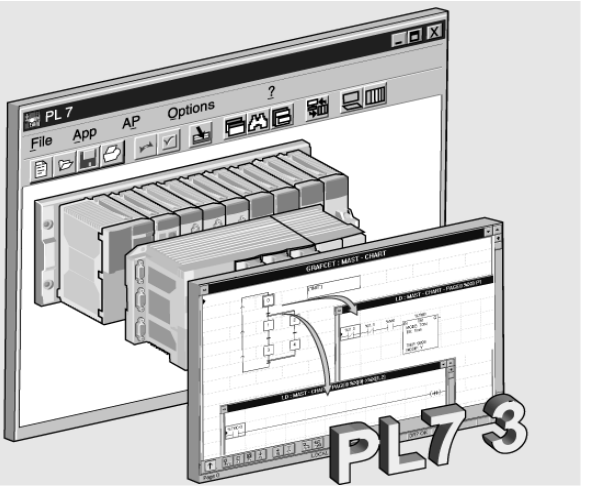
PL7 Junior

PL7-3 Application Converter

User's Manual

March 2005



a brand of Schneider Electric

Telemecanique

PL7-3 application converter

eng March 2005

W91590515030 1A_03

a brand of Schneider Electric

Telemecanique

1.1 Introduction

The PL7-3 converter is used to convert the various modules of a PL7-3 application to PL7. This **modular conversion** thus requires firstly that the PL7-3 application is saved as a source modules : program modules (PRL, POST, CHART, etc), symbol or constants file.
The binary file (.BIN), which contains the whole application, may be needed to retrieve some information on the software configuration which it has not been possible to save as a source file (such as the preset value of standard function blocks).
Although it is integrated into PL7, installation of this conversion tool is optional, in order not to "overload" those users who have no use for it.

Once this converter is installed, it can convert :

- The symbols and comments, contained in an .SCY file,
- The constants, contained in a .CST file,
- The Ladder language rungs of a PL7-3 program module (MAIN, PRL, POST, SRi, contents of a Grafcet step, contents of a transition condition), contained in a .LAD file,
- The program lines of a PL7-3 program module (MAIN, PRL, POST, SRi, contents of a Grafcet step, contents of a transition condition), contained in an .LIT file,
- The Grafcet pages of a PL7-3 program module (CHART, XMi), contained in a .GR7 file. Depending on the choice made when archiving the module under PL7-3, the Ladder or Structured Text code contained in its steps and transitions may or may not be included.

Note

The source files (.LAD, .LIT, .GR7) at the converter input must be in **non symbol** form (without symbols). However, they can be associated with a symbols file (.SCY) and/or a constants file (.CST). In this case, the symbols and/or constants **contained in the program** are also converted.

Warning : if a word or an **indexed** constant double word is used in the program, all the constants contained in the associated constants file are converted (in fact, indexed addressing can be used to indirectly access all the file constants via the value of the index word).

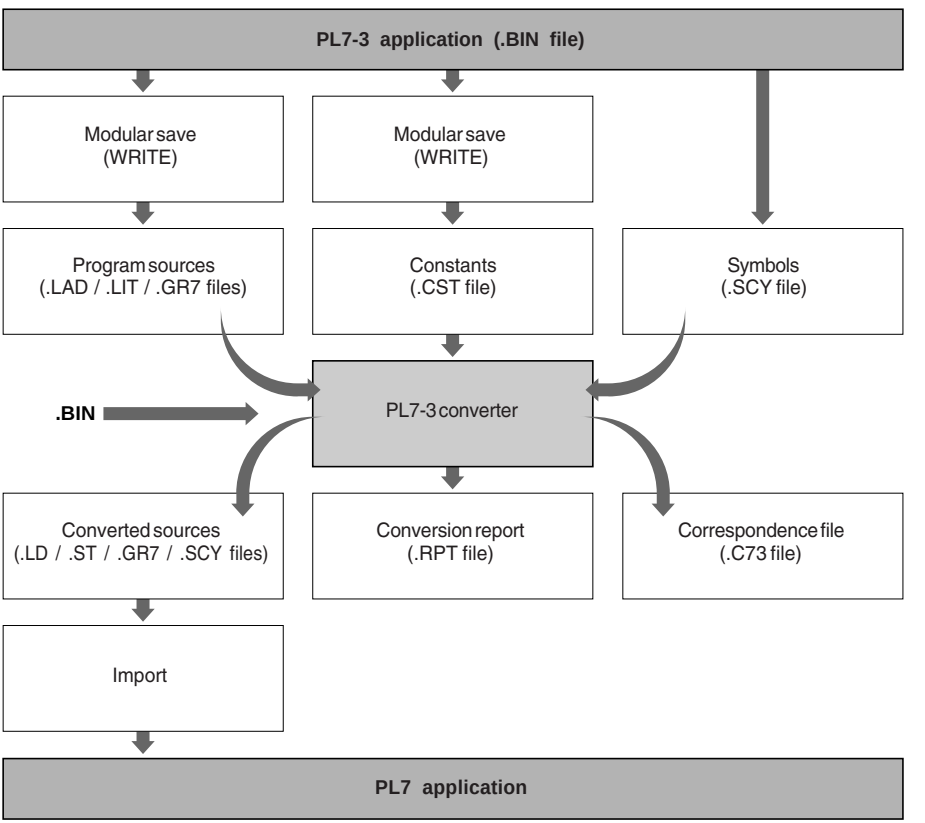
Conversion is automatic, with the exception of some PL7-3 objects which have no PL7 equivalent. On completion of the conversion, the user has the following files at his disposal :

- a **program source file**, the result of the conversion, referenced .LD (conversion of a Ladder language module), .ST (conversion of a Structured Text language module) or .GR7 (conversion of a Grafcet module). This file can be imported under PL7.

  If the conversion is only performed on a symbols file and/or a constants file (no program source file entered), the converter generates a **symbols source file**, referenced .SCY. In this case, all symbols and/or constants are translated.

- a **report file**, referenced .RPT, which contains the context of the conversion, the list of configured and non-configured objects, the list of objects which have not been converted, etc, the list of program elements (Ladder language rungs, Structured Text statements, Grafcet pages) and their converted state (complete or incomplete conversion). This file can be displayed or printed from the converter,
- a **correspondence file**, referenced .C73, which gives the correspondence between the PL7-3 objects present in the source file and the objects which have been converted (automatically or manually). This file is optional.

Principle of modular conversion



Warning

Since some objects do not have an automatic correspondence in PL7 (I/O, text blocks, etc) a program module may not operate correctly following the conversion. The following manual operations must therefore be performed in order to complete the conversion :

- modify the PL7 configuration, following the instructions contained in the conversion report,
- import the source (*.LD, *.ST or *.GR7), following the instructions contained in the conversion report,
- complete the incomplete program elements (Ladder language rungs, Structured Text statements or charts).

1.2 Modular save of the PL7-3 application

Since the PL7-3 converter input files are source files and not the application binary file, firstly the application must be saved in modular form. This involves archiving all the program modules contained in the application (PRL, POST, CHART, etc) one after the other, as well as the symbols and constants (see description of the WRITE command in PL7-3 operating modes). The corresponding source files (.LAD, .LIT, .GR7, .SCY and .CST) are archived under the directory ...\PL7_3\MOD.

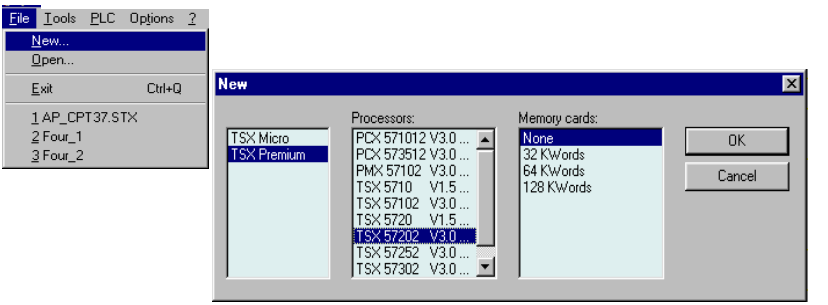
Converting a PL7-3 application to a PL7 application will mainly involve converting all the PL7-3 modules to PL7 modules. Global application problems, such as the following, will also have to be resolved :

- task organization, which is different in PL7-3 and PL7,
- PL7-3 and PL7 hardware and software configuration similarities and differences.

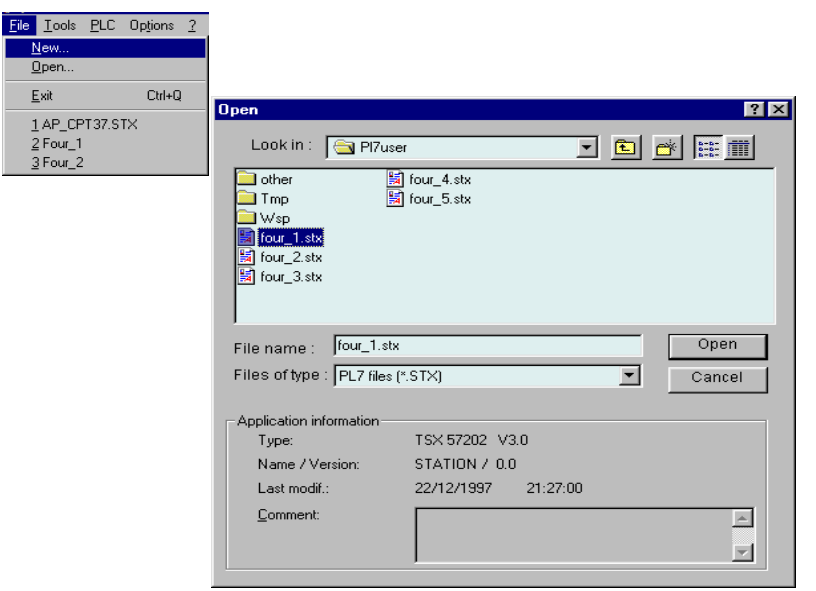
1.3 Procedure for converting PL7-3 modules to PL7

The PL7-3 converter is called from a PL7 station, which can be accessed using one of the following commands :

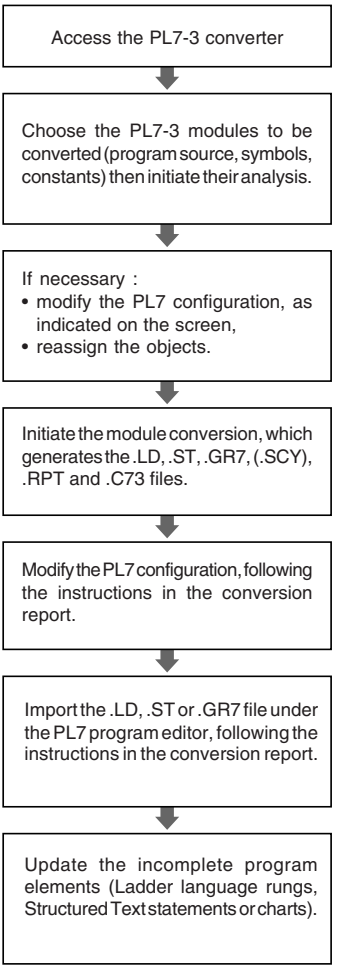
- the **File/New** command, to recover PL7-3 modules converted to a "new" PL7 application. A dialog box is used to select the type of processor (and thus to define the station) :



- the **File/Open** command, to recover PL7-3 modules converted to a PL7 application which already exists. A dialog box is used to select which of the PL7 applications on the disk (.STX files), must be opened :



Then perform the following conversion procedure to convert each PL7-3 module :



File / **Convert** / **PL73 Converter** command (see section 2.1)

(see section 2.2)

(see sections 2.2-3 and 2.4)

(see section 2.3)

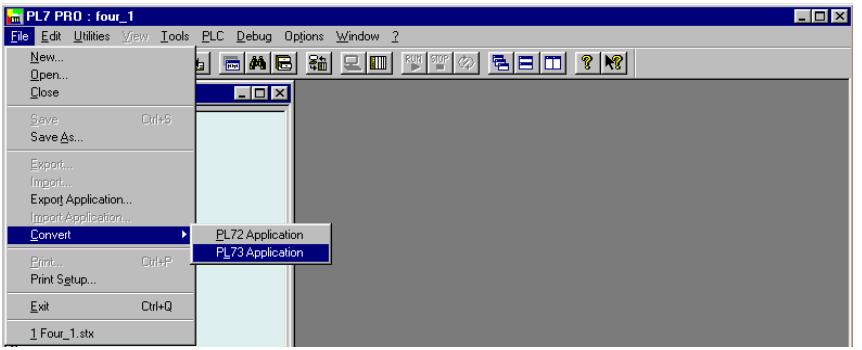
(see section 2.4)

(see section 2.5)

(see section 2.5)

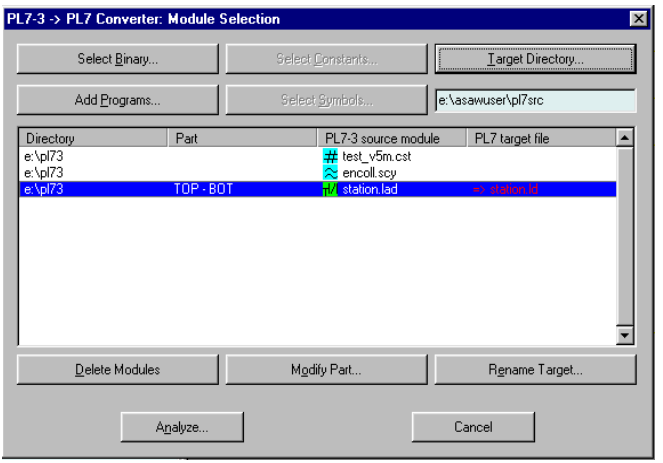
2.1 Accessing the converter

To access the conversion tools, a PL7 application must first be opened (**File**/**New** or **File**/**Open** command). The PL7-3 converter can then be accessed via the **File** menu, by activating the **Convert/PL73 Application** command**.**



2.2 Choosing the modules to be converted

This is performed using the following dialog box where the program module(s) to be converted (.LAD, .LIT or .GR7) can be selected. These module(s) can be associated with a symbols module (.SCY) and/or a constants module (.CST), to ensure that the symbols and constants contained in the program are also converted.



Selecting the PL7-3 modules to be converted
The buttons in the top part of the window can be used to select the different PL7-3 modules to be converted :

Add Programs... : This key displays a dialog box which is used to define which PL7-3 program module is to be converted. Several PL7-3 program modules can be selected and converted simultaneously. It is possible to partially convert a module (see section 2.2-1).

Select Symbols... : This key displays a dialog box which is used to define which of the symbols files on the disk is associated with the program module. If an .SCY file is selected (not compulsory), only those symbols contained in the program module will be converted.

Select Constants... : This key displays a dialog box which is used to define which of the constants files on the disk is associated with the program module. If a .CST file is selected (not compulsory), only those constants contained in the program module will be converted.

Target Directory... : This key displays a dialog box which is used to modify the target directory to which the files will be copied after conversion. By default, the destination file is archived in the PL7 source directory (the **Options / Customize** menu is used to display and if necessary modify this directory).

Select Binary... : This key displays a dialog box which is used to select the PL7-3 binary file (directory path and name) from all those present on the disk. This PL7-3 binary file (.BIN) gives the parameters for the standard function blocks contained in the source file to be converted.

List of selected modules
This list in the central part of the window displays all the information on the selected conversion.

The **Directory** column shows the tree-structure of the selected modules.

The **PL7-3 source module** column displays the name of the **.LAD** (Ladder), **.LIT** (Literal) or **.GR7** (Grafcet) program module, the **.SCY** symbols module and the **.CST** constants module. The name of the module is preceded by an icon to facilitate identification.

It is possible to select part of a program module (a selection of Ladder rungs, Structured Text statements or Grafcet statements). The **Part** column shows the conversion limits (first and last element to be converted).

The **PL7 target file** column is used for defining the name of the file after conversion. If a module is selected, by default this file will take the same name as the input file, followed by the extension **.LD** (Ladder input file), **.ST** (Structured Text input file) or **.GR7** (Grafcet input file). If only a symbols and/or constants file is selected (no program file), the target file will take the same name as the symbols file (or the constants file if necessary), followed by the extension **.SCY**.

Module Selection window command zone

The buttons at the bottom of the window are used to select the following actions :

- **Delete Modules** : This key can be accessed after a single or multiple selection has been made in the list. It deletes the selected module(s) from the list.
- **Modify Part...**: This key can be accessed if a PL7-3 program module is selected. It is used to define the conversion limits for the module to be converted (see section 2.2-1).
- **Rename Target...** : This key can be accessed if a PL7-3 program module is selected. It displays a dialog box which can be used to modify the name of the target file. If the file name defined already exists, it appears in red. The .LD (Ladder), .ST (Structured Text), .GR7 (Grafcet) or .SCY extension must, however, stay the same.
- **Analyze..** : This key starts the analysis of the selected modules (see section 2.2-2).

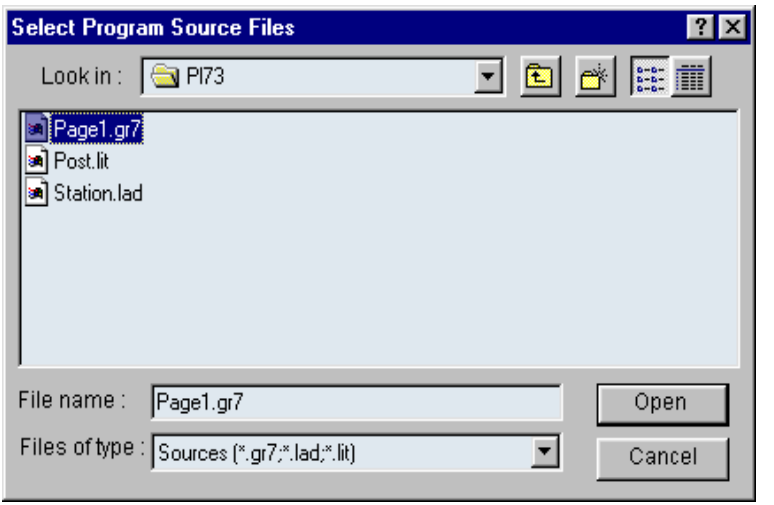
Notes :

Multiple selection is only possible for program type modules**.** It is not possible to associate more than one symbols file, constants file or application binary file with the conversion.
In order for a binary file to be associated with the conversion, the user must ensure that all the program module files to be converted actually belong to this binary file. If not, the information obtained after the conversion will not be consistent with the source files.

2.2-1 Selecting the program module

The program module is selected from the converter dialog box, which can be accessed using the **Add Programs...** key in the **Module Selection** screen. It enables the source file(s) to be selected as the converter input, and also, if necessary, the program elements to be converted (partial conversion of the module).

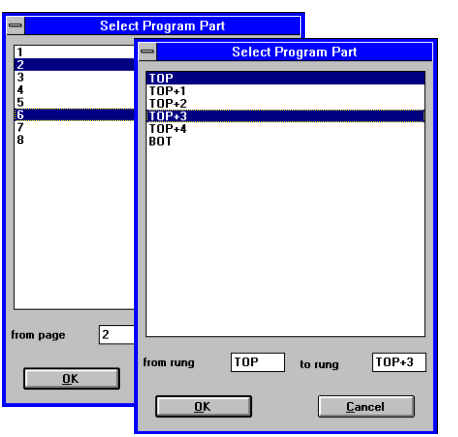
The secondary window which is used to select the files is standard. It can be used to select a drive, the file access path and the file.



Once the selection has been made and confirmed, the main module selection window reappears. It is then possible, by selecting a program source file in the main window list, to select the whole or part of the module to be selected.

Modify Part...

This key enables the whole module (default option) or part of the module to be selected. It displays the following dialog box which gives the index of all the module program elements and is used to define the conversion limits (first and last element to be converted).



Select the first program element to be converted (Ladder language rungs, Structured Text statement or Grafcet page) using the mouse. The element then appears in the **from rung**, **from statement** or **from page** field.

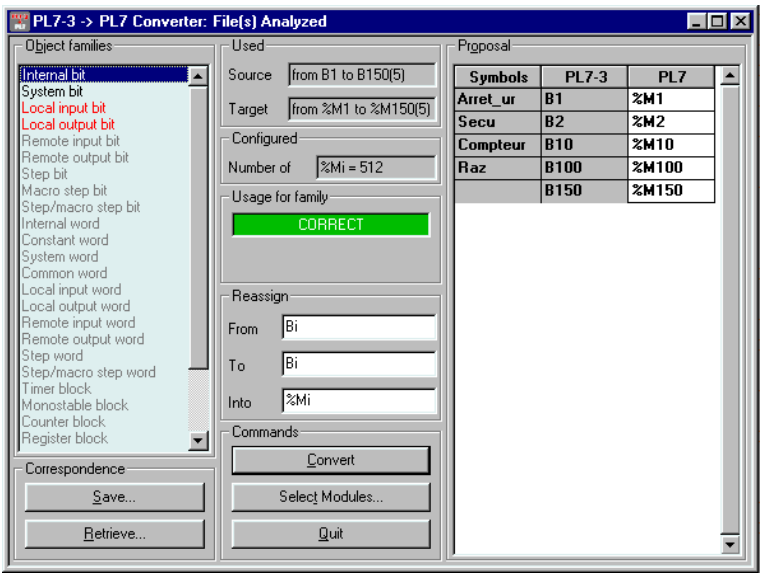
Repeat this operation for the last program element to be converted, which then appears in the **to rung**, **to statement** or **to page** field.

After confirming with **OK**, the partial selection performed is recalled in the **Part** column in the main window.

2.2-2 Analysis and reassignment of objects

After choosing the PL7-3 modules to be converted, select the **Analyze** key to initiate the analysis of the program module by the converter (see section 2.2).
All the PL7-3 objects in the module, which have an equivalent in PL7 are translated to the new syntax. The PL7-3 objects which have no equivalent in PL7 are not translated and their positions under the program editor will remain empty.

After the analysis, the following dialog box summarizes all the PL7-3 objects in the module and enables some objects to be reassigned.



Object families
This field lists the PL7-3 object families and is used to choose a family whose objects are displayed with their symbol and their PL7 equivalent.
The color in which families are displayed indicates whether the objects have been translated :

- black indicates that PL7-3 objects were found during the analysis of the module (or the module part) and that :
  - objects of the same type have been automatically assigned in PL7,
  - if the family is shown between "<<" and ">>" characters (<<Text block>>, for example), there is no equivalent in PL7,
  - if the family is shown between "<" and ">" characters (for example, <Remote input word>), different types of object have been automatically assigned in PL7.

- gray indicates that no object of this type was found during the analysis of the module (or the module part) but that :
  - the converter can automatically assign objects of the same type in PL7,
  - if the family is shown between "<<" and ">>" characters (<<Control block>>, for example), the converter cannot assign equivalent objects in PL7,
  - if the family is included between "<" and ">" characters (<Remote output word>, for example), the converter can automatically assign different types of object in PL7.
- red indicates that objects were found during the analysis of the module (or the module part) but that conversion requires help from the user :
  - these objects must be assigned manually (discrete I/O bits, system bits without any equivalent). If the family is shown between "<" and ">", this means that the objects must be assigned by objects of a different type,
  - these objects are not fully configured in PL7.

Warning

Not all of the PL7-3 objects used as parameters during the execution of a diagnostic OFB (statement # EXEC in Structured Text language) appear in the correspondence table (only those used elsewhere appear).

Used

For the selected family this field displays :

- in the **Source** zone : the first object and the last object found during the analysis of the module (or the module part). The total number of objects found appears in brackets. For example, from B0 to B208 (64),
- in the **Target** zone : the PL7 objects equivalent to the objects found during the analysis of the module (or the module part). The total number of objects converted appears in brackets. For example, from %M0 to %M208 (64). During automatic conversion, the address of the objects is not modified. For example, B**56** becomes %M**56**.

Configured

This field indicates the number of objects configured under PL7, for the selected family.

Usage for family

This field clearly shows the conversion result for the selected family :

- the message "CORRECT", displayed on a green background, means that the PL7-3 objects found during the analysis were automatically and converted to PL7 objects with no problem,
- the message "NO OBJECT", displayed on a green background, means that no PL7-3 object was found during the analysis,
- the message "NOT ASSIGNED", displayed on a red background, means that the objects found during the analysis were not assigned. It is the user who must perform this operation manually.

- the message "NOT CONFIGURED", displayed on a red background, means that the objects found during the analysis are not fully configured in PL7. In this case it is recommended that the configuration editor should be initiated and the configuration of the PL7 objects should be modified so that the objects of this family are fully configured,
- the message "NOT CONVERTIBLE", displayed on a red background, means that the objects found during the analysis have no equivalent in PL7, and cannot therefore be converted.

When the message NOT ASSIGNED, NOT CONFIGURED or NOT CONVERTIBLE is displayed, the **Number of objects** field indicates the number of objects in the family in question, for the conversion problem displayed.

Warning : for objects which appear several times in the PL7-3 source program and which cause a conversion problem, the **Number of objects** counter will take into account the number of times the objects have been found, even if they only appear once in the PL7-3 / PL7 correspondence table (**Proposals** field).

Reassign

One or more objects of the selected family can be reassigned in this field. As the syntax of the objects is given in the entry zones (for example, Bi), only the address of the object needs to be entered (for example, B**6**) :

- **From** : first PL7-3 object to be reassigned,
- **to** : last PL7-3 object to be reassigned,
- **into** : first PL7 object of the destination range.

Notes

Reassignment is not possible for all families : in some cases there is no reason for it (system bits); in other cases it is subject to restrictions (for local input bits the limit objects must have the same rack index and module index). For families between "<" and ">" characters, reassignment is not possible.

It is also possible to define the first and last object to be reassigned, by clicking on objects in the PL7-3 column of the **Proposals** field.

To reassign a single object, it is advisable to directly enter the destination object in the PL7 column of the **Proposals** field directly.

Proposals

This field gives the list of conversions performed for the selected family. For each object converted it gives the symbol of the object, the name of the PL7-3 object and its equivalent in PL7.

If the PL7 object is displayed in black, this means that there is perfect correspondence between PL7-3 and PL7. The address of the PL7 object can be modified by positioning the cursor on the object concerned then by changing its address via the keyboard.

If the PL7 object is displayed in red, this means there has been a conversion problem :

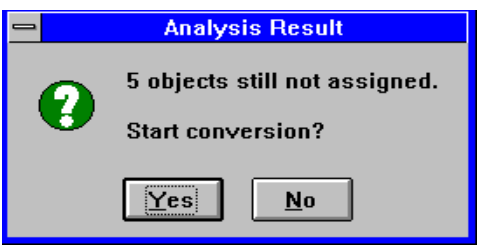
- the object offered is not configured. Initiate the configuration editor under PL7 and modify the number of objects of this type configured. Having confirmed the new configuration (if it is sufficient), the family and the objects (or some of the objects) of this type will be displayed in black,

- the object offered is not perfectly defined (for example, the I/O or some system bits). The user can click (single or double-click) on the object and update his address. When red has disappeared for one type of object suggested, the corresponding family will be displayed in black.

Correspondence
This field makes it possible to save the correspondence table in a .C73 file (**Save...** key) or conversely to rebuild the correspondence table from a .C73 file (**Retrieve...** key), saved previously (see section 2.6).

Commands
This field offers three keys which are used to start conversion of the module (**Convert** key), return to the program module selection screen (**Select Modules...** key) or abandon the conversion (**Quit** key).

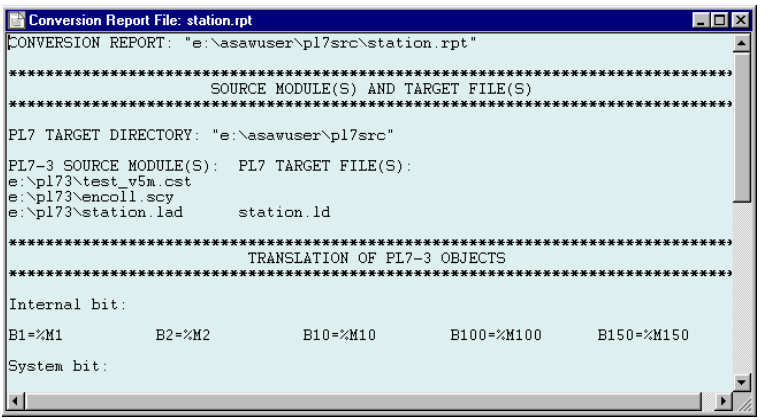
The conversion of the PL7-3 module (or the module part) in PL7 generates an .LD, .ST or .GR7 source file, and also the report file .RPT (see section 2.3).
Selecting the **Convert** key displays a dialog box for confirming the conversion which indicates the number of remaining conversion problems (unassigned or unconfigured objects).



2.3 Result of the conversion

After converting a PL7-3 module (or module part), the user has at his disposal a Ladder source file (.LD), a Structured Text source file (.ST) or a Grafcet source file (.GR7), and also a report file (.RPT) which is automatically displayed and which can be printed. This text file enables the converted module to be retrieved under PL7. To do this, it has two parts :

- the conversion **report**, which retrieves the converter environment for this conversion (name of report file, name of source files converted and name of target file(s) created) and which gives, for all PL7-3 objects found and classified by family, the result of the conversion (PL7 equivalent) :



- objects which have been correctly translated are displayed with their PL7 equivalent. If some of these objects are not configured in PL7, in the second part of the report (recommendations) it will be requested that the configuration be modified in order to include these objects.

```
Internal bit:
B1=%M1          B2=%M2          B3=%M3          B100=%M100

System bit:
SY0=%S0         SY1=%S1

Internal word:
W10=%MW10       W257=%MW257

Constant word:
CW0=%KW0
```

- objects which must be completed manually are partially translated. The PL7 equivalent of these objects contains the character "?".

```
Local input word:
IW4,0=%IW4.?     IW5,2=%IW5.?

Local output word:
OW4,0=%QW4.?     OW4,1=%QW4.?     OW5,1=%QW5.?
```

- objects which have no PL7 equivalent are indicated by "?" characters and the corresponding family is displayed between "<<" and ">>".

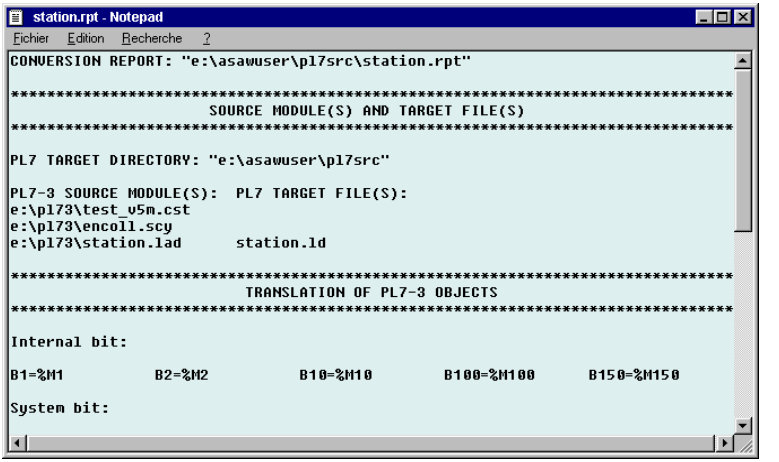
```
<<Control block>>:
CTRL4=????
```

- a **set of recommendations** or warnings which guide the user through the remaining operations :
  - objects to include in the PL7 configuration, for types of objects which are not fully configured. For a given type of object, the last address used determines the minimum number of objects to be configured. For example, if the converted module includes two timers %T0 and %T6, at least seven timers will have to be configured in PL7 (last index + 1),
  - the procedure for importing the converted module into PL7 : open an application in an identical language to that of the PL7-3 source module then select the File / Import command,
  - the list of incomplete program elements with, for each one, the problems encountered during conversion : unconvertible object(s), unassigned object(s), etc.

```
*******************************************************************************
                                RECOMMENDATIONS
*******************************************************************************

start importing file in language Structured Text (Menu File -> Import...)..

Problems during conversion:

Line: TOP+10    1 object(s) not convertible
Line: TOP+15    1 object(s) not assigned

Line: TOP+18    2 object(s) not assigned

Line: TOP+23    2 object(s) not assigned
```

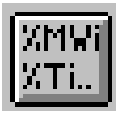
Accessing the report file

Apart from during its creation, when the report file is automatically displayed, the Notepad in the Window Accessories group must be used to open this file and hence access the contents of the report.
The Notepad enables the report to be displayed and printed and, if required, customized using the entry functions.

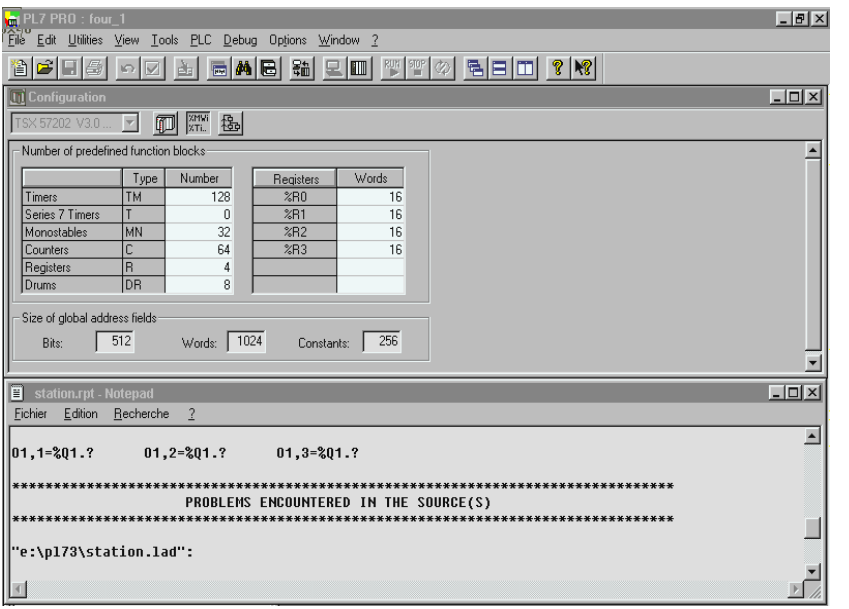


2.4 Reconfiguring PL7 objects

If families of PL7 objects were not configured when the objects were reassigned (see section 2.2-2), the objects concerned are indicated in the RECOMMENDATIONS part of the conversion report. The following must be performed :

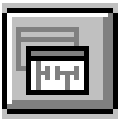


- launch the application browser, then access the configuration of PL7 objects,
- resize the configuration window so that it is displayed in the upper half of the screen,
- open the report file (see section 2.4) then resize it so that it occupies the lower half of the screen,
- follow the recommandations of the report file.
  Confirm the new configuration.

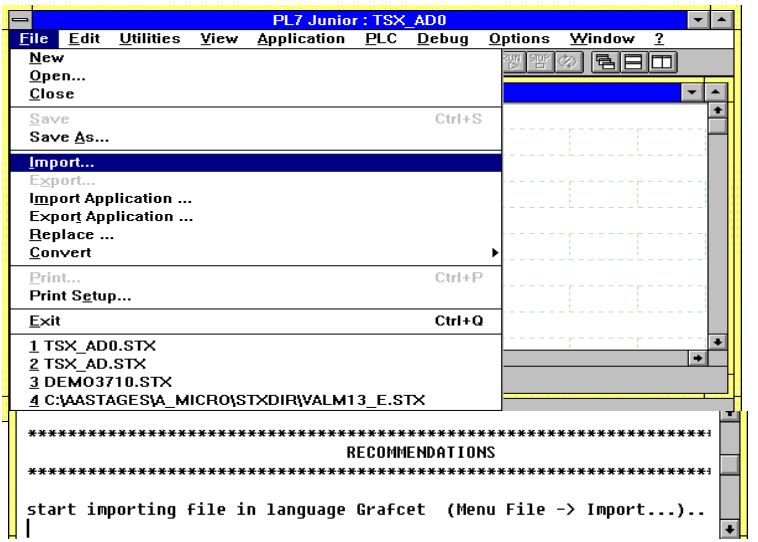


2.5 Importing the converted file into PL7

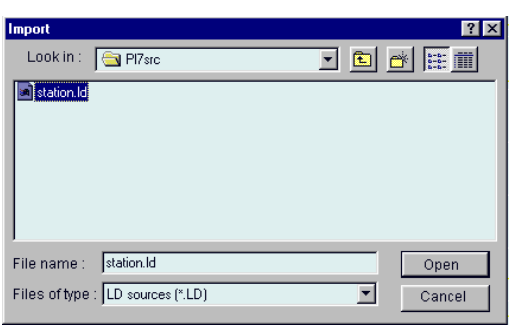
If the conversion has been interrupted after the creation of the .LD, .ST or .GR7 source file, re-establish the conversion context by opening the .STX file which has been saved. To import the source file into PL7 :



- launch the application browser (Ladder, Structured Text or Grafcet editor), then resize the screen so that it is displayed in the upper half of the screen,
- open the report file if it is not on the screen (see section 2.3) then resize it so that it occupies the lower half of the screen,
- follow the recommendations in the report file, that is, initiate the import of the .LD or .GR7 file (**File** menu, **Import** command).

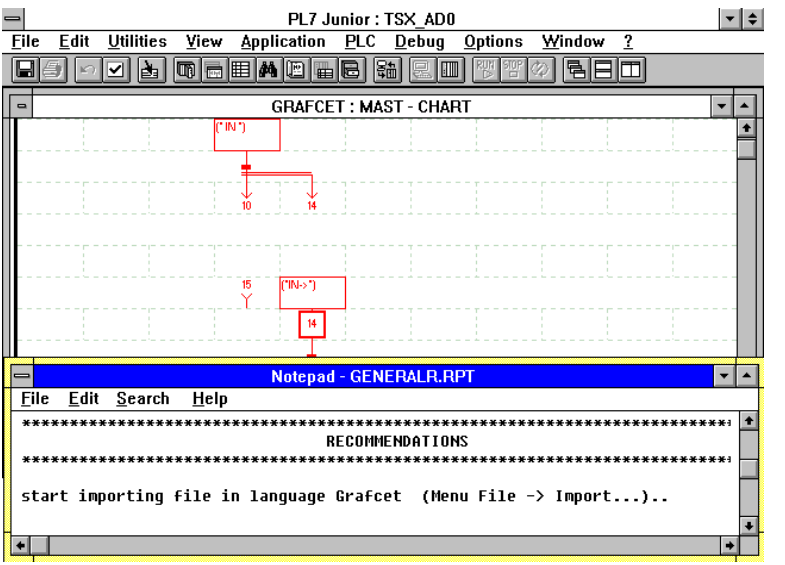


- define the tree-structure of the source file to be imported (logical drive, directories, .LD, .ST or .GR7 file name) then start its import with **OK**.



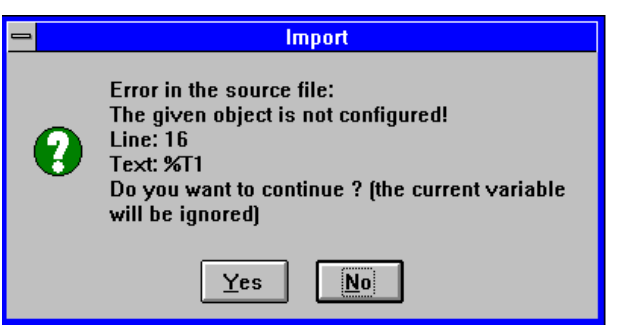
During the import operation, all the program elements which have been converted correctly are inserted automatically into the selected PL7 module. These program elements (Ladder language rungs, Structured Text statements, or charts) are displayed in black.

If a conversion error occurs, the program elements are incomplete and displayed in red. The recommendations in the conversion report must be followed to update the elements and thus validate them.



Note :

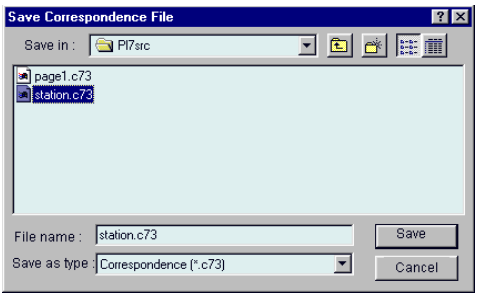
Program elements are imported as "discrete" logic and the import stops when it meets an incorrectly converted program element. The error is indicated in a dialog box which enables the user to either correct or continue the import.



2.6 Correspondence file

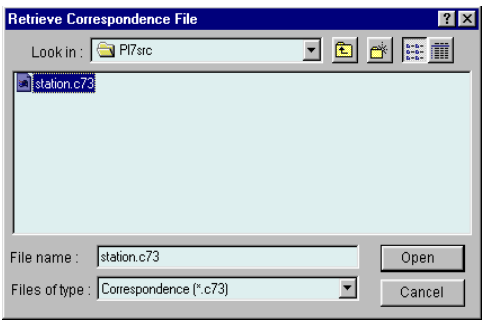
The .C73 correspondence file enables a copy (save) to be kept of a source file analysis and the reassignments, so they can be retrieved if need be. This file is therefore assigned to a PL7-3 source file, whose name it takes by default.

The correspondence file is created systematically during each analysis. It can also be created by the user from the file analysis screen, using the **Save** command (see section 2.2-2). This displays the following dialog box :



By default the correspondence file takes the same name as the source file, followed by the extension .C73. The file name can be modified, but not its extension (.C73). This will be saved in the same directory as the source file.

Reading a correspondence file enables objects resulting from the analysis of a program module (or a module part) to be reassigned, according to the values saved in this save file.
The correspondence file is read from the module analysis screen using the **Retrieve** command (see section 2.2-2). This displays the following dialog box :



3.1 Correspondence between PL7-3 and PL7 objects

The conversion procedure creates, as far as is possible, a table of correspondence between PL7-3 objects encountered in the program source on input, and their PL7 equivalents. PL7-3 objects can be divided into three categories :

(1) objects which have a PL7 equivalent and which are automatically translated.

(2) objects which have a PL7 equivalent, but which are not automatically translated. This is the case for I/O objects whose correspondences must be defined by the user. It is to be noted that for these PL7-3 objects which do not have any direct equivalent, the user must choose a PL7 object belonging to the same family, except in the following two cases :
- bits and words associated with macro-steps are replaced by step bits and words,
- PL7-3 remote I/O objects can become local PL7 I/O objects.

(3) objects which exist in PL7-3, but not in PL7 (Text block, Control block, OFB).

Note
System bits and words form heterogeneous families of objects and can belong to the three categories : SY0 becomes %S0, SY11 cannot be replaced by %S11 because these two objects have different functions, SY12 exists but %S12 does not exist.

The following tables give the correspondence and the possible differences between PL7-3 and PL7 objects :

Immediate values

Objects	PL7-3	PL7
Base 10 integer	1234	1234
Base 2 integer	L'0000000010011110'	2#10011110
Base 16 integer	H'ABCD'	#ABCD
BCD integer	B'1234'	#1234
Floating point	-1.32e12	-1.32e12
Character string	M'aAbBcB'	'aAbBcC'

Labels

Objects	PL7-3	PL7
Label	L i i = 1 to 999	%L i i = 1 to 999

Bits

Objects	PL7-3	PL7
Rack input bit	Ixy,i	%I<rack_mod>.<channel> (1)
Indexed rack input bit	Ixy,i (Wj)	
Remote input bit	RIx,y,i	%I\<path>\<mod>.<channel> (1)
Indexed remote input bit	RIx,y,i (Wj)	
Rack output bit	Oxy,i	%Q<rack_mod>.<channel> (1)
Indexed rack output bit	Oxy,i (Wj)	
Remote output bit	ROx,y,	
%Q\<path>\<mod>.<chan.>(1)		
Indexed remote output bit	ROx,y,i (Wj)	
Rack I/O fault bit		
• module fault bit	Ixy,S / Oxy,S	%I<rack_mod>.MOD.ERR
• channel fault bit		%I<rack_mod>.<channel>.ERR (1)
Remote I/O fault bit		
• module fault bit		%I\<path>\<mod>.MOD.ERR (1)
• channel fault bit	RDx,y,i / ERRORx,y,i	%I\<path>\<mod>.<chan.>.ERR (1)
• output channel tripping bit	TRIPx,y,i	
• output channel reset bit	RSTx,y,i	
Internal bit	Bi	%Mi
Indexed internal bit	Bi(Wj)	%Mi[%MWj]
System bit	SYi	%Si (2)
Step bit	Xi	%Xi
Macro-step bit	XMj	%XMj
Step bit i of macro-step j	Xj,i	%Xj.i
Input step bit of macro-step j	Xj,I	%Xj.I
Output step bit of macro-step j	Xj,O	%Xj.O
Bit j of internal word i	Wi,j	%MWi:Xj
Bit j of indexed internal word i	Wi(Wk),j	%MWi[%MWk]:Xj
Bit j of constant word i	CWi,j	%KWi:Xj
Bit j of indexed constant word i	CWi(Wk),j	%KWi[%MWk]:Xj
Bit j of register i	I/OWxy,i,j	
Bit k of common word j of station i	COMi,j,k COMXi,j,k (where X = B, C, D)	%NWi.j:Xk %NXWi.j:Xk
Bit j of system word i	SWi,j	%SWi:Xj

Note

(1) These objects must be reassigned by the user :
path = <rack_module>.<channel>.<connection_point>

(2) When the correspondence between PL7-3 and PL7 objects is not correct, the object %Si appears in red.

Words

Objects	PL7-3	PL7
Single length internal word	Wi	%MWi
Indexed single length internal word	Wi(Wj)	%MWi[%MWj]
Double length internal word	DWi	%MDi
Indexed double length internal word	DWi(Wj)	%MDi[%MWj]
Real internal word		%MFi
Indexed real internal word		%MFi[%MWj]
Single length constant word	CWi	%KWi
Indexed single length constant word	CWi(Wj)	%KWi[%MWj]
Double length constant word	CDWi	%KDi
Indexed double length constant word	CDWi(Wj)	%KDi[%MWj]
Real constant word		%KFi
Indexed real constant word		%KFi[%MWj]
Single length input register word	IWxy,i	%IW<rack_mod>.<channel> (1)
Double length input register word		%ID<rack_mod>.<channel> (1)
Single length output register word	OWxy,i	%QW<rack_mod>.<channel> (1)
Double length output register word		%QD<rack_mod>.<channel> (1)
Remote input register word	RIWx,y,i	%IW\<path>\<mod>.<channel>
Remote output register word	ROWx,y,i	%QW\<path>\<mod>.<channel>
System word	SWi	%SWi (2)
Common word j of station i	COMi,j COMXi,j (where X = B, C, D)	%NW{i}j (1) %NW{[r.]i}j r = network number
Status word of a remote discrete module	STATUSAx,y,i STATUSBx,y,i	
Status word of a remote disc.module channel	STSx,y,i	
Activity time of Grafcet steps	Xi,V	%Xi.T
Activity time of step i of macro-step j	Xj,i,V	%Xj.i.T
Activity time of input step of macro-step j	Xj,I,V	%Xj.IN.T
Activity time of output step of macro-step j	Xj,O,V	Xj.OUT.T

Note

(1) These objects must be reassigned by the user :
path = <rack_module>.<channel>.<connection_point>

(2) When the correspondence between PL7-3 and PL7 objects is not correct, the object %Si appears in red.

Function blocks

Objects	PL7-3	PL7
Timer	Ti	%Ti
• preset value word	Ti,P	%Ti.P
• current value word	Ti,V	%Ti.V
• timer running bit	Ti,R	%Ti.R
• timer done bit	Ti,D	%Ti.D
Monostable	Mi	%MNi
• preset value word	Mi,P	%MNi.P
• current value word	Mi,V	%MNi.V
• monostable running bit	Mi,R	%MNi.R
Up/down counter	Ci	%Ci
• preset value word	Ci,P	%Ci.P
• current value word	Ci,V	%Ci.V
• upcount overflow bit	Ci,E	%Ci.E
• preset bit reached	Ci,D	%Ci.D
• downcount underflow bit	Ci,F	%Ci.F
Register	Ri	%Ri
• input word	Ri,I	%Ri.I
• output word	Ri,O	%Ri.O
• register bit full	Ri,F	%Ri.F
• register bit empty	Ri,E	%Ri.E
Drum controller		%DRi
Text	TXTi	
• table length word in bytes	TXTi,L	
• status word	TXTi,S	
• module address word and channel no.	TXTi,M	
• request code word	TXTi,C	
• TELWAY station address word	TXTi,A	
• communication text block no. word	TXTi,T	
• exchange report word	TXTi,R	
• exchange over bit	TXTi,D	
• incorrect exchange bit	TXTi,E	
Control	CTRLi	
• activated task bit	CTRLi,R	

Bit and word tables

Objects	PL7-3	PL7
Bit strings		
• internal bit string	Bi[L]	%Mi:L
• input bit string	Ixy,i[L]	%Ixy.i:L
• output bit string	Oxy.i[L]	%Qxy.i:L
• Grafcet step bit string	Xi[L]	%Xi:L
• macro-step bit string	XMi[L]	
Character strings		%MBi:L (1) (with i even)
Word tables		
• table of internal words	Wi[L]	%MWi:L
• table of indexed internal words	Wi(Wj)[L]	%MWi[%MWj]:L
• table of double internal words	DWi[L]	%MDi:L
• table of double indexed internal words	DWi(Wj)[L]	%MDi[%MWj]:L
• table of constant words	CWi[L]	%KWi:L
• table of indexed constant words	CWi(Wj)[L]	%KWi[%MWj]:L
• table of double constant words	CDWi[L]	%KDi:L
• table of indexed double constant words	CDWi(Wj)[L]	%KDi[%MWj]:L
• table of reals		%MFi:L
• table of indexed reals		%MFi[%MWj]:L
• table of constant reals		%KFi:L
• table of indexed constant reals		%KFi[%MWj]:L
• table of remote input elements	RIx,y,i[L]	
• table of remote output elements	ROx,y,i[L]	
• table of indexed remote input elements	RIx,y,i(Wj)[L]	
• table of indexed remote output elements	ROx,y,i(Wj)[L]	

Optional function blocks

Objects	PL7-3	PL7
Optional function block	<OFB>i	
OFB element	<OFB>i, <element>	(2)
Indexed OFB element	<OFB>i,<element>(Wj)	(2)
Table of OFB elements	<OFB>i,<element>[L]	(2)
Table of indexed OFB elements	<OFB>i,<element>(Wj)[L]	(2)

Note

(1) In PL7, a character string is a series of adjacent bytes of the same type.

(2) <element> can be a bit, a word or a double word.

Instructions

Objects	PL7-3	PL7
Instructions on bits		
• Logical inversion	NOT	NOT
• AND	•	AND
• OR	+	OR
• Exclusive OR		XOR
• Rising edge	RE	RE
• Falling edge	FE	FE
• Set to 1	SET	SET
• Reset	RESET	RESET
Instructions on words and double words		
• Addition	+	+
• Subtraction	-	-
• Multiplication	*	*
• Division	/	/
• Comparisons	>, >=, <, <=, =, <>	>, >=, <, <=, =, <>
• Remainder of a division	REM	REM
• Square root	SQRT	SQRT
• Absolute value		ABS
• Logic AND	AND	AND
• Logic OR	OR	OR
• Exclusive logic OR	XOR	XOR
• Logic complement	CPL	NOT
• Incrementation	INC	INC
• Decrementation	DEC	DEC
• Logic shift to the left	SHL	SHL
• Logic shift to the right	SHR	SHR
• Circular shift to the left	SLC	ROL
• Circular shift to the right	SRC	ROR
Floating point type instructions (1)		
• Addition	ADDF	+
• Subtraction	SUBF	-
• Multiplication	MULF	*
• Division	DIVF	/
• Square root	SQRTF	SQRT
• Absolute value		ABS
• Equality test	EQUF	=
• Strict superiority test	SUPF	>
• Strict inferiority test	INFF	<
• Other tests		>=, <=, <>

Note

(1) Floating point objects do not exist in PL7-3. These instructions handle PL7-3 objects of the following types : internal double word (DWi), constant double word (CDWi) or OFB extract double word.

Instructions (cont'd)

Objects	PL7-3	PL7
Instructions on byte strings		
• Circular shift	SLCWORD	
Conversion instructions		
• BCD binary conversion	DTB	BCD_TO_INT
• Binary BCD conversion	BTD	INT_TO_BCD
• ASCII binary conversion	ATB	STRING_TO_INT or STRING_TO_DINT
• Binary ASCII conversion	BTA	INT_TO_STRING or DINT_TO_STRING
• Gray binary conversion	GTB	GRAY_TO_INT
• Floating point integer conversion	FTB	REAL_TO_INT or REAL_TO_DINT
• Integer floating point conversion	FTF	INT_TO_REAL or DINT_TO_REAL
• BCD floating point conversion	DTF	BCD_TO_REAL
• Floating point BCD conversion	FTD	REAL_TO_BCD
• ASCII floating point conversion	ATF	STRING_TO_REAL
• Floating point ASCII conversion	FTA	REAL_TO_STRING
Instructions on tables		
• Arithmetic operations	+, -, *, /, REM	+, -, *, /, REM
• Logic operations	AND, OR, XOR	AND, OR, XOR, NOT
• Addition of words in a table	+	SUM
• Search for the 1st different word	EQUAL	EQUAL
• Search for the 1st equal word	SEARCH	FIND_EQU
Instructions on program		
• Jump	JUMP Li	JUMP %Li
• Call subroutine	CALL SRi	SRi
• Subroutine return	RET	RETURN
• Halt the application	HALT	HALT
• Conditional statement	IF / THEN / ELSE	IF / THEN /ELSE /END_IF
• Iterative statement	WHILE / DO	WHILE / DO / END_WHILE
Instructions on interrupts		
• Test	READINT	
• Mask	MASKINT	MASKEVT
• Unmask	DMASKINT	UNMASKEVT
• Acknowledge	ACKINT	
• Generate an IT to the module	SETIT	
Explicit I/O instructions		
• Read discrete inputs	READBIT	
• Write discrete outputs	WRITEBIT	
• Read registers	READREG	
• Write registers	WRITEREG	
• Read words	READEXT	
• Write words	WRITEEXT	

Instructions (cont'd)

Objects	**PL7-3**	**PL7**
Instructions on function blocks		
• Preset	PRESET Ti / Ci	PRESET %Ti / %Ci
• Start	START Ti / Mi	START %Ti / %MNi
• Activate the task	START CTRLi	
• Reset	RESET Ci / Ri / TXTi	RESET %Ci / %Ri
• Deactivate the task	RESET CTRLi	
• Upcount	UP Ci	UP %Ci
• Downcount	DOWN Ci	DOWN %Ci
• Store in a register	PUT Ri	PUT %Ri
• Retrieve from a register	GET Ri	GET %Ri
• Receive a message	INPUT TXTi	
• Send a message	OUTPUT TXTi	
• Send/Receive a message	EXCHG TXTi	
• Execute an OFB	EXEC <OFBi>	
• Read telegrams	READTLG	

3.2 Differences between PL7-3 and PL7

1 Application structure

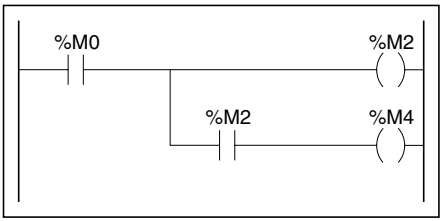
A PL7-3 application comprises a maximum of 7 tasks (MAST, FAST, AUX0 to AUX3 and an interrupt task), whereas a PL7 application comprises :

- 2 tasks (MAST and FAST) plus event processing

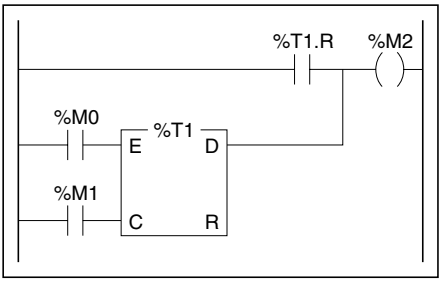
2 Ladder Language

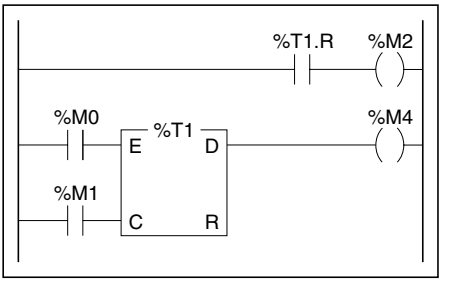
- **Order of evaluation of Ladder language rungs**
  In PL7-3, evaluation is performed from **left to right**, **column by column** and **from top to bottom** of each column. In PL7, it is performed **rung by rung** and **in the direction of the equation** for each rung. Although they are converted in an identical graphic form, some Ladder language rungs can be evaluated differently (give a different result on execution). These Ladder language rungs contain an object which is simultaneously evaluated and updated (using a coil or a function block).

  Updating performed by a coil
  - with PL7-3, the B2 contact (%M2) was evaluated before coil B2 (%M2),
  - with PL7, coil %M2 is updated before contact %M2 is evaluated.



  Updating performed by a function block
  - with PL7-3, timer T1 (%T1) was called before contact T1,R (%T1.R) was evaluated,
  - with PL7, contact %T1.R is evaluated before timer %T1 is called.





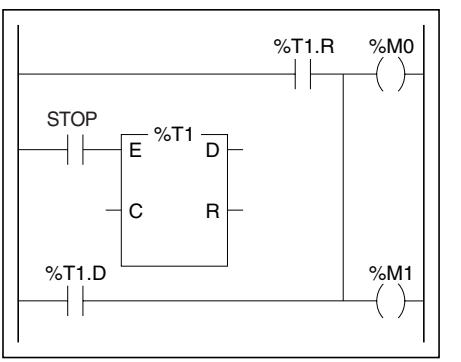
Note : these different evaluations are not detected by the converter.

- **JUMP coil**
  In PL7-3, the JUMP coil is symbolized by -(J)- and supports a connecting label. In PL7, its new symbol is -->> %Li. It still enables jumps to a labelled rung in the same programming entity.

- **OPERATE blocks**
  OPERATE blocks, containing the instructions HALT or RET in PL7-3, are replaced in PL7 by special coils (--<HALT>- and -<RETURN>-).

- **Comment attached to a Ladder language rung**
  In PL7 a comment attached to a Ladder language rung does not reduce the size of the entry zone. This remains as 7 lines of 11 columns, unlike PL7-3 where there are only 6 lines.

- **Nesting rungs**
  Due to the different evaluation logic, some Ladder language rungs allowed in PL7-3 are not permitted in PL7. Since the converter does not take into account the automatic reorganization of such Ladder language rungs, the user must perform the necessary modifications to eliminate nesting.



3 Grafcet language

- **Structure of a page**
  The structure of the Grafcet page differs between PL7-3 and PL7 :
  - the number of columns changes from 8 in PL7-3 to 11 in PL7. However the vertical page structure remains the same (alternating between 6 lines of steps and 6 lines of transitions).
  - there are no longer distinct fields for charts and comments (this is the reason for the increase in the number of columns). In PL7, comments can be entered in any cell on the page. However for reasons of similarity, comments from a PL7-3 module are placed by the converter from the ninth column onwards in PL7. The maximum size for a comment has been increased by 4 characters, thus changing from 60 to 64 characters. In fact, PL7 syntax requires that a comment be surrounded by character strings (* and *).

- **Transition conditions**
  In PL7-3, an empty transition condition (ie unprogrammed) is considered as always being true : the transition is cleared as soon as it is validated.

  In PL7, an empty transition condition acts as a block, ie it is always false. For it to be cleared a Ladder or Structured Text program containing a single # coil must be assigned to it. The role of this coil is to force movement to the following step.

4 Structured Text language

- **Assignment**
  In PL7-3, assignment (or multiple assignment) is defined by the instruction -> : `<source> -> <target>` or in the case of multiple assignment `<source> -> <targ1> -> ... -> <targn>`

  In PL7, the assignment instruction becomes := and the assignment direction changes : `<target> := <source>` or in the case of multiple assignment `<targn> := ... := <targ1> := <source>`

- **Comment**
  In PL7-3, the comment is associated with each Structured Text statement and is the first line of this statement. It contains a maximum of 78 characters.

  In PL7, the comment is restricted by delimiters (* and *) and can take up several lines anywhere in the statement. It can be up to 256 characters.

- **Comparison**
  In PL7-3, comparisons are written in square brackets, whereas in PL7 they are written in rounded brackets (which are not compulsory) :
  `[<expr1> <oper> <expr2>]` in PL7-3
  `(<expr1> <oper> <expr2>)` or `<expr1> <oper> <expr2>` in PL7
  `<oper>` : `< / > / <= / >= / = / <>`

- **Boolean operators**
  In PL7-3, the arithmetic symbols * and + are used to write the logic AND and the logic OR. In PL7, the logic AND becomes AND and the logic OR becomes OR.
  Moreover in PL7-3, the exclusive OR (XOR) has a lower priority than the logic OR, whereas the opposite is true in PL7 : XOR has a higher priority than OR.

- **Terminators**
  In PL7-3, the instructions IF, WHILE, REPEAT and FOR have no terminator. However in PL7 a terminator is compulsory :
  - IF ends with END_IF,
  - WHILE ends with END_WHILE,
  - REPEAT ends with END_REPEAT, and
  - FOR ends with END_FOR.

- **Calling a subroutine SRi**
  In PL7-3, a subroutine is called by writing CALL SRi. In PL7, "CALL" disappears and this instruction becomes SRi.
- **Separators for instruction parameters**
  In PL7-3, instruction parameters are separated by semi-colons (;), while in PL7, the separator is the comma (,).
- **Instructions on IT**
  PL7-3 instruction on interrupts (READINT, MASKINT, DMASKINT, ACKINT and SETINT) do not exist in PL7 : system bits are used to program interrupts.
- **Text block**
  The text block (TXT), used in PL7-3, no longer exists in PL7.
- **Explicit I/O instructions**
  PL7-3 explicit I/O instructions (READBIT, WRITEBIT, READREG, WRITEREG, READEXT et WRITEEXT) no longer exist in PL7.
- **Operations on floating point objects**
  The PL7-3 instructions : ADDF, SUBF, MULF and DIVF no longer exist in PL7. In this language, operations on floating point objects use the usual arithmetic symbols (+, -, * and /).
- **Action separators**
  The semi-colon (;) is used to separate actions in both PL7-3 and PL7. However, it is used much more frequently in PL7 : **all** actions are followed by an ";" which is a **terminator** character. Even a single action is ended with a semi-colon.

5 Symbols

The converter is used to convert a symbol file, with the exception of certain special characters, permitted in PL7-3 and prohibited in PL7 :

PL7-3 character	PL7 replacement character
~ (tilde)	é
| (pipe)	è
_ (underscore)	à
$ (dollar)	ê
% (percentage)	î
# (sharp)	â

For example :

`Pump_A`	becomes	`PumpàA`	in PL7
`Valve$$`	becomes	`Valveêê`	in PL7
`#1234`	becomes	`â1234`	in PL7

Symboles

A

B

C

D

E

F

G

S

T

U

W

W915905150301A 03

Schneider Electric Industries SAS

Headquarters

89, bd Franklin Roosevelt
F - 92506 Rueil Malmaison Cedex

http://www.schneider-electric.com

Printed in March 2005