to Simulation

a. Click on the 'Project Control Dialog' icon to open the resource control dialog.

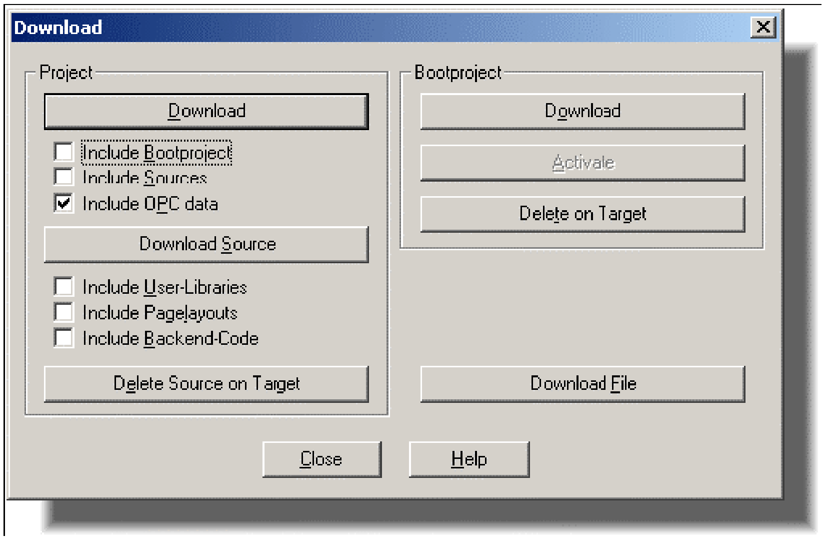
The control dialog appears showing the resource name in its titlebar.

Figure 47:
' Resource' dialog
f or controlling the
PLC or simulation


State: On

b. Press the 'Download' button.
The 'Download' dialog appears:

Figure 48: 'Download' dialog for initiating the project download



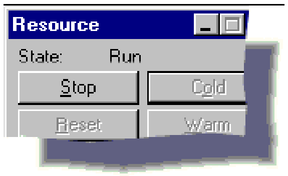
□ Include Bootproject
□ Include Sources
☑ Include OPC data
□ Include User-Libraries
□ Include Pagelayouts
□ Include Backend-Code

The dialog is used to start the download process. You can send either a "normal" project or the zipped project source (which can be used as a backup) to the KinCon-8000 or simulation.

c. Press the 'Download' button in the 'Project' section to download the project:
The successful download process is indicated by a blue status bar at the bottom of the screen.

d. Press the 'Cold' button in the control dialog to execute a cold start:

The state of the resource changes from 'Stop' to 'Run':

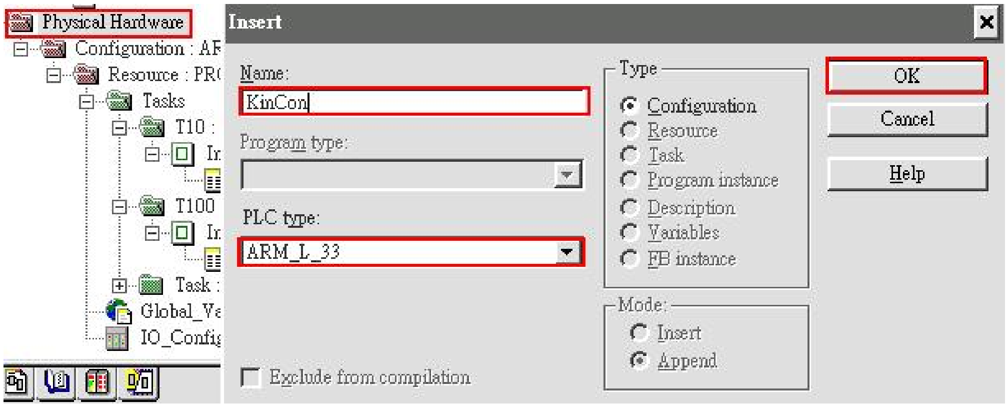


State: Run

Download to KinCon-8000

For download to KinCon-8000, We have to create a new **Configuration** in **MultiProg**. The process is as below:

a. Press right mouse button on Physical Hardware. Insert ‘Configuration’ then enter ‘**Name**’, and choose ’**ARM_L_33**’ as PLC type. Press ’OK’ to finish.



Name: KinCon

Program type:

PLC type: ARM_L_33

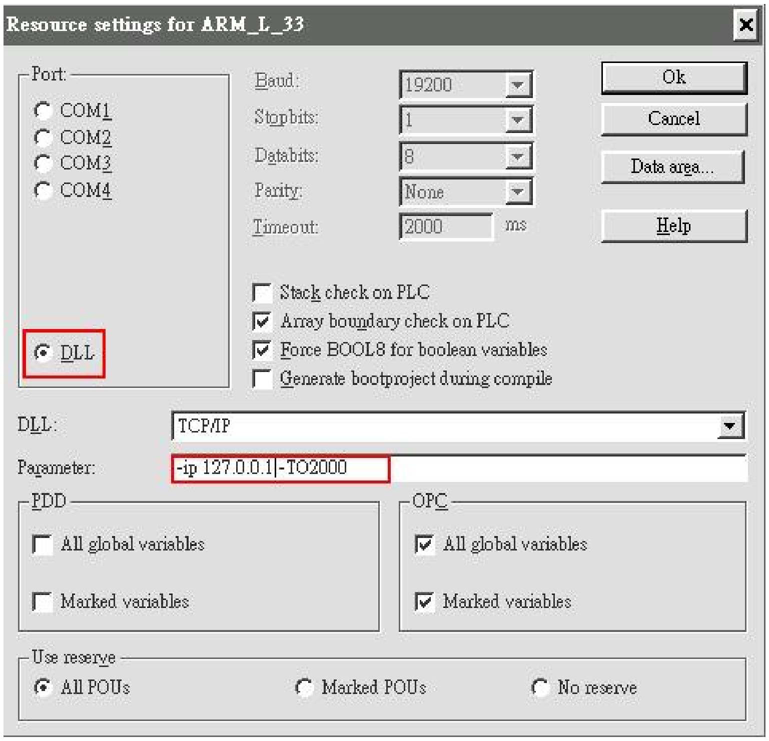
Type
● Configuration
○ Resource
○ Task
○ Program instance
○ Description
○ Variables
○ FB instance

Mode:
○ Insert
● Append

□ Exclude from compilation

b. Copy ‘**Resource:PROCONOS**’ under ‘**Configuration:IPC_32**’ to ‘**KinCon:ARM_L_33**’

c. Adjust Resource setting. Choose ‘**DLL**’ and then set the **KinCon-8000 IP address** in Parameter field.



Port:
○ COM1
○ COM2
○ COM3
○ COM4
● DLL

Baud: 19200
Stopbits: 1
Databits: 8
Parity: None
Timeout: 2000 ms

□ Stack check on PLC
☑ Array boundary check on PLC
☑ Force BOOL8 for boolean variables
□ Generate bootproject during compile

DLL: TCP/IP

Parameter: -ip 127.0.0.1 -TO2000

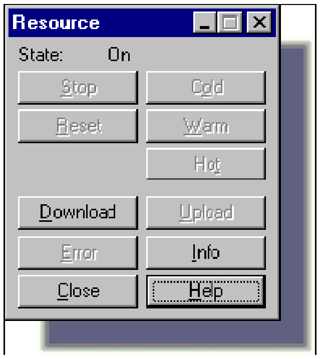
PDD
□ All global variables
□ Marked variables

OPC
☑ All global variables
☑ Marked variables

Use reserve
● All POUs ○ Marked POUs ○ No reserve

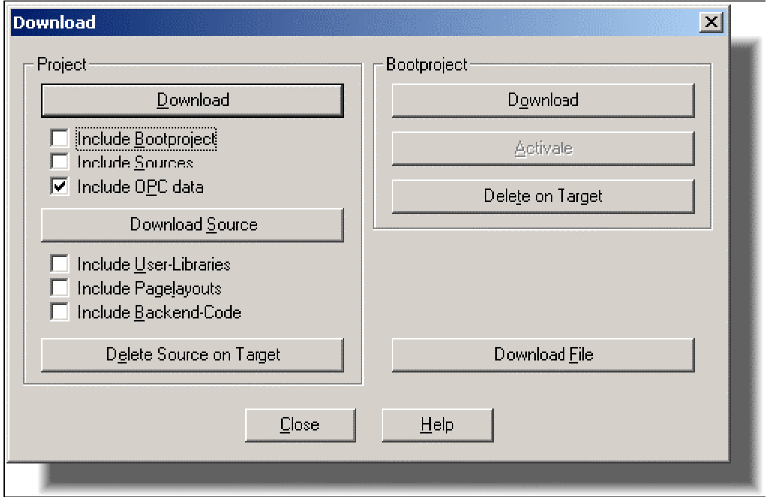
d. Click on the 'Project Control Dialog' icon to open the resource control dialog.

The control dialog appears showing the resource name in its titlebar.



State: On

e. Press the 'Download' button. The 'Download' dialog appears:



☐ Include Bootproject
☐ Include Sources
☑ Include OPC data

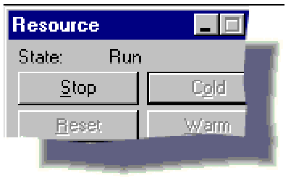
☐ Include User-Libraries
☐ Include Pagelayouts
☐ Include Backend-Code

The dialog is used to start the download process. You can send either a "normal" project or the zipped project source (which can be used as a backup) to the KinCon-8000 or simulation.

f. Press the 'Download' button in the 'Project' section to download the project:
The successful download process is indicated by a blue status bar at the bottom of the screen.

g. Press the 'Cold' button in the control dialog to execute a cold start:

The state of the resource changes from 'Stop' to 'Run':



State: Run

PHASE 5
DEBUGGING THE PROJECT

In the following, the programming system debug tools will be explained. The system supports several debug tools providing a fast and easy way to bring your application online. Although these functions are described on ‘**Simulation’,** it is the same on KinCon-8000.

STEP 1

DEBUG MODE

Worksheets can be switched from edit mode (offline) to debug mode (online) and vice versa. The Online mode is used to detect programming errors and to make sure that the PLC program is running correctly. In Online mode the current values and states of the variables are displayed.

a. Make sure that the KinCon-8000/Simulation is running. The PLC state is shown at the top of the 'Resource' control dialog. If the program is not running, perform a cold start by pressing the 'Cold' button in the control dialog.

b. To activate the debug mode, make sure that the code body of our POU 'Main' is open, and click on the 'Debug on/off' icon in the toolbar:

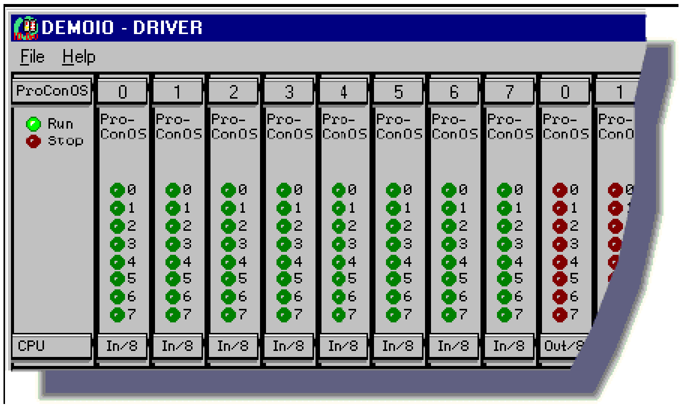
Note that the states and current values of the variables are displayed in several colors indicating the different states:

□ blue = false
□ red = true

You can toggle between the online and the offline mode by clicking on the 'Debug on/off' icon.

c. Click on the button 'DEMOIO - DRIVER' in the Windows taskbar to open the I/O simulator:

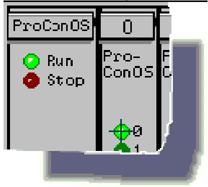
Figure 49:
I/O simulator



d. If necessary, place the I/O Simulator (by drag and drop) to a corner of the screen so that the worksheet is not hidden.

e. Turn bit 0 of module 0 on and off three times by clicking on the first green "virtual LED" of the first input module:

Figure 50:
Toggle bit to start sample program



Watch the reaction in the worksheet:

□ The motor starts running after marking 'Motor_Start' three times, because the current value 'CV' reaches the preset value 'PV' (note the update on the screen).

□ When the 'Motor' Set coil is switched on, it also starts the timer 'M_Time' in rung 002.

□ 'M_Time' (the actual time) runs for 20 seconds until 'ET' (elapsed time) reaches the preset time value 'PT'. The 'Motor' Reset coil in rung 002 is switched on, thus unlatching the 'Motor' Set coil in rung 001 and turning the motor off.

STEP 2

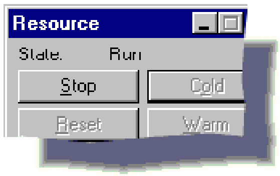
ONLINE EDITING

Online editing is possible without stopping the program execution on the PLC / simulation. This operation is called 'Patch POU'. When you use 'Patch POU', the changes you made in the code are compiled, the related code is generated and then downloaded automatically to the PLC. During the whole patch process the code execution on the PLC is not interrupted.

As a Patch POU example we want to insert an Emergency Stop for the motor: Activating the input 'Emergency_Stop' will stop the motor immediately.

a. Switch to Offline Mode by clicking on the 'Debug on/off' icon:

Our code body worksheet appears in editing mode again. The resource, however, just like a real controller, is still running:



State: Run

b. Set the insertion mark in the LD code worksheet below rung 002.

c. Click on the 'Network' icon to insert a new LD network:

Your worksheet should look as follows:

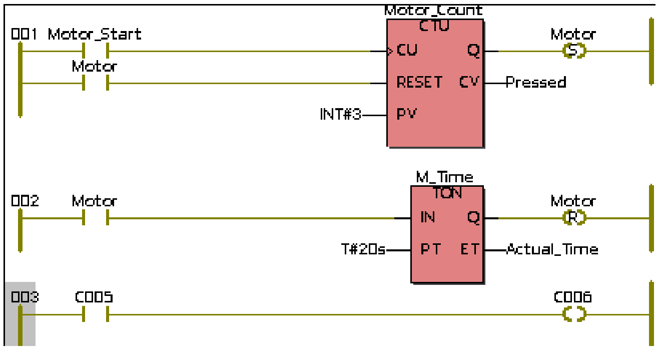
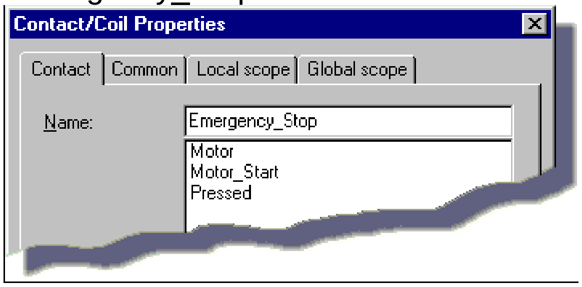


Figure 51: Inserting an LD network

d. Double click on the contact 'C005' to open the 'Contact/Coil Properties' dialog.

e. Change the name from the default name 'C005' to 'Emergency_Stop'.



Name: Emergency_Stop

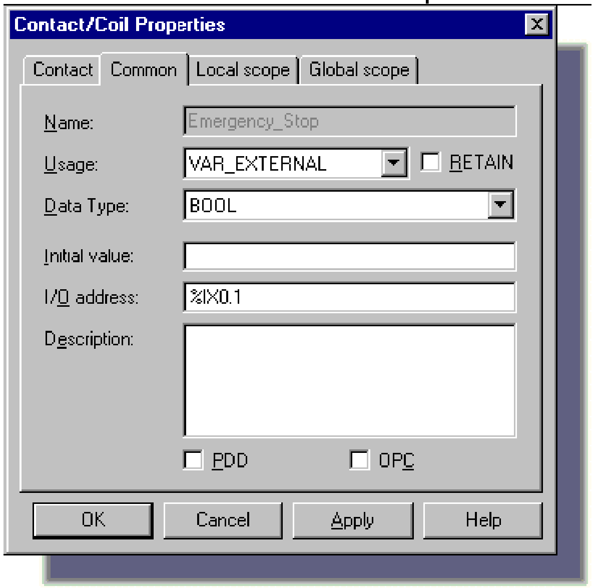
Motor
Motor_Start
Pressed

Figure 52: Dialog 'Contact/Coil Properties' for setting the contact properties

f. Click 'OK'. The 'Common' dialog page is opened automatically.

g. From the list box 'Usage', select 'VAR_EXTERNAL' to declare 'Emergency_Stop' as a global variable.

h. We will use the second input point in the I/O simulator for the emergency stop. Enter '%IX0.1' as the address for this variable in the 'I/O address' input field and click 'OK'.



Name: Emergency_Stop

Usage: VAR_EXTERNAL ☐ RETAIN

Data Type: BOOL

Initial value:

I/O address: %IX0.1

Description:

☐ PDD ☐ OPC

Figure 53: 'Contact/Coil Properties' dialog for setting the contact properties

For further information on the location and size prefixes refer to the Appendix, section 'Variables and Data Types'.

i. Double click on the contact 'C006'. The 'Contact/Coil Properties' dialog appears. Select '-(R)-' from the 'Type' list box and 'Motor' from the global 'Variable' list box. Then click 'OK'.

Your worksheet should look as follows:

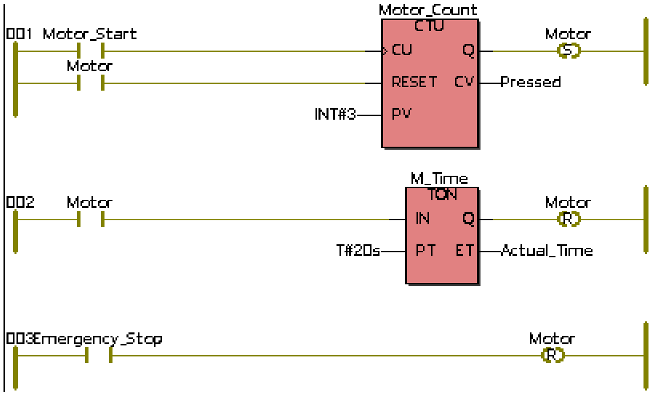


Figure 54: Online editing, changed variable properties

Now that we have changed the code, we make a Patch POU. This process will compile the changes and download them to the PLC without stopping the PLC.

j. Click on the 'Patch POU' icon in the toolbar to compile the modified code and download it to the I/O Simulator:

After the patch process has been successfully completed, the worksheet will be automatically set to Online mode.

k. Click on the 'DEMOIO - DRIVER' button in the Windows taskbar to open the I/O simulator.

l. Turn bit 0 of module 0 on and off three times by clicking on the corresponding input point (LED) (refer to Figure 50 on page 38).

m. Use the new Emergency_Stop contact to immediately stop the motor by clicking on bit 1 of the input module 0 in the I/O simulator.

STEP 3

CROSS REFERENCE WINDOW

The cross reference list contains all variables, function blocks, jumps, labels and connectors which are used within the current project. This tool is particularly helpful for debugging and error isolation.

a. Click on the 'Cross Reference Window' icon in the toolbar to open the Cross Reference Window (if not already opened):

b. Place the cursor in the Cross Reference Window and right click on the window background to open the context menu:

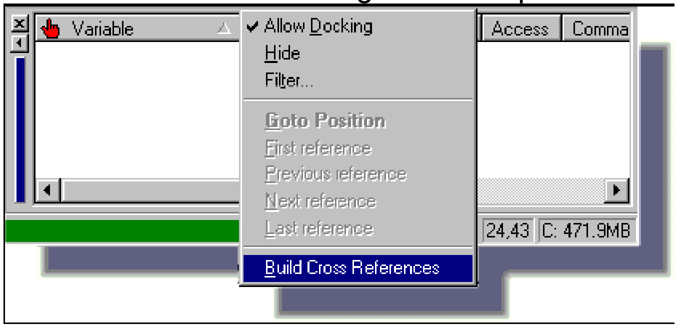
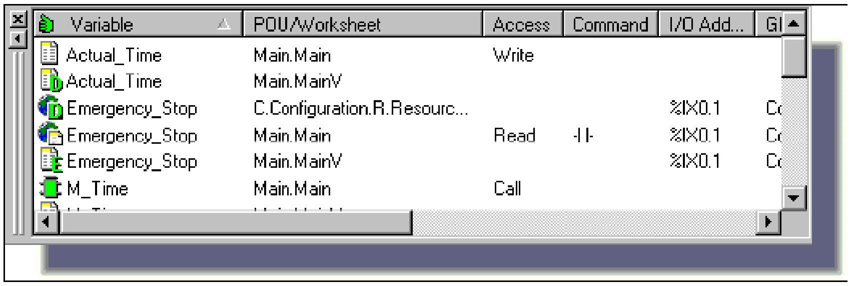


Figure 55: Cross Reference Window with context menu for building the cross references

c. Select the menu item 'Build Cross References'.

Figure 56: Cross reference list in the sample project

The cross reference list will be created.



Variable	POU/Worksheet	Access	Command	I/O Add...	Gl
Actual_Time	Main.Main	Write			
Actual_Time	Main.MainV				
Emergency_Stop	C.Configuration.R.Resourc...			%IX0.1	C
Emergency_Stop	Main.Main	Read	-| |-	%IX0.1	C
Emergency_Stop	Main.MainV			%IX0.1	C
M_Time	Main.Main	Call			

d. Double clicking on a variable in the Cross Reference Window will open the worksheet in which this variable is used and highlight the variable.
In addition if you mark a variable in the worksheet, the corresponding variable in the Cross Reference Window will be marked as well.

e. Close the Cross Reference Window and the Message Window by clicking on the corresponding toolbar icons.
And

STEP 4

VARIABLES WATCH WINDOW

The Variables Watch Window is a powerful tool allowing you to insert different variables easily into a list and observe their runtime behavior. Once a variable is added to the Watch Window, the corresponding worksheet does not have to be open to monitor its current value. As a result, you can focus on those variables you want to see for easier access.

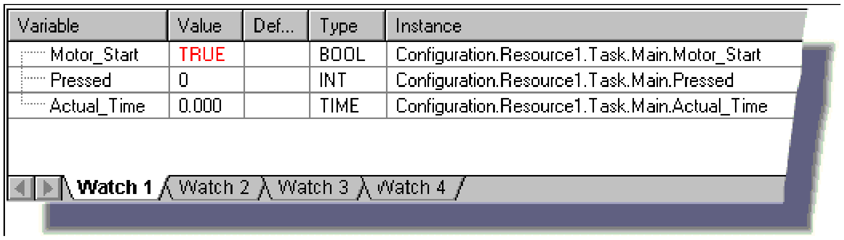
a. If this is not yet the case, switch the worksheet into Online Mode by pressing the 'Debug on/off' icon.

b. Right click in the worksheet and select 'Open Watch Window...' from the context menu or click on the 'Watch Window' button.
The Watch Window appears.

c. In the online worksheet right click on 'Motor_Start' to open the context menu and select 'Add to Watch Window' to insert this variable into the list.

d. Repeat this procedure for the variables 'Pressed' and 'Actual_Time'. The three variables will appear in the list as shown in the figure below.

Figure 57: Variable Watch Window



Variable	Value	Def...	Type	Instance
Motor_Start	TRUE		BOOL	Configuration.Resource1.Task.Main.Motor_Start
Pressed	0		INT	Configuration.Resource1.Task.Main.Pressed
Actual_Time	0.000		TIME	Configuration.Resource1.Task.Main.Actual_Time

You can now use the I/O Simulator to manipulate the contacts and observe the changes of the values, both in the logic and in the Watch Window simultaneously.

STEP 5

FORCING AND OVERWRITING

In Online mode variables can be forced or overwritten. In both cases a new value is assigned to the corresponding variable.

Force: A value is assigned to a variable (usually a contact or coil). The value remains until the force is reset.

Overwrite: A value is temporarily assigned to a variable by the user. The value remains until the program overwrites this value again with the original value in the next program cycle.

The steps necessary for forcing and overwriting a variable are nearly the same.

Be very careful forcing or overwriting variables while your PLC is running. Forcing and overwriting variables mean that the PLC program is executed with the values of the forced or overwritten variable.

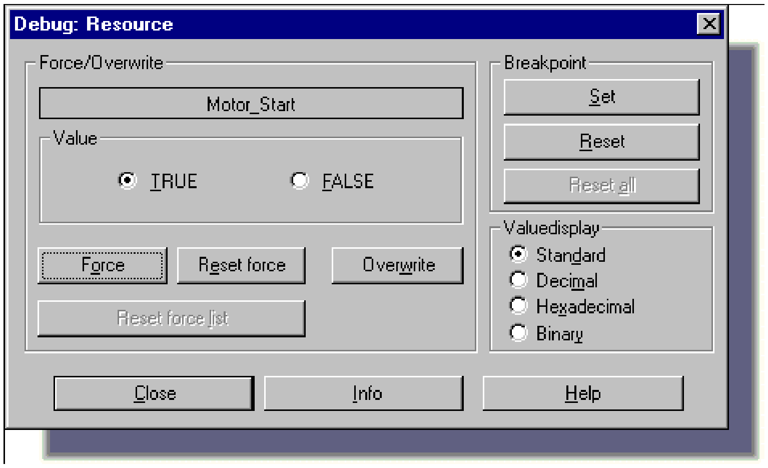
In our example we want to **force the variable 'Motor_Start'**:

a. Make sure, that the worksheet is in Online Mode. If not, press the 'Debug on/off' icon in the toolbar:

b. Click on the button 'DEMOIO - DRIVER' in the Windows taskbar to open the I/O simulator.

c. Make sure, that all inputs are set to 'FALSE' by clicking on each illuminated LED (no input LED should be highlighted).

d. Double click on 'Motor_Start'. The 'Debug: Resource' dialog appears:



Motor_Start

● TRUE ○ FALSE

● Standard
○ Decimal
○ Hexadecimal
○ Binary

Figure 58: 'Debug: Resource' dialog for forcing and overwriting variables

e. Select the radio button 'TRUE', then click on 'Force'. As a result 'Motor_Start' will be forced 'on' and highlighted red in the online worksheet.

f. Double click 'Motor_Start' again and select 'Reset force' to deactivate the force.

If you repeat the steps e. and f. the logic will start executing.

g. In the 'Debug' dialog click on 'Reset force list.
Now we want to overwrite the variable 'Motor'.

h. Double click on the 'Motor' coil in rung 001, then click on 'Overwrite'. This starts 'M_Time'. After 20 seconds, the 'Motor' Reset coil in rung 002 will turn off 'Motor'.

STEP 6

BREAKPOINTS

Breakpoints can be set online in all worksheets using the controls in the right area of the 'Debug: Resource' dialog shown in Figure 58.

When a breakpoint is set, program execution halts at that breakpoint until the developer makes it continue. The programming system provides the possibility to execute a program until the next breakpoint is reached (Single Step) or until the same breakpoint is reached again (Single Cycle).

If a breakpoint is reached, the PLC state changes to HALT [DEBUG] and the control dialog shows the buttons 'Go', 'Step' and 'Trace' to continue with.
Go: Clicking 'Go' causes the program to execute until the next breakpoint is met.

Step: Clicking 'STEP' causes the program to execute the next instruction.

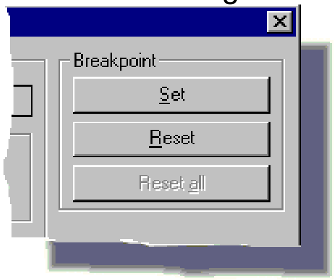
Trace: If an user defined function or a function block call is reached, the function or function block code body is opened and debugged step by step.

Be very careful using breakpoints while the PLC is running, since the breakpoint actually halts program execution. The behavior of the I/Os when reaching a breakpoint depends on the PLC type.

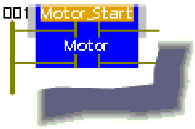
a. To observe breakpoint operation, the corresponding worksheet must be in Online mode ('Debug on/off' icon pressed):

b. Click on the 'PLC Control' icon to open the resource control dialog:

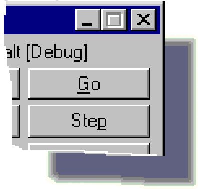
c. Double click on 'Motor_Start' and select 'Set' in the 'Debug: Resource' dialog to set a breakpoint at this variable.



'Motor_Start' is highlighted orange in the online worksheet to indicate the point at which program execution is stopped.



d. Press the 'Go' button in the 'Resource' dialog to activate program execution until the next breakpoint is met.



As we have set only one breakpoint, the program stops again on 'Motor_Start'. This is called a Single Cycle.

e. Click on 'Step' several times and notice that the orange highlight moves to the next instruction each time to indicate the point at which the program execution has stopped. This is a Single Step execution. Also notice that 'Motor_Start' has a red highlight to indicate where the breakpoint has been set.

f. Double click on 'Motor_Start' and press 'Reset' in the debug dialog to reset the breakpoint. Then click 'Go' in the 'Resource' dialog to resume program execution.

g. Click on the 'Debug on/off' icon again to switch to Offline mode:

h. In the Control dialog, click on the 'Stop' button to stop the PLC, and close the 'Resource' control dialog using the 'Close' button.
Stop and Close

PHASE 6
PRINTING THE PROJECT DOCUMENTATION

For documentation purposes it is useful to print the whole project. The programming system offers several possibilities to print your project documentation. The 'File' menu contains the commands for a preview of the current page, for defining the printer settings, and for printing the entire project or single worksheets.

STEP 1

SELECTING A PRINTER

Printer options are located under 'Print Setup...' in the 'File' menu. Selecting this item will open the standard Windows dialog 'Print Setup'.

STEP 2

SETTING THE PAGELAYOUT

A pagelayout defines the appearance of printed worksheets such as the page size, page margins (borders), the source area, foot and head lines containing information such as a company's logo, date, project name or page numbers.

The programming system allows the usage of different pagelayouts. In addition the pagelayouts can be changed. When printing your project or parts of it the default pagelayout is used automatically.

To change the default pagelayout:

a. Select menu 'Extras | Options...'.

b. Open the tab 'Pagelayouts'.

c. Select the desired pagelayouts. In our example we use the default pagelayout.

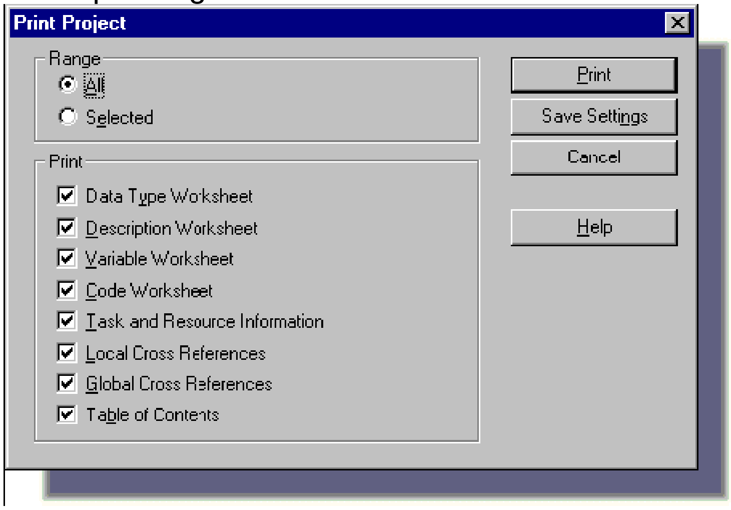
A pagelayout is created and edited using the pagelayout editor. For further information refer to your online help system.

STEP 3

PRINTING THE PROJECT

a. Choose the menu item 'Print Project...' from the 'File' menu. The 'Print Project' dialog appears.

b. In the 'Print Project' dialog deselect the parts of the project you do not want to be printed by deactivating the corresponding checkboxes.



Range
● All
○ Selected

Print
☑ Data Type Worksheet
☑ Description Worksheet
☑ Variable Worksheet
☑ Code Worksheet
☑ Task and Resource Information
☑ Local Cross References
☑ Global Cross References
☑ Table of Contents

Figure 59:
'Print Project' dialog

c. Click on the 'Print' button.

STEP 4

PRINT PREVIEW

The print preview allows you to have a look at how the worksheet would look like when being printed and to modify it if required. It helps you organize the elements on the page in a clear and structured way.

Cross references are not displayed in the preview.

How to call the preview:

a. Make sure that the worksheet you want to see is the active window.

b. Choose the menu item 'Print Preview...' from the 'File' menu. The print preview of the active worksheet will be displayed.

c. To print the single worksheet which is displayed, click on the 'Print' button.

STEP 5

PRINTING A SINGLE WORKSHEET

You can print single worksheets that are opened in the graphic editor or text editor.

Cross references are not printed using the menu item 'Print'.

How to proceed:

a. Make sure that the worksheet you want to print is active.

b. Select the menu item 'Print' from the 'File' menu.
The worksheet will be printed.

USING THE I/O CONFIGURATION

The 'I/O Configuration' dialog is used to edit the I/O configuration worksheet. The I/O configuration normally contains declarations of the I/O modules, such as the logical addresses of a module (start and end address), device declarations (driver name or memory address), etc.

Using Simulation

The following steps explain how to use the I/O Configuration. In our example we will change the number of input modules in the existing I/O group to 10.

a. To change the I/O Configuration, double click on the 'IO_Configuration' in the subtree 'Physical Hardware':

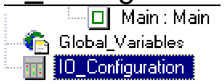
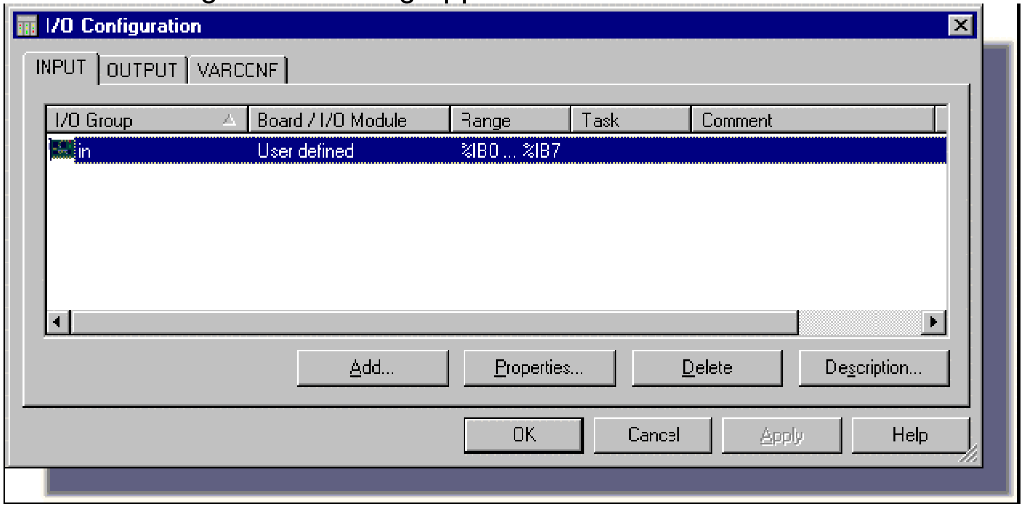


Figure 60:
Icon 'IO_Configuration
in the subtree 'Physical
Hardware'

The 'I/O Configuration' dialog appears:



I/O Group	Board / I/O Module	Range	Task	Comment
in	User defined	%IB0 ... %IB7		

Figure 61:
'I/O Configuration'
dialog

At this point you need to select the driver you will use from the list and configure it as described in the manual for the corresponding driver.
The current I/O configuration is shown in the dialog. We want to change this configuration, i.e. we want to define 10 input modules in the existing group.

b. In the 'I/O Configuration' dialog click on the 'Properties' button:

The 'Properties' dialog appears.

c. In the field 'Length' enter 10 and press the <TAB> key to update the entry in the field 'End address':

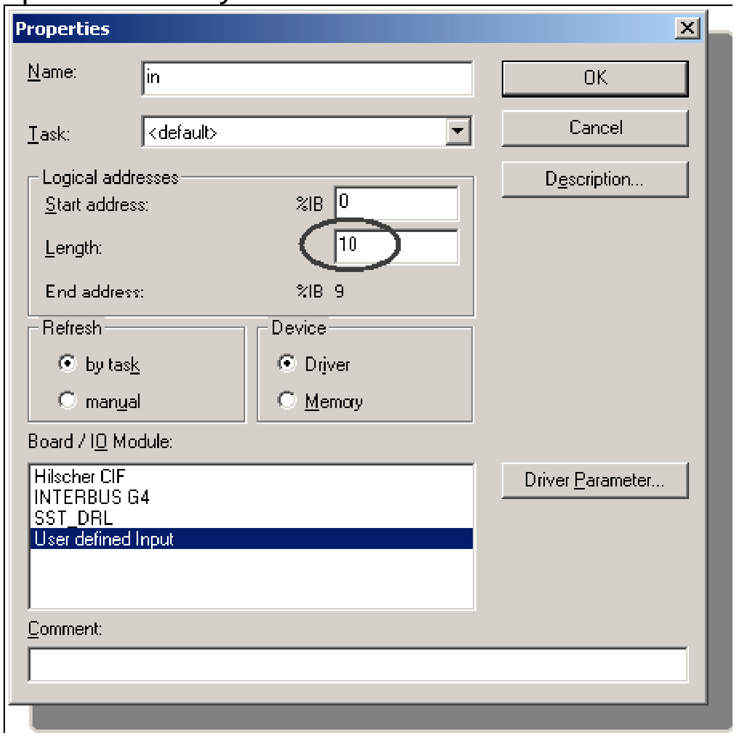


Figure 62: 'Properties' dialog for configuring the I/O Simulator

Properties

Name: in

Task: <default>

Logical addresses

Start address: %IB 0

Length: 10

End address: %IB 9

Refresh

● by task

○ manual

Device

● Driver

○ Memory

Board / IO Module:

Hilscher CIF
INTERBUS G4
SST_DRL
User defined Input

Comment:

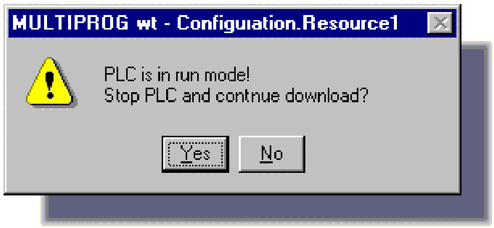
d. Confirm the 'Properties' and I/O Configuration dialogs with 'OK' to return to programming.

e. Compile the project by clicking on the 'Make' icon in the toolbar.

For detailed information on the compilation refer to Phase 3 on page 32 of this Quick Start Manual.

f. Download the project to the target as described in Phase 4 on page 35.

If the PLC/simulation is still running at this moment, the following message dialog appears before the download:

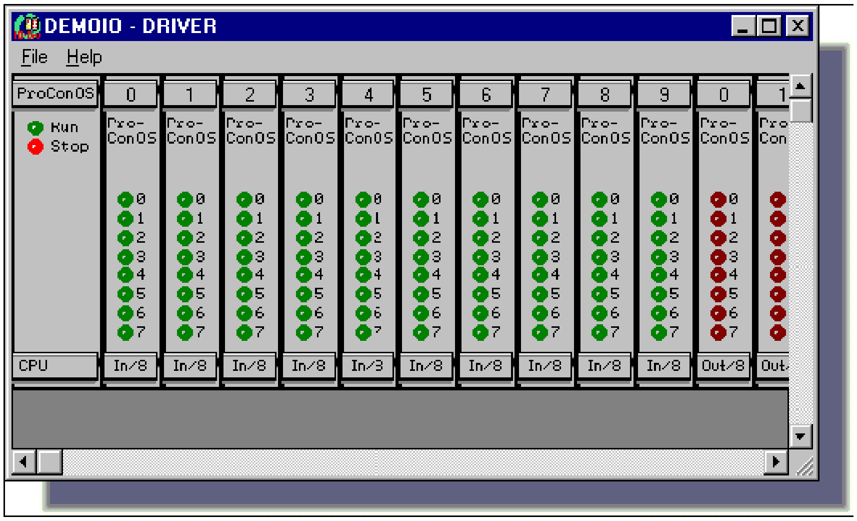


MULTIPROG wt - Configuation.Resource1

PLC is in run mode!
Stop PLC and contnue download?

If this is the case, click on 'Yes' to continue the download.

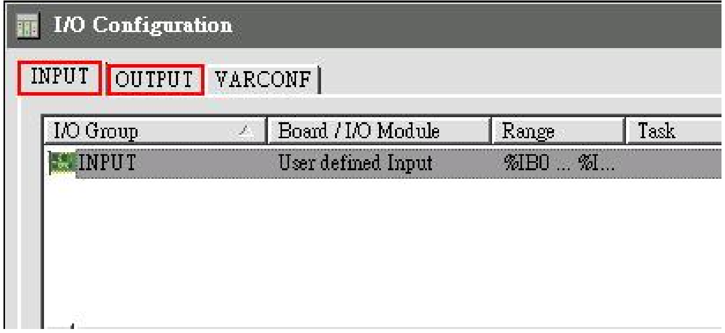
g. Click on the button 'DEMOIO - DRIVER' in the Windows taskbar to open the I/O Simulator. There should be 10 input modules available now:



Using KinCon-8000

The following describe how to configure '**I/O Configuration**' for using ICP DAS I-8K/I-87K/I-7K modules in KinCon-8000.

The user communicates with the channels of the KinCon-8000 modules by defining "I/O Groups" in the "IO_Configuration" part of MULTIPROG project. These I/O Groups need (at least)



I/O Group	Board / I/O Module	Range	Task
INPUT	User defined Input	%IB0 ... %I...	

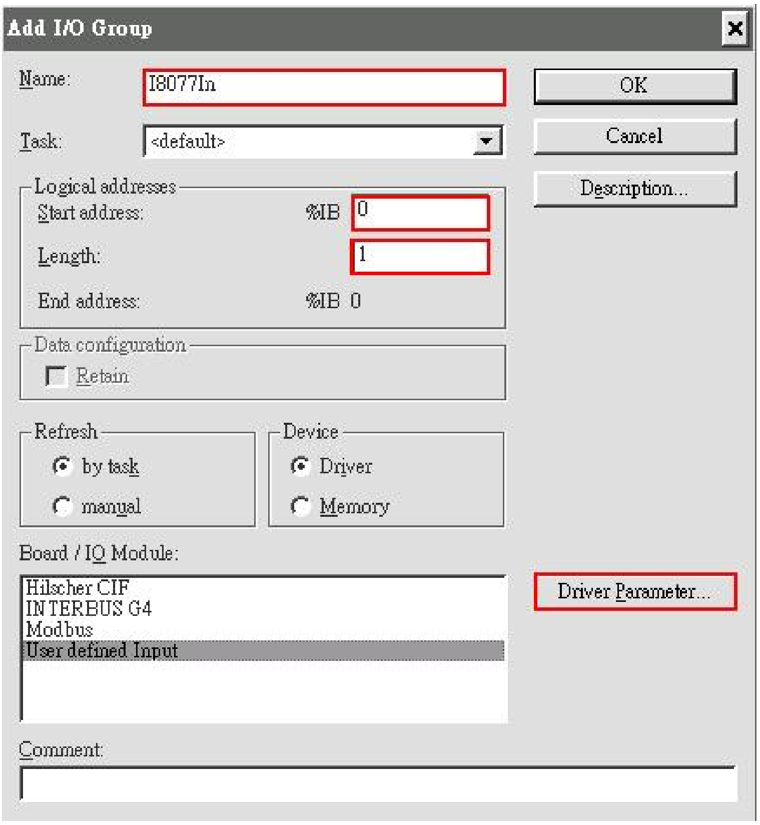
1) a starting address and the address width (the allowed/necessary widths depend on the specific module)
2) driver parameters specifying the KinCon-8000 module and its parameters.

These driver parameters are set through the "Driver information of standard device" dialog (select the "User defined input" of the "Board / IO Module and click the "Driver parameter…" button).

For example:

Add I-8077 in Slot 1

Step 1: Enter **'I8077In**' in '**INPUT I/O Group**'. The name can be any word. Start address will prompt appropriate address. The I-8077 is 8 DIs - 8 DOs module, so the ‘Length’ is ‘1 Byte(8 bits)’. And then press ‘**Driver Parameter’** button.



Name: I8077In

Task: <default>

Logical addresses

Start address: %IB 0

Length: 1

End address: %IB 0

Data configuration

☐ Retain

Refresh

● by task

○ manual

Device

● Driver

○ Memory

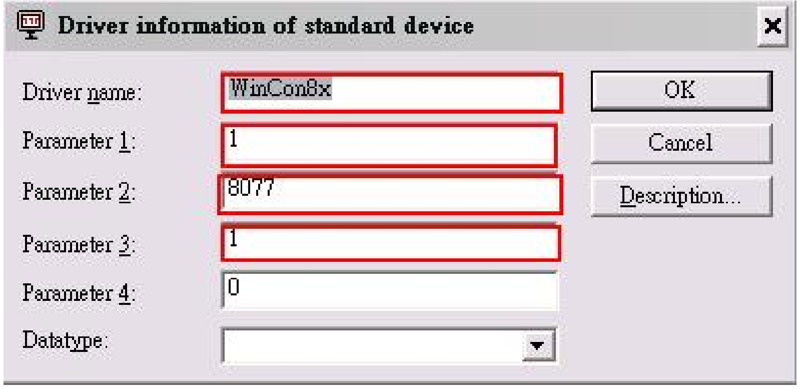
Board / IO Module:

Hilscher CIF
INTERBUS G4
Modbus
User defined Input

Comment:

Step 2: Enter parameters.

Driver name: The driver name of the KinCon I/O-Driver is “WinCon8x”.
Parameter 1: The slot number of the KinCon (1).
Parameter 2: The module type ID of this slot(8077).
Parameter 3: The timeout value for this module(ms).
Parameter 4: Some flags used for certain modules.



Driver name: WinCon8x

Parameter 1: 1

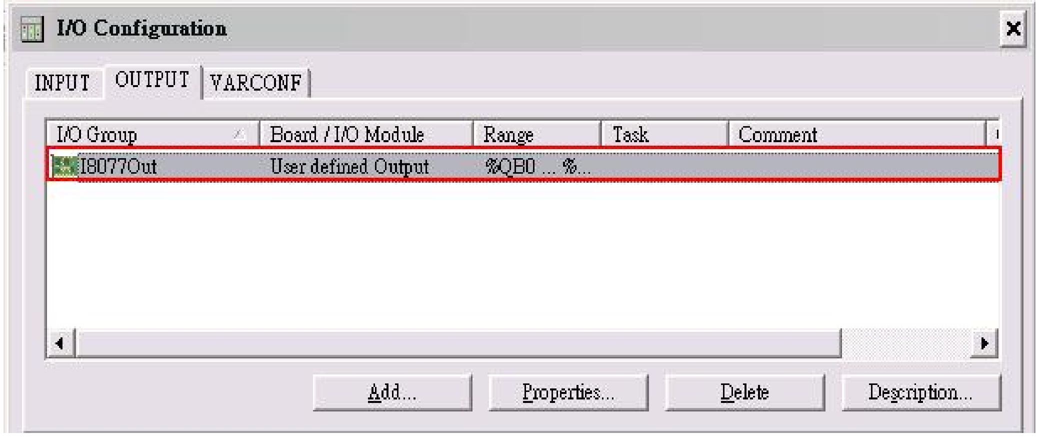
Parameter 2: 8077

Parameter 3: 1

Parameter 4: 0

Datatype:

Step 3: Repeat step1 to step2 in ‘**OUTPUT**’ I/O Group for I-8077 DOs



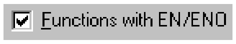
I/O Group	Board / I/O Module	Range	Task	Comment
I8077Out	User defined Output	%QB0 ... %...		

See the APPENDIX for details of the driver parameters.

CREATING AN USER DEFINED FUNCTION

In this chapter, we want to insert an user defined function in our project. The function should be generated using the textual language ST. It counts how often the motor has been active.

Before you start:
Select 'Extras | Options'. In the 'Options' dialog go to the 'Graphical editor' tab and ensure that 'Functions with EN/ENO' is enabled:



☑ Functions with EN/ENO

EN/ENO designates an additional boolean input 'EN' (= enable) and output 'ENO' (= enable output) for IEC 61131 functions in the programming languages LD and FBD.

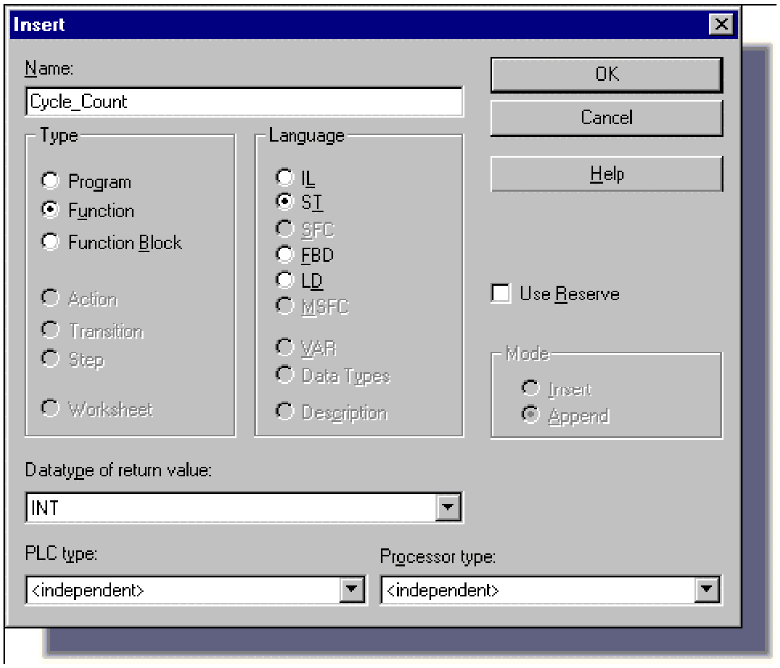
EN/ENO is not supported for all targets.

a. To insert an user defined function, click on the 'Add Function' button in the toolbar:

The 'Insert' dialog appears.

b. Enter 'Cycle_Count' as name and select the language 'ST'. In the 'Datatype of return value' field you specify which datatype is applied to the function output. The variable connected to the function output has to fit to the datatype of the return value. In our example we use 'INT'.

Figure 64: Dialog 'Insert' for inserting an user defined POU



Name: Cycle_Count

Type
○ Program
● Function
○ Function Block
○ Action
○ Transition
○ Step
○ Worksheet

Language
○ IL
● ST
○ SFC
○ FBD
○ LD
○ MSFC
○ VAR
○ Data Types
○ Description

☐ Use Reserve

Mode
○ Insert
● Append

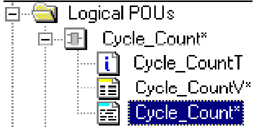
Datatype of return value: INT

PLC type: <independent>

Processor type: <independent>

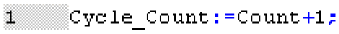
c. Press 'OK'. The function is added to the project tree. The asterisk at the end of the function name indicates that the new POU has not yet been compiled.

d. Open the code worksheet 'Cycle_Count' for editing the ST code by double clicking on it.



```
Logical POUs
    Cycle_Count*
        Cycle_CountT
        Cycle_CountV*
        Cycle_Count*
```

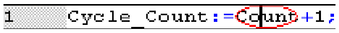
e. In the worksheet, enter the following code:



```
Cycle_Count:=Count+1;
```

This line of code will generate a value that continuously increments each time the motor is started.

f. Place the cursor on the variable 'Count':

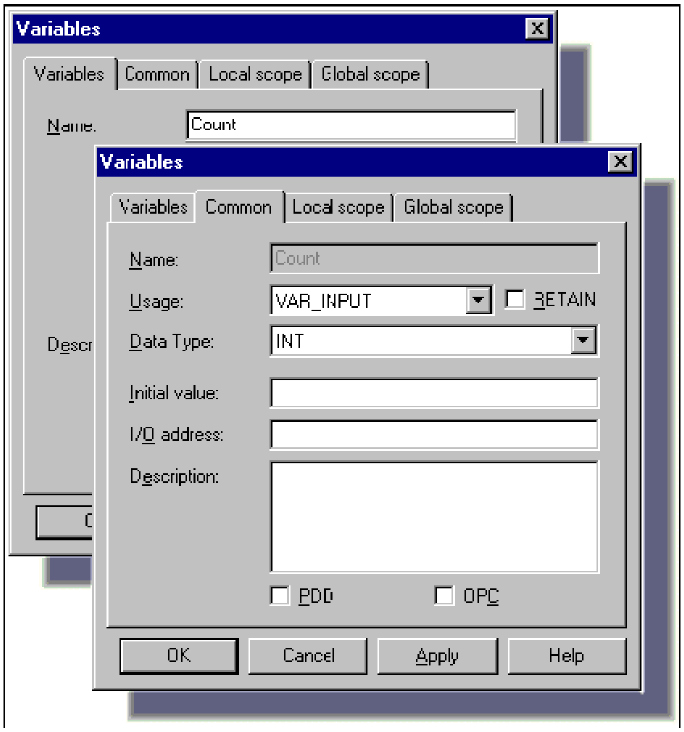


```
Cycle_Count:=Count+1;
```

g. Click on the 'Variables' icon

and declare the local variable as 'INT' and 'VAR_INPUT' as shown in Figure 65:

Figure 65:
Dialog 'Variables'



Name: Count

Usage: VAR_INPUT □ RETAIN

Data Type: INT

Initial value:

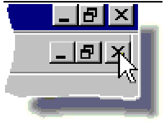
I/O address:

Description:

□ PDD □ OPC

h. Confirm the dialog with 'OK'. The declaration will be automatically inserted in the variables worksheet of the new ST POU.

i. Close the ST worksheet and save the changes.



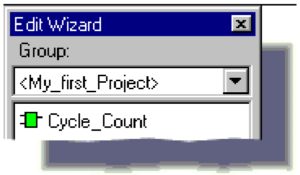
After closing the ST worksheet, the new user defined function is available in the Edit Wizard and can be inserted into other worksheets of the project.

We now want to call the user defined function 'Cycle_Count' in our program POU 'Main'.

j. In the code body of the program 'Main', set the insertion mark below the LD network '003' and insert a new network using the 'Network' icon.

k. Mark the LD network '004' line between C007 and C008.

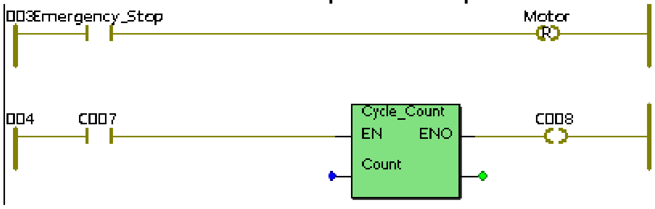
l. Open the Edit Wizard and select the group 'My_First_Project':



This group contains all user defined functions and function blocks of the current project (only 'Cycle_Count' in our example).

m. Double click on the function 'Cycle_Count' to insert this user defined function at the position specified before.

Figure 66: Inserted user defined function 'Cycle_Count'



Having done this, close the Wizard again.

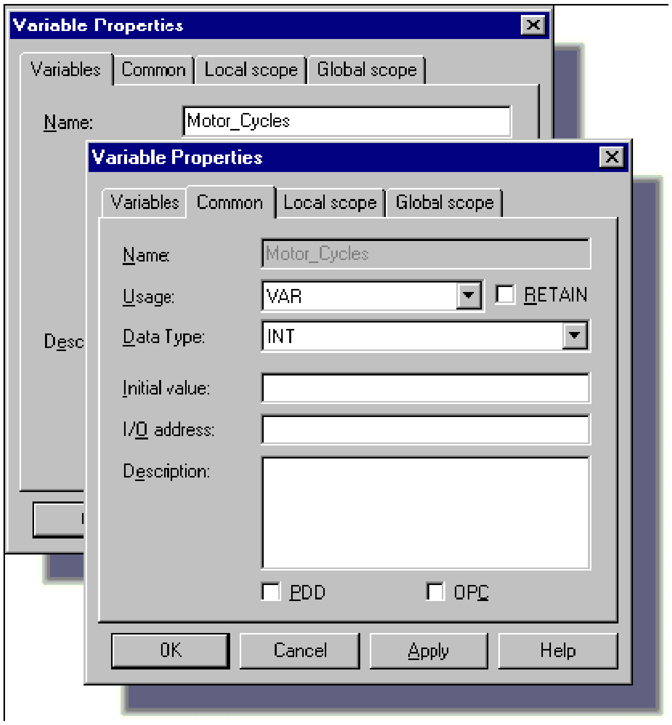
Now we have to connect a new variable to input 'Count' so that we can see the internal value.

n. Double click on the blue connection point of the 'Count' input of 'Cycle_Count'.

o. The 'Variable Properties' dialog appears.

p. Declare the local variable 'Motor_Cycles' as follows:

Figure 67: Dialog 'Variable Properties"



Name: Motor_Cycles

Usage: VAR □ RETAIN

Data Type: INT

Initial value:

I/O address:

Description:

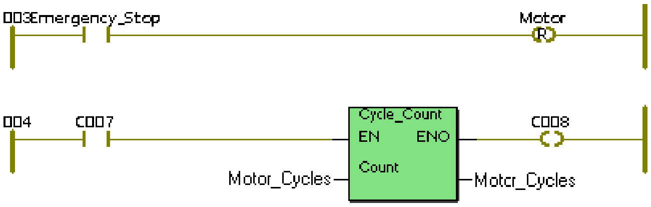
□ PDD □ OPC

q. Confirm the dialogs with 'OK' to insert the variable into the code body and its declaration into the local variables worksheet.

The same variable now has to be connected to the function output.

r. Double click on the green connection point of the output of 'Cycle_Count' and select the variable 'Motor_Cycles' from the variables selection list of the 'Variables Properties' dialog.

s. Confirm with 'OK' to insert the variable.



t. Change the name of contact 'C007' to 'Motor' using the 'Contact/Coil Properties' dialog. To open the dialog double click on the contact.

Be sure that the contact 'Motor' is marked.

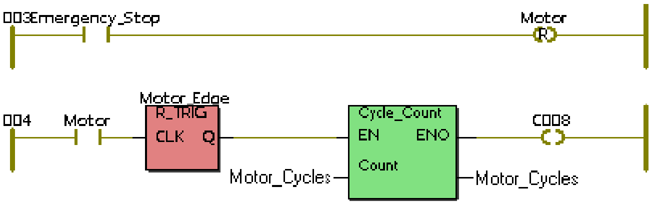
u. Activate the Edit Wizard. Select the group 'Function blocks' and insert the function block R_TRIG by double clicking on it.

v. Enter 'Motor_Edge' in the 'Name' field of the 'Variable Properties' dialog.

w. Press 'OK'. The 'Common' dialog appears. Enter a description if desired and press 'OK' again to insert the function block and its declaration.

x. Hide the Edit Wizard by clicking on the 'Edit Wizard' icon again:

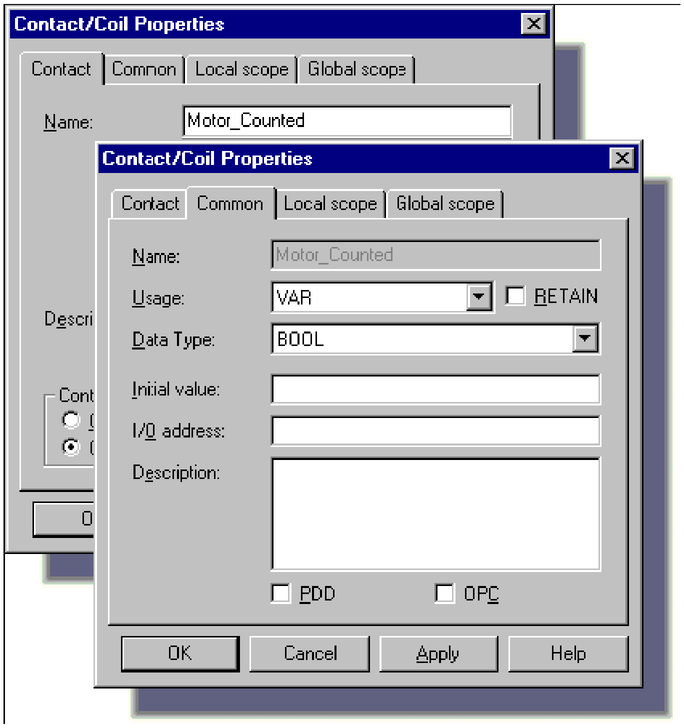
As the contact 'Motor' was marked when selecting the function block in the Edit Wizard, 'Motor_Edge' is directly connected to 'Motor'.



y. Double click on the coil 'C008'.

z. Declare the coil as follows:

Figure 68: 'Contact/Coil Properties' dialog



Name: Motor_Counted

Usage: VAR □ RETAIN

Data Type: BOOL

Initial value:

I/O address:

Description:

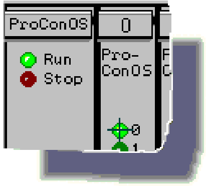
□ PDD □ OPC

Compile the project using the 'Make' icon, start and then download it.

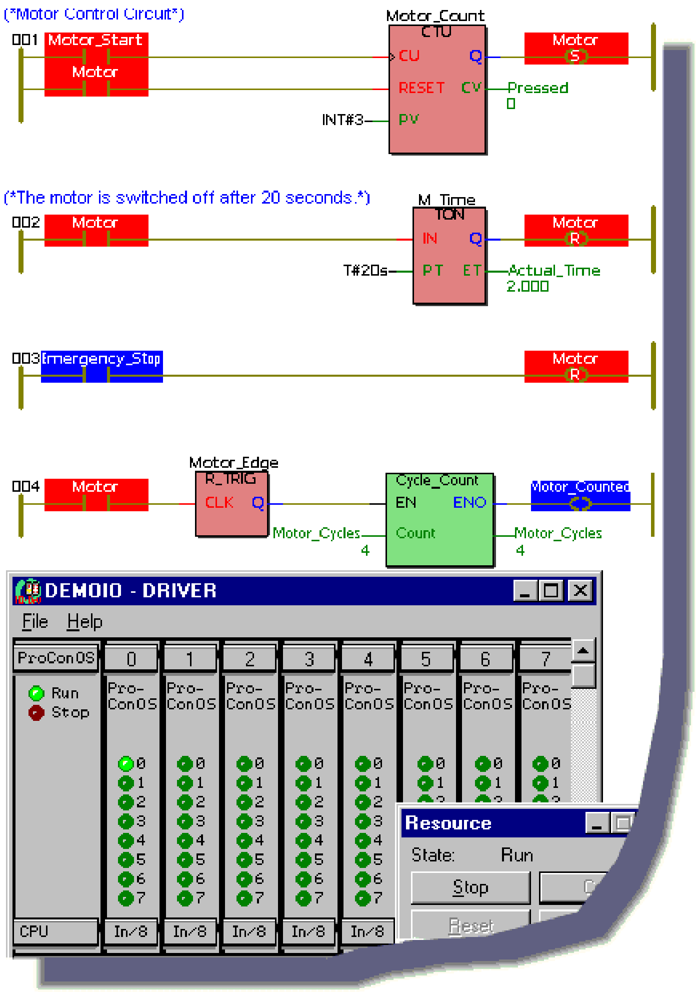
Now our sample project is complete. You can check the behavior of the program using the worksheets in Online mode and the programming system I/O Simulator (refer to Phase 5 on page 37 of this manual).

a. Switch the worksheet into online mode and click on the button 'DEMOIO - DRIVER' in the Windows taskbar to open the I/O Simulator.

b. Turn bit 0 of module 0 on and off three times by clicking on the input point.



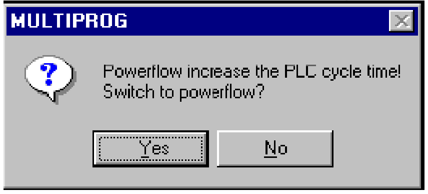
The program executes:



Notice that each time the logic in the main program executes (motor starts, runs 20 seconds and stops) the value of 'Motor_Cycles' is incremented by 1.

It is possible to jump into the user defined function 'Cycle_Count' (i.e. to call the related code body worksheet) without leaving the current worksheet.

a. Double click on the function 'Cycle_Count' in the LD worksheet. The following dialog appears:



Powerflow increase the PLC cycle time!
Switch to powerflow?

b. Confirm the dialog with 'Yes' to switch from variable status to powerflow.
The code body worksheet of the function 'Cycle_Count' will be opened. In the now activated powerflow, the current values of the accumulator are displayed in the worksheet by symbols.

Detailed information about powerflow and the symbols used can be found in the online help system.

c. Close the code body worksheet to return to the LD worksheet.

CHANGING THE TASK CYCLE TIME

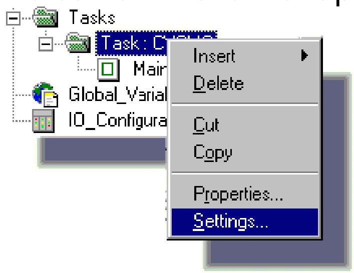
The programming system allows you to change the Task Cycle Time, i.e. the time interval in which the cyclic task is executed. Thus, decreasing the task cycle time will speed up process execution. It is important to get the execution of the cycle as close to scan time as possible.

In our example we change the Task Cycle Time from 100ms to 90ms.

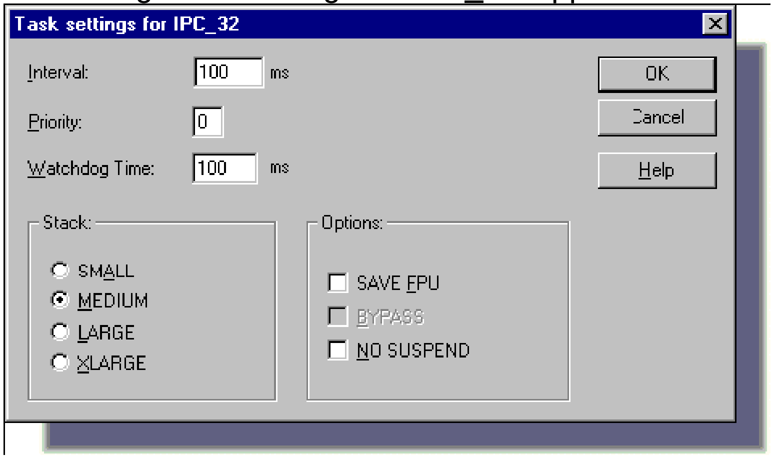
The shortest possible Task Cycle Time depends on the PLC used.

a. Make sure, that the system is in Offline mode, i.e. the icon 'Debug on/off' is not pressed.

b. To change the Task Configuration, right click on the 'TASK : CYCLIC' icon in the subtree 'Physical Hardware'. In the context menu that appears select 'Settings...'.



The dialog 'Task settings for IPC_32' appears:



Task settings for IPC_32

Interval: 100 ms

Priority: 0

Watchdog Time: 100 ms

Stack:
○ SMALL
● MEDIUM
○ LARGE
○ XLARGE

Options:
□ SAVE FPU
□ BYPASS
□ NO SUSPEND

Figure 69: Dialog 'Task settings for IPC_32' for changing the Task Cycle Time

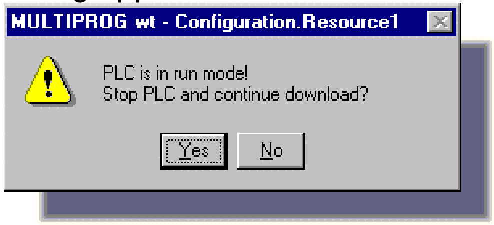
The current Task Configuration is shown in the dialog. We want to change the Task Cycle Time from 100ms to 90ms.

c. In the field 'Interval' enter 90 and click 'OK' to confirm the dialog.

d. Compile the project by clicking on the 'Make' icon. For detailed information on the compilation refer to Phase 3 on page 32 of this Quick Start Manual.

e. Download the project to the target as described in Phase 4 on page 35.

If the PLC/simulation is still running, the following message dialog appears before the download:



MULTIPROG wt - Configuration.Resource1

PLC is in run mode!
Stop PLC and continue download?

If this is the case, click 'Yes' to continue the download.

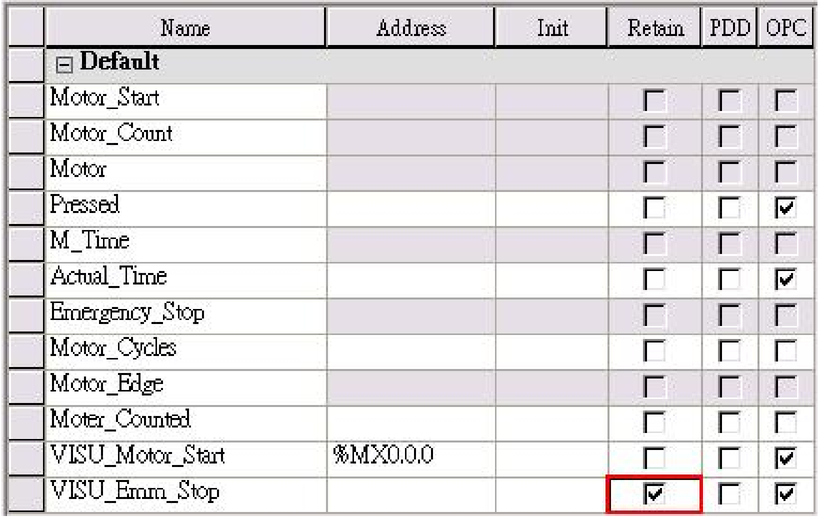
f. If desired, debug the project as described in Phase 5, 'Debugging the project' on page 37 of this manual.

Using Retain Variable

Step1: Make sure the S-256 or S-512 is plugged in KinCon-8000 and refer to '**KinCon-8000 Configuration**' section to start it up correctly.

Step2: Check the '**Retain**' variable in worksheet

NOTE: Retain variable can **NOT** set address. If the variable already has address, please modify it by programming technique.

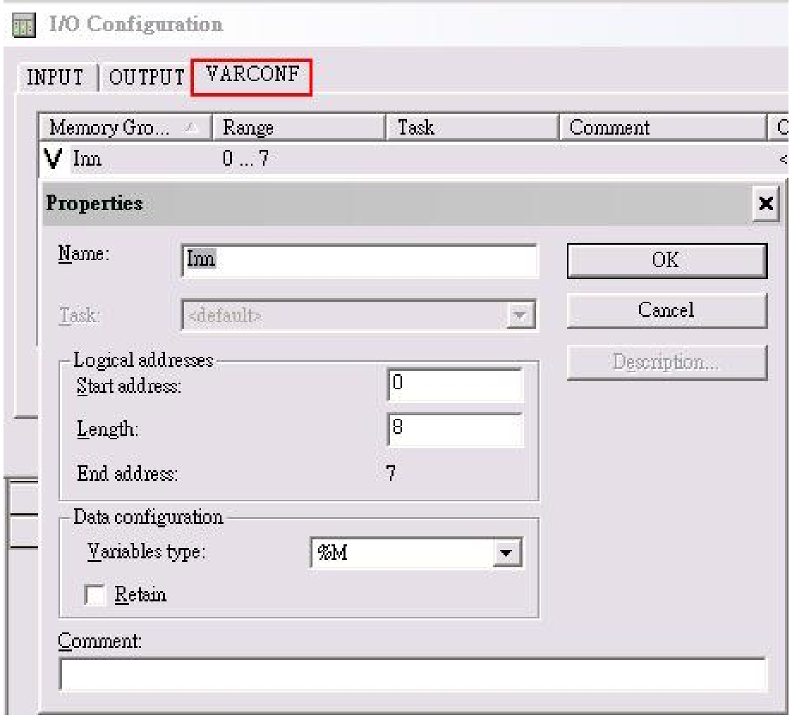


Name	Address	Init	Retain	PDD	OPC
⊟ **Default**					
Motor_Start			☐	☐	☐
Motor_Count			☐	☐	☐
Motor			☐	☐	☐
Pressed			☐	☐	☑
M_Time			☐	☐	☐
Actual_Time			☐	☐	☑
Emergency_Stop			☐	☐	☐
Motor_Cycles			☐	☐	☐
Motor_Edge			☐	☐	☐
Moter_Counted			☐	☐	☐
VISU_Motor_Start	%MX0.0.0		☐	☐	☑
VISU_Emm_Stop			☑	☐	☑

Step3: Re-Compile, and download to KinCon-8000

Using Modbus TCP Slave

Modbus TCP Slave function is default setting in KinCon-8000. You do not need to start it specially but need to add %M declaration in 'VARCONF' of I/O Group in MultiPROG. The range depends on how many variables you want to use. Also, you have to give the variable a modbus address at 'Address' column. The relationship between 'Modbus address' and 'Internal variable address'. Please refer to APPENDIX.



I/O Configuration

Memory Gro...	Range	Task	Comment	C
V Inn	0 ... 7			

Properties

Name: Inn

Task: <default>

Logical addresses

Start address: 0

Length: 8

End address: 7

Data configuration

Variables type: %M

☐ Retain

Comment:

After that, third party software which supports ‘Modbus TCP Master’ can access data via LAN1 and LAN2 of KinCon-8000 easily.

Using Modbus RTU Slave

Modbus RTU Slave function needs to configure at ProCos.bat. Please refer to ‘Page 30: Configuration’.

Start with Modbus/RTU Slave at baudrate:19200 COM port:2 Slave No: 1:

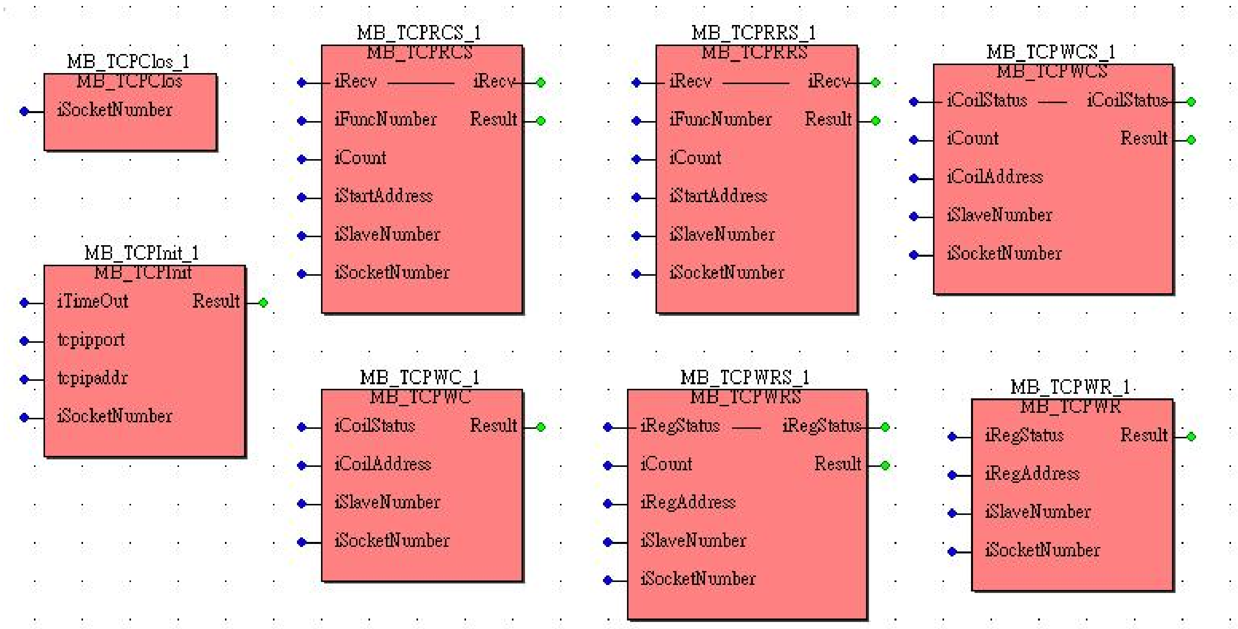
ProCos.bat Context
@echo off \CompactFlash\KW_Pcos\pcwce4.exe –B19200 –COM2 –SN1

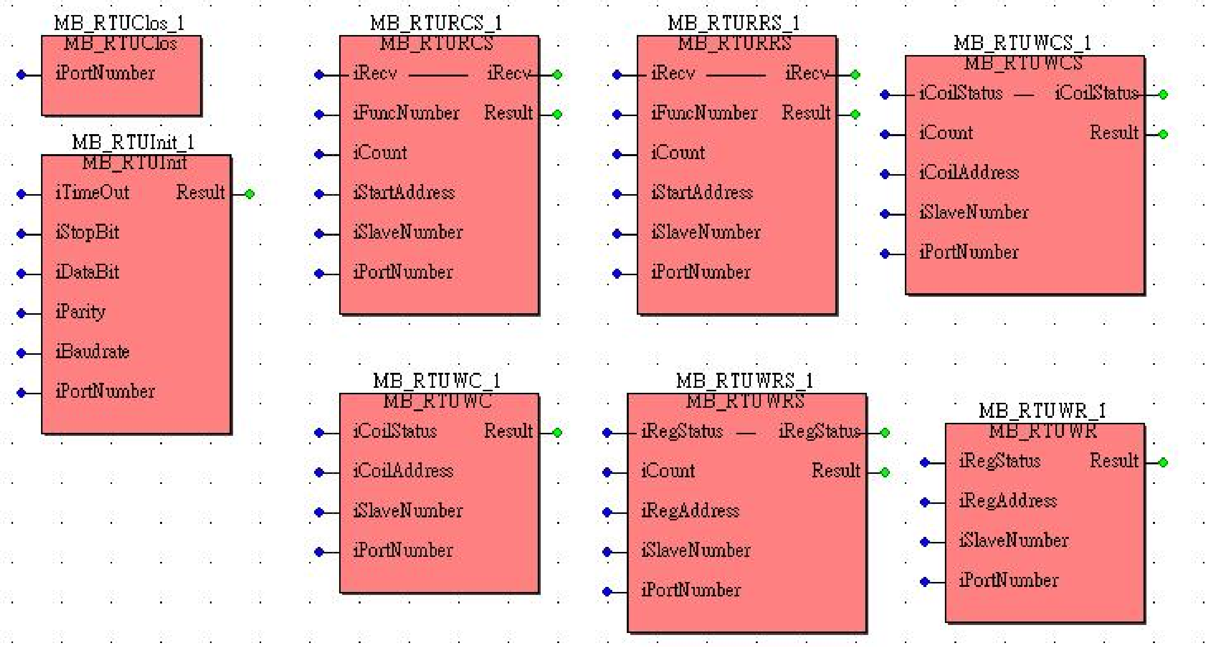
NOTE: Please double check the setting whether it conflicts with I/O configuration of remote devices. COM port can NOT share with another use.

Also, you need to add %M declaration in ‘VARCONF’ of I/O Group in MultiPROG. The range depends on how many variables you want to use. And eyou have to give the variable a modbus address at ‘Address’ column. The relationship between ‘Modbus address’ and ‘Internal variable address’. Please refer to ‘APPENDIX - Modbus Address V.S. Internal Address’.

Using Modbus TCP/RTU Master

KinCon-8000 v1.02 implements 8 ‘Modbus TCP Master Function Blocks’ and 8 ‘Modbus RTU Master Function Blocks’. In order to help you developing your modbus master project, ICPDAS provides a ‘MBMaster’ library including 16 modbus master function blocks and a ‘Modbus_Master’ template including necessary data type declaration and test POUs. We highly recommend starting your modbus master project from this template.





Step1: Configure Modbus Library in MultiProg

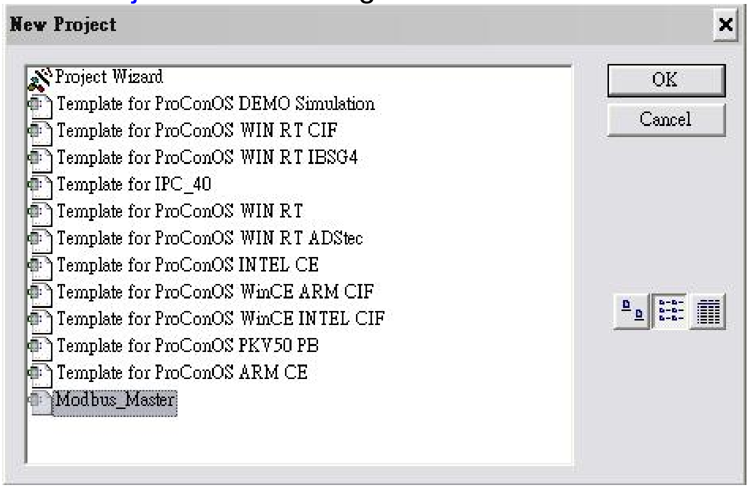
Copy 'MBMaster' folder from product CD\\KW-Software\FW_Lib to your MULTIPROG ROOT\\PLC\FW_LIB. MBMaster library also can download from 'Download Library' at http://www.icpdas.com/products/PAC/kincon/indusoft_kincon.htm

Step2: Configure Template in MultiProg

Copy 'Modbus_Master' folder and 'Modbus_Master.twt' from product CD\\KW-Software\Templates\ to your MULTIPROG ROOT\\templates\ . 'Modbus_Master' folder and 'Modbus_Master.twt' also can download from 'Download Template' at http://www.icpdas.com/products/PAC/kincon/indusoft_kincon.htm

Step3: New Project in MultiProg

'New Project' in MultuProg and then choose 'Modbus_Master'



New Project

Project Wizard
Template for ProConOS DEMO Simulation
Template for ProConOS WIN RT CIF
Template for ProConOS WIN RT IBSG4
Template for IPC_40
Template for ProConOS WIN RT
Template for ProConOS WIN RT ADStec
Template for ProConOS INTEL CE
Template for ProConOS WinCE ARM CIF
Template for ProConOS WinCE INTEL CIF
Template for ProConOS PKV50 PB
Template for ProConOS ARM CE
Modbus_Master

This template includes 1 'MBMaster' library, 1 'Data type' declaration, 5 POUs, 3 Tasks and I-8077/I-8024/I-8017H settings. This template can make KinCon to be 'Modbus TCP Slave' and 'Modbus RTU Slave'. Also, to be 'Modbus TCP Master' and 'Modbus RTU Master'. KinCon-8000 can communicate to itself by Modbus TCP and Modbus RTU protocol. The settings are as bellows:

I/O Configuration:

Slot 1: I-8077
Slot 2: I-8024
Slot 3: I-8017H

I-8024_Vout0 -----------→ I-8017H_Vin4
I-8024_Vout1 -----------→ I-8017H_Vin5
I-8024_Vout2 -----------→ I-8017H_Vin6
I-8024_Vout3 -----------→ I-8017H_Vin7

Libraries:

MBMaster Library

Data Types:

SYS_FLAG_TYPE
Include MB_R_Coils, MB_W_Coils, MB_R_Regs, and MB_W_Regs array declaration.

Logical POUs:

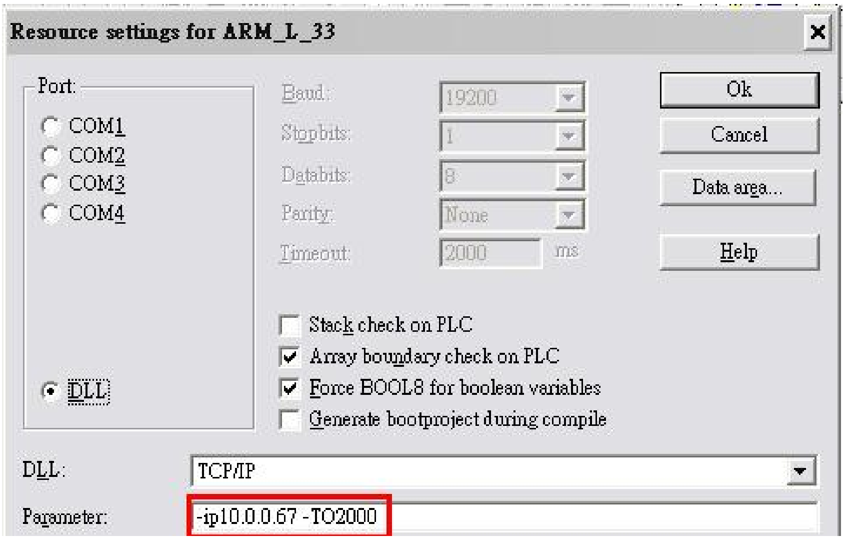
1) Assign: Connect physical and virtual variables
2) MBTCP_RW_CoilANDReg: Modbus TCP Read/Write Coils & Registers
3) MBRTU_RW_Coil: Modbus RTU Read/Write Coils
4) MBRTU_RW_Register: Modbus RTU Read/Write Registers
5) STOP: Stop Modbus TCP/RTU Master

Tasks:

1) Assign:DEFAULT: Bind 'Assign' POU
2) Stop:SYSTEM: Bind 'Stop' POU
3) Task:CYCLE: You can bind 'MBTCP_RW_CoilANDReg', 'MBRTU_RW_Coil', or 'MBRTU_RW_Register' POU

Step4: Modify your Resource

Set IP parameter with KinCon-8000 IP address in Resource settings for ARM_L_33. In this template, the IP address is '10.0.0.67'



Resource settings for ARM_L_33

Port:
○ COM1
○ COM2
○ COM3
○ COM4
● DLL

Baud: 19200
Stopbits: 1
Databits: 8
Parity: None
Timeout: 2000 ms

□ Stack check on PLC
☑ Array boundary check on PLC
☑ Force BOOL8 for boolean variables
□ Generate bootproject during compile

DLL: TCP/IP
Parameter: -ip10.0.0.67 -TO2000

Step5: Test Modbus TCP

1) Task:CYCLE: Bind 'MBTCP_RW_CoilANDReg'
2) 'Make' and 'Download' to KinCon-8000
3) Cold start

Step6: Test Modbus RTU - Coils

1) Check the context of ProCos.bat:

ProCos.bat Context
@echo off \CompactFlash\KW_Pcos\pcwce4.exe -B 19200 -COM 2 -SN 2 -ST 4

2) Connect COM2 and COM3 via I-7520
3) Task:CYCLE: Bind "MBRTU_RW_Coil'
4) 'Make' and 'Download' to KinCon-8000
5) Cold start

Step7: Test Modbus RTU - Registers

1) Check the context of ProCos.bat:

ProCos.bat Context
@echo off \CompactFlash\KW_Pcos\pcwce4.exe -B 19200 -COM 2 -SN 2 -ST 4

2) Connect COM2 and COM3 via I-7520
3) Task:CYCLE: Bind "MBRTU_RW_Register'
4) 'Make' and 'Download' to KinCon-8000
5) Cold start

NOTE: For more information of Modbus FBs, please refer to **APPENDIX - Modbus TCP/RTU Master FBs**.

PART 3: THE OPC SERVER

INTRODUCTION

What is the OPC Server?

"OPC" means OLE for Process Control and defines the communication between Windows NT, Windows 2000 and Windows XP applications.

Thus, the OPC Server enables the communication between any OPC Client (e.g. ProVisIT) and your PLC (or simulation in our current context).

Via the OPC Server any OPC Client can read and write variable values from/to the running PLC in order to visualize and control the running processes.

Only variables stored in the CSV file of a project can be used by an OPC Server. This requires that the appropriate OPC flags are set in the programming system (please refer to the topic "

Generating the CSV file" starting on page 63). Otherwise, the variables are not written into the CSV file and therefore can neither be read nor written by the OPC Server.

Starting the OPC Server

The OPC Server is started automatically, if an OPC Client is started which is connected to the server. In our context, two clients are available: The OPC Test Client (see page 67) and

the visualization ProVisIT.
For example, the OPC Server starts automatically, if you are browsing for an OPC variable in the visualization's 'Variable Browser' (see page 78) or if the visualization is switched to runtime mode (page 92).

ADDING AN OPC RESOURCE

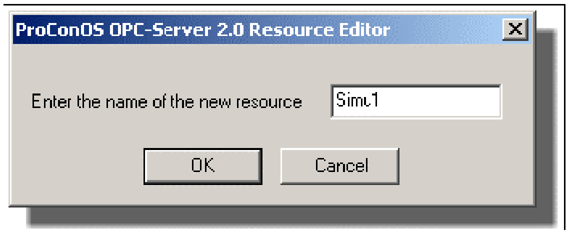
On PC:

As already mentioned, the OPC Server reads and writes values from/to a PLC. For that purpose the communication between PLC and OPC Server must be established.

This is done by defining an OPC resource for each PLC to be connected using the OPC Resource Editor. In our Quick Start Manual we have to add the **PLC simulation** as a new OPC resource. Proceed as follows:

a. In the KW-Software program group start the 'OPC Resource Editor' by selecting the corresponding icon. As you can see now, the Resource Editor consists of only one dialog.

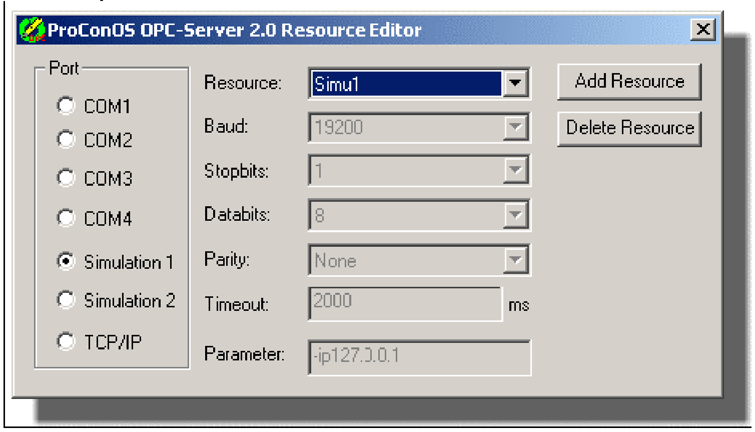
b. Click on the button 'Add Resource' and enter 'Simu1' into the appearing dialog (representing simulation 1). Then confirm with 'OK'.



ProConOS OPC-Server 2.0 Resource Editor

Enter the name of the new resource: Simu1

c. Define the resource settings as shown below. As we are using our simulation no interface settings or TCP/IP settings are required.



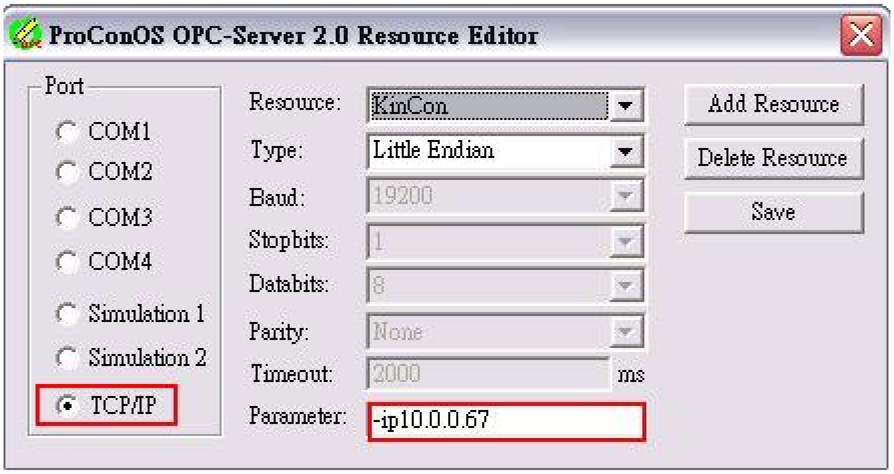
ProConOS OPC-Server 2.0 Resource Editor

Port:
○ COM1
○ COM2
○ COM3
○ COM4
● Simulation 1
○ Simulation 2
○ TCP/IP

Resource: Simu1
Baud: 19200
Stopbits: 1
Databits: 8
Parity: None
Timeout: 2000 ms
Parameter: -ip127.0.0.1

Figure 70:
OPC Resource Editor with added 'Simu1" resource

NOTE: If you want to connect OPC Server in KinCon-8000, please choose TCP/IP and enter the IP address of KinCon-8000.



ProConOS OPC-Server 2.0 Resource Editor

Port: ○ COM1 ○ COM2 ○ COM3 ○ COM4 ○ Simulation 1 ○ Simulation 2 ● TCP/IP

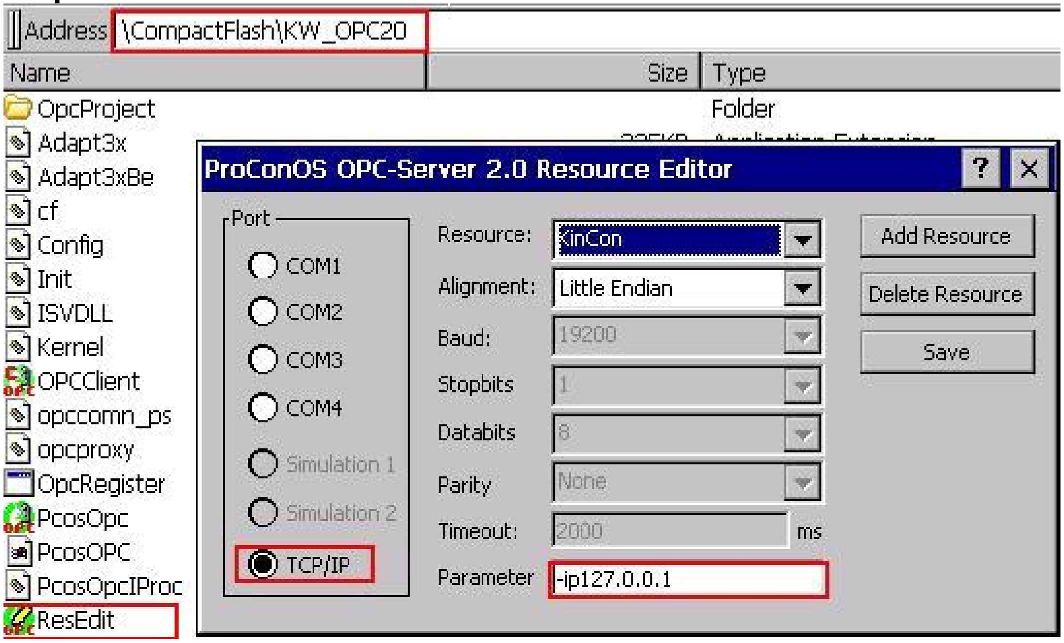
Resource: KinCon
Type: Little Endian
Baud: 19200
Stopbits: 1
Databits: 8
Parity: None
Timeout: 2000 ms
Parameter: -ip10.0.0.67

d. Close the OPC Resource editor.
The resource is now added to the OPC Server. Each time, the server is started by an OPC Client, you can browse within the 'Simu1' resource for OPC variables.

On KinCon-8000:

Also, you should configure the ‘**Resource**’ of OPC server in KinCon-8000. Process as follows:

Step1: Execute \CompactFlash\KW_OPC20\ResEdit.exe
Step2: Press ‘**Add Resource**’ button
Step3: Parameter is local IP: 127.0.0.1



Address \CompactFlash\KW_OPC20

Name	Size	Type
OpcProject		Folder
Adapt3x		
Adapt3xBe		
cf		
Config		
Init		
ISVDLL		
Kernel		
OPCClient		
opccomn_ps		
opcproxy		
OpcRegister		
PcosOpc		
PcosOPC		
PcosOpcIProc		
ResEdit		

ProConOS OPC-Server 2.0 Resource Editor

Port: ○ COM1 ○ COM2 ○ COM3 ○ COM4 ○ Simulation 1 ○ Simulation 2 ● TCP/IP

Resource: KinCon
Alignment: Little Endian
Baud: 19200
Stopbits 1
Databits 8
Parity None
Timeout: 2000 ms
Parameter -ip127.0.0.1

GENERATING THE CSV FILE

As already mentioned, the OPC Server only considers variables which are listed in the OPC CSV file.

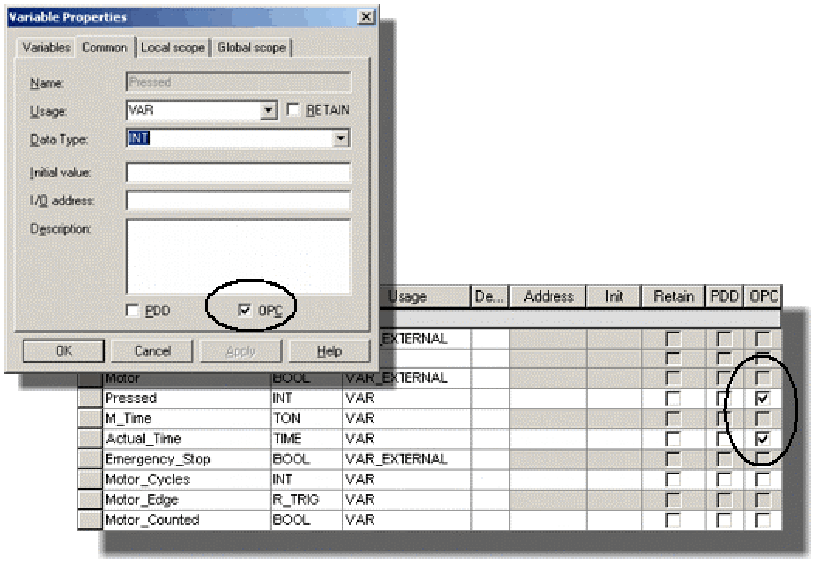
This file is generated by the programming system when building the project. It has to be included when downloading the project to the PLC.

Which variables are included in the CSV file?

Basically there are two "flags" in the programming system deciding which variables are contained in the CSV file.

- For each variable an OPC flag exists in the variables properties dialog and in the variables grid respectively:

Figure 71:
OPC flag for each individual variable



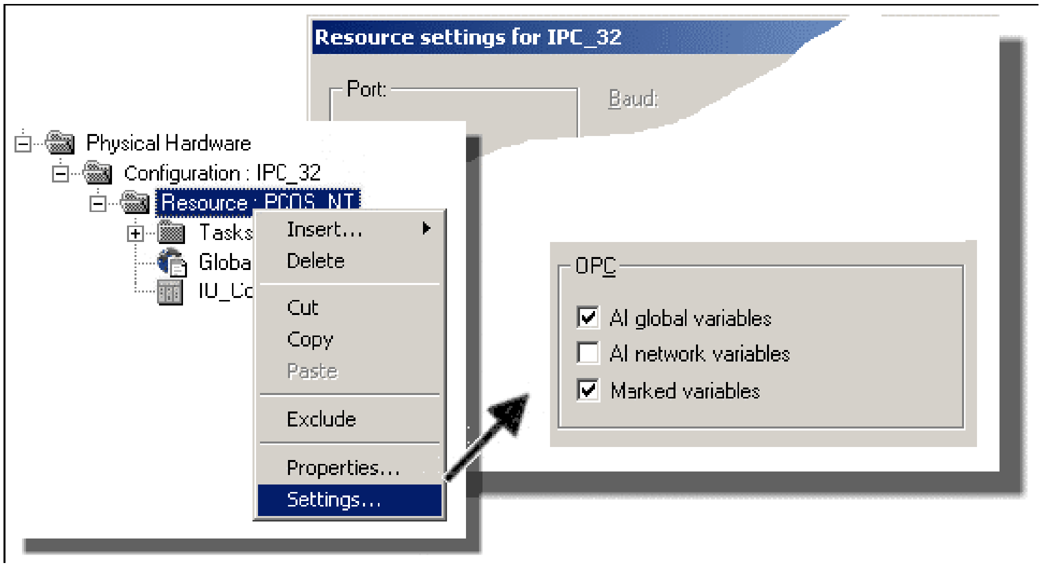
Name: Pressed
Usage: VAR □ RETAIN
Data Type: INT
Initial value:
I/O address:
Description:
□ PDD ☑ OPC

		Usage	De...	Address	Init	Retain	PDD	OPC
		EXTERNAL				□	□	□
						□	□	□
Motor	BOOL	VAR_EXTERNAL				□	□	□
Pressed	INT	VAR				□	□	☑
M_Time	TON	VAR				□	□	□
Actual_Time	TIME	VAR				□	□	☑
Emergency_Stop	BOOL	VAR_EXTERNAL				□	□	□
Motor_Cycles	INT	VAR				□	□	□
Motor_Edge	R_TRIG	VAR				□	□	□
Motor_Counted	BOOL	VAR				□	□	□

These individual flags are only considered, if the flag 'Marked variables' is checked in the resource settings (see next item).

- Further OPC flags are available in the 'Resource settings' dialog which is called via the context menu of the resource in the project tree of the programming system.

In the 'OPC' area of this dialog three different settings are possible. Please note, that the entry 'Marked variables' relates to the OPC flag of each individual variable (see item above).



☑ Al global variables
□ Al network variables
☑ Marked variables

Figure 72:
OPC settings in the 'Resource settings' dialog

PREPARING AND DOWNLOADING THE PROJECT WITH OPC DATA

Before we can access the project variables via the OPC server, we have to set the OPC flags accordingly, rebuild the modified project and download it to the PLC including its OPC data (i.e. the CSV file).

a. If already closed, start the programming system and open the project 'My_first_project.mwt' again.

b. Open the 'Main' variables grid worksheet.

c. In the variables grid, ensure that the 'OPC' flag is set for the local variables we want to access via the OPC Server. Mark the checkbox for the variables 'Pressed' and 'Actual_Time'. The grid is shown on page 64

'Motor' and 'Motor_Start' are global variables for which the 'OPC' setting is done in the resource settings (see next step).

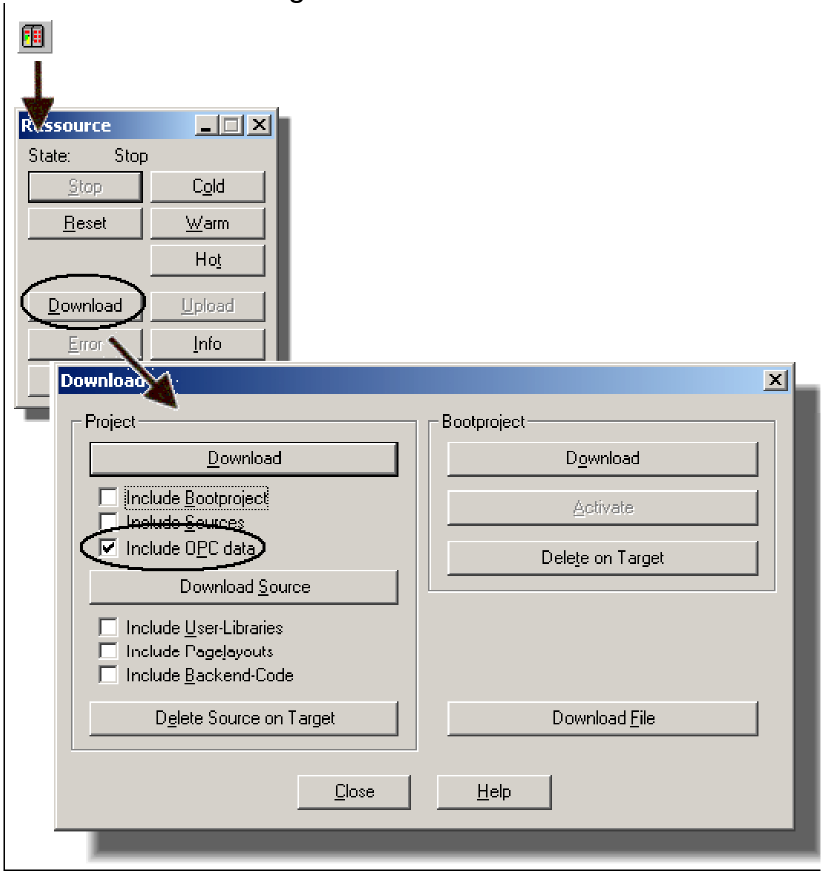
d. Right click on the 'Resource' node in the subtree 'Physical Hardware' and select the context menu item 'Settings...' (see Figure 72).

In the 'OPC' area of the appearing 'Resource settings' dialog activate the checkboxes 'All global variables' and 'Marked variables' as shown in the figure above. Click 'OK' to confirm

the settings.
e. Compile the modified sample project by clicking on the 'Make' icon in the toolbar as described starting at page 32.

f. Download the changed project to the PLC (simulation) as described starting at page 35.

Ensure that the checkbox 'Include OPC data' is marked in the 'Download' dialog!



☐ Include Bootproject
☐ Include Sources
☑ Include OPC data
☐ Include User-Libraries
☐ Include Pagelayouts
☐ Include Backend-Code

Figure 73: Including OPC data (CSV file) when downloading a project

g. After the download has been completed, press the 'Cold' button in the control dialog to execute a cold start:

Cold

h. Exit the programming system.

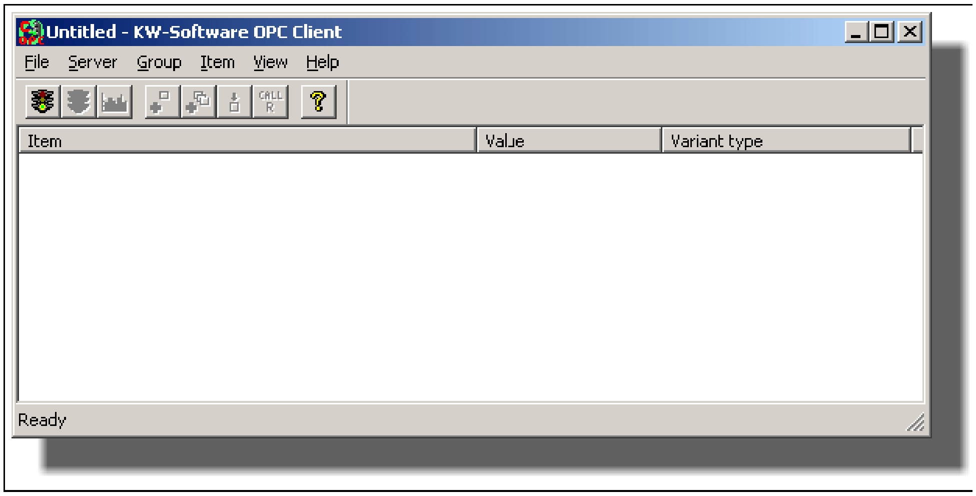
Now that the PLC is running with the modified project and the OPC data are downloaded with the newly compiled project, we are able to access the OPC variables via the OPC Server using the OPC Test Client.

USING THE OPC TEST CLIENT

By the means of the OPC Test Client, you can monitor and manipulate variables processed on the PLC via the OPC Server. It can be used to simulate any other OPC Client (e.g. the visualization) in order to verify the communication between client and server.

a. Start the OPC Test Client by double clicking on the program icon in the folder 'KW-Software\Tools'. The Client appears with an empty workspace.

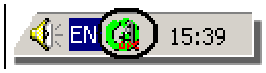
Figure 74: Empty OPC Test Client



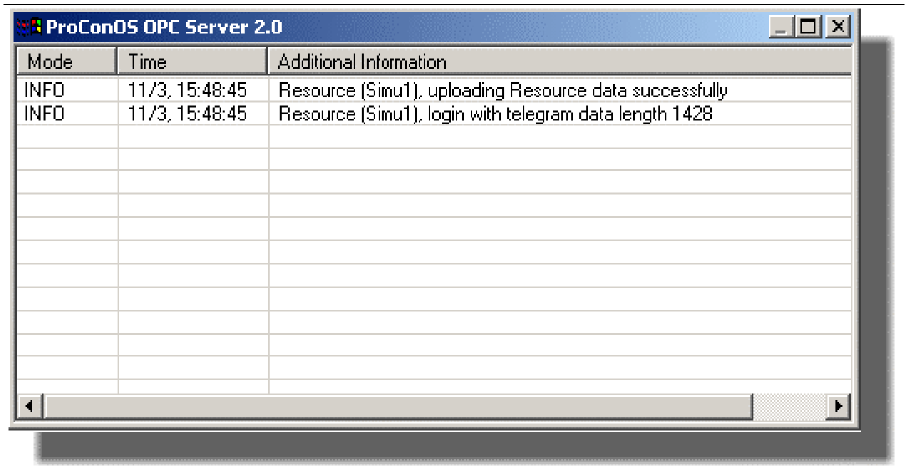
b. Connect the Test Client to the OPC Server by clicking on the 'Connect' icon in the toolbar:

If the OPC Server is not already running, the 'Connect' command automatically starts it. When the OPC Server is running, the other toolbar icons for adding items, disconnecting, etc. become active.

When the OPC Server is running, its icon is displayed in the SysTray on your desktop:



Right clicking on the icon calls a context menu. Select the entry 'Server Status' to get the following status dialog which is continuously updated:



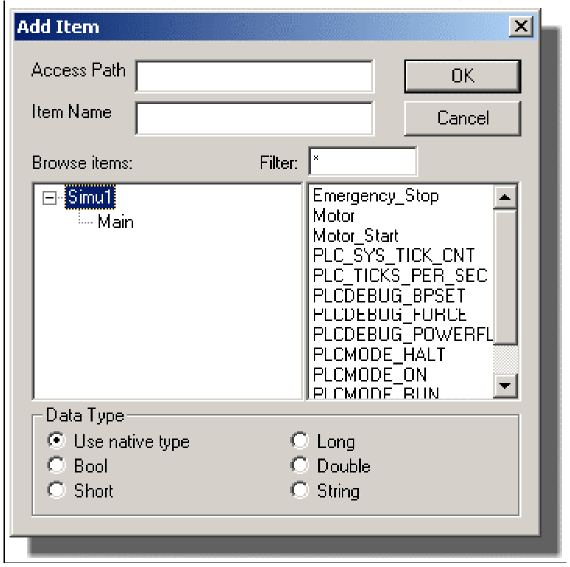
```
Mode   Time           Additional Information
INFO   11/3, 15:48:45 Resource (Simu1), uploading Resource data successfully
INFO   11/3, 15:48:45 Resource (Simu1), login with telegram data length 1428
```

c. Close the status dialog.

d. Click on the icon 'Add Item' in the Test Client toolbar:

The browse dialog appears listing all available OPC Resources ('Simu1' in our example).

Figure 75: Adding an OPC variable to the Test Client workspace



Access Path:

Item Name:

Browse items:

Filter: *

Simu1
Main

Emergency_Stop
Motor
Motor_Start
PLC_SYS_TICK_CNT
PLC_TICKS_PER_SEC
PLCDEBUG_BPSET
PLCDEBUG_FORCE
PLCDEBUG_POWERFL
PLCMODE_HALT
PLCMODE_ON
PLCMODE_RUN

Data Type

● Use native type ○ Long
○ Bool ○ Double
○ Short ○ String

Browse for the desired variable (e.g. Emergency_Stop), mark it in the list on the right and confirm with 'OK'.

The browse dialog is then closed and the added item appears in the Test Client workspace.

e. Repeat step d. for each variable to be added. In our example we add the global variables 'Emergency_Stop', 'Motor' and 'Motor_Start' as well as the local variables 'Actual_Time' and 'Pressed' (located in the subfolder 'Main').

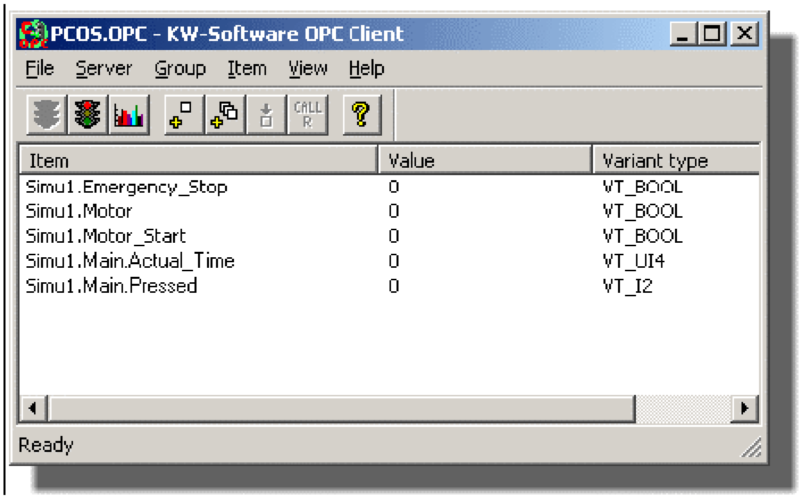


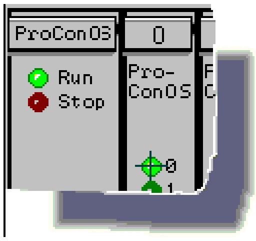
Figure 76:
OPC Test Client with items added from resource 'Simu1'

PCOS.OPC - KW-Software OPC Client
File Server Group Item View Help
Ready

Item	Value	Variant type
Simu1.Emergency_Stop	0	VT_BOOL
Simu1.Motor	0	VT_BOOL
Simu1.Motor_Start	0	VT_BOOL
Simu1.Main.Actual_Time	0	VT_UI4
Simu1.Main.Pressed	0	VT_I2

f. Click on the icon 'DEMOIO – DRIVER' in the Windows taskbar to open the I/O Simulator.
Arrange the Simulator and the OPC Test Client in way, that both are visible.

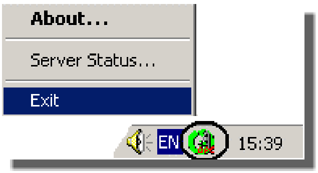
g. Turn bit 0 of module 0 (contact 'Motor_Start') on and off three times to start the motor and watch the reaction in the OPC Test Client.
Also manipulate bit 1 in module 0 ('Emergency_Stop').



h. Having monitored the variables values disconnect the OPC Test Client from the OPC Server by clicking on
🚦.

i. Terminate the OPC Test Client by selecting 'File' > 'Exit'.

After exiting the OPC Client, the OPC Server is shutdown automatically. Otherwise you can also terminate the OPC Server manually by right clicking on the OPC SysTray icon and selecting 'Exit' from the context menu.



Once the OPC Server has been exited, you can also stop the PLC. For that purpose click on the icon 'PcSim32' in the Windows taskbar. In the PcSim32 window press 'Terminate'. The PLC is stopped and shuts down. The I/O Simulator is terminated too.

PART 4: PROVISIT

PREPARING THE SAMPLE PROJECT FOR THE VISUALIZATION

We will now visualize the sample project developed in part 1 of this manual.

The aim is, to design a visualization screen which provides the most important operating and display elements used in our motor control program. The planned elements are shown in the figure on page 74.

Required modifications in our MULTIPROG project

However, some additions are required first in our MULTIPROG project because physical PLC inputs can not be forced. Due to this, the values of the variables 'Motor_Start' and 'Emergency_Stop' can not be manipulated via the visualization. To solve this, we insert a new parallel contact to each of these located variables, declare them as local, non-located variables and designate them '**VISU_Motor_Start**' and '**VISU_Emergency_Stop**'.**Fehler! Verweisquelle konnte nicht gefunden werden.** Using these "dummies" we can control our motor via the visualization.

Inserting additional contacts

a. If already closed, start the programming system and open the project 'My_first_project.mwt' again.

b. Open the 'Main' code body worksheet.

c. In rung 001 mark the contact 'Motor_Start' and click on the toolbar icon 'Add Contact/Coil above'. The new contact appears with its default name:

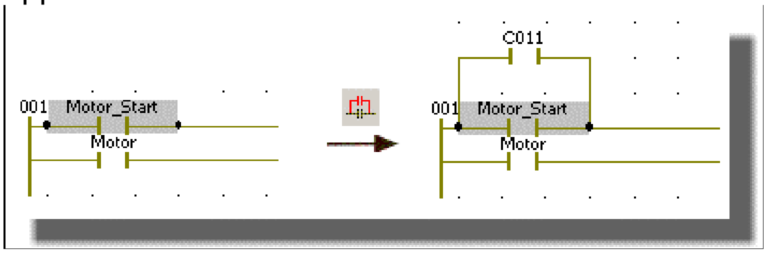
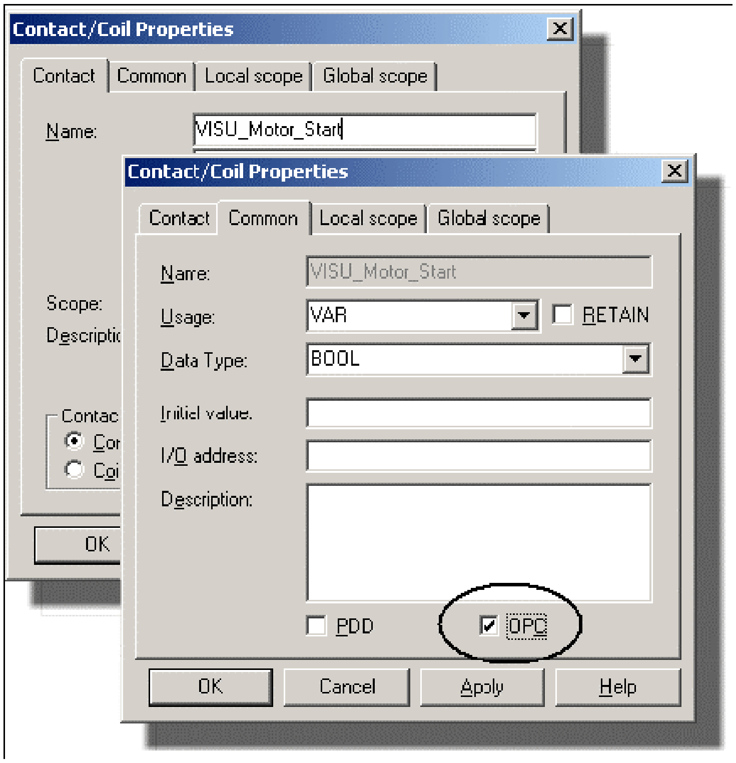


Figure 77: Inserting a parallel contact to 'Motor_Start'

d. Double click on the new contact to open its properties dialog. Define the new local variable 'VISU_Motor_Start' as shown in Figure 78.

Figure 78: Declaring the variable 'VISU_Motor_Start'



Name: VISU_Motor_Start

Name: VISU_Motor_Start

Usage: VAR

☐ RETAIN

Data Type: BOOL

Initial value:

I/O address:

Description:

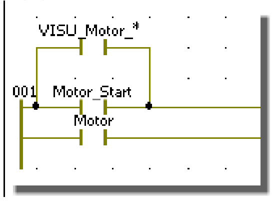
☐ PDD

☑ OPC

Don't forget to mark the checkbox 'OPC'! If this checkbox is not marked, the variable will not be included in the CSV file. This means it can not be read by the OPC server and thus not be used by the visualization.

After confirming the 'Contact/Coil Properties', the contact appears as follows:

Figure 79: Inserting a parallel contact to 'Motor_Start'



The contact 'VISU_Motor_Start' appears with an * because the LD grid width is not big enough to show the entire variable name.

e. In the same way as shown for 'VISU_Motor_Start' you have to insert a parallel contact to 'Emergency_Stop': Designate it 'VISU_Emergency_Stop' and declare it as local Boolean variable. Don't forget to activate the checkbox 'OPC'!

After this, the complete code body worksheet should look as follows:

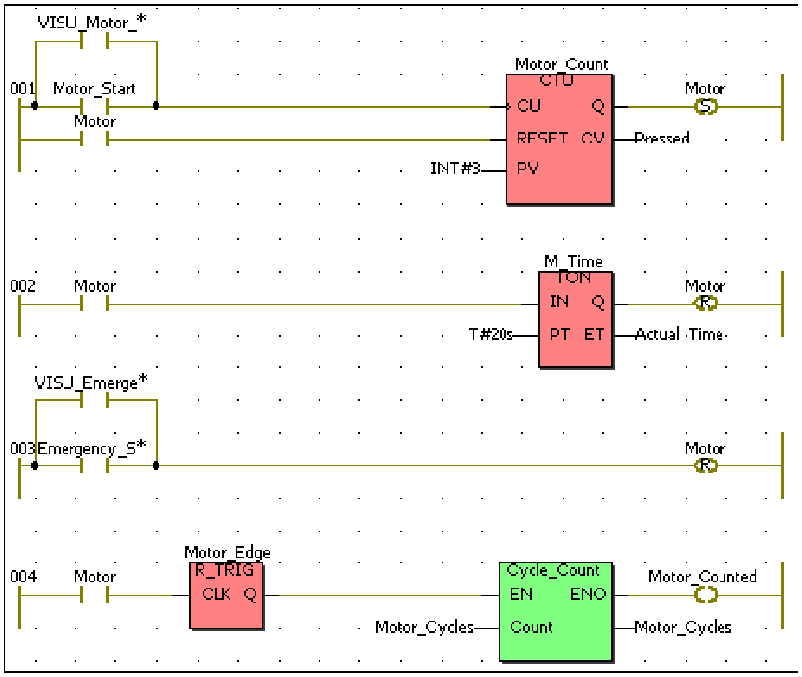
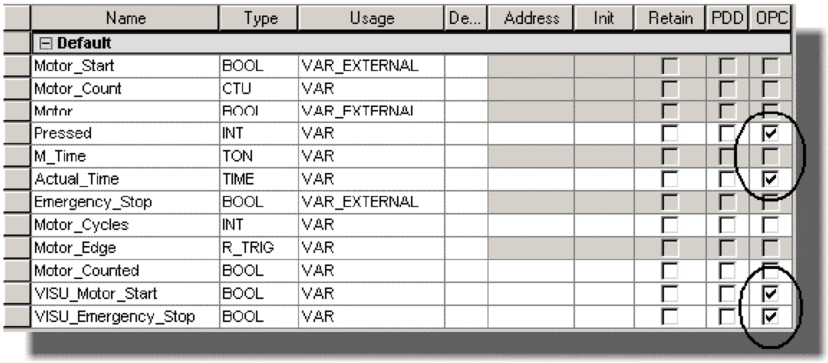


Figure 80: Completed code body worksheet

f. In the 'Main' variables worksheet, ensure that the 'OPC' flag is set for the variables needed by the visualization (we have already set these flags in the OPC chapter of this manual). Note in this context, that 'Motor' is a global variable for which the 'OPC' setting is done in the resource settings (see step g).



Name	Type	Usage	De...	Address	Init	Retain	PDD	OPC
⊟ **Default**								
Motor_Start	BOOL	VAR_EXTERNAL				□	□	□
Motor_Count	CTU	VAR				□	□	□
Motor	BOOL	VAR_EXTERNAL				□	□	□
Pressed	INT	VAR				□	□	☑
M_Time	TON	VAR				□	□	□
Actual_Time	TIME	VAR				□	□	☑
Emergency_Stop	BOOL	VAR_EXTERNAL				□	□	□
Motor_Cycles	INT	VAR				□	□	□
Motor_Edge	R_TRIG	VAR				□	□	□
Motor_Counted	BOOL	VAR				□	□	□
VISU_Motor_Start	BOOL	VAR				□	□	☑
VISU_Emergency_Stop	BOOL	VAR				□	□	☑

Figure 81: OPC settings in the 'Main' variables grid worksheet

g. Ensure that the OPC flags in the resource settings dialog are still set correctly (We have already defined them in the OPC chapter of this manual).

Right click on the 'Resource' node in the subtree 'Physical Hardware' and select the context menu item 'Settings...'.

In the 'OPC' area of the appearing 'Resource settings' dialog the checkboxes 'All global variables' and 'Marked variables' must be activated.

Please refer to the figure on page 65.

h. Compile the modified sample project by clicking on the 'Make' icon in the toolbar as described starting at page 32:

i. Download the changed project to the PLC (simulation) as described in step f on page 66.

Before downloading, ensure that the checkbox 'Include OPC data' is marked in the 'Download' dialog!

Please refer to the figure on page 66.

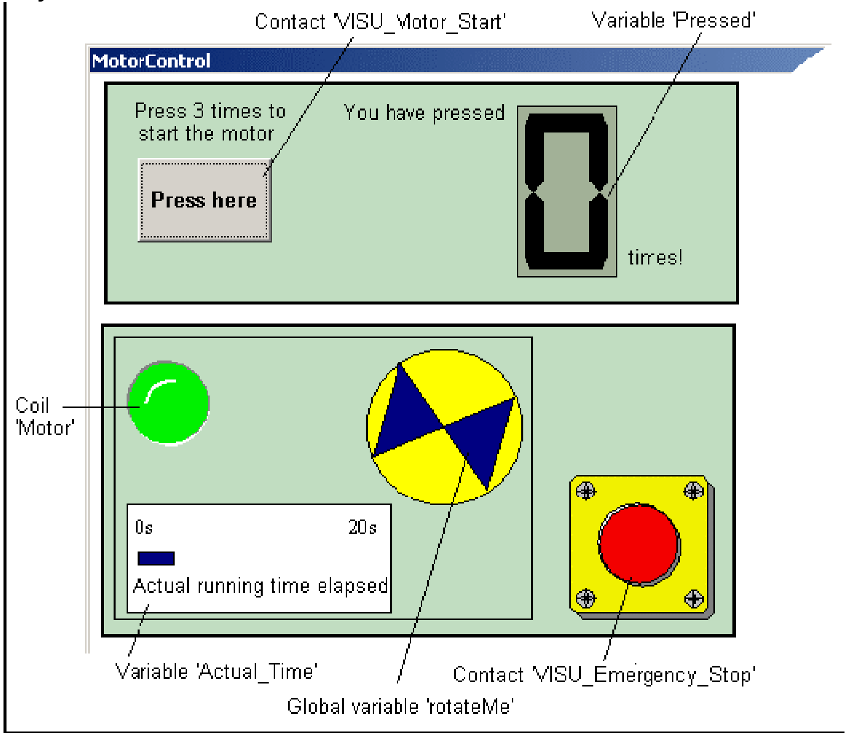
j. After the download has been completed, press the 'Cold' button in the control dialog to execute a cold start:

k. Exit the programming system.

Now that the PLC is running with the modified project and the OPC data are downloaded with the newly compiled project, we are ready to design our visualization project.

DESIGNING A VISUALIZATION PROJECT

Let us begin with a preview to the finished visualization screen (in online mode) to get an appreciation for the project we are going to develop. The figure shows to which variables the objects are connected.



Contact 'VISU_Motor_Start'
Variable 'Pressed'
MotorControl
Press 3 times to start the motor
Press here
You have pressed
0
times!
Coil 'Motor'
0s
20s
Actual running time elapsed
Variable 'Actual_Time'
Contact 'VISU_Emergency_Stop'
Global variable 'rotateMe'

Start the visualization software.

If an already existing project is loaded automatically, you first have to create a new project.

STEP 1

CREATING A NEW VISUALIZATION PROJECT

Click on the 'New Project' icon:

The new project will be created and inserted in the project tree window. The 'Screens' folder already contains one empty visualization worksheet named 'Screen1' which is opened in the design window.

STEP 2

SETTING THE VISUALIZATION SCREEN PROPERTIES

At the beginning of the design process we are going to define the properties of our visualization screen (only one screen will be required in this sample project).

a. In the visualization project tree right click on the worksheet icon and select the context menu item 'Properties...'.

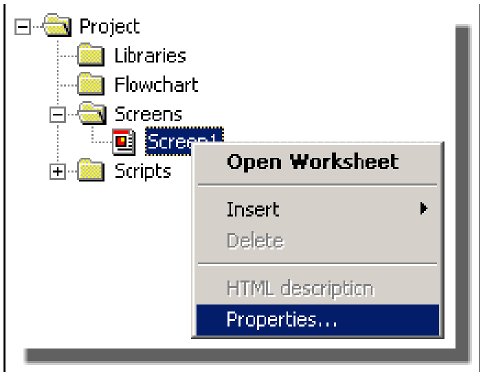


Figure 82: Calling the 'Worksheet Properties' dialog

b. On the page 'General' of the appearing 'Worksheet Properties' dialog enter the name 'MotorControl'.

c. Open the page 'Properties'.
Select the 'Runtime representation' 'Non-modal dialog' and enter '800' (pixel) as dialog 'Width' and '600' as dialog 'Height'. These settings specify, that your visualization screen will appear as modeless dialog during runtime mode.

d. Click 'OK' in the properties dialog.

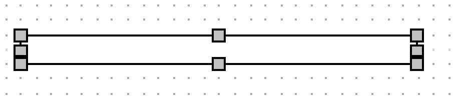
STEP 3

VISUALIZING THE 'ACTUAL_TIME' VARIABLE BY A DYNAMIC RECTANGLE

The first object we are going to design represents the variable 'Actual_Time'. This variable holds the elapsed time, the motor is running. To visualize this variable we use a rectangle which changes its horizontal size depending on the elapsed time.

a. Left click into the design window to activate the design toolbar. Click on the icon 'Rectangle'.

b. In the worksheet draw a rectangle:



The rectangle should change its size depending on the value of the variable 'Actual_Time'. For that purpose we add the dynamic property 'Size' to the object.

c. Right click on the rectangle and select the menu item 'Dynamics > Size'.

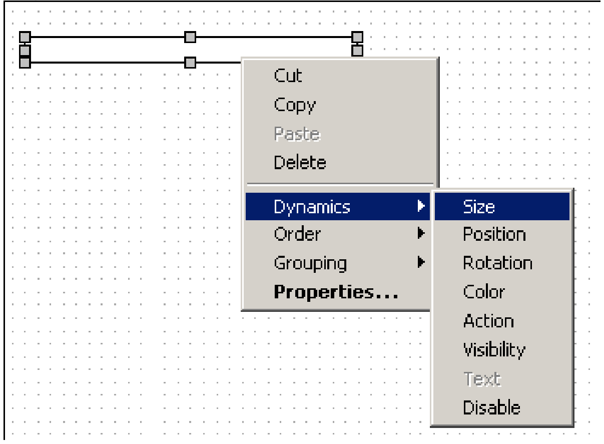
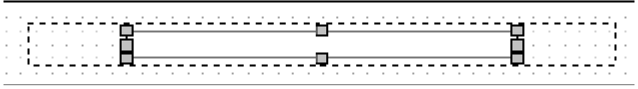


Figure 83: Assigning the dynamic property to a rectangle

A dashed rectangle is added which represents the dynamic property.



Now, the size of the dashed and the solid rectangle representing the minimum and maximum object size have to be defined.

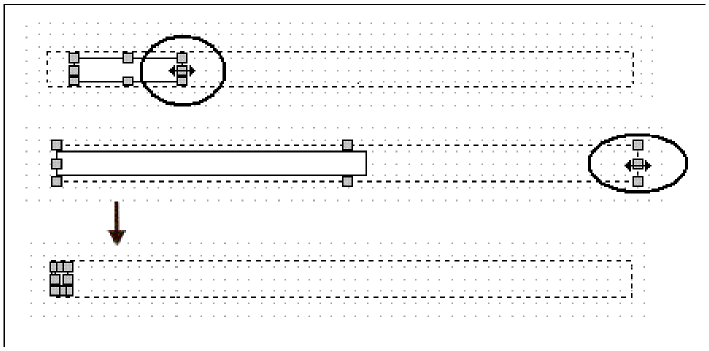
d. Adjust the size of the solid rectangle to the minimum possible width (Actual_Time = 0). Then adjust the size of the dashed rectangle to the desired maximum size (Actual_Time = 20 s).

Changing the size is done by marking the solid/dashed rectangle, placing the mouse pointer on the corresponding object handle and dragging the mouse while keeping the left mouse button pressed.

Observe that both rectangles should be aligned at their left borders and have the same height!

Use the zoom functionality to facilitate drawing and designing. Enlarge the worksheet contents by clicking on the toolbar icon 'Zoom in'.

Figure 84: Rectangle with dynamic property 'Size'



e. Right click on the small solid rectangle and select the context menu item 'Properties...'. The 'Object Properties' dialog appears.

f. Enter 'TimeBar' as 'Name' on the dialog page 'General'.

g. Select a line 'Color', Width' and 'Style' on the dialog page 'Line' and filling properties on the page 'Fill'.
In our example we design the rectangle as follows:
Line: black, 2 point, solid and
Fill: 'Foreground color' blue and 'No Hatch'.

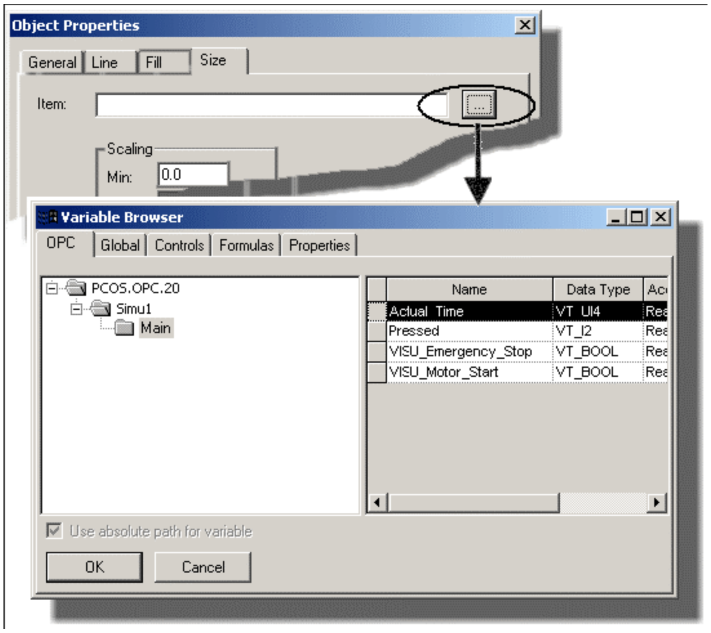
h. On the page 'Size' we have to assign the variable on which the rectangle size should depend. For that purpose click on the browse button beneath the 'Item' field (see figure on next page).

The dialog 'Variable Browser' appears to select the desired variable (see figure on next page).

As we want to assign the rectangle's 'Size' to an OPC variable, the dialog page 'OPC' is applicable.

i. In the tree on the left dialog side open the branch of the PCOS.OPC.20 Server and browse for the resource 'Simu1'.

In part 2 of this manual we have configured our simulation as OPC resource 'Simu1' using the OPC Resource Editor. Due to this definition, 'Simu1' is now supported by the OPC Server.



Item:

Min: 0.0

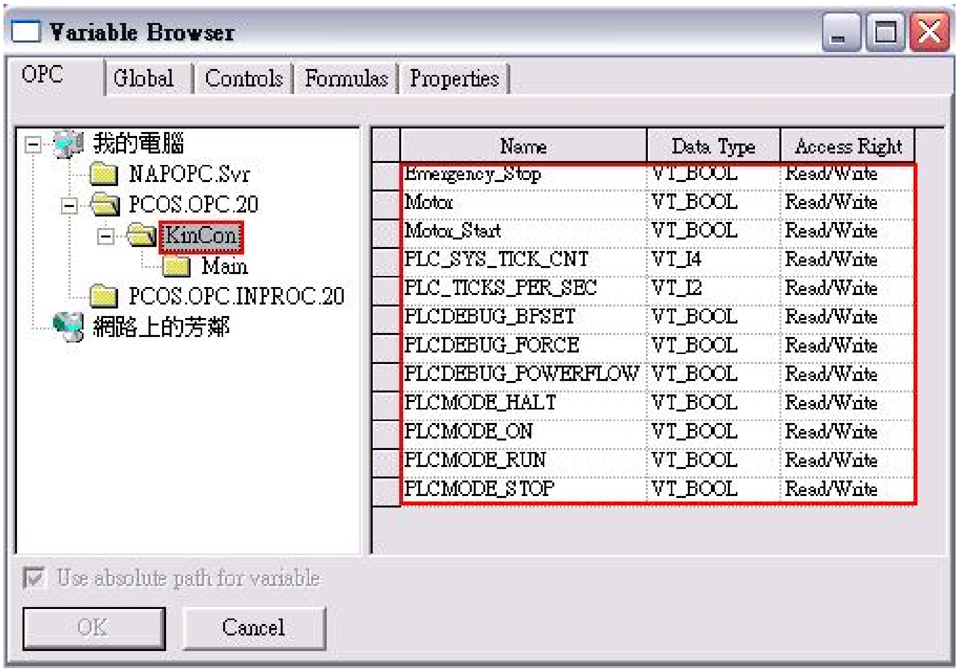
- PCOS.OPC.20
  - Simu1
    - Main

Name	Data Type	Ac
Actual_Time	VT_UI4	Rea
Pressed	VT_I2	Rea
VISU_Emergency_Stop	VT_BOOL	Rea
VISU_Motor_Start	VT_BOOL	Rea

☑ Use absolute path for variable

Figure 85: Assigning a variable to the dynamic property 'Size' using the dialog 'Variable Browser'

If ‘**OPC Resource’** has already linked to KinCon-8000, you can choose KinCon folder and choose the variable inside.



- 我的電腦
  - NAPOPC.Svr
  - PCOS.OPC.20
    - KinCon
      - Main
  - PCOS.OPC.INPROC.20
- 網路上的芳鄰

Name	Data Type	Access Right
Emergency_Stop	VT_BOOL	Read/Write
Motor	VT_BOOL	Read/Write
Motor_Start	VT_BOOL	Read/Write
PLC_SYS_TICK_CNT	VT_I4	Read/Write
PLC_TICKS_PER_SEC	VT_I2	Read/Write
PLCDEBUG_BPSET	VT_BOOL	Read/Write
PLCDEBUG_FORCE	VT_BOOL	Read/Write
PLCDEBUG_POWERFLOW	VT_BOOL	Read/Write
PLCMODE_HALT	VT_BOOL	Read/Write
PLCMODE_ON	VT_BOOL	Read/Write
PLCMODE_RUN	VT_BOOL	Read/Write
PLCMODE_STOP	VT_BOOL	Read/Write

☑ Use absolute path for variable

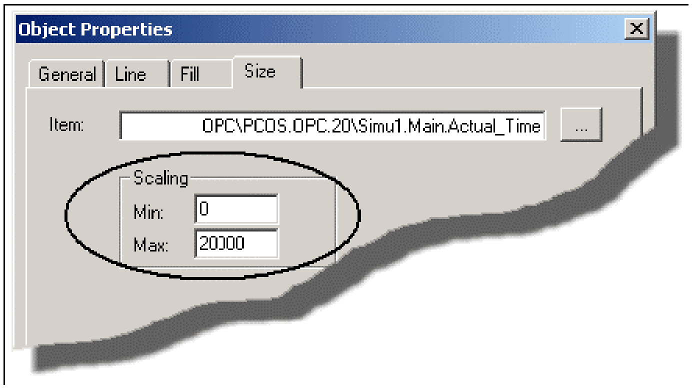
j. Mark the variable 'Actual_Time' (located in the subfolder 'Main') because the size of our rectangle should depend on this value.

k. Click 'OK' to close the 'Variable Browser'. The variable with its path is now entered in the 'Item' field of the 'Object Properties' dialog (see next figure).

l. Now we have to scale the size change, i.e. we must define the value range which has to be covered by the minimum and maximum rectangle size. For that purpose, the 'Size' page of the properties dialog provides the fields 'Min' and 'Max'.

As our 'Actual_Time' variable reaches from 0 to 20000 milliseconds fill in the fields as shown in the following figure.

Figure 86: Scaling the value range of the dynamic property 'Size'



Item: OPC\PCOS.OPC.20\Simu1.Main.Actual_Time

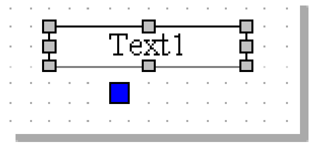
Min: 0

Max: 20000

m. Click 'OK' to close the 'Object Properties' dialog. The object is now displayed as short blue rectangle.

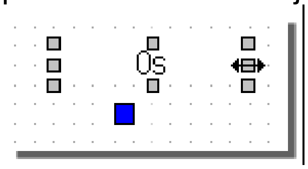
The rest of this step is "beautification": We will insert a symbolic scale ('0s' and '20s') as well as a static text below our TimeBar.

n. To insert a static text element first select the toolbar icon . Left click into the worksheet and drag the mouse diagonally to draw the static text object.



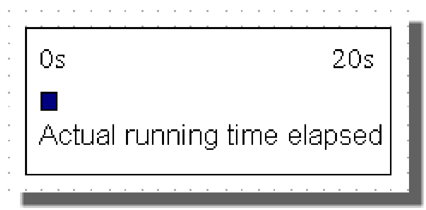
o. Double click on the object to open its properties dialog. On the dialog page 'Static Text' enter '0s' in the 'text' field. On the dialog page 'Line' check 'Transparent' to hide the object border. On the page 'Font' select 'Arial', 'Regular', '10' pt. Click 'OK' to confirm the settings

p. Resize the text object.



q. Duplicate the text object twice by Drag & Drop while holding the <Ctrl> key pressed (alternatively you can copy & paste the object twice). Change one string to '20s' and the other to 'Actual running time elapsed'. Move each changed object to its position above or below the TimeBar.

r. Insert a rectangle as frame, send it to the back and group all objects.



VISUALIZING THE 'VISU_MOTOR_START' CONTACT BY A LIBRARY PUSH BUTTON

STEP 4

We now want to visualize the contact which must be activated three times in order to start the motor. For that purpose we are going to use an object which is provided in the firmware library. We just have to insert the library object, scale it to the desired size and connect it to the OPC variable 'VISU_Motor_Start' – ready!

a. Click on the toolbar icon 'Library Object':

b. The appearing 'Libraries' dialog contains all objects provided by firmware or user libraries. Browse into the 'Buttons' branch and double click on the object 'Push_Button'.

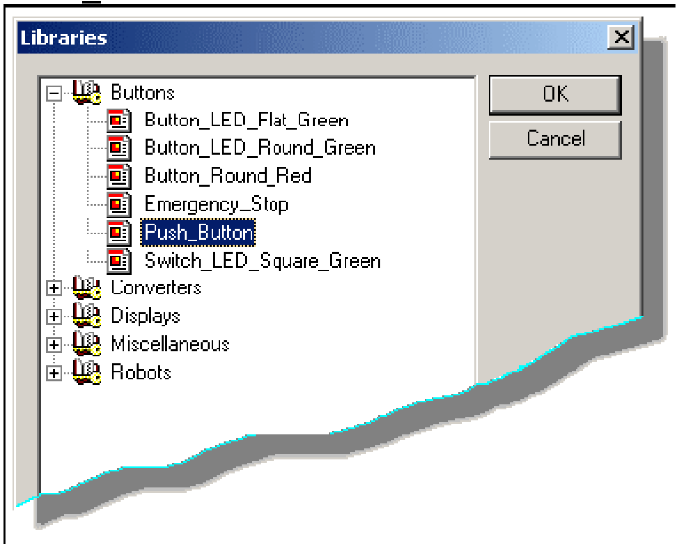
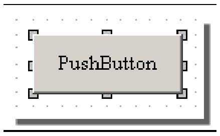
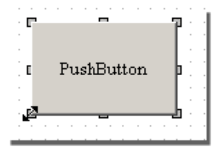


Figure 87: Dialog 'Libraries' for inserting library objects

The object is inserted into the worksheet:



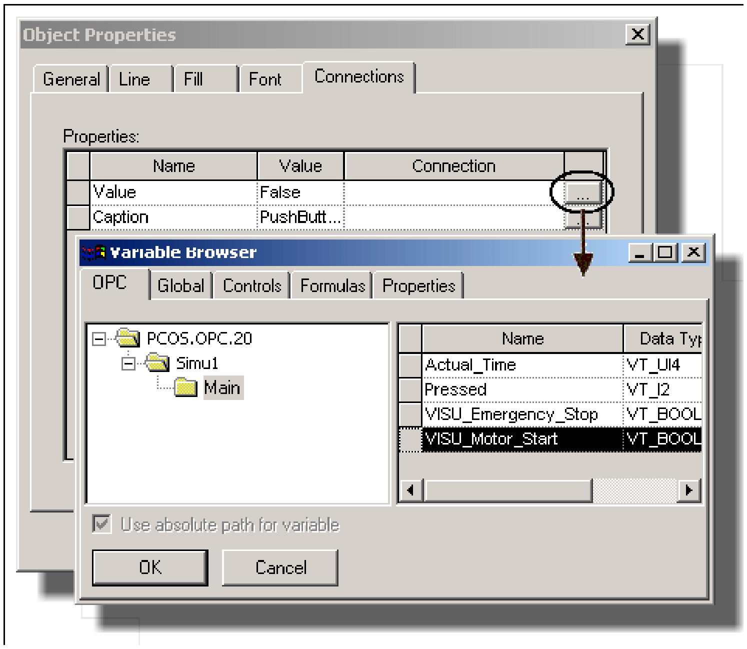
c. Resize it by placing the mouse pointer to an object handle and drag the mouse while keeping the mouse button pressed:



d. Double click on the object to open its properties dialog.

e. On the dialog page 'Connections' we connect the object to the OPC variable 'VISU_Motor_Start' which should be overwritten when pressing the button. For that purpose click on the browse button in the table row 'Value'.

In the appearing 'Variable Browser' stay on the page 'OPC', open the subfolder 'Main' in the resource branch 'Simu1' and mark the variable 'VISU_Motor_Start'.



Properties:

Name	Value	Connection
Value	False	
Caption	PushButt...	

Name	Data Typ
Actual_Time	VT_UI4
Pressed	VT_I2
VISU_Emergency_Stop	VT_BOOL
VISU_Motor_Start	VT_BOOL

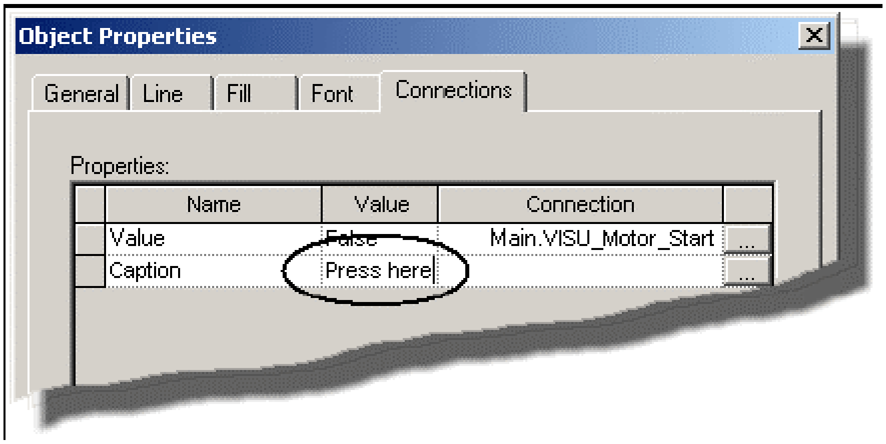
☑ Use absolute path for variable

Figure 88: Connecting the library push button to the OPC variable 'VISU_Motor_Start'

Confirm the assignment by clicking on 'OK' in the 'Variable Browser'.

f. Now change the caption of the button which is only visible during runtime. To do this, overwrite the default text 'PushButton' with 'Press here' as shown:

Figure 89: Changing the runtime caption of the push button



Name	Value	Connection
Value	False	Main.VISU_Motor_Start
Caption	Press here	

g. To change the object appearance open the dialog page 'Font'. Select the same font settings as for our previously inserted static texts: Arial, Regular, 10pt.

h. Finally we need the descriptive text 'Press 3 times to start the motor':

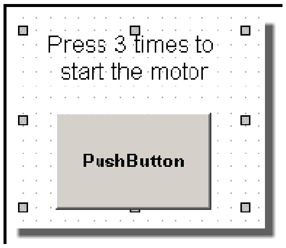
Insert a new Static Text object and open its properties dialog. Enter the desired string on the dialog page 'Static Text'. Activate the checkbox 'Multiple Lines'.

On the page 'Line' check 'Transparent' and on the page 'Fill' activate 'Transparent Fill'. Finally adjust the font settings to Arial.

'Click 'OK' to confirm the Static Text settings.

i. Resize the text object and move it to the desired position.

Now the push button with its explaining text is complete. If desired mark both objects and group them:



STEP 5

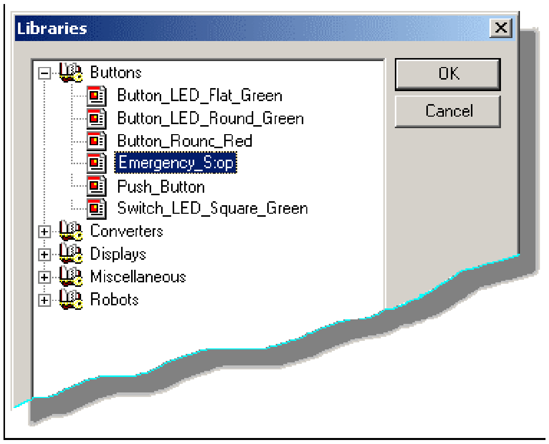
VISUALIZING THE 'VISU_EMERGENCY_STOP' CONTACT BY A LIBRARY EMERGENCY SWITCH

We now want to visualize the emergency stop switch, i.e. the contact which stops the motor when activated. Again we use a library object provided in the firmware library. As in step 4, we insert the object, scale it to the desired size and connect it to the OPC variable 'VISU_Emergency_Stop'.

a. Click on the toolbar icon 'Library Object':

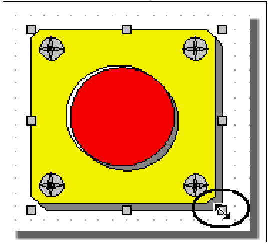
b. In the appearing 'Libraries' dialog double click on 'Emergency_Stop' in the 'Buttons' folder.

Figure 90: Dialog 'Libraries' for inserting library objects



The object is inserted into the worksheet.

c. Resize it by placing the mouse pointer to an object handle in a corner and drag the mouse while keeping the mouse button pressed. To resize the object proportionally press the <Shift> key while dragging the mouse.

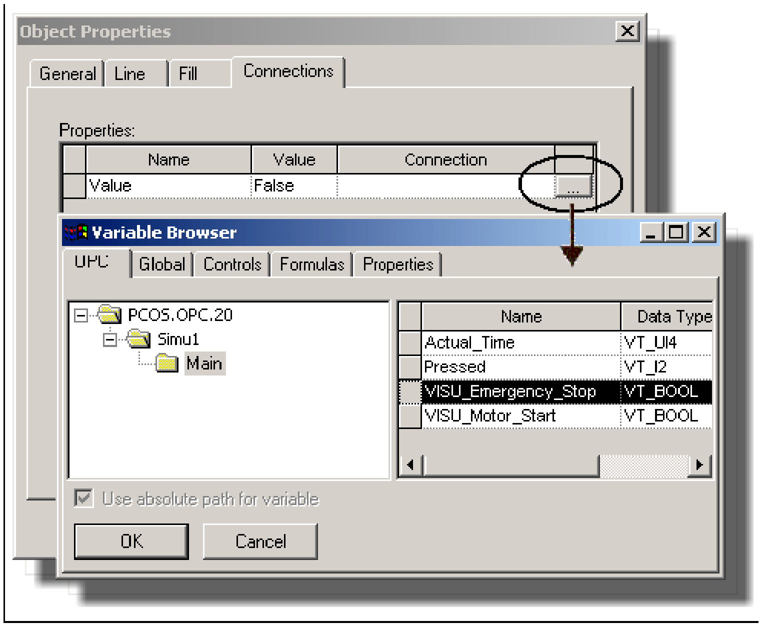


d. Double click on the object to open its properties dialog.

e. On the dialog page 'Connections' we connect the object to the OPC variable 'VISU_Motor_Start' which should be overwritten when pressing the button. For that purpose click on the browse button in the table row 'Value'.

In the appearing 'Variable Browser' stay on the page 'OPC', open the subfolder 'Main' in the resource branch 'Simu1' and mark the variable 'VISU_Emergeny_Stop'.

Figure 91: Connecting the library emergency stop switch to the OPC variable 'VISU_Emergency_Stop'



Object Properties

General | Line | Fill | Connections

Properties:

Name	Value	Connection
Value	False	...

Variable Browser

OPC | Global | Controls | Formulas | Properties

- PCOS.OPC.20
  - Simu1
    - Main

Name	Data Type
Actual_Time	VT_UI4
Pressed	VT_I2
VISU_Emergency_Stop	VT_BOOL
VISU_Motor_Start	VT_BOOL

☑ Use absolute path for variable

OK | Cancel

Click 'OK' in the 'Variable Browser' and in the 'Object Properties' dialog.

Now the emergency switch is complete.

STEP 6

VISUALIZING THE VARIABLE 'PRESSED' BY A LIBRARY LCD ELEMENT

In our example motor control the variable 'Pressed' counts how often the contact 'Motor_Start' has been energized.

If 'Pressed' = 3 the motor starts running and the counter is automatically reset. In the visualization we want to use a 7 segment LCD element provided in a firmware library. For that purpose we have to connect the LCD element to the OPC variable 'Pressed'.

a. Click on the toolbar icon 'Library Object':

b. In the appearing 'Libraries' dialog browse into the 'Displays' folder and double click on 'Display_LCD_1'. The object is inserted into the worksheet.

c. Resize it by placing the mouse pointer to an object handle in a corner and drag the mouse while keeping the mouse button pressed. To resize the object proportionally press the <Shift> key while dragging the mouse.

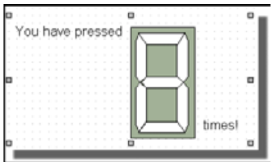
d. Double click on the object to open its properties dialog.

e. On the dialog page 'Connections' we connect the object to the OPC variable it should display. For that purpose click on the browse button in the table row 'Value'. In the appearing 'Variable Browser' stay on the page 'OPC', open the subfolder 'Main' in the resource branch 'Simu1' and mark the variable 'Pressed'.

Click 'OK' in the 'Variable Browser' and in the 'Object Properties' dialog.

f. Finally we need the descriptive texts 'You have pressed' and 'times'. For example you can create them by copying the descriptive text twice from the push button and change its contents accordingly. Arrange the copied and changed objects around the LCD element as shown below.

Now the LCD element is complete. If desired, group the objects.



STEP 7 VISUALIZING THE 'MOTOR' COIL BY A LIBRARY LED

In our sample project we want to visualize the running motor in two ways:

- By a green LED provided by the firmware library. This LED is connected to the coil 'Motor' and lights up if the coil is energized, i.e. if the motor is running.

- By a self-designed rotating motor symbol connected to a visualization global variable which is processed in a script. Please refer to page 86.

To insert and connect the LED proceed as follows:

a. Click on the toolbar icon 'Library Object':

b. In the appearing 'Libraries' dialog browse into the 'Miscellaneous' folder and double click on 'LED_Green'. The object is inserted into the worksheet.

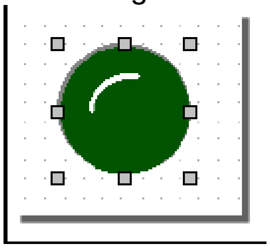
c. Resize it by placing the mouse pointer to an object handle in a corner and drag the mouse while keeping the mouse button pressed. To resize the object proportionally press the <Shift> key while dragging the mouse.

d. Double click on the object to open its properties dialog.

e. On the dialog page 'Connections' we connect the object to the OPC variable it should represent. For that purpose click on the browse button in the table row 'Value'. In the appearing 'Variable Browser' stay on the page 'OPC', open the resource branch 'Simu1' and mark the variable 'Motor'.

Click 'OK' in the 'Variable Browser' and in the 'Object Properties' dialog.

Now the green LED is complete.



VISUALIZING THE RUNNING MOTOR USING A SELF DESIGNED OBJECT AND A SCRIPT

Finally, we want to visualize the running motor in a more complex way: We are going to

- design a motor symbol which basically consists of a polygon with the dynamic property 'Rotation'.

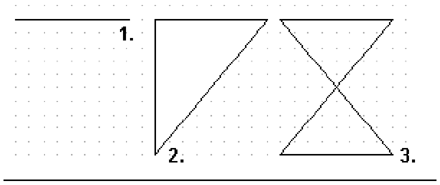
- connect the dynamic property 'Rotation' to a visualization global variable 'rotateMe' which is calculated by a script.

- write a script which calculates the variable 'rotateMe'.

Designing the object

a. Left click into the design window to activate the design toolbar. Click on the icon 'Polygon'.

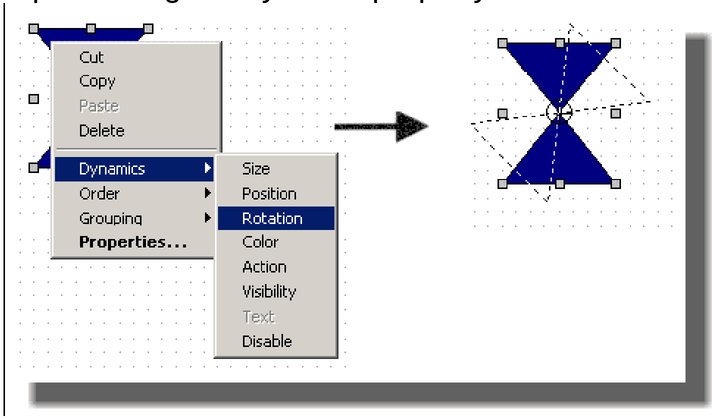
b. In the worksheet draw the shown figure by clicking two times to set the corners (1.) and (2.). At position (3.) double click to finish the polygon.



c. Double click on the polygon to open its 'Object Properties' dialog. In the tab 'Fill' select dark blue as 'Foreground Color'. Click 'OK'.

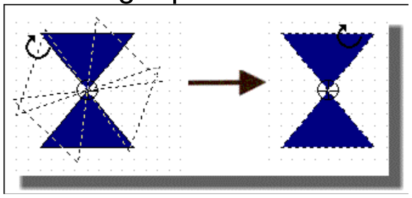
d. Right click in the colored polygon and assign the dynamic property 'Rotation'. A dashed polygon is added to the object, representing the dynamic property.

Figure 92: Assigning the dynamic property 'Rotation' to a polygon



e. Define the start and end position of the rotation. For that purpose left click on the dashed polygon. The mouse cursor changes its shape into a symbolic circular arrow.
Place the mouse cursor on the dashed frame. Keep the mouse button pressed while dragging the dashed polygon to its target position.

Figure 93: Defining the 'Rotation' of a polygon



Since we want the polygon to rotate all around, the dashed object must be congruent with the solid polygon but rotated by 180° as shown above.

f. Click elsewhere in the design window to deselect the object.

Declaring visualization global variables for the script

Before we can complete the rotation of the polygon by assigning it to a variable, we have to declare two visualization global variables.

Why declare visualization global variables?
This is necessary, because the rotation of our polygon is calculated by a global scriptAs we can not directly process our Boolean coil 'Motor' in a script, we have to use two global variables instead:

- 'motorIsRunning' is connected to the OPC variable 'Motor' and therefore represents the state of our coil.
- rotateMe is calculated by a script depending on the value of 'motorIsRunning'. rotateMe is assigned to our polygon and causes its rotation.

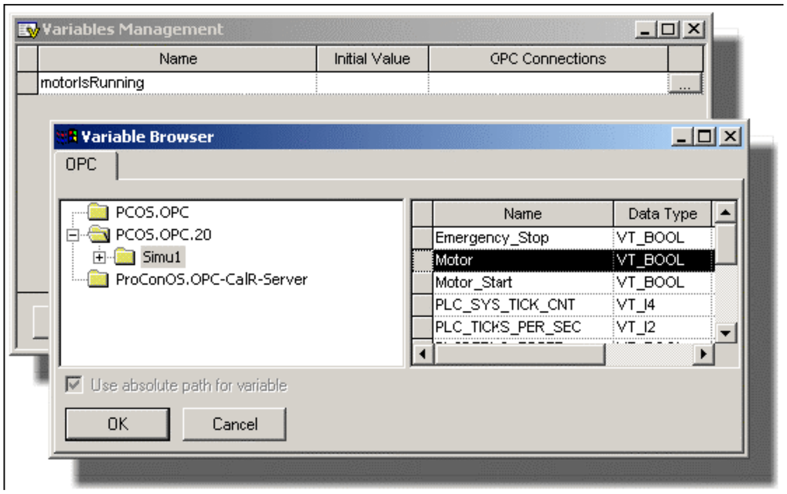
Declare the variables as follows:

a. Open the dialog 'Variables Management':

b. In the dialog click on 'Insert'. A new row is inserted with the default name 'Var1'. Click into the 'Name' field and overwrite this default entry with 'motorIsRunning'.

Now we have to connect the new variable to the coil 'Motor'. For that purpose, click on the browse button beneath the 'OPC Connections' field to call the 'Variables Browser'.

Browse for the OPC resource 'Simu1' and mark the variable 'Motor' as shown below.



Variables Management

Name	Initial Value	OPC Connections
motorIsRunning		

Variable Browser

OPC

- PCOS.OPC
- PCOS.OPC.20
  - Simu1
- ProConOS.OPC-CalR-Server

Name	Data Type
Emergency_Stop	VT_BOOL
Motor	VT_BOOL
Motor_Start	VT_BOOL
PLC_SYS_TICK_CNT	VT_I4
PLC_TICKS_PER_SEC	VT_I2

☑ Use absolute path for variable

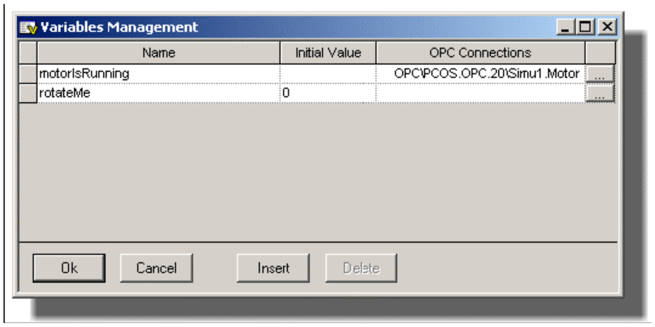
Figure 94: Connecting a new declared visualization global variable to an OPC variable

Confirm the 'Variables Browser' with 'OK'. The variable is entered in the variables table (see figure on next page).

c. In the dialog 'Variables Management insert a second variable and change the default name to 'rotateMe'. Since this is the variable which will be calculated by a script and connected to our rotating polygon we do not assign any OPC variable.

However, it is necessary to define an initial value: Click into the table field and enter '0'.

Your variables table now looks as follows.



Name	Initial Value	OPC Connections
motorIsRunning		OPC\PCOS.OPC.20\Simu1.Motor
rotateMe	0	

Figure 95:
Global variable with OPC connection

Developing a global script

The declared visualization global variables can now be used in scripts. So, we are going to write a global script which is executed at the beginning of each visualization cycle.

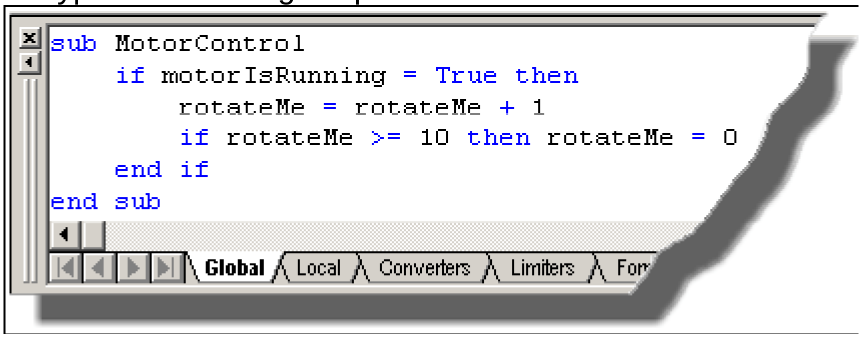
In our script the value of the variable 'rotateMe' is calculated depending on the value of the variable 'motorIsRunning' which is connected to the OPC variable 'Motor'.

Proceed as follows:

a. In the script window click on the tab 'Global'.

b. Click into the global script worksheet to set a text cursor.

c. Type the following script:



```
sub MotorControl
    if motorIsRunning = True then
        rotateMe = rotateMe + 1
        if rotateMe >= 10 then rotateMe = 0
    end if
end sub
```

Figure 96:
Global script calculating the variable for the polygon rotation

Now that we have edited the script, only one step is left to complete the rotating polygon: It has to be connected to the variable 'rotateMe'.

Connecting the dynamic polygon property to a variable

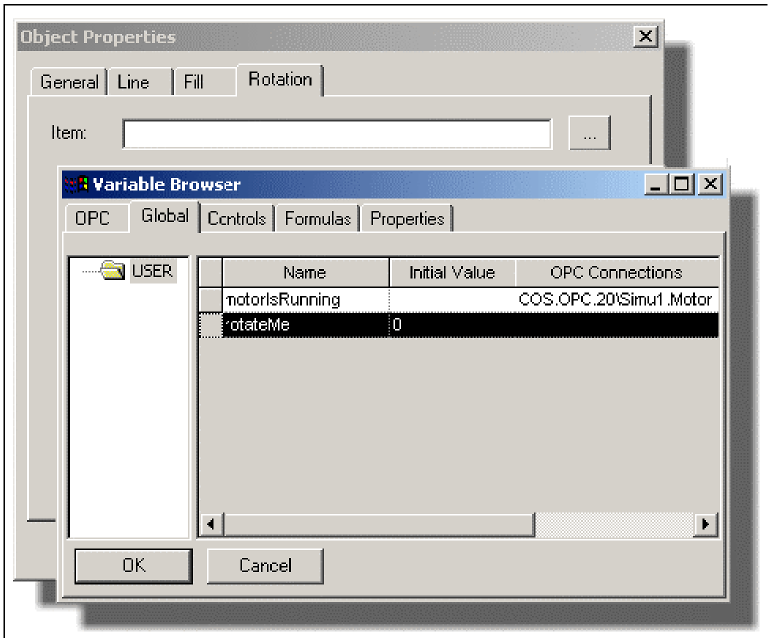
a. In the design window right click on the polygon and select the context menu item 'Properties...'. The 'Object Properties' dialog appears.

b. Open the dialog page 'Rotation'. Click on the browse button beneath the 'Item' field to select the variable on which the rotation shall depend.

Since we want to assign a visualization global variable to the property, open the browser page 'Global'.

Mark the variable 'rotateMe'.

Figure 97: Assigning the polygon rotation to a visualization global variable

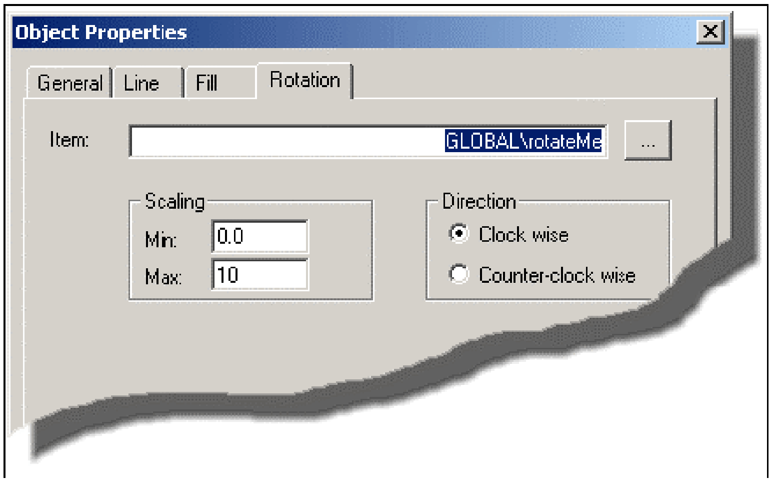


Name	Initial Value	OPC Connections
notorIsRunning		COS.OPC.20\Simu1.Motor
otateMe	0	

c. Click 'OK' in the 'Variable Browser'. The variable is entered in the 'Object Properties' dialog (see figure below).

d. Now we have to scale the rotational motion. According to our script, the global variable 'rotateMe' which controls the rotation can assume values between 0 and 10. Thus, we define the rotation scaling accordingly:

Figure 98: Scaling the polygon rotation


Object Properties

Item: GLOBAL\rotateMe

Scaling
Min: 0.0
Max: 10

Direction
● Clock wise
○ Counter-clock wise

e. Confirm the properties dialog.

Final beautification

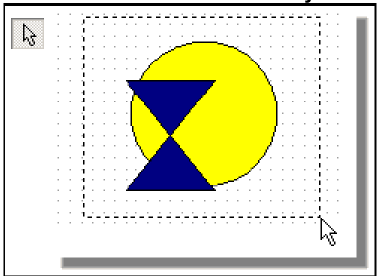
The polygon is now "ready to rotate"... Finally we will do some "cosmetics" again: Completing our motor symbol, arranging the individual objects and groups and framing them by rectangles:

a. Draw a circle around the polygon and open its properties dialog. Select yellow as 'Foreground Color' and confirm the dialog.

b. Right click on the circle and select 'Order > Send to Back' in the context menu.

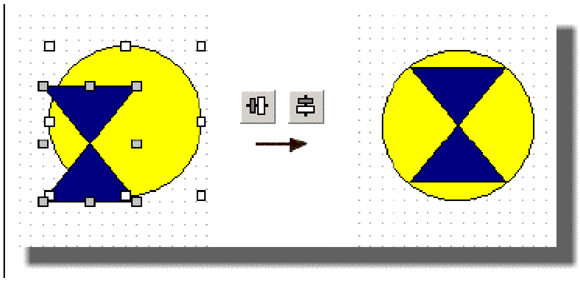
c. Mark both, the polygon and the circle by dragging the mouse around the objects:

Figure 99: Marking two objects in mark mode



d. Align them to the center and middle:

Figure 100: Aligning objects

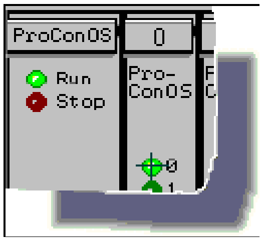


e. Group them by right clicking while both are still marked and selecting the context menu item 'Grouping > Group'.

f. Arrange all objects in the worksheet as shown below.

g. Insert rectangles, fill them and send them to the background. Use these rectangles to visually frame object groups as shown in the figure on the next page.

SWITCHING THE VISUALIZATION TO RUNTIME



Prior to switching the visualization to runtime mode, make sure that the PLC (i.e. the simulation) is still running correctly. For that purpose click on the 'Demo IO' icon in the Windows taskbar. The 'Run' LED should be on.

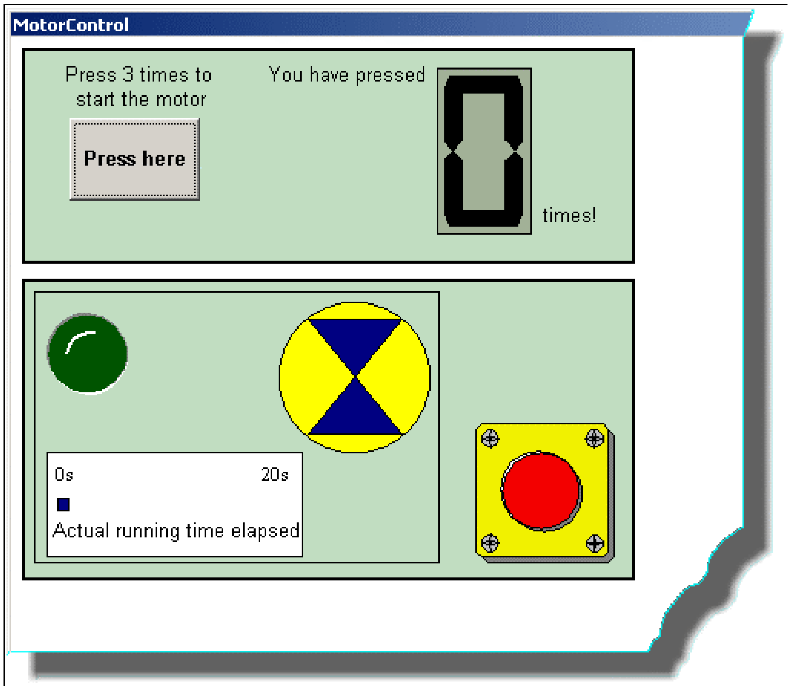
If this is not the case, restart the programming system and start the PLC via the resource control dialog as described starting at page 73.

Switching to runtime mode

a. Click on the 'Runtime' toolbar icon.

The screen appears as defined in its runtime settings: A modeless dialog with the size 800 x 600 pixel.

Figure 101: Finished visualization screen in runtime mode



b. Press the push button three times. Observe the display. After pressing three times, the display should be reset to 0, the green LED should be illuminated and the motor symbol should start rotating. The time bar increases its horizontal size.

After 20 seconds the motor symbol stops spinning, the LED extinguishes and the time bar is reset.

c. Start the motor again by pressing the push button three times. Then actuate the emergency switch (within the 20s running time period!).

Observe the result: The motor should stop immediately and all displaying objects should be reset.

Note, that the emergency stop works as a switch. This means, that you have to release the switch by pressing it again.

Correcting errors in the visualization screen

If any element is not reacting as desired, proceed as follows:

a. Observe the message window for any messages, i.e. errors.

b. Switch the visualization offline (i.e. back to design mode):

▮

c. Check the properties of the suspected object, i.e. the assigned (OPC) variable.

d. Correct any errors, save the project and switch to runtime mode again.

Changing the visualization cycle time and OPC update rate

a. Switch the visualization offline (i.e. back to design mode):
▮

b. Select 'Extras > Options'. In the appearing dialog open the page 'Runtime'.

c. Change the time values for the 'OPC connectivity' and the 'Cycle time'. Confirm the dialog.

d. Switch the visualization screen online again.

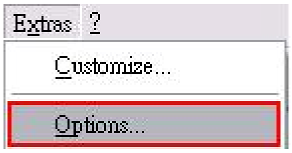
e. Start the motor and observe the effect of the changed time settings.

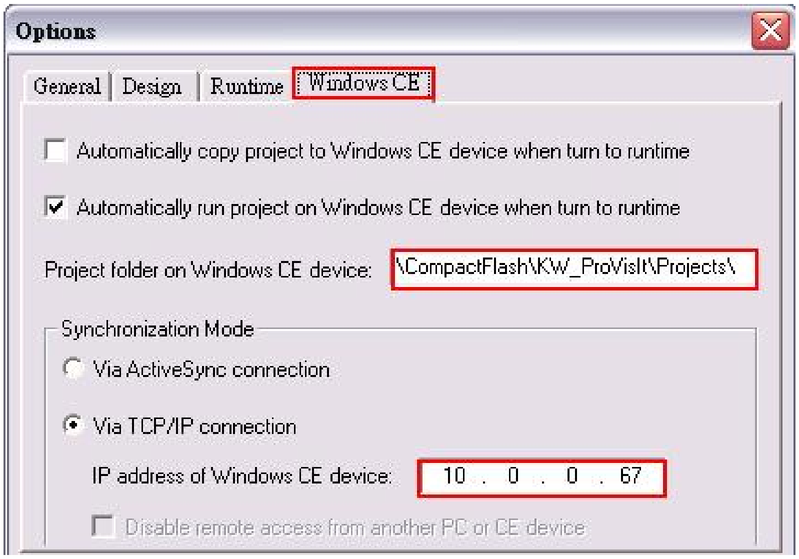
If desired you can repeat these steps with different time settings.

Downloading Project to KinCon-8000

For download ProVisIT project to KinCon-8000, please start ProVisIt(RT) by executing KWBoot.exe. After that, follow the procedure below to download the project to KinCon-8000.

Step1: Set up download target. Choose ’**Extras**’ in **Menu** and then click ‘**Option**’ into ’**Windows CE** ’ page. Enter \CompactFlash\KW_ProVisIt\Projects\ in ’**Project Folder on Windows CE device**’ and IP address in ’**IP address in Windows CE device**’





Options

General | Design | Runtime | Windows CE

□ Automatically copy project to Windows CE device when turn to runtime

☑ Automatically run project on Windows CE device when turn to runtime

Project folder on Windows CE device: \CompactFlash\KW_ProVisIt\Projects\

Synchronization Mode

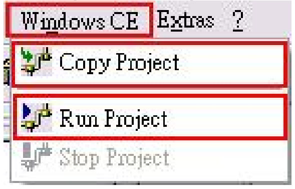
○ Via ActiveSync connection

● Via TCP/IP connection

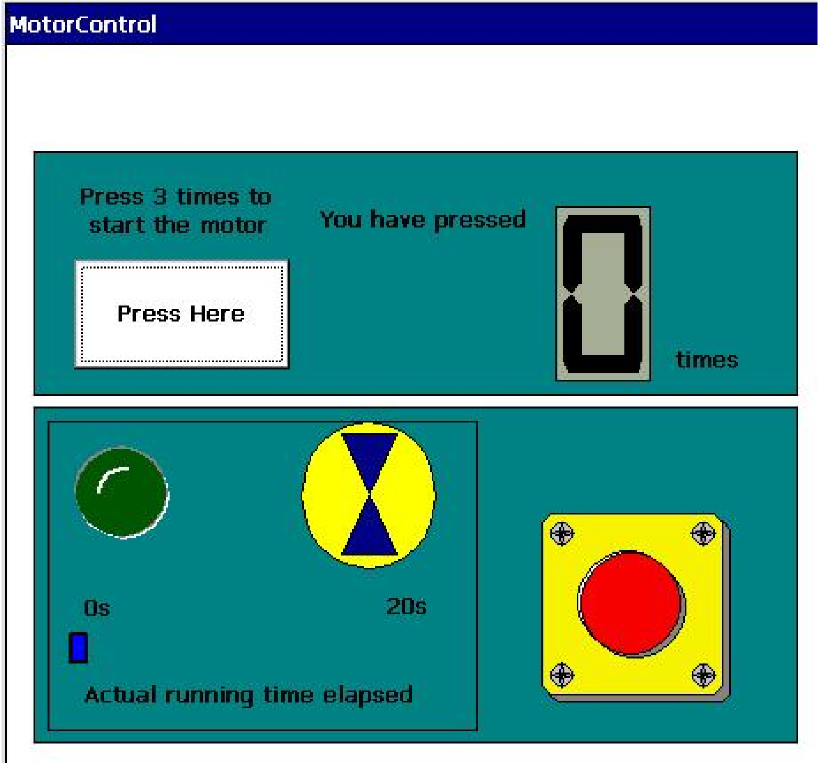
IP address of Windows CE device: 10 . 0 . 0 . 67

□ Disable remote access from another PC or CE device

Step2: Choose '**WindowsCE**' in **Menu**. Press '**Copy Project**' and '**Run Project**' sequentially.



Step3: After that, you can see the runtime mode on KinCon-8000



APPENDIX

IEC PROJECT COMPONENTS IN THE PROGRAMMING SYSTEM

Programming systems that conform to IEC 61131-3 contain the following component elements:

- Configurations
- Resources
- Tasks

These will be displayed if you select the 'Hardware' tab of the project tree.

Configurations can be compared to a programmable controller system such as a rack.

Resources can be compared to a CPU that can be inserted in the rack. In a resource, global variables can be declared which are only valid within this resource. In a resource, one or several tasks can be executed.

In general, **tasks** determine the time scheduling of the associated programs. This means that programs have to be associated to tasks. The settings of the task determine the time scheduling. The system provides one cyclic task to be assigned to your program.

PROGRAM ORGANIZATION UNITS (POUS)

Program organization units (POUs) are the language building blocks of an IEC 61131-3 control program. They are small, independent software units containing the program code. The name of a POU must be unique within the project.

In IEC 61131-3 three types of POUs are supported:

- Functions
- Function blocks
- Programs

Functions are POUs with multiple input parameters and exactly one output parameter. Calling a function with the same values returns always the same result. Return values can be single data types. Within a function it is possible to call another function but not a function block or a program. Recursive calls are not allowed.

IEC 61131-3 lists different types of standard functions:

- Type conversion functions, such as ANY_INT_TO_REAL
- Numerical functions, such as ABS and LOG
- Standard arithmetic functions, such as ADD and MUL
- Bit-string functions, such as AND and SHL
- Selection and comparison functions, such as SEL and GE
- Character string functions, such as RIGHT and INSERT
- Functions of time data types, such as SUB with the data type TIME ('SUB_T_T')

Function blocks are POUs with multiple input/output parameters and internal memory. The value returned by a function block depends on the value of its internal memory. Within a function block it is possible to call another function block or functions. Recursive calls are not allowed.

IEC 61131-3 lists different types of standard function blocks:

- Edge detection function blocks, such as R_TRIG and F_TRIG
- Counters, such as CTU and CTD
- Timer function blocks, such as TON and TOF
- Bistable function blocks SR and RS

Programs are POUs that contain a logical combination of functions and function blocks according to the needs of the controller process. The behavior and the use of programs are similar to function blocks. Programs have an internal memory. Programs must be associated to tasks. Within a Program it is possible to call functions and function blocks. Recursive calls are not allowed.

INSTANTIATION OF POUS AND FUNCTION BLOCKS

According to IEC 61131-3 the code of a FB POU (Function Block) can be reused in a project by calling the FB in another POU using an unique name. This is known as "Instantiation". By calling the FB instance the FB code must be defined only once. If the FB instance is called, the internal memory of the FB is allocated to the called instance. This allows the use of different memory areas.

Each instance has an associated identifier the "instance name" and contains the input and output parameters and the internal memory for the POU or FB. A FB can be instantiated in another FB or in a program. The instance name of an FB has to be declared in the VAR declaration of the program or FB where it is going to be used.

VARIABLES AND DATA TYPES

Another powerful feature of IEC 61131 is the use of variables rather than the direct addressing scheme of traditional PLC systems. This increases flexibility and broadens the scope of functionality that can be performed in the programs.

VARIABLE TYPES

Variables must be declared first in order to be used in the logic. When inserting a variable into a worksheet, you can declare two variable types:

1. Local variables
2. Global variables

A **local variable** is only used in one POU, whereas a **global variable** can be used in every POU of the corresponding project.

The **local variable** is declared in the local variables worksheet of the POU in which it is used.

The **global variable** has to be declared as 'VAR_GLOBAL' in the global variables' declaration of a resource and as 'VAR_EXTERNAL' in each POU in which it is used.

The programming system provides for automatic declaration of variables and their properties during program creation as the I/O address/logical name are assigned. Variables can also be manually declared in the variables worksheet.

VARIABLES ADDRESSES

You can directly address your variables using the 'I/O address' input field.

In accordance to IEC 61131, a location declaration consists of the keyword AT, the percent sign '%', a location prefix, a size prefix and the name of the logical address.
In the programming system it is not necessary to enter the keyword AT. However, the sign '%' must be entered.
Example of a possible variable address: '%QX0.0'.

The following table shows the location and size prefixes for located variables:

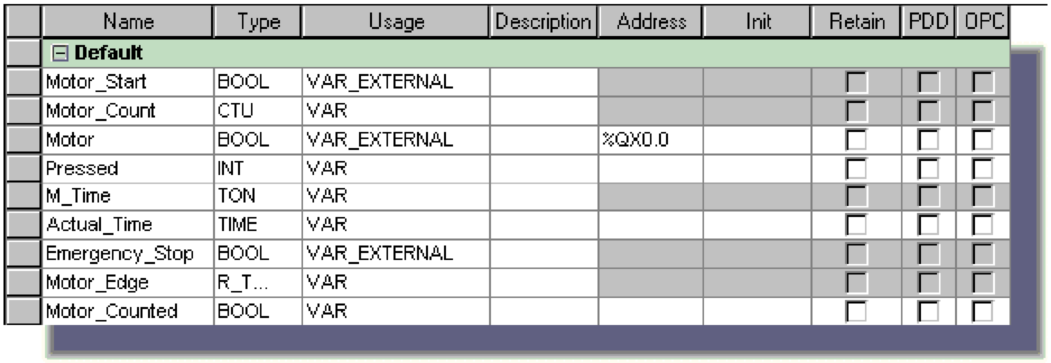
Location prefix	**Description**
I	Physical input
Q	Physical output
M	Physical address in the PLC memory
Size prefix	**Description**
X	Single bit size (only with data type BOOL)
None	Single bit size
B	Byte size (8 bits)
W	Word size (16 bits)
D	Double word size (32 bits)

When declaring a variable in the system the 'Variables Properties' dialog is automatically opened. Using this dialog the declaration of the current variable is inserted or changed automatically in the corresponding variables' worksheet.
Local variables are inserted in the variables worksheet of the corresponding POU in the project tree, global variables in the global variables' worksheet in the subtree 'Physical Hardware'.

The 'Variables Properties' dialog can also be called by clicking on 'Properties'.

If you want to have a look at the declarations, click on the 'Variables Worksheet' icon in the toolbar

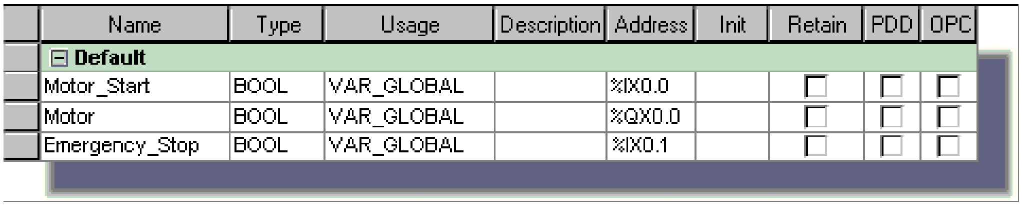
to open the variables grid worksheet of the POU (**local variables grid worksheet**):



Name	Type	Usage	Description	Address	Init	Retain	PDD	OPC
⊟ **Default**								
Motor_Start	BOOL	VAR_EXTERNAL				□	□	□
Motor_Count	CTU	VAR				□	□	□
Motor	BOOL	VAR_EXTERNAL		%QX0.0		□	□	□
Pressed	INT	VAR				□	□	□
M_Time	TON	VAR				□	□	□
Actual_Time	TIME	VAR				□	□	□
Emergency_Stop	BOOL	VAR_EXTERNAL				□	□	□
Motor_Edge	R_T...	VAR				□	□	□
Motor_Counted	BOOL	VAR				□	□	□

Figure 102: Local variables grid worksheet

or double click on 'Global Variables' in the project tree to open the **global variables grid worksheet**:



Name	Type	Usage	Description	Address	Init	Retain	PDD	OPC
⊟ **Default**								
Motor_Start	BOOL	VAR_GLOBAL		%IX0.0		□	□	□
Motor	BOOL	VAR_GLOBAL		%QX0.0		□	□	□
Emergency_Stop	BOOL	VAR_GLOBAL		%IX0.1		□	□	□

Figure 103: Global variables grid worksheet

DATA TYPES

Data types determine the kind of value the variable can have. Data types define the initial value, range of possible values and the number of bits.

IEC 61131-3 distinguishes three kinds of data types:

□ **Elementary Data Types:**
The value ranges and size of elementary data types described in IEC 61131-3 are shown in the following table:

Data Type	Description	Size	Range
BOOL	Boolean	1	0...1
SINT	Short integer	8	-128...127
INT	Integer	16	-32768 ... 0 ... 32767
DINT	Double integer	32	-2,147,483,648 up to 2.147.483.647
USINT	Unsigned short integer	8	0 up to 255
UINT	Unsigned integer	16	0 up to 65535
UDINT	Unsigned double integer	32	0 up to 4.294.967.295
REAL	Real numbers	32	+/-1.18 x 10^-38 up to +/-3.40x10^38
TIME	Duration	32	+# 4.294.976.295 ms up to +# 4.294.976.295 s
BYTE	Bit string of length	8	0x00...0xFF
STRING	Sequence of characters	80	
WORD	Bit string of length	16	0x0000 ... 0xFFFF
DWORD	Bit string of length	32	0x00000000 ... 0xFFFFFFFF

□ **Generic Data Types:**
Generic data types include groups of elementary data types. They are called e.g. ANY_BIT or ANY_INT.

□ **User Defined Data Types:**
User Defined data types are groups of different data types, assembled for a specific purpose, and defined as ARRAYs and STRUCTures.

Driver Parameters

For Embedded I-8K/I87K Modules

Driver name: The driver name of the KinCon I/O-Driver is “WinCon8x”.
Parameter 1: The slot number of the KinCon.
Parameter 2: The module ID of this slot.
Parameter 3: The timeout value for this module.
Parameter 4: Some flags used for certain modules(e.g. i-8017H) in order to enable different settings.

The KinCon K-8745 has 7 slots running from 1 to 7 and the KinCon K-83445 has 3 slots running from 1 to 3. A value in this range must be entered as parameter 1. Right now, only those modules which are on a main unit and therefore have a slot number can be used.

Currently several KinCon modules of the I-8000 and I-87000 series are supported by the ProConOS. “WinCon8x” IO-Driver. The following list describes these KinCon modules briefly.

The “**Module ID**” is the ID to enter as parameter 2 of the drivers setting. The “**Address width**” shows the allowed entries (in bytes) for the width of the I/O address in the I/O Group.

The “**Address value type**” explains how these address are interpreted, i.e. which data types will be used for that address.

The “**Address value**” describes what kind of data will be read/written.

Module name	Module ID	Address width	Address value type	Address value	Flag (value)/ Gain
8017H, 8017HS	8017	4,8,12,,32	REAL (4 bytes)	Float values of voltage / current corresponding to the gain set.	0 +/-10V 2 +/-5V 4 +/-2.5V 6 +/-1.25V 8 20mA
8024	8024	4, 8, 12, 16	REAL (4 bytes)	Float values of voltage in the range -10V to +10V.	-
8037	8037	1, 2	WORD (2 bytes)	Each bit of the output WORD represents the output of the corresponding channel.	-
8040	8040	1, 2, 3, 4	DWORD(4 bytes)	Each bit of the input DWORD represents the input of the corresponding channel.	-
8041	8041	1, 2, 3, 4	DWORD(4 bytes)	Each bit of the output DWORD represents the output of the corresponding channel.	-
8042	8042	1, 2,	WORD (2 bytes)	Each bit of the input/output WORD represents the input/output of the corresponding channel.	-
8050	8050	1, 2	WORD (2 bytes)	Each bit of the input/output WORD represents the input/output of the corresponding channel.	-
8051	8051	1, 2	WORD (2 bytes)	Each bit of the input WORD represents the input of the corresponding channel.	-
8052	8052	1	BYTE (1 byte)	Each bit of the input BYTE represents the input of the	-

				corresponding channel.	
8053	8053	1, 2	WORD (2 bytes)	Each bit of the input WORD represents the input of the corresponding channel.	-
8054	8054	1	BYTE (1 byte)	Each bit of the input/output BYTE represents the input/output of the corresponding channel.	-
8055	8055	1	BYTE (1 byte)	Each bit of the input/output BYTE represents the input/output of the corresponding channel.	-
8056	8056	1, 2	WORD (2 bytes)	Each bit of the output WORD represents the output of the corresponding channel.	-
8057	8057	1, 2	WORD (2 bytes)	Each bit of the output WORD represents the output of the corresponding channel.	-
8058	8058	1	BYTE (1 byte)	Each bit of the input BYTE represents the input of the corresponding channel.	-
8060	8060	1	BYTE (1 byte)	Each bit of the output BYTE represents the output of the corresponding channel.	-
8063	8063	1	BYTE (1 byte)	Each bit of the input/output BYTE represents the input/output of the corresponding channel.	-
8064	8064	1	BYTE (1 byte)	Each bit of the output BYTE represents the output of the corresponding channel.	-
8065	8065	1	BYTE (1 byte)	Each bit of the output BYTE represents the output of the corresponding channel.	-
8066	8066	1	BYTE (1 byte)	Each bit of the output BYTE represents the output of the corresponding channel.	-
8068	8068	1	BYTE (1 byte)	Each bit of the output BYTE represents the output of the corresponding channel.	-
8069	8069	1	BYTE (1 byte)	Each bit of the output BYTE represents the output of the corresponding channel.	-
8077	8077	1	BYTE (1 byte)	Each bit of the input/output BYTE represents the input/output of the corresponding channel.	-
87013	17013	4, 8, 12, 16	REAL (4 bytes)	Float values of temperatures (in degree Celsius)	-
87017	17017	4,8,12,,32	REAL (4 bytes)	Float values of voltage/ current	-
87018	17018	4,8,12,,32	REAL (4 bytes)	Float values of voltage/	-

				current	
87019	17019	4,8,12,,32	REAL (4 bytes)	Float values of voltage/ current	-
87022	17022	4, 8	REAL (4 bytes)	Float values of voltage in the range -10V to +10V.	-
87024	17024	4, 8, 12, 16	REAL (4 bytes)	Float values of voltage in the range -10V to +10V.	-
87026	17026	4, 8	REAL (4 bytes)	Float values of voltage in the range -10V to +10V.	-
87040	17040	1, 2, 3, 4	DWORD(4 bytes)	Each bit of the input DWORD represents the input of the corresponding channel.	-
87041	17041	1, 2, 3, 4	DWORD(4 bytes)	Each bit of the output DWORD represents the output of the corresponding channel.	-
87051	17051	1, 2	WORD (2 bytes)	Each bit of the input WORD represents the input of the corresponding channel.	-
87052	17052	1	BYTE (1 byte)	Each bit of the input BYTE represents the input of the corresponding channel.	-
87053	17053	1, 2	WORD (2 bytes)	Each bit of the input WORD represents the input of the corresponding channel.	-
87054	17054	1	BYTE (1 byte)	Each bit of the input/output BYTE represents the input/output of the corresponding channel.	-
87055	17055	1	BYTE (1 byte)	Each bit of the input/output BYTE represents the input/output of the corresponding channel.	-
87057	17057	1, 2	WORD (2 bytes)	Each bit of the output WORD represents the output of the corresponding channel.	-
87058	17058	1	BYTE (1 byte)	Each bit of the input BYTE represents the input of the corresponding channel.	-
87063	17063	1	BYTE (1 byte)	Each bit of the input/output BYTE represents the input/output of the corresponding channel.	-
87064	17064	1	BYTE (1 byte)	Each bit of the output BYTE represents the output of the corresponding channel.	-
87065	17065	1	BYTE (1 byte)	Each bit of the output BYTE represents the output of the corresponding channel.	-
87066	17066	1	BYTE (1 byte)	Each bit of the output BYTE represents the output of the corresponding channel.	-
87068	17068	1	BYTE (1 byte)	Each bit of the output BYTE	-

				represents the output of the corresponding channel.	
87069	17069	1	BYTE (1 byte)	Each bit of the output BYTE represents the output of the corresponding channel.	-

The **'Timeout'** values to be entered in parameter 3 of the driver parameters of a KinCon-8000 module runs from 0 to 65535. "0" means no timeout is set (for this module). Otherwise it specifies the value in milliseconds used for the following procedure:

If the KinCon-8000 module is an input module and a task defined by the user needs values from this module and if current values are available these values will be transferred to the input address and the driver will ask for new values starting a new timeout period.

If there are no current values available the previously retrieved data will be used instead and the amount of time (from the last start period to the actual time) will be measured. If this time exceeds the user defined timeout value of this module an I/O driver error will be reported via ProConOS to MULTIPROG (which can handle this error by defining a system task). (Note: A timeout error of a KinCon-8000 input module will lead to an additional error because the corresponding output for that task cycle in which the input error occurred will be skipped.)

If the KinCon-8000 module is an output module and a task defined by the user wants to write out new data to the channels of that module the I/O-Driver will check if the previous data were already written. If so, a new timeout period starts. If not, the amount of time (from the last start period to the actual time) will be measured. If this time exceeds the user defined timeout value of this module an I/O driver error will be reported.

For Remote I-7K/I87K Modules

Driver name: The driver name of the KinCon I/O-Driver is "WinCon8x".
Parameter 1: The combinative number of COM port, baudrate index and module address.
Parameter 2: The module ID
Parameter 3: The timeout value for this module.
Parameter 4: 0

Currently several KinCon modules of the I-87000 and I-7000 series are supported by the ProConOS. "WinCon8x" IO-Driver. The following list describes these remote modules briefly.

For communicating via COM port, we have to configure the COM port parameters first. Parameter 1 is a combinative number of COM port, baudrate index and module address. The rule is:

parameter 1 = a x 2^{12} + b x 2^{8} + c
a is COM port number (2 ~ 9)
b is Baudrate index (0 ~ 7)

index 0: 1200
index 1: 2400
index 2: 4800
index 3 :9600
index 4: 19200
index 5: 38400
index 6: 57600
index 7: 115200

c is Module address (0 ~ 255)

For example:
If the COM port number is 2, baudrate is 9600, and module address is 1, the parameter 1 is
2 x 2^{12} + 3 x 2^{8} + 1 = 8961

The "**Module ID**" is the ID to enter as parameter 2 of the drivers setting.
The "**Address width**" shows the allowed entries (in bytes) for the width of the I/O address in the I/O Group.

The "**Address value type**" explains how these address are interpreted, i.e. which data types will be used for that address.

The "**Address value**" describes what kind of data will be read/written.

Module name	Module ID	Address width	Address value type	Address value
7011, 7011P	7011	4	REAL (4 bytes)	Float values of temperatures (in degree Celsius)
7012, 7012F	7012	4	REAL (4 bytes)	Float values of voltage/ current
7013	7013	4	REAL (4 bytes)	Float values of temperatures (in degree Celsius)
7014	7014	4	REAL (4 bytes)	Float values of voltage/ current
7015	7015	4, 8, 12, .. 24	REAL (4 bytes)	Float values of temperatures (in degree Celsius)
7016	7016	4, 8	REAL (4 bytes)	Float values of voltage/ current
7017, 7017F 7017C, 7017FC, 7017R, 7017RC, 7017FR,7017RC	7017	4, 8, 12, .. 32	REAL (4 bytes)	Float values of voltage/ current
7018, 7018P, 7018BL, 7018R	7018	4, 8, 12, .. 32	REAL (4 bytes)	Measure V, mV, mA, temperature(Wiht thermocouple sensor)
7019R	7019	4, 8, 12, .. 32	REAL (4 bytes)	Measure V, mV, mA, temperature(Wiht thermocouple sensor)
7021, 7021P	7021	4	REAL (4 bytes)	Float values of voltage/ current
7022	7022	4, 8	REAL (4 bytes)	Float values of voltage/ current

7024	7024	4, 8, 12, 16	REAL (4 bytes)	Float values of voltage/ current
7033	7033	4,8,12	REAL (4 bytes)	Float values of temperatures (in degree Celsius)
7041	7041	1, 2	WORD (2 bytes)	Each bit of the input WORD represents the input of the corresponding channel.
7042	7042	1, 2	WORD (2 bytes)	Each bit of the output WORD represents the output of the corresponding channel.
7043	7043	1, 2	WORD (2 bytes)	Each bit of the output WORD represents the output of the corresponding channel.
7044	7044	1	BYTE (1 byte)	Each bit of the input/output BYTE represents the input/output of the corresponding channel.
7045	7045	1, 2	WORD (2 bytes)	Each bit of the output WORD represents the output of the corresponding channel.
7050, 7050A	7050	1	BYTE (1 byte)	Each bit of the input BYTE represents the input of the corresponding channel.
7051	7051	1, 2	WORD (2 bytes)	Each bit of the input WORD represents the input of the corresponding channel.
7052	7052	1	BYTE (1 byte)	Each bit of the input BYTE represents the input of the corresponding channel.
7053	7053	1, 2	WORD (2 bytes)	Each bit of the input WORD represents the input of the corresponding channel.
7055	7055	1	BYTE (1 byte)	Each bit of the input/output BYTE represents the input/output of the corresponding channel.
7058	7058	1	BYTE (1 byte)	Each bit of the input BYTE represents the input of the corresponding channel.
7059	7059	1	BYTE (1 byte)	Each bit of the input BYTE represents the input of the corresponding channel.
7060	7060	1	BYTE (1 byte)	Each bit of the input/output BYTE represents the input/output of the corresponding channel.
7063, 7063A, 7063B	7063	1	BYTE (1 byte)	Each bit of the input BYTE represents the input of the corresponding channel.
7065, 7065A, 7065B	7065	1	BYTE (1 byte)	Each bit of the input/output BYTE represents the input/output of the corresponding channel.
7066	7066	1	BYTE (1 byte)	Each bit of the output BYTE represents the output of the corresponding channel.

7067	7067	1	BYTE (1 byte)	Each bit of the output BYTE represents the output of the corresponding channel.
87013	17013	4, 8, 12, 16	REAL (4 bytes)	Float values of temperatures (in degree Celsius)
87017	17017	4,8,12,,32	REAL (4 bytes)	Float values of voltage/ current
87018	17018	4,8,12,,32	REAL (4 bytes)	Float values of voltage/ current
87019	17019	4,8,12,,32	REAL (4 bytes)	Float values of voltage/ current
87022	17022	4, 8	REAL (4 bytes)	Float values of voltage in the range -10V to +10V.
87024	17024	4, 8, 12, 16	REAL (4 bytes)	Float values of voltage in the range -10V to +10V.
87026	17026	4, 8	REAL (4 bytes)	Float values of voltage in the range -10V to +10V.
87040	17040	1, 2, 3, 4	DWORD(4 bytes)	Each bit of the input DWORD represents the input of the corresponding channel.
87041	17041	1, 2, 3, 4	DWORD(4 bytes)	Each bit of the output DWORD represents the output of the corresponding channel.
87051	17051	1, 2	WORD (2 bytes)	Each bit of the input WORD represents the input of the corresponding channel.
87052	17052	1	BYTE (1 byte)	Each bit of the input BYTE represents the input of the corresponding channel.
87053	17053	1, 2	WORD (2 bytes)	Each bit of the input WORD represents the input of the corresponding channel.
87054	17054	1	BYTE (1 byte)	Each bit of the input/output BYTE represents the input/output of the corresponding channel.
87055	17055	1	BYTE (1 byte)	Each bit of the input/output BYTE represents the input/output of the corresponding channel.
87057	17057	1, 2	WORD (2 bytes)	Each bit of the output WORD represents the output of the corresponding channel.
87058	17058	1	BYTE (1 byte)	Each bit of the input BYTE represents the input of the corresponding channel.
87063	17063	1	BYTE (1 byte)	Each bit of the input/output BYTE represents the input/output of the corresponding channel.
87064	17064	1	BYTE (1 byte)	Each bit of the output BYTE represents the output of the corresponding channel.
87065	17065	1	BYTE (1 byte)	Each bit of the output BYTE represents the output of the corresponding channel.
87066	17066	1	BYTE (1 byte)	Each bit of the output BYTE represents the output of the

				corresponding channel.
87068	17068	1	BYTE (1 byte)	Each bit of the output BYTE represents the output of the corresponding channel.
87069	17069	1	BYTE (1 byte)	Each bit of the output BYTE represents the output of the corresponding channel.

Modbus Address V.S. Internal Address

The mapping relationship between '**Internal address'** and 'M**odbus address'** in KinCon-8000 is as below.

Modbus Address	Internal Address	Rule

Modbus Coil
[0xxxxx][1xxxxx]
	%MX a.b.c	a=0
		b=(modbus address - 1)/8
		c=(modbus address - 1)%8

Modbus Register
[3xxxxx][4xxxxx]
	%MW a.b	a=0
		b=modbus address - 1

For example:

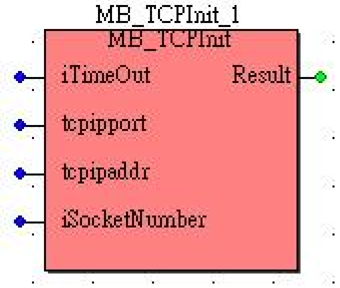
Output Coil	[000002]	➔	%MX 0.0.1
Output Register	[400003]	➔	%MW 0.2

Modbus TCP/RTU Master FBs

MB_TCPInit

This function initializes the socket you want to create.

FBD



ST

MB_TCPInit_1(iTimeOut:=(* INT *),tcpipport:=(* INT *),tcpipaddr:=(* STRING *), iSocketNumber:=(* INT *));
(* INT *):=MB_TCPInit_1.Result;

Name	Data Type	Description
iTimeOut	INT	Specifies the timeout (Response time) value for communication.
tcpipport	INT	The port number of the target Modbus/TCP device.
tcpipaddr	STRING	The IP address of the target Modbus/TCP device.
iSocketNumber	INT	The socket ID number which's range is from 0 to 255.
Result	INT	1 indicates success. (Please refer to the **APPENDIX - Error list and description**)

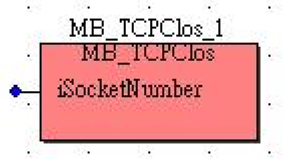
Remarks

Before you use the following Modbus/TCP function, you have to call this function to initialize your socket.

MB_TCPClos

This function close the existing socket which you created using MB_TCPInit.

FBD



ST

MB_TCPClos_1(iSocketNumber:=(* INT *));

Name	Data Type	Description
iSocketNumber	INT	The socket ID number which's range is from 0 to 255.

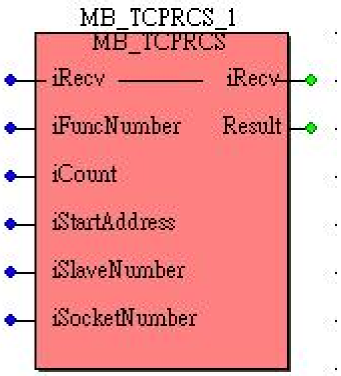
Remarks

If you don't want to use the socket anymore, you had better call this function to close the socket.

MB_TCPRCS

This function allows you to read continuous coil statuses from the Modbus/TCP device.

FBD



ST

```
MB_TCPRCS_1(iRecv:=(* MB_R_Coils *),iFuncNumber:=(* INT *),iCount:=(* INT *),
iStartAddress:=(* INT *),iSlaveNumber:=(* INT *),iSocketNumber:=(* INT *));
(* MB_R_Coils *):=MB_TCPRCS_1.iRecv;
(* UINT *):=MB_TCPRCS_1.Result;
```

Name	Data Type	Description
iRecv	MB_R_Coils	The array which contains coil statuses. The size of array must be no more than 256.
iFuncNumber	INT	The function number is either 1 or 2 which depends on your Modbus/TCP device.
iCount	INT	The count of the coils you want to read. It must be no more than 256.
iStartAddress	INT	The decimal starting address of the coils you want to read.
iSlaveNumber	INT	The slave number of your Modbus/TCP device.**(ANY)**
iSocketNumber	INT	The socket ID number you used to create using MB_TCPInit.
Result	UINT	1 indicates success. (Please refer to the **APPENDIX - Error list and description**)

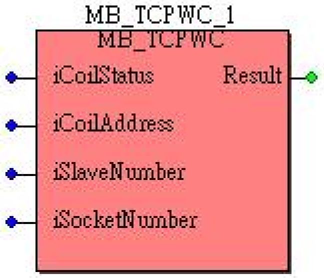
Remarks

This function uses modbus function number 1 or 2.

MB_TCPWC

This function allows you to write a coil status to the Modbus/TCP device.

FBD



ST

MB_TCPWC_1(iCoilStatus:=(* INT *),iCoilAddress:=(* INT *),iSlaveNumber:=(* INT *), iSocketNumber:=(* INT *));
(* UINT *):=MB_TCPWC_1.Result;

Name	Data Type	Description
iCoilStatus	INT	The coil status you want to give. 1 indicates TRUE. 0 indicates FALSE.
iCoilAddress	INT	The decimal address of the coil you want to write.
iSlaveNumber	INT	The slave number of your Modbus/TCP device.**(ANY)**
iSocketNumber	INT	The socket ID number you used to create using MB_TCPInit.
Result	UINT	1 indicates success. (Please refer to the **APPENDIX - Error list and description**)

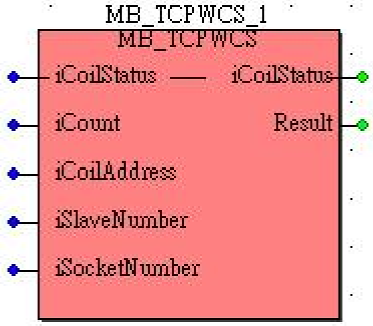
Remarks

This function uses modbus function number 5.

MB_TCPWCS

This function allows you to write several coil statuses to the Modbus/TCP device.

FBD



ST

MB_TCPWCS_1(iCoilStatus:=(* MB_W_Coils *),iCount:=(* INT *),iCoilAddress:=(* INT *), iSlaveNumber:=(* INT *),iSocketNumber:=(* INT *));
(*MB_W_Coils *):=MB_TCPWCS_1.iCoilStatus;

```
(* UINT *):=MB_TCPWCS_1.Result;
```

Name	Data Type	Description
iCoilStatus	MB_W_Coils	The array which contains coil statuses. The size of array must be no more than 256.
iCount	INT	The count of the coils you want to write. It must be no more than 256.
iCoilAddress	INT	The decimal starting address of the coils you want to write.
iSlaveNumber	INT	The slave number of your Modbus/TCP device.**(ANY)**
iSocketNumber	INT	The socket ID number you used to create using MB_TCPInit.
Result	UINT	1 indicates success. (Please refer to the **APPENDIX - Error list and description**)

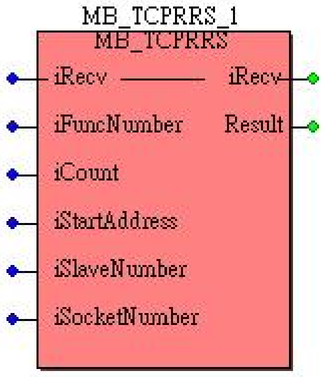
Remarks

This function uses modbus function number 15.

MB_TCPRRS

This function allows you to read continuous register values from the Modbus/TCP device.

FBD



ST

```
MB_TCPRRS_1(iRecv:=(* MB_R_Regs *),iFuncNumber:=(* INT *),iCount:=(* INT *),
iStartAddress:=(* INT *),iSlaveNumber:=(* INT *),iSocketNumber:=(* INT *));
(* MB_R_Regs *):=MB_TCPRRS_1.iRecv;
(* UINT *):=MB_TCPRRS_1.Result;
```

Name	Data Type	Description
iRecv	MB_R_Regs	The array which contains register values. The size of array must be no more than 100.
iFuncNumber	INT	The function number is either 3 or 4 which depends on your Modbus/TCP device.
iCount	INT	The count of the registers you want to read. It must be no more than 100.
iStartAddress	INT	The decimal starting address of the registries you want to read.
iSlaveNumber	INT	The slave number of your Modbus/TCP device.**(ANY)**

iSocketNumber	INT	The socket ID number you used to create using MB_TCPInit.
Result	UINT	1 indicates success. (Please refer to the **APPENDIX - Error list and description**)

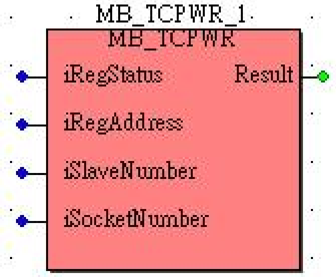
Remarks

This function uses modbus function number 3 or 4.

MB_TCPWR

This function allows you to write a register value to the Modbus/TCP device.

FBD



ST

```
MB_TCPWR_1(iRegStatus:=(* INT *),iRegAddress:=(* INT *),iSlaveNumber:=(* INT *),
iSocketNumber:=(* INT *));
(* UINT *):=MB_TCPWR_1.Result;
```

Name	Data Type	Description
iRegStatus	INT	The register value you want to give. The range is from -32768 to 32767.
iRegAddress	INT	The decimal address of the register you want to write.
iSlaveNumber	INT	The slave number of your Modbus/TCP device.**(ANY)**
iSocketNumber	INT	The socket ID number you used to create using MB_TCPInit.
Result	UINT	1 indicates success. (Please refer to the **APPENDIX - Error list and description**)

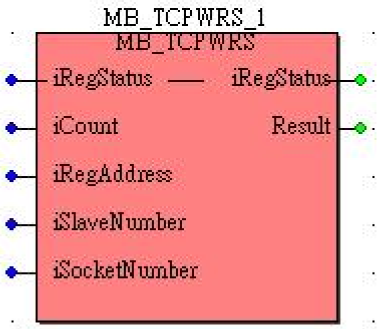
Remarks

This function uses modbus function number 6.

MB_TCPWRS

This function allows you to write several register values to the Modbus/TCP device.

FBD



ST

```
MB_TCPWRS_1(iRegStatus:=(* MB_W_Regs *),iCount:=(* INT *),iRegAddress:=(* INT
*), iSlaveNumber:=(* INT *),iSocketNumber:=(* INT *));
(* MB_W_Regs *):=MB_TCPWRS_1.iRegStatus;
(* UINT *):=MB_TCPWRS_1.Result;
```

Name	Data Type	Description
iRegStatus	MB_W_Regs	The array which contains register values. The size of array must be no more than 100. The range is from -32768 to 32767.
iCount	INT	The count of the registers you want to write. It must be no more than 100.
iRegAddress	INT	The decimal starting address of the register you want to write.
iSlaveNumber	INT	The slave number of your Modbus/TCP device.**(ANY)**
iSocketNumber	INT	The socket ID number you used to create using MB_TCPInit.
Result	UINT	1 indicates success. (Please refer to the **APPENDIX - Error list and description**)

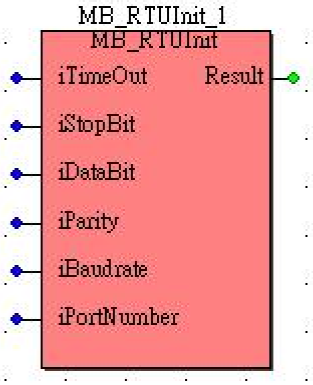
Remarks

This function uses modbus function number 16.

MB_RTUInit

This function initializes the COM port you want to create.

FBD



ST

```
MB_RTUInit_1(iTimeOut:=(* INT *),iStopBit:=(* INT *),iDataBit:=(* INT *),iParity:=(* INT
*),iBaudrate:=(* INT *),iPortNumber:=(* INT *));
(* UINT *):=MB_RTUInit_1.Result;
```

Name	Data Type	Description
iTimeOut	INT	Specifies the timeout (Response time) value for communication.
iStopBit	INT	1 means 1 stop bit 2 means 2 stop bits 3 means 1.5 stop bits
iDataBit	INT	Specifies the number of bits in the bytes transmitted and received.
iParity	INT	0 means No parity 1 means Even 2 means Mark 3 means Odd 4 means Space
iBaudrate	INT	The baud rate of COM port which should be equal to the target Modbus/RTU device.
iPortNumber	INT	The COM port number which's range is from 2 to 9.
Result	UINT	1 indicates success. (Please refer to the **APPENDIX - Error list and description**)

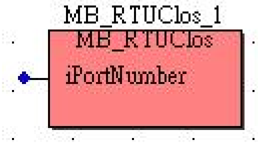
Remarks

Before you use the following Modbus/RTU function, you have to call this function to initialize your COM.

MB_RTUClos

This function close the existing COM port which you created using MB_RTUInit.

FBD



ST

```
MB_RTUClos_1(iSocketNumber:=(* INT *));
```

Name	Data Type	Description
iPortNumber	INT	The COM port number you used to create using MB_RTUInit.

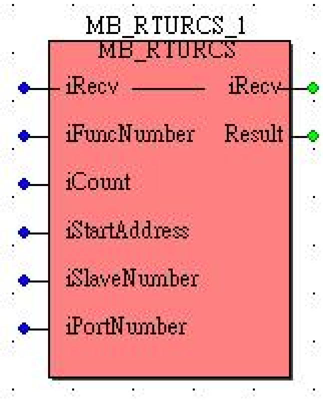
Remarks

If you don't want to use the COM port anymore, you had better call this function to close the COM port.

MB_RTURCS

This function allows you to read continuous coil statuses from the Modbus/RTU device.

FBD



ST

```
MB_RTURCS_1(iRecv:=(* MB_R_Coils *),iFuncNumber:=(* INT *),iCount:=(* INT *),
iStartAddress:=(* INT *),iSlaveNumber:=(* INT *),iPortNumber:=(* INT *));
(* MB_R_Coils *):=MB_RTURCS_1.iRecv;
(* UINT *):=MB_RTURCS_1.Result;
```

Name	Data Type	Description
iRecv	MB_R_Coils	The array which contains coil statuses. The size of array must be no more than 256.
iFuncNumber	INT	The function number is either 1 or 2 which depends on your Modbus/RTU device.
iCount	INT	The count of the coils you want to read. It must be no more than 256.
iStartAddress	INT	The decimal starting address of the coils you want to read.
iSlaveNumber	INT	The slave number of your Modbus/RTU device.
iPortNumber	INT	The COM port number you used to create using MB_RTUInit.
Result	UINT	1 indicates success. (Please refer to the **APPENDIX - Error list and description**)

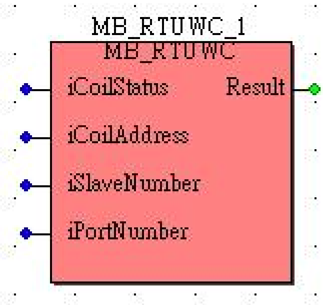
Remarks

This function uses modbus function number 1 or 2.

MB_RTUWC

This function allows you to write a coil status to the Modbus/RTU device.

FBD



ST

MB_RTUWC_1(iCoilStatus:=(* INT *),iCoilAddress:=(* INT *),iSlaveNumber:=(* INT *), iPortNumber:=(* INT *));
(* UINT *):=MB_RTUWC_1.Result;

Name	Data Type	Description
iCoilStatus	INT	The coil status you want to give. 1 indicates TRUE. 0 indicates FALSE.
iCoilAddress	INT	The decimal address of the coil you want to write.
iSlaveNumber	INT	The slave number of your Modbus/RTU device.
iPortNumber	INT	The COM port number you used to create using MB_RTUInit.
Result	UINT	1 indicates success. (Please refer to the **APPENDIX - Error list and description**)

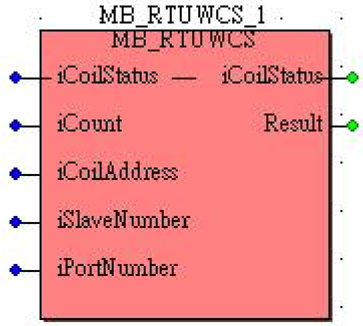
Remarks

This function uses modbus function number 5.

MB_RTUWCS

This function allows you to write several coil statuses to the Modbus/RTU device.

FBD



ST

MB_RTUWCS_1(iCoilStatus:=(* MB_W_Coils *),iCount:=(* INT *),iCoilAddress:=(* INT *), iSlaveNumber:=(* INT *),iPortNumber:=(* INT *));
(* MB_W_Coils *):=MB_RTUWCS_1.iCoilStatus;
(* UINT *):=MB_RTUWCS_1.Result;

Name	Data Type	Description
iCoilStatus	MB_W_Coils	The array which contains coil statuses. The size of array must be no more than 256.
iCount	INT	The count of the coils you want to write. It must be no more than 256.
iCoilAddress	INT	The decimal starting address of the coils you want to write.
iSlaveNumber	INT	The slave number of your Modbus/RTU device.
iPortNumber	INT	The COM port number you used to create using MB_RTUInit.
Result	UINT	1 indicates success. (Please refer to the **APPENDIX - Error list and description**)

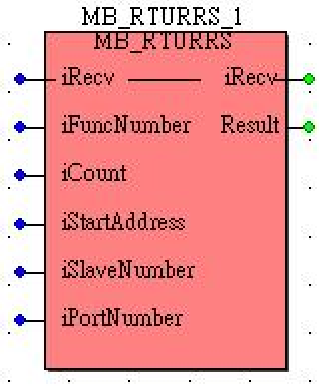
Remarks

This function uses modbus function number 15.

MB_RTURRS

This function allows you to read continuous register values from the Modbus/RTU device.

FBD



ST

```
MB_RTURRS_1(iRecv:=(* MB_R_Regs *),iFuncNumber:=(* INT *),iCount:=(* INT *),
iStartAddress:=(* INT *),iSlaveNumber:=(* INT *),iPortNumber:=(* INT *));
(* MB_R_Regs *):=MB_RTURRS_1.iRecv;
(* UINT *):=MB_RTURRS_1.Result;
```

Name	Data Type	Description
iRecv	MB_R_Regs	The array which contains register values. The size of array must be no more than 100.
iFuncNumber	INT	The function number is either 3 or 4 which depends on your Modbus/RTU device.
iCount	INT	The count of the registers you want to read. It must be no more than 100.
iStartAddress	INT	The decimal starting address of the registers you want to read.
iSlaveNumber	INT	The slave number of your Modbus/RTU device.
iPortNumber	INT	The COM port number you used to create using MB_RTUInit.
Result	UINT	1 indicates success. (Please refer to the **APPENDIX - Error list and description**)

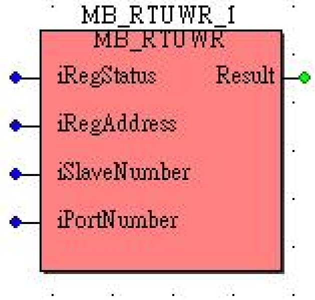
Remarks

This function uses modbus function number 3 or 4.

MB_RTUWR

This function allows you to write a register value to the Modbus/RTU device.

FBD



ST

MB_RTUWR_1(iRegStatus:=(* INT *),iRegAddress:=(* INT *),iSlaveNumber:=(* INT *),
iPortNumber:=(* INT *));
(* UINT *):=MB_RTUWR_1.Result;

Name	Data Type	Description
iRegStatus	INT	The register value you want to give. The range is from -32768 to 32767.
iRegAddress	INT	The decimal address of the register you want to write.
iSlaveNumber	INT	The slave number of your Modbus/RTU device.
iPortNumber	INT	The COM port number you used to create using MB_RTUInit.
Result	UINT	1 indicates success. (Please refer to the **APPENDIX - Error list and description**)

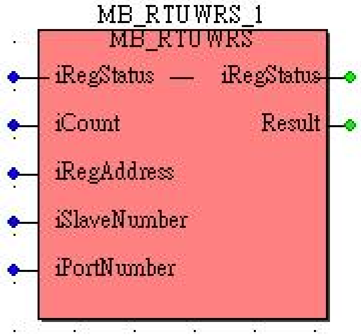
Remarks

This function uses modbus function number 6.

MB_RTUWRS

This function allows you to write several register values to the Modbus/RTU device.

FBD



ST

```
MB_RTUWRS_1(iRegStatus:=(* MB_W_Regs *),iCount:=(* INT *),iRegAddress:=(* INT
*), iSlaveNumber:=(* INT *),iPortNumber:=(* INT *));
(* MB_W_Regs *):=MB_RTUWRS_1.iRegStatus;
(* UINT *):=MB_RTUWRS_1.Result;
```

Name	Data Type	Description
iRegStatus	MB_W_Regs	The array which contains register values. The size of array must be no more than 100. The range is from -32768 to 32767.
iCount	INT	The count of the registers you want to write. It must be no more than 100.
iRegAddress	INT	The decimal starting address of the register you want to write.
iSlaveNumber	INT	The slave number of your Modbus/RTU device.
iPortNumber	INT	The COM port number you used to create using MB_RTUInit.
Result	UINT	1 indicates success. (Please refer to the **APPENDIX - Error list and description**)

Remarks

This function uses modbus function number 16.

Error list and description

Code Description I/O Unit Min Max		
Code	Define	**Description**
101	**MB_OPEN_PORT_ERROR**	Open COM/TCP Port error
102	**MB_PORTNO_OVER**	COM Port is 1 - 8
103	**MB_PORT_NOT_OPEN**	COM/TCP Port does not open yet
104	**MB_FUN_ERROR**	Modbus Fun. No. error
105	**MB_READ_COUNT_OVER**	reading Count of Register or Bits is over range RTU: 120 register, 1920 coils ASCII: 60 register, 960 coils TCP: 120 register, 1920 coils
106	**MB_SLAVENO_OVER**	Modbus Slave No. must be 1 - 247
107	**MB_ADDRESS_OVER**	Register or Coil Address must count from 1
108	**MB_COMM_TIMEOUT**	Comm. timeout
109	**MB_CRC_ERROR**	RTU CRC Check error
110	**MB_LRC_ERROR**	ASCII LRC Check error
111	**MB_INVALID_SOCKET**	Initial Socket error
112	**MB_TCP_CONNECT_ERROR**	Connect Remote Modbus Server error
113	**MB_TCP_SEND_ERROR**	Send TCP Data error
114	**MB_TCP_TIMEOUT**	Waiting Modbus Response Timeout
115	**MB_WSA_INIT_ERROR**	WSA Startup error
116	**MB_TCP_SOCKET_ERROR**	Create Socket error
117	**MB_TCP_BIND_ERROR**	TCP Server Bind error
118	**MB_TCP_LISTEN_ERROR**	TCP Server Listen error
119	**MB_TCP_HAS_DATA**	it has data from remote Modbus Master
120	**MB_WRITE_COUNT_OVER**	reading Count of Register or Bits is over range RTU: 120 register, 1920 coils ASCII: 60 register, 960 coils TCP: 120 register, 1920 coils

Demo List

MultiProg Project	Description
My_First_Project_Rt_Mb.zwt	Project of Quick_Start_Guide
MBMasterDemo.zwt	Modbus Master Demo Project
MultiLanguage.zwt	IEC-61131 Language Demo Project
ProVisIt Project	**Description**
Demo1_800x600.vwt	ProVisIt demo showing basic component from ICPDAS
MotorControl.vwt	ProVisIt project of Quick_Start_Guide