ICAM Foundation Webinar
Macro Programming

Presented by:

Alexandre Gordon

Daniel Wang

ICAM Foundation Webinar: Modules

1. ICAM Macro Language
2. String Formatting
3. User Defined Syntax Macro Introduction

MODULE 1 : The ICAM macro language - Fundamentals

MODULE1:APT Instructions

- **APT** : **A**utomatically **P**rogrammed **T**ool
- **APT File** : a set of manufacturing instructions
- Each **APT line** represent **one** manufacturing instruction
- **APT** contains tool paths (or cutter location)
- First CAM system generation was **APT**-based

Example – Samples of APT Instructions

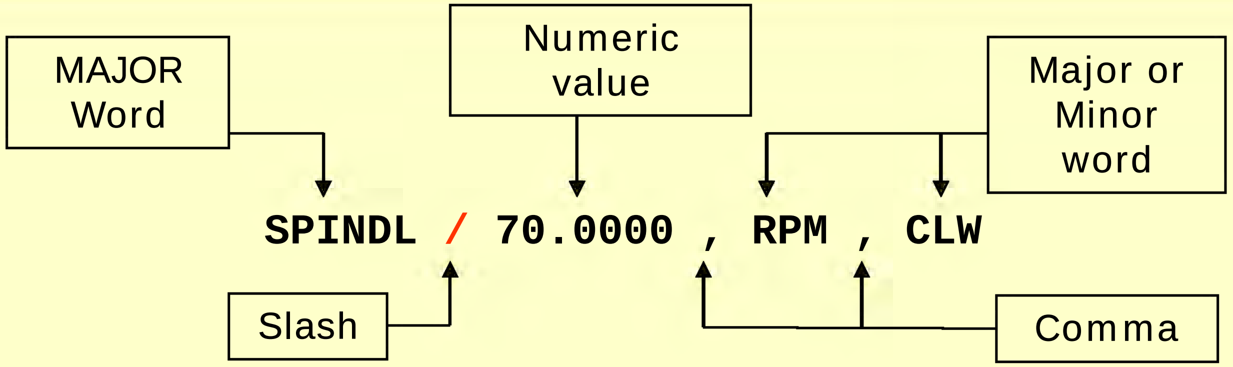
```
                                                    I, J, K
LOADTL / 5
COOLNT / FLOOD
FEDRAT / 11.8110, IPM
GOTO   / 1,2,3
GOTO   / 3.45500, 3.05000, -1.33800, 0.70710, 0.70710, 0.00000
PPRINT / 'This is an APT command'
CLAMP  / BAXIS, ON
SPINDL / 70.0000, RPM, CLW
MODE   / CONTUR, ON
```

Common Error
GOTO instructions are referenced to part origin, not the machine origin!

MODULE 1: APT Instructions

MAJOR word	: Main instruction subject (Only one per line)
MINOR word	: Arguments of the main instruction subject
/	: Separation between the major word and its arguments
,	: Separation between arguments

Example – Translation of APT instruction in machine instruction (G Code)



In ISO, this APT command turns on the spindle clockwise at 70.0000 RPM. On most ISO controllers, a post-processor would translate this as:

SPINDL / 70.0000 , RPM , CLW $\longrightarrow$ **S70 M3**

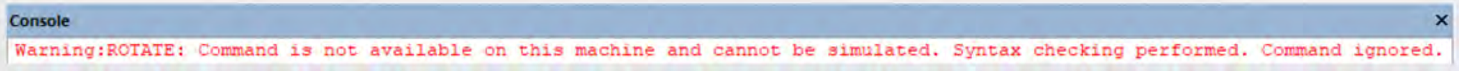
MODULE 1 : Macros - Introduction

There are four main applications for macros

1) Supporting CAM aptsource syntax to CAM-POST standard

Without Macro

ROTATE/CAXIS,90

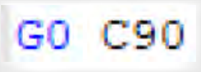


Console

```
Warning:ROTATE: Command is not available on this machine and cannot be simulated. Syntax checking performed. Command ignored.
```

With Macro

ROTATE/CAXIS,90



```
G0 C90
```

MODULE 1 : Macros - Introduction

2) Modifying the commands functions of the CL file:

Without Macro:
SPINDL / OFF $\Longrightarrow$ M5

With Custom Macro:
SPINDL / OFF $\Longrightarrow$ M9
M5

3) Support non-standard features or functions:
Example: MILL / … , TURN / … PROBE / …

4) Realize impossible output from Questionnaire:

```
Rapid
GOTO 1,1,-25
…
RAPID
GOTO 2,2,2
```

$\Longrightarrow$

```
X1 Y1
Z-25
…
X2 Y2 Z2
```

Can't be done with the Questionnaire

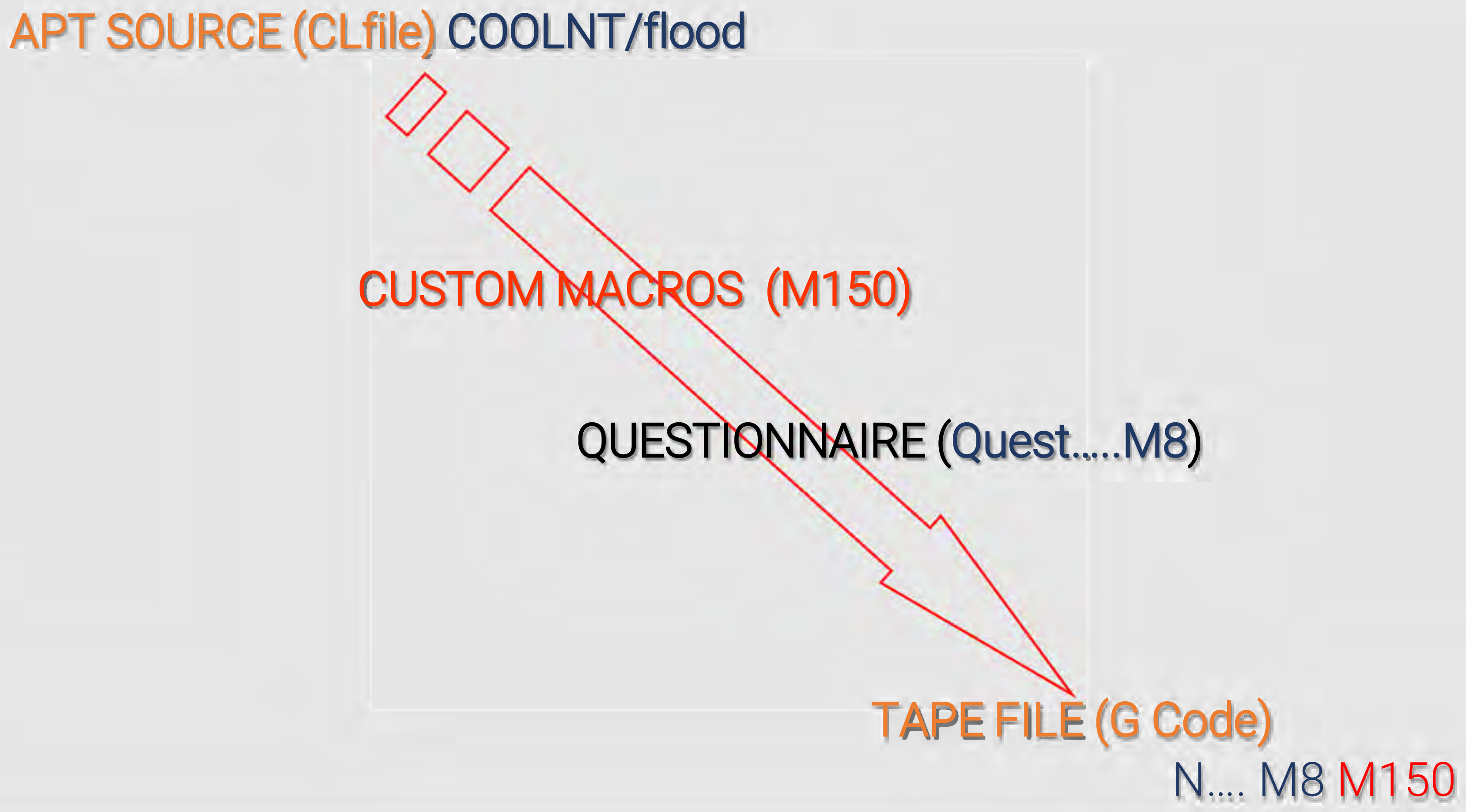
$\Longrightarrow$

```
X1 Y1
Z-25
…
X2 Y2
Z2
```

Can be done with macro

MODULE 1 : Macro - Processing order

CAM-POST Processing order


APT SOURCE (CLfile) COOLNT/flood
CUSTOM MACROS (M150)
QUESTIONNAIRE (Quest.....M8)
TAPE FILE (G Code)
N.... M8 M150

MODULE 1 : Macros - Introduction

CAM-POST macro language is similar to most programming languages. It uses the following types of data:

- VARIABLES
- FUNCTIONS
- BRANCHING CONTROL (IF, CASE)
- LOOPING CONTROL (WHILE, DO, REPEAT)

MODULE 1 : Macro data types

- **REAL** – any numeric value (integer or fractional)

  Examples: 0
  15
  0.2253

- **STRING** – any character string (including an empty string)

  Examples: ‘ ‘
  '0.2253’
  'This is a string’

- **LOGICAL** – either *true* or *false* (system constants)

  Examples: $TRUE
  $FALSE

- **RECORD** – any major word

  Examples: GOTO
  COOLNT
  FEDRAT

- **KEYWORD** – any minor word

  Examples: RPM
  FLOOD
  CCLW

- **SEQUENCE** – an ordered set of data elements of any type

  Examples: {RPM,1200,CLW,MAXRPM,4000}
  {ON}
  {}

MODULE 1 : Macro variables

Variable Type	Variable Format	Example
User Defined	AZ, 0…9, _	MY_VAR01
Predefined Global	%Gnn	%G1,%G05,%G23
Predefined Local	%Lnn	%L2,%L03,%L23
$P argument	$Pnn	$P1,$P02,$P23
System	$name	$XC,$PI,$T

MODULE 1 : Explicit variable declaration

DECLAR / *variable_scope*, *variable_type*, *variable_name(s)* [= *initial_value(s)*]

Examples:

```
DECLAR/GLOBAL,REAL,FIRST_TOOL=0
DECLAR/GLOBAL,STRING,OP_NAME,OP_COMMENTS='N/A'
DECLAR/LOCAL,TMP
```

- *variable_scope* : GLOBAL or LOCAL
- *variable_type* : REAL, STRING, LOGICAL, SEQUENCE, RECORD or KEYWORD
- *variable_name* :
  - ❑ max. 32 characters
  - ❑ only alphanumeric characters and underscore
  - ❑ must begin with a letter
- *variable_assignment* (optional): initial value preceded by an equal sign

Array variables:

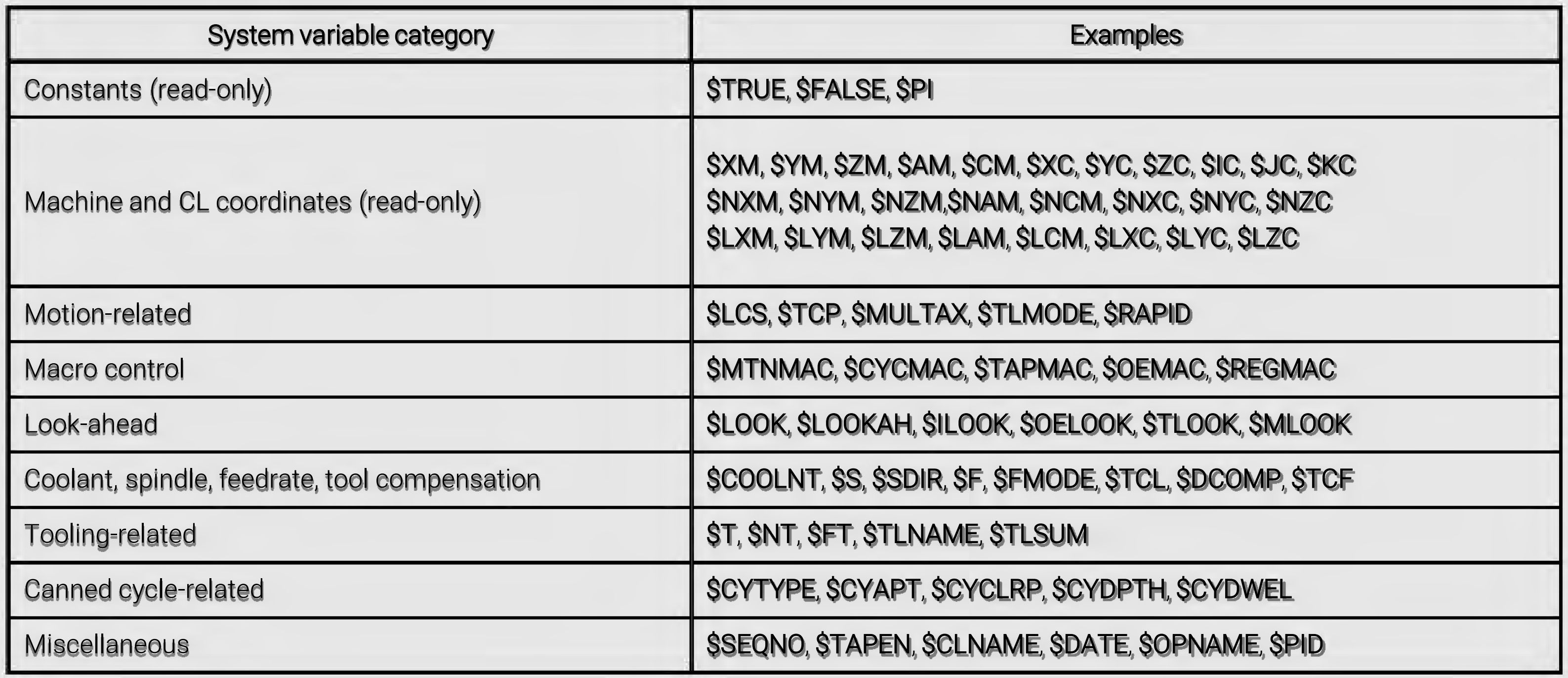
- ❑ number of array elements specified within parentheses
- ❑ cannot be initialized on the declaration line

Example:

```
DECLAR/GLOBAL,REAL,TOOLS_USED(50)
```

MODULE 1 : System variables

- Examples:



System variable category	Examples
Constants (read-only)	$TRUE, $FALSE, $PI
Machine and CL coordinates (read-only)	$XM, $YM, $ZM, $AM, $CM, $XC, $YC, $ZC, $IC, $JC, $KC $NXM, $NYM, $NZM,$NAM, $NCM, $NXC, $NYC, $NZC $LXM, $LYM, $LZM, $LAM, $LCM, $LXC, $LYC, $LZC
Motion-related	$LCS, $TCP, $MULTAX, $TLMODE, $RAPID
Macro control	$MTNMAC, $CYCMAC, $TAPMAC, $OEMAC, $REGMAC
Look-ahead	$LOOK, $LOOKAH, $ILOOK, $OELOOK, $TLOOK, $MLOOK
Coolant, spindle, feedrate, tool compensation	$COOLNT, $S, $SDIR, $F, $FMODE, $TCL, $DCOMP, $TCF
Tooling-related	$T, $NT, $FT, $TLNAME, $TLSUM
Canned cycle-related	$CYTYPE, $CYAPT, $CYCLRP, $CYDPTH, $CYDWEL
Miscellaneous	$SEQNO, $TAPEN, $CLNAME, $DATE, $OPNAME, $PID

MODULE 1 : Operators

- Numeric operators:

Operator	Function	Examples
+	Addition	%G00+%L21
-	Subtraction	%G01–2
*	Multiplication	$P1*10
/	Division	%L00/5
**	Exponentiation	%L00**2
//	String concatenation	%L12//'.TXT'
:	Sub-string / sub-sequence	$P2(1:3)
( )	Grouping	%L01/(%L02+1)
{ }	Sequence	{$P1,$P2,$P3}

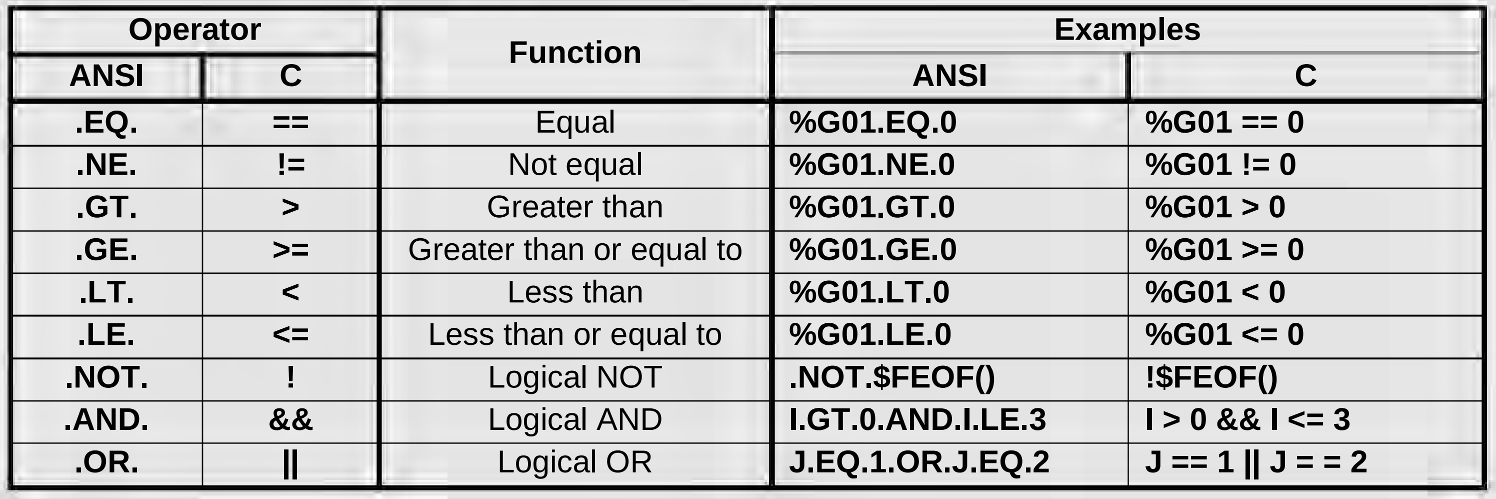
MODULE 1 : Operators

- Assignment operators:

Operator	Function	Examples
=	Assignment	%G00=10.5
+=	Addition assignment	I+=1 (same as I=I+1)
-=	Subtraction assignment	I-=1 (same as I=I-1)
*=	Multiplication assignment	%L01*=2 (same as %L01=%L01*2)
/=	Division assignment	%G01/=2 (same as %G01=%G01/2)

MODULE 1 : Operators

- Logical (*Boolean*) operators:



Operator		Function	Examples	
ANSI	C		ANSI	C
.EQ.	==	Equal	%G01.EQ.0	%G01 == 0
.NE.	!=	Not equal	%G01.NE.0	%G01 != 0
.GT.	>	Greater than	%G01.GT.0	%G01 > 0
.GE.	>=	Greater than or equal to	%G01.GE.0	%G01 >= 0
.LT.	<	Less than	%G01.LT.0	%G01 < 0
.LE.	<=	Less than or equal to	%G01.LE.0	%G01 <= 0
.NOT.	!	Logical NOT	.NOT.$FEOF()	!$FEOF()
.AND.	&&	Logical AND	I.GT.0.AND.I.LE.3	I > 0 && I <= 3
.OR.	||	Logical OR	J.EQ.1.OR.J.EQ.2	J == 1 || J = = 2

MODULE 1 : Functions

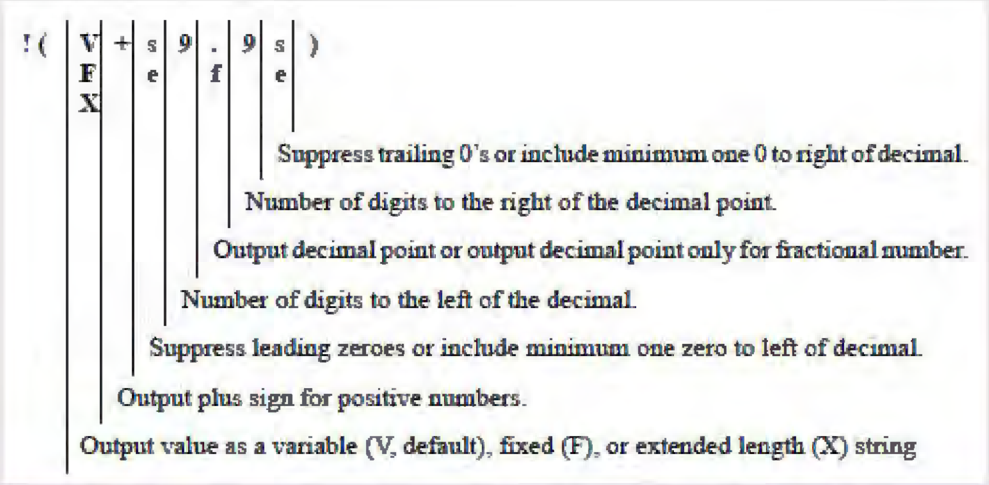
- Examples:

Function category	Examples
Mathematical functions	$FSIN, $FCOS, $FSQRT, $FLN
Numeric functions	$FABS, $FFRAC, $FINT, $FMOD, $FMAX, $FMIN
Geometric, vector and matrix functions	$FGPLPT3, $FVCROSS, $FVDOT, $FMX, $FMXMULT
Conversion functions	$FATOF, $FCVINT, $FCVREAL
Character string and sequence functions	$FMATCH, $FEDIT, $FLEN, $FTOUPER, $FTOLOWR
File and directory functions	$FACCESS, $FDIRNAM, $FBASNAM, $FEOF
CL data parsing functions	$FGET, $FCL, $FSIZE, $FGETARG, $FCLASS, $FSUBCL
Miscellaneous functions	$FDIALOG, $FINFO, $FLOOK, $FDIST, $FDK, $FIK

MODULE 2 : The ICAM macro language – Output string formatting

2.1. Numeric formats
2.2. String formats
2.3. Minor word formats
2.4. Logical formats
2.5. Wildcard formats
2.6. Specific register formats
2.7. Specific register values

MODULE 2 : Numeric formats

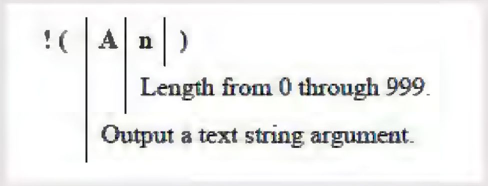


! (V F X + s e 9 . f 9 s e)
Suppress trailing 0's or include minimum one 0 to right of decimal.
Number of digits to the right of the decimal point.
Output decimal point or output decimal point only for fractional number.
Number of digits to the left of the decimal.
Suppress leading zeroes or include minimum one zero to left of decimal.
Output plus sign for positive numbers.
Output value as a variable (V, default), fixed (F), or extended length (X) string

Example:

String format	Argument=**12.345**	Argument=**0.52**	Argument=**10**	Argument=**0.0001**
'X!(s3.4s)'	X12.345	X.52	X10.	X.0001
'X!(+s3.4s)'	X+12.345	X+.52	X+10.	X+.0001
'X!(3.4)'	X012.3450	X000.5200	X010.0000	X000.0001
'X!(e3.4s)'	X12.345	X0.52	X10.	X0.0001
'X!(s3.4e)'	X12.345	X.52	X10.0	X.0001
'X!(e3.4e)'	X12.345	X0.52	X10.0	X0.0001
'X!(s3f4s)'	X12.345	X0.52	X10	X0.0001
'X!(s34)'	X123450	X5200	X100000	X1
'X!(s3)'	X12	X1	X10	X0
'X!(3)'	X012	X001	X010	X000
'X!(X+s3.4s)'	X+ 12.345	X+ .52	X+ 10.	X+ .0001
'X!(F+s3.4s)'	X+12.345	X +.52	X+10.	X +.0001

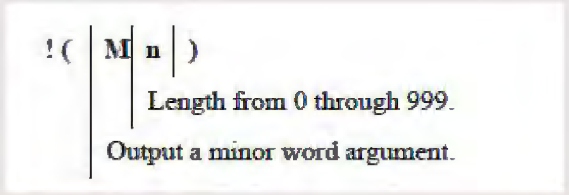
MODULE 2 : String formats



Example:

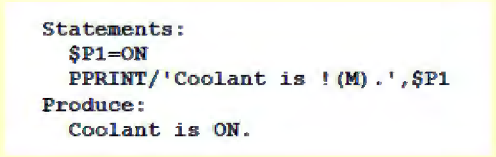
String format	Argument='ABC'	Argument='abc'	Argument='abcdef'
`'***!(A)***'`	`'***ABC***'`	`'***abc***'`	`'***abcdef***'`
`'***!(a)***'`	`'***abc***'`	`'***abc***'`	`'***abcdef***'`
`'***!(A4)***'`	`'***ABC ***'`	`'***abc ***'`	`'***abcd***'`
`'***!(A8)***'`	`'***ABC     ***'`	`'***abc     ***'`	`'***abcdef  ***'`

MODULE 2 : Minor word and logical formats

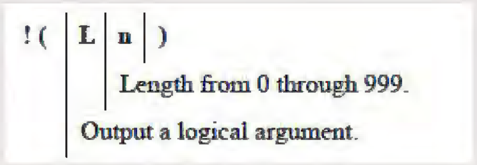


```
!( | M | n | )
   |   | Length from 0 through 999.
   | Output a minor word argument.
```

Example:

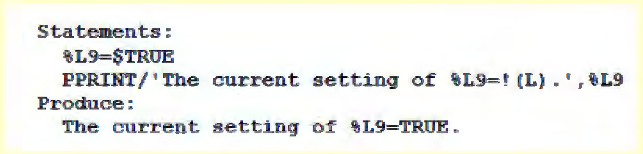


```
Statements:
  $P1=ON
  PPRINT/'Coolant is !(M).',$P1
Produce:
  Coolant is ON.
```



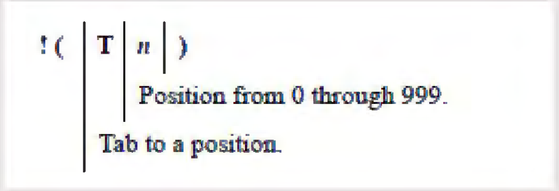
```
!( | L | n | )
   |   | Length from 0 through 999.
   | Output a logical argument.
```

Example:



```
Statements:
  %L9=$TRUE
  PPRINT/'The current setting of %L9=!(L).',%L9
Produce:
  The current setting of %L9=TRUE.
```

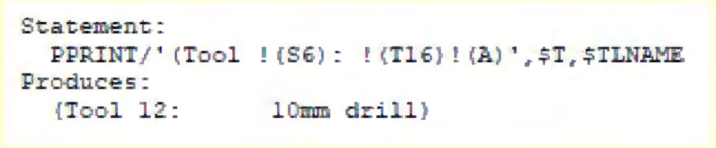
MODULE 2 : TAB and wildcard formats

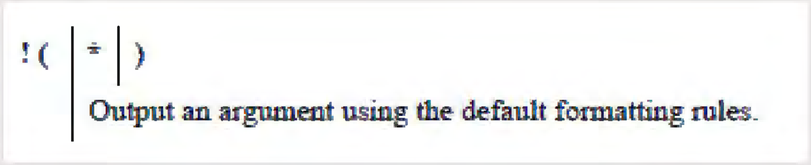

!(T *n*)

Position from 0 through 999.

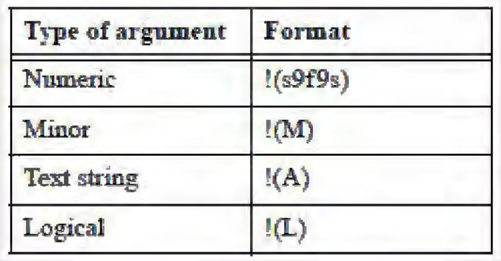
Tab to a position.

Example:


```
Statement:
  PPRINT/'(Tool !(S6): !(T16)!(A)',$T,$TLNAME
Produces:
  (Tool 12:      10mm drill)
```

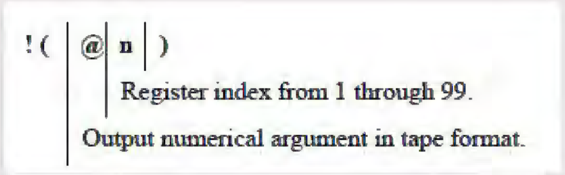

!(*)

Output an argument using the default formatting rules.

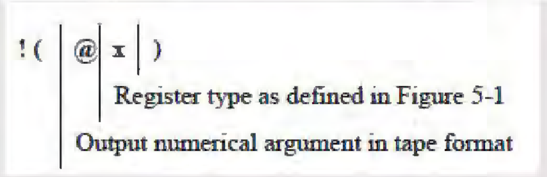


Type of argument	Format
Numeric	!(s9f9s)
Minor	!(M)
Text string	!(A)
Logical	!(L)

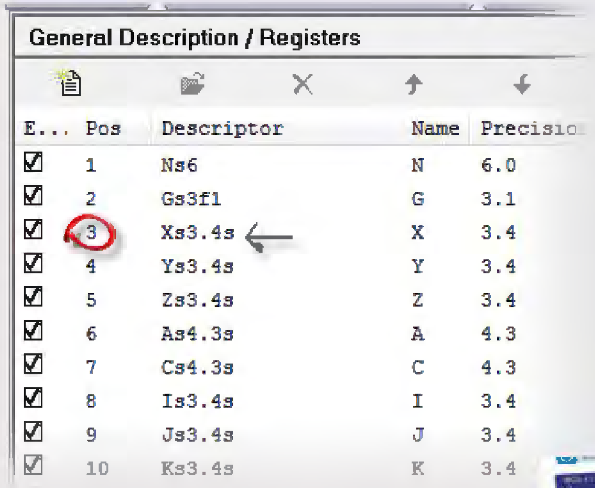
MODULE 2 : Using Registers formats



```
!( | @ | n | )
         Register index from 1 through 99.
   Output numerical argument in tape format.
```

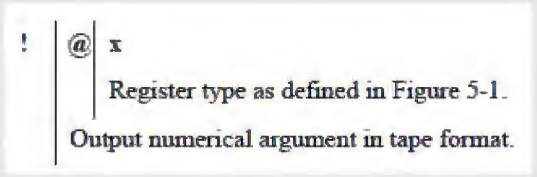


```
!( | @ | x | )
         Register type as defined in Figure 5-1
   Output numerical argument in tape format
```



General Description / Registers

E...	Pos	Descriptor	Name	Precisio
☑	1	Ns6	N	6.0
☑	2	Gs3f1	G	3.1
☑	3	Xs3.4s	X	3.4
☑	4	Ys3.4s	Y	3.4
☑	5	Zs3.4s	Z	3.4
☑	6	As4.3s	A	4.3
☑	7	Cs4.3s	C	4.3
☑	8	Is3.4s	I	3.4
☑	9	Js3.4s	J	3.4
☑	10	Ks3.4s	K	3.4



```
! | @ | x
        Register type as defined in Figure 5-1.
  Output numerical argument in tape format.
```

Example:

```
INSERT/'!(@3)',$XM
INSERT/'!(@X)',$XM        will generate:   X3.6833
INSERT/'!@X'
```

MODULE 3: User-defined Syntax macros

MODULE 3 : Macro – Processing order

CAM-POST Processing order

APT SOURCE (CLfile)

MACRO BEFORE (Kit)

CUSTOM MACROS (UDSM)

MACRO AFTER (Kit)

This type of MACRO is active if the CL file syntax is a match to a user defined macro. The first line of the MACRO defines the syntax required to initiate the programmed action of this macro.

STARTUP/…PROCEDURES MACROS

QUESTIONNAIRE (Quest)

…/ SHUTDOWN PROCEDURES MACROS

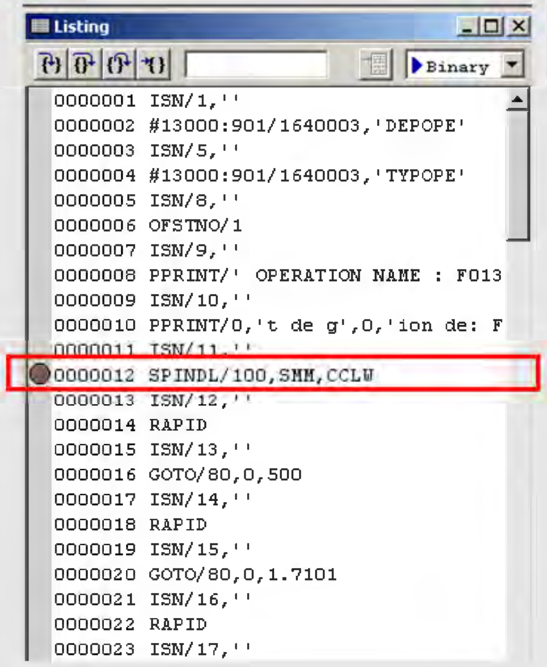
Register MACROS

TAPE MACROS

TAPE EDIT

TAPE FILE (G Code)

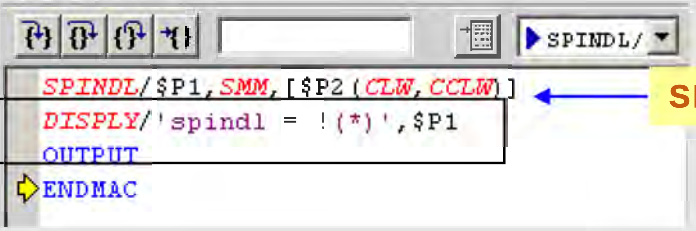
MODULE 3 : Macro – Processing order



```
0000001 ISN/1,''
0000002 #13000:901/1640003,'DEPOPE'
0000003 ISN/5,''
0000004 #13000:901/1640003,'TYPOPE'
0000005 ISN/8,''
0000006 OFSTNO/1
0000007 ISN/9,''
0000008 PPRINT/' OPERATION NAME : F013
0000009 ISN/10,''
0000010 PPRINT/0,'t de g',0,'ion de: F
0000011 ISN/11,''
0000012 SPINDL/100,SMM,CCLW
0000013 ISN/12,''
0000014 RAPID
0000015 ISN/13,''
0000016 GOTO/80,0,500
0000017 ISN/14,''
0000018 RAPID
0000019 ISN/15,''
0000020 GOTO/80,0,1.7101
0000021 ISN/16,''
0000022 RAPID
0000023 ISN/17,''
```

UDSM

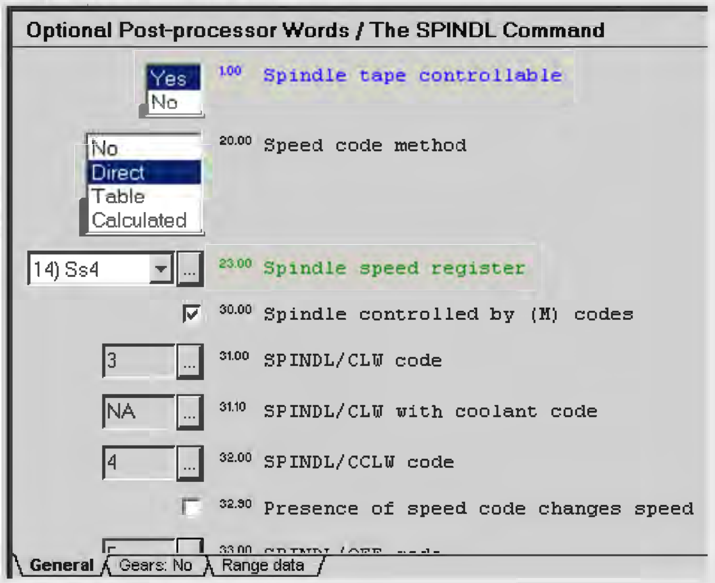
Existing Macro: YES



```
SPINDL/$P1,SMM,[$P2(CLW,CCLW)]
DISPLY/'spindl = !(*)',$P1
OUTPUT
ENDMAC
```

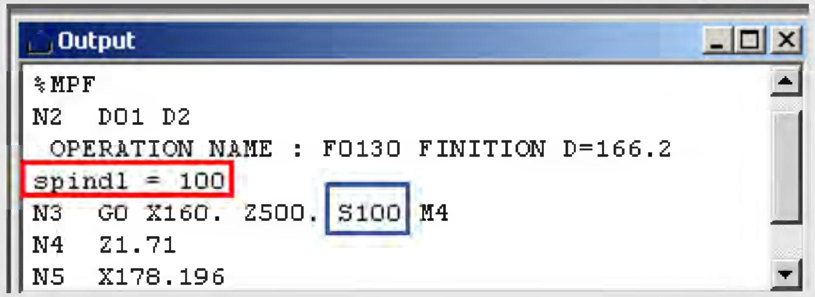
SDL

Questionnaire



Optional Post-processor Words / The SPINDL Command

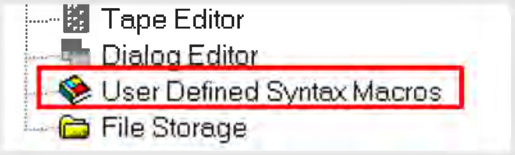
[✓ Yes] No	1.00	Spindle tape controllable
No [✓ Direct] Table Calculated	20.00	Speed code method
14) Ss4	23.00	Spindle speed register
✓	30.00	Spindle controlled by (M) codes
3	31.00	SPINDL/CLW code
NA	31.10	SPINDL/CLW with coolant code
4	32.00	SPINDL/CCLW code
☐	32.30	Presence of speed code changes speed



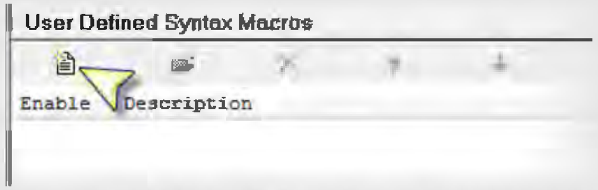
```
%MPF
N2  D01 D2
 OPERATION NAME : F0130 FINITION D=166.2
spindl = 100
N3  G0 X160. Z500. S100 M4
N4  Z1.71
N5  X178.196
```

MODULE 3 : Adding user-defined syntax macros to a post

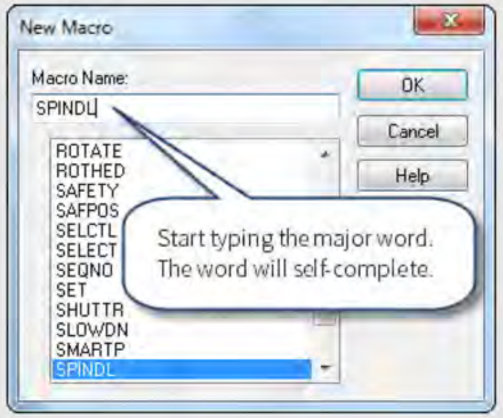
- Select *Post-processor Customization → User Defined Syntax Macros*



- Click *Add* button

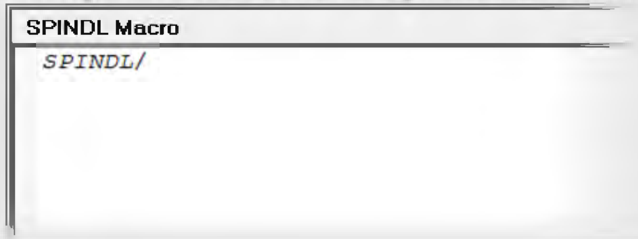


- Start typing the major word (e.g. SPINDL) the macro will be built on. It will self-complete.



Start typing the major word. The word will self-complete.

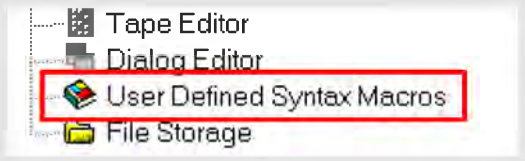
- The ICAM macro editor opens with the major word on the Syntax Definition Line. Add the required and optional arguments.

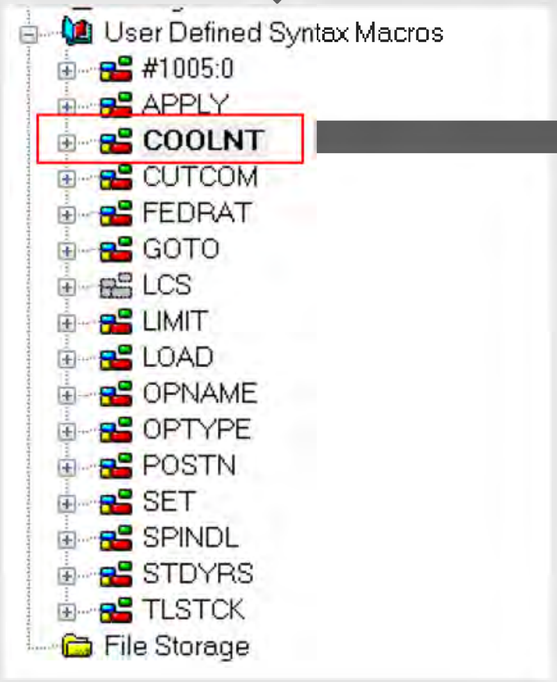


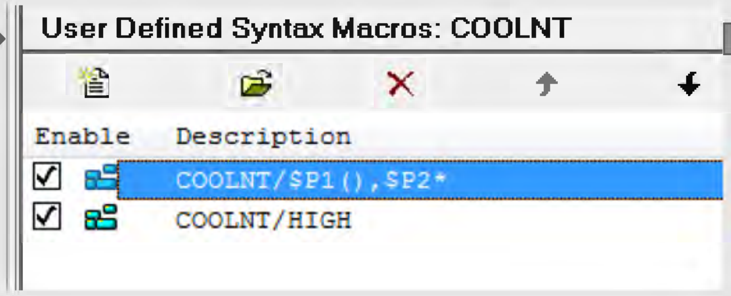
```
SPINDL/
```

MODULE 3 : The Syntax Definition Line (SDL)

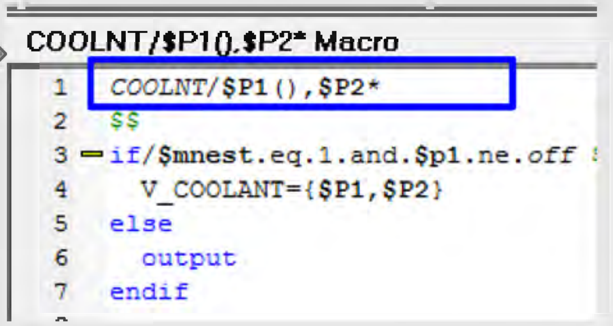
- The SDL is used to *match* a “target” CL command about to be processed
- The SDL is the first non-comment line in any user-defined macro
- The remaining macro lines (i.e. the *body*) contain the actions to be taken when matching occurs
- A user-defined macro *must* have an SDL but *may not* have a body






User Defined Syntax Macros: COOLNT

Enable	Description
✓	COOLNT/$P1(),$P2*
✓	COOLNT/HIGH


COOLNT/$P1(),$P2* Macro

```
COOLNT/$P1(),$P2*
$$
if/$mnest.eq.1.and.$p1.ne.off
  V_COOLANT={$P1,$P2}
else
  output
endif
```

MODULE 3 : The Syntax Definition Line (SDL)

```
MAJOR WORD /  argument1, (argument n1, …),  [argument n2, …]
```

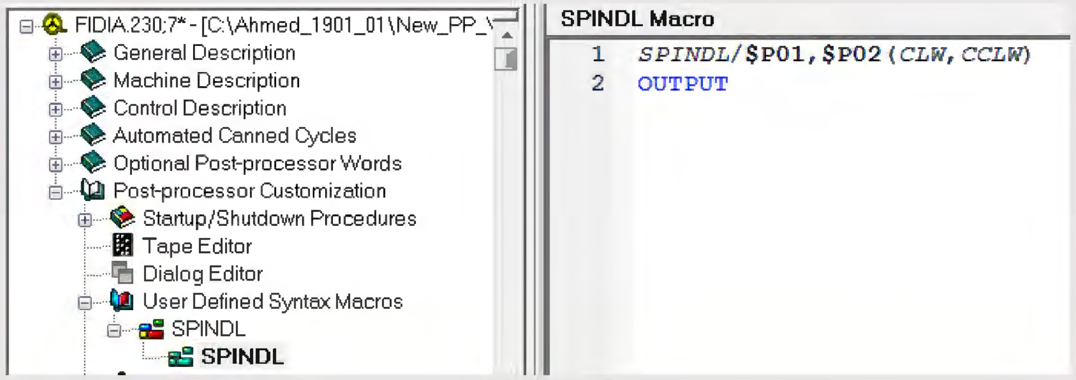
- The SDL always begins with a *major word*, optionally followed by a slash "/" and zero or more comma-separated arguments (e.g. COOLNT/ON,LOW)
- Arguments can be required (always present) or optional (may or may not be present)
  - ❑ Arguments between (…) MUST appear
  - ❑ Arguments between […] are OPTIONAL and may appear in ANY order
- Arguments can be minor words, numbers or strings
- An SDL with no arguments (e.g. RAPID, GOHOME, OPSTOP, etc.) matches the major word only if coded without arguments
- A user-defined macro *must* have an SDL but *may not* have a body

MODULE 3 : The Syntax Definition Line (SDL)

- SDL arguments are either $P variables (matching anything) or hard-coded minor words (matching only themselves). Example: **FEDRAT/IPM,$P1**
- $P variables take the value of the "matched" arguments on the actual CL record
- The type of each $P argument is indicated by optional characters immediately after the $P number
- These characters are:
  - ❑ open and closed parentheses ()
  - ❑ two single quotes or apostrophes ''
  - ❑ a question mark ?
  - ❑ an asterisk *

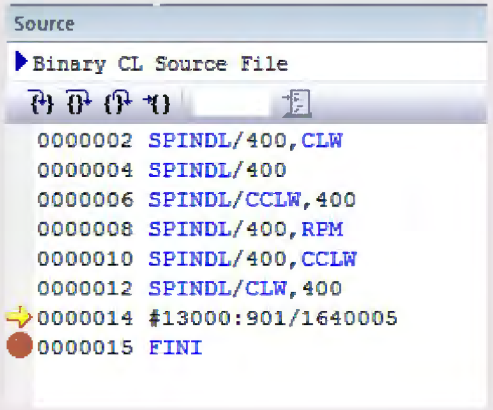
Form	Examples	Description
minor_word	CLW	Matches the minor word specified
$P*n*	$P4	Matches any number
$P*n*()	$P5()	Matches any minor word
$P*n*(*minor_word*)	$P1(BRKCHP)	Matches the minor word listed
$P*n*(*minor_word,minor_word,...*)	$P1(ON,FLOOD,THRU)	Matches any of the minor words listed
$P*n*''	$P1''	Matches any string
$P*n*?	$P12?	Matches any single argument of any type
$P*n**	$P6*	Matches zero or more arguments of any type

MODULE 3 : Example 1 – Required SDL arguments



FIDIA.230;7* - [C:\Ahmed_1901_01\New_PP_\
General Description
Machine Description
Control Description
Automated Canned Cycles
Optional Post-processor Words
Post-processor Customization
Startup/Shutdown Procedures
Tape Editor
Dialog Editor
User Defined Syntax Macros
SPINDL
SPINDL
SPINDL Macro

```
SPINDL/$P01,$P02(CLW,CCLW)
OUTPUT
```



```
0000002 SPINDL/400,CLW
0000004 SPINDL/400
0000006 SPINDL/CCLW,400
0000008 SPINDL/400,RPM
0000010 SPINDL/400,CCLW
0000012 SPINDL/CLW,400
0000014 #13000:901/1640005
0000015 FINI
```

Syntax Definition Line (SDL):

SPINDL/$P01,$P02(CLW,CCLW)

Post-processor commands:

SPINDL/ 400, CLW ………

SPINDL/ 400 ………

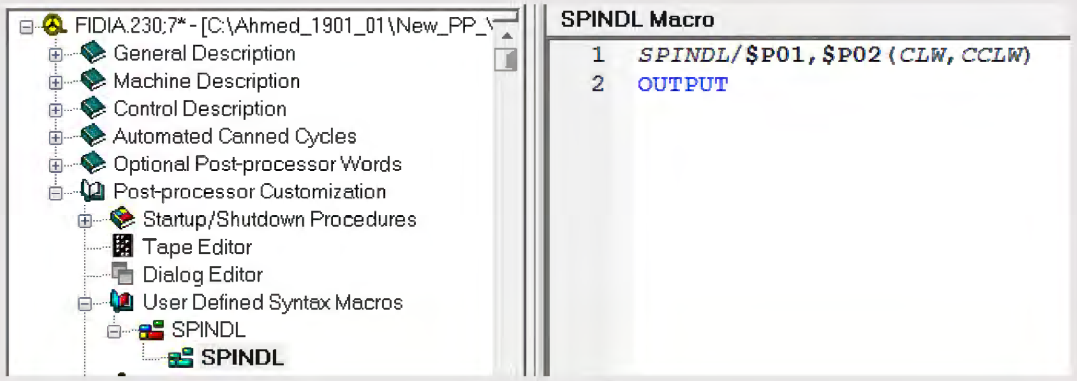
SPINDL/ CCLW, 400 ………

SPINDL/ 400, RPM ………

SPINDL/ 400, CCLW ………

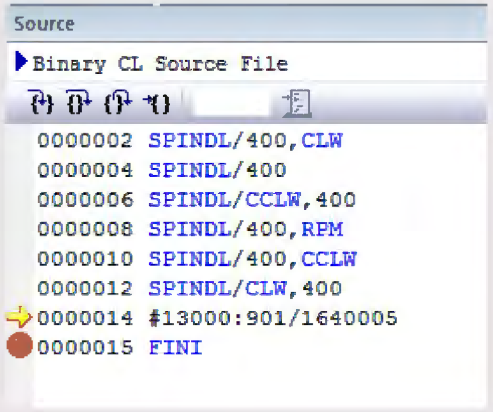
SPINDL/ CLW, 400 ………

MODULE 3 : Example 1 – Required SDL arguments


FIDIA.230;7* - [C:\Ahmed_1901_01\New_PP_\
- General Description
- Machine Description
- Control Description
- Automated Canned Cycles
- Optional Post-processor Words
- Post-processor Customization
  - Startup/Shutdown Procedures
  - Tape Editor
  - Dialog Editor
  - User Defined Syntax Macros
    - SPINDL
      - SPINDL

SPINDL Macro

```
SPINDL/$P01,$P02(CLW,CCLW)
OUTPUT
```


Source

Binary CL Source File

```
0000002 SPINDL/400,CLW
0000004 SPINDL/400
0000006 SPINDL/CCLW,400
0000008 SPINDL/400,RPM
0000010 SPINDL/400,CCLW
0000012 SPINDL/CLW,400
0000014 #13000:901/1640005
0000015 FINI
```

Syntax Definition Line (SDL):

SPINDL/$P01,$P02(CLW,CCLW)

Post-processor commands:

SPINDL/ 400, CLW	……… ✓
SPINDL/ 400	……… ✗
SPINDL/ CCLW, 400	……… ✗
SPINDL/ 400, RPM	……… ✗
SPINDL/ 400, CCLW	……… ✓
SPINDL/ CLW, 400	……… ✗

MODULE 3: Example 2 – Writing a correct SDL

Post-processor commands:

```
LOADTL / 1, OSETNO, 1
LOADTL / 1,LENGTH, 12.0,MANUAL
LOADTL / 1, MANUAL
LOADTL / 2, OSETNO, 2, LARGE
LOADTL / 4, OSETNO, 4, SMALL
LOADTL / 5, LENGTH, 100
```

Required Syntax Definition Line (SDL):

```
LOADTL / …………………………………………………………………………………………………………………
```

MODULE 3 : Example 2 – Writing a correct SDL

Post-processor commands:

LOADTL	1	OSETNO	1				
LOADTL	1			LENGTH	12.0	MANUAL	
LOADTL	1					MANUAL	
LOADTL	2	OSETNO	2				LARGE
LOADTL	4	OSETNO	4				SMALL
LOADTL	5			LENGTH	100		

Required Syntax Definition Line (SDL):

LOADTL / ……………………………………………………………………………………………………

MODULE 3 : Example 2 – Writing a correct SDL

Post-processor commands:

LOADTL	1	OSETNO	1				
LOADTL	1			LENGTH	12.0	MANUAL	
LOADTL	1					MANUAL	
LOADTL	2	OSETNO	2				LARGE
LOADTL	4	OSETNO	4				SMALL
LOADTL	5			LENGTH	100		

Required Syntax Definition Line (SDL):

LOADTL / $P1, [OSETNO,$P2], [LENGTH,$P3], [MANUAL], [$P4(SMALL,LARGE)]

MODULE 3 : The OUTPUT command

- Used to instruct GENER to process the trapped CL record "*as-is*"
- Example:

```
SPINDL/ON        |   SPINDL/ON
AIR/ON           |   AIR/ON
SPINDL/ON        |   OUTPUT
```

(both macros have the same results)

- Any modifications made to $P arguments are ignored
- OUTPUT *should not* be used when the trapped CL record is required to be modified in any way
- Example:

```
SPINDL/RPM,$P01,$P02(CLW,CCLW)
IF/$P01.LE.$MAXRPM
   OUTPUT
ELSE
   SPINDL/RPM,$MAXRPM,$P02
ENDOF/IF
```

- Variation:

```
SPINDL/RPM,$P01,$P02(CLW,CCLW)
SPINDL/RPM,$FIF($P01.LE.$MAXRPM,$P01,$MAXRPM),$P02
```

MODULE 3 : The TERMAC command

- Used to **exit** a macro
- There is an implicit **TERMAC** at the end of each macro (no need to end a macro with **TERMAC**)
- Can be used to *force* the current macro to exit prematurely when a specific condition occurs
- Can be used in any type of macro (user-defined or Startup/Shutdown)
- Example:

```
PROBE/$P1(ON,OFF)
IF/$FMATCH($TLTAB(20,$TI),'PROBE').EQ.''
  TERMAC
ENDOF/IF
INSERT/$FIF($P1.EQ.ON,'O9832','O9833')
```