NWPKU / 93 sue

uell Rof.

The Nottingham Trent University

Faculty of Engineering and Computing

Department of Manufacturing Engineering

Intelligent integration of computer aided inspection with CAD/CAM systems

Ane Miren Leibar

This thesis is submitted as partial fulfilment of the requirements of the Nottingham Trent University for the degree of Master of Philosophy.

October 1993

Summary

As a result of Coordinate Measuring Machines (CMMs) becoming widely used in industry, a need to link CAD and inspection together and program CMMs off-line has emerged. The work undertaken for this research involves the development of a software interface to link any CAD system with any CMM utilising standard data exchange interfaces. The IGES (Initial Graphics Exchange Specification) interface was used for a limited two way communication with any CAD system and the DMIS (Dimensional Measuring Interface Specification) interface was used for improved communication with the CMM. Although the CAD systems tested were AutoCad and Unigraphics, it is expected that other systems too could be used easily for the data transactions.

The system was developed using the C++ language, to run on any PC with a VGA colour monitor. The different modules created and their functions are:

1. An IGES post processor to read 2½D information from a CAD system.
2. A user interface to allow interaction with the extracted CAD data, to create the inspection model. This involves using the keyboard, mouse and monitor. In order to create the inspection model, the user has to select the features which are to be measured and apply the corresponding tolerances to them. Then the user has to define the measurement sequence and if needed generate a 'shorter' measurement path. A limited error avoidance algorithm is incorporated.
3. A pre-processor to create DMIS and CMES output files, to drive the CMM to measure the created inspection tasks.
4. A simulation of the movements of the measuring probe on the screen, when the inspection programs are executed.
5. A post-processor to read DMIS output programs and get the measured values into the system. The measured features and their values will be displayed in different colours depending on whether they are within or outside the tolerances.
6. A pre-processor to write an IGES file and input the measured values into the CAD system.

The results of the IGES/DMIS inputs and outputs have been successfully tested for all the features provided by the system in 2½D parts several times to ensure the application was working properly. The limitations of the system are presented.

Table of Contents

Acknowledgements

For the support during my MPhil I would like to thank the following:

My parents, for their continual support, patience and love, and for all the sacrifies they have made throughout my education.
My sisters Edurne and Miriam and granny Justa who have supported and encouraged me from the very start of this work.

My friends Rosa, Lucia, Aitor, Asier, Fernando and Xabier who came with me from Basque Country to extend our knowledge in the manufacturing area.
The people in the office Andy, John and Bala who have all become very good friends.
Special thanks to Chris, who has become a very good friend, for correcting my English during the writing process.

I would like to express my gratitude to my supervisor Dr. Siva for his advices, constant support, and guidance throughout the fulfilment of this project.

And finally special thanks to Alex for his continuous support, friendship and love.

Chapter 1 : Introduction

Computer Aided Design (CAD) and Computer Aided Manufacturing (CAM) are increasingly becoming popular in industry. Furthermore, increased complexity in part model design, tighter tolerances and higher functional specifications have resulted in the need for computer controlled coordinate measuring machines (CMM), which are widely used for efficient and accurate dimension verification.

CMMs are expensive equipment which spend most of their time -"learning" to do what they are supposed to do - being programmed on line. Approximately 80% of CMM users program their CMMs on-line, thus they tie-up the machines by programming in the "teach" mode rather than off-line [Mason 92].

However, there is a growing trend to link CAD and inspection together and to start off-line programming for CMMs. This can be done with the use of either a CAD system, or a dedicated software designed for CMM simulation and programming. Most expensive Computer Numerical Control (CNC) machine tools, like sophisticated machining and turning centres, are programmed either off-line, or at the machine control while a part program is running, to improve the productivity of the system. CMMs are only just beginning to be used in a similar manner.

Ideally, CMM users would like to generate complete CMM programs on their CAD terminals without manual intervention - i.e a complete automatic inspection. One of the drawbacks is that many present day commercial CAD systems were designed for easy documentation, draughting, design, and normally without automated inspection in mind. Thus they do not have all the necessary information to create the inspection program [Hahn 88]. In order to solve this problem the user has to interact with the program using a computer graphics system in which a model of the part can be displayed.

1.1 Off-Line programming

The off-line programming of CMMs by linking CAD and inspection has various advantages:

- Reduces the labour required to create the part program. Instead of typing commands, a programmer might use the interactive graphics feature of a CAD/CAM system to select commands from a menu and indicate points to be measured.

- Forces engineers and designers to examine whether their designs can be inspected early in the design process. Too often designers specify tolerances which cannot be gauged or devise systems which cannot be tested.

- Is much less expensive than "teach programming" the CMM on line as the cost of a programming station is well below that of a major CMM.

- Part programming can begin at the programming station without a part physically available.

- Potential collisions during measurements can be spotted in simulation and avoidance steps taken before the task. Thus, saving a lot of grief and increasing productivity of the CMMs.

- The measured values can be read back into the program, (showing them on the screen) and also fed back into the CAD systems.

- Most importantly, a link between CAD systems and inspection machines enables engineers to analyze the data measured by the CMM to determine whether the parts have met specified tolerances, and the reasons for nonconformity.

One of the reasons that more CMM programming is not done on off-line graphical programming systems (typical of CNC machine-tool programming) is that "the state of CMM programming is about where CNC programming was a decade ago" [Mason 92]. Some manufacturing firms have been linking CAD systems to coordinate-measuring machines for more than five years [Hahn 88]. However, early attempts were not very successful. Since then, new software developments have made CAD to CMM links more productive. Over the past couple of years, a few CAD software designers have begun to develop off-line programming facilities for CMMs. A software (preprocessor) is then necessary, for the integration of a CAD system and a CMM. Some CMM vendors have taken advantage of these developments and have provided post-processors for the different CMMs. Most CAD system vendors, however, are slow in supplying the link.

1.2 The aim and objectives of the project

The aim of this research was to create a link between any CAD system and any CMM. To do so, a standard input and a standard output must be used. The international standard IGES (Initial Graphics Exchange Specification) is used to extract data from the CAD system. Then, using the mouse and the keyboard the user interacts with the program. The required features to be inspected on the part are then selected with the mouse. Once the selection is finished, the manner of execution of the measurement has to be defined. This can be performed either following the order of selection or taking the computer solution. The output of the inspection program will be in the international standard DMIS (Dimensional Measuring Interface Specification) or in the CMES language (specific for LK machines). The reason of using CMES is because it is the programming language used by the CMM utilized in this research. The DMIS standard provides the vocabulary for the language that allows for 2 way communication between

Dimensional Measuring Equipments (DMEs) and their 'host' computers.

In this research, the results of the measurements are read from DMIS into the application and displayed in different colours (depending on whether the values are inside tolerances or not) on the screen. Thus the user can easily see if the part has met specified tolerances. The application also gives the option to create an IGES file to get the measured values back into the CAD system and to manipulate them.

The following diagram explains graphically the work done in this research:

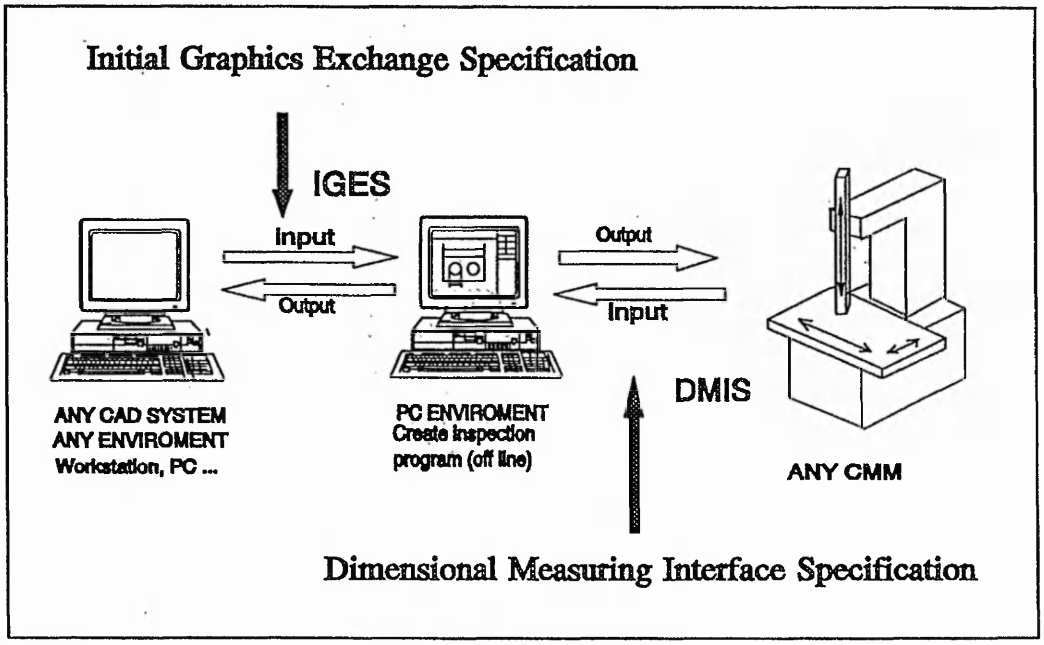


Figure 1 : Linking CAD with CMM

This thesis is divided into three main areas. The first area presented in chapter 2 is of general purpose. In this chapter an introduction to the tools used for this work, such as the languages IGES, DMIS, CMES and the CMM and an overview of current work to solve this problem, are presented. The second area which includes chapters 3,4 and 5 explains the work undertaken in this research. Chapter 3 explains the process followed to get the information from the CAD system and to display it on the screen. Chapter 4 explains how to create the inspection model and the simulation. Chapter 5 explains how to create the inspection program in both inspection languages and how to get the information back into the system via DMIS and IGES.

Finally in the third area, which includes chapters 6, 7 and 8 conclusions, further work and the results obtained are presented.

In the appendices, extended information of IGES, DMIS and CMES, along with examples and a list of program files are presented.

Chapter 2 : Background information

This chapter is divided into eight sections. In the first three sections, IGES and DMIS standards, and CMES language are explained in general. The fourth section describes a CMM and its functions and the fifth section briefly surveys the available commercial CAD systems and their functions and capabilities relevant to this area. Finally, the last three sections describe the different approaches taken in recent researches to create the inspection model, the inspection planning and the feedback. The aim of this chapter is to give the reader general information around the area of this research.

2.1 The IGES standard

2.1.1 Introduction

The Initial Graphics Exchange Specification (IGES) is a widely accepted neutral file format that establishes information structures to be used for the digital representation and communication of product definition data used by various CAD/CAM systems. Initiated in late 1979, IGES is a mature mechanism that provides a stable, standardized, vendor independent format to aid in the management and use of data for CAD/CAM systems [Dori 92]. A detailed description of the IGES specification is given in Appendix 1.

2.1.2 Technical Overview

IGES was originally designed to avoid problems encountered during the 1970s with the proliferation of direct translators. IGES was developed in 1979 under the leadership of the National Bureau of Standards, whose goal was to facilitate the transfer of product definition data between different CAD systems [Bloor 91]. Version 1.0 was published as a part of the ANSI standard in 1981. Several

versions have been released since the first one in 1979. The version at the time of writing this thesis was V5.1 and the next version (V6) is expected to be released in 1995. The advantage of neutral file transfer, compared with direct translators can be explained as follows:
Writing direct translators between 4 different systems requires 12 different translators called pre and post processors. Adding a fifth system adds 8 additional translators, and the number of translators continues to go up geometrically with the number of systems (see figure 2).

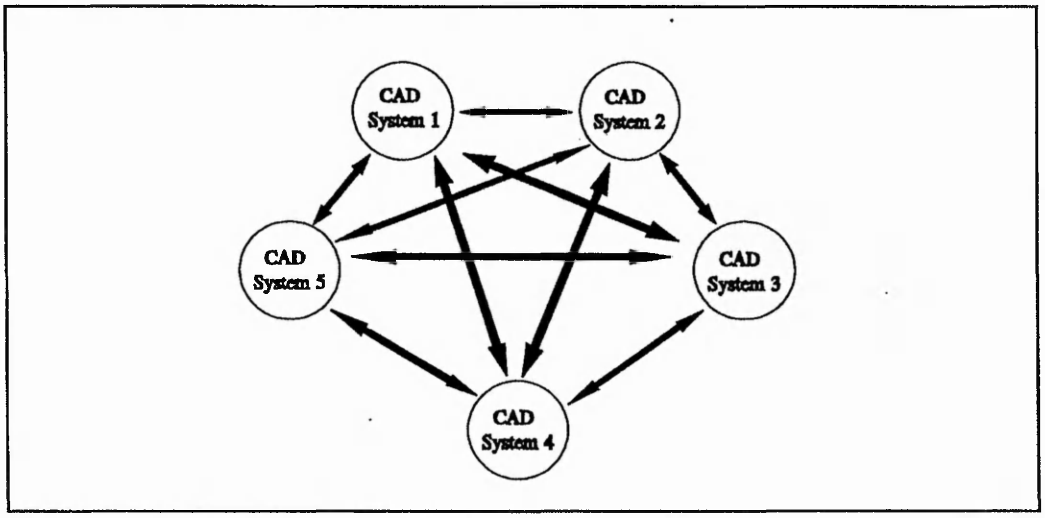


Figure 2 : Direct Translators

IGES uses the neutral file concept. Thus when translation using a neutral file is carried out from one native format to the neutral file and then to another native format (see figure 3), four systems require eight translators. For each system added thereafter, only two more translators are required.

A side benefit of neutral files is that they can potentially be archived. Some companies in the aerospace industry, for example, need to keep CAD databases for 20 to 50 years. The IGES organization has a commitment for upward compatibility so that IGES files created earlier can be read by new CAD systems as they are developed.

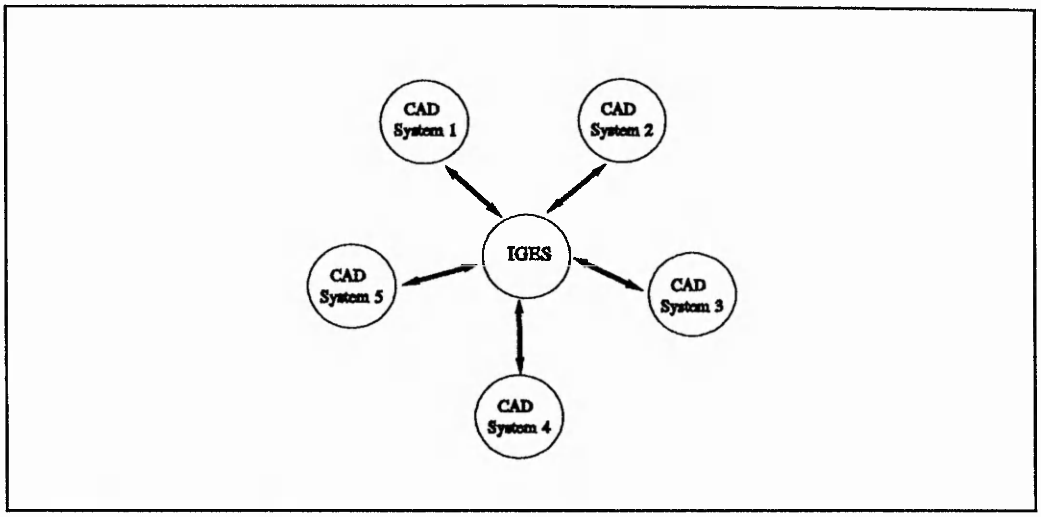


Figure 3 : Neutral translator

IGES defines a file structure format, a language format, and the representation of geometric, topological, and non geometric product definition, in these formats [IGES 91]. Developers must write software to translate from their system to IGES format, or vice versa. The software that translates from a CAD system to an IGES file is called a pre-processor. The software that goes in the reverse way (translates from IGES to a CAD system) is called a post-processor (see figure 4). The combination of pre-processor and post-processor determines the success of an IGES translation.

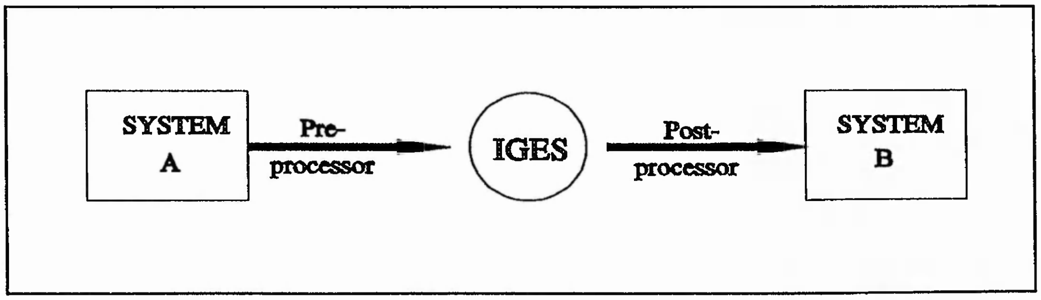


Figure 4 : Neutral translator

2.1.3 General Structure

IGES is defined in an ASCII, user readable format using 80-character records. These records are referred to as lines. To create smaller files (IGES file sizes are usually 5 to 10 times the size of CAD native databases), a binary format and a compressed ASCII format are also defined, but the majority of IGES processors still only support the original ASCII form.

The files are divided into six sections. They are, flag, start, global, directory entry, parameter data, and terminate sections (Appendix 1).

- The flag section (optional) indicates if the binary or compressed ASCII format is used, with letters B (for binary) and C (for compressed) in columns 73 of the first line.
- The start section is just readable text at the start of the file used for documentation.
- The global section is 24 parameters of a global nature, such as the name of the file, its author, date of creation, units of measurement, precision of the numbers, and so on.
- The directory entry (DE) section contains data that is common for each entity in the file, such as its type, colour, line style, layer, views it's visible in, and a transformation matrix to position the entity. There are two lines in the DE section for each entity.
- The parameter data (PD) section contains specific entity information. There are one or more lines in the PD section for each entity.
- The terminate section, is a single line at the end of the file that contains the number of lines in each section.

Since IGES files typically contain lots of entities, the PD and DE sections normally form the major part of these files.

2.1.4 Current Status

IGES is supported by most of the CAD vendors. It is a popular method attempted by developers trying to move CAD data from one system to another. However it has not lived up to expectations, although many of the expectations were unrealistic. The primary expectation was that a user can take any CAD file, translate it into IGES, read the IGES file into another CAD system, and have 100 per cent of the data transferred, including resolution of system differences. In real life some transfers are 100 per cent, whereas some are still very minimal, and most lie somewhere in between [Mayer 87]. It is difficult to give an average figure, since the success rate depends on the IGES processors used and the types of entities in the files to be translated.

The quality of IGES processors varies widely. Some processors handle surfaces and complicated structures whereas others handle little more than lines, points and circles. Obviously success depends in part on the type of systems employed.

2.1.5 Problems with IGES

Many of the problems with IGES are caused by the manner of implementation. Each CAD developer implements his/her own IGES processors, and some are better than others. After all, the IGES organization is a voluntary one, and developers implement as much (or as little) of the specification as they see fit. Because of the large variety of data defined in IGES, no developer supports the complete specification; in fact, the majority probably support less than half [Vosn 89]. To make matters worse, the problem is compounded by inevitable differences in interpretation and any bugs in software.

The fact that each vendor implements only those IGES entities that are perceived to be relevant to their system, hampers the communication between different systems, because each system supports a different set of entities and sometimes not all of them are in the IGES specification. Therefore only the common entities between the three of them will transferred successfully as shown in figure 5 [Bloor 91]. Some existing specifications like VDA-IS and MIL-D-2800 (see section 2.1.6), are trying to solve this problem using an application subset, which reduces this problem by presenting an enumerated list of entities that are to be used in a particular application.

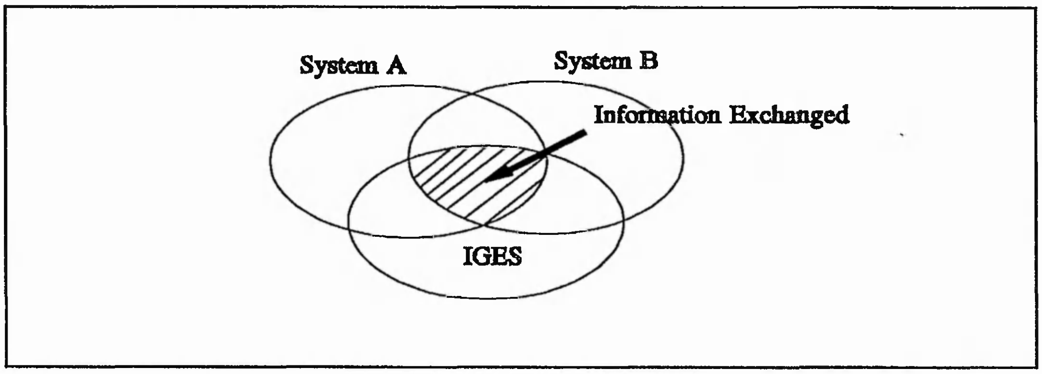


Figure 5 : Data Exchanged

A further problem, is finding an agency to certify IGES processors. In USA, the National Bureu of Standards runs the IGES organization, but by its very nature, it is prohibited from taking actions which resemble certification. Also in the light of current legal issues, no standards organization wants to say that a company's product does or does not meet a particular standard. Instead, the IGES organisation is working with the Society of Automotive Engineers (SAE) to develop a validation program. Under the program, the SAE will certify that an IGES processor works as well as the developer claims [Mayer 87]. Yet there are some organizations trying to bring order into the confusion surrounding IGES. They are the large CAD users who want IGES to succeed. These users have issued a simple ultimatum: support IGES or we will not buy [Smith 90].

In conclusion, it can be said that the IGES format is complex. However it is widely used because it is one of the few standards that is both available and supported; therefore, it will continue to be used, supported and expanded for the foreseeable future [King 92].

2.1.6 Alternatives

Even though the need for CAD data transfer is great, IGES has not been a perfect answer, and a number of alternatives, each with its own strengths and weaknesses, exist.

• **Direct translators :** A direct translator is a software that reads a specific CAD/CAM system data base format and converts it to another specific CAD/CAM system data base format. Direct translators have the advantage of being fast; they need to deal only with the entities that the two systems have in common. One disadvantage is the number of processors that need to be created (see section 2.1.2). Another disadvantage is that they need to access the data base formats of each CAD/CAM system, which are not widely available and access is limited.

• **VDA-FS (Verband des Automobilindustrie FlachenSchnittselle):** This is a standard developed by the German Car Manufacturers Association that covers points and parametric polynomial curves and patches. It has been designed to keep the interface simple and handles essential elements only for surface transfer. It is used in Germany (as German national standard DIN 66301) [Smith 90] and in the UK mainly in the automotive industry. Although it is less complicated than IGES, it is limited to curves and surfaces.

• **VDA-IS:** This was published in 1987 and it defined five subsets for the IGES specification: three subsets for geometry and two for annotation and structure. For a processor to support any of the subsets, it has to translate **all** the entities within it. It tries to complement VDA-FS introducing some IGES entities.

• **MIL-D-28000:** This is the military specification for the US Department of

Defence. The standard provides subsets for technical illustrations, engineering drawings, electrical and electronics applications, and geometry for CNC manufacturing by mandating that no entities other than those in the list shall appear in a file produced by a processor that claims conformance to the application subset; a common 'subset' of entities may be used in the exchange.

• **SET (Standard d'Echange et de Transfer):** This was developed as a more compact data exchange form than IGES and has facilities for transferring geometry (including rational polynomial surfaces, annotation, and structure), but it is no more functionally complete than IGES and it is less popular, consequently no many CAD systems support it. It has become a French national standard (Z68300).

• **Other formats :** Several other formats are in use for data exchange that are not organized by standard bodies. These include DXF, which is a format used by AutoDESK, and the Integraph Standard Interchange Format (ISIF) as well as others used by various automotive manufacturers. Of these DXF is the most widely used, but is different from such standards as IGES, in that it is a proprietary product.

2.1.7 Future Development

Work currently undertaken by the International Standard Organization (ISO) for the Exchange of Product Data (STEP) project is to develop a three-layer model to represent the total information content of a product. This is achieved by creating a product model which contains all the information a product needs throughout its life time [King 92].

Formal methods are used in the development of models and as a result STEP avoids some of the problems that were encountered with IGES, such as ambiguities with respect to entity definitions.

The tendency of present CAD systems is the move towards 'feature-based' systems, the object orientated programming provides a powerful tool to create this kind of systems. The idea is to create the designs based not only in the geometry but also in the functionality of the features. This will provide more relevant information about the part to the subsequent processes (manufacturing, inspection ...).

The ideal standard output for this type of CAD systems is the STEP specification. The problem is that this option is not commercially available and even when the tendency is towards a feature-based system, CAD systems do not support it yet.

2.2 The DMIS standard

2.2.1 Introduction

The objective of the Dimensional Measuring Interface Specification (DMIS) is to provide a standard for the bidirectional communication between computer systems and inspection equipment. The specification is a vocabulary of terms which establishes a neutral format for inspection programs and inspection results data. While primarily designed for communication between automated equipment, DMIS is designed to be both man-readable and man writable, allowing inspection programs to be written (and inspection results to be analyzed) without the use of computer aids. Appendix 2 gives a detailed description of DMIS.

Even before its approval as a national standard, DMIS was opening up a new industry, with DMIS products and DMIS supported equipment appearing regularly in the market. Vendors of measuring and CAD equipment see this as a boon to the industry, as these companies can now write a single interface to the standard [Anon 90].

2.2.2 Technical Overview

The development of the DMIS specification was funded by the Quality Assurance Program under the Guidance of Computer Aided Manufacturing International Inc (CAM-I) group. The first version of DMIS was developed by ITT research under contract to CAM-I, and completed in 1986. The second version of the specification was developed by Pratt & Whitney, a division of United Technologies Corporation, yet again, under contract to CAM-I, and was completed in September 1987.

DMIS version 2.1 (current one) is an update of DMIS 2.0 as developed by the Technical Advisory group in response to the ANSI Standard investigations and system improvement requests. DMIS version 2.0 was submitted to the American National Standard Institute (ANSI) as a quality interface and received a 75% acceptance. This was acceptable to ANSI for approval of the specification, but the Technical Advisory group wanted a higher acceptance rate, leading to the revision of DMIS 2.1 which was again submitted to the companies in the canvass and received 98% acceptance. The process for approval of DMIS specification as an American Standard was completed in March 1990 [Daniels 92].

DMIS like IGES is also based in the concept of neutral format (see section 2.1.2) as shown in the diagram of figure 6:

An equipment which interfaces to others through the DMIS vocabulary will have a pre-processor to convert its own internal data into the DMIS format, and a post-processor to convert the DMIS format into its own data structure.

The implementation of DMIS is dependant on individual users. Some may choose to link CAD systems directly to DME's (Dimensional Measuring Equipment), and some may choose to use a host computer; some may choose a serial data link, some may choose parallel, and so forth. DMIS simply defines a vocabulary set to be transmitted by ASCII files. The method for the transmission, storage, and management of these files is user-dependant.

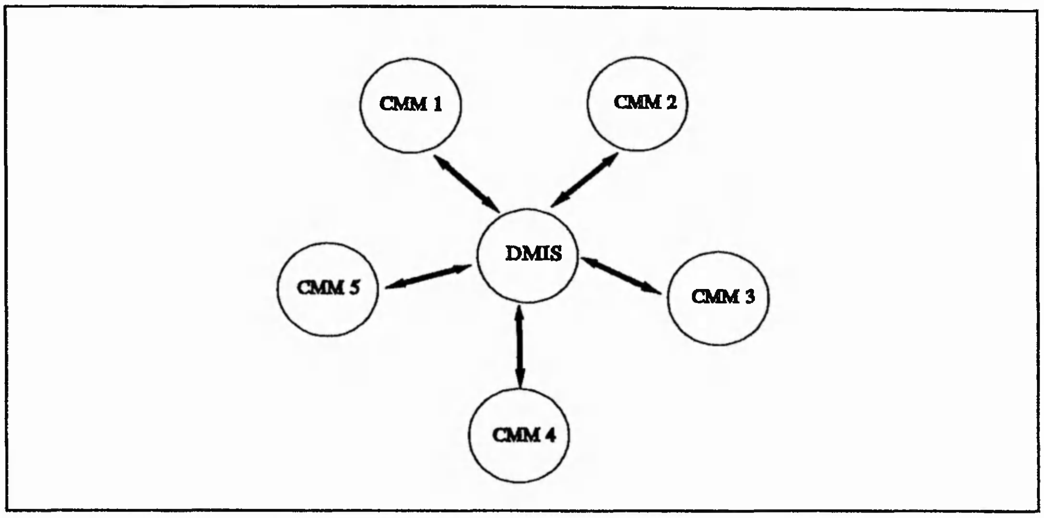


Figure 6 : The DMIS standard

2.2.3 General Structure

DMIS is a vocabulary of major and minor words. It is similar to the APT (Automatic Program Tool) numerical control language, with the major and minor word separated by a slash, and proceeded by a list of parameters.
There are two basic types of DMIS statements. These are:

- **Process-oriented commands** which consist of motion commands, machine parameter commands, and other commands which are unique to the inspection process itself.

- **Geometry-oriented definitions** that are used to describe geometry, tolerances, coordinate systems, and other types of data which may be included in a CAD database.

Presently part models do not include all of the data needed in the DMIS interface, so supplementary data must be added manually. The evolution of CAD systems, though, is in the direction of complete part models, and DMIS has been designed to be compatible with this growth path.

2.2.4 Current status

The CAM-I group have developed a test piece that combines various, but commonly encountered features for manufacturing to check vendors translators. Most CMM vendors usually produce their own test parts, so it is now becoming increasingly difficult to test the various DMIS functions that the CMM vendor can offer. By using a standard test piece (ANC101) and generating a DMIS off-line program, one can test the quality of the vendors translators, physically measure the part, and then return the results of the measurement programs back through DMIS. A comparison of vendor capabilities can then be made [Daniels 92].

The CMM/DMIS interface places programming efforts in the quality assurance office, where you can respond to engineering product releases and change orders with new and updated inspection data required to meet manufacturing needs.

Using the DMIS interface improves the productivity of the company, by eliminating manual labour intensive data entry techniques for the quality operations. This interface reduces the program preparation times, and increases machine availability to inspect parts.

The CAM-I group in Europe have had little response from the engineering and quality assurance market place with regards to this specification, and it seems that industry is not fully aware of the capabilities that CAD/CAM and DME vendors can offer to provide an integrated solution in a CIM (Computer Integrated Manufacturing) environment.

2.2.5 Future Development

This version of DMIS incorporates the commands necessary to drive coordinate measuring machines and video inspection devices for dimensional inspection of discrete mechanical parts. The intention is to provide a standard of communication for all DME. Future versions will expand optical capabilities, incorporate other DMEs (Robotics, photogrammetry, theodolite, laser ...), and include other application areas (Surface analysis, composites inspection, electronic applications...).

2.3 CMES language

2.3.1 Introduction

The objective of the CMES (Coordinate MEasuring Software) language is to provide a communication language for the LK company's machines. The language is a group of commands which establishes a communication between the machine and the user. The latest version at the time of writing this thesis was V11.1. Appendix 3 gives a detailed description of the CMES language.

2.3.2 General Structure

- **Command codes**

CMES responds to command codes, which normally comprise two characters sometimes with one or more parameters.

- **Command parameters**

In order to increase the capability of each command, most CMES commands are equipped with command parameters which are used to cause the command to function in a specific manner.

Parameters are always shown in italics (Appendix 3). The parameters not enclosed in brackets must be entered, and the parameters enclosed in brackets are optional. Where applicable, the command may assume default values or actions if no parameter is used.

Parameters are usually separated from the command and each other by a single comma (,). The only exception being the /±/ and /tol/ parameters. Where commas are required, they are shown in the command line.

• **Probe compensation**

Compensation for the probe stylus is achieved either by the CMES command performing the measurement or by the use of an optional probe compensation symbol; the latter is used when the probe error is radial.

Commands which perform radial probe compensation indicate this by using the /±/ parameter. This means that you can enter the "+" symbol to increase the dimension by the probe radius or use the "-" symbol to reduce it. The symbol is entered after the command name.

• **Multi-point measurement**

The multi-point facility allows the user to increase the number of points that a particular command requires in order to measure a feature. The availability of the multi-point feature is indicated by the presence of the [.points] parameter in the command line. In practice, the parameter number should be replaced with a numeric value and separated from the previous parameter with a comma. If the parameter is not given, the default value is used.

• **Tolerancing**

Certain commands allow the user to enter tolerance data with the command so that a full dimensional inspection report can be prepared. This is indicated by the command line containing the [tol] parameter, in practice the parameter is replaced with the appropriate tolerance symbol.

2.4 The Coordinate Measuring Machine (CMM)

2.4.1 Introduction

A sky-hook holding a probe, a computer with antennae - however one thinks of a CMM, it is more than just a sophisticated replacement for traditional measuring tools. It can reduce the time to inspect components dramatically. It can measure shapes that would be difficult or impossible to measure otherwise. It can provide the production shops with valuable feedback, and the time is ripe when it will be an integral part of CAD/CAM or CIM [Anon 91]. To understand how CAD systems and CMMs can work together, it is helpful to review how CMMs work.

2.4.2 What is a CMM ?

A CMM is a machine having a series of movable members, a sensing probe and a workpiece support member, which can be operated in such a way that the probe can be brought into a fixed and known relationship with points on the workpiece surface, and the coordinates of these points can be displayed or otherwise determined with respect to the origin of the coordinate system [BS6808 87].

CMMs can have contact ("touch probe") or non-contact sensor system. The "touch probe", is usually mounted on a mechanism that permits the probe to move along one or more perpendicular axes. The mechanism contains electronic sensors which provide a digital readout of the position of the probe. When the probe "touches" the surface of a part being measured, the location of the probe in space is recorded by the mechanism.

A variety of mechanisms are used to support CMM probes. Some machines mount the probe on a cantilevered arm. Others employ a gantry or bridge arrangement. Still others employ moving tables beneath a probe whose axis moves in one or two directions (see figure 7). The most popular one is the bridge design [Quinlan 88].

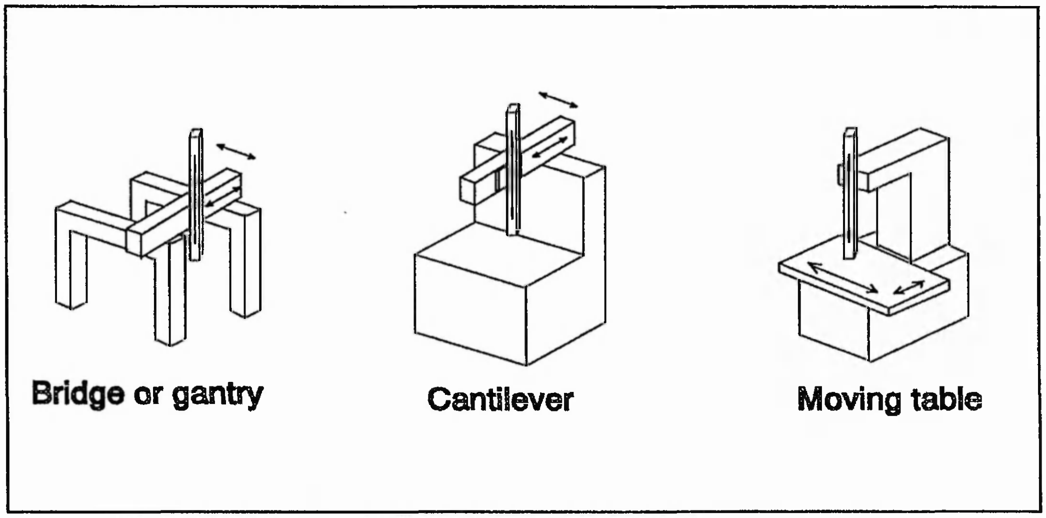


Figure 7 : Different types of CMMs

The motion of the CMM can be manually controlled, or can be controlled by servo-motors. The latter ones have more interest for CAD users. The motor driven machines are generally controlled by a small computer, similar to those found in numerical controlled machine tools. These machines can be programmed to follow a predefined path and to stop at various points along the path to measure the position of the part surface. The data gathered in this way can be stored and analyzed by other programs or presented as an ASCII file.

2.4.3 Setting-up the CMM

• Mechanical reference

The CMM has a mechanical reference artefact to undertake the verification and periodic verification of the CMM performance. The main characteristic of this device is its accuracy. Suppliers and users of CMMs have devised and developed a wide range of devices to create a number of well defined features, the measurement of whose spatial coordinates can be used checked day to day.

The first step undertaken when the CMM is going to be used is to determine the position of the mechanical reference (e.g, a sphere at the end of a bar). The X,Y,Z position is determined and stored by the computer for future use. This datum position is used for two functions:

- As an origin point for the set-up commands to determine the attitude of the component to that of the machine.
- As a means of probe qualification.

• Probe qualification

The probe is first qualified by using the mechanical reference. The probe will touch several points in the reference and the diameter of the probe stylus being used will be automatically calculated. The position and "diameter" of the probe can be saved in a variable to be used later on.

If the probe position is altered accidentally during an inspection sequence then the user must set-up the new position and requalify the probe returning to the mechanical reference before any measurements are taken. This operation will effectively cancel the errors induced by the accident.

• Setting the part coordinate system

Before a component can be inspected, the relationship of its axes to the machine axes must be established. If the user does not wish to use the set-up commands then the component must be set square to the machine axes by standard methods.

The usual method of proceeding with an inspection piece is to issue the commands in strict sequence, either by taking points on the component as required or by saving points established by previous indirect commands and releasing these for use by the set-up commands. There are different methods of setting-up the part coordinate system, depending on the CMM used.

2.4.4 Programming the CMM

A part program can be described as a program written in a specific application language containing all the necessary instructions for the automatic measurement of a given part. A part program includes instructions for:

- Positioning the probe in x,y and z coordinates.
- To pick-up a point, that is the approaching direction in x,y and the point to be scanned.
- Processing of geometrical elements, relation between geometrical elements and choice of result format.

A modern part-program language must also allow the execution of all the performances belonging to any type of advanced language; that is, it must be possible to define conditioning and iteration commands to repeat functions. The part program can be created by two different techniques. One is creating the part program teaching the machine and the other technique is to create the part program using an application language.

• Self-teaching

Self teaching is one of the most widely used part programming techniques [Ercole 91]. It consists of creating a "master" by performing an inspection cycle on the first part in a batch by hand, with the CMM computer system memorising the routine to allow for automatic repetition on the subsequent parts.

While this is simple to enforce, its popularity is also based on the capability to set-up a CMM "in the field" - though it does have drawbacks: the necessity to set-up the part to be measured, plus the fact that the CMM is out of action in an inspection sense. Also not all the programs allow the simultaneous control of three coordinate measuring machines axis, nor can they generate point pick-up along vectors that are not parallel to x, y and z axis.

• Application languages

Application languages are used by the programmer to generate part-programs for automatic inspection cycles. And, while this calls for deep knowledge of the language, which is not uncommon, there is a problem because until recently no standard language existed. So, different machines may require different programming language, which can be confusing and restrictive. This can lead to possible errors if the same measurement task is required on different machines in the same company.

2.4.5 CMM Software

Measurement software has evolved from a laborious programming language into an application package of modular tools readily accessible by hierarchical menu or graphics-based operator interface systems. Some years ago, the software programmer could do just about anything he/she wanted. With today's CMM packages shopfloor people can accomplish 80% or more of what the old programmer could do in a fraction of the time [Tackes 90].

The problem is that software programs for CMMs are not written by inspection staff; they are generally written by programmers who are lacking in sufficient experience of inspection requirements. As a result a software package eventually proves to be incomplete.

Consequently the software widely used by CMMs to process measurement data, particularly geometric form assessment software, is of very varied quality. Some packages, are user friendly; some are distinctly not. It is important to choose the right software as it determines the speed of inspection. Recognition of the failings of a significant proportion of such software has caused a loss of confidence among users. Some methods for the assessment and comparison of this software quality is being currently discussed. The UK Standards Committee, AMT/8, is already working on a standard to test the software [Peggs 91].

All these problems make the CMM software the single most important item in choosing a coordinate measuring machine [Cox 93].

2.5 CAD vendors approach

With the exception of big CAD system vendors, the majority have largely ignored the problem of linking design and test. This is probably because CAD users have not given design-test links a high priority compared to other issues. A few major CAD systems offer some capability in this area but are less than optimal for CMM programming and are not widely used for off-line programming [Mason 92]. The following firms have done significant work in developing products that link CAD to CMM systems:

• **Cimlinc**

Cimlinc was an early entrant into the CMM programming game, thanks to its shopfloor orientation. The company's product, called "CIM CMM" runs with the "CIM CAD" software, a three D design, draughting, and modelling program that runs on Sun Microsystems work stations.

The product lets a user drive a probe around a part model and take measurements at various points. The system is driven by typing DMIS commands or by selecting screen icons which invoke a series of DMIS commands. Commands to move the probe to a particular location may be invoked by selecting a point or other feature in the CIM CAD model. As the programmer works, probe motion is displayed graphically on the Cimlinc screen. The DMIS command files generated by CIM CMM can be processed to run in a variety of CMM using software from the CMM vendors. CIM CMM is also capable of receiving DMIS data from the CMM and displaying the measured values in the graphic screen.

• **Computervision**

Computervision's CMM programming software, called "Automeasure", works with its popular CADDS 4X software. It allows engineers to interactively create CMM programs which inspect models created at the CADDstation.

Automeasure employs a command language which may be invoked by typing or by selecting on-screen icons with the mouse cursor. It developed its own neutral file output (Neutral Data File) as an alternative to DMIS, so it does not offer DMIS support. It attempts to address most of the requirements for a good off-line programming system [Schaffer 85].

• **Cadam**

Cadam's CMM programming facility, called "CADIMS" was originally developed by Lockheed's Missiles and Space division using Cadam's NC2 numerical control software as a base. The software runs on IBM's System 370 family of mainframes and compatible models only. It requires NC2 and base Cadam as prerequisites.

CADIMS permits users to drive a model of a CMM probe around a Cadam model using Cadam's interactive graphics commands and to automatically generate a CMM program. Output in the CMM language can be produced by direct post-processors or by DMIS.

• **Calma**

Calma's "DDM/CMM" software provides off-line programming for specific CMMs manufacturers. It runs with Calma's software which can be used to create the part model to be inspected. The software runs on Apollo or DEC Microvax computers. DDM/CMM does not employ DMIS, but relies on direct processing of DDM motion statements into the language of the target CMM.

• **Valysis**

Valysis has put together all the pieces to efficiently link CAD with inspection, without being married to any CAD vendor [Ercole 91]. To use Valysis software the designs and drawing models must be realized using the ANSI Y14.5 "Geometric Dimensioning and Tolerancing". This software is currently handled by IBM.

• **Unigraphics Interface**

A generic CMM interface module, written in GRIP (McDonnell Douglas's

Graphics Interactive Programming Language), enables the user to interact with Unigraphics geometry for CMM programming. It also provides a DMIS output [Schaffer 86].

• **PDGS**

Provides an off-line programming system with a DMIS output. This system has been efficiently used by FORD companies. ALL CMM part programs are designed off-line in Ford proprietary DMIS-compatible FINS (Ford Inspection System) CMM programming module that resides within PDGS. This off-line graphical programming system adds tremendous productivity to Ford's overall inspection operation because the CMMs are devoted to measurement, not program generation [Genest 91].

2.6 Representation of the product model

In the feature and solid model based design paradigm, a part is represented in terms of "features" which represent high-level concepts rather than the geometric primitives used in traditional CAD systems. There are a variety of ways of representing the part model, here the most common ones are explained:

2.6.1 Feature-based model

One of these models novel aspects is to represent dimensioning and tolerancing information within the part model using the feature paradigm. The representation of dimensions and tolerances conforms to the system called "geometric dimensioning and tolerancing" or GD&T. This system is a U.S. standard (ANSI Y14.5M) and is essentially the same as that specified in ISO standard 1101 [Sprow 90]. The part model consists of a collection of data structures representing all information about the part. In addition to form features information is stored on dimensions and tolerances, manufacturing and inspection specifications and process plans.

The design of the part model starts with a starting feature and then other features are attached to this one. Features can be positive, corresponding to addition of material to the already existing design, or negative corresponding to the removal of material. Associated with each form feature there is a set of alternative schemes for dimensioning and tolerancing the feature, both internally and with respect other features. This model is implemented in a variation of the Common LISP Object System (CLOS) and the Concept Modeller [Merat 92].

2.6.2 Solid-based model

Another way or representing the part model is using solid modelling (CSG) within a computer. In this scheme, an object is represented as the set-theoretic (union, difference and intersection)information of simpler objects (primitives). The set-theoretic operators are used to build the require shape from these primitives. The primitives used are half-spaces - surfaces that divide three-dimensional space into regions that represent either solid or air. Using these primitives, a set of bounded shapes, such as cylinders, cuboids and spheres can be represented, and using combinations of these and further half spaces a wide range of engineering components may be modelled. The technique to represent tolerances is to allow the user to attach tolerance attributes to features on the model as it is designed [Walker 92].

2.6.3 CSG and B-Rep models

Feature identification procedures may not be easy if the CSG tree does not contain a "particular feature" (of our specific interest) as one of its "nodes". Identification of that feature (which may be a shape as a part of a much larger model) becomes extremely difficult.

On the other hand the B-Rep data base provides an "evaluated" form of the part and gives an explicit representation of the solid in terms of its lower level entities. Lower level feature based tolerance representation is comparatively easy in this case. Identification of the lower-level features is still difficult because B-Rep maintains all the geometrical and topological information of the entire object in 1-level only, and manipulation of higher level features is difficult because the B-Rep does not record the creation history of the object.

Neither a strictly CSG nor a strictly B-Rep model is acceptable for the representation scheme in a solid model in a tolerancing point of view. The CSG method does not contain information regarding lower level features and its topology, whereas the B-Rep model suffers from the lack of information regarding the higher level features. Therefore, it is preferable to create a hybrid data structure exploiting the advantages of CSG and B-Rep models [Uptal 88].

2.6.4 The STEP model

STEP specifies the so called IPIM (Integrated Product Information Model). The IPIM is separated into partial models according to the classification of the information describing a manufactured product throughout its life cycle. The information units (entities) of each partial model are disjunctive to avoid redundant data storage [Evers 91]. The inspection planning application determines how a component should be inspected by analyzing the geometry and by reference to various rules and algorithms. The resulting inspection plan is then used to generate a machine specific part program to drive a CMM. The key activities in the process are inspection machine planning, inspection code generation and inspection machine control. The data required are dimensions and geometry from the design component and the tool data from the manufacturing information.

In order to make use of such product data the STEP/PDES organization uses the data modelling language EXPRESS [Corrigal 92].

2.6.5 2½D model

A 2½D model represents a component in which surfaces are either parallel or perpendicular to the cutter axis so they may be machined by a two-axis movement normal to the spindle. The shape of a 2½D component can be represented by profiles and associated heights. A high proportion of prismatic components are of this form, because they are relatively easy to design and manufacture. Thus, the data associated with 2½D components is essentially 3D. The 2½D information can be extracted from the CAD model using IGES [Tao 92].

2.7 The inspection planning

The specific problem raised after the part model has been defined (explained in the previous paragraph) is how to use this information to guide the inspection planning process.

The basic strategy of the inspection planner is to generate inspection code fragments which represent the instructions required to inspect individual features. These are then pieced together to create a complete inspection plan.

To calculate the inspection path two basic aims must be followed:

2.7.1 Inspection sequence

There are different criteria to find the inspection sequence, the most common ones will be explained in the following paragraphs:

• **Shortest path:** To find the inspection sequence one of the criteria to follow can be try to find the shortest path possible. This is equivalent to the travelling salesman problem, which is computionally intractable. To solve this problem the path is calculated always moving the probe to the closest point from the last contact point taken [Walker 92], [Merat 92].

• **Groups of similar geometry:** Another solution can be to measure all the accessible features with similar geometric elements after a probe has been chosen. This is based upon efficient considerations for CMM operations. The next step is to find other features which belong to other geometric elements, from inspection point of view, and can be inspected by the current probe. This process is repeated until all accessible items which can be inspected by the present probe have been inspected. The next step is to choose another feature and find a suitable probe, then the same steps are followed [ElMar 87].

• **Based on the tolerances:** Another inspection sequence can be the traditional one; that is the measuring operations are related only to the tolerance of the components. The tolerances will be measured one by one, but this sequence may hide redundancy in measuring operations. To avoid this, the relationships between the measuring operations and tolerances are stored for calculating the tolerances later. In this way the features are measured once, stored and used to calculated all the tolerances applied to them according to the relationship between features and tolerances stored before [Tao 92].

2.7.2 Check possible collisions

As in the inspection sequence, here there are also different criteria to solve the probe collision; the most common ones will be explained in the following paragraphs:

• **Lifting the probe:** One algorithm to detect a free collision path is to find if the straight line the probe has to follow to reach the target point intersects with any of the geometric features in the way. If it does the probe will be lifted higher to a safe plane. This guarantees that the probe path will avoid any collision but large movements to and from the Z-safe plane may be generated [Walker 92], [Tao 92].

• **Going around:** Another solution is to try to go around the surface instead of lifting the probe. This solution gives two possible options. If any one of these options is collision free, a suitable option has been found; if not the process can be repeated for each sub-path. Usually, more than one path will be found, the shortest one can then be chosen. Difficulties arise from the need to deal with complicated probe geometries and to plan paths in three dimensions when the motions cannot all be in a single plane [Hopp 85].

2.8 Feedback

When the Coordinate Measuring Machine (CMM) was introduced in 1960s, it brought about a subtle, yet significant, change in the manufacturing practices of machined parts. The driving force of the development and introduction of the CMM was the machine centre.

When machine tools performed individual operations, the machine operator performed the measuring function and made adjustments accordingly. This was a closed loop situation. When the milling machine was made with additional degrees of freedom (multiple axis) and automatic tool changers were applied, the measurement task became beyond the capability of the machine tool operator.

This change created the market demand for 3-D flexible measuring capability. The complexity of the measuring task and the need for a clean, controlled environment resulted in the CMM being located in the gage room, or what has become the quality control laboratory. **The feedback path was interrupted.** The individual operator who had made the measurement himself to control the process was now operating automatic machine tools without the benefit of continuous measuring data [Bosch 92].

There have been several attempts to restore this feedback:

2.8.1 CMM in the shopfloor

Inspection methods which employ indirect adjustments made manually will soon be obsolete. New production lines with completely automated material flows will soon be installed by many manufacturers; such systems only require flexible quality assurance which can keep abreast of the new level of system automation.

Workpieces will be manipulated in the future by special systems to support machine tools. Data relating to evaluation measurement and corrective action will be collected by a special measurement processor. The experimental incorporation of the CMM typifies the adaption of the measuring technique to the production system [Sostar 88]. The information obtained by the CMM can be linked to data processing units or micro-computers for data recording, statistical analysis and the supply of much more valuable information [Atkey 86], [Wright 86].

2.8.2 Specific software

An example of specific software is the C2C software which was developed at the University of Detroit in order to demonstrate the application of linking CMM and CAD systems. The program reads the coordinates of the points taken by the CMM. These data are received at the PC and recorded by C2C. IGES file entries are then created based on the geometric entity being measured [Kwok 91].

2.8.3 DMIS

Another attempt to link the CMM with the CAD system is the standard DMIS. It provides a way of transferring the information back into the CAD system (see section 2.2).

This background information will enable the reader to understand the work presented in this thesis which will be explained in the following chapters.

Chapter 3 : Extracting Data from the CAD system and displaying onto the screen

3.1 Introduction

Normally free access to the internal data structures of commercial CAD systems is very difficult. Thus in order to extract data from CAD systems, it was decided to use a neutral interface that is available with most CAD systems, as the first step to extract the CAD data to enable a model to be created. The neutral interface chosen was IGES (V5.1).

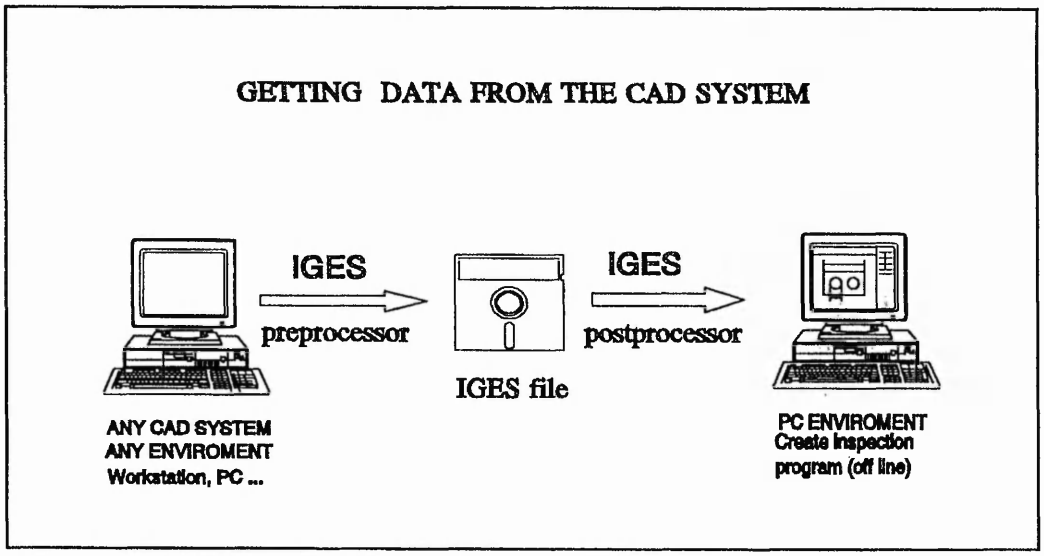


Figure 8 : Extracting data

To extract the geometric information from the IGES file a postprocessor (the software which reads an IGES file) to read a 2½D drawing had to be created. This postprocessor will read the necessary information for inspection planning and store it in dynamic memory for faster and easier access for the downstream activities. To avoid memory problems when reading the IGES file, the information which is not of any use for this application is filtered at this stage. However, it could happen that the IGES file is too big to be processed completely because of memory limitations but this will not be the case for the kind of drawings this application deals with.

The IGES standard has many entities to support the options the existing CAD systems offer. The preprocessors (the programs that write IGES files) use different entities to define the part depending on the application. For example, when a prismatic part is designed using a CAD system, the different faces of the prismatic part are drawn sequentially. These faces are drawn using simple geometry (lines, arcs ...) and annotation. Thus, when the post_processors create the IGES file they only translate these types of entities. Consequently, the entities to be identified are known, i.e those ones which define simple geometry and annotation. These entities are explicitly defined in the IGES file and can be easily identified.

3.2 Why IGES?

The standard chosen was IGES because it is a popular standard and consequently supported by most of the CAD/CAM vendors. IGES is a mature mechanism that provides a stable, standardized, vendor independent format to aid the management and use of data from CAD/CAM systems [Dori 92], [Smith 90]. IGES has gained worldwide acceptance as the most popular method of moving from one CAD system to another and it is generally the first method attempted by developers trying to move CAD data [Mayer 87].

In this particular case, there was the option of choosing IGES or DXF because it was possible to get a CAD drawing in both formats using Unigraphics or AutoCad (the CAD systems provided in the department) and both specification were well documented. The reason of using IGES instead of DXF was because IGES is a vendor independent format whereas DXF was created by Autodesk to transfer AutoCad data. Besides IGES is more popular than DXF and consequently more CAD systems provide it, which increases the number of CAD systems this application is valid for.

3.3 Reading the IGES file into memory

The first step in the extraction of CAD data is to read the IGES file into memory to make the task of processing the entities faster and easier. The second step is to sequentially process the entities that are present in the IGES file.

During the period the IGES file is read into the computer memory, any information which is not of use for this specific application can be filtered. For example of the five sections that comprise the IGES file only the directory entry and parameter data sections are considered; the other three are discarded. Thus memory is only allocated for the necessary data of the directory entry and the parameter data sections present in the IGES file. This directory entry and the parameter data are handled as follows:

3.3.1 Directory Entry

From the twenty fixed fields (Appendix 1) defined in two lines for each entity in the directory entry only five of them are taken into account:

- **Entity type number:** This number identifies the different entities. It must agree with the entity type number in the corresponding Parameter Data record.

- **Status number:** This value contains four pieces of information which are concatenated into a single integer number. The values used in this application are the third and fourth digits. They allow geometric and non-geometric (annotation) entities to be differentiated.

- **Sequence number:** This value indicates in which line of the directory entry the information for a certain entity is located.

• **Form number:** This value indicates an individual interpretation of the entity to be used when processing the parameter data for this entity.

• **Parameter record count number:** This number tell us how many lines are used in the Parameter Section to define this particular entity.

These fields are read into memory for each entity and the rest of them are discarded. All the necessary information from this section is processed and stored in a linked list. It is possible to process the information at this stage because the format is the same for all the entities.

3.3.2 Parameter Data

The parameter data varies with each type of entity. However it always begins with the entity type number, and has a pointer to the corresponding Directory Entry record in columns 66-72. The rest of the information is data such as coordinate values in free format. The information of the parameter section is stored in a linked list of arrays of characters. It cannot be processed at this stage because the format for each entity is different and it would not be possible to store the information in a single linked list. However a filtering process also takes place here because only the parameter data of the entities which are to be processed are read into memory.

The program steps for the above are as follows:

▸ Read the name of the IGES file
▸ Check whether or not there exists a file of that name.
▸ Go through the file until the first line of the directory entry is read. To determine which section the read line belongs to, character 72 of the line is checked.
▸ While the line belongs to the directory entry:

- ◻ Create a new link for the linked list which contains the information of the directory entry:
    - ▸ Read from the two lines which form the directory entry for each entity the four fields which are considered for this application
    - ▸ Allocate the necessary memory to store this information
    - ▸ Add the new link to the list
- ◻ Obtain a new line of the file

The directory entry information is read for all the entities that are defined in the IGES file. However for this application only certain types of entities are considered, therefore when the parameter data section is read, only the records to define these entities will be considered. This is achieved using a pointer to index through the directory entry list and taking into account the entity type and parameter count number in each link, selecting the lines to read and discarding the lines not needed from the parameter section. If there is any entity in the directory entry which is not to be processed, then the link will also be deleted from the directory entry list. The procedure is as follows:

- ▸ Index through directory list and for all the entities:
    - ◻ Index through the parameter data list as many times as the number of lines are in the parameter data for this entity:
        - ▸ If directory entry entity type is an entity to be processed:
            - ◻ Create a new link for the linked list which contains the information of the parameter data.
            - ◻ Read the free format data into an array.
            - ◻ Allocate the necessary memory to keep this information.
            - ◻ Add the new link to the list.
        - ▸ Obtain a line of the file

This procedure is required for two reasons:

- ☞ The structure of the IGES data file is without a sequential order (pointers etc., appendix A). Thus it becomes necessary to organise the data in order for the information to be accessed easily at a later stage.

☞ Also having the information in memory (a linked list of pointers) makes the task of searching much faster.

3.4 The procedure to process the entities

Once, the IGES file is read into memory, the data from each entity has to be processed. The polylines are read first because they are formed by different entities which are previously defined. This way one can avoid the entities which form the polyline being read twice, i.e, as part of the polyline and also as normal geometric entities.

The following steps are implemented in the program:

- ▸ Index through the directory entry list and if a polyline entity is found:
  - ◻ Allocate memory to read the information for a polyline.
  - ◻ Take the information which forms the polyline (explained in the next paragraph).
  - ◻ Add the new polyline to the list of created polylines.
  - ◻ Knowing the number of lines in the parameter section index through the linked list of parameter data.

3.4.1 Processing a polyline (Entity 102, Composite Curve entity)

A composite curve (polyline) is a connected curve that results from the grouping of certain individual entities (E1, E2, E3 ...) into a logical unit. It is defined as an ordered list of lines and circular arcs. The list of entities appears in the parameter data entry. Here, each entity that appears in the defining list is indicated by means of a pointer to the directory entry of that entity. The order within the defining list is derived from the order of the listing of these pointers.

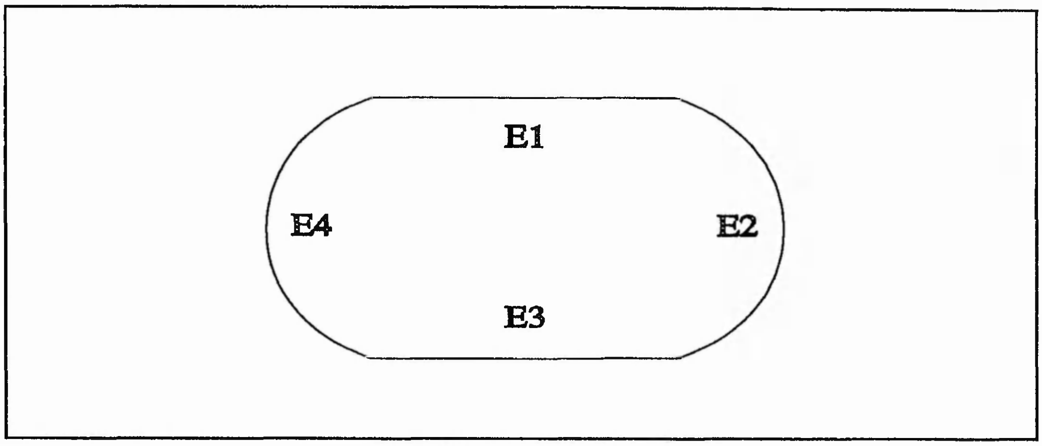


Figure 9 : Composite curve

The information for each element of the polyline can be determined, from the knowledge of the lines of the directory entry, where the entities which form the polyline are present and the number of lines in the parameter section for each entity. For processing a polyline, a pointer is used to index through each list (directory entry and parameter data) looking for the directory entry and parameter data information of the entities which form the polyline. Whenever an entity of the polyline is processed, the links to define it in the directory entry and in the parameter data lists are deleted, so that they can not be read again. Once all the information to create a polyline has been taken, the links for the polyline entity itself are also deleted from the directory entry and parameter data lists. The diagram of figure 10 illustrates the previous steps.

The procedure to get the information of all the entities which form the polyline is as follows:

- ▸ Link the information of the polyline in one line.
- ▸ Obtain the number of entities which compound the polyline.

▸ For each entity which forms the polyline:

- ◻ Obtain the line number where the entity is defined in the directory entry.
- ◻ Place a pointer in the link where the directory entry is defined for this entity.
- ◻ Place a pointer in the link where the first parameter data line is defined for this entity.
- ◻ In case the entity is:
    - ▸ An arc: get the information of the arc and add it to the polyline previously defined.
    - ▸ A line: get the information of the line and add it to the polyline previously defined.
- ◻ Delete from the directory entry and parameter data the links used to defined this entity.

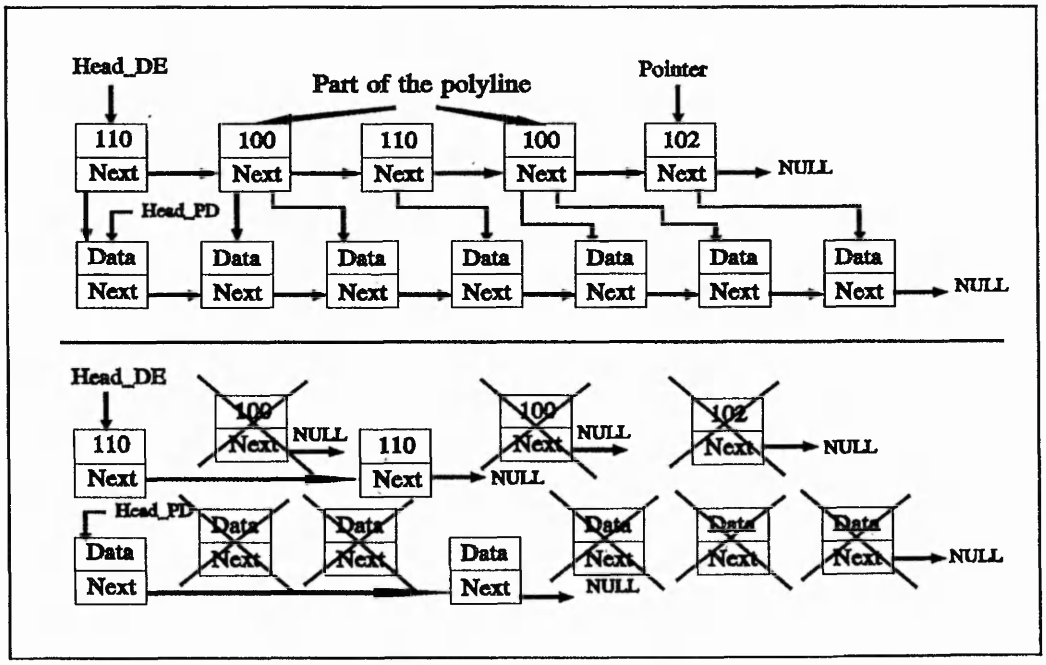


Figure 10 : Reading a polyline

After the polylines have been read, the rest of the entities have to be processed. To do so, a pointer will index through each list to process the entity, getting the necessary information from the directory entry and parameter data for all the entities. Whenever an entity is read, the links to define this entity in the parameter data and directory entry lists are deleted.

The following procedure explains how this conversion occurs:

- ▸ Index through the directory list and for each link:
    - ◻ Detect what type of entity it is and process the information (See following sections).
- ▸ Identifying the number of lines in the parameter section for each entity index through the parameter data list.
- ▸ Delete the links used in both lists to define the entity which has just being processed.

In the following paragraphs the steps to follow for each entity, depending on the type are explained.

3.4.2 Processing a line (Entity 110)

In order to process a line (entity 110), initially the coordinates (x, y, z) of the start point (P1) and end point (P2) of the line, described in the parameter data section (appendix 1) must be read into memory (figure 11).

A line can be a part of the geometry or a part of the annotation. This distinction is set in the directory entry with the status number (appendix 1). The following steps illustrate how the information is processed and read into memory:

- ▸ In case the entity is a line:
    - ◻ Link in one line all the lines in the parameter data to define this entity. The sequence number and the pointer to the directory entry will be excluded.

- Process the line (read the coordinates of the points).
- Compare all the coordinates with the maximum and minimum x and y coordinates and update them in case it is necessary.
- Add the new element to the corresponding list depending if it belongs to the geometry or the dimension.

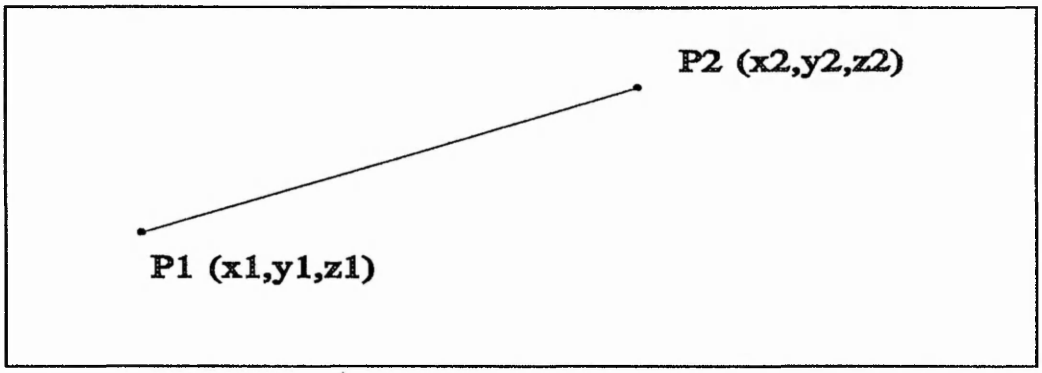


Figure 11 : Entity line

3.4.3 Processing an arc (Entity 100)

In order to process the arc entity (figure 12), the information read are the coordinates (x, y) of the centre (P1) of the parent circle the start point (P2) and the end point (P3), which are defined in the parameter data section (see Appendix 1). If the start point and the end point are the same, then the arc is a circle.

By considering the arc end points to be enumerated and listed in an ordered manner, i.e, start point first, followed by termination point, a direction for the arc can be defined. The ordering of the end points corresponds to that necessary to trace the arc in a counterclockwise manner as defined in the IGES specification. This convention serves to distinguish the desired circular arc from its complementary arc (i.e. clockwise). Another point to remember is that all the points of the arc must be positioned at the same height, i.e, the arc cannot lie in an inclined plane.

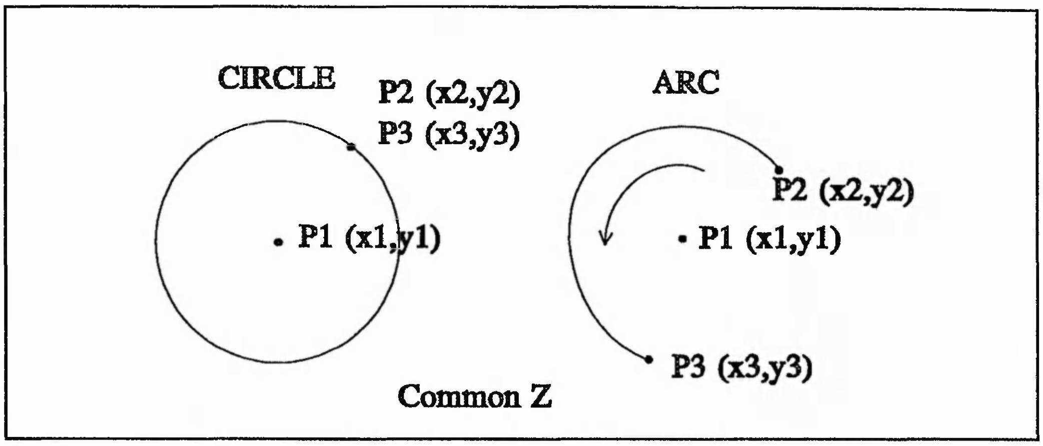


Figure 12 : Arc Entity

An arc can be part of the geometry and part of the annotation. This distinction is set in the directory entry with the status number (appendix 1).

The algorithm for processing an arc is as follows:

▸ In case the entity is an arc:

- ◘ Link in one line all the lines in the parameter data to define this entity. The sequence number and the pointer to the directory entry will be excluded.
- ◘ Allocate memory and process the arc information (z, xc, yc, x1, y1, z1, x2, y2, z2).
- ◘ Compare all the coordinates with the maximum and minimum x and y coordinates and update them in case it is necessary.
- ◘ Add the new element to the corresponding list depending on whether it belongs to the geometry or dimension.

3.4.4 Processing witness lines and hatch (Entity 106)

In order to process the witness lines and hatching lines, entity 106 is utilised. The witness lines are assigned form 40 and hatching lines are assigned form 31-38 in IGES. The form is defined in the directory entry of this entity. The information in the parameter data comes as a list of points (N) and a flag (IP) that defines how the points are to be described. If IP=1, then x, y coordinates with a common z are defined. If IP=2, then x, y, z coordinates are defined and if IP=3, then x,y,z coordinates and the vectors i,j,k are defined for each point.

• **Witness Line entity (form 40):** Coordinates (x, y) with a common z displacement (IP=1) are defined for each point and every set of two points (start point and end point) will form a line segment associated with draughting entities of various types. All the points will be collinear, and the number of points will be odd and at least three (figure 13). Within the Copious Data entity, there will be the location from which the witness line must be maintained. P1 will be the point coincident with the geometry being dimensioned or equal to second point P2 when the location of the geometry is unknown.

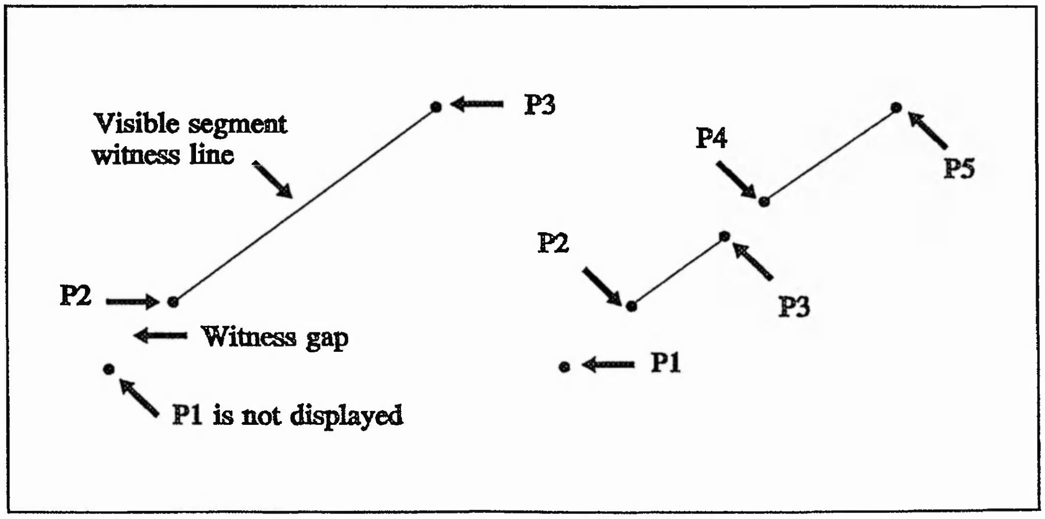


Figure 13 : Witness Line

• **Section entity (form 31-38):** This is used by IGES to define hatching. Two methods for hatch definition are possible and can be described as follows:

1. First Method

Coordinates (x,y) with a common z displacement (IP=1) will be given for each of the N points. The display of the lines consists of solid line segments between the points (Xn,Yn,Z) and (Xn+1, Yn+1, Z) where n=1,3,5,...,N-1. The form number describes different patterns for the section entity.

2. Second Method

The hatched area can also be defined the by using entity 230 which defines the enclosed area of hatching instead of defining the end points of each line.

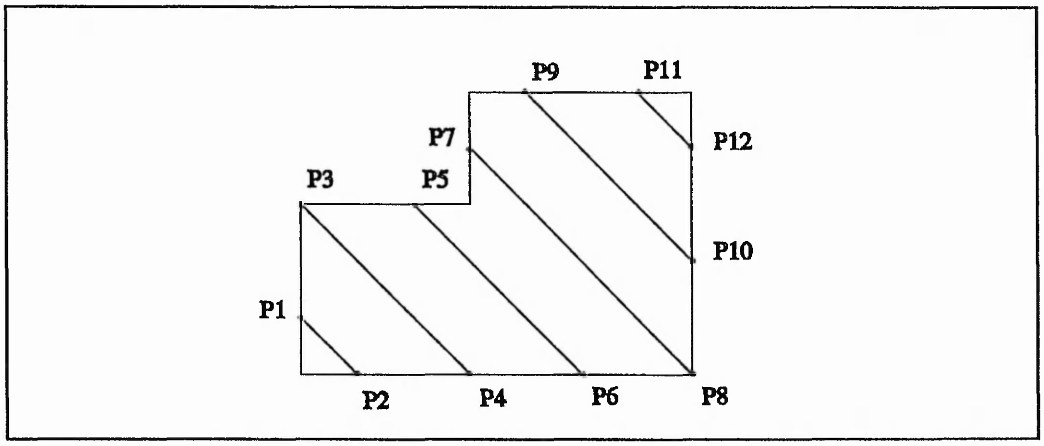


Figure 14 : Section entity

However, even when it is recommended by the IGES specification to use this entity (230), CAD systems still define the hatching by the first method. Since the intention in this program was not to create a CAD system, with the necessary algorithms to create the hatching, the first method was found to be quite useful, and so was employed in this project.

Once the coordinates have been read, they have to be transformed into the different lines which will form the witness lines or the hatching, and stored in a list of auxiliary lines. As the Z coordinate of each point is not necessary to draw them in the screen, it is not stored in the dynamic memory.

The algorithm to process the copius data entity is as follows:

```
▸ In case it is a copius data, check the form number.
▸ Link in one line all the lines in the parameter data to define this entity. The sequence number and the pointer to the directory entry will be excluded.
▸ If a hatch is defined:
    ◘ As many times as lines exist:
        ▸ Allocate memory, process the information (get x,y coordinates of the start and end point), and add the new entity to the auxiliary lines list.
▸ If some auxiliary lines are defined:
    ◘ Allocate memory, get the coordinates of the first line and add the new element to the list of auxiliary lines.
    ◘ As many times as the number of segments are defined:
        ▸ Allocate memory.
        ▸ Get the start coordinates of the line from the previous line created (because it is a coincident point).
        ▸ Get the end coordinates and add the new element to the list of auxiliary lines.
▸ Compare all the coordinates with the maximum and minimum x and y coordinates and update them in case it is necessary.
```

3.4.5 Processing a text (Entity 212)

For processing text, general note entity (212) is utilised. A general note entity (figure 15) consists of one or more text strings. Each text string contains text, a starting point, text size, and angle of rotation of the text. The type of font can also be defined (see Appendix 1).

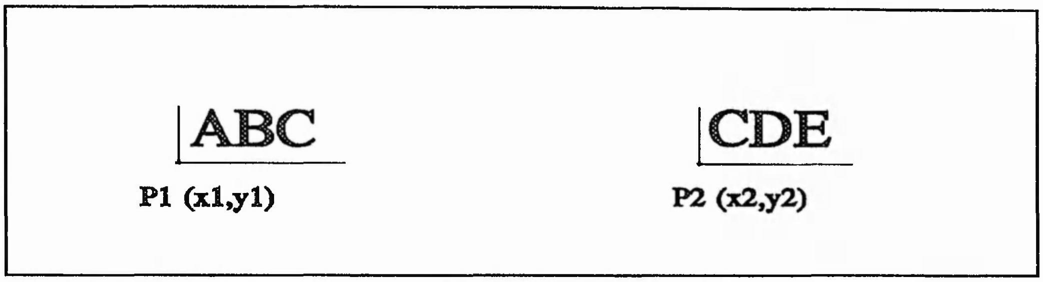


Figure 15 : General note

Although quite a few parameters are defined for entity 212, only the coordinates of the starting point (Pn) of each text and the text itself will be stored. As the drawing is always scaled to fit on the screen, the text size will be always the same. The rest of the parameters are redundant because of the limitations of the available C functions to write a text on the screen.

The procedure for writing text is as follows:

▸ In case the entity is a string:

- ¤ Link in one line all the lines in the parameter data to define this entity. The sequence number and the pointer to the directory entry will be excluded.
- ¤ Process the information (x,y coordinates of the origin of the text and the text itself).
- ¤ Compare all the coordinates with the maximum and minimum x and y coordinates and update them in case it is necessary.
- ¤ Add the new element to the list of texts.

3.4.6 Processing an arrow (Entity 214)

In order to process a dimension from an IGES drawing, arrows have to be processed. Entity 214 is used for this purpose. In the case of arrows, the information given is the height, width, the coordinates (x,y) of the arrow head and the coordinates that will form one or more line segments (figure 16). The first segment begins with the x,y coordinates of the arrow head.

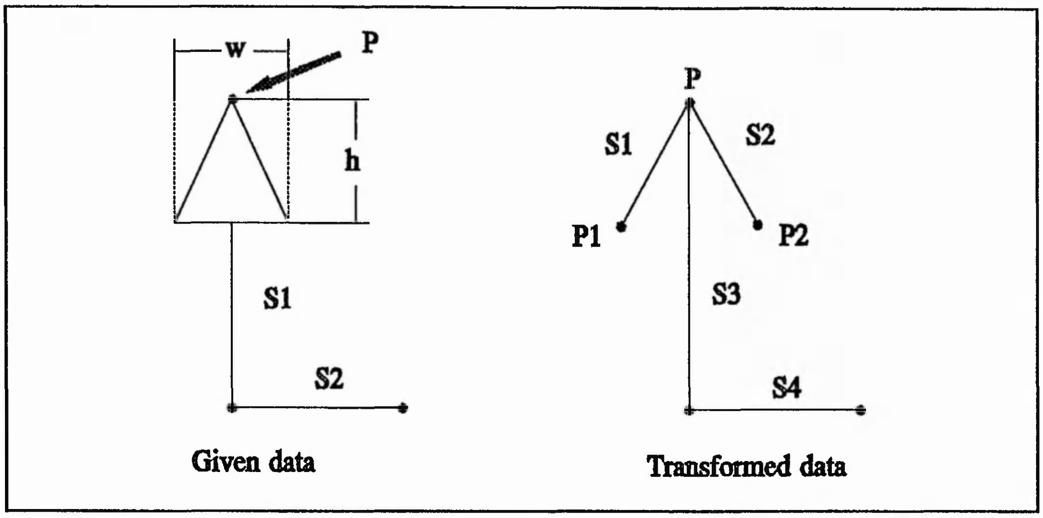


Figure 16 : Arrow entity

The coordinates (P1, P2) of the segments (S1, S2) that form the arrow have to be calculated using the arrow head coordinates, to define the height and the width of the arrow. These two segments will form the actual drawing of the arrow head. The following segments (S3, S4,...) are also read and stored with the arrow head segments in the list of the auxiliary lines. It should be noted that since the segments are consecutive, they share one coordinate which is not duplicated in the IGES file. Thus the same point should be read twice.

The algorithm for processing an arrow is as follows:

- ▸ In case the entity is an arrow:
  - ◻ Link in one line all the lines in the parameter data to define this entity. The sequence number and the pointer to the directory entry will be excluded.

- Process the entity (start and end coordinates of all the lines which represent the arrow).
- Compare all the coordinates with the maximum and minimum x and y coordinates and update them in case it is necessary.
- Add the new lines to the auxiliary lines list.

3.5 Information taken

All the entities drawn in the CAD system relate to a base plane. So even when it is a two and a half (2½D) drawing the height related to the base plane of the read entities is known. It is understood as 2½D components, those ones whose surfaces are either parallel or perpendicular to the cutter axis so that they may be machined by a two axis movement normal to the spindle. The shape of a 2½D component can be represented by profiles and associated heights [Tao 92]. The figures 17 and 18 illustrate the differences between the real part (from IGES) and the partial information extracted from the IGES file.

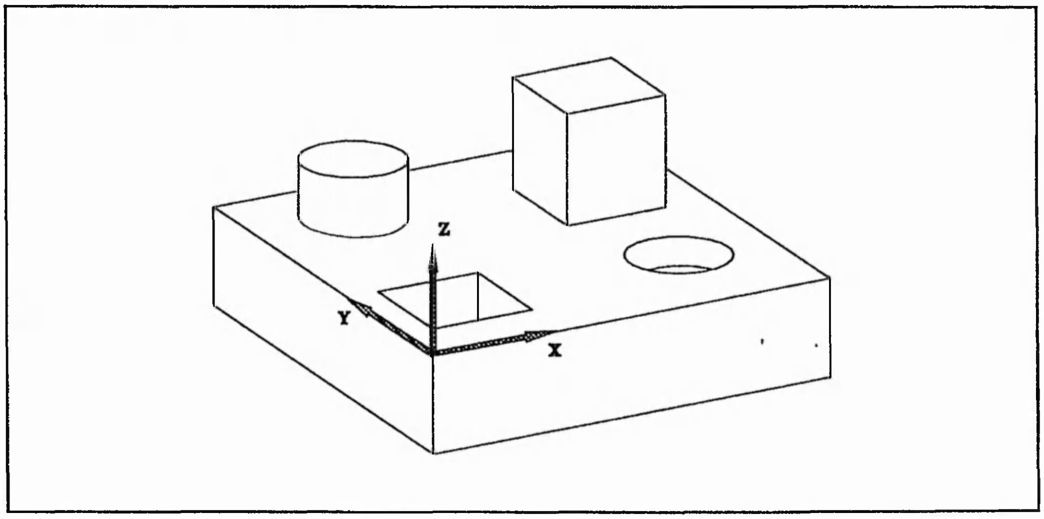


Figure 17 : Real Part

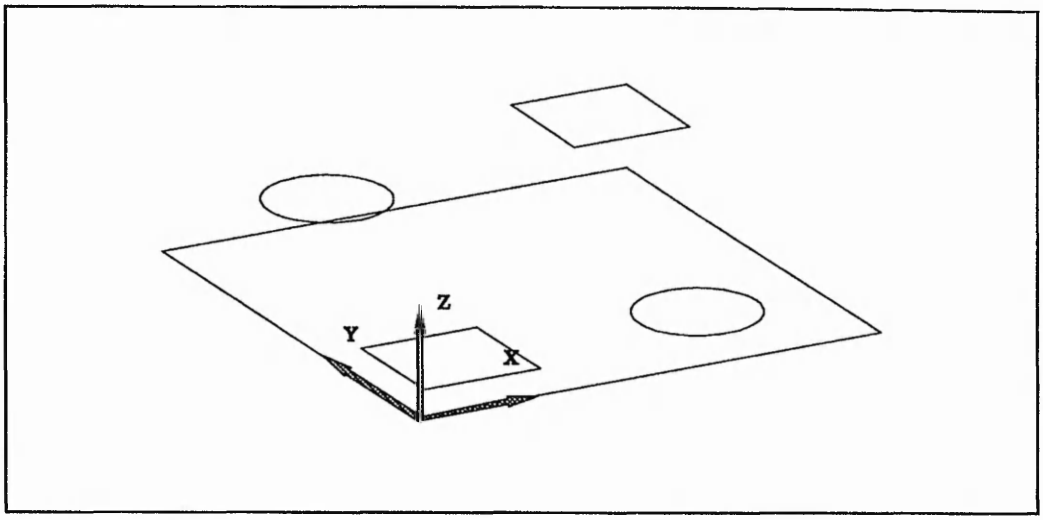


Figure 18 : Information taken

3.6 How the entities are mapped

The information extracted from the IGES file will be translated and stored in six different types of linked lists which will be used later in the program. The list of auxiliary lines, auxiliary arcs and texts will be used to stored the annotation information and they will be always displayed in red on the screen. The list of geometry lines, geometry arcs and polylines will be used to stored the geometric information and they will be always displayed in white on the screen. The information kept in each of the six different types is different and is described as follows:

- **Auxiliary lines:** x,y screen coordinates of the start and end point.
- **Auxiliary arcs:** x,y screen coordinates of the end points and the centre of the parent circle.
- **Geometry lines:** x,y screen coordinates and x,y,z actual coordinates of the start and the end point.
- **Geometry arcs:** x, y screen coordinates and x,y,z actual coordinates of the end points and the centre of the parent circle.
- **Texts:** x,y screen coordinates of the starting point and the text.

- **Polylines:** a list of geometry lines and a list of geometry arcs.

After completing processing of the necessary entities, the next step is to display all this information on the screen and interact with the user. To do so a graphical environment is required.

3.7 Creation of the environment

After creating the IGES post-processor to get the information from the CAD system a graphical environment (figure 19) needs to be created, for the part to be displayed on the computer screen (monitor) and to provide the user an environment to interact with the display. For this purpose, the screen is divided into 4 areas (figure 19) as follows:

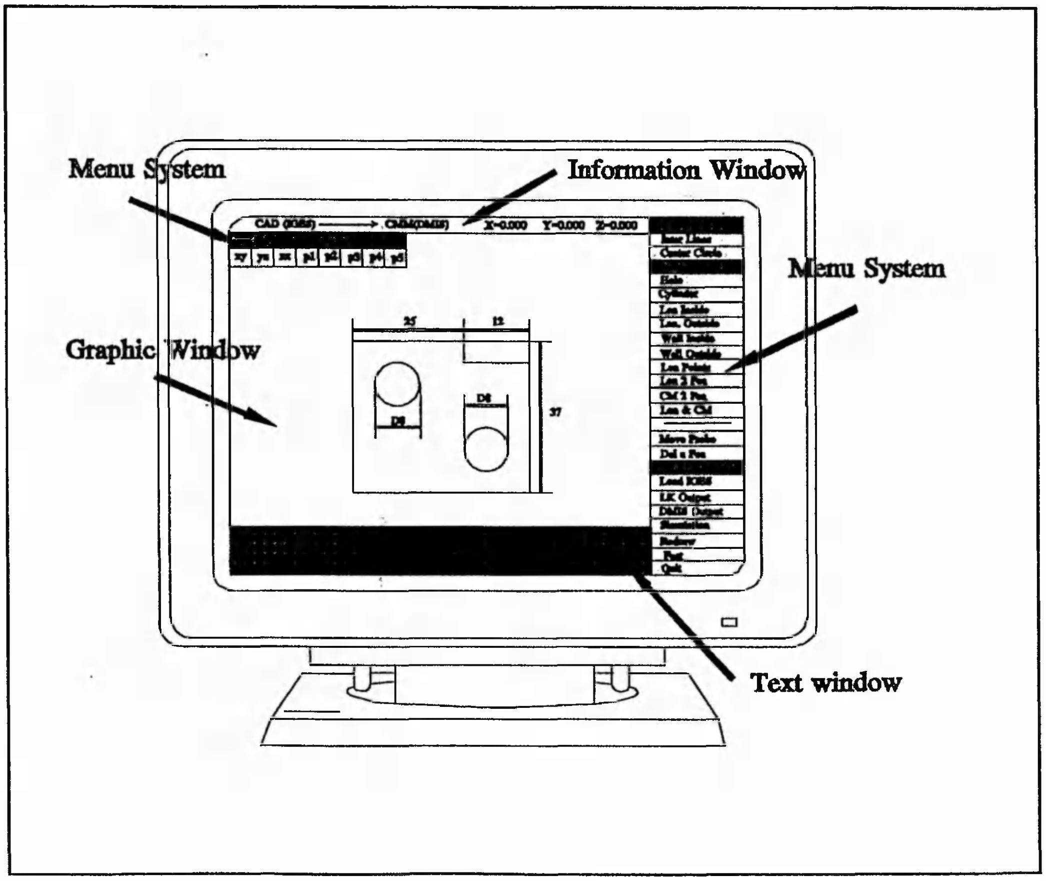


Figure 19 : Environment

- **Menu system:** The user makes a selection from this area, depending on the type of operation needed.
- **Graphics window:** The extracted IGES data is displayed here. The user can use the mouse to select features etc. in this area.
- **Text window:** In this area the system inquiries (text) are displayed and the user's replies from the keyboard are shown.
- **Information window:** This area is for displaying information only and is the only area on the screen that the user cannot directly interact.

All the information the program needs from the user will come from the mouse, the keyboard and the disk.

3.7.1 The menu system

This area consists of fixed menus where the options of the programs are offered. The user makes a menu selection using the mouse as the pointer.
There are two different types of menus:

- **Program option menus:** They are only highlighted when the mouse passes over them, or once they are clicked till the execution of the option is finished.

- **Probe and workplane menus:** There is always one of these highlighted, which indicates the current probe and the actual working plane. The highlighting will change only when a different menu is selected. This menus are highlighted in concordance because probe and working planes are related.

3.7.2 The graphics area

The graphics window is where the part is redrawn and where all the graphical events occur.

3.7.3 The user interaction

There are two different ways for the user to interact with the system, using the mouse and using the keyboard.
All the functions that have been created to handle the environment (windows and mouse) have been done using C++ object oriented programming. Some of the salient functions are described below.

3.7.3.1 Using the mouse

All the menus displayed in the screen are selected using the mouse. Whenever the mouse passes over a menu it will be highlighted and if at that moment the mouse is clicked it will be selected. The menu will remain highlighted until all the information asked by the program is completed.

Depending on the menu that has been chosen the user can be asked to select one of the entities shown on the screen. For instance, if the menu to measure a hole has been selected the user will be asked to click on one of the displayed circles. Only the geometry entities (the white ones) can be clicked.

The procedure for handling the menu system, using the mouse is as follows:

- ▸ While the "Quit" menu is not clicked:
    - ◘ Get mouse event.
    - ◘ If the he mouse is over one of the menus of the right hand side:
        - ▸ Take the highlight of the previous menu off.

```
        ▸ Highlight the new menu and check wether the left hand side
          button has been clicked.
    ▫ If the mouse is over one of the menus of the left hand side and the left
      button has been clicked:
        ▸ Highlight the clicked menu and its correspondent in the other set
          of menus.
        ▸ Take the highlight of the previous menu off.
    ▫ If the left hand side button has been pressed, detect which menu has
      been clicked and execute its corresponding action. If the clicked menu is
      "Quit", go out of the program.
```

3.7.3.2 Using the keyboard

Depending on the menu selected the user can be asked for further information. For instance, when the user clicks the menu that loads an IGES file, the user will be asked to type the name of the file. The option of deleting characters in case of error is also incorporated in the system.

The information is taken as a string of characters and converted later to reals, integers or strings depending on the type of information required. The basic function to read a string from the keyboard is:

```
▸ Get a character from the keyboard and while it is different to the carriage return:
    ▫ If the key pressed is delete and there was any character written:
        ▸ Draw the written text in the background colour
        ▸ Delete from the text string the last character
    ▫ otherwise:
        ▸ add the character to the text string
    ▫ Draw the text into the screen.
▸ return the text
```

3.8 Calculating the screen coordinates

It should be noted that all the coordinates given by the IGES file are actual coordinates. Thus to make the drawing fit in the graphical window defined for the graphical environment, a scale factor is calculated considering the maximum and minimum value of the data in both axes read from the IGES file. So when reading the IGES file these values have to be computed and stored to be used the part is displayed on the screen.

The scale factor is calculated as follows:

- ▸ distx = xmax - xmin;
- ▸ disty = ymax - ymin;
- ▸ fact1=distance in x of the graphical window/distx;
- ▸ fact2=distance in y of the graphical window/disty;
- ▸ fact=the lower of fact1 or fact2;

Once the scale factor has been calculated, another point to consider is that the screen coordinates are always positive, from 0 to 640 in x axis and from 0 to 480 in y axis. Thus if the coordinates of the drawing are negative, the drawing has to be translated. Also consideration has to be given the fact that the graphical window created is smaller than the screen. Thus a bottom margin and a left margin are also computed.

The final screen coordinates are calculated as follows:

- ▸ if the minimum y coordinate is negative:
    - ◻ y screen = bottom margin + (actual y minimum + y) * fact;
- ▸ otherwise:
    - ◻ y screen = bottom margin + (y - yminimum) * fact);

A similar process is used to calculate the x screen coordinates.

These coordinates are calculated at the same time as the drawing is displayed on the screen, and the values are saved in case the part has to be redrawn.

3.9 Problems found

3.9.1 Size of the file

One of the problems can be the size of the IGES file [Vosn 89]. Since the processed information is stored in dynamic memory to make the process of handling the information faster, when the amount of data to be stored is more than the available memory, the program would not be able to read the IGES file completely.

Despite this constraint of not being able to read very big files (more than 6000 line drawings), it was decided to do it this way because usually, the IGES files of simple 2½D drawing (face of a part) are not very big and the available memory is enough to store all the processed information.

A solution to this problem could be to store the processed information into the hard disk. The problem with this solution is that makes the task of handling the information much more slower because the program will have to access the disk whenever it is necessary to handle the information stored for each element which is quite often. This option was not adopted because the available memory is more than sufficient to handle the kind of drawings this application is intended for.

3.9.2 Entities chosen

Another problem was to choose the entities of the original system to be used, and into which entities they will be mapped. The task is made easier with a narrower application domain, as in this case, when there is a frame work (the inspection planning) defining expected entities and the intent of their use.

The main reason behind choosing IGES entities for processing is to achieve a compromise between simplicity of part representation and excessive restrictions in the processor. Only two of all the areas covered by IGES are taken into account. These two areas are geometry and annotation:

- To get the geometry information only arcs, lines and polylines (a combination of the two former ones) entities were taken into account because they are enough to describe a simple 2½D drawing.

- To get the annotation information only the basic entities are used. IGES offers more ways of representing annotation data but as it is only used to clarify the drawing displayed on the screen and all the information has to be converted into simple lines only basic information is taken from the annotation entities.

3.9.3 Tolerance information

Initially, an attempt was made to obtain tolerance information from the IGES file. The problem encountered was that in the IGES file annotation entities were not related to the geometry, so to obtain tolerance information for the required entities user interaction is needed.

In theory, there is in IGES a special associativity form (dimension associativity) to be used in such cases [Nnaji 91][Vos 90]. However this is rarely implemented, so this potential piece of useful information is lost.

This limitation of the IGES file makes it necessary human intervention to get the tolerances required for each feature. Therefore the automatization of the process is then limited.

Chapter 4 : Inspection model and simulation

4.1 Introduction

The second stage, after loading the IGES file and displaying the part on the screen as described in chapter 4 is to generate the inspection planning using the information obtained from the IGES file and through interaction with the user. In other words, to create a model in the computer memory of the features the user wants to measure, the sequence of measuring, the probe and workplane selected, and the coordinate system defined for each of the faces the inspection model is to be.

This is done by selecting the features the user wants to measure. Before starting the inspection planning, the coordinate part system, the work plane and the probe number have to be defined. These values are necessary when the set-up of the CMM to measure the part is carried out.

The different features the user can create within this application are the following ones: hole, cylinder, arc (inside and outside), block, slot, radial distance (inside and outside), perpendicular distance, and the distance and centre point between features. There is also the option of applying tolerances to any of these features.

4.2 Load the IGES file

To create the inspection planning, all the different faces of the part which are to be measured should be loaded into the system sequentially. Once a face is loaded into the system, the inspection planning for that particular face will be created. Other faces could be loaded until all of them have been measured. Before loading the IGES file, the program has to know if the face which is going to be loaded belongs to a new part or to the part the user has been working with previously.

- **If the inspection planning of a new part is to be created,** the user has to click the menu "Begin Part", this action will release all the memory which was being used by the previous part. After the memory has been freed the user will click the menu "Load IGES" which will load the first face of the part that is going to be measured.

- **If another face of the part is to be loaded,** the user after finishing with the inspection planning of the previous face will click the menu "Load IGES". This will allocate additional memory for this face without deleting the inspection planning created for the previous ones.

4.3 Define work plane and select probe

The plane in which the loaded face lies has to be selected. It can lie in five different planes (XY+, ZY+, ZY-, ZX+, ZX-) and five different probes (p1, p2, p3, p4, p5), one for each plane, can be used. These planes are related to the coordinate system of the part that has also to be defined (see figure 20).

First the plane XY or YZ or ZX has to be selected and depending on the chosen probe the direction of the plane (positive or negative) will be determined, creating five different planes. The plane XY and the probe "p1" are selected by default.

All the planes except xy, have been assigned with two probes (The XY- negative is taken as the base on which the part rests). Depending on which probe has been selected the program decides on which of the two possible planes (positive or negative) the face is placed. The probes are assigned to the planes, as follows:

- "XY": This plane has only been assigned with one probe (p1), because the bottom face of the part is that on which the part lies. If the bottom face has to be measured, then the part has to be turned around and a new set-up has to be done.

• "YZ": This plane has been assigned with two probes (p2 and p4). "p4" will be used when the "x" axis is positive and "p2" when the "x" axis is negative.

• "XZ": This plane has been assigned with another two probes (p3 and p5). "p5" will be used when "y" axis is positive and "p3" when the "y" axis is negative.

The probes have to be numbered as shown in figure 20.

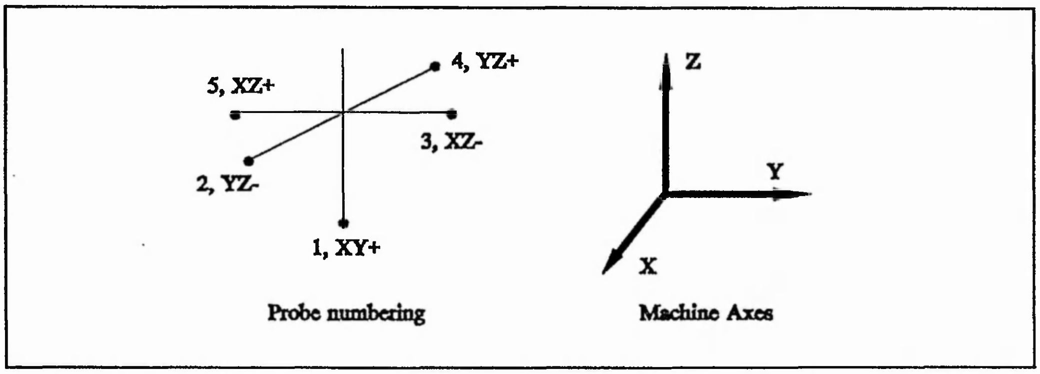


Figure 20 : Probe Definition

The above is implemented in the next algorithm:

▸ Highlight by default plane "XY" and probe "p1".

▸ If any work plane menu has been clicked:

 ▫ Select the corresponding default probe for that plane.

 ▫ Highlight the current plane and probe and return the old ones to the normal state.

When a work plane is selected only the appropriate probe can be selected. This is displayed in a menu during the execution of the program (See figure 21).

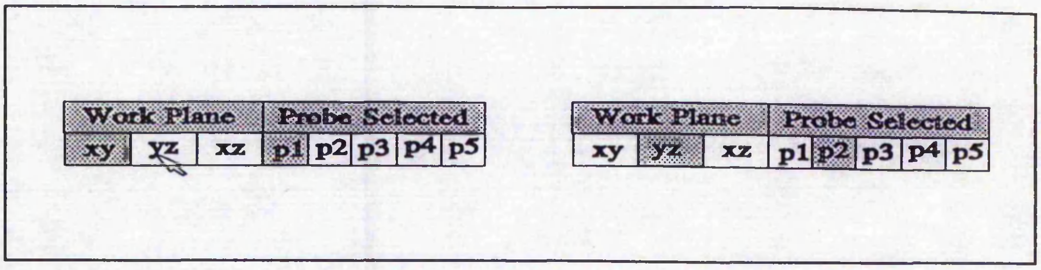


Figure 21 : Select work-plane and probe

4.4 Create the coordinate axes

Once, the part is displayed on the screen and the probe and working plane have been defined the coordinate system (the origin for the part) has to be defined. This can be done in two different ways:

• **Intersection of two lines :** this means that the origin is to be on one of the edges of a feature and the user will be asked to click on the lines that intersect on that edge. The program will not allow selection of two lines which do not intersect (see figure 22).

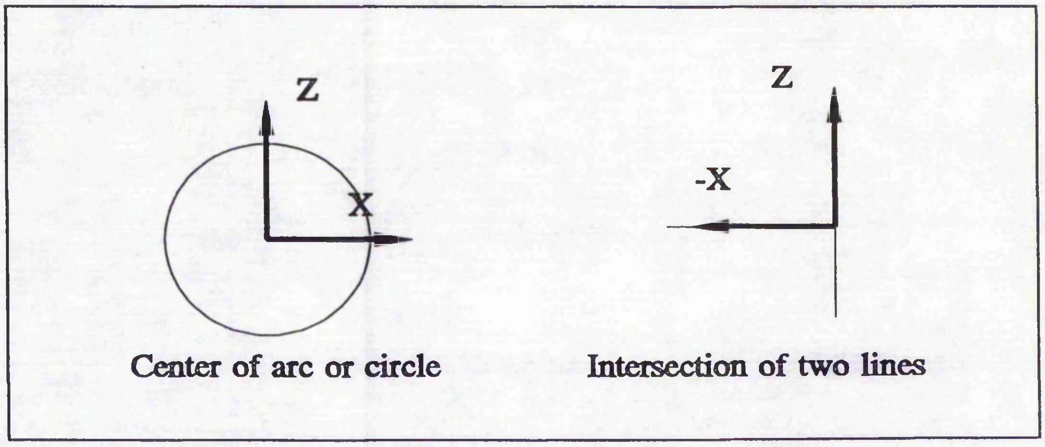


Figure 22 : Create the coordinate system

• **Centre of arc :** this means that the origin is placed on one of the holes, cylinders or arc centres that are on that face. The user will be asked to click on one of the circles or arcs. The program will not allow selection of any other entity which is not an arc or a circle (see figure 22).

The coordinate system will be created when the inspection planning of a new part is initiated. When the first plane is loaded, the coordinate system has to be defined and this will set the zero (Master Datum) of the part. Whenever a new face is loaded the program will ask whether the Coordinate System is going to be translated, and if so the distance in X, Y, Z of the displacement. It should be noted that the Master Datum always has to lie in the plane in which the face to be inspected lies. Consequently, three faces at most could be measured without translating the Master Datum. This is illustrated in figure 23.

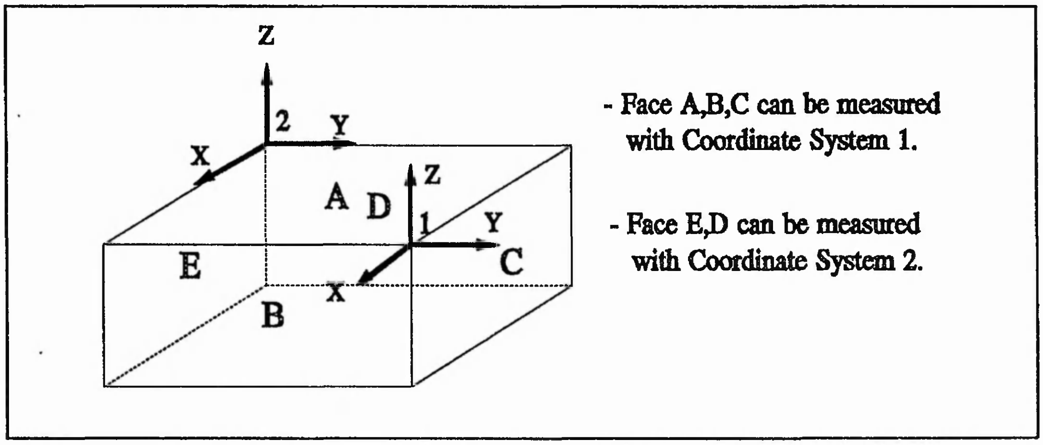

Z
2
Y
X
A
D
Z
1
Y
E
C
X
B
- Face A,B,C can be measured with Coordinate System 1.
- Face E,D can be measured with Coordinate System 2.

Figure 23 : Common coordinate system

4.5 Select the features to be measured

The next step is creating the features the user wants to measure like hole, cylinder, arc, slot (length inside), block (length outside), radial distance (wall thickness), perpendicular length (length true) and distance and centre point between features (see figure 24). All the information that is created in this way will be stored in internal memory, so that the program the program can used the information when different outputs are created.

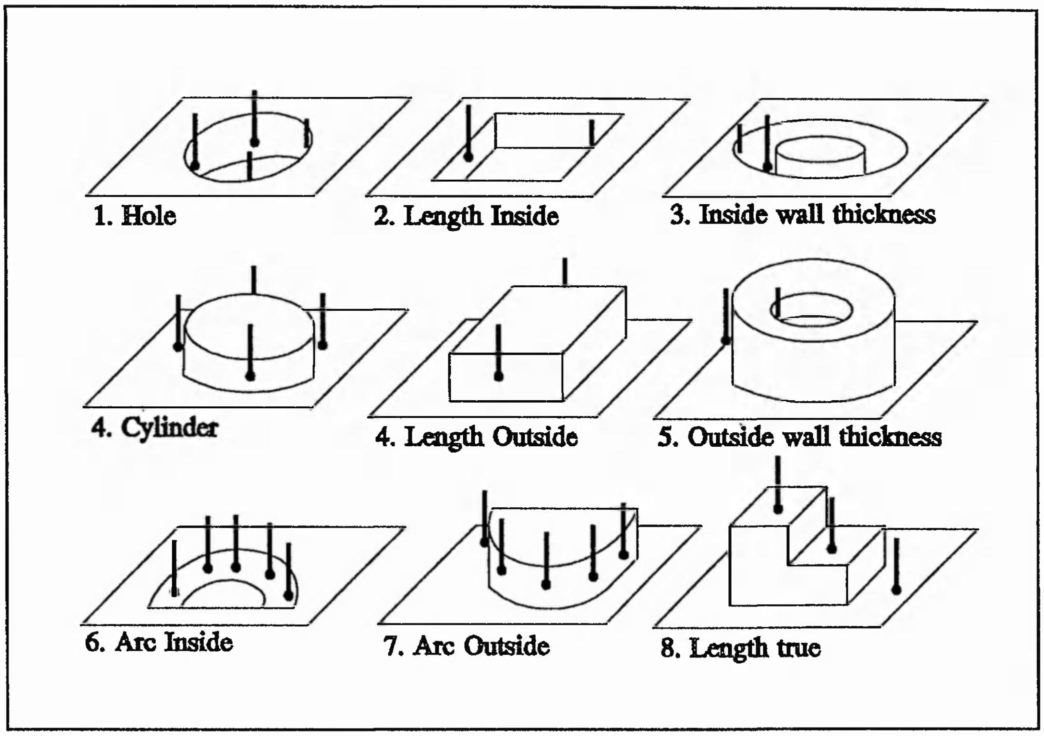


Figure 24 : Types of features

Two different types of features can be created:

- ☞ Features, where the output depends on a single measured feature.
- ☞ Features, where the output depends on more than one previously measured features (e.g : Distance between). The output of these measured features has to be a point and not a distance.

To calculate the coordinates of the new features, the following transformation is needed:

- ▸ X_feature = X_design - X_coor_sys;
- ▸ Y_feature = Y_design - Y_coor_sys;

This enables any changes in the axis system during inspection to be accommodated.

All the created features will be displayed on the screen in a different colour (light blue) to let the user know that is a measured feature and distinguish them from the geometry (white) and annotation (red) entities. These entities cannot be selected again, to avoid measuring the same feature twice, unless is to create a feature between previous measured features or to delete them.

4.5.1 Hole and cylinder

The command hole is used to determine the diameter and centre coordinates of a complete hole. The command cylinder is used to determine the diameter and centre coordinates of a complete boss or pin (see figure 24). The user in both cases will be asked to select a circle in the drawing. The user has to ensure that the chosen circle is a hole or cylinder, as the program has no means of discerning between them.

To select an entity the user has to click with the mouse on a point in the graphical window, close to the entity to be chosen (figure 25). The selected circle will be the one whose radial distance to the clicked point is the shortest one and it is less than a previously set minimum distance (3 pixels) (setup a define); this distance can easily be changed in the program. Since arcs and circles are stored in the same list, the algorithm will look for arcs or circles initially. If an entity found is an arc it will be disregarded. If more than two circles are found at the same distance the first one in the list will be selected. This option allows the user to select one or more circles to measure at the same time, or to skip without choosing any one.

For example in the case shown in figure 25, the distance from the clicked point to the three entities is less than the minimum distance. The smallest one is d2, the with d1 is slightly larger than and smaller than d3. The closest entity is an arc and so it will be disregarded. The selected entity will be then the circle.

The algorithm works as follows:

▸ Index through the list of geometry arcs and only for the arcs which have not been previously selected:

- ◻ Calculate the radial distance to the point: calculating the distance from the point to the arc centre and subtracting from this distance the radius.
- ◻ If the distance is less than the minimum distance, then this will be the new minimum distance.

▸ If the entity chosen is an arc, disregard it

▸ If a circle has been found, draw the circle in the selection colour,

▸ Otherwise: put an error message

The information stored for this feature consist of the actual coordinates of the centre point, diameter and depth of the hole related to the defined coordinate system. Also stored is a pointer to the circle which represents the measured hole in the drawing.

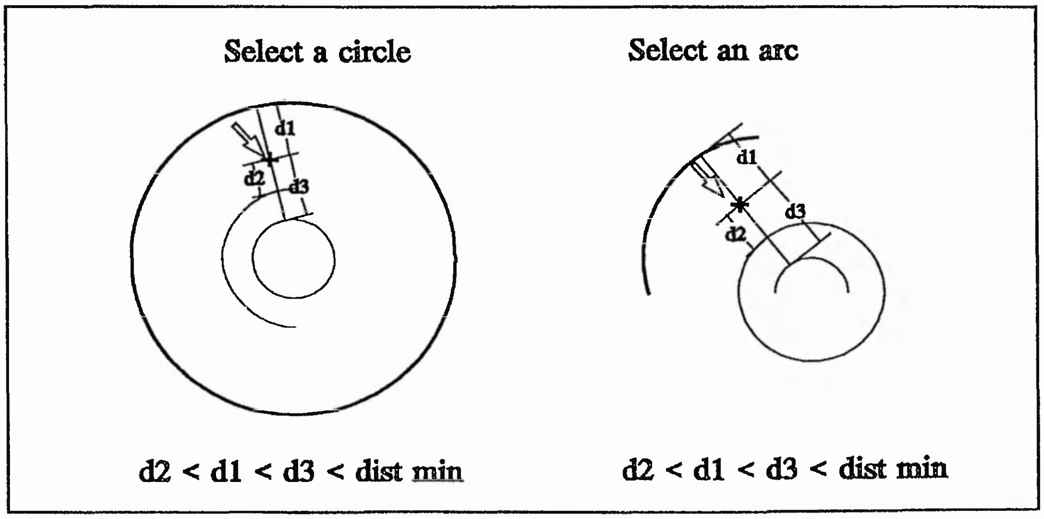


Figure 25 : Selecting a circle and an arc

4.5.2 Arc (inside/outside)

The command "arc inside" is used to determine the diameter and centre coordinates of a partial hole. The command "arc outside" is used to determine the diameter and centre coordinates of a complete boss or pin (see figure 24). The user is asked to select an arc in the drawing but has to ensure that the probe will have enough room to go inside or outside the arc, depending on the case to take the points.

The selection algorithm is the same as the one used for the hole, but has an additional function to determine whether the point is actually within the limits of the arc, or somewhere else in the parent circle (figure 25). The algorithm is as follows:

▸ Calculate the start and end angle of the arc.
▸ Calculate the angle of the clicked point.
▸ In case any of the selected angles is negative, calculate the positive in phase angles (add 360º), making sure that the start angle is smaller than the end angle.
▸ if the angle of the point is between the start and end angle of the arc the point is valid.

For example in the case shown in figure 25, the three distances are smaller than the minimum distance, being d2 the smallest one. However this one is disregarded because it is a circle. The selected one is d1 because it is the less than d3 and also less than the minimum distance.

The information stored for this feature is, the position of the centre point of the arc, the start point and end point, the radius, and a pointer to the arc which represents the measured feature in the drawing.

4.5.3 Length (inside/outside)

The distance between 2 parallel planes are measured either using "length inside" (e.g :slot) or "length outside" (e.g: dog or lug) commands (see figure 24). In both cases the user will be asked to select the lines (representing walls) between which the perpendicular distance has to be calculated. He/She has to make sure there is an inside or an outside wall down these two lines. The program will not allow the user to select two lines which are not parallel. A perpendicular line between the chosen lines will be drawn to represent the feature (see figure 26). The line will go from the midpoint of the shorter of the two lines perpendicular to the other.

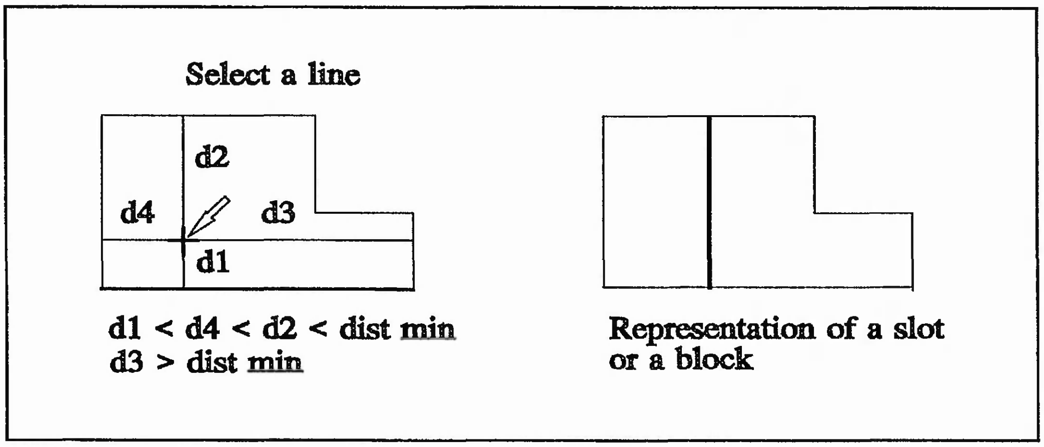


Figure 26 : Selecting a line

The criteria to select a line is the same as that used to select an arc. The line selected will be the closest one to the point within a minimum distance. For example, in the case shown in figure 26, d3 is automatically disregarded because is bigger than the minimum distance. The selected one is d1 because is smaller than the other two (d4 and d2) and smaller than the minimum distance.

The method of calculating the distance between a clicked point and a geometry line is as follows:

- The parameters of the equation of the geometry line a, b, c, are calculated.
- The distance between the point and the line is then calculated:

$$\frac{c+a*x+b*y}{\sqrt{a^2+b^2}}$$

- Calculate if the intersection point belongs to the segment:
    - The offsets (c1, c2, c3) of the perpendicular lines that pass from the start and end point of the segment and from the intersection point are calculated.
        - if the parameter "a" of the equation of the line is not zero:

$$c1=y1-\frac{b}{a}*x1;c2=y2-\frac{b}{a}*x2;c3=y-\frac{b}{a}*x;$$

        - otherwise: c1=x1; c2=x2; c3=x;
    - If c3 between c1 and c2 the point is valid

After two lines have been selected (the same one cannot be selected twice), it will be checked wether they are parallel or not. Checking is done by comparing the slopes of both lines. If they are equal the lines are parallel.

- if -b1/a1 == -b2/a2, the lines are parallel

The line that represents the feature on the screen has also to be calculated. The starting and end point of this line represents where the probe of the CMM is taking the contact points to measure this feature. The calculation is explained in the next algorithm:

- Calculate the length of both the chosen lines
- The start point of the new line is the coordinates of the middle point of the shortest chosen line.
- The coordinates of the end point of this line are calculated adding the

perpendicular distance between the chosen lines to the start point.

4.5.4 Length True

This command is used to calculate the perpendicular distance between two points (see figure 24). The user will be asked to select points in different planes of the drawing. The program has to ask for the height of the points because when the user clicks one point on the screen it can only calculate the x and y actual coordinates from the screen coordinates. The process is just the inverse of the one used to calculate the screen coordinates (see section 4.7). They are calculated as follows:

- ▸ If the actual minimum x is smaller than zero:
    - ▫ actual x = (screen x - left margin)/fact - actual value of x minimum;
- ▸ otherwise:
    - ▫ actual x = (screen x - left margin)/fact + actual value of x minimum;
- ▸ Subtract to this value the x coordinate of the coordinate system;

A mark for each point and a line joining all the points will be drawn in blue to represent the measured feature. Whenever a point is selected it is linked to the list of points in the order of selection. The list of selected points is then re-arranged in increasing depth. The function to add a new point in order works as follows:

- ▸ If the list is empty add the point to the list
- ▸ otherwise:
    - ▫ If the depth of the new point is smaller than the depth of the first point of the list add the point in front of the first point making it the head of the list.
    - ▫ otherwise:
        - ▸ Index through the list of points:
            - ▫ If a point less deep than the new point is found, go out of

the list

- ▸ If it has indexed through all the list, compare it with the last element and if it is smaller link it at the end of the list
- ▸ otherwise, link it in front of the found point which is smaller than the new point.

The information stored is the x and y coordinates, the depth of each point and a pointer to the points which represent this feature in the drawing.

4.5.5 Wall thickness (inside/outside)

The command wall thickness inside is used to determine the inside radial length between two points whereas the command wall thickness outside is to measure the outside radial length between two points (see figure 24). The user will be asked to select two arcs or circles from the drawing. The program will not allow the user to select two arcs which are not concentric. A radial line between the concentric arcs will be drawn in a different colour to represent the measured feature. The line will be drawn radially in the middle of the common area between both arcs (figure 27).

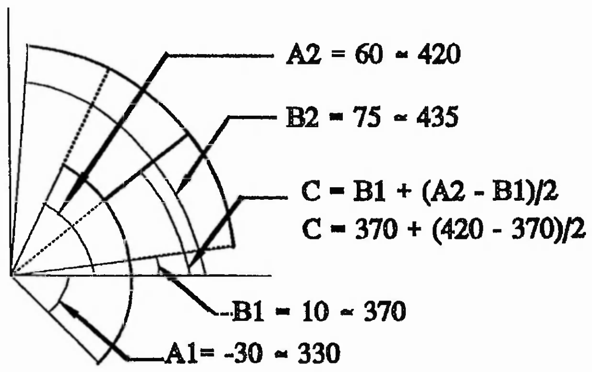


Figure 27 : Radial length

For example in the case shown in figure 27 firstly the angles of the common area are calculated. To do this the biggest starting angle and the smallest ending angle of both arcs (A, B) are taken. To decide this (as one of the angles is negative (A1=-30)), the in phase angles are calculated (A1=330, A2=420, B1=370, B2=435). The starting angle of the common area is B1 and the ending angle A2. The last step is to calculate the bisector of this common area, that is C=B1+(A2-B1)/2. If there is not common are the radial distance cannot be calculated.

The first step in the algorithm to calculate the coordinates of this line is to calculate the start and end angles of both arcs. To calculate the angle where the line is to be drawn, it is a requirement that all angles defining the arcs are positive in anticlockwise direction. Depending on the location of the arcs (if one of them is negative) complementary or in phase angles (as shown in the previously explained example) may sometimes need to be calculated. The algorithm works as follows:

- ▸ Calculate the start and end angle of the first selected arc.
- ▸ Calculate the start and end angle of the second selected arc.
- ▸ Calculate the positive in phase angles in anticlockwise direction.
- ▸ The start and end angles of the common area is calculated, allowing the angle of the bisector to be calculated.
- ▸ Knowing the polar coordinates of the line (radius and angle of both points) the cartesian coordinates are calculated and the line is drawn in a different colour.

The information stored for this feature are the x, y, z coordinates for each point of the radial line between the arcs, and a pointer to the line which represents the measured feature in the drawing.

4.5.6 Pitch Circle Hole and Boss

The command Pitch Circle Hole is used to determine the diameter and centre coordinates of an imaginary circle created through a group of measured holes, whereas the command Pitch Circle Boss will be used for a group of measured bosses (see figure 28). The user is asked to choose a minimum number of four holes or cylinders and a circle through their centres will be drawn in the screen to represent the measured feature. If it is not possible to create this circle the program will disregard the chosen entities.

The first step in the algorithm to calculate the centre coordinates and radius of the drawn circle is to calculate the equation of the circle which passes through the centre of the first three chosen holes or cylinders. If the equation exists it will be checked whether the rest of the chosen circles satisfy the equation. The following algorithm explains this calculation:

▸ The first step is to calculate the equation of the circle that goes through the centres of the first three chosen circles.Thus, the values m (displacement in x), n (displacement in y) and r (radius), have to be calculated. To do so, the values of the centre points of each circle are substituted into the equation. From the resulting three equations the values m,n and r are calculated in relation to the centres of these three circles. The steps taken are:

▸ The equation of the circle is $x^2+y^2+mx+ny=r^2$

▸ Substituting the three points, the values m,n and r are calculated:

$$m=\frac{(y_2-y_1)(x_3^2+Y_3^2-x_1^2-y_1^2)-(y_3-y_1)(x_2^2+y_2^2-x_1^2-y_1^2)}{2(y_3-y_1)(x_1-x_2)-2(y_2-y_1)(x_1-x_3)}$$

$$n=\frac{x_3^2+y_3^2-x_1^2-y_1^2+2m(x_1-x_3)}{2(y_3-y_1)} \qquad r=(x_1-m)^2+(y_1-n)^2$$

▸ Index through the list of the remaining arcs and verify if they satisfy this equation, and if they do draw the arc which represents this feature. If there is not the chosen entities are disregarded.

To create the new feature, the selected features will be unlinked from the list of measured entities and will be defined as a part of the new entity created. The new feature will be linked at the end of the list, so when the program is created the holes or bosses will be measured just before the pitch circle is defined.

The information stored for this feature is a pointer to the measured holes, the coordinates x, y and diameter of the imaginary circle and a pointer to the circle in the drawing which represents the measured feature.

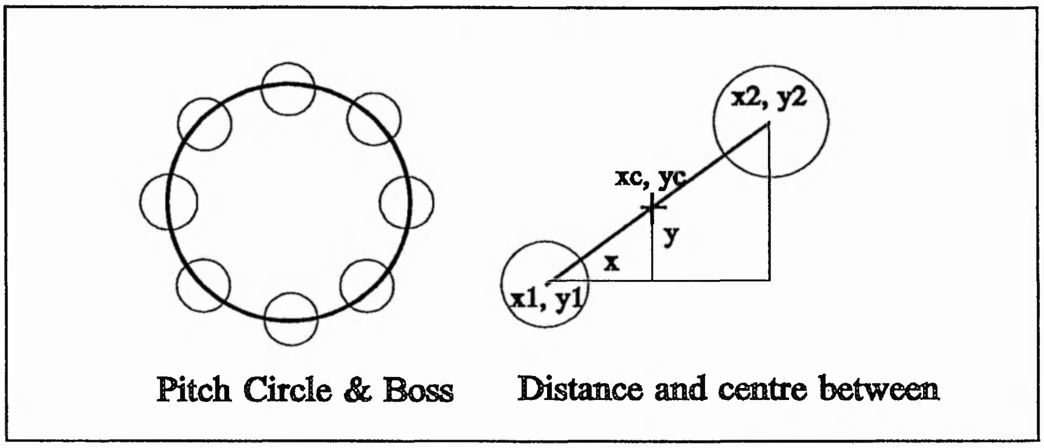


Figure 28 : Pitch circle and Distance between

4.5.7 Distance between two features

This command will be used to determine the distance between two measured features. The user will be asked to select two measured arcs, holes or cylinders. The program will disregard any other feature. A line between the centre of the chosen entities will be drawn in a different colour to represent the measured feature (figure 28).

To create the new feature as in the previous case, the features selected will be unlinked from the list of created features and will be defined as part of the new one created. This feature will be linked at the end of the list so when the program is created the features between which the distance is calculated will be measured just before the command is invoked.

The information stored is a pointer to the measured entities and a pointer to the line in the drawing which represents the measured feature.

4.5.8 Centre point between two features

This command will be used to determine the centre point between two measured features. The user will be asked to select two measured arcs, holes or cylinders. The program will disregard any other feature. A line between the centre of the chosen entities with a mark in the middle will be drawn to represent the measured feature (figure 28).

To create the new feature, the mid point between two features the same steps as for the previous feature are used. This is done as follows:

- First the distance between the two features is calculated.
- Knowing the distance and the x and y increment between features the sin and the cosine are calculated (figure 28).
- Calculating the distance to the middle point (dist/2) and knowing the sine and cosine of the angle, the coordinates of the middle point are then calculated (see figure 28).

$$xc = x1 + x; \qquad yc = y1 + y;$$

The information stored is a pointer to the measured entities and a pointer to the line in the drawing which represents the measured feature.

4.5.9 Move probe

This command will be used when the user wants to change the path of the probe. To do so, the user will be asked to select the measured feature after which the movement of the probe takes place. Then, the user will have to pick two points which will describe the movement of the probe. These points will be transformed from screen coordinates into actual coordinates (reverse of how the actual coordinates were calculated see section 4.7). The probe after measuring the previous feature will follow the defined route to get to the next position. This command will then be treated as a feature in itself and the line representing it will be drawn on the screen.

The information stored for this feature is a pointer to the previous measured feature and the coordinates of the displacement.

4.5.10 Delete Feature

This command will be used when a created feature needs to be deleted. To do so, the program will ask the user to select any created feature from the ones which are displayed in a different colour in the screen. The selection criteria will be to select the closest one to the clicked point within a minimum distance. The feature will be drawn again in the background colour on the screen and it will be unlinked from the list of created features (see figure 29). The memory will not be released to give the user the option of undelete. Various algorithms are involved in this command and the steps are summarized as follows:

▸ The first step is to select a feature. To do this, the user clicks near to any of the lines, arcs or circles which represent the feature. A pointer will index through all the created features and in each of them the following actions take place:

- ◻ Detect if the entity to represent this feature is an arc or a line.
- ◻ If it is an arc calculate the distance between the arc and the clicked

point.

◻ If it is a line, calculate the distance between the line and the point.

▸ Once a feature has been selected it will be unlinked of the list and the memory of the feature previously deleted will be released definitely.

◻ First it has to be determined which type of feature is the one to delete (the previous one).

◻ The memory allocated for that feature is then released.

◻ The feature selected will be unlinked from the list.

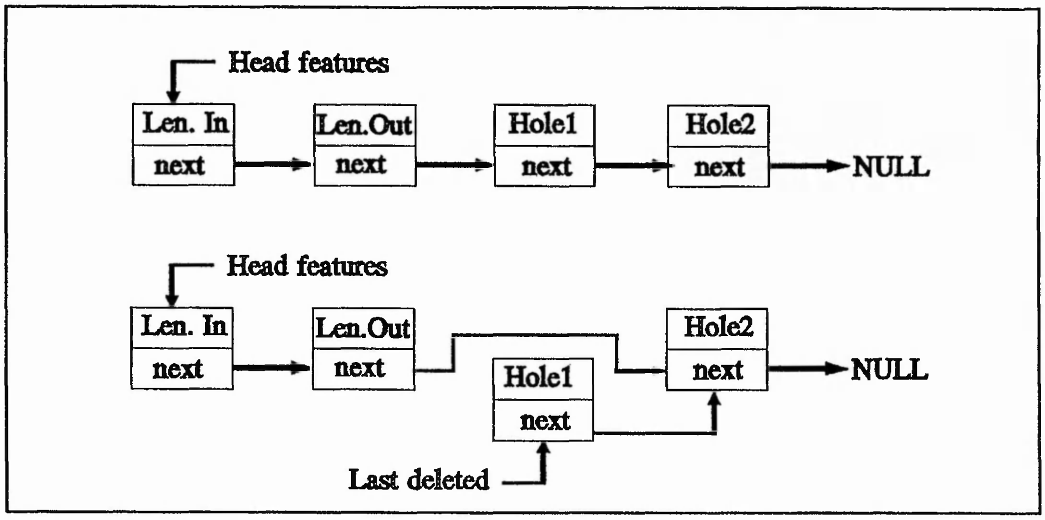


Figure 29 : Delete a feature

4.5.11 Undelete

The program gives the option of undelete the last feature deleted. The feature will be redrawn again on the screen and will be linked to the list of features in the same position it was before the deletion. The algorithm follows the next steps:

▸ Link the feature in the same position it was when it was deleted. It must be considered if it is the first or last element of the list.

▸ Draw the feature on the screen in blue.

4.6 Create tolerances

Dimension tolerances can be applied to all the features created. To do this, once the features have been created, the user will click the menu "tolerances" and will be asked to select a feature. The criteria used to select a feature is the same as in the previous option. The program will ask the user for different values depending on the selected feature. The user can decide the number of values for each entity selected.

▸ Select a feature.
▸ Detect how it is represented on the screen (with an arc or with a line)

The user will be asked for specific values since the feature is known. If the feature is represented with an arc, the horizontal and vertical tolerances of centre point of the arc and the diameter tolerances are required:

▸ Detect the corresponding axis for the horizontal and vertical tolerances and ask for the values
▸ Ask for the diameter tolerance

If the feature is represented by a line different cases can occur and for each one the required values will be different:

▸ If the selected feature is the distance between two points, the required value is the tolerance for this distance.
▸ If it is the length of a slot or a block, the required value will be the tolerance in the corresponding axis, which has to be calculated.
▸ If it is radial length, the required value will be the tolerance of the radial length.
▸ If it is the length in the depth axis, the required value is the tolerance in the depth axis, which has to be calculated.

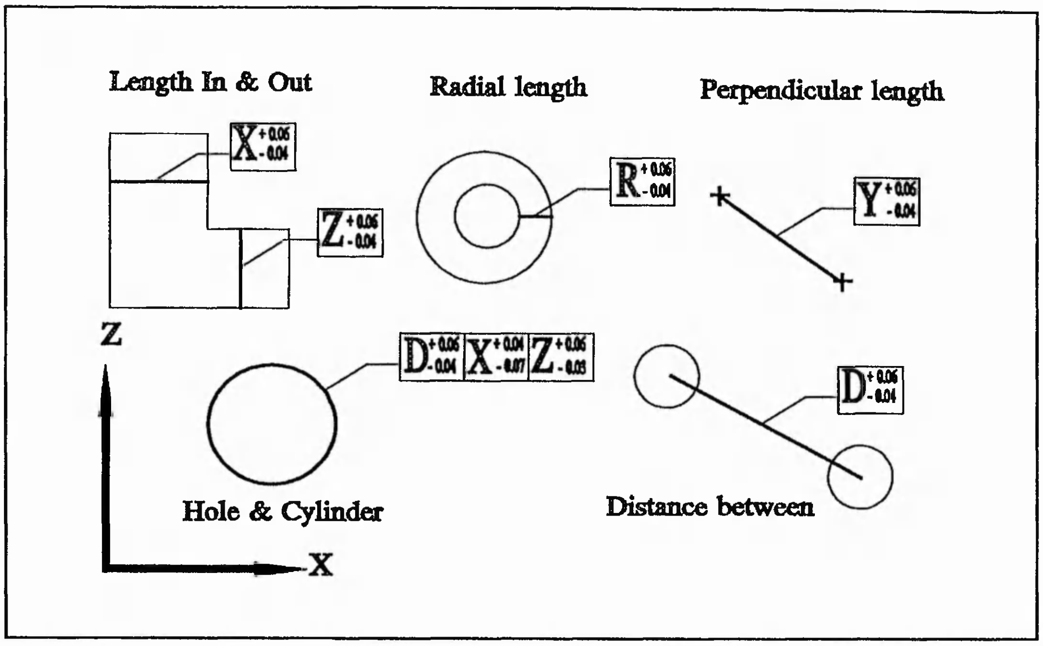


Figure 30 : Available Tolerances

Once, all the tolerance values for an entity have been collected, they have to be displayed on the screen. Only the defined values for each entity will be drawn and the representation will allow the user to identify the tolerances for each feature (figure 30).

4.7 Path planning of measurements

After the user has finished defining the inspection model, the order of measuring the features has to be defined. The user has two options:

4.7.1 Users solution

The features will be measured in the same order as they have been created. This sequence is already known as they were linked in order when they were created. Even when the order of measuring features has been decided by the user, the required points taken for each feature will follow the shortest path possible. In other words, the first point to be taken will be the one which is closest to the last probe position.

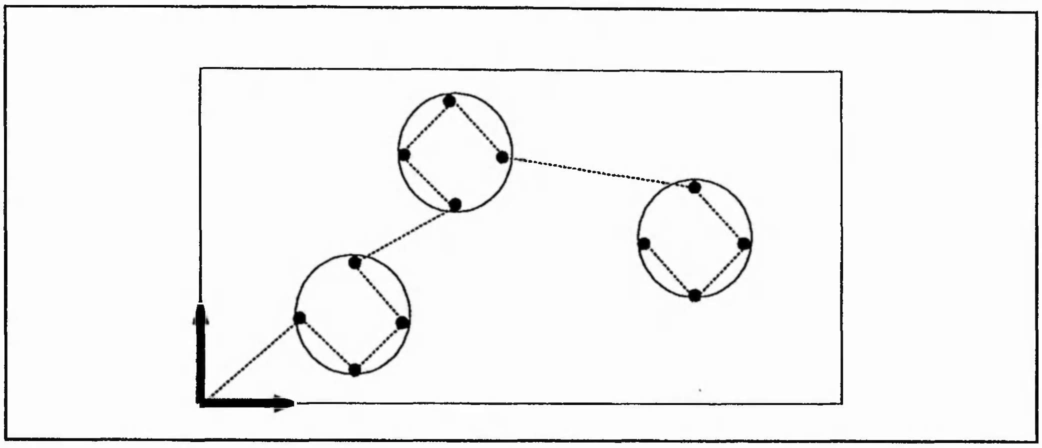


Figure 31 : Selected measurement order

4.7.2 Program's solution

Finding a rule to minimize the total travel of the CMM would be desirable, but it is equivalent to the travelling salesman problem, which is computionally intractable [Walker 92], [Merat 92].

The criteria used to determine the path of the probe is to measure the closest feature to the last position of the probe. To decide which feature is the closest one, first the distance from the last position of the probe to all the contact points is calculated; then the shortest distance of all of them is calculated and therefore its respective feature is determined. This feature is then measured starting from that point. Once, all the points have been taken the rule is applied again from the last position of the probe until all the features are measured (see figure 32). This rule works for the limited features provided in the system.

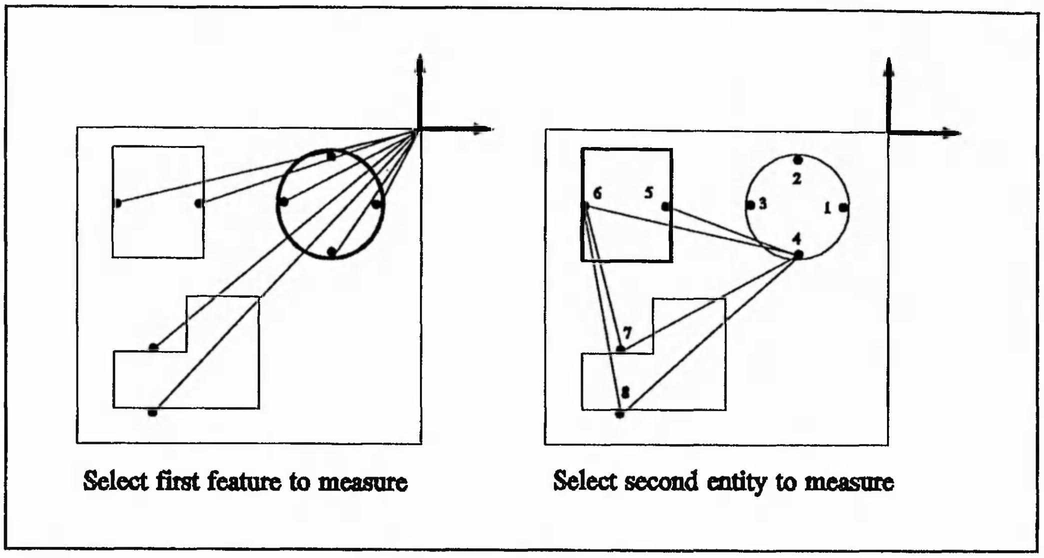

Select first feature to measure
Select second entity to measure
1
2
3
4
5
6
7
8

Figure 32 : Measurement order

The steps followed to order the features following this criteria are the following ones:

- ▸ As many times as features have been created:
  - ◻ Index through the list of created features:
    - ▸ Calculate the distance from the last position to the feature:
      - ◻ All the distances from the last position to all contact points to measure this feature will be calculated. The returned values are the shortest distance of all of them and the position the probe will end up when this feature is measured. It should be noted that the starting point will be the one whose distance to the previous position of the probe is the shortest one and from there following the measuring rules calculate the last position.
    - ▸ Compare if the new distance is shortest than the last distance calculated. If it is:
    - ▸ Keep the coordinates of the last position of the probe, the distance to the feature and a pointer to this feature;
  - ◻ Unlink from the list of created features the selected one;
  - ◻ Link the feature at the end of the new list created;

▸ Link the new list created to the link where the elements to create the inspection program for this specific face are defined (see figure 33).

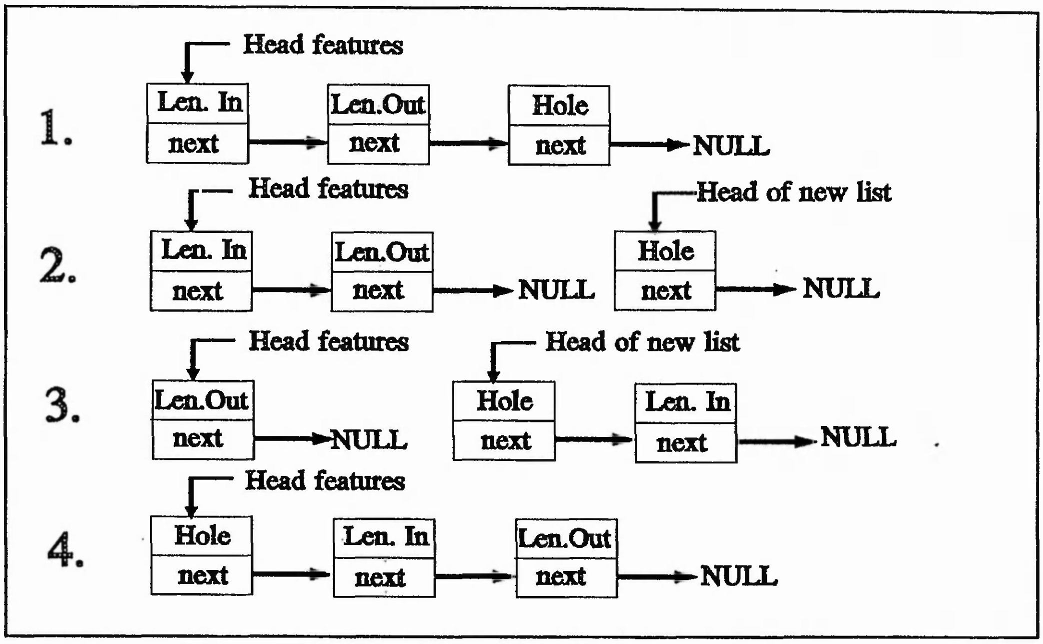


Figure 33 : Programming method

4.8 Simulation of the inspection model

Once the inspection planning has been completed, a simulation of the path the probe will follow to measure all the selected features and a visualization of the X, Y, Z coordinates of the probe at every moment can be created. At the same time depending on the high (travelling) and low (contact) speed of the machine, the estimated time to execute the inspection planning will be calculated. The simulation can be invoked at any moment and it will show the development of the inspection model created up to that stage. Therefore, if the user is not satisfied with the evolution of the inspection model it can be modified before creating an output.

4.8.1 Simulation of the path to follow

The probe will be represented by a small disc which will move along the drawing as the actual probe would do over the part. When the disc is moving in a plane parallel to the actual face it will be represented by a blue disc and when it is moving in a perpendicular axis to the face it will be represented by a green disc (see figure 33). The actual coordinates of the probe will be transformed into screen coordinates using the formula explained in section 4.7.

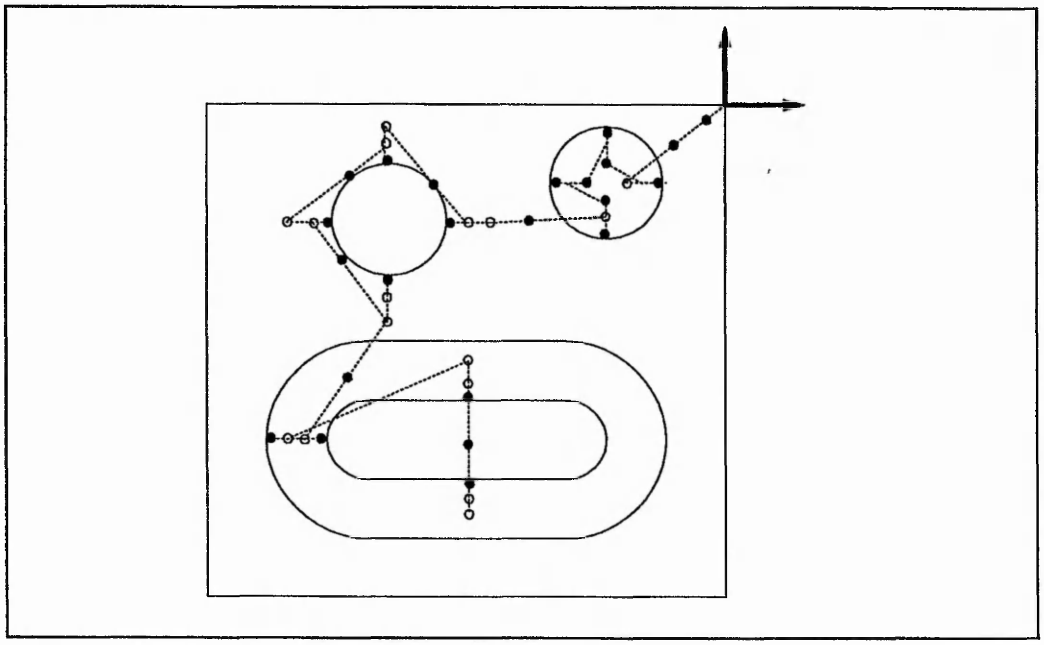


Figure 34 : Simulation of the path

When the probe is moving parallel to the workplane, the coordinates of the disc will be calculated incrementing the horizontal or vertical coordinate of the last position of the probe, depending on which is largest, until the target point is reached. The other coordinate will be calculated using the angle. Knowing the actual coordinates at every point they will be transformed into screen coordinates to represent the simulation. The portion of the screen where the disc is represented will be saved, and restored once the disc has passed that point. Depending on the movement of the probe (travelling or contact) the disc will be driven at a different speed. At the same moment the coordinates of the movement of the disc will be represented on the screen. The algorithm works as follows:

- ▸ While the target point has not been reached:
    - ◻ If the movement is slow, the distance will be added to the total distance travelled at slow speed,
    - ◻ otherwise: it will be added to the total distance travelled at fast speed.
    - ◻ Delay some time, depending on if it is travelling at slow or high speed.
    - ◻ Save the screen, where the probe is to be drawn.
    - ◻ Draw the probe.
    - ◻ Change the axis (vertical and horizontal).

When the probe is moving perpendicular to the workplane, the disc will not vary its position on the screen, but it will change the colour (green) and remain in that position until the simulation of the movement has finished. The probe always will be moved at the travelling speed and the simulation will be created according to that. The coordinate displayed at the top of the screen which represents the perpendicular movement of the probe will change reflecting the real movement of the probe.

- ▸ Add the distance to the total distance travelled at high speed.
- ▸ Save the screen.
- ▸ Draw the probe in green.
- ▸ While the target point is not reached:
    - ◻ Delay the time set for high speed travelling.
    - ◻ Change the perpendicular axis.
- ▸ Reset the screen.

The probe will start at the zero of the part, where the coordinate system has been defined and will measure all the features the user has created. The order of measuring can be done in two different ways:

☞ following the order the user has created the features

☞ following the order created by the program, if the user has clicked previously the command "Path".

In both cases the probe, when it moves to measure the next feature, will go from the last position to the nearest contact point of the feature.

4.8.2 Simulation of the coordinates

The coordinates X, Y, Z of the probe are also represented on the screen and they change according to the movements of the probe along the part. They are displayed at the top right section of the screen.

Knowing what kind of movement is taking place at every moment, horizontal, vertical or perpendicular, and the actual workplane and probe used, the corresponding axis and signs (+ or -) for each movement can be determined.

4.8.3 Measuring time

Another option the system offers is to calculate the time it will take the CMM to measure that particular face. The system will ask the user for the travelling and contact speed and with this information and knowing the movements of the probe the travelling time at slow speed and high speed, and the total time will be calculated.

4.9 Simulation of an existing CMES file

Another option available is to simulate on the screen a CMES file which has been previously created. The movements described in the CMES program will be read and translated onto the screen coordinates using the previously defined formula.

It has to be taken into account whether the movements are at high or low speed to simulate them at the correct velocity. The coordinate system also has to be defined according to how it was specified when the CMES program was created. The program works as follows:

- ▸ Obtain a line of the file until the command "ET" is found.
- ▸ If the command is "DM", change the coordinates of the master datum.
- ▸ if the command is "MA" or "PT":
    - ◻ Calculate the screen axis to represent the movements of the probe, depending on the plane where the measurements have taken place. It must be considered that the movements must always be represented in horizontal and vertical axis on the screen.
    - ◻ As many times as coordinates are defined:
        - ▸ Obtain the number.
        - ▸ Calculate the sign and position on the screen, depending on the axis defined.
    - ◻ If the simulated movement on the screen represents that to take a point it will be represented by a slow speed whereas a travelling movement will be represented by a fast speed.
- ▸ If the command is "RM", reinstall the previous master datum.

It will also be calculated how much time is taken to execute that program and also displayed on the screen will be the coordinates of the probe while it is moving along the part. The algorithms followed to calculate this time and display the coordinates are the same as in the simulation of an inspection model.

Chapter 5 : Creating the inspection program and the feed back

5.1 Introduction

This chapter explains how the probe (CMM) is to be driven along the part for inspection. In other words, how to measure all the different features with their tolerances, how to link the different faces in both programming languages DMIS (standard) and CMES (LK language), and how to create the prototype output. Also an explanation of how the algorithms to detect probe crashing and to move the probe were created, and how to measure a point in both languages.

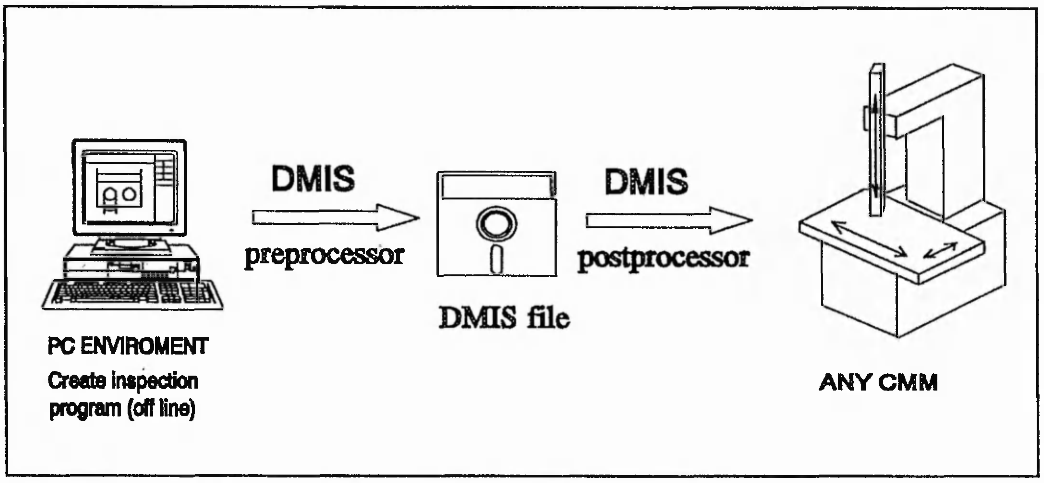


Figure 35 : Creating the inspection program

There are three parameters that should be set in the program that will affect the movements of the probe. These parameters are:

- **Approach distance:** the distance from the point to be measured, from where the probe is to be driven at slow speed.

- **Depth distance:** the distance below the feature surface where the contact points are taken, in a direction perpendicular to the plane in which the feature lies.

- **Probe Length:** this is the distance between the centre of the tip and the centre of the mount. This parameter is important to move the probe from one face to another.

When driving the probe the working plane for which the inspection planning is being created has to be considered. The coordinates when the feature was created were stored as horizontal, vertical and perpendicular coordinates (i.e, this values will be replaced by their respective coordinates, X, Y or Z depending on the workplane) but when the program is created they have to be related to the plane where the measurements are taking place. This relation is different in CMES and DMIS.

- In CMES, before giving any coordinates values, the axis to which these coordinates refer have to be set.

- In DMIS, the three coordinates (x, y, z) have always to be defined. For instance, if a movement of the probe only in one axis direction is to take place, even when only one of the coordinates will change, the three coordinates of the target point must be given.

5.2 Crash detection

To move the probe from one point to another, the following two steps are taken:

- Initially the probe is moved from the last position to the horizontal and vertical positions of the target point. Therefore the probe is moving parallel to the plane.

- The second step is to move the probe perpendicular to the work plane.

Finally, the probe will reach the target point.

Whenever the probe moves parallel to the workplane (step 1), an algorithm to detect if there is any collision is invoked. This algorithm will detect all the possible collisions when the probe is driven from one point to another and will solve the problem lifting the probe over the highest crashing point (see figure 36).

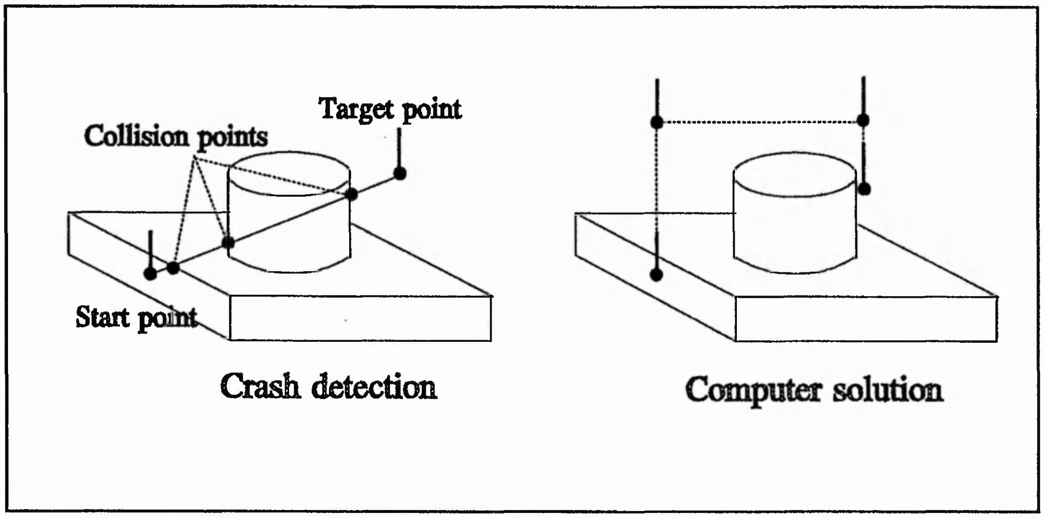


Figure 36 : Crash detection

To detect the crashing points an imaginary line is drawn from the last point "ps" and the target point "pe" (see figure 37). The second stage is to calculate any possible intersection with all the geometric entities of the drawing. If any intersection occurs, the height of the last point will be compared with the height of all the intersecting points. If any of them is higher than the probe it will be moved up over the highest point.

The algorithm works as follows:

- d = height of the last point
- For each of the geometry lines:
    - Calculate intersection point between geometry line and imaginary line

(using the equations of the lines)

◻ If the intersection point is within both segments and the geometry line is higher than "d" then

d=height line + 5.00;

▸ For each geometry arc:

◻ Calculate the intersection point between the arc and the imaginary line (using the equation of the line and the arc)

◻ Check if the intersection point is within the imaginary line and if the height of the arc is higher than "d" then

d= height of the arc + 5.00;

▸ If "d" is different from the last height then:

◻ Move probe perpendicularly to the working plane.

The intersection point between two lines is calculated as follows:

▸ Calculate the equations of both lines (a1,b1,c1,a2,b2,c2).

▸ if (b2*a1 - a2*b1) ≠ 0.0 $y=\frac{a_2*c_1-c_2*a_1}{b_2*a_1-a_2*b_1}$

▸ if (c1*b2 - c2*b1) ≠ 0.0 $x=\frac{c_1*b_2-c_2*b_1}{b_1*a_2-a_1*b_2}$

The intersection points between an arc and a line are calculated as follows:

▸ The equation of the line is y = mx + c, being

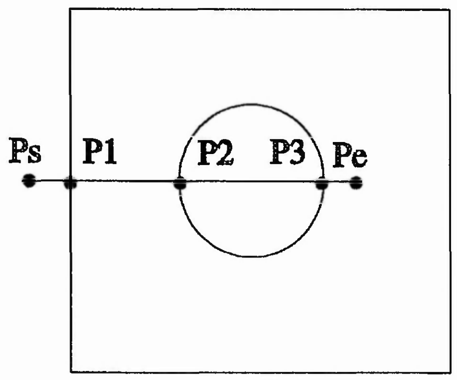
$$m=\frac{y_2-y_1}{x_2-x_1} \quad \text{and} \quad c=\frac{y_1*x_2-y_2*x_1}{x_2-x_1}$$

▸ The equation of the circle is $(x-x_c)^2+(y-y_c)^2-r^2=0$

▸ Substituting the equation of the line in the equation of the circle and organizing in the format Ax+By+C=0, the values of A, B,C are the following ones:

$$A=1+m^2; \qquad B=x_c-m(c-y_c); \qquad C=x_c^2+(c-y_c)^2-r^2;$$

then $x_1=\frac{B-\sqrt{B^2-4AC}}{2A}$; $y_1=m*x_1+c$; and $x_2=\frac{B-\sqrt{B^2-4AC}}{2A}$; $y_2=m*x_2+c$;



```
▸ final depth =Ps depth;
▸ if final depth < P1 depth
      final depth = P1 depth;
▸ if final depth < P2 depth
      final depth = P2 depth;
      ⋮
▸ if final depth ≠ Pe depth
      Move probe up/down
```

Figure 37 : Crash solution

5.3 Move probe

The movements to drive the probe along the CMM are divided in two (parallel and perpendicular to the working plane), as explained in the previous paragraph.

5.3.1 Move probe parallel to the working plane

To drive the CMM from the last point taken to the next one, the probe will try to move parallel to the working plane. The algorithm to detect any collision will move the probe at a higher plane in case it is necessary. The criteria followed to drive the CMM is the same in both languages but the commands are different so the steps followed in both languages are explained:

• DMIS :

To move the probe in DMIS the coordinates (x, y, z) of the target point are required. So knowing the horizontal, vertical and depth values of the point, the working plane and the probe used, the coordinates with the respective signs can be calculated. The command is:

```
GOTO/CART,x,y,z
```

• CMES :

To move the probe parallel to the plane, first the axes have to be defined. Knowing the working plane and the probe selected, the axes can be determined. Secondly, the horizontal and vertical values to move the probe must be given, as follows:

```
#MA, horizontal axis, vertical axis
Horizontal value
Vertical value
```

5.3.2 Move the probe perpendicular to the working plane

This movement will be controlled by the function which detects the crash collision. Perpendicular movements of the probe are made as small as possible and only when necessary to avoid a collision. When a collision is detected the probe will move 5 mm (safety distance) higher than the crashing point. The criteria followed is the same in both languages but the commands are different so the steps followed in both languages are explained:

• **DMIS :**

To move the probe perpendicularly the same steps as to move it parallel are followed. The coordinates of the target point must be calculated:

```
GOTO/CART,x, y, z
```

• **CMES :**

First, the perpendicular axis depending in the working plane has to be calculated and secondly the depth distance with its correspondent sign (+ or -) depending on the direction of the axis must be given, as follows

```
#MA,vertical axis
Depth distance
```

5.4 Take a point

To measure the features the probe of the CMM has to touch a certain number of points on them. The CMM has to be driven at slow speed when the contact takes place, so before calling the command to take a point the probe will be driven at high speed to the contact point minus the approaching distance. The commands to take a point are different in both programming languages and explained as follows:

• **DMIS :**

The coordinates x, y, z of the contact point have to be calculated taking into account the horizontal, vertical and depth values and the plane where the measurements are taking place.

The vector (i, j, k) perpendicular to the face where the point lies has to be calculated as well. The command is:

```
PTMEAS/CART,x,y,z,i,j,k
```

• CMES :

As in the command to move a probe, the axes in which direction the probe is moving to take the point are defined. If the probe is moving parallel to the work plane the horizontal and vertical axes will be defined and if it is moving perpendicular to it the one which defines the depth will be set. These axis will vary depending on the working plane and the probe used. The commands can be:

1. #PT, horizontal axis, vertical axis
   Horizontal coordinate of the point
   Vertical coordinate of the point

2. #PT, depth axis
   Depth coordinate of the point

After measuring a point, as it is going to be used later on to measure a feature, the measured value has to be saved in a variable. The command to do this is:

```
SP,number
```

5.5 How to measure the features

The explanation of how each feature is measured is given here. To achieve this, the different points the sensor has to touch and where they are placed in the feature have to be determined. Also calculated are the points where the CMM will be driven at slow speed, searching for the contact points. These points are calculated by subtracting the (perpendicular) approaching distance from the contact point.

Even though when the machine movements are the same for DMIS and CMES to measure the different features, the commands to drive the machine are different in DMIS and CMES. So the final output will be completely different. It will be explained how to measure the different features in both programming languages.

Also there are some features which depend on previously measured features so that there is no need to take any more points to measure them, but it is necessary to define in the program the features used to construct them.

5.5.1 Hole

To measure a hole the probe has to touch four different points inside it at the same depth. To do this, the probe will move from the last position to the closest point (minus the approaching distance) of these four points. Then, the probe will take four equidistant points in anticlockwise direction as shown in figure 38.

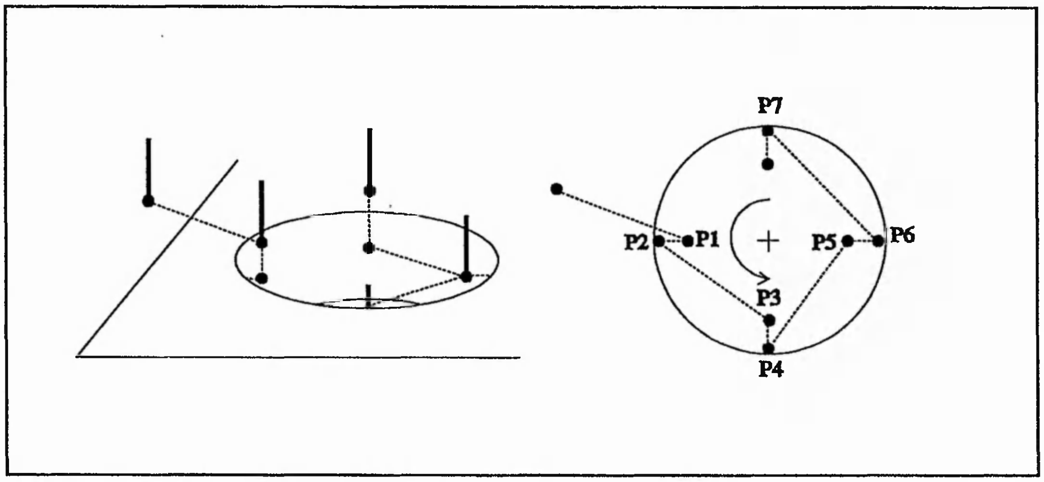


Figure 38 : Measuring a hole

The coordinates of the four contact points are calculated adding and subtracting the radius to the coordinates of the hole centre.
The depth at which the points are collected inside the hole will be calculated subtracting from the depth of the hole centre the previously set depth distance.

Once the sequence of the contact points has been calculated, the points where the CMM begins searching for the contact points can be determined by subtracting from them the previously defined parameter "approach distance".

After the eight points have been calculated the commands to drive the CMM have to be written into a file in both programming languages.

Both programming languages have the option to measure a hole automatically, giving the coordinates of the centre hole and the diameter, i.e the CMM will drive the probe using its own algorithms. This option has not been used because it does not allow one to use the crash detection algorithm and also it is possible that there are some features inside the hole against which the probe could crash if automatic commands are used.

• **DMIS :**

The first step is to define the feature, with the coordinates of the centre point (xc, yc,zc), the vector perpendicular to the plane in which the hole lies (ic,jc,kc) and the diameter of the hole. After the definition of the feature a measurement command to measure a circle is invoked. The number of contact points and the name of the feature to measure are defined. If any tolerance has been created it will be defined after the feature definition (see section 6.6).

```
F(Hole1)=FEAT/CIRCLE,INNER,CART,xc,yc,zc,ic,jc,kc,diameter
MEAS/CIRCLE,F(Hole1),4
Move to the approach distance and take a point (four times)
ENDMES
OUTPUT/FA(Hole1)
```

• **CMES :**

To measure a hole in CMES, the four contact points are taken and saved in four different variables (1,2,3,4). Before using the command to measure a hole (ID) the program needs to know which points are to be used. To do this the command UP is used. If any tolerance has been created it must then be defined (see section 6.6).

```
UP,1,2,3,4
ID, axis in which the hole lies
```

5.5.2 Cylinder

To measure a cylinder the probe has to touch four different points outside the cylinder. To do this, the probe will move from the last position to the closest point of the four (plus the approaching distance). The probe will go down the depth distance and take the first point. Then, it will go up to the clearance point over the cylinder and go to the second point plus the approaching distance in the anticlockwise direction. This process will be followed to take the four points (see figure 39).

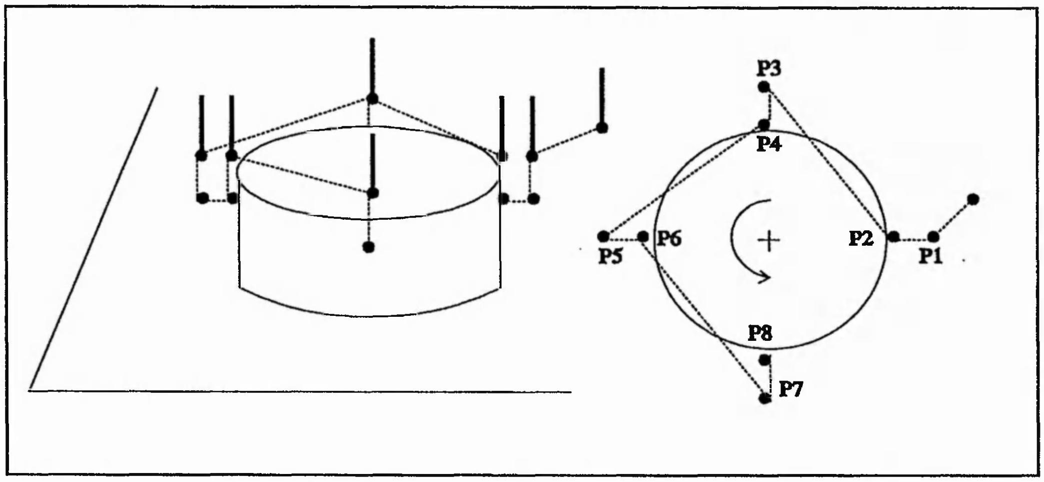


Figure 39 : Measuring a cylinder

The coordinates of the four contact points are calculated by adding or subtracting the radius from the coordinates of the top cylinder centre. The depth in which the points are collected outside the cylinder will be calculated subtracting to the depth of the top cylinder centre the previously set depth distance.

Once the sequence of the contact points has been calculated, the points where the CMM has to start to search for the contact points can be determined by adding to them the previously defined parameter "approach distance". The perpendicular movements to avoid crashing the probe against the cylinder when it moves to take

the next point will be controlled by the algorithm to detect probe collision.
The commands in both languages are:

• **DMIS :**

The first step is to define the feature, with the coordinates of the centre point (xc, yc,zc), the vector perpendicular to the plane in which the cylinder lies (ic,jc,kc) and the diameter of the cylinder. After the definition of the feature a measurement command to measure a circle is invoked. The number of contact points and the name of the feature to measure are defined, as follows:

```
F(Cylinder1)=FEAT/CIRCLE,OUTER,CART,xc,yc,zc,ic,jc,kc,diam
MEAS/CIRCLE,F(Hole1),4
```

Go up, move to the approaching distance go down and take a point (four times).

```
ENDMES
OUTPUT/FA(Cylinder1)
```

• **CMES :**

To measure a cylinder in CMES, the four contact points are taken and saved in four different variables (1,2,3,4). Before using the command to measure a hole (OD) the program needs to know which points are going to be used. To do so the command UP is used, as below:

```
UP,1,2,3,4
OD, axis in which the cylinder lies
```

5.5.3 Length Inside

To measure the perpendicular length of a slot, a point in each inside wall of it has to be taken. The points taken are: one in the middle of the shortest wall and the next one, moving perpendicularly to the opposite wall (see figure 40). The points where the CMM changes the speed will be calculated by subtracting the "approaching distance" from the contact points, and the depth the probe is going inside the slot will be calculated by subtracting from the height of the slot the "depth distance".

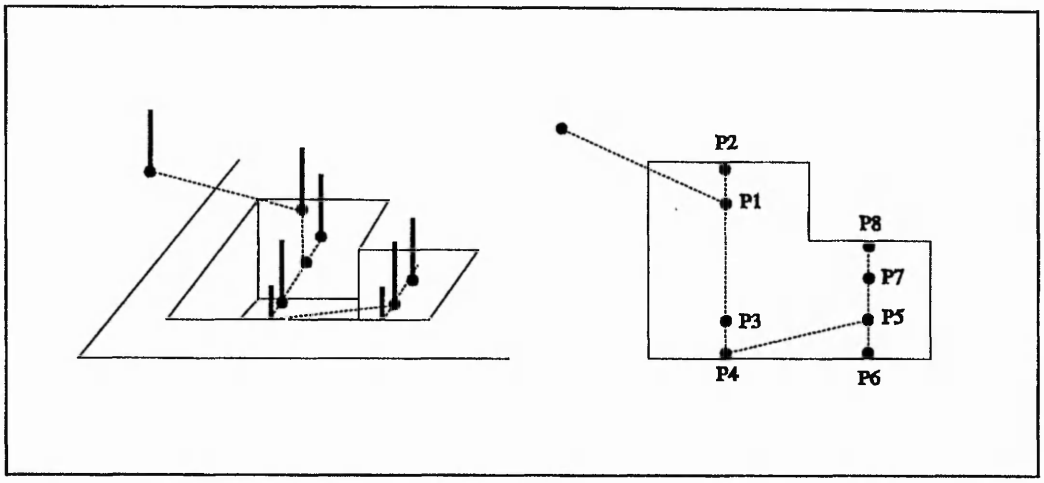


Figure 40 : Measuring a slot

The probe will move from the last position to the closest point of these two minus the approaching distance. It will go down the depth distance, take the first point and move perpendicularly to the opposite wall to take the second point.

• DMIS :

To measure a slot a feature representing the line whose distance is being measured must be defined. The cartesian coordinates of the start and end line where both points have been taken are also defined as features in case the tolerancing commands are going to be used. After the definition of the points a command to take each point will be invoked. The line will be measured as a construction of both points, as follows:

```
F(LenIn1)=FEAT/LINE,BND,CART,x1,y1,z1,x2,y2,z2
F(Point1)=FEAT/POINT,CART,x1,y1,z1
MEAS/POINT,1
Take a point
ENDMES
F(Point2)=FEAT/POINT,CART,x2,y2,z2
```

```
MEAS/POINT,1
Take point
ENDMES
CONST/LINE,F(LenIn1),BF,FA(Point1),FA(Point2)
OUTPUT/FA(LenIn1)
```

• **CMES :**

To measure a slot in CMES two contact points are taken and saved in two different variables. Before using the command to measure the slot the CMM needs to know the number of the variables where the points where saved. The commands are:

```
UP,1,2
LI, axis of the distance of the slot
Distance
```

5.5.4 Length Outside

To measure the perpendicular length of a block a point on each outside wall of it has to be taken (see figure 41). The contact points will be calculated as in the previous case. The points where the CMM is driven to a different speed are calculated adding to the contact points, the value of "approach distance". The depth where the points are taken is calculated by subtracting from the height of the block the "depth distance".

The probe will move from the last position to the closest point of these two plus the approaching distance. The probe will go down the depth distance, take the first point, go up over the box height and move perpendicularly to the opposite wall to the next point plus the approaching distance, go down and take the second point. The crash detection algorithm will control how much the probe has to go up and will move the probe high enough to avoid crashing against the block itself.

• **DMIS :**

The steps followed are the same as in the previous case.

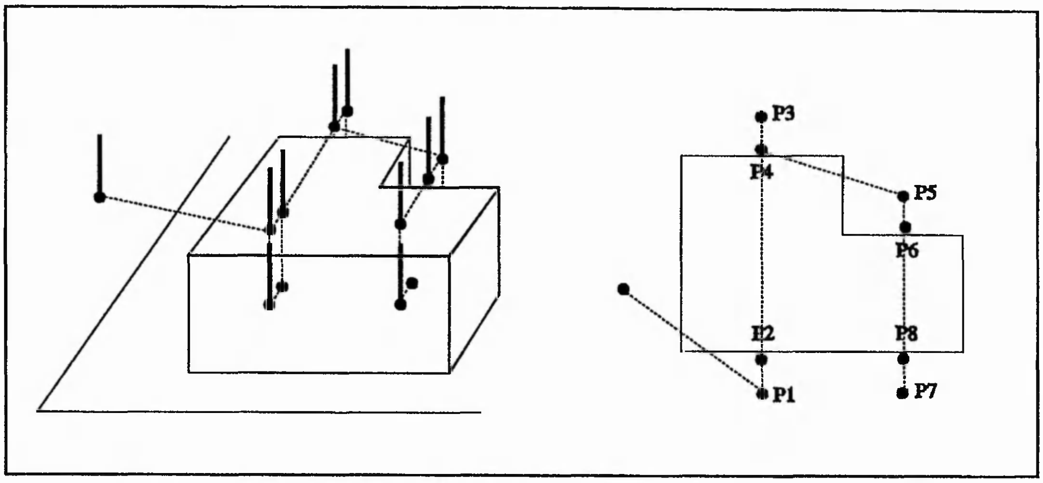


Figure 41 : Measuring a block

• **CMES :**

The steps followed are the same as in the previous case but using the "LO" command instead of the "LI" one.

5.5.5 Wall thickness Inside

To measure the inside radial length between two arcs or circles a point in each wall along the radius has to be taken. The points will be taken one in the middle of the shortest arc and the other one moving the probe along the radius until it touches the other wall (see figure 42). The points where the CMM is driven to different speed are calculated by subtracting from the contact points the "approaching distance". The depth where the points are taken is calculated by subtracting from the height of the hole or arc the "depth distance".

The probe will move from the last position to the closest point of these two minus the approaching distance. Then it will go down by the depth distance, take the first point and move along the radius until a position just before the second point where the speed of the CMM will be changed.

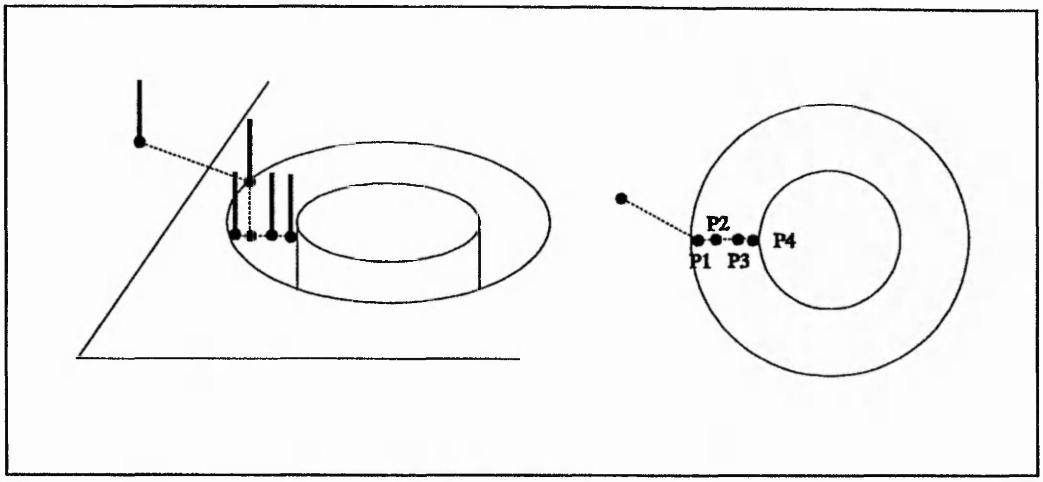


Figure 42 : Inside radial distance

• **DMIS :**

To measure an inside radial distance a feature representing the line whose distance is being measured must be defined. The cartesian coordinates of the start and end line where both points have been taken are also defined as features in case the tolerancing commands are going to be used. After the definition of the points a command to take each point will be invoked. The line will be measured as a construction of both points, as follows:

```
F(WallIn1)=FEAT/LINE,BND,CART,x1,y1,z1,x2,y2,z2
F(Point1)=FEAT/POINT,CART,x1,y1,z1
Move the probe
MEAS/POINT,1
Take a point
ENDMES
Move the probe
F(Point2)=FEAT/POINT,CART,x2,y2,z2
MEAS/POINT,1
```

```
Take point
ENDMES
CONST/LINE,F(WallIn1),BF,FA(Point1),FA(Point2)
OUTPUT/FA(WallIn1)
```

• **CMES :**

To measure an inside radial distance in CMES two contact points are taken and saved in two different variables. Before using the command to measure the slot the CMM needs to know the number of the variables where the points were saved. The commands are:

UP,1,2

WI, + (to compensate the probe radius), plane where the measurement is taken. Distance

5.5.6 Wall thickness Outside

To measure the outside radial length between two arcs or circles a point in each wall along the radius has to be taken (see figure 43). The contact points will be calculated as in the previous case. The points where the CMM has to change the speed will be calculating by adding to the contact points "the approaching distance".

The probe will move from the last position to the closest point of these two, plus the approaching distance. Then it will go down the depth distance, take the first point, and go up over the height of the wall and move along the radius until it reaches the second point plus the approaching distance where it goes down and touches the second point. The algorithm which detects the crash collision will determine how much the probe must go up to avoid crashing against the cylinder itself.

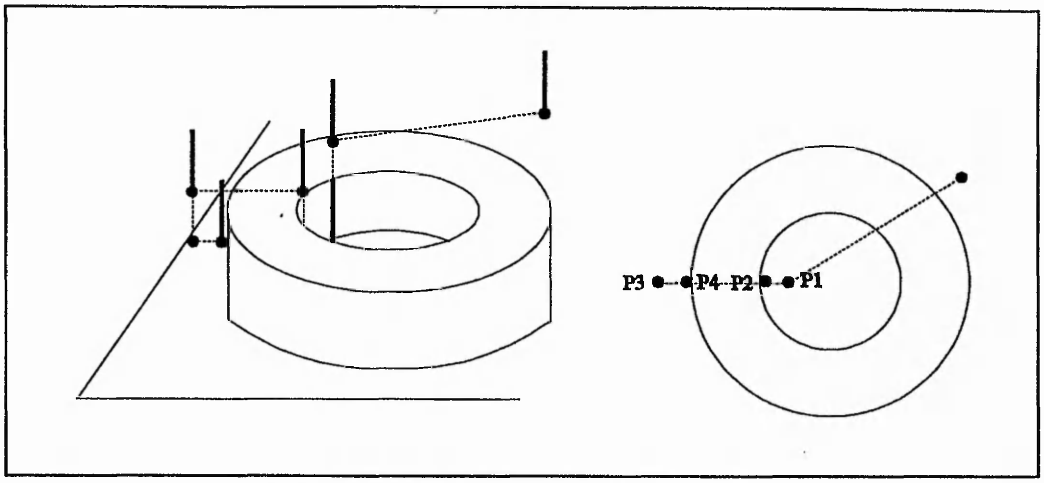


Figure 43 : Radial length

• **DMIS :**

The steps followed are the same as in the previous case.

• **CMES :**

The steps followed are the same as in the previous case but negative compensation is used instead of positive.

5.5.7 Arc Inside

To measure the inside of a partial hole five points at an equidistant angle are taken (see figure 44). The coordinates of these points are calculated dividing the arc angle in six sectors and calculating the start and end point of each of them. The start point of the first sector and the end point of the last one are not taken into account because they are at the extremities of the arc.

The probe will move from the last position to the closest point between the first and the last contact point of the partial hole. Then it will go down the depth inside and take the rest of the points.

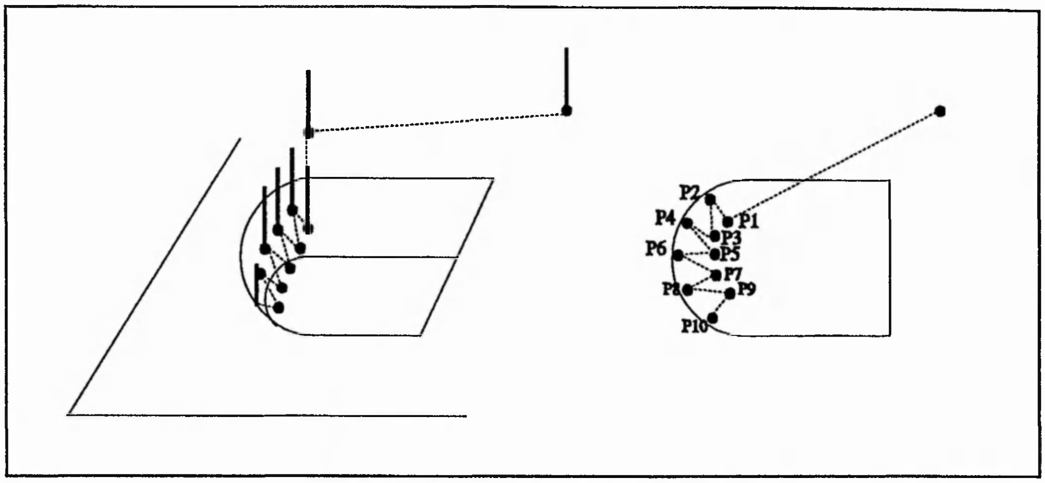


Figure 44 : Partial hole

• DMIS :

The first step is to define the feature with the cartesian coordinates of the centre point, the vector perpendicular to the plane in which the arc lies (ic,jc,kc), the radius, the start angle and end angle which have to be calculated from the centre, and start and end coordinates of the arc. After the definition of the feature a measurement command to measure an arc is invoked. The number of contact points and the name of the feature to measure are defined. If any tolerance has been created it will be defined after the feature definition, as follows:

F(ArcIn1)=FEAT/ARC,INNER,CART,xc,yc,zc,ic,jc,kc,ang1,ang2,rad
MEAS/ARC,F(ArcIn1),5
Move to the approaching distance and take a point (five times)
ENDMES
OUTPUT/(ArcIn1)

• CMES :

To measure an arc in CMES, the five contact points are taken and saved in four different variables (1,2,3,4,5). Before using the command to measure a partial hole (ID) the program needs to know which points are going to be used. To do this the command UP is used, as follows:

UP,1,2,3,4,5

ID, axis in which the arc lies

5.5.8 Arc Outside

To measure the inside of a partial cylinder five points at an equidistant angle are taken (see figure 45). The coordinates of these points are calculated as in the previous case. The difference is that the points where the CMM is driven at slow speed are calculated by adding, instead of subtracting, the "approaching distance".

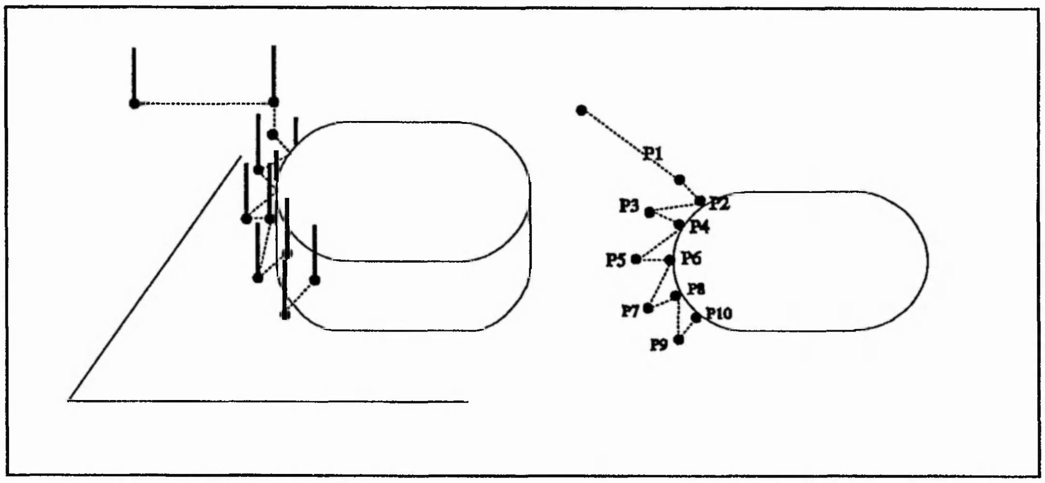


Figure 45 : Partial cylinder

The probe will move from the last position to the closest point between the first and the last contact point of the cylinder. It will then go down the depth inside and take the first point, go back a security distance to take the next one, and repeat the same process until all the points are taken.

• DMIS :

The steps followed are the same as in the previous case, but the command OUTER is used instead of INNER in the feature declaration.

• **CMES :**

The steps followed are the same ones as in the previous case, using the "LO" command instead of "LI" one.

5.5.9 Length true

To measure the perpendicular length all the selected points have to be measured. The points are taken starting from the deepest point to the highest point (see figure 46). The coordinates of these points are calculated by converting screen coordinates into actual coordinates.

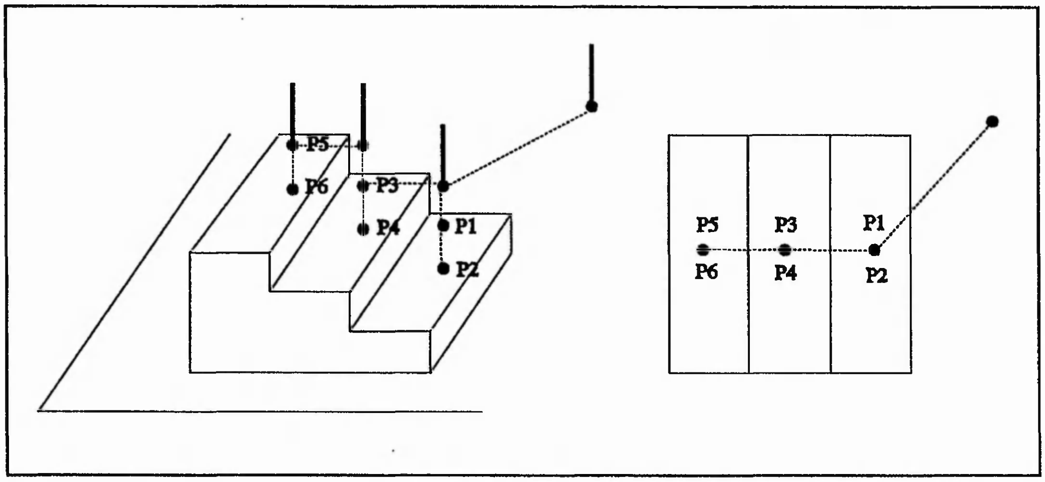


Figure 46 : Measuring steps

The probe will move from the last position to the deepest point of all of them. Then it will go down the height of the point plus the depth distance, take the point and go up the height of the next point plus the security distance, move to the next point and follow the same process until all of them are taken.

• **DMIS :**

To measure the distance in height between two different planes a feature representing this line must be declared. If more than one distance has to be measured (like going up a stair) a feature for each of them must be declared. The coordinates of the starting and end point (x, y, z) must be given. These points are also defined as features in case tolerances are applied. The line will be measured as a construction of the defined points, as follows:

```
F(LenTru1)=FEAT/LINE,BND,CART,x1,y1,z1,x2,y2,z2
F(Point1)=FEAT/POINT,CART,x1,y1,z1
Move the probe
MEAS/POINT,1
Take a point
ENDMES
Move the probe
F(Point2)=FEAT/POINT,CART,x2,y2,z2
MEAS/POINT,1
Take point
ENDMES
CONST/LINE,F(LenTru1),BF,FA(Point1),FA(Point2)
OUTPUT/FA(LenTru1)
```

• CMES :

The first step is to measure all the points from the different planes in order of depth and save them in different number variables (1,2,3 in case three points are taken). The command UP, to know which points must be used will be defined before using the command to measure the distance in height (LT), as follows:

```
UP,1,2
LT, axis for the height
Distance in height
UP,2,3
LT, axis for the height
Distance in height
```

5.5.10 Pitch circle

To measure a pitch circle the probe does not have to be moved because it depends on already measured holes or cylinders and it has already been explained how the CMM has to be driven to measure them but still has to be defined in the inspection program.

• **DMIS :**

First the imaginary circle whose diameter and centre coordinates are being measured is defined. Secondly, instead of invoking a measurement command, a construction command with the features whose centres form the pitch circles is defined, as follows:

```
FEAT(PcirHol1)/CIRCLE,INNER,CART,xc,yc,jc,ic,jc,kc,diam
CONST/CIRCLE,F(PcirHol1),BF,FA(hole1),FA(hole2),FA(hole3),FA(hole4)
```

• **CMES :**

When the features which form the pitch circle are measured the centre points must be saved in number variables. The starting number will be five, because numbers 1 to 4 are used to save the points to measure a hole or a cylinder. Before calling the command "PC" the used points must be defined, as follows:

```
UP,4,5,6,7
PC, plane where the feature is located, number of features
```

5.5.11 Length and centre of measurement between two features

This feature, as in the previous case, depends on already measured features (partial or complete holes and cylinders), but it has to be defined in the part program.

• **DMIS :**

The line which represents the distance between two features must be defined. The start and end points of this line will be declared as features in case tolerances are applied. The measurement will be done as a construction of both features, as follows:

```
FEAT(Dist1)/LINE,BND,CART,x1,y1,z1,x2,y2,z2
CONST/LINE,F(Dist1),BF,(Hole1),(Hole2)
```

The centre point between two features will be represented as a point. The point will be measured as a construction between the two features, as follows:

```
FEAT(MidPt1)/POINT,CART,i,j,k,x,y,z
CONST/POINT,F(MidPt1),MIDPT,FA(Hole1),FA(Hole2)
```

• **CMES :**

When the features between which the distance is calculated are measured the centre points must be saved in number variables because they are used later on. The number will start at least in five because the numbers 1 to 4 are used to measure the holes or cylinders, as follows:

```
UP,5,6
LT, axis in which the measurement is taken
Distance between the points
```

To measure the centre point the number variables are also used.

```
UP,5,6
CM, horizontal and vertical axis
Horizontal coordinate of the centre point
Vertical coordinate of the centre point
```

5.6 Tolerances

Tolerances can also be defined in the part program in both languages. The definition is different in CMES and DMIS but the type of tolerance defined in this application can be described in both languages.

5.6.1 Tolerances in DMIS

Tolerance definitions provide label names to be assigned to each one. The label names are used with the EVAL and OUTPUT statements to associate the tolerance to the feature.

- **Position tolerance**

Specifies the tolerance of the position of a point (x,y,z) in relation to the coordinate system. This tolerance is applied to the centre of a complete or partial hole or cylinder. The tolerances are defined as follows:

T(tol1)=TOL/CORTOL,horizontal axis of the centre, lowtol, uptol
T(tol1)=TOL/CORTOL,vertical axis of the centre, lowtol, uptol
EVAL/FA(Hole1),TA(Tol1) or OUTPUT/FA(Hole1),TA(Tol1)

- **Diameter**

Specifies the diameter tolerance of a feature. This tolerance is applied to a partial or a complete hole or cylinder, shown as follows:

T(DiaTol1)=TOL/DIAM,lowtol, uptol
EVAL/FA(Hole1),TA(DiaTol1) or OUTPUT/FA(Hole1),TA(DiaTol1)

- **Distance in one axis**

Specifies the distance in one axis and the tolerance applied. This tolerance is applied to the distance of a slot or a block or to the distance in height between two different planes, as below:

T(DistTol1)=TOL/DISTB,NOMINL,distance in that axis, axis
EVAL/FA(Point1),FA(Point2),TA(DistTol1) or
OUTPUT/FA(Point1),FA(Point2),TA(DistTol1)

• **Distance between two points**

This tolerance is applied to the radial distances and to the distance between two features, as below:

```
T(DistTol1)=TOL/DISTB,NOMINL,distance between two points, PT2PT
EVAL/FA(Point1),FA(Point2),TA(DistTol1) or
OUTPUT/FA(Point1),FA(Point2),TA(DistTol1)
```

5.6.2 Tolerances in CMES

The nominal and tolerance data may be entered with any of the measurement commands. The tolerance parameter is usually found at the end of a command line and its presence indicates that the command will accept an output nominal and tolerance data.

Two types of tolerances can be defined:

- Equal bi-lateral (represented by "/" after the measurement command): same upper and lower tolerances with different sign , for example --> ±0.05
- Unequal bi-lateral (represented by "//" after the measurement command): different upper and lower tolerances, for example --> $3^{+0.05}_{-0.03}$

The tolerances are defined after the measurement command, and the upper and lower tolerances will be defined after the definition of each value of the feature. If the tolerance defined is "/" only one value is required and if it is "//", the lower and upper tolerance for each value will be required.

• **Tolerances for ID and OD commands**

The tolerances for the horizontal and vertical coordinates for the hole centre and the diameter must be given after the definition (one or two depending on the type of tolerance), only the parameters which have a tolerance will be defined, as follows:

```
ID,Z/          OD,Z//
60,.025        60,-.025,+.025
50,0.25        50,-0.25,+0.25
20,0.25        20,-.03,+.04
```

• **Remaining commands**

The remaining commands only have a value defined, so if the command is followed by the tolerance sign, the value must be given depending on the type of tolerance defined. For example:

```
LI,Y/          WI,+//
10,.010        15,+0.25,-0.4
```

5.7 Move next face

After all the features of one face have been measured, the probe must be changed and the new one must move to the next face to start the measurements. Before selecting the sensor the probe will move up to avoid crashing into any feature and then it rotates to get the new position. The probe will then move from the last position to 10 mm above the new plane (see figure 47). It must be taken into account that the faces must be consecutive.

The algorithm to move the probe to the next face follows these steps:

▸ Calculate the coordinates x,y,z of the last position of the probe.

▸ The coordinates of the target point (10 mm above the plane) are calculated adding 10 mm to the height of the new plane. Depending on the plane it will be x, y or z. It must also be considered whether the master datum is going to be translated or not.

▸ After calculating the coordinates of the target point, the horizontal, vertical and height coordinates are calculated and it must be checked if any collision is likely when moving the probe from the last position to the target point.

▸ The new sensor is selected.

▸ Finally, the probe moves to the next face.

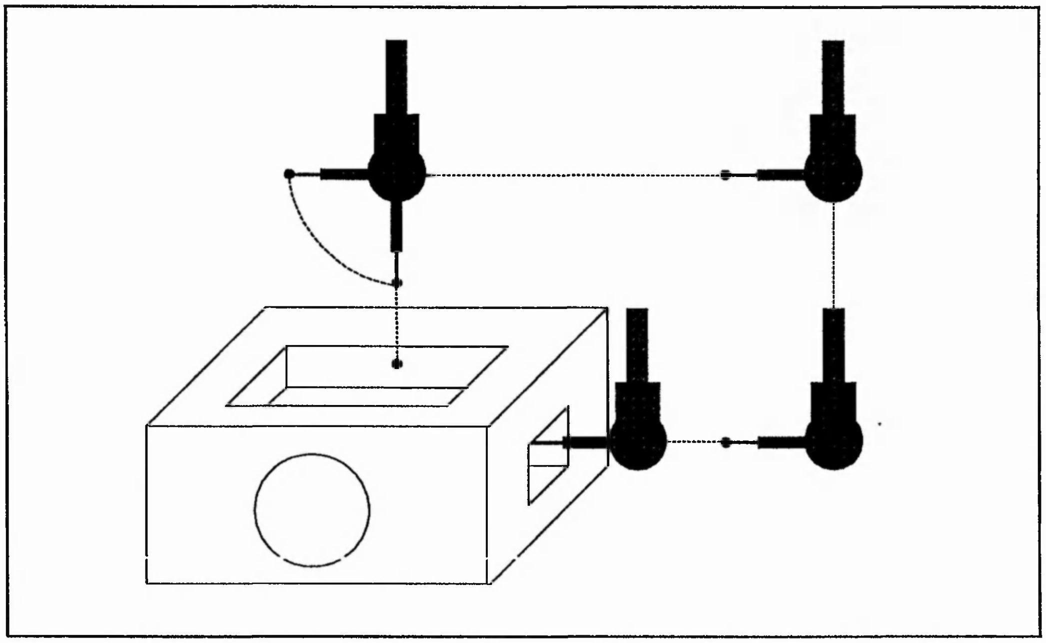


Figure 47 : Move next face

5.8 Prototype part program

The user has the option to create a prototype part program. This program will ask the operator in the CMM whether the next feature to measure is what it was supposed to be. For example, if the user had created a hole feature when the inspection model was created, the probe will be placed above this feature in the CMM and the operator will be asked whether it is a hole or not. If the operator answers negatively, the program will omit this measurement and will move to the next feature where the same procedure will be followed (see appendix 4).

To create this prototype program three different macros or program sub-units had to be created for each feature:

• The first one is the question macro. The CMM programm asks the user if that specific feature is what was defined when the inspection model was created. Depending on the answer given the CMM program executes a different macro.

• Another macro defines the steps followed to measure the feature. This macro is used when the operator answers positively.

• The third macro defines what to do if the feature is not to be measured. Before the operator answers the question, the probe is placed in the first contact point of that feature, but at a safe height. The next movement is to move the probe to the last contact point of the feature also at a safe height. The crash detection algorithm will also be applied to this movement. The reason for this movement is to allow the program to follow the same steps even if the feature is not measured.

This prototype program was created to avoid errors in the CMM. The problem is that once the inspection program has been created the system does not have any control over what is happening in the CMM and if something non-expected occurs (the probe crashes against something, or it does not find a contact point where expected) the program would not be reliable any longer. Even when the CMM has the option to recover on error, the inspection program should stop and the error cause should be found out. The problem is that there are lots of reasons that could cause an error and it is impossible to take all of them into account when the program is being created (see section 7.10). Therefore, there is a limitation of the possible errors that can occur in the CMM that can be taken into account when creating the inspection program, because once it is created the interaction with the user is lost and the program cannot modify any more.

5.9 Getting the information back into the CAD system

5.9.1 Introduction

The aim of this module is to read the measured values provided by the CMM back into the CAD system using the DMIS output file generated by the CMM. The first step is to load the IGES file and then read the CMM information back into the interface where the measured features are drawn in a different colour, depending on the tolerance attainment The measured actual values are also displayed (see figure 48).

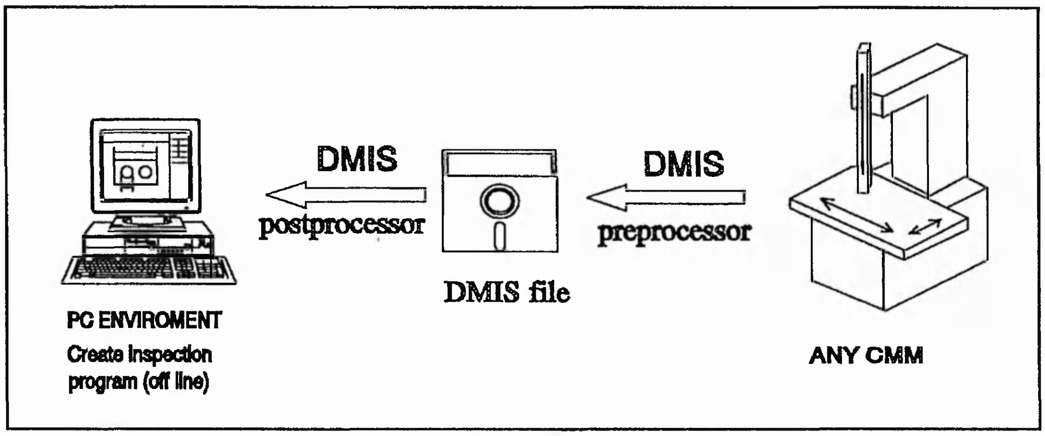


Figure 48 : Getting A DMIS file back into the system

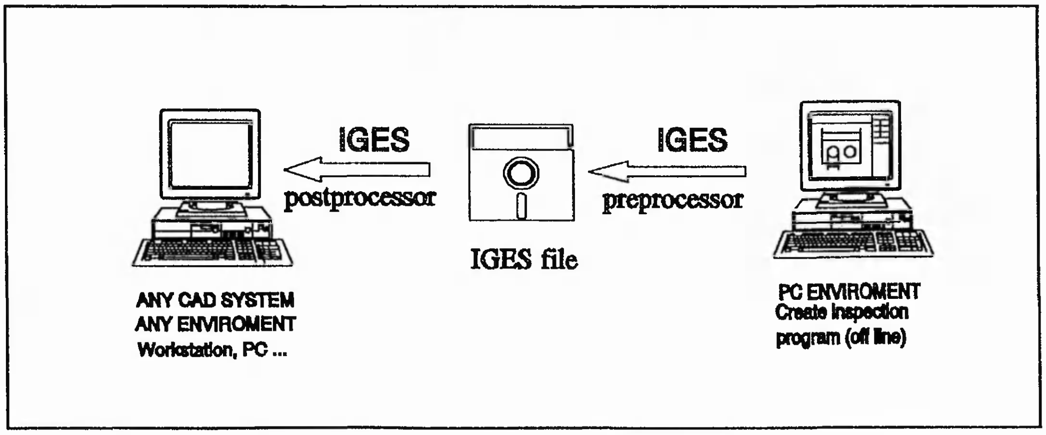


Figure 49 : Getting the information back into the CAD system

The user can then create an IGES output of the drawing displayed onto the screen with all the additional information and get the data obtained by the CMM back into the CAD system (see figure 49).

5.9.2 Reading DMIS output file

The features defined with the measured values and their respective tolerances in the DMIS output file are read back into the system. Having checked the tolerances specified for the features, the within tolerance values (arc, circle or line) are drawn in green, and the out of tolerance values are drawn in red. If no tolerance has been defined for this particular feature, then it is drawn in blue. Different colours are used to discern measured features from defined ones, as the screen resolution is not high enough to distinguish between them.

In the case of an arc or a circle, where two values are measured (i.e. diameter and centre coordinates), an additional cross is drawn in the centre of the arc or circle in the corresponding colour. This represents whether the centre coordinates meet the specified tolerances (see figure 50). The actual values are also displayed in different colours next to the entity which represents the measured feature. The reason for displaying the values is to give the user a graphical picture of the actual values rather than giving an output file full of numbers. Figure 50 shows how this information is displayed on the screen.

5.9.2.1 How to read the DMIS file

The first step is to set the coordinate system in the same position as when the measurements were taken. Once it has been defined, the DMIS output file can be loaded into the system to process the information. The analyzed information consists of the measured features and evaluated tolerances. The actual values are read from the feature definition (FA) and whether they are within tolerance or not is described in the tolerance definition (TA) (see Appendix 2).

The read values are linked in two different types of lists depending on whether the entity to represent the feature is an arc (partial or complete) or a line. In these linked lists it is also stored whether it is within tolerance or there is no tolerance defined for each single value. This information will be used later to decide what colour to use to display the feature on the screen. The procedure followed to read the DMIS file is as follows:

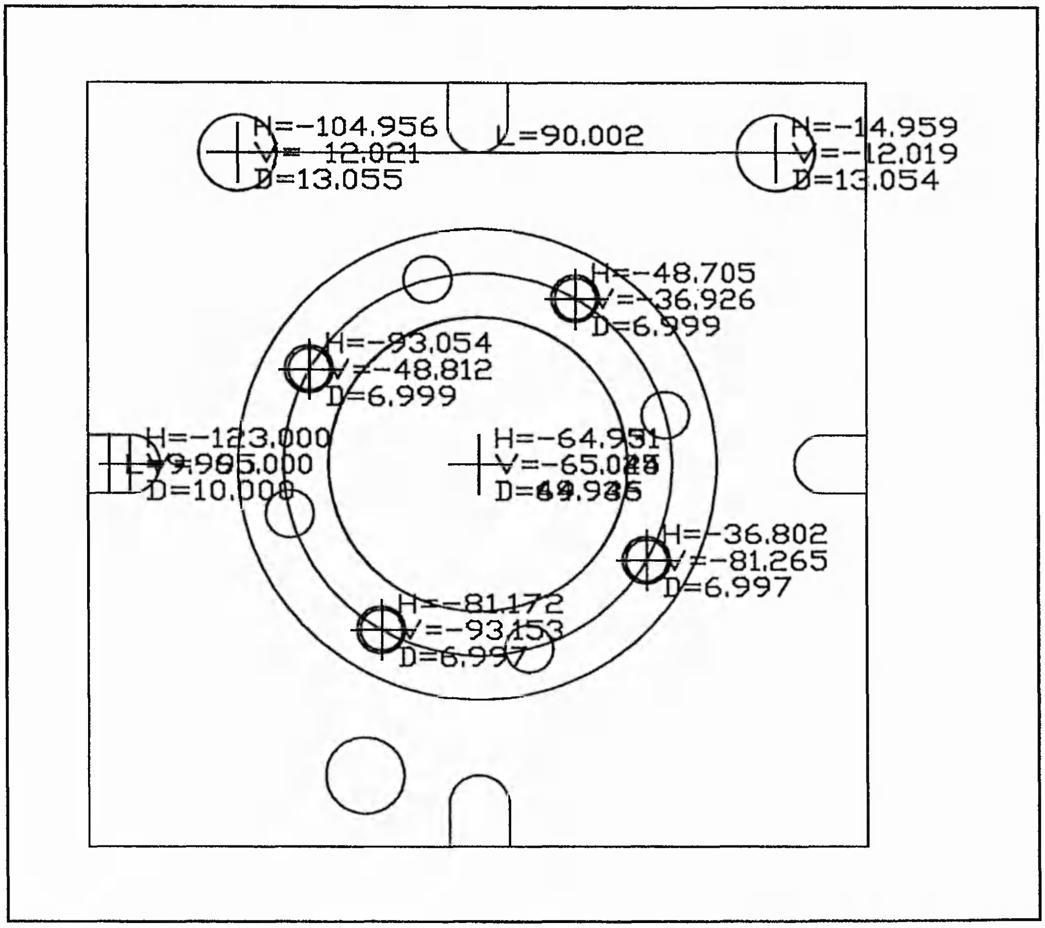


Figure 50 : Displaying a DMIS file on the screen

- Obtain a statements in single line.
- If it is a measured feature (FA):
    - If it is a line : Read the information for a line.
    - If it is an arc : Read the information for an arc.

- ◻ If it is a circle : Read the information for a circle.

▸ If it is an evaluated tolerance (TA) :

- ◻ If it is DIA : check whether is in or out tolerance and applied to the respective arc or circle.
- ◻ If it is DISTB : check whether it is in or out tolerance, read the actual value and applied it to the respective line.
- ◻ If it is CORTOL : check whether it is the vertical or horizontal axis and whether it is in or out tolerances and applied it to the respective arc or circle.

Once all the information has been read into memory, it is ready to display onto the screen with the respective colours. To calculate the screen coordinates from the actual coordinates the formula explained in section 4.7 is used.

5.9.3 Writing the IGES file

Once all the information has been read back into the system and displayed onto the screen, the user can create an IGES file to get the information back into the CAD system. A new IGES file which will have the geometry information and the measured values will be created. Therefore the same drawing shown on the system screen can be transferred into the CAD system. The only entities which will not be mapped into the IGES file will be the annotation entities which can be created again with the actual values once the information is back into the CAD system.

The user can identify at once which features meet the specified tolerances because they are displayed with different colours. He/She can then create the dimension entities for the important features and plot the new drawing. A graphical comparison can then be done (the measured features are displayed with their respective values in different colours) between the design values and the actual ones and whether they meet the specified tolerances.

5.9.3.1 How the IGES file is written

The drawing displayed on the screen must be converted into an IGES file to transfer it back into the CAD system. All the geometry entities which represent the drawing and all the entities which represent the measured features must be defined in the IGES format.

The first step is to write the start and global section which will be the same for all the IGES outputs. Thus this information store in a fixed initialisation file is written into the new IGES file.

Once the start and global section are written, the directory entry for all the entities must be defined. Most of the twenty fields which comprise the directory entry are the same for all the entities; only the following parameters will vary:

• **Entity type number :** Three different entities will be defined, line (110), arc (100) and text (212).

• **Parameter count number :** This parameter specifies how many lines are to be defined for a particular entity in the parameter data section. In case it is a line or an arc, only one line will be necessary to define all the parameters and if it is text two lines will be required.

• **Pointer to the parameter section :** This parameter defines in which line of the parameter data section is the information for this particular entity. This number will be calculated using a counter which will start with number one and will add one unit if the entity defined is a line or a circle and two units if it is text. The number of added units is the number of lines in the parameter section for that entity which is already known.

• **Colour :** The colour of each entity is defined with this parameter. All the entities have an assigned colour depending on whether they are part of the geometry (white) or they are within the tolerance limits (green) or out of tolerance (red), or if they are measured features but no tolerance has been defined for them (blue).

The procedure to write the directory entry is as follows:

- For all the geometry lines, add a directory entry (110,white)
- For all the geometry arcs, add a directory entry (100,white)
- For all the measured features represented with a line,
    - Add a directory entry for the line (110, colour);
    - Add a directory entry for the text (212, colour);
- For all the measured features represented with an arc, add a directory entry
- For all the measured features represented with an arc:
    - Add a line to represent the vertical coordinate (110, colour);
    - Add text to represent the measure value for this coordinate (212, colour);
    - Add a line to represent the horizontal coordinate (110, colour);
    - Add text to represent the measured value for this coordinate (212, colour);
    - Add a circle to represent the diameter (100, colour);
    - Add text to represent the measured value for the diameter (212, colour);

Once the directory entry has been transcribed, the parameter data section must be written into the file. The procedure followed is the same as the previous one, but instead of typing directory entry information the parameter data for each entity is written into the file.

It must be remembered that in all the entries of the parameter data section a pointer indicating the position of the directory entry information for that particular entity must be defined. This number is calculated by initializing a counter to one and adding a unit every time a new entity is defined (two in case the defined entity is a text).

The last section (i.e terminate section) is just a line defining how many lines occupy each section in the file including the terminate section. These numbers are calculated with pointers which are updated when each section is mapped into the file.

Chapter 6 : Results

6.1 Introduction

In this chapter the results obtained testing the different parts of the program will be discussed. Also discussion of the problems found at the different stages of development and the attempts made to solve them.

6.2 Testing IGES

Initially the created IGES post-processor was tested. After several trials and consequent corrections everything seemed to work properly apart from the texts. There was a problem to locate them with the appropriate x, y coordinates on the screen. That was because the program was firstly created under "Quick C" (Microsoft C), which did not support any function to write a text in x, y pixels on the screen. It only possible to locate the text in rows (30) and columns (80), which is less than the 640/480 pixels of the screen, so the text seemed to be a little bit misplaced. This problem was solved when the program was transferred to turbo C++ (Borland) which supports a function to place the text in any of the 640/480 pixels of the screen.

Most of the IGES files tested were created by AutoCad although some Unigraphics IGES files were also tried. The post-processor worked correctly for all those IGES files which were defined with the type of entities supported by the post-processor (see chapter 4). Some IGES files with different entities (that were not defined in this system, e.g. matrix, view ...) were also tested and the result was that all the unknown entities were disregarded and the known ones read. In the Appendix 4 there is an example of the IGES file of the drawing of the top face of the "capability block" (a standard test component) created by AutoCad and the drawing itself. In this particular case 100% of the information shown was transferred into the system. In other words, after checking AutoCad screen and the system screen, one could decide the same drawing was displayed. This drawing was defined in the IGES file with the type of entities defined in chapter 4 and others but the latter ones were not related to the geometry of the drawing and even when they were disregarded the output was not altered.

	Drawings tested with errors	Drawings tested with no errors	% of success of information taken	% of success of drawing display
AutoCad	30	25	100%	95%
Unigraphics	10	15	100%	95%

Table 1 : Testing IGES files

Around 25 attempts to read different drawings created by AutoCad and around 10 created by Unigraphics were necessary to correct all the programming errors. Once, the post-processor was working properly around 20 different AutoCad drawings were tested to check the reliability of the program. The post-processor worked properly for those entities that are defined in the system disregarded the unknown entities. Most of the drawings contain unknown entities like the transformation matrix but despite not considering them, the final output was successful, i.e the internal information read for each entity was necessary for this application and the drawing looked the same than in the CAD system where they were created. Further 15 drawings created by the Unigraphics pre-processor were tested obtaining 100% success. The table 1 shows the results for the IGES post-processor.

6.3 Creating the environment

The next step was the creation of the environment. Initially it was created under

"Quick C" (microsoft C) and although the distribution of the menus was similar to the final one, the highlighting did not occur when the mouse was passing over them. When the program was transferred to turbo C++, the advantage offered by the C++ functions were used to handle all the menu system. All the highlighting of the menus could be then created. Besides these were created in such a way that if any changes or addition of menus could be done in an easy manner.

It must be taken into account that the "interface" created is not a CAD system. Thus, typical CAD system functions like transformation, scaling, mirroring, deleting or adding entities are not presented. All these activities should be carried out in the CAD system before creating the IGES output.

6.4 The inspection model

To create the inspection program itself, firstly the features the system was going to offer had to be decided. The CMES reference manual was taken as a guide of the features usually CMMs provide and the features and tolerances there described were supported. The inspection model could be created selecting different geometric entities from the screen depending on the case and stored in memory to eventually create the inspection program.

It was convenient to have the information of all the different features created in a single linked list as the measurement order (if the selection order is taken into account) was already established. The problem was that the information stored for each feature was different therefore the registers defined were of different types and they could not be linked in a single list. To solve this problem a union type of entity was created, which would point to a different register depending on the type of feature created. An identifier to discern the different features was also defined with the union type of entity. When the option to minimize the time of measuring was provided, the measurement order had to altered. The algorithms created would reorganize this list of features with this criteria (shortest path). The final outputs in simulation, CMES program and DMIS program, did not change because the data organization was the same in both cases.

Every time an IGES file of the same part was loaded into the program a new list of features was created without deleting the old one. One could think that memory shortage could be a problem. But the information stored for each feature (floats) is small and in all the tests done (some of them five different faces were loaded) there was still enough memory available. It has to be noted that not more than 5 faces can be measured in a part with the same inspection program with this interface. Besides all the information which is not needed is deleted. So when a new face is loaded all the geometry information of the previous face is deleted. If the new face belongs to a new a part all the information related to the inspection model of the previous part would also be deleted, whereas if it belongs to the same part this information would be maintained to eventually create the inspection program.

One of the restrictions of the programs is that only one coordinate system, one probe and one plane can be created for each list of features or in other words for each face, auxiliary axes to measure features in inclined planes cannot be created. This is because each plane is assigned with one probe and all the features which lie on that plane can only be measured with that particular probe.

6.5 Testing the CMES inspection programs

The first of the inspection programs to be tested were the CMES programs because the DMIS translator was not still available. The first programs were checked in the old CMM with the fixed(star) probe mount. These probes were taken as a reference to define the probes in the system. To measure a part the set-up of the machine had to be done manually, although a program was created to do the set-up of the capability block automatically to make the task of testing easier. After several corrections in the program the whole capability block with different tolerances was measured successfully.

Different special measurements were made to check the crash detection algorithm. For instance, measuring features with intercepting features inside it (e.g, a hole with a cylinder higher than the hole inside it) or whenever the probe moves along the part to measure the next feature. This latter was tested every time the probe has to come out of (inside) a feature (hole, slot..) to measure the next one, in order to avoid crashing with the feature walls. This was also tested whenever a solid feature was measured (cylinder, block ...) because to move the probe from one point to another the probe has to go above the feature and move to the next point. This movement is also controlled by the crash detection algorithm. In summary, it can be said that all the parallel movements realized to measure the different features are controlled by this algorithm and it worked properly in all the tests realized (see section 6.2).

Further improvements could be done in this area of testing possible collisions not only with the probe but with the rest of the mechanism (mount, arm ...).

The CMES inspection programs were the first ones to be tested. Initially several simple programs were tested to check programming errors and around 30 complete programs with different features were tested successfully. Soon it was realized that a simulation of the machine movements to check and correct all the small modifications that were made during the development of the program was necessary. The simulation was made using the same algorithms created for the generation of CMES inspection program, so that if the simulation worked, the inspection program worked as well (see table 2, pag 131).

6.6 Testing the simulation

Once the inspection program was created a simulation of the movements the probe does to measure the different features was created. The same steps as in the previous case but changing the output commands were followed to create the screen simulation. In this case instead of giving CMES output commands to move the probe some functions were created to move a disc (which simulates the probe) over the part. The probe changes the colour when a perpendicular movement to the work-plane takes place. Also some changes in the coordinates had to be made because the coordinates of the screen are always x and y positive whereas the coordinates of the part can be positive, negative or a combination of both depending on the current work-plane. The inspection program could be then checked first on the screen and later on the CMM several times verifying that the simulation represented correctly the movements of the CMM.

Around 30 complete inspection programs with different combinations of features were tested to make sure that the machine movements correspond to the simulation. The simulation was very useful to check the crash detection and the "fast way" algorithm which were only tried in the machine once they were working in the simulation. These worked almost straight away without much modifications. These algorithms were tested hundreds of times in the simulation.

6.7 Testing DMIS inspection programs

After the simulation and the CMES output were debugged, the DMIS output was generated. It could not be tried though until the new CMM with the DMIS translator arrived. The steps followed were the same as in the CMES output, changing the commands and the manner of referring to the coordinates. The three coordinates values (x, y, z) must always be given. When the new CMM arrived the same programs which had been tested in CMES were executed, even though the outputs in CMES and DMIS were slightly different. In CMES there are commands to measure exactly all the defined features whereas in DMIS to get certain values (e.g, length of a slot or a block, radial distance ...) a tolerance had to be always defined. Otherwise not only the coordinates of the points taken would be given. For example, to measure the radial distance in CMES there is a special command "WT" which gives that measurement, whereas in DMIS to get that value apart from the commands to take both points a tolerance for that length and the length itself must be defined within the same command. This is because these measurements are controlled by the tolerancing commands instead by the feature definitions. Another problem was that the DMIS translator in the LK machine did not support the command "EVAL" which was the one used in the DMIS programs to relate features and tolerances so this relation had to be done using the "OUTPUT" command to be able to test the programs. The algorithm to detect crash collisions worked exactly the same as it did for the CMES programs (the movements are determined by the same algorithm in both languages).

Another deficiency of the DMIS post-processor was that it did not accept "IF" statements which are defined in the DMIS specification. The problem was that the prototype programs could not be tested because the programs could not be compiled.

The DMIS output files by default are written to a file. This file contains all the measured values in DMIS output format.

Several small programs with few features were tested to check the errors in the DMIS statements. The coordinate values were correct because the algorithms to calculate them were the same as for the simulation, and for the generation of CMES programs which had been tested several times before. In spite of that around 30 complete DMIS inspection programs were created to verify the relationships between the simulation and the DMIS programs (see table 2).

	Simple programs/with no success	Simple programs /with success	Complete programs /with no success	Complete programs /with success
CMES files	30	20	5	25
Simulation	100	400	50	300
DMIS files	30	20	5	25

Table 2 : Testing CMES, DMIS and the simulation

6.8 Inspection program restrictions

The inspection program was generated to take the minimum number of contact-points the CMM needs to calculate the actual feature values. But for some features (e.g, hole) more contact-points could be taken. The advantage of having more contact points is that the precision of the measured values is bigger, on the contrary it takes longer to complete the measurement. Furthermore, to calculate the measurement sequence would be more complicated because there are more points to consider.

One of the constraints is where the contact-points are taken in the feature. For the same type of feature the contact-point position is defined and cannot be altered (see section 6.5), but the CMM allows to take a point in any part of the feature. If this is achieved it could be applied to find the possible shortest-path. However a new problem would arise because the number of combinations would increase drastically as the number of possible points to consider would be enormous. In the solution proposed by the system as the contact-points are fixed, the number of combinations decreases, even then some assumptions have to be done to calculate the path, otherwise it would be impossible to find a solution (travelling salesman problem) (see section 5.7.2).

One of the problems which can happen testing the inspection programs is that the contact points are taken when the probe is still in travelling mode, which will give an error. The problem is that the distance from the contact-point at which the probe changes the speed is very small (3mm) (although it can be changed within the program) to avoid wasting travelling time at slow speed. But if the real part is too wrong (i.e. the real dimensions are very different from the geometry ones) the contact-point would be taken at high speed and an error would be displayed stopping the measuring process. The solution then is changing the "approaching distance" in the program slowing down all the measuring process (see section 6.1). However this error is very rare because usually parts are not that far from the geometry (not more that 1 mm)

6.9 The PH9 and the new CMM

Another problem with the new CMM was that the PH9 could not position the probe in one of the five positions supported within the system (particularly probe 3, see chapter 5). This meant that not more than four faces (the bottom one is where the part lies) could be measured in one program. This problem can always be solved using a star probe centre instead of the PH9.

Several programs to measure different features of the capability block were created and tested in both languages (CMES and DMIS) in the new machine. For the same inspection model both programs drive the CMM in the same manner, and the measurement outputs are the same.

The problematic points are two. One is changing measuring face which involves a rotation of the PH9 (to change the probe) and a movement to the next face. The other one is to change the coordinate system which means changing all the coordinates and refer them to a new system, were tried several times and the programs performed successfully in both CMES and DMIS.

6.10 Testing the link CMM-CAD

The next step was to return the measured values into the system. To do that the DMIS output values which had been stored into a file are read into the system. A post-processor to read this values was created. This post-processor was tested with most of the DMIS output files created by the inspection programs which had been previously created within the system. Once the information was back into the system an IGES output could generated by an IGES pre-processor which is provided within the system. This pre-processor maps the geometry of the part, the measured values and the text to represent this values into an IGES file.

The IGES files created for the DMIS output files read into the system were transferred to AutoCad without any problems. After reading the information into AutoCad it was necessary to centre and scale the drawing on the screen because usually the coordinates of the drawing were not within the default limits of AutoCad. Dimensions were created for the measured features and whether they were the same as the values given by the CMM was checked, so that the link with the CAD system was correctly done.

Several simple inspection program,s outputs were tried to get back into the system until all the errors were corrected. Around 20 full inspection program's output were read into the system and transferred to AutoCad via IGES successfully.

6.11 Testing summary

The program has been tested a number of times during its evolution. Whenever a new option was created it was tested until it worked properly. The figures shown in table 1 and 2 are just indicative to give the reader and idea of how reliable the program is.

The first module to be tested was the IGES translator, using several drawings from

AutoCad and Unigraphics. The information transferred was completely successful, i.e. the required data were always read. The drawing display was transferred almost completely; sometimes slight variations in the annotations could be seen. Once, the IGES translator was working module to create the inspection model was made and tested. To check whether the values stored for each feature were correct, they were printed on the screen because there was not yet any output program to check them. Afterwards the CMES inspection programs and the simulation were created and tested during their evolution in the CMM and in the screen. The DMIS programs were much easier to test because they rely on the algorithms created for the simulation and the CMES files, which had been tested several times before (as it has been discussed earlier). the DMIS output was created. Finally, the feedback into the CAD system was created using DMIS and IGES. To check, it most of the DMIS output files created when the DMIS inspection programs were tested were read and transferred back into the CAD system (AutoCad) via IGES.

6.12 Error handling

In the move towards full automation, error detection, error diagnosis, and error recovery are extremely essential. In this proposed system some form of error handling has been attempted in a very simple manner. These are briefly discussed below.

In order to avoid wasting time checking whether the programs work properly in the CMM, or causing any damage to the machine, some facilities to detect error are necessary. Two approaches have been provided in this research to check possible errors before they actually occur.

One of them is the simulation, which represents graphically on the screen the movements the probe is to perform in the CMM. This allows the user to correct possible errors before the inspection program is created. This means that the errors can still be corrected before the measurement takes place in the CMM (check error off-line) (see section 5.8).

The other approach is to create a "prototype" program (see section 6.8). This allows the user to check in the machine, if an error could happen. In this case, before measuring a feature, the program asks the operator whether to measure that feature or not, and states the type of measurement to take place. If the user wants to omit that measurement, the program will still move the probe to the position where the probe would have finished, if the measurement had taken place. Any possible probe crashing is also checked in this movement. The reason for this movement was to keep the algorithms simple and robust.

In general to program off-line, and to institute on-line error detection is complex. This is because when the program is being generated, it does not have an interaction with the user and it cannot act accordingly to his/her answers. For instance, in the prototype program the user could be asked instead of whether or not to measure the next feature - to measure a hole or a boss, and do the respective measurement. The problem is that the final probe position for measuring the hole or the boss is different, and at the time the program is created it is not possible to know the user's answer, therefore automation functions like the crash detection or the generation of the inspection sequence would not be viable because of the lack of information. Thus the reason why in the prototype program the solution followed was always to finish in the last expected position in a measurement sequence even when the measurement does not take place.

In the LK CMM there is a way to create an inspection program with error detection because there is a command which detects any possible error. It will detect, if a point has not being taken when it was expected and if something touches the probe when it is realizing a travelling movement, and execute the corresponding actions. The problem with an inspection program created off-line, is that if an error of this kind occurs, even when the execution of the program could go on it would not be any longer reliable, because the movements of the probe are not any longer the ones expected.

Finally it could be said that error detection is important and necessary, but to create off-line, an on-line error detection is very complicated. The program cannot control all the different situations that can occur in the CMM because there is not enough information. In order to achieve this additional sensor information (e.g, vision) may be needed.

Chapter 7 : Conclusions and Discussions

7.1 State of the art

Linking design and inspection is becoming increasingly popular as the use of CMMs within industry grows. However, the state of CMM programming is about where CNC programming was a decade ago.

A considerable number of CMMs are still programmed on-line in self-teaching mode, therefore the machine is constantly utilized creating part-programs instead of executing the part program itself. One must take into account that CMMs are expensive machines for their time to be wasted creating part programs which could be generated elsewhere. Another inconvenience is that the quality inspector needs to have knowledge about programming, which is not generally the case. Thus, there is an obvious need to create off-line programming software to free CMMs of any task which does not involve measuring, and also to provide an inexpensive environment (i.e, PCs, Workstations ...) within which to create the part program easily.

Another deficiency found in the use of CMMs was the lack of feedback of measured data to improve the manufacturing process. Initially machining tools performed individual operations and the machine operator carried out the measuring operation and made adjustments accordingly. When machine centres were made with more degrees of freedom and automated tool changers were incorporated, the measuring task was becoming more complicated and CMMs with faster output were introduced into the market. However a problem arose in that CMMs needed to be situated in a clean environment to work properly and therefore they could not operate on the shopfloor near the machine, so feedback between the two was reduced. A network system could not be created yet because CNCs were not prepared to understand the CMMs outputs.

Major CAD vendors are now working on providing a two way link between CAD and CMM. Based on a 3D CAD model, and interacting with the user, they create a CMM part program which is either in a specific CMM language or in the standard DMIS. Most of them also provide the option of getting a DMIS output file with the actual values back into the system. Once the information is transferred back into the CAD system, features within and out of tolerance can be displayed in different colours and dimensions with the real values can be created. This drawing with the measured values can be plotted to give the operator a graphical impression of the state of the part. The major disadvantage is that the products CAD vendors develop are sometimes restricted to a specific CAD software and usually run on workstations, and not on personal computers. However recent trends seems to indicate a move towards powerful PC systems.

It is possible to define many engineering components using a 2½D model. The shape of these components can generally be represented by profiles and associated heights and they are relatively easy to design and manufacture. They are usually designed on small CAD systems which run on PCs and which do not provide any inspection link.

Companies which use this type of CAD system to design their products have two options. One is to buy a bigger CAD system which provides a CMM programming module. The problem with this, however is that usually this type of CAD system and associated hardware can be very expensive, resulting a large investment for the company. The other option is to either create the part programs manually or to use the CMM in self-teaching mode with all the involved disadvantages as explained previously.

7.2 Work done

The work undertaken during this research was to provide both missing links between CAD and CMM by developing off-line programming software for this type of prismatic component, which is capable of running on a PC. The idea was to create a program independent of any particular CAD system or CMM, and thus widening the range of application. This was achieved by using standard interfaces to allow communication with both the CAD system and the CMM.

1. The standards: The standard selected to get information from and to the CAD system was IGES because it is the most widely used graphic standard and consequently most of the CAD systems support it. On the other hand the standard DMIS, which is the most important standard in inspection, was used to create the CMM program and to transfer the actual values back into the system.

2. Linking CAD with CMM: The software developed provides the user with an easy way of creating the part program. The need for an expert programmer is therefore not necessary and it is possible for a person with just inspection knowledge to decide and select the features to measure. The program calculates any possible collision between probe and workpiece automatically solving the problem if required, and it also provides the option of generating an automatic inspection sequence. Another useful option is the simulation which shows on screen the movements the probe will follow on the CMM and the user can take decisions accordingly. The output program can be written in DMIS or in CMES which is a specific application language for LK machines.

3. Linking CMM with CAD: Once the part has been measured in the CMM, the DMIS output file with the real values generated by the CMM can be read back into the system and the features with the real data will be displayed in different colours depending on whether they are within or out of tolerance. This gives the inspector a graphical impression which is always much easier to read than a file full of numbers. The user also has the option of retrieving this drawing into the CAD system via IGES in case the drawing requires manipulation or plotting.

This research can be considered as further advancing the work carried out to link

CAD and inspection (CAD/CAI). However further work is still required to reach the level currently attained between CAD and CAM.

7.3 Discussions

One of the problems of using IGES for inspection planning is that the tolerance information cannot be processed (see section 4.8), therefore human intervention is needed to decide what features to measure. This lack of information makes it necessary the creation of an environment to interact with the user to create the inspection model, which was the solution proposed in this research. Besides the information taken is not feature-based so human intervention is also necessary to create the measurement features with their respective tolerances out of the geometry.

With the information taken from the IGES file and the human interaction an inspection model can be created. This inspection model is stored in memory and it is expanded every time a new feature is created. When a new IGES file (e.g, a new face) is loaded only the necessary information to eventually create the inspection program is stored, the rest (geometry information) is deleted and the memory released. This is done like that to avoid memory problems (see chapter 7).

However, there is a restriction, features located in inclined planes cannot be measured. In this application only one coordinate system and one probe can be applied to each plane because there is not enough information about the part. The problem is that features located in inclined planes cannot be measured that way because auxiliary axes for the inclined plane must be created. This problem could be solved using a 3D model.

The next step is to create the inspection programs, which can be created in DMIS and CMES language. Both of them are created following the same steps and their performance in CMM is exactly the same. The simulation was also generated following the same steps than the DMIS and CMES programs so it represents exactly the same movements the CMM will do which was very useful to debug the program.

Finally, the link to connect CMM with CAD was created. The information was successfully transferred into the CAD system and further manipulation of the data could then be done (plotting, dimensions ...). See chapter 7 for more detailed information.

If more information about the shape of the part and about the tolerance information could be processed the degree of automatization would increase. The ultimate idea would be to create the inspection program automatically, without any human intervention.

Chapter 8 : Further work

8.1 3D representation

In this work, all the information taken from the IGES file by the post-processor is defined as 2½D information, i.e two and a half dimension (see section 3.1.5). However, 3D information can also be generated from the IGES specification. There are two ways of defining 3D information; surface definition and solid modelling, the latter is more suitable to describe prismatic parts.

The description of both B-Rep (Boundary REPresentation) and CSG (Constructive Solid Geometry) is the primary reason for the existence of version 5.1 of IGES. However, it is defined in the grey pages of the specification (in Appendix G), indicating that the technology has not been tested by actual implementation. Because of this, software vendors are reluctant to incorporate B-Rep and CSG support into their products until these entities are defined in the main body of the specification. Another version of IGES, V6, is planned for release in 1995 and one of the aims is to incorporate B-Rep and CSG in the main body of the IGES specification. Although B-Rep and CSG entities are only described in the Appendix of the IGES specification, it does give an idea of how IGES is developing [Puttre 93].

Most CAD systems use Boundary representation (B-rep) to describe solid models but the solid modelling representation (CSG) is more appropriate to describe prismatic parts. The problem is that CAD systems mainly use the B-Rep representation to define the solid model, therefore most of the 3D IGES output will use these kind of entities. However CSG representation is becoming increasingly popular (especially among researchers) for the representation of solids. Both representations have advantages and disadvantages, so a decision must be made on which one to use, or to use them both.

Another advantage of having more information about the shape of the part is that the machine set-up could be done semi-automatically, but it would still be necessary to know the position of the part on the table. This problem could be solved, for example using a vision system attached to the CMM to provide information on the location of the part on the table.

Moreover with this information, more collision apart from the probe against the part could be detected. For instance, the whole probe system and the arm of the machine could be considered and the movements they realize could be controlled to avoid any crash.

Furthermore, auxiliary axis to measure special features could be created. For instance to measure features which are not located in any of the five planes of the part without the need of re-locating the part and avoiding the creation of a new set-up.

8.2 Tolerance representation

Once it has been decided what representation use, the next step is to read the information from the IGES file. The advantage of both these representations against the 2½D model is that much more information of the geometry of the part can be taken from the IGES file but the problem of representing the tolerance information still exists and this is essential to create the inspection plan. Unless this information can be taken from the part model human intervention will be still required to create the inspection plan. Without the tolerance information no rule can be created to generate the inspection program automatically.

There is some relevant work trying to relate geometry to tolerance information such as GD&T (geometric dimension and tolerancing), which is an ANSI standard. If it is possible to obtain the geometry with its respective tolerance values, rules could be formulated and the inspection program created automatically.

8.3 The STEP standard

Ultimately, the STEP standard could be used. The idea of this standard is to define all the information relating to a product throughout its entire life cycle, so all the data necessary for the creation of the inspection program will be available. However the standard is still under development and CAD systems do not currently support it. Thus, STEP is still in its early stages of development but is destined to be the standard of the future.

8.4 Feature-based CAD systems

The tendency of the CAD systems is towards feature-based systems. The idea is to create the designs based not only in the geometry but also in the functionality of the features. This will provide to the subsequent processes (manufacturing, inspection ...) with more relevant information about the part.

The ideal standard output for this type of CAD systems is the STEP specification. As have been mentioned earlier this option is not commercially available and even when the tendency is towards a feature-based system, CAD systems do not support it yet.

Finally, it could be said that the more information one can take from the design and manufacturing process about the part the less human intervention that would be needed to create the inspection program.

References

[Anon 90] Anonimus, **DMIS approved as American National Standard**, Tooling & Production, June 1990, pp 24.

[Anon 91] Anonimus, **Coordinate Measuring machines**, Engineering, May 1991, pp 27-30.

[Atkey 86] M.Atkey, **Time to Measure Quality Costs**, Quality Control, January 1986, pp 85-93.

[Bloor 91] M.S.Bloor and J.Owen, **CAD/CAM Product-Data Exchange: the Next Step**, Computer aided design, Vol. 23, Nº 4, May 1991, pp 237-243.

[Bosch 92] J.A.Bosch, **The Changing Roles of Coordinate Measuring Machines**, Industrial Engineering, November 1992, pp 46-48.

[BS6808 87] **Coordinate Measuring Machines,** British Standard Institution, 1987.

[Corrigal 92] M.J.Corrigal, M.K.Lee, and R.I.M.Young, **Manufacturing Code Generation in a Product Model Enviroment**, Proc. Ins. Mech. Engineers, Vol 206, 1992, pp 165-175.

[Cox 92] M.Cox, **Assesing CMM Software**, Quality Today, January 1992, pp s1-s2.

[Dori 92] D.Dori, **Dimensioning Analysis**, Communications of the

ACM, Vol. 35, Nº10, October 1992, pp 92-103.

[Daniels 92] G.E.Daniels, **DMIS: A Quality Standard**, Quality Today, 1992, pp s12-s13.

[ElMar 87] H.A.ElMaraghy and P.H.Gu, **Expert System for Inspection Planning**, Annals of the CIRP, Vol. 36/1/1987, pp 85-89.

[Ercole 91] M.Ercole, **Command Performance**, Manufacturing Engineer, September 1991, pp 22-23.

[Evers 91] W.Eversheim, G.Marczinski, and R.Cremer, **Structured Modelling of Manufacturing Process as NC-Data Preparation**, Annals of the CIRP, Vol. 40/1/1991, 1991, pp 429-432.

[Genest 90] D.Genest, **Ford-Using CMMs for all they are Worth**, Quality Today, October 1990, pp 17-18.

[Hahn 88] D.Hahn, J.Roder, W.Michalowsky, H.Purucker, J.Hornung, **Linking Design and Test**, Computer Aided Design report, July 1988, pp 1-9.

[Hopp 85] T.H.Hopp and K.C.Lau, **A Hierarchical Model-Based Control System for Inspection**, Automated Manufacturing, ASTM STP 862, 1985, pp 169-187.

[IGES 91] **Initial Graphics Exchange Specification V5.1**, IGES/PDES Organization, September 1991.

[King 92] B.J.King anf P.W.Norman, **A Step in the Right Direction**, Professional Engineering, November 1992, pp 12-14.

[Kwok 91] W.L.Kwok and P.J.Eagle, **Reverse Engineering: Extracting CAD Data from Existing Parts**, Mechanical Engineering, March 1991, pp 52-55.

[Mason 92] F.Mason, **Program your CMM Off-Line**, American machinist, October 1992, pp 45-47.

[Mayer 87] R.J.Mayer, **IGES: One Answer to the Problems of CAD**, Computer Aided Design, June 1987, pp 209-214.

[Merat 92] F.L.Merat and G.M.Radack, **Automatic Inspection Planning within a Feature-Based CAD System**, Robotics & Computer Integrated Manufacturing, Vol 9, Nº1, 1992, pp 61-69.

[Nnaji 91] B.O.Nnaji, T.S.Kang, S.Yeh, J.P.Chen, **Feature reasoning for metal components**, Int.J. Prod.Res., Vol. 29, Nº 9, 1991, pp 1867-1896.

[Peggs 91] G.Peggs, **Developing Standards for CMMs**, International Conference of CMMs (Birmingham), 1991, pp s1-s3.

[Puttre 93] M.Puttre, **New Version of IGES Supports B-Rep Solids**, Mechanical Engineering, January 1993, pp 50-52.

[Quinlan 88] J.C.Quinlan, **How to select a CMM**, Tooling & Production, June 1988, pp 40-46.

[Schaffer 85] G.H.Schaffer, **Integrated QA:Closing the CIM Loop**, American Machinist, Vol 775, 1985, pp 138-158.

[Schaffer 86] G.H.Schaffer, **QA plugs into CAD/CAM,** American Machinist, April 1986, pp s1-s3.

[Smith 90] P.Smith, **CAD/CAM Data Exchange**, CIM review, Winter 1990, pp 30-33.

[Sostar 88] A.Sostar, **Coordinate Measuring Techniques in Quality Assurance**, Robotics & Computer Integrated Manufacturing, Vol. 4, Nº1/2, 1988, pp 259-265.

[Sprow 90] E.Sprow, **Challenges to CMM Precission**, Tooling and Production, November 1990, pp 54-61.

[Tackes 90] D.Tackes, **Tools to Simplify CMM programming**, Tooling & Production, May 1990, pp 81-86.

[Tao 92] L.G.Tao and B.J.Davies, **Knowledge-Based 2½D Prismatic Component Inspection Planning**, Int.J Adv. Manuf. Tech., Vol 7, 1992, pp 339-347.

[Uptal 88] R.Uptal and C.R.Liu, **Feature-Based Representational Scheme of a Solid Modeler for Providing Dimensioning and Tolerancing Information**, Robotics & Compuetr-Integrated Manufacturing, Vol. 4, Nº 3/4, 1988, pp 335-345.

[Vosn 89] G.C.Vosniakos and B.J.Davies, **An IGES post-processor for Interfacing CAD and CAPP of 2½D Prismatic Parts,** Int.J. of Adv.Manuf.Tech., Vol. 5, 1990, pp 135-164.

[Vosn 90]	G.C.Vosniakos and B.J.Davies, **Knowledge-Based Automatic Allocation of CAD Drawing Annotation to Wireframe IGES Models of 2½D Prismatic Parts**, Int.J of Adv.Manuf.Tech., Vol 5, 1990, pp 224-239.
[Walker 92]	I.Walker and A.F.Wallis, **Applications of 3-D Solid Modelling to Coordinate Measuring Inspection**, Int.J. Mach. Tools Manufact., Vol.32, № 1/2, 1992, pp 195-201.
[Wright 86]	D.A.Wright, **CMMs Provide Factory Feedback**, CME, October 1986, pp 26-28.

Appendix 1 : IGES data format

1.1 General

An ASCII[ANSI68, ANSI77] form is defined in the IGES specification to represent data.

1.1.1 Defaults

A specific interpretation of an omitted item has been provided in some cases. Distinctions may be needed among empty fields, blank fields and zero fields.

- **Empty field:** Two consecutive field delimiters or a field delimiter and a record delimiter is an empty field. They are only possible in free formatted data.
- **Blank field:** It is a field containing only blanks.
- **Zero field:** It is a numeric field with only one digit, where that digit is zero.

1.2 ASCII fixed form

The ASCII fixed form has a fixed line length of 80 characters. The term column refers to the character position in each line. The file is divided into sections. The section identification character shall occupy Column 73 of each line. Every line in the file must have a sequence number, i.e., completely blank lines are not permitted. The remaining columns are assigned to fields as defined in the file section description. The term "record" refers to the set of parameters for one entity within one file section. A record consist of one or more lines.

1.2.1 Sequence number

A sequence number is a string of from one to seven digits and is the means of indexing lines within the various sections of the data file. The sequence number for each section begins with 1 (00000001) and continue sequentially without interruption to the value corresponding to the number of lines in the section. A sequence number may have either leading zeros or leading blanks and it is right justified in the line (Columns 74-80).

The sequence number is preceded in the line by a single letter code in Column 73 identifying the section in which the line resides:

Section	Code
Start	S
Global	G
Directory Entry	D
Parameter Data	P
Terminate	T

1.2.2 Constants

The specification defines six types of constants: integer(or fixed point), real (or floating point), string, pointer, language statement, and logical. The rules applied to them are the following ones:

• Blanks are only significant in string and language statement constants. A numeric field of all blanks is considered to denote the default value for that default value has been defined in the specification. No blanks are allowed between the beginning and end of a numeric constant.

• Numeric constants shall not contain embedded commas.

• The absolute magnitude of an integer constant may not exceed the value 2*(N-1), where N is the number of bits used to represent the integer value (Global Parameter 7).
Similarly, the absolute magnitude and precision of a real constant may not exceed that indicated by a Global Parameters 8-9-10-11.

• Only string and language statement constants may cross filed/line boundaries. When such a constant does cross a boundary, it is considered to extend to the last usable column in the current line and then to continue in with the first column of the succeeding line. The last usable column on lines in the Parameter Data Section is column 64; on lines in all other sections is column 73. A string constant may not be broken before the Hollerit delimiter (H).

• A numeric constant may be either signed or unsigned. If signed, the leading plus or minus determines the sense of the constant. If unsigned, the sense is assumed to be non-negative.

1.2.2.1 Integer Constants

An integer constant is always an exact representation of an integer value. It may assume a positive, negative, or zero value.

The form of an integer constant is an optional sign followed by a non-empty string of digits. The digit string is interpreted as a decimal number. The following are examples of valid integer constants:

0	150	2147456	+3245
1	-10	-2147456	

1.2.2.2 Real Constants

A real constant is a processor approximation of the value of a real value. The following rules apply to real constants :

• It may be a basic real constant, a basic real constant followed by an exponent, or an integer constant followed by an exponent.

• The form of a real constant is in order, an optional sign, an integer part, and a fractional part. Both the integer part and the fractional part are strings of digits; either of these parts may be omitted but not both. A basic real constant is interpreted as a decimal number.

• The form of a real exponent is the letter E followed by an optionally signed integer constant. A real exponent denotes a decimal power of ten by which the preceding constant is multiplied.

The following are examples of valid real constants :

256.091	0.	-0.58	+4.21
1.36E1	-1.26-02	0.1E-3	1.E+4

1.2.2.3 String constants

String constants are represented in the Hollerit form. A string constant is an arbitrary sequence of ASCII characters. Blanks, parameter delimiters, and record delimiters are treated simply as characters within the string. There is no limit on the length of a string constant.

The form of a string constant is non zero, unsigned integer constant (character count), followed by the letter H, followed by a string of characters consisting of the number of contiguous characters specified by the character count. The following examples are valid string constants:

3H123	10HABC.,.;ABCD
8H0.475E03	12H HELLO THERE

1.2.2.4 Pointer constants

A pointer constant is represented by a string of zero to seven characters. An empty field, a blank filed, and a zero value are all equivalent. However, such null pointers are valid only where the meaning of the null value for that pointer has been specifically provided. Furthermore, a negative integer in the field is valid only where the interpretation of a negative fields has been explained.

Pointer constants are used to identify a line in either the same or a different section of the data file. The magnitude of the pointer constant corresponds to the sequence number of the referenced line, and the referenced file section is determined by the context of the reference. Pointer constants are unsigned except where they are alternative parameters in a field. Pointer constants whose magnitude requires fewer than seven digits may use leading zeros or leading blanks in fixed format files.

1.2.3 Rules for forming and interpreting free formatted data

The data in several sections of the file may be entered in free format. The free format will apply to the range of columns of successive lines as needed. This free format feature allows the specification of parameters in the prescribed order without restricting the placement of the parameter to a particular location on a line. When free format is permitted, the following rules apply:

• The parameter delimiter (Global parameter 1-defaulting to a comma) is used to separate parameters.

• The record delimiter (Global parameter 2-defaulting-to a semicolon) is used to terminate the record (i.e., to terminate a list of parameters).

• When two parameter delimiters, or a parameter and record delimiter, appear adjacent to each other, or are separated by only blanks, the delimited parameter is considered not to have been specified in the file and should be given its default value. Unless specifically noted, the default value for a numeric parameter is zero, and the default value for a string parameter is null. Pointer constants can be defaulted only when a specific definition of the meaning of the default field has been provided in the Specification. It is the responsibility of the processor to ensure that these default values are reasonable for the particular parameter in question.

• When a record delimiter appears before the list of parameters is complete, all remaining parameters should be given their default values. In the case of early termination of the Parameter Data record, either or both groups of the additional parameters need to be present. This is valid because the pointer count in the parameter preceding the unused pointers have been defaulted to zero. Thus, the unused pointers are not expected.

• The end of the data portion of the physical line (i.e., Column 72 in the Global Section, and Column 64 in the Parameter Data Section) is not to be constructed to act as either a parameter delimiter or a record delimiter.

• The parameter delimiter and record delimiter characters do not maintain their special significance when included within a string constant.

• A numeric constant, including its trading delimiter, cannot extend across a line boundary.

1.2.4 File structure

The file contains six different subsections which must appear in order as follows:

• a. Flag Section (Not always present)
• b. Start Section

- c. Global Section
- d. Directory Entry Section
- e. Parameter Data Section
- f. Terminate Section

These sections are contiguous with no intervening blank lines.

1.2.4.1 Start Section

The start Section of the file is designed to provide a human readable prologue to the file. There must be at least one start record. All records in the record must have the letter S in Column 73 and a sequence number in Column 73 through 80. The information in Columns 1 through 72 need not be formatted in any special way except that the ASCII character set shall be used. The following is an example of an start section.

1 72	73 80
This section is a human readable prologue to the file. It can contain an arbitrary number of lines using ASCCI characters in column 1-72	S0000001 S0000002 S0000003

1.2.4.2 Global Section

The Global Section of the file contains the information describing the preprocessor and the information needed by the postprocessor to handle the file. All records in the Global Section shall contain the letter G in Column 73 and a sequence number.

The parameters in the Global Section are the following ones:

1. String ▸ Parameter delimiter character (default is comma).
2. String ▸ Record delimiter character (default is semicolon).
3. String ▸ Product identification from sending system
4. String ▸ File name.
5. String ▸ System ID.
6. String ▸ Preprocessor version
7. Integer ▸ Number of binary bits for integer representation.
8. Integer ▸ Maximum power of ten representable in a double precision floating point number.
9. Integer ▸ Number of significant digits in a single precision floating point number on the sending system.
10. Integer ▸ Maximum power of ten representable in a double. precision floating point number on the sending system.
11. Integer ▸ Number of significant digits in a double precision floating point number in the sending system.

12. String ▸ Product identification fro the receiving system.
13. Real ▸ Model space scale.
14. Integer ▸ Unit Flag.
15. String ▸ Units.
16. Integer ▸ Maximum number of line weight gradations (default=1).
17. Real ▸ Width of maximum line weight in units.
18. String ▸ Date & time of exchange time generation 13HYYMMDD.HHNNSS (year, month, day, hour, minutes and seconds).
19. Real ▸ Minimum user intended resolution of the granularity of the model expressed in units.
20. Real ▸ Approximate maximum coordinate occurred in the model expressed in units.
21. String ▸ Name of the author.
22. String ▸ Author's organization.
23. Integer ▸ Integer value corresponding to the version of the Specification used to create this file.
24. Integer ▸ Drafting standard in compliance to which the data encoded in this file was generated.
25. String ▸ Date and time the model was created or last modified, whichever occurred last, 13HYYMMDD.HHNNSS.

1.2.4.3 Directory Entry Section

The Directory entry section has one directory entry for each entity in the file. The directory entry for each entity is fixed in size and contains twenty fields of eight characters each, spread accrues two consecutive eighty character lines. Data are right justified in each field. With the exception of the fields numbered 10, 16,17,18, and 20, entries in all the fields in this section will be either integer constants and pointer constants.

The purposes of the Directory Entry Section are to provide an index to the file and to contain attribute information for. The order of the directory entries within the Directory Entry Section is arbitrary with the exception that a definition entity must precede all of its instances.

Within the Directory Entry Section, a field consisting of whole of blanks is to be considered to have not been specified and should be given a default value where possible. Default values are not allowed in Fields 1,2,10,11,14 and 20. The actual values to be assigned as defaults will vary depending on the entity type.

Some of the fields in the directory entry can contain either an attribute value or pointer to en entity containing a set of such values. In these fields a positive value indicates an integer constant while for a negative value the absolute value should be taken and the result interpreted as a pointer constant.

Since valid fields have sequence numbers increasing from one, zero is a valid pointer value only when a specific interpretation of a zero has been defined in the specification. In such cases an empty field or a blank field is equivalent to the zero field.

1 8	9 16	17 24	25 32	33 40	41 48	49 56	57 64	65 72	73 80
(1) Entity Type Number #	(2) Para-meter Data ⇒	(3) Structure #,⇒	(3) Line Font Pattern #,⇒	(5) Level #,⇒	(6) View 0,⇒	(7) Trans-forma. Matrix 0,⇒	(8) Label Display Assoc. 0,⇒	(9) Status Number #	(10) Sequen. Number D#
(11) Entity Type Number #	(12) Line Weight Number #	(13) Colour Number #,⇒	(14) Parame-ter Line Count #	(15) Form Number #	(16) Reser-ved	(17) Reser-ved	(18) Entity Label	(19) Subs-cript Number #	(20) Sequen. Number D#+1

Nomenclature:

n	- Field Number
#	- Integer
⇒	- Pointer
#,⇒	- Integer or pointer (pointer has negative sign)
0,⇒	- Zero or pointer

The previous figure gives an abbreviated listing of the fields making up the directory entry for each entity. This nomenclature is used to describe all the tables in the rest of the specification with the following additions an exceptions:

• If the field is blank, it is defaulted, and the postprocessor will interpreted as a zero. (Exception: Fields 16,17, which are undefined, and 18, 18 which is treated as an empty text string).

• Explicit values in fields are the only allowed values,e.g., the Entity type Number and the form Number.

• The symbol < n.a > is used to indicate that the field has no meaning for this entity. A preprocessor must set the field to either zero or blank. A postprocessor will ignore the value altogether.

• In the status Number field, the following symbols are used:

  • The symbol (**) has the same meaning as < n.a >; a preprocessor must set this field to 00.

  • The symbol (??) means that an appropriate value from the defined range for this field must be used for each instance of the entity.

  • An explicit numeric value (e.g. 00 or 22) is the only value that may be used in the field. The value 00 will often be used in place of **.

• Footnotes are used to indicate that the values of some fields should be ignored under certain conditions.

The following is a descriptio of all the entries of the Directory Entry :

▸ **Entity Type Number:** Identifies the entity type.
▸ **Parameter Data:** Pointer to the first line of the parameter data record for the entity. The letter P is not included.
▸ **Structure:** Negated pointer to the directory entry of the definition entity that specifies this entity's meaning. The letter D is not included. The integers values 0, 1, and 2 are permissible in this field but should be disregarded.
▸ **Line Font Pattern:** Line font pattern or negated pointer to the directory entry of a Line Font Definition Entity (Type 304).
▸ **Level:** Number of the level upon which the entities resides, or a negated pointer to the directory entry of Definition Levels Property Entity (Type 406, form 1) which contains a list of levels upon which the entity resides.
▸ **View:** Pointer to the directory entry of a View Entity (Type 410), or pointer to a views Visible Associativity Instance (Type 402, form 3 or 4), or integer zero (default).
▸ **Transformation matrix:** Pointer to the directory entry of a Transformation Matrix Entity (Type 124) used in defining this entity; zero (default) implies the identity transformation matrix and the zero translation vector will be used.
▸ **Label Display Association:** Pointer to the directory entry of a label Display Associativity (Type 402, Form 5). The value of zero indicates no label display associativity.
▸ **Status Number:** Provides four two-digit status values which are entered from left to right in the status number field in the order given below:

- 1-2 Blank Status
  - 00 - Visible
  - 01 - Blanked
- 3-4 Subordinate Entity Switch
  - 00 Independent
  - 01 Physically Dependant
  - 02 Logically Dependant
  - 03 Both (01) and (02)
- 5-6 Entity Use Flag
  - 00 Geometry
  - 01 Annotation
  - 03 Other
  - 04 Logical/Positional
  - 05 2D Parametric
- 7-8 Hierarchy
  - 00 Global top down
  - 01 Global defer
  - 02 Use hierarchy property

▸ **Section Code and Sequence Number:** Physical count of this line from the beginning of the Directory Entry Section, precede by the letter D (odd number).
▸ **Entity Type Number:** (Same as field 1).

▸ **Line Weight Number:** System display thickness; given as a gradation value in the range of 0 to the maximum (Parameter 16 of the Global Section).
▸ **Colour Number:** Colour number or negated pointer to the directory entry of a Colour Definition Entity (Type 314).
▸ **Parameter Line Count:** Number of lines in the parameter data record for this entity.
▸ **Reserved for future**
▸ **Reserved for future**
▸ **Entity Label:** Up to eight alphanumerical characters (right justified).
▸ **Entity Subscript Number:** 1 to 8 digit unsigned number associated with the label.
▸ **Section Code and Sequence Number:** Same meaning as field 10 (even number).

1.2.4.4 Parameter Data Section

The Parameter Data Section of the file contains the parameter data associated with each entity. The following information is true for all the parameter data.

Parameter data are placed in free format with the first field always containing the entity type number. Therefore, the entity type number and a parameter delimiter (default is comma) precede parameter for each entity. The free format part of a parameter line ends in Column 64. Column 65 shall contain a blank. Columns 66 through 72 on all parameter lines contain the sequence number of the first line in the directory entry of the entity for which parameters data is being presented. Column 73 of all lines in the parameter section shall contain the letter P and Column 74 through 80 shall contain the sequence number.

1 64		66 72	73 80
Entity type number followed by parameter delimiter followed by parameters separated by parameter delimiters.		DE Pointer	P0000001
Parameters separeted by parameter delimiters followed by record delimiter		DE Pointer	P0000002
⋮		⋮	⋮

1.2.4.5 Terminate Section

There is only one line in the Terminate Section of the file. It is divided into ten fields of eight columns each. The Terminate Section must be the last line of the file. It has a "T" in Column 73 and Columns 74 through 80 contain the sequence number with a value of one.

Each field in the Terminate record contains a section identifier, left-justified in the field, and the last sequence number in that section, right justified in the field. The fields are defined in the figure below:

1 8	9 16	17 24	25 32	33 40	41 48	49 56	57 64	65 72	73 80
S 2	G 3	D 500	P 261	Not used	Not used	Not used	Not used	Not used	T 1

1.3 Entity Types

1.3.1 General

The Parameter Data record for each of the entities used in this application is described in this chapter. The fields for this record vary from entity to entity.
The meanings of the directory entry fields remain the same across all entities, those entities making used of the Field 15 (Form) in the directory entry are indicated and the various options are listed.

1.3.2 Line Entity (Type 110)

A line is a bounded, connected portion of a parent straight line which consist of more than one point. A line is defined by its end points. Each end point is specified relative to definition space by triple coordinates. A direction is associated with the line by considering the start point to be listed first and the terminate point second.

• **Directory Entry**

1 8	9 16	17 24	25 32	33 40	41 48	49 56	57 64	65 72	73 80
(1) Entity Type Number 110	(2) Para-meter Data ⇒	(3) Structure <n.a>	(3) Line Font Pattern #,⇒	(5) Level #,⇒	(6) View 0,⇒	(7) Trans-forma. Matrix 0,⇒	(8) Label Display Assoc. 0,⇒	(9) Status Number #	(10) Sequen. Number D#
(11) Entity Type Number 110	(12) Line Weight Number #	(13) Colour Number #,⇒	(14) Parame-ter Line Count #	(15) Form Number 0	(16) Reser-ved	(17) Reser-ved	(18) Entity Label	(19) Subs-cript Number #	(20) Sequen. Number D#+1

• **Parameter Data**

Index	Name	Type	Description
1	X1	Real	Start point P1
2	Y1	Real	
3	Z1	Real	
4	X2	Real	End point P2
5	Y2	Real	
6	Z2	Real	

1.3.3 Arc Entity (Type 100)

A circular arc is a connected portion of a parent circle which consists of more than one point. The definition space coordinate system is always chosen so that the circular arc lies in a plane either coincident with or parallel to the XT,YT plane.

• Directory Entry

1 8	9 16	17 24	25 32	33 40	41 48	49 56	57 64	65 72	73 80
(1) Entity Type Number 100	(2) Para-meter Data ⇒	(3) Structure <n.a>	(3) Line Font Pattern #,⇒	(5) Level #,⇒	(6) View 0,⇒	(7) Trans-forma. Matrix 0,⇒	(8) Label Display Assoc. 0,⇒	(9) Status Number #	(10) Sequen. Number D#
(11) Entity Type Number 100	(12) Line Weight Number #	(13) Colour Number #,⇒	(14) Parame-ter Line Count #	(15) Form Number 0	(16) Reser-ved	(17) Reser-ved	(18) Entity Label	(19) Subs-cript Number #	(20) Sequen. Number D#+1

• Parameter Data

Index	Name	Type	Description
1	ZT	Real	Parallel ZT displacement of the arc
2	X1	Real	Arc centre abscissa
3	Y1	Real	Arc centre ordinate
4	X2	Real	Start point abscissa
5	Y2	Real	Start point ordinate
6	X2	Real	Terminate point abscissa
7	Y2	Real	Terminate point ordinate

1.3.4 Copius Data Entity (Type 106)

This entity stores data points in the form of pairs, triples or sextuples. An interpretation flag value signifies which of this forms is being used. This value is one of the parameter data entries. The interpretation flag is abbreviated by the letters IP.

Fields 15 of the Directory Entry accommodates a Form Number:

- Form 31-39: represents the Section Entity.
- Form 40: represents witness line.

• **Directory Entry**

1 8	9 16	17 24	25 32	33 40	41 48	49 56	57 64	65 72	73 80
(1) Entity Type Number 106	(2) Para- meter Data ⇒	(3) Structure <n.a>	(3) Line Font Pattern #,⇒	(5) Level #,⇒	(6) View 0,⇒	(7) Trans- forma. Matrix 0,⇒	(8) Label Display Assoc. 0,⇒	(9) Status Number #	(10) Sequen. Number D#
(11) Entity Type Number 106	(12) Line Weight Number #	(13) Colour Number #,⇒	(14) Parame- ter Line Count #	(15) Form Number 11-13,63	(16) Reser- ved	(17) Reser- ved	(18) Entity Label	(19) Subs- cript Number #	(20) Sequen. Number D#+1

• **Parameter Data**

Index	Name	Type	Description
1	IP	Int	Interpretation Flag:IP=1
2	N	Int	Number of data points: N is even
3	ZT	Real	Common Z displacement
4	X1	Real	First data point abscissa
5	Y1	Real	First data point ordinate
.	.	.	
.	.	.	
3+2N	YN	Real	Last data point ordinate

1.3.5 General Note Entity (Type 212)

A general note entity consist of one or more text strings. Each text string contains text, a starting point, a text size, an angle of rotation of the text. The font code (FC) is an integer specifying the desired characters set and its associated displayed characteristics. Positive values are predefined fonts. Negative values point to implementor-defined fonts or modifications to a predefined font, through the use of text definition font entity.

• **Directory Entry**

1 8	9 16	17 24	25 32	33 40	41 48	49 56	57 64	65 72	73 80
(1) Entity Type Number 212	(2) Para- meter Data ⇒	(3) Structure <n.a>	(3) Line Font Pattern 1	(5) Level #,⇒	(6) View 0,⇒	(7) Trans- forma. Matrix 0,⇒	(8) Label Display Assoc. 0,⇒	(9) Status Number #	(10) Sequen. Number D#
(11) Entity Type Number 212	(12) Line Weight Number #	(13) Colour Number #,⇒	(14) Parame- ter Line Count #	(15) Form Number 0	(16) Reser- ved	(17) Reser- ved	(18) Entity Label	(19) Subs- cript Number #	(20) Sequen. Number D#+1

• **Parameter Data**

Index	Name	Type	Description
1	NS	Int	Number of text strings
2	NC1	Int	Number of chars of the first text
3	WT1	Real	Box width
4	HT1	Real	Box Height
5	FC1	Int,Pt	Font Code (default=1)
6	SL1	Real	Start ang of text in rads
7	A1	Real	Rotation angle in radians
8	M1	Int	Mirror flag
9	VH1	Int	Text hor=0, Text ver=1
10	XS1	Real	First text start point
11	YS1	Real	
12	ZS1	Real	Z depth
13	T1	Str	First text string
14	NC2	Int	Num. of chars in 2 str
.	.	.	
.	NCN	Int	Number of chars in last str
.	.	.	
.	TNS	Str	Last String

1.3.6 Arrow Entity (Type 214)

A Leader Entity consist of one or more line segments. The first segment begins with an arrow head. Remaining segments successively link to a presumed text item. An individual segment is assumed to extend from the end point of its predecessor in the segment list to its defined point.

• **Directory Entry**

1 8	9 16	17 24	25 32	33 40	41 48	49 56	57 64	65 72	73 80
(1) Entity Type Number 214	(2) Para-meter Data ⇒	(3) Structure <n.a>	(3) Line Font Pattern #,⇒	(5) Level #,⇒	(6) View 0,⇒	(7) Trans-forma. Matrix 0,⇒	(8) Label Display Assoc. 0,⇒	(9) Status Number #	(10) Sequen. Number D#
(11) Entity Type Number 214	(12) Line Weight Number #	(13) Colour Number #,⇒	(14) Parame-ter Line Count #	(15) Form Number 0	(16) Reser-ved	(17) Reser-ved	(18) Entity Label	(19) Subs-cript Number #	(20) Sequen. Number D#+1

• **Parameter Data**

Index	Name	Type	Description
1	N	Int	Number of segments
3	AD2	Real	Arrowhead width
4	ZT	Real	Z depth
5	XH	Real	ArrowHead coordinates
6	YH	Real	
7	X1	Real	First segment tail coordinate pair
8	Y1	Real	
.	.	.	
5+2*N	XN	Real	Last segment tail coordinate pair
6+2*N	YN	Real	

1.3.7 Composite Curve Entity (Type 102)

A composite curve is a continuous curve that results from the grouping of certain individual constituent entities into a logical unit.

A composite curve is defined as an ordered list of entities consisting of point,connect point, and parameterized curve entities. The list of entities appears in the parameter data entry. There, each entity to appear in the defining list is indicated by means of a pointer to the directory entry of that entity.

• **Directory Entry**

1 8	9 16	17 24	25 32	33 40	41 48	49 56	57 64	65 72	73 80
(1) Entity Type Number 102	(2) Para-meter Data ⇒	(3) Structure <n.a>	(3) Line Font Pattern #,⇒	(5) Level #,⇒	(6) View 0,⇒	(7) Trans-forma. Matrix 0,⇒	(8) Label Display Assoc. 0,⇒	(9) Status Number #	(10) Sequen. Number D#
(11) Entity Type Number 102	(12) Line Weight Number #	(13) Colour Number #,⇒	(14) Parame-ter Line Count #	(15) Form Number 0	(16) Reser-ved	(17) Reser-ved	(18) Entity Label	(19) Subs-cript Number #	(20) Sequen. Number D#+1

• **Parameter Data**

Index	Name	Type	Description
1	N	Int	Number of entities
2	DE1	Ptr	Ptr to DE of the first entity
.	.	.	
1+N	DEN	Ptr	Ptr the DE of the last entity

Appendix 2 : DMIS specification

2.1 Syntaxis

The DMIS vocabulary consists of ASCII characters which are combined to form words, labels, parameters and variables. These are then combined to form definitions and commands. Definitions and commands are combined to form programm subunits, along with more commands and definitions, are combined to form entire DMIS programs.

2.1.1 Character

1. Only the ASCII printable characters, plus the carriage return and line feed are allowed in DMIS.
2. Upper and lower case alpha characters are considered to be the same. The exception to this rule is the text strings passed with TEXT and FILNAM commands.
3. Numerical data can be positive or negative.

2.1.2 Words, Labels, Parameters and Variables

Characters are combined to form vocabulary words, labels, parameters and variables. Vocabulary words are classified as either major words or minor words. Variables can be either real variables or string variables.

- **Major words** : consist of a minimum of two and a maximum of six alpha characters (list of major words).

- **Minor words** : consist of a minimum of two and maximum of six alpha characters. A minor word is used to modify a major word or to describe certain parameters in the command.

- **Labels** : consist of one to ten alphanumeric characters(except datum labels that are from one to two) enclosed in parentheses, and are assigned in the inspection program. They are used to name features, tolerances, coordinate systems, sensors, output data formats, datums, macro routines, text strings and program lines.

- **Parameters** : are numeric values (positive or negative).

- **Real variables** : they are either one alpha character, or an alpha character followed by an integer between 0 and 9. They are used to compared measured results against some values.

2.1.3 Commands and definitions

Words, parameters, labels and variables are combined to form commands and definitions:

• Commands direct the DME or receiving system to perform some function. The basic structure of a command is as follows :

```
MAJOR_WORD/MINOR_WORD(S), PARAMETER(S)
```

• Definitions describe various things. The basic structure of a definition is as follows:

```
F(label)=MAJOR_WORD/MINOR_WORD(S),PARAMETER(S)
```

Often a command contains a pointer to a definition. In this case, the definition is used in the execution of the command.

2.1.4 Program Subunits

Commands and/or definitions can be combined to form program subunits. They are a logically grouped list of statements which perform some function. They are recognized by the first and last line in the subunit. There are four types of program subunits:

• **Measurement sequence :** Measurement sequences begin with MEAS and are terminated with the ENDMES command. Example:

```
MEAS/CIRCLE,F(CIR11),4
PTMEAS/CART,-15.5,35.5,107.5,-1,0,0
PTMEAS/CART,-25.5,35.5,107.5,1,0,0
GOTO/-20.5,35.5,107.5
PTMEAS/CART,-20.5,40.5,107.5,0,-1,0
PTMEAS/CART,-20.5,30.5,107.5,0,1,0
ENDMEAS
```

Depending on the MODE command the DME will perform different when MEAS is encountered. Only in the program mode the DME will follow the commands after the MEAS. In the automatic mode the DME uses its own internal algorithm to perform the measurement of the circle. In the manual mode the operator will take manually the four points.

• **Motion Sequence :** Motion sequences direct a series of non-measurement moves. They begin with GOTARG and are terminated with ENDGO. Example:

```
GOTARG/-20.5,35.5,107.5
GOTO/20.5,35.5,120
GOTO/-20.5,35.5,107.5
ENDGO
```

If the program MODE is in AUTO, only the GOTARG command is executed. The DME calculates its own path to the end point. In the program mode the given moves are executed and in the manual mode the operator will move the probe to the end point.

• **Conditional :** Conditionals are used to compare a measurement result to some value and then branch the execution of the program based on the result. They begin with an IF and are terminated with ENDIF. Example:

```
IF X,GT,.125
---------
ELSE
---------
ENDIF
```

If a variable (X), which was previously assigned a value equal to a measurement result is tested to see if it is greater than .125.

• **Macro routines :** They are subroutines which can be executed with different values in the place of dummy parameters. They begin with a MACRO definition and are terminated with an ENDMAC. Example:

```
M(HOLPAT)=MACRO/X1,Y1,R,DIAM,"LABEL1","LABEL2"
---------------------------
---------------------------
ENDMAC
```

If a dummy parameters is to be replaced by a real number or variable, it is defined without quotation marks (X1, Y1, R, DIAM in the example). If a dummy parameter is to be replaced with a label, it is listed with quotation marks (LABEL1 and LABEL2 in the example).
To execute the MACRO, it has to be called with the real values.

```
CALL/M(HOLPAT),5, 7, 2.5, 5, (CSYS1), (CIRL2)
```

2.1.5 Programs and Output files

Programs and output files consist of a combination of commands, definitions and programs subunits. These files always have FILNAM as first line and ENDFIL as the last line.

DME DMIS output files are similar in syntaxis to the input files. Measurement results are passed in the form of actual feature and tolerance definitions that are similar in format to the output file (nominal) definitions. In addition, certain commands are passed through to the output file to indicate parameter settings at the time of measurement.

2.1.6 Delimiters

Slashes, commas, parentheses, apostrophes and quotation marks are used as delimiters in DMIS. Blank lines are insignificant and are ignored during translation. Spaces are not allowed within DMIS words, labels, parameters and variables.

2.1.6.1 Slash (/)

The slash character '/' (ASCII 47) is used to separate major and minor words. A major word can either be a statement by itself, or can have some modifying minor words. When a major word is used by itself it has no slash character: MAJOR_WORD

When a major_word is used with a minor word, the slash follows the major word and precedes the minor word.

```
MAJOR_WORD/MINOR_WORD
```

2.1.6.2 Commas (,)

Commas ',' (ASCII 44) as used as general delimiters to separate minor words and parameters:

```
MAJOR_WORD/MINOR_WORD1,MINOR_WORD2,PARAM1,PARAM2
```

2.1.6.3 Parentheses '()'

Parentheses '()' (ASCII 40 and 41) are used to delimit labels. A label is precede with a left parentheses and followed by a right parenthesis. For example, if a circle has a label of CIRCLE_1, it is denoted by:

```
F(CIRCLE_1)
```

All labels have balanced parentheses. That is, a left parentheses is always followed by a right parentheses.

2.1.6.4 The apostrophe (')

The apostrophe (') (ASCII 39), is used to delimit the start and end of a text string. For example:

```
FILNAM/'456 test dated 9/04/93'
```

Use two apostrophes, one before the one required, when an apostrophe is required within a text string.

2.1.7 Line length and Terminator

A DMIS file consist of records with a variable line length(maximum 80 characters). Each line is terminated with a carriage return and line feed. A single dollar sign ($) acts as a line continuation.

2.1.8 Programming comments

Programming comments are lines of text insert into a DMIS inspection program to aid in program debugging and to document portions of the program. They are not to be interpreted by any automated system.

Programming comments are signified by two dollars signs ($$) as the first two characters in line.

2.2 Data Structure

2.2.1 General Programming Considerations

Some general programming considerations should be kept in mind when generating an inspection program in the DMIS vocabulary. These refer to pointer structure, branching, modality and default settings.

- **Pointers** : Several commands in the DMIS vocabulary have pointers to definitions. For example, a measurement command has a pointer to the feature definition of the feature being measured. A definition MUST occur in the file prior to the command that points to it.

- **Branching** : Branching must always be forward in a program. A program cannot branch in the middle of a program subunit.

- **Modality** : Modal commands set some machine parameter or other condition which stays set until the command is reissued. Non-modal are in effect only for a single execution of the program.

- **Default settings** : Some commands have default settings. If the command is not issued in the program, it is equivalent to issuing it with the default values.

2.2.2 Features

Features are geometric elements which may or may not be on the part (point in the space). If the feature is on the part, it may be referred to in two ways :

- **Nominal feature** : is the feature definition that comes from the CAD model or part drawing. Gives the nominal size, location, and orientation of the feature. A label is assigned to the feature in the definition. Example :

  F(CIRCLE1)=FEAT/CIRCLE, INNER,CART,10,10,5,0,0,1,8

- **Actual feature** : is the measured feature. The actual feature definition gives the measured size, location and orientation of the feature. The label is the same to the nominal one except than it is preceded by an FA.
  FA(CIRCLE1)=FEAT/CIRCLE,INNER,CART,9.8,9.9,5,0,0,1,7.9

Features may either be measured by the DME or they may be constructed by the DME as a best fit through other features. When a feature is constructed, at least one of the features used in its construction must be a measured feature.

As illustrated in the two examples above, feature orientation consist of an x,y,z point and an i,j,k vector. The i,j,k is a unit vector and areal number between -1 and 1. The vector can be either normal or directional. Unless otherwise indicated, all normals point away from the part surface and all directional vectors point from the first point towards the second.

The current version of DMIS supports a set of feature definitions which are simple geometric elements:

- arc	- gcurve(complex curve)
- circle	- gsurf (complex surface)
- cone	- line
- cube	- plane
- cylinder	- point
- ellipse	- sphere

2.2.3 Tolerances

Tolerances are referred to in two ways:

• **Nominal tolerances :** are those ones found in the CAD model or part drawing.

• **Actual tolerances :** are the evaluated tolerances computed from measured features.

Tolerances are assigned the same labels in the nominal and actual definitions. Nominals are preceded by "T" and actuals are preceded by "TA". The syntaxis of the actual is very similar but not identical to the nominal.

The tolerances supported are the next ones :

- angle(size tolerance)	- profile of surface
- diameter (size tolerance)	- circular runout
- radius (size tolerance)	- total runout
- circularity (roundness)	- angularity
- cylindricity	- parallelism
- flatness	- perpendicularity
- straightness	- concentricity
- profile of line	- position (true position)

In addition two relationships have been added and included with the tolerances. While these are not strictly tolerances, they are relationships between two features or between a feature and a datum. These additions are angle between features (ANGB) and distance between features (DISTB).

2.2.4 Coordinate Systems

Part coordinate systems are set-up with datums, and are used to align the part with respect to the DME machine coordinate system as well as to establish a reference for feature and tolerance evaluations. This is done by establishing a part coordinate system with the DATSET command. DATSET specifies which features are to be used in establishing the datum reference frame and assigns labels to the datums.

Once a coordinate system has been established, it can be rotated or translated in order to establish a new one by reusing the DATSET statement. All coordinate systems are assigned labels. Translations and rotations of the part coordinate system can be defined by a specified amount or by an alignment with the feature.

The DATDEF statement provides a means to assign a datum label, dat is DAT (x), to a previously measured feature. This datum label can then be used with the DATSET statement to define the part coordinate system.

2.3 Feature Definitions

Feature definitions are used to describe the nominal size, location and orientation of features, and to assign labels to the features. They all have a label to identify them which is an alphanumeric name assigned to the feature, and is up to 10 characters in length.

2.3.1 FEAT/ARC

Function:	Defines a nominal arc whose plane lies parallel to the workplane and assigns it to a label.
Input Form:	F(label)=FEAT/ARC,var1, var2,i,j,k,rad,ang1,ang2
Output Form:	FA(label)=FEAT/ARC,var1,var2,i,j,k,rad,ang1,ang2
Where:	var1 can be: INNER
	OUTER
	var2 can be: CART,x,y,z
ARC:	signifies that the features is an arc.
INNER:	signifies that the inside of an arc is going to be measured (i.e.,a fillet).
OUTER:	signifies that the outside of an arc is going to be measured (i.e.,a round).
CART:	signifies that the centre is given by cartesian coordinates.
x,y,z:	are the cartesian coordinates of the centre point of the arc.
i,j,k:	is the direction vector of the plane that the arc lies in.
rad:	is the radius of the arc.
ang1:	is the start angle of the arc. Use the right hand rule for sign conventions.
ang2:	is the positive include angle of the arc relative to ang1.

2.3.2 FEAT/CIRCLE

Function:	Defines a normal circle and assigns it a label.
Input Form:	F(label)=FEAT/CIRCLE,var1,var2,i,j,k,diam
Output Form:	FA(label)=FEAT/CIRCLE,var1,var2,i,j,k,diam
Where:	var1 can be: INNER
	OUTER
	var2 can be: CART,x,y,z
CIRCLE:	signifies that the feature is a circle.

INNER:	signifies that the inside of the circle is to be measured (i.e., a hole).
OUTER:	signifies that the outside of a circle is to be measured (i.e., a boss).
CART:	signifies that the centre is given by cartesian coordinates.
x,y,z:	are the cartesian coordinates of the centre point of the circle.
i,j,k:	is the direction vector of the plane that the circle lies in.
diam:	is the diameter of the circle.

2.3.3 FEAT/LINE

Function:	Defines a nominal line and assigns it to a label.
Input Form:	F(label)=FEAT/LINE,var1,ni,nj,nk
Output Form:	FA(label)=FEAT/LINE,var1,ni,nj,nk
Where:	var1 can be: BND, var2
	var2 can be: CART,e1x,e1y,e1z,e2x,e2y,e2z

LINE: signifies that the feature is a line.

BND:	signifies that a bounded line is to be defined.
CART:	signifies that the coordinates of the points in the line are given in cartesian coordinates.
ni,nj,nk:	is the normal vector of the plane in which the line lies, which can be used for probe compensation.
e1x,e1y,e1z e2x,e2y,e2z	are the cartesian coordinates of the two end points of the line.

2.3.4 FEAT/POINT

Function:	Defines a nominal point and assigns it to a feature label.
Input Form:	F(label)=FEAT/POINT,var1,i,j,k
Output Form:	FA(label)=FEAT/POINT,var1,i,j,k
Where:	var1 can be: CART,x,y,z
POINT:	signifies that the feature is a point.
x,y,z:	are the cartesian coordinates of the point itself.
i,j,k:	is a vector, normal to and pointing away from, in which the point lies, that can be used for probe compensation.

2.4 Tolerance Definition

Tolerance definitions are used to describe generic tolerances. They also provide for label names to be assigned to each tolerance. This label names are used with EVAL or OUTPUT statements to associate the tolerances to the features. The label is an alphanumeric name assigned to the tolerance, and is up to 10 characters in length.

2.4.1 TOL/DIAM

Function:	Specifies a diameter tolerance and assigns it a label.
Input Form:	T(label)=TOL/DIAM,lotol,uptol
Output Form:	TA(label)=TOL/DIAM,dev,var1

Where:	var1 can be: INTOL OUTOL
DIAM:	signifies that the tolerance is a diameter tolerance.
lotol:	is the signed lower tolerance value applied to the diameter.
uptol:	is the signed upper tolerance value applied to the diameter.
dev:	is the deviation - the arithmetic difference between the actual value and the nominal value.
INTOL:	signifies the actual is within tolerance.
OUTOL:	signifies the actual is out of tolerance.

2.4.2 TOL/WIDTH

Function:	Specifies a linear size (width) tolerance and assigns to it a label.
Input Form:	T(label)=TOL/WIDTH,lotol,uptol
Output Form:	TA(label)=TOL/WIDTH,dev,var1
Where:	var1 can be: INTOL OUTOL
WIDTH:	signifies that the tolerance is a linear size (width) tolerance.
lotol:	is the signed lower tolerance value applied to the linear size (width).
uptol:	is the signed upper tolerance value applied to the linear size (width).
dev:	is the deviation - the arithmetic difference between the actual value and the nominal value.
INTOL:	signifies the actual is within tolerance.
OUTOL:	signifies the actual is out tolerance.

2.4.3 TOL/DISTB

Function:	Specifies a distance and a tolerance and assigns a label to them.
Input Form:	T(label)=TOL/DISTB,var2,var3
Output Form:	TA(label)=TOL/DISTB,var1,var2,var3
Where:	var1 can be: INTOL OUTOL var2 can be: NOMINL,dist,lotol,uptol var3 can be: XAXIS YAXIS ZAXIS PT2PT
DISTB:	the value and tolerances are applied to the distance between two features.
INTOL:	signifies the actual is within tolerance.
OUTOL:	signifies the actual is out of tolerance.
NOMINL:	signifies a nominal distance with lower and upper tolerance.
dist:	is the nominal or the actual measured value.
lotol:	is the signed lower tolerance assigned to the nominal distance.
uptol:	is the signed upper tolerance assigned to the nominal distance.
XAXIS:	signifies that the distance between is along the X axis.
YAXIS:	signifies that the distance between is along the Y axis.

ZAXIS:	signifies that the distance between is along the Z axis.
PT2PT:	signifies that the distance between is point to point, or feature to feature.

2.4.4 TOL/CORTOL

Function:	Specifies bidirectional positional tolerancing of features in Cartesian or polar coordinates and assigns it to a label.	
Input Form:	T(label)=TOL/CORTOL,var1,lotol,uptol	
Output Form:	TA(label)=TOL/CORTOL,var1,dev,var2	
Where:	var1 can be:	XAXIS
		YAXIS
		ZAXIS
	var2 can be:	INTOL
		OUTOL

CORTOL:	signifies bidirectional positioning tolerancing
XAXIS:	signifies that the rectangular coordinate method is to be used to tolerance the position along the X axis.
YAXIS:	signifies that the rectangular coordinate method is to be used to tolerance the position along the Y axis.
ZAXIS:	signifies that the rectangular coordinate method is to be used to tolerance the position along the Z axis.
INTOL:	signifies the actual is within tolerance.
OUTOL:	signifies the actual is without tolerance.
lotol:	is the signed lower tolerance value.
uptol:	is the signed upper tolerance value.
dev:	is the deviation from the nominal value.

2.5 Part Coordinate System

This section defines the statements used to manipulate the coordinate system.

2.5.1 TRANS

Function:	Translates a part coordinate system along an axis, and assigns it to a label.	
Input Form:	D(label)=TRANS/var1,var2	
Output Form:	DA(label)=var1,var2	
Where:	var1 can be:	XORIG
		YORIG
		ZORIG
	var2 can be:	value

XORIG:	signifies that the coordinate system origin is to be translated on the X axis if a value is given.
YORIG:	signifies that the coordinate system origin is to be translated on the Y axis if a value is given.
ZORIG:	signifies that the coordinate system origin is to be translated on the Z

	axis if a value is given.
value:	is the distance the coordinate system origin is to be translated.

2.5.2 WKPLAN

Function:	Used to explicitly declare or change a working plane.
Input Form:	WKPLAN/var1
Output Form:	None
Where:	var1 can be: XYPLAN YZPLAN ZXPLAN
XYPLAN:	signifies that the XY plane of the current part coordinate system is the working plane.
YZPLAN:	signifies that the YZ plane of the current part coordinate system is the working plane.
ZXPLAN:	signifies that the ZX plane of the current part coordinate system is the working plane.

2.5.3 SNSLCT

Function:	Selects the sensor to be used for the measurement.
Input Form:	SNSLCT/var1
Output Form:	None.
Where:	var1 can be: S(label1)
S(label1):	is a previously defined sensor.

2.6 Motion and measurement statements

2.6.1 GOTO

Function:	Executes a sensor move and defines the endpoint of the move.
Input Form:	GOTO/x,y,z
Output Form:	None
x,y,z:	are the cartesian coordinates of the endpoint to which the sensor will travel relative to the origin of the active coordinate system.

2.6.2 MEAS

Function:	causes the DME to measure a feature.
Input Form:	MEAS/var1,F(label),n
Output Form:	None
Where:	var1 can be: ARC CIRCLE LINE POINT
F(label):	is the name of the previously defined feature to be measured.

n:	is the number of points to be taken in the measurement of the feature.

The MEAS statement is usually followed by a series of PTMEAS and GOTO statements. The MEAS statement is terminated with the ENDMES statement.

2.6.3 PTMEAS

Function:	Signifies that an automatic point measurement is to be performed.
Input Form:	PTMEAS/CART,x,y,z,i,j,k
Output Form:	None.
x,y,z:	are the nominal cartesian coordinates of the point to be measured.
i,j,k:	is the direction vector pointing away from the surface of the feature used in making the measurement.

2.7 Feature Construction

The CONST statement provides for the construction of features useful to the inspection process.

2.7.1 CONST

Input Form:	CONST/var1,F(label),BF,FA(label2),FA(label3)
Output Form:	None
Where:	var1 can be: CIRCLE LINE
F(label):	identifies the feature to be constructed.
BF:	signifies that the constructed feature is a best fit through the features that follow.

FA(label2):
FA(label3)
	are actual measured features to be used for construction.

2.7.2 CONST/POINT

Input Form:	CONST/POINT,F(label),MIDPT,FA(label2),FA(label3)
Output Form:	None.
POINT:	signifies that a point is to be constructed.
F(label):	is the previously defined nominal feature to be constructed.
MIDPT:	signifies that the feature to be constructed is to be the midpoint of the two previously defined features.

FA(label2):
FA(label3)
	are the two previously defined features to be used for the construction.

Appendix : 3 List of CMES commands

3.1Workpiece datum commands

3.1.1 Datum Move (DM)

Format: DM[,axis a]...[,axis c]

The DM command allows the user to move the workpiece datum to a theoretical point from which all subsequent dimensions will be related.

Note: If the command is used twice, the shift will be relative to the datum's current position, NOT the Master Datum.

[,axis a]...[,axis c]
These parameters are optional and may be used to specify in which axis or axes the datum is to be moved. If it is omitted then all Datums (X,Y and Z) are moved. The parameter is used to move one axis only (e.g, DM,Y) or group of axis (e.g, DM,X,Z).

Example:
```
DM,Y
30
```
Signifies that the master datum is going to be moved 30 units in the Y axis.

3.1.2 Restore Master datum (RM)

Format: RM[,axis]

The RM command is used to restore the master datum as the current datum from which all subsequent dimensions will be related. It may be used to cancel datums created either by the SD (Sub Datum) or DM (Datum Move).

[,axis]
This parameter is optional and may be used to specify in which axis the master datum is to be restored (e.g, RM,Z). If it is omitted then the master datum is restored in all the axes (X,Y and Z).
Note: Only one axis may be entered with each use of the command.

3.1.3 Sub Datum

Format: SD[,axis a]...[,axis c]

The SD command may be used to create an additional datum that temporarily overrides the master datum.

A sub datum can be created by using the command immediately after the measurement of the feature to be datumed or by using co-ordinates recalled from the save point list.

There is no limit to the number of sub datums that can be created, however only one datum can be in force at a specific time.

[,axis a]...[,axis c]

These parameters are optional and may be used to specify which axes are to be datumed. If they are omitted then all axes will be datumed. The parameters are used to zero one axis only (e.g, SD,X) or groups of axes (e.g, SD X,Z).

3.2 Workpiece Measurement Commands

3.2.1 Inside Diameter (ID)

Format: ID,plane[,points][tol]

The ID command is used to determine the diameter and centre coordinates of a complete or partial hole. The result is output either in RC or PO mode and is relative to the current axis system and datum.

,plane
This parameter is used to define the plane that the hole is located in and should be entered as either X,Y or Z.

[,points]
This parameter is used to define the number of points to be taken, the maximin being 40. If the parameter is omitted the command defaults four.

[tol]
The [tol] parameter is optional, it may be used to allow the input of the nominal and tolerance data to the command so that the result may be output with actual, nominal, tolerance and error headings (refer to "tolerancing Commands").

Example:

ID,Z 30 40 10	This signifies that the location of the hole is in the Z plane at 30 units in the X axis and 40 units in the Y axis from the master. The diameter of the hole is 10 units

3.2.2 Outside Diameter (OD)

Format: OD,plane[,points][tol]

The OD command is used to determine the diameter and centre co-ordinates of a complete or partial boss or pin. The result is output either in RC or PO mode and is relative to the current datum and axis system.

,plane
This parameter is used to define the plane that the boss or pin is located in and should be entered as either X,Y or Z.

[,points]
This parameter is used to define the number of points to be taken, the maximin being 40. If the parameter is omitted, the command defaults to four points.

[tol]
The [tol] parameter is optional, it may be used to allow the input of nominal and tolerance data to the command so that the result may be output with actual, nominal, tolerance and error headings. (refer to the "Tolerancing and Commands" section).

Example:

OD,Y	This signifies that the location of the boss is in the Y plane at 20
20	units in the X axis and 30 units in the Z axis from the master.
30	The diameter of the hole is 10 units
10	

3.2.3 Pitch Circle (PC)

Format: PC,plane[,points][tol]

The PC command is used to determine the diameter and centre co-ordinates of a group of holes or bosses. The result is output in either PO or RC mode and is relative to the current axis system and datum.

To measure a PCD of holes/pins, use the ID or OD commands to determine the feature centres, save each centre using the SP (Save Point) command then use the UP (Use Point) or (UG Use Group) to supply the centres to the PC command.

,plane
This parameter is used to define the plane that the hole is located in and should be enter as either X,Y, or Z.

[,points]
This parameter is used to define the number of points to be taken, the maximin being 40. If the parameter is omitted, the command defaults to four points.

[tol]
The [tol] parameter is optional, it may be used to allow the input of the nominal and tolerance data to the command so the result may be output with actual, nominal, tolerance and error headings (refer to "Tolerancing Commands").

Example:

```
PC,X
10
20
50
```

This signifies that the location of the pitch circle is in the X plane at 10 units in the X axis and 20 units in the Z axis from the master. The diameter of the hole is 50 units

3.2.4 Centre Measurement (CM)

Format: CM,axis[tol]

The CM command is used to determine the centre point between two measured points.

The command is primarily used to determine the dimension to the centre of a slot, but it may be also used to measure the centre point of various features such as 2 holes, a hole and a boss or a slot and a hole. The result is always in RC mode.

,axis
This is used to define in which axis the result is to be output and should be entered as X,Y or Z.

[tol]
The tol parameter is optional, it may be used to allow the input of nominal and tolerance data to the command so that the result may be output with the actual, nominal, tolerance and error headings (refer to "Tolerancing Commands").

Example: CM,Y. This signifies that the center of measurement is going to be calculated in the Y axis.

3.2.5 Length Inside (LI)

Format: LI,axis[tol]

The LI command is used to determine the length of a slot by taking one point either side of the feature, compensation for the probe stylus is applied automatically. The output is always in RC mode.

,axis
This parameter is used to define the axis in which the measurement is to be taken, it should be entered as either X,Y or Z.

[tol]
The tol parameter is optional, it may be used to allow the input of nominal and tolerance data to the command so that the result may be output with the actual, nominal, tolerance and error headings (refer to "Tolerancing Commands").

Example: LI/X

10	This signifies that calculation of the length is in the X axis, being the nominal distance 10.

3.2.6 Length Outside (LO)

The LO command is used to determine the width of a lug or dog by taking one point on either side of the feature, compensation for the probe stylus is applied automatically. The result is always in RC mode.

,axis
This parameter is used to define the axis in which the measurement is to be taken, it should be entered as either X,Y or Z.

[tol]
The tol parameter is optional, it may be used to allow the input of nominal and tolerance data to the command so that the result may be output with the actual, nominal, tolerance and error headings (refer to "Tolerancing Commands").

Example:

LO/Y 10	This signifies that calculation of the length is in the Y axis, being the nominal distance 10.

3.2.7 Length True (LT)

Format: LT,axis[tol]

The LT command is used to determine the length between two points without the need of a sub-datum to be created. No probe compensation is applied to the result which is always in RC mode.

,axis
This parameter is used to define the axis in which the measurement is to be taken, it should be entered as either X,Y or Z.

[tol]
The tol parameter is optional, it may be used to allow the input of nominal and tolerance data to the command so that the result may be output with the actual, nominal, tolerance and error headings (refer to "Tolerancing Commands").

Example:

LT/Z 10	This signifies that calculation of the length is in the Z axis, being the nominal distance 10.

3.2.8 Wall Thickness (WT)

The WT command is used to determine the radial length between two points, for example an oil seal groove. The output is always in PO mode.

In order to determine a correct radius a sub-datum must be created at the origin point, the datum may be created either by the SD or DM commands.

[±]
This parameter is optional, it is used to compensate the measured point by one probe radius. If it is omitted, probe compensation will not be applied. As the result is in PO mode, the probe compensation will be a radial value, i.e. a + sign will increase the measured radius whilst a - sign will reduce it. This command allows to measure an internal or external radius between two points.

,axis
This parameter is used to define the axis in which the measurement is to be taken, it should be entered as either X,Y or Z.

[tol]
The tol parameter is optional, it may be used to allow the input of nominal and tolerance data to the command so that the result may be output with the actual, nominal, tolerance and error headings (refer to "Tolerancing Commands").

Example:

WT+/Z 10	This signifies that calculation of the length is along the raidus, being the nominal distance 10.

3.3 Auto Inspection Commands

3.3.1 Move Absolute (#MA)

Format: #MA[,axis a]...[,axis c]

The #MA command causes the CMM to move to an absolute co-ordinate relative to the current axis system and datum. The target co-ordinate are input at the keyboard or via part program and may be input in rectangular or polar mode.

[,axis a]
This parameter is optional and may be used when in RC mode, to specify the axis (or axes) which are to move. If omitted, the command request target co-ordinates for all three axes.

Example:

#MA,X,Y 20 30	This signifies that the probe is going to move to the point x=20, y=30 and previous z.

3.3.2 Point (#PT)

Format: #PT[,axis a]...[,axis c]

The #PT command is used to automatically 'collect' points from the part being inspected.
Note: If the probe fails to touch the part, the CMM will continue to traverse at the touch speed until overtravel limit is reached, when the CMM will stop and generate the message: 'No touch detected'.
If the probe successfully contacts the part, the co-ordinate should then be saved by the SP (Save Point) command after which it may be supplied to any command requiring measured data. The #PT command in combination with #MA and SP commands provides an effective method of automating all measurements.

Example:

#PT,X,Y	This signifies that the probe is going to take the point located in the coordinates x=20, y=30 and previous z.
20	
30	

3.4 Data Manipulation

3.4.1 Save Point (SP)

Format: SP,store n[,line n]

The SP command is used after a measurement to store the X,Y and Z co-ordinates in a specific location. The co-ordinates will remain saved until the same location is used again.

Use of save points
The ability to save co-ordinates provides the CMM user with the ability to 'collect' points and supply them to any command that requires measured data, any number of times. This will significantly reduce inspection times by removing the necessity for repeat probings for the same point.

,store n
This is used to define the save location in which the co-ordinates are to be stored, there are 250 locations available.

Example: SP,1

3.4.2 Use Point (UP)

Format: UP,store n...[,store n]

The UP command is used to recall co-ordinates that have previously been saved by the

SP (Save Point) command.

,store n
This parameter is used to define the point(s) that are to recalled. When the UP command is entered (before one of the commands listed above, that command will use the points defined by [,store n] instead of displaying the usual prompt $.

Example: UP,1,2,3

3.4.3 Use Group (UG)

Format: UG,start point,points

The UG command is used to unsave groups of points. The points must have been saved previously.

,start point
This parameter is used to specify the save point location from which to start unsaving the group of points.

,points
This parameter is used to define the number of points to be unsaved.

Example: UG,5,6

3.5 Tolerancing Commands

3.5.1 Equal bi-lateral Tolerancing (/)

The / symbol allows nominal and tolerance data to be added to a measurement command where the tolerance value required is equal bi-lateral, that is, + or - the same value.

Example:
```
ID,Z/
-69,.25
-106,.025
10,.015
```

3.5.2 Unequal bi-lateral Command

The // symbol allows nominal and tolerance data to be added to a measurement command where the tolerance value required is unequal bi-lateral, that is, + or - different values.

Example:
```
OD/Z
```

```
-50,.025,-.025
-53,.025,-.025
50,.02,-.03
```

Appendix 4 : Examples of IGES, DMIS and CMES files

4.1 Drawing created in the CAD System

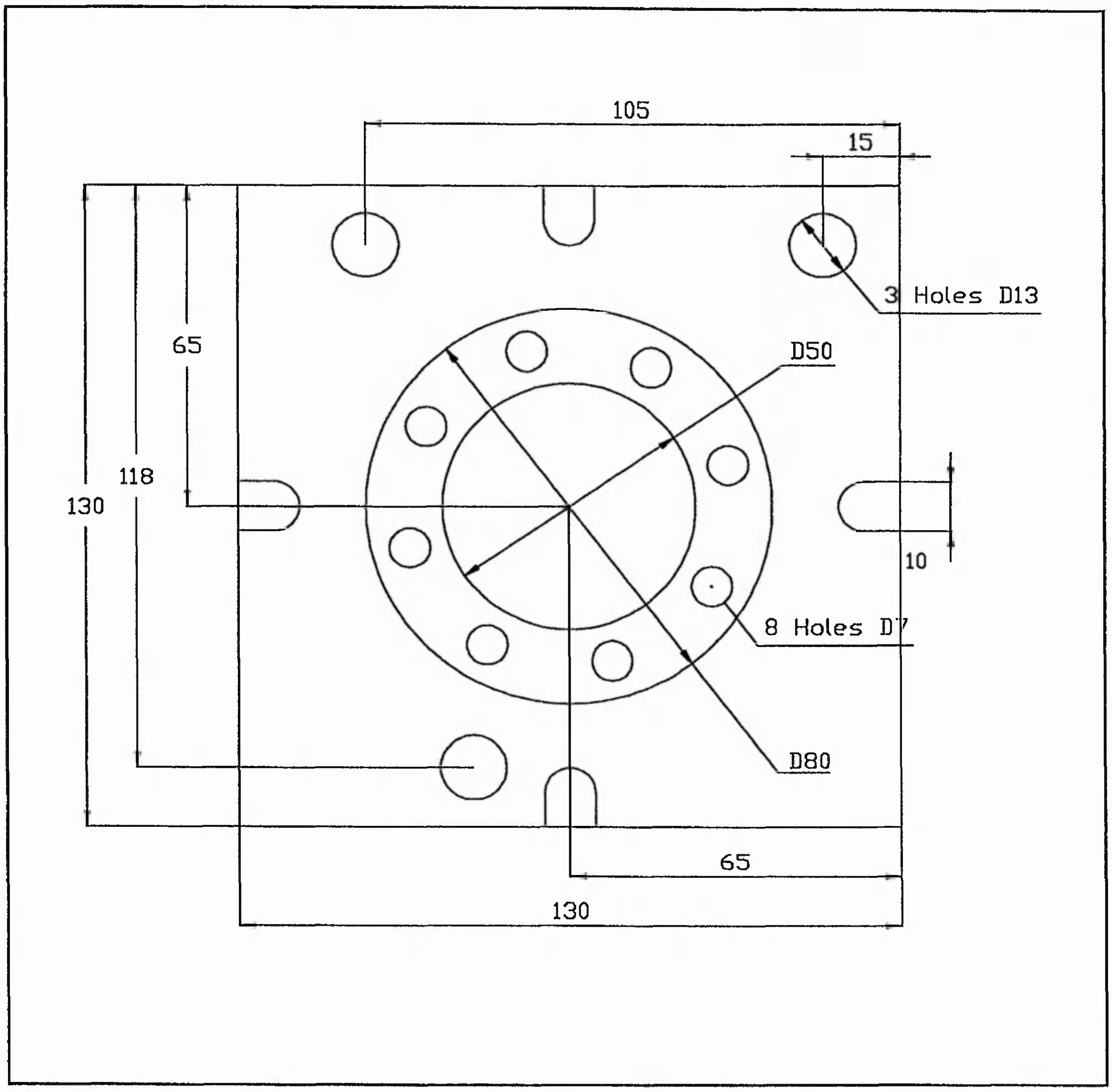


Figure 1 : Imported drawing

After creating this drawing in any CAD system an IGES file is generated by the CAD system preprocessor. This IGES file is read into a system to get the geometry information from the CAD drawing. Once the geometry is processed an inspection model will be created.
In the following pages is shown the IGES file generated for the drawing above:

4.2 IGES file input

```
IGES file generated from an AutoCAD drawing by the IGES                 S0000001
translator from Autodesk, Inc., translator version IGESOUT-3.04.        S0000002
,,15HA:\FACES\NFACEA,18HA:\FACES\OPEN1.IGS,27HAutoCAD-12_c1 InternationaG0000001
1,12HIGESOUT-3.04,32,38,6,99,15,15HA:\FACES\NFACEA,1.0,1,4HINCH,32767,  G0000002
3.2767D1,13H930525.113851,2.9785587507836D-7,2.9785587507836D2,10HPhil MG0000003
oore,23H Nottm Trent University,6,0;                                    G0000004
     110       1       1       1                                00000000D0000001
     110                       1                                        D0000002
     110       2       1       1                                00000000D0000003
     110                       1                                        D0000004
     110       3       1       1                                00000000D0000005
     110                       1                                        D0000006
     110       4       1       1                                00000000D0000007
     110                       1                                        D0000008
     100       5       1       1                       0        00000000D0000009
     100                       1                                        D0000010
     100       6       1       1                       0        00000000D0000011
     100                       1                                        D0000012
     100       7       1       1                       0        00000000D0000013
     100                       1                                        D0000014
     100       8       1       1                       0        00000000D0000015
     100                       1                                        D0000016
     100       9       1       1                       0        00000000D0000017
     100                       2                                        D0000018
     100      11       1       1                       0        00000000D0000019
     100                       2                                        D0000020
     100      13       1       1                       0        00000000D0000021
     100                       2                                        D0000022
     100      15       1       1                       0        00000000D0000023
     100                       2                                        D0000024
     100      17       1       1                       0        00000000D0000025
     100                       2                                        D0000026
     100      19       1       1                       0        00000000D0000027
     100                       2                                        D0000028
     100      21       1       1                       0        00000000D0000029
     100                       2                                        D0000030
     100      23       1       1                       0        00000000D0000031
     100                       2                                        D0000032
     100      25       1       1                       0        00000000D0000033
     100                       1                                        D0000034
     110      26       1       1                                00000000D0000035
     110                       1                                        D0000036
     110      27       1       1                                00000000D0000037
     110                       1                                        D0000038
     100      28       1       1                       0        00000000D0000039
     100                       1                                        D0000040
     100      29       1       1                       0        00000000D0000041
     100                       1                                        D0000042
     110      30       1       1                                00000000D0000043
     110                       1                                        D0000044
     110      31       1       1                                00000000D0000045
     110                       1                                        D0000046
     110      32       1       1                                00000000D0000047
     110                       1                                        D0000048
     110      33       1       1                                00000000D0000049
     110                       1                                        D0000050
     100      34       1       1                       0        00000000D0000051
     100                       1                                        D0000052
     100      35       1       1                       0        00000000D0000053
     100                       1                                        D0000054
     110      36       1       1                                00000000D0000055
     110                       1                                        D0000056
     110      37       1       1                                00000000D0000057
     110                       1                                        D0000058
     212      38       1       1                                00010100D0000059
     212             256       2                                        D0000060
     106      40       1                               0        00000101D0000061
     106                       2      21                                D0000062
     214      42       1       1                                00010100D0000063
     214             256       3       3                                D0000064
     206      45       1       1                       0        00000101D0000065
     206                       1                                        D0000066
     212      46       1       1                                00010100D0000067
     212             256       1                                        D0000068
     106      47       1                                        00010100D0000069
     106             256       1      40                                D0000070
```

```
106      48       1                                        00010100D0000071
106             256       1      40                                D0000072
214      49       1       1                                00010100D0000073
214             256       1       3                                D0000074
214      50       1       1                                00010100D0000075
214             256       1       3                                D0000076
216      51       1       1                       0        00000101D0000077
216                       1                                        D0000078
212      52       1       1                                00010100D0000079
212             256       1                                        D0000080
106      53       1                                        00010100D0000081
106             256       2      40                                D0000082
106      55       1                                        00010100D0000083
106             256       2      40                                D0000084
214      57       1       1                                00010100D0000085
214             256       2       3                                D0000086
214      59       1       1                                00010100D0000087
214             256       2       3                                D0000088
216      61       1       1                       0        00000101D0000089
216                       1                                        D0000090
212      62       1       1                                00010100D0000091
212             256       1                                        D0000092
106      63       1                                        00010100D0000093
106             256       2      40                                D0000094
106      65       1                                        00010100D0000095
106             256       2      40                                D0000096
214      67       1       1                                00010100D0000097
214             256       2       3                                D0000098
214      69       1       1                                00010100D0000099
214             256       2       3                                D0000100
216      71       1       1                       0        00000101D0000101
216                       1                                        D0000102
212      72       1       1                                00010100D0000103
212             256       1                                        D0000104
106      73       1                                        00010100D0000105
106             256       1      40                                D0000106
106      74       1                                        00010100D0000107
106             256       1      40                                D0000108
214      75       1       1                                00010100D0000109
214             256       1       3                                D0000110
214      76       1       1                                00010100D0000111
214             256       1       3                                D0000112
216      77       1       1                       0        00000101D0000113
216                       1                                        D0000114
212      78       1       1                                00010100D0000115
212             256       1                                        D0000116
106      79       1                                        00010100D0000117
106             256       1      40                                D0000118
106      80       1                                        00010100D0000119
106             256       1      40                                D0000120
214      81       1       1                                00010100D0000121
214             256       1       3                                D0000122
214      82       1       1                                00010100D0000123
214             256       1       3                                D0000124
216      83       1       1                       0        00000101D0000125
216                       1                                        D0000126
212      84       1       1                                00010100D0000127
212             256       1                                        D0000128
106      85       1                                        00010100D0000129
106             256       1      40                                D0000130
106      86       1                                        00010100D0000131
106             256       1      40                                D0000132
214      87       1       1                                00010100D0000133
214             256       1       3                                D0000134
214      88       1       1                                00010100D0000135
214             256       1       3                                D0000136
216      89       1       1                       0        00000101D0000137
216                       1                                        D0000138
110      90       1       1                                00000000D0000139
110                       1                                        D0000140
110      91       1       1                                00000000D0000141
110                       1                                        D0000142
110      92       1       1                                00000000D0000143
110                       1                                        D0000144
110      93       1       1                                00000000D0000145
110                       1                                        D0000146
212      94       1       1                                00010100D0000147
212             256       2                                        D0000148
106      96       1                               0        00000101D0000149
106                       2      21                                D0000150
```

```
     214      98       1       1                                00010100D0000151
     214             256       2       3                                D0000152
     206     100       1       1                       0        00000101D0000153
     206                       1                                        D0000154
     212     101       1       1                                00010100D0000155
     212             256       2                                        D0000156
     106     103       1                               0        00000101D0000157
     106                       2      21                                D0000158
     214     105       1       1                                00010100D0000159
     214             256       3       3                                D0000160
     206     108       1       1                       0        00000101D0000161
     206                       1                                        D0000162
     212     109       1       1                                00010100D0000163
     212             256       2                                        D0000164
     106     111       1                               0        00000101D0000165
     106                       2      21                                D0000166
     214     113       1       1                                00010100D0000167
     214             256       3       3                                D0000168
     206     116       1       1                       0        00000101D0000169
     206                       1                                        D0000170
     212     117       1       1                                00010100D0000171
     212             256       1                                        D0000172
     106     118       1                                        00010100D0000173
     106             256       1      40                                D0000174
     106     119       1                                        00010100D0000175
     106             256       1      40                                D0000176
     214     120       1       1                                00010100D0000177
     214             256       1       3                                D0000178
     214     121       1       1                                00010100D0000179
     214             256       1       3                                D0000180
     216     122       1       1                       0        00000101D0000181
     216                       1                                        D0000182
     212     123       1       1                                00010100D0000183
     212             256       1                                        D0000184
     106     124       1                                        00010100D0000185
     106             256       1      40                                D0000186
     106     125       1                                        00010100D0000187
     106             256       1      40                                D0000188
     214     126       1       1                                00010100D0000189
     214             256       1       3                                D0000190
     214     127       1       1                                00010100D0000191
     214             256       1       3                                D0000192
     216     128       1       1                       0        00000101D0000193
     216                       1                                        D0000194
110,260.0,1.0D2,0.0,260.0,230.0,0.0;                                   1P0000001
110,260.0,230.0,0.0,1.3D2,230.0,0.0;                                   3P0000002
110,1.3D2,230.0,0.0,130.0,100.0,0.0;                                   5P0000003
110,130.0,100.0,0.0,260.0,1.0D2,0.0;                                   7P0000004
100,0.0,245.0,218.0,2.515D2,218.0,2.515D2,218.0;                       9P0000005
100,0.0,155.0,218.0,1.615D2,218.0,1.615D2,218.0;                      11P0000006
100,-5.0,195.0,165.0,220.0,165.0,220.0,165.0;                         13P0000007
100,0.0,195.0,165.0,235.0,165.0,235.0,165.0;                          15P0000008
100,-10.0,1.8658838103417D2,1.9639258935439D2,1.9058838103417D2,      17P0000009
1.9639258935439D2,1.9058838103417D2,1.9639258935439D2;                17P0000010
100,-10.0,1.66854174377D2,1.8125D2,1.70854174377D2,1.8125D2,          19P0000011
1.70854174377D2,1.8125D2;                                             19P0000012
100,-10.0,1.6360741064561D2,1.5658838103417D2,1.6760741064561D2,      21P0000013
1.5658838103417D2,1.6760741064561D2,1.5658838103417D2;                21P0000014
100,-10.0,1.7875D2,1.3685417437701D2,1.8275D2,1.3685417437701D2,      23P0000015
1.8275D2,1.3685417437701D2;                                           23P0000016
100,-10.0,2.0341161896583D2,1.3360741064561D2,2.0741161896583D2,      25P0000017
1.3360741064561D2,2.0741161896583D2,1.3360741064561D2;                25P0000018
100,-10.0,2.23145825623D2,1.4875D2,2.27145825623D2,1.4875D2,          27P0000019
2.27145825623D2,1.4875D2;                                             27P0000020
100,-10.0,2.2639258935439D2,1.7341161896583D2,2.3039258935439D2,      29P0000021
1.7341161896583D2,2.3039258935439D2,1.7341161896583D2;                29P0000022
100,-10.0,2.1125D2,1.93145825623D2,2.1525D2,1.93145825623D2,          31P0000023
2.1525D2,1.93145825623D2;                                             31P0000024
100,0.0,176.0,112.0,1.825D2,112.0,1.825D2,112.0;                      33P0000025
110,130.0,1.7D2,0.0,137.0,170.0,0.0;                                  35P0000026
110,130.0,1.6D2,0.0,137.0,160.0,0.0;                                  37P0000027
100,0.0,1.37D2,165.0,1.37D2,160.0,1.37D2,170.0;                       39P0000028
100,0.0,195.0,107.0,200.0,107.0,190.0,107.0;                          41P0000029
110,200.0,100.0,0.0,200.0,107.0,0.0;                                  43P0000030
110,190.0,100.0,0.0,190.0,107.0,0.0;                                  45P0000031
110,2.0D2,230.0,0.0,2.0D2,223.0,0.0;                                  47P0000032
110,1.9D2,230.0,0.0,1.9D2,223.0,0.0;                                  49P0000033
100,0.0,195.0,223.0,190.0,223.0,200.0,223.0;                          51P0000034
100,0.0,2.53D2,165.0,2.53D2,170.0,2.53D2,160.0;                       53P0000035
110,260.0,160.0,0.0,253.0,160.0,0.0;                                  55P0000036
```

```
110,260.0,170.0,0.0,253.0,170.0,0.0;                                    57P0000037
212,1,10,2.75D1,3.0,1,,0.0,0,0,2.337873312091D2,1.38993280632D2,        59P0000038
0.0,10H8 Holes D7;                                                      59P0000039
106,1,4,0.0,2.23055825623D2,1.4875D2,2.23235825623D2,1.4875D2,          61P0000040
2.23145825623D2,1.4866D2,2.23145825623D2,1.4884D2;                      61P0000041
214,2,3.0,1.0,0.0,2.2540242785743D2,1.454473122528D2,                   63P0000042
2.287873312091D2,1.40493280632D2,2.317873312091D2,                      63P0000043
1.40493280632D2;                                                        63P0000044
206,59,63,0,2.23145825623D2,1.4875D2;                                   65P0000045
212,1,2,3.5,3.0,1,,0.0,0,0,2.615D2,1.525D2,0.0,2H10;                    67P0000046
106,1,3,0.0,260.0,170.0,2.600625D2,170.0,2.7018D2,170.0;                69P0000047
106,1,3,0.0,260.0,160.0,2.600625D2,160.0,2.7018D2,160.0;                71P0000048
214,1,3.0,1.0,0.0,270.0,170.0,270.0,176.0;                              73P0000049
214,2,3.0,1.0,0.0,270.0,160.0,270.0,154.0,267.0,154.0;                  75P0000050
216,67,73,75,69,71;                                                     77P0000051
212,1,2,4.0,3.0,1,,0.0,0,0,2.505D2,2.3765437419281D2,0.0,2H15;          79P0000052
106,1,3,0.0,260.0,230.0,260.0,2.300625D2,260.0,                         81P0000053
2.3633437419281D2;                                                      81P0000054
106,1,3,0.0,245.0,218.0,245.0,2.180625D2,245.0,                         83P0000055
2.3633437419281D2;                                                      83P0000056
214,1,3.0,1.0,0.0,260.0,2.3615437419281D2,2.525D2,                      85P0000057
2.3615437419281D2;                                                      85P0000058
214,1,3.0,1.0,0.0,245.0,2.3615437419281D2,2.525D2,                      87P0000059
2.3615437419281D2;                                                      87P0000060
216,79,85,87,81,83;                                                     89P0000061
212,1,3,6.5,3.0,1,,0.0,0,0,2.0425D2,2.4417884290413D2,0.0,3H105;        91P0000062
106,1,3,0.0,260.0,230.0,260.0,2.300625D2,260.0,                         93P0000063
2.4285884290413D2;                                                      93P0000064
106,1,3,0.0,155.0,218.0,155.0,2.180625D2,155.0,                         95P0000065
2.4285884290413D2;                                                      95P0000066
214,1,3.0,1.0,0.0,260.0,2.4267884290413D2,2.075D2,                      97P0000067
2.4267884290413D2;                                                      97P0000068
214,1,3.0,1.0,0.0,155.0,2.4267884290413D2,2.075D2,                      99P0000069
2.4267884290413D2;                                                      99P0000070
216,91,97,99,93,95;                                                    101P0000071
212,1,2,5.0,3.0,1,,0.0,0,0,1.175D2,196.0,0.0,2H65;                     103P0000072
106,1,3,0.0,130.0,230.0,1.299375D2,230.0,1.1982D2,230.0;               105P0000073
106,1,3,0.0,195.0,165.0,1.949375D2,165.0,1.1982D2,165.0;               107P0000074
214,1,3.0,1.0,0.0,120.0,230.0,120.0,201.0;                             109P0000075
214,1,3.0,1.0,0.0,120.0,165.0,120.0,194.0;                             111P0000076
216,103,109,111,105,107;                                               113P0000077
212,1,3,6.0,3.0,1,,0.0,0,0,107.0,1.695D2,0.0,3H118;                    115P0000078
106,1,3,0.0,130.0,230.0,1.299375D2,230.0,1.0982D2,230.0;               117P0000079
106,1,3,0.0,176.0,112.0,1.759375D2,112.0,1.0982D2,112.0;               119P0000080
214,1,3.0,1.0,0.0,110.0,230.0,110.0,1.745D2;                           121P0000081
214,1,3.0,1.0,0.0,110.0,112.0,110.0,1.675D2;                           123P0000082
216,115,121,123,117,119;                                               125P0000083
212,1,3,6.5,3.0,1,,0.0,0,0,9.675D1,1.635D2,0.0,3H130;                  127P0000084
106,1,3,0.0,130.0,230.0,1.299375D2,230.0,9.982D1,230.0;                129P0000085
106,1,3,0.0,130.0,100.0,1.299375D2,100.0,9.982D1,100.0;                131P0000086
214,1,3.0,1.0,0.0,100.0,230.0,100.0,1.685D2;                           133P0000087
214,1,3.0,1.0,0.0,100.0,100.0,100.0,1.615D2;                           135P0000088
216,127,133,135,129,131;                                               137P0000089
110,1.9491D2,165.0,0.0,1.9509D2,165.0,0.0;                             139P0000090
110,195.0,1.6491D2,0.0,195.0,1.6509D2,0.0;                             141P0000091
110,1.9491D2,165.0,0.0,1.9509D2,165.0,0.0;                             143P0000092
110,195.0,1.6491D2,0.0,195.0,1.6509D2,0.0;                             145P0000093
212,1,3,7.5,3.0,1,,0.0,0,0,2.3846736428779D2,1.1197869405793D2,        147P0000094
0.0,3HD80;                                                             147P0000095
106,1,4,0.0,1.9491D2,1.65D2,1.9509D2,1.65D2,1.95D2,1.6491D2,           149P0000096
1.95D2,1.6509D2;                                                       149P0000097
214,1,3.0,1.0,0.0,2.19214867371D2,1.3316228340152D2,                   151P0000098
2.3646736428779D2,1.1047869405793D2;                                   151P0000099
206,147,151,0,1.95D2,1.65D2;                                           153P0000100
212,1,3,7.5,3.0,1,,0.0,0,0,2.390057927396D2,1.9511806342625D2,         155P0000101
0.0,3HD50;                                                             155P0000102
106,1,4,0.0,1.9491D2,1.65D2,1.9509D2,1.65D2,1.95D2,1.6491D2,           157P0000103
1.95D2,1.6509D2;                                                       157P0000104
214,2,3.0,1.0,0.0,2.1566077589571D2,1.7907594896933D2,                 159P0000105
2.370057927396D2,1.9361806342625D2,2.465057927396D2,                   159P0000106
1.9361806342625D2;                                                     159P0000107
206,155,159,0,1.95D2,1.65D2;                                           161P0000108
212,1,11,2.95D1,3.0,1,,0.0,0,0,2.5737662688661D2,                      163P0000109
2.0641245071853D2,0.0,11H3 Holes D13;                                  163P0000110
106,1,4,0.0,2.4491D2,218.0,2.4509D2,218.0,245.0,2.1791D2,245.0,        165P0000111
2.1809D2;                                                              165P0000112
214,2,3.0,1.0,0.0,2.4903831218238D2,2.1290666762153D2,                 167P0000113
2.5537662688661D2,2.0491245071853D2,2.8687662688661D2,                 167P0000114
2.0491245071853D2;                                                     167P0000115
206,163,167,0,245.0,218.0;                                             169P0000116
```

```
212,1,2,5.0,3.0,1,,0.0,0,0,225.0,9.15D1,0.0,2H65;                      171P0000117
106,1,3,0.0,260.0,100.0,260.0,9.99375D1,260.0,8.982D1;                 173P0000118
106,1,3,0.0,195.0,165.0,195.0,1.649375D2,195.0,8.982D1;                175P0000119
214,1,3.0,1.0,0.0,260.0,90.0,2.275D2,90.0;                             177P0000120
214,1,3.0,1.0,0.0,195.0,90.0,2.275D2,90.0;                             179P0000121
216,171,177,179,173,175;                                               181P0000122
212,1,3,6.5,3.0,1,,0.0,0,0,1.9175D2,8.15D1,0.0,3H130;                  183P0000123
106,1,3,0.0,260.0,100.0,260.0,9.99375D1,260.0,7.982D1;                 185P0000124
106,1,3,0.0,130.0,100.0,130.0,9.99375D1,130.0,7.982D1;                 187P0000125
214,1,3.0,1.0,0.0,260.0,80.0,195.0,80.0;                               189P0000126
214,1,3.0,1.0,0.0,130.0,80.0,195.0,80.0;                               191P0000127
216,183,189,191,185,187;                                               193P0000128
S0000002G0000004D0000194P0000128                                          T0000001
```

4.3 Creating the inspection model

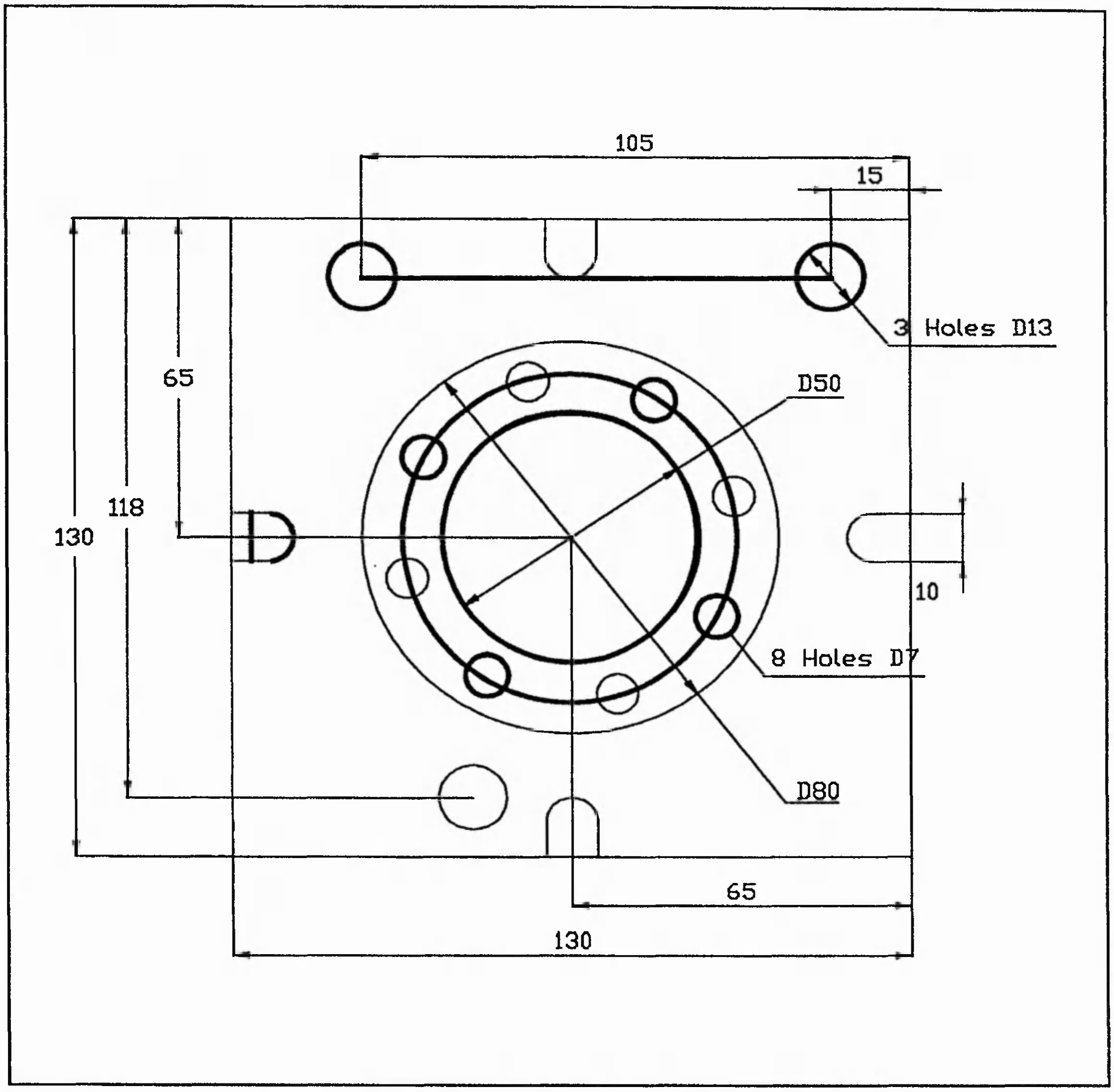


Figure 2 : Creating the inspection model

After the inspection model has been created, the inspection program to drive the CMM will be generated in two different languages. One of the languages is CMES, which is a specific language to drive LK machines; the other language is DMIS which is an ANSI standard.

An example of the DMIS and CMES files (prototype and not prototype) for this particular inspection model will be shown in the following pages.

4.4 CMES file output (not prototype)

```
UR,1,PH
#MA,1,2
-101.500
-12.000
#MA,3
-3.000
#PT,1,2
-98.500
-12.000
SP,1
#MA,1,2
-105.000
-8.500
#PT,1,2
-105.000
-5.500
SP,2
#MA,1,2
-108.500
-12.000
#PT,1,2
-111.500
-12.000
SP,3
#MA,1,2
-105.000
-15.500
#PT,1,2
-105.000
-18.500
SP,4
UP,1,2,3,4
ID,3
SP,5
#MA,3
5.000
#MA,1,2
-18.500
-12.000
#MA,3
-3.000
#PT,1,2
-21.500
-12.000
SP,1
#MA,1,2
-15.000
-15.500
#PT,1,2
-15.000
-18.500
SP,2
#MA,1,2
-11.500
-12.000
#PT,1,2
-8.500
-12.000
SP,3
#MA,1,2
-15.000
-8.500
#PT,1,2
-15.000
-5.500
SP,4
UP,1,2,3,4
ID,3
SP,6
UP,5,6
LT,1,2//
90.00,0.50,-0.40
#MA,3
5.000
#MA,1,2
-126.500
-63.000
#MA,3
-3.000
#PT,1,2
-126.500
-60.000
SP,1
#MA,1,2
-126.500
-67.000
#PT,1,2
-126.500
-70.000
SP,2
UP,1,2
LI,2//
10.00,0.20,-0.10
#MA,1,2
-122.000
-66.732
#MA,3
-3.000
#PT,1,2
-120.500
-69.330
SP,1
#MA,1,2
-122.000
-66.732
#MA,1,2
-121.268
-66.000
#PT,1,2
-118.670
-67.500
SP,1
#MA,1,2
-121.268
-66.000
#MA,1,2
-121.000
-65.000
#PT,1,2
-118.000
-65.000
SP,1
#MA,1,2
-121.000
-65.000
#MA,1,2
-121.268
-64.000
#PT,1,2
-118.670
-62.500
SP,1
#MA,1,2
-121.268
-64.000
#MA,1,2
-122.000
-63.268
#PT,1,2
-120.500
-60.670
SP,1
#MA,1,2
-122.000
-63.268
UP,1,2,3,4
ID,3//
-123.00,0.60,-0.30
-65.00,0.00,0.00
10.00,0.40,-0.30
```

```
#MA,3
5.000
#MA,1,2
-97.393
-73.412
#MA,3
-13.000
#PT,1,2
-100.393
-73.412
SP,1
#MA,1,2
-96.393
-74.412
#PT,1,2
-96.393
-77.412
SP,2
#MA,1,2
-95.393
-73.412
#PT,1,2
-92.393
-73.412
SP,3
#MA,1,2
-96.393
-72.412
#PT,1,2
-96.393
-69.412
SP,4
UP,1,2,3,4
ID,3
SP,5
#MA,3
0.000
#MA,1,2
-57.588
-96.393
#MA,3
-13.000
#PT,1,2
-60.588
-96.393
SP,1
#MA,1,2
-56.588
-97.393
#PT,1,2
-56.588
-100.393
SP,2
#MA,1,2
-55.588
-96.393
#PT,1,2
-52.588
-96.393
SP,3
#MA,1,2
-56.588
-95.393
#PT,1,2
-56.588
-92.393
SP,4
UP,1,2,3,4
ID,3
SP,6
#MA,3
0.000
#MA,1,2
-33.607
-57.588
#MA,3
-13.000
#PT,1,2
-33.607
-60.588
SP,1
#MA,1,2
-32.607
-56.588
#PT,1,2
-29.607
-56.588
SP,2
#MA,1,2
-33.607
-55.588
#PT,1,2
-33.607
-52.588
SP,3
#MA,1,2
-34.607
-56.588
#PT,1,2
-37.607
-56.588
SP,4
UP,1,2,3,4
ID,3
SP,7
#MA,3
0.000
#MA,1,2
-72.412
-33.607
#MA,3
-13.000
#PT,1,2
-69.412
-33.607
SP,1
#MA,1,2
-73.412
-32.607
#PT,1,2
-73.412
-29.607
SP,2
#MA,1,2
-74.412
-33.607
#PT,1,2
-77.412
-33.607
SP,3
#MA,1,2
-73.412
-34.607
#PT,1,2
-73.412
-37.607
SP,4
UP,1,2,3,4
ID,3
SP,8
UP,5,6,7,8
PC,3,4//
-65.00,0.60,-0.50
-65.00,0.00,0.00
65.00,0.50,-0.40
#MA,3
-5.000
#MA,1,2
-65.000
-37.000
#MA,3
-8.000
#PT,1,2
-65.000
-40.000
SP,1
#MA,3
0.000
#MA,1,2
```

```
-93.000
-65.000
#MA,3
-8.000
#PT,1,2
-90.000
-65.000
SP,2
#MA,3
0.000
#MA,1,2
-65.000
-93.000
#MA,3
-8.000
#PT,1,2
-65.000
-90.000
SP,3
#MA,3
0.000
#MA,1,2
-37.000
-65.000
#MA,3
-8.000
#PT,1,2
-40.000
-65.000
SP,4
UP,1,2,3,4
OD,3//
-65.00,0.40,-0.30
-65.00,0.50,-0.40
50.00,0.30,-0.50
#MA,3
10.000
ET
```

4.5 CMES file output (prototype)

```
UR,1,PH

:MOVE_NEXT_1
#MA,1,2
-11.500
-12.000
:ASK_1
IP,'Is this a hole?',F1
IF,F1,ASK_1,NO_MEAS_1,
MEAS_1
:MEAS_1
#MA,3
-3.000
#PT,1,2
-8.500
-12.000
SP,1
#MA,1,2
-15.000
-8.500
#PT,1,2
-15.000
-5.500
SP,2
#MA,1,2
-18.500
-12.000
#PT,1,2
-21.500
-12.000
SP,3
#MA,1,2
-15.000
-15.500
#PT,1,2
-15.000
-18.500
SP,4
UP,1,2,3,4
ID,3
GO,MOVE_NEXT_2
:NO_MEAS_1
#MA,1,2
-15.000
-15.500
SP,5
:MOVE_NEXT_2
#MA,3
5.000
#MA,1,2
-101.500
-12.000
:ASK_2
IP,'Is this a hole?',F1
IF,F1,ASK_2,NO_MEAS_2,
MEAS_2
:MEAS_2
#MA,3
-3.000
#PT,1,2
-98.500
-12.000
SP,1
#MA,1,2
-105.000
-8.500
#PT,1,2
-105.000
-5.500
SP,2
#MA,1,2
-108.500
-12.000
#PT,1,2
-111.500
-12.000
SP,3
#MA,1,2
-105.000
-15.500
#PT,1,2
-105.000
-18.500
SP,4
UP,1,2,3,4
ID,3
GO,ASK_3
:NO_MEAS_2
#MA,1,2
-105.000
-15.500
:ASK_3
IP,'distance between two
features?',F1
IF,F1,ASK_3,MOVE_NEX
T_4,MEAS_3
:MEAS_3
SP,6
UP,5,6
LT,1,2
:MOVE_NEXT_4
#MA,3
5.000
#MA,1,2
-126.500
-63.000
:ASK_4
IP,'Is this a slot?',F1
IF,F1,ASK_4,NO_MEAS_4,
MEAS_4
:MEAS_4
#MA,3
-3.000
#PT,1,2
-126.500
-60.000
SP,1
#MA,1,2
-126.500
-67.000
#PT,1,2
-126.500
-70.000
SP,2
UP,1,2
LI,2
GO,MOVE_NEXT_5
:NO_MEAS_4
#MA,1,2
-126.500
-67.000
:MOVE_NEXT_5
#MA,1,2
-122.000
-66.732
:ASK_5
IP,'Is this a partial hole?',F1
IF,F1,ASK_5,NO_MEAS_5,
MEAS_5
:MEAS_5
#MA,3
-3.000
#PT,1,2
-120.500
-69.330
SP,1
#MA,1,2
-122.000
-66.732
#MA,1,2
```

```
-121.268
-66.000
#PT,1,2
-118.670
-67.500
SP,1
#MA,1,2
-121.268
-66.000
#MA,1,2
-121.000
-65.000
#PT,1,2
-118.000
-65.000
SP,1
#MA,1,2
-121.000
-65.000
#MA,1,2
-121.268
-64.000
#PT,1,2
-118.670
-62.500
SP,1
#MA,1,2
-121.268
-64.000
#MA,1,2
-122.000
-63.268
#PT,1,2
-120.500
-60.670
SP,1
#MA,1,2
-122.000
-63.268
UP,1,2,3,4
ID,3
GO,MOVE_NEXT_6
:NO_MEAS_5
#MA,1,2
-122.000
-63.268
:MOVE_NEXT_6
#MA,3
5.000
#MA,1,2
-97.393
-73.412
:ASK_6
IP,'Is this a hole?',F1
IF,F1,ASK_6,NO_MEAS_6,
MEAS_6
:MEAS_6
#MA,3
-13.000
#PT,1,2
-100.393
-73.412
SP,1
#MA,1,2
-96.393
-74.412
#PT,1,2
-96.393
-77.412
SP,2
#MA,1,2
-95.393
-73.412
#PT,1,2
-92.393
-73.412
SP,3
#MA,1,2
-96.393
-72.412
#PT,1,2
-96.393
-69.412
SP,4
UP,1,2,3,4
ID,3
GO,MOVE_NEXT_7
:NO_MEAS_6
#MA,1,2
-96.393
-72.412
SP,5
:MOVE_NEXT_7
#MA,3
0.000
#MA,1,2
-57.588
-96.393
:ASK_7
IP,'Is this a hole?',F1
IF,F1,ASK_7,NO_MEAS_7,
MEAS_7
:MEAS_7
#MA,3
-13.000
#PT,1,2
-60.588
-96.393
SP,1
#MA,1,2
-56.588
-97.393
#PT,1,2
-56.588
-100.393
SP,2
#MA,1,2
-55.588
-96.393
#PT,1,2
-52.588
-96.393
SP,3
#MA,1,2
-56.588
-95.393
#PT,1,2
-56.588
-92.393
SP,4
UP,1,2,3,4
ID,3
GO,MOVE_NEXT_8
:NO_MEAS_7
#MA,1,2
-56.588
-95.393
SP,6
:MOVE_NEXT_8
#MA,3
0.000
#MA,1,2
-33.607
-57.588
:ASK_8
IP,'Is this a hole?',F1
IF,F1,ASK_8,NO_MEAS_8,
MEAS_8
:MEAS_8
#MA,3
-13.000
#PT,1,2
-33.607
-60.588
SP,1
#MA,1,2
-32.607
-56.588
#PT,1,2
```

```
-29.607
-56.588
SP,2
#MA,1,2
-33.607
-55.588
#PT,1,2
-33.607
-52.588
SP,3
#MA,1,2
-34.607
-56.588
#PT,1,2
-37.607
-56.588
SP,4
UP,1,2,3,4
ID,3
GO,MOVE_NEXT_9
:NO_MEAS_8
#MA,1,2
-34.607
-56.588
SP,7
:MOVE_NEXT_9
#MA,3
0.000
#MA,1,2
-72.412
-33.607
:ASK_9
IP,'Is this a hole?',F1
IF,F1,ASK_9,NO_MEAS_9,
MEAS_9
:MEAS_9
#MA,3
-13.000
#PT,1,2
-69.412
-33.607
SP,1
#MA,1,2
-73.412
-32.607
#PT,1,2
-73.412
-29.607
SP,2
#MA,1,2
-74.412
-33.607
#PT,1,2
-77.412
-33.607
SP,3
#MA,1,2
-73.412
-34.607
#PT,1,2
-73.412
-37.607
SP,4
UP,1,2,3,4
ID,3
GO,ASK_10
:NO_MEAS_9
#MA,1,2
-73.412
-34.607
SP,8
:ASK_10
IP,'Pitch Circle?',F1
IF,F1,ASK_10,MOVE_NEX
T_11,MEAS_10
:MEAS_10
UP,5,6,7,8
PC,3,4
:MOVE_NEXT_11
#MA,3
-5.000
#MA,1,2
-65.000
-37.000
:ASK_11
IP,'Is this a cylinder?',F1
IF,F1,ASK_11,END,MEAS
_11
:MEAS_11
#MA,3
-8.000
#PT,1,2
-65.000
-40.000
SP,1
#MA,3
0.000
#MA,1,2
-93.000
-65.000
#MA,3
-8.000
#PT,1,2
-90.000
-65.000
SP,2
#MA,3
0.000
#MA,1,2
-65.000
-93.000
#MA,3
-8.000
#PT,1,2
-65.000
-90.000
SP,3
#MA,3
0.000
#MA,1,2
-37.000
-65.000
#MA,3
-8.000
#PT,1,2
-40.000
-65.000
SP,4
UP,1,2,3,4
OD,3
:NO_MEAS_11
#MA,1,2
-37.000
-65.000
:END
ET
```

4.6 DMIS file output (not prototype)

```
SNSLCT/S(PROBE1)
WKPLAN/XYPLAN

F(Hole1)=FEAT/CIRCLE,INNER,CART,-105.000,-12.000,0.000,0,0,1,13.000
MEAS/CIRCLE,F(Hole1),4
GOTO/-101.500,-12.000,10.000
GOTO/-101.500,-12.000,-3.000
PTMEAS/CART,-98.500,-12.000,-3.000,-1,0,0
GOTO/-105.000,-8.500,-3.000
PTMEAS/CART,-105.000,-5.500,-3.000,0,-1,0
GOTO/-108.500,-12.000,-3.000
PTMEAS/CART,-111.500,-12.000,-3.000,1,0,0
GOTO/-105.000,-15.500,-3.000
PTMEAS/CART,-105.000,-18.500,-3.000,0,1,0
ENDMES

TEXT/OUTFIL,' '
TEXT/OUTFIL,'****** Measured Hole1 *****'
OUTPUT/FA(Hole1)

F(Hole2)=FEAT/CIRCLE,INNER,CART,-15.000,-12.000,0.000,0,0,1,13.000
MEAS/CIRCLE,F(Hole2),4
GOTO/-105.000,-15.500,5.000
GOTO/-18.500,-12.000,5.000
GOTO/-18.500,-12.000,-3.000
PTMEAS/CART,-21.500,-12.000,-3.000,1,0,0
GOTO/-15.000,-15.500,-3.000
PTMEAS/CART,-15.000,-18.500,-3.000,0,1,0
GOTO/-11.500,-12.000,-3.000
PTMEAS/CART,-8.500,-12.000,-3.000,-1,0,0
GOTO/-15.000,-8.500,-3.000
PTMEAS/CART,-15.000,-5.500,-3.000,0,-1,0
ENDMES

TEXT/OUTFIL,' '
TEXT/OUTFIL,'****** Measured Hole2 *****'
OUTPUT/FA(Hole2)

F(Dist2F1)=FEAT/LINE,BND,CART,-105.000,-12.000,0.000,-15.000,-12.000,0.000,0,0,1

CONST/LINE,F(Dist2F1),BF,FA(Hole1),FA(Hole2)
TEXT/OUTFIL,' '
TEXT/OUTFIL,'****** Measured Dist2F1 *****'
OUTPUT/FA(Dist2F1)

T(TolDist2F1)=TOL/DISTB,NOMINL,90.000,-0.400,0.500,PT2PT
EVAL/FA(Hole1),FA(Hole2),TA(TolDist2F1)
OUTPUT/TA(TolDist2F1)

F(LenIn1)=FEAT/LINE,BND,CART,-126.500,-60.000,0.000,-126.500,-70.000,0.000,0,0,1
```

```
GOTO/-15.000,-8.500,5.000
GOTO/-126.500,-63.000,5.000
GOTO/-126.500,-63.000,-3.000
F(Point1)=FEAT/POINT,CART,-126.500,-60.000,-3.000,0,-1,0
MEAS/POINT,F(Point1),1
PTMEAS/CART,-126.500,-60.000,-3.000,0,-1,0
ENDMES

GOTO/-126.500,-67.000,-3.000
GOTO/-126.500,-67.000,-3.000
F(Point2)=FEAT/POINT,CART,-126.500,-70.000,-3.000,0,1,0
MEAS/POINT,F(Point2),1
PTMEAS/CART,-126.500,-70.000,-3.000,0,1,0
ENDMES

CONST/LINE,F(LenIn1),BF,FA(Point1),FA(Point2)
TEXT/OUTFIL,' '
TEXT/OUTFIL,'****** Measured LenIn1 *****'
OUTPUT/FA(LenIn1)

T(TolLenIn1)=TOL/DISTB,NOMINL,10.000,-0.100,0.200,YAXIS
EVAL/FA(Point1),FA(Point2),TA(TolLenIn1)
OUTPUT/TA(TolLenIn1)

F(ArcInt1)=FEAT/ARC,INNER,CART,-123.000,-65.000,0.000,0,0,1,5.000,-1.571,1.571
MEAS/ARC,F(ArcInt1),5
GOTO/-122.000,-66.732,-3.000
GOTO/-122.000,-66.732,-3.000
PTMEAS/CART,-120.500,-69.330,-3.000,-1,-1,0
GOTO/-122.000,-66.732,-3.000
GOTO/-121.268,-66.000,-3.000
PTMEAS/CART,-118.670,-67.500,-3.000,-1,-1,0
GOTO/-121.268,-66.000,-3.000
GOTO/-121.000,-65.000,-3.000
PTMEAS/CART,-118.000,-65.000,-3.000,-1,-1,0
GOTO/-121.000,-65.000,-3.000
GOTO/-121.268,-64.000,-3.000
PTMEAS/CART,-118.670,-62.500,-3.000,-1,-1,0
GOTO/-121.268,-64.000,-3.000
GOTO/-122.000,-63.268,-3.000
PTMEAS/CART,-120.500,-60.670,-3.000,-1,-1,0
GOTO/-122.000,-63.268,-3.000
ENDMES

TEXT/OUTFIL,' '
TEXT/OUTFIL,'****** Measured ArcInt1 *****'
OUTPUT/FA(ArcInt1)

T(TxArcInt1)=TOL/CORTOL,XAXIS,-0.300,0.600
OUTPUT/FA(ArcInt1),TA(TxArcInt1)
T(TdArcInt1)=TOL/DIAM,-0.300,0.400
OUTPUT/FA(ArcInt1),TA(TdArcInt1)
```

```
F(Hole3)=FEAT/CIRCLE,INNER,CART,-96.393,-73.412,-10.000,0,0,1,8.000
MEAS/CIRCLE,F(Hole3),4
GOTO/-122.000,-63.268,5.000
GOTO/-97.393,-73.412,5.000
GOTO/-97.393,-73.412,-13.000
PTMEAS/CART,-100.393,-73.412,-13.000,1,0,0
GOTO/-96.393,-74.412,-13.000
PTMEAS/CART,-96.393,-77.412,-13.000,0,1,0
GOTO/-95.393,-73.412,-13.000
PTMEAS/CART,-92.393,-73.412,-13.000,-1,0,0
GOTO/-96.393,-72.412,-13.000
PTMEAS/CART,-96.393,-69.412,-13.000,0,-1,0
ENDMES

TEXT/OUTFIL,' '
TEXT/OUTFIL,'****** Measured Hole3 *****'
OUTPUT/FA(Hole3)

F(Hole4)=FEAT/CIRCLE,INNER,CART,-56.588,-96.393,-10.000,0,0,1,8.000
MEAS/CIRCLE,F(Hole4),4
GOTO/-96.393,-72.412,0.000
GOTO/-57.588,-96.393,0.000
GOTO/-57.588,-96.393,-13.000
PTMEAS/CART,-60.588,-96.393,-13.000,1,0,0
GOTO/-56.588,-97.393,-13.000
PTMEAS/CART,-56.588,-100.393,-13.000,0,1,0
GOTO/-55.588,-96.393,-13.000
PTMEAS/CART,-52.588,-96.393,-13.000,-1,0,0
GOTO/-56.588,-95.393,-13.000
PTMEAS/CART,-56.588,-92.393,-13.000,0,-1,0
ENDMES

TEXT/OUTFIL,' '
TEXT/OUTFIL,'****** Measured Hole4 *****'
OUTPUT/FA(Hole4)

F(Hole5)=FEAT/CIRCLE,INNER,CART,-33.607,-56.588,-10.000,0,0,1,8.000
MEAS/CIRCLE,F(Hole5),4
GOTO/-56.588,-95.393,0.000
GOTO/-33.607,-57.588,0.000
GOTO/-33.607,-57.588,-13.000
PTMEAS/CART,-33.607,-60.588,-13.000,0,1,0
GOTO/-32.607,-56.588,-13.000
PTMEAS/CART,-29.607,-56.588,-13.000,-1,0,0
GOTO/-33.607,-55.588,-13.000
PTMEAS/CART,-33.607,-52.588,-13.000,0,-1,0
GOTO/-34.607,-56.588,-13.000
PTMEAS/CART,-37.607,-56.588,-13.000,1,0,0
ENDMES

TEXT/OUTFIL,' '
TEXT/OUTFIL,'****** Measured Hole5 *****'
OUTPUT/FA(Hole5)
```

```
F(Hole6)=FEAT/CIRCLE,INNER,CART,-73.412,-33.607,-10.000,0,0,1,8.000
MEAS/CIRCLE,F(Hole6),4
GOTO/-34.607,-56.588,0.000
GOTO/-72.412,-33.607,0.000
GOTO/-72.412,-33.607,-13.000
PTMEAS/CART,-69.412,-33.607,-13.000,-1,0,0
GOTO/-73.412,-32.607,-13.000
PTMEAS/CART,-73.412,-29.607,-13.000,0,-1,0
GOTO/-74.412,-33.607,-13.000
PTMEAS/CART,-77.412,-33.607,-13.000,1,0,0
GOTO/-73.412,-34.607,-13.000
PTMEAS/CART,-73.412,-37.607,-13.000,0,1,0
ENDMES

TEXT/OUTFIL,' '
TEXT/OUTFIL,'****** Measured Hole6 *****'
OUTPUT/FA(Hole6)

F(PCirHol1)=FEAT/CIRCLE,INNER,CART,-65.000,-65.000,-10.000,0,0,1,65.000
CONST/CIRCLE,F(PCirHol1),BF,FA(Hole3),FA(Hole4),FA(Hole5),FA(Hole6)

TEXT/OUTFIL,' '
TEXT/OUTFIL,'****** Measured PCirHol1 *****'
OUTPUT/FA(PCirHol1)

T(TxPCirHol1)=TOL/CORTOL,XAXIS,-0.500,0.600
OUTPUT/FA(PCirHol1),TA(TxPCirHol1)
T(TdPCirHol1)=TOL/DIAM,-0.400,0.500
OUTPUT/FA(PCirHol1),TA(TdPCirHol1)

F(Cyl1)=FEAT/CIRCLE,OUTER,CART,-65.000,-65.000,-5.000,0,0,1,50.000
MEAS/CIRCLE,F(Cyl1),4
GOTO/-73.412,-34.607,-5.000
GOTO/-65.000,-37.000,-5.000
GOTO/-65.000,-37.000,-8.000
PTMEAS/CART,-65.000,-40.000,-8.000,0,1,0
GOTO/-65.000,-37.000,0.000
GOTO/-93.000,-65.000,0.000
GOTO/-93.000,-65.000,-8.000
PTMEAS/CART,-90.000,-65.000,-8.000,-1,0,0
GOTO/-93.000,-65.000,0.000
GOTO/-65.000,-93.000,0.000
GOTO/-65.000,-93.000,-8.000
PTMEAS/CART,-65.000,-90.000,-8.000,0,-1,0
GOTO/-65.000,-93.000,0.000
GOTO/-37.000,-65.000,0.000
GOTO/-37.000,-65.000,-8.000
PTMEAS/CART,-40.000,-65.000,-8.000,1,0,0
ENDMES

TEXT/OUTFIL,' '
TEXT/OUTFIL,'****** Measured Cyl1 *****'
```

```
OUTPUT/FA(Cyl1)

T(TxCyl1)=TOL/CORTOL,XAXIS,-0.300,0.400
OUTPUT/FA(Cyl1),TA(TxCyl1)
T(TyCyl1)=TOL/CORTOL,YAXIS,-0.400,0.500
OUTPUT/FA(Cyl1),TA(TyCyl1)
T(TdCyl1)=TOL/DIAM,-0.500,0.300
OUTPUT/FA(Cyl1),TA(TdCyl1)

GOTO/-37.000,-65.000,10.000
ENDFIL
```

4.7 DMIS file output (Prototype)

```
DECL/CHAR,ans
DECL/INTGR,F1
SNSLCT/S(PROBE1)

MACRO(check)/answer
F1=-1
IF ((ans .EQ. 'N') .OR. (ans .EQ. 'n'))
F1=0
IF ((ans .EQ. 'Y') .OR. (ans .EQ. 'y'))
F1=1
ENDMAC
WKPLAN/XYPLAN

F(Hole1)=FEAT/CIRCLE,INNER,CART,-15.000,-12.000,0.000,0,0,1,13.000
MEAS/CIRCLE,F(Hole1),4
(MOVE_NEXT_1)
GOTO/-11.500,-12.000,10.000
(ASK_1)
TEXT/QUERY,(R),1,A,L,'Is this a hole?'
READ/1,ans
CALL/M(check),ans
IF,F1,(ASK_1),(NO_MEAS_1),(MEAS_1)
(MEAS_1)
GOTO/-11.500,-12.000,-3.000
PTMEAS/CART,-8.500,-12.000,-3.000,-1,0,0
GOTO/-15.000,-8.500,-3.000
PTMEAS/CART,-15.000,-5.500,-3.000,0,-1,0
GOTO/-18.500,-12.000,-3.000
PTMEAS/CART,-21.500,-12.000,-3.000,1,0,0
GOTO/-15.000,-15.500,-3.000
PTMEAS/CART,-15.000,-18.500,-3.000,0,1,0
ENDMES

TEXT/OUTFIL,' '
TEXT/OUTFIL,'****** Measured Hole1 *****'
OUTPUT/FA(Hole1)

JUMPTO,(MOVE_NEXT_2)
(NO_MEAS_1)
GOTO/-15.000,-15.500,10.000
F(Hole2)=FEAT/CIRCLE,INNER,CART,-105.000,-12.000,0.000,0,0,1,13.000
MEAS/CIRCLE,F(Hole2),4
(MOVE_NEXT_2)
GOTO/-15.000,-15.500,5.000
GOTO/-101.500,-12.000,5.000
(ASK_2)
TEXT/QUERY,(R),1,A,L,'Is this a hole?'
READ/1,ans
CALL/M(check),ans
IF,F1,(ASK_2),(NO_MEAS_2),(MEAS_2)
(MEAS_2)
```

```
GOTO/-101.500,-12.000,-3.000
PTMEAS/CART,-98.500,-12.000,-3.000,-1,0,0
GOTO/-105.000,-8.500,-3.000
PTMEAS/CART,-105.000,-5.500,-3.000,0,-1,0
GOTO/-108.500,-12.000,-3.000
PTMEAS/CART,-111.500,-12.000,-3.000,1,0,0
GOTO/-105.000,-15.500,-3.000
PTMEAS/CART,-105.000,-18.500,-3.000,0,1,0
ENDMES

TEXT/OUTFIL,' '
TEXT/OUTFIL,'****** Measured Hole2 *****'
OUTPUT/FA(Hole2)

JUMPTO,(ASK_3)
(NO_MEAS_2)
GOTO/-105.000,-15.500,5.000
(ASK_3)
TEXT/QUERY,A,L,'distance between two features?'
READ/1,ans
CALL/M(check),ans
IF,F1,(ASK_3),(MOVE_NEXT_4),(MEAS_3)
(MEAS_3)
F(Dist2F1)=FEAT/LINE,BND,CART,-15.000,-12.000,0.000,-105.000,-12.000,0.000,0,0,1

CONST/LINE,F(Dist2F1),BF,FA(Hole1),FA(Hole2)
TEXT/OUTFIL,' '
TEXT/OUTFIL,'****** Measured Dist2F1 *****'
OUTPUT/FA(Dist2F1)

F(LenIn1)=FEAT/LINE,BND,CART,-126.500,-60.000,0.000,-126.500,-70.000,0.000,0,0,1
(MOVE_NEXT_4)
GOTO/-105.000,-15.500,5.000
GOTO/-126.500,-63.000,5.000
(ASK_4)
TEXT/QUERY,(R),1,A,L,'Is this a block?'
READ/1,ans
CALL/M(check),ans
IF,F1,(ASK_4),(NO_MEAS_4),(MEAS_4)
(MEAS_4)
GOTO/-126.500,-63.000,-3.000
F(Point1)=FEAT/POINT,CART,-126.500,-60.000,-3.000,0,-1,0
MEAS/POINT,F(Point1),1
PTMEAS/CART,-126.500,-60.000,-3.000,0,-1,0
ENDMES

GOTO/-126.500,-67.000,-3.000
GOTO/-126.500,-67.000,-3.000
F(Point2)=FEAT/POINT,CART,-126.500,-70.000,-3.000,0,1,0
MEAS/POINT,F(Point2),1
PTMEAS/CART,-126.500,-70.000,-3.000,0,1,0
ENDMES
```

```
CONST/LINE,F(LenIn1),BF,FA(Point1),FA(Point2)
TEXT/OUTFIL,' '
TEXT/OUTFIL,'****** Measured LenIn1 *****'
OUTPUT/FA(LenIn1)

JUMPTO,(MOVE_NEXT_5)
(NO_MEAS_4)
GOTO/-126.500,-67.000,5.000
F(Hole3)=FEAT/CIRCLE,INNER,CART,-96.393,-73.412,-10.000,0,0,1,8.000
MEAS/CIRCLE,F(Hole3),4
(MOVE_NEXT_5)
GOTO/-126.500,-67.000,5.000
GOTO/-97.393,-73.412,5.000
(ASK_5)
TEXT/QUERY,(R),1,A,L,'Is this a hole?'
READ/1,ans
CALL/M(check),ans
IF,F1,(ASK_5),(NO_MEAS_5),(MEAS_5)
(MEAS_5)
GOTO/-97.393,-73.412,-13.000
PTMEAS/CART,-100.393,-73.412,-13.000,1,0,0
GOTO/-96.393,-74.412,-13.000
PTMEAS/CART,-96.393,-77.412,-13.000,0,1,0
GOTO/-95.393,-73.412,-13.000
PTMEAS/CART,-92.393,-73.412,-13.000,-1,0,0
GOTO/-96.393,-72.412,-13.000
PTMEAS/CART,-96.393,-69.412,-13.000,0,-1,0
ENDMES

TEXT/OUTFIL,' '
TEXT/OUTFIL,'****** Measured Hole3 *****'
OUTPUT/FA(Hole3)

JUMPTO,(MOVE_NEXT_6)
(NO_MEAS_5)
GOTO/-96.393,-72.412,5.000
F(Hole4)=FEAT/CIRCLE,INNER,CART,-56.588,-96.393,-10.000,0,0,1,8.000
MEAS/CIRCLE,F(Hole4),4
(MOVE_NEXT_6)
GOTO/-96.393,-72.412,0.000
GOTO/-57.588,-96.393,0.000
(ASK_6)
TEXT/QUERY,(R),1,A,L,'Is this a hole?'
READ/1,ans
CALL/M(check),ans
IF,F1,(ASK_6),(NO_MEAS_6),(MEAS_6)
(MEAS_6)
GOTO/-57.588,-96.393,-13.000
PTMEAS/CART,-60.588,-96.393,-13.000,1,0,0
GOTO/-56.588,-97.393,-13.000
PTMEAS/CART,-56.588,-100.393,-13.000,0,1,0
GOTO/-55.588,-96.393,-13.000
```

```
PTMEAS/CART,-52.588,-96.393,-13.000,-1,0,0
GOTO/-56.588,-95.393,-13.000
PTMEAS/CART,-56.588,-92.393,-13.000,0,-1,0
ENDMES

TEXT/OUTFIL,' '
TEXT/OUTFIL,'****** Measured Hole4 *****'
OUTPUT/FA(Hole4)

JUMPTO,(MOVE_NEXT_7)
(NO_MEAS_6)
GOTO/-56.588,-95.393,0.000
F(Hole5)=FEAT/CIRCLE,INNER,CART,-33.607,-56.588,-10.000,0,0,1,8.000
MEAS/CIRCLE,F(Hole5),4
(MOVE_NEXT_7)
GOTO/-56.588,-95.393,0.000
GOTO/-33.607,-57.588,0.000
(ASK_7)
TEXT/QUERY,(R),1,A,L,'Is this a hole?'
READ/1,ans
CALL/M(check),ans
IF,F1,(ASK_7),(NO_MEAS_7),(MEAS_7)
(MEAS_7)
GOTO/-33.607,-57.588,-13.000
PTMEAS/CART,-33.607,-60.588,-13.000,0,1,0
GOTO/-32.607,-56.588,-13.000
PTMEAS/CART,-29.607,-56.588,-13.000,-1,0,0
GOTO/-33.607,-55.588,-13.000
PTMEAS/CART,-33.607,-52.588,-13.000,0,-1,0
GOTO/-34.607,-56.588,-13.000
PTMEAS/CART,-37.607,-56.588,-13.000,1,0,0
ENDMES

TEXT/OUTFIL,' '
TEXT/OUTFIL,'****** Measured Hole5 *****'
OUTPUT/FA(Hole5)

JUMPTO,(MOVE_NEXT_8)
(NO_MEAS_7)
GOTO/-34.607,-56.588,0.000
F(Hole6)=FEAT/CIRCLE,INNER,CART,-73.412,-33.607,-10.000,0,0,1,8.000
MEAS/CIRCLE,F(Hole6),4
(MOVE_NEXT_8)
GOTO/-34.607,-56.588,0.000
GOTO/-72.412,-33.607,0.000
(ASK_8)
TEXT/QUERY,(R),1,A,L,'Is this a hole?'
READ/1,ans
CALL/M(check),ans
IF,F1,(ASK_8),(NO_MEAS_8),(MEAS_8)
(MEAS_8)
GOTO/-72.412,-33.607,-13.000
PTMEAS/CART,-69.412,-33.607,-13.000,-1,0,0
```

```
GOTO/-73.412,-32.607,-13.000
PTMEAS/CART,-73.412,-29.607,-13.000,0,-1,0
GOTO/-74.412,-33.607,-13.000
PTMEAS/CART,-77.412,-33.607,-13.000,1,0,0
GOTO/-73.412,-34.607,-13.000
PTMEAS/CART,-73.412,-37.607,-13.000,0,1,0
ENDMES

TEXT/OUTFIL,' '
TEXT/OUTFIL,'****** Measured Hole6 *****'
OUTPUT/FA(Hole6)

JUMPTO,(ASK_9)
(NO_MEAS_8)
GOTO/-73.412,-34.607,0.000
(ASK_9)
TEXT/QUERY,A,L,'Pitch Circle?'
READ/1,ans
CALL/M(check),ans
IF,F1,(ASK_9),(MOVE_NEXT_10),(MEAS_9)
(MEAS_9)
F(PCirHol1)=FEAT/CIRCLE,INNER,CART,-65.000,-65.000,-10.000,0,0,1,65.000
CONST/CIRCLE,F(PCirHol1),BF,FA(Hole3),FA(Hole4),FA(Hole5),FA(Hole6)

TEXT/OUTFIL,' '
TEXT/OUTFIL,'****** Measured PCirHol1 *****'
OUTPUT/FA(PCirHol1)

F(Cyl1)=FEAT/CIRCLE,OUTER,CART,-65.000,-65.000,-5.000,0,0,1,50.000
MEAS/CIRCLE,F(Cyl1),4
(MOVE_NEXT_10)
GOTO/-73.412,-34.607,-5.000
GOTO/-65.000,-37.000,-5.000
(ASK_10)
TEXT/QUERY,(R),1,A,L,'Is this a cylinder?'
READ/1,ans
CALL/M(check),ans
IF,F1,(ASK_10),(END),(MEAS_10)
(MEAS_10)
GOTO/-65.000,-37.000,-8.000
PTMEAS/CART,-65.000,-40.000,-8.000,0,1,0
GOTO/-65.000,-37.000,0.000
GOTO/-93.000,-65.000,0.000
GOTO/-93.000,-65.000,-8.000
PTMEAS/CART,-90.000,-65.000,-8.000,-1,0,0
GOTO/-93.000,-65.000,0.000
GOTO/-65.000,-93.000,0.000
GOTO/-65.000,-93.000,-8.000
PTMEAS/CART,-65.000,-90.000,-8.000,0,-1,0
GOTO/-65.000,-93.000,0.000
GOTO/-37.000,-65.000,0.000
GOTO/-37.000,-65.000,-8.000
PTMEAS/CART,-40.000,-65.000,-8.000,1,0,0
```

```
ENDMES

TEXT/OUTFIL,' '
TEXT/OUTFIL,'****** Measured Cyl1 *****'
OUTPUT/FA(Cyl1)

(NO_MEAS_10)
GOTO/-37.000,-65.000,-5.000
GOTO/-37.000,-65.000,10.000
(END)
ENDFIL
```

4.8 Feed back

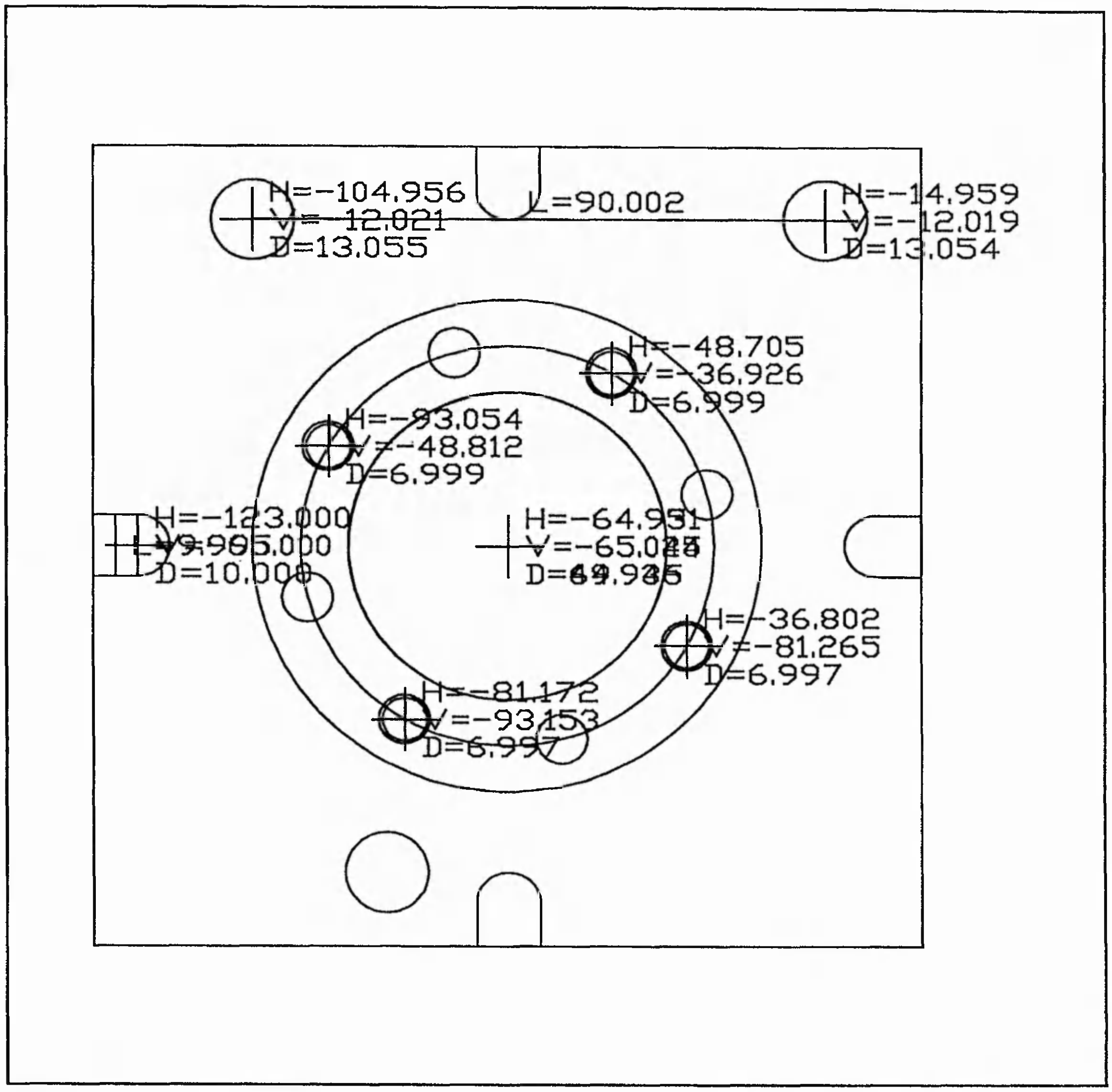


Figure 3 : Reading a DMIS file

The CMM generates a DMIS file with all the measured values and the tolerance information. This DMIS file can be read back into the system and a graphical displayed of the measured values can be shown as in the above drawing. If the user wants to get this information back into the CAD system an IGES output can be generated, this file will contain the information to display the drawing as in the drawing above.

The DMIS file generated by the CMM for this particular inspection program and the IGES file generated by the system are shoen in the following pages.

4.9 DMIS file input

```
FILNAM/'Test'
$$ LK-DMIS Version 1.3, Mon Apr 26 18:19:36 1993
UNITS/MM,ANGDMS,TEMPF
D(mcs)=DATSET/MCS
PRCOMP/ON
DATDEF/FA(top_face),DAT(A)
DATDEF/FA(line),DAT(B)
DATDEF/FA(line),DAT(B)
DATDEF/FA(point),DAT(C)
D(part)=DATSET/DAT(A),ZDIR,ZORIG,DAT(B),XDIR,YORIG,DAT(C),YDIR,XORIG
RECALL/D(part)

TEXT/OUTFIL,'****** Measured Hole1 *****'
OUTPUT/FA(Hole1)
FA(Hole1)=FEAT/CIRCLE,INNER,CART,-14.959,-12.019,0,0,0,1,13.054

TEXT/OUTFIL,'****** Measured Hole2 *****'
OUTPUT/FA(Hole2)
FA(Hole2)=FEAT/CIRCLE,INNER,CART,-104.956,-12.021,0,0,0,1,13.055

TEXT/OUTFIL,'****** Measured Dist2F1 *****'
OUTPUT/FA(Dist2F1)
FA(Dist2F1)=FEAT/LINE,BND,CART,-15.000,-12.000,0.000,-105.000,-12.000,0.000,0,0,1

OUTPUT/TA(TolDist2F1)
TA(TolDist2F1)=TOL/DISTB,INTOL,NOMINL,90.002,-0.100,0.500,PT2PT

TEXT/OUTFIL,'****** Measured LenIn1 *****'
OUTPUT/FA(LenIn1)

FA(LenIn1)=FEAT/LINE,BND,CART,-126.500,-60.000,0.000,-126.500,-70.000,0.000,0,0,1
FA(Point1)=FEAT/POINT,CART,-126.500,-60.000,-3.000,0,-1,0
FA(Point2)=FEAT/POINT,CART,-126.500,-70.000,-3.000,0,1,0

OUTPUT/TA(TolLenIn1)
TA(TolLenIn1)=TOL/DISTB,NOMINL,OUTOL,9.995,-0.200,0.100,YAXIS

TEXT/OUTFIL,'****** Measured ArcInt1 *****'
OUTPUT/FA(ArcInt1)
FA(ArcInt1)=FEAT/ARC,INNER,CART,-123.000,-65.000,0.000,0,0,1,5.000,-1.571,1.571

OUTPUT/FA(ArcInt1),TA(TxArcInt1)
TA(TxArcInt1)=TOL/CORTOL,XAXIS,0.300,INTOL
OUTPUT/FA(ArcInt1),TA(TdArcInt1)
TA(TdArcInt1)=TOL/DIAM,0.400,INTOL

TEXT/OUTFIL,'****** Measured Hole3 *****'
OUTPUT/FA(Hole3)
FA(Hole3)=FEAT/CIRCLE,INNER,CART,-93.054,-48.812,-10,0,0,1,6.999
TA(xaxis)=TOL/CORTOL,XAXIS,0.300,INTOL
TA(yaxis)=TOL/CORTOL,YAXIS,0.300,INTOL
TA(DIA)=TOL/DIAM,0.400,INTOL

TEXT/OUTFIL,'****** Measured Hole4 *****'
OUTPUT/FA(Hole4)
FA(Hole4)=FEAT/CIRCLE,INNER,CART,-81.172,-93.153,-10,0,0,1,6.997
TA(xaxis)=TOL/CORTOL,XAXIS,0.300,OUTOL
TA(yaxis)=TOL/CORTOL,YAXIS,0.300,OUTOL
TA(DIA)=TOL/DIAM,0.400,OUTOL

TEXT/OUTFIL,'****** Measured Hole5 *****'
OUTPUT/FA(Hole5)
FA(Hole5)=FEAT/CIRCLE,INNER,CART,-36.802,-81.265,-10,0,0,1,6.997
TA(xaxis)=TOL/CORTOL,XAXIS,0.300,INTOL
TA(yaxis)=TOL/CORTOL,YAXIS,0.300,OUTOL
TA(DIA)=TOL/DIAM,0.400,OUTOL

TEXT/OUTFIL,'****** Measured Hole6 *****'
OUTPUT/FA(Hole6)
FA(Hole6)=FEAT/CIRCLE,INNER,CART,-48.705,-36.926,-10,0,0,1,6.999
TA(xaxis)=TOL/CORTOL,XAXIS,0.300,OUTOL
TA(yaxis)=TOL/CORTOL,YAXIS,0.300,OUTOL
TA(DIA)=TOL/DIAM,0.400,INTOL

TEXT/OUTFIL,'****** Measured PCirHol1 *****'
OUTPUT/FA(PCirHol1)
FA(PCirHol1)=FEAT/CIRCLE,INNER,CART,-64.931,-65.044,-10,0,0,1,64.935

TA(TxPCirHol1)=TOL/CORTOL,XAXIS,0.300,OUTOL
OUTPUT/FA(PCirHol1),TA(TxPCirHol1)
TA(TdPCirHol1)=TOL/DIAM,0.300,INTOL
OUTPUT/FA(PCirHol1),TA(TdPCirHol1)

TEXT/OUTFIL,'****** Measured Cyl1 *****'
OUTPUT/FA(Cyl1)
FA(Cyl1)=FEAT/CIRCLE,OUTER,CART,-64.951,-65.025,-5,0,0,1,49.946
TA(TxCyl1)=TOL/CORTOL,XAXIS,0.100,INTOL
OUTPUT/FA(Cyl1),TA(TxCyl1)
TA(TyCyl1)=TOL/CORTOL,YAXIS,0.300,OUTOL
OUTPUT/FA(Cyl1),TA(TyCyl1)
TA(TdCyl1)=TOL/DIAM,0.400,OUTOL
OUTPUT/FA(Cyl1),TA(TdCyl1)
ENDFIL
```

4.10 IGES file output

```
IGES file generated from a DMIS output from LK CMM by the IGES          S0000001
translator from Ane CAD-INSPECTON program, version IGES 1.1             S0000002
,,5HFACEA,18HB:\FACES\FACEA.IGS,13HAutoCAD-10 c2,11HIGESOUT-2.0,16,38,6,G0000001
99,15,5HFACEA,1.0,1,4HINCH,32767,3.2767D1,13H930218.091532,1.0D-8,      G0000002
2.8570410776966D2,6HThroop,14HAutodesk, Inc.,4,0;                       G0000003
     110       1       0       0       0       0       0 000000000D0000001
     110       0       8       1       0          LABEL         0D0000002
     110       2       0       0       0       0       0 000000000D0000003
     110       0       8       1       0          LABEL         0D0000004
     110       3       0       0       0       0       0 000000000D0000005
     110       0       8       1       0          LABEL         0D0000006
     110       4       0       0       0       0       0 000000000D0000007
     110       0       8       1       0          LABEL         0D0000008
     110       5       0       0       0       0       0 000000000D0000009
     110       0       8       1       0          LABEL         0D0000010
     110       6       0       0       0       0       0 000000000D0000011
     110       0       8       1       0          LABEL         0D0000012
     110       7       0       0       0       0       0 000000000D0000013
     110       0       8       1       0          LABEL         0D0000014
     110       8       0       0       0       0       0 000000000D0000015
     110       0       8       1       0          LABEL         0D0000016
     110       9       0       0       0       0       0 000000000D0000017
     110       0       8       1       0          LABEL         0D0000018
     110      10       0       0       0       0       0 000000000D0000019
     110       0       8       1       0          LABEL         0D0000020
     110      11       0       0       0       0       0 000000000D0000021
     110       0       8       1       0          LABEL         0D0000022
     110      12       0       0       0       0       0 000000000D0000023
     110       0       8       1       0          LABEL         0D0000024
     110      13       0       0       0       0       0 000000000D0000025
     110       0       8       1       0          LABEL         0D0000026
     110      14       0       0       0       0       0 000000000D0000027
     110       0       8       1       0          LABEL         0D0000028
     110      15       0       0       0       0       0 000000000D0000029
     110       0       8       1       0          LABEL         0D0000030
     110      16       0       0       0       0       0 000000000D0000031
     110       0       8       1       0          LABEL         0D0000032
     100      17       0       0       0       0       0 000000000D0000033
     100       0       8       1       0          LABEL         0D0000034
     100      18       0       0       0       0       0 000000000D0000035
     100       0       8       1       0          LABEL         0D0000036
     100      19       0       0       0       0       0 000000000D0000037
     100       0       8       1       0          LABEL         0D0000038
     100      20       0       0       0       0       0 000000000D0000039
     100       0       8       1       0          LABEL         0D0000040
     100      21       0       0       0       0       0 000000000D0000041
     100       0       8       1       0          LABEL         0D0000042
     100      22       0       0       0       0       0 000000000D0000043
     100       0       8       1       0          LABEL         0D0000044
     100      23       0       0       0       0       0 000000000D0000045
     100       0       8       1       0          LABEL         0D0000046
     100      24       0       0       0       0       0 000000000D0000047
     100       0       8       1       0          LABEL         0D0000048
     100      25       0       0       0       0       0 000000000D0000049
     100       0       8       1       0          LABEL         0D0000050
     100      26       0       0       0       0       0 000000000D0000051
     100       0       8       1       0          LABEL         0D0000052
     100      27       0       0       0       0       0 000000000D0000053
     100       0       8       1       0          LABEL         0D0000054
     100      28       0       0       0       0       0 000000000D0000055
     100       0       8       1       0          LABEL         0D0000056
     100      29       0       0       0       0       0 000000000D0000057
     100       0       8       1       0          LABEL         0D0000058
     100      30       0       0       0       0       0 000000000D0000059
     100       0       8       1       0          LABEL         0D0000060
     100      31       0       0       0       0       0 000000000D0000061
     100       0       8       1       0          LABEL         0D0000062
     100      32       0       0       0       0       0 000000000D0000063
     100       0       8       1       0          LABEL         0D0000064
     100      33       0       0       0       0       0 000000000D0000065
     100       0       8       1       0          LABEL         0D0000066
     110      34       0       0       0       0       0 000000000D0000067
     110       0       3       1       0          LABEL         0D0000068
     212      35       0       0       0       0       0 000000000D0000069
     212       0       3       2       0          LABEL         0D0000070
     110      37       0       0       0       0       0 000000000D0000071
     110       0       2       1       0          LABEL         0D0000072
```

```
212     38      0       0       0       0       0       000000000D0000073
212      0      2       2       0               LABEL           0D0000074
110     40      0       0       0       0       0       000000000D0000075
110      0      4       1       0               LABEL           0D0000076
212     41      0       0       0       0       0       000000000D0000077
212      0      4       2       0               LABEL           0D0000078
110     43      0       0       0       0       0       000000000D0000079
110      0      4       1       0               LABEL           0D0000080
212     44      0       0       0       0       0       000000000D0000081
212      0      4       2       0               LABEL           0D0000082
100     46      0       0       0       0       0       000000000D0000083
100      0      4       1       0               LABEL           0D0000084
212     47      0       0       0       0       0       000000000D0000085
212      0      4       2       0               LABEL           0D0000086
110     49      0       0       0       0       0       000000000D0000087
110      0      4       1       0               LABEL           0D0000088
212     50      0       0       0       0       0       000000000D0000089
212      0      4       2       0               LABEL           0D0000090
110     52      0       0       0       0       0       000000000D0000091
110      0      4       1       0               LABEL           0D0000092
212     53      0       0       0       0       0       000000000D0000093
212      0      4       2       0               LABEL           0D0000094
100     55      0       0       0       0       0       000000000D0000095
100      0      4       1       0               LABEL           0D0000096
212     56      0       0       0       0       0       000000000D0000097
212      0      4       2       0               LABEL           0D0000098
110     58      0       0       0       0       0       000000000D0000099
110      0      3       1       0               LABEL           0D0000100
212     59      0       0       0       0       0       000000000D0000101
212      0      3       2       0               LABEL           0D0000102
110     61      0       0       0       0       0       000000000D0000103
110      0      4       1       0               LABEL           0D0000104
212     62      0       0       0       0       0       000000000D0000105
212      0      4       2       0               LABEL           0D0000106
100     64      0       0       0       0       0       000000000D0000107
100      0      3       1       0               LABEL           0D0000108
212     65      0       0       0       0       0       000000000D0000109
212      0      3       2       0               LABEL           0D0000110
110     67      0       0       0       0       0       000000000D0000111
110      0      3       1       0               LABEL           0D0000112
212     68      0       0       0       0       0       000000000D0000113
212      0      3       2       0               LABEL           0D0000114
110     70      0       0       0       0       0       000000000D0000115
110      0      3       1       0               LABEL           0D0000116
212     71      0       0       0       0       0       000000000D0000117
212      0      3       2       0               LABEL           0D0000118
100     73      0       0       0       0       0       000000000D0000119
100      0      3       1       0               LABEL           0D0000120
212     74      0       0       0       0       0       000000000D0000121
212      0      3       2       0               LABEL           0D0000122
110     76      0       0       0       0       0       000000000D0000123
110      0      2       1       0               LABEL           0D0000124
212     77      0       0       0       0       0       000000000D0000125
212      0      2       2       0               LABEL           0D0000126
110     79      0       0       0       0       0       000000000D0000127
110      0      2       1       0               LABEL           0D0000128
212     80      0       0       0       0       0       000000000D0000129
212      0      2       2       0               LABEL           0D0000130
100     82      0       0       0       0       0       000000000D0000131
100      0      2       1       0               LABEL           0D0000132
212     83      0       0       0       0       0       000000000D0000133
212      0      2       2       0               LABEL           0D0000134
110     85      0       0       0       0       0       000000000D0000135
110      0      3       1       0               LABEL           0D0000136
212     86      0       0       0       0       0       000000000D0000137
212      0      3       2       0               LABEL           0D0000138
110     88      0       0       0       0       0       000000000D0000139
110      0      2       1       0               LABEL           0D0000140
212     89      0       0       0       0       0       000000000D0000141
212      0      2       2       0               LABEL           0D0000142
100     91      0       0       0       0       0       000000000D0000143
100      0      2       1       0               LABEL           0D0000144
212     92      0       0       0       0       0       000000000D0000145
212      0      2       2       0               LABEL           0D0000146
110     94      0       0       0       0       0       000000000D0000147
110      0      2       1       0               LABEL           0D0000148
212     95      0       0       0       0       0       000000000D0000149
212      0      2       2       0               LABEL           0D0000150
110     97      0       0       0       0       0       000000000D0000151
110      0      2       1       0               LABEL           0D0000152
```

```
     212      98      0       0       0       0       0        000000000D0000153
     212       0      2       2       0            LABEL              0D0000154
     100     100      0       0       0       0       0        000000000D0000155
     100       0      3       1       0            LABEL              0D0000156
     212     101      0       0       0       0       0        000000000D0000157
     212       0      3       2       0            LABEL              0D0000158
     110     103      0       0       0       0       0        000000000D0000159
     110       0      2       1       0            LABEL              0D0000160
     212     104      0       0       0       0       0        000000000D0000161
     212       0      2       2       0            LABEL              0D0000162
     110     106      0       0       0       0       0        000000000D0000163
     110       0      4       1       0            LABEL              0D0000164
     212     107      0       0       0       0       0        000000000D0000165
     212       0      4       2       0            LABEL              0D0000166
     100     109      0       0       0       0       0        000000000D0000167
     100       0      3       1       0            LABEL              0D0000168
     212     110      0       0       0       0       0        000000000D0000169
     212       0      3       2       0            LABEL              0D0000170
     110     112      0       0       0       0       0        000000000D0000171
     110       0      3       1       0            LABEL              0D0000172
     212     113      0       0       0       0       0        000000000D0000173
     212       0      3       2       0            LABEL              0D0000174
     110     115      0       0       0       0       0        000000000D0000175
     110       0      2       1       0            LABEL              0D0000176
     212     116      0       0       0       0       0        000000000D0000177
     212       0      2       2       0            LABEL              0D0000178
     100     118      0       0       0       0       0        000000000D0000179
     100       0      2       1       0            LABEL              0D0000180
     212     119      0       0       0       0       0        000000000D0000181
     212       0      2       2       0            LABEL              0D0000182
110,260.000,100.000,0.0,260.000,230.000,0.0;                            1P0000001
110,260.000,230.000,0.0,130.000,230.000,0.0;                            3P0000002
110,130.000,230.000,0.0,130.000,100.000,0.0;                            5P0000003
110,130.000,100.000,0.0,260.000,100.000,0.0;                            7P0000004
110,130.000,170.000,0.0,137.000,170.000,0.0;                            9P0000005
110,130.000,160.000,0.0,137.000,160.000,0.0;                           11P0000006
110,200.000,100.000,0.0,200.000,107.000,0.0;                           13P0000007
110,190.000,100.000,0.0,190.000,107.000,0.0;                           15P0000008
110,200.000,230.000,0.0,200.000,223.000,0.0;                           17P0000009
110,190.000,230.000,0.0,190.000,223.000,0.0;                           19P0000010
110,260.000,160.000,0.0,253.000,160.000,0.0;                           21P0000011
110,260.000,170.000,0.0,253.000,170.000,0.0;                           23P0000012
110,194.910,165.000,0.0,195.090,165.000,0.0;                           25P0000013
110,195.000,164.910,0.0,195.000,165.090,0.0;                           27P0000014
110,194.910,165.000,0.0,195.090,165.000,0.0;                           29P0000015
110,195.000,164.910,0.0,195.000,165.090,0.0;                           31P0000016
100,0.0,245.000,218.000,251.500,218.000,251.500,218.000;               33P0000017
100,0.0,155.000,218.000,161.500,218.000,161.500,218.000;               35P0000018
100,0.0,195.000,165.000,220.000,165.000,220.000,165.000;               37P0000019
100,0.0,195.000,165.000,235.000,165.000,235.000,165.000;               39P0000020
100,0.0,186.588,196.393,190.588,196.393,190.588,196.393;               41P0000021
100,0.0,166.854,181.250,170.854,181.250,170.854,181.250;               43P0000022
100,0.0,163.607,156.588,167.607,156.588,167.607,156.588;               45P0000023
100,0.0,178.750,136.854,182.750,136.854,182.750,136.854;               47P0000024
100,0.0,203.412,133.607,207.412,133.607,207.412,133.607;               49P0000025
100,0.0,223.146,148.750,227.146,148.750,227.146,148.750;               51P0000026
100,0.0,226.393,173.412,230.393,173.412,230.393,173.412;               53P0000027
100,0.0,211.250,193.146,215.250,193.146,215.250,193.146;               55P0000028
100,0.0,176.000,112.000,182.500,112.000,182.500,112.000;               57P0000029
100,0.0,137.000,165.000,137.000,160.000,137.000,170.000;               59P0000030
100,0.0,195.000,107.000,200.000,107.000,190.000,107.000;               61P0000031
100,0.0,195.000,223.000,190.000,223.000,200.000,223.000;               63P0000032
100,0.0,253.000,165.000,253.000,170.000,253.000,160.000;               65P0000033
110,245.000,218.000,0.0,155.000,218.000,0.0;                           67P0000034
212,1,8,24.0,3.0,1,,0.0,0,0,198.500,219.500,0.000,                     69P0000035
8HL=90.002;                                                            69P0000036
110,133.500,170.000,0.0,133.500,160.000,0.0;                           71P0000037
212,1,7,21.0,3.0,1,,0.0,0,0,136.500,163.500,0.000,                     73P0000038
7HL=9.995;                                                             73P0000039
110,250.041,217.981,0.0,240.041,217.981,0.0;                           75P0000040
212,1,9,27.0,3.0,1,,0.0,0,0,248.041,220.981,0.000,                     77P0000041
9HH=-14.959;                                                           77P0000042
110,245.041,222.981,0.0,245.041,212.981,0.0;                           79P0000043
212,1,9,27.0,3.0,1,,0.0,0,0,248.041,216.481,0.000,                     81P0000044
9HV=-12.019;                                                           81P0000045
100,0.0,245.041,217.981,251.568,217.981,251.568,217.981;               83P0000046
212,1,8,24.0,3.0,1,,0.0,0,0,248.041,211.981,0.000,                     85P0000047
8HD=13.054;                                                            85P0000048
110,160.044,217.979,0.0,150.044,217.979,0.0;                           87P0000049
212,1,10,30.0,3.0,1,,0.0,0,0,158.044,220.979,0.000,                    89P0000050
```

```
10HH=-104.956;                                                   89P0000051
110,155.044,222.979,0.0,155.044,212.979,0.0;                     91P0000052
212,1,9,27.0,3.0,1,,0.0,0,0,158.044,216.479,0.000,               93P0000053
9HV=-12.021;                                                     93P0000054
100,0.0,155.044,217.979,161.572,217.979,161.572,217.979;         95P0000055
212,1,8,24.0,3.0,1,,0.0,0,0,158.044,211.979,0.000,               97P0000056
8HD=13.055;                                                      97P0000057
110,142.000,165.000,0.0,132.000,165.000,0.0;                     99P0000058
212,1,10,30.0,3.0,1,,0.0,0,0,140.000,168.000,0.000,             101P0000059
10HH=-123.000;                                                  101P0000060
110,137.000,170.000,0.0,137.000,160.000,0.0;                    103P0000061
212,1,9,27.0,3.0,1,,0.0,0,0,140.000,163.500,0.000,              105P0000062
9HV=-65.000;                                                    105P0000063
100,0.0,137.000,165.000,136.999,160.000,136.999,170.000;        107P0000064
212,1,8,24.0,3.0,1,,0.0,0,0,140.000,159.000,0.000,              109P0000065
8HD=10.000;                                                     109P0000066
110,171.946,181.188,0.0,161.946,181.188,0.0;                    111P0000067
212,1,9,27.0,3.0,1,,0.0,0,0,169.946,184.188,0.000,              113P0000068
9HH=-93.054;                                                    113P0000069
110,166.946,186.188,0.0,166.946,176.188,0.0;                    115P0000070
212,1,9,27.0,3.0,1,,0.0,0,0,169.946,179.688,0.000,              117P0000071
9HV=-48.812;                                                    117P0000072
100,0.0,166.946,181.188,170.445,181.188,170.445,181.188;        119P0000073
212,1,7,21.0,3.0,1,,0.0,0,0,169.946,175.188,0.000,              121P0000074
7HD=6.999;                                                      121P0000075
110,183.828,136.847,0.0,173.828,136.847,0.0;                    123P0000076
212,1,9,27.0,3.0,1,,0.0,0,0,181.828,139.847,0.000,              125P0000077
9HH=-81.172;                                                    125P0000078
110,178.828,141.847,0.0,178.828,131.847,0.0;                    127P0000079
212,1,9,27.0,3.0,1,,0.0,0,0,181.828,135.347,0.000,              129P0000080
9HV=-93.153;                                                    129P0000081
100,0.0,178.828,136.847,182.327,136.847,182.327,136.847;        131P0000082
212,1,7,21.0,3.0,1,,0.0,0,0,181.828,130.847,0.000,              133P0000083
7HD=6.997;                                                      133P0000084
110,228.198,148.735,0.0,218.198,148.735,0.0;                    135P0000085
212,1,9,27.0,3.0,1,,0.0,0,0,226.198,151.735,0.000,              137P0000086
9HH=-36.802;                                                    137P0000087
110,223.198,153.735,0.0,223.198,143.735,0.0;                    139P0000088
212,1,9,27.0,3.0,1,,0.0,0,0,226.198,147.235,0.000,              141P0000089
9HV=-81.265;                                                    141P0000090
100,0.0,223.198,148.735,226.697,148.735,226.697,148.735;        143P0000091
212,1,7,21.0,3.0,1,,0.0,0,0,226.198,142.735,0.000,              145P0000092
7HD=6.997;                                                      145P0000093
110,216.295,193.074,0.0,206.295,193.074,0.0;                    147P0000094
212,1,9,27.0,3.0,1,,0.0,0,0,214.295,196.074,0.000,              149P0000095
9HH=-48.705;                                                    149P0000096
110,211.295,198.074,0.0,211.295,188.074,0.0;                    151P0000097
212,1,9,27.0,3.0,1,,0.0,0,0,214.295,191.574,0.000,              153P0000098
9HV=-36.926;                                                    153P0000099
100,0.0,211.295,193.074,214.794,193.074,214.794,193.074;        155P0000100
212,1,7,21.0,3.0,1,,0.0,0,0,214.295,187.074,0.000,              157P0000101
7HD=6.999;                                                      157P0000102
110,200.069,164.956,0.0,190.069,164.956,0.0;                    159P0000103
212,1,9,27.0,3.0,1,,0.0,0,0,198.069,167.956,0.000,              161P0000104
9HH=-64.931;                                                    161P0000105
110,195.069,169.956,0.0,195.069,159.956,0.0;                    163P0000106
212,1,9,27.0,3.0,1,,0.0,0,0,198.069,163.456,0.000,              165P0000107
9HV=-65.044;                                                    165P0000108
100,0.0,195.069,164.956,227.536,164.956,227.536,164.956;        167P0000109
212,1,8,24.0,3.0,1,,0.0,0,0,198.069,158.956,0.000,              169P0000110
8HD=64.935;                                                     169P0000111
110,200.049,164.975,0.0,190.049,164.975,0.0;                    171P0000112
212,1,9,27.0,3.0,1,,0.0,0,0,198.049,167.975,0.000,              173P0000113
9HH=-64.951;                                                    173P0000114
110,195.049,169.975,0.0,195.049,159.975,0.0;                    175P0000115
212,1,9,27.0,3.0,1,,0.0,0,0,198.049,163.475,0.000,              177P0000116
9HV=-65.025;                                                    177P0000117
100,0.0,195.049,164.975,220.022,164.975,220.022,164.975;        179P0000118
212,1,8,24.0,3.0,1,,0.0,0,0,198.049,158.975,0.000,              181P0000119
8HD=49.946;                                                     181P0000120
S      2G      3D     182P     120                                 T0000001
```

Appendix 5 : List of files

Data Structure

```
#define MY_NEW(a)  malloc(sizeof(*a))
#define pi 3.1415927

typedef enum {xy, yz, xz} wp;
typedef enum {p1, p2, p3, p4, p5} probe;

typedef struct line
{
  float                   x1;
  float                   y1;
  float                   x2;
  float                   y2;
  float                   xr1;
  float                   yr1;
  float                   zr1;
  float                   xr2;
  float                   yr2;
  float                   zr2;
  int                     select;
  struct line             *next;
}*pointer_line;

typedef struct scr_line
{
  float                   x1;
  float                   y1;
  float                   x2;
  float                   y2;
  struct scr_line         *next;
}*pointer_scrL;

typedef struct arc
{
  float                   z;
  float                   xc;
  float                   yc;
  float                   x1;
  float                   y1;
  float                   x2;
  float                   y2;
  float                   xrc;
  float                   yrc;
  float                   xr1;
  float                   yr1;
  float                   xr2;
  float                   yr2;
  float                   rad;
  int                     select;
  struct arc              *next;
}*pointer_arc;

typedef struct scr_arc
{
  float                   xc;
  float                   yc;
  float                   x1;
  float                   y1;
  float                   x2;
  float                   y2;
  struct scr_arc          *next;
}*pointer_scrA;

typedef struct selarc
{
  pointer_scrA            arc;
  struct selarc           *next;
}*pointer_SelArc;

typedef struct PD
{
  char                    line[100];
  struct PD               *next;
}*pointer_PD;

typedef struct
{
  char                    see[4];
  char                    sub[4];
  char                    use[4];
  char                    hie[4];
}state_type;

typedef struct DE
{
  int                     type;
  int                     punt_PD;
  int                     n_line;
  state_type              state;
  int                     color;
  int                     lines_PD;
  int                     form;
  struct DE               *next;
}*pointer_DE;

typedef struct pline
{
  pointer_line            line;
  pointer_arc             arc;
  int                     select;
  struct pline            *next;
}*pointer_pline;

typedef struct ent_sel
{
  pointer_line            line;
  pointer_arc             arc;
  pointer_pline           pline;
  int                     is_line;
  int                     is_arc;
  int                     is_pline;
  struct ent_sel          *next;
}*pointer_ent_selec;

typedef struct string
```

```
{
 float                   x1;
 float                   y1;
 char                    text[80];
 struct string     *next;
}*pointer_string;

typedef struct tolarc
{
 float                   UpH;
 float                   LowH;
 float                   UpV;
 float                   LowV;
 float                   UpD;
 float                   LowD;
 int                     equal;
 int                     NumCas;
 int                     h;
 int                     v;
}*pointer_Tol;

typedef struct fea_arc
{
 int               ext;
 float                   hc;
 float                   vc;
 float                   h1;
 float                   v1;
 float                   h2;
 float                   v2;
 float                   dep;
 float                   rad;
 pointer_scrA            arc;
 pointer_Tol             Tol;
 struct fea_arc          *next;
}*pointer_feat_arc;

typedef struct fea_cir
{
 float                   hor;
 float                   ver;
 float                   dep;
 float                   dia;
 int               ext;
 pointer_scrA            circle;
 pointer_Tol             Tol;
 struct fea_cir    *next;
}*pointer_feat_cir;

typedef struct fea_lenin
{
 float                   hac;
 float                   dist_h;
 float                   vac;
 float                   dist_v;
 float                   dep;
 float                   inside;
 pointer_scrL            line;
 pointer_Tol             Tol;
}*pointer_feat_len;

typedef struct fea_wTh
{
 float                   h1;
 float                   v1;
 float                   h2;
 float                   v2;
 float                   hc;
 float                   vc;
 float                   ang;
 float                   dep1;
 float                   dep2;
 float                   inside;
 pointer_scrL            line;
 pointer_Tol             Tol;
}*pointer_feat_wTh;

typedef struct point
{
 float                   h;
 float                   v;
 float                   dep;
 struct point      *next;
}*pointer_point;

typedef struct fea_lentru
{
 pointer_point           point;
 pointer_scrL            line;
 pointer_Tol             Tol;
}*pointer_feat_lentru;

typedef struct PCir
{
 pointer_feat_cir        list_circle;
 pointer_scrA            pc_cir;
 float                   hc;
 float                   vc;
 float                   dia;
 pointer_Tol             Tol;
 int                     num;
}*pointer_feat_PCir;

typedef struct cirarc
{
 pointer_feat_arc        Arc;
 pointer_feat_cir        Cir;
 struct cirarc           *next;
}*pointer_2fea_sel;

typedef struct Twofea
{
 int                     cm;
 int                     len;
 pointer_2fea_sel        CirArc;
 pointer_scrL            line;
 pointer_Tol             Tol;
 pointer_point           HeadPt;
}*pointer_feat_2fea;

typedef struct MoveProbe
{
 float                   h;
 float                   v;
 float                   d;
}*pointer_MovePr;

typedef union fea
```

```
{
  pointer_feat_cir                circle;
  pointer_feat_arc                arc;
  pointer_feat_len                len;
  pointer_feat_lentru  lentru;
  pointer_feat_wTh     wTh;
  pointer_feat_PCir    PCir;
  pointer_feat_2fea    Twofea;
  pointer_MovePr                  MovePr;
}dif_feat;

typedef enum {cir, arc_ext, arc_int, cilynder, lenin, lenout,
lentru,
              wThIn, wThOut, PCirHol, PCirBos, L2fea,
MPr}type_feat;

typedef struct feat_sel
{
  dif_feat                     point_feat;
  type_feat                       feat;
  struct feat_sel                 *previous;
  struct feat_sel                 *next;
}*pointer_feat_sel;

typedef struct selFS
{
  pointer_feat_sel                FeatSel;
  struct selFS                    *next;
}*pointer_sel_FS;

typedef struct meas_feat
{
  probe                           sensor;
  wp                              work_pl;
  float                           or_ha;
  float                           or_va;
  float                           or_dep;
  float                           IncX;
  float                           IncY;
  float                           IncZ;
  pointer_feat_sel                HeadFS;
  pointer_feat_sel                TailFS;
  pointer_line                    line;
  pointer_arc                     arc;
  struct meas_feat                *previous;
  struct meas_feat                *next;
}*pointer_meas_feat;

typedef struct real_line
{
  float                           h1;
  float                           v1;
  float                           h2;
  float                           v2;
  int                             Tol;
  float                           length;
  struct real_line                *next;
}*pointer_res_line;

typedef struct real_cir
{
  float                           h1;
  float                           v1;
  float                           h2;
  float                           v2;
  float                           hc;
  float                           vc;
  float                           dia;
  int                             TolD;
  int                             TolH;
  int                             TolV;
  struct real_cir                 *next;
}*pointer_res_cir;
```

The functions created for this program are defined in the following pages. Due to the length of the program only the header functions and some example functions will be displayed.

4.2 Read_IGES

The functions defined in this file were created to read IGES.

```
/************************** MAIN ********************************/
void main ()
{
  LastPr=p1; LastWp=xy;
  draw_enviroment ();
  initGloVar ();
  DelResCir (&HeadResCir);
  DelResLine (&HeadResLine);
}
```

```
/********************** READ_FIL *********************************/
```
This function reads an IGES file.

```
void read_fil (FILE *figes)
{
  char              caracter;
  pointer_DE        linkDE;
  int               count;

  /* The directory entry is read */

  fgets (TipoL, 82, figes);
  TipoL[80]='\0';
  caracter = TipoL[72];
  while (caracter != 'D') {
    fgets (TipoL, 82, figes);
    TipoL[80]='\0';
    caracter = TipoL[72];
  }
  while (caracter == 'D') {
    if (caracter == 'D')
      make_de_element (figes);
    fgets (TipoL,82,figes);
    TipoL[80]='\0';
    caracter = TipoL[72];
  }
  /* The parameter data is read */

  for (linkDE=head_DE; linkDE != NULL; linkDE=linkDE->next){
    for (count=0; count < link_DE->cont_PD; count++) {
      if (TypeToProcess (linkDE->type))
         add_pd_list ();
       fgets (TipoL, 82, figes);
       TipoL[80]='\0';
    }
  }
}
```

```
/********************** CREATE_STRUCTURE ***********************/
void create_structure ()
```
This function reads into the created structure what was defined by IGES.

```
/*********************** CREATE_PLINES *************************/
void create_plines ()
```
This function reads polylines.

```
/******************** TAKE_DATA_PLINE **************************/
```

void take_data_pline (pointer_PD actual_PD, int numlines, pointer_pline p_pline)
Takes the information of the polyline.

/************************ MAKE_DRAWING ***************************/
void make_drawing ()
Draws the read information on the screen.

/********************** MAKE_DE_ELEMENT *************************/

void make_de_element (FILE *figes)
Read DE information.

/********************** WRITE_PARAM1_DE_ELEMENT *****************/
void write_param1_de_element (int type, int punt_PD, state_type state, int n_line, pointer_DE element_DE)
Reads PD information.

/********************** WRITE_PARAM2_DE_ELEMENT *****************/
void write_param2_de_element (int color, int cont_PD, int form, pointer_DE element_DE)
Reads PD information.

/*************************** ADD_PD_LIST ************************/
void add_pd_list ()
Adds a PD link into the list.

/*************************** ADD_DE_LIST ************************/
void add_de_list (pointer_DE element_DE)
Adds a DE link into the list.

/******************** INITIALIZE_DE_FIELDS **********************/
void initialize_de_fields (pointer_DE element_DE)
Initialize DE fields.

/******************** INITIALIZE_PD_FIELDS **********************/
void initialize_pd_fields (pointer_PD element_PD)
Initialize PD fileds.

/******************** TAKE_DATA1_PARAM_DE ***********************/
void take_data1_param_de (int *type, int *punt_PD, state_type *state, int *n_line)
Reads DE information.

/******************** TAKE_DATA2_PARAM_DE ***********************/
void take_data2_param_de (int *color, int *cont_PD, int *form)
Reads DE information.

/************************ TAKE_DATA_LINE ************************/
void take_data_line (pointer_PD actual_PD, int numlines, pointer_line *L)
Reads the information of a line.

/************************* TAKE_DATA_ARC ************************/
void take_data_arc (pointer_PD actual_PD, int numlines, pointer_arc *A)
Reads the information of an arc.

/************************* TAKE_PARAMETER ***********************/
void take_parameter (int number)
Reads a parameter of PD.

/************************* LINK_LINES ***************************/
void link_lines (pointer_PD element_PD, int numlines)
Links PD lines.

/******************* TAKE_REAL_PARAMETER ************************/
void take_real_parameter (int i, float *number)
Reads a PD real parameter.

```
/********************* TAKE_INT_PARAMETER ************************/
void take_int_parameter (int i, int *number)
Reads an integer parameter.

/****************** TAKE_STRING_PARAMETER **********************/
void take_string_parameter (int num_char, char *name)
Reads a string of PD.

/************************** DRAW_LINES ***************************/
void draw_lines (pointer_line list_lines, int color)
Draws a list of lines.

/*************************** DRAW_ARCS ***************************/
void draw_arcs (pointer_arc list_arcs, int color)
Draws a list of arcs.

/*********************** DRAW_PLINES *****************************/
void draw_plines (pointer_pline list_plines, int color)
Draws a list of polylines.

/************************* DRAW_STRINGS *************************/
void draw_strings (pointer_string list_strings, int color)
Draws a list of strings.

/********************** INITIALIZE_LISTS ************************/
void initialize_lists ()
Initialize the auxiliar lists.

/********************* TAKE_DATA_AUX_LINE ************************/
void take_data_aux_line (pointer_PD actual_PD, int numlines)
Reads the information of an auxiliar line.

/************************ TAKE_DATA_ARROW ***********************/
void take_data_arrow (pointer_PD actual_PD, int numlines)
Reads the information of an arrow.

/************************ TAKE_DATA_STRING **********************/
void take_data_string (pointer_PD actual_PD, int numlines)
Reads the information of a string.

/************************ TAKE_DATA_HATCH ***********************/
void take_data_hatch (pointer_PD actual_PD, int numlines)
Reads the information of a hatch definition.

/******************** INITI_MAIN_STRUCT ************************/
void initi_main_struct (pointer_line line, pointer_arc arc)
Initializes the main linked lists.

/********************* INITI_AUX_STRUCT ************************/
void initi_aux_struct ()
Initializes the auxiliar linked lists.

/*********************** INITI_LIST_LINES ***********************/
void initi_list_lines (pointer_line *list_lines)
Initializes the list of lines.

/********************** INITI_LIST_ARCS ************************/
void initi_list_arcs (pointer_arc *list_arcs)
Initializes the list of arcs.

/**************************** XTOXSCR *************************/
float XtoXscr (float x)
Converts the X real into X screen.
```

```
/*************************** LINKLAUX ****************************/
void linkLaux (pointer_line NewL)
Links a line into the list.

/*************************** LINKAAUX ****************************/
void linkAaux (pointer_arc NewA)
Links an arc into the list.

/*************************** COPYAUX *****************************/
void CopyAux ()
Copies the axiliar information into other lists.

/*************************** INILISAUX ***************************/
void IniLisAux ()
Initializes the auxiliar lists.
```

4.3 Enviroment

The functions defined in this file were created to do create the enviroment.

```
/************************* DEF_BOX_FUNC **************************/
/*---------------------- constructor BOX ----------------------*/
box::box(int y, int x, int w, int h, int st, int th, int ce, int ci)

/*---------------------- constructor BOX ----------------------*/
void box::ini (int y, int x, int w, int h, int st, int th, int ce, int ci)

/*----------------------- public DRAWBOX ----------------------*/
void box::drawbox(void)

/************************ DEF_MENU_CLASS ************************/
/*---------------------- constructor MENU ---------------------*/
menu::menu(int y, int x, int wide, int hi, int Mfont, int ex_col,
           int int_col, int txt_col, char *t) :
           box(y,x, wide, hi, SOLID_LINE, 1, ex_col, int_col)

/*----------------- public_virtual PROCESS_EVENT ---------------*/
int menu::process_event ()
Process the menu depending on the mouse information.

/*------------------- public_virtual PUTSTR -------------------*/
void menu::putstr(int col)

/*------------------- public_virtual MODIFY -------------------*/
void menu::modify (int hi)

/*********************** DEF_LADDER_CLASS ***********************/
/*--------------------- constructor LADDER --------------------*/
ladder::ladder(int Lfont, int Lcoltx, int Lcoltxtit, int Lcoltxin,
               int Bcol_ext, int Bcol_int, int LIst, int LIth) :
               box(0,0,0,0, LIst, LIth, Bcol_ext, Bcol_int)

/*--------------------- public CREATE_LADDER ------------------*/
void ladder::create_ladder(int x, int y, int wid, char **p)

/*------------------- public_virtual MODIFY -------------------*/
void ladder::modify(int hi)

/*-------------------- public DRAW_LADDER ---------------------*/
void ladder::draw_ladder()

/*---------------- public_virtual PROCESS_EVENT ---------------*/
int ladder::process_event()
```

```
/*-------------------- public_virtual PUTSTR --------------------*/
void ladder::putstr(int x, int y, int col, char *str)

/************************** DRAW_ENVIROMENT ***********************/
void draw_enviroment ()
Controls the menu system.

/************************** DEFINE_BOXES *************************/
void define_boxes ()

/***************************** DEF_LADDER *************************/
void def_ladder ()

/***************************** DEF_MENU ***************************/
void def_menu ()

/************************** DETECT_PRESS *************************/
int detect_press (int ld, int rung, int mn1, int mn1Sel,int mn2, int mn2Sel)
Detects what menu the mose has pressed.

/************************ FILE_MANAGEMENT ************************/
void file_management (int rung)
Manages the files menu system.

/*************************** PROBEVALID ***************************/
int ProbeValid (int plane, int probe)
```

4.4 Create_Feature

The functions defined in this file were written to create the inspection model.

```
/************************* CREATE_FEA_ARC *************************/
void create_fea_arc (int ext)
Creates an arc feature.

/************************ CREATE_FEA_CIRC *************************/
void create_fea_circ ()

/************************ CREATE_FEA_PCIR *************************/
void create_fea_PCir (int hole)

/************************ CREATE_FEA_2FEA *************************/
void create_fea_2fea (int cm, int len)
Creates a feature between two features.

/*********************** CREATE_FEA_LEN ***************************/
void create_fea_len (int inside)

/******************** CREATE_FEA_LENTRU **************************/
void create_fea_lentru ()

/********************* ADD_POINT_ORDER **************************/
void AddPointOrder (pointer_point *head_point, pointer_point point)
Orders the points selected to measure lentru.

/********************* CREATE_WALL_THICK ************************/
void create_wall_thick (int inside)
{
  char                    text1[50];
  pointer_ent_selec       sel_1=NULL, sel_2=NULL;
  int                     found=0, go_out=0;
  pointer_arc             arc1, arc2;
  pointer_scrL            new_line;
  float                   ang;
```

```
 msm_showcursor ();
 do {
  e = eq.get();
  if (e.is() == mouse){
    strcpy (text1, "Select first arc or circle =>");
    select_arc_cir (text1, &sel_1, &found);
    if (found){
            arc1 = sel_1->arc;
            strcpy (text1, "Select second arc or circle=>");
            select_arc_cir (text1, &sel_2, &found);
    }
    if (found){
            arc2 = sel_2->arc;
            if ((arc1->xrc == arc2->xrc) && (arc1->yrc == arc2->yrc)) {
              new_line = (pointer_scrL) MY_NEW (new_line);
              new_line->next = NULL;
              ang=DrawBetwArc (arc1, arc2, new_line);
              fill_fea_wt (arc1, arc2, new_line, ang, inside);
            }
            else put_error (10);
    }
    if (e.value () == MOUSE_rightup) go_out=1;
    if (sel_1 != NULL) DelEntSel (&sel_1);
    if (sel_2 != NULL) DelEntSel (&sel_2);
  }
 }while (!go_out);
 setcolor(15);
 puttext_window (" ");
}

/************************ MOVEPROBE *****************************/

void MoveProbe ()

/************************ FILL_FEA_CIR ***************************/
void fill_fea_circle (pointer_ent_selec cir_selec, pointer_feat_cir *feat_circle, int OthFea, int ext)
Fills the information of a hole and cylinder structure.

/************************ FILL_FEA_PCIR *************************/
void fill_fea_PCir (pointer_feat_cir FeatCir, pointer_scrA arc, int cont, int hole)

/************************ FILL_FEA_LEN *************************/
void fill_fea_len (pointer_line line1,  pointer_line line2, pointer_scrL new_line, int inside)

/*********************** FILL_FEA_LENTRU ************************/
void fill_fea_lentru (pointer_point head_point)

/************************ FILL_FEA_ARC ***************************/
void fill_fea_arc (pointer_ent_selec cir, int ext)

/************************ FILL_FEA_WT ***************************/
void fill_fea_wt (pointer_arc arc1, pointer_arc arc2, pointer_scrL line, float ang, int inside)

/************************ FILL_FEA_2FEA ************************/
void fill_fea_2fea (pointer_sel_FS HeadFS, pointer_scrL line, pointer_point HeadPt, int cm, int len)

/*************************** CHANGE_WP ***************************/
void change_wp (int wp_sel)

/************************ CHANGE_PROBE *************************/
void change_probe (int prob_sel)

/************************* CSYS_INT_LINES **********************/
void csys_int_lines ()
```

Creates the coordinate system beween two lines.

```
/************************** CSYS_CEN_CIR ************************/
void csys_cen_cir ()
```
Creates the coordinat system in the centre of a circle.

```
/************************ CREA_GLO_VAR **************************/
void creaGloVar ()
```
Allocates memry and initialize global variables.

```
/************************ INIT_GLO_VAR **************************/
void initGloVar ()
```

```
/************************ LINK_FEATURE **************************/
void link_feature (pointer_feat_sel new_feat)
```

```
/************************ LINK_MEAS_FEAT ************************/
void link_meas_feat (pointer_meas_feat new_meas_feat)
```

```
/************************ MEASURE_FEATURES **********************/
void measure_features (int rung)
```
Depending on the menu clicked calls to a different function.

```
/**************************** XSCR_XR **************************/
float Xscr_Xr (float scr_x, float orH)
```
Converts screen coordinates into real coordinates.

```
/*************************** DRAWCOOR **************************/
void DrawCoor (float h, float v)
```
Draws the coordinate system.

4.5 Simulation

The functions defined in this file were written to create the simulation of the inspection model and of an existent CMES file.

```
/*************************** SIMULATION ************************/
void simul(int inside)
{
  pointer_feat_sel          FS;
  div_t                     time1, time2, timeT;
  float                     FastSpeed=10.0, SlowSpeed=3.0;
  char                      text[100];

  ActFM=TailFM;
  sim_gen_in ();
  FastSpeed=read_real ("High Speed of the probe (mm/sec) =>", 15);
  SlowSpeed=read_real ("Slow Speed of the probe (mm/sec) =>", 15);
  sprintf (text, "Fast Speed = %.1f;  Slow Speed = %.1f", FastSpeed, SlowSpeed);
  puttext_window (text);
  DistFast=0.0, DistSlow=0.0;

  if (inside) {
    for (FS=ActFM->HeadFS; FS != NULL; FS=FS->next)
      SimOneFea (FS);
    move_probeDep (lastH, lastV, 10.0);
  }
  else LoadCmesFile ();
  time1 = div (round(DistFast/FastSpeed), 60);
  time2 = div (round(DistSlow/SlowSpeed), 60);
  timeT.quot=time1.quot+time2.quot;
  timeT.rem = time1.rem + time2.rem;
  if ((timeT.rem - 60.0) >= 0.0){
    timeT.quot += 1;
    timeT.rem -= 60.0;
  }
  sprintf (text, "Time : Fast = %d min %d sec; Slow = %d min %d sec;",
                time1.quot, time1.rem, time2.quot, time2.rem);
  puttext_window (text);
  sprintf (text, "Total = %d min %d sec; ", timeT.quot, timeT.rem);
  setcolor (15);
  outtextxy (146, 457, text);
}

/*************************** SIMONEFEA *************************/
void  SimOneFea (pointer_feat_sel FS)
Calls to the respective function to simulate that feature.

/**************************** SIM_GEN_IN ************************/
void sim_gen_in ()
Initialize general information.

/************************* SIM _CIRCLE ***************************/
void sim_circle (pointer_feat_cir cir)
Simulates a hole.

/*************************** SIMPTCIR ****************************/
void SimPtCir (float h, float v, float d, int ord, int hole)
Calculates what point to measure next.

/************************** MOVENEXT *****************************/
void MoveNext (float h, float v, float d)
Move next feature.

/**************************** SIM_CILYN *************************/
void sim_cilyn (pointer_feat_cir cilyn)

/*************************** SIM_LENIN **************************/
void sim_lenin (pointer_feat_len len)

/*************************** SIM_LENOUT *************************/
void sim_lenout (pointer_feat_len LO)

/*************************** SIM_ARC ****************************/
void sim_arc (pointer_feat_arc arc, int ext)
{

  float              ang1, ang2, alfa, h, v, hf, vf, ang;
  int                i, change;

  ang1 = (float)atan2 ((double)(arc->v1-arc->vc), (double)(arc->h1-arc->hc));
  ang2 = (float)atan2 ((double)(arc->v2-arc->vc), (double)(arc->h2-arc->hc));
  if (ang1  < 0)  ang1 = (2*pi) + ang1;
  if (ang2  < 0)  ang2 = (2*pi) + ang2;
  if (ang2 <= ang1) ang2 = ang2 + (2*pi);
  alfa = (ang2 - ang1)/6;
  change=FirstArcPt (arc->h1, arc->v1, arc->h2, arc->v2);
  for (i=1; i<6; i++) {
    if (change) ang=ang2-alfa*i;
    else ang=ang1+alfa*i;
    v = arc->rad * (float)sin((double)ang) + arc->vc;
    h = arc->rad * (float)cos((double)ang) + arc->hc;
    if (ext){
      hf = (arc->rad+depth)*(float)cos((double)ang) + arc->hc;
      vf = (arc->rad+depth)*(float)sin((double)ang) + arc->vc;
    }
    else{
      hf = (arc->rad-depth)*(float)cos((double)ang) + arc->hc;
      vf = (arc->rad-depth)*(float)sin((double)ang) + arc->vc;
    }
    if (i==1) MoveNext (hf, vf, arc->dep-depth);
    else move_probe (hf, vf, 1);
    move_probe (h, v, 1);
  }
  move_probe (hf, vf, 1);
}

/************************** SIM_LENTRU **************************/
void sim_lentru (pointer_feat_lentru lentru)

/*************************** SIM_WTHIN **************************/
void sim_wThIn (pointer_feat_wTh wThIn)

/*************************** SIM_WTHOUT *************************/
void sim_wThOut (pointer_feat_wTh wThOut)

/************************** SIM_PCIRHOL *************************/
void sim_PCirHol (pointer_feat_PCir PCir)

/************************** SIM_PCIRBOS *************************/
void sim_PCirBos (pointer_feat_PCir PCir)

/************************** SIM_2FEA ***************************/
void sim_2fea (pointer_feat_2fea TwoFea)

/*************************** SIM_MOVEPR *************************/
void sim_MovePr (pointer_MovePr  MPr)

/**************************** MOVE_PROBE ************************/
void move_probe (float hor, float ver, int slow)
Moves the ball which simulates the CMM probe.

/************************** MOVE_PROBEDEP ***********************/
void move_probeDep (float hor, float ver, float dep)
Simulates the probe in a perpendicular plane.

/**************************** SAVE_SCREEN ***********************/
void *save_screen (float h1, float v1, float h2, float v2)

/**************************** REST_SCREEN ***********************/
void rest_screen (float hor1, float ver1, void *screen)

/**************************** DRAW_PROBE ************************/
void draw_probe (float hor, float ver, int color)

/*************************** CHANGE_AXEH ************************/
void change_axeH (float hor)
Calculates the horizontal coordinate of the probe and change its value on the screen.
/*************************** CHANGE_AXEV ************************/
void change_axeV (float ver)

/*************************** CHANGE_AXED ************************/
void change_axeD (float dep)

/**************************** DETCRASH **************************/
void DetCrash (float h2, float v2, int simul)
{
  pointer_line             L, line;
  float                    h1, v1, h, v, d1, orH, orV;
  int                      inter=0;
  pointer_arc              A, arc;

  orH=ActFM->or_ha;
```

```
 orV=ActFM->or_va;
 h1=lastH; v1=lastV; d1=lastD;
 line=ActFM->line;
 arc=ActFM->arc;
 for (L=line; L != NULL; L=L->next) {
   inter=Inter2Lines (h1, v1, h2, v2, L->xr1-orH, L->yr1-orV,
                              L->xr2-orH, L->yr2-orV, &h, &v);
   if (inter) inter=PtInSeg (h1, v1, h2, v2, h, v);
   if (inter) inter=PtInSeg (L->xr1-orH, L->yr1-orV, L->xr2-orH,
                                       L->yr2-orV, h, v);
   if (inter){
     if (d1 < (L->zr1+5))  d1=L->zr1+5;
   }
 }
 for (A=arc; A != NULL; A=A->next) {
   inter=InterLineArc (h1, v1, h2, v2, A);
   if (inter){
     if (d1 < (A->z+5)) d1=A->z+5;
   }
 }
 if (d1 != lastD) {
   switch (simul) {
     case 0 : MoPrD (d1);
              break;
     case 1 : move_probeDep (lastH, lastV, d1);
              break;
     case 2 : GotoD (d1);
              break;
   }
 }
}

/************************** LOADCMESFILE *************************/
void LoadCmesFile ()

/************************* READCMESFILE **************************/
void ReadCmesFile (FILE *CmesIn)

/**************************** GETAXIS ****************************/
void GetAxis (char ax, int *ax_ord)

/************************ CHECKEXTENSION ************************/
int CheckExtension (char *text, char *ext)
```

4.6 Operations

The functions defined here are auxiliary functions.

```
/*************************** GET_SIGN ****************************/
int  get_sign (float number)

/************************ ABSOLUT_IQUAL *************************/
int absolut_iqual (float a, float b, float epsi)

/************************ NOR_LINE_COEF *************************/
int nor_line_coef (float *a, float *b, float *c)

/*********************** LINE_POINT_POINT ***********************/
void line_point_point (float x1, float y1, float x2, float y2, float *a, float *b, float *c)

/************************ POLAR_TO_CART ************************/
void pol_to_cart (float xpolar, float ypolar, float pol_rad, float ang, float *x, float *y)
```

```
/****************************** EXIST *****************************/
int exist (char *fil_name)

/********************** PUTTEXT_WINDOW *************************/
int puttext_window (char *text)

/*************************** PUT_ERROR **************************/
void put_error (int num)

/****************************** BEEP *****************************/
void beep (int times)

/***************************** WAIT *****************************/
void wait (int seconds)

/*************************** DRAW_ARC ***************************/
void draw_arc (float xc, float yc, float x1, float y1, float x2, float y2)

/*************************** DRAW_LINE **************************/
void draw_line (float x1, float y1, float x2, float y2)

/************************** DRAW_POINT **************************/
void draw_point (float h, float v)

/********************** DRAWLINE_BETW ***************************/
void drawline_betw (pointer_line L1, pointer_line L2, pointer_scrL L)
Draws a line between two lines.

/************************ DRAW_PCIR ****************************/
int DrawPCir (pointer_feat_cir  FeatCir, pointer_scrA *arc)

/**************************** SQR ********************************/
float sqr (float num)

/********************* DRAW_BETW_ARC ***************************/
float DrawBetwArc (pointer_arc A1, pointer_arc A2, pointer_scrL line)

/************************ READ_STRING *************************/
char *read_string (int x, int y, int color)
{
  char                    ch, text[20];
  int                     i;

  ch=getch();
  text[0]='\0';
  for (i=0;ch!='\r';i++) {
    if (ch=='\b'){
      if (i > 0) {
              setcolor(1); outtextxy (x, y, text);
              i=i-2; text[i+1]='\0';
      }else i--;
    }
    else{
      text[i]=ch;
      text[i+1]='\0';
    }
    setcolor (color);
    if (text[0] != '\0') outtextxy (x, y, text);
    ch=getch();
  }
  return text;
}

/************************** READ_REAL *************************/
```

float read_real (char *message, int color)

/**************************** SAME_ARC ***************************/
int same_arc (pointer_arc arc1, pointer_scrA arc2)

/************************** LINETHRARC ***************************/
void LineThrArc (pointer_SelArc HeadArc, pointer_scrL *HeadLine, int cm, int len)

/**************************** MIDPOINT ***************************/
void MidPoint (pointer_scrL HeadLine, pointer_point *HeadPt)

/**************************** INTER2LINES ************************/
int Inter2Lines (float x1, float y1, float x2, float y2, float x3, float y3, float x4, float y4, float *x, float *y)

/************************** INTERLINEARC ************************/
int InterLineArc (float x1, float y1, float x2, float y2, pointer_arc arc)

/************************** DELENTSEL ***************************/
void DelEntSel (pointer_ent_selec *ESel)

/**************************** PTINSEG ***************************/
int PtInSeg (float Lx1, float Ly1, float Lx2, float Ly2, float x1, float y1)

/****************************** ROUND *****************************/
int round (float num)

/**************************** CLEARSCREEN ************************/
void ClearScreen ()

/*************************** POINTINARC *************************/
int PointInArc (float x1, float y1, pointer_arc arc)

/*************************** FIRSTCIRPT *************************/
int FirstCirPt (pointer_feat_cir cir)

/*************************** FIRSTCIRPT *************************/
float DistTwoPt (float h1, float v1, float h2, float v2)

/*************************** FIRSTLENPT *************************/
int FirstLenPt (float *h1, float *v1, float *h2, float *v2)

/*************************** FIRSTARCPT *************************/
int FirstArcPt (float h1, float v1, float h2, float v2)

4.7 Selection

The function defined here were written to select the different entities.

/*********************** SELECT_ONE_ENTITY *********************/
void select_one_entity (float x1, float y1, pointer_ent_selec *EntSel)

/*********************** SELECT_ONE_LINE ***********************/
void select_one_line (char *text, pointer_ent_selec *EntSel, int *found)

/************************ SELECT_ONE_ARC ***********************/
void select_one_arc (char *text, pointer_ent_selec *EntSel,int *found)

/************************ SELECT_ARC_CIR ***********************/
void select_arc_cir (char *text, pointer_ent_selec *EntSel,int *found)

/********************** SELECT_ONE_CIRCLE **********************/
void select_one_circle (char *text, pointer_ent_selec *EntSel,int *found)

```
/************************ LOOK_FOR_LINE **************************/
void look_for_line (float x1, float y1, pointer_line list_lines, pointer_ent_selec EntSel, int *found, float *dist_min)

/************************ DIST_LINE_POINT ***********************/
float dist_line_point (float x1, float y1, pointer_line line,int *valid)

/************************ LOOK_FOR_ARC **************************/
void look_for_arc (float x1, float y1, pointer_arc list_arcs, pointer_ent_selec EntSel, int *found, float *dist_min)

/************************ DIST_ARC_POINT ************************/
float dist_arc_point (float x1, float y1, pointer_arc arc)

/************************ LOOK_FOR_PLINE ************************/
void look_for_pline (float x1, float y1, pointer_ent_selec EntSel, int *found, float *dist_min)

/************************ SELECT_ENTITY *************************/
void select_entity (pointer_ent_selec *head_ent_selec)
{
 int                        x1, y1, go_out=0;
 float                      x2, y2;
 pointer_ent_selec          EntSel=NULL;

 msm_showcursor ();
 do {
   e = eq.get();
   if (e.is() == mouse){
     if (e.value ()== MOUSE_leftup) {
            x1=e.x(); y1=e.y();
            x2 = (float) x1;
            y2 = (float) y1;
            select_one_entity (x2, y2, &EntSel);
            if (EntSel != NULL) {
              if (*head_ent_selec == NULL)
                *head_ent_selec = EntSel;
              else {
                EntSel->next = *head_ent_selec;
                *head_ent_selec = EntSel;
              }
            }
     }if (e.value () == MOUSE_rightup) go_out=1;
   }
  }while (!go_out);
  msm_hidecursor ();
}

/********************** SELECT_ONE_POINT ***********************/
void select_one_point (float *h, float *v, int *end)

/********************* LOOK_FOR_FEATURE ************************/
pointer_feat_sel look_for_feature (float h, float v, int *go_out, int col, int OnArc)

/************************ LOOK_LINE_FEA *************************/
int look_line_fea (float h, float v, pointer_scrL scr_line, int col)

/********************** LOOK_ARCFEA ****************************/
int look_arc_fea (float h, float v, pointer_scrA scr_arc, int col)

/************************** DEL_A_FEA **************************/
void del_a_fea (pointer_feat_sel  *FeatSel)

/************************** UNDELETE ***************************/
void undelete ()

/************************* DELLENTRU ***************************/
```

```
void DelLentru (pointer_feat_lentru lentru)

/***************************** DELLISTL *************************/
void DelListL (pointer_scrL *HeadL)

/***************************** DELLISTPT ************************/
void DelListPt (pointer_point *HeadPt, int draw)

/*************************** DELCIRLIST *************************/
void DelCirList (pointer_feat_cir FCir)

/************************* UNLINK_FEATURE ***********************/
void unlink_feature (pointer_feat_sel  FeatSel)

/************************* DELETE_FEATURE ***********************/
void delete_feature ()

/*************************** SELANYFEA **************************/
int SelAnyFea (pointer_feat_sel *FeatSel, int col, int *hi, int *vi)

/************************* SELECT_ARC_FEA ***********************/
pointer_feat_sel select_arc_fea (pointer_scrA *arc)

/**************************** REDRAW ****************************/
void redraw ()

/**************************** DETTYPE ***************************/
int DetType (type_feat type)

/************************** DETARCLINE **************************/
void DetArcLine (pointer_feat_sel FS, pointer_scrA *A, pointer_scrL *L, pointer_point *P)

/***************************** DETTOL ***************************/
void DetTol (pointer_feat_sel FS, pointer_Tol *Tol)
```

4.8 CMES_Out

The functions written here were created to write CMES programs.

```
/************************** PRINT_LK_OUT ************************/
void print_lk_out ()
{
  char           *fil_name, text[60];
  int            x, type;
  float          x1=0.0, y1=0.0, z1=0.0;

  strcpy (text, "Introduce the name of the LK file =>");
  x=puttext_window (text);
  strcpy (fil_name, read_string (x, 445, 15));
  lk_fil = fopen (fil_name, "w");
  strcpy (text, "Prototype program (Y/N) =>");
  prot=ControlYes (text);
  SelLkSensor (HeadFM->sensor);
  for (ActFM=HeadFM; ActFM != NULL; ActFM=ActFM->next) {
    pr_gen_in (x1, y1, z1);
    for (FS=ActFM->HeadFS; FS != NULL; FS=FS->next)
      PrOneFea ();
    if (ActFM->next != NULL)
      MoveNextFace (&x1, &y1, &z1,0);
  }
  fprintf (lk_fil, "%s\n", ":END");
  fprintf (lk_fil, "%s\n", "ET");
  fclose (lk_fil);
```

```
 puttext_window (" ");
}

/*************************** PRONEFEA ***************************/
void PrOneFea ()

/*************************** PR_GEN_IN ***************************/
void pr_gen_in (float x, float y, float z)

/************************* SEL_LK_SENSOR ************************/
void SelLkSensor (probe sensor)

/*************************** PR_A_HOLE ***************************/
void pr_a_hole (pointer_feat_cir circle)

/**************************** NOMEAS ****************************/
void NoMeas ()

/***************************** MEAS *****************************/
void Meas (float h, float v, float d)

/*************************** PR_A_CYL ***************************/
void pr_a_cyl (pointer_feat_cir cilyn)

/*************************** PR_AN_ARC ***************************/
void pr_an_arc (pointer_feat_arc arc, int ext)

/*************************** MOVE_AXES ***************************/
void move_axes (int hor_ax, int ver_ax, int dep_ax)

/**************************** PR_POINT ***************************/
void pr_point (float hor_ax, float ver_ax, int num_pt)

/**************************** PR_LENIN ***************************/
void pr_lenin (pointer_feat_len lenin)

/*************************** PR_LENOUT ***************************/
void pr_lenout (pointer_feat_len lenout)

/*************************** PR_LENTRU ***************************/
void pr_lentru (pointer_feat_lentru lentru)

/**************************** PR_WTHIN ***************************/
void pr_wThIn (pointer_feat_wTh wThIn)

/*************************** PR_WTHOUT ***************************/
void pr_wThOut (pointer_feat_wTh wThOut)

/*************************** PR_A_PCHOL *************************/
void prA_PCHol (pointer_feat_cir circle, int num_pt)

/***************************** PR_PCIR ***************************/
void pr_PCir (pointer_feat_PCir PCir, int hole)
{
  pointer_feat_cir              FCir;
  int                           cont=4;
  char                          line[20], str[5];

  for (FCir=PCir->list_circle; FCir != NULL; FCir=FCir->next){
    cont++;
    if (FCir->next == NULL) last=1;
    if (hole) prA_PCHol (FCir, cont);
    else prA_PCBos (FCir, cont);
  }
 last=0;
 if (prot){
  fprintf (lk_fil, ":ASK_%d\n", FeaNum);
  fprintf (lk_fil, "%s\n", "IP,'Pitch Circle?',F1");
  if (FS->next==NULL)
    fprintf (lk_fil, "IF,F1,ASK_%d,END,MEAS_%d\n", FeaNum,FeaNum);
  else
    fprintf (lk_fil, "IF,F1,ASK_%d,MOVE_NEXT_%d,MEAS_%d\n", FeaNum,FeaNum+1,FeaNum);
  fprintf (lk_fil, ":MEAS_%d\n", FeaNum);
  FeaNum++;
 }
 strcpy (line, "UP");
 for (cont=5; cont<(PCir->num+5); cont++){
   sprintf (str, ",%d\0", cont);
   strcat (line, str);
 }
 fprintf (lk_fil, "%s\n", line);
 sprintf (line, "PC,%d,%d", dep, PCir->num);
 AddTol (PCir->Tol, line);
 if (PCir->Tol != NULL)
   PrTolArc (PCir->hc, PCir->vc, PCir->dia, PCir->Tol);
}

/**************************** PRA_PCBOS **************************/
void prA_PCBos (pointer_feat_cir cilyn, int num_pt)

/**************************** PR_2FEA ****************************/
void pr_2fea (pointer_feat_2fea TwoFea)

/*************************** PR_MOVEPR **************************/
void pr_MovePr (pointer_MovePr MPr)

/**************************** PR_COOR ****************************/
void PrCoor (float num)

/**************************** MOPROBHV **************************/
void MoPrHV (float h, float v)

/**************************** MOPROBD ****************************/
void MoPrD (float d)

/***************************** SAVEPT ****************************/
void SavePt (int pt)

/*************************** PRTOLARC ***************************/
void PrTolArc (float hor, float ver, float dia, pointer_Tol Tol)

/************************** PRTOLLINE ****************************/
void PrTolLine (float hor, float ver, pointer_Tol Tol)

/*************************** PRTOLONE ****************************/
void PrTolOne (float num, float Up, float Low, int equal)

/*************************** DetDHDV *****************************/
void DetDHDV (pointer_2fea_sel CirArc1, pointer_2fea_sel CirArc2, float *h, float *v, float *hc, float *vc)

/************************** ADD_TOL *****************************/
void AddTol (pointer_Tol Tol, char *line)
```

4.9 Tolerances

The functions defined here were written to create the tolerances.

```
/*************************** TOLERANCES **************************/
void tolerances ()
{
 int                    found, go_out=0, h, v, type;
 pointer_feat_sel       FeatSel=NULL;
 pointer_scrL           line=NULL, L;
 pointer_scrA           A=NULL;
 pointer_point          pt=NULL;

 msm_showcursor ();
 do {
   e = eq.get();
   if (e.is() == mouse){
     found=SelAnyFea (&FeatSel, MAGENTA, &h, &v);
     if (found) {
            type=DetType(FeatSel->feat);
            DetArcLine (FeatSel, &A, &line, &pt);
            if (A != NULL) {
              Tol (&FeatSel, h, v, 1);
              setcolor (LIGHTCYAN);
              draw_arc (A->xc, A->yc, A->x1, A->y1, A->x2, A->y2);
            }
            if (line != NULL) {
              TolOne (&FeatSel, h, v, type);
              setcolor (LIGHTCYAN);
              for (L=line; L != NULL; L=L->next)
              draw_line (L->x1, L->y1, L->x2, L->y2);
            }
     }
     if (e.value () == MOUSE_rightup) go_out=1;
   }
 }while (!go_out);
 puttext_window (" ");
}

/******************************** TOL ****************************/
void Tol (pointer_feat_sel *FeatSel, int h, int v, int arc)

/*************************** CONTROLYES **************************/
int ControlYes (char *str)

/***************************** DETHOR ****************************/
char DetHor ()

/*************************** INITOLARC **************************/
void IniTol  (pointer_Tol          *PTol)

/***************************** DRAWTOL **************************/
void DrawTol (int col, pointer_Tol PTol)

/************************* DRAWFRAME ****************************/
void DrawFrame (int h, int v, int num)

/************************* DRAWONETOL ***************************/
void DrawOneTol (int h, int v, float Low, float Up, char *tol)

/*************************** DELTOLARC *************************/
void DelTol (pointer_Tol  *Tol)

/***************************** ASIGNTOL *************************/
void AsignTol (pointer_feat_sel *FeatSel, pointer_Tol PTol)

/*************************** TOLONE ****************************/
void TolOne (pointer_feat_sel *FeatSel, int h, int v, int type)
```

4.10 Fast_Way

The functions defined here were written to calculate the measurement sequence.

```
/**************************** FASTWAY ****************************/
void FastWay ()
{
 float                    dist, ShortDist, FeaLastX, FeaLastY, x, y;
 pointer_feat_sel         FS, SelFS, FirstFS=NULL, LastFS=NULL;
 int                      NumFea=0, i;

 LastX = Xr_Xscr (0.0);  LastY = Yr_Yscr (0.0);
 for (FS=TailFM->HeadFS; FS != NULL; FS=FS->next)
   NumFea ++;
 for (i=0; i < NumFea; i++) {
   FS=TailFM->HeadFS;
   ShortDist=DistFea (FS, &FeaLastX, &FeaLastY);
   SelFS=FS;
   x=FeaLastX; y=FeaLastY;
   for (FS=FS->next; FS != NULL; FS=FS->next){
     dist=DistFea (FS, &FeaLastX, &FeaLastY);
     if (dist < ShortDist)  {
            ShortDist=dist;
            SelFS=FS;
            x=FeaLastX; y=FeaLastY;
     }
   }
   unlink_feature (SelFS);
   SelFS->next=NULL;
   SelFS->previous = NULL;
   if (FirstFS == NULL) {
     FirstFS = SelFS;
     LastFS = SelFS;
   }
   else {
     LastFS->next = SelFS;
     SelFS->previous = LastFS;
     LastFS = SelFS;
   }
   LastX=x; LastY=y;
 }
 TailFM->HeadFS = FirstFS;
 TailFM->TailFS = LastFS;
}

/**************************** DISTFEA ****************************/
float DistFea (pointer_feat_sel FS, float *x, float *y)

/**************************** DISTCIRCLE **************************/
float  DistCircle (pointer_scrA cir, float *x, float *y)

/***************************** DISTARC ***************************/
float  DistArc (pointer_scrA arc, float *x, float *y)

/**************************** DISTLINE ***************************/
float  DistLine (pointer_scrL line, float *x, float *y)

/**************************** DISTPCIR ***************************/
float DistPCir (pointer_feat_PCir PCir, float *x, float *y)

/**************************** DISTTWOFEA ************************/
float DistTwoFea (pointer_feat_2fea TwoFea, float *x, float *y)
```

4.11 DMIS_Output

The functions defined here were written to create the DMIS inspection programs.

```
/*********************** PRINT_DMIS_OUT ************************/
void PrDmis ()
{
 pointer_meas_feat          NowFM;
 char                       fil_name[15], text[105];
 int                        x;
 float                      x1=0.0, y1=0.0, z1=0.0;
 FILE                       *set;

 lent=0; Dist2F=0; Mid2F=0; hole=0; cyl=0; ArcE=0; ArcI=0; WI=0;
 WO=0; PCirH=0; PCirB=0; LenI=0; LenO=0; Point=0, FeaNum=1, last=0;

 strcpy (text, "Introduce the name of the DMIS file =>");
 x=puttext_window (text);
 strcpy (fil_name, read_string (x, 445, 15));

 dmis = fopen (fil_name, "w");
 strcpy (text, "Prototype program (Y/N) =>");
 prot=ControlYes (text);

 set = fopen ("dmis.set", "r");
 do {
   fgets (text, 100, set);
   fputs(text,dmis);
 } while (!feof(set));
 fclose(set);
 if (prot) {
   fprintf (dmis, "%s\n", "DECL/CHAR,ans");
   fprintf (dmis, "%s\n", "DECL/INTGR,F1");
 }
 lastH=0.0; lastV=0.0; lastD=0.0;
 SelectSensor(HeadFM->sensor);
 Macro();
 for (ActFM=HeadFM; ActFM != NULL; ActFM=ActFM->next) {
   PrDmisGen (x1, y1, z1);
   for (FS=ActFM->HeadFS; FS != NULL; FS=FS->next)
     PrDmisOneFea ();
   if (ActFM->next != NULL)
     MoveNextFace (&x1, &y1, &z1,1);
   else GotoD (10.0);
 }
 fprintf (dmis, "%s\n", "(END)");
 fprintf (dmis, "%s\n", "ENDFIL");
 fclose (dmis);
 puttext_window (" ");
}

/************************** PRDMISONEFEA ************************/
void PrDmisOneFea ()

/*************************** PRDMISGEN **************************/
void PrDmisGen (float x, float y, float z)

/*********************** TRANS_COOR_SYS ************************/
void TransCoorSys (float x, float y, float z)
{
  char                 line[100], txt[50];

  if (ActFM != HeadFM){
    if ((ActFM->IncX != 0.0) || (ActFM->IncY != 0.0) || (ActFM->IncZ != 0.0))
```

```
      fprintf (dmis, "%s\n", "RECALL/D(part)");
    if (ActFM->IncX != 0.0) {
      sprintf (line, "D(NEW_DAT)=TRANS/XORIG,%.3f", ActFM->IncX);
      x=x-ActFM->IncX;
      if ((ActFM->IncY != 0.0) || (ActFM->IncZ != 0.0)) strcat (line,",");
    }
    if (ActFM->IncY != 0.0) {
      sprintf (txt, "YORIG,%.3f", ActFM->IncY);
      strcat (line, txt);
      y=y-ActFM->IncX;
      if (ActFM->IncZ != 0.0) strcat (line,",");
    }
    if (ActFM->IncZ != 0.0) {
      sprintf (txt, "ZORIG,%.3f", ActFM->IncZ);
      strcat (line, txt);
      z=z-ActFM->IncX;
    }
    fprintf (dmis, "%s\n", line);
  }
  AssignHVD (x, y, z, &lastH, &lastV, &lastD);
}

/************************* SELECT_SENSOR *************************/
void SelectSensor (probe sensor)

/************************** MOVENEXTFACE ************************/
void MoveNextFace (float *x, float *y, float *z, int DM)
{
  float               h, v, d;

  AsignAxes (lastH, lastV, lastD, x, y, z);

  if (ActFM->next->work_pl == xy)
      *z =  ActFM->next->IncZ + 10;
  if (ActFM->next->work_pl == yz){
    if (ActFM->next->sensor == p2)
      *x = ActFM->next->IncX - 10;
    else
      *x = ActFM->next->IncX + 10;
  }
  if (ActFM->next->work_pl == xz)  {
    if (ActFM->next->sensor == p3)
      *y = ActFM->next->IncY - 10;
    else
      *y = ActFM->next->IncY + 10;
  }
  AssignHVD (*x, *y, *z, &h, &v, &d);
  if (DM) {
    DetCrash (h, v, 2);
    SelectSensor (ActFM->next->sensor);
    GotoHV (h, v);
  }
  else  {
    DetCrash (h, v, 0);
    SelLkSensor (ActFM->next->sensor);
    MoPrHV (h, v);
  }
}

/************************** PRDMISHOLE *************************/
void PrDmisHole (pointer_feat_cir cir)

/************************** PRDMISCYL ***************************/
void PrDmisCyl (pointer_feat_cir cylin)
```

```
/************************* PRDMISARC ***************************/
void PrDmisArc (pointer_feat_arc arc, int ext)
{
 char            line[100], str[10], label[12];
 float           ang1, ang2, ang, h, v, hf, alfa, vf, x, y, z;
 int             i, hi=0, vi=0, change;

 ang1 = (float) atan2 ((double)(arc->v1-arc->vc), (double)(arc->h1-arc->hc));
 ang2 = (float) atan2 ((double)(arc->v2-arc->vc), (double)(arc->h2-arc->hc));
 if (ang2 <= ang1) ang2 = ang2 + (2*pi);
 if (ext) {
   strcpy (str, "OUTER");
   ArcE++;
   sprintf (label, "ArcExt%d", ArcE);
 }
 else {
   strcpy (str, "INNER");
   ArcI++;
   sprintf (label, "ArcInt%d", ArcI);
 }
 AsignAxes (arc->hc, arc->vc, arc->dep, &x, &y, &z);
 sprintf (line, "F(%s)=FEAT/ARC,%s,CART,%.3f,%.3f,%.3f,%d,%d,%d,%.3f,%.3f,%.3f",
              label, str, x, y, z, vec[0], vec[1], vec[2],
              arc->rad, ang1, ang2);
 fprintf (dmis, "%s\n", line);
 sprintf (line, "MEAS/ARC,F(%s),5", label);
 fprintf (dmis, "%s\n", line);

 alfa = (ang2 - ang1)/6;
 change=FirstArcPt (arc->h1, arc->v1, arc->h2, arc->v2);
 for (i=1; i<6; i++) {
   if (change) ang=ang2-alfa*i;
   else ang=ang1+alfa*i;
   v = arc->rad * (float)sin((double)ang) + arc->vc;
   h = arc->rad * (float)cos((double)ang) + arc->hc;
   if (ext){
     hf = (arc->rad+apr)*(float)cos((double)ang) + arc->hc;
     vf = (arc->rad+apr)*(float)sin((double)ang) + arc->vc;
   }
   else{
     hf = (arc->rad-apr)*(float)cos((double)ang) + arc->hc;
     vf = (arc->rad-apr)*(float)sin((double)ang) + arc->vc;
   }
   if (i == 1) {
     if (prot) fprintf (dmis, "(MOVE_NEXT_%d)\n", FeaNum);
     DetCrash (hf, vf, 2);
     GotoHV (hf, vf);
     MoPrHV (hf, vf);
     if (prot){
          fprintf (dmis, "(ASK_%d)\n", FeaNum);
          if (ext) fprintf (dmis, "%s\n", "TEXT/QUERY,(R),A,L,'Is this a partial cylinder?'");
          else fprintf (dmis, "%s\n", "TEXT/QUERY,(R),A,L,'Is this a partial hole?'");
          fprintf (dmis, "%s\n", "READ/1,ans");
          fprintf (dmis, "%s\n", "CALL/M(check),ans");
          DmMeas(h,v,lastD);
     }
     if (lastD > arc->dep-depth) GotoD (arc->dep-depth);
     GotoD (arc->dep-depth);
   }
   else GotoHV (hf, vf);
   hi = 0;  vi=0;
   hi = (h < 0.0) ? -1:1;
   vi = (v < 0.0) ? -1:1;
   PtMeas (0, h, v, lastD, hi, vi, 0);
```

```
    GotoHV (hf, vf);
  }
  fputs("ENDMES\n", dmis);
  fputs(" \n", dmis);
  OutputFea (label);
  if (arc->Tol != NULL) {
    EvalTolHV (arc->Tol, arc->hc, arc->vc, label);
    EvalTolD (arc->Tol, label);
    fputs(" \n", dmis);
  }
  if (prot) DmNoMeas();
}

/**************************** PRDMISLENIN ************************/
void PrDmisLenin (pointer_feat_len lenin)

/**************************** PRDMISLENOUT ***********************/
void PrDmisLenout (pointer_feat_len lenout)

/************************ TAKE_POINT_LEN ************************/
void TakePtLen (float h, float v, float d, float hf, float vf, float df, int hi, int vi, int di, int slot)

/************************** PRDMISLENTRU ************************/
void PrDmisLentru (pointer_feat_lentru lentru)

/************************* PRDMISWIN *****************************/
void PrDmisWIn (pointer_feat_wTh wThIn)

/************************** PRDMISWOUT **************************/
void PrDmisWOut (pointer_feat_wTh wThOut)

/************************** PRDMISPCIR **************************/
void PrDmisPCir (pointer_feat_PCir PCir, int hol)

/************************* PRDMIS2FEA ***************************/
void PrDmis2Fea (pointer_feat_2fea TwoFea)

/**************************** PRCIRARC **************************/
void PrCirArc (pointer_2fea_sel CirArc, float *h, float *v, float *d, char *name)

/************************** PRDMISMOVEPR ************************/
void PrDmisMovePr (pointer_MovePr MPr)

/****************************** GOTOHV **************************/
void GotoHV (float h, float v)

/****************************** GOTOD ****************************/
void GotoD (float d)

/****************************** PTMEAS **************************/
void PtMeas (int def, float ha, float va, float da, int h, int v , int d)

/*************************** ASIGNAXES **************************/
void AsignAxes (float h, float v, float d, float *x, float *y, float *z)

/**************************** EVALTOLD **************************/
void EvalTolD (pointer_Tol Tol, char *fea)

/*************************** EVALTOLHV **************************/
void EvalTolHV (pointer_Tol Tol, float h, float v, char *fea)

/************************ EVALTOLDEP ****************************/
void EvalTolDep (pointer_Tol Tol, float dist, char *fea)
```

```
/************************* EVALTOLDIST ***************************/
void EvalTolDist (pointer_Tol Tol, float dist, int DistH, char *Name)

/********************** EVAL_TOL_DIST_2PT **********************/
void EvalTolDist2Pt (pointer_Tol Tol, float dist, char *str, char *label1, char *label2)
{
  char                    line[100];

  sprintf (line, "T(Tol%s)=TOL/DISTB,NOMINL,%.3f,%.3f,%.3f,PT2PT", str,
               dist, Tol->LowH, Tol->UpH);
  fprintf (dmis, "%s\n", line);
  sprintf (line, "EVAL/FA(%s),FA(%s),TA(Tol%s)", label1, label2, str);
  fprintf (dmis, "%s\n", line);
  sprintf (line, "OUTPUT/TA(Tol%s)", str);
  fprintf (dmis, "%s\n", line);
}

/************************** ASSIGNHVD ****************************/
void AssignHVD (float x, float y, float z, float *h, float *v, float *d)

/************************** PR_FEA_LINE *************************/
void PrFeaLine (char *label, float x1, float y1, float z1, float x2, float y2, float z2)

/************************ PR_CONST_LINE *************************/
void PrConstLine (char *label, char *label1, char *label2)

/************************** OUTPUT_FEA ***************************/
void OutputFea (char *label)

/***************************** MEAS *****************************/
void DmMeas (float h, float v, float d)

/************************** DMNOMEAS *****************************/
void DmNoMeas ()

/***************************** MACRO ****************************/
void Macro ()
```

4.12 DMIS_Output

The functions defined here were written to read a DMIS output file.

```
/************************** LOADDMISFILE ************************/
void LoadDmisFile ()

/************************* READDMISFILE *************************/
void ReadDmisFile (FILE *DmisIn)

/************************ ANALIZEFEATURE ***********************/
void AnalizeFeature ()

/***************************** TAKEFEA **************************/
void TakeFea (char *name)

/*********************** ANALIZETOLERANCE **********************/
void AnalizeTolerance ()

/*************************** TAKE_STRING *************************/
void TakeString (char *name)

/*********************** READRESULTLINE ************************/
void ReadResultLine ()
```

```
/************************* READRESULTCIR *************************/
void ReadResultCir ()

/************************* READRESULTARC *************************/
void ReadResultArc ()

/************************** READTOLDIST **************************/
void ReadTolDist ()

/************************** READTOLCOR ***************************/
void ReadTolCor()

/************************** READTOLDIA ***************************/
void ReadTolDia ()

/************************ CHANGEVALUES ***************************/
void ChangeValues (float x, float y, float z, float *h, float *v)

/************************ LINK_RES_LINE **************************/
void LinkResLine (pointer_res_line L)

/************************ LINK_RES_CIR ***************************/
void LinkResCir (pointer_res_cir circle)

/************************** SETFEACOL ****************************/
void SetFeaCol (int state)

/*************************** DELRESLINE **************************/
void DelResLine (pointer_res_line  *HeadResLine)

/*************************** DELRESCIR  **************************/
void DelResCir (pointer_res_cir *HeadResCir)

/************************** DRAWRESLINE **************************/
void DrawResLine (pointer_res_line  Rline)

/************************** DRAWRESCIR ***************************/
void DrawResCir (pointer_res_cir  cir)

/**************************** TEXTXY *****************************/
void TextXY (pointer_res_line line, float *xc, float *yc)
```

4.13 IGES_Output

The functions defined here were written to create an IGES file.

```
/**************************** IGESOUT ****************************/
void IgesOut ()
{
  int              x;
  char             fil_name[50],record[20];

  dnum=0;
  pnum=0;
  x=puttext_window ("Introduce the name of the IGES file =>");
  strcpy (fil_name, read_string (x, 445, 15));
  igesout= fopen (fil_name, "w");
  WriteHeader ();
  GoThrough (1);
  pnum=1;
  dnum=1;
  GoThrough (0);
  WriteTerminate ();
  fclose (igesout);
  settextjustify(LEFT_TEXT, TOP_TEXT);
}

/*************************** ADDINTDE ***************************/
void AddIntDE (int param)

/*************************** ADDINTPD ***************************/
void AddIntPD (int param)
/*************************** ADDREALPD ***************************/
void AddRealPD (float param)

/***************************** ITOSK *****************************/
void  itosk (int n, char *line, int k)

/*********************** COMPLETLINE *****************************/
void CompleteLine (int n,char section)
{
  int                     i,j;

  for (i=0;TipoL[i] != '\0'; i++);
  for (j=i;j<80;j++) {
    if (j==72) TipoL[72]=section;
    else TipoL[j]=' ';
  }
  i=79;
  do{
   TipoL[i--]=n %10 + '0';
  } while ((n/=10) > 0);
  for (j=i; j>72;j--) TipoL[j]='0';
  TipoL[80]='\0';
}

/********************** ADDPOINTDEINPD ***************************/
void AddPointDEinPD (int num)
{
  int                     i,j;
  char                    chain[30];

  for (i=0; TipoL[i] != '\0'; i++);
  TipoL [i-1]=';';
  for (j=i;j<64;j++) TipoL[j]= ' ';
  TipoL[j]='\0';
  chain[0]='\0';
  itosk (num,chain,8);
  strcat (TipoL,chain);
}

/*************************** SETIGESCOL ***************************/
int SetIgesCol (int state)

/***************************** ADDDE *****************************/
void AddDE (int type, int color)

/************************* WRITEHEADER ***************************/
void WriteHeader ()

/*************************** GOTHROGH ***************************/
void GoThrough (int DE_Sect)

/************************** ADDPDLINE ***************************/
void AddPDLine (float x1, float y1, float x2, float y2)

/************************** ADDPDARC ***************************/
void AddPDArc (float xc, float yc, float x1, float y1, float x2, float y2)
```

```
/************************** ADDPDTEXT ****************************/
void AddPDText (float x, float y, char *text)

/*************************** WRITETERMINATE **********************/
void WriteTerminate ()

/*************************** FINISHPDLINE ************************/
void FinishPDLine ()
```