The simulation of NC programs generated with the UnigraphicsII software

Melio, J.P.

Award date:
1992

The simulation of NC programs generated with the UnigraphicsII software.

J.P. Melio
WPA nr. 1355
June 1992

Eindhoven University of Technology
Faculty Mechanical Engineering
Division WPA

Professor: prof. dr. ir. A.C.H. van der Wolf

Coaches: ing. J.J.M. Schrauwen
F.G.J. Soers

Eindhoven, july 1992

TECHNISCHE UNIVERSITEIT EINDHOVEN
Faculteit Werktuigbouwkunde
Vakgroep WPA

Date : December 1991

Student : J.P. Melio

Prof : Prof. Dr. Ir. A.C.H. van der Wolf

Coaches : Ing. J.J.M. Schrauwen
F.G.J. Soers

Title : The simulation of NC programs generated with the UnigraphicsII software.

Subject :

Within the division WPA the CAD software UnigraphicsII of EDS International is used. This software contains a Manufacturing Module. With this module and a post-processor it is possible to create NC programs for a five axis milling machine, like the MAHO 700S. The manufacturing module provides the opportunity to show the tool paths on the screen. As a result a lot of lines appear on the screen, which makes it difficult to check the process. Therefore it is useful to have a simulation program which shows material removal, so the product is being processed on the screen.

A simulation program of the material removal does not show collisions. For this purpose an other simulation should be made which uses the software Robotics of EDS International.

Assignment:

Make a start for a simulation program which shows material removal by milling operations with an extension for the simulation of the movements of the milling machine.

Prof. Dr. Ir. A.C.H. van der Wolf

Ing. J.J.M. Schrauwen

F.G.J. Soers

J.P. Melio

Summary

Computer Aided Design (CAD) systems often contain a manufacturing module which is able to generate NC programs for the designed products. Instead of testing the NC program using the real milling machine, it is desirable to perform a computer simulation of the NC program. This way irregularities can be corrected before the NC program is executed on the milling machine.

Simulation programs for NC programs are being sold by several software firms. Instead of purchasing simulation software, the choice was made to try to develop a simulation program with the use of the software already available within the division WPA, part of the faculty Mechanical Engineering at the TUE. Besides a material removal simulation a motion simulation program is also made to monitor the motions of the milling machine. This way it can be checked whether the material is removed correctly and if any collisions occur during the manufacturing.

It appeared that the material removal simulation and the motion simulation could not be made with the use of one software environment. Therefore, the material removal simulation is made with the use of UnigraphicsII, which is an extensive CAD system. The Unigraphics tool User Function is used to program the simulation. The motion simulation program is being performed with Robotics. Robotics is designed to perform off-line robot programming, but in the simulation environment of Robotics, PLACE, it is possible to simulate a milling machine.

For the material removal simulation, an NC program is being translated into a User Function program. This program builds a tool path and subtracts this from the raw material. This assignment only deals with the simulation of vertical linear planar milling actions and a program concept for the translation from NC program to User Function program is made. This concept has to be transformed into a computer program.

Within Robotics a model of the milling machine has been brought into a PLACE simulation cell. In order to perform a milling simulation, an NC program has to be translated into a Robotics sequence. A program concept for the translation from NC program to Robotics sequence is made, but this concept has to be transformed into a computer program. At this moment G02 and G03 commands can not be translated.

The two simulation programs have been tested by performing a manual translation from the NC program. The material removal simulation gives a good reflection of the material being removed. With the motion simulation it is possible to check the NC program for collisions.

With this research assignment a start has been made for the simulation of NC programs but in the future the simulations have to be developed further.

Samenvatting

Computer Aided Design (CAD) systemen bezitten vaak een manufacturing module waarmee NC-programma's gegenereerd kunnen worden voor de ontworpen produkten. In plaats van NC-programma's te testen op de werkelijke freesbank, is het wenselijk om de programma's te simuleren met behulp van de computer. Op deze manier kunnen fouten gecorrigeerd worden voordat het programma uitgevoerd wordt op de freesbank.

Simulatieprogramma's voor NC-programma's worden door verschillende software bedrijven verkocht. In plaats van het aanschaffen van simulatie software, is de keuze gemaakt om deze zelf te ontwikkelen met de software die beschikbaar is binnen de vakgroep WPA, deel van de faculteit Werktuigbouwkunde aan de TUE. Naast een materiaalafname-simulatie is ook een bewegingsimulatie gemaakt, om de bewegingen van de assen van de freesbank te bekijken. Met deze simulaties kan bekeken worden of het materiaal correct wordt afgenomen en of er zich botsingen voor doen tijdens het bewerken van het produkt.

Het bleek dat de materiaalafname-simulatie en de bewegingssimulatie niet met één software pakket gemaakt kon worden. De materiaalafname-simulatie is gemaakt met UnigraphicsII, een krachtig CAD pakket. Het Unigraphics deelpakket User Function is gebruikt om de simulatie te programmeren. De bewegingssimulatie wordt uitgevoerd met behulp van Robotics. Robotics is ontworpen voor het off-line programmeren van robots, maar binnen de simulatie omgeving van Robotics, PLACE, is het ook mogelijk om een freesbank te simuleren.

Voor de materiaal afname simulatie wordt een NC-programma vertaald in een User Function programma. Dit programma modelleert een freesbaan en trekt deze van het uitgangsmateriaal af. Deze opdracht behandelt alleen de simulatie van het vlak frezen met verticale freeskop en een opzet voor een vertaal programma van NC-programma naar User Function programma is gemaakt. Deze opzet moet nog vertaald worden in een computer programma.

Binnen Robotics is een model van de freesbank in de PLACE-simulatie-cel gebracht. Om een bewegingsimulatie uit te voeren, moet een NC-programma vertaald worden in een Robotics-sequence. Een concept voor de vertaling van NC-programma naar Robotics sequence is gemaakt, maar dit concept moet nog vertaald worden in een computer programma. Op dit moment kunnen G02 en G03 commando's nog niet vertaald worden.

De simulatie-programma's zijn getest door het uitvoeren van handmatige vertalingen van het NC-programma. De materiaalafname-simulatie geeft een goed beeld van het materiaal dat wordt afgenomen. Met de bewegingssimulatie is het mogelijk om te controleren of er botsingen voorkomen in het NC-programma.

Met deze opdracht is een begin gemaakt voor de simulatie van NC-programma's, maar in de toekomst moeten de simulaties verder ontwikkeld worden.

Preface.

This report is the result of my graduation assignment, with which I conclude my study Mechanical Engineering at the Eindhoven University of Technology. The assignment was carried out at the TUE itself within the division WPA.

With this research assignment a start has been made for the simulation of NC programs. Nowadays for all kind of problems simulation programs are being made. The reason for this is often to save money by simulating instead of performing the situation for real. An important issue which has to be kept in mind is that the costs of a simulation program should not exceed the costs for the real situation.

I would like to thank the people who contributed to this report. In the first place my coaches: prof. dr. ir. A.C.H. van der Wolf, ing. J.J.M. Schrauwen and mr F.G.J. Soers. Also I would like to thank ing. H.W.A.M. van Rooij for his help with computer and software matters. Finally I want to thank all the people who made my graduation possible.

Hans Melio
Eindhoven, 6 july 1992

CONTENTS

The appendices are to be found in a seperate part with the same WPAnr.

Chapter 1. Introduction

Nowadays, more and more products are being designed with computer aided design (CAD) systems. Often, these systems contain a module which generates manufacturing programs for products which are designed with the CAD system. The manufacturing program is called a Numerical Control (NC) program, and this program is used to control a numerical controlled machine. This makes it easier to design and produce products with difficult shapes. The manufacturing programs of these products are difficult to understand, and therefore it is not possible to check them by just reading them.

When an NC program has been generated, it is always the question if the program is correct. That is why NC programs are tested using the milling machine. If there are errors, the NC program has to be adjusted. This testing of NC programs takes expensive machining time, and also contains the risk of collisions with possible damage. That is why simulation programs are created in order to check the NC programs.

At the TUE, faculty Mechanical Engineering, within the division WPA, the CAD system UnigraphicsII is available. With this system it is possible to design products with wireframe, surface and solid modelling. UnigraphicsII also contains a manufacturing module which can generate a Cutter Location Source (CLS) file. This file contains all the information necessary to manufacture the product. With a pre-processor a Cutter Location (CL) file is made and a post-processor is used to generate the actual NC file.

A simulation program for the NC programs, generated with Unigraphics, is not available within the division WPA. At present simulation programs by several software houses are being released. Unigraphics has developed the simulation program "Vericut" (Appendix A). Cimplex has released the simulation program "NCVerification". These two programs simulate the material removal with the use of shaded images.

The division WPA wished not only a material removal simulation, but also a simulation which shows the motions of the axes of the milling machine to see if there are any collisions. In the first place, the simulation will be used for NC programs generated with the Unigraphics manufacturing module. Secondly it also will be possible to simulate NC programs generated elsewhere. The choice was made to develop these simulation programs on the TUE, instead of purchasing. This way, insight is obtained in the way of developing a simulation program, with the use of the present software, instead of developing a new simulation environment. With this research assignment a start has been made for the development of these simulation programs. For the completion of the simulation programs, the program concepts have to be transformed into computer programs.

Chapter 2. Problem description.

2.1 Present situation.

UnigraphicsII is a powerful tool for the designer in designing products. It gives him the opportunity to design products with difficult shapes. A great deal of the products designed with UnigraphicsII, are milling products and have to be manufactured on a milling machine. With the manufacturing module it is possible to generate NC programs for the products which have been designed. Naturally, there are limitations in designing shapes, and it is possible that designs are made which can not be produced. So during his work the designer has to consider the manufacturing possibilities. The designer does not have to worry about how he is going to program his design in an NC program. The manufacturing module of UnigraphicsII will take care of that problem.

When the product design has been finished, the NC program can be made with the use of the manufacturing module. This module guides the user through the process of making an NC program. First, the operation type and the machine coordinate system is chosen. Then, a number of parameters have to be set for each surface or part that has to be machined. Some of these parameters are: the tool, the face which has to be machined, the avoidance geometry, display options, the feed, the engage and retract vector and the stock which has to be left. When all parameters have been given, the tool paths are shown, and a Cutter Location Source file is made. Now the next surface or part of the product can be "machined" by giving a new parameter list.

The Cutter Location Source file (CLS) contains the coordinates of the cutter location points, in values of the coordinate system chosen in Unigraphics. It consists of tool motion commands, control commands, feed rate commands, display commands, and postprocessor commands. In addition, macro commands may be written into the CLS file or retrieved from a library. The CLS file is converted into a Cutter Location file (CL), which is written in standard APT (Automatically Programmed Tools). This file CL contains the same information as the CLS file, except for the Unigraphics display commands. The CL file is still machine independent and can be translated to different milling machines. With a postprocessor the CL file is converted into an NC program for a specific milling machine. This NC file is ready to be used on the machine.

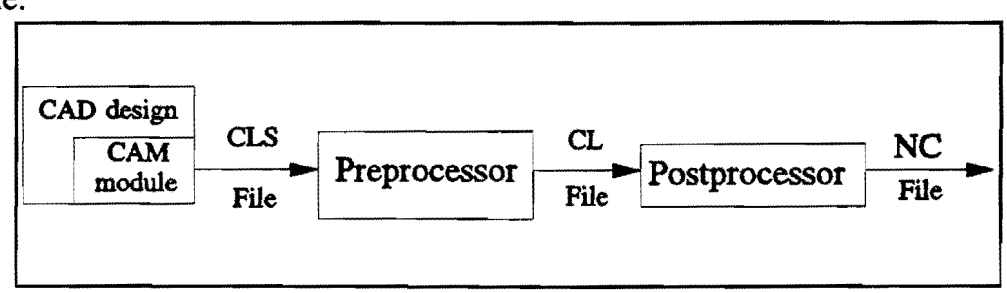


Figure 1: Realisation of the NC file.

When the manufacturing module creates the CLS file, the tool paths of the milling tool are shown. On the screen the designed product and a large number of coloured lines can be seen. These lines represent the tool paths. There is also the possibility to place a model of the milling tool in every cutter location point. These tool paths and the model of the milling tool stay on the screen until the whole product is manufactured. With complicated products this will result in a coloured plane of tool paths and tool models. This makes it hard to check whether there are irregularities during manufacturing.

First the question may rise whether it is likely that irregularities occur when the manufacturing module is used to make the CLS file. Errors that might occur are programming errors by the operator of the manufacturing module: a clearance plane which is set too low, the wrong milling tool is used, entrance vector is wrong and so on. An other class of errors are collisions of the Z-axis and the product table of the milling machine or a collision of the product and an axis of the machine or a collision of the tool with the fixtures. (See figure 2). These irregularities will not be shown with the present simulation of the manufacturing module. To improve this situation, a simulation program is desired.

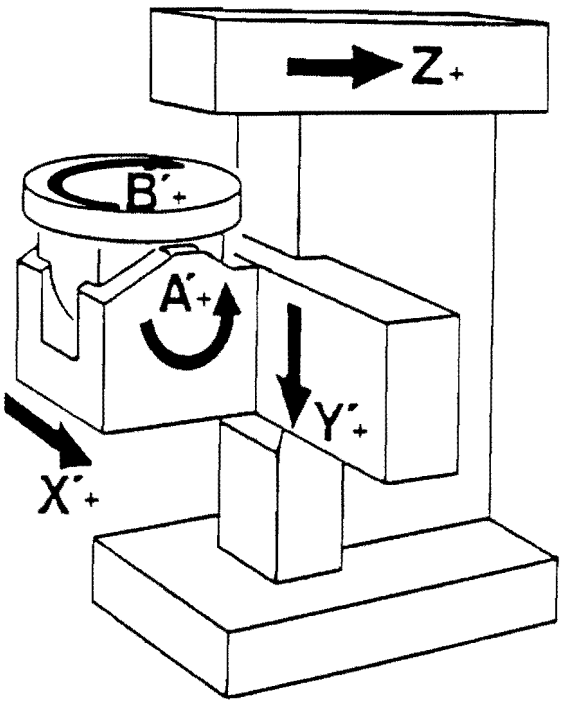


Figure 2: Representation of the MAHO 700S.

2.2 Purpose and demands.

The main purpose of this simulation program can be described as:

> to provide insight in the working of NC programs without having to test the program on the milling machine.

This rough description can be split up in a few demands. The simulation program has to:

1. show material removal to check if this is being done properly.
2. perform a collision detection between the milling tool, the fixtures, the product and the milling machine.
3. check if enough or too much material has been removed at the end of the manufacturing program, by comparing the result of the simulation with the designed product.

Some topics which can be filled in later on are:

4. calculate the total manufacturing time.
5. calculate the manufacturing time for each milling tool.

2.3 Programming environment.

Because the intention is to realize the goals with the use of relatively simple means instead of creating a whole new simulation environment, it has to be examined which software can be helpful making a simulation program. The first and the second demand imply two completely different issues. Material removal is a graphical manipulation of the raw material drawing. The collision detection demands for a motion simulation of the milling machine.

The software available is: UnigraphicsII and Robotics, both from EDS International. With UnigraphicsII it is possible to design the raw material and subtract material which is removed during the milling operation. It is not possible to create a motion simulation of the milling machine within UnigraphicsII. With the Robotics software, it is possible to create a model of the milling machine and to simulate its motions. Within this software a product can be loaded into the simulation environment and this product can be deleted. It is not possible to subtract material from this product.

This means that it is not possible to unite the first two demands in one software packet. The material removal can be simulated with UnigraphicsII and the motion simulation with Robotics. Therefore, the simulation program will be split into two parts:

1. material removal simulation using UnigraphicsII.
2. motion simulation of the milling machine using Robotics.

Chapter 3. The material removal simulation.

3.1 Goals and limitations.

The first question that has to be asked is, what the material removal simulation has to show. The answer to this question will contain a number of items. The main part of the simulation will show the material being removed. During this part of the simulation it can be checked if the material is removed correctly with the right milling tool. The simulation also should show if fixtures are placed at the right places. By using different colours, it will be possible to show what speed is used: rush speed or milling speed.

An NC program can contain a number of different machine instructions. Senden [12] divided the machine instructions for the MAHO 700S into two groups (appendix E). Because, it is too difficult to create a simulation program for all the possible milling operations in one time, some limitations have to be made. The material removal simulation will first deal with planar milling with straight paths and with the spindle head in vertical position. The A and B axes, of the milling machine, are left out of consideration at this moment. This way it is possible to see if the concept of the material removal simulation works satisfactory, with just three axes. Later on, the simulation can be extended with the A and B axes. The machine instructions which will be simulated are: G00, G01, M6 and T**. The two G commands are instructions for linear motion and the M6 command is the instruction for a tool change to tool T**.

3.2 The simulation environment.

To program the items mentioned above, suitable software and a source program have to be chosen. It will be obvious that UnigraphicsII will function as the programming environment for the material removal simulation. Within UnigraphicsII there are two sub-programs that can be used to perform programming with UnigraphicsII tools: GRIP and User Function.

GRIP stands for GRaphics Interactive Programming. By using a vocabulary of English-like words, GRIP can perform most of the operations in UnigraphicsII and its associated modules. In some cases GRIP can be used to perform advanced, customized operations, in a more efficient manner than using interactive UnigraphicsII. For example, operations which might take many interactive steps can be executed by a GRIP program, often without user intervention. Repetitive operations with some interactive actions, can also be performed more quickly with a GRIP program. A disadvantage is that it is not possible to program with parameters.

User Function (UFUNC) is a software tool designed to enable an easy interface between UnigraphicsII and the outside world. User Function is not intended to

replace GRIP, but rather to make it easier to interface to UnigraphicsII from a high-level language. There are many tasks in which the use of a high-level language program is more appropriate, such as operating systems interfaces, translation programs and user-written native language programs. Also it is possible to use parameterized solids with User Function. That is why User Function is chosen to be used for the programming of the material removal simulation.

The source program for the simulation can be the CLS file, the CL file or the actual NC file. The CLS file has the advantage that the structure is very close to the UnigraphicsII environment. The NC file on the other hand is the real manufacturing program. That is why the NC program is used as the source program.

3.3 User Function.

User Function consists of:

- a large set of user callable subroutines that access the Unigraphics Graphical Terminal, File Manager and Database.
- command procedures to link and run user programs.
- an interactive interface in Unigraphics to run those programs.

User Function programs can run in two different environments, depending on how the program was linked. The two environments are: internal and external.

Internal: these User Function programs can only be run from inside a Unigraphics session. These programs are loaded into the main memory along side Unigraphics, and access routines within Unigraphics. One advantage to this is that the executables are much smaller and link much faster than the external User Function programs. Once an internal User Function program is loaded into memory, it stays resident for the remainder of the Unigraphics session. If the program is called again, it executes without reloading. Internal User Function programs work on the current part and automatically modifies the part display. The Design Module and internal User Function programs can be used interchangeably to create and modify a current part.

External: these User Function programs are self-sufficient programs that can run from the operating system, outside of Unigraphics, or as a child process started from within Unigraphics. These programs set up their own environment. External User Function programs are not used during this project and thus will not be discussed further.

3.3.1 Writing User Function programs.

According to the User Function manual, it is possible to write User Function programs in any programming language as long as all calling conventions are followed. The most common language is Fortran, followed by C and Pascal. This project is carried out on an HP station with a UNIX operating system. The common programming language for UNIX systems is C. During this project it was attempted to write User Function programs with C, which did not succeed. According to E. Gerritsen [private communication] there are too many differences between C and Fortran (Also see appendix D). Unigraphics version 9.0 will provide in this problem. The User Function library of that version will be divided in three classes: routines that can be called by either Fortran or C, routines that can be called by Fortran and routines that can be called by C. The Unigraphics version 9.0 will be released about the middle of 1992. As a result of this Fortran is being used to write the User Function programs.

Before User Function can be used a number of environment variables has to be set. These variables define the places of certain files so the computer system can locate them. Also a User Function menu has to be made. This is explained in appendix B. In appendix C information can be found about compiling and linking User Function programs.

Internal User Function programs can be considered as a user-written subroutine to Unigraphics. To start the execution of the program, Unigraphics loads the program into memory and then calls the routine UFUSR. The internal User Function program takes over execution at that point. At the end of the execution, the program executes a RETURN statement which returns control to Unigraphics. Because of this special interface, the startup and the termination of an internal User Function are slightly different from normal. The format needed is:

```
SUBROUTINE UFUSR(EXITNM, PARAM, RETCOD)
INTEGER*2          EXITNM
INTEGER*2          RETCOD
CHARACTER*132      PARAM

The main program is placed here.

RETURN
END
```

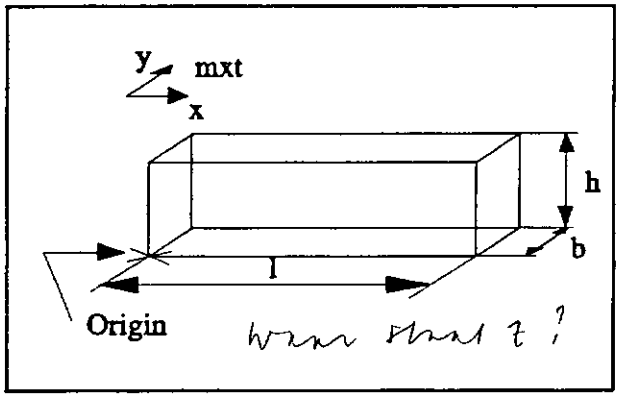
3.4 Simulation concept.

The intention of the simulation is to show the material removal. To accomplish this, a drawing of the raw material is placed on the screen. Then the material which is removed during one sentence of the NC program is subtracted from the drawing of the raw material. The result is a product on which one milling path is performed. For this operation it is necessary that the raw material and the drawing of the material

which is removed are modelled in solids. It is not possible to subtract a wire frame from an other wire frame. The raw material will be called the target solid, because from that solid the milling paths are subtracted. The model of the milling path will be called the tool solid.

The first item which has to be defined, at the start of the program, is the target solid. When this is a rectangular piece of material it is possible to program this in the User Function program. Often the raw material has a different shape. In that case the raw material has to be designed in UnigraphicsII, in solids. The advantage of this last possibility is that the fixtures can also be designed and connected to the raw material. After that the whole design has to be grouped to one entity, because there can only be one target solid. At the beginning of the simulation program the raw material and the fixtures have to be declared as the target solid. During the simulation it can also be monitored if the fixtures are in the way of the milling tool.

The information which is to be found in the NC program, is the coordinates of the cutter location points. This information is used to remove material from the raw material. The material between two cutter location points will be removed in one time. To accomplish this, a tool solid will be placed from the first cutter location point to the next cutter location point, after which the tool solid will be subtracted from the raw material. Again a tool solid will be positioned from the second cutter location point to the third cutter location point, and is subtracted from the raw material. This will go on until the last cutter location point has been reached. The tool solid, which will be needed for the subtract operation, is built in the User Function program with three solids: a solid block and two solid cylinders. The User Functions that are used are: UF6500: create solid block
UF6502: create solid cylinder



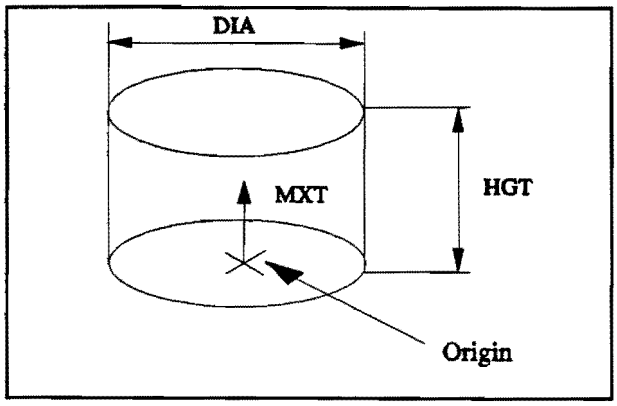
parameters for UF6500, solid block:

CRNER0(1) = x
CRNER0(2) = y
CRNER0(3) = z
SIZE0(1) = l
SIZE0(2) = b
SIZE0(3) = h
MXT0(1)
MXT0(2) vector x direction
MXT0(3) of the solid
MXT0(4)
MXT0(5) vector y direction
MXT0(6) of the solid

Figure 3. A solid block with its parameters.

In figure three the solid block is shown which forms the middle part of the tool solid. To create this block the parameters, beside the figure, have to be filled with the

dimensions named in the figure. The origin of the block is represented by the three values CRNER0, and is dependent on the first cutter location point. Of course it is also possible to give the origin a different name. The size of the block is defined with the parameters SIZE0. The width and the height are dependent of the cutting tool being used, and the length depends of the distance between the two cutter location points. The length of the block is in the x-direction, the width is in the y-direction and the height is in the z-direction of the solid. The orientation of the solid with respect to the absolute coordinate system is defined with the six matrix values MXT0.



parameters for UF6502, solid cylinder:

CORI1(1) = x
CORI1(2) = y
CORI1(3) = z
HGT1 = h
DIA1 = D
AXS1(1) vector
AXS1(2) cylinder
AXS1(3) axis

Figure 4. A solid cylinder with its parameters

Figure four shows the solid cylinder. Such a cylinder has to be placed on each end of the solid block. The parameters to form a solid cylinder are listed beside the figure. The origin of the solid cylinder, which is the arc center, is defined with the three parameters CORI1, and depends on the Cutter Location point. The cutting tool that is being used defines the height (HGT1) and the diameter (DIA1) of the cylinder. The cylinder axis is defined with AXS1. When the three solids are combined the complete solid is formed, which is shown in figure 5.

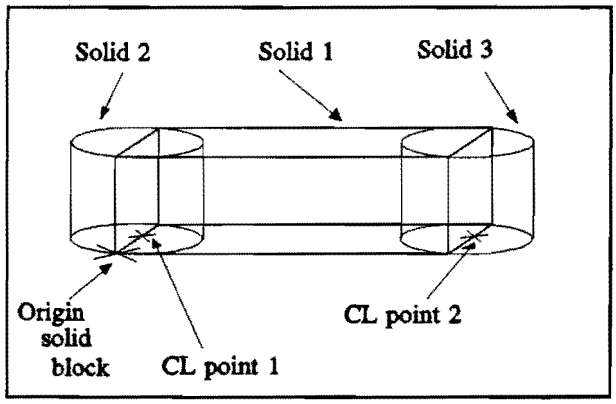


Figure 5. A complete tool solid.

An important issue is the location of the raw material with respect to the absolute coordinate system. The raw material should be placed in such a way that the designed product has exactly the same location as the product had during the manufacturing

module. This way there is no misunderstanding of locations during the milling operation.

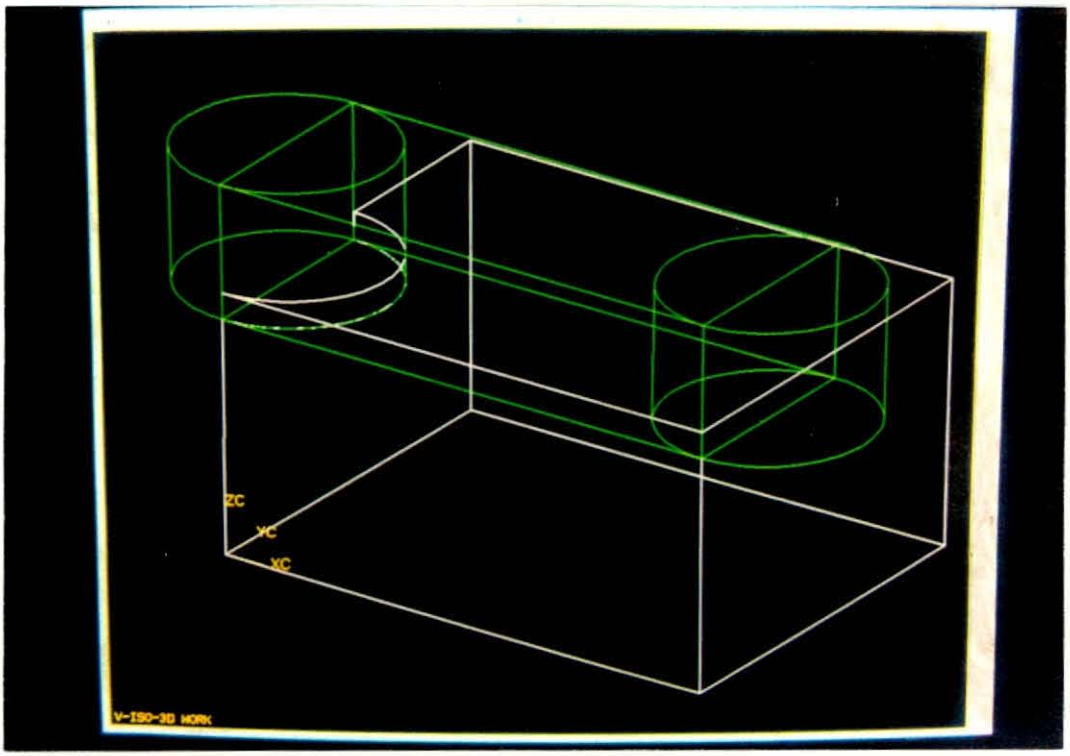


Figure 6: *The target solid and the tool solid in Unigraphics.*

The information to place the tool solid at the right location has to be obtained from the NC program. The parameters of the solids, which form the tool solid, have to be filled with the information from the first and the second Cutter Location point with their corresponding values: $CL1 = [x_1, y_1, z_1]$

$CL2 = [x_2, y_2, z_2]$

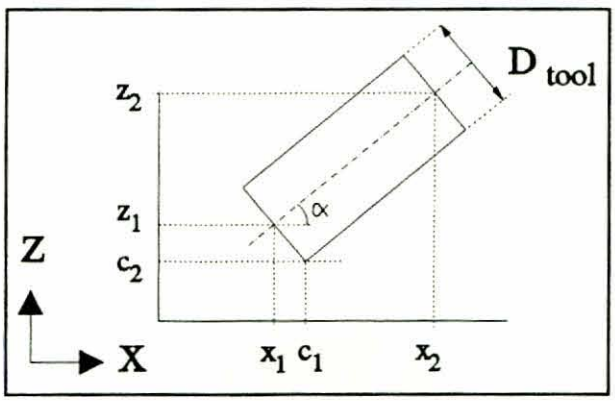


Figure 7: *Orientation of a tool solid in respect to the machine zero point.*

With some calculations it is possible to fill the parameters with values from the Cutter Location points. N.B. the coordinate system of the milling machine and the coordinate system of Unigraphics are not the same.

$$CRNER1(1) = x_1 + \frac{1}{2} D_{tool} \frac{z_2 - z_1}{\sqrt{(x_2 - x_1)^2 + (z_2 - z_1)^2}} \quad (1)$$

$$CRNER1(2) = y_1 - \frac{1}{2} D_{tool} \frac{x_2 - x_1}{\sqrt{(x_2 - x_1)^2 + (z_2 - z_1)^2}} \quad (2)$$

$$CRNER1(3) = y_1 \quad (3)$$

$$SIZE1(1) = \sqrt{(x_2 - x_1)^2 + (z_2 - z_1)^2} + 0.0001 \quad (4)$$

$$SIZE1(2) = D_{tool} \quad (5)$$

$$SIZE1(3) = h_{tool} \quad (6)$$

$$MXT1(1) = \cos\alpha = \frac{(x_2 - x_1)}{\sqrt{(x_2 - x_1)^2 + (z_2 - z_1)^2}} \quad (7)$$

$$MXT1(2) = \sin\alpha = \frac{(z_2 - z_1)}{\sqrt{(x_2 - x_1)^2 + (z_2 - z_1)^2}} \quad (8)$$

$$MXT1(3) = 0 \quad (9)$$

$$MXT1(4) = \sin\alpha = \frac{(z_2 - z_1)}{\sqrt{(x_2 - x_1)^2 + (z_2 - z_1)^2}} \quad (10)$$

$$MXT1(5) = \cos\alpha = \frac{(x_2 - x_1)}{\sqrt{(x_2 - x_1)^2 + (z_2 - z_1)^2}} \quad (11)$$

$$MXT1(3) = 0 \quad (12)$$

The parameters for the solid cylinder:

CORI1(1) = x_1	CORI2(1) = x_2
CORI1(2) = z_1	CORI2(2) = z_2
CORI1(3) = y_1	CORI2(3) = y_2
HGT1 = h_{tool}	HGT2 = h_{tool}
DIA1 = d_{tool}	DIA2 = d_{tool}
AXS1(1) = 0	AXS2(1) = 0
AXS1(2) = 0	AXS2(2) = 0
AXS1(3) = 1	AXS2(3) = 1

When the target solid and the tool solid are created, the tool solid can be subtracted from the target solid which is the raw material. For this operation User Function UF 6520 is used. The parameters for the subtract operation are:

OPTYPE1 = operation type, 2 for subtract
TARGS1 = target solid
TOOLS(1) = block
TOOLS(2) = cyl1 list of tool solids
TOOLS(3) = cyl2
TLNUM = number of tools in the tool list
MAXRTN = number of solids to be returned

The operation type for this User Function can be either 0 for uniting solids, 1 for intersecting solids or 2 for subtracting solids. The target solid is the raw material. In the list TOOLS the solids of the complete tool solid are listed. Now a module for the subtraction of one tool path can be written.

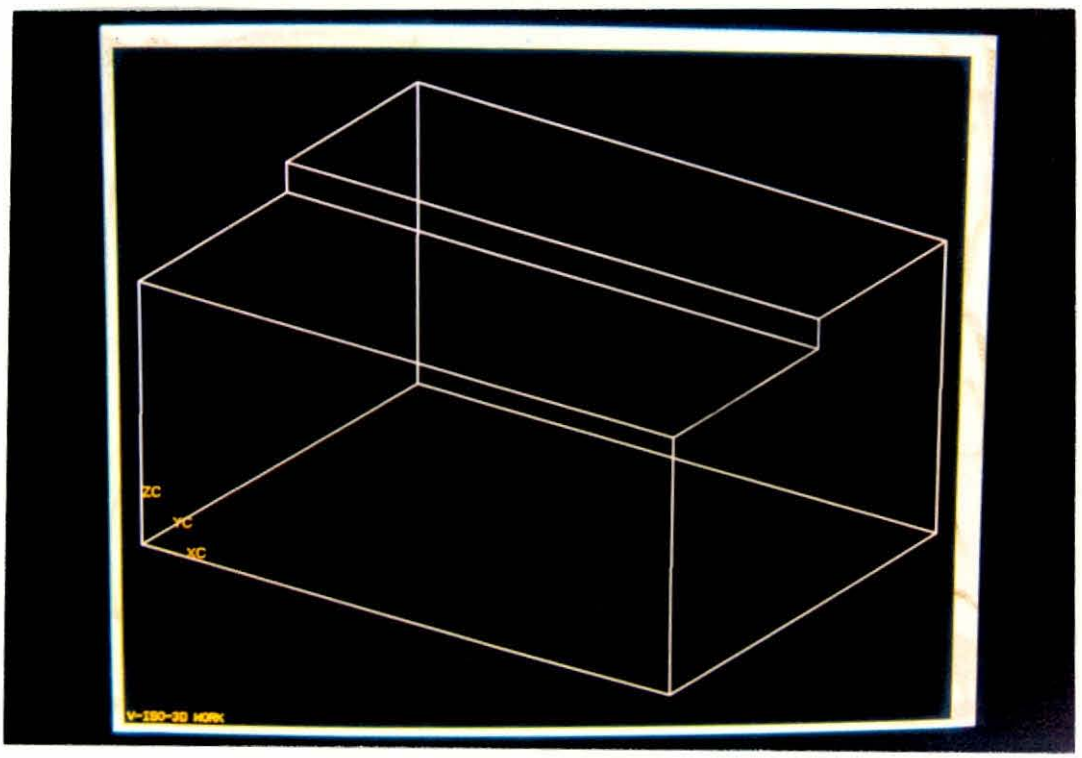


Figure 8: A target solid with a tool solid subtracted from it.

3.5 Program concept for the material removal simulation.

To guarantee that the translation from NC program to material removal simulation is being executed correctly, a translation program has to be written. This translation program has to get the information which is needed for the building of the tool solid from the NC file. In order to do this, every line of the NC file has to be checked for the information which is needed. The following program concept will describe the translation of milling actions (G00 and G01) and tool changes (M6 and tool T**), with the spindle head in the vertical position.

When the program has been started, the user is asked which NC program has to be translated. The user gives in the name of the NC program and it is checked whether the NC program is available in the current directory. If the NC program is not available, a message will appear on the screen and the translation program will be aborted. In the other case, the header will be written to the NCname.f file. The NCname.f file will eventually form the material removal simulation program. The header contains the title and all the declaration of the variables which will be used in the fortran program.

The header for the translation program:

```
C Definition of the sub routine.
      SUBROUTINE UFUSR(EXITNM, PARAM, RETCOD)
      INTEGER*2           EXITNM
      CHARACTER*132       PARAM
      INTEGER*2           RETCOD

C Declearation of the variables that are used.
      CHARACTER   TITLE
      INTEGER*2   MAX
      INTEGER*4   BLK0EID
      INTEGER*2   NUMEID
      REAL*8      POINT
      INTEGER*2   ROOP
      REAL*8      CRNER1(3)
      REAL*8      SIZE1(3)
      REAL*8      MXT1(6)
      INTEGER*4   BLK1EID
      REAL*8      CORI1(3)
      REAL*8      HGT1
      REAL*8      DIA1
      REAL*8      AXS1(3)
      INTEGER*4   C1EID
      REAL*8      CORI2(3)
      REAL*8      HGT2
      REAL*8      DIA2
      REAL*8      AXS2(3)
      INTEGER*4   C2EID
      INTEGER*2   OPTYPE1
      INTEGER*4   TARGS1
      INTEGER*4   TOOLS(3)
      INTEGER*2   TLNUM
```

```
      INTEGER*2       MAXRTN
      INTEGER*2       SOER9R
      INTEGER*2       RESNUM
      INTEGER*4       RESLST(1)
      CHARACTER*(80)          TEXT
      INTEGER*2       WOPT
      INTEGER*2       COLOUR

C waiting text and waiting option.
      TEXT = 'press Entry Complete to continue.'
      WOPT = 1

C definition of the raw material as the target solid by the user.
      TITLE = 'pick the target solid with the mouse.'
      MAX = 1
      UF1617(TITLE,MAX,BLK0EID,EIDRTN,POINT,ROOP)
```

When the raw material has been modelled, it will have to be defined as the target solid. So, the user will be prompted to define the raw material by clicking on it's model on the screen, with the mouse. This is defined at the end of the header.

From this point the translation begins and a message will be shown on the screen. The program "reads" the first line of the NC program. The program scans the NC line for G00, G01, G02, G03 or T**. If a T** is found, a check is made if there is a M6 command to see if the tool really is being changed. In the case of a tool change, T** is defined to tool1 and the dimensions of the tool are being retrieved from the tool library. Then the next line of the NC program is read, and the characters are being compared again. If the characters match G02 or G03, a message will appear on the screen which tells the user that these commands can not be translated right now, and the translation program will be aborted. If the characters match G00, colour1 is defined red, and if the characters match G01, colour1 is defined green. In the case of a G00 or G01 command, the line is being scanned for X, Y, Z, A or B values and these values are being attached to x_1, y_1, z_1, a_1 or b_1. Now the next line of the NC program is read.

The new line of the NC program is being checked if it is the end of the NC program. If this is the case the footer is written to the NCname.f file, the file is saved and the translation program is ended with the message: Translation Complete.

The footer of the translation program:

```
      RETURN
      END
```

If the new line is not the end of the NC program, the same checks and actions are being performed as in the last paragraph, but now the values are defined to: tool2, colour2, x_2, y_2, z_2, a_2 and b_2. When the end of line character has been reached, the program module G00/G01 has to be written to the NCname.f file.

C parameters for the right cylinder of the tool solid.

CORI1(1) = x_1
CORI1(2) = z_1
CORI1(3) = y_1
HGT1 = h_{tool1}
DIA1 = d_{tool1}
AXS1(1) = 0
AXS1(2) = 0
AXS1(3) = 1

C parameters for the block of the tool solid.

CRNER1(1) = formula 1
CRNER1(2) = formula 2
CRNER1(3) = formula 3
SIZE1(1) = formula 4
SIZE1(2) = formula 5
SIZE1(3) = formula 6
MXT1(1) = formula 7
MXT1(2) = formula 8
MXT1(3) = formula 9
MXT1(4) = formula 10
MXT1(5) = formula 11
MXT1(6) = formula 12

C parameters for the right cylinder of the tool solid.

CORI1(1) = x_2
CORI1(2) = z_2
CORI1(3) = y_2
HGT1 = h_{tool1}
DIA1 = d_{tool1}
AXS1(1) = 0
AXS1(2) = 0
AXS1(3) = 1

C parameter for the colour of the entity.

COLOUR = colour1

C creation of the tool solid.

```
CALL UF6502(CORI1,HGT1,DIA1,AXS1,C1EID)
CALL UF5016(C1EID,COLOUR)
CALL UF6500(CRNER1,SIZE1,MXT1,BLK1EID)
CALL UF5016(BLK1EID,COLOUR)
CALL UF6502(CORI2,HGT2,DIA2,AXS2,C2EID)
CALL UF5016(C2EID,COLOUR)
CALL UF1601(TEXT,WOPT)
```

C parameters for the subtract operation.

OPTYPE = 2
TARGS1 = BLK0EID
TOOLS(1) = C1EID
TOOLS(2) = BLK1EID
TOOLS(3) = C2EID
TLNUM = 3
MAXRTN = 1

C subtract operation.

```
CALL UF6520(OPTYPE,TARGS1,TOOLS,TLNUM,MAXRTN,SOERR,RESNUM,RESLST)
CALL UF1601(TEXT,WOPT)
```

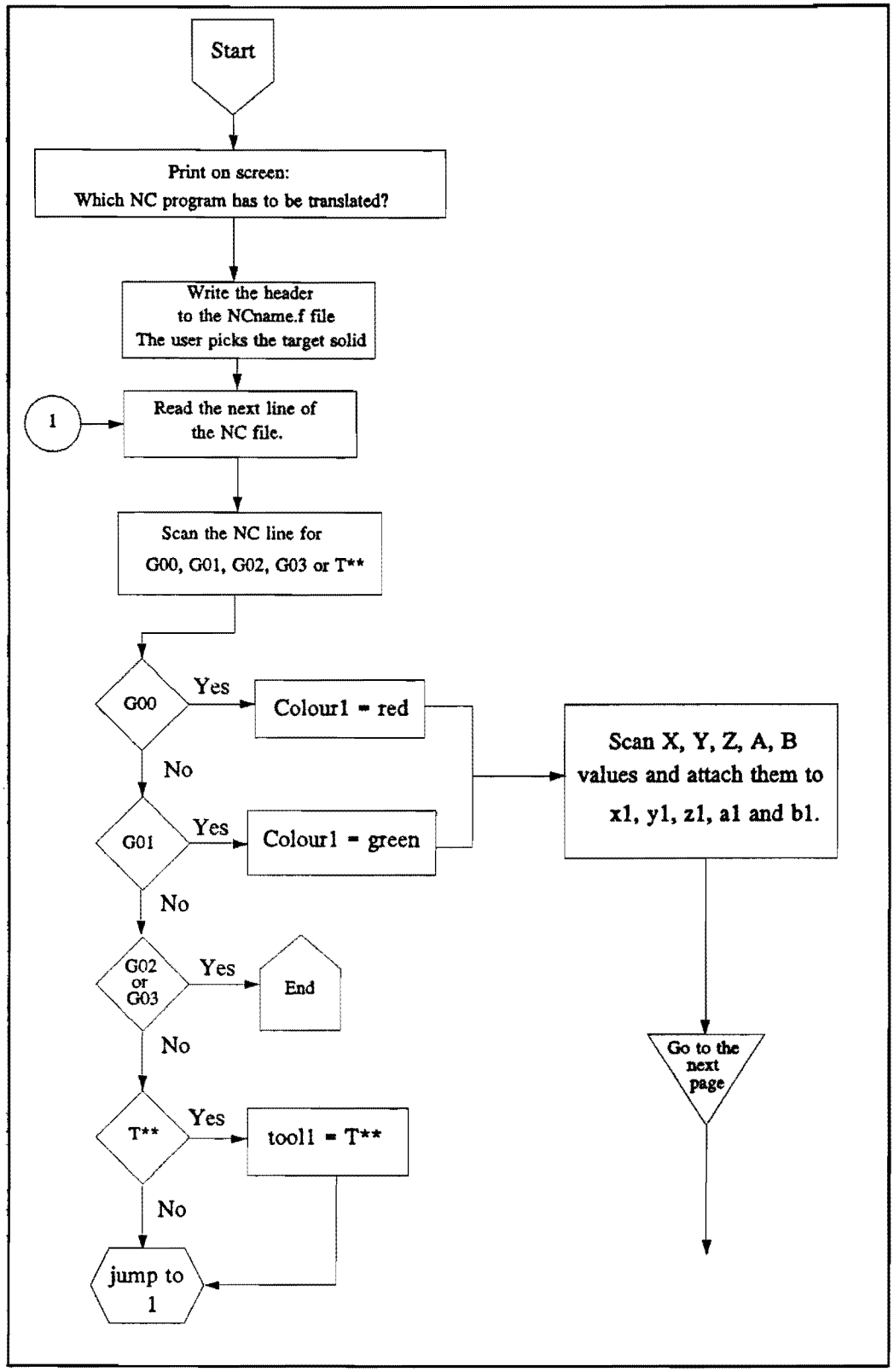
When the module G00/G01 has been written to the NCname.f file, the parameters1 get the values of parameters2. Then the next line of the NC program is read and the parameters2 get new values. This continues until the complete NC program has been scanned.

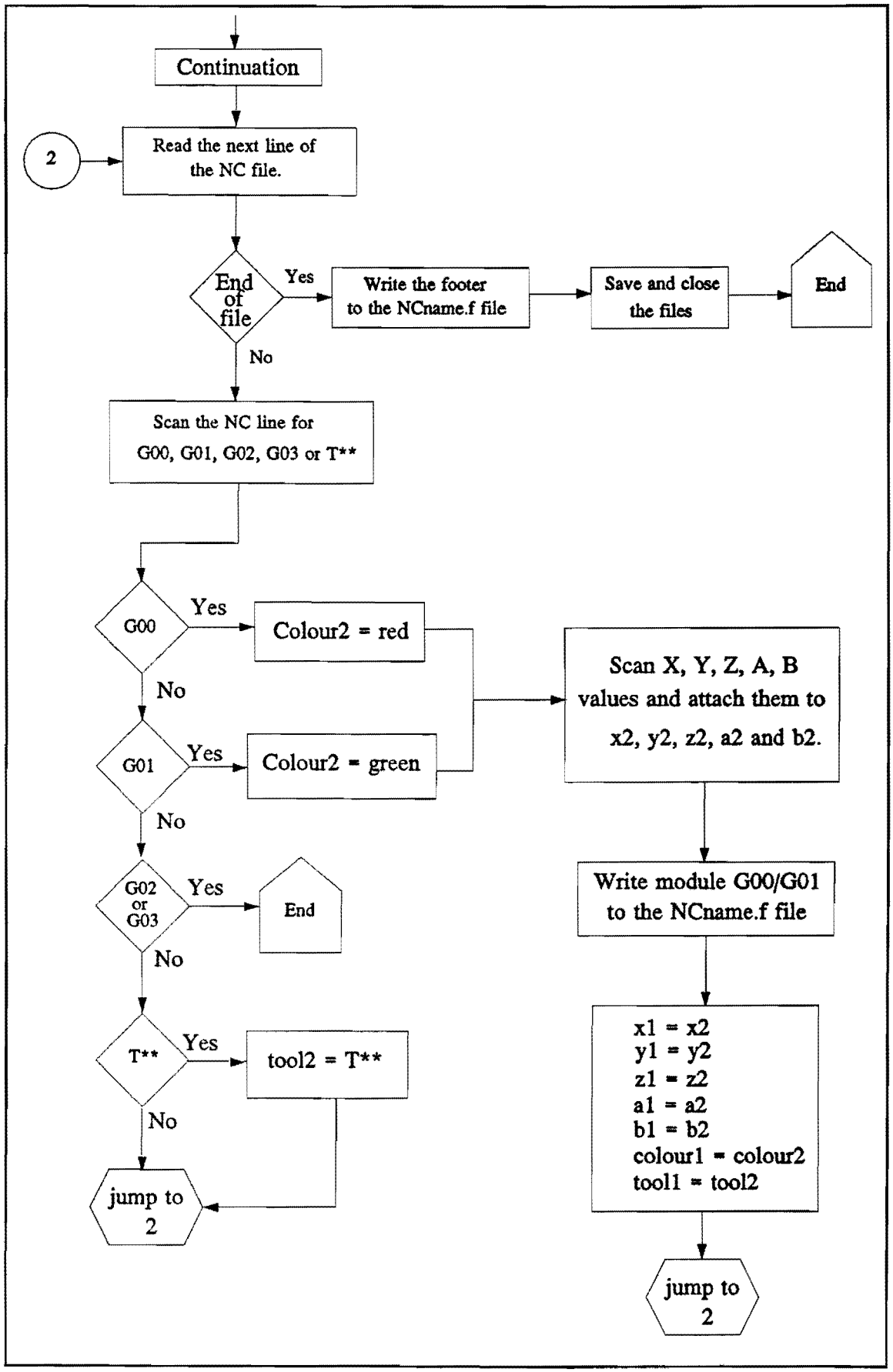
When the module for the G02 and G03 commands will be active, the parameters module1 and module2 have to be defined, because the command of the first NC line has to be carried out. When the command in the first line has been carried out, module1 gets the value of module2, and module2 gets a new value.

With a few changes the module G00/G01, with the cutting tool in the vertical position, can be changed in a module G00/G01 for milling with a horizontal cutting tool. The matrices with the direction vectors have to be changed and the y and z values have to be exchanged.

The third demand mentioned in paragraph 2.2 has not been taken up in this program concept. To perform this check, the target solid, after the simulation program, should be subtracted from the designed product, to check if too much material is removed. When the designed product is subtracted from the target solid, it can be checked if enough material is removed.

3.6 Flowchart of the translation program.




Continuation
2
Read the next line of the NC file.
End of file
Yes
Write the footer to the NCname.f file
Save and close the files
End
No
Scan the NC line for G00, G01, G02, G03 or T**
G00
Yes
Colour2 = red
No
G01
Yes
Colour2 = green
No
G02 or G03
Yes
End
No
T**
Yes
tool2 = T**
No
jump to 2
Scan X, Y, Z, A, B values and attach them to x2, y2, z2, a2 and b2.
Write module G00/G01 to the NCname.f file
x1 = x2
y1 = y2
z1 = z2
a1 = a2
b1 = b2
colour1 = colour2
tool1 = tool2
jump to 2

Chapter 4. The motion simulation.

4.1 The milling machine.

The milling machine that is to be simulated is the universal machining center MAHO 700S. This machine is equipped with automatic tool change and automatic changing of the spindle head, from vertical to horizontal milling. The system is supplied with a five axis path control: the Philips' CNC 432. For more information see appendix I.

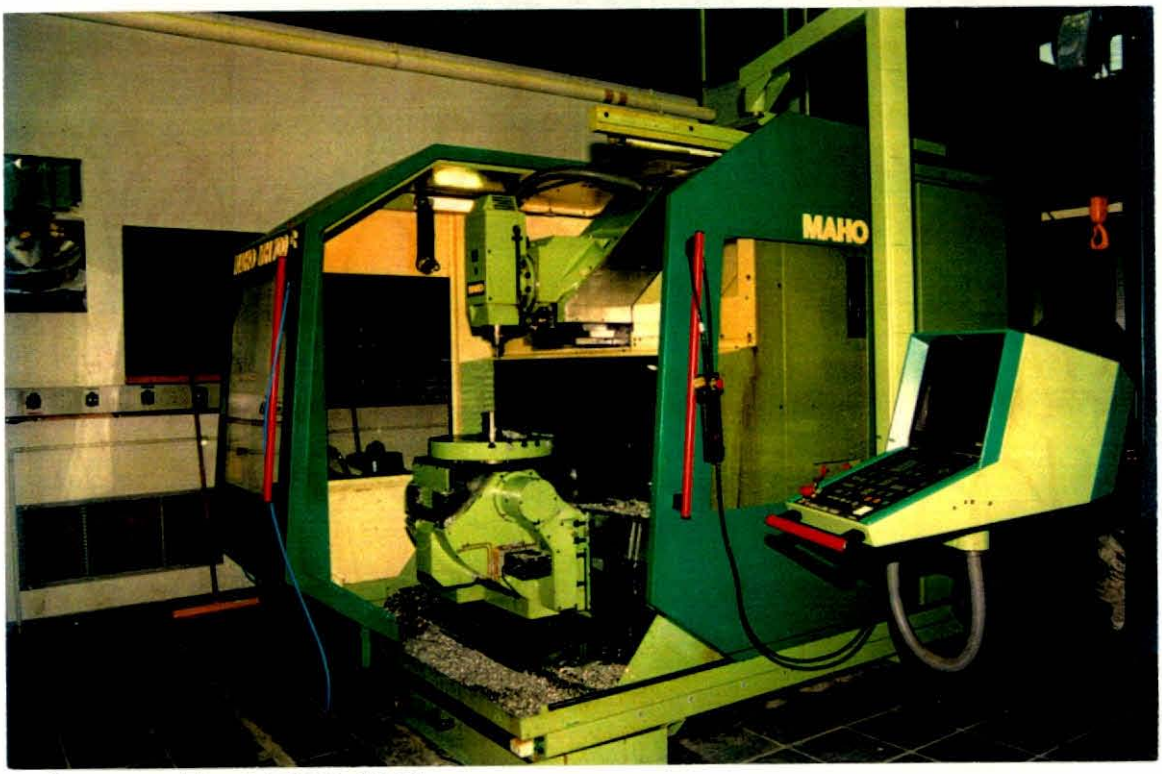


Figure 9: The MAHO 700S.

To control the machine, an NC program is needed which contains all the information to transform raw material into the desired product. The control unit reads the NC program and directs the different axes to the desired value. The NC program consists of sentences. Each sentence contains command letters and numerical information. An example of an NC sentence is:

N10 G01 X125 Y25 Z65 A-1.7 B22.5 F100 S1000 M3

sentence number
path condition
path information
feed rate 100 mm/min
number of revolutions 1000 rpm
help function: spindle turns clockwise

The MAH7O 700S is a machine with five axes: three translation axes and two rotation axes. It has a rectangular right-handed coordinate system. The tool performs a relative motion with respect to the milling table in Z direction. All motions in X-, Y-, A- and B- direction are performed by the milling table with respect to the tool. The motion directions have to be declared unambiguous. It has been defined that the tool always moves with respect to the product. The following picture shows the coordinate system of the MAHO 700S.

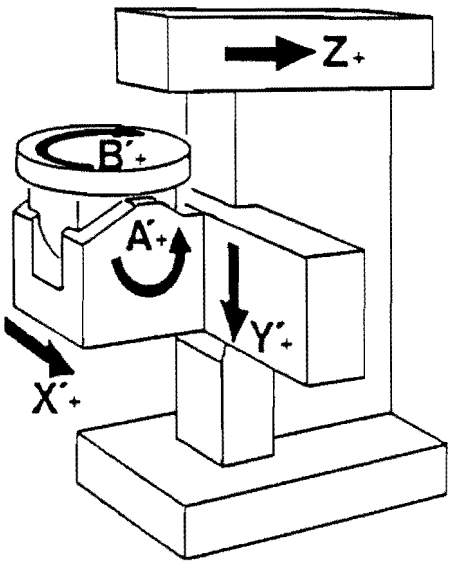


Figure 10. *The coordinate system of the MAHO 700S.*

4.2 The simulation environment.

Starting-point of the motion simulation of the MAHO 700S, is based on the results of Vernooij [13]. For the motion simulation the Robotics software is used.

The Robotics software of EDS International is designed to perform off-line robot programming. Robotics consists of 5 modules: BUILD, PLACE, COMMAND, CTA and ADJUST. BUILD is used to compose a robot cell, building it link by link. The simulation environment is presented in PLACE. Here the robot programs can be made and simulated on the screen. In PLACE programs are made for one specific robot, which is made with BUILD. But this program is independent from the robot controller. In other words: it is not written in native robot language. COMMAND translates the independent program into a robot dependent program. With CTA (= Cycle Time Analyzer) PLACE can learn the actual robot speeds and accelerations. The last module, ADJUST, is used to calibrate the PLACE cell. If a cell has been created in PLACE, ADJUST can modify the software cell in order to match the real world cell. [11] Further information can be found in appendix J.

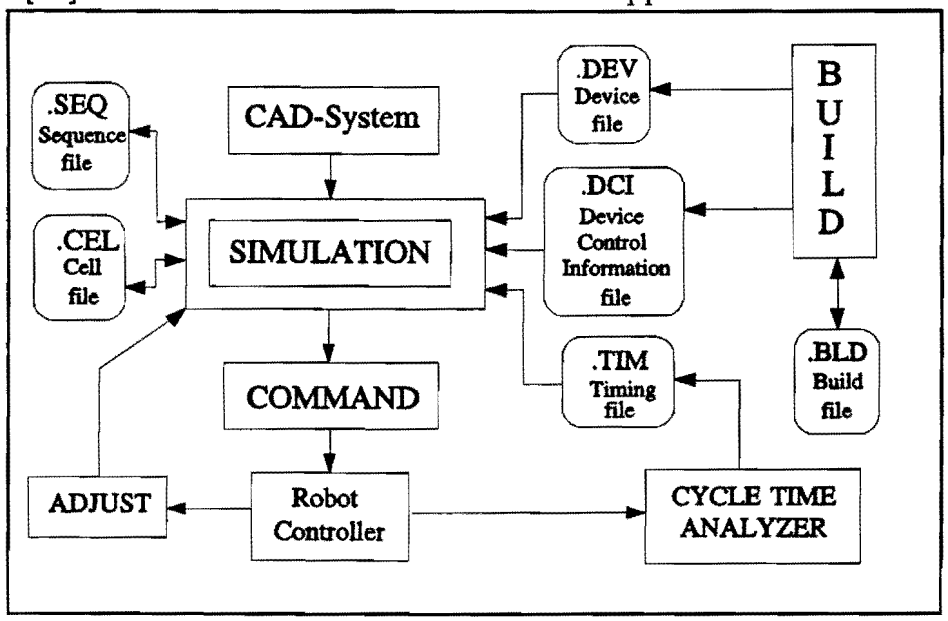


Figure 11: The structure of the Robotics software.

On the TUE, within the division WPA, Robotics is used to simulate the movements of a milling machine. For the motion simulation only BUILD and PLACE are used. COMMAND is not used because the NC programs are generated by the manufacturing module of UnigraphicsII. Because Robotics is only used for the simulation of the milling machine, CTA and ADJUST will not be needed.

Vernooij designed the work cell, containing the model of the milling machine, with the Y axis pointing upwards. A misunderstanding is that the coordinate system in which the work cell is build has anything to do with the directions of the motion of the work cell. However, when the coordinate system of the work cell is being rotated, this results in problems with the display functions. When the "display adjust" dials are

used, the devices in the workcell move not according to what should be expected and some sides of the device can not be rotated in front. That is why the whole work cell has been rebuild with the use of the original drawings of the links. This means that the dimensions of the model have not been compared with the real milling machine.

The MAHO 700S is a five axes machine, with three translation axes and two rotation axes. The Z-axis moves independently of the other four axes. In Robotics such a kind of independent working axes are modelled seperately as two different devices. One device will only contain the Z-axis, the other device will contain the other four axes.

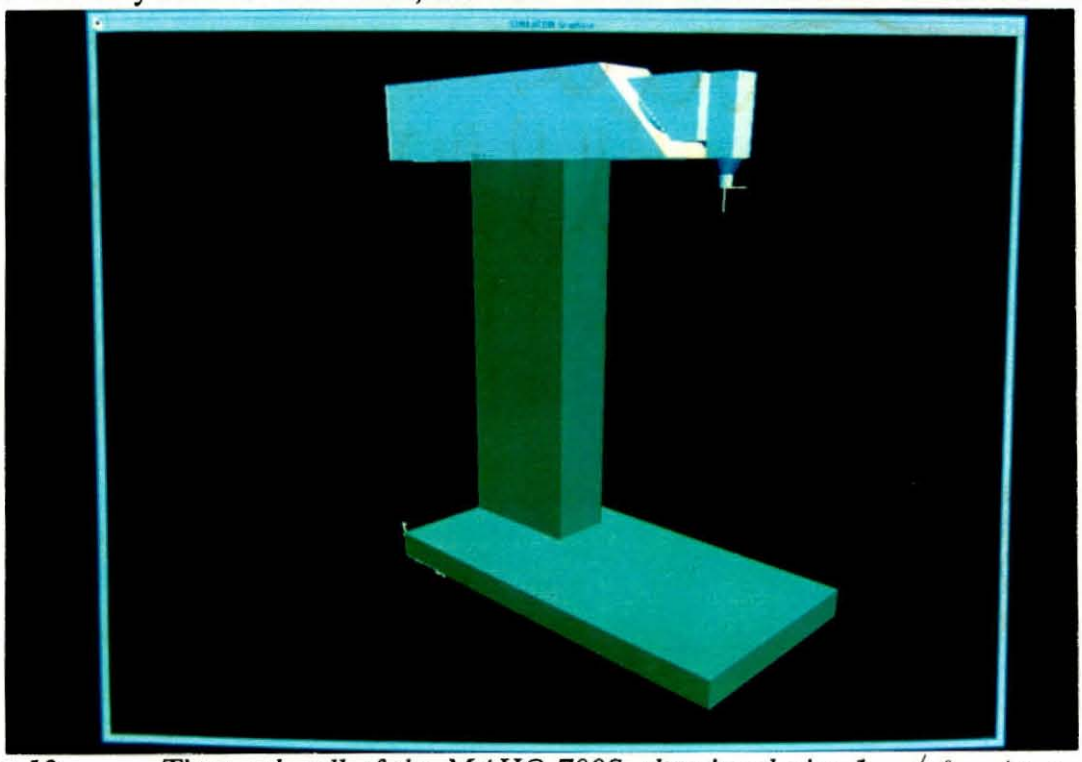


Figure 12. The work cell of the MAHO 700S, showing device 1.

0	World	
1	Freesb0	Device 1
2	Freesb1	
1	Freestaf0	
2	Freestaf1	
3	Freestaf2	
4	Freestaf3	Device 2
5	Freestaf4	
6	Leeg	

Figure 13: The connection tree of the work cell FRB1.

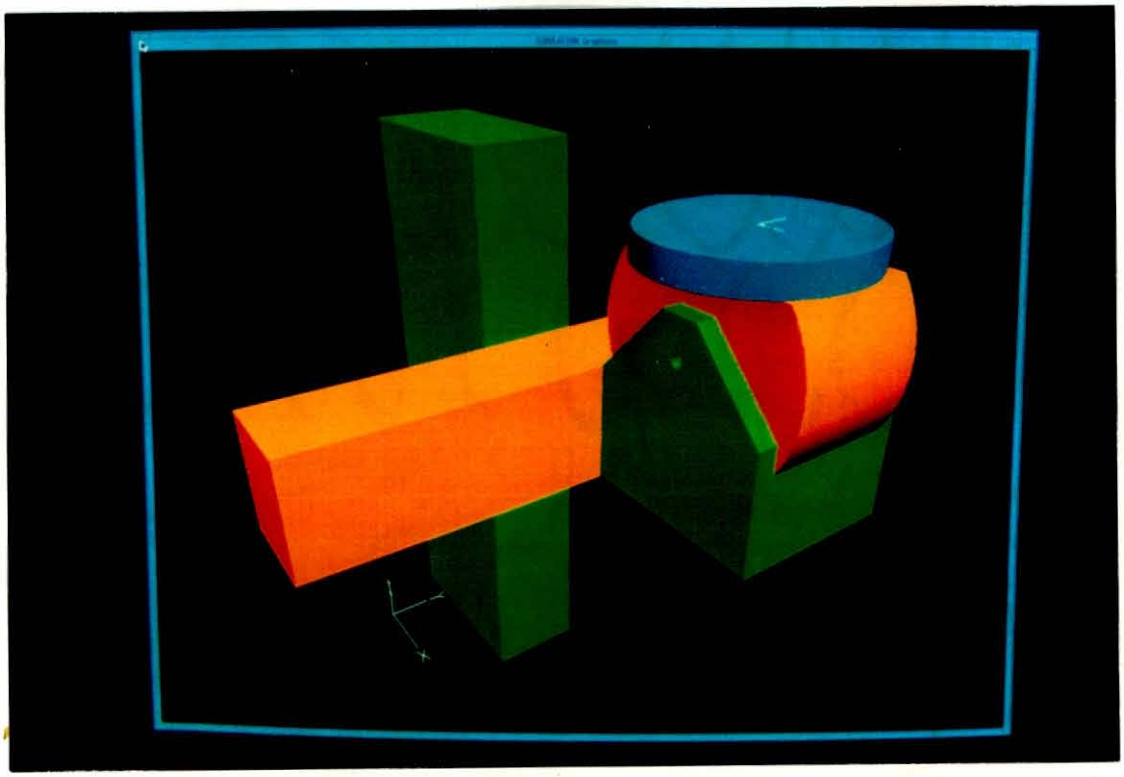


Figure 14: The work cell of the MAHO 700S, showing device 2.

The machine zero point has also been changed during the rebuilding of the milling machine model. The machine zero point of the work cell is in the middle of the turning table: axis B. It is not possible to perform a zero point movement in a work cell with Robotics. This requires that the product zero point always has to be placed in the middle of the turning table. If the product zero point has to be placed on a different position, it requires an adaption of the devices. When the product zero point is moved in the X- and Y-direction, only device2 has to be adapted. However when the product zero point is also moved in the Z- direction, all the devices with the Z-axis and cutting tools have to be adapted. An other possibility is to process a translation from the middle of the turning table to the product zero point in the program which translates the NC file into the Robotics sequence.

When there is a tool change, some adjustments have to be made because the new tool may have another stick-out length. In the Robotics work cell, the tool is part of the Z-axis so that the working tpoint is in the tooltip. (The working tpoint is the point from which Robotics calculates the axis values). So, when a tool change is required, the device with the groundframe and the Z-axis has to be deleted and an other device with the right tool has to be loaded into the work cell. The same problem occurs when the spindle head is rotated for horizontal milling. This also requires a new device, now with a horizontal tool. As a result of this, a device has to be made for every tool available . At this moment there are three tools available for the simulation. Appendix K contains more information about the cell FRB1, and appendix L contains the BUILD files.

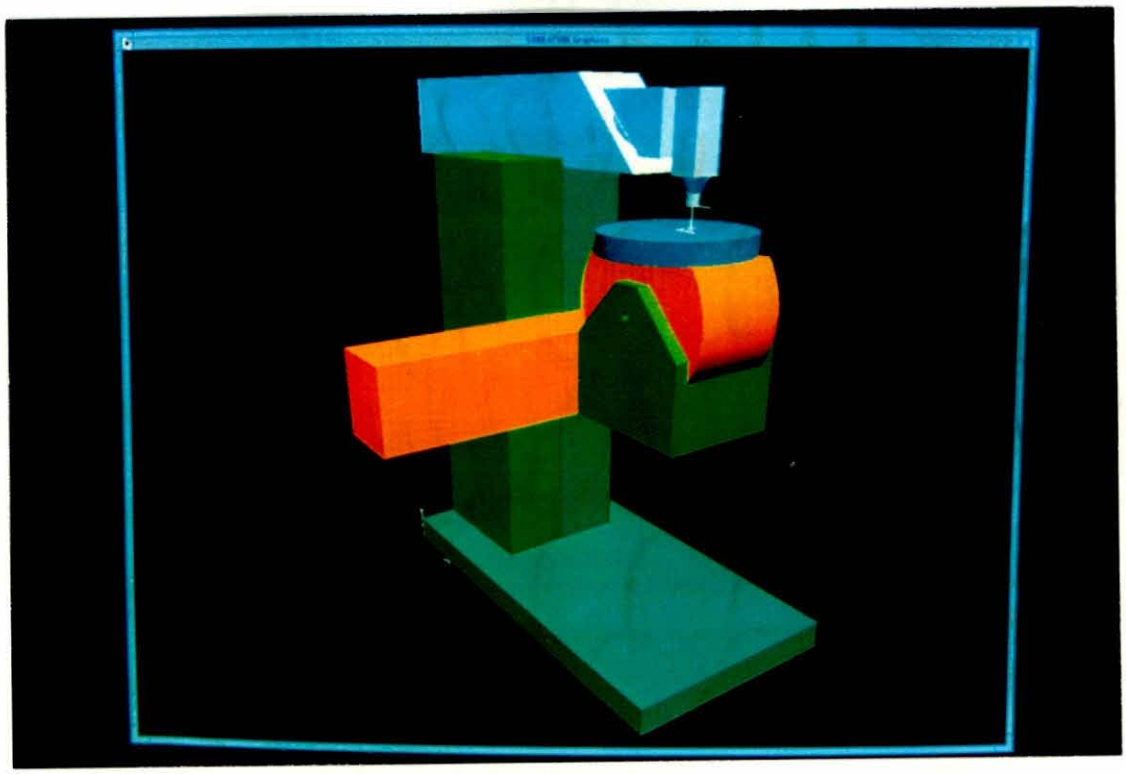


Figure 15: The complete software cell of the MAHO 700S.

4.3 The simulation sequence.

A work cell for simulation has been created, but there has not been any motion simulation so far. The two devices, the Z-axis and the milling table with the X, Y, A and B axes, can move independently after one another. After one of the devices has been made the "active device", joint values can be given with the "goto joints" command. Each joint will move until it reaches the given value.

During an NC program it occurs frequently that both the Z-axis and one of the other axes are moving simultaneously. To accomplish this, a compound device has to be defined of the two devices in the work cell. Now the two devices can move at the same time using the "coordinated goto" command and giving in the joint values.

In Robotics it is possible to place a number of commands in a program in a way it can be replayed. Such a program is called a sequence. When some motion commands are placed in a sequence, they can be performed after another and the motion simulation can be monitored.

```
;****** PLACE Release 9.0 ******
ACTIVE_DEVICE: KUKA;
WORKING_TPOINT: TOORTS,TP;
GOTO_CRD: 1000.000,-430.000,1800.000,65.000,45.000,90.000;
GOTO_TPOINT: TABLE,TPT1,NOP;
GOTO_TPOINT: TABLE,TPT2,NOP;
GOTO_TPOINT: TABLE,TPT3,NOP;
GOTO_HOME: NOP;
```

Figure 16: An example of a sequence.

The designed product can also be brought in the cell. The Unigraphics design of the product has to be converted to Robotics and has to be loaded in the cell as a new frame. When this frame is created it has to be connected to the table of the milling machine. Once it is connected to the table it will move with the table. The orientation of the product has to be the same as during the generation of the CLS file in the manufacturing module. This way it is also possible to show the fixtures that are used. A requirement is that the fixtures are designed with the same orientation with respect to the absolute coordinate system as the product. The easiest way is to design the fixtures in the same part or drawing as the product, but in a different layer of that part.

The Robotics software has the possibility to detect possible collisions during the motion simulation. Previously, the parts on which the detection has to take place have to be defined, and the collision detection has to be turned on. The collision detection option is not included in the simulation program, but is left to the user of the program. The user can decide which parts will be taken up in the collision detection, and when it will be performed.

When the motions of the milling machine have to be simulated, the manufacturing program has to be transformed into a Robotics sequence. Robotics has the possibility to convert a CLS file to the Robotics environment. This results in a cell with tpoints and a sequence. A tpoint is a local coordinate system toward which a device can move. Each cutter location point is represented by a tpoint. The sequence contains the commands to direct the device to the several tpoints.

At this moment the CLS conversion does not work correctly. When a CLS conversion is performed the computer makes a core dump which is a sort of reset of the computer. The cause of this problem could be that Unigraphics and Robotics can not communicate correctly with each other, due to different versions.

When the CLS conversion works, it is possible that it can not work with a cell containing two devices. Also there may occur alignment problems with the tpoints. A milling machine is less flexible than an industrial robot. These problems can not be investigated until the CLS conversion works properly.

An other option is to write a program which translates the NC file into a sequence suitable for Robotics. As far as the motion simulation is concerned, there are four commands which have to be translated: G00, G01, G02 and G03. The G00 and the G01 commands are linear motions which can be performed by goto joints in the work cell. The G02 and G03 command are circular motions.

In Robotics it is possible to perform a circular motion. In the NC file a circular motion is defined by the location of the end point and the location of the arc center. In Robotics a circular motion is defined with an end tpoint and an intermediate tpoint, which is on the circle in between the starting- and end-point. The tpoints have to be placed at the right location and with the right orientation. When this is done a device can make a circular motion. The circular motions of a G02 or G03 command are situated in the X-Z plane. During a circular motion the X-axis and the Z-axis will move. These two axes are part of different devices. So, for both devices the starting-intermediate and end-point are situated on a straight line, which means that it is not possible to place a circle through these points. This means that the circular motions can not be translated in a simple way and requires some kind of interpolater within the translation program.

Figure 17: Simulation of the milling process.

4.4 Program concept for machine simulation.

To simulate an NC program in the Robotics environment, it is necessary to translate this program into a sequence which has a format that Robotics can understand. To ensure that this translation is being performed correctly, a translation program has to be written. This translation program will be simular to the translatin program for the material removal simulation. The translation program has to read the NC file line by line. Every line has to be checked for which action it performs, and then translated in the proper way. The following program concept will scan the NC file on the following actions: G00, G01, G02, G03, M6 and T**.

When the translation program has been started, the user is asked which NC program has to be translated. The user gives in the name of the NC program and it is checked whether the NC program is available in the current directory. If the NC program is not available, a message will appear on the screen and the translation program will be aborted. In the other case, a message will appear on the screen that the translation has begun and the header will be written to the NCname.seq file. The NCname.seq file will eventually form the motion simulation program. The header contains the first lines of the Robotics sequence file. These lines are always the same, but may have to be changed in time when a new version of Robotics is installed.

The header of the sequence:

```
;****** PLACE Release 9.0 ******
DEFINE_COORD_MOTION_DEVICE: FREESBANK,FREESTAF6,FREESB;
SET_DEVICES_MONITORED: FREESBANK,FREESTAF6,FREESB;
ACTIVE_DEVICE: FREESB;
WORKING_TPOINT: FREESB2,TP1;
GOTO_HOME: NOP;
ACTIVE_DEVICE: FREESTAF6;
GOTO_HOME: NOP;
```

Now the program reads the first line of the NC program. If this is the end of the NC program, the footer will be written to the NCname.seq file. Then the NCname.seq file is saved and a message appears on the screen: Translation Complete.

The footer of the sequence:

```
ACTIVE_DEVICE: FREESB;
GOTO_HOME: NOP;
ACTIVE_DEVICE: FREESTAF6;
GOTO_HOME: NOP;
```

If the line is not the end of the file, the first characters are skipped, and from the first space three characters are read. These characters are being compared with: G00, G01, G02, G03 and T**. If the characters match T**, a check is made if there is an M6 command to see if the tool really is being changed. In the case of a tool change, the right device has to be found which matches with T**. Then the tool change

module has to be written to the NCname.seq file.

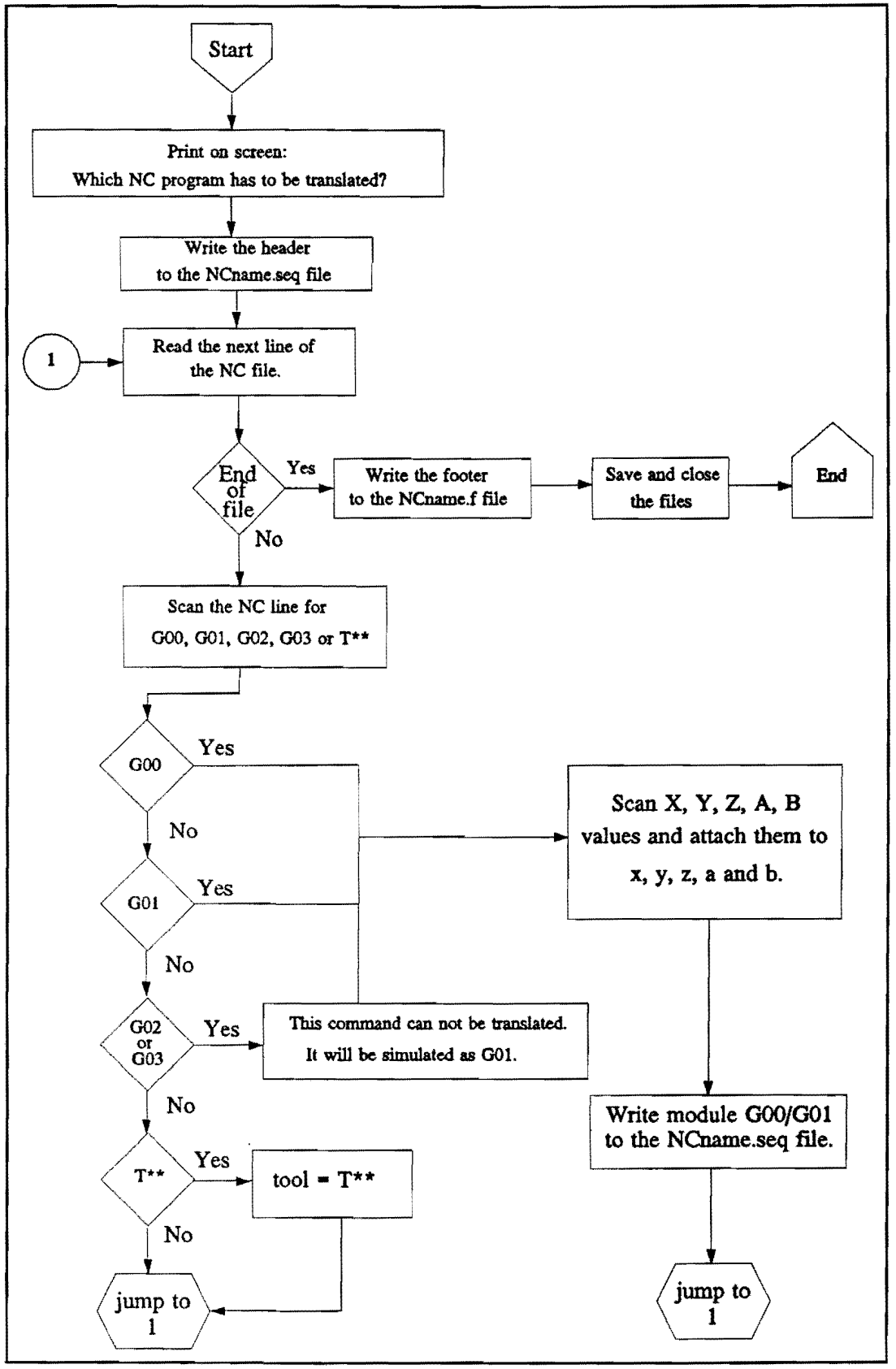
```
DELETE_DEVICE: Current Device;
MERGE_DEVICE: New Device,WORLD;
COLOR: FREESB0,PART,RGB,FORESTGREEN,0.3137,0.6235,0.4118,1.0000;
COLOR: FREESB1,PART,RGB,LIGHTBLUE,0.6902,0.8863,1.0000,1.0000;
DEFINE_COORD_MOTION_DEVICE: FREESBANK,FREESTAF6,New Device;
SET_DEVICES_MONITORED: FREESBANK,FREESTAF6,New Device;
ACTIVE_DEVICE: New Device;
WORKING_TPOINT: FREESB2,TP1;
```

Then the next line of the NC program is read, and the characters are being compared again. If the characters match G02 or G03, a message will appear on the screen which tells the user that these commands can not be translated right now, and the translation program is aborted. In the case of a G00 or G01 command, the next space is skipped and one character is read. This character is being compared with: end of line, X, Y, Z, A or B. In the case of X, Y, Z, A or B, the digits after the X, Y, Z, A or B are read and are defined to x, y, z, a or b. If the character is the end of line, the module G00/G01 has to be written to the NCname.seq file.

```
COORDINATED_GOTO:FREESBANK,FREESTAF6,JOINTS,(MM),
y,x,A,B,0.000,FREESB,JOINTS,(MM),z,0.000,NOP;
```

Now the next line of the NC program is read, and the process of checking starts over again.

4.5 The flowchart of the translation program.



Chapter 5. Conclusions

It turned out to be impossible to perform a material removal simulation and a motion simulation within one software environment. This resulted in the dividing of the two simulations:
- material removal simulation with the use of Unigraphics and
- motion simulation with the use of Robotics.

The material removal simulation is being performed by translating an NC program into a User Function program. This program shows the subtraction of milling paths from the raw material. First a program concept is made for G00, G01 and tool change commands. The simulation program gives a good reflection of the material removal.

The motion simulation is being performed by translating an NC program into a Robotics sequence. This program shows the motions of the axes of the milling machine. The G00 and G01 commands can be simulated without problems. The G02 and G03 commands can not be simulated right now, because the devices representing the milling machine can not perform a circular motion together.

Both programs have been tested by performing a manual translation from an NC program to a simulation program. For both simulations a program concept has been made but the actual programming of the translation programs still has to be done.

In the future, the material removal simulation can be extended with more modules for translating NC commands. The G00/G01 module has to be adapted in a way that the motions of the A and B axes motions can be simulated as well. With the change of the direction vectors, the module G00/G01 can be used for a horizontal milling module.

For the material removal simulation it would be helpful if it was possible to design a parameterized tool solid instead of building the tool solid with the User Function routines. Perhaps this will be possible with UG Concept.

For the motion simulation, more cutting tools have to be modelled, which means the modelling of more devices. Before this the dimensions of the current work cell with the milling machine have to be compared to the actual milling machine: the MAHO 700S. For the simulation of the G02 and G03 commands it has to be determined whether it is possible include a circular interpolator in the translation program.

On the computer, which is used right now, the material simulation is modelled in solids, however the representation is in wireframe. On a computer with a greater speed it is possible to represent the simulation with shaded images.

Literature

[1] Friedman, F.L., Koffman, E.B., *Problem Solving and Structured Programming in Fortran 77*, third edition

[2] Heintjes, T.B., *Handleiding voor het gebruik van de Manufacturing Operations Module uit het Unigraphics systeem*, WPAnr. 1168, Eindhoven, 1991

[3] Hewlett Packerd Company, *A Beginners Guide to HP-UX*, 1991.

[4] Kaas, E.A., Stakenborg, M.J.L., *CAD/CAM/CAE in de Werktuigbouw*, Deventer, 1990

[5] McDonnell Douglas Corporation, *Build User Guide*, version 7.0, 1990.

[6] McDonnell Douglas Corporation, *GRIP Programming Manual Vol1 & Vol2*, Version 8.0, 1991.

[7] McDonnell Douglas Corporation, *Place User Guide*, Version 7.0, 1990.

[8] McDonnell Douglas Corporation, *UG Solids operational description*, Version 8.0, 1991.

[9] McDonnell Douglas Corporation, *UnigraphicsII Concepts and Common Functions*, Version 8.0, 1991.

[10] McDonnell Douglas Corporation, *User Function Manual*, Version 8.0, 1991.

[11] Melio, J.P., *Executing ADJUST on the FALC cell*, WPAnr, 1196 Eindhoven,1991.

[12] Senden, R., *Realiseren van een CAD/CAM koppeling*, WPAnr. 0930, Eindhoven, 1990.

[13] Vernooij, J, *The simulation of the MAHO-700S milling machine by the use of the software package Robotics*, WPAnr. 1088, Eindhoven, 1991.

[14] Willems, M.C., *The off-line programming of welded products with the Robotics software*, WPAnr. 1321, Eindhoven, 1992.

Appendices.

The simulation of NC programs generated with the UnigraphicsII software.

J.P. Melio
WPA nr. 1355
June 1992

Eindhoven University of Technology
Faculty Mechanical Engineering
Division WPA

Professor: prof. dr. ir. A.C.H. van der Wolf

Coaches: ing. J.J.M. Schrauwen
F.G.J. Soers

Eindhoven, july 1992

Summary

Computer Aided Design (CAD) systems often contain a manufacturing module which is able to generate NC programs for the designed products. Instead of testing the NC program using the real milling machine, it is desirable to perform a computer simulation of the NC program. This way irregularities can be corrected before the NC program is executed on the milling machine.

Simulation programs for NC programs are being sold by several software firms. Instead of purchasing simulation software, the choice was made to try to develop a simulation program with the use of the software already available within the division WPA, part of the faculty Mechanical Engineering at the TUE. Besides a material removal simulation a motion simulation program is also made to monitor the motions of the milling machine. This way it can be checked whether the material is removed correctly and if any collisions occur during the manufacturing.

It appeared that the material removal simulation and the motion simulation could not be made with the use of one software environment. Therefore, the material removal simulation is made with the use of UnigraphicsII, which is an extensive CAD system. The Unigraphics tool User Function is used to program the simulation. The motion simulation program is being performed with Robotics. Robotics is designed to perform off-line robot programming, but in the simulation environment of Robotics, PLACE, it is possible to simulate a milling machine.

For the material removal simulation, an NC program is being translated into a User Function program. This program builds a tool path and subtracts this from the raw material. This assignment only deals with the simulation of vertical linear planar milling actions and a program concept for the translation from NC program to User Function program is made. This concept has to be transformed into a computer program.

Within Robotics a model of the milling machine has been brought into a PLACE simulation cell. In order to perform a milling simulation, an NC program has to be translated into a Robotics sequence. A program concept for the translation from NC program to Robotics sequence is made, but this concept has to be transformed into a computer program. At this moment G02 and G03 commands can not be translated.

The two simulation programs have been tested by performing a manual translation from the NC program. The material removal simulation gives a good reflection of the material being removed. With the motion simulation it is possible to check the NC program for collisions.

With this research assignment a start has been made for the simulation of NC programs but in the future the simulations have to be developed further.

Samenvatting

Computer Aided Design (CAD) systemen bezitten vaak een manufacturing module waarmee NC-programma's gegenereerd kunnen worden voor de ontworpen produkten. In plaats van NC-programma's te testen op de werkelijke freesbank, is het wenselijk om de programma's te simuleren met behulp van de computer. Op deze manier kunnen fouten gecorrigeerd worden voordat het programma uitgevoerd wordt op de freesbank.

Simulatieprogramma's voor NC-programma's worden door verschillende software bedrijven verkocht. In plaats van het aanschaffen van simulatie software, is de keuze gemaakt om deze zelf te ontwikkelen met de software die beschikbaar is binnen de vakgroep WPA, deel van de faculteit Werktuigbouwkunde aan de TUE. Naast een materiaalafname-simulatie is ook een bewegingsimulatie gemaakt, om de bewegingen van de assen van de freesbank te bekijken. Met deze simulaties kan bekeken worden of het materiaal correct wordt afgenomen en of er zich botsingen voor doen tijdens het bewerken van het produkt.

Het bleek dat de materiaalafname-simulatie en de bewegingssimulatie niet met één software pakket gemaakt kon worden. De materiaalafname-simulatie is gemaakt met UnigraphicsII, een krachtig CAD pakket. Het Unigraphics deelpakket User Function is gebruikt om de simulatie te programmeren. De bewegingssimulatie wordt uitgevoerd met behulp van Robotics. Robotics is ontworpen voor het off-line programmeren van robots, maar binnen de simulatie omgeving van Robotics, PLACE, is het ook mogelijk om een freesbank te simuleren.

Voor de materiaal afname simulatie wordt een NC-programma vertaald in een User Function programma. Dit programma modelleert een freesbaan en trekt deze van het uitgangsmateriaal af. Deze opdracht behandelt alleen de simulatie van het vlak frezen met verticale freeskop en een opzet voor een vertaal programma van NC-programma naar User Function programma is gemaakt. Deze opzet moet nog vertaald worden in een computer programma.

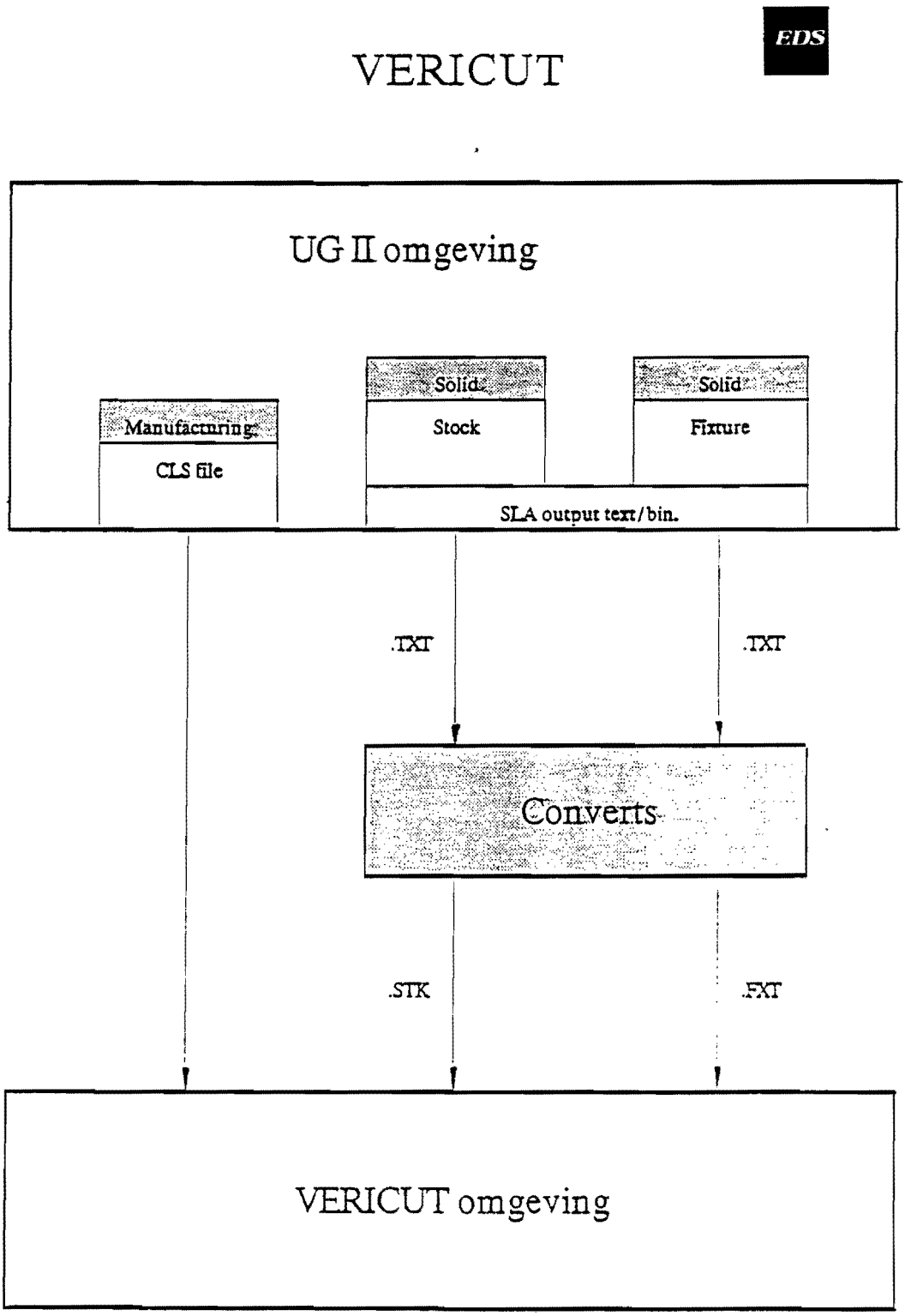
Binnen Robotics is een model van de freesbank in de PLACE-simulatie-cel gebracht. Om een bewegingsimulatie uit te voeren, moet een NC-programma vertaald worden in een Robotics-sequence. Een concept voor de vertaling van NC-programma naar Robotics sequence is gemaakt, maar dit concept moet nog vertaald worden in een computer programma. Op dit moment kunnen G02 en G03 commando's nog niet vertaald worden.

De simulatie-programma's zijn getest door het uitvoeren van handmatige vertalingen van het NC-programma. De materiaalafname-simulatie geeft een goed beeld van het materiaal dat wordt afgenomen. Met de bewegingssimulatie is het mogelijk om te controleren of er botsingen voorkomen in het NC-programma.

Met deze opdracht is een begin gemaakt voor de simulatie van NC-programma's, maar in de toekomst moeten de simulaties verder ontwikkeld worden.

CONTENTS

Appendix A. Vericut.



Appendix B. Preparing the HP_UX system for User Function Programs.

B.1 Environment variables.

Before User Function Programs can be used, the following environment variables have to be defined.

UGII_UGUSER_FILE	This variable is defined at installation time and initially points to the UGUSER.DAT file.
UGII_USERFCN	This variable is defined at installation time and points to the UnigraphicsII directory where the User Function libraries reside.
UFUN_USERn	You define this variable to point to your executable image link made with UFLINK. You use this link for identifying internal executable programs delimited by a pound sign and a number "n" (#n) in your UGUSER.DAT file. n can be an integer from 1 to 99.

These variables have to be defined in your .profile file. This file is used when you login to set the environment variables you need and to start the menu. With the vi editor the variables mentioned above can be added to your .profile file. First the variable has to be defined and after that it has to be exported so the system knows that you defined it. Example:

```
UGII_UGUSER_FILE=/usr/disk2/users/melio/c/uguser.dat
export UGII_UGUSER_FILE
```

This is the .profile file I used:

```
# @(#) $Revision: 64.3 $

# Default user .profile file (/bin/sh initialization).

# Set up the terminal:
        eval ' tset -s -Q -m ':?hp' '
        stty erase "^H" kill "^U" intr "^C" eof "^D"
        stty hupcl ixon ixoff
        tabs

# Set up the search paths:
        PATH=/bin:/usr/bin:/usr/contrib/bin:/usr/local/bin:.
```

```
# Set up the shell environment:
        set -u
        trap "echo 'logout'" 0

# Set up the shell variables:
        EDITOR=vi
        export EDITOR

# Set up variable for User Function Menu:
        UGII_UGUSER_FILE=/usr/disk2/users/melio/c/uguser.dat
        export UGII_UGUSER_FILE

# Set up variable for User Function Libraries:
        UGII_USERFCN=/usr/disk2/mdc/ufunc
        export UGII_USERFCN

# Set up variable for User Function Programs:
        UFUN_USER1=/usr/disk2/users/melio/f/name
        export UFUN_USER1

        UFUN_USER3=/usr/disk2/users/melio/f/blok
        export UFUN_USER3

# Tbv Simulation V8.0 dd 7 okt 1991
        . /usr/disk2/simroot/SimProfile
        $UGII_ROOT_DIR/ugmenu
```

B.2 The UGUSER.DAT file.

The uguser.dat file contains the menu which is displayed in UnigraphicsII when User Functions is started. It contains the menu title, program numbers, task identifiers and passwords. This file can be made with the vi editor and may reside in your own directory. Make sure that the environment variable mentioned above points to this directory. The uguser.dat I used is shown below; it does not contain passwords.

```
User Function Menu
#1
Testprogramma uftest
/usr/disk2/mdc/bin/fmexec
file management
#3
programma: blok
```

Appendix C. Compiling and linking User Function Programs.

When you write a User Function program (or any other Fortran program) the program name must have the extension ".f", for example: uftest.f. This program can be written using any text editor. At the HP station the vi editor is available. Information about this editor can be found in the manual: Beginners guide to the HP-UX. [3]

When the program is finished, it has to be compiled. Therefor you type:

```
f77 -o uftest uftest.f
```

and the compiler does all the work for you. The compiler uses the file uftest.f to generate an output file with an ".o" extension, in this case uftest.o.

After the programs has been compiled, it has to be linked to the libraries it uses. For an internal User Function program the following has to be typed:

```
uflink -r uftest
```

A message is displayed wether the linking has been successful. The linking program generates two files; one file without an extension, and one file with the extension ".snt", for example: uftest
uftest.snt

Now the User Function Program can be executed.

Appendix D. Advice about C and Fortran.

Aan: prof.dr.ir. v.d. Wolf
Van: E. Gerritsen, (2771)
Cc: ir. W.Senden
Datum: 09 april 1992
Onderwerp: afstudeerprojekt Melio

Naar aanleiding van een gesprek met dhr Melio de volgende opmerkingen:

1. Het doel van het (deel-)projekt is om de materiaalafname bij het frezen te simuleren op het HP-werkstation in de Unigraphics-omgeving. (SOLIDS).

2. De Unigraphics-software is geschreven in FORTRAN, en de libraries die gebruikt worden om een ufunc-programma te maken zijn dat eveneens.

3. Er is op het systeem geen FORTRAN-compiler aanwezig (niet aangeschaft). Melio probeert het nu via de taal C (die compiler is aanwezig), maar dat leidt door de incompatibiliteit tussen de calls van FORTRAN en C, tot een aantal problemen, zoals core dumps, disk full en system crash.

4. Mogelijke oplossingen.

   A. Aanschaf FORTRAN-compiler (kosten Fl 3864,- licentie, Fl 495,- documentatie, Fl 495,- 1/4"-tape) ; er is misschien de optie om het via de onderwijskorting van 90% aan te schaffen.

   B. Onderzoek door dhr Braam van het RC naar een oplossing voor de incompatibiliteit tussen C en FORTRAN op de HP.

   C. Wachten op versie 9 van Unigraphics, daarin zou het beter geregeld zijn. (eind '92)

   D. Ontwikkelen op de VAX (de machine is traag en geen SOLIDS).

   E. Compiler "lenen" bij BdK (HP 9000/300), als er daar een compiler is.

   F. Compiler "lenen" bij WFW, als er daar een compiler is en als HP-9000/400 compatibel is.

5. Aanbevelingen.

   A. Dit is de beste optie.
   B. Kan in ieder geval gedaan worden.
   C. Gaat niet (tijdsdruk).
   D. Niet realistisch.
   E. Misschien.
   F. Misschien.

Swidbert Dohmes

Note 5.1 Userfunctions, C and Character Arrays 1
UGUSER::DOHMES "Swidbert Dohmes (EDS Koeln)" 43 lines 3-DEC-1991 1
-< Details on "C" callable UFUNC.... >-

<<< BBS$NOTE:[NOTES$LIBRARY]LANG01.NOTE;1 >>>
-< LANGUAGES Conference >-

Note 397.5 Link FORTRAN and C on ULTRIX 5
SYSJJ::JBAKER "JOHN R. BAKER, M&E CAD PRODUCT MAN" 35 lines Wed 27-NOV-91 1
-< Details on "C" callable UFUNC.... >-

#> <<< Note 397.4 by BBS::JORGENSEN "GLENN JORGENSEN (TEXTRON LYCOMING)"
#> -< Wait till V9 >-
#>
#> I know this may not be of much right now, but I recall from the Bosto
#> User's meeting in September that UG v9 is suppose to allow C to work
#> with User Function.

UFUNC works with "C" now, just that any routine that passes strings must have a "shell" built around it so that it works. What we are doing with V9.0 is breaking the UFUNC libaray into 3 classes of routines:

Those routines that can be called by either FORTRAN or "C".

Those routines that can be called by FORTRAN.

Those routines that can be called by "C".

The last 2 sets are made up of the routines that pass strings and are actually duplicates of each other in the sense that for every FORTRAN callable routine there will be an equivalent "C" callable routine. The new "C" callable routines will have a new pre-fix of UCXXXX. For each routine that now has the UCXXXX designation, the XXXX portion will remain the same as the equivalent UFXXXX routine for easy conversion of existing programs. The current naming convention, UFXXXX, will continue for routines that are callable by either FORTRAN or "C", and for those that are callable only by FORTRAN. In that way all existing programs can be left as is and they will still operate.

Note that all 3 sets of routines will be delivered with the single license of UFUNC. All existing clients with valid UFUNC licenses will be recieving this updated set of routines when V9.0 is delivered.

I hope this helps explain what we are doing for V9.0.

John

Appendix E. Realized and not relevant machine instructions.

from: "realiseren van een CAD/CAM koppeling" door R. Senden.[12]

Realized	Not relevant
%PM	G04
N	G11
G00	G14
G01	G19
G02	G22
G03	G29
G17	
G18	G51
	G52
G40	G70/G73
G41	G77
G42	G78
G43	
G44	G87
	G88
G53/G59	G89
G79	G90/G92
G81	G94
G83	G95
G84	
G85	R
G86	D
	P
G93	M0
	M13
X	M14
Y	M19
Z	M66
I	E
J	
K	
A	
B	
F	
S	
T	
M3	
M4	
M5	
M6	
M8	
M9	
M30	
M53	
M54	

Appendix F. User Functions which were used.

UF1601 Display a Simple Message on the Message Monitor

Internal & External

FORMAT CALL UF1601(CP1,IP2)

PARAMETERS

Parameter	I/O	Type	Description
CP1	I	CHARACTER*(80)	Message
IP2	I	INTEGER*2	Wait Option 0 = Display Message Only 1 = Wait For Acknowledgment - E/C

DESCRIPTION

INTERNAL: A maximum of 80 characters can be displayed in CP1. An asterisk may be used to indicate where CP1 should be split to continue on the second line.

EXTERNAL: Asterisks in CP1 do not split the display string.

EXAMPLE

```
C
C     Inquire the name of new part.
C
      TITLE   = 'Enter New Part Name'
      PARTNM = 'BLANK.PRT'
C
      CALL UF1600(TITLE,PARTNM,NUMCHR,IRC)
       IF (IRC .LE. 2) GOTO 100
C
       IF (PARTNM .EQ. ' ') THEN
     *    CALL UF1601('BLANK PART NAME IS INVALID',1)
        GOTO 100
       ENDIF
```

UF1617

Internal

Entity Select with Class Selection Menu

FORMAT CALL UF1617(CP1,IP2,NR3,IR4,RR5,IR6)

PARAMETERS

Parameter	I/O	Type	Description
CP1	I	CHARACTER*(40)	Menu Title
IP2	I	INTEGER*2	Maximum Number Of Entities To Return
NR3(*)	O	INTEGER*4	Array Of Entity EIDS
IR4	O	INTEGER*2	Number Of Entities Returned
RR5(3)	O	REAL*8	Last Cursor Position - If Cursor Used
IR6	O	INTEGER*2	Response Returned 1 = Reject 2 = Terminate Operation 3 = Entry Complete 5 = Entities Returned 6 = More Entities Picked Than Returned 7 = No Active Part 8 = Not In Internal User Function

DESCRIPTION

NOTE: Chaining is allowed through the Alternate Action button.

With this routine, you can pick entities displayed on the screen using the class selection menu.

EXAMPLE

```
C
C     Select Masked Entities and change the font.
C
      CALL UF1617(T1617,EIDDIM,EIDARY,EIDCNT,PNT1,IRC)
      IF (IRC .EQ. 1) GOTO 5000      ! Reject
      IF (IRC .EQ. 2) GOTO 100       ! terminate
      IF (IRC .EQ. 7) GOTO 80000     ! No Active Part
      IF (IRC .EQ. 6) THEN
       CALL UF1601('MORE ENTITIES SELECTED THAN ALLOWED',1)
       ENDIF
```

UF5001 Retrieve Part File

Internal & External

FORMAT CALL UF5001(CP1,IR2)

PARAMETERS

Parameter	I/O	Type	Description
CP1	I	CHARACTER*132	Filespec Of Part To Retrieve
IR2	O	INTEGER*2	Result Of Operation 0 = Good Return 1 = Corrupt Part Or Version Mismatch <0 = UGFM Error

DESCRIPTION

NOTE: When retrieving a part, you must have access to the following UNIGRAPHICS files:

AOS/VS:
- 00.UG
- 06.UG
- 10.UG
- CONFIG.UG

The file is accessed through the same searchlist (AOS/VS), logical (VAX/VMS), or Environment Variables (UNIX) normally used to run UNIGRAPHICS.

UF5001 uses UNIT 9 during part retrieval. Do not use unit 9 for any input or output.

UF5001 will retrieve parts that were selectively filed.

If a User Function program linked with a version X library tries to retrieve a UG part filed with UG Version X+1, a return code (IR2) of 1 will result.

UF5016 Edit Entity Color

Internal & External

FORMAT CALL UF5016(NP1,IP2)

PARAMETERS

Parameter	I/O	Type	Description
NP1	I	INTEGER*4	Entity Identifier
IP2	I	INTEGER*2	New Value For Color

NOTE: The UG color corresponding to a color code is a function of the user-defined color table. These are the defaults.

Table 10-1 User Defined Color table

1 = Blue
2 = Green
3 = Cyan
4 = Red
5 = Magenta
6 = Yellow
7 = White
8 = Olive
9 = Pink
10 = Brown
11 = Orange
12 = Purple
13 = Dark Red
14 = Aquamarine

EXAMPLE

```
C
C     Edit entity color.
C
      CALL UF5016(LINEID(I),15-OLDCOL)
      NEWCOL = UF5015(LINEID(I))
      IF (NEWCOL .NE. 15-OLDCOL) THEN
       CALL UF4403('*UF5015/6 - DID NOT CHANGE COLOR',RTC)
      ENDIF
      CALL UF5016(LINEID(I),I)
210   CONTINUE
```

UF6500 Create a Solid Block

Internal & External

FORMAT CALL UF6500 (RP1,RP2,RP3,NR4)

PARAMETERS

Parameter	I/O	Type	Description
RP1(3)	I	REAL*8	A Corner Of The Block - Absolute Coordinates
RP2(3)	I	REAL*8	Block Dimensions = Size (1) = X Length (2) = Y Length (3) = Z Length
RP3(6)	I	REAL*8	Matrix Of Direction Vectors (1-3) X Direction (4-6) Y Direction
NR4	O	INTEGER*4	Block EID 0 = Error

DESCRIPTION

Creates a block at CORNER in the Absolute Coordinate System. A block has 8 corners; which corner is picked depends on the sign of SIZE: If SIZE(1) > 0 then the X side is measured from CORNER, along the positive X axis, and vice versa. The same method of measurement is applied for the Y and Z sides. Together, the X, Y, and Z sides determine the "direction" of the block, and the appropriate corner to pick.

EXAMPLE

```
C
C     Create two (2) blocks.
C
      CALL UF6500(CRNER1,SIZE1,MXT1,BLKEID)
      IF (BLKEID .EQ. 0) THEN
       CALL UF6560(ERMESS)
       CALL UF4403(ERMESS,RTC)
      ENDIF
C
      CALL UF6500(CRNER2,SIZE2,MXT2,BLKEID)
      IF (BLKEID .EQ. 0) THEN
       CALL UF6560(ERMESS)
       CALL UF4403(ERMESS,RTC)
      ENDIF
```

UF6502 Create a Solid Cylinder

Internal & External

FORMAT CALL UF6502 (RP1,RP2,RP3,RP4,NR5)

PARAMETERS

Parameter	I/O	Type	Description
RP1(3)	I	REAL*8	Cylinder Base Origin - Absolute Coordinates
RP2	I	REAL*8	Cylinder Height
RP3	I	REAL*8	Cylinder Diameter
RP4(3)	I	REAL*8	Cylinder Axis - Absolute Coordinates
NR5	O	INTEGER*4	Cylinder EID 0 = Error

DESCRIPTION

The solid cylinder is created in the direction of its axis. Cylinder height RP2 could be negative, in which case the cylinder is created in the opposite sense of its axis.

EXAMPLE

```
C
C     Create the solid cylinder.
C
      CALL UF6502(CYLORI,CYLHGT,CYLDIA,CYLAXS,CYLEID)
      IF (CYLEID .EQ. 0) THEN
       CALL UF6560(ERMESS)
       CALL UF4403(ERMESS,RTC)
      ENDIF
```

UF6520

Internal & External

Unite, Intersect or Subtract Solid Entities

FORMAT **CALL UF6520 (IP1,NP2,NP3,IP4,IP5,IR6,IR7,NR8)**

PARAMETERS

Parameter	I/O	Type	Description
IP1	I	INTEGER*2	Function 0 = Unite 1 = Intersect 2 = Subtract
NP2	I	INTEGER*4	Target Solid EID
NP3(*)	I	INTEGER*4	List Of EID's Of Tool Solids —must Be Dimensioned (IP4) Or Greater
IP4	I	INTEGER*2	Number Of Tool Solids In The List NP3
IP5	I	INTEGER*2	Maximum Number Of Returned Solids
IR6	O	INTEGER*2	Error Code 0 = Success 1 = Failure
IR7	O	INTEGER*2	Number Of (Actual) Resulting Solids
NR8(*)	O	INTEGER*4	List Of EID's Of Resulting Solids —must Be Dimensioned (IP5) Or Greater

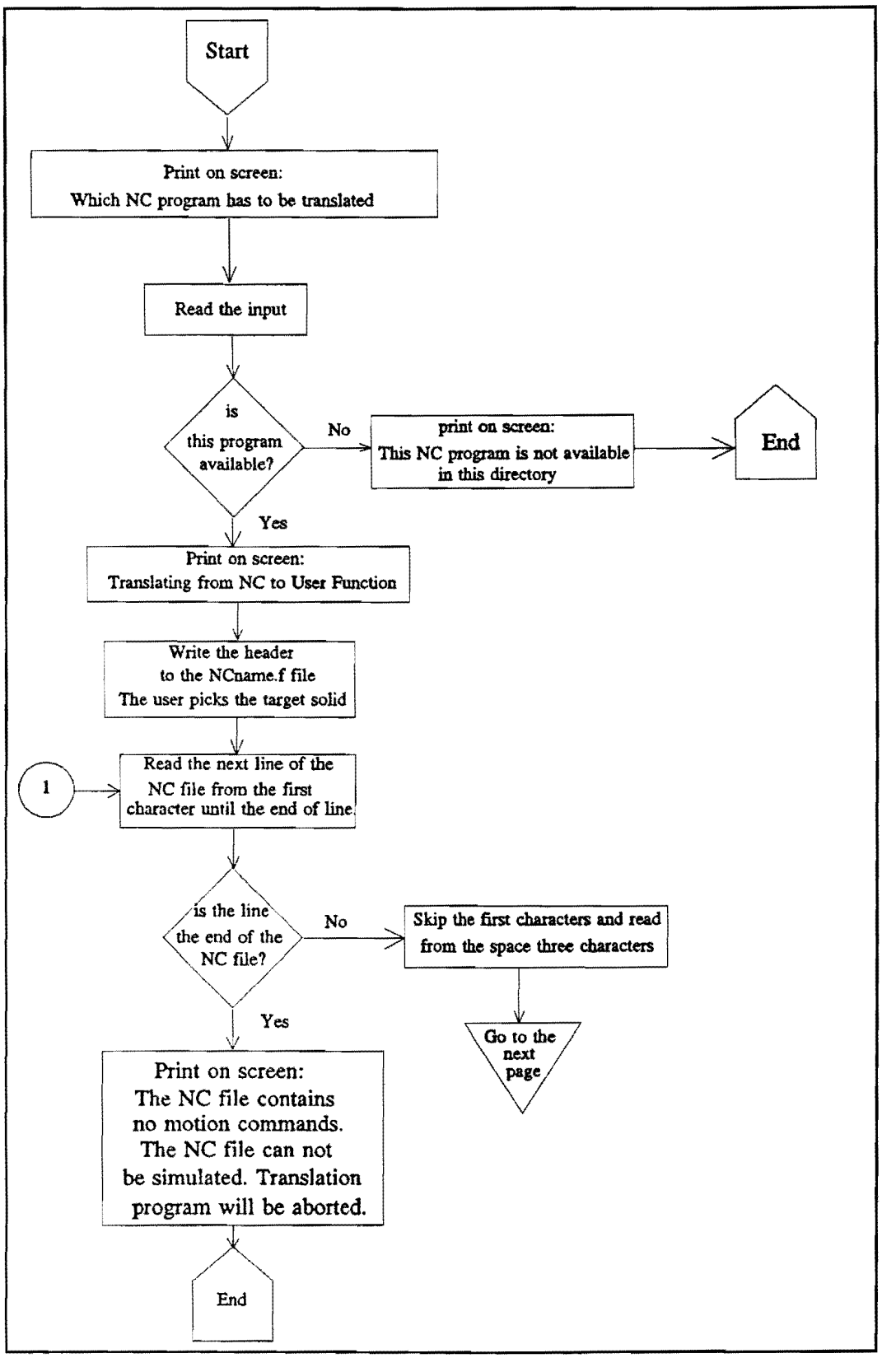
DESCRIPTION

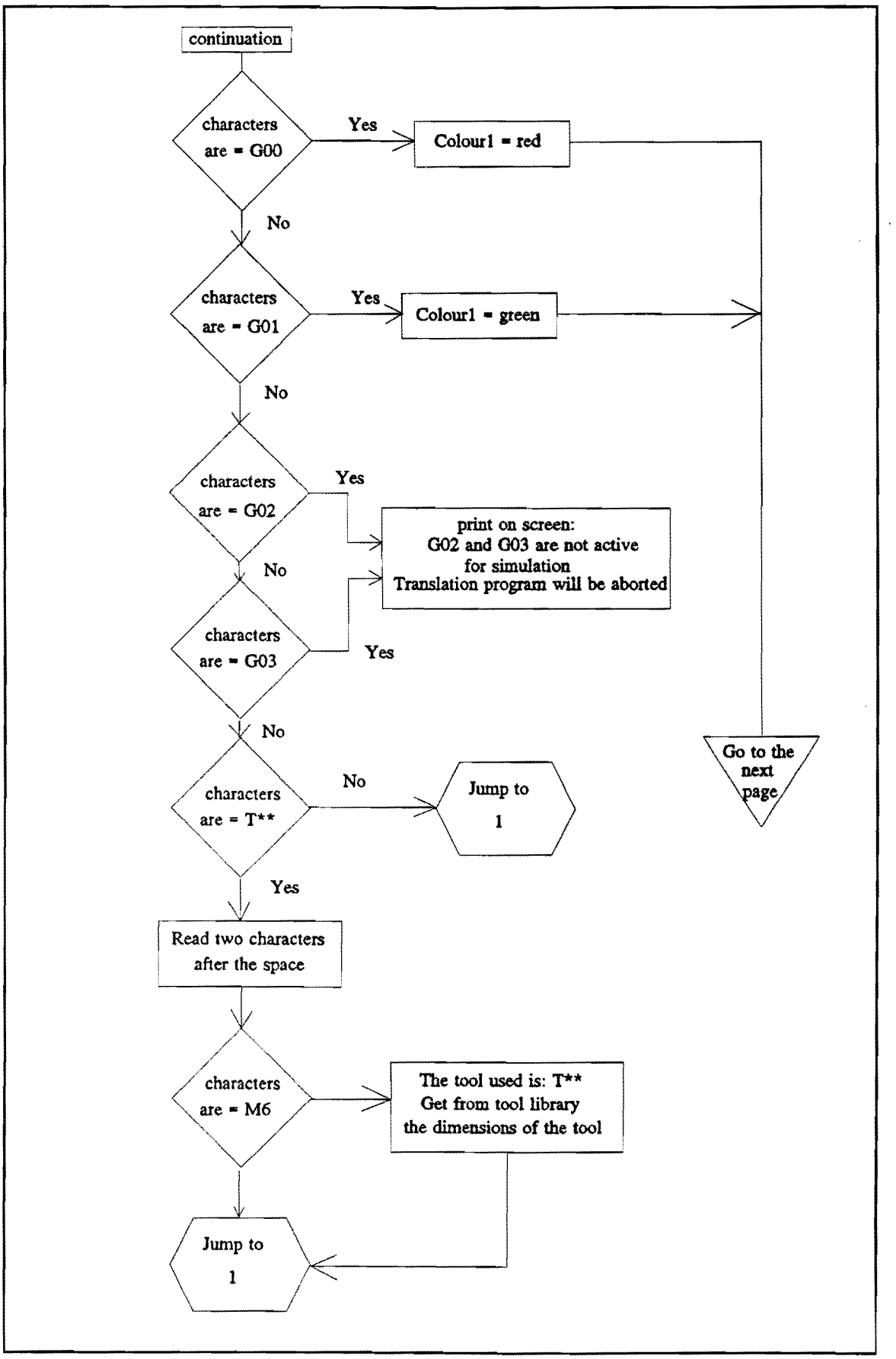
Unites, intersects or subtracts the solids in the list NP3 with/from the target solid NP2. If IR7 > IP5 then only the first IP5 resulting solids get returned.

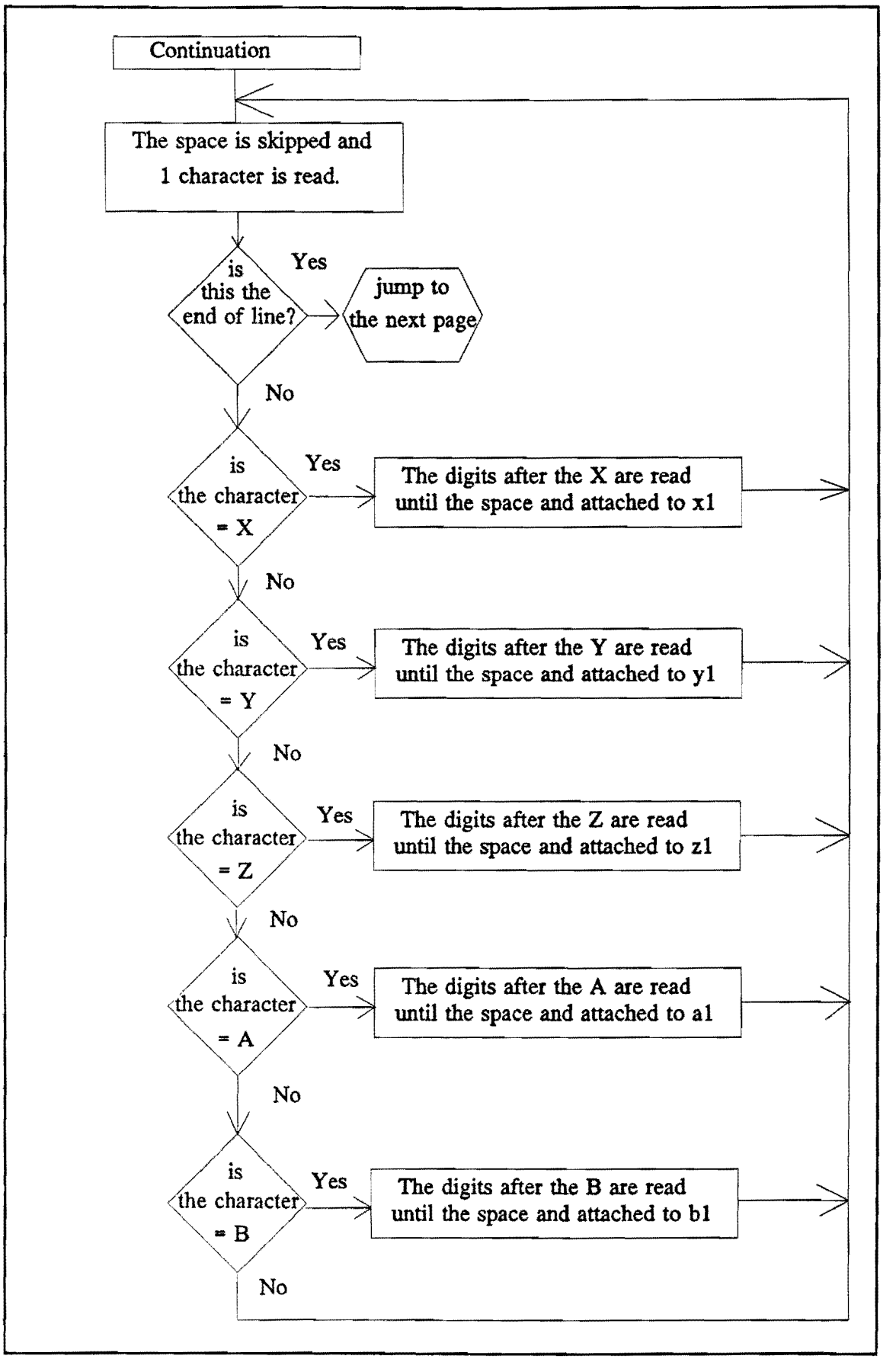
EXAMPLE

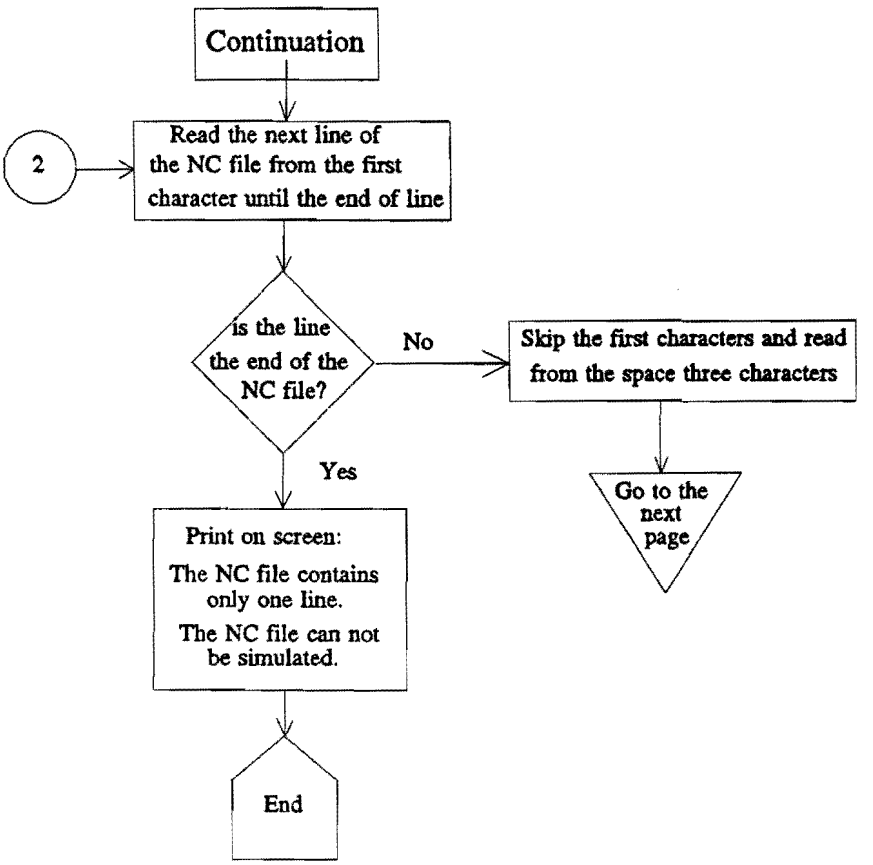
Refer to example for subroutine UF6511.

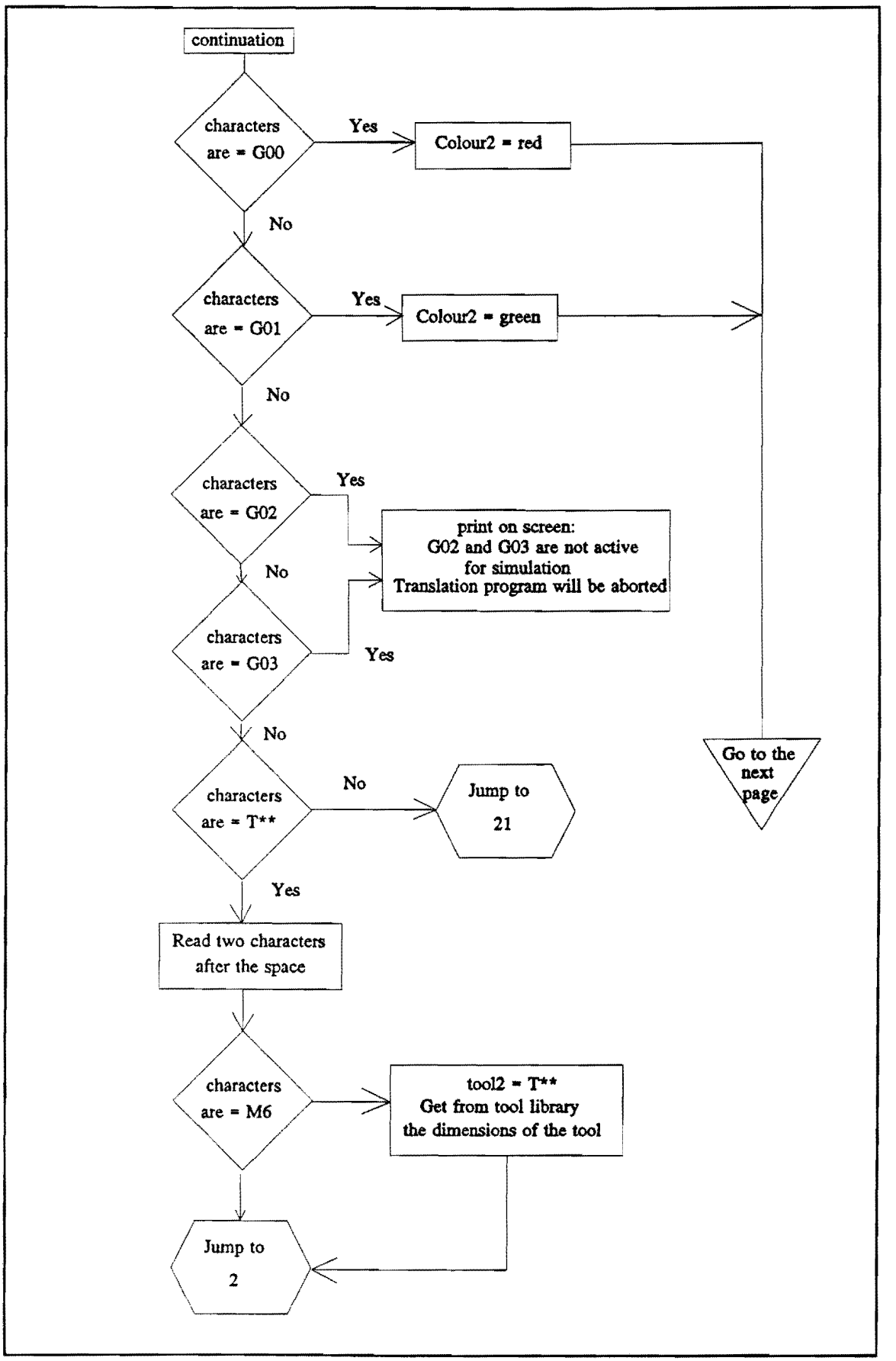
Appendix G. Extensive flowchart for material removal simulation.

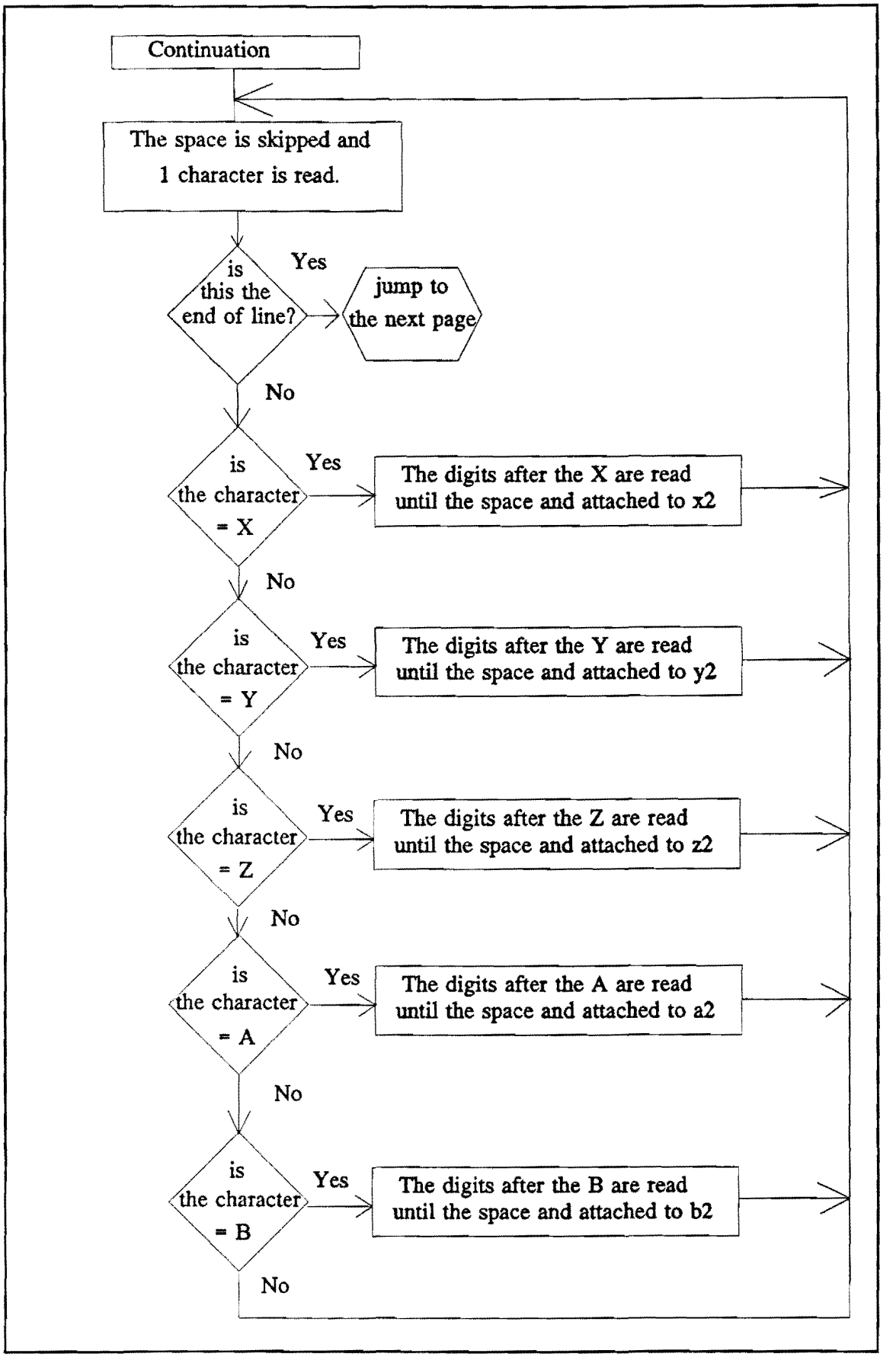


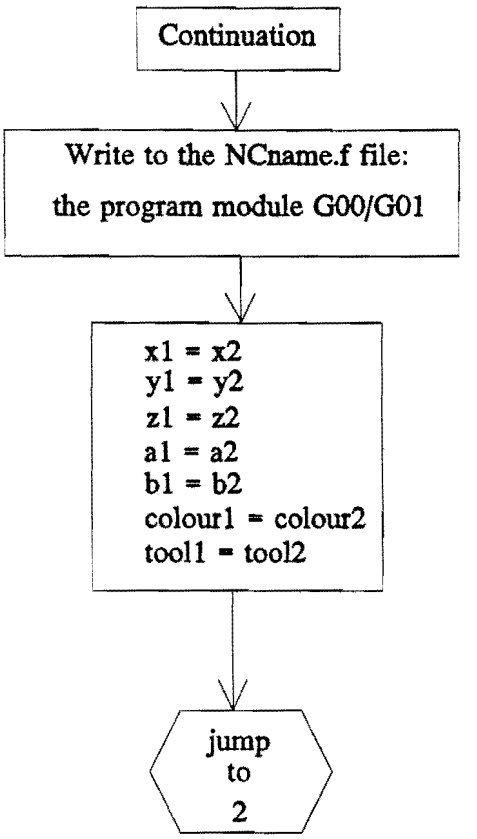

continuation
characters are = G00
Yes
Colour1 = red
No
characters are = G01
Yes
Colour1 = green
No
characters are = G02
Yes
print on screen:
G02 and G03 are not active
for simulation
Translation program will be aborted
No
characters are = G03
Yes
No
characters are = T**
No
Jump to 1
Yes
Read two characters after the space
characters are = M6
The tool used is: T**
Get from tool library
the dimensions of the tool
Jump to 1
Go to the next page


Continuation
The space is skipped and 1 character is read.
is this the end of line?
Yes
jump to the next page
No
is the character = X
Yes
The digits after the X are read until the space and attached to x1
No
is the character = Y
Yes
The digits after the Y are read until the space and attached to y1
No
is the character = Z
Yes
The digits after the Z are read until the space and attached to z1
No
is the character = A
Yes
The digits after the A are read until the space and attached to a1
No
is the character = B
Yes
The digits after the B are read until the space and attached to b1
No


Continuation
2
Read the next line of the NC file from the first character until the end of line
is the line the end of the NC file?
No
Skip the first characters and read from the space three characters
Go to the next page
Yes
Print on screen:
The NC file contains only one line.
The NC file can not be simulated.
End


continuation
characters are = G00
Yes
Colour2 = red
No
characters are = G01
Yes
Colour2 = green
No
characters are = G02
Yes
print on screen:
G02 and G03 are not active
for simulation
Translation program will be aborted
No
characters are = G03
Yes
No
characters are = T**
No
Jump to
21
Yes
Read two characters after the space
characters are = M6
tool2 = T**
Get from tool library
the dimensions of the tool
Jump to
2
Go to the next page


Continuation
The space is skipped and 1 character is read.
is this the end of line?
Yes
jump to the next page
No
is the character = X
Yes
The digits after the X are read until the space and attached to x2
No
is the character = Y
Yes
The digits after the Y are read until the space and attached to y2
No
is the character = Z
Yes
The digits after the Z are read until the space and attached to z2
No
is the character = A
Yes
The digits after the A are read until the space and attached to a2
No
is the character = B
Yes
The digits after the B are read until the space and attached to b2
No


Continuation
Write to the NCname.f file:
the program module G00/G01
x1 = x2
y1 = y2
z1 = z2
a1 = a2
b1 = b2
colour1 = colour2
tool1 = tool2
jump
to
2

Appendix H. Program example for material removal simulation.

Subroutine for the generating of the raw material.

```
      SUBROUTINE UFUSR( EXITNM, PARAM, RETCOD )
      INTEGER*2    EXITNM
      CHARACTER*132          PARAM
      INTEGER*2    RETCOD

      CHARACTER*132          PRTNAM
      INTEGER*2    ROOP
      REAL*8 CRNER0(3)
      REAL*8 SIZE0(3)
      REAL*8 MXT0(6)
      INTEGER*4    BLK0EID
      REAL*8 CRNER1(3)
      REAL*8 SIZE1(3)
      REAL*8 MXT1(6)
      INTEGER*2    COLOUR
```

C waiting text.

```
      TEXT = 'press Entry Complete to continue.'
      WOPT = 1

      PRTNAM = '!3d-body'

      CALL UF5001(PRTNAM,ROOP)
```

C parameters for the creation of the raw material.

```
      CRNER0(1) =  0.0
      CRNER0(2) =  0.0
      CRNER0(3) =  0.0
      SIZE0(1) = 100.0
      SIZE0(2) = 75.0
      SIZE0(3) = 50.0
      MXT0(1) = 1
      MXT0(2) = 0
      MXT0(3) = 0
      MXT0(4) = 0
      MXT0(5) = 1
      MXT0(6) = 0
```

C creation of the raw material.

```
      CALL UF6500(CRNER0,SIZE0,MXT0,BLK0EID)

      CALL UF1601(TEXT,WOPT)

      RETURN
      END
```

Material removal simulation program example, manually programmed.

```
      SUBROUTINE UFUSR( EXITNM, PARAM, RETCOD )
      INTEGER*2    EXITNM
      CHARACTER*132          PARAM
      INTEGER*2    RETCOD

      CHARACTER*(40)         TITLE
      INTEGER*2    MAX
      INTEGER*4    ARRAY
      INTEGER*2    EIDCNT
      REAL*8       POINT(3)
      INTEGER*2    IRC

      REAL*8 CRNER0(3)
      REAL*8 SIZE0(3)
      REAL*8 MXT0(6)
      INTEGER*4    BLK0EID
      REAL*8 CRNER1(3)
      REAL*8 SIZE1(3)
      REAL*8 MXT1(6)
      INTEGER*4    BLK1EID
      REAL*8 CORI1(3)
      REAL*8 HGT1
      REAL*8 DIA1
      REAL*8 AXS1(3)
      INTEGER*4    C1EID
      REAL*8 CORI2(3)
      REAL*8 HGT2
      REAL*8 DIA2
      REAL*8 AXS2(3)
      INTEGER*4    C2EID
      INTEGER*2    OPTYPE1
      INTEGER*4    TARGS1
      INTEGER*4    TOOLS(3)
      INTEGER*2    TLNUM
      INTEGER*2    MAXRTN
      INTEGER*2    SOERR
      INTEGER*2    RESNUM
      INTEGER*4    RESLST(1)
      CHARACTER*40 ERMESS
      INTEGER*2    RTC
      CHARACTER*(80)         TEXT
      INTEGER*2    WOPT
      INTEGER*2    EID
      INTEGER*2    COLOUR
```

C waiting text.

```
      TEXT = 'press Entry Complete to continue.'
      WOPT = 1
```

C entity colour parameter.

```
      COLOUR = 4
```

```
C definition of the raw material.
      TITLE = 'pick the target solid with the mouse.'
      MAX = 1

      CALL UF1617(TITLE,MAX,BLK0EID,EIDCNT,POINT,IRC)
      CALL UF1601(TEXT,WOPT)

C parameters for the middle part of the cutting tool D=40.
      CRNER1(1) = 0.0
      CRNER1(2) = 0.0
      CRNER1(3) = 90.0
      SIZE1(1) = 0.001
      SIZE1(2) = 0.001
      SIZE1(3) = 0.001
      MXT1(1) = 1
      MXT1(2) = 0
      MXT1(3) = 0
      MXT1(4) = 0
      MXT1(5) = 1
      MXT1(6) = 0
C parameters for the right cylinder of the tool solid.
      CORI1(1) = 0.0
      CORI1(2) = 20.0
      CORI1(3) = 85.0
      HGT1 = 25.0
      DIA1 = 40.0
      AXS1(1) = 0
      AXS1(2) = 0
      AXS1(3) = 1

C parameters for the left cylinder of the tool solid.

      CORI2(1) = 0.0
      CORI2(2) = 20.0
      CORI2(3) = 45.0
      HGT2 = 25.0
      DIA2 = 40.0
      AXS2(1) = 0
      AXS2(2) = 0
      AXS2(3) = 1

C creation of the tool solid.
      CALL UF6502(CORI1,HGT1,DIA1,AXS1,C1EID)
      CALL UF5016(C1EID,COLOUR)
      CALL UF6500(CRNER1,SIZE1,MXT1,BLK1EID)
      CALL UF5016(BLK1EID,COLOUR)
      CALL UF6502(CORI2,HGT2,DIA2,AXS2,C2EID)
      CALL UF5016(C2EID,COLOUR)
      CALL UF1601(TEXT,WOPT)

C parameters for the subtract operation.
      OPTYPE1 = 2
      TARGS1 = BLK0EID
      TOOLS(1) = BLK1EID
      TOOLS(2) = C1EID
```

```
        TOOLS(3) = C2EID
        TLNUM = 3
        MAXRTN = 1

        CALL UF6520(OPTYPE1,TARGS1,TOOLS,TLNUM,MAXRTN,SOERR,RESNUM,RESLST)
        CALL UF1601(TEXT,WOPT)

C parameters for the middle part of the cutting tool D=40.
        CRNER1(1) = 0.0
        CRNER1(2) = 0.0
        CRNER1(3) = 45.0
        SIZE1(1) = 100.0
        SIZE1(2) = 40.0
        SIZE1(3) = 25.0
        MXT1(1) = 1
        MXT1(2) = 0
        MXT1(3) = 0
        MXT1(4) = 0
        MXT1(5) = 1
        MXT1(6) = 0

C parameters for the right cylinder of the tool solid.
        CORI1(1) = 0.0
        CORI1(2) = 20.0
        CORI1(3) = 45.0
        HGT1 = 25.0
        DIA1 = 40.0
        AXS1(1) = 0
        AXS1(2) = 0
        AXS1(3) = 1

C parameters for the left cylinder of the tool solid.
        CORI2(1) = 100.0
        CORI2(2) = 20.0
        CORI2(3) = 45.0
        HGT2 = 25.0
        DIA2 = 40.0
        AXS2(1) = 0
        AXS2(2) = 0
        AXS2(3) = 1

C entity colour paramater.
        COLOUR = 2

C creation of the tool solid.
        CALL UF6502(CORI1,HGT1,DIA1,AXS1,C1EID)
        CALL UF5016(C1EID,COLOUR)
        CALL UF6500(CRNER1,SIZE1,MXT1,BLK1EID)
        CALL UF5016(BLK1EID,COLOUR)
        CALL UF6502(CORI2,HGT2,DIA2,AXS2,C2EID)
        CALL UF5016(C2EID,COLOUR)

C parameters for the subtract operation.
        OPTYPE1 = 2
        TARGS1 = BLK0EID
```

```
      TOOLS(1) = BLK1EID
      TOOLS(2) = C1EID
      TOOLS(3) = C2EID
      TLNUM = 3
      MAXRTN = 1


C subtraction of the first tool solid.
      CALL UF6520(OPTYPE1,TARGS1,TOOLS,TLNUM,MAXRTN,SOERR,RESNUM,RESLST)
      CALL UF1601(TEXT,WOPT)

C parameters for the middle part of the cutting tool D=40.
      CRNER1(1) =  120.0
      CRNER1(2) =  20.0
      CRNER1(3) =  45.0
      SIZE1(1) = 35.0
      SIZE1(2) = 40.0
      SIZE1(3) = 25.0
      MXT1(1) = 0
      MXT1(2) = 1
      MXT1(3) = 0
      MXT1(4) = -1
      MXT1(5) = 0
      MXT1(6) = 0
C parameters for the right cylinder of the tool solid.
      CORI1(1) = 100.0
      CORI1(2) = 20.0
      CORI1(3) = 45.0
      HGT1 = 25.0
      DIA1 = 40.0
      AXS1(1) = 0
      AXS1(2) = 0
      AXS1(3) = 1
C parameters for the left cylinder of the tool solid.
      CORI2(1) = 100.0
      CORI2(2) = 55.0
      CORI2(3) = 45.0
      HGT2 = 25.0
      DIA2 = 40.0
      AXS2(1) = 0
      AXS2(2) = 0
      AXS2(3) = 1

C entity colour parameter.
      COLOUR = 2

C creation of the second tool solid.
      CALL UF6502(CORI1,HGT1,DIA1,AXS1,C1EID)
      CALL UF5016(C1EID,COLOUR)
      CALL UF6500(CRNER1,SIZE1,MXT1,BLK1EID)
      CALL UF5016(BLK1EID,COLOUR)
      CALL UF6502(CORI2,HGT2,DIA2,AXS2,C2EID)
      CALL UF5016(C2EID,COLOUR)

C     parameters for the subtract operation.
```

```
      OPTYPE1 = 2
      TARGS1 = BLK0EID
      TOOLS(1) = BLK1EID
      TOOLS(2) = C1EID
      TOOLS(3) = C2EID
      TLNUM = 3
      MAXRTN = 1

C subtraction of the second tool solid.
      CALL UF6520(OPTYPE1,TARGS1,TOOLS,TLNUM,MAXRTN,SOERR,RESNUM,RESLST)
      CALL UF1601(TEXT,WOPT)

C parameters for the middle part of the cutting tool D=40.
      CRNER1(1) =  100.0
      CRNER1(2) =  75.0
      CRNER1(3) =  45.0
      SIZE1(1) = 100.0
      SIZE1(2) = 40.0
      SIZE1(3) = 25.0
      MXT1(1) = -1
      MXT1(2) = 0
      MXT1(3) = 0
      MXT1(4) = 0
      MXT1(5) = -1
      MXT1(6) = 0
C parameters for the right cylinder of the tool solid.
      CORI1(1) = 100.0
      CORI1(2) = 55.0
      CORI1(3) = 45.0
      HGT1 = 25.0
      DIA1 = 40.0
      AXS1(1) = 0
      AXS1(2) = 0
      AXS1(3) = 1
C parameters for the left cylinder of the tool solid.
      CORI2(1) = 0.0
      CORI2(2) = 55.0
      CORI2(3) = 45.0
      HGT2 = 25.0
      DIA2 = 40.0
      AXS2(1) = 0
      AXS2(2) = 0
      AXS2(3) = 1

C entity colour parameter.
      COLOUR = 2
C creation of the third tool solid.
      CALL UF6502(CORI1,HGT1,DIA1,AXS1,C1EID)
      CALL UF5016(C1EID,COLOUR)
      CALL UF6500(CRNER1,SIZE1,MXT1,BLK1EID)
      CALL UF5016(BLK1EID,COLOUR)
      CALL UF6502(CORI2,HGT2,DIA2,AXS2,C2EID)
      CALL UF5016(C2EID,COLOUR)

C parameters for the subtract operation.
```

```
      OPTYPE1 = 2
      TARGS1 = BLK0EID
      TOOLS(1) = BLK1EID
      TOOLS(2) = C1EID
      TOOLS(3) = C2EID
      TLNUM = 3
      MAXRTN = 1

C third tool solid is subtracted.
      CALL UF6520(OPTYPE1,TARGS1,TOOLS,TLNUM,MAXRTN,SOERR,RESNUM,RESLST)
      CALL UF1601(TEXT,WOPT)

C parameters for the middle part of the cutting tool D=40.
      CRNER1(1) = 0.0
      CRNER1(2) = 0.0
      CRNER1(3) = 40.0
      SIZE1(1) = 100.0
      SIZE1(2) = 40.0
      SIZE1(3) = 25.0
      MXT1(1) = 1
      MXT1(2) = 0
      MXT1(3) = 0
      MXT1(4) = 0
      MXT1(5) = 1
      MXT1(6) = 0
C parameters for the right cylinder of the tool solid.
      CORI1(1) = 0.0
      CORI1(2) = 20.0
      CORI1(3) = 40.0
      HGT1 = 25.0
      DIA1 = 40.0
      AXS1(1) = 0
      AXS1(2) = 0
      AXS1(3) = 1

C parameters for the left cylinder of the tool solid.
      CORI2(1) = 100.0
      CORI2(2) = 20.0
      CORI2(3) = 40.0
      HGT2 = 25.0
      DIA2 = 40.0
      AXS2(1) = 0
      AXS2(2) = 0
      AXS2(3) = 1
C entity colour parameter.
      COLOUR = 2


C creation of the fourth tool solid.
      CALL UF6502(CORI1,HGT1,DIA1,AXS1,C1EID)
      CALL UF5016(C1EID,COLOUR)
      CALL UF6500(CRNER1,SIZE1,MXT1,BLK1EID)
      CALL UF5016(BLK1EID,COLOUR)
      CALL UF6502(CORI2,HGT2,DIA2,AXS2,C2EID)
      CALL UF5016(C2EID,COLOUR)
```

```
C parameters for the subtract operation.
      OPTYPE1 = 2
      TARGS1 = BLK0EID
      TOOLS(1) = BLK1EID
      TOOLS(2) = C1EID
      TOOLS(3) = C2EID
      TLNUM = 3
      MAXRTN = 1


C subtraction of the fourth tool solid.
      CALL UF6520(OPTYPE1,TARGS1,TOOLS,TLNUM,MAXRTN,SOERR,RESNUM,RESLST)
      CALL UF1601(TEXT,WOPT)

C parameters for the middle part of the cutting tool D=40.
      CRNER1(1) =  120.0
      CRNER1(2) =  20.0
      CRNER1(3) =  40.0
      SIZE1(1) = 35.0
      SIZE1(2) = 40.0
      SIZE1(3) = 25.0
      MXT1(1) = 0
      MXT1(2) = 1
      MXT1(3) = 0
      MXT1(4) = -1
      MXT1(5) = 0
      MXT1(6) = 0
C parameters for the right cylinder of the tool solid.
      CORI1(1) = 100.0
      CORI1(2) = 20.0
      CORI1(3) = 40.0
      HGT1 = 25.0
      DIA1 = 40.0
      AXS1(1) = 0
      AXS1(2) = 0
      AXS1(3) = 1
C parameters for the left cylinder of the tool solid.
      CORI2(1) = 100.0
      CORI2(2) = 55.0
      CORI2(3) = 40.0
      HGT2 = 25.0
      DIA2 = 40.0
      AXS2(1) = 0
      AXS2(2) = 0
      AXS2(3) = 1
C entity colour parameter.
      COLOUR = 2

C creation of the fifth tool solid.
      CALL UF6502(CORI1,HGT1,DIA1,AXS1,C1EID)
      CALL UF5016(C1EID,COLOUR)
      CALL UF6500(CRNER1,SIZE1,MXT1,BLK1EID)
      CALL UF5016(BLK1EID,COLOUR)
      CALL UF6502(CORI2,HGT2,DIA2,AXS2,C2EID)
      CALL UF5016(C2EID,COLOUR)
```

```
C parameters for the subtract operation.
        OPTYPE1 = 2
        TARGS1 = BLK0EID
        TOOLS(1) = BLK1EID
        TOOLS(2) = C1EID
        TOOLS(3) = C2EID
        TLNUM = 3
        MAXRTN = 1

C subtraction of the fifth tool solid.
        CALL UF6520(OPTYPE1,TARGS1,TOOLS,TLNUM,MAXRTN,SOERR,RESNUM,RESLST)
        CALL UF1601(TEXT,WOPT)

C parameters for the middle part of the cutting tool D=40.
        CRNER1(1) =  100.0
        CRNER1(2) =  75.0
        CRNER1(3) =  40.0
        SIZE1(1) = 100.0
        SIZE1(2) = 40.0
        SIZE1(3) = 25.0
        MXT1(1) = -1
        MXT1(2) = 0
        MXT1(3) = 0
        MXT1(4) = 0
        MXT1(5) = -1
        MXT1(6) = 0
C parameters for the right cylinder of the tool solid.
        CORI1(1) = 100.0
        CORI1(2) = 55.0
        CORI1(3) = 40.0
        HGT1 = 25.0
        DIA1 = 40.0
        AXS1(1) = 0
        AXS1(2) = 0
        AXS1(3) = 1
C parameters for the left cylinder of the tool solid.
        CORI2(1) = 0.0
        CORI2(2) = 55.0
        CORI2(3) = 40.0
        HGT2 = 25.0
        DIA2 = 40.0
        AXS2(1) = 0
        AXS2(2) = 0
        AXS2(3) = 1
C entity colour parameter.
        COLOUR = 2

C creation of the sixth tool solid.
        CALL UF6502(CORI1,HGT1,DIA1,AXS1,C1EID)
        CALL UF5016(C1EID,COLOUR)
        CALL UF6500(CRNER1,SIZE1,MXT1,BLK1EID)
        CALL UF5016(BLK1EID,COLOUR)
        CALL UF6502(CORI2,HGT2,DIA2,AXS2,C2EID)
        CALL UF5016(C2EID,COLOUR)
```

```
C parameters for the subtract operation.
      OPTYPE1 = 2
      TARGS1 = BLK0EID
      TOOLS(1) = BLK1EID
      TOOLS(2) = C1EID
      TOOLS(3) = C2EID
      TLNUM = 3
      MAXRTN = 1

C subtraction of the sixth tool solid.
      CALL UF6520(OPTYPE1,TARGS1,TOOLS,TLNUM,MAXRTN,SOERR,RESNUM,RESLST)
      CALL UF1601(TEXT,WOPT)

C parameters for the middle part of the cutting tool D=40.
      CRNER1(1) =  0.0
      CRNER1(2) =  0.0
      CRNER1(3) =  90.0
      SIZE1(1) = 0.001
      SIZE1(2) = 0.001
      SIZE1(3) = 0.001
      MXT1(1) = 1
      MXT1(2) = 0
      MXT1(3) = 0
      MXT1(4) = 0
      MXT1(5) = 1
      MXT1(6) = 0
C parameters for the right cylinder of the tool solid.
      CORI1(1) = 0.0
      CORI1(2) = 55.0
      CORI1(3) = 40.0
      HGT1 = 25.0
      DIA1 = 40.0
      AXS1(1) = 0
      AXS1(2) = 0
      AXS1(3) = 1

C parameters for the left cylinder of the tool solid.
      CORI2(1) = 0.0
      CORI2(2) = 55.0
      CORI2(3) = 80.0
      HGT2 = 25.0
      DIA2 = 40.0
      AXS2(1) = 0
      AXS2(2) = 0
      AXS2(3) = 1
C entity colour parameter.
      COLOUR = 4

C creation of the tool solid.
      CALL UF6502(CORI1,HGT1,DIA1,AXS1,C1EID)
      CALL UF5016(C1EID,COLOUR)
      CALL UF6500(CRNER1,SIZE1,MXT1,BLK1EID)
      CALL UF5016(BLK1EID,COLOUR)
      CALL UF6502(CORI2,HGT2,DIA2,AXS2,C2EID)
      CALL UF5016(C2EID,COLOUR)
```

```
        CALL UF1601(TEXT,WOPT)
```

C parameters for the subtract operation.

```
        OPTYPE1 = 2
        TARGS1 = BLK0EID
        TOOLS(1) = BLK1EID
        TOOLS(2) = C1EID
        TOOLS(3) = C2EID
        TLNUM = 3
        MAXRTN = 1

        CALL UF6520(OPTYPE1,TARGS1,TOOLS,TLNUM,MAXRTN,SOERR,RESNUM,RESLST)
        CALL UF1601(TEXT,WOPT)

        RETURN
        END
```

Appendix I. MAHO information.

MAHO MH 700 S

De MH 700S is een universele frees en boormachine met een universele gereedschapwisselaar, voorzien van een 5-assige contourbesturing, de CNC 432 van de firma Philips. Voor het positioneren in de assen X, Y en Z beschikt de machine over lineaire meetsystemen en voor de assen A en B over rotatie meetsystemen.

Technische gegevens:

Machine bereik:

langs	X-as	700mm
vertikaal	Y-as	500mm
dwars	Z-as	600mm
rondtafel	B-as	n * 360 gr.
zwenkas	A-as	-20 tot +45 gr.

Hoofdspil :

aandrijving: regelbare wisselstroommotor met een vermogen van 10 kW
toerentallenreeks: 20-6300 omw/min
toerentallen: trappeloos regelbaar binnen 4 reeksen.

Freesspillen:

automatisch weg te draaien vertikale freeskop
gereedschapsconus ISO 40
gereedschapsopspanning hydro-mechanisch

Aanzetbewegingen:

individuele gelijkstroomaandrijving, per as regelbaar.
aanzetsnelheid in X,Y en Z: 1-4000 mm/min
A en B-as: 1-3000 graden/min

IJlgang:

X en Z-as	12 m/min
Y-as	10 m/min
B-as	15 omw/min
A-as	16 gr/sec

Meetsysteem:

X, Y en Z-as:	inkrementeel - lineair.
A en B-as:	inkrementeel - roterend.
oplossend vermogen:	X, Y en Z-as 0.001 mm.
	A en B-as 0.001 graad.

Gereedschapwisselaar:

aantal gereedschappen in het magazijn: 36

Tafel:

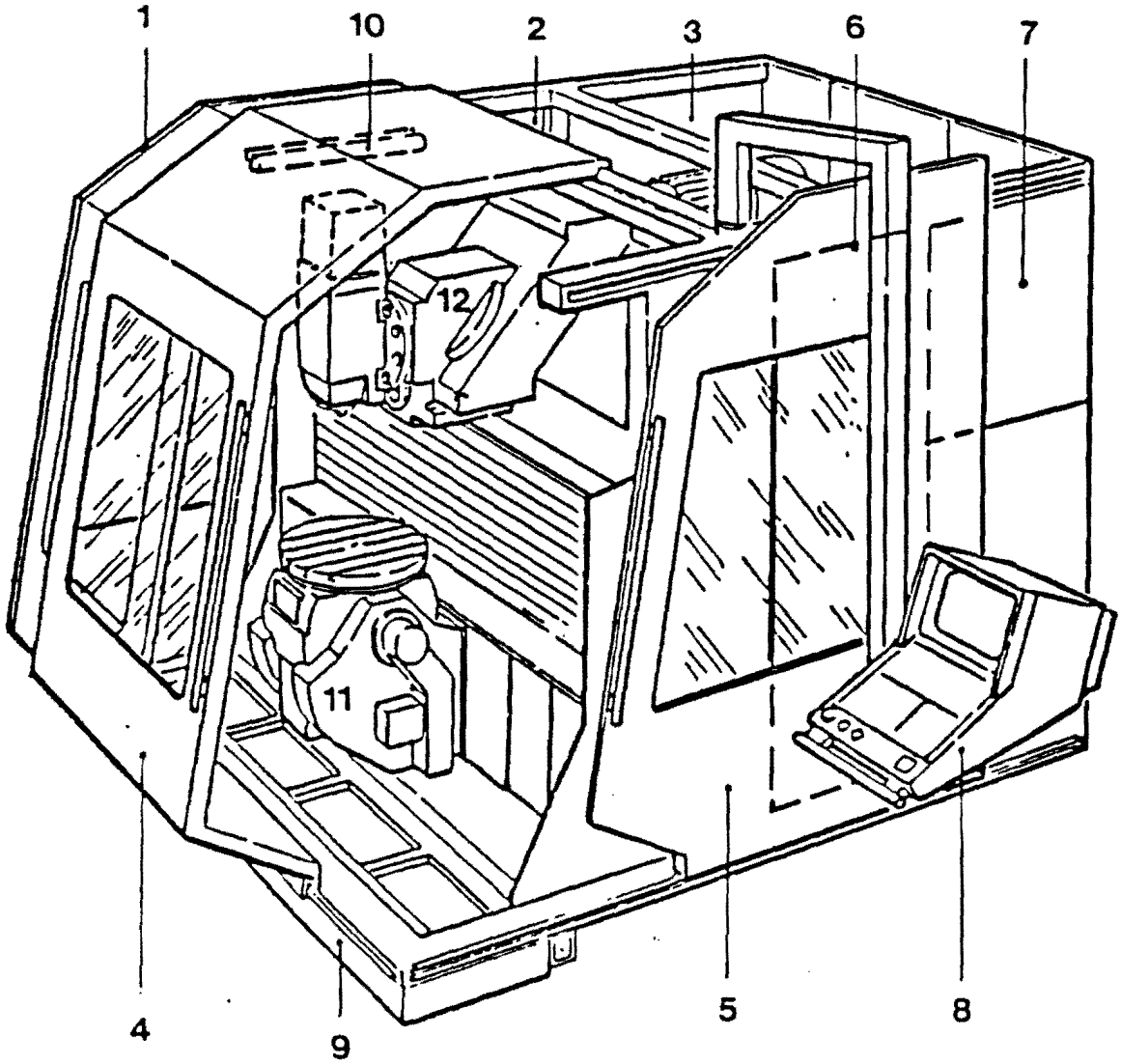
zwenk-rond-tafel, diameter 520mm

Besturing:

Philips CNC 432 contourbesturing met beeldscherm

Gewicht:

machine compleet ongeveer 8500 kg.



De MAHO 700 S

1. Schuifdeur rechts
2. Toegangsluik hydrauliek
3. Toegangsluik ventielblok hydrauliek, pneumatiek
4. Schuifdeur voor
5. Schuifdeur links
6. Toegangsluik koelmiddelleidingen en gereedschapwisselaar
7. Toegangsluik gereedschapmagazijn en centraalsmering
8. Bedieningspaneel en beeldscherm
9. Koelvloeistof reservoir
10. Verlichting
11. Kantelbare rondtafel
12. Kantelbare vertikale kop

Programmeersleutel voor MAHO CNC 432

ADRES	CODE	BETEKENIS EN VERKLARING
% PM		programmabegin en geheugentoevoeging
N N	9001-9999999 1-8999	deel- en onderprogrammanummer regelnummer
G	0 * 1 2 3	ijlgang lineaire interpolatie cirkelinterpolatie, rechtsom cirkelinterpolatie, linksom
G	4 **	pauze (0.1 - 983 seconde)
G	11 **	polarkoördinaten, hoekafronding, fase-overgang
G	14 **	sprongbevel en herhaalfunktie
G	17 * 18 19	vlakkeuze XY, horizontaal vlakkeuze XZ, vertikaal vlakkeuze YZ, horizontaal 90 gedraaid
G	22 ** 29 **	onderprogramma oproep beperkt sprongbevel in onderprogramma
G	40 * 41 42 43 44	geen radiuskorrektie radiuskorrektie, links radiuskorrektie, rechts radiuskorrektie, tot radiuskorrektie, over
G	51 52	wissen van G-52 verschuiving RESET AXIS aktiveren
G	53 * 54 55 56 57 58 59	nul-punt verschuiving wissen nul-punt verschuiving 1 nul-punt verschuiving 2 nul-punt verschuiving 3 nul-punt verschuiving 4 nul-punt verschuiving 5 nul-punt verschuiving 6
G	70 71 *	inch maat ingavesysteem metrisch maat ingavensysteem
G	72 * 73	geen spiegelbeeld bewerking spiegelbeeldbewerking
G	77 ** 78 **	gatencirkeldefinitie puntdefinitie

ADRES	CODE	BETEKENIS EN VERKLARING
G	79 **	cyclusoproep
G	81	boorcyclus
	83	diepboorcyclus
	84	tapcyclus
	85	ruimcyclus
	86	uitdraaicyclus
	87	kamerfreescyclus
	88	spiebaanfreescyclus
	89	cirkelkamerfreescyclus
G	90 *	absoluutmaat programmeren
	91	inkrementeel programmeren
G	92	nulpuntverschuiving inkrementeel
	93	nulpuntverschuiving absoluut
G	94 *	voeding in mm/min
	95	voeding in mm/omw
X	+/- 9999.999	weginformatie in mm.
Y	+/- 9999.999	weginformatie in mm.
Z	+/- 9999.999	weginformatie in mm.
A	+/- 9999.999	weginformatie in graden
B	+/- 9999,999	weginformatie in graden
R	+/- 9999.999	cirkelradius in mm.
I	+/- 9999.999	cirkelmiddelpunt in X-richting
J	+/- 9999.999	cirkelmiddelpunt in Y-richting
K	+/- 9999.999	cirkelmiddelpunt in Z-richting
D	0 - 360	hoekpositie hoofdspil
P	0 - 99	puntdefinitie
F	0 - 4000	voeding in mm/omw. of mm/min.
S	20 - 6300	hoofdspiltoerental in omw/min.
T	0 - 99	gereedschapnummer
M	0 **	programma-stop
	3	hoofdspil rechtsom
	4	hoofdspil linksom
	5	hoofdspil stop
	6 **	gereedschapwissel met terugtrekking
	8	koelmiddel aan
	9	koelmiddel uit
	13	spil rechtsom koelmiddel aan
	14	spil linksom koelmiddel aan
	19	georienteerde spindelstop
	30 **	einde programma
	53	vertikale kop wegdraaien (G17)
	54	vertikale kop indraaien (G18)
	66 **	gereedschapwisseling
E	0 - 99	parameters in het onderprogramma

Tekenverklaring: * = wordt aktief bij inschakelen van de besturing
** = alleen per regel werkzaam

Appendix J. ROBOTICS-software overview.

From: "The off-line programming of welded products fith the Robotics software" by M.C. Willems [14].

The ROBOTICS software is a set of separate programs, developed by McDonnell Douglas, for the development of a computer based simulation- and off-line programming environment. The ROBOTICS system enables the user to effectively design, model, pre-analyze, implement, modify and adjust manufacturing robotic work cells to accommodate the use and reuse of a wide variety of robots. To provide the functions, the ROBOTICS software consists of a number of separate programs communicating with each other via a small number of files. These programs are :

- BUILD
- PLACE (SIMULATIONS)
- COMMAND
- ADJUST
- CTA Cycle Time Analyzer.

Though these separate programs make up the complete ROBOTICS package, it is not necessary to use them all in a ROBOTICS session. The way the programs cooperate with each other can be seen in the ROBOTICS system overview.

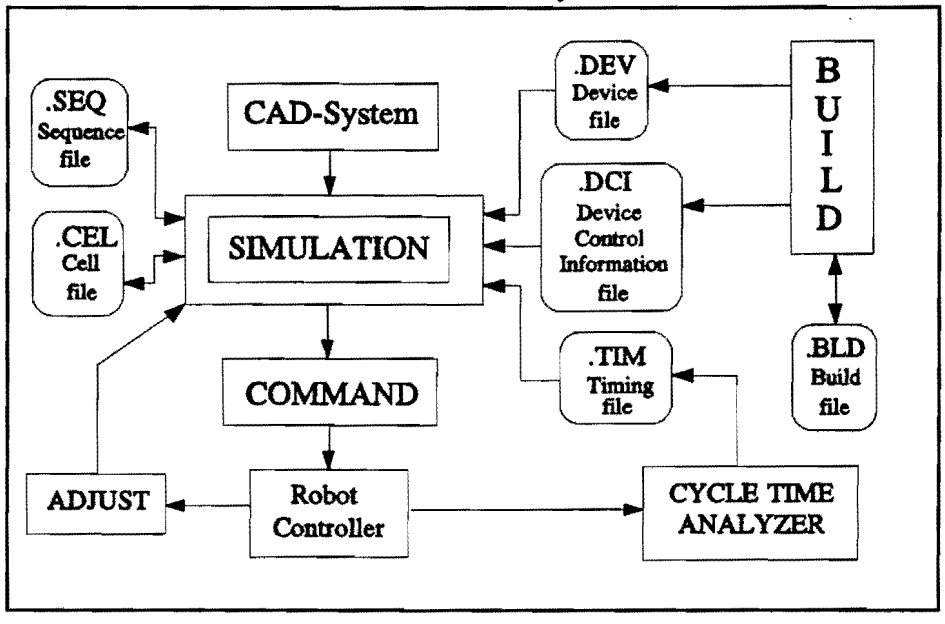


Figure 1. ROBOTICS system overview.

In BUILD you can create a device, such as robots, mechanisms, positioners, etc. Using BUILD, the geometric model of a device is automatically combined with its kinematic description. Build directs you to define the kinematic model of a device, which is used by PLACE to drive the simulation.

The data output file from BUILD eliminates the need to perform custom kinematic analysis. BUILD can generate the kinematic equations for devices which have up to six degrees of freedom. BUILD creates three files which are used by PLACE (as well as COMMAND, CTA and ADJUST) to describe the device.- The BUILD file (BLD) contains the basic device description.
- The DEVICE file (DEV) contains the connection tree which describes the device along with the link names and the associated part names.
- The device control information file (DCI) defines the device characteristics such as kinematic attributes, allowable motion modes, maximum joint speed and accelerations, etc.

PLACE (simulation), Positioner Layout and Cell Evaluator system, is a software module which is designed to create, analyze and modify robot cells and to simulate device movements through the use of high speed vector refresh and a raster colour graphics display station. PLACE uses wireframe and facet-faced graphic display models to represent robots, equipment, work pieces and tools (within a manufacturing cell). The user can easily position the graphic models within a cell, either individually, as in case of a workpiece, or together as in case of a robot with manipulator. PLACE is equipped with a lot of features to work with, such as collision detection (visual and automatically), tracking, dimension analysis, hierarchical definition of connected parts, continuous readout of joint angle data, hardware controlled dynamic 3-D scaling, translating and rotating of a view, an expanding library of more than 130 of the most common robots, etc.

COMMAND is a software module used for programming robots off-line. It is used in conjunction with PLACE and a specific robot translator to generate a complete robot program, which can be loaded and executed on a particular robot. The user directs COMMAND to merge a PLACE sequence. This sequence contains all the robot movements created in PLACE and eventually commands to change speeds, open or close tools, I/O-communication, etc. Together with the sequence file another file is merged: the user file (USR). The USR-file is created with a standard text editor and includes the robot program skeleton: begin and end statements, sequence placement and additional commands in the robot native language. It is also possible in this USR-file to define subroutines which are used in the sequence.

ADJUST and CTA are additional programs which are not used frequently. The Cycle Time Analyzer is used to predict accurate cycle times for a device. With CTA the total work area, for every axis, for the whole speed range of a real robot is examined and stored in a file. This file is then connected to the robot model (in the software package). Every time the robot model moves during a simulation the dynamics of the matching move of the real robot is checked for accurate cycle time prediction. ADJUST is used to adjust the software model of the cell to the real cell. ADJUST modifies the matrices of PLACE work cell-frames to match the locations of the corresponding objects in the real work cell. ADJUST uses the actual robot controller position readings as the robot lightly touches or aligns with key work cell objects to compute matrix values for PLACE frames. In many cases ADJUST can put the PLACE theoretical cell into precise agreement with the real work cell.

Appendix K. The place cell FRB1.

```
;****** PLACE Release 9.0 ******
FRAMES
WORLD      WORLD
   1.0000   0.0000   0.0000   0.0000
   0.0000   1.0000   0.0000   0.0000
   0.0000   0.0000   1.0000   0.0000
FREESB0    WORLD
   1.0000   0.0000   0.0000   0.0000
   0.0000   1.0000   0.0000   0.0000
   0.0000   0.0000   1.0000   0.0000
FREESB1    FREESB0
   0.0000   0.0000   1.0000  39.3701
   0.0000   1.0000   0.0000  20.6693
  -1.0000   0.0000   0.0000  58.2677
FREESB2    FREESB1
   1.0000   0.0000   0.0000   0.0000
   0.0000   0.0000  -1.0000   0.0000
   0.0000   1.0000   0.0000   0.0000
FREESTAF0  WORLD
   1.0000   0.0000   0.0000   0.0000
   0.0000   1.0000   0.0000   5.9055
   0.0000   0.0000   1.0000   3.9370
FREESTAF1  FREESTAF0
   1.0000   0.0000   0.0000  19.6850
   0.0000   1.0000   0.0000 -23.6220
   0.0000   0.0000   1.0000  24.8032
FREESTAF2  FREESTAF1
   1.0000   0.0000   0.0000   9.8425
   0.0000   1.0000   0.0000  22.6378
   0.0000   0.0000   1.0000   0.0000
FREESTAF3  FREESTAF2
   1.0000   0.0000   0.0000   9.8425
   0.0000   1.0000   0.0000   1.9685
   0.0000   0.0000   1.0000  15.7480
FREESTAF4  FREESTAF3
   0.0000   0.0000  -1.0000   0.0000
   0.0000   1.0000   0.0000   5.9055
   1.0000   0.0000   0.0000   9.8425
LEEG       FREESTAF4
   1.0000   0.0000   0.0000   0.0000
   0.0000   1.0000   0.0000   0.0000
   0.0000   0.0000   1.0000   0.0000
END/FRAMES
CONTROL
FREESB    DEV FREESB    FREESB2   FREESB0
FREESTAF6 DEV FREESTAF6 LEEG      FREESTAF0
END/CONTROL
DISPLAY
FREESB0    FREESB0     FORESTGREEN,R(0.3137),G(0.6235),B(0.4118) TOLER(0.0500)
TRANSP(1.0000)
   1.0000   0.0000   0.0000   0.0000
   0.0000   1.0000   0.0000   0.0000
   0.0000   0.0000   1.0000   0.0000
```

```
FREESB1   FREESB1    LIGHTBLUE,R(0.6902),G(0.8863),B(1.0000) TOLER(0.0500)
TRANSP(1.0000)
   1.0000   0.0000   0.0000   0.0000
   0.0000   1.0000   0.0000   0.0000
   0.0000   0.0000   1.0000   0.0000
FREESTAF0 FREESTAF0  LIMEGREEN,R(0.0000),G(0.6863),B(0.0784) TOLER(0.0500)
TRANSP(1.0000)
   1.0000   0.0000   0.0000   0.0000
   0.0000   1.0000   0.0000   0.0000
   0.0000   0.0000   1.0000   0.0000
FREESTAF1 FREESTAF1  ORANGE,R(1.0000),G(0.5294),B(0.0000) TOLER(0.0500)
TRANSP(1.0000)
   1.0000   0.0000   0.0000   0.0000
   0.0000   1.0000   0.0000   0.0000
   0.0000   0.0000   1.0000   0.0000
FREESTAF2 FREESTAF2  LIMEGREEN,R(0.0000),G(0.6863),B(0.0784) TOLER(0.0500)
TRANSP(1.0000)
   1.0000   0.0000   0.0000   0.0000
   0.0000   1.0000   0.0000   0.0000
   0.0000   0.0000   1.0000   0.0000
FREESTAF3 FREESTAF3  ORANGE,R(1.0000),G(0.5294),B(0.0000) TOLER(0.0500)
TRANSP(1.0000)
   1.0000   0.0000   0.0000   0.0000
   0.0000   1.0000   0.0000   0.0000
   0.0000   0.0000   1.0000   0.0000
FREESTAF4 FREESTAF4  CADETBLUE,R(0.3725),G(0.5725),B(0.6196) TOLER(0.0500)
TRANSP(1.0000)
   1.0000   0.0000   0.0000   0.0000
   0.0000   1.0000   0.0000   0.0000
   0.0000   0.0000   1.0000   0.0000
END/DISPLAY
TPOINTS
FREESB2     WHITE,R(1.0000),G(1.0000),B(1.0000)
1
TP1
   0.0000   0.0000   0.0000  1.0000  0.0000  0.0000  0.0000  1.0000  0.0000
LEEG        WHITE,R(1.0000),G(1.0000),B(1.0000)
1
TP1
   0.0000   0.0000   0.0000  1.0000  0.0000  0.0000  0.0000  1.0000  0.0000
END/TPOINTS
```

Appendix L. The build files.

```
;****** BUILD Release 9.0 ******

DEVICE NAME = FREESB
DEVICE TYPE = ROBOT
UNITS = MILLIMETERS

*******

Constant
Translation along
Z axis
Amount =  1480.0000 (MM)

*******

Constant
Translation along
Y axis
Amount =   525.0000 (MM)

*******

Constant
Translation along
X axis
Amount =  1000.0000 (MM)

*******

Constant
Rotation about
Y axis
Amount =    90.0000 (DEG)

*******

Variable
Translation along
-Z axis
Joint Name = FREESB1
Joint Constraints --
High Value =  1000.0000 (MM)   Low Value = -1000.0000 (MM)
Home Position =     0.0000 (MM)
Joint Speed =   100.0000 (MM/SEC)
Joint Acceleration =    10.0000 (MM/SEC/SEC)
END OF LINK

*******
Constant
Rotation about
X axis
Amount =    90.0000 (DEG)
```

****

Variable
Rotation about
Z axis
Joint Name = FREESB2
Joint Constraints --
High Value = 360.0000 (DEG) Low Value = -360.0000 (DEG)
Home Position = 0.0000 (DEG)
Joint Speed = 100.0000 (DEG/SEC)
Joint Acceleration = 10.0000 (DEG/SEC/SEC)
END OF LINK
END OF DEVICE

****

INVERSE KINEMATICS DATA --
SOURCE -- STANDARD

****

CONFIGURATIONS --
=
=
SHORT REACH OF JT 1 = SHORT REACH OF JT 1
LONG REACH OF JT 1 = LONG REACH OF JT 1
=
=
Initial Configuration = 1

****

MOTION TYPES --
NUMBER OF TYPES = 2
STRAIGHT
SLEW
HOME MOTION TYPE = SLEW

****

TOOL COORDINATE SYSTEM =
MAX TOOL SPEED = 0.0000 (MM/SEC)
MAX TOOL ACCEL = 0.0000 (MM/SEC/SEC)

****

COORDINATE SYSTEM REPRESENTATIONS --
NUMBER OF COORDINATE SYSTEMS = 1
JOINTSM = MM

****

World to Robot Base Transformation --
Translations --
0.0000 0.0000 0.0000 (MM)

Rotations --
0.0000 0.0000 0.0000 (DEG)

****

Link Names --
Number of Links = 3
1. FREESB0
2. FREESB1
3. FREESB2

****

Part Names --
Number of Parts = 3
1. FREESB0
2. FREESB1
3.

****

Ground Link Name = FREESB0

```
;****** BUILD Release 9.0 ******

DEVICE NAME = FREESB40
DEVICE TYPE = ROBOT
UNITS = MILLIMETERS

*******

Constant
Translation along
Z axis
Amount =  1480.0000 (MM)

*******

Constant
Translation along
Y axis
Amount =   525.0000 (MM)

*******

Constant
Translation along
X axis
Amount =  1000.0000 (MM)

*******

Constant
Rotation about
Y axis
Amount =    90.0000 (DEG)

*******

Variable
Translation along
-Z axis
Joint Name = FREESB1
Joint Constraints --
High Value =  1000.0000 (MM)   Low Value = -1000.0000 (MM)
Home Position =     0.0000 (MM)
Joint Speed =   100.0000 (MM/SEC)
Joint Acceleration =    10.0000 (MM/SEC/SEC)
END OF LINK

*******

Constant
Rotation about
X axis
Amount =    90.0000 (DEG)

*******
```

Variable
Rotation about
Z axis
Joint Name = FREESB2
Joint Constraints --
High Value = 360.0000 (DEG) Low Value = -360.0000 (DEG)
Home Position = 0.0000 (DEG)
Joint Speed = 100.0000 (DEG/SEC)
Joint Acceleration = 10.0000 (DEG/SEC/SEC)
END OF LINK
END OF DEVICE

****

INVERSE KINEMATICS DATA --
SOURCE -- STANDARD

****

CONFIGURATIONS --
=
=
SHORT REACH OF JT 1 = SHORT REACH OF JT 1
LONG REACH OF JT 1 = LONG REACH OF JT 1
=
=
Initial Configuration = 1

****

MOTION TYPES --
NUMBER OF TYPES = 2
STRAIGHT
SLEW
HOME MOTION TYPE = SLEW

****

TOOL COORDINATE SYSTEM =
MAX TOOL SPEED = 0.0000 (MM/SEC)
MAX TOOL ACCEL = 0.0000 (MM/SEC/SEC)

****

COORDINATE SYSTEM REPRESENTATIONS --
NUMBER OF COORDINATE SYSTEMS = 1
JOINTSM = MM

****

World to Robot Base Transformation --
Translations --
0.0000 0.0000 0.0000 (MM)
Rotations --
0.0000 0.0000 0.0000 (DEG)

****

Link Names --
Number of Links = 3
1. FREESB0
2. FREES4
3. FREESB2

****

Part Names --
Number of Parts = 3
1. FREESB0
2. FREES4
3.

****

Ground Link Name = FREESB0

```
;****** BUILD Release 9.0 ******

DEVICE NAME = FREESTAF6
DEVICE TYPE = ROBOT
UNITS = MILLIMETERS

*******

Constant
Translation along
Z axis
Amount =   730.0000 (MM)

*******

Constant
Translation along
Y axis
Amount =  -600.0000 (MM)

*******

Constant
Translation along
X axis
Amount =   500.0000 (MM)

*******

Variable
Translation along
-Z axis
Joint Name = FREEST0
Joint Constraints --
High Value =  1000.0000 (MM)   Low Value = -1000.0000 (MM)
Home Position =   200.0000 (MM)
Joint Speed =   100.0000 (MM/SEC)
Joint Acceleration =    10.0000 (MM/SEC/SEC)
END OF LINK

*******

Constant
Translation along
X axis
Amount =   250.0000 (MM)

*******

Constant
Translation along
Y axis
Amount =   775.0000 (MM)

*******
```

Variable
Translation along
Y axis
Joint Name = FREEST2
Joint Constraints --
High Value = 1000.0000 (MM) Low Value = -1000.0000 (MM)
Home Position = 0.0000 (MM)
Joint Speed = 100.0000 (MM/SEC)
Joint Acceleration = 10.0000 (MM/SEC/SEC)
END OF LINK

****

Constant
Translation along
Y axis
Amount = 50.0000 (MM)

****

Constant
Translation along
Z axis
Amount = 400.0000 (MM)

****

Constant
Translation along
X axis
Amount = 250.0000 (MM)

****

Variable
Rotation about
Y axis
Joint Name = FREEST3
Joint Constraints --
High Value = 150.0000 (DEG) Low Value = -150.0000 (DEG)
Home Position = 0.0000 (DEG)
Joint Speed = 100.0000 (DEG/SEC)
Joint Acceleration = 10.0000 (DEG/SEC/SEC)
END OF LINK

****

Constant
Translation along
Y axis
Amount = 150.0000 (MM)

****

Constant
Translation along
Z axis
Amount = 250.0000 (MM)

****

Constant
Rotation about
Y axis
Amount = -90.0000 (DEG)

****

Variable
Rotation about
X axis
Joint Name = FREEST4
Joint Constraints --
High Value = 180.0000 (DEG) Low Value = -180.0000 (DEG)
Home Position = 0.0000 (DEG)
Joint Speed = 100.0000 (DEG/SEC)
Joint Acceleration = 10.0000 (DEG/SEC/SEC)
END OF LINK

****

Constant
Translation along
X axis
Amount = 0.0000 (MM)

****

Constant
Translation along
Y axis
Amount = 0.0000 (MM)

****

Constant
Translation along
Z axis
Amount = 0.0000 (MM)

****

Constant
Rotation about
Y axis
Amount = 0.0000 (DEG)

****

Variable
Translation along
Y axis
Joint Name = LEEG
Joint Constraints --
High Value = 0.0000 (MM) Low Value = 0.0000 (MM)
Home Position = 0.0000 (MM)
Joint Speed = 10.0000 (MM/SEC)
Joint Acceleration = 10.0000 (MM/SEC/SEC)
END OF LINK
END OF DEVICE

****

INVERSE KINEMATICS DATA --
SOURCE -- STANDARD

****

CONFIGURATIONS --
=
=
=
=
JT 4 BETWEEN 0.0 & 180.0 DEG = JT 4 BETWEEN 0.0 & 180.0 DEG
JT 4 BETWEEN 180.0 & 0.0 DEG = JT 4 BETWEEN 180.0 & 0.0 DEG
Automatic wrist configuration
Initial Configuration = 2

****

MOTION TYPES --
NUMBER OF TYPES = 2
STRAIGHT
SLEW
HOME MOTION TYPE = SLEW

****

TOOL COORDINATE SYSTEM =
MAX TOOL SPEED = 0.0000 (MM/SEC)
MAX TOOL ACCEL = 0.0000 (MM/SEC/SEC)

****

COORDINATE SYSTEM REPRESENTATIONS --
NUMBER OF COORDINATE SYSTEMS = 1
JOINTSM = MMDEG

****

World to Robot Base Transformation --
Translations --
0.0000 150.0000 100.0000 (MM)
Rotations --
0.0000 0.0000 0.0000 (DEG)

****

Link Names --
Number of Links = 6
1. FREESTAF0
2. FREESTAF1
3. FREESTAF2
4. FREESTAF3
5. FREESTAF4
6. LEEG

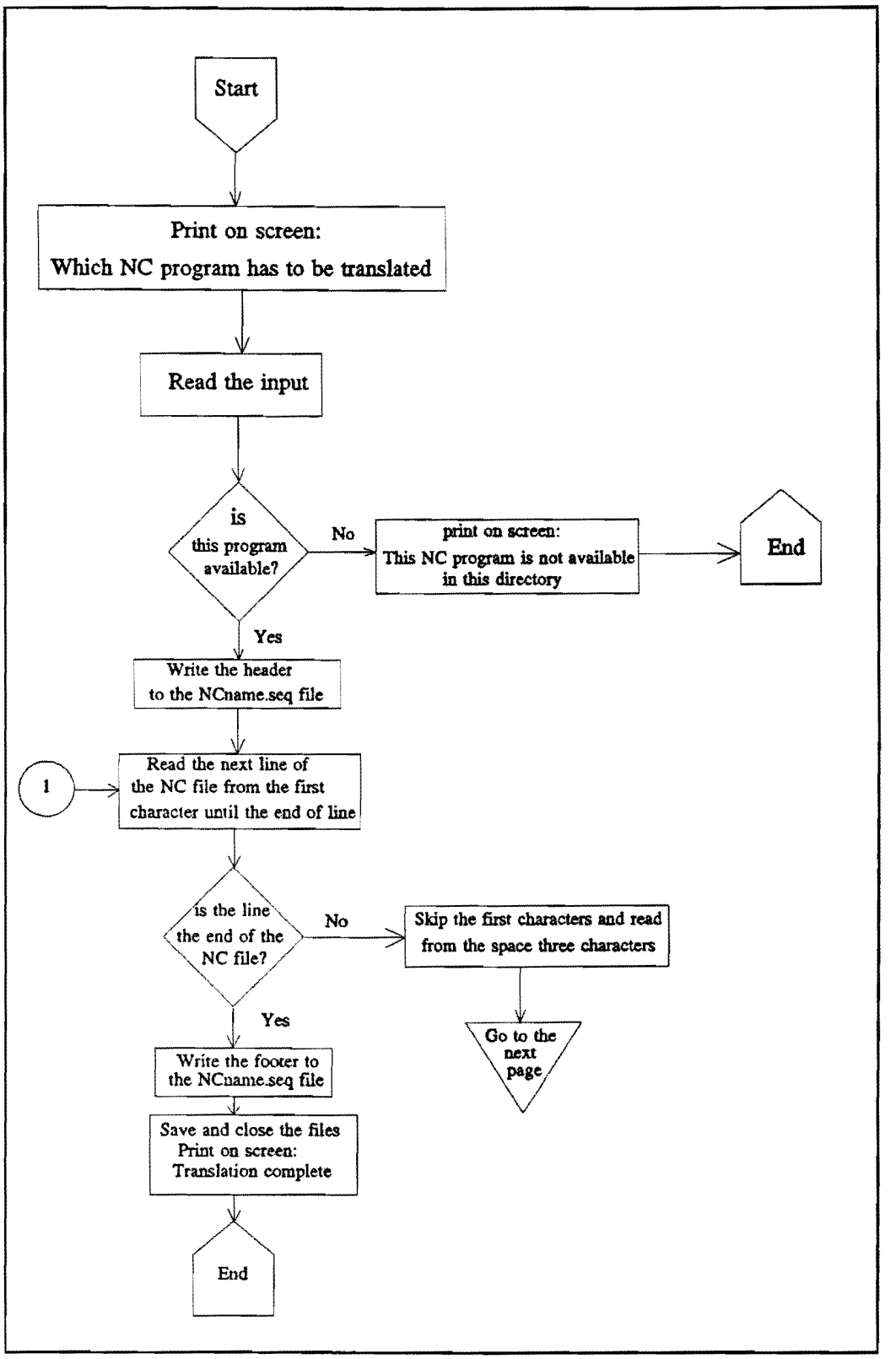
****

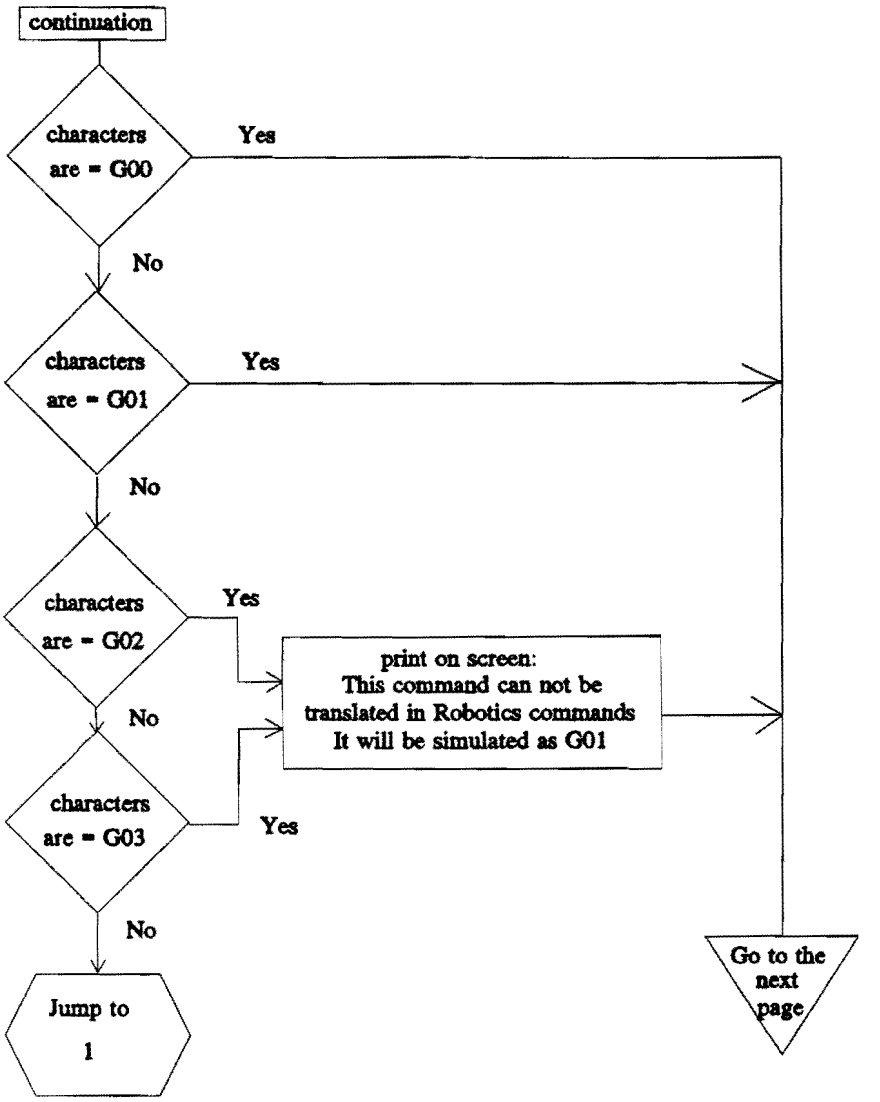
Part Names --
Number of Parts = 6
1. FREESTAF0
2. FREESTAF1
3. FREESTAF2
4. FREESTAF3
5. FREESTAF4
6.

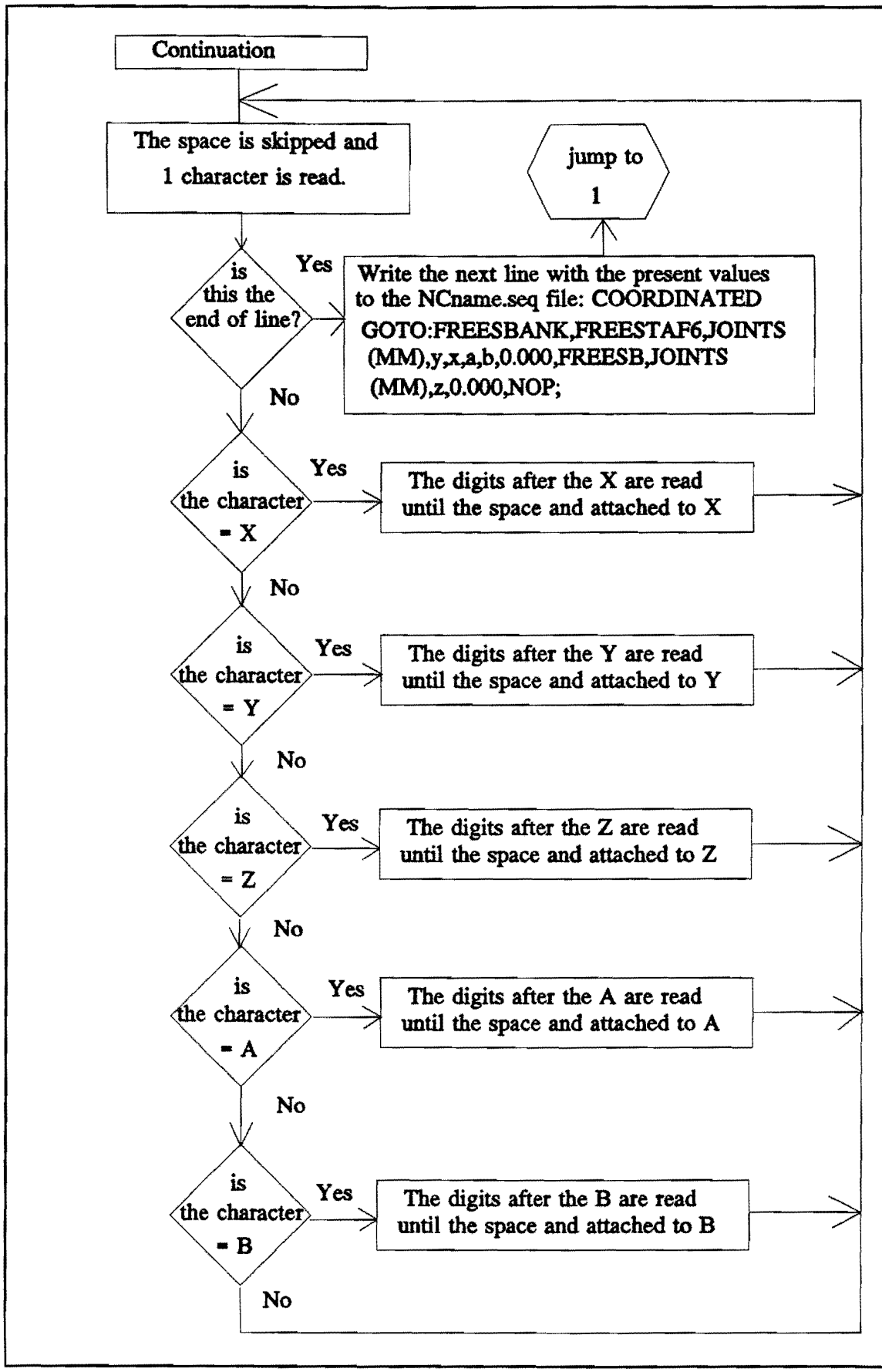
****

Ground Link Name = FREESTAF0

Appendix M. Extensive flowchart for motion simulation.




continuation
characters are = G00
Yes
No
characters are = G01
Yes
No
characters are = G02
Yes
No
characters are = G03
Yes
No
print on screen:
This command can not be translated in Robotics commands
It will be simulated as G01
Jump to 1
Go to the next page


Continuation
The space is skipped and
1 character is read.
jump to
1
is
this the
end of line?
Yes
Write the next line with the present values
to the NCname.seq file: COORDINATED
GOTO:FREESBANK,FREESTAF6,JOINTS
(MM),y,x,a,b,0.000,FREESB,JOINTS
(MM),z,0.000,NOP;
No
is
the character
= X
Yes
The digits after the X are read
until the space and attached to X
No
is
the character
= Y
Yes
The digits after the Y are read
until the space and attached to Y
No
is
the character
= Z
Yes
The digits after the Z are read
until the space and attached to Z
No
is
the character
= A
Yes
The digits after the A are read
until the space and attached to A
No
is
the character
= B
Yes
The digits after the B are read
until the space and attached to B
No

Appendix N. Motion simulation program example.

```
;****** PLACE Release 9.0 ******
DEFINE_COORD_MOTION_DEVICE: FREESBANK,FREESTAF6,FREESB;
SET_DEVICES_MONITORED: FREESBANK,FREESTAF6,FREESB;
ACTIVE_DEVICE: FREESB;
WORKING_TPOINT: FREESB2,TP1;
ACTIVE_DEVICE: FREESTAF6;
GOTO_JOINTS: (MM),200.0000,0.0000,0.0000,0.0000,0.0000,NOP;
CREATE_FRAME: PROEF2,PROEF2,FREESTAF4,(MM),1.0000,0.0000,0.0000,0.0000,1.0000,
0.0000,0.0000,0.0000,1.0000,0.0000,0.0000,0.0000;
MOVE_ABS: PROEF2,,FREESTAF4,,PROEF2,,0.0000,1.0000,0.0000,90.0000,(MM),0.0000,
0.0000,0.0000;
COORDINATED_GOTO: FREESBANK,FREESTAF6,JOINTS,(MM),90.0000,-30.000,0.0000,0.0000,
0.0000,FREESB,JOINTS,(MM),-30.0000,0.0000,NOP;
COORDINATED_GOTO: FREESBANK,FREESTAF6,JOINTS,(MM),75.0000,-30.000,0.0000,0.0000,
0.0000,FREESB,JOINTS,(MM),-30.0000,0.0000,NOP;
COORDINATED_GOTO: FREESBANK,FREESTAF6,JOINTS,(MM),75.0000,-30.000,0.0000,0.0000,
0.0000,FREESB,JOINTS,(MM),-120.0000,0.0000,NOP;
COORDINATED_GOTO: FREESBANK,FREESTAF6,JOINTS,(MM),75.0000,-70.0000,0.0000,
0.0000,0.0000,FREESB,NOMOVE,NOP;
COORDINATED_GOTO:
FREESBANK,FREESTAF6,NOMOVE,FREESB,JOINTS,(MM),-30.0000,0.0000,
NOP;
COORDINATED_GOTO: FREESBANK,FREESTAF6,JOINTS,(MM),75.0000,0.0000,0.0000,0.0000,
0.0000,FREESB,JOINTS,(MM),0.0000,0.0000,NOP;
COORDINATED_GOTO: FREESBANK,FREESTAF6,NOMOVE,FREESB,JOINTS,(MM),-150.0000,
0.0000,NOP;
COORDINATED_GOTO: FREESBANK,FREESTAF6,JOINTS,(MM),75.0000,-100.0000,0.0000,
0.0000,0.0000,FREESB,NOMOVE,NOP;
COORDINATED_GOTO:
FREESBANK,FREESTAF6,NOMOVE,FREESB,JOINTS,(MM),0.0000,0.0000,
NOP;
COORDINATED_GOTO: FREESBANK,FREESTAF6,JOINTS,(MM),75.0000,0.0000,0.0000,0.0000,
0.0000,FREESB,JOINTS,(MM),0.0000,0.0000,NOP;
COORDINATED_GOTO: FREESBANK,FREESTAF6,JOINTS,(MM),90.0000,0.0000,0.0000,0.0000,
0.0000,FREESB,NOMOVE,NOP;
COORDINATED_GOTO: FREESBANK,FREESTAF6,JOINTS,(MM),200.0000,0.0000,0.0000,0.0000,
0.0000,FREESB,NOMOVE,NOP;
ACTIVE_DEVICE: FREESB;
GOTO_HOME: NOP;
DELETE_DEVICE: FREESB;
MERGE_DEVICE: FREESB40,WORLD;
COLOR: FREESB0,PART,RGB,FORESTGREEN,0.3137,0.6235,0.4118,1.0000;
COLOR: FREES4,PART,RGB,LIGHTBLUE,0.6902,0.8863,1.0000,1.0000;
DEFINE_COORD_MOTION_DEVICE: FREESBANK,FREESTAF6,FREESB40;
SET_DEVICES_MONITORED: FREESBANK,FREESTAF6,FREESB40;
ACTIVE_DEVICE: FREESB40;
WORKING_TPOINT: FREESB2,TP1;
ACTIVE_DEVICE: FREESTAF6;
GOTO_JOINTS: (MM),200.0000,0.0000,0.0000,0.0000,0.0000,NOP;
COORDINATED_GOTO: FREESBANK,FREESTAF6,JOINTS,(MM),90.0000,-30.000,0.0000,0.0000,
0.0000,FREESB40,JOINTS,(MM),-30.0000,0.0000,NOP;
COORDINATED_GOTO: FREESBANK,FREESTAF6,JOINTS,(MM),75.0000,-30.000,0.0000,0.0000,
0.0000,FREESB40,JOINTS,(MM),-30.0000,0.0000,NOP;
```

```
COORDINATED_GOTO: FREESBANK,FREESTAF6,JOINTS,(MM),75.0000,-30.000,0.0000,0.0000,
0.0000,FREESB40,JOINTS,(MM),-120.0000,0.0000,NOP;
COORDINATED_GOTO: FREESBANK,FREESTAF6,JOINTS,(MM),75.0000,-70.0000,0.0000,
0.0000,0.0000,FREESB40,NOMOVE,NOP;
COORDINATED_GOTO:
FREESBANK,FREESTAF6,NOMOVE,FREESB40,JOINTS,(MM),-150.0000,
0.0000,NOP;
COORDINATED_GOTO: FREESBANK,FREESTAF6,JOINTS,(MM),75.0000,-100.0000,0.0000,
0.0000,0.0000,FREESB40,NOMOVE,NOP;
COORDINATED_GOTO:
FREESBANK,FREESTAF6,NOMOVE,FREESB40,JOINTS,(MM),0.0000,0.0000,
NOP;
COORDINATED_GOTO: FREESBANK,FREESTAF6,JOINTS,(MM),75.0000,0.0000,0.0000,0.0000,
0.0000,FREESB40,JOINTS,(MM),0.0000,0.0000,NOP;
COORDINATED_GOTO: FREESBANK,FREESTAF6,JOINTS,(MM),90.0000,0.0000,0.0000,0.0000,
0.0000,FREESB40,NOMOVE,NOP;
COORDINATED_GOTO: FREESBANK,FREESTAF6,JOINTS,(MM),200.0000,0.0000,0.0000,0.0000,
0.0000,FREESB40,NOMOVE,NOP;
DELETE_FRAME: PROEF2;
COORDINATED_GOTO: FREESBANK,FREESTAF6,HOME,FREESB40,HOME,NOP;
```

Literature

[1] Friedman, F.L., Koffman, E.B., *Problem Solving and Structured Programming in Fortran 77*, third edition

[2] Heintjes, T.B., *Handleiding voor het gebruik van de Manufacturing Operations Module uit het Unigraphics systeem*, WPAnr. 1168, Eindhoven, 1991

[3] Hewlett Packerd Company, *A Beginners Guide to HP-UX*, 1991.

[4] Kaas, E.A., Stakenborg, M.J.L., *CAD/CAM/CAE in de Werktuigbouw*, Deventer, 1990

[5] McDonnell Douglas Corporation, *Build User Guide*, version 7.0, 1990.

[6] McDonnell Douglas Corporation, *GRIP Programming Manual Vol1 & Vol2*, Version 8.0, 1991.

[7] McDonnell Douglas Corporation, *Place User Guide*, Version 7.0, 1990.

[8] McDonnell Douglas Corporation, *UG Solids operational description*, Version 8.0, 1991.

[9] McDonnell Douglas Corporation, *UnigraphicsII Concepts and Common Functions*, Version 8.0, 1991.

[10] McDonnell Douglas Corporation, *User Function Manual*, Version 8.0, 1991.

[11] Melio, J.P., *Executing ADJUST on the FALC cell*, WPAnr, 1196 Eindhoven,1991.

[12] Senden, R., *Realiseren van een CAD/CAM koppeling*, WPAnr. 0930, Eindhoven, 1990.

[13] Vernooij, J, *The simulation of the MAHO-700S milling machine by the use of the software package Robotics*, WPAnr. 1088, Eindhoven, 1991.

[14] Willems, M.C., *The off-line programming of welded products with the Robotics software*, WPAnr. 1321, Eindhoven, 1992.