World Academy of Science, Engineering and Technology
International Journal of Mathematical and Computational Sciences
Vol:6, No:8, 2012

Feature-Based Machining using Macro

M. Razak, A. Jusoh, A. Zakaria

***Abstract*—**This paper presents an on-going research work on the implementation of feature-based machining via macro programming. Repetitive machining features such as holes, slots, pockets etc can readily be encapsulated in macros. Each macro consists of methods on how to machine the shape as defined by the feature. The macro programming technique comprises of a main program and subprograms. The main program allows user to select several subprograms that contain features and define their important parameters. With macros, complex machining routines can be implemented easily and no post processor is required. A case study on machining of a part that comprised of planar face, hole and pocket features using the macro programming technique was carried out. It is envisaged that the macro programming technique can be extended to other feature-based machining fields such as the newly developed STEP-NC domain.

***Keywords*—**Feature-based machining, CNC, Macro, STEP-NC.

I. Introduction

A machining feature can be defined as a closed volume in space such that upon completion of machining operation, there shall be no material left in the feature. And at the same time, there shall be no material removed outside the feature [1]. Since a feature defines shape, faces and location of the part, it can be easily revised and updated. In a nutshell, features are not only describing shapes but also describe how to produce those shapes using CNC machine [2].

Machining features have now been standardized [3] and an object oriented methodology is used to define them. Because of this, it is common to find that majority of current commercial CAM systems are feature-based. A typical CAM system converts design feature to machining features by feature recognition [2] and extraction approaches. Ouyang & Shen [2] proposed a three modules algorithm, namely design feature, feature conversion and machining feature. The system depends on finding of the inner link between independent, composite and convex features. Thus for a given part with design features, a CAM system will first identify the machining features and automatically generate the standard NC part program based on ISO 6983. Due to nature of the current standard, the CAM generated part program can be very long and occupy large memory space [4]. This paper deals with an alternative method of implementing feature-based machining using macro programming.

M. Razak is with Universiti Kuala Lumpur Malaysian Spanish Institute, Kulim, 09000 Kedah, Malaysia (e-mail: alhapis@msi.unikl.edu.my).
A. Jusoh and A. Zakaria is with Universiti Kuala Lumpur Institute of Product Design and Manufacturing, 119 Jalan 7/91, Tmn Shamelin Perkasa, 56100 Kuala Lumpur, Malaysia (e-mail: abrahim@iprom.unikl.edu.my, dzakaria@iprom.unikl.edu.my).

II. Feature-Based Machining

Most of recent papers on feature-based machining primarily focus on new standards ISO 14649 (data model for computerized numerical control) and ISO 10303-238 (application interpreted model for numerical controllers). Both are collectively known as STEP-NC [3]. In view of the fact that the two standards were developed by different committees [3], there are major differences in their implementation. While the former supports only one kind of data exchange from CAM to CNC, the later can communicate all the information required to manufacture a product across the supply chain from design to manufacture. However ISO 14649 has an edge over ISO 10303-238 in terms of program size and ease of programming. An effort is being carried out to integrate both standards into one [3].

Machining feature as defined in ISO 14649 ARM (application reference model) is shown in Fig.1. Because of its object oriented structure, C++ programming language is normally used in its proposed STEP-NC part program implementation. One of the serious drawbacks found in the proposed technique is the lack of feedback ability from user to designer. Also its complex entity cross-referencing makes the part program not human readable. Radical approach of STEP-NC requires a totally new kind of CNC controller. Hence current implementation on existing machine uses 'plug and play' method where the final STEP-NC codes have to be converted to the conventional NC codes. This paper proposes an alternative method to implement feature-based machining using macro technique, and by extension of its adaptation to implement STEP-NC machining, without the need of machine-specific post-processor.

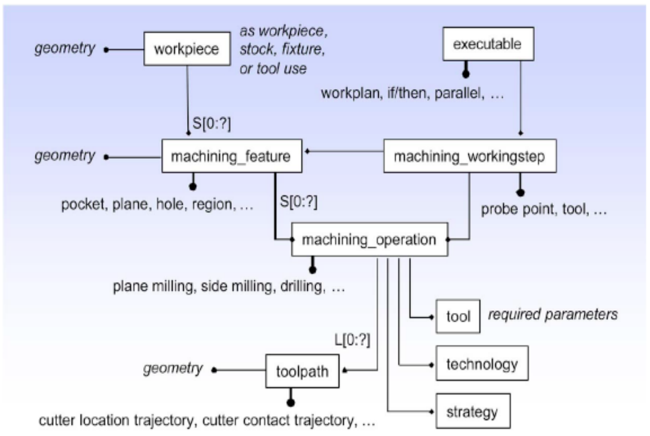
geometry
workpiece
as workpiece, stock, fixture, or tool use
executable
workplan, if/then, parallel, ...
S[0:?]
geometry
machining_feature
machining_workingstep
pocket, plane, hole, region, ...
S[0:?]
probe point, tool, ...
machining_operation
plane milling, side milling, drilling, ...
tool
required parameters
L[0:?]
geometry
toolpath
technology
strategy
cutter location trajectory, cutter contact trajectory, ...

Fig. 1 STEP-NC ARM

III. Macro

Custom macro or user macro approach adopted in this paper is specifically referred to Fanuc or Fanuc compliant CNC controller. For many years, it has been ignored by both industry and academia [4]. In general, it is similar to subroutine except it has additional control features [5]. These features allow macros to be programmed in similar manner as other high level language programming such as VB or C. More importantly, the user may include elements of intelligence in decision making by using IF, THEN or WHILE functions.

In the current standard of ISO 6983 (also known as 'G-code'), NC code is produced according to the path of cutter location data (CL) with respect to the machine axes [3]. In macro, similar to the high level programming language, users have the ability to control the variables [5]. This method is powerful and useful in improving the programming efficiency [4]. Unlike the normal NC program, macro structure offers solutions for much more sophisticated part machining [6]. There are two programs involved in this type of programming technique. They are the main macro program and macro subprogram. Macro can be called not only by the main program, but also by any other subprogram or macro as well [6]. The normal subprogram always uses fixed data, which values cannot be changed. On the other hand, Macro uses variable data that can be changed very quickly [6]. Therefore, this technique is appropriate to be employed in feature-based machining [7].

IV. Concept of Implementation

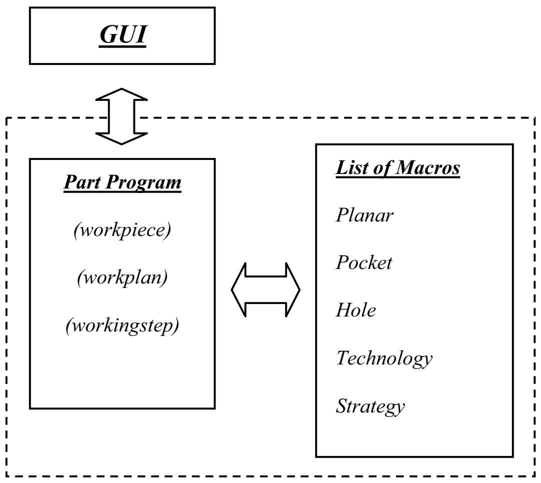

Fig. 2 Concept of IMS

In this research, a new machining system called Intelligent Machining System (IMS) is being developed. The concept of implementation for IMS is shown in Fig.2, which comprises of two modules. The first module contains machining function routines such as determination of machining conditions, selection of machining cycles and machining strategies written in macros resided in the machine memory. The second module is a part program generator where user defined workpiece geometry and features are input via GUI (graphical user interface). Final part program generated will consist of process plan and machining steps. In the operation, this part program will interact with the contents of the machine memory via macro call commands.

The machining parameters can be retrieved and stored by means of global variables of the macro. This advantage enables user to view the machining history.

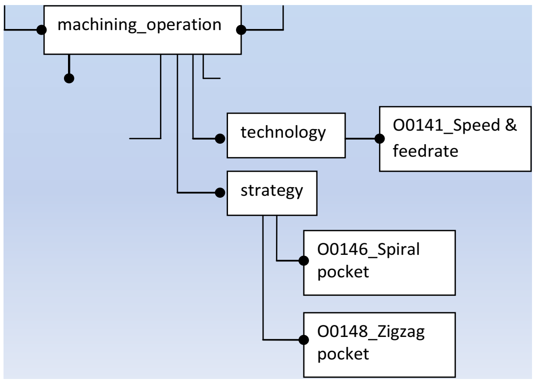

Fig. 3 Example of how macros are implemented

In the macro approach, macro program number can represent a function. In the example as shown in Fig.3, there are two strategies for pocketing. They are spiral and zigzag. These two strategies are represented by a macro program number. Similarly, speed and feedrate calculations can be represented by a macro program number.

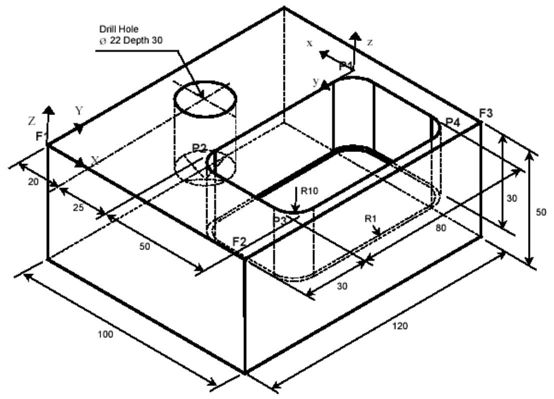

Fig. 4 Example of workpiece

An example from ISO 14649-11 [8] as shown in Fig.4, comprised of three features namely planar face, hole, and pocket has been selected as a case study. To produce the part, IMS will call macros for cutting operation according to workplan steps. In this case, it starts with planar face and continues with hole, and finally the pocket.

The flowchart of macro execution of machining feature is shown in Fig.5. The system will first determine the type of feature to be machined. Then, it will call macros for executing the operation. If there is more than one workingstep to produce the feature, specific macros are called according to the sequence. After the feature is completed, the system will check for the next feature.

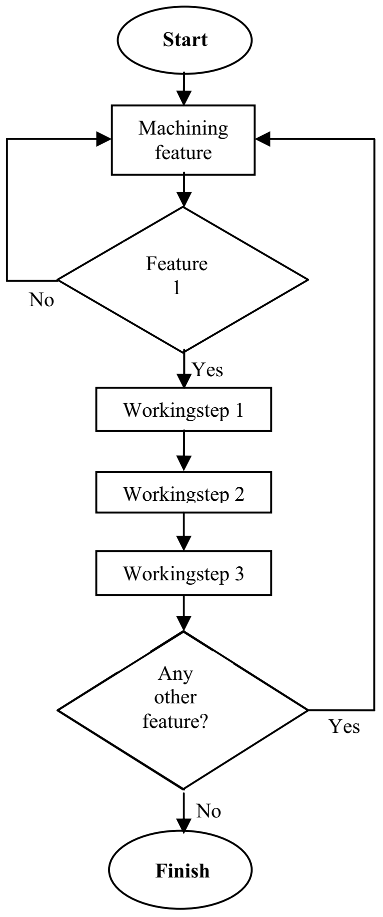

Fig. 5 Macro execution of machining feature

In macro program, variable numbers represent specific information declared by user or referred to earlier part programs. For instance, #1 is representing how many features involved and #2 refers to what feature to be machined. Every single function or operation can be written separately using macros. For example, if the main program calls program number 0100 for workingstep 1, the system will then find program number 0100 to be executed.

To generate a new program for different design, the user needs only to redefine the variables in the macro. As there is no new program added, the memory used in the controller remains more or less constant.

V. Methodology

The example given in the ISO 14649-11 [8] has been used as the case study (as shown in Fig.4). A process planning for the part is carried out as a prerequisite before the actual machining. The origin point is clearly shown in this example.

By applying the developed IMS, an experiment was carried out on a chemical wood workpiece to confirm the IMS functionality. High speed steel (HSS) cutting tools were used in this study. Basically, all dimensions in the example were utilized for graphical simulation. Fanuc Robodrill α-T14ίFse machine with Fanuc Series 31ί-Model-A controller was used for this purpose. Single machining process for planar, hole, zigzag rectangular pocket and spiral rectangular pocket were carried out. Subsequently combination of all features machining was carried out twice: one was for pocket with zigzag strategy and the other for spiral.

VI. Result

```
%

O0140(MAIN PROGRAM IMS)

 (Blank Definition)

G1902B100.D120.H50.I0.J0.K0.

(user input: start)

#1=3(NO_FEATURE)

#2=1(FEATURE_i/START FEATURE)

#3=1(WORKPIECE MATERIAL)(M)

 (Feature 1: PLANAR)

#4=100(WORKPIECE LENGTH, I)(A)

#5=120(WORKPIECE WIDTH, J)(B)

#6=2(PLANAR_HIGH TO CUT, K)(C)

#7=1(PLANAR DEPTH OF CUT)(U)

(Feature 2: HOLE)

#520=20.(HOLE LOCATION, X)(X)

#521=60.(HOLE LOCATION, Y)(Y)

#522=30.(HOLE DEPTH)(Z)

#523=2(DRILLING STRATEGY)(S)

(Feature 3 : POCKET)

#9=2(POCKET STRATEGY)(1=SPIRAL, 2=ZIGZAG)

#527=45.(POCKET LOCATION, X)(X)

#528=30.(POCKET LOCATION, Y)(Y)

#529=50(POCKET LENGTH, I)(I)

#530=80(POCKET WIDTH, J)(J)

#531=30(POCKET DEPTH)(K)

#532=20(ENDMILL DIAMETER)(D)

#533=1(POCKET DEPTH OF CUT)(V)

(user input : end))
```

```
IF[#1EQ0]GOTO9020
IF[#1GT3]GOTO9021
IF[#2EQ0]GOTO9022
IF[#2GT3]GOTO9023
IF[#3EQ0]GOTO9024
IF[#3GT2]GOTO9025
#130=#1(NO_FEATURE)
#131=#2(FEATURE_i)
#132=#9(POCKET STRATEGY)
#510=[0-#6](Z-OFFSET AFTER PLANAR CUT)
WHILE[#131LE#130]DO1
IF[#131EQ1]GOTO1(PLANAR)
IF[#131EQ2]GOTO2(HOLE)
IF[#131EQ3]GOTO3(POCKET)
(workingstep : PLANAR))
N1G65P0142A#4B#5C#6D#8U#7M#3
GOTO20
(workingstep : HOLE)
N2G65P0144X#520Y#521Z#522S#523M#3A#524B#525C#526
GOTO20
(workingstep: POCKET )
N3(SELECT POCKET STRATEGY)
IF[#132EQ1]GOTO10(SPIRAL)
IF[#132EQ2]GOTO11(ZIGZAG)
(SPIRAL)
N10G65P0146X#527Y#528I#529J#530K#531D#532V#533W#534M#3
GOTO20
(ZIG_ZAG)
N11G65P0148X#527Y#528I#529J#530K#531D#532V#533W#534M#3(POCKET_
ZIGZAG)
N20#131=#131+1
END1
GOTO100
(ERROR MESSAGES)
N9020#3000=120(PLEASE INPUT NO. OF FEATURE)
N9021#3000=121(EXCEED NO. OF FEATURES)
N9022#3000=122(SELECT START FEATURE)
N9023#3000=123(WRONG FEATURE CODE)
N9024#3000=124(SELECT MATERIAL 1 OR 2)
N9025#3000=125(WRONG MATERIAL CODE INPUT)
N100M30
%
```

Fig. 6 IMS main program

The example of main program of IMS is shown in Fig.6. From the observation during the machining process and inspection of finished product, it was verified that IMS has met its expectations. All the features were produced with correct dimensions. Fig.7 and 8 shows the graphic animation on CNC controller and finished part.

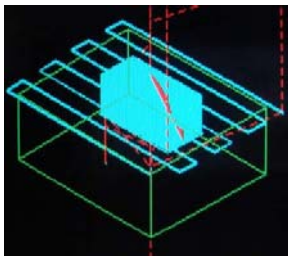
Fig. 7 Simulation on CNC controller

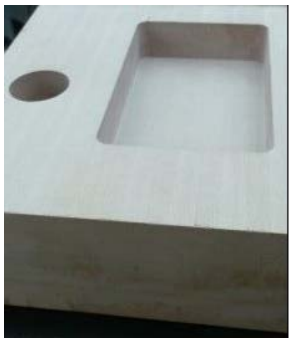
Fig. 8 Front view simulation

Memory size on the controller for the whole system is only 16 Kbytes. With this system, the program's memory size remain constant regardless the changes of feature size. This is due to the fact that the feature size changes can be done by simply redefining the feature variables.

VII. Conclusion

The IMS has successfully machined the features using the macro programming technique, without the need of machine-specific post-processor. In term of part accuracy, there was no difference between this system and the normal standard or commercial CAM system. The difference in the accuracy of the machining results would depend on the machine and tool conditions. As can be seen in Fig. 6, the macro program is simple, concise and human readable. A simple interactive GUI can now be developed to automatically generate the part program. With the macro technique, user can expect comparatively much smaller program size compared to the CAM generated NC, and also with the ease of implementation. The ability to record changes made during the actual machining process is distinct advantage of the macro. It is envisaged that the macro programming technique can be extended to other feature-based machining fields such as the newly developed STEP-NC domain. In view of absence of hitherto new type of controller that can implement the STEP-NC approach, the work in this research opens an exciting opportunity to venture into. Contingent to the availability of the new STEP-compliant controller, this approach is more practical and a way forward in implementing the STEP-NC without the need to disrupt the current entrenched G-code system.

Acknowledgment

The authors acknowledge Universiti Kuala Lumpur for funding this research with Short Term Research Grant (STRG) and the technical support from UniKL Institute of Product Design and Manufacturing.

References

[1] Peter Smid, *CNC Programming Handbook: A Comprehensive Guide to Practical CNC Programming*, Industrial Press, 2003.

[2] Bor-Tyng Sheen and Chun-Fong You, “Machining feature recognition and tool-path generation for 3-axis CNC milling,” *Computer-Aided Design*, vol. 38, pp. 553-562, 2006.

[3] X.W. Xu and S.T. Newman, “Making CNC machine tools more open, interoperable and intelligent-a review of the technologies,” *Computers in Industry,* vol. 57, pp.141-152, 2006.

[4] Muhammad Al’Hapis Abdul Razak and A. Zakaria, “Review on the evolutions of CNC programming methods,” 2nd Colloquium on Manufacturing Technology, Kedah, 2010.

[5] Peter Smid, *FANUC CNC Custom Macros Programming Resources for Fanuc Custom Macro B Users*, Industrial Press, 2005.

[6] Fanuc series 30i/300i/300is-Model A, User’s Manual, Common to Lathe System/Machining Center System, Vol. 1-3.

[7] M. Al’Hapis A. Razak and A. Zakaria, “A framework for a feature based machining using macro,” *Applied Mechanics and Materials*, vols. 110-116, pp. 1711-1715, 2012.

[8] ISO/FDIS 14649-11: 2002, Data Model for Computer Numerical Controllers-Part 11: Process Data for Milling, ISO/TC 184/SC 1/WG7, 2002.

[9] ISO/FDIS 14649-10: 2002, Data Model for Computer Numerical Controllers-Part 10: General Process Data, ISO/TC 184/SC1/WG7, 2002