# TpaCAD

2.2.0

# *Configuring TpaCAD*

Tecnologie e Prodotti per l'Automazione

# Table of Contents

# 1 How to configure TpaCAD

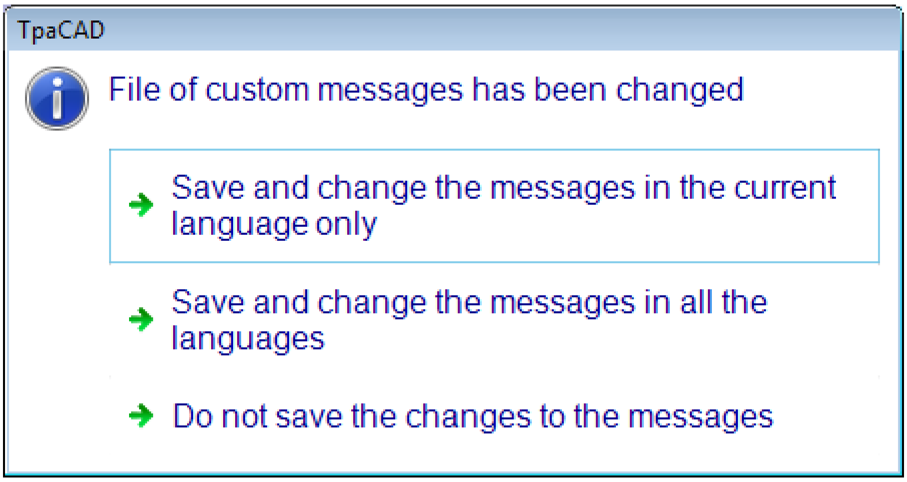
This document describes how to configure TpaCAD. It provides technical support to the product and completes other support tools such as the customer information. Some parts of the document go into the specific configuration functionalities, while other ones concern more general subjects. Tpa **reserves the right to modify and/or to integer** the TpaCAD configuration modes, while respecting the usage functionalities and product compatibility.

The **Configure TpaCAD** command is selected from the menu ⚙ with closed program and from the Maintenance or Manufacturer level. The command is not available in Demo functionalities or if the instance of TpaCAD has not been executed first.

At Maintenance level the user can modify the assignment of the only pages that define the composition of the Menu and of the Palette of workings.

The composition of the pages shown for the configuration of TpaCAD suits the key used, like also the configuration that is saved at the end, when the **Configure TpaCAD** command is confirmed. More specifically: the enabling processes, available in Professional features only, are reset, if the program recognizes that a Basic feature is functioning.

Under the assignments available in configuration command there are the modification and the entering of the messages which can be customized: these can be stored in the file CADAUX.XMLNG –, which will be considered in a later paragraph.
On his turn the file CADAUX.XMLNG can be split in two rows, according to the translations into the language of the available specific application. If some messages are modified while closing and confirming the command **Configure TpaCAD** the window below appears with three ending mode possibilities:


TpaCAD

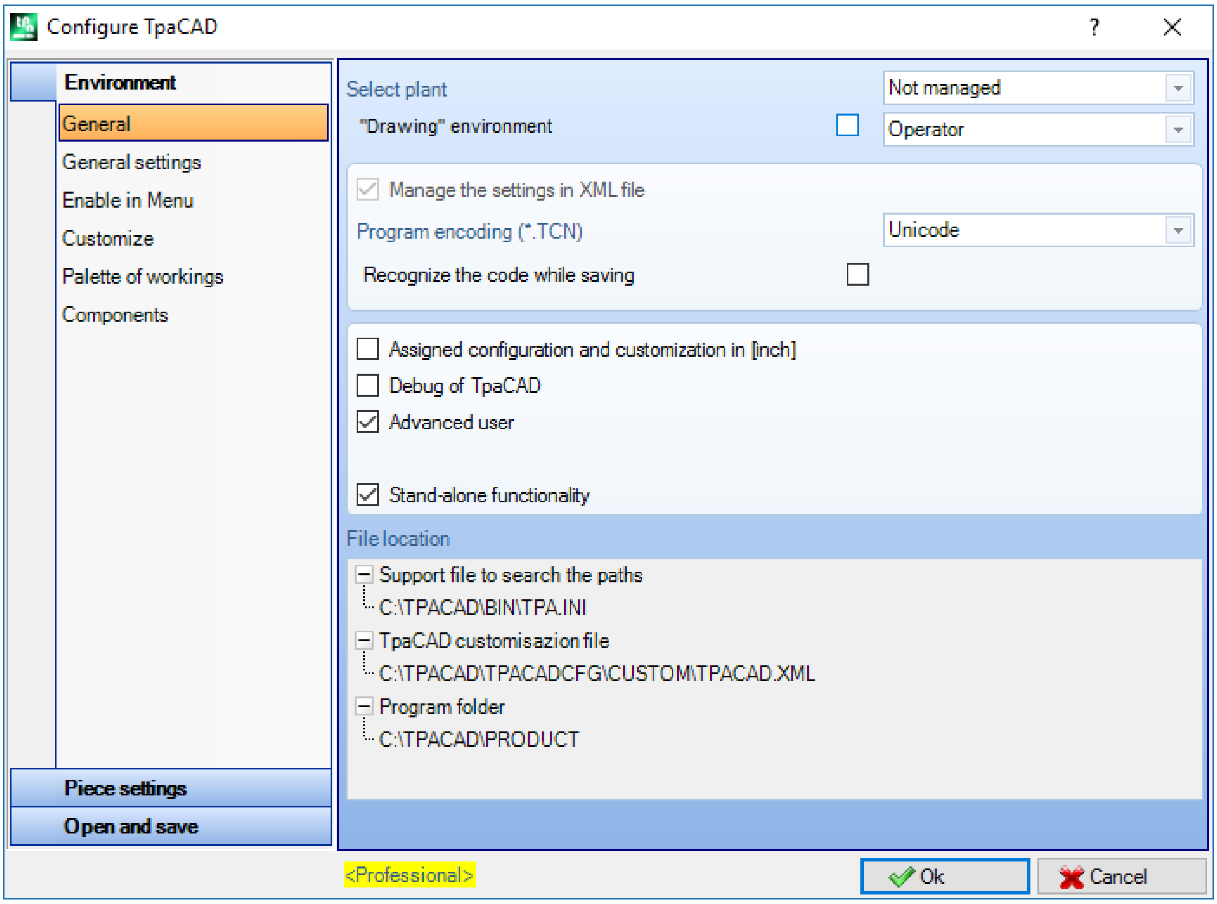
File of custom messages has been changed

➔ Save and change the messages in the current language only
➔ Save and change the messages in all the languages
➔ Do not save the changes to the messages

- **Save and change the messages in the current language only:** select to save the changes to the messages, but only in the section of the current functioning language.
- **Save and change the messages in all the languages:** select to save the changes to messages in all the languages in which the message is assigned. In all languages the messages shall be assigned equal.
- **Do not save the changes to the messages:** the command is closed without saving the changes.

# 1.1 Environment

## General



Configure TpaCAD

Environment
- General
- General settings
- Enable in Menu
- Customize
- Palette of workings
- Components

Piece settings

Open and save

Select plant: Not managed

"Drawing" environment □ Operator

☑ Manage the settings in XML file

Program encoding (*.TCN): Unicode

Recognize the code while saving □

□ Assigned configuration and customization in [inch]

□ Debug of TpaCAD

☑ Advanced user

☑ Stand-alone functionality

File location
- Support file to search the paths
  - C:\TPACAD\BIN\TPA.INI
- TpaCAD customisazion file
  - C:\TPACAD\TPACADCFG\CUSTOM\TPACAD.XML
- Program folder
  - C:\TPACAD\PRODUCT

<Professional>

**Select plant**

This option defines the way the selection of the plant is enabled. There are three options available:

- **Not managed**: impossible to select a plant. The plant used is always the defined one in the file TPA.INI
- **Enable in menu**: the selection of the plant is performed from the menu ⚙***->Plant***. The menu item is available only if the program is closed.
- **Enable on startup**: the plant is selected at the startup of TpaCAD and from the menu ⚙***->Plant***. The menu item is available only if the program is closed.
  The plant selection box proposes the list of the plants defined in the file TPA.INI.
  ***WARNING***: the selection of a plant, directly performed in TpaCAD, does not change the selection defined in the file TPA.INI.
  ***WARNING:*** the selection is not available in case of *Essential functionality.*

**"Drawing" environment**

Select to enable the control over a second operational environment, called "Drawing", which is alternative to the normal environment used. called "Machine". It is also possible to set the layer on which the switching to the "Drawing" environment can be enabled.

Both environments coexist in the same plant (i.e., they do not require the installation of a double configuration for TpaCAD); they only work on different configuration files:

- tpacad.ini, for “Machine” configuration
- tpacad_adv.ini, for “Drawing” environment.

This functionality is for TpaCAD only and it does not concern the TpaWorks application program or the program optimiser.

The switching command between the two environments is on the main bar and it is enabled with closed program and at the layer set as follows:

"Drawing" environment is active

"Machine" environment is active.

If the management of both the operational environment is active, next to the first item of the menu of the configuration window, you will find the icon of the environment currently in use.
The control of the "Drawing" environment can meet particular requirements, such as

- a highly specific programming environment in the machine to enable the geometries and/or the section of the piece and/or the menu composition
- a sub-programs and/or macro-programs development environment, which is necessarily very diversified.

Controlling the "Drawing" environment can also be convenient only to differentiate an environment for normal use of TpaCAD, marked by simplified menus and one more rich and powerful, but requiring more experience in using the program.

When the option*Drawing Environment* , in Configuration and TpaCAD customisation is active not everything can be modified; following items are excluded:

- Configuration →Environment, page *General Settings*: setting "Configuration in [inch]"
- Configuration →Environment, page *Enable in Menu* : "Drawing Environment" setting
- Configuration →Environment, page *Components*: you cannot assign an Optimiser
- Configuration→Piece settings, pages: General, Custom sections
- Configuration →Piece settings, page *Piece geometry* : the selection "Piece Geometry"
- Configuration →Piece settings, page*Workings*: settings only can be set: "Include custom workings", "Active custom encryption codes" and "Include the Plug-in"
- Configuration →Open and save, page *Assign in matrix* : only the settings of Preview and Execution parameters can be modified
- Customisation: the pages of the technological Assignment

When the option '*Drawing Environment* is active, il possible to assign separately:

- the file of custom logo (name: TpaCadCfg\CUSTOM\LOGOCUST_ADV.png)
- the prototype files for a new program.

***WARNING:*** The selection is not available in case of *Essential functionality.*

- **Manage the settings in XML file:** the selected entry enables the data storage of the data configuration for the TpaCAD package in XML files, therefore with Unicode encoding. This feature is introduced with version 1.4 of TpaCAD, whereas earlier versions maintain the format of the file such as INI and ANSI encoding. This selection is stated only in order to focus on editing functionality but cannot be changed. The XML format management is always enabled.
- **(*.TCN) Program encoding:** this options assigns the encoding that can be applied to read-write programs. The available options are two:
  - **ANSI**: ANSI files only ca be read and written
  - **Unicode**: ANSI and Unicode files only can be read and written.

  The management of the ANSI format allows you to assign characters only for the encoding not greater than one byte.
  The management of the Unicode format allows you to assign encoding characters on two bytes.

  To achieve consistency of operation all the programs supplied with the base installation contain ANSI encoding.
  - Macroprograms
  - Examples of programs and subroutines

  As already mentioned, the encoding selection affects the recording format of the programs, not the syntax. The possibility to manage programs in Unicode encoding eliminates any restriction arising from the culture of use of TpaCAD; there limitations still are remain in case of operation with ANSI encoding.
  By means of the Unicode encoding, for example, you can assign a comment or a text using Chinese or Japanese ideograms or Arabic characters, or simply distinguish a accented vowel from a Cyrillic character.
  In the same cases, working with ANSI encoding Chinese or Japanese ideograms, or Arabic characters cannot be used. Furthermore, an accented vowel logged in Italy will be displayed with a Cyrillic character in Russia, thanks to the use of a different table of character encoding (CodePage), used for the ANSI format.

  A program that normally is logged in ANSI is usually managed with both the encoding settings.
  A program logged in Unicode format requires the Unicode encoding setting.

- **Recognize the code while saving:** select the item to ask the application to choose the saving encoding according to the program settings. The selection is significant if the Unicode encoding of TCN programs is enabled and if you ask to save with Unicode encoding: in this case, the Unicode encoding will be applied only if the programming has employed Unicode characters.

- **Assigned configuration and customization in [inch]:** it sets the measurement unit of the configuration data. If selected, the setting of the dimensions concerning the configuration and the customization of the application program are assigned in [inch], otherwise are assigned in millimetres [mm]. The setting is not applied to the machine technology. The programming units of the plant parameters is assigned in a different mode.
- **Debug of TpaCAD :** if selected, this option enables the debug functionality of TpaCAD. In this case:
  - in the Command area the time needed for the execution of particular operations is recorded (program loading, recording, overall graphic, element selection in graphic area, working selection)
  - recording of report file related to the development of a complex code useful, e.g. to assess FOR and IF cycles, declared in a macro or in a subroutine is enabled. If so, the Debug tab is active
  - in the area of ASCII text the expanded list is also open for both complex workings, whose explosion is disabled and for codes of conic (ellipse, oval);
  - in the expanded list of complex workings (in the area of ASCII text) also the information of Sequence number is recorded;
  - in interactive mode, when the Snap is active in the face, the Tooltip associated to the cursor shows the snap face number
  - in graphic selection of the area, in the Command section the indication of the selected area is shown;
  - this option contributes to the activation of protected features

  **WARNING:** active debug setting should be limited to the stage of an actual enhancement of a programand should be cancelled once this stage is finished. Debug setting is automatically disabled after a certain number of closures of TpaCAD at operator password.

  **WARNING:** the information on the selection is read and stored in the folder Tpa.ini (searched in the folder where the application program is installed or in the installation folder of the operating system) and thus it is common to each TpaCAD instance.
- **Power user:** if selected, it enables the functions considered useful to an user, who already approached TpaCAD environment. This selection is not available in case of *Essential* functionality. More specifically, following commands are available:
  - this command duplicates the current working and moves the current working to the previous o or to the next position in the window of working assignment
  - In the status bar, the command which activates or deactivates the complete program representation
  - the selection of the execution modes of the tool **Explode**
  - the management of the advanced tool **Create font from geometry**
  - clipboard management in the application of **General Tools**
  - complete help menu of parametric programming (complete list of the variable arguments and functions available in parametric programming).

- **Stand-alone functionality:** to be selected, if the TpaCAD application is used outside a TPA environment or in an old TPA environment (that is with Albatros application program prior to the 3.0. release).or to enable always the associated functionalilties. This selection enable the user to manage independently the functionalities that normally are started and controlled by the same TPA environment in an unified way. They concern the selection of the language and the management of the access level. More specifically:
  - in the TpaCAD status bar you will find an additional and working list of language selection.

    NOTE: The structure of the list of the available languages depends on a specific setting from an USB hardware key device.

    NOTE: The change of the language is not made immediately, but at the next startup of TpaCAD.

    In TPA environment, the language management is normally available through the combination of **[CTRL + /] keys** or by clicking the icon of the "**Windows** application bar
  - in the **menu the** Password level command is available to change the level access; it works, for example, by recognizing the daily password. It is always possible to activate the change of level by means of the **[CTRL + *]** combination key or click the iconfrom the " **Windows application bar** ". However, the activation may not work in TpaCAD, if it works in an old TPA environment ;

**WARNING:**
The access level that is selected with this second is common to the TPA environment too, that is installed and that works in the computer. The level access selected by the **Password level** command can still belong to the TpaCAD environment**.**

In the environment of TPA the access at the constructor level is possible in both modes: 🔑 **and [CTRL + *]):**

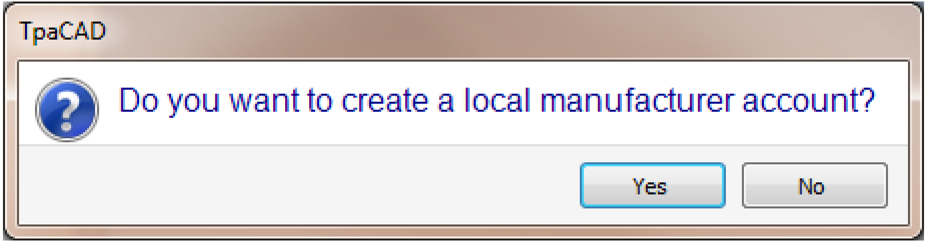
1. the operator knows the password corresponding to a constructor user (i.e: TPA has provided a licence file for that purpose) and can access only through the **[CTRL+* (asterisk)] command;** or, more usually
2. the operator knows the daily password (i.e. the pw provided by TPA and valid for the day of its release only)

Generally, the constructor level allows the operator to modify those parts of configuration of the various application packages, that normally are configured by the Machine constructor and that require specific knowledge also concerning the functioning of the numeric control.
if the **Stand-alone** functionality is enabled, also a mode to change the access level can be available for TpaCAD only (and TpaWorks). Therefore, the operator is not allowed to access the whole environment of TPA by means of the Constructor access. This is an additional functionality that does not replace the above mentioned ones. That is, the procedure that can be activated by the keyboard shortcuts **[CTRL+* (asterisk]** still works also for TpaCAD;
This functionality is activated by the Machine constructor in order to avoid the need to modify the access level of the whole TPA system.

**The Constructor takes the full responsibility for accessing locally the TpaCAD and allows the user to access and to freely modify the configuration of TpaCAD, which includes also the Custom Functions, the Global variables and the Database of the workings, defined by the Constructor.**
**T.p.a shall not be responsible for any incorrect operation or damage or consequences resulting from any changes inappropriate to the operation of the application.**

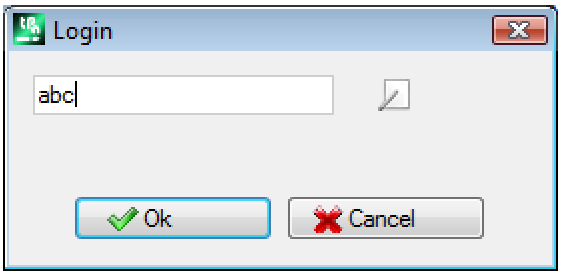
Now let us see how to activate it:
- first of all, the user should activate the access to the constructor level (according to one of the above mentioned modes);
- activate the **Stand-alone** functionality;
- then, select the **Password level** 🔑 command, (always from the Constructor level).

A window appears


TpaCAD

Do you want to create a local manufacturer account?

Confirm by selecting [Yes] and write the password (max. 20 characters, no spaces allowed) in the following window.


Login

abc

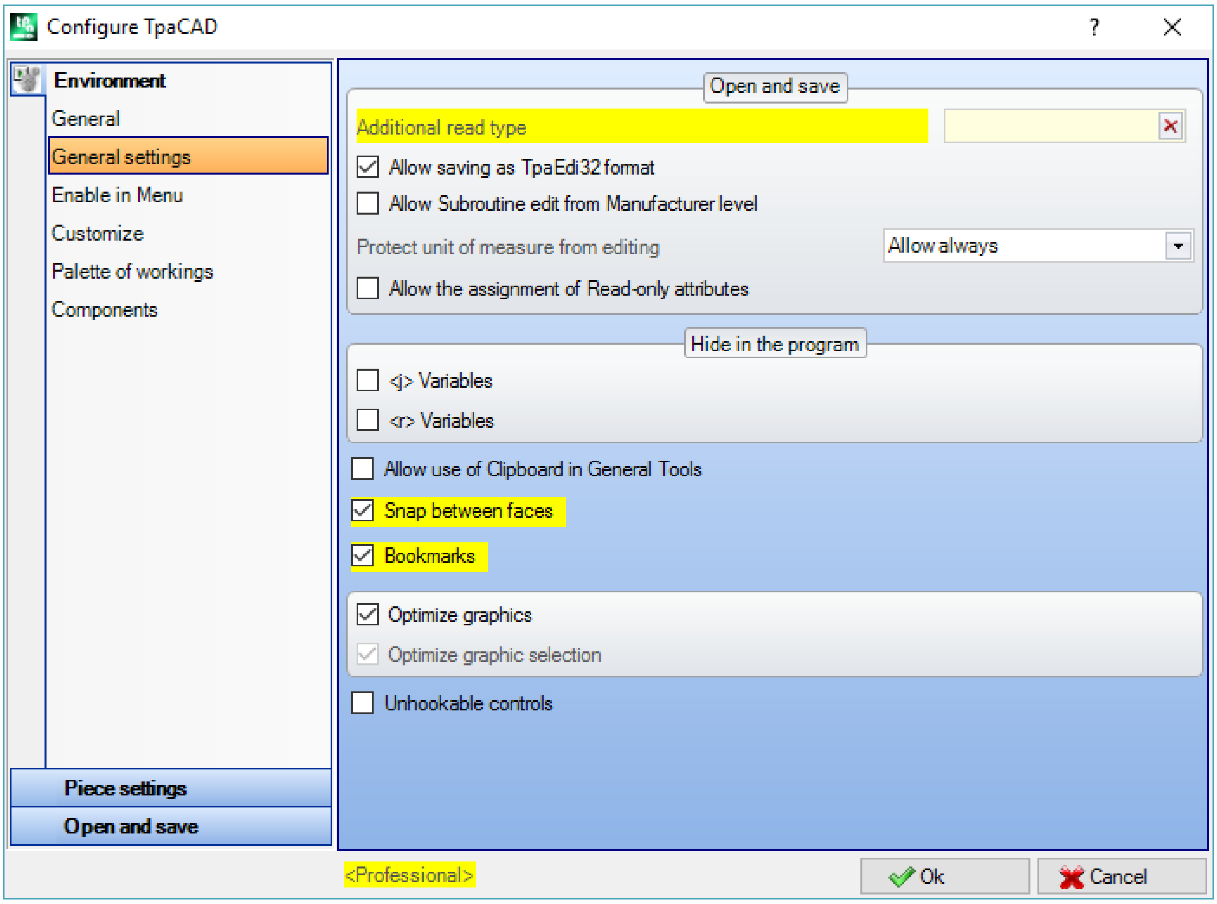
- The Constructor must release the password to the user.

This password defines the **Constructor** access to Tpa and can only be used from the **Password level** 🔑**:** then, the user can use the same password, after a proper setting, so as to customize the privileged access to his own installation of TpaCAD.

The **File location** group shows some information that are crucial for the definition of the work environment

- **Support file to search the paths** : position of the TPA.INI file, from which the settings are taken to find the predefined paths.
- **TpaCAD customisation file** : position of the TpaCAD configuration file. This file contains all the configuration and customisation settings of the TpaCAD suite.
- **Program folder**: default folder to store the TCN programs. The SUB folder inside it is the default location for the Subroutines.

## General Setting



Open and save

Additional read type

☑ Allow saving as TpaEdi32 format

□ Allow Subroutine edit from Manufacturer level

Protect unit of measure from editing: Allow always

□ Allow the assignment of Read-only attributes

Hide in the program

□ <j> Variables

□ <r> Variables

□ Allow use of Clipboard in General Tools

☑ Snap between faces

☑ Bookmarks

☑ Optimize graphics

☑ Optimize graphic selection

□ Unhookable controls

- **Additional read type:** a type of files, recognized as working programs for TpaCAD, may be added, so that it can be added to the default (*TCN) type. This setting works in Professional mode only. The user can assign a string, which should observe the following syntax: "*.EXT" where EXT assigns the extension, that will be recognized. A "*.*" extension cannot be assigned Following extensions are not accepted:*.TCN (default for programs and subroutines), *.TMCR (default for programs and macros).
- **Allow saving as TpaEdi32 format:** if enabled, it allows the storage of the programs under a format compatible with TpaEdi32. By compatibility we mean the possibility to read the program by use of TpaEdi32. As for the actual possibility to interpret the program, this is related to the workings and parametric programming in use, which should not include new functionalities introduced in TpaCAD.
- **Allow Subroutine edit from Manufacturer level :** if enabled, this allows the edit of a subroutine (creation, change) from the constructor password only. If the selection is active, you will be able to create some "client" Workings from Constructor password only.
- **Protect unit of measure from editing:**this allows to limit the access to the edit field of the unit of measure of a
- program. You can choose among three items:
  - **Allow always**: the change of the edit field of the measurement unit
  - **Lock if [mm]** : it locks the load and the creation of a program with the unit of measure in [mm]
  - **Lock if [inch]**: it locks the load or the creation of a program with the unit of measure in [inch].

  However, the Unit of measure field of a program can always be edited in case of macro or subroutine, when the **Allow Subroutine edit from Manufacturer level** is active.

  This option allows the configuration of an application with one programmable unit of measure ([mm] or [inch]) only. It is recommended to arrange the default prototype file, in which the correct unit of

measure is set (ex: [mm]). In each program created from the prototype file the unit of measure [mm] cannot be modified.
This option prevents the unit of measure field from changing also in case of import from a non-TpaCAD format to a TpaCAD-format.

- **Allow the assignment of Read-only attributes:** it allows the access to the editing of single sections or information groups of any program by constructor password only. The attributes concern the possibility to change dimensions, execution modes, variables (o, v, r), fictive faces, custom sections, sequences.
  - If the option is not enabled, it is never possible to assign the access status to the editing of the program sections. When a program is read, in which some restrictions in the section edit process are set, these are ignored.
  - If the option is enabled by a password only, it is possible to assign the access status to the editing of the program sections. This selection is not available in case of *Essential* functionality.

**Hide in the program**

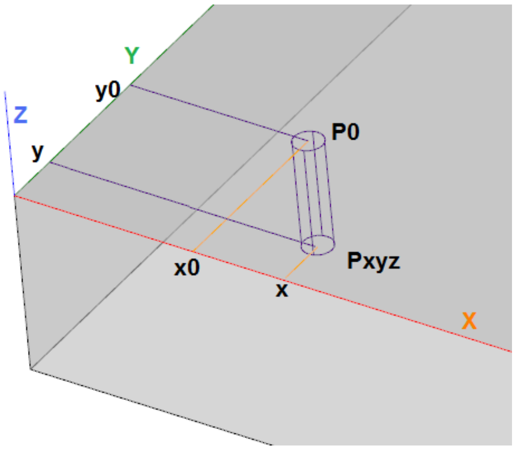
- **<j> Variables:** if enabled, this option does not show the tab of the <j> variables, if you are editing a program.
- **<r> Variables:** if enabled, this option does not show the tab of the <r> variables, if you are editing a program.

Both the tabs (<j> and <r> variables) are instead shown in a macro-program or subprogram edition.

- **Allow use of Clipboard in General Tools:** if enabled, it allows the application of general Tools (Translation, Rotation,..) to the workings copied into the Clipboard. This selection is not available in case of *Essential* functionality.
- **Snap between faces :** if selected, enables the activation of the snap between the faces during the execution of an interactive procedure (draw, tools, assignment of the current working). Snap between the faces is only available in Professional mode.
- **Bookmarks:** if selected, it enables the assignment of bookmarks on the piece. The assignment of bookmarks is only available in Professional mode.
- **Optimize the graphics:** if enabled, it activates the use in the graphic representation of the display lists, allowing the display times to be optimized The functioning by selected voice requires the use of more memory for the graphics and should be valued according to the basis of the PC configuration.
- **Optimize the graphic selection:** if enabled, it activates the mechanism to determine in the best way the working which corresponds to an interactive graphic selection. The functioning by selected voice requires more time of elaboration and should be valued on the basis of the TpaCAD program application type. The activation concerns two specific functioning aspects of the application program:
  1) the choice of the current working made by a mouse "click" in the graphic area;
  2) the search of the snap working, in interactive procedure (examples: draw, tool), both in mouse movement and during a final confirmation;

(for both points we apply the limitation to the case of 3D piece display).
If the authorization is active, the working is performed at graphic level first, by activating a special procedure finding the graphic object, which is represented inside a specific search area, around the mouse position. In case of positive outcome and if the element found is valid, the selection is confirmed for the use required, otherwise the search is made by direct comparison of the position for the mouse and the programmed geometries.
Requiring snap between the faces: the search of the working is first made in any case graphically.

Let's focus the matter by an example

| | |
|---|---|
|  | The figure concerns a point working applied to the upper face, by the usual representation of an overall cylinder in the piece:<br>• the entry of the piece is located above, in the point P0;<br>• the point of the tool is located below, in the point Pxyz.<br><br>If the working is programmed at the (100;100;-20) coordinates:<br>• P0 is at the (100;100;0) coordinates;<br>• Pxyz is at the (100;100;-20) coordinates.<br><br>In the view of face, simply by moving the mouse to P0, you can see - in the status bar - that the position of the corresponding face is around (100; 100), in accordance to what you have programmed. Moving to Pxyz, the corresponding position obtained can be around (130;25), depending on the piece rotation. |

The position of the face is derived from the mouse position with a simple geometric construction, assuming that the mouse is on the xy plan of the face, that is at the coordinate z=0. In the figure we can see the geometric construction in both cases:

- if the mouse is next to P0: we obtain the (x0;y0) coordinates around of (100;100);
- if the mouse is next to Pxyz: we obtain the (x;y) coordinates in our example around of (130;25);

Supposing that the mouse is on the xy plan of the face, that is at the z=0 position, this compensates the loss of information because the video has 2 dimensions.

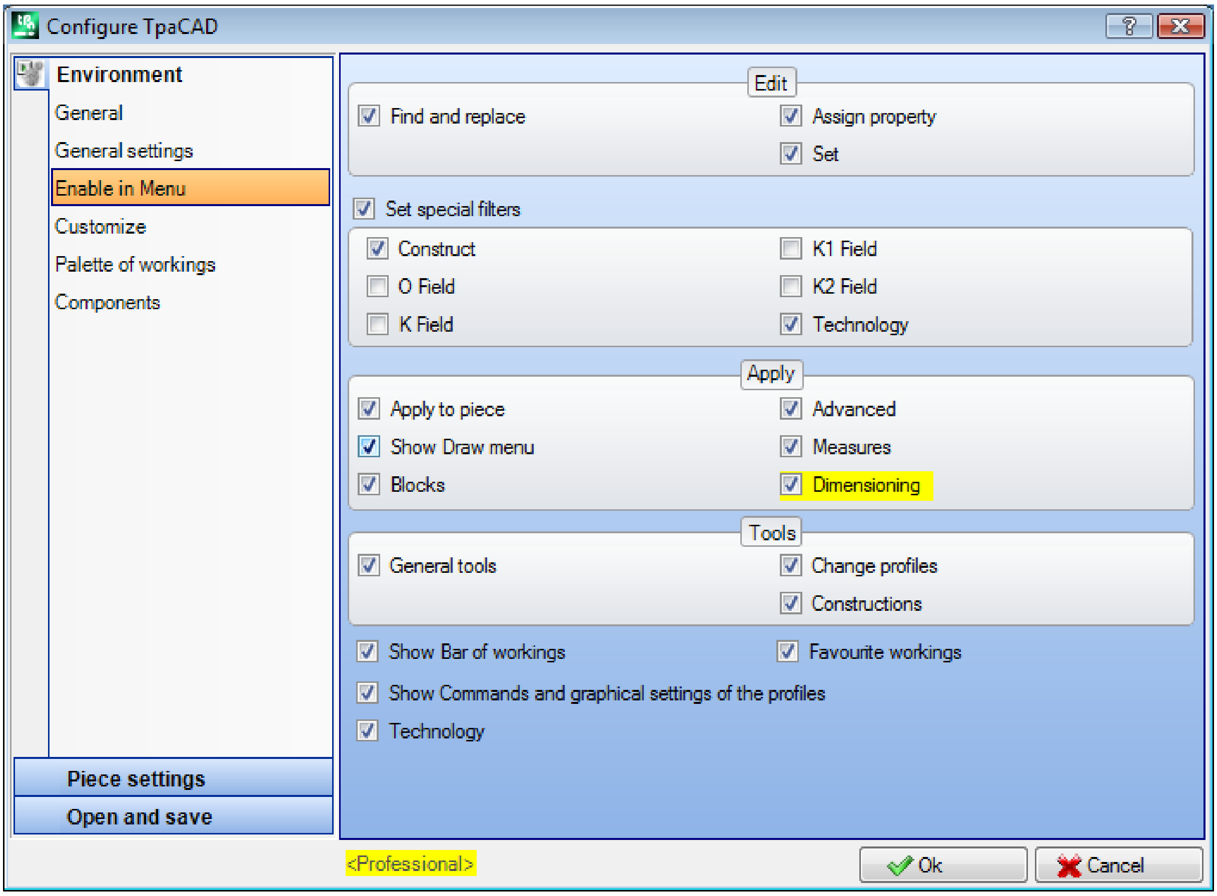
- The three-dimensional representation of the piece always looses the information on the programmed depths, because the video has 2 dimensions. The working of the example, programmed in (100;100;-20), is displayed on a plane, the screen;
- this loss of information is not relevant until affects only the representation of the working, while it is relevant when you want to make the representation interactive.. For example: I click on the piece and "go" to the clicked working.

And this is the issue key: the interaction with the graphics.
For example, if a snap procedure to programmed quotes is activated

- and the search by graphic selection is active: if the mouse is near Pxyz, our working can be directly identified,
- otherwise, the search is performed by the evaluation of a geometric correspondence among the programmed entities, by shifting a face position to (130;25) and our working cannot be "identified".

- **Unhookable controls:** by selecting this option, you can unhook some controls. This selections is not available in the *Essential* function or if the *Advanced user* selection is not active. The activation applies to the controls that correspond to three areas: data processing, ancillary tab (commands, errors, ...) and general assignments of the piece (dimensions, variables, special sections, etc.).

## Enable in Menu



Edit
☑ Find and replace ☑ Assign property
☑ Set

☑ Set special filters
☑ Construct □ K1 Field
□ O Field □ K2 Field
□ K Field ☑ Technology

Apply
☑ Apply to piece ☑ Advanced
☑ Show Draw menu ☑ Measures
☑ Blocks ☑ Dimensioning

Tools
☑ General tools ☑ Change profiles
☑ Constructions

☑ Show Bar of workings ☑ Favourite workings
☑ Show Commands and graphical settings of the profiles
☑ Technology

<Professional>

This page is available from the Maintenance level.

**Edit** groups the activation items in the corresponding menu

- **Find and replace** : enables the corresponding command in the *Edit group.*
- **Assign property** : overall activation of the command group for the global assignment of the working properties.

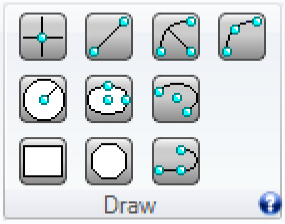
- **Set**: overall activation of the command group to assign Exclusions and Filters of display and modification.

- **Set special filters:** if selected, this option enables the assignment and the evaluation of the conditions of view and change by mean of the application of the special filters. In the next grouping check the boxes, which correspond to the properties of the workings or to other assignments for which it is required to enable the setting and the application of special filters:
  - **Construct:** if selected, this option enables the assignment of the "B" field values. This selection is ignored if the priority is not enabled;
  - **O field:**if selected, it enables the assignment of the "O" field values. This selection is ignored if the property is not enabled;
  - **K field:** if selected, it enables the assignment of the "K" field values. This selection is ignored if the property is not enabled;
  - **K1 field:** if selected, it enables the assignment of the "K1" field values. This selection is ignored if the property is not enabled;
  - **K2 field:** if selected, it enables the assignment of the "K2" field values. This selection is ignored if the property is not enabled;
  - **Technology:** if selected, it enables the assignment of the technology settings.

  ***WARNING:***The selection is not available in case of *Essential functionality.*

Apply groups activation items in the corresponding menu

- **Apply to piece** : overall activation of the groups of the program overall assignment tools.
- **Display Draw Menu:** if selected, this option enables the view and the usage of the command group **Draw** in the tab**Edit**. The commands of the group Draw allow the direct insertion of the workings by means of the selection of geometric entities, without needing to access to the tab of**Workings**. Managing the commands of the group**Draw** and at the same time deactivating the tab commands of **Workings** (see following selection) may allow, for instance, the configuration of TpaCAD in CAD functionality only. In this configuration it is possible to manage the workings read by a program already recorded, and inserted by means of the command group **Draw** or by means of tools such as Text generation, Emptying, Profile building. The group **Draw** is made of the following:



| | |
|---|---|
| ✛ | **Point:** Available, if a point working default code is assigned (for the current face or without distinction for the face). |
| ╱ | **Line:** available, if in the database of the workings the code COPL01 is assigned. |
| ◜ | **Arc (centre, beginning, end):** available if in the database of the workings the code COPA01 is assigned, with managed parameters assigning the initial point of the segment in the xy-plane. |
| ◠ | **Arc (3 points):** available, if in the database of the workings the code COPA04 is assigned. |
| ◎ | **Circle:** available if in the database of the workings the code COPA45 is assigned, with managed parameters assigning the initial point of the segment in the xy-plane. |
| ⬭ | **Ellipse:** available if in the database of the workings the code COPA42 is assigned, with managed parameters assigning the initial point of the segment in the xy-plane. |
| ◡ | **Arc of ellipse:** available if in the database of the workings the code COPA43 is assigned, with managed parameters assigning the initial point of the segment in the xy-plane. |
| □ | **Rectangle:** available if in the database of the workings the code COPL16 is assigned, with managed parameters assigning the initial point of the segment in the xy-plane. |
| ⯃ | **Polygon:** available if in the database of the workings the code COPL17 is assigned, with managed parameters assigning the initial point of the segment in the xy-plane. |
| ⌇ | **Polyline:** available, if in the database of the workings the code COPL01 and/or COPA04 is assigned. |

- **Blocks**: overall activation of the insertion group for the logical blocks (IF ... ENDIF FOR .. ENDFOR)

- **Advanced** : overall activation of the group for advanced tools.
- **Measures**: overall activation of the group for the *Measure tools.*
- **Dimensions** : overall activation of the group for *Dimensions*
  (if all the selections of the*Apply*  menu are disabled, the same menu cannot be seen)

**<u>Tools</u>** groups activation items in the corresponding menu

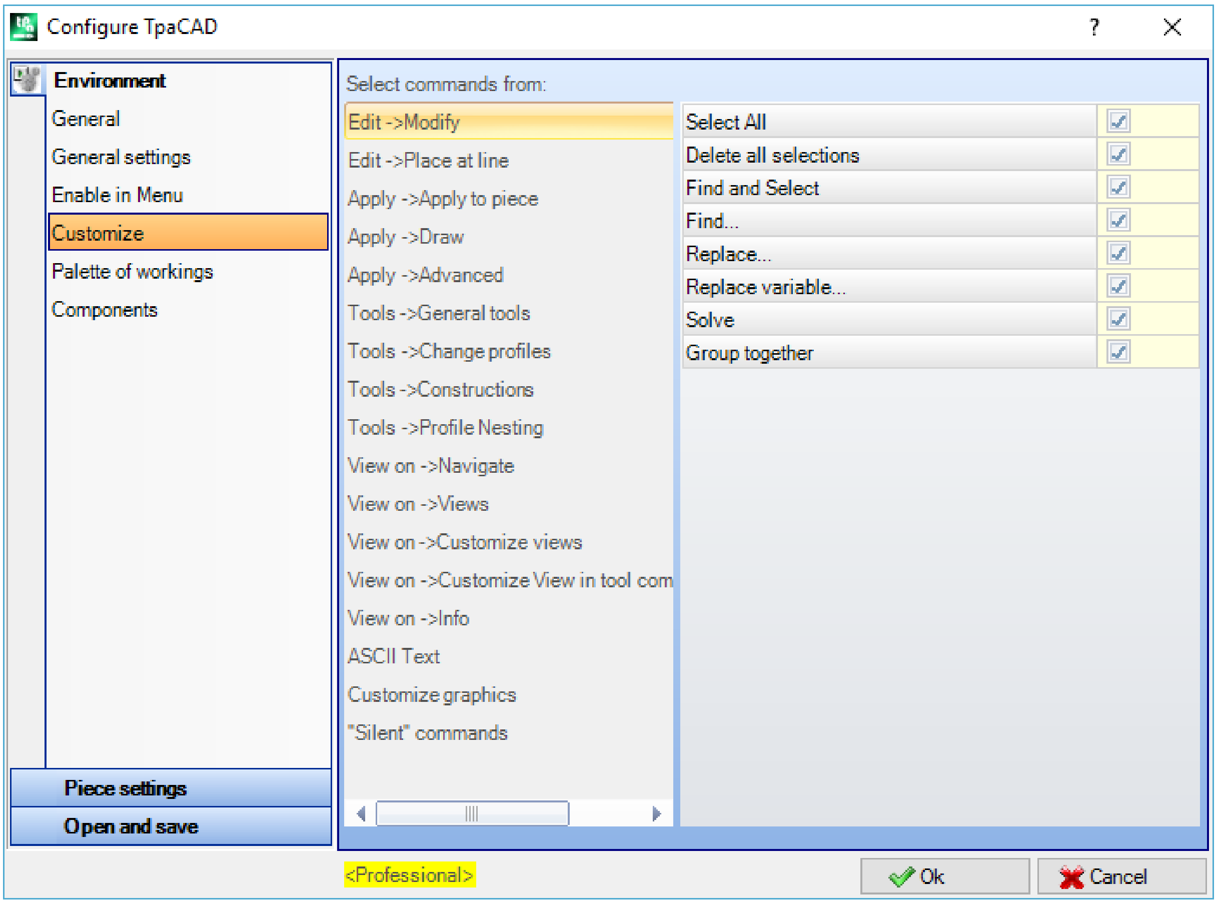
- **General**: overall activation of the group for General Tools (Translate, Rotate…)
- **Edit profiles**: overall activation of the group for Profile editing tools (Change, Reverse…)
- **Constructions**: overall activation of the group for Construction tool (Compensated profile, Apply connector,…)
- (if all the selections of the*Tools*  menu are disabled, the same menu cannot be seen)

- **Display Bar of workings:** if enabled, this options enables the view of the tab **Workings**. ***WARNING***: managing the database of the workings does not depend on the fact that the tab **Workings** can be seen. More specifically the groups of which the tab is made should be assigned in a proper way, even if the tab **Workings** cannot be seen. When assigning the tools, that apply a technology (example: Text generation) the only workings available in the tab Workings are those available in the Working tab.
- **Favourite workings:** when selected, this option enables the display of the list of favourite workings, from the Quick Access Toolbar

- **Display profile commands and graphical settings:** if selected, this option enables the view of the commands of the settings for the profiles, such as:
  - in the group **Customise views**  in the tab **View on** the commands: Direction of profiles, Points on profiles, Profile thickness in compensation, Original profiles in compensation
  - in the group **Views**  of the tab **View on** the command Tool Compensation
  - group **Customise View**  in Tool compensation of the tab **View on**.

- **Technology:** if selected, this option enables the view and the interaction with the plant technology window. If this option is not selected, TpaCAD deactivates the command in the opening menu of the technology window and it does not manage the interaction with the window itself from the parameter of the working (tool, electric spindle).

## Customize



| | |
|---|---|
| Select All | ☑ |
| Delete all selections | ☑ |
| Find and Select | ☑ |
| Find... | ☑ |
| Replace... | ☑ |
| Replace variable... | ☑ |
| Solve | ☑ |
| Group together | ☑ |

This page is available from the Maintenance level.

This page allow a specific activation for every single option available in certain groups of command of the menu. More specifically, you can

- disable single commands, in the main menu or in the local menus
- customise commands and specific functions, that are not available in another way.

**Setting a page requires to work very carefully, in such a way that situations of "wrong functioning", as better explained later, do not arise.**

The list on the left shows the available groups: by selecting an option, the list on the left shows the options of the available command for the group. The groups and the single options are listed with the message used in the menu, possibly as a tooltip message.
As initial condition, all the listed options are selected.

Selection of groups shown in the main menu:

- **Edit ->Modify:**

Group of commands in the *Edit*menu.
Do not set anything if the group is completely deactivated in the section *Enable in menu* .

- **Edit ->Place at line:**

Group of commands in the *Edit*menu.

- **Apply -> Apply to piece:**

Group of commands in the *Apply* menu.
Do not set anything if the group is completely deactivated in the section *Enable in menu* .

- **Apply ->Advanced:**

Group of commands in the *Apply* menu.
Do not set anything if the group is completely deactivated in the section *Enable in menu* .
A particular case concerns the option “*Convert [mm]-[inch]*”: in fact, the command is not available in the menu and can be activated only by activating the selection here.

- **Tools ->General:**
- **Tools ->Change profiles:**
- **Tools ->Constructions:**
- **Tools->Profile Nesting:**

Group of commands in the *Tools* menu.
Do not set anything if the group is completely deactivated in the section *Enable in menu* .

- **View on ->Navigate:**
- **View on ->Views:**
- **View on ->Customize views:**
- **View on ->Customize view in tool compensation :**
- **View on ->Info:**
- Group of commands in the *View* menu.

The availability of some groups or single commands is already managed in the *Enable in Menu* section of from activations of working properties (see: Special views).
For the options that correspond to *active /non active setting,*, if the selection is taken away , the last stored status is applied; so, we suggest that you should save one situation you consider to be "valid", before modifying the configuration. For example: take away the selection in the option **Grid**, in **View ->Customize views:**

- If TpaCAD is working with active option, the grid will be now always displayed, without any possibility to deactivate it.
- However, if TpaCAD is working with non active option, it will be never possible to display the grid.

The second case probably correspond to the result you wish to obtain, while the first situation could be interpreted as a functioning error of TpaCAD..

A second example explains a situation that can be totally different: take away the selection on the **Direction of profiles**
in **View on -> Customize views**

- If TpaCAD is working with active option, the direction arrows on the profiles are always displayed now, without any possibility of deactivation.
- If TpaCAD is working with non active option, the opposite will be right.

Both these functions can correspond to a precise functioning option.

Selection of groups corresponding to local menus:

- **ASCII Text:** shows the options that are part of the local menu of the ASCII text. Let's see two options in detail:
    - **Expand the working**: remove the selection to deactivate the possibility to open the window that corresponds to the expanded list of a complex working;
    - **Explosion**: if the option is active and the previous one is non active, this option enables the *Explosion* general tool in the local menu of the ASCII text.

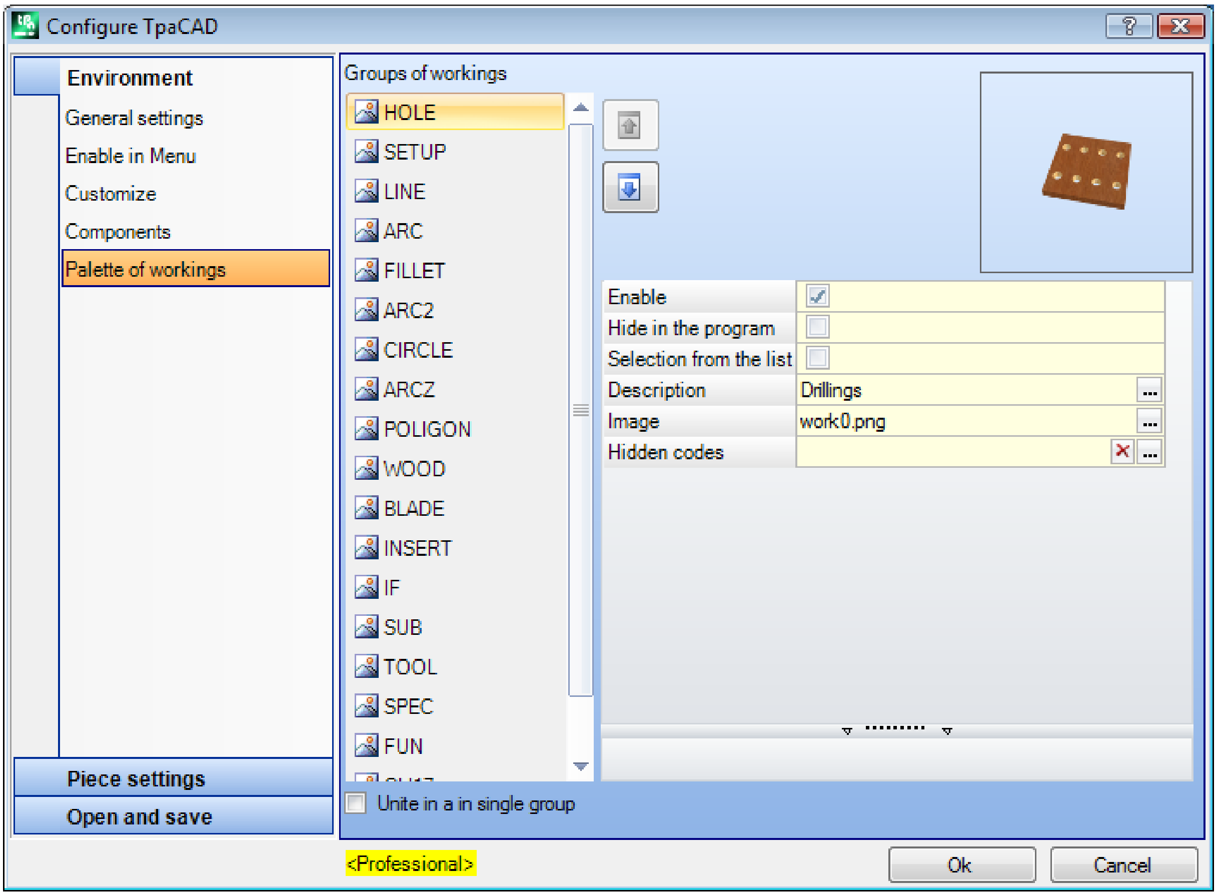
We recommend you to set possible deactivations according to a general criterion of homogeneity of the commands. For example, if you wish to remove the navigation commands among the logical branches of the program, remove the whole group of the concerned command, not only a part of them.

- **Customize graphics:** this option show the items that make up the local menu in the graphic area.

The last option allows you to customize and to simplify some commands:

- **no Verbose:**
    - **New File** : by selecting the option, the creation of a new program does not suggest the choice of his type (program, sub-program, macro-program) and selects directly the type of the program. It is always possible to change the type in the window of the dimensions.
      Do not select the option, if you need to manage differentiated prototypes by type.
    - **Explosion**: by selecting the option, the corresponding general tool activates directly the widest explosion available for the working(s) to be exploded

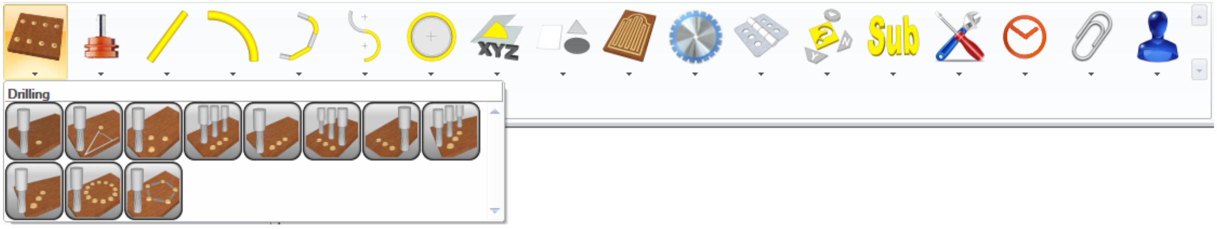
## Palette of Workings



| | |
|---|---|
| Enable | ☑ |
| Hide in the program | □ |
| Selection from the list | □ |
| Description | Drillings |
| Image | work0.png |
| Hidden codes | |

□ Unite in a in single group

This page is available from the Maintenance page. In this page we set the tab of the Workings:



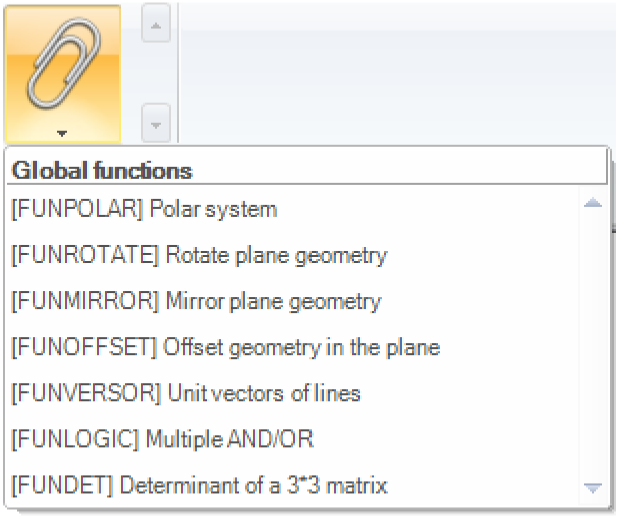
## Groups of workings

The groups of workings defined in the database of the workings are listed as follows:

- the name of each group corresponds to the grouping in the database;
- the icon shows if the group is enabled or disabled ;
- the displayed order represents the view order in the tab of **Workings** (see the picture).

To customize the single group, you need to modify the values of the fields listed on the right.

- **Enable:** if selected, it enables the view of the group in the tab of **Workings**. It is possible to enable up to 25 groups. After reaching this number of permissions, to add a new group you need to take away the permission of another one.
- **Selection in the list:** if selected, it enables the composition of the group into a list of items, instead of images. The selection meets some working groups to which the preparation of images is not of usefulness: an example of application is the group of the global functions:



Global functions

[FUNPOLAR] Polar system

[FUNROTATE] Rotate plane geometry

[FUNMIRROR] Mirror plane geometry

[FUNOFFSET] Offset geometry in the plane

[FUNVERSOR] Unit vectors of lines

[FUNLOGIC] Multiple AND/OR

[FUNDET] Determinant of a 3*3 matrix

- **Description:** this is a message describing the name of the group of workings. The message is placed above the list which corresponds to a button of the tab of **Workings** (Example: Drilling, Sawing works, Insertions). If you select the button ..., the window of the list of messages in the custom file CADAUX (stored in the folder tpacadcfg\custom) is opened. Messages from 1 to 50 are listed, where you can choose, modify or entry the message required.
- **Image:** it is the image associated to the group (button) and displayed in the tab of **Workings** on the top of the page. If you select the button ...,a window is opened to select the image. The image files stored in the configuration folder (tpacadcfg\custom\dbbtn) are displayed. The formats recognized valid are *.JPG, *.PNG, *.BMP.
- **Hidden codes:** it assigns the list of the group workings, which should not be displayed. If you select the button ... the window containing the list of the group workings is opened. Check the workings to be excluded from the display. If some codes seem to be excluded, the field shows the writing "{…}", otherwise it is set empty. By means of the button ☒ you delete the hidden codes already assigned.

To change the order of the groups of the workings, it is sufficient to select the group and then select the buttons ⬇ or ⬆, until the order required is reached.

Whole groups or workings here excluded from the view are always valid; however, they cannot be selected from the tab of the **Workings**.

A group without assigned workings is not displayed.

- **Unite in a single group.**

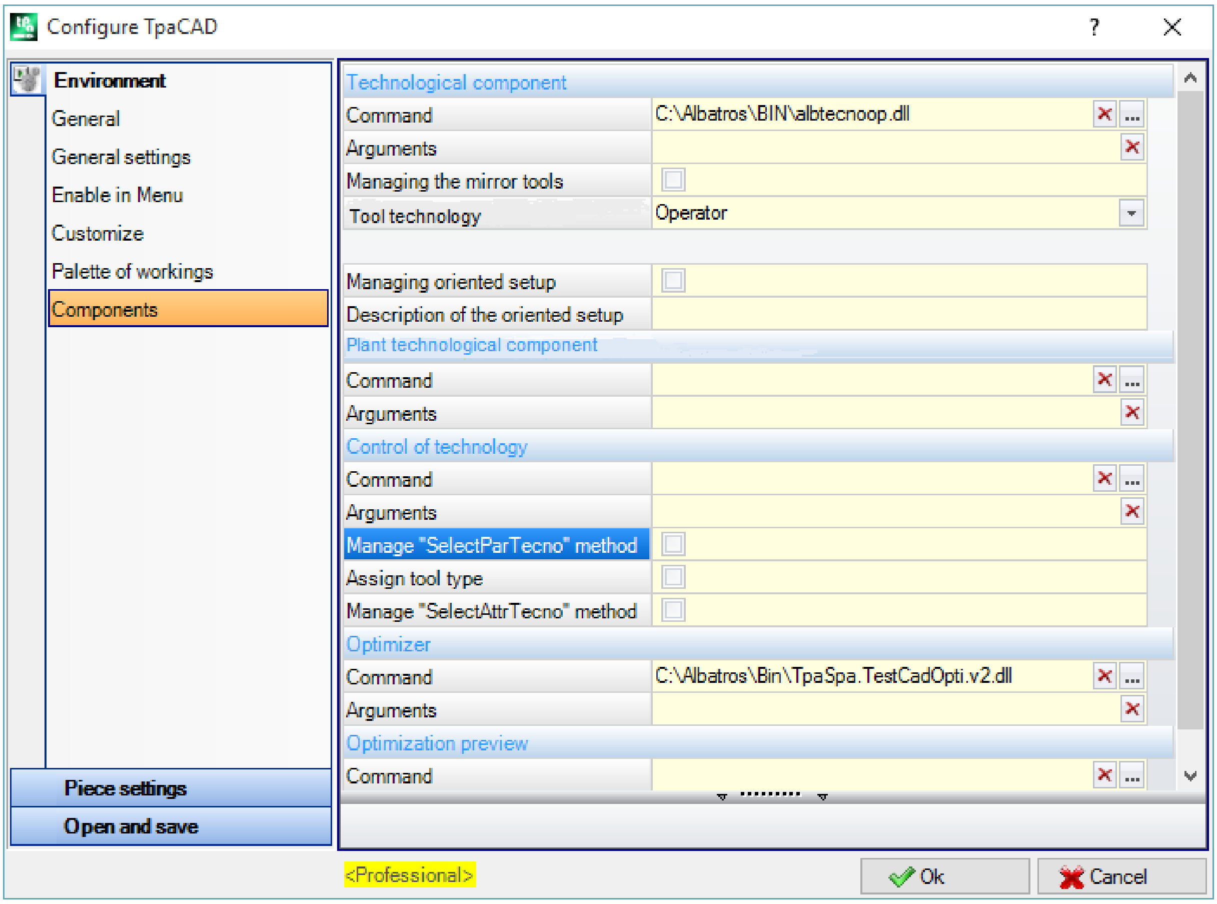
If selected, this option gathers all the workings in one list. In this case, only the images assigned to the workings are used and not those assigned to the buttons of the groups.

Anyway, you need to assign the working palette divided into groups, because

- only the workings enabled on the palette are shown;
- the presentation order of the working corresponds to that of the groups;
- the groups for those the option *Select in the list* is selected are excluded.

This option is useful when you require to enable few workings.

## Components



| | |
|---|---|
| **Technological component** | |
| Command | C:\Albatros\BIN\albtecnoop.dll |
| Arguments | |
| Managing the mirror tools | □ |
| Tool technology | Operator |
| Managing oriented setup | □ |
| Description of the oriented setup | |
| **Plant technological component** | |
| Command | |
| Arguments | |
| **Control of technology** | |
| Command | |
| Arguments | |
| Manage "SelectParTecno" method | □ |
| Assign tool type | □ |
| Manage "SelectAttrTecno" method | □ |
| **Optimizer** | |
| Command | C:\Albatros\Bin\TpaSpa.TestCadOpti.v2.dll |
| Arguments | |
| **Optimization preview** | |
| Command | |

In this page we are defining the external components used by TpaCad.
In the **Command** field the complete path of the component shown, whose extensiont is *.exe, or *,dll.
In the field **Arguments** the arguments are set, which are have been given to the component and whose meaning is assigned by the control operation. To assign the field Arguments a generic syntax we suggest to adopt, is defined in order to keep unified the specifications of realisation of the components used in TpaCAD environment:

- field divided by a space and headed by '/'
- structure of each field: "/name=date", where: "name" univocal name of the field '=' separator between name and value, 'date' value assigned to the field.

Example: "/aa=12 /bb=78"

The image ❗ displayed in a row of **Command** shows that it has not been possible to create an instance of the component: by selecting the corresponding row, in the reserved area for descriptive message, are kept those indications, that explain the reason for the negative result.

**Enable** field: if selected, it enables the control of the component shown.

**Technological component**

The technological component allows the access to all the technological data of the plant. Its ProgId is "ExecutableName.CAlbTecno", where ExecutableNameis the name of the component. E.g., in the picture the executable name is "customopti".
In general, the technology of the plant can include each information useful to define how each technological part works, from the composition of a line to the definition of each individual machine, from the definition of the equipment of the groups of a machine to the description of the tool holders.
For the description of the methods used, see the dedicated documentation.

- **Managing the mirror tools:** select the option to enable control and presentation of the mirror tools. This information is used to:
    - modify the assignment of technology for the profiles, in case of modification of mirror tool or inversion;
    - enable the presentation of the mirror technologies in the technological table.
- **Managing oriented setup:** select this option to enable the solution of the geometries oriented to the level of the technological component. This control allows you to assign customized programming and interpretation of an oriented setup and of the associated profile. The next field **Oriented setup description** identifies all the parameters you need to program the setup, in the form "n1-c1;n2-c2;.." and with a maximum length of 150 characters for the string:
    - n1=numeric identifier of the parameter (ex: 9123). It accepts values between 1 and 10000.
    - c1= column of storage in the matrix (ex: 25). It accepts values between 1 and 100.

If the control is enabled, the **Description of the oriented setup** field must be assigned.

**WE MEAN THAT THE DEVELOPMENT OF THE TECHNOLOGICAL COMPONENT IS CARRIED OUT BY AN APPLICATOR, BECAUSE OF THE CRITICALITY THAT CAN ARISE, IF YOU LET AN EXTERNAL COMPONENT DETERMINE THE REPRESENTATION OF THE ORIENTED PROFILES . IN THE EVENT THAT YOU ENCOUNTER DOUBTS RELATING TO THE OPERATION AND/OR ALSO UNUSUAL SLOWDOWNS ONLY, DUE TO THE GRAPHICS PART, BEFORE FORMULATING ANY OTHER HYPOTHESIS, WE RECOMMEND TO DISABLE THE MANAGEMENT AND TO TRY THE STANDARD OPERATION.**

### Plant technological component

The **Plant technological component** can allow accessing the technological data of the plant depending on the tooling technology. Its ProgId is "ExecutableName.CAlbTecno", where ExecutableNameis the name of the component.

If it not assigned, the access to the technology is always defined by the **Technological component**. Otherwise,

- the **Plant technological component** is queried to request information on plant composition (how many machines, how many groups are assigned on a machine, ...)
- The **Technological component** is queried to request information on the tooling of a particular group or on the availability in the tool holder.

For the description of the methods used, see the documentation provided (**Technological component**).

### Control of Technology

The Control of Technology is the control showing the technology.
It can be assigned by selecting typologies of executable file: *.exe, or *,dll. Its ProgId "ExecutableName.CTecnoView", where ExecutableNameis the name of the component.
Following methods are used:

- **void NameMachine(string stFolder, string stMachine)**
  the method is invoked, when in the TpaCAD environment a selection of Machine change has been assigned. Return value of the method is ignored.

| | |
|---|---|
| string stFolder | this is the folder where the Tpa.ini file is searched (if the string is empty, the default folder is considered) |
| string stMachine | this is the name of the machine (section defined in the Tpa.ini file If the string is empty, the name of the default machine is considered) |

- **bool ViewTecno(string stArgs, string stSettings)**
  the method is invoked, when the opening of the technological table from the menu is required. Return value of the method is not tested.

| | |
|---|---|
| string stArgs | arguments assigned in the Arguments field |
| string stSettings | these are the remarkable arguments that correspond with the call Format and meaning of the fields (separator: space; fields headed by '/', without distinction between upper- and lower case letters (case-insensitive)<br>"**/unit**=.."client unit of measure (0=mm; 1=inch). Default is 0<br>"**/attr**=.."current tooling (if >=0); if it is negative (by default), the window does not show the tooling information. If the value is positive (>=0) it shows that for the section Special Settings the field of Tooling is set.<br>"**/lng**=.."current language (three letters abbreviation. Example: "ITA", "DAN")<br>"**/lngdef**="default language (three letters abbreviation among: "ITA", "ENG", "FRA", "DEU", "ESP) |

- **bool SelectTecno(string stArgs, string stSettings, ref int nMac, ref int nGroup, ref int nSubGroup, ref int nTool)**

  the method is invoked, when it is required to open the table of the tools from Working edit from Tool or Spindle parameter.
  Method return value:
  - **false** the call has not determined any assignment of the technology (it may also mean that call failed)
  - **true** the call has determined the assignation of the technology and the values machine, group, spindle tool are returned.

| string stArgs | arguments assigned in the Arguments field |
|---|---|
| string stSettings | these are the remarkable arguments that correspond with the call Format and meaning of the fields (separator: space; fields headed by '/', without distinction between upper- and lower case letters (case-insensitive)):<br>"**/unit**=.."client unit of measure (0=mm; 1=inch). Default is 0<br>"**/attr**=.." current tooling:<br>• if the tooling is numeric: value (>=0) is the tooling number; if negative (default), the window does not show the tooling information;<br>• if the tooling is selected by name: the string between double quotation marks is the name of the tooling (example: "… ./attr="attrezza4"…."); negative value (-1) shows that no tooling is assigned;<br>"**/editM**=.." it manages the machine selection if >0; the selection is not managed if =0.<br>"**/editG**=.." it manages the group selection if >0; the selection is not managed if =0<br>"**/editS**=.." it manages the spindle selection if >0; the selection is not managed if =0<br>"**/editS**=.." it manages the tool selection if >0; the selection is not managed if =0<br>"**/nSide**=.." working application face (automatic number)<br>"**/nCode**=.." working operating code<br>"**/nTipo**=.." operating code typology (0=point, 1=setup, 2=arc, 3=linear, 4=logic). In the event of a complex working, the typology is assigned according to the sub-type of the operation code (set in the database of the workings): 1=setup, if the sub-type is (1, 2); otherwise: 0=punctual<br>"**/tectip**=.."extracts the homonymous field from the attribute "Trasformate" of the corresponding parameter (Tool or Spindle). In the tool selection the fields can assign a filter associated to the specific working (for example, a filter of the typology of the tools: "1-30", "50,51,52", "100-120")<br>"**/lng**=.."current language (three letters abbreviation. Example: "ITA", "DAN")<br>"**/lngdef**=" language by default (three letters abbreviation: "ITA", "ENG", "FRE", "GER", "SPA") |
| ref int (nMac, nGroup, nSubGroup, nTool) | with return**true**, the technology assignments concerning machine, group, spindle, tool are returned. |

- **bool SelectParTecno(string stArgs, string stSettings, ref int nMac, ref int nGroup, ref int nSubGroup, ref int nTool, ref int nParam)**

  the method is invoked when the opening of the tool table is required from the help menu in parametric programming, if the field **Manage SelectParTecno** is selected. (***ATTENTION:*** the selection of the field is not available in case of *Essential functionality).* Return value of the method can be:
  - **false** the call has not determined any assignment of the technology (it may also mean that call failed)
  - **true** the call has determined the assignation of the technology and the values machine, group, spindle tool are returned.

| string stArgs | arguments assigned in the Arguments field |
|---|---|

| | |
|---|---|
| string stSettings | these are the remarkable arguments that correspond with the call Format and meaning of the fields (separator: space; field headed by '/', without distinction between upper- and lower case letters (case-insensitive)):<br>"/unit=.."client unit of measure (0=mm; 1=inch) - (default=0)<br>"**/attr**=.." current tooling:<br>• if the tooling is numeric: value (>=0) is the tooling number; if negative (default), the window does not show the tooling information;<br>• if the tooling is selected by name: the string between double quotation marks is the name of the tooling (example: "… ./attr="attrezza4"….”); negative value (-1) shows that no tooling is assigned;<br>"/lng=.."current language (three letters abbreviation. Example: "ITA", "DAN")<br>"/lngdef="default language (three letters abbreviation among: "ITA", "ENG", "FRA", "DEU", "ESP) |
| ref int (nMac, nGroup, nSubGroup, nTool, nParam) | with return**true**, in the variables the technology assignments concerning machine, group, spindle, tool, parameter identification number are returned. Anyway, the return assignments are ignored if nParam<=0 (negative or invalid). The assignment returned are use to make up a string of parametric programming. Possible cases are as follows:<br><br>read a plant parameter: (nMac<=0) primp[nParam]<br>reads a machine parameter: (nGroup <= 0) prmac[nMac; nParam]<br>reads a group parameter: (nSubGroup <= 0, nTool <= 0) prgr[nMac; nGroup; nParam]<br>reads a tool parameter: (altrimenti) prtool[nMac; nGroup;; nSubGroup; nTool; nParam] |

- **bool SelectAttrTecno(string stArgs, string stSettings, ref int numAttr, ref string nameAttr, ref string tecnoAdd)**

  the method is invoked when the Tooling field (numeric) is assigned, if the field **Manage SelectAttrTecno** is selected.

  This method is used in the Special settings section, in case selecting the tooling involves the assignment of set of additional information of the program. (***ATTENTION:*** the selection of the field is not available in case of *Essential functionality).* Return value of the method can be:

  - **false**the call has not determined any assignment of the technology (it may also mean that call failed)
  - **true** the call has determined the assignation of the technology and the assignments to be set are returned.

| | |
|---|---|
| string stArgs | arguments assigned in the Arguments field |
| string stSettings | these are the remarkable arguments that correspond with the call Format and meaning of the fields (separator: space; field headed by '/', without distinction between upper- and lower case letters (case-insensitive)):<br>"**/unit**=.."client unit of measure (0=mm; 1=inch) - (default=0)<br>"**/attr**=.." current tooling:<br>"**/lng**=.."current language (three letters abbreviation. Example: "ITA", "DAN")<br>"**/lngdef**="default language (three letters abbreviation among: "ITA", "ENG", "FRA", "DEU", "ESP) |
| int numAttr | [in] current tooling<br>[out] selected tooling |
| string nameAttr | it returns the name related to the the selected tooling. If a name is not returned, the display in the piece shows the numeric value. |
| string tecnoAdd | Technology associated to the tooling, significant as [in]/[out]. Example: "#1=.. #2=.."<br>Format and meaning of the fields:<br>• separator: space<br>• single field "#n=v", with:<br>n=number from 1 to 20; identifies a technological information<br>v=associated value<br>field not returned are not modified.<br><br>The association of the field in the Special settings section concerns the name of the section: names from "ATTR1" to "ATTR20" must be configured when the fields from "#1=.." to "#20=.." are assigned. Valid typologies for the fields are: numeric (integer, double, list), parametric, string. |

- **Assign tool type:** select the item to enable the assignment also of the tool type, with interactive selection of a working technology.

if the control of the technology is not assigned, TpaCAD uses a control by default, that shows the technology data organized in a table.
Also in this field (field empty Command) it is possible to assign some arguments, whose meaning is interpreted by the control by default.

**Optimizer**
It is possible to configure an application program to optimize the programs, choosing among executable file typologies: *.exe, or *,dll. Its ProgId is "ExecutableName..Optimize", where ExecutableName is the name of the component.
The optimizer is specific to each application, therefore we talk about custom Optimizer. On his turn the custom Optimizer uses a module of standard optimizer, available with the installation of TpaCad, that is the Optimizer of the piece.
The custom Optimizer generates the necessary information for the start-up of a program, in the format and according to the specifications required from the individual application.
The Optimizer of the piece supplies to a generic custom Optimizer the information of a program after each processing, such as development of subroutines, solution of parametric programming, application of the logical conditions. The information processed in this way are organized in a file of in a storage area (called Piece matrix). This is a set of data organized in a matrix (structure: rows * columns), defining the program as a whole.

Of the custom optimizer following methods are used:

- **long Settings (string newFolder, string newMachine, string szPathConfig, string szArgs)**
  this method is invoked to manage the customizations provided by the module. This method can be implemented if a record for the customization of the optimisation module is available with an automatic reading process of the same at each optimisation request.
  Method return value: 0 if the result is positive, otherwise error number.

| | |
|---|---|
| string newFolder | Tpa.ini - file search folder (if the string is empty, the default folder is taken) |
| string newMachine | name of the machine (section in Tpa.ini file section, if the string is empty, the name by default of the machine is taken) |
| string szPathConfig | folder in which the functioning assignments can be read/written (the example for DXF converter is the folder from which the INI files of assignments of the conversion customisations can be read) |
| string szArgs | string of arguments.<br>Format and meaning of the fields:<br>• '/': field name<br>• space: optional separator<br>• the string is case-insensitive<br>"**/key**=.." =.." >0 enables the advanced layer, if used by the import module (example for DXF converter: assignment of workings from layers and/or blocks)<br>"**/lng**=.."current language (three letter abbreviation. Example: "ITA", "DAN")<br>"**/lngdef**=" default language (three letter abbreviation from: "ITA", "ENG", "FRE", "GER", "SPA")<br>"**/**…"assigned arguments in the field Arguments follow |

- **long SettingsArgs (bool bMode, ref string szArgs)**
  this method is invoked to manage the customizations run by the module. We recommend to implement this method only if a customization through ***Settings*** method is not managed.
  Method return value: 0 if the result is positive, otherwise error number.

| | |
|---|---|
| bool bMode | ******FREE* |
| string szArgs | The string returns the default settings managed by the module (with regard to the format, reference is made to previous information). If, for instance, the module interprets two numeric customizations transmitted as "/aaa=… bbb=…", the method must return the string containing the values that are interpreted in case of non-transmitted arguments (example: "/aaa=1 bbb=0"). |

- **void UpdateAll(string stFolder, string stMachine)**

the method is invoked if in the TpaCAD environment a selection of Machine change has been assigned, Return value of the method is ignored.
If necessary, the call should be also provided in the internal component Piece Optimizer.

| | |
|---|---|
| string stFolder | this is the folder where the Tpa.ini file is searched (if the string is empty, the default folder is considered) |
| string stMachine | this is the name of the machine (section defined in the Tpa.ini file If the string is empty, the name of the default machine is considered) |

- **void InitMode(string szSetting, string szSettingRun, string szSettingMatrix)**
  It initializes the generic functioning modes of the component, Return value of the method is ignored.
  If needed, the call should be provided also for the internal component Piece Optimizer. The meaning of the assigned fields, as recorded here, corresponds with the functioning specifications of the Piece Optimizer.

| | |
|---|---|
| string szSetting | string formatted to define generic settings. Format and meaning of the fields (fields headed by '//'; optional separator: space; to interpret: case-insensitive; items not assigned assume the value by default; in case of repeated items, the first occurrence is taken on)<br>"**//enconv**=.."if > 0 enables the direct import of a file from another format 0=disables=default). In case of enabled import, the component read the assignments concerning the import modules as defined in the the TpaCAD.ini file. TpaCAD gives the substring "//enconv=0"<br>"**//entools**=..">  0 enables the application of global processing tools, after an import from another format (0=disable=default). If enabled, the component reads the relative assignments as they are defined in the file TpaCAD.ini or TpaCAD on page **Customize->Environment->Import the format**. TpaCAD gives the substring "//entools=0"<br>"**//enattr**=.." >0 enables the management of the tools (the tooling selection is managed by the technology component) (0=disable=default). TpaCAD gives an enabling substring which corresponds to the control identification of the Technology Toolings. |
| string szSettingRun | formatted string of the piece compiling options. For the format and the meaning of the fields, reference is made to the specific documentation of the piece Optimizer.<br>TpaCAD gives an empty string (=""), which imposes the use of settings configured on page **Configure TpaCAD->Open and Save->Assign in Piece Matrix** . |
| szSettingMatrix | formatted string of the the creation option of the matrix piece. TpaCAD gives an empty substring (=""). Format and meaning of the fields (see szSetting):<br>"**//nmode**=.."if > 0 defines the creation mode of the matrix piece (default=0). The meaning of the value is:<br>• 0: creates a bidimensional matrix (row and column)<br>• 1: creates a three dimensional matrix (it shows also the typologies of the parameters)<br>"//**mxcolumn**=.."if > 0 rrepresents the number of colums for each matrix row (max 100). (The default value is the value set on page **Configure TpaCAD->Open and Save->Assign in Matrix Piece** .<br>"**//sideempty**=.."if > 0 records the heading also of the empty faces (default: 0)<br>"**//cstsector**=.."if >0 records also the custom sections (default: 1)<br>"**//varsr**=.."if >0 records also the "r" variables (default: 0) |

- **short Compute(string szPathFile, string szArgs, ref string szModeExe, ref string szDims, ref string szVarsO, ref string szVarsV, ref string szVarsR, ref string szCstSpec, ref string szCstOpti, ref string szCstLink, ref string szCstInfo, string szEsclude, ref string szError)**
  It requires the optimisation of the program indicated. If needed, the call should be provided also for the internal component Piece Optimizer. Method return value: 0 if the optimizer succeeded, otherwise it returns the number of the error.

| | |
|---|---|
| string szPathFile | path + name of the file of which the optimisation is required. |
| string szArgs | string of arguments. The string gives information to the custom Optimizer. Format and meaning of the fields:<br>• '/': field name<br>• space: optional separator<br>• the string is case insensitive (no difference between lowercase caracters and uppercase characters)<br>• for the items that have not been assigned, the default value is taken<br>• in the case of repeated items, the first occurrence is considered. |

| | |
|---|---|
| | ”**//mx**=..”: 0 requires a minimal optimization (corresponding to the command of program saving); >0 requires a complete optimization (corresponding to the command of Program Optimization)<br>”**//store**=..”:=0 does not store the matrix piece in a file: >0 stores the matrix in a file<br>”**//tipstore**=..”: storage file format of the matrix: =0 ASCII; greater of binary 0<br>“**//…….**” : assigned arguments in the field Arguments follow |
| ref string szModeExe | execution mode and additional parameters in execution |
| ref string szDims | dimensions |
| ref string szVarsO, szVarsV, szVarsR | “o”,“v”, “r” variables |
| ref string szCstSpec | assignments in Special settings |
| ref string szCstOpti | assignments in the unit Optimization settings |
| ref string szCstLink | assignments in Constraint unit |
| ref string szCstInfo | assignments in Additional info unit |
| ref string szEsclude | exclusion for the ‘L’, ‘K’, ‘K1’, ‘K2’ fields. |
| ref string stError | if not equal to 0 it returns an error message, otherwise it returns 0 |

The assignments for the optimization required (szModeExe, szDims,….. szCstInfo) follow the same format of the component Piece Optimizer .
In case of call from TpaCAD, further to a requested Optimization, the same arguments are interpreted also when they return, and it is possible to change the same settings of the current program. In this case the strings should be returned headed by "**>>**" and after them the assignments to make should be listed with the same formalism assigned in entry.

**Example:**

- in entry **szModeExe =“#0=0 #1=2”** normal execution mode; workspace 2
- in exit **szModeExe =“>> #0=1”** assigns x specular execution mode

***WARNING:*** Seemingly locked units shall not be executed.

***WARNING:*** No assignments will be carried out in case of *Essential* functionality.

- **void Dispose()**
  The method is invoked before releasing the class instance, to release the managed and the unmanaged resources, such as handle, connections to database,...).

**Optimization preview**

The **Component of Preview optimisation** according to the specific request of the application this option can achieve a particular representation of the program reaching the executive phase.
**WARNING*:*** this unit is not available in case of *Essential* functionality.

Of the control we use the following method:

**long Settings (string newFolder, string newMachine, string szPathConfig, string szArgs)**
the method is invoked to manage the customizations provided by the module. This method can be implemented if a record for the customisation of the optimisation module is available with an automatic reading process of the same at each optimisation request.
Method return value: 0 if the result is positive, otherwise error number.

| | |
|---|---|
| string newFolder | Tpa.ini - file search folder (if the string is empty, the default folder is taken) |
| string newMachine | name of the machine (section in Tpa.ini file section, if the string is empty, the name by default of the machine is taken) |
| string szPathConfig | folder from which the functioning assignments can be read/written (the example for DXF converter is the folder from which the INI files of assignments of the conversion customisations can be read) |
| string szArgs | string of arguments.<br>Format and meaning of the fields:<br>• '/': field name<br>• space: optional separator<br>• the string is case-insensitive |

| | |
|---|---|
| | "**/key**=.." =.." >0 enables the advanced layer, if used by the import module (example for DXF converter: assignment of workings from layers and/or blocks)<br>"**/lng**=.."current language (three letter abbreviation. Example: "ITA", "DAN")<br>"**/lngdef**=" default language (three letter abbreviation from: "ITA", "ENG", "FRE", "GER", "SPA")<br>"**/**…"assigned arguments in the field Arguments follow |

- **long SettingsArgs (bool bMode, ref string szArgs)**
  the method is invoked to manage the customizations run by the module. We recommend to implement this method only if a customization through ***Settings*** method is not managed.
  Method return value: 0 if the result is positive, otherwise error number.

| | |
|---|---|
| bool bMode | ******FREE* |
| string szArgs | The string returns the default settings managed by the module (about the format, reference is made to previous information). If, for instance, the module interprets two numeric customizations transmitted as "/aaa=… bbb=…", the method must return the string containing the values that are interpreted in case of non-transmitted arguments (example: "/aaa=1 bbb=0"). |

Of the control we use the following method:

- **long PreviewFromAlb(string szFileMatrix, string szArgs, ref string szError)**
  Method return value: 0 if the format is recognised as valid, otherwise error number

| | |
|---|---|
| string szFileMatrix | pathname of the file in matrix-piece format |
| string szArgs | string of arguments. The string gives information to the custom control. |
| ref string szError | Error message |

In the preview control is not assigned, TpaCAD uses an automatic display, that corresponds to the processing started for the normal optimisation of the program.

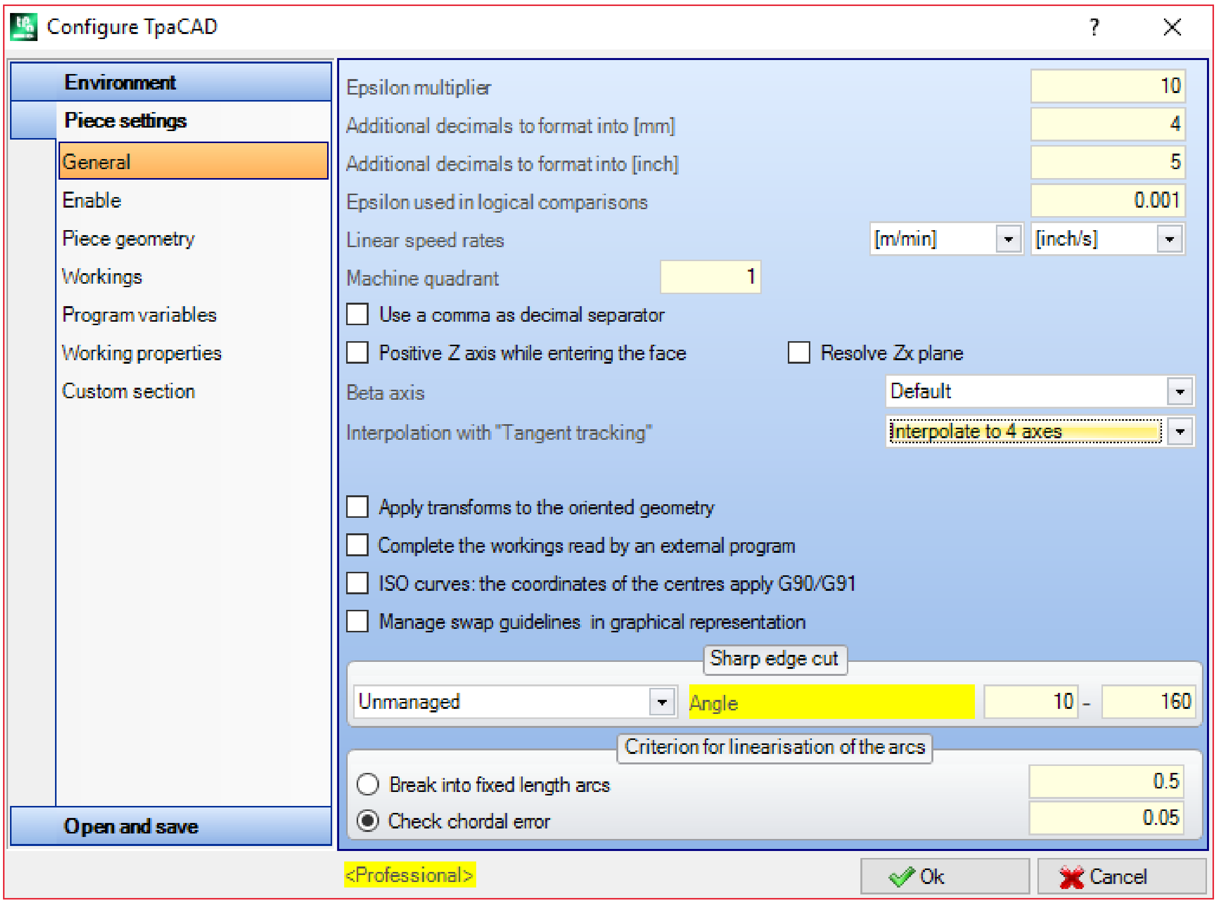
**Nesting**

In this case it enables the functionality of **Nesting** managed in TpaCAD.
**Password level:** select the level from which the settings of the functionality can be modified.

To describe the functionality of **Nesting,** please read the documentation dealing with the topic.

# 1.2 Piece Settings

## General



Configure TpaCAD

Epsilon multiplier: 10

Additional decimals to format into [mm]: 4

Additional decimals to format into [inch]: 5

Epsilon used in logical comparisons: 0.001

Linear speed rates: [m/min] [inch/s]

Machine quadrant: 1

□ Use a comma as decimal separator

□ Positive Z axis while entering the face □ Resolve Zx plane

Beta axis: Default

Interpolation with "Tangent tracking": Interpolate to 4 axes

□ Apply transforms to the oriented geometry

□ Complete the workings read by an external program

□ ISO curves: the coordinates of the centres apply G90/G91

□ Manage swap guidelines in graphical representation

Sharp edge cut: Unmanaged, Angle 10 - 160

Criterion for linearisation of the arcs

○ Break into fixed length arcs 0.5

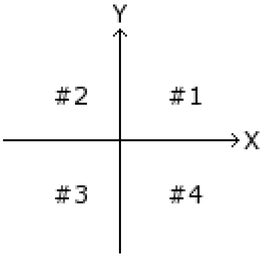
● Check chordal error 0.05

- **Epsilon multiplier:** numeric value, which multiply the internal resolution (=0.001 mm), in order to determine the actual geometric resolution applied to the piece (indicates as epsilon), then epsilon=0.001 * multiplier. The resolution epsilon is used to differentiate geometric situations of comparison and not as a precision in calculating the resolutions. For instance, it is used to determine:
  - if two positions coincide or if they are distinct. A P2 pointcoincides with another P1 point, if it falls within the circumference with centre in P1 and radius equal to epsilon
  - if the programming of an arc should be considered valid. the value of the two radii calculated as the distance of the centre respectively from the starting and the ending point cannot differ by more epsilon, otherwise a programming error is reported;
  - if the programming of an arc needs to interpret an arc or a circle. If both the points at the beginning and at the end of the arc are separated by less than an epsilon, the programming interprets a circle.

  Values (with comma) between 1.0 and 100.00 are accepted.. The resolution definite epsilon (=0.001 * multiplier) has a corresponding variable argument of parametric programming (eps). It is necessary to set the epsilon multiplier significantly; in this way it will lead neither to rigid programmed values (see cases of errors of arc programming) nor to a too much large range of tolerance (where some objectively "distant" points are seen as coincident).
- **Additional decimals to format into [mm]:** set the number of figures to put after the comma in the conversion of the working parameters in ASCII format. This setting is applied to programs with measure unit in millimetres [mm]. It accepts a value from 3 to 6. ***WARNING:*** if the **epsilon multiplier factor** is less than 10.0, the value is from 4 to 6.
- **Additional decimals to format into [inch]:** set the number of figures to put after the comma in the conversion of the working parameters in ASCII format. This setting is applied to programs with measure unit in inches [inch]. It accepts a value from 3 to 6. ***WARNING:*** if the **epsilon multiplier factor** is less than 10.0, the value is from 4 to 6.
- **Epsilon used in the logical comparisons :** it sets the epsilon that is applied to calculate a logical comparison. The value is used in the parametric programming of logical functions (examples: ifelse, ifcase) It accepts a value from 0.0 to 0.001, with 6 decimals figures max.
- **Linear speed rates:** it sets the unit programming of the linear speeds, applied to pieces with units of measure in mm [mm] or inches [inch]. For pieces with units of measure [mm], possible selections are:

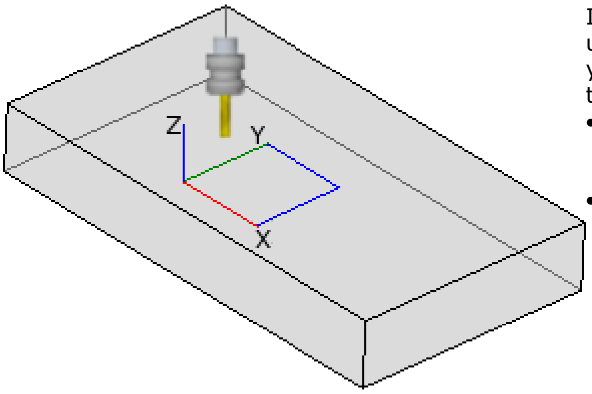
[mt/min] or [mm/min]. For pieces with units of measure [mm], possible selections are: [inch/sec] or [inch/min]. The setting has a corresponding variable argument of parametric programming (sysfeed). The unit of measure and the unit of measure of the plant parameters must be uniform. The assignment has a correspondence of interpretation with the module of Optimization and/or with the plant cycle PLC cycle) In this sense it should be arranged during the design phase of the plant.

- **Machine quadrant:** number of the XY quadrant of the machine. It may take any number from 1 to 4. The meaning can be the same one, as planned in the previous applications (Edicad), so it declares how the XY plan of the machine is managed.



The setting has a corresponding variable argument of parametric programming (sysquad).

- **Use the comma as a decimal separator:** if enabled TpaCAD uses the comma as separator character of the decimal figures (example: "12,7"), otherwise it uses the dot character (example: "12.7"). This choice affects every direct assignment of decimal values and each assignment of parametric type: the program variables, the processing parameters, the fields and not only the numeric ones in windows setting tools, for which both comma and point are considered valid separators, provided that they are not used together.
- **Positive Z-axis while entering the face:** if selected, the Z - coordinate at the beginning of the face should be set with a positive value.



In the figure on the left the Z-axis is represented upwards so, the tool enters the piece from above. If you wish to program a drilling with entry depth into the face equal to 10 mm:

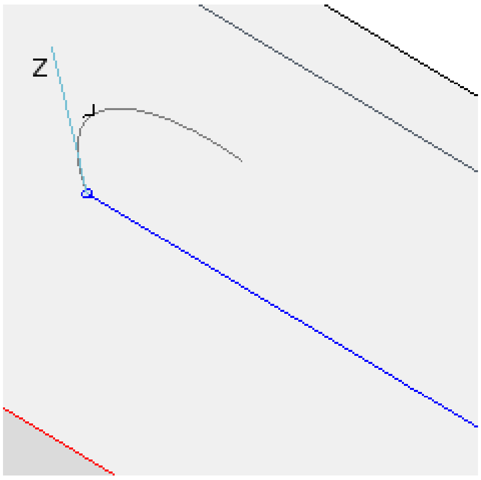
- if **Positive Z-axis entering the face** is not selected, you should set a a negative coordinate (Z=-10)
- if **Positive Z-axis entering the face** is selected, you should set a positive coordinate (Z=10)

This setting has a corresponding variable argument of parametric programming (sysz).

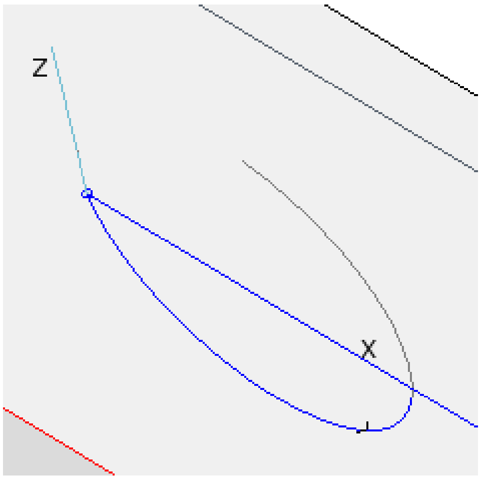
***WARNING:*** do not confuse this option with the assignment of the z-axis above the face, that shows which part enters the tool to work in the face (in the figure: Z direction).

This is a setting which changes the interpretation of a program. The macros used should be written with the same sitting, or they should verify the option and adapt accordingly.
If the option is enabled during the optimization of the program, the sign of the coordinates on the Z-axis can be reversed, in such a way as to create univocal information. In the case of assignation of custom parameters that can be used as coordinates on the axis of face, we recommend to set the relative field while defining the parameter in the database of the workings. The option is interpreted also in the module of Optimization and in the plant cycle (PLC cycle). In this sense it should be arranged during the design phase of the plant.

- **Resolving Zx-plane:** if selected, the arcs on the xz-plane of the face are interpreted with a solution of Zx primary plane.



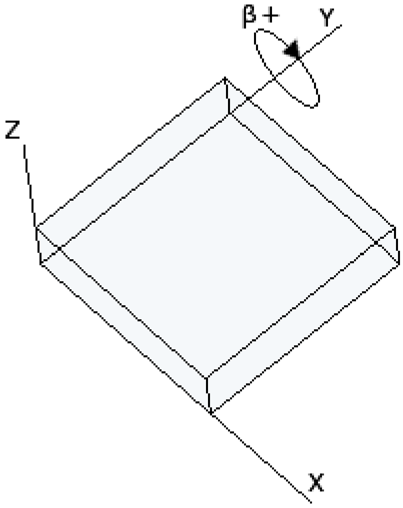
**Case of non-enabled Option Resolving Zx plane**
In the figure the case of a quarter of circle assigned on the xz-plan is represented with solution for the case of CLOCKWISE rotation.
The arcs on the xz-plane of the face are interpreted with solution or primary Xz-plane (X axis as primary axis of the plane). This setting derives from the need to maintain compatibility with the previous products; however, it does not correspond to the solution generally adopted in CAD systems.



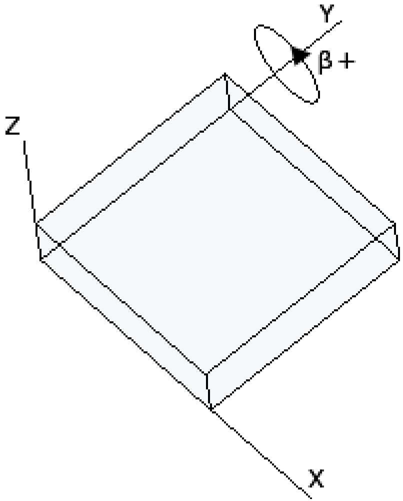
**Case of Option Resolving Zx plane enabled**
In the figure the same arc is represented as in the previous case, now with solution of primary Zx-plane (Z-axis as primary axis of the plane). In this case the rotation is inverted.
This setting corresponds to the solution normally used in CAD systems.

The setting has a corresponding variable argument of parametric programming (sysxz).

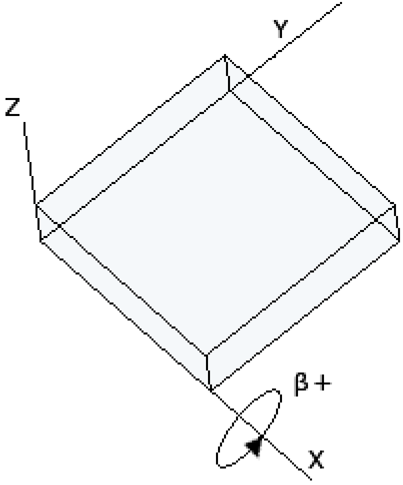
- **Pivoting axis:** it sets the functioning mode of the pivoting axis. The available typologies are three:



- **Default:** the pivoting movement occurs around the Y axis of the absolute Cartesian coordinate system. In the figure the option for the (+/-) rotation of the axis is selected.



- **Invert:** the pivoting movement occurs around the Y axis of the absolute Cartesian coordinate system, but the direction of the rotation is inverted. In the figure the option for the (+/-) rotation of the axis is selected.



- **Rotate:** the pivoting movement occurs around the X axis of the absolute Cartesian coordinate system (the rotation axis is rotated of 90° with respect to the default setting). In the figure the option for the (+/-) rotation of the axis is selected.

The setting has a corresponding variable argument of parametric programming (sysbeta).
This selection affects:

- the graphical representation of the oriented profiles (inclined setup or blade)
- the use in the parametric programming of the functions geo[alfa;..] and geo[beta;..].

- **Interpolation in "Tangent tracking" :** sets the operating mode of the interpolation on rotary axes in *Tangent tracking* mode. This selection influences the execution of oriented profiles ant the available options are four:
  - *Not managed*: this application does not manages the *Tangent tracking*. You cannot apply for the repositioning of the tool carrying out a profile
  - *Interpolate to 4/5 axes* : the selection requires that both rotary axes are repositioned in the execution of the profile. This selection does not automatically determine that an oriented profile interpolates on both rotary axes, but it enables the possibility of choice for the workings involved
  - *Interpolate to 4 axes*: this selection excludes the repositioning of the pivoting axes (beta), but it enables the rotation axis (alpha) to be repositioned. This selection excludes the possibility of choice at the programming level.
  - *Interpolate to 4/5 axes*: the selection requires that both rotary axes are repositioned in the execution of the profile, without any possibility of choice at the level of programming

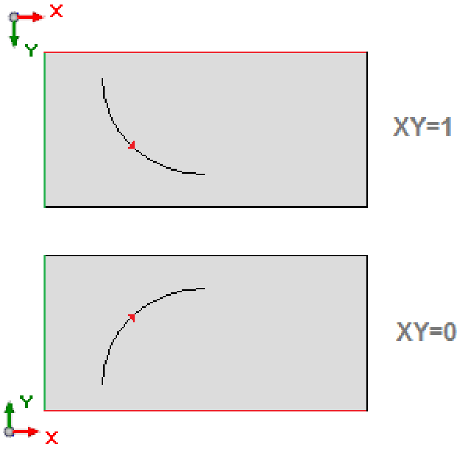
  Interpolation possibilities depend on the physical machine configuration and of the installed functionalities.

- **Apply transforms to the oriented geometry:** select to modify the assignment of the axis (alpha, beta) in the workings of the oriented setup, when the tools of rotation, mirror and profile inversion are applied. Anyway, the transform can be applied only in the event of a plane face, that is not curved or assigned as a surface.
- **Complete the workings read by an external program:** if this option is selected, when you read a program in TpaCAD format that is created in an environment external to TpaCAD, it integrates the programming of the workings by the default settings.
  The origin of a program is recognized by interpreting the head row of the program. More specifically: a progressive number concerning the database of the workings (it is not assigned or its value is 0) must not be deected.

It is customary that the programs generated by an import process (ex.: by DXF or ISO format) verify the above mentioned situation. If an import assigns the geometric information only (ex.: coordinates of the application), all the default settings, configured for the technological parameters, can be automatically set, straight while opening the program.

- **ISO Curves: the coordinates of the application centres apply G90/G91:** the setting concerns the ISO curve reading process from the program and the interpretation of an arc coordinates:
  - if selected, the coordinates are interpreted incremental or absolute according to the settings of G90/G91;
  - otherwise, the coordinates are always interpreted incremental with respect to the arc beginning point.
- **Manage exchange guidelines in the graphical representation:** the selection of the field concerns the activation of specific exchange, with reference to the 2D graphic representation of a single face.

A typical example is the selection of the direction of an arc rotation:



Asides, the face 1 is represented (top face) with two different assignments of the YX plane (see paragraph: Piece Geometry), marked as:
XY=0: the origin of the face is bottom left
XY=1: the origin of the face is top left
The 2D graphic representation of the face is assigned corresponding with the face geometry.

In both cases you represent an arc, that is "seen ”
CLOCKWISE in case of
. XY=0
CONTERCLOCKWISE in case of
. XY=1

If the selection is active, if you program, for instance, the arc with A01 working, the list selecting the rotation sense is adapted so that the rotation chosen corresponds to the *rotation seen* on the arc. That is:
CLOCKWISE / COUNTERCKLOCKWISE
. for the case XY=0;
CLOCKWISE / COUNTERCKLOCKWISE
. for the cases XY=1;

**WARNING:** the arc is always programmed with CLOCKWISE rotation, in line with the pure Cartesian system of the coordinated axes (case YY=0),
while the options in the list are only swapped. For example, if you wish to program the rotation in parametric form, CLOCKWISE is alwayus 0 and COUNTERCLOCKWISE is always 1 (or ≠ 0).

Similar aids are used in other remarkable assignments of workings:
tool compensation side in setup working (options LEFT/RIGHT are exchanged)
the option to mirror by applying the subprogram (MIRROR X/ MIRROR Y).

**WARNING:** these message exchanges are possible, if the corresponding function in the working database is assigned to the corresponding parameters (see TpaWorks manual: Working parameters, Auxiliaries, Enhanced features, Function of list messages swap).

**WARNING:** The selection is provided only to keep a view compatibility for applications already controlled through previous versions of the CAD of TPA: TpaEdi32 or Edicad. However, we would like to point out that the developments of 3D rendering of TpaCAD make this functionality obsolete, or even ineffective in many cases.

- **Sharp edge cut:** select to enable the rise functionality on the edges of a profile, worked by a conic tool, in order to perform prominent edges according to the maximum limit permitted by the tool dimensions. A typical application is used in frame machining. This selection is managed on three items:
  - **Not managed** : excludes the functionality
  - **V-Cutter only**: enables the functionality for V-cutters only
  - **Managed always** : enables the functionality also for non-conic tools.

To complete the functionality, set the fields associated to the item **Minimum angle**: assigns the (minimum or maximum) angle between two profile segments to determine the cut sequence. Set

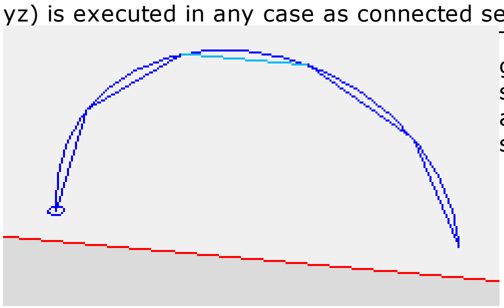
respectively a value between (10 and 90)° and between (90 and 170)°. The value 90 represents the typical situation of frame machining, whose segments are perpendicular to each other.
Given two linear segments and the Angle A° (whose value range is between 0° and 180°) the angle is valid, if the situations are verified as follows:

- A° >=Minimum angle and
- A° <= Maximum angle

## Criterion for linearisation of the arcs

- **Break into fixed length arcs:**if selected, it enables arcs to be split and to be linearised into segments of assigned length. It also possible to set the sampling length of the arcs. Typical values can also be of some mm (2 mm, 3 mm).
- **Check chordal error:** if selected, it enables arcs to be split and to be linearised into segments assessing a maximum chordal error. It is possible to set the allowable chordal error value. We recommend to assign a value not less than 1 mm. Default is 0.05 mm. The values must be set in the unit of measurement of the configurations. Values included in the interval of 0 – 100.0 mm are accepted. This parameter is used to
  - represent graphically and file assigned arcs in a plane different from xy
  - represent graphically arcs in three-dimensional mode.

  An arc assigned in a generic xyz-plane, non coinciding with one of the three coordinated plans (xy, xz, yz) is executed in any case as connected series of line segments.

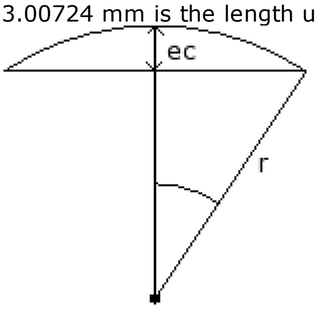


This figure represents an example of arc assigned in a generic xyz plan. This polygonal chain is obtained by splitting the arc into sub-arcs with maximum assigned lengtł and by joining the endpoints of each sub-arc with a line segment.

The excessive assigned length in the figure is useful to show the technique developed to split an arc. The fragmentation applies also the distribution of the residues, as emphasized in the figure. Here also a numerical example:

- the length of the original arc is 415 mm
- the required split length is 3 mm

We have: 415/3= 138.3333 sub-arcs whose length is 3 mm. If we distribute the residue: 415/138= 3.00724 mm is the length used to split into 138 sub-arcs.



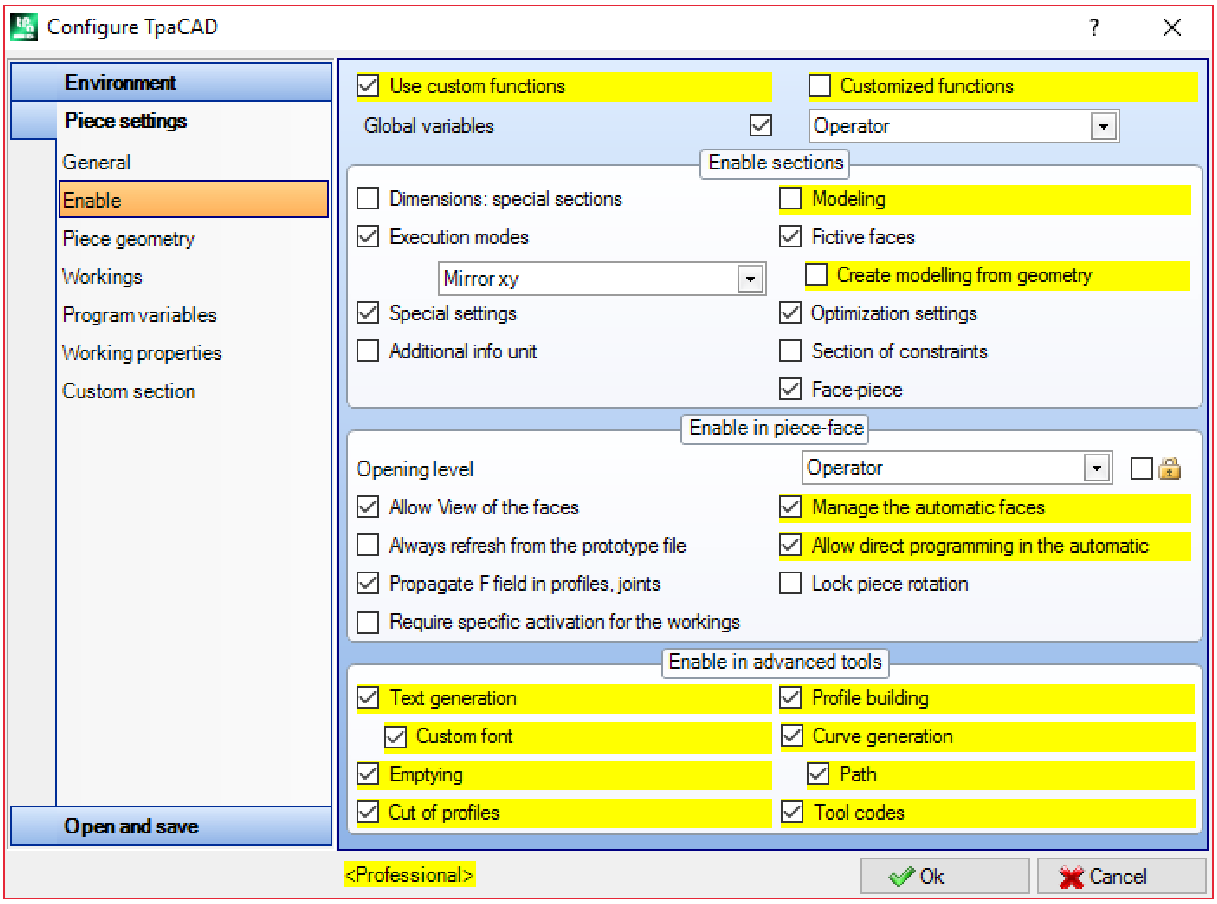
In case of selection of **chordal error control** , the figure shows the geometric meaning assigned to the value set (ce). A typical chordal error is 0.05 mm. Splitting an arc according to the criterion of the chordal error determines samplings of length, which changes according the radius of the arc: if the radius increases the length of the segments increases.

The splitting of an arc occurs by applying some general criteria, which can use a different value from the one set here; the aim is to ensure a minimal precision and to avoid excessive splitting. Let us see the details:

- a minimum absolute value is anyway applied. The minimal length of splitting is equal to 10* epsilon, while the minimum chordal error is equal to epsilon;
- using the chordal Error for each splitting a maximum chordal error equal to 50% of the radius of the arc is accepted;
- for an arc a sampling number is resolved anyway and it is not lower than the entire fractions of 45° (of the arc dimension);
- for each splitting precise limits are accepted for the sampling angle, that is calculated = 1°, maximum = 45° .

Splitting an arc, programmed in a curved face, applies a further calculation in the event of selection of **check the chordal error**: arc splitting is determined by considering the smaller one between radius of the arc and radius of curvature of the face.

## Enable



☑ Use custom functions
□ Customized functions
Global variables ☑ Operator

Enable sections

□ Dimensions: special sections
□ Modeling
☑ Execution modes
☑ Fictive faces
Mirror xy
□ Create modelling from geometry
☑ Special settings
☑ Optimization settings
□ Additional info unit
□ Section of constraints
☑ Face-piece

Enable in piece-face

Opening level: Operator □
☑ Allow View of the faces
☑ Manage the automatic faces
□ Always refresh from the prototype file
☑ Allow direct programming in the automatic
☑ Propagate F field in profiles, joints
□ Lock piece rotation
□ Require specific activation for the workings

Enable in advanced tools

☑ Text generation
☑ Profile building
☑ Custom font
☑ Curve generation
☑ Emptying
☑ Path
☑ Cut of profiles
☑ Tool codes

**Use Custom functions:** if selected, it enables the use of the custom functions in the parametric programming. Custom functions are read by two files: (tpacadcfg\CADFUN.DEF) is the file of the basic functions; (tpacadcfg\custom\CADFUNCUST.DEF) is the file of the functions developed for the specific application. The selection of the item enables the use of the custom functions, bringing the upload of the basic functions.
**Customized functions:** if selected, it enables the use of the custom functions developed for the specific application. The selection is significant only if the previous item is selected.
If both files are read, the functions read are unified and univocally recognized by the name (WARNING: not by the number) of the functions and up to 100 functions. More specifically, if a function is assigned with the same name in both files, the function from the customized file will be considered valid. As for the writing and the maintenance of the custom functions, see the next paragraph.
If you exclude the use of the custom functions, also the workings, which are related to the global Functions, are excluded,
The custom functions are available in <Professional> mode only.
***WARNING:*** The chosen type of setting modifies the interpretation and the execution of a program.
**Global variables:** if selected, it enables the assignment and the user of Global Variables in the parametric programming. They are strictly numerical variables, they are no more than 300 and can only be recalled by name.
It is possible to set the Layer on which the assignment of the variables can be enabled.
As for the writing and the maintenance of the Global variables, see the next paragraph.
***WARNING:*** The chosen type of setting modifies the interpretation and the execution of a program.

### Enable sections

- **Dimensions: special sections**: if enabled for the dimensions of the piece, this option manages the possibility to set the Diameter of the section alternatively to (Height + Thickness). This command is useful if you need to program cylindrical bars: the Diameter is automatically recorded both in H (Height) and S (Thickness).
- **Execution mode** if enabled, the section Execution mode in the page Dimension is managed. The next control of the list allows you to enable the available items to select the execution mode in the same section of a program. The available items are four: Normal, X mirror, Mirror, x+y mirror. The selected item settles the maximum selection available. Selecting for instance *x Mirror*: while developing the

program you can choose between 2 items only: *Normal* and *x Mirror* . Chose the first item (Normal) to exclude the corresponding field management in the program development.

- **Special Settings**: if enabled, the page of Special Settings can be managed.
- **Additional info unit:** if enabled, the page Added Info can be managed ***WARNING:*** The selection is not available in case of *Essential functionality.*
- **Modelling:** through this option the Modelling section is enabled (please, read the specific documentation in the manual of TpaCadMD). The use of the functionalities requires a specific activation from HW key besides the *Professional* functionality.
- **Fictive Faces:** if enabled, the page of Fictive Faces can be enabled.
- **Create modelling from geometry:** through this option you can control the modelling elements directly while creating the fictive faces (please, read the specific documentation in the manual of TpaCadMD). The use of the functionalities requires a specific activation from HW key besides the *Professional* functionality.
- **Optimization settings:** if enabled, the page Optimization Settings can be managed
- **Section of constraints** if enabled, the page Section of constraints can be managed ***WARNING:*** The selection is not available in case of *Essential functionality.*
- **Face piece:** if enabled, Face piece can be managed Face piece is a particular face without its own geometric identification. It can be stated that it represents the piece as a whole, including all faces which characterize it. The face-piece program allows workings to be directly assigned to different faces of the workpiece in a single program list. For the workings it is also managed here the property that assigns the application face (except for the logical IF workings, assignment of variables, ..), shown as "F". A program written in piece-face cannot be used as a subroutine. For this reason the face-piece is made available only in case of a piece with program typology. If the end user of TpaCad needs to write subroutines, to apply them, he must first program workings directly in the views of the enabled faces. The Face piece offers specific possibilities, which are not available on other faces:
  - a peculiarity is related to the executive Sequences. The working list of Face-piece assigns directly the execution order (the sequence in assigned in an implicit way) and corresponds to the order of the programmed list.
  - It is possible to create faces of variable geometry directly during the programming. This is the management of the Automatic Faces .

***WARNING***: all activations of Sections modify the interpretation and the record of a program. More specifically, a non-managed section is not read or written in the programs.

<u>**Enable in piece-face**</u>

- **Opening level** : this option assigns the level from which you can open and modify the face-piece. Setting at a layer higher than *Operator* allows, for example, using the piece-face to display the fixed overall dimensions, assigned in the prototype file; in this event, the opening Layer of the prototype file must be higher than *Operator*.
- 🔒 **Edit status:** select the field to prevent piece-face to be accessed. If the selection is active, piece face is foreclosed regardless the **Opening level** set.
- **Allow View of the faces:** if enabled for the piece with program typology, it is possible to program directly on both the single (real and fictive) faces and on the face-piece. If the option is not enabled for the piece with program typology, it is possible to program the workings <u>in</u> Face Piece only. In this case the program list is only one. If you upload programs, whose workings are programmed on other faces, these latter are set to zero. In cases of subroutines and macros the direct programming is always managed on the individual faces of the the workpiece and not on Face-piece.
- **Always update from the prototype file:** select the field to request the assignment of the Piece-face from the prototype file, also when the program is opened. The application of the selection depends on following verifications:
  - **Allows view of the faces** : active;
  - the file must be of program type and not a subprogram or macro type.

  If applied, this assignment also concerns the format settings and the optimisation of a program.
- **Propagate F field in profiles, joints:** if enabled, the identification of a continuing profile (due to the condition of Point Hook) prevails on the programming of the "F" field. if enabled, the identification of a continuing profile (due to the condition of Point Hook) prevails on the programming of the "F" field.
  If the field is enabled:
  - if a profile segment (arcs and lines) opens a profile, the original programming of the F field is maintained
  - if the segment is part of a profile, the value of the F-field of the previous segment is propagated
  - in a setup or complex working with request for point hook, the F-field is propagated from the previous working.

  If the field is not enabled:
  - the propagation of the "F" field from a previous segment is never applied
  - to evaluate if a profile segment (arcs and lines) opens a profile you must take into account the "F" field assignments (relative to the current and the previous working). Different settings determine in any case the interruption of the profile
  - in the case of setup or complex working **requiring** a point hook , the "F" field is not propagated from the previous working. The hook point is not executed, if the value of the "F" field of the previous working is not equal to the value of the "F" field of the working requiring the hooking.

  - **Manage the automatic faces:** it enables the management of the automatic faces. The automatic faces visibility is limited to the face-piece and their numbering is managed in automatic and progressive way from 101 to 500.

  The creation of an automatic face enables the following application of workings to it, always and only in face-piece programming. On the contrary it is not possible:
  - to access directly to a view of an automatic face outside of the face-piece
  - to create and/or assign workings to an automatic face from a face different from the piece-face

  An automatic face cannot be directly selected by the number automatically assigned to the face, always and only the last assigned face is accessible

  Therefore, the mechanism of use of the automatic faces is the following:
  - ...
  - assigning an automatic face (automatic number: the first one free, example 105);
  - workings are applied to the last created automatic face (last created ->105)
  - ...
  - assigning an automatic face (automatic number: the first one free, example 106);
  - workings are applied to the last created automatic face (last created ->106);
  - ...

  In any point of the piece-face program is then available only a specific automatic face: the last created before the selected working. In this sens the general expression  Application in automatic faceis used. If the selection of Automatic faces is active, la relative authorization shall be available in <Professional> mode only.

- **Allow direct programming in the automatic Faces:** If enabled, it is possible to apply workings on an automatic face from the main list of the program in face-piece. If not enabled, workings can be applied on an automatic face only from subroutines (or macros) applied on face-piece; in this way the control of the automatic faces remains completely transparent for the end user . This choice must be supported by a working database, which does not take into account the needs of managing the automatic faces. The selection is significant only if also the management of automatic faces is enabled.
- **Lock piece rotation** it is available for use in application that program in face-piece only (read: *Allow View of the faces* not enabled, even if no control is applied for this purpose) and if, for example, the only face 1 is enabled. If enabled, it works on *Overall View* and in *View of face-piece*, making the commands for the graphic rotation of the piece inactive. If the selection is active, we recommend you to assign the information *Preview* in the custom section *Special settings* to oblige the opening of the piece in a particular view.

  For example, let us set in prototype file the information*Preview*  in the value that corresponds to the view from the top face (field value: 1) a new program will always open with View from the top and it will be possible to change the detailed view (right side, left side, front side, rear side) but not to rotate it as much as one likes.
- **Require specific activation for the workings:** : if enabled, through this option you can insert in the face-piece only the workings enabled in a specific way. It can be used for example for the placement of constraints (rails and/or suckers) by means of provided workings.

**Enable in Advanced Tools**

- **Text generation:** if enabled, both Text generation tool and all the complex workings, which apply a Text generation are managed. If not enabled, the tool and all the complex workings, that apply a text generation, are disabled.
- **Custom font:** if enabled, it manages the of the texts from Custom font file (the activation is real only if the option *Text generation*  is enabled). Anyway, the advanced tool of Create font from geometry depends also on the section of **Power user**

- **Emptying:** if enabled, both the Emptying tool and and all complex workings, which use an Emptying process, are managed. If not enabled, the tool and all the complex workings, that apply an Emptying process, are disabled.

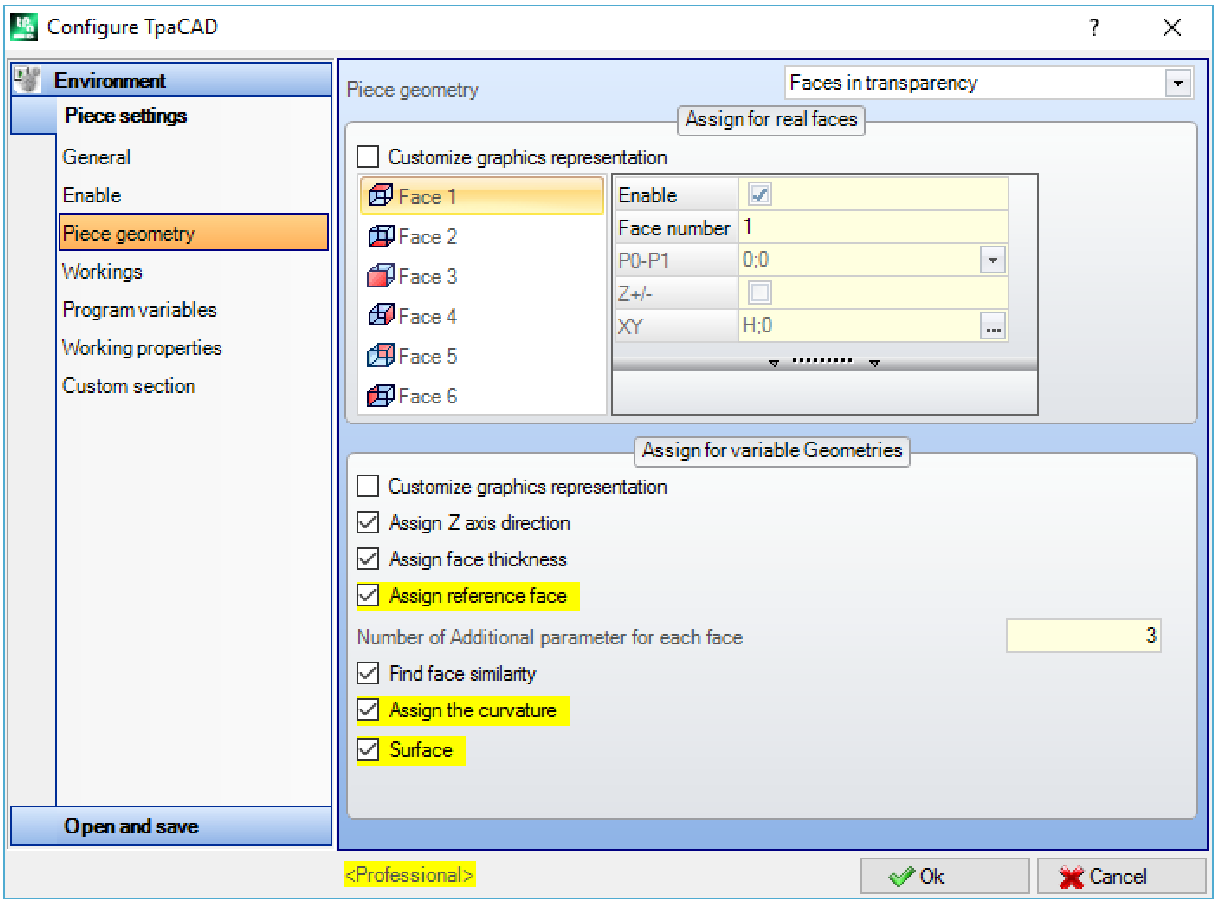
- **Cut of profiles:** if enabled, the tool Cut of profiles can be managed. .

- **Profile building:** if enabled, the Profile building tool can be managed. .

- **Curve generation:** if enabled, both Spline curve generation tool and all complex workings, that apply a Spline generation curve, can be managed. If not enabled, both tool and all complex workings, that apply a Generation of Spline curves, are disabled.
- **Path:** if enabled, the element Path (L24) is managed. The activation is real only if the option *Curve generation*  is enabled. In Generation of spline curves the selection of Curve of the Path is available.
- **Tool codes:** if enabled, it enables the complex working codes programming tools (Stool). We suggest that also the management of the "N" property should be enabled with this option.

All these releases are applied in Professional mode only.

## Piece geometry



Piece geometry: Faces in transparency

Assign for real faces

□ Customize graphics representation

- Face 1
- Face 2
- Face 3
- Face 4
- Face 5
- Face 6

| | |
|---|---|
| Enable | ☑ |
| Face number | 1 |
| P0-P1 | 0;0 |
| Z+/- | □ |
| XY | H;0 |

Assign for variable Geometries

□ Customize graphics representation

☑ Assign Z axis direction

☑ Assign face thickness

☑ Assign reference face

Number of Additional parameter for each face: 3

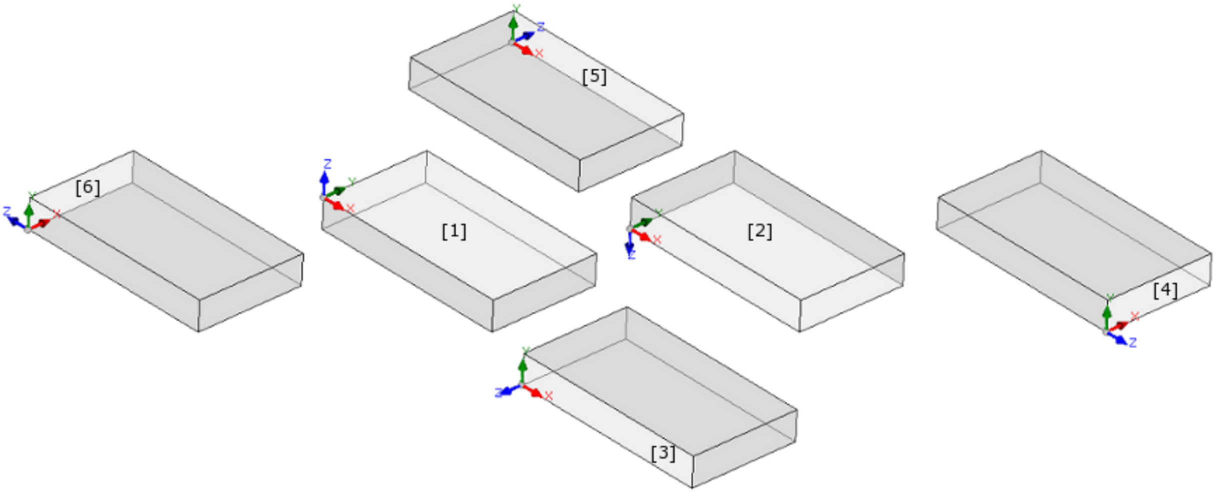
☑ Find face similarity

☑ Assign the curvature

☑ Surface

- **Piece geometry:** it sets the geometry applied to the real faces of the piece.
  - **Faces in transparency** {by default}: the coordinate system of the points which defines the plane of a face is assigned automatically and cannot be modified;
  - **Custom Systems**: the coordinate system of the points which defines the plane of a face is assigned directly in the window (see following area);
  - **Absolute system**: it forces the programming of the only face 1 in the transparency system and with the assignments of the only dimension of workpiece length and height. This selection is not available in case of *Essential* functionality. Being this selection active: the area **Assign for real Faces** is totally disabled. The programming is not longer meant for an application to a three-dimensional workpiece, but to the absolute Cartesian system, which is represented by the reference XY-plan (and bounded by the dimensions of length and height) and by the depth Z - axis. The notations of feed-through or programming above the workpiece or programming within the piece have no longer a specific meaning and in this mode, all the technological processes, both the punctual or the setup ones, are always represented as oriented technologies. Selecting **Absolute system:**
    - anyway, the management of the Fictive faces and the Automatic faces section (in face-piece) remains possible;
    - The programmed workings are displayed by deleting each particular control not bound to the recognition of the feed-through or depth programming above the workpiece (dotted lines or particular colours).

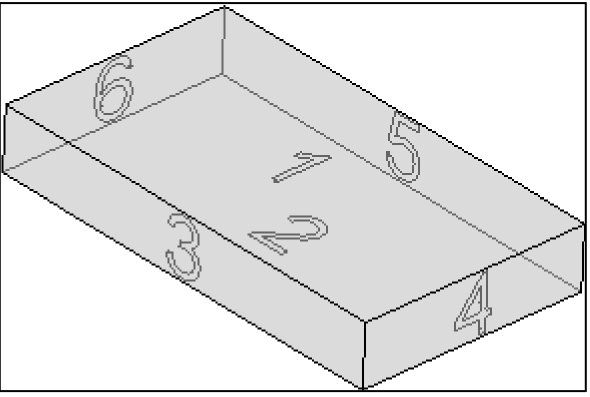
In the figure the coordinate axes for the faces of the workpiece are represented with the optionFace in transparency enabled.



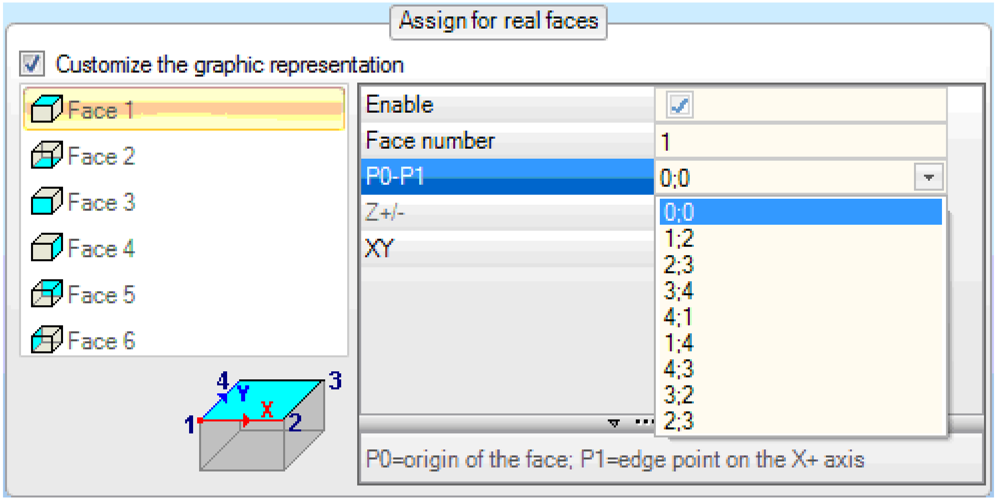
**Assign for real faces**

- **Customizes the graphic representation:** if selected, this option allowsthe representation of the real faces to be set. This is is different from the standard representation on the Cartesian plane (where the origin is on the bottom left, the horizontal and positive x-axis is rightwards, the vertical and positive y-axis is upward). (see later: option XY in the table of the faces)
- **List of faces:** list of the real faces of the parallepiped:
  - the writing shows the automatic face numbering (e.g.: "Face 1")
  - the icon marks the position of the face on the workpiece and shows if the face is enabled (example for the face 1:□) or disabled (example for the face 1:⊟
  - the order is progressive in the automatic numbering of the faces.

  To customise the individual face, move the selection to the list on the right and change the fields:
  - **Enable**: select to enable the face
  - **Face number** : sets the number to use in TpaCAD for the face

| | |
|---|---|
| In the picture the automatic numbering of the real faces of the piece is displayed. If you want to assign a custom numbering of the faces, each number must be different (embracing also the non-enabled faces) and must be included between 1 and 6. Generally, a custom numbering of the piece faces should match in the technological assignments in the same way. It must be arranged during the plant design. |  |

- **P0-P1:** it customises the XY-plane of the face (the selection is enabled only if Geometry set for Custom systems). If enabled, a list of possible selections is assigned to choose the points that define the origin of the XY-plane (point: P0) an the edge point of the face along the X-face+ respectively (point: P1). The list assigns 9 items of which the first one ("0;0" represents the selection by default (corresponding to Geometry set for Faces in transparency). A picture helps to place the edges of the face on the piece. The pictures shows the case of a setting for the face 1. The 4 edges of the face are numbered from 1 to 4.



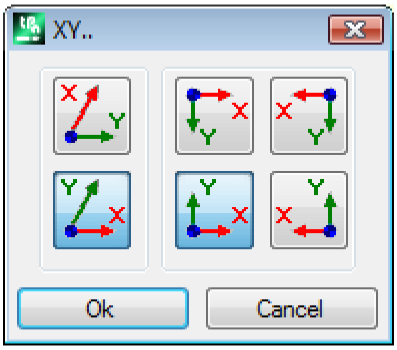
Assign for real faces

☑ Customize the graphic representation

| | |
|---|---|
| Enable | ☑ |
| Face number | 1 |
| P0-P1 | 0;0 |
| Z+/- | |
| XY | |

P0=origin of the face; P1=edge point on the X+ axis

- **Z+/-:** assigns the direction of the Z-axis over the piece with respect to the piece (view only). If the box is not ticked, that means that the XYZ- coordinate system is right-handed; a ticked box means that the XYZ-coordinate system is left-handed
- **XY::** selects la graphic representation for the face. The selection is enabled if the optionCustomize the graphic representationis selected. Select the icon ... to o pen the window:



- To set the direction of the x-axis representation, select one of the bitmaps of the group on the left:
  - horizontal (bitmap below)
  - vertical (bitmap above)
- to set how the YX-plane must be represented, select one of the 4 bitmaps of the group on the right.

The graphic display relates to how the face is displayed in plane view (2d). For each face the display by default occurs with horizontal X-axis and XY-origin of the plane on the bottom left.
***WARNING:*** the settings of the column*P0;P1* modify the interpretation and the execution of a program. The positioning of the workings in the piece depend on the reference geometry. Nothing changes instead at the level of program record.

***WARNING:*** If all the six faces are disabled, you need to enable the management of the Fictive faces and/or of the Automatic faces in piece-face. Otherwise, the activation of the face 1 is automatically forced as active.
***WARNING:*** if all the 6 real faces are disabled, in the subroutine the face 1 is always enabled.
***WARNING:*** if all the six real faces are disabled, a format import assigns the subroutine typology.

## Assign variable geometries

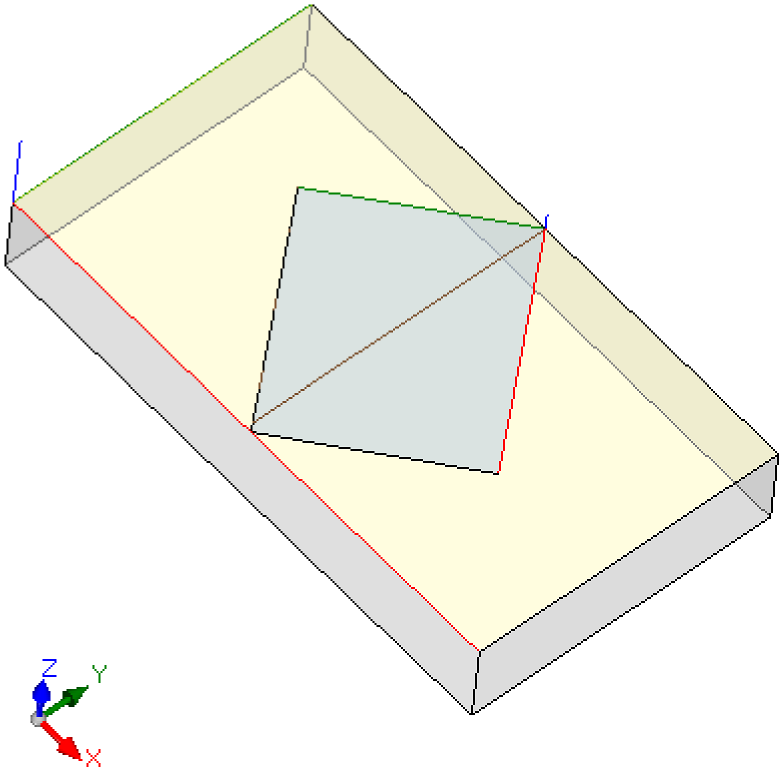
They are a group of selections defining particular aspects of the assignment of the fictive faces and automatic faces in face-piece.
***WARNING:*** all the settings modify the interpretation of a program.

- **Customize the graphic representation**: if selected, this option allows the representation of the fictive faces to be set in a different way from the standard representation on the Cartesian plane
- **Assign Z axis direction:** if selected, it allows the direction of the Z-axis in air of the fictive faces to be assigned. If this option is not selected, the Z-axis of a fictive face is so assigned as to have always a right-handed coordinate system
- **Assign face thickness:** if selected, it allows the thickness of the fictive faces to be assigned. If this option is not assigned, the thickness of a fictive face is always equal to the thickness of the piece. This selection is deleted anyway with geometry on the workpiece **Absolute System**.

- **Assign reference face:** if selected, it allows a reference face of the fictive faces to be assigned besides the flag of Construction face. This selection enables also the tool creating fictive faces from a face program. This all can only be enabled in <Professional> mode.
- **Number of additional parameter for each face:** the field sets up to 5 parameters added to each fictive face. **Note**: the additional parameters can be set also for the automatic faces (if correctly configured in the database of the workings). the value 0 deactivates the control of the parameters. The additional parameters in the faces have a numerical type (double), the programming is available in parametric form (maximum length = 100 characters) and they have a generic meaning of linear dimension (they have a linear dimension of [mm] o [inch]).
  ***WARNING:*** This setting is not available in case of *Essential functionality.*

- **Find face similarity:** if selected, it enables the applications of similarity criteria of the fictive and automatic faces, If the option is selected, a fictive (or automatic) face similar to one of the six real faces of the workpiece, can apply the sameworkings and the sametechnology of the real face. The similarity between faces is evaluated only between a fictive (or automatic) face and a real one.
  Two faces are similar, if the triad of the faces can be overlapped only by:
  - translating (in every direction) and/or
  - rotating, only on the plane of a face.

The planes of the two faces must be parallel and the Z semi-axis must overlap.



The figure shows an example of fictive face (e.g. face 7) similar to the face 1. More specifically:

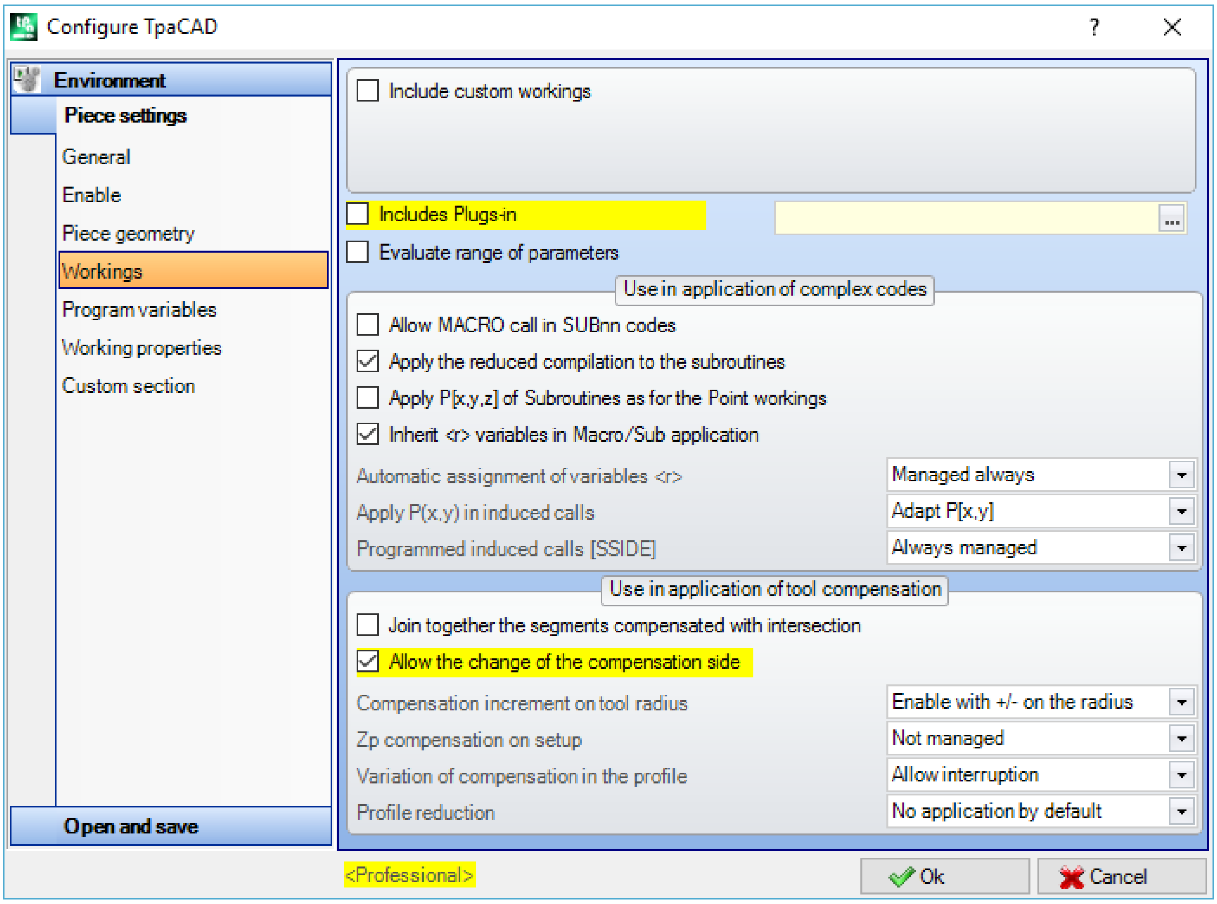
- the plan of the face coincides with that of the face 1.
- The two z - semi-axis are both oriented upward.

In the example of the face 7:

- a working, that can be applied only in face 1, can be applied also in face 7;
- the technology if the face 1 will be used also for the face 7.

- **Assign the curvature:** if selected, this option enables the assignment of fictive faces and automatic curves (please, read the specific documentation in the manual of TpaCADMD). The use of the functionalities requires a specific activation from HW key besides the *Professional* functionality.
- **Surface:** if selected, this option enables the assignment of fictive faces of Surface typology (please, read the specific documentation in the manual of TpaCadMD). The use of the this functionality needs a specific authorization from HW key, in addition to the *Professional* functionality*.*

## Workings



☐ Include custom workings

☐ Includes Plugs-in

☐ Evaluate range of parameters

Use in application of complex codes

☐ Allow MACRO call in SUBnn codes

☑ Apply the reduced compilation to the subroutines

☐ Apply P[x,y,z] of Subroutines as for the Point workings

☑ Inherit <r> variables in Macro/Sub application

| | |
|---|---|
| Automatic assignment of variables <r> | Managed always |
| Apply P(x,y) in induced calls | Adapt P[x,y] |
| Programmed induced calls [SSIDE] | Always managed |

Use in application of tool compensation

☐ Join together the segments compensated with intersection

☑ Allow the change of the compensation side

| | |
|---|---|
| Compensation increment on tool radius | Enable with +/- on the radius |
| Zp compensation on setup | Not managed |
| Variation of compensation in the profile | Allow interruption |
| Profile reduction | No application by default |

- **Include custom workings:** if selected, the database reading the custom workings is enabled. TpaCAD reads by default the database of the base database (from the file tpacadcfg\ DBWORKS.WCAD). If the selection is active, TpaCAD reads also the database of the custom workings (from the file: tpacadcfg\custom\ DBWCUST.WCAD).
  **WARNING**: This selection is not available in case of *Essential* functionality.
- **Evaluate range of parameters:** if selected, the control of parameters of the workings is enabled. The value set is evaluated with respect to the range (if not null), as assigned in the database configuration of the workings. If the value given to the parameter exceeds the range, an error is reported. Non-editable parameters are excluded from the evaluation.

**Use in application of complex codes**

- **Allow MACRO call in SUBnn codes:** if selected, a macro can be called from a generic call to a subprogram (SUB0, SUB2).
  - **WARNING:** The option can be enabled while testing the macros written for a customer, but **it must remain disabled**, when the machine is installed (the option must be disabled by the developer). In normal conditions:
    - a macro can be called only by means of a complex code to configure in the database of the workings;
    - the programs, which apply macros directly with a working of generic subprogram call, must be only used for a first test, then deleted.
    - **WARNING**: This selection is not available in case of *Essential* functionality.
- **Apply the reduced compilation to the subroutines:** if selected, it allows the application of the same valid criteria for a macro to the compilation of a subprogram. **WARNING:** this is a setting which modifies the interpretation and the execution of a program.
  A macro is applied after checking row by row, evaluating and directly applying the following:
  - the logical conditions (IF...ELSE..ELSE; FOR.. ENDFOR; BREAK; CONTINUE; EXIT);
  - programmed error conditions (BREAK; ERROR);
  - the assignation of variables (<j>, <$>).

  The working list corresponding to the macro development is built step-by-step on these considerations. In case of application of a macro, a Reduced Compilation is always meant, because only what is verified as TRUE is evaluated and applied.
  If a complex code (both generic call and generic SUB, SMAT) applies a subprogram, a Non-reduced Compilation can be applied, if the field we are here examining is not selected. In this case:

- the subprogram is processed fully, without taking into account the programmed logical conditions;
- only after the process has been completed, the logical conditions are evaluated and applied and therefore the lines of program, which correspond to the status of FALSE, are deleted.

All the transforms of assigned developments (emptying closed areas, positioning to coordinates, geometric transforms, repetitions) in the application of the subprogram are in any case applied after the compilation, after reducing the workings and after checking the logical conditions.
A complex code, which applies a subprogram, can directly select which kind of compilation must be applied. The option is assigned with the parameter PRSUBCOMPILE (ID=8102, integer):

- non-configured parameter for the working: it applies the compilation as set in the configuration;
- 1: applies Non-reduced compilation;
- 2: applies Reduced compilation.

Some examples may clarify what differences can lead to a different choice of the criterion of compilation.

**Example 1: conditioned execution of drilling in relative programming**

| **Text examined** | **Non-reduced compilation** | **Reduced compilation,** |
|---|---|---|
| 1. IF (r1<1) | (TRUE) | |
| 2. HOLE (X=r0 Y=100) | ***HOLE (X=200 Y=100)*** | ***HOLE (X=200 Y=100)*** |
| 3. ENDIF | (TRUE) | |
| 4. IF (r1>=1) | (FALSE) | |
| 5. HOLE (X=100 y=r0) | HOLE (X=100 Y=200) (FALSE) | == |
| 6. ENDIF | (FALSE) | |
| 7. IF (r5>1) | (FALSE) | |
| 8. HOLE (X=32) incremental | HOLE (X=132 Y=200) - (FALSE) | == |
| 9. HOLE (X=32) incremental | HOLE (X=164 Y=200) - (FALSE) | == |
| 10. ENDIF | (FALSE) | |
| 11. HOLE (X=32) incremental | **HOLE (X=196 Y=200)** | **HOLE (X=232 Y=100)** |
| 12. HOLE (X=32) incremental | **HOLE (X=228 Y=200)** | **HOLE (X=264 Y=100)** |

The subprogram is applied with r0=200, r1=0, r5=0.

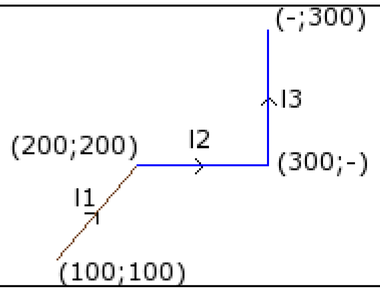
Drilling workings, which the complex code actually develops in both cases, are marked in bold and italic. The calculated coordinates highlight that their development can also drastically change. The situation, represented in the Non-reduced column, follows the drilling scheme as it is actually represented while writing a subprogram.

**Example 2: Profile construction with logical conditions**

Let us see now how the Reduced Compilation allow profiles to be constructed, also in the application of a subprogram.
The subprogram of the example sets three linear segments, with exclusive condition of execution in the second and third segment:

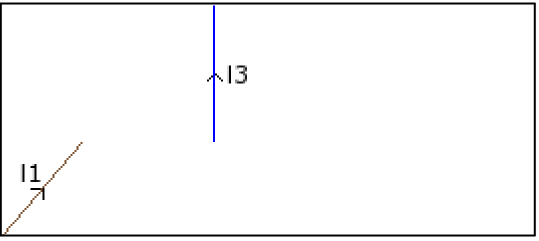
- if r0=0: it executes the second segment (it moves the X to 300 position);
- if r0 is not equal to 0: it executes the third segment (it moves the Y to 300 position).

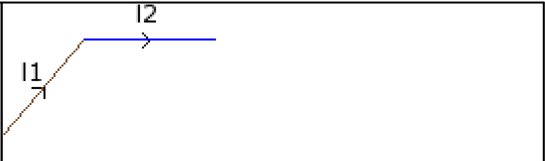
The first segment opens the profile. The starting point is programmed in (100;100), the final point in (200;200). The two next segments program the final point only by moving the X and the Y-axis.
The outlines of the subprogram text are as follows:

| | |
|---|---|
| 1. LINE (100;100) -> (200;200)<br>2. IF (r0=0)<br>3. LINE final X=300<br>4. ELSE<br>5. LINE final Y=300<br>6. ENDIF |  |

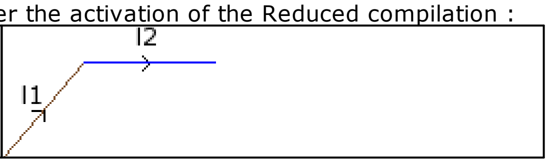
The picture shows the profile as it appears during the programming of the subprogram.

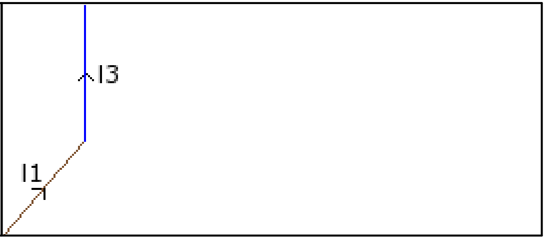
Let us call the subprogram with the SUB code, after the activation of the Non-reduced compilation:

| | |
|---|---|
| r0=1<br>two distinct profiles appear, geometrically non consecutive and made as follows:<br>• the 1°: by the original segment l1<br>• the 2°: by the original segment l3 |  |

| | |
|---|---|
| r0=0<br>two distinct profiles appear, geometrically consecutive and made as follows:<br>• the 1°: by the original segment l1<br>• the 2°: by the original segment l2 |  |

Let us call now the subprogram with the SUB code, after the activation of the Reduced compilation :

| | |
|---|---|
| r0=0<br>a single profile, made of the original segments l1 and l2, appears |  |

| | |
|---|---|
| r0=1<br>a single profile, made of the original segments l1 and l2, appears |  |

The option of Reduced compilation has performed the construction of a profile in a very similar way to what happens, when a macro is applied.
In the same way, the three original profiles could be programmed with a hooking setup. In this case, the only difference occurred for the case of Non-reduced compilation. With r0=1 the two profiles l1 and l3 could be geometrically consecutive, but still distinct.

- **Apply P[x,y,z] of Subroutines as for the point workings:** if selected, it allows the application point of a subprogram and of a macro to be interpreted with the same valid criteria for a point working. More specifically, a non-assigned coordinate is propagated from the previous working. However, if the selection is not active, a non-active coordinate is interpreted as calculated in the subprogram and macro text. In any case, a complex code can directly select which criterion should be applied by setting the field in the database of the workings.
  **WARNING:** this is a setting which modifies the interpretation and the execution of a program.
  **WARNING:** this is a selection which was not available in Edicad. The recovery of programs written in Edicad can determine a different interpretation of the program. The operation of Edicad is always related to the case of a non-active selection.
- **Inherit the <r> variables by applying Macro/Sub:** if selected, it enables the public r-variables (re-assignable) to be inherited by applying a subprogram and a macro. The selection covers the case of a non-assigned public r-variable (the data-entry field is empty). In this case:
  - if the option is selected, the variable is calculated with the value set in the macro and subprogram text;
  - if the option is not selected, the value of the variable is 0.

  The non-active selection involves the adoption of some devices while writing subprograms and macros. Let us take, for example, the case of a macro which develops a cycle of drilling at a constant step. The public r3-variable for the cycle step is assigned to the text of the macro, with value equal to 32.0.
  - In case of non-inherited r-variables (non-active selection), if the setting field of the step is left empty, for r3 the value 0.0 is used and a possible different assignment by default must be managed with programming tools (logical condition, functions: rempty, ifcase, ..).
  - In case of inherited r-variables (non-active selection), if the setting field of the step is left empty, for r3 the value 32.0 is used.

  In any case a complex code can select directly which criterion must be applied by setting the field in the database of the workings.
  **WARNING:** this is a setting which modifies the interpretation and the execution of a program.

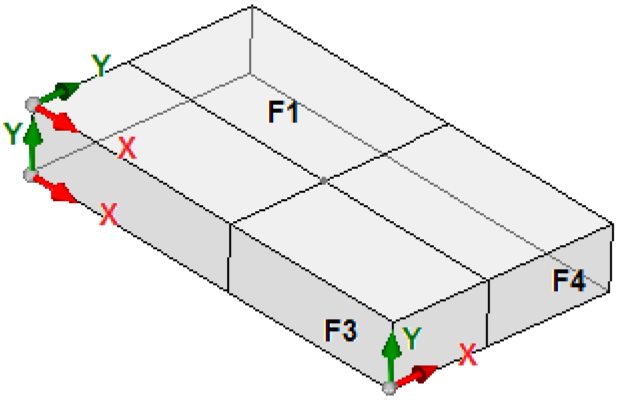
**WARNING:** this is a non available selection in Edicad. The recovery of programs written in Edicad can determine a different interpretation of the program. The operation of Edicad is always related to the case of active selection.

- **Automatic assignment of variables <r>:** this option sets the assignment modes of the r variables used in a subprogram or in a macroprogram, without being defined in the, when the subprogram is called by a program. The available typologies are three:
  - **Not managed:** the value assigned to the variables is always 0
  - **Managed always:** {by default} the variables searched in the caller program and, in case of more cascading calls, the backward search of the value to be assigned can continue until the main program.
  - **Managed by "r\name" only:** this selection makes the functioning differ from the called variable and automatic name (example : "r0") or specific name (example: "r\fitool"): in the first case the value assigned to the variables is always 0, in the second case a previous search is applied.

***WARNING:*** this is a setting that modifies the interpretation and the execution of a program.

- **Apply P(x,y) in induced calls:** it sets the default mode of adaptation for the application point in the induced calls, limited to the X and Y coordinates. The available typologies are three:
  - **Adapt (x,y):** the assignments of X and Y comply with a geometric correspondence of the X and Y axes on the different faces. The following table defines the cases:

| Face of Master call | Face of Induced call | Coordinate of Induced call |
|---|---|---|
| (1, 2) | (4,6) | X = Y coordinate from master face (if not set ="")<br>Y = "" |
| (1, 2) | (3,5) | X= X coordinate from master face<br>Y = "" |
| (3,5) | (1,2) | X= C coordinate from master face<br>Y = "" |
| (4,6) | (1,2) | X = Y coordinate from master face (if not set ="")<br>Y = "" |
| any other case | any other case | X = X coordinate from master face<br>Y = Y coordinate from master face |



The table above takes reference to the case of geometry of the Piece in transparency. In case of geometry of the piece assigned with Custom systems, if the assignments of the table do not meet the requirements, they must be excluded by the assignment of a different setting in the Placing mode of the induced calls.
- **Do not adapt (x,y):** the assignments of X and Y are given as programmed for each induced call.
- **Do not pass (x,y):** the assignments of X and Y are reset for each induced call.

For each application of subprogram it is anyway possible to modify the kind of selection by setting the parameter [8110] (mentioned in the base database under the item: induced XY).
**WARNING:** this is a setting which modifies the interpretation and the execution of a program.
**WARNING:** this is a non available selection in Edicad. The recovery of programs written in Edicad can determine a different interpretation of the program. The operation of Edicad always meets the case of Adapt (x,y).

- **Programmed induced calls [SSIDE]:** it sets the activation required to manage the programmed induced calls. This option concerns the programming of the SSIDE working (operating code: 2012. Three options are listed, as follows:
  - **Not managed:** SSIDE working is never interpreted

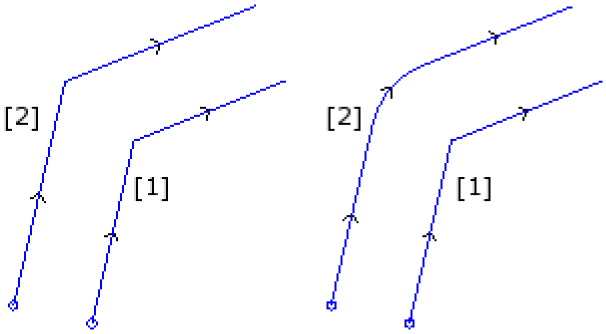
- **Managed in piece-face:** {default} SSIDE working is used in subroutines or macro-program writing and it must be used in piece-face programming.
- **Always managed:** {default} SSIDE working is used in subroutines or macro-program writing and it must be used in piece-face programming. This selection is available and active in <Professional> mode only.

***WARNING:*** This selection is not available in case of ***Essential*** functionality***.***

**Use in application of tool compensation**

This is a group of selections, which define particular aspects of application of the compensation tool.
**WARNING:** all settings modify the interpretation of a program.

- **Join together the segments compensated with intersection:** if selected, as a default condition in the application of the tool compensation, the research of the intersection points outside of compensated segments which do not intersect (left picture), is enabled. If the option is not selected, as a default condition the application of the tool compensation with insertion of connection fillets (right picture) is enabled.



For each single profile the selection can be locally customized by setting the parameter [39] (mentioned in the base database at the option Contouring), assigned for the Setup workings.

- **Allow the change of the compensation side:** if selected, it enables the management of the change parameter of the compensation sides in the profile segments (from the left to the right one or viceversa). If the option is not selected, side change cannot be programmed.
  **Note:** the control of the side change is available in <Professional> mode only.
- **Compensation increment on tool radius:** it sets the way in which the variation to apply to the value of the tool radius must be set. This selection influences the way to interpret the coding of the Setup working parameter, assigning the value of the compensation radius. Three options are listed:
  - **Not managed** {by default}**:** it is not possible to interpret a change to be applied to the tool radius.
  - **Enable with +/- on the radius:** it interprets the change to be applied to the radius of the tool with programming in the form "+.." o "-.."

    Examples of programming of the Compensation radius:

| | |
|---|---|
| "" (empty field) | applies the tool radius (as read from the plant technology) |
| "12.5" | applies compensation radius=12.5 |
| "+3" | applies compensation radius = (tool radius) + 3 [mm] |
| "-0.5" | applies compensation radius = (tool radius) - 0.5 [mm] |
| "+(r5/r2)" | applies compensation radius = (tool radius) + absolute value of (r5/r2) |

  - **Enable always:** it directly applies the change to the tool radius with programming set.

| | |
|---|---|
| "" (empty field) | applies the tool radius (as read from the plant technology) |
| "12.5" | applies compensation radius = (tool radius) + =12.5 [mm] |
| "+3" | applies compensation radius = (tool radius) + 3 [mm] |
| "-0.5" | applies compensation radius = (tool radius) - 0.5 [mm] |
| "+(r5/r2)" | applies compensation radius = (tool radius) + absolute value of (r5/r2). |

- **Zp compensation on setup:** it sets the way in which the compensation tool is managed at startup, adding a linear movement from the programmed position of the Setup to the start point of the compensated profile. Three options are listed as follows:
  - **Not managed** {by default}: it is not possible to interpret a compensation adding a linear movement at the startup
  - **Do not apply by default:** it manages the application, but only on request in the single profile
  - **Apply by default:** it manages the compensation and applies it in each profile, unless it is not explicitly excluded in the single profile.

If the selection is different from *Not managed* , for each single profile it is possible to customize the selection by setting the parameter [38] (mentioned in the base database di base under the option: Start compensation from the setup), assigned for the Setup working.

- **Variation of compensation in the profile:** it sets the way in which the change of the tool compensation on the profile is applied. Three options are listed:
  - **Not managed** {by default}: it is not possible to interpret a change to be applied during the tool compensation;
  - **Allow interruption**: it is possible to interpret an interruption in the profile compensation and a possible following resumption
  - **Allow suspension** (include also the interruption): it is possible also to interpret a suspension in the profile compensation and a possible following resumption.

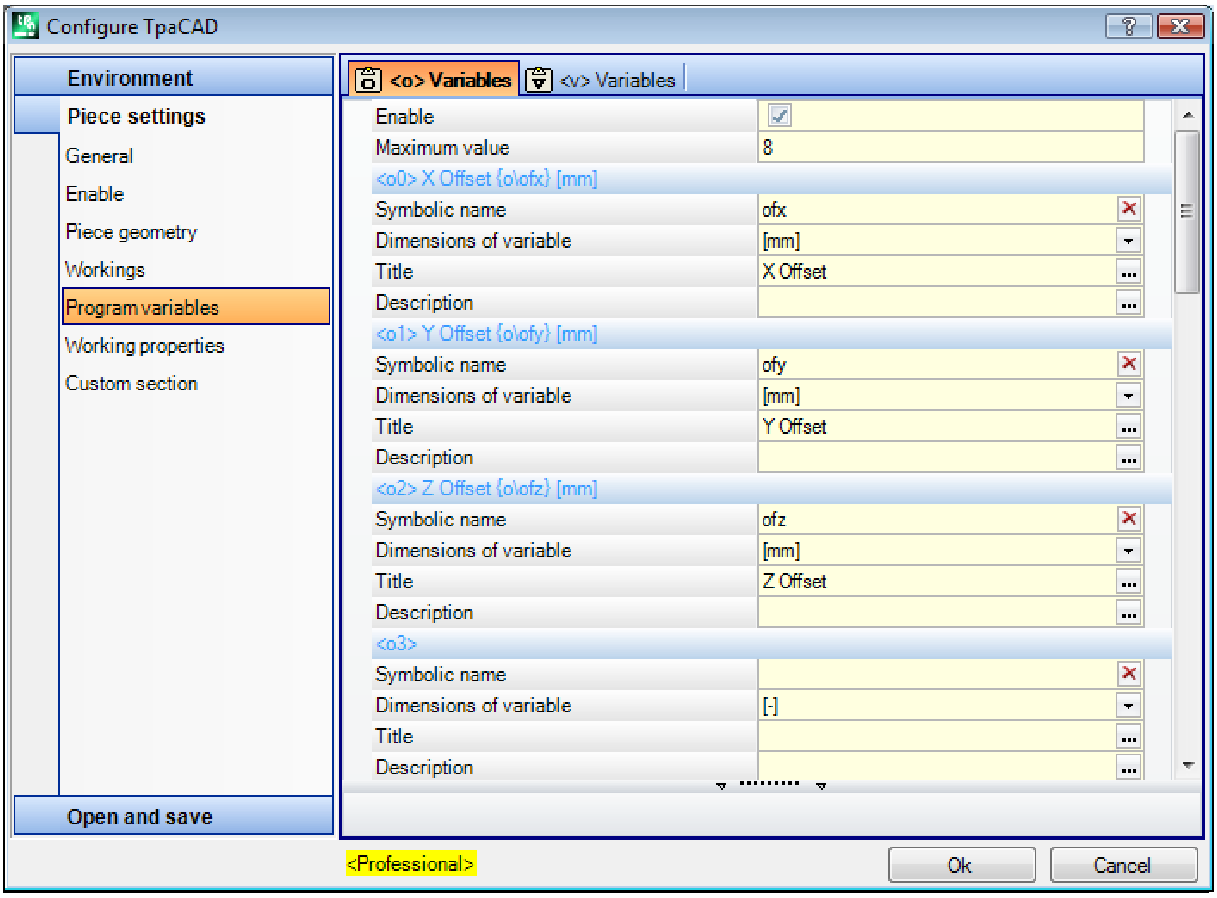
  If Not managed is not selected, for each single profile the selection can be locally customized by setting the parameter [42] (mentioned in the database under the option: Compensation).

  **WARNING**: if the selection is active, the relative management shall be available in <Professional> Mode only.
- **Profile reduction** it sets the authorization to remove the segments in the correct profile, with respect to the original one, on the basis of geometric clearance restrictions exceeding compensation. Three options are listed:
  - **Not managed** {by default}: it is not possible to manage the reduction of the profiles
  - **No application by default**: it manages the reduction but only in the single profile on request (recommended)
  - **Apply by default:** it manages the reduction and applies it in each profile, unless it is not explicitly excluded in the single profile.

  If *Not managed*, is not selected, for each single profile le selection can be locally modified by setting the parameter [46] (mentioned in the base database under the option: Reduce profile), assigned for the workings of Setup.

  **WARNING:** the reduction of a profile can only be applied where needed and can also delete more consecutive segments. We need to point out that, anyway, the reduction does not take into account the profile as a whole. When a segment must be deleted, a solution of intersection between segments respectively before and after the deleted segment is searched, without examining whether the intersection interferes with other parts of the profile. For very complex profiles, we therefore suggest that the reduction is enabled, if required, and its result is evaluated, especially with values of compensation widely exceeding the overall dimensions of a profile.

## Program variables



| | |
|---|---|
| Enable | ☑ |
| Maximum value | 8 |
| <o0> X Offset {o\ofx} [mm] | |
| Symbolic name | ofx |
| Dimensions of variable | [mm] |
| Title | X Offset |
| Description | |
| <o1> Y Offset {o\ofy} [mm] | |
| Symbolic name | ofy |
| Dimensions of variable | [mm] |
| Title | Y Offset |
| Description | |
| <o2> Z Offset {o\ofz} [mm] | |
| Symbolic name | ofz |
| Dimensions of variable | [mm] |
| Title | Z Offset |
| Description | |
| <o3> | |
| Symbolic name | |
| Dimensions of variable | [-] |
| Title | |
| Description | |

Both sections allow the assignment of the operation of the <o> and >v> Variables:

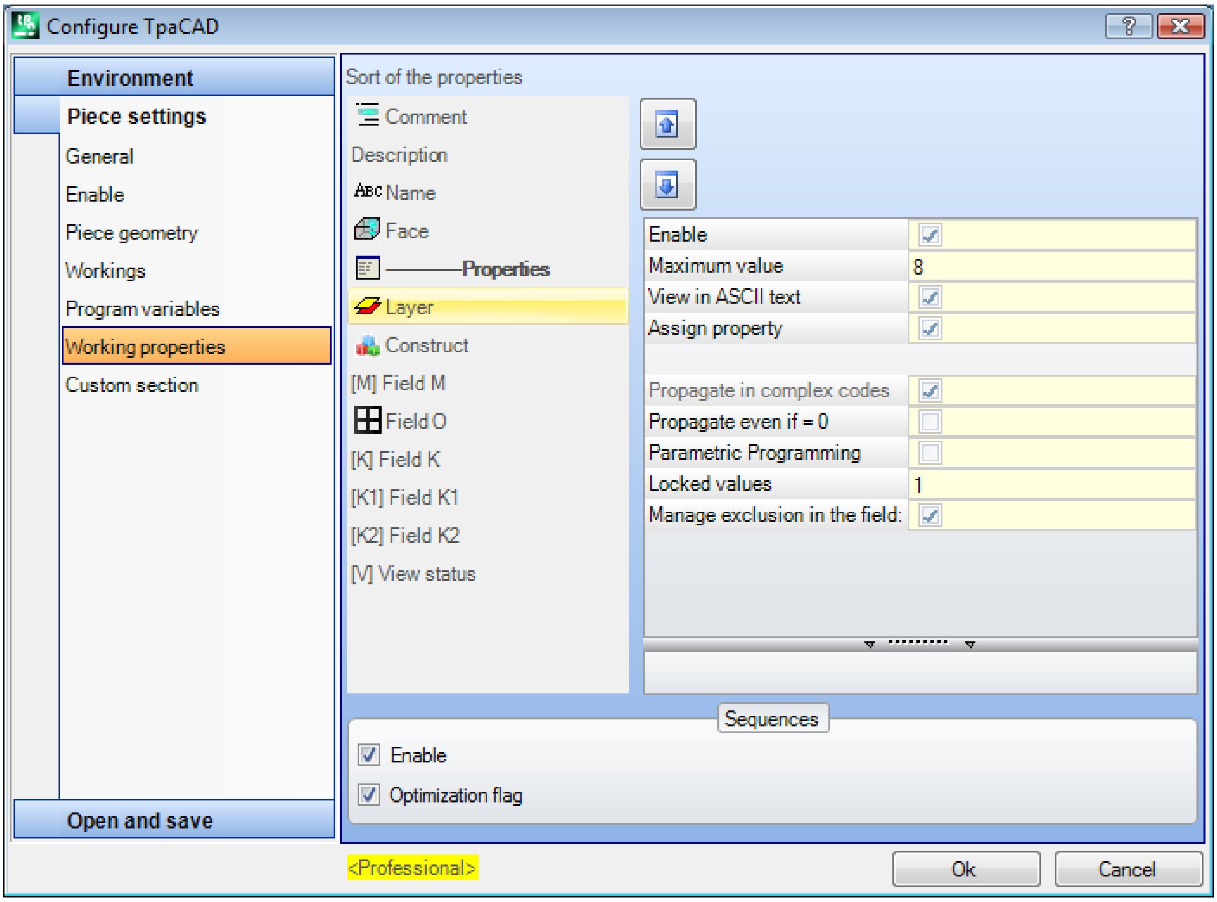
- **Enable:** if selected, it enables the management of the <v> variables or of the <o> variables.
- **Maximum value:** it sets the number of the managed variables (from 1 to 16). Non-managed variables are not read or written in the programs.

For each configurable variable a node is assigned with the following fields:

- **Symbolic name:** it sets the symbolic name in letters for the variable (max. length 16 alphanumeric characters, first alphabetic character). Repeated names are deleted.
- **Dimension of the variable**: it sets the dimension assigned to the variable.
  - [-]  the variable is non-dimensional {by default}
  - [mm] the variable is a coordinate ([mm] or [inch])
  - [m/’] the variable is a moving speed rate (Linear speed rates **from the** page ***Piece settings ->General*** is set)
- **Title:** it sets a message to be used as a title of the property (shown in the property table in TpaCAD). If you select the button ..., the window of the list of messages defined in the custom file messages CADAUX (in the folder tpacadcfg\custom) is opened.
- **Description:** it sets a message to be used as a description of the property (shown in the corresponding table). If you select the button ..., the window of the list of messages defined in the custom file messages CADAUX (in the folder tpacadcfg\custom) is opened.

The dimensions of the "o" and "v" variables provide the application of the conversions during the creation of a piece matrix.

## Working properties



Configure TpaCAD

Environment

Piece settings

- General
- Enable
- Piece geometry
- Workings
- Program variables
- Working properties
- Custom section

Open and save

Sort of the properties

- Comment
- Description
- Name
- Face
- ———Properties
- Layer
- Construct
- [M] Field M
- Field O
- [K] Field K
- [K1] Field K1
- [K2] Field K2
- [V] View status

| | |
|---|---|
| Enable | ☑ |
| Maximum value | 8 |
| View in ASCII text | ☑ |
| Assign property | ☑ |
| Propagate in complex codes | ☑ |
| Propagate even if = 0 | □ |
| Parametric Programming | □ |
| Locked values | 1 |
| Manage exclusion in the field: | ☑ |

Sequences

☑ Enable

☑ Optimization flag

<Professional>

**Sort of the properties**

The list on the left assigns the order by which the properties are given in the window of working assignments. To change the order it is enough to select the corresponding option, then select the buttons ⇩ or ⇧, until the required order is reached.

▤ ———Properties represents the group of properties, normally carried an the end of the working window:

- the items assigned before are listed before all the groups of parameters
- the items assigned aftger are listed in the group of properties (if correctly configured, otherwise at the end of all).

For each property different options can be set:

- **Enable:** it enables the management of the property (this selection is available for all properties)
- **Max. value:** it sets the maximum value that can be assigned to the property. (this selection is not available for the properties, which can enable on/off only)
- **View in ASCII text:** this option enables the view of the column of the values in ASCII text
- **Assign property:** this option enables the management of the corresponding command for the overall assignment in the *Assign property* group of the *Edit menu.*

- **Description:**it sets the message to be used as a name of the property (e.g. in the data-entry control of a working). If you select button … the window containing the list of the messages defined in the file of custom messages CADAUX (in the folder:cadcfg\custom) is opened. It is possible to modify or to enter the message. This option is available for the following properties: O, K, K1,K2.
- **Listed items:** select the button to … to open the window containing the list of the messages that mus be associated to the property value if *Selection from the list* is enabled (during the customisation). As usual, the messages are defined in the CADAUX file (in the folder: pacadcfg\custom) and the list is

restricted to 17 items, according to the values from 0 to 16. This option is available for the following properties: O, K, K1,K2

- **Set in profile segments:** select to enable the possibility to assign the property also in the single segments of a profile (lines, arcs). This option is available for the O and M properties.
- **Propagate in complex codes:** this options enables the propagation of the property in the development of a subroutine or of a macro (complex code). If this option is not selected, the subroutine working or the complex code do not propagate their property to the expansions.
- **Propagate also if = 0:** ifPropagate in complex codes is selected, it enables the propagation of the property 0 value, as well.
- **Parametric programming:** if selected, it allows the parametric programming of the property. ***WARNING:***This selection is not available in case of *Essential functionality.*
- **Locked values:** it sets the number of the values which are locked in the configuration. A value, which cannot exceed the value set in the field **Max. value** can be assigned. This setting is available for the following properties: L, B, O (if the fieldSet the profile segments is not selected), K, K1, K2 and is is reset, if the Max. value higher assigned is greater than 16 or if the field Parametric Programmingis selected. Let us consider the case of the L-field, with maximum value = 8 and locked values = : the (numerical) values of the L-field between 0 and 6 are of free programming, while the last 2 values (L=7, L=8) cannot be accessed by the TpaCAD user. Workings with L=7/8 field can only derive from the reading of a program (typically generated by a format import) and cannot be changed, deleted and also entered Paste function included). During the programming, the maximum value accepted for the L - field is 6. The assignment of locked values is related to the external generation of programs in TpaCAD format. ***WARNING:***The setting of the field is not available in case of *Essential functionality.*
- **Manage the exclusion in the field:** select to enable the assignment of the execution exclusions. This selection is available for the properties: L, K, K1, K2. For each property with active selection, only exclusions until values not exceeding 16 can be assigned. ***WARNING:***This selection is not available in case of *Essential functionality.*

The following table shows the valid selection to set. In case of one only value given, the corresponding cell cannot be modified (it is automatically assigned). Each property is identified by one or two capital letters, shown in brackets in the first column:

| **Field** | Set in profile segments | Max. value | Propagate in complex codes | Propagate if =0 | Parametric Programming |
|---|---|---|---|---|---|
| **Comment (C)** | ON | = | ON | OFF | OFF |
| **Livello (L)** | OFF | 1-255 | ON | OFF/ON | OFF/ON |
| **(B)construct** | OFF | 1-255 | ON | OFF/ON | OFF/ON |
| **O Field (O)** | OFF/ON | 1-255 | ON | OFF/ON | OFF/ON |
| **M field (M)** | OFF/ON | 1-65000 | ON | OFF/ON | OFF/ON |
| **K field(K)** | OFF | 1/-65000 | OFF/ON | OFF/ON | OFF/ON |
| **K1 field(K1)** | OFF | 1-255 | OFF/ON | OFF/ON | OFF/ON |
| **K2 field(K2)** | OFF | 1-255 | OFF/ON | OFF/ON | OFF/ON |
| **(N) Name** | ON | = | OFF | OFF | OFF |
| **V field** | OFF | = | ON | OFF | OFF/ON |
| **Modelling** | OFF | = | OFF | OFF | OFF |

Let us examine the properties specifically:

- **Comment (C)**: the value, which the property can take,is 0 or 1. Workings with C-field=1 do not affect the program (they correspond to "commented" program lines) and its execution. If the property is not managed, workings with active C-field are not read by the program. The branch suggests the option **Change in ASCII text**: select to allow the change of the property directly from the program ASCII table.
- **Layer (L):**the assignment of different levels allows the assignment of different display colours, display filters and/or change of the workings, exclusions in the program execution. The layers allow homogeneous program lines to be grouped in order to simplify the processes during their view rather than "freeze" temporarily particular layers. Uploading a program, if the property is not managed, the field is reset even if a value is set; if the property is managed, the field is limited to the maximum value in any case,
- **Construct (B):** the assignment of different constructs allows the assignment of different display colours, display filters and/or change of the workings. The construct (value>0) is primarily used to define graphical help for workings. For this reason with workings containing an assigned Construct:
  - the graphics exclude the application of arrows, extreme points of the profile and depth extension
  - normally they are not executed.

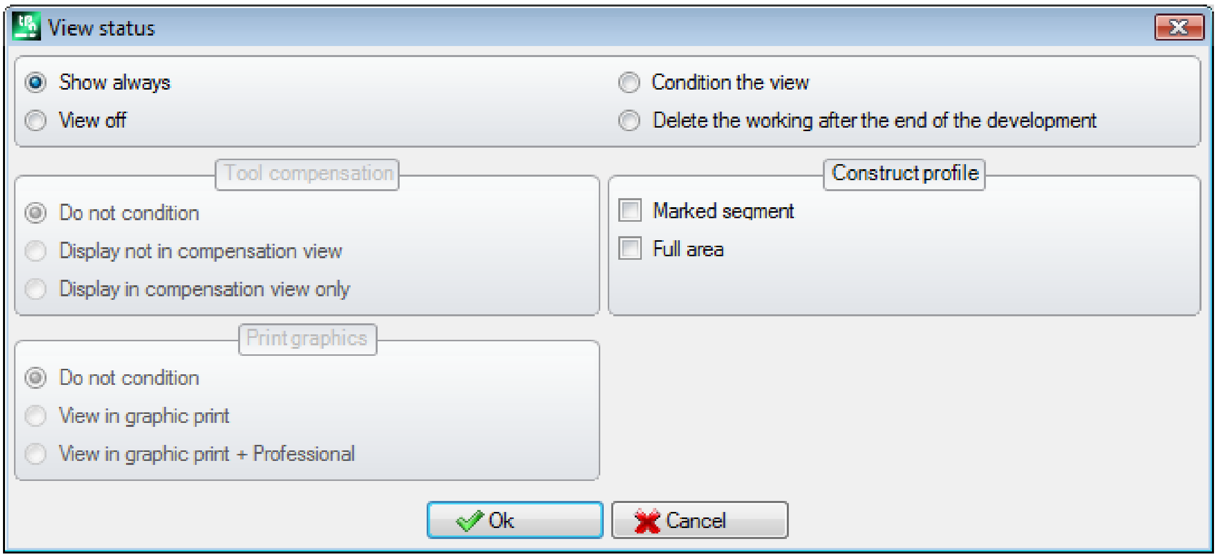
  Uploading a program, if the property is not managed, the field is reset even if a value is set; if the property is managed, the field is limited to the maximum value in any case,
  If the maximum programmable value is set at 1, the branch suggests the option **Change in ASCII text**: select to allow the change of the property directly from the program ASCII table.
- **O field (O):** the assignment of different values in O-field allows the assignment of different display colours, display filters and/or change of the workings. To the field can be assigned a meaning, as follows:

- a specific meaning of reference origin in the face. This case is recognized for Max. value assigned up to 4. The reference is meant as associated to:
  - a side (low/high or left/right) of the face, withMax. value assigned to 1
  - a face edge, with Max. value assigned greater than 1.

  TpaCAD shows this interpretation by means of the view of specific images, but only if none of the settings of the Piece geometry is verified. Selected option: *Custom Systems* or *Customize graphics representation (*of the faces). However, it is not a program according to the origin, but only a generic association, whose interpretation is delegated to the program execution process.
- fully generic: this case is recognized for Max. value assigned up to 4.

The meaning of the values assigned to the field must be given in accordance with the single application. It is possible to enable the option Set in profile segments. In this case the field can be changed within a profile. Uploading a program, if the property is not managed, the field is reset even if a value is set; if the property is managed, the field is limited to the maximum value in any case,

- **M field (M):** The usefulness of the M-field may reflect the general use in M-code in the ISO languages (the meaning of the values assigned to the field must be given according to the individual application). More specifically, it is possible to enable the item Set in profile segments. Inthis case the field can be changed within a profile. Uploading a program, if the property is not managed, the field is reset even if a value is set; if the property is managed, the field is limited to the maximum value in any case,
- **K,K1,K2 field:**the meaning of the three fields is totally generic and must be given according to the single application. Uploading a program, if the property is not managed, the field is reset even if a value is set; if the property is managed, the field is limited to the maximum value in any case, The branches show respectively the **idK** (**idK1**,**idK2**) setting field: set the the identification number of the parameter a working is able to recover in the corresponding property. The value is used to convert fields from programs written using Edicad or TpaEdi32, in cases of custom code or of complex workings assigned as fixed Cycle: the value here set (if positive) normally matches a custom parameter (examples: 9540, 9600). The value is only used if strictly positive and if the corresponding field in the working database is not assigned. ***WARNING:***The properties are not available in case of *Essential functionality.*
- **Name (N)**: it is a string field (alphanumeric, not exceeding 16 characters). The use of the N field concerns the application of Tool codes and to the use of advanced functions of parametric programming. If the property is not managed, the filed is reset during the upload of the program. The branch suggests the option **Change in ASCII text**: select to allow the change of the property directly from the program ASCII table. ***WARNING:***The properties are not available in case of *Essential functionality.*
- **Sequences (S)**: the maximum value the property can take is very large and it is always transparent for the operator, because it is managed automatically. If the property is not managed, the file is reset anyway when the program is read. ***WARNING:***The properties are not available in case of *Essential functionality.*
- **Modelling (E field)**: this option enables the management of the piece modelling resolved by programmed profiles (please, read specific documentation in the manual of TpaCadMD). The use of the functionalities concerning the modelling requires a specific authorisation from HW key and *Professional* functionality.
- **V Field**: the value the property can take on is included between 0 and 255. 255. In this page the use can be enabled or disabled. ***WARNING:*** The property is not available in case of *Essential functionality.* V - field management is only available in edit of macro-program only. If the property is not managed, the filed is reset when the program is read. V-field allows the user to assign particular display status associated with a working, The field is set in a window provided for that purpose and the states managed are displayed in the figure:



View status

● Show always
○ View off
○ Condition the view
○ Delete the working after the end of the development

Tool compensation

● Do not condition
○ Display not in compensation view
○ Display in compensation view only

Construct profile

□ Marked segment
□ Full area

Print graphics

● Do not condition
○ View in graphic print
○ View in graphic print + Professional

- **Show always**: no limits are given in displaying the working
- **View off**: working is never displayed.
- **Condition the display**: the display is affected according to the other settings shown.
- **Delete working** : the selection asks to delete the working after ending the application of the macro-program. The selection presumes that the working has never been displayed. The selection allows you to delete directly while applying the macro-program those workings that correspond to partial developments; thus an improper display, so to speak, is avoided.

### Tool compensation

- **Do not condition**: no conditions are set from the View on toll compensation
- **Display not in compensation view**: working is displayed IF the view in compensation tool is NOT applied.
- **Display in compensation view only**: working is displayed ONLY IF the view in compensation tool is applied.

### Print graphics:

- **Do not condition**: no conditions are set from the command of Pring graphics
- **Display in graphic print**: the working is displayed in print execution ONLY
- **Display in graphic print + Professional**: the working is displayed in print execution ONLY, but in <Professional > Mode only.

### Construct profile

Both the settings of the group are not associated to a particular condition, but they are applied, if the previous conditionings allow the representation of the working. The settings are significant, if they are applied to a construct profile, open from a non-oriented setup:

- · Marked segment: if enabled, it enables the profile elements with marked segments. The color used is the color associated to the construct value (field B) assigned for the setup.
- **Full area**: if enabled, it fills up the internal area limited by the closed profile assigned during the customization. This selection is ignored if the profile is not closed.

**WARNING***:* the **V field** propagation in the development of the complex code is applied on the whole to the value assigned to the property and not individually to the assignable states (see the window above).

**WARNING***:* to stop the propagation of the **V Field** in developments carried out, for example, with subsequent uses of STOOL codes, select the **Condition the View** item, without any additional settings. This defines the assignment to the field of a different value from 0 (zero) and for example it can stop the propagation of the selection **Delete the working after the end of the development,** , active for intermediate developments. As an alternative, it is possible to enable the propagation also of the 0 value; in this case, if necessary, you need to repeat the settings also in the programming of the STOOL code.

If the display of the column of the V field in ASCII text is enabled, the display only occurs in Edit of macro-program. Due to the fact that the possible values associated with the property are complex and not homogeneous, we have chosen to display in the table the only display state and not the corresponding value:

gives a name to the column
corresponds to the selection "Show always"

corresponds to the selection "View off "

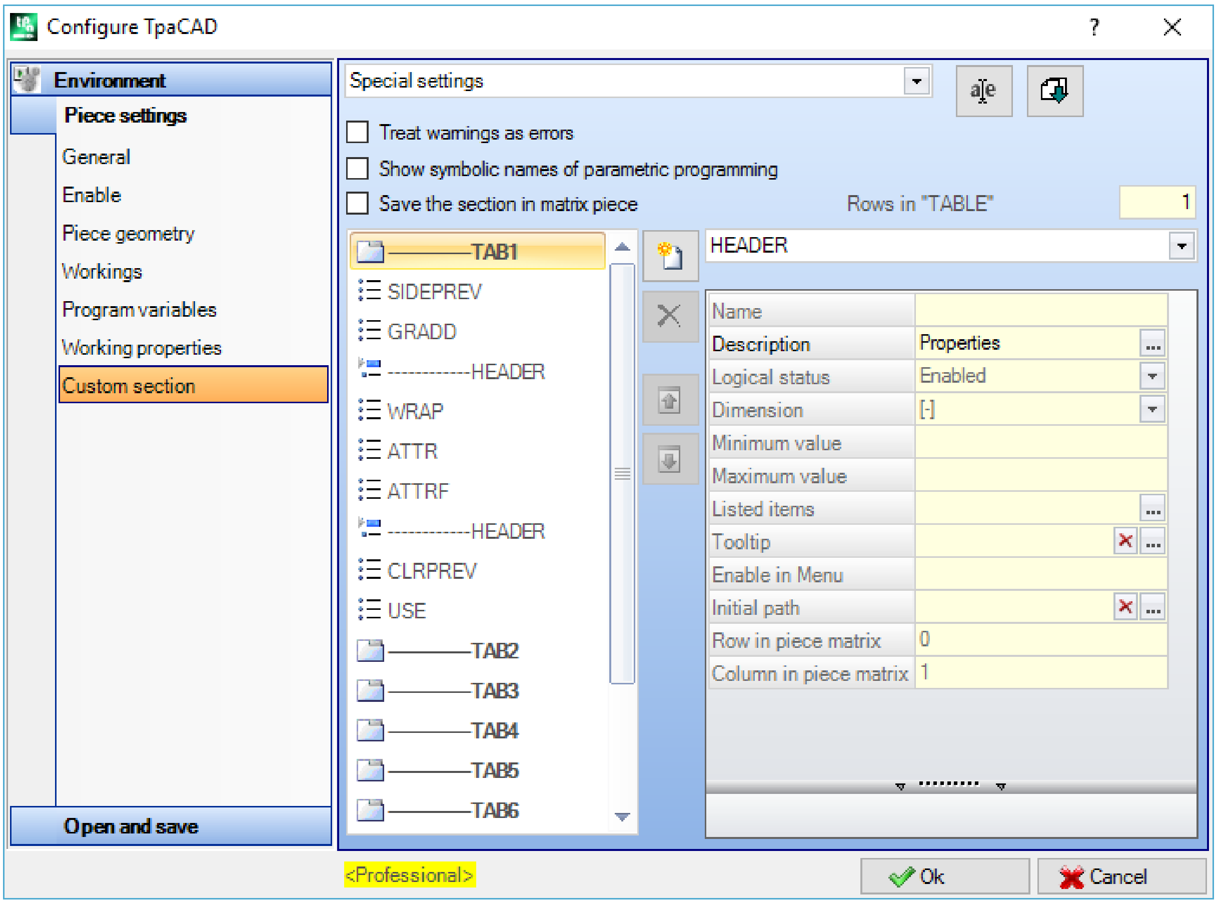
corresponds to the selection "Condition the view"

corresponds to the selection " Delete the working after the end of the development"

**Sequences:**

- **Enable**: select to manage the sequences of the execution. The maximum value the property can take is very large and it is always transparent for the operator, because it is managed automatically. If the property is not managed, the file is reset anyway when the program is read.
- **Optimization Flage:** it sets the management of an optimization flag in the assignment of the sequences. The interpretation of this information is totally delegated to an optimisation module, as it generally happens for the sequences.

## Custom Sections



Configure TpaCAD

Environment
Piece settings
- General
- Enable
- Piece geometry
- Workings
- Program variables
- Working properties
- Custom section

Open and save

Special settings

□ Treat warnings as errors
□ Show symbolic names of parametric programming
□ Save the section in matrix piece

Rows in "TABLE" 1

TAB1
SIDEPREV
GRADD
------------HEADER
WRAP
ATTR
ATTRF
------------HEADER
CLRPREV
USE
TAB2
TAB3
TAB4
TAB5
TAB6

HEADER

| | |
|---|---|
| Name | |
| Description | Properties |
| Logical status | Enabled |
| Dimension | [-] |
| Minimum value | |
| Maximum value | |
| Listed items | |
| Tooltip | |
| Enable in Menu | |
| Initial path | |
| Row in piece matrix | 0 |
| Column in piece matrix | 1 |

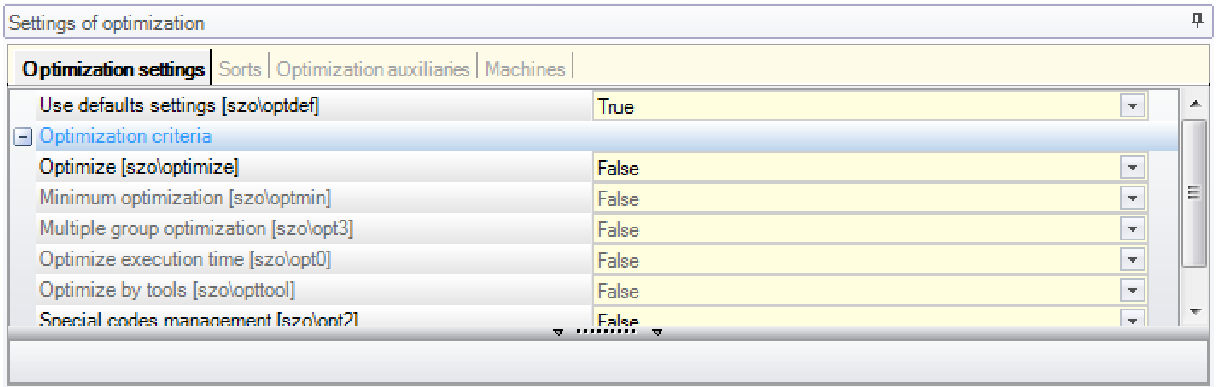
4 custom section can be configured for the piece as follows:

- Special settings
- Additional Info section
- Section of constraints
- Optimization settings

It is possible to modify the configuration also of the non-enabled sections (see page **Piece Settings->Enable->Enable sections**).
The meaning assigned to the custom section corresponds to the interpretation and the application to the module of Piece optimization and/or of the plant cycle. For this reason it must be arranged during the

design time of the plant. Remarkable interpretations in TpaCAD environment can be taken from the configuration of particular field in the section of the Special settings, as mentioned later.

To customize the section, the selection must be moved on the left list. The Optimization settings are displayed in the picture below:



Settings of optimization

| | |
|---|---|
| Use defaults settings [szo\optdef] | True |
| Optimization criteria | |
| Optimize [szo\optimize] | False |
| Minimum optimization [szo\optmin] | False |
| Multiple group optimization [szo\opt3] | False |
| Optimize execution time [szo\opt0] | False |
| Optimize by tools [szo\opttool] | False |
| Special codes management [szo\opt2] | False |

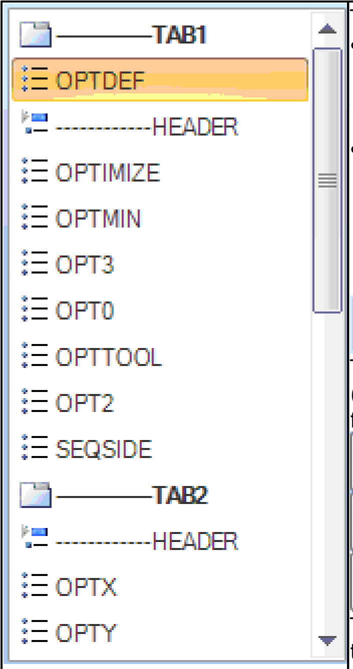
- The button a|e allows the setting of the message to be used as a section header (Example: Optimizations). A window opens, showing the message recorded in the CADAUX custom message file.
- **Treat warnings as errors:** select to report as errors the diagnostic situations arising in settings outside of intervals set. This situation concerns the fields of a type for which it is meaningful to set a range of values (integer, long, double, string parametric) and determines also a fail during the optimization program.
- **Show symbolic names of parametric programming:** if selected, it enables the view of the symbolic names of the section fields next to the descriptive messages while editing. For example in the picture:
  - "Use default settings [szo\optdef]" the symbolic name =”szo\optdef” is shown
  - “Optimize [szo\optimize]” shows the symbolic name =”szo\optimize”
  - “Minimum Optimization [szo\optmin]” shows the symbolic name =”szo\optmin”.

  The symbolic names are used in the parametric programming. For instance, ”szo\optdef” is equal to 0 with programming on “False” and equal to 1 with programming on “True”.
  Not every field is enabled to use the corresponding symbolic name. The fields with typology different from INTEGER, DOUBLE, LIST, PARAM. STRING are not included.
  **WARNING**: This selection is not available in case of *Essential* functionality.
- **Save the section in matrix piece:** if selected, it enables the custom section to be stored in the matrix piece. This section stores up to 9 rows of information and each row can contain 50 fields. The association of the single fields with the stored rows and columns will be defined later.
- **Rows in “TABLE”:** it sets the number of rows managed in a page with table settings (see later). Set a value between 1 and 16.

In a custom section up to 150 entries (fields) can be assigned, which can be arranged in 10 tabs:

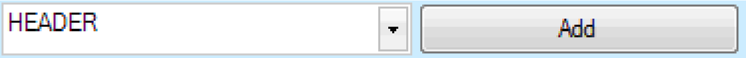
- the first 9 tabs present an arrangement of the entries in the lists with possible grouping in nodes;
- the last tab (the tenth one) presents an arrangement of the entries in the table, where the number of the columns corresponds to the entries assigned in the tab, but up to a maximum of 20 only, while the row number is the number set in the field **Rows in “TABLE”**. The TABLE **tab** sets some limitations in assigning the fields: for more details see later. In the TABLE **tab** only the names of the first row fields are set. The name of the fields of the next rows are created in automatic and transparent way by TpaCAD up to a maximum of 320 fields (20 columns for 16 rows).

In the picture of the section Optimization settings 3 tabs, organized in lists (Optimization settings, Sorts, Optimization auxiliaries) are set.



The list defines the organization of a section:

- it is initialized with 10 items, which are related to the sections allowed (TAB1,….TAB9: for the first 9; TABLE for the last tab). They are items which cannot be deleted and their reciprocal order cannot be modified (TAB1, then TAB2, … and finally TABLE).
- The remaining items (up to 150 at the most) correspond to the fields that are configured for this section: each item belongs to the tab where it is inserted. As in the picture: the entries from "OPTDEF" to "SEQSIDE" belong to the TAB1 tab. A tab without assigned items will remain invisible.



To insert a field, move the selection on the list, select the field typology (HEADER,…) in the list provided, then select the button **[Add]**. The new field is inserted after the current position.

⇧ ⇩ ✕

To delete or move a field in the list, move the selection on the list and use the buttons provided.
In the **TABLE** tab the user can assign up to 20 entries with limited typology.

Move the selection on the list to customize the single item (tab or section field), whose assigned fields are located in the right list:

- **Name:** it sets the name which identifies the field (max. 10 characters, alphanumeric, first non-numeric character, capital). Examples: "OPT0", "DEFSIDE". The name must be assigned as univocal in the section and it is compulsory for every field typology, but the HEADER typology.
- **Typology:** it sets the field typology to be selected in a list:

| | |
|---|---|
| HEADER | Header and grouping field of other fields. According to informatic terms the typology corresponds to a node. The following assigned fields for the same tab and until a following field of HEADER technology are grouped in the node itself. It's possible to use the field as a separator instead than a node: in this case, the field *Description* has to be left not assigned.<br>This typology cannot be used in the **TABLE** tab. |
| DOUBLE | numeric field with decimals |
| INTEGER | numeric field without decimals |
| STRING | generic field to which no specific control is applied. The maximum length of the field is 100 characters in capitol letters. |
| LIST | Field creating a list. Its minimum form is made of 2 items (e.g.; "False"/"True) and no more than 20 sections are managed. The value associated with the field is the index of the selection: integer between 0 and 19. |
| ORDERED LIST | Field made up of a list of ordered items. Its minimum form is made of 2 items; no more than 10 items in the list are managed. The value associated with the field is a string showing the order in which the items are displayed.<br>This typology cannot be used in the **TABLE** tab. |
| PARAM STRING | numeric field, with possibility to parametric assignment (allowed parametrizations are: technology, piece dimensions, execution mode, o,v,r - variables). The maximum length of the field is 100 characters in capitol letters. |
| COLOR | Color selected from a graphic palette. The value associated to the field is a whole number linked to the chosen color.<br>This typology cannot be used in the **TABLE** tab. |
| FILE | Alphanumeric field. It describes the path of a file + the name of the file. The maximum length of the field is 250 characters.<br>This typology cannot be used in the **TABLE** tab. |
| PATH | Alphanumeric field describing a path. The field can assign up to 250 characters.<br>This typology cannot be used in the **TABLE** tab. |

- **Description:** message to be used as a field header (Examples: Use defaults settings). If you select the button ..., the window of the list of messages stored in the custom file messages CADAUX (in the folder tpacadcfg\custom) opens. From the available messages the user can choose, modify or enter the message required. This is the only field which can be assigned for a tab field. In the **TABLE** tab it corresponds to the column heading.

- **Status:** it sets the enabling status of the field:
  **Enabled**: the field is managed
  **Non editable**: the field is displayed, but cannot be modified. This status allows some protected fields to be assigned and its use is linked to the external generation of programs in TpaCAD format (e.g. to import from an external format).
  **Disabled**: the field is not shown in the window of the custom section. This status allows the field control to be deactivated, without changing the assignment in the configuration.

- **Change level:** sets the Level to enable the change of the field**.** If the current level is lower than that set, the field may be displayed but not changed. This setting is significant only if the field is managed (field **State** is set as **Enabled**).
- **Minimum value:** it sets the minimum value, which can be assigned to the field. If the field is left empty, the minimum value set is not checked. The field is significant for the following typologies: DOUBLE, INTEGER, PARAM. STRING.
- **Maximum value:** it sets the maximum value, which can be assigned to the field. If the field is left empty, the maximum value set is not checked. The field is significant for the following typologies: DOUBLE, INTEGER, PARAM. STRING.
  For the typology LIST, ORDERED LIST the maximum value is the number of the item in the list, less 1. The maximum value that can be assigned is 9 and corresponds to 10 items in the list. For example, for a list made of 2 items, the maximum value = 1 must be set.
- **Listed items: :** it sets the message to be used in the field LIST or ORDERED LIST (select the button ... to assign the message). The selected message must list all items of the list, in the order corresponding to the values (0,1...) and split by the character '%'. For instance, "Rare%Occasional% Frequent".
- **Tooltip:** it sets the message to be shown as an aid in the process of item setting (select the button ... to assign the message).
- **Enable in Menu:** it sets the condition to make the field changeable, in form of "NAME%val", where:

| | |
|---|---|
| **NAME** | is the name of another field of the section , whose type must be:<br>(DOUBLE, INTEGER, LIST, PARAM STRING, STRING). The setting is ignored, if the NAME shown is not valid or if the filed type is not valid. |
| **%** | comparison character. Valid characters are: **'=', '#', '<', '>'**. The setting is ignored, if no valid character is recognized |
| **val** | value |

This field is not significant in the **TABLE** tab.
**Example:**
"OPTDEF=0" the field is enabled, if the value of the field called "OPTDEF" is =0
"OPTH<700.5" the field is enabled, if the value of the field called "OPTH" is less then 700.5 (typical: OPTH is of type: DOUBLE).

- **Initial path:** the path by default in the assignment of the field of FILE type.
- **Row in matrix piece:** it sets the number of the row, where the field information of the custom section must be written in the file of the matrix piece. The number can be edited for the first 9 tabs and must be between 1 and 9. If number = 0, no storage is made in matrix piece.
- **Column in matrix piece:** it sets the number of the column, where the field information of the custom section must be written in the file of the matrix piece. The field accepts values from 1 to 50.

If you select the button , the custom section in stored in the file of the matrix piece according to the following rules:

- the number of rows, whose value of each field is written, is deducted from the position of the field inside the tabulation to which it belongs (from 1 to 9);
- in each tabulation group the column is progressively assigned from 1 and up to 50 field each row.

**Remarkable information in the section of the Special Settings**
At the moment remarkable information are recognized, as follows:

- **tooling**:
  - name: "ATTR"
  - typology: Integer or List

When the typology *List* is selected and if no message is assigned for *Listed items*: the list can be assigned by requiring the name of the tooling from the technological component.
The selection of tooling is effective, if the technological parameters loaded include their assignment. A tool modification provides the complete upgrade of technology and program.

- **Tooling**:

- name: “ATTRNAME”
- typology: File

The field is recognized as an alternative to the previous one; the aim is to select a tooling by indicating a file to read the tooling itself.

- **Program usage frequency**:
  - name: “USE”
  - typology: Integer or List

The selection is effective, if the optimization process includes the assignation and interpretation. The recognized values are: 0=rare usage; 1= occasional usage; 2=frequent usage;

- **Preview color**:
  name: “CLRPREV”
  typology: Color
  The selection is used by the component of Graphic preview to assign the color of the panel and optionally also in TpaCAD, for the current program, as it is from the authorization in Customizing TpaCAD.

- **Preview pattern:**
- Name: "CLRNAME"
- Type: File
  The field is recognized as an alternative and with previously with respect to the previous one, for a selection of graphic pattern for the filling of the panel.
  The selection is used by the component Graphical preview to assign the fill pattern of the panel and optionally also in the TpaCAD, for the current program, as it is from the authorization in Customizing TpaCAD.

- **Application of the additional graphic elements** (arrows, extreme points, overall 3D-dimensions) in the representation of the program:
  - name: “GRADD”
  - typology: Integer or List

The selection is effective both in TpaCAD environment and in component of Graphic Preview with the following interpretation:

- 0: applies the settings by default
- 1: deletes the graphics of the additional elements (arrows and extreme points applied to profile segments, three-dimensional overall dimensions)
- >1: adds the graphics of additional elements.

For instance, the field can be set from a conversion format module (in the reading process) applied to an ISO program, which generally can assign a hight number of segments (for instance: 100.000 or more) and whose "graphic comprehension" can be compromised by the display of arrows, extreme points, overall 3D-dimensions.

- **Preview** in the representation of the program:
  - name: “SIDEPREV”
  - typology: Integer or List.

The selection is effective both in TpaCAD environment and in component of Graphic Preview; view in 3D piece display with the following interpretation:

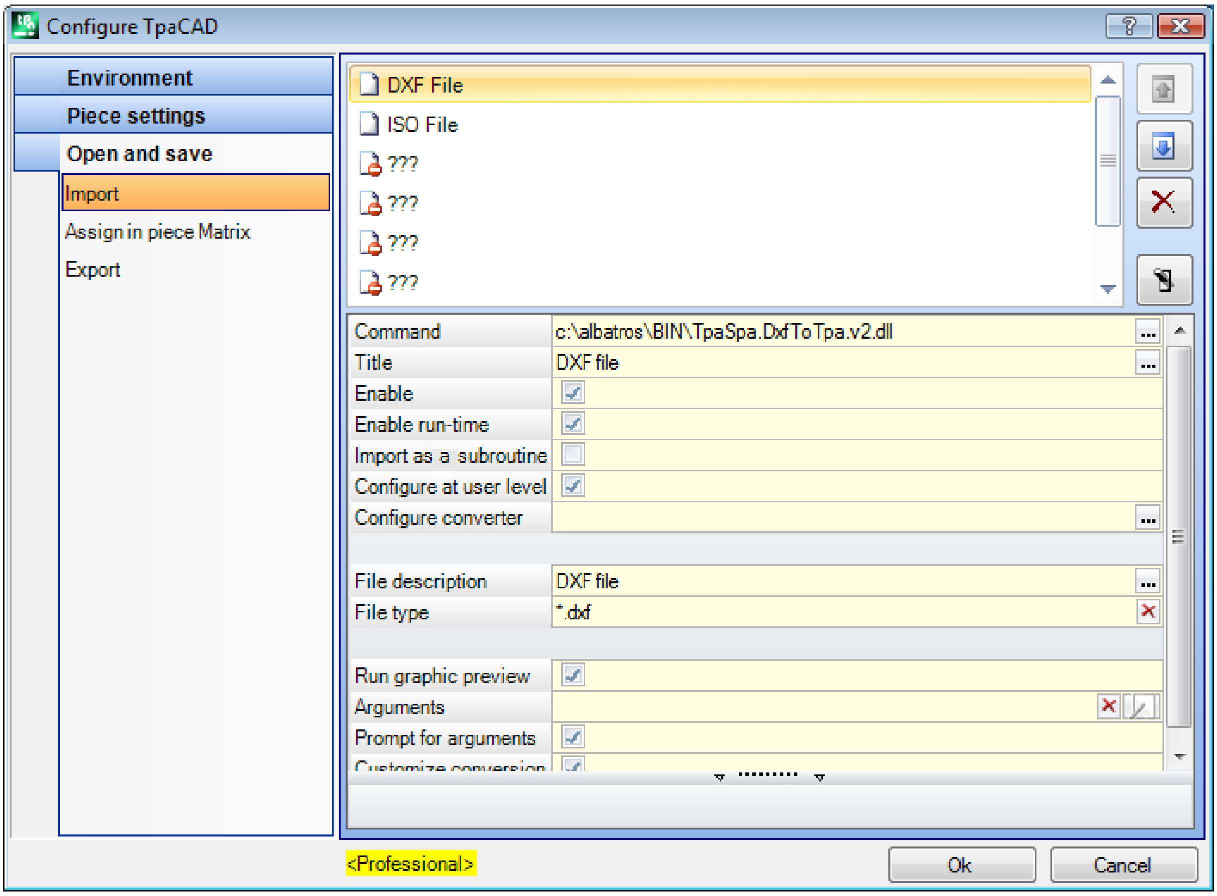
- 0 (or invalid value): applies the orientation by default of the piece
- value between 1 and 6 assigns the view of the face where the program is opened.

If managed, the a.m. remarkable information are shown in the matrix piece, at the header row, whose code is [2500] and to the columns:

- (41) tooling field;
- (42) special geometry field of shaped piece
- (43) program frequency use
- (44) preview colour
- (45) application of the additional graphic elements
- (46) preview

# 1.3 Open and save

## Import



| | |
|---|---|
| Command | c:\albatros\BIN\TpaSpa.DxfToTpa.v2.dll |
| Title | DXF file |
| Enable | ☑ |
| Enable run-time | ☑ |
| Import as a subroutine | □ |
| Configure at user level | ☑ |
| Configure converter | |
| File description | DXF file |
| File type | *.dxf |
| Run graphic preview | ☑ |
| Arguments | |
| Prompt for arguments | ☑ |
| Customize conversion | ☑ |

This page shows the configuration of the programs to import non-TpaCAD format files in a TpaCAD environment. You can configure up to 8 import programs. To add a conversion, you must select the item of the list and then compile the fields of the table below:

- **Command:** it inserts the complete pathname of the conversion program. Select the button … to open the window of resource manager. Files with extension *.exe, *.dll. can be selected. The button ✕ resets the field, by deactivating the importer. The import program has ProgId "ExecutableName.Converte", whereExecutableName is the name of the component. In case of return with confirmation, the **SettingsArgs** method (see later) is invoked to read and program any possible default settings.

- **Title:** it inserts the description of the program (in the picture: DXF Files). If you select the button …, the list of the messages stored in the custom file messages CADAUX (in the folder tpacadcfg\custom) is displayed. Messages, among those the message required can be chosen, modified or entered, are listed;
- **Enable:** if selected, it enables the management of the converter in TpaCAD
- **Enable run-time:** if selected, this option enables the management of the converter during the execution. Anyway, the converter must be enabled (**Enable** field selected).
- **Import as a subroutine**: if selected, for the imported programs the typology of Subprogram is assigned
- **Configure at user level**: if selected, it allows the converter to be configured also in TpaCAD customization process, by accessing from a level lower than Constructor.
- **Configure converter** : select the button … to start the configuration of the converter. **WARNING**: the configuration of the converter is managed by the converter itself; therefore, the configuration, if not correctly assigned, could fail.
- **File description**: it sets the message which is shown while opening the program at the option File Type. (select the button … to assign the message)
- **File type**: it lists the typology of the files which can be converted. Examples of valid assignments: "*.*" , "*.cnc", "*.cnc;*.abc"

- **Run graphic preview**: if selected, it enables the graphic representation of the program already converted
- **Arguments:** string of arguments to pass to the converter tool. Select the button on the right side of the field to query the conversion program for the default argument (see later, **SettingsArgs** method).
- **Prompt for arguments:** if selected, it enables the assignment of the arguments at each conversion (from File open window). **WARNING**: This selection is not available in case of *Essential* functionality.
- **Customize conversion** if selected, it enables the customisation of each single conversion. From the file open window, customization can be required. **WARNING**: This selection is not available in case of *Essential* functionality.

⬆
⬇ Buttons to change the order of the converters.

✕ Select to remove the converter

Button to check the creation of an instance of the component: a message reports the test result.

Of the import modules following methods are used:

- **long InfoToAlb (string szPathFile, ref string szDescr, ref short prgUnit, ref double prgDimX, ref double prgDimY, ref double prgDimZ)**
- **long InfoToAlb (string szPathFile, ref string szDescr, ref short prgUnit, ref double prgDimX, ref double prgDimY, ref double prgDimZ, ref string szError)**

  the method is invoked to check, if the file shown has a format recognised for a conversion to TpaCAD and if it reads the general information of the workpiece. The second prototype can return a message of an error situation. The acquisition of the piece dimensions may also cause a partial or full interpretation of the file.

  Method return value: 0, if the format is recognised as valid, otherwise error number

| | |
|---|---|
| string szPathFile | pathname of the file (selected from File->Open menu) |
| ref string szDesc | returns the comment to the program |
| ref short prgUnit | returns the unit of measurement |
| ref double prgDimX, prgDimY, prgDimZ | returns the dimensions of length, height, thickness |
| ref string szError | returns an error message (if the return value is not equal to 0). |

- **long TestToAlb (string szPathFile)**
- **long TestToAlb (string szPathFile, ref string szError)**

  the method is invoked to check, if the file shown has a format recognised for a conversion to TpaCAD and if it reads the general information of the piece. It is invoked also if the **InfoToAlb** method is not assigned. The second prototype can return a message of an error situation.

  Method return value: 0, if the recognized format is valid; otherwise error number.

| | |
|---|---|
| string szPathFile | path file + file name (selected from menu ***File->Open***) |
| ref string szError | returns an error message (if the return value is not equal to 0). |

- **long ImportToAlb (string szPathFileCst, string szPathFileOut, string szArgs, string szPathConfig, ref string szError)**

  the method is invoked to convert the indicated file to TpaCAD.

  Method return value: 0 if the recognized format is valid; otherwise error number.

| | |
|---|---|
| string szPathFileCst | pathname of the file in original format (selected from menu File->Open) |
| string szPathFileOut | pathname of the file to write (in TpaCAD format) |

| | |
|---|---|
| string szArgs | string of arguments.<br>Format and meaning of the fields:<br>• '/': field name<br>• space: optional separator<br>• the string is case-insensitive<br><br><u>Some fields are added automatically:</u><br>"**/?**=.." greater than 0 the conversion manages the customizations used by the import module (example for DXF converter: layers to import or to exclude)<br>"**/key**=.." greater than 0 the advanced conversion level is enabled, if used by the import module (example for DXF converter: assignment of workings from layers and/or blocks)<br>"**/lng**=.."current language (three letters abbreviation. Ex.: "ITA", "DAN")<br>"**/lngdef**="language by default (three letters abbreviation: "ITA", "ENG", "FRE", "GER", "SPA")<br>"**/cad_sysgeo**=.."piece geometry (0=faces in transparency, 1=custom system)<br>"**/cad_syseps**=.."epsilon multiplier (multiplies 0.001 for a program in mm, to calculate the system epsilon)<br>"**/cad_sysfaces**=.. "string of enabled real faces ("123456" for all)<br>"**/cad_syssep**=.."character used as decimal separator ('.'/',' )<br>"**/cad_sysdecm**=.."number of decimals in a program in millimetres<br>"**/cad_sysdeci**=.." number of decimals for a program in inches<br>"**/cad_sysxz**=.."assignment for xz (0=Xz; 1=Zx) plane<br>"**/cad_sysz**=.." positive value (1), if the z-axis enters the faces<br>"**/cad_sysbeta**=.."positive value (1): inverts the slewing axis sign of the aggregates<br>"**/cad_sysfeedm**=.." linear unit speed in [mm]: 0=mt/min;1=mm/min<br>"**/cad_sysfeedi**=.." linear unit speed in [inch]: 0=inch/sec;1=inch/min<br>"**/**..." arguments assigned in the Argument field follow |
| string szPathConfig | folder from which the functioning assignments can be read/written (example for DXF converter: folder from which the INI files of assignments of the conversion customizations can be read) |
| ref string szError | error message |

The method is invoked also after an instance of the control, being szPathFileCst="" and szPathConfig assigned valid, in order to allow the possible storage of the folder from which one can read/write the assignments of functioning. The folders archived in this way can be used in the functioning of the ***InfoToAlb*** method***.***

- **long Settings (string newFolder, string newMachine, string szPathConfig, string szArgs)**
  the method is invoked to manage the customisations used by the conversion module. This method can be implemented if a record for the customisation of the import module is available with an automatic reading process of the same at each conversion request.
  Method return value: 0 if the result is positive, otherwise error number.

| | |
|---|---|
| string newFolder | Tpa.ini - file search folder (if the string is empty, the default folder is taken) |
| string newMachine | name of the machine (section in Tpa.ini file section, if the string is empty, the name by default of the machine is taken) |
| string szPathConfig | folder from which the functioning assignments can be read/written (the example for DXF converter is the folder from which the INI files of assignments of the conversion customizations can be read) |
| string szArgs | string of arguments.<br>Format and meaning of the fields: |

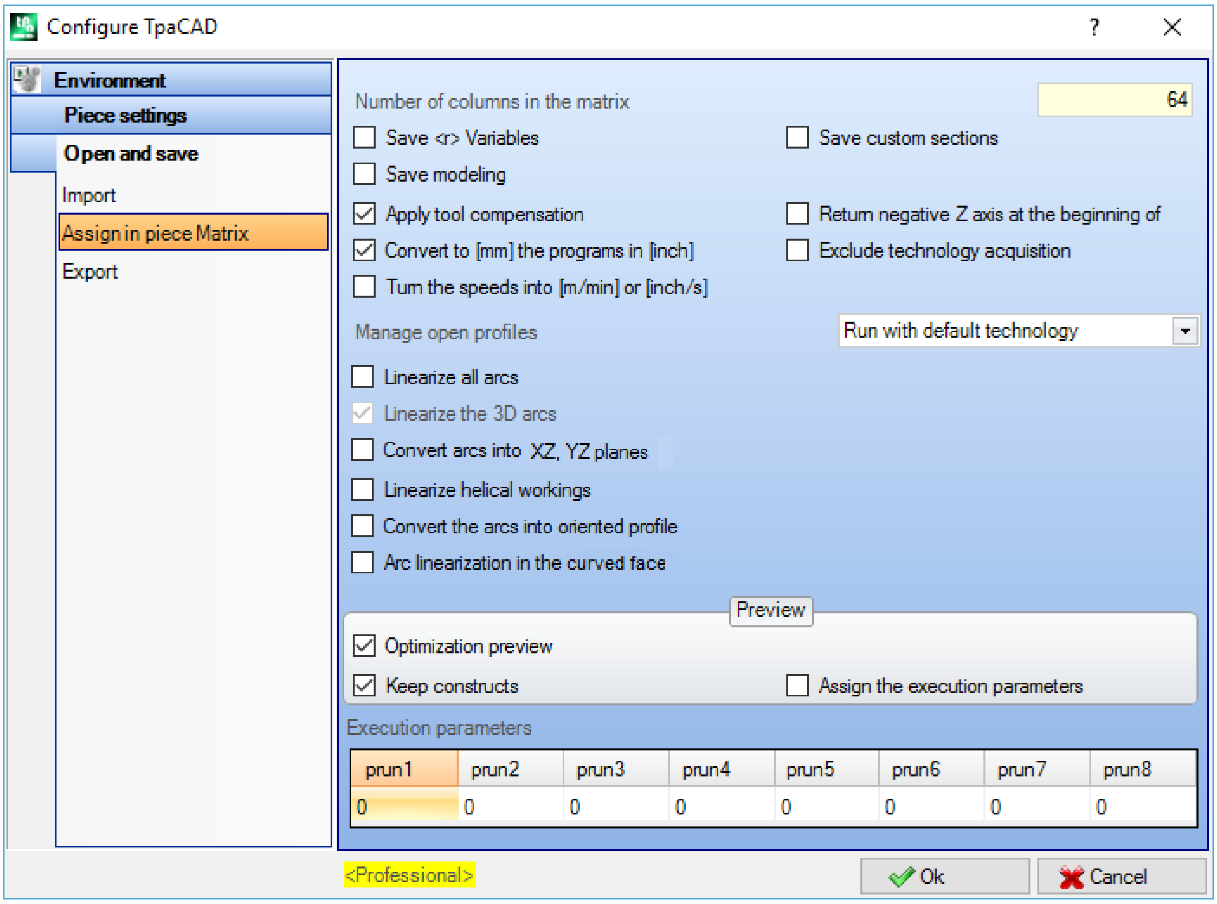
| | |
|---|---|
| | • '/': field name<br>• space: optional separator<br>• the string is case insensitive<br>"**/key**=.." =.." >0 enables the advanced layer, if used by the import module (example for DXF converter: assignment of workings from layers and/or blocks)<br>"**/mode**=.." =0 enables the customizations of the module as an importer. The field is managed for a module managing the conversion both in reading and writing<br>"**/lng**=.."current language (three letters abbreviation. Ex.: "ITA", "DAN")<br>"**/lngdef**="default language (three letters abbreviation: "ITA", "ENG", "FRE", "GER", "SPA")<br>"**/**..."assigned arguments in the field Arguments follow |

**long SettingsArgs (bool bMode, ref string szArgs)**
the method is invoked to manage the customizations run by the conversion module. We recommend to implement this method only if a customization through ***Settings***method is not managed. This method can be implemented, if a customization of the import module that must be directly specified on each request of conversion (through the string of the arguments), is managed.
Method return value: 0 if the result is positive, otherwise error number.

| | |
|---|---|
| bool bMode | *false*, if it is invoked as an importer<br>true, if it is invoked as an importer<br>(The allows you to assign a module managing the conversion both to reading and writing: |
| string szArgs | The string returns the default settings managed by the module (about the format, reference is made to previous information). If, for instance, the module interprets two numeric customizations transmitted as "/aaa=… bbb=…", the method must return the string containing the values that are interpreted in case of non-transmitted arguments (example: "/aaa=1 bbb=0"). |

- **void Dispose()**
  The method is invoked before releasing the class instance, to release the managed and the unmanaged resources, such as handle, connections to database,...).

## Assign in piece matrix



Number of columns in the matrix: 64

☐ Save <r> Variables ☐ Save custom sections

☐ Save modeling

☑ Apply tool compensation ☐ Return negative Z axis at the beginning of

☑ Convert to [mm] the programs in [inch] ☐ Exclude technology acquisition

☐ Turn the speeds into [m/min] or [inch/s]

Manage open profiles: Run with default technology

☐ Linearize all arcs

☑ Linearize the 3D arcs

☐ Convert arcs into XZ, YZ planes

☐ Linearize helical workings

☐ Convert the arcs into oriented profile

☐ Arc linearization in the curved face

☑ Optimization preview

☑ Keep constructs ☐ Assign the execution parameters

| prun1 | prun2 | prun3 | prun4 | prun5 | prun6 | prun7 | prun8 |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

This page shows the following criteria for the creation of the **Matrix piece** for a program:

- **Number of columns in the matrix:** number of columns of the piece matrix. The field can only accept values between 64 and 100. **Note:** the matrix piece is interpreted by the Piece optimizer; for that reason it is very important to define the number of the columns during the phase of plant design.
- **Save <r> variables:** if enabled, it saves the <r> variables in the piece matrix.
- **Save modeling:** if enabled, this option saves the data of the piece modelling
- **Save the custom sections:** if enabled, it saves the data of the custom section in the piece matrix.
- **Apply tool compensation:** if selected, it enables the tool compensation during the creation of a piece matrix.
- **Convert to [mm] the programs written in [inch]:** if selected, it enables the conversion of the programs from inches [inch] to millimetres during the creation of the piece matrix.
- **Turn the speeds into [m/min] or [inch/sec]:** if this option is selected, it enables the conversion of variables and parameters whose meaning is linear speed rate in in [m/min], if the piece matrix is created in [mm], or [inch/sec] if the piece matrix is created in [inch].
- **Return negative Z-axis at the beginning of the faces:** it enables the inversion of the programmed coordinates for Z-axes, if it interprets positive values of Z at the beginning of the piece (see the paragraph: ***Piece settings ->Piece Geometry***).
- **Exclude technology acquisition:** if selected, it disables in the matrix the automatic integration, by the technological component, of programming for tool. This selection is active only during the creation of the piece matrix and concerns the information on the diameter. For instance, in case of a drilling programmed for a tool:
  - non-selected field: the real tool diameter is shown in the matrix, even though it is differently set in the working.
  - with selected field: the actual diameter of the tool is only shown in the matrix, if it is not set in the working, otherwise the value set in the working is shown.

**Manage the open profiles:** it sets how the open profiles must be converted (profiles without setup).
Three options are possible:
**Run with default technology**: the open profiles are shown in piece matrix and to them the technology by default is assigned (if any technology by default is not available, an error is reported). The assignment concerns also the case of profiles beginning with a geometric Setup.
**Exclude from the execution**: the open profiles are not shown in the piece matrix.
**Report an error** : it reports an error, when an open programmed file is found.

- **Linearize all the arcs**: if selected, it requests the fragmentation and linearization of all arcs during the creation of the piece matrix
- **Linearise all the XYZ-arcs:** it requests the fragmentation and the linearisation of the arcs assigned in a generic xyz plane during the creation of the piece matrix. This item is always active and cannot be changed. Note: an arc assigned with operating code of xyz-arc (ex: A10), if it is calculated on one of the three coordinated plans of the face (xy,xz, yz), it is given in the piece matrix:
  - as an arc (ex: xz), if the fragmentation of the arcs (xz,yz) is not required;
  - as a polyline, if the fragmentation of the arcs in the corresponding plane.
- **Convert arcs into segments in XZ, YZ-planes:** if selected, it requests all the xz and yz-arcs to be linearised and to be fragmented in line segments.
- **Convert arcs into segments in XZ, YZ-planes** if selected, it requests all the xz and yz-arcs to be linearised and to be fragmented in line segments.
- **Linearize helical workings:** if selected, all the arcs with a non-zero depth component can be linearised and fragmented.
- **Convert the arcs into oriented profile:** if selected, the arcs assigned in profiles that have an oriented setup code, can be linearised and fragmented.
- **Arc linearization in the curved face:** if selected, it requests the assigned arcs in curved faces to be linearised and to be fragmented or assigned as surfaces.

In the frame below the function of **Optimization preview** is configured **(of optimization and graphic)**:

- **Optimization preview:** select to enable the command in *Menu Display*. The command selection starts the compilation of the current program in a execution environment (the variable argument of parametric programming *prgrun* is equal to 1 ) and gives its graphic representation. **WARNING**: This selection is not available in case of *Essential* functionality.
- **Keep the Constructs:** select to maintain displayed the Construct workings which always verify the logical conditions. With non-active selection, all construct working are instead excluded from the view of *Optimization and Graphic Preview* ;
- **Assign Execution parameters:** select to enable the application of the optimization parameters, as set in the following table. If the selection is inactive, all the Execution Parameters remain null and the views of *Optimization and Graphic Preview* are activated.

The settings *Keep the Constructs* and *Assign Execution parameters* remain null if the views of *Optimization and Graphic Preview* are activated.

The settings *Keep the Constructs* and *Assign Execution parameters* are applied:

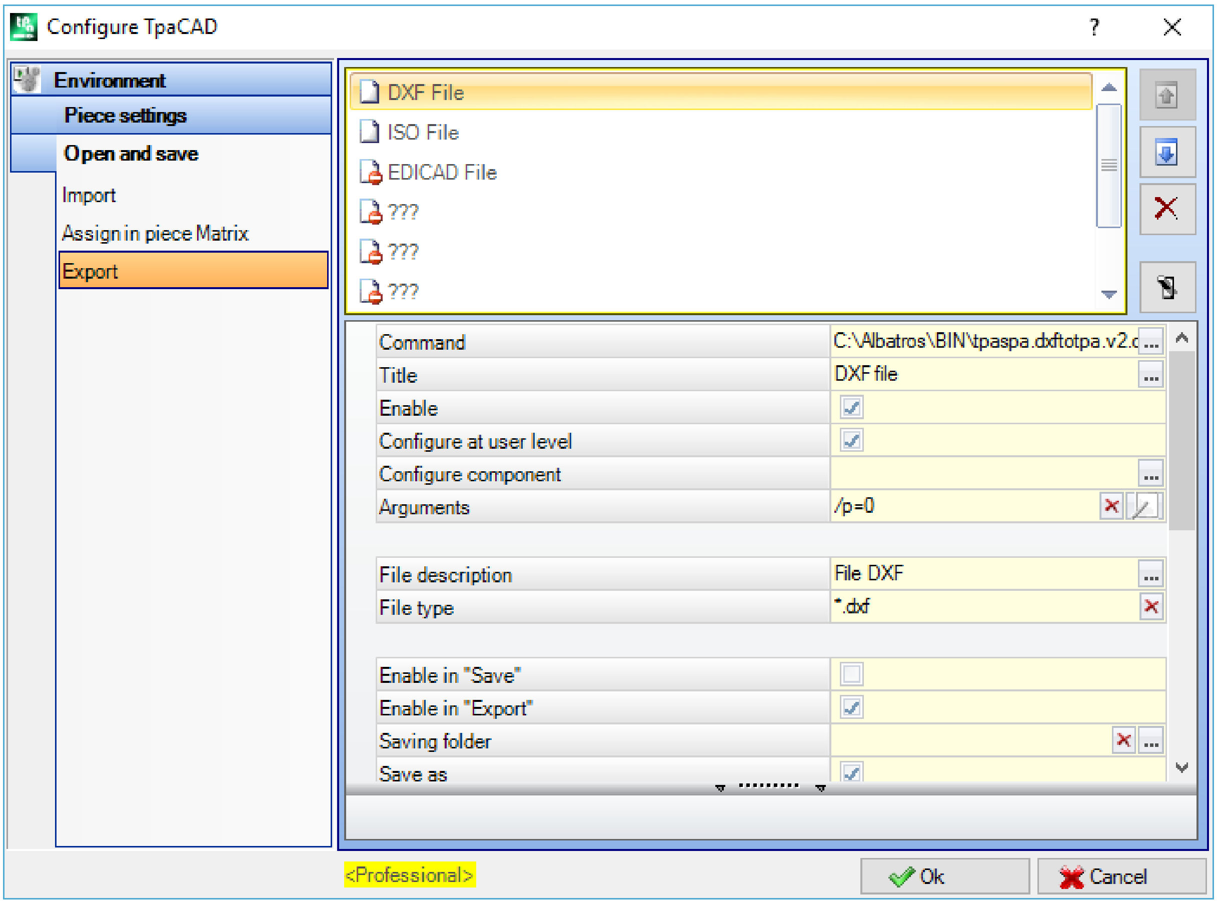
- to the view of *Optimization Preview*, if enabled;
- to the component of *Graphic Preview* provided with the TpaCAD installation, if it's used and it's not specified in a different way.

**Execution Parameters**: parameters of entire type (each field accepts a value between -1000000 and 1000000), which have corresponding variable arguments of parametric programming, as shown for the column header (prun1, ..., prun8). The settings in the table are likely to correspond to the values used during the execution of the programs.

The values assigned here can be used in execution of:

- *Optimization Preview* and *Graphic Preview* (see above)
- *Export* format (see next paragraph).

## Export



| | |
|---|---|
| Command | C:\Albatros\BIN\tpaspa.dxftotpa.v2.c |
| Title | DXF file |
| Enable | ☑ |
| Configure at user level | ☑ |
| Configure component | |
| Arguments | /p=0 |
| File description | File DXF |
| File type | *.dxf |
| Enable in "Save" | □ |
| Enable in "Export" | ☑ |
| Saving folder | |
| Save as | ☑ |

In this page the programs to export from TpaCAD environment to non-TpaCAD format are configured. Up to 8 export programs can be configured.
To add a conversion, you must select the item of the list and then compile the fields of the table below:

- **Command:** enter the complete pathname of the conversion program. Select the button ... to open the window of resource manager. Files with extension *.exe, *,dll. The button ✕ resets the field and deactivates the exporter. The ProgId of the export program is "ExecutableName.Converte", whereExecutableName is the name of the component. In case of return with confirmation, the **SettingsArgs** method is invoked (see later) to read and set possible settings by default.
- **Title:** inserts the description of the program (in the figure: DXF Files). If you select the button ... the window of the list of the messages stored in the CADAUX custom file messages (in the tpacadcfg\custom folder) opens. Messages are listed, among those the message required can be chosen, modified or entered
- **Enable:** if selected, it enables the management of the converter in TpaCad
- **Configure at user level**: if selected, it allows the converter to be configured also in TpaCAD customization process, by accessing from a level lower than Constructor.
- **Configure component:** select the button ... to start the configuration of the converter. ***WARNING:*** the configuration is managed at the level of the converter itself; therefore, if the selection is not correctly assigned, cannot be effective.
- **Arguments:** string of arguments to pass to the converter tool
- **File description**: it sets the message which is shown during the saving process at the option File Type (select the button ... to assign the message)
- **File type:** file typology to be set by default during the file saving process. The user can assign a string, which should observe the following syntax: "*.EXT" where EXT assigns the extension, that will be recognized. A "*.*" extension cannot be assigned
***WARNING:*** NOTE: the export module can modify anyway the extension of the generated file.
- **Enable in "Save"** select to activate the conversion during the saving process of the TpaCAD file.
***WARNING***: the selection can be active for one only of the assigned converters and it is operative only if the optimisation from an external program does not seem enabled.

- **Enable in "Export"** select to activate the conversionby means of the command **Export** from the menu Application
- **Saving folder:** it sets the folder of the converted file recording (select the button ... to open the window of resource manager). If the field is empty, the folder of the temporary files is used, as it is assigned in the TPA.INI file.
- **Save as...** select to activate the opening of the window **Save as** in such a way that the user can choose the folder and the name of the program created by the conversion. Let the option unselected to save in the **Saving folder** and with the name by default which correspond to the name of the input file without extension.
- **Convert from the TCN - file** select to activate the conversion directly from the file in ASCII format of TpaCAD(TCN file). In this case a conversion request saves the TpaCad file in any case. If the option is not selected, the conversion is applied to the program piece matrix, that is obtained by applying the rules that correspond to the **Assign in the Matrix Piece.group.**

⇧
⇩ buttons to change the order of the converters.

✕ Select to remove the converter

Select the button to test the creation of an instance of the component: a message reports the test result.

**<u>Assign in piece matrix</u>**

The items of the node define the criteria according to which the piece matrix of a program is created; the conversion is not enabled during the saving or from the file in ASCII format. The meaning of the items corresponds to what has been seen for the setting of the previous page:

- **Apply tool compensation:** if selected, it enables the tool compensation during the creation of a piece matrix.
- **Convert to [mm] the programs in [inch]:** if selected, it enables the conversion of the programs from inches [inch] to millimetres during the creation of the piece matrix.
- **Turn the speeds into [m/min] or [inch/sec]:** if this option is selected, it enables the conversion of variables and parameters whose meaning is linear speed rate, if the piece matrix is created in [mm], or [inch/sec] if the piece matrix is created in [inch].
- **Return negative Z-axis at the beginning of the faces:** : it enables the inversion of the programmed coordinates for Z-axes, if it interprets positive values of Z at the beginning of the workpiece (see the paragraph: **<u>Piece settings->Piece geometry</u>**)
- **Exclude technology capture:** if selected, it disables in the matrix the automatic integration, by the technological component, of programming for tool.
- **Keep constructs** select to maintain displayed the Construct workings that verify anyway the logical conditions. With non-active selection, <u>all</u> construct working are excluded instead from the conversion;
- **Assign the execution parameters:** select to enable the application of the optimization parameters, as set in the previous page;
- **Linearize all arcs**: if selected, it asks the fragmentation and the linearization of all the arcs while creating the piece-matrix
- **Linearise the 3D arcs:** it asks the fragmentation and the linearisation of the arcs assigned in a generic xyz plane during the creation of the piece matrix. This item is always active and cannot be changed. Note: an arc assigned with operating code of xyz-arc (ex: A10), if it is calculated on one of the three coordinated plans of the face (xy,xz, yz), it is given in the piece matrix:
  - as an arc (ex: xz), if the fragmentation of the xz, yz - arcs is not required (xz, yz)
  - as a polyline, if the fragmentation of the arcs in the corresponding plane
- **Convert arcs into XZ, YZ-planes:** if selected, it asks all the xz and yz-arcs to be linearised and to be fragmented in line segments.
- **Linearize helical workings:** if selected, it asks the linearization and the fragmentation of all the arcs with a non-zero depth component.
- **Convert the arcs into oriented profile** if selected, it asks the linearization and the fragmentation of the arcs assigned in profiles, that have an oriented setup code.
- **Arc linearization in the curved face:** if selected, it asks the linearization and the fragmentation of the arcs assigned in curved faces or assigned as surfaces

Of the export modules following methods are used:

- **long ExportFromAlb (string szPathFileCst, ref string szPathFileOut, string szArgs, string szPathConfig, ref string szError)**

  the method is invoked to convert the file shown by TpaCad.

  <u>Method return value: 0 if the format is recognised as valid, otherwise error number</u>

| | |
|---|---|
| string szPathFileCst | pathname of the file in original format (ASCII or matrix-piece) |

| | |
|---|---|
| ref string szPathFileOut | pathname of the file to write. The converter can change the assignment (example: it can assign a different extension) |
| string szArgs | string of arguments.<br>Format and meaning of the fields:<br>• '/': field name<br>• space: optional separator<br>• the string is case insensitive<br>"**/key**=.." greater than 0 the conversion enables the advanced level, if planned by the export module<br>"**/lng**=.."current language (three letters abbreviation. Example: "ITA", "DAN")<br>"**/lngdef**=" language by default (three letters abbreviation: "ITA", "ENG", "FRE", "GER", "SPA")<br>"**/**…"assigned arguments in the field Arguments follow |
| string szPathConfig | folder, where functioning assignments can be read/written (e.g. for the converter with DXF-format it is the name of the folder from which INI files can be read and where the parameters to customise the conversion are assigned) |
| ref string szError | error message |

- **long Settings (string newFolder, string newMachine, string szPathConfig, string szArgs)**

  the method is invoked to manage the customisations used by the conversion module.
  Method return value: 0 if the result is positive, otherwise error number

| | |
|---|---|
| string newFolder | Tpa.ini - file search folder (if the string is empty, the default folder is taken) |
| string newMachine | name of the machine (section Tpa.ini file If the string is empty, the name of the default machine is considered. |
| string szPathConfig | folder from which the functioning assignments can be read and written (example for DXF converter: folder from which the INI files of assignments of the conversion customisations can be read) |
| string szArgs | string of arguments.<br>Format and meaning of the fields:<br>• '/': field name<br>• space: optional separator<br>• the string is case insensitive<br>"**/key**=.." >0 enables the advanced layer, if used by the import module (example for DXF converter: assignment of workings from layers and/or blocks)<br>"**/mode**=.." =1 enables the customizations of the module. The field is managed for a module managing the conversion both in reading and writing:<br>"**/lng**=.."current language (three letters abbreviation. Example: "ITA", "DAN")<br>"**/lngdef**=" language by default (three letters abbreviation: "ITA", "ENG", "FRE", "GER", "SPA")<br>"**/**…"assigned arguments in the field Arguments follow |

- **long SettingsArgs (bool bMode, ref string szArgs)**

  the method is invoked to manage the customizations run by the conversion module. We recommend to implement this method only if a customization through ***Settings***method is not managed. This method can be implemented, if a customization of the export module that must be directly specified on each request of conversion (through the string of the arguments), is managed.
  Method return value: 0 if the result is positive, otherwise error number.

| | |
|---|---|
| bool bMode | *false*, if it is invoked as an importer<br>true, if it is invoked as an importer<br>(The allows you to assign a module managing the conversion both to reading and writing: |
| string szArgs | The string returns the default settings managed by the module (about the format, reference is made to previous information). If, for instance, the module interprets two numeric customizations transmitted as "/aaa=… bbb=…", the method must return the string containing the values that are interpreted in case of non-transmitted arguments (example: "/aaa=1 bbb=0"). |

- **void Dispose()**

The method is invoked before releasing the class instance, to release the managed and the unmanaged resources, such as handle, connections to database,...).

# 2 Custom Functions

## 2.1 Overview of a custom function

A custom function is a program that resolves calculations and logical evaluations, but it does not apply workings. A custom function can be recalled while writing a program, like any other function available in parametric programming. However, the development concerning a logical function is determined according to specific custom. Below, an example for a better understanding of the use of a custom.

In a face program we wish to determine the position of a point P with (r0;r1) coordinates and mirrored around the P1-P2 axis, assigned by two points: P1 (r2;h/2) , P2 (l/2;r3).
By means of a simple research in the available function for the parametric programming, you will find that the geometric library functions geo[pxmir;…] and geo[pymir;…] solve directly the problem, returning the x and y-coordinate of the mirrored point respectively. At the moment, we want to solve the problem in another way. A possible way is to obtain the formulas for the transform required and to assign the first <j> variable (or directly a coordinate parameter) for the x-coordinate and the second <j> variable for the y-coordinate.
If the transform concerns a single case, this solution can be surely suitable. Suppose, instead, we need to calculate the transform several times in different program; in this case the solution we have chosen may not be convenient.
We have deliberately chosen a simple example and we also know that a geometric function solves the problem. Not always is the matter so simple, but above all a solution may not have been provided.
Thinking to generalize the question, it would be useful to write once and for all the formulas that are used for this case (in the example the formula to mirror a point around a generic axis) and recall them by means of a function provided for that purpose: a **custom function.**
A custom function can be used in parametric programming as each other function of parametric programming (for example, the functions of geometric librarygeo[pxmir;…] and geo[pymir;…]).

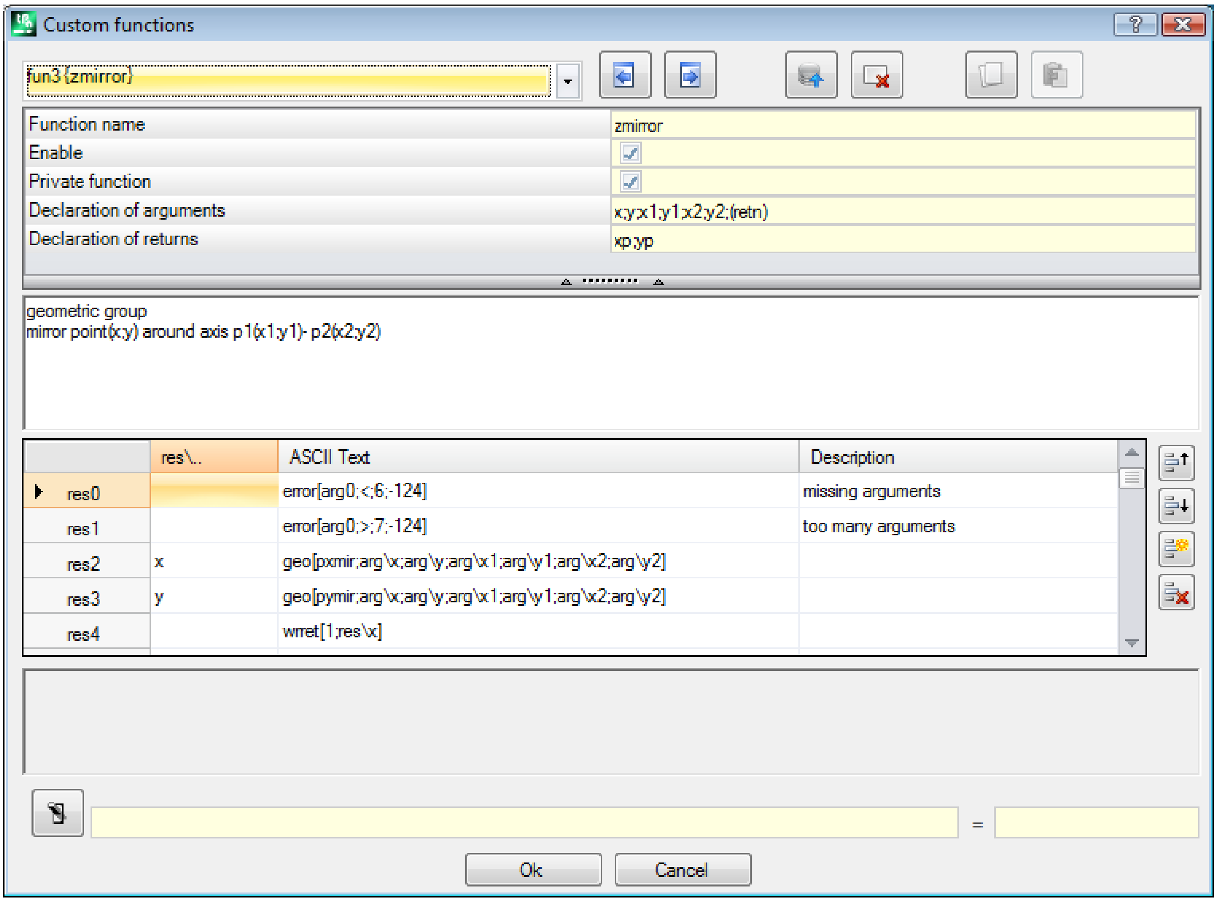
Let us call the custom function in our example: funmirror (we will see later how free we are in assigning the name). Let us define the argument we need for the function:

- x-coordinate of P (let us call the argument: x)
- y-coordinate of P (let us call the argument: y)
- x-coordinate of P1 (let us call the argument: x1)
- y-coordinate of P1 (let us call the argument: y1)
- x-coordinate of P2 (let us call the argument: x2)
- y-coordinate of P2 (let us call the argument: y2).

Let us recall the help of quick info for the functiongeo[pxmir;..] and let us define an analogue prototype for our function: funmirror[x;y;x1;y1;x2;y2].

Now, after the complete definition of what we want to do, let us write the functionfunmirror[..].

## 2.2 Writing a custom function



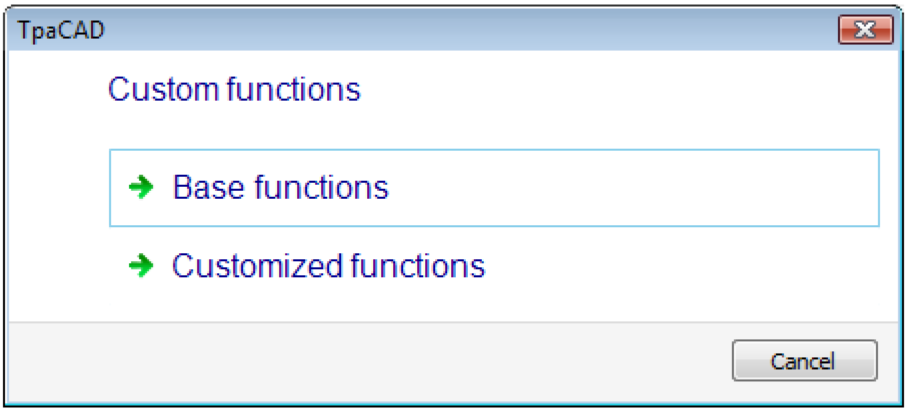
| | |
|---|---|
| Function name | zmirror |
| Enable | ☑ |
| Private function | ☑ |
| Declaration of arguments | x;y;x1;y1;x2;y2;(retn) |
| Declaration of returns | xp;yp |

geometric group
mirror point(x;y) around axis p1(x1;y1)- p2(x2;y2)

| | res\.. | ASCII Text | Description |
|---|---|---|---|
| res0 | | error[arg0;<;6;-124] | missing arguments |
| res1 | | error[arg0;>;7;-124] | too many arguments |
| res2 | x | geo[pxmir;arg\x;arg\y;arg\x1;arg\y1;arg\x2;arg\y2] | |
| res3 | y | geo[pymir;arg\x;arg\y;arg\x1;arg\y1;arg\x2;arg\y2] | |
| res4 | | wrret[1;res\x] | |

The command **Custom functions** is selected from the menu ⚙ and enabled in Professional mode, at **Manufacturer** level and when the program is closed. The use of the custom functions custom must be enabled in TpaCAD Configuration. The command is not available from the menu in Demo functionalities or if the instance of TpaCAD has not been executed first.

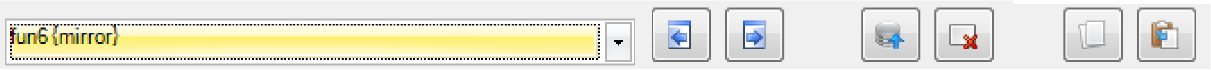
Confirm the choice in selection window:



To open the base database of custom functions, confirm the window in the option *Base function*.
To open the database of the custom functions for the application, confirm the window on the option *Customized functions*.
The base database is stored in the file FUNCTUST.def in the folder tpacadcfg. The base database is stored in the file CADFUNCUST.def in the folder tpacadcfg\custom.
The selection window is shown even if the reading of the custom functions is not active.
**Normally, it is not possible to modify the file of Base functions.**

**Even the modification of the Customized functions can be blocked,** if the machine constructor signed the workings database custom.

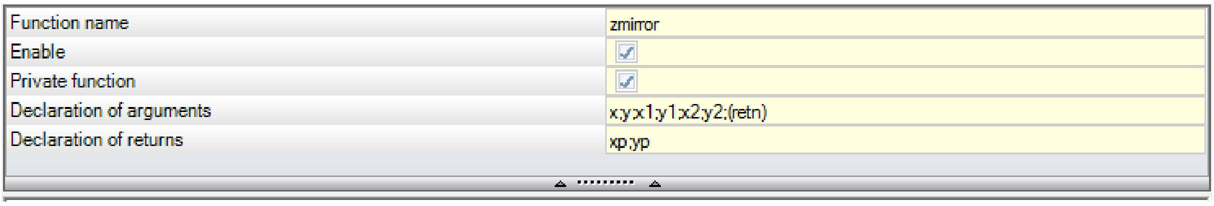
Upon opening the window, a message signals if the reading of the file has led to some adjustments (for invalid format of the file). Adjustments to the file will be only made, if the window is closed with **[OK]**.



fun6 {mirror}

The first case allows the selection of the function. The list assigns 100 functions, identified from “fun0” a “fun99”.

The two buttons ⇦ and ⇨ permit to move directly up and down in the list of the functions.
The different areas of the window can be resized in such a way that the part of more interest can be increased.



| | |
|---|---|
| Function name | zmirror |
| Enable | ☑ |
| Private function | ☑ |
| Declaration of arguments | x;y;x1;y1;x2;y2;(retn) |
| Declaration of returns | xp;yp |

- **Function name:** custom name of the function. The field must be assigned valid for each function, which is reported enabled. Syntax: no more than 16 characters (letters from 'a' to 'z', digits); first character is only a letter and the name is not repeated on more than a one function. The complete name recalling a function is assigned as "fun" + "custom name".

> In the example: funmirror is the name that can be used to recall the function here shown as fun6. The functionality of the numbering for the functions (“fun0” to “fun99”) is auxiliary only: a function is identified only by means of the name.
> As already mentioned, if both the files of functions are read in TpaCAD, the functions read are unified and univocally recognized by the name and for no more of 100 functions.
> More specifically, if in both the files a function is assigned with the same name, this is kept valid the functions from the customized file. Writing the customized functions, we recommend to adopt a rule in the choice of the names in such a way that they are directly associated to the specific application and do not "unwillingly" delete basic function; examples of criteria are suggested as follows: beginning a name with the letter "x" (the complete numbers shall be: “funx….”), or using a customized abbreviation (“eu”, “wood”).
> All names of the functions for the basic file begin with the letter 'z' (the complete names shall be: “funz….”).

- **Enable:** select the field to enable the use of the function
- **Private function:** if enabled, the function is declared private, that is it can only be used in the test of another custom function or in the configuration of a global function (see later). If not enabled, the function is declared public and can be used also in a program text. Continuing our example, we declare the function public.
- **Declaration of arguments:** this is an optional field (max. length: 300 lower-case characters). The field assigns the symbolic names of the arguments according to the following formalism:
  - no more than 30 names separated by "," (semicolon);
  - a single name made of no more of 16 alphanumeric characters;
  - non repeated names;
  - round brackets are admitted to show an optional argument (in the picture: “(retn)”);
  - the use of the point is admitted to characterize a function with a variable number of arguments (see following example);
  - if the function controls an array of return elements (see next point), it is possible to call by name *retn* the last argument (in round brackets, if optional), to show which argument is required as a return of the function (see following example).

Proceeding on our example: we wrote the list of the first 6 arguments (“x;y;x1;y1;x2;y2”) by adding the return argument as an optional argument. Let us say from now that:

- with the seventh non-assigned argument the function returns value 0, if the number of the arguments is not correct, otherwise it returns 1;
- with the seventh argument = 1, the function returns the x-coordinate mirrored;
- with the seventh argument = 2 (not equal to 1), the function returns the y-coordinate mirrored;

The field **Declaration of arguments** is optional, so it can be left empty. We suggest that the arguments are set for the following reasons:

- the name here set for each argument can be used as a symbolic name in the text of the custom function. In our example the name of the third argument is “x1”;

- the string is used in TpaCAD for the contextual use of the syntax of the function (quick info help).

  Examples in assigning function arguments:

  “q1x;q1y;q1z;q2x;q2y;q2z” assigns the names for the first 6 arguments;

  “q1x;q1y;q1z;q2x;q2y;q2z;(retn)” assigns the names for 7 arguments (name of the seventh argument “retn”, optional, to be used to indicate the return of the function);

  “fat1;fat2;fat3;.....” assigns the names for the 3 first arguments and does not define the other ones (typical form for a function with variable number of arguments).
- **Declaration of returns:** this is an optional field (max. length 300 lower case characters). The field assigns the symbolic names for the returns of the function according to the following formalism:
  - no more than 30 names separated by "," (semicolon);
  - a single name made of no more of 16 alphanumeric characters;
  - non repeated names.

Let us proceed on our example: "xm;ym” assigns name for the 2 return element.
Let us determine the position of a point P of original coordinates (r0;r1), mirrored around the P1-P2 axis, assigned by two points: P1(r2;h/2), P2(l/2;r3). To each call of the **funmirror[..]** one only return value can correspond and it is the result of the function. Changing the value of the seventh argument, the meaning of the function result is changed:

- let us assign the x - coordinate mirrored in the r5 variable of the program ”funmiror[r0;r1;r2;h/2;l/2;r3;1]”. The value of the seventh argument is 1 and we want it returns the x-coordinate. Given the assignment of the return elements, we obtain the same result by programming: ”funmiror[r0;r1;r2;h/2;l/2;r3;ret\xm]”;
- let us assign the y-coordinate mirrored in the r6 variable of the program: ”funmiror[r0;r1;r2;h/2;l/2;r3;2]”. The value of the seventh argument is 2 and we want it returns the y-coordinate. Given the assignment of the return elements, we obtain the same result by programming: ”funmiror[r0;r1;r2;h/2;l/2;r3;ret\ym]”;
- let us assign in the r7 variable of the program the value returned by the programming: ”funmiror[r0;r1;r2;h/2;l/2;r3]”. The 7th argument is not assigned and we want it returns 0/1 as a generic result of the function 1 if the result is OK, 0 if the results are NOK. A custom function can generally verify geometric situations or invalid logics.

In summary, a function can generate two kind of information:

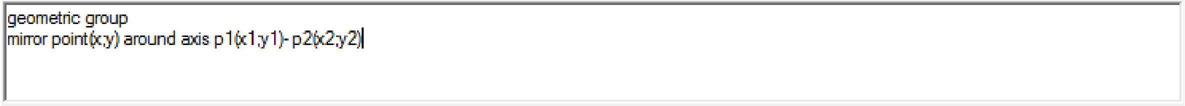
- the first one informs on the result of the function call (we can say: result 1 or 0);
- a second one is the set of the results calculated from the function: from one only result up to 30.

How many and which information can be available with a function call depends primarily on 2 factors:

- the way in which the function is written;
- the way in which the function is called.

As for the writing of a function, the list of the arguments and of the available parametric functionalities (see following paragraphs) for the writing of the custom functions reveals a picture of everything you can do.

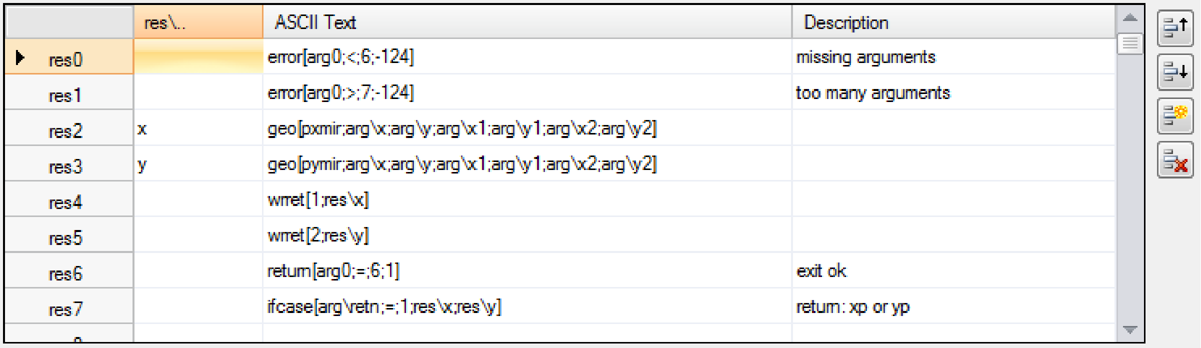
### Assignment comment area



geometric group
mirror point(x;y) around axis p1(x1;y1)-p2(x2;y2)

Comment lines for the function (up to 50 lines are stored), available to record the functionality of the function and of the arguments. The comment lines are not stored in the language file and so they cannot be translated.

### Assigning table of the function test:



|  | res\.. | ASCII Text | Description |
|---|---|---|---|
| res0 |  | error[arg0;<;6;-124] | missing arguments |
| res1 |  | error[arg0;>;7;-124] | too many arguments |
| res2 | x | geo[pxmir;arg\x;arg\y;arg\x1;arg\y1;arg\x2;arg\y2] |  |
| res3 | y | geo[pymir;arg\x;arg\y;arg\x1;arg\y1;arg\x2;arg\y2] |  |
| res4 |  | wrret[1;res\x] |  |
| res5 |  | wrret[2;res\y] |  |
| res6 |  | return[arg0;=;6;1] | exit ok |
| res7 |  | ifcase[arg\retn;=;1;res\x;res\y] | return: xp or yp |

The table is initialized on 300 lines with headings from res0 to res299. Each line can assign a string of parametric programming (column: ASCII text), with the same modes of a program variable (example: r - variable) or of a working parameter. In general, the value for the programming of the line is numeric

(double) and it is assigned to the line itself. For example, if we set "1000/2" to the line res3, the value of res3 shall be 500 and could be used in the programming of the next lines.
For each row the user can enter a description which  is not stored in a language file and so cannot be translated.

In the easiest writing of a custom function, the table is interpreted by solving each programmed line in succession from res0 to res99 and ends with the resolution of the last line. The value given to the last line assigns the function return.
Empty lines are ignored.
Let us see the column of the table:

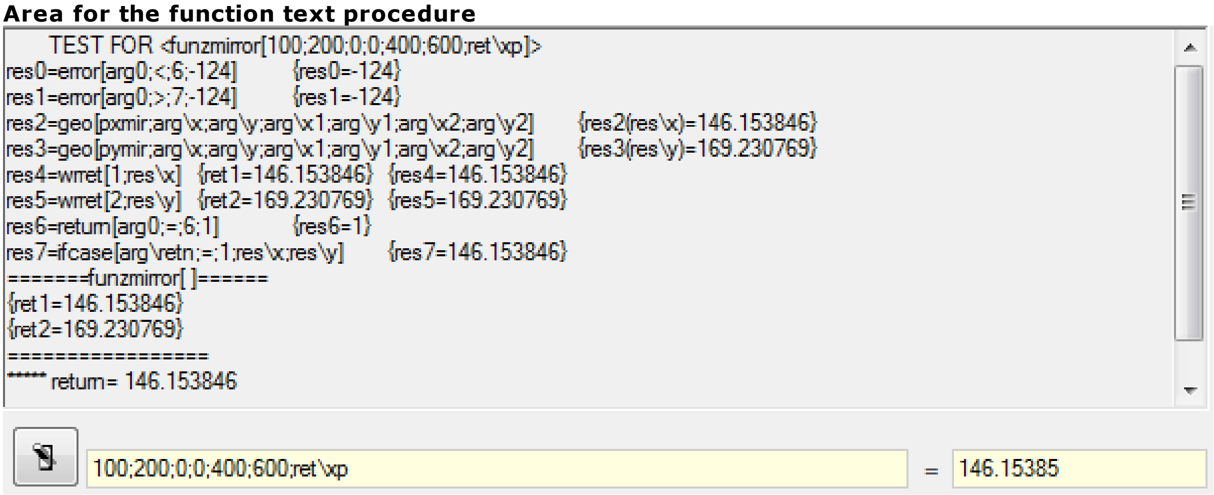
- **res\..:** symbolic name (label) of the line. This is an optional field (up to max. 16 alphanumeric characters, non-repeated names) assigning a "name" (alias) to the program line.
- **ASCII Text:** text of the line (max 100 characters);
- **Description:** comment to the line (max. 50 characters);

The number of the lines remains fixed at 100; however, it is possible

- to move the current line above. The current line and the previous one are automatically renamed.
- to move the current line below. The current line and the next one are automatically renamed.
- to insert a line above the current one. The last line of the table is automatically deleted.
- to delete the current line. A line is automatically added at the end of the table

**WARNING:** Moving, deleting, inserting the lines can make the parametric assignments of the number of lines no more valid (direct use of res0 - res99) and require subsequent changes to the text of the function, You can overcome this problem by using the symbolic names of the lines (in the example we do not useres2 but res\x).

**Area for the function text procedure**



```
     TEST FOR <funzmirror[100;200;0;0;400;600;ret\xp]>
res0=error[arg0;<;6;-124]        {res0=-124}
res1=error[arg0;>;7;-124]        {res1=-124}
res2=geo[pxmir;arg\x;arg\y;arg\x1;arg\y1;arg\x2;arg\y2]      {res2(res\x)=146.153846}
res3=geo[pymir;arg\x;arg\y;arg\x1;arg\y1;arg\x2;arg\y2]      {res3(res\y)=169.230769}
res4=wrret[1;res\x]  {ret1=146.153846}  {res4=146.153846}
res5=wrret[2;res\y]  {ret2=169.230769}  {res5=169.230769}
res6=return[arg0;=;6;1]          {res6=1}
res7=ifcase[arg\retn;=;1;res\x;res\y]      {res7=146.153846}
=======funzmirror[ ]======
{ret1=146.153846}
{ret2=169.230769}
=================
***** return= 146.153846
```

100;200;0;0;400;600;ret\xp = 146.15385

The upper area presents the indications for the function development, according the the assigned values of the argument list (in the example: “100;200;0;0;40;600;ret\xm”). Numeric arguments are set, with the exception of the return value, separated by ",". To check the custom function call with the argument given, select the test button .

Once the test has been made:

- if the function has not detected any error situation, in the area on the right the value returned by the function appears (in the example: 146.15385);
- if the function has detected error situations, the message of the error detected in the parametric programming appears.

In the area of the test procedures the flow of the instructions executed during the function development is shown.

The button checks the function settings. Messages report invalid situations and possible changes to the function (name, arguments, returns). For the syntax of the programmed text see more specifically the following section.

The button ✕ deletes the function.

The button creates the copy of the current function and allows the application of all assignment to another function in the same file or in another one.

The button assigns the current function to the last copy of the performed function. The last copy can derive from a different opening of the window; for example it is possible:

- to open the file of the base functions and copy a function (for example: "fun20");
- to close the window and select again the command while opening the file of customized functions;
- to go to the function you want to assign (for example: "fun5") and select the button.

The function "fun5" shall be now the copy of the original "fun20". In the working is copied inside the same file, it will be necessary to change the name of the function "fun5" in such a way to make it univocal.
Exit after recording the function file is possible only if all the error situations are solved.

## 2.3 Syntax of a custom function

In the text of a custom function it is possible to use the parametric forms, with the exception of

- piece variables (o, v, r, $, j) and related function (strlen,..)
- custom section settings (szs\.., szi\.., szo\.., szl\..)
- cycle variable arguments (subx,...)
- function of parametric programming using the working name.

Furthermore, arguments and functions which are set aside to write custom functions are available and for this reason they are not recorded in the usage manual of TpaCAD, but described in the next paragraphs.

### Arguments to develop custom functions

| | |
|---|---|
| res0 –<br>res299<br>res\name | returns the result of the corresponding function development line:<br>• the first form corresponds to the formalism by default (resn, with n= line number, valid from 0 a 299)<br>• the second form corresponds to the symbolic formalism, with:<br>• fix part "res\";<br>• variable part: the symbolic name assigned for the development line.<br>Error situations :<br>• 136: use in a context of no custom function<br>• 138: invalid index or variable name *res* (<0 o >299). |
| arg0 | returns the number of arguments passed to the custom function.<br>Error situations :<br>• 136: use in a context of no custom function.<br>Example:<br>funmirror[100;200;0;0;40;600;ret\xm] is arg0=7<br>funmirror[100;200;0;0;40;600] is arg0=6 |
| arg1-<br>arg30<br>arg\name | returns the 1°,.., 30° argument passed to the custom function (0.0 if it does not exist):<br>• the first form corresponds to the formalism by default (argn, with n= number of argument, valid from 1 to 30)<br>• the second form corresponds to the symbolic formalism, with:<br>• fix part "arg\"<br>• variable part: the symbolic name assigned for the argument.<br>Error situations :<br>• 136: use in a context of no custom function.<br>• 137: index or invalid name of variable var (index: <0 o >30). |
| var0-<br>var99 | Auxiliary variables in the development of a custom function (initial values: 0.0).<br>The variables can be modified by means of the function wrvar[].<br>Error situations :<br>• 136: use in a context of no custom function.<br>• 141: invalid index of variable var (<0 o >99). |

| | |
|---|---|
| ret0 | returns the greater index of the compiled values (with the function wrret[]) in the return array of the custom function.<br>Error situations :<br>• 136: use in a context of no custom function. |
| ret1-ret30<br>ret\name | returns the 1°,.., 30° value compiled in the array returning from the custom function:<br>• the first form corresponds to the formalism by default (retn, with n= number of element, valid from 1 a 30)<br>• the second form corresponds to the symbolic formalism, with:<br>• fix part "ret\";<br>• variable part: the symbolic name assigned for the argument.<br>Usable:<br>• in the development of a custom function<br>• using a custom function (as last call argument).<br>Error situations :<br>• 136: use in a context of no custom function.<br>• 143: invalid index or name of value ret (<0 o >30).<br>Example:<br>fun0[12;r7;lf/2;ret1]<br>fun0[12;r7;lf/2;ret\vet] |

## Functions to develop custom functions

| | |
|---|---|
| error [nc1;nesp;nc2]<br>error [nc1;nesp;nc2; nerr] | The function evaluates the condition (nc1 ? nc2), with ?=nesp:<br>if the condition is TRUE, it ends the development of the custom function and returns error number 139 (custom function call error).<br>The arguments of the function must be 3 or 4, numbers or parameters<br>The argument nesp is interpreted to assign the condition between nc1 andnc2 :<br>• value 0 corresponds to < (less)<br>• value 1 corresponds to < (less or equal)<br>• value 2 corresponds to < (greater)<br>• value 3 corresponds to < (greater or equal)<br>• value 4 corresponds to < (equal)<br>• value 5 corresponds to < (not equal to)<br><br>For the argument nesp it is also possible to assign the symbolic corresponding forms instead of the numeric value. Example: the condition of "greater or equal to" can be set with value 3 or as ">=".<br>The inequality relation can be expressed as "<>" or as "#".<br>The comparison condition set between nc1 and nc2 is evaluated at less than one epsilon = 0.001.<br><br>In the 4 arguments format nerr assigns a custom error number. Following values are considered valid:<br>• between 1 and 500: the messages of these errors are managed in the file of messages CADAUX.XMLNG, in the interval (1001-1500)<br>• if nerr is negative and if an error corresponding to nerr is valid, it is managed as an error (Example: you can set nerr=-124 to produce this error message "Parametric programming: function with a wrong number of operands").<br><br>Error situations:<br>• 123: Number of operands =0;<br>• 124: Number of operands different from 3<br>• 140: use in a context of no custom functions<br>• 139: custom function programmed error, if nerr is not assigned or if it is assigned invalid<br>• 130 - 138: custom function programmed error, if nerr is assigned valid. The returned error has value = (130 + nerr –1). |
| return [nc1;nesp;nc2; nval]<br>return [nval] | In the first format (with 4 arguments) the function evaluates the condition of the first three arguments and if the condition is TRUE, it ends the development of the custom function and returns the value assigned in nval (numeric or parametric).<br><br>In the second format (to an argument) the function returns directly nval, without valuing conditions .<br>The arguments are numbers or parameters.<br>The function must appear **alone** in the custom function block.<br>The comparison condition set between nc1 and nc2 is evaluated at less than one epsilon = 0.001. |

| | |
|---|---|
| | Error situations:<br>• 123: Number of operands =0;<br>• 124: Number of operands not equal to 1 and not equal to 4<br>• 140: Use in a context of no custom functions.<br><br>Example<br>At the first line of the function funmirror (see paragraph **Writing a custom function**) a return condition is programmed:<br>{res0}= return[arg0;<;6].<br>If the number of the argument of the function is less than 6, the function returns value 0. |
| goto [nc1;nesp;nc2; nval]<br>goto [nval] | In the first format (with 4 arguments) the function evaluates the condition on the first three arguments and, if the condition is TRUE, jumps to examine the development of the custom function from the res# line, with #=nval (numbers or parameters).<br><br>In the second format (with one argument) the function jumps directly to the indicated line, without valuing conditions.<br>The arguments are numbers or parameters.<br>The function must appear **alone** in the custom function block.<br>The comparison condition set between nc1 and nc2 is valued at less than one epsilon = 0.001.<br><br>Remarkable format<br>If the last argument of the function is parametric and has a remarkable format “resn” (or: res\name), the function jumps to the line resn (or res\name).<br><br>Error situations:<br>• 123: Number of operands =0;<br>• 124: Number of operands different from 1 and different from 4<br>• 140: Use in a context of no custom functions.<br><br>Example:<br>goto[2] jumps to the res2 line<br>goto[res5] jumps to the res5 line<br>goto[res\loop] jumps to the line called "loop". |
| parg[nn] | Reference to an argument passed to the custom function.<br>The parametric form (use of brackets [.]) is mandatory. Returns the value of the argument argnn, with nn=value calculated within square brackets.<br><br>Error situations :<br>• 123: Number of operands =0;<br>• 124: Number of operands #1;<br>• 137: Index or name of not valid argument (<0 or >30);<br>• 140: Use in a context of no custom functions.<br><br>Example:<br>parg[5] returns the value of the fifth argument (it is the same as to write: arg5)<br>parg[arg1] with arg1=4 returns the value of the forth argument (it is the same as to write: arg4) |
| pvar[nn] | Reference to a variable of the custom function.<br>The parametric form (use of brackets [.]) is mandatory. Returns the value of the variable var nn, with nn=value calculated within the square brackets.<br><br>Error situations:<br>• 123: Number of operands =0;<br>• 124: Number of operands different from 1;<br>• 141: Index or name of not valid argument (<0 or >99);<br>• 140: Use in a context of no custom functions. |
| pret[nn] | Reference to an element compiled by the custom function.<br>The parametric form (use of brackets [.]) is mandatory. Returns the value of the argument retnn,with nn=value calculated within square brackets.<br><br>Error situations:<br>• 123: Number of operands =0;<br>• 124: Number of operands different from 1;<br>• 143: Index of not valid argument (<0 or >30);<br>• 140: Use in a context of no custom functions. |

| | |
|---|---|
| argempty[nn] | verifies that the argument whose index is indicted by nn is assigned. It returns:<br>• 0 if the argument is assigned;<br>• otherwise 1.<br>The parametric form is compulsory (use of [] brackets)<br>If the argument of the function is parametric and has a remarkable format "arg*nn*" or "arg\name" the function operates directly upon the argument arg *nn*.<br><br>Error situations :<br>• 123: Number of operands =0;<br>• 124: Number of operands #1;<br>• 137: Index or name of not valid argument (<0 or >30);<br>• 140: Usages in a context of no custom functions. |
| wrvar [nvar; nval]<br>wrvar [nvar; nval;nval1;..] | In the first format (with 2 arguments) it assigns the variable var of index nvar to the value nval and returns the value of the variable after the assignment.<br><br>In case of more then 2 arguments:<br>• assigns the variable var of index nvar to the value nval<br>• assigns the variable var of index nvar to the value nval1<br>• ..<br>• up to max 29 available assignments.<br>The function returns the value that corresponds to the first assignment.<br><br>The arguments are numbers or parameters.<br><br>Remarkable format<br>If the first argument of the function is parametric and has a remarkable "varn" format, the function operates directly upon the variable var n<br><br>Error situations :<br>• 123: Number of operands =0;<br>• 124: Number of operands =1;<br>• 141: Index of var variable not valid (<0 o >99);<br>• 140: Use in a context of no custom functions.<br><br>Example:<br>wrvar[0;arg1] writes the variablevar0=arg1 and returns the value of arg1 |
| wrret [nret; nval] | assigns an element in the array of the return variables. It assigns the element of index nret to the value nval and returns the assigned value.<br>In the test of a custom function it is possible to assign an array of 30 elements (indicated in a symbolic way as ret1-ret30), that remain available at the function return. This array of elements can be only used in the test of custom function. Then it can be used in case of nested calls of custom functions.<br>The arguments are numbers or parameters.<br><br>Remarkable format<br>If the first argument of the function is parametric and has a remarkable "retn" format, the function operates directly upon the variable ret n<br><br>Error situations:<br>• 123: Number of operands =0;<br>• 124: Number of operands different from 2;<br>• 143: Index of not valid argument (<1 or >30);<br>• 140: Use in a context of no custom functions.<br><br>In the example of the picture:<br>at the line res4 of the function an assignment of the variable retn is programmed:<br>{res4}= wrret[1;res\x]<br>the variableret1 (return variable) takes the value of the variable res\x (line variable) and res4 takes the value of res\x. |
| getret [nret] | reads an element from the array of return variables. It reads the element of index nret and returns the value read.<br>The index nret=0 reads the highest index of writing in the array.<br>The argument is numeric or parametric.<br><br>Remarkable format<br>If the argument of the function is parametric and has a remarkable format "retn" (or: ret\name), the function operates directly upon the element retn. |

Error situations:

- 123: Number of operands =0;
- 124: Number of operands different from 1;
- 143: Index of not valid argument (<0 or >30);
- 140: Use in a context of no custom functions.

As an example, let us examine a possible custom function text using our function funmirror:

```
..
res5=funmirror[arg1;arg2;arg3;arg4;arg5;arg6]
res6=error[res5;=;0]
res7=getret[1]
res8=getret[2]
..
```

- to the line (res5) the function funmirror is called
- to the line (res6) the return of the functionfunmirroris valued: if =0, the development appears failed
- at the line (res7) the element ret1 is read as compiled by the functionfunmirror
- at the line (res8) the element ret2 is read as compiled by the functionfunmirror

The array of the return elements compiled in the development of a custom function can be directly accessed (by means of the function getret) at the level of custom functions only. Editing a variable or a working parameter a custom function returns at any rate a value.
Back to our example: we can write the function in such a way that we can choose which value, among the possible ones, is returned by the function. So:

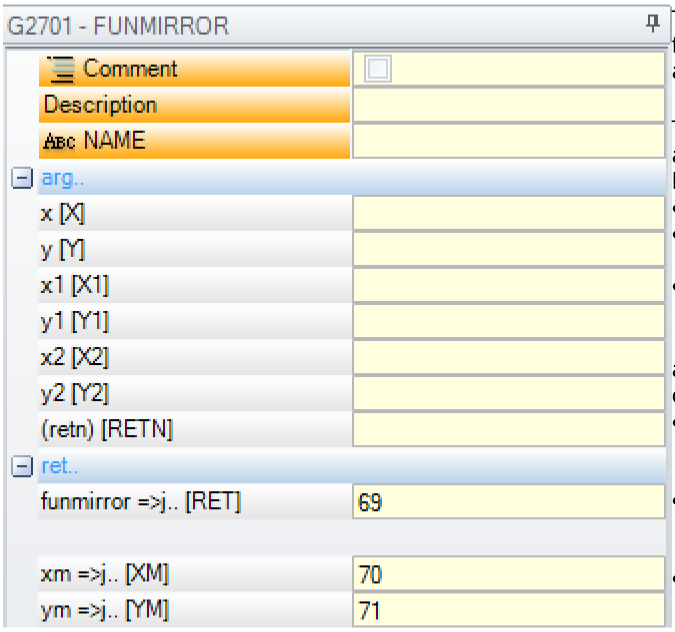
funmirror[0;0;400;0] returns 0: the result of the call is NOT correct
funmirror[0;0;400;0;0;400] returns 1: the result of the call is correct
funmirror[0;0;400;0;0;400;1]returns the x - coordinate mirrored
funmirror[0;0;400;0;0;400;2]returns the y - coordinate mirrored

## 2.4 The global functions

We closed the prospectus of the functions for the programming of custom functions and indicated how you can choose among the different values compiled by a function.
Obviously, all this makes sense, if a function needs to compile different information. For example, if we write a custom function calculating the hypotenuse of a right triangle, given the two catheti, the function needs only one return value (the hypotenuse).

In the case of a function that fills more values, we have an array of elements of return, for a maximum number of 30 elements.
We are able to use a custom function when we program a program variable or a working parameter.
Example: r5=funmirror[0;0;400;0;0;400;1] returns the x-coordinate mirrored.

The **global functions** are particular logical instructions allowing the execution of a more or less complex calculation procedure and the direct assignment of results in <j> variables.
They must be set up in the configuration phase of an application, by evaluating specific customization needs in details.



G2701 - FUNMIRROR

| | |
|---|---|
| Comment | □ |
| Description | |
| NAME | |
| arg.. | |
| x [X] | |
| y [Y] | |
| x1 [X1] | |
| y1 [Y1] | |
| x2 [X2] | |
| y2 [Y2] | |
| (retn) [RETN] | |
| ret.. | |
| funmirror =>j.. [RET] | 69 |
| xm =>j.. [XM] | 70 |
| ym =>j.. [YM] | 71 |

The picture is an example of how a global function to recall the funmirror function can be assigned:

The node arg.. groups together the assignments of the arguments. Each argument has:

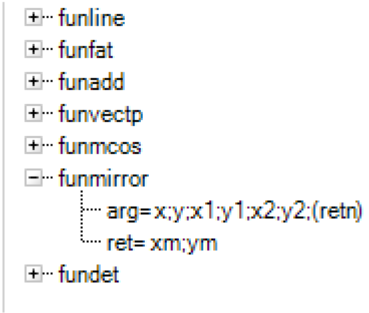
- a separated field of assignments;
- a descriptive message (in the picture: the names of the arguments are used);
- an own assignment mode (direct edit field, selection in the list, check box);

a ret.. node groups together the assignments of the return elements:

- funmirror =>j..: it assigns the index of the <j> variable, that sets the return (result) of the function (here: j69);
- xm =>j..: assigns the index of the <j> variable, that sets the x - coordinate transformed (here:j70);
- ym =>j..: it assigns the index of the <j> setting the y - coordinate transformed (here: j71).

**Automatic setting procedure of a global function**
A global function code must be assigned in the database of the Tpaworks workings.
100 codes are set aside for the global functions: from 2701 to 2800.

Requiring that a code of a global function is assigned, TpaWorks suggests that an automatic procedure is launched, which allows an easy setting of the working.



```
+ funline
+ funfat
+ funadd
+ funvectp
+ funmcos
- funmirror
    arg=x;y;x1;y1;x2;y2;(retn)
    ret= xm;ym
+ fundet
```

a window opens and shows the list of the custom functions as they are read by the function file. The association between global function and custom function occurs with the function name. The node expands on:

- arguments
- returns

In the list only the following functions are shown:

- valid (written correctly)
- declared public or private
- with or without declared returns.

The association between global function and custom function is made.
Select the row of the table you wish and confirm your choice. The working is automatically configured according to the available declarations (name of the function, arguments and returns).

## 2.5 Redifining the custom functions

A custom function provides a new implementation of a system function. This can happen for a specific group of system function only, more specifically for:

- some functions working of strings: toolex, tooltip
- all the technological functions: primp, prmac, prgr, prface, prtface, prfi, prtool, prtip, prfulcrox/y/z, prmxmac, prmxgru, prmxtool, prmxhtool, prmxstore.

The redefinition of a system function allows the customisation of the functioning mode and/or its result, with respect of the functioning automatically declared.
As an example, let us consider the redefinition of the technological function: **prtool**.
The text of the custom function must be written according to the usage required. More specifically, it is not obligatory that the primary technological functionprtool is recalled.
A custom function have to be defined with

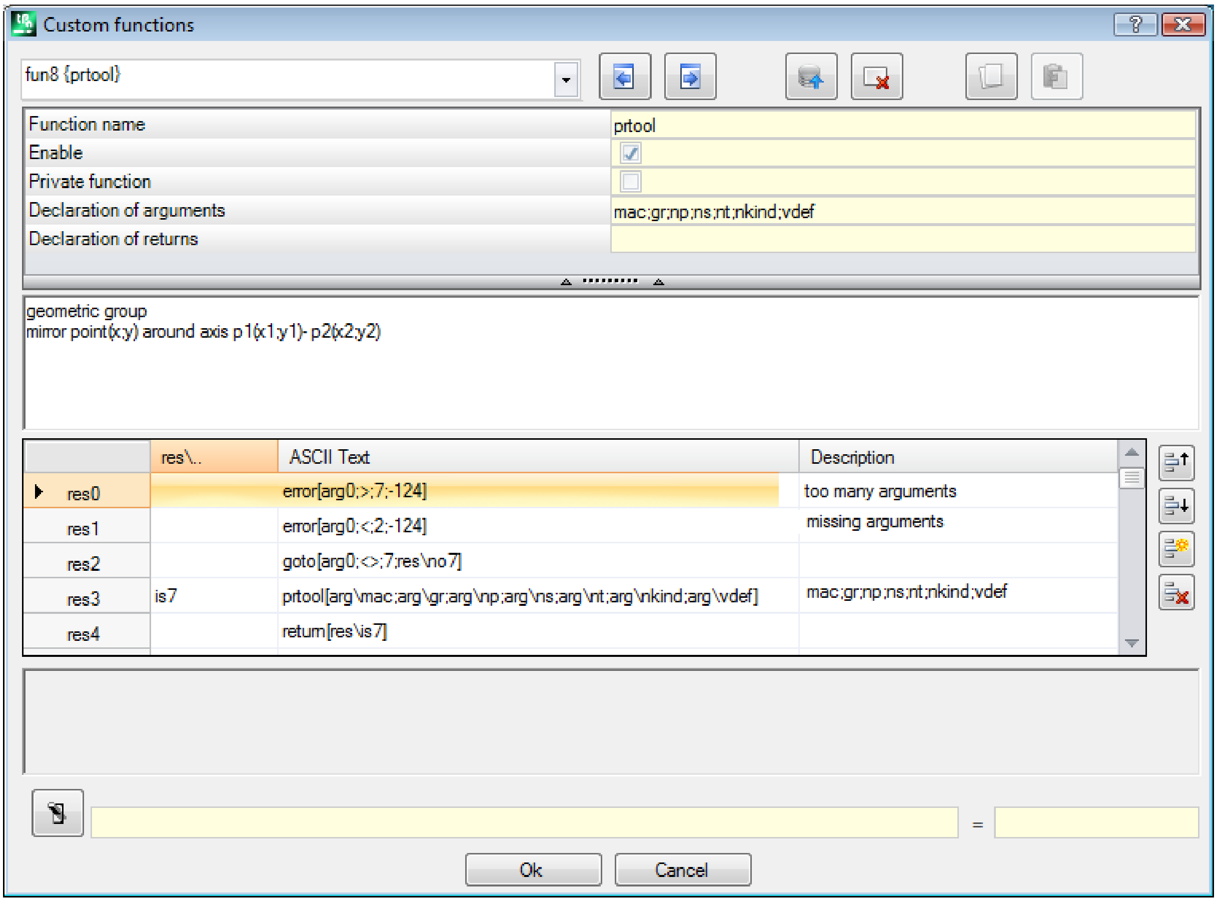
- “prtool” as a custom name;
- argument list compatible wqithe the technological function prtool. Therefore, it must accept all the syntaxes valid for prtool:
  - with 7 arguments: “nmac;ngr;np;ns;nt;nkind;vdef”
  - with 6 arguments: “nmac;ngr;np;ns;nt;nkind;vdef”
  - with 5 arguments: “nmac;ngr; nt;nkind;vdef”

  - with 4 arguments: “nmac;ngr;nt;nkind”
  - with 3 arguments: “ngr; nt;nkind”
  - with 2 arguments: “nt;nkind”
- the function must be declared public.

When programming the piece, and more specifically when programming a variable or a working parameter, if you use the prtool function, the custom function and not the system technological function is called.

Inside a custom function the function system and not the possible redefined function is always recalled. In this case, if the custom function recalls the technology function prtool, the prtoll function system is recalled.

In the picture below an example of rprtool function redefinition:



|  |  |
|---|---|
| Function name | prtool |
| Enable | ☑ |
| Private function | □ |
| Declaration of arguments | mac;gr;np;ns;nt;nkind;vdef |
| Declaration of returns |  |

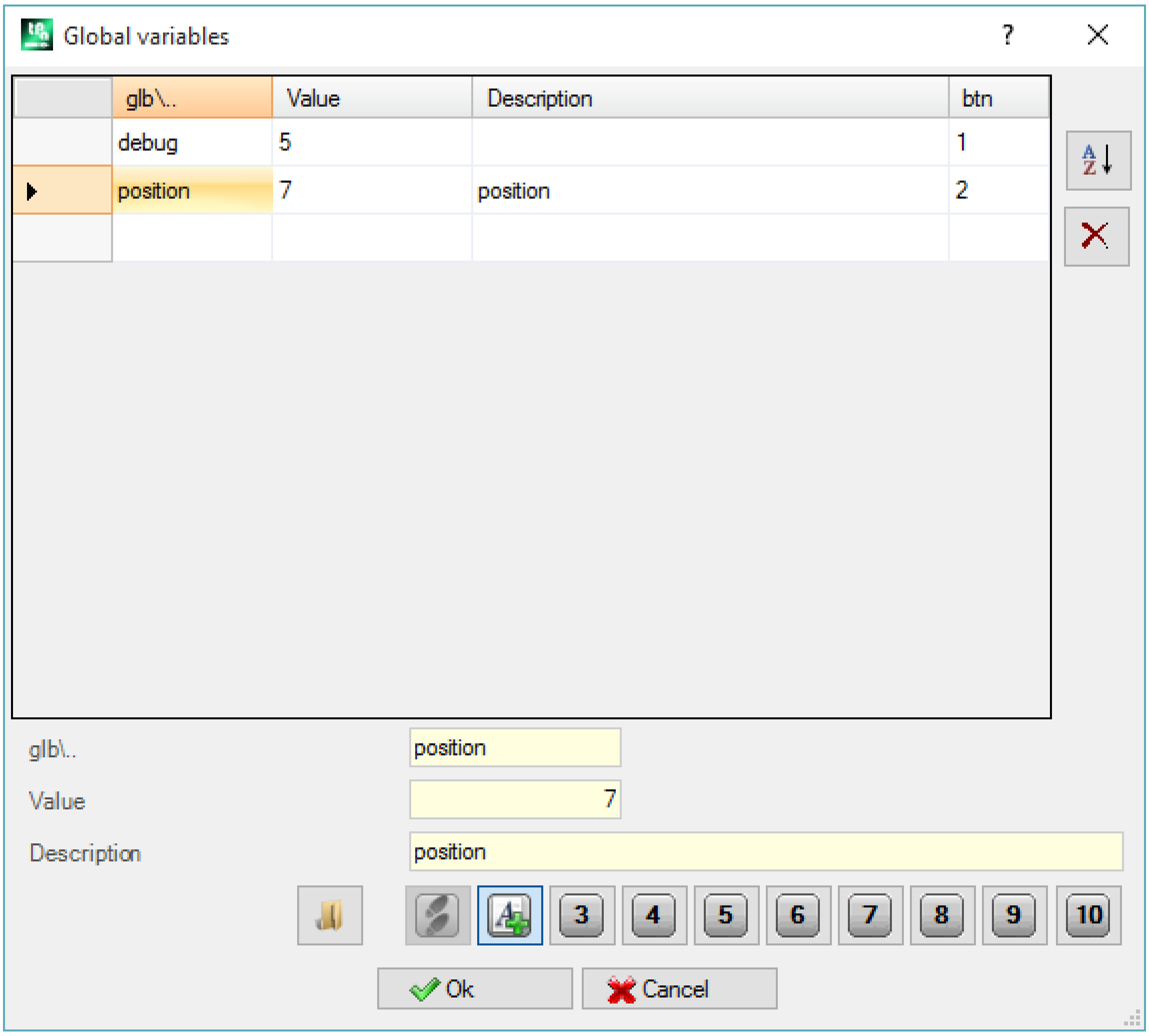
geometric group
mirror point(x;y) around axis p1(x1;y1)- p2(x2;y2)

|  | res\.. | ASCII Text | Description |
|---|---|---|---|
| res0 |  | error[arg0;>;7;-124] | too many arguments |
| res1 |  | error[arg0;<;2;-124] | missing arguments |
| res2 |  | goto[arg0;<>;7;res\no7] |  |
| res3 | is7 | prtool[arg\mac;arg\gr;arg\np;arg\ns;arg\nt;arg\nkind;arg\vdef] | mac;gr;np;ns;nt;nkind;vdef |
| res4 |  | return[res\is7] |  |

The function recalls the technological prtool[..], by differentiating the call according the the number of the arguments.

# 3 Global variables

This command manages the assignments of Global variables.
The level required to enable the entry in the menu is determined in TpaCAD Configuration
They are strictly numerical variables, they are no more than 300 and can only be recalled by name.



| glb\.. | Value | Description | btn |
|---|---|---|---|
| debug | 5 | | 1 |
| position | 7 | position | 2 |

glb\.. position

Value 7

Description position

The window shows the list of the variables already assigned and sorted by name. Moving the selection to an item of the list, the fields show the settings that can be modified:

- **glb\..:** symbolic name of the variable. This is the only obligatory field for the assignment of a variable (max. 16 alphanumeric characters, names not repeated), that assigns the "name" to the variable.
- **Value:** numeric value of the variable (default: 0.0)
- **Description:** description of the variable (max. 50 characters).
- **Command:** it is possible to associate a selection button in the TpaCAD menu for max. 10 global variables. As in the image:
  - the first available button, for which the icon was selected, is associated to the “debug” variable;
  - the second available button, for which the icon was selected, is associated to the “position” variable.

  This functionality is used to manage the "state variables". They are variables whose significant value is 0 or ≠ 0 (0/1):
    - 0 value corresponds to the "top" button;
    - value 1 (generally: ≠ 0 corresponds to the "low" button

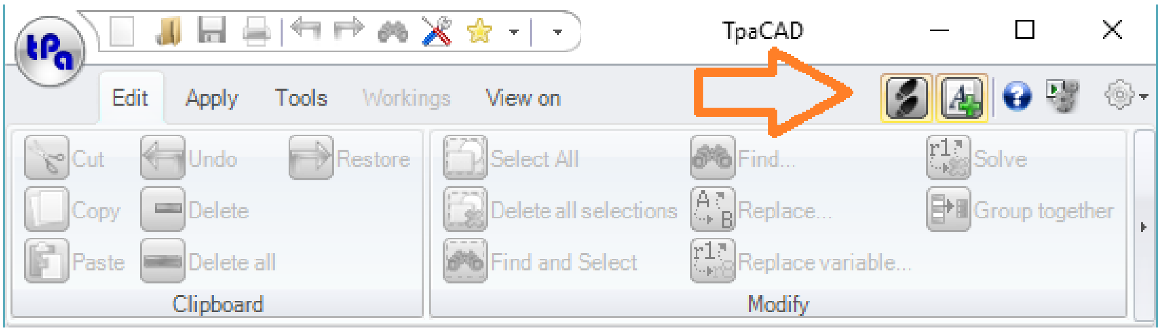
  and is mainly used in the macro-programs and/or customized subroutines.

In the example in the figure:

- it is possible to associate to the value 1 of the "debug" variable the view of elements of particular constructions, that highlight how the development of a working has worked;
- it is possible to associate the the value 1 of the "quote" variable the view of dimensionings that normally are not developed.

To associate a command to a variable you only need to select one of the 10 available buttons originally numbered from 1 to 10.
This choice corresponds to the choice of a position in a command list in the menu.

To associate an icon to a button, select the button . A window opens to select the image. The image files stored in the configuration folder (tpacadcfg\custom\dbglb) appear. The valid formats are *.JPG, *.PNG, *.BMP. The image must be square, max. dimension 22x22 pixels.
The buttons are included in the main menu of TpaCAD to the right of the multifunction menu:



To assign a new variable:

- move the selection to the last item of the list (empty line)
- compile the fields, starting from the symbolic name.

You can add up to 300 variables max.

Sort the list of the variables by name

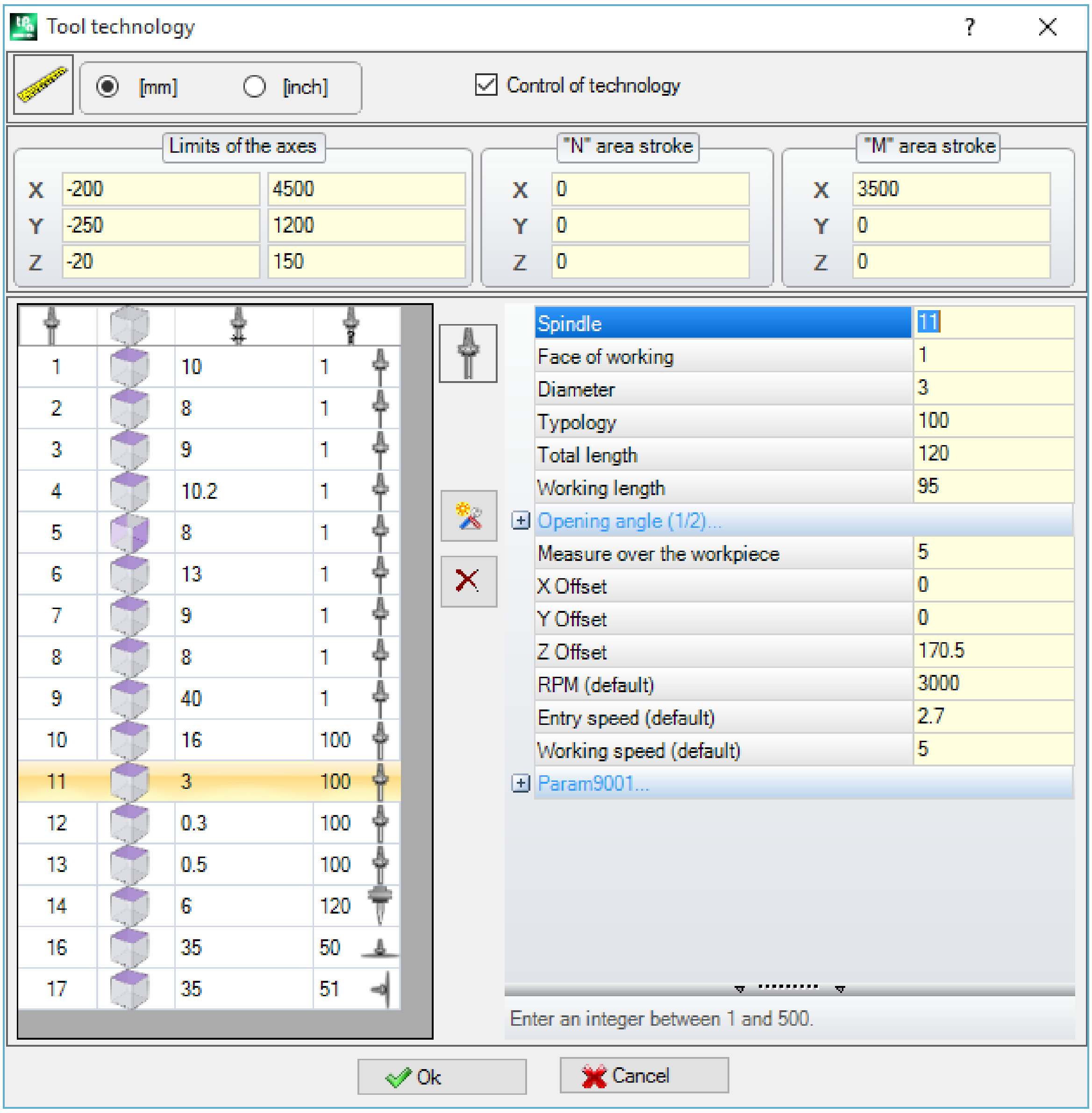
Remove the selected variables

# 4 Tool Technology

This command manages the assignment of the local tools to TpaCAD.
The required level to enable this menu entry is determined in Configuration of TpaCAD.

They are technological parameters of a plant defined by:

- a machine;
- a single group for the selection of the spindles (positions on the group);
- max 500 equipped spindles;
- max 2 working areas (normal and x mirror).



Tool technology

● [mm] ○ [inch] ☑ Control of technology

| Limits of the axes | | |
|---|---|---|
| X | -200 | 4500 |
| Y | -250 | 1200 |
| Z | -20 | 150 |

| "N" area stroke | |
|---|---|
| X | 0 |
| Y | 0 |
| Z | 0 |

| "M" area stroke | |
|---|---|
| X | 3500 |
| Y | 0 |
| Z | 0 |

| | | |
|---|---|---|
| 1 | 10 | 1 |
| 2 | 8 | 1 |
| 3 | 9 | 1 |
| 4 | 10.2 | 1 |
| 5 | 8 | 1 |
| 6 | 13 | 1 |
| 7 | 9 | 1 |
| 8 | 8 | 1 |
| 9 | 40 | 1 |
| 10 | 16 | 100 |
| 11 | 3 | 100 |
| 12 | 0.3 | 100 |
| 13 | 0.5 | 100 |
| 14 | 6 | 120 |
| 16 | 35 | 50 |
| 17 | 35 | 51 |

| | |
|---|---|
| Spindle | 11 |
| Face of working | 1 |
| Diameter | 3 |
| Typology | 100 |
| Total length | 120 |
| Working length | 95 |
| ⊞ Opening angle (1/2)... | |
| Measure over the workpiece | 5 |
| X Offset | 0 |
| Y Offset | 0 |
| Z Offset | 170.5 |
| RPM (default) | 3000 |
| Entry speed (default) | 2.7 |
| Working speed (default) | 5 |
| ⊞ Param9001... | |

Enter an integer between 1 and 500.

- 📏 : ] select the unit of measure of the Technology data.
- **Limits of the axes**: set the work limits of the three coordinated axes of the machine. The unit of measure is: [mm] or [inch]
- **"N" area stroke** : set the coordinates of the strokes of the piece in "N" area (normal). The unit of measure is: [mm] or [inch]
- **"N" area stroke** : set the coordinates of the strokes of the piece in "N" area (normal). The unit of measure is: [mm] or [inch]

**Table of Spindles**: this table can assign up to 500 positions on the machine head. The table is arranged by increasing value of the spindle number. To customise a spindle, move the selection to a raw of the table (on the left) and assign the listed fields on the right side of the window:

- **Spindle**: number of the spindle (the setting appears in the first column of the table). Set a value between 1 and 500.
- **Face of working** : identifier of the face/s of the piece on which the spindle works (this setting appears in the second column of the table). Set one of the following value:
  - ✍ 0 = no face (use of for special tools);
  - ✍ 1, 2, 3, 4, 5, 6 = one face only (the number corresponds to the automatic face number);
  - ✍ 12, 35, 46, 3456 = associated faces;
  - ✍ 123456 = all faces, included those fictive and/or automatic;

The setting is assisted: select the button ... to open a window in which you shall make the selection directly in a list of images.

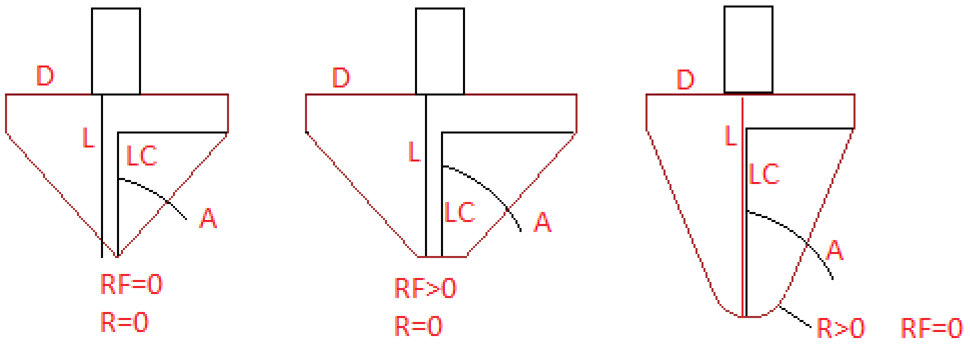
- **Spindle**: diameter of the tool (the setting appears in the third column of the table). The unit of setting is: [mm] or [inch]
- **Tipology**: tool typology. Values and intervals of remarkable values are assigned for which a special icon appears:
  - ✍ (1 - 99) (excluded the following values: 50, 51, 52) = drill bits
  - ✍ 50 = X blade;
  - ✍ 51 = Y blade;
  - ✍ 52 = oriented blade;
  - ✍ 120 = conic milling cutter
  - ✍ (100 – 149) = milling cutters;
  - ✍ (150 – 169) = inserters;

(the setting appears in the fourth column of the table). Set a value between 0 and 1000)

The setting is assisted: select the button ... to open a window in which you shall make the selection directly in a list of images.

- **Total length** : distance between hook point and tool bit (the unit setting is: [mm] or [inch];
- **Working length** : maximum working length of the tool into the piece (the unit setting is: [mm] or [inch];

Following four parameters are characteristic of a conic tool (typology: 120). We would like to remind you, that recognizing a conic tool can influence the procedure of **Sharp edge cut** . Below, the stylized representation of the three possible cases:



(D= Diameter; L= Working length)

- **Opening angle (1/2)**: opening half-angle of the cone (set a positive value less than 90.0°: setting unit: [degrees]) (A" in the figure)
- **Radius of the plat part**: radius on the bit of the cone (setting unit: [mm] or [inch]) - (in the figure: R)
- **Radius of the plat part**: radius on the bit of the cone (setting unit: [mm] or [inch]) - (in the figure: RF)
- **Groove length** : the field cannot be edited, but is is automatically determined (unit setting: [mm] or [inch]) - (in the figure: LC)

- **Measure over the workpiece**: safety distance of the spindle from the piece (the unit setting is: [mm] or [inch];
- **X Offset**: distance in x of the spindle from the zero point of the machine (the unit setting is: [mm] or [inch];
- **Y Offset**: distance in y of the spindle from the zero point of the machine (the unit setting is: [mm] or [inch];

- **Z Offset**: distance in z of the spindle from the zero point of the machine (the unit setting is: [mm] or [inch];
- **RPM (default)**: rotation by default (the unit setting is: [revolutions/min]);
- **Entry speed (default)**: speed by default whilst entering the piece (the unit setting is: [m/min] or [inch/sec]);
- **Working speed (default)**: speed by default whilst working in the piece (the unit setting is: [m/min] or [inch/sec]);
- **Specular tool** : specular tool position. Set 0 (tool not assigned) or the number of an assigned spindle.
- **Param9001 … Param9010**: unassigned significant parameters and unit of setting.

Finally, let us examine the selection beside the selection of the unit of measure:

- **Control of technology** : select the field to use the current window as Technological table, normally opened in TpaCAD from menu command or from the Working tool field or from the command of parametric programming

In these uses, the window is shown with some variations, so as to implement the usual interactions that normally are available in the default technological table:

- the presentation of the values converted to the unit of measure of the current program;
- the sort order of the tools according the values of diameter, typology, work faces;
- possibility to filter the tools according to the application face;
- the direct selection by double-clicking on each row of the table.

# 5 Writing a macro-program

A macro-program is assigned with the same mode of a program, apart from some exceptions; however, it is designed to a different use:

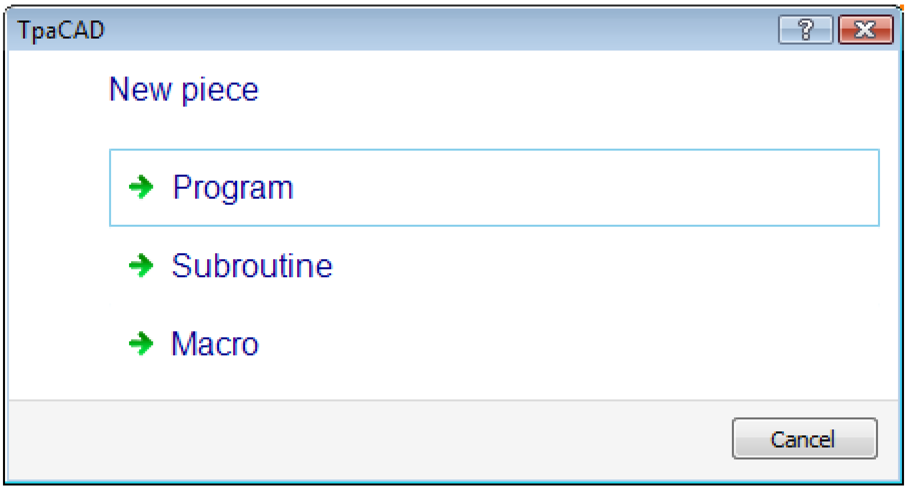
- general information is subject to predefined rules (program variables, custom sections,...);
- it does not develop induced calls;
- it cuts by 1 the maximum admissible number for nesting the subprogram calls of another macro-program or of another macro-program;
- it cannot be executed;
- it does not generate any piece matrices.

A macro-program allows the use of some particular workings, all of logical statements: FOR cycles, direct assignments of local variables.
Also the macro-program, like a subprogram, allows a group of workings to be defined and then to be used time and time again. Furthermore, the group of workings may include logical cycles that cannot be defined in advance.

A macro-program can be written and modified at the Manufacturer (or higher) level only. Then, it is recalled within a complex working (normally called macro) that must be defined by the machine manufacturer. In this way the macro-program is defined and managed by the final user in a completely transparent way.

So, to create a macro-program you need:

- to activate the Manufacturer access level;
- to select the command New in the menu File and choose the entry Macro.



## 5.1 Area of assignments

In the assignment area the same sections and the same fields of a generic program are proposed, as follows:

- **Length, Height, Thickness**: dimensions of the piece.
- **Unit***:* unit of measurement of the piece ([mm] or [inch]).
- **Program Typology**: in this case the option macro-program is displayed (modification allowed). For example, starting from a program of different type it is possible to set the macro-program option.
- **Access and Edit level**: in both levels the option Manufacturer is set.
- **Description** of the program (modification allowed).

The dimensions and the unit of measurement set in this page are valid for local use only. When the macro-program is applied, the information of the main program in which it is called are used.

It is particularly important to pay attention to the field Unit:

- if an application always works with only one unit of measurement, no problem arises. In this case, programs, subprograms and macro-programs are all written in [mm] or in [inch].
- Attention is necessary, if TpaCAD can work with programs written in [mm] or in [inch].

Let us examine the following example:

- you write a macro-program (ONE) in [mm] unit;
- the macro-program executes drilling works, spaced each other by an offset in x;
- the distance between holes is assigned in an r variable (public): e.g. r3;
- however, you may wish to apply a minimum 20 mm distance. r3 and 20 are compared.

There are no problems, even if the program (say: PRG1) using UNO is written in [mm].
If PRG1 is written in [inch]:

- number 20 is interpreted as 20 [inch] (a lot);
- r3 is set to [inch]. The operator writing PRG1 now thinks in [inch];
- in this case, it is no longer valid to compare r3 directly with 20.

Both cases are valid, if the comparison is made with the quantity “20*cnq” (see TpaCAD manual, chapter "Parametric Programming"):
20*cnq is worth 20, if the program PRG1 has [mm] as unit of measurement.
20*cnq is worth 0,7874, if the program PRG1 has [inch] as unit of measurement.

In order that the macro-program (and under-program) may be recalled into the programs, regardless of their unit of measure, it is essential to use the cnq factor.

Though applications with different unit of measure are not foreseen, we recommend the use of the cnq factor, because it frees the macro-program (and macro-program) from the configured unit of measure.

All the macro-programs basically supplied with TpaCAD are written in unit [mm] and use, where necessary, the cnq factor. We suggest this procedure as a rule to avoid dangerous errors and worthless waste of time.

"o" and "v" variables are managed as defined in the configuration. Nevertheless, the use for the maximum assignable configuration is allowed. When the macro-program is used, these variables are assigned by the main program.
The macro-programs basically supplied with TpaCAD do not assign and do not use the "o" and "v" variables.

The "r" managed variables , by means of which the information pass from the main program to the macro program, are always 300.
As already seen, in case of a call of a sub-routine, it possible to assign each single "r" variable marked as reassignable one (public). Thus, the development of the macro-program can be totally modified on the basis of how the "r" public variables are passed, as well as for a subprogram.

Custom sections are managed as defined in the configuration.
The macro-programs basically supplied with TpaCAD do not assign and do not use variable arguments defined for the sections.

Variable Geometry section is managed as defined in the configuration.
The macro-programs basically supplied with TpaCAD do not assign variable Geometries.

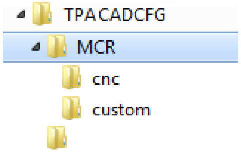
The execution Sequences section is managed as defined in the configuration.
Assigning the execution sequences is not required, if the codes of macro configured for the application of the macro-program, exclude the application of the sequences in the expanded list of the macro-program.
If not, execution sequences must be assigned.

## 5.2 How to save a macro-program

A macro-program is saved with extension "TMCR".

To choose the folder where to record the file, it is advisable to comply with the following rules:

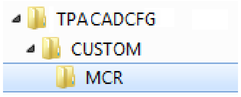
- the folder we suggest is tpacadcfg\mcr, (e.g., created in c:\albatros):



in tpacadcfg\mcr we find the macro-programs supplied with the installation of TpaCAD

- record the macro-program:
  - directly i tpacadcfg\mcr, or
  - in a child folder (e.g. tpacadcfg\mcr\cust)

  but only for a possible correction and/or modification that might be necessary for a base macro-program.



For a custom file we recommend to:

- record a file starting from: tpacadcfg\custom\mcr.

In this case the macro-programs written for a specific application (custom macro-programs) remain also in the files and in the custom folders and make the recovery of all the customisations of an installation more immediate.

# 5.3 How to program a macro-program

The faces of a macro-program are programmed as a generic program.
The selection includes real and fictive faces:

- the real faces are always six. Also those which have been disabled during the configuration are displayed.
- The fictive faces are those that have been assigned in the section of the variable Geometries, except for the faces set as construction auxiliaries.
- Face-piece does not appear.

Before starting to define a face program it must be quite clear how you intend to use the macro-program.
For this purpose let us examine some particular situations.

**Macro-program of multiple boring toward x**
Let us open the macro-program, supplied with the installation of TpaCAD, repeatx.tmcr, which is in the folder tpacadcfg\mcr\. The macro develops the cycles of blade workings.
The macro develops the multiple drilling cycles in x-direction. More specifically, in the palette of the workings it corresponds to the selection of the following processes:

| | |
|---|---|
| **[FITTINGX] Fitting X** | develops drilling works with constant step, given the extreme towards x (starting X and arriving X). |
| **[HOLEX] Repeat X** | develops drilling works with constant step, given the starting X-coordinate and the total number of the drilling works |

[FITTINGX] working runs the program assigned in face 1 of the macro-program.
[HOLEX] working runs the program assigned in face 2 of the macro-program.

In this case: the macro-program is used as a container of functions and its application must specify which function you intend to choose. Here the choice is made on the face:

- in the case of FITTING the program written in face 1 is run;
- in the case of REPEAT the program written in face 2 is run.

**Macro-program working a blade**
Let us open the macro-program, supplied with the installation of TpaCAD, lame.tmcr, which is located in the folder tpacadcfg\mcr\.
The macro develops the working cycles of a blade. More specifically, in the palette of the workings it corresponds to the selection of the following processes:

| | |
|---|---|
| **[BLADEX] X - sawing work** | performs a blade working towards x |
| **[BLADEY] Y - sawing work** | performs a blade working towards y |
| **[BLADEXY] XY - sawing working** | performs a blade working inclined towards xy. |

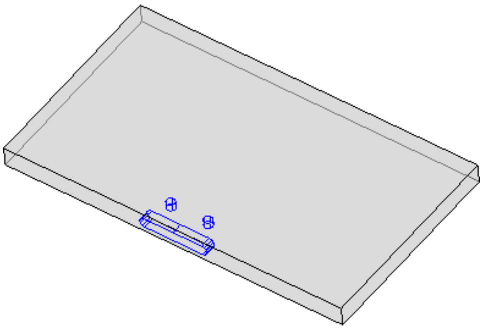
In all the cases the working performs the program assigned in face 1 of the macro-program.
Also in this case: the macro-program is used as a container of functions and its application must specify which function the user intends to choose. Here the choice is made through a public r- variable (r9) and the groups of workings on the value of the r9 - variable are conditioned by the program written in face 1:

```
IF r9=0
… develops the blade working in x
END IF

IF r9=1
… develops the blade working in y
END IF

IF r9=2
```

```
… develops the working of inclined blade
END IF
```

In the definition of the code of macro, the r9 - variable will have to be set as hidden, with default value, respectively: 0, 1, 2.

**Working macro-program for lock-handle of a door**
In this case we have to write the macro-program.
We want to be able to perform all the workings needed to cavities to insert the lock and the handle of a door:



- cavity for the handle and the key in face 1;
- cavity of the lock in side face (face 3).

The picture simplifies the problem and shows the external cavities (without marking the cycles for the emptying of the areas).

In face 1 we can program two circles with fixed dimensions and distance or parameters (with public r-variables).
In face 3 we program a closed pocket-shaped profile.

A reciprocal position towards x is assigned with programming of application points:

- in face 1 the application point coincides with the centre of the circle on the left;
- in face 1 the application point coincides with the centre of the semi-circle of the cavity, on the left.

Let us save the macro-program as serratura.tmcr in the folder tpacadcfg\custom\mcr

To check the application of the macro, without being obliged to assign a code of macro at once, let us activate the entry **Allow macro call in SUBnn codes**, in TpaCAD configuration (WARNING: do not forget later to deactivate the entry).
Let us create a new program and let us open the face 1:

- in the palette of the workings let us choose the SUB code;
- in the field Subroutine select the macro-program tpacadcfg\custom\mcr\serratura.tmcr;
- let us confirm the working insertion.

In face 1 we see the execution of two circles.
Also in face 3 we have an (induced) call of the macro-program. Let us see the execution of the pocket.

In the call in face 1 let us try to change the x - coordinate of the application. We will see that also the pocket in face 2 moves consequently.

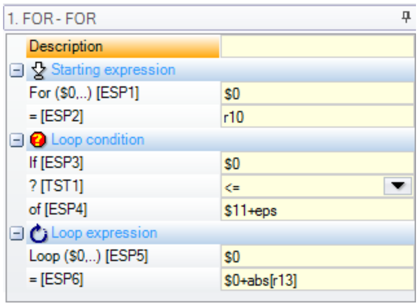
Here the macro-program is not longer used as a container of functions, but rather as a prototype for the program that executes it. The macro-program matches the program structure, as though it were summed to it, both in the geometric component and in that of the structure of the programmed text.

# 5.4 Logical instructions that can be used in a macro program

Logical instructions are particular simple workings that do not generate any profile machining.
During the writing of a macro-program all the available logical instructions for a program are available: IF..ELSE..ENDIF structure, assignment of <j> variables, global functions, application points.
Additionally, further instruction are available and can be used in the macro-programs only.

## FOR...ENDFOR instructions

In the group of LOGICAL INSTRUCTIONS we find the following instructions: FOR..ENDFOR.
The FOR instructionallows one or more workings to be repeated for a fixed number of times. A FOR instruction must always be closed by an **ENDFOR** instruction, which marks out the workings conditioned by **FOR**.
The **FOR** instruction is executed from zero to a variable number of time, until the condition imposed is verified.



1. FOR - FOR

| | |
|---|---|
| Description | |
| Starting expression | |
| For ($0,..) [ESP1] | $0 |
| = [ESP2] | r10 |
| Loop condition | |
| If [ESP3] | $0 |
| ? [TST1] | <= |
| of [ESP4] | $11+eps |
| Loop expression | |
| Loop ($0,..) [ESP5] | $0 |
| = [ESP6] | $0+abs[r13] |

**Initial expression:** assigns a value to an initial counter

**Loop condition:** indicates the condition, which allows carrying on in the FOR cycle

**Loop expression:** performs an assignment operation, which is repeated at every cycle.

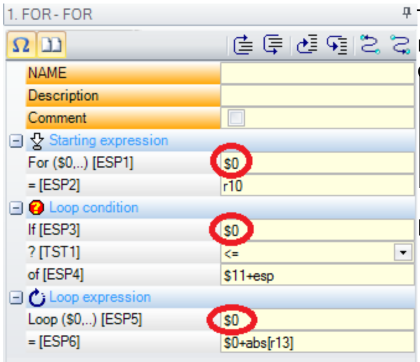
No term is obligatory. Not assigned terms, are not evaluated.
FOR… ENDFOR structures can be nested without limits.

FOR cycle in the picture corresponds to the evaluation of a logic expression:

**FOR** $0=r10 to $0<=$11+eps; $0=$0+abs[r13]
**{**
.
**} ENDFOR**

- $0 counter is assigned to the r10 initial value (parametric value);
- the cycle is repeated until the value of the $0 counter remains less or equal (<=) to the comparison value calculated in ($11+eps). The assigned comparison test ($0<=$11+eps) is performed also for the initial assignment (in the example: $0=r10). Therefore, the number of the performed repetition can be invalid, if the assignment does not verify the comparison;
- at every repetition after the first one, the value of the $0 counter is increased by (abs[r13]).

The here examined programming introduces the concept of macro-program local variables, generally shown as $nn (nn of assigned value from 0 to 299). In the next paragraphs we will examine these variables.



1. FOR - FOR

| | |
|---|---|
| NAME | |
| Description | |
| Comment | □ |
| Starting expression | |
| For ($0,..) [ESP1] | $0 |
| = [ESP2] | r10 |
| Loop condition | |
| If [ESP3] | $0 |
| ? [TST1] | <= |
| of [ESP4] | $11+esp |
| Loop expression | |
| Loop ($0,..) [ESP5] | $0 |
| = [ESP6] | $0+abs[r13] |

The use of variables of $nn type is reserved to the preparation of macro-programs to assign locally used variables. A FOR cycle counter is mainly used for this kind of variables.

More specifically, the circled fields in the picture aside, if set, must be a variable and display a name of $ variable.

Different variables for each of the three terms can be used.

If the cycle condition is not assigned, the cycle has to be managed through a BREAK instruction (see later).
**FOR**.....**ENDFOR** structures can be nested without limits.
In a macro-program up to max.100000 total interactions can be performed. Going beyond the maximum number of interactions, the development of the macro-program is broken off and an error is reported, This control is activated to avoid the development of endless cycles, resulting from a wrong writing of the macro text.
The maximum number of FOR cycles, than can be used in a face program, is 300.

## CONTINUE instruction

CONTINUE instruction programs some jump cases in the programmed test, forcing an immediate transfer of the control to the FOR instruction of the nearest external cycle.
The jump condition is expressed with a three terms formalism as in the IF instruction.
If the instruction condition is TRUE or is not set up, the instruction interprets a jump condition, so:

- it determines the direct return to the beginning of the FOR cycle at the nearest nesting level;
- in case that the instruction is not executed in a FOR cycle, this implies the return to the beginning of the program (realising in this way an implicit cycle of the whole face program).

If the test result is FALSE, the development of the macro is normally carried on. In any case the test result is TRUE, if no logical condition is required.

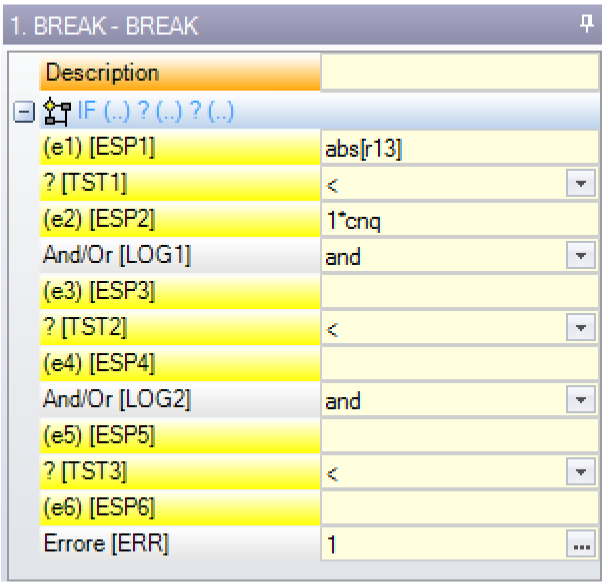
Example of structure FOR.. CONTINUE...ENDFOR:

```
FOR (..)
 ...
    FOR ($0=0; $0<=r12; $0=$0+1)

        CONTINUE ($0=r5 or $0=r6)    ; if the condition is verified, it jumps to the instruction FOR
   ...
    ENDFOR

ENDFOR
......
```

## BREAK instruction

The instruction BREAK

- allows the control transfer to be immediately forced to the instruction, which is next to the ENDFOR of the external nearest cycle;
- allows the programming of error situations, in the similar way as the case of the ERROR statement



1. BREAK - BREAK

| | |
|---|---|
| Description | |
| IF (..) ? (..) ? (..) | |
| (e1) [ESP1] | abs[r13] |
| ? [TST1] | < |
| (e2) [ESP2] | 1*cnq |
| And/Or [LOG1] | and |
| (e3) [ESP3] | |
| ? [TST2] | < |
| (e4) [ESP4] | |
| And/Or [LOG2] | and |
| (e5) [ESP5] | |
| ? [TST3] | < |
| (e6) [ESP6] | |
| Errore [ERR] | 1 |

If an error code with positive and non-null value is assigned in the instruction, then an error condition is programmed, otherwise a jump condition is programmed.

The BREAK statement in the picture programs an error condition, if it is (abs[r13] < 1*cnq). The development of the macro-program is cancelled and the custom error, corresponding to the number 1, is reported.
The entry Error opens an auxiliary box, which shows a list of error messages (number+message) and where a new message can be entered or edited.

If the instruction is used to condition a jump and if the conditioning of the instruction is TRUE or if it is not set:

- the macro-program execution jumps to the next instruction of the ENDFOR of the nearest external cycle;
- if the statement is not executed within a FOR cycle, the BREAK implies the end of the macro-program development.

Example of structure FOR.. BREAK … ENDFOR:

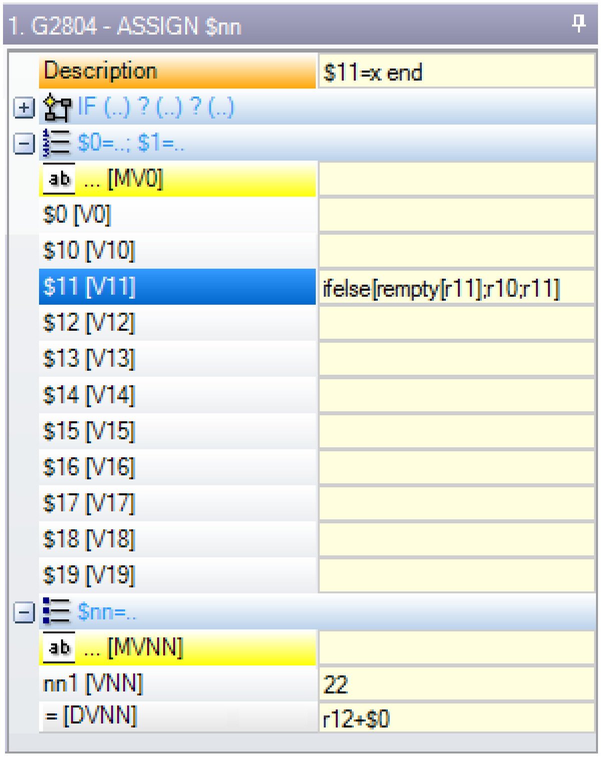
```
FOR (..)
...
   FOR ($0=0; $0<=r12; $0=$0+1)
     …
     BREAK ($0=r5 or $0=r6) ; if the condition is verified, it jumps to the statement following the
     ENDFOR.
   ENDFOR
     ...
ENDFOR
```

## Assigning $nn working variables

<$> variables are local variables declared in a macro-program. They are 300 variables (identified by name from $0 to $299) of numeric type.
At the beginning of the macro-program development the <$> variables are all initialized to zero. In case of nested macro-program applications (e.g. MACRO 1 macro, calling MACRO2 macro), each development level falls into its own <$> variables.

A <$> variable can be used in each working applied to a macro-program face and, for instance, can set the drilling diameter or a working position or a logical conditioning.

To assign some <$> variables several instructions are available.

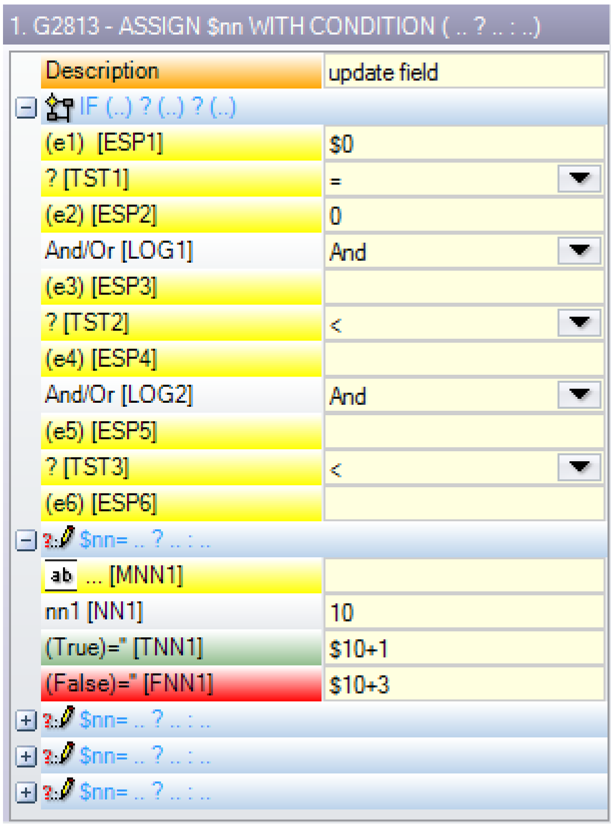


1. G2804 - ASSIGN $nn

| | |
|---|---|
| Description | $11=x end |
| ⊞ IF (..) ? (..) ? (..) | |
| ⊟ $0=..; $1=.. | |
| ab ... [MV0] | |
| $0 [V0] | |
| $10 [V10] | |
| $11 [V11] | ifelse[rempty[r11];r10;r11] |
| $12 [V12] | |
| $13 [V13] | |
| $14 [V14] | |
| $15 [V15] | |
| $16 [V16] | |
| $17 [V17] | |
| $18 [V18] | |
| $19 [V19] | |
| ⊟ $nn=.. | |
| ab ... [MVNN] | |
| nn1 [VNN] | 22 |
| = [DVNN] | r12+$0 |

The "Assign $nn" instruction allows the value of one or more <$> variable to be set.

- Assignments are performed only if the set logical conditions are verified.
- The first $0=... node groups a number of direct assignments. The entries shown are 20, from $0 to $19 (for the first 10 variables a comment in the field ...[MV..] can also be written. In the following example the variable: $11=ifelse[rempty[r11];r10;r11] is assigned.
- The following ($nn=..) nodes enable to assign the same number of variables, by specifying the variable index. In this example a variable
  - $22=r12+$0 is assigned, where $0 is previously assigned also in the same instruction.



1. G2813 - ASSIGN $nn WITH CONDITION ( .. ? .. : ..)

| | |
|---|---|
| Description | update field |
| IF (..) ? (..) ? (..) | |
| (e1) [ESP1] | $0 |
| ? [TST1] | = |
| (e2) [ESP2] | 0 |
| And/Or [LOG1] | And |
| (e3) [ESP3] | |
| ? [TST2] | < |
| (e4) [ESP4] | |
| And/Or [LOG2] | And |
| (e5) [ESP5] | |
| ? [TST3] | < |
| (e6) [ESP6] | |
| $nn= .. ? .. : .. | |
| ab ... [MNN1] | |
| nn1 [NN1] | 10 |
| (True)=" [TNN1] | $10+1 |
| (False)=" [FNN1] | $10+3 |
| $nn= .. ? .. : .. | |
| $nn= .. ? .. : .. | |
| $nn= .. ? .. : .. | |

The "Assign $nn with condition" instruction allows the assignment of one or more <$> variables by the evaluation of logical conditions. For each variable a node is assigned with three available fields:

- a first field sets the variable index (value from 0 to 299);
- the second field indicates the assignment to make, if the condition is verified;
- the third field indicates the assignment to make, if the condition is not verified.

In the picture the example is as follows:

- the condition is: ($0 = 0)
- the variable set is $10

if the condition is TRUE, it performs: $10=$10+1
if the condition is FALSE, it performs: $10=$10+3

## Assigning jnn working variables

In the macro-program text <j> variables can be written and verified like in a program or a subprogram.
J-variables are 100 variables, identified by name from j0 to j99 of numeric type.
<j> variables are located in the face in which the macro-program is applied, so:

- at the beginning of the application the macro-program sees the <j> variables as set by the workings applied before.
- At the end of the macro-program application, the <j> applications can have been changed by the macro-program itself.

The j-variables can be used to return one or more information after the application of a macro-program.

## IF...ELSE...ENDIF structure

In the macro-program text IF... ELSE... ENDIF structures can be programmed like in a program or in a subprogram.
IF... ELSE... ENDIF conditioning structures can be nested without limits and can include or be included in FOR...ENDFOR structures.
In an IF structure the EXIT instruction can still be used.

IF closure with ENDIF instruction is obligatory, unless among the IF instruction parameters the open IF field is not selected.
In case of open IF, the conditioning is not correct, if the next working is:

- logical instruction (IF, ELSE, ENDIF)
- cycle instruction (FOR, ENDFOR)
- point of application instruction

The conditioning of an open IF, if used in a macro-program, also affects:

- instruction of assignment of (j or $) variables
- setup or profile workings

## 5.5 Evaluating a text while writing a macro-program text

While writing a macro-program, verification and development of the text are only partially evaluated. Cycle conditions and conditioned jumps are not applied. Thus:

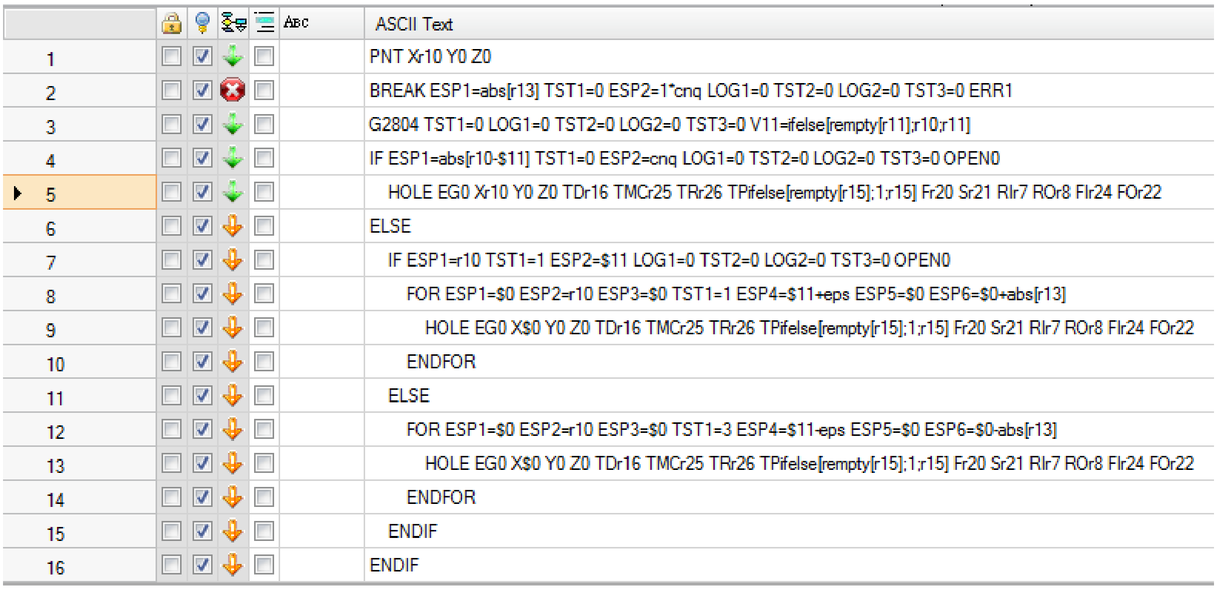
- in a FOR cycle only the assignment of the initial expression is evaluated. It is not developed by the number of the repetitions set.
- CONTINUE and BREAK instructions do not affect in any way the sequential development of the text.
- Logical conditionings
- Settings of the <$> and <j> variables are evaluated instead.

To see how the logical conditionings are applied, the application of the following logical conditions must be required:

- workings which do not verify logical conditions are visible
- workings which do not verify logical conditions are not visible

The logical status of the workings is also shown in the ASCII text.
As an example let us open the macro-program tpacadcfg\mcr\repeatx.tmcr, basically supplied with TpaCAD

- let us set the r12=0 variable to reset the x-fitting step
- let us open the face 1
- let us request the application of the logical conditions.



| | | | | | ABC | ASCII Text |
|---|---|---|---|---|---|---|
| 1 | □ | ☑ | ↓ | □ | | PNT Xr10 Y0 Z0 |
| 2 | □ | ☑ | ✖ | □ | | BREAK ESP1=abs[r13] TST1=0 ESP2=1*cnq LOG1=0 TST2=0 LOG2=0 TST3=0 ERR1 |
| 3 | □ | ☑ | ↓ | □ | | G2804 TST1=0 LOG1=0 TST2=0 LOG2=0 TST3=0 V11=ifelse[rempty[r11];r10;r11] |
| 4 | □ | ☑ | ↓ | □ | | IF ESP1=abs[r10-$11] TST1=0 ESP2=cnq LOG1=0 TST2=0 LOG2=0 TST3=0 OPEN0 |
| ▸ 5 | □ | ☑ | ↓ | □ | | HOLE EG0 Xr10 Y0 Z0 TDr16 TMCr25 TRr26 TPifelse[rempty[r15];1;r15] Fr20 Sr21 RIr7 ROr8 FIr24 FOr22 |
| 6 | □ | ☑ | ⇩ | □ | | ELSE |
| 7 | □ | ☑ | ⇩ | □ | | IF ESP1=r10 TST1=1 ESP2=$11 LOG1=0 TST2=0 LOG2=0 TST3=0 OPEN0 |
| 8 | □ | ☑ | ⇩ | □ | | FOR ESP1=$0 ESP2=r10 ESP3=$0 TST1=1 ESP4=$11+eps ESP5=$0 ESP6=$0+abs[r13] |
| 9 | □ | ☑ | ⇩ | □ | | HOLE EG0 X$0 Y0 Z0 TDr16 TMCr25 TRr26 TPifelse[rempty[r15];1;r15] Fr20 Sr21 RIr7 ROr8 FIr24 FOr22 |
| 10 | □ | ☑ | ⇩ | □ | | ENDFOR |
| 11 | □ | ☑ | ⇩ | □ | | ELSE |
| 12 | □ | ☑ | ⇩ | □ | | FOR ESP1=$0 ESP2=r10 ESP3=$0 TST1=3 ESP4=$11-eps ESP5=$0 ESP6=$0-abs[r13] |
| 13 | □ | ☑ | ⇩ | □ | | HOLE EG0 X$0 Y0 Z0 TDr16 TMCr25 TRr26 TPifelse[rempty[r15];1;r15] Fr20 Sr21 RIr7 ROr8 FIr24 FOr22 |
| 14 | □ | ☑ | ⇩ | □ | | ENDFOR |
| 15 | □ | ☑ | ⇩ | □ | | ENDIF |
| 16 | □ | ☑ | ⇩ | □ | | ENDIF |

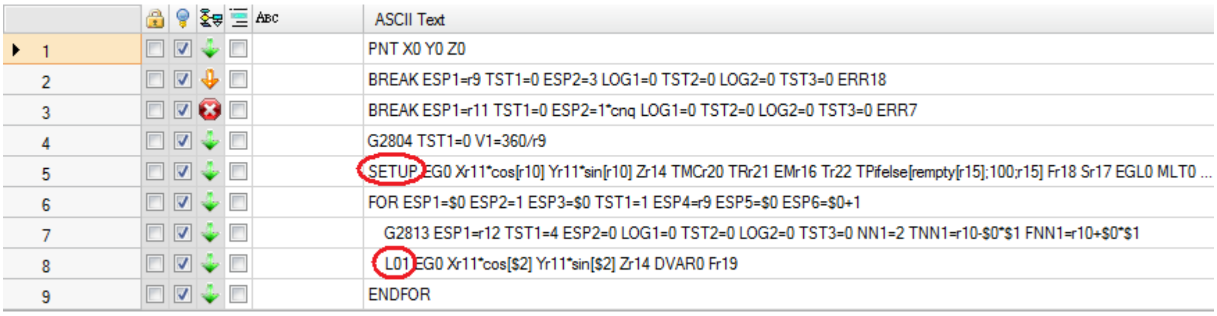
Let us structure the text so that it can be more directly read:

| | |
|---|---|
| ↓ PNT (X=r10 Y=0 Z=0) | application point of the macro |
| ✖ BREAK (abs[r13] < 1* cnq) error=1 | evaluates the fitting step: ERROR if less than (1*cnq):<br>(stop icon ✖ shows that the condition is verified. During the application it would cancel the interpretation of the macro-program because of an error condition. |
| ↓ G2804 ($11=ifelse[rempty[r11];r10;r11]) | Assign the $11 variable to the final position of the fitting |
| ⇩ IF (abs[r10-$11]< cnq) | Evaluates coinciding initial X and final X (by less than cnq)<br>the icon ⇩shows that it is verified as FALSE: during the application the IF ..ELSE ..ENDIF structure would develop the first branch; |
| ⇩ HOLE (X=r10 Y=0 Z=0) | Drilling on initial fitting point |
| ↓ ELSE | ELSE branch: the icon ↓ shows that it is verified as TRUE: during the application the development carries on this branch |
| ↓ IF (r10 <= r11) | The icon ↓ shows that the condition is verified as TRUE. During the application IF ..ELSE ..ENDIF structure would develop the first branch; |

| | |
|---|---|
| ↓ FOR ($0=r10; $0 >= r11+eps; $0=$0+abs[r12]) | Drilling execution cycle The cycle es the $0 variable, increases it at each execution and ends, when the condition ($0 <= r11+eps) is verified as FALSE. |
| ↓ HOLE (X=$0 Y=0 Z=0) | Drilling |
| ↓ ENDFOR | |
| ↓ ELSE | executes, if X initial fitting > final X. Stop icon ↓ shows that the condition is not verified. During the application IF..ELSE..ENDIF would not develop this branch |
| ↓ FOR ($0=r10; $0 >= r11-eps; $0=$0-abs[r12]) | Drilling execution cycle |
| ↓ HOLE (X=$0 Y=0 Z=0) | Drilling |
| ↓ ENDFOR | |
| ↓ END IF | |

## 5.6 Building profiles

Writing a macro-program, a profile can be developed with interpretation of logical conditions, cycle structures, assignment of variables.
For instance:
Let us open the macro-program basically supplied with TpaCAD*:* tpacadcfg\mcr\policer.tmcr and let us enable the view of face 1.



| | | | | | | ASCII Text |
|---|---|---|---|---|---|---|
| 1 | □ | ☑ | ↓ | □ | | PNT X0 Y0 Z0 |
| 2 | □ | ☑ | ↓ | □ | | BREAK ESP1=r9 TST1=0 ESP2=3 LOG1=0 TST2=0 LOG2=0 TST3=0 ERR18 |
| 3 | □ | ☑ | ✖ | □ | | BREAK ESP1=r11 TST1=0 ESP2=1*cnq LOG1=0 TST2=0 LOG2=0 TST3=0 ERR7 |
| 4 | □ | ☑ | ↓ | □ | | G2804 TST1=0 V1=360/r9 |
| 5 | □ | ☑ | ↓ | □ | | SETUP EG0 Xr11*cos[r10] Yr11*sin[r10] Zr14 TMCr20 TRr21 EMr16 Tr22 TPifelse[rempty[r15];100;r15] Fr18 Sr17 EGL0 MLT0 ... |
| 6 | □ | ☑ | ↓ | □ | | FOR ESP1=$0 ESP2=1 ESP3=$0 TST1=1 ESP4=r9 ESP5=$0 ESP6=$0+1 |
| 7 | □ | ☑ | ↓ | □ | | G2813 ESP1=r12 TST1=4 ESP2=0 LOG1=0 TST2=0 LOG2=0 TST3=0 NN1=2 TNN1=r10-$0*$1 FNN1=r10+$0*$1 |
| 8 | □ | ☑ | ↓ | □ | | L01 EG0 Xr11*cos[$2] Yr11*sin[$2] Zr14 DVAR0 Fr19 |
| 9 | □ | ☑ | ↓ | □ | | ENDFOR |

The text develops a polygon inscribed in a circle. The circled codes mark off the text lines that develop the execution profile of the polygon.

Let us arrange the text in a scheme in a more legible way:

| | |
|---|---|
| PNT (X=0 Y=0 Z=0) | macro application point (centre of the polygon) |
| BREAK (r9 < 3) error=18 | evaluates the number of the sites of the polygon: ERROR if less than 3 |
| BREAK (r11 < 1*cnq) error=7 | values the radius of the polygon: ERROR if less than (1*cnq) |
| G2804 ($1=360/r9) | writes $1 to the value of the angle subtended by a side of the polygon |
| SETUP (X=r11*cos[r10] Y=r11*sin[r10] Z=r14) | initial setup of the polygon |
| FOR ($0=1; $0 <= r9; $0=$0+1) | cycle of polygon development |
| G2813 ($2=(r12=0) ? (r10-$0*$1):(r10+$0*$1)) | the cycle initializes the variable $0 (and counts the number of the sides), increments it at each execution and sorts when it verifies FALSE the condition ($0 <= r9); |
| L01 (X=r11*cos[$2] Y=r11*sin[$2] Z=r14) | writes $2 with evaluation of the r12 value<br>(0: clockwise rotation;<>0: counterclockwise rotation) |
| ENDFOR | $0-th side of the polygon. Note: the linear does not set the initial application point |

Some text peculiarities of the macro-program are:

- the setup is not immediately executed by the linear workings that develop the profile
- one only linear is programmed; this is then repeated in a FOR..ENDFOR cycle (as it happened in the previous example of the x-fitting for the drillings
- profile continuity from the setup to the closure of the polygon is automatically guaranteed by the same development of the macro-program. The only lines, kept by the development, are here SETUP and the L01 linears generated in the FOR..ENDFOR cycle.

<u>The instructions listed below can be entered and they condition the profile construction:</u>

- cycle instructions (FOR, ENDFOR)
- jump and/or error instructions (CONTINUE, BREAK, ERRORE, EXIT)
- instructions to assign (<$>, <j>) variables
- structure instructions (IF, ELSE, ENDIF).

## 5.7 How to configure a macro code

The direct application of a macro-program by means of a generic subprogram code, without the assignment of a macro code, must be limited to the test preliminary phase and be deactivated in the final configuration of a TpaCAD installation.
The application of a macro-program always requires a macro code (code of fixed cycle).

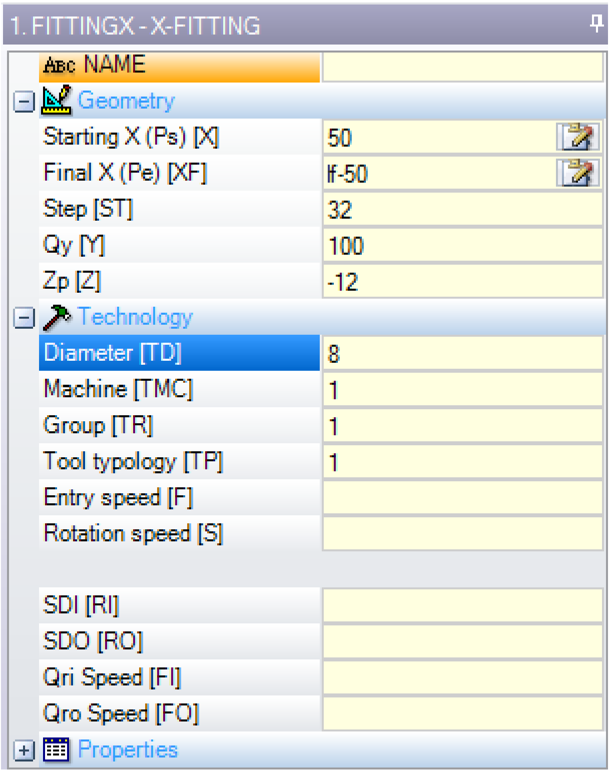
A macro code must be assigned in the database of the workings. For a more detailed explanation on how to create a macro code, see the TpaWorks manual.

### How to verify the application of a macro

TpaCAD supplies a tool to verify a macro-program. To enable it, the activation of the debug functionality in Configuration is required. The debug functionality adds a page in the area dedicated to view the errors, the commands and the j-variables.

A file with a fixed name is read; this file is updated according to the latest application of a macro code (but also of a sub-routine).
The debug file is updated, when the change of a complex code is entered of modified.

Let us create a new program. Let us enter in face 1 and edit FITTING X code by setting the data as in the figure.

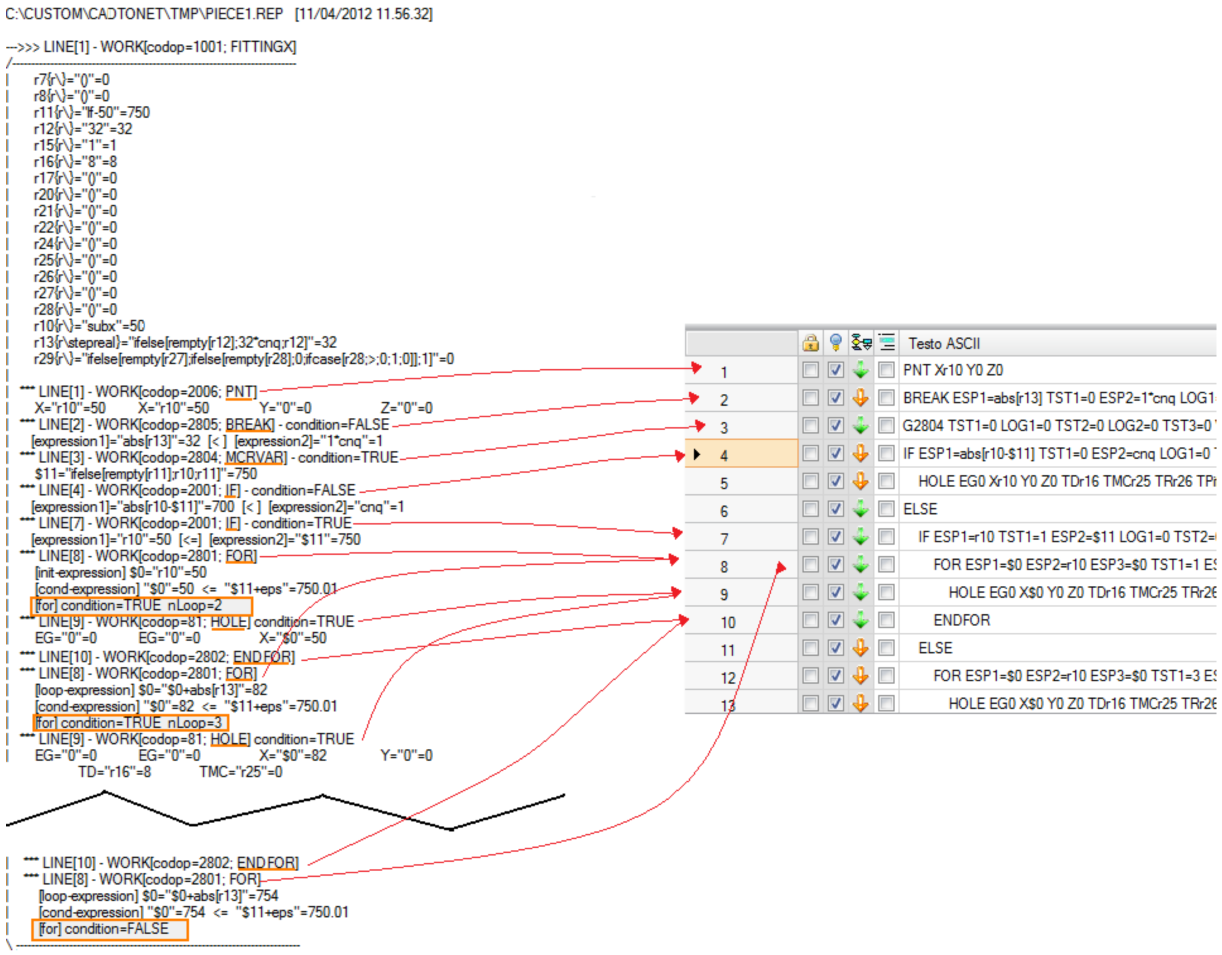


1. FITTINGX - X-FITTING

| | |
|---|---|
| NAME | |
| **Geometry** | |
| Starting X (Ps) [X] | 50 |
| Final X (Pe) [XF] | lf-50 |
| Step [ST] | 32 |
| Qy [Y] | 100 |
| Zp [Z] | -12 |
| **Technology** | |
| Diameter [TD] | 8 |
| Machine [TMC] | 1 |
| Group [TR] | 1 |
| Tool typology [TP] | 1 |
| Entry speed [F] | |
| Rotation speed [S] | |
| SDI [RI] | |
| SDO [RO] | |
| Qri Speed [FI] | |
| Qro Speed [FO] | |
| **Properties** | |

Press the button **[Enter]** to confirm

Now let us open the tpacadcfg\mcr\repeatx.tmcr macro program, enter the face 1 and open the debug file.
In the opened window, in the title, the name of the debug file (here: C:\ALBATROS\TMP\PIECE.REP) and his creation date appear.
The figure shows the debug file next to the text of the macro-program to whom it is related (x fitting development):



```
C:\CUSTOM\CADTONET\TMP\PIECE1.REP   [11/04/2012 11.56.32]

--->>> LINE[1] - WORK[codop=1001; FITTINGX]
/------------------------------------------------------------------
|    r7{r\}="()"=0
|    r8{r\}="()"=0
|    r11{r\}="lf-50"=750
|    r12{r\}="32"=32
|    r15{r\}="1"=1
|    r16{r\}="8"=8
|    r17{r\}="()"=0
|    r20{r\}="()"=0
|    r21{r\}="()"=0
|    r22{r\}="()"=0
|    r24{r\}="()"=0
|    r25{r\}="()"=0
|    r26{r\}="()"=0
|    r27{r\}="()"=0
|    r28{r\}="()"=0
|    r10{r\}="subx"=50
|    r13{r\stepreal}="ifelse[rempty[r12];32*cnq;r12]"=32
|    r29{r\}="ifelse[rempty[r27];ifelse[rempty[r28];0;ifcase[r28;>;0;1;0]];1]"=0
|
|  *** LINE[1] - WORK[codop=2006; PNT]
|    X="r10"=50       X="r10"=50          Y="0"=0            Z="0"=0
|  *** LINE[2] - WORK[codop=2805; BREAK] - condition=FALSE
|   [expression1]="abs[r13]"=32  [< ]  [expression2]="1*cnq"=1
|  *** LINE[3] - WORK[codop=2804; MCRVAR] - condition=TRUE
|    $11="ifelse[rempty[r11];r10;r11]"=750
|  *** LINE[4] - WORK[codop=2001; IF] - condition=FALSE
|   [expression1]="abs[r10-$11]"=700  [< ]  [expression2]="cnq"=1
|  *** LINE[7] - WORK[codop=2001; IF] - condition=TRUE
|   [expression1]="r10"=50  [<=]  [expression2]="$11"=750
|  *** LINE[8] - WORK[codop=2801; FOR]
|    [init-expression] $0="r10"=50
|    [cond-expression] "$0"=50  <=  "$11+eps"=750.01
|    [for] condition=TRUE  nLoop=2
|  *** LINE[9] - WORK[codop=81; HOLE] condition=TRUE
|    EG="0"=0         EG="0"=0           X="$0"=50
|  *** LINE[10] - WORK[codop=2802; ENDFOR]
|  *** LINE[8] - WORK[codop=2801; FOR]
|    [loop-expression] $0="$0+abs[r13]"=82
|    [cond-expression] "$0"=82  <=  "$11+eps"=750.01
|    [for] condition=TRUE  nLoop=3
|  *** LINE[9] - WORK[codop=81; HOLE] condition=TRUE
|    EG="0"=0         EG="0"=0           X="$0"=82          Y="0"=0
|         TD="r16"=8         TMC="r25"=0

|  *** LINE[10] - WORK[codop=2802; ENDFOR]
|  *** LINE[8] - WORK[codop=2801; FOR]
|    [loop-expression] $0="$0+abs[r13]"=754
|    [cond-expression] "$0"=754  <=  "$11+eps"=750.01
|    [for] condition=FALSE
\------------------------------------------------------------------
```

| | | | | | Testo ASCII |
|---|---|---|---|---|---|
| 1 | ☐ | ☑ | ↓ | ☐ | PNT Xr10 Y0 Z0 |
| 2 | ☐ | ☑ | ↓ | ☐ | BREAK ESP1=abs[r13] TST1=0 ESP2=1*cnq LOG1 |
| 3 | ☐ | ☑ | ↓ | ☐ | G2804 TST1=0 LOG1=0 TST2=0 LOG2=0 TST3=0 |
| ▶ 4 | ☐ | ☑ | ↓ | ☐ | IF ESP1=abs[r10-$11] TST1=0 ESP2=cnq LOG1=0 |
| 5 | ☐ | ☑ | ↓ | ☐ | HOLE EG0 Xr10 Y0 Z0 TDr16 TMCr25 TRr26 TP |
| 6 | ☐ | ☑ | ↓ | ☐ | ELSE |
| 7 | ☐ | ☑ | ↓ | ☐ | IF ESP1=r10 TST1=1 ESP2=$11 LOG1=0 TST2= |
| 8 | ☐ | ☑ | ↓ | ☐ | FOR ESP1=$0 ESP2=r10 ESP3=$0 TST1=1 E |
| 9 | ☐ | ☑ | ↓ | ☐ | HOLE EG0 X$0 Y0 Z0 TDr16 TMCr25 TRr2 |
| 10 | ☐ | ☑ | ↓ | ☐ | ENDFOR |
| 11 | ☐ | ☑ | ↓ | ☐ | ELSE |
| 12 | ☐ | ☑ | ↓ | ☐ | FOR ESP1=$0 ESP2=r10 ESP3=$0 TST1=3 E |
| 13 | ☐ | ☑ | ↓ | ☐ | HOLE EG0 X$0 Y0 Z0 TDr16 TMCr25 TRr2 |

In the debug file at the beginning the r-variables assigned in the macro-program are listed: public variables first (r7, r8,...r27) and after the other ones (r10).
For all the complex codes following rules shall apply:

- the public variables, for which a non-empty setting string is given, take on the given assignment;
- the public variables, for which an empty string is set, take on the assignment specified in the original text of the sub-routine (or macro-program) or they take on an invalid value, according to the selection in Configuration of TpaCAD and also of the macro code.

Therefore, in the original text of the macro-program it is important to record some significant assignments (especially for variables establishing different behaviours in the execution of the subprogram) or, alternatively, determining a diagnostic situation (in this way the operator is obliged in any case to carry out a valid setting).

For the non-reassignable variables all setting strings are ignored and their value is recalculated on the basis of the setting specified in the original text of the macro/subprogram.

# 6 Custom database signature and macro-program Encryption

TpaCAD provides the tools to distribute a custom database of the workings and securely associated macro-programs. The procedure, explained in the following parameters, provides to the *Manufactuer* a good tool to customize the TPACAD application, where proper "plug-ins" can be prepared for the base operation.
We suggest reading carefully the whole section, in order to get a clear and complete view into this subject.
The following topics concern both the TpaCAD and TpaWorks applications and precondition a good preliminary knowledge of the same.

Examining the following paragraphs, we might start from an initial condition, where no custom database of working has been assigned, yet.

The features of the database signature are appointed to the machine constructor: To be activated they need a specific authorisation from an HW key, called *Enterprise*.
The *Enterprise* authorisation, including the features *Professional*, is only necessary for the manufacturer, while the final installation for the customers need the *Professional* key.

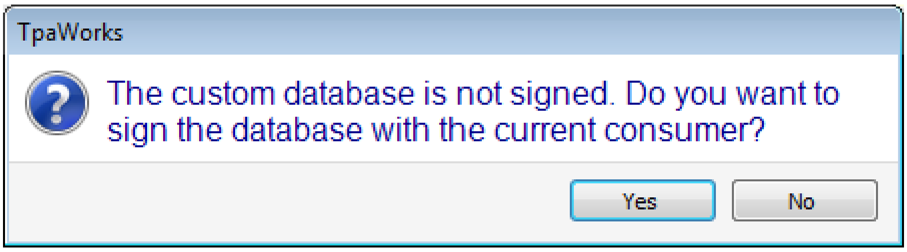
Opening the working database requires a manufacturer access level, that must be supplied by the TPA developer. Furthermore, this should be an unlimited access, like, for instance, that managed by a daily password, because it requires to recognize a specific account.

## 6.1 Signing the custom database

The first step consists in signing the custom database of the workings.
The signing operation is performed in a *TpaWorks* environment and requires an existing database: to create it, you need to open the same database and to close it, confirming the saving.
Then, create a Manufacturer user in TpaPass32 program and connect to the user (for example: "myname"): this is the owner user of the custom database.
Always in the *TpaWorks* environment, select the command **Account** in the menu ⚙. This command is available as follows:

- ✓ by recognizing TPACAD in *Enterprise* version*;*
- ✓ from Manufacturer password;
- ✓ if no database is loaded.

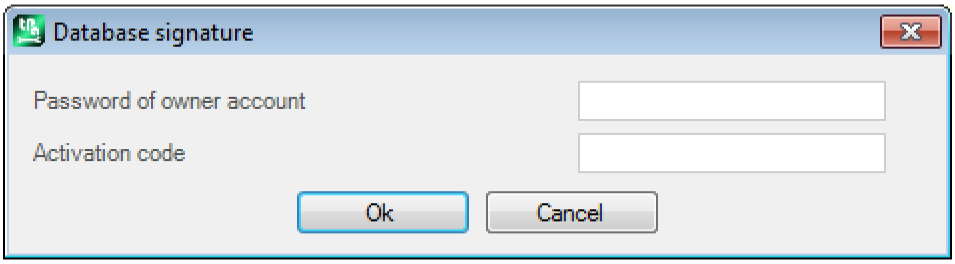
A message informs that the confirmed command performs the signature of the custom database by means of the current account ("myname", in our hypothesis):


TpaWorks

The custom database is not signed. Do you want to sign the database with the current consumer?

**"Myname" user must always be used to change the custom database.**

After confirming the enquiry, a window requires to set two password, as follows:


Database signature

Password of owner account

Activation code

- ✓ *Manufacturer account password:* must be associated to the user signature to increase the control in accessing the database by means of the owner account. The password must be set by the "myname" user for a full access to the custom database and the encrypted macros.
- ✓ *Activation code*: this is a code (password) to activate the use of the custom database in those sections the manufacturer did not want to assign in a secure way (see: use of encrypted macros). The code must be set after the installation of the TpaCAD package, which confirms the authorization to use the specific application. At the moment, we might not assign an activation code.

Let us take into account some cautions before assigning the two passwords:

- max. 20 characters;
- no use of space;
- attention to the CAPS-LOCK status to distinguish uppercase and lowercase letters;
- reading back and taking note of the typed password for the next usages.

Anyway, the window can be confirmed without assigning a password.
Window and setting confirmation is intentionally redundant, because the signing operation cannot be repeated.

**Now the database is signed**: you must remember account and passwords. The application of the signature involves two operations:

1. change of the custom database (..\TPACADCFG\CUSTOM\DBWCUST.WCAD);
2. creation of the signature file (..\TPACADCFG\CUSTOM\DBWCUST.LIC);
3. eventual licence activation (if an activation Code is assigned).

The signature (account, password) is saved in the two files (database and licence) obviously in an "encrypted" format:
it is the manufacturer that should remember the account ("myname") and the password.

The signature is saved in the two files (account, password), obviously under an "Encrypted" format: it is the operator that must remember the account and the password.

**The signature file cannot be directly changed:** its alternation makes the database signature not valid and the possibilities of change may be lost.
Information on the file distribution from the machine Manufacturer will be provided later.

**If the database is already signed**, the user ("myname", password) can decide to delete the signature, by recalling the same command **Account** from the menu ⚙. To delete the signature:

- both the signature file and the database must match the signature ("myname", password);
- access by means of "myname" user and if a *password* is associated to the signature, this must be confirmed in the window, besides the eventual *Activation code.*

Deleting the signature implies:

1. the change of the custom database (..\TPACADCFG\CUSTOM\DBWCUST.WCAD);
2. the removal of the signature file (..\TPACADCFG\CUSTOM\DBWCUST.LIC);
3. the eventual licence deactivation

However, why signing the custom database of the workings? We can answer in different ways:

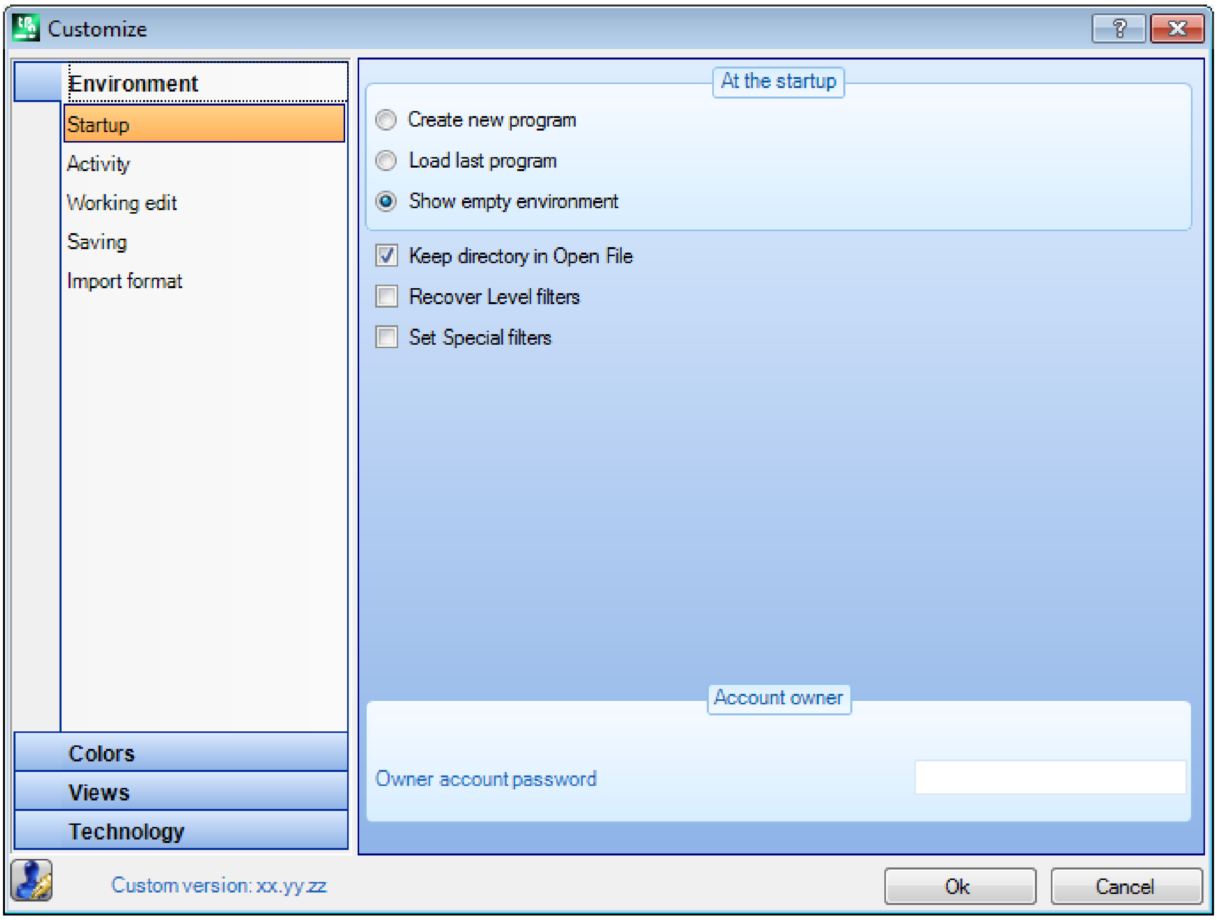
- to lock the changes to the custom database by non-enabled users;
- to lock changes to modifications to TpaCAD configuration;
- to write and use the encrypted (custom) macro-programs;
- to prevent non-enabled users from "decrypting" the workings that apply the same encrypted macro-programs.

If none of these circumstances is specifically required, do not proceed with the signature operation.

Finally, the *Activation code* deserves some careful considerations : why should it be assigned? We will answer shortly. Currently, we simply say that it is useful to assign an *Activation code* only if the use of encrypted macro-programs is required. If your only target, instead, is to lock the modifications to the same database, it is of no benefit to assign the *Activation code.*

## 6.2 How to encrypt a macro-program

The macro-program (custom) is written in a TpaCAD environment, which you access by username "myname". Even if a *Manufacturer account Password* has been assigned, you must complete the access as owner user by selecting the command **Customize** from the Application menu:



At the startup

○ Create new program

○ Load last program

● Show empty environment

☑ Keep directory in Open File

□ Recover Level filters

□ Set Special filters

Account owner

Owner account password

Custom version: xx.yy.zz

Let us see some particular fields:

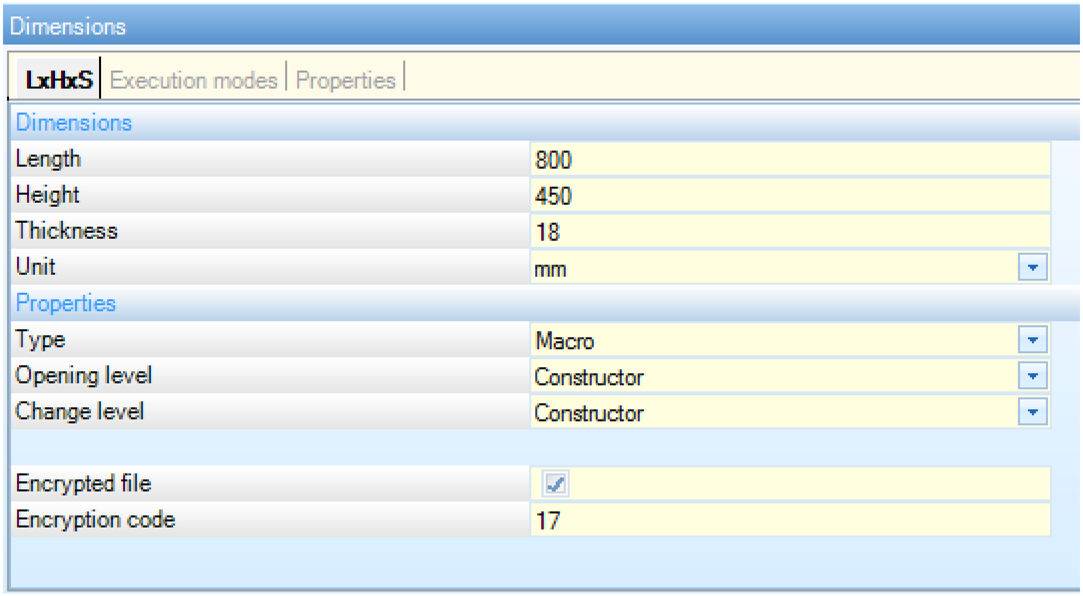
- ✓ : the picture at the bottom left-hand corner shows that the access is recognized by "myname" user;
- ✓ The display of the *Owner account* area in the first selectable tab shows that you need to set the Password to complete the Owner user access. Set password and confirm: subsequent openings of the window will keep the area invisible. Of course, the password will be requested only if it has been assigned in the activation of the database signature.

Anyway: you have to set the *Manufacturer account password* every time you start TpaCAD.

Following checks:

- ✓ TpaCAD recognition in a *Professional version:*
- ✓ "signed" custom database and valid signature file (both belonging to the user: "myname");
- ✓ access by ("myname"; password) user (must be at the manufacturer level);
- ✓ macro program type of the file

allow the activation of the macro program encryption.
Now, see how the dimension folder appears:



| | |
|---|---|
| **Dimensions** | |
| Length | 800 |
| Height | 450 |
| Thickness | 18 |
| Unit | mm |
| **Properties** | |
| Type | Macro |
| Opening level | Constructor |
| Change level | Constructor |
| Encrypted file | ☑ |
| Encryption code | 17 |

There are two new entries:

- **Encrypted file**: select the field to save as encrypted format;
- **Encryption code**: set an integer value from 0 to 99.

The encryption code has multiple valences:

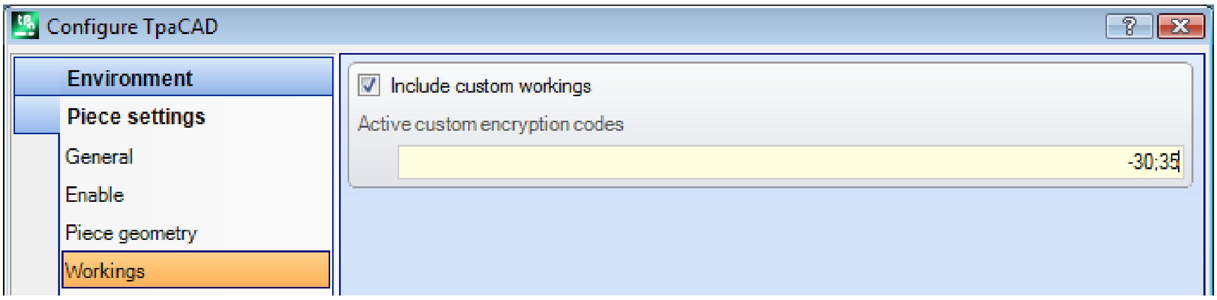
- ✓ it can require the check of a setting code in a HW key. In this case, you need to set a value from 1 to 16 corresponding to the same bit specific settings in the key. This is the case where a custom encryption is associated to a base encryption (base plug-in), already supplied by TPA. In this case, you have to set the **same enabling bit,** already checked by TPA;
- ✓ a 0 value or greater than 16 (accepted maximum value: 99) does not enable any setting check in a HW: this is a normal case of custom macro program assignment. Differentiating the values (greater than 16) allows you to create sub-groups of workings, that can be activated individually.
- ✓ Anyway, the value set here is used to create the correlation between the macro-program and a working code that applies it. So, it is important to decide which value has to be set.

An macro-program recorded as encrypted:

- ✓ can never be applied with a generic SUB code, also with corresponding access to the custom database signature;
- ✓ can never be exploded;
- ✓ manages the detailed compilation of the debug file, only if it is a recognition account of an owner user;
- ✓ can be opened as "unencrypted" only if you access with the same signature of the custom database;
- ✓ the format file is maintained as ASCII, but not as "unencrypted". Concerning this, it is not recommended that you try to change directly an encrypted test, in order to not compromise the correct interpretation.
- ✓ it can be executed only if it is combined with a custom database and with a signature file created with its same "signature".

Choosing to change the file type and or the encryption request, messages of specific situations can appear in accepting the changes. In a similar manner, if you cancel the encryption from a file already saved, a confirmation message appears in prompting the user to save.

Let us see now better how to activate different Encryption codes. In TpaCAD, launch the command **ConfigureTpaCAD**, from the menu ⚙, select the page **Workings**, in the section **Piece settings** :



☑ Include custom workings

Active custom encryption codes: -30;35

In the figure the upper part of the page is captured. Let us see now the field *Active custom encryption codes* to set the values that should be activated or (like in the figure) should be deactivated:

- ✓ empty field: so enabled code;
- ✓ "*" all the codes are enabled;
- ✓ "2;30;35": only the codes with value 2, 30 and 35 are activated;
- ✓ "-30;35": the string begins with '-': enable all the codes with the exception of the values (30, 35).

The field is significant only if the application loads up the Custom database and if this one is signed. It is realized that the assignment of the field is essential to activate the encryption codes: in fact, the empty field means to exclude them all. The custom database owner account only can modify the field. The string here displayed is saved in setting files in encrypted format.

## 6.3 How to use an encrypted macro-program

An encrypted macro-program can be used only if you assign one or more complex codes in the custom database.
In our example, you must open the custom database as user ("myname"; password).

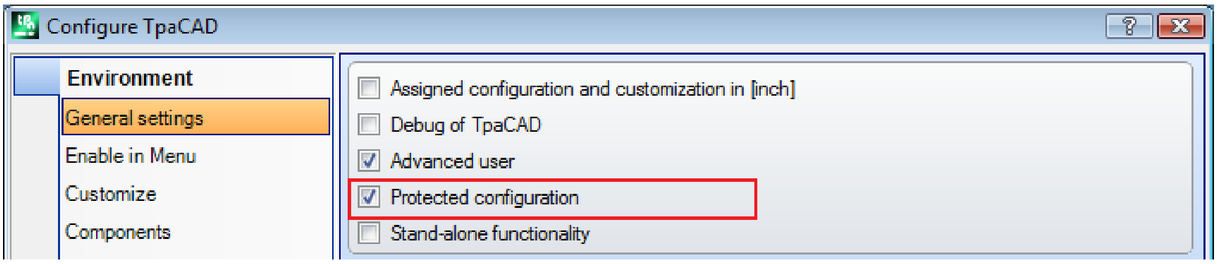
Then, continue with the usual selections to insert a new *Fixed Cycle* code: confirm for the automatic procedure of working setting by starting the macro-program in *Resource Manager* window.

You will see that in the *Insertion options* working attribute the field "bit=nn" has been set, where *nn* corresponds to the *Encryption code*indicated for the macro-program: in the example the figure "bit=17". If you change or delete the field, the working does not work. On the other hand: if you change the *Encryption code* of the macro-program, the working must be changed in the same way.

In the attribute of the working *Limit in transforms* the exploded view of the working is configured excluded. Anyway, controls are carried out in TpaCAD to prevent exploded views, but it is much better to assign this setting directly.

## 6.4 How to lock TpaCAD configuration

As already said, the signature of the custom database allows you to lock the TpaCAD configuration by means of a setting that the manufacturer can assign in the configuration:



☐ Assigned configuration and customization in [inch]
☐ Debug of TpaCAD
☑ Advanced user
☑ Protected configuration
☐ Stand-alone functionality

In the figure we see the field *Protected configuration* that is available through the access to the owner account; select the option to prevent an unqualified user from modifying the configuration. This locking function concerns the Custom functions and the Global variables.

## 6.5 Signature file distribution

At this point of our machine Manufacturer simulation we find:

- ✓ a TpaCAD key - *Enterprise* version
- ✓ a "privileged" user assigned in the TpaPass32 ("myname") program

- ✓ a "signed" custom database (user: “myname” and password)
- ✓ a signature file matching the database
- ✓ a macro-program saved in an encrypted format (for example: “ABC.TMCR”) from the user (“myname” and password)
- ✓ a Fixed Cycle working, that uses the encrypted macro-program (for example: operating code=3500, ASCII name=”W1ABC”): for convenience, let us label also the working as encrypted.

Now, we need to understand what should arrive at the final customer's computer.
Let us say first that **never** must an installation assign the “privileged” user (“myname”) in the program TpaPass32 (the accounts are assigned in the file Albatros\System\TpaPass.x: this file must not be copied) : the Manufacturer of the machine must make sure that the final customer will not be able to access the custom database and the programs with “myname” credentials.

Assuming that the application must work entirely, even if the the "W1ABC" working is applied:

- ✓ the HW key requested for TpaCAD must be PROFESSIONAL;
- ✓ if the “ABC.TMCR”*Encryption code* is between 1 and 16, the bits of the HW key must be already configured according to the specific TPA development settings already installed. We can say that usually no check will be necessary on the HW key;
- ✓ also the signature file must be distributed.

Assuming that the application must work excluding the application of "W1ABC" working: **the signature file must not be distributed.** Reading the custom database, the “W1ABC” working will be deleted.

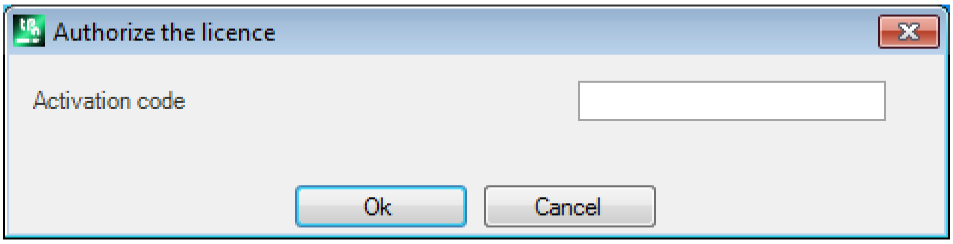
In both situations as above the *Final customer*, even if he identifies himself at a manufacturer level (not as “myname”):

- ✓ cannot open the encrypted macro-programs,
- ✓ cannot see the custom database "unencrypted" in the encrypted workings,
- ✓ cannot change the custom database (in none of its parts),
- ✓ if the Manufacturer, for example, has excluded the functioning of the encrypted group of workings (for example the workings matching the*Encryption code*=30), he cannot change this configuration in TpaCAD.
- ✓ if the machine Manufacturer has chosen to lock the configuration of TpaCAD, anything of the configuration will not be modified.

## 6.6 Distributing the code activation

Finally, let us consider again the *Activation code*, that until now has not been used.
Until now, to activate the functioning of the "signed" and "encrypted" custom database you only need to have a complete installation: it is not be possible to unencrypt the macros and the encrypted workings, but all runs without problems.
What can a manufacturer do to increase the protection of his own customizations, say from the point of view of the distribution? He needs a kind of enabling key: just the *Activation code*.
We might imagine to have signed the custom database with user ("myname" and password) and activation code "xxx123":

let us now to come back to the *Final customer*: now, starting TpaCAD in the computer requires to complete its own enabling of the licence by means of the activation Code.



The activation of the signature is carried out in the register of the system: this can require that the activation should be carried out from the account of the system Administrator.
Needless to say, that if the installation is authorized, the customer knows the activation code to be set and the manufacturer must inform the customer about this code. Selecting OK:

- if the set code is not correct: the window remains open and the icon ✖ appears alongside the setting field;

- if the set code is verified as correct, the application is completed. In this case a message informs that the application will be closed and that a restart to complete and check the result of the activation will be required. If, when restarting TpaCAD, the window requiring the authorization appears again, an error occurred at the registry access.

The activation of the signature is related to the single installation, so it must repeated if performed on different computers.
Uninstalling TpaCAD does not delete either the signature file nor the register activation option. Anyway, we suggest that you should keep a saved copy of the signature file, in case you need to perform a restore.

If the licence activation is not properly performed or if the signature file is not available and verified as valid, the database of the custom workings is loaded anyway, but it will not working for the encrypted parts.

# 7 Installing the TpaCAD software package

TpaCAD software package must be entirely installed on each computer, also where only a part of the application programs or components of the same package can work. If this package is expected to work on two office computers and on one machine computer, three installations are required. An entire installation concerns:

- ✓ the package base installation, that must be always performed entirely (supplied by TPA);
- ✓ optionally, one or more plug-ins configurations (supplied by TPA);
- ✓ optionally, the custom configuration as configured by the machine manufacturer.

The package operation working must be the same on all the installations: base or professional TpaCAD. For each installation we suggest, if not requested, starting the TpaCAD program, so as to complete or to validate the package configuration:

- ✓ custom database signature activation, if requested;
- ✓ acquisition and validation of the configuration assignments.

## 7.1 The external resources

| **Resource** | **Meaning** |
|---|---|
| Bin\TPACAD.EXE<br>Bin\TPAWORKS.EXE | Application programs of Cad-Cam and working database management. |
| Bin\TPA.INI | Initialization file. The file is stored in the folder of the application program. |
| Bin\TPACAD.CHM<br>Bin\ TPACAD_ITA.CHM<br>Bin\ TPACAD_ENG.CHM<br>Bin\ TPACAD_FRA.CHM<br>Bin\ TPACAD_DEU.CHM<br>Bin\ TPACAD_ESP.CHM<br>... | Help files of TpaCAD application program.<br>The files are stored in the folder of the application program |
| Bin\TPACADAD.CHM<br>Bin\TPACADAD_ITA.CHM<br>Bin\TPACADAD_ENG.CHM<br>... | Help files of TpaCAD application program for the advanced assignments (Configuration, Custom functions).<br>The files are stored in the folder of the application program. |
| Bin\TPACADMD.CHM<br>Bin\ TPACADMD_ITA.CHM<br>Bin\ TPACADMD_ENG.CHM<br>... | Help files of TpaCAD application program, concerning the following features: modelling, curved faces.<br>The files are recorded in the folder of the application program. |
| Bin\TPAWORKS.CHM<br>Bin\ TPAWORKS_ITA.CHM<br>Bin\ TPAWORKS_ENG. CHM | Files di help dell'applicativo TPAWORKS.<br>The files are recorded in the folder of the application program. |
| Bin\DXFTOTPA.CHM<br>Bin\ DXFTOTPA _ITA.CHM<br>Bin\ DXFTOTPA _ENG.CHM<br>Bin\ DXFTOTPA _FRA.CHM<br>Bin\ DXFTOTPA _DEU.CHM<br>Bin\ DXFTOTPA _ESP.CHM<br>... | Help files concerning the configuration of the importer from DXF format.<br>The files are recorded in the folder of the application program. |
| Lng\TPACAD_ITA.DLL,….<br>Lng\TPAWORKS_ITA.DLL,…<br>Lng\CADPIECE_ITA.DLL,...<br>LNG\CADMIX_ITA.DLL….<br>LNG\DXFTOTPA_ITA.DLL…. | Message file of the TpaCAD application program<br>Message file of the TpaWorks application program<br>File of the messages of the components added to the control of the application programs.<br>Filed of the messages to configure the importer from DXF format. |

**Base configuration resources**

They are files stored from the tpacadcfg folder. They define the base configuration of the package. Normally, the files are updated version after version and do not need to be changed by the customer. Updating the version, any changes made after the installation could be get lost.

| **Resource** | **Meaning** |
|---|---|
| TpaCadcfg\DBWORKS.WCAD | Database of base workings |
| TpaCadcfg\DBWORKS.XMLng<br>TpaCadcfg\DBWORKS.XMLna | Messages of base workings |
| TpaCadcfg\PROTOMX.WCAD | File of the matrix piece prototypes for the working database |
| TpaCadcfg\CADPPARAM.INI | File of contextual help for parametric programming |
| Tpacadcfg\DBBMP\Wcop.PNG (JPG, BMP) | Individual images for the graphic selection of the base workings |
| Tpacadcfg\DBBMPHLP\Wcop.PNG (JPG, BMP) | Help individual images for the base workings |
| Tpacadcfg\DBHLP\Wcop.CHM<br>Tpacadcfg\DBHLP\Wcop_ITA.CHM<br>Tpacadcfg\DBHLP\Wcop_ENG.CHM<br>Tpacadcfg\DBHLP\Wcop_FRA.CHM<br>Tpacadcfg\DBHLP\Wcop_DEU.CHM<br>Tpacadcfg\DBHLP\Wcop_ESP.CHM<br>… | Help files for the base workings |
| TpaCadcfg\CADFUN.DEF | File of the custom functions |
| Tpacadcfg\FUNHLP\FUN*.CHM<br>Tpacadcfg\FUNHLP\FUN*_ITA.CHM<br>Tpacadcfg\FUNHLP\FUN*_ENG.CHM<br>Tpacadcfg\FUNHLP\FUN*_FRA.CHM<br>Tpacadcfg\FUNHLP\FUN*_DEU.CHM<br>Tpacadcfg\FUNHLP\FUN*_ESP.CHM<br>… | Help files for the custom functions * indicates the name of the custom function. Thus:<br>• FUNabc_…. for the custom with custom name "abc" |
| **TpaCadcfg\MCR\*.tmcr** | Base macro files |

**Resources of custom configuration (PlugIns)**
They are files stored from the *Cadcfg folder*. They define the custom configuration of the package. Normally, the files are updated between a version and another and they do not need to be changed by the customer. Updating the version, any changes made after the installation could be lost.
PlugIn installation can be supplied separately or with the base installation and it lies within the exclusive competence of TPA.

| **Resource** | **Meaning** |
|---|---|
| TpaCadcfg\PLUGINS\DBW*.WCAD | PlugIn Working Database: one or more then one |
| TpaCadcfg\PLUGINS\DBW*.XMLNG | File of plugin messages (one for each plugin) |
| TpaCadcfg\PLUGINS\DBBMP\Wcop.PNG (JPG, BMP) | Single images for the graphical selection of the workings (of PlugIns) |
| TpaCadcfg\PLUGINS\DBBMPHLP\Wcop.PNG (JPG, BMP) | Help single images for the workings (of PlugIns) |
| TpaCadcfg\PLUGINS\DBHLP\Wcop.CHM<br>TpaCadcfg\PLUGINS\DBHLP\Wcop_ITA.CHM<br>TpaCadcfg\PLUGINS\DBHLP\Wcop_ENG.CHM<br>TpaCadcfg\PLUGINS\DBHLP\Wcop_FRA.CHM<br>TpaCadcfg\PLUGINS\DBHLP\Wcop_DEU.CHM<br>TpaCadcfg\PLUGINS\DBHLP\Wcop_ESP.CHM<br>… | Help files for the workings (of PlugIns) |

| | |
|---|---|
| TpaCadcfg\PLUGINS\MCR\*.tmcr | PlugIn macro files |

**Custom configuration resources**
They are files stored from the TpaCadcfg\CUSTOM folder. They define the custom configuration of the package.
The base installation of the package writes some files only, but only if a custom configuration was not already installed: the cases are marked by the [Note (1)].
Provision and maintenance of the TpaCadcfg\CUSTOM folder are borne by the machine manufacturer.

| **Resource** | **Meaning** |
|---|---|
| TpaCadcfg\CUSTOM\TPACAD.INI<br>TpaCadcfg\CUSTOM\TPACAD.XML | File of initializations related to customization and configuration of the application program. The file extension can be INI or XML according to the TpaCAD version and to the assigned configuration. [Note (1)] |
| TpaCadcfg\CUSTOM\CADAUX.XMLng<br>TpaCadcfg\CUSTOM\CADAUX.XMLna | File of custom messages for the application program [Note (1)] |
| TpaCadcfg\CUSTOM\CADSECTOMX.DEF<br>TpaCadcfg\CUSTOM\CADSINTAX.DEF | File of the matrix piece prototypes for the custom sections<br>Syntax control file and format for the custom sections |
| TpaCadcfg\CUSTOM\PIECE.TCN | Prototype file by default for the creation of a new piece [Note (1)] |
| TpaCadcfg\CUSTOM\LOGOCUST.PNG (JPG, BMP) | Custom image in the menu bar |
| TpaCadcfg\CUSTOM\*.INI | File of initializations for the modules of import/export format [Note (1)] |
| TpaCadcfg\CUSTOM\DBBTN\*.PNG (JPG, BMP) | Search folder of the files for the composition of the toolbar of graphic selection of the workings |
| TpaCadcfg\CUSTOM\CUSTOM.INF | Custom folder version file |
| TpaCadcfg\CUSTOM\DBPATTERN\*.PNG (JPG, BMP) | Search folder of the files for the assignment of filling patterns for the graphic representation of the panel. |
| TpaCadcfg\CUSTOM\DBFONTS\*.FCAD | Search folder of the files assigning the custom fonts [Note (1)] |
| TpaCadcfg\CUSTOM\DBWCLI.WCAD | Database of client workings |
| TpaCadcfg\CUSTOM\DBWCUST.WCAD | Database of custom workings |
| TpaCadcfg\CUSTOM\ DBWCUST.XMLng<br>TpaCadcfg\CUSTOM\ DBWCUST.XMLna | Messages of custom workings |
| TpaCadcfg\CUSTOM\DBWCUST.LIC | License file of Database custom workings |
| | |
| TpaCadcfg\CUSTOM\MCR\*.tmcr | Custom macro files |
| TpaCadcfg\CUSTOM\DBBMP\Wcop.PNG (JPG, BMP) | Individual images for the graphic selection of the custom of client workings |
| Tpacadcfg\CUSTOM\DBBMPHLP\Wcop.PNG (JPG, BMP) | Help individual images for custom or client workings. |
| Tpacadcfg\CUSTOM\DBHLP \Wcop.CHM<br>Tpacadcfg\CUSTOM \DBHLP\Wcop_ITA.CHM<br>Tpacadcfg\CUSTOM \DBHLP\Wcop_ENG.CHM<br>cadcfg\CUSTOM \DBHLP\Wcop_FRA.CHM<br>cadcfg\CUSTOM \DBHLP\Wcop_DEU CHM<br>cadcfg\CUSTOM \DBHLP\Wcop_ESP CHM<br>… | Help files for custom or client workings. |
| TpaCadcfg\CUSTOM\CADFUNCUST.DEF | File of customized custom workings |

| | |
|---|---|
| TpaCadcfg\CUSTOM\CADGLOBAL.DEF | File of the global variables |
| Tpacadcfg\ CUSTOM \FUNHLP\FUN*.CHM<br>Tpacadcfg\ CUSTOM \FUNHLP\FUN*_ITA.CHM<br>Tpacadcfg\ CUSTOM \FUNHLP\FUN*_ENG.CHM<br>Tpacadcfg\ CUSTOM \FUNHLP\FUN*_FRA.CHM<br>Tpacadcfg\ CUSTOM \FUNHLP\FUN*_DEU.CHM<br>Tpacadcfg\ CUSTOM \FUNHLP\FUN*_ESP.CHM<br>… | Help files for the custom functions: * indicates the name of the custom function. Thus:<br>• FUNabc_…. for the function with custom name "abc" |

## 7.2 The CADAUX.XMLNG file

This is a message file containing all customized messages for TpaCAD.
More specifically:

| ID | MESSAGE EXAMPLE | USE |
|---|---|---|
| 1 | Drillings | [1 – 50]: Help tests for the main buttons of the working graphic palette |
| 2 | Setup | |
| .. | | |
| 50 | | |
| 51 | x Offset | [51-66]: Tests for the description of the "o" variables : from o0 to o15 |
| .. | | |
| 66 | | |
| 71 | x Offset for piece positioning | [71-86]: Assistance tests in assignment of the "o" variables: from o0 to o15 |
| .. | | |
| 86 | | |
| 101 | Special geometries | [101-110]: Title of the pages for Special settings |
| 102 | Technologies | |
| .. | | |
| 110 | | |
| 111 | Clamping devices | [111-120]: Titles of the pages for the Additional settings |
| .. | | |
| 120 | | |
| 121 | Optimization settings | [121-130]: Titles of the pages for the Optimization settings |
| 122 | Sorts | |
| .. | | |
| 130 | | |
| 131 | Section of Constraints | [131-140]: Titles of the pages for the Section of Constraints |
| .. | | |
| 140 | | |
| 141 | | [141-850] : Messages used in individual entries of the sections: Special settings, Additional settings, Optimization settings, Constraints |
| .. | | |
| 850 | | |
| 890 | Special of machine | [890] Custom title of the Special setting section |

| **ID** | **MESSAGE EXAMPLE** | **USE** |
|---|---|---|
| 891 | Special technology<br>settings | [891] Custom title of the additional Setting section |
| 892 | Setting Optimisation | [892] Custom title of the Optimisation Setting section |
| 893 | Clamping devices | [893] Custom title of the Constraint section |
| 900 | O-Field | O-Field custom message |
| 901 | K-Field | K-Field custom message |
| 902 | K1-Field | K1-Custom field |
| 903 | K2-Field | K2-Field custom message |
| 911 | Piece load | [911-926]: Description tests of the “v” variables: from v0 to v15 |
| ..926 | | |
| 931<br>.. 946 | 1= enables the piece load | [921-946]: Assistance tests in assignment of the "v" variables: d3 v0 ad v15 |
| 950 | | [950-999] : Messages for various configurations |
| .. | | |
| 999 | | |
| 1001 | Null x-step | [1001-1500] : Messages for custom errors (errors generated with ERROR, BREAK instructions, the error function can be used in the text of a custom function) |
| 1002 | Null y-step | |
| .. | | |
| 1500 | | |
| 1510 | | [1510-1526] : Tests for the description of the O variable (values from 0 to 16) |
| .. | | |
| 1526 | | |
| 1530 | | [1530-1546] : Tests for the description of the K variable (values from 0 to 16) |
| .. | | |
| 1546 | | |
| 1550 | | [1550-1566] : Tests for the description of the K1 variable (values from 0 to 16) |
| .. | | |
| 1566 | | |
| 1570 | | [1570-1586] : Tests for the description of the K2 variable (values from 0 to 16) |
| .. | | |
| 1586 | | |

## 7.3 Custom font files

Let us examine the format of a custom function definition file. FCAD is the extension file *F* and you find them in the folder *tpacadcfg\CUSTOM\DBFONTS.*

They are the INI files, then they are simple text files with a base structure made of "section" and "property".

| | |
|---|---|
| [TPAFONT] | It is the main section, that define the font general properties. Let us see the properties: |

| | |
|---|---|
| hightY=100<br>tmDescent=61<br>unit=0 | hightY = height of a capitalized letter of the font<br>tmDescent = unit below the base line (how much you can descend below the writing line)<br>unit = unit of measure with which the font has been created (each character after the first one is converted into this unit of measure) |
| .. | |
| 65]<br>$=char 'A'<br>#1=g0x0y0s1<br>#2=x30.1818y77.0431<br>#3=g2x37.2963y84.9152i44.1483j71.5716 | Section describing the 'A' character (decimal value=65):<br>"$" = comment to the section<br>#1….#nn: assign the polylines/s defining the 'A' character<br>"g0" = starts a polyline (on #1=.. is by default)<br>"x..y.."= the final position of the segment (non assigned coordinate propagates from the previous one<br>"s1" = shows that the profile that starts here, has some curved segments (arcs or paths)<br>#2=x.. : linear segment ("g1" is by default)<br>If in the line of the linear segment the "a..", "b.." fields (both or one) appear, a drawn element (L24) is interpreted.<br>The actual interpretation of the segment as an arc is conditioned by:<br>• field "s1" assigned on the corresponding setup<br>• TpaCAD application program can manage the Paths.<br>The assignment of a profile with curved segments excludes the spline application.<br>#3=g2x..: curved segment (x..y..= final point; i..j..= absolute centre)<br>"g2"= clockwise rotation; "g3"= counterclockwise rotation<br>The actual interpretation of the segment as an arc is conditioned by:<br>· field "s1" assigned on the corresponding setup<br>· "i.." field programming or "j..": the other coordinate is calculated automatically;<br>· if both the field of the centre are programmed, the arc must be compensated. |

| | |
|---|---|
| #4=x45.9866y89.3778<br>#5=g2x67.5474y78.976i52.8387j76.0343<br>#6=x83.1729y0.8487<br>#7=x66.6298y35.2584<br>#8=g3x51.2504y43.6432i53.111j28.759<br>#9=x14.428y39.0404 | The assignment of a profile with curved segments excludes the spline application.<br>The horizontal overall dimension of the character (coordinate:x) is determined automatically on the overall dimension of the polyline/s and brought to 0: then, it is not necessary to describe the character from x0.<br>Anyway, it is possible to increase the actual overall dimensions of the character, for example letting an empty space at the beginning and assigning the property: #0=x…<br>For instance: we want to describe a character between x20 and x50 and to add an initial overall dimension of 10. Then, the line “#0=x10” be added.<br>The same line linea “#0=x…” can assign a “y..” coordinate, to sum up to the coordinate of the character: in this way it is possible to position the vertical coordinates above zero and assign here the units below or above the base line:<br>· negative value moves the character (partially or wholly) below the base line;<br>· positive character moves the character (partially or wholly) above the base line. |
| .. | |
| [81]<br>#1=g0x0y0<br>#2=y40<br>#3=x10y50<br>#4=x30<br>#5=y0<br>#6=x0<br>#7=g0x20y10<br>#8=y-10 | Section describing the 'Q' character (decimal value=81):<br>#1….#nn: assign the polyline/s defining the 'Q' character<br>Here we see the two polylines have been assigned: the first one from (#1 to #6); the second one from (#7 to #8); |
| .. | |
| [97]<br>usekey=65 | Section describing the 'a' character (decimal value=97):<br>usekey= directs the description of the character to the section [65] (section describing the 'A' character) |
| [0] | Description section of the characters otherwise not assigned.<br>The section doesn't manage the entries:<br>“#0=…”<br>“usekey=..” |

The valid characters are those recognized for the normal programming of variable or parameter of string type.
A differentiated description in the font style (normal, bold or italics) is not provided.

**Tecnologie e Prodotti per l'Automazione**

Via Carducci, 221
I - 20099 Sesto S. Giovanni (MI)
Tel. +39 023 65 27 550
Fax. +39 022 48 10 08

www.tpaspa.it

info@tpaspa.it

10/5/2017