

Post pro

C5 - Virtualiser un produit mécanique ou un process du concept au jumeau numérique selon les besoins de l'usine du futur - Niveau initial - Virtualiser dans un contexte mono-disciplinaire

AC35.03 - Echanger des données entre différents systèmes numériques

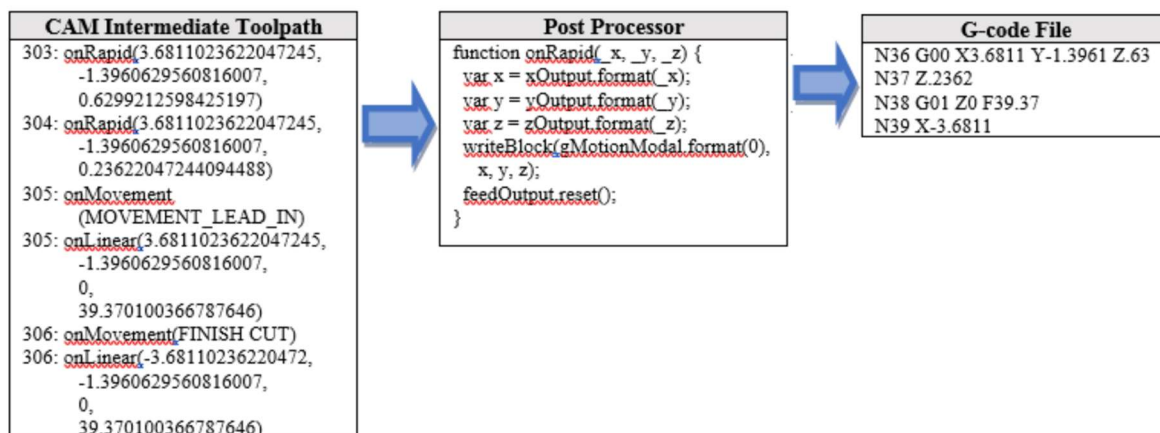
AC35.04 - Comprendre les couplages réel/virtuel, virtuel/réel (calibration, ajustement physique & virtuel ...) et les jumeaux numériques

1. Qu'est-ce qu'un post pro ?

Le post pro fait le lien entre la FAO et la machine CN.

La FAO génère généralement un fichier intermédiaire neutre contenant des informations sur chaque opération de parcours d'outil comme les données d'outil, le type d'opération (perçage, fraisage, tournage, etc.) et les données du parcours de l'outil.

Ce fichier intermédiaire est introduit dans le post-processeur où il est traduit dans le langage compris par une machine CN. Dans la plupart des cas, ce langage est une forme d'ISO/EIA, un G-code standard.



Le Post pro est constitué, entre autres, de fonctions qui vont traduire les données du logiciel de FAO en code machine

```
C: > Users > xwall2 > Desktop > C 1001.h
```

```
1 0 BEGIN PGM 1001 MM
2 1 BLK FORM 0.1 Z X-50 Y-30 Z-60
3 2 BLK FORM 0.2 X+50 Y+30 Z+0
4 3 ;-----
5 4 ;Outils
6 5 ; #1 D=20 - fraise deux tailles
7 6 ; #2 D=10 TAPER=118deg - foret
8 7 ;-----
9 8 ;
10 9 PLANE RESET STAY
11 10 * - 2D Contour1
12 11 M5
13 12 TOOL CALL 1 Z S5000
14 13 TOOL DEF 2
15 14 L M140 MB MAX
16 15 M126
17 16 M3
18 17 CYCL DEF 9.0 DWELL TIME
19 18 CYCL DEF 9.1 DWELL 3
20 19 LBL 1
21 20 CYCL DEF 247 DATUM SETTING ~
22 | Q339=54 ; DATUM NUMBER
23 21 LBL 0
24 22 PLANE RESET TURN FMAX
25 23 L X=24 Y=20 FMAX
```

Outils		
Outil T2 TL Type: fraise deux tailles Longueur: 30mm Diamètre: 10mm Longueur de coupe: 20mm Pointes: 3 Déscriptions: fraise 2 tailles 420	Découpe VITESSE DE BROCHES MAXIMUM: 5000tr/min CUTTING DIRECTION: 311 95mm FEEDING RATE: 10mm EXTENDED CUT TIME: 21s (38.0%)	Porte-outil 
Outil T2 D2 Type: fraise Longueur: 15mm Angle de coupe: 11° Longueur: 130mm Longueur de coupe: 120mm Pointes: 3 Déscriptions: fraise 410	Découpe VITESSE DE BROCHES MAXIMUM: 5000tr/min CUTTING DIRECTION: 200mm FEEDING RATE: 10mm EXTENDED CUT TIME: 21s (37.0%)	Porte-outil 

2

Programme 1001					Fiche technique Outils							Date September 10, 2023 14:14		
Commentaire												Tronçonner piece post pro v5		
Tool #	Length #	Diameter #	Tool Orientation in Turret	Diamètre	Rayon de bout d'outil	Angle de dépouille	Type	Porte-outil	Longueur du corps	Distance d'avnice	Vitesse de broche maximum	Maximum Feed	Temps d'usinage	
T1	L1	D1		20.00	0.00		0 fraise deux tailles		50.00	311.96	5000	1000	0:00:21	
T2	L2	D2		10.00	0.00		118 foret		130.00	260.00	5000	1000	0:00:21	

Programme 1001				Fiche technique Opérations							Date September 10, 2023 14:14			
Commentaire											Tronçonner piece post pro v5			
ID	Opération	WCS #	Tool #	Stratégie	Temps d'usinage	Distance d'avance	Lubrifiant	Surépaisseur	Surépaisseur axiale	Tolérance	Pas en Z	Recouvrement	Vitesse de broche	Maximum Feed
1	2D Contour1	54	T1	Contour 2D	0:00:11	155.98	Jet	0.00	0.00	0.01	0.00	19.00	5000	1000
2	Drill1	54	T2	Perçage	0:00:21	260.00	Jet			0.01			5000	1000
3	2D Contour2	54	T1	Contour 2D	0:00:11	155.98	Jet	0.00	0.00	0.01	0.00	19.00	5000	1000

Ou d'autres formats pour être exploités par d'autres machines... par exemple le banc de
préréglage

[illegible]

```
C: > Users > xwaller2 > Desktop > 1001.csv
1 T;X;Z;Code;ANGLE;rayon;Xth;Xth+;Xth-;Zth;Zth+;Zth-
2 4;;; "fraise deux tailles";;;8;0.2;0.2;105;2;0.1
3 10;;; "spot drill";;;6;0.2;0.2;112;2;0.1
4 13;;; "foret";;;3.3;0.2;0.2;120;2;0.1
5
```

2. Extrait d'un post pro personnalisé pour ajouter de la sécurité par rapport à la machine et explications

```

if (spindleSpeed < 1) {
    error(localize("FREQUENCE DE ROTATION TROP BASSE!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!"));
    return;
}
if (spindleSpeed == 123) {
    error(localize("REMPLI TES CONDITIONS DE COUPE GROS NAZE!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!"));
    return;
}
if (spindleSpeed > (getProperty("robom") ? 10000 : 24000)) {
    error(localize("TU AS UN OUTIL QUI TOURNE TROP VITE, BLAIREAU!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!"));
}
if (!tapping || (tapping && !(getProperty("useRigidTapping") == "without"))) {
    writeBlock(
        sOutput.format(spindleSpeed), mFormat.format(tool.clockwise ? 3 : 4)
    );
    writeBlock(gFormat.format(4), "P3000");
}

```

3. Javascript

3.1. La syntaxe

Le langage informatique utilisé pour les post pro est du Javascript ...

Quelques explications sur le langage...

Tout d'abord la syntaxe :

La casse (minuscule/majuscule) doit être respectée !

Exemple :

currentCoolant = 7;

currentCoolant = 8;

currentcoolant = 9;

Les noms de variables ou de fonctions ne doivent pas contenir d'espace

Les fins de lignes sont toujours un ;

Il est possible de mettre des commentaires pour se repérer dans les lignes de codes

// pour passer en commentaire une ligne ou fin de ligne

/* ... */ pour passer plusieurs lignes en commentaire

```
/**
 * Output a comment.
 */
function writeComment(text) {
    writeln(formatComment(text)); // write out comment line
}
..
/*
 * switch (unit) {
 *   case IN:
 *     writeBlock(gUnitModal.format(20));
 *     break;
 *   case MM:
 *     writeBlock(gUnitModal.format(21));
 *     break;
 * }
 */
```

3.2. Ecriture et édition

Habitudes à prendre dans la présentation lors de l'écriture du code :

Définition de l'indentation en informatique : Disposition particulière du texte d'un programme faisant apparaître des décalages au niveau de la marge.

L'utilisation de l'indentation pour le contenu des fonctions, les tests conditionnels, les boucles, etc., est recommandée. Cela facilite la visualisation du code. Les caractères de tabulation sont déconseillés. Il est préférable d'utiliser des tabulations virtuelles de deux espaces pour indenter le code dans le post pro. La plupart des éditeurs, peuvent être configurés pour

convertir automatiquement les caractères de tabulation en espaces qui aligneront chaque retrait sur deux espaces.

```
function test (input) {
  // indent 2 spaces inside of function
  if (input == 1) {
    writeBlock( // indent 2 more spaces in if block or loop
      gAbsIncModal.format(90), // indent 2 more spaces for continuation lines
      gMotionModal.format(0)
    );
  }
}
```

```
function test (input) {
  // indent 2 spaces inside of function
  if (input == 1) {
    writeBlock( // indent 2 more spaces in if block or loop
      gAbsIncModal.format(90), // indent 2 more spaces for continuation lines
      gMotionModal.format(0)
    );
  }
}
```



Editeur de texte de base

3.3. Les variables

Ce sont des noms associés à des valeurs

La valeur peut être un nombre, une chaîne, un booléen, un tableau, ou un objet.

Les variables en JavaScript ne sont pas typées, ce qui signifie qu'elles sont définies par la valeur qui leur sont assignés et le type de valeur peut changer tout au long du programme. Par exemple, vous pouvez attribuer un numéro à une variable et plus tard dans le programme, vous pouvez attribuer à la même variable une valeur de chaîne.

Le mot-clé « var » est utilisé pour définir une variable.

```
var a;           // define variable 'a', it will have the value of undefined
var b = 1;       // assign a value of 1 to the variable 'b'
var c = "text";  // assign a text string to the variable 'c'
c = 2.5;         // 'c' now contains a number instead of string
```

Les variables ne peuvent pas utiliser le même nom que ceux utilisés par Javascript ou Fusion 360

Exemples : var, for, cycle, currentSection, etc...

JavaScript prend en charge à la fois les **variables globales** et les **variables locales**.

Une **variable globale** est définie en dehors d'une fonction, par exemple en haut du fichier avant de définir une fonction.

Les **variables globales** sont accessibles à toutes les fonctions du programme et auront la même valeur d'une fonction à l'autre fonction.

Les **variables locales** ne sont accessibles qu'à partir de la fonction pour laquelle elles sont définies. Il est possible d'utiliser le même nom pour les variables locales dans plusieurs fonctions et elles auront chacune leur propre valeur dans les fonctions séparées.

Exemples de variables globales

```
var gFormat = createFormat({prefix:"G", width:2, zeropad:true, decimals:1});
var mFormat = createFormat({prefix:"M", width:2, zeropad:true, decimals:1});
var hFormat = createFormat({prefix:"H", width:2, zeropad:true, decimals:1});
var xyzFormat = createFormat({decimals:(unit == MM ? 3 : 4), forceDecimal:true});
var ijkFormat = createFormat({decimals:6, forceDecimal:true}); // unitless
var rFormat = xyzFormat; // radius
var abcFormat = createFormat({decimals:3, forceDecimal:true, scale:DEG});
var feedFormat = createFormat({decimals:(unit == MM ? 0 : 1), forceDecimal:true});
var xOutput = createVariable({prefix:"X", xyzFormat});
var yOutput = createVariable({prefix:"Y", xyzFormat});
var zOutput = createVariable({onChange:function () {retracted = false;}, prefix:"Z", xyzFormat});
var feedOutput = createVariable({prefix:"F", feedFormat});
var inverseTimeOutput = createVariable({prefix:"F", force:true}, inverseTimeFormat);
var pitchOutput = createVariable({prefix:"F", force:true}, pitchFormat);
var sOutput = createVariable({prefix:"S", force:true}, rpmFormat);
var dOutput = createVariable({}, dFormat);
var peckOutput = createVariable({prefix:"Q", force:true}, peckFormat);
```

Exemples de variables locales

```
function onRapid(_x, _y, _z) {
  var x = xOutput.format(_x);
  var y = yOutput.format(_y);
  var z = zOutput.format(_z);
  if (x || y || z) {
    if (pendingRadiusCompensation >= 0) {
      error(localize("Radius compensation mode cannot be used in rapid mode"));
      return;
    }
    writeBlock(gMotionModal.format(0), x, y, z);
    forceFeed();
  }
}
```

3.4. Calculs et tests sur les expressions

P	Operator	Operands	Description
13	()	Expression	Overrides the assigned precedence of operators
12	++	Integer	Unary increment
	--	Integer	Unary decrement
	~	Integer	Unary bitwise complement
	!	Boolean	Unary logical complement (not)
11	*	Number	Multiplication
	/	Number	Division
	%	Number	Remainder
10	+	Number, String	Addition
	-	Number	Subtraction
9	<<	Integer	Bitwise shift left
	>>	Integer	Bitwise shift right
8	<	Number, String	Less than
	<=	Number, String	Less than or equal to
	>	Number, String	Greater than
	>=	Number, String	Greater than or equal to
7	==	Any	Equal to
	!=	Any	Not equal to
	===	Any	Equal to and same variable type
	!==	Any	Not equal to and same variable type

6	&	Integer	Bitwise AND
5	^	Integer	Bitwise XOR
4		Integer	Bitwise OR
3	&&	Boolean	Logical AND
2		Boolean	Logical OR
1	=	Any	Assignment
	+=	Number, String	Assignment with addition
	-=	Number	Assignment with subtraction
	*=	Number	Assignment with multiplication
	/=	Number	Assignment with division

Exemples de calculs et tests sur les expressions

x	y	Expression	Result	Expression	Result
3	5	$z = x + y * 3$	18	$z = (x + y) * 3$	24
		$z = ++x$	$z = 4, x = 4$	$z = x++$	$z = 3, x = 4$
		$x += y$	8	$x *= y$	15
		$z = y / x$	1.667	$z = y \% x$	2.0
"Start"	"-End"	$z = x + y$	"Start-End"	$x += y$	"Start-End"
2	3	$z = x \& y$	2	$z = x y$	3
1	"1"	$z = x == y$	true	$x === y$	false
true	false	$z = x$	true	$z = !y$	true
		$z = x y$	true	$z = x \&\& y$	false

3.5. Les tests conditionnels

Si... Alors... Sinon If...Then...Else

```

if (conditional1) {
    // execute code if conditional1 is true
}
if (conditional1) {
    // execute code if conditional1 is true
} else {
    // execute code if conditional1 is false
}
if (conditional1) {
    // execute code if conditional1 is true
} else if (conditional2) {
    // execute code if conditional1 is false and conditional2 is true
} else {
    // execute code if all conditionals are false
}

```

Opérateur conditionnel :

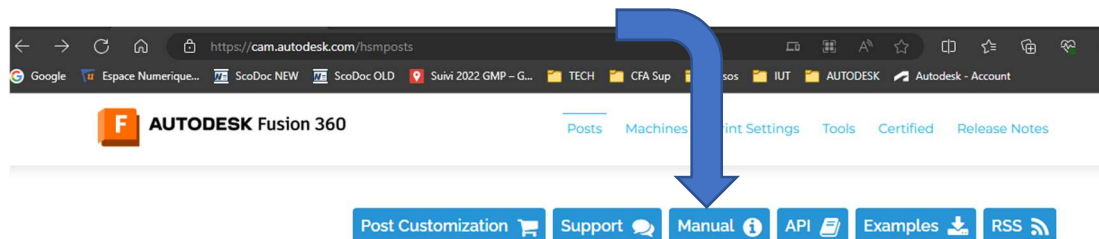
var a = conditional ? true_value : false_value;

```
var homeGcode = getProperty("useG30") ? 30 : 28;
// identique à cette version avec if
if (getProperty("useG30")) {
  homeGcode = 30;
} else {
  homeGcode = 28;
}
```

Il existe encore beaucoup d'autres possibilités offertes par le Javascript mais c'est un cours d'initiation...

Vous référer au document de référence d'Autodesk si vous souhaitez aller plus loin...

[Post Processor Training Guide \(autodesk.com\)](https://cam.autodesk.com/hsm/posts)



3.5 Conditional Statements	
3.5.1 The if Statement	
3.5.2 The switch Statement	
3.5.3 The Conditional Operator (?) ..	
3.5.4 The typeof Operator	
3.5.5 The conditional Function	
3.5.6 try / catch	
3.5.7 The validate Function	
3.5.8 Comparing Real Values	
3.6 Looping Statements	
3.6.1 The for Loop	
3.6.2 The for/in Loop	
3.6.3 The while Loop	
3.6.4 The do/while Loop	
3.6.5 The break Statement	
3.6.6 The continue Statement	

3.6. Les fonctions

Une fonction se compose de l'instruction de fonction, d'une liste d'arguments, du corps de la fonction (code JavaScript) et des instructions return facultatives.

La liste d'arguments est facultative et contient les identifiants transmis à la fonction par la routine appelante. Les arguments passés à la fonction sont considérés en lecture seule toute modification apportée à ces variables sera conservée localement par rapport à la fonction.


```
function name(arg1 ,arg2 ... , argn) {
  code
}
```

Exemple

```
function writeBlock() {
  var text = formatWords(arguments);
  if (!text) {
    return;
  }
  if (getProperty("showSequenceNumbers") == "true") {
    if (optionalSection || skipBlock) {
      if (text) {
        writeWords("/", "N" + sequenceNumber, text);
      }
    } else {
      writeWords2("N" + sequenceNumber, arguments);
    }
    sequenceNumber += getProperty("sequenceNumberIncrement");
  } else {
    if (optionalSection || skipBlock) {
      writeWords2("/", arguments);
    } else {
      writeWords(arguments);
    }
  }
  skipBlock = false;
}
```

Un appel de fonction est traité de la même manière que toute autre expression. Il peut être autonome, attribuer une valeur et être placé n'importe où dans une expression. La valeur renvoyée par la fonction appelée est traitée comme n'importe quelle autre variable. Vous tapez simplement le nom de la fonction avec ses arguments

```
if (tool.number > 99) {
  warning(localize("Tool number exceeds maximum value."));
}
writeBlock(gFormat.format(28), gAbsIncModal.format(91), "Z" + xyzFormat.format(0));
disableLengthCompensation(true);
writeBlock("G0 G90 G53 Y0");
writeBlock("T" + toolFormat.format(tool.number), mFormat.format(6));
dOutput.reset();
writeBlock(dOutput.format(d));
if (getProperty("limiterapide") ) {writeBlock("M47 C20");}
if (tool.comment) {
  writeComment(tool.comment);
}
```

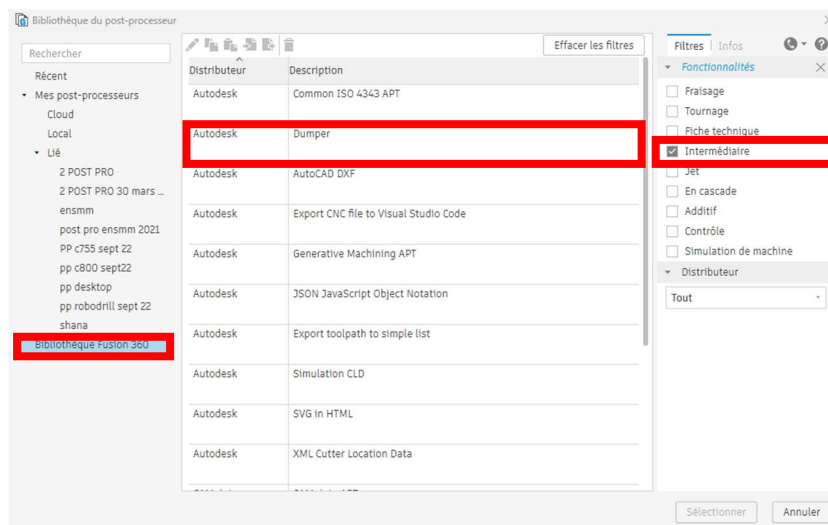
4. Le post pro

4.1. Le fichier intermédiaire

Liens entre les fonctions du post pro et le fichier intermédiaire de la FAO

Utilisez ce post pro pour comprendre quelles informations sont disponibles lors du développement d'un nouveau post pro. Le fichier généré affichera les informations principales pour chaque fonction d'entrée appelée.

Exemple de fichier intermédiaire :



Les fonctions d'entrée du post-processeur constituent l'interface entre le noyau et le post-processeur.

La fonction d'entrée sera appelée pour chaque enregistrement du fichier intermédiaire. La fonction d'entrée appelée est déterminée par le type d'enregistrement du fichier intermédiaire. Toutes les fonctions d'entrée ont le préfixe « on », il est donc recommandé de ne pas utiliser ce préfixe avec les fonctions que vous ajoutez.

```
347: onSection()
currentSection.unit=1
currentSection.workOrigin=(0, 0, 0)
currentSection.workPlane=[[1, 0, 0], [0, 1, 0], [0, 0, 1]]
currentSection.wcsOrigin=(0, 0, 0)
currentSection.wcsPlane=[[1, 0, 0], [0, 1, 0], [0, 0, 1]]
currentSection.workOffset=54
currentSection.probeWorkOffset=54
currentSection.wcs=''
currentSection.wcsIndex=-1
currentSection.dynamicWCSOrigin=(0, 0, 0)
```

```
348: onRapid(20.200000762939453, -3.9000000953674316, 15)
349: onRapid(20.200000762939453, -3.9000000953674316, 5)
350: onMovement(MOVEMENT_PLUNGE /*plunge*/)
350: onLinear(20.200000762939453, -3.9000000953674316, -0.699999988079071, 200)
351: onMovement(MOVEMENT_LEAD_IN /*lead in*/)
351: onCircular(false, 19.399999618530273, -3.9000000953674316, -0.699999988079071, 19.3
direction: CCW
sweep: 90deg
normal: X=0 Y=1 Z=0 (ZX)
radius: 0.8
352: onLinear(15, -3.9000000953674316, -1.5, 2534.54248046875)
```

4.2. Fonctions courantes du post pro

Voici une liste des fonctions d'entrée courantes prises en charge et quand elles sont appelées. Vous référer au document de référence d'Autodesk si vous souhaitez plus de détail... [Post Processor Training Guide \(autodesk.com\)](https://autodesk.com/post-processor-training-guide)

onOpen : écrit le début du programme avec des éléments comme le nom du programme etc...

onSection : cette fonction est traitée pour chaque opération de la configuration (outil, conditions de coupe, mouvement d'approche, etc... donc ci contre appelée 4 fois...

onRapid : la fonction pour les mouvements en rapide exemple G0

onLinear : la fonction pour les mouvements à vitesse programmée exemple G1

onCircular : la fonction pour les mouvements en interpolation circulaire exemple G2/G3

onDwell : la fonction qui va générer une temporisation dans le programme

onSpindleSpeed : la fonction qui va générer le code pour la fréquence de rotation de l'outil

onCycle : la fonction pour générer le code des cycles fixes de perçage, alésage, taraudage...

onCyclePoint : la fonction qui va générer les points suivants des cycles fixes

onCycleEnd : la fonction qui finit les cycles fixes et les annule

onSectionEnd : la fonction qui clos la section (opération d'usinage) et génère par exemple le retour en position de sécurité, etc...

onClose : la fonction pour générer la fin du programme : arrêt de la broche, du lubrifiant, retour aux origines etc...

4.3. Autres éléments du Post pro

Les variables utilisées par le noyau

Certaines des variables définies dans la section globale sont en fait définies et utilisées par le moteur de post process.

Ces variables se trouvent généralement tout en haut du fichier et sont faciles à discerner, car elles ne sont pas précédées du « var ».

Exemple des paramètres du noyau que vous trouverez dans la plupart des post-processeurs.

```
description = "ROBODRILL 2022-2";
vendor = "IUT GMP 25";
vendorUrl = "http://www.fanuc.com";
legal = "Copyright (C) 2012-2022 by Autodesk, Inc.";
certificationLevel = 2;
minimumRevision = 45821;

longDescription = "Generic post for Fanuc Robodrill like the Alpha

extension = "nc";
programNameIsInteger = true;
setCodePage("ascii");

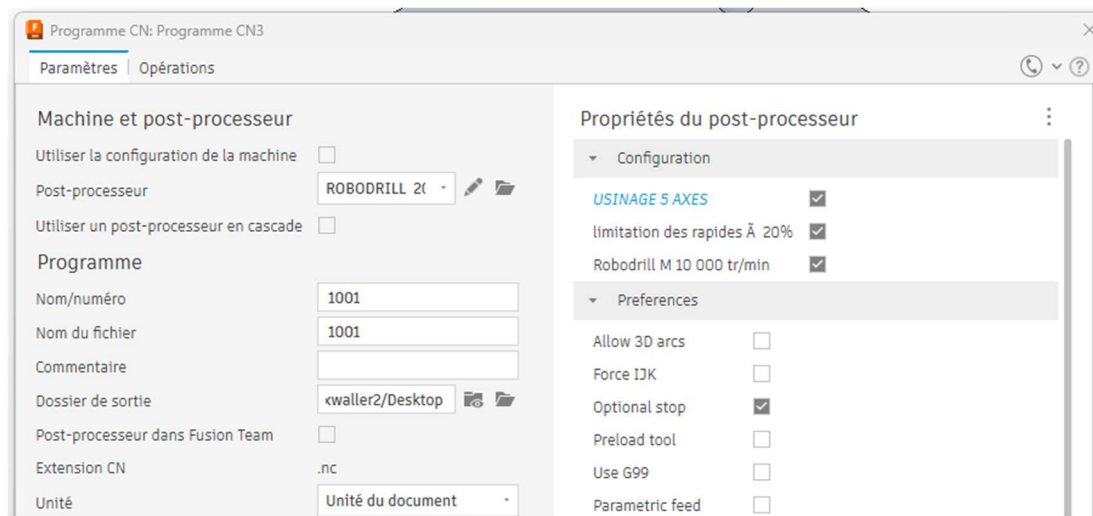
capabilities = CAPABILITY_MILLING | CAPABILITY_MACHINE_SIMULATION;
tolerance = spatial(0.002, MM);

minimumChordLength = spatial(0.01, MM);
minimumCircularRadius = spatial(0.01, MM);
maximumCircularRadius = spatial(1000, MM);
minimumCircularSweep = toRad(0.01);
maximumCircularSweep = toRad(180);
allowHelicalMoves = true;
allowedCircularPlanes = undefined; // allow any circular motion
highFeedrate = (unit == IN) ? 500 : 5000;
```

La table de propriétés

La table des propriétés contient des paramètres qui peuvent être modifiés au moment du post traitement afin que le post pro de la bibliothèque puisse rester de nature générique, tout en restant facilement personnalisable par les utilisateurs. Les paramètres de la table des propriétés seront généralement utilisés pour contrôler de petites variations dans la sortie créée par le post-pro.

Lorsque vous utilisez la boîte de dialogue Post-traitement pour exécuter le post-pro, la table des propriétés sera affichée dans la boîte de dialogue vous permettant de remplacer les paramètres du post-pro à chaque exécution.



```
robom: {
  title : "Robodrill M 10 000 tr/min",
  description: "Enable to output table position",
  group : "configuration",
  type : "boolean",
  value : true,
  scope : "post"
},
fifthAxis: {
  title : "USINAGE 5 AXES",
  description: "Enable to output table position",
  group : "configuration",
  type : "boolean",
  value : false,
  scope : "post"
},
limiterapide: {
  title : "limitation des rapides à 20%",
  description: "limitation des rapides à 20%",
  group : "configuration",
  type : "boolean",
  value : true,
  scope : "post"
},
}
```

4.4. Structure du Post pro

Les variables utilisées par le moteur du post pro

- Les tolérances
- Les capacités machine
- Les extensions du fichier

Page 11 les variables du noyau

Les tables de propriétés

- Le paramétrage laissé à l'utilisateur du post pro
- Les options de la machine

Ci-dessus

Les variables globales et les formats

- Le formatage du code CN
- Les paramètres (« variables ») pour l'ensemble du post pro

Page 6 les variables globales et locales

Les fonctions

- Les fonctions d'ouverture et de fermeture du post traitement
- Les fonctions de début et de fin d'usage
- Les fonctions qui « traduisent » les opérations d'usinage

Page 10 les fonctions courantes

4.5. Fonctionnement du post pro

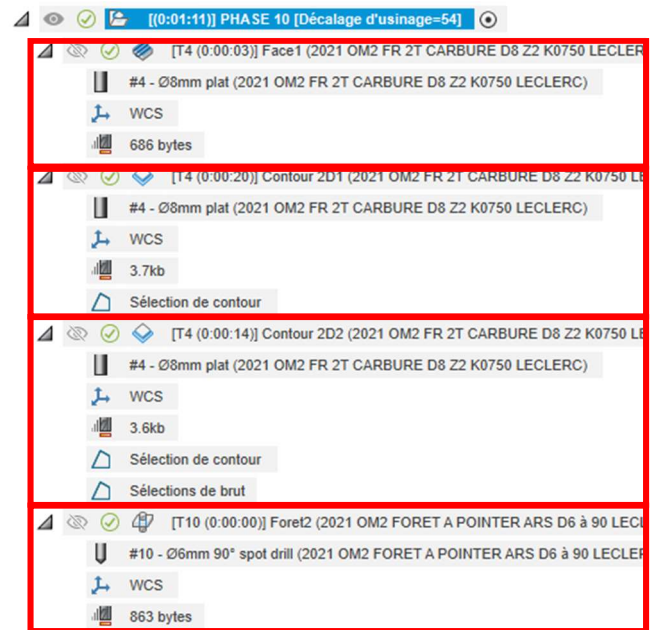
Un post pro a une extension .cps lequel est basé sur Javascript.

Le cœur du post pro est composé de fonctions qui sont interprétées dans une séquence linéaire depuis le début du fichier intermédiaire jusqu'à sa fin. Une fois le fichier intermédiaire terminé, il est supprimé.

Ces fonctions (les principales) sont répertoriées page 10:

onOpen : écrit le début du programme comme le programme avec des éléments comme le nom du programme etc...

onSection : cette fonction est traitée pour chaque opération de la configuration (outil, conditions de coupe, mouvement d'approche, etc... donc ci-contre appelée 4 fois...



4.6. Astuces pour debugger un Post pro

Insérer ces fonctions en début de post pro pour l'analyse et le debug... enlever les « // » pour les activer

```
//setWriteStack(true);
```

```
//setWriteInvocations(true);
```

La fonction setWriteStack vous permet de mettre en lien la ligne de code utilisée dans le post pro avec le code CN généré

La fonction setWriteInvocations vous inclura en plus les éléments du fichier intermédiaires exploitables

```
N45 G28 G91 Z0.
!DEBUG: 1 ROBODRILL IUT 2022-2 .cps:632
!DEBUG: 2 ROBODRILL IUT 2022-2 .cps:998
!DEBUG: 3 ROBODRILL IUT 2022-2 .cps:1715
N50 G49
!DEBUG: 1 ROBODRILL IUT 2022-2 .cps:632
!DEBUG: 2 ROBODRILL IUT 2022-2 .cps:1716
N55 G0 G90 G53 Y0
!DEBUG: 1 ROBODRILL IUT 2022-2 .cps:632
!DEBUG: 2 ROBODRILL IUT 2022-2 .cps:1717
N60 T4 M06
!DEBUG: 1 ROBODRILL IUT 2022-2 .cps:632
!DEBUG: 2 ROBODRILL IUT 2022-2 .cps:1719
N65 D04
!DEBUG: 1 ROBODRILL IUT 2022-2 .cps:632
!DEBUG: 2 ROBODRILL IUT 2022-2 .cps:1720
N70 M47 C20
!DEBUG: 1 ROBODRILL IUT 2022-2 .cps:632
!DEBUG: 2 ROBODRILL IUT 2022-2 .cps:1776
N75 S24000 M03
```

```
621 function writeBlock() {
622   var text = formatWords(arguments);
623   if (!text) {
624     return;
625   }
626   if (getProperty("showSequenceNumbers") == "true") {
627     if (optionalSection) {
628       if (text) {
629         writeWords("/", "N" + sequenceNumber, text);
630       }
631     } else {
632       writeWords2("N" + sequenceNumber, arguments);
633     }
634     sequenceNumber += getProperty("sequenceNumberIncrement");
635   } else {
636     if (optionalSection) {
637       writeWords2("/", arguments);
638     } else {
639       writeWords(arguments);
640     }
641   }
642 }
```

```
1714 writeBlock(gFormat.format(28), gAbsIncModal.format(91), "Z" + xyzFormat.format(0));
1715 disableLengthCompensation(true);
1716 writeBlock("G0 G90 G53 Y0");
1717 writeBlock("T" + toolFormat.format(tool.number), mFormat.format(6));
1718 dOutput.reset();
1719 writeBlock(dOutput.format(d));
1720 if (getProperty("limiterapide") ) {writeBlock("M47 C20");}
```

13

```
1775 writeBlock(
1776   sOutput.format(spindleSpeed), mFormat.format(tool.clockwise ? 3 : 4)
1777 );
```

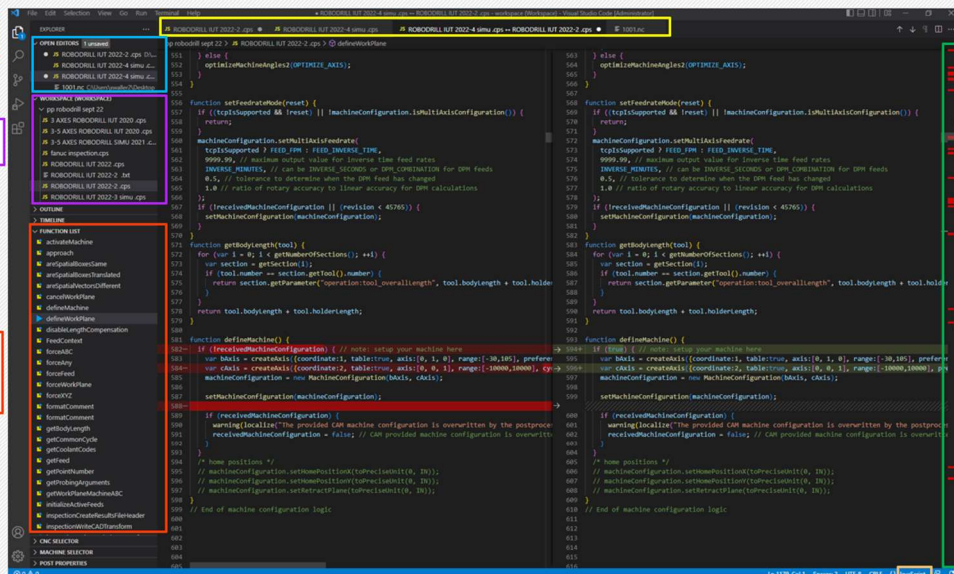
Encore quelques astuces: Si le nom devient rouge, c'est qu'il y a une erreur dans le code Javascript....

Les fichiers ouverts

Le dossier ouvert

Accès direct aux fonctions

Mise en évidence des différences entre deux fichiers comparés
Et accès direct à un endroit dans le post pro



Langage de programmation pour la mise en forme

5. Exercice de compréhension

Exercice de compréhension du post pro : on écrit un post pro à partir de rien ! (Cas extrêmement rare, en général on repart d'un post pro de base ou d'un existant)

5.1. onOpen

1. Tout d'abord créer, à l'aide de VisualCodeStudio, un fichier avec l'extension .CPS appelé myfirstpost.cps dans un dossier sur votre disque dur.
2. Puis dans ce fichier inclure la fonction onOpen qui génère le code du début de programme CN.

```
function onOpen() {
  writeln("%");
  writeln((programName ? (programName) : ""));
}
```

3. Charger et Ouvrir l'exemple de pièce « myfirstpost.f3d » et sélectionnez la première opération puis générer le programme avec votre post pro !
4. La fonction onOpen a donc envoyé un % à la fonction writeln. Ensuite, sur une nouvelle ligne, elle a envoyé le numéro de programme de la boîte de dialogue du post traitement à la fonction writeln récupéré dans le fichier intermédiaire.
5. Vous venez de créer votre premier morceau de code CN ! D'accord, c'est juste le signe de pourcentage et le numéro de programme mais c'est un début.

5.2. writeBlock

La plupart des armoires de commande ont une limite quant à la longueur du nom du fichier CN. Plus tard, nous contrôlerons cela. Maintenant il faut générer les appels d'outils...

Il faut tout d'abord créer la fonction writeBlock pour nous permettre de numéroter les lignes du programme.

```
var blockNumber = 5;
//writes the specified block of G code.
function writeBlock(block) {
  writeln( "N" + blockNumber + SP + block);
  ++blockNumber;
}
```

Définir la variable blockNumber et lui attribuer la valeur 5

Définir la fonction writeBlock avec comme argument block

Écrire à la ligne en concaténant le caractère N avec la valeur de blockNumber, un espace SP, et le contenu de l'argument block

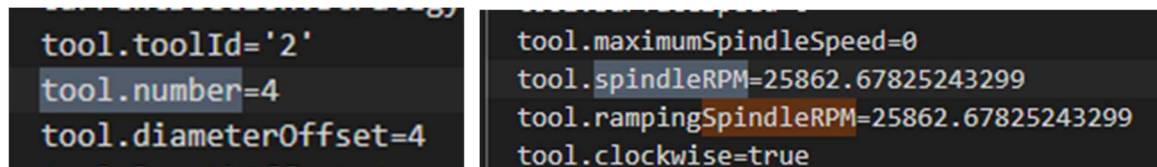
Incrémenter de 1 la variable blockNumber (pour le bloc suivant qui sera généré...)

5.3. onSection

Ensuite on ajoutera la fonction onSection

```
function onSection() {
  writeBlock("T" + tool.number + SP + " S" + tool.spindleRPM);
}
```

Les données du fichier intermédiaire sont exploitées pour les mettre sous la forme d'un programme CN



```
tool.toolId='2'
tool.number=4
tool.diameterOffset=4

tool.maximumSpindleSpeed=0
tool.spindleRPM=25862.67825243299
tool.rampingSpindleRPM=25862.67825243299
tool.clockwise=true
```

La fonction writeBlock est utilisée avec l'argument concaténé de T avec le numéro d'outil, un espace SP, puis S avec la fréquence de rotation. Donc on génère des blocs machine de type : N... T... S...

Remarque : il manque encore l'appel de l'outil (« M6 ») puis la mise en rotation...

Tester en post traitant le code CN

Nous allons intégrer maintenant les mouvements de l'outil, donc modifier la fonction onSection en conséquence

```
function onSection() {
  var initialPosition = getFramePosition(currentSection.getInitialPosition());
  writeBlock("T" + tool.number + SP + " S" + tool.spindleRPM);
  writeBlock("G0 " + initialPosition.x + initialPosition.y + initialPosition.z);
}
```

Définir la variable locale initialPosition

L'utiliser dans le bloc machine qui suit
« l'appel » de l'outil

Tester en post traitant le code CN

Le résultat aux valeurs près ressemble à ça :

%1001

N5 T1 S25862.67825243299

```
tool.spindleRPM=25862.67825243299
```

N6 G0 20.200001-3.915

```
STATE position=[20.200001, -3.9, 15]
```

Nous aurions préféré ceci :

%1001

N5 T1 S25862

N6 G0 X20.2 Y-3.9 Z15

Il faut donc formater la mise forme des données

5.4. Variables globales et formats

Nous allons définir les formats et les variables associées :

Ces formats vont être utiles à travers tout le post pro, ce sont donc des variables globales, si vous avez suivi, vous saurez où les écrire dans le post pro...

```
var xyzFormat = createFormat({decimals:(unit == MM ? 3 : 4), forceSign:true});
var feedFormat = createFormat({decimals:(unit == MM ? 0 : 2)});
```

//linear Output

```
var xOutput = createVariable({prefix:"X"}, xyzFormat);
var yOutput = createVariable({prefix:"Y"}, xyzFormat);
var zOutput = createVariable({prefix:"Z"}, xyzFormat);
var feedOutput = createVariable({prefix:"F"}, feedFormat);
```

Tester en post traitant le code CN

decimals:(unit == MM ? 3:4) signifie que le nombre de décimales dépend de l'unité mm ou pouce du document...

3 chiffres après la virgule en mm Revoir l'opérateur conditionnel

Aucun changement... Maintenant modifier la fonction onSection pour utiliser les formats définis !

```
function onSection() {
  var initialPosition = getFramePosition(currentSection.getInitialPosition());
  writeBlock("T" + tool.number + SP + "S" + tool.spindleRPM);
  writeBlock("G0 " + xOutput.format(initialPosition.x) + yOutput.format(initialPosition.y) +
  zOutput.format(initialPosition.z));
}
```

Tester en post traitant le code CN

5.5. onRapid

Il faut ensuite définir la fonction qui va générer les déplacements en rapide

```
function onRapid(x, y, z) {
  writeBlock("G0", xOutput.format(x), yOutput.format(y), zOutput.format(z));
  feedOutput.reset();
}
```

Les arguments x,y,z sont récupérés dans le fichier intermédiaire

```
348: onRapid(20.200000762939453, -3.9000000953674316, 15)
349: onRapid(20.200000762939453, -3.9000000953674316, 5)
350: onMovement(MOVEMENT_PLUNGE /*plunge*/)
350: onLinear(20.200000762939453, -3.9000000953674316, -0.699999988079071, 200)
351: onMovement(MOVEMENT_LEAD_IN /*lead in*/)
351: onCircular(false, 19.399999618530273, -3.9000000953674316, -0.699999988079071, 19.3
  direction: CCW
  sweep: 90deg
  normal: X=0 Y=1 Z=0 (ZX)
  radius: 0.8
352: onLinear(15, -3.9000000953674316, -1.5, 2534.54248046875)
```

feedOutput.reset() sert à forcer l'affichage de la vitesse d'avance lorsqu'un déplacement à vitesse d'avance programmé a lieu suite à un déplacement en rapide

Tester en post traitant le code CN

Si la mise en forme ne vous convient pas, modifier la fonction writeBlock comme ceci

```
var blockNumber = 5;

// This function Writes an NC block of G code.
function writeBlock() {
  writeWords2("N" + blockNumber, arguments);
  ++blockNumber;
}
```

Tester en post traitant le code CN

5.6. onLinear

Nous allons rajouter les déplacements en interpolation linéaire

```
function onLinear(x, y, z, feed) {  
  writeBlock("G1", xOutput.format(x), yOutput.format(y), zOutput.format(z),  
  feedOutput.format(feed));  
}
```

Les arguments x ,y ,z et feed sont récupérés dans le fichier intermédiaire

Le format de feedOutput a été défini précédemment

```
348: onRapid(20.200000762939453, -3.9000000953674316, 15)  
349: onRapid(20.200000762939453, -3.9000000953674316, 5)  
350: onMovement(MOVEMENT_PLUNGE /*plunge*/)  
350: onLinear(20.200000762939453, -3.9000000953674316, -0.699999988079071, 200)  
351: onMovement(MOVEMENT_LEAD_IN /*lead in*/)  
351: onCircular(false, 19.399999618530273, -3.9000000953674316, -0.699999988079071, 19.3  
  direction: CCW  
  sweep: 90deg  
  normal: X=0 Y=1 Z=0 (ZX)  
  radius: 0.8  
352: onLinear(15, -3.9000000953674316, -1.5, 2534.54248046875)
```

Tester en post traitant le code CN

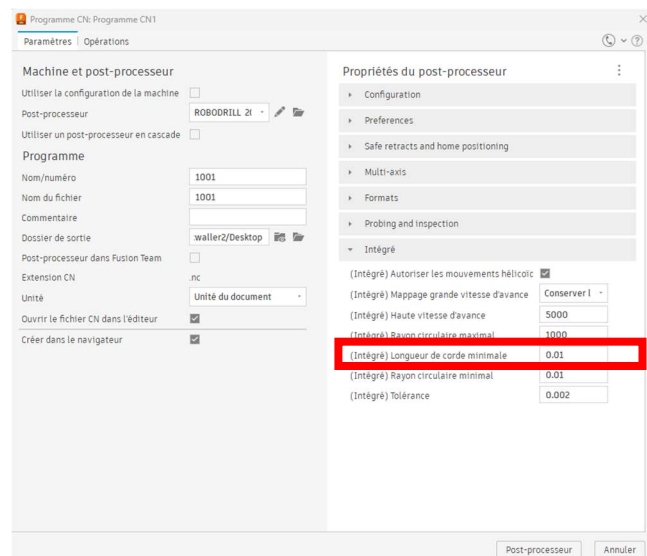
5.7. Paramètres intégrés (noyau)

Normalement le programme généré doit vous sembler beaucoup plus long que nécessaire en considération de la simplicité des usinages du fichier FAO donné.

En effet, le programme de post-process détectera les interpolations circulaires et s'il n'y a aucune fonction pour les gérer, il décomposera simplement les arcs en un grand nombre de petits mouvements linéaires.

Vous pouvez modifier la longueur de la corde utilisée en modifiant la propriété de tolérance (intégrée).

**Changer la valeur de longueur
de corde minimale et
Tester/comparer en post
traitant le code CN**



5.8. onCircular

Donc nous allons intégrer la fonction pour générer les interpolations circulaires

Tout d'abord il faut rajouter ces variables globales (vous savez à quel endroit du post pro... normalement...)

```

minimumChordLength = spatial(0.01, MM);
minimumCircularRadius = spatial(0.01, MM);
maximumCircularRadius = spatial(1000, MM);
minimumCircularSweep = toRad(0.01);
maximumCircularSweep = toRad(180);
allowHelicalMoves = true;
allowedCircularPlanes = undefined; // allow any circular motion

```

Ensuite définir la fonction onCircular et les variables globales de format manquantes, vous devriez y arriver seul...

Vous commencez à avoir quelques fonctions dans votre post pro, assurez-vous qu'il est organisé correctement !

```

function onCircular(clockwise, cx, cy, cz, x, y, z, feed) {
  var start = getCurrentPosition();
  var f = feedOutput.format(feed);
  writeBlock("G" + (clockwise ? 2 : 3), xOutput.format(x), yOutput.format(y), zOutput.format(z),
  iOutput.format(cx - start.x, 0), jOutput.format(cy - start.y, 0), f); }

```

```

351: onCircular(false, 19.399999618530273, -3.9000000953674316, -0.699999988079071, 19.399999618530273, -3.9000000953674316, -1.5, 2534.54248046875)
direction: CCW
sweep: 90deg
normal: X=0 Y=1 Z=0 (ZX)
radius: 0.8

```

Tester en post traitant le code CN

5.9. Correction de rayon G40/G41/G42

Nous allons intégrer les corrections de rayon G41/G42 dans les interpolations linéaires, pour cela modifier la fonction onLinear comme ci-dessous :

```

function onLinear(x, y, z, feed) {
  var xyz = xOutput.format(x) + " " + yOutput.format(y) + " " + zOutput.format(z);
  var f = feedOutput.format(feed);
  if (xyz) {
    writeBlock("G1_" + xyz + radiusCompensationTable.lookup(radiusCompensation) + " " + f);
  }
}

```

La fonction va donc chercher dans un tableau de données s'il le parcours est en G40, G41 ou G42

Il faut définir la variable globale correspondante

Tester en post traitant le code CN

```

var radiusCompensationTable = new Table(
  [" G40", " G41", " G42"],
  {initial:RADIUS_COMPENSATION_OFF,
  "Invalid radius compensation"
});

```

5.10. onSectionEnd

Nous allons intégrer la fonction pour générer la fin de section

```
function onSectionEnd(){
    writeBlock("G0 G91 G28 Z0"); // retract
    writeBlock("G90");
}
```

Il faut remarquer ici que l'on écrit du code machine en « dur ».

C'est-à-dire que ce qui va être généré par le post pro ne dépend pas du fichier intermédiaire.

Tester en post traitant le code CN

Est-ce à cet endroit qu'il fallait écrire ceci en dur ?

5.11. onClose

Puis intégrer la fonction pour générer la fin de programme

```
function onClose() {
    writeBlock("M30"); // stop program, spindle stop, coolant off
    writeln("%");
}
```

5.12. onCyclePoint

Dans le fichier FAO, il y a également des opérations de perçage...

Générer le programme de ces opérations et analyser le code CN.

Pour améliorer la lisibilité et diminuer la taille du programme, nous allons insérer les cycles fixes.

```
function onCyclePoint(x, y, z) {

    if (isFirstCyclePoint()) {
        repositionToCycleClearance(cycle, x, y, z); // return to clearance plane and set absolute mode
        var F = cycle.feedrate;
        if (cycleType == "drilling") {
            writeBlock(
                "G98 G90 G81" + xOutput.format(x)+ yOutput.format(y)+ zOutput.format(z) +
                feedOutput.format(F));
        }

        } else {
            writeBlock(xOutput.format(x) + yOutput.format(y));
        }
    }
}
```



```

420: onCyclePoint(42, 18.649185180664062, -60)
420: EXPANDED onRapid(42, 18.649185180664062, 15)
420: EXPANDED onRapid(42, 18.649185180664062, 5)
420: EXPANDED onLinear(42, 18.649185180664062, -60, 1000)
420: EXPANDED onRapid(42, 18.649185180664062, 5)
  STATE position=[42, 18.649185, 5]
  STATE spindleSpeed=5000
  STATE radiusCompensation=RADIUS_COMPENSATION_OFF // off
421: onCyclePoint(42, -22, -60)
421: EXPANDED onRapid(42, -22, 5)
421: EXPANDED onRapid(42, -22, 5)
421: EXPANDED onLinear(42, -22, -60, 1000)
421: EXPANDED onRapid(42, -22, 5)
  STATE position=[42, -22, 5]
  STATE spindleSpeed=5000
  STATE radiusCompensation=RADIUS_COMPENSATION_OFF // off
422: onCyclePoint(-42.59062194824219, -22, -60)
422: EXPANDED onRapid(-42.59062194824219, -22, 5)
422: EXPANDED onRapid(-42.59062194824219, -22, 5)
422: EXPANDED onLinear(-42.59062194824219, -22, -60, 1000)
422: EXPANDED onRapid(-42.59062194824219, -22, 5)
  STATE position=[-42.590622, -22, 5]
  STATE spindleSpeed=5000
  STATE radiusCompensation=RADIUS_COMPENSATION_OFF // off
423: onCyclePoint(-42.59062194824219, 18.649185180664062, -60)
423: EXPANDED onRapid(-42.59062194824219, 18.649185180664062, 5)
423: EXPANDED onRapid(-42.59062194824219, 18.649185180664062, 5)
423: EXPANDED onLinear(-42.59062194824219, 18.649185180664062, -60, 1000)
423: EXPANDED onRapid(-42.59062194824219, 18.649185180664062, 5)
  STATE position=[-42.590622, 18.649185, 5]
  STATE spindleSpeed=5000
  STATE radiusCompensation=RADIUS_COMPENSATION_OFF // off
424: onCycleEnd()

```

```

420: onCycle()
  cycleType='drilling'
  cycle.clearance=5
  cycle.retract=5
  cycle.stock=0
  cycle.depth=60
  cycle.feedrate=1000
  cycle.retractFeedrate=1000
  cycle.plungeFeedrate=1000
  cycle.bottom=-60
  cycle.dwell=0
  cycle.incrementalDepth=60
  cycle.incrementalDepthReduction=0
  cycle.minimumIncrementalDepth=0
  cycle.plungesPerRetract=1
  cycle.shift=0
  cycle.shiftOrientation=0
  cycle.compensatedShiftOrientation=0
  cycle.shiftDirection=3.141592653589793
  cycle.backBoreDistance=0
  cycle.stopSpindle=0

```

Analyser et décoder le fonctionnement de la fonction

Appeler le prof pour valider !

Tester en post traitant le code CN

5.13. onCycleEnd

Il faut également insérer la fonction qui annule les cycles fixes

```

function onCycleEnd() {
  if (!cycleExpanded) {
    writeBlock("G80");
    zOutput.reset();
  }
}

```

Tester en post traitant le code CN

5.14. Ajout d'un 4^{ème} axe

Il faut définir l'axe supplémentaire de la machine.

Comme la machine ne change pas durant tout l'usinage, il faut modifier la fonction onOpen

```
function onOpen() {
  var aAxis = createAxis({coordinate:0, table:true, axis:[1, 0, 0], range:[-360,360], cyclic:true,
  preference:1});
  machineConfiguration = new MachineConfiguration(aAxis);
  setMachineConfiguration(machineConfiguration);
  optimizeMachineAngles2(1);

  writeln("%");
}
```

Analyser et décoder le fonctionnement de la fonction

Appeler le prof pour valider !

Ensuite modifier la fonction onSection pour générer le code utilisant le 4^{ème} axe en positionné

```
function onSection() {
  var abc = machineConfiguration.getABC(currentSection.workPlane);
  setRotation(machineConfiguration.getRemainingOrientation(abc, currentSection.workPlane));
  writeBlock("G0", "A" + abcFormat.format(abc.x));

  writeBlock("T" + tool.number, "S" + rpmFormat.format(tool.spindleRPM));

  xOutput.reset();
  yOutput.reset();
  zOutput.reset();
  var initialPosition = getFramePosition(currentSection.getInitialPosition());
  writeBlock("G0", xOutput.format(initialPosition.x), yOutput.format(initialPosition.y));
  writeBlock("G0", zOutput.format(initialPosition.z));
}
```

Tester en post traitant le code CN du programme complet

Le code ne veut pas se générer ☺

Analysez le fichier log contenant l'erreur et corrigez là !

Changez dans la fonction onOpen le 1 en -1 post traitez et constatez la différence

```
var aAxis = createAxis({coordinate:0, table:true, axis:[-1, 0, 0], range:[-360,360], cyclic:true,
```

5.15. Sécurisation de post pro

Revenons au début des exercices, nous avons récupéré le nom du programme.

Or dans certaines machines CN, le nom de programme ne doit comporter que des chiffres et être un entier.

Modifier la partie de la fonction onOpen qui s'occupe du nom de programme

```
function onOpen() {  
    writeln("%");  
    if (programName) {  
        var programId;  
        try {  
            programId = getAsInt(programName);  
        } catch(e) {  
            error(localize("Program name must be a number."));  
            return;  
        }  
        writeln((programName ? (programName) : ""));  
    }  
}
```

Error(localize(« »)) arrête la génération du programme et affiche dans le fichier log le message à destination de l'utilisateur.

On ne générera donc pas un programme avec un nom que la machine ne peut pas accepter !

Tester en post traitant le code CN

6. Exercices supplémentaires

1. La prise en compte du correcteur de longueur de l'outil n'est pas effectuée.
2. Le code d'origine n'apparaît pas...G54, G55 etc...
3. L'appel de l'outil « M6 » n'est pas fait...
4. La mise en rotation de l'outil n'est pas faite... (M3 ou M4 ?)
5. Il n'y a pas de commentaires avec la désignation de l'outil...
6. Il n'y a pas de commentaire concernant l'opération générée...
7. Il n'y a pas de commentaire en début de programme...
8. La lubrification n'est pas mise en route M8...
9. La lubrification ne s'arrête jamais M9...
10. Il n'y a pas de blocs de sécurité en début de programme...

11. Les plans d'interpolation circulaire G17, G18, G19 ne sont pas définis...
12. Il n'y a pas de sécurité si le programmeur n'utilise pas une origine entre 54 et 59
13. Il n'y a pas de sécurité sur la fréquence de rotation maxi de l'outil (10 000 tr/min)
14. Définir une propriété sélectionnable par l'utilisateur pour fixer la fréquence de rotation maxi à 10 000 ou 24 000
15. Définir une propriété sélectionnable par l'utilisateur pour choisir d'utiliser R plutôt que I, J, K en G2/G3
16. Il n'y a pas de sécurité sur la longueur totale de l'outil (200mm par exemple outil + porte outil)
17. Vous êtes trop fort ! Personnalisez un post pro de génération de document pour éditer vos documents techniques de manière automatique avec les éléments dont vous avez besoin !

7. Pour aller plus loin...

[Autodesk Fusion 360 Post Processor Utility](#)

[Autodesk CAM Post Processor Documentation](#)

[Fusion 360 Manufacture Forum - Autodesk Community](#)

Table des matières

1. Qu'est-ce qu'un post pro ?	1
2. Extrait d'un post pro personnalisé pour ajouter de la sécurité par rapport à la machine et explications	3
3. Javascript.....	4
3.1. La syntaxe	4
3.2. Ecriture et édition.....	4
3.3. Les variables.....	5
3.4. Calculs et tests sur les expressions.....	6
3.5. Les tests conditionnels	7
3.6. Les fonctions	8
4. Le post pro	10
4.1. Le fichier intermédiaire	10
4.2. Fonctions courantes du post pro.....	10
4.3. Autres éléments du Post pro	11

4.4.	Structure du Post pro	12
4.5.	Fonctionnement du post pro.....	13
4.6.	Astuces pour debugger un Post pro	13
5.	Exercice de comprehension	14
5.1.	onOpen	14
5.2.	writeBlock.....	15
5.3.	onSection	15
5.4.	Variables globales et formats	16
5.5.	onRapid.....	17
5.6.	onLinear	18
5.7.	Paramètres intégrés (noyau)	18
5.8.	onCircular.....	19
5.9.	Correction de rayon G40/G41/G42	19
5.10.	onSectionEnd	20
5.11.	onClose.....	20
5.12.	onCyclePoint	20
5.13.	onCycleEnd.....	21
5.14.	Ajout d'un 4 ^{ème} axe	22
5.15.	Sécurisation de post pro	23
6.	Exercices supplémentaires.....	23
7.	Pour aller plus loin.....	24