ORPHEE applications converter

May 2007 eng

35015478 00

a brand of Schneider Electric

Telemecanique

1.1 Object of the Converter

The Orphee Converter is made up of two parts:

1. The Ladder Converter which is used for the partial retrieval of combinational processing of a Series 1000 Application (Program Entities with associated Data) thus ensuring similar operation on ATS.
   The processing operations concerned are exclusively combinational and describe:
   - a combinational logic which corresponds to one part of an ENTITY,
   - or a series of ENTITIES whose functional behavior in PL7 will be the same as in the original S1000.

2. The Variables Converter which is used to retrieve the variables in an Orphee Application (S1000) with their associated symbols and comments.

A reassignment (or association) function for the variables involved in the conversion helps the user to achieve the best possible organization of these variables in PL7.

1.2 Installation

The Orphee Converter is an optional tool for PL7 Junior. As such, it can be installed:

- at the same time as PL7. In this case it simply needs to be selected from the list of elements suggested by the installation procedure.
- or after PL7. Only the Converter need be selected from the list of elements to be installed. The procedure checks that PL7 Junior is installed on the PC. If PL7 Junior is not present, installation is refused.

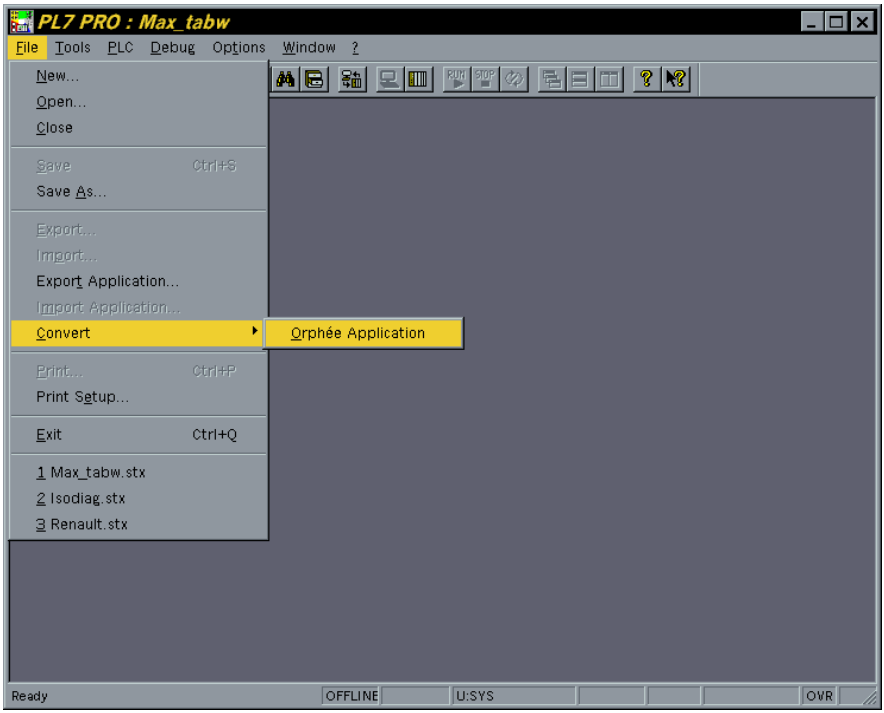
1.3 Accessing the Converter

To access the Orphee Converter, a PL7 application (new or existing) must be opened. Then go to Convert on the File menu. This item offers the choice of converting a PL7_2, PL7_3 or Orphee (S1000) application depending on the converter(s) installed.

Procedure:

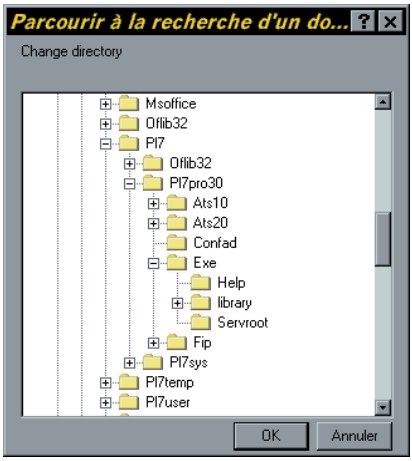
To activate the Orphee Converter:

1. open a PL7 application (new or existing),
2. select the **File** menu,
3. select **Convert**,
4. select the **Orphee Application** item.



After launching the tool for converting an Orphee application, the main window is displayed.
This windows allows the user to select a local Orphée application (...\SERVROOT).

For applications in another directory, you may click on the "Go through" button that aperes the apposite screen, alllowing to select a distant application



1.4 Coexistence with Orphee

1.4-1 Sharing information

The execution of Orphee and the Converter is mutually exclusive, as both these tools are single-user. For example, if Orphee is already running, the following message will be displayed:

"The Orphee Converter cannot be run at the same time as ORPHEE".

Orphee and PL7 can run at the same time. The exclusion is only effective when the Converter is activated.

If Orphee is installed, the Converter and Orphee share the same “Orphee.ini” file in the Windows directory.

With “remote Servroot” (see section 2.3), this file enables the Converter to share the same application installation space as Orphee.

With “local Servroot”, each tool has its own application installation space.

1.4-2 Activation

The Orphee workshop need not be installed on the same PC for the Converter to run.

The Converter does not require the use of a dongle.

The Orphee application need not be validated in order to convert the application.

2.1 Description of stages

Section 2 describes the various processing operations performed by the Converter. Logical and dynamic sequencing is explained in section 3.

The various processing operations performed by the Converter are divided into 4 main stages:

1	Selection of elements to convert
2	Extraction and Analysis
3	Configuration of the Conversion
4	Conversion

When selecting the elements to be converted, the user specifies which parts of the application he wishes to retrieve.

The Converter undertakes the whole extraction and analysis stage. This consists of retrieving the selected elements from the Orphee database and analyzing them.

The next stage invites the user to reassign the variables in question and to provide other information necessary for the conversion.

The final stage corresponds to the conversion itself, ie. the generation of code and/or variables in the PL7 source format.

The results of the conversion should be imported using the appropriate PL7 tool (Ladder Editor or Variables Editor), in order for them to appear in the target application.

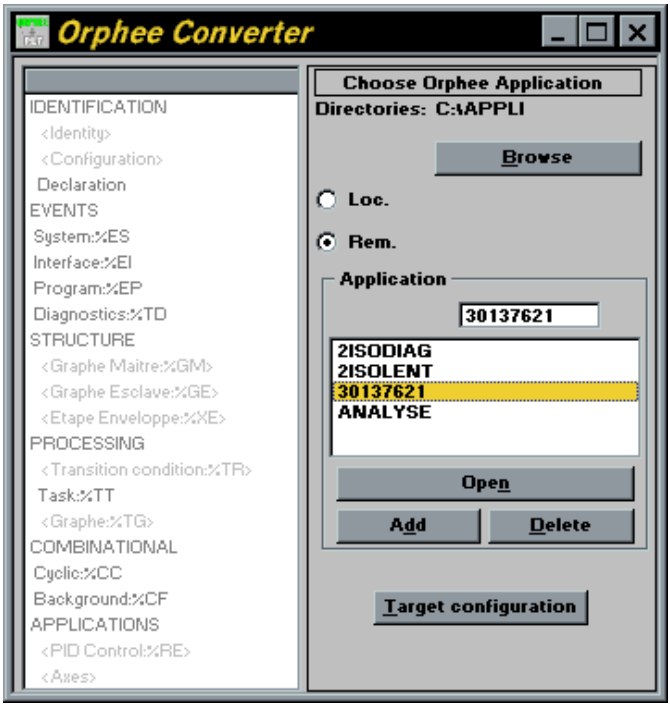
NB : Each page which describes a processing operation has a footer at the bottom of the page which highlights the stage it belongs to.

Eg:

1 - Selection of elements to convert	2	3	4

2.2 Retrieving Orphee applications

This stage is not essential for retrieving applications if they can be accessed using the “remote servroot” mechanism. However, the Converter works faster with its own servroot. The tool offers the opportunity of including the applications in it. To do this, it uses the same function for archiving and retrieving Orphee applications, which can be accessed via the “**Add**” button.



For more detailed information on this function, refer to the Orphee user manual.

Reminder: The servroot space only accepts a maximum of 16 applications.

To manage the applications in the servroot space, the tool has a “delete” function which is used to delete applications which are no longer useful.

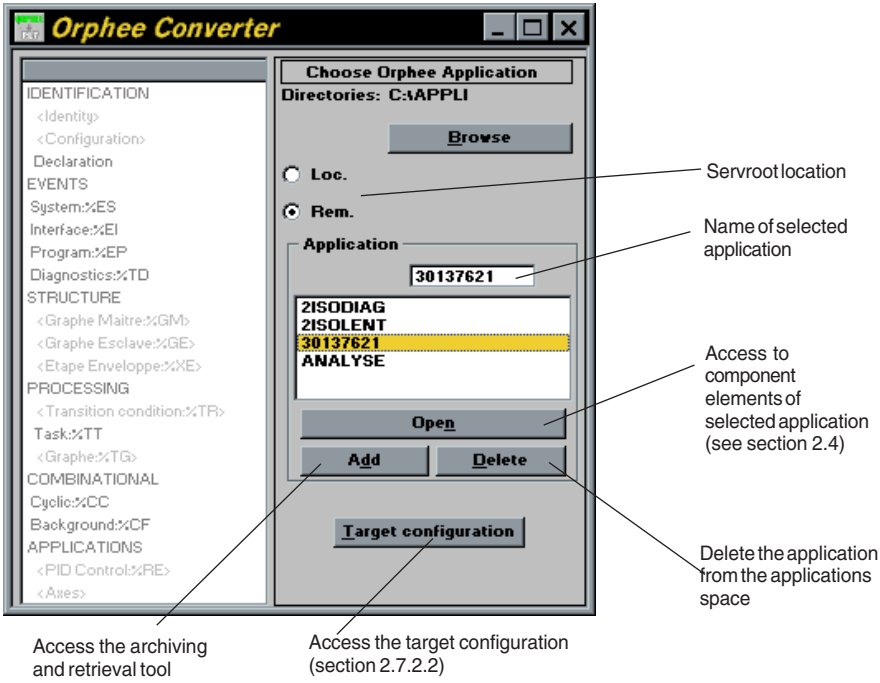
Use of the DOS or Windows “File Manager” is not recommended, as certain checks are not performed.

1 - Selection of elements to convert	2	3	4

2.3 Selecting the Orphee application

The left-hand side shows the generic structure of an Orphee application in which the title bar will contain the name of the selected Orphee application.

The right-hand side is reserved for selecting the initial Orphee application.



This window integrates three functions for managing the application **Open, Add** and **Delete** and a field which will display the application name.

“Open” gives access to the component elements of the application
“Add” is used to retrieve an Orphee application
“Delete” deletes the application from the applications directory

Reminder:

1 - Remember that the ORPHEE.INI file contains information used dynamically by Orphee and the Converter to determine the location of the applications.

2. The remote servroot is limited to 8 characters.
   Any path which does not respect this constraint is not recognized.
   Example: “c:\servd’.

2.3-1 Local Servroot

If ORPHEE.INI is missing or does not contain the heading “Servroot=”, the Converter automatically assigns the SERVROOT “local” directory to PL7. The “**Loc.**” button is then selected.

The path is shown under the heading: “Directory” (ASAW\ASAW21\EXE\SERVROOT in our example).

2.3-2 Remote Servroot

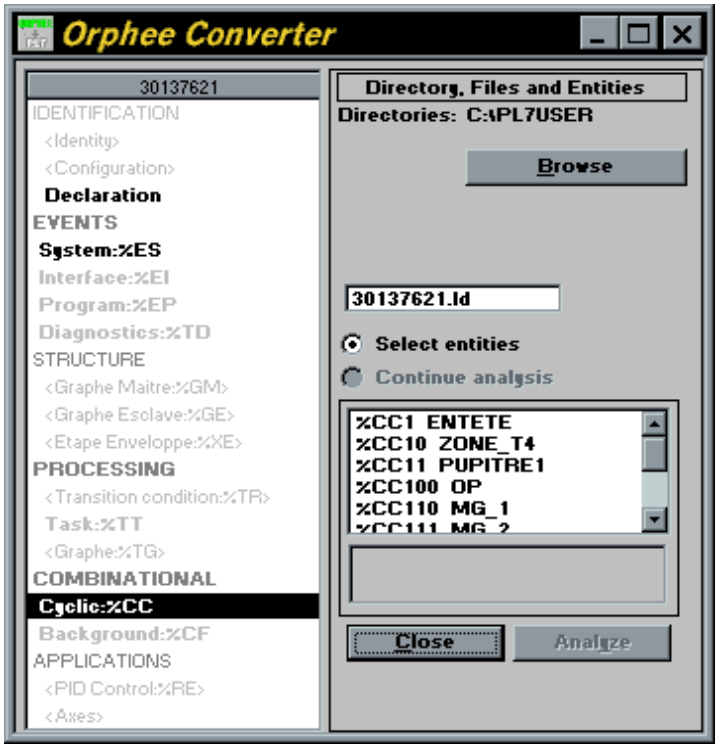
If ORPHEE.INI exists and the “Servroot=” heading is filled in, the Converter automatically selects the path defined by the heading. The “**Dist.**” button is then selected. If the “**Dist.**” button is selected voluntarily by the user, the Converter shows a dialog box for selecting the drive and access path.

Information on the location of the current servroot is saved in the ORPHEE.INI file. This space may exist on any drive which can be accessed from the PC.

If Orphee is activated between two sessions of the Converter, this information can be modified by the user. It is always the last servroot to be used which is memorized.

2.4 Displaying the component elements of the application

The **Open** function is used to access the contents of the Orphee application. The application structure is updated at the same time. The “convertible” program elements are shown in **Bold** and the program elements which the Converter does not recognize are <grayed out>.



Once an application is open, the left-hand side of the window shows the generic structure of the application with a field in the header where the name of the selected application will be indicated.

Within this generic structure, the component elements are grouped by “generic category” : Declaration for the variables, %ES for the System Events Entities, %CC for the Cyclic Combination Entities, etc.

2.4-1 Entering the destination file

The file generated by the conversion has the extension:
- **.ld** for relay entities,
- **.scy** for variables.

It is stored in the PL7 “Source Directory” (see the “Customize” item on the PL7 “Options” menu).
By default, the Converter offers the name of the Orphee application with the appropriate extension.
It is possible to make several conversions (several passes) before executing the import operations, especially for entities in which the generic categories cannot be mixed. In this case, a different file name must be given for each pass as the Converter overwrites the contents of the file.

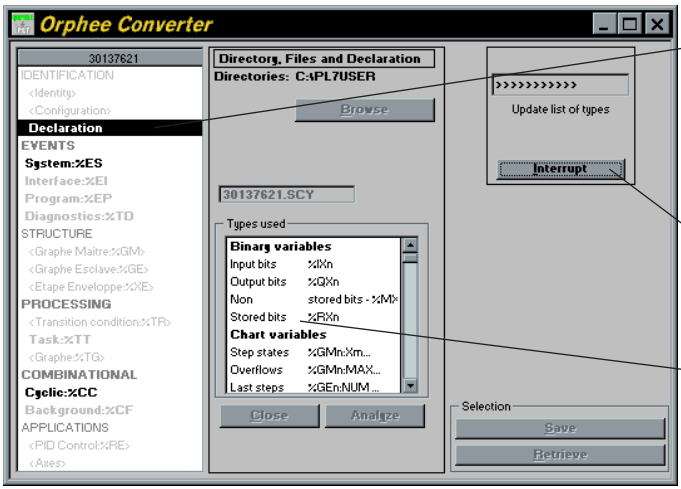
Procedure :
To enter the destination file:
- select the drive and select the directory with the "Go through" button,
- enter the file name.

2.4-2 Declaration

Procedure :

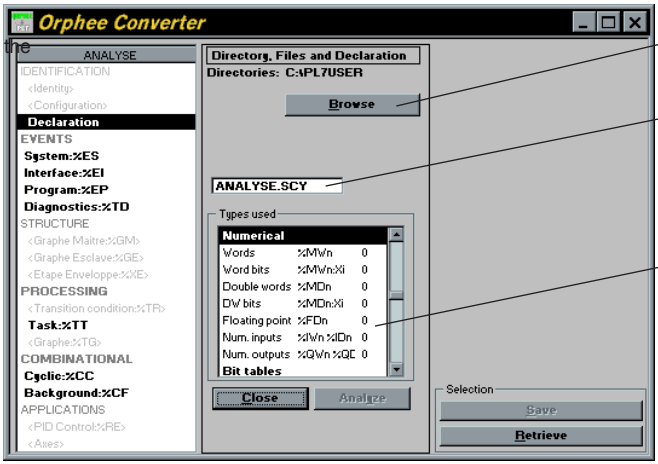
To access the list of data:
- select the “Declaration” heading,
- wait for updating of the list of variable types to be completed. This process can be interrupted,
- enter the name of the file which will contain the converted variables.

The variable types are presented in a list, grouped by family.



When updating is complete, the Converter displays the following in the central section:

- the file access system resulting from the conversion (see section 2.4.1: entering the destination file)
- list of variable types grouped by family, together with the number of variables of each type selected.



File access system with "Go through" button

Name of resulting file

Number of variables selected

2.4-3 Relay entities

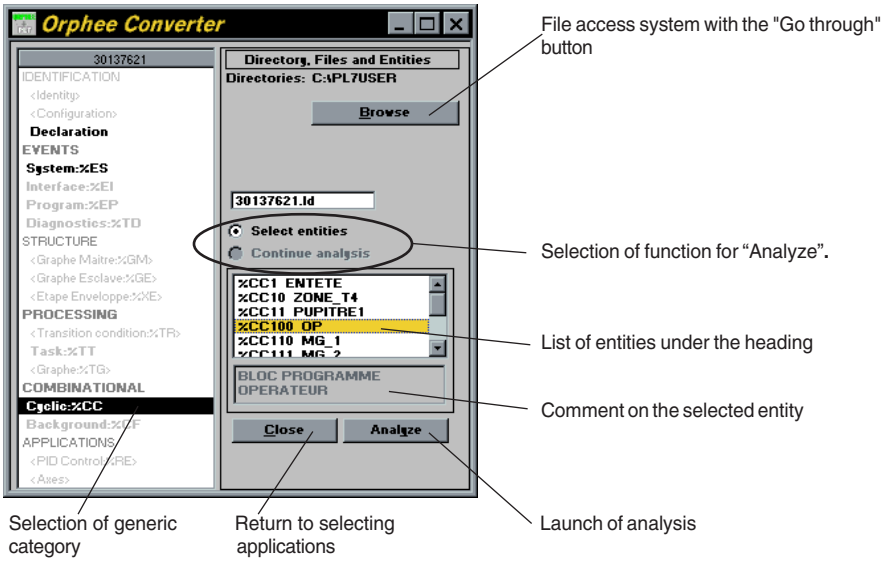
Procedure :

To access the relay entities:
- select a generic category (%ES, %CC, %CF, %TD, etc.),
- enter the name of the file which will contain the code for the selected entities and the variables used in them.

The **Analyze** button has two functions:

on **Select Entities** it extracts the selected entities from Orphee and then displays the conversion parameters.

on **Resume Analysis** it redisplays the parameters of the previous conversion.



As long as analysis is not requested, it will remain possible to return to the list of applications by selecting the “**Close**” button.

2.5 Selecting the elements

The selection of at least one variable or one relay entity accesses the “**Analyze**” and “**Record**” buttons. The analysis procedure is described in section 2.6: Extraction and Analysis.

2.5-1 Declaration

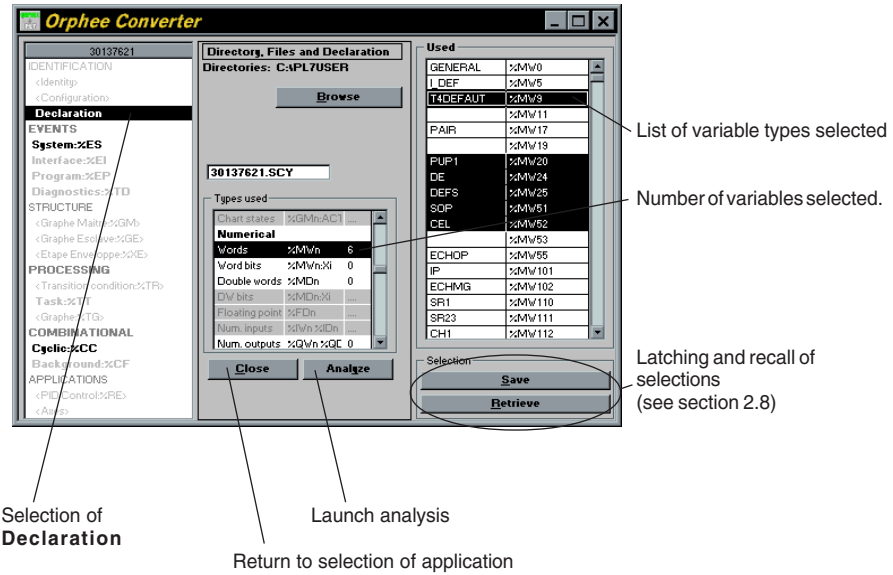
Variables are selected by type.

To create the list of variables to be translated:

Select “Declaration”.

Select a variable type (%IXn, %QXn, %RXn, etc.) under the heading “Types used”. The selection of a type of variable displays all the variables of this type used in the Orphee application.

Select the variables from the right-hand side of the screen. The number of variables selected in this way is displayed in real time under the heading “Types used”, together with the corresponding variable types.



2.5-2 Relay Entities

Entities are selected by generic category. Entities from different generic categories cannot be converted at the same time.

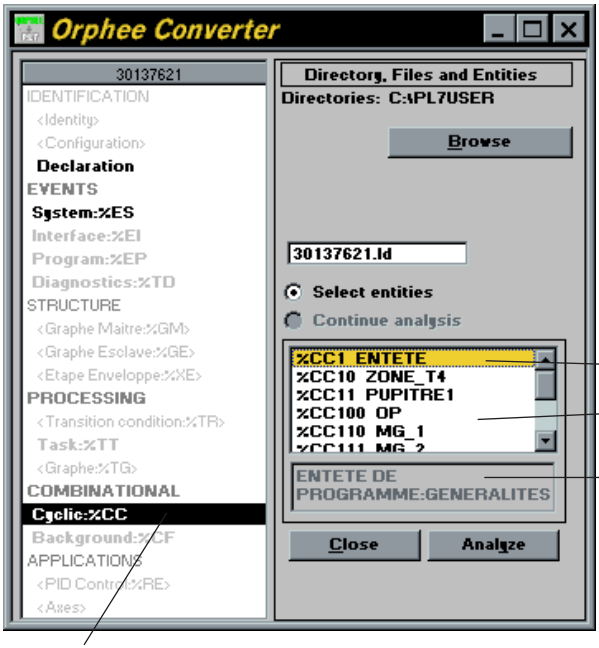
Procedure :

To create the list of relay entities to be translated:

- select a generic category
- select the entities from the list displayed on the right-hand side.

N.B.

The comment on the last selected entity is displayed for information.



The generic categories which can be translated by the Converter are as follows:

- events (%ESn,%EIn, %EPn, %TDn),
- processing (%TTn),
- combinational entities (%CCn, %CFn).

2.6 Extraction and Analysis

All Orphee elements must be extracted from the application before being analyzed by the Converter.

For variables, elements are extracted when the types are updated.

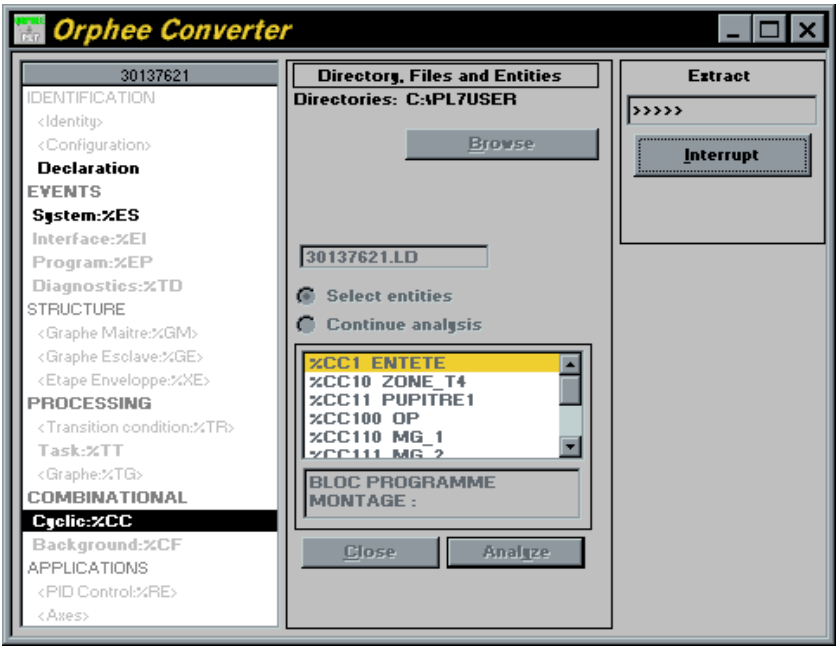
For entities, extraction is launched automatically once the analysis is requested.

The extraction process is displayed by a progress bar. It can be interrupted. The analysis provides:

- declarations of the original application,
- indeterminate logic states,
- indeterminate functions.

Procedure :

- click on the "**Analyze**" button when the elements to be converted are selected.



1	2 -Extraction and analysis	3	4

2.7 Setting the conversion parameters

After analysis, a window shows:

- **in bold :** any element which causes no problems in conversion.

- **in "red" :** any element which requires the intervention or attention of the user, namely:
  - program elements involved in the conversion for which no correspondence has been found in the software or hardware configuration of the target application,
  - parameters which must be entered,
  - variables which are in conflict with the target application.

- **"grayed out" :** any element which is not affected by the current conversion.

2.7-1 Conversion rules

Variables:

See section 2.10.1 for correspondence of variables.

Expressions:

See section 2.10.2 for correspondence of combinational logic elements.
The %Ecn expressions containing the CFB calls are translated by:
- Operate blocks: language, Function or Comparison operators,
- Language blocks (FB).

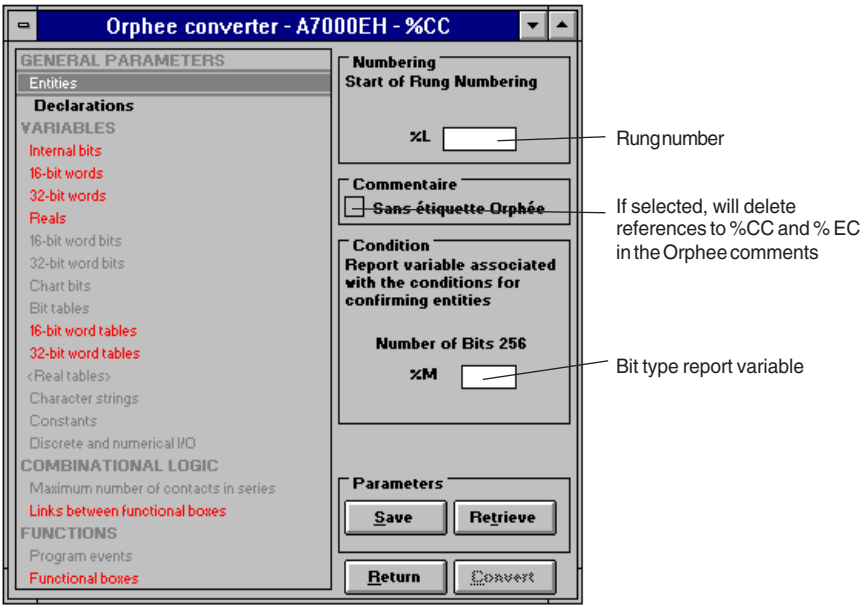
An expression containing a succession of CFBs with **"named links"** is translated by a cascade of **"Operate Blocks"** in a rung.

2.7-2 Setting general parameters

By general parameters, we mean the following information:
- the source of the Rung numbering in the PL7 code,
- the “report” variable used to convert conditions for enabling the Entities,
- the target configuration,
- the type of import for declarations.

Entities

To set the entities parameters, the following must be entered:
- the source of the Rung numbering in the PL7 code
- the report variable for the enabling condition

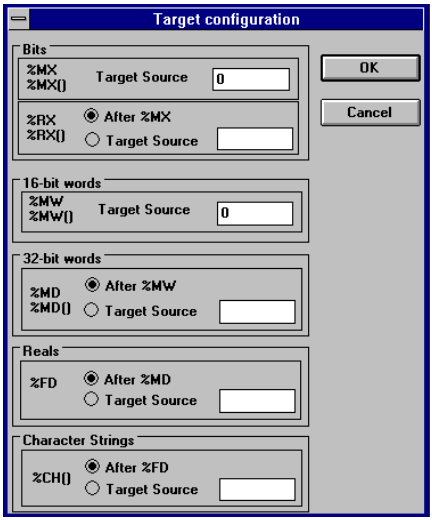


Determining the target configuration

The “Target Configuration” option is used to set the parameters for the ranges of variables starting with which the Converter will perform the default correspondences. The default parameter setting is:

. %MX start at address 0,
. %RX “following %MX”,
. %MW start at address 0,
. %MD “following %MW”,
. %FD “following %MD”,
. %CH “following %FD”.

This setting can only be accessed during modification before an application is opened. Subsequently, the target configuration can be viewed via the “Utilities” menu.



In Orphee, the variables space is divided into blocks, one block being reserved for one type.

Example of distribution of	%MWn	from 0	to	4999
manufacturer names of	%MDn	from 5000	to	5999
Orphee objects	%FDn	from 6000	to	6999
(A5000 without RAM extension):	%CHn	from 7000	to	11262

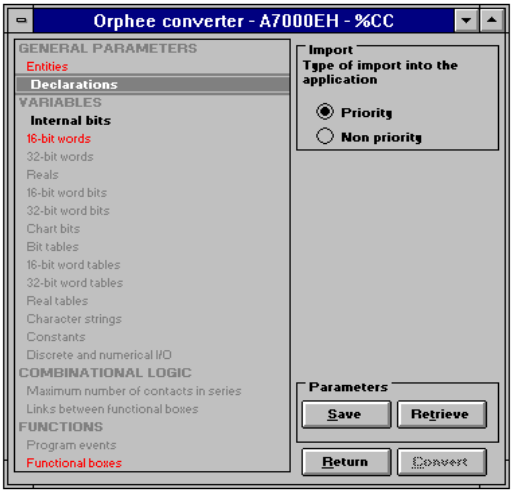
Under PL7, the user can mix the types, using %MW0, %MD2 and %MW5.

If the Converter is able to associate %MD5000 (Orphee) with %MD5000 (PL7), the majority of associated addresses will exceed the PLC capacities, thus obliging the user to re-enter all the variables. Also, the Converter uses a target configuration (default or entered by the user), which specifies for each type whether it must be put **following** (default) or start at the **Origin:.**

Thus if the user wishes the %MD to be placed “following” the %MW, the Converter finds the largest %MW (single word or table) and starts the space for the %MD at the next address. If the user wishes to reserve some space for other declarations, he can then enter the **Origin:** of the %MD, ie. the index of the first %MD generated. If this address is not sufficiently large, it generates conflicts with the words or word tables, and the Converter offers the user a recommended configuration (minimum) which can be modified before relaunching the analysis.

Recommended procedure:
- select the default values,
- launch the selection and the analysis,
- in the event of conflicts, the Converter suggests a new configuration depending on the variables to be translated.

Declarations



Setting the parameters for the declaration means selecting the mport mode from the following 2 modes:

Priority Mode:
If the variable has already been declared in the target application, it will be overwritten by the imported variable

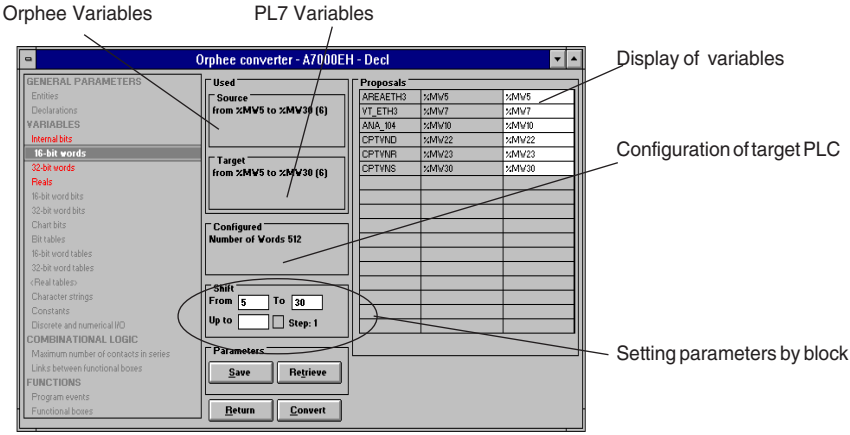
Non Priority Mode:
The Converter is limited to the addition of new variables.

2.7-3 Setting variable parameters

The parameters are set by type of variable. For each type, the following information is required:

- its operating context in the selected Entities and in the target application after conversion.
- the configured number of variables of the same type corresponding to it in the target application.

Example :



Two methods are given to set the variable parameters. These methods consist of offering:

1. a transcription, variable by variable. This is presented with the option of modifying the target index of the selected variable.
2. a shift, block by block, of all or part of the variables towards a destination index specified by the user (from .. to .. towards ...).

The procedure for the 1st method is as follows:

- select a variable from those displayed
- enter the target index in the entry window which is provided for that purpose.

Multiple selection of suggested variables is not possible.

For the 2nd method, there are two ways of performing the shift operation:

1 - without the **increment by 1** option
shift the index of the 1st data item towards the target index and those of the other variables with the same shift value while maintaining the original deviations.

Example:
Block to shift: %MX10, %MX12, %MX13, %MX24 towards 100
Result block: %MX100, %MX102, %MX103, %MX114

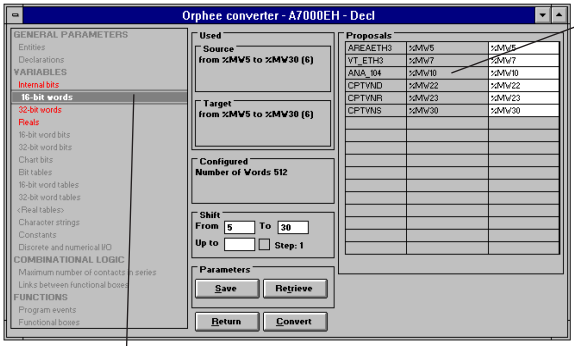
2 - with the **increment by 1** option
shift the index of the 1st variables towards the target index. For the other variables, the next number is assigned automatically.

Example:
Block to shift: %MX10, %MX12, %MX13, %MX24 towards 100
Result block: %MX100, %MX101, %MX102, %MX103

N.B.:
- Whatever modification is made to the index of a variable (individual or by block), ***dual assignment is prohibited (including in the tables).***
- Every variable included in the block is automatically reassigned, even if its index has already been modified separately.

User interface, general example

The general interface for setting the variable parameters is as follows:



Display of variables of selected type

Selection of variable category

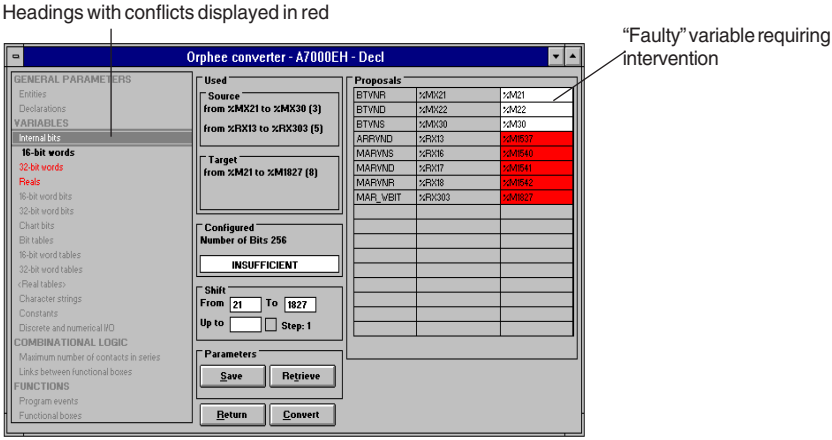
The generic categories involved are as follows:

- internal bits,
- 16-bit words,
- 32-bit words,
- reals,
- 16-bit word bits,
- 32-bit word bits,
- bit tables,
- 16-bit word tables,
- 32-bit word tables,
- real tables,
- character strings,
- constants.

The parameter settings associated with the configuration data are displayed and modified by selecting the corresponding line (not “grayed out”). The parameter setting is then displayed to the right of the list.

If any user intervention is required, the “faulty” parameters are indicated in red.

If a configuration data item appears “grayed out” and between “<>” this means that the objects are present in the original application but that the type cannot be translated.



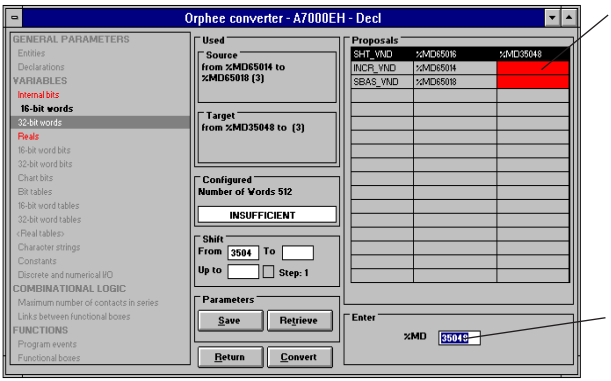
If a “fault” is eliminated, the color is updated in real time.

The Converter displays the message **“INSUFFICIENT”** as well as the number of objects configured if one suggested variables index is greater than the number of configured variables.
This can be corrected in 3 different ways

1. by reassigning the variables within the limits of the configuration,
2. by modifying the configuration,
3. by “clearing” (" Del " and " Enter " keys) the variable(s) which exceed the configuration. In this case, the Converter will generate special symbols for these variables, in the following form: ORPHEE_”variable name”, which will need to be replaced when the code is imported.

As with any other “fault”, this situation is blocking and inhibits access to the “Convert” function.

Heading with conflicting variables under modification.



Variables for which parameters need to be set.

Individual variable parameter setting.

Procedure:

- select a heading in red,
- select each variable in red,
- enter a permissible value in the individual parameter-setting window, or “clear” the variable.

NB:

When setting the conversion parameters, any modification made to the hardware or software configuration by the user (simultaneous use of the Converter and the Configuration Editor) will result in automatic updating of the configuration parameters and the current display in the Converter.

Setting Chart bit parameters

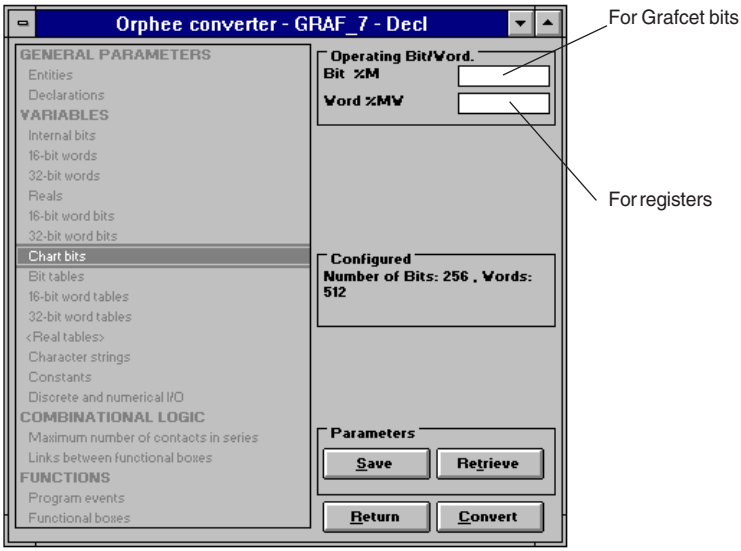
PL7 only manages one level of chart. The chart bit parameters are set by providing two operating variables :

- a bit-type variable which replaces all the Gratcet bits,
- a word-type variable which replaces all the Grafcet word-type registers (see 2.10.1).

Procedure:

- select the **Chart bit** heading,
- enter the values for the required variables. These values must correspond to the configured variables in the target PLC.

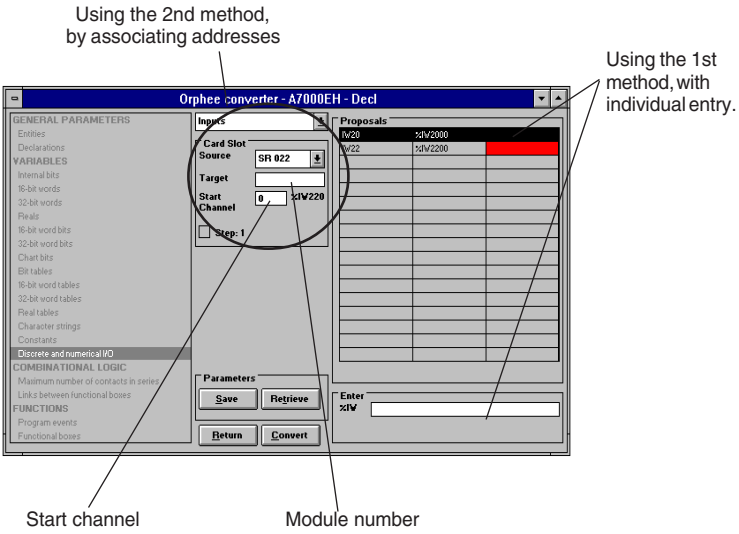
The interface is as follows:



Setting the parameters for Numeric I/O

This is performed in one of two ways:

1. Variable by variable with the option of modifying the topological index.
   The parameters are set in the “Proposals” field. The entry window of the index can be accessed by selecting a variable from the list.
2. By associating the module topological addresses in the hardware configurations.
   The “*Card Slots*” field gives the addresses of the I/O cards used in the Orphee application. For each of these, the start channel field specifies the index of the first variable in the module. The target field specifies the number (address) of the target module.
   The parameters are set in the “*Card Slots*” field by associating the module slots followed by a suggestion for the corresponding start channel. This can be modified.
   It is possible to “compress” the target addresses by selecting the **Increment by 1** option. This option automatically suggests adjacent channel numbers.
   In the event of an association problem (eg. result variable has a configuration fault), incorrectly “associated” variables will be displayed in the list in red.



NB : The target and start channel fields must match the configuration of the target PLC.

2.7-4 Limitations and restrictions

When translation is activated, the expressions are ranked in three categories:
- those which cannot be translated,
- those which require parameter-setting,
- those which can be translated.

Compatibility with processors V 1.5:

- Some instructions will not be accepted during the import function, when the hardware configuration is a V 1.5 processor.

These instructions are :
- immediate index,
- ROL and ROR when the number of bits is a variable %MW,
- PULSOR.

The conditions which require parameter setting are as follows:

Expression of 11 contacts and one coil	During the conversion of a %EC comprising 11 contacts and one coil, the Converter structures it in 2 rungs with a “report” variable.
Sequencing of CFBs	During conversion of an expression comprising a sequence of “convertible” CFBs with a DIRECT binary link (NOT filled in) the Converter requests the entry of a work bit. The maximum number of report variables is calculated on the most “complex” expression.
Timer and Up/Down counter CFB	Entry of a working word (16 bits word for CFB TON) or 32 bits word for CFB PULSOR) or instance number (see section 2.7.6.1)

The conditions which prevent translation are as follows:

Logic expressions	• Expressions containing a %CONT, • Expressions containing at least one Chart register initialization: MIN, MAX and NUM (%ACQGRAF),
Using CFBs and UFBs	• Expressions containing UFBs, Expressions containing non-convertible CFBs (see 2.7.6.2 list of convertible CFBs), CFB/CFB linking with indirect binary links.

⚠ AVERTISSEMENT

UNEXPECTED APPLICATION BEHAVIOR

Conversion of the PULSOR function sets the Enable parameter of this function to TRUE. This means that even if the function is no longer executed, the output will remain in the same state.

Observe the following instructions to ensure that the PULSOR function works correctly.

Failure to comply with this directive may result in death, serious injury or equipment damage.

When converting a PULSOR CFB, the converter creates two rungs.
The first rung has a timer for which the output activates a bit (%Mi).
On the second rung, the bit (%Mi) enables a call to an OPERATE block which contains the PULSOR function. The Enable parameter of this function is fixed to the value TRUE.
When the timer output disables the bit, the PULSOR function is no longer executed as the operate bloc is no longer called, but the function output remains in the same state it was in at that time.

To correct this incorrect conversion, set the %Mi bit (activated by the timer) in the Enable parameter of the PULSOR function and permanently enable the operate block.

The following conditions prevent translation:

Logical expressions	• Expressions containing a %CONT, • Expressions containing at least one initialization of the Chart registers: MIN, MAX and NUM (%ACQGRAF),
Use of CFBs and UFBs	• Expressions containing UFBs, Expressions containing non-convertible CFBs (see 2.7.6.2 for the list of convertible CFBs), CFB/CFB linking with indirect binary links.

2.7-5 Setting combinational logic parameters

General rule:

Each %EC is translated as one rung.

The functional transcription of a %EC :
- is either performed as a whole.
- or is an **empty rung**, if transcription fails.

The transcription of a complete %EC can require special parameter setting.

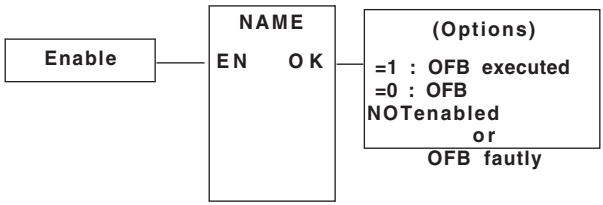
The list of restrictions is given in the section “Limitations and restrictions” (see 2.7.4.).

2.7-6 Setting the parameters of function boxes

Reminder : In the Orphee workshop, the CFBs are ranked in families. The Ladder Converter only takes into account the families associated with the standard Series 1000 language.
The following families will therefore not be taken into account:
- chart management,
- measurements and PID control,
- PID control,
- axis,
- communication,
- safety and availability.

General rules for the conversion of CFBs :

Generally, the execution environment for an Orphee CFB is shown as follows:

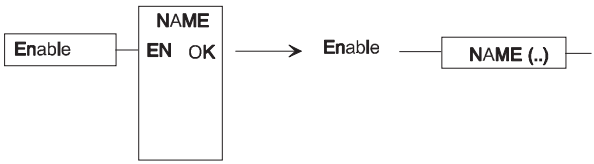


The conversion rules are as follows:

Rule 1 :

IF the OK output of the CFB is not programmed

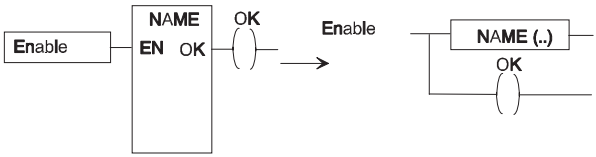
the conversion is performed bijectively.



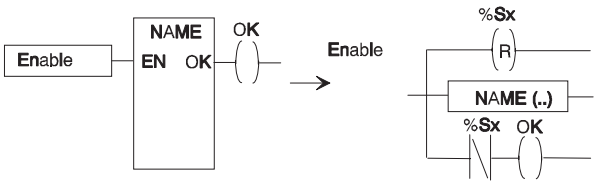
Rule 2:

IF the OK output of the CFB is programmed

1st Case : the operation always succeeds.



2nd Case : the operation can fail.

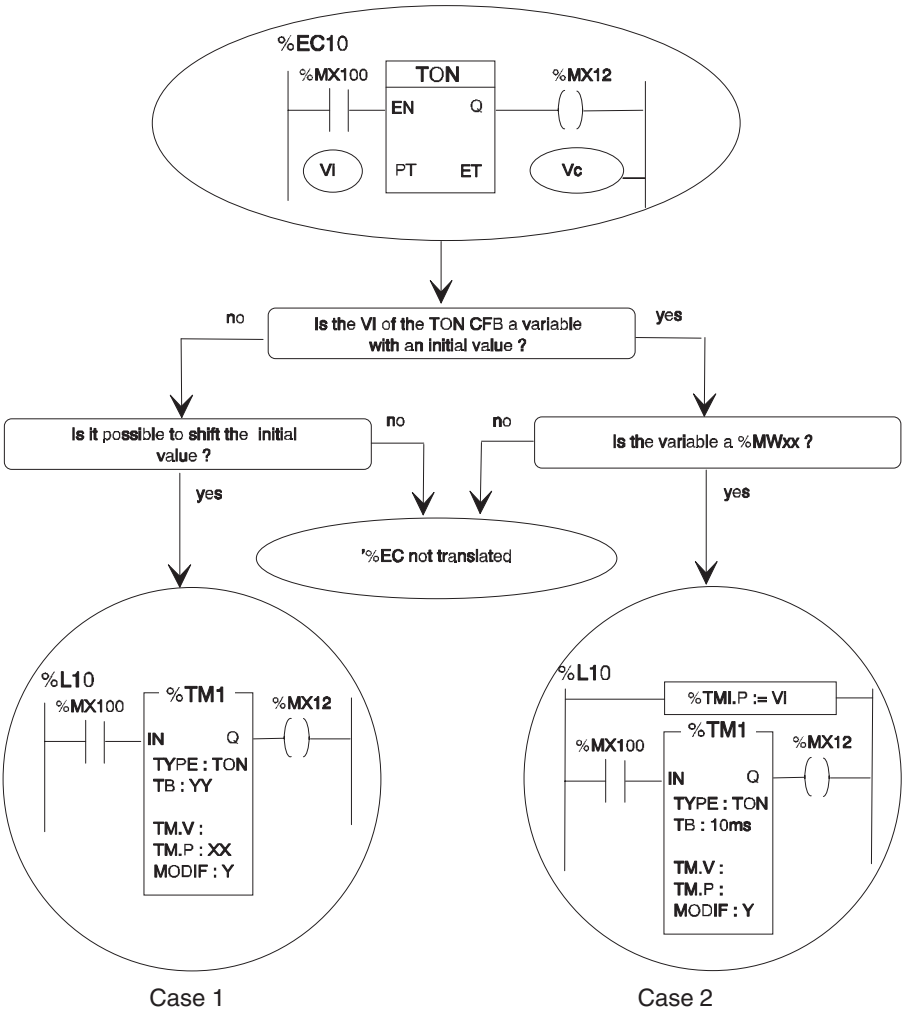


Particular cases:

Timers

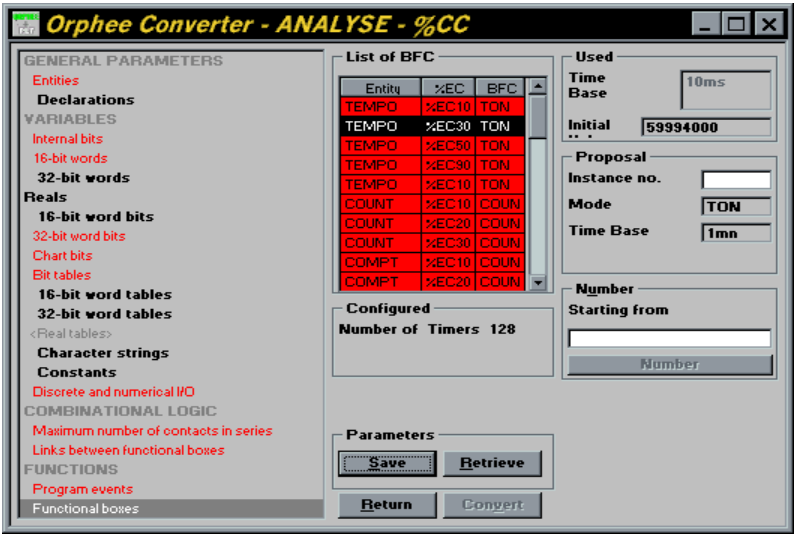
This concerns the parameter settings for the following CFBs: TON.
The equivalent functions in the PL7 target language are performed starting with the %Tmi Function Block which can be configured as TON with an instance number "I" (from 0 to a configured number) to be specified during association.
In the ORPHEE CFBs, the pair giving the time delay limit: initial value/time base should be readjusted according to the "configurable limit" of %TMi : time base (Const) = 1min, 1s, 100ms at 10ms/ initial value (Const) = {0..99991}.
Thus a TON CFB with 45000 as initial value will be transformed to TIMER (TON, 100ms, 4500).

The translation algorithm is described by the following TON CFB example:

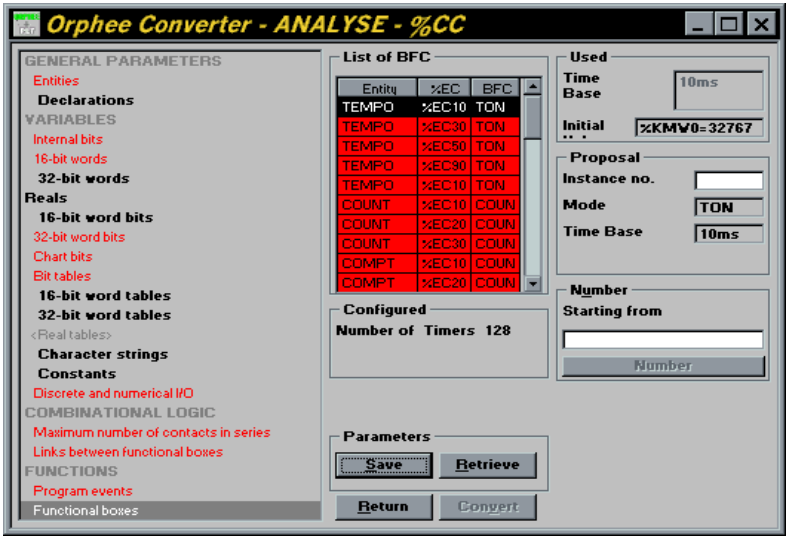


Note:
The Vc variable of the CFB containing the current value is ignored.

Case 1 : No initial value

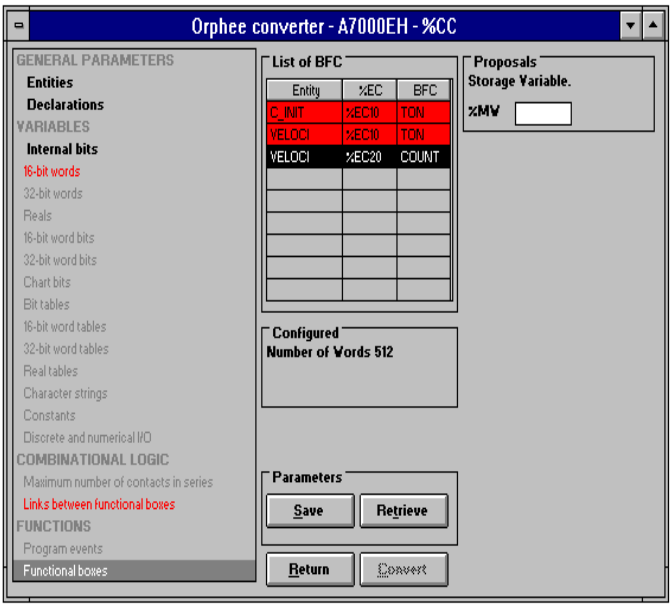


Case 2 : With initial value



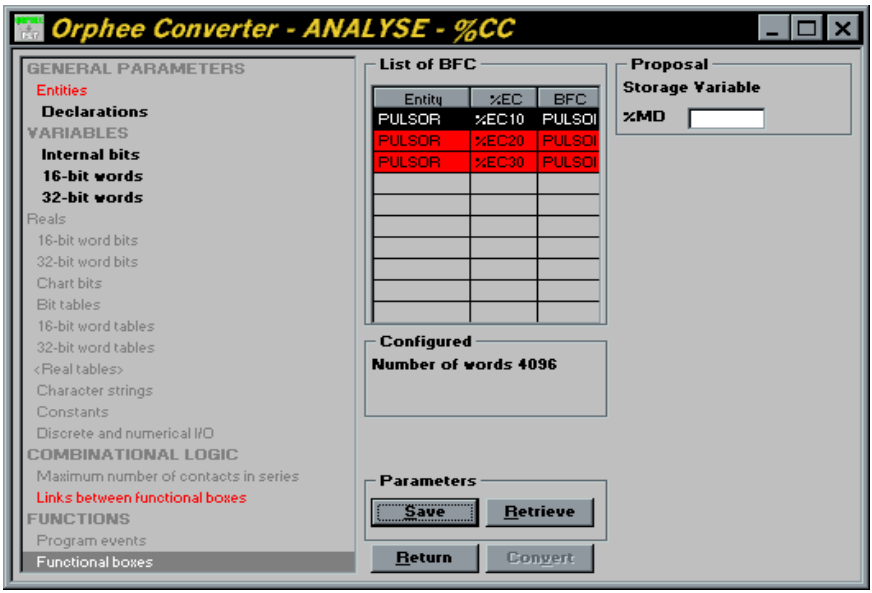
Up/Down counter

The COUNT up/down counter CFB in an equivalent function in the PL7 target language is translated from a "**Series 1000 special OF**", identical to the COUNT CFB.
Latching CU and CD, and the "report" of the <bit expression> type inputs are performed using a numeric variable passed as a parameter to the OF.
This variable is entered during the setting of the parameters of the following "Function boxes":



Pulsor

For the PULSOR CFB, the latching for internal states for the equivalent function in the PL7 target language is performed using a %MD variable.



List of CFBs accepted by the Converter

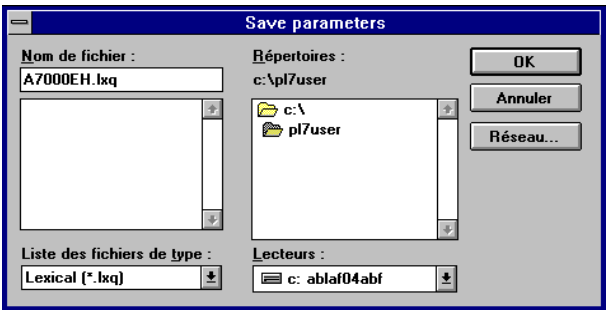
FAMILY	CFB	DESIGNATION
MATH. CALCULATIONS	*ADD*	addition of values
	SUB	subtraction of values
	MUL	multiplication of values
	DIV	division of values with remainder
	SQRT	square root with remainder
LOGIC OPERATIONS	*AND*	logical AND
	NEG	logical negation (complement)
	OR	logical OR
	XOR	exclusive OR
SHIFTS of	*ROL*	rotate left by n bits
	ROR	rotate right by n bits
	SHL	shift left by n bits, replace with 0s and retrieval of shifted bits
	SHR	shift right by n bits with sign extension and retrieval shifted bits
	SHRZ	shift left by n bits, replace with 0s and retrieval of shifted bits
TIMERS/ COUNTER	*COUNT*	up and downcounter with over/underflow of min and max threshold
	PULSOR	Pulse output (Remark on conversion: The AND output decreases with ORPHEE but increases with PL7. The AND output is a 32-bit word with ORPHEE but a 16-bit word with PL7). See warning on page 2/26.
	TON	on delay
COPY/ CONVERSION	*COPY*	copy one variable to another
	ASCII_M	ASCII to integer conversion
	BIN_DCB	decimal to BCD conversion
	DCB-BIN	BCD to binary conversion
	FD_M	floating point to integer conversion
	GRAY_BIN	Gray to binary conversion
	M_FD	integer to floating point conversion
	MD_MW	32-bit integer to 16-bit integer conversion
	MW_MD	16-bit integer to 32-bit integer conversion
	M_ASCII	integer to ASCII conversion

FAMILY	CFB	DESIGNATION
COPY/ CONVERSION (cont'd)	*WD-BIT*	Transfer a word or word table to a bit table (* Word table to bit table with designation of Origin and Destination rows, * Double word table to bit table with designation of Origin and Destination rows)
	BIT _WD	Transfer a bit table to a word or word table *(* Bit table to word table with designation of Origin and Destination rows, * Bit table to double word table with designation of rows)*
TESTS	*COMP*	comparison of 2 values
	COMP4	comparison of 1 value with 4 others
STRING PROCESSING	*CONCAT*	concatenation of 2, 3 or 4 character strings
	LENGTH	calculation of the actual length of a string
	MID	extraction of a substring
	COMPCH	comparison of strings
	DELETE	delete a substring
	INSERT	insert a substring
	REPLACE	replace a substring
TABLE PROCESSING	*FIND_EQ*	Search for the first equal element to a value
	OCCUR	Calculate the occurence of a value

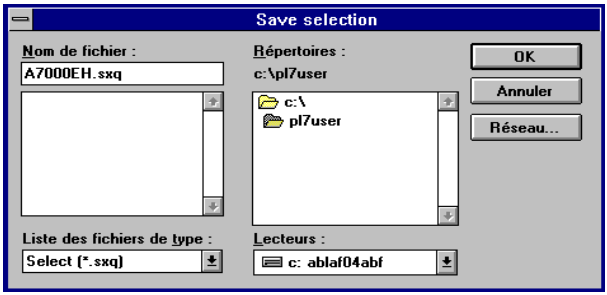
2.8 Memorizing and recalling the parameters

The “Save” function is used to save all the parameters (modified or not) of each heading in one file.

The extensions for these files are:
- *.lxq for the conversion parameters,



- *.sxq for the selection of variables.



The parameter settings are memorized as long as the Converter is active. Between 2 conversion sessions, the parameters are updated according to the principle of the **Recall** function described below.
The **Recall** function is used to associate the parameters of each heading previously saved in a file with the current configuration data.

The following are not saved:
- the CFB associations,
- the type of import,
- the number of the first rung.

The target configuration is not saved, it is up to the user to make a note of it before exiting, if he wishes to reuse it.

N.B.
The parameter files are linked to one conversion. The save files are then protected using the names of the Orphee and PL7 applications. Any attempt to use these files on other applications displays error and warning messages.

Example:
- 1st case: if the *.Ixq file has the correspondence %MX0<=>%M10 and if the current parameter settings give %MX0 <=> %M0 then the “Recall” function performs the final correspondence %MX0<=>%M10.
- 2nd case : if the *.Ixq file has the correspondence %MX20 <=> %M30 and if %MX20 is not used in the current parameter settings, then the “Recall” function has no effect on the current parameter settings.

2.9 Conversion

The “Convert” function is used to launch the actual conversion, if all the essential configuration data has been filled in.
Once the conversion is complete, the report is displayed automatically.

The report is divided into 4 parts: the environmental data, the conflicts to be resolved, the summary of the conversion and the list of variables used. Each part is described in one of the following chapters.

2.9-1 The context block

```
**********************************************************************
                              CONTEXT
**********************************************************************
File generated :ANALYZE.LD
Conversion date : 01/12/1997
From the Orphee ANALYZE application towards the PL7 <no name>
application Orphee converter Code V3.0
```

The file generated, the conversion date, the name of the Orphee (source) and PL7 (destination) applications and the version of the Converter can be found here.

2.9-2 The elements to be configured

```
**********************************************************************
          ELEMENTS TO BE CONFIGURED BEFORE IMPORTING
**********************************************************************
PL7             Symbol     Orphee
%M1025          TOTALRX    %RX0
%M1026          PASPAR     %RX1
          Entity    Expression    PL7 Mode
%TM512      TEMPO   EC10          TON
SR PL7 (must be present when importing)   SR0
```

Three categories:
- the variables exceeding the current capacities of the PLC (Outside Conf)
- the CFB operating words or Timer which are outside Conf,
- the Subroutines (SR) which are used in the code and which must be declared in the task before the import.

2.9-3 Result of the code conversion

```
**********************************************************************
                        RESULT  OF  THE  CONVERSION
**********************************************************************
 Entity      Name         Rung No.

 %CC0        TESTBIT    %L0
 %EC10                  Translated
 %EC20                  Translated  To be completed on import.
 %EC30                  Not translated   CFB COPY :  parameters
                                                ...     refused.
 %EC40                  Not translated    CFB ASCII_F  cannot be
                        translated
 %CC45       CFBTOTAL   %L1          Entity with enabling condition
 %EC50                  Not translated    The indirect links are not
                                          translated.
                                    The  expression  is  aborted.
 %EC60       %L2        Translated
 %EC70                  Not translated    The expression is an OR
with
  11 contacts.  No  translation
 %EC80                  Not  translated     Variable  unknown!
 %EC90                  Not  translated    Not  enough  lines  for  the
                                          expression.  No translation.
 %EC100                 Not  translated     The  following  variable
could
                                          not be identified  PENAMED
  Enabling  Jump  %L3
```

Every entity generates at least one line which recalls its number, name and the (obligatory) associated label.
Every expression (%EC) generates a line which signals the success or failure of the conversion.
For entities, the only possible message is "**Entity with enabling condition**": the entity has an enabling condition which is generated before the first expression.
For expressions (%EC), There are 2 levels of message:

"Translated":	the expression is generated, with perhaps one small limitation: To be completed on import": one of the variables has no translation, the code needs to be completed during import
"Not translated":	the expression is not generated, all that will be found in the code will be a comment and the reason for the failure.

The reasons for the "non translation" are:

"CFB xxx cannot be translated"	the CFB in question has no equivalent in PL7
"CFB xxx: parameters refused"	the CFB exists in PL7, but does not accept these types of parameter
"The indirect links are not translated"	a CFB sequence contains a complex binary link, which cannot be converted
"The expression is an OR with 11 contacts"	the expression is made up of 2 lines with 11 contacts, the translation of this expression creates a rung of more than 10 contacts in series; the OR cannot be translated
"Not enough lines for the expression"	the equivalent expression occupies more than 7 lines in PL7
"Variable unknown! "	the expression contains an incomplete parameter, which cannot be translated
"The next variable could not be identified"	in general, the code contains a call to a symbolized Program Event, and the Converter cannot establish the link between the Symbol and the PE number

An additional line may be found. Example: "**Enabling Jump %L5**". This indicates that the last entity generated contained an enabling condition, which requires a jump label. An additional rung (%L5) has thus been generated to obtain this label.

2.9-4 Summary of data

```
=========================================================================
                          VARIABLES  AND  CFB

  Symbol     Orphee            PL7                          Fault
  TOTALMX    %MX0              %M0
  PASCTMX    %MX1              %M1
  TOTALMW    %MW0              %MW0                    (Timer inhibited)
  PASCMW     %MW1              %MW1                    (Timer inhibited)
  BITMD      %MD4002:X16             ORPHEE_MD4002X16       To be filled
in

  LONGTRX    %RX330(0..50)           %M1355:51             Symbol ignored
  TOTALCH    %CH8000(0..9)           %MB1846:10          Init.Val. ignored

  <CFB  Instance>
  Entity     Expression  Mode     PL7
  TEMPO           EC10        TON   %TM512
  COUNT           EC30        COUNT %MW78
```

This part gives all the associations used to convert the code. The equivalencies between Orphee and PL7 variables are therefore found here, together with timer assignments and CFB operating words.

The possible faults are:

"To be completed"	the variable is not associated with a PL7 address. The code should be completed during the import. In this case, the symbol generated by the tool includes the ORPHEE_ prefix followed by the simplified address. (If the rung in which the variable appears is translated).
"Conflict of symbol"	the address has another symbol in the application or in the translated variables.
"Symbol ignored"	the symbol is conflicting with another declaration. It is not imported or will generate a message.
"PL7 word reserved"	the symbol is a PL7 reserved word.
"Init V ignored"	for strings, remember there are no initial values in PL7.
"(Timer inhibited)"	the word or double word was used as a current value for a Timer CFB. Since the conversion does not generate the code used to retrieve this value, the word is no longer linked to the CFB.

2.10 Correspondence tables

2.10-1 Table of correspondence between variables

ORPHEE data	PL7 data	Conversion	Symbol
%MXn	**%Mn or %MXn (1)**	AUTOMATIC	RETRIEVABLE
%RXn	**%Mn or %MXn**	AUTOMATIC	RETRIEVABLE
%MWn	**%MWn**	AUTOMATIC	RETRIEVABLE
%MWn:Xi	**%MWn:Xi**	AUTOMATIC	RETRIEVABLE
%MDn	**%MDn**	AUTOMATIC	RETRIEVABLE
%MDn:Xi	**%MWin:Xi** **%MXn**	ASSISTED	RETRIEVABLE
%FDn	**%MFn**	AUTOMATIC	RETRIEVABLE
%MXn(0..k)	**%Mn:L**	AUTOMATIC	RETRIEVABLE
%MXn(%MWj) **%MXn(i)**	**%Mn(%MWj]** **%M(n+i)**	AUTOMATIC	RETRIEVABLE
%RXn(0..k)	**%Mn:L**	AUTOMATIC	RETRIEVABLE
%RXn(%MWj) **%RXn(i)**	**%Mn[%MWj]** **%M(n+i)**	AUTOMATIC	RETRIEVABLE
%MWn(0..k)	**%MWn:L**	AUTOMATIC	RETRIEVABLE
%MWn(%MWj) **%MWn(i)**	**%MWn[%MWj]** **%MWn(i)**	AUTOMATIC	RETRIEVABLE

(1) - WARNING:
Monostable bits have no equivalent in PL7.
ALL BITS ARE MEMORIZED.

1	2	3	4 - Conversion

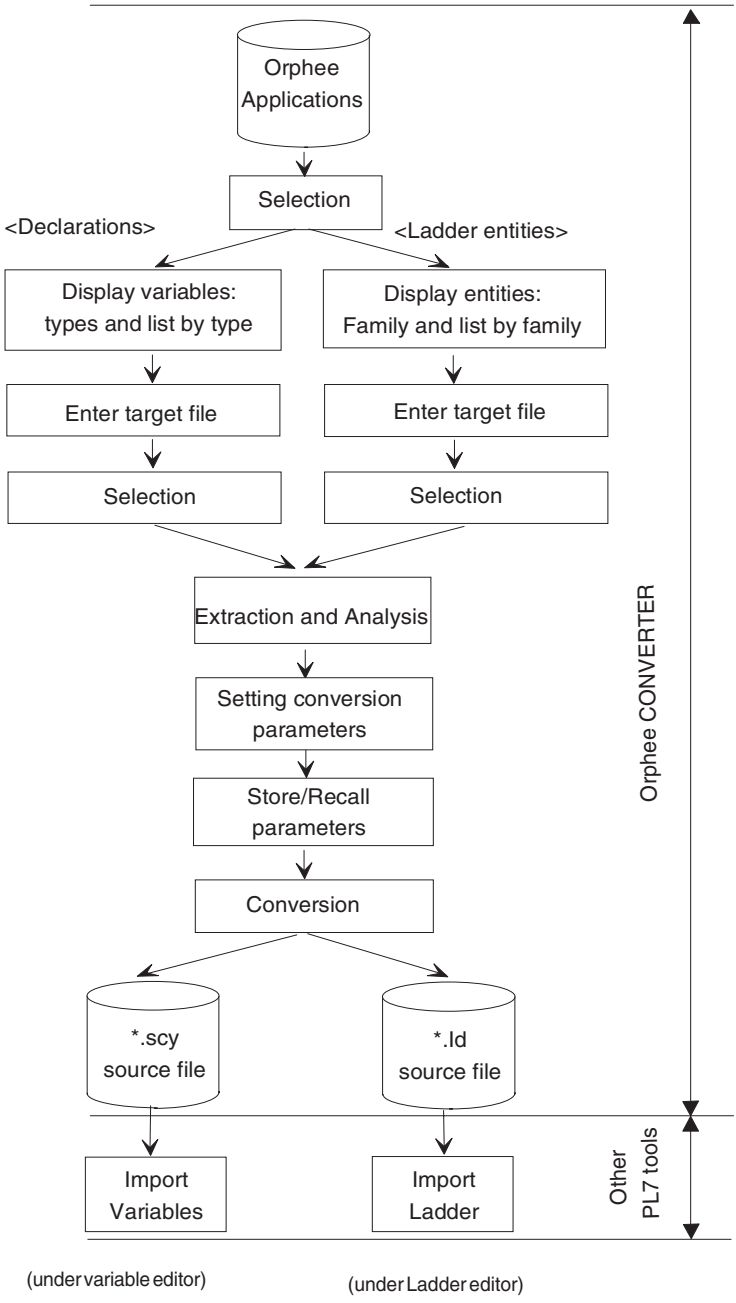
ORPHEE data	PL7 data	Conversion	Symbol
%MDn(0..k)	**%MDn:L**	AUTOMATIC	RETRIEVABLE
%MDn(%MWj) **%MDn(i)**	**%MDn[%MWj]** **%MDn(i)**	AUTOMATIC	RETRIEVABLE
%FDn(0..k)	**%MFn:L**	NOT TRANSLATED	NOT RETRIEVABLE
%FDn(%MWj) **%FDn(i)**	**%MFn[%MWj]** **%MFn(i)**	NOT RETRIEVABLE	NOT RETRIEVABLE
%CHn(0..k)	**%MBn:L**	AUTOMATIC	RETRIEVABLE
%KMWn	**%KWn**	AUTOMATIC	RETRIEVABLE
%KMDn	**%KDn**	AUTOMATIC	RETRIEVABLE
%KHWn	**no**	AUTOMATIC	RETRIEVABLE
%KHDn	**no**	AUTOMATIC	RETRIEVABLE
%KFDn	**%KFn**	AUTOMATIC	RETRIEVABLE
%KCHn(0..k)	**%KBn:L**	AUTOMATIC	RETRIEVABLE
%IXn	**%In.MOD.channel**	ASSISTED	RETRIEVABLE
%QXn	**%Qn.MOD.channel**	ASSISTED	RETRIEVABLE
%IWn	**%IWn.MOD.channel**	ASSISTED	RETRIEVABLE
%IDn	**%IDn.MOD.channel**	ASSISTED	RETRIEVABLE
%QWn	**%QWn.MOD.channel**	ASSISTED	RETRIEVABLE
%QDn	**%QDn.MOD.channel**	ASSISTED	RETRIEVABLE

ORPHEE data	**PL7 data**	Conversion	Symbol
%GMm:Xn		ASSISTED	NOT RETRIEVABLE
%GEm:Xn		ASSISTED	NOT RETRIEVABLE
%XEm:Xn		ASSISTED	NOT RETRIEVABLE

2.10-2 Table of correspondence of combinational logic which can be translated

ORPHEE data		PL7 data	
Open contact		Open contact	
Closed contact		Closed contact	
Rising edge contact		Rising edge contact	
Falling edge contact		Falling edge contact	
Open coil		Open coil	
Closed coil		Closed coil	
SET coil		SET coil	
RESET coil		RESET coil	
Jump to a %ECn:	 **%ECn**	Jump to a RUNG	 **%Li**
Prog. stop	 **%STOP**	Halt coil:	
Prog. Init	 **%INIT**	By setting bit %S0 to 1	
Start event masking	 **%MSKEVT**	**MSKEVT** instruction in operate block	
End event masking	 **%DMSKEVT**	**UNMASKEVT** instruction	
Call program element	 %EPn	SR parameters not set	

3.1 Overview


Orphee Applications
Selection
<Declarations>
<Ladder entities>
Display variables: types and list by type
Display entities: Family and list by family
Enter target file
Enter target file
Selection
Selection
Extraction and Analysis
Setting conversion parameters
Store/Recall parameters
Conversion
*.scy source file
*.ld source file
Import Variables
Import Ladder
(under variable editor)
(under Ladder editor)
Orphee CONVERTER
Other PL7 tools

3.2 Preliminary preparation

First of all, the applications to be retrieved should be placed in the SERVROOT of the PL7 Junior execution directory, or in a remote SERVROOT. Refer to section 2.2 for application retrieval.
It is then advisable to have the Orphee documentation file for the application (data, cross references and program).

3.3 Retrieval of variables only

The user wishes to retrieve the Orphee application declarations, ie. the symbols and comments. The initial values are not retrieved, except for the constants.

3.3-1 On a new PL7 application

The application variables space contains only the imported variables. Thus, there are no problems with the existing PL7 when the associations are entered, or when the import takes place. Only the conflicts between the imported variables and the configuration errors remain.

Example of use

1. From the Orphee application documentation file, prepare the **list of I/O** and the declarations required.
2. Open PL7 and create an application. Configure the **I/O modules** according to the list, and configure the number of bits, words, timers and constants required.
   This will avoid the need to move between the configuration editor and the converter too frequently.

   N.B. This operation can only be performed before the import, using the conversion report.
3. After the configurations have been confirmed, save the application.
   This then gives a stable base for saving the parameters.
4. Launch the converter, select and open the Orphee application.
5. Select the **Declaration** heading, the list of types of variables is created.
6. Select a type of variable then the variables within the type, and repeat for other types.
7. Request **Analyzer.**

8. Check or enter the associations and resolve the conflicts which appear in red.

9. Request to “Convert” the declarations.

10. Read the conversion report carefully.

11. Open the variables editor, go to the **File** menu, then select **Import.**

12. Save the PL7 application obtained, close the converter.

3.3-2 On an existing PL7 application

The main problem is that of managing the conflicts between the added variables and the existing variables. The procedure is the same, apart from the fact that it may be necessary to enter a target configuration. You must also monitor carefully any conflicts of symbols so as not to lose any variables. Finally, the action of importing into the variables editor must be made in **Dialog** mode to resolve any conflicts.
It is advisable to have the PL7 documentation file for the target application before starting.

3.4 Retrieval of control system and combinational logic functions (entities + variables)

The aim is to retrieve the Orphee application functions in a new PL7 application. Since there is no point in importing the code on its own, the generated file contains the declarations required for this code. Conflicts can arise from incorrect target configuration parameter settings or from incorrect variable associations. The only conflict generated by the code itself is incorrect rung numbering.

3.4-1 On a new PL7 application

For the new PL7 application, the target configuration is based only on the requirements of the translated functions.
It is advisable to use the rung numbers in ascending order, using the previous conversion report.

Example of use

1. Using the Orphee application documentation file, list the variables required for the conversion and note especially the maximum indexes for the bits (memorized, and not memorized), words, double words and reals. Also list the I/O requirements.
2. Open PL7, create an application. Using the list, configure the I/O modules and PLC memory. Confirm and save the application.
3. Launch the converter. Enter the target configuration.
4. Select and open the Orphee application,
5. Select the types of entity then the entities to be converted. Selection need not be adjacent.
6. WARNING: The selection will be converted to a single result file, which will be imported to a single program module. Example: several PEs can only be imported into one SR.
7. Request to **Analyze** the selection, the code and its data are extracted.
8. Check or enter the conversion parameters.
9. Request to **Convert** the code. Read the conversion report carefully.
10. Open the Ladder editor in a section, SR,..... Go to the menu file, Import option.
11. Save the PL7 application, close the converter.

3.4-2 On an existing PL7 application

- the determination of the target configuration: the Orphee and PL7 application requirements must be taken into account.
- the selection of the number of the first rung generated must not cause conflicts with what already exists.

You must pay attention to any conflicts of symbols for all types of variable, and select a conversion in **NON PRIORITY MODE.**

3.5 Maximum retrieval from an application

We recommend that the conversion actions are performed first for the declarations then for the code. It is advisable to import the variables file before converting the code.