PLC500N series

Programmable Logic Controllers

PLC500 Nseries
Overview

Edition 1.1
November 2007

Overview

Creating a project

When MULTIPROG wt (MWT) is first installed, a shortcut to the program will be created on the desktop of your computer. Double clicking on this shortcut runs MWT.

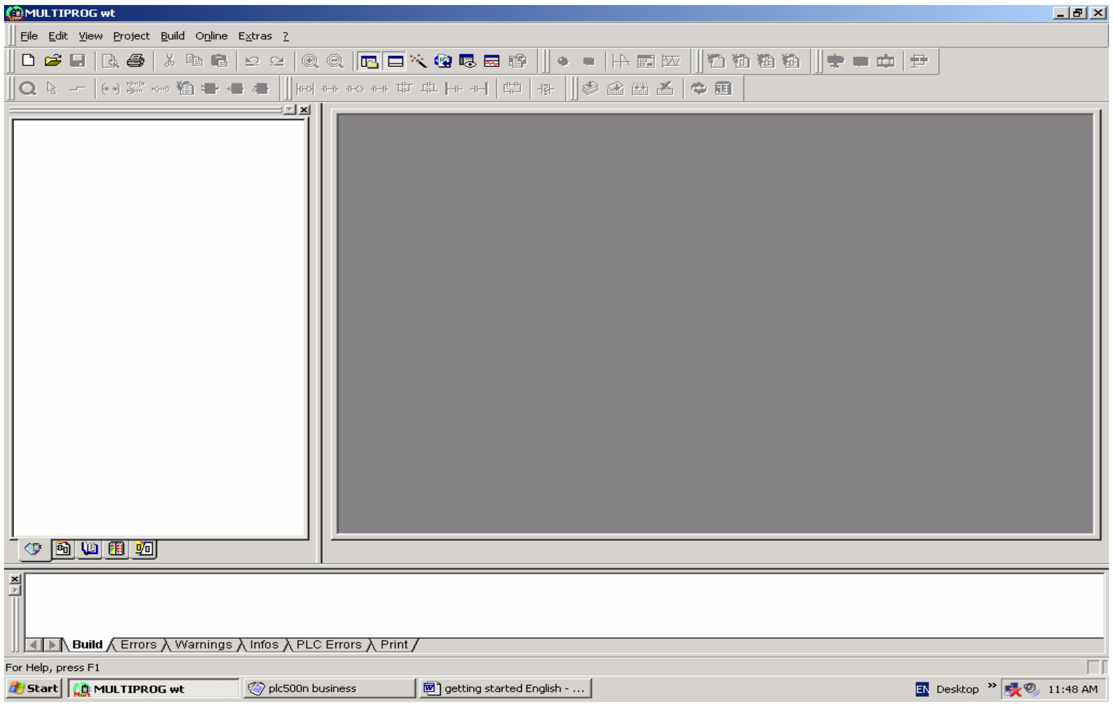

MULTIPROG wt
File Edit View Project Build Online Extras ?
Build Errors Warnings Infos PLC Errors Print
For Help, press F1
Start MULTIPROG wt plc500n business getting started English - ...
Desktop 11:48 AM

Figure 1: Basic window of MWT

Note: A manual of MWT named MWTMAN21_001.PDF will also be generated in install directory of MWT. In this manual you will find all details about creating and handling a project including the procedures relating to editing POUs in different programming languages LD, FBD, SFC, IL and ST. Please refer to this manual in future.

From menu bar of MWT select File and then New Project. The following dialog box appears.

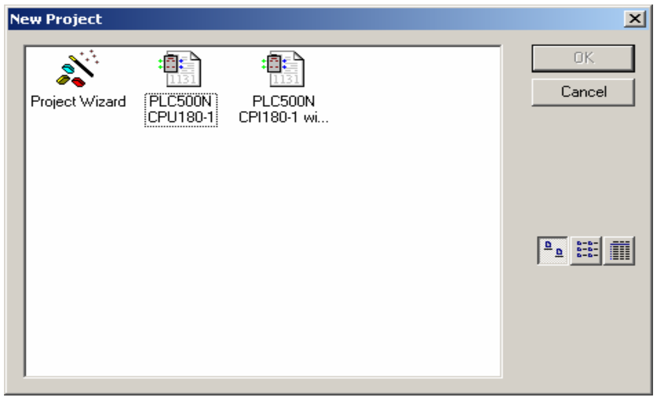


Figure 2: Project wizard window

There are some options to create a new project.

- You may select “PLC500N CPU180-1” to create a small project with default settings and files.
- You may select “PLC500N CPU180-1 with Modbus” to create a small project with default settings and files and also a Modbus Global variable worksheet.
- Or you may select “Project Wizard” to create a project by asking questions in different steps.

If you select “PLC500N CPU180-1”, a small project “Untitled” will be created. Overall structure of the project will be shown in the “Project Tree Window” on the left side of the screen.

Untitled project will have the following units.

- A blank POU Main to be edited.
- A Configuration based on PLC500 Nseries.
- A Resource or CPU based on CPU180-1
- A CYCLIC task with execution order 0 and a time interval of 10 ms.
- POU Main is inserted in Task1 to be executed every 10 ms.

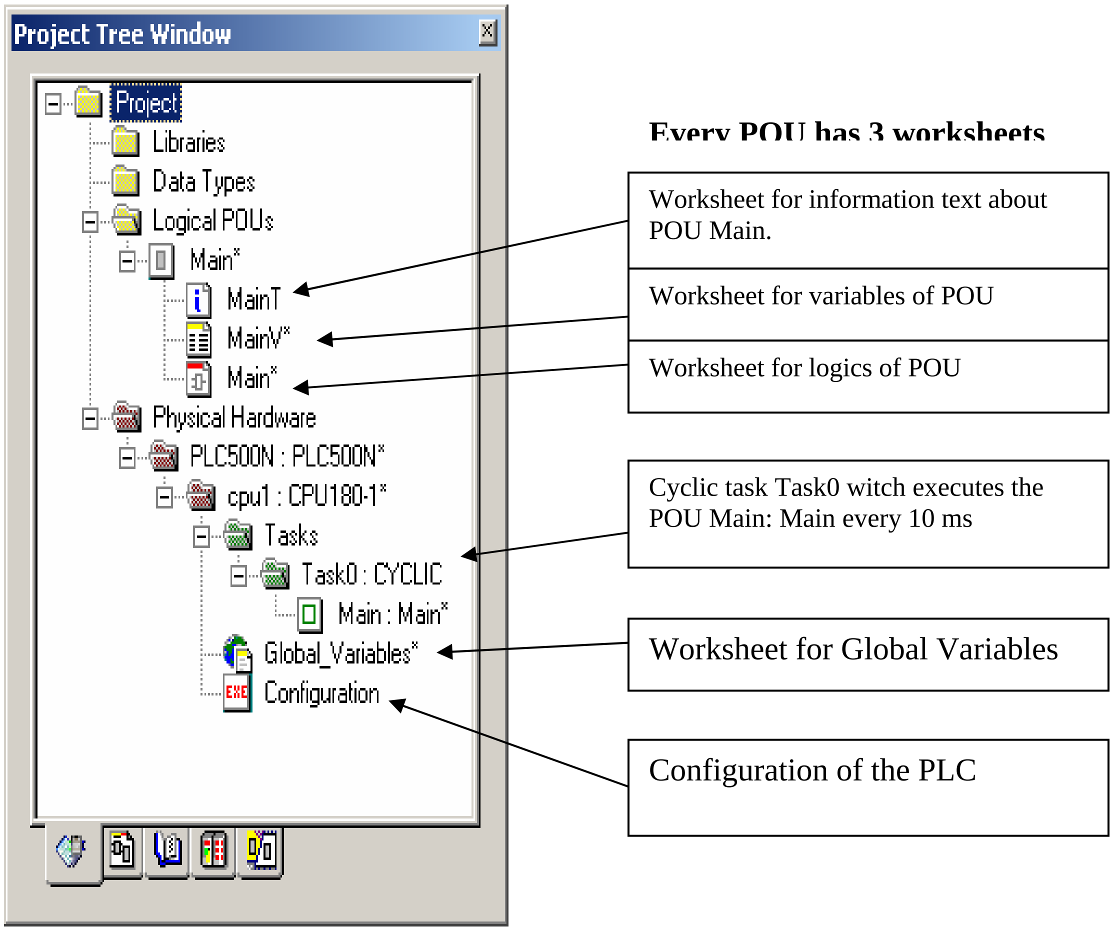
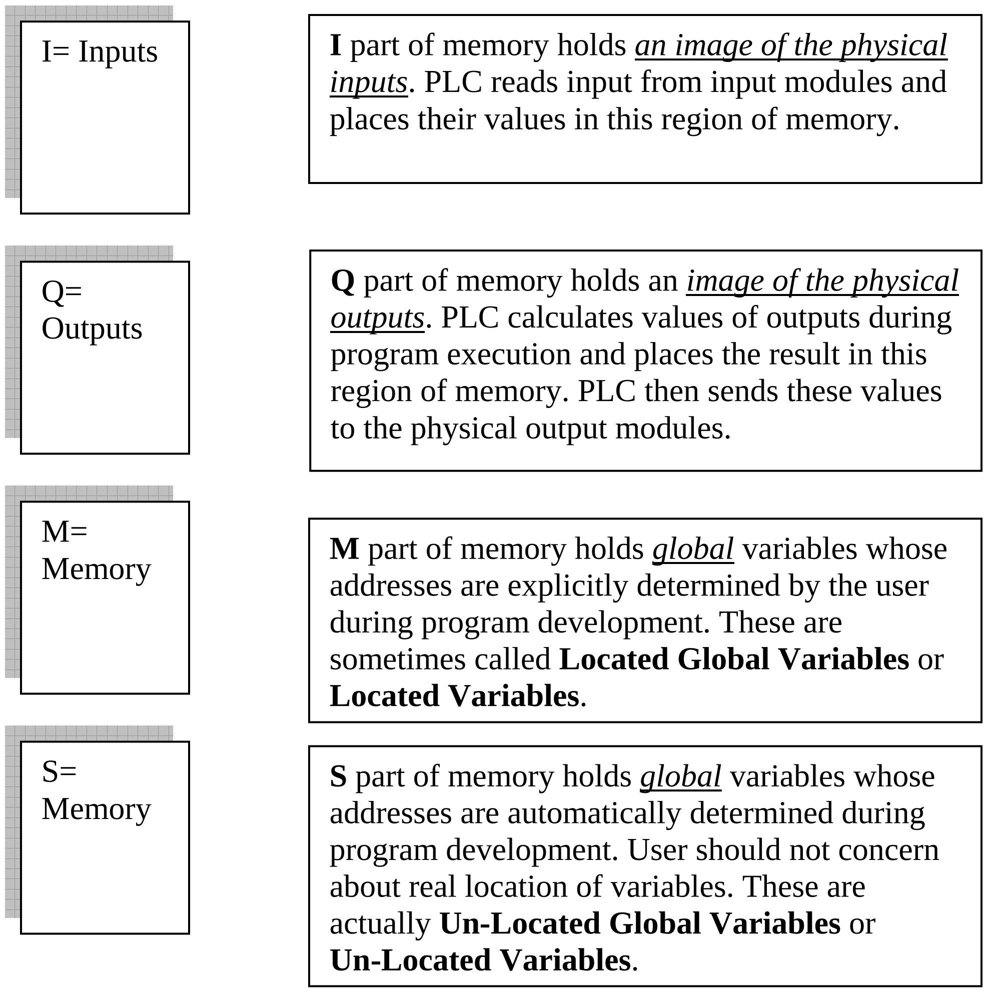

Project Tree Window
Project
Libraries
Data Types
Logical POUs
Main*
MainT
MainV*
Main*
Physical Hardware
PLC500N : PLC500N*
cpu1 : CPU180-1*
Tasks
Task0 : CYCLIC
Main : Main*
Global_Variables*
Configuration
Every POU has 3 worksheets
Worksheet for information text about POU Main.
Worksheet for variables of POU
Worksheet for logics of POU
Cyclic task Task0 witch executes the POU Main: Main every 10 ms
Worksheet for Global Variables
Configuration of the PLC

Figure 3: Project Tree Window

In this way you have built the basic elements of a typical project. Although this is a complete project it does nothing for you. You have to enter data and information in its blank worksheets.
Now that the basic skeleton of a project is available, you have to fill out the blank worksheets according to your requirements. Before doing this, let's review some concepts regarding Process Images, Structure of POUs, Local variables, Global variables, Located variables and memory structure of the CPU.

Process Images in CPU180
Inputs, Outputs and Global variables are hold in some dedicated parts of memory commonly known as process images.



I part of memory holds *<u>an image of the physical inputs</u>*. PLC reads input from input modules and places their values in this region of memory.

Q part of memory holds an *<u>image of the physical outputs</u>*. PLC calculates values of outputs during program execution and places the result in this region of memory. PLC then sends these values to the physical output modules.

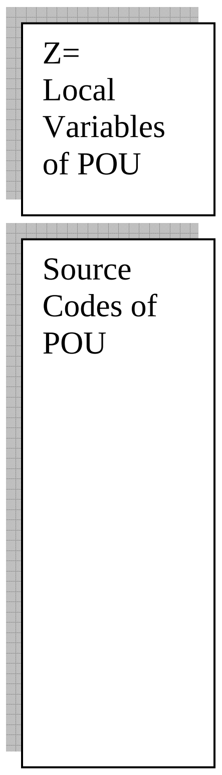
M part of memory holds *<u>global</u>* variables whose addresses are explicitly determined by the user during program development. These are sometimes called **Located Global Variables** or **Located Variables**.

S part of memory holds *<u>global</u>* variables whose addresses are automatically determined during program development. User should not concern about real location of variables. These are actually **Un-Located Global Variables** or **Un-Located Variables**.

Figure 4: Process Images in CPU180

Structure of POUs in CPU180

Every POU in CPU180 has a special part of memory allocated for its Local variables. This part of memory is automatically generated during program development along with its source codes. Once a POU is generated, local variables are generated and initialized appropriately. Local variables are only accessible to the POU they belong to.



Z part of a POU holds *local* variables whose addresses are automatically determined during program development. User should not concern about real location of variables.

Figure 5: Structure of POUs in CPU180

Local and Global Variables

The scope of each variable is limited either to a POU or to the whole project. Therefore two types can be distinguished:

- Local variables
- Global variables

If a variable can be used only within a POU it is called *local variable*.
If a variable can be used within the whole project it is called *global variable*

Normally all I/Os as well as variables that are to be available in all other POUs must be declared as global variable.

Local variables are the variables that are valid in just the POU they are declared.

Located and Non-Located Variables

Global variables are either Located or Non-Located.
Located variables are placed in I, Q and M-part of process images and their addresses are determined by the user. User is also responsible for possible overlap of addresses and probable side-effects.
Non-located variables are automatically generated and placed in S-part of process image. System is responsible for appropriate placement of these variables and there is no possibility of misplacement.
As far as there is no need to assign a specific address for a global variable (of course different from inputs and outputs), it is wise to leave the system free to determine its address.

Size of variables

The following tables show the size prefix for directly represented and located variables:

Size prefix	*Description*
X	Single bit size
None	Single bit size
B	Byte size (8 bits)
W	Word size (16 bits)
D	Double word size (32 bits)

Examples:

Variable	*description*
%IX4.5	Input Bit 5 of byte 4
%QX6.2	Output Bit 2 of byte 6
%Q6.2	Output Bit 2 of byte 6
%MB56	Memory Byte 56
%QW32	Output Word 32 (Bytes 33, 32)
%ID73	Input Double 73 (Bytes 76, 75, 74, 73)

Please note that address of a variable in case of W and D is LSB (Least Significant Byte) of the variable.
Please also note that there may be overlap of variables when addressing in multi-byte variables. Figure 6 describes this overlap. Values of some variables in this figure will be as follows:

IB0	= 32
IW0	= 5A32
IW1	= 9C5A
ID2	= 7D43DF9C

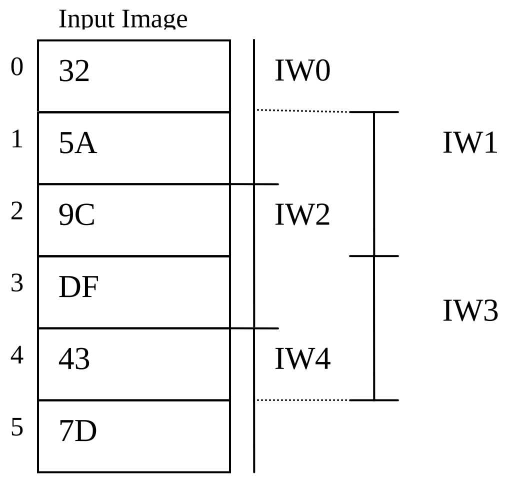


Figure 6

Data Types

Traditionally, PLCs could manipulate BIT (BOOL), BYTE, WORD and DWORD variables. These data types were all they needed because they did not support many sophisticated arithmetic functions. When some limited arithmetic operations were encountered, the user was responsible to handle them with his or her manipulations. For example the user had to check the MSB of a WORD to check if it is a positive or negative number.

In addition to basic logical operations on bits, bytes, words and double words, modern PLCs such as PLC500 Nseries, can handle many arithmetic and trigonometric functions. That's why a more sophisticated definition of data types is proposed by the standard IEC1131-3. In this standard following data types are defined as elementary data types.

Data type	**Description**	**Size**	**Range**	**Default initial value**
BOOL	Boolean	1	0...1	0
SINT	Short integer	8	-128...127	0
INT	Integer	16	-32768...32767	0
DINT	Double integer	32	-2.147.483.648 up to 2.147.483.647	0
USINT	Unsigned short integer	8	0 up to 255	0
UINT	Unsigned integer	16	0 up to 65535	0
UDINT	Unsigned double integer	32	0 up to 4.294.967.295	0
REAL	Real numbers	32	3.4E^-38 up to 3.4E^38	0.0
TIME	Duration	32		t#0s
BYTE	Bit string of length 8	8		0
WORD	Bit string of length16	16		0
DWORD	Bit string of length 32	32		0

It is now clear that addition, subtraction, division and multiplication on INT variables does make sense, but on BOOL or BYTE variables, it doesn't.

Note: In case of located variables, one can locate variables of type INT or UINT at a WORD address because both of these types have the same size of 32 bits. Similarly SINT and USINT may be located at BYTE boundaries.

Generic data types

Generic data types are data types which include hierarchical groups of elementary data types. ANY_INT includes the elementary data types DINT, INT, SINT, UDINT, UINT and USINT. If a function can be connected with ANY_INT it means that variables of the data types DINT, INT, SINT, UDINT, UINT and USINT can be handled.

Generic data types are shown in the following table:

```
ANY
        ANY_NUM
                ANY_REAL
                        REAL
                ANY_INT
                        DINT, INT, SINT
                        UDINT, UINT, USINT
        ANY_BIT
                DWORD, WORD, BYTE, BOOL
        TIME
```

Symbolic Variables

In PLC500 Nseries and in consistency with IEC1131-3, variables are recognized by a symbolic name and a data type rather than their addresses.

The following example shows some variables:

```
PressHigh_163                           :BOOL;
PressSwitch_105:     AT     %IX0.5      :BOOL;
TempValue_210        AT     %IW4        :INT;
```

Variables are declared with a symbolic name and a data type. The initial value is optional.

For more information on variables, please refer to the manual of MULTIPROG wt.

Filling the blank worksheets of the project

Every POU has three worksheets. For POU Main these worksheets are MainT, MainV and Main. Purpose of the three worksheets is:

- MainT worksheet for help texts about POU and its objectives
- MainV worksheet for variables of POU
- Main worksheet for body of programs

Double click on the Main worksheet of the POU. The corresponding worksheet opens.

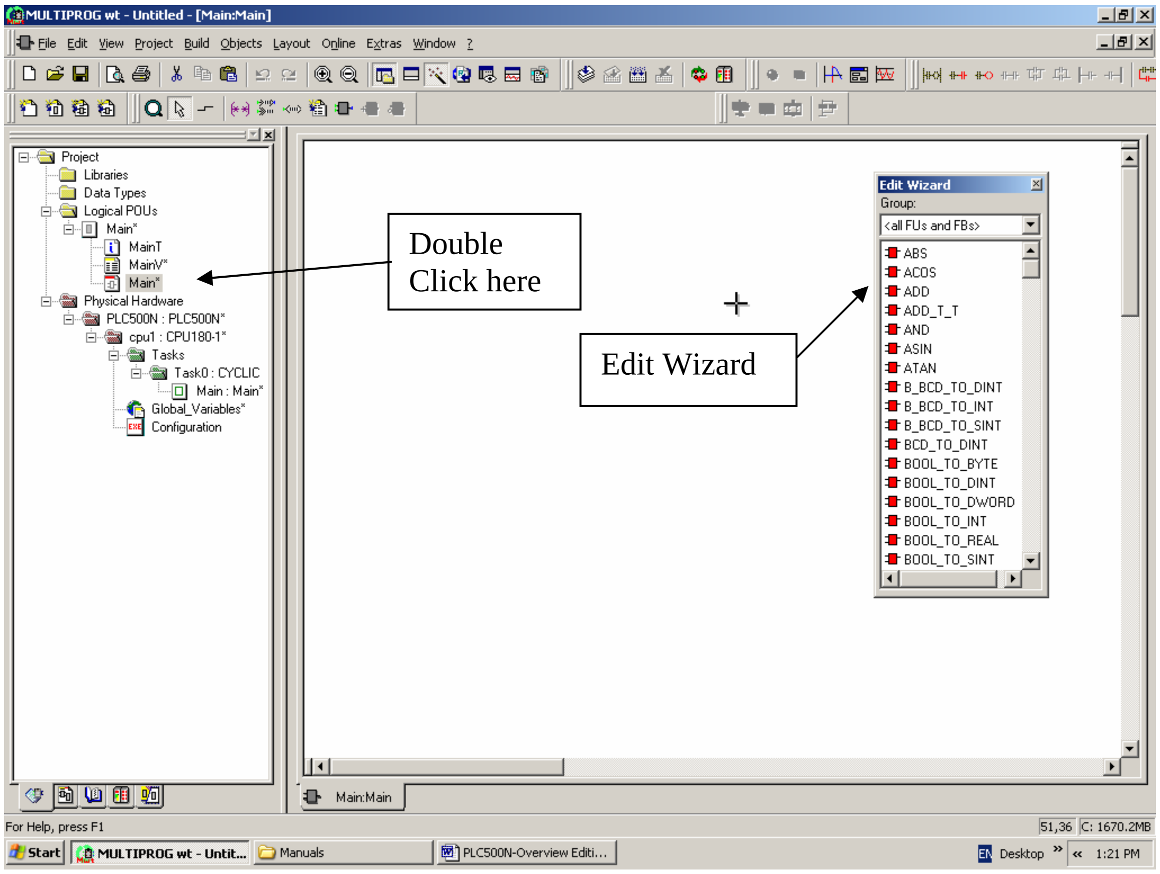


Figure 7: General view of MWT

To help you edit your control programs easier, edit wizards are provided in MWT. Edit wizard will be displayed by selecting it from the tool bar as shown below.

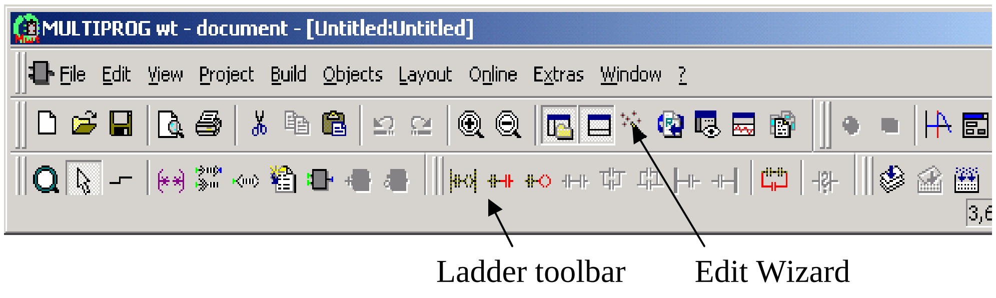

MULTIPROG wt - document - [Untitled:Untitled]
File Edit View Project Build Objects Layout Online Extras Window ?
Ladder toolbar
Edit Wizard

Figure 8: Menus and Tool bars of MWT

If you are going to edit your POU in LD, use Ladder toolbar and if in FBD, use functions shown in edit wizard.
Please note that edit wizard may be inactive. Place cursor somewhere in worksheet and click to activate it.

Place some logics into the POU such as shown below.

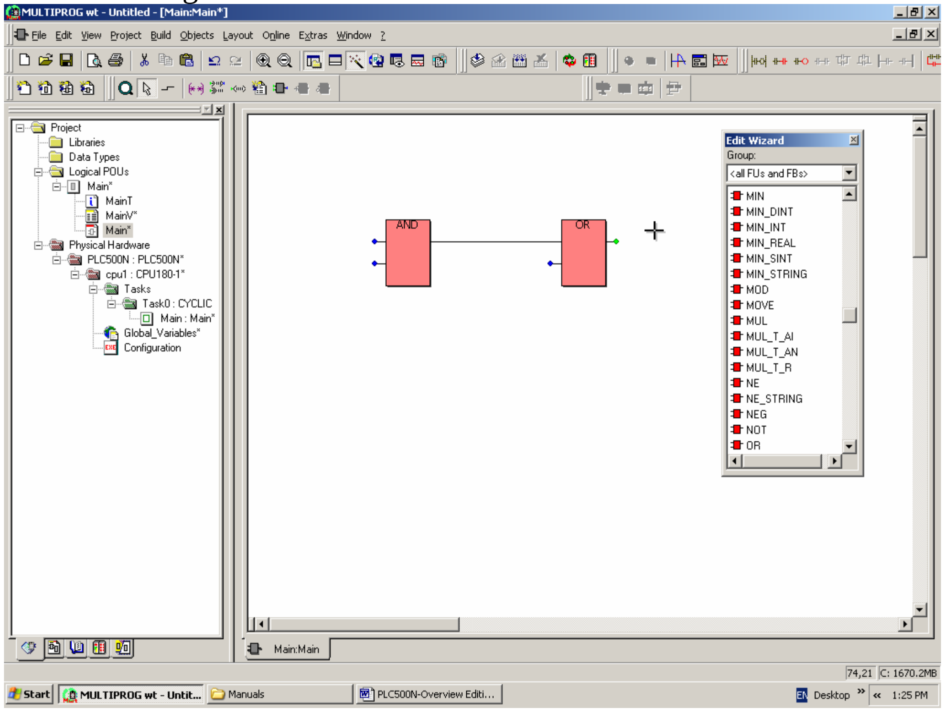


Figure 9: Editing POUs in MWT

Now you have to define variables connected to the AND and OR functions.
Double click on the first input of the AND function. Variable dialog box appears.

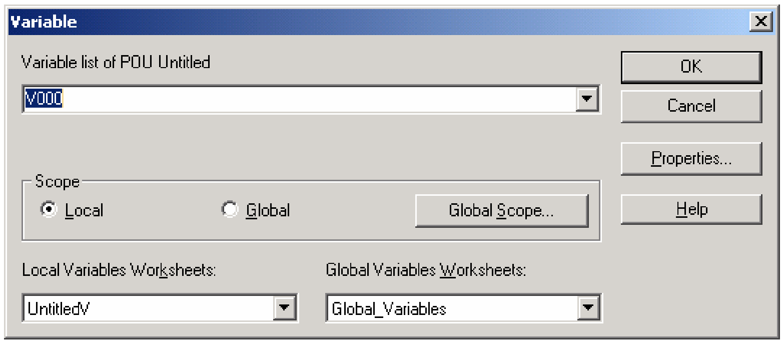


Figure 10: Variable declaration dialog box

In this dialog box a variable with a default name (V000) is proposed. You have to decide on its name, scope and data type.

- Change the default name to PS100_High.
- Change its scope as Global.
- Click on the button OK, a second dialog box appears.

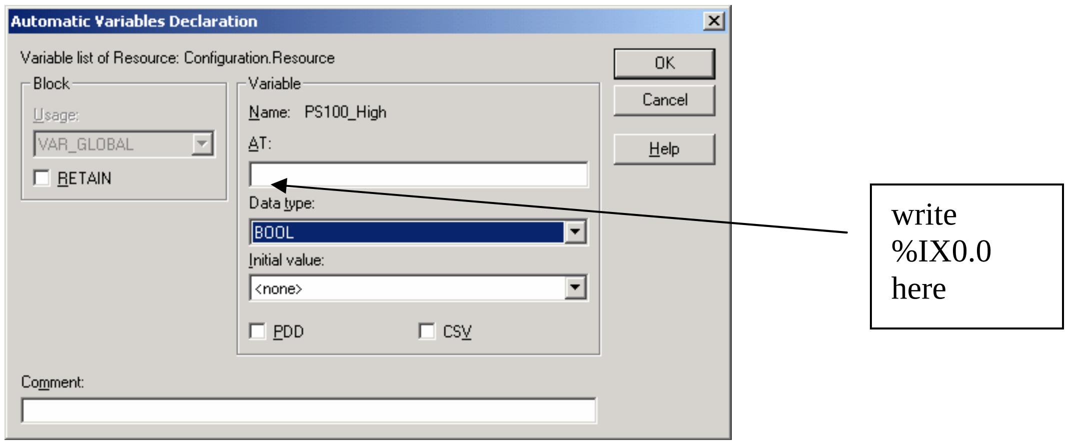


Figure 11: Automatic Variable Declaration dialog box

Here you have to decide upon data type and optionally its location and initial value. Let's this variable be a pressure switch connected to the digital input I0.0.
Enter the following entries

- Type %IX0.0 at the edit box AT.
- Accept data type BOOL.
- Accept Initial value of <none>
- Press the button OK

Now double click on the Global_Variables in the Project Tree Window. The worksheet for Global_Variables will open and following texts will be shown in it.

```
VAR_GLOBAL     (*AUTOINSERT*)
        PS100_High   AT %IX0.0    :        BOOL;
END_VAR
```

This means that the variable “PS100_High” that is connected to the input IX0.0 is now added to the global variables as a Boolean variable. This variable will be available to all other POUs throughout this project.
This time double click the worksheet MainV in the Project Tree Window. The worksheet for variables of POU Main will open and following text will be shown in it.

```
VAR_EXTERNAL     (*AUTOINSERT*)
        PS100_High   :        BOOL;
END_VAR
```

This means that the variable “PS100_High” that is know a global variable is now imported in worksheet of variables of POU Main. The keyword VAR_EXTERNAL indicates that this variable (defined externally) is a global variable imported to this POU..

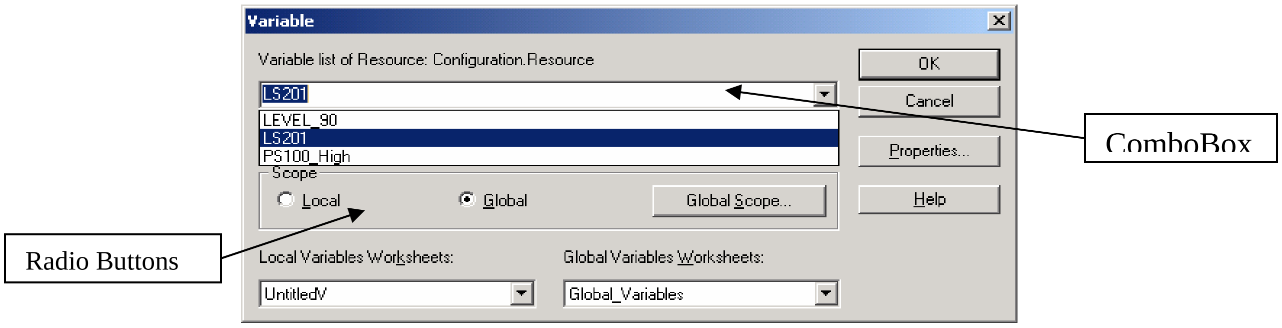
You can add other variables in the same way but we add other variables differently in order to show other methods.

Adding Global Variables Manually

Open Global_Variable worksheet, insert the required lines as follows and save it..

```
VAR_GLOBAL     (*AUTOINSERT*)
        PS100_High   AT %IX0.0    :      BOOL;
        LS201        AT %IX0.1    :      BOOL;
        LEVEL_90     AT %IX0.2    :      BOOL;
END_VAR
```

Now double click on the second input of AND function. Variable dialog box appears.



Variable list of Resource: Configuration.Resource

LS201

LEVEL_90
LS201
PS100_High

Scope

Local [✓ Global]

Local Variables Worksheets: UntitledV

Global Variables Worksheets: Global_Variables

Figure 12: All variables declared are accessible from the ComboBox

Select global scope with Global radio button. Now drop down the contents of the ComboBox of variable list.

You will see that all three entries for the variables existing in the worksheet of Global_Variables are shown in the drop-down list. Select LS201 (Limit Switch 201) from the list and press OK.
In the same way select LEVEL_90 for the second input of OR function. Overall view of Main will be as shown below.

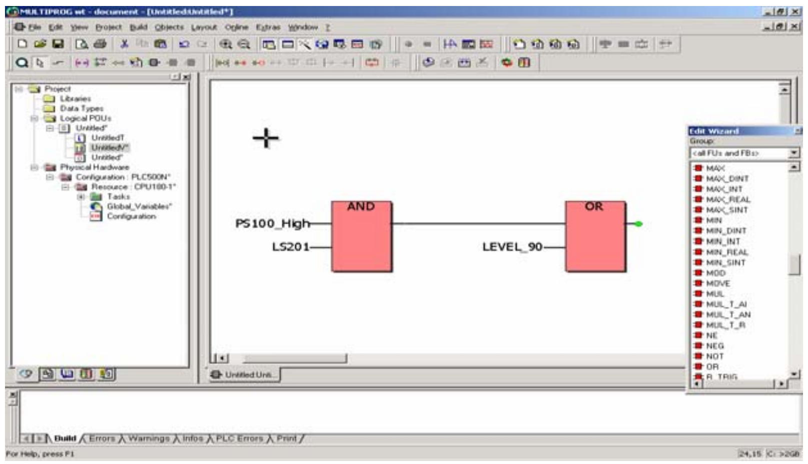
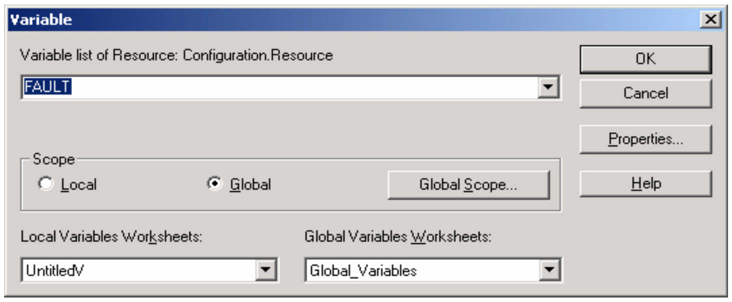


Figure 13: Overall View of Main

Note that only *Located Global Variables* are declared so far.
Open the worksheet of MainV and see the variables added.

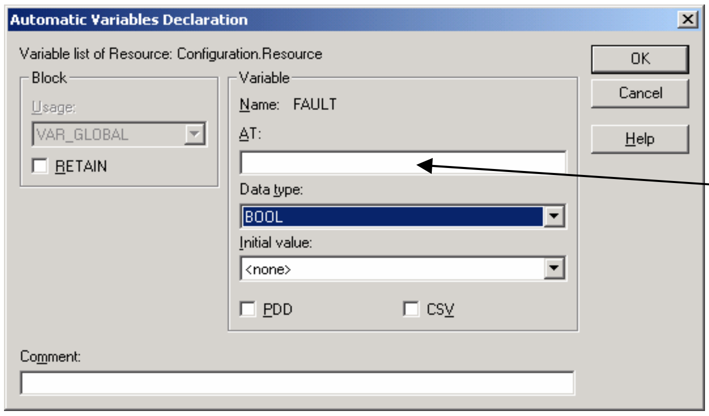
Now let's add a Non_Located Global Variable to the project.
Double click on the output of OR function. Add a new variable named "FAULT" with global scope.



Variable list of Resource: Configuration.Resource
FAULT
Scope: Local [✓ Global]
Local Variables Worksheets: UntitledV
Global Variables Worksheets: Global_Variables

Figure 14: Declaration of variable FAULT

The dialog box "Automatic Variables Declaration" will appear.



Variable list of Resource: Configuration.Resource
Block
Usage: VAR_GLOBAL
RETAIN
Variable
Name: FAULT
AT:
Data type: BOOL
Initial value: <none>
PDD
CSV
Comment:

No entry in this position means that you have left it to the system to determine its location in S-memory.

Figure 15: Automatic variable declaration dialog box for FAULT

Accept the data type BOOL with no location specification (no AT entry) and press OK. Another global variable is automatically generated with some location in *S-memory*. Please check worksheets of Global_Variables and MainV to see how they are updated. You could declare the same variable to be located at some specific location in M-memory but please remember that total management of M-memory is at your responsibility.

Adding Local Variables

There may be a lot of intermediate results that must be preserved for subsequent references throughout a single POU. While other POUs are not interested to such values, why should we waste valuable global variables to hold them?
There also arise many situations that we insert FBs (FUNCTION-BLOCKS) such as Timers, Counters and Flip-Flops in our POUs. Recalling that FBs have internal variables for their own internal activities, POUs can simply lend their own local variables to FBs

automatically. This way the user doesn't need to keep track of variables as they are automatically allocated or removed. This feature helps having infinite number of Timers, Counters etc. without having to deal with a great number of variables and/or data blocks.

Adding local variables to a POU is done either intentionally by a user or automatically by inserting FBs or sometimes by interconnecting I/Os of functions together. Automatic insertion of local variables has nothing to be concerned about. Explicit insertion of local variables by users is very easy as described below.

Add a function AND to the Main as shown below.

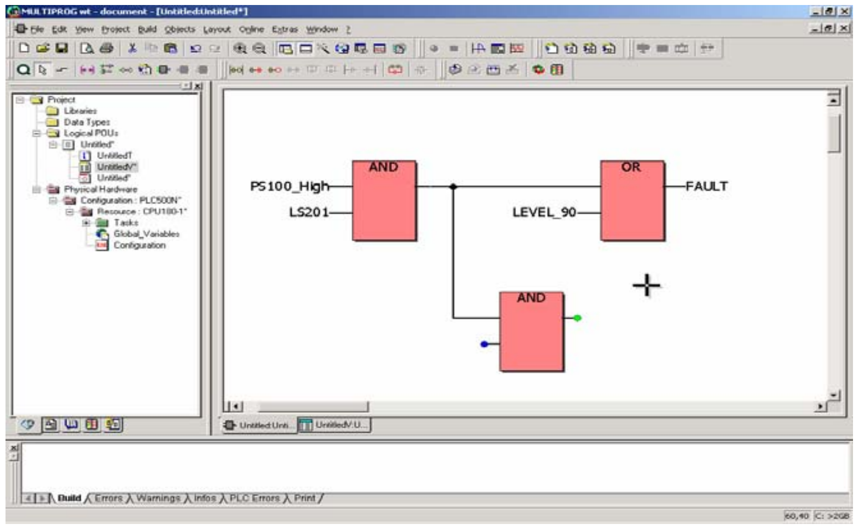
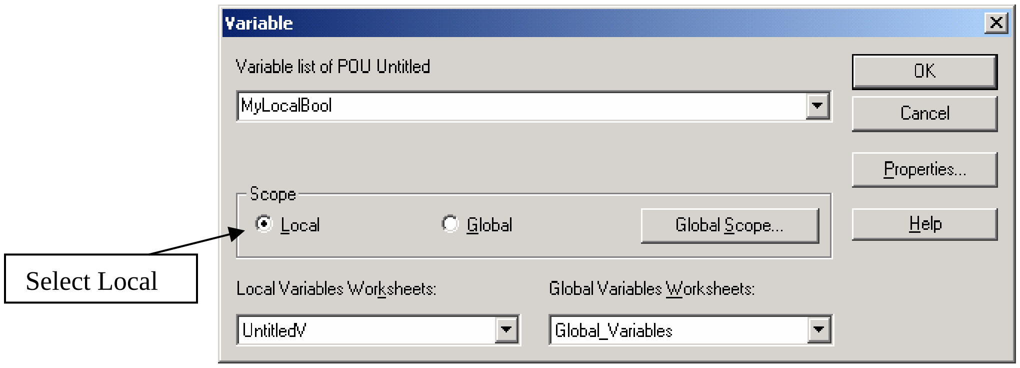


Figure 16: Adding a function AND to the worksheet

Double click on the output terminal of the new AND function. Variable dialog box appears. Type a name for this variable such as MyLocalBool and select <u>Local scope</u> for it.



Variable

Variable list of POU Untitled: MyLocalBool

Scope: [✓ Local] Global

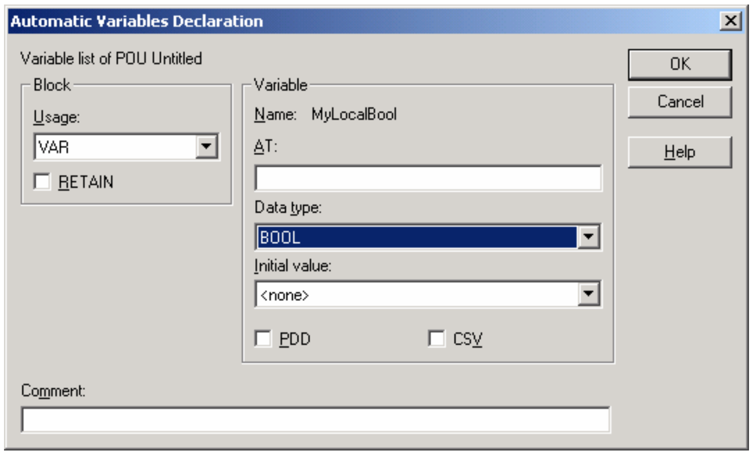
Local Variables Worksheets: UntitledV

Global Variables Worksheets: Global_Variables

Figure 17: Variable declaration dialog box

Press OK button. Automatic Variables Declaration dialog box will appear as shown.

- Select Usage as VAR which means Local Variable.
- *Do not specify a location for it*.
- Select a data type of BOOL.
- Press OK button.



Automatic Variables Declaration

Variable list of POU Untitled

Block

Usage: VAR

☐ RETAIN

Variable

Name: MyLocalBool

AT:

Data type: BOOL

Initial value: <none>

☐ PDD ☐ CSV

Comment:

Figure 18: Automatic variable declaration dialog box.

A local variable is now generated that is available within the POU Main only. If you open the worksheet MainV you will see;

```
VAR_GLOBAL      (*AUTOINSERT*)
        PS100_High   AT %IX0.0    :      BOOL;
        LS201        AT %IX0.1    :      BOOL;
        LEVEL_90     AT %IX0.2    :      BOOL;
END_VAR

VAR          (*AUTOINSERT*)
        MyLocalBool :        BOOL;
END_VAR
```

Complete the POU as shown below.

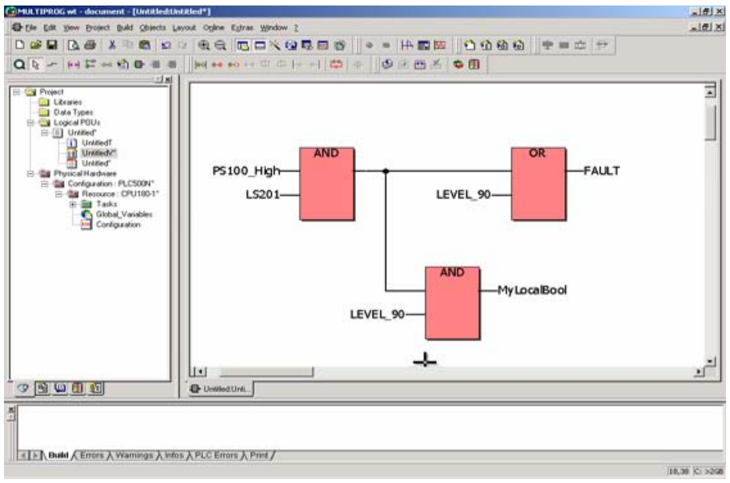


Figure 19: POU Main

Review Task1

Recalling that Task1 is a cyclic task which executes every 10 ms, any POU inserted to this task will also be executed every 10 ms. You can check these settings by a right click on Task1..

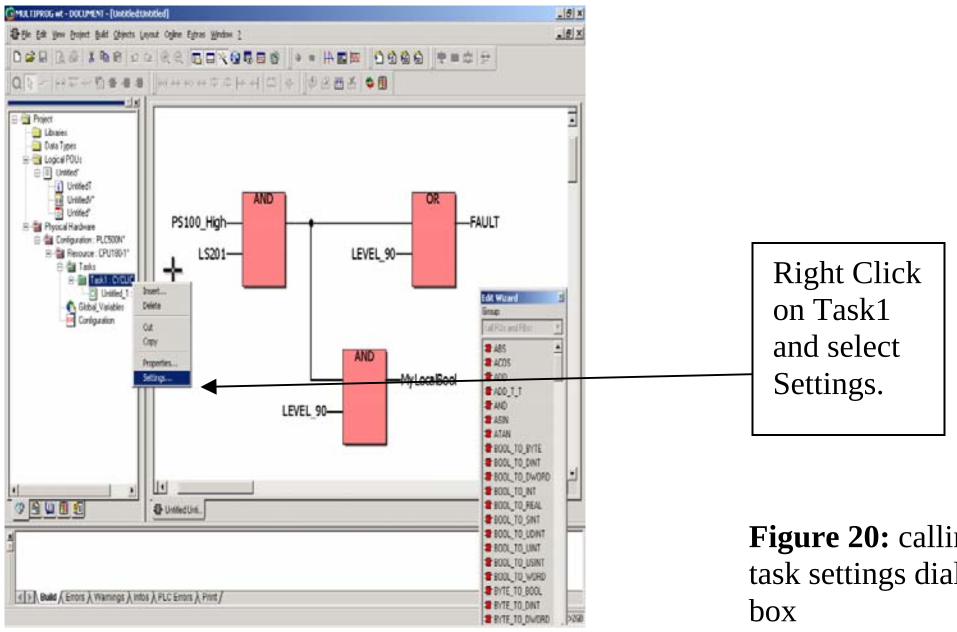
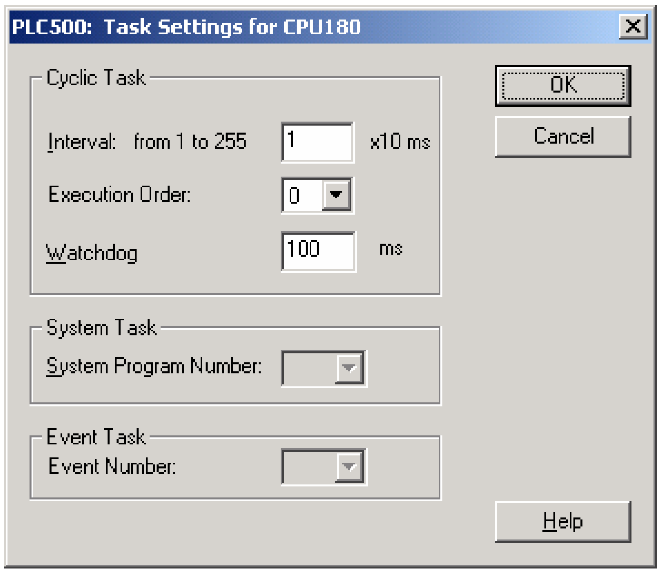


Figure 20: calling task settings dialog box

Dialog Box Task Settings will appear. In this dialog box default settings values for Task1 will be shown and can be changed. Task1 with execution order of 0 (highest priority) and cyclic time interval 0f 10 mSec is set. Different types of tasks with setting values will be discussed.



PLC500: Task Settings for CPU180

Cyclic Task

Interval: from 1 to 255 [1] x10 ms

Execution Order: [0]

Watchdog [100] ms

System Task

System Program Number: [ ]

Event Task

Event Number: [ ]

OK
Cancel
Help

Figure 21: task settings dialog box

Tasks in IEC 1131-3

Tasks determine the time scheduling of the programs associated with them. This means that programs have to be associated to tasks. The settings of the task determine the time scheduling.
IEC 1131-3 describes different time scheduling models which results in three different task types:

- **Cyclic tasks** are activated in a certain time interval and the program is executed periodically.
- **System tasks** are called automatically by the PLC operating system if an error or a change of the operational state of the PLC occurs. They are also known as system programs or SPGs.
- **Event or interrupt** tasks are activated if a certain event has happened.

Each task has a certain priority. In so called preemptive scheduling systems, an active task with low priority is interrupted immediately, when a task with higher priority becomes active due to a certain event. In systems with non-preemptive scheduling, task interruptions by tasks with higher priority are not possible.

Cyclic Tasks in PLC500 Nseries, CPU180

CPU180 supports up to ten cyclic tasks. Priority in cyclic tasks is set with a parameter called *execution order*. Cyclic time interval of these tasks can be set from 10 to 2550 ms. When several cyclic tasks are defined, there is always the possibility that two or more tasks are to be resumed at the same time. Priority of execution depends on the parameter execution order.
Let's assume that there are three cyclic tasks called Task_A, Task_B and Task_C with cycle time interval of 10, 20 and 30 ms and execution order of 0, 1 and 2 respectively. The following time diagram shows execution of them.

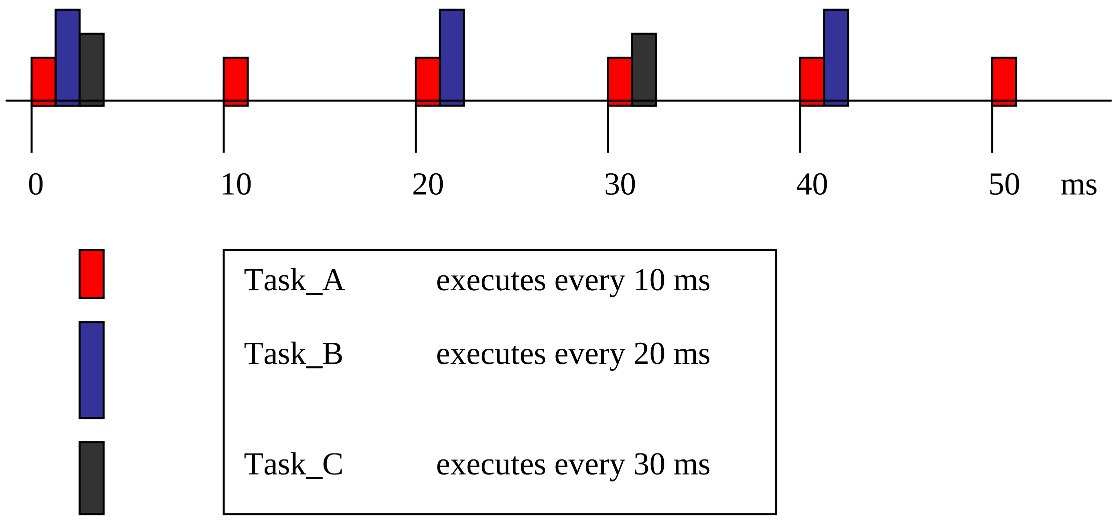

0
10
20
30
40
50
ms
Task_A executes every 10 ms
Task_B executes every 20 ms
Task_C executes every 30 ms

Figure 22: Execution of three cyclic tasks.

You see that any combination of Task_A Task_B and Task_C may be in turn of execution at some moments. The parameter execution order determines priority of execution when more that one task is to in turn of execution.
For example, Task_A with execution order of 0 has the highest priority, Task_B with execution orders of 1 has less priority and Task_C has the least priority of 3. Therefore in any moment when all these tasks are to be executed, their execution turn is determined from their execution order parameter.
In PLC500 Nseries with CPU180_X, execution order 0 has the highest and execution order 9 has the least priority.

Important Note:
Reading inputs from physical inputs is done before execution of the cyclic task with execution order 0.
Similarly writing outputs to the physical outputs is done after execution of cyclic task with execution order 0.
This is shown in figure 23.

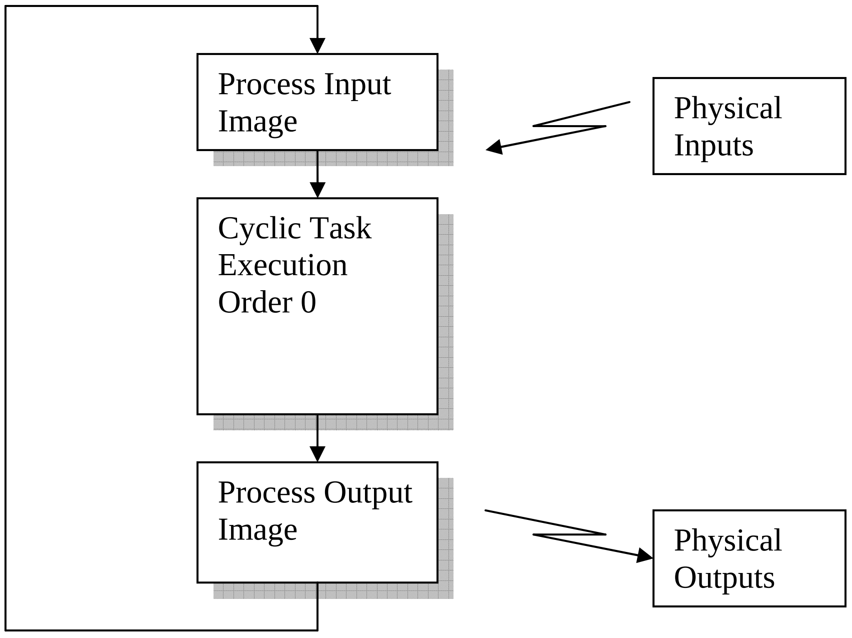


Figure 23: Execution of cyclic task with execution order 0 in synchronization with reading inputs and writing to outputs.

System Tasks in PLC500 Nseries, CPU180

CPU180-1 can support up to 6 system tasks. Among these 6 tasks only system tasks 0, 1 and 2 are available now and others are reserved for the future extensions.

SYTEM TASK0 (SPG0):	Runs once in Cold restart.
SYTEM TASK1 (SPG1):	Runs once in Warm restart.
SYTEM TASK2 (SPG2):	Runs when PLC is in STOP mode
SYTEM TASK3 (SPG3):	Runs when an I/O module is removed or fails
SYTEM TASK4 (SPG4):	reserved
SYTEM TASK5 (SPG5):	reserved

To insert a system task, right click on Tasks node in the “Project Tree Window” as shown in the two windows below;
Select SYSTEM as its Task type, give a name to it such as StartUp and press OK.

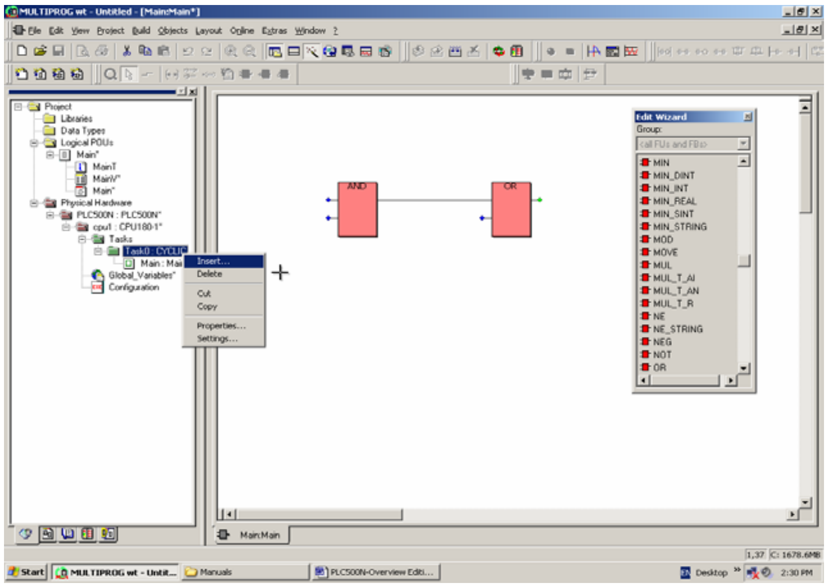
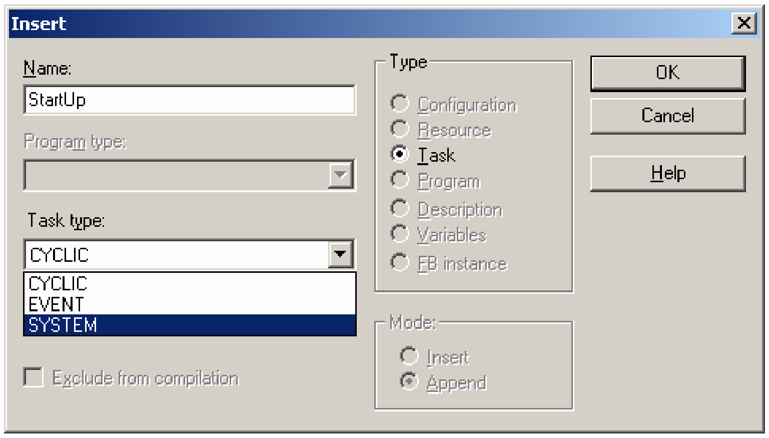


Figure 24: Inserting a new Task into the project


Insert

Name: StartUp

Program type:

Task type: CYCLIC

CYCLIC
EVENT
SYSTEM

Type: Configuration, Resource, [✓ Task], Program, Description, Variables, FB instance

Mode: Insert, [✓ Append]

Exclude from compilation

Figure 25: Choosing Name and type of task

The following dialog box appears. Select System Program Number 0 and press OK.

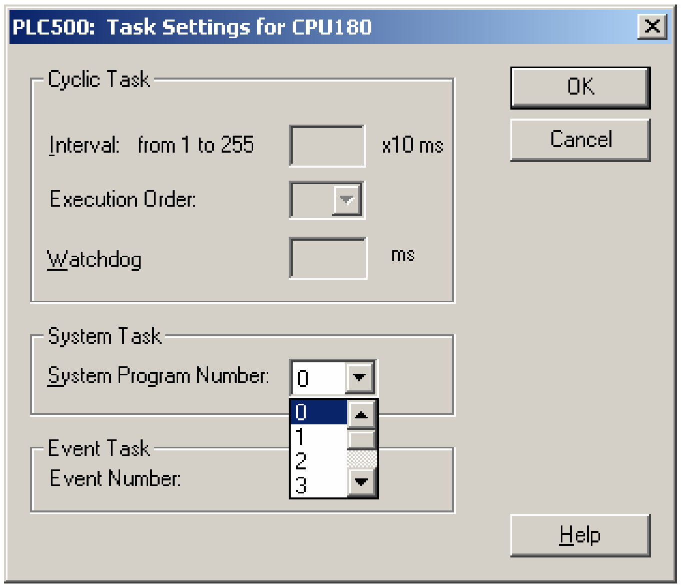


Figure 26: Selecting System task number

SYSTEM task StartUp will be inserted as shown in Project Tree Window. Add a POU that you want to be executed in cold restart.

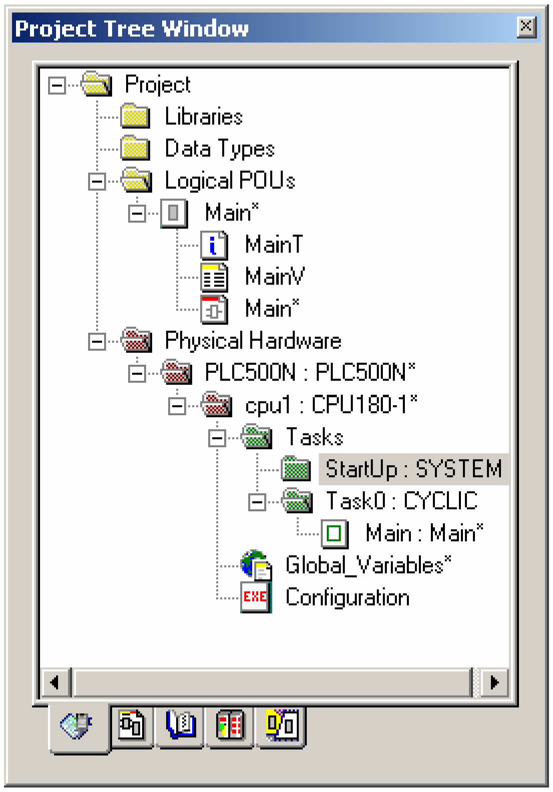


Figure 27: StartUp Task inserted into the project

You should now insert a POU into the StartUp task.

Event Tasks in PLC500 Nseries, CPU180

CPU180-1 can support up to 6 event tasks. All event tasks are reserved for the future extensions.

EVENT TASK0	Reserved
EVENT TASK1	Reserved
EVENT TASK2	Reserved
EVENT TASK3	Reserved
EVENT TASK4	Reserved
EVENT TASK5	Reserved

Retentive and Non-retentive variables
Retentive variables are variables whose values are stored even if the power is switched off. In the case of a warm start (normal power-up) the last value of the variable is going to be used.
Non-retentive variables are variables whose values are cleared (zeroed) in warm restarts.

For a discussion about Hot, Warm and Cold restarts please refer to the corresponding topic.

Initializing variables
According to IEC 1131-3 initial values can be assigned to variables. This means that a variable which is going to be used for the first time in the PLC program is used with its initial value. Initial values can be given to all kind of variables except in VAR_EXTERNAL declarations.
The initial value has to fit to the data type. It is not possible to use e.g. the data type BOOL and an initial value '5'. In this case the system displays an error message.

The initial value is optional. If no initial value is used, the variable is initialized with the default initial value of the data type (zero) or with the retained value in case of retentive variables.

In summary, normal non-retentive variables are initialized with initial values or zeros in both warm and cold restarts.
In case of retentive variables they are not changed upon warm restart and are initialized in cold restarts.

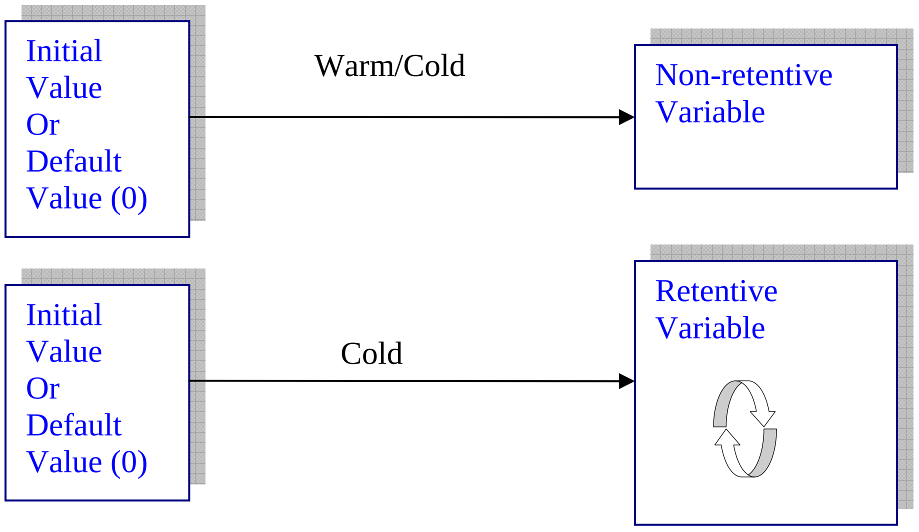

Initial Value Or Default Value (0)
Warm/Cold
Non-retentive Variable
Initial Value Or Default Value (0)
Cold
Retentive Variable

Figure 28: Variables in cold and warm restarts

Example:
Let MyInt is a retentive global variable of type INT and has an initial value of INT#1234.

```
VAR_GLOBAL   RETAIN
             MyINT    :        INT := 1234;
END_VAR
```

When global variable file is sent to the PLC, MyINT will be initialized by the value 1234. If your PLC program does not overwrite on it, its value will remain unchanged even if the PLC is switched off and on again (Warm Restart). However if your program changes this value, the new value will be available upon the next warm restart. In order to load the initial value again, a cold restart is necessary.

Similarly, local variables of a POU may be retentive or non-retentive, initialized or un-initialized. When a POU is sent to the PLC, it will be initialized with its initial value. If the variable is retentive, its value remains unchanged upon warm restart. It will be initialized again by the next cold restart.

Cold, Warm and Hot restarts

Cold restart of the PLC can be accomplished with or without using a programmer computer.
Cold restart using programmer computer can be simply done by pressing the Cold restart button in resource control dialog box as shown below.

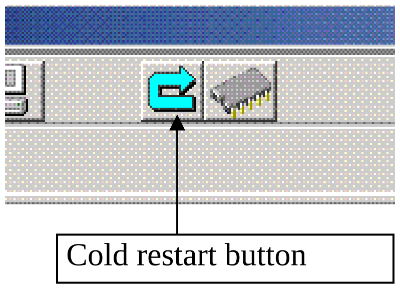


Figure 29: Cold restart Button

Cold restart can also be done in power up of the PLC with Run/Stop switch as will be discussed here. Place the switch in stop position and power up. The Red and Green LEDs will blink for about 5 seconds. If you do not change the switch in this time interval, a cold restart will take place and the Red LED will remain steady on. You can then change it to the run position. A complete cold restart routine is performed now.

Warm restart is nothing but a simple power up with the Run/Stop switch in Run position. The retentive variables will remain unchanged and the others will reset to zero.

Hot restart is changing mode of operation of the PLC from Run to Stop and vice versa. This action does not change values of the variables either retentive or non-retentive. Hot restart can be done with either Run/Stop switch on the CPU module or the Run and Stop buttons located on the resource control dialog box as shown below.

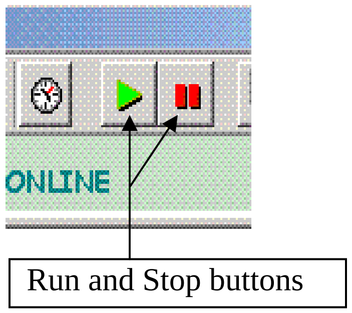


Figure 30: Run and Stop Buttons

Watch-Dog Mechanism

Executions of tasks are monitored by the operating system in order to make sure that every task is executed thoroughly at the correct time interval and no part of them are left unexecuted for the next cycle.
In ordinary multitask systems, this not important as far as it will be continued in other cycles.
In real-time industrial control systems that all tasks are related one another and every task is controlling some portion of a single process, integrity and consistency of the process variables are of critical importance.
In the other hand, a poorly designed program with large or even infinite loops may cause CPU being involved in large or infinite loops executing only portions of program and hence causing catastrophic events to the process.

That is why the operating system of a PLC must make sure that all parts of a single task are completely executed within the allocated time interval.

PLC500 Nseries uses a watch-dog mechanism to inhibit such situations.
The watch-dog mechanism works on the basis that every task must be done thoroughly from top to bottom in the time interval allocated for it. This means that if a task has not been completed in the prior cycle, will not be executed any more and the system will force the CPU to STOP mode for safety purposes and raises watch-dog error flag.

A *well organized program* is designed in such a way that fast changing process variables (inputs and outputs) are handled in fast tasks and slowly varying process variables are handled with slow tasks (by fast task we mean the tasks that have short execution time intervals). Normally, digital inputs should be handled fast and slow varying analog variables that are normally processed in control loops with considerable mathematical manipulations may be handled in slower tasks.
It must be taken into account that although you may distribute the tasks in different time intervals, there happens occasions that many or all the tasks must be executed in one time-slot. In other words the minimum time-slot should be large enough to let all the tasks be executed completely. Otherwise, exact time intervals are not guaranteed.

Let’s consider the following case:

Task Name	**Execution Order**	**Time-Interval mS**	**Required Time for Execution**
Task_A	0	10	7
Task_B	1	50	2
Task_C	2	100	4

In this example every 100mS all the three tasks are candidate to be executed. But the total execution time of them is 13mS (7+2+4). In this case the next execution of Task_A will be delayed by 3mS. As long as such situation can be tolerated there is no problem otherwise it must be corrected somehow.

Now consider another case that the tasks require more time for execution as shown in the following table.

Task Name	**Execution Order**	**Time-Interval mS**	**Required Time for Execution**
Task_A	0	10	7
Task_B	1	50	2
Task_C	2	100	13

In this case when all the tasks are being resumed, there happens a case that Task_A may not be called for 2 successive time slots. This will be identified by the system and will raise Watch-Dog error and STOPs the CPU.
To avoid such situations one should either break down the programs among other tasks or extend time interval of Task_A.

Although CPU180-1 is very fast and can run very large programs in few m-Seconds, in very large and intensive applications you may need to take some measures to let the system function more efficiently. Among the most severe cases are those applications that the CPU serves many HMI services, operating panels and the programmer computers simultaneously.
Please take into account that when on-line debugging of a POU is being conducted, execution time of a POU called in a task will increase by 30%. If all the POUs called in that task are being in debug-mode (of coarse this is a very rare situation), the total execution time of the task will increase by %30 and can trigger watch-dog timer if the time interval is not suitably adjusted.

As a rule of thumb, in order to avoid watch-dog mechanism being active, and have fast response to the external client devices, the execution time of a task should be less than 70% of its activation Time-Interval.

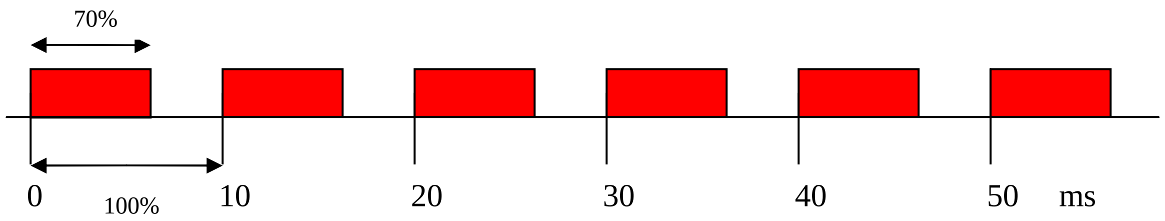


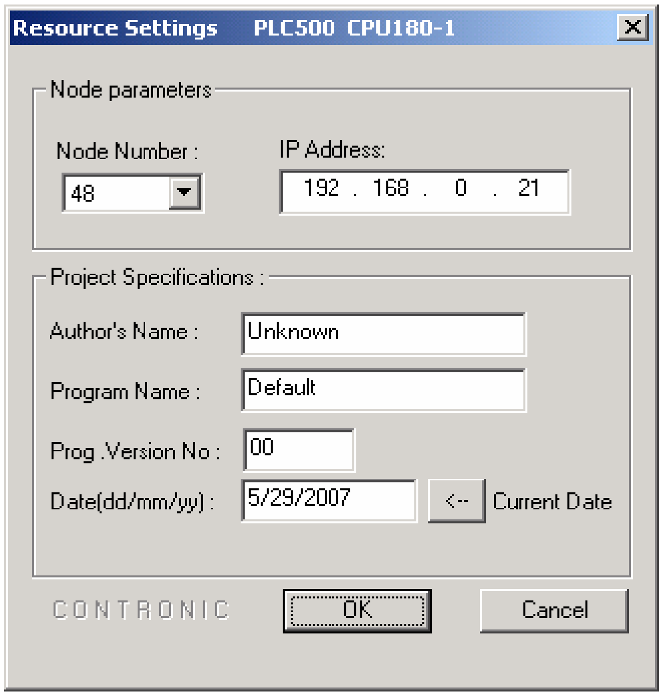
Figure 31: Execution time of a task should be less than 70% of its activation Time-Interval.

Resources in IEC 1131-3

A resource can be compared to a CPU which can be inserted into 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 IEC1131-3 a PLC may contain more than a single resource or CPU.

Resource settings in CPU180-X

Two parameters determine communication settings in CPU180-1. These parameters are "Node number" and "IP Address".
By default, Node number 48 and IP Address "192.168.000.021" are set in resource settings of your project.



Resource Settings PLC500 CPU180-1

Node parameters

Node Number :	IP Address:
48	192 . 168 . 0 . 21

Project Specifications :

Author's Name :	Unknown
Program Name :	Default
Prog .Version No :	00
Date(dd/mm/yy) :	5/29/2007

Figure 32: Resource setting dialog box

Four other parameters are optional project specifications. It is a good habit to fill these parameters carefully.
When you click on OK button, these settings will be saved in a file called "Settings" and the dialog box will be closed. Later you will see how you can send this file to PLC.

Configuration of PLC500 Nseries

Hardware of PLC is determined by introduction of I/O modules into the rack of PLC. This is done by double clicking on Configuration icon in project tree window of MWT as shown below.

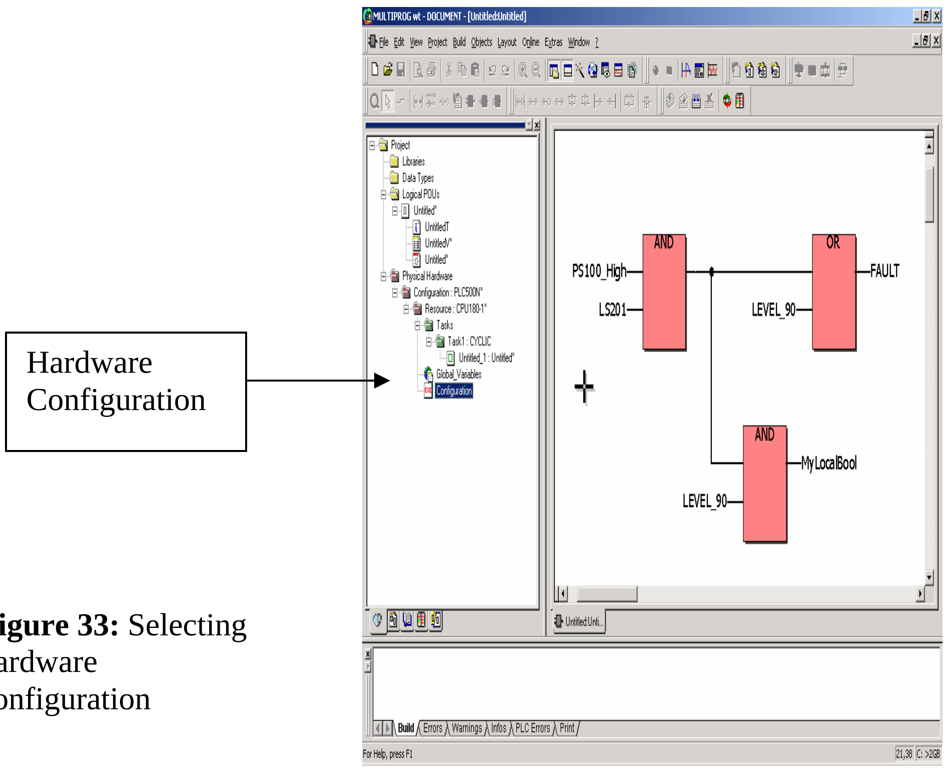
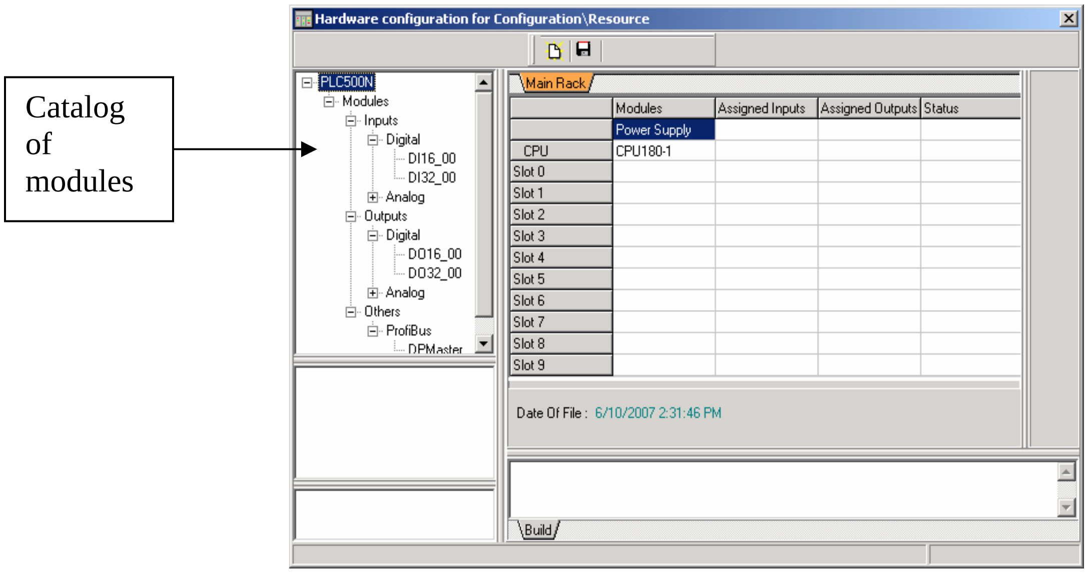


Figure 33: Selecting hardware configuration

The next dialog box (Hardware Configuration dialog box) will be shown. In the project created, no modules are added to the rack of PLC. So you have to populate the rack with I/O modules by yourself.
Depending on your actual hardware, you have to insert corresponding modules from the catalog of modules located on the left side of the dialog box.
Populating the rack is very simple. Just position the mouse on the module, hold down the left key of the mouse, drag it to the rack slot and release the key. The selected module will be dropped there.
Every module reserves a specific amount of addresses on input, output or both of them in I/O area of CPU. For example if two successive slots 0 and 1 are populated with DI16-00 and DI32-00, input addresses IB0 and IB1 will be reserved for DI16-00 and IB2, IB3, IB4 and IB5 for DI32-00. Any other input module in the next slots will be granted addresses from IB6 and above.



Hardware configuration for Configuration\Resource

- PLC500N
  - Modules
    - Inputs
      - Digital
        - DI16_00
        - DI32_00
      - Analog
    - Outputs
      - Digital
        - DO16_00
        - DO32_00
      - Analog
    - Others
      - ProfiBus
        - DPMaster

Main Rack

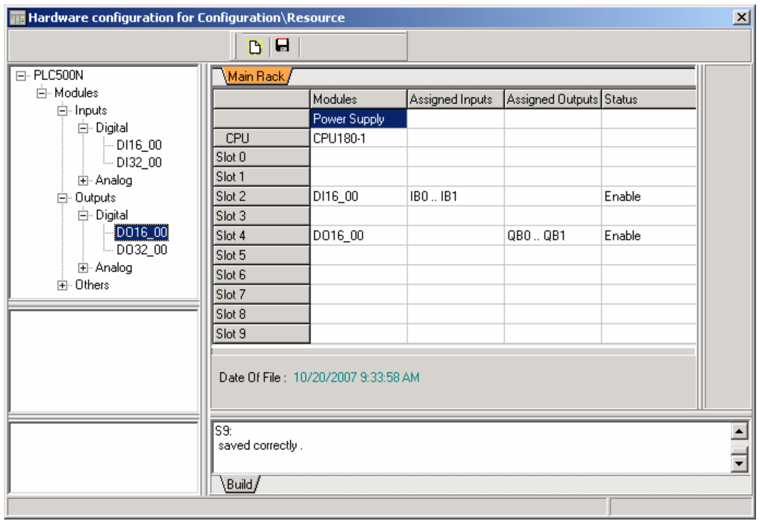
	Modules	Assigned Inputs	Assigned Outputs	Status
	Power Supply			
CPU	CPU180-1			
Slot 0				
Slot 1				
Slot 2				
Slot 3				
Slot 4				
Slot 5				
Slot 6				
Slot 7				
Slot 8				
Slot 9				

Date Of File : 6/10/2007 2:31:46 PM

Build

Figure 34: Hardware configuration dialog box

The same addressing concept applies for output modules.
Please complete the hardware configuration table above and save it. A hardware configuration file called "HWConfig" will be created. You will soon see how you can transfer this file to the CPU.
Let's assume that you have inserted one input module of type DI16-00 and one output module of type DO16-00 into the rack at slots 2 and 4. The hardware configuration table will be as shown below.



Hardware configuration for Configuration\Resource

- PLC500N
  - Modules
    - Inputs
      - Digital
        - DI16_00
        - DI32_00
      - Analog
    - Outputs
      - Digital
        - DO16_00
        - DO32_00
      - Analog
    - Others

Main Rack

	Modules	Assigned Inputs	Assigned Outputs	Status
	Power Supply			
CPU	CPU180-1			
Slot 0				
Slot 1				
Slot 2	DI16_00	IB0 .. IB1		Enable
Slot 3				
Slot 4	DO16_00		QB0 .. QB1	Enable
Slot 5				
Slot 6				
Slot 7				
Slot 8				
Slot 9				

Date Of File : 10/20/2007 9:33:58 AM

```
S9:
 saved correctly .
```

Build

Figure 35: Hardware configuration dialog box

The fourth column in hardware configuration table is “Status” column. You can enable or disable modules inserted into the slots. Any module enabled will be checked for conformity and presence in PLC startup.
As you may have already noticed, module addresses are automatically set and placed in “Assigned inputs” and “Assigned outputs” columns. This is true even if a module is disabled.
If a module is disabled the CPU assumes it as a dummy module and skips working with it. As a matter of fact the disabled modules may not even be inserted into that slot. You may also use such a skim in order to intervene the automatic addressing of modules.

In near future when Contronic releases its I/O simulator software, you may disable all physical input and output modules and apply these signals via the I/O simulator software. This will provide a tool that you can use to debug your applications without any physical inputs or outputs and the associated wirings.
Afterwards, when your application is fully developed and debugged you may do the wirings and enable modules in hardware configuration table.

Projects in Programmer Computer

By default, projects will be located at PROJECTS folder of installation path of MWT. Every project has a file and a folder with the same name as the project's name. For example the project SHORT has a file SHORT.MWT and a folder SHORT. You can see some projects in figure 36.

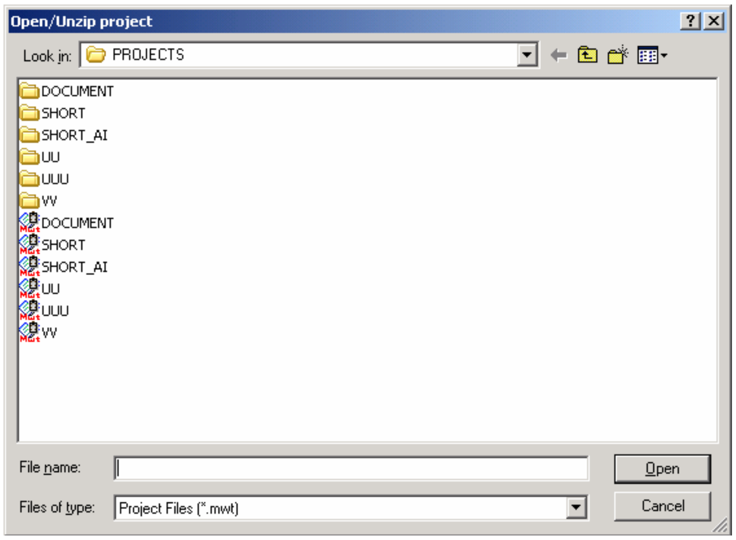


Figure 36: Project folder of MWT

To open a project in MWT you have to open its "name.mwt" file. Normally you are not concerned about its accompanied folder.
You may also save your project in PROJECT folder of MWT in any of the two formats of ordinary mwt format or in zipped or compressed zwt format. To do this select File in menu bar of MWT and then select *Save Project As /Zip Project As…*as shown in figure 37.

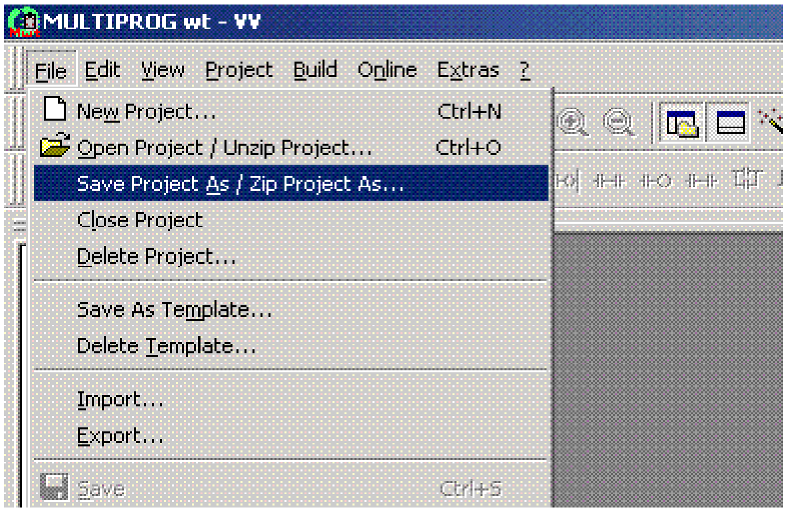
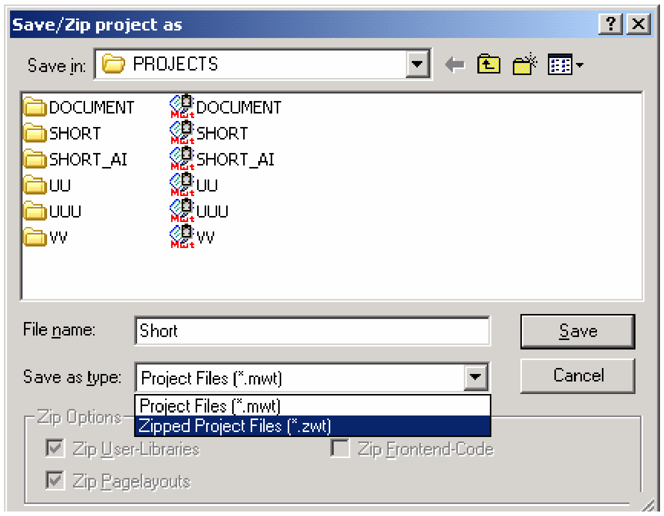


Figure 37: Saving a project in *.mwt or *.zwt format.

The dialog box shown in figure 38 will appear. Give it a name and select the mwt or zwt format for its extension.



File name: Short

Save as type: Project Files (*.mwt)

Figure 38: Select format of save for the project

In most of the cases when you are developing your project you only deal with mwt format of your project.
Finally, when your project is fully debugged you may want to transmit the zipped version of your project to the PLC. If your PLC has a copy of your zipped project file, you can upload it from the PLC. Then MWT can unzip and make a complete copy of your project including all the details of your project.
In case of any problem with your project you may also send the zipped copy of your project to Contronic Company and ask for help.

Communication with PLC

Communication with PLC500 Nseries is done via communication ports of CPU180 directly.
CPU180-1 has two ports dedicated for communication with the programmer computer, one serial RS232 (Optionally RS485) and one Ethernet port. Communication with PLC may be handled via any of these two ports.
Since PLC500 Nseries are designed to work in networks, any PLC must have a unique address through which connections be established. This is true even if your PLC is not to be connected to any network and you want to have a direct link to it.
PLC500 Nseries support two types of networks, RS485 and Ethernet as shown below.

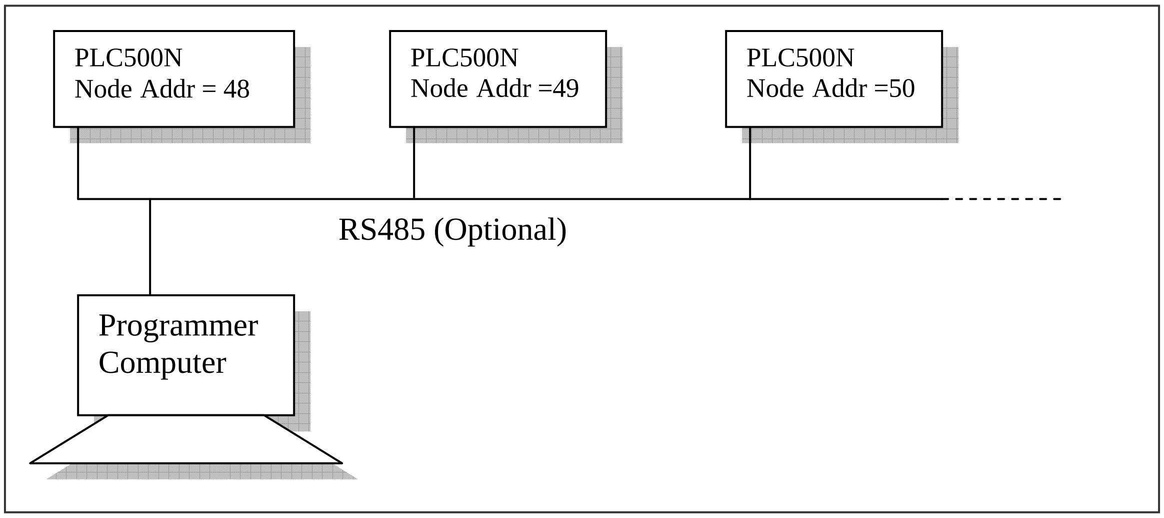


Figure 39: PLCs connected in a local RS485 network.

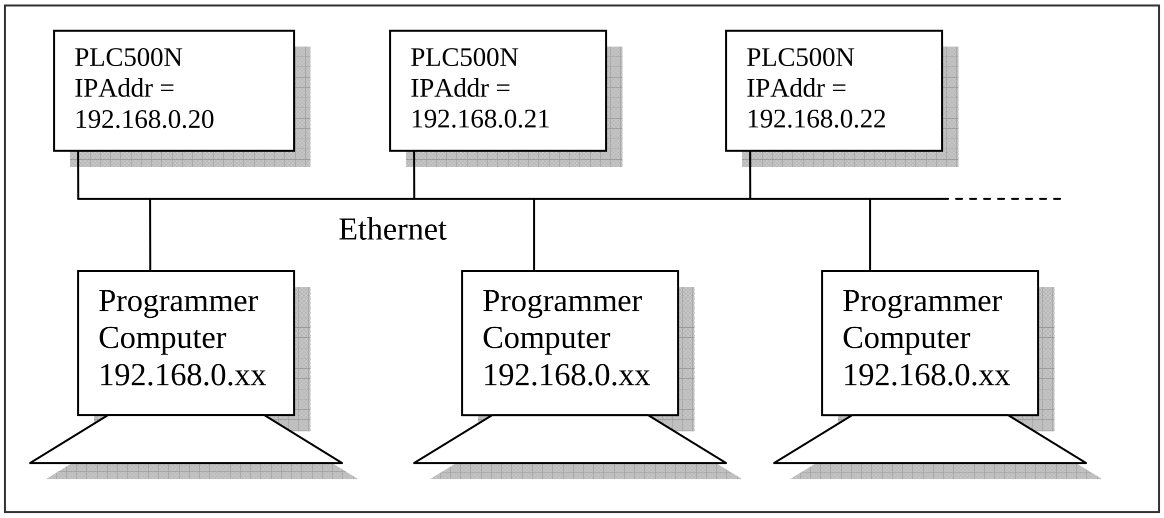


Figure 40: PLCs connected in an Ethernet network

As shown in figure 39, when the PLCs are in an RS485 network, a single Programmer computer can program a number of PLCs. The same thing is true if a single computer supports HMI services.

As shown in figure 40, In Ethernet networks, one or more programmer computers may be connected to an Ethernet LAN in which multiple PLCs are present with virtually no restrictions. In order to show the networking concepts, Ethernet switches/hubs are not shown in figure 40. In practice, all these devices must be connected to an Ethernet switch/hub.

Direct connection to PLC

Direct communication with PLC can be established in two ways:

- RS232 Serial port through a special serial cable (*order number* RS232_180_4M)
- Ethernet port through a cross Ethernet cable (*order number* ETH_CR_4M)

When serial cable is used, a node number (from 2 to 56) for CPU must be agreed upon between the PLC and programmer computer. Default Node-Number is *48*.

When Ethernet cable is used an IP-Address must be assigned. This address has the same role that Node address has in serial communication. Default IP-Address is *192.168.000.020*

Please note that Ethernet cable for direct connection to PLC is different from that of network cable. In network connection through Ethernet switch/hub, Straight Ethernet Cable (order number ETH-ST-4M) must be used.

When a total memory clear procedure (discussed later) is done, default Node-Number and IP-Address will be assigned for communication. When you create a new project in MWT, the same Node-Number and IP-Addresses will be generated so that you will have no problem in direct connection to the PLC.

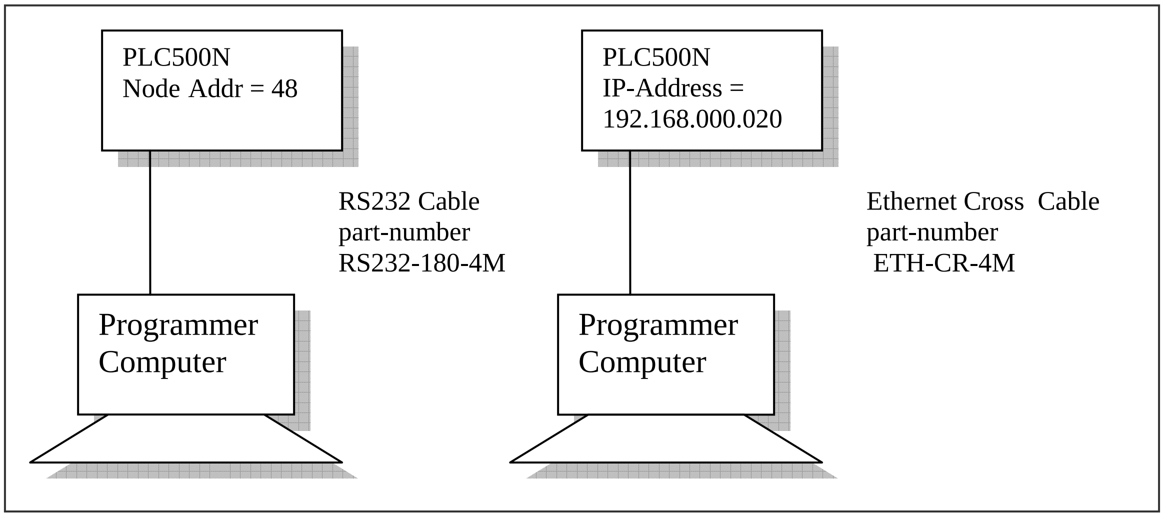


Figure 41: direct connection of computers to the PLC via serial and Ethernet cross cables

Total Memory Clear Procedure

In some occasions, you may need to clear all RAM memory of the CPU. This can be done directly via the switch RUN/STOP located on the CPU module. With such an approach a Programmer Computer is not needed.

1. Put the RUN/STOP switch in STOP position.
2. Power up the PLC.
3. The two top most LEDs (green Run LED and red Stop LED) start blinking for about 5 seconds.
4. In the 5-Second blinking time interval, an internal counter starts and counts number of times you toggle RUN/STOP switch from stop to run positions. Speed of toggling has no importance
5. If number of toggles is more than 2, a total memory clear will be done. All POUs, Tasks and Global variables will be erased and default values for Node and IP addresses will be loaded.

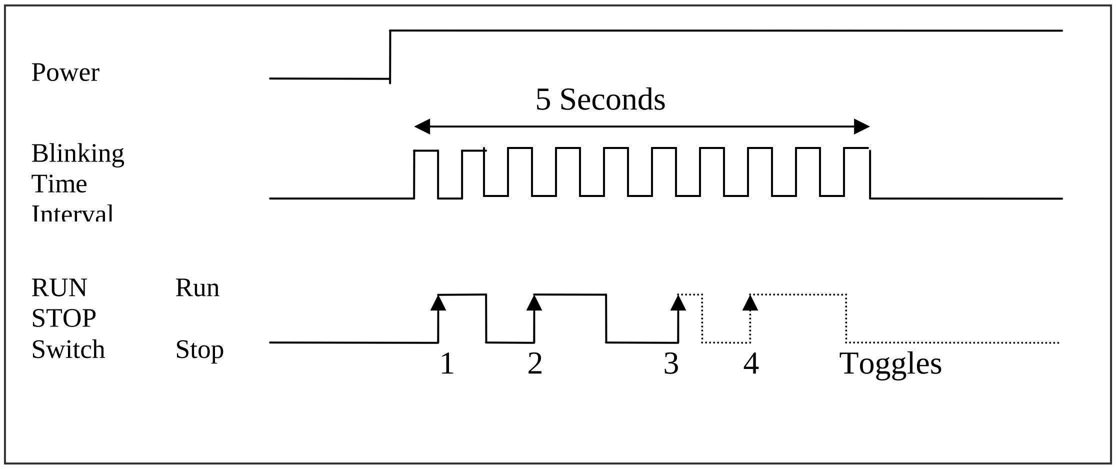


Figure 42: If number of toggles is bigger that 2 (3 and above) all RAM memory will be erased.
If number of toggles is less than 2 (0 or 1) nothing happens and PLC makes transition to Run or Stop mode depending on the last position of the key.

Read Back from Flash Memory Procedure

If a copy of program is already saved (fixed) in flash memory of the CPU, it can be copied back to RAM as described below.

Follow the 5 steps described previously, regarding total memory clear procedure, with the exact number of toggles of 2. All contents of flash memory will be copied back to the RAM.

Caution: It is strongly recommended to make sure that flash memory contains a valid and fully debugged program suitable for the process connected to your PLC. Catastrophic events may occur otherwise. This can be checked by downloading the program from flash memory and inspecting its contents as described later

Control Dialog Box

All transactions with the CPU takes place from “Control Dialog Box”. This dialog box will appear when one clicks on corresponding button in toolbar of MWT as shown below.

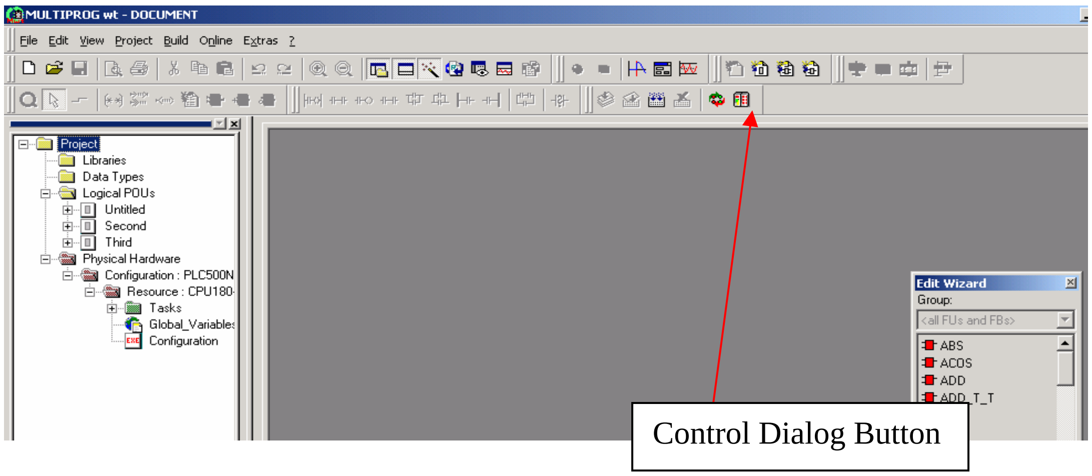


Figure 43: Activating Control Dialog Box.

Control dialog box is the most important interface through which one can download or upload programs, tasks, hardware configuration and etc. to and from PLC, besides checking online status of the CPU.
When the control dialog box is going to appear, it checks the contents of your project in the programmer computer and also contents of the CPU. The control dialog box has two main parts. The left side shows your project residing in hard disk of computer and the right side current contents of the CPU.
If connection to the PLC cannot be established the right side of control dialog box will be dark and most of the buttons on its toolbar, except 3 buttons, will be disabled as shown in figure 44.

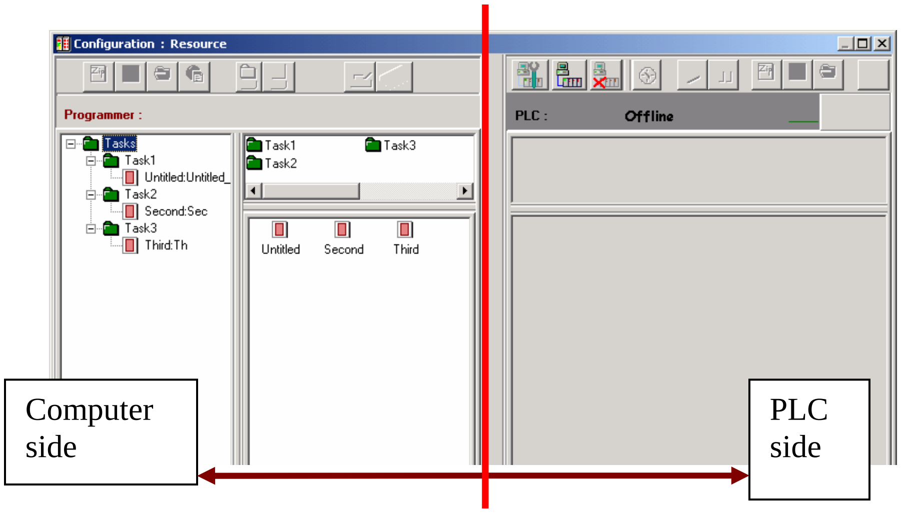


Figure 44: Control Dialog Box is offline and the right PLC side is dark

The three buttons still enabled on the PLC side are shown in figure 45 below.

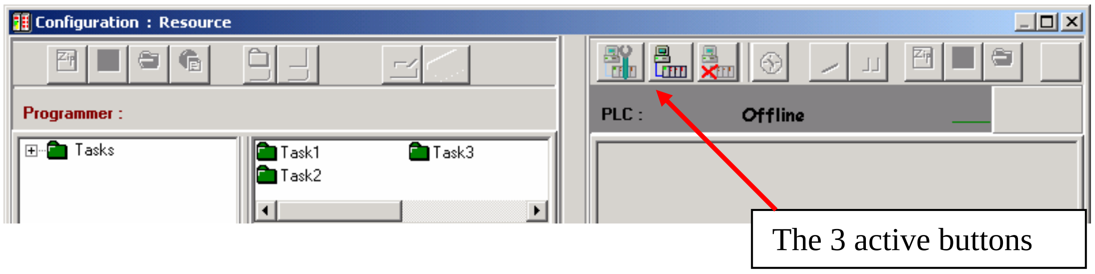


Figure 45: three buttons are always active in PLC side

These buttons help establish communication, interrupt connection and change settings of communication parameters.

Communication Settings button opens the corresponding dialog box shown in figure 46.

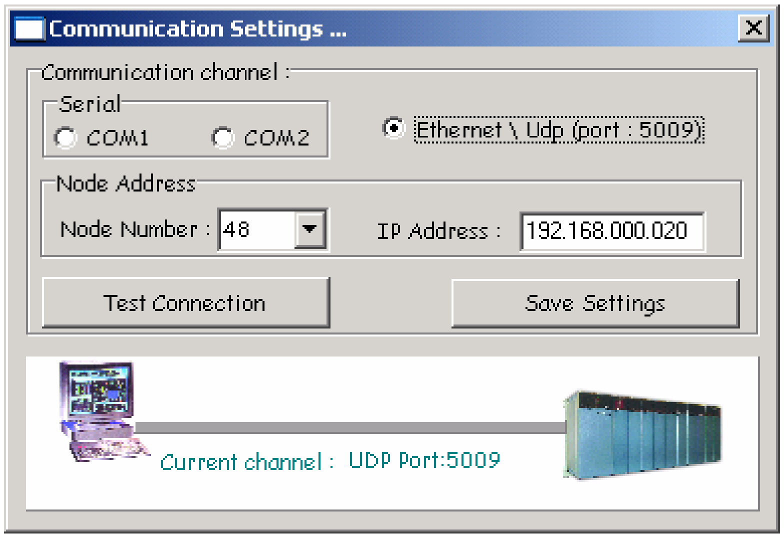

Communication Settings ...
Communication channel :
Serial
COM1
COM2
Ethernet \ Udp (port : 5009)
Node Address
Node Number : 48
IP Address : 192.168.000.020
Test Connection
Save Settings
Current channel : UDP Port:5009

Figure 46: dialog box Communication Settings

Here you set the Node Number and IP Address suitable for connection to the PLC.
By default these addresses are 48 and 192.168.000.020.
When serial communication is selected, serial cable must be connected between computer and the PLC.
If Ethernet is selected, either cross Ethernet cable or straight Ethernet cable must be used as discussed before.
Select your settings and close the dialog box. You will be prompted if you want to save the settings.

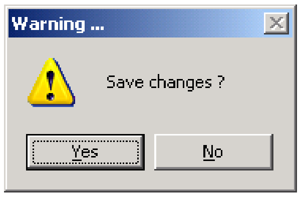


Figure 47: Saving Communication Settings.

If you save the settings, communication to the PLC will take place with these new settings throughout this project. In other words, every project has a specific communication parameter set the same way every PLC has a specific parameter set.

If a matching parameter set is agreed upon, PLC side of Control dialog box will turn on visible as shown in figure 48.

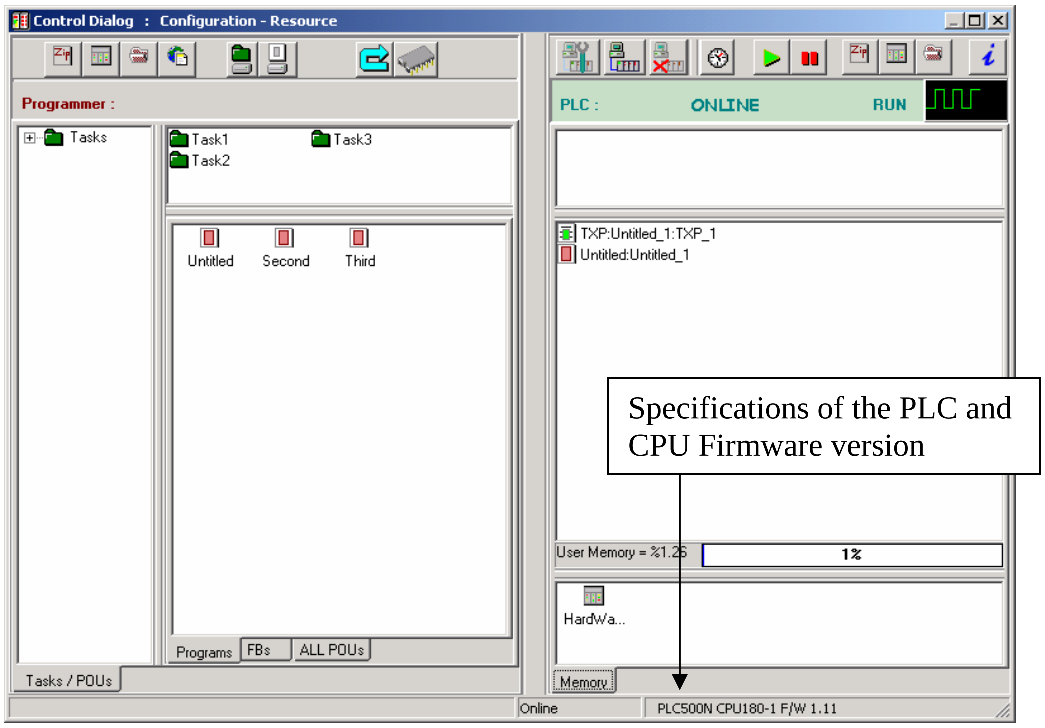


Figure 48: the PLC side of Control dialog box will be visible if communication parameters are set correctly.

The button Disconnect interrupts and the button Reconnect reconnects your computer to the PLC.

Note: if you are using Ethernet communication, your computer must be set with a static IP Address suitable for the LAN. Suitable IP address means that IP address of both your computer and the PLC must be identical in the first three parts. For example use 192.168.000.20 and 192.168.000.11 for the PLC and Computer respectively. You can set IP address of your computer via the Control panel of windows as described in appendix A.

Once connected, you are able to transmit your programs and files to the PLC.

Before trying to transmit your programs to the PLC, please make sure that your project is compiled with no errors in MWT.

Programs and files to be sent to the PLC.

Following types of files can be transmitted to the PLC where some of them are optional and one of them is better to be transferred at the last stage of project development.

- Hardware Configuration file (HardWareConfig)
- Program Organization Units (POUs)
- Task files (Tasks)
- Zipped project file (ZippedProject)
- Resource Configuration file (ResConfig)–*optional*
- Global variables file (GlobalVars)-*optional*

The optional *ResConfig* file is necessary when your PLCs are to be connected in a LAN where every PLC must be uniquely addressed.
The optional *GlobalVars* file is needed where some of the global variables have initial values and/or are of RETAIN property.
The *ZippedProject* file must be transmitted to the PLC when your project is fully developed, debugged and is to be fixed in flash memory.

Other files are absolutely necessary and must be transmitted in order to be used by the PLC.

Sending Programs and files to the PLC.

You can send *ZippedProject*, *HardWareConfig, ResConfig* and *GlobalVars* files to the PLC by clicking on the four buttons located at top-left corner of Control dialog box as shown in figure 49.

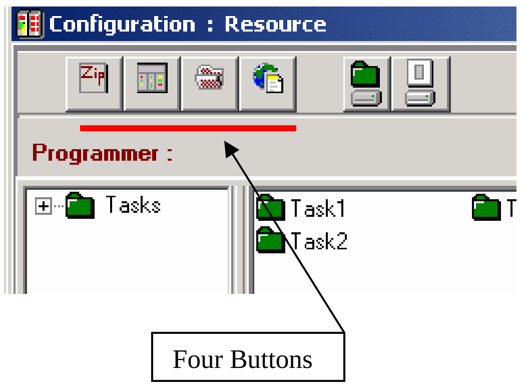


Figure 49: use these four buttons to send ZippedProject, HardWareConfig, ResConfig and GlobalVars to PLC

Hardware configuration file may be transmitted with no restrictions. If HardWareConfig file matches the installed hardware of the PLC, the yellow hardware error LED on the CPU module turns off indicating that configuration is ok.

To send other three types of files please take into account the recommendations of the preceding topic.

Before attempting to transmit Zipped Project file to the PLC, you have to save your project in zipped format (*.zwt).
You may send ResConfig file the same way. Since communication parameters will take effect at PLC power up, sending ResConfig file will be followed by an automatic restart of the PLC.
Please note that if communication parameter set changes on the PLC-side, you may lose online communication with the PLC because you have changed them purposely. Therefore you have to readjust communication settings of your computer accordingly. using the button Communication Settings.

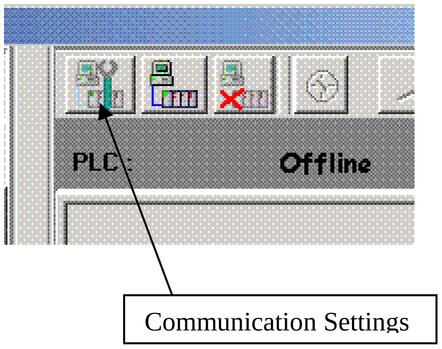
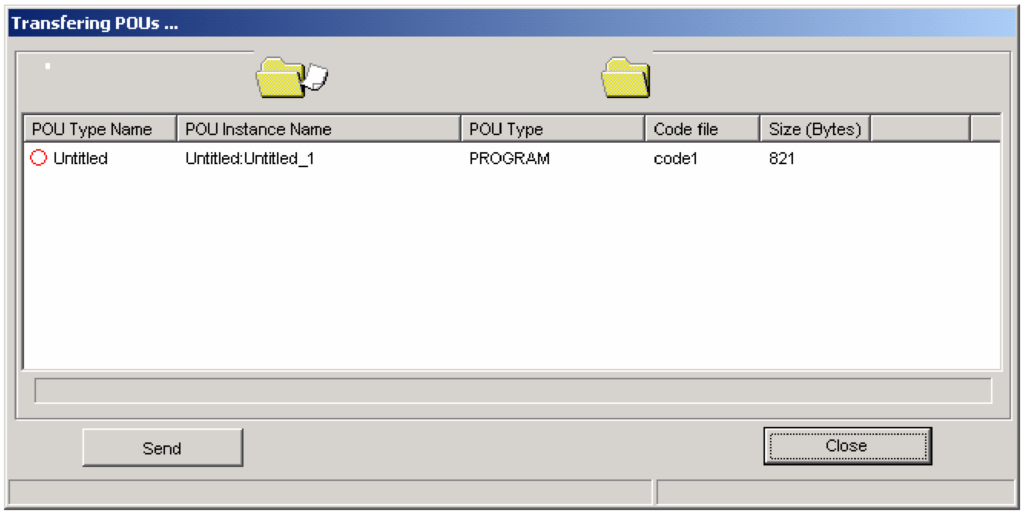


Figure 50: the three buttons dedicated for communication with the PLC.

Sending POUs and tasks is simply done by familiar windows drag and drop method. Click and hold down the left button of the mouse on the POU to be sent and drag it to the corresponding PLC-side window. Release the left button, transmission dialog box appears. Press send button to start transmission. Press Close button to close the window.



POU Type Name	POU Instance Name	POU Type	Code file	Size (Bytes)
Untitled	Untitled:Untitled_1	PROGRAM	code1	821

Figure 51: Dialog box for sending POUs to the PLC

Please note that tasks may only be sent to the PLC when their called POUs are sent before. Otherwise a dialog box prompts you to do that before.

Debugging your application programs

After you have sent your POUs and their calling tasks to the CPU, you may debug them in online sessions.
Click on the button “Debug”. A green bar appears on bottom line of MWT indicating that online communication is established between programmer computer and the PLC.

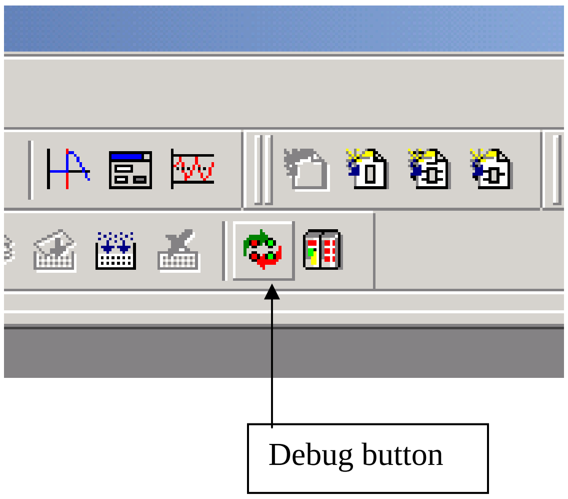


Figure 52: Debug button helps online communication with the PLC

Now open worksheet of your POU to be debugged. Following figure shows a sample online session.

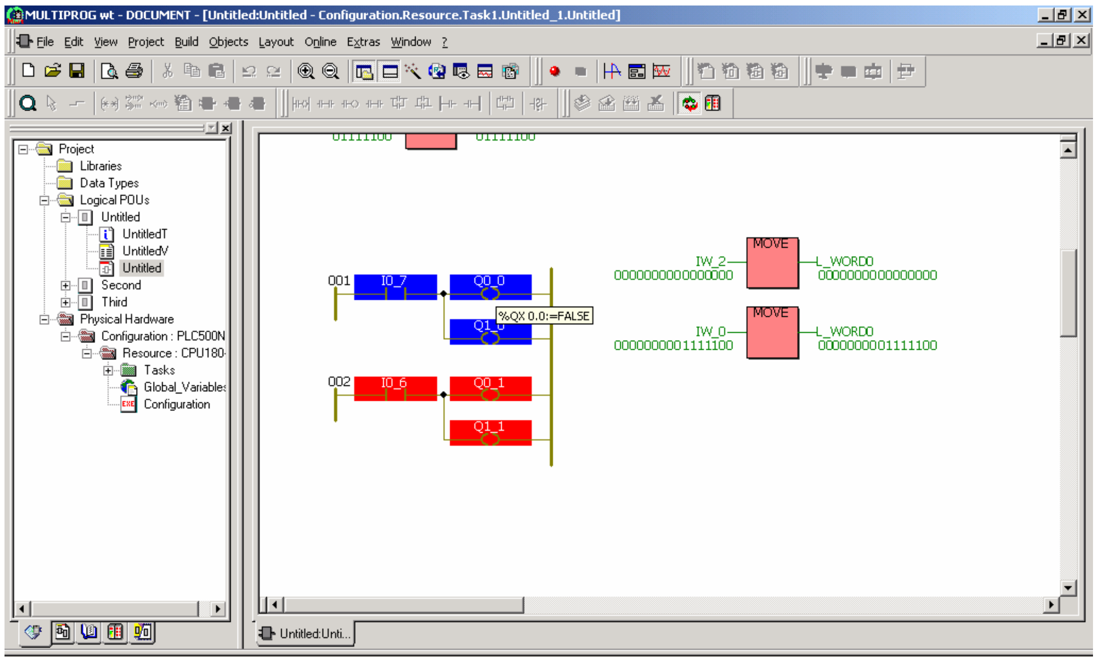


Figure 53: A sample online session in MWT.

You may add more POUs by opening their worksheets. Select windows from the menu toolbar of MWT and select the best views for your POUs. For further possibilities please refer to the manual of MULTIPROG.

Forcing variables

Some variables such as global variables, inputs and outputs can be forced. If these variables are not overwritten by user programs, one can set them appropriately.

While in debug mode, position the mouse on a variable and right click. A pull down selection menu appears. Select “Debug dialog…”. A suitable force dialog box for each type of variable appears. Change the variable and press click the force button.

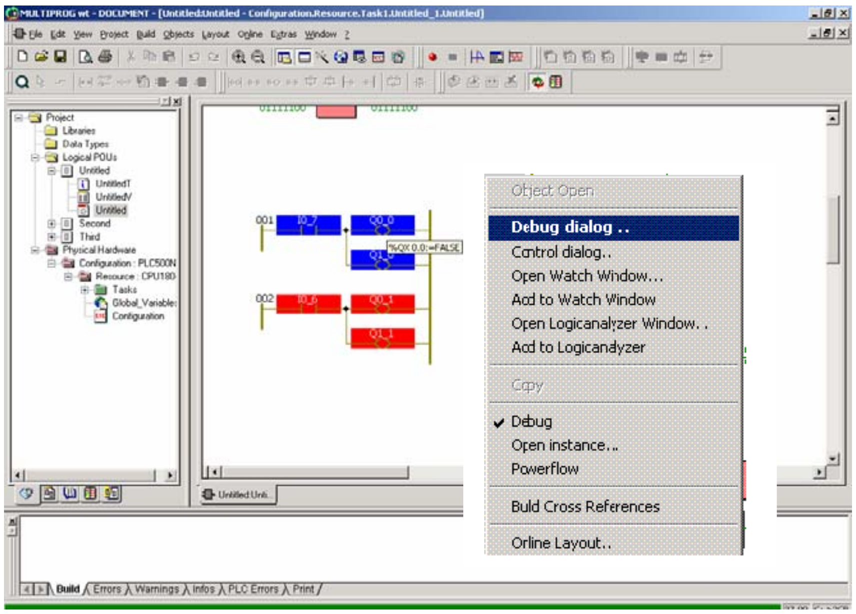


Figure 54: Variables may be forced while in debug mode.

Forcing capability can be of great help in developing user programs without doing actual I/O wirings to a PLC. For this purpose disable input modules in hardware configuration and send it to the PLC. When disabled, I/O modules are not scanned by the CPU thus you can easily force them in debug mode and test your program.
Most of your control program can be developed in this way without waiting for the actual wirings being done.

User Functions (FU) and Function Blocks (FB) in PLC500 Nseries

PLC500 Nseries have several firmware functions and function blocks that you may freely use in your applications with no limitations. Beside these blocks, depending on your particular application you may need to write your own functions or function blocks in order to perform some specific action that is repetitive in nature and used in many different parts of your project. Such functions or function blocks can be programmed once and used frequently.

When you develop a Function Block, you actually develop a model, a structure or a behavior that is being used frequently in your applications. That's why we sometimes say that a function block encapsulates a behavior relating inputs, outputs and its history.

Now it is clear that a function block must be defined with some inputs and outputs that only their data types are clear in development phase but their locations are not known. When you invoke them in your applications you specify their input and output locations.

With the same token, when you call a function or function block, an instance (or copy) of that block will be generated that must be subsequently sent to the PLC. These blocks are self sufficient and work independently. In case of Function blocks they also have a dedicated memory block that is totally independent from others. A function block receives its block of memory from the CPU automatically.

The same way that a PROGRAM has a local memory region, a FUNCTION BLOCK also has one. In addition to local memory, a function block has two more memory regions for its inputs and outputs. This is shown in the following figure.

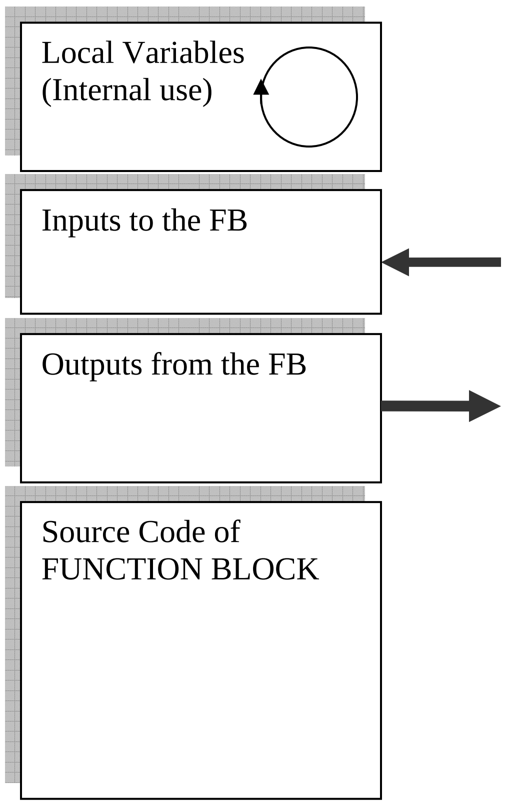


Figure 55: structure of a Function Block

When you write a function block, it will appear in edit wizard of MWT in green color. Then you simply use it in your applications the same way you do with firmware FBs.

Design of a sample Function Block

Let's you need some sort of a timer that is not directly supported by IEC1131-3 standard timers. Then you have to design it as your own user function block.

Let's define its behavior as follows:

- It starts with a rising edge on its input
- Duration of input has no effect on its timing.
- Its output remains high for a definite duration of time
- Its timing extends if there are multiple transitions on its input terminal.
- Its name is extended pulse timer- TXP.

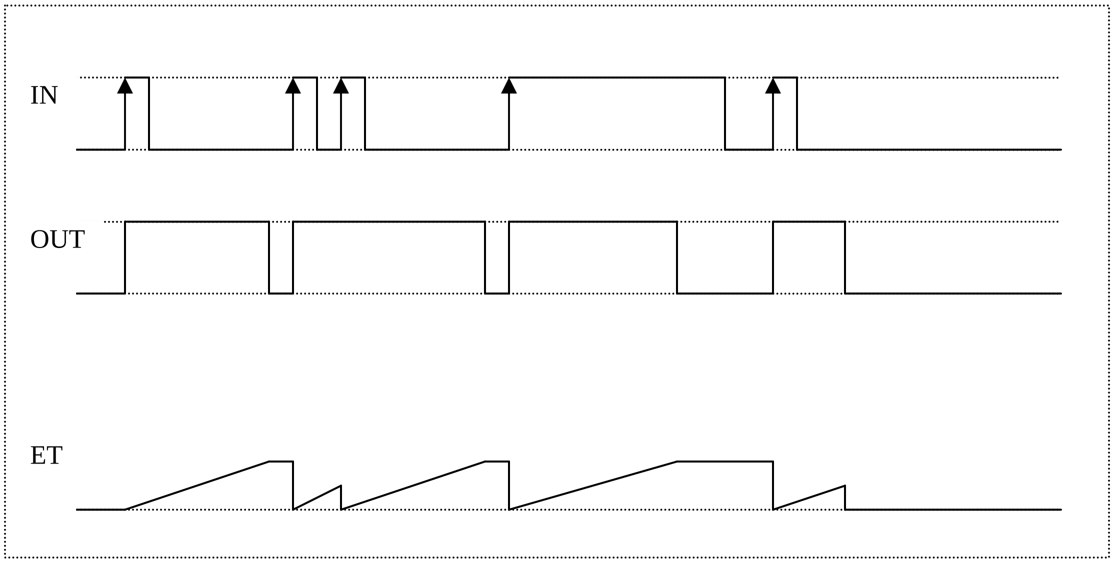


Figure 56: Time diagram of a user developed function block TXP.

Follow the steps below to develop it in MWT.

1. Insert a logical POU as you already know

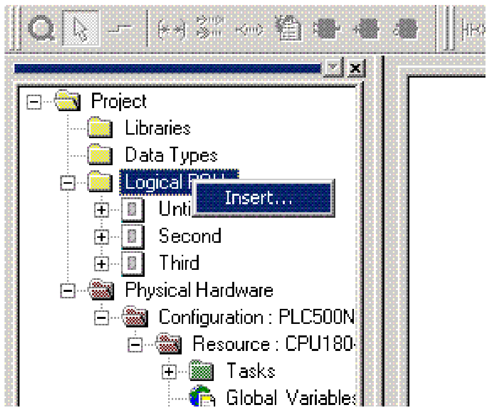
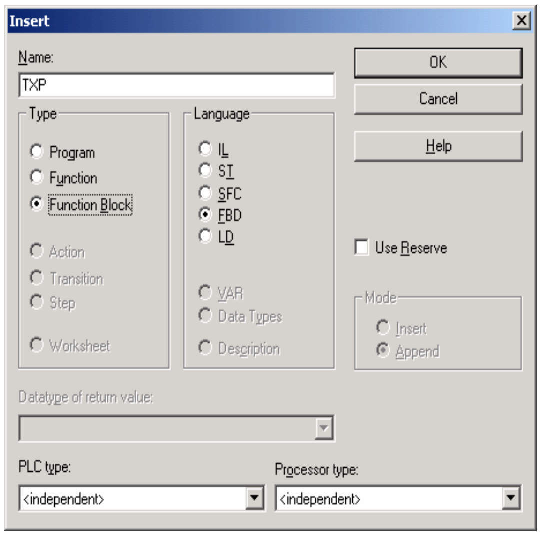


Figure 57: Inserting a new logical POU

2. Choose TXP as its name and Function Block as its type. Select FBD language and press ok to close the window.



Name: TXP

Type: ( ) Program ( ) Function [✓ Function Block] ( ) Action ( ) Transition ( ) Step ( ) Worksheet

Language: ( ) IL ( ) ST ( ) SFC [✓ FBD] ( ) LD ( ) VAR ( ) Data Types ( ) Description

PLC type: <independent>

Processor type: <independent>

Figure 58: Filling specifications of the POU

3. Open the worksheet of TXP to edit. Place a TOF timer, an R_TRIG and make connections as shown below.

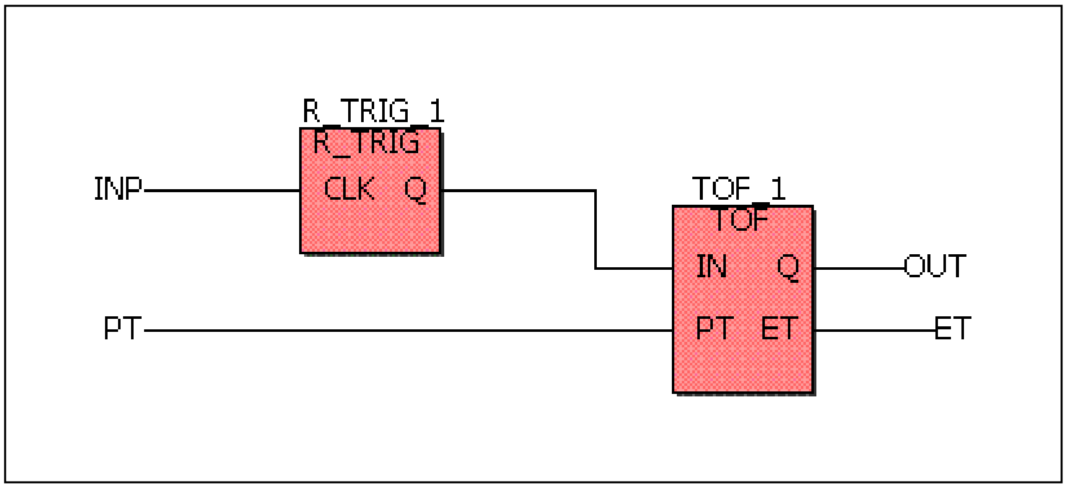


Figure 59: basic model of function block TXP.

4. Now you have to introduce inputs and outputs of TXP. Double click on IN input of TP_1 the Variable dialog box appears as shown in figure 60. Since this variable is a candidate for being an input to the function block, select a

short suitable name for it such as INP and select Local Scope for it.

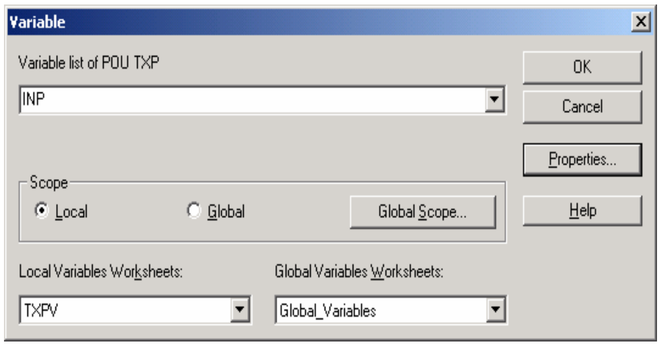


Figure 60: Variable dialog box

Press ok to go to the next dialog box.

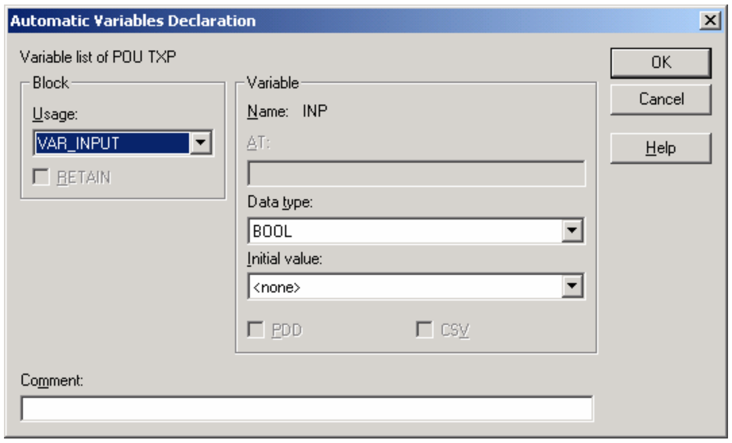


Figure 61: Variable declaration dialog box

Select VAR_INPUT as its Usage and data type of BOOL with no initial value. ‘s That’s all about this variable.

5. Define other input and output variables the same way as listed in following table.

Name	**Scope**	**Data Type**	**Usage**
PT	Local	TIME	VAR_INPUT
IN	Local	BOOL	VAR_INPUT
OUT	Local	BOOL	VAR_OUPUT
ET	Local	TIME	VAR_OUPUT

6. Have a look into the Edit Wizard. You will see that TXP is now added to the edit wizard in green color.

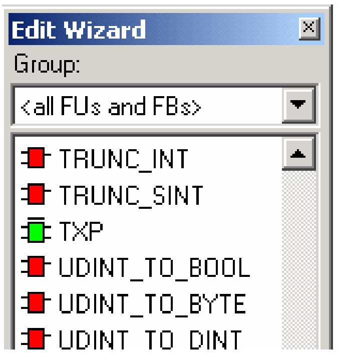


Figure 62: Your FB is included to Edit Wizard

7. .Add an instance of TXP into the worksheet Untited. You will see that your FB will be displayed as if it is a firmware FB but in color green.

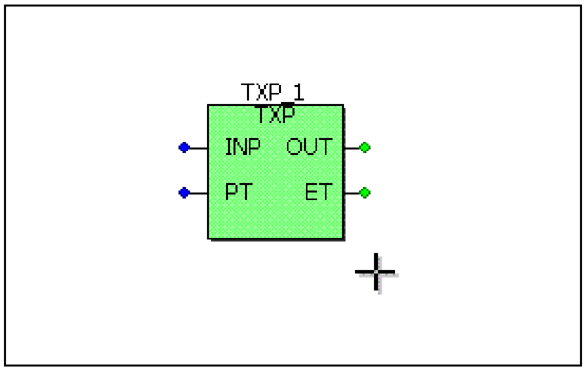


Figure 63: Placing an instance of TXP into the worksheet

8. Double click on inputs and outputs of the function block TXP and insert variables as shown in the following table.

Variable	**Scope**	**Data Type**	**Connection**	**AT**
Start	Global	BOOL	INP	%IX1.0
TIME#5S	Local	TIME	PT	
Output	Global	BOOL	OUT	%QX1.0
ElapsedTime	Local	TIME	ET	

9. Now all connections are complete and you can debug your FB. First compile your project by F9. If compilation is successful, send the POU containing an instance of TXP to the PLC and check its operation.

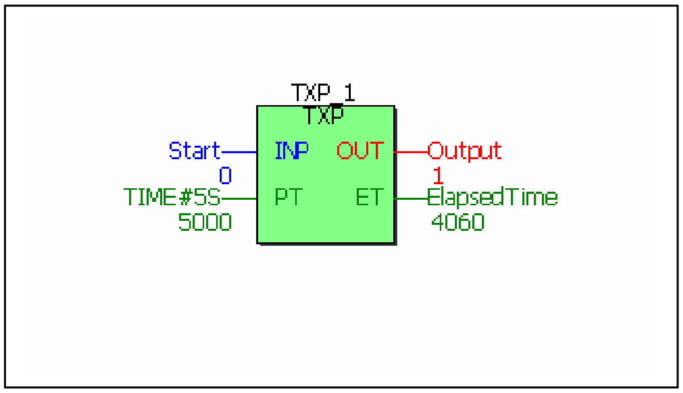


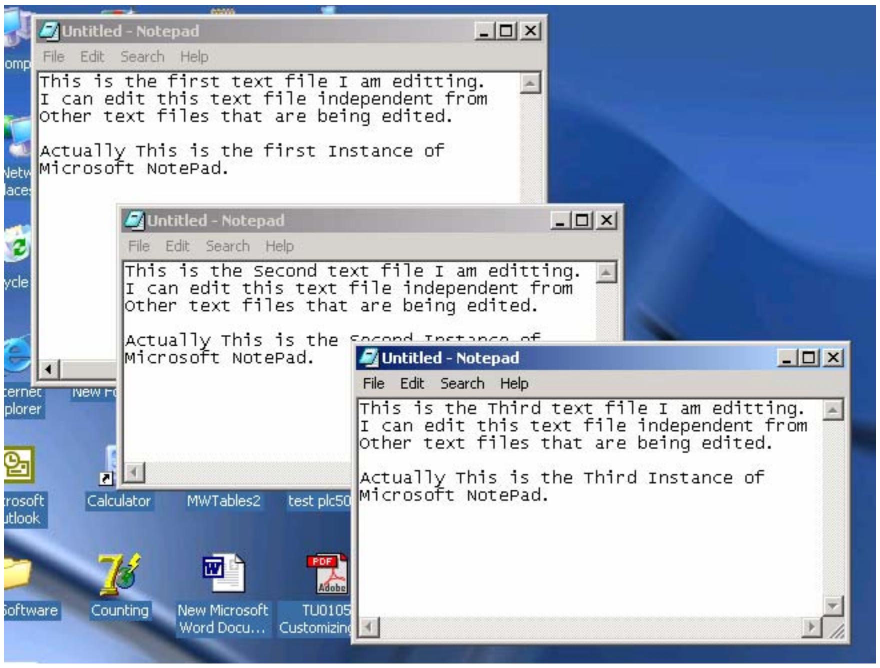
Figure 64: Online Status of execution of TXP

10. Now that TXP is built and works fine, you may use it several times in your project as if it is a firmware timer.

Note: You may have noticed that when you tried to transmit the POU containing an instance of TXP for the first time, TXP_1 was included in the list of the POUs that had to be sent before the POU containing it. For a more complete discussion about this subject please refer to the topic “Instantiation and Naming Conventions” that will be followed.

Instantiation

Thanks to the processing capabilities of modern computers and their powerful multi-task Operating Systems such as Microsoft Windows®, a computer may run different applications simultaneously. Moreover many copies (instances) of an application may run independently and simultaneously in a computer.
You may have noticed that many copies of an application such as “Microsoft Notepad” can run and edit multiple text files simultaneously. This is shown in figure 65 below.



```
This is the first text file I am editting.
I can edit this text file independent from
other text files that are being edited.

Actually This is the first Instance of
Microsoft NotePad.
```

```
This is the Second text file I am editting.
I can edit this text file independent from
other text files that are being edited.

Actually This is the Second Instance of
Microsoft NotePad.
```

```
This is the Third text file I am editting.
I can edit this text file independent from
other text files that are being edited.

Actually This is the Third Instance of
Microsoft NotePad.
```

Figure 65: Three instances of Microsoft Notepad are running in Windows®

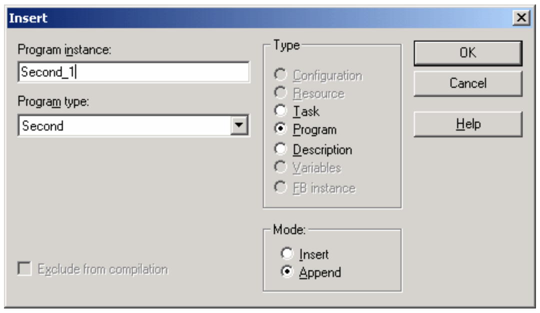
Instantiation and Naming Conventions in PLC500 Nseries

The POUs are called by tasks. Actually the POUs of type PROGRAM may be called by more than one task (Although it is rare but possible).

The same way and more often, the POUs of type FUNCTION-BLOCKS and FUNCTIONS may be called by many PROGRAMS. How calling tasks and PROGRAMS differentiate between different copies of called POUS?

The answer is the technique called "Instantiation" that you have frequently seen it before.

You may have noticed that when you want to add a PROGRAM to a task an instance name is asked by MWT.



Program instance: Second_1

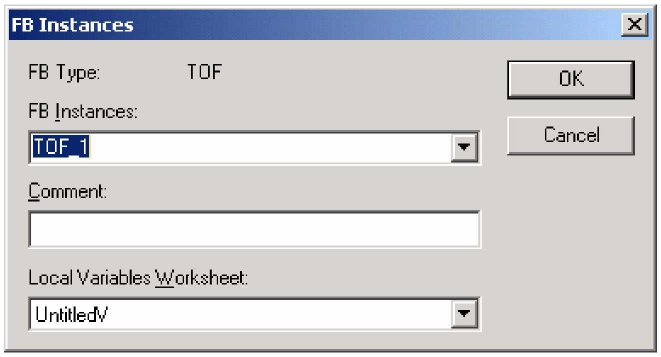
Program type: Second

Type: Program

Mode: Append

Figure 66: A Program Instance name must be specified when you add a POU to a task.

Similarly, when you insert an FB (firmware or user defined) into your worksheet, a dialog box appears immediately and asks you for an instance name. In such cases a default instance name will be provided by MWT. You may accept or type a different name for it.



FB Type: TOF

FB Instances: TOF_1

Comment:

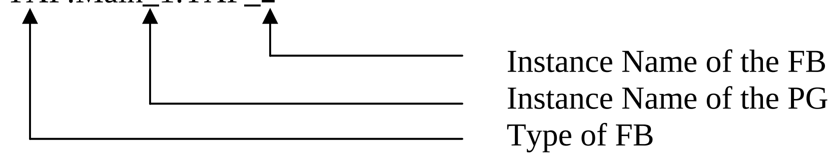
Local Variables Worksheet: UntitledV

Figure 67: An FB Instance name must be specified when you add it into a Program.

In the example above, proposed Instance name is TOF_1. But you may have many TOF timers with the same instance name (TOF_1) in different Programs. How CPU can differentiate them?

In PLC500 Nseries, name of called FB is extended by its type, its calling programs instance-name and instance name of FB each separated by one ":". For example if there are two instances of TXP, TXP_1 and TXP_2 in Main program, its extended name may be as follows:

TXP:Main_1:TXP_1
TXP:Main_1:TXP_2



Recalling that user defined function blocks may also be used in other user defined function blocks, name strings extends with the same naming convention.

Information Stack of CPU

CPU has an internal first-in first-out information stack. Events will be placed in this stack in plain text along with date and time stamps. Events will be placed in information stack in order of their occurrence. The first event is the mode of startup and will never be overwritten by subsequent events. The next events will be overwritten if a newer one occurred and there is no more room for them. Normally these events are abnormal events that cause the CPU to take a safety action such as invalid hardware configuration, invalid function call and the like.
Contents of information task can be viewed by clicking on information button in the resource dialog box as shown below.

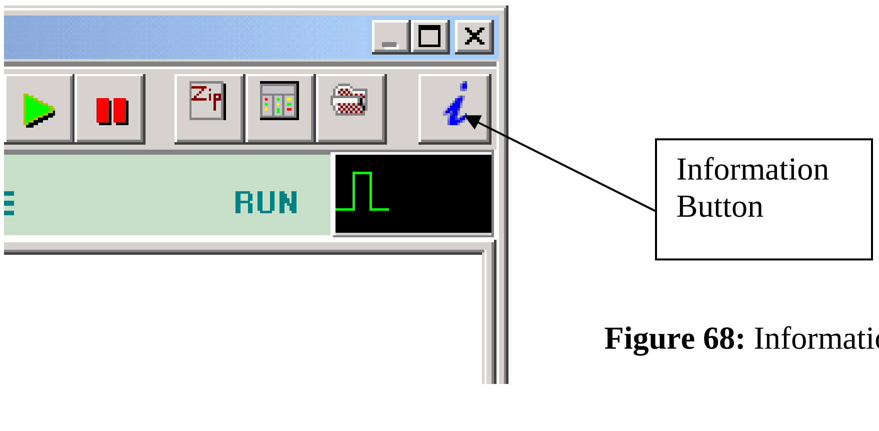
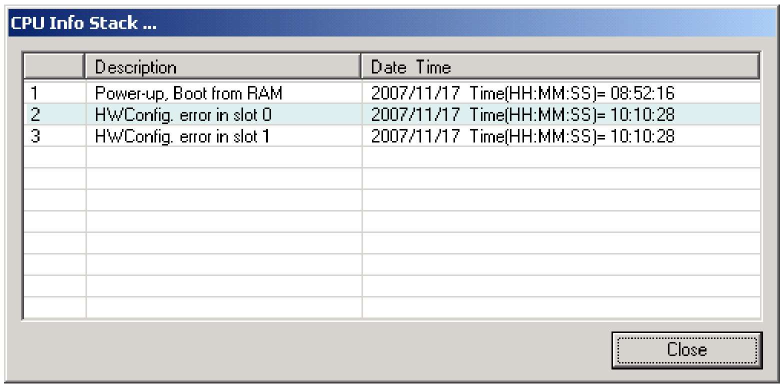


Figure 68: Information Stack

The following figure shows a sample information stack.


CPU Info Stack ...

	Description	Date Time
1	Power-up, Boot from RAM	2007/11/17 Time(HH:MM:SS)= 08:52:16
2	HWConfig. error in slot 0	2007/11/17 Time(HH:MM:SS)= 10:10:28
3	HWConfig. error in slot 1	2007/11/17 Time(HH:MM:SS)= 10:10:28

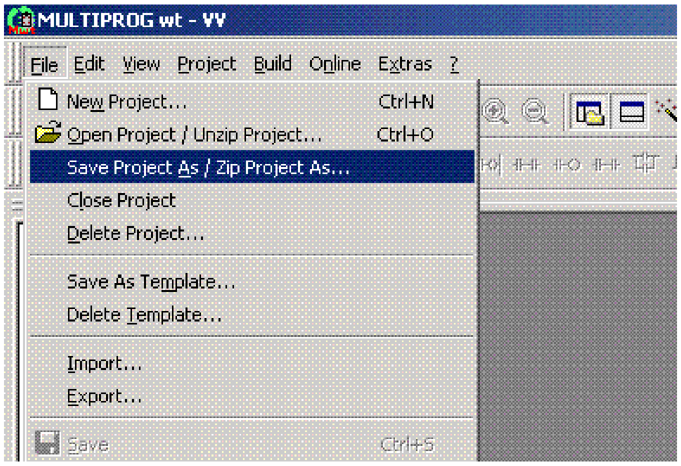
Figure 69: CPU Information stack.

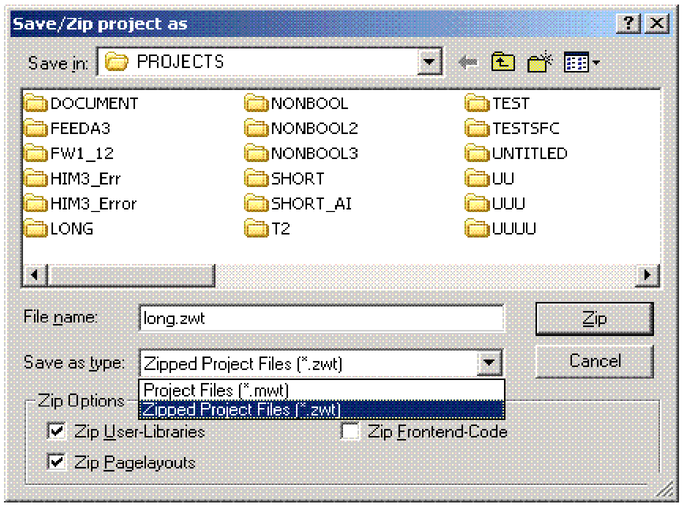
Creating a Zipped Project

The POUs generated in five different languages of IEC1131-3 will be eventually translated into a language very similar to the language IL before being sent to the PLC. Analysis of such codes although possible, is very complicated in a typical project. That’s why; reading back such codes from PLC is rather useless. In modern PLCs that complicated programs with many different data types are common, this complexity will grow even more.

In MWT, although the codes and files generated for a project is very detailed and complete, it is too big for a PLC to be held in memory. Fortunately, MWT can produce a small zipped project that contains all the details of a project. MWT can also unzip it back to the original project.

In order to make a zipped project, you should simply save it as a zipped project. Recalling that a MWT project has an extension of .mwt, the zipped project has an extension of .zwt. Following figures shows the procedure. Size of a several mega-byte project can be reduced to a few hundred kilo-byte one.





File name: long.zwt
Save as type: Zipped Project Files (*.zwt)
Zip Options
☑ Zip User-Libraries
☐ Zip Frontend-Code
☑ Zip Pagelayouts

Figure 70: Saving a project as a zipped project.

Fixing Programs in Flash Memory

User programs when sent to the PLC will be placed in battery backed up RAM. Therefore they will execute from RAM.
In order to securely hold the valuable codes generated for long time, you must fix them into the flash memory.
Before fixing you must send the zipped project to the PLC otherwise, fixing is not possible. This restriction is implemented in PLC500 Nseries purposely in order to remind you of importance of such action.
The zipped project contains all the details of your programs including POUs, Tasks, Global variables, hardware configuration, settings and also your comments and texts. Since the zipped project is still large and is not going to be executed by the CPU, it will be sent to the RAM area that is not backed by the battery. Usually, you send a zipped project to a PLC at the last steps of a project where you want to fix data in the flash memory of your PLC.
In power-up a copy of the zipped project will be loaded back from flash memory into the RAM if one exists.

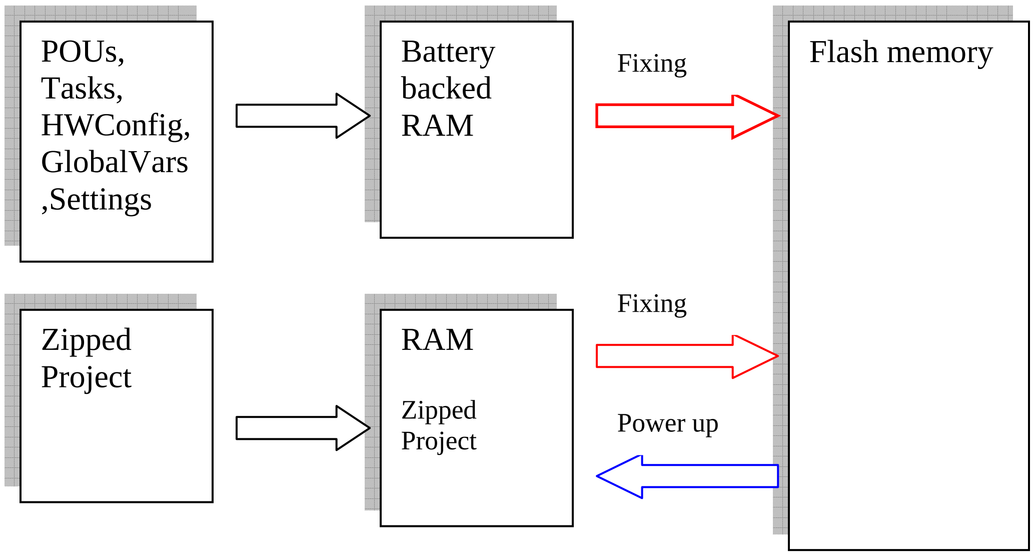


Figure 71: All data except the Zipped project are held in battery backed up RAM.

In order to do a complete fixing procedure, step through the followings:

1. Send all your generated and tested codes to the PLC.
2. Send the zipped project to the PLC.
3. Click on the “Fix Memory” button as shown below.

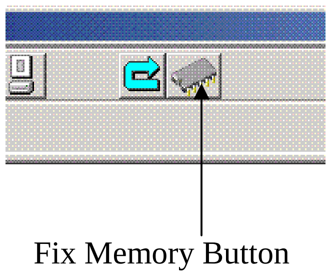


Figure 72: Fix memory button

Saving a project into the flash memory (Fixing) is a slow process and may take several minutes to complete. During the operation, the yellow Memory LED blinks with different frequencies. Please be patient for the process to complete.

During the fixing time, the PLC executes its tasks and there is no interruption to the control of the process being controlled.

Since PLC500 Nseries is a modern PLC with great capabilities, a description to all the details of its operation and programming techniques is not possible in one book. Therefore, the present material provided in this manual is just a start and gives a brief overview about some key points.
Users of PLC500 Nseries should write their own programs and make more practice to discover more details. At the same time they should refer to the manual of MULTIPROG wt to get more understanding about supported languages and editing techniques. Using online help system of MWT and specific help on any firmware FB and FU is also helpful.

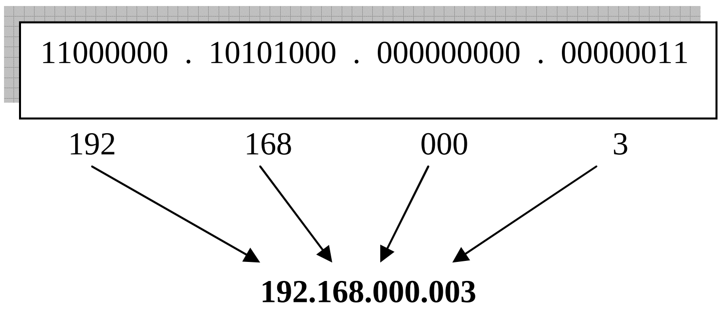
Contronic Co. Nov. 2007 (Aban 1386)

Appendix A:
Setting IP address of the Programmer Computer

This appendix gives a very simplified and brief overview about IP addressing and the way you can set it for your computer assuming that operating system of your computer is Windows XP. Other systems are very similar and you can invoke their corresponding help systems.

In order your computer to share in a local area network (LAN), the IP addresses of your computer must be unique and be in a predefined range.

Every IP address is a 32 bit binary number. Therefore a maximum of 4,294,967,296 unique addresses can be defined in a network. To make Internet addresses easier for human users to read and write, IP addresses are often expressed as four decimal numbers, each separated by a dot. This format is called "dotted-decimal notation."
Dotted-decimal notation divides the 32-bit Internet address into four 8-bit(byte) fields and specifies the value of each field independently as a decimal number with the fields separated by dots.
Following figure shows how a typical internet address can be expressed in dotted decimal notation.



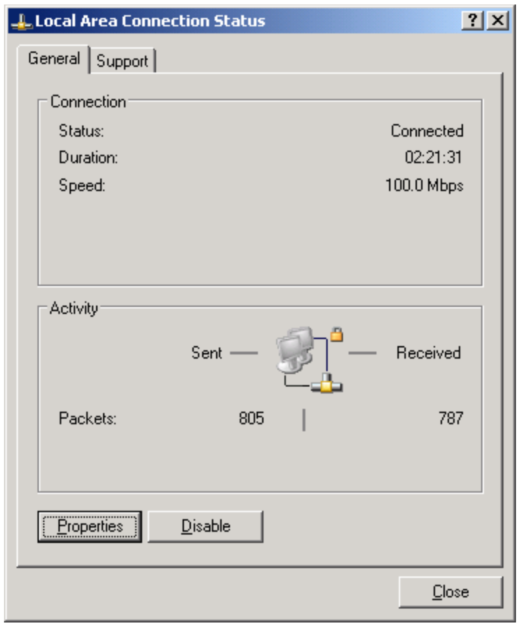
Since all the devices are not going to be connected to the internet directly, a standard procedure for sub-netting, or division of network to many sub-nets has been developed by international authorities. In these standards some groups and classes are defined which their discussions are beyond the scope of this article.

What we really need to know at the moment is that in a small LAN all the devices must have an IP address with identical first three bytes. The last byte may be freely assigned to the devices connected to the LAN. For example following addresses can be assigned.
192.168.000.020, 192.168.000.011, 192,168,000,030

There are two methods that IP address of a computer can be assigned. This address is either statically set once in setup of your computer or dynamically set by a server computer at system power up.
In this discussion we assume that your LAN does not have a server and we have to assign it manually (static IP addressing).

Please follow these steps if your computer has an OS of Windows XP®.

1. Open the Control Panel and select Network Connections.
2. Double click on “LAN or high speed internet”.
3. Following window will appear.



Local Area Connection Status

General | Support

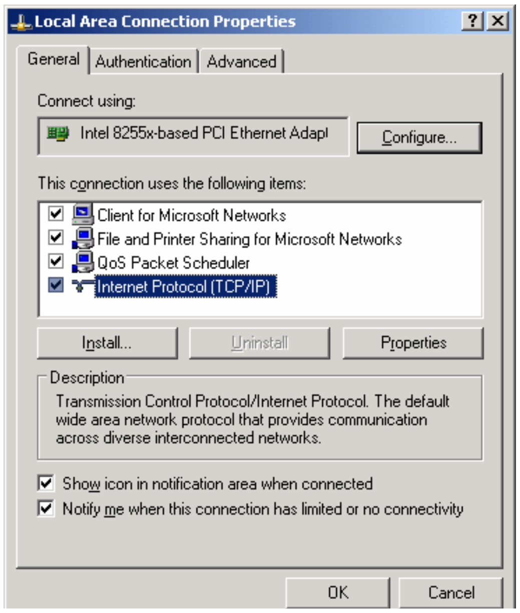
Connection

Status:	Connected
Duration:	02:21:31
Speed:	100.0 Mbps

Activity

	Sent	Received
Packets:	805	787

4. Click on the Properties button. Following widow will appear. Select Internet Protocol (TCP/IP) as shown below.



Local Area Connection Properties

General | Authentication | Advanced

Connect using:

Intel 8255x-based PCI Ethernet Adapt

This connection uses the following items:

- [✓] Client for Microsoft Networks
- [✓] File and Printer Sharing for Microsoft Networks
- [✓] QoS Packet Scheduler
- [✓] Internet Protocol (TCP/IP)

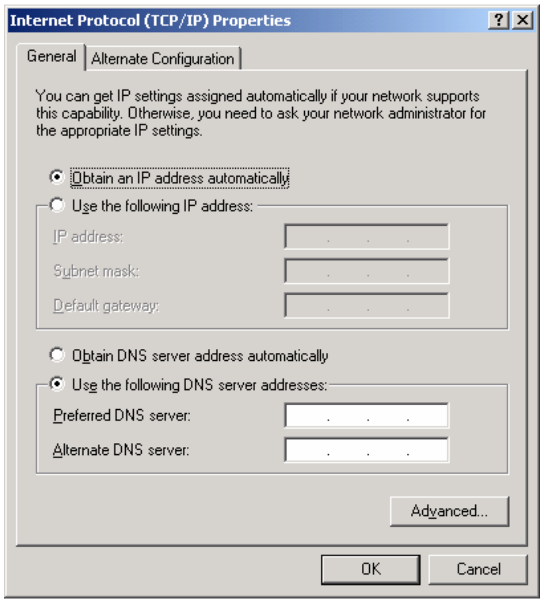
Description

Transmission Control Protocol/Internet Protocol. The default wide area network protocol that provides communication across diverse interconnected networks.

[✓] Show icon in notification area when connected

[✓] Notify me when this connection has limited or no connectivity

5. Click on the Properties button. If a static address is not already set, following widow will appear.



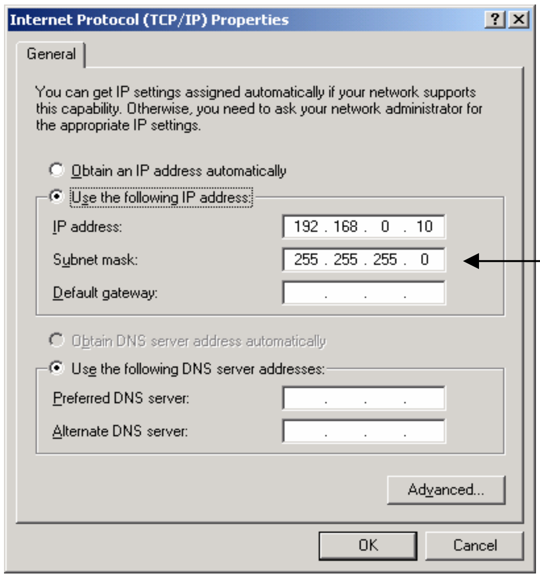
Internet Protocol (TCP/IP) Properties

You can get IP settings assigned automatically if your network supports this capability. Otherwise, you need to ask your network administrator for the appropriate IP settings.

[✓ Obtain an IP address automatically]
Use the following IP address:
IP address: . . .
Subnet mask: . . .
Default gateway: . . .

Obtain DNS server address automatically
[✓ Use the following DNS server addresses:]
Preferred DNS server: . . .
Alternate DNS server: . . .

6. Select "Use the following IP address" radio button and enter your desired IP address. You will see an example below.



Internet Protocol (TCP/IP) Properties

You can get IP settings assigned automatically if your network supports this capability. Otherwise, you need to ask your network administrator for the appropriate IP settings.

Obtain an IP address automatically
[✓ Use the following IP address:]
IP address: 192 . 168 . 0 . 10
Subnet mask: 255 . 255 . 255 . 0
Default gateway: . . .

Obtain DNS server address automatically
[✓ Use the following DNS server addresses:]
Preferred DNS server: . . .
Alternate DNS server: . . .

Always select a Subnet mask of 255.255.255.0

7. Press OK and terminate the session. The IP address of your computer is now statically set.