IsoUs – Ultimate Step
G Code User Manual

www.promax.it

PROMAX Motion & Control

.

1 INTRODUCTION

This manual explains ISO programming and all relative functions. The method of programming can change with the used PC model (keyboard, touch-screen), however main concept remains unchanged.
To manage ISO programs refer to USER INTERFACE manual.

2 ISOUS RULES

2.1 BLOCK

A block is formed by one or more ISO function ended with CR/LF character.
The sequence of more blocks composes a program (defined PartProgram) that it is managed by user interface.
All characters forming the block must be UPPER-CASE.
Isous editor makes an automatic conversion of lower-case characters.

G1X100Y100Z10

2.2 LINE NUMBER OR BLOCK NUMBER

Line number or block number defines block position inside the PROGRAM.
Such position is useful in some features (block restart, retrace, etc.).
Isous ignores line marks Nxx before block, therefore the real line number is the number showed on ISO editor window.

Isous ignores **N10** and **N30** characters, therefore the real line is showed on the left bar. 1 and 2 in this example.

2.3 PROGRAM RUNNING

Program running is made by Isous user interface. When in run, program can be PAUSED or STOPPED.

2.4 AXIS NAME

Isous don't use any particular convention to define axis type (linear or rotative) by name (X,Y, etc.). Definition must be setted only in configuration. Isous manages up to 9 axis with the following names:

X,Y,Z,A,B,C,U,V,W

2.5 HOMING

All axis refer to an HOME POSITION made after switch on. Home position follows an HOMING procedure selected by relative machine parameters.

2.6 WORK ORIGIN

Isous manages up to 256 different WORK ORIGINS. Each origin defines a reference point of zero, all following positions refers to this point. Origins can be set both manually by interface and with dedicated instructions inserted in PartProgram. Furthermore they can be SUSPENDED, RESTORED or DISABLED.

2.7 WORK OFFSET

As well origin, Isous can manage up to 256 WORK OFFSET. An OFFSET works as a new origin manageable only with PartProgram. Also OFFSETS can be SUSPENDED, RESTORED or DISABLED.

2.8 HEADS

Isous manages 256 working HEADS for a single machine.
Each head refers to machine origin and it is automatically pre-setted (do not need to use WORK ORIGIN e WORK OFFSET).
Used head can be recall by PartProgram with Hn instruction. It programs automatically head offset of all axis (refered to machine origin).

2.9 MODAL FUNCTIONS

MODAL functions remains active for all current PartProgram and also the subsequent ones until a (or more) particular function disables it. G0,G1,G2,G3 etc. are modal functions and therefore it doesn't need repeat them until another function exclude it. G1 excludes G0,G2 e G3, G2 excludes G0,G1 e G3 etc.
G1X100Y100
X300Y120Z10
X450
In this example it is used modal G1 function. In block 2 and 3 it isn't necessary to insert G1 because it remains active.

2.10 RECOGNIZED CODES

Almost all recognized codes are highlighted with a colour. However, in the few cases when it doesn't happen, it doesn't mean instruction isn't recognized.
When Isous editor doesn't recognize an instruction it is underlined with RED and it shows error window.

2.11 NUMERIC VALUES SETTING

Isous works with two numeric values types:

INTEGER
FLOATING

Integer values are used for all defined quantities (Loop numbers, etc.). When it is insert a floating value instead of integer one, Isous generate an error warning the user.

Floating values are used for all REAL quantities (axis position, radius, etc.).
Isous uses as decimal separator character "." (DOT).

2.12 PROGRAM REMARKS

Isous don't use CNC classic convention for program remarks.
Remarks starts with **"//"** characters and terminates with end line. Isous editor highlights it with GREEN colour.
Remarks can be insert also on the right of block.
Remarks are ignored by PartProgram and all its content are discarded.

```
// This is a remark
G1X100Y100      // This is a remark too
```

3 ISO NS INSTRUCTIONS

3.1 RECOGNIZED G CODES

Code G	Description	TASK1	TASK2
G0	RAPID positioning	NO	NO
G0.1	RAPID positioning single Axis with private Acceleration	NO	NO
G0.2	RAPID positioning for “TRANSPORTED AXIS”	NO	NO
G1	Linear Interpolation at F programmed speed	NO	NO
G1.1	G1-G2-G3 Suspension and set G0	NO	NO
G1.2	G1 Resume	NO	NO
G2	CW circular interpolation	NO	NO
G3	CCW circular interpolation	NO	NO
G4	Timed pause	YES	YES
G4.1	Time Add on Calc Time	NO	NO
G10	Enable external axis OVERRIDE ASSI on potentiometer	YES	YES
G11	Disable external axis OVERRIDE ASSI on potentiometer	YES	YES
G17	Set work plane on X-Y	NO	NO
G18	Set work plane on Z-X	NO	NO
G18.1	Set work plane on Z-X but not in PREVIEW	NO	NO
G19	Set work plane on Y-Z	NO	NO
G19.1	Set work plane on Y-Z but not in PREVIEW	NO	NO
G20	Axes in Inch	NO	NO
G21	Axes in millimeters	NO	NO
G22	Change axis of the work plane	NO	NO
G23	Restore axis of the work plane (disable G22)	NO	NO
G24	Enable horizontal mirror	NO	NO
G25	Disable horizontal mirror G24	NO	NO
G26	Change a couple of axis	NO	NO
G27	Suspend G26	NO	NO
G28	Resume G26	NO	NO
G30	Enable automatic insert of fillet on edges	NO	NO
G31	Suspend G30	NO	NO
G32	Resume G30	NO	NO
G33	Enable automatic insert of bevel on edges	NO	NO
G34	Suspend G33	NO	NO
G35	Resume G33	NO	NO
G36	Gestione assi rotativi	NO	NO
G40	Disable TOOL RADIUS compensation	NO	NO

G41	Enable TOOL RADIUS compensation LEFT	NO	NO
G42	Enable TOOL RADIUS compensation RIGHT	NO	NO
G43	Enable TOOL LENGTH compensation	NO	NO
G44	Disable TOOL LENGTH compensation	NO	NO
G44.1 G44.2	Suspend G43 Resume G43	NO	NO
G45	Enable TOOL LENGTH compensation, parameter from TOOL TABLE	NO	NO
G46	Disable G45	NO	NO
G47	Set mode of start/stop TOOL compensation G41 G42	NO	NO
G48	Define Depth Axis	NO	NO
G49	MILD MODE – Edge Smoothing	NO	NO
G50	Working plane rotation	NO	NO
G51	Suspend plane rotation G50	NO	NO
G52	Restore plane rotation G50	NO	NO
G53	As G98 (for compatibility to other cn)	NO	NO
G54 G54.n	Work origin from memory, AXIS X (or axis selected by following parameters XYZ etc	NO	NO
G55	Work origin from memory, AXIS Y	NO	NO
G56	Work origin from memory, AXIS Z	NO	NO
G57	Work origin from memory, AXIS XY	NO	NO
G58	Work origin from memory, AXIS YZ	NO	NO
G59	Work origin from memory, AXIS XZ	NO	NO
G60	Enable FAST interpolation without stop on segments	NO	NO
G61	Enable interpolation with stop on segments	NO	NO
G62	Wait stop axis	NO	NO
G63	G1 Enable PX_MOVETO (3D interpolation outside selected plane) without stop	NO	NO
G64	G1 Enable interpolation on selected plane PX_LINETO (2D plane interpolation	NO	NO
G65	Enable 3D interpolation PX_MOVETO with calculated stop on edge by parameters	NO	NO
G66	AFC – Adaptive Feed Control	NO	NO
G66 X-100	NEW AFC – Adaptive Feed Control	NO	NO
G67	PX_MOVETO for movement outside plane and PX_LINETO for inside ones	NO	NO
G68	Always using of PX_LINETO in G1 “TRANSPORTED AXIS”	NO	NO
G69	LHK – Depth of buffer Look Ahead on virtual CN	NO	NO
G70	Set work plane on a generic couple of axis	NO	NO
G71 G71.1 G71.2	Start axis home searching Enable Axis Disable Axis	NO	NO
G72	Enable N.U.R.B.S (Not Uniform Rational BSpline) filter	NO	NO
G73	Enable Noise filter	NO	NO

G74	Enable RLS (Remove Len Segment) filter	NO	NO
G75	Use G64 for movement inside work plane and G65 for movement outside work plane	NO	NO
G80 (G1080)	Forced pause by code	YES	YES
G81 (G1081)	Management secondary axes LIMITS	YES	YES
G82 (G1082)	Work ORIGIN at current position with sensor offset	NO	NO
G83 (G1083)	Update CPU1 counter	NO	NO
G84 (G1084)	Preset CPU axis	NO	NO
G85	Work OFFSET at current position with sensor offset	NO	NO
G86	Hardware preset axis on module 360 degrees	NO	NO
G87	Suspend head offset selected with Hn	NO	NO
G88	Resume head offset selected with Hn	NO	NO
G89	HardWare Preset of axis position	NO	NO
G90	Values with ABSOLUTE position	NO	NO
G91 G91.1 G91.2	Values with INCREMENTAL position Force the values in ABSOLUTE (G90) and store the current state (G90-G91) Resume the state stored with G91.1 (G90 if was G90 or G91 if was G91)	NO	NO
G92	Work ORIGIN at current position	NO	NO
G93	Work OFFSET at selected position	NO	NO
G94 G94.n	Work ORIGIN at selected position Save work origins	NO	NO
G95	Work OFFSET at current position	NO	NO
G96	Suspend work offset G93-G95	NO	NO
G97	Resume work offset G92-G94	NO	NO
G98	Suspend work origin G92-G94	NO	NO
G99	Resume work origin G92-G94	NO	NO
G100	Syncronous comand for VIRTUAL axis	NO	NO
G101	AXIS STOP comand	YES	YES
G102	Start sensor acquisition	NO	NO
G102.1	Acquisition from PxVision System	NO	NO
G102.2	Start sensor acquisition without Alarm	NO	NO
G103	Set RTCP parameters	NO	NO
G104 G104.1	Enable RTCP	NO	NO
G105	Suspend RTCP	NO	NO
G106	Smoothing Filter	NO	NO
G107	Management Interrupt Macro	NO	NO
G108	Management Special Axes	NO	NO
G120	Enable vertical mirror	NO	NO
G121	Disable vertical mirror G20	NO	NO

G940	MOVE axes excluding WORK ORIGIN only in the actual block	NO	NO
G1028 (G28)	Return to Home	NO	NO
G1050	Disable Scaling	NO	NO
G1051	Enable Scaling	NO	NO
G1080 (G80)	Drilling	NO	NO
G1081 (G81)	Drilling	NO	NO
G1082 (G82)	Drilling	NO	NO
G1083 (G83)	DRILLING	NO	NO
G1084 (G84)	TAPPING	NO	NO

4 OTHERS RECOGNIZED CODES

Code	Description	TASK1	TASK2
X	Axis X position	NO	NO
Y	Axis Y position	NO	NO
Z	Axis Z position	NO	NO
A	Axis A position	NO	NO
B	Axis B position	NO	NO
C	Axis C position	NO	NO
U	Axis U position	NO	NO
V	Axis V position	NO	NO
W	Axis W position	NO	NO
QX	Channel 0 position	NO	NO
QY	Channel 1 position	NO	NO
QZ	Channel 2 position	NO	NO
QA	Channel 3 position	NO	NO
QB	Channel 4 position	NO	NO
QC	Channel 5 position	NO	NO
QU	Channel 6 position	NO	NO
QV	Channel 7 position	NO	NO
QW	Channel 8 position	NO	NO
DX	Axis X relative position	NO	NO
DY	Axis Y relative position	NO	NO
DZ	Axis Z relative position	NO	NO
DA	Axis A relative position	NO	NO
DB	Axis B relative position	NO	NO
DC	Axis C relative position	NO	NO
DU	Axis U relative position	NO	NO

DV	Axis W relative position	NO	NO
DW	Axis W relative position	NO	NO
OX	Set Origin number for axis X	NO	NO
OY	Set Origin number for axis Y	NO	NO
OZ	Set Origin number for axis Z	NO	NO
OA	Set Origin number for axis A	NO	NO
OB	Set Origin number for axis B	NO	NO
OC	Set Origin number for axis C	NO	NO
OU	Set Origin number for axis U	NO	NO
OV	Set Origin number for axis V	NO	NO
OW	Set Origin number for axis W	NO	NO
DOX	Origin Ofset fo axis X	NO	NO
DOY	Origin Ofset fo axis Y	NO	NO
DOZ	Origin Ofset fo axis Z	NO	NO
DOA	Origin Ofset fo axis A	NO	NO
DOB	Origin Ofset fo axis B	NO	NO
DOC	Origin Ofset fo axis C	NO	NO
DOU	Origin Ofset fo axis U	NO	NO
DOV	Origin Ofset fo axis V	NO	NO
DOW	Origin Ofset fo axis W	NO	NO
F	Axis FEED	NO	NO
S	Spinale speed	YES	YES
N	Line number (optional)	NO	NO
D	Tool diameter	NO	NO
M	M function	YES	YES
HM	HM function	YES	YES
H	Select head	NO	NO
T	Select tool	NO	NO
I	CENTER X coordinate for G2 G3	NO	NO
J	CENTER Y coordinate for G2 G3	NO	NO
K	CENTER Z coordinate for G2 G3 (optional)	NO	NO
R	Radius for G2 G3	NO	NO
USER_ZERO	Index of WORK ORIGIN LIST	NO	NO
USER_OFFSET	Index of WORK OFFSET LIST	NO	NO
STOP_MODE	Defines the STOP Button function mode	NO	NO
PAUSE_MODE	Defines the Pause function mode	NO	NO
FILTER_MODE	Defines the Filter function mode	NO	NO

4.1 PROGRAM FLOW INSTRUCTIONS

Code	Description	TASK1	TASK2
IF	Start IF cycle	YES	YES
ELSE	Relative to IF cycle	YES	YES
END_IF	End IF cycle	YES	YES
SELECT-CASE	Select – Case cycle	YES	YES
LOOP	Start LOOP cycle	YES	YES
END_LOOP	End LOOP cycle	YES	YES
FOR	Start For Cycle	YES	YES
NEXT	End For Cycle	YES	YES
BREAK	Break FOR or WHILE	YES	YES
CONTINUE	Continue FOR or WHILE	YES	YES
WHILE	Start WHILE Cycle	YES	YES
END_WHILE	End WHILE	YES	YES
GOTO	Jump to LABEL or line number	YES	YES
GOSUB	Call of Soubrutine	YES	YES
RETURN	Retrun from Soubrutine	YES	YES
@	LABEL definition	YES	YES
//	Start remarks	YES	YES
END_PROGRAM	End of program	YES	YES
WAIT_INPUT	Wait of digital input with Time Out	YES	YES
TWAIT_INPUT	Wait of digital input with Time Out and with state time	YES	YES
ERROR	Force stop program with error	YES	YES

4.2 MULTI TASK INSTRUCTIONS

Code	Description	TASK1	TASK2
TASK.RUN	Run TASK	YES	YES
TASK.STOP	Stop TASK	YES	YES
TASK.PAUSE	Pause TASK	YES	YES
TASK.READVAR	Read Variable from TASK	YES	YES
TASK.WRITEVAR	Write variable to TASK	YES	YES
TASK.STATUS	Read status TASK	YES	YES
TASK.LOADCMD	Load CMD in the TASK	YES	YES
TASK.PRIORITY	Set priority TASK	YES	YES
USTASK	Start Gcode Section for TASK1	NO	NO
ENDUSTASK	End Gcode section for TASK1	NO	NO

4.3 GENERIC INSTRUCTIONS

Code	Description	TASK1	TASK2
SDO_DL	Sdo download (can open)	YES	YES
SDO_UL	Sdo upload (can open)	YES	YES
GET	Read parameter of HM function	YES	YES
READ_PARMAC	Read machine parameter	YES	YES
WRITE_PARMAC	Write machine parameter	YES	YES
LOAD_VAR	Load a file of variable	YES	YES
GET_VAR	Read a variable from loaded file	YES	YES
WRITE_VAR	Wrtie a variable in memory list	YES	YES
SAVE_VAR	Save a file of variable on harddisk	YES	YES
DIM_VAR	Set dimension of memory list	YES	YES
FILE_EXISTS	Check if file exists	YES	YES
REMOVE_VAR	Remove a value from current list	YES	YES
CLEAR_VAR	Remove ALL values from current list	YES	YES
OPT	Compiler option	YES	YES
IMPORT	Import external file	NO	NO
END_IMPORT	End of Import program	NO	NO
PA(n,par)	Set ABSOLUTE target positioner	YES	YES
PD(n,par)	Set REALTIVE target positioner	YES	YES
PF(n)	Set FEED positioner	YES	YES
PS(n)	STOP positioner	YES	YES
_PM(n,par)	STATUS positioner	YES	YES
RESUME_T	Resume the PartProgram from last instruction Tn used	NO	NO
DIM	ARRAY management	YES	YES
DEBUG_INFO	Write informations in Log file	YES	YES
SET_TABPAR	Set a Parameters Table	YES	YES
SAVE_T	Save the tools table in the Isous cfg	YES	YES
IF_RUN	Check NORMAL Run Type ($[X7]=0)	YES	YES
IF_NOTRUN	Check SIMULATED Run Type ($[X7]<>0)	YES	YES
USEFORM	Open a Form compiled by UsFormManager	YES	YES

4.4 MATH AND LOGICAL OPERATORS

Code	Description	TASK1	TASK2
+	Addiction	YES	YES
-	Subtraction	YES	YES
*	Multiplication	YES	YES
/	Division	YES	YES
(	Open bracket	YES	YES
)	Cosed bracket	YES	YES
[	Start expresion for axis count ex: G1X[$var+$var1]	YES	YES
]	End expresion for axis count ex: G1X[$var+$var1]	YES	YES
^	POW	YES	YES
%	XOR	YES	YES
>	Greater	YES	YES
<	Less	YES	YES
>=	Greater or equal	YES	YES
<=	Less or equal	YES	YES
<>	Not equal	YES	YES
=	Equal	YES	YES
||	Logic Or	YES	YES
&&	Logic And	YES	YES
|	Or Bit	YES	YES
&	And Bit	YES	YES
!	Negate	YES	YES
~	Not Bit	YES	YES
>>	Shift bit right	YES	YES
<<	Shift bit left	YES	YES
$VAR++ :xxx++	$VAR=$VAR+1 :1000=:1000+1	SI	SI
$VAR— :xxx--	$VAR=$VAR-1 :1000=:1000-1	SI	SI
$VAR+= :xxx+=	$VAR=$VAR+exp :1000=:1000+exp	SI	SI
$VAR-= :xxx-=	$VAR=$VAR-exp :1000=:1000-exp	SI	SI
$VAR*= :xxx*=	$VAR=$VAR*exp :1000=:1000*exp	SI	SI
$VAR/= :xxx/=	$VAR=$VAR/exp :1000=:1000/exp	SI	SI

4.5 MATH FUNCTIONS

Code	Description	TASK1	TASK2
SIN	Sinus	YES	YES
COS	Cosinus	YES	YES
LOG	Logarithm	YES	YES
EXP	Exponetial	YES	YES
SQR	Square root	YES	YES
TAN	Tangent	YES	YES
ATAN	Arctangent	YES	YES
ASIN	Arcsinus	YES	YES
ACOS	Arccos	YES	YES
INT	Integer part of float rounded	YES	YES
FIX	Integer part of float whitout rounded	YES	YES
ABS	Absolute value	YES	YES
DRG	Set degrees for COS,SIN,TAN,ACOS,ASIN,ATAN	YES	YES
RAD	Set radiants for COS,SIN,TAN,ACOS,ASIN,ATAN	YES	YES

4.6 VARIABLES and CONSTANTS

Code	Description	TASK1	TASK2
NUMBER	Numeric constant ex: 432.12	YES	YES
$VARNAME	Generic double value ex: $VAR1	YES	YES
$STRUCT.VAR	Data Structures	YES	YES
:Addr	Variable for Address	YES	YES
:$VAR	Variable for pointer	YES	YES
$[Qn]	Demand axis position ex: $[Q0] - $[Q100]	YES	YES
$[Rn]	Actual axis position ex: $[R0] - $[R100]	YES	YES
$[In]	Digital input ex: $[I10]	YES	YES
$[On]	Digital output ex: $[O10]	YES	YES
$[Tn]	TIMER ex: $[T3]	YES	YES
$[Un]	Tool parameter ex: $[U12]	YES	YES
$[Cn]	Axis counter ex: $[C0]	YES	YES
$[Hn]	Head parameter ex: $[H1]	YES	YES
$[Xn]	Special parameters Variables ex: $[X1]	YES	YES
$[Yn]	Work origin position ex: $[Y0]	YES	YES
$[Wn]	Work offset position ex: $[W0]	YES	YES
$[Kn]	User generic on CN	YES	YES
$[En]	Positive Software limit axis set	YES	YES
$[Pn]	Preview Parameters	YES	YES
$[Sn]	Negative Software limit axis set	YES	YES
$[A0]	Write Analog Out for SPINDLE	YES	YES
$[Jn]	Macro and Gcode parameters Management	YES	YES

4.7 PREDEFINED VARIABLES

Code	Description	TASK1	TASK2
$_PARM_1	Parameter 1 for M function	YES	YES
$_PARM_2	Parameter 2 for M function	YES	YES
$_PARM_3	Parameter 3 for M function	YES	YES
$_PARM_4	Parameter 4 for M function	YES	YES
$_PARM_5	Parameter 5 for M function	YES	YES

4.8 UsForms INSTRUCTIONS

Code	Description	TASK1	TASK2
LIB.MESSAGE	Show Message Box	NO	NO
LIB.SHOWFORM	Show NsForm	NO	NO
LIB.CLOSEFORM	Close NsForm	NO	NO
LIB.FORMPROP	Set Property NsForm	NO	NO
LIB.FORMTEXT	Set Caption NsForm	NO	NO
LIB.ADDLABEL	Add a NsLabel to NsForm	NO	NO
LIB.LABELPROP	Set NsLabel Property	NO	NO
LIB.LABELTEXT	Set Text NsLabel	NO	NO
LIB.LABELPRINT	Print Isous Variable to NsLabel	NO	NO
LIB.LABELF	Set Format for print from TIMER	NO	NO
LIB.ADDBUTTON	Add a NsButton to NsForm	NO	NO
LIB.BUTTONPROP	Set NsButton Property	NO	NO
LIB.BUTTONTEXT	Set Text NsButton	NO	NO
LIB.BUTTONPRINT	Print Isous Variable to NsButton	NO	NO
LIB.BUTTONF	Set Format for print from TIMER	NO	NO
LIB.ADDINPUT	Add a NsInput Object to NsForm	NO	NO
LIB.INPUTPROP	Set NsInput Property	NO	NO
LIB.INPUTSETVALUE	Set NsInput init value	NO	NO
LIB.ADDITEXT	Add ITEXT Object to NsForm	NO	NO
LIB.ITEXTPROP	Set ITEXT Property	NO	NO
LIB.ITEXTSETVALUE	Set ITEXT init value	NO	NO
LIB.ADDCHECK	Add CHECK Object to NsForm	NO	NO
LIB.CHECKPROP	Set CHECK Property	NO	NO
LIB.CHECKSETVALUE	Set CHECK init value	NO	NO
LIB.CHECKTEXT	Set caption of CHECK	NO	NO
LIB.ADDCOMBO	Add COMBO Object to NsForm	NO	NO
LIB.COMBOPROP	Set COMBO Property	NO	NO
LIB.COMBOSETVALUE	Set COMBO init value	NO	NO
LIB.COMBOITEM	Add an ITEM to COMBO	NO	NO
LIB.ADDSLIDER	Add SLIDER Object to NsForm	NO	NO
LIB.SLIDERPROP	Set SLIDER Property	NO	NO
LIB.SLIDERSETVALUE	Set SLIDER init value	NO	NO
LIB.GETVAR	Read a Form Variable	NO	NO
LIB.SETVAR	Write a Form Variable	NO	NO
LIB.DEBUG	Write a Debug Gcode informations	NO	NO

4.9 EXTENDED INSTRUCTIONS EXD

Code	Description	TASK1	TASK2
EXD.STL_LOAD	Load STL file in Preview	YES	YES
EXD.READ_TOOLPAR	Read Tool Table Parameter	YES	YES
EXD.READ_HEADPAR	Read Head Table Parameter	YES	YES
EXD.WRITE_TOOLPAR	Write Tool Table Parameter	YES	YES
EXD.WRITE_HEADPAR	Write Head Table Parameter	YES	YES
EXD.LOAD_LAST	Load the Last Tool Used	YES	YES
EXD.SAVE_LAST	Save the current use as Last Tool used	YES	YES
EXD.RESET_LAST	Reset the Last Tool used file	YES	YES
EXD.SET_OUT	Set/Reset digital output by address - **$[On]**	YES	YES
EXD.READ_OUT	Read digital output by address - **$[On]**	YES	YES
EXD.READ_INP	Read digital input by address - **$[In]**	YES	YES
EXD.WRITE_USER	Write User Generic by address - **$[Kn]**	YES	YES
EXD.READ_USER	Read User Generic by address - **$[Kn]**	YES	YES
EXD.READ_DEMAND	Read Axis Demand Position by address - **$[Qn]**	YES	YES
EXD.READ_REAL	Read Axis Real Position by address - **$[Rn]**	YES	YES
EXD.MASK	Set Digital Output MASK	YES	YES
EXD.WRITE_BIT	Set/Reset Bit of Variable	YES	YES
EXD.READ_BIT	Read Bit of Variable	YES	YES
EXD.PXV_ALL_DETECTORS	Read all detectors of PXV System	YES	YES
EXD.PXV_SINGLE_DETECTOR	Read a single detector of PXV System	YES	YES
EXD.PXV_READ_PROBE	Read a PROBE of PXV System	YES	YES
EXD.PXV_RESET_PROBE	Reset a PROBE of PXV System	YES	YES
EXD.PXV_SET_DETECTOR	Set Detector Type of PXV System	YES	YES
EXD.PXV_SAVE_IMAGE	Save the current image acquired in PXV	YES	YES
EXD.PXV_GET_IMAGE	Capture an Image from PXV	YES	YES
EXD.PXV_SET_JOB	Set PXV Job configuration File	YES	YES
EXD.RUN_SCRIPT	Execute a C# SCRIPT	YES	YES
EXD.OPEN_DLL	Open UsDLL .NET	YES	YES
EXD.CALL_DLL	Call UsDLL .NET	YES	YES
EXD.CLOSE_DLL	Close UsDLL .NET	YES	YES
EXD.SYNK_DLL	Synk UsDLL .NET	YES	YES
EXD.STL_SETVIS	Show or Hide STL by Name	YES	YES
EXD.STL_READVIS	Read the STL Visibility by Name	YES	YES
EXD.STL_SETVISIDX	Show or Hide STL by Address	YES	YES
EXD.STL_READVISIDX	Read the STL Visibility by Address	YES	YES

4.10 Compilator Switch and Directive

Code	Description	TASK1	TASK2
IFDEF	Compilator Switch IF	YES	YES
ELSEDEF	Compilator Switch ELSE	YES	YES
ENDIFDEF	Compilator Switch ENDIF	YES	YES
NOAXESREADY	Compilator directive Axes not Ready	YES	YES
ONERROR	Compilator directive On Error	YES	YES
ONSTOP	Compilator directive On Stop	YES	YES
ENDON	Compilator directive End On	YES	YES
USET	Compilator directive Use T	YES	YES
USEH	Compilator directive Use H	YES	YES

4.11 INSTRUCTIONS for remote controls

Code	Description	TASK1	TASK2
REMOTE.LOAD	Load Gcode in Remote CN	YES	YES
REMOTE.RUN	Run Gcode in Remote CN	YES	YES
REMOTE.STOP	Stop Gcode in Remote CN	YES	YES
REMOTE.PAUSE	Pausa Gcode in Remote CN	YES	YES
REMOTE.STATUS	Read Status Gcode in Remote CN	YES	YES
REMOTE.MOVE	Read Axes Mov in Remote CN	YES	YES
REMOTE.INFO	Read Info in Remote CN	YES	YES
REMOTE.AXIS	Read Axis value in Remote CN	YES	YES
REMOTE.GROUP	Read Axes Values in Remote CN	YES	YES
REMOTE.READISOVAR REMOTE.READVARNAME	Read Gcode Variable in Remote CN	YES	YES
REMOTE.WRITEISOVAR REMOTE.WRITEVARNAME	Write Gcode Variable in Remote CN	YES	YES
REMOTE.READCNVAR	Read User Generic variable in Remote CN	YES	YES
REMOTE.WRITECNVAR	Write User Generic variable in Remote CN	YES	YES
REMOTE.READINPUT	Read Digital Input in Remote CN	YES	YES
REMOTE.READOUT	Load Gcode in Remote CN	YES	YES
REMOTE.WRITEOUT	Run Gcode in Remote CN	YES	YES

4.12 INSTRUCTIONS for Multi Process Control

Code	Description	TASK1	TASK2
CNC.LOAD	Load Part Program On CN Process	YES	YES
CNC.RUN	Run Part Program On CN Process	YES	YES
CNC.PREVIEW	Preview Part Program On CN Process	YES	YES
CNC.STOP	Stop Part Program On CN Process	YES	YES
CNC.PAUSE	Pause Part Program On CN Process	YES	YES
CNC.STATUS	Read Status On CN Process	YES	YES
CNC.STATUSBIT	Read Status bit On CN Process	YES	YES
CNC.INFO	Read Informations On CN Process	YES	YES
CNC.AXIS	Read Axes Values On CN Process	YES	YES
CNC.GROUP	Read Group Axes Values On CN Process	YES	YES
CNC.READVARADDR CNC.READVARNAME	Read Gcode Variable On CN Process	YES	YES
CNC.WRITEVARADDR CNC.WRITEVARNAME	Write Read Gcode Variable On CN Process	YES	YES
CNC.READPARMAC	Read Machine Parameter On CN Process	YES	YES
CNC.WRITEPARMAC	Write Machine Parameter On CN Process	YES	YES
CNC.ENABLEAXIS	Enable Axis On CN Process	YES	YES
CNC.HOMEAXIS	Homing Axis On CN Process	YES	YES
CNC.READGENERIC	Read Generic Variable On CN Process	YES	YES
CNC.WRITEGENERIC	Write Generic Variable On CN Process	YES	YES

5 PROGRAM FLOW

Isous allows programming with an extension of standard ISO code. The adding instruction to control the program flow are BASIC style increasing programming performance. Therefore it's possible to manage conditional cycles, iterative cycles, jumps to labels, etc.

5.1 IF-ELSE-END_IF

It allows conditional execution of an instructions block depending of the result of expression. IF cycles can be nested without limits.

Syntax

```
IF condition
        [instr. true]
ELSE
        [instr. false]
END_IF
```

condition Mandatory. Any numeric expression with a result of True or False.
Instr. true Instructions block executed if condition is TRUE
Istr. false Optional. Instructions block executed if condition is FALSE
END_IF End of **IF ELSE** cycle
Ex:

```
IF $VAR=$VAR2*15+($VAR4+18)
        $VAR1=10
        G1X10Y20
ELSE
        $VAR1=0
        G1X0Y0
END_IF
```

5.2 LOOP - END_LOOP

Iterative cycle of the program. Instructions inside LOOP END_LOOP cycle are executed a number of time depended of expression result. LOOP cycles can be nested without limits.

Syntax

```
LOOP variable K
        [instructions]
END_LOOP
```

variable Mandatory. Any variable type $name or numeric value
K Optional – If it is insert, in preview only one time (LOOP 100 K)
Instructions Instructions block executed in each cycle
END_LOOP End of **LOOP** cycle
Ex:

```
G91
$VAR=20+($VAR1*5)
LOOP $VAR
        G1X1.3Y1.2
        IF $VAR1=20
                GOTO EXIT_LOOP          // FORCED EXIT FROM LOOP
        END_IF
END_LOOP
@EXIT_LOOP
```

FOR – NEXT – BREAK - CONTINUE

Cycle FOR-NEXT

Iterative cycle of the program. Instructions inside FOR NEXT cycle are executed a number of time depended of expression result.

Syntax

FOR InitFor;CondFor;IterFor

[instructions]
BREAK
CONTINUE

NEXT

InitFor	Mandatory .Initialization Variable of For cycle
CondFor	Mandatory – Condition FOR cycle – Continue If condition is TRUE
IterFor	Mandatory – Values of increase/decrease Condition Variable
Instructions	Instructions block executed in each cycle
BREAK	Break For cycle
CONTINUE	Jump to next iteration
NEXT	End **FOR**

Ex:

```
G0X0Y0
$VAR1=0
$VAR2=0
$VAR3=0
$VAR4=0
FOR $V=0;$V<20;$V+=2
  G0X[$VAR1]Y[$VAR2]
  G4F0.5
  $VAR1+=10
  $VAR2+=20
  $VAR+=4
  IF$VAR<20
    CONTINUE
  END_IF
  IF$VAR4=200
    BREAK
  END_IF
  FOR $V1=0;$V1<100;$V1++
    $VAR3+=1
  NEXT
NEXT
```

WHILE – END_WHILE – BREAK - CONTINUE

Cycle WHILE-END_WHILE

Iterative cycle of the program. Instructions inside WHILE END_WHILE cycle are executed a number of time depended of expression result.

Syntax

WHILE CondWhile

[instructions]

BREAK

CONTINUE

END_WHILE

CondWhile	Mandatory – Condition WHILE cycle – Continue If condition is TRUE
Instructions	Instructions block executed in each cycle
BREAK	Break While cycle
CONTINUE	Jump to next iteration
END_WHILE	End **WHILE**

Es:

```
G0X100
$VAR1=10
$VAR=0
$VAR3=0
$VAR4=5
WHILE $VAR<$VAR1
  $VAR++
  WHILE $VAR3<200
    IF $VAR3=204
      BREAK
    END_IF
    $VAR3++
  END_WHILE
  IF $VAR4=10
    CONTINUE
  END_IF
END_WHILE
G0X200
```

5.3 GOTO

Unconditional jump to a label or line number identified by Nxx

Syntax
GOTO labelname
GOTO @linenumber

labelname Mandatory. A label of program defined with @labelname
@linenumber Line number identified by **Nxx. if LineNumber = -1 restart the Partprogram**

Ex:

```
IF $VAR1=0
        GOTO CYCLE_A
END_IF
IF $VAR1=1
        GOTO @100 // GOTO @-1 JUMP TO BEGIN
END_IF
// EXECUTE CYCLE A
@ CYCLE_A
G1X100Y100
.......
......
// EXECUTE CYCLE B
N100G1X10Y10
```

5.4 GOSUB – RETURN

Jump to and return from a subroutine defined by a label.

Syntax
GOSUB labelname
GOSUB @linenumber

labelname Mandatory. A label of program defined with @labelname
@linenumber Line number identified by Nxx

it is mandatory that an instructions block which start with label, must to end with a RETURN
Ex:

```
IF $VAR1=0
        GOSUB CYCLE_A
END_IF
IF $VAR1=1
        GOSUB @100
END_IF
// EXECUTE CYCLE A
@ CYCLE_A
G1X100Y100
RETURN
// EXECUTE CYCLE B
N100G1X10Y10
RETURN
```

5.5 LABEL

A LABEL defines a jump point that can be use with GOTO e GOSUB. It isn't possible to have two or more LABEL with the same name, because it can cause a system conflict.
LABELs are exclusive for every PART PROGRAM, therefore ISOUS defines as private the LABELs that will be found in M or HM functions, so they don't have any effect in the PART PROGRAM.
It's a good thing to have LABEL with relevant names.

Syntax
@labelname

@	Mandatory. Indentifier of LABEL
labelname	Mandatory. Unique name of the label All characters excluded math and logic operator are allowed

Ex:
```
@ CYCLE_A
G1X100Y100
.......
......
@ CYCLE_B
G1X10Y10
```

5.6 END_PROGRAM

It forces an end of program. Isous inserts automatically this instruction at end of program list (end of file). However in some cases it can be necessary to end program in a particular point other than end of file.

Syntax
END_PROGRAM

Ex:
```
IF $VAR1=0
        END_PROGRAM // END PROGRAM IN PREMATURE MODE
END_IF
```

5.7 WAIT_INPUT

It waits a digital input for a programmed time. If time elapses without input is active it is possible stop the program reporting an ALARM.

Syntax
WAIT_INPUT Input State Timeout Alarm

Input	Mandatory. Digital input number from 0 to 255
State	Mandatory. Logic state of input to continue the program (0 = OFF state , 1 = ON state)
Timeout	Time in sec. (0.1 resolution). If time elapses it's possible to signal an alarm stopping the PartProgram **TIME=0 infinite wait**
Alarm	Generated alarm code (refer to configuration file) 0 = No allarm generated

Ex:
```
WAIT_INPUT 2 1 5.3 12   // Wait input 2 at logic state ON for 5.3 sec. and
                        // signal alarm 12 if condition is not happen
```

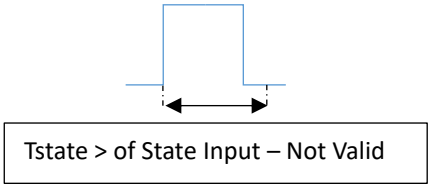
5.8 TWAIT_INPUT

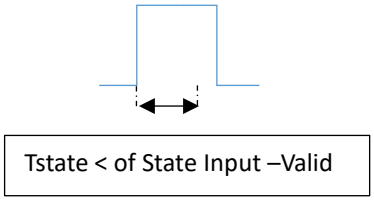
It waits a digital input for a programmed time Out and for a state time valid. If time elapses without input is active for a time valid it is possible stop the program reporting an ALARM.

Syntax

TWAIT_INPUT Input State TimeState Timeout Alarm

Input	Mandatory. Digital input number from 0 to 255
State	Mandatory. Logic state of input to continue the program (0 = OFF state , 1 = ON state)
TimeState	TimeState in sec. (0.1 resolution). The input must be in the programmed state for this time **TIME=0 like to WAIT_INPUT**
Timeout	Time in sec. (0.1 resolution). If time elapses it's possible to signal an alarm stopping the PartProgram **TIME=0 infinite wait**
Alarm	Generated alarm code (refer to configuration file) 0 = No allarm generated





Ex:

```
TWAIT_INPUT 2 1 2 5.3 12        // Wait input 2 at logic state ON Tstate 2 sec for 5.3 sec. and
                                // signal alarm 12 if condition is not happen
```

5.9 ERROR

It cause an exit from PartProgram reporting an error. This instruction is useful to generate exceptions of PartProgram at events occurrences. Execution is stopped with an alarm.

Syntax
ERROR err_code

err_code Mandatory. Error code (refer to configuration file).

Ex:

```
IF $VAR1=0
        ERROR 10          // STOP PARTPROGRAM WITH ALLARM 10
END_IF
```

5.10 RESUME_T

Resumes the part program from the last statement used Tn.
It is used for tool management Damaged. In practice, when the measurement tool detects that the tool is damaged, through **RESUME_T** you can redo the work from last tool change made.
The parameter that follows **RESUME_T** is the tool alternative to that Damaged. This parameter is taken from the table tools.

Syntax
RESUME_T Tool_nr

Tool_nr Alternative Tool Number

5.11 SAVE_T

Saves the tools table in Isous cfg file.
With PartProgram is possible modify the tools table value. **SAVE_T** saves in permanent mode the modify.

Syntax
SAVE_T

```
T1 // SELECTED T1
$[U0]=10 // WRITE TOOL DIAMETER INDEX TABLE 0
$[U1]=100.3 // WRITE TOOL LENGHT INDEX TABLE 1
SAVE_T  // SAVE THE TOOLS TABLE
```

5.12 SET_TABPAR

Set a Parameter Table and activates the parameters values of the table.
The parameters Tables are generated by IsoUs configurator, and these contain a copy of original machine parameters, but with different values. That allows a dynamic parametrization by Gcode.
The TABLE 0 contains a copy of DEFAULT PARAMETERS.

Syntax
SET_PARTAB Tab_nr

Tab_nr is the TABLE number to activate. Table 0 original parameters

5.13 SELECT – CASE - END_SELECT

Executes one of different groups of instructions conditioned by value of expression.

Syntax

```
SELECT Test Variable
        CASE Exp1
                [Instructions]
        CASE Exp2
                [Instructions]
        .
        .
        CASE ELSE
                [Instructions]

END_SELECT
```

SELECT Test Variable	Mandatory must be a variable $ - Start cycle
CASE Num	Condition True– Value=Num
CASE Exp	Condition True – Value =Exp
CASE Exp1 TO Exp2	Condition True - Values between Exp1 1 and Exp2
CASE Exp1,Exp2,Exp3..	Condition True – Values equal to Exp1, or Exp2,or Exp3, etc.
CASE IF..	Condition True – is evaluated the IF expression
CASE ELSE	Optional. List of Instructions executed if not any Case cycle was True (must be as last condition, before the END_SELECT)
END_SELECT	End Select Cycle

Ex:

```
SELECT $VAR
        CASE 10
        .
        .
        CASE $VAR1
        .
        .
        CASE $VAR2,15,$VAR3
        .
        .
        CASE 20 TO $VAR4
        .
        .
        CASE IF>=$VAR5
        .
        .

        CASE ELSE
        .
        .
END_SELECT
```

5.14 IF RUN

Check type of Gcode execution. Condition True **NORMAL** execution.
Like to **IF $[X7]=0**

Syntax

IF RUN

..
..
..

END_IF

5.15 IF NOTRUN

Check type of Gcode execution. Condition True **SIMULATED** execution.
Like to **IF $[X7]<>0**

Syntax

IF NOTRUN

..
..
..

END_IF

5.16 USEFORM

Open a FORM compiled by PlugIn **UsFormManager**

Syntax

USEFORM "FORMNAME"

FORMNAME Form Name without extension compiled (folder \Data_Form)

xs:
USEFORM "FORM1"

6 GENERIC INSTRUCTIONS

6.1 SDO_DL

CanOpen SDO Down-Load function (refer to CanOpen documentation).
Write data in a CanOpen node.

Syntax
SDO_DL node index subindex len_data $var

node	Mandatory. CanOpen node
index	Mandatory. Object index
subindex	Mandatory. Object sub-index
len_data	Mandatory. Lenght of datas inside $Var (1 char – 2 int – 4 long)
$Var	Mandatory. Variable containing data.

Ex:
```
// SEND TO NODE 2 AT INDEX 24596 AND SUBIND. 0
// 4 BYTE FROM VARIABLE $VAR1
SDO_DL 2 24596 0 4 $VAR1
```

6.2 SDO_UL

CanOpen SDO Up-Load function (refer to CanOpen documentation).
Read data in a CanOpen node.

Syntax
SDO_UL node index subindex len_data $var

node	Mandatory. CanOpen node
index	Mandatory. Object index
subindex	Mandatory. Object sub-index
len_data	Mandatory. Lenght of datas inside $Var (1 char – 2 int – 4 long)
$Var	Mandatory. Variable where data will be write

Ex:
```
// READ FROM NODE 2 AT INDEX 24596 AND SUBIND. 0
// 4 BYTE WRITED IN VARAIBLE $VAR1
SDO_UL 2 24596 0 4 $VAR1
```

6.3 GET

Read parameter for HM function

ATTENTION: PARAMETERS ARE IN REVERSE ORDER COMPARED TO HOW THEY WERE WRITED

Ex:
```
HM 10 20 30 40
..
GET $PAR1 $PAR2 $PAR3 $PAR4     // READ PARAMETERS
```
then
$PAR1=40 $PAR2=30 $PAR3=20 $PAR4=10

6.4 READ_PARMAC

Read a machine parameter with name indication.

Syntax
READ_PARMAC ParName $var

ParName Mandatory. Parameter name to read (ex: VJOG_X)
$Var Mandatory. Variable where parameter value will be write

A runtime error will be generated if "parname" doesn't exist

Ex:
READ_PARMAC VJOG_X $VAR1

6.5 WRITE_PARMAC

Write a machine parameter with name indication.

Syntax
WRITE_PARMAC ParName $var

ParName Mandatory. Parameter name to read (ex: VJOG_X)
$Var Mandatory. Variable containing value to be write

A runtime error will be generated if "parname" doesn't exist
ATTENTION: write of parameters can be cause system errors if execute inappropriately

Ex:
$VAR1=100
WRITE_PARMAC "VJOG_X" $VAR1

6.6 OPT

Set the option of compiler for M functions.
Change the function M for actual PartProgram

Syntax
OPT Options Value

Options Options:
MSTOP,MSTART,MEND,MGOBLOCK,MGORETRACE,MERROR,MPAUSE,MGOPAUSE
Value New value for M functions
EX:
OPT MSTOP 10 //M STOP = 10 for actual PartProgram
OPT MERROR -1 //Disable M error for actual PartProgram

6.7 PAUSE_MODE

Defines the way in which CNC must act PAUSE.
The modes are as follows:

0	Normal Mode(Pause Enabled,M Pause Enabled,M Resume Pause Enabled)
1	Pause Enabled, M Pause Enabled, **M Resume Pause Disabled**
2	Pause Enabled, **M pause Disabled**, M Resume Pause Enabled
3	Pause Enabled, **M pause Disabled**, **M Resume Pause Disabled**
4	**PAUSE DISABLED ON CNC**

Syntax

PAUSE_MODE **val**

Val	Value or expression

Pause_Mode used to disable or limit the functionality of the PAUSE cycles during some special work: Ex: TOOL CHANGE etc..

6.8 FILTER_MODE

Defines the Filters function G69,G66 and other

The modes are as follows:

0	**LOOK AHEAD** disabled on IsoUs
1	**LOOK AHEAD** enabled with last value setted
2	Disabled **PAUSE and STOP**
3	Enabled **PAUSA and STOP**
4	Disabled **SAVE**_T function
5	Enabled **SAVE**_T function

Sintassi

FILTER_MODE **val**

Val	Valore o espressione

6.9 IMPORT

Import an external file with various formats.
The predefined format is ISOUS. If present **NsImportFile.dll** file, this defines the import format.

Syntax

IMPORT **“filename.ext”**

filename	It containes the absolute or relative file path. If **FileName** initiates with $APPPATH,, the file is searched in the folder ISOUS Import generates the events StartImport and EndImport when the file terminated execution The MAIN part program resume execution when Import File is terminated. import multiple files can be nested .

Ex:

IMPORT “$APPPATH\PROJECT\IMPORT\TESTIMPORT.ISO” **// Import from Isous folder**

IMPORT “C:\FILEIMPORT\TESTIMPORT.ISO” **// import from absolute path**

6.10 END_IMPORT

Terminate in anticipated mode a program called from function IMPORT and return to back program calling

Syntax

END_IMPORT

Ex:

```
IF $VAR=0
        END_IMPORT
END_IF
```

6.11 STOP_MODE

Defines the buttons STOP function
The modes are as follows:

0	Normal Mode
1	The STOP is disabled when the MACRO STOP is in EXECUTION This allows to don't INTERRUPT NEVER the MACRO STOP
2	Interrupt the MACRO STOP with TWO PRESSIONS of BUTTON STOP This allows to INTERRUPT the MACRO STOP, but must be press 2 times the button STOP

Syntax

STOP_MODE val

Val Value or expression

6.12 DEBUG_INFO

Write informations in the IsoUs_x.log file for Debug

Syntax

DEBUG_INFO "TextInfo" $var

TextInfo Mandatory. Text of Informations

TextInfo controls some special chars for define the **DEBUG_INFO** action.

@ If inserted as **FIRST** char of **TextInfo**, the message is inserted in the **Log** file and send also to **Notify** panel.

Ex: DEBUG_INFO "@TextInfo" $var

& If inserted as **FIRST** char of **TextInfo** the message is write in the file **_Us_DebugInfo.txt**. this file is located in the same folder of IsoUs

Ex: DEBUG_INFO "&TextInfo" $var

$ If inserted as **LAST** char of **TextInfo** is generated the event **UsDebugInfo**

Ex: DEBUG_INFO "TextInfo$" $var

$Var Mandatory. Variable of Debug can contain more informations.

The variable is printed after the TextInfo

UsDebug --> TextInfo : $Var

ATTENTION: TextInfo must be inside "..."

```
$VAR1=1
DEBUG_INFO "TEST DEBUG INFO" $VAR1
```

7 INSTRUCTIONS FOR MANAGEMENT DISPLAY REMOTE HANDWHEEL WHC

7.1 LIB.HMESSAGE

Show text on display Remote HandWheel Promax WHC.
The WHC display has 4 ROW x 20 COL

Syntax

LIB.HMESSAGE "TEXT" DURATION_TIME

TEXT → Text of message
Special characters:
@RCC
WHERE R=Row Number (from 0 to 3)
CC=Col Number (from 0 to 19) the Col Number must have always 2 characters)
Ex: LIB.HMESSAGE "@105TEEXT" 5 Write on Row 1 Col 5

DURATION_TIME → Duration in Sec. of permanency (after this time the text will be deleted)

LIB.HMESSAGE "@100TEST TEXT" 5
write "TEST TEX"T Row 1 Col 0 for 5 seconds

8 EXTENDED INSTRUCTIONS

8.1 EXD.STL_LOAD

Allows to load a 3D model file STL in the preview. Valid only for **RealMachine** simulation.

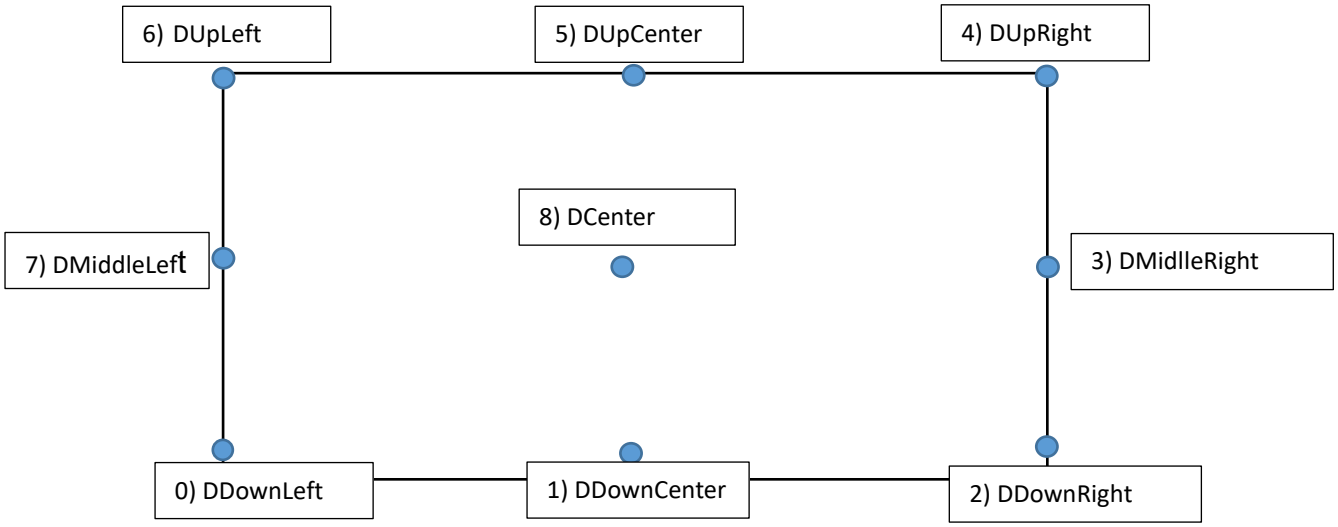
Syntax

EXD.STL_LOAD "STLNAME" Xval Yval Zval Aval

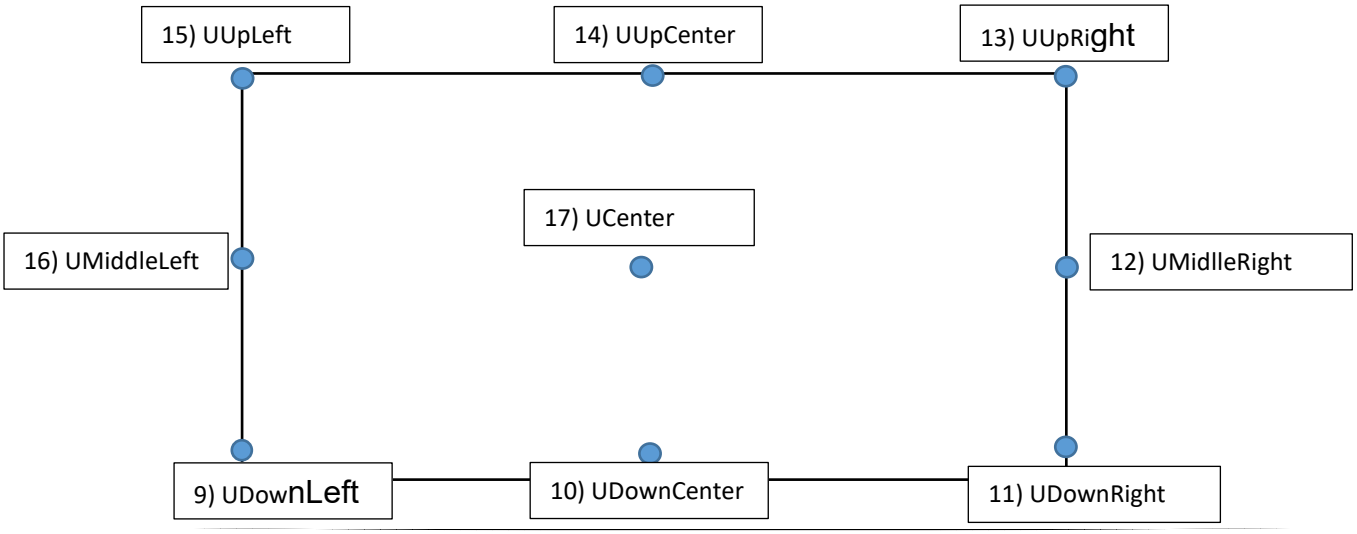
STLNAME	STL name (without extension) the file must be saved in the folder **\UsMachines\DynamicStl**
X	Optional Center X (if not inserted is used position in MachineSettings)
Y	Optional Center Y (if not inserted is used position in MachineSettings)
Z	Optional Center Z (if not inserted is used position in MachineSettings)
A	Optional Alignement STL: (if not inserted is used position in MachineSettings)

VALUEs FOR A PARAMETER

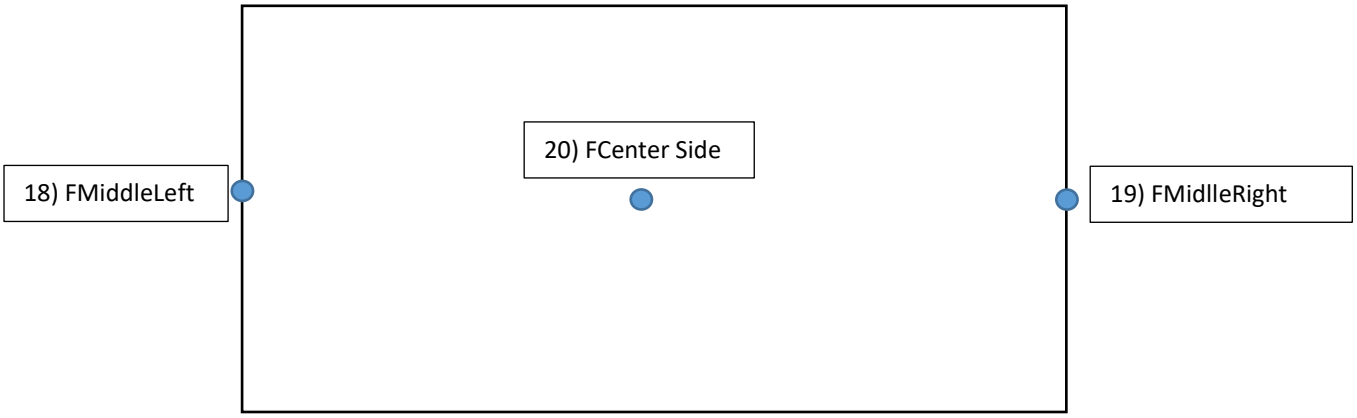
SIDE DOWN



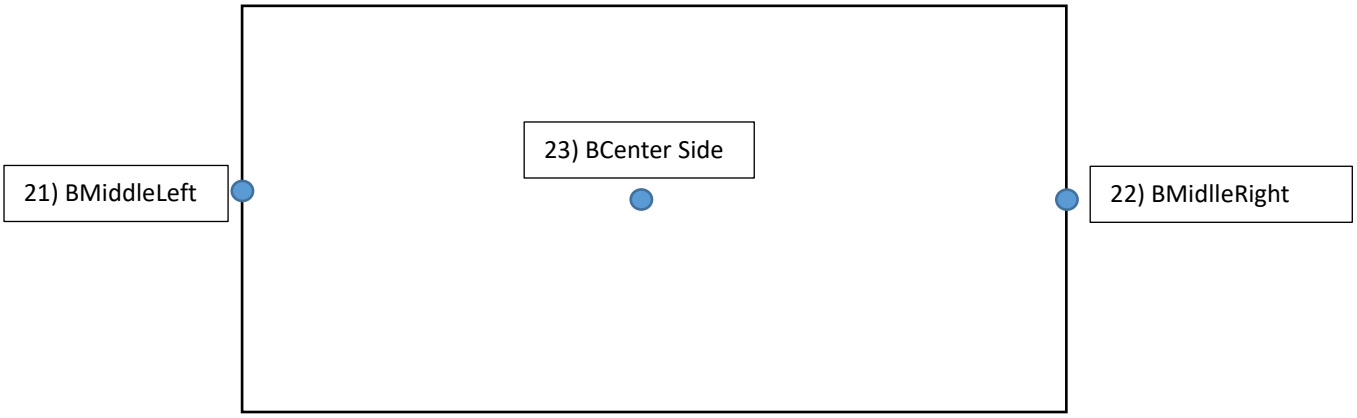
SIDE UP



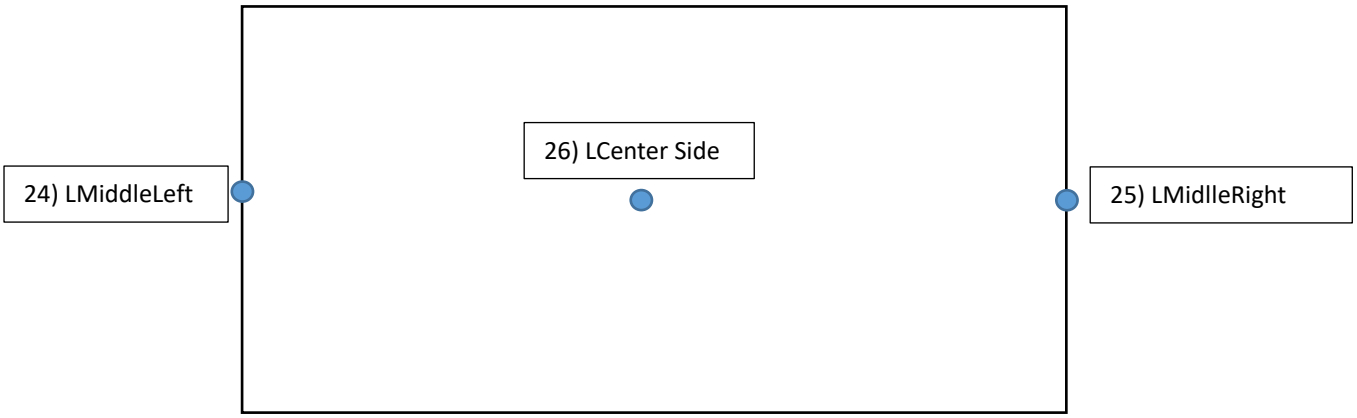
SIDE FRONT



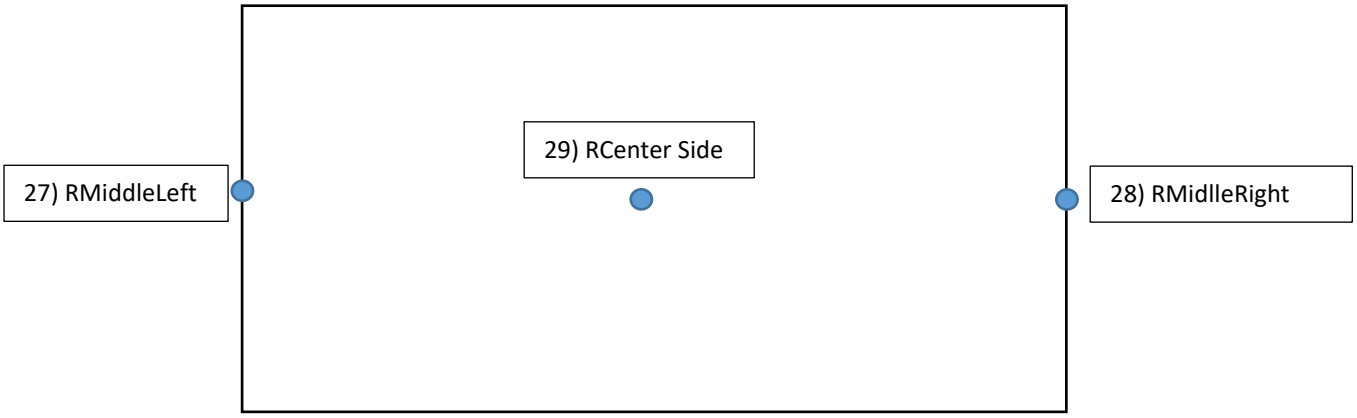
SIDE BACK

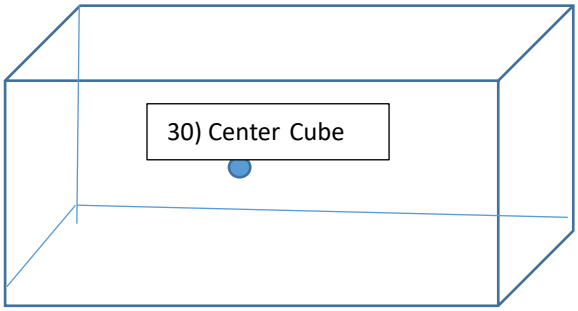


SIDE LEFT



SIDE RIGHT

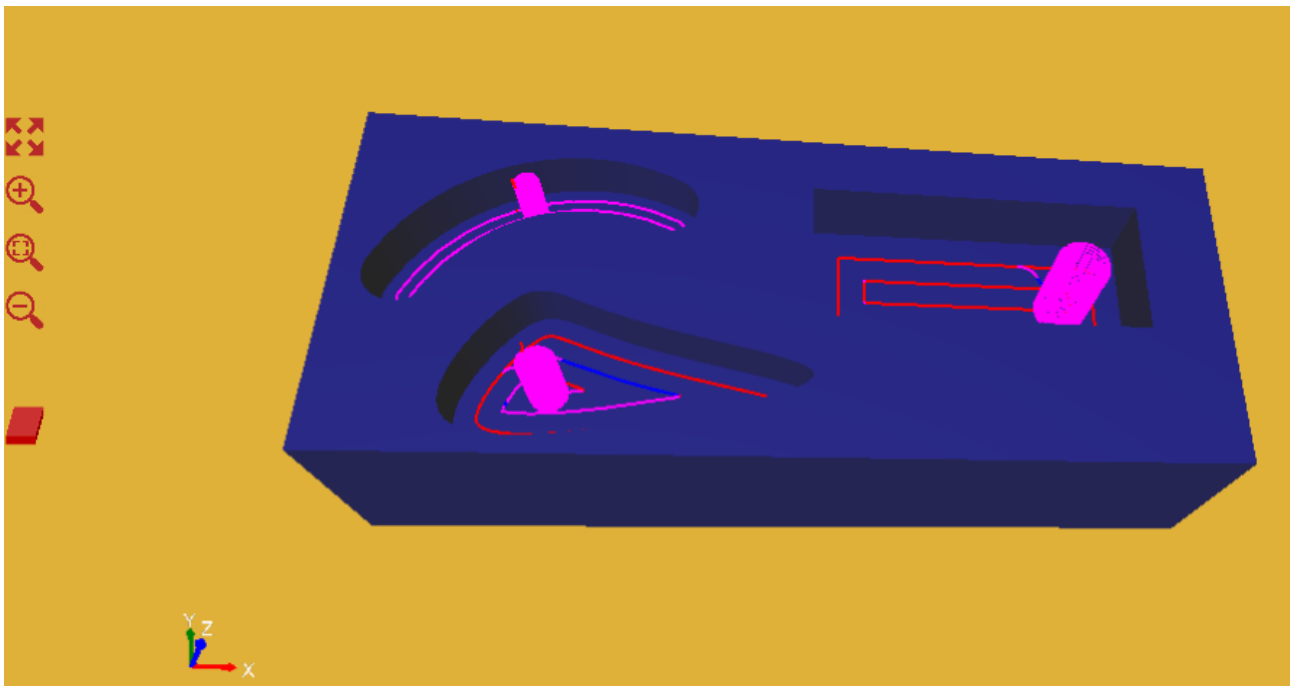




31) Default defined in XML
32) Model Origin – Origin get for STL file

Ex:

EXD.STL_LOAD "TEST"



8.2 EXD.READ_TOOLPAR

Allows to read a Tool Table parameter. Like to $[Un] but allows to set the Tool Table Index

Syntax

EXD.READ_TOOLPAR TabIndex ParIndex $DEST

TabIndex Table Index (Numeric or Variable)
ParIndex Parameter Index (Numeric or Variable)
$DEST Destination Variable

Ex:

$TABINDEX=2

$PARINDEX=1

EXD.READ_TOOLPAR **2 1 $Dest** //READ THE PARAMETER 1 TOOL DIAMATER FROM TABLE 2
EXD.READ_TOOLPAR **$TABINDEX $PARINDEX $Dest** //READ THE PARAMETER 1 TOOL DIAMATER FROM TABLE 2

8.3 EXD.READ_HEADPAR

Allows to read a Head Table parameter. Like to $[Hn] but allows to set the Head Table Index

Syntax

EXD.READ_HEADPAR TabIndex ParIndex $DEST

TabIndex Table Index (Numeric or Variable)
ParIndex Parameter Index (Numeric or Variable)
$DEST Destination Variable

Ex:

$TABINDEX=2

$PARINDEX=1

EXD.READ_HEADPAR **2 1 $Dest** //READ THE PARAMETER 1 FROM TABLE 2
EXD.READ_HEADPAR **$TABINDEX $PARINDEX $Dest**

8.4 EXD.WRITE_TOOLPAR

Allows to write a Tool Table parameter. Like to $[Un] but allows to set the Tool Table Index

Syntax

EXD.WRITE_TOOLPAR TabIndex ParIndex $SOURCE

TabIndex Table Index (Numeric or Variable)
ParIndex Parameter Index (Numeric or Variable)
$SOURCE Source Variable

Ex:

$TABINDEX=2

$PARINDEX=1

$SOURCE=10

EXD.WRITE_TOOLPAR **2 1 $SOURCE** //WRITE THE PARAMETER 1 TOOL DIAMATER IN TABLE 2 AT VALUE 10
EXD.WRITE_TOOLPAR **$TABINDEX $PARINDEX $SOURCE**

8.5 EXD.WRITE_HEADPAR

Allows to write a Head Table parameter. Like to $[Hn] but allows to set the Head Table Index

Syntax

EXD.WRITE_HEADPAR TabIndex ParIndex $SOURCE

TabIndex Table Index (Numeric or Variable)
ParIndex Parameter Index (Numeric or Variable)
$SOURCE Source Variable

Ex:

$TABINDEX=2

$PARINDEX=1

$SOURCE=10

EXD.READ_HEADPAR **2 1 $SOURCE** //WRITE THE PARAMETER 1 IN TABLE 2 AT VALUE 10
EXD.WRITE_HEADPAR **$TABINDEX $PARINDEX $SOURCE**

8.6 EXD.LOAD_LAST

Load in a variable the last Tool Table used by **Tn from file** (“UTENSILE.INF”)

Syntax

EXD.LOAD_LAST $DEST

$DEST Destination Variable

Ex:

EXD.LOAD_LAST **$DEST**

8.7 EXD.SAVE_LAST

Save in the file (“UTENSILE.INF”) the current Tool Table set by **Tn**

Syntax

EXD.SAVE_LAST

Es:

EXD.SAVE_LAST

8.8 EXD.RESET_LAST

Reset the file (“UTENSILE.INF”) No Last Tool Used

Syntax

EXD.RESET_LAST

Es:

EXD.RESET_LAST

8.9 EXD.SET_OUT

Set/Reset of Digital Outputs by Index like to **$[On]** but is used the variables for output index and output state.

Syntax

EXD.SET_OUT $OutIndex $State

Ex:

$INDEX=10

$STATE=1

EXD.SET_OUT **$INDEX $STATE** //SET OUT 10

8.10 EXD.READ_OUT

Read of Digital Outputs by Index like to **$[On]** but is used the variable for output index.

Syntax

EXD.READ_OUT $OutIndex $Dest

Ex:

$INDEX=10

EXD.READ_OUT **$INDEX $DEST** //READ OUT 10 IN $DEST

8.11 EXD.READ_INP

Read of Digital Inputs by Index like to **$[In]** but is used the variable for input index.

Syntax

EXD.READ_INP $InptIndex $Dest

Ex:

$INDEX=10

EXD.READ_INP **$INDEX $DEST** //READ INP 10 IN $DEST

8.12 EXD.WRITE_USER

Write an User generic variable in the CN by Index. Like to **$[Kn]** but is used the variable for index.

Syntax

EXD.WRITE_USER $Index $Value

Ex:

$INDEX=10

$VALUE=100

EXD.WRITE_USER **$INDEX $VALUE** //WRITE USER 10 VALUE 100

8.13 EXD.READ_USER

Read an User generic variable in the CN by Index. Like to **$[Kn]** but is used the variable for index.

Syntax

EXD.READ_USER $Index $Dest

Ex:

$INDEX=10

EXD.READ_USER **$INDEX $DEST** //RED USER 10 IN $DEST

8.14 EXD.READ_DEMAND

Read demand position of Axis. Like to **$[Qn]** but is used the variable for axis index.
Value <100 from 0 to 8 read position with OFFSET (Work Origins etc.)
Value>= 100 from 100 to 108 read position without OFFSET

Syntax

EXD.READ_DEMAND $Axis $Dest

Ex:

$Axis=1

EXD.READ_DEMAND **$AXIS $DEST** //READ DEMAND AXIS 1 WITH OFFSET IN $DEST

$Axis=101

EXD.READ_ DEMAND **$AXIS $DEST** // READ DEMAND AXIS 1 WITHOUT OFFSET IN $DEST

8.15 EXD.READ_REAL

Read Real position of Axis. Like to **$[Rn]** but is used the variable for axis index.
Value <100 from 0 to 8 read position with OFFSET (Work Origins etc.)
Value>= 100 from 100 to 108 read position without OFFSET

Syntax

EXD.READ_REAL $Axis $Dest

Ex:

$Axis=1

EXD.READ_REAL **$AXIS $DEST** //READ REAL AXIS 1 WITH OFFSET IN $DEST

$Axis=101

EXD.READ_ REAL **$AXIS $DEST** // READ REAL AXIS 1 WITHOUT OFFSET IN $DEST

8.16 EXD.MASK

Set A **MASK** for force a state of Digital Output.

Syntax

EXD.MASK Type OutIndex

Type	Variable or numeric value that defines the MASK type
	Type=0 Force the output **OFF**
	Type=1 Force the output **ON**
	Type=2 Invert the state of Output
	Type=3 Reset MASK of output
OutIndex	Digital Output Index

Ex:

$TYPE=0

$OUTINDEX=10

EXD.MASK **$TYPE $OUTINDEX** //FORCE OUT 10 OFF

EXD.MASK **3 10** //RESET MASK OUT 10

8.17 EXD.WRITE_BIT

Set/Reset Variable bit

Syntax

EXD.WRITE_BIT $BIT $VAR $VAL

BIT	Variable or Value Nr. Bit to Set/Res (from 0 to 31)
VAR	Destination Variable
VAL	Variable or Value state of Bit (0 Reset 1 Set)

Ex:

$BIT=5

$VAL=1

EXD.WRITE_BIT **$BIT $VARDEST $VAL** //SET BIT 5 OF VARDEST

EXD.WRITE_BIT **5 $VARDEST 0** //RESSET BIT 5 OF VARDEST

8.18 EXD.READ_BIT

Read bit state of variable

Syntax

EXD.READ_BIT $BIT $VAR $DEST

BIT	Variable or Value Nr. Bit to Set/Res (from 0 to 31)
VAR	Source Var
VAL	Destination Var state of bit (0 or 1)

Ex:

$BIT=5

EXD.READ_BIT **$BIT $VARSOURCE $VARDEST** //READ BIT 5 OF VARSOURCE

EXD.READ_BIT **5 $ VARSOURCE $VARDEST** // READ BIT 5 OF VARSOURCE

8.19 EXD.PXV_ALL_DETECTORS

Read ALL detectors of PXV Sytem
Ex. For control the tools presence in the charger

Syntax

EXD.PXV_ALL_DETECTORS $SHOWMODE $TOOLS $ISOK

SHOWMODE Allows to show an image in the PXV Simulation. Variable or Value (for debug)
SHOWMODE=0 NONE
SHOWMODE=1 Show image **FILTERED**
SHOWMODE=2 Show **ORIGINAL** Image

TOOLS Return state of detectors read – Bit Mapped (max 32 detectors from 0 to 31)
Bit=0 Detector ON
Bit=1 Detector OFF

ISOK If **=-1 ERROR READ DETECTORS** . Else Nr. Detectors read

Ex:

```
EXD.PXV_ALL_DETECTORS 0 $TOOLS $ISOK //READ DETECTORS
IF $ISOK<>-1 //IF OK
        EXD.READ_BIT 0 $TOOLS $VARDEST //READ DETECTOR 0
        IF $VARDEST=1   //ON
                ....
        ELSE            //OFF
                ....
        END_IF
END_IF
```

8.20 EXD.PXV_SINGLE_DETECTOR

Read state of detector in PXV system

Syntax

EXD.PXV_SINGLE_DETECTOR $NRDET $SHOWMODE $VAR

NRDET Detector number to read (from 0 to 31) Variable or Value

SHOWMODE See **EXD.PXV_ALL_DETECTORS**

$VAR Value Read
-1 ERROR
0 OFF
1 ON

Ex:

```
EXD.PXV_SINGLE_DETECTOR 0 0 $RET //READ DETECTOR 0
IF $RET<>-1 // OK
        IF $RET=1       //ON
                ....
        ELSE            //OFF
                ....
        END_IF
END_IF
```

8.21 EXD.PXV_READ_PROBE

Read state of Probe in PXV system
Return the X,Y position Set

Syntax

EXD.PXV_READ_PROBE $PROBE $XVAL $YVAL $ISOK

PROBE	Probe Number Variable or Value
XVAL	IF ISOK=1 X Position of Probe
YVAL	IF ISOK=1 Y Position of Probe
ISOK	**=-1 PROBE READ ERROR**

Ex:

```
EXD.PXV_READ_PROBE 0 $VX $VY  $ISOK    //READ PROBE 0
IF $ISOK=1          // OK
        G0 X[$VX] Y[$VY]          //MOVE X,Y TO PROBE 0 POSITION
END_IF
```

8.22 EXD.PXV_RESET_PROBE

Reset a Probe in PXV system
This Probe will be deactivated and if is read a ERROR will Return

Syntax

EXD.PXV_RESET_PROBE $PROBE

PROBE	Probe Number Variable or Value

Ex:

```
EXD.PXV_RESET_PROBE 0          //RESET PROBE 0
```

EXD.PXV_SET_DETECTOR

Set the type of Detector (used in system PXV)

Syntax

EXD.PXV_SET_DETECTOR $MODE

MODE	Type of Detector Variable or Value	
	MODE=0	Contrast Detector
	MODE=1	Brightness Detector
	MODE=2	Area Detector
	MODE=3	Color Detector

Ex:

```
EXD.PXV_SET_DETECTOR 0          //SET MODE TO CONTRAST DETECTOR
```

8.23 EXD.PXV_SAVE_IMAGE

Save current image acquired in PXV in file jpg
Used for capture machine state (ex. After an ERROR)
Used in combination to **EXD.PXV_GET_IMAGE**
Will be saved TWO Images FILTERED and ORIGINAL:

_PxvImgFiltered.jpg	**FILTERED**
_PxvImgNoFiltered.jpg	**ORIGINAL**
_PxvImgNoFiltered1.jpg	**ORIGINAL ALTERNATIVELY TO _PxvImgNoFiltered.jpg**

Syntax

EXD.PXV_SAVE_IMAGE

8.24 EXD.PXV_GET_IMAGE

Image Capture for PXV
The captured Image can be save by **EXD.PXV_SAVE_IMAGE**

Syntax

EXD.PXV_GET_IMAGE $ISOK

ISOK	**0**	**ERRORE CAPTURE**
	1	**OK**

```
EXD.PXV_GET_IMAG  $ISOK        //CAPTURE
IF $ISOK=1 // OK
        EXD.PXV_SAVE_IMAGE     //SAVE THE IMAGE CAPTURED
END_IF
```

8.25 EXD.PXV_SET_JOB

Set a **JOB PXV** Configuration File
The **JOB** must be created by **PxVisionBrowser**

Sintassi

EXD.PXV_SET_JOB «JOBNAME» $ISOK

JOBNAME	**JOB File Name del file JOB (in quote)**	
ISOK	**0**	**ERROR SET JOB**
	1	**OK**

Ex:

```
EXD.PXV_SET_JOB  "MYJOB" $ISOK
```

8.26 EXD. RUN_SCRIPT

This function allows to execute a **SCRIPT** code in **C#**. The **SCRIPT** is compiled in **RUN.-TIME** and isn't necessary any external development software
The **SCRIPT** has access to all resources of current process **ISO US** , ex . Read or Write Gcode **VARIABLES**.
This allows to use a **STANDARD PROGRAMMING LANGUAGE** with high performances.
All parameters of **EXD.RUN_SCRIPT** are separated by **COMMA "," ** and they have an identifier (*,&,@,X) that define the parameter type.
The **SCRIPT** must be found in the **FOLDER ISO US->Us_Script** and **MUST HAVE AN EXTENSION .USS.**
There are no limit to **SCRIPT** number.
The SCRIPT code will finish when the RETURN is performed from MAIN function.

IDENTIFIERS :

- ***)** Script Name (without extension) **MANDATORY**
  - ***Only ONE parameter of this type is admitted***
- **&)** Input parameter for script **Main** function **NOT MANDATORY**
  - ***No limit for this parameter***
  - ***Can be:***
    - ***Variable $***
    - ***Variable:***
    - ***DIRECT Values***
- **@)** Output parameter to script **Main** function **NOT MANDATORY**
  - ***No limit for this parameter***
  - ***Can be:***
    - ***Variable $***
- **X**) Mode of Script execution **NOT MANDATORY**
  - ***Only ONE parameter of this type is admitted***
  - ***Can be:***
    - ***DIRECT VALUE 0 or 1***
    - ***X0, The Script isn't executed in DEMO MODE (Preview) DEFAULT***
    - ***X1, The Script is executed in DEMO MODE (Preview)***

Syntax

EXD.RUN_SCRIPT *SCRIPT_NAME, &$VAL,&:ADDR,&100,@$RET,@$RET1

SCRIPT_NAME	**Script name WITHOUT EXTENSION**
&	**INPUT Parameters**
@	**OUPUT Parameters**

Es:

```
$VAR=127.14
:5000=37.18
EXD.RUN_SCRIPT *TESTSCRIPT,&$VAR,&:5000,&500.14,@$RET,@$RET1,@$RET2
//$VAR FIRST  INPUT VALUE (127.14)
//:5000 SECOND INPUT VALUE (37.18)
//500.14 THIRD INPUT VALUE
//$RET FIRST OUTPUT VALUE
//$RET1 SECOND OUTPUT VALUE
//$RET2 THIRD OUTPUT VALUE
```

RULES OF SCRIPT CODE

The **SCRIPT** must follow some **RULES** for a correct COMPILATION and EXECUTION.
ISO US generate an example in the folder **Us_Script-> UsExample.uss**

BASE RULES:

namespace Must be **UsFunctions**

Class Must be **UsFunctions type PUBLIC**

Entry Point Must be:
public static bool Main(UsWork.IsoUs MyIsoUs,double[] Par,out double[] ValOut)
where:

MyIsoUs (type UsWork.IsoUs) current instance of **FRAMEWORK** running this allows to access at all resources of **IsoUS**
See
https://www.promax.it/file_download/ENG/IsoUs%20Framework_eng.pdf

Par Array of double that contains **INPUT** values (if presents) in the sorted of defined in the function **EXD.RUN_SCRIPT**
If not **PRESENT** any **INPUT** parameter, **Par=null**

ValOut Array of double that contains **OUTPUT** values (if presents) read in the sorted of defined in the function **EXD.RUN_SCRIPT**
If not **PRESENT** any **OUTPUT** value, **ValOut =null**

Ex: **EXD.RUN_SCRIPT *XXX,&$x,@$VAL1,@$VAL2,@$$VAL3**
$VAL1=ValOut[0]
$VAL2=ValOut[1]
$VAL3=ValOut[2]

The function Main return a bool value:

True **SCRIPT EXECUTION OK**
False **EXECUTION ERROR**

ERROR EXECUTION SCRIPT
The script can generate **ERRORS** if **COMPILATION** or **EXECUTION** (exceptions)
From Gcode will be generate an **ALARM** "**ERROR EXECUTION SCRIPT**".
For understand better the ERROR type, IsoUs will write (in the same folder of IsoUs) a file **"_UsScript.Err"**
This contains more informations about the errors.

SCRIPT EXAMPLE

```
@ASSEMBLY System.dll,Microsoft.CSharp.dll,System.Core.dll,System.Data.dll,ComSynk.dll,Compiler.dll,UsWork.dll
//@ASSEMBLY references - add new references if necessary
using System;
using System.Windows;
namespace UsFunctions
{
	public class UsFunctions
	{
		//Function Main 3 par input return 3 ValOut
		public static bool Main(UsWork.IsoUs MyIsoUs,double[] Par,out double[] ValOut)
		{
			ValOut=null;
			try
			{
				ValOut=new double[3];	// 3 ValOut
				ValOut[0]=Par[0]*2;	// Par input 0*2
				ValOut[1]=Par[1]*2;	// Par input 1*2
				ValOut[2]=Par[2]*2;	// Par input 2*2
				for(int n=0;n<300;n++)	//Write IsoUs Var From 1000 to 1300
					MyIsoUs.MyMaster.GetLink.MyCpu.VarCpu[0].VarIsoNs[1000+n]=n*10;
				TestCall(); //Test function call
				return true; // return OK
			}
			Catch {  return false; // Error Script}
		}
		//Test Function Call
		static void TestCall()
		{
			…
		}
	}
}
```

Call by Gcode:

```
$VAR=127.14
:5000=37.18
EXD.RUN_SCRIPT *TESTSCRIPT,&$VAR,&:5000,&500.14,@$RET,@$RET1,@$RET2
```

3 input parameters

Par[0]=127.14 -> $VAR
Par[1]=37.18 -> :5000
Par[2]=500.14

3 Output values

ValOut[0]=Par[0]*2 -> 254.28
ValOut[1]=Par[1]*2 -> 74.36
ValOut[2]=Par[2]*2 -> 1000.28

The example writes also the IsoUs variables from addr 1000 to addr 1300 with values from 0 to 300

SCRIPT LIMITS

There are no limits for code of SCRIPT.
You need to be careful about the execution TIME of SCRIPT.
Time > 2-3 sec can block the IsoUs application whit a fatal error

1) EXD.OPEN_DLL

Allows to execute a **DLL .NET** . The DLL must be write with Visual Studio in **C#** or **VbNet**.
The **DLL** has access to all resources of current process **ISO US** , ex . Read or Write Gcode **VARIABLES**.
All parameters of **EXD.OPEN_DLL** are separated by **COMMA "," ** and they have an identifier (*,&,@,X) that define the parameter type.
The **DLL** must be found in the **FOLDER ISO US->Us_Dll** and **MUST HAVE AN EXTENSION .dll.**
There are no limit to **DLL** number.
The DLL code will finish when the EXD.CLOSE_DLL will be invoke or when the Gcode is ended.

Is invoked the function public bool **Load**(**UsWork**.IsoUs **IsoUs**, double[] **ParInput**, out double[] **ValOut**)

IDENTIFIERS :

- ***)** Dll Name (without extension) **MANDATORY**
  - ***Only ONE parameter of this type is admitted***
- **&)** Input parameter for script **Load** function **NOT MANDATORY**
  - ***No limit for this parameter***
  - ***Can be:***
    - ***Variable $***
    - ***Variable:***
    - ***DIRECT Values***
- **@)** Output parameter to script **Load** function **NOT MANDATORY**
  - ***No limit for this parameter***
  - ***Can be:***
    - ***Variable $***
- **X**) Mode of Dll execution **NOT MANDATORY**
  - ***Only ONE parameter of this type is admitted***
  - ***Can be:***
    - ***DIRECT VALUE 0 or 1***
    - ***X0, The Dll isn't executed in DEMO MODE (Preview) DEFAULT***
    - ***X1, The Dll is executed in DEMO MODE (Preview)***

The DLL MUST IMPORT THE INTERFACE UsWork.IsoUs.IUsDll.
This force the following Methods and Property:

public bool **Load**(**UsWork**.IsoUs **IsoUs**, double[] **ParInput**, out double[] **ValOut**)

Must be call for **FIRST ONLY ONE TIME**

IsoUs	Current process of IsoUs
ParInput	Array of double **INPUT** parameters if presents, **null if not presents**
ValOut	Array of **OUTPUT** parameters if presents, **null if not presents**
Ritorna	**True** Load **DLL OK**
	False Load **DLL ERROR**

public bool **CallFunc**(double[] **ParInput**, out double[] **ValOut**)

Can be called during the Gcode cycle
The DLL must be open by Load

ParInput	Array of double **INPUT** parameters if presents, **null if not presents**
ValOut	Array of **OUTPUT** parameters if presents, **null if not presents**
Ritorna	**True** Call **DLL OK**
	False Call **DLL ERROR**

public void **Close**()

Close DLL.
During the Closing all **DLL** resources will be disposed programmatically
The Close is always called in **AUTOMATIC MODE** at Gcode **END** or Gcode **ALARM**

public int **Synk** { get; set; }

It is used for DLL synchronization with Gcode without block the application.
The **SYNK** must be set during the Load DLL at value -1 programmatically (synk not ready).
Values different to -1 **ENABLE THE SYNK AND RELEASE THE GCODE TO NEXT LINE**.
Generally is used to wait the **INPUT PARAMETERS** in a **FORM**.
The **RETURN VALUE** (different to -1) can be read by Gcode

Syntax

EXD.OPEN_DLL *DLL_NAME, &$VAL,&:ADDR,&100,@$RET,@$RET1

DLL_NAME	**DLL name without extension**
&	**INPUT parameters**
@	**OUPUT parameters**

Ex:

```
//$VAR FIRST INPUT VALUE (127.14)
//:5000 SECOND INPUT VALUE (37.18)
//500.14 THIRD INPUT VALUE
//$RET FIRST OUTPUT VALUE
//$RET1 SECOND OUTPUT VALUE
//$RET2 THIRD OUTPUT VALUE
$VAR=127.14
:5000=37.18
EXD.OPEN_DLL *TESTUSDLL,&$VAR,&:5000,&500.14,@$RET,@$RET1,@$RET2 //OPEN
EXD.SYNK_DLL *TESTUSDLL,@$SYNCKRET //WAIT SYNK
IF $SYNCKRET=1 // IF SYNK = 1 END
   END_PROGRAM
END_IF
$_LOOP=:100
$COUNT=0
LOOP $_LOOP
   F[:101]
   G1X250Y250Z-50A150B50
   G62
   F[:101]
   X0Y0Z0A0B0
   G62
   $COUNT++
   EXD.CALL_DLL *TESTUSDLL,&$COUNT // CALL FOR UPDATE FORM VALUES
END_LOOP
EXD.SYNK_DLL *TESTUSDLL,@$SYNCKRET // SYNK WAIT END
EXD.CLOSE_DLL *TESTUSDLL // CLOSE
```

2) EXD.CALL_DLL

Dll Call.
Generally used for update Dll parameters.
Is invoked the function public bool **CallFunc**(double[] **ParInput**, out double[] **ValOut**)

Syntax

EXD.CALL_DLL *DLL_NAME, &$VAL,&:ADDR,&100,@$RET,@$RET1

DLL_NAME	**DLL name without extension**
&	**INPUT parameters**
@	**OUPUT parameters**

Ex:

```
EXD.CALL_DLL *TESTUSDLL,&$COUNT // CALL FOR UPDATE FORM VALUES
```

3) EXD.CLOSE_DLL

Dll close.
Is invoked the function public void **Close**()

Syntax

EXD.CLOSE_DLL *DLL_NAM

DLL_NAME	**DLL name without extension**

Ex:

```
EXD.CLOSE_DLL *TESTUSDLL
```

4) EXD.SYNK_DLL

Gcode synchronization from Dll
Is used for wait some operations in the DLL , like to Data Input.
Is invoked the property public int **Synk** { get; set; }
For enable the **SYNK**, the **DLL** must write Synk property !=-1

Syntax

EXD.SYNK_DLL *DLL_NAM,@SYNKRET

DLL_NAME	**DLL name without extension**
SYNKRET	**Return value from Synk (the value is always different to -1) NOT MANDATORY**

Ex:

```
EXD.SYNK_DLL *TESTUSDLL,@$SYNKRET // SYNK WAIT IN $SYNKRET THE VALUE OF PROPERTY Synk
```

5) EXD.STL_SETVIS

Allow to set the visibility status of STL component in the **REALMACHINE** simulation.
The component is called by **NAME** defined in the ***MachineSettings.xml***

Syntax

EXD.STL_SETVIS "STLNAME" Xvis

STLNAME **STL component NAME**

Xvis **State of visibility**

X0 **Hidden**

X1 **Visible**

Ex:

$VIS=0

EXD.STL_SETVIS **"STLPART1" $VIS** // HIDE COMPONENT STLPART1

6) EXD.STL_READVIS

Allow to read the visibility of STL component in the **REALMACHINE** simulation.
The component is called by **NAME** defined in the ***MachineSettings.xml***

Syntax

EXD.STL_READVIS "STLNAME" $STATE

STLNAME **STL component NAME**

$STATE **Return Variable of State visibility**

$STATE =0 **Hidden**

$STATE =1 **Visible**

Ex:

EXD.STL_READVIS **"STLPART1" $STATE** // READ STATUS COMPONENT STLPART1

7) EXD.STL_SETVISIDX

Allow to set the visibility status of STL component in the **REALMACHINE** simulation.
The component is called by **ADDRESS** defined in the ***MachineSettings.xml***

Syntax

EXD.STL_SETVISIDX ADDR VIS

ADDR **Address of STL component VARIABLE or NUMERIC VALUE**

VIS **State of STL VARIABLE or NUMERIC VALUE**

0 **Hidden**

1 **Visible**

Ex:

$VIS=0

$ADDR=5

EXD.STL_SETVISIDX **$ADDR $VIS** // HIDE COMPONENT ADDR 5
EXD.STL_SETVISIDX **7 0** // HIDE COMPONENT ADDR 7

8) EXD.STL_READVISIDX

Allow to read the visibility of STL component in the **REALMACHINE** simulation.
The component is called by **ADDRESS** defined in the ***MachineSettings.xml***

Syntax

EXD.STL_READVISIDX ADDR $STATE

ADDR **Address of STL component VARIABLE or NUMERIC VALUE**

$STATE **Return Variable of State visibility**

$STATE =0 **Hidden**

$STATE =1 **Visible**

Ex:

```
EXD.STL_READVISIDX 5 $STATE     // READ STATUS COMPONENT ADDR 5
```

RULES OF US DLL

The Us DLL must follow some **RULES**.
ISO US generate a **EXAMPLE** in the folder **Us_Dll-> ExampleLink.txt** this contains a LINK for download a complete C# project.

RULSE:

namespace	Must be **UsDll**
Class	Must be **UsDll and derivate the INTERFACE UsWork.IsoUs.IUsDll** **public class UsDll:UsWork.IsoUs.IUsDll**

Methods and Proprties are described in **EXD.OPEN_DLL**

DLL EXECUTION ERRORS
From Gcode will be generate an **ALARM** "**ERROR EXECUTION DLL**".
For understand better the **ERROR** type, IsoUs will write (in the same folder of IsoUs) a file **"_UsDll.Err"**
This contains more informations about the errors.
Is possible to use the Visual Studio Debug

USDLL EXAMPLE IN VISUAL STUDIO C#

Complete project https://www.promax.it/file_download/ZIP/TestUsDll.zip

UsDll Class

```
using System.Windows;
using TestUsDll;


namespace UsDll
{
   public class UsDll:UsWork.IsoUs.IUsDll
   {
      //Gcode IsoUs for Test
      /*
         $VAR=127.14
         :5000=37.18
         EXD.OPEN_DLL *TESTUSDLL,&$VAR,&:5000,&500.14,@$RET,@$RET1,@$RET2
         EXD.SYNK_DLL *TESTUSDLL,@$SYNCKRET
         IF $SYNCKRET=1
                  END_PROGRAM
         END_IF
         $_LOOP=:100
         $COUNT=0
         LOOP $_LOOP
                  F[:101]
                  G1X250Y250Z-50A150B50
                  G62
                  F[:101]
                  X0Y0Z0A0B0
                  G62
                  $COUNT++
                  EXD.CALL_DLL *TESTUSDLL,&$COUNT
         END_LOOP
         EXD.SYNK_DLL *TESTUSDLL,@$SYNCKRET
         EXD.CLOSE_DLL *TESTUSDLL
      */
      internal UsWork.IsoUs _MyIsoUs;    // instance of UsWork
      TestW _TestW;             //Test Window
      internal int Loop;        //Loop Number
      #region Events
      /// <summary>
      /// Values changed
      /// Update form labels
      /// </summary>
      /// <param name="sender"></param>
      /// <param name="e"></param>
      void _MyIsoUs_AxesDemandValueChanged(object sender, UsWork.AxesValueArg e)
      {
         _TestW.LblValX.Content = e.Format[0];
         _TestW.LblValY.Content = e.Format[1];
         _TestW.LblValZ.Content = e.Format[2];
         _TestW.LblValA.Content = e.Format[3];
         _TestW.LblValB.Content = e.Format[4];
      }
```

```
    #endregion Events

#region IUsDll
    /// <summary>
    /// IsoUs Instance of current Framwork running
    /// Load Dll
    /// ParInput Input parameters
    /// ValOut Output values
    /// </summary>
    /// <param name="IsoUs"></param>
    /// <param name="ParInput"></param>
    /// <param name="ValOut"></param>
    /// <returns></returns>
    public bool Load(UsWork.IsoUs IsoUs, double[] ParInput, out double[] ValOut)
    {
      Synk = -1;
      ValOut = new double[ParInput.Length]; // ValOut the same length of ParInput
      for (int n = 0; n < ParInput.Length; n++)
        ValOut[n] = ParInput[n] * 2;    // Copy values * 2
      _MyIsoUs = IsoUs;   // save IsoUs instance
      _SetEvents();      // Set IsoUs events
      _TestW = new TestW(this);   // test Window
      _TestW.Owner = (Window)_MyIsoUs.Tag;    // Owner
      _TestW.Show();  // Show
      return true;    // return OK
    }
    /// <summary>
    /// Call Function
    /// ParInput Input parameters
    /// ValOut Output values
    /// </summary>
    /// <param name="ParInput"></param>
    /// <param name="ValOut"></param>
    /// <returns></returns>
    public bool CallFunc(double[] ParInput, out double[] ValOut)
    {
      ValOut = null; // no ValOut
      int _Val = (int)ParInput[0]; // Get ParInput[0]
      if (_Val == Loop) // if cycle reached
      {
        _TestW.LblLoop.Content = "END PCS: " + _Val.ToString(); // update label END PCS
        _TestW.BtnExit.IsEnabled = true; // enable Exit button
      }
      else
        _TestW.LblLoop.Content = _Val.ToString(); //Update PCS Number
      return true;
    }
    /// <summary>
    /// Close Dll
    /// </summary>
    public void Close()
    {
      _TestW.Close(); // Close Window
    }
```

```
        /// <summary>
        /// Synk Dll
        /// return SYNK value if not -1
        /// </summary>
        public int Synk { get; set; }

        #endregion IUsDll
        /// <summary>
        /// Stop the current Gcode execution
        /// called by form button
        /// </summary>
        internal void Stop()
        {
            _MyIsoUs.UsGcodeRun.StopGcode(); // Stop Gcode
        }
        /// <summary>
        /// Set events
        /// </summary>
        void _SetEvents()
        {
            _MyIsoUs.AxesDemandValueChanged += _MyIsoUs_AxesDemandValueChanged; // axes values changed
            _MyIsoUs.MyMaster.ResetEvent(); // reset events for update labels
        }
    }
}
```

TestW.xaml

```
<Window x:Name="TestW1" x:Class="TestUsDll.TestW"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    xmlns:local="clr-namespace:TestUsDll"
    mc:Ignorable="d"
    Title="TestW" Height="400" Width="400" WindowStyle="None">
  <Grid x:Name="MyGrid" >
    <Grid.RowDefinitions>
      <RowDefinition Height="*"/>
      <RowDefinition Height="*"/>
      <RowDefinition Height="*"/>
      <RowDefinition Height="*"/>
      <RowDefinition Height="*"/>
      <RowDefinition Height="*"/>
      <RowDefinition Height="*"/>
      <RowDefinition Height="*"/>
    </Grid.RowDefinitions>
    <Grid Grid.Row="0">
      <Grid.ColumnDefinitions>
        <ColumnDefinition Width="0.7*"/>
        <ColumnDefinition Width="0.8*"/>
        <ColumnDefinition Width="0.7*"/>
        <ColumnDefinition Width="*"/>
      </Grid.ColumnDefinitions>
      <Label x:Name="LblNrLoop" Tag="2" Grid.Column="0" Content="NrLoop" HorizontalAlignment="Center"
Height="40" FontSize="18" VerticalAlignment="Center" />
      <TextBox x:Name="TxtLoop"  Tag="0" Grid.Column="1" HorizontalAlignment="Center" Height="30"  Text="50"
FontSize="20" VerticalAlignment="Center" Width="90" TextAlignment="Center" PreviewTextInput="TxtLoop_Preview-
TextInput" />
      <Label x:Name="LblCurrentLoop" Grid.Column="2" Tag="2" Content="Current" HorizontalAlignment="Center"
Height="40" FontSize="18" VerticalAlignment="Center"  />
      <Label x:Name="LblLoop" Tag="2" Grid.Column="3" FontSize="18" Content="0" HorizontalAlignment="Left"
Height="40" HorizontalContentAlignment="Center" VerticalContentAlignment="Center" VerticalAlignment="Center"
/>
    </Grid>
    <Grid Grid.Row="1">
      <Grid.ColumnDefinitions>
        <ColumnDefinition Width="*"/>
        <ColumnDefinition Width="*"/>
        <ColumnDefinition Width="*"/>
      </Grid.ColumnDefinitions>
      <Label x:Name="LblFeed" Tag="2" Grid.Column="0" Content="Feed" HorizontalAlignment="Center"
Height="40" FontSize="18" VerticalAlignment="Center" Width="50" />
      <TextBox x:Name="TxtF" Tag="0" Grid.Column="1" HorizontalAlignment="Center" Height="30" FontSize="20"
Text="1" VerticalAlignment="Center" Width="100" TextAlignment="Center" PreviewTextInput="TxtF_PreviewTextIn-
put"/>
      <Button x:Name="BtnSetF" Grid.Column="2" Tag="1" Content="Set Feed" ToolTip="Set Work Feed" Horizonta-
lAlignment="Center" FontSize="16" Height="30" VerticalAlignment="Center" Width="80" Click="BtnSetF_Click"/>
    </Grid>
    <Grid Grid.Row="2">
```

```
        <Grid.ColumnDefinitions>
          <ColumnDefinition Width="*"/>
          <ColumnDefinition Width="5*"/>
        </Grid.ColumnDefinitions>
        <Border  x:Name="BLblX" Grid.Column="0" Tag="3"  Height="40" Width="40"  VerticalAlignment="Center"
HorizontalAlignment="Center"  >
          <Label x:Name="LblX" Tag="2" Content="X"  Height="40" FontSize="24" HorizontalContentAlignment="Cen-
ter" VerticalAlignment="Center" Width="40"/>
        </Border>
        <Border  x:Name="BLblValX" Grid.Column="1" Tag="4"  Height="40" Width="200"  VerticalAlignment="Center"
HorizontalAlignment="Center">
          <Label x:Name="LblValX" Tag="2" Content="..."  Height="40" FontSize="24" HorizontalContentAlign-
ment="Center" VerticalAlignment="Center" />
        </Border>
      </Grid>
      <Grid Grid.Row="3">
        <Grid.ColumnDefinitions>
          <ColumnDefinition Width="*"/>
          <ColumnDefinition Width="5*"/>
        </Grid.ColumnDefinitions>
        <Border  x:Name="BLblY" Grid.Column="0" Tag="3"  Height="40" Width="40"  VerticalAlignment="Center"
HorizontalAlignment="Center" >
          <Label x:Name="LblY" Tag="2" Content="Y"  Height="40" FontSize="24" HorizontalContentAlignment="Cen-
ter" VerticalAlignment="Center" Width="40"/>
        </Border>
        <Border  x:Name="BLblValY" Grid.Column="1" Tag="4" Height="40" Width="200"  VerticalAlignment="Center"
HorizontalAlignment="Center" >
          <Label x:Name="LblValY" Tag="2" Content="..."  Height="40" FontSize="24" HorizontalContentAlign-
ment="Center" VerticalAlignment="Center" />
        </Border>
      </Grid>

      <Grid Grid.Row="4">
        <Grid.ColumnDefinitions>
          <ColumnDefinition Width="*"/>
          <ColumnDefinition Width="5*"/>
        </Grid.ColumnDefinitions>
        <Border  x:Name="BLblZ" Tag="3" Grid.Column="0" Height="40" Width="40"  VerticalAlignment="Center"  Hor-
izontalAlignment="Center" >
          <Label x:Name="LblZ" Tag="2" Content="Z"  Height="40" FontSize="24" HorizontalContentAlignment="Cen-
ter" VerticalAlignment="Center" Width="40"/>
        </Border>
        <Border  x:Name="BLblValZ"  Grid.Column="1" Tag="4" Height="40" Width="200"  VerticalAlignment="Center"
HorizontalAlignment="Center" >
          <Label x:Name="LblValZ" Tag="2" Content="..."  Height="40" FontSize="24" HorizontalContentAlign-
ment="Center" VerticalAlignment="Center" />
        </Border>
      </Grid>
      <Grid Grid.Row="5">
        <Grid.ColumnDefinitions>
          <ColumnDefinition Width="*"/>
          <ColumnDefinition Width="5*"/>
        </Grid.ColumnDefinitions>
```

```
        <Border  x:Name="BLblA" Grid.Column="0" Tag="3" Height="40" Width="40"  VerticalAlignment="Center"  Hor-
izontalAlignment="Center">
          <Label x:Name="LblA" Tag="2" Content="A"  Height="40" FontSize="24" HorizontalContentAlignment="Cen-
ter" VerticalAlignment="Center" Width="40"/>
        </Border>
        <Border  x:Name="BLblValA" Grid.Column="1" Tag="4" Height="40" Width="200"  VerticalAlignment="Center"
HorizontalAlignment="Center" Grid.Row="1">
          <Label x:Name="LblValA" Tag="2" Content="..."  Height="40" FontSize="24" HorizontalContentAlign-
ment="Center" VerticalAlignment="Center" />
        </Border>
      </Grid>
      <Grid Grid.Row="6">
        <Grid.ColumnDefinitions>
          <ColumnDefinition Width="*"/>
          <ColumnDefinition Width="5*"/>
        </Grid.ColumnDefinitions>
        <Border  x:Name="BLblB" Grid.Column="0" Tag="3"  Height="40" Width="40"  VerticalAlignment="Center"
HorizontalAlignment="Center" >
          <Label x:Name="LblB" Tag="2" Content="B"  Height="40" FontSize="24" HorizontalContentAlignment="Cen-
ter" VerticalAlignment="Center" Width="40"/>
        </Border>
        <Border  x:Name="BLblValB" Grid.Column="1" Tag="4" Height="40" Width="200"  VerticalAlignment="Center"
HorizontalAlignment="Center" >
          <Label x:Name="LblValB" Tag="2" Content="..."  Height="40" FontSize="24" HorizontalContentAlign-
ment="Center" VerticalAlignment="Center" />
        </Border>
      </Grid>
      <Grid Grid.Row="7">
        <Grid.ColumnDefinitions>
          <ColumnDefinition Width="*"/>
          <ColumnDefinition Width="*"/>
          <ColumnDefinition Width="*"/>
        </Grid.ColumnDefinitions>
        <Button x:Name="BtnStop" Grid.Column="0" Tag="1" IsEnabled="False" Content="STOP" ToolTip="Stop Gcode
and Exit" HorizontalAlignment="Center" Height="30" VerticalAlignment="Center" Width="60" Click="BtnStop_Click"/>
        <Button x:Name="BtnRun" Grid.Column="1" Tag="1" Content="Run" ToolTip="Run Gcode" HorizontalAlign-
ment="Center" Height="30" VerticalAlignment="Center" Width="60" Click="BtnRun_Click"/>
        <Button x:Name="BtnExit" Grid.Column="2" Tag="1" ToolTip="Exit" Content="Exit" HorizontalAlignment="Cen-
ter" Height="30"  VerticalAlignment="Center" Width="60" Click="BtnExit_Click" />
      </Grid>
    </Grid>
</Window>
```

TestW.xaml.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Shapes;

namespace TestUsDll
{

   public partial class TestW : Window
   {
      UsDll.UsDll _MyUsDll;
      //Style components
      /*
      MainGridStyle         -> Grid
      PlugInButton          -> Button
      PlugInButtonSelect    -> Button
      BtnVisType            -> Button
      ButtonCommandsStyle   -> Button
      TextBoxStyle          -> TextBox
      LabelStyle            -> Label
      LabelSelector         -> Label
      LabelSelector1        -> Label
      LabelAbsStyle         -> Label
      LabelMonitorStyle     -> Label
      LabelCommandsStyle    -> Label
      LabelNameAxesStyle    -> Label
      LabelFlashStyle       -> Label
      LabelHomeStyle        -> Label
      LabelInc              -> Label
      UsPreviewCheckBox     -> CheckBox
      CheckBoxStyle         -> CheckBox
      UsContextMenuStyle    -> ContextMenu
      UsGridViewHeaderStyle -> DataGridColumnHeader
      UsDataGridCellStyle   -> DataGridCell
      UsProgressBar         -> ProgressBar
      TabItemStyle          -> TabItem
      BorderSelector        -> Border
      BorderSelector1       -> Border
      BorderStyle           -> Border
      BorderAbsStyle        -> Border
      BorderMonitorStyle    -> Border
      BorderNameAxesStyle   -> Border
      BorderInc             -> Border
```

```
    SliderThumbStyle      -> Thumb
    NotifyListViewFont    -> ListViewItem
    NotifyListViewStyle   -> ListView
    */
    public TestW(UsDll.UsDll MyUsDll)
    {
      _MyUsDll = MyUsDll;
      InitializeComponent();
      _SetStyle(); // Set Style from IsoUs
    }
    #region Events
    /// <summary>
    /// Move Window
    /// </summary>
    /// <param name="e"></param>
    protected override void OnMouseLeftButtonDown(MouseButtonEventArgs e)
    {
      base.OnMouseLeftButtonDown(e);
      DragMove();
    }
    /// <summary>
    /// Run Gcode SYNK OK
    /// </summary>
    /// <param name="sender"></param>
    /// <param name="e"></param>
    void BtnRun_Click(object sender, RoutedEventArgs e)
    {
      _MyUsDll.Loop = int.Parse(TxtLoop.Text); // Loop Value
      _MyUsDll._MyIsoUs.MyMaster.GetLink.MyCpu.VarCpu[0].VarIsoNs[100] = _MyUsDll.Loop; // Write IsoUs Var
Loop
      _MyUsDll._MyIsoUs.MyMaster.GetLink.MyCpu.VarCpu[0].VarIsoNs[101] = double.Parse(TxtF.Text,
      _MyUsDll._MyIsoUs.NewProvider); // Write FEED Value
      _MyUsDll.Synk = 0; // Synk Ok
      BtnStop.IsEnabled = true;
      BtnExit.IsEnabled = false;
      BtnRun.IsEnabled = false;

    }
    /// <summary>
    /// Exit
    /// </summary>
    /// <param name="sender"></param>
    /// <param name="e"></param>
    void BtnExit_Click(object sender, RoutedEventArgs e)
    {
      _MyUsDll.Synk = 1; // SYNK EXIT
    }
    /// <summary>
    /// Stop Gcode
    /// </summary>
    /// <param name="sender"></param>
    /// <param name="e"></param>
    void BtnStop_Click(object sender, RoutedEventArgs e)
    {
```

```
    _MyUsDll.Stop();
}
/// <summary>
/// Set Feed
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
void BtnSetF_Click(object sender, RoutedEventArgs e)
{
      _MyUsDll._MyIsoUs.MyMaster.GetLink.MyCpu.VarCpu[0].VarIsoNs[101] = double.Parse(TxtF.Text,
      _MyUsDll._MyIsoUs.NewProvider); // Set Gcode Feed
}
/// <summary>
/// Feed Preview Text Input
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
void TxtF_PreviewTextInput(object sender, TextCompositionEventArgs e)
{
    TextBox _Txt = (TextBox)sender;
    _CheckIsNumeric(e, _Txt.Text, true, false);
}
/// <summary>
/// Loop Preview Text Input
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
void TxtLoop_PreviewTextInput(object sender, TextCompositionEventArgs e)
{
    TextBox _Txt = (TextBox)sender;
    _CheckIsNumeric(e, _Txt.Text, false, false);
}
#endregion Events
/// <summary>
/// Check if string contains only numerical value
/// </summary>
/// <param name="e">TextCompositionEventArgs PreviewTextInput Argument</param>
/// <param name="_Text">String Text for check</param>
/// <param name="_DecimalPoint">True accept decimal point</param>
/// <param name="_NegativeValue">True accept negative value</param>
void _CheckIsNumeric(TextCompositionEventArgs e, string _Text, bool _DecimalPoint, bool _NegativeValue)
{
    if (!_DecimalPoint)
        _CheckIsNumeric(e, _Text, 0, _NegativeValue);
    else
        _CheckIsNumeric(e, _Text, 20, _NegativeValue);
}
/// <summary>
/// Check if string contains only numerical value
/// </summary>
/// <param name="e">TextCompositionEventArgs PreviewTextInput Argument</param>
/// <param name="_Text">String Text for check</param>
/// <param name="_NrDecimal">Int32 Number of Decimal Digit</param>
/// <param name="_NegativeValue">True accept negative value</param>
```

```
void _CheckIsNumeric(TextCompositionEventArgs e, string _Text, Int32 _NrDecimal, bool _NegativeValue)
{
   int result;
   if (e.Text == "-")
   {
      if (!_NegativeValue)
         e.Handled = true;
      if (_Text == "")
         return;
      if (_Text.Contains('-'))
      {
         e.Handled = true;
         return;
      }
      TextBox _Txt = (TextBox)e.OriginalSource;
      if (_Txt.CaretIndex == 0)
         return;
      e.Handled = true;
      return;
   }
   if (e.Text == ".")
   {
      if (_NrDecimal <= 0)
         e.Handled = true;
      if (_Text.Contains("."))
         e.Handled = true;
      return;
   }
   if (!(int.TryParse(e.Text, out result)))
      e.Handled = true;
   // Check Decimal after dot
   Int32 AfterDot;
   Int32 PosDot;
   PosDot = _Text.IndexOf('.');
   if (PosDot == -1)
      return;
   // get char after dot
   AfterDot = _Text.Length - PosDot - 1;
   TextBox _MText = (TextBox)e.Source;

   if (AfterDot >= _NrDecimal && _MText.CaretIndex > PosDot)
      e.Handled = true;
}
/// <summary>
/// Set STyle
/// </summary>
void _SetStyle()
{
   _SetStyle(MyGrid);
}
```

```
/// <summary>
/// Set Styles
/// Set Components Styles like to IsoUs
/// </summary>
void _SetStyle(Grid _Grid)
{
   for(int n=0;n<_Grid.Children.Count;n++)
   {
      int _Tag = -1;
      object _Obj = _Grid.Children[n];
      if(_Obj is Grid)
      {
         Grid _Grd = (Grid)_Obj;
         _Grd.Style = _MyUsDll._MyIsoUs.UsStylesProvider.GetStyle("MainGridStyle");
         _SetStyle(_Grd);
      }
      Label _Lbl = null;
      Border _Brd = null;
      Button _Btn = null;
      TextBox _Txt = null;
      if (_Obj is Label)
      {
         _Lbl = (Label)_Obj;
         _Tag = int.Parse(_Lbl.Tag.ToString());
      }
      if (_Obj is Button)
      {
         _Btn = (Button)_Obj;
         _Tag = int.Parse(_Btn.Tag.ToString());
      }
      if (_Obj is TextBox)
      {
         _Txt = (TextBox)_Obj;
         _Tag = int.Parse(_Txt.Tag.ToString());
      }
      if (_Obj is Border)
      {
         _Brd = (Border)_Obj;
         object _Obj1 = _Brd.Child;
         _Lbl = (Label)_Obj1;
         _Tag = int.Parse(_Brd.Tag.ToString());
      }
      switch (_Tag)
      {
         case 0:
            _Txt.Style = _MyUsDll._MyIsoUs.UsStylesProvider.GetStyle("TextBoxStyle");
            break;
         case 1:
            _Btn.Style = BtnExit.Style = _MyUsDll._MyIsoUs.UsStylesProvider.GetStyle("PlugInButton");
             break;
         case 2:
            _Lbl.Style = _MyUsDll._MyIsoUs.UsStylesProvider.GetStyle("LabelNameAxesStyle");
            break;
         case 3:
```

```
                _Brd.Style = _MyUsDll._MyIsoUs.UsStylesProvider.GetStyle("BorderNameAxesStyle");
                _Lbl.Style = _MyUsDll._MyIsoUs.UsStylesProvider.GetStyle("LabelNameAxesStyle");
                break;
            case 4:
                _Brd.Style = _MyUsDll._MyIsoUs.UsStylesProvider.GetStyle("BorderStyle");
                _Lbl.Style = _MyUsDll._MyIsoUs.UsStylesProvider.GetStyle("LabelAbsStyle");
                break;
        }
      }
    }
  }
}
```

9 NSFORMS INSTRUCTIONS

LIB. Instructions are used for open the dialog windows (FORM) by Gcode.

This allows to prepare or condition the Gcode from **OBJECTS** insert in the **FORM**:

- ***BUTTON***
- ***LABEL***
- ***CHECK BOX***
- ***INPUT***
- ***TEXT BOX***
- ***COMBO BOX***
- ***SLIDER***

All objects can **READ** or **WRITE** Gcode variables.

All text messages used by Objects, if begin by prefix ***"@_"*** can be get from dictionary ***"UsLanguages.lang"*** in the folder ***Languages***.

For a easy and visual programming of the **FORM**, is possible use the PlugIn **UsFormManager**.

9.1 LIB.MESSAGE

Show a MessageBox with specifed mode.
The program is suspended up to executed action in MessageBox

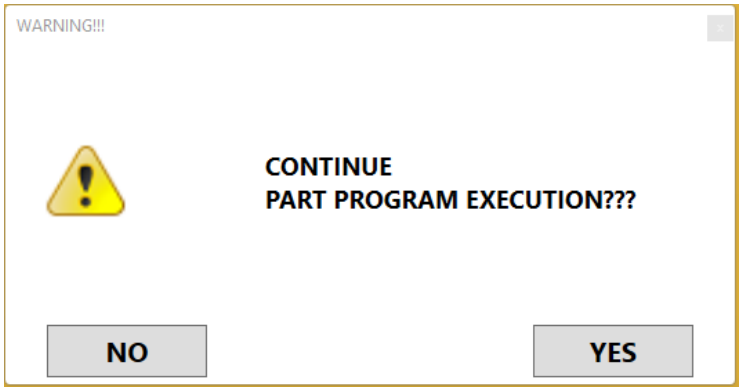
Syntax
LIB.MESSAGE "TEXT" "CAPTION" BUTTON ICON

TEXT
Body Message Text **\N** line feed carriage return
CAPTION
Message Title
BUTTON (value or variable)
0 OK
1 YES - NO
ICON (value or variable)
0 ERROR
1 WARNING
2 INFORMATION
3 QUESTION
4 NONE

Ex:

```
LIB.MESSAGE "CONTINUE\NPART PROGRAM EXECUTION???" "WARNING!!!" 1 1
$BTN=$[X13] // READ THE BUTTON PRESSED
IF $BTN=2
    END_PROGRAM // END PROGRAM IF PRESSED NO
END_IF
```

The LIB.MESSAGE instruction returned the code button pressed in the variable **$[X13]**.
this contain
0 → Button **OK** pressed
1 → Button **SI** pressed
2 → Button **NO** pressed



9.2 LIB.SHOWFORM

Show a NsForm. The properties must be set by ***LIB.FORMPROP*** instruction.
The NsForm is in any case closed when the actual Part Program is finished.
Only one NsForm is displayed
Syntax
LIB.SHOWFORM

9.3 LIB.CLOSEFORM

Close a NsForm previously opened .
Syntax
LIB.CLOSEFORM

9.4 LIB.FORMPROP

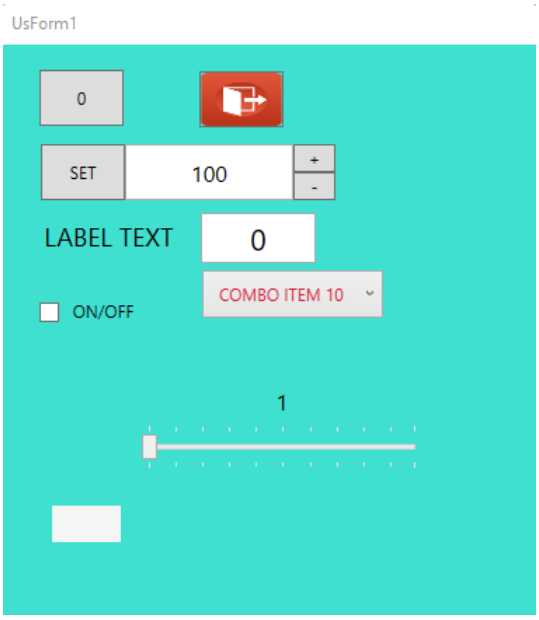
Set the NsForm properties. The properties can be setted before a **LIB.SHOWFORM**
Syntax
LIB.FORMPROP "PROPNAME" PROPVALUE (value or variable)
PROPNAME (Property Name TEXT) :

VISIBLE
PropValue 0 (or < ZERO) NsForm Invisible
PropValue 1 (or > ZERO) NsForm Visible
WIDTH
PropValue > zero set **WIDTH** NSFORM
HEIGHT
PropValue > zero set **HEIGTH** NSFORM
LEFT
PropValue > zero Set the **LEFT** position
TOP
PropValue > zero Set the **TOP** position
BACKCLOR
PropValue Value 0 to 140.Set BACKCOLOR (see color table)
STARTPOSITION (must be set before SHOWFORM)
PropValue 0 (o r< di ZERO) Use the **LEFT** e **TOP** position
PropValue 1 (or > di ZERO) NsForm is centered on screen
WINDOWSTATE
PropValue 0 (or < di ZERO) Use **WIDTH** e **HEIGHT** dimensions
PropValue 1 (or > di ZERO) Use a Max dimension
USESTYLE
PropValue 0 Use BackColor and ForeColor for form and all objects
PropValue 1 use IsoUs Style for form and all objects

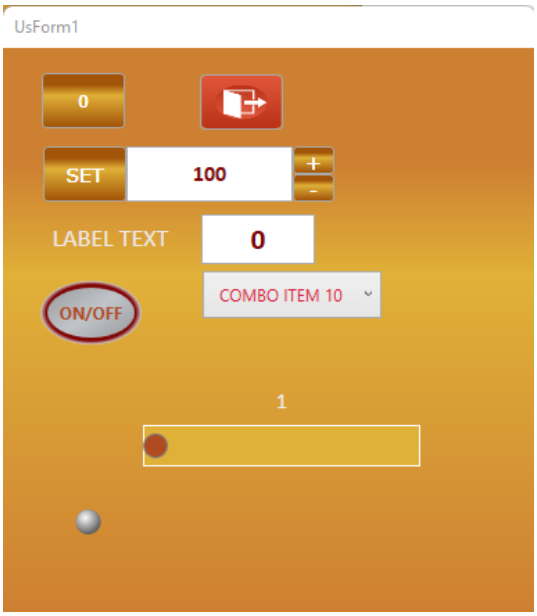
Ex:

```
LIB.FORMPROP "WIDTH" 400
LIB.FORMPROP "HEIGHT" 450
LIB.FORMPROP "LEFT" 748
LIB.FORMPROP "TOP" 293
LIB.FORMPROP "STARTPOSITION" 1
LIB.FORMPROP "WINDOWSTATE" 0
LIB.FORMPROP "BACKCOLOR" 134
LIB.FORMTEXT "UsForm1"
LIB.ADDLABEL "Lbl0"
LIB.LABELPROP "Lbl0" "WIDTH" 111
LIB.LABELPROP "Lbl0" "HEIGHT" 34
LIB.LABELPROP "Lbl0" "LEFT" 22
LIB.LABELPROP "Lbl0" "TOP" 120
LIB.LABELPROP "Lbl0" "DIMFONT" 18
LIB.LABELPROP "Lbl0" "ALIGN" 1
LIB.LABELPROP "Lbl0" "VISIBLE" 1
LIB.LABELPROP "Lbl0" "ENABLED" 1
LIB.LABELPROP "Lbl0" "COLORON" 113
LIB.LABELPROP "Lbl0" "COLOROFF" 138
LIB.LABELTEXT "Lbl0" "LABEL TEXT"
LIB.ADDITEXT "Txt0"
LIB.ITEXTPROP "Txt0" "WIDTH" 82
LIB.ITEXTPROP "Txt0" "HEIGHT" 36
LIB.ITEXTPROP "Txt0" "LEFT" 143
LIB.ITEXTPROP "Txt0" "TOP" 120
LIB.ITEXTPROP "Txt0" "DIMFONT" 21
LIB.ITEXTPROP "Txt0" "ALIGN" 1
LIB.ITEXTPROP "Txt0" "VISIBLE" 1
LIB.ITEXTPROP "Txt0" "ENABLED" 1
LIB.ITEXTPROP "Txt0" "AUTOSET" 1
LIB.ITEXTPROP "Txt0" "NDECIMAL" 3
LIB.ITEXTPROP "Txt0" "MIN" -2000000000
LIB.ITEXTPROP "Txt0" "MAX" 2000000000
LIB.ITEXTPROP "Txt0" "DEST" $[K15]
LIB.ITEXTSETVALUE "Txt0" $[K15]
LIB.ADDCHECK "Check0"
LIB.CHECKPROP "Check0" "WIDTH" 73
LIB.CHECKPROP "Check0" "HEIGHT" 46
LIB.CHECKPROP "Check0" "LEFT" 27
LIB.CHECKPROP "Check0" "TOP" 168
LIB.CHECKPROP "Check0" "DIMFONT" 12
LIB.CHECKPROP "Check0" "ALIGN" 1
LIB.CHECKPROP "Check0" "VISIBLE" 1
LIB.CHECKPROP "Check0" "AUTOSET" 1
LIB.CHECKPROP "Check0" "ENABLED" 1
LIB.CHECKSETVALUE "Check0" $[O10]
LIB.CHECKPROP "Check0" "DEST" $[O10]
LIB.CHECKTEXT "Check0" "ON/OFF"
LIB.ADDCOMBO "Combo0"
LIB.COMBOPROP "Combo0" "WIDTH" 129
LIB.COMBOPROP "Combo0" "HEIGHT" 34
LIB.COMBOPROP "Combo0" "LEFT" 144
LIB.COMBOPROP "Combo0" "TOP" 161
```

LIB.FORMPROP "USESTYLE" 0



LIB.FORMPROP "USESTYLE" 1



```
L LIB.COMBOPROP "Combo0" "ALIGN" 1
  LIB.COMBOPROP "Combo0" "VISIBLE" 1
  LIB.COMBOPROP "Combo0" "ENABLED" 1
  LIB.COMBOPROP "Combo0" "AUTOSET" 1
  LIB.COMBOPROP "Combo0" "FOREGROUND" 19
  LIB.COMBOPROP "Combo0" "DEST" $VAR1
  LIB.COMBOSETVALUE "Combo0" 2
  LIB.COMBOITEM "Combo0" "COMBO ITEM 10" 1
  LIB.COMBOITEM "Combo0" "COMBO ITEM 20" 2
  LIB.COMBOITEM "Combo0" "COMBO ITEM 30" 3
  LIB.ADDSLIDER "Slider0"
  LIB.SLIDERPROP "Slider0" "WIDTH" 200
  LIB.SLIDERPROP "Slider0" "HEIGHT" 35
  LIB.SLIDERPROP "Slider0" "LEFT" 101
  LIB.SLIDERPROP "Slider0" "TOP" 237
  LIB.SLIDERPROP "Slider0" "DIMFONT" 16
  LIB.SLIDERPROP "Slider0" "VISIBLE" 1
  LIB.SLIDERPROP "Slider0" "VISLABEL" 1
  LIB.SLIDERPROP "Slider0" "ENABLED" 1
  LIB.SLIDERPROP "Slider0" "AUTOSET" 1
  LIB.SLIDERPROP "Slider0" "NDECIMAL" 3
  LIB.SLIDERPROP "Slider0" "TICKP" 3
  LIB.SLIDERPROP "Slider0" "TICKF" 10
  LIB.SLIDERPROP "Slider0" "MIN" 1
  LIB.SLIDERPROP "Slider0" "MAX" 100
  LIB.SLIDERPROP "Slider0" "LARGE" 10
  LIB.SLIDERPROP "Slider0" "SMALL" 1
  LIB.SLIDERPROP "Slider0" "DEST" $PIPPO1
  LIB.SLIDERSETVALUE "Slider0" 0
  LIB.ADDBUTTON "Btn0"
  LIB.BUTTONPROP "Btn0" "WIDTH" 60
  LIB.BUTTONPROP "Btn0" "HEIGHT" 40
  LIB.BUTTONPROP "Btn0" "LEFT" 28
  LIB.BUTTONPROP "Btn0" "TOP" 18
  LIB.BUTTONPROP "Btn0" "DIMFONT" 12
  LIB.BUTTONPROP "Btn0" "ALIGN" 1
  LIB.BUTTONPROP "Btn0" "CLOSEFORM" 0
  LIB.BUTTONPROP "Btn0" "VISIBLE" 1
  LIB.BUTTONPROP "Btn0" "ENABLED" 1
  LIB.BUTTONPROP "Btn0" "INPUTUPDATE" 0
  LIB.BUTTONPROP "Btn0" "DEST" $VAR1
  LIB.BUTTONPRINT "Btn0" "I" @VART
  LIB.BUTTONPROP "Btn0" "COLORON" 113
  LIB.BUTTONPROP "Btn0" "COLOROFF" 138
  LIB.ADDINPUT "Input0"
  LIB.INPUTPROP "Input0" "WIDTH" 120
  LIB.INPUTPROP "Input0" "HEIGHT" 40
  LIB.INPUTPROP "Input0" "LEFT" 29
  LIB.INPUTPROP "Input0" "TOP" 71
  LIB.INPUTPROP "Input0" "DIMFONT" 16
  LIB.INPUTPROP "Input0" "ALIGN" 1
  LIB.INPUTPROP "Input0" "VISIBLE" 1
  LIB.INPUTPROP "Input0" "ENABLED" 1
  LIB.INPUTPROP "Input0" "AUTOSET" 1
  LIB.INPUTPROP "Input0" "NDECIMAL" 3
  LIB.INPUTPROP "Input0" "MIN" -2000000000
  LIB.INPUTPROP "Input0" "MAX" 2000000000
  LIB.INPUTPROP "Input0" "INCREMENT" 1
  LIB.INPUTPROP "Input0" "DEST" $PIPPO
  LIB.INPUTSETVALUE "Input0" 100
  LIB.ADDLABEL "Lbl1"
  LIB.LABELPROP "Lbl1" "WIDTH" 49
  LIB.LABELPROP "Lbl1" "HEIGHT" 26
  LIB.LABELPROP "Lbl1" "LEFT" 37
  LIB.LABELPROP "Lbl1" "TOP" 329
  LIB.LABELPROP "Lbl1" "DIMFONT" 18
  LIB.LABELPROP "Lbl1" "ALIGN" 1
  LIB.LABELPROP "Lbl1" "VISIBLE" 1
  LIB.LABELPROP "Lbl1" "ENABLED" 1
  LIB.LABELF "Lbl1" "B" $[O10]
  LIB.LABELPROP "Lbl1" "TIMER" 200
  LIB.LABELPROP "Lbl1" "COLORON" 113
  LIB.LABELPROP "Lbl1" "COLOROFF" 138
  LIB.ADDBUTTON "Btn1"
  LIB.BUTTONPROP "Btn1" "WIDTH" 60
  LIB.BUTTONPROP "Btn1" "HEIGHT" 40
  LIB.BUTTONPROP "Btn1" "LEFT" 142
  LIB.BUTTONPROP "Btn1" "TOP" 19
  LIB.BUTTONPROP "Btn1" "DIMFONT" 12
  LIB.BUTTONPROP "Btn1" "ALIGN" 1
  LIB.BUTTONPROP "Btn1" "CLOSEFORM" 1
  LIB.BUTTONPROP "Btn1" "VISIBLE" 1
  LIB.BUTTONPROP "Btn1" "ENABLED" 1
  LIB.BUTTONPROP "Btn1" "INPUTUPDATE" 0
  LIB.BUTTONPROP "Btn1" "COLORON" 113
  LIB.BUTTONPROP "Btn1" "COLOROFF" 138
  LIB.BUTTONTEXT "Btn1" "(@)EXIT"
  LIB.SHOWFORM
```

9.5 LIB.FORMTEXT

Set the NsForm Caption. The properties can be setted befor a LIB.SHOWFORM

Syntax
LIB.FORMTEXT "CAPTION"
CAPTION (TEXT) :
Text of Caption

Ex:
LIB.FORMTEXT "CAPTION"

9.6 LIB.ADDLABEL

Add a NsLabel to NsForm. The NaLabel properties can be setted befor a LIB.SHOWFORM

Syntax
LIB.ADDLABEL "NAME"
NAME (TEXT) :
Univocal Name of NsLabel Whitout special characters

Ex:
LIB.ADDLABEL "LBL"

9.7 LIB.LABELPROP

Set the NsLabel properties. The properties can be setted before a **LIB.SHOWFORM** but before, is necessary that the NsLabel is added to NsForm with **LIB.ADDLABEL**

Syntax
LIB.LABELPROP "LBLNAME" "PROPNAME" PROPVALUE (value or variable)
LBLNAME NsLabel Name
PROPNAME (Property Name TEXT) :

VISIBLE
PropValue 0 (< of ZERO) NsLabel Invisible
PropValue 1 (> of ZERO) NsLabel Visible (default)
AUTOSIZE
PropValue 0 (< of ZERO) NsLabel dimension by WHIDTH e HEIGHT
PropValue 1 (> of ZERO) NsLabel automatic dimension (default)
ENABLED
PropValue 0 (< of ZERO) NsLabel enabled (default)
PropValue 1 (> of ZERO) NsLabel disabled
WIDTH
PropValue > zero NSLabel Width (if autosize = 0)
HEIGHT
PropValue > zero NSLabel Height (if autosize = 0)
LEFT
PropValue > zero NsLabel Left position in NsForm
TOP
PropValue > zero NsLabel Top position in NsForm
DIMFONT
PropValue Value 6 to 100. Font Dimension

BACKCLOR

PropValue Value 0 to 140.Set BACKCOLOR (see color table)

FORECOLOR

PropValue Value 0 to 140.Set FORECOLOR (see color table)

COLORON

PropValue Value 0 to 140.Set color ON for ON/OFF format (LABELF – LABELPRINT)

COLOROFF

PropValue Value 0 to 140.Set color OFF for ON/OFF format (LABELF – LABELPRINT)

STYLE (default 0)

PropValue:

0 Normal **1** *Medium* ***2 Bold*** ***3 Ultra Bold***

ALIGN (default 0)

PropValue:

0 Middle Left **1** Middle Center **2** Middle Right
3 Top Left **4** Top Center **5** Top Right
6 Bottom Left **7** Bottom Center **8** Bottom Right

BORDER (default 0)

PropValue:

0 None **1** *Fixed Single* **2** *Fixed 3D*

TIMER (default 0)

PropValue > zero(millisecond) enable the print by **TIMER** for **LABELF**

Ex:

```
LIB.FORMPROP "WIDTH" 500
LIB.FORMPROP "HEIGHT" 300
LIB.FORMPROP "STARTPOSITION" 1
LIB.FORMTEXT "TEST LABEL"
LIB.ADDLABEL "LBL"
LIB.LABELPROP "LBL" "AUTOSIZE" 0
LIB.LABELPROP "LBL" "WIDTH" 100
LIB.LABELPROP "LBL" "HEIGHT" 50
LIB.LABELPROP "LBL" "LEFT" 5
LIB.LABELPROP "LBL" "TOP" 10
LIB.LABELPROP "LBL" "DIMFONT" 12
LIB.LABELPROP "LBL" "ALIGN" 0
LIB.LABELPROP "LBL" "BACKCOLOR" 9
LIB.LABELPROP "LBL" "FORECOLOR" 137
LIB.LABELTEXT "LBL" "MY LABEL"
LIB.SHOWFORM
```

9.8 LIB.LABELTEXT

Write a Text in to NsLabel

Syntax

LIB.LABELTEXT "LBLNAME" "TEXT"

LBLNAME NsLabel Name

TEXT NsLabel Text

9.9 LIB.LABELPRINT

Write a Isous Variable in to NsLabel
VAR Type: **$,:,$[I], $[O], $[Q], $[R], $[P], $[E], $[S], $[U], $[X], $[Y], $[W], $[K]**

Syntax
LIB.LABELPRINT "LBLNAME" "FORMAT" VAR
LBLNAME NsLabel Name
FORMAT Print Format:

"B"	**ON/OFF** show **COLORON-COLOROFF** if **USESTYLE=0**, or the standard image of **IsoUs LED ON/OFF**
"I"	Only Integer value
"Fndec"	Number of decimal

B (USESTYLE=1)  **(off)**  **(on)**
B (USESTYLE=0)  **(COLOROFF)**  **(COLORON)**

I 33
F3 52.676

9.10 LIB.LABELF

Set the FORMAT for PRINT VAR BY TIMER

Syntax
LIB.LABELF "LBLNAME" "FORMAT" VAR
LBLNAME NsLabel Name
FORMAT Print Format:

"B"	**ON/OFF** show **COLORON-COLOROFF** if **USESTYLE=0**, or the standard image of **IsoUs LED ON/OFF**
"I"	Only Integer value
"Fndec"	Number of decimal

VAR Type: **$,:,$[I], $[O], $[Q], $[R], $[P], $[E], $[S], $[U], $[X], $[Y], $[W], $[K]**

9.11 LIB.ADDBUTTON

Add a NsButton to NsForm. The NsButton properties can be setted befor a LIB.SHOWFORM

Syntax
LIB.ADDBUTTON "NAME"
NAME (TEXT) :
Univocal Name of NsButton Whitout special characters

Ex:
LIB.ADDBUTTON "BTN"

9.12 LIB.BUTTONPROP

Set the NsButton properties. The properties can be setted before a **LIB.SHOWFORM** but before, is necessary that the NsButton is added to NsForm with **LIB.ADDBUTTON**

Syntax

LIB.BUTTONPROP "BTNNAME" "PROPNAME" PROPVALUE (value or variable)

BTNNAME NsButton Name

PROPNAME (Property Name TEXT) :

VISIBLE

PropValue 0 (< of ZERO) NsButton Invisible

PropValue 1 (> of ZERO) NsButton Visible (default)

ENABLED

PropValue 0 (< of ZERO) NsButton enabled (default)

PropValue 1 (> of ZERO) NsButton disabled

WIDTH

PropValue > zero NsButton Width

HEIGHT

PropValue > zero NsButton Height

LEFT

PropValue > zero NsButton Left position in NsForm

TOP

PropValue > zero NsButton Top position in NsForm

DIMFONT

PropValue Value 6 to 100. Font Dimension

BACKCLOR

PropValue Value 0 to 140.Set BACKCOLOR (see color table)

FORECOLOR

PropValue Value 0 to 140.Set FORECOLOR (see color table)

COLORON

PropValue Value 0 to 140.Set color ON for ON/OFF format (BUTTONF – BUTTONPRINT)

COLOROFF

PropValue Value 0 to 140.Set color OFF for ON/OFF format (BUTTONF – BUTTONPRINT)

STYLE (default 0)

PropValue:

0 Normal **1** *Medium* ***2 Bold*** ***3 Ultra Bold***

ALIGN (default 1)

PropValue:

0 Middle Left	**1** Middle Center	**2** Middle Right
3 Top Left	**4** Top Center	**5** Top Right
6 Bottom Left	**7** Bottom Center	**8** Bottom Right

DEST

Destination variable writing. Indicates the destination variable where the value is written to Source

VAR Type: **$,:,$[O],$[P],$[K], $[A]**

SOURCE

Source variable. Value to write to DEST

When you press the value of the source variable is written to DEST.

VAR Type: **$,:,$[I], $[O], $[Q], $[R], $[P], $[E], $[S], $[U], $[X], $[Y], $[W], $[K]**

CLOSEFORM (default 0)

PropValue 0 (o < of ZERO) When press the NsForm is Not closed

PropValue 1 (o > of ZERO) When press the NsForm is closed

TIMER (default 0)

PropValue > zero(milliseconds) enable the print by **TIMER** for **BUTTONF**

INPUTUPDATE (default 0)

PropValue =1 when click all INPUT objects in yhe Form, (ITEXT,INPUT,ICHECK) update the destination VAR

9.13 LIB.BUTTONTEXT

Write a Text in to NsButton

Syntax
LIB.BUTTONTEXT "BTNNAME" "TEXT"
LBLNAME NsButton Name
TEXT NsButton Text

If the TEXT has the prefix **(@)**text, is showed the image **text.png** saved in ***\DataForm\Images***

LIB.BUTTONTEXT "BTNNAME" "EXIT" 

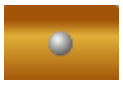
LIB.BUTTONTEXT "BTNNAME" "(@)EXIT"  (image **Exit.png** saved in ***\DataForm\Images***)

9.14 LIB.BUTTONPRINT

Write a Isous Variable in to NsButton

Syntax
LIB.BUTTONPRINT "BTNNAME" "FORMAT" $VAR
BTNNAME NsButton Name
FORMAT Print Format:

"B"	**ON/OFF** show **COLORON-COLOROFF** if **USESTYLE=0**, or the standard image of **IsoUs LED ON/OFF**
"I"	Only Integer value
"Fndec"	Number of decimal

B (USESTYLE=1)  **(off)**  **(on)**

B (USESTYLE=0) **(COLOROFF)**  **(COLORON)**

I 

F3 

9.15 LIB.BUTTONF

Set the FORMAT for PRINT VAR BY TIMER

Syntax

LIB.BUTTONF "BTNNAME" "FORMAT" VAR

BTNNAME Button Name

FORMAT Print Format:

"B"	**ON/OFF** show **COLORON-COLOROFF** if **USESTYLE=0**, or the standard image of **IsoUs LED ON/OFF**
"I"	Only Integer value
"Fndec"	Number of decimal

VAR Type: **$,:,$[I], $[O], $[Q], $[R], $[P], $[E], $[S], $[U], $[X], $[Y], $[W], $[K]**

9.16 LIB.ADDINPUT

Add a NsInput to NsForm. The NsInput properties can be setted befor a LIB.SHOWFORM
The NsInput Object is used to write numerical value to Isous variable

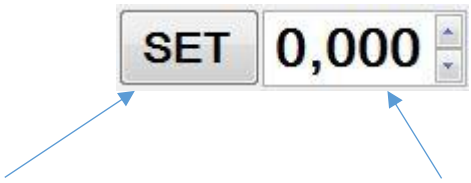
Syntax

LIB.ADDINPUT "NAME"

NAME (TEXT) :
Univocal Name of NsInput Without special characters

Ex:

LIB.ADDINPUT "INP"



SET VAL Button **NUMERIC UP DOWN**

Button **SET** when pressed writes the value in Isous variable identified in **"DEST"**
If **AUTOSET=1** the value is automatically written when this is changed

9.17 LIB.INPUTPROP

Set the NsInput properties. The properties can be setted before a **LIB.SHOWFORM** but before, is necessary that the NsInput is added to NsForm.

Syntax

LIB.INPUTPROP "INPNAME" "PROPNAME" PROPVALUE (value or variable)

INPNAME NsInput Name

PROPNAME (Property Name TEXT) :

VISIBLE

PropValue 0 (< of ZERO) NsInput Invisible
PropValue 1 (> of ZERO) NsInput Visible (default)

ENABLED

PropValue 0 (< of ZERO) NsInput enabled (default)
PropValue 1 (> of ZERO) NsInput disabled

WIDTH

PropValue > zero NsInput Width

LEFT

PropValue > zero NsInput Left position in NsForm

TOP

PropValue > zero NsInput Top position in NsForm

DIMFONT

PropValue Value 6 to 100. Font Dimension

BACKCLOR

PropValue Value 0 to 140.Set BACKCOLOR (see color table)

FORECOLOR

PropValue Value 0 to 140.Set FORECOLOR (see color table)

ALIGN (default 0)

PropValue:

0 Left **1** Center **2** Right

DEST

Destination variable writing. Indicates the destination variable where the value is written

VAR Type: **$,:,$[O],$[P],$[K], $[A]**

MIN

PropValue Minimum value accepted (double)

MAX

PropValue Maximum value accepted (double)

NDECIMAL

PropValue > zero Number of decimals displayed

INCREMENT

PropValue Increment value when pressed **up** - **down** buttons (double)

AUTOSET

PropValue *0* (< ZERO) The value is writes when pressed the button SET

PropValue 1 (> ZERO) The value is automatically writes when this changed

VAR Type: **$,:,$[O],$[P],$[K], $[A]**

9.18 LIB.INPUTSETVALUE

Set the initial value of NsInput

Syntax

LIB.INPUTSETVALUE "INPNAME" VAR

INPNAME NsInput Name

VAR Type: **NUM**,**$,:,$[I], $[O], $[Q], $[R], $[P], $[E], $[S], $[U], $[X], $[Y], $[W], $[K]**

Ex:

```
$VAR=103.2569
LIB.INPUTSETVALUE "INP" $VAR //SET VALUE AT 103.2569
```

9.19 LIB.ADDITEXT

Add an ITEXT to NsForm. The ITEXT properties can be setted before a LIB.SHOWFORM
The ITEXT Object is used to write numerical value to Isous variable

Syntax
LIB.ADDITEXT "NAME"
NAME (TEXT) :
Univocal Name of ITEXT without special characters

Ex:
LIB.ADDITEXT "TEXT1"

9.20 LIB.ITEXTPROP

Set the ITEXT properties. The properties can be setted before a **LIB.SHOWFORM** but before, is necessary that the ITEXT is added to NsForm.

Syntax
LIB.ITEXTPROP "TXTNAME" "PROPNAME" PROPVALUE (value or variable)
TEXTNAME ITEXT Name
PROPNAME (Property Name TEXT) :

VISIBLE
PropValue 0 (< of ZERO) ITEXT Invisible
PropValue 1 (> of ZERO) ITEXT Visible (default)
ENABLED
PropValue 0 (< of ZERO) ITEXT enabled (default)
PropValue 1 (> of ZERO) ITEXT disabled
WIDTH
PropValue > zero ITEXT Width
HEIGHT
PropValue > zero ITEXT Height
LEFT
PropValue > zero ITEXT Left position in NsForm
TOP
PropValue > zero ITEXT Top position in NsForm
DIMFONT
PropValue Value 6 to 100. Font Dimension
BACKCLOR
PropValue Value 0 to 140.Set BACKCOLOR (see color table)
FORECOLOR
PropValue Value 0 to 140.Set FORECOLOR (see color table)
ALIGN (default 0)
PropValue:
0 Left **1** Center **2** Right
DEST
Destination variable writing. Indicates the destination variable where the value is written
VAR Type: **$,:,$[O],$[P],$[K], $[A]**
MIN
PropValue Minimum value accepted (double)
MAX
PropValue Maximum value accepted (double)
NDECIMAL
PropValue > zero Number of decimals displayed

AUTOSET

PropValue *0* (< ZERO) The value is writes when pressed a BUTTON with UPDATEINPUT
PropValue 1 (> ZERO) The value is automatically writes when is pressed CR

9.21 LIB.ITEXTSETVALUE

Set the initial value of ITEXT

Syntax

LIB.ITEXTSETVALUE "TXTNAME" VAR
TXTNAME ITEXT Name
VAR Type: **NUM,$,:,$[I], $[O], $[Q], $[R], $[P], $[E], $[S], $[U], $[X], $[Y], $[W], $[K]**

Ex:

```
$VAR=103.2569
LIB.ITEXTSETVALUE "TEXT" $VAR //SET VALUE AT 103.2569
```

9.22 LIB.ADDCHECK

Add an CHECK to NsForm. The CHECK properties can be setted before a LIB.SHOWFORM
The CHECK Object is used to write a ON/OFF value

Syntax

LIB.ADDCHECK "NAME"
NAME (TEXT) :
Univocal Name of CHECK without special characters

Ex:

```
LIB.ADDCHECK "CHECK1"
```

9.23 LIB.CHECKPROP

Set the CHECK properties. The properties can be setted before a **LIB.SHOWFORM** but before, is necessary that the CHECK is added to NsForm.

Syntax

LIB.CHECKPROP "CHECKNAME" "PROPNAME" PROPVALUE (value or variable)
CHECKNAME CHECK Name
PROPNAME (Property Name TEXT) :

VISIBLE

PropValue 0 (< of ZERO) CHECK Invisible
PropValue 1 (> of ZERO) CHECK Visible (default)

ENABLED

PropValue 0 (< of ZERO) CHECK enabled (default)
PropValue 1 (> of ZERO) CHECK disabled

WIDTH

PropValue > zero CHECK Width

HEIGHT

PropValue > zero CHECK Height

LEFT

PropValue > zero CHECK Left position in NsForm

TOP

PropValue > zero CHECK Top position in NsForm

DIMFONT

PropValue Value 6 to 100. Font Dimension

BACKCLOR

PropValue Value 0 to 140.Set BACKCOLOR (see color table)

FORECOLOR

PropValue Value 0 to 140.Set FORECOLOR (see color table)

ALIGN (default 0)

PropValue:

0 Left **1** Center **2** Right

DEST

Destination variable writing. Indicates the destination variable where the value is written

VAR Type: **$,:,$[O],$[P],$[K], $[A]**

AUTOSET

PropValue *0* (< ZERO) The value is writes when pressed a BUTTON with UPDATEINPUT

PropValue 1 (> ZERO) The value is automatically writes when CLICK

9.24 LIB.CHECKSETVALUE

Set the initial value of CHECK

Syntax

LIB.CHECKSETVALUE "CHECKNAME" VAR

CHECKNAME CHECK Name

VAR Type: **NUM**,**$,:,$[I], $[O], $[Q], $[R], $[P], $[E], $[S], $[U], $[X], $[Y], $[W], $[K]**

Ex:

```
$VAR=1
LIB.CHECKTSETVALUE "CHECK" $VAR //SET VALUE AT 1 CHECK CHECKED
```

9.25 LIB.CHECKTEXT

Caption Check

Syntax

LIB.CHECKTEXT "CHECKNAME" "TEXT"

CHECKNAME CHECK Name

TEXT Check Caption Text

9.26 LIB.ADDCOMBO

Add an COMBO to NsForm. The COMBO properties can be setted before a LIB.SHOWFORM
The COMBO Object is used to write a ON/OFF value

Syntax
LIB.ADDCOMBO "NAME"
NAME (TEXT) :
Univocal Name of COMBO without special characters

Ex:
LIB.ADDCOMBO "COMBO1"

9.27 LIB.COMBOPROP

Set the COMBO properties. The properties can be setted before a **LIB.SHOWFORM** but before, is necessary that the COMBO is added to NsForm.

Syntax
LIB.COMBOPROP "COMBONAME" "PROPNAME" PROPVALUE (value or variable)
COMBONAME COMBO Name
PROPNAME (Property Name TEXT) :

VISIBLE
PropValue 0 (< of ZERO) COMBO Invisible
PropValue 1 (> of ZERO) COMBO Visible (default)
ENABLED
PropValue 0 (< of ZERO) COMBO enabled (default)
PropValue 1 (> of ZERO) COMBO disabled
WIDTH
PropValue > zero COMBO Width
HEIGHT
PropValue > zero COMBO Height
LEFT
PropValue > zero COMBO Left position in NsForm
TOP
PropValue > zero COMBO Top position in NsForm
DIMFONT
PropValue Value 6 to 100. Font Dimension
FORECOLOR
PropValue Value 0 to 140.Set FORECOLOR (see color table)
ALIGN (default 0)
PropValue:
0 Left **1** Center **2** Right
DEST
Destination variable writing. Indicates the destination variable where the value is written
VAR Type: **$,:,$[O],$[P],$[K], $[A]**
AUTOSET
PropValue *0* (< ZERO) The value is writes when pressed a BUTTON with UPDATEINPUT
PropValue 1 (> ZERO) The value is automatically writes when the ITEM is SELECTED

9.28 LIB.COMBOSETVALUE

Set the initial value of COMBO

Syntax

LIB.COMBOSETVALUE “COMBONAME” VAR

COMBONAME COMBO Name

VAR Type: **NUM**,**$**,**:**

Ex:

$VAR=2

LIB.**COMBOSETVALUE** "COMBO" $VAR //SET VALUE to ITEM 2

9.29 LIB.COMBOITEM

Add an ITEM to COMBO

Syntax

LIB.COMBOITEM “COMBONAME” “ITEMNAME” VAR

COMBONAME COMBO Name

ITEMNAME ITEM Text

VAR value for ITEM

VAR Type: **NUM**,**$**,**:**

Ex:

LIB.COMBOITEM "COMBO" “ITEM 1” 1 //ITEM 1 VALUE 1

LIB.COMBOITEM "COMBO" “ITEM 2” 2 //ITEM 2 VALUE 2

LIB.COMBOITEM "COMBO" “ITEM 3” 3 //ITEM 3 VALUE 3

9.30 LIB.ADDSLIDER

Add an SLIDER to NsForm. The SLIDER properties can be setted before a LIB.SHOWFORM
The SLIDER Object is used to write numerical value to Isous variable

Syntax
LIB.ADDSLIDER "NAME"
NAME (TEXT) :
Univocal Name of SLIDER without special characters

Ex:
LIB.ADDSLIDER "SLIDER1"

9.31 LIB.SLIDERPROP

Set the SLIDER properties. The properties can be setted before a **LIB.SHOWFORM** but before, is necessary that the SLIDER is added to NsForm.

Syntax
LIB.SLIDERPROP "SLIDERNAME" "PROPNAME" PROPVALUE (value or variable)
SLIDERNAME SLIDER Name
PROPNAME (Property Name TEXT) :

VISIBLE
PropValue 0 (< of ZERO) SLIDER Invisible
PropValue 1 (> of ZERO) SLIDER Visible (default)
ENABLED
PropValue 0 (< of ZERO) SLIDER enabled (default)
PropValue 1 (> of ZERO) SLIDER disabled
VISLABEL
PropValue 0 (< of ZERO) LABEL SLIDER Invisible
PropValue 1 (> of ZERO) LABEL SLIDER Visible (default)
WIDTH
PropValue > zero SLIDER Width
HEIGHT
PropValue > zero SLIDER Height
LEFT
PropValue > zero SLIDER Left position in NsForm
TOP
PropValue > zero ITEXT Top position in NsForm
DIMFONT
PropValue Value 6 to 100. Font Dimension
BACKCLOR
PropValue Value 0 to 140.Set BACKCOLOR (see color table)
FORECOLOR
PropValue Value 0 to 140.Set FORECOLOR (see color table)
LARGE
PropValue defines the quantity of variation when the slider bar has a large change
SMALL
PropValue defines the quantity of variation when the slider bar has a small change
TICKP
PropValue Sign position:
0 None **1** TopLeft **2** BottomRight **3** Both
TICKF
PropValue Sign frequency

DEST

Destination variable writing. Indicates the destination variable where the value is written
VAR Type: **$,:,$[O],$[P],$[K], $[A]**

MIN

PropValue Minimum value accepted (double)

MAX

PropValue Maximum value accepted (double)

NDECIMAL

PropValue **>** zero Number of decimals displayed

AUTOSET

PropValue *0* (< ZERO) The value is writes when pressed a BUTTON with UPDATEINPUT
PropValue 1 (> ZERO) The value is automatically writes when the bar change

9.32 LIB.SLIDERSETVALUE

Set the initial value of SLIDER

Syntax

LIB.SLIDERSETVALUE "SLIDERNAME" VAR
SLIDERNAME SLIDER Name
VAR Type: **NUM**,**$,:,$[I], $[O], $[Q], $[R], $[P], $[E], $[S], $[U], $[X], $[Y], $[W], $[K]**

Ex:

```
$VAR=103
LIB.SLIDERSETVALUE "SLIDER" $VAR //SET VALUE AT 103
```

9.33 LIB.GETVAR

Allows to read a FORM variable (double) in a IsoUs Variable.
The FORM variables are proprietary of FORM in use (MAX 1000 Variables)
The Variable 999 contains the status FORM
999=1 Form Open 999=0 From Closed

Syntax

LIB.GETVAR SOURCEINDEX DEST

SOURCEINDEX Form Variable Index to Read (0 -999), can be a number or a IsoUs variable.

DEST IsoUs variable destination

Ex:

```
$VARFORM=100
LIB.GETVAR $VARFORM $VAR //READ FORM VAR 100 IN $VAR
LIB.GETVAR 10 $VAR // READ FORM VAR 10 IN $VAR
```

9.34 LIB.SETVAR

Allows to write a FORM variable (double) by a IsoUs Variable.
The FORM variables are proprietary of FORM in use (MAX 1000 Variables)

Syntax

LIB.SETVAR DESTINDEX SOURCE

DESTINDEX Form Variable Index to Write (0 -999), can be a number or a IsoUs variable.

SOURCE IsoUs source Var

Ex:

```
$VARFORM=100
$SOURCE=10.123
LIB.SETVARVAR $VARFORM $ SOURCE //WRITE FORM VAR 100 TO VALUE 10.123
LIB.SETVAR 10 $ SOURCE // WRITE FORM VAR 10 TO VALUE 10.123
```

9.35 LIB.DEBUG

Allows to write in a Window by Gcode, informations for Debug

Syntax

LIB.DEBUG "TEXT" VAR

TEXT Text to write

VAR Variable to write

Ex:

```
$VAR=10
LIB.DEBUG "DEBUG TEST 1" $VAR
$VAR=15
LIB.DEBUG "DEBUG TEST 2" $VAR
$VAR=20
LIB.DEBUG "DEBUG TEST 3" $VAR
```

9.36 COLOR TABLE

Following color table to refer when the property is a color

0	ALICEBLUE	35	DARKSLATEGRAY	70	LIGHTSALMON
1	ANTIQUEWHITE	36	DARKTURQUOISE	71	LIGHTSEAGREEN
2	AQUA	37	DARKVIOLET	72	LIGHTSKYBLUE
3	AQUAMARINE	38	DEEPPINK	73	LIGHTSLATEGRAY
4	AZURE	39	DEEPSKYBLUE	74	LIGHTSTEELBLUE
5	BEIGE	40	DIMGRAY	75	LIGHTYELLOW
6	BISQUE	41	DODGERBLUE	76	LIME
7	BLACK	42	FIREBRICK	77	LIMEGREEN
8	BLANCHEDALMOND	43	FLORALWHITE	78	LINEN
9	BLUE	44	FORESTGREEN	79	MAGENTA
10	BLUEVIOLET	45	FUCHSIA	80	MAROON
11	BROWN	46	GAINSBORO	81	MEDIUMAQUAMARINE
12	BURLYWOOD	47	GHOSTWHITE	82	MEDIUMBLUE
13	CADETBLUE	48	GOLD	83	MEDIUMORCHID
14	CHARTREUSE	49	GOLDENROD	84	MEDIUMPURPLE
15	CHOCOLATE	50	GRAY	85	MEDIUMSEAGREEN
16	CORAL	51	GREEN	86	MEDIUMSLATEBLUE
17	CORNFLOWERBLUE	52	GREENYELLOW	87	MEDIUMSPRINGGREEN
18	CORNSILK	53	HONEYDEW	88	MEDIUMTURQUOISE
19	CRIMSON	54	HOTPINK	89	MEDIUMVIOLETRED
20	CYAN	55	INDIANRED	90	MIDNIGHTBLUE
21	DARKBLUE	56	INDIGO	91	MINTCREAM
22	DARKCYAN	57	IVORY	92	MISTYROSE
23	DARKGOLDENROD	58	KHAKI	93	MOCCASIN
24	DARKGRAY	59	LAVENDER	94	NAVAJOWHITE
25	DARKGREEN	60	LAVENDERBLUSH	95	NAVY
26	DARKKHAKI	61	LAWNGREEN	96	OLDLACE
27	DARKMAGENTA	62	LEMONCHIFFON	97	OLIVE
28	DARKOLIVEGREEN	63	LIGHTBLUE	98	OLIVEDRAB
29	DARKORANGE	64	LIGHTCORAL	99	ORANGE
30	DARKORCHID	65	LIGHTCYAN	100	ORANGERED
31	DARKRED	66	LIGHTGOLDENRODYELLOW	101	ORCHID
32	DARKSALMON	67	LIGHTGRAY	102	PALEGOLDENROD
33	DARKSEAGREEN	68	LIGHTGREEN	103	PALEGREEN
34	DARKSLATEBLUE	69	LIGHTPINK	104	PALETURQUOISE

105	PALEVIOLETRED		140	YELLOWGREEN	
106	PAPAYAWHIP				
107	PEACHPUFF				
108	PERU				
109	PINK				
110	PLUM				
111	POWDERBLUE				
112	PURPLE				
113	RED				
114	ROSYBROWN				
115	ROYALBLUE				
116	SADDLEBROWN				
117	SALMON				
118	SANDYBROWN				
119	SEAGREEN				
120	SEASHELL				
121	SIENNA				
122	SILVER				
123	SKYBLUE				
124	SLATEBLUE				
125	SLATEGRAY				
126	SNOW				
127	SPRINGGREEN				
128	STEELBLUE				
129	TAN				
130	TEAL				
131	THISTLE				
132	TOMATO				
133	TRANSPARENT				
134	TURQUOISE				
135	VIOLET				
136	WHEAT				
137	WHITE				
138	WHITESMOKE				
139	YELLOW				

10 COMPILATOR SWITCH

The compilator switch, allows to give some directives to compilator by the constructs:
IFDEF
ELSEDEF
ENDIFDEF
This allow to compile only the code which is part of the logic of switch selected.
Currently is available only one system switch:
AXES
It contains the number of Axes configured in the IsoUs application:
Ex: for 3 AXES configuration

```
IFDEF AXES=3 // AXES NUMBER=3
   G1 X100 Y100 Z200
ENDIFDEF
IFDEF AXES=4 //AXES NUMBER=4
   G1 X100 Y100 Z200 A200
ENDIFDEF
```

In this example, only the code ***G1X100Y100Z200*** will be compiled, the code ***G1X100Y100Z200A200*** will be removed.
In the normal condition, without switch directive, will be showed an error AXIS NOT CONFIGURED (A)
Is possible add a custom switch, see:
System Utility – ***3.5.2 Add a Custom Parameter***

10.1 IFDEF

Instruction IF on the switch selected:
IFDEF AXES=3
If the condition is true, the code content between **IFEDF** and **ELSEDEF** or **ENDIFDEF**, will be compiled, otherwise will be removed and compiled the code insert in the **ELSEDEF**
In the IFDEF can be used the following operators:

=	***Equal***
>	***Greater***
<	***Lower***
>=	***Greater Equal***
<=	***Lower Equal***
<>	***Different***

10.2 ELSEDEF

Instruction ELSE on the switch selected:

```
IFDEF AXES=3
   G1 X100
ELSEDEF
   G1 Y30
ENDIFDEF
```

If the condition is true, AXES<>3 the code content between ELSEDEF and ENDIFDEF, will be compiled

10.3 ENDIFDEF

Instruction **ENDIFDEF**

10.4 NOAXESREADY

This Directive allows to run a Gcode though the axes are not ready. (generally it is inserted at the beginning the Gcode)

```
NOAXESREADY
G71.1 X  // ENABLE X
G71.1 Y  // ENABLE Y
G71  X          // HOMING X
G71  Y          // HOMING Y
```

10.5 ONERROR

This Directive allows to programming a Gcode part that will be performed when an error is occured (Like to M Error)
Following will be performed the **M Error** if is set

```
ONERROR
$[O1]=0 // DISABLE OUT
$[O2]=0 // DISABLE OUT
ENDON
```

10.6 ONSTOP

This Directive allows to programming a Gcode part that will be performed when the CN go from RUN to STOP (Like to M Stop)
Following will be performed the **M Stop** if is set

```
ONSTOP
$[O1]=0 // DISABLE OUT
$[O2]=0 // DISABLE OUT
ENDON
```

10.7 ENDON

Ends **ONERROR** and **ONSTOP**

10.8 USET

This Directive allows performed a Macro after the instruction **Tn**

```
USET 5  // AFTER Tn WILL BE PERFORMED M5 MACRO
```

10.9 USEH

This Directive allows performed a Macro after the instruction **Hn**

```
USEH 6  // AFTER Hn WILL BE PERFORMED M6 MACRO
```

11 MATH FUNCTIONS

11.1 SIN

Return the sinus of an angle in a DOUBLE type value.
Syntax
SIN (angle)
angle Mandatory. Angle in radiant (it can bea an expression)
Ex:
$VAR=SIN($VAR1*3)

11.2 COS

Return the co-sinus of an angle in a DOUBLE type value.
Syntax
COS (angle)
angle Mandatory. Angle in radiant (it can be an expression)
Ex:
$VAR=COS($VAR1*3)

11.3 LOG

Return the base of natural logarithm in a DOUBLE type value.
Syntax
LOG (expression)
expression Mandatory. Expression.
Ex:
$VAR=LOG($VAR1*3)

11.4 EXP

Return the exponential of an expression in a DOUBLE type value.
Syntax
EXP (expression)
expression Mandatory. Expression
Ex:
$VAR=EXP(10.15)

11.5 INT

Return the integer part of a DOUBLE value rounded
Syntax
INT (expression)
expression Mandatory. Numeric expression
Es:
$VAR=35.14
$VAR1=INT($VAR)

11.6 FIX

Return the integer part of a DOUBLE value truncated
Syntax
FIX (expression)
expression Mandatory. Numeric expression
Es:
$VAR=35.14
$VAR1=FIX($VAR)

11.7 ABS

Return the absolute value
Syntax
ABS (expression)
expression Mandatory. Numeric expression
Es:
$VAR=-35.14
$VAR1=ABS($VAR)

11.8 DRG

Enable angle in **DEGREES** for COS,SIN,TAN,ACOS,ASIN,ATAN functions.
By default at each start of PART PROGRAM angle are setted in radiant
Syntax
DRG

11.9 RAD

Enable angle in **RADIANT** for COS,SIN,TAN,ACOS,ASIN,ATAN functions.
By default at each start of PART PROGRAM angle are setted in radiant
Syntax
RAD

11.10 SQR

Return the square root of an expression in DOUBLE type value.
Syntax
SQR (expression)
expression Mandatory. Expression
Ex:
$VAR=SQR($VAR1)

11.11 TAN

Return the tangent of an angle in a DOUBLE type value.
Syntax
TAN (angle)
angle Mandatory. Angle in radiant (it can be an expression)
Ex:
$VAR=TAN($VAR1*3)

11.12 ATAN

Return the arctangent of an expression. Result is from -3.14 to 3.14

Syntax

ATAN (expression)

expression Mandatory. Expression

Ex:

$VAR=ATAN($VAR1*3)

11.13 ASIN

Return the arc-sinus of an expression.

Syntax

ASIN (expression)

expression Mandatory. Expression

Ex:

$VAR=ASIN($VAR1*3)

11.14 ACOS

Return the arc-cos of an expression.

Syntax

ACOS (expression)

expression Mandatory. Expression

Ex:

$VAR=ACOS($VAR1*3)

I. ISTRUZIONI PER IL CONTROLLO MULTIPROCESSO

L' istruzioni per il controllo del MultiProcesso, servono per gestire più CNC che sono eseguiti nella stessa unità PC.
Non necessita di nessuna configurazione particolare, in quanto questa è automaticamente rilevata dal file IsoUs.cfg.

12 FUNCTIONS for Multi process Control

The instructions for multi process control allow to management more CNC (IsoUs Process) in the same PC
These functions no need any special configuration.

12.1 CNC.LOAD

Load a Gcode File in the Process.

Syntax

CNC.LOAD CN TypeRun InEditor "PATH"

CN Process Number from 1 to 8

TypeRun **0** Load Only
1 Load and Run
2 Load and Preview
3 Load, Preview and Run

InEditor **0** Not Send to UsEditor
1 Send to UsEditor

PATH (in quotes) If **PATH** starts with **$APPPATH**, the path is Relative to IsoUs Folder installation
The file name must have the extension

Ex:

CNC.LOAD 1 1 1 "$APPPATH\PROJECT\IMPORT\TEST.ISO" **// RELATIVE PATH**
CNC.LOAD 1 1 1 "C:\FILE\TEST.ISO" **// ABSOLUTE PATH**

12.2 CNC.RUN

Run Gcode in the Process.

Syntax

CNC.RUN CN

CN Process Number from 1 to 8
If **CN =-1** Run on all Process

Ex:

CNC.RUN 1 **// RUN PROCESS 1**
CNC.RUN -1 **// RUN ALL PROCESS**

12.3 CNC.PREVIEW

Preview Gcode in the Process.

Syntax

CNC.PREVIEW CN

CN Process Number from 1 to 8
If **CN =-1** Preview on all Process

Ex:

CNC.PREVIEW 1 **// PREVIEW PROCESS 1**
CNC. PREVIEW -1 **// PREVIEW ALL PROCESS**

12.4 CNC.STOP

Stop Gcode in the Process.

Syntax

CNC.STOP CN

CN Process Number from 1 to 8
If **CN =-1** Stop on all Process

Ex:

```
CNC.STOP 1  // STOP PROCESS 1
CNC. STOP -1  // STOP ALL PROCESS
```

12.5 CNC.PAUSE

Pause Gcode in the Process.

Syntax

CNC.PAUSE CN

CN Process Number from 1 to 8
If **CN =-1** Pause on all Process

Ex:

```
CNC.PAUSE 1  // PAUSE PROCESS 1
CNC. PAUSE -1  // PAUSE ALL PROCESS
```

12.6 CNC.STATUS

Read Status from Process.

Syntax

CNC.STATUS CN $VAR

CN Process Number from 1 to 8

$VAR Return Status (bit mapped)

Bit 0	Run	**Bit 1**	Error	**Bit 2**	Axes Move
Bit 3	Pause	**Bit 4**	Home X Ok	**Bit 5**	Home Y OK
Bit 6	Home Z Ok	**Bit 7**	Home A Ok	**Bit 8**	Home B Ok
Bit 9	Home C Ok	**Bit 10**	Home U Ok	**Bit 11**	Home V Ok
Bit 12	Home W Ok	**Bit 13**	Enable X	**Bit 14**	Enable Y
Bit 15	Enable Z	**Bit 16**	Enable A	**Bit 17**	Enable B
Bit 18	Enable C	**Bit 19**	Enable U	**Bit 20**	Enable V
Bit 21	Enable W	**Bit 22**	Ready to Run (Home and enable Ok on all Axes)		

Ex:

```
CNC.STATUS 1  $VAR
$VAR1=$VAR & 1 //TEST RUN
IF $VAR1 = 1
…// IS RUN
```

12.7 CNC.STATUSBIT

Read Status bit from Process.

Syntax

CNC.STATUS CN NrBit $VAR

CN	Process Number from 1 to 8
NrBit	Bit number (see CNC.STATUS)
$VAR	Return Status bit

Ex:

```
CNC.STATUSBIT 1  1 $VAR
IF $VAR = 1
…// IS RUN
```

12.8 CNC.INFO

Read informations from Process.

Syntax

CNC.INFO CN InfoType $VAR

CN	Process Number from 1 to 8
InfoType	Tipo informazione
	0 Read Demand Gcode Line in execution
	1 Read Real Gcode Line in execution
	2 Read Axes Resolution
	3 Read Feed Resolution
$VAR	Return Info

12.9 CNC.AXIS

Read Axis informations from Process.

Syntax

CNC.INFO CN AxisIndex AxisType $VAR

CN	Process Number from 1 to 8
AxisIndex	Axis Index 0 to 8
AxisType	Axis Information
	0 Read Demand Axis Value
	1 Read Real Axis Value
	2 Read Demand Axis Value (Syncro – Precise mode)
	3 Read Real Axis Value (Syncro – Precise mode)
	4 Read Total Offset Value (Work Origins,Offset Hn G43 etc.)
$VAR	Return Info

12.10 CNC.GROUP

Read Group Axis informations from Process.

Syntax

CNC.GROUP CN NrAxes AxisType $VAR

CN	Process Number from 1 to 8
NrAxes	Group of Axis 1 to 9
AxisType	Axis Information
	0 Read Demand Axis Value
	1 Read Real Axis Value
	2 Read Total Offset Value (Work Origins,Offset Hn G43 etc.)
$VAR	Return Info Start Variable or Array

Ex:

```
// ALLOC 3 VARIABLES FOR 3 AXES
$VARX=0
$VARY=0
$VARZ=0
CNC.GROUP 1 3 0  $VARX
// $VARX=VALUE ASSE X
// $VARY= VALUE ASSE Y
// $VARZ= VALUE ASSE Z
```

Ex:

```
// ALLOC ARRAY
DIM $ARR 3
CNC.GROUP 1 3 0  $ARR
// $ARR[0]= VALUE ASSE X
// $ARR[1]== VALUE ASSE Y
// $ARR[2]= VALUE ASSE Z
```

12.11 CNC.READVARADDR

Read a Gcode Variable by Address from Process.

Syntax

CNC.READVARADDR CN AddrVar $VAR

CN	Process Number from 1 to 8
AddrVar	Variable Address from 0 to 327667
$VAR	Return Value

12.12 CNC.READVARNAME

Read a Gcode Variable by Name from Process.

Syntax

CNC.READVARNAME CN $VAR "VARNAME"

CN	Process Number from 1 to 8
$VAR	Return Value
VARNAME	Variable Name UpperCase without $ (in quotes)

12.13 CNC.WRITEVARADDR

Write a Gcode Variable by Address from Process.

Syntax

CNC.WRITEVARADDR CN AddrVar $VAR

CN	Process Number from 1 to 8
	If **CN =-1** Write on all Process
AddrVar	Variable Address from 0 to 327667
$VAR	Source Value

12.14 CNC.WRITEVARNAME

Write a Gcode Variable by Name from Process.

Syntax

CNC.WRITEVARNAME CN $VAR "VARNAME"

CN	Process Number from 1 to 8
	If **CN =-1** Write on all Process
$VAR	Source Value
VARNAME	Destination Variable Name UpperCase without $ (in quotes)

12.15 CNC.READPARMAC

Read Machine Parameter from Process.

Syntax

CNC.READPARMAC CN $VAR "PARNAME"

CN	Process Number from 1 to 8
$VAR	Return Value
PARNAME	Parameter Name (in quotes)

12.16 CNC.WRITEPARMAC

Write Machine Parameter to Process.

Syntax

CNC.WRITEPARMAC CN $VAR "PARNAME"

CN	Process Number from 1 to 8
$VAR	Source Value
PARNAME	Parameter Name (in quotes)

12.17 CNC.ENABLEAXIS

Enable/Disable Axis to Process. (wait sequence finished)

Syntax

CNC.ENABLEAXIS CN AXIS STATE

CN	Process Number from 1 to 8
AXIS	Axis Index
STATE	Axis State
	0 Disable
	1 Enable

12.18 CNC.HOMEAXIS

Homing Axis to Process. (wait sequence finished) CNC.STOP for force Stop Homing

Syntax

CNC.HOMEAXIS CN AXIS

CN	Process Number from 1 to 8
AXIS	Axis Index

12.19 CNC.READGENERIC

Read Generic Variable (see below Type Generic)

Syntax

CNC.READGENERIC CN TYPE $PAR $VAR

CN	Process Number from 1 to 8
TYPE	Tipo Variable
$PAR	Additional Parameter (if it is not managed put $PAR=0)
$VAR	return Value

12.20 CNC.WRITEGENERIC

Write Generic Variable (see below Type Generic)

Syntax

CNC.WRITEGENERIC CN TYPE $PAR $VAR

CN	Process Number from 1 to 8
TYPE	Tipo Variable
$PAR	Additional Parameter (if it is not managed put $PAR=0)
$VAR	Source Value

Type Generic

TYPE	READ	WRITE	Description	$PAR
0	V		Read Demand Axis Value ($[Qn])	Axis Index
1	V		Read Real Axis Value ($[Rn])	Axis Index
2	V		Read Digital Input ($[In])	Index (0 to 255)
3	V	V	Read/Write Digital Output ($[On])	Index (0 to 255)
4	V	V	Read/Write Tool Table Parameter ($[Un])	Parameter Index
5	V		Read Head Parameter ($[Hn])	Parameter Index
6	V		Read SPEED ($[X0])	Not Used
7	V		Read Work Origins Enable ($[X3])	Not Used
8	V		Read Offset Enable ($[X4])	Not Used
9	V		Read FEED ($[X8])	Not Used
10	V		Read Origin Value ($[Yn])	Not Used
11	V		Read Offset Value ($[Wn])	Axis Index
12	V		Read H Set	Not Used
13	V		Read T Set	Not Used
14	V	V	Read/Write User Generic ($[Kn])	User Generic Index
15		V	Write Analo Value for SPINDLE	Not Used

13 REMOTE CONTROL FUNCTIONS

Remote functions allows to control more CNC connected in LAN (See **CN REMOTE CONFIGURATION**)

RunTimeError:

E1082 Remote CNC not connect
E1083 Remote CNC operation failure
E1084 Remote CNC Gcode Error
E1085 Remote CNC RUN Error
E1086 Remote CNC File not found
E1087 Remote CNC Axes not ready
E1088 Remote CNC Already in RUN
E1089 Remote CNC Axes not available
E1090 Remote CNCD Gcode Variable not found

13.1 REMOTE.LOAD

Load a Gcode file on Remote CNC

Syntax
REMOTE.LOAD CN RUN "PATH"

CN	Number of CNC set in ***clientNs.cfg file***
RUN	**0** Only Load
	1 Load and Run
PATH	Remote Gcode file Path
	If **PATH** starts with **$APPPATH**, the path is Relative to Isous Folder installation

Ex:

REMOTE.LOAD 0 1 "$APPPATH\PROJECT\IMPORT\TEST.ISO" **//RELATIVE PATH**

REMOTE.LOAD 0 1 "C:\FILE\TEST.ISO" **//ABSOLUTE PATH**

13.2 REMOTE.RUN

Run a Gcode file on Remote CNC

Syntax
REMOTE.RUN CN

CN	Number of CNC set in ***clientNs.cfg file***

Ex:

REMOTE.RUN 0 **//run on CN 0**

13.3 REMOTE.STOP

Stop a Gcode file on Remote CNC

Syntax
REMOTE.STOP CN

CN	Number of CNC set in ***clientNs.cfg file***

Ex:

REMOTE.STOP 0 **//stop CN 0**

13.4 REMOTE.PAUSE

Pause a Gcode file on Remote CNC

Syntax
REMOTE.PAUSE CN

CN	Number of CNC set in ***clientNs.cfg file***

Ex:

```
REMOTE.PAUSE 0  //pause CN 0
```

13.5 REMOTE.STATUS

Read a STATUS WORD on Remote CNC

Syntax
REMOTE.STATUS CN $VAR

CN	Number of CNC set in ***clientNs.cfg file***
$VAR	**Gcode** Variable Return in $VAR (bit Mapped): bit 0 → CN ERROR bit 1 → CN RUN bit 2 → CN AXES MOV bit 3 → CN PAUSE

Ex:

```
REMOTE.LOAD 0 0 "…"
REMOTE.LOAD 1 0 "…"
REMOTE.RUN 0
REMOTE.RUN 1
@LBL_1
REMOTE.STATUS 0 $VAR //READ STATUS CN 0 IN $VAR
REMOTE.STATUS 1 $VAR1 // READ STATUS CN 1 IN $VAR1
$V1=$VAR&2 //TEST BIT RUN
$V2=$VAR1&2
IF $V1=2 || $V2=2 //WAIT IF RUN
   GOTO LBL_1
END_IF
```

13.6 REMOTE.MOVE

Read Axes Mov on Remote CNC

Syntax
REMOTE.MOVE CN $VAR

CN	Number of CNC set in ***clientNs.cfg file***
$VAR	**Gcode** Variable Return in $VAR: 0 → Axes Stop 1 → Axes Mov

Es:

```
REMOTE.MOVE 0 $VAR //Read Axes Mov in $VAR
```

13.7 REMOTE.INFO

Read Info on Remote CNC

Syntax

REMOTE.INFO CN TYPE $VAR

CN Number of CNC set in ***clientNs.cfg file***

TYPE Type of Information:

0 → Get Demand Line Worked

1 → Get Real Line Worked

2 → Get Current FEED

3 → Get % of Override

4 → Get Tool Table Set (T)

5 → Get Head Set (H)

6 → Get Axes Resolution Set (1000,10000 etc,)

7 → Get Feed Resolution Set (1000 etc.)

$VAR **Gcode** Variable

Ex:

REMOTE.INFO 0 0 $VAR **//Read Demand Line Worked in $VAR**

13.8 REMOTE.AXIS

Read Axis Info on Remote CNC

Syntax

REMOTE.AXIS CN AXIS TYPE $VAR

CN Number of CNC set in ***clientNs.cfg file***

AXIS Axis 0=X - 1=Y - 2=Z - 3=A – 4=B – 5=C – 6=U -7=V - 8=W

TYPE Type of Information:

0 → Get Demand Absolute Axis Value (from ZERO ORIGIN)

1 → Get Demand Relative Axis Value (from Work Origin Set)

2 → Get Real Absolute Axis Value (from ZERO ORIGIN)

3 → Get Real Relative Axis Value (from Work Origin Set)

4 → Not Use

5 → Not Use

6 → Get Value Work Origin Set

7 → Get Value OffsetOrigin Set

8 → Get Work Origin Index Set

9 → Get Offset Origin Index Set

$VAR **Gcode** Variable

Ex:

REMOTE.AXIS 0 1 0 $VAR **//Get Y Demand Position value**

13.9 REMOTE.GROUP

Read Axes group Info on Remote CNC

Syntax

REMOTE.AXIS CN NAXES TYPE $VAR

CN Number of CNC set in ***clientNs.cfg file***

NAXES Number of Axes to read

TYPE Type of Information:

0 → Get Group Demand Absolute Axis Value (from ZERO ORIGIN)

1 → Get Group Demand Relative Axis Value (from Work Origin Set)

2 → Get Group Real Absolute Axis Value (from ZERO ORIGIN)

3 → Get Real Relative Axis Value (from Work Origin Set)

4 → Not Use

5 → Not Use

6 → Get Group Value Work Origin Set

7 → Get Group Value OffsetOrigin Set

8 → Get Group Work Origin Index Set

9 → Get Group Offset Origin Index Set

$VAR **Gcode Start** Variable

The Successive variables are get following the start Variable

ES:

```
$VARX=0 //Start Variable
$VARY=0
$VARZ=0
REMOTE.GROUP 0 3 0 $VARX //Read  X,Y,Z in VARX,VARY,VARZ
```

13.10 REMOTE.READISOVAR

Read a Gcode Variable on Remote CNC from Address

Syntax

REMOTE.READISOVAR CN ADDR $VAR

CN Number of CNC set in ***clientNs.cfg file***

ADDR **Addr** Gcode IsoVar from 10 to 32768 (from 0 to 9 are reserved)

$VAR **Gcode** Variable

Ex:

```
REMOTE.READISOVAR 0 10 $VAR //Read the variable addr 10 (first variable declared)
```

13.11 REMOTE.READVARNAME

Read a Gcode Variable on Remote CNC from Name

Syntax

REMOTE.READVARNAME CN $VAR "RemoteVarName"

CN Number of CNC set in ***clientNs.cfg file***

$VAR **Gcode** Variable

RemoteVarName Remote Variable Name without $

Ex:

REMOTE.READVARNAME 0 $VAR **"VAR1"//Read the variable name $VAR1**

13.12 REMOTE.WRITEISOVAR

Write a Gcode Variable on Remote CNC from Address

Syntax

REMOTE.WRITEISOVAR CN ADDR $VAR

CN Number of CNC set in ***clientNs.cfg file***

ADDR **Addr** Gcode IsoVar from 10 to 32768 (from 0 to 9 are reserved)

$VAR **Gcode** Variable

Ex:

$VAR=100.1

REMOTE.WRITEISOVAR 0 10 $VAR **//Write the variable addr 10**

13.13 REMOTE.WRITENAMEVAR

Write a Gcode Variable on Remote CNC from Name

Syntax

REMOTE.WRITENAMEVAR CN $VAR "RemoteVarName"

CN Number of CNC set in ***clientNs.cfg file***

$VAR **Gcode** Variable

RemoteVarName Remote Variable Name without $

Ex:

$VAR=100.1

REMOTE.WRITENAMEVAR 0 $VAR **"VAR1"// Write the variable $VAR1**

13.14 REMOTE.READCNVAR

Read a User Generic Variable on Remote CNC

Syntax

REMOTE.READCNVAR CN ADDR $VAR

CN Number of CNC set in ***clientNs.cfg file***

ADDR User Generic Variable addr from 0 to 9

$VAR **Gcode** Variable

Ex:

REMOTE.READCNVAR 0 1 $VAR **//Read User Generic 1**

13.15 REMOTE.WRITECNVAR

Write a User Generic Variable on Remote CNC

Syntax

REMOTE.WRITECNVAR CN ADDR $VAR

CN	Number of CNC set in ***clientNs.cfg file***
ADDR	User Generic Variable addr from 0 to 9
$VAR	**Gcode** Variable

Ex:

$VAR=10

REMOTE.WRITECNVAR 0 1 $VAR **//Write value 10 in USER GENERIC 1**

13.16 REMOTE.READINPUT

Read a Digital Input on Remote CNC

Syntax

REMOTE.READINPUT CN NINP $VAR

CN	Number of CNC set in ***clientNs.cfg file***
NINP	Digital Input Number from 0 to 255
$VAR	**Gcode** Variable
	0 → Input OFF
	1 → Input ON

Ex:

REMOTE.READINPUT 0 1 $VAR **//Read Digital Input 1**

13.17 REMOTE.READOUTPUT

Read a Digital Output on Remote CNC

Syntax

REMOTE.READOUT CN NOUT $VAR

CN	Number of CNC set in ***clientNs.cfg file***
NOUT	Digital Output Number from 0 to 255
$VAR	**Gcode** Variable
	0 → Output OFF
	1 → Output ON

Ex:

REMOTE.READIOUT 0 1 $VAR **//Read Digital Output 1**

13.18 REMOTE.WRITEOUTPUT

Write a Digital Output on Remote CNC

Syntax

REMOTE.WRITEOUT CN NOUT $VAR

CN	Number of CNC set in ***clientNs.cfg file***
NOUT	Digital Output Number from 0 to 255
$VAR	**Gcode** Variable
	0 → Output OFF
	1 → Output ON

Ex:

```
$VAR=1
REMOTE.WRITEOUT 0 1 $VAR //Set Digital Output 1
```

14 CN REMOTE CONFIGURATION

First to use the Remote Functions, need to configure the remote CNC in the LAN.
The CNC can be in the same PC, or in the different PC, but connected at same LAN.
The configuration of Remote CNC is in the following file::

NSCLIENT
NSSERVER

NSCLIENT

This represents the configuration of one or more CNCs that open the connection to a NS SERVER CNC. To configure client CNCs, just have a file named "ClientNs.cfg" in the Isous installation folder.

ClientNs.cfg

[USER_CN 0]
CN=IpAddr,port → CN 0 to process 0
CN=IpAddr,port → CN 1 to process 0
.
.
[USER_CN 1]
CN=IpAddr,port → CN 0 to process 1 (if present)
CN=IpAddr,port → CN 1 to process 1 (if present)

Where:
IpAddr → Server IP address for CN 0,1,2 etc.
port → Port Number for CN 0,1,2 etc.

NSSERVER

Represents a CNC server in a LAN network. This is to be configured as a server, you must have the PLUGIN "ServerNs.dll" (in PlugIn \ Bin) with the following setup in UsPlugin.cfg file:

<Set Name="UsServer" Image="" Title="@_US SERVER" Path="UsPlugIn\Bin\UsServer.dll" NameSpace="UsServer.GestUsServer" Autorun="False" Button="False" Enable="True" />

Additionally, the "ServerNs.cfg" file must be present in the Isous server installation folder that configures the server process listening port.

ServerNs.cfg
[USER_CN 0]
Port → Port in listening for CN 0
[USER_CN 1]
Port → Port in listening for CN 1 (if present)
.
.

Note

Be careful when configuring the Port. You must choose values that are not occupied by other operating system processes (typically higher values, eg 36000)

Example

Local LAN Configuration, Server IP Address 10.0.0.34 Port 36000
The PC that contains the server process must then have a static IP 10.0.0.34
The PC that contains the client process can have the same IP address as the same PC, but must have an appropriate address to the network eg: 10.0.0.50

ClientNs.cfg

[USER_CN 0]

CN=10.0.0.34,36000

ServerNs.cfg

[USER_CN 0]

36000

15 USING LOGIC OPERATORS

In this chapter will be explained how to use logical operator omitting classic math operators (+-*/).

15.1 ROUND BRACKETS ( )

They separate level of expressions, doing priority to content inside the brackets.

15.2 ISO EXPRESSIONS and SQUARE BRACKETS [ ]

The use of square brackets is not permitted in generic expression. They are valid only for specific type of variable and exclusively in the functions listed below. In these ISO instructions identify, in them, an expression or a variable. That is because in origin ISO code accepted only numeric fields, therefore to remain compatible at this standard but to have the possibility of a powerful and more flexible language, Isous introduced square brackets [ ] to identify an expression or variable.

ISO INSTRUCTIONS WHERE ARE PERMITTED SQUARE BRACKETS [ ]

Axis positions	**X,Y,Z,A,B,C,U,V,W - DX,DY,DZ,DA,DB,DC,DU,DV,DW**
Axis Feed	**F**
Speed	**S**
Tool diameter	**D**
Arc Center	**I,J**
Arc Radius	**R**
Tool	**T**
Head	**H**
Origin index	**USER_ZERO**
Offset index	**USER_OFFSET**

Ex:

```
G1X1000                    // STANDARD ISO GCODE
G1DX1000                   // INCREMENTAL VALUE (THE SAME G91)
G1X[$VAR]                  // USE OF VARIABLE AS AXIS POSITION
G1X[$VAR+$VAR1*SQR(18)]    // USE OF EXPRESION AS AXIS POSITION
```

15.3 VARIABLE TEST

To test variable Isous uses classic conditional operators **= > < <> >= <= !**
They are inserted in **IF ELSE END_IF** cycles allowing to execute cycles conditioned to one or more variable. Equal operators (=) can be combined to other logical operators OR (||) AND (&&). Round brackets can be used to prioritize expressions.

Ex:

```
IF $VAR=0 && $VAR1=5
        G1X100
ELSE
        G1X0
END_IF
```

15.4 BIT CONTROL

Isous makes available also operator to control the state of a single bit of a variable. They can be used in **IF ELSE END_IF** cycles or to **set/reset** single bit of a variable. It is also possible make a shift of the bits (left or right) and negate its state. All bit operations can be made only on INTEGER variables (excluding therefore decimal part).

Ex:

```
IF $VAR & 4                 // TEST BIT 3 OF $VAR
        G1X100 // BIT IS ON
ELSE
        G1X0                // BIT IS OFF
END_IF

$VAR=$VAR | 1   // SET BIT 1 OF $VAR
$VAR=$VAR & 1   // RESET ALL BITS EXCEPT BIT 1 OF $VAR

$VAR=1
$VAR=$VAR<<3   // $VAR IS SHIFTED LEFT 3 TIME
```

16 VARIABLE TYPES

Isous introduced VARIABLE to increase programming facilities. There are two types of variable: INTEGER o DOUBLE (floating point). Variables can be combined in expression as all other common languages.

16.1 NUMERIC CONSTANTS

All explicit numeric values are numeric constants. Decimal point can be used.

Ex:

```
G1X100.13
$VAR=25.143
```

16.2 GENERIC VARIABLES "$"

All generic variable are DOUBLE type and they can be used to store values inside PartProgram. When they are inserted in ISO instructions (X,Y,Z ecc) it must be delimited between **square brackets […].**

Minimum value **- 1.79769313486232 E308**
Maximum value **1.79769313486232 E308**
The maximum number of available variables is 2048 and are recognized by $ prefix followed by an ALPHANUMERIC NAME.

Ex:

```
$VAR=1
$VAR=$VAR1*5
$VAR=SQR($VAR1)
G1X[$VAR]
```

16.3 VARIABLES FOR ADDRESS

The Variables management can be do by **ADDRESS**, by a value from 0 to 32767..

:1000=10 Assign the value 10 to variable with ADDRESS 1000

WARNIG

The variables by ADDRESS are shared by variables by NAME ($), the compiler assign to $ an address from 0 to 32767 by the number of variables in the Gcode file. May be that the variable by ADDRES is overlapped to variable by NAME.

Recommended use address from 10000

16.4 VARIABLES FOR POINTER

Like to variables for ADDRESS, but in this case the ADDRESS is assigned by a Variable for NAME ($).

$VAR=1000

:$VAR=10 Assign the value 10 to variable with ADDRESS 1000

$VAR++

:$VAR=20 Assign the value 10 to variable with ADDRESS 1001

The type of variables can generate a RUN TIME ERROR

16.5 DATA STRUCTURES

IsoUs can manage the Variables in Data Structure. The Data Structure are used for organizing the **$** variables. This allows to simplify the GCODE programmation.

The Data Structures are always **GLOBAL** therefore are visible in the **M functions**.

These are created by **UsStructManager** (see documentation in Promax store).

A structure is defined by a **CATEGORY** (structure name) and one or more **VARIABLES** (data of structure).

Ex.

NAME	***TOOLPAR***
VARIABLES	***PAR1***
	PAR2
	PAR3

In **GCODE** is used the following syntax:

$TOOLPAR.PAR1=10

$TOOLPAR.PAR2=10+$TOOLPAR.PAR1

The data structures are always available once defined by **UsStructManager**

16.6 AXIS POSITION VARIABLES

These variables contain the position value of axis and accordingly they are read only.
The maximum number of these variables is 9. They can be used in PartProgram to read axis position in a given time.

Demand axis position - ReadOnly

$[Qn] With **n** from **0** to **8** it identifies the axis. The value of **DEMAND POSITION** is written in the variable and it refers to a **RELATIVE** value inclusive of WORK ORIGIN and OFFSET
With **n** from **100** to **108** it identifies the axis (subtracting 100). . The value of **DEMAND POSITION** is written in the variable and it refers to an **ABSOLUTE** position from HOME position

Actual axis position - ReadOnly

$[Rn] With **n** from **0** to **8** it identifies the axis. The value of **ACTUAL POSITION** is written in the variable and it refers to a **RELATIVE** value inclusive of WORK ORIGIN and OFFSET
With **n** from **100** to **108** it identifies the axis (subtracting 100). . The value of **ACTUAL POSITION** is written in the variable and it refers to an **ABSOLUTE** position from HOME position

Relative position Absolute position

n=0 AXIS X	n=100 AXIS X
n=1 AXIS Y	n=101 AXIS Y
n=2 AXIS Z	n=102 ASSE Z
…etc.	…etc.

Ex:

```
IF $[Q0]>100.3    // TEST OF X AXIS DEMAND POSITION
        G101
END_IF
$VAR=$[R1]                // READ ACTUAL POSITION OF Y AXIS
```

16.7 DIGITAL INPUT/OUTPUT VARIABLES

These variables allow to read all digital input or set/reset all digital output of CN. Rember digital output can be shared with PLC cycle on CN. This facilities is useful to make a simple PLC structure directly from ISO language without CN programming. I/O variables consider only 0 and 1 values. ***These variables when inserted in PartProgram allow direct access to I/O without waiting axis movement sync. Therefore in specific cases can need to use G62 code (wait end movement) to have sync between axis and I/O.***

Input Variables – Read only

$[In] Where **n** is a value from **0** to **255** and it identifies CN input
Value=0, Input OFF - Value=1, Input ON

Output Variables – Read/Write

$[On] Where **n** is a value from **0** to **255** and it identifies CN output
Value=0, Input OFF - Value=1, Input ON

Ex:

```
IF $[I0]=1                    // IF 1st INPUT IS ACTIVE
        $[O5]=1       //      SET 6th OUTPUT
ELSE
        $[O5]=0       // ELSE CLEAR IT
END_IF
```

16.8 TIMER VARIABILES

These variables are useful to set system TIMERS. Timers can be used to make delays, time-outs, etc. The maximum numbers of timers available is 10. TIMERS are written with a time in milliseconds. After TIMER is set, it starts automatically to decrement until its value reach 0.

Timer Variables – Read/Write

$[Tn] Where **n** is a value from **0** to **9** and it identifies the timer number

Ex:

```
$[T0]=1500              // SET TIMER 0 AT 1550 MILLISECONDS (1.5 SECONDS)
@CICLO
IF $[I0]=1
        GOTO EXIT       // EXIT IF INPUT 1 IS ON
END_IF
IF $[T0]=0              // TEST TIMER
        ERROR 10        // IF TIMER ELAPSES STOP PARTPROGRAM WITH ERROR 10
END_IF
GOTO CICLO              // REPEAT CYCLE
@EXIT
```

16.9 TOOL TABLE VARIABLES

They allow to read PARAMETERS from selected TOOL TABLE with **Tn**. The list below describes all parameters available for each TOOL TABLE and, as showed, they are up to maximum of 20 parameters per each tools.

Tool table variable – Read/Write
$[Un] Where **n** is a value from **0** to **19** indicating the parameter address

Ex:
$VAR=$[U0] // READ DIAMETER OF THE SELECTED TOOL

Address	Description
0	Tool diameter
1	Tool length
2	Maximum tool rpm speed
3	User Define 1
4	User Define 2
5	User Define 3
6	User Define 4
7	User Define 5
8	User Define 6
9	User Define 7
10	User Define 8
11	User Define 9
12	User Define 10
13	User Define 11
14	User Define 12
15	User Define 13
16	User Define 14
17	User Define 15
18	User Define 16
19	User Define 17

16.10 HEAD TABLE VARIABLES

These variables allow to read and write parameters of the selected HEAD TABLE with **Hn**. The list below describes all parameters available for each HEAD TABLE and, as showed, they are up to maximum of 20 parameters per each head.

Tool table variable – ReadOnly
$[Hn] Where **n** is a value from **0** to **19** indicating the parameter address

Ex:
$VAR=$[H0] // READ OFFSET X

Address	Descrizione
0	Head offset from home position of axis X
1	Head offset from home position of axis Y
2	Head offset from home position of axis Z
3	Head offset from home position of axis A
4	Head offset from home position of axis B
5	Head offset from home position of axis C
6	Head offset from home position of axis U
7	Head offset from home position of axis V
8	Sensor offset from 0 head of axis W
9	Sensor offset from 0 head of axis X
10	Sensor offset from 0 head of axis Y
11	Sensor offset from 0 head of axis Z
12	Sensor offset from 0 head of axis A
13	Sensor offset from 0 head of axis B
14	Sensor offset from 0 head of axis C
15	Sensor offset from 0 head of axis U
16	Sensor offset from 0 head of axis V
17	Sensor offset from 0 head of axis W
18	User Define 1
19	User Define 2

16.11 AXIS COUNTER VARIABLES

They are useful to read axis position from the interpolator. They don't contain the actual axis position but the position where it will be after position command. Generally these variable anticipate axis position before it has reached it. That is very useful for block restart facilities.

Axis counter variables – ReadOnly
$[Cn] Where **n** is a value from **0** to **8** indicating the axis

17 LIMIT AXIS VARIABLES

They are useful to read the limit axis setted

Positive Limit – ReadOnly
$[Nn] Where **n** is a value from **0** to **8** indicating the axis

Negative Limit – ReadOnly
$[Sn] Where **n** is a value from **0** to **8** indicating the axis

18 PREVIEW PARAMETERS

This variable type reads or sets the Preveiw Parameters (simulation Gcode)

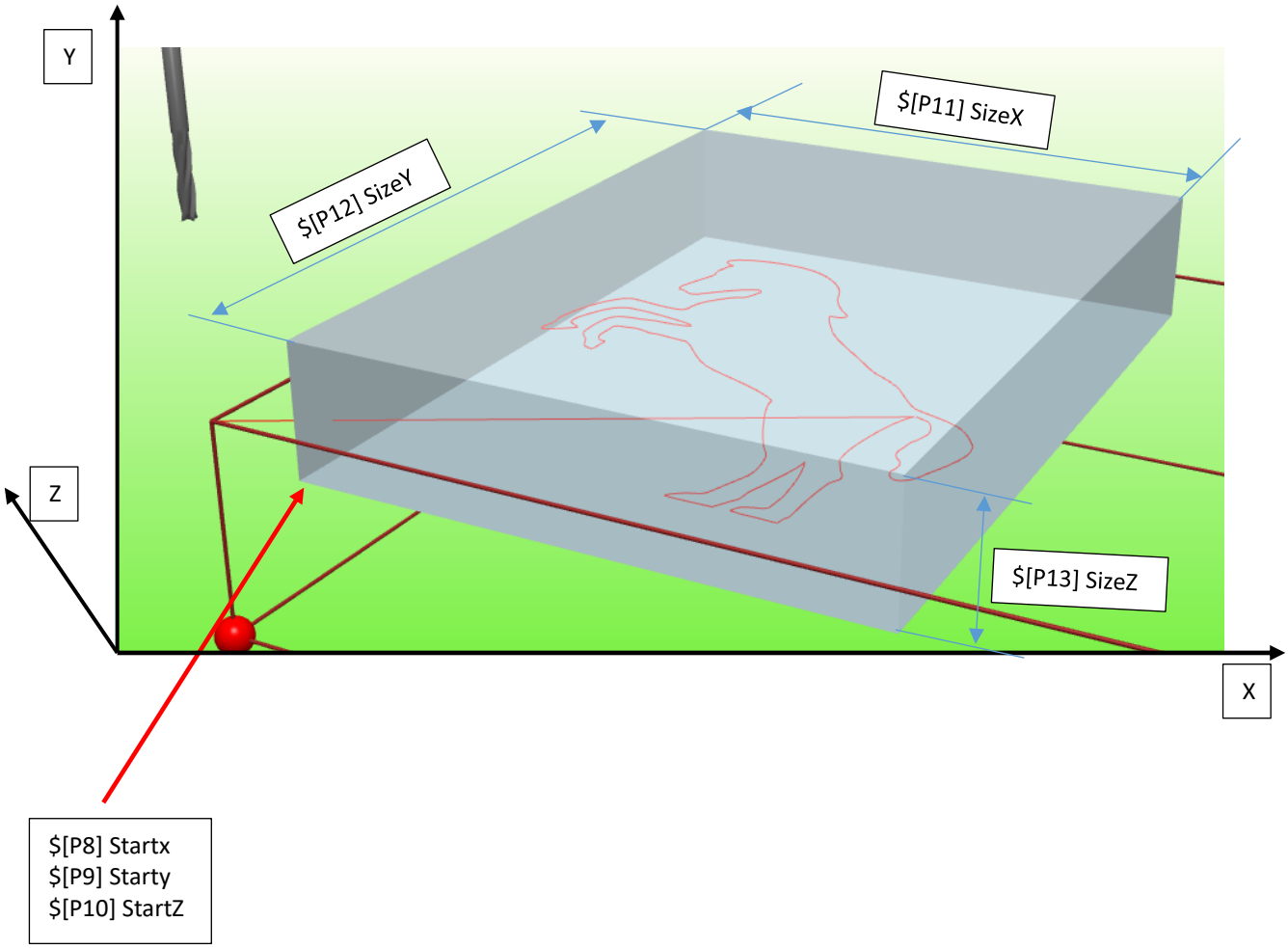
$[Pn] Where "**n"** is a value from 0 to 16 that indicates the parameter number

Pn	Descrizione	Valori
0	Enable/Disable Rotative Axis	0 Disable 1 Enable
1	Set axis Z direction for rotative Axis	0 Positive direction downward 1 Negative direction downward
2	Simulation Type	0 Line 1 Mesh 2 Lathe 3 Close Mesh 4 LayerT
3	Enable/Disable Heads Offset	0 Disable 1 Enable
4	Select cursor type	0 Pointer 1 RTCP 10-49 Type Tools where 10 Tool0-11 Tool1 etc. 50-99 Type Blade where 50 blade0 -51 Blade1 etc
5	Index Axis Blade	from 0 to Axis number-1
6	Offset Blade	From 0 to 360
7	Tool Diameter	Positive value only
8	Box Start X	See figure below
9	Box Start Y	See figure below
10	Box Start Z	See figure below
11	Box Size X	See figure below
12	Box Size Y	See figure below
13	Box Size Z	See figure below
14	Show box	Value >0 shows the box with parameters P8..P13. This parameter must be written after the parameters P8..P13 are written
15	LASER ON/OFF for RMS (Real machine Simulation)	Value>0 Enable Laser On for simulation focus point of laser in Machine Simulation
16	TOOL ON/OFF for RMS (Real machine Simulation)	Value>0 Enable Tool Spindle On for simulation point of tool contact in Machine Simulation
17	TEST COLLISION ON/OFF for RMS (Real machine Simulation)	Value>0 Enable Test Axes Collision in Gcode Run
18	TOOL LENGTH for RMS (Real machine Simulation)	Value for update the tool length ex: after the G102
19	SHOW MODEL 3D BOX for RMS (Real machine Simulation)	Value>0 show 3D model piece

20	Depth MODEL 3D BOX for RMS (Real machine Simulation)	Depth 3D model piece
21	Width MODEL 3D BOX for RMS (Real machine Simulation)	Width 3D model piece
22	Height MODEL 3D BOX for RMS (Real machine Simulation)	Height 3D model piece
23	Acq like G102 for RMS (Real machine Simulation). If SHOW MODEL 3D BOX Acq on Up of the Box, else on Up of the machine plane	Value = 1 Acq Value = 0 Reset
24	OFFSET on $[P23]	Add an offset on Acq by $[P23]
25-36	RESERVED	Read MinX,MaX,MinY,MaxY,MinZ,MaxZ MaX1,MinY1,MaxY1,MinZ1,MaxZ1
37	Preview Laser Raster	Value =1 Enable Preview Laser Raster Mode In $[p7] Diameter Spot Laser Value = 0 Disable

Show box for piece dimensions

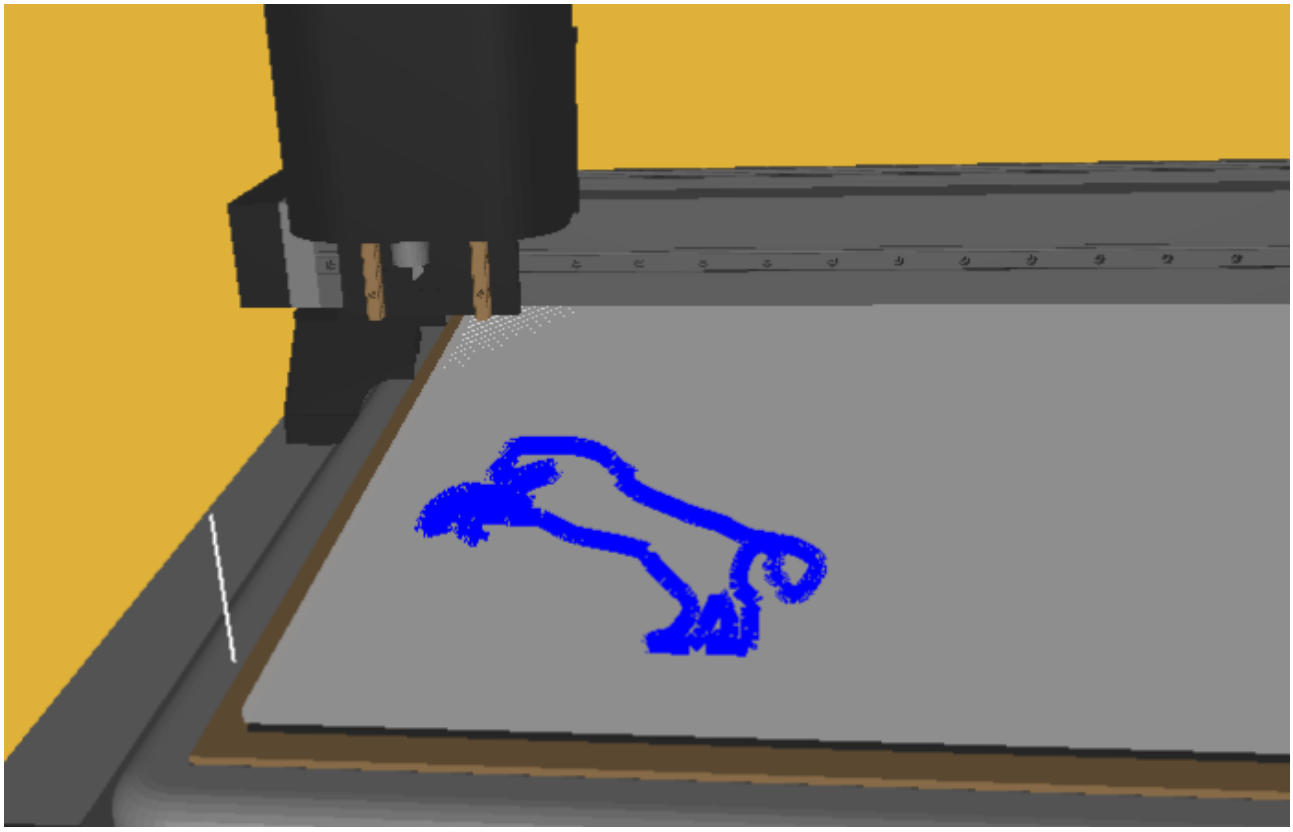
With the parameters P8..P14 is possible draw a Box for emulate the piece dimensions in working



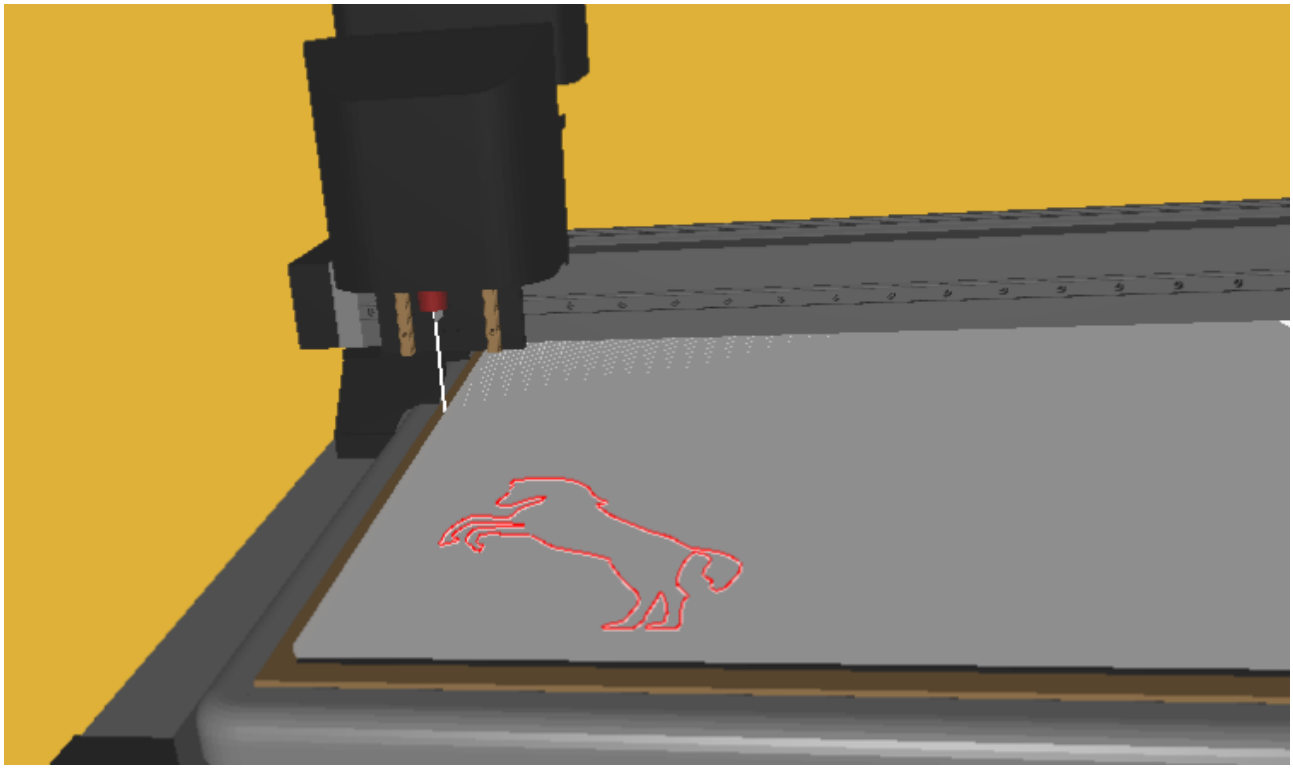
Example:

```
$[P8]=50      // START X
$[P9]=150     // START Y
$[P10]=0      // START Z
$[P11]=300    // SIZE X
$[P12]=200    // SIZE Y
$[P13]=70     // SIZE Z
$[P14]=1      // SHOWS BOX
```

Draw with Tool On outside of piece



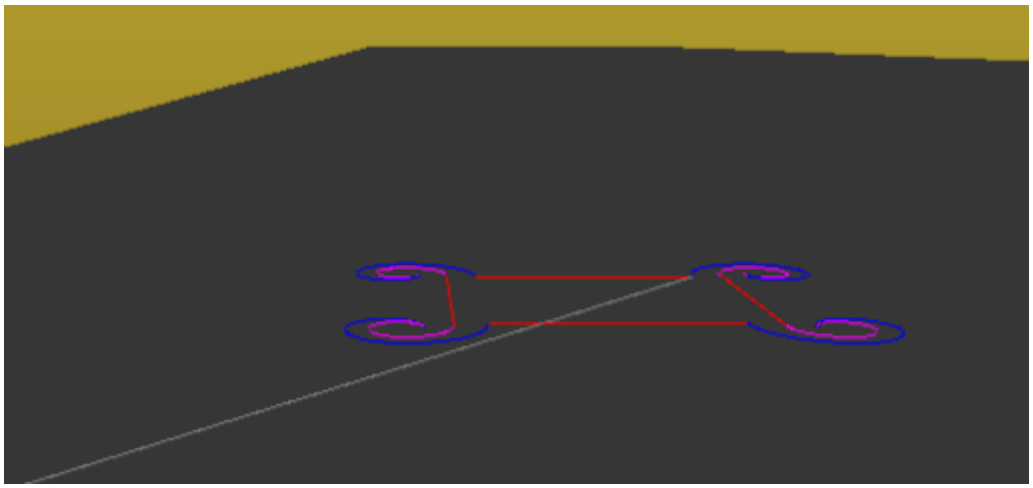
Draw with Tool On inside of piece



Acq Piece origin by $[P23]

Ex:

```
H2
$[P23]=1
G1X0Y0
G60
G0X202.018Y271.361
G2X203.211Y280.401I236.814J271.368
.
.
```



18.1 GLOBAL VARIABLE

GLOBAL prefix can be applied only to GENERIC variables ($name) and to use it in M or HM functions. GLOBAL prefix must be used when we want to share a variable between PartProgram and M/HM function, because by default Isous considers LABEL and VARIABLE private.

Ex:

```
//--- EXAMPLE FOR CODE OF PAUSE MACRO
//DEFINE $SAVEX AND $SAVEY AS GLOBAL
GLOBAL $SAVEX
GLOBAL $SAVEY
G62                          // WAIT FOR AXIS IN POSITION
$SAVEX=$[Q0]    // SAVE AXIS X POSITION
$SAVEY=$[Q1]    // SAVE AXIS Y POSITION
G0X0Y0

//--- EXAMPLE FOR CODE OF RESTART FROM PAUSE MACRO
//DEFINE $SAVEX AND $SAVEY AS GLOBAL
GLOBAL $SAVEX
GLOBAL $SAVEY
G0X[$SAVEX] Y[$SAVEY]   // MOVE AXIS TO POSITION SAVED IN PAUSE MACRO
G62
```

18.2 M FUNCTION VARIABLES (FOR CN)

These variable are useful to pass parameter to M functions defined in CN. That allows to condition M cycle by the value of them. The parameters must be write before to call M function. The number of passing parameters is defined in configuration up to maximum of 10 (default 5 parameters).

ATTENTION:

The CN get these parameters as integer value.

Ex:

```
// RUN M FUNCTION ON CN PASSING 2 PARAMETER
$_PARM_1=100
$_PARM_2=300
M15
```

18.3 SPECIAL PARAMETERS VARIABLES

They serve to read some parameters of Isous. The list below shows the readable parameters with these variables.

$[X0] **Read the current value of SPEED selected by S function - *ReadOnly***
$[X1] **Read the index of current WORK ORIGIN - *ReadOnly***
$[X2] **Read the index of current WORK OFFSET - *ReadOnly***
$[X3] **Read state of WORK ORIGIN – Value 1 if enabled or 0 if disabled - *ReadOnly***
$[X4] **Read state of WORK OFFSET – Value 1 if enabled or 0 if disabled – *ReadOnly***
$[X5] **Read the number of current head selected by H function – *ReadOnly***
$[X6] **Read the number of current tool selected by T function – *ReadOnly***
$[X7] ***Read The RUN Type – ReadOnly***

0 NORMAL RUN
1 SIMULATION RUN
2 GO BLOCK RUN
3 RETRACE RUN
4 GCODE EXECUTION TIME CALCULATION RUN

X8 Read Last FEED – ReadOnly
X9 Read coordinate mode – ReadOnly
0 G90 Absolute
1 G91 Relative
X10 Read work plane actived – ReadOnly
0 XY G17
1 XZ G18
2 YZ G19
-1 other
X11 Read tool length offset set – ReadOnly
0 None
1 Value (+/-) tool length offset set
X12 Read axis tool length offset set – ReadOnly
0 Axis X
1 Axis Y
2 Axis Z
3 Axis A
4 Axis B
5 Axis C
6 Axis U
7 Axis V
8 Axis W
X13 Read the button pressed by LIB MESSAGE – ReadOnly
X14 Read the type of GoBlock invoked – ReadOnly
0 GoBlock from line or marker
1 GoBlock from M6Tn
X15 Read the Axis Index selected for JOG (Ex: 0=X – 1=Y etc.) – ReadOnly
X16 Read the Axis status in Velocity mode – ReadOnly
0 Axis Stop
1 Axis in Acceleration
2 Axis in Deceleration
3 Axis in Velocity reached
X17 Read Run from CMD – ReadOnly
0 NORMAL RUN (no CMD)
1 RUN CMD
2 RUN SCRIPT CMD
X18 Parameter P1 CMD – ReadOnly
X19 Parameter P2 CMD – ReadOnly
X20 Parameter P3 CMD – ReadOnly
X21 Parameter P4 CMD – ReadOnly
X22 Parameter P5 CMD – ReadOnly
X23 Parameter P6 CMD – ReadOnly
X24 Parameter P7 CMD – ReadOnly
X25 Parameter P8 CMD – ReadOnly
X26 Parameter P9 CMD – ReadOnly
X27 Parameter P10 CMD – ReadOnly
X28 Parameter X Canned Cycles – ReadOnly
X29 Parameter Y Canned Cycles – ReadOnly
X30 Parameter Z Canned Cycles – ReadOnly
X31 Parameter R Canned Cycles – ReadOnly
X32 Parameter P Canned Cycles – ReadOnly
X33 Parameter Q Canned Cycles – ReadOnly
X34 Parameter K Canned Cycles – ReadOnly
X35 Parameter F Canned Cycles – ReadOnly

X36 ***Parameter A Canned Cycles – ReadOnly***
X37 ***Parameter B Canned Cycles – ReadOnly***
X38 ***Reserved***
X39 ***Read G43 Status – ReadOnly***
0 Not Activated
1 Activated
X40-X48 ***Read Home Stae Axis (X,Y,Z,A,B,C,U,V,W) – ReadOnly***
0 Home Not Performed
1 Home Performed
X49-X57 ***Read Enable stae Axis (X,Y,Z,A,B,C,U,V,W) – ReadOnly***
0 Enable Not Performed
1 Enable Performed
$[X58] Read state of SUSPENSION WORK ORIGIN – Value 1 if suspended or 0 if not - *ReadOnly*
$[X59] Read state of SUSPENSION WORK OFFSET – Value 1 if suspended or 0 if not – *ReadOnly*
$[X60] Read current state of STOP_MODE - *ReadOnly*
$[X61] Read current state of PAUSE_MODE – *ReadOnly*
$[X62] Read current state of FILTER_MODE - *ReadOnly*
$[X63] Read horizontal mirror state - *ReadOnly*
0 Disable
1 Enable
$[X64] Read vertical mirror state - *ReadOnly*
0 Disable
1 Enable
$[X65] Read G from Retrace – *ReadOnly*
0 G0
1 G1
2 G2
3 G3

18.4 WORK ORIGIN AND OFFSET VARIABLES

These variables contain the current position of WORK ORIGIN and WORK OFFSET. The maximum number of these variables is 9 (one for each axis). They can be used in PartProgram to read the current zero positions.

Work Origin Variable - ReadOnly

$[Yn] Where **n** is a value from **0** to **8** indicating the axis. The variable will contain the current position of WORK ORIGIN of axis indicated by n

Work Offset Variable - ReadOnly

$[Wn] Where **n** is a value from **0** to **8** indicating the axis. The variable will contain the current position of WORK OFSET of axis indicated by n.

n=0 axis X, n=1 axis Y, n=2 axis Z, etc.

18.5 USER GENERIC VARIABLES

These variables are used to exchange data between CN and Isous. The maximum number available is **30**. They can be read or written and are 32 bit INTEGER type.

User generic variables - Read/Write

$[Kn] Where **n** is a value from **0** to **29** indicating the variable

18.6 ARRAY VARIABLES - DIM

These variables are used like to USER GENERIC VARIABLES, but these, are **get/set** by Index.

First to use, is necessary declare the ARRAY DIMENSION with Function DIM, and the VARIABLE ARRAY NAME:

Ex:

DIM $ARR 10

In the above example, was declared an ARRAY with name ARR with 10 elements.

The ELEMENTS are indexed from 0 to 9

ARRAY USAGE

The ARRAY VARIABLES contain the same values of GENERIC VARIABLES (double).

For Get/Set the value from ARRAY, the INDEX value must be enclosed between **"[..]"**.

Ex:

```
DIM $ARR 10
$INDEX=0
$VAR=2
LOOP 10
      $ARR[$INDEX]=$VAR*$INDEX
      $INDEX=$INDEX+1
END_LOOP
$INDEX=0
LOOP 10
      G1 X[$ARR[$INDEX]]
      $INDEX=$INDEX+1
END_LOOP
```

18.7 MARKER VARIABLES

Markers are normal variables written in PartProgram. In enhanced programming, using LOOP cycles and VARIABLES, restart from a LINE NUMBER is not enough, because axis position can be defined by variables written in a LOOP cycle. Using MARKERS it's possible restart PartProgram at value of one of these, not at a specific line number, allowing restart also inside LOOP cycles. MARKERS are defined with instruction **MARKER $NAMEVAR DESCRIPTION.**
Markers can be restored from PlugIn "**BLOCK RESTART"**
Ex:
In the next example it is defined a Marker variable named **$INC**.
It specifies a counter of the number of piece in working

```
MARKER $INC PIECE NUMBER
$VAR=0
$INC=0
F5
G1X0Y0
LOOP 10
        $INC=$INC+1
        G1X200
        $VAR=$VAR+50
        G0X0Y[$VAR]
END_LOOP
```

It will be possible restart PartProgram when **$INC** variable (set as MARKER) takes a specific value.

18.8 VARIABLE FOR ANALOG SPINDLE OUTPUT

Allows to write in the Analog Output a value generally used for the **SPINDLE SPEED**.

Variable Analog Output - Read/Write

$[A0]=val Where **val** depends of the Channel resolution

SETTABLE CHANNEL

Are available up to **16** different channels. These are set from **MACHINE PARAMETERS** table **SPINDLE** by parameter **SPEED_ANALOG_CH**.

The Channel from 0 to 15 are refer to Analog output on the **NGIO-NGPP** for **NGWARP** or **NGMSX** for **NGMEVO**. The **Channel 0** is the first analog output of the **FIRST** board **NGIO-NGPP-NGMSX**, the **Channel 1** is the second analog output of the **FIRST** board **NGIO-NGPP-NGMSX**, The **Channel 2** is the first analog output of the **SECOND** board **NGIO-NGPP-NGMSX etc**

The Channel **NGMEVO PWM** is the Analog Output only for **NGMEVO** (optional)

The Channels **SPEED_X,Y,Z,A,B,C,U,V,W** are referred to control the Axis in SPEED MODE for the function **G108.4 e G108.5**.
These indicate which axis is connected to **G108.4**
Ex: **SPEED_X**, the **G108.4** function, is connected to Axis X and the **S** function set the speed for this axis

CHANNELS RESOLUTION

The Channels have the following resolution**:**
Channel 0 a 15 **12 bit** **Value from 0 to 2047**
Channel NGMEVO PWM **8 bit** **Value from 0 to 255**
(WARNING The PWM Channel can saturate ,i.e. reach the 10V before of 255 value)

The Channel resolution is set from **MACHINE PARAMETERS** table **SPINDLE** by parameter **ANALOG_BIT_RES**.
Therefore set the following values:
Channel 0 -15 **2048**
NGMEVO PWM **Max 256 generally a value from 210 to 230**
(First to all set this value to 256 , check by write **$[A0]** with a value from 210 to 255, when the ouput is 10V this is the correct value for the parameter **ANALOG_BIT_RES**)

Example M3 CW SPINDLE

```
READ_PARMAC "SPEEDMAXSPINDLE" $MAX_RPM // READ MAX RPM AT 10V
READ_PARMAC "ANALOG_BIT_RES" $BIT_RES // READ BIT RESOLUTION
$CURRENT_SPEED=$[X0] // READ THE CURRENT SPEED SET BY S GCODE FUNCTION
$ANALOG_OUT=$CURRENT_SPEED*$BIT_RES/$MAX_RPM  // CALC
$[A0]=$ANALOG_OUT // WRITE
$[O1]=1 //SET CW DIRECTION
$[O2]=1 // START
```

18.9 MACRO and Gcode Parameters MANAGEMENT

The **J** variable (Read/Write) allows to Management the internal Macro

$[Jn]	Valore	Descrizione
$[J0]	**1**	Suspend: **Mstop,Merror,Mpause,MEndProgram,Mtime,MinterruptDI, MinterruptDO, M50003,M50004,M60001,M60002,M60003,M60004**
	0	Resume: **Mstop,Merror,Mpause,MEndProgram,Mtime,MinterruptDI, MinterruptDO, M50003,M50004,M60001,M60002,M60003,M60004**
$[J1]	**1**	Suspend: **Mstop**
	0	Resume: **Mstop**
$[J2]	**1**	Suspend: **Merror**
	0	Resume: **Merror**
$[J3]	**1**	Suspend: **Mpause**
	0	Resume: **Mpause**
$[J4]	**1**	Suspend: **MEndProgram**
	0	Resume: **MEndProgram**
$[J5]	**1**	Suspend: **Mtime**
	0	Resume: **Mtime**
$[J6]	**1**	Suspend: **MinterruptDI**
	0	Resume: **MinterruptDI**
$[J7]	**1**	Suspend: **MinterruptDO**
	0	Resume: **MinterruptDO**
$[J8]	**1**	Suspend: **M50003**
	0	Resume: **M50003**
$[J9]	**1**	Suspend: **M50004**
	0	Resume: **M50004**
$[J10]	**1**	Suspend: **M60001**
	0	Resume: **M60001**
$[J11]	**1**	Suspend: **M60002**
	0	Resume: **M60002**
$[J12]	**1**	Suspend: **M60003**
	0	Resume: **M60003**
$[J13]	**1**	Suspend: **M60004**
	0	Resume: **M60004**
$[J14]	**1**	Suspend: **STOP – Emergency Only**
	0	Resume: **STOP**
$[J15]	**1**	Suspend: **PAUSE**
	0	Resume: **PAUSE**
$[J16]	**1**	Activates automatic counter axes update when used Hn or G43
	0	Deactivates automatic counter axes update when used Hn or G43
$[J17]		RESERVED
$[J18]		RESERVED
$[J19]		RESERVED
$[J20]	**0**	**ABSOLUTE** center for G2 G3 like to **ARC_REL** parameter
	1	**RELATIVE** center for G2 G3 like to **ARC_REL** parameter
$[J21]	**0-13**	Buffer dimension for G66 X-100 (0 defualt value) Values from 0 to 13 0-13 **1**-16 **2**-32 **3**-64 **4**-128 **5**-256 **6**-512 **7**-1024 **8**-2048 **9**-4096 **10**-8192 **11**-16384 **12**-32768 **13**-65536
$[J22]	**0-1**	Value =1 Enable conversion **S** in **G100 for Laser Power** **SVal->G100 XVal** Value=0 Disable (at Run always disable)

19 POSITIONER

Isous provides the complete management of one or more axes positioners. These can be of different type and in any case their management is entrusted to the application of CNC in use VTB

20 PA-PD(n,par)espr Set Absolute position

PA allows movement of the positioner at **ABSOLUTE** target
PD allows movement of the positioner at **RELATIVE** target

n Positioner NUMBER from 0 to MAX
par Controlo parameter OPTIONAL
- **null** Start at TARGET **ESPR** with Pending end of movement
- **0** Start at TARGET **ESPR** without **Pending end of movement**
- **1** Start at TARGET **ESPR** with **Pending end of movement**
- **2** Start HOMING (Espr must be present but not used)
- **3** Espr >0 Enable positioner
  Espr <=0 Disable positioner

Ex:
PA(0)1000 // START AT TARGET 1000 POSITIONER 0
PA(0,0)1000 // START AT TARGET 1000 without pending end of movement
PA(0)[$VAR] // START AT TARGET$VAR i
PA(0,2)1// START HOMING
PA(0,3)1// ENABLE POSITIONER 0

21 PF(n)espr Set Positioner FEED

Set the FEED positioner. Espr FEED VALUE
n Positioner NUMBER from 0 to MAX

Ex:
PF(0)100 //FEED 100
PF(0)[$VAR] // FEED $VAR

22 PS(n) STOP Positioner

Stop POSITIONER
n Positioner NUMBER from 0 to MAX
Ex:
PS(0) // Stop positioner 0

23 _PM(n,par) Read status Positioner

Read the status postioner
n Positioner NUMBER from 0 to MAX
par Parameter
- **0** Return 0 Axis Stop fermo - 1 Axis movment
- **1** READ demand position
- **2** READ actual position
- **3** Return 1 axis enabled – 0 axis disabled
- **4** Return 1 HOMING make– 0 HOMING not make

5 Return 1 axis in ALLARM – 0 OK

Ex:

$VAR=_PM(0,par)

24 WORK SETTING

In this charter are defined all axis motion functions.

24.1 DEFINITION OF AXIS POSITIONS

They define axis positions in the several interpolation functions (G0-G1-G2-G3). The axis code can be followed by a numeric value or by an expression identifying the position.

AXIS CODES

X ***Axis X***
Y ***Axis Y***
Z ***Axis Z***
A ***Axis A***
B ***Axis B***
C ***Axis C***
U ***Axis U***
V ***Axis V***
W ***Axis V***
I ***Arc center for X axis of work plane***
J ***Arc center for Y axis of work plane***
K ***Arc center for Z axis of work plane (only for G18 -G19)***
R ***Arc radius***

Syntax
CODE value or expression
Ex:

```
X100Y20                    // POSITION BY NUMERIC VALUE
DX100Y20                   // RELATIVE AND ABSOLUTE
X[$VAR]Y[$VAR1]            // POSITION BY VARIABLES
X[COS($VAR)+$VAR1]   // POSITION BY EXPRESSION
```

24.1.1 Axis definition for Channel Address

QX	***Channel 0***	***QY***	***Channel 1***
QZ	***Channel 2***	***QA***	***Channel 3***
QB	***Channel 4***	***QC***	***Channel 5***
QU	***Channel 6***	***QV***	***Channel 7***
QW	***Channel 8***		

The axis position for channel address, allows to always refer to Axis not considering the AXES NAME
Ex if the axes are called in the following mode:

X	Channel 0
Y	Channel 1
Z	Channel 2
B	Channel 3
C	Channel 4
W	Channel 5

QX,QY etc. always refer to the channel and not at name
So
G1 X100 Y200 B300
Is like to
G1 QX100 QY200 QA300

24.1.2 G90 – PROGRAMMING WITH ABSOLUTE POSITIONS

Syntax

G90

Function type

MODAL (default)

Cancel

G91

Description

All following positions refer to current ZERO (HOME POSITION, WORK ORIGIN, WORK OFFSET), If no work origin or offset are used positions refer directly to home position.

24.1.3 G91 - PROGRAMMING WITH INCREMENTAL POSITIONS

Syntax

G91 G91.1 G91.2

Function type

MODAL

Cancel

G90

Description

All following positions refer to current axis position.

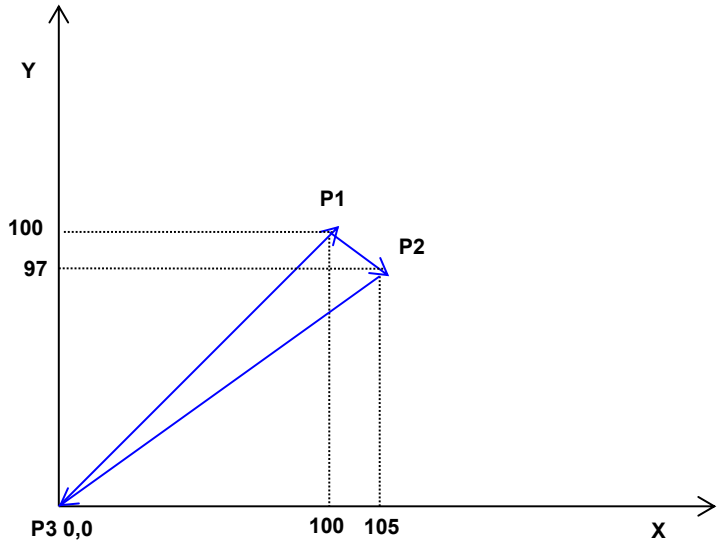
G91.1 Force the values in ABSOLUTE (G90) and store the current state (G90-G91)

G91.2 Resume the state stored with G91.1 (G90 if was G90 or G91 if was G91)

24.1.4 Definition of absolute and incremental positions

Ex:

```
G90                 // SET ABSOLUTE POSITIONS
G1X100Y100          // AXIS MOVE TO P1=100,100
G91                 // SET INCREMENTAL POSITIONS
X5Y-3               // AXIS MOVE TO P2=105,97
G90                 // SET ABSOLUTE POSITIONS
X0Y0                // AXIS MOVE TO P3=0,0
```



24.2 WORK ORIGIN

Isous can manage up to **256 different work origins** selectable by **G94, G92** and **USER_ZERO** instructions. A work origin define a point of reference in relationship with home position.

24.2.1 Work origin By INDEX

These origins can be set on the fly by taking the value contained in 'file index' ZERI.VAL ". Unlike USER_ZERO, this can have different indexes for each axis.

Syntax

OXIndex OYIndex ….

Function type

MODAL

Cancel

G92-G82

Parameters

Index –Index Value from file "ZERI.VAL"

Description

Defines the new origin of the axes based on the value contained in 'index set

Es:

```
OX0 OY1 OZ3      // SET origin X from INDEX 0, Y Index 1,Z Index 3
```

24.2.2 G94 – Work origin at position defined with parameter

Syntax

G94 Xvalue Yvalue …. (DX,DY ecc. Relative value)

Function type

MODAL

Cancel

G92-G82

Parameters

X,Y,Z,A,B,C,U,V,W (QX,QY,QZ,QA,QB,QC,QU,QV,QW) followed by origin position

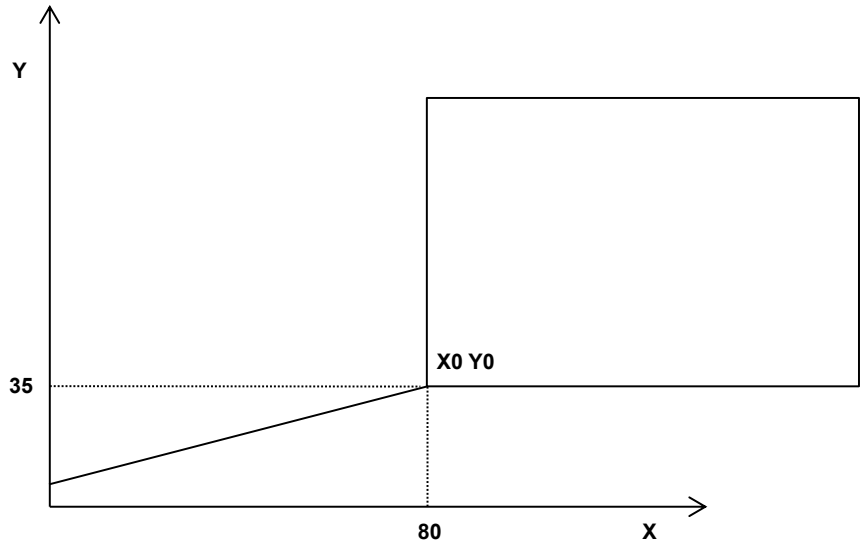
Description

It defines the new origin of one or more axis. Axis not included in parameters list keep the origin unchanghed. The new position of zero axis will be defined in the parameter value.

The position value in the parameter refer to an ABSOLUTE position from HOME POSITION regardless of **G90 G91**. The new origin is automatically enabled with G94.

Ex:

G94 X80Y35 // SET NEW WORK ORIGIN AT 80,35



24.2.3 G94.n – Save work orgini at Index

Syntax

G94.n X Y Z etc

Function type

MODAL

Parameters

X,Y,Z,A,B,C,U,V,W Axis to save

.n Number of INDEX ZERO (from 0 to 255).

Description

Save the current index origins for the axes selected

Ex:

```
G94.1 XY          // SAVE THE ORIGINS INDEX 1 FOR X,Y
```

24.2.4 G54, G55, G56, G57, G58, G59 - Work origin from memory file

Syntax

G54 (axis code)

G54 XYZ …. or G54.n XYZ - G54 QX QY QZ …. o G54.n QX QY QZ

Function type

MODAL

Cancel

G92-G82

Parameters

X,Y,Z,A,B,C,U,V,W (QX,QY,QZ,QA,QB,QC,QU,QV,QW)

.n Number of INDEX ZERO (from 0 to 255). If omitted, uses the parameter USER_ZERO to set index

Description

It defines the new origin of one or more axis. Axis not included in parameters list keep the origin unchanged. The new position of zero axis will be defined in the parameter value.
The position value will be get from the ORIGIN FILE made by **PLUGIN WORK ORIGIN.**
The new origin is automatically enabled with these instructions.
If **G54** is invoked without parameters **AXIS X** will be set.
The **G54** function, can use the INDEX ZERO indicate in the **.n**
G54.n where **n** is a value from 0 to 255 refered to INDEX ORIGIN
If the parameter **.n** is omitted, is used USER_ZERO to set index.
The .n parameter not changes the value of USER_ZERO parameter.

Similarly: (these can not use the **.n** parameter)

G55 for AXIS Y
G56 for AXIS Z
G57 for AXES XY
G58 for AXES YZ
G59 for AXES XZ

Unlike G94, these functions set WORK ORIGIN from a file saved on pc allowing reload of ORIGINS after switch-off.

Ex:

```
USER_ZERO 0 // uses INDEX 0 (Option  default)
G54 XYZ          // Set ORIGIN to  X Y Z
USER_ZERO 1 //  uses INDEX 1
```

```
G55       // Set origin Y
G54.2 XYZ          // SET Origin on XYZ by INDEX 2
```

24.2.5 G92 – Work origin in current axis position

Syntax

G92 XYZ etc.

Function type

MODAL

Cancel

G94-G82

Parameters

X,Y,Z,A,B,C,U,V,W (QX,QY,QZ,QA,QB,QC,QU,QV,QW) axis to be zero

Description

It defines the new origin of one or more axis in the current position. Axis indicated are zeroed in current position.

24.2.6 G82 – Work origin in current axis position with sensor offset

(Shifted on G1082 if USE_G80_CYCLES=TRUE)

Syntax

G82 XYZ G82 QX QY QZ etc.

Function type

MODAL

Cancel

G94-G92

Parameters

X,Y,Z,A,B,C,U,V,W (QX,QY,QZ,QA,QB,QC,QU,QV,QW) axis to be zero

Description

It defines the new origin of one or more axis in the current position taking into account acquisition sensor (G102). Axis indicated are zeroed in current position adding sensor offset.

This function must be used with origin acquisition cycles by G102.

24.2.7 G98 – G53 - Suspend work origin

Syntax

G98

G53

Function type

MODAL

Cancel

G82 ***G92 G94 -G54-G55-G56-G57-G58-G59***

Description

It suspends all work origins. All following positions will be refer to HOMING POSITION (if WORK OFFSETS G93 or G95 are disabled)

24.2.8 G99 – Restore work origin

Syntax

G99

Function type

MODAL

Cancel

G98

Description

Restore work origin set with G92 or G94, disabling G98.

24.2.9 G940 MOVE axes excluding WORK ORIGIN only in the actual block

Syntax
G940 G1Xval Yval

Function type
deleting

Description
All axis movments follow G940 are performed with reference to home axes (WORK ORIGIN EXCLUDES)

24.2.10 USER_ZERO – Index of WORK ORIGIN LIST

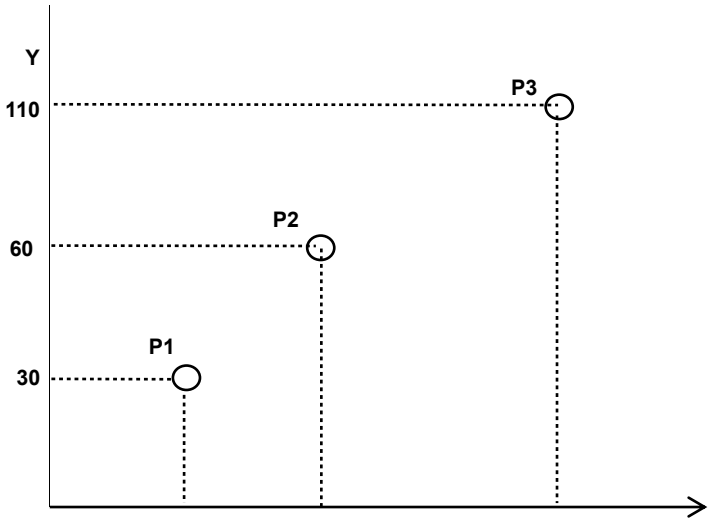
Syntax
USER_ZERO expression

Function type
MODAL
Parameters
Expression or value indicating the index of ORIGIN LIST to be used

Description
Work origin is enabled with the position indexed in the ORIGIN LIST by USER_ZERO.

Ex:

```
USER_ZERO 0
G94 X20Y20              // STORE ORIGIN 0
USER_ZERO 1
G94 X50Y50              // STORE ORIGIN 1
USER_ZERO 2
G94 X150Y100    // STORE ORIGIN 2
USER_ZERO 0             // ENABLE ORIGIN 0
G1X10Y10                // MOVE TO P1
USER_ZERO 1             // ENABLE ORIGIN 1
G1X10Y10                // MOVE TO P2
USER_ZERO 2             // ENABLE ORIGIN 2
G1X10Y10                // MOVE TO P3
```



24.3 WORK OFFSET

Isous can manage up to **256 different work offsets** selectable by **G93, G95** and **USER_OFFSET i**nstructions. A work offset is an additional work origin respecting G92 and G94 functions. As the last one these define a point of reference in relationship with home position. For example see WORK ORIGIN.

24.3.1 G93 - Work offset at position defined with parameter

Syntax

G93 Xvalue Yvalue …. (DX,DY ecc. Relative value)
G93 P(n)value DP(n)value …. for positioner
Or DOXvalue DOYvalue

Function type

MODAL

Cancel

G85 G95

Parameters

X,Y,Z,A,B,C,U,V,W (QX,QY,QZ,QA,QB,QC,QU,QV,QW) flollowed by offset position

Description

It defines the new offset of one or more axis. Axis not included in parameters list keep the offset unchanged. The new position of zero axis will be defined in the parameter value.
The position value in the parameter refer to an ABSOLUTE position from HOME POSITION regardless of **G90 G91**. The new offset is automatically enabled with G93.

24.3.2 G95 - Work offset in current axis position

Syntax

G92 XYZ etc.

Function type

MODAL

Cancel

G85 G93

Parameters

X,Y,Z,A,B,C,U,V,W (QX,QY,QZ,QA,QB,QC,QU,QV,QW) axis to be offset

Description

It defines the new offset of one or more axis in the current position. Axis indicated are zeroed in current position.

24.3.3 G85 - Work offset in current axis position with sensor offset

Syntax

G85 XYZ G85 QX QY QZ etc.etc.

Function type

MODAL

Cancel

G93-G95

Parameters

X,Y,Z,A,B,C,U,V,W (QX,QY,QZ,QA,QB,QC,QU,QV,QW) axis to be zero

Description

It defines the new offset of one or more axis in the current position taking into account acquisition sensor (G102). Axis indicated are zeroed in current position adding sensor offset.

This function must be used with origin acquisition cycles by G102.

24.3.4 G86 - Hardware preset axis on module 360 degrees

Syntax

G86 XYZ etc. G86 QX QY QZ etc.

Function type

IMMEDIATE

Parameters

X,Y,Z,A,B,C,U,V,W (QX,QY,QZ,QA,QB,QC,QU,QV,QW) axis to be preset

Description

The G86 function, is used for rotative axes.
It preset in hardware mode, the absolute axis position to the relative angle.
Ex:
I position axis is 1.450, the relative degrees is 10

24.3.5 G96 - Suspend work offset

Syntax

G96

Function type

MODAL

Cancel

G85 G93 G95

Description

It suspends all work offset. All following positions will be refer to HOMING POSITION (if WORK ORIGINS G92 or G94 are disabled).

24.3.6 G97 - Restore work origin

Syntax

G97

Function type

MODAL

Cancel

G96

Description

Restore work offset set with G93 or G95, disabling G96.

24.3.7 USER_OFFSET - Index of WORK OFFSET LIST

Syntax

USER_OFFSET expression

Function type

MODAL

Parameters

Expression or value indicating the index of OFFSET LIST to be used

Description

Work offset is enabled with the position indexed in the OFFSET LIST by USER_OFFSET.

Ex:

```
USER_ OFFSET 0
G94 X20Y20                  // STORE OFFSET 0
USER_ OFFSET 1
G94 X50Y50                  // STORE OFFSET 1
USER_ OFFSET 2
G94 X150Y100    // STORE OFFSET 2
USER_ OFFSET 0  // ENABLE OFFSET 0
G1X10Y10                    // MOVE TO P1
USER_ OFFSET 1  // ENABLE OFFSET 1
G1X10Y10                    // MOVE TO P2
USER_ OFFSET 2  // ENABLE OFFSET 2
G1X10Y10                    // MOVE TO P3
```

Refer to ORIGIN example for drawing.

24.4 Work head selection – H function

Isous can manage up to **256 different tools heads**.
Each head refer to home position by the value in the HEADS TABLE.
By default no head is enabled, with Hn function it's possible to select one of 256 available heads (first head =0). All offset (one for each axis) of the selected head will be automatically enabled.

24.4.1 Hn – Select head

Syntax
H expression
Function type
MODAL
Parameters
Expression or value identifying selected head (firts head 0)

Description
Offset of the selected head are enabled.

Ex:
```
H0              // SELECT FIRST HEAD
```

24.4.2 G87 – Suspend head offset

Syntax
G87
Function type
MODAL
Cancel
G88

Description
Offset of selected head are disabled.

Ex:
```
G87             // SUSPEND HEAD OFFSET
```

24.4.3 G88 – Restore head offset

Syntax
G88
Function type
MODAL
Cancel
G87

Description
Last enabled head offset are restored.

Ex:
```
G88             // RESTORE HEAD OFFSET
```

24.5 ROTATIVE AXIS

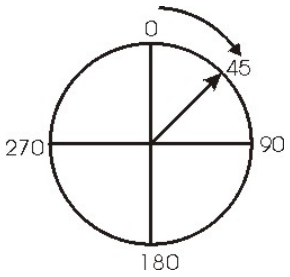
Isous can manage rotative axis in several modes. G36 function selects mode.

24.5.1 G36 – Rotative axis definition

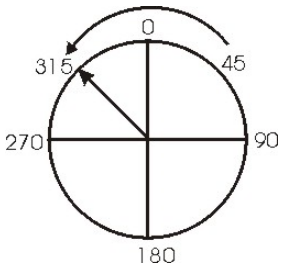
Syntax
G36 Xmode AXISNAME
Function type
MODAL
Parameters
Mode: 0 → Disable (set linear axis)
1 → Shorter path.
2 → Always positive
3 → Always negative
4 → Shorter path when G1, disabled when G2/G3
AXISNAME name of rative axis (ex A,B etc.)

Description

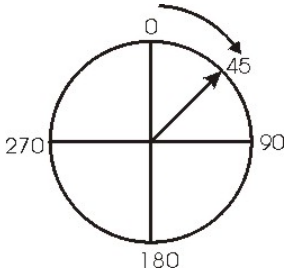
Mode 1,4 - Shorter path
CN chooses the shorter path to reach the position selected.
Mode 4 is similar to mode 1 but in G2/G3 interpolation this facilities is disabled.
Ex:
G36 X1 A
G1 A45
G1 A315
first movement **G1 A45**



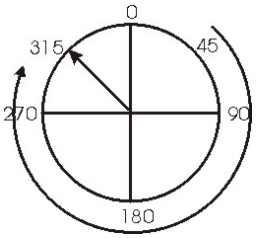
second movement **G1 A315**



Mode 2,3 – Always positive/negative
The selected axis rotates always in the same sense.
G36 X2 A
G1 A45
G1 A315
first movement **G1 A45**



second movement **G1 A315**



24.6 HARDWARE PRESET OF AXIS

In partucuar cases can be necessary preset axis positions in Hardware mode allowing to avoid Overflow in single direction axis.
Hardware preset must be used with particular attention, because if axis works with a following error, it doesn't take into consider. Consequently it's possible to lost encoder pulses and reference to home position.

24.6.1 G89 – Hardware preset of axis position

Syntax
G89 Xvalue Yvalue …. (DX,DY ecc. Relative value)
Function type
DIRECT
Cancel
All origin and offset functions
Parameters
X,Y,Z,A,B,C,U,V,W (QX,QY,QZ,QA,QB,QC,QU,QV,QW) followed by preset value
Description
It defines a new value for current axis positions.

24.7 HARDWARE PRESET OF AXIS

In some cases it is necessary to preset the counters of the axes after recovering from a PAUSE, or to RETRACE. This is because during the PAUSE or first RETRACE, the axes are moved using special functions (shift-type etc..).
Not properly updating the counters, abnormal movements may occur during the step of interpolation axes. Generally counters axes are always updated and therefore there is no need an update, but there are cases such as those listed above that require an update. Always use an upgrade at the end of the axis M of recovery from the PAUSE by RETRACE is still a good thing .

24.7.1 G84 – Preset Axes counters
(Shifted on G1084 if USE_G80_CYCLES=TRUE)
Syntax
G84 X Y Z G84 QX QY QZ….
Function type
DIRECT
Parameters
X,Y,Z,A,B,C,U,V,W (QX,QY,QZ,QA,QB,QC,QU,QV,QW) axes to update
Unless indicated no axis are updated all axes
Description
The G84 function copies the values of the coordinates axes by process (typically taken from PAUSE or RETRACE) the process of WORK.

24.7.2 G83 – Preset CPU 1 counters
(Shifted on G1083 if USE_G80_CYCLES=TRUE)

Syntax
G83
Function type
DIRECT

Description
The G83 function is only used to force the update of the counters of the CPU 1, the one that works on MACRO PAUSE, STOP, ERROR.
it is advisable to start with the MACRO code G83

24.8 CANNED CYCLES

IsoUs Use 4 Canned Cycles:
G81,G82,G83,G84 (G1081,G1082,G1083,G1084)

24.8.1 G1080 – Disabled Canned Cycles

(Shifted on G80 if USE_G80_CYCLES=TRUE)

Syntax
G1080

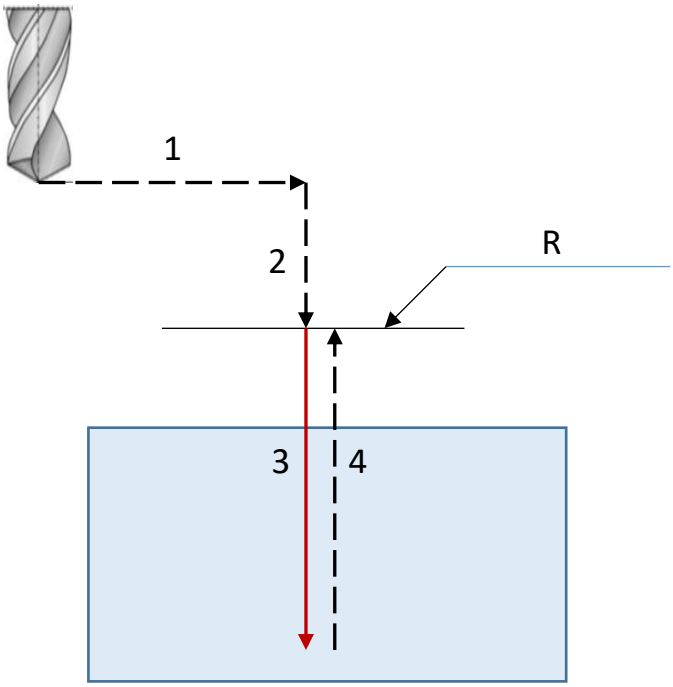
24.8.2 G1081 – Drilling Cycle

(Shifted on G81 if USE_G80_CYCLES=TRUE)

Syntax

G1081 Xval Yval Zval Qval Rval Kval Fval

X **X** position first drilling (optional) not inserted start for current point
Y **Y** position first drilling (optional) not inserted start for current point
Z Drilling Depth
Q Return Type (Optional) if **Q=1** return to **point 2** in **G1** with **F** programmed, if **Q<>1** or **not inserted** return to **point 2** in **G0**
R Disengagement
K Repetition (Optional) Active only if G91 and X or Y is inserted)
F Drilling Feed (Optional)



1) Move in G0 X,Y
2) Move in G0 ad R
3) Drilling in G1 a Z
4) Return in G0 ad R if Q<>1 or not inserted.
   Return in G1 if Q=1

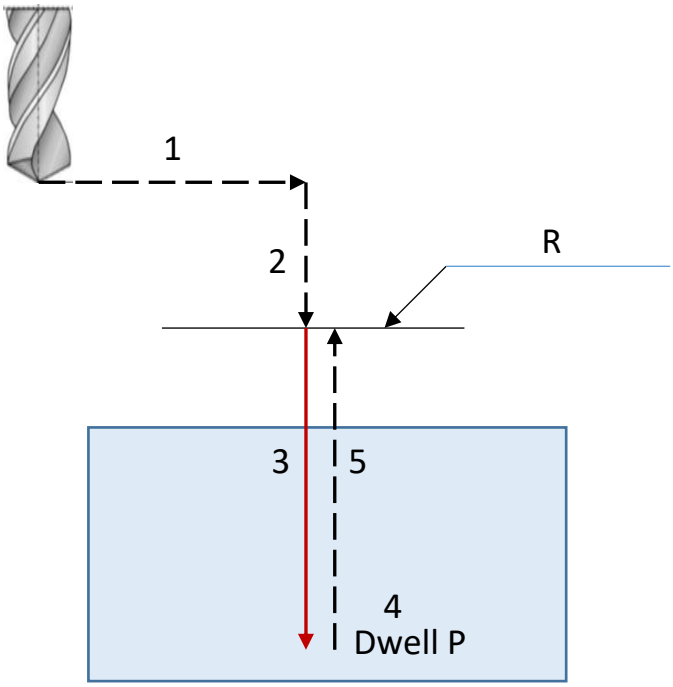
24.8.3 G1082 – Drilling Cycle

(Shifted on G82 if USE_G80_CYCLES=TRUE)

Syntax

G1082 Xval Yval Zval Qval Pval Rval Kval Fval

X **X** position first drilling (optional) not inserted start for current point
Y **Y** position first drilling (optional) not inserted start for current point
Z Drilling Depth
Q Return Type (Optional) if **Q=1** return to **point 2** in **G1** with **F** programmed, if **Q<>1** or **not inserted** return to **point 2** in **G0**
P Dwell Ms
R Disengagement
K Repetition (Optional) Active only if G91 and X or Y is inserted)
F Drilling Feed (Optional)



1) Move in G0 X,Y
2) Move in G0 ad R
3) Drilling in G1 a Z
4) Dwell P Ms
5) Return in G0 ad R if Q<>1 or not inserted.
   Return in G1 if Q=1

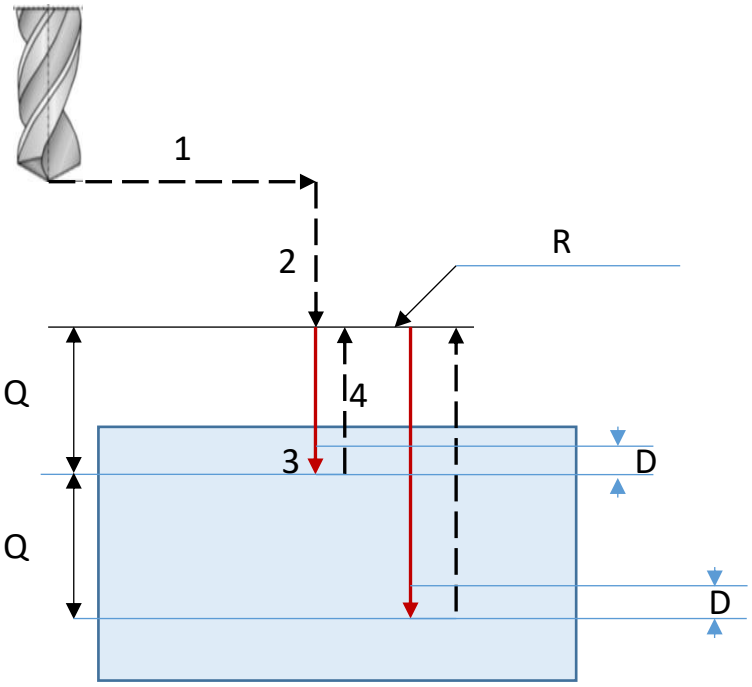
24.8.4 G1083 – Drilling Cycle

(Shifted on G83 if USE_G80_CYCLES=TRUE)

Syntax

G1083 Xval Yval Zval Rval Qval Kval Fval

X **X** position first drilling (optional) not inserted start for current point
Y **Y** position first drilling (optional) not inserted start for current point
Z Drilling Depth
P Peck Drilling distance (optional)
R Disengagement
Q Increase
K Repetition (Optional) Active only if G91 and X or Y is inserted)
F Drilling Feed (Optional)



1) Move in G0 X,Y
2) Move in G0 to R
3) Drilling in G1 to Q
4) Return G0 to R
5) Drilling in G0 to last + DELTA_G83 (D) Parameter
6) Repeat to end

If is insert the P parameter, in the point 4 return not in R position, but in P value position

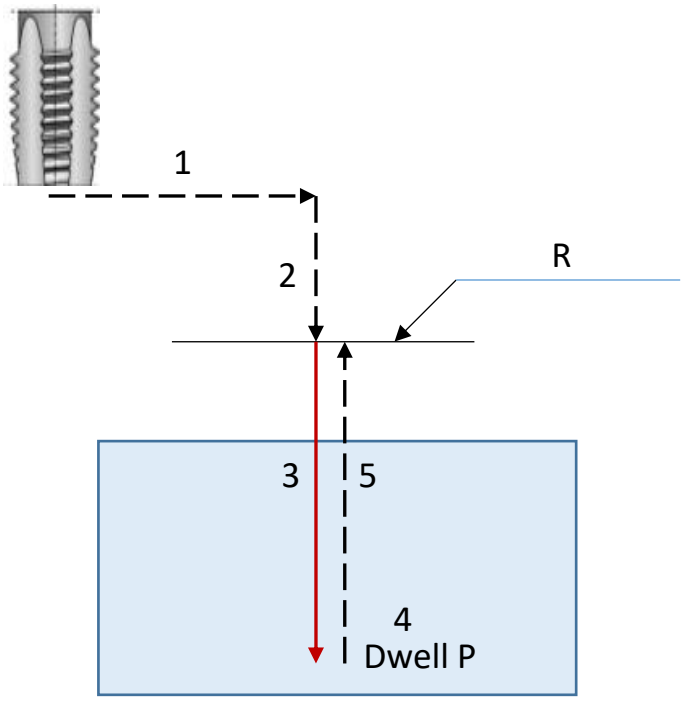
24.8.5 G1084 – Tapping Cycle

(Shifted on G84 if USE_G80_CYCLES=TRUE)

Syntax

G1084 Xval Yval Zval Pval Rval Kval Qval Fval Aval Bval

X **X** position first drilling (optional) not inserted start for current point
Y **Y** position first drilling (optional) not inserted start for current point
Z Tapping Depth
P Dwell Ms
R Disengagement
K Repetition (Optional) Active only if G91 and X or Y is inserted)
Q Tapping STEP (mm) only for INTERPOLATE MODE **(MODE_G84=TRUE)**
F Tapping Feed (Optional)
A Tapping CW or CCW (Optional) if **A**=1 CCW
B Return FEED Moltiplicator (Optional) from point 4 to point 5
Ex: B=2 the FEED (**F**) is multiplied for 2
Only value >0



1) Move in G0 X,Y
2) Move in G0 ad R
3) Tapping in G1 to Z interpolate with Axis A A (Set S and F ROT_G84 parameter defines CW or CCW rotation
4) Dwell P Ms
5) Return in G1 to R

24.9 SELECT WORK PLANE

Work plane defines:

- The plane for circular interpolation
- The plane where to calculate the axis stop (fast interpolation G60)
- The plane where to calculate the radius tool compensation

24.9.1 G17 – Select work plane on X-Y

Syntax
G17
Function type
MODAL
Cancel
G18 G19 G70

24.9.2 G18 - Select work plane on X-Z

Syntax
G18
Function type
MODAL
Cancel
G17 G19 G70

24.9.3 G18.1 - Select work plane on X-Z but not in PREVIEW

Syntax
G18.1
Function type
MODAL
Cancel
G17 G19 G70
Select the work plane on X-Z, but in PREVIEW remains X-Y

24.9.4 G19 - Select work plane on Y-Z

Syntax
G19
Function type
MODAL
Cancel
G17 G18 G70

24.9.5 G19.1 - Select work plane on Y-Z but not in PREVIEW

Syntax
G19.1
Function type
MODAL
Cancel
G17 G18 G70
Select the work plane on Y-Z, but in PREVIEW remains X-Y

24.9.6 G70 - Select work plane on axis by parameters

Syntax
G70 AXIS1 AXIS2
Function type
MODAL
Cancel
G17 G18 G19
Parametersi
Name of the couple af axis formed the new work plane
Ex:
G70XA // select work plane on axis X and A
G70QXQA // select work plane on Channel 0 Channel 3

24.10 START SEARCH OF HOME POSITION

With G71 function Isous allows to execute searching of home position directly in PartProgram.
It can be used, for example, when there are spindle axis with double function:

SPEED AXIS
CONTROLLED AXIS

In these cases when we switch function mode to a controlled axis it will be necessary to make searching of home position.

24.10.1 G71 – Start Homing cycle

Syntax
G71 AXIS
Function type
DIRECT
Parameter
Name of axis to be homed

Description
It starts searching of home position for selected axis. Searching mode is defined by machine parameter RZERO_MODE. PartProgram waits for ending of homing procedure.
However STOP PROGRAM is always active. By the parameter of configuration TIME_OUT_HOME can be generated an error (1005 Time Out axis Home) if searching isn't terminated in the set time.

Ex:
G71 X

24.10.2 G71.1 Enable Axis
Syntax
G71.1 AXIS
Function type
DIRECT
Parameter
Name of axis to be enabled

Description
Enable the Axis selected
Ex:
G71.1 X

24.10.3 G71.2 Disable Axis
Syntax
G71.2 AXIS
Function type
DIRECT
Parameter
Name of axis to be disabled

Description
Disable the Axis selected
Ex:
G71.2 X

24.11 AXIS MOVEMENT FUNCTIONS

24.11.1 G0 – Rapid positioning

Syntax
G0
Function type
MODAL
Cancel
G1 G2 G3

Description
G0 defines linear movement type with maximum axis speed.
Maximum speed of each axis is defined by machine parameters.
Isous use a specific calculation of axis speed reducing to minimum the axis STRESS during rapid positioning. Unlike other CN where G0 defines **NOT-INTERPOLATED** movement, Isous move axis always in **INTERPOLATION** mode, but calculating the best speed. In this mode we can obtain always a **FLUID MOTION** where all axis start and arrive simultaneously.
Furthermore it is used a specific **ACCELERATION** parameter different from the working one. It is recommended to use G0 only for movement with tool not in working.

Ex:
```
G0X0Y0Z0          // MOVE TO 0,0,0 POSITION
```

24.11.2 G0.1 – Rapid positioning with private Acceleration

Syntax
G0.1 AxisName Value
Function type
Immediate
Cancel
G1 G2 G3

Description
G0.1 defines linear movement type with maximum axis speed with private Acceleration. The Acceleration is got from **ACC_G0.1_AxisName** parameter. Only one axis at time can be insert in the **G0.1** function. The others features are the same of **G0**.

Ex:
```
G0.1 X0 // MOVE TO 0 POSITION
```

24.11.3 G0.2 – Rapid positioning for "Transported Axes"

Syntax
G0.2
Function type
MODAL
Cancel
G1 G2 G3

Description
G0.2 is like to G0, but the velocity vector is calculated only in the couple of axes od **WORK PLANE**, and the others axes are "**TRANSPORTED**". This allows to Tangential Axes don't use the **VELOCITY VECTOR**.

Ex:
G0X100**Y**100**A**150 // MOVE TO 100,100,150 POSITION WHERE A IS A TANGENTIAL AXIS

24.11.4 G1 – Linear interpolation a programmed F speed

Syntax
G1
Function type
MODAL
Cancel
G0 G2 G3

Description
G1 defines linear interpolation mode at programmed speed. If movement are on selected work plane and it's enabled **FAST interpolation (without stop on edge) G60**, Isous calculates automatically stop on edge. G1 enables movement at programmed F speed and can be modified by override potentiometer (if not disabled with G11). According to CN we can obtain different performance about number of segment processed in a time unit. Cn of NG35 series process about **350 segments per second,** instead **120** for CN of NGM13 series. That defines the maximum working velocity when the path is formed by **micro-segments**. All CN have a movement buffer, Isous tries to maintain it full (in G60 mode). If buffer empties CN stops axis obtaining a not fluid movement.

Ex:
G1X300Y200Z20

24.11.5 G1.1 – G1-G2-G3 Suspension and Set G0

Syntax
G1.1
Function type
Immediate
Cancel
G1 G2 G3

Description
G1.1 save the current Gx setted (G1 G2 G3) and force a G0 for the next movements

Ex:
G1.1X300Y200Z20

24.11.6 G1.2 – Resume Gx saved with G1.1

Syntax
G1.2
Function type
Immediate
Cancel
G0

Description
G1.2 Resume the G saved with G1.1

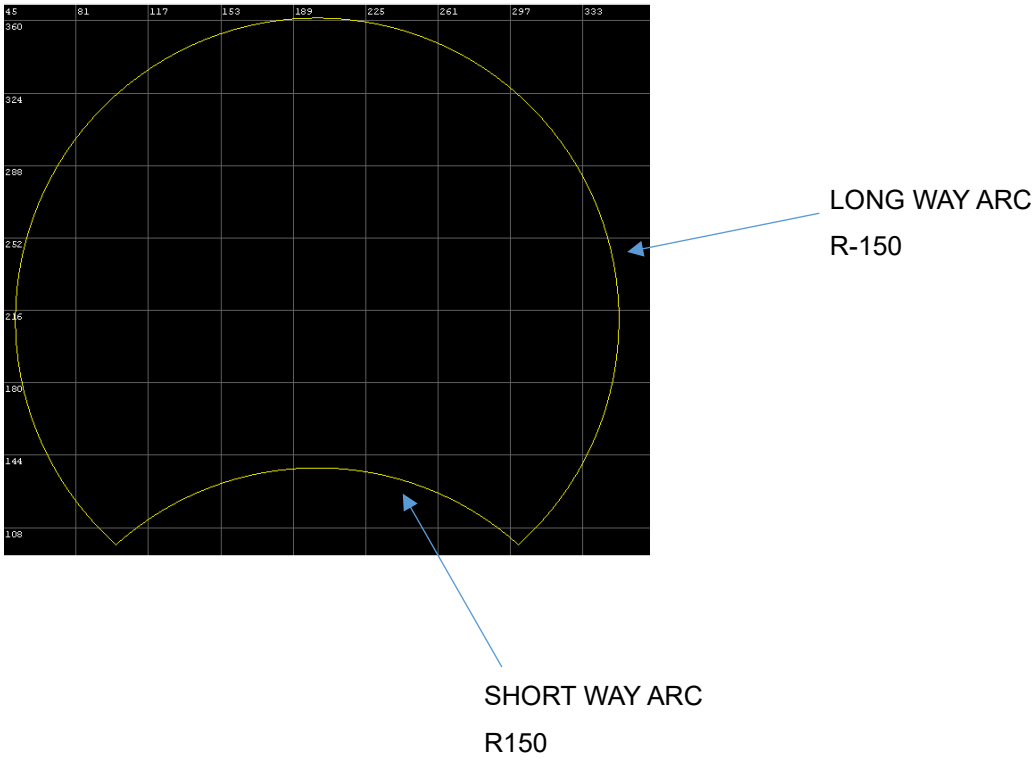
Ex:
G1.2X300Y200Z20

24.11.7 G2/G3 - Circular interpolation a programmed F speed

Syntax
G2 (cw)
G3 (ccw)
Function type
MODAL
Cancel
G0 G1

Description
G2 and G3 define circular interpolation at programmed speed. If movement are on selected work plane and it's enabled **FAST interpolation (without stop on edge) G60**, Isous calculates automatically stop on edge. G2 and G3 enable movement at programmed F speed and can be modified by override potentiometer (if not disabled with G11).
Circular interpolation can be executed by center parameters I and J, or by radius parameter R. With the second one it's not possible to generate a complete circle, for that it must be used I and J.

SHORT WAY and LONG WAY arc
Using radium parameter there are two arcs defined by the same parameters. The selection is via radius sign.
If RADIUS is positive a SHORT WAY ARC is generated.
If RADIUS is negative a LONG WAY ARC is generated.
See the example for details.



Ex:

```
G0X100Y100
G2X300Y100R-150        // EXECUTE LONG WAY ARC
G0X100Y100
G2X300Y100R150         // EXECUTE SHORT WAY ARC
```

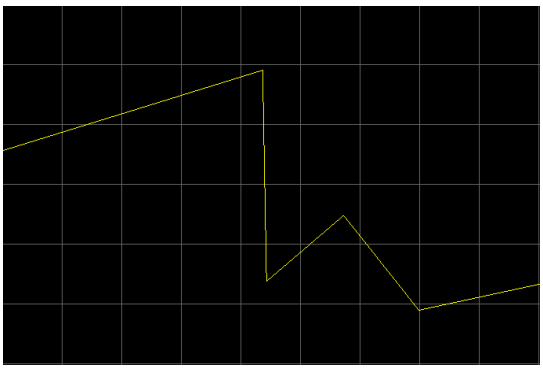
24.11.8 G30 - Enable automatic insert of fillet on edges

Syntax
G30 Rradius Aangle
Function type
MODAL
Cancel
G33
Parameters
R ***value of fillet radius (radius=0 disables the function)***
A ***Angle degree of insert threshold***

Description
G30 inserts automatically fillets on edges. The fillet radius is defined with parameter R, parameter A defines the angle threshold under which G30 function doesn't intervenes leaving path unchanged. This function is useful to soften movement in angular path.
Fillet is inserted only if it can be container inside two segment.
Setting a RADIUS=0 the function is disabled.

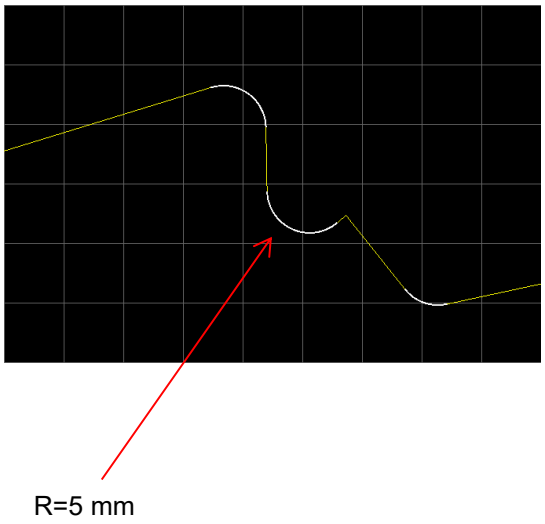
Ex:

```
// NORMAL PATH WITHOUT G30
G0 X75.45 Y157.46
G1 X107.84 Y167.43
G1 X108.3 Y142.73
G1 X117.36 Y150.44
G1 X126.2 Y139.34
G1 X140.92 Y142.51
```



Ex:

```
// SAME PATH USING G30
G0 X75.45 Y157.46
G30 R5 A20  // RADIUS=5 ANGLE=20 deg.
G1 X107.84 Y167.43
G1 X108.3 Y142.73
G1 X117.36 Y150.44
G1 X126.2 Y139.34
G1 X140.92 Y142.51
```



G31 – Suspend of automatic insert of fillet

Syntax
G31
Function type
MODAL
Cancel
G30

Description
Suspend G30. It can be restore with G32.

24.11.9 G32 - Restore of automatic insert of fillet

Syntax
G32
Function type
MODAL
Cancel
G31

Description
Restore G30 suspended by G31.

24.11.10 G33 - Enable automatic insert of bevel on edges

Syntax

G33 Rlen Aangle

Function type

MODAL

Cancel

G30

Parameter

R lenght of bevel (len=0 disables the function)

A Angle degree of insert threshold

Description

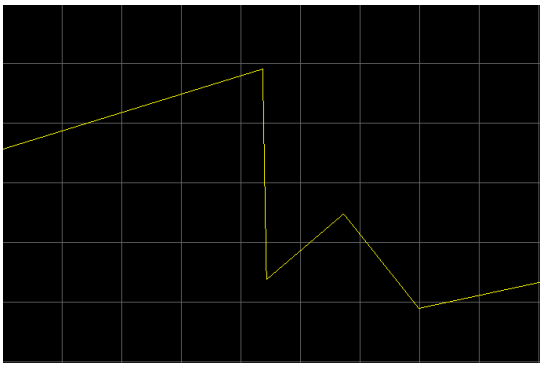
G33 inserts automatically bevels on edges. The bevel length is defined with parameter R, parameter A defines the angle threshold under which G33 function doesn't intervenes leaving path unchanged. This function is useful to soften movement in angular path.

Bevel is inserted only if it can be container inside two segment.

Setting a RADIUS=0 the function is disabled.

Ex:

```
// NORMAL PATH WITHOUT G33
G0 X75.45 Y157.46
G1 X107.84 Y167.43
G1 X108.3 Y142.73
G1 X117.36 Y150.44
G1 X126.2 Y139.34
G1 X140.92 Y142.51
```



Ex:

```
// SAME PATH USING G33
G0 X75.45 Y157.46
G33 R5 A20  // LEN=5MM ANGLE=20 deg.
G1 X107.84 Y167.43
G1 X108.3 Y142.73
G1 X117.36 Y150.44
G1 X126.2 Y139.34
G1 X140.92 Y142.51
```

24.11.11 G34 - Suspend of automatic insert of bevel

Syntax

G34

Function type

MODAL

Cancel

G33

Description

Suspend G33. It can be restore with G35.

24.11.12 G35- Restore of automatic insert of bevel

Syntax

G35

Function type

MODAL

Cancel

G34

Description

Restore G33 suspended by G34.

24.11.13 G102 – Start searching sensor position

Syntax

G102

G102.2

Function type

DIRECT

Description

It start acquisition from sensor.

Axis move to positions selected by X,Y,Z etc. The values refer to an absolute or relative position according with G90 G91. Selected **WORK ORIGIN** and **WORK OFFSET** are considered too. More axis can be move simultaneously (interpolated) and the speed is set with the machine parameter **ACQ_VE**L.

Also a acquisition mode can be selected by machine parameter **ACQ_MODE**.

Executing can be stopped by activation of **SENSOR** or by reaching **TARGET** position. In the first case PartProgram continues to execute from next block to G102.

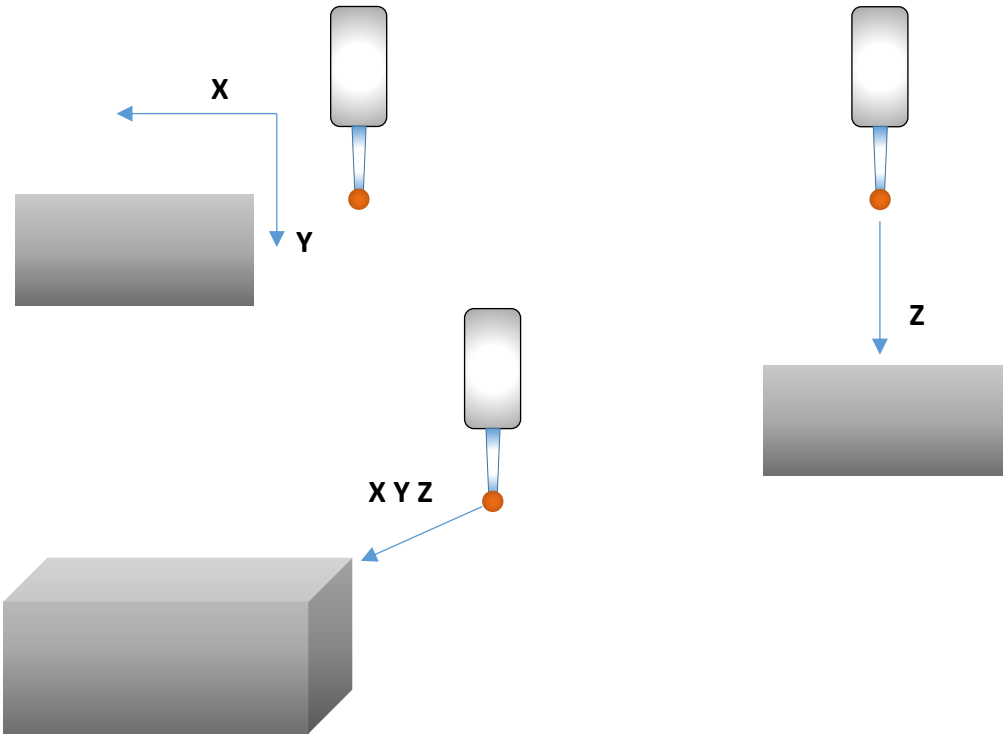
In case of reaching of **TARGET** position PartProgram ends with **ACQ. SENSOR ERROR**.

G102 waits for axis stop before to proceed with acquisition cycle.

G102.2 In case of reaching of **TARGET** position doesn't generate any ALARM

Ex:

```
G90
G102 X100          // ACQUISITION MOVING ONLY AXIS X UP TO POSITION 100
G92 X              // SET WORK ORIGIN IN ACQUIRED POINT
```



24.11.14 G102.1 – Acquisition from PxVision System

Syntax

G102.1 XYZABCUVWFS (parameters)

Function type

DIRECT

Description

IsoUs can acquired data from **PxVision** system. For use **G102.1** must be installed the PlugIn:

UsPxVisionAcq

With this function is possible the following operations:

- **Work Origins**
- **Tool Length and Diameter measure**
- **Objects Measures**
- **Objects Detection**

G102.1 function uses the **X** parameter for define the acquisition type (see the PxVision document for more informations)

X PARAMETER ***Defines the acquisition Type***

Marker Detection **Detect a Marker and get Center**

- **X=0** **Circular Marker Detection**
- **X=1** **Square Marker Detection**
- **X=2** **Type1 Marker Detection**
- **X=3** **Hole Marker Detection**

Probe Acquisition **Virtual probe and get data acquisition**

- **X=100** **Probe Single Point**
- **X=101** **Probe Work Origins**
- **X=102** **Probe Tool Measure**

Detector **Objects detection**

- **X=200** **Gray Detector**
- **X=201** **Contrast Detector**
- **X=202** **Brightness Detector**
- **X=203** **Color Detector**
- **X=204** **Area Detector**

Measures **Generic Objects measures**

- **X=300** **Cross Section**
- **X=301** **Calliper**
- **X=302** **Fit Line**
- **X=303** **Fit Circle**
- **X=304** **Gap Cross Section**

Y PARAMETER ***Defines the return data referred to axes values***

- **Y=0** **The values are in RELATIVE all values include Work Origins, Offset etc. set by IsuUs**
- **Y=1** **The values are in ABSOLUTE all values are refer to machine origins**

The data returned by **G102.1** are put in the IsoUs Variables by ADDRESS **":"**.
The start Address is configured by **PlugIn UsPxVisionAcq** :

START_ADDRESS	**DATA_INPUT**	Data Inputs for single function (Reserved)
START_ADDRESS+100	**DATA_OUTPUT**	Data returned from function

Eg: START_ADDRESS= **10000** **DATA_OUTPUT** =**10100.**
Is recommended use Address from 10000 to 15000

G102.1 X0-1-2-3 MARKER DETCTION

G102.1 X0 e G102.1 X2 (Circular, Type1)

Add Parameters - None

DATA_OUTPUT	**1** Acq Ok **0** Acq Error
DATA_OUTPUT +1	**X** Center Marker (mm) referred to Camera Center
DATA_OUTPUT +2	**Y** Center Marker (mm) referred to Camera Center
DATA_OUTPUT +3	**Radius** Marker (mm)
DATA_OUTPUT +4	**Angle** Marker (drg)

G102.1 X1 (Square)

Add Parameters - None

DATA_OUTPUT	**1** Acq Ok **0** Acq Error
DATA_OUTPUT +1	**X** Center Marker (mm) referred to Camera Center
DATA_OUTPUT +2	**Y** Center Marker (mm) referred to Camera Center
DATA_OUTPUT +3	**Width** Marker (mm)
DATA_OUTPUT +4	**Height** Marker (mm)
DATA_OUTPUT +5	**Angle** Marker (drg)

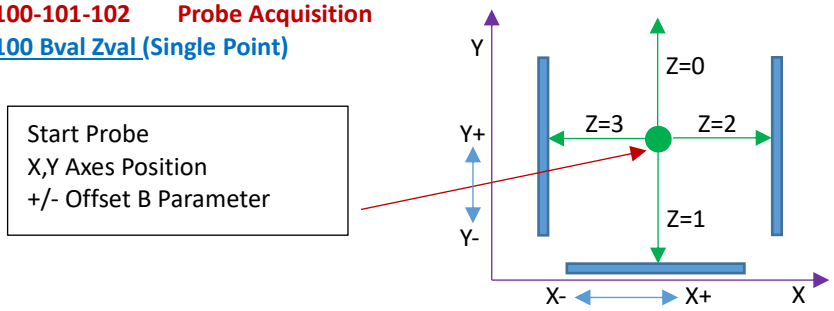
G102.1 X3 (Hole)

Add Parameters - None

DATA_OUTPUT	**1** Acq Ok **0** Acq Error
DATA_OUTPUT +1	**X** Center Marker (mm) referred to Camera Center
DATA_OUTPUT +2	**Y** Center Marker (mm) referred to Camera Center
DATA_OUTPUT +3	**Radius** Marker (mm)

G102.1 X100-101-102 Probe Acquisition

G102.1 X100 Bval Zval (Single Point)



Add Parameters

B **Optional Offset in mm +/- in the acquisition direction set by Z of Probe (default 10mm)**

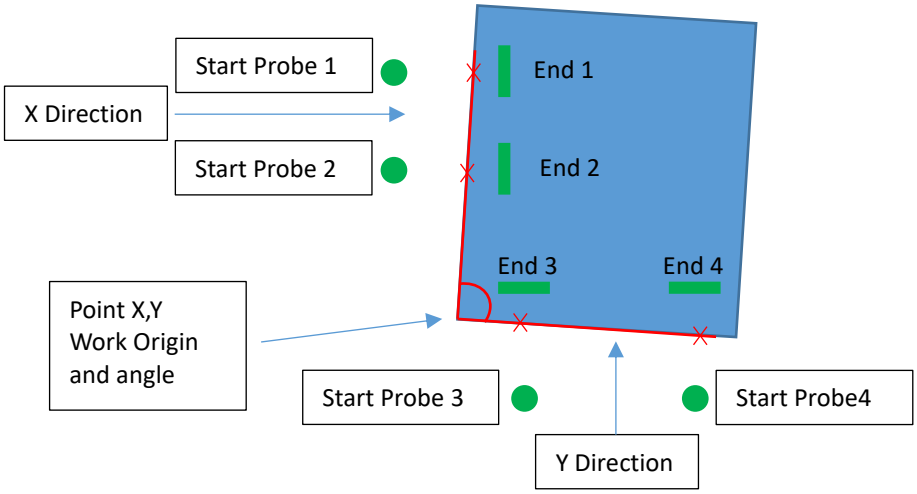
Z **Mandatory Probe direction**

Z=0 ***Y POSITIVE***
Z=1 ***Y NEGATIVE***
Z=2 ***X POSITIVE***
Z=3 ***X NEGATIVE***

DATA_OUTPUT **1** Acq Ok **0** Acq Error
DATA_OUTPUT +1 **X** Probe (mm) **referred to Machine Origins**
DATA_OUTPUT +2 **Y** Probe (mm) **referred to Machine Origins**

G102.1 X101 Vval(Work Origins)

For thie function, is necessary configure one or more **JOBS from PxVisionBrowser** with **4 ProbeSinglePoints** that define **2 points Start, End for X direction and 2 points Start, End for Y direction**



Add Parameters

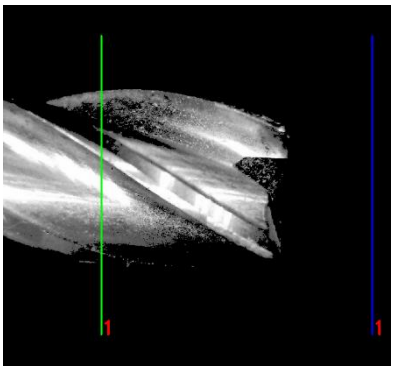
V **Mandatori JOB index (Inserit in PlugIn UsPxVisionAcq) for use**

DATA_OUTPUT **1** Acq Ok **0** Acq Error
DATA_OUTPUT +1 **X** Origin
DATA_OUTPUT +2 **Y** Origin
DATA_OUTPUT +3 **Angle**

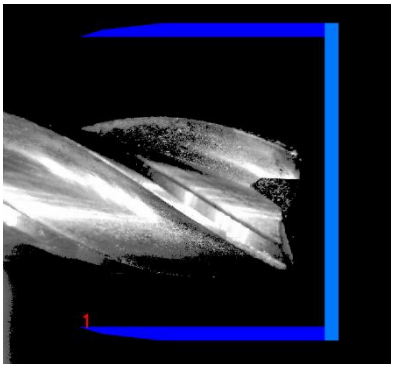
G102.1 X102 Vval (Tool Measure)

Diameter and Tool Length measure
For use this function, is necessary to configure one or more **JOBS** by **PxVisionBrowser** application with parameters for Length Measure in **ProbeLine** and Diameter measure in **Calliper**
Configure the **ProbeLine** and **Calliper** for contain the max tool dimensions

ProbeLine



Calliper



Add Parameters

V **Mandatori JOB index (Inserit in PlugIn UsPxVisionAcq) for use**

DATA_OUTPUT	**1** Acq Ok	**0** Acq Error
DATA_OUTPUT +1	**Length** Tool (mm)	
DATA_OUTPUT +2	**Diameter** Tool (mm)	

G102.1 X200-201-202-203-204 Detector

For use this function, is necessary to configure the Detectors by **PxVisionBrowser** application

G102.1 X200 G102.1 X201 G102.1 X202 X203 X204 (Gray,Contrast,Brightness,Color,Area)

Add Parameters - None

DATA_OUTPUT	**1** Acq Ok	**0** Acq Error (detector 1)
DATA_OUTPUT +1	**1** Acq Ok	**0** Acq Error (detector 2)
DATA_OUTPUT +2	**1** Acq Ok	**0** Acq Error (detector 3)

..
Etc.

G102.1 X300-301-302-303-304 Measures

For use this function, is necessary to configure the Measures by **PxVisionBrowser** application

G102.1 X300 G102.1 X301 (Cross Section,Calliper)

Add Parameters - None

DATA_OUTPUT	**1** Acq Ok	**0** Acq Error (Config 1)
DATA_OUTPUT +1	**Measure** (mm) (Config 1)	
DATA_OUTPUT +2	**1** Acq Ok	**0** Acq Error (Config 2)
DATA_OUTPUT +3	**Measure** (mm) (Config 2)	
DATA_OUTPUT +4	**1** Acq Ok	**0** Acq Error (Config 3)
DATA_OUTPUT +5	**Measure** (mm) (Config 3)	

..
Etc..

G102.1 X302 (Fit Line)
Add Parameters - None

DATA_OUTPUT	**1** Acq Ok **0** Acq Error (Config 1)
DATA_OUTPUT +1	**Len** Line (mm) (Config 1)
DATA_OUTPUT +2	**Angle** Line (drg) (Config 1)
DATA_OUTPUT +3	**1** Acq Ok **0** Acq Error (Config 2)
DATA_OUTPUT +4	**Len Line** (mm) (Config 2)
DATA_OUTPUT +5	**Angle** Line (drg) (Config 2)
DATA_OUTPUT +6	**1** Acq Ok **0** Acq Error (Config 3)
DATA_OUTPUT +7	**Len Line** (mm) (Config 3)
DATA_OUTPUT +8	**Angle** Line (drg) (Config 3)

..
Etc.

G102.1 X303 (Fit Circle)
Add Parameters - None

DATA_OUTPUT	**1** Acq Ok **0** Acq Error (Config 1)
DATA_OUTPUT +1	**Center X** (mm) referred to Camera Center (Config 1)
DATA_OUTPUT +2	**Center Y** (mm) referred to Camera Center (Config 1)
DATA_OUTPUT +3	**Radius** (mm) (Config 1)
DATA_OUTPUT +4	**1** Acq Ok **0** Acq Error (Config 2)
DATA_OUTPUT +5	**Center X** (mm) referred to Camera Center (Config 2)
DATA_OUTPUT +6	**Center Y** (mm) referred to Camera Center (Config 2)
DATA_OUTPUT +7	**Radius** (mm) (Config 2)

..
Etc.

G102.1 X304 (Gap Cross Section)
Add Parameters - None

DATA_OUTPUT	**1** Acq Ok **0** Acq Error (Config 1)
DATA_OUTPUT +1	**Max Val** (mm) (Config 1)
DATA_OUTPUT +2	**Min Val** (mm) (Config 1)
DATA_OUTPUT +3	**1** Acq Ok **0** Acq Error (Config 2)
DATA_OUTPUT +4	**Min Val** (mm) (Config 2)
DATA_OUTPUT +5	**Min Val** (mm) (Config 2)

Etc.

BarCode Reader

Can be read **BarCode** and **QrCode**, before to use this function, must create a list by PlugIn **UsPxVisionAcq**, for associate the BarCode or QrCode to an **INDEX**

G102.1 X400

Add Parameters - None

DATA_OUTPUT	**1** Acq Ok	**0** Acq Error (Config 1)
DATA_OUTPUT +1	**Index in the list (-1) not found**	

G102.1 X401 **Ocr Reader**

Can be read a text string without Cr/Lf.

As the BarCode, also the Text String must coded by PlugIn **UsPxVisionAcq** for associate the OcrCode to an **INDEX**

G102.1 X401

Add Parameters - None

DATA_OUTPUT	**1** Acq Ok	**0** Acq Error (Config 1)
DATA_OUTPUT +1	**Index in the list (-1) not found**	

24.12 PROGRAMMING AXIS INTERPOLATION SPEED

Interpolation speed is a very important parameter to obtain a fine working. Speed value is expressed in Mt/min with a resolution dependent by Isous configuration. Selected interpolation speed can be changed in automatic way (arc auto-reduction), or manually with parameter **F** or override potentiometer. F parameter acts only when invoked in PartProgram and however at begin of next segment, OVERRIDE potentiometer instead acts in any time allowing immediately change of speed.

24.12.1 F – Speed of axis interpolation

Syntax

Fval

Function type

MODAL

Parameter

Val ***Value in Mt/min of speed***

Description

F function set vectorial speed of the axis. Such speed will be separated by interpolator according with the programmed trajectory.

G0 movement are not affected by parameter F.

Ex:

```
G0 X75.45 Y157.46          // RAPID MOVEMENT AT F MAX
F10.3                      // SET AXIS SPEED AT 10.3 MT/MIN
G1 X107.84 Y167.43
```

24.12.2 ARC speed auto-reduction

Function type

Configuration programming

Parameter

See Isous configuration

Description

Isous uses a special algorithm which allows to reduce automatically axis speed during circular interpolation. It simplifies F programming in path formed by arcs without to insert an appropriate speed for each arc (lower speed for lower radius – bigger speed for bigger radius), Isous does it for you.

The calculation is base on the centrifugal acceleration and it must be manually searched experimentally .

24.13 Work Plane Transformation

Here they are explained all function relative to manage with work plane such as: rotation and mirror.

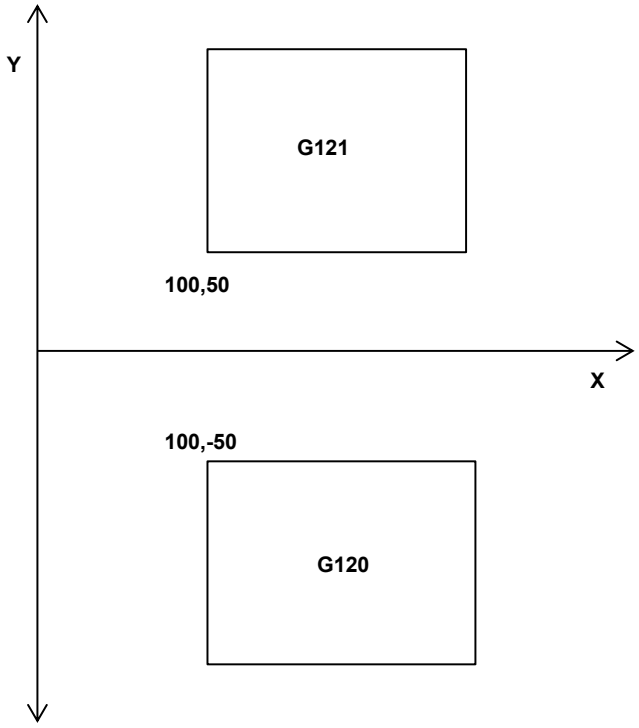
24.13.1 G120 – Vertical mirror

Syntax
G120
Function type
MODAL
Cancel
G121

Description
It enables vertical mirror on selected work plane. It acts inverting automatically the sign of axis Y.

Ex:

```
G121 // DISABLE G120
// ***** EXECUTE FIRST SQUARE
G0 X100 Y100
G1 X150
Y50
X100
Y100
// ***** END FIRST SQUARE
G120 // ENABLE VERTICAL MIRROR
// ***** EXECUTE SECOND SQUARE
G0 X100 Y100
G1 X150
Y50
X100
Y100
// ***** END SECOND SQUARE
```



24.13.2 G121 – Disable vertical mirror

Syntax
G121
Function type
MODAL
Cancel
G120

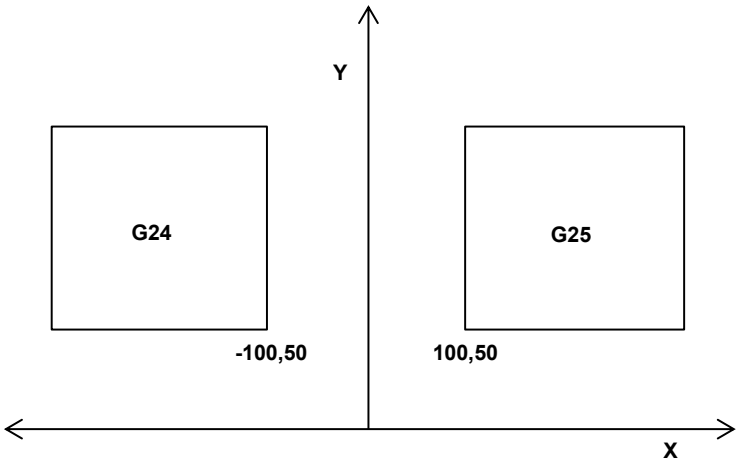
Description
Disable vertical mirror on work plane.

24.13.3 G24 – Horizontal mirror

Syntax
G24
Function type
MODAL
Cancel
G25

Description
It enables horizontal mirror on selected work plane. It acts inverting automatically the sign of axis X.

Es:
```
G25 // DISABLE G24
// ***** EXECUTE FIRST SQUARE
G0 X100 Y100
G1 X150
Y50
X100
Y100
// ***** END FIRST SQUARE
G24// ENABLE VERTICAL MIRROR
// ***** EXECUTE SECOND SQUARE
G0 X100 Y100
G1 X150
Y50
X100
Y100
// ***** END SECOND SQUARE
```



24.13.4 G25 – Disable horizontal mirror

Syntax
G25
Function type
MODAL
Cancel
G24

Description
Disable horizontal mirror on work plane.

24.13.5 G22 – Echange axis of work plane

Syntax
G22
Function type
MODAL
Cancel
G23

Description
Axis of work plane are exchanged between them (ex. X with Y).

24.13.6 G23 – Restore axis of work plane

Syntax
G23
Function type
MODAL
Cancel
G22

Description
Restore the original axis of work plane.

24.13.7 G26 – Exchange axis by parameter

Syntax
G26 Axis1 Axis2

Function type
MODAL
Cancel
G27
Description
A couple of selected axis by parameters are exchanged between them.
Ex:
G26 YZ // EXCHANGE Y WITH Z
G26 QYQZ // EXCHANGE CHANNEL 1 WITH CHANNEL 2

24.13.8 G27 – Suspend G26

Syntax
G27
Function type
MODAL
Cancel
G26

Description
It suspends the axis exchanging set with G26

24.13.9 G28 – Restore G26
(Shifted on G1028 if USE_G80_CYCLES=TRUE)

Syntax
G28
Function type
MODAL
Cancel
G27

Description
Restore axis exchanging set withG26

24.13.10 G1028 – Return to Home

(Shifted on G28 if USE_G80_CYCLES=TRUE)

Syntax

G1028 Xval Yval Zval….

Description

Return to **HOME** , the Axes values are indicates in the parameters **G28HOME_X**, **G28HOME_Y** etc.
Xval,Yval,Zval etc. Optional Parameters
If they are inserted before to go to HOME position the Axes are moved to Xval,Yval,Zval position
The FEED is get to previous F set or the G0 function

24.13.11 G51 – Suspend work plane rotation

Syntax
G51
Function type
MODAL
Cancel
G50 G52

Description
Suspend the rotation of work plane.

24.13.12 G52 – Restore work plane rotation

Syntax
G52
Function type
MODAL
Cancel
G51

Description
Restore last work plane rotation set with G50.

24.13.13 G50 - Work plane rotation

Syntax

G50 Xcenterx Ycentery Zangle (DX,DY Relative value)

Function type

Disabled when PartProgram start

Parameters

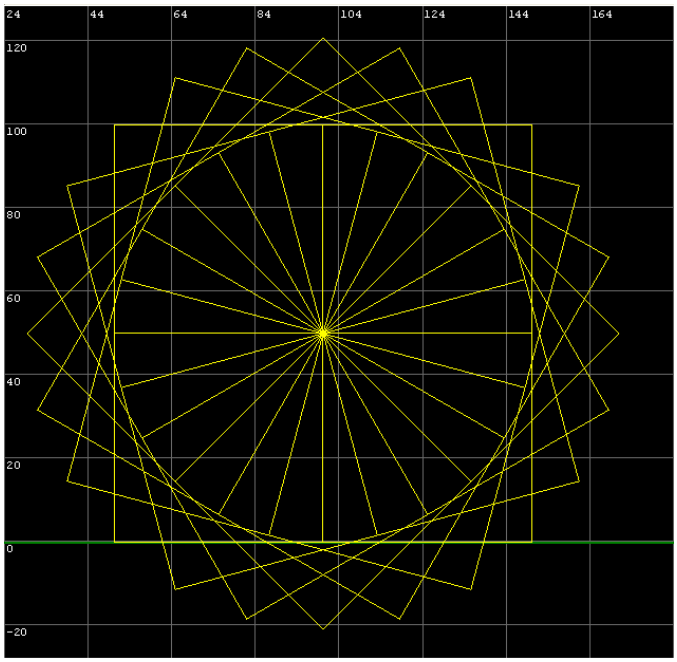
X	***Rotation center axis X***
Y	***Rotation center axis Y***
Z	***Rotation angle in degree***

Description

It execute a rotation of work plane. All following position will be modified with this transformation.
At each start of PartProgram rotation center is disabled, or if Angle is 0.

Ex:

```
// ROTATION OF A SQUARE
$ROT=0
LOOP 24
G0 X100 Y100
G1 X150
Y50
X100
Y100
$ROT=$ROT+15// INCR. ANGLE
G50 X100 Y50 Z[$ROT]// ROTATION
END_LOOP
```



24.13.14 G1050 – Disable Scaling

Syntax
G1050
Function type
MODAL
Cancel
G1051

Description
G1050 function, disable SCALING G1051

24.13.15 G1051 – Enable Scaling

Syntax
G1051 Xcentrox Ycentroy Zcentroz IratioX Jratioy Kratioz Pratioxyz
Function type
MODAL
Cancel
G1050
Parameters

X	***X Scaling Center – se non inserito viene presa la posizione attuale dell' asse X***
Y	***Y Scaling Center – se non inserito viene presa la posizione attuale dell' asse Y***
Z	***Z Scaling Center – se non inserito viene presa la posizione attuale dell' asse Z***
I	***X Ratio – X<1 reduces X>1 increases - if it is not inserted, uses the previous value***
J	***Y Ratio – Y<1 reduces Y>1 increases - if it is not inserted, uses the previous value***
K	***Z Ratio – Z<1 reduces Z>1 increases - if it is not inserted, uses the previous value***
P	***XYZ Ratio – P<1 reduces P>1 increases – Uses the same Ratio for X,Y,Z***

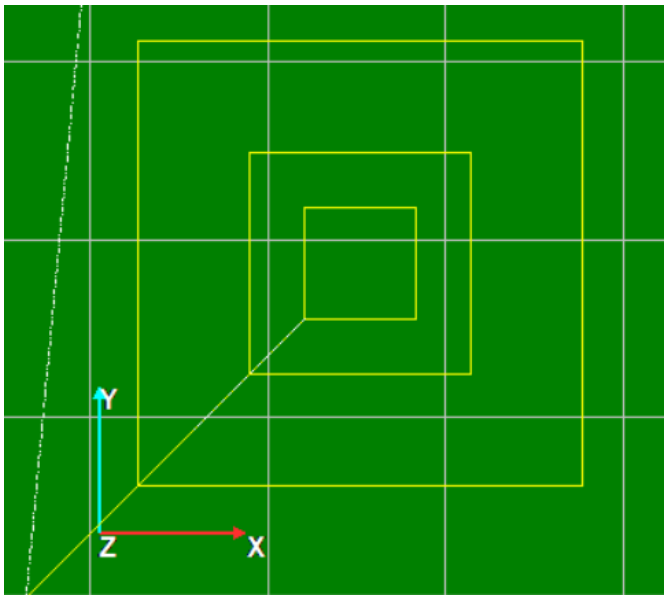
IF NO PARAMETERS ARE INSERTED (Only G1051) the previous values are used

Description
G1051 allows to scaling the Gcode with X,Y,Z Coordinate center.

Ex:

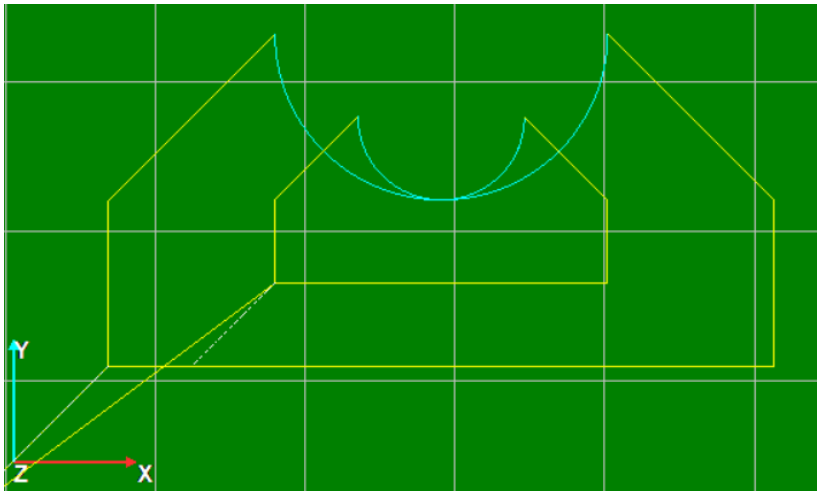
```
//ORIGINAL
G0X0Y0
G1X10Y10
Y20
X20
Y10
X10
G0X0Y0
//SCALING
G1051 X15Y15I0.5J0.5
G1X10Y10
Y20
X20
Y10
X10
G0X0Y0
//SCALING
G1051 X15Y15I2J2
G1X10Y10
Y20
X20
Y10
```

X10



Ex:

```
//ORIGINAL
G0X0Y0
G1X10Y10
Y20
X20Y30
G3X40Y30R10
G1X50Y20
X50Y10
X10Y10
G0X0Y0
//SCALING
G1051 X30Y20I0.5J0.5
G1X10Y10
Y20
X20Y30
G3X40Y30R10
G1X50Y20
X50Y10
X10Y10
G0X0Y0
```



24.13.16 G103 – Set RTCP parameters

Syntax
G103 Xval Yval Zval ….. G103 QXval QYval QZval
Function type
MODAL

Parameters
X ***Value of parameter X***
Y ***Value of parameter Y***
Z ***Value of parameter Z***
……..

Description
Isous uses a **RTCP** (rotate tool center point) adaptable to any geometry of the machine. Therefore the value passed to function by G103 (max 9 value X,Y,Z,A,B,C,U,V,W) will determine the right parameters for a specific geometry. The calculation of RTPC must be set in installation phase and it will be necessary the intervention of a specialized technician. **G103** passes the parameters to CN without enabling RTCP function. Enable/disable is demanded to **G104**-**G105** function.

24.13.17 G104 – Enable/restore RTCP

Syntax
G104
Function type
MODAL
Cancel
G105
Description
G104 enable or restore **RTCP** function on CN.

24.13.18 G104.1 – Enable/restore RTCP

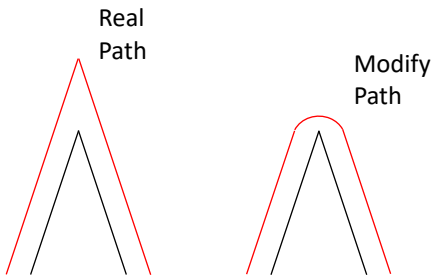
Syntax
G104.1
Function type
MODAL
Cancel
G105
Description
G104 enable or restore **RTCP** function on CN moving the X,Y,Z based the rotative Axes position

24.13.19 G105 – Disable RTCP

Syntax
G105
Function type
MODAL
Cancel
G104
Description
G105 disable **RTCP** function on CN restoring normal interpolation. RTCP function can be restored with **G104.**

25 TOOL RADIUS COMPENSATION

ISOUS manages the tool offset compensation, that can be activated with G41 and G42 ISO function. This functions allow to adjust the tool track, according the tool diameter. Using this mode, it's possible to have some interference on tool track, if we have high diameter tools.
ISOUS can display on its interface, a graphic preview, where will be showed interfering points, due to make possible to adjust the track, previous the work start.
However the tool compensation will take high attention on the track elaboration.
Furthermore it will insert automatically connection arcs (with tool radius as arc radius) , if the intersection point of the compensation tracks, is very far away from real intersection point.



25.1.1 G41/G42 – Enable left/right radius tool compensation

Syntax
G41
Function type
MODAL
Cancel
G40

Description
G41 Enable left compensation.
G42 Enable right compensation.
Tool diameter is set by **D** function or by tool table **Tn**.

25.1.2 G40 – Disable radius tool compensation

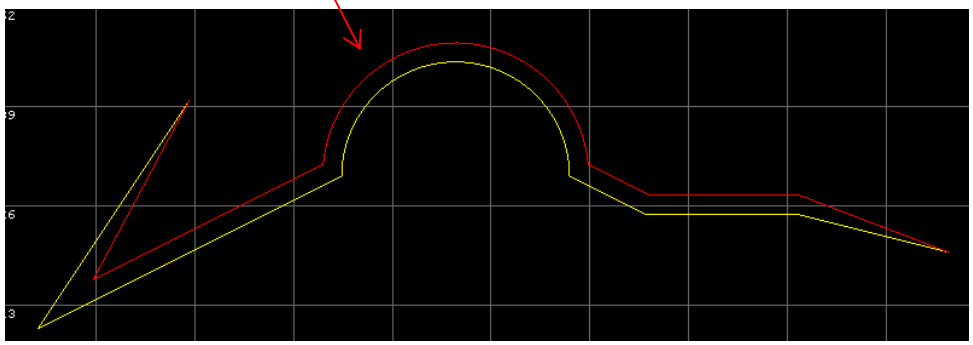
Syntax
G40
Function type
MODAL
Cancel
G41 - G42

Description
G40 disables the radium tool compesation (both left and right).

PATH WITH G41

Ex:

```
D5// DIAMETER=5mm
G41// LEFT COMP.
G0X30Y40
G1X10Y10
G1X50Y30
G2X80Y30R15
G1X90Y25
G1X110
G1X130Y20
G40
```

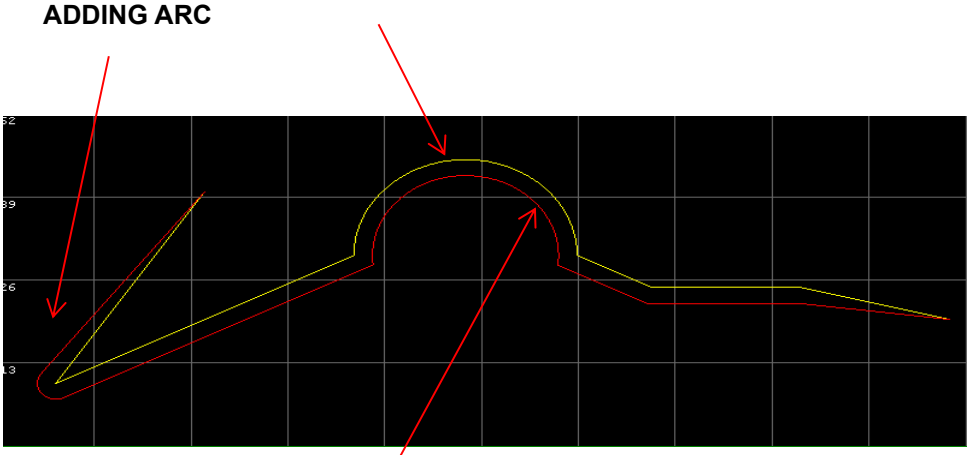


PATH WITH G40

ADDING ARC

Ex:

```
D5// DIAMETER=5mm
G42// RIGTH COMP.
G0X30Y40
G1X10Y10
G1X50Y30
G2X80Y30R15
G1X90Y25
G1X110
G1X130Y20
G40
```



PATH WITH G42

25.1.3 D – Tool diameter

Syntax
Dvalue

Function type
MODAL
Cancel
Tn ***(optional correction set by TOOL TABLE)***

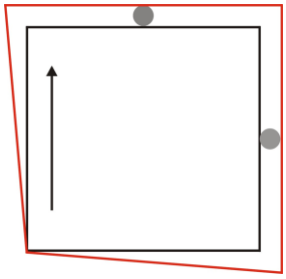
Parameter
Value ***Tool diameter in mm***

Description
D allows to set the value in mm of tool diameter to be used with G41 G42.
Tool diameter D must be set before G41-G42 and it must not changed until the compensation remains active.

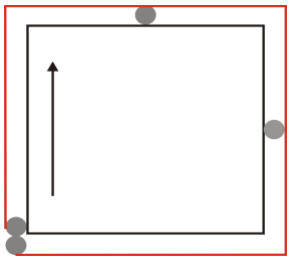
25.1.4 G47 – Set tool entry mode

Syntax
G47 Xval
Function type
MODAL

Description
G47 allow to choose two modes of tool entry. The default mode is **G47 X0**: compensation starts on next segment. With **G47 X1** compensation starts immediately and it is useful with closet shape.



Ex. Working with **G47 X0** (default)
Ex. Working with **G47 X1**



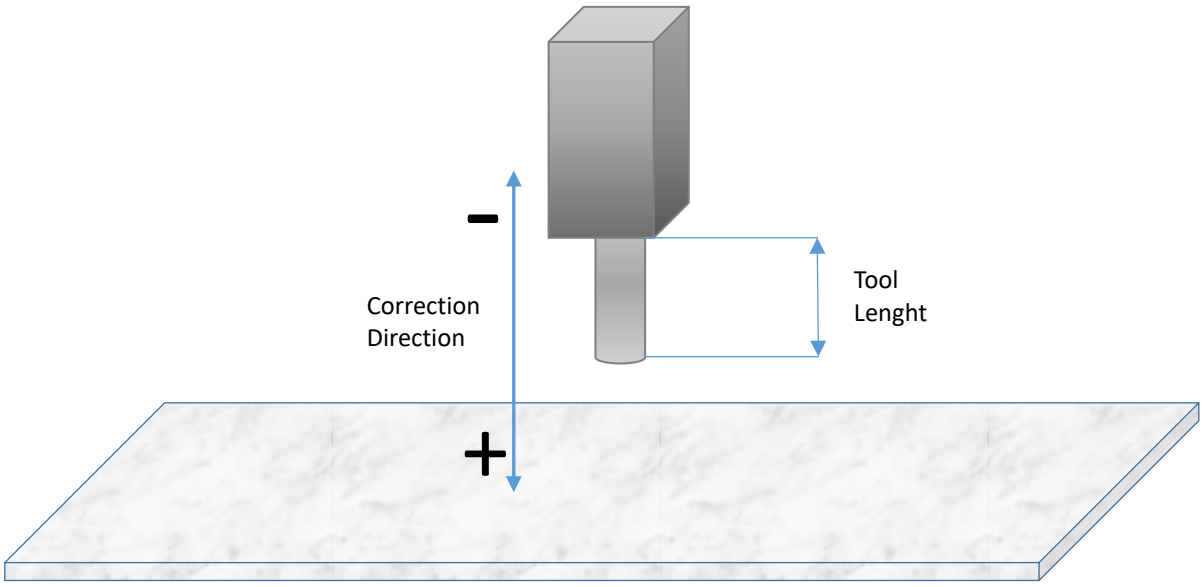
NOTE:
Using G47X1 tool must be positioned in start point byl PartProgram (so already shifted from shape of tool radium). The end of compensation instead must follow this scheme:
G0 Z (UP) // raise Z
***G40 // disable* compensation**
G1 Z (UP) // raise Z again

Example for compensation of a square:

```
G47 X1
F 2.5
G0 X 26.2332 Y 26.1708   // move tool on start point
F 5
G1 Z 0 // move Z to work position
D 3      // set tool diameter
G42  // enable compensation G42
G1 X 24.7332 Y 26.1708
G1 X 24.7332 Y 76.1708
G1 X 74.7332 Y 76.1708
G1 X 74.7332 Y 26.1708
G1 X 24.7332 Y 26.1708
G0 Z 20 // raise Z
G40 // disable compensation
G1 Z 20 // raise Z again
```

25.2 TOOL LENGHT COMPENSATION

It' possible to compensate tool length on any axis. It is useful when we want to work the same PartProgram with tools of different length. Tool length compensation acts only in the direction selected by G43 function.
The axis affected by compensation is selectable as well as length of correction. Length correction compensates movement on the selected axis. The effective tool length must be the measure of projection tools from spindle.



25.2.1 G43 – Enable tool length compensation from parameter

Syntax
G43 Xlen Kmode AXISdir(+/-) (DX Relative value)

Function type
MODAL
Cancel
G45 ***(optional correction set by Tool Table)***
Parameters
len ***tool length in mm es: 12.35***
If LEN=0 tool length is taken by tool table Tn
Mode ***Optional K parameter, this can have the following values:***
K0 ***Normal Mode how if K is omitte***
K1 ***With this parameter, the tool length is used how additional OFFSET , therefore the axis value (ABSOLUTE,RELATIVE) is immediately updated In this mode the "dir" parameter is not considered, the direction is defined by "LEN" sign (+ or -)***
K2 ***As K1 but the sign of LEN TOOLS from table is inverted***
AXIS ***Axis on which apply length compensation***
dir ***Direction of compensation***
+ positive
- negative

Description
G43 enables tool length compensation. Corrections are applied only in the selected axis.

Ex:
G43 X10.3 Z+ // ENABLE POSITIVE COMPENSATION ON AXIS Z OF LEN=10.3mm

25.2.2 G44 – Disable tool length compensation G43

Syntax
G44

Function type
MODAL
Cancel
G43

Description
G44 disable tool length compensation set with G43

25.2.3 G44.1 (2) – Suspend/Resume G43

Syntax
G44.1 // Suspend G43
G44.2 // Resume G43 with old parameters

Function type
MODAL
Cancel
G43
Parameters

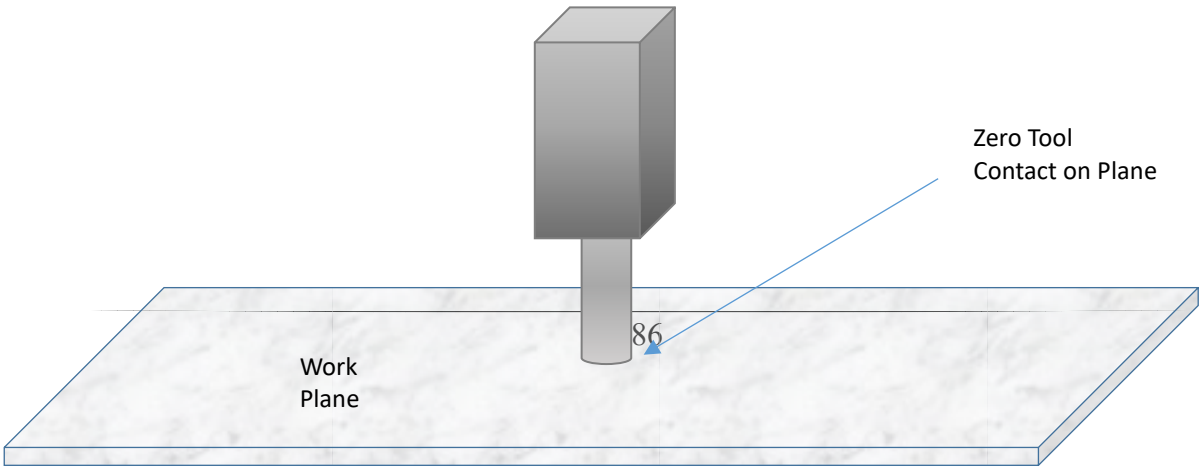
25.2.4 G45 – Enable tool length compensation from tool table T

Syntax
G45 AXISdir(+/-)

Function type
MODAL
Cancel
G43 ***(optional correction set by Tool Table)***
Parameters
AXIS ***Axis on which apply length compensation***
dir ***Direction of compensation***
+ positive
- negative

Description
G45 works like G43 but the length of tool is taken only in the tool table with **Tn**.

Ex:
G45 Z+ // ENABLE POSITIVE TOOL LENGTH COMPENSATION ON AXIS Z



25.2.5 G46 – – Disable tool length compensation G45

Syntax
G46

Function type
MODAL
Cancel
G45

Description
G46 disable tool length compensation set with G45

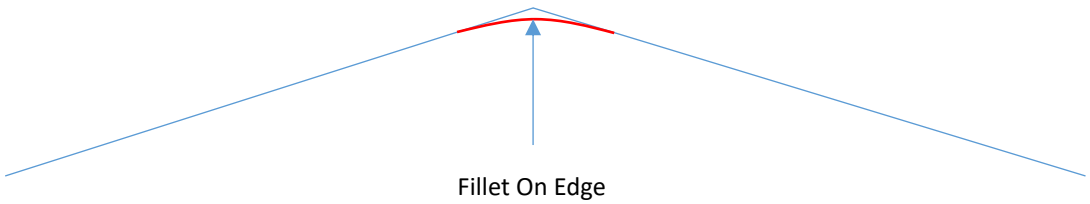
25.3 INTERPOLATION MODES

Isous can use several axis interpolation modes allowing to adapt it in a lot of types of machine.

25.3.1 G60 – Enable FAST interpolation without stop on segments

Syntax
G60
Function type
MODAL
Cancel
G61

Description
G60 enables interpolation without stop on segments. The threshold of angle under which axis don't stop is set by machine parameter.
In the points where axis don't stop, it is generated an optimal trajectory minimizing error. **G60** avails a buffer of movements inside CN which can be set from a value up to **1024** on NG35 series. It allows to work in fluid way paths formed by micro-segments.



Fillet On Edge

25.3.2 G61 – Enable interpolation with stop on segments

Syntax
G61
Function type
MODAL
Cancel
G60

Description
G61 enables interpolation with stop on segment. In this interpolation mode axis are stop at the end of each segment.

25.3.3 G62 – Wait for stop axis

Syntax
G62
Function type
MODAL

Description
G62 waits until axis are stopped and movement buffer is empty. It is useful when CN is in fast interpolation and we need to wait all movements are terminated. Obviously It is obsolete working in G61 (stop on segment).

25.3.4 G63 – Interpolation outside work plane (PX_MOVETO)

Syntax
G63
Function type
MODAL
Cancel
G64-G65

Description
G63 enables interpolation with axis outside work plane. In this way it's possible to work trajectories in the space (3D) without to consider the work plane.
The stop on segment is programmed by G62. G63 is particularly useful in 3D working where movement is formed by three or more axis. Generally in this interpolation mode are executed working from CAM at which is demanded the calculation of edge on segments inserting G62 in the fit point.

25.3.5 G64 – Interpolation on work plane (default)

Syntax
G64
Function type
MODAL
Cancel
G63-G65

Description
G64 enables interpolation mode considering the work plane. This is the default mode selected on CN. **G64** calculates vectorial speed only with the axis of work plane (PX_LINETO), moving consequently the other.
Isous verifies automatically if there are movement of the work plane axis, otherwise it works as G63.
Using this mode it can verify some inconvenient when the two axis of the work plane do very little movement. The following example explains that.

G68 MUST BE ENABLED

Ex:

G64
G17
G1X0.001Y0.001Z10

In the above example axis X and Y on work plane move very little in relation to axis Z causing an high speed on axis Z that can be bigger than its limit.
The solution to that is the following:

G63
G17
G1X0.001Y0.001Z10
G64

Enabling interpolation outside work plane, Isous calculates the vectorial speed with all axis instead only the two axis of work plane.

25.3.6 G65 – Enable 3D interpolation PX_MOVETO with calculated stop on edge by parameters

Syntax
G65
Function type
MODAL
Cancel
G63-G64

Description
G65 Enables 3D interpolation calculating stop on edge by CN parameters.

G63-G64 MUST NOT BE ENABLED

Ex:
G65
G1X100Y100Z10

In the example PX_MOVETO (interpolation outside work plane) is used and stop on edge is calculated automatically by CN.

25.3.7 G75 – Enable G64 for movement inside the work plane , G65 for movement outside the work plane

Syntax
G75

Function type
MODAL
Cancel
G63-G64

Description
G75 Enable interpolation 2D **G64** when the movements are inside the work plane with **SGLP**, Enable interpolation 3D **G65** when the movements are outside the work plane with **SGL3D.**
G63-G64 MUST NOT BE ENABLED

25.3.8 G66 – AFC – Adaptive Feed Control

Syntax
G66 Xval Yvin Zval Aval

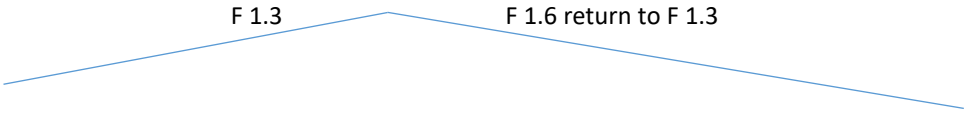
Function type
Disabled when PartProgram start

Parameters

Xval ***0=disable AFC, 1=Enable AFC***
if greater than 1, is carried out a moving average on the value of F calculated and the number of samples indicated in xVal. This makes it more "soft" means the change of the F calculated.

Yvin ***(optional) axis inertia***
Vin > 1 deceleration space increases
If it isn'n present Vin=1 is set (no inertia)

Zval ***(optional)*** **Indicates the values of Delta F to below which are not considered.**
ex:
Val = 0.3, a delta F d 0.3 or less is not considered and therefore the F is left to the previously calculated.



Aval **Minimum value below which is not diminished the F**
Ex: if Val = 0.1 to 0.1 lower values of F are limited to 0.1

ATTENTION
To correct working of G66, also G69 (look ahead depth) must be set

Description

G66 enables the **AFC** filter to auto-calculate interpolation speed by path type. With **SGLAFC_n** parameters, the **AFC** algorithm computes the optimal speed for each segment. The analysis of speed is carried out in the ***look ahead*** segments buffer (set by G69). Obviously more the buffer is deep, more the result is accurate. The **AFC** algorithm works only on **G1** segments and it makes sense only in paths formed by **MICRO-SEGMENTS**. It's basilar understand **AFC** can change the axis speed in each single segment and therefore the resulting speed can be inconstant along path. The **SGLAFC_** parameter of each single axis defines the instant speed gap it can bear without problems. If, in the analyzed segment, even a single axis exceeds its threshold, the speed is reduce to fit set parameters. In the presence of edges (defined by **SGL3D_n** parameters) axis are decelerated to a stop.
AFC is very useful in works where it doesn't need a constant speed along path. In such a cases **AFC** allows to move at speed as high as the path makes it possible, slowing in critical points (curves).

25.3.9 G66 X-100 – NEW AFC – Adaptive Feed Control

Syntax

G66 X-100 Yarc Zaccs Afmin Bflevel Cval UaxixIndex VminLen WangDiff FnumSeg SradiusMax

Function type

Disabled when PartProgram start

Parameters

X-100	***NAFC - New AFC enabled***
	0 Disabled
	-101 Disabled in PREVIEW
	Use X-101 for enabled the NEW AFC but disabled in PREVIEW
	Some times the new AFC can slow the PREVIEW function
Ypeaks	***(optional) Enabled only if C parameter has the bit 8 set to 1 (new algorithms) define the value for peaks feed reduction(see following)***
	default=0
Zaccs	***(optional) Inserts a SMOOTHING filter on ACC PEAKS***
	Decimal Values from 0 to 1 (ex Z0.7) big values, means filter weak. **Default value=0**
Afmin	***(optional)*** **Minimum value below which is not diminished the F**
	Ex: if Val = 0.1 to 0.1 lower values of F are limited to 0.1. Default value Disable
BFastMode	***(optional) Fast Mode for Acc calculation***
	FastMode=0 The Acceleration is controlled also on the segments of the FindArc function
	FastMode=1 The Acceleration is not controlled on the segments of the FindArc function Default FastMode=1
Cval	***(Optional)*** **Special Functions see description. Default value=0**
UAxisIndex	***(Optional) If is used the tangential Axis*** **(like to cutter) insert the Axis Index (0,1,2) that indicate the tangential Axis.**
VminLen	***(Optional)*** **Indicates the minimum length (mm) for G1 element to be considered an ARC. G1 elements greater this value, are not considered as an ARC. Used only when the FindArc function is enabled parameter C bit 4=1. Default value=10 mm**
WRadiusTol	***(Optional)*** **Indicates *the tolerance of the Radius for FindArc function***
	All the G1 segments which fall within the indicated toleranceare considered members of the same ARC. Higher values give less precise data.
	Default value 5 (5 % of tolerance
FnumSeg	***(Optional)*** **Indicates the minimum number of consecutive G1 elements that respect the parameters V and W to be considered an ARC. Used only when the FindArc function is enabled parameter C bit 4=1. Default value=3**
SradiusMax	***(Optional)*** **Indicates the maximum radius calculated by FindArc function. Radius greater this value are not considered as an ARC. Used only when the FindArc function is enabled parameter C bit 4=1. Default value=100 mm**

Description

G66 X-100 enables the new algorithm **NAFC** for Feed calculate on the Gcode files.

This new algorithm is better than old **G66** and uses the following parameters:

ACC_MAX_X,Y,Z etc.

The **ACC_MAX_** parameter must be set to a value greater or equal to **ACC_LAV** parameter. It means the max acceleration that the Axis can support, over it , the FEED is reduced.

G66 X-100 uses a look ahead depth equal to CNC, therefore is not necessary set the **G69** function.

Usually is not necessary insert other optional parameters for the **G66 X-100**, only the parameters **ACC_MAX_** are setted, but is possible improve the algorithm by the **Z** and **B** parameters.

Z decimal values from **0** (disabled) to **5** (minimum effect) uses a smoothing on ACC PEAKS Reference value could be **Z0.5**.

With this parameter, are avoided big feed reduction.

B integer values from **0** (disabled) to **5** (maximum effect) improve the feed calculation.

If is necessary excluded the **G2 G3** from the algorithm, set the **Y** to **1**.

In this case,the **G2-G3** feed reduction is performed by **ACC_RAGGIO_MAX** parameter.

Set **ACC_RAGGIO_MAX** to a value **90** if the **G2-G3** feed reduction is included in the algorithm.

If is used the tangential Axis (Cut Plotter) insert in the **U parameter**, the index of axis (0-1-2 etc.)

The tangential axis speed control is made when the system must interpolate on the work-plane, with the optional third axis moved in position “in time”. For example with the “transported axis” (**G68**) or on the arcs (**G2-G3**) where the speed is always and only calculated on the work-plane axes.

In these cases, on every segment, the actual speed of the third axis will be compared with the maximum (**VMAX_**) inserted in the parameter list. If it’ll be overcome, a new interpolation speed will be calculated, to make the tangential axis to move at his maximum speed. In this way, the segment will be executed at the maximum possible speed, but always supported by all axes.

Setting the bit 5 to 1, this control will be disabled.

FindArc

If the Bit 4 of parameter C is set to 1 (decimal value=16), is enabled the algorithm **FindArc**. It search ARCS in the consecutive G1 elements by parameters **V W F S**.

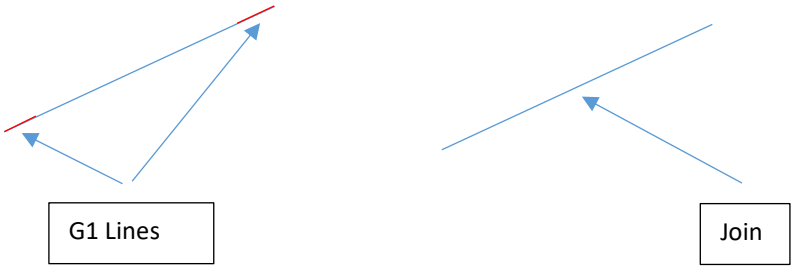
If an ARC is found, the RADIUS is calculated and the FEED is reduced by **ACC_RAGGIO_MAX parameter**, as G2-G3 element.

This function can operate with **NAFC** algorithm (feed reduced by **ACC_MAX_**) but this has the priority.

C Parameter Special Functions Bit Mapped

Bit 0=1 **RLS Removes the Short Segments G1 tath can't be worked at current FEED**

Bit 1=1 **JOING1 Join the G1 belonging the same line G1**



Bit 2=1 **Excludes the FEED reduction**
Only RLS and JOING1

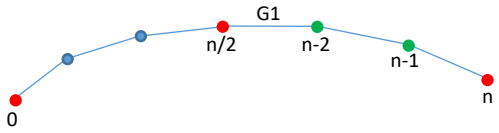
Bit 3=1 **Excludes NAFC. Feed calculation by ACC_MAX_**

Bit 4=1 **Enables FINDARC**

Bit 5=1 **Excludes tangential axis from FEED control**

Bit 6=1 **Use First Point, Middle Point and Last Point for FindArc (0,n/2,n)**

Bit 6=0 **Use Last Three Points for FindArc (n-2,n-2,n)**



Bit 7=1 **G2-G3 Outside ACC_RAGGIO_MAX are taken for FindArc as G1**

Bit 8=1 **Enable the new *algorithms for G*66 X-100 (recommended) some parameters have the following meaning**

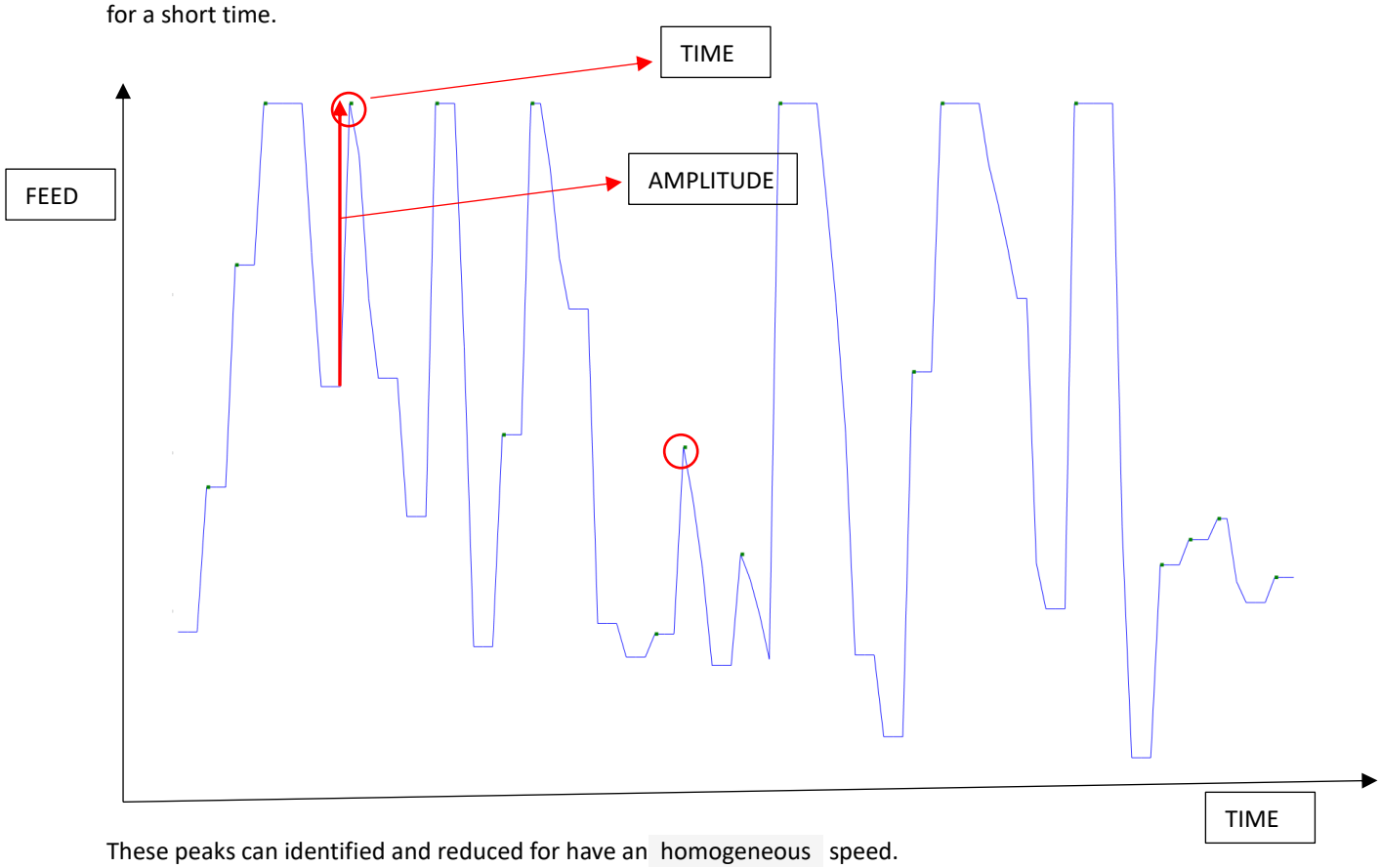
ZFdiv ***(optional) Divisor factor for FEED reduction during acceleration peaks. The FEED calculate by algorithm is further divided by this value.***
Default 0 (disabled)

FPamp ***(optional) Value of FEED amplitude for define a peak (see following)***

BPtime ***(optional) Value of FEED time for define a peak (see following)***

Feed peaks reduction

The algorithms for automatic FEED reduction, can produce some unnecessary FEED PEAKS i.e. movements at max feed for a short time.

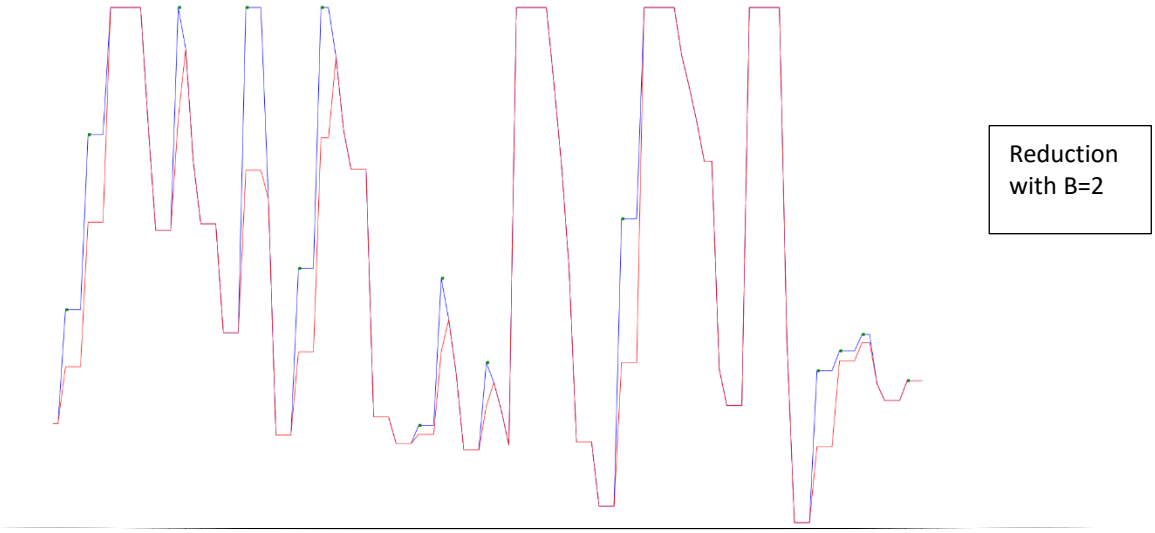


These peaks can identified and reduced for have an homogeneous speed.

B parameter enable the peaks reduction feed. It defines the TIME.
B=1 all peaks are reduced. B=4 only the peaks that have the time<=4 unity are reduced (default 0 disabled)

F parameter defines the peaks amplitude (mt/min). Praticamente vengono considerati picchi solamente se hanno una
F=0 all peaks are reduced. F=8.1 only the peaks that have the feed >= 8.1 mt/min unity are reduced (default 0 all peaks)

Y parameter defines the divisor factor for feed reduction in a peak. **Y**=2 the feed is decreased of 50%.
(default **2** 50%)



25.3.10 G67 – Px_Moveto for movement outside plane and Px_Lineto for inside one

Syntax
G67
Function type
MODAL
Cancel
G68
Description
G67 works with a combination of PX_MOVETO for movement outside work plane (3D) and PX_LINETO for movements inside work plane. Furthermore when PX_MOVETO is used axis are stop at end of segment.

G63-G65 MUST NOT BE ENABLED
G68

25.3.11 G68 – Always using of PX_LINETO in G1 "TRANSPORTED AXIS" (with possibility to combine with PX_MOVETO

Syntax
G68 Xval YTangAxis
Function type
MODAL
Cancel
G67
Parameter

Xval	***Optional parameter (it must be enabled G69)*** ***val>0 limit speed of eventually tangential axis*** ***If tangential axis exceed this speed transported axis is disabled and PX_MOVETO is invoked*** ***Val=0 disable*** ***Val=-1 axis is never transported, but it is only calculated the edge threshold on work plane (as G65 but with SGLP)***
YTangAxis	***Optional Index of axis to be set as TANGENTIAL***

Description
G68 invokes always PX_LINETO for movement both inside and outside work plane, allowing to manage interpolation for particular machine types (for ex. cutting with tangent blade).
AXIS OUTSIDE WORK PLANE BECAME "TRANSPORTED"
These axis (outside plane), moving to final position with determination of vectorial speed by the only axis on work plane, don't' affect the resulting speed of tool.
An example of use with this interpolation mode can be the CUTTING PLOTTER one, where the blade must be tangent to path. Resulting working speed is realy the programmed one by F, tangential axis instead will be moved to its final position in the same time of the other.
X parameter allows to obtain mixed interpolation between "TRANSPORTED AXIS" and normal axis.
Using this interpolation mode there can come out a problem: the "TRANSPORTED AXIS" can reach a speed over its limit. Setting this limit with X parameter Isous identifies such points and invoke PX_MOVETO reducing resulting speed.

G63-G65 MUST NOT BE ENABLED

25.3.12 G69 – LHK – Buffer look ahead depth

Syntax

G69 Xval AXISTANG

Function type

Disabled when PartProgram start

Parameters

val ***Number of elements in the buffer***
If VAL<20 AFC is disabled
VAL is limited to number of blocks of PartProgram

ASSIXTANG ***Optional axis to be set as TANGENTIAL (Ex. Z A etc.)***
If it isn't present no tangential axis is set

Description

G69 defines and enables the deep of **look ahead** segments buffer. It allows proper operation in functions which use it (G66 – G72 – G73 – G74). More the buffer is deep, more the result is accurate.
Depth is however limited to a value <20 (LHK disabled) to value major of part program length. A typical value can be 200.
This function also defines an optional tangential axis (for ex. cutting blade). This is necessary because functions related to **G69** (G66 – G72 – G73 – G74) can change the original path, adding or removing segments necessitating to recalculate the angle of tangent axis.

Ex:

G69 X200 A

In the above example buffer depth is set at 200 elements and axis A is set as TANGENTIAL.

25.4 ISOUS FILTERS

Isous is equipped with several filtering algorithms to modify original path formed by PartProgram. Filter can be useful especially when PartPrograms comes from CAM (for ex. scanner acquisition) having a “NOISE” path.
Filters acts only in G1 blocks and however it must be enabled G69 function (LHK depth).
They can be **2D** features (work plane axis) or **3D** features (work plane axis and depth axis).
Filters however **DOESN’T ACT ON MORE THAN THREE AXIS**.

WARNING
The depth axis is automatically taken from axis XYZ which remains outside work plane.
Work plane X,Y Depth axis Z
Work plane X,Z Depth axis Y
Work plane Y,Z Depth axis X

When filtering is active current working lines in PART PROGRAM can be shifted to original ones.
It can happen some of these are removed and other are divided in more segments.

SEQUENCE OF FILTER
All available filters can be combined together, it can be invoked in following sequence:

NOISE	**G73**	**(if enabled)**
NURBS G72	**(if enabled)**	
FINE NOISE	**G73**	**(if enabled)**
RLS	**G74**	**(if enabled)**
SMOOTHING	**G106**	**(if enabled)**
AFC	**G66**	**(if enabled)**

25.4.1 G72 – N.U.R.B.S (Non Uniform Rational Bspline) (2D 3D)

Syntax
G72 Xval Yminlen Zorder Alenseg

Function type
Disabled when PartProgram start

Parameters

val ***0 – disable NURBS filter***
1 – enable NURBS filter

minlen ***Minimum G1 segment length to be inserted in NURBS filter.***
Filter acts in a buffer which has a continuous series of segments with a length less than MINLEN parameter. When a segment greater than this length is met, the NURBS buffer is computer making an interpolation on points.
The typical default value is 0.2 mm
Optional parameter, if it isn't present last value or default value is taken.

order ***Order of NURBS curve (value from 1 to 7)***
It defines the accuracy of resulting. Lower value corresponds to a best accuracy.
order=1 original point are carried back.
Typical default value is3.
Optional parameter, if it isn't present last value or default value is taken.

lenseg ***It defines the minimum segments length at output curve resulting to points interpolation. At lower value corresponds to a less segments length and to a greater number of resulting segments (curve more accurate)***
Typical default value is 0.2mm
Optional parameter, if it isn't present last value or default value is taken.

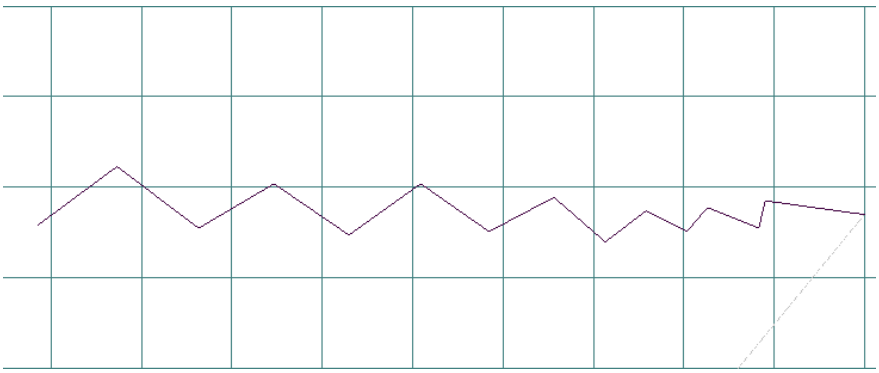
Description
NURBS are a type of curve suitable to solve B-Spline problem, mainly the impossibility to draw simple shape as circle. By this evolution it's possible to generate conic curves, and to represent very complex curves with less number of control points.
N.U.R.B.S. (Not Uniform Rational B-Spline) are rational curves defined by control points and its relative weights. Increasing the weights value of a point the curve approaches to the point. On the contrary the point will cause a less variation on the curve.

ISOUS uses this interpolation mode to **"SOFTEN"** a series of segments.
Some example are showed below to better explain NURBS filter.
NURBS filter acts also on 3D paths

NURBS FILTER IS THE SECOND TO BE INVOKED ON ORIGINAL PATH

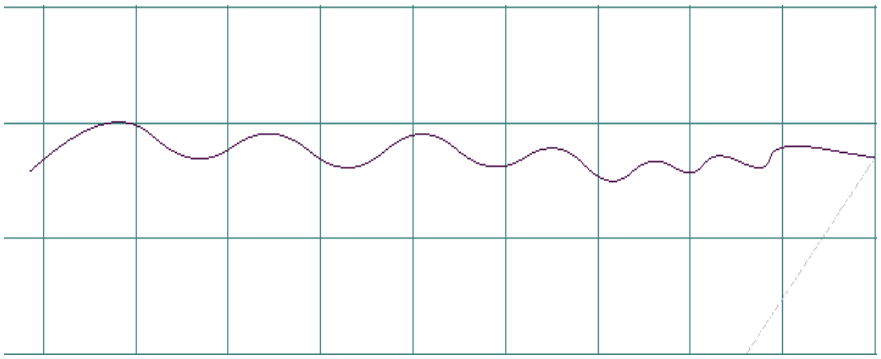
ORIGINAL PATH

G0 X107.16 Y130.27
G1 X100.59 Y131.18
G1 X100.14 Y129.37
G1 X96.74 Y130.73
G1 X95.38 Y129.14
G1 X92.66 Y130.5
G1 X89.95 Y128.46
G1 X86.55 Y131.41
G1 X82.24 Y129.14
G1 X77.71 Y132.31
G1 X72.95 Y128.91
G1 X67.97 Y132.31
G1 X62.98 Y129.37
G1 X57.55 Y133.45
G1 X52.34 Y129.59



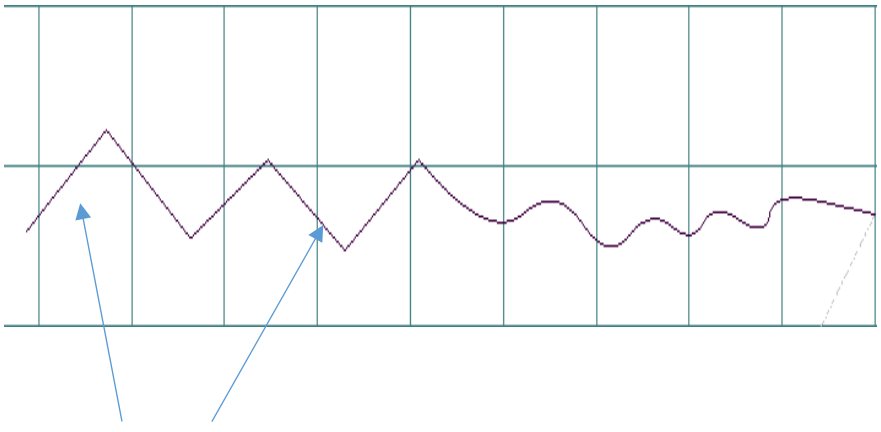
PATH FILTERED with NURBS MiLen=7mm Order=3 LenSeg=0.2mm

G69X200 // ENABLE LHK
// ENABLE NURBS LEN 7 MM ORDER 3 SEGMENT 0.2
G72X1Y7
G0 X107.16 Y130.27
G1 X100.59 Y131.18
G1 X100.14 Y129.37
G1 X96.74 Y130.73
G1 X95.38 Y129.14
G1 X92.66 Y130.5
G1 X89.95 Y128.46
G1 X86.55 Y131.41
G1 X82.24 Y129.14
G1 X77.71 Y132.31
G1 X72.95 Y128.91
G1 X67.97 Y132.31
G1 X62.98 Y129.37
G1 X57.55 Y133.45
G1 X52.34 Y129.59



PATH FILTERED with NURBS MiLen=5mm Order=3 LenSeg=0.2mm

G69X200 // ENABLE LHK
// ENABLE NURBS LEN 5 MM ORDER 3 SEGMENT 0.2
G72X1Y5
G0 X107.16 Y130.27
G1 X100.59 Y131.18
G1 X100.14 Y129.37
G1 X96.74 Y130.73
G1 X95.38 Y129.14
G1 X92.66 Y130.5
G1 X89.95 Y128.46
G1 X86.55 Y131.41
G1 X82.24 Y129.14
G1 X77.71 Y132.31
G1 X72.95 Y128.91
G1 X67.97 Y132.31
G1 X62.98 Y129.37
G1 X57.55 Y133.45
G1 X52.34 Y129.59

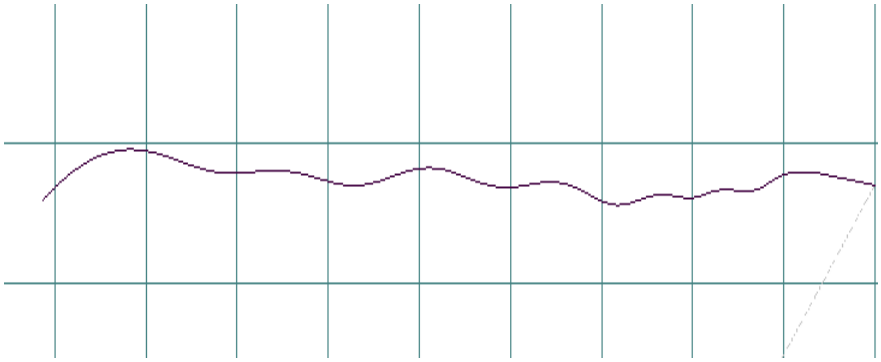


SEGMENT LENGTH >= 5 mm (not filtered)

PATH FILTERED with NURBS MiLen=7mm Order=5 LenSeg=0.2mm

With higher order NURBS path deviates most from the original path

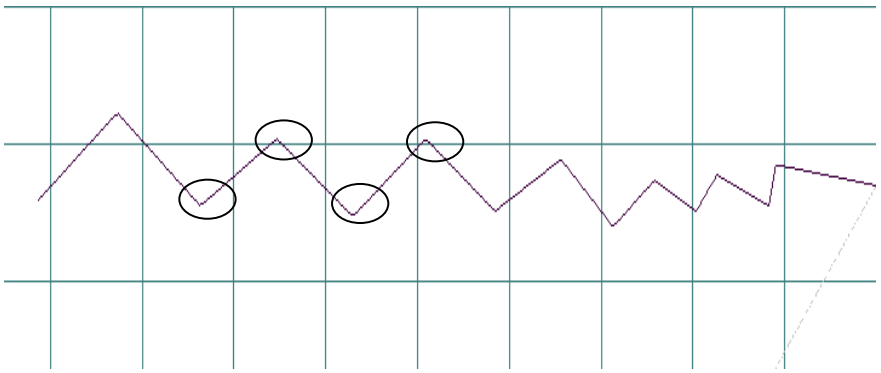
G69X200 // ENABLE LHK
G72X1Y7Z5
G0 X107.16 Y130.27
G1 X100.59 Y131.18
G1 X100.14 Y129.37
G1 X96.74 Y130.73
G1 X95.38 Y129.14
G1 X92.66 Y130.5
G1 X89.95 Y128.46
G1 X86.55 Y131.41
G1 X82.24 Y129.14
G1 X77.71 Y132.31
G1 X72.95 Y128.91
G1 X67.97 Y132.31
G1 X62.98 Y129.37
G1 X57.55 Y133.45
G1 X52.34 Y129.59



PATH FILTERED with NURBS MiLen=7mm Order=2 LenSeg=0.2mm

With **Order=2** changes are almost null even in marked circles original path is slightly modified

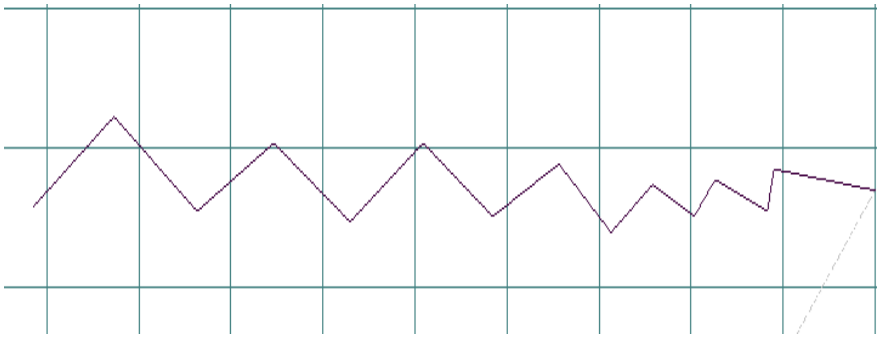
G69X200 // ENABLE LHK
G72X1Y7Z1
G0 X107.16 Y130.27
G1 X100.59 Y131.18
G1 X100.14 Y129.37
G1 X96.74 Y130.73
G1 X95.38 Y129.14
G1 X92.66 Y130.5
G1 X89.95 Y128.46
G1 X86.55 Y131.41
G1 X82.24 Y129.14
G1 X77.71 Y132.31
G1 X72.95 Y128.91
G1 X67.97 Y132.31
G1 X62.98 Y129.37
G1 X57.55 Y133.45
G1 X52.34 Y129.59



PATH FILTERED with NURBS MiLen=7mm Order=1 LenSeg=0.2mm

Original path not filtered

G69X200 // ENABLE LHK
G72X1Y7Z1
G0 X107.16 Y130.27
G1 X100.59 Y131.18
G1 X100.14 Y129.37
G1 X96.74 Y130.73
G1 X95.38 Y129.14
G1 X92.66 Y130.5
G1 X89.95 Y128.46
G1 X86.55 Y131.41
G1 X82.24 Y129.14
G1 X77.71 Y132.31
G1 X72.95 Y128.91
G1 X67.97 Y132.31
G1 X62.98 Y129.37
G1 X57.55 Y133.45
G1 X52.34 Y129.59



25.4.2 G73 – NOISE (2D-3D)

Syntax
G73 Xval Yminang Zminlen Aonlyvar

Function type
Disabled when PartProgram start

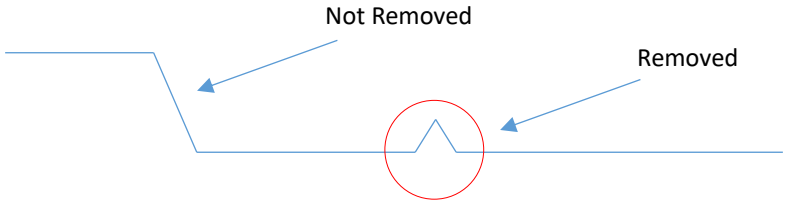
Parameters

val ***0 – disable NOISE, disable FINE NOISE***
1 – enable NOISE, disable FINE NOISE
2 – disable NOISE, enable FINE NOISE
3 – enable NOISE, enable FINE NOISE

minang Minimum edge angle to be consider noise
Typical default value is 25 degree
Optional parameter, if it isn't present last value or default value is taken.

minlen ***Minimum G1 segment length to be inserted in NOISE filter.***
Filter acts in a buffer which has a continuous series of segments with a length less than MINLEN parameter. When a segment greater than this length is met, the NOISE buffer is computer making an interpolation on points.
Typical default value is 0.4 mm
Optional parameter, if it isn't present last value or default value is taken.

onlyvar 0 – remove all edge matching the parameters
1 – remove only cusps matching the parameters (sudden changes of angle)
Typical default value is 0
Optional parameter, if it isn't present last value or default value is taken.

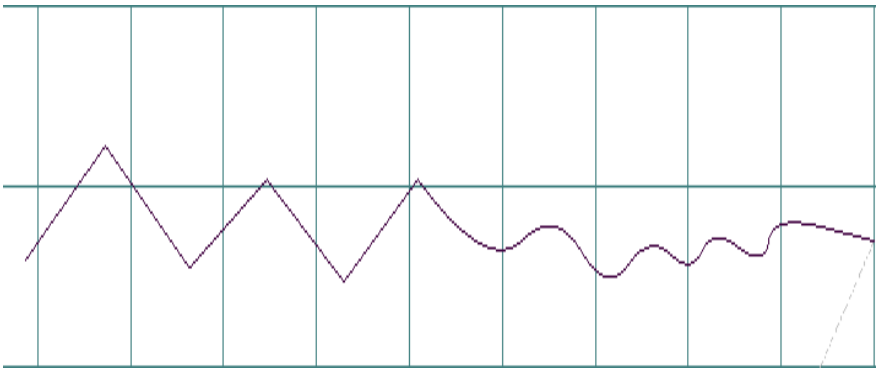


Description
NOISE filter helps to remove **G1** segments forming small edge (or cusps) which are defined as noise.
This is very useful in combination with NURBS filters increasing its performance.
Some example are showed below to better explain NOISE filter.
NOISE filter acts in 2D or 3D paths

NOISE FILTER IS THE FIRST FILTER IT IS INVOKED IN THE ORIGINAL PATH

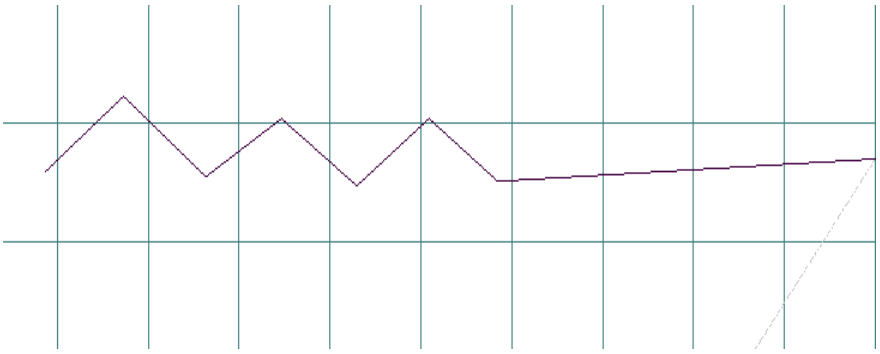
ORIGINAL PATH

G0 X107.16 Y130.27
G1 X100.59 Y131.18
G1 X100.14 Y129.37
G1 X96.74 Y130.73
G1 X95.38 Y129.14
G1 X92.66 Y130.5
G1 X89.95 Y128.46
G1 X86.55 Y131.41
G1 X82.24 Y129.14
G1 X77.71 Y132.31
G1 X72.95 Y128.91
G1 X67.97 Y132.31
G1 X62.98 Y129.37
G1 X57.55 Y133.45
G1 X52.34 Y129.59



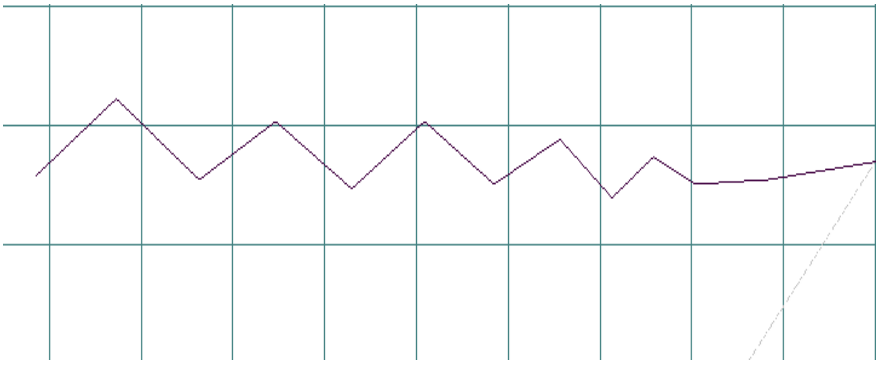
PATH FILTERED with MinAng=45 MinLen=5

G69X200 // ABILITA LHK
G73X1Y45Z5
G0 X107.16 Y130.27
G1 X100.59 Y131.18
G1 X100.14 Y129.37
G1 X96.74 Y130.73
G1 X95.38 Y129.14
G1 X92.66 Y130.5
G1 X89.95 Y128.46
G1 X86.55 Y131.41
G1 X82.24 Y129.14
G1 X77.71 Y132.31
G1 X72.95 Y128.91
G1 X67.97 Y132.31
G1 X62.98 Y129.37
G1 X57.55 Y133.45
G1 X52.34 Y129.59



PATH FILTERED with MinAng=45 MinLen=3

G69X200 // ABILITA LHK
G73X1Y45Z3
G0 X107.16 Y130.27
G1 X100.59 Y131.18
G1 X100.14 Y129.37
G1 X96.74 Y130.73
G1 X95.38 Y129.14
G1 X92.66 Y130.5
G1 X89.95 Y128.46
G1 X86.55 Y131.41
G1 X82.24 Y129.14
G1 X77.71 Y132.31
G1 X72.95 Y128.91
G1 X67.97 Y132.31
G1 X62.98 Y129.37
G1 X57.55 Y133.45
G1 X52.34 Y129.59



25.4.3 G74 – RLS Remove Len Segment (2D 3D)

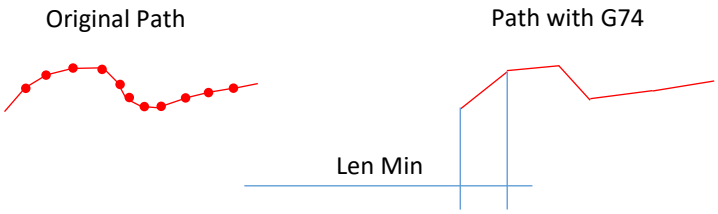
Syntax
G74 Xval Ylenmin

Function type
Disabled when PartProgram start

Parameters

val ***0 – diable RLS filter***
1 – enable RLS filter

lenmin ***Minimum G1 segment length allowed.***
Segments with length less than this value are removed.
Typical default value is 0.5 mm
With lenmin=0 the value is automatically calculated by programmed speed (F).
Optional parameter, if it isn't present last value or default value is taken.



Description
RLS filter allows to remove **"SHORT" G1** segments which couldn't be worked at requested speed (travel time less than CN sample). Setting a **ZERO** value, Isous calculates automatically the length optimizing it to current speed.
The result is an exclusion of "void segment", freeing the movement BUFFER and allowing greater work speed.

ATTENTION
When RLS is enable override potentiometer is disabled.

RLS filter atcs on both 2D and 3D paths

RLS FILTER IS 4th TO BE INVOKED ON ORIGINAL PATH

25.4.4 G106 – Smoothing (2D 3D)

Syntax

G106.ax Xval ALevel Bminlen

Function type

Disabled when PartProgram start

Parameters

.ax	***Axis index where is set the filter (from 0 to n) normally 0 is X axis ex: G106.0***
val	***0 – Filter Smoothing disabled***
	1 - Filter Smoothing Enabled
Level	***Smoothing level (from 0 to 1)***
	Values low have more effect to smoothing ex. A0.3
	Typical value is 0.7
	Optional parameter, if it isn't present last value or default value is taken.
minlen	***Minimum G1 segment length allowed.***
	Segment more large this value, not are inserted in the filter
	Typical default value is disable
	The length is calculated by axes vector (for all axes).
	Is necessary set only one time
	If the parameter is not present, the last length value is used

Description

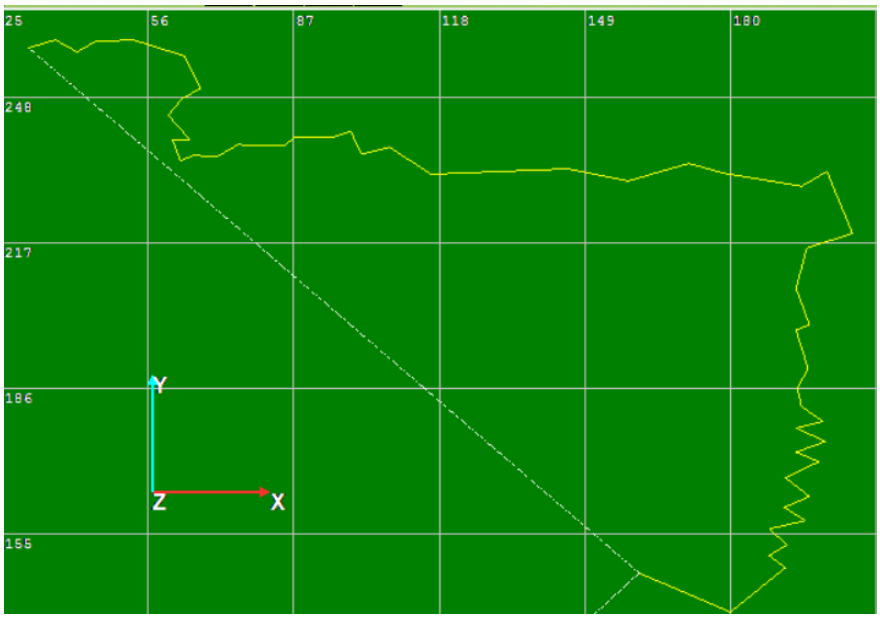
The **SMOOTHING** filter, is used to smoothing G1 segment, it allows to increase the execution speed, but decreased the precision respect to original path.

The **SMOOTHING** is activated for any axes with separate parameters.

THE SMOOTHING FILTER IS 5th TO BE INVOKED ON ORIGINAL PATH

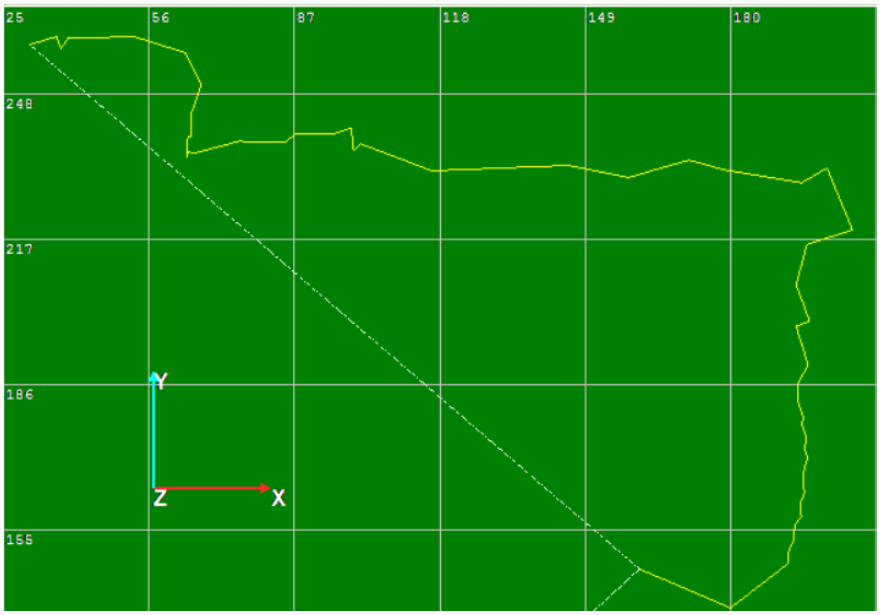
ORIGINAL PATH

G60
G0 Z3
G0X30.4845 Y258.703
F15
G01 Z-1
X36.1945 Y260.416 F15
X40.7625 Y257.8465
X44.7595 Y260.1305
X52.7535 Y260.416
X63.6025 Y256.99
X67.0285 Y250.138
X63.0315 Y247.854
X60.1765 Y244.428
X64.7445 Y239.289
X61.033 Y239.289
X62.746 Y234.721
X65.601 Y235.863
X70.4545 Y235.5775
X75.0225 Y238.147
X84.7295 Y237.8615
X87.299 Y239.86
X95.5785 Y239.86
X99.0045 Y241.002
X101.2885 Y236.1485
X107.284 Y237.576
X116.1345 Y231.866
X145.2555 Y233.008
X158.103 Y230.4385
X170.9505 Y234.15
X178.088 Y232.1515
X194.9325 Y229.2965
X200.357 Y232.437
X205.7815 Y219.304
X196.0745 Y216.1635
X193.7905 Y207.5985
X196.6455 Y199.89
X193.7905 Y198.748
X196.36 Y190.4685
X194.076 Y186.4715
X194.9325 Y182.4745
X199.5005 Y179.334
X193.7905 Y177.9065
X200.0715 Y175.0515
X193.7905 Y172.7675
X198.644 Y170.769
X191.5065 Y167.343
X196.6455 Y163.346
X191.221 Y161.062
X195.789 Y158.207
X188.0805 Y156.494
X191.792 Y153.068
X188.0805 Y150.784
X191.5065 Y148.2145
X179.801 Y138.793
X160.387 Y147.0725
G0 X160.387 Y147.0725



PATH – Smoothing on X,Y level A0.2 Min Len Segment 8 mm B8

G60
G0 Z3
G69X200
G106.0 A0.2 B8
G106.1
G0X30.4845 Y258.703
F15
G01 Z-1
X36.1945 Y260.416 F15
X40.7625 Y257.8465
X44.7595 Y260.1305
X52.7535 Y260.416
X63.6025 Y256.99
X67.0285 Y250.138
X63.0315 Y247.854
X60.1765 Y244.428
X64.7445 Y239.289
X61.033 Y239.289
X62.746 Y234.721
X65.601 Y235.863
X70.4545 Y235.5775
X75.0225 Y238.147
X84.7295 Y237.8615
X87.299 Y239.86
X95.5785 Y239.86
X99.0045 Y241.002
X101.2885 Y236.1485
X107.284 Y237.576
X116.1345 Y231.866
X145.2555 Y233.008
X158.103 Y230.4385
X170.9505 Y234.15
X178.088 Y232.1515
X194.9325 Y229.2965
X200.357 Y232.437
X205.7815 Y219.304
X196.0745 Y216.1635
X193.7905 Y207.5985
X196.6455 Y199.89
X193.7905 Y198.748
X196.36 Y190.4685
X194.076 Y186.4715
X194.9325 Y182.4745
X199.5005 Y179.334
X193.7905 Y177.9065
X200.0715 Y175.0515
X193.7905 Y172.7675
X198.644 Y170.769
X191.5065 Y167.343
X196.6455 Y163.346
X191.221 Y161.062
X195.789 Y158.207
X188.0805 Y156.494
X191.792 Y153.068
X188.0805 Y150.784
X191.5065 Y148.2145
X179.801 Y138.793
X160.387 Y147.0725
G0 X160.387 Y147.0725



25.5 Interrupt MACRO

Allows to use 3 types of MACRO that are execute in **INTERRUPT MODE**.
These MACRO have a fixed NUMBER and must be created with this number

Time	***M50000***
On Digital Input	***M50001***
On Digital Output	***M50002***
Each Axes Movement	***M50003 e M50004***

The management of these macro is performed by **G107** function

25.5.1 G107 – Management Interrupt Macro

Syntax

G107 X0

Disable All Interrupt Macro – Excluding M50003 and M50004

G107 X1

Enable All Interrupt Macro – Excluding M50003 and M50004

G107 X2

Suspend All Interrupt Macro – Excluding M50003 and M50004

G107 X3

Resume All Interrupt Macro – Excluding M50003 and M50004

(If they are disabled remain disabled)

G107.0 Xtime

Macro M50000 activated at Time

Time in millisecond (X-1 Disabled)

When the MACRO is in execution, it is automatically disabled, and will be enabled when the execution is finished

Example:

1) **Generate MACRO 50000**

```
G62                     // WAIT AXES STOP
$SAVEX=$[Q0]            // SAVE X
$SAVEY=$[Q1]            // SAVE Y
$SAVEZ=$[Q2]            // SAVE Z
G0X0Y0Z0                // MOVE AXES
G4F1                    // PAUSE
G0X[$SAVEX]Y[$SAVEY]    // RESUME AXES VALUES XY
G0Z[$SAVEZ]             // RESUME AXIS VALUE Z
G4F1                    // PAUSE
```

2) **Activation in the Gcode each 5 Seconds**

```
G107.0 X5000
```

The macro M50000 will be executed each 5 seconds

G107.1 XDI

Macro M50001 Activated when the DIGITAL INPUT IS ON

DI Indicate the DIGITAL INPUT NUMBER from 0 to 31 (-1 Disabled)

When the MACRO is in execution, it is automatically disabled, and will be enabled when the execution is finished

Example:

1) **Generate MACRO 50001**

```
G62                        // WAIT AXES STOP
$SAVEX=$[Q0]               // SAVE X
$SAVEY=$[Q1]               // SAVE Y
$SAVEZ=$[Q2]               // SAVE Z
G0X0Y0Z0                   // MOVE AXES
G4F1                       // PAUSE
G0X[$SAVEX]Y[$SAVEY]       // RESUME AXES VALUES XY
G0Z[$SAVEZ]                // RESUME AXIS VALUE Z
G4F1                       // PAUSE
```

2) **Activation in the Gcode when the FIRST DIGITAL INPUT IS ON**

G107.1 X0

The M50001 will be executed when the DIGITAL INPUT will be from OFF to ON

G107.2 XDO

Macro M50002 Activated when the DIGITAL OUTPUT IS ON

DO Indicate the DIGITAL OUTPUT NUMBER from 0 to 31 (-1 Disabled)

When the MACRO is in execution, it is automatically disabled, and will be enabled when the execution is finished

Example:

1) **Generate MACRO 50002**

```
G62                        // WAIT AXES STOP
$SAVEX=$[Q0]               // SAVE X
$SAVEY=$[Q1]               // SAVE Y
$SAVEZ=$[Q2]               // SAVE Z
G0X0Y0Z0                   // MOVE AXES
G4F1                       // PAUSE
G0X[$SAVEX]Y[$SAVEY]       // RESUME AXES VALUES XY
G0Z[$SAVEZ]                // RESUME AXIS VALUE Z
G4F1                       // PAUSE
```

2) **Activation in the Gcode when the FIRST DIGITAL OUPUT IS ON**

G107.2 X0

The M50002 will be executed when the DIGITAL OUPUT will be from OFF to ON

G107.3 Xp1 Yp2 Zp3 Ap4 Bp5 Cp6

G107.4 Xp1 Yp2 Zp3 Ap4 Bp5 Cp6

Macro M50003 and M50004 Activated each Axes movement
If the Gcode line, contains the Axes movement, the M50003 or M50004, will called

Parameter:

P1	**0**	**Disable Macro M50003 or M50004**
	1	**Enable Macro M50003 or M50004**
P2..P6		**User Parameter can be used in the M50003 and M50004 code**
		These Parameters are passed by Address
		P2 Address 20000
		P3 Address 20001
		P4 Address 20002
		P5 Address 20003
		P6 Address 20004

Example:

1) **Generate MACRO 50003**

```
$FEED1=$[:20000]          // FEED1 ON P2 Y
$FEED2=$[:20001]          // FEED2 ON P3 Z
F[$FEED1]
G1X0Y0Z0                  // MOVE AXES WITH FEED1 (Y)
G4F1                      // PAUSE
F[$FEED2]
G1X100Y100Z100            // MOVE AXES WITH FEED2 (Z)
G4F1                      // PAUSE
```

1) **Activation in the Gcode**

```
G0X0Y0Z0
G107.3 X1 Y10.2 Z5.3      // ENABLE M50003 WITH FEED1=10.2 AND FEED2=5.3
F2
G1X20Y20Z20     // TO END THIS LINE THE M50003 IS CALLED
G1X30Y30Z30     // TO END THIS LINE THE M50003 IS CALLED
G1X40Y40Z40     // TO END THIS LINE THE M50003 IS CALLED
G107.3 X0       // DISABLE M50003
```

The FEED setted in the Gcode, are saved first the M50003 or M50004 execution, and restored to end Macro

25.6 EMERGENCY MACRO M60000 in STOP Mode

The **EMERGENCY MACRO M60000** is activated when the **CN** finds an **EMEGENCY ALARM** and it is in **STOP** mode
When the CN is in **RUN** mode will be activated the **MACRO ERROR** configured.
The **EMERGENCY MACRO** has a fixed code **M60000**

Therefore is necessary generate a macro **M60000** by PluGIn **MHM** and it will be immediately ready.
Generally in this macro is insert only the code for Digital or Analog outputs management, because the Axes movement will be stooped by CN when the Emergency will be found.

For disactivate the EMERGENCY MACRO M60000 is necessary remove from the folder **DATA_M** the file **M60000.bin**
The folder **DATA_M** is in the IsoUs folder installation.

25.7 G108 Management Special Axes

This function allows of management the Special Axes. This enable special functions on the axis different from normal use.

25.7.1 G108.0 G108.1 G108.2 G108.3 - Master Slaves Axes

The function G108.0 .1 .2 .3 allows to enable the MASTER-SLAVE.
In other words, allows to connect one or more axes to single master. So, when the master it is moved, the slaves axes are moved in the same mode of master axes
Is possible management up to 4 Groups of Master Slaves respectively with G108.0 G108.1 G108.2 and G108.3
The max number of slaves connected to master, is 8.

Syntax

G108.x 0

Remove all slaves of the group

G108.x 1

Suspend all slaves of the group

G108.x 2

Resume all slaves of the group previously programmed

G108.x M,S1,S2,S3..(-)S4 ecc.

Coonects the Slaves

M = Axis Master (is always the first programmed with G108.x)

S1 = Slave 1

S2 = Slave 2

Etc.

The sign "-" before of slave, inverts the slave direction, respect to master

Example:

```
G108.0  X ,A ,-C        // X Master A Slave C slave with invert direction
G108.1  Y,Z ,B,U        // Y Master Z,B,U Slaves
.
.
G108.0  0               // Remove first Group of slaves
```

25.7.2 G108.4 G108.5 G108.6 SPEED MODE AXIS

The **G108.4** function, allows to connect one Axis from Interpolator to **SPEED MODE**, therefore the axis is no longer controlled by a target value, but in velocity mode. The **SPEED** can be set by the Function **S** if the channel is set in correct mode (see 16.8)
The function **G108.5** retrieve the Axis from **SPEED MODE** to **INTERPOLATOR MODE**
The function **G108.6** is like to **G108.5** but the Axis value is RESET to ZERO.
This allows to avoid the AXIS LIMITS ERROR

The Axis status in Speed mode can be read by special variable $[X16]

Syntax

G108.4 X,Y,Z,A,B,C,U,V,W

Connects the Axis (one Only) From Interpolator to Speed mode.

G108.5 X,Y,Z,A,B,C,U,V,W

Retrieve the Axis from Speed Mode To Interpolator mode.

G108.6 X,Y,Z,A,B,C,U,V,W

Retrieve the Axis from Speed Mode To Interpolator mode. VALUE IS RESET TO ZERO

25.7.3 G108.7 G108.8 G108.9 Physical exchange Axes

The **G108.7** allows to exchange a pair of Axes in the **CNC** level
Is possible to exchange one or more pair of Axes
G108.8 Restore the pair of Axes previously exchanged
G108.9 Restore **ALL** pair of Axes previously exchanged

Syntax

G108.7 XA

Exchange X with A

G108.8 XA

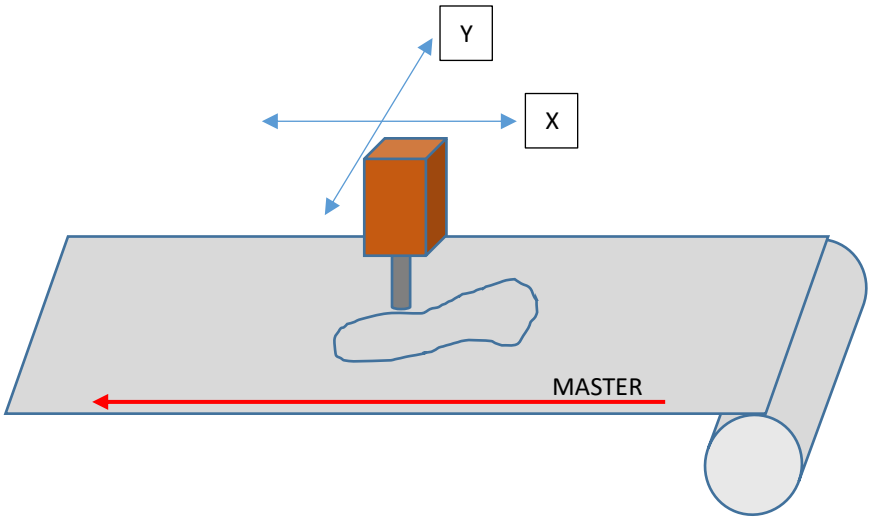
Restore the pair XA in the original mode

G108.9

Restore ALL pair previously exchanged

25.7.4 G108.10 G108.11 eGear Axes Management

G108.10 .11 function, allow to management the Electronic Gear Axes.
By this function, is possible to **LINK** an **INTERPOLATED AXIS X,Y,Z** etc. to an external **ASYNCHRONOUS AXIS** .
For example, a conveyor belt that move the object to work, that can be **FOLLOW** by an interpolated **AXIS** for make the **JOB**.



In the above example, the MASTER AXIS is move in a direction transporting the material to be processed.
The **G108.10** function, allow the **LINK** the **X AXIS** (slave) to **MASTER AXIS** and the **MASTER FEED** will be add to **X FEED AXIS.**
The **G108.10** work with parameters EGEAR_KEM, EGEAR_KED, EGEAR_ACC

Syntax

G108.10 X,Y,Z,A,B,C,U,V,W

LINK the axis (only one) to MASTER defined in the VTB application.

G108.11 X,Y,Z,A,B,C,U,V,W

UNLINK the axis to MASTER defined in the VTB application.

25.7.5 G108.12 G108.13 - Peripheral speed management

G108.12/13 instructions allow to harmonize the peripheral speed of a rotary axis, against the speed of other axes.
This algorithm calc the “shadow” radius, the distance between the work point and the centre of rotary axis and maintain constant the speed of work point, aven with a varing radius.
Result will be a movement with a speed different from the given one, but with a constant speed of work point.
Can be set up two rotary axes, with the related rotation axis.

Syntax

G108.12 Y,B

set axis to be used

G108.13 1

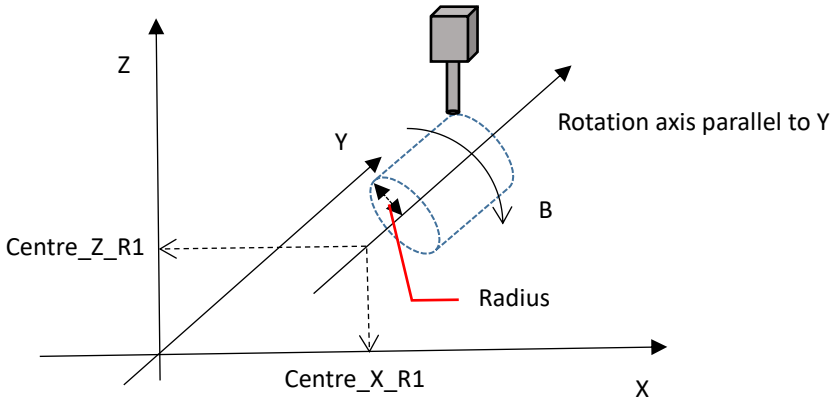
enable management

G108.13 0

disable management

In the example, B axis is set as rotary axis, with Y as rotation axis.
In this case, the “shadow” radius is given by the distance between the work point and the centre of rotation axis defined by CENTRE_X_R1

, **Errore. L'origine riferimento non è stata trovata.** parameters.
More information on centre position definition, check the related section in this manual (see Cap 25.21).
The example declaration, set the rotary axis as explained in picture below.



25.8 VIRTUAL AXIS

Virtual axis are commands inserted in the interpolation buffer but they don't refer to a real axis. In detail they are synchronous commando which can enable events in precise point of path. This commands must be used when CN works with fast interpolation. Furthermore command function must be implemented in CN.

25.8.1 G100 – Comando sincrono per asse virtuale

Syntax
G100 Xcmd

Function type
Direct
Parameter
Cmd ***Command code to send to CN***

Description
G100 sends a command to CN synchronously with path. Such a command must be implemented in CN by VTB environment to work properly (refer to CN documentation).

Example to manage a digital output in fast interpolation.

```
PartProgram
G60       //ENABLE FAST INTERPOLATION
G1X100Y100
X120Y110
X130
G100 X1// SET OUTPUT
X200Y160
X210Y180
G100 X0// CLEAR OUTPUT
X300Y200
X310Y220
```

Two function must be implemented in CN for:
G100 X0 Clear output
G100 X1 Set output
This is the VTB code to be written in TaskPlc using output n.0

```
' G100 command for set/clear output
ISOV1.out0=INTERPOLA1.cmd
```

25.9 OTHER GENERIC G FUNCTIONS

25.9.1 G4 – Timed pause

Syntax
G4 Fvalue
Function type
Direct
Parameter
Value ***Time seconds (resolution up to tenth of second)***

Description
G4 allows to pause PartProgram for the time set by F parameter.

Ex:
G4F2.5 // PAUSE OF 2.5 SEC

25.9.2 G4.1 – Time Add on Calc Time

Syntax
G4.1 Fvalue
Function type
Direct
Parameter
Value ***Time in sec to add***

Description
G4.1 adds an extra time to Calc Time Function. This allows to include extra PLC time (wait input etc.)
Normally it is inserted in the M functions M3,M4,M5 M6 etc.
This function is activated only during Calc Time function, and it is excluded in a normal RUN.

25.9.3 G10 – Enable external OVERRIDE potentiometer

Syntax
G10
Function type
Modal - default
Cancel
G11

Description
G10 enables speed control by external potentiometer.

25.9.4 G11 – Disable external OVERRIDE potentiometer

Syntax
G11
Function type
M***odal***
Cancel
G10

Description

G11 disables speed control by external potentiometer.

25.9.5 G101 – Axis Stop command

Syntax
G101
Function type
Direct

Description
G101 commands a stop to axis in current position. The example below explains this function.

Ex:
In this example axis X move to position 100. During positioning a digital input is tested (Input 0), if it activates the movements is stopped. If axis terminates its movement PartProgram is ended with an error.

```
G60
F1
G1X100
@INITLOOP
IF $[Q0]=100      // IF X REACHES 100
        GOTO ERR        // ERROR
END_IF
IF $[I0]=1        // WAIT FOR INPUT
        G101    // IF ACTIVE, AXIS STOP AND EXIT
        GOTO EXIT
ELSE
        GOTO INITLOOP  // CONTINUE LOOP
END_IF
@EXIT
.
.

@ERR
.
.
```

25.9.6 G80 – Forced pause by code
(Shifted on G1080 if USE_G80_CYCLES=TRUE)

Syntax
G80 Xcode
Function type
Direct
Parameter
Code ***Code of pause (refer to configuration)***

Description
G80 forced CN to pause state and setting a pause code. **CODE** parameter defines the message code to be showed. If code hasn't been configured a message "NO CODE PAUSE" is showed. The PAUSE procedure is the same of the hardware one.

Ex:
```
G80 X4 // PAUSE IS FORCED WITH CODE 4
```

25.9.7 G81 – Management secondary axes LIMITS

(Shifted on G1081 if USE_G80_CYCLES=TRUE)

Syntax

G81 Xmode

Function type

Direct

Parameter

mode	***Mode of LIMITS management***
X0	***Set FIRST POSITIVE LIMITS (default – LIMITE_P_)***
X1	***Set FIRST NEGATIVE LIMITS (default – LIMITE_N_)***
X2	***Set SECOND POSITIVE LIMITS (2ND_LIMITE_P_)***
X3	***Set SECOND NEGATIVE LIMITS (2ND_LIMITE_N_)***

Description

Sometimes is necessary change in temporary mode the Axes limits for make the axes movements outside to primary limits (Eg: TOOL CHANGE)

In Isous is possible used the SECONDARY LIMITS positive and negative

Machine parametrs:

2ND_LIMITE_N_

2ND_LIMITE_P_

This parameters can be activated or disactivated by G81 function.

If the secondary limits remain activated, when the PART PROGRAM is finished, automatically are reset to primary LIMITS.

This function is tipically used to TOOL CHANGE MACRO (M6)

Ex:

```
G81 X3 // ACTIVE SECONDARY NEGATIVE LIMITS
.
G81 X1 // ACTIVE PRIMARY NEGATIVE LIMITS
```

25.9.8 G20 – Axes Values in Inch

Syntax

G20

Function type

Modal

Cancel

G21

Description

G20 Enable the Programmation Axes Values in Inch.(G0-G1-G2-G3 etc)

25.9.9 G21 – Axes Values in Millimeters

Syntax

G21

Function type

Modal - Default

Cancel

G20

Description

G20 Enable the Programmation Axes Values in Millimeters.(G0-G1-G2-G3 etc)

25.10 VARIABLE MANAGEMENT FUNCTIONS

Isous manages files which can contain the state of some variable. These files are stored on the PC HardDisk.
It's possible to load them (one at a time) by Partprogram. When file has been loaded a list of saved variables is filled. The length of the list depends to the number of stored variable. There is no limit to list length. These variables can be read or written in the list by specific instructions in the PartProgram.
In other words it's possible to obtain Backups of variables state for generic use.

25.10.1 LOAD_VAR – load a variables file in current list

Syntax
LOAD_VAR filename

Parameter
Filename ***Name of the file to be loaded***
The name can contain extension but there must not be delimitation characters (+- etc.)*

Description
LOAD_VAR load the file named initializing the internal list with values container in the file.
The list is defined as an ARRAY of double values and we can access them by **GET_VAR** and **WRITE_VAR** functions. If the file will be not found Isous will show the relative error.

Ex:
```
LOAD_VAR MYFILE.PNT   // LOAD FILE
GET_VAR $VAR1 0              // SET VAR1 FROM INDEX 0 OF THE LIST
GET_VAR $VAR2 1              // SET VAR2 FROM INDEX 1 OF THE LIST
```

25.10.2 GET_VAR – Read a value from the loaded list and store it in a variable

Syntax
GET_VAR $var index

Parameters
$var ***Name of the destination variable***
Index ***Expression indicating the index in the list***

Description
GET_VAR reads a value in the current list and stores it in a variable. The list must have been sized (by DIM_VAR) or however it must have a size greater or equal to INDEX. If index is over the list size an error will be showed.

Ex:
```
LOAD_VAR MYFILE.PNT   // LOAD FILE
$INDEX=0
LOOP 10
        GET_VAR $VAR1 $INDEX        // LOAD X POSITION
        GET_VAR $VAR2 $INDEX+1      // LOAD Y POSITION
        G1 X[$VAR1]Y[$VAR2]         // MOVE
        $INDEX=$INDEX+2             // INCREASE INDEX
END_LOOP
```

25.10.3 WRITE_VAR – Write a value in the loaded list getting it from a variable

Syntax
WRITE_VAR $var index

Parametri

$var	***Name of the source variable***
index	***Expression indicating the index in the list***
	ATTENTION: *list must be sized for INDEX value*

Description
WRITE_VAR allows to store a value of the current list in a variable. The list must have been sized (by DIM_VAR) or however it must have a size greater or equal to INDEX. If index is over the list size an error will be showed.
Ex:

```
DIM_VAR 10                  // SIZE LIST WITH 10 EMPTY ELEMENTS
WRITE_VAR $VAR1 0           // STORE VAR1 IN INDEX 0 OF THE LIST
WRITE_VAR $VAR1 1           // STORE VAR2 IN INDEX 1 OF THE LIST
SAVE_VAR MYFILE.PNT         // SAVE FILE
```

25.10.4 SAVE_VAR – save a variables file with current list

Syntax
SAVE_VAR filename

Parameter

filename	***Name of the file to be saved***
	The name can contain extension but there must not be delimitation characters (+- etc.)*

Description
SAVE_VAR saves the value in the current list to the named file.

Ex:

```
DIM_VAR 10                  // SIZE LIST WITH 10 EMPTY ELEMENTS
WRITE_VAR $VAR1 0           // STORE VAR1 IN INDEX 0 OF THE LIST
WRITE_VAR $VAR1 1           // STORE VAR2 IN INDEX 1 OF THE LIST
SAVE_VAR MYFILE.PNT         // SAVE FILE
```

25.10.5 FILE_EXISTS – test if a file exists

Syntax
FILE_EXISTS $var filename

Parameters

$var	***Variable where to store result:***
	0=file not found, 1=file exists
filename	***Name of the file***
	The name can contain extension but there must not be delimitation characters (+- etc.)*

Description
FILE_EXISTS allows to test if a variable file exists.

Ex:

```
FILE_EXISTS $VAR MYFILE.PNT // TEST FILE
IF $VAR=0                   // FILE NOT FOUND
....
END_IF
```

25.10.6 ADD_VAR – Add a value to current list

Syntax
ADD_VAR value

Parameter
value ***Expression or variable which value will be appended to the list***

Description
ADD_VAR appends a value to the current list. The list will be resizing to contain the new element.

Ex:
```
ADD_VAR $VAR1                    // APPEND VALUE FROM VAR1
ADD_VAR 10.23                    // APPEND VALUE FROM CONSTANT
ADD_VAR $VAR2*($VAR3+$VAR4)      // APPEND VALUE FROM EXPRESSION
```

25.10.7 REMOVE_VAR – Remove a value from current list

Syntax
REMOVE_VAR Index

Parameter
index ***Expression indicating index of list to be removed***

Description
REMOVE_VAR remove a value from the current list. All index above the removed one will be re-indexed.
If the index is over the list size an error will be showed.

Ex:
```
REMOVE_VAR 1  // REMOVE VALUE AT INDEX 1
```

25.10.8 CLEAR_VAR – Remove all values from the current list

Syntax
CLEAR_VAR

Description
CLEAR_VAR erase all the values from the current list. List will be sized at zero and the value cannot be restored.

Ex:
```
CLEAR_VAR                // CLEAR ALL CURRENT LIST
```

25.10.9 DIM_VAR – Size current list to a specific number of elements

Syntax
DIM_VAR Nelem

Parametri

Nelem	***Expression indicating the number of elements to be sized*** *(the value of all elements will be set with ZERO)*

Description
DIM_VAR allows to re-size the current list. All last value will be lost and the new values will be set with ZERO.

Ex:

```
DIM_VAR 10          // SIZE LIST WITH 10 ELEMENTS
```

25.10.10 COUNT_VAR – Get the number of elements in the current list

Syntax
COUNT_VAR $var

Parameter

$var	***Variable where to store the list size***

Description
COUNT_VAR gets the number of elements of the current list and store the value in a variable.

Ex:

```
COUNT_VAR $VAR1          // STORE SIZE IN VAR1
IF $VAR1=0               // IF LIST IS EMPTY
    DIM_VAR 10           // RE-SIZE THE LIST
```

25.11 HM FUNCTIONS

HM functions are an upgrade to standard M functions. They can use parameters passed to function when invoked like common high level languages as **C**.
Parameters are then read by HM function writing it in private variables.
HM functions reside on PC and they must be created with the specific utility of human interface.

25.11.1 Call of HM functions

Syntax
HMnum par1 par2 par3 ….

Function type
Direct
Parameters
Num ***Number code of HM function***
PARn ***Variable or constant. Parameters must be separated by space***

Description
HM with code **NUM** is invoked passing it the parameters defined with **PARn**.
HM functions always wait for emptying of the MOVEMENT BUFFER before it is really invoked.
All parameters defined in the declaration must be inserted, else an errore will be generated.

Ex:

```
HM10 $VAR1 $VAR2 12.3 24.125    // CALL OF HM 10 PASSING 4 PARAMETERS
```

25.11.2 Building of a HM function

A **HM** function, before to be used, must be built and pre-compiled by specific option in the human interface.
HM functions are formed by ISO code and they can be used all Isous functions. After pre-compiling **HM** is available to PartProgram.
VARIABLES and **LABELS** declared inside HM function are **PRIVATE** (not visibile in PartProgram). In other words **VARIABLES** and **LABEL** could be a same name both in HM function and PartProgram, Isous recognized that automatically. It's possible to share VARIABLES with Partprogram declaring them with **GLOBAL prefix**.
Inside **HM** function it's possible call other **HM** or **M** functions both on CN and PC.

Example of building of a HM function

```
GET $PAR1 $PAR2 $PAR3 $PAR4     // READ PARAMETERS
GLOBAL $PAUSE                   // SHARE $PAUSE WITH PART PROGRAM
F[$PAR3]                                // SET FEED AT $PAR3
G1 X[$PAR1] Y[$PAR2] Z[$PAR3]   // LINEAR INTERPOLATION
G4 F[$PAUSE]                            // PAUSE
```

As we can see from the example, a HM function is very similar to the high level languages. If there was a need to return a value we can do it using **GLOBAL** variables.

25.12 M FUNCTIONS

There are two type of M function:

- **Inside CN**
- **Inside PC**

Obviously function cannot have the same number. **M inside PC** are priority to **M inside CN**. If both they were present Isous invokes the PC one.

25.12.1 M functions inside CN

M function inside CN have to be built with VTB environment (refer to relative documentation). They can read some parameters from PartProgram or return it a value by some predefined variables.

PREDEFINED VARIABLES

$_PARM_1 **parameter 1**
$_PARM_2 **parameter 2**
$_PARM_3 **parameter 3**
.
.
$_PARM_n **parameter n**

The number of parameters available depends to Isous configuration, but it's a limit of 10 parameters (default 5). Each time PartProgram writes or reads a variable of $_PARM_n type, they are read from CN memory. $_PARM variables are 32 bit **INTEGER**. All parameters must be written before invoking M function and read at the end of it.
M functions always wait for emptying of the MOVEMENT BUFFER before it is really invoked.

Ex:

```
$_PARM_1=134   // WRITE PARAMETER 1
$_PARM_2=$VAR// WRITE PARAMETER 2
$_PARM_3=12600          // WRITE PARAMETER 3
M100                    // INVOKE M100 FUNCTION
$VAR1=$_PARM_1          // READ PARAMETER 1 MODIFIED BY M100
```

25.12.2 M function inside PC

M function inside PC are very similar to HM functions, the only difference is that M functions cannot have any parameter. Therefore exchange of value must be made with GLOBAL variables.
A **HM** function, before to be used, must be built and pre-compiled by specific option in the human interface.
HM functions are formed by ISO code and they can be used all Isous functions. After pre-compiling **HM** is available to PartProgram.
VARIABILES and **LABELS** declared inside HM function are **PRIVATE** (not visibile in PartProgram). In other words **VARIABLES** and **LABEL** could be a same name both in HM function and PartProgram, Isous recognized that automatically. It's possible to share VARIABLES with Partprogram declaring them with **GLOBAL prefix**.
Inside **HM** function it's possible call other **HM** or **M** functions both on CN and PC.

Example of building of a M function

```
GLOBAL $VAR1                    // SHARE $VAR1 WITH PART PROGRAM
GLOBAL $VAR2                    // SHARE $VAR2 WITH PART PROGRAM
G1 X$VAR1 Y$VAR2                // LINEAR INTERPOLATION
```

25.13 Essential M FUNCTIONS

In this section are described the essential M functions. These are necessary for correct machine operation.

25.13.1 START M

This function is execute when START PROGRAM is required. Its use is not important, because the preparatory M functions are inserted in to Part Program

25.13.2 END M

This function is execute when END PROGRAM is required. Its use is not important, because the preparatory M functions are inserted in to Part Program.

25.13.3 STOP M

This function is execute when STOP PROGRAM button is pressed. This function is VERY IMPORTANT.
It usually STOP ALL AXES and Spendle, water etc

Essential Functions In M STOP

1) **G101** **Stop axes**
2) **G62** **Wait axes stop**
3) **SET/RESET output for spenlde,water etc.**
4) **Eventually parking axes**

Se nessuna M di STOP viene configurata, il CNC blocca solamente gli assi nel punto di STOP.
If no M STOP configured, will stop only the axes by CNC.

25.13.4 PAUSE M

This function is execute when PAUSE PROGRAM button is pressed. This function is VERY IMPORTANT.
usually this function stops the spindle. the water and saves the positions axes for pause resume.

Essential Functions In M PAUSE (ex 3 axes)

1) GLOBAL $SAVEX GLOBAL variable for X AXIS
2) ***GLOBAL $SAVEY GLOBAL variable for Y AXIS***
3) ***GLOBAL $SAVEZ GLOBAL variable for Z AXIS***
4) **G62 Wait axes stop**
5) $SAVEX=$[Q0] Save X position
6) ***$SAVEY=$[Q1] Save Y position***
7) ***$SAVEZ=$[Q2] Save Z position***
8) SET/RESET output for spenlde,water etc.
9) Eventually parking axes

If M PAUSE is not configured, will stop only the axes by CNC.

25.13.5 RESUME PAUSE M

This function is execute when START button is pressed after PAUSE PROGRAM. This function is VERY IMPORTANT. usually this function repositions the axes and SET/RESET output for spendle, water etc.

Essential Functions In M RESUME PAUSE (ex 3 assi)

1) **GLOBAL $SAVEX** ***GLOBAL variable for X AXIS***
2) **GLOBAL $SAVEY GLOBAL variable for Y AXIS**
3) **GLOBAL $SAVEZ GLOBAL variable for Z AXIS**
4) **G96 Offset suspension**
5) **G98 Work origins suspension**
6) **G0 Z0 Z axis to safety position**
7) **F1 SPEED axes**
8) **G1 X[$SAVEX] Y[$SAVEY] Axes X,Y, etc. To initial positions (these are saved by M pause)**
9) **G62 Wait axes in positions**
10) **G1 Z[$SAVEZ] Axis Z to initial position (this is saved by M pause)**
11) **G62 Wait Z axis in position**
12) **G97 ResumeOffset**
13) **G99 Resume Work origins**
14) **SET/RESET output for spenlde,water etc.**

If no M resume pause configured, the CNC axes back to the starting point in the interpolation of a set F in the machine parameter "VRIPOS"

25.13.6 GO BLOCK M

This function is execute when START button is pressed by PlugIn ***"GO BLOCK"***.
usually this function repositions the axes and SET/RESET output for spendle, water etc.

Essential Functions In GO BLOCK M (ex 3 assi)

1) **$POSX=$[C0] Axis X position calculated by CNC**
2) **$POSY=$[C1] Axis Y position calculated by CNC**
3) **$POSZ=$[C2] Axis Z position calculated by CNC**
4) **G96 Offset suspension**
5) **G98 Work origins suspension**
6) **G0 Z[0] Z axis to safety position**
7) **F1 SPEED axes**
8) **G1 X[$POSX] Y[$POSY] Axes X,Y, etc. To initial positions**
9) **G62 Wait axes in positions**
10) **G1 Z[$POSZ] Axis Z to initial position**
11) **G62 Wait Z axis in position**
12) **G97 ResumeOffset**
13) **G99 Resume Work origins**
14) **SET/RESET output for spenlde,water etc.**

25.13.7 GO RETRACE M

This function is execute when START button is pressed by PlugIn ***"RETRACE"***.
usually this function repositions the axes and SET/RESET output for spendle, water etc.

Essential Functions In M RETRACE

1) **SET/RESET output for spenlde,water etc.**

25.14 DEFINING DEPTH AXIS

The depth axis is used for some functions Isous. Normally, this axis is related to the work plan selected:

Work Plane XY	***Depth axis Z***
Work Plane XZ	***Depth axis Y***
Work Plane YZ	***Depth axis X***

The depth axis is used by the simulation process and the way PREVIEW and by G47 function

25.14.1 G48 Define Depth axis

Syntax
G48 axis name
Function type
immediate

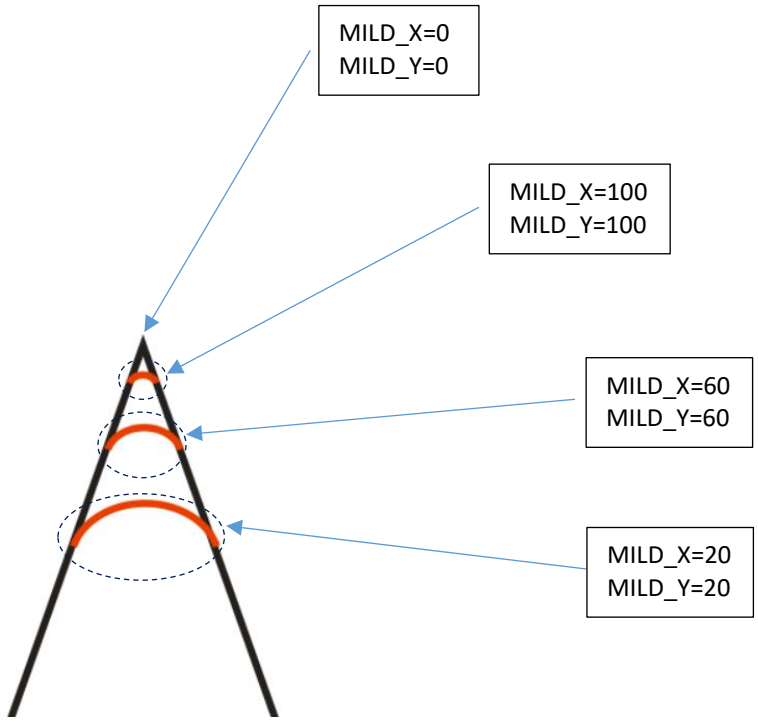
Parameters

Axis Name	***ex: X,Y,Z,A,B etc.***

Description
G48 definisce un asse di profondità diverso da quelli scelti in automatico da Isous.
Se l' asse scelto è uguale ad uno degli assi del piano di lavoro impostato, viene generato un errore di Run Time.
G48 defines a depth axis different to axis chose automatically by Isous
Ex:
G48 A // A is depth axis

25.15 MILD MODE – EDGE SMOOTHING

Isous can use a special algorithm for EDGE SMOOTHING
With the PARAMETERS MILD_ (MILD_X – MILD_Y etc.) you can Smooth the edge with various value.
The MILD MODE can be activated on single Axis.
The MILD_ set of value can be used from 200 to 20 (estimated values)
The G49 function, manages the MILD MODE.
Following you can see a MILD MODE example values
The MILD function, works for EDGE under the parameters **SGLP_MILD** for 2D or **SGL3D_MILD_** for 3D (G65)
Edges, above these values aren't SMOOTHED by MILD



25.15.1 G49 MILD MODE Managing

Syntax
G49 Axes Name where enable MILD MODE – None Axis, MILD MODE DISABLED ON ALL AXES
G49.0 Suspends MILD MODE
G49.1 Resumes MILD MODE on the Axes indicated in previous G49

Function type
Immediate – Disabled on PROGRAM STOP

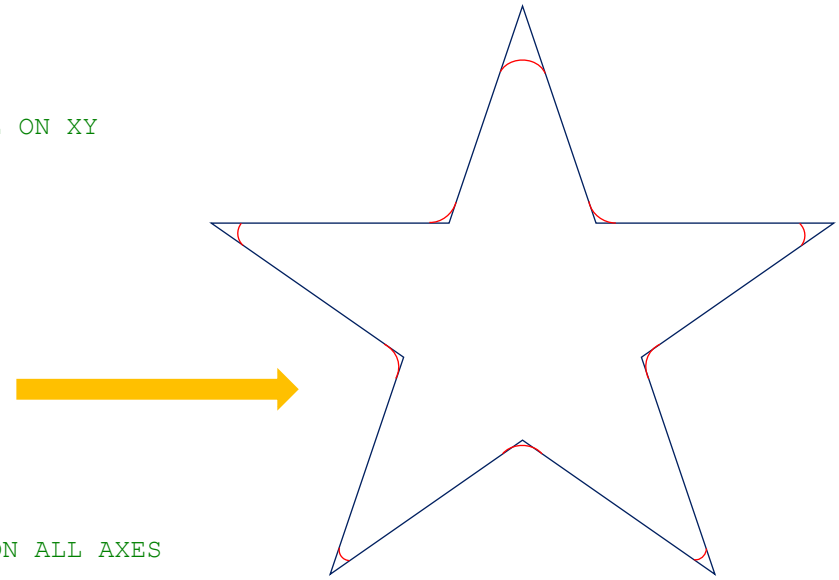
Parameters
Axes Name ***Name of Axes where enable Mild Mode: ex: G49 XYZ (QX,QY,QZ,QA,QB,QC,QU,QV,QW)***

Description
G49 manages ENABLE/DISABLE MILD MODE.
G49 with **Axes Name** (ex. G49 XY) enabled MILD MODE on the Axes indicated.
G49 Without parameters DISABLES MILD MODE on the ALL AXES.

G49.0 Without parameters **SUSPENDS MILD MODE**
G49.1 Without parameters **RESUMES MILD MODE** if it was activated by **G49**.

Ex:

```
G60
F15
G0 Z3
G49 XY // ENABLED MILD MODE ON XY
G0X63.7738 Y160.6052
F10
G01 Z0
X70.9693 Y140.2459 F15
X92.5556 Y139.694
X75.4163 Y126.5593
X81.5619 Y105.8589
X63.7738 Y118.1006
X45.9857 Y105.8589
X52.1313 Y126.5593
X34.992 Y139.694
X56.5783 Y140.2459
X63.7738 Y160.6052
G49 // DISABLED MILD MODE ON ALL AXES
G0 Z3
G0  X56.5783 Y140.2459
```



25.16 PARALLEL TASKS

IsoUs can execute two Gcode parallel Tasks besides the main Gcode
This allows to execute functions in asynchronous mode to increase the machine performances.
The parallel tasks **TASK1** and **TASK2** can't manage all main gcode functions (see single instructions)

25.16.1 USTASK-ENDUSTASK

These two instructions define the Gcode that is load in the TASK1 automatically from the compilator.
USTASK and ENDUSTASK MUST BE Write in the Main Gcode

```
USTASK
LOOP 1000
        $[O1]=1
        G4F1
        $[O1]=0
        G4F1
END_LOOP
ENDUSTASK
TASK.RUN 1 // FOM MAIN PROCESS RUN TASK1 CODE
LOOP 1000
        G1X100Y100F5
        G4F1
        X0Y0
        G4F1
END_LOOP
```

The gcode enclosed in USTASK ENDUSTASK will be load in the TASK1 but it is not execute during the RUN the Main Gcode. For execute it must be used TASK.RUN.
The Gcode external to USTASK ENDUSTASK will be execute in the Main Process during the RUN

25.16.2 TASK.RUN

Starts the execution the code in TASK
Syntax

TASK.RUN Ntask
Ntask TASK number where:
- **0** Task Main (not used)
- **1** TASK1
- **2** TASK2

One task can't run itself

25.16.3 TASK.STOP

Stop the execution the code in TASK
Syntax

TASK.STOP Ntask
Ntask TASK number where:
- **0** Task Main (not used)
- **1** TASK1
- **2** TASK2

One task can't stop itself

25.16.4 TASK.PAUSE

Pause the execution the code in TASK

Syntax

TASK.PAUSE Ntask

Ntask TASK number where:

- **0** Task Main (not used)
- **1** TASK1
- **2** TASK2

One task can't pause itself

25.16.5 TASK.READVAR

Read a Variable by Address in the Task

Syntax

TASK.READVAR Ntask AddrVar $VarDest

Ntask TASK number where:

- **0** Task Main
- **1** TASK1
- **2** TASK2

AddrVar Variable Address to Read

$VarDest Destination Variable

Ex:

TASK.READVAR 1 2000 $VAR // READ THE VARIABLE 2000 FROM TASK 1 AND STOR IN VAR

25.16.6 TASK.WRITEVAR

Write a Variable by Address in the Task

Syntax

TASK.WRITEVAR Ntask AddrVar $VarSource

Ntask TASK number where:

- **0** Task Main
- **1** TASK1
- **2** TASK2

AddrVar Variable Address to Write

$VarSource Source Variable

Ex:

TASK.WRITE 1 2000 $VAR // WRITE $VAR IN THE VARIABLE 2000 OF TASK 1

25.16.7 TASK.STATUS

Read the Task status

Syntax

TASK.STATUS Ntask $VAR

Ntask TASK number where:

- **0** Task Main (not used)
- **1** TASK1
- **2** TASK2

$VAR Status Destination Variable

- 0 - Task Stop
- 1 – Task Run
- 2 – Task Pause

Ex:

TASK.STATUS 1 $VAR // READ TASK 1 STATUS IN $VAR

25.16.8 TASK.LOADCMD

Load a CMD FILE in the Task
The CMD file Must be created by PlugIn MHM
The CMD isn't executed but is necessary to use TASK.RUN

Syntax

TASK.LOADCMD Ntask "CMDNAME"
Ntask TASK number where:
0 Task Main (not used)
1 TASK1
2 TASK2
"CMDNAME" CMD Name (in quotes and without extension) to load in the TASK
Ex:
TASK.LOADCMD 1 "TESTCMD" // LOAD TH CMD TESTCMD IN TASK1

25.16.9 TASK.PRIORITY

Set Task Priority

Syntax

TASK.PRIORITY Ntask Pval
Ntask TASK number where:
0 Task Main
1 TASK1
2 TASK2
Pval Priority from 0 to 100
0 High Priority (default)
Ex:
TASK. PRIORITY 1 2 // TASK 1 PRIORITY 2

25.17 MAIN MACHINE PARAMETERS

In this chapter are explained the MAIN machine parameters to be set in Isous to configure the machine.
MAIN MACHINE PARAMETERS are always present on any application, however it's possible that applications share more PARAMETERS. Indeed Isous allows to customize parameters to ad just Isous for any type of machine.
NOTE TO CALCULATE ACCELERATION IN Mt/sec²
In Isous acceleration is expressed in increase for CN sample, to obtain the value in Mt/sec² this is the calculation to do:
ACC = ACC_xx / TAU²
TAU = Sample time set on CN in msec
ACC_xx = value of acceleration parameter
If for example we set an ACC_xx=100 and a sample of TAU=5 msec, we obtain:
ACC=100/(5*5)=4 Mt/sec²

25.17.1 Generic parameters

25.17.1.1 FEEDMAX

It defines the maximum **FEED** of the system. The unit of measure depends by system resolution (typically mm/min). If it's set a **F** at speed greater than this, it is limited at **FEEDMAX**.
This parameter is affected by unit of measure set. Working with resolution of 0.0001 (instead of typycal 0.001) this parameter must be expressed in tenth of mm/min.

25.17.1.2 FEEDMIN

It defines the minimum **FEED** of the system. The unit of measure depends by system resolution (typically mm/min). If it's set a **F** at speed less than this, it is limited at **FEEDMIN**.
This parameter is affected by unit of measure set. Working with resolution of 0.0001 (instead of typycal 0.001) this parameter must be expressed in tenth of mm/min.

25.17.1.3 FEEDDEF

It defines the default **FEED** set by the system at start-up. The unit of measure depends by system resolution (typically mm/min).
This parameter is affected by unit of measure set. Working with resolution of 0.0001 (instead of typycal 0.001) this parameter must be expressed in tenth of mm/min.

25.17.1.4 FEEDRES

It defines the resolution of the speed set by F.

1	**mm/min**	**ex: F1500 = 1,5 Mt/min**
1000	**mt/min**	**ex: F1.5 = 1.5 Mt/min**

25.17.1.5 SPEEDMAX

It defines the maximum speed (**S)** of the system. The unit of measure depends by system resolution. If it's set a **S** at speed greater than this, it is limited at **SPEEDMAX**.

25.17.1.6 SPEEDMIN

It defines the minimum speed (**S)** of the system. The unit of measure depends by system resolution. If it's set a **S** at speed less than this, it is limited at **SPEEDMIN**.

25.17.1.7 SPEEDDEF

It defines the default speed (**S)** set by the system at start-up. The unit of measure depends by system resolution.

25.17.1.8 VMAXG0

It defines the maximum speed for the positioning with G0. It is in percent of the speed calculated by system based to maximum speed of each axis.
Value can be from 0 to 100 %.
This parameter is affected by unit of measure set. Working with resolution of 0.0001 (instead of typycal 0.001) this parameter must be expressed in tenth of mm/min.

25.17.1.9 ACC_G0

It defines the acceleration for positioning with G0. See the above note to detail for unit of measure.

25.17.1.10 ACC_G0.1_

It defines the private acceleration for positioning with G0.1

25.17.1.11 ACC_MAX_

It defines the max acceleration for single used in the function G66 X-100 for Adaptive Feed Control.

25.17.1.12 ACC_LAV

It defines the acceleration for interpolation with G1, G2 and G3. See the above note to detail for unit of measure.

25.17.1.13 ACC_QSTOP

It defines the acceleration for quick-stop when the position limit is reached. See the above note to detail for unit of measure.

25.17.1.14 VEL_GO_LINE_RETRACE

It defines the speed for positioning "GO LINE" during **RETRACE** function. The unit of measure depends by system resolution (typically mm/min).
This parameter is affected by unit of measure set. Working with resolution of 0.0001 (instead of typycal 0.001) this parameter must be expressed in tenth of mm/min.

25.17.1.15 ACC_RAGGIO_MAX

It defines the centrifuge acceleration to auto-reducing speed on circular interpolation. It must be set after machine test. Normally this parameter is set with values from 1 to 15
Small values defines a greater feed reduction when the radius are small. If the minimum value 1 is not sufficient to reduction your desired feed, is necessary use the decimal part in the following mode
Insert the parameter ACC_RAGGIO_MAX with value greater to 100 or 1000, the effective value is:
ACC_RAGGIO_MAX=(ACC_RAGGIO_MAX-100)/10 *(for values over 100)*
ACC_RAGGIO_MAX=(ACC_RAGGIO_MAX-1000)/100 *(for values over 1000)*

So:
109 = 0,9
1018 = 0,18
Etc.

25.17.1.16 RED_ACC_RADIUS

If it is >0 allows to have a further reduction of **ACC_RAGGIO_MAX** parameter. **RED_ACC_RADIUS** indicates the minimum radius in micron below this value **ACC_RAGGIO_MAX** is reduced in proportional mode
ACC_RAGGIO_MAX =***(RADIUS /RED_ACC_RADIUS)*ACC_RAGGIO_MAX***
Ex:
RED_ACC_RADIUS=3000
ACC_RAGGIO_MAX=5
Radius =2000 (<3000)
ACC_RAGGIO_MAX =***(2000 /3000)*5=3.33***

25.17.1.17 MIN_ACC_RADIUS

Indicate the minimum value of reduction for **ACC_RAGGIO_MAX** by **RED_ACC_RADIUS**.
RED_ACC_RADIUS =(RED_ACC_RADIUS -*100)/10* *(for values over 100)*
RED_ACC_RADIUS =(RED_ACC_RADIUS -*1000)/100* *(for values over 1000)*
Value=0 disable

25.17.1.18 VRIPOSO

It defines the speed for re-positioning in start after PAUSE. The unit of measure depends by system resolution (typically mm/min). This speed acts only if axis are moved in PAUSE state and they aren't repositioned in the exact point when restarted.

25.17.1.19 SGLP

Threshold in tenth of degree (default 200 = 20 degree) to stop axis on edge when CN works in FAST INTERPOLATION. Threshold acts only on work plane axis. CN stop automatically the axis when on work plane it recognizes an angle greater than SGLP (edge found).

25.17.1.20 SGLP_RED

Percentage of SGLP to begin the gradual slowdown in the proportion of the edge. If SGLP_RED = 0 parameter is disabled and is done only stop on the corner SGLP, otherwise the percentage set, the CNC starts slowing down (see examples below). When SGLP_RED is set to a value greater than 0, it would be necessary to increase the value of SGLP.

25.17.1.21 MAX_RED

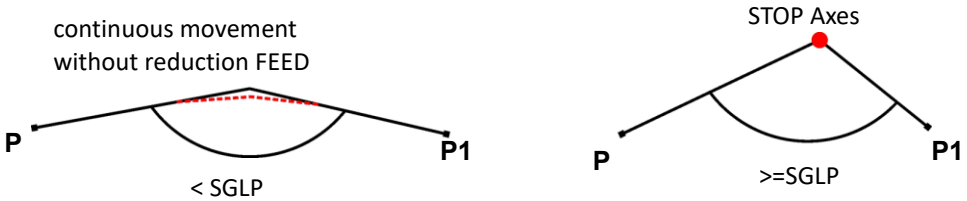
Maximum percentage reduction in speed caused by the parameter SGLP_RED. This is always limited to a value below 100% (to avoid that the speed goes to ZERO)
(see examples below)

Example Parameter SGLP_RED and MAX_RED

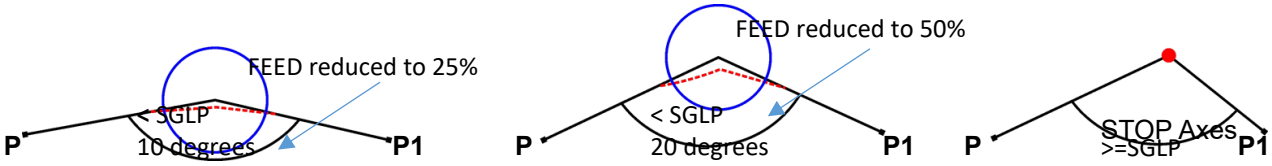
If **SGLP_RED** is disabled (value set to ZERO) work only parameter SGLP or SGL3D in the following mode:

Corner less to **SGLP** or **SGL3D** the CNC not commands the stop axes
Corner greater to **SGLP** or **SGL3D** the CNC decelerates and commands the STOP Axes

Ex: SGLP=200 (20 degrees) , SGLP_RED=0



Ex: SGLP=450 (45 degrees) , SGLP_RED=10 (10% of 45 degrees – at 4,5 degrees start the slowdown)
MAX_RED=85



25.17.1.22 SGLR

Threshold to signal an error for circular interpolation.
In circular interpolation CN computer radium in start point and in end point, if the difference is greater than SGLR program is stopped showing an error. That happens when arc is made by center positions **I,J** with value not accurate.
The unit of measure depends by system resolution (typically 0.001 mm, default 10 thousandths). Ex:

Unit of measure 0.001 mm ***10 = 10 thousandths of mm***
Unit of measure 0.0001 mm ***100= 10 thousandths of mm***

25.17.1.23 ACQ_MODE

It define the acquisition mode for searching of the **SENSOR** with **G102** function and for **StartAcqSens** method of Framework. Actually not in use, reserved for future expansion.

25.17.1.24 ACQ_VEL

It defines the axis speed for acquisition sensor cycle.
Contrary to other speed parameters this is NOT affected by unit of measure and must be expressed always in mm/min

25.17.1.25 RESQUOTE

Resolution for axis position. It depends by DSOF-axis parameter.
It defines the minimum unit for axis position:

10 → ***Tenth of mm***
100 → ***Hundredths of mm***
1000 → ***Thousandths of mm***
10000 → ***0.1 Thousandths of mm***
100000 → ***0.01 Thousandths of mm***

25.17.1.26 VISUAREAL

Enable or disable showing of axis ACTUAL positions
-1 = Disable reading of actual position
0 = Enable by INTERFACE
1 = Enable reading of ACTUAL positions
2 = Enable reading of ACTUAL FOLLOWING ERROR position

25.17.1.27 ENABLE_OW_G0

Enable or disable override potentiometer on G0
NO = Potentiometer disabled
YES = Potentiometer enabled
NO G0 ACK = Potentiometer disabled with Acknowledged on CN
YES G0 ACK = Potentiometer enabled with Acknowledged on CN

25.17.2 Axis parameters

25.17.2.1 VMAX_

It defines the maximum speed of the single axis for positioning with G0. The unit of measure depends by system resolution (typically mm/min).
This parameter is affected by unit of measure set. Working with resolution of 0.0001 (instead of typycal 0.001) this parameter must be expressed in tenth of mm/min.

25.17.2.2 VOVERLIMIT_

Defines the speed limit for a single Axis used in G66 X-100 during the check speed in RTCP Axes or tangential Axes. The unit is like to VMAX.
If one or more Axes exceed the value in VOVERLIMIT the FEED is decrease

25.17.2.3 VMAX_Pn

It defines the maximum speed of the positioner number

25.17.2.4 VJOG_

It defines the speed of the single axis manual positioning (jogging). The unit of measure depends by system resolution (typically mm/min).
Contrary to other speed parameters this is NOT affected by unit of measure and must be expressed always in mm/min

25.17.2.5 ACC_JOG_

Acceleration for JOGGING positioning. See the above note to detail for unit of measure.

25.17.2.6 RZERO_MODE

This parameter defines the mode of homing procedure. These are the value accepted:

RZERO_MODE=0 – Backward (negative), sensor ACTIVATED

1) Fast backward to zero-sensor with **F=RZERO_VEL**
2) Slow forward to exit from zero-sensor with **F=RZERO_VELF**
3) Slow backward to zero-sensor with **F=RZERO_VELF**
4) Move axis to offset position (**RZERO_OFFSET**) with **F=RZERO_VEL**
5) Preset position at **RZERO_PRESET**

RZERO_MODE=1 - Forward (positive), sensor ACTIVATED

1) Fast forward to zero-sensor with **F=RZERO_VEL**
2) Slow backward to exit from zero-sensor with **F=RZERO_VELF**
3) Slow forward to zero-sensor with **F=RZERO_VELF**
4) Move axis to offset position (**RZERO_OFFSET**) with **F=RZERO_VEL**
5) Preset position at **RZERO_PRESET**

RZERO_MODE=2 - *Backward (negative), sensor ACTIVATED with* encoder INDEX

1) Fast backward to zero-sensor with **F=RZERO_VEL**
2) Slow forward to exit from zero-sensor with **F=RZERO_VELF**
3) Slow backward to zero-sensor with **F=RZERO_VELF**
4) Continue to first encoder INDEX always with **F=RZERO_VELF**
5) Move axis to offset position (**RZERO_OFFSET**) with **F=RZERO_VEL**
6) Preset position at **RZERO_PRESET**

RZERO_MODE=3 - *Forward (positive), sensor ACTIVATED with* encoder INDEX

1) Fast forward to zero-sensor with **F=RZERO_VEL**
2) Slow backward to exit from zero-sensor with **F=RZERO_VELF**
3) Slow forward to zero-sensor with **F=RZERO_VELF**
4) Continue to first encoder INDEX always with **F=RZERO_VELF**
5) Move axis to offset position (**RZERO_OFFSET**) with **F=RZERO_VEL**
6) Preset position at **RZERO_PRESET**

RZERO_MODE=4 - *Backward (negative), sensor DISACTIVATED*

1) Fast backward to zero-sensor with **F=RZERO_VEL**
2) Slow forward to exit from zero-sensor with **F=RZERO_VELF**
3) Move axis to offset position (**RZERO_OFFSET**) with **F=RZERO_VEL**

4) Preset position at **RZERO_PRESET**

RZERO_MODE=5 - *Forward (positive), sensor DISACTIVATED*

1) Fast forward to zero-sensor with **F=RZERO_VEL**
2) Slow backward to exit from zero-sensor with **F=RZERO_VELF**
3) Move axis to offset position (**RZERO_OFFSET**) with **F=RZERO_VEL**
4) Preset position at **RZERO_PRESET**

RZERO_MODE=6 - *Backward (negative), sensor DISACTIVATED with* encoder INDEX

1) Fast backward to zero-sensor with **F=RZERO_VEL**
2) Slow forward to exit from zero-sensor with **F=RZERO_VELF**
3) Continue to first encoder INDEX always with **F=RZERO_VELF**
4) Move axis to offset position (**RZERO_OFFSET**) with **F=RZERO_VEL**
5) Preset position at **RZERO_PRESET**

RZERO_MODE=7 - *Forward (positive), sensor DISACTIVATED with* encoder INDEX

1) Fast forward to zero-sensor with **F=RZERO_VEL**
2) Slow backward to exit from zero-sensor with **F=RZERO_VELF**
3) Continue to first encoder INDEX always with **F=RZERO_VELF**
4) Move axis to offset position (**RZERO_OFFSET**) with **F=RZERO_VEL**
5) Preset position at **RZERO_PRESET**

RZERO_MODE=8 – Backward to encoder INDEX (no sensor)

1) Slow backward to first encoder INDEX with **F=RZERO_VELF**
2) Move axis to offset position (**RZERO_OFFSET**) with **F=RZERO_VEL**
3) Preset position at **RZERO_PRESET**

RZERO_MODE=9 – Forward to encoder INDEX (no sensor)

1) Slow forward to first encoder INDEX with **F=RZERO_VELF**
2) Move axis to offset position (**RZERO_OFFSET**) with **F=RZERO_VEL**
3) Preset position at **RZERO_PRESET**

RZERO_MODE=32 – No searching, HOME position on ENABLE DRIVE

RZERO_MODE=64 – No searching, position from ABSOLUTE ENCODER

RZERO_MODE=128+HOME_MODEx256 – CanOpen DS402 mode of HOMING procedure

25.17.2.7 LIMITE_N_

Primary Negative limit of axis position **SOFTWARE-LIMIT**. The unit of measure depends by system resolution (typically 0.001 mm, for ex. -10000 = -10mm).

25.17.2.8 LIMITE_P_

Primary Positive limit of axis position **SOFTWARE-LIMIT**. The unit of measure depends by system resolution (typically 0.001 mm, for ex. 2000000 = 2000mm).

25.17.2.9 2ND_LIMITE_N_

Secondary Negative limit of axis position **SOFTWARE-LIMIT**. The unit of measure depends by system resolution (typically 0.001 mm, for ex. -10000 = -10mm).

25.17.2.10 2ND_LIMITE_P_

Secondary Positive limit of axis position **SOFTWARE-LIMIT**. The unit of measure depends by system resolution (typically 0.001 mm, for ex. 2000000 = 2000mm).

25.17.2.11 LIMITE_N_Pn

Negative limit positioner

25.17.2.12 LIMITE_P_Pn

Positive limit of positioner

25.17.2.13 DSOFV

Software divider for the position of electronic manual hand-wheel. It reduces or increases its resolution. A negative value reverse the direction.
It is strictly related to VTB application, for ex:

Encoder Pulse = 100 pulse/rev.
SCALA=1 (VTB application parameter)
FILTRO=10 (VTB application parameter)
DSOFV=10000 (pulse x filtro x **10 thousandths**)
Working with default resolution of 0.001mm we obtain a hundreds for each encoder pulse.

25.17.2.14 RZERO_OFFSET

Value of axis positioning after homing procedure (see RZERO_MODE parameter for details).
The unit of measure depends by system resolution (typically um).

25.17.2.15 RZERO_PRESET

Value of preset position after homing procedure (see RZERO_MODE parameter for details).
The unit of measure depends by system resolution (typically um).

25.17.2.16 RZERO_VEL

Fast speed for homing procedure (see RZERO_MODE parameter for details).
The unit of measure depends by system resolution (typically mm/min).

25.17.2.17 RZERO_VELF

Slow speed for homing procedure (see RZERO_MODE parameter for details).
The unit of measure depends by system resolution (typically mm/min).

25.17.2.18 RZERO_ACC

Acceleration for all movement of homing procedure. See the above note to detail for unit of measure.

25.17.2.19 MSOF_

It defines the number of pulse of one shaft revolution. If incremental encoder are connected to CN the pulse number must be multiplied for 4 to adjust hardware decoder. It works strictly related with DSOF, indeed the ratio MSOF/DSOF is used to transform unit from pulse to millimetres.

25.17.2.20 DSOF_

It defines the envelope one shaft revolution. It works strictly related with DSOF, indeed the ratio MSOF/DSOF is used to transform unit from pulse to millimetres.
This value determines the unit of measure of the system. Typically it's put a value in thousandths of mm. Accordingly it must set **RESQUOTE** parameter properly.

10 → ***Tenth of mm***
100 → ***Hundredths of mm***
1000 → ***Thousandths of mm***
10000 → ***0.1 Thousandths of mm***
100000 → ***0.01 Thousandths of mm***

25.17.2.21 GANTRY

It defines the index to which axis get the position value. This function can be used to enable GANTRY function. This is the relationship from index and axis name:

0 → ***AXIS X***
1 → ***AXIS Y***
2 → ***AXIS Z***
3 → ***AXIS A***
4 → ***AXIS B***
5 → ***AXIS C***
6 → ***AXIS U***
7 → ***AXIS V***
8 → ***AXIS W***

To enable GANTRY function:

GANTRY_X → 0=disabled or index of axis master
GANTRY_Y → 1=disabled or index of axis master
GANTRY_Z → 2=disabled or index of axis master
GANTRY_A → 3=disabled or index of axis master
GANTRY_B → 4=disabled or index of axis master
GANTRY_C → 5=disabled or index of axis master
GANTRY_U → 6=disabled or index of axis master
GANTRY_V → 7=disabled or index of axis master
GANTRY_W → 8=disabled or index of axis master

ATTENTION!!!
To enable GANTRY function it must be supported by VTB application on CN.

25.17.2.22 SGL3D

It defines the threshold to detection of **EDGE** in 3D interpolations. This is used by **G65 G67** or by **AFC** filter **G66**. The value is approximatively expressed in tenth of degree. The proper value must be found from dynamics of the machine. In **G65** or **G67** interpolation when an axis overcomes this threshold axis are stopped.

Reference value for SGL3D

THRESHOLD in DEGREES	VALUE OF SGL3D (min-max)
5	60-90
10	125-175
20	250-350
30	300-500
45	400-700

25.17.2.23 SGLAFC

It defines the threshold for **AFC G66** to automatic reduction of the speed. This value determines the maximum speed change of each axis tolerates. When in the path are found values higher than SGLAFC speed is reduced. The value depends by dynamic of machine.

25.17.2.24 MILD

It Defines the MILD MODE value (see G49 functions)
Estimated values from 200 to 20
0 or 1000 DISABLED

25.17.2.25 SGLP_MILD

It Defines the threshold 2D **ABOVE THIS VALUE THE MILD MODE** Doesn't Works

25.17.2.26 SGL3D_MILD_

It Defines the threshold 3D **ABOVE THIS VALUE THE MILD MODE** Doesn't Works
(see SGL3D for degrees references)

25.17.2.27 BACKLASH

Parameter for BACKLASH compensation. The unit of measure depends by system resolution (typically 0.001 mm, for ex. -10 = 0.01mm).

25.17.2.28 TBCK

It define a time to achieve the BACKLASH recovery. If Se TBCK>=BACKLASH it is recovered in a single CN sample, else it is divided in more samples The number of samples will be: **BACKLASH / TBCK**
With the result rounded-up.
Ex:

BACKLASH=100
TBCK=80
N.SAMPLES=1.25 → 2 samples

Sample is defined in the VTB application on CN (typically 2 msec).
It's recommended to distribute the BACKLASH recovery in more sample when it is high (over 3-4 hundredths).

25.17.2.29 TSHF

It define a time to achieve the SetShiftAxis recovery.
This value indicate the increment xTAU
Ex:

TAU=2 Ms
ValShift=100
TSHF_=10

The entire position is reached in 20 Ms

25.17.2.30 WR_SPD9

It define automatic write SPEED "S" to variable USER GENERIC 10 to NC.
WR_SPD=0 Automatic write disabled
WR_SPD=1 Automatic write Enabled

If WR_SPD=1 when the S value of PartProgram is automatically writing to NC Variable User Generic 10

25.17.2.31 RFG

It define automatic limitation FEED in G1-G2-G3
RFG=0 Whitout limitation
RFG=1 If the velocity exceeds that of the single axis (VMAX_), it is limited

25.17.2.32 JERK

It define the value of JERK for S Ramp.
If the value is ZERO the S ramp is disabled (use trapezoidal ramp)
Value > 0 enable S ramp.
Max value 100

25.17.2.33 **CR_LIMIT**

Enables or disables the prior control of axis limits
If enabled, each START, PART PROGRAM is not performed if the axes exceed the limits set machine
CR_LIMIT=0 Disable
CR_LIMIT=1 Preventive control G0-G1 activated only on the final point of G2-G3
CR_LIMIT=2 Preventive control activated G0-G1-G2-G3 (with explosion of ARCS)
CR_LIMIT=3 Preventive control activated G0-G1-G2-G3 only in run time, before executed the block (G2-G3 Only final point is controlled)

25.17.2.34 **ARC_REL**

Enables or disables the center of arc I,J in relative mode
ARC_REL=0 The center I,J is in absolute mode (if set G90)
ARC_REL=1 The center I,J is in relative mode (if set G90)

25.17.2.35 **NO_SHORT**

Enables or disables the CNC to remove the short segments
NO_SHORT=0 Function disactivated
NO_SHORT=1 short segment are removed by CNC. This facilitates the machining of complex contours
NO_SHORT=2 Only Warning message is displayed if the short segment is worked

25.17.2.36 **USE_G60**

If the value is >0, enabled defaut mode G60 and not G61

25.17.2.37 **TIME_OUT_CMD**

TimeOut Commands invoked on CNC. Value in Ms (default 5000)

25.17.2.38 **TIME_OUT_M**

TimeOut **M** invoked on CNC.. Value in Ms (default 5000)

25.17.2.39 **G62_TO_G40**

If it is set to 1 (YES) will be add a **G62** After **G40** (end tool offset).
This allows to empty the movements buffer

25.17.2.40 **LIMIT_G41G42**

If it is set to 1 (YES) the previous limits, or simulation limits will be check only in the **G41** or **G42** phase, and not in **G40** phase (center of tool).
WARNING
In this case, could be that the center of tools is inside the machine limits, and the external of tools is outside of machine limits

25.17.2.41 **INVERT_G2G3**

If it is set to 1 (YES) the **G2** (CW circular interpolation) is inverted with **G3** (CCW circular interpolation) when is set the Work Plane **G18 XZ**

25.17.2.42 **STANDARD_IJ**

If it is set to 1 (YES) uses the standard convention for parameters I,J,K

25.17.2.43 **ACC_VMODE_X,Y,Z,A,B,C,U,V,W**

Acceleration for the axis in SPEED MODE, G108.4 Function

25.17.2.44 **ENABLE_RFEED**

If ON (1) enables the events generation and display of Real Axes Feed
Available on **ComSynk** 3.0.0.500 or greater and **IsoVirtual** 2.5.0 or greater

25.17.2.45 **NEW_ORIGINS**

If ON (1) enables the Origins file management. The file "Zeri.val" will be renamed **"Origins_n.val"** where **n** is the IsoUs process.
Now all Origins are separated for each process

25.17.2.46 **USE_G80_CYCLES**

If ON (1) enables the migrations of codes:

G1080 → G80
G1081 → G81
G1082 → G82
G1083 → G83
G1084 → G84
G1028 → G28

And vice versa

25.17.2.47 **G28_HOME_**

Defines the Home position for single Axis for the **G1028 (G28)** function
The unit of measure depends by system resolution (typically um).

25.17.2.48 **DIVACC**

Defines the acceleration divisor for interpolator MSL.
Typical value 10, higher values define lower accelerations

25.17.2.49 **AXES_Z**

Defines **AXIS Z** index (depth) for canned cycles G81,82,83,84

25.17.2.50 **AXES_G84**

Defines **AXIS Tapping** index for canned cycle G84
Only for **INTERPOLATE MODE**

25.17.2.51 **MODE_G84**

G84 Mode
NORMAL (0) for **NON INTERPOILATE AXIS** (normal Spindle)
INTERPOLATE (1) for Interpolate Axis Spindle
The Tapping Step is calculated from **S** and **F** set

25.17.2.52 **ROT_G84**

Rotation direction for INTERPOLATE Tapping Axis
POSITIVE (0)
NEGATIVE (1)

25.17.2.53 **MOLT_G84**

Feed multiplier for tapping Axis G84

25.17.2.54 **DELTA_G83**

D parameter for function G1083 (G83)

25.17.2.55 **PRIORITY_TASK1 – PRIORITY_TASK2**

Set the priority for the TASK1 and TASK2. That indicate the interval time for execution task.
Value=0 (default) Max priority
Set value from 0 to 100

25.17.2.56 **USE_J_M6**

If Yes (1) during M6 cycle is disabled PAUSE and STOP

25.17.2.57 SH_FILTER_X,Y,Z,A,B,C,U,V,W

Filter for Shift Axes
0 Disabled
Less value more filtering
From 0 to 200

25.18 PID PARAMETERS

The PID Parameters are used for the analogical Axes velocity control with Encoder Loop.
Usually, these Axes are controlled by +/- 10 Analog Outputs.
These Parameters are inserted by Isous configurator "Machine Parameters→Default Par PID":

25.18.1.1 PID_KP

Proportional Gain

25.18.1.2 PID_KI

Integrative Gain

25.18.1.3 PID_KV

Feed Forward Gain.
For a correct setting You have to use the following method:

1) Set to ZERO (0) the PID_KP and PID_KI
2) Insert a PID_KV value
3) Move the axes and check if the real position is near to target (+/- 5 %)
4) Insert the PID_KP and PID_KI

WARNING
This operation must be did with Axis disconnected from mechanical part

25.18.1.4 PID_I_LIMIT

Integrative Limit

25.18.1.5 PID_DIV

PID parameters Magnitude. Increase this parameter for Increase the PID parameter resolution:
Ex: PID_DIV=10 PID_KP=1
It Is the same
PID_DIV=100 PID_KP=10

25.18.1.6 PID_SERVO

Maximum following error (micron).
If the Axis exceeds this value, an alarm is invoked
(must be exceeds this value for a PID_TIME_SERVO time)

25.18.1.7 PID_TIME_SERVO

If the Axis exceeds the PID_SERVO for a time lower of PID_TIME_SERVO (Millisecond)
The ERROR isn't invoked

25.18.1.8 PID_DIR

Analog Outputs Direction (0 or 1)
Set this parameter for tuning the Axis direction with Analog Output
WARNING
For change the Axis direction (ex: CW in CCW) is necessary Change this Parameter and also change the SIGN of MSOF parameter (ex: 5000 to -5000)

25.18.1.9 PID_OFFS_ANA

Reset the analog output Offset.
With all PID parameters (KP,KI) set to ZERO, set this parameter until the axis is stationary (when it is enabled)

25.19 SPINDLE PARAMETERS

Parameters for SPINDLE

25.19.1.1 SPEEDMAXSPINDLE

Max rpm for SPINDLE when the analog output is 10V

25.19.1.2 ANALOG_BIT_RES

Analog resolution for the set Channel (see Chapr. 12.8)

25.19.1.3 SPEED_ANALOG_CH

Analog Channel for SPINDLE (see Chapr. 12.8)

25.19.1.4 ENABLE_OE_SPEED

Manage the override for spindle

DISABLE	No Override
EXTERNAL	External Override managed by VTB application
INTERNAL VIRTUAL	Enable the virtual override by PlugIn **USSPINDLEMANAGER (IsoUs)**

25.19.1.5 SPEED_OW_MIN

Minimum value in percentage for override refer to current SPEED set (0-100%)

25.19.1.6 SPEED_OW_MAX

Maximum value in percentage for override refer to current SPEED set (0-100%)

25.20 EGEAR PARAMETERS

25.20.1.1 EGEAR_KEM

Multiplier for adapt feed **MASTER** - **SLAVE**

25.20.1.2 EGEAR_KED

Divisor for adapt feed **MASTER** – **SLAVE**

25.20.1.3 EGEAR_ACC

Acceleration of **LINK MASTER – SLAVE** (um/TAU)

Example

Axis **MASTER CONVEYOR** controlled by motor with **ENCODER 1000** p/r (the hardware will multiply the pulses x 4), axis **X SLAVE** brushless interpolated motor.
Space **MASTER** each encoder round = 100 mm.
Link **SLAVE** in 500 msec (0,5 sec)
TAU=2 Msec

EGEAR_KEM =100000 um
EGEAR_KED = 4000 i/g
EGEAR_ACC = 100000/(500/TAU) = 100000/250=400

25.21 Peripheral speed parameters

25.21.1.1 CENTRE_X_R1
X axis position of rotation centre for first rotational axis [um]

25.21.1.2 CENTRE_Y_R1
Y axis position of rotation centre for first rotational axis [um]

25.21.1.3 CENTRE_Z_R1
Z axis position of rotation centre for first rotational axis [um]

25.21.1.4 CENTRE_X_R2
X axis position of rotation centre for second rotational axis [um]

25.21.1.5 CENTRE_Z_R2
Y axis position of rotation centre for second rotational axis [um]

25.21.1.6 CENTRE_Z_R2
Z axis position of rotation centre for second rotational axis [um]

For any rotational axis, only two positions are mandatory to define the rotation centre.
Referring to the example in the G108.12 instructions (see Cap.25.7.5) considering that we have Y as rotation axis, only X and Z positions are needed to define it.

Index

2.1 BLOCK 3
2.2 LINE NUMBER OR BLOCK NUMBER 3
2.3 PROGRAM RUNNING 3
2.4 AXIS NAME 3
2.5 HOMING 3
2.6 WORK ORIGIN 3
2.7 WORK OFFSET 4
2.8 HEADS 4
2.9 MODAL FUNCTIONS 4
2.10 RECOGNIZED CODES 4
2.11 NUMERIC VALUES SETTING 4
2.12 PROGRAM REMARKS 4
3.1 RECOGNIZED G CODES 5
4 OTHERS RECOGNIZED CODES 8
4.1 PROGRAM FLOW INSTRUCTIONS 10
4.2 MULTI TASK INSTRUCTIONS 10
4.3 GENERIC INSTRUCTIONS 11
4.4 MATH AND LOGICAL OPERATORS 12
4.5 MATH FUNCTIONS 13
4.6 VARIABLES AND CONSTANTS 14
4.7 PREDEFINED VARIABLES 14
4.8 USFORMS INSTRUCTIONS 15
4.9 EXTENDED INSTRUCTIONS EXD 16
4.10 COMPILATOR SWITCH AND DIRECTIVE 17
4.11 INSTRUCTIONS FOR REMOTE CONTROLS 18
4.12 INSTRUCTIONS FOR MULTI PROCESS CONTROL 19
5.1 IF-ELSE-END_IF 20
5.2 LOOP - END_LOOP 20
FOR – NEXT – BREAK - CONTINUE 21
WHILE – END_WHILE – BREAK - CONTINUE 22
5.3 GOTO 23
5.4 GOSUB – RETURN 23
5.5 LABEL 24
5.6 END_PROGRAM 24
5.7 WAIT_INPUT 24

5.8 TWAIT_INPUT 25
5.9 ERROR....26
5.10 RESUME_T 26
5.11 SAVE_T....26
5.12 SET_TABPAR....26
5.13 SELECT – CASE - END_SELECT....27
5.14 IF RUN....28
5.15 IF NOTRUN....28
5.16 USEFORM....28
6.1 SDO_DL....29
6.2 SDO_UL....29
6.3 GET 29
6.4 READ_PARMAC 30
6.5 WRITE_PARMAC 30
6.6 OPT....30
6.7 PAUSE_MODE 31
6.8 FILTER_MODE....31
6.9 IMPORT....31
6.10 END_IMPORT....32
6.11 STOP_MODE 33
6.12 DEBUG_INFO....33
7.1 LIB.HMESSAGE 34
8.1 EXD.STL_LOAD 35
8.2 EXD.READ_TOOLPAR....39
8.3 EXD.READ_HEADPAR....39
8.4 EXD.WRITE_TOOLPAR 39
8.5 EXD.WRITE_HEADPAR 40
8.6 EXD.LOAD_LAST 40
8.7 EXD.SAVE_LAST....40
8.8 EXD.RESET_LAST 40
8.9 EXD.SET_OUT....41
8.10 EXD.READ_OUT....41
8.11 EXD.READ_INP 41
8.12 EXD.WRITE_USER....41
8.13 EXD.READ_USER 42
8.14 EXD.READ_DEMAND....42
8.15 EXD.READ_REAL....42

8.16 EXD.MASK....43
8.17 EXD.WRITE_BIT....43
8.18 EXD.READ_BIT....43
8.19 EXD.PXV_ALL_DETECTORS....44
8.20 EXD.PXV_SINGLE_DETECTOR....44
8.21 EXD.PXV_READ_PROBE....45
8.22 EXD.PXV_RESET_PROBE....45
EXD.PXV_SET_DETECTOR....45
8.23 EXD.PXV_SAVE_IMAGE....46
8.24 EXD.PXV_GET_IMAGE....46
8.25 EXD.PXV_SET_JOB....46
8.26 EXD. RUN_SCRIPT....47
1) EXD.OPEN_DLL....50
2) EXD.CALL_DLL....52
3) EXD.CLOSE_DLL....52
4) EXD.SYNK_DLL....52
5) EXD.STL_SETVIS....53
6) EXD.STL_READVIS....53
7) EXD.STL_SETVISIDX....53
8) EXD.STL_READVISIDX....54
9.1 LIB.MESSAGE....69
9.2 LIB.SHOWFORM....70
9.3 LIB.CLOSEFORM....70
9.4 LIB.FORMPROP....70
9.5 LIB.FORMTEXT....73
9.6 LIB.ADDLABEL....73
9.7 LIB.LABELPROP....73
9.8 LIB.LABELTEXT....74
9.9 LIB.LABELPRINT....75
9.10 LIB.LABELF....75
9.11 LIB.ADDBUTTON....75
9.12 LIB.BUTTONPROP....76
9.13 LIB.BUTTONTEXT....78
9.14 LIB.BUTTONPRINT....78
9.15 LIB.BUTTONF....79
9.16 LIB.ADDINPUT....79
9.17 LIB.INPUTPROP....79

9.18 LIB.INPUTSETVALUE 80
9.19 LIB.ADDITEXT 81
9.20 LIB.ITEXTPROP 81
9.21 LIB.ITEXTSETVALUE 82
9.22 LIB.ADDCHECK 82
9.23 LIB.CHECKPROP 82
9.24 LIB.CHECKSETVALUE 83
9.25 LIB.CHECKTEXT 83
9.26 LIB.ADDCOMBO 84
9.27 LIB.COMBOPROP 84
9.28 LIB.COMBOSETVALUE 85
9.29 LIB.COMBOITEM 85
9.30 LIB.ADDSLIDER 86
9.31 LIB.SLIDERPROP 86
9.32 LIB.SLIDERSETVALUE 87
9.33 LIB.GETVAR 88
9.34 LIB.SETVAR 88
9.35 LIB.DEBUG 88
9.36 COLOR TABLE 89
10.1 IFDEF 91
10.2 ELSEDEF 91
10.3 ENDIFDEF 91
10.4 NOAXESREADY 92
10.5 ONERROR 92
10.6 ONSTOP 92
10.7 ENDON 92
10.8 USET 92
10.9 USEH 92
11.1 SIN 93
11.2 COS 93
11.3 LOG 93
11.4 EXP 93
11.5 INT 93
11.6 FIX 94
11.7 ABS 94
11.8 DRG 94
11.9 RAD 94

11.10 SQR 94
11.11 TAN....94
11.12 ATAN....95
11.13 ASIN 95
11.14 ACOS 95
I. ISTRUZIONI PER IL CONTROLLO MULTIPROCESSO 96
12.1 CNC.LOAD 97
12.2 CNC.RUN 97
12.3 CNC.PREVIEW 97
12.4 CNC.STOP 98
12.5 CNC.PAUSE 98
12.6 CNC.STATUS 98
12.7 CNC.STATUSBIT 99
12.8 CNC.INFO 99
12.9 CNC.AXIS 99
12.10 CNC.GROUP 100
12.11 CNC.READVARADDR 100
12.12 CNC.READVARNAME 101
12.13 CNC.WRITEVARADDR 101
12.14 CNC.WRITEVARNAME 101
12.15 CNC.READPARMAC 101
12.16 CNC.WRITEPARMAC 101
12.17 CNC.ENABLEAXIS 102
12.18 CNC.HOMEAXIS 102
12.19 CNC.READGENERIC 102
12.20 CNC.WRITEGENERIC 102
13.1 REMOTE.LOAD 104
13.2 REMOTE.RUN 104
13.3 REMOTE.STOP 104
13.4 REMOTE.PAUSE 105
13.5 REMOTE.STATUS 105
13.6 REMOTE.MOVE 105
13.7 REMOTE.INFO 106
13.8 REMOTE.AXIS 106
13.9 REMOTE.GROUP 107
13.10 REMOTE.READISOVAR 107
13.11 REMOTE.READVARNAME 107

13.12 REMOTE.WRITEISOVAR 108
13.13 REMOTE.WRITENAMEVAR 108
13.14 REMOTE.READCNVAR 108
13.15 REMOTE.WRITECNVAR 109
13.16 REMOTE.READINPUT 109
13.17 REMOTE.READOUTPUT 109
13.18 REMOTE.WRITEOUTPUT 110
15.1 ROUND BRACKETS ( ) 113
15.2 ISO EXPRESSIONS AND SQUARE BRACKETS [ ] 113
15.3 VARIABLE TEST 113
15.4 BIT CONTROL 114
16.1 NUMERIC CONSTANTS 114
16.2 GENERIC VARIABLES “$” 114
16.3 VARIABLES FOR ADDRESS 115
16.4 VARIABLES FOR POINTER 115
16.5 DATA STRUCTURES 115
16.6 AXIS POSITION VARIABLES 116
16.7 DIGITAL INPUT/OUTPUT VARIABLES 116
16.8 TIMER VARIABILES 117
16.9 TOOL TABLE VARIABLES 118
16.10 HEAD TABLE VARIABLES 119
16.11 AXIS COUNTER VARIABLES 119
17 LIMIT AXIS VARIABLES 120
18 PREVIEW PARAMETERS 121
18.1 GLOBAL VARIABLE 126
18.2 M FUNCTION VARIABLES (FOR CN) 126
18.3 SPECIAL PARAMETERS VARIABLES 126
18.4 WORK ORIGIN AND OFFSET VARIABLES 129
18.5 USER GENERIC VARIABLES 129
18.6 ARRAY VARIABLES - DIM 129
18.7 MARKER VARIABLES 130
18.8 VARIABLE FOR ANALOG SPINDLE OUTPUT 131
18.9 MACRO AND GCODE PARAMETERS MANAGEMENT 132
20 PA-PD(N,PAR)ESPR SET ABSOLUTE POSITION 133
21 PF(N)ESPR SET POSITIONER FEED 133
22 PS(N) STOP POSITIONER 133
23 _PM(N,PAR) READ STATUS POSITIONER 133

24.1 DEFINITION OF AXIS POSITIONS 135

24.1.1 Axis definition for Channel Address 135
24.1.2 G90 – PROGRAMMING WITH ABSOLUTE POSITIONS 136
24.1.3 G91 - PROGRAMMING WITH INCREMENTAL POSITIONS 136
24.1.4 Definition of absolute and incremental positions 137

24.2 WORK ORIGIN 137

24.2.1 Work origin By INDEX 137
24.2.2 G94 – Work origin at position defined with parameter 139
24.2.3 G94.n – Save work orgini at Index 140
24.2.4 G54, G55, G56, G57, G58, G59 - Work origin from memory file 140
24.2.5 G92 – Work origin in current axis position 142
24.2.6 G82 – Work origin in current axis position with sensor offset 142
24.2.7 G98 – G53 - Suspend work origin 142
24.2.8 G99 – Restore work origin 142
24.2.9 G940 MOVE axes excluding WORK ORIGIN only in the actual block 144
24.2.10 USER_ZERO – Index of WORK ORIGIN LIST 144

24.3 WORK OFFSET 145

24.3.1 G93 - Work offset at position defined with parameter 145
24.3.2 G95 - Work offset in current axis position 145
24.3.3 G85 - Work offset in current axis position with sensor offset 145
24.3.4 G86 - Hardware preset axis on module 360 degrees 146
24.3.5 G96 - Suspend work offset 146
24.3.6 G97 - Restore work origin 146
24.3.7 USER_OFFSET - Index of WORK OFFSET LIST 146

24.4 WORK HEAD SELECTION – H FUNCTION 148

24.4.1 Hn – Select head 148
24.4.2 G87 – Suspend head offset 148
24.4.3 G88 – Restore head offset 148

24.5 ROTATIVE AXIS 149

24.5.1 G36 – Rotative axis definition 149

24.6 HARDWARE PRESET OF AXIS 150

24.6.1 G89 – Hardware preset of axis position 150

24.7 HARDWARE PRESET OF AXIS 150

24.7.1 G84 – Preset Axes counters 150
24.7.2 G83 – Preset CPU 1 counters 150

24.8 CANNED CYCLES 151

24.8.1 G1080 – Disabled Canned Cycles 151
24.8.2 G1081 – Drilling Cycle 151
24.8.3 G1082 – Drilling Cycle 152
24.8.4 G1083 – Drilling Cycle 153
24.8.5 G1084 – Tapping Cycle 154

24.9 SELECT WORK PLANE 155

24.9.1 G17 – Select work plane on X-Y 155
24.9.2 G18 - Select work plane on X-Z 155
24.9.3 G18.1 - Select work plane on X-Z but not in PREVIEW 155
24.9.4 G19 - Select work plane on Y-Z 155

24.9.5 G19.1 - Select work plane on Y-Z but not in PREVIEW 155
24.9.6 G70 - Select work plane on axis by parameters 156

24.10 START SEARCH OF HOME POSITION 157

24.10.1 G71 – Start Homing cycle 157
24.10.2 G71.1 Enable Axis 157
24.10.3 G71.2 Disable Axis 157

24.11 AXIS MOVEMENT FUNCTIONS 158

24.11.1 G0 – Rapid positioning 158
24.11.2 G0.1 – Rapid positioning with private Acceleration 158
24.11.3 G0.2 – Rapid positioning for "Transported Axes" 159
24.11.4 G1 – Linear interpolation a programmed F speed 160
24.11.5 G1.1 – G1-G2-G3 Suspension and Set G0 160
24.11.6 G1.2 – Resume Gx saved with G1.1 160
24.11.7 G2/G3 - Circular interpolation a programmed F speed 161
24.11.8 G30 - Enable automatic insert of fillet on edges 162
G31 – SUSPEND OF AUTOMATIC INSERT OF FILLET 163
24.11.9 G32 - Restore of automatic insert of fillet 163
24.11.10 G33 - Enable automatic insert of bevel on edges 164
24.11.11 G34 - Suspend of automatic insert of bevel 165
24.11.12 G35- Restore of automatic insert of bevel 165
24.11.13 G102 – Start searching sensor position 166
24.11.14 G102.1 – Acquisition from PxVision System 167

24.12 PROGRAMMING AXIS INTERPOLATION SPEED 173

24.12.1 F – Speed of axis interpolation 173
24.12.2 ARC speed auto-reduction 173

24.13 WORK PLANE TRANSFORMATION 174

24.13.1 G120 – Vertical mirror 174
24.13.2 G121 – Disable vertical mirror 174
24.13.3 G24 – Horizontal mirror 175
24.13.4 G25 – Disable horizontal mirror 175
24.13.5 G22 – Echange axis of work plane 175
24.13.6 G23 – Restore axis of work plane 176
24.13.7 G26 – Exchange axis by parameter 176
24.13.8 G27 – Suspend G26 176
24.13.9 G28 – Restore G26 176
24.13.10 G1028 – Return to Home 177
24.13.11 G51 – Suspend work plane rotation 177
24.13.12 G52 – Restore work plane rotation 177
24.13.13 G50 - Work plane rotation 178
24.13.14 G1050 – Disable Scaling 179
24.13.15 G1051 – Enable Scaling 179
24.13.16 G103 – Set RTCP parameters 181
24.13.17 G104 – Enable/restore RTCP 181
24.13.18 G104.1 – Enable/restore RTCP 181
24.13.19 G105 – Disable RTCP 181

25 TOOL RADIUS COMPENSATION 182

25.1.1 G41/G42 – Enable left/right radius tool compensation 182
25.1.2 G40 – Disable radius tool compensation 182
25.1.3 D – Tool diameter 183
25.1.4 G47 – Set tool entry mode 184

25.2 TOOL LENGHT COMPENSATION 185

25.2.1 G43 – Enable tool length compensation from parameter 185
25.2.2 G44 – Disable tool length compensation G43 186
25.2.3 G44.1 (2) – Suspend/Resume G43 186
25.2.4 G45 – Enable tool length compensation from tool table T 186
25.2.5 G46 – – Disable tool length compensation G45 187

25.3 INTERPOLATION MODES 188

25.3.1 G60 – Enable FAST interpolation without stop on segments 188
25.3.2 G61 – Enable interpolation with stop on segments 188
25.3.3 G62 – Wait for stop axis 188
25.3.4 G63 – Interpolation outside work plane (PX_MOVETO) 189
25.3.5 G64 – Interpolation on work plane (default) 189
25.3.6 G65 – Enable 3D interpolation PX_MOVETO with calculated stop on edge by parameters 190
25.3.7 G75 – Enable G64 for movement inside the work plane , G65 for movement outside the work plane 190
25.3.8 G66 – AFC – Adaptive Feed Control 191
25.3.9 G66 X-100 – NEW AFC – Adaptive Feed Control 192
25.3.10 G67 – Px_Moveto for movement outside plane and Px_Lineto for inside one 196
25.3.11 G68 – Always using of PX_LINETO in G1 "TRANSPORTED AXIS" (with possibility to combine with PX_MOVETO 196
25.3.12 G69 – LHK – Buffer look ahead depth 197

25.4 ISOUS FILTERS 198

25.4.1 G72 – N.U.R.B.S (Non Uniform Rational Bspline) (2D 3D) 199
25.4.2 G73 – NOISE (2D-3D) 202
25.4.3 G74 – RLS Remove Len Segment (2D 3D) 204
25.4.4 G106 – Smoothing (2D 3D) 205

25.5 INTERRUPT MACRO 208

25.5.1 G107 – Management Interrupt Macro 208

25.6 EMERGENCY MACRO M60000 IN STOP MODE 211

25.7 G108 MANAGEMENT SPECIAL AXES 212

25.7.1 G108.0 G108.1 G108.2 G108.3 - Master Slaves Axes 212
25.7.2 G108.4 G108.5 G108.6 SPEED MODE AXIS 213
25.7.3 G108.7 G108.8 G108.9 Physical exchange Axes 213
25.7.4 G108.10 G108.11 eGear Axes Management 214
25.7.5 G108.12 G108.13 - Peripheral speed management 215

25.8 VIRTUAL AXIS 216

25.8.1 G100 – Comando sincrono per asse virtuale 216

25.9 OTHER GENERIC G FUNCTIONS 217

25.9.1 G4 – Timed pause 217
25.9.2 G4.1 – Time Add on Calc Time 217
25.9.3 G10 – Enable external OVERRIDE potentiometer 217
25.9.4 G11 – Disable external OVERRIDE potentiometer 217
25.9.5 G101 – Axis Stop command 219
25.9.6 G80 – Forced pause by code 219
25.9.7 G81 – Management secondary axes LIMITS 220
25.9.8 G20 – Axes Values in Inch 220
25.9.9 G21 – Axes Values in Millimeters 220

25.10 VARIABLE MANAGEMENT FUNCTIONS 221

25.10.1 LOAD_VAR – load a variables file in current list 221
25.10.2 GET_VAR – Read a value from the loaded list and store it in a variable 221
25.10.3 WRITE_VAR – Write a value in the loaded list getting it from a variable 222
25.10.4 SAVE_VAR – save a variables file with current list 222
25.10.5 FILE_EXISTS – test if a file exists 222
25.10.6 ADD_VAR – Add a value to current list 223
25.10.7 REMOVE_VAR – Remove a value from current list 223
25.10.8 CLEAR_VAR – Remove all values from the current list 223
25.10.9 DIM_VAR – Size current list to a specific number of elements 224
25.10.10 COUNT_VAR – Get the number of elements in the current list 224

25.11 HM FUNCTIONS 225

25.11.1 Call of HM functions 225
25.11.2 Building of a HM function 226

25.12 M FUNCTIONS 227

25.12.1 M functions inside CN 227
25.12.2 M function inside PC 228

25.13 ESSENTIAL M FUNCTIONS 229

25.13.1 START M 229
25.13.2 END M 229
25.13.3 STOP M 229
25.13.4 PAUSE M 229
25.13.5 RESUME PAUSE M 230
25.13.6 GO BLOCK M 230
25.13.7 GO RETRACE M 230

25.14 DEFINING DEPTH AXIS 231

25.14.1 G48 Define Depth axis 231

25.15 MILD MODE – EDGE SMOOTHING 232

25.15.1 G49 MILD MODE Managing 232

25.16 PARALLEL TASKS 234

25.16.1 USTASK-ENDUSTASK 234
25.16.2 TASK.RUN 234
25.16.3 TASK.STOP 234
25.16.4 TASK.PAUSE 235
25.16.5 TASK.READVAR 235
25.16.6 TASK.WRITEVAR 235
25.16.7 TASK.STATUS 235
25.16.8 TASK.LOADCMD 236
25.16.9 TASK.PRIORITY 236

25.17 MAIN MACHINE PARAMETERS 237

25.17.1 Generic parameters 237
25.17.2 Axis parameters 240

25.18 PID PARAMETERS 250

25.19 SPINDLE PARAMETERS 251

25.20 EGEAR PARAMETERS 252

25.21 PERIPHERAL SPEED PARAMETERS.....253