2013•2014
FACULTEIT INDUSTRIËLE INGENIEURSWETENSCHAPPEN
master in de industriële wetenschappen: energie

Masterproef

Study and create the post-processor for CAD/CAM software for 5 axes CNC milling

Promotor :
ing. John BIJNENS

Promotor :
Prof. HOANG VINH SINH

Bart Nullens

Proefschrift ingediend tot het behalen van de graad van master in de industriële wetenschappen: energie

Gezamenlijke opleiding Universiteit Hasselt en KU Leuven

2013•2014

Faculteit Industriële ingenieurswetenschappen

master in de industriële wetenschappen: energie

Masterproef

Study and create the post-processor for CAD/CAM software for 5 axes CNC milling

Promotor :
ing. John BIJNENS

Promotor :
Prof. HOANG VINH SINH

Bart Nullens

Proefschrift ingediend tot het behalen van de graad van master in de industriële wetenschappen: energie

universiteit hasselt | KU LEUVEN

Preface

In my final year of industrial engineer automation at the Faculty of Industrial Engineering Sciences we had to make a master's thesis. I got the opportunity to make my thesis abroad. I conducted this thesis at the Hanoi University of Science and Technology (HUST).

I would like to thank ing. John Bijnens from the KULeuven, campus Diepenbeek and prof. dr. Hoang Vinh Sinh from the Hanoi University of Science and Technology for supporting and mentoring me during my thesis.

I also would like to thank a few people that made this thesis possible. I would like to thank ir. Greet Raymaekers and the department of internationalization of the University of Hasselt, they made it possible for me to go to Vietnam.

Finally, I would like to thank everyone else that helped me in any way to realise this thesis.

Bart Nullens, student at the Faculty of Industrial Engineering Sciences

Hanoi University of Science and Technology, Vietnam

June 2014

Table of contents

List of tables

List of figures

Abstract

This master's thesis aims to the study and creation of a post-processor for 5 axes CNC milling at the Hanoi University of Science and Technology (HUST). The problem with 5 axes milling is the complexity, the ability to use contour milling, compared to 3 axes milling because of the added degrees of freedom. Another problem is that a post processor is unique for every machine. The goal of this thesis is to develop a post processor for a specific 5 axes milling machine, the Mikron UCP 600.

The first part of this thesis consists of research on post-processing in CAD/CAM software Cimatron E11. The processor is written according to the Cimatron E11 standard, General Post Processing (GPP). To simplify the complex 5 axes structure, the Denavit-Hartenberg method is used. This method makes it possible to easily transform coordinates between the axes of the machine.

The post-processor is evaluated only by using software. The software tests show that the GPP uses the kinematic structure of the Mikron UCP 600. The problem of the complexity is not fixed due to fact that GPP can only handle position milling. The machine test was not possible and instead a G-code analysis is made. The possibility to use 5 axes milling is a great opportunity for the HUST. This thesis can be considered as a part of a larger development, meaning that other students can use this thesis in further research.

1 Introduction

My master's thesis is the study and creation of a post-processor for CAD/CAM software for 5 axes CNC milling. This master's thesis takes place at different locations. First I will do research at the Hanoi University of Science and Technology. Then I will continue my work at the companies HTMP and BKMech.

Hanoi University of Science and Technology is the first and largest technical university of the country. The university is located at the Hai Ba Trung district in Hanoi, Vietnam. I will do most part of my theoretical research in the school of Mechanical Engineering at the department of Metal Cutting and Industrial Instruments (MCII).

HTMP is a company that is active in the field of mold design and manufacturing. To do this they use the CAD/CAM software package Cimatron E11. This is the same software I will use for my master's thesis. My work at this company will help me better understand the process from CAD design to manufacturing molds. I will spend two weeks at HTMP studying Cimatron E11 to use it to design and to generate toolpaths and to learn about CNC manufacturing. (HTMP, 2006)

BKMech is a quality manufacturer of CNC machines and provide solutions for mechanical product shaping. My promoter, Mr. Sinh, is closely related to this company. My work at BKMech consists of working together with the engineers to find a solution for the stated problem and to guide me in the process of creating a post-processor. (BKMech, 2014)

In this master's thesis I am going to study and create a post-processor. The work can be devided into four sections:

1. Gathering information about software and hardware needed to create a post-processor;
2. The study of the cutting strategies used for 5 axes milling and research about the flaws/pitfalls of 5 axes milling;
3. Try to create a post-processor that will overcome these flaws;
4. Testing and adjusting/fine-tuning the post-processor.

2 What is a post-processor?

If we take a look at the process of creating a CNC program with the use of a CAD/CAM system we see at one end we have the computer, the software, which we use to draw the part, on the other end we have the machine which cuts the part. The translation of the CNC independent toolpath generated by the CAD/CAM software to the CNC specific code is done by the postprocessor.

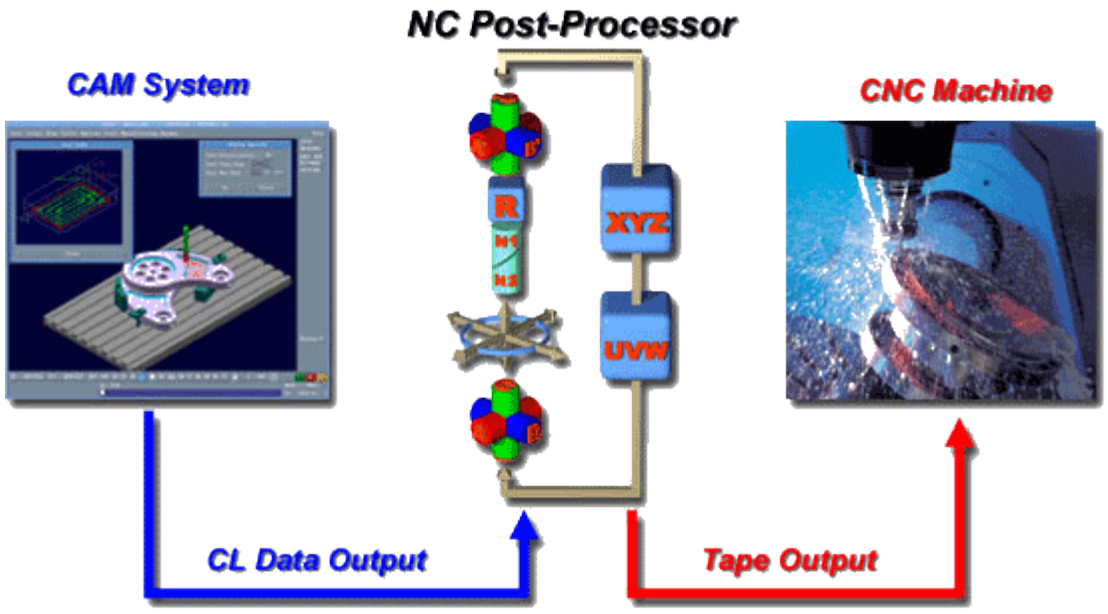


Figure 2-1: Chart showing the use of a post-processor (What is Post-Processing?)

A post-processor takes the toolpath made in CAM and converts it to usable code for the CNC machine. It translates machine-independent format to machine-dependent format. The most used independent format is APT or CLDATA which is a subset of APT. The standard for machine dependent code is G-code (ISO 6983 and DIN 6025). In the early days of CNC, the generation of CLDATA was also called geometry processing. Hence the translation of this process is called post-processing.

Post-processors are unique for every machine. On one hand this is due to the fact that every CNC machine has its own kinematic behaviour. On the other hand, every machine has its own ISO code. Manufacturers of CNC machines try to make them excel in a specific field, resulting in a variety of CNC controls to choose from. (What is Post-Processing?; Bijnens, 2011; G-code)

3 From design to product

To make a post-processor we first need to understand how the CNC machine works and how the CAD files are processed into real parts. In the next section the whole process from design to finished product will be explained. This process can be divided into three main steps: CAD, CAM and post-processing. The first two steps will be merged into a single section.

3.1 CAD/CAM

The first step of the process is to make a 3D design of the part we want. I will use the software Cimatron E11 for this.

The second step in the process is making a NC file (CAM). There are eight steps to be done in order to make a complete NC file:

1. **Load model**
   - As the first step a CAD design needs to be loaded.

2. **Cutters**
   - We define and/or add the cutters necessary for the milling process.

3. **Toolpath**
   - Set a new toolpath, this means the number of axes and the workplane.

4. **Part**
   - Select the surfaces of the part that are used in the machining simulation for comparing actual milling results to the desired product.
   - It is also possible to define a fixture part such as clamps, vised, jigs,….

5. **Stock**
   - We can use this function to specify the raw material from which the final part will be produced. In the simulation this can be used to calculate the remaining stock.

6. **Procedure**
   - Insert procedures for milling, e.g. volume milling, surface milling, finishing,…

7. **Machining simulation**
   - We can simulate both tools and machining procedures. This function will be used to determine and avoid collisions.

8. **Post-processing**
   - A post-processor is used to process the toolpaths defined in previous steps. I will go further into this topic in the next section. (Cimatron)

3.2 Post-processing

The last step is post-processing. Cimatron E11 has a tool to create a GPP (General Post Processor). This GPP translates toolpath data (CLDATA) and procedures into specific CNC commands, also known as G-code.

The GPP process consists of five stages:

1. Define the machine using DFPost
2. Write the code for the post-processor. I will use notepad++ for this stage
3. Compile the post-processor program file using Dfexf
4. Add command line in the dfexf file if an operating system file is to be run afterwards (optional).
5. The data compiled in the previous stages will be combined with toolpath data in order to produce a specific NC output program. This step is done in Cimatron E11 (step 8 from previous chapter) and can be done either internally or externally. (Cimatron)

Note: For stage 1 and 3 we need to use the Cimatron NC function. These can be found in the Cimatron Control Panel under the NC tab.

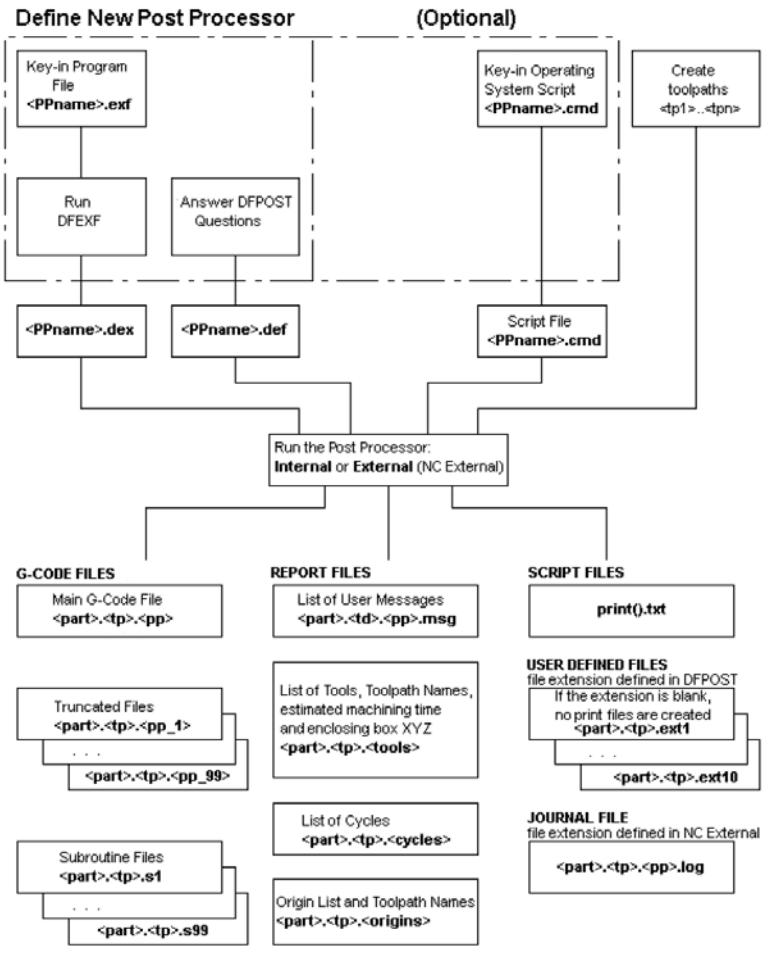


Figure 3-1: Flowchart showing the different steps (Cimatron)

3.2.1 Step 1: DFPost

In this step the machining parameters are defined. This is done in a command window and saved as a <name>.def file. To open the command window we need to go to the NC tab in the Cimatron Control Panel and click 

.

Then we get a command window as shown in figure 3-2. Here we can explore and define the machine parameters. (Cimatron)

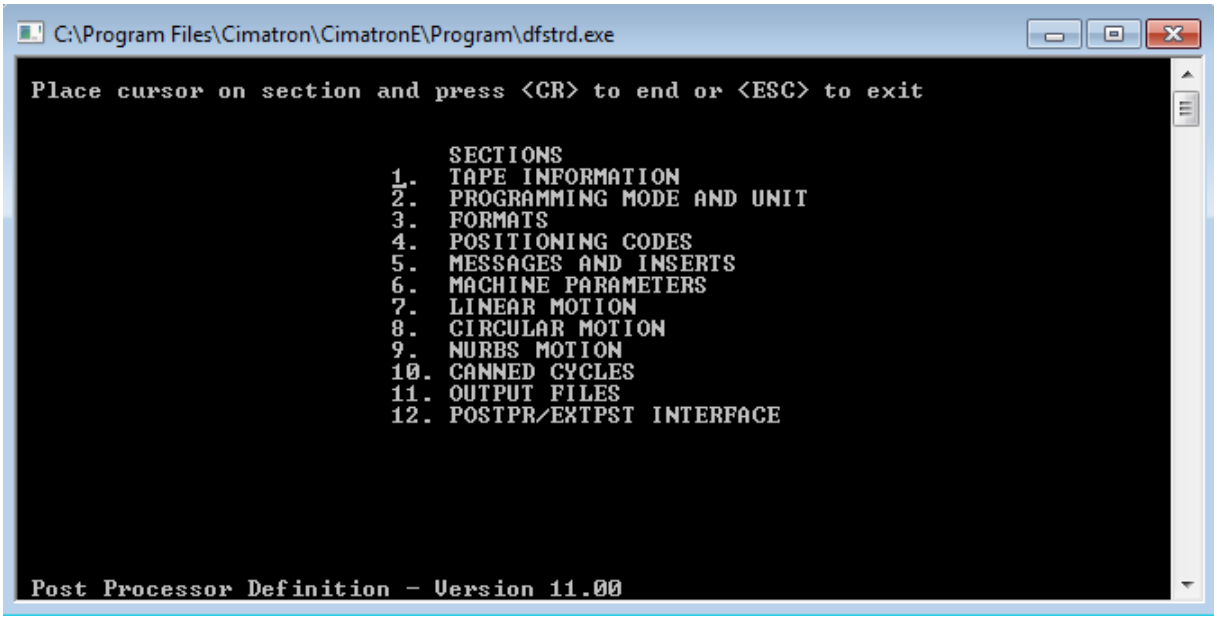


Figure 3-2: CMD-windows to define machine parameters

3.2.2 Step 2: Post processor program file

Using an editor (Notepad++) we can create a program file. This file must have the extension <name>.exf, also the file name must be the same as file name from the previous step. The layout of the program file consist of three parts: first section, body and comment lines.

- The first section contains declaration statements which set the operating conditions for the variables before block statements can be executed.
- The body is made of blocks containing executable statements. These can be system defined blocks and/or user defined blocks.
- Comment lines can be added to make it easier to read and understand file.

Declaration statements used in the first section of the program file:

FORMAT
The FORMAT statement is used to define variables. To assign a format, the name of the format type must be enclosed in parentheses and preceded by FORMAT, preceded and follow by a space. The variable(s) which are being defined follow, separated by spaces. The variable is follow by a semi-colon (;).
Example: FORMAT (COORDINATES) XHOME;

IDENTICAL
The IDENTICAL statement is used to save assignment operations. All variables appearing after the identical statement will always have the same value. If one of these values is changed by either the user of the system, the others will automatically be updated.
Example: IDENTICAL X_CURPOS X_ENDPT;

INTERACTION
The INTERACTION statement is used to define interactable variables. These variables can be assigned when running the post-processor. Interaction variables are always given an initial value.
Example: INTERACTION (COORDINATES) "ENTER_TABLEDX" TABLEDX = 0;
When we run the processor, we are able to change the value TABLEDX, making it interactable.

MODAL/NON-MODAL
Variables can be assigned to be MODAL or NON-MODAL. Modal variables are turned 'off' when their values are output and turned 'on' when the variable is assigned a new variable. Non-modal variables are never turned 'off'. Modal variables can be reset to 'on' using the statement RESET or SET_ON, even when not changed in value. In addition they can be turned off by using the SET_OFF statement. All variables are assigned modal or non-modal as defined in the DFPost stage.
Example: FORMAT (COORDINATES) PECK;
MODAL PECK;

NEW_LINE_IS
The NEW_LINE_IS statement defines the symbol which will trigger a procedure each time it is encountered, and what that procedure will be.
Example: NEW_LINE_IS ! ;
OUTPUT \J "N" SEQ;
SEQ = SEQ + SEQINCR;

SET-TABS
Some controllers require the output to be in fixed columns. The SET_TABS statement makes it possible to set the tab positions. When “TAB_” is used in a block, the output will be shifted according to the previous set position.
Example: SET_TABS 4 8 12 16;
OUTPUT \J “12345”;
OUTPUT \J TAB_ “A” TAB_ “B”;
OUTPUT \J “123” TAB_ TAB_ TAB_ TAB_ “A”

Output file:
12345
A B
123 A

Blocks used in the body of the program file:

System-defined blocks
A program file contains several system defined blocks. These blocks are activated when an event happens. For example, the “BEGINNING OF PROC:” block will be activated when a new procedure is initiated. The system-defined blocks contain all basic events like: circular motion, linear motion, cycle,….

Block Syntax Rules
Block name serve as delimiters between operations. They appear alone on a line unless qualifiers are used. The block name is always followed by a colon (:). The name is case insensitive, the result will be the same whether uppercase or lowercase characters are used.

Qualifiers
Qualifiers can be added directly behind the block name. These qualifiers are used to express different situations of a block. Qualifiers are also followed by a colon (:). For example, when “CUTTER COMPENSATION: Coff” is encountered while the cutter compensation is off, this block is activated. Same blocks can be used numerous times with a different qualifier.

User-Defined Blocks
It is possible to define your own blocks. The syntax rules for system-defined blocks apply to user-defined blocks. User-defined blocks will be executed when encountered in the toolpath file. For example, “MyBlock:” will be executed when there is a command “MyBlock” in the toolpath file. (Cimatron)

3.2.3 Step 3: Compile

Now the written code is compiled. We can do this by using the Dfexf 

function in the NC panel.

This function converts the <name>.exf file to a <name>.dex file. (Cimatron)

3.2.4 Step 4: Insert operating system script

We can run an operating system script after the post-processing is done. This can be pieces of code written in another language. In the <name>.dex file we just need to add a CALL command with the input and output parameters. This function is optional and not necessary in order for the post-processor to work. (Cimatron)

3.2.5 Step 5: Post-process

In the final step we post-process the toolpath. This is done after the seven steps in Cimatron NC. This can be done internally or externally. Internally mean we can only post-process the current file open in Cimatron E11. Externally means we can post-process multiple saved NC files. (Cimatron)

4 Machine structure

The machine I am going to use is the Mikron UCP 600 as shown in figure 4-1. This is a 5 axes CNC machine and uses a Heidenhain iTNC530 controller. Figure 4-2 shows the simulation model. Machine specifications of the Mikron UCP 600 can be found in appendix 1.

Figure 4-1: Mikron UCP 600

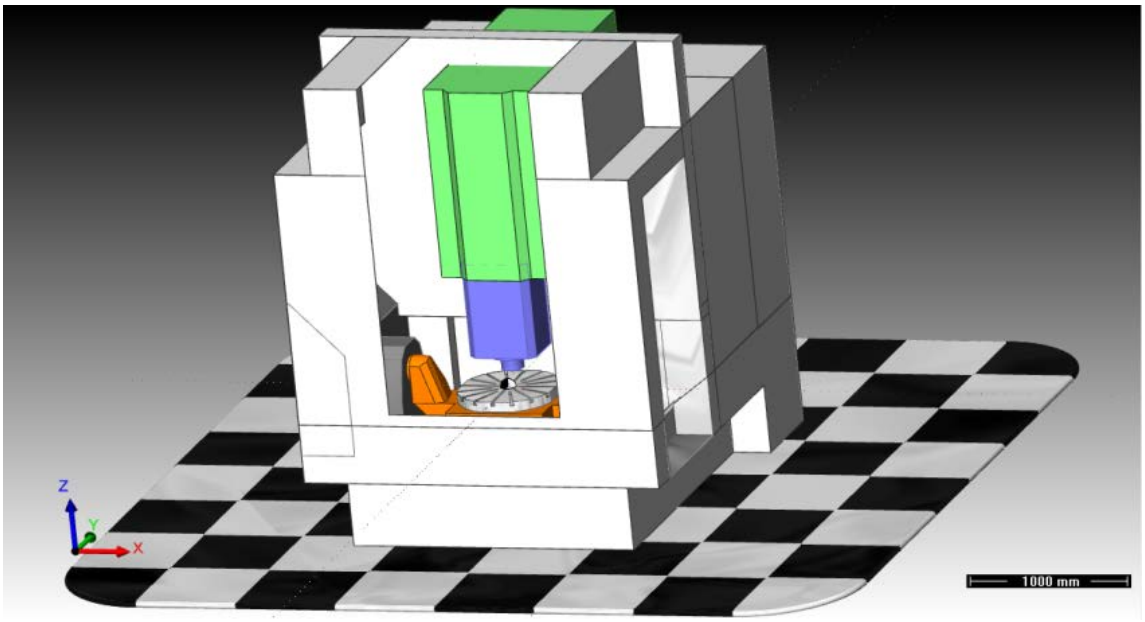


Figure 4-2: Simulation model of the Mikron UCP 600

4.1 Standard axis configuration

Figure 4-3 shows the industrial standards of axis configuration.

- XYZ are linear axes where Z is aligned with the spindle of the machine;
- ABC are rotary axes rotating around XYZ respectively;
- UVW are parallel linear axes along XYZ respectively. (Arpo, 2008)

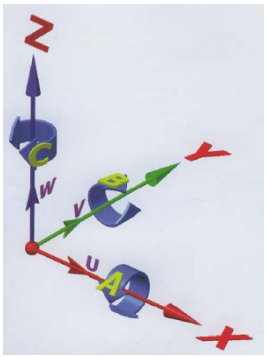


Figure 4-3: Standard axis configuration (Arpo, 2008)

4.2 Mikron UCP 600 configuration

The Mikron UCP 600 is a TTRR-T, Table/Table CNC.

TTRR-T refers to the axes of the machine. As seen from the cutting tool to the base it has two translation axes, the Z and Y axis. As seen from the part to the base it has two rotational axes and one translation axis, the C, A and X-axis. Table/Table means that the rotation of the axes is executed by the dual rotary table. The first rotary table carries the fixture and part, the second rotary table carries the first rotary table.

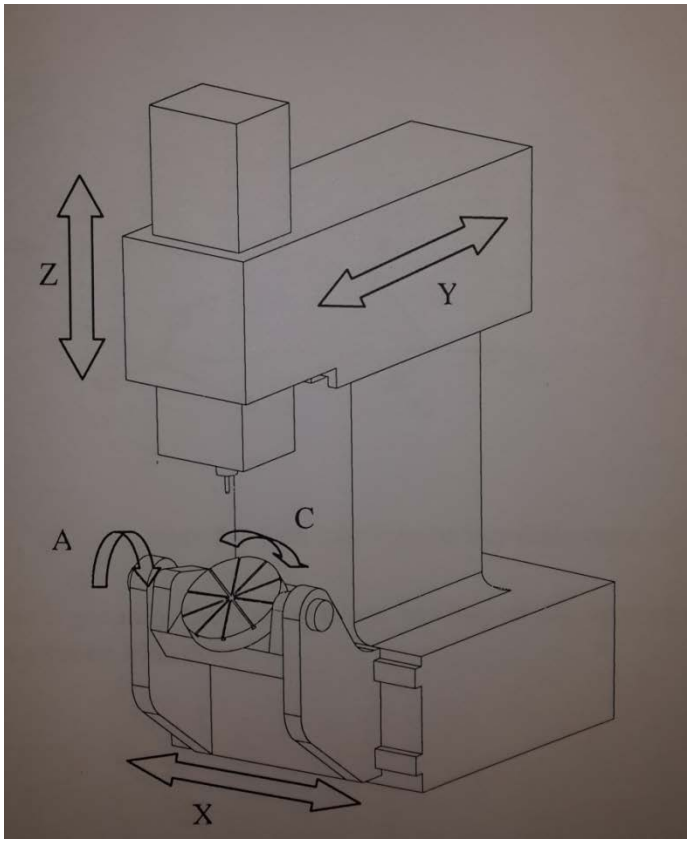


Figure 4-4: Axes configuration of the Mikron UCP 600

4.3 Heidenhain iTNC530

The Heidenhain iTNC530 can be programmed in two different languages. The first language is conversational programming. This is a language made by Heidenhain en gives more freedom to the programmer. The output files in this case will be an .H file. The other language is the DIN/ISO format. This is a standard for CNC machines. The output in this case will be an .I file. In this thesis the post-processor will be programmed to output DIN/ISO files.

4.4 Zero position

In order to cut at the right position we have to define the different zero positions used by the machine. Following positions can be defined:

- Machine Home Position (MHP), this is the position the machine moves to when you turn it on or when you select the ZERO command.
- Machine Rotary Zero Position (MRZP), this is at the intersection between the C and A axes.
- Program Zero Position (PZP), this is the UCS coordinate system in the CAM software.

All active (or local) coordinate systems are relative to the MRZP. It is necessary for the PZP and MRZP to be coincident. (Arpo, 2008)

4.5 Denavit-Hartenberg notation

For the forward kinematics of the Mikron UCP 600 the Denavit-Hartenberg method is used. In this method the kinematic model is defined regardless of the dynamic load. The Denavit-Hartenberg method contains following steps:

1. Define the frame in each joint;
2. Find the four parameters in for each transition;
3. Calculate the Denavit-Hartenberg matrix.

To define the frames in each joint following rules need to be applied:

- Z_i is the axis according to the direction of the joint. This can be either positive or negative
- X_i is defined as the common perpendicular to Z_{i-1} and Z_i. If Z_{i-1} and Z_i intersect the positive direction of X_i is undefined and can be freely assigned. Henceforth, we will us the right-hand rule in this case. If Z_{i-1} and Z_i are parallel the location of X_i is undefined, in order to define the location we will specify X_i as passing through the origin of the (i-1)st frame.
- When the Z_i and X_i axes are defined we can define Y_i using the right-hand rule.
- The X- and Y-axis of frame 0 can be assigned freely.
- The Y- and Z-axis of frame n can be assigned freely. (Angeles, 2003)

Where the transition from frame (i-1) to frame (i) is described with the following four parameters:

- α_i is the angle between the Z_{i-1} and the Z_i axes about the X_i-axis;
- a_i is the distance between the Z_{i-1} and the Z_i axes along the X_i-axis;
- d_i is the distance from the origin of frame i-1 to the X_i-axis along the Z_{i-1}-axis;
- θ_i is the angle between the X_{i-1} and the X_i-axes about the Z_{i-1}-axis. (Angeles, 2003)

These parameters are to be found for each frame transition. These parameters can then be filled into the Denavit-Hartenberg matrix. This matrix describes the rotation and translation of one frame to another. (Angeles, 2003)

$$ {}^{n-1}A_n = \begin{bmatrix} \cos(\theta_i) & -\sin(\theta_i) * \cos(\alpha_i) & \sin(\theta_i) * \sin(\alpha_i) & a_i\cos(\theta_i) \\ \sin(\theta_i) & \cos(\theta_i) * \cos(\alpha_i) & -\cos(\theta_i) * \sin(\alpha_i) & a_i\sin(\theta_i) \\ 0 & \sin(\alpha_i) & \cos(\alpha_i) & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

5 Cutting strategies

Toolpath is the path the cutter follows during the milling operation. For a machine to follow this path we must first set a cutter pattern. This is done in CAD/CAM software. The cutter pattern, however, is written in APT language (CLDATA) which the machine cannot understand. This is where the post-processor comes in. The post-processor will convert these toolpaths, which are written in CLDATA, to useable code for the machine, e.g. G-code.

5.1 Optimum work envelope

The optimum work envelope is the space in which the machine's rotary axes rotate about the same diameters. This will be explained in following example.

Suppose we want to do the machining of a human head as seen in figure 5-1. In this picture we can see that there is a very big offset between de MRZP and the cutting point. This will result in an uneven rotation of the A en C axes. Even though the angular values are the same, the A axis will have a much bigger circumferential speed.

A way to solve this problem is to put the object close to the MRZP, as shown in figure 5-2. However sometimes we cannot avoid this. Another way to solve this problem is the Inverse Time Feedrate (Arpo, 2008).

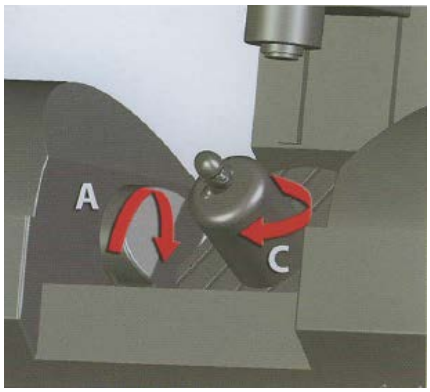


Figure 5-1: Part close to MRZP (Arpo, 2008)

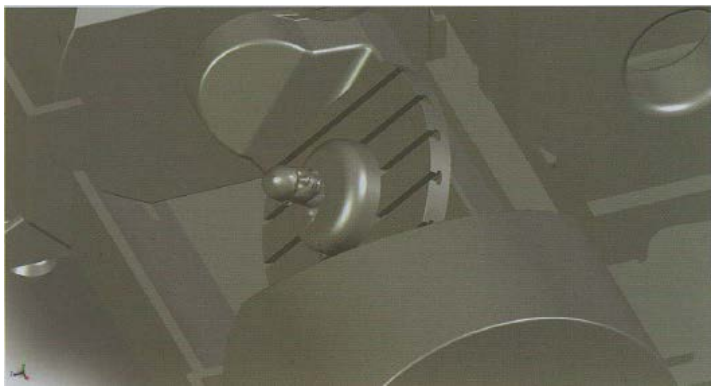


Figure 5-2: Part placed far from MRZP (Arpo, 2008)

5.2 Inverse time federate

During the simultaneous movement of the rotational and translation axes, we want to maintain the correct cutter location. Due to the difference of the interpolation algorithms used for translation and rotation the simultaneous ending cannot be guaranteed. The solution for this problem is ‘inverse time feedrate’.

The inverse feed is usually generated by the post-processor and makes sense only when there is a simultaneous movement of translational and rotational axes. The CAM system generates very fine steps based on the specified curve tolerance which indicates the allowed deviation (chord height) of the linear approximation with respect to the curve. For each of the endpoints of this linear approximation a vector orientation of the milling tool is provided. The speed by which the position change of the tool is carried out is now not determined by the feed, but by the time that is needed to do the position change. As it are very small steps done by the linear approximation the time needed to do these steps are very small. Hence the reciprocal is taken so the end result is a much bigger number. This explains the name 'inverse time feed'.

5.3 Tilt angels

Under certain conditions the tool cannot maintain normal to its cutting surface. In this case we have to tilt the cutting tool. Finding the best tilting angle is very important if we want to keep the toolpath collision free. In figure 5-3 we can see a toolpath without tilting angles. This would result in a collision in the left lower corner. Also the upper right corner can be smoothened by using a tilt angle. In figure 5-4 we can respectively see the cutting tool having no-, leading-, lag- and side angle. In figure 5-5 we can see the toolpath with the tilt angles.

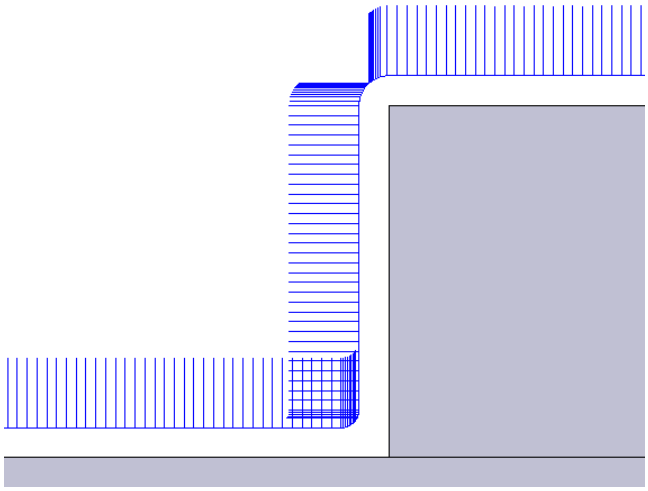


Figure 5-3: Toolpath without tilt angles (End & jaje, 2008)

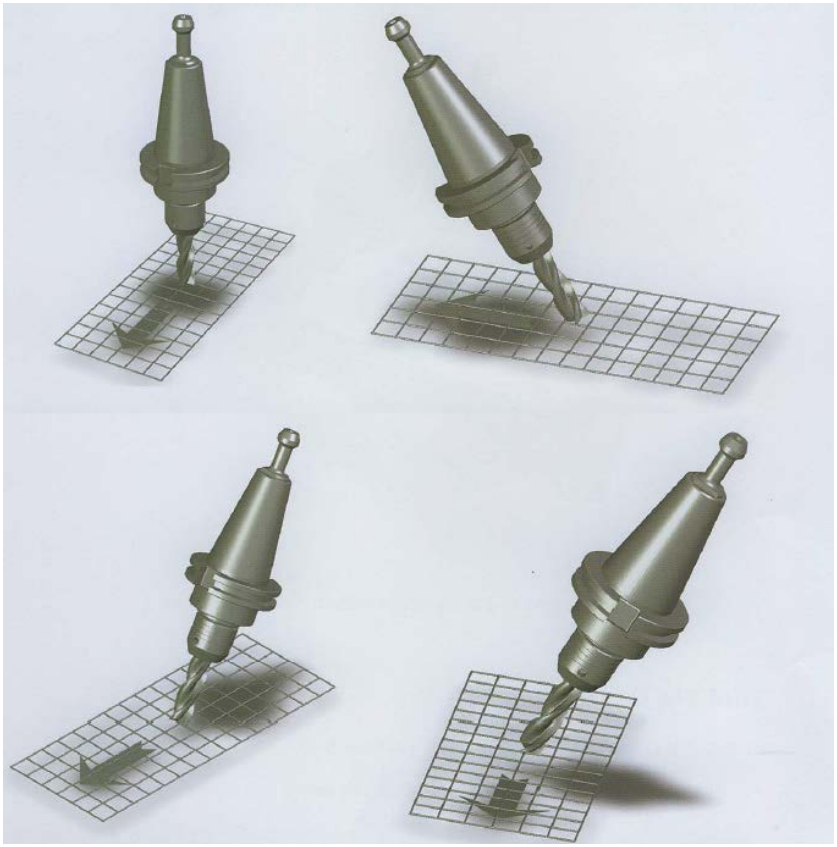


Figure 5-4: Tilt angels from top left to bottom right: no angle, lading angle, lag angle, side angle (Arpo, 2008)

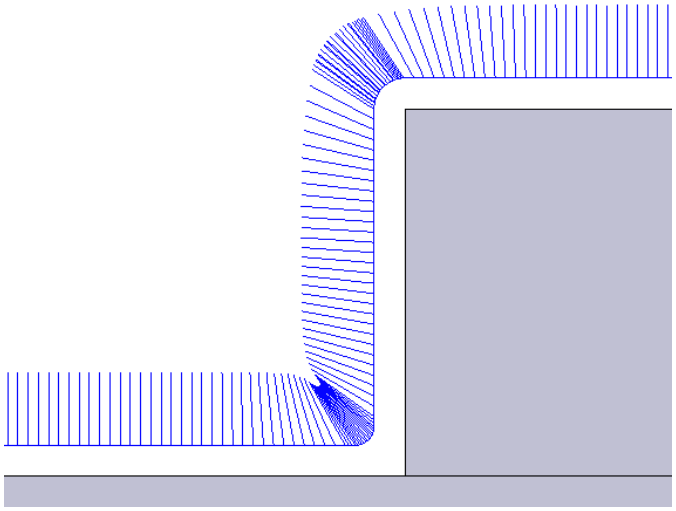


Figure 5-5: Toolpath with tilt angles (End & jaje, 2008)

5.4 Additional issues

5.4.1 Dovetail

The dovetail effect will arise when we insert a straight tool into to the centreline of a cylindrical part and rotate the part, as shown in figure 5-6 . If we want to avoid this problem we need to use an offset, which is shown in figure 5-7. (Arpo, 2008)

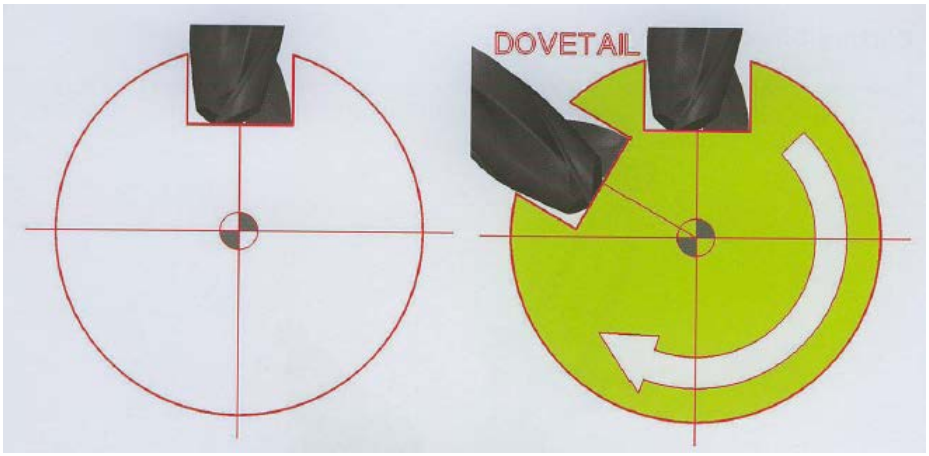


Figure 5-6: Dovetail effect (Arpo, 2008)

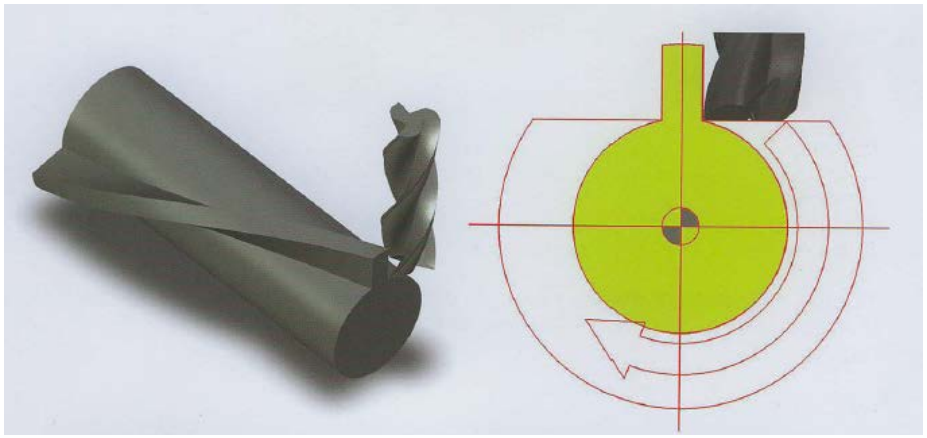


Figure 5-7: Offset used to avoid the dovetail effect (Arpo, 2008)

5.4.2 Cutting direction

For 3 axes CNC machines it is easy to determine the cutting direction as the cutting tool is always perpendicular to the part. When using 5 axes it is possible to adjust the cutting direction by adjusting the lead/lag angle. As shown in figure 5-8 different angels can result in the same cut. The direction will be most dependent on the kind of tool used. For example if we use hollow-center or non-bottom cutting tools it is highly advised to use a lead angle (top example in fig. 5-8). (Arpo, 2008)

Figure 5-8: Upper example shows a lead cut, lower example shows a lag cut (Arpo, 2008)

5.4.3 Roughing

For roughing it is often necessary to use long tools. This, however, is dictated by the part features. Common example of this problem is an impeller which consists of tall blades with small gaps between them. This will cause large side-load onto the cutting tool which can cause the tool to deflect , vibrate, chatter, poor surface finish or drastically shorter tool life.

The best way to solve this issue is to use plunge milling. This means that instead of milling sideways, we will plunge the cutting tool vertically in the part, retract, then move the part a bit sideways and plunge in again (figure 5-9). (Arpo, 2008)

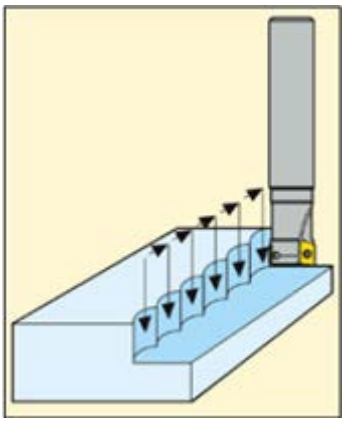


Figure 5-9: Plunge milling

5.5 Impact on post-processing

All previous stated issues have a big impact on designing a post-processor. The processor must be able de determine when to use specific tilt angles depending on the cutter pattern and cutting strategy.

The processor must also have some intelligence in detecting rotary limits and automatically retract and reposition. For a given toolpath the machine can use two different starting position to start cutting. These two positions 180° rotated from each other. Because of machine limits, however, only one position can be reached. The post-processor should be able to detect this.

Another problem is pole singularities. These singularities occur when two joints line up. An intelligent post-processor will try to evade these situations (Arpo, 2008).

6 Machine simulation

Machining simulations can be split up into two big groups: CAM simulation and G-code simulation. CAM simulation is usually integrated in the CAD/CAM software, but can also be a third party program. Both groups can be subdivided into tool and machining simulation.

CAM simulation simulates the CLDATA. In tool simulation, the removed material will be simulated, as well as tool collisions. In the machine simulation the kinematic behaviour will be simulated. This is useful to check for machine limits and collisions between machine, part, holder and spindle.

G-code simulation is used to simulate the generated G-code. G-code simulation will replicate the exact process used by the specific CNC machine. In G-code simulation, the machining simulation is preferred. This simulation uses a reverse post-processor in order to replicate the exact kinematic behaviour of the CNC machine. Examples of G-code simulation software are: Vericut by CGTECH, NCSimul, Predator Virtual CNC,… The downside is that the software is very expensive. Tool simulation for G-code can be done with freeware (trial versions) like MCU (MetaCut Utilities) or CIMCO Edit v7.0. This software simulates only the toolpath and not the kinematic behaviour. The software is very limited compared to machining simulation software, but it is possible to get the same result. It will take more time and needs a lot of insight.

In this thesis I will use the machine simulation software in Cimatron E11 to check the kinematic behaviour and to check for collisions. Then I will post-process the CLDATA to G-code. To simulate G-code I will use tool simulation in order to check the correct use of kinematic model.

7 Developing the post-processor

The development of the post-processor starts with defining the machine structure and kinematics. Herefore the Denavit-Hartenberg method is used. Next will be the definition of the machining codes in DFPost. Next the program file will be written and tested using software. Finally the post-processor will be tested and evaluated.

7.1 Denavit-Hartenberg method applied on the Mikron UCP 600

When we apply the rules stated in chapter 4.4 we get the following configuration (More detailed drawings can be found in appendix 2:

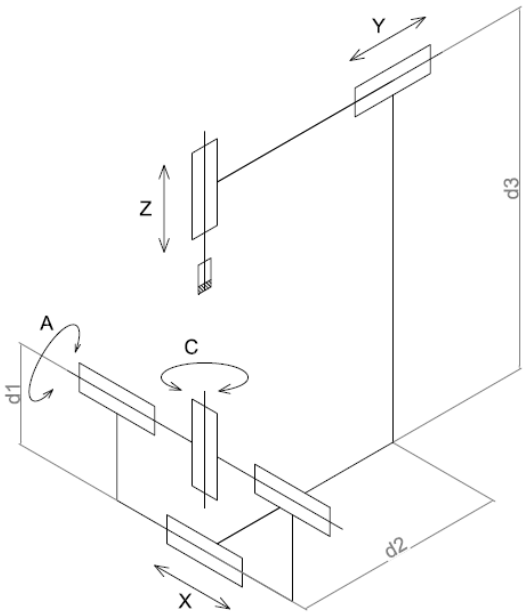


Figure 7-1: Direction of the axes

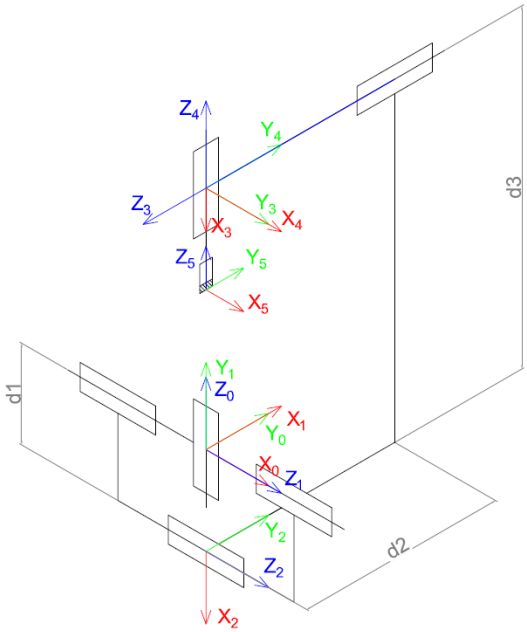


Figure 7-2: Kinematic model

The transition parameters of the Mikron UCP 600:

	θ_i	d_i	a_i	α_i
1	θ_1	0	0	π/2
2	θ_2	0	d_1	0
3	0	X	$-d_3$	π/2
4	π/2	Y	0	-π/2
5	0	Z-d_4	0	0

Table 7-1: Denavit-Hartenberg parameters

With $\theta_1 = \frac{\pi}{2} - C$, $\theta_2 = -\frac{\pi}{2} - A$, where C and A are the angles of the respective axes. X, Y and Z are the positions of the respective axes. d_1 and d_3 are parameters depending on the machine structure.

To transform the position from frame 0 to frame n we use the Denavit-Hartenberg notation:

$$ {}^{n-1}A_n = \begin{bmatrix} \cos(\theta_i) & -\sin(\theta_i) * \cos(\alpha_i) & \sin(\theta_i) * \sin(\alpha_i) & a_i \cos(\theta_i) \\ \sin(\theta_i) & \cos(\theta_i) * \cos(\alpha_i) & -\cos(\theta_i) * \sin(\alpha_i) & a_i \sin(\theta_i) \\ 0 & \sin(\alpha_i) & \cos(\alpha_i) & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

(Denavit–Hartenberg parameters)

For the Mikron we get followingmatrices:

$$ H_1 = \begin{bmatrix} \cos(\theta_1) & 0 & \sin(\theta_1) & 0 \\ \sin(\theta_1) & 0 & -\cos(\theta_1) & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ H_2 = \begin{bmatrix} \cos(\theta_2) & -\sin(\theta_2) & 0 & d_1 \cos(\theta_2) \\ \sin(\theta_2) & \cos(\theta_2) & 0 & d_1 \sin(\theta_2) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ H_3 = \begin{bmatrix} 1 & 0 & 0 & -d_3 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & X \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ H_4 = \begin{bmatrix} 0 & 0 & -1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & Y \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ H_5 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & Z - d_4 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ H_{1-5} = H_1 * H_2 * H_3 * H_4 * H_5 $$

$$ H_{1-5} = \begin{bmatrix} \sin(\theta_1) & -\cos(\theta_1)\sin(\theta_2) & -\cos(\theta_1)\cos(\theta_2) & H[1,4] \\ -\cos(\theta_1) & -\sin(\theta_1)\sin(\theta_2) & -\sin(\theta_1)\cos(\theta_2) & H[2,4] \\ 0 & \cos(\theta_2) & -\sin(\theta_2) & H[3,4] \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ \mathrm{H}[1,4] = -\cos(\theta_1)\cos(\theta_2)(Z - d_4) + \cos(\theta_1)\sin(\theta_2)Y - \cos(\theta_1)\cos(\theta_2)\, d_3 + \sin(\theta_1)\, X + \cos(\theta_1)\cos(\theta_2) d_1 $$

$$ \mathrm{H}[2,4] = -\sin(\theta_1)\cos(\theta_2)(Z - d_4) + \sin(\theta_1)\sin(\theta_2)Y - \sin(\theta_1)\cos(\theta_2)\, d_3 - \cos(\theta_1)\, X + \sin(\theta_1)\cos(\theta_2) d_1 $$

$$ \mathrm{H}[3,4] = -\sin(\theta_2)(Z - d_4) - \cos(\theta_2)\, Y - \sin(\theta_2) d_3 + \sin(\theta_1) d_1 $$

$H_{1\text{-}5}$ is the transformation matrix from the C-axis to the Z-axis. In order to use part coordinates we add a frame before the C-axis. In reality we can put this frame on the coordinates of the part reference frame. This way the coordinates in the software are coincident with the coordinates in reality. The new frame will be F(Xp,Yp,Zp).

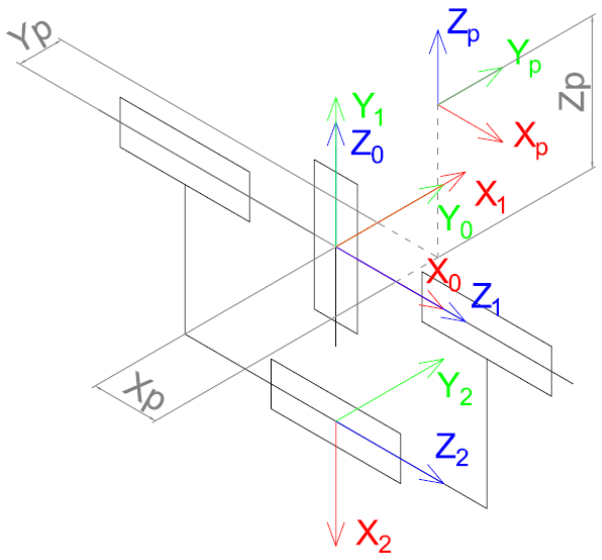


Figure 7-3: Added F(Xp, Yp, Zp) frame

The translation matrix corresponding with F(Xp,Yp,Zp):

$$H_p = \begin{bmatrix} 1 & 0 & 0 & -X_p \\ 0 & 1 & 0 & -Y_p \\ 0 & 0 & 1 & -Z_p \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Where Xp,Yp and Zp are the coordinates of the origin of the frame relative to F(X0,Y0,Z0).

The new transformation matrix is then:

$$H_{p-5} = \begin{bmatrix} \sin(\theta_1) & -\cos(\theta_1)\sin(\theta_2) & -\cos(\theta_1)\cos(\theta_2) & H[1,4] \\ -\cos(\theta_1) & -\sin(\theta_1)\sin(\theta_2) & -\sin(\theta_1)\,cos(\theta_2) & H[2,4] \\ 0 & \cos(\theta_2) & -\sin(\theta_2) & H[3,4] \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\mathrm{H}[1,4] = -\cos(\theta_1)\cos(\theta_2)(Z - d_4) + \cos(\theta_1)\sin(\theta_2)Y - \cos(\theta_1)\cos(\theta_2)\,d_3 + \sin(\theta_1)\,X + \cos(\theta_1)\cos(\theta_2)d_1 - X_p$$

$$\mathrm{H}[2,4] = -\sin(\theta_1)\cos(\theta_2)(Z - d_4) + \sin(\theta_1)\sin(\theta_2)Y - \sin(\theta_1)\cos(\theta_2)\,d_3 - \cos(\theta_1)\,X + \sin(\theta_1)\cos(\theta_2)d_1 - Y_p$$

$$\mathrm{H}[3,4] = -\sin(\theta_2)(Z - d_4) - \cos(\theta_2)\,Y - \sin(\theta_2)d_3 + \sin(\theta_1)d_1 - Z_p$$

To transform a point from frame to frame we use following formula:

$$\begin{bmatrix} X_{n-1} \\ Y_{n-1} \\ Z_{n-1} \\ 1 \end{bmatrix} = H * \begin{bmatrix} X_n \\ Y_n \\ Z_n \\ 1 \end{bmatrix}$$

Now it is possible to calculate the position of a point in frame F(Xp,Yp,Zp) relative to frame F(X5,Y5,Z5).

$$\begin{bmatrix} X_p \\ Y_p \\ Z_p \\ 1 \end{bmatrix} = H_{p-5} * \begin{bmatrix} X_5 \\ Y_5 \\ Z_5 \\ 1 \end{bmatrix}$$

7.2 DFPOST

DFPOST is the first stage in creating a post-processor. In this stage the machine will be defined. This can be done by using the DFPOST function in Cimatron E11. In order to define the machine a list of questions needs to be answered. There are 12 sections in total, each section contains questions concerning the machine. The questions will be answered as explained in chapter 3.2.1.

12 Sections of DFPOST:

1. TAPE INFORMATION
2. PROGRAMMING MODE AND UNIT
3. FORMATS
4. POSITIONING CODES
5. MESSAGES AND INSERTS
6. MACHINE PARAMETERS
7. LINEAR MOTION
8. CIRCULAR MOTION
9. NURBS MOTIONS
10. CANNED CYCLES
11. OUTPUT FILES
12. POSTPR/EXPST INTERFACE

If we take a closer look at section 6, machine parameters, we can see that it contains 11 questions. The question assign G/M-codes to certain machining actions. A full list of the answers can be found in appendix 3.

Machine parameters:

1. Code for clockwise spin <M3>
2. Code for counterclockwise spin <M4>
3. Code for spin stop <M5>
4. Code for flood on <M8>
5. Code for mist on <M7>
6. Code for air on <>
7. Code for through on <>
8. Code for coolant off <M9>
9. Cutter compensation off <G40>
10. Cutter compensation left <G41>
11. Cutter compensation right <G42>

(Cimatron)

7.3 Program file

The program file is the most import file of the GPP. This is where the code is written and the movement of the machine is determined. The program file of the Mikron UCP 600 will be based upon a file from a 3 axes milling machine. The file will be build according to the structure defined in chapter 3.2.2.

7.3.1 Implementing the rotation matrix

In order to use the 5 axes, we need to implement the DH-method in the program file. GPP uses a ROTMAC for this. ROTMAC uses ROTMAT, which is a rotation matrix, in order to rotate the coordinates around the correct UCS. (Cimatron)

Setting up the ROTMAT and TRANSMAT will be done as followed:

ROT_MAT1=1; ROT_MAT2=0; ROT_MAT3=0;
ROT_MAT4=0; ROT_MAT5=1; ROT_MAT6=0;
ROT_MAT7=0; ROT_MAT8=0; ROT_MAT9=1;
TRANS_MATX=0; TRANS_MATY=0; TRANS_MATZ=0; (Cimatron)

ROT_MAT1 to ROT_MAT9 will be the values calculated with the Denavit-Hartenberg method:

$$H_{p-5} = \begin{bmatrix} \sin(\theta_1) & -\cos(\theta_1)\sin(\theta_2) & -\cos(\theta_1)\cos(\theta_2) & H[1,4] \\ -\cos(\theta_1) & -\sin(\theta_1)\sin(\theta_2) & -\sin(\theta_1)cos(\theta_2) & H[2,4] \\ 0 & \cos(\theta_2) & -\sin(\theta_2) & H[3,4] \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$ROTMAT = \begin{bmatrix} \sin(\theta_1) & -\cos(\theta_1)\sin(\theta_2) & -\cos(\theta_1)\cos(\theta_2) \\ -\cos(\theta_1) & -\sin(\theta_1)\sin(\theta_2) & -\sin(\theta_1)cos(\theta_2) \\ 0 & \cos(\theta_2) & -\sin(\theta_2) \end{bmatrix}$$

Substitute $\theta_1 = \frac{\pi}{2} - C$, $\theta_2 = -\frac{\pi}{2} - A$ Where C and A are the angles of the respective axes.

$$ROTMAT = \begin{bmatrix} \cos(C) & \sin(\mathrm{C}) * \cos(\mathrm{A}) & \sin(\mathrm{C}) * \sin(\mathrm{A}) \\ -\sin(C) & \cos(\mathrm{C}) * \cos(\mathrm{A}) & \cos(\mathrm{C}) * \sin(\mathrm{A}) \\ 0 & -\sin(\mathrm{A}) & \cos(\mathrm{A}) \end{bmatrix}$$

ROTMAT automatically transforms all variables with FORMAT (COORDINATES) according to these values. This matrix transforms coordinates from the A-axis to the C-axis. In order to transform from the C-axis to the A-axis we need to take the inverse matrix.

$$ROTMAT = \begin{bmatrix} \cos(C) & \sin(\mathrm{C}) * \cos(\mathrm{A}) & \sin(\mathrm{C}) * \sin(\mathrm{A}) \\ -\sin(C) & \cos(\mathrm{C}) * \cos(\mathrm{A}) & \cos(\mathrm{C}) * \sin(\mathrm{A}) \\ 0 & -\sin(\mathrm{A}) & \cos(\mathrm{A}) \end{bmatrix}^{-1}$$

This matrix is orthogonal so the inverse matrix is also the transposed matrix.

$$ROTMAT = \begin{bmatrix} \cos(C) & -\sin \mathrm{C} & 0 \\ \sin(\mathrm{C}) * \cos(\mathrm{A}) & \cos(\mathrm{C}) * \cos(\mathrm{A}) & -\sin(\mathrm{A}) \\ \sin(\mathrm{C}) * \sin(\mathrm{A}) & \cos(\mathrm{C}) * \sin(\mathrm{A}) & \cos(\mathrm{A}) \end{bmatrix}$$

In code this becomes:

** Calculate the ROTMAT values*

ROT_MAT1 = COSC ; ROT_MAT2 = -SINC ; ROT_MAT3 = 0 ;
ROT_MAT4 =SINC COSA ; ROT_MAT5 = COSC*COSA ; ROT_MAT6 = -SINA ;*
*ROT_MAT7 = SINC*SINA ; ROT_MAT8 = COSC*SINA ; ROT_MAT9 = COSA ;*

7.3.2 Translation matrix

In the previous section the rotation transformation was calculated. This transformation is done around the UCS of the model. In reality the part is rotated around the MRZP and not the model UCS. This would only be correct when the MRZP and the model UCS were coincident. This is however not possible since the rotating table has a certain thickness. Because of this distance, the position after rotating around MRZP will differ from the position after rotating the model UCS. A translation matrix is used to shift the toolpath along this distance. The TRANSMAT values will be calculated in a few steps.

Step1

In the first step the distance between the rotation centre and the ORIGIN is calculated.

TOTALDX = X_ORIGIN - X_MACH - TABLEDX;
TOTALDY = Y_ORIGIN - Y_MACH - TABLEDY;
TOTALDZ = Z_ORIGIN - Z_MACH - TABLEDZ;

TABLEDX, TABLEDY and *TABLEDZ* are the distances between the rotation centre and the ORIGIN. These are parameters that are machine dependent and need to be assigned when post-processing.

Step 2

In the second step the distance between the rotation centre and origin after rotation is calculated. This distance is calculated using ROTMAC:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix}_{after\ rotation} = ROTMAT * \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}_{before\ rotation}$$

*TOTALAFTERROTX = TOTALDX*ROT_MAT1 + TOTALDY*ROT_MAT2;*
*TOTALAFTERROTY = TOTALDX*ROT_MAT4 + TOTALDY*ROT_MAT5 + TOTALDZ*ROT_MAT6;*
*TOTALAFTERROTZ = TOTALDX*ROT_MAT7 + TOTALDY*ROT_MAT8 + TOTALDZ*ROT_MAT9;*

Step 3

In the third step the distance between the origin before and after rotation is calculated.

DELTAORIGINX = TOTALAFTERROTX - TOTALDX;
DELTAORIGINY = TOTALAFTERROTY - TOTALDY;
DELTAORIGINZ = TOTALAFTERROTZ - TOTALDZ;

Step 4

In this step the distance between the current origin and the first origin (model UCS) is calculated. This is only the case for multiple origin output or in other words, when there are multiple 5 axes procedures.

DELTAFIRSTX = DELTAORIGINX + (X_ORIGIN - FIRSTORIGINX);
DELTAFIRSTY = DELTAORIGINY + (Y_ORIGIN - FIRSTORIGINY);
DELTAFIRSTZ = DELTAORIGINZ + (Z_ORIGIN - FIRSTORIGINZ);

Step 5
In the last step the TRANS_MAT values are applied. Notice that these values are negative of previous calculated values. This is due to the fact that in previous step the distance from the current origin to the original is calculated, but the toolpath needs to be transformed from the original to the current origin, hence the negation.

TRANS_MATX = - DELTAFIRSTX;
TRANS_MATY = - DELTAFIRSTY;
TRANS_MATZ = - DELTAFIRSTZ;

7.3.3 Calculating the angles

The rotation matrix can only work if the angles of the axes are given. These angles can be calculated using the unit vector of the current UCS. The I,J and K components of the unit vector are in the direction of the Z axis of the current UCS.

In figure 7-4 a random vector R is shown. Angle C can be calculated directly:

$$C = tan^{-1}\left(\frac{I}{J}\right)$$

Angle A is the angle between S, the resulting vector of the rotation around the C-axis, and K:

$$S = \frac{J}{\cos(C)}$$

$$A = tan^{-1}\left(\frac{S}{K}\right) = tan^{-1}\left(\frac{J}{K * \cos(C)}\right)$$

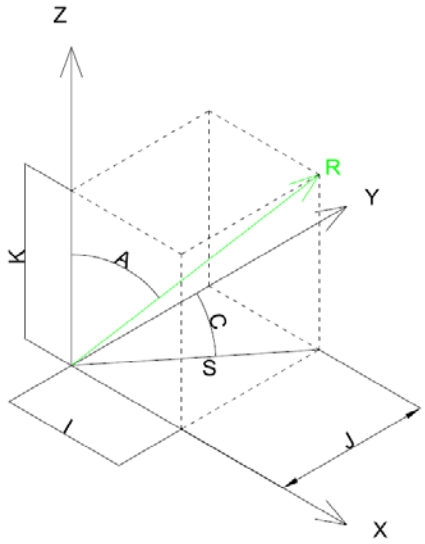


Figure 7-4: Calculating the angles of the unit vector

There is only one exception. When $J = 0$ or C= $\pm 90°$. In this case the solution of S results in $\frac{0}{0}$, which is undefined. To solve this problem we make a separate *if-loop* to calculate the angle in this case. In this loop $S = I$.

Now that it is possible to calculate the angles, we need to take a look at the machine conventions. For the Mikron UCP 600 both axes are positive in the counter clockwise direction (figure 7-5).

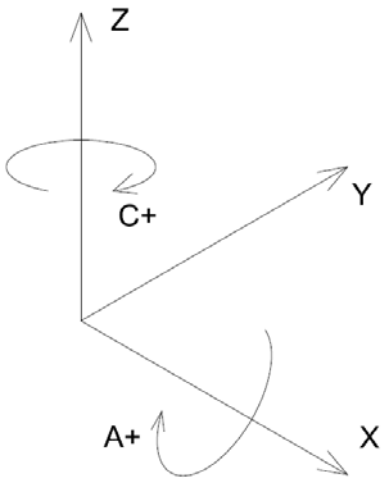


Figure 7-5: Mikron rotation axes direction

The rotation of the C-axis can be reduced to eight cases. For each case we want the post-processor to take the fastest way. For instance, in case one, the table can be rotated 90° or 270°. The only difference is that the A-rotation will be opposite. This can be said for any of the eight cases. For case one to four there is no problem with the size of the angle, since the arctangent of the angle components is always 90° degrees or less. The only problem here is the direction of the rotation. In case five and seven the problem is that the arctangent of infinite can be both positive and negative. GPP always takes the positive angle. For case six and eight there is no problem since the angle is 0.

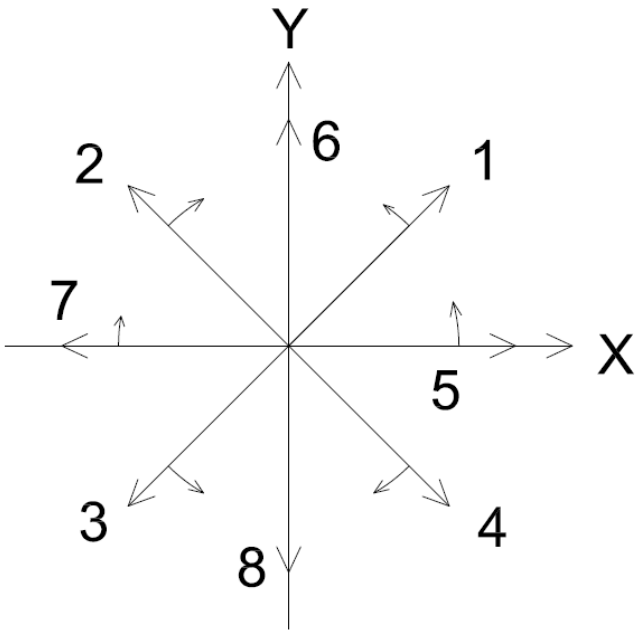


Figure 7-6: Anlge direction

Table 7-2 shows the calculated angle and the angle we wish to have for all case. Figure 7-6 shows the shortest rotation of every case. The arrows indicate the direction of the rotation. The direction of the errors are the same as used in the CAM machine simulation. To determine these angles I made a simulation for every possible case. The directions in figure 7-6 are the resulting direction from these simulations.

For example, in case one the calculated angle is 45 and the rotation is clockwise. Since the positive direction of the machine is counter clockwise, the direction will be negative.

Case	Calculated angle (°)	Machine angle (°)
1	45	-45
2	-45	45
3	45	-45
4	-45	45
5	+90	-90
6	0	0
7	+90	90
8	0	0

Table 7-2: Comparison between calculated and machine C angle

From this table we can establish that the C-rotation of the machine has to be opposite to the calculated values. Notice that calculated angle is still used to calculate the ROTMAC values. The machining angle is only used to instruct the machine to rotate in the right direction. This is because the machine direction is opposite to the sign convention.

The same can be done for the A-axis. The A-axis operates between +90° and -90°. With the A-axis there are only two angles that are a problem, A=+90° and A=-90°. These angles occur only when K=0. Like I stated before, GPP will always take the positive angle when the tangent of infinity is taken. This would result in A=+90° for all values.

Case	**I**	**J**	**K**	**A by GPP**	**Desired ROTMAC angle**	**Machine angle**
5	1	0	0	+90	90	-90
7	-1	0	0	+90	90	-90
1,2,6	X	> 0	0	+90	+90	-90
3,4,8	X	< 0	0	+90	-90	+90

Table 7-3: Comparison between calculated an desired A angle

This can be solved by using *if-loops*. For case 5 and 7 it can be said that $A = 90$. For case 1, 2 and 6 it can als be said that $A = 90$. For case 3,4 and 8, $A = -90$.

This will result in following code:

```
* Calculate the rotation angle of the C axis, around Z.
        I = MI_ORIGIN; J = MJ_ORIGIN; K = MK_ORIGIN;
        C = ATAN(I/J);
        CROT = -C;
        COSC = COS(C); SINC = SIN(C);
* Calculate the rotation angle of the A axis, around X.
        IF_SET (J _NE_ 0)
        S = J/COSC;
        ELSE
        S = I;
        END_IF;
                IF_SET (K _EQ_ 0)
                IF_SET (J _EQ_ 0)
                        A = 90;
                END_IF;
                IF_SET (J _LT_ 0)
                        A=-90;
                END_IF;
                IF_SET(J _GT_ 0)
                        A=90;
                END_IF;
        ELSE
                A = ATAN(S/K);
        END_IF;
        AROT = -A;
        COSA = COS(A); SINA = SIN(A);
```

7.3.4 ORIGIN CHANGE

It is recommended to use the ROTMAC in the ORIGIN CHANGE block. This block will be activated when the UCS changes in position or angle. First we need determine when there is a 5 axes motion at hand. To do this we can check the state of the unit vector. When 3 axes motions are used, the unit vector will always be of the form $\begin{bmatrix} I \\ J \\ K \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$, this represents the orientation where the Z-axis of the UCS is vertical. When 5 axes motion is present the k-component will never be equal to 1. We can compare this every time there is an origin change. This will result in a change of state of a system flag.

The code for this will be as following:

```
 Flagrotmac = no ;
IF_SET ( MK_ORIGIN _NE_ 1 )
        FLAGROTMAC = YES ;
END_IF ;
```

If the flag is set to YES the ROTMAC values will be calculated. The full code for the ORIGIN CHANGE block can be found in appendix 4.

Note1: If it is wanted to start with 5 axes motion (start with a K-value not equal to 1), it is necessary do adjust the parameter in section 2.7 of DFPost to <MACSYS>. Else the machine will think that the coordinate orientation is the rotated UCS of the first procedure. This would result in a double rotation of the UCS.

Note 2: If we want that the axes are moved on every action (e.g. drilling, milling,…) we need to add the following code to each block (e.g. LINEAR MOTION, CIRCULAR MOTION,…):

```
IF_SET (FLAGROTMAC)
        OUTPUT $;
        OUTPUT " G00";
        OUTPUT " C" CROT;
        OUTPUT " A" AROT;
END_IF;
```

7.3.5 Verifying ROTMAC

The code in previous section can be subjected to errors. To be fully aware of the effect of the code we need to compile it and verify the result. To do this a part is created in Cimatron E11 as shown in figure 7-7. This part contains a surface that needs to be milled under an angle of 45°.

First I will calculate the rotation and translation of the toolpath in theory. Next I will compare these results with G-code simulation done in CIMCO edit v7.0.

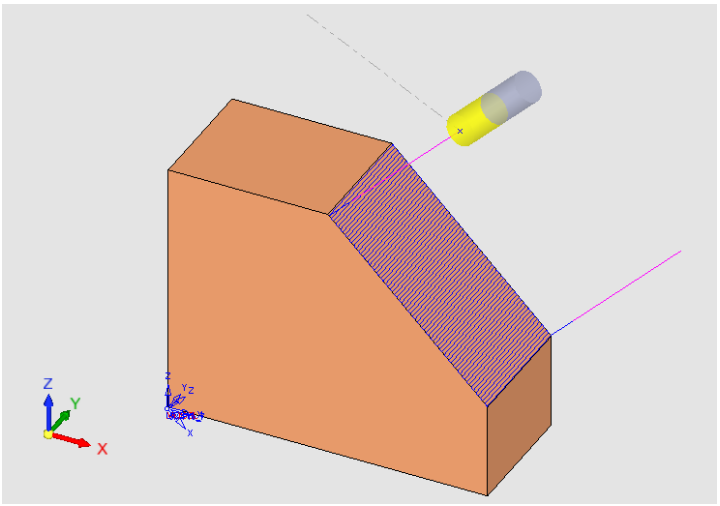


Figure 7-7: Test part in Cimatron E11 with toolpath under 45°

To be able to mill this surface, the part needs to be rotated 90° around the C-axis and 45° around the A-axis. Figure 7-8 shows the part in the original position. Figure 7-9 shows the part after rotation of the C-axis. Figure 7-10 shows the part after rotation of the A-axis. The part in broken line is rotated around its own UCS, without the translation matrix. The part in red is rotated around the MRZP, using both the rotation and translation transformations.

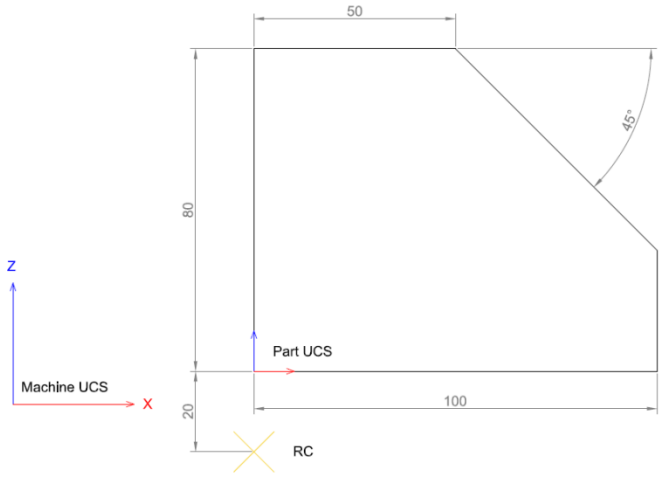


Figure 7-8: original position

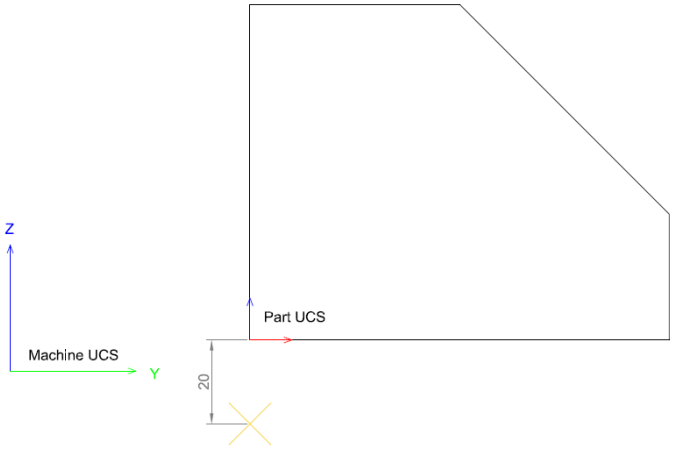


Figure 7-9: Position after rotation of the A-axis

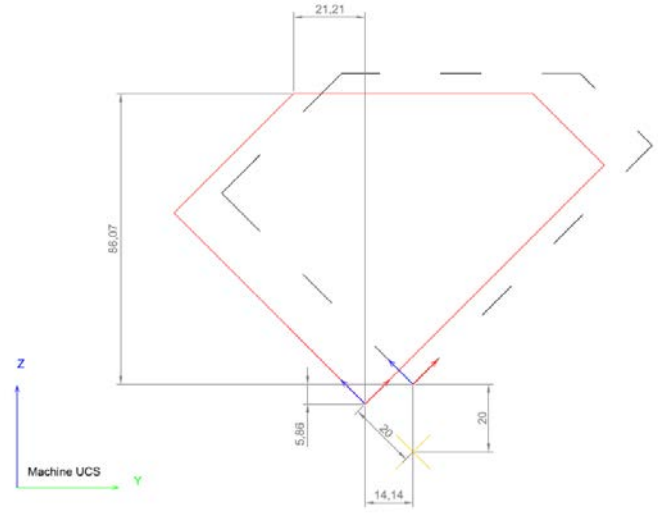


Figure 7-10: Position after rotation of both axes

If we simulate the code in CIMCO we get the resulting toolpath as shown in figure 7-11. This is an basic way of verifying G-code and requires the right interpretation. In order to get useful information of this simulation we need to compare the visual image with the generated G-code. For instance. Take a look at the G-code at the start of the program:

```
%
O0100
T01
G90 G80 G00 G17 G40 M23
G43 H01 Z130. S1000 M03
G00 X0.0 Y-34.855 Z144.142 M09
Z106.066
Z96.066
G01 Z86.066 F350
X-40.
Y-33.855
X0.0
Y-32.855
X-40.
Y-31.855
X0.0
Y-30.855
X-40.
```

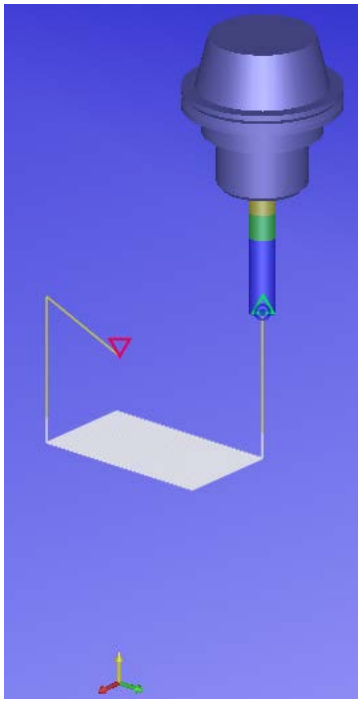


Figure 7-11: Simulation in CIMCO Edit v7.0

To evaluate the rotation transformation we just have to look at the position of the toolpath. In figure 7-11, it is seen that the start-end direction is in the direction if the Y-axis. This means that the toolpath is rotated 90° around the Z-axis which corresponds with the C-axis rotation. We can also see that the toolpath is horizontal, meaning that there is a 45° rotation about the X-axis which corresponds with the A-axis rotation. This can also be seen in the G-code as the Z values never changes.

The highlighted value, Z86.066, is the Z position where the linear movement starts, or the beginning of the milling. Notice that this value is the same as calculated in figure 7-10. If the Y value, $Y = 21{,}21 + 14{,}14 = 35{,}35$, is compared to the G-code value, Y-34.855, we notice a small difference. This is because the cutter does not necessarily needs to start cutting at 35.36 mm because of the width of the cutter. From this we can safely assume that the rotation and translation transformations are correct.

.

7.3.6 Linear and circular motion in 5 axes

Until now it is only possible to use the rotation axes of the CNC machine when the origin changes. This limits the machine freedom to in such a way it can only be used to cut planes. In order to manufacture pars with complex surfaces it is need for the axes to rotate after every cut or after every movement. To ensure this movement the ROTMAC will be integrated in the LINEAR MOTION and the CIRCULAR MOTION block.

We use the same parts from the ORIGIN CHANGE block and adjust it to LINEAR MOTION paramters. We replace I_ORIGIN by I_COORD.

This results in following code:

```
LINEAR MOTION:
* check for change in orientation ( 5 axis positioining )
   Flagrotmac = no ;
  IF_SET ( K_COORD _NE_ 1) FLAGROTMAC = YES ; END_IF ;

* ROTMAC - change the tool orientation
        IF_SET(FLAGROTMAC)

* Calculate the rotation angle of the C axis, around Z.
        I = I_COORD; J =J_COORD; K = K_COORD;
        C = ATAN(I/J);
        CROT = -C;
        COSC = COS(C); SINC = SIN(C);

* Calculate the rotation angle of the A axis, around X.
        IF_SET (J _NE_ 0)
        S = J/COSC;
        ELSE
        S = I;
        END_IF;
        IF_SET (K _EQ_ 0)
                IF_SET (J _EQ_ 0)
                        A = 90;
                END_IF;
                IF_SET (J _LT_ 0)
                        A=-90;
                END_IF;
                IF_SET(J _GT_ 0)
                        A=90;
                END_IF;
        ELSE
                A = ATAN(S/K);
        END_IF;
        AROT = -A;
        COSA = COS(A); SINA = SIN(A);
        OUTPUT $;
        OUTPUT " G00";
        OUTPUT " C" CROT;
        OUTPUT " A" AROT;
```

** Calculate the ROTMAT values*

ROT_MAT1 = COSC ; ROT_MAT2 = -SINC ; ROT_MAT3 = 0 ;
ROT_MAT4 =SINC COSA ; ROT_MAT5 = COSC*COSA ; ROT_MAT6 = -SINA ;*
*ROT_MAT7 = SINC*SINA ; ROT_MAT8 = COSC*SINA ; ROT_MAT9 = COSA ;*

END_IF;

Figure 7-12 and 7-13 show the result of this code. The toolpath is not rotated, and exactly the same as the 3 axes toolpath. The reason for this is that the ROTMAC is can only be used in the block ORIGIN CHANGE. This heavily limits the possibilities of the GPP.

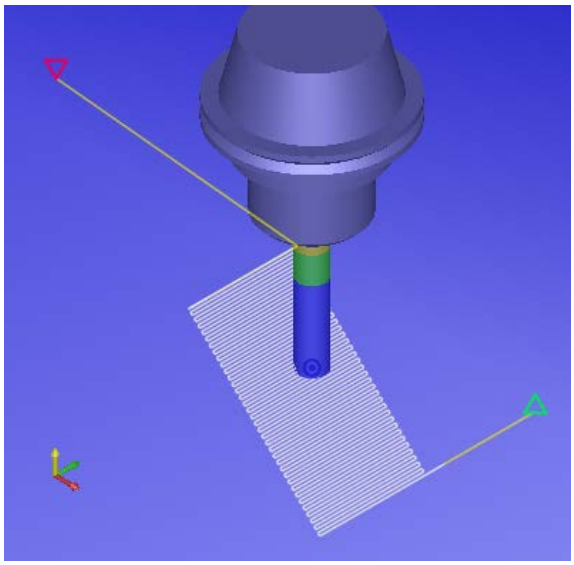


Figure 7-12 Simulation of lin. motion with rotation (a)

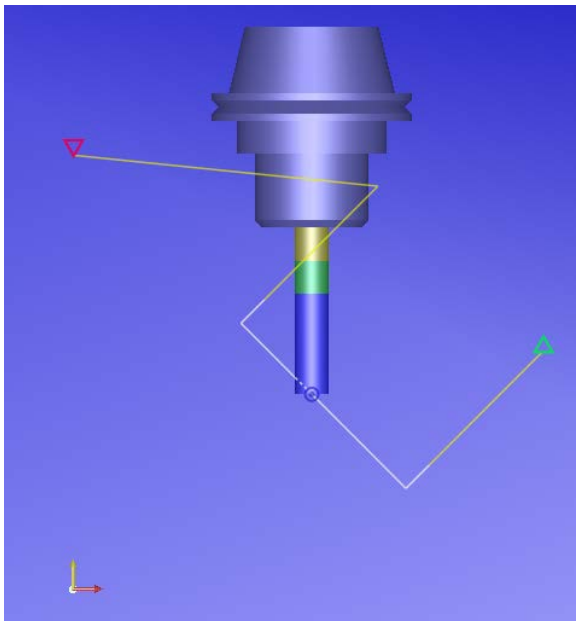


Figure 7-13: Simulation of lin. Motion with rotation (b)

GPP is an older type of processor used in Cimatron E11. It was only used for 3 axes milling and sometimes for 4-axis milling. In 5 axes milling there are two types of milling: contour and positioning milling. With GPP we can only achieve positioning milling. If contouring is required we need to use a more advanced post-processor.

Cimatron E11, however, contains a new type of post-processor, GPP2. This processor is used in modern day 5 axes machines. This post is able to use contour milling and thus making complex surface milling possible. This post-processor will not be further researched/developed. Originally it was planned to test this post-processor on a sphere. To make this sphere it is necessary to use contouring. This is not possible with GPP, so Instead the GPP will be tested for position milling on the Mikron UCP 600.

7.4 Testing

7.4.1 Test part

In order to test my post-procesor on the Mikron UCP 600 I have made the part as shown in figure 7-14. There are four general procedures used. The first procedure is 3 axes volume milling, the second procedure is 3 axes drilling, the third procedure is 5 axes milling and the last procedure is 5 axes drilling.

Figure 7-15 shows the procedure listing of this part. Here it can be seen that all 5 axes procedures are divided into four parts. This has to be done in order for the post-processor to recognise every milling surface. In chapter 7.3.4 it is said that the post-processor can only calculate ROTMAC values when there is an origin change. To get these origin changes, every surface has to be a different procedure with its own UCS. This is mandatory for every five axes procedure.

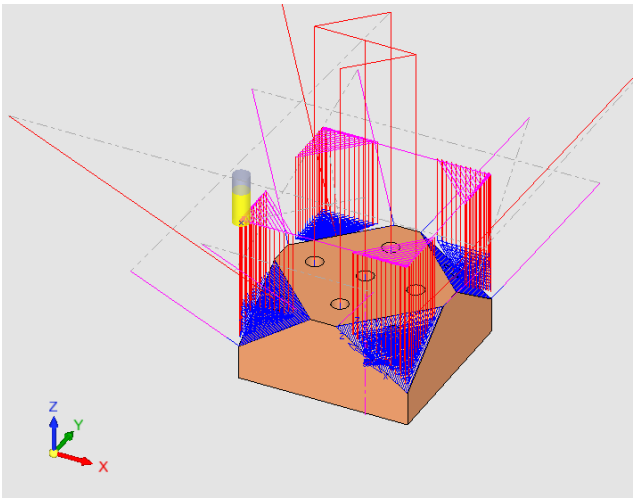


Figure 7-14: Test part for post-processing

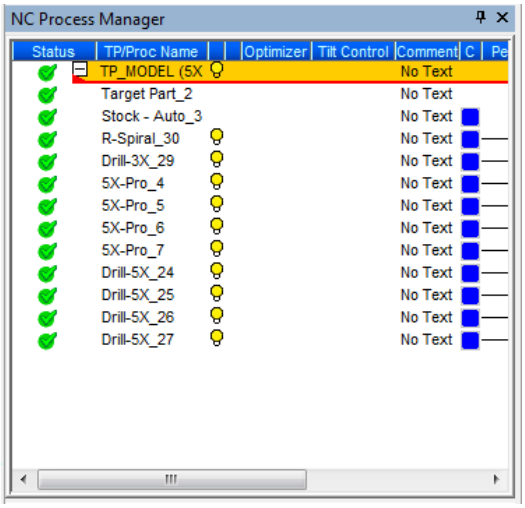


Figure 7-15: NC procedures in Cimatron

7.4.2 Mikron UCP 600

This post-processor is designed specifically for a Mirkon UCP 600. At the CNC department at the HUST, there is such a machine available. However, do to technical errors this machine was not operating. Some parameters were changed/deleted so at this moment the machine could only operate as a 3 axes machine. This is a major drawback in this thesis. Its is necessery to determine if the post-processor is in some way useable on the machine. Therefore I will do a G-code analysis to check for flaws in the output file.

7.4.3 G-code analysis

For the G-code analysis a combination of CAM machine simulation and G-code tool simulation will be used. The G-code analysis will not be done on the test part for the machine. Instead a simpler part will be used to check for similarities between the G-code and the CAM machine simulation. The part can be seen in figure 7-16. This is a part with two holes on each side. The holes on side 2 are 10mm lower than the holes on side 3, and so on.

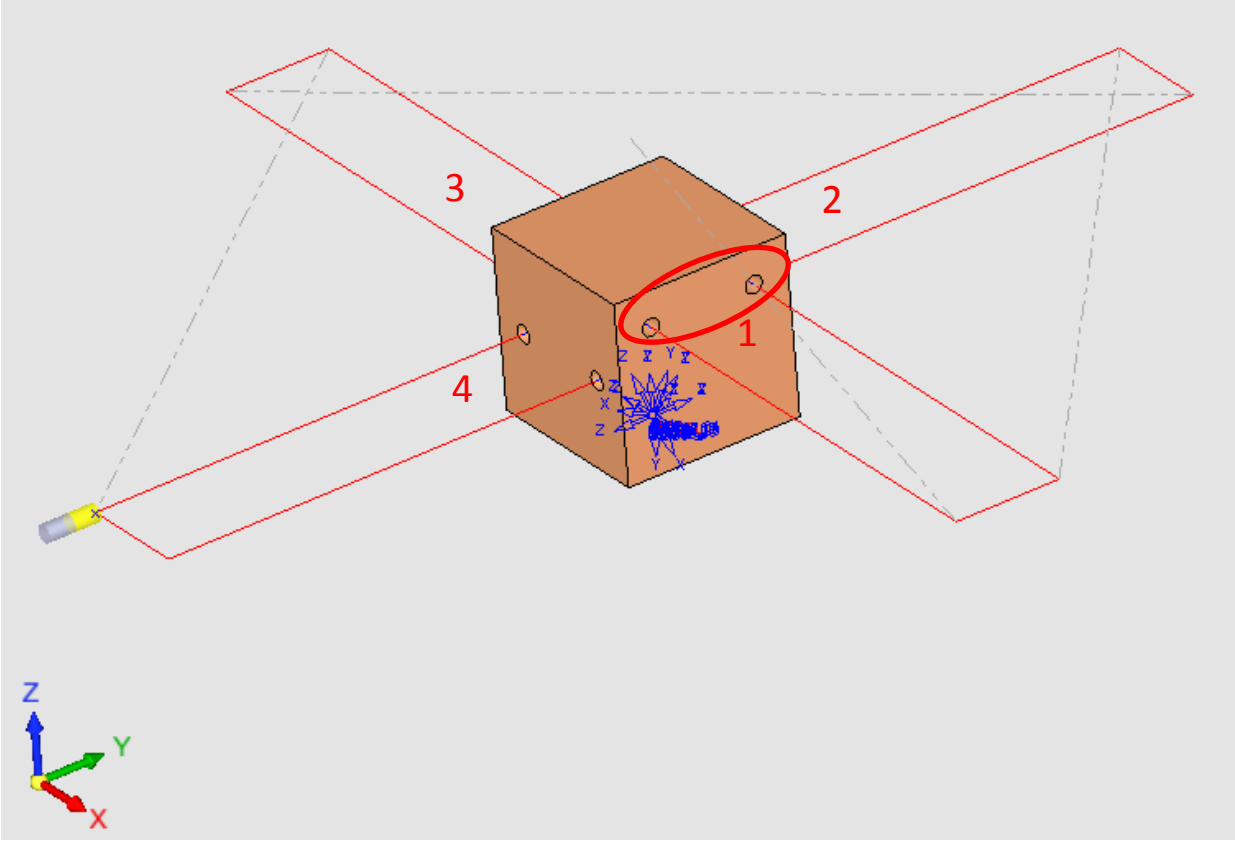


Figure 7-16: Test part 2 for problematic angles

First we need to define the MRZP in Cimatron. It does not matter what the MRZP is in the simulation, as long as the same offset is used in the post-processor. The MRZP will be at (0,0,-20) as indicted in figure 7-17. The table used in this figure might be misguiding, since it looks like the C- and A-axis don't intersect. However, this is the table of another Mikron. I have only used this part to get a better visual image of the CAM simulation. In figure 7-18 the picture of the real Mikron UCP 600 is shown. Here we can see that the MRZP is as indicated by the coordinate system in figure 7-17 resembles the real situation.

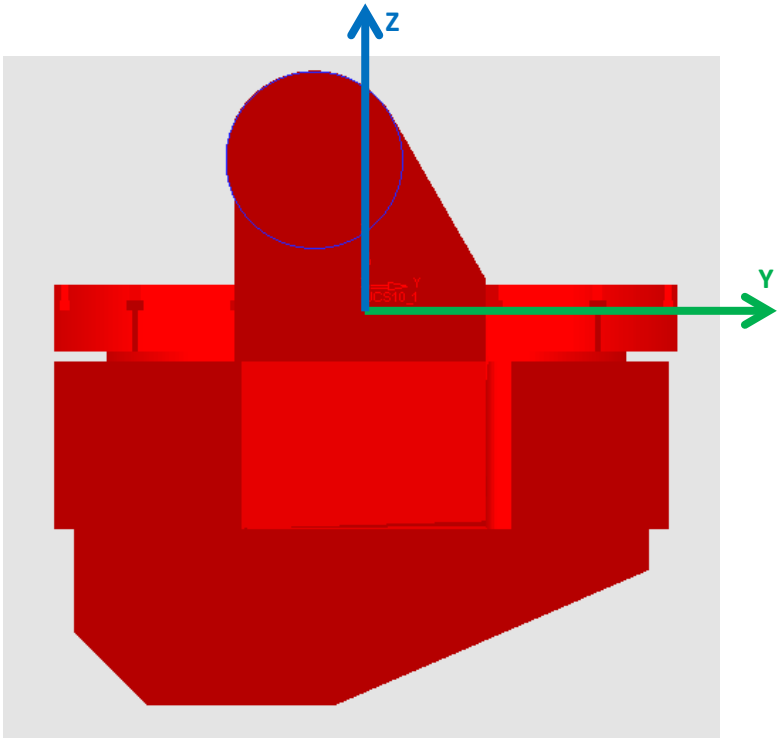


Figure 7-17: MRZP in CAM simulation

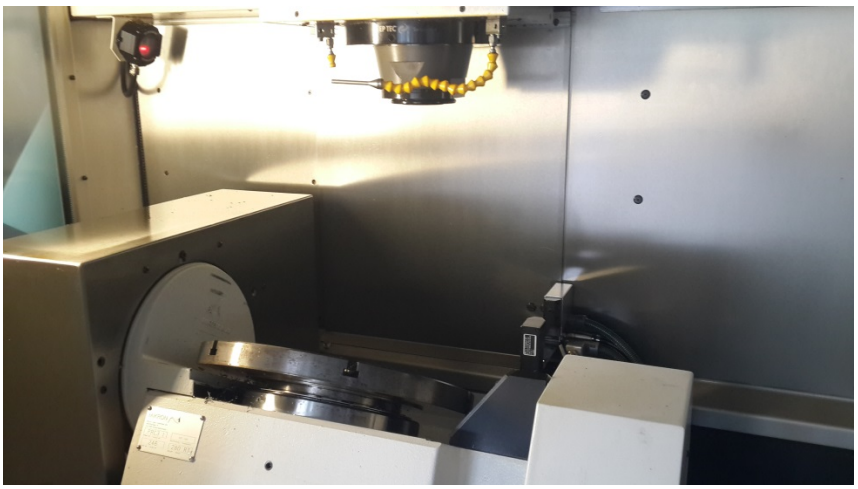


Figure 7-18: The rotary table of the Mikron UCP 600

Now we can post-process the file shown in figure 7-16 using following values:
$TABLEDX = 0;$
$TABLEDY = 0;$
$TABLEDZ = -20.$

After doing this we get following output:

```
%
O0100                                                    1
T02
G90 G80 G00 G17 G40
G43 H02 Z150. S1000 M03
G00 C-90.0 A-90.0
Z320.
G98 G81 X30. Y-60. Z40. R57. F350 M08
X-30.
G80 Z320.
G00 C0.0 A-90.0
Z320.                                                    2
G98 G81 X-30. Y-50. Z40. R71. F350 M08
X30.
G80 Z320.
G00 C+90.0 A-90.0
Z320.
G98 G81 X-30. Y-40. Z0. R71. F350 M08                    3
X30.
G80 Z320.
G00 C+0.00000000000002 A+90.0
Z320.                                                    4
G98 G81 X-30. Y40. Z40. 751. F350 M08
X30.
G80 Z320.
G00 C0.0 A0.0
M30
%
```

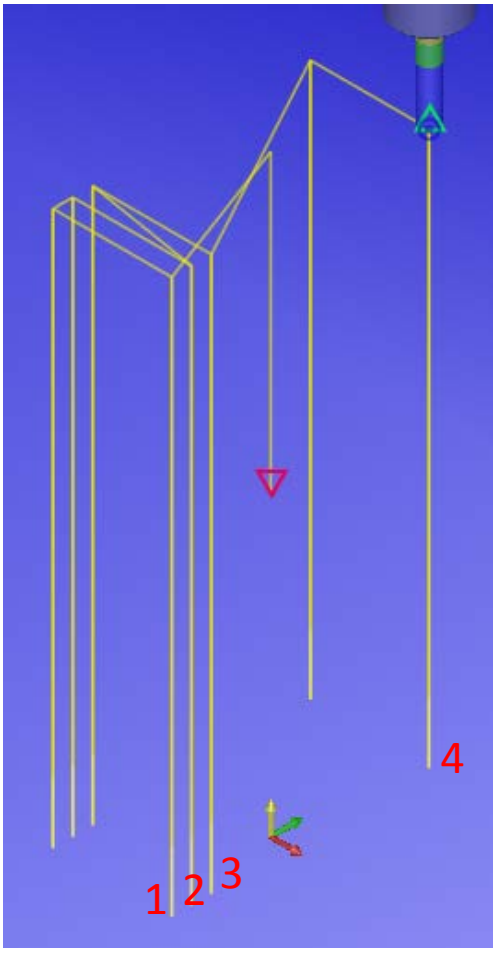


Figure 7-19: G-code toolpath simulation

We can compare this output with the simulation data for each of the four cases. For case 1 the G-code will also be explained.

Case 1

G-code	Explanation
T02	Tool listing.
G90 G80 G00 G17 G40 G43 H02 Z150. S1000 M03	This a standard command after tool change. G90=absolute programming, G80=canned cycle cancel, G00=rapid motion, G17=XY-plane select, G40=cutter compensation off, G43=tool offset, with offset for tool 02, M03 is a machine code for spindle on clockwise with speed 1000/min. Z150 is the clearance height.
G00 C-90.0 A-90.0	Rotates the axis to the appropriate position.
Z320. G98 G81 X30. Y-60. Z40. R57. F350 M08 X-30. G80 Z320.	This is a command to start a drilling cycle. Z320 is the cycle initiation height, G98=return to initial height, G81= simple drilling cycle. The following X and Y values are from the first drilling point, Z40 is the drilling depth, R57 is the retract height, F350=feedrate 350mm/min, M08=coolant on.

Table 7-4: G-code explanation

CAM simulation	G-code output
G00 C-90.0 A-90.0 Z320. G98 G81 X30. Y-60. Z40. R57. F350 M08 X-30. G80 Z320.	C = -90 A = -90 X = 30,-30 Y=-60 Z=350

Table 7-5: Comparison bewteen CAM simulation and G-code output for case 1

We can see that the rotation is correct, both are the same. The Y-values are also the same. Only the Z-value of the clearance height is a bit off.

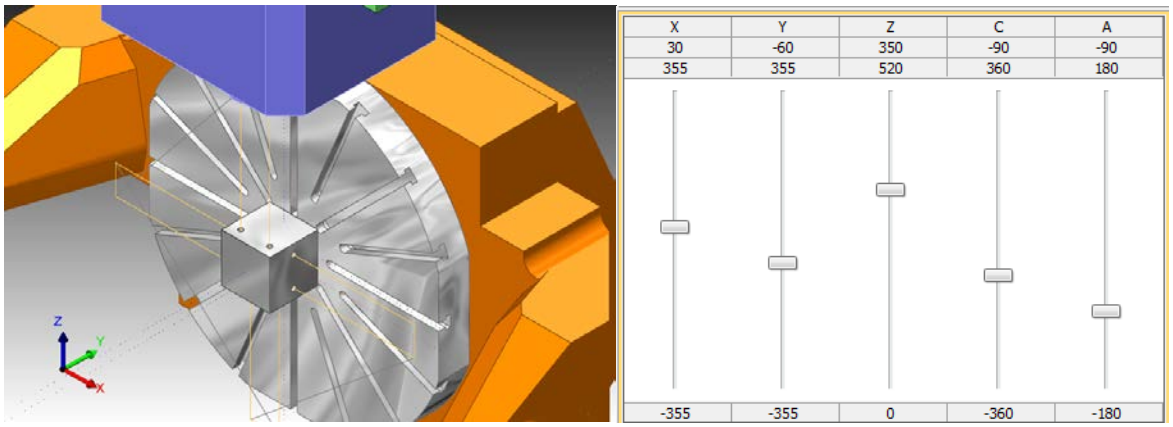


Figure 7-20: CAM simulation with axes values for case 1

Case 2

CAM simulation	G-code output
G00 C0.0 A-90.0 Z320. G98 G81 X-30. Y-50. Z40. R71. F350 M08 X30. G80 Z320.	C = 0 A = -90 X = 30,-30 Y=-50 Z=350

Table 7-6: Comparison bewteen CAM simulation and G-code output for case 2

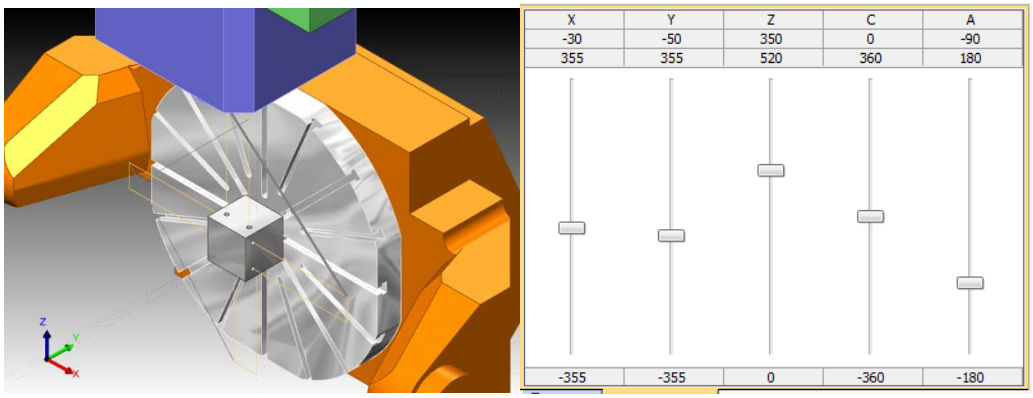


Figure 7-21: CAM simulation with axes values for case 2

Case 3

CAM simulation	G-code output
G00 C+90.0 A-90.0 Z320. G98 G81 X-30. Y-40. Z0. R71. F350 M08 X30. G80 Z320.	C = +90 A = -90 X = 30,-30 Y=-40 Z=350

Table 7-7: Comparison bewteen CAM simulation and G-code output for case 3

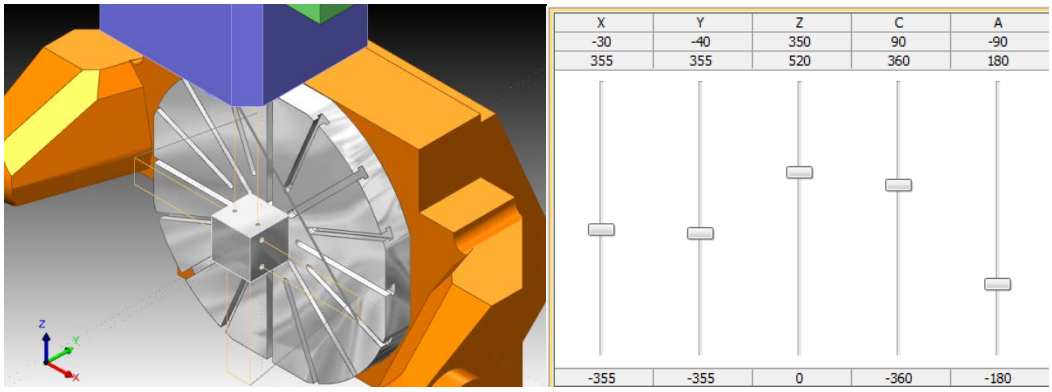


Figure 7-22: CAM simulation with axes values for case 3

Case 4

CAM simulation	G-code output
G00 C+0.00000000000002 A+90.0 Z320. G98 G81 X-30. Y40. Z40. 751. F350 M08 X30. G80 Z320.	C = 0 A = 90 X = 30,-30 Y=-50 Z=350

Table 7-8: Comparison bewteen CAM simulation and G-code output for case 4

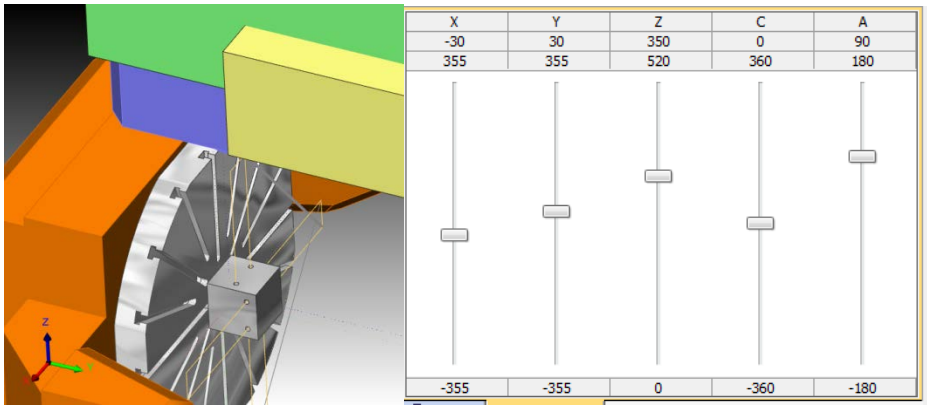


Figure 7-23: CAM simulation with axes values for case 4

We see that in every case the G-code has the same values as the simulation. This means that the toolpath is rotated and shifted correctly. Only the Z-value has a different clearance offset. This is because Cimatron has not incalulated the tool offset.

From this analysis we can conclude that the transformation matrices are correct. From the G-code explanation in case 1 we can see that the G-code is executed in appropriate order. First the tool is changed, then the tool goes the initial place, then the tables rotate and at last the drilling starts.

This is the best possible way to analyse the post-processor not using a real machine. From this analysis it is clear that the post-processor builds the G-code in the right order and the part is rotated correctly. However there might be some flaws in the code that can only be seen when testing on a real machine. These flaws cannot be detected by further software testing. The next step in the continuation of this thesis could be contour milling and machine testing.

8 Conclusion

The goal of this thesis was to make a post-processor for the Mikron UCP 600. I solved the problem of making a unique post-processor, which is made specifically for the Mikron UCP 600. I calculated the kinematic structure by using the Denavit-Hartenberg method. This was successfully implemented in the GPP.

However, at this point it also became clear that GPP can only be used for position milling and not for contour milling, the latter is used for complex 5 axes milling. I can therefor state that the problem of the complexity is not solved. If more time was at hand, this could be done by using GPP2. This is a newer version of post-processing.

The testing of the GPP did not go as expected. In order to test my post-processor I had to use it on the Mikron UCP 600. This was not possible because of technical errors. Instead I made a G-code analysis. This analysis showed that the results are acceptable and that the post-processor could work the Mikron UCP 600 apart from a few adjustments.

9 References

Angeles, J. (2003). *Fundamentals of Robotic MAchanical Systems: Theory, Methods, and Algorithms, Second Edition.* New York: Springer.

Arpo, K. (2008). *Secrets of 5-axis mchining.* 989 Avenue of the Americas, New York, NY 10018: Industrial Press, Inc.

Bijnens, J. (2011, April 17). Postprocessoren. Diepenbeek.

BKMech. (2014). Retrieved March 12, 201, from BKMech: http://www.bkmech.com.vn/gioi-thieu-ve-bkmech.html

Cimatron. (n.d.). GPP Cimatron E10.0 User Guide.

Denavit–Hartenberg parameters. (n.d.). Retrieved March 26, 2014, from Wikipedia: http://en.wikipedia.org/wiki/Denavit%E2%80%93Hartenberg_parameters

End, R., & jaje, J. (2008). *The Challenges for CAM Systems and Users.*

G-code. (n.d.). Retrieved March 10, 2014, from Wikipedia: http://en.wikipedia.org/wiki/G-code

HTMP. (2006). Retrieved March 12, 2014, from HTMP: http://www.htmp.com.vn/en/index.php

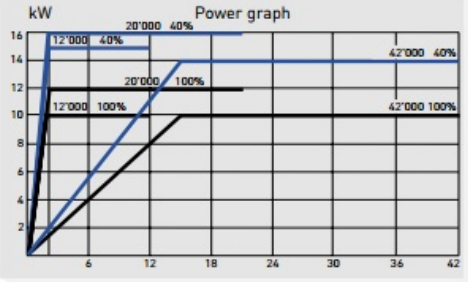
What is Post-Processing? (n.d.). Retrieved March 10, 2014, from ICAM: http://www.icam.com/html/products/whatis/what_is_post.php

10 Appendix

10.1 Appendix 1: Mikron UCP 600 machine specifications

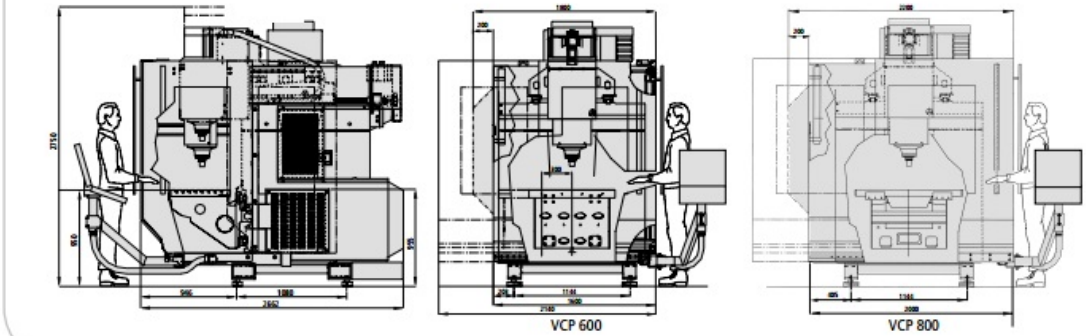
VCP 600/800, UCP 600

		VCP 600	VCP 800	UCP 600
Work area				
Longitudinal	X mm	600	800	530
Lateral	Y mm	450	450	450
Vertical	Z mm	450	450	450
High-performance spindles				
Spindle performance at 40% ED / S6	kW	15 / 16	15 / 16	15 / 16
Maximum rpm	min^{-1}	12'000 / 20'000	12'000 / 20'000	12'000 / 20'000
Spindle water-cooling		•	•	•
HSC spindle				
Spindle performance at 44% ED / S6	kW	14	14	14
Maximum rpm	min^{-1}	42'000	42'000	42'000
Spindle water-cooling		•	•	•

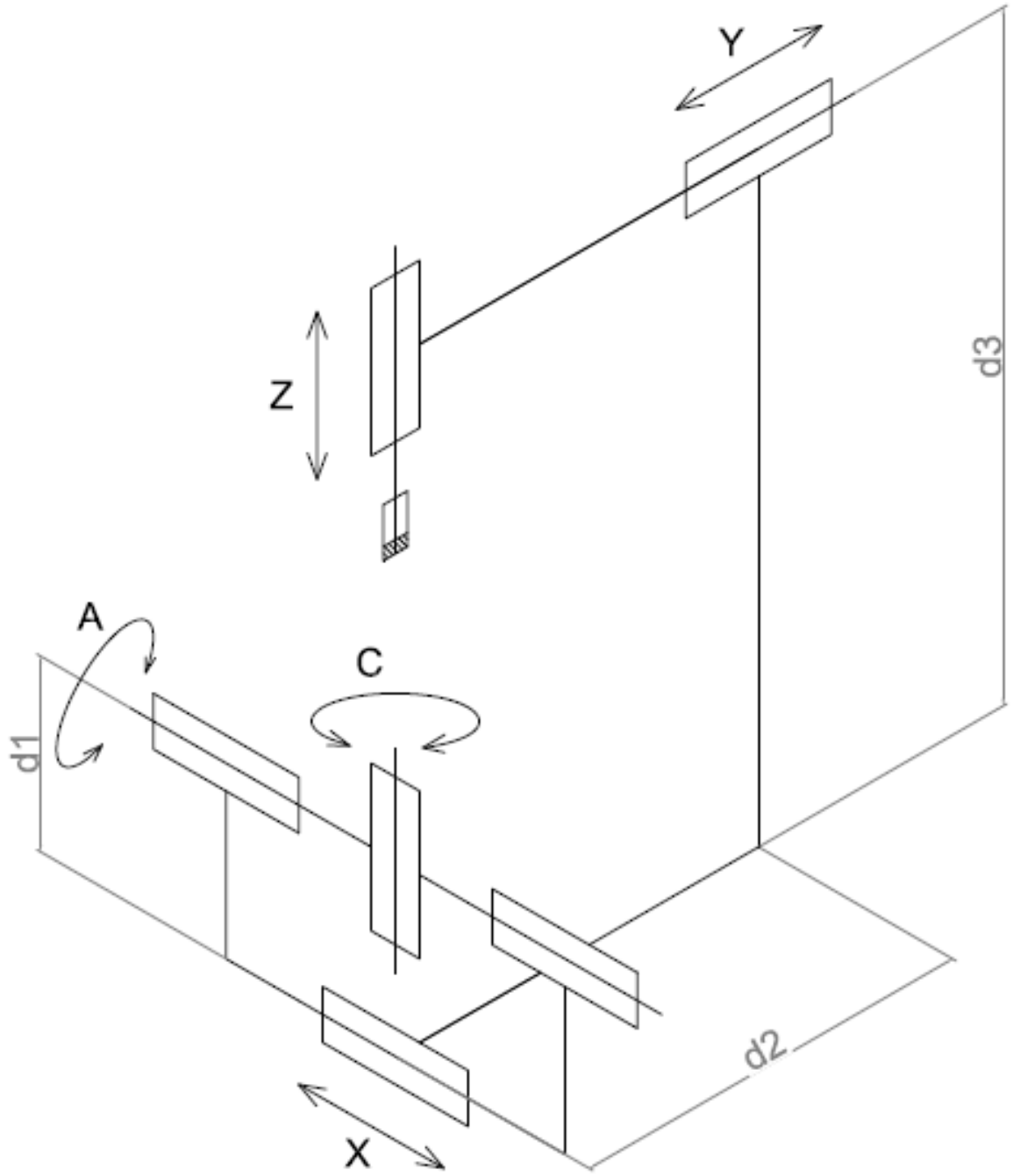


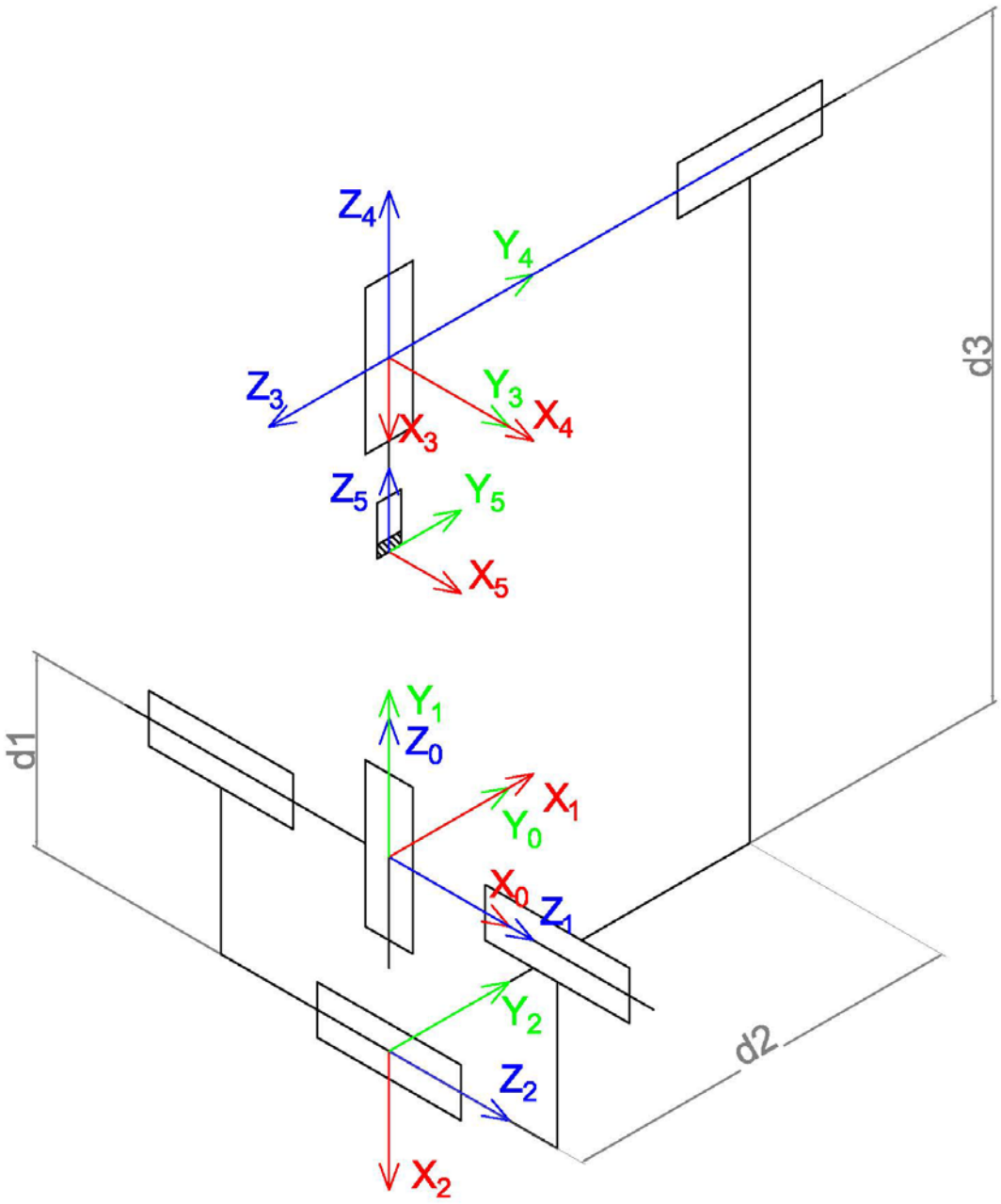
Feed drives				
Recommended max. working feed	m / min	15	15	15
Rapid traverse	m / min	22	22	22
Feed force X and Y/Z		5'000N	5'000N	5'000N
Tool changer				
Magazine tool capacity for 12 / 20'000	No.	30 (opt. 58)	30 (opt. 58)	30 (opt. 58)
Magazine tool capacity for 42'000	No.	36	36	36
Max. tool diameter for 12 / 20'000	mm	90	90	90
Max. tool diameter for 42'000	mm	16	16	16
Max. tool length for 12/20'000	mm	250 (350 manual)	250 (350 manual)	250 (350 manual)
Max. tool length for für 42'000	mm	100	100	100
Max. tool weight for 12 / 20'000	kg	6	6	6
Max. tool weight for 42'000	kg	1,5	1,5	1,5
Tool change time	sec	8	8	8
Chip-to-chip times to VDI	sec	10	10	10
Quality data				
Positional tolerance DIN / ISO 230-2/97	µm	8	8	8
Control unit				
with digital drives	iTNC 530	•	•	•
No. of axes	5 + spindle	•	•	•
Linear interpolation	5 axes out of 5	•	•	•
Circular interpolation	2 axes out of 5	•	•	•
Safety equipment				
Machine enclosure/complete machine enclosure		•	•	•
Door opening to the front	mm	950	1200	950
Door opening to the side	mm	850	850	850
Coolant				
Coolant tank	l	120	140	120
Coolant flow	30 l / min – 2,5 bar	•	•	•

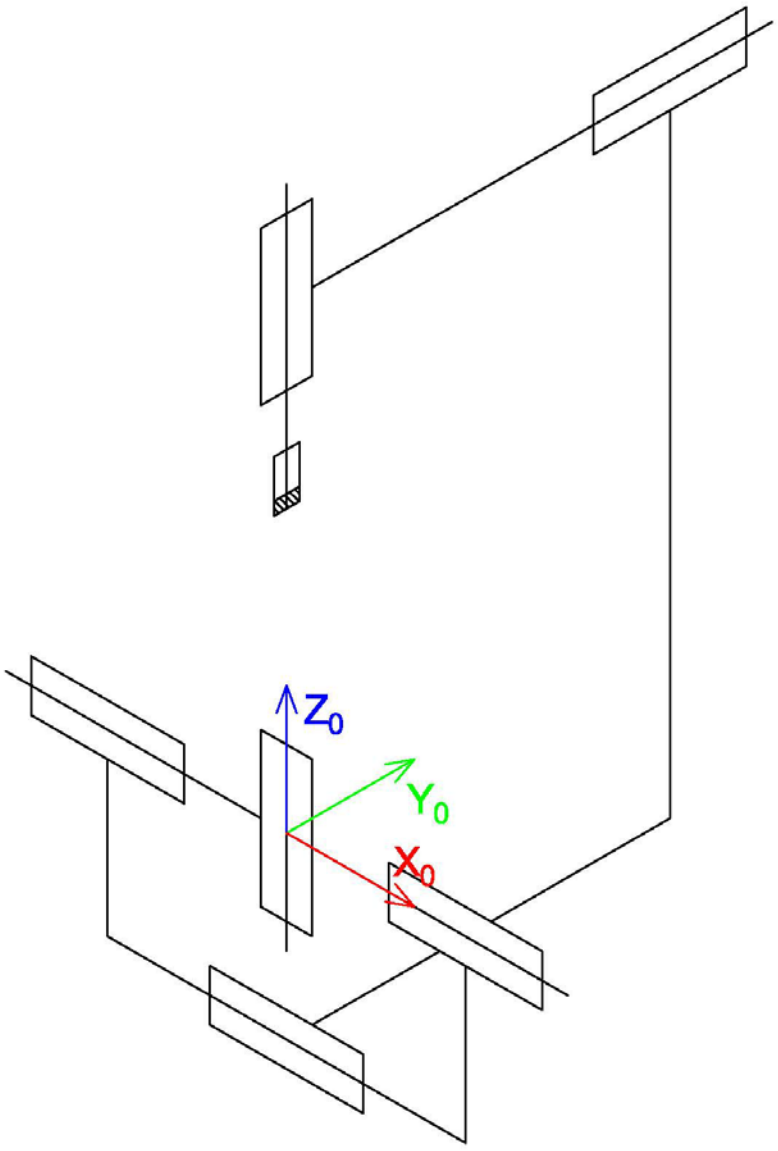
		VCP 600	VCP 800	UCP 600
Weight				
Approx. machine weight	kg	6'100	6'400	6'400
Transport dimensions	mm	1'600 x 2'500 x 2'650	2'000 x 2'500 x 2'650	1'600 x 2'500 x 2'650
Accessories				
Scrapper chip conveyor		100 l – 2,5 bar	120 l – 2,5 bar	100 l – 2,5 bar
Coolant with chip conveyor and filter (for through tool coolant supply)	580 l – 18, 40 or 70 bar	•	•	•
Coolant washdown gun		•	•	•
Oil mist extraction		•	•	•
OMP 40 setup probe		•	•	•
Tool measurment (sensor or laser)		•	•	•
Oil mist coolant spray unit		•	•	•
Dividing head		•	•	
Graphite dust extraction		•	•	•
Small pallet changer for 24 small pieces/electrodes		•	•	
Connection data				
Rated power (average value)	kW	15	15	15
Total power requirements (maximum value)	kVA	44	44	44
Fuse protection	A	30	30	30
Line voltage/Frequency	3 x 400 V–50 / 60 Hz	•	•	•
Connection cross section	mm^2	6	6	6
Control voltage	VDC	24	24	24
Pneumatic connection	200 l / min – 6 bar	•	•	•
Connection diameter	mm	12	12	12
Work tables		Work table	Work table	Rotary tilting table
Table surface area	mm	850 x 530	1050 x 590	Ø 280, Ø 400
Table load	kg	400	400	200
Clamping		14 – H12 (H7)	14 – H12 (H7)	various clamping systems
Space between slots	mm	63	63	
Tilt range	°			+122 / –100
Tilt speed (30% ED)	min^{-1}			20
Rotary speed	min^{-1}			30

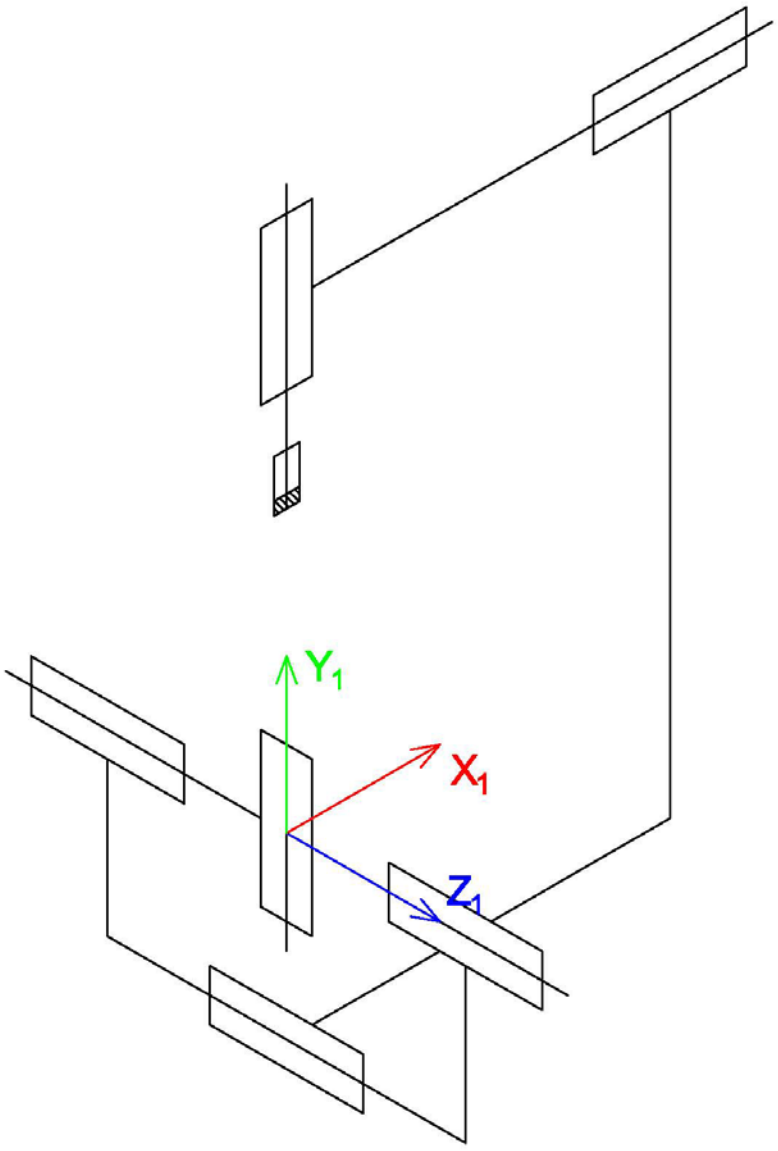


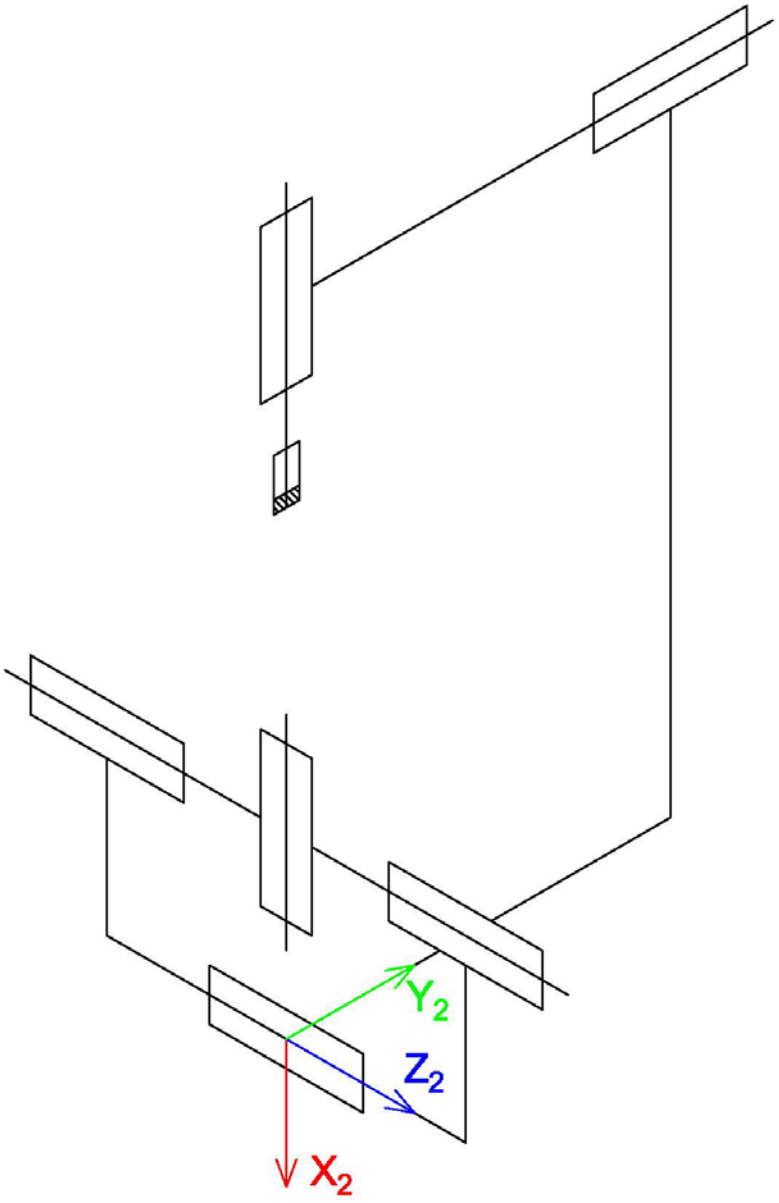
10.2 Appendix 2: kinematic structure

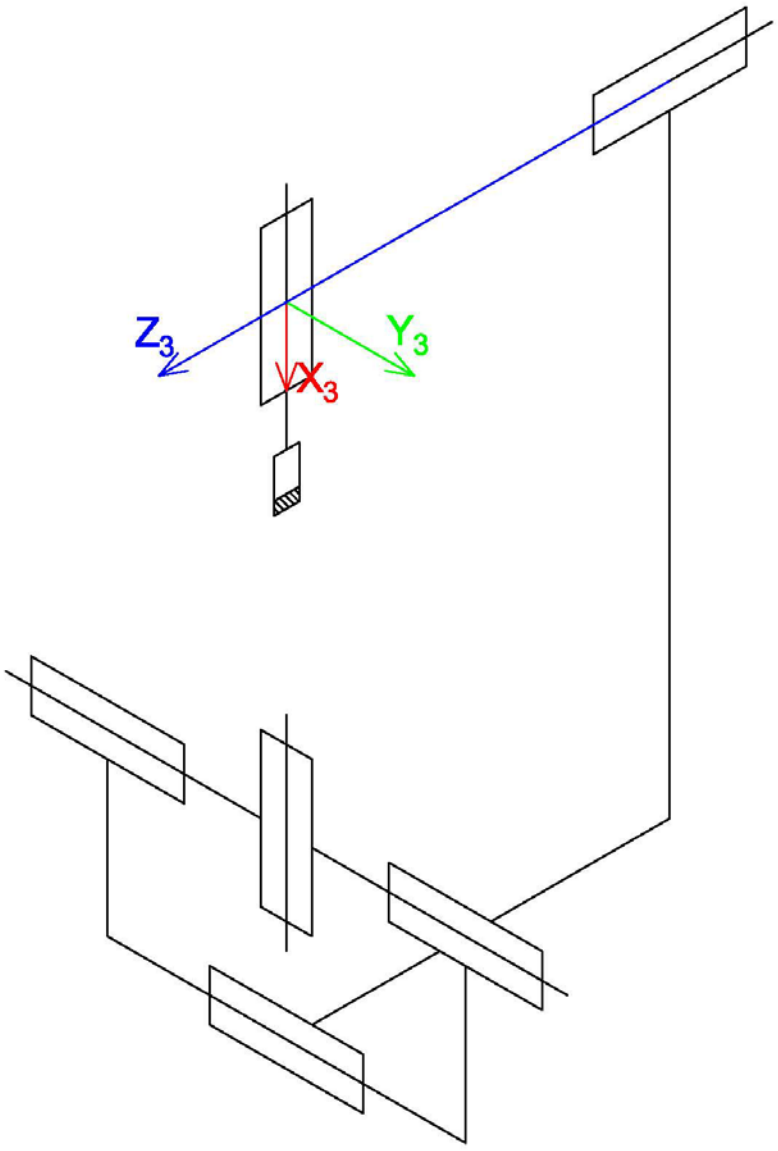


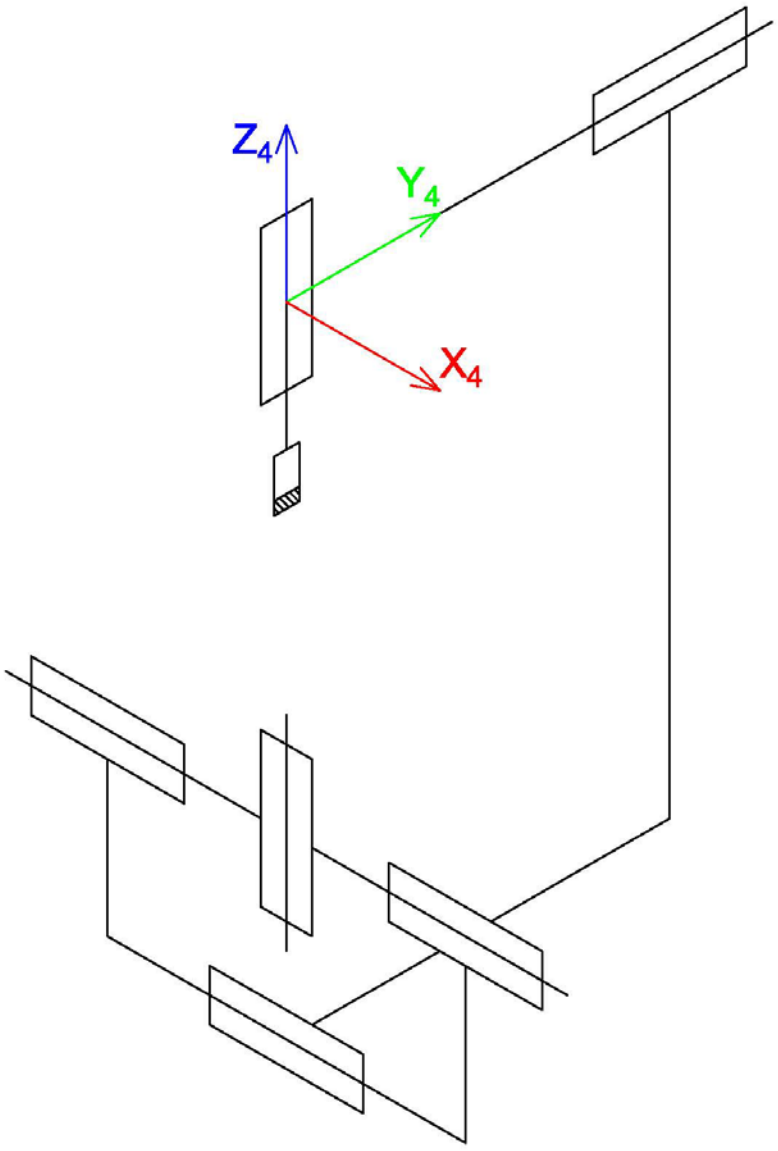

Z4
Y4
Z3
Y3
X4
X3
Z5
Y5
X5
d3
Y1
Z0
X1
Y0
X0
Z1
d1
Y2
Z2
d2
X2

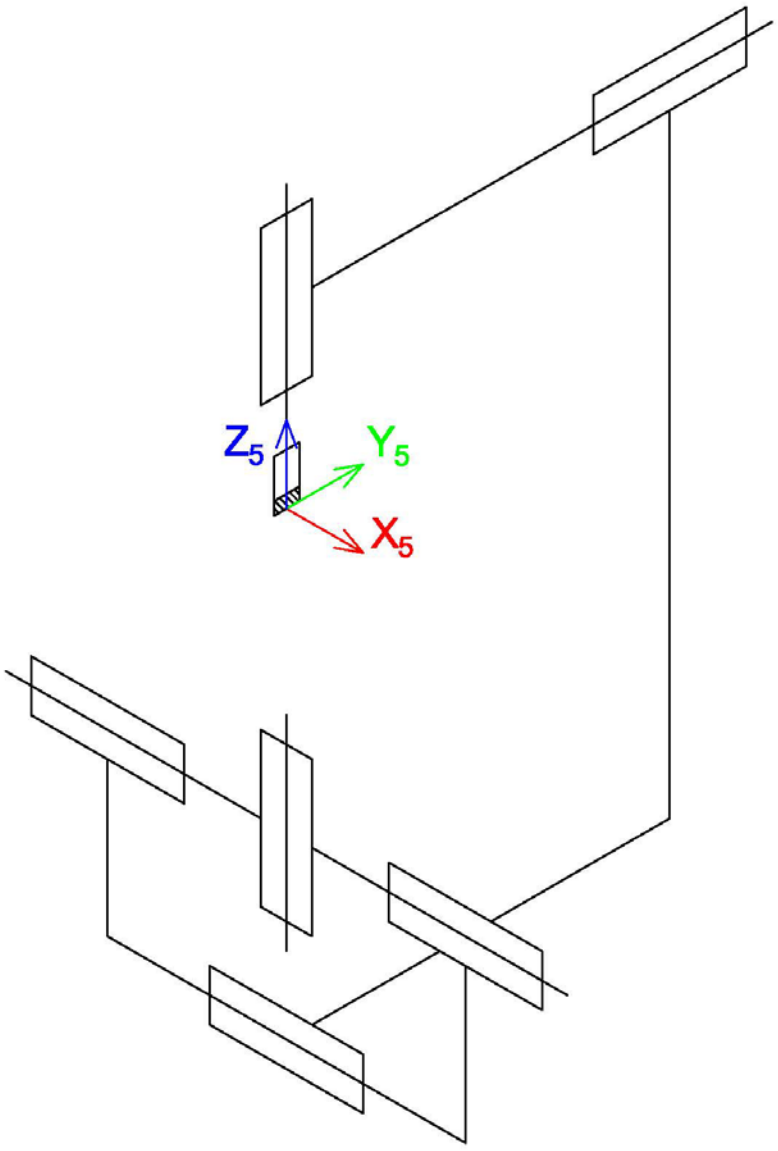

Z0
Y0
X0


Y1
X1
Z1


Y2
Z2
X2


Z3
Y3
X3


Z4
Y4
X4


Z5
Y5
X5

10.3 Appendix 3: DFPOST

1. Tape information

1. Maximum no. of blocks in tape <>
2. Maximum tape length (in meters) <>
3. Subroutines in separate files <NO>

2. Programming mode and unit

1. Programming mode <BOTH>
2. Code for absolute coordinates <G90>
3. Code for incremental coordinates <G91>
4. Length unit of measurement <METRIC >
5. Factor for length units <1.0>
6. Tool Start Point <TP HOME>
7. Coordinate orientation <MACSYS>

3. Formats

1. Coordinates <430000~-.101>
2. Angles <430000~-.101>
3. Feed <5 1001~~ 101>
4. Spindle_speed <5 1001~~.101>
5. Dwell <430000~~.101>
6. Sequencing <5 1001~~ 101>
7. Tool <4 0000~~.001>
8. User_1 <430000~-.001>
9. User_2 <430000~-.001>
10. User_3 <430000~-.101>
11. User_4 <430000~-.101>
12. User_5 <430000~-.101>
13. User_6 <430000~-.101>
14. User_7 <430000~-.101>
15. User_8 <430000~-.101>
16. User_9 <430000~-.101>
17. User_10 <430000~-.101>
18. Real <355511~-.100>

4. Positioning codes

1. Code for A axis <A>
2. Code for B axis <B>
3. Code for C axis <C>

5. Messages and inserts

1. Max no. of messages <>
2. Max no. of characters in messages <>

6. Machine parameters

1.	Code for clockwise spin	<M3>
2.	Code for counterclockwise spin	<M4>
3.	Code for spin stop	<M5>
4.	Code for flood on	<M8>
5.	Code for mist on	<M7>
6.	Code for air on	<>
7.	Code for through on	<>
8.	Code for coolant off	<M9>
9.	Cutter compensation off	<G40>
10.	Cutter compensation left	<G41>
11.	Cutter compensation right	<G42>

7. Linear motion

1.	Code for rapid motion (FAST)	<G0>
2.	Code for cutting/feed motion	<G1>
3.	Maximum length of movement	<>
4.	Maximum X dimension for part	<530>
5.	Maximum Y dimension for part	<450>
6.	Maximum Z dimension for part	<450>
7.	Machine speed for rapid motion in X (mm/m)	<22000.0>
8.	Machine speed for rapid motion in Y (mm/m)	<22000.0>
9.	Machine speed for rapid motion in Z (mm/m)	<22000.0>
10.	Machine speed for table positioning (rpm)	<999.0>
11.	User speed for rapid motion (MAX FEED)	<NO>
12.	User speed for rapid motion (MAX FEED)	N.A.

8. Circular motion

1.	Code for clockwise motion is	<G02>
2.	Code for counterclockwise is	<G03>
3.	Angles limited to quadrant?	<NO>
4.	Tolerance for linear approximation is	<0.01>
5.	Min. segments for linear approximation is	<4>

9. Nurbs motions

1.	Code for nurbs motion	<G6.2>
2.	Knot normal factor	<YES>
3.	Knot normal factor is	<10000>
4.	Linear approximation	<NO>
5.	Tolerance for linear approximation	<0.01>

10. Canned cycles

1. Code for high speed peck <G73>
2. Code for left hand tapping <G74>
3. Code for fine boring <G76>
4. Code for spot drill <G81>
5. Code for counter boring <G82>
6. Code for deep hole peck <G83>
7. Code for tapping <G84>
8. Code for boring <G85>
9. Code for boring + spindle stop <G86>
10. Code for back boring <G87>
11. Code for boring + dwell + manual out <G88>
12. Code for boring + dwell + feed <G89>
13. Code for retract to initial point <G98>
14. Code for retract to clearance <G99>

11. Output files

1. Tools file <NO>
2. Origins file <NO>
3. Messages file <NO>
4. Cycles file <NO>
5. PRINT1 file extension pr1
6. PRINT2 file extension pr2
7. PRINT3 file extension pr3
8. PRINT4 file extension pr4
9. PRINT5 file extension pr5
10. PRINT6 file extension pr6
11. PRINT7 file extension pr7
12. PRINT8 file extension pr8
13. PRINT9 file extension pr9
14. PRINT10 file extension pr10
15. Run script file after Post? <NO>

12. POSTPR/EXPST interface

1. Ask for machine zero? , <MACSYS>
2. Ask for UCS order? <NO>
3. Ask to save a session? <NO>
4. Ask to send files to screen? <NO>

(Cimatron)

10.4 Appendix 4: Program file

```
*****************************************************
* define private variables:
FORMAT (SEQUENCING)  Seq SubSeq CNTRL_NUM NURBS_DEG count first deg ;
FORMAT (TOOL)       CutterComp FirstTool LastTool ;
FORMAT (DWELL)              CurDwell ;
FORMAT (COORDINATES) Xold Yold Zold DXcenter DYcenter DZcenter ;
FORMAT (COORDINATES) Zinit Clear Depth CNTRL_X  CNTRL_Y  CNTRL_Z  KNOT_ ;
FORMAT (COORDINATES) Xhome Yhome Zhome ;
FORMAT (USER_1)     CurrSubNum ;


* 5 axes

FORMAT (REAL) A C I J K S AR CR;
FORMAT (REAL) COSA SINA COSC SINC;
FORMAT (REAL) AROT CROT;
FORMAT (COORDINATES) FIRSTORIGINX FIRSTORIGINY FIRSTORIGINZ ;
FORMAT (COORDINATES) X_ORIGIN Y_ORIGIN Z_ORIGIN ;
FORMAT (COORDINATES) TOTALDX TOTALDY TOTALDZ;
FORMAT (COORDINATES) TOTALAFTERROTX TOTALAFTERROTY TOTALAFTERROTZ;
FORMAT (COORDINATES) DELTAORIGINX DELTAORIGINY DELTAORIGINZ ;
FORMAT (COORDINATES) DELTAFIRSTX DELTAFIRSTY DELTAFIRSTZ ;

* define private flags:
FORMAT (USER_2)     FlagSub FlagSeq FlagSpin;
FORMAT (USER_2)     FirstOriginChange Flagrotmac  ;

* define private constants:
FORMAT (USER_2)     YES NO ;

* change the format of existing variables:
FORMAT (USER_1)     SUB_NUMBER ;

*****************************************************
INTERACTION (USER_1)     "MAIN-PROGRAM-NUMBER"    MainNum     = 100  ;
INTERACTION (TOOL)       "DIACOMP=TOOL+<xx>"      DiaComp     = 50   ;
INTERACTION (USER_1)     "TOOL-CHANGE-PROGRAM"    ChangeTool  = 8000 ;
INTERACTION (CHARACTER)  "SEQUENCING<Y/N>"        NumYN       = "N"  ;
INTERACTION (SEQUENCING) "SEQUENC-START"          SeqStart    = 10   ;
INTERACTION (SEQUENCING) "SEQUENC-INCR."          SeqIncr     = 10   ;
INTERACTION (CHARACTER)  "SUBROUTINES<Y/N>"       Sub         = "Y"  ;
INTERACTION (USER_1)     "SUB-PROGRAM-NUMBER"     StartSubNum = 1000 ;
INTERACTION (COORDINATES)  "ENTER_TABLEDX"                          TABLEDX = 0;
INTERACTION (COORDINATES)  "ENTER_TABLEDY"                          TABLEDY = 0;
INTERACTION (COORDINATES)  "ENTER_TABLEDZ"                          TABLEDZ = 0;
*****************************************************
NON_MODAL  ALL_VAR;
MODAL     X_CURPOS Y_CURPOS Z_CURPOS ;
```

```
MODAL     LIN_MOV CIRC_MOV MCH_FEED SPIN_SPEED SPIN_DIR MCH_COOL MCH_DWELL ;
MODAL     CUTCOM_ON  CUTCOM_OFF ;
MODAL     CYC_DEPTH CYC_PECK CYC_DWELL CYC_RETR CYC_CLEAR Depth Clear NURBS_MOV;

IDENTICAL  X_CURPOS X_ENDPT ;
IDENTICAL  Y_CURPOS Y_ENDPT ;

*******************************************************
NEW_LINE_IS $ ;
    IF_SET (FlagSeq _EQ_ NO)
          OUTPUT \J ;
    ELSE
          IF_SET (FlagSub _EQ_ NO)
            OUTPUT \J "N" Seq ;
            Seq = Seq + SeqIncr ;
          ELSE
            OUTPUT \J "N" SubSeq ;
            SubSeq = SubSeq + SeqIncr ;
          END_IF ;
    END_IF ;

*******************************************************
BEGINNING OF TAPE:
    YES           = 1 ;
    NO            = 0 ;
    Seq           = SeqStart ;

    FlagSeq = NO ;
    FlagSub = NO ;
    IF_SET (NumYN _EQ_ "y") FlagSeq = YES ; END_IF ;
    IF_SET (NumYN _EQ_ "Y") FlagSeq = YES ; END_IF ;

    IF_SET (Sub   _EQ_ "y") Sub     = "Y" ; END_IF ;
    IF_SET (Sub   _EQ_ "Y")
          SET_OFF NO_SUBROUT ;
    ELSE
          SET_ON  NO_SUBROUT ;
    END_IF ;

*-----Unit matrix (MUST be input in this order !!!)
    ROT_MAT1   = 1.0 ; ROT_MAT2   = 0.0 ; ROT_MAT3   = 0.0 ;
    ROT_MAT4   = 0.0 ; ROT_MAT5   = 1.0 ; ROT_MAT6   = 0.0 ;
    ROT_MAT7   = 0.0 ; ROT_MAT8   = 0.0 ; ROT_MAT9   = 1.0 ;

*-----shift all data according to the MCHINE ZERO indicated by the
*     user in the POSTPR interaction
    TRANS_MATX = 0 - X_MACH  ;
    TRANS_MATY = 0 - Y_MACH  ;
    TRANS_MATZ = 0 - Z_MACH  ;

*-----shift the HOME according to the MCHINE ZERO indicated by the
*     user in the POSTPR interaction
```

```
    Xhome = X_HOME - X_MACH  ;
    Yhome = Y_HOME - Y_MACH  ;
    Zhome = Z_HOME - Z_MACH  ;

*-----tool location is HOME
    Xold = X_HOME ;
    Yold = Y_HOME ;
    Zold = Z_HOME ;

*-----For 1st ORIGIN CHANGE
    FirstOriginChange = YES ;

    MCH_FEED   = 9999 ;

*-----output
    IF_SET (FlagSeq _EQ_ YES )
          OUTPUT "%" \J "O" MainNum  ;
    ELSE
          OUTPUT " %" \J " O" MainNum  ;
    END_IF ;

*--For the first origin change
  flagrotmac = no ;



BEGINNING OF PROC:
    KEEP   PROC_NAME ;
    SET_ON MCH_COOL ;
    FlagSub = NO ;
    SubSeq  = SeqStart ;
    CurDwell = 0.0;


END OF TAPE:
   IF_SET (FirstTool _NE_ LastTool)
                  OUTPUT $ " T" NEXT_TOOL " M98 P" ChangeTool;
   END_IF ;
         OUTPUT $;
         OUTPUT " G00 C0.0 A0.0";
    OUTPUT $ " M30 " ;


END OF FILE:
    IF_SET (FlagSeq _EQ_ YES )
          OUTPUT \J "%" ;
    ELSE
          OUTPUT \J " %" ;
    END_IF ;


*******************************************************
```

```
FEED:
    KEEP MCH_FEED ;

SPIN:
    KEEP SPIN_SPEED ;

COOLANT:
    KEEP MCH_COOL ;

DWELL:
    KEEP MCH_DWELL ;

MillDWELL:
    KEEP DWELL_TIME ;
    CurDwell =  DWELL_TIME ;

CUTTER COMPENSATION:
    KEEP  CUTCOM_ON ;

CUTTER COMPENSATION: COFF:
    KEEP  CUTCOM_OFF ;

*****************************************************
TOOL CHANGE: FIRST:
    SET_ON  SPIN_SPEED SPIN_DIR ;
*-----save first tool number
    FirstTool = CURR_TOOL ;
    LastTool  = CURR_TOOL ;

*-----tool cutter compensation register number
    CutterComp = CURR_TOOL +  DiaComp ;

*-----assuming 1st tool is in the spindle, there is no need
*     for tool change)
    OUTPUT $ " T" CURR_TOOL ;
*-----more then one tool in this run of the Post Processor
    IF_SET (NEXT_TOOL _NE_ CURR_TOOL)
          OUTPUT $ " T" NEXT_TOOL ;
    END_IF ;
*-----standard tool change commands. (assuming 1st tool is in
    OUTPUT $ " G90 G80 G00 G17 G40" ;
    OUTPUT $ " G43 H" CURR_TOOL " Z" Zhome " S" SPIN_SPEED " " SPIN_DIR ;

TOOL CHANGE:
    SET_ON  SPIN_SPEED SPIN_DIR ;

*-----tool cutter compensation register number
    CutterComp = CURR_TOOL +  DiaComp ;

*-----tool change command (by using an internal tool change
*     subroutine)
    OUTPUT $ " T" CURR_TOOL " M98 P"  ChangeTool ;
```

```
    OUTPUT $ " T" NEXT_TOOL ;
*-----standard tool change commands. (assuming 1st tool is in
    OUTPUT $ " G90 G80 G00 G17 G40" ;
    OUTPUT $ " G43 H" NEXT_TOOL " Z" Zhome " S" SPIN_SPEED " " SPIN_DIR ;

TOOL CHANGE: LAST:
    SET_ON  SPIN_SPEED SPIN_DIR ;
*-----save last tool number
    LastTool = CURR_TOOL ;

*-----tool cutter compensation register number
    CutterComp = CURR_TOOL +  DiaComp ;

    OUTPUT $ " T" CURR_TOOL " M98 P"  ChangeTool ;
*-----the last tool is the same as the first tool
    IF_SET (NEXT_TOOL _NE_ CURR_TOOL)
          OUTPUT $ " T" NEXT_TOOL ;
    END_IF ;
*-----standard tool change commands. (assuming 1st tool is in
    OUTPUT $ " G90 G80 G00 G17 G40" ;
    OUTPUT $ " G43 H" CURR_TOOL " Z" Zhome " S" SPIN_SPEED " " SPIN_DIR ;

**************************************************
LINEAR MOTION: FAST:

          IF_SET (FLAGROTMAC)
                    OUTPUT \j "lin mot fast";
                    OUTPUT $;
                    OUTPUT " G00";
                    OUTPUT " C" CROT;
                    OUTPUT " A" AROT;
          END_IF;

    FlagSpin = NO ;
    IF_SET (SPIN_SPEED) FlagSpin = YES ; END_IF ;
    IF_SET (SPIN_DIR)   FlagSpin = YES ; END_IF ;
    IF_SET (FlagSpin _EQ_ YES)
          SET_ON SPIN_SPEED SPIN_DIR ;
          OUTPUT $ " S" SPIN_SPEED " " SPIN_DIR ;
          if_SET (CurDwell _NE_ 0.0)
                    OUTPUT $ " G04 P" CurDwell ;
                    CurDwell = 0.0 ;
          END_IF ;
    END_IF ;

    OUTPUT $ ;
    IF_SET (LIN_MOV)  OUTPUT " " LIN_MOV   ; END_IF ;
    IF_SET (X_CURPOS) OUTPUT " X" X_CURPOS ; END_IF ;
    IF_SET (Y_CURPOS) OUTPUT " Y" Y_CURPOS ; END_IF ;
    IF_SET (Z_CURPOS) OUTPUT " Z" Z_CURPOS ; END_IF ;
    IF_SET (MCH_COOL) OUTPUT  " " MCH_COOL ; END_IF ;
    SET_ON CIRC_MOV NURBS_MOV ;
```

```
    Xold = X_CURPOS ;
    Yold = Y_CURPOS ;
    Zold = Z_CURPOS ;



LINEAR MOTION:
        IF_SET (FLAGROTMAC)
                OUTPUT \j "lin mot";
                OUTPUT $;
                OUTPUT " G00";
                OUTPUT " C" CROT;
                OUTPUT " A" AROT;
        END_IF;

    FlagSpin = NO ;
    IF_SET (SPIN_SPEED) FlagSpin = YES ; END_IF ;
    IF_SET (SPIN_DIR)   FlagSpin = YES ; END_IF ;
    IF_SET (FlagSpin _EQ_ YES)
         SET_ON SPIN_SPEED SPIN_DIR ;
         OUTPUT $ " S" SPIN_SPEED " " SPIN_DIR ;
         if_SET (CurDwell _NE_ 0.0)
                OUTPUT $ " G04 P" CurDwell ;
                CurDwell = 0.0 ;
         END_IF ;
   END_IF ;

    OUTPUT $ ;
    IF_SET (LIN_MOV)    OUTPUT " "  LIN_MOV    ; END_IF ;
    IF_SET (CUTCOM_ON)  OUTPUT " "  CUTCOM_ON " D" CutterComp ; END_IF ;
    IF_SET (CUTCOM_OFF) OUTPUT " "  CUTCOM_OFF ; END_IF ;
    IF_SET (X_CURPOS)   OUTPUT " X" X_CURPOS   ; END_IF ;
    IF_SET (Y_CURPOS)   OUTPUT " Y" Y_CURPOS   ; END_IF ;
    IF_SET (Z_CURPOS)   OUTPUT " Z" Z_CURPOS   ; END_IF ;
    IF_SET (MCH_FEED)   OUTPUT " F" MCH_FEED   ; END_IF ;
    IF_SET (MCH_COOL)   OUTPUT  " " MCH_COOL   ; END_IF ;
    SET_ON CIRC_MOV NURBS_MOV;
    Xold = X_CURPOS ;
    Yold = Y_CURPOS ;
    Zold = Z_CURPOS ;

*****************************************************
CIRCULAR MOTION:
        IF_SET (FLAGROTMAC)
                OUTPUT \j "circ mot";
                OUTPUT $;
                OUTPUT " G00";
                OUTPUT " C" CROT;
                OUTPUT " A" AROT;
        END_IF;

    FlagSpin = NO ;
```

```
    IF_SET (SPIN_SPEED) FlagSpin = YES ; END_IF ;
    IF_SET (SPIN_DIR)   FlagSpin = YES ; END_IF ;
    IF_SET (FlagSpin _EQ_ YES)
          SET_ON SPIN_SPEED SPIN_DIR ;
          OUTPUT $ " S" SPIN_SPEED " " SPIN_DIR ;
          if_SET (CurDwell _NE_ 0.0)
                OUTPUT $ " G04 P" CurDwell ;
                CurDwell = 0.0 ;
          END_IF ;
   END_IF ;

    DXcenter = X_CENTER - Xold ;
    DYcenter = Y_CENTER - Yold ;
    DZcenter = Z_CENTER - Zold ;
    OUTPUT $ ;
    IF_SET (CIRC_MOV)        OUTPUT " " CIRC_MOV  ; END_IF ;
    IF_SET (X_CURPOS)        OUTPUT " X" X_ENDPT  ; END_IF ;
    IF_SET (Y_CURPOS)        OUTPUT " Y" Y_ENDPT  ; END_IF ;
    IF_SET (Z_CURPOS)        OUTPUT " Z" Z_ENDPT  ; END_IF ;
    IF_SET (DXcenter _NE_ 0.0) OUTPUT " I" DXcenter ; END_IF ;
    IF_SET (DYcenter _NE_ 0.0) OUTPUT " J" DYcenter ; END_IF ;
    IF_SET (DZcenter _NE_ 0.0) OUTPUT " K" DZcenter ; END_IF ;
    IF_SET (MCH_FEED)        OUTPUT " F" MCH_FEED ; END_IF ;
    IF_SET (MCH_COOL)        OUTPUT  " " MCH_COOL ; END_IF ;
    SET_ON LIN_MOV NURBS_MOV ;
    Xold = X_CURPOS ;
    Yold = Y_CURPOS ;
    Zold = Z_CURPOS ;
*****************************************************************
NURBS MOTION: START:
          IF_SET (FLAGROTMAC)
                OUTPUT $;
                OUTPUT \j "nurbs";
                OUTPUT " G00";
                OUTPUT " C" CROT;
                OUTPUT " A" AROT;
          END_IF;

          OUTPUT $ NURBS_MOV "P" NURBS_DEG ;
          count = 0;
          first = 1;
          SET_ON MCH_FEED;
          NURBS MOTION:
          IF_SET (first _EQ_ 1)
                OUTPUT "K" KNOT_ "X" CNTRL_X "Y" CNTRL_Y "Z" CNTRL_Z "F" MCH_FEED;
          END_IF;
          IF_SET (first _EQ_ 0)
                OUTPUT $ "K" KNOT_ "X" CNTRL_X "Y" CNTRL_Y "Z" CNTRL_Z;
          END_IF;
          first = 0;
          NURBS MOTION: END:
          deg = NURBS_DEG ;
```

```
        REPEAT
        count = count + 1 ;
        OUTPUT $ "K" KNOT_ ;
        UNTIL (count _EQ_ deg) ;
        SET_ON LIN_MOV CIRC_MOV MCH_FEED;
*****************************************************
INSERT WITH:
   OUTPUT $ INS_STR ;

INSERT WITHOUT:
   OUTPUT  \J INS_STR ;

MESSAGE:
   OUTPUT $ "( " MESS_STR " )"  ;

*****************************************************
CYCLE: ON:
        IF_SET (FLAGROTMAC)
                OUTPUT $;
                OUTPUT " G00";
                OUTPUT " C" CROT;
                OUTPUT " A" AROT;
        END_IF;

   FlagSpin = NO ;
   IF_SET (SPIN_SPEED) FlagSpin = YES ; END_IF ;
   IF_SET (SPIN_DIR)   FlagSpin = YES ; END_IF ;
   IF_SET (FlagSpin _EQ_ YES)
        SET_ON SPIN_SPEED SPIN_DIR ;
        OUTPUT $ " S" SPIN_SPEED " " SPIN_DIR ;
   END_IF ;

   SET_ON X_CURPOS Y_CURPOS Z_CURPOS   ;
   SET_ON CYC_DEPTH CYC_RETR CYC_CLEAR MCH_FEED MCH_COOL ;
   SET_ON Zinit Depth Clear ;
   Zinit    = Z_CURPOS + CYC_DZINIT ;
   Depth    = Z_CURPOS - CYC_DEPTH ;
   Clear    = Z_CURPOS + CYC_CLEAR ;
   IF_SET (Zold _LT_ Zinit)
                        OUTPUT $ " Z" Zinit ;
        ELSE
                        OUTPUT $ " Z" Zold ;
        END_IF ;
                         OUTPUT $ ;
                         OUTPUT " "  CYC_RETR ;
                         OUTPUT " "  CYC_CODE ;
                         OUTPUT " X" X_CURPOS ;
                         OUTPUT " Y" Y_CURPOS ;
                         OUTPUT " Z" Depth    ;
                         OUTPUT " R" Clear ;
   IF_SET (CYC_PECK)  OUTPUT " Q" CYC_PECK  ; END_IF ;
   IF_SET (CYC_DWELL) OUTPUT " P" CYC_DWELL ; END_IF ;
```

```
    IF_SET (CYC_XSHFT) OUTPUT " I" CYC_XSHFT ; END_IF ;
    IF_SET (CYC_YSHFT) OUTPUT " J" CYC_YSHFT ; END_IF ;
                       OUTPUT " F" MCH_FEED ;
                       OUTPUT " "  MCH_COOL ;
    SET_ON  LIN_MOV CIRC_MOV ;
    SET_OFF CYC_DEPTH CYC_CLEAR;
    Xold = X_CURPOS ;
    Yold = Y_CURPOS ;
    Zold = Z_CURPOS ;



CYCLE:
    Depth = Z_CURPOS - CYC_DEPTH ;
    Clear = Z_CURPOS + CYC_CLEAR ;
    OUTPUT $ ;
    IF_SET (CYC_RETR)   OUTPUT " "  CYC_RETR   ; END_IF ;
    IF_SET (X_CURPOS)   OUTPUT " X" X_CURPOS   ; END_IF ;
    IF_SET (Y_CURPOS)   OUTPUT " Y" Y_CURPOS   ; END_IF ;
    IF_SET (Depth)      OUTPUT " Z" Depth      ; END_IF ;
    IF_SET (Clear)      OUTPUT " R" Clear      ; END_IF ;
    IF_SET (CYC_PECK)   OUTPUT " Q" CYC_PECK   ; END_IF ;
    IF_SET (CYC_DWELL)  OUTPUT " P" CYC_DWELL  ; END_IF ;
    IF_SET (CYC_XSHFT)  OUTPUT " I" CYC_XSHFT  ; END_IF ;
    IF_SET (CYC_YSHFT)  OUTPUT " J" CYC_YSHFT  ; END_IF ;
    IF_SET (SPIN_SPEED) OUTPUT " S" SPIN_SPEED ; END_IF ;
    IF_SET (SPIN_DIR)   OUTPUT " "  SPIN_DIR   ; END_IF ;
    IF_SET (MCH_FEED)   OUTPUT " F" MCH_FEED   ; END_IF ;
    IF_SET (MCH_COOL)   OUTPUT " "  MCH_COOL   ; END_IF ;
    SET_ON  LIN_MOV CIRC_MOV ;
    SET_OFF CYC_DEPTH CYC_CLEAR;
    Xold = X_CURPOS ;
    Yold = Y_CURPOS ;
    Zold = Z_CURPOS ;

CYCLE: OFF:
    OUTPUT $ " G80 Z" Zinit ;
    SET_ON  LIN_MOV CIRC_MOV ;
    Zold = Zinit ;

*****************************************************
ORIGIN CHANGE:
* Save the first origin data for later use
  IF_SET (FirstOriginChange _EQ_ YES )
          FIRSTORIGINX = X_ORIGIN  ;
          FIRSTORIGINY = Y_ORIGIN  ;
          FIRSTORIGINZ = Z_ORIGIN  ;
          FirstOriginChange = NO       ;
  END_IF ;


* check for change in orientation ( 5 axis positioining )
```

```
   Flagrotmac = no ;
  IF_SET ( MK_ORIGIN _NE_ 1 ) FLAGROTMAC = YES ; END_IF ;

* ROTMAC - change the tool orientation
        IF_SET(FLAGROTMAC _EQ_ YES)

* Calculate the rotation angle of the C axis, around Z.
        I = MI_ORIGIN; J = MJ_ORIGIN; K = MK_ORIGIN;

        IF_SET (J _EQ_ 0)
                IF_SET(I _LT_ 0)
                        C = -90;
                ELSE
                        C = 90;
                END_IF;
        ELSE
        C = ATAN(I/J);
        END_IF;

        CROT = -C;
        COSC = COS(C); SINC = SIN(C);

* Calculate the rotation angle of the A axis, around X.
        IF_SET (J _NE_ 0)
        S = J/COSC;
        ELSE
        S = I;
        END_IF;

        IF_SET (K _EQ_ 0)
                IF_SET (J _EQ_ 0)
                        A = 90;
                END_IF;
                IF_SET (J _LT_ 0)
                        A=-90;
                END_IF;
                IF_SET(J _GT_ 0)
                        A=90;
                END_IF;
        ELSE
                A = ATAN(S/K);
        END_IF;

        AROT = -A;
        COSA = COS(A); SINA = SIN(A);


* Calculate the ROT_MAT values for the rotating table machine
        ROT_MAT1 = COSC ; ROT_MAT2 = -SINC ; ROT_MAT3 = 0 ;
        ROT_MAT4 = SINC* COSA ; ROT_MAT5 = COSC*COSA ; ROT_MAT6 = -SINA ;
        ROT_MAT7 = SINC*SINA ; ROT_MAT8 = COSC*SINA ; ROT_MAT9 = COSA ;
```

```
* distance between the rotation center and the ORIGIN
        TOTALDX = X_ORIGIN - X_MACH - TABLEDX;
        TOTALDY = Y_ORIGIN - Y_MACH - TABLEDY;
        TOTALDZ = Z_ORIGIN - Z_MACH - TABLEDZ;

* distance between the rotation center and the ORIGIN
* after the rotation
        TOTALAFTERROTX = TOTALDX*ROT_MAT1 + TOTALDY*ROT_MAT2;
        TOTALAFTERROTY = TOTALDX*ROT_MAT4 + TOTALDY*ROT_MAT5 + TOTALDZ*ROT_MAT6;
        TOTALAFTERROTZ = TOTALDX*ROT_MAT7 + TOTALDY*ROT_MAT8 + TOTALDZ*ROT_MAT9;


* distance between the ORIGIN and the new position of the ORIGIN
* after the rotation
        DELTAORIGINX = TOTALAFTERROTX - TOTALDX;
        DELTAORIGINY = TOTALAFTERROTY - TOTALDY;
        DELTAORIGINZ = TOTALAFTERROTZ - TOTALDZ;

* calculate the distance between the ORIGIN and the FIRST ORIGIN.
* In the case of MULTIORIGINS output, the parameters may be used
* to calculate the MACHINE position of each ORIGIN related to the
* first one. In the case of ONEORIGIN output, TRANS_MAT parameters
* should get the values of DELTAFIRSTX..DELTAFIRSTZ.
        DELTAFIRSTX = DELTAORIGINX + (X_ORIGIN - FIRSTORIGINX);
        DELTAFIRSTY = DELTAORIGINY + (Y_ORIGIN - FIRSTORIGINY);
        DELTAFIRSTZ = DELTAORIGINZ + (Z_ORIGIN - FIRSTORIGINZ);


ELSE

* origin change without rotation.
* In the case of ONEORIGIN output, TRANS_MAT parameters should get
* the values of DELTAFIRSTX..DELTAFIRSTZ.
        DELTAFIRSTX = X_ORIGIN - FIRSTORIGINX;
        DELTAFIRSTY = Y_ORIGIN - FIRSTORIGINY;
        DELTAFIRSTZ = Z_ORIGIN - FIRSTORIGINZ;



END_IF;

* apply the transmat values
        TRANS_MATX = - DELTAFIRSTX;
        TRANS_MATY = - DELTAFIRSTY;
        TRANS_MATZ = - DELTAFIRSTZ;



**************************************************
SUBROUTINE CALL:
```

```
    CurrSubNum = SUB_NUMBER + StartSubNum ;
    OUTPUT $ " M98 P" CurrSubNum  ;

BEGINNING OF SUB:
    SET_ON  MCH_FEED MCH_COOL LIN_MOV CIRC_MOV ;
    FlagSub = YES ;
    OUTPUT \J " " ;
    IF_SET (FlagSeq _EQ_ YES)
          OUTPUT \J "O" CurrSubNum ;
    ELSE
          OUTPUT \J " O" CurrSubNum ;
    END_IF ;

END OF SUB:
    OUTPUT $  " M99" ;
    FlagSub = NO ;

SUBROUTINE RETURN:
    SET_ON LIN_MOV CIRC_MOV X_CURPOS Y_CURPOS Z_CURPOS;

*******************************************************
```

(Cimatron)

Auteursrechtelijke overeenkomst

Ik/wij verlenen het wereldwijde auteursrecht voor de ingediende eindverhandeling:
Study and create the post-processor for CAD/CAM software for 5 axes CNC milling

Richting: **master in de industriële wetenschappen: energie-automatisering**
Jaar: **2014**

in alle mogelijke mediaformaten, - bestaande en in de toekomst te ontwikkelen - , aan de Universiteit Hasselt.

Niet tegenstaand deze toekenning van het auteursrecht aan de Universiteit Hasselt behoud ik als auteur het recht om de eindverhandeling, - in zijn geheel of gedeeltelijk -, vrij te reproduceren, (her)publiceren of distribueren zonder de toelating te moeten verkrijgen van de Universiteit Hasselt.

Ik bevestig dat de eindverhandeling mijn origineel werk is, en dat ik het recht heb om de rechten te verlenen die in deze overeenkomst worden beschreven. Ik verklaar tevens dat de eindverhandeling, naar mijn weten, het auteursrecht van anderen niet overtreedt.

Ik verklaar tevens dat ik voor het materiaal in de eindverhandeling dat beschermd wordt door het auteursrecht, de nodige toelatingen heb verkregen zodat ik deze ook aan de Universiteit Hasselt kan overdragen en dat dit duidelijk in de tekst en inhoud van de eindverhandeling werd genotificeerd.

Universiteit Hasselt zal mij als auteur(s) van de eindverhandeling identificeren en zal geen wijzigingen aanbrengen aan de eindverhandeling, uitgezonderd deze toegelaten door deze overeenkomst.

Voor akkoord,

Nullens, Bart

Datum: **12/06/2014**