SYNTEC

Macro Development Tool

User Guide

by : Syntec Inc.
date : 2006/07/20
ver : 9.0

Abstract

The develop environment is designed for the programmer to design and test the Macro program. By executing his program offline, the programmer can test it and know where the logical error is by graphic interface.

Edition check in

ID	Record	Date	Author	Edition
01	1.new	2006/07/20	James_lin	V9.0

Content

Chapter One – OpenCNC Developer Tool Introduction

A.System require

Processor：80486 or above

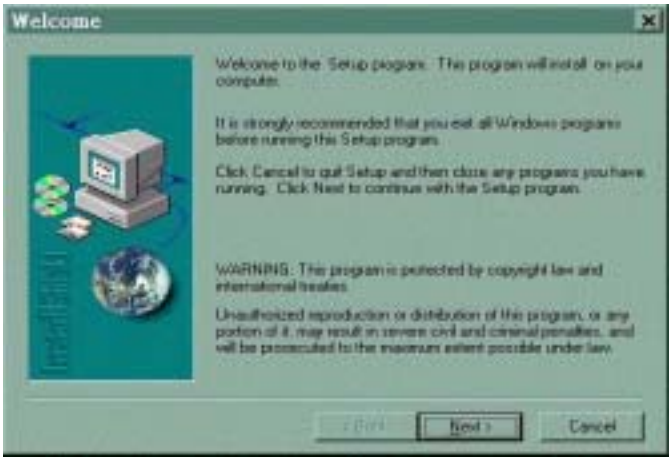
OS：Windows2000 Xp

HD：about 1MB free space

B.Install

1、Close other program first， And run setup.exe file。

2、 Wait a moment and then follow screen will appear。



3、Click Next。

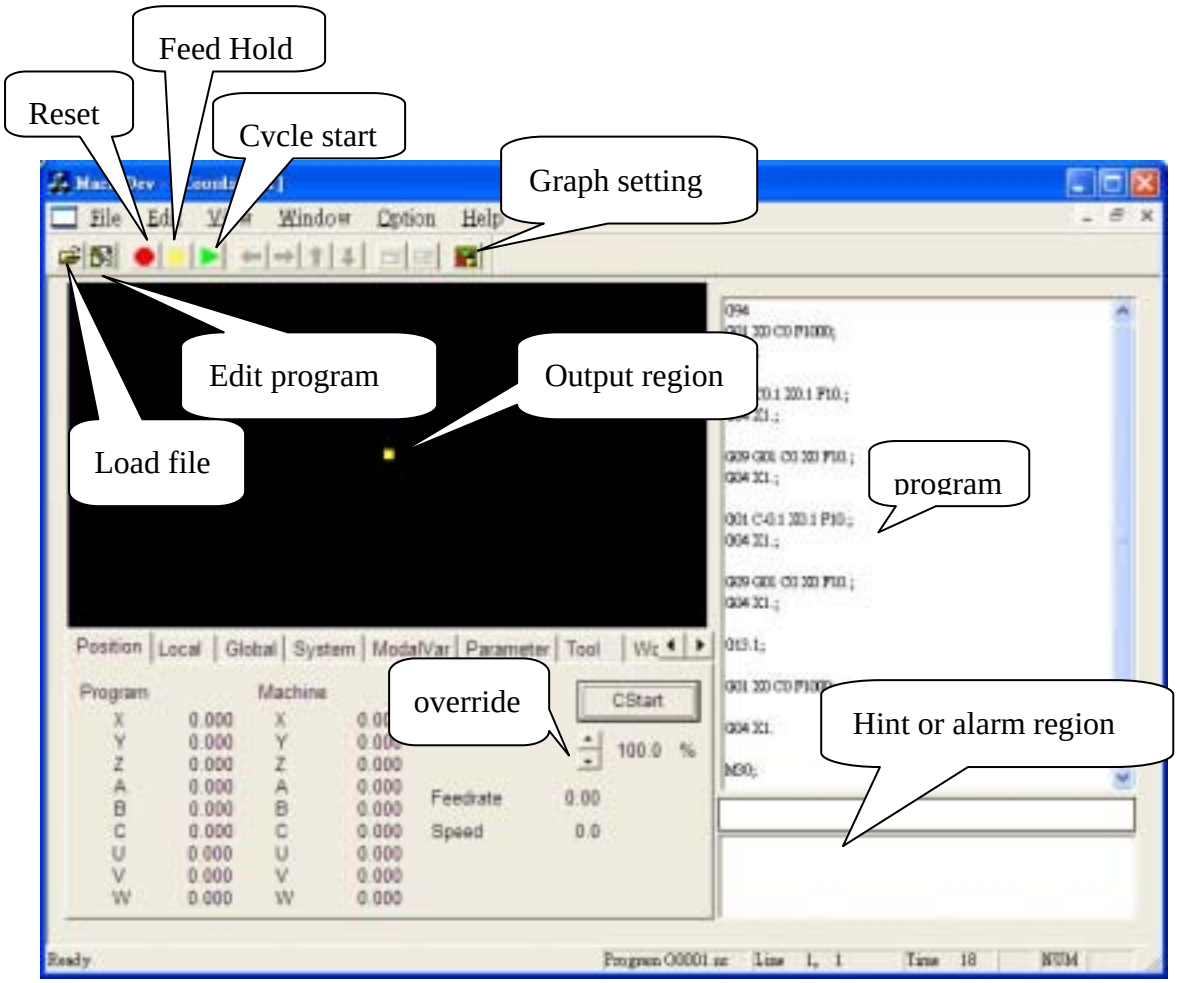
4、Choice Browse to change program install folder and click NEXT to continue。

5、 Wait a moment to install done。

C.Using opencnc

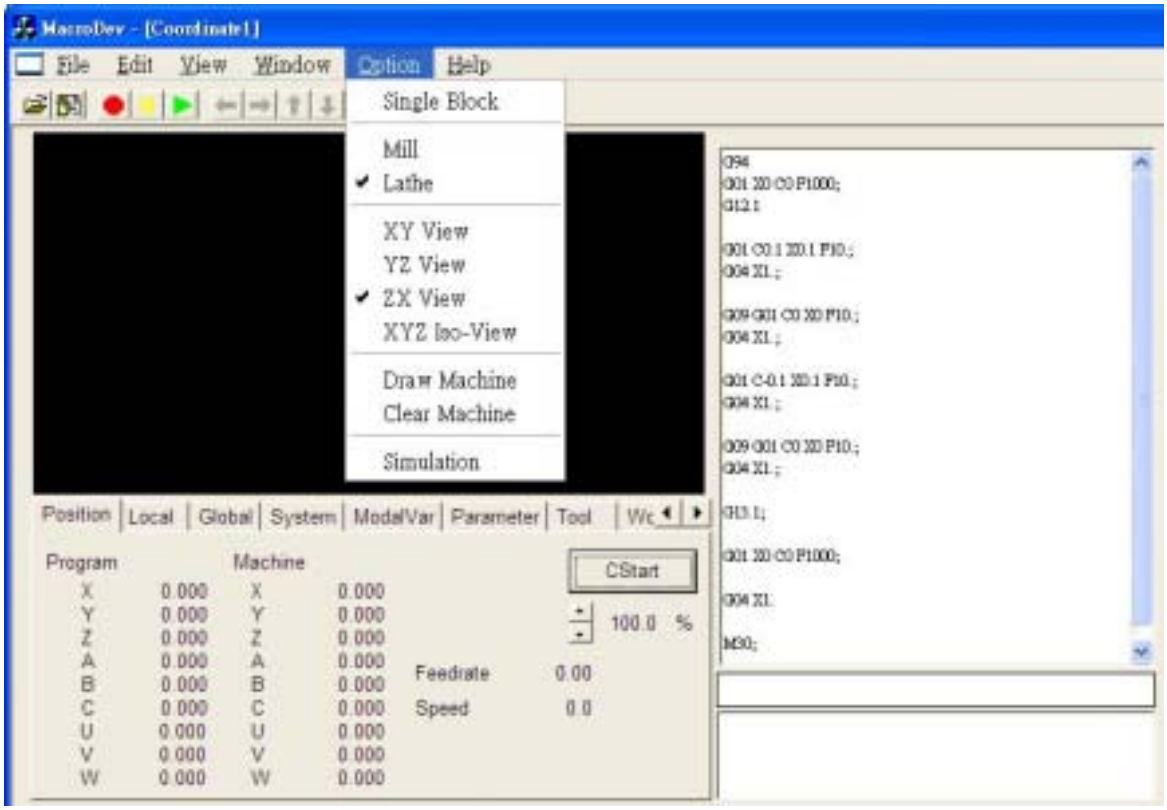
Operation guidance：

1、Click [start]->[Program]->[opencnc]->[macro dev] to enter themain program screen：



2、write a macro and load it， then click Cyclestart， the output will simulation on the screen。

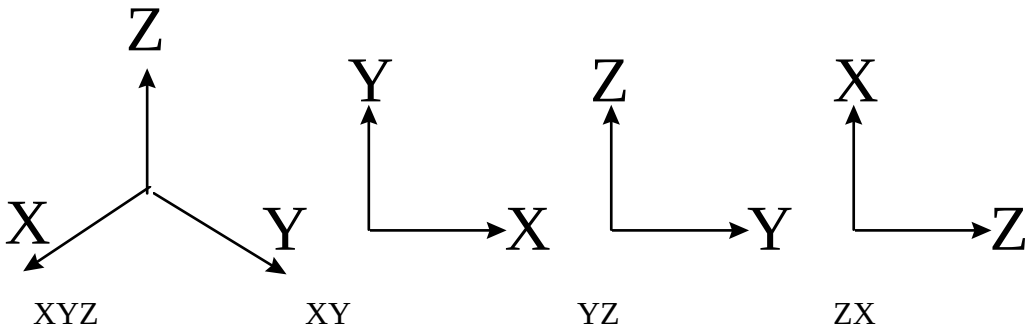
function：



1、Single Step：step and step to run the program,and click cycle start again between every step。

2、Mill or Lathe select

3、set the space：



4、Draw Machine or Clear Machine

5、Simulation：direct draw the resoult on the output Graph region。

Chapter Two –Macro Structure Motion language

A.Block Format

The format for a Motion Control block(single line of code) is describe below.

/ N G X Y Z A B C I J K F S T D M

- / Block Delete function validated at he option of operator.
- N If you use a sequence number, it must be the first in the block.
- G The preparatory function(s) G must follow N.
- X The linear dimension words follow G. Specify the X axis first.
- Y The linear dimension words follow G. Specify the Y axis second.
- Z The linear dimension words follow G. Specify the Z axis third.
- A The rotary dimension words follow G. Specify the X axis first.
- B The rotary dimension words follow G. Specify the Y axis second.
- C The rotary dimension words follow G. Specify the Z axis third.
- I The interpolation words follow the dimension words. Specify the X axis first.
- J The interpolation words follow the dimension words. Specify the Y axis second.
- K The interpolation words follow the dimension words. Specify the Z axis third.
- D The selection of tool compensation must follow K.
- F If you specify a feed rate that applies to more than one axis, F must follow the last dimension(and interpolation) to which it applies.
- S Spindle Speed Function shall follow immediately the “Feed Function“ or “Dimension” word.
- T The tool function selection follow S.
- M Any miscellaneous function(s) that you specify must last in the block, just ahead of the end of block character.
- End of Block Indicate the end of block with the carriage return / line feed character.

B.File format

‘%’ is head char，the line is also called head line。When head line have keyword ’@MACRO’，all statement at this file will process with this MACRO guidebook。If head line without keyword ’@MACRO’，statement at this file will process with standart ISO file，that means that will can’t use MACRO Syntax。’@MACRO’ keyword are all capitals char。

Example 一：MACRO file formate

```
% @MACRO
IF @1 = 1 THEN
    G00 X100.;
ELSE
    G00 Y100.;
END_IF;
M99;
```

Example：ISO file formate

```
%     //head line
G00 X100.;
G00 Y100.;
G00 X0;
G00 Y0;
M99;
```

C.Expressions

Operators

Operator	Symbol	Precedence
Parenthesis	( ) [ ]	1
Function Evaluation	Identifier(argument list)	2
Negative	-	3
Complement	NOT	3
Multiply	*	4
Divide	/	4
Modulus	MOD	4
Add	+	5
Subtract	-	5
Comparison	<,>,<=,>=	6
Equality	=	7
Inequality	<>	8
Boolean/Bitwise AND	&,AND	9
Boolean/Bitwise Exclusive OR	XOR	10
Boolean/Bitwise OR	OR	11

D.Statements

Assignment

Syntax：<Variable>: = <expression>;
Description：Assign a value to variable.
Example：

```
@1 := 123;
#1 := #3;
```

GOTO

Syntax：**GOTO** n;
Description：Jump to line numbers N
Example 1：

```
GOTO #3;
```

Example：

```
% @MACRO        // Start MACRO language
…
IF( #1 = 2 ) THEN GOTO 100;
G01 X10. Y10.;
…
N100 G01 X30. Y30.;
…
M02;
```

EXIT

Syntax：EXIT;
Description：Break loop or exit jump control
Example：

Refer to WHILE example

CASE

Syntax：

CASE <INT expression> **OF**
    <INT>:
        <Statement list>
    <INT>,<INT>,<INT>:
        <Statement list>
    <INT>,…<INT>:
        <Statement list>
ELSE
    <Statement list>
END_CASE;

Description：Conditional execution by cases. According to the result of INT expression in the CASE, controller executes corresponding program block.

Example：

```
% @MACRO         // Start MACRO language
#1 := 8;
G01 G91 G92 X20. Y15. F200000;

CASE #1 OF
1:
    X(1.0*1);
    Y(1.0*1);
2:
    X(1.0*2);
    Y(1.0*2);
3,4,5:
    X(1.0*#1);
    Y(1.0*#1);
ELSE
    X(1.0*6);
    Y(1.0*6);
END_CASE;

X(1.0) Y(1.0);
M30;
```

REPEAT

Syntax：

REPEAT
<Statement list>
UNTIL <Condition> **END_REPEAT**;

Description：REPEAT loop control

Example：

```
% @MACRO        // Start MACRO language
#10 := 30.;
#11 := 22.5.;
#12 := #10/2;
#13 := #11/2;
#14 := 2.0;
#15 := 1.5;
G01 G92 X#12 Y#13 F200.0;

REPEAT
    G00 X(#12+#14) Y(#13+#15);
    G01 X(#12+#14) Y(#13-#15);
    X(#12-#14) Y(#13-#15);
    X(#12-#14) Y(#13+#15);
    X(#12+#14) Y(#13+#15);
    #14 := #14 + 2.0;
    #15 := #15 + 1.5;
UNTIL (#14 > #12) OR (#15 > #13) END_REPEAT;

X(1.0) Y(1.0);
M30;
```

WHILE

Syntax：

WHILE <Condition> **DO**
<Statement list>
END_WHILE;

Description：WHILE loop control

Example：

```
% @MACRO        // Start MACRO language
#10 := 20.;
#11 := 15.;
#12 := #10/2;
#13 := #11/2;
#14 := 2.0;
#15 := 1.5;
G01 G92 X#12 Y#13 F200.0;

WHILE (#14 <= #12) AND (#15 <= #13) DO
    G00 X(#12+#14) Y(#13+#15);
    G01 X(#12+#14) Y(#13-#15);
    X(#12-#14) Y(#13-#15);

    IF #14 > 6.0 THEN
        EXIT;
    END_IF;

    X(#12-#14) Y(#13+#15);
    X(#12+#14) Y(#13+#15);
    #14 := #14 + 2.0;
    #15 := #15 + 1.5;
END_WHILE;

X(-5.0) Y(5.0);
M02;
```

FOR

Syntax：

FOR <INT variable1> := <expression1> **TO** <expression2> [**BY** <expression3>]
DO <Statement list>
END_FOR;

Description：FOR loop control

variable1 : loop control variable
expression1 : loop start number，long or double
expression2 : loop end number，long or double
expression3 : loop increase（decrease）number，long or double
statement list :execute statement

Examples：

```
% @MACRO        // Start MACRO language
#1 := 2.0;              (*INITIAL RADIUS*)
#2 := 8.0;              (*FINIAL RADIUS*)
#3 := 9;                (* SIDES*)
#4 := 360.0 / #3;       (*THETA*)
#5 := (180.0 + #4)/2;   (*START ANGLE*)
G91 G92 X0. Y0. F300000;
G01 X(#1);

FOR #6:=#1 TO #2 BY 2.0 DO
    #7 := 2.0 * #6 * COS(180.0-#5);
    #8 := (#7/2.0) / COS(180.0/6);
    #9 := #5;
    G01 X(1.0);

    FOR #10:= 1 TO #3 DO
        G03     X(#7*COS(#9))
                Y(#7*SIN(#9))
                I(#8*COS(#9-180.0/6))
                J(#8*SIN(#9-180.0/6));
        #9 := #9 + #4;
    END_FOR;
END_FOR;
```

IF

Syntax：

IF <Condition> **THEN**
    <Statement list>
ELSEIF <Condition> **THEN**
    <Statement list>
ELSE
    <Statement list>
END_IF;

Description：conditional execution

Examples：

```
% @MACRO        // Start MACRO language
#1 := 3.0;
G01 G91 G92 X20. Y15. F200000;

IF #1 = 1 THEN
    X(1.0*1);
    Y(1.0*1);
ELSEIF #1 = 2 THEN
    X(1.0*2);
    Y(1.0*2);
ELSEIF #1 = 3 THEN
    X(1.0*3);
    Y(1.0*3);
ELSE
    X(1.0*4);
    Y(1.0*4);
END_IF;

X(1.0) Y(1.0);
M30;
```

E.Functions Listing

Function	**Description**
ABS	Calculates the absolute value of a number Ex: #1 := ABS(-2.3); // #1 will be 2.3
ACOS	Calculates the arc cosine of a number Ex: #1 := ACOS(#10);
ASIN	Calculates the arc sine of a number Ex: #1 := ASIN(#10);
ATAN	Calculates the arc tangent of a number Ex: #1 := ATAN(#10);
COS	Calculates the cosine of a number Ex: #1 := COS(#10);
MAX	Determines the maximum of two inputs Ex1: #1 := MAX(10,20); // #1 will be 20 Ex2: #1 := MAX(#2,#3);
MIN	Determines the minimum of two inputs Ex1: #1 := MIN(10.0,20.0); // #1 will be 10.0 Ex2: #1 := MIN(#10,#11);
SIN	Calculate the sine of a number. Ex: #1 := SIN(#10);
SQRT	Calculates the square root of a number. Ex1: #2 := SQRT(3); // #2 will be 1.732.. Ex2: #2 := SQRT(#10);
TAN	Calculates the tangent of a number. Ex: #1 := TAN(#10);

Function	**Description**
SIGN	Return sign of a number, –1 for negative number, 1 for positive number, 0 for zero number. Ex: IF(SIGN(#10) > 0) THEN …. END_IF;
CEIL	Return the smallest integer that is greater than or equal to a number. Ex1: #2 := CEIL(2.3); // #2 will be 3 Ex2: #2 := CEIL(#10);
FLOOR	Return the largest integer that is less than or equal to a number Ex1: #2 := FLOOR(2.3); // #2 will be 2 Ex2: #2 := FLOOR(#10);
ROUND	Return the value of the argument rounded to the nearest long value Ex1: #2 := ROUND(2.3); // #2 will be 2 Ex2: #2 := ROUND(#10);
STD	Standardize arguments, read a number, in argument one, by least increment method, in argument two, when necessary for decimal point programming Ex: #9 := STD(#9,#1600); // normalize by distance axis

PUSH	Push value into Macro stack.
	Ex:
	PUSH(#1); // push #1 variable into stack
	PUSH(#3); // push #3 variable into stack

POP	Pop value from Macro stack. Ex: #1 := POP(); // popup a value to #1

Function	**Description**
STKTOP	Peek the stack value by index form top one. Ex: STKTOP(0) the most top element value STKTOP(1) the element value below top one STKTOP(2) the element value below top two …etc

ALARM	Issue Macro alarm
	Ex:
	ALARM(300); // issue macro alarm id 300
	ALARM(#1); // #1 must be integer

SLEEP	Temporarily give up this cycle execution Ex: SLEEP();
WAIT	Wait until all previous motion/logic commands are finished. Ex: WAIT();
RANDOM	Generates a pseudorandom number. Ex: #1 := RANDOM();
AXID	Lookup axis identifier, the axis identifier is the machine axis number. When the specified axis's name not exist, then return value will be vacant. Ex: Assume 6^{th} axis's name is Y2, 2^{nd} axis's name is Y, then AXID(Y) will return 2 AXID(Y2) will return 6

STDAX	Standardize arguments, read a number, in argument one, by least increment method, in argument two is axis address
	Ex:
	#24 := STDAX(#24,X); // normalize by X dimension
	#3 := STDAX(#3,A); // normalize by A dimension

OPEN("file name")	Open file, if success then return 1, otherwise return 0。 PRINT() function will work after this function execute。 If file name is "COM", system will open RS232 port, and parameter 3905..etc will need to set。

Function	**Description**

	Example：
	OPEN(“PROBE.NC”);
	//open PROBE.NC file for output data
	Example
	OPEN(“COM”); //open serial port
	PRINT("\p"); //output ’%’ char
	FOR #1 = 1 TO 5000 DO
	#30 = #1 * 10.;
	PRINT("G01 X#30"); //output G01 X10.000…
	END_FOR;
	PRINT(“\p”); //outpur ‘%’char
	CLOSE(); //close serial port

CLOSE()	Close the file which open with “OPEN” function。When program end then all open file will close automatic。Avoid “PRINT” function after close file。 example： CLOSE(); // close file
PRINT(“this is output string”)	This function is for output string to file use, the variable that inside the string will change to it’s value, if this function run success then it will return 1, otherwise will return 0。 example： @53 = 20; #3 = 23.1; PRINT(“G01 X#3 Y@53 Z20.0”); Output result G01 X23.100 Y20 Z20.0; Char ‘\’ is skip char, special char define as follow： ‘\\’ outpur ’\’ char ‘\@’ outpur ’@’ char ‘\#’ outpur ’#’ char ‘\p’ outpur ’%’ char Example for output: G01 X(@20/@30) Y#20/2.0; The Syntax format is: PRINT(“G01 X(\@20/\@30) Y\#20/2.0”);
GETARG(*address*)	Read caller argument in subroutine

Function	**Description**

	example：
	O0001 main program
	G101 X30. Y40. Z1=40. Z2=50.;
	.
	G0101 extension G code macro
	#1 = GETARG(X); // the value of X argument will store in #1
	#2 = GETARG(Z1); // the value of Z1 argument will put in #2
	#3 = GETARG(W); // without W argument, #3 will be "VACANT"

GETTRAPARG(*address*)	For G66/G66.1 modal macro call handler to get trap block's information
	example：
	O0001 main program
	G66 P100 X100. Y100.
	G01 X20.
	.
	O0100 subroutine
	#1 = GETARG(X); // Get X argument 100. to #1
	#2 = GETTRAPARG(X); // Get trap block X argument 20. to #2

DBOPEN("filename")	Load specify XML database。
	example：
	DBOPEN("FLAT\\TAB01");
	// load FLAT\\TAB01 database
	example：
	#1 = 51;
	DBOPEN("FLAT\\AB#1[3]ZZ");
	//load FLAT\\AB051ZZ database, [3] mean 3 available value。

DBLOAD(CycleNo)	Load data from current XML data base
	example：
	// load FLAT\\TAB01 database
	DBOPEN("FLAT\\TAB01");
	// load first data
	DBLOAD(0);
	…

Function	Description

	// load second data
	DBLOAD(1);
	…

COMMENT(“comment string”)	This function can output comment string, the variable that inside the string will change to it’s value, if this function run success then it will return 1, otherwise will return 0。
	example：
	@53 = 20;
	#3 = 23.1;
	COMMENT(“// G01 X#3 Y@53 Z20.0”);
	Resoult:
	// G01 X23.100 Y20 Z20.0;
	Char ‘\’ is skip char, special char define as follow：
	‘\\’ define ’\’ char
	‘\@’ define ’@’ char
	‘\#’ define ’#’ char
	‘\p’ define ’%’ char
	If the output is:
	// THIS IS TURNING CYCLE
	Then Syntax format is
	COMMENT(“// THIS IS TURNING CYCLE”);

DRAWHOLE()	Draw a hole using current tool radius, line color, fill color at current position. This function only take effect under graphics simulation
DRAWMARK(*shape,size,color*)	Draw a mark with specified shape, size, color at current position. The marker will fix in size not regards to zoom scaling. Size: In Pixel Shape:0:Circle,1:Square;2:Diamond. This function only take effect under graphics simulation
SETDRAW(*LineColor*) or SETDRAW(*LineColor,FillColor,ToolRadius*)	To assign draw style LineColor: use for draw contouring line ToolRadius: use for draw hole radius size FillColor: use for fill hole interior. This function only take effect under graphics simulation
PARAM(no)	To read specified system parameter number #1 = PARAM(3204) // to access PLC scan time interval

Function	**Description**
SYSVAR(AxisGroupID, No)	To read system variable of specified coordinate system. AxisGroupID axis group identifier, 1 for first axis group, 2 for 2nd axis group, etc… No The system variable number. e.g. #1 = SYSVAR(1, 1000); // to read interpolation mode of first axis group.

SCANTEXT(No)	To scan text string from global variable.
	Notes: Because string is local, so only can stores in local variable, and can not save to global variable. That is, following will get wrong result.
	
	e.g.
	// scan string text from @300
	#1 = SCANTEXT(300);
	
	// following will get wrong result
	@100 = #1; // @100 will loss string information
	DBOPEN(“ABC_@100”); // this will got wrong result
	
	// following will get correct result
	#100 = #1; // #100 contain valid string from @300~
	DBOPEN(“ABC_#300”); // correct result

F.Variables

Vacant	#0, @0 is always VACANT
Local	#1 ~ #50
System	#1000 ~
Global	@1~

Globol variable tabla

Variables	Usage	Macro	PLC	MMI
@1~@400	Macro Internal variables	R/W	N/A	R
@401~@449, @481~@500	CNC⇔PLC Interface	R	※	R
@450~@480	Macro⇔PLC Interface	R/W	R/W	R
@10000~ @14095	Corresponding to PLC register R0~R4095	※	※	※

※R Resource table

Variables	Example	R/W rule		Bit Access	Permanent
		Macro or MMI	PLC		
R0~R39	CNC system interface	R	Refer to PLC guide book	Yes	No
R40~R49	PLC Alarm area				
R50~R80	User define area	R/W	R/W	Yes	No
R81~R100	Corresponding to system parameter 3401~3420	R	R	Yes	No
R101~R102	Tool state	R/W	R/W	Yes	Yes
R103~R255	User define	R/W	R/W	Yes	Yes
R256~R511					No
R512~R639	CNC system interface	R	Refer to PLC guide book	Yes	No
R640~R1023				No	
R1023~R4095	User define	R/W	R/W	No	No

Comment

(* This is comment *)

// This is comment

G.Macro Program

Call Methods:

Syntax	Description	Examples
M98 P_ H_ L_	Subprogram call, P_ subroutine name H_ start N number L_ repeat times	M98 P10 L2;
G65 P_ L_ *addresses*	Macro call P_ subroutine name L_ repeat times	G65 P10 X10.0 Y10.0;
G66 P_ L_ *addresses*	Modal macro call, for every move block P_ subroutine name L_ repeat times	Example： G66 P10 X10.0 Y10.0; X20. Y20. Description： X20 and Y20. move command block will call O0010
G66.1 P_ L_ *addresses*	Modal macro call, for every block P_ subroutine name L_ repeat times	Example： G66.1 P10 X10.0 X20. G04 X2.; M31; Description： X20、G04 X2 and M31.every block will call O0010
G_ L_ *addresses*	External G call L_ repeat times	G128 L3 X1.0;(*will call G0128 three times*)
T_	Tool selection by subprogram, any T code inside T-subprogram will be treat as ordinary T call.	T3;*(will call T0000)*
M_ *addresses*	M Code Macro Call	M13 A_ B_ C_; Call M0013 Macro。 M13 must register in parameter No.3601~

Return Methods:

Syntax	Description	Examples
M99	Return	M99;
M99 P_	Return and go to specified label P_ sequence number	M99 P100; Return to main program N100
M99 Q_	Return and go to specified line number Q_ line number	M99 Q100; Return to main program line100
G67	Modal macro call cancel	G67;

Argument specification

Address	Variable Number	Address	Variable Number	Address	Variable Number
A	#1	J	#5	U	#21
B	#2	K	#6	V	#22
C	#3	M	#13	W	#23
D	#7	P	#16	X	#24
E	#8	Q	#17	Y	#25
F	#9	R	#18	Z	#26
H	#11	S	#19		
I	#4	T	#20		

About extension address, X1=, Please use Macro function GETARG(*address*) to get value

System Variables

No.	Account	R/W
#0	VACANT	R/W
#1~#50	Local variable for macro program	R/W

Modal information

#1000	Interpolation mode, 00/01/02/03/33/34/35	R/W
#1002	Contouring plane selection mode, 17/18/19	R
#1004	Absolute/Incremental command mode, 90/91	R
#1006	Stored stroke check mode, 22/23	R
#1008	Cutting feed mode, 94/95	R
#1010	Inch/Metric mode, 20/21	R
#1012	Cutter compensation mode, 40/41/42	R
#1014	Tool length compensation mode, 43/44/49	R
#1016	Scaling mode, 50/51	R
#1018	Spindle speed mode, 96/97	R
#1020	Cutting feedrate control mode, 61/62/63/64	R
#1022	Rotation mode, 68/69	R
#1024	Spindle speed fluctuation detection mode, 25/26	R
#1026	Polar coordinate interpolation mode, 12/13	R
#1028	Polar coordinate command mode, 15/16	R
#1030	Cutter radius offset selection number, D Code	R
#1032	Tool length offset selection number, H Code	R
#1034	Cutting condition selection number, S Code	R
#1036	Tool selection number, T Code	R
#1038	Miscellaneous function number, M Code	R
#1040	Current workpiece coordinate number	R
#1042	Program sequence number, N Code	R
#1044	Last block interpolation mode, could be 4(dwell) or vacant(M_ S_ T_ F_) for G66.1 modal macro call	R
#1046	Feedrate command, F Code	R
#1048	Caller’s current line number	R
#1050	Program start sequence number	R
#1052	Program start line number	R
#1054	Spindle operation state(M03/M04/M05)	R
#1056	Program sequence number after corner processing, N Code	R
#1058	The spindle number before restart	R

Operation control/status

#1500	Quiet mode, 1(Quiet mode), 0(Normal mode)	R/W
#1502	Single block control word	R/W
#1504	Feed control word	R/W
#1506	Simulation mode, 1(in simulation mode),0(in normal mode)	R
#1508	my session ID inside mode group	R
#1510	The current active session of multi-session program in CNC main system. 0 for execute the multi-session program simultaneously; 1 to execute $1 program only; 2 to execute $2 program only.	R
#1600	Distance least input increment	R
#1602	Time/Rotation angle least input increment	R
#1604	Use U/V/W addresses as X/Y/Z axis incremental command mode, 1(Use as X/Y/Z incremental command), 0(As normal axis command)	R
#1606	The count of element in macro stack.	R
#1608	Flag for skip function position latched, 1 for latched, 0 for not latched.	R
#1610	Spindle orientation stop angle	R/W
#1612	Default workpiece number	R/W
#1614	Default spindle speed	R/W
#1616	Break point sequence number	R
#1618	Break point line number	R
#1620	Current sequence number	R
#1622	Current point line number	R
#1624	Current active spindle ID	R

Single Block Control Word(#1502)

	Description
Bit 0	Single block inhibit. 0(default): single block enabled, 1: single block disabled. When single block stop is disabled, single block stop operation is not performed even if the single block switch(by MLC C bit) is set to ON.
Bit 1	(*not yet implement*) Completion of an auxiliary function. 0(default): to be awaited, 1: not to be wait. When a wait for the completion of auxiliary functions (M, S, and T functions) is not to be waited, program execution proceeds to the next block before completion of auxiliary functions. Also, distribution completion signal DEN is not output.

- When the power is turned on, the value of this variable is 0.
- When single block stop is disabled, single block stop operation is not performed even if the single block switch(by MLC C bit) is set to ON.

Feed Control Word(#1504)

	Description
Bit 0	Feedhold Inhibit. 0(default): feedhold enabled, 1: feedhold disabled. When feed hold is disabled: When the feed hold button is held down, the machine stops in the single block stop mode. However, single block stop operation is not performed when the single block mode is disabled with variable #1502. When the feed hold botton is pressed then released, the feed hold lamp(by MLC S bit) come on, but the machine does not stop; program execution continues and the machine stops at the first block where feed hold is enabled.
Bit 1	Override Inhibit. 0(default): override enabled, 1: override disabled. When ederate override is disabled, an override of 100% is always applied regardless of the setting of the ederate override switch(by MLC Register) on the machine operator's panel
Bit2	(*not yet implement*) Exact stop inhibit. 0(default): exact stop enabled, 1: exact stop disabled. When exact stop check is disabled, no exact stop check(position check) is made even in blocks including those which do not perform cutting

- When the power is turned on, the value of this variable is 0.
- When feed hold is disabled:
  1. When the feed hold button is held down, the machine stops in the single block stop mode. However, single block stop operation is not performed

when the single block mode is disabled with variable #1502.

2. When the feed hold botton is pressed then released, the feed hold lamp(by MLC S bit) come on, but the machine does not stop; program execution continues and the machine stops at the first block where feed hold is enabled.

- When feedrate override is disabled, an override of 100% is always applied regardless of the setting of the feedrate override switch(by MLC Register) on the machine operator’s panel.
- When exact stop check is disabled, no exact stop check(position check) is made even in blocks including those which do not perform cutting.

Current position

#1301~#1316	Block end position in workpiece position	R
#1321~#1336	Current position in machine coordinate, this value can't be read during movement.	R
#1341~#1356	current position in workpiece coordinate	R
#1361~#1376	Skip signal position in workpiece coordinate, the tool position where the skip signal is turned on in a G31 (skip function) block is held in these variables,	R
#1381~#1396	Tool length compensation vector	R
#1401~#1403	Last arc block center vector,(I,J,K)	R
#1411~#1413	Block end position in workpiece position, index by 1441(X);1442(Y);1443(Z)	R
#1421~#1436	Current encoder position of in workpiece coordinate machine	R

Runtime state variable

#1800	Rigid tapping tracking error in revolution	R
#1801	Rigid tapping z-direction tracking error in BLU	R
#1802	Rigid tapping tracking error, maximum magnitude	R
#1810	Guidance remain distance, in LIU	R
#1811	Guidance x-direction remain distance, in LIU	R
#1812	Guidance y-direction remain distance, in LIU	R
#1815	Indicator for is in guidance function, 0: No; 1: Yes	R
#1816	Guidance ederate, in IU/min	R/W
#1820	Mute state, discard all command during mute state ON. 0: OFF, 1: ON, this state variable also available from G10 L1100 command	R/W
#1821	Accumulated cutting length, in IU	R/W
#1822	Cutting Feedrate Command, in mm/min	R/W
#1823	Spindle Speed Command, in RPM	R/W
#1824	Active Feed Control Mode, G61/G62/G63/G64	R
#1825	Active interpolation G code mode	R
#1827	Active workpiece coordinate number	R
#1829	Active number of high precision high speed parameter	R
#1901~1916	Workpiece coordinate system shifting amount.	R/W

Modal variables

Modal variables will automatically be clear to vacant when system been reset

#2001~#2100	Modal variable for internal use	R/W
#3001~#3100	Modal variable for manufacturer	R/W

Custom parameter

#4001~#4100	Custom parameter for internal use	R
#5001~#5100	Custom parameter for manufacturer	R

Interface signals

#6001~#6032	Bit value interface of MLC (Corresponding to C101~C132/ S101~S132) ,example:
	@1 := #6001; // assign C101 value into @1
	#6001 := @2; // assign @2 value into S101
		R/W

Mode Group Variables

Mode group variables will automatically be clear to vacant when specified mode group be reset

#7001~#7050	Modal variable for internal use	R/W
#7101	The number of axis group in mode group	R

Tool compensation variable(R/W)

The compensation number 0 all have zero value

There are only 400 compensation value in this version

Compensation Number	Tool length compensation(H)		Cutter compensation(D)	
	Geometric compensation	Wear compensation	Geometric compensation	Wear compensation
0	#11000	#10000	#13000	#12000
1	#11001	#10001	#13001	#12001
.	.	.	.	.
.	.	.	.	.
200	#11200	#10200	#13200	#12200
.	.	.	.	.
.	.	.	.	.
400	#11400	#10400	#13400	#12400
.	.	.	.	.
.	.	.	.	.
999	#11999	#10999	#13999	#12999

Workpiece coordinate system compensation values (workpiece zero point offset values)

There are only 16 workpiece coordinate system in this version

#20001~#20006	External workpiece zero point offset value	R/W
#20021~#20026	Workpiece 1 zero point offset value,G54	R/W
#20041~#20046	Workpiece 2 zero point offset value,G55	R/W
…		R/W
#25981~#25986	Workpiece 299 zero point offset value	R/W

Reference point position

The reference point 1 is always at home position

There are only 4 reference points in this version

#26001~#26006	Reference point 1 position	R/W
#26021~#26026	Reference point 2 position	R/W
#26041~#26046	Reference point 3 position	R/W
…		R/W
#31981~31986	Reference point 300 position	R/W

H.Hinting of write extension G code：

- Use locol variable #1~#50 for program variable usually。
- Modal Variables(#4001~#4100；#5001~#5100) is the resource for extended G-code. For saving the use of this resource, please use it when share data between G-code.。
- Please use the argument to call the extended G-code.。
- Please use the default value in the extension G-code by Custom Parameter (#4001~#4100, #5001~ #5100) 。
- Don't change the status of Modal G-code(G91/G90,G40/G41/G42,…,etc). If needs, please recover it when you exit.。
- Please use the function named "STD()" when you using the argument of length and angle.。
- The Simulation function is invalid when you setting the coordinator by G-code like G92, G54 and G52.。

I.Extended Interpolation G Code：

The variable #1000 record the interpolation mode, so when exit macro and you want to keep this macro for interpolation mode, you only need to set #1000 as G code number。

Then every axis move statement after this macro will call this macro。

When read G00/G01/G02/G03/G31/G33, system will stop extended G code interpolation。

Example：

Developer a macro with cycle mode G21, and following are program Syntax：

```
.
.
    G21 X_ Z_ R_;        // G21 mode
    X_;                  // G21 mode
    X_;                  // G21 mode
    X_;                  // G21 mode
    G00 Z_;              // G00 mode
.
.
```

Macro architectonic

```
% @MACRO

.           //  Macro main program
.
.

#1000 := 21;   // Set G21 as current interpolation mode

M99;
```

J.MACRO example：

This sample show the lathe G21 macro, please refer Lathe Programming Manual for other G code specification

```
% @MACRO
// WHEN NO X(U) Z(W) ARGUMENTS,THEN LOAD MODAL INFORMATION
// ELSE DO THREAD CUTTING
// #8(E)        LEAD COUNT PER INCH
// #18(R)       TAPER AMOUNT
// #2070        RECORDED TAPER AMOUNT
// #2071        RECORDED Z AXIS AMOUNT

// PROCESS NO ARGUMENT CASE WHICH MAY CAUSE FROM MODAL
RESTORE
IF( #21 = #0 AND #23 = #0 AND #24 = #0 AND #26 = #0 ) THEN
    M99;
END_IF;

// PROCESS TAPER
IF (#1000 <> 21 ) THEN
    // WHEN FIRST ENTRY, CLEAR TAPER MODAL STATE
    #2070 := 0;
    #2071 := 0;
    #2072 := #0;
END_IF;

IF( #18 <> #0 ) THEN
    // THERE ARE TAPER ADDRESS, RECORD IT INTO MODAL VARIABLE
    #2070 := #18;
ELSE
    // NO TAPER ADDRESS APPEAR, INHERIT MODAL STATE
    #18 := #2070;
END_IF;

IF( #9 <> #0 ) THEN
    // THERE ARE LEAD ADDRESS, RECORD IT INTO MODAL VARIABLE
    #2072 := #9;
```

```
ELSE
    // NO LEAD ADDRESS APPEAR, INHERIT MODAL STATE
    #9 := #2072;
END_IF;

IF( #23 <> #0 OR #26 <> #0 ) THEN
    // WHEN THERE ARE W OR Z ADDRESS APPEAR, THEN
    // CHECK WHICH KIND OF COMMAND BEEN ADDRESS, AND
    // SAVE IT INTO MODAL VARIABLE
    IF( #26 <> #0 ) THEN
        // Z ADDRESS
        #2071 := #26 - #1303;
    ELSE
        // W ADDRESS
        #2071 := #23;
    END_IF;
ELSE
    // WHEN THERE ARE NO Z/W ADDRESS, THEN INHERIT IT FROM
    // MODAL VARIABLE
    #26 := #2071 + #1303;
END_IF;

// PROCESS E ADDRESS
IF( #8 <> #0 AND #9 = #0 ) THEN
    IF( #1008 = 94 ) THEN
        // FEED PER MINUTE, CALCULATE MM/MIN = LEAD * SPINDLE
    SPEED
        #9 := (25.4 * #1034) / #8;
        #2072 := #9;
    ELSE
        // FEED PER REVOLUTION, CALCULATE MM/REV = LEAD
        #9 := 25.4 / #8;
        #2072 := #9;
    END_IF;
END_IF;

// STANDARDIZE ARGUMENT
#9 := STD(#9,#1600);
```

```
#21 := STD(#21,#1600);
#23 := STD(#23,#1600);
#24 := STD(#24,#1600);
#26 := STD(#26,#1600);
#18 := STD(#18,#1600);

// working variable
// #31          chamfer start point relative to block end X
// #32          chamfer block X-direction displacement
// #33          chamfer amount
// #36          thread head number iterative count
// #37          thread start angle

// READ CHAMFER AMOUNT
#33 := (#4043 * #9) / 10.0;

// COPY X,Z INFORMATION INTO U,W

// PROCESS X ADDRESS
IF( #24 <> #0 ) THEN
      #21 := #24 - #1301;
END_IF;

// PROCESS Z ADDRESS
IF( #26 <> #0 ) THEN
     #23 := #26 - #1303;
END_IF;

// process H addesss, the head number
IF( #11 <> #0 ) THEN
     #11 := ROUND(#11);
ELSE
     // set default head number
     11 := 1;
END_IF;

// CALCULATE CHAMFER START POINT RELATIVE TO BLOCK END POINT IN
X
```

```
#31 := (SIGN(#23) * #33 * 2 * #18)/#23;

// CALCULATE CHAMFER BLOCK X-DIRECTION DISPLACEMENT
#32 := -SIGN(#21)*#33*2;

FOR #36:=1 TO #11 DO

    // calculate thread start angle
    #37 := (360.0 / #11) * (#36 - 1);

    G00 U( #21 + #18*2 );
    G33 U-(#18*2-#31) W(#23-SIGN(#23)*#33) Q#37 F(#9*#11);
    G33 U#32 W(SIGN(#23)*#33) Q#37;
    G00 U-#21-#32-#31;
    G00 W-#23;

END_FOR;

// SET INTERPOLATION MODE TO 21
#1000 := 21;

// RETURN
M99;
```

Chapter Three－Appendix

Basic G Code Table

Code	Function
G00	POSITIONING
G01	LINEAR INTERPOLATION
G02	CIRCULAR INTERPOLATION (Clockwise)
G03	CIRCULAR INTERPOLATION (CounterClockwise)
G04	Dwell
G10	PROGRAMMABLE DATA INPUT
G15	CANCEL POLAR COORDICATES COMMAND MODE
G16	POLAR COORDICATES COMMAND MODE
G17	XY PLANE SELECTION
G18	ZX PLANE SELECTION
G19	YZ PLANE SELECTION
G28	RETURE TO REFERENCE POSITION
G29	RETURE FROM REFERENCE POSTION
G30	2^{nd},3^{rd} and 4^{th} REFERENCE PPOSTION RETURE
G31	SKIP FUNCTION
G33	THREAD INTERPOLATION
G40	CANCEL CUTTER COMPENSTAION
G41	LEFT CUTTER COMPENSTAION
G42	RIGHT CUTTER COMPENSTAION
G43	POSTIVE TOOL LENGTH COMPENSATION
G44	NEGATIVE TOOL LENGTH COMPENSATION
G49	CANCEL TOOL LENGTH COMPENSATION
G50	SCALING
G51	PROGRAMMABLE MIRROR IMAGE
G52	LOCAL COORDINATE SYSTEM
G53	MECHINE COORDICATE SYSTEM SELECTION
G54	WORKPIECE COORDICATE SELECTION
G55	SECOND WORKPIECE COORDICATE SELECTION
G56	THIRD WORKPIECE COORDICATE SELECTION
G57	FOURTH WORKPIECE COORDICATE SELECTION
G58	FIVETH WORKPIECE COORDICATE SELECTION
G59	SIXTH WORKPIECE COORDICATE SELECTION
G65	SIMPLE CALL
G66	MACRO CALL
G67	CANCEL MACRO CALL
G67	CANCEL COORDINATE ROTATION
G68	COORDINATE ROTATION
G70	INPUT IN INCH
G71	INPUT IN MM

Code	Function
G90	ABSOLUTE COMMEND
G91	INCREMENT COMMEND
G92	SETTING OF WORK COORDICATE SYSTEM
G94	FEED UNIT SETTING (mm/min.)
G95	FEED UNIT SETTING (mm/rev.)
G96	CONSTANT LINEAR VELOCITY CONTROL ON SURFACE
G97	CANCEL CONSTANT LINEAR VELOCITY CONTROL ON SURFACE