Study on an Integrated System of Rapid Prototyping and Manufacturing for 3D Digitizer to CNC Mill

A thesis presented in partial fulfilment of the
requirements for the degree of
Master in Technology
at Massey University, Palmerston North
New Zealand

Jinghui Li

2002

ABSTRACT

The main purpose of this project is to develop a low cost, effective, user friendly interface software for staff and students to integrate the designing and manufacturing facilities in the Institute of Technology and Engineering (ITE) at Massey University, Palmerston North, New Zealand. The project involves establishment of an integrated CAD/CAM/CAE system, the identification of software requirements, selection of software development tool kit, definition of hardware configuration, software development and final experiments and tests.

ITE has a laboratory, where are equipped with one CNC milling machine, one CNC lathe, one Injection Moulding machine, one desktop 3D scanner and one 3D plotter. In addition, all the CAD/CAM/CAE software have been installed on the PCs. Based on the analysis and utilisation of these existing facilities, it is found that they are not smoothly integrated; no linkage between the CAD/CAM/CAE system and desktop Rapid Prototyping facilities; file formats used by each of the system are not compatible.

Through this project, the investigation of the possibility to integrate the system and the feasibility to develop a software to bridge the 3D scanner and the CNC mill, was carried out. A first try was successfully made using Borland C++5.0 to convert the 3D scanned data into NC program. Then, using Borland C++ Builder 5.0 created a user-friendly interface for conversion of 3D Digitizer to CNC Mill. Next, the different scales of wax models were satisfactorily processed on the CNC milling machine by inputting the converted NC program.

ACKNOLOWDGEMENTS

I wish to sincerely thank my supervisor, Dr. Liqiong Tang, for her assistance, guidance and brilliant advice during the whole period of my studies at Massey University, also deeply appreciate for her friendly support for getting my working experience.

I would like to extend my thanks to Leith Baker, for his cooperation, encouragement and discussion during I worked in the workshop.

I wish to acknowledge the friendship, assistance and encouragement from all postgraduate students in the Institute of Technology and Engineering at Massey University. Particularly, I want to express special thanks for my classmate, Weerawate Utto, for his friendship, discussion, cooperation and helping during my studying at Massey University.

I am grateful for the assistance from Yongqiu Liu and Bing Luo, who are computer science Ph.D. students in the Institute of Information Science and Technology at Massey University, during my learning computer programming language C and C++.

To the staff within the Institute of Technology and Engineering for helping me in the past. Particular thanks to Joan Brooks, the secretary of the Institute of Technology and Engineering, for her appreciate and warm service.

To my wife and my son, Jun Teng and Tianyuan Li, for their fully support and understanding during my struggle to finish my studying.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter1 Introduction

Computer-aided design (CAD), Computer-aided manufacturing (CAM) and Computer-aided engineering (CAE) have become three essential parts of many manufacturing industrial companies. Computer Integrated Manufacturing (CIM) is no longer a new term to industrial companies. However, to establish a low cost, effective computerised design and manufacturing environment is still a challenging topic to small or medium-sized companies. This research project is to demonstrate how to develop low cost software to integrate design and manufacturing facilities. The project work involves the identification of software requirements, selection of software development tool kit, definition of hardware configuration, software development and final experiments and tests.

The project was proposed based on the facilities of the Institute of Technology and Engineering (ITE). ITE has a workshop which equipped with one CNC milling machine, one CNC lathe and injection moulding machine. All the CNC machines are networked to the CAD/CAM laboratory where all the CAD/CAM/CAE software installed. Beside these above-mentioned facilities, ITE also has a desktop 3D scanner and a 3D plotter. However, all these facilities are not smoothly integrated, especially the desktop facilities. Figure 1-1 illustrates the facilities in ITE before this research project proposed.

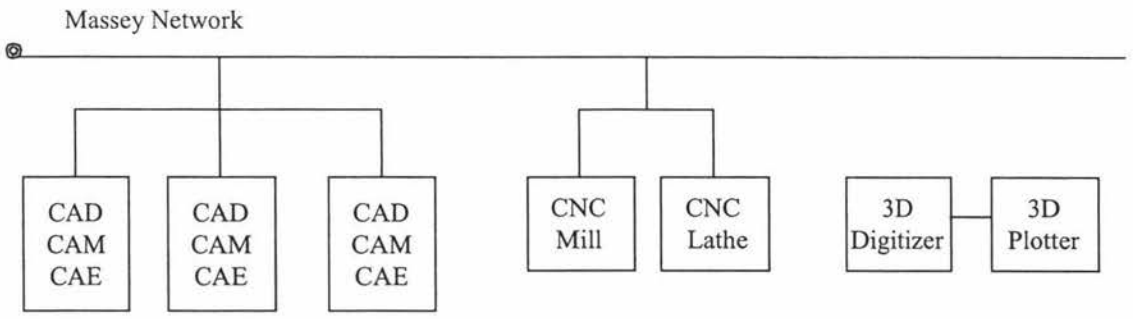


Figure 1-1 The Facilities in ITE Before This Project

The problems with these facilities are:

1) Any model scanned by the 3D scanner can only be reproduced by the 3D plotter. There is no linkage between the CNC machines and the 3D scanner.
2) There is no linkage between the CAD/CAM/CAE system and the desktop Rapid Prototyping facilities.
3) The file formats used by each of the system are not compatible.

It is expected that, through this project, a software could be developed to bridge the 3D scanner to the CNC mill, and to demonstrate an integrated CAD/CAM/CAE and CNC system. Based on such a background, the project first made a study on ITE's hardware and software to investigate the possibility to integrate the system and the feasibility to develop a software to bridge the 3D scanner and the CNC mill. The study clearly showed that the software is a must to convert the scanned data into NC program. A first try was carried out using Borland C++ 5.0. This is because Massey has Borland's development tool kit. Following the success of the first try, the project then identified Borland C++ Builder 5.0 as the development tool kit and began to develop the window-based software for bridging the 3D scanner to the CNC facilities.

The main objectives of this project are:

- To study the possibility of setting up a system which integrates CAD/CAM/CAE systems, CNC, a 3D scanner and a 3D plotter.
- To investigate the possibility of using a CNC Mill machine to produce models with scales by converting the scanned data from the physical part into NC Mill programs.
- To develop a user-friendly interface to bridge the 3D scanner to the CNC mill

Chapter2 Literature Review

With rapid development of computer technology, more and more new technologies, such as, Rapid Prototyping (RP), Rapid Tooling, Reverse Engineering (RE), Virtual Prototyping (VP), have become available for to be applied in engineering areas. However, an impediment to success is the considerable confusion over the terminology used in both technical and marketplace discussions. It is essential to rationalize the terms already widely used today.

2.1 Rapid Prototyping

2.1.1 Definition

Rapid Prototyping can be defined as a group of techniques used to quickly fabricate a scale model of a part or assembly using three-dimensional computer aided design data (CAD or geometric data) without traditional tools or material removal (Efunda Process Homepage). It is also known as layer manufacturing, material deposit manufacturing, material addition manufacturing, solid freeform manufacturing, desktop manufacturing and three-dimensional printing (Chua, 1999).

The first commercial process, developed by 3D Systems, Inc., was shown in Detroit (US) in November 1987 (Dolenc, 1994). The first RP technique commonly considered was Stereolithography. Since then, a number of different Rapid Prototyping techniques have become available. Rapid Prototyping is so widely used in manufacturing that Grimm (1999) commented it could be used to as a competitive weapon in the areas of quality, delivery and price. The term RP is now normally reserved for the new technologies that build parts by adding material instead of removing it (Dolenc, 1994).

2.1.2 The Advantages of Rapid Prototyping

Rapid prototyping is a way in which a geometric model of a part is used in manufacturing. Various rapid prototyping processes are available, but all of them generate a prototype by laying composite material layer by layer. The main advantage of rapid prototyping processes is that they build a prototype in one step, directly produce the physical model from the geometric model of the part. Thus RP processes do not require planning of process sequences, specific equipment for handling materials, transportation between machining stations, and so on.

Generally, the advantages of Rapid Prototyping are:

- To increase effective communication or 3d visualisation of product design.
- To decrease development time, especially for lead times.
- To decrease costly mistakes and improve product quality.
- To minimize sustaining engineering changes.
- To extend product lifetime by adding necessary feature and eliminating redundant features early in the design (Belludi N. and D. K. Thakral; Efunda Process Homepage; Wohlers, 1995).

2.1.3 The Rapid Prototyping Techniques

According to the function of the prototype, RP techniques can be classified into the following four main classes (Karapatis, 1998):

- Visual prototype, which is just a "solid", real CAD model used to reveal design defects and gain marketing clearance;
- The functional prototype ("form & fit"), which is mainly used to detect assembly problems;
- The material prototype, which has the same geometrical and mechanical properties as the final part;
- The production prototype, which is made by the same process as the final part.

Currently commercially available techniques include Stereolithography Apparatus (SLA), Laminated Object Manufacturing (LOM), Selective Laser Sintering (SLS), Solid Ground Curing (SGC), Fused Deposition Modelling (FDM) and Printed Computer Tomography (PCT) (Efunda Process Homepage).

2.1.3.1 The Stereolithography Apparatus

The Stereolithography, the most widely installed system from 3D Systems (Valencia, California, USA, 1986), is a layer-by-layer manufacturing process (Crockett; Dolenc, 1994; Efunda; Novitski, 1999; Wohlers, 1993). The part is built in a photopolymer liquid environment. When exposed to ultra-violet (UV) laser beam, the liquid solidifies or is cured. First, a 3D surface or solid model of a part is created with a CAD system, and then a support structure is built and attached to the bottom of the CAD Model. Especially critical for parts contained overhanging geometry, the support structure attaches the part to the SLA's elevator platform and supports the part as the machine builds it. It is removed after building the part.

2.1.3.2 Laminated Object Manufacturing

Laminated Object Manufacturing, invented by Michael Feygin of Russia who is the president of Helisys, Inc. (Torrance, California), uses foils or sheet material cuts and binds methods to build parts (Crockett; Dolenc, 1994; Efunda; Novitski, 1999; Wohlers, 1993). The machine automatically positions a thin sheet of paper material from a roll on an elevator platform, and then uses a CO_2 laser to cut the sheet with a computer-controlled x-y plotter system to direct the laser beam. The cut corresponds precisely to the first cross section of the CAD model – in STL format.

The fresh sheet material is bonded to the previous sheet by a heated roller that presses the two sheets together. The heat from the roller causes the polyethylene-coated paper to fuse together. The laser then cuts the next sheet and the process repeats until the part is complete. In order to remove the part, the unwanted pieces, which supports the part during the building process, must be separated from the part.

LOM parts do not require post-curing and additional support, although they are usually sealed with polyurethane. LOM parts have a wood-like texture composed of paper layer. For some parts, the LOM process promises to be faster than other layering processes because the laser outlines the periphery of the part only instead of making contact with the entire surface of the part.

The LOM developer continues to improve the process with sheets of stronger materials such as plastic and metal. The available sheets of powder metal (bound with adhesive) can produce a "green" part, which is then heat-treated material to its final state (Efunda Process Homepage).

2.1.3.3 Selective Laser Sintering

Selective Laser Sintering, made by DTM Corporation (Austin, TX) in 1988, operates on the same basic layer-by-layer principle as the SLA (Crockett; Dolenc, 1994; Efunda; Novitski, 1999; Wohlers, 1993). However, it uses different materials of powders instead of a liquid polymer to spread over a platform by a roller. A laser sinters selected areas causing the particles to melt and then solidify. To reduce the laser energy for sintering the powder, to reduce thermal shrinkage of the layers and reduce part distortion, the powder is preheated in the build chamber.

For some applications, because the surrounding powder material serves as a natural support, minimizing the need for support structures to support overhanging geometry, the SLS process may has advantage over liquid-based system. Therefore, SLS process can save time during the creation of the part, as well as after you remove it from the machine. In addition, the SLS process can produce parts in several materials, including polycarbonate, investment casting wax and nylon. However, the low density of parts is usually weaker than solid parts.

2.1.3.4 Solid Ground Curing

Solid Ground Curing, also known as the solider process, is a process that was invented and developed by Cubital Inc. of Israel. It uses a sensitive to UV-light photopolymer to solidify part (Dolenc, 1994; Efunda; Wohlers, 1993). The solider system irradiates and solidifies a whole layer in a few seconds, regardless of size, complexity and number of parts.

The system charges a glass mask plate as it passes over an ion gun, which in turn shoots ions on the glass at a resolution of 118 dots per cm and forms a negative image of the cross section pattern. The black electrostatic toner in the system adheres to the ion charged portions of the plate; the transparent areas reflect precisely the cross section of the parts. The plate is positioned closely over the top of a thin layer of liquid polymer, and then an intense flood of ultraviolet light shines through the plate, exposing the entire layer.

After being exposed to the light for a few seconds, the polymer solidifies and the partially finished part moves away from the exposing chamber. The solidified layer then undergoes several processes before producing a new layer. While these processes are underway, the toner is wiped from the glass plate and the next cross section is processed.

The unhardened liquid surrounding the hardened layer is removed by suction. A thin layer of wax is then spread over the entire layer, filling the areas that before held liquid polymer. The wax, which goes through a cooling process, surrounds and supports the part. It mills the layer and produces a flat surface ready for the next layer, when the partially finished part moves to a milling station. All of these operations automatically repeat until the part is completed. The surrounding wax can be melted away using hot water, a hot air gun, a microwave oven, or a conventional dishwasher.

2.1.3.5 Fused Deposition Modelling

Stratasys of Eden Prairiek, Minneapolis makes Fused Deposition Modelling (FDM) machine. The FDM process was developed by Scott Crump in 1988. The fundamental process involves heating a filament of the thermoplastic polymer and squeezing it out like toothpaste from a tube to form the RP layers. The machines range from fast concept modellers to slower high-precision machines. The materials include polyester, ABS, elastomers, and investment casting wax (Crockett; Efunda; Novitski, 1999).

FDM consists of the main 3D Modeller unit, ProtoSlicing software and a Silicon Graphics workstation. The lightweight FDM head operates at up to 38.1cm per second. Successive laminations adhere to one another to form models up to 30.5cm × 30.5cm × 30.5cm in size. The system does not waste material during or after producing the model, so the process requires little cleanup.

The system uses thermoplastic materials, including a machinable wax, an investment casting wax, a material called Plastic200 and a new tough nylon-like plastic material called Plastic300. A glassy finish on the surface of the wax part can be achieved by applying a solvent.

2.1.3.6 Ink Jet Printing Techniques

Ink jet printing, another name of the printed computer tomography, comes from the printer and plotter industry where the technique involves shooting tiny droplets of ink on paper to produce graphic images. RP ink jet techniques utilise ink jet technology to shoot droplets of liquid-to-solid compound and form a layer of an RP model. Common ink jet printing techniques, such as Sanders ModelMaker, Multi-Jet Modelling, Z402 Ink Jet System, and Three-Dimensional Printing (Efunda; Novitski, 1999; Wohlers, 1993).

The Sander ModelMaker product is produced and distributed by Sandders Prototype, Inc. of Witon, NH, USA. Somooth cosmetic surface quality can be achieved by pre-tracing the perimeter of a layer prior to filling in the interior. The supporting wax material is deposited at the same time as the thermoplastic (Efunda).

Another product of 3D Systems from the makers of the SLA system, Multi-Jet Modelling uses a 96-element print head to deposit molten plastic for layering. The system is fast compared to most other RP techniques, and produces good appearance models with minimal operator effort. The main market is targeted at the engineering office where the system must be non-toxic, quiet, small, and with minimal odour.

Three-Dimensional Printing, developed by Massachusetts Institute of Technology (Cambridge, MA) and Soligen, Inc., is commercialized by Z Corporation using the technique based on the ink jet printing process (Novitski, 1999). The part is built layer by layer, by jet printing a thin powder layer with a binder material (Karapatis, 1998).

The Z402 (Figure 2-1), made by Z Corporation (Wohlersassociates Hompage, 2000), is one of the fastest 3D printers known to RP. The ability to produce quick models means greater productivity for the lab and quick prototypes for customers. Since manufacturing parts is easy, almost anyone in the lab can produce quality part without extensive RP experience.

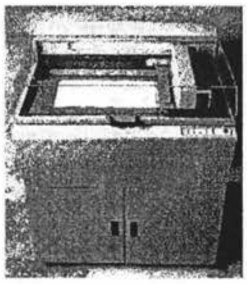


Figure 2-1 Z402 3D Printer Courtesy of Z Corp

(Source: Wohlersassociates Homepage)

In addition, Printed Computer tomography, developed by Texas Instruments (McKinney, TX), uses an inexpensive ink jet printing mechanism (ProtoJet 3D Printing System) to deposit wax material, layer by layer (Wohlers, 1993).

Compared to other RP systems, 3D Printing System can reduce operation, maintenance and material costs. It suits for operating in an office environment near the CAD systems because its small size, office friendliness and using non-toxic material. Also, Vendor companies are positioning them as machines that can give you a quick and inexpensive model early in the design process. Support structures for overhanging geometry can automatically generate in a water-soluble material. Therefore, it is easily removed by soaking the part in warm water.

2.1.3.7 Photochemical Machining

Photochemical Machining, explored by Formigraphic Engine Company (Bolinas, CA), Battelle (Columbus, OH) and Osaka Prefecture (Japan), would function similarly to other liquid and laser-based systems (Wohlers, 1993). However, solidification would occur in the interior of the material not on the surface of the liquid. Two lasers of different wavelengths would be used to initiate cross-linking. Polymer would be chemically altered and solidified at the intersection point of the two beams.

Researchers are exploring several possibilities. One is that the scanned areas on a solid block would become cross-linked and insoluble. The final part would appear after soaking the block in a solvent. Another possibility would involve a block of soft gel, hardening where the laser beams meet. Yet another process would use a frozen solution. Only the unscanned material would liquefy when heated, leaving a freestanding part. Finally, the unwanted material could be degraded from a solid block like machining the part with the two lasers.

2.1.4 Rapid Tooling

Rapid Tooling (RT) is evolved from Rapid Prototyping technology and its application (Wolers Associates, 2000). It can be used to describe a process that either uses rapid prototyping model as a pattern to create a mould quickly or uses the prototyping process directly to fabricate a tool for a limited volume of prototypes (Efunda Process Homepage). Many in the RP business agree that RT means "RP-driven tooling", the key to making it rapid. Rapid tooling can be seen as the second wave in rapid prototyping because, with rapid tooling, the production process can be prototyped instead of the final product (Radstok, 1999).

Rapid tooling techniques (RT), allowing the manufacture of production tools, offer a high potential for a faster response to market needs, creating a new competitive edge. The purpose of RT is not the manufacture of final parts, but the preparation of the means to manufacture final parts: mass production tools such as molds, dies, etc., can be ready in very short times (Karapatis N. P., 1998).

Based on a criterion related to the number of operations, Rapid Tooling can be divided into two broad categories: *Indirect Rapid Tooling* and *Direct Rapid Tooling*.

Indirect Rapid Tooling uses RP master patterns to produce a mold. These techniques is described as "non-direct tooling routes", since they induce at least one intermediate step in the tooling process (Karapatis, 1998). The illustrations of this are aluminum-filled epoxy tooling and 3D Keltool from 3D System.

Direct Rapid Tooling means that an RP machine builds the actual core and cavity mold inserts without intermediate steps. For example, Rapid Tool from DTM, direct metal laser sintering from EOS, and Direct AIM from 3D Systems (Wohlers Associates, 2000).

Karapatis (1998) stated: "Direct rapid tooling (DRT) is an industrial concept aimed at the realization of production tooling directly from CAD data files, with the smallest possible process chain (number of operations). Its purposes is the manufacture of tools that can be used under normal production conditions, in terms of durability, accuracy and surface quality."

2.2 Reverse Engineering

2.2.1 Definition

Reverse Engineering can be defined as a process starting with the known product and working backwards to define the process which aided in its development or manufacture. With respect to software, Reverse Engineering essentially refers to an analysis and deconstruction of an existing computer program in an effort to determine how it operates (Percival, 1999).

Reverse Engineering (RE) is the process or analysing a subject system to identity the system's components and their interrelationships and create representations of the system in another form or at a higher level of abstraction. The primary purpose of reverse engineering a software system is to increase the overall comprehensibility of the system for both maintenance and new development (Chikofsky, 1990).

Generally, Reverse Engineering involves extracting design artifacts and building or synthesizing abstractions that are less implementation-dependent. Reverse engineering in and of itself dose not involve changing the subject system or creating a new system based on the reverse-engineered subject system. It is a process of examination, not a process of change or replication (Chikofsky, 1990).

According to Christensen J. and Amit B. (2000), reverse engineering also refers to the virtual replication of physical object, which a process where an object is digitized by recording surface coordinates to capture the original design features.

2.2.2 The Application of Reverse Engineering

The reverse engineering is mainly applied to industrial, medical, historical, and entertainment industries (Christensen J. and Amit B., 2000).

One application is to modify legacy industrial products which could be reverse engineered and then edited with computer aided design (CAD) software. Once a CAD file is obtained then some form of rapid manufacturing can be used to produce the new product.

The second one is used in the medical field for visualization of the human body that is critical to solving many diseases and fractures. Doctors will be able to view a virtual image of the area of concern.

The third one is used to preserve virtual copies of historical artefacts. Because a digital copy of an object takes up on space (excluding that of the storage device) many artefacts can be stored and accessed by anyone without any danger to the physical object.

The last one is in the entertainment industry. The physical models are often sculpted, reverse engineered, and then animated on a computer.

2.2.3 The Classification of the Reverse Engineering

Based on the process mechanism, the techniques can be broadly classified into two categories such as *contact* and *non-contact*. In another words, the contact RE process makes contact with the part that will be reversed engineered while the non-contact one does not. In addition, different RE processes have also been classified as either *a destructive* or *non-destructive process*. Most of the common RE processes are non-destructive, either contact or non-contact based, instead of contact destructive ones.

Most of the mainstream RE systems use one or more of the following technologies to obtain three-dimensional data from a part (Christensen J. and Amit B., 2000).

- Surface contact based
- Optics based

- Laser scanning based
- Radiation/Wave/or Field based

2.2.4 Why Reverse Engineering

The reverse engineering process, with respect to software, is dedicated to a determination of what a computer programme contains. It is often a last resort to obtain information that is not otherwise available. The goal is to be able to read the original copyrighted program that has been stored in a format not easily comprehensible by humans (Percival, 1999). So, reverse engineering is often as expensive or more expensive than the creation and development of the original program. If expensive and extremely difficult process, why reverse engineering software?

The first reason is that reverse engineering geared towards the creation of a functionally equivalent and competitive product may assist the competitor by showing her how the programme works, and perhaps how the programme handles particular tasks or concepts.

The second reason is directed towards creating a product compatible with or interoperated with the original software product. To function, the various components of the computer must be able to communicate. The hardware components, operating system and application software must all be compatible when communicating in the sense that each component understands the format of the other's communication.

The last reason is that manufacturers of hardware and software products often do not wish to provide the technical interface specifications which are necessary to achieve compatibility or may only provide a basic level of information leaving out useful details which enable the original manufacturer to retain technical advantages. In such circumstances, reverse engineering is the only way to unearth the interface and other technical specifications necessary to ensure compatibility between the existing interfaces and the interfaces being designed into the new product (Percival, 1999).

2.2.5 Reverse Engineering Techniques

One tipical technique of Reverse Engineering is 3D digitizing which can be used to create a digital model from a physical part. The process is appealing because it can be difficult to create models of complex objects using computer tools without the aid of a 3D input device. Recreating an existing part from scratch, even with a computer, is like copying a printed page by retyping it. Although 3D digitizers are not as straightforward as a photocopy machine, the intent is the same (Wohlers Associates, 2000).

The user can render and print a digitized model to communicate shape information, extract dimensions from it to show size information, and use the 3D database to manufacture a replica using rapid prototyping and CNC machines. You can also include the 3D Model in multimedia or animation software as a learning or assembly aid.

The challenge of the digitization process in manufacturing is to capture adequate detail and resolution (Ideas in 3D, 2000), because in manufacturing, RP and CNC machines require clean, complete, and accurate information. If areas on the model are incomplete or missing, it may be difficult or impossible to build the part.

3D digitizing systems are best at digitizing organic shapes such as free-form sculpted surfaces. Many objects, such as human anatomy, animals, bones, skeletons, can be seen on an advertisement or a catalogue from companies offering digitized models.

Most 3D digitizing systems are either a non-contact device or one that uses a touch probe. Touch probe systems are the least expensive. You must touch the object for every 3D point your want to produce. This makes it challenging to produce compute models at a high resolution (Ideas in 3D, 2000).

Non-contact digitizing systems are the only practical choice for high resolution scanning of physical parts. Laser digitizers have become the most popular non-contact systems for most applications. They collect from 20 to more than 25,000 points per second and provide resolution of better than 0.001-inch to 0.02o-inch, depending on the particular technology.

1) Digitizing Arms and Coordinate Measuring Machines (CMM's)

These types of digitizing machines are more suited in the field of metrology and are primarily used for quality control work. These types of digitizers can be made to CAD surface creation needs, but require different techniques developed to fit each unique model design.

2) Probe Scanning

Probe scanning is based on a sensitive probe of feeler that registers the 3 dimensional coordinates of any point on a surface that it touches. A rigid x-y-z mechanism moves the probe along one plane as it records point coordinates on the surface in the other two axes. The coordinates are recorded at regular increments, chosen by the operator in a trade off between accuracy and workable file sizes. At the end of each pass, the system then steps over and makes another scan in a parallel plane one increment over, much like the raster scan of a TV or computer screen.

The advantages of probe scanning are that it can raster scan 3 Dimensional objects in an organized point order at about 100 points/sec and also pick up 2D contours that are flat, such as punch contours very fast and accurately.

3) Laser Scanning

Laser digitizers use a basic principle called triangulation. When light arrives on a surface from one direction, and the light is seen from another direction, the location of the point can be inferred. That is how lasers and sensors work in harmony to create x/y/z coordinates of points on the surface of an object. Movement of the laser light and sensors, as well as the part itself, provides multiple degrees of freedom. Through this combination of motion, laser digitizers are capable to capture the shape and size of odd-shaped objects.

The main advantage of laser scanning is that this method can create ten's of thousands of points on the surface per second. Edge definition using this method is a function of how close the points are at the edges which in turn affect the size and manipulation of those files. This in turn requires a point manipulation program that is specifically designed for this purpose however the edge definition can never be accurate. Also, it is very useful for digitizing clay sculpted animation models.

Cyber F/X Inc. (Gendale, CA) uses the laser- and video-based technology to scan maquettes, objects, and people (including whole body, Figure 2-2). An astounding 15,000 points per second are "captured" by the laser scanner, which records colour, as well as shape (Cyber F/X Inc, 2000).

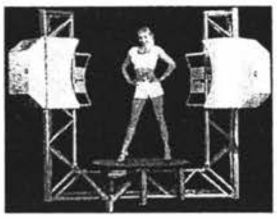


Figure 2-2 Body scanner of Cyber F/X Inc.

Source: Cyber F/X Inc. homepage

TriForm is a new non-contact 3D image capture system made by Wick and Wilson Ltd. (UK). This system (Figure 2-3) can digitizes the surfaces of real objects, converting them to point clouds containing up to 400,000 three-dimensional co-ordinates and associated colour data. The data is captured and processed in a matter of seconds to produce full colour three-dimensional images which can be exported for use in a variety of third party software applications for display or analysis (3D Scanning System, 2000).

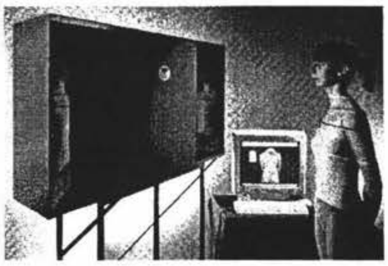


Figure 2-3 A dual view TriForm system

Source: Wicks and Wilson Lit. homepage

4) Reverse Engineering and Digitizing (R.E.A.D.)

R.E.A.D. is more than a reverse engineering solution – it's a revolution in the industry. Imagine tracing complex moulds inside a true CAD environment. It's no surprise that this process will rapidly become the industry standard. We've met the wish lists of Engineering, Manufacturing, Industrial Design and Product Development with a system that's fast, accurate, and user friendly. R.E.A.D. shows off its true power when you need to digitize aerodynamic type surfaces. Typically, surfaces and contours are digitized with a touch probe or drag scanned using a hard probe. This flexible process allows you to gather data from an actual part and into the CAD model in real time as you design.

2.3 Virtual Prototyping

2.3.1 Definition

Virtual prototyping (VP), sometimes referred to as computer-aided engineering (CAE) or engineering analysis simulation, is used to carry out analysis and simulation on a fully developed computer model and then perform the same tests as those on the physical prototypes (Chua, 1999). Virtual means the result of the product design is not yet physically created but a visual representation is presented for observation, analysis, and manipulation (Ogilvie, 1997).

Since the 1970s, the Virtual Prototyping has been in steady development. It is taken to as the testing and analysis of 3D solid models on computing platforms. Nowadays, VP is often tightly integrated with CAD/CAM software and tests part behaviour in a simulated context without the need to manufacture the part first.

Virtual Prototyping is based on physics, models and simulation. It is a useful engineering tool for all kinds of technical designers. The idea is to provide designers an opportunity to do what-if-analysis during the planning process. VP programmes calculate accurate numbers and show paragraphs to engineers to help them to decide wherever or not it is possible to do something. For example when integrated into CAD program, virtual prototyping can give direct answer what will happen, when a change is made to the designed particle (Virtual Prototyping).

2.3.2 The function of Virtual Prototyping

Virtual Prototyping includes the following functions (Chua, 1999):

- Finite element analysis;
- Mechanical form, fit and interference checking;
- Mechanical simulation.

- Virtual reality applications;
- Cosmetic modelling;
- Assemblability.

2.3.3 The Benefit of Virtual Prototyping

By using virtual prototyping techniques such as interference checking, moult-flow analysis, static and dynamic stress analysis, it is possible to predict and eliminate potential problems at an early stage in the design process.

The long-term benefits of using such virtual prototyping in designing are (Virtual Prototyping):

- Time-to-market reduces significantly
- Increased accessibility to CAE technology with CAD environment and user interface
- Continuos access to advanced simulation

2.4 Facilities Available In the Institute of Technology and Engineering

2.4.1 PICZA 3D Scanner

Many desktop digitizing and 3D machining systems in a reasonable price range have come onto the market. The ability to copy a part or model and send its image to a milling machine and begin instant machining, including complex three-dimensional parts, is here. Files are scanned, digitized or drawn geometrically and fed directly into the milling machines.

A wonderful device is the "PICZA" 3D scanner/digitizer. This machine is a desktop unit with a moving table that holds the object to be scanned. As it moves, a digitizing needle touches the part, creating a computer replica of the part. The digitized data is in a format which is compatible with a 3D plotter called 3D Modela. Such an integrated system makes a user, at the entry level, be up and running quickly.

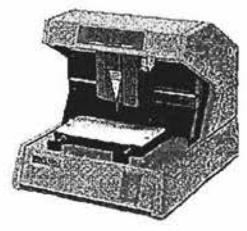


Figure 2-4 PICZA 3D Scanner

Source: International Engraving Devices, Inc. homepage

PICZA Model Pix-3 3D Scanner made by Roland Digital Group is a desktop contact-type 3D scanner. Nearly any objects can be scanned with Picza to create a digital 3D image which can then be further edited and processed.

Table 2-1 Picza Specifications

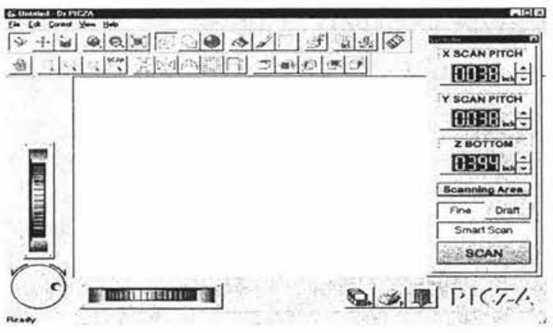
Table Area	170mm×110mm
Work Area (x×y×z)	152.4mm×101.6mm×40.65mm
Feed Rate	4mm / sec. – 20mm / sec.
Sensing Unit	Piezo sensor
Scan Pitch	X, Y axis: 0.05m –1.0mm (increments of 0.05mm) Z axis:0.025mm
Dimensions (W×D×H)	350mm×330mm×300mm
Weight	10kg
Acceptable Material Weights	Up to 400g

Source: International Engraving Devices, Inc.

Picza has a special sensor, called "Roland Active Piezo Sensor," which can scan data with hair-splitting precision, picking up even the most minute shape variations. This makes Picza capable of scanning all kinds of objects, such as, transparent materials, clay figures, even fruits or other soft objects (International Engraving Devices, Inc. Home page).

The software included with the Picza 3D scanner is called Dr. PICZA (Figure2-5a) run under Window95/98 or Windows NT. Dr. PICZA features a dynamic graphic display and diverse editing functions. It can control functions such as scan pitch and area settings, plus numerous editing functions including a handy convex / concave inversion function for making moulds, a mirror function for creating symmetric data, a tilt adjustment function, a curve smoothing function and a function for adjusting the height of surfaces. The image from any angle with a wire frame can be checked using a preview function. The colour and texture renderings can also be displayed (International Engraving Devices, Inc. Home page).

(a)



(b)

Figure 2-5 The software of Dr.PICZA

The scanned data can be stored in its original format, in DXF (AutoCAD Release12) and VRML (Virtual Reality Modeling Language for Internet use) file formats for wide range of applications. Also, Dr. PICZA can output 3D data directly to Modela Player. Figure2-5b is the user interface of Dr. PICZA.

2.4.2 Modela 3D Plotter

Modela 3D Plotter (MDX-3), also made by Roland Digital Group, is a full 2.5 axis CNC milling machine inside a very small housing: its size does not exceed a desktop printer (see Figure 2-6 and Table 2-2). It is capable of producing three-dimensional objects from a CAD model and the machine can be driven directly from a PC.

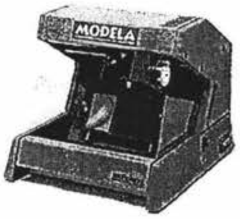


Figure 2-6 Modela MDX-3

Table 2-2 Specifications of Modela Player MDX-3

Table Area	170mm × 110mm
Work Area (X×Y×Z)	152.4mm×101.6mm×40.65mm
Feed Rate	0.1mm / sec. – 15mm / sec.
Spindle Motor	5W (DC Motor)
Revolution Speed	4500 rpm (+/- 10%)
Tool Chuck	6mm
Dimension (W×D×H)	350mm×330mm×300mm
Weight	7kg
Acceptable Materials	Foamboard, jeweller's wax, OBOModulan, Balsawood, ABS, Polyacetal, styrofoam, other light-wight materials

Source: International Engraving Device Inc.

The associated software with Modela 3D Plotter is Modela Player (Figure 2-7). Modela Player is a windows program to perform cutting three-dimensional objects on the Modela, CAMM-3, or CAMM-2 three-dimensional modelling machines made by Roland Digital Group.

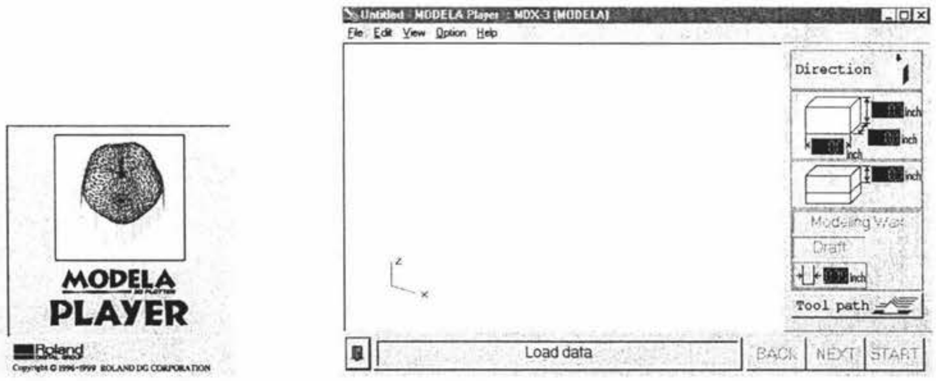


Figure 2-7 The software of Modela Player

Modela Player can import objects created using Modela 3D Design (Figure2-8), Modela 3D Text (Figure 2-9) and Dr.Picaza as well as other commercially available 3D CAD software. Within Modela Player, user can set the orientation of the part to be machined, specify the cutting tools, and define machining parameters. The software is also capable of starting and stopping the machining process.

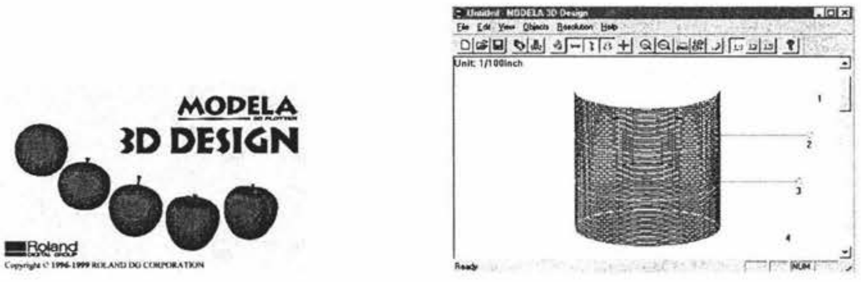


Figure 2-8 The software of Modela 3D Design

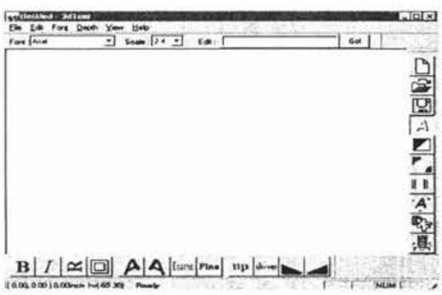


Figure 2-9 The Interface of Modela 3D Text

Roland DGA Corp. (Ivine, CA) has also introduced a new combination of 3D scanner and 3D modelling machine, the MDX-15 equipped with interchangeable scanning and milling units, the MDX-15 is suited for 3D model making moulds, rapid prototyping, and product design (Figure2-10)

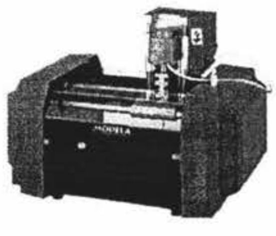


Figure 2-10 The feature of MDX-15

Source: MDX-15 features

2.4.3 CNC Milling Machine

The CNC Milling Machine is Bridgeport Series I Interact. It has the additional capability for 2 axis simultaneous linear movement and circular contouring plus mirror image, datum shift and 3 axis positioning. The interactive CNC control provides direct programming on the machine.

Chapter3 Investigation of Input / Output Format Between Picza and Bridgeport 145 Controller

3.1 Investigation of Desktop Scanner Output Format

3.1.1 Features of Picza 3D Scanner

3.1.1.1 System Setup

PICZA is a compact desktop contact-type and PC-based 3D scanning device. It can scan clay figures, model, fruit, even glass and other transparent materials to create a digital 3D image which can be editing and processing on PC. The scanner is very simply to use. After turn on the power switch button, set up the basic scanning parameters (like area and pitch) with mouse using the Dr.PICZA software, and then click start button, the scanning process will be done automatically.

With "Roland Active Piezo Sensor (R.A.P.S.)," the scanner can scan data with hair-splitting precision, picking up even the most minute shape variations, Minimum scanning pitches are 0.025mm in height (Z axis) and 0.05mm in width (X axis) and depth (Y axis). The scanning volume measures 150mm in the X axis, 100mm in the Y axis and 40mm in Z axis.

The main component of the scanner is shown in the Figure 3-1. It has three axes (X, Y, Z). The workpiece plate works like a small machine table, which can moves back and forth in the Y direction along two parallel bars. The Z unit in which the scanning probe is housed moves in the X direction along two parallel rods. While, the probe itself is driven up and down in the Z direction by a small DC motor.

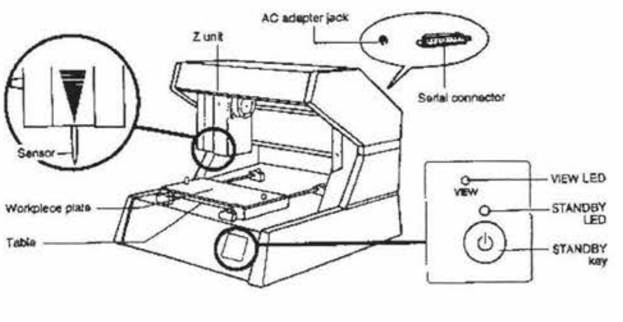
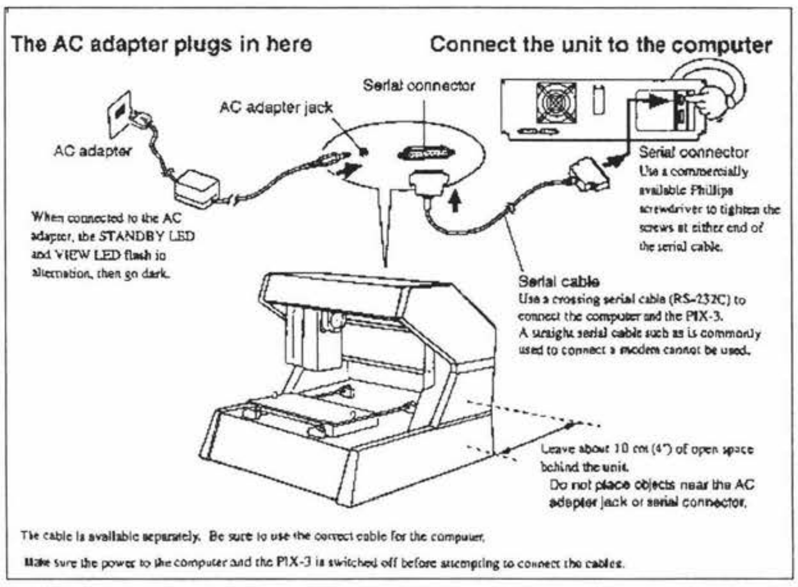


Figure 3-1 The Main Component of the Scanner



When connected to the AC adapter, the STANDBY LED and VIEW LED flash in alternation, then go dark.

Use a commercially available Phillips screwdriver to tighten the screws at either end of the serial cable.

Use a crossing serial cable (RS-232C) to connect the computer and the PIX-3. A straight serial cable such as is commonly used to connect a modem cannot be used.

Leave about 10 cm (4") of open space behind the unit.

Do not place objects near the AC adapter jack or serial connector.

The cable is available separately. Be sure to use the correct cable for the computer.

Make sure the power to the computer and the PIX-3 is switched off before attempting to connect the cables.

Figure 3-2 Setting up and Connection of the Picza with Computer

The communication between the scanner and a PC is through a RS232 cable (Figure 3-2). An AC adapter is required for power supply. The communication speed is 9600bps. For every scanning job, initialisation is required. Datum has to be set-up before any job starts.

The configurations of the computer which is going to be used to run the scanning software, Dr. Picza is:

- Pentium 100 MHZ or better
- At least 16 Mbytes memory
- 3 Mbytes of free space
- Operating system Windows 95/98 or Windows NT

- 3.5' floppy disk

3.1.1.2 *Functions of Dr. Picza*

Dr. PICZA is a Windows 95/98/NT application for controlling the 3D scanner. Main Features of Dr. PICZA are as followings:

- Scan a three-dimensional object with the 3D scanner
- Set the scan pitch and the scanning area
- Set the scan quality
- After scanning, set the scanning area and rescan
- Edit three-dimensional scan data
- Invert a solid object (Invert function) which is especially useful for injection die cavities
- Create a mirror-image copy of a object (Mirror function)
- Adjust the slant of an object (Slant function)
- Change the height of the selected surface
- Reduce the number of scan points (Data Thinning function)
- Smoothing an entire object (Smoothing function)
- Examine the object from different angles
- Add color to the object's faces (Rendering function)
- Export 3D data directly to Modela Player to cut a 3D object
- Save the file in DXF, VRML, STL, 3DMF, BMP, Grayscale format and the point-group format.

3.1.1.3 *The Scanning Process*

1. Installation and quick start of Dr. Picza:

1) Insert the Dr. Picza disks into the drive A and follow the messages to carry out setup. Once this has been done, the software Dr Picza will be installed into the hard disk C.
2) Press the Start button and select Roland Dr. Picza program and Dr. PICZA submenu, the main interface (see Figure 2-5) for scanning will come out.
3) Press the File button and select "preference" and make the chose communication port match the serial terminal where the cable is connected.

2. Prepare the model to be scanned.

Clean the surface dust of model, as the scanner is so sensitive. Check the shape or profile to be scanned, because shapes with portions that are not visible from every angle, the interiors of cavities, and shapes such as helixes do not yield their actual form when scanned (see Figure 3-3). The 3D data can be made to match the actual shape of the scan object by using Export to save the data to a file, then editing with a commercially available 3D application.

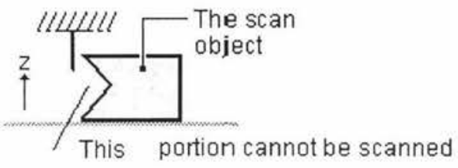


Figure 3-3 Difficulty Shape to Scan

3. Load the object on the workpiece plate

The plate can be removed by loosening the mounting screws. The common way to attach the object to the plate is through double-sided tape. Once the object is firmly hold on the plate, it is screwed back onto the sliding table. Then the user can control the scanning process from the interface of Dr. Picza installed on the PC.

4. Adjusting the setup or using the rescan function to shorten scan times.

Scan times are shorter for shapes moving downward along the X axis (descending-slope surfaces) than for shapes moving upward (ascending-slope surfaces, see Figure 3-4).

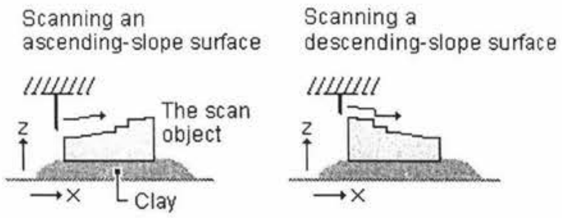

Scanning an ascending-slope surface
Scanning a descending-slope surface
The scan object
z
x
Clay

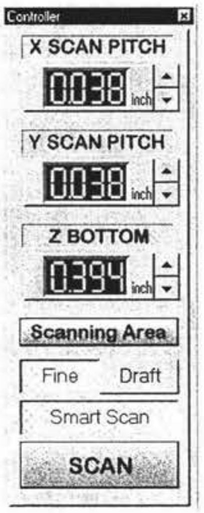
Figure 3-4 The Comparison of Two Kind of Slop Surface

When using the Rescan function, the user can perform scanning with partial changes in scanning pitch. After doing coarse scanning of the entire object, the user can perform fine scanning for a specific area. As the user can optimize the scanning pitch according to the shape, judicious use of this can shorten scan times.

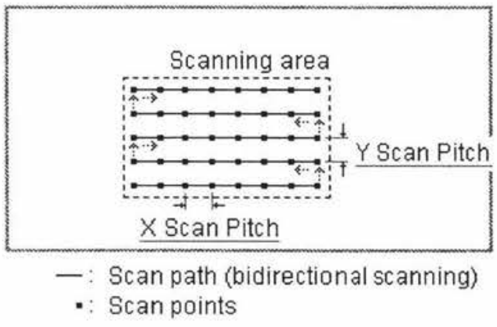
5. Set the scanning conditions

Set the scanning resolution, the lower limit for the height of surfaces to be scanned, and the scan quality.

1) Click Controller icon on the tool bar (see Figure 2-5(b)), make sure the Controller window opens. When Dr. PICZA is started, the Controller window (Figure 3-5(a)) is already open.



(a)



(b)

Figure 3-5 Remote Controller, Scan Pitch and Scanning Path

2) Make the settings for X scan pitch and Y scan pitch (about scan pitch refer to Figure 3-5 (b)).
3) Set the height of Z Bottom
4) Select the desired scan quality.

Click the Fine button or the Draft button or Smart Scan to activate the desired quality level (see Figure 3-5 (a)).

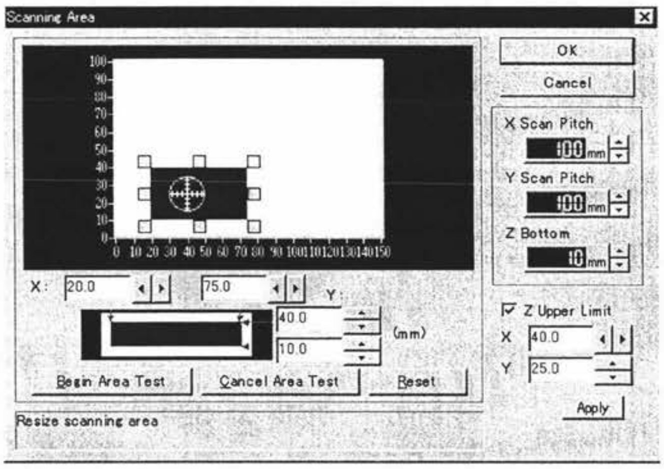
Draft button causes bi-directional scanning to be carried out along the x-axis. Which results in shorter scan times than unidirectional scanning, but the scanning precision is reduced.

Fine button causes scanning along the x-axis to perform in the same direction at all times (unidirectional scanning). This is more precise than bi-directional scanning, but scanning time is correspondingly longer.

Smart button automatically adjusts the scanning area to fit the scan object. Dr PICZA will automatically determines a rectangle circumscribing the scan object and sets the scanning area (along the x and y axes). No adjustment is made for height (z axis). To limit the scanning area in the height direction activate Z Upper Limit button.

6. Set the scanning area and start scanning

Click Scanning Area in the remote controller, the Scanning Area Dialog Box appears (Figure 3-6). Make the setting for the scanning area. The location of the scanning area and its size can be altered by either dragging the blue frame on the screen or by entering the numerical values.



X: 20.0 75.0
Y: 40.0 10.0 (mm)
X Scan Pitch 100 mm
Y Scan Pitch 100 mm
Z Bottom 10 mm
☑ Z Upper Limit
X 40.0
Y 25.0
Resize scanning area

Figure 3-6 Scanning Area Dialogue Box

The cursor is used to specify the highest position of the scan object by pointing and clicking. Once all the settings have been made it is time to check if the area that has been set is the right size for the object.

It should be noticed that the maximum scanning size of the model is 152.4mm (X) × 101.6mm (Y) × 40.65mm (Z) and the maximum weight of the object of the model is 400g (see Table 2-1).

1) Click Begin Area Test

The sensor moves to a position above an outer point on the scanning area that has been set. Make sure the scan object that has been secured in place lies within the area. When the scanner begins the area test the probe begins testing for the Z upper limit and then the outside perimeter of the scan area.

It is usually best to watch the probe if any adjustments to the scan area are required. The path that the probe follows to check the scan area is shown in Figure 3-7. If the scanning area is too large or too small, adjust the X and Y value for the upper-right (UR) and lower- left (LL) point (refer to Figure 3-6). Once the area is set proper size, the user is ready to begin scanning the object.

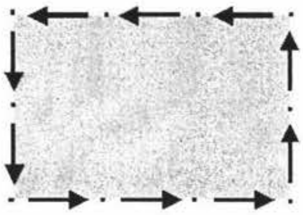


Figure 3-7 The Path of Probe checking the Scan Area

2) Click Z Upper Limit

The cursor is display on the Z upper-limit setting on screen. Specify the highest position of the scan object and then click Apply.

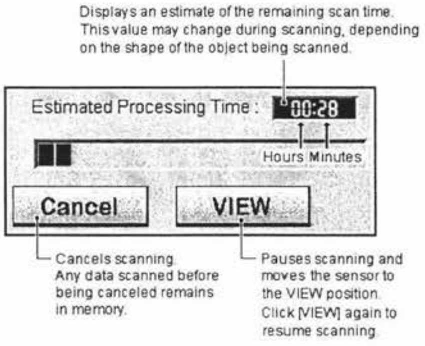
3) The path of scanning object

The probe moves to the lower left corner of the scan area and begins scanning in the X direction at a constant Y value until it reaches the end of the scan area. It then moves up one step in the Y direction (the step sizes in both the X and Y directions are predetermined by the resolution enter by the user) and begins scanning back the way it came. The probe follows this path until it reaches the upper right corner of the scan area where it stops and retracts into its protective housing. The path the probe follows is illustrated in Figure 3-5(b).

4) Scanning the object

After determining the scanning area, click OK. Check the scanning conditions in the Remote Controller window again and then click Scan, scanning starts. During scanning, the estimated processing time window appears shown in Figure 3-8. To cancel the scanning process, click Cancel. Any data scanned before the cancellation remains in the memory. The View function is to resume scanning process again.

Once the scanning process finishes, the 3D graphics of the scanned model will show on the screen (Figure 3-9). It can also be viewed in different angles and in shaded mode.


Displays an estimate of the remaining scan time. This value may change during scanning, depending on the shape of the object being scanned.

Estimated Processing Time : 00:28

Hours Minutes

Cancel VIEW

Cancels scanning. Any data scanned before being canceled remains in memory.

Pauses scanning and moves the sensor to the VIEW position. Click [VIEW] again to resume scanning.

Figure 3-8 The Estimated Processing Time Window

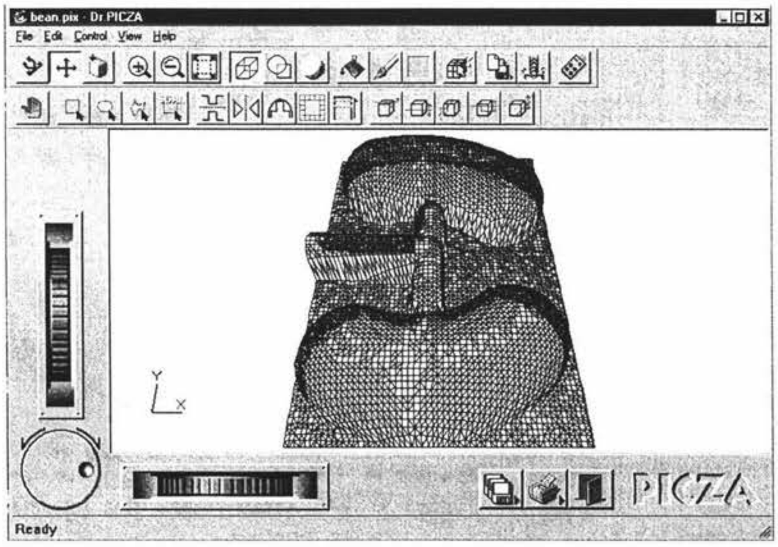


Figure 3-9 3D Graphics of Scanned Model

7. Edit the scanned data

The shape of an object can be edited. It is possible to vary the height, adjust the slant, or perform concave/convex inversion (height inversion) for a desired surface.

8. Save the file

Dr. Picza has a standard file save dialogue window. To save the scanned file, choose Save As from the File menu and then choose the desired location for saving the file, enter a file name, and click Save. The extension ".pix" is appended to the file name.

3.1.2 Output Format of PICZA

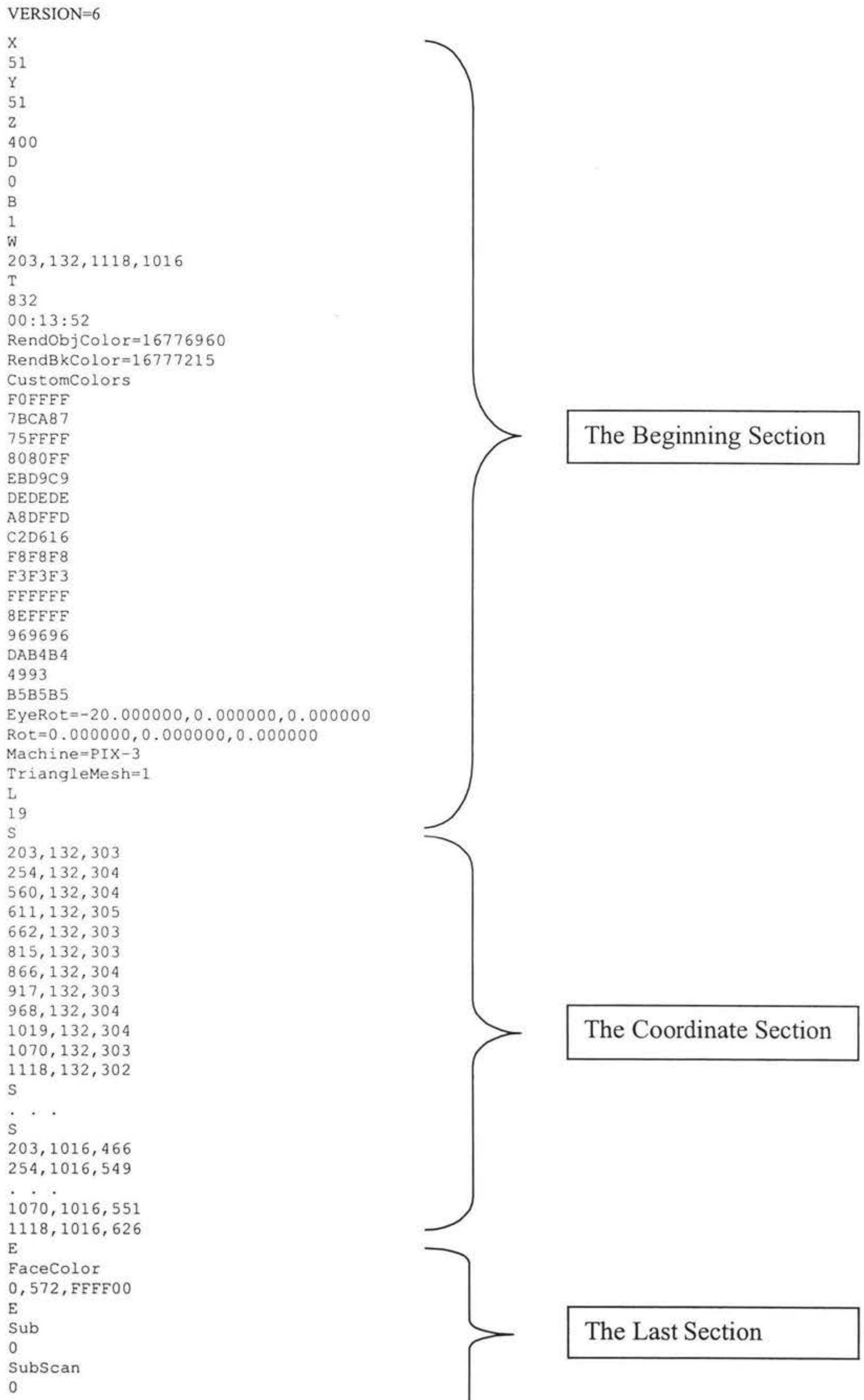
Dr. PICZA can export the scanned data in DXF, VRML, PIX, STL, 3DMF and BMP format. However, considering the available facilities in the Institute of Technology and Engineering at Massey University, DXF format for AutoCAD Release 12, VRML used for web purpose, only PIX format was chosen for future conversion into NC program for CNC Milling.

3.2 The Investigation of the Feasibility to Convert the Output from Picza to CNC Program.

The PIX file is the proprietary format for Dr. PICZA. It can be open with the Microsoft Word Processor, Notepad, Wordpad, Borland C++5.0 and C++Builder 5.0. The example of PIX format file opened with Microsoft Word Processor is illustrated in Figure 3-10.

Through the study of this example file, it was found that the PIX file structure could be divided into three main sections. From the beginning section of the file until the letter "S", it describes the version, scanning area (X, Y range), rendering colour of the scanned object, the scanning machine type and the 3D data type (TriangleMesh).

The coordinate section is the main part about x, y, z coordinate value of scanned object in different section which starts with symbol "S". The last section starts with letter "E", and also gives the face color of the scanned object, finally ends with "0".



```
VERSION=6
X
51
Y
51
Z
400
D
0
B
1
W
203,132,1118,1016
T
832
00:13:52
RendObjColor=16776960
RendBkColor=16777215
CustomColors
F0FFFF
7BCA87
75FFFF
8080FF
EBD9C9
DEDEDE
A8DFFD
C2D616
F8F8F8
F3F3F3
FFFFFF
8EFFFF
969696
DAB4B4
4993
B5B5B5
EyeRot=-20.000000,0.000000,0.000000
Rot=0.000000,0.000000,0.000000
Machine=PIX-3
TriangleMesh=1
L
19
S
203,132,303
254,132,304
560,132,304
611,132,305
662,132,303
815,132,303
866,132,304
917,132,303
968,132,304
1019,132,304
1070,132,303
1118,132,302
S
. . .
S
203,1016,466
254,1016,549
. . .
1070,1016,551
1118,1016,626
E
FaceColor
0,572,FFFF00
E
Sub
0
SubScan
0
```

Figure 3-10 The Example of PIX Format File

Taking a careful investigation into the coordinate value of x, y, z in different sections, it was found that:

- Y value keeps constant in the same section (the data between two "S") and gets increment from one section to another according to the pitch in y direction.
- X value gets increment within each section, but keep the same range and the same increment (based on the X scan pitch) in different section.
- Z value varies within each section. It indicates the variation of Z coordinate of the scanned model within each section.

The PIX file can not be directly accepted by 3D CAD/CAM system such as Solidworks and Camworks. It is necessary to convert PIX file into CNC program if CNC machining is desired.

3.3 Input Format of CNC Machine Tools

3.3.1 CAD / CAM / CAE System Layout at ITE

The CAD/CAM/CAE system at ITE has been setup which is illustrated in Figure 3-11. In this system, by means of Computer-Aided Design and Manufacturing Technologies and Workshop, the ideas or concept design can be converted into the final product. The continuous solid arrow lines indicate that the links between different blocks have been established. However, Dr Picza cannot directly link with CAD and CAM system where is shown in the dashed arrow lines. Therefore, it is essential to design a Data Convert System connecting them to make the whole system function effectively.

Figure 3-12 indicates the layout of the CAD/CAM/CAE laboratory and workshop. There are fourteen Personal Computers installed CAD/CAM/CAE softwares for students and staff. All PCs (at the top left-hand side) are connected with CNC Mill and CNC Lathe through RS 232 cable (Figure 3-15). One PC is connected with the 3D digitizer and a 3D Plotter (refer to Figure 3-2). All PCs are also hooked on the Massey University's Network.

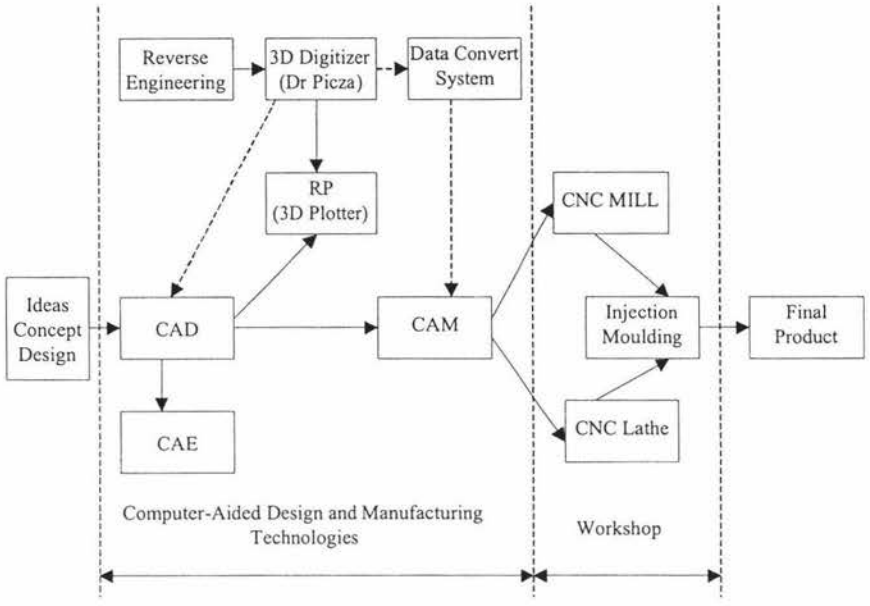


Figure 3-11 CAD/CAM/CAE System at ITE

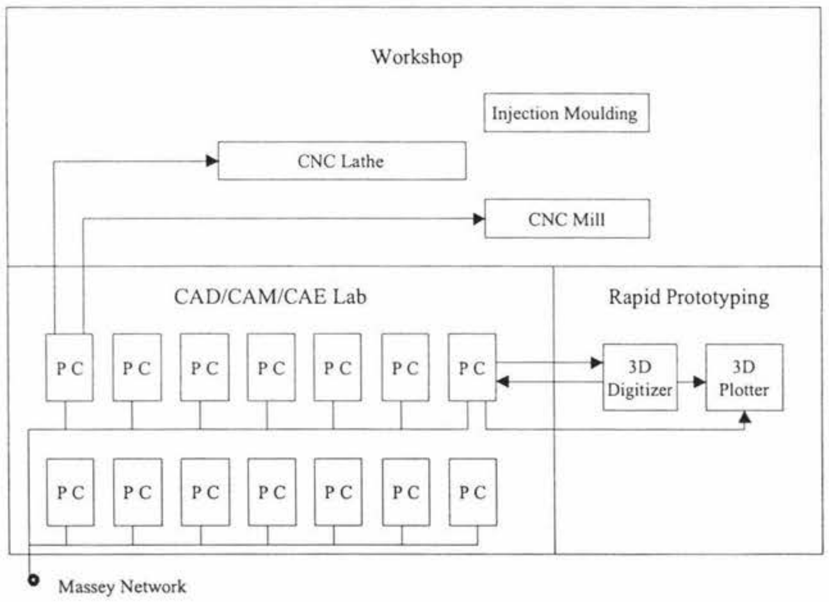


Figure 3-12 ITE CAD/CAM/CAE Laboratory and Workshop Layout

3.3.2 Input Format to Bridgeport CNC Mill

3.3.2.1 Functions of Heidenhain TNC 145 Controller

The existing CNC Mill Controller at ITE is a Bridgeport Series I Interact controller that is a 2_-axis variant milling, drilling and boring machine manufactured by Bridgeport Machines Division of Textron Ltd, England (Figure 3-13).

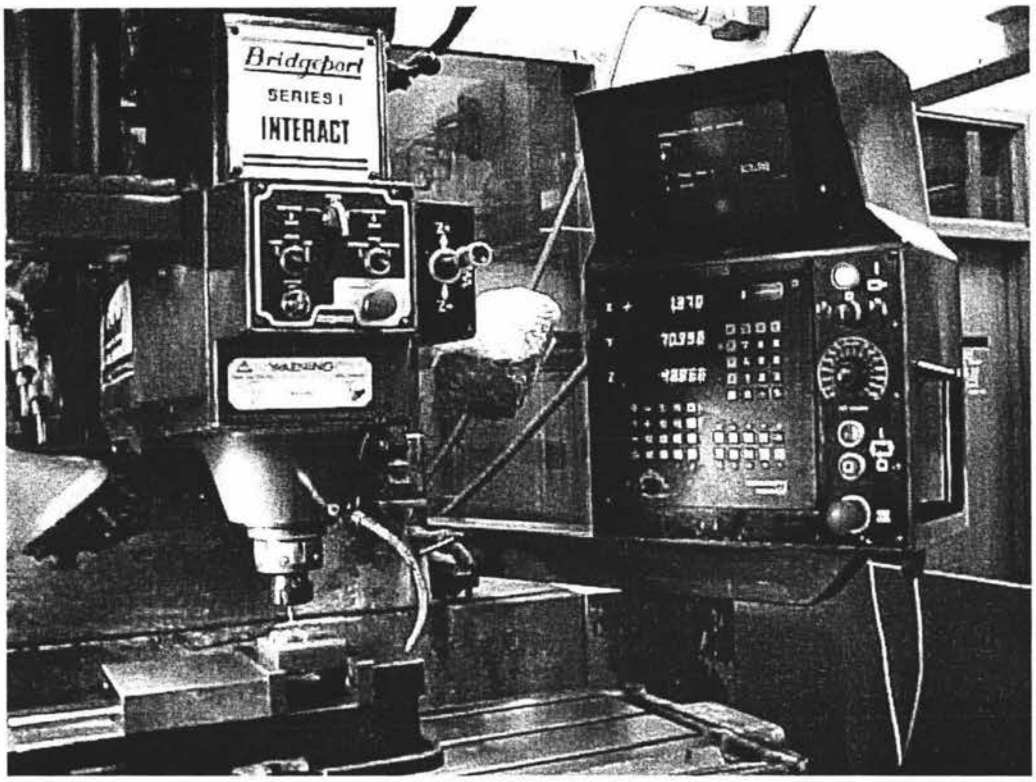


Figure 3-13 Bridgeport CNC Mill

The operator's machine controls are Heindenhain TNC 145 controller located on two conveniently placed panels, i.e. Spindle Controls on the front of the milling head and Machine Function Controls on the Pendant Control (Figure 3-14). At the top of Pendant Control, is the Visual Display Unit that is mainly for showing the information about programming in plain language. And at the bottom of Pendant Control, there are many key buttons. Press these buttons, operator can operate machine to carry out different functions, such as, entry value, datum setup, programming editing and running.

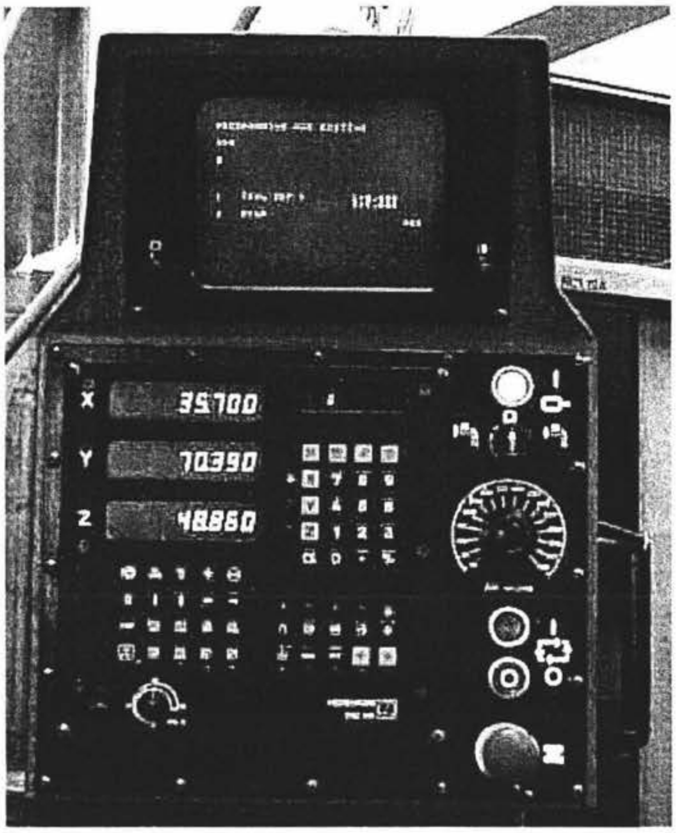


Figure 3-14 The Visual Display Unit and Pendant Control

The Bridgeport CNC Mill provides direct programming on the machine by the machine operator. Programming can be performed either manual operation on the Pendant control or external programming via the DNC cable connected with the CAM system.

If the machine parts are created by CAD software and NC code is generated by CAM software, these NC codes can be run on the machine automatically and then produce the final product (see 3.2.2.8).

3.3.2.2 CNC Machine Tool Initialization

Machine initialization is required each time when the main switch is turned on. The steps to carry out the initialization for the Bridgeport CNC Mill in ITE are:
"POWER INTERRUPTED".

Pressing of the [CE] key will clear the control and place it in the ✋ manual mode. The dialogue display will then read:

"PASS OVER Z REFERENCE MARK".
"PASS OVER Y REFERENCE MARK".
"PASS OVER X REFERENCE MARK".

This action is necessary to reference the axes location, enabling the software limits for axis stroke to effective. To pass over the reference points proceed as follows:
Press POWER ENABLE button and Press CYCLE START button, Z axis will move in rapid traverse to its Z+ limit of travel. Z axis dialogue display will disappear.

Press CYCLE START button, Y axis will move in rapid traverse to its Y + limit of travel. Y axis dialogue display will disappear.
Press CYCLE START button X axis will move in rapid traverse to its X+ limit of travel. X axis dialogue display will disappear.

It has to be noticed that, if prior to switch on, one two or three of the axes are sitting on extreme 'PLUS' limit then the dialogue display will be shown with a light background. This indicates which axis if any is sitting on limit.

By pressing [CE] key and then Power Enable button and then Cycle Start button, the axis displayed with light background will move off the limit. This procedure will be repeated for the number of axes on limit. Then normal start up procedure can be carried out.

3.3.2.3 Absolute and Incremental Dimensions

All absolute dimension (see the example in Figure 3-15) are related to a single "absolute datum" (absolute reference position), whereas incremental dimensions (Figure 3-16), also called chain dimensions, are always related to the previously established position relative datum.

Programming in absolute dimensions offers the advantage of being able to perform geometric amendments of single positions without influencing the remaining positions. Re-entry into an interrupted program after power failure or any other defect is also much simple with absolute programming.

Furthermore a suitable location of the workpiece dataum can help to dispense with negative values.

On the other hand, incremental programming eliminates calculation work in many cases. The incremental mode-key I must only be pressed before axis selection. With incorrect operation, press DEL key and re-commence with I key.

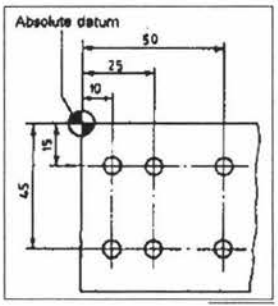


Figure 3-15 The Example of Absolute Dimension

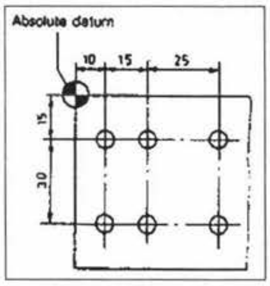


Figure 3-16 The Example of Incremental Dimension

3.3.2.4 Setting Component Datum

Once the machine initialization is completed (in *3.2.2.2*), the next step is to set the machining datum which involves the following steps.

- Press "manual mode" key or press "electronic handwheel" key.
- Move X, Y and Z axes manually to chosen datum position for component.
- Press X key, key in datum value for X axis datum setting, and press ENT key.
- Repeat step 3 for Y and Z axis datum setting, the component datum setting will be finished.

3.3.2.5 Tool Length and Radius Compensation

1. Tool Length Compensation

Tool length compensation, as its name indicates, is used to compensate for tool length difference. It can perform this function in two directions, either away from the part or toward the part. The major advantage of this feature is that it provides the programmer with the ability to write a complete program without knowing exactly the length of the tools to be used.

When entering the tool length compensation. It should be decided as to whether the workpiece surface is to be declared as "zero" or related to zero by means of another value.

1) Workpiece surface Z = 0

- Insert longest tool into machine spindle.
- Position tool to touch worksurface using handwheel or power feed mode.
- Key in zero value to entry value display.
- Press TOOL DEF key and enter value of tool number.
- Enter tool length L = 0.
- Define relationship length of other tools in correct sequence using the following procedure.
- Insert tool number "n" into spindle.
- Position tool to work surface
- Transfer value shown in Z axis DRO display (including sign) into entry value display by pressing $\boxed{+}$ key.
- Press TOOL DEF key and enter this value into tool definition, TOOL LENGTH L =

2) Workpiece surface Z not equal to 0

- Insert longest tool into machine spindle.
- Position tool to touch worksurface using handwheel or power feed mode.
- Key in Z + 50 value to entry value display.
- Press TOOL DEF key and enter value of tool number.
- Enter tool length L = 0.

- Define relationship length of other tools in correct sequence using the following procedure.
- Insert tool number "n" into spindle.
- Position tool to work surface.
- Note down value shown in Z axis DRO display (including sign).
- Calculate compensation value "L" as follows:

  L = (position display value Z) – (surface position) i.e.

  (+40) – (+50) = -10
- Key in this value into entry value display.
- Press TOOL DEF key and enter this value into tool definition, TOOL LENTH L =.

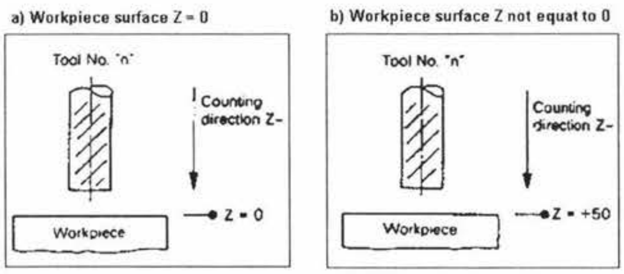


Figure 3-17 Tool Length Compensation

2. Tool Radius Compensation (Single Axis)

Tool radius compensation is to ensure that the workpiece finished contour is actually and precisely milled. As a result, the milling cutter centre point has to traverse along the equidistant path or at uniform distance following the finished contour. On most modern CNC systems, the equidistant path is automatically calculated by cutter radius compensation. This compensation demands the dimension of the milling cutter radius and which side (left or right, see Figure 3-18) of the programmed finished contour.

Single axis tool radius compensation is illustrated in Figure 3-18, and Traversing direction is indicated in Figure 3-19.

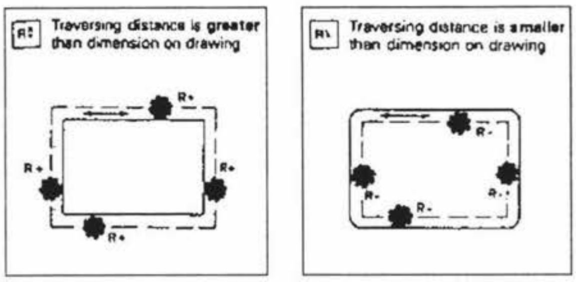

Traversing distance is greater than dimension on drawing
R+
R+
R+
R+
Traversing distance is smaller than dimension on drawing
R-
R-
R-
R-

Figure 3-18 Single Axis Tool Radius Compensation

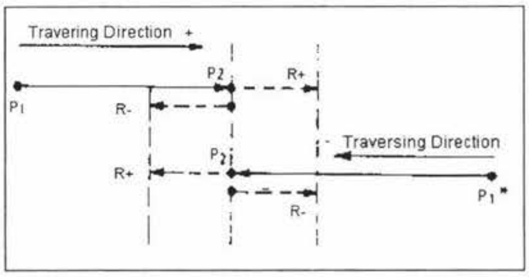


Figure 3-19 Traversing Direction

$\boxed{R^{R}_{+}}$ always extends the traversing distance whereas $\boxed{R^{L}_{-}}$ always decrease the traversing distance. If the traversing direction is changed, the directions for the radius compensation (offset) also changes accordingly. If the tool is re-ground or another tool is inserted, the new tool radius has only to be amended once in the appropriate tool definition-block. The rest of the program remains unchanged.

If $\boxed{R^{R}_{+}}$ and $\boxed{R^{L}_{-}}$ are not pressed, no tool radius compensation (offset). This is indicated in the dialogue by "R0."

It has to be noticed that $\boxed{R^{R}_{+}}$ and $\boxed{R^{L}_{-}}$ are used simply for R+ and R- functions in single axis.

3. Tool Radius Compensation (Multiple Axes)

Multiple Axes tool radius compensation is shown in Figure 3-20.

$\boxed{R^{R}_{+}}$ always places tool to the right of part surface related to the direction of travel.

$\boxed{R^{L}_{-}}$ always places tool to the left of part surface related to the direction of travel.

If the traversing direction is changed, the radius compensation (offset) also requires changing.

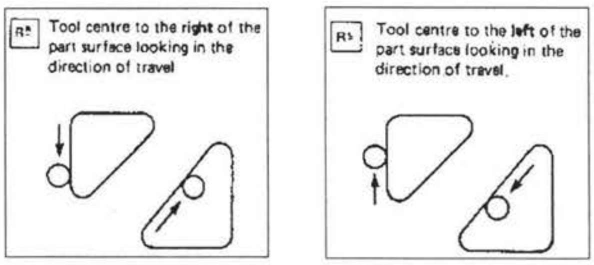

R
Tool centre to the right of the part surface looking in the direction of travel
R
Tool centre to the left of the part surface looking in the direction of travel.

Figure 3-20 MultipleAxes Tool Radius Compensation

If the tool is required or another tool is inserted the new tool radius has only to be amended once in the appropriate tool definition block. The rest of the program remains unchanged.

If $\boxed{R^R}$ or $\boxed{R^L}$ are not pressed then no tool radius compensation is provided (this is indicated in the dialogue by R0).

3.3.2.6 Programming Data Transfer

Data input and output from / to a remote device using the special cable and external key control. The CNC Mill in ITE is equipped with a (RS232) – compatible data input and output which is activated via the V-24 Ext key (refer to Figure 3-21).

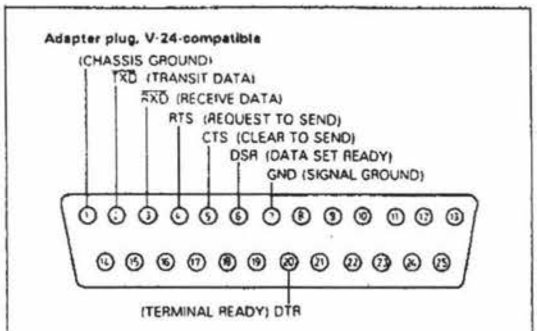


Figure 3-21 RS232 Data Transfer Cable

A complete program or part program can be stored in the hard disk or floppy disk for further usage. Conversely the program may be reloaded to the PC when required through V-24 compatible connection. There is a limitation on CNC programs sent to the CNC Mill in ITE. The controller can only accept programs not more than on thousands lines for distributed numerical control.

3.3.2.7 The Milling Process

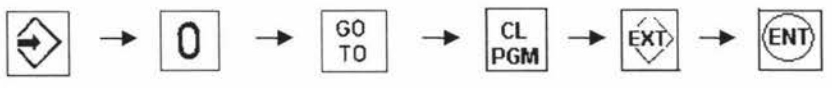
Generally, the milling procedure can be shown in the following steps:

1. Secure the object on the working table. The object is tightened and fixed in place with the aid of steel clamps and bolts.

2. Ensure the machine is turned on. Check the key on the side of the machine is turned on and the red switch is illuminated.

3. Clear the control panel and decide the zero position of the cutter (setup datum). This is done by pressing the button with CE on it and then using the joystick or electrical wheel to move the object to get a zero point under the cutter. Once a zero position is decided on the datum is set by entering the X, Y and Z zero points into the control panel and pressing [Enter] key.

4. Enter a program into the controller. By pressing the following buttons sequentially, the CNC Mill will be ready to receive external data input.



5. Select correct setting on the serial communications dialogue. MasterCam Mill 5.0 will be chosen for the data input software. On the serial COM screen the following settings should be in place (Figure 3-22). Once all settings are selected correctly, pressing [send] button, the program begins scrolling onto the screen of the CNC Mill.

Format	Connector	Parity
ASCII	COM2	EVEN
Data Bit	**Baud Rate**	**Stop Bits**
7	2400	1

Figure 3-22 Setting Up the Serial Communications

6. Run the program in blocks. On the Pendant Control of the milling machine, pressing the → button makes the program run line by line. This is done to ensure any mistakes in the program won't result in the cutting tool or workpiece being damaged through an incorrect path being entered.

3.3.2.8 Format of the CNC Program to Heidenhain TNC 145 Controller

The only code that accepted by the milling machine is NC code. NC code is a series of instructions consist of cutters parameters, machining parameters, X, Y, Z coordinate values, and other symbols. The program shown below (Figure 3-23) is an example of a simple NC program used to mill 100_100_40 metal block into a workpiece which has 10 mm external contour 20mm deep step shape (Figure 3-24). The absolute coordinate system is used. The actual path the cutter followed can be shown in Figure 3-25.

```
0 Program an external contour
1 A Z+50.000 R0 F9999 M
2 A X-20.000 R0 F9999 M
3 A Y-20.000 R0 F9999 M05
4 TOOL DEF 1 L
5 TOOL DEF 1 R +10.000
6 STOP M25
7 TOOL CALL 1 Z S 250.000
8 A Z-20.000 R0 F9999 M03
9 A Y+10.000 R- F9999 M
10 A X-2.000 R- F9999 M
11 A X+90.000 R+ F40 M
12 A Y+90.000 R+ F40 M
13 A X+10.000 R+ F40 M
```

Moves the cutter to its zero position above the workpiece. F9999 is the feed rate at which the cutter follows the path. M05 indicates spindle stop during program. M represents machine code.

Tool DEF: defines what tool is to be used, here is tool 1. Values are placed after L if required tool compensation. R+10,000 means the radius of the cutting tool is 10mm.

Tool 1 is called up. S means spinle speed is at 250 rpm

This section is the main positioning commands for the cutter approach to the workpiece and milling the contour. M03 is the spindle start during program when spindle rotation required.

```
14 A Y+10.000 R+ F40 M
15 A Z+50.000 R0 F9999 M05
16 A X-20.000 R0 F9999 M
17 A Y-20.000 R0 F9999 M
18 STOP M25
```

Traverse the cutter back to its zero position.

Stop M25: stop the milling machine and end the program

Figure 3-23 Example of the CNC Mill Program Format

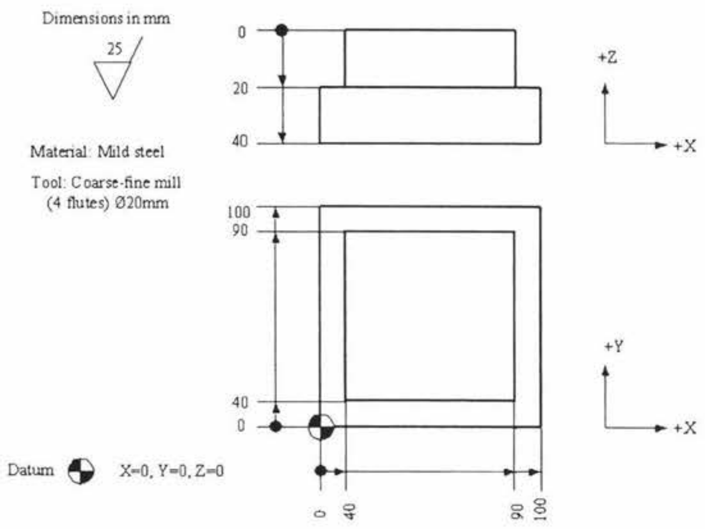


Figure 3-24 The Drawing of Workpiece

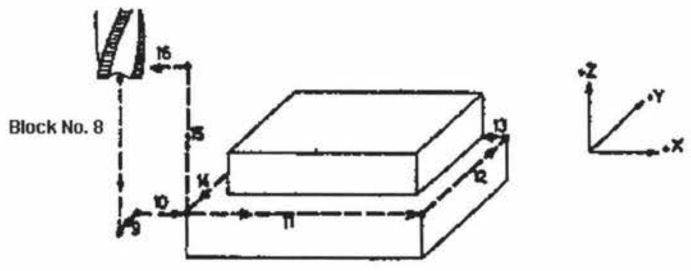


Figure 3-25 The Path of the Cutter

3.4 Data Conversion

3.4.1 Comparison of Scanned Data and the CNC Program Format to Hedenhain TNC 145 Controller

The comparison of scanned data and the CNC program format is shown in the Figure 3-26. Apparently, the scanner output will always be longer than the input that's required for the CNC Mill. The middle section of the code at right side is the exact NC equivalent of the scanner output coordinate values on the left. This section is the key part to convert the scanner output format into the CNC program format. As for the beginning section and the last section of the scanned data output and CNC program format, they are very different. It has to be built according their different requirements.

As can be seen on the left column in Figure 3-26, it is a small section of a scanned data. The beginning and last part of the scanned data contain the model information which provides the general information such as area size, colour, etc, for file conversion. Therefore, the information from these parts is important.

Compared with the scanned data, CNC program format file (on the right column) can also be divided into three parts. The first part (line 1~3 in Figure 3-26) contains miscellaneous functions such as Tool Definition, Tool Call, "zero point" position etc.

The second part is the part which deals with X, Y and Z coordinates in different cross sections. Considering CNC Mill only has 2_ axes, at the beginning of each cross section, the cutter is positioned by X, Y and Z value (line 3 and line4).

Then, the cutter moves on X-Z plane until it finishes the section. When starting a new section, the cutter moves to another position determined by a new X, Y and Z value (line 47 and line 48). Eventually, the cutter will finish producing all of the cross sections of the scanned model.

The last part (line 703 and 704) is to move the cutter to a safe position and stop the machine.

Scanned Data (PIX format)

```
VERSION=6
X
18
Y
18
Z
400
D
0
B
1
W
528,203,1575,1595
T
4720
01:18:40
RendObjColor=16776960
RendBkColor=16777215
CustomColors
FFFFFF
FFFFFF
. . .
FFFFFF
FFFFFF
EyeRot=-20.000000,0.000000,0.000000
Rot=0.000000,0.000000,0.000000
Machine=PIX-3
TriangleMesh=1
L
79
S
528,203,303
546,203,303
564,203,304
582,203,304
600,203,303
618,203,304
636,203,303
654,203,303
672,203,304
690,203,303
708,203,303
726,203,304
744,203,303
762,203,303
780,203,304
798,203,303
816,203,303
834,203,304
852,203,303
888,203,303
906,203,304
996,203,304
. . .
1428,203,301
1446,203,302
1500,203,302
1518,203,303
1536,203,302
1575,203,302
S
. . .
S
. . .
E
FaceColor
0,8385,FFFF00
E
Sub
0
```

CNC Program Format (scale = 1)

```
SubScan
0
0 Program for cutting bean
1 TOOL DEF 1 L0.000 R0.000
2 TOOL CALL 1 Z S710.0
3 Z 308 R0 F9999 M03
4 L X528 Y203 R0 F9999 M
5 L X528 Z303 R0 F900 M
6 L X546 Z303 R0 F900 M
7 L X564 Z304 R0 F900 M
8 L X582 Z304 R0 F900 M
9 L X600 Z303 R0 F900 M
10 L X618 Z304 R0 F900 M
11 L X636 Z303 R0 F900 M
12 L X654 Z303 R0 F900 M
13 L X672 Z304 R0 F900 M
14 L X690 Z303 R0 F900 M
15 L X708 Z303 R0 F900 M
16 L X726 Z304 R0 F900 M
17 L X744 Z303 R0 F900 M
18 L X762 Z303 R0 F900 M
19 L X780 Z304 R0 F900 M
20 L X798 Z303 R0 F900 M
21 L X816 Z303 R0 F900 M
22 L X834 Z304 R0 F900 M
23 L X852 Z303 R0 F900 M
24 L X888 Z303 R0 F900 M
25 L X906 Z304 R0 F900 M
26 L X996 Z304 R0 F900 M
27 L X1014 Z303 R0 F900 M
28 L X1086 Z303 R0 F900 M
29 L X1104 Z304 R0 F900 M
30 L X1122 Z303 R0 F900 M
31 L X1140 Z304 R0 F900 M
32 L X1158 Z303 R0 F900 M
33 L X1194 Z303 R0 F900 M
34 L X1212 Z302 R0 F900 M
35 L X1320 Z302 R0 F900 M
36 L X1338 Z301 R0 F900 M
37 L X1356 Z301 R0 F900 M
38 L X1374 Z300 R0 F900 M
39 L X1392 Z300 R0 F900 M
40 L X1410 Z301 R0 F900 M
41 L X1428 Z301 R0 F900 M
42 L X1446 Z302 R0 F900 M
43 L X1500 Z302 R0 F900 M
44 L X1518 Z303 R0 F900 M
45 L X1536 Z302 R0 F900 M
46 L X1575 Z302 R0 F900 M
47 Z 313 R0 F9999 M03
48 L X528 Y221 R0 F9999 M
49 L X528 Z308 R0 F900 M
. . .
672 Z 596 R0 F9999 M03
673 L X528 Y1595 R0 F9999 M
674 L X528 Z300 R0 F900 M
. . .
703 Z 600 R0 F9999 M
704 STOP M25
```

Figure 3-26 The Comparison of Scanned Data and CNC Program Format

Based on the above analysis and the comparison of the scanned data and CNC program format, two tasks were set to convert the scanned data into a CNC program:

1) Establish an intermediate data file from the scanned data.

2) Output the correct CNC program from the Intermediate data file.

3.4.2 Screening of Scanned Data into the Intermediate Data File

As stated in ***3.4.1***, the miscellaneous information at the beginning and last part of the scanned data were carefully studied and the whole scanned data were screened out into an intermediate data file. Figure 3-27 illustrates a small section of an intermediate file.

```
528,203,303
546,203,303
564,203,304
582,203,304
600,203,303
618,203,304
636,203,303
654,203,303
672,203,304
690,203,303
708,203,303
726,203,304
744,203,303
762,203,303
780,203,304
798,203,303
816,203,303
834,203,304
852,203,303
888,203,303
906,203,304
996,203,304
1014,203,303
1086,203,303
1104,203,304
1122,203,303
1140,203,304
1158,203,303
1194,203,303
1212,203,302
1320,203,302
1338,203,301
1356,203,301
1374,203,300
1392,203,300
1410,203,301
1428,203,301
1446,203,302
1500,203,302
1518,203,303
1536,203,302
1575,203,302

528,221,308
546,221,312
564,221,310
582,221,312
600,221,312
618,221,311
636,221,310
654,221,311
672,221,307
690,221,304
708,221,303
744,221,303
762,221,301
780,221,303
798,221,304
816,221,303
852,221,303
870,221,355
888,221,386
906,221,412
924,221,426
942,221,436
960,221,439
978,221,437
996,221,440
1014,221,437
1032,221,421
1050,221,403
1068,221,389
1086,221,328
1104,221,303
1122,221,303
1140,221,316
1158,221,307
1176,221,322
1194,221,302
1230,221,302
1248,221,303
1266,221,303
1284,221,302
1302,221,302
1320,221,303
1338,221,302
1356,221,303
1374,221,301
1392,221,298
1428,221,298
1446,221,299
1464,221,300
1482,221,302
1500,221,303
1518,221,302
1554,221,302
1572,221,301
1575,221,302
. . .

528,1595,300
546,1595,302
564,1595,301
582,1595,302
618,1595,302
636,1595,301
654,1595,302
672,1595,302
690,1595,303
708,1595,302
798,1595,302
816,1595,301
852,1595,301
870,1595,302
978,1595,302
996,1595,301
1068,1595,301
1086,1595,302
1104,1595,301
1356,1595,301
1374,1595,300
1392,1595,301
1410,1595,301
1428,1595,302
1464,1595,302
1482,1595,301
1536,1595,301
1554,1595,300
1575,1595,300
```

Figure 3-27 the Intermediate Data File

The reason to establish an intermediate data file lies in that it is easy to be converted into the CNC program format. Therefore, following two steps will be taken as the first procedure to screen the scanned data into the intermediate data file.

1) Read the scanned file (PIX Format) from the beginning, remove the unnecessary miscellaneous information.

2) Output the X, Y, Z coordinate value between each pair of "S" into the intermediate data file. Once the programme reaching the end of the file, retrieving data will stop.

Figure 3-27 is the example of the intermediate data file ("Test1" is used in the source code). From left to right on each line, the digital values respectively represent X, Y and Z. It is very clear to find that Y value keeps constant on each section. Later, this advantage will be used for recognising different sections during converting the intermediate data file into the CNC program format.

3.4.3 Intermediate Data File Conversion

3.4.3.1 Definition of Machining Parameters

After the scanned data is converted into the intermediate data file, it will reach the key step: intermediate data file conversion. At this stage, it includes definition of machining parameters and generation of the correct output CNC program format.

The definition of machining parameters refers to tool definition, tool compensation, cutter radius, "zero point" cutter position spindle speed, feed rate (range 1-9999 mm/min), and so on.

Considering the tools are changed manually and not by automatic means, the definition of tool, the radius of the cutter and the speed of the spindle are set for instance (see the right column in Figure 3-26):

1 TOOL DEF 1 L0.000 R0.000

```
2 TOOL CALL 1 Z S710.0
3 Z 308 R0 F9999 M03
```

The Z value in line 3 will automatically be changed by program according to different sections. This Z value can be considered as the cutter home position for each section. In order to avoid cutter cutting the previous section of the model when cutter returns, the adjustment of this Z value, controlled by program, is set 5mm above the maximum Z value of current section.

3.4.3.2 Generate the Correct Output CNC Program Format

After the machining parameters are defined, next step is to generate the correct output CNC program format, in other words, to convert the coordinate X, Y and Z value from the intermediate data file into the required CNC program format.

As can be seen in Figure 3-26, the beginning line for each cross section in CNC Mill Input File is formatted like line 4 on the right column: "4 L X528 Y203 R0 F9999 M". This can be done by insertting letter "X" and "Y" in front of the corresponding values in Figure 3-27. Z value in Line 3 and X and Y value in line 4 (Figure 3-26) will define the starting cutting position or "zero point" position of the cutter.

After the beginning lines, only X and Z value will be converted into the format like line 5: "5 L X528 Z303 R0 F900 M". This process will continue until the new Y value is encountered which means a new cross section starting again. Before the cutter moves to a new cross section, the cutter has to lift up to enough height to avoid cutting the previous section of the model.

The last line for each section will be like line 47: "47 Z 313 R0 F9999 M03". Following this line, a new cross-section will start again. The procedure will repeat until reaching the end of line of the intermediate data file. Finally, the programme will automatically output the last line like "704 STOP M25". Up to here, all of the conversion process will be finished. The whole output CNC format file will be like the right column in Figure 3-26.

Chapter4 Software Development

4.1 Introduction of C++

Nowadays, C++ has become the language of choice for worldwide professional programmers because it represents the current state of the art as applied to computer programming languages. C++ combines power with flexibility, efficiency with elegance, and tradition with innovation. There is very little of the "boat," or redundant features, that are so common in other programming languages. In fact, one of the most important aspects of C++ is its streamlined design.

Because C++ was designed for professional programming, C++ is not the easiest programming language to learn, but one of the best programming languages to learn. Once C++ have been mastered, the computer will virtually completely be controlled over. Above all, C++ is one of the most powerful programming languages ever invented.

C++ is built upon the foundation of C, but it is a superset of C. Indeed, all C++ compilers can also be used to compile C programs. Specifically, C++ is an expanded and enhanced version of C that embodies the philosophy of object-oriented programming. C++ also includes several other improvements to the C language, like an extended set of library routines. However, much of the spirit and flavour of C++ is inherited directly from C. Understanding the "how and why" behind C will be helpful to fully understand and appreciate C++.

4.1.1 The Creation of C

The C language ever shook and impacted the computer world because it fundamentally changed the way programming was approached and thought about. C is considered by many years to be the first modern "programmer's language." Prior to the invention of C, computer languages were generally designed either as academic exercises or by bureaucratic committees.

However, C is different. It was designed, implemented, and developed by real, working programmers, and it reflected the way they approached the job of programming. Its features were honed, tested, thought about and rethought by the people who actually used the language. This process resulted in a language that programmers liked to use. Soon later, C quickly attracted many followers who had a near-religious zeal for it, and it found wide and rapid acceptance in the programmer community. In short, C is a language designed by and for programmers.

C was invented and first implemented by Dennis Ritchie on a DEC PDP-11 using the UNIX operating system. C is the result of a development process started with an older language BCPL, which was developed by Martin Richards. BCPL influenced a language called B, Invented by Ken Thompson, which led to the development of C in the 1970s (Schildt, 1998).

For many years, the de facto standard for C was the one supplied with the UNIX version 5 operating system and described in The C Programming Language, by Brian Kernighan and Dennis Ritchie (Prentice-Hall, 1978). A committee was established in the beginning of the summer of 1983 to work on the creation of an ANSI (American National Standards Institute) standard that would define once and for all the C language. The final version of the standard was adopted in December 1989, the first copies of this became available in early 1990. That is the ANSI standard C, which becomes the foundation of C++.

C is thought of as a middle-level language because it combines elements of high-level languages, such as Pascal or Modula-2, with the functionality of assembler.

From a theoretical point of view, a high-level language attempts to give the programmer everything he or she could possibly want. A low-level language provides nothing other than access to the actual machine instructions. A middle-level language gives the programmer a concise set of tools and allows the programmer to develop higher-level constructions on his or her own. A middle-level language offers the programmer built-in power, coupled with flexibility.

Being a middle-level language, C can manipulate the bits, bytes, address, and ports that are the constituent components of the computer. That is, C does not attempt to buffer the hardware of the machine from program, to any significant extent. For example, unlike many high-level languages, which can operate directly on strings of characters to perform a multitude of string manipulations, C can directly operate only on individual characters. Furthermore, in most high-level languages there are built-in statements for reading and writing disk files. In C all of these procedures are performed by calls to library routines that are not, technically, part of the language. This approach increases C's flexibility. Although all ANSI standard compilers supply functions capable of performing disk I/O, your programs can bypass them if they choose.

C allows and needs the programmer to define routines for performing high-level operations. These routines are called functions, and they are very important to the C language. In fact, functions are the building blocks of both C and C++.

C is a structured language. The most distinguishing feature of a structured language is that it uses blocks. A block is a set of statements that are logically connected. A structured language allows a variety of programming possibilities. It supports the concept of subroutines with local variables.

4.1.2 Understanding the Need for C++

Since C is a successful and useful computer programming language, why was there a need for something else? The answer is complexity. Throughout the history of programming, the increasing complexity of programs has driven the need for better ways to manage that complexity. C++ is a response to that need.

With structured languages, it was, for the first time, possible to write moderately complex programs fairly easily. However, once a project reaches a certain size, its complexity exceeds what a programmer can manage. By the late 1970s, many projects were near or at this point. To solve this problem, a new way to program began to emerge. This method is called object-oriented programming (OOP). Using OOP, a programmer could handle larger programs. The trouble was that C did not support object-oriented programming. The desire for an object-oriented version of C ultimately led to the creation of C++. The purpose of C++ is to allow this barrier to be broken and to help the programmer comprehend and manage large, more complex programs.

4.1.3 C++ Is Born

In response to the need to manage greater complexity, C++ was born. Bjarne Stroustrup invented it in 1979 at Bell Laboratories in Murray Hill, New Jersey. He initially called the new language "C with Classes," However, in 1983, the name was changed to C++.

C++ contains the entire C language. It includes all of C's features, attributes, and benefits. It also adheres to C's philosophy that the programmer, not the language, is in charge. At this point, it is critical to understand that the invention of C++ was not an attempt to create a new programming language. Instead, it was an enhancement to an already highly successful language.

Most of the additions that Stroustrup made to C were designed to support object-oriented programming. In essence, C++ is the object-oriented version of C. By building upon the foundation of C, Stroustrup provided a smooth migration path to OOP. Instead of having to learn an entirely new language, a C programmer needed to learn only a few new features to reap the benefits of the object-oriented methodology.

But C is not the only language that influenced C++. Stroustrup states that some of its object-oriented features were inspired by another object-oriented language called Simula67. Therefore, C++ represents the blending of two powerful programming methods.

Although C++ was initially designed to aid in the management of very large programs, it is in no way limited to this use. In fact, the object-oriented attributes of C++ can be effectively applied to virtually any programming task.

4.1.4 The Evolution of C++

Since C++ was first invented, it has undergone three major revisions, with each revision adding to and altering the language. The first revision was in 1985 and the second occurred in 1990. The third revision occurred during the C++ standardization process. Several years ago, work began on a standard for C++. Towards that end, a joint ANSI and ISO standardization committee was formed. The first draft of the proposed standard was created on January 25, 1994. In that draft, the ANSI/ISO C++ committee kept the features first defined by Stroustrup and added some new ones as well. But, in general, this initial draft reflected the state of C++ at the time.

It is fair to say that the standardization of C++ took far longer than any one had expected when it began. In the process, many new features were added to the language and much small change was made. In fact, the version of C++ defined by the C++ committee is much larger and more complex than Stroustrup's original design. However, the standard is now complete. The final draft was passed out of committee on November 14, 1997. A standard for C++ is now a reality. Compilers are already beginning to support all of the new features.

4.1.5 Object-Oriented Programming

Since object-oriented programming was fundamental to the development of C++, it is important to define precisely what object-oriented programming is. Object-oriented programming has taken the best ideas of structured programming and has combined them with several powerful concepts that allow programmers to organise programs more effectively. In general, when programming in an object-oriented fashion, a problem is decomposed into its constituent parts. Each component becomes a self-contained object that contains its own instructions and data related to that object. Through this process, complexity is reduced and you can manage large programs.

All object-oriented programming languages have three things in common: encapsulation, polymorphism, and inheritance.

1. Encapsulation

All programs are composed of two fundamental elements: program statements (code) and data. Code is that part of a program that performs action, and data is the information affected by those actions. Encapsulation is a programming mechanism that binds together code and the data in manipulates, and that keeps both safe from outside interference and misuse.

In an object-oriented language, code and data may be bound together in such a way that a self-contained black box is created. Within the box are all necessary data and code. When code and data are linked together in this fashion, an object is created. In other words, an object is the device that supports encapsulation.

Within an object, the code, data, or both may be private to that object or public. Private code or data is known as and accessible only by another part of the object. That is, private code or data may not be accessed by a piece of the program that exists outside the object. When code or data is public, other parts of program may access it even though it is defined within an object. Typically, the public parts of an object are used to provide a controlled interface to the private elements of the object.

2. Polymorphism

Polymorphism (from the Greek, meaning "many forms") is the quality that allows one interface to be used for a general class of actions. The specific action is determined by the exact nature of the situation. A simple example of polymorphism is found in the steering wheel of an automobile. The steering wheel (i.e., the interface) is the same no matter what type of actual steering mechanism is used. That is, the steering wheel works the same whether your car has manual steering, power steering, or rack-and-pinion steering. Therefore, once you know how to operate the steering wheel, you can drive any type of car. The same principle can also apply to programming.

More generally, the concept of polymorphism is often expressed by the phase "one interface, multiple methods." This means that it is possible to design a generic interface to a group of related activities. Polymorphism helps reduce complexity by allowing the same interface to be used to specify a general class of action. It is the compiler's job to select the specific action (i.e., method) as it applies to each situation. Programmers don't need to do this selection manually. They need only remember and utilise the general interface.

The first object-oriented programming languages were interpreters, so polymorphism was, of course, supported at run time. However, C++ is a compiled language. Therefore, In C++, both run-time and compile-time polymorphism are supported.

3. Inheritance

Inheritance is the process by which one object can acquire the properties of another object. The reason is that it supports the concept of hierarchical classification. In fact, most knowledge is made manageable by hierarchical (i.e., top-down) classifications.

Without the use of hierarchies, each object would have to explicitly define all of its characteristics. However, using inheritance, an object needs to define only those qualities that make it unique within its class. It can inherit its general attributes from its parent. Thus, it is the inheritance mechanism that makes it possible for one object to be a specific instance of a more general case.

4.1.6 How C++ Relates to Java

There is a relatively new force in computer programming languages, called Java. Java is the language of the Internet. It was strongly influenced by C++. Java and C++ both use the same basic syntax, and Java's object-oriented features are similar to C++'s. In fact, at first glance it is possible to mistake a Java program for a C++ program. Because of their surface similarities, it is a common misconception that Java is simply an alternative to C++. Before continuing, it is necessary to set the record straight on this account.

Although related, Java and C++ were designed to solve differing sets of problems. C++ is optimized for the creation of high-performance programs. Towards this end, C++ compiles highly efficient, executable code. Java is optimized for the creation of portable programs. Obtain portability, Java compiles to pseudo-code (called Java bytecode), which is usually interpreted. This makes Java code very portable, but not very efficient.

Thus, Java is excellent for Internet applications, which must work on a wide variety of computers. But for high-performance programs, C++ will remain the language choice.

Because of the similarities between Java and C++, most C++ programmers can readily learn Java. The skills and knowledge in C++ will be translated into Java easily. Knowing Java will be no trouble to learn C++. Of course, it has to mind that some important differences do exist.

4.2 The Features of Borland C++ 5.0

Borland C++ 5.0 is an object-oriented software development system. It provides developers with tools to migrate to 32-bit operating systems, including support for both 32-bit and 16-bit platforms, a new version of Object Windows Library, and support for both 32- and 16-bit VBX controls. Compatible with the latest user-interface standards and Win95 features, including OLE, registry, and long file names. Also includes advance C++ language features such as namespaces, which eliminates name conflicts between applications and class libraries, and the new C++ keywords, including bool, explicit, mutable, and type name.

4.2.1 System Requirements

- Intel 486-based PC or higher
- Windows 95 or Windows NT
- 8 Mb extended memory or higher
- Hard disk from 25Mb to 100Mb.

4.2.2 The Features to Support Window 95

- 32-bit application (more powerful and robust computing)
- Plug and Play (automatic installation and configuration of hardware devices)
- Long File and Folder names
- CD Plus (provides pictures and images with music on a CD)
- OLE Server (allows data from other applications to be embedded in documents)
- OLE Container (allows data from other applications to be embedded in documents)

- OLE Automation (integration with products which use high level scripting tools and languages)
- Uninstall capability (program can easily be removed from the system)
- Generates Windows 95 logoable applications (developer tool)
- Multithreading (able to run multiple functions within a single application)
- Briefcase - File Synchronization (updates files between portable and desktop PCs)
- Hot Docking (can simply dock/remove from docking station)
- Windows keys on keyboards (function shortcuts through special keystrokes)

4.2.3 The Advantage of Borland C++ 5.0

1. Target DOS, 16-bit Windows, and 32-bit Windows from a single 32 bit hosted IDE.

With Borland C++ users can save time and frustration, because they don't have to use two different environments with different features and different C++ language implementations.

2. New 32-bit IDE now provides seamless debugging and resource editing to boost productivity.

Debug, modify source code, edit your resources, and rebuild users' application all within the same robust 32-bit hosted environment.

3. New Object Components Framework (OCF) simplifies OLE 2 Custom Controls (OCXs).

OCF abstracts the complexities of OLE interfaces, encapsulates OCX containers so users don't have to manage all the low level OLE details, and supports OLE server DLLs which allows users' application to run as much as 5 times faster than an OLE executable.

4. Powerful Borland C++ Project Manager makes managing projects faster and easier.

Let the Borland C++ project manager manage entire project for users, using a single project file to check dependencies and simultaneously build 16- and 32-bit targets including DLLs and libraries in a single build.

5. Only advanced Borland C++ technology provides VBX support for 32-bit applications.

Borland C++ is unique in offering users the ability to reuse a wealth of available 16-bit VBXs in the 32-bit applications by simply recompiling.

4.3 The Requirement of the Software to Be Developed

1. Commonality of Microsoft Product

Since Microsoft products, such as Windows 95 and Microsoft Office 97, were released to the computer world, Windows has become a unique operating system for the user of computer, and Microsoft office has also become the main software that is popularly used in the office environment. The reason that the users of computers have satisfied Microsoft products lies in that they leave everything behind the user and create user friendly interface or desktop that is easy for users to learn and use.

Figure 4-1 is the Microsoft Word application window that we are very familiar with. It only needs users to use mouse and keyboard to click menu bars, drop-down menu, buttons, icons and so on to carry out their different documents works. All of these commonalties can be seen in any Microsoft products.

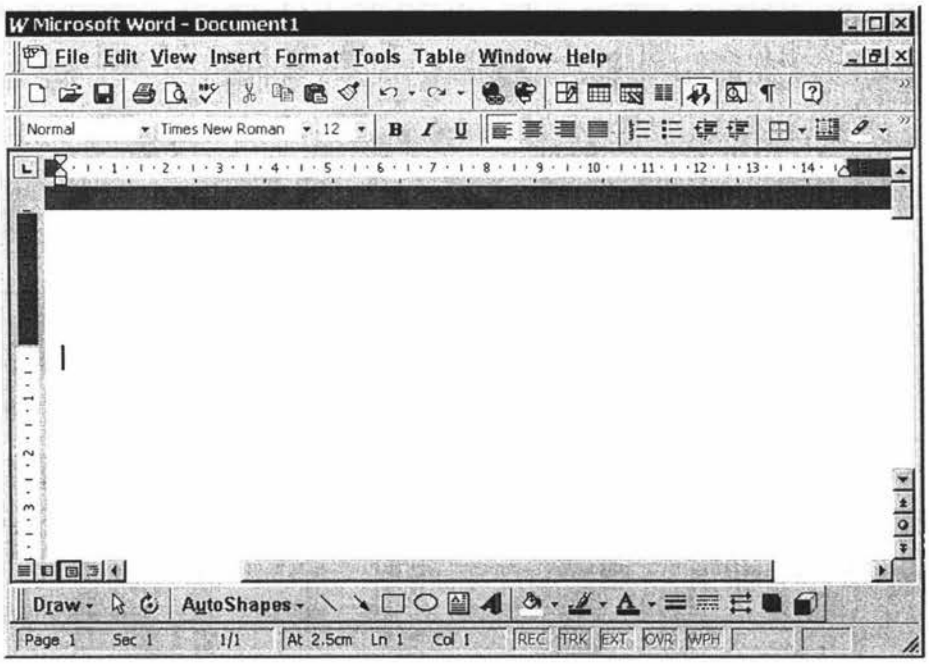

Microsoft Word - Document1
File Edit View Insert Format Tools Table Window Help
Normal
Times New Roman
12
Draw
AutoShapes
Page 1 Sec 1 1/1 At 2.5cm Ln 1 Col 1 REC TRK EXT OVR WPH

Figure 4-1 The Application Window of Microsoft Word

The general system requirement is similar to Borland C++ 5.0 and Borland C++ Builder 5.0.

* Intel Pentium 90 or higher (P166 recommended)
* Microsoft Windows 2000, Windows 98,
  Windows 95, or Windows NT4.0
  with Service Pack 3 or later
* 32MB RAM (64MB recommended)
* Hard disk space:

Standard edition:

  120 MB for compact install

  185 MB for full install

* CDROM drive
* VGA or higher resolution monitor
* Mouse or other pointing device

2. Functions to be developed

The main users of this software will be the academic staff and students in ITE. Considering the available facilities of CAD/CAM laboratory at Institute of Technology and Engineering, the software to be developed should have the following capabilities:

- The software can be used in a console-based application in Windows 95 or 98 operating system.
- The software can be compiled and built with software Borland C++ 5.0 or Borland C++ Builder 5.0
- Input the scanned data file (Dr Picza .PIX format file) and output CNC Mill format file, which can be directly worked on the CNC Mill machine.
- According to the size of model to be scanned, it can scale any size of the physical model that will be machined on the CNC Mill Machine.
- Display the graphical view and dimension of the scanned model.
- Should have an interactive user interface for end users to define machining parameters.

Together with the development of the software, a set of document needs to be developed such as:

- Introduction of this software.
- Instructions to use this software
- File format for read, edit and save the output file for CNC Mill format.
- Automatically load correct number of lines from a large NC program to the NC machine without splitting into different sections of programme.

3. Using a scale factor to control the physical size of a model produced by NC program
In practice, it is often required to consider dividing a model to be scanned into different zones where the scale factor in the local zone may be different from those in the other area. A typical example is that R. T. Lee and W. S. Cheng (2002) presented a multizone scaling method for CAD in Shoe Sole Design. From their investigation, the dimensional grade increments of the front and rear sole zones in length are not the same (Figure 4-2), some soles in a range of sizes have the same heel and /or height (Figure 4-3).

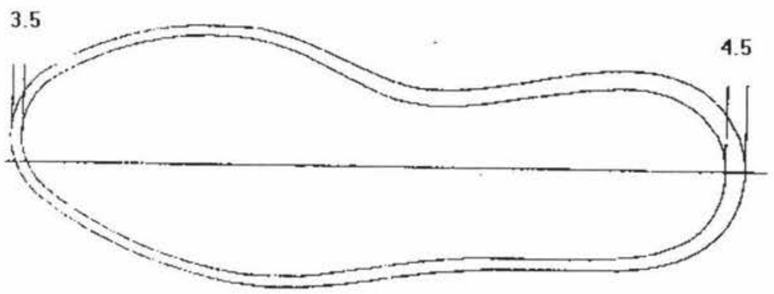


Figure 4-2 Differential dimensional grades in fore and rear zones of a sole

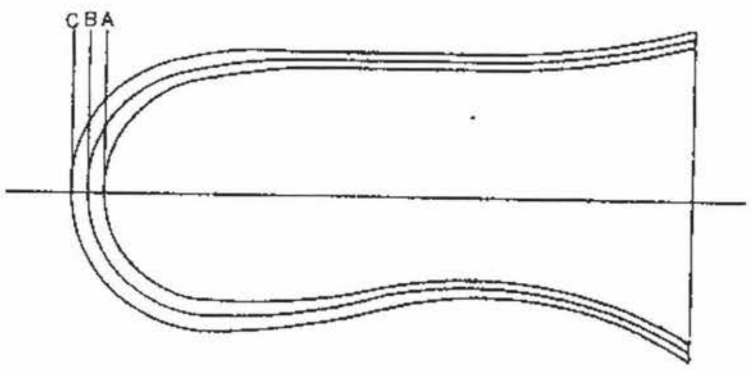


Figure 4-3 Some soles with the same heel

A-small, for size 5 and 6; B- middle, for size 7 and 8; C-large, for size 9 and 10.

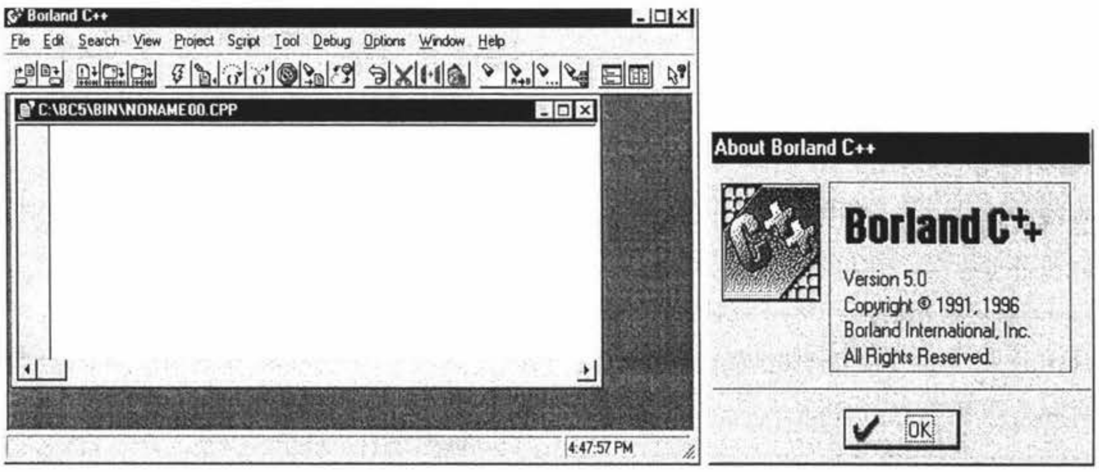
Although this application needs some improvements for practical use, it fully supports that it is very important and necessary to use one scale factor to control the physical size of a model produced by NC program output from the presented software. Therefore, a scale factor was applied into this presented software to control the size of a model to be produced. With this scale factor, any size of the physical model can be get by scaling and machining on the CNC Mill machine. In addition, a physical model can be divided into different zones for separate scanning purpose. Each zone scanning data can be scaled differently, and then all of these different zones and scales scanning data could make various combinations to make different shapes of the scanned model to meet different requirements.

4.4 Software Development

4.4.1 Retrieve Data from Scanned File

Borland C++ 5.0 was chosen as the software development tool to develop the software, because it is one of the most powerful Object-Oriented languages which is widely used today and easily integrated into a lot of mainstream software packages. In addition, Borland has an agreement on free use of Borland C++ 5.0 with Massey University.

Following is a simple example as shown in Figure 4-4 which indicates the interface and software of Borland C++ 5.0.


Borland C++
File Edit Search View Project Script Tool Debug Options Window Help
C:\BC5\BIN\NONAME00.CPP
4:47:57 PM
About Borland C++
OK

Borland C++
Version 5.0
Copyright © 1991, 1996
Borland International, Inc.
All Rights Reserved.

Figure 4-4 The Borland C++ 5.0 User Interface and Software

The main source code can be divided into following two parts:

1) Input scanned data, remove the miscellaneous information and output the useful data in the desired format to an intermediate file.

- Use “ifstream” to input and output the data. Following example shows how to read a scanned file in and how to output the data to an intermediate file.

```
*file1 = scanfile;
ifstream in(*file1, ios::in|ios::binary);
ofstream out ("Test1", ios::out|ios::binary);
```

- The code below is used to get the data character by character from a input data file.

```
str [ 0 ] = '\ 0';                                  1
 while (in) {                                       2
    in.get (ch);                                    3

    if (ch! = '\ n') {                              4
       temp [ 0 ] = ch;                             5
    strcat (str, temp);                             6
    }

    if (ch == '\ n') {                              7
          if (str [ 0 ]== 'S' {                     8
          s = 1;                                    9
           }                                        10

        if ((s == 1) && (str [ 0 ] == 'E')) break;  11

        if(( s == 1) && (str [ 0 ] ! = 'S')) {      12
           out << str << "\ n";                     13
            }
         temp [ 0 ] = '0';                          14
         str [ 0 ] ='\ 0';                          15
                        }
                 }
```

Line 2 and Line 3 is the main function used to read each character of one line from the scanned file into the “ifstream in” stream.

Line 4, Line 5 and Line 6 are used for reading each character of one line. Line 5 takes each character into temporary array “temp [0]”, Line 6 copies one line of characters into the character string “str”. In the end, “str” will hold whole line of characters.

Line 7 ~ Line 15 are the operation when reading the last character of each line. Line 8 and Line 9 set a recognizing sign for the block of coordinate data in order to remove the miscellaneous information at the beginning and at the end of the scanned file.

Line 11 indicates stopping reading when reaching the single capital letter "E". Line 12 and Line13 output the each line of coordinate X, Y and Z value between two "S" into the file "Test1".

After outputting one line of X, Y and Z values into the intermediate file "Test 1", Line 14 and Line 15 clear the array "temp[0]" and string "str[0]" in order to continue to hold next line of characters.

2) Convert the intermediate file "Test1" into the CNC Mill input format file "outputfile".

- Similar to 1), create "ifstream in1" stream for reading intermediate file "Test 1" and create "ofstream out1" stream for outputting into the file2:

```
*file2 = outputfile;
ifstream in1("Test1", ios::in|ios::binary);
ofstream out1 (*file2 , ios::out|ios::binary);
```

- Give the scale rate of model and adjust the model size and using the following code to read file "Test1":

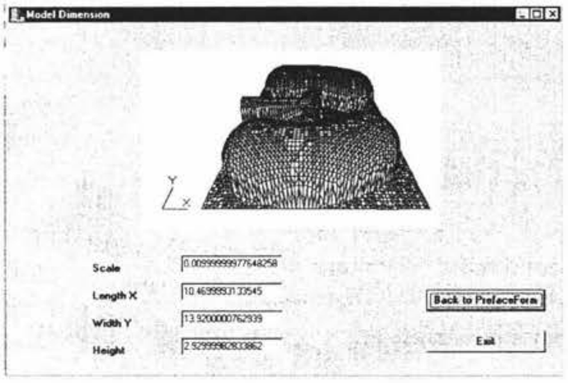
```
cin>>scale;
while (in1){
    in1>>x1>>ch1>>y1>>ch1>>z1;
    x=x1*scale;
    y=y1*scale;
    z=z1*scale;
    . . . .
            }
```

4.4.2 Display the Size of the Model

Defining the model size (length L, width W and height H) uses these codes:

```
if(m==0){
     xmin=xmax=x1;
     ymin=ymax=y1;
    zmin=zmax=z1;
     m=1;
 }
if(m!=0){
     xmin=min(x1, xmin);                        8
     ymin=min(y1, ymin);                        9
     zmin=min(z1, zmin);                        10
     xmax=max(x1, xmax);                        11
```

```
        ymax=max(y1, ymax);                          12
        zmax=max(z1, zmax);                          13
    }
       . . .
    L=(xmax-xmin)*scale;                             15
    W=(ymax-ymin)*scale;                             16
    H=(zmax-zmin)*scale;                             17
```



Scale 0.00999999977648258
Length X 10.4699993133545
Width Y 13.9200000762939
Height 2.52999982833862

Figure 4-5 The Size of the Model to be Cut

From the above code, line 8, 9,10 have the information of the minimum coordiate values of X, Y and Z. Line 10, 11, 12 are for the maximum coordinate values of X, Y and Z. Last three lines (line 15, 16, and 17) perform the calculation to get the length, width and height of the model to be cut on the CNC Mill Machine.

4.4.3 Set Cutter Clearance Plane

The cutter clearance position (Zc) for each cross section (see *3.2.2.1)* is determined by the formula:

Zc = zmax*scale + 5 (see below Line 17 and Line 18 in ***4.4.4***).

For each cross section, zmax is the maximum value of coordinate Z from the starting cutting position until the current cutting position (refer to ***4.4.2***). So, the cutter will locate 5mm above the maximum value of Z times scale value. This will avoid cutting the previous model surface which has been machined.

Figure 4-6 illustrates how to set the cutter clearance plane. Three cross sections are marked 1, 2, and 3. The maximum z values for these three sections are respectively zmax1, zmax2 and zmax3. Supposed zmax2>zmax1>zmax3, the current cutter position is located in section 3 and cutter starting position is in section1. In this case, programme will calculate zmax = zmax2 and set cutter clearance Zc = zmax2*scale + 5, because zmax is the maximum Z value which covers from the starting cutting position to the current section.

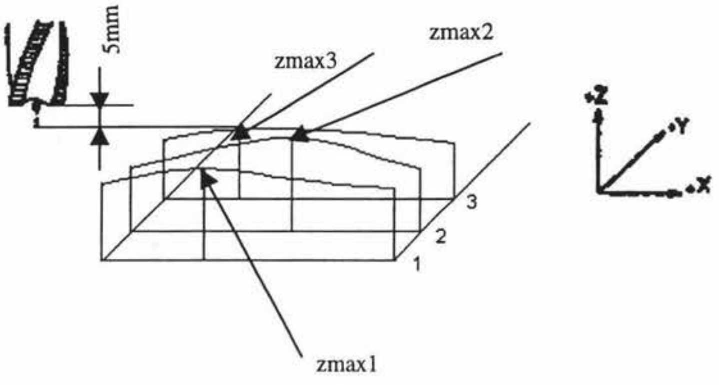


Figure 4-6 Set Cutter Clearance Plane

Of course, when cutter reaches the last cross section, zmax will be the maximum value of coordinate Z for all of the cross sections of the scanned model.

4.4.4 Output CNC Programme

- Convert code between two different sections is shown as below:

```
p1=y;
if(in1){
      if(p1!=p2){ //compare Y value and decide different scan section.
  if((Blocknumber>900)&&(Blocknumber<999)){
    out1<<(Blocknumber++)<<" Z "<<Zc2<<" R0 F9999 M\r\n";
    out1<<(Blocknumber++)<<" STOP M25\r\n";
    k++;
    Blocknumber=0;
    out1<<Blocknumber<<"\r\n";
    out1<<Blocknumber<<" RUN "<<k<<"\r\n";
    Blocknumber++;
    out1<<(Blocknumber)<<" TOOL DEF 1 L0.000 R0.000\r\n";
    Blocknumber++;
    out1<<(Blocknumber)<<" TOOL CALL 1 Z S710.0\r\n";
    Blocknumber++;
    }
    Zc=zmax*scale+5; //return cutter height above the parts.
Zc=zmax*scale+5; //return cutter height above the parts.                                    17
out1<<Blocknumber<<" Z "<<Zc<<" R0 F9999 M03\r\n";                                          18
Blocknumber=Blocknumber +1;
out1<<Blocknumber<<" L X"<<x<<" "<<"Y"<<y<<" "<<"R0 F9999 M\r\n";                           20
Blocknumber=Blocknumber +1;
p2=p1;                                                                                       21
}
```

Line1 and Line 2 compare the y value and determine different sections.

Line 4 ~ Line 14 is for splitting the program into several small programs which are less than 1000 lines. Line 7 indicates the number of programs split.

Line 20 defines the starting cutting position of cutter for each section.

Line 21 save the current y value (p1) into p2 for later comparison.

- Convert code within each cross section is a through output of the X and Z value into "file2":

```
if(p1==p2){
    out1<<Blocknumber<<" L X"<<x<<" "<<"Z"<<z<<" "<<"R0 F900 M"<<"\r\n";
        }
```

Chapter5 User Interface Development

5.1 Introduction to Borland C++ Builder

The software was first developed using C++. It can run smoothly and output the desired NC program. However, as every software engineer understands that a friendly user interface is one of the most important requirements for almost all of the software developed today. This is the very reason that drove to develop a user-friendly interface for the " 3D Scanner to CNC Mill" software.

How to effectively develop a user interface and make use of the existing C++ code was carefully considered. Finally, Borland C++ Builder 5.0 was chosen to be the software developing tool kit.

This allows the developer maximally make use of the former C++ code and quickly develop the user interface. Detailed reasons and the advantages of Borland C++ Builder 5.0 are given below.

1. What is Borland C++ Builder

When Delphi 1.0 was released by Borland back in 1993-94, the programming community was buzzing with glory over the fact that something was coming to destroy Microsoft Visual Basic. Delphi 1.0 had its bugs and quarks, but it was far better than Visual Basic. It could be compiled into a small executable that didn't need an interpreter DLL or special VBXs in order to run. And Delphi was always quicker compiling and running than Visual Basic.

Borland released Delphi 2.0 back in 1995, which had fixed most of the bugs in Delphi 1.0, and added 32-bit code generation, as well as a much better database engine. Specially, the drag and drop capabilities of Delphi have advantages over the resource editor that came with Borland C++.

In January 1997, Borland released C++ Builder with new and novel inventions in programming. Though it came in quietly (Borland made very little mention about C++ Builder at the last BDC), C++ Builder was the C++ programmer's paradise, and rapidly received praise. At last, C++ programmers could drag and drop components into their forms just like Delphi users could. Now, with the release of Borland C++ Builder 5.0, there is far more development options available to the user.

C++ Builder is one of the very first true rapid application development (RAD) tools available for C++, and it's the only RAD tool that provides true component-based drag and drop programming (Telles, 1997). Originally, Windows programming was an error-prone nightmare of MS-DOS based editors, C compilers and linkers, and the Software Development Kit (SDK). Therefore, it is not surprisingly the first Windows programs were bug-ridden and took years to develop. However, today's programs are written in weeks instead of years.

The first evolution of the Windows programming model was the advent of the C++ programming language and C++ class libraries, which encapsulate the hundreds of lines of code needed to do even simple window display and processing into a few lines of C++ code. This code simply wraps up the few hundred previous lines of C code into C++ class. Not only do we no longer have to write them, but also the number of errors in the code wrapped by the C++ classes drops precipitously. After all, once code is written and debugged, it doesn't have to be written again. Code without rewriting isn't likely to develop new bugs.

This second generation of Windows development also brought with it the first Integrated Development Environments (IDEs). These tools allow programmers to edit, compile, and link all within a single application. Integrated debugging followed a bit later and was quickly embraced by the Windows programming community.

The next evolution of Windows development resulted from the development of frameworks. A framework is a skeleton application that holds all of the pieces of your application code together. In retrospect, code framework may have actually been a step backward or at the very least sideways, although many people would disagree with this and remain staunch framework supporters. Then again, there are those who would passionately argue that MS-DOS is still a viable operating system. The reason that frameworks were a step backward is simple. Rather than making programs easier to write and more flexible, they forced us to write programs within a constricting set of rules.

The real problem with framework technology is that it's confining. Although frameworks speed the development of applications by providing many of the basic functions of normal Windows applications, they can quickly get in the way of applications that don't fit the norm.

The solution to this is component-based programming. Components are like building blocks for applications. Components, in fact, are the basis for ActiveX technology, the cornerstone of Internet programming.

C++ Builder does components well, simply and intuitively. Like most great applications, C++ Builder works simply and consistently, thanks in large parts to the component-based nature of the product. Each piece of the system has a specific job and does it as simply and easily as possible. Components are as close to pure C++ programming, as user is ever likely to get and make the whole development effort easier.

2. System Requirements

General system requirement is to run C++ Builder as following:

- Pentium 90 or faster (Pentium 166 recommended).
- Microsoft Windows 95, 98, 2000, or NT 4.0 with Service Pack 3 (or later).
- 32 MB RAM (64 MB recommended).
- Standard edition: 120 MB hard disk for compact install; 185 MB for full install.
- Professional edition: 240 MB hard disk for compact install; 360 MB for full install.
- CD-ROM drive.
- VGA or higher-resolution monitor.
- Mouse or other pointing device.

5.2 The Advantage of Borland C++ Builder 5.0

C++ Builder essentially works along the same lines as Delphi and Visual Basic. It's a Rapid Application Development (RAD) tool that provides fast drag-and-drop development capability to the C++ programming environment (Vokes, 1997). C++ Builder enables to produce 32-bit Windows 95 or higher, or Windows NT program files, like *.exe files that computer can run directly.

There are many advantages of using Borland C++ Builder over other C++ based programming environments. First of all, there is Rapid Application Development technology built into Borland C++ Builder (BCB) that dominates the world. With Visual C++, you have to use MFC or the clunky Resource Editor to build and link forms and components together into an application, while BCB allows users to draw the application onto the screen, then link the different components of the application together using C++.

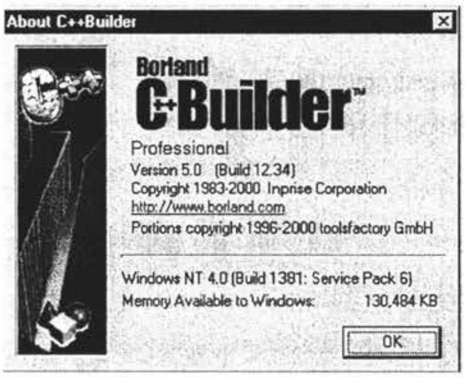
Second, Borland has been fairly religious on making their environment and compiler backward compatible and ANSI compliant. Sure, Borland C++ Builder has added plenty of useful extensions to their code, native ANSI C++ applications, which will compile under any environment (though Visual Component Library, in short, VCL is an extension that is not ANSI C++ compliant), can be still coded by users. Microsoft throws a lot of "extensions" into their code, which makes the code incompatible with other compilers. Borland C++ Builder is fully compatible with Delphi, MFC and ANSI C++.

Furthermore, Borland has the best license agreement in the business (which Microsoft has no plans on toping.) Essentially you can install Borland C++ Builder on any number

of machines, so long as only one machine is running the software at the same time. Try getting away with that from Microsoft.

5.3 User Interface Design

The reason using Borland C++ Builder 5.0 (the software is shown in Figure 5-1) to create user interface can be commented as below. One reason is the main part source code was written by Borland C ++ 5.0 software. The second reason is that Borland C++ Builder 5 fully supports the whole C++ and ANSI C++ language. The third reason lies in that Borland C++ Builder 5.0 is a Rapid Application Tool to provide component-based drag and drop programming method for the C++ Windows programming world.


About C++Builder

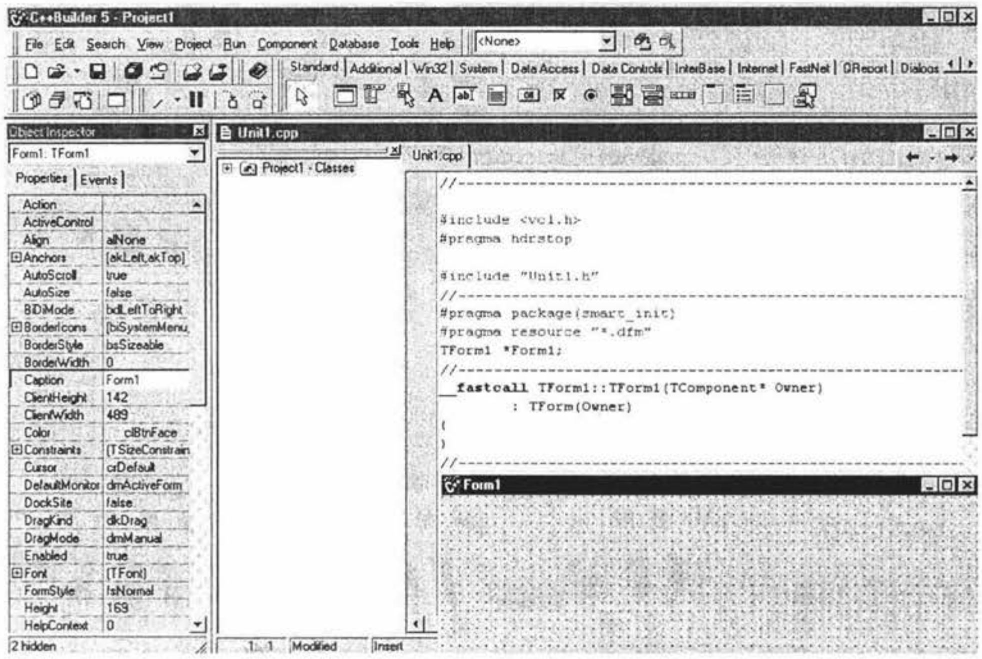
Borland C++Builder™
Professional
Version 5.0 (Build 12.34)
Copyright 1983-2000 Inprise Corporation
http://www.borland.com
Portions copyright 1996-2000 toolsfactory GmbH

Windows NT 4.0 (Build 1381: Service Pack 6)
Memory Available to Windows: 130,484 KB

Figure 5-1 The Borland C++ Builder 5.0 Software

Figure 5-2 is the main user interface of Borland C++ Builder 5.0. The Unit form, which is generally behind the main form, indicates the main source code relative to the main form.

The user interface of this software mainly includes Preface Form, Process Form, Graphic Form and Model Dimension Form. Each form consists of different buttons and dialogue boxes.


C++Builder 5 - Project1
File Edit Search View Project Run Component Database Tools Help
<None>
Standard Additional Win32 System Data Access Data Controls InterBase Internet FastNet QReport Dialogs
Object Inspector
Form1: TForm1
Properties Events
Unit1.cpp
Project1 - Classes
Form1
2 hidden
1: 1 Modified Insert

Action	
ActiveControl	
Align	alNone
⊞Anchors	[akLeft,akTop]
AutoScroll	true
AutoSize	false
BiDiMode	bdLeftToRight
⊞BorderIcons	[biSystemMenu,
BorderStyle	bsSizeable
BorderWidth	0
Caption	Form1
ClientHeight	142
ClientWidth	489
Color	clBtnFace
⊞Constraints	(TSizeConstrain
Cursor	crDefault
DefaultMonitor	dmActiveForm
DockSite	false
DragKind	dkDrag
DragMode	dmManual
Enabled	true
⊞Font	(TFont)
FormStyle	fsNormal
Height	169
HelpContext	0

```
//---------------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop

#include "Unit1.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------
```

Figure 5-2 The Borland C++ Builder 5.0 Interface

5.3.1 The Preface Form

Figure 5-3 is the Preface Form. It gives the name of this software and the location where this software to be used. More important, there are five buttons at the bottom of the right hand side of the form. Through these different buttons, the programme will present the Introduction Form, About Form, Help Form and Process Form. Each of these forms will show the user corresponding information.

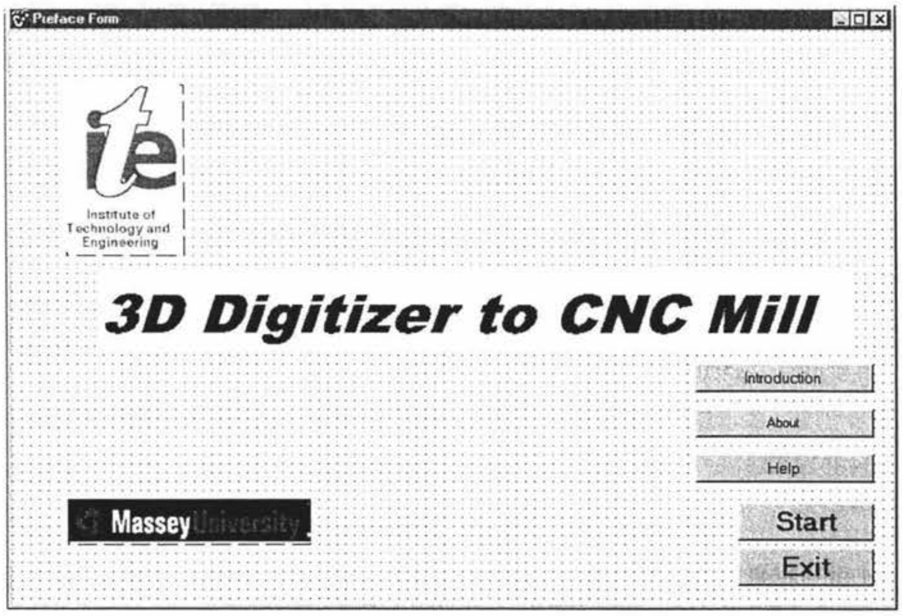


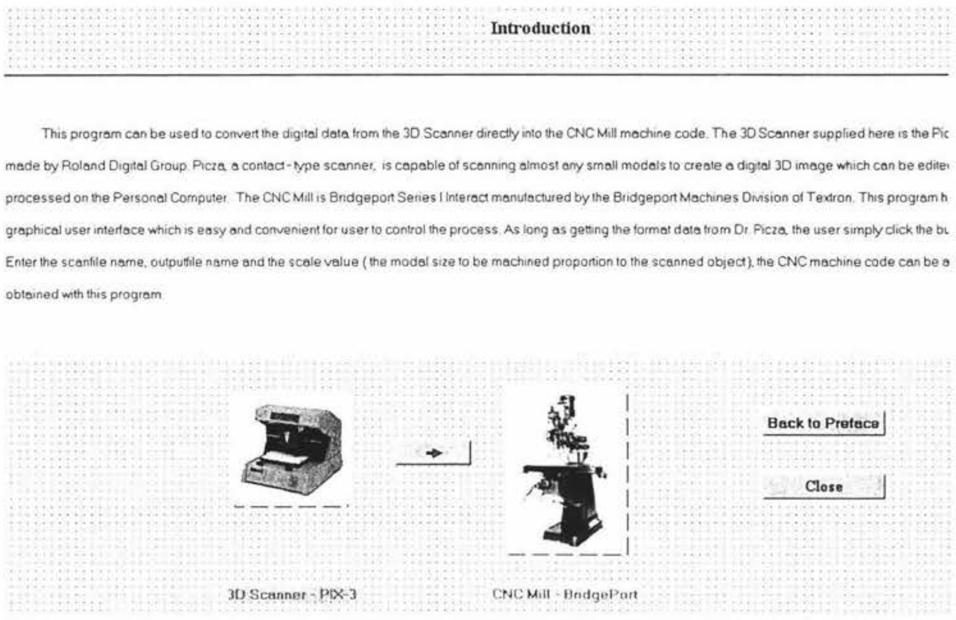
Figure 5-3 The Preface Form

The main unit code of the preface form is as below:

```
void __fastcall TPrefaceForm::Button1Click(TObject *Sender)
{

 ProcessForm->ShowModal( );
}
//---------------------------------------------------------------------------
void __fastcall TPrefaceForm::Button5Click(TObject *Sender)
{
 PrefaceForm->Close();
}
//---------------------------------------------------------------------------
void __fastcall TPrefaceForm::Button3Click(TObject *Sender)
{
 AboutBox->ShowModal( );
}
//---------------------------------------------------------------------------
void __fastcall TPrefaceForm::Button2Click(TObject *Sender)
{
  IntroductionForm->ShowModal( );
}
//---------------------------------------------------------------------------
void __fastcall TPrefaceForm::Button4Click(TObject *Sender)
{
 Help->ShowModal( );
}
//---------------------------------------------------------------------------
```

On the Preface Form, the "Introduction" button is designed to invoke the Introduction Form. This form gives the brief background description of the presented software as shown in Figure 5-4. There are two buttons on the Introduction Form. One is for backing to Preface Form; the other is for closing Introduction Form.


Introduction

This program can be used to convert the digital data from the 3D Scanner directly into the CNC Mill machine code. The 3D Scanner supplied here is the Pic made by Roland Digital Group. Picza, a contact-type scanner, is capable of scanning almost any small models to create a digital 3D image which can be edite processed on the Personal Computer. The CNC Mill is Bridgeport Series I Interact manufactured by the Bridgeport Machines Division of Textron. This program h graphical user interface which is easy and convenient for user to control the process. As long as getting the format data from Dr. Picza, the user simply click the bu Enter the scanfile name, outputfile name and the scale value (the model size to be machined proportion to the scanned object), the CNC machine code can be a obtained with this program.

Figure 5-4 the Introduction Form

The main code of the introduction form is:

```
//---------------------------------------------------------------------------
void __fastcall TIntroductionForm::Button1Click(TObject *Sender)
{
 IntroductionForm->Close( );
}
//---------------------------------------------------------------------------
void __fastcall TIntroductionForm::Button2Click(TObject *Sender)
{
  IntroductionForm->Close( );
  PrefaceForm->Show( );
}
//---------------------------------------------------------------------------
```

The "About" button on the Preface Form does the same work like the "About" menu item in any other software, which gives information about the version of the software, the name of the developer and the copyright. The "About" Form of the presented software is shown in Figure 5-5.

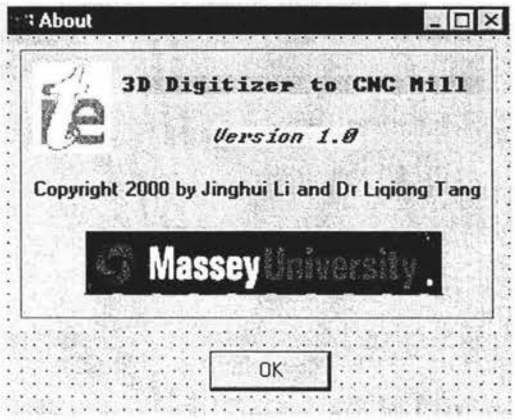


Figure 5-5 the About Form

The main unit code for the About Form is as below:

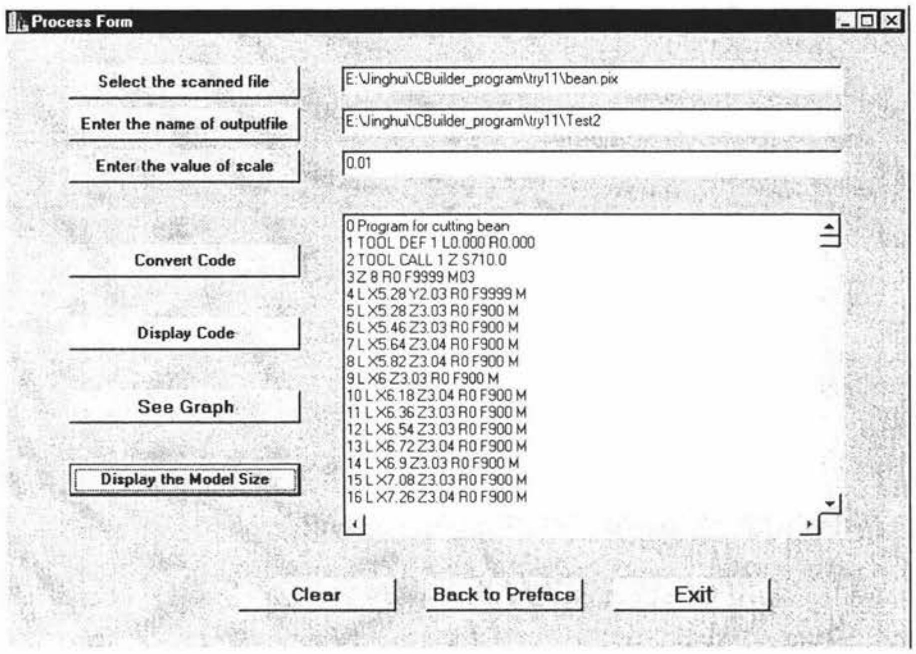
```
//---------------------------------------------------------------------
__fastcall TAboutBox::TAboutBox(TComponent* AOwner)
        : TForm(AOwner)
{
}
//---------------------------------------------------------------------
```

Designing a "Help" item in the menu or providing a "Help Button" in any software has become an important part. Information provided in the Help nowadays almost covers all the information needed by the users. It almost become a must for software developer to present the instructions on how to use the help, explain each particular function and library and give step by step tutorials. The software provided also followed the same trend and provide all the possible information in the "Help Form" shown in Figure 5-10. The form is invoked by clicking the "Help" button on the Preface Form.

The " Start" button on the Preface Form is a key button. It will bring up the Process From which will be discussed in the next section.
The function of the "Exit" button is certainly like any other exit button, which will make the user exit from the Preface Form and this is the correct way to terminate the execution of the presented software.

5.3.2 Process Form

The Process Form is the most important and central part of this software. Nearly all of the functions for converting scanned data into the NC program with the right scale and correct format were designed and presented on this form. Figure 5-6 displays the arrangement of the functions and the corresponding buttons.



```
E:\Jinghui\CBuilder_program\try11\bean.pix
E:\Jinghui\CBuilder_program\try11\Test2
0.01
```

```
0 Program for cutting bean
1 TOOL DEF 1 L0.000 R0.000
2 TOOL CALL 1 Z S710.0
3 Z 8 R0 F9999 M03
4 L X5.28 Y2.03 R0 F9999 M
5 L X5.28 Z3.03 R0 F900 M
6 L X5.46 Z3.03 R0 F900 M
7 L X5.64 Z3.04 R0 F900 M
8 L X5.82 Z3.04 R0 F900 M
9 L X6 Z3.03 R0 F900 M
10 L X6.18 Z3.04 R0 F900 M
11 L X6.36 Z3.03 R0 F900 M
12 L X6.54 Z3.03 R0 F900 M
13 L X6.72 Z3.04 R0 F900 M
14 L X6.9 Z3.03 R0 F900 M
15 L X7.08 Z3.03 R0 F900 M
16 L X7.26 Z3.04 R0 F900 M
```

Figure 5-6 the Process Form

Before converting any data, the first step is input the data into the system. The button named "Select the scanned file" brings up a dialogue box, which allows the user to specify the path and the name of the file which stalled the scanned data. Once the "Open Button" on the dialogue box is clicked, the dialogue box is automatically closed. The path and the name of the scanned file will be shown in the Edit box next to the button of "Seclect the scanned file". The programme then loads the scanned data into the system and converts it into an intermediate file. The intermediate file will be further processed to output as a NC program. This will detailed discussed in section 4.4.1.

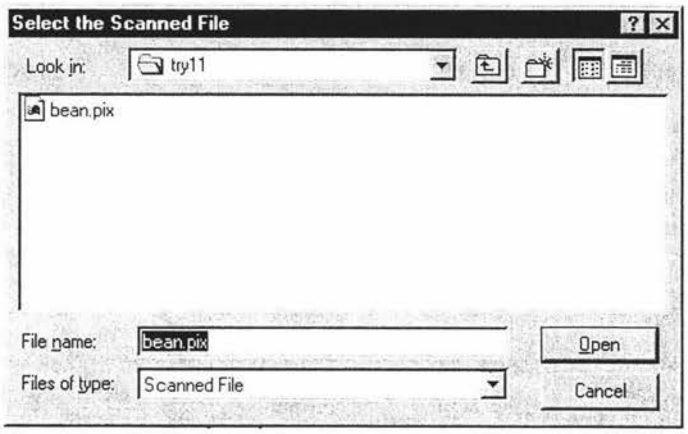


Figure 5-7 the Select the Scanned File Dialogue Box

The unit code for "select the scanned file button" consists of the following programming code:

```
if(OpenDialog1->Execute ( )){
 Edit1->Text = OpenDialog1->FileName;
 strcpy(scanfile, Edit1->Text.c_str( ));
 }
 else Edit1->Text = " ";
 *file1=scanfile;
 ifstream in (*file1, ios::in|ios::binary);
 ofstream out ("Test1",ios::out|ios::binary);
```

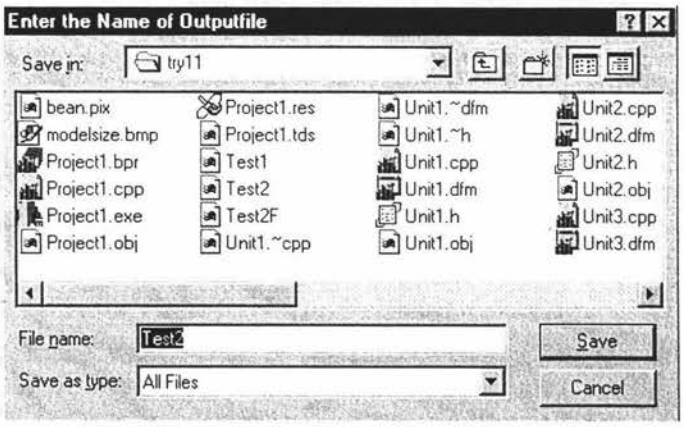


Figure 5-8 the Enter the Name of Output File Dialogue Box

The unit code corresponding to "Enter the Name of Output File" button is:

```
void __fastcall TProcessForm::Button7Click(TObject *Sender)
{
 if(SaveDialog1->Execute()){
  Edit2->Text = SaveDialog1->FileName;
```

```
        }
    }
```

The size of a physical model produced by the NC program output from the presented software is controlled by a scale factor. The value of the scale factor can be input through the button called "Enter the value of scale". This button will bring up the Input dialogue box as shown in Figure 5-9. Once the OK button is clicked, the input value of the scale will be shown in the third Edit box on the right hand side of the Process Form.

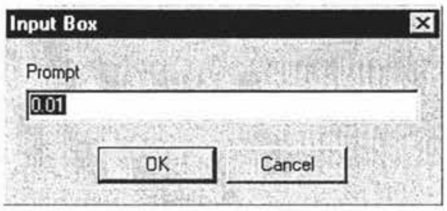


Figure 5-9 the Scale Value Input Box

The unit code for "Enter the Scale Value" button is:

```
void __fastcall TProcessForm::Button6Click(TObject *Sender)
{
  AnsiString InputString = InputBox("Input Box", "Prompt", "0.01");
  Edit3->Text = InputString;
}
```

After specify the name of the scanned data file, the name of the output NC program and the value of the scale, the next step is to generate the NC program from the scanned data. This task is carried out by the converting process which is executed through the button named "Convert code". When this button is pressed, the intermediate file generated from the scanned data and discussed in the previous section is read and converted in the NC program with the right scale and format.

The final NC program can then be viewed by clicking the "Display code" button on the Process Form. The actual NC code will then be displayed in the "Rich Edit Box" on the right hand side of the Process Form. A horizontal and vertical bar is designed for the "Rich Edit Box" for users to view and check the whole NC program before sending to the CNC machines.

The NC program generated by this software is a text format file which can be opened and edited by using Microsoft Word, Notepad, Wordpad or any other text editor.

5.4 Produce User Instruction

5.4.1 System Configuration and Software Execution

The presented software (3D Digitizer to CNC Mill) is developed using Borland C++ Builder 5.0. The primary task of this software is to bridge the 3D scanner and the CNC mill in ITE Engineering Lab.

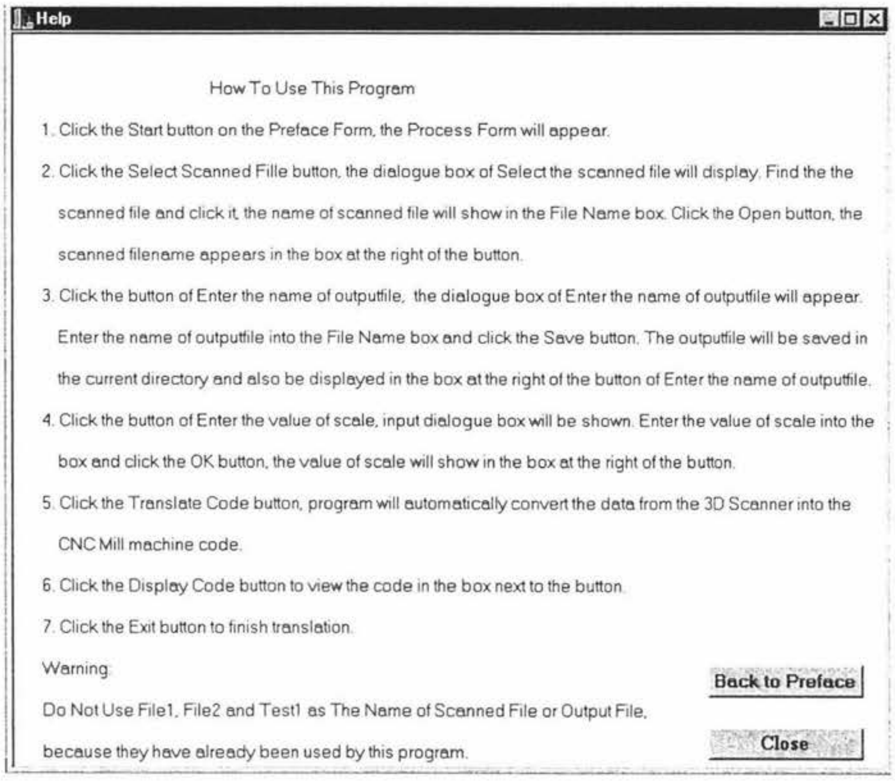
The output file of the 3D scanner is the input file of the presented software. In order to effectively run the software and to directly send the NC program to the CNC mill, an integrated system as shown in Figure and the corresponding configuration is recommended below. Once loading the software in a window's environment, by executing the corresponding file, the 3D Digitizer to CNC Mill software will start to run.

Recommended Configuration:

- A 3D Scanner PICZA and software Dr. PICZA,
- A CNC Mill (Bridgeport Series I Interact) and Heidenhain TNC145 Controller,
- Intel Pentium 486 –based PC or higher,
- 32 MB RAM (64 MB RAM recommended),
- Minimum 120 MB hard disk space,
- Windows 95 higher or Windows NT 4.0 with service pack3 or later,
- Borland C++ 5.0 and Borland C++ Builder 5.0,
- VGA or higher-resolution monitor,
- Mouse or other pointing device.

5.4.2 Instructions for Converting Scanned data to NC Program

Although the presented software is very user friendly, like any other software on the market, this software also provides a detail step by step instruction for converting scanned data to NC program. The instructions are provided in the "Help Form" which can be accessed by the "Help" button on the "Preface Form". The procedure to convert a scanned data which is a file in PIX format to NC program which is a text format is clearly explained in the "Help Form". It is a very straight procedure. Anyone follows the steps should be able to convert a PIX file to a NC program. Figure 5-10 gives an example of the instructions in the Help Form.



How To Use This Program

1. Click the Start button on the Preface Form, the Process Form will appear.
2. Click the Select Scanned Fille button, the dialogue box of Select the scanned file will display. Find the the scanned file and click it, the name of scanned file will show in the File Name box. Click the Open button, the scanned filename appears in the box at the right of the button.
3. Click the button of Enter the name of outputfile, the dialogue box of Enter the name of outputfile will appear. Enter the name of outputfile into the File Name box and click the Save button. The outputfile will be saved in the current directory and also be displayed in the box at the right of the button of Enter the name of outputfile.
4. Click the button of Enter the value of scale, input dialogue box will be shown. Enter the value of scale into the box and click the OK button, the value of scale will show in the box at the right of the button.
5. Click the Translate Code button, program will automatically convert the data from the 3D Scanner into the CNC Mill machine code.
6. Click the Display Code button to view the code in the box next to the button.
7. Click the Exit button to finish translation.

Warning:
Do Not Use File1, File2 and Test1 as The Name of Scanned File or Output File, because they have already been used by this program.

Figure 5-10 the Help Form

The main unit code of this is:

```
//---------------------------------------------------------------------------
void __fastcall THelp::HelpFormClick(TObject *Sender)
{
 Help->Close( );
}
//---------------------------------------------------------------------------

void __fastcall THelp::Button1Click(TObject *Sender)
{
 Help->Close( );
 PrefaceForm->Show( );
}
//---------------------------------------------------------------------------
```

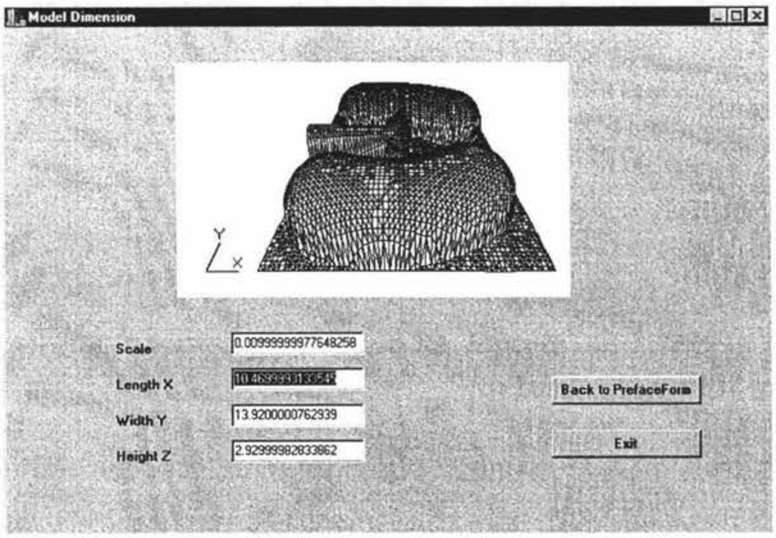
5.4.3 Machining Parameters

As this is the first version of the presented software and the main focus is to convert scanned data to NC program with different scales, the machining parameters such as tool number, cutting speed, feed rate are temporally fixed. This means, at present, users can not change the cutting speed, a feed rate, etc through the NC program. Currently, all these values are altered on the CNC controller by the operator. However, in the later version of this software, users will be able to specify the values of these machining parameters.

The machining parameters defined in current source code are:
Tool number = 1;
Spindle speed = 710 RPM;
Feed rate = 99999;

5.4.4 View the Model

In the preface form, clicking the "Display the Model Size" button, the Model Dimension Form will present to the user (Figure 5-11). The graphic area shows the wireframe of the model to be cut. It also shows the scale, length, width and height of the model.



Scale	0.00999999977648258
Length X	10.4699993133545
Width Y	13.9200000762939
Height Z	2.92999982833862

Figure 5-11 the Model Dimension Form

The unit code for "Display the graphic form" is:

```
void __fastcall TGraphicForm::FormPaint(TObject *Sender)
{
  char ch1, s1[10];
  float x, y, z, scale, p1, p2;
  ifstream in1("Test1", ios::in|ios::binary);
  if(!in1){
    ShowMessage( "Cannot transfer the scanfile.");
    }
  strcpy(s1, ProcessForm->Edit3->Text.c_str( ));
  scale = atof (s1);//convert str to float.
  p2=0;
  while(in1){
    in1>>x>>ch1>>y>>ch1>>z;
    x=x*scale;
    y=y*scale;
    z=z*scale;
    p1=y;

    if(in1){
           if(p1!=p2){
      GraphicForm->Canvas->MoveTo(x,z);
      p2=p1;
      }

     if(p1==p2){
      GraphicForm->Canvas->LineTo(x,z);
      }
     }
    }
  in1.close( );
}
```

The main unit code for displaying the model dimension is as following:

```
  L=(xmax-xmin)*scale;
  W=(ymax-ymin)*scale;
  H=(zmax-zmin)*scale;
  ModelDimensionForm->Edit1->Text=L;
  ModelDimensionForm->Edit2->Text=W;
  ModelDimensionForm->Edit3->Text=H;
  ModelDimensionForm->Edit4->Text=scale;
//---------------------------------------------------------------------------
void __fastcall TModelDimensionForm::Button1Click(TObject *Sender)
{
  ModelDimensionForm->Close( );
  ProcessForm->Close( );
  PrefaceForm->Show( );
}
//---------------------------------------------------------------------------
void __fastcall TModelDimensionForm::Button2Click(TObject *Sender)
{
 ModelDimensionForm->Close( );
}
//---------------------------------------------------------------------------
void __fastcall TModelDimensionForm::Button3Click(TObject *Sender)
{
 Edit1->Text=" ";
 Edit2->Text=" ";
 Edit3->Text=" ";
 Edit4->Text=" ";
}
//---------------------------------------------------------------------------
```

Chapter6 Software Test

6.1 Data Covert

The core of this software is to bridge the 3D scanner and the CNC milling machine in ITE's workshop. A detail study on the format of the scanned data and the NC format for the CNC machine was carried out at the early stage of this project. Although software development stage did try to cover all the possible situations and unusualness in the scanned data and the output format, to develop a bug free software, software testing is a must.

The test of the presented software was carried out using the facilities in ITE's workshop. The software was installed on the same PC which also installed 3D scanning software. This is purely to provide convenience for quickly up-load scanned data into the presented software.

Several scans were carried out and then performed the data conversion to generate the corresponding NC programs with different scales. Each of the process went smoothly and output the NC programs with correct format. The software also correctly constructed the scanned model on the screen using the scanned data. Some modifications were made after the test only to make the software more efficient. All the tests demonstrated the functions developed in this software performed to the expected level.

6.2 NC Program Testing

NC program testing is the final stage of the test designed for the presented software. The output of this software is expected to drive CNC machines. This test again was based on the facilities in ITE's workshop. The CNC milling machine has a Heidenhain TNC 145 Controller. The communication between the CNC mill and the PC was realised through a RS 232 port.

A pair of plastic bean was chosen as the sample model (Figure 6-1). And it is intended to produce two pairs of beans with two different scales through the same scanned data but using two scale factors to generate two NC programs. A block of wax was used as the test material and was clamped on the machine worktable through a vice. Both these two NC programs successfully produced the pair of the beans with the correct size as shown in Figure 6-2. The experiments fully approve the accuracy and effectiveness of the NC code generated by the presented software.

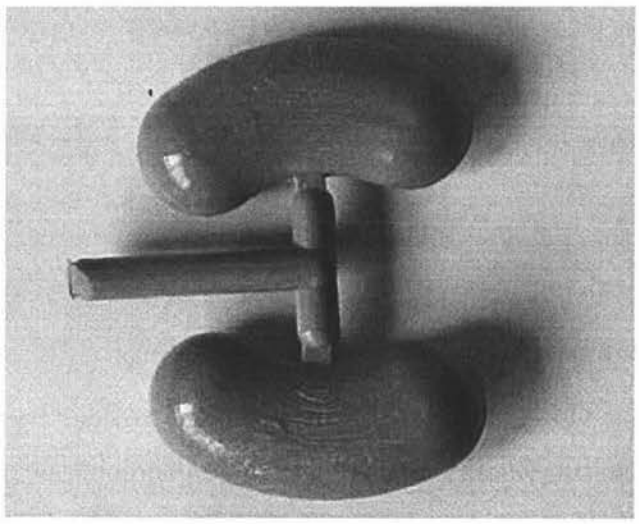


Figure 6-1 A Pair of Plastic Bean to be scanned

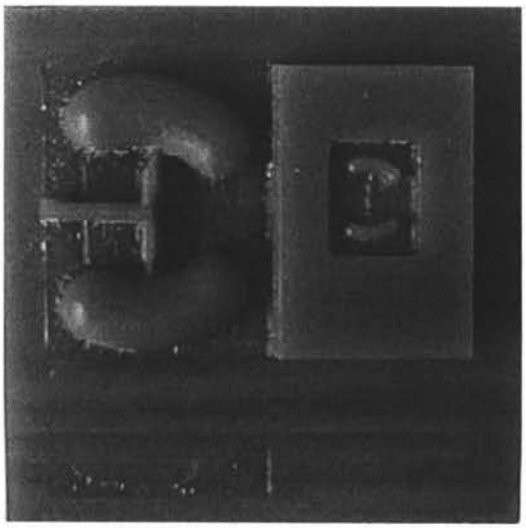


Figure 6-2 the Wax Model Processed by CNC Mill Machine

Chapter7 Results and Conclusions

The software (3D digitizer to CNC Mill) has been run on different PCs at ITE and tested under different assumptions. The software can now be used with the existing CAD/CAM/CAE system and the CNC facilities on the shop floor. It indeed established an integrated system as illustrated below (Figure 7-1). Such an integrated system opens a convenient and effective working environment for studying, teaching and research.

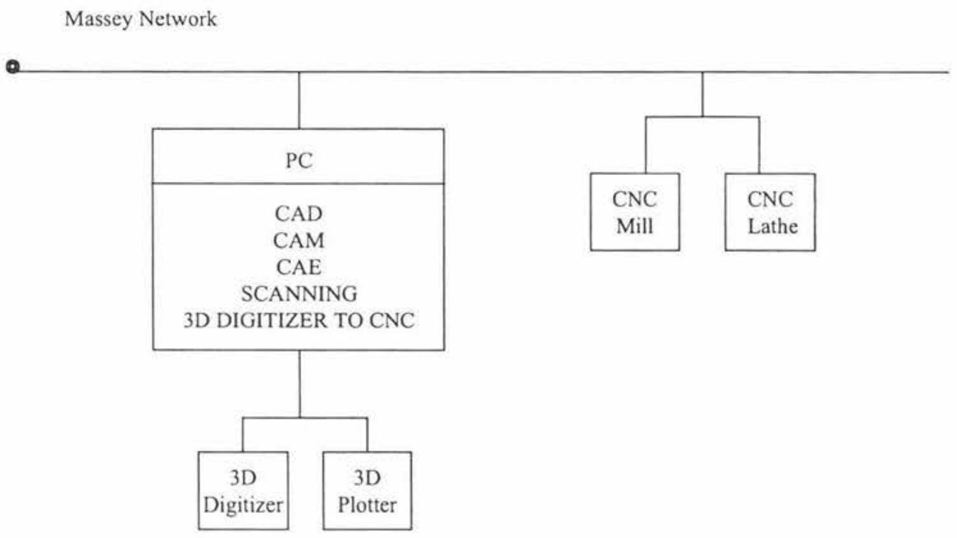


Figure 7-1 The Presented Integrated System

At ITE, once a computer model is created using CAD, a 3D physical model can be produced by the 3D plotter within hours. Any dummy model brought to the workshop. A duplicate can be made through the 3D scanner, the presented software and the CNC machines. Integration maximises the use of the facilities and fully demonstrates the advantages of a computer-based design and manufacturing system.

It is clear that the software developed brought the CAD/CAM/CAE, CNC, 3D Scanner and the 3D digitizer together. The software 3D digitizing to CNC mill also provides a very effective way to convert 3D scanned data to NC program. The software can automatically load correct number of lines from a large NC program to the CNC controller without splitting the NC program into several sections.

The software gives students hands-on opportunities in the study of computer-integrated manufacturing and rapid prototyping. It also helps students to understand data exchange from one system to another. It is no doubt that the presented system is one of the ideal low cost systems for teaching and research.

Reference

1. Alciatore D. G. and Terry T. W., 'Importing and reshaping digitized data for use in rapid prototyping: a system for sculpting polygonal mesh surfaces', Rapid Prototyping Journal, Vol. 2, No. 1, 1996, pp13-23.
2. Bass L. and J. Coutaz, 1991, Developing Software for the User Interface, U. S. A., Addison-Wesley Publishing Company, Inc.
3. Bell R. C. and L. Tang, 'Rapid Prototyping & Manufacturing', Projects, Vol. 8, 1999.
4. Bell R. C., 1999, Rapid Prototyping and Manufacturing, Fourth Year Project, Institute of Technology and Engineering, Massey University.
5. Bøhn J. H., 'Integrating rapid prototyping into the engineering curriculum-a case study', Rapid Prototyping Journal, Vol. 3, No. 1, 1997, pp32-37.
6. Chikofsky E. J., 'Reverse Engineering and Design Recovery: A Taxonomy', IEEE Software, Vol. 7, Jan. 1990, pp13-17.
7. Christensen J. and Amit B., 2000,'Reverse engineering of clear solids using refractive index matching', Rapid Prototyping Journal, Vol. 6, No. 2, 2000, pp87-96.
8. Chua C. K., et al, ' Rapid Prototyping Versus Virtual Prototyping in Product Design And Manufacturing', 'International Journal of Advanced Manufacturing Technology', Vol. 15, No. 8, 1999, pp597 – 603.
9. Chui W. H., and P. K. Wright, 'A WWW Computer Integrated Manufacturing Environment for Rapid Prototyping and Education', International Journal of Computer Integrated Manufacturing, Vol. 12, No. 1, 1999, pp54 – 60.
10. Connell J. L. and Linda B. S., 1989, Structured Rapid Prototyping: An Evolutionary Approach to Software Development, New Jersey, Prentice Hall, Inc.
11. Engelke W. D., 1987, How to Integrate CAD/CAM Systems: Management and Technology, New York, Marcel Dekker, Inc.
12. Fadel G. M. and Chuck K., 'Accuracy issues in CAD to RP translations', Rapid Prototyping Journal, Vol. 2, No. 2, 1996, pp4-17.
13. Gan W. S., et al, 'Rapid Prototyping System for Teaching Real-Time Digital Signal Processing', IEEE Transactions on Education, Vol. 43, No. 1, Feb. 2000, pp19-24.

14. Gordon V. S. and J. M. Bieman, 'Rapid Prototyping: Lessons Learned', IEEE Software, Jan. 1995, pp85-90.
15. Hamblen J. O., 'Rapid Prototyping Using Field-Programmable Logic Devices', IEEE Micro, May-June 2000, pp29-37.
16. Hoxie S., et al, 'Developments in Standards for Networked Virtual Reality', IEEE Computer Graphics and Applications, March/April 1998, pp6 – 9.
17. Jamieson R. and Herbert H., 'Direct slicing of CAD models for rapid prototyping', Rapid Prototyping Journal, Vol. 1, No. 2, 1995, pp4-12.
18. Jurrens K. K., 'Standards for the Rapid Prototyping Industry', Rapid Prototyping Journal, Vol. 5, No. 4, 1999, pp169-178.
19. Karapatis N. P., et al, 'Direct rapid tooling: a review of current research', Rapid Prototyping Journal, Vol. 4, No. 2, 1998, pp77-89.
20. Kent R. and Henderson K., 1997, Teach Yourself Borland C++ Builder in 21 Days, Indianapolis, Ind: Sams Publishing.
21. Kochan A., 'Rapid prototyping trends', Rapid Prototyping Journal, Vol. 3, No. 4, 1997, pp150-152.
22. Lanzagorta M., et al, 'Rapid Prototyping of Virtual Environments', Computing in Science & Engineering, May/June 2000, pp68-73.
23. Leach R. J, 2000, Introduction to Software Engineering, U. S. A, CRC Press LLC.
24. Ng P., et al, 'Motion compensation for five-axis rapid prototyping system', Rapid Prototyping Journal, Vol. 4, No. 2, 1998, pp68-76.
25. Park J., et al, 'Characterization of the Laminated Object Manufacturing Process', Rapid Prototyping Journal, Vol. 6, No. 1, 2000, pp36-49.
26. Petrov M., et al, 'Optical 3D Digitizers: Bringing Life to the Virtual World', IEEE Computer Graphics and Applications, May/Jun 1998, pp28 – 41.
27. Radstok E., 'Rapid Tooling', Rapid Prototyping Journal, Vol. 5, No. 4, 1999, pp164-168.
28. R. T. Lee, and W. S. Cheng, 2002, 'A Multizone Scaling Method for CAD in Shoe Sole Design', The Int J Adv Manuf Technol (2002) 19:313-317.
29. Schildt H., 1998, C++ from the ground up, 2nd edition, California, USA, Osborne/McGraw-Hill.
30. Smith D. A., 1982, Rapid Software Prototyping: A Dissertation Submitted in Partial Satisfaction of the Requirements for the Degree Doctor of Philosophy in

Information and Computer Science (University Microfilms), University of California, Ivine.

31. Smith G. T., 1993, CNC Machining Technology Vol. 3: Part Programming Techniques, London, Springer-Verlag London Limited.
32. Telles M., 1997, Borland C++ Builder, U.S.A, Coriolis Group Books.
33. Vokes J., 1997, Borland C++ Builder for Dummies, Foster City, Canada: IDG Books Worldwide.
34. Wohlers T., 'Future potential of rapid prototyping and manufacturing around the world', Rapid Prototyping Journal, Vol. 1, No. 1, 1995, pp4-10.
35. Yan Y., et al, 'Study on Multifunctional Rapid Prototyping Manufacturing System', Integrated Manufacturing Systems, 9/4, 1998, pp236 – 241.
36. Belludi N. and D.K. Thakral, 1998, 'Rapid prototyping…the future is wow!' www.members,tripod.com/ybnaga/rp.htm, 24/11/2000.
37. Wicks and Wilson Ltd. homepage, www.wwl.co.uk/triform.htm#Introduction, 26/10/2000.
38. 3D Scanner, www.spline.nl/machines/picza.html, 26/10/2000.
39. Applied Research Associates NZ Ltd., www.aranz.co.nz/rbf/index.html, 14/11/2000.
40. Crokett R., 'Rapid Prototyping: A Subtle Industrial Revolution', www.ienonline.com/ien/CuttingEdge/cutapr1.html, 13/11/2000.
41. CyberF/X, Inc. homepage, www.cyberfx3d.com, 31/10/2000.
42. Dolenc A., 1994, 'An Overview of Rapid Prototyping Technologies in Manufacturing,' www.cs.hut.fi/~ado/rp/rp.html, 02/11/2000.
43. Efunda Process Home page, 'Rapid Prototyping: An Overview,' www.efunda.com/processes/rapid_prototyping/intro.cfm, 02/11/2000.
44. Generating Reverse Engineering Tools, http://cds.unina.it/~andeluci/FPL.html, 31/10/2000.
45. Grimm T., May 1999, 'Rapid Prototyping as a competitive Weapon', www.acceltechinc.com/nojava/competitive.html, 31/03/2000.
46. Ideas in 3D Hompage, 'Introduction to 3D Digitizing Systems,' www.ideasin3d.com/introdig.html.
47. International Engraveing Devices Inc, www.gravers.com/iedpicza.htm, 14/09/2000.
48. Lifecycle Process Model "V-Model" Part 3 "Collection of Manuals" RE – Reverse Engineering, www.informatik.uni-bremen.de/uniform/gdpa_d/part3/p3re.htm, 31/10/2000.

49. McCreight G., 'CNC: Computer Numerical Control', www.awi-net.org/cnc.html, 13/11/00.
50. MDX-15 Features, www.ahearn.com/MDX_15.htm, 13/11/2000.
51. Minolta Corporation, ISD – 3D Scanner, www.minolta3d.com, 26/10/2000.
52. Modela, Type MDX-3, Ultralight CNC milling machine, www.spline.nl/machines/modela.html, 26/10/2000.
53. MODELA MDX-3 3D Plotter, www.rolanddga.com/products/3d/modelers/modela.asp, 26/10/2000.
54. New report finds rapid prototyping market in slump, www.wohlersassociates.com/99state.htm, 14/11/2000.
55. Novitski B. J., 1999, 'Rapid prototyping, a process borrowed from industrial design, lets architects make scale models from computer files', www.architecturalrecord.com/DIGITAL/DA_ARTIC/DA12_99.ASP, 13/11/2000.
56. Oglivie L., 1997, 'Final report virtual prototyping,' http://real.uwaterloo.ca/~lars/second_final.htm#Virtual, 24/11/2000.
57. PC ToolBox, www.goldmachinery.com/pctool.htm, 14/11/2000.
58. Percival R. L., 1999, 'Standing on the shoulders of Giants: The Reverse Engineering of Computer Software and the Law of Copyright in Canada,' Smith Lyons publications, www.smithlyons.ca/Publications/Articles/IT_99_10_2.htm#R_E, 02/11/2000.
59. PICZA 3D Scanner, www.rolanddga.com/products/3d/scanners/picza.asp, 26/10/2000.
60. Rapid Prototyping-Advantage Prototype Systems- SLA, RTV, www.advproto.com, 26/10/2000.
61. Rapid Reverse Engineering, www.geomagic.com/advantage/adcinc/implications.php3, 31/10/2000.
62. Research Paper Abstracts, www.me.uvic.ca/~cbr/abstracts.html#ReverseEngineering, 13/11/2000.
63. Reverse Engineering Reading List for Beginners, www.cc.gatech.edu/reverse/introREBib.html, 31/10/2000.
64. RPDI Purpose and Capabilities, www.udri.udayton.edu/rpdl/purpose.htm.
65. Series I Standard, www.bpt.com/catalog/milling/ser1.htm, 14/09/2000.
66. Virtual Prototyping, www.cs.hut.fi/Opinnot/Tik-86.141/gs1999/haahtela_tero_vp.html, 24/11/2000.

67. What is Rapid Prototyping, www.rapidpro.com/files/whatis.htm, 31/03/2000.
68. Wohlers Associates, www.wohlersassociates.com, 13/11/2000.
69. Wohler T., 'The Challenge of 3D Digitizing,' www.ideasin3d.com/nurbs2.html, 13/11/2000.
70. Wohler T., 'The 3D Digitizers for Engineering,' www.ideasin3d.com/nurbs3.html, 13/11/2000.
71. Wohlers T. T., 1993, 'Rapid Prototyping Systems', Proceedings of the First European Rapid Prototyping Convention, Paris, France, www.wohlersassociates.com/overview.html, 13/11/2000.
72. Zhang Y. et al, 'An Internet based STEP data exchange system for virtual enterprises', www.eng.fsu.edu/...ng/html/abstract15.html, 15/11/2000