NC_Code_Checker.pl について

舞鶴工業高等専門学校
技術職員　石井貴弘

NC_Code_Checker.pl は、2019 年度科学研究費助成事業(科学研究費補助金)
奨励研究(課題番号:19H00247)の助成を受けて作成されたものです。

1. NC_Code_Checker.pl の概要

FANUC、MELDAS、MAPPS 系のマシニングセンタ用の NC プログラムの誤りを検出する
Perl スクリプトです。
主に手書きで作成した NC プログラムのチェックを想定しています。
誤りの内容を具体的に表示できるので、初学者への NC コード教育の場に強くお勧めします。
加工やプログラム作成に慣れている方のケアレスミスのチェックにもぜひご活用ください。

機能としては、マシニングセンタ用の NC プログラムが記述されたファイルを入力として
文法の誤り、構文の誤り、手順の誤りなどを検出し、
誤りを含むブロックの行番号と原文、誤りの内容を出力します。
(プログラム言語のコンパイラのような機能です。)
誤りが検出されなかった場合は「(誤りは見つかりませんでした。)」と出力されます。

サブプログラムにも対応しています。
サブプログラムはメインプログラムのファイル中、
メインプログラムの下に記述されていれば認識します。
または、サブプログラムを記述したファイルを
O<サブプログラム番号>で始まる名前(例えば O100.ncd)で作成して、
メインプログラムのファイルと同じフォルダ内に置くか、
オプションで設定可能なフォルダ(後述)内に置いてください。
カスタムマクロやマクロ呼び出し(G65・G66)にも一応対応していますが、
誤りがある場合はその内容のみで、変数の代入などの処理の過程は出力されないので、
マクロプログラムの場合は別途、transpose_macro(FANUC).pl で平文の NC コードに変換してから
NC_Code_Checker.pl を実行していただいた方が確実です。

また、チェック内容が出力されるファイル等とは別に、
マイドキュメントフォルダ(Windows の場合)にログファイルが生成されます。
どのような誤りが検出されたかの履歴が分かるようになっていますので、ご活用ください。

2. NC_Code_Checker.pl の使い方

2.1. 実行環境について

本スクリプトは Perl スクリプトですので、まずは Perl が実行できる環境を用意してください。
Windows 環境の場合、ActiveState 社より ActivePerl をダウンロードし、インストールしてください。

2.2. スクリプト内の設定項目について

様々な機種、様々な用途に対応するために、各種の設定項目が用意されています。
NC_Code_Checker.pl をテキストエディタで開いて、冒頭部分の設定項目を編集してください。
(Perl モード対応のテキストエディタをお勧めします。#で始まる行が色分けされるので分かりやすくなります。)
#で始まる行は説明文です。$で始まる行の = より右が設定値になります。
説明文に従って設定値を変更してください。
$ で始まる行は必ず ; で終わるので、; を消去しないようにしてください。

スクリプト冒頭の設定項目とその説明文を以下に示します。
マシニングセンタの機種や使用用途に合わて設定を作ったスクリプトファイルを
それぞれ別名で保存して管理することをお勧めします(NC_Code_Checker(NVX5060 用).pl など)。
とりあえず使ってみてから設定を見直したい方は、**2.3 実行方法について** を先にご覧ください。

```
### 設定項目 ###########################################################
# チェック後の結果ファイルとは別に、ログファイル(追記型)が
# ユーザードキュメントフォルダに、以下のファイル名で生成されます。
$log_file = "NC_Code_Checker_log.txt";

# O コード(プログラム番号)が指令されていない場合に
# 警告する場合は 0、しない場合は 1 を設定してください。
$O_warn_flag = 0;

# X,Y,Z,I,J,K,R,Q コードの数値に小数点が入っていない場合に
# 警告する場合は 0、しない場合は 1 を設定してください。
$DP_warn_flag = 0;

# S コードの数値に小数点がついているとエラーになる機種の場合は 0、
# 小数点がついていてもエラーにならない機種の場合は 1 を設定してください。
$S_DP_flag = 0;
```

```
# 固定サイクルの P コードの数値に小数点がついている場合に
# 警告する場合は 0、しない場合は 1 を設定してください。
$P_DP_flag = 0;

# 工具長補正番号 H が工具番号 T と違う場合に
# 警告する場合は 0、しない場合は 1 を設定してください。
$H_warn_flag = 0;

# 工具径補正番号 D が工具番号 T と違う場合に
# 警告する場合は 0、しない場合は 1 を設定してください。
$D_warn_flag = 0;

# 工具交換前に実行すべきコード(原点復帰等)を入力してください。
# 特に必要ない場合は "" としてください。(複数ブロックには未対応)
# このコードを実行後であっても工具交換する前に原点復帰以外の移動が
# 指令された場合、警告します。
$pre_TC_code = "G91G28Z0";

# T コードと M06(工具交換)を同じブロックに入れると
# M06 の動作が先で、その後に T 番号の工具が交換待機位置に呼び出される機種の場合は 0、
# T 番号の工具呼び出しが先で、その後 M06 が動作する機種の場合は 1 を設定してください。
$TC_flag = 0;

# T コードと M06 が同じブロックにないと工具交換されない機種の場合は 1、
# そうでない場合は 0 を設定してください。
$TC_flag2 = 0;

# 基本的には同じブロックにあると不具合の元となるので、
# T コードと M06 が同じブロックにある場合、
# 警告する場合は 1、警告しない場合は 0 を設定してください。
$TC_warn_flag = 1;

# 初期状態で主軸についている工具の工具番号を設定する場合、入力してください。
# 特に必要なければ 1 のままにしておいてください。
$present_T = 1;
```

```
# 工具交換が指令されていない状態で、上の $present_T の工具番号を
# T 番号と H 番号が違う場合等の判定に使用する場合は 1 を
# 使用しない場合は 0 を設定してください。
$present_T_flag = 0;

# 主軸回転数を指令せずに主軸正転(M03)すると警告しますが、
# (リジッド(同期式)タッピングサイクルのときは除く)
# 工具交換後に改めて S コードを指令し直して主軸正転しないと
# エラーになる機種の場合は 0、エラーにならない機種の場合は 1 を設定してください。
$S_flag = 0;

# 早送り以外で Z を下降させるとき(G01Z_や固定サイクル)に
# クーラント ON(M08)状態でない場合、
# 警告する場合は 1、警告しない場合は 0 を設定してください。
$M08_warn_flag = 1;

# コメント文中に全角文字(全角スペースを含む)が入っている場合、
# 警告する場合は 0 、しない場合は 1 を設定してください。
$ComZen_warn_flag = 0;

# 1 ブロック中で M コードを 1 つだけ指令できる機種の場合は 0、
# 1 ブロック中で M コードを複数指令できる機種の場合は 1 を設定してください。
$multi_M = 0;

# G74、G84 コードの前のブロックに M29S_がないとき警告する場合は 1、
# 必要ない、またはフロート(非同期式)サイクルを使用する場合は 0 を設定してください。
$G84_flag = 1;

# 上の $84_flag が 0 の場合で、G74、G84 コードの前のブロックが M29S_でなくても
# G98(G99)G84X_Y_Z_R_F_S_;
# の構文で同期式タッピングサイクルが実行される機種(舞鶴高専には存在する)の場合で
# F コード、S コードが G74、G84 のブロックにないときに警告する場合は 1、
# 警告しない、あるいはそういう仕様ではない場合は 0 を設定してください。
$G84_flag2 = 0;

# 工具径補正で先読みできるブロック数を設定してください。
# 工具径補正モード中、XY の移動がないブロックがそれを超えた場合、警告します。
$foresee_block = 2;
```

```
# 固定サイクルのモーダル中に他の移動命令(G00 等)があれば
# 一般的には固定サイクルはキャンセルされますが、
# 固定サイクルキャンセル命令(G80)でキャンセルしていない場合に
# 警告する場合は 0、しない場合は 1 を設定してください。
$G80_flag = 0;

# S コードで指令する主軸回転数の上限値と下限値を設定してください。
# 範囲外の値の場合、警告します。
$S_max = 10000;
$S_min = 200;

# F コードで指令する送り速度の上限値と下限値を設定してください。
# 範囲外の値の場合、警告します。
# (F でタップのピッチを設定する機種もあるので
#    G74,G84 と同じブロックのとき、下限値は無視します。)
$F_max = 600;
$F_min = 30;

# G73,G81,G82,G83 で下穴をあけるサイクルがある場合で
# 同一のプログラム内で
# そのサイクルで指令した XY 座標にあけた穴に対して
# G74,G84,G85,G86,G88,G89 で追加加工するサイクルがある場合、
# すでに開けられた穴の Z 座標に対して
# この値以上高い位置(単位:ミリ)で止まっていない場合、警告します。
# 警告が必要ない場合は 0 を設定してください。
$Z_prepare_gap = 4;

# G74,G84,G85,G86,G88,G89 で加工するサイクルがある場合、
# そのサイクルで指令している XY 座標に対して
# 同一のプログラム内の前工程に
# G73,G81,G82,G83 で下穴をあけるサイクルがない場合に
# 警告する場合は 1、警告しない場合 0 を設定してください。
# (何ヵ所かタップ加工等する場合で、1 ヵ所だけ下穴と座標が違うミスなども検出できます。
# タッピングやボーリング加工のみのプログラムの場合は 0 を設定してください。)
$prepared_hole_flag = 0;
```

```
# 主軸をZ軸マイナス方向に早送りした場合で
# その高さでXY方向に直線補間、または円弧補間している場合に
# 警告する場合は 1、警告しない場合は 0 を設定してください。
$Z_G00_warn_flag = 1;

# 主軸をZ軸プラス方向に逃がす場合(Z軸単独の移動)に
# 直線補間で移動させている量が
# この値を超えている場合、警告します(単位:ミリ)。
# (早送りの方がふさわしい場合です)
# 必要ない場合は 0 を設定してください。
$Z_G01escape_max = 30;

# Z軸マイナス方向へ直線補間で移動する量(1ブロックの命令で)が
# この値を超えている場合、警告します(単位:ミリ)。
# 正の値で設定し、必要ない場合は 0 を設定してください。
$Z_G01_max = 50;

# サブプログラムはメインプログラムと同じファイル中に
# メインプログラムの下方に記述されていれば認識します。
# サブプログラムを別のファイルに記述する場合は、
# O[プログラム番号]で始まるファイル名のファイル(例: O100.ncd)を
# メインプログラムのファイルと同じフォルダに入れるか、
# 以下に設定するパスのフォルダにサブプログラムのファイルを入れてください。
$sub_folder= 'C:¥Program Files¥NCVC¥subpro';

# NCVC等のCAMソフトで出力されたプログラムに手動で工具交換命令を追加するとき、
# G90G54G00X0Y0も追加する場合がありますが(少なくとも舞鶴高専では)、
# 次の移動先が工具交換前のXY座標の片方もしくは両方と一致しているとき
# 出力されたプログラムには移動のない軸のコードは記述されていないことがありますが、
# その場合、次の移動先がX0かY0で書き換えられていることになります。
# その可能性がある時、警告する場合は 1、警告しない場合は 0 を設定してください。
# (1.工具交換、2.G00X0Y0、3.XまたはY軸のみの移動か、XY移動のない固定サイクル
#  の条件を満たした場合、警告します)
$after_TC_warn_flag = 0;

# オプショナルブロックスキップを有効(ON)にしたい場合は 1、
# 無効(OFF)にしたい場合は 0 を設定してください。
$OBS_switch = 0;
```

```
# M98 の繰り返し数の指定方法について、
# M98P○○○○L○○○○の L で指定する方式の場合は  0、
# M98P○○○○○○○○の前 4 桁で指定する方式の場合は  1  を設定してください。
$M98_houshiki = 0;

# WHILE や DO ループ内でのループ数の上限数を設定してください。
#  条件式などでループが終わらない場合に無限ループを防ぎます。
$loop_max = 300;

#  プログラム全体での GOTO 命令の上限数を設定してください。
# GOTO 処理での無限ループを防ぎます。
$GOTO_exe_max = 500;

#  システム変数を使用する場合で、初期値が必要なものは値を登録してください。
#  初期状態の G コードのモーダル情報が違う場合も同様(スクリプト中の  %initial_G  参照)。
#%system_value= (,)

#  Windows 環境でない場合で、実行エラーになる場合は
#  以下の行頭に  #  を付けてください。
use Win32::OLE;
########################################################################
```

2.3. 実行方法について

NCVC と連携させて結果をファイルに出力したい場合は **2.3.1**
コマンドプロンプトのみを使い、結果をファイルに出力したい場合は **2.3.2**
出力結果をコマンドプロンプト上で表示させたい場合は **2.3.3**
を参照してください。

2.3.1 NCVC と連携させて結果をファイルに出力する方法

まずは NCVC(※1)がインストールされていることが前提です。
NCVC は NC プログラムの動作や切削後の形状をシミュレートできますし、
さらに NC_Code_Checker.pl を実行することで
NC プログラムのチェックには非常に強力なツールになりえます。

(※1)NCVC（NC Viewer and Converter）は舞鶴工業高等専門学校 眞柄賢一氏の著作物です。

NCVC のメニューバーから[オプション]-[外部アプリケーションの設定]を開き、
詳細設定エリアの「参照」ボタンを押し、
NCVC に付属している Scriptorium(※2)(NCVC 本体と同じディレクトリにあります)を選択して、
「追加」ボタンを押し、外部アプリケーション一覧に追加します。

このとき、外部アプリケーション一覧の一番上はテキストエディタが望ましいので、
もし行番号が表示可能なテキストエディタをインストール済みなら、
それも追加して一番上に設定することをお勧めします。(参照を選択して「UP」ボタン)
なければ一番上は「メモ帳」アプリのままにしておいてください。
二番目以降に Scriptorium を置くようにしてください。

チェックしたい NC プログラムのファイルを NCVC で開きます。
NCVC ウインドウ上部の外部アプリケーションアイコン群から
Scriptorium(羽根の画像)アイコンをクリックすると、
(見当たらない場合は外部アプリケーションのアイコンのツールバーを少し移動させると出てきます)
Scriptorium のウインドウが立ち上がり、
[入力ファイル]には NCVC で開いている NC プログラムのファイルのパス、
[出力ファイル]には、入力ファイルのファイル名に「_cvted」(設定により変更可能)という文字列が
追加されたファイル名のパスが自動的に指定されています。
[スクリプトファイル]に NC_Code_Checker.pl への参照を設定し、実行ボタンを押すと、
[出力ファイル]のパスにチェック結果のファイルが出力されます。

このとき NCVC のウインドウでは「有効な NC データがありませんでした」という警告が出ますが、
[OK]ボタンで次に進んでください(NCVC Ver3.8.2 以降では警告が出ないようにしていただきました)。

誤りが検出された場合、
出力結果に表示される行番号は、入力ファイル内の(テキストファイルとしての)行番号になります。
プログラムの修正は、入力ファイルをテキストエディタで開いて行ってください。
NCVC の外部アプリに行番号が表示できるテキストエディタを設定しておけば、
NCVC 上のアイコンから入力ファイルをテキストエディタで開けるので、
行番号を確認しながら編集する作業が簡単に行えます。

なお、プログラムの編集については、入力ファイルと出力ファイルの NCVC のウインドウは閉じて、
入力ファイルと出力ファイルのテキストエディタのウインドウのみにしてから
行番号を見比べながらの作業がお勧めです。

(※2)Scriptorium は舞鶴工業高等専門学校 能勢嘉朗氏の著作物です。

2.3.2 コマンドプロンプトから実行する方法

コマンドプロンプトを起動すると、コマンドを実行するウインドウが開きます。
perl A B C
　A: NC_Code_Checker.pl のファイルのパス
　B: チェックしたい NC プログラムのファイルのパス
　C: チェック結果を出力したいファイルのパス
　(perl と A, B, C の間に半角スペースを入れる)
と入力(A, B はファイルのアイコンをマウスでコマンドプロンプトのウインドウに
ドラッグアンドドロップすれば簡単に入力できます。C は B をもう一度ドラッグアンド
ドロップして、ファイル名の一部を変更するのがお勧めです。)して Enter キーを押します。
C で指定したパスの場所に出力結果ファイルが作成されます。

2.3.3 出力結果をコマンドプロンプトに表示させる場合

コマンドプロンプトを起動すると、コマンドを実行するウインドウが開きます。
perl A B
　A: NC_Code_Checker.pl のファイルのパス
　B: チェックしたい NC プログラムのファイルのパス
　(perl と A, B の間に半角スペースを入れる)
と入力(A, B はファイルのアイコンをマウスでコマンドプロンプトのウインドウに
ドラッグアンドドロップすれば簡単に入力できます)してエンターキーを押します。
そのウインドウにチェック結果が表示されます。
プログラム言語のコンパイラのような使い方になります。

3. 出力結果について

3.1 スクリプト実行後の出力内容について

2.2 で示した、スクリプト冒頭にある設定内容に従って、入力ファイル内の NC プログラムから
文法の誤り、構文の誤り、手順の誤りなどを検出し、
誤りを含むブロックの行番号と原文、誤りの内容を出力します。

[<使用スクリプトのファイル名>実行結果]
?行目: ＮＣコードの原文１ ← 誤りの内容１
?行目: ＮＣコードの原文２ ← 誤りの内容２
…

上記のようなフォーマットで、検出した誤りを全て出力します。
誤りが検出されなかった場合は「(誤りは見つかりませんでした。)」と出力されます。
実行すべきコードを実行しないまま終了した、など特定の行が誤りとはいえない誤りについては、
このフォーマットではない場合もあります。

出力される行番号は、入力ファイル内の(テキストファイルとしての)行番号になります。
誤りが検出された場合は、入力ファイルを、行番号が表示できるテキストエディタで開き、
行番号を参照しながら修正してください。

サブプログラムやマクロプログラムを呼び出している場合で、その中で誤りが検出された場合には、
出力される行番号は呼び出し先の O 番号の行から数えた行数になり、その後ろに
(O[呼び出し先プログラム番号]中 / O[呼び出し元のプログラム番号] L[呼び出し元の行番号])
という文字列が付きます。呼び出し元のプログラムがメインプログラムの場合は、
O[ ]の代わりに MAIN という文字列になります。

多重呼び出しの場合は、/ 以下が増えていくことになります。
例えば、O1000 の(メイン)プログラムの 10 行目で、サブプログラム O2000 を呼び出し、
その 15 行目で O3000 のマクロを呼び出して O3000 の 20 行目で誤りを検出したときは、

20 行目(O3000 中/O2000 L15/MAIN L10): NC コードの原文 ← 誤りの内容

と出力されます。

3.2 ログファイルについて

誤り内容の詳細の出力とは別に、Windows の場合はマイドキュメントフォルダ、
他の OS の場合はホームディレクトリ直下にログファイルが生成されます。
ログファイルのファイル名は 2.2 の設定項目にて設定可能です。
同じ名前のファイルがすでにある場合は、ログファイル内の一番下に追記されていきます。
スクリプト 1 回の実行で出力されるログファイルの内容を以下に示します。

<日付・時刻>
[使用スクリプト: <使用したスクリプトのファイル名>]
[対象ファイル: <入力ファイルのファイルパス>]
　誤りの内容 1
　誤りの内容 2
　…

3.3 検出される誤りについて

以下に、検出される誤りの一覧を示します。
(オプション)がついている項目は、検出するかどうかを設定可能な内容です。
2.2.にあるように、検出するかどうかの設定・検出に使用する値や NC コードなどは、
スクリプトの冒頭で設定することができます。

・行末に;<EOB>がついている(テキストデータでは改行がその意味を持つので不要)
・%が単独で記述されていない
・O コードの O<オー>が 0<数字のゼロ>
・O コードが指令されていない(オプション)
・O コードがプログラムの先頭で指令されていない
・1 つのプログラム中で O コードが複数指令されている
・0<数字のゼロ>であるべき場所に O<アルファベットのオー>がある
・アドレスが小文字になっている
・アドレスとは考えられない位置に小文字のアルファベットがある
・1 つの数値に小数点が複数ある
・.<小数点>であるべき場所が ,<コンマ>になっている
・NC プログラム中に全角文字が入っている
・コメント文中に全角文字が入っている(オプション)
・アドレスの前にマイナス符号がついている
・アドレスの後に数値がない
・アドレスのない数値がある
・アドレス直後のマイナスと数字の間にスペースがある
・工具長補正 G43 実行時に H コードが呼び出されていない
・工具径補正 G41 または G42 実行時に D コードが呼び出されていない
・M98,G65,G66 のブロックに P コードがない
・M98,G65,G66 の P で指令している番号のプログラムが見つからない
・G00,G01 のブロックに R・I・J コードがある(実機ではエラーにならないが不必要)
・G02,G03 のブロックに R・I・J コードのいずれも指令されていない(G17 のとき)
・円弧補間の R コードの数値に 0 が指令されている(モーダルで G コード省略の場合も含む)
・円弧補間で I・J コードのいずれかと R コードの両方を指令している(G17 のとき)
・円弧補間で R コードを指令しているが、X・Y コードのいずれも指令されていない(G17 のとき)
・円弧補間で R 指令されている半径値では終点に到達できない
・円弧補間で I・J(・K)指令されている中心点では終点に到達できない
・１ブロックで複数の固定サイクルのコードを指令している(下と重複するが、あえて明示)
・１ブロック中で同一グループの G コードを複数指令している(実機ではおそらく一番後ろの G コードが有効になり、エラーにはならないが、その前の G コードは不必要)

・各固定サイクルで最低限必要なアドレスが指令されていない(G82 に Z や P、G83 に Z や Q など)
・各固定サイクルで不要なアドレスが指令されている(G81 に P や Q など)
・固定サイクルの P コードや Q コードの数値に 0 が指令されている
・メインプログラムが M30(または M02)で終わっていない
・サブプログラム・マクロプログラムが M99 で終わっていない
・NCVC の予約語(WorkRect,WorkCylinder,Endmill,Drill,Tap,Reamer)と思われる記述の場合、
スペル、構文の間違い等があれば指摘する
・主軸回転数が指令されていない状態で、主軸正転命令(M03)が指令している
(工具交換後に改めて主軸回転数を指令し直さなければならない機種の場合は、オプションで設定)
・工具交換前に実行すべきコードが実行されていない(実行すべきコードはオプションで設定)
(実行すべきコードがない場合もオプションで設定)
・長さアドレス(X,Y,Z,I,J,K,R,Q)の数値に小数点がない(オプション) (マクロの引数は除く)
・S コードの数値に小数点がついている(オプション) (マクロの引数は除く)
・固定サイクルの P コードの数値に小数点がついている(オプション) (マクロの引数は除く)
・1 ブロックで M コードが複数指令されている(複数指令できる機種は、オプションで設定)
・工具番号 T と工具長補正番号 H が異なる(オプション)
・工具番号 T と工具径補正番号 D が異なる(オプション)
・工具呼び出し(T コード)はあるが、工具交換命令(M06)が指令されていない
(別の T コードを指令時とプログラム終了時にチェック)
・同じブロックに T コードと M06 がある場合、どちらが先に実行されるかをオプションで設定、
さらに同じブロックにないと工具交換できない機種はそうでない場合に警告、
基本的には同じブロックにあると不具合の元となるので同じブロックで指令すると警告するオプションも設定可能
・主軸正転命令(M03)の後、主軸停止命令(M05)をしていない(工具交換時とプログラム終了時にチェック)
・クーラント ON(M08)の後、クーラント OFF(M09)していない(工具交換時とプログラム終了時にチェック)
・一度も F コードが指令されていない状態で、G01・G02・G03 または固定サイクルを指令している
・ワーク座標系呼び出し(G54・G55・G56・G57・G58・G59)が指令されていない(最初の XY 移動の際にチェック) (G92 が指令されている場合と、G91 のみで動作している場合は除く)
・工具長補正 G43 が指令されていない(工具交換後初めての Z 移動の際にチェック)
(G92 が指令されている場合と、G91 のみで動作している場合は除く)
・G74・G84 のひとつ前のブロックに M29S_がない(オプション)
・M29S_がなくても G74・G84 で同期式タッピングサイクルができる機種で、
G74・G84 と同じブロックに F コード・S コードがない(オプション)
・固定サイクルで指令されている Z 座標がイニシャル点、または R 点よりも高い
・工具径補正中に XY の移動のないブロックが既定数(オプションで設定)を超えて続く
・工具径補正がキャンセルされていない(工具交換とプログラム終了時にチェック)

・工具径補正のスタートアップブロックで円弧補間が使われている
・工具径補正のキャンセルブロックで円弧補間が使われている
・固定サイクルがキャンセルされていない(工具交換とプログラム終了時にチェック)
(他の移動コードでの固定サイクルキャンセルを認める場合は、オプションで設定)
・主軸回転数指令の値が上限設定値を超えている(オプション)
・主軸回転数指令の値が下限設定値よりも小さい(オプション)
・送り速度指令の値が上限設定値を超えている(オプション)
・送り速度指令の値が下限設定値よりも小さい(オプション)(G74・G84 のときはピッチを示す場合があるので除く)
・主軸を正転させずに G01Z_や固定サイクルを指令している(同期式タッピングサイクルを除く)
・クーラントを出さずに G01Z_や固定サイクルを指令している(オプション)
・アブソリュート指令(G90)で原点復帰している
・Z 軸を早送りや直線補間で下げた後、Z プラス方向に逃がさずに XY 軸を原点復帰している
・Z マイナス方向に直線補間してその高さのまま、早送りで XY 移動している(G40 のブロックは除く)
・Z マイナス方向に直線補間して(XY 移動した後)、主軸を上に逃がさずに主軸の回転を停止(M05)している
・1 つのブロックで Z マイナス方向に直線補間する距離が、設定値(オプションで設定)を超えている
・Z マイナス方向に早送りしてその高さのまま XY 方向に直線補間、または円弧補間している(オプション)
・1 つのブロックで Z プラス方向に直線補間で上昇させる距離が、設定値(オプションで設定)を超えている(効率化のため早送りすべき)
・NCVC の予約語で主軸工具がタップであることを明示している場合で、タッピングサイクル以外の固定サイクルを指令している
・工具交換後 X0Y0 に早送りで移動していて、その直後の移動が XY のどちらかの移動しかない、
または固定サイクルで XY どちらの移動もない場合、CAM ソフトからの出力プログラムに
手入力で工具交換と一連のコードを足した可能性があるので警告(オプション)
・G73・G81・G82・G83 のサイクルで穴あけが実行されており、同一のプログラム内で同じ XY 座標に G74・G84・G85・G86・G88・G89 が実行されている場合で、その Z 座標の指令値が、
すでに開けられた穴の Z 座標に対して設定値(オプションで設定)以上浅いものになっていない
・G74・G84・G85・G86・G88・G89 が実行されている場合で、同一のプログラム内で同じ XY 座標に G73・G81・G82・G83 のサイクルが実行されていない(オプション)

(これ以降、カスタムマクロ関係の誤り)
・マクロの呼び出し多重度 4、マクロを含めたサブプログラムの呼び出し多重度 8 を超えている
・G66 に対応する G67 がない
・マクロ関数(SIN や ABS など)の直後に [がない
・アドレス直後にマクロ関数があり、[ ]で囲まれていない
・アドレス直後に数値、それに続く四則演算記号があり、[ ]で囲まれていない

・#で始まるブロックに=がない。または、#の前にアドレスがない
・数値や] の直後に、関数や変数が演算子なしで記述されている
・[ ]が閉じていない
・IF 構文や WHILE 構文が正しくない
・比較演算子 EQ・NE・GT・LT・GE・LE を誤って=・<>・>・<・>=・<=と記述している
・WHILE～DO 文・DO 文に対応する END ブロックがない
・WHILE～DO 文・DO 文の繰り返し回数が上限数(オプションで設定)を超えている(無限ループ対策)
・GOTO 文で指令している N 番号のブロックがない
・GOTO の呼び出し回数が上限数(オプションで設定)を超えている(プログラム内無限ループ対策)

4. 最後に

NC_Code_Checker.pl を使えば、実行エラーになりそうな NC コードや実行中に危険を及ぼす可能性のある NC プログラムは概ね検出できると思います。しかしながら、全ての誤りを検出できるわけではありませんし、切削条件不良や実機での設定ミスなども起こりえます。実際の加工の際は細心の注意を払って作業を行ってください。
NC プログラムの教育・学習の効率化や、プログラミング作業における生産性の向上、プログラムミスによる事故の未然防止などの一助になれば幸いです。