Post-processor Manual

Release #4

www.irai.com

General Information

Post-processors are software modules used to translate pivot code files generated by the AUTOMGEN compiler into executable files on a target as well as ensuring dynamic connection of the target.

The word « Target» is a generic reference for a programmable system capable of executing an application.

An AUTOMGEN post-processor is used to program a type or type set of targets (generally a family of processors sharing the same language can be programmed with the same post-processor in AUTOMGEN).

The first part of this manual contains basic information that is common to all post-processors. Specific information regarding applications for each post-processor is described further on.

Configuration

Please carefully read the explanations in this chapter.

Configuration files

Four configuration elements are used for each post-processor. Each is used for a specific purpose.

System

This contains the hardware configuration of the target, software configuration, options for modifying the way the post-processor generates the code as well as reserved variable statements (for using inside the post-processor). Generally, depending on the target, you will modify the hardware configuration contained in that element (for example the type of UC or a configuration of the type of input/output cards)

Variable functions

Mastering variable functions is one of the fundamental elements for mastering the use of post-processors.

When the post-processor translates a AUTOMGEN pivot language file into a specific target language, it must attribute AUTOMGEN variables to the target variables.

This element contains the exact description of variable attribution. By modifying this element you have total control over the use of the target variable space.

Start-up manufacturer code

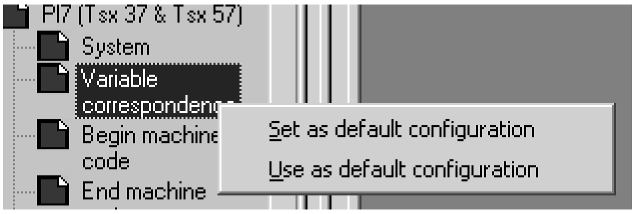
This element contains the machine language for the target which will be placed at the beginning of the executable code generated by the post-processor (executed at the beginning of the cycle).

End manufacturer code

This element contains the machine language for the target which will be placed at the end of the executable code generated by the post-processor (executed at the end of the cycle).

Configuration by default

When a project is created, the default configuration elements are duplicated in the project. Modifications to the project configuration elements will not affect the default statements.



Modifying the default statements

Use the right side of the mouse to click on the element « Configuration / Post-processor / <target name> / … » and select « Set as default configuration». The project configuration will then be set by default (canceling of the default configuration).

Be careful, this operation is irreversible. The configuration element can only be restored if the post-processor is reinstalled.

Using the default statements.

Use the right side of the mouse to click on the element « Configuration / Post-processor / <target name> / … » and select « Using default configuration». The project configuration in progress is cancelled by the default configuration.

Displaying and modifying configuration elements

Access the configuration files by double clicking on the « Configuration / Post-processors / <target name> / … » element. A window will open that can be used to display and modify the configuration element.

System

This configuration element is specific for each post-processor

Hardware configuration

This area (optional) is modified to establish the hardware configuration of a target (for example, type of CPU, input/output cards)

Software configuration

This area (optional) is modified to establish the characteristics of an application configuration (for example, the value of a watchdog).

Code generation options

This area contains the settings for the translation method that the post-processor must use (only for specialists). The number of options may differ from one post-processor to another. The list below shows the options which all post-processors have in common:

« Optimize generated code »

Generally set on « Yes».. A « No » setting is used for an easier analysis of generated code.

« Do not generate the code of Grafcet steps »

Set to « No ». by default. If set to « Yes», in the « End machine code » you must write the instructions used to recopy the immediate states of boolean variables transferred to the past states (see chapterManaging AUTOMGEN boolean variables)

« Do not generate the evolution code of user bits »

Identical to the previous option but applied to AUTOMGEN user bits (« U » variables).

Stating the variables

These are statements of variables used internally by the post-processor. Only specialists can modify these variables.

Other elements

There may be other specific elements for each post-processor.

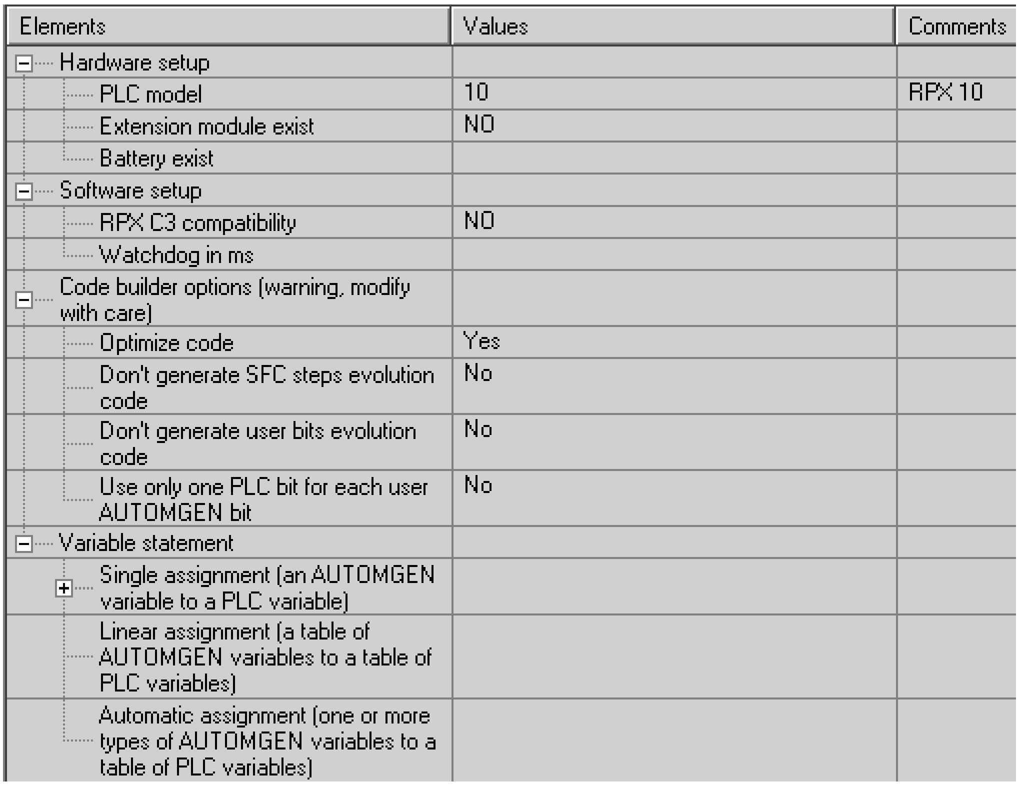
See the « System » element in text format.

By clicking on the ▤ icon in the toolbar, you go from tree mode to « text » mode (format of old AUTOMGEN versions). In « Text » format you can copy and paste information into the configuration files.

Modification to « text » mode must be made by specialists, inopportune modifications can lead to compiling errors which are difficult for an inexperienced person to find.

Displaying system elements

By double clicking on « Configuration / Post-processors / <name of post-processor> / System » the following window opens.



Elements	Values	Comments
⊟ Hardware setup		
PLC model	10	RPX 10
Extension module exist	NO	
Battery exist		
⊟ Software setup		
RPX C3 compatibility	NO	
Watchdog in ms		
⊟ Code builder options (warning, modify with care)		
Optimize code	Yes	
Don't generate SFC steps evolution code	No	
Don't generate user bits evolution code	No	
Use only one PLC bit for each user AUTOMGEN bit	No	
⊟ Variable statement		
⊞ Single assignment (an AUTOMGEN variable to a PLC variable)		
Linear assignment (a table of AUTOMGEN variables to a table of PLC variables)		
Automatic assignment (one or more types of AUTOMGEN variables to a table of PLC variables)		

Example of system configuration

Variable functions

Mastering variable functions is one of the fundamental elements for mastering the use of post-processors.

When the post-processor translates a AUTOMGEN pivot language file into a specific target language, it must attribute AUTOMGEN variables to the target variables.

For example, if you use the AUTOMGEN word 200 in your application (called M200 or %MW200 in AUTOMGEN) this word must exist in the target memory and thus must have a name for that target.

AUTOMGEN proposes three types of statements for variable functions.

- single assignment;
- linear assignment
- automatic assignment

Variable functions for a project will be composed of « n » assignments each one using one of the three types.

Single assignment

This is used to associate an AUTOMGEN variable to a target variable. It is the simplest of the statements

It can only be used if a single statement is necessary.

This statement uses two pieces of information. the name of the AUTOMGEN variable and the name of the target variable.

« Associate this AUTOMGEN variable to that target variable », to effect a single assignment..

Linear assignment

This form is more evolved than a single assignment.

It is used to associate a series of consecutive AUTOMGEN variables (multiple variables of the same type where the numbers are in order) to a series of consecutive target variables.

The assignment is normally used for:

- stating input/output variables;
- stating bit or word tables which must have a fixed address (for example, for a link with an operator control panel).

This statement uses three pieces of information: the name of the first AUTOMGEN variable, the name of the first target variable and the dimension of the table in number of variables.

« Associate this AUTOMGEN variable table in order to that target variable table », to effect a linear assignment.

Automatic assignment

This is the most complex and powerful type of statement. It is used to associate one or more types of AUTOMGEN variables to a range of target variables.

This assignment gives the compiler the task of finding an assignment for each variable in the generated code (as long as it corresponds to one of the types) of the statement.

This type of statement is normally used for all AUTOMGEN application variables where the address of the variable associated in the target does not need a precisely fixed address.

This statement uses three pieces of information:

- the type of AUTOMGEN variables (see the chapter Types of AUTOMGEN elimination variables)
- the name of the first variable of the target range;
- the number of the last variable (included) of the target range.

Automatic assignment is not used for a post-processor if another statement has not been found for a variable. For example, if a linear assignment command sets an attribution for the words 200 to AUTOMGEN 210, the post-processor will not use automatic assignment to try and allocate these words.

If multiple automatic assignments exist for the same type of AUTOMGEN variable, the post-processor will use the first range of target variables until saturation, then the second until saturation, then the third etc.

If a variable has not been allocated at the end of using all the automatic assignments, the compiler generates an error message indicating that the variable has not been set.

« When you find one of these types of variables, use a variable of the target in that area», to effect an automatic assignment.

Types of AUTOMGEN elimination variables

These are used to state variable functions, they are a subset (because more than one target variable may be needed to allocate an AUTOMGEN variable) of AUTOMGEN variable types.

Managing AUTOMGEN boolean variables

One of the basic principles of boolean language translation for the AUTOMGEN compiler is to be able to access two states for the same boolean variable.

This concept refers to the idea of « execution cycle » : an entity representing the action created by the target, consisting in reading the application instructions in a linear mode (from the beginning to the end) and accomplishing the processing they correspond to.

These two states are set as follows:

1- The immediate state of the variable: the state written for the last instruction executed by the target transfers that variable either by default to the state of the variable at the end of the last execution cycle, or if it is the first execution cycle by default to the initialization state of the variable.

2- The past state of the variable: the state of the variable at the end of the last execution cycle.

Comments: the two states are only valid for the main application task. Only the immediate state has meaning for asynchronous tasks.

The code generated by the AUTOMGEN compiler assumes the following:

- assignment of a boolean variable is effected on its immediate state;
- testing of a boolean variable is effected on its past state.

These two rules are used to ensure a consistent evolution of boolean applications and to observe the evolution rules of programs generated by a Grafcet language description.

The code generated by the post-processor manages recopying of the variable immediate states to variable past states at cycle end.

When a boolean variable is used in AUTOMGEN two boolean variables are used on the target.
There are three exceptions:

1- for an all or none input, if no edge test is used, only the past state (« bi ») is used (economy of a boolean variable).
2- for an all or none output, if no edge test is used, only the immediate state (« o ») is used.

(this explains why only the « bi » and « o » variables are found in variable attribution commands).

3- for the ZELIO post-processor, which effects time management of the variables (almost identical to AUTOMGEN's) only immediate states are used in ZELIO programming language.

Standard element syntax

« <AUTOMGEN variable name> » refers to the immediate state of a boolean variable or a numeric variable..

« b<AUTOMGEN variable name> » refers to the past state of a boolean variable.

Special edge syntaxes

« u<AUTOMGEN variable name> » refers to the « rising edge » state of a boolean variable.

« d<AUTOMGEN variable name> » refers to the « falling edge » state of a boolean variable.

Special time delay syntaxes

« time <number> » refers to a time delay number.

« tproc<number> » refers to a time delay procedure..

« tcount<number> » refers to a time delay time counter.

Other special syntaxes

(only for specialists)

« ac » refers to the 16 bit accumulator.

« al » refers to the 32 bit accumulator.

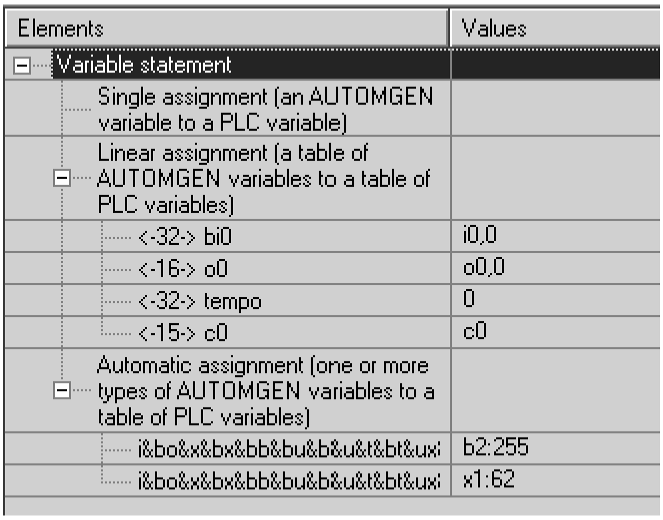
« af » refers to the float accumulator.

« cf » refers to the carry flag.

« zf » refers to the zero result flag

« sf » refers to the negative result flag.

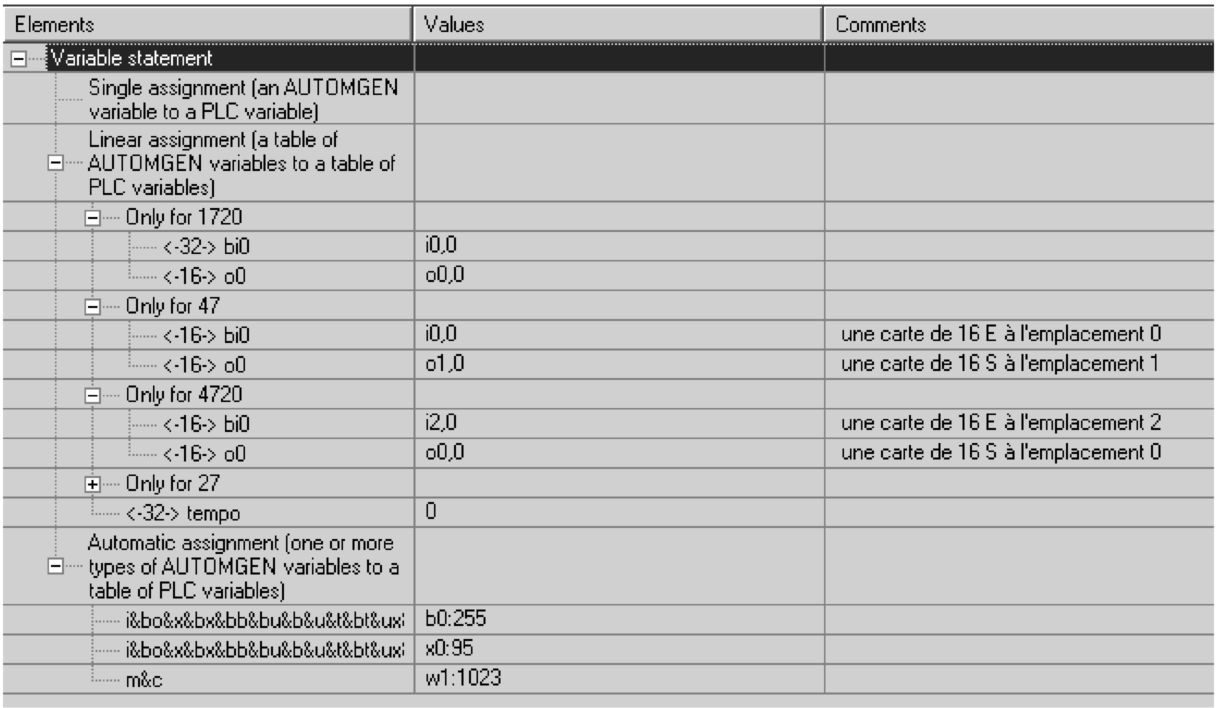
« of » refers to the overflow flag.
Displaying the variable function elements
By double clicking on « Configuration / Post-processors / <name of post-processor> / Variable function » the following window opens.



Elements	Values
Variable statement	
Single assignment (an AUTOMGEN variable to a PLC variable)	
Linear assignment (a table of AUTOMGEN variables to a table of PLC variables)	
<-32-> bi0	i0,0
<-16-> o0	o0,0
<-32-> tempo	0
<-15-> c0	c0
Automatic assignment (one or more types of AUTOMGEN variables to a table of PLC variables)	
i&bo&x&bx&bb&bu&b&u&t&bt&uxi	b2:255
i&bo&x&bx&bb&bu&b&u&t&bt&uxi	x1:62

Example of variable functions

Comment: in the event that the same post-processor can generate code for multiple types of target (for example multiple types of processor CPU's) the different elements can be conditioned for all of the target types or for one particular target type. If the elements are conditioned, they are associated to « Only for xxx » lines. See the examples below.

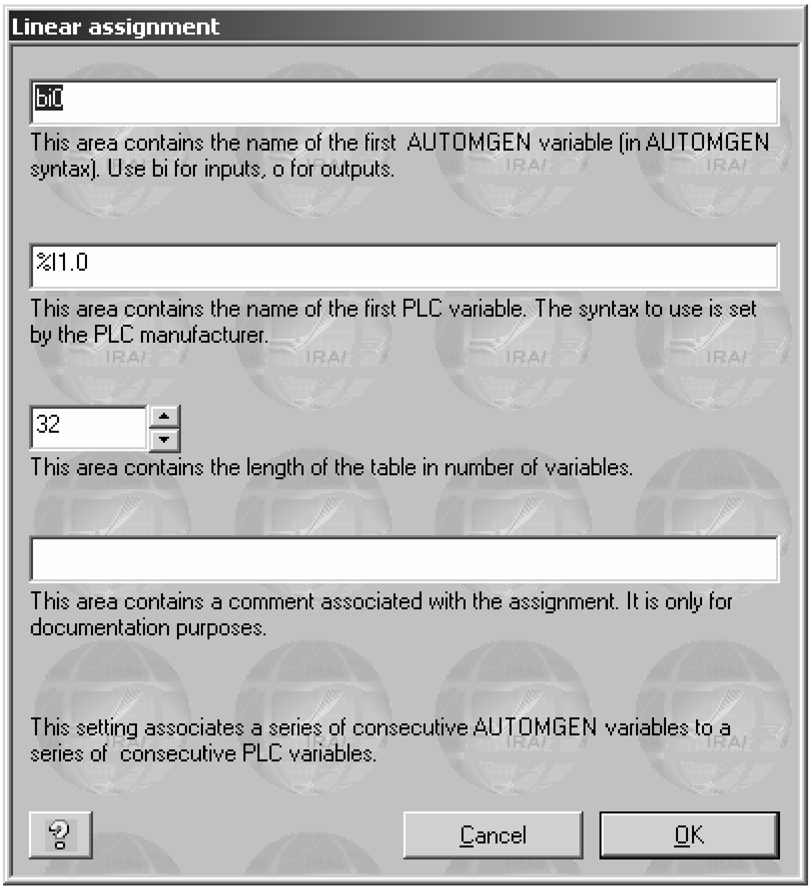


Elements	Values	Comments
⊟ Variable statement		
Single assignment (an AUTOMGEN variable to a PLC variable)		
⊟ Linear assignment (a table of AUTOMGEN variables to a table of PLC variables)		
⊟ Only for 1720		
<-32-> bi0	i0,0	
<-16-> o0	o0,0	
⊟ Only for 47		
<-16-> bi0	i0,0	une carte de 16 E à l'emplacement 0
<-16-> o0	o1,0	une carte de 16 S à l'emplacement 1
⊟ Only for 4720		
<-16-> bi0	i2,0	une carte de 16 E à l'emplacement 2
<-16-> o0	o0,0	une carte de 16 S à l'emplacement 0
⊞ Only for 27		
<-32-> tempo	0	
⊟ Automatic assignment (one or more types of AUTOMGEN variables to a table of PLC variables)		
i&bo&x&bx&bb&bu&b&u&t&bt&uxi	b0:255	
i&bo&x&bx&bb&bu&b&u&t&bt&uxi	x0:95	
m&c	w1:1023	

By clicking on the elements « + » on the tree you open the branches, « - » closes them.

Modifying a variable function element

You can modify the elements by double clicking on them.



Linear assignment

bi0

This area contains the name of the first AUTOMGEN variable (in AUTOMGEN syntax). Use bi for inputs, o for outputs.

%I1.0

This area contains the name of the first PLC variable. The syntax to use is set by the PLC manufacturer.

32

This area contains the length of the table in number of variables.

This area contains a comment associated with the assignment. It is only for documentation purposes.

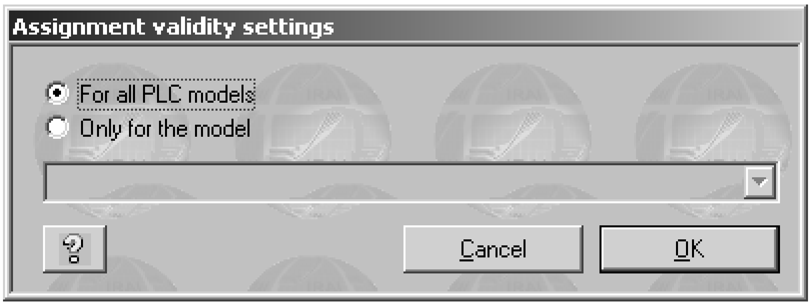
This setting associates a series of consecutive AUTOMGEN variables to a series of consecutive PLC variables.

Example of a configuration dialogue box for a linear assignment.

Adding a variable function element

To add a new assignment, click with the right side of the mouse on the « Assignment … » elements of the tree and select « Add » from the menu.

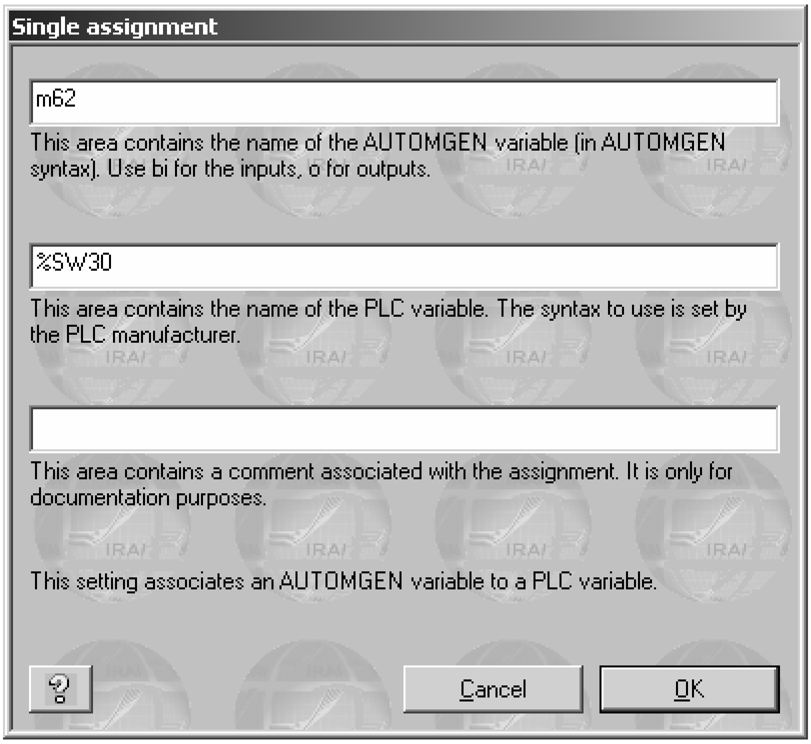
If multiple target types are managed by the post-processor, the following dialogue box is used to establish if the new assignment is only for one type in particular or all the types.



Assignment validity settings

● For all PLC models

○ Only for the model



Single assignment

m62

This area contains the name of the AUTOMGEN variable (in AUTOMGEN syntax). Use bi for the inputs, o for outputs.

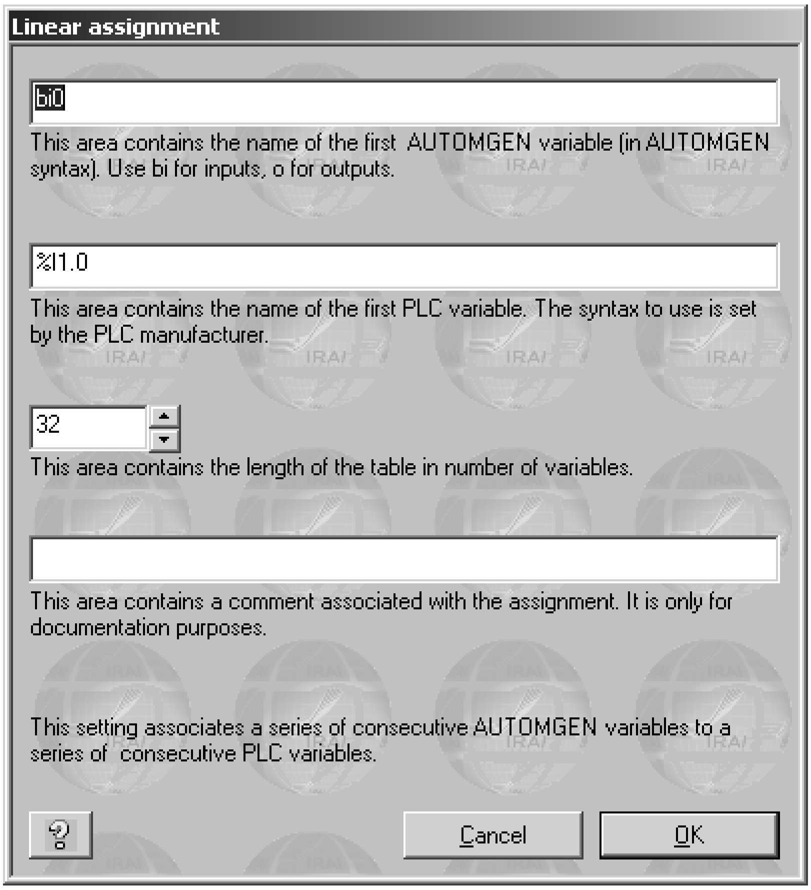
%SW30

This area contains the name of the PLC variable. The syntax to use is set by the PLC manufacturer.

This area contains a comment associated with the assignment. It is only for documentation purposes.

This setting associates an AUTOMGEN variable to a PLC variable.

Single assignment



Linear assignment

bi0

This area contains the name of the first AUTOMGEN variable (in AUTOMGEN syntax). Use bi for inputs, o for outputs.

%I1.0

This area contains the name of the first PLC variable. The syntax to use is set by the PLC manufacturer.

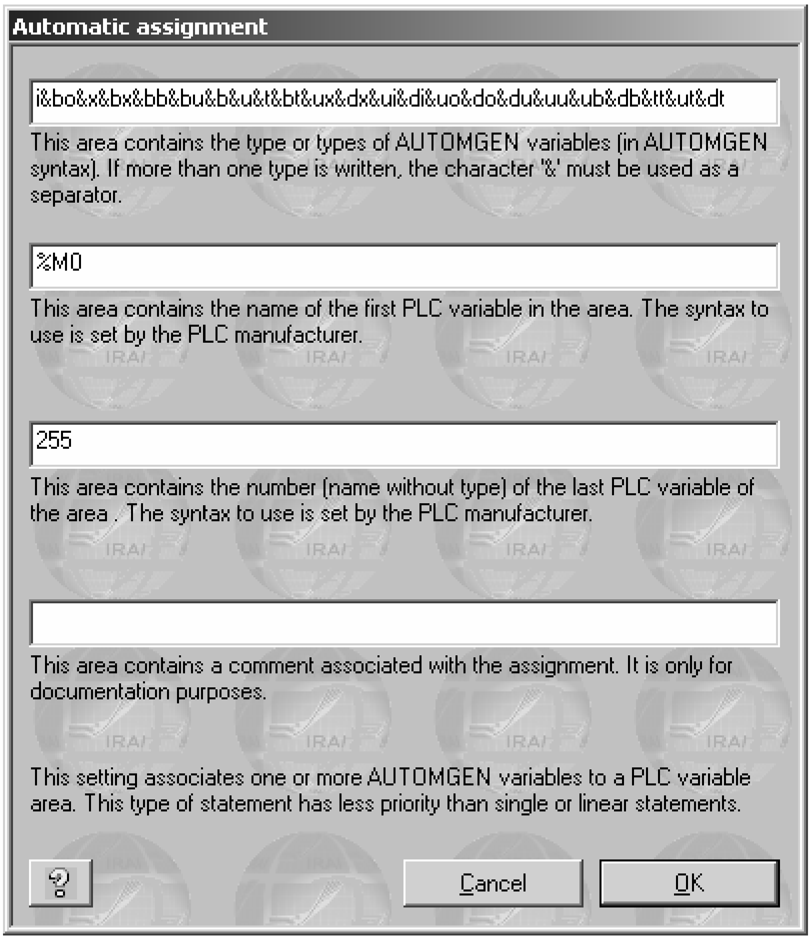
32

This area contains the length of the table in number of variables.

This area contains a comment associated with the assignment. It is only for documentation purposes.

This setting associates a series of consecutive AUTOMGEN variables to a series of consecutive PLC variables.

Linear assignment



Automatic assignment

i&bo&x&bx&bb&bu&b&u&t&bt&ux&dx&ui&di&uo&do&du&uu&ub&db&tt&ut&dt

This area contains the type or types of AUTOMGEN variables (in AUTOMGEN syntax). If more than one type is written, the character '&' must be used as a separator.

%M0

This area contains the name of the first PLC variable in the area. The syntax to use is set by the PLC manufacturer.

255

This area contains the number (name without type) of the last PLC variable of the area . The syntax to use is set by the PLC manufacturer.

This area contains a comment associated with the assignment. It is only for documentation purposes.

This setting associates one or more AUTOMGEN variables to a PLC variable area. This type of statement has less priority than single or linear statements.

Automatic assignment

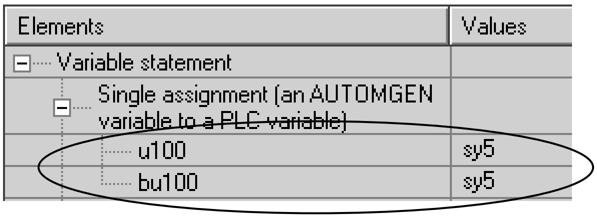
Note that AUTOMGEN variables must always be separated using the « & » character.

Deleting a variable function element

With the right side of the mouse, click on the variable function element and select « Delete » from the menu.

Associating an AUTOMGEN bit to a target system bit

Two statements are necessary, the two state variables of a bit « U » (« u » and « bu ») must be associated to a target system bit. You must create two single assignments, for example:



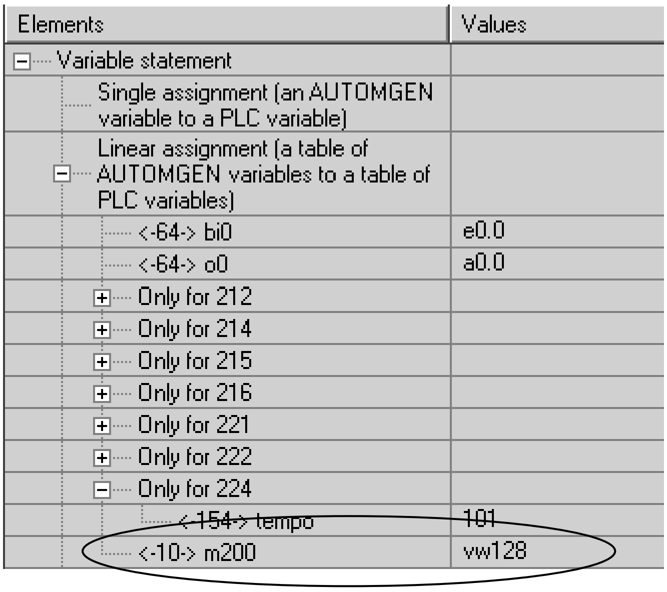
Elements	Values
Variable statement	
Single assignment (an AUTOMGEN variable to a PLC variable)	
u100	sy5
bu100	sy5

Be careful, when you assign the AUTOMGEN u and bu variable to the same target system bit, you eliminate the possibility of creating a rising or falling edge test in the application. You can get around this problem by using the syntax « ↑(u<n>) » or

« ↓(u<n>) » (where « <n> » represents the bit number) in the application (this syntax generates an intermediate bit where the edge will be correctly evaluated).

Associating a table of AUTOMGEN words to a table of fixed target words

Only a single linear statement is necessary for this, for example:

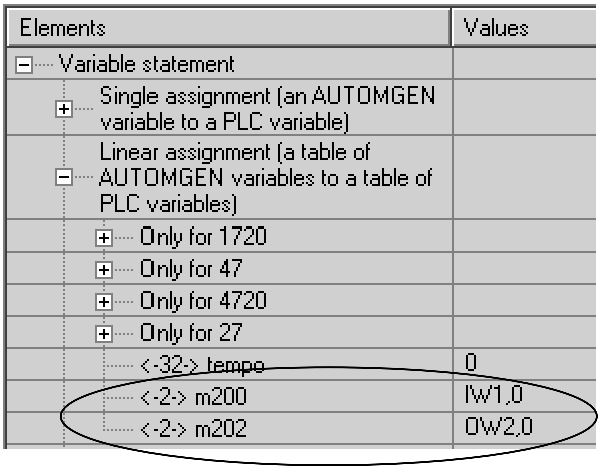


Elements	Values
⊟ Variable statement	
Single assignment (an AUTOMGEN variable to a PLC variable)	
⊟ Linear assignment (a table of AUTOMGEN variables to a table of PLC variables)	
<-64-> bi0	e0.0
<-64-> o0	a0.0
⊞ Only for 212	
⊞ Only for 214	
⊞ Only for 215	
⊞ Only for 216	
⊞ Only for 221	
⊞ Only for 222	
⊟ Only for 224	
<-154-> tempo	101
<-10-> m200	vw128

The target words that are allocated must be free from other assignments or the same target variables may be assigned twice to different AUTOMGEN variables.

Associating AUTOMGEN words to target analog inputs or outputs

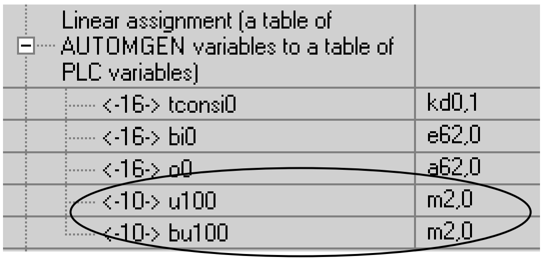
Use linear statements, for example:



Elements	Values
⊟ Variable statement	
⊞ Single assignment (an AUTOMGEN variable to a PLC variable)	
⊟ Linear assignment (a table of AUTOMGEN variables to a table of PLC variables)	
⊞ Only for 1720	
⊞ Only for 47	
⊞ Only for 4720	
⊞ Only for 27	
<-32-> tempo	0
<-2-> m200	IW1,0
<-2-> m202	OW2,0

Associating a table of AUTOMGEN bits to a table of target bits

Two linear assignments are necessary, for example: For example:



Linear assignment (a table of AUTOMGEN variables to a table of PLC variables)	
<-16-> tconsi0	kd0,1
<-16-> bi0	e62,0
<-16-> o0	a62,0
<-10-> u100	m2,0
<-10-> bu100	m2,0

This example (the immediate states and past states are associated to the same target variables) does not allow using the edge tests on AUTOMGEN bits u100 to u109.

Two solutions are possible for getting around this problem:

- use the syntax « ↑(u<n>) » or « ↓(u<n>) » (where « <n> » represents the bit number) in the application,
- associate the immediate and past state bits to two tables of different bits in the target.In this case, external access to the application which can be created on these bits (for example by a dialogue terminal or supervision software) must comply with the AUTOMGEN philosophy: for reading access for past states, writing access for immediate states (reading access for immediate states is possible in practice).

The allocated target bits must be cleaned of any other assignments or they may assign the same target variables twice to different AUTOMGEN variables.

See the variable functions in text format

When you click on the ▤ icon on the toolbar, you go from « tree » mode to « text » mode (format of older AUTOMGEN versions). In « Text » format you can copy and paste information into the configuration files..

Modification to « text » mode must be made by specialists, inopportune modifications can lead to compiling errors which are difficult for an inexperienced person to find.

Start-up manufacturer code, end manufacturer code

These configuration elements contain machine code for each target in text format.

The syntax to be used in these sections is similar to low level languages that can be used on each target. Observation of the code generated in pass 1 for each post-processor allows you to display the syntax to be used.

Reference to an AUTOMGEN variable

Use the syntax « _<AUTOMGEN variable name>_ » to refer to an AUTOMGEN variable (remember to add the character « b » at the beginning of the variable to access the past state of a boolean variable. For example « _bu100_ »).

Referring to an AUTOMGEN application symbol

Syntax:
|symbol name|
The character « | » is normally associated to key 6 on the keyboard.

Setting and referring to a label

« @<label name> » indicates a jump destination,
« _*<label name>*_ » refers to a label.

Entering machine code in an application

The key words « #BEGIN_MACHINE_CODE » and « #END_MACHINE_CODE » are used to enter machine codes in an AUTOMGEN code box.

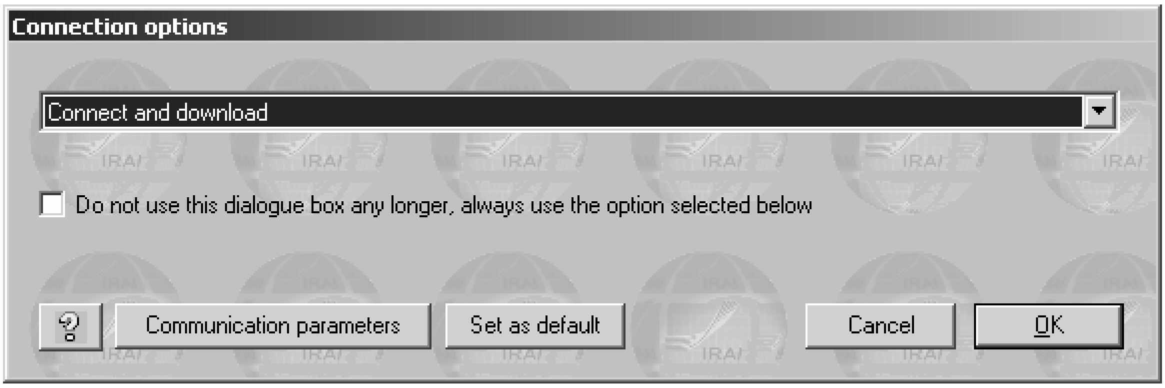
These two commands must be placed at the beginning of a line, no other characters should be placed on the same line.

The lines within these two commands establish an area called « Machine language section ».

The syntax to be used in a machine language section is the same as that used in the « Begin machine code » and « End machine code » elements.

Selecting connection options

Double click on the element « Configuration / Post-processor / <post-processor name> / Connection options».



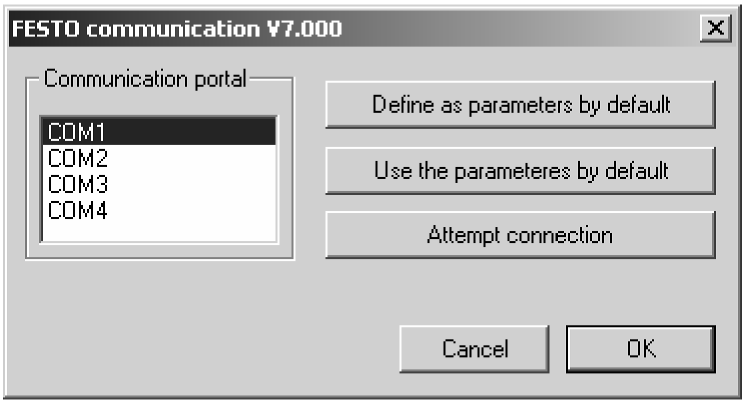
Selecting a connection mode

The number of connection modes is based on the post-processor. The « Only connected » mode is normally used to create a supervising application.

This dialogue box opens automatically when connection to a target is requested. If you check « Do not open … », it will no longer open automatically. To open it again, push the [Shift] key or launch the connection command or the « Go » command.

Setting communication module parameters

Double click on the element « Configuration / Post-processor / <post-processor name> /Communication module».


FESTO communication V7.000
Communication portal
COM1
COM2
COM3
COM4
Define as parameters by default
Use the parameteres by default
Attempt connection
Cancel
OK

Example of setting communication module parameters

The current configuration can be set as a default configuration (for new projects) or default reset.

A connection test can be created.

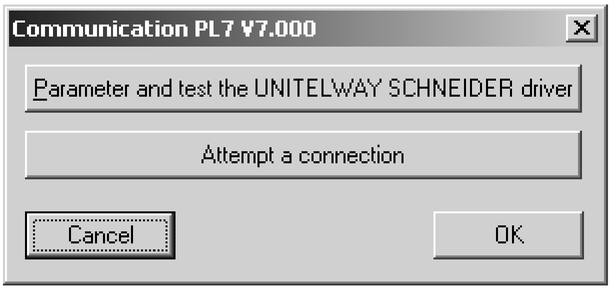
Post-processor PL7

This post-processor is used to program the processors MODICON TELEMECANIQUE SCHNEIDER TSX 37, (TSX MICRO) and TSX 57 (TSX PREMIUM).

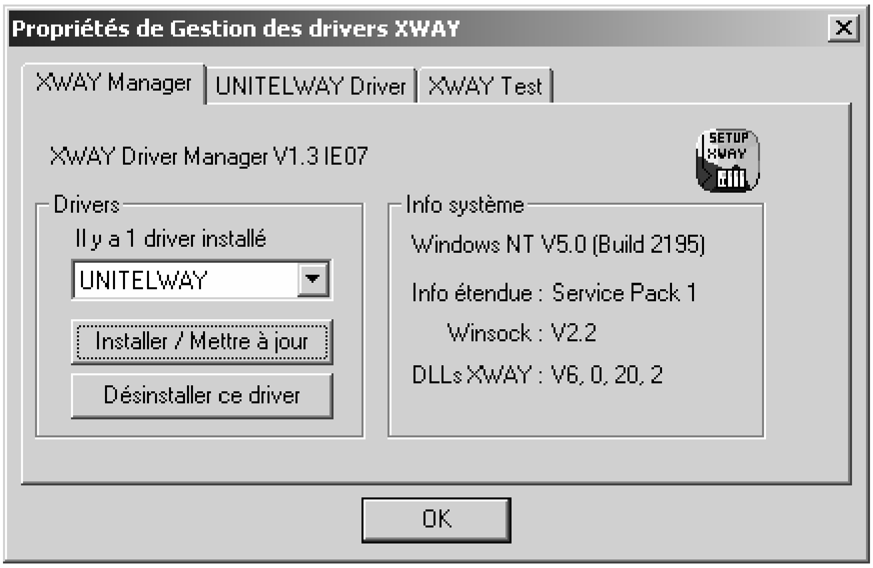
Communication module

The UNITELWAY SCHNEIDER driver must be installed on the computer (locally) to be able to communicate with processors SCHNEIDER TSX 37 and TSX 57. Drivers adapted to one or more versions of WINDOWS are on the CD-ROM and can be downloaded from the IRAI site. www.irai.com.

The communication module uses the driver conceived for SCHNEIDER AUTOMATION. Click on « Setting parameters and testing … » to directly access SCHNEIDER communication driver menus.



Setting communication module parameters



XWAY Driver Manager V1.3 IE07

Il y a 1 driver installé

UNITELWAY

Windows NT V5.0 (Build 2195)

Info étendue : Service Pack 1

Winsock : V2.2

DLLs XWAY : V6, 0, 20, 2

UNITELWAY communication module properties

Generating an executable file

The post-processor can generate a binary file which can be directly downloaded in the processor (only available on TSX 37, not available on TSX 57) or a file which can be imported in the SCHNEIDER tools (available for TSX 37 and TSX 57). The first solution is preferable (saves time, easier to use).

The mode selection is made in the post-processor software configuration.

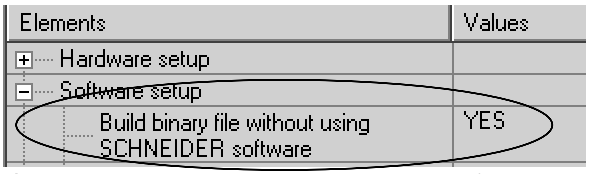
Direct generation of a binary file

This mode is highly recommended for TSX 37.

It has the following restrictions:

- sub-program instruction cannot be used;
- it does not support specific instructions (for example communication or PID).

If your application must use very specific elements, use one of the importation methods described above.



Elements	Values
⊞ Hardware setup	
⊟ Software setup	
Build binary file without using SCHNEIDER software	YES

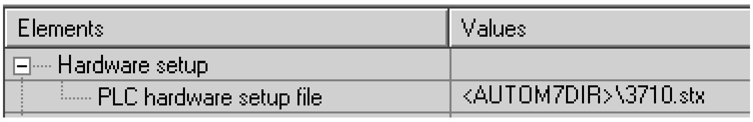
Selecting automatic generation of a binary file.

Processor configuration file

For the basic versions of TSX 37-05 and TSX 37-08 (with only one input/output card), files 3705.stx and 3708.stx are provided in the AUTOMGEN installation directory.

If a configuration file is not created, the processor TOR outputs will not be activated.

Once the file is created or downloaded (see above), give an access path to the file in the following configuration element:



Elements	Values
⊟ Hardware setup	
PLC hardware setup file	<AUTOM7DIR>\3710.stx

File name containing the configuration

There are three possible methods for obtaining a configuration file:

Download the configuration file from the IRAI site

1- download a file corresponding to the configuration of your processor from the IRAI site: www.irai.com, « Download / AUTOMGEN7 / configuration files for TSX 37 processor » section (recommended if the configuration file is present on the site),

2- recopy the downloaded file without unzipping it (« .STX » files are zipped files) into the AUTOMGEN installation directory or put it into AUTOMGEN project resources.

If you unzip the « .STX » file the post-processor will not be able to use it.

Creating a file with the SCHNEIDER programming tools

SCHNEIDER (PL7MICRO V3.1, PL7JUNIOR V3.1 or PL7PRO V3.4) software tools can be used.. Files created with other versions may not work, in this case, the processor goes into error mode when the application is downloaded (« ERR » light goes on on the processor).

To create an « .STX » file :

1- launch one of the SCHNEIDER tools, create an application following the rules below:
   - select your processor's type of CPU and always select the 1.0 version of the CPU;
   - select the input/output card/s on your processor and if necessary set their parameters;

2- save the file created in the AUTOMGEN installation directory or enter it into AUTOMGEN project resources.

Send an e-mail to IRAI to obtain the configuration file

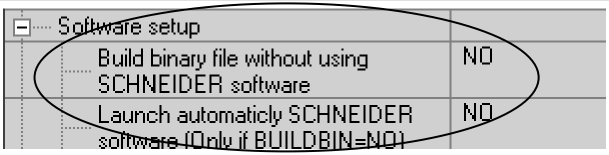
1- send an e-mail to IRAI requesting a configuration file, the following information needs to be provided:
   - the type of CPU TSX 37-05, 37-08, 37-10, 37-21 or 3722,
   - the position and exact type of input/output cards (DMZ …)

2- when you receive the file, recopy it into the AUTOMGEN installation directory (without unzipping it) or put it into AUTOMGEN project resources.

Generating an « .FEF » executable file

In this mode, importation in the SCHNEIDER programming tools (PL7 MICRO (TSX 37), PL7 JUNIOR (TSX 37 or TSX 57) or PL7 PRO (TSX 37 or TSX 57) can be automatic or manual.

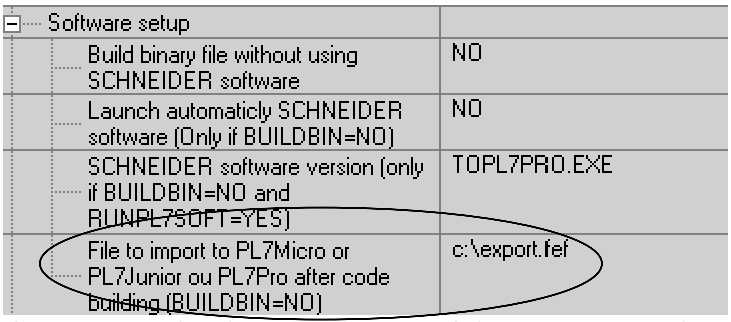
Manual importation



⊟ Software setup	
Build binary file without using SCHNEIDER software	NO
Launch automaticly SCHNEIDER software (Only if BUILDBIN=NO)	NO

Selecting manual importation mode

You must select a file name which will be exported for AUTOMGEN:



⊟ Software setup	
Build binary file without using SCHNEIDER software	NO
Launch automaticly SCHNEIDER software (Only if BUILDBIN=NO)	NO
SCHNEIDER software version (only if BUILDBIN=NO and RUNPL7SOFT=YES)	TOPL7PRO.EXE
File to import to PL7Micro or PL7Junior ou PL7Pro after code building (BUILDBIN=NO)	c:\export.fef

Selecting a file for exporting to SCHNEIDER software workgroup

Procedure:

1- Compile the application in AUTOMGEN using the « Compile » command from the « Program » menu or clicking on the button on the toolbar,

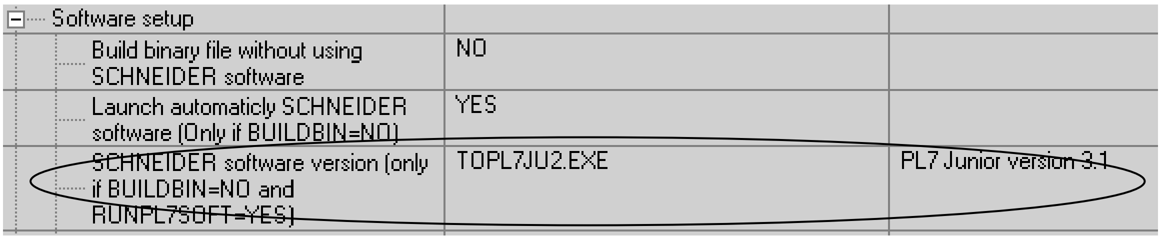
2- Launch a SCHNEIDER software workgroup, create a new project and use the « Import an application» command from the « File » menu,

3- When the importation process is finished, transfer the application to the processor.

4- To obtain a dynamic display in AUTOMGEN, click the « Go » button on the toolbar and select the « Only connect» connection mode.

Automatic importation

The SCHNEIDER software tool will be launched automatically. Only a limited number of SCHNEIDER software can be used. The type and version of SCHNEIDER software must be set in the software configuration.



⊟ Software setup		
Build binary file without using SCHNEIDER software	NO	
Launch automaticly SCHNEIDER software (Only if BUILDBIN=NO)	YES	
SCHNEIDER software version (only if BUILDBIN=NO and RUNPL7SOFT=YES)	TOPL7JU2.EXE	PL7 Junior version 3.1

Selecting the type and version of SCHNEIDER software

Operation of the automatic importation procedure with other SCHNEIDER software versions is not guaranteed.

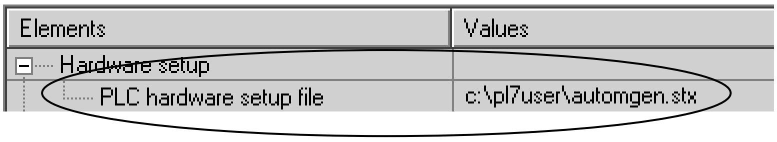
Procedure to be effected only once.

1- Launch a SCHNEIDER programming tool and create a new application.

2- Configure the application: the type of processor, input/output cards etc.

3- Save the file you have created;

4- Give the complete access path to that file in the « hardware configuration» section of the « System» element, for example:



Elements	Values
⊟ Hardware setup	
PLC hardware setup file	c:\pl7user\automgen.stx

Create an application for each execution:
Launch the SCHNEIDER software tool (if you have not already done so);

Click the « GO » button on the AUTOMGEN toolbar.

Using interrupt tasks

When setting a task type sheet, you can enter AUTOMGEN low level language or machine language for a processor task. The table below provides the correspondence in the number of tasks and the type of interrupr task of the processor.

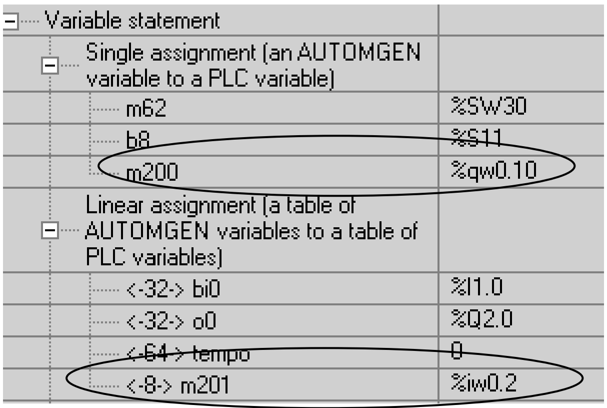
Task number (AUTOMGEN sheet)	Processor TSX 37 task type	Processor TSX 57 task type
0	Fast task	Fast task
1	EVT1	EVT0
2	EVT2	EVT1
3	EVT3	EVT2
etc…		

Specific examples

These examples are in the directory « <AUTOMGEN installation directory> /Examples/Post-processors/PL7 ».The files have the same names as the titles of the following chapters.

Analog inputs/outputs

This example illustrates the use of analog inputs/outputs.



Variable statement	
Single assignment (an AUTOMGEN variable to a PLC variable)	
m62	%SW30
b8	%S11
m200	%qw0.10
Linear assignment (a table of AUTOMGEN variables to a table of PLC variables)	
<-32-> bi0	%I1.0
<-32-> o0	%Q2.0
<-64-> tempo	0
<-8-> m201	%iw0.2

Stating analog inputs/outputs on processor TSX 37-22

Fast counter TSX 37-10

This example illustrates the use of a fast counter on a TSX 37-10 processor.

Fast counter TSX 37-10 used in counting

This example illustrates the use of a fast counter on a TSX 37-10 processor in counting mode.

Fast counter TSX 37-22

This example illustrates the use of a fast counter on a TSX 37-22 processor.

ASI

Example of using ASI inputs/outputs

MAGELIS

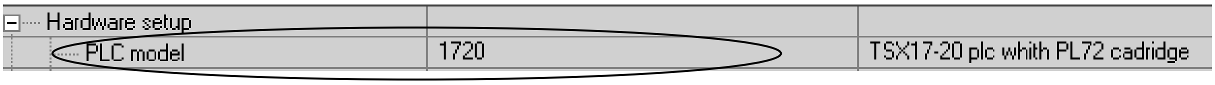
Example of using a MAGELIS terminal

Post-processor PL72

This post-processor is used to program TELEMECANIQUE SCHNEIDER TSX 17-20 (with PL72 cartridge), TSX 27, TSX 47 and TSX 47-20 processors.

Selecting processor type

Use the « Configuration / Post-processor / PL72 / System / Hardware configuration» browser element to select the type of processor.



⊟ Hardware setup		
PLC model	1720	TSX17-20 plc whith PL72 cadridge

Specific syntax elements

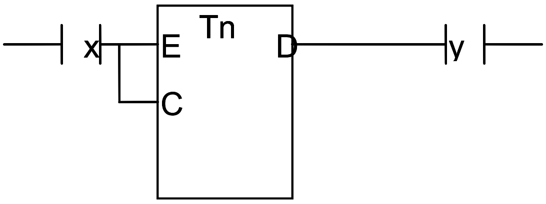
Calling up PL72 function blocks

The following syntaxes are used to call up time delay, text and fast counter (TSX 17-20) blocks in a text format used in the « Begin machine code », « End machine code » elements and sections in machine language.

Time delay block

x.Tn=y

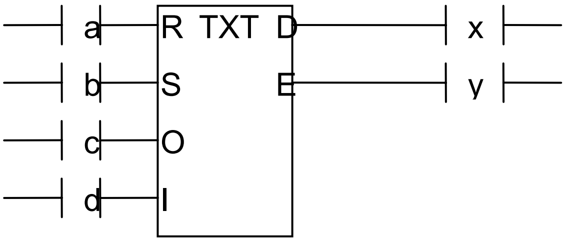
PL72 equivalent:



Text block

a+b+c+d.TXTn=x:y

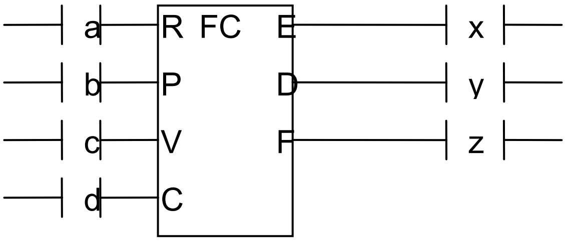
PL72 equivalent:



Fast counter block

a+b+c+d.FC=x:y:z

PL72 equivalent:



WEEK type time/date block (only on TSX 17-20)

a.H,W (days), (start time), (end time)=x :y :z

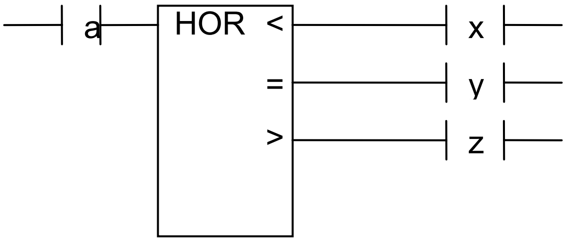
« days » represents the days of the week, this is an encoded value on 7 bits, each bit represents a day of the week. The day is active if the bit 1.b0 corresponds to Sunday and b6 to Saturday. For example, to validate Monday and Wednesday the value $2^1 + 2^3$ must be written: 2 + 8 = 10. To validate the seven days of the week: the value is 127.

« start time » and « end time » are expressed as HH:MM: hours and minutes:

YEAR type time/date block (only on TSX 17-20)
a.H,Y,(start date),(end date)=x :y :z

« start date » and « end date » are expressed as DD:MM: day and month.

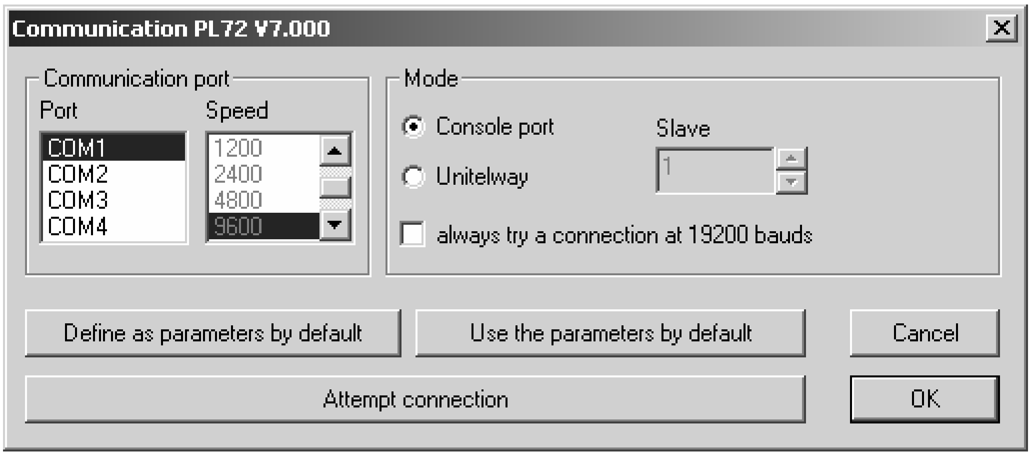
PL72 equivalent:



Using a fast task

A« Task » type sheet bearing the number « 1 » is used to associate literal code or PL72 code written on the sheet to the fast task. The sheet must not contain anything other than low level literal code or PL72 code written in an organizational chart rectangle.

Communication module



Communication PL72 V7.000

Communication port

Port: COM1, COM2, COM3, COM4 (COM1 selected)

Speed: 1200, 2400, 4800, 9600 (9600 selected)

Mode

● Console port

○ Unitelway

Slave: 1

□ always try a connection at 19200 bauds

Define as parameters by default
Use the parameters by default
Cancel
Attempt connection
OK

Setting communication module parameters

If you connect the PC on the processor console outlet, you must select « Console outlet».

Do not check « Always attempt 19200 baud connection » unless your processor is a recent TSX 17-20 (this option is used for a faster dialogue between the PC and processor).

The « UNITELWAY » mode is used to connect the PC to a UNITELWAY coupler. In this case, the speed must correspond to the coupler configuration.

If you check « Always attempt 19200 baud connection » and your TSX 17-20 does not support 19200 baud communication the connection will fail.

If the mode does not correspond with the connection (for example, UNITELWAY mode selected and a connection on the console processor) the connection will fail.

Specific examples

These examples are in the directory « <AUTOMGEN installation directory> /Examples/Post-processors/PL72 ».The files have the same names as the titles of the following chapters.

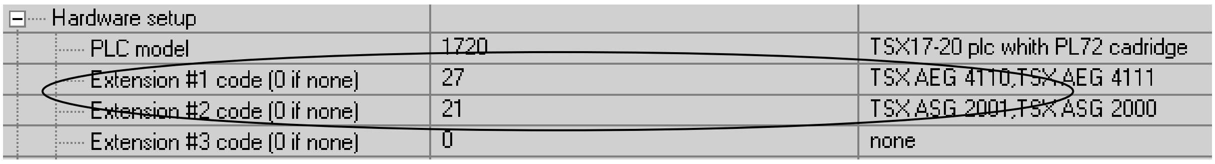
Analog inputs/outputs

To use the analog outputs on a TSX 17-20 processor you must:

- state the analog input/output blocks in the « System » element of the configuration .
- associate one or more AUTOMGEN variables to the TELEMECANIQUE input/output words (IW and OW).

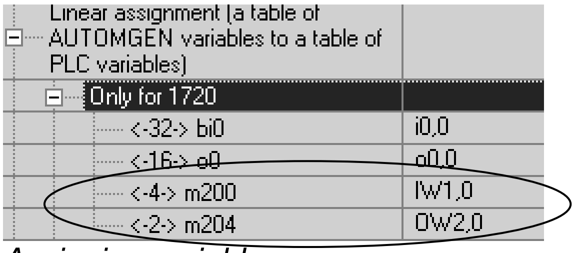
Example:

- processor TSX 17-20 using a block of 4 analog inputs (code 27) in position 1 and a block of 2 analog outputs (code 21) in position 2:
- the program will simply recopy the state of the first analog input on the first analog output. it will also compare the second analog input with the value 500 (arbitrary value) and position two boolean outputs: O0 if the input is less than 500 and O1 if the input is greater than or equal to 500.



Hardware setup		
PLC model	1720	TSX17-20 plc whith PL72 cadridge
Extension #1 code (0 if none)	27	TSX AEG 4110,TSX AEG 4111
Extension #2 code (0 if none)	21	TSX ASG 2001,TSX ASG 2000
Extension #3 code (0 if none)	0	none

Statement of two extension modules



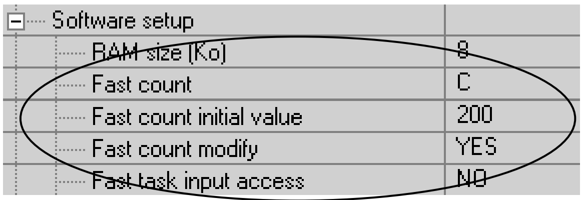
Linear assignment (a table of AUTOMGEN variables to a table of PLC variables)	
Only for 1720	
<-32-> bi0	i0,0
<-16-> o0	o0,0
<-4-> m200	IW1,0
<-2-> m204	OW2,0

Assigning variables

These two statements associate the AUTOMGEN words M200 to M203 to processor variables IW1,0 to IW1,3 as well as AUTOMGEN variables M204 and M205 to processor variables OW2,0 and OW2,1.

Fast counter

The goal is to count 200 pulses on the fast counter. The current value of the fast counter will be recopied in AUTOMGEN word M200. Output O5 will be activated by a interrupt task at the end of the count.



⊟ Software setup	
RAM size (Ko)	8
Fast count	C
Fast count initial value	200
Fast count modify	YES
Fast task input access	NO

Setting fast counter parameters

Text blocks and xbt

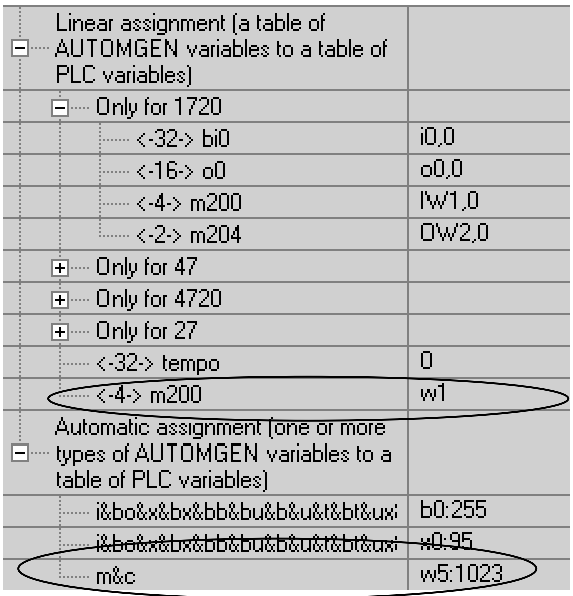
The goal is to dialogue with an XBT connected on the console port of processor TSX 17-20.

Inputs I0 to I3 start displaying messages number 0 to number 3 registered in the XBT.

Text block TXT1 will be used to dialogue on the console port.

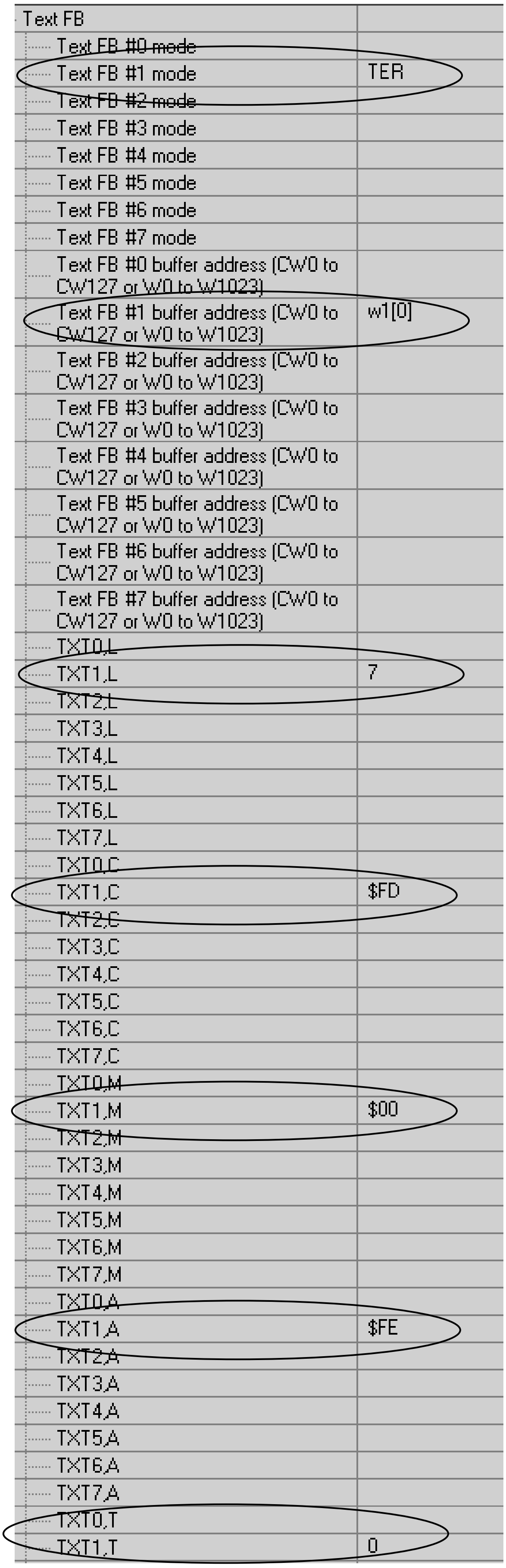
The message format to send to the XBT for displaying a message is as follows:
ESC V xxx LF CR

xxx represents the number of the message encoded in three character decimals.



⊟ Linear assignment (a table of AUTOMGEN variables to a table of PLC variables)	
⊟ Only for 1720	
<-32-> bi0	i0,0
<-16-> o0	o0,0
<-4-> m200	IW1,0
<-2-> m204	OW2,0
⊞ Only for 47	
⊞ Only for 4720	
⊞ Only for 27	
<-32-> tempo	0
<-4-> m200	w1
⊟ Automatic assignment (one or more types of AUTOMGEN variables to a table of PLC variables)	
i&bo&x&bx&bb&bu&b&u&t&bt&uxi	b0:255
i&bo&x&bx&bb&bu&b&u&t&bt&uxi	x0:95
m&c	w5:1023

Allocation of a word table for exchanges

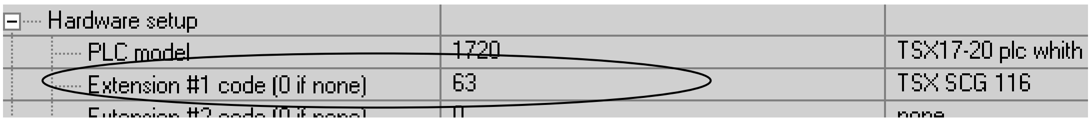


Text FB	
Text FB #0 mode	
Text FB #1 mode	TER
Text FB #2 mode	
Text FB #3 mode	
Text FB #4 mode	
Text FB #5 mode	
Text FB #6 mode	
Text FB #7 mode	
Text FB #0 buffer address (CW0 to CW127 or W0 to W1023)	
Text FB #1 buffer address (CW0 to CW127 or W0 to W1023)	w1[0]
Text FB #2 buffer address (CW0 to CW127 or W0 to W1023)	
Text FB #3 buffer address (CW0 to CW127 or W0 to W1023)	
Text FB #4 buffer address (CW0 to CW127 or W0 to W1023)	
Text FB #5 buffer address (CW0 to CW127 or W0 to W1023)	
Text FB #6 buffer address (CW0 to CW127 or W0 to W1023)	
Text FB #7 buffer address (CW0 to CW127 or W0 to W1023)	
TXT0,L	
TXT1,L	7
TXT2,L	
TXT3,L	
TXT4,L	
TXT5,L	
TXT6,L	
TXT7,L	
TXT0,C	
TXT1,C	$FD
TXT2,C	
TXT3,C	
TXT4,C	
TXT5,C	
TXT6,C	
TXT7,C	
TXT0,M	
TXT1,M	$00
TXT2,M	
TXT3,M	
TXT4,M	
TXT5,M	
TXT6,M	
TXT7,M	
TXT0,A	
TXT1,A	$FE
TXT2,A	
TXT3,A	
TXT4,A	
TXT5,A	
TXT6,A	
TXT7,A	
TXT0,T	
TXT1,T	0

Setting text block parameters

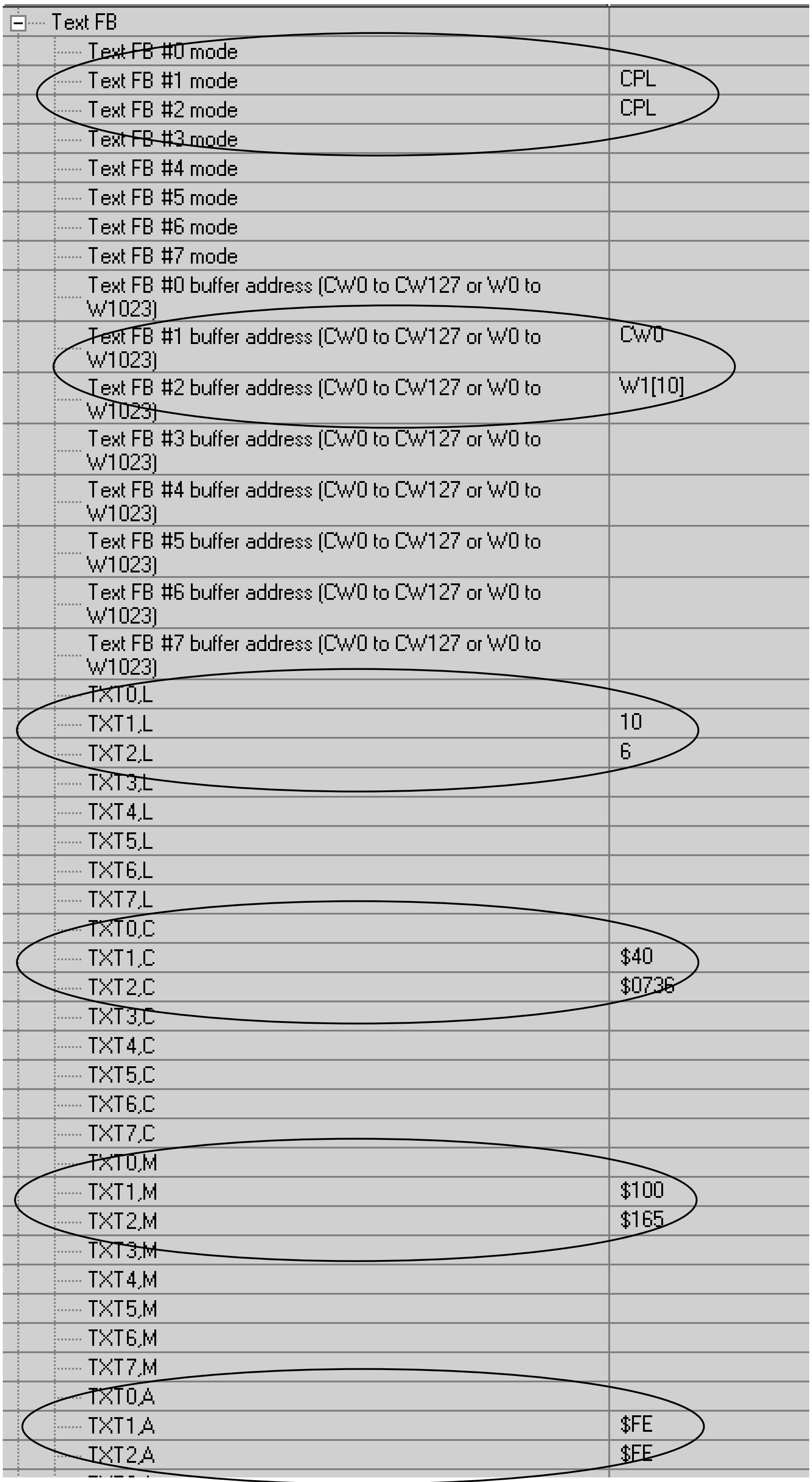
UNITELWAY text blocks

The goal is to use a UNITELWAY coupler to acquire a table of 3 words on a target processor. The UNITELWAY coupler is installed as a first extension, it will be configured as the master to use two slaves. Processor Iu will be slave number 1.



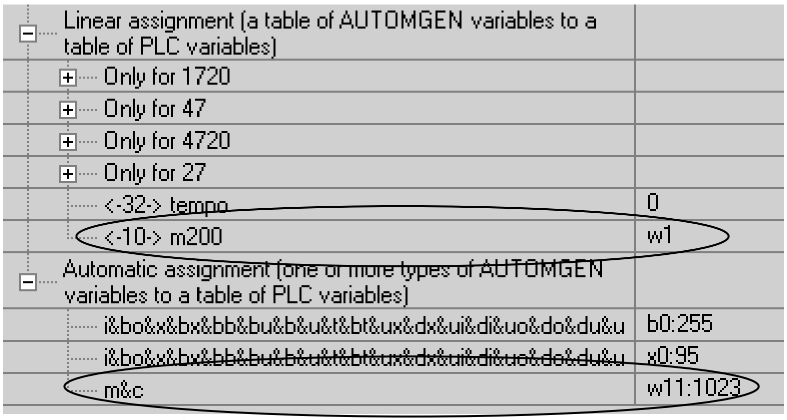
Hardware setup		
PLC model	1720	TSX17-20 plc whith
Extension #1 code (0 if none)	63	TSX SCG 116
Extension #2 code (0 if none)	0	none

Configuration of an UNITELWAY coupler



Text FB	
Text FB #0 mode	
Text FB #1 mode	CPL
Text FB #2 mode	CPL
Text FB #3 mode	
Text FB #4 mode	
Text FB #5 mode	
Text FB #6 mode	
Text FB #7 mode	
Text FB #0 buffer address (CW0 to CW127 or W0 to W1023)	
Text FB #1 buffer address (CW0 to CW127 or W0 to W1023)	CW0
Text FB #2 buffer address (CW0 to CW127 or W0 to W1023)	W1[10]
Text FB #3 buffer address (CW0 to CW127 or W0 to W1023)	
Text FB #4 buffer address (CW0 to CW127 or W0 to W1023)	
Text FB #5 buffer address (CW0 to CW127 or W0 to W1023)	
Text FB #6 buffer address (CW0 to CW127 or W0 to W1023)	
Text FB #7 buffer address (CW0 to CW127 or W0 to W1023)	
TXT0,L	
TXT1,L	10
TXT2,L	6
TXT3,L	
TXT4,L	
TXT5,L	
TXT6,L	
TXT7,L	
TXT0,C	
TXT1,C	$40
TXT2,C	$0736
TXT3,C	
TXT4,C	
TXT5,C	
TXT6,C	
TXT7,C	
TXT0,M	
TXT1,M	$100
TXT2,M	$165
TXT3,M	
TXT4,M	
TXT5,M	
TXT6,M	
TXT7,M	
TXT0,A	
TXT1,A	$FE
TXT2,A	$FE

Setting parameters for two text blocks

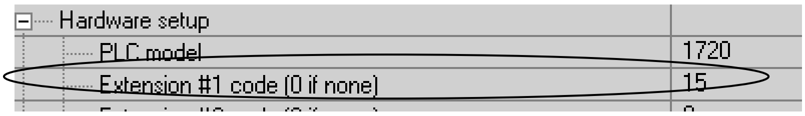


Linear assignment (a table of AUTOMGEN variables to a table of PLC variables)	
Only for 1720	
Only for 47	
Only for 4720	
Only for 27	
<-32-> tempo	0
<-10-> m200	w1
Automatic assignment (one or more types of AUTOMGEN variables to a table of PLC variables)	
i&bo&x&bx&bb&bu&b&u&t&bt&ux&dx&ui&di&uo&do&du&u	b0:255
i&bo&x&bx&bb&bu&b&u&t&bt&ux&dx&ui&di&uo&do&du&u	x0:95
m&c	w11:1023

Attribution of a word table for exchanges

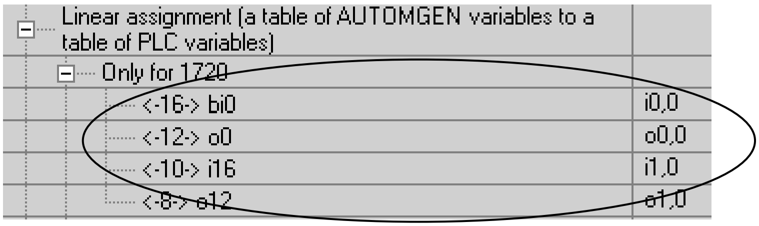
TOR extension module

This example illustrates the configuration of a TOR extension module. We are using a basic module equipped with 16 inputs and 12 outputs and an extension module equipped with 10 inputs and 8 outputs.



Hardware setup	
PLC model	1720
Extension #1 code (0 if none)	15

Setting the extension module



Linear assignment (a table of AUTOMGEN variables to a table of PLC variables)	
Only for 1720	
<-16-> bi0	i0,0
<-12-> o0	o0,0
<-10-> i16	i1,0
<-8-> o12	o1,0

Assigning variables

Conversion

Shows how to call up the PL72 language conversion functions.

Time/date

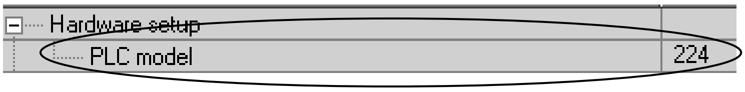
Example of using a time/date function block.

Post-processor S7200

This post-processor is used to program SIEMENS S7200 (all 2xx CPU's) processors .

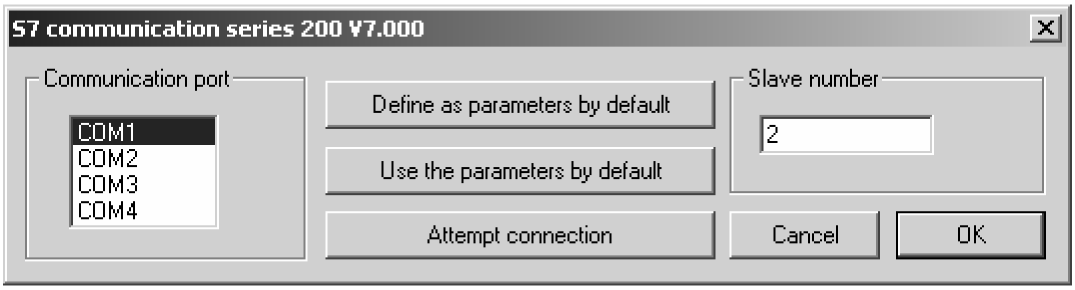
Selecting CPU type

Use the « Configuration / Post-processor / STEP7 (S7200) / System / Hardware configuration» browser element to select the type of CPU.



Hardware setup
PLC model: 224

Communication module



S7 communication series 200 V7.000

Communication port: COM1 (selected), COM2, COM3, COM4

Slave number: 2

Define as parameters by default
Use the parameters by default
Attempt connection
Cancel
OK

Setting communication module parameters

Be sure to set the slave number so it corresponds with the processor configuration.

Specific example

This example is in the directory « <AUTOMGEN installation directory> /Examples/Post-processors/S7200 ».The file has the same name as the title of the following chapter.

Interrupt task

Example of calling up an interrupt task

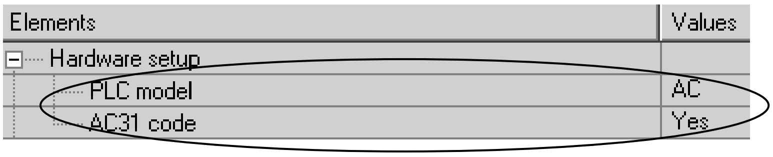
Post-processor ABB

This post-processor is used to program ABB CS31 and AC31 processors .

Selecting processor type

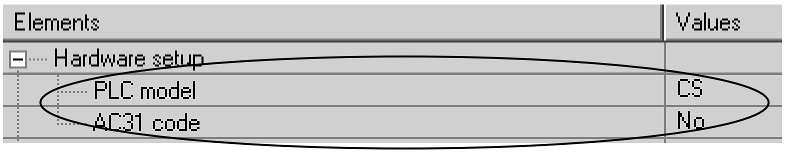
Use the « Configuration / Post-processor / ABB / System / Hardware configuration» browser element to select the type of processor.

Processor AC31



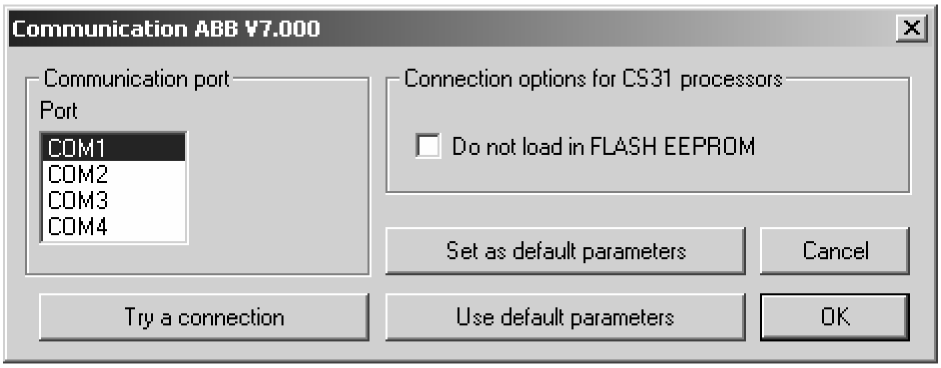
Elements	Values
⊟ Hardware setup	
PLC model	AC
AC31 code	Yes

Processor CS31



Elements	Values
⊟ Hardware setup	
PLC model	CS
AC31 code	No

Communication module



Communication ABB V7.000

Communication port

Port: COM1

Connection options for CS31 processors

□ Do not load in FLASH EEPROM

Setting communication module parameters

Utility

The « Configuration / Post-processor / ABB / Terminal emulator » browser element is used to access a terminal emulator for dialoguing with the processor.

Specific examples

These examples are in the directory « <AUTOMGEN installation directory> /Examples/Post-processors/ABB ». The files have the same names as the following chapters:

Analog inputs/outputs

The example illustrates the use of an analog extension module on processor AC31.

Interrupt

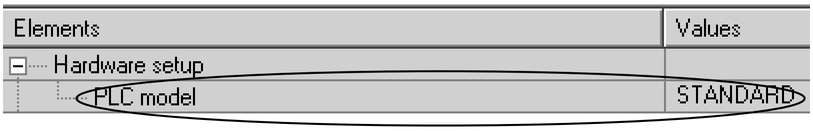
The example illustrates the use of interrupt tasks on processor AC31.

Post-processor GE-FANUC / ALSPA

This post-processor is used to program GE-FANUC 90 MICRO and 9030 or ALSPA 8005 and 8035 processors.

Selecting processor type

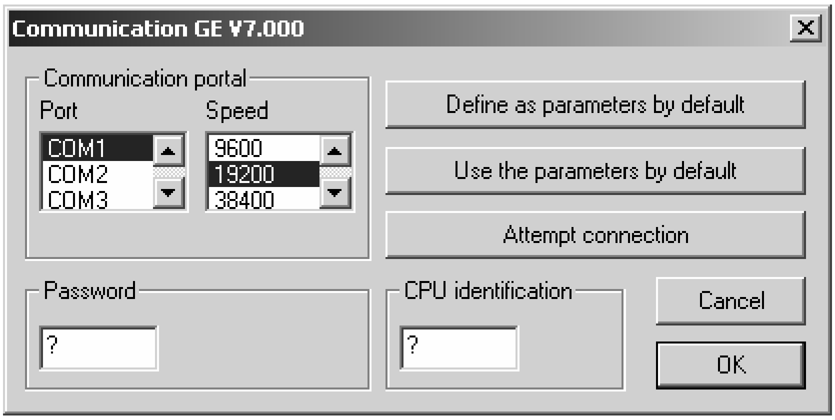
Use the « Configuration / Post-processor / GE-FANUC / System / Hardware configuration» browser element to select the type of processor.



Elements	Values
⊟ Hardware setup	
PLC model	STANDARD

Select standard for CPU's other than 350, 351 or VERSAMAX.

Communication module



Communication GE V7.000

Communication portal

Port: COM1 (COM2, COM3)

Speed: 19200 (9600, 38400)

Password: ?

CPU identification: ?

Define as parameters by default
Use the parameters by default
Attempt connection
Cancel
OK

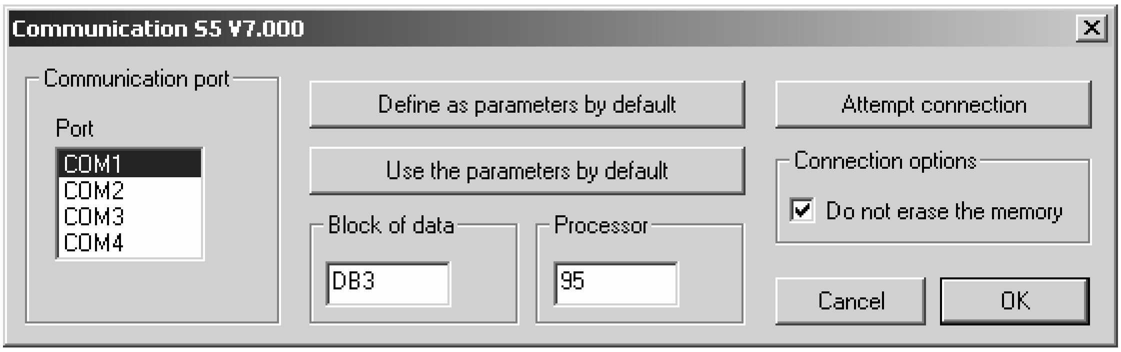
Setting communication module parameters

Utility

The « Configuration / Post-processor / GE-FANUC / Hardware configuration & diagnostic » browser element is used to access a configuration and diagnostic utility.

Post-processor STEP5

Communication module



Communication S5 V7.000

Communication port

Port: COM1

Block of data: DB3

Processor: 95

Connection options

☑ Do not erase the memory

COM1
COM2
COM3
COM4
Define as parameters by default
Use the parameters by default
Attempt connection
Cancel
OK

Setting communication module parameters

Application structure

SIEMENS STEP5 language is organized in program and data blocks. AUTOMGEN applications translated by the STEP5 post-processor are broken down into multiple blocks. By default, the post-processor uses the following blocks

- OB1 organization block this block calls up all the blocks that must be cyclically processed.

- OB20, OB21, OB22 : blocks executed when the processor starts. These blocks set a bit to activate the initial Grafcet steps.

The PB blocks are used to process pre-settings, to manage the evolution of boolean variables and time delays.

The FB or FX blocks are used for the code issued by the application and for the code written in the « .SRT » and « .END » files. An FB or FX file is created for each application sheet.

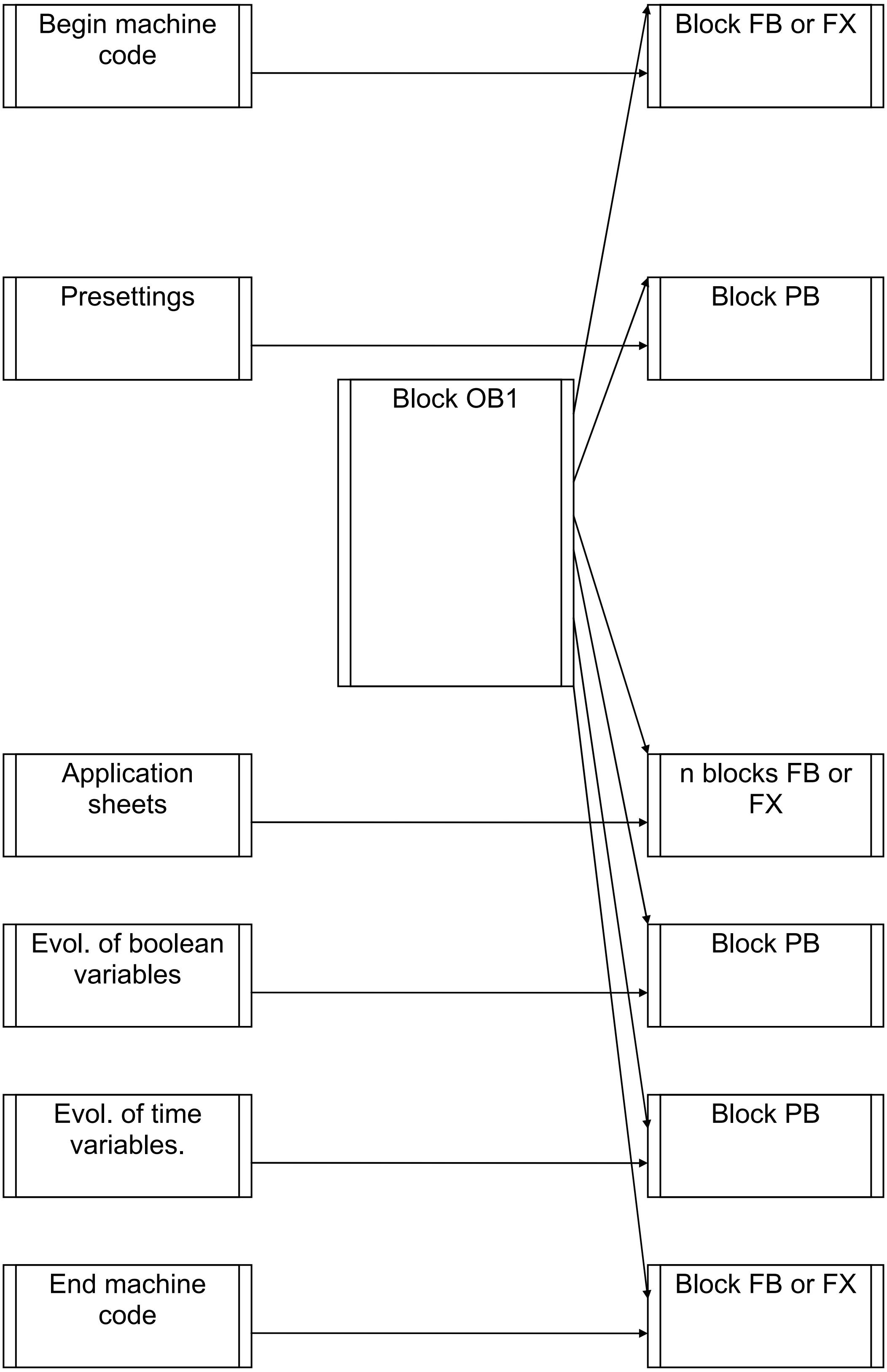
In addition, the sheets can be directly associated to a block of codes or data.

If the volume of generated code is too much for a block (for example, code issued from a sheet containing a voluminous program), the post-processor automatically uses a new block.

By default, the post-processor uses the blocks PB1 to PB255 and FB1 to FB239 when needed.

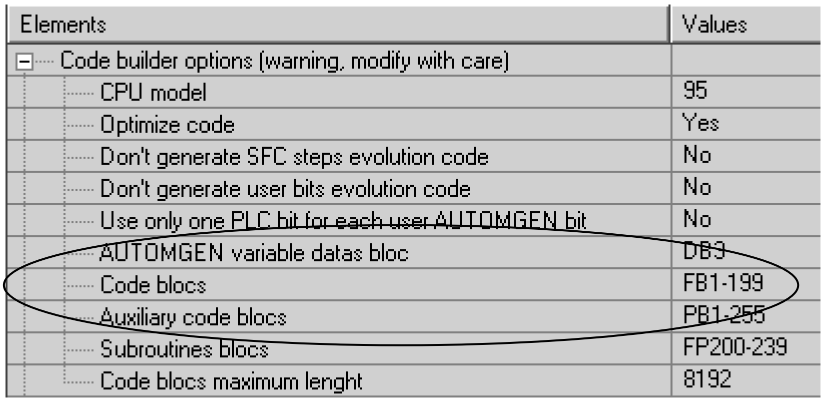
These values can be modified, (see the chapter Selecting program blocks to use).

The figure below illustrates the structure of code generated by the SIEMENS post-processor.



Selecting program blocks to use

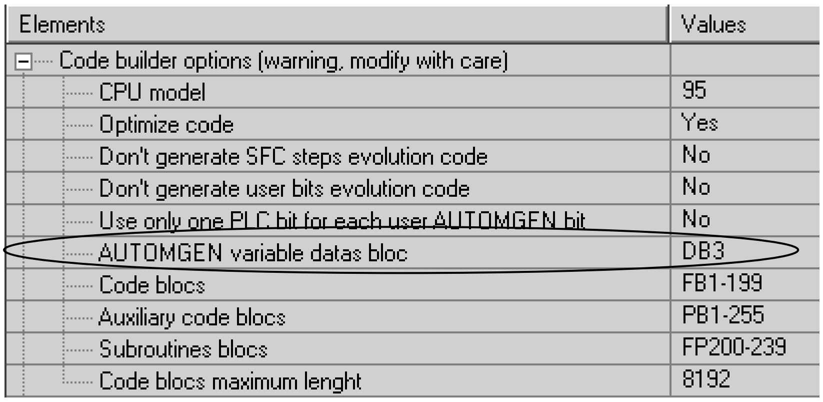
Blocks PB1 to PB 255 and FB 1 to FB 239 are used by default. Three configuration elements are used to select other blocks.



Elements	Values
⊟ Code builder options (warning, modify with care)	
CPU model	95
Optimize code	Yes
Don't generate SFC steps evolution code	No
Don't generate user bits evolution code	No
Use only one PLC bit for each user AUTOMGEN bit	No
AUTOMGEN variable datas bloc	DB3
Code blocs	FB1-199
Auxiliary code blocs	PB1-255
Subroutines blocs	FP200-239
Code blocs maximum lenght	8192

Selecting data blocks

Block DB 3 is used by default for numeric variables. This command is used to select another block.



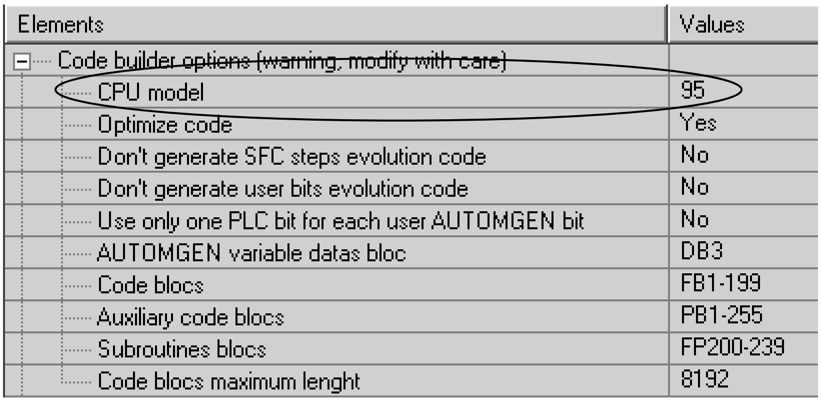
Elements	Values
⊟ Code builder options (warning, modify with care)	
CPU model	95
Optimize code	Yes
Don't generate SFC steps evolution code	No
Don't generate user bits evolution code	No
Use only one PLC bit for each user AUTOMGEN bit	No
AUTOMGEN variable datas bloc	DB3
Code blocs	FB1-199
Auxiliary code blocs	PB1-255
Subroutines blocs	FP200-239
Code blocs maximum lenght	8192

Changing the data block involves two other modifications.

- in the begin machine code, it is necessary to create corresponding data blocks,

- it is necessary to select the data block in the dialogue module parameter settings.

Selecting processor type



Elements	Values
Code builder options (warning, modify with care)	
CPU model	95
Optimize code	Yes
Don't generate SFC steps evolution code	No
Don't generate user bits evolution code	No
Use only one PLC bit for each user AUTOMGEN bit	No
AUTOMGEN variable datas bloc	DB3
Code blocs	FB1-199
Auxiliary code blocs	PB1-255
Subroutines blocs	FP200-239
Code blocs maximum lenght	8192

Associating code written on a sheet to a program block

When writing low level literal code or machine code on an organizational chart on a « Task » type sheet, associate the code to a STEP5 block.

The task number determines the block type and number.

The generated code for that sheet must take into account the block type and the instructions used in that block type (the set of instructions for blocks OB and PB is limited).

The table below provides the correspondence between the value « n » and the block.

Task number	*STEP5 block*
0 to 255	OB 0 to OB 255
256 to 511	PB 0 to PB 255
512 to 767	FB 0 to FB 255
768 to 1023	FX 0 to FX 255
1024 to 1279	DB 0 to DB 255
1280 to 1535	DX 0 to DX 255

Specific syntaxes

Setting blocks

The command « $BLOCK <block type> <number>» is used to set the beginning of a program or data block.

The block type can be « OB », « FB », « FX », « PB » for the code or « DB », « DX » for the data..

The block number is a value between 0 and 255. Blocks « FX » and « DX » can only be used on processors 135U and 155U. The command « BE » indicates the end of a block.

Example:

```
$BLOCK DB1
...
$BE
$BLOCK OB1
...
$BE
```

the commands « KH= », « KY= », « KC= », « KM= » and « KF= » add constants to the data blocks DB.

« KH= » adds a 16 bit constant expressed in hexadecimal.
« KY= » adds a 16 bit constant expressed in the form of two values.
between 0 and 255 separated by a comma.
« KC= » adds a series of characters surrounded by the characters
« ' » (apostrophe).
« KM= » adds a 16 bit constant expressed in binary.
« KF= » adds a 16 bit constant expressed in signed decimal.

Example:

```
$BLOCK DB4
KH= 1234
KY=100.5
KC= 'This is a text'
KM=11111111 00000000
KF=-200
$BE
```

Blocks FB and FX

SIEMENS language blocks FB and FX can have parameters.

Call up

If the parameters need to go to a functional block , use the following syntax:

- call up needs to be followed by the character « * »,
- the following line must contain a jump instruction « SPA » to the line beyond the parameters;
- the following lines must contain the parameters preceded by the mnemonic « U » for bits or « L » for words. The constants must be written in the form « Kx=value» (x is the type of constant, see the chapter Setting blocks).

Example of calling up a functional block without parameters:

```
SPA FB 5
```

Examples of calling up a functional block with parameters:

```
SPA FB 242*
SPA=_label_
L MW10     ; first parameter
L MW12     ; second parameter
U M0.0     ; third parameter
L MW14     ; fourth parameter
L MW16     ; fifth parameter
@label
SPA FB200*
SPA=_label2_
KY=0,4     ; Example of a constant parameter
@label2
```

Writing

In blocks FB and FX mnemonics must be used ending with the character '=' followed by a parameter number (1=first parameter).

Example of a two parameter function block (first parameter recopied in the second):

```
$BLOCK FB 100
L=1
T=2
$BE
```

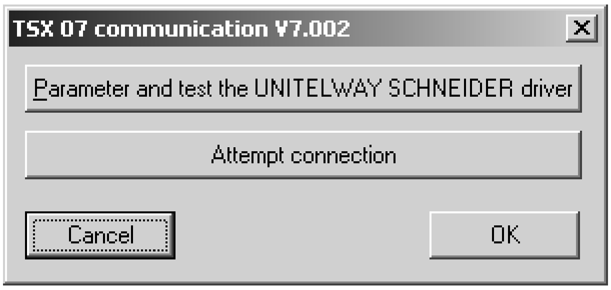
Post-processor TSX 07

This post-processor is used to program the processors MODICON TELEMECANIQUE SCHNEIDER TSX 07.

Communication module

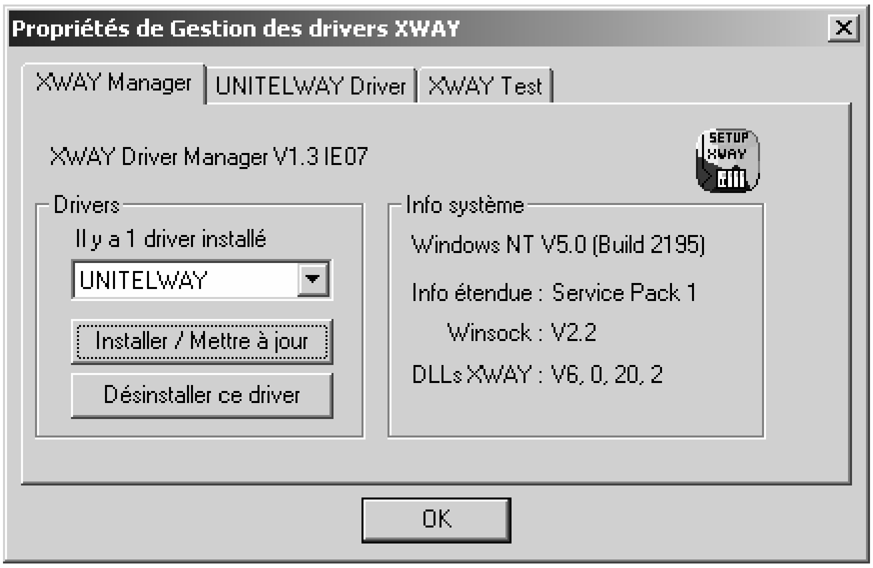
The UNITELWAY SCHNEIDER driver must be installed on the computer (locally) to be able to communicate with processors SCHNEIDER TSX 07. Drivers adapted to one or more versions of WINDOWS are on the CD-ROM and can be downloaded from the IRAI site: www.irai.com. Be careful, some versions of WINDOWS are incompatible with certain types of TSX 07 processors (TSX 0720 … and TSX 0721…are incompatible with WINDOWS ME, WINDOWS 2000 or WINDOWS XP).

The communication module uses the driver conceived for SCHNEIDER AUTOMATION. Click on « Setting parameters and testing … » to directly access SCHNEIDER communication driver menus.



TSX 07 communication V7.002
Parameter and test the UNITELWAY SCHNEIDER driver
Attempt connection
Cancel
OK

Setting communication module parameters



Propriétés de Gestion des drivers XWAY

XWAY Driver Manager V1.3 IE07

Drivers: Il y a 1 driver installé — UNITELWAY

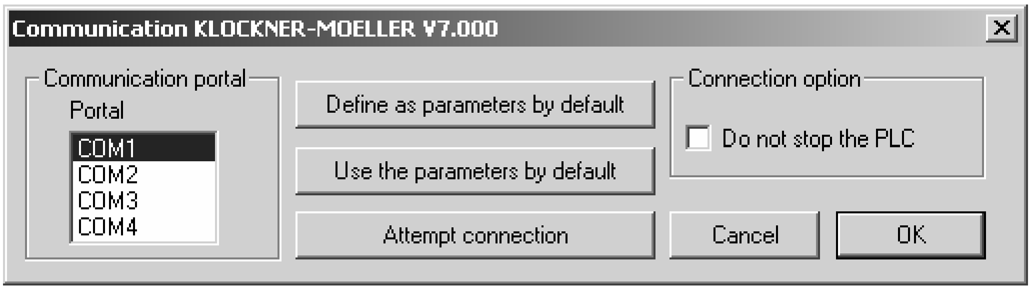
Info système:
Windows NT V5.0 (Build 2195)
Info étendue : Service Pack 1
Winsock : V2.2
DLLs XWAY : V6, 0, 20, 2

UNITELWAY communication module properties

Post-processor PS3-PS4

This post-processor is used to program KLOCKNER-MOELLER PS3 and PS4 processors..

Communication module

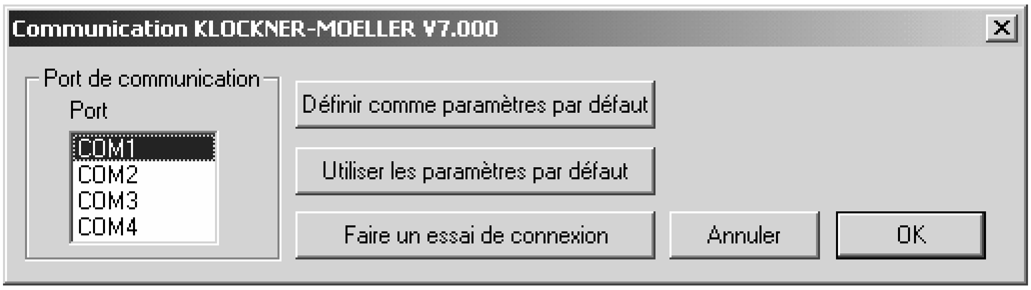


Setting communication module parameters

Post-processor PS4

This post-processor is used to program MOELLER PS4-200, PS4-300 and PS416 processors. MOELLER SUCOSOFT S40 V5 or higher software must be used (the demo version of this software can be used).

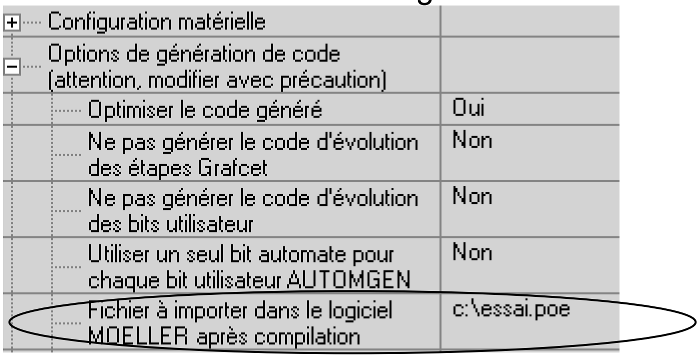
Module de communication


Communication KLOCKNER-MOELLER V7.000
Port de communication
Port
COM1
COM2
COM3
COM4
Définir comme paramètres par défaut
Utiliser les paramètres par défaut
Faire un essai de connexion
Annuler
OK

Setting communication module parameters

Transferring programs to MOELLER SUCOSOFT software

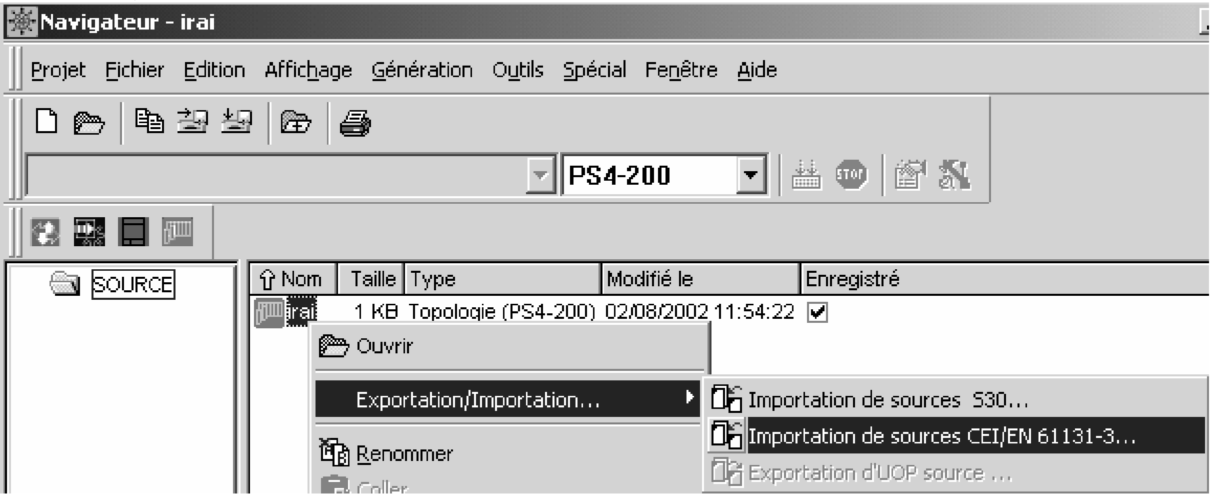
In the element below you create a file used for exchange between AUTOMGEN and SUCOSOFT. This file will be generated after compilation in AUTOMGEN.



Configuration matérielle	
Options de génération de code (attention, modifier avec précaution)	
Optimiser le code généré	Oui
Ne pas générer le code d'évolution des étapes Grafcet	Non
Ne pas générer le code d'évolution des bits utilisateur	Non
Utiliser un seul bit automate pour chaque bit utilisateur AUTOMGEN	Non
Fichier à importer dans le logiciel MOELLER après compilation	c:\essai.poe

Proceed as follows to import the file generated by AUTOMGEN in the MOELLER software then inject it in the processor.

- launch SUCOSOFT,
- create a new project,
- with the right side of the mouse click on the topology configuration in SUCOSOFT and select the option « Export/Import / Import IEC/EN 61131-3 sources»,



- enter the name of the file exported by AUTOMGEN,

- double click on « _MAIN » which appears on the list,

- in the « Special », menu select « Code generation »,

- answer « Yes » to :

- then « OK » at :

- in the« Generation » menu, select « Program parameter settings … »,

- Select an appropriate size to store all the internal variables (%M) of your application,

- In the «Generation » menu, select « Generate program code »,

- If there are no compilation errors, you can transfer the application to the processor. In the « Tools » menu, select « Test and start »,

- In the « Device » menu, select « Transfer/File Manager »,

- Click on transfer,

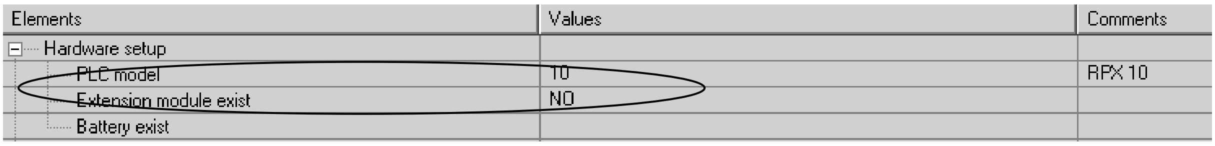
- At the end of the transfer disconnect from the SUCOSOFT software to be able to connect AUTOMGEN to the processor and activate the dynamic tuning mode

Post-processor RPX

This post-processor is used to program CROUZET RPX processors. .

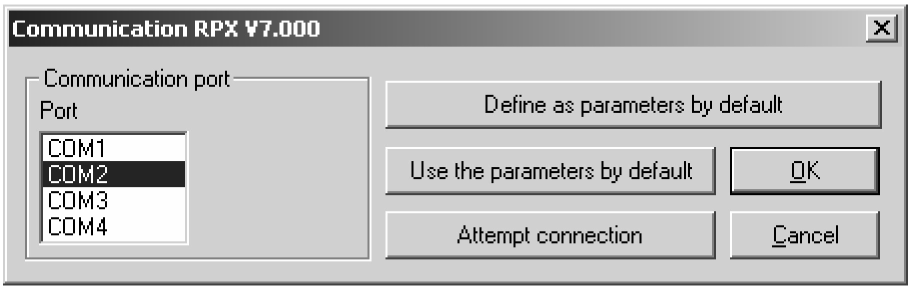
Selecting processor type

Use the « Configuration / Post-processor / RPX / System / Hardware configuration» browser element to select the type of processor.



Elements	Values	Comments
⊟ Hardware setup		
PLC model	10	RPX 10
Extension module exist	NO	
Battery exist		

Communication module



Communication RPX V7.000

Communication port

Port: COM2

COM1
COM2
COM3
COM4
Define as parameters by default
Use the parameters by default
OK
Attempt connection
Cancel

Setting communication module parameters

Utility

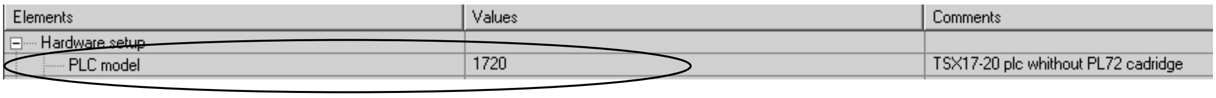
The « Configuration / Post-processor / RPX / Terminal emulator » browser element is used to access a terminal emulator for configuring processor communication couplers.

Post-processor PL71

This post-processor is used to program the processors SCHNEIDER TSX 17-10 and TSK 17-20 (without PL72 cartridge).

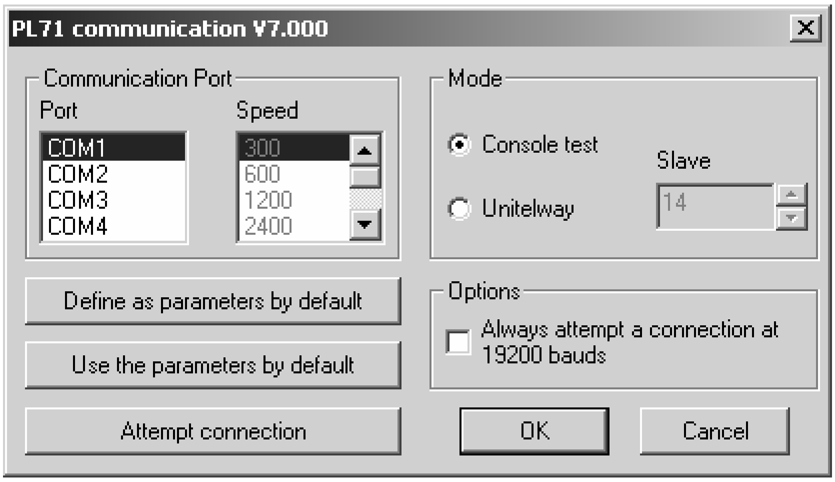
Selecting processor type

Use the « Configuration / Post-processor / PL71 / System / Hardware configuration» browser element to select the type of processor.



Elements	Values	Comments
⊟ Hardware setup		
PLC model	1720	TSX17-20 plc whithout PL72 cadridge

Communication module



PL71 communication V7.000

Communication Port

Port	Speed
COM1	300
COM2	600
COM3	1200
COM4	2400

Mode

● Console test

○ Unitelway

Slave: 14

Options

□ Always attempt a connection at 19200 bauds

Define as parameters by default
Use the parameters by default
Attempt connection
OK
Cancel

Setting communication module parameters

(For more information, please See the configuration of the Communication module PL72)

Fast counter task

A task type sheet bearing the number 1 will be associated to the processor fast counter task.

Specific examples

These examples are in the directory « <AUTOMGEN installation directory> /Examples/Post-processors/PL71 ».The files have the same names as the titles of the following chapters.

Counting

Since increments and decrements of PL71 counters are limited (only on the rising edge) in relation to AUTOMGEN and TSX processor possibilities it is necessary to use machine language code if you want to use them (see the example contents).

Fast counter

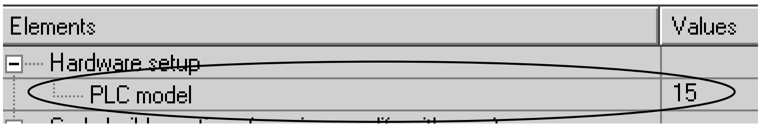
The goal is to count 200 pulses on the fast counter. Output O5 will be activated by a interrupt task at the end of the count.

Post-processor PB

This post-processor is used to program SCHNEIDER APRIL PB processors .

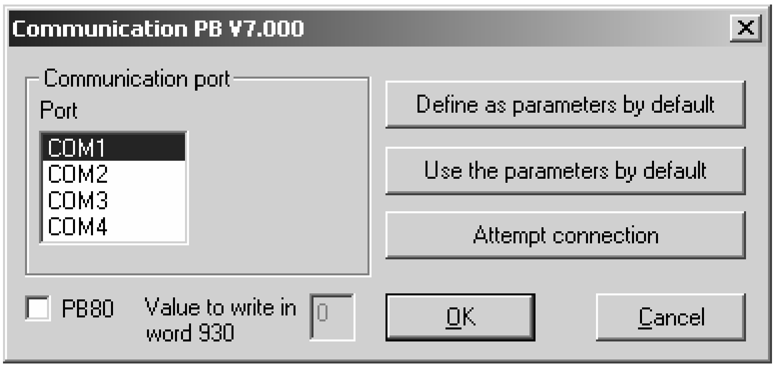
Selecting processor type

Use the « Configuration / Post-processor / PB / System / Hardware configuration» browser element to select the type of processor.



Elements	Values
Hardware setup	
PLC model	15

Communication module



□ PB80 Value to write in word 930: 0

Setting communication module parameters

Specific syntaxes

The command « $ORG=xxxx » is used to set the beginning of the assembly address, the starting assembly address is set at 0C30;

Example:
$ORG=1C30

The command « $TOP=xxx » sets the maximum address for the page jump. It sets the three digits of address lower weight, below these a page jump is automatically generated by the assembler.

The command « $CONST=xxxx,yyyy » sets the begin and end address for constant storage. The constants are stored in a table outside the program.

The command « WORD xxxx » enters the value xxxx (four hexadecimal digits) in the program.

The command « ADR xxxx » enters the address of variable xxxx (four hexadecimal digits) in the program.

The syntax #nnnn is used to refer to a constant value.

For example:

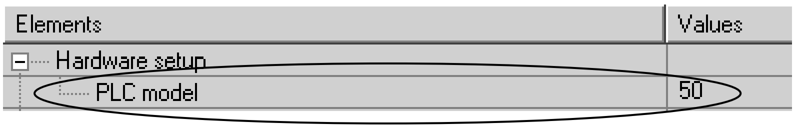
```
apl #1234     ; puts constant 1234 (hexadecimal) in the accumulator.
```

Post-processor SMC

This post-processor is used to program SCHNEIDER APRIL SMC processors .

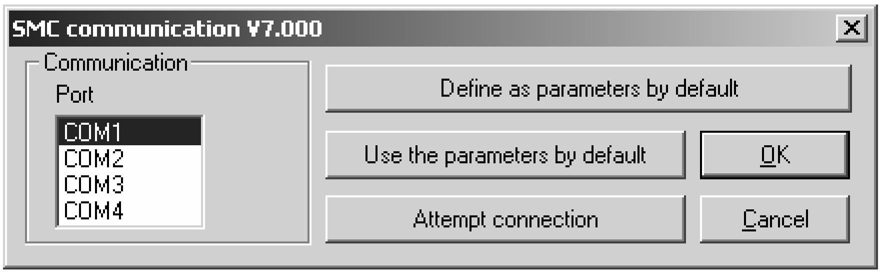
Selecting processor type

Use the « Configuration / Post-processor / SMC/ System / Hardware configuration» browser element to select the type of processor.



Elements	Values
⊟ Hardware setup	
PLC model	50

Communication module



SMC communication V7.000

Communication
Port
COM1
COM2
COM3
COM4
Define as parameters by default
Use the parameters by default
OK
Attempt connection
Cancel

Setting communication module parameters

Specific syntaxes

The command « $SEQ » indicates the beginning of a boolean area.

The command « $CAL » starts a calculation area.

The command « $PRD » starts a variable presetting area.

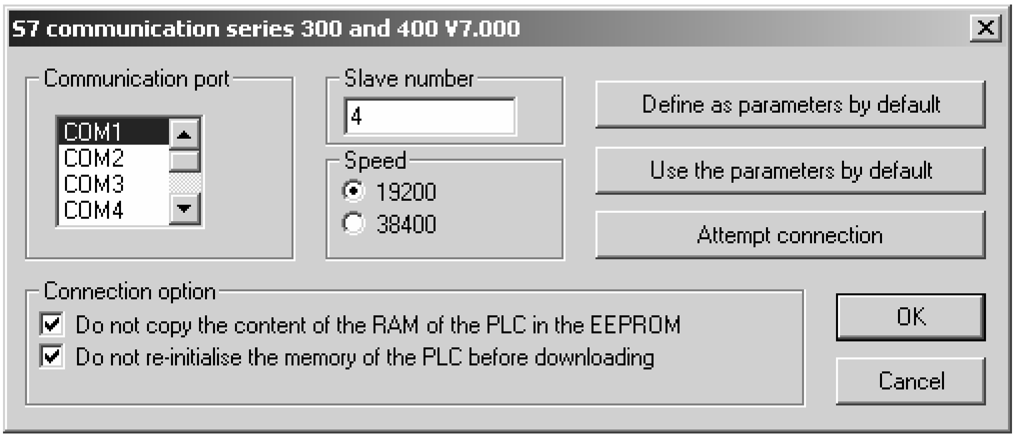
Boolean variables can be used which are bistable or monostable regardless of the SMC language conventions. The character « ! » before the sign « = » sets the variable to bistable (set to one or reset), the character « ? » before the sign « = » sets the variable to monostable (assignment or complement assignment).

The syntax « SS.cccccccc » is used to write a security sequence (necessary on processors SMC 25 and 600), « cccccccc » represents a program name with a maximum of 8 characters.

Post-processor S7300

This post-processor is used to program SIEMENS S7300 processors .

Communication module



S7 communication series 300 and 400 V7.000

Communication port: COM1 (selected), COM2, COM3, COM4

Slave number: 4

Speed: ● 19200 ○ 38400

Connection option:

☑ Do not copy the content of the RAM of the PLC in the EEPROM

☑ Do not re-initialise the memory of the PLC before downloading

Define as parameters by default
Use the parameters by default
Attempt connection
OK
Cancel

Setting communication module parameters

The slave number must correspond with that set as a parameter for the processor.

Specific syntaxes

The command « $BLOCK <block type> <number>» is used to set the beginning of a program or data block.

The block type can be « OB », « FB », « FC », « SFC » « SFB » for the code or « DB » for the data. The block number is a value between 0 and 65535. The command «$ENDBLOCK » indicates the end of a block.

Example:

$BLOCK DB1
...
$ENDBLOCK
$BLOCK OB1
...
$ENDBLOCK

The following syntax is used to state the variables for a block:

For an input variable:

$VAR-type nature {:=initialization}

or

$VAR-symbol nature : (type=initialization)

« nature » can be:

- « - «IN » for an input variable,
- « OUT » for an output variable,
- « - «INOUT » for an input/output variable,
- « TEMP » for a temporary variable;
- « STAT » for a static variable.

« type » can be one of the STEP 7 language variable types: BOOL, INT, WORD, STRUCT, ENDSTRUCT, etc …

« symbol » is used to associate a mnemonic with a variable.

« initialization » is optional and sets the default value of a variable.

The DB blocks only authorize static variables.

The OB blocks only authorize temporary variables.

The FC and SFC blocks do not authorize static variables.

As in SIEMENS software, variable statements must appear in the following order: input, output, input/output, static and temporary.

Setting block variables

The syntax « £D block statement » is used to set a statement associated to a particular block. When the block is generated by the compiler, then the statement is used.

Calling up blocks

The syntax « CALL block name {instance DB} (parameter list) » is used to call up an FC, FB, SFC or SFB block.

Example:

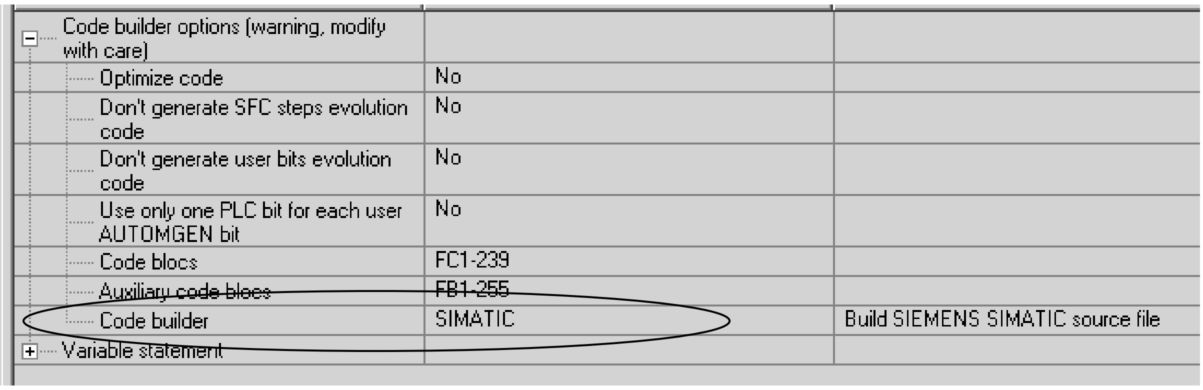
```
$BLOCK FC1
$VAR-IN input1 :BOOL :=FALSE ;
$VAR-IN input2 :BOOL :=FALSE ;
$VAR-OUT output :BOOL ;
u_input1_
u_input2_
=_output_
$ENDBLOCK

$BLOCK OB1
CALL FC1(_input1_ :=e0.0,_input2_ :=e0.1,_output_ :=a0.0)
$ENDBLOCK
```

Importing in SIEMENS SIMATIC software

Use the process below to import the code generated by AUTOMGEN in the SIEMENS SIMATIC software :

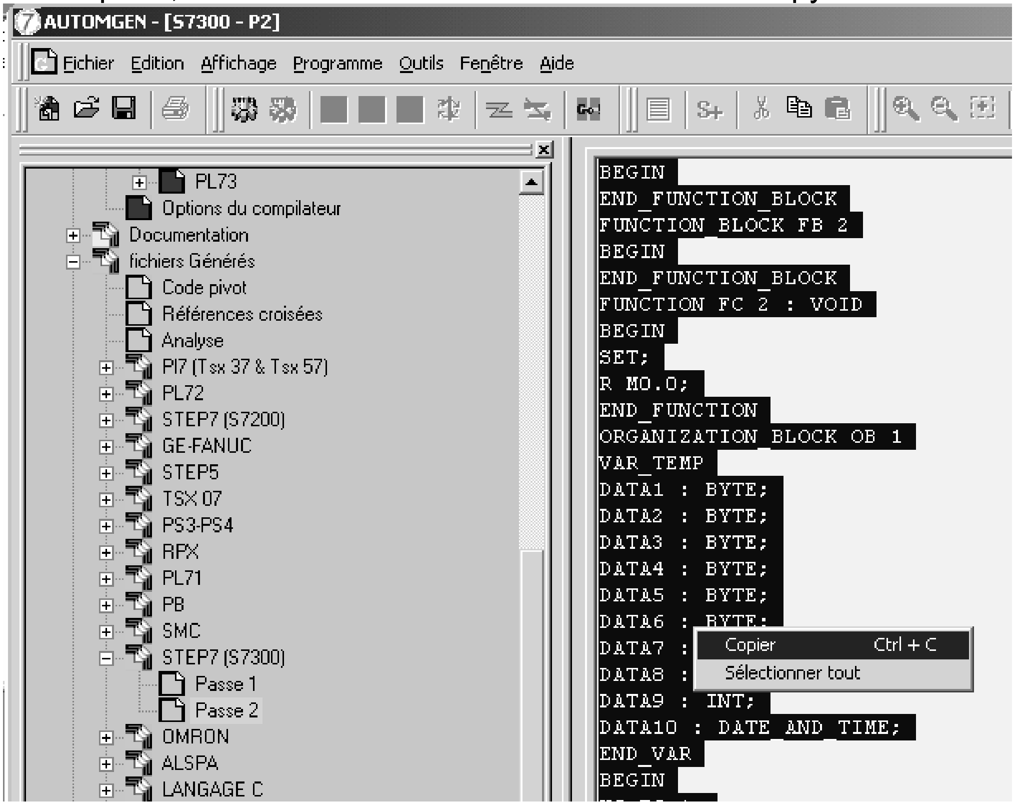
1- In the « System » part of the S7300 post-processor configuration, select SIMATIC in the following element



Code builder options (warning, modify with care)		
Optimize code	No	
Don't generate SFC steps evolution code	No	
Don't generate user bits evolution code	No	
Use only one PLC bit for each user AUTOMGEN bit	No	
Code blocs	FC1-239	
Auxiliary code blocs	FB1-255	
Code builder	SIMATIC	Build SIEMENS SIMATIC source file
Variable statement		

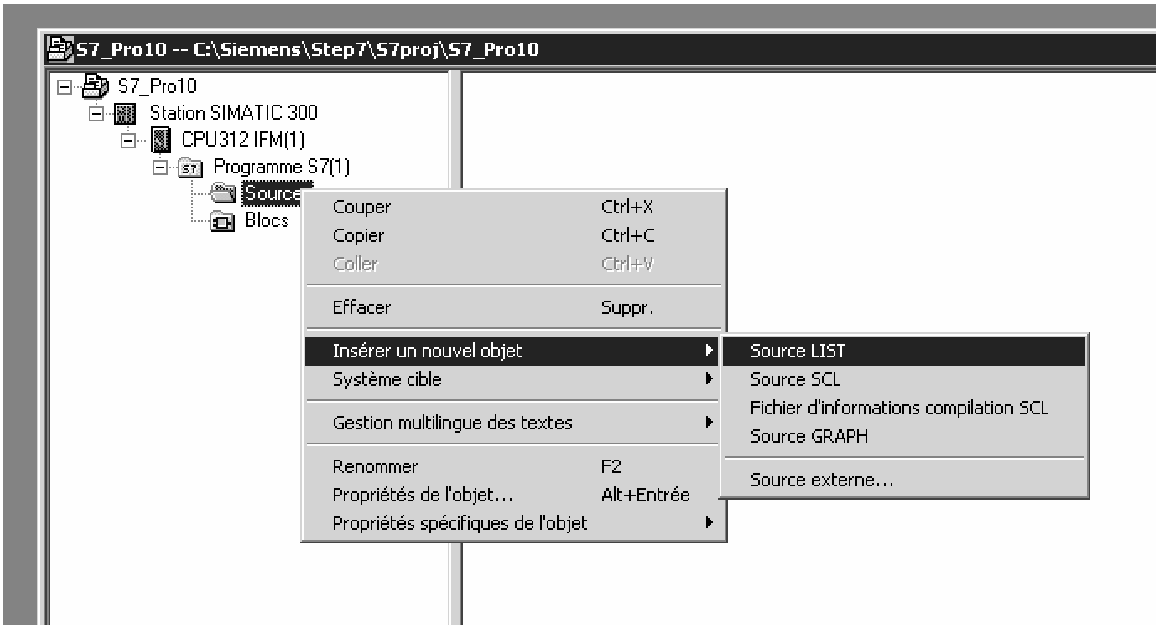
2- Compile the application,

3- In AUTOMGEN, open the element « Generated code / Post-processor S7300 / Step 2 », select all the lines then the command « Copy » in the « Edit » menu.

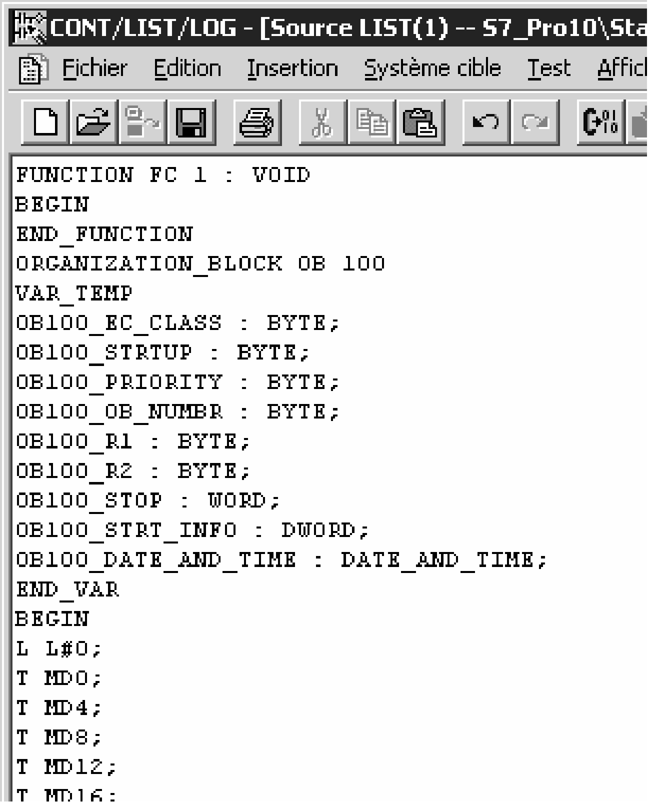


```
BEGIN
END_FUNCTION_BLOCK
FUNCTION_BLOCK FB 2
BEGIN
END_FUNCTION_BLOCK
FUNCTION FC 2 : VOID
BEGIN
SET;
R M0.0;
END_FUNCTION
ORGANIZATION_BLOCK OB 1
VAR_TEMP
DATA1 : BYTE;
DATA2 : BYTE;
DATA3 : BYTE;
DATA4 : BYTE;
DATA5 : BYTE;
DATA6 : BYTE;
DATA7 :
DATA8 :
DATA9 : INT;
DATA10 : DATE_AND_TIME;
END_VAR
BEGIN
```

4- In the SIMATIC software, create a « Source LIST » type element.



5- In SIMATIC, paste the code in the window containing the source LIST using the « Paste » command in the « Edit » menu,



```
FUNCTION FC 1 : VOID
BEGIN
END_FUNCTION
ORGANIZATION_BLOCK OB 100
VAR_TEMP
OB100_EC_CLASS : BYTE;
OB100_STRTUP : BYTE;
OB100_PRIORITY : BYTE;
OB100_OB_NUMBR : BYTE;
OB100_R1 : BYTE;
OB100_R2 : BYTE;
OB100_STOP : WORD;
OB100_STRT_INFO : DWORD;
OB100_DATE_AND_TIME : DATE_AND_TIME;
END_VAR
BEGIN
L L#0;
T MD0;
T MD4;
T MD8;
T MD12;
T MD16;
```

6- In SIMATIC, compile the source by clicking on.

At this point the importation is completed.

Structure of generated code

SIEMENS STEP7 language is organized in program and data blocks. AUTOMGEN applications translated by the STEP7 post-processor are broken down into multiple blocks. By default, the post-processor uses the following blocks:

- OB1 organization block: this block calls up all the blocks that must be cyclically processed,
- OB100 : blocks executed when the processor starts. This block sets a bit to activate the initial Grafcet steps.

The FB blocks are used to process pre-settings, to manage the evolution of boolean variables and time delays.

The FC blocks are used for the code issued by the application and for the begin and end machine code.

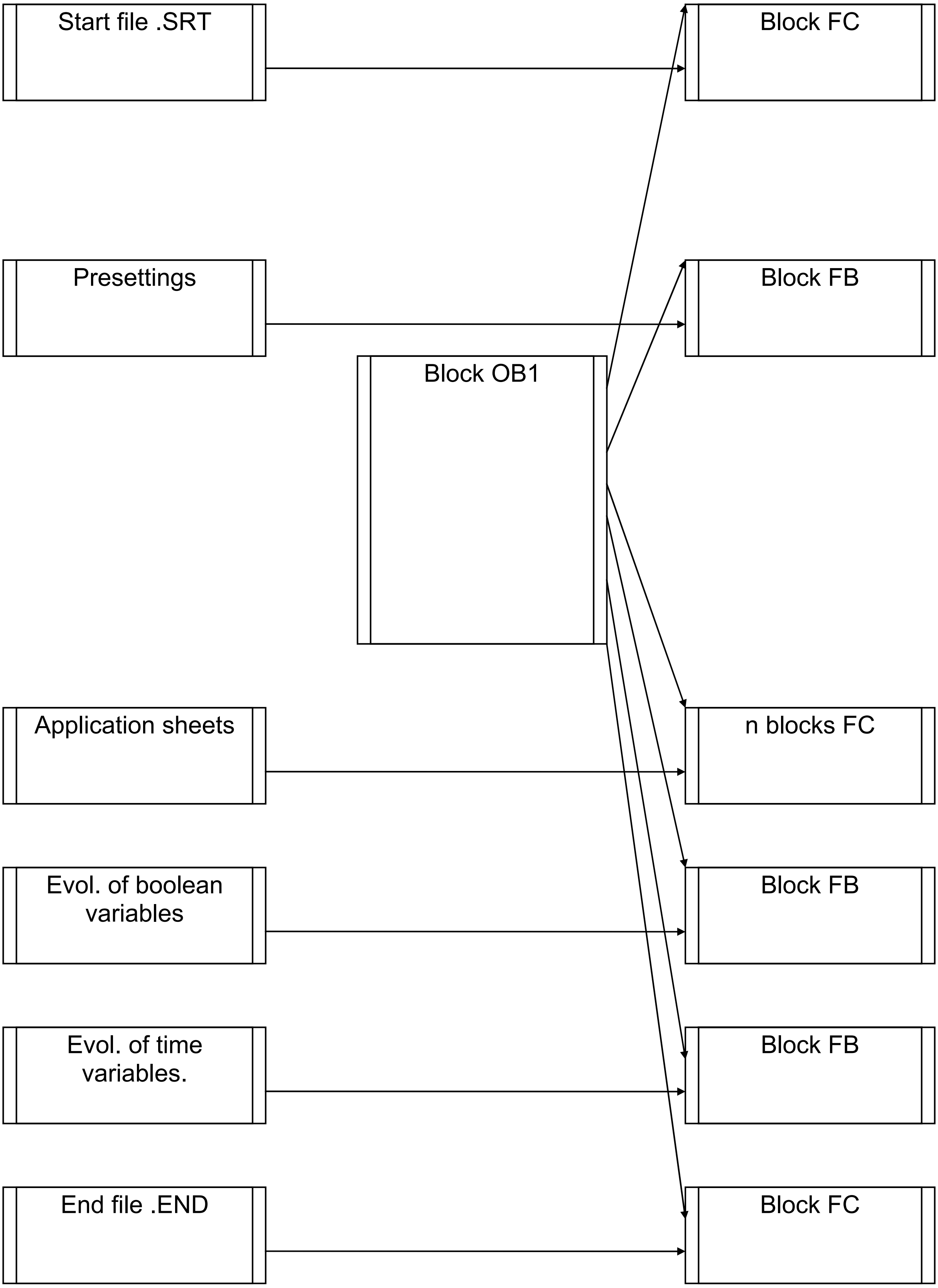
An FB block is created for each application sheet.

In addition, the sheets can be directly associated to a block (see chapterAssociating code written on a sheet to a program block)..

By default, the post-processor uses the blocks FB1 to FB255 and FC1 to FC239 when needed.

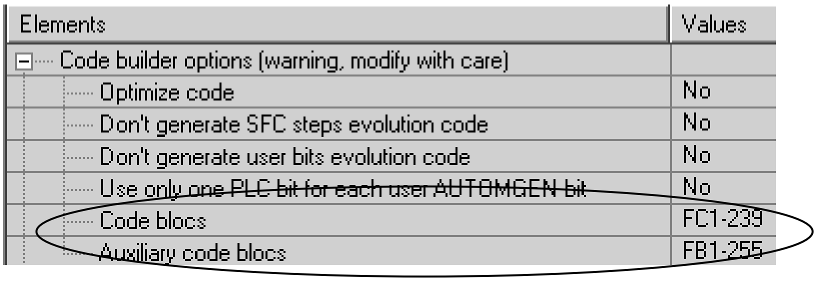
These values can be modified in the « System » element of the configuration.

The figure below illustrates the structure of code generated by the SIEMENS post-processor:



Selecting program blocks to use

Blocks FC1 to FC 239 and FB 1 to FB 255 are used by default. Two configuration elements are used to select other blocks.



Elements	Values
Code builder options (warning, modify with care)	
Optimize code	No
Don't generate SFC steps evolution code	No
Don't generate user bits evolution code	No
Use only one PLC bit for each user AUTOMGEN bit	No
Code blocs	FC1-239
Auxiliary code blocs	FB1-255

Associating code written on a sheet to a program block

When writing low level literal code or machine code on an organizational chart on a « Task » type sheet, associate the code to a STEP7 block.

The task number determines the block type and number.

The generated code for that sheet must take into account the block type and the instructions used in that block type (The set of instructions for blocks OB and PB is limited).

The table below provides the correspondence between the task number and the STEP7 block.

Task number	*STEP7 block*
0 to 255	OB 0 to OB 255
256 to 511	FC 0 to FC 255
512 to 767	FB 0 to FB 255

Specific examples

These examples are in the directory « <AUTOMGEN installation directory> /Examples/Post-processors/S7300 ».The files have the same names as the titles of the following chapters.

Calling up a STEP7 block

Example of calling up STEP7 function blocks

Using an OB block

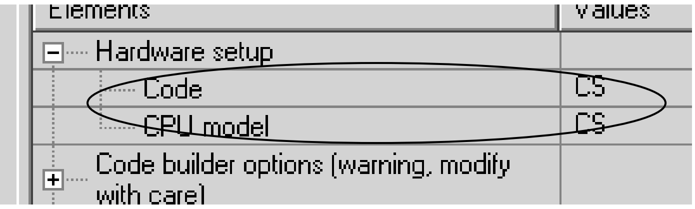
Example of an association of code written on a sheet to an OB block.

Post-processor OMRON

This post-processor is used to program OMRON processors .

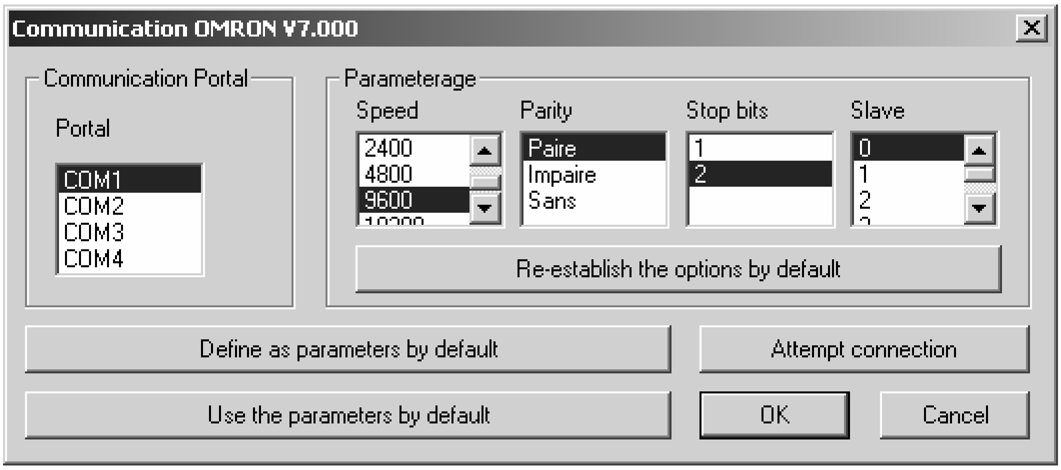
Select PLC model

Tht item « Configuration / Post-processor / OMRON / System / Hardware configuration » must be used to select PLC model.



Elements	Values
Hardware setup	
Code	CS
CPU model	CS
Code builder options (warning, modify with care)	

Communication module



Communication OMRON V7.000

Communication Portal

Portal: COM1 (COM1, COM2, COM3, COM4)

Parameterage

Speed: 9600 (2400, 4800, 9600, …)

Parity: Paire (Paire, Impaire, Sans)

Stop bits: 2 (1, 2)

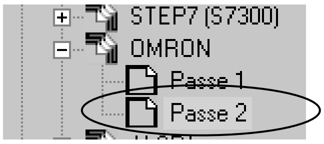
Slave: 0 (0, 1, 2, …)

Re-establish the options by default
Define as parameters by default
Attempt connection
Use the parameters by default
OK
Cancel

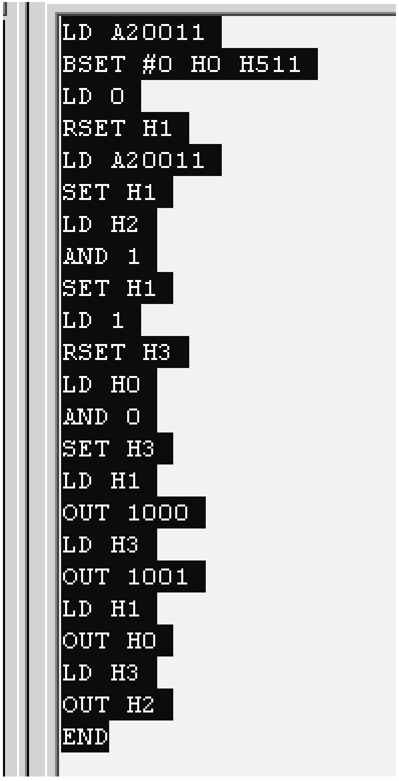
Setting communication module parameters

Transferring applications to the CX-PROGRAMMER software

- at the end of compiling in AUTOMGEN, double click on the « Generated files / OMRON / step 2 » element,

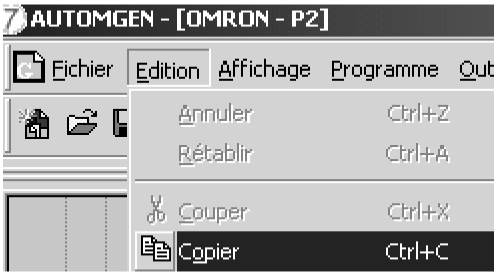


- select all the lines,

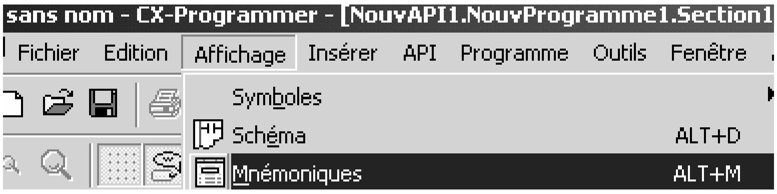


```
LD A20011
BSET #0 H0 H511
LD 0
RSET H1
LD A20011
SET H1
LD H2
AND 1
SET H1
LD 1
RSET H3
LD H0
AND 0
SET H3
LD H1
OUT 1000
LD H3
OUT 1001
LD H1
OUT H0
LD H3
OUT H2
END
```

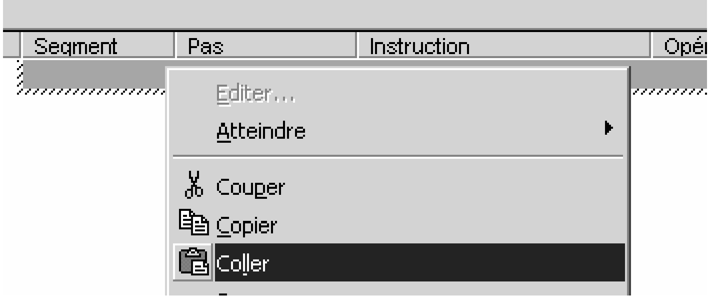
- select the « Copy » command in the « Edit » menu

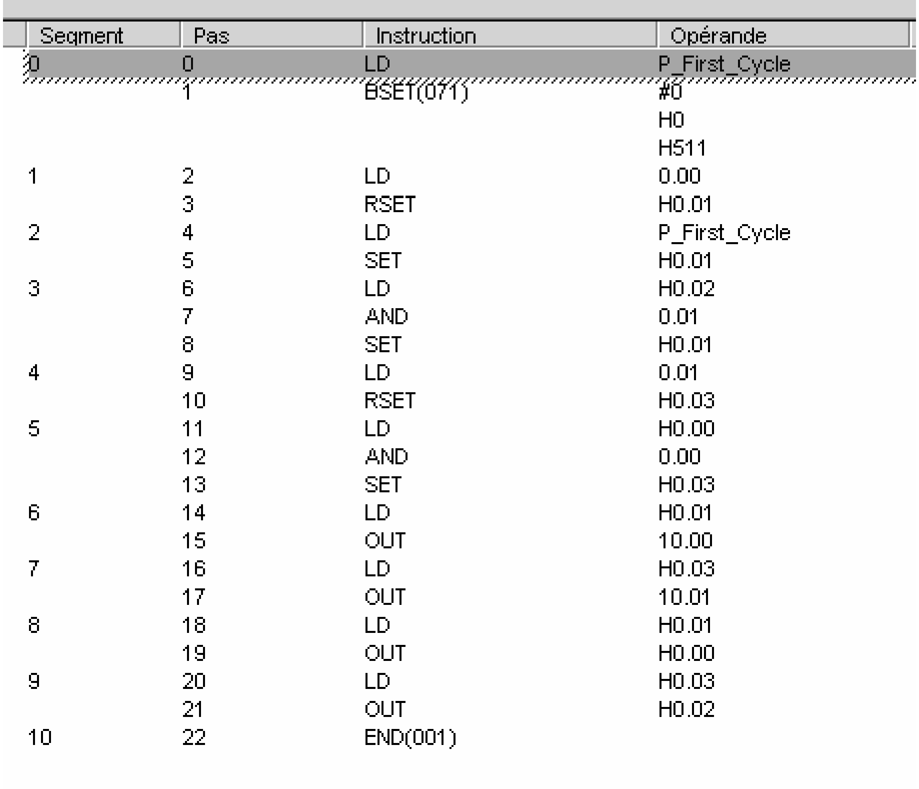


- in CX-PROGRAMMER, create a new application, display the program area in mnemonics format,



- select the program area and then paste the lines,





Segment	Pas	Instruction	Opérande
0	0	LD	P_First_Cycle
	1	BSET(071)	#0
			H0
			H511
1	2	LD	0.00
	3	RSET	H0.01
2	4	LD	P_First_Cycle
	5	SET	H0.01
3	6	LD	H0.02
	7	AND	0.01
	8	SET	H0.01
4	9	LD	0.01
	10	RSET	H0.03
5	11	LD	H0.00
	12	AND	0.00
	13	SET	H0.03
6	14	LD	H0.01
	15	OUT	10.00
7	16	LD	H0.03
	17	OUT	10.01
8	18	LD	H0.01
	19	OUT	H0.00
9	20	LD	H0.03
	21	OUT	H0.02
10	22	END(001)	

You can upload the application in the processor starting from CX-PROGRAMMER then return to AUTOMGEN to tune the program in connected mode (remember to disconnect CX-PROGRAMMER from the processor to be able to communicate from AUTOMGEN).

Specific syntax

The following syntax is used to set the value of a data word.

$DMn=value
« n » is the word number;
« value » is a 16 bit value expressed by default in decimal or hexadecimal if preceded by the character 'H'.

Example:

```
$DM10=50
$DM100=HA000
```

Associating code written on a sheet to a program block

When writing low level literal code or machine code on an organizational chart on a « Task » type sheet, associate the code to a interrupt task. The task number is equivalent to the stop number.

Specific example

This example is in the directory « <AUTOMGEN installation directory> /Examples/Post-processors/S7200 ».The file has the same name as the title of the following chapter.

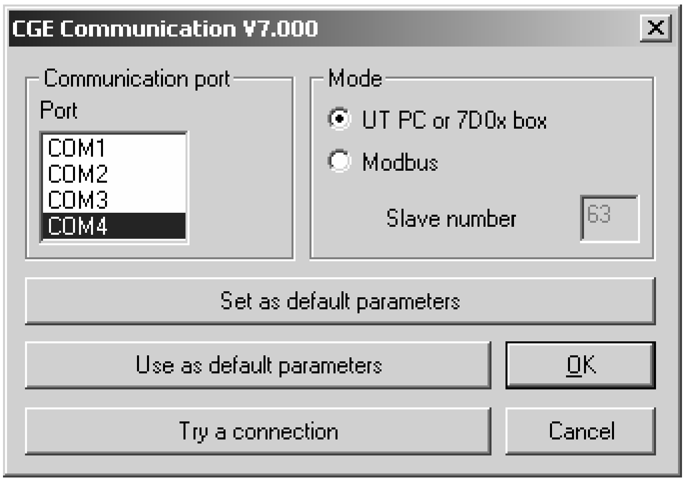
Interrupt task

Example of calling up a interrupt task

Post-processor ALSPA

This post-processor is used to program CEGELEC ALSPA C50 and C100 processors.

Communication module



CGE Communication V7.000

Communication port

Port

- COM1
- COM2
- COM3
- COM4 (selected)

Mode

● UT PC or 7D0x box

○ Modbus

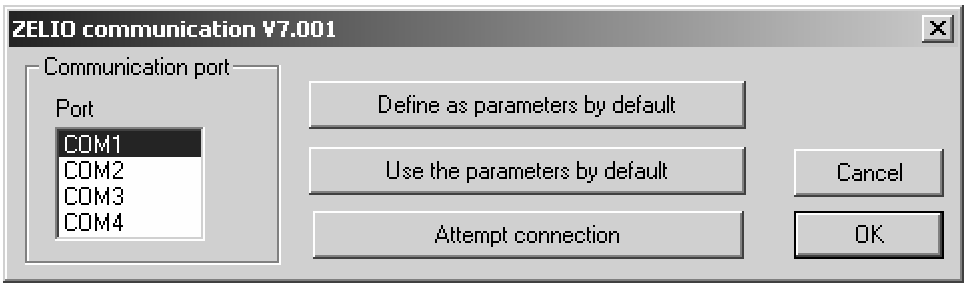
Slave number: 63

Setting communication module parameters

Post-processor ZELIO

This post-processor is used to program SECHNEIDER ZELIO modules.

Communication module

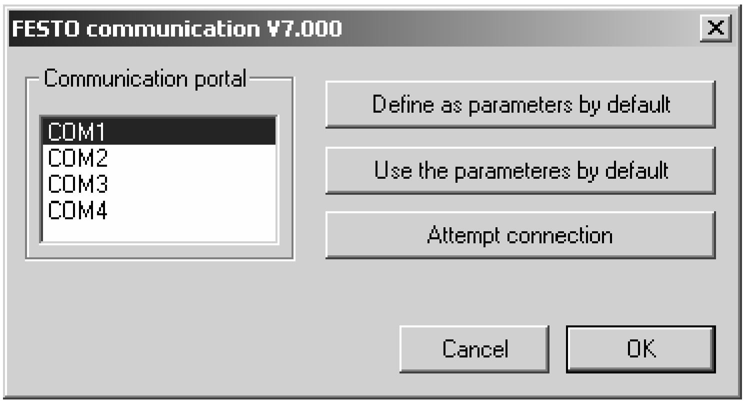

ZELIO communication V7.001
Communication port
Port
COM1
COM2
COM3
COM4
Define as parameters by default
Use the parameters by default
Attempt connection
Cancel
OK

Setting communication module parameters

Post-processor FESTO

This post-processor is used to program FPC 101, FPC 103 and FEC FESTO processors .

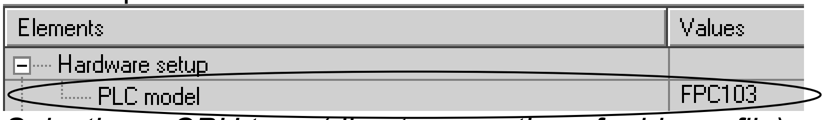
Communication module


FESTO communication V7.000
Communication portal
COM1
COM2
COM3
COM4
Define as parameters by default
Use the parameteres by default
Attempt connection
Cancel
OK

Setting communication module parameters

Generating a binary file

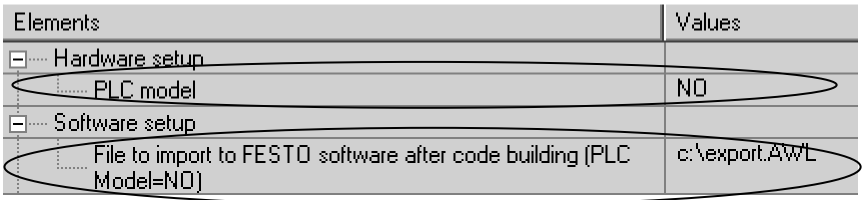
Binary files can be directly generated and downloaded in FPC 101 and FPC 103 processors. The FST FESTO software workgroup in DOS or WINDOWS is necessary for FEC processors.



Elements	Values
⊟ Hardware setup	
PLC model	FPC103

Selecting a CPU type (direct generation of a binary file)

Importation in a FESTO software workgroup



Elements	Values
⊟ Hardware setup	
PLC model	NO
⊟ Software setup	
File to import to FESTO software after code building (PLC Model=NO)	c:\export.AWL

Generation of an AWL file compatible with FESTO workgroups

If you use the FST FESTO software workgroup in DOS, create the AWL file in this software.

If you use the FST FESTO software in WINDOWS, open the generated file in AUTOMGEN by double clicking on « Generated files / FESTO / Pass 2 », select all the files, use the command « Copy » in the « Edit » menu and then the command « Paste » in the FESTO software to retrieve the generated code.

Transfer the program to the processor with the FESTO software.

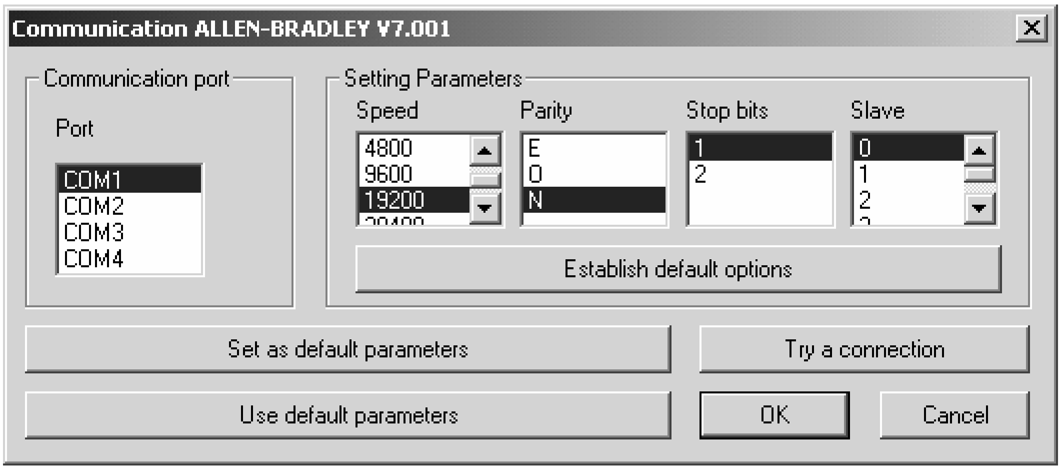
Then you can connect to AUTOMGEN (after you have disconnected the FESTO software) by using the « Only connect ».connection mode.

Post-processor ALLEN-BRADLEY

This post-processor is used to program ROCKWELL SLC processors. ROCKWELL RSLogix 500 V5 software or higher is required.

The RSLogix 500 STARTER version cannot be used to import files generated by AUTOMGEN.

Communication module



Communication ALLEN-BRADLEY V7.001

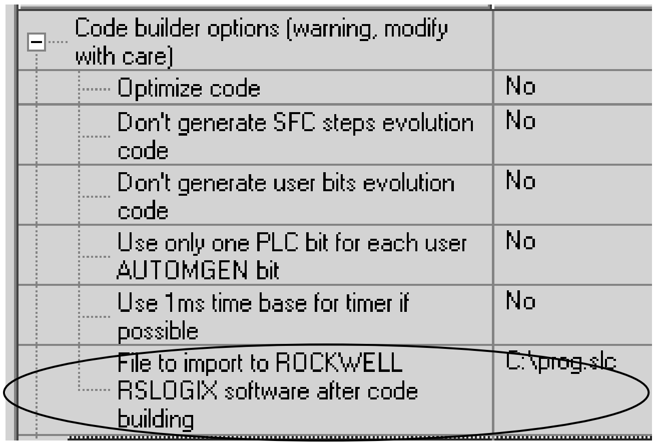
Communication port: Port: COM1 (selected), COM2, COM3, COM4

Setting Parameters:
- Speed: 4800, 9600, 19200 (selected), 38400
- Parity: E, O, N (selected)
- Stop bits: 1 (selected), 2
- Slave: 0 (selected), 1, 2, 3

Setting communication module parameters

Transferring programs to ROCKWELL RS-Logix 500 Software

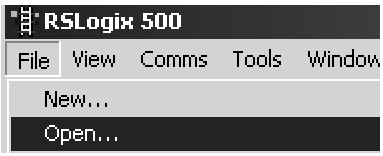
In the element below you create a file used for exchange between AUTOMGEN and RSLogix 500. This file will be generated after compilation in AUTOMGEN

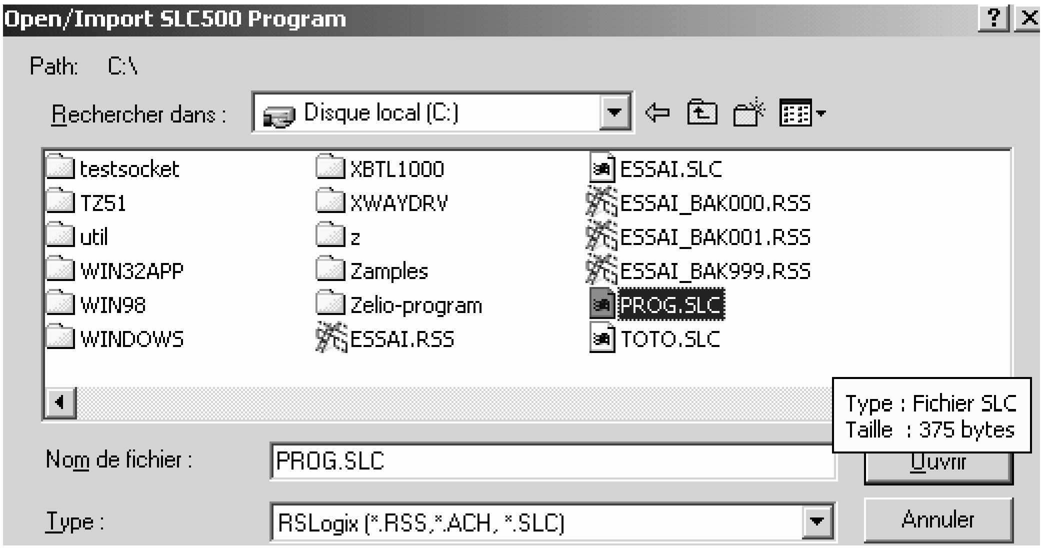


Code builder options (warning, modify with care)	
Optimize code	No
Don't generate SFC steps evolution code	No
Don't generate user bits evolution code	No
Use only one PLC bit for each user AUTOMGEN bit	No
Use 1ms time base for timer if possible	No
File to import to ROCKWELL RSLOGIX software after code building	C:\prog.slc

Generating an .SLC file compatible with RSLogix 500

Launch RSLogix500, then open the .SLC file generated by AUTOMGEN.





Path: C:\

Rechercher dans : Disque local (C:)

Type : Fichier SLC
Taille : 375 bytes

Nom de fichier : PROG.SLC

Type : RSLogix (*.RSS,*.ACH, *.SLC)

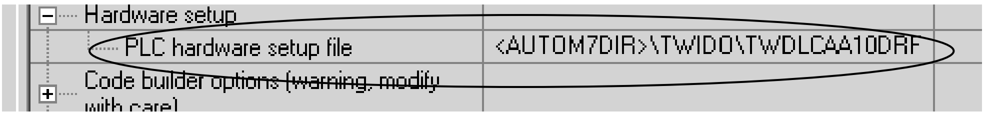
Transfer the program to the processor using the RSLogix 500 software. After disconnecting the RSLogix 500 software from the processor, you can tune it from AUTOMGEN in connected mode.

Post-processor TWIDO

This post-processor is used to program SCHNEIDER TWIDO processors.

Processor configuration selection

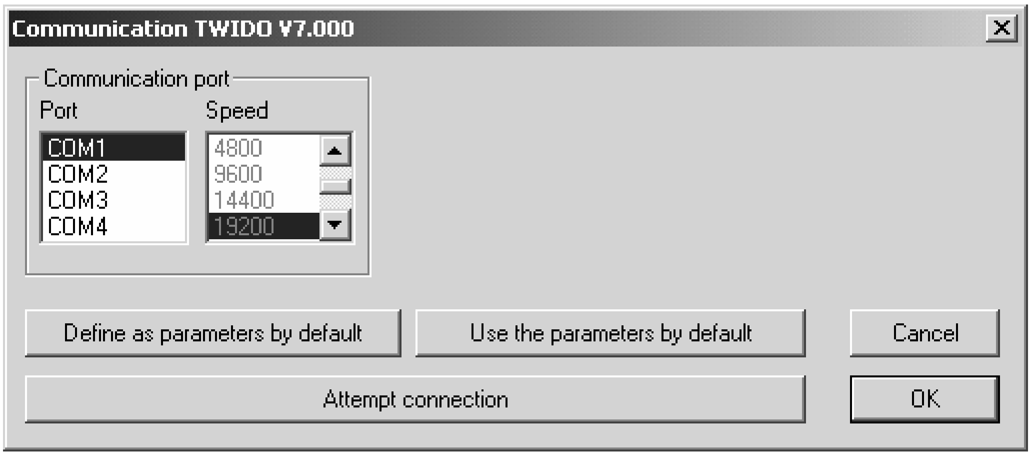
The « Configuration / Post-processor / TWIDO / System / Hardware configuration » element on the browser is used to select the type of processor.



Hardware setup	
PLC hardware setup file	<AUTOM7DIR>\TWIDO\TWDLCAA10DRF
Code builder options (warning, modify with care)	

The « .TWD » file is a configuration file which must be generated with the SCHNEIDER TWIDOSOFT software workgroup. The sub-directory « TWIDO » in the AUTOMGEN installation directory contains the configuration files for various types of TWIDO processors.

Communication module



Communication TWIDO V7.000

Communication port

Port: COM1

Speed: 19200

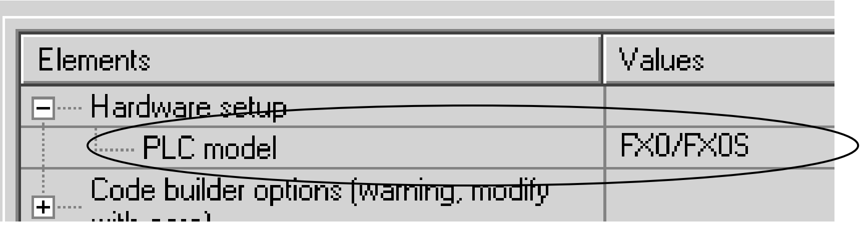
COM1
COM2
COM3
COM4
4800
9600
14400
19200
Define as parameters by default
Use the parameters by default
Cancel
Attempt connection
OK

Post-processor MITSUBISHI

This post-processor is used to program FX series MITSUBISHI processors. The code generated by AUTOMGEN can be directly sent to FX series MITSUBISHI processors or imported in MITSUBISHI FX-WIN or GX-DEVELOPPER software.

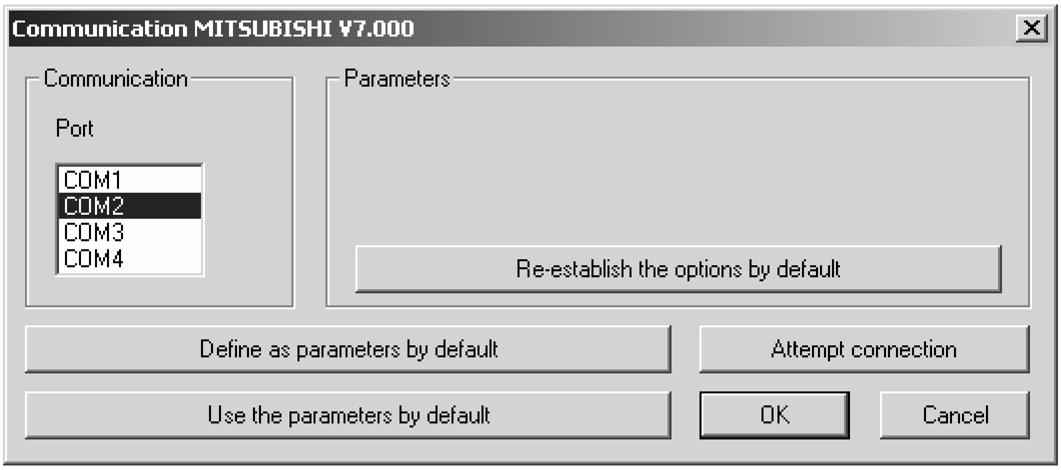
Selecting the type of processor

« Configuration / Post-processor / MITSUBISHI / System / Hardware configuration» on the browser is used to select the type of PLC.



Elements	Values
⊟ Hardware setup	
PLC model	FX0/FX0S
⊞ Code builder options (warning, modify	

Communication module



Communication MITSUBISHI V7.000

Communication

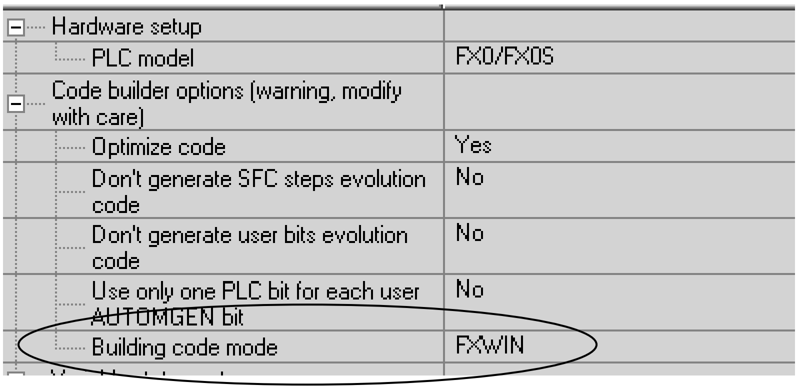
Port: COM2

COM1
COM2
COM3
COM4
Parameters
Re-establish the options by default
Define as parameters by default
Attempt connection
Use the parameters by default
OK
Cancel

Setting the communication module

Transferring programs to MITSUBISHI FX-WIN software

Select FXWIN in the element below.



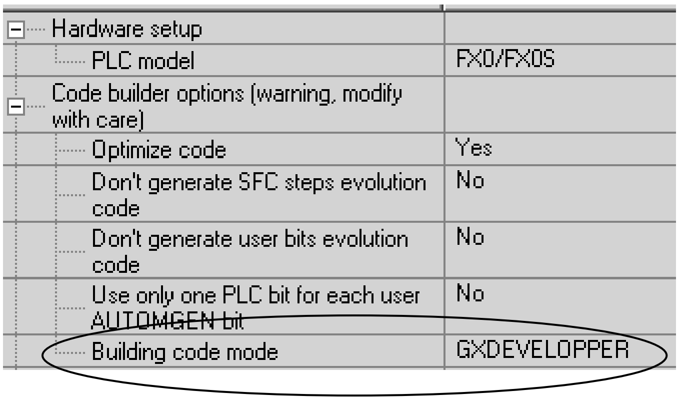
⊟ Hardware setup	
PLC model	FX0/FX0S
⊟ Code builder options (warning, modify with care)	
Optimize code	Yes
Don't generate SFC steps evolution code	No
Don't generate user bits evolution code	No
Use only one PLC bit for each user AUTOMGEN bit	No
Building code mode	FXWIN

The English version of FXWIN must be used. Importation has been validated for the 3.20 version of FX-WIN.

After compilation in AUTOMGEN, create a project in FX-WIN. Open the program in editing, instruction list and select « Enter » in the « Edit » menu.

Transferring programs to MITSUBISHI GX-DEVELOPPER software

Select GXDEVELOPPER in the element below.



⊟ Hardware setup	
PLC model	FX0/FX0S
⊟ Code builder options (warning, modify with care)	
Optimize code	Yes
Don't generate SFC steps evolution code	No
Don't generate user bits evolution code	No
Use only one PLC bit for each user AUTOMGEN bit	No
Building code mode	GXDEVELOPPER

Launch the executable « A7TOGX.EXE » in the AUTOMGEN installation directory. After launching it, an icon « A7 → GX » appears in the WINDOWS icon bar.

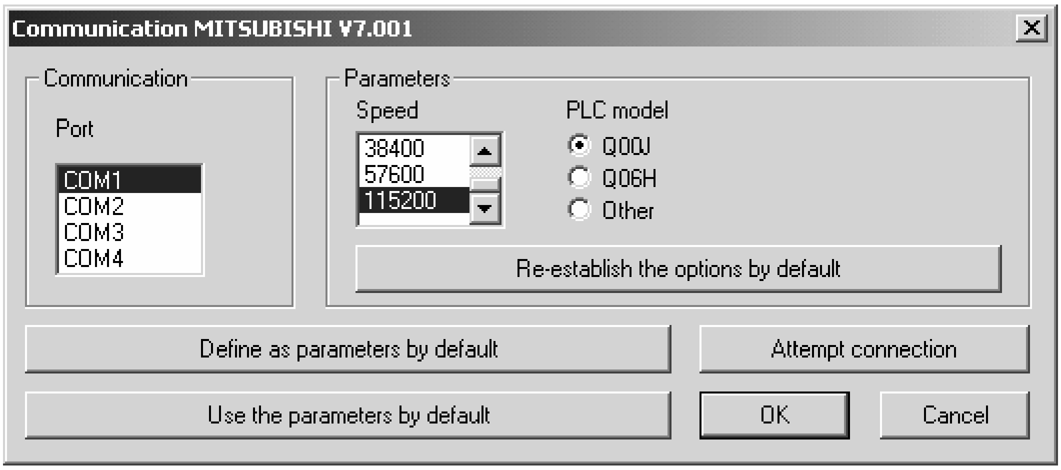
After compiling AUTOMGEN, create a project in GX-DEVELOPPER. Open the program in editing, instruction list. Press the two SHIFT keys or the LEFT SHIFT key and the F11 key at the same time for one second. The program will be transferred to GX-DEVELOPPER.

To uninstall « A7TOGX » click on the icon with the right side of the mouse. You can leave « A7TOGX » installed as long as it is necessary, so you do not need to install and uninstall it every time you want to import an application in GX-DEVELOPPER.

Post-processor MITSUBISHI-Q

This post-processor is used to program Q series MITSUBISHI processors. The code generated by AUTOMGEN must be imported in MITSUBISHI GX-DEVELOPPER software.

Communication module



Communication MITSUBISHI V7.001

Communication — Port: COM1 (selected), COM2, COM3, COM4

Parameters — Speed: 38400, 57600, 115200 (selected)

PLC model: ● Q00J ○ Q06H ○ Other

Re-establish the options by default
Define as parameters by default
Attempt connection
Use the parameters by default
OK
Cancel

Setting the communication module

Transferring programs to MITSUBISHI GX-DEVELOPPER software

Launch the executable « A7TOGX.EXE » in the AUTOMGEN installation directory. After launching it, an icon « A7 → GX » appears in the WINDOWS icon bar.

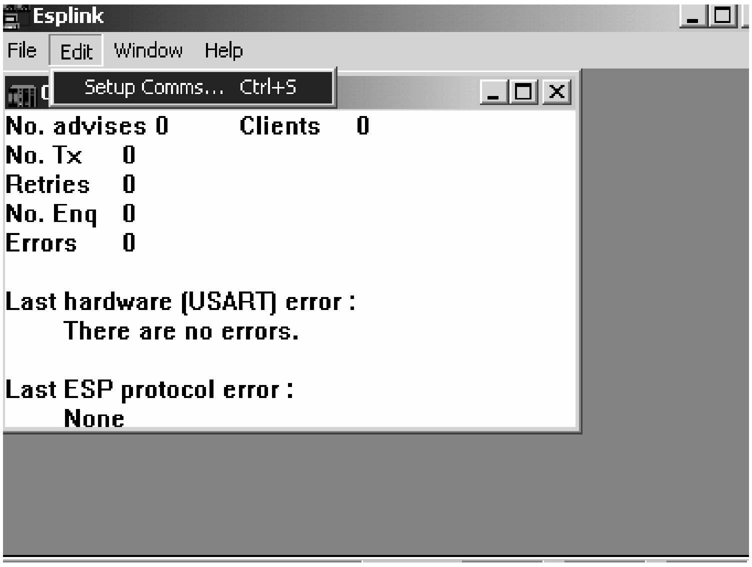
After compiling AUTOMGEN, create a project in GX-DEVELOPPER. Open the program in editing, instruction list. Press the two SHIFT keys or the LEFT SHIFT key and the F11 key at the same time for one second. The program will be transferred to GX-DEVELOPPER.

To uninstall « A7TOGX » click on the icon with the right side of the mouse. You can leave « A7TOGX » installed as long as it is necessary, so you do not need to install and uninstall it every time you want to import an application in GX-DEVELOPPER.

Post-processor GEM

This post-processor is used to program GEM PLCs.

Communication module



```
No. advises 0      Clients    0
No. Tx      0
Retries     0
No. Enq     0
Errors      0

Last hardware (USART) error :
     There are no errors.

Last ESP protocol error :
     None
```

Setting the communication module

For setting the communication module parameters, launch the ESPLINK.EXE file into the AUTOMGEN installation directory and select the “Edit / Setup Comms ...” menu.