UNIVERSITY OF ADELAIDE

DEPARTMENT OF MECHANICAL ENGINEERING

Generic Methodology for Developing a Non Standard-specific Rationalised Information Flow and Storage System for Computer-based Manufacturing

By

Henry C.W. Lau

A Thesis submitted as the Requirement for the Degree of Doctor of Philosophy

August, 1996

ABSTRACT

For a manufacturing company to become world competitive, it must, among other things, have an effective information processing capability. In today's environment, where there are a great variety of computer systems which an even greater variety of standards, the linking of computers to enable electronic data interchange is difficult and costly. This problem is addressed in this thesis with the development of a Generic, Non-standard Specific Information Flow System which enables computer users to select data which is to be shared in a company and store it in a common database from which it can be accessed by anyone in the company. The information in the common database is stored in a neutral format and the translation to and from this neutral format is done in the background by a Generic Format Translator, which has also been developed as part of this project. It is proposed that the information in the database is carefully selected to minimize memory requirements and speed up access time. A Generic Database Management System has been developed to control the data storage and retrieval processes. Trials of this package in industry have demonstrated that it is workable, cost effective and user-friendly.

SIGNED STATEMENT

I declare that this Thesis contains no material which has been accepted for the award of any Degree or Diploma in any University, and that to the best of my knowledge and belief, contains no material previously published or written by another person, except where due references is made in the text of this Thesis.

I consent to this Thesis being made available for photographing and loan, if applicable and if accepted, for the Degree sought.

Henry C.W. Lau

ACKNOWLEDGEMENTS

The author wishes to thank Associate Professor Manfred Zockel and Dr. Alan Wells for their supervision of this project. Without Dr. Zockel's effort and expertise, this project would not have been possible. Special thanks go to Mr. Pat Tucker and Mr. David Mylechreest who have supported this project and allowed me time to work on it. Further thanks go to Jeff Groves, Darrick Spaven and the staff of South Australian Centre for Manufacturing for all the ways in which they have helped.

Thanks are also due to the staff of N.S. Architects and Gerard Industries for their help in the evaluation of the program.

TABLE OF CONTENTS

NOMENCLATURE

AMF	Advanced Manufacturing Facility
ASCII	American Standard Code for Information Interchange
ATE	Automatic Testing Equipment
BOM	Bill Of Material
CAD	Computer Aided Design
CAM	Computer Aided Manufacturing
CAPP	Computer Aided Process Planning
CIM	Computer Integrated Manufacturing
CNC	Computer Numerical Control
DBMS	DataBase Management System
DEI	Data Extraction Information
DNC	Direct Numerical Control
DOS	Disk Operating System
DXF	Direct eXchange Format
GDBMS	Generic DataBase Management System
GDES	Generic Data Extraction System
GFT	Generic Format Translator
GNSIFS	Generic Non-Standard-specific Information Flow System
ISO	International Standards Organisation
KBS	Knowledge-Based System

LAN	Local Area Network
MRP	Material Requirements Planning
MRP-II	Manufacturing Resource Planning
OSI	Open Systems Interconnection
PC	Personnal Computer
QC	Quality Control
SACFM	South Australian Centre For Manufacturing
WIP	Work In Progress

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

This is the era of information technology and it is crucial that the information is easily accessible and accurate. In manufacturing it has been said that manufacturing is information processing. Yet although most companies have computers doing various tasks they are seldom set up so that the most recent and most accurate information is available to the decision makers. Computer Integrated Manufacture (CIM) was conceived to achieve this by connecting all computers in a company to a central database which is accessible to all machines, and that the information flow from all computers is bi-directional. Unfortunately this integration of computers is in reality difficult to achieve, mainly because of the variety of standards and formats used by the computer operating systems and packages. In this thesis cost effective and easy procedures are proposed which allow the information exchange between any computer systems, packages or operating systems. Computer system, in this thesis, is defined as information processor encompassing hardware and software components.

1.1 MRP-II and CAD

With the advance of computers, manufacturers are able to enhance their productivity by introducing new technology which would be difficult to implement without the help of computers. One of the very important production tools in manufacturing is Material Requirements Planning (MRP) which allows production planners to determine when to order parts, based on the analysis of parts on hand, work in progress (WIP), parts on order and the Bill Of Materials (BOM).

MRP helps considerably in production planning and control, and subsequently improves the productivity of the company. However, this method is not a perfect solution to production control because it is mainly concerned with inventory, production schedules and parts forecasts, whereas important areas, such as resource planning, capacity planning, sales forecasts versus actual sales, are not addressed in MRP system. These aspects are addressed in the Manufacturing Resource Planning (MRP-II) systems, which take into account the various phases of planning undertaken in a manufacturing organisation and which give due consideration to all the resources employed to ensure balance and optimum utilisation of available resources. In short, MRP-II is a management information system which provides planning, simulation, monitoring and performance appraisal tools and information, which can help a company achieve its goals and strategies (Flack, 1985).

MRP-II is an expansion of the traditional MRP system and most important of all, it is a fully integrated system with an internal database which stores information normally located in various areas including inventory, resource planning data, costs of various operations, schedules, forecasts etc., which may be required by other manufacturing subsystems e.g. Computer Aided Design (CAD). If subsystems are linked electronically then for example, the design engineer on a CAD package can access such things as inventory records, work-in-process levels, quality data and cost information and pass the Bill Of Materials (BOM) to the MRP-II system to provide the basis for external ordering of material and components. The electronic data transfer effectively eliminates errors associated with manual transfer of data between these two systems.

An early part of this project was to establish a link between AutoCAD (a proprietary CAD package) and IMPCON (a proprietary MRP-II package) for the transmission of BOM information. The details of this link are discussed in Chapter 3.

1.2 Generic Link

The work with AutoCAD and IMPCON highlighted the need for a generic link between CAD & MRP-II systems which have different internal formats. The link was made generic to allow most CAD packages including AutoCAD, CADDSMAN or CATIA, to communicate with different MRP-II packages, such as IMPCON, MENTAT and BPCS.

1.2.1 Ways to Achieve Linking of Systems

There are three ways to achieve the linking of systems for information exchange :

(a) Turnkey systems - A turnkey system consists of several computer systems and software which can communicate with each other so that information can be shared by all departments including design, planning and manufacture. Computer companies such as SAP America Inc., IBM and Hewlett Packard have turnkey systems aimed at implementing Computer Integrated Manufacturing (CIM) (Greenbaum, 1993; IBM, 1992; HP in Manufacturing, 1990). However, these turnkey systems are "company-specific", i.e. they only work with the software which supports the standards set by that company. Other popular packages, such as AutoCAD, SmartCAM, are not usually supported in these turnkey systems because these packages do not belong to these companies.

(b) Specific Format Translation - In this approach, if data is to be transferred from system A to system B, a specific program has to be written to convert the data from the format of system A to the format required by system B. For example, the transfer of the BOM from AutoCAD to IMPCON is achieved by a specific conversion program, written in AutoLISP (Zockel & Lau, 1990), and another program has to be written if data is to be transferred from IMPCON to AutoCAD. This is obviously "software-specific" as new conversion programs are required for each new package.

(c) Neutral Data Format - Unlike turnkey systems and specific format translation, neutral data format is not company or software-specific. As shown in Fig. 1.1, the neutral format acts as a bridge which links two systems together. It can be regarded as the "critical piece of the puzzle" to make two systems "talk" to each other (Millett, 1990) because it is a format which is "understood" by the other systems. The functions of neutral formats are covered in Chapters 3 and 4.

As indicated in the previous context, "Turnkey Systems" and "Specific Format Translation" are not appropriate for the generic linking of manufacturing subsystems and the neutral format appears to be the best choice among these three options. The generic link between CAD and MRP-II systems, using neutral format, is covered in Chapter 3.

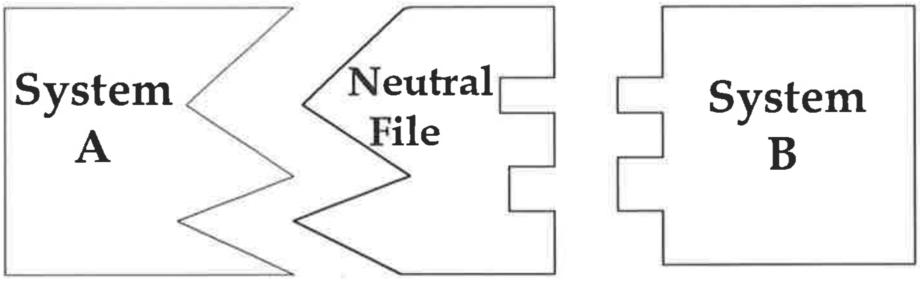


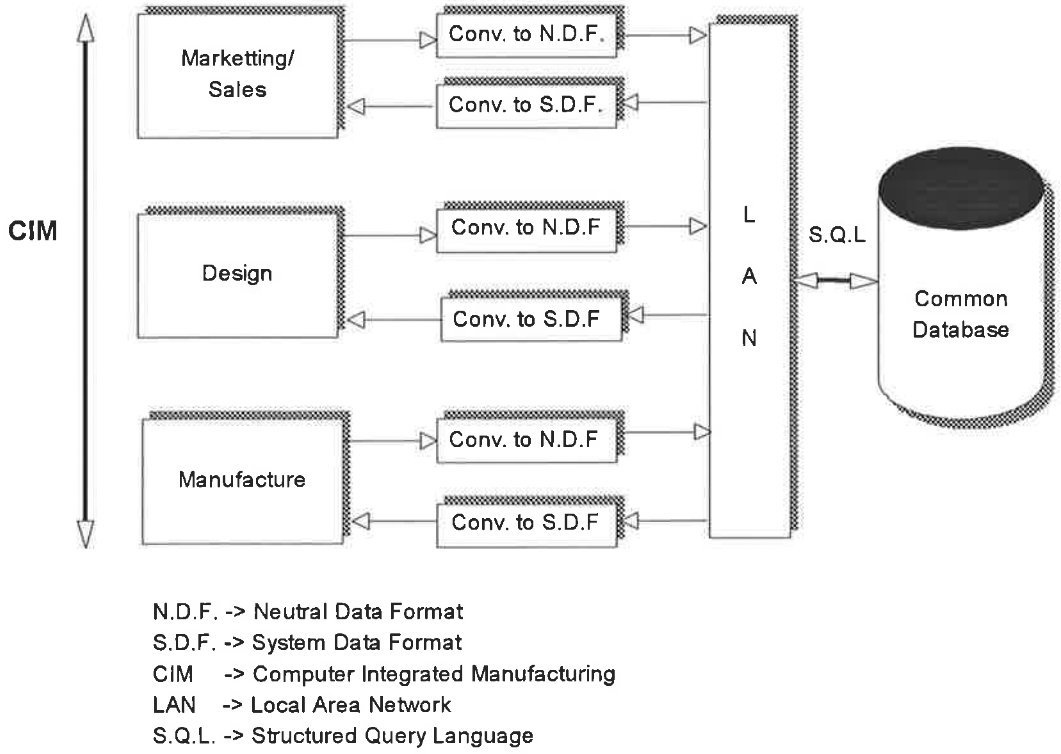
Figure 1.1 : Linking of Two Systems Using Neutral File

1.2.2 Linking of Groups of Systems

In actual situations, companies have employees working in areas of different job responsibilities such as design offices, administration and in the machine shops. Cole (1989) points out that efficient data integration among various computer systems in a company will create a "tightly controlled and responsive manufacturing system". As shown in Fig. 1.2, a company can be divided into 3 groups, namely (a) marketing, (b) sales/design and (c) manufacture, with each group using several different computer systems. Ideally, these systems are linked to a common database so that information can be shared among these groups.

If data to be shared by these groups is stored in the common database in a neutral format then other systems can easily access it. The Local Area Network (LAN) serves as a means of delivering the neutral data file to its destination where it is converted to the system format. Motorola has been using this neutral format concept to integrate their design and manufacturing groups for data transfer (Millett, 1990). Millett states that "in this age of dynamic technological advances Neutral Manufacturing Data is the sanctuary Motorola has chosen for the perpetual integration of design and manufacturing". It should be noted that the neutral format approach also requires specially-written conversion programs, but this is different from the generic information flow system, proposed in this thesis, which performs format conversion transparently.

If all the data, including drawing files was to be stored in the common database, a large database would be created with large memory requirements. In order to rationalise the



N.D.F. -> Neutral Data Format
S.D.F. -> System Data Format
CIM -> Computer Integrated Manufacturing
LAN -> Local Area Network
S.Q.L. -> Structured Query Language

Figure 1.2 : Generic Linking of Groups of Systems

storage system, it is suggested that only alphanumeric data is stored there, because drawing files can be sent for viewing if they are required by the other system.

If only selected alphanumeric data is stored the common database can be much smaller, is more easily managed and can be accessed more quickly. The alphanumeric information can further be arranged in two groups; one group would deal with the alphanumeric data related to manufactured components such as quality data, costs data, BOM , while the other deals with general data such as labour and material costs, process times and machine availability.

The size of the common database can be further reduced by only storing information which is needed by other systems. For example, the system source datafile (alphanumeric file) from SmartCAM is typically 40 pages of information which requires about 400 kbytes of memory (one A4 size page roughly occupies 10 kbytes of disk space). Instead of storing the whole datafile to the common database only the required data, such as the tool sizes, are stored. This data may only be one page, so that the disk space required is reduced to 10 kbytes. The same principle can be used with other systems; for instance storing only the BOM information from CAD system, or the specific cost and inventory data from the MRP-II system, saves space. Due to its much reduced size, such a database enables faster and more efficient data transfer, is more economical on hardware requirements, and makes it easier for the user to deal with several systems at the same time.

1.3 Traditional Way to Achieve Information Exchange

Traditionally, data transfer among computer systems is achieved by using established standards which were designed for specific tasks or operations. For example, PDES (Product Data Exchange Specification) is used for product design data exchange, IGES (Initial Graphic Exchange Specification) for CAD files exchange and IRDATA (Industrial Robot Data) for robotic data exchange (Mult *et al,* 1991). The pre-requisite of these standards is that all the data to be exchanged has to be in a strictly-defined format which means translator programs are required to convert the system file to the specific format and vice versa.

The transfer of data with the PDES standard works like this. Firstly, a complex translator converts the CAD drawing file to a file with the PDES standard; the data is then retrieved from this file by other systems, like CAM, CNC machines or robots, provided that these systems have conversion programs which translate the PDES format information to that required by these packages. The reality is that most companies have several systems such as Computer Aided Design, Computer Aided Manufacture, MRP etc, so that more than one standard is necessary to achieve data flow among all the systems. The success of these standards also depends very much on the exact type of computer packages and computers being used in the company.

Application-specific standards make the integration of computer systems in a company difficult and unaffordable for most small to medium size companies. Some big companies use IBM turnkey system because this achieves “seamless” integration of computer systems. However, most small to medium size companies cannot afford such a system and buy specialised system for special functions.

Studies of the information flow between areas or departments in a company (discussed in Sections 2.6 and 2.7) indicate that there are no clear methodologies available to achieve information flow which is independent of the standards being used in various subsystems. Consequently, a Generic Non Standard-specific Information Flow System (GNSIFS) is proposed in this thesis which is cost effective and therefore affordable even to small to medium size companies (Lau & Zockel, 1994).

1.4 Generic Non Standard-specific Information Flow System

The question arises as to how data can be extracted from, and used by, computer systems if the formats and standards used by these systems are not to be taken into consideration. The concept of the non-standard-specific approach is that the output information, e.g. tool size data from SmartCAM, is directly extracted from the SmartCAM system datafiles by the user with the help of a special written program (explained in Chapter 4). Because of this direct data extraction, the format or standard of the files is no longer important. But that raises another two questions. Firstly, what happens if there are no alphanumeric (text) datafiles available ? Secondly, can the users easily identify the data in these datafiles ? These two points are important as they determine whether this non-standard-specific approach is viable or not. Studies (Zockel & Lau, 1990, Lau & Zockel, 1992) show that alphanumeric datafiles are available from virtually all computer systems including CAD and MRP-II systems (discussed in Chapter 2). Alphanumeric data from AutoCAD is available from drawing files with DXF format, which is the unique data format for AutoCAD, but DXF files are not easily read. However, AutoCAD has a built-in command called **attribute** (Raker *et al*, 1990) which allows users to extract text from the drawing files. Similar functions are available for other CAD packages like CADDSMAN, CATIA etc. with different function (command) names. MRP-II packages also produce text data files for users to perform data exchange or for references purposes. Another study (discussed in Chapter 6) has revealed that the computer systems of the advanced manufacturing equipment, such as Selective Laser Sintering Machine (SLS), Laminated Object Manufacturing Machine (LOMM) and Non-Contact Laser Scanning Machine, produce comprehensive technical datafiles in text format.

As for the second question regarding data identification with datafiles, the experience when using the self-developed program in the tests in companies (covered in Chapter 6) show that all the data in system datafiles can be read and identified without difficulty. The processes of data identification from the system datafiles based on the non-standard-specific approach is covered in Chapter 4.

1.4.1. Features of a Generic Non Standard-specific System

A generic, non-standard-specific, rationalised information flow and storage system for computer-based manufacturing is developed with the following features :

(a) Independence of number and types of computer systems

(b) Ability to handle alphanumeric data

(c) Ability to handle various computer operating systems

1.4.1.1 Independence of number and types of computer systems

Since companies may have various types of computer systems, the generic system must be able to deal with any combination of systems. A company may have several CAD packages, (such as AutoCAD or/and CADDSMAN), a CAM package, (e.g. SmartCAM), an MRP-II package, (e.g. IMPCON) as well as word processors and spreadsheets. If the information flow system is generic then it must be able to deal with all of these packages.

1.4.1.2 Ability to handle various computing operating systems

Data must be able to be exchanged between computers with different operating systems such as DOS, Windows, Macintosh, OS/2 and UNIX, as long as the data to be transferred is in text (or ASCII) format. The constraint of using GNSIFS is that all the data must be in ASCII format which is a "cross-platform" standard and therefore supported by all the operating systems.

1.4.1.3 Ability to handle alphanumeric data items

The viability of the non-standard-approach lies in the capability that users can identify data directly from the source system datafiles. This is achieved by bringing up the system datafile on the computer screen so that users can directly view the actual datafile, selecting the data to be extracted from the datafile, identifying the destination file to store the extracted data.

1.5 Objectives of Research

The objectives of this research are as follows :

- To provide a methodology leading to the design and implementation of an information flow system independent of standards.
- To suggest an approach to optimise the size of common database system to achieve faster and more efficient data access and transfer.
- To investigate and develop a methodology for creating an economical and self-configurable information flow system.
- To introduce a tool which plays a significant role in helping to implement CIM by providing instant access to update data from other manufacturing areas.

1.6 Summary of Contents of Thesis

This thesis proposes an information management system which is independent of the formats and standards being used by the computer systems. The overview of the contents of the Chapters is covered in the following paragraphs.

The literature review is presented in Chapter 2. The existing state of research in the area of information exchange is reviewed. It is found that most of the research is focused on the creation of new and better standards which enables the transfer of data from one system with a specified standard to the other with another standard. The survey also covers the state of research in the areas of DataBase Management System (DBMS) used in information flow system, the data extraction methodologies and the format conversion areas.

Chapter three discusses the existing methodologies for linking computer systems for information exchange. Basically, the specific link among computer systems is achieved through the use of established standards as well as neutral formats e.g. IGES. The merits and demerits of using established standards for linking systems are also discussed.

Chapter 4 covers the details of the proposed generic link among computer systems. The concept and features of a non-standard-specific information flow system are presented together with the main components of this system. The methodology for the design and implementation of GNSIFS is also included in this Chapter.

In Chapter 5, a program called GenFlow is developed based on the methodology explained in Chapter 4. This program has been tested in several companies to find out any operational difficulties. Chapter 5 discusses the program structure of GenFlow and it's six main functions.

Chapter 6 discusses the evaluation of GenFlow in three organisation including the Advanced Manufacturing Facility (AMF) of South Australian Centre For Manufacturing (SACFM), N.S. Architects and Gerard Industries. Feedbacks from the companies are also covered.

Chapter 7 contains conclusions and recommendations for further research. The conclusions associated with the non-standard-specific information flow research are drawn. This research not only provides an innovative approach to the design of an information flow system, but also suggests the methodology for a rationalised database storage system. It offers insight into the processes of data identification and data extraction, flexible database structure and established standards adopted in industry. Further work beyond the scope of this research is raised.

2. LITERATURE REVIEW

2.1 Protocols and Communication Links

In a traditional factory, it is common practice to use machines from different vendors and most of the machines are used independently. In order to achieve data communication among these machines, it is essential that techniques and methodologies be devised to link the equipment together so that data can be interchanged. In order to ensure that all data sent by one system is received and understood by the other system, i.e. maintenance of data integrity, the involved computer systems must use the same protocols which are the "hand-shaking" rules to enable computer systems to communicate with each other (Kessler & Shepard, 1994).

The simplest way to achieve a rudimentary linking between a computer and another machine is through the communication links provided on the computer (COM Port and Printer Port) and the machine itself (Mo, 1993). This could be either a serial link (COM Port) of the RS-series (a standard for serial link) or it can be a parallel link (e.g. printer port of PC).

There is a common misconception that serial ports are meant for establishing communication links with computers for data transmission. The reality is that most serial ports use no transmission protocols and as such data integrity is always a concern because there is no common "communication language" adopted by the involved systems (Sangoma, 1995). Serial ports are consequently unsuitable for large scale, accurate data transfer. The standard RS-series link is mainly used to connect tape punches or simple Direct Numerical Control (DNC) type of application. To solve the problem, commercial firms have designed and produced various communication link components, the most common of which are :

(a) Machine Interface Terminal (MIT)

MIT can help accomplish a two-way data communication links between the supervisory computer and other machines by means of serial or parallel links.

(b) Secondary Adaptor (SA)

This is a key communication link component for the integration of multi-vendor machines. The SA comprises a microprocessor which has a mixture of Read Only Memory (ROM) and Random Access Memory (RAM) according to the interface required. This adaptor is used in conjunction with a Local Area Network (LAN) which consists of network cables and network hardware boards for linking computers (Nance, 1992).

(c) Host / Network Adaptor

This is a general purpose control interface connecting the host computer and the local area network. The host adaptor directs data from the host computer to the various nodes of the network. It also ensures data integrity in transmission of data in both directions.

(d) Network Interface Module (NIM)

Programmable Logic Controllers (PLC) are common in automation and this product is exclusively designed to link PLC to the Local Area Network. It performs the same role as the secondary adaptor.

2.2 Needs for Standardisation of Communication Protocols

The communication link components described in the previous section are mostly custom made to suit a specific type of protocol and therefore can be quite expensive. It is not uncommon that both custom-made software as well as hardware are needed in order to achieve data communication between various machines in the company. This is one of the main considerations when making decisions regarding the implementation of such a data exchange system.

In view of the above problems, there is a need for a standardised communication protocol. In an effort to provide a structure for developing standards for protocols, the International Standards Organisation (ISO), which is a non-governmental organisation established in 1947 (ISO Online, 1996), developed the Open Systems Interconnection (OSI) model. This was introduced to the industrial community in 1981 and, as shown in Fig. 2.1, is a “layered” control of communications over a data network (Quisumbing & Le, 1994; Lloyd & Rozmu, 1991).

The main feature of this model is that the protocol functions are divided in layers which are easier to administer and also provide an opportunity for standardisation. Although protocols adopted in this situation are rather complex, functions in each layer can be modularised so that communication personnel can deal with each layer separately. The objective of the OSI model is to establish a framework that will allow any conforming computer systems to be connected together for the exchange of data. It should be pointed out that the OSI model is not a standard in its own right although it can be used for the development of standards. The detailed functions of the OSI model are described in a number of publications e.g. (Lloyd & Rozmu, 1991).

The first three layers of the OSI model indicate the interfaces between terminals and a shared network while the other layers indicate the end-to-end connections between software during communication. The functions of each layer are given below :

- Layer 1 -- physical

  Defines the electrical and mechanical aspects concerning the interfacing to a given physical medium.
- Layer 2 -- data link

  Establishes the format and the procedure for communication within a network channel.
- Layer 3 -- network

  Specifies how to communicate between network.
- Layer 4 -- transport

  Ensures the end-to-end exchange of data and selects the optimum travel path between networks.

*   Layer 5 -- session

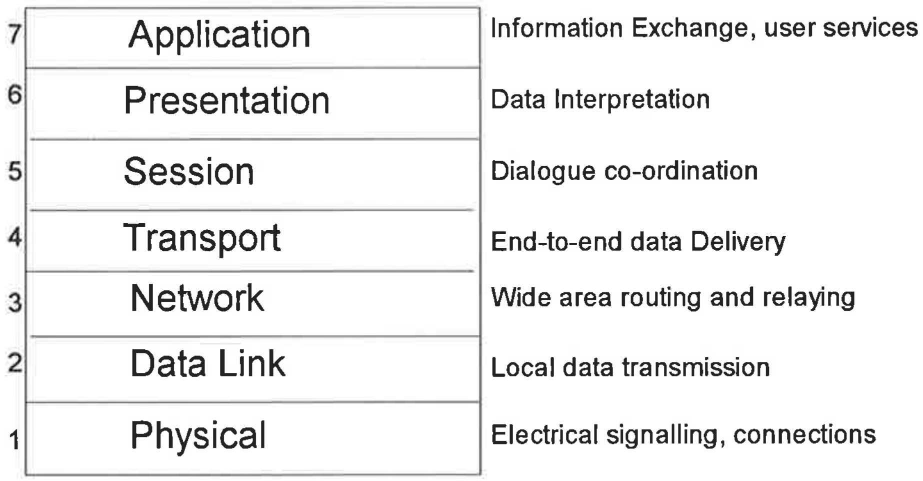
Negotiates connections with another layer and manages the connection.

*   Layer 6 -- presentation

Interprets the character stream that flows between terminals during the session.

*   Layer 7 -- application

Provides user interface to the network.



7	Application	Information Exchange, user services
6	Presentation	Data Interpretation
5	Session	Dialogue co-ordination
4	Transport	End-to-end data Delivery
3	Network	Wide area routing and relaying
2	Data Link	Local data transmission
1	Physical	Electrical signalling, connections

Figure 2.1: OSI Reference Model

2.3 MAP/TOP - Their Roles in CIM

This OSI reference model is an "abstract" model as it does not specify how the functionality at each layer should be implemented. Thus there are quite a number of implementations of the OSI model and, as a result, the user is faced with a wide range of choice of networks. This has retarded the progress of evolving it into a generic integration model.

In 1984, General Motors (GM) decided not to wait for the natural evolution of a universally accepted standards and established their own communication protocols, based on the OSI reference model, and announced that they would no longer buy equipment which was not compatible with the protocols (Olsan, Randy & Bahrololoomi, 1988; MAP Information Packet, 1986). This led to the development of Manufacturing Automation Protocol (MAP) and GM persuaded a large number of its computer suppliers to help develop equipment which would conform to this standard (Nicoletti, 1987; Richardson, 1987).

As MAP is mainly designed for communication on the factory floor environment, other protocols became necessary to communicate in an office environment e.g. Technical Office Protocol (TOP). TOP is also based on the OSI reference model with some specification differences from MAP. The MAP/TOP specifications have been modified in recent years, the latest specifications being MAP version 3.0 and TOP 3.0.

Together, MAP/TOP provide an open systems interconnection protocol suitable for the transfer of data between equipment located at various areas on the shop floor and in the office, and is therefore an important step for implementing CIM which requires bidirectional information exchange among all the computer system within an organisation. MAP/TOP are regarded as "backbone" networks which link planning, design and manufacturing and from which small subnetworks branch out to link other automated equipment to form manufacturing cells (Ferme, 1993; MAP/TOP, 1988).

It was forecast that MAP/TOP would be well supported by companies, and in the 90s the implementation of CIM would be much easier due to the worldwide acceptance of MAP/TOP standards. However, this forecast did not eventuate. Ferme (1993) points out that an American Survey in late 1990 shows that MAP has not been well received by companies (having only 17% of the market in Europe) though some companies in Japan, such as Fanuc's Tsukuba machine tool plant, Toyota's machine works in the Tadomisaki plant and Komatsu's machine tool division have implemented MAP 3.0. The reason for MAP's failure, as Ferme (1993) indicates, is due to the lack of an unanimous agreement of the specifications of MAP 3.0, and the general acceptance of Ethernet by Digital Equipment Corporation and System Network Architecture (SNA) by IBM in the manufacturing sector.

2. 4 Networking Technology

Networking computers for data transmission has developed strongly in recent years and big computer companies such as Microsoft, Novell, IBM have developed reliable Network Operating Systems (NOSs) in this area (Nance, 1992). Normally there is no difficulty to transfer files from one workstation to the other and some NOSs e.g. Netware and Windows NT, can even handle file transfer among different computing platforms e.g. DOS to Macintosh or UNIX to DOS. It should be noted that proper networking ensures the transfer of files among the computer systems within the network, based on some handshaking rules (protocols). It does not deal with the delivery of a specific part of the file (data), e.g cost data of a MRP-II file, or the conversion of formats (reformatting) of the data which are the two important features of GNSIFS proposed in this thesis.

2.5 File Transfer Programs

If computers are connected together via LAN and a NOS is installed, file transfer between the computers can be accomplished by using the built-in functions of the NOS. If the connection is done via network cables such as serial or parallel connection cables, commercial programs such as Norton Commander (Norton User Guide, 1993) can be used to achieve file transfer. These programs allow users to view the files of the other computers and perform single or multiple files transfer easily. Some programs, such as PC Anywhere or LapLink for Windows, can be used to "remotely access" a computer which is located on a "remote" site through telephone line connection (Windows Sources Australia, 1995). These programs are used to access and transfer "remote" files but again they do not deal with the delivery of specific data or format conversion of the files.

2.6 Standards Used in Design Area

A particular file from computer system A can be sent to computer system B if the two systems are within the same LAN. However, that file is useless if the format standard of the file is not supported by the receiving system. For example, if a .DWG file from AutoCAD is transmitted to ComputerVision system the file cannot be used since ComputerVision does not support .DWG format. However, if that file is in IGES format, which is a neutral format for CAD files, ComputerVision is able to interpret it. Jon Fairall (1992), stressed that "standards of all types will be the single overriding issue of the 1990s. It will be more important than improved CPU performance, more important than better software."

Douglas (1992) points out that Computer Aided Design (CAD) is the area which has been well-developed in terms of data exchange. IGES is the accepted standard for CAD but Douglas (1992) states that due to the popularity of AutoCAD, Direct eXchange Format (DXF), which was introduced by AutoDesk to allow other programs to transfer information to and from AutoCAD, has become another "defacto" standard in CAD systems. Most of the CAD systems running on PCs, such as CADDSMAN and QuikDraw, support DXF format.

With regard to graphics or paintbrush files the common formats adopted include EPS (Encapsulated Postscript File), GIF (Graphics Interchange Format), PCX (Zsoft PC Paintbrush Bitmap), TIF (Tag Image File Format), WMF (Windows Metafile) and BMP

(Bitmap). Sullivan (1992) indicates that the data conversion of one type of graphical format to another graphical format is easy with the commercially available conversion programs. In fact, packages like Paintbrush, CorelDraw (CorelDraw, 1993) support at least two of the above graphics formats. Consequently, it can be summarised that the format exchange among CAD files as well as graphical files is quite well developed (CAD Conferencing, 1993; CAD/CAM Planning Guide, 1988).

2.7 Established Format Standards

In manufacturing, it is important that data transmission is possible among all systems in a company, like MRP-II, CAD, CAPP, CAM, ATE and the Store,. The literature survey shows that there are examples in industry where neutral files have been adopted for the integration of manufacturing subsystems for data exchange. Millett (1990), senior manufacturing engineer of Motorola Incorporated, states that since 1987, neutral manufacturing data has been utilized in Motorola for the data exchange between CAD systems and other manufacturing systems (Fig. 2.2). He points out that "neutral data can be uninterrupted or timeless because the format is independent of all brands of CAD and factory equipment. Because the format is CAD specific, and not brand specific, it can be created from most (if not all) electrical CAD software used today (Millett, 1990)". Some companies make use of established standards such as IGES and PDES to develop a neutral format file. Everitt (1988) states that the U.S. Navy has linked CAD, CAM and Process Planning by creating neutral files based on the Product Data Exchange Specification (PDES) standards. Everitt emphasizes that the Navy (U.S.) receives data from many

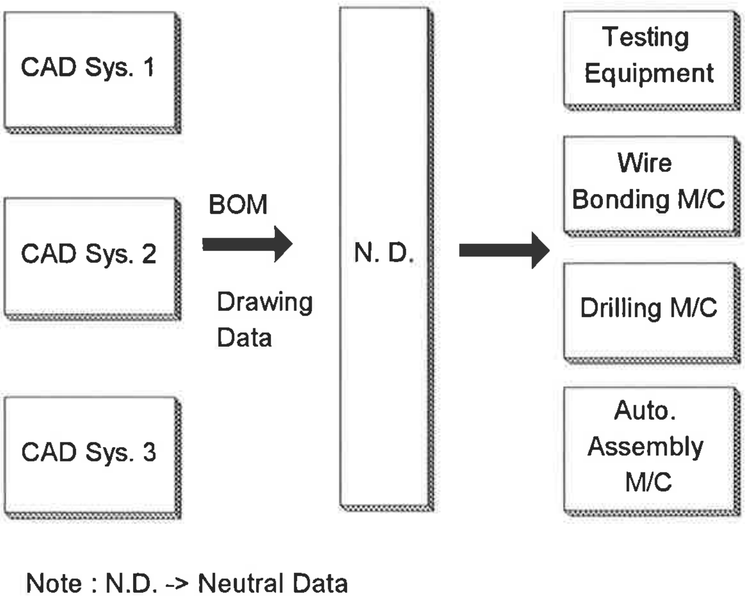


Figure 2.2 : Data Transfer Using Neutral Format in Motorola

different contractors for many different computer systems. By using neutral data files, based on PDES, the Navy ensures that the data from outside contractors is "understood" and utilized by all other computer systems. However, in order to convert data to PDES neutral format, specific conversion programs are necessary. In this project, the translation programs were specifically developed on the SUN Microsystem UNIX system using the "C" programming language. Exact number of translation programs required was not revealed in the publication.

Luca (1993) suggests the adoption of "official' and "defacto" standards for the integration of computer systems. He describes a "step by step approach" to achieve computer integrated manufacturing. In the example with a Company called BALTEA in Italy, a program called SILOS was specifically written to integrate the existing sales information system with other computer systems. The second stage was to expand the system integration to other areas. Special programmes had to be developed in C++ (an object-oriented programming language) and 4GL (a programming language used in mainframe systems) to link the systems together.

2.7.1 STEP (ISO-10303)

In recent years, STEP (ISO-10303) has become a widely used standard for product data representation and exchange (Bohn, 1996). STEP includes a series of International Standards with the aim of defining data across the full engineering and manufacturing life cycle and the ability to share data across various applications. One of the benefits of using STEP as the product data exchange standard is that it works with a data description language called EXPRESS which can be used to manipulate data of STEP-compatible files (ECRC, 1996). Like BASIC, EXPRESS is a procedural computing language which can perform string (text) manipulation, data management (data retrieval, storage, search) on files which comply with STEP standard. With this computing language capability, EXPRESS can be used to extract "selected" data from datafiles and perform data format conversion.

The other benefit of using STEP is that a program can be written in EXPRESS to convert the STEP data format into a relational data format. Once the data is in a relational database, searching and retrieval functions can be carried out more effectively. This is particularly beneficial when dealing with a significant amount of products within a company because relational data model provides efficient control of data which is put in different data fields. However, small to medium size companies may not afford to adopt this standard for data exchange. Bohn (1996) points out that STEP has a strictly-defined format for storing product data (geometric and alphanumeric) and that means complex format converters are essential to enable data exchange between various systems complying with this standard. For example, a CAD system using IGES standard must have converters to translate the IGES system files to STEP-compatible files before they are transferred to other systems and all the incoming STEP-compatible files from other systems have to be converted to IGES-compatible files before they can be accepted by the CAD system. This means that two conversion programs are required for one system and the number of conversion programs increase when more computer systems are involved. This can be an expensive process as these conversion programs requires to be developed by experienced EXPRESS (or other programming language such as "C") programmers.

2.7.2 Review on Format Standards

The review shows that companies are using neutral files with self-designed standards (Millett, 1990) or established standards (Mult *et al*, 1991; Luca, 1993; Bohn, 1996) to achieve integration of manufacturing systems. More important of these standards, as summarised by Mult *et al* (1991) and Bohn (1996), are :

GKS-3-D	=	Graphical Kernel System
CGI	=	Computer Graphics Interface
CGM	=	Computer Graphics Metafile
IGES	=	Initial Graphic Exchange Specification
SET	=	Standard d Exchange et de Transport
VDAFS	=	VDA-Flachenschnitt
PDES	=	Product Data Exchange Specification
STEP	=	Standard for Exchange of Product
CAD-NT	=	CAD-Normteile
CLDATA	=	Cutter Location Data
IRDATA	=	Industrial Robot Data
PHIGS	=	Programmers Hierachica Graphic System

As explained in Section 1.3, the pre-requisite of using any of these well established standards is that all the data for exchange has to be arranged in a strictly-defined format. These standards are designed for a particular type of manufacturing subsystem, e.g. PDES for product design, IGES for CAD files, IRDATA for robots. However, most companies have several manufacturing systems, so that more than one standard has to be used in order to achieve data flow among all the systems. Rui *et al* (1988) outlines an approach to automate information transfer in manufacturing systems by adopting various standards, including MAP/TOP, IGES, PDES/STEP and EDIF (Electronic Data Interchange Format) which is a standard for the design and manufacture, in both Very Large Scale Integration

(VLSI) printed circuit board production. Sargent (1990) suggests a computer-based materials information system based on the adoption of standards including PDES, STEP and EDIFACT (Electronic Data Interchange For Administration, Commerce and Transport). Some companies provide turnkey systems which guarantee "seamless" integration of software applications provided that these applications comply with the standards set by the company. Greenbaum (1993) states that a computer company SAP America Inc. introduces a package called "R/3" which is a software "suite" which provides links between finance, manufacturing, sales, distribution and human resources applications. IBM Application System/400 (IBM, 1993) also provides links of packages which follow the IBM specified standards. Another giant computer company, Hewlett Packard, developed a system named "Integrated Manufacturing Solution" for manufacturing companies. Software applications running on UNIX are provided by HP to achieve data integration. This is a turnkey system which includes application softwares, computers and networking facilities for customers. Australian companies such as Cadbury Schweppes Pty Ltd, Gayland (Melbourne) and So-Rite Engineering Pty Ltd are using this system (HP in Manufacturing, 1993). Another company, EDS Unigraphics started a project named AeroSTEP in 1995 with the aim to provide "digital" pre-assembly data exchange between various departments and assembly plants for Boeing with STEP/PDES as the standards for product data definition (EDS Unigraphics, 1995). With STEP and PDES as the formats for data exchange, product information, which includes geometric as well as alphanumeric data, can be shared among various groups of the company.

The successful use of application-specific standards depends very much on the exact type of computer packages and computers being used and is therefore not affordable by most of the medium and small size companies, as individual system elements (software and hardware) may be an unacceptably highpriced “overkill” for their requirements.

Based on this literature review of standards, it is clear that the existing methodology concerned with integration of manufacturing systems for data exchange is focused on the adoption of standards which are supported by the involved systems but that standards-independent data transfer study requires further work.

2.8 DataBase Management System

The Generic DataBase Management System (GDBMS) of GNSIFS handles the storage and retrieval processes of “selected” text data (no full files or geometric files) between the various systems and the common database. Such a database system must be flexible in order to cope with data transfer among different combinations of computing systems. The DataBase Management System (DBMS) handles the data transmission between the database application program and the database. The DBMS and the database application program belong to the computer system, e.g. DBase IV or Microsoft Access but in reality, the DBMS and the database application program work together as one integrated unit. For example, the database program developed from DBase IV works with the DBMS which is an integral part of DBASE IV package (DBase IV, 1991). The DBMS stores data in the database as well as retrieves data from the database and responds to requests from the application software which is designed by application developer (Leonard, 1992; Stamper *et at*, 1990).

2.8.1 Various Types of Data Models

There are various types of database models (Buddenhagen, 1986; Coombes, 1987; Derus, 1986; Navathe, 1992) including relational, hierarchical, network, free format, multi-user distributed structures and object-oriented data structure. These are summarised below :

(a) Relational - Relational databases are the most common data models (Glen,1992; Markowitz, 1994). House(1991) states the advantages of the relational database as :

(i) it is well-developed with a large number of supporting tools and publications for those who want to develop a relational data model for special application,

(ii) it requires only 2 Megabytes RAM memory or more , and

(iii) it is reliable to handle alphanumeric files.

The relational database files have no related parent files i.e. all files are indepenent. However, relationships between files can be created, based on common data fields in the files, e.g. Employee Identification Number. For example, there are two files in the database; an Employees File, which records the details of employees such as Name, Address, Position, Employee Identification Number, and a Sales File, which records the details of sales data such as Sale Amount, Sale Description, Employee Identification Number responsible for the sale. When the application program requests the DBMS to look for the sales amount for certain employee with name, e.g. George Smith, the DBMS will get information from the two files i.e. Employees File and Sales File because the Employee Identity Number is in both files.

(b) Hierarchical - A hierarchical database is one where the relationships between the files form a hierarchy where the files are ranked from top to bottom, with higher level files being the parents of lower files. In a hierarchy, a parent file may have several lower-level files under it but the lower-level file may have only one parent file e.g. directories and subdirectories.

(c) Network - A network database is similar to a hierarchical database except that a file can have multiple parents.

(d) Free Format - Free format database is in the form of text files, like the text part of this report. Any character, word, or combination of these can be found changed, deleted, inserted or replaced using the text editor.

(e) Multi-user Distributed Structures - Multi-user, distributed structures databases, as the name implies, can have any of the above structures. The main feature of this model is that they allow the database to be read by many users at the same time. However, only one user is allowed to write into a record at a given time to avoid data inconsistency.

(f) Distributed Data Model - This model is designed to be used in "heterogeneous" environment where data is distributed in different levels such as corporate data in factory level, design data in center level. The data in various levels is separated from the others by a security "wall" but particular data can be interchanged between specified levels provided that there is authorized permission by top management (Bell & Grimson, 1992). This model is appropriate for companies where there is a complex data structure and data security at different levels is essential as in a manufacturing company. Although data is stored in various "levels", it is accessible to all and therefore from the user's perspective, this data model is no different from a central database and basically whole files are transferred to the user without any format conversion.

(g) Object-oriented - In an object-oriented database model, an "object" is a representation of a complex data type. Barry (1992) points out that DBMS based on object-oriented data model can maintain the integrity of complex entities by providing a way to manage complex data types through "inheritance", "encapsulation", and "polymorphism". However, Khoshafian (1991) cautions that object-oriented databases have their own problems. Object-oriented database technologies and products are still in their infancy and lack a consistent, agreed-upon conceptual model. In addition, it requires computers of at least 8 Mbytes RAM to achieve efficient operation (Price & Gray, 1994; Glen, 1992; Ahmed *et al*, 1992; Loomis, 1992).

2.9 Data Exchange Among Computer Packages

Data transfer among computer applications from different manufacturers is a perennial problem despite the efforts put in by computer firms over recent years (Barrett, 1993, Biegel *et al*, 1994). Microsoft has developed a standard called Object Linking and Embedding (OLE) which is aimed at transferring information between all Windows applications (Citidata, 1994). OLE uses a set of routines known as Dynamic Link Library (DLL) to perform the data transfer operations and allows one application to request information from another. The requesting application is called the *client* and the application providing the information is called *server*. The information passed from server to client is known as the *object*. An object could be a single cell of a spreadsheet, a few lines of text or a complex drawing. The information supplied by the server can be either *embedded* or *linked* into the current document. Embedding is similar to copying, using the Windows ClipBoard (a temporary storage area located in computer memory) except that to edit the object, the user must choose the object in the destination document and the application in which the object was created automatically opens. Therefore, there is no longer any need to keep switching between applications to view and change information. Unlike Embedding, Linking does not create a copy of the object but creates a link with the source document. Consequently, when the user edits the linked object it is the information in the source application which is edited. A single source may have links to several other areas and if any of these destination documents make a change to the source document, the change will appear in the source and all the other documents. OLE is a powerful feature for data transfer. The pre-requisite of using OLE is that the linking or embedding must be performed in the Windows environment and the software applications involved must support OLE standards. In manufacturing, a number of popular applications do not support OLE. SmartCAM and QuikDraw are still DOS programs but some "high end" products, such as ComputerVision and Pro-engineer run on the UNIX system. While OLE is appropriate for data transfer between a group of applications developed by the same company, e.g suites of packages, it is not appropriate for manufacturing computer systems such as CAD, CAM and MRP-II systems.

Several big software companies including Microsoft, Borland and Lotus have developed and released "suites" of software which include windows applications such as spreadsheet program, a wordprocessor program, a database program and other application programs. These suites are characterised by "seamless" integration between involved applications (Gavron *et al*, 1994). Borland's suite includes Paradox 4.5 for Windows, Quattro Pro 5.0 for Windows, WordPerfect 6.0 for Windows while Lotus's suite includes Ami Pro 3.01, Approach 2.1, Lotus Freelance Graphics 2.01 for Windows, Lotus 1-2-3 Release 4 and Lotus Organiser 1.1. Microsoft Office suite includes Word 6.0, Excel 5.0, Access 2.0 and Microsoft Mail. Integration is the main feature of these suites. While the software developers guarantee seamless integration of their suite of applications for data exchange, this promise only applies to software released by their own company (Gavron *et al*, 1994). Venditto (1994) pointed out that big software companies such as Microsoft, Lotus and Novell want to set the standard in the huge computer market because the success of setting the standard will bring more profits to the company. The huge profit of Microsoft due to the popularity of Windows 3.x is a good example. The release of Windows 95 in 1995 with 32-bit Windows applications standards indicates that Microsoft intends to set the standards in the 32-bit Windows platform.

In other developments in the data exchange field, Richer & Associates Systems Group released a software called R&A/Synergy. As reported in a magazine called Midrange Systems (November, 1993), this software enables users of a package called Computer-Aided Software Engineering (CASE) to transfer data between other applications. Dodge (1993) also indicates that Computer Associates International Inc's (CA) Visual Express, which is a data-extraction and reporting program, combines such tools as CA-RET 2.0, report writer and QbyX (query-by-example) tools for the company's host database products. He states that CA expects Visual Express to be widely accepted because with one product users can perform extraction, merging and reporting across databases from different applications. As discussed in Section 2.7, a number of turnkey systems have also been developed, which are characterised by the capability of seamless data exchange among the application softwares included in the individual turnkey systems.

The survey on data exchange among computer systems shows that computer companies have been working in this area in the past years and a number of products have been developed which guarantee seamless integration of individual applications for data exchange. Nevertheless, the survey also shows that the integration only applies to those applications which are in compliance with the specified standard. In other words, no generic data exchange system, which guarantees "seamless" data transfer among various computer systems regardless of computer platforms, software and standards being used in those systems, is available at present. Consequently, a generic system is needed with non-standard-specific data exchange capability for the data exchange between computer systems.

2.10 Database Formats

There are mainly three types of formats for alphanumeric data namely Comma Delimited Format (CDF), Standard Data Format (SDF) and Blank Delimited Format (Raker *et al*, 1993). CDF has commas separating the data fields of the file, e.g. "PartCode1,Description1,Quantity1", and the spacing of the data field is dependent on the data width within the field; for instance, "PartCode10" has a longer data field than "PartCode1".

In the SDF format, the data is separated by spaces and the fields have a standard field length. The data field length is standardized or pre-formatted to a specified value regardless of the actual data value length. If the data exceeds the value, the data is truncated. The following example shows the field lengths of the data are pre-formatted to “15”, “20” and “10” respectively.

```
PartCode1      Description1        Quantity1
PartCode1      Description1        Quantity1
PartCode10     Description10       Quantity10
<-----15------><--------20---------><---10--->
```

Most of the database development tools, such as Dbase IV or Microsoft Access support both CDF and SDF format. Simpson (1993) describes the necessary steps to write a text file to a database. He also mentions that apart from CDF and SDF, DBase supports the Blank Delimited Format where there is a blank space separating the data fields, for example, “PartCode1^ Description1^ Quantity1”, where the notation sign “^” indicates a blank space.

Although most of the popular commercial packages support CDF, SDF and BDF, some computer system may require other formats. For instance the computer system used in the Selective Laser Sintering Machine Model 2000 uses the character “/” so that the data appears as follows :

part-6-STL_filename/usr2/sls2k/geometry/stl/parts/6014/_stl/6014-05-01.bo

part-7-STL_filename/usr2/sls2k/geometry/stl/parts/6024/_stl/6014-01-01.bo

Even though this is not a common format, it has been included in the generic file transfer package which is discussed in Chapter 4, and other formats can be defined.

It should be noted that programs such as Excel, Lotus, Access and DBase have a built-in feature which allows the import of external text files. However, if the program itself does not have this "files-importing" capability, it is difficult (if not impossible) to perform the import operation as stated above. CAD systems such as AutoCAD, CADDSMAN and QuikDraw allow the import of files with neutral formats, e.g. IGES and DXF, but the import of text file is not supported.

2.11 Summary

The literature survey shows that while networking technology to transfer files from one computer to the other is quite well developed, the transfer of selected data and the automatic reformatting of data is not easily or economically achievable except perhaps between a limited set of proprietary software applications. Computer firms have made great effort to enable data exchange between software applications. That can be demonstrated by the release of turnkey systems and software suits from companies such as IBM, HP, Microsoft, Lotus, Borland and Novell. However, the promised "seamless" data transfer only applies to their own products and the computing platform must comply with the specific platform such as Windows or UNIX. In some companies, data exchange between various computer systems is done by means of self-designed standards, established standards or turnkey systems. However, these approaches are not affordable by small to medium companies as these integration methodology is basically "standard-specific" or "company-specific". The development of a system which provides the extraction of selected data and automatic reformatting of data is necessary. In addition, this system must be able to handle data exchange within different computing platforms. Consequently, an economical non-standard-specific information flow system would overcome a significant problem.

In the following Chapters, an generic information flow system is introduced which is able to perform "selected" data exchange among various computer systems regardless of the computer platforms and software being used in those systems and in addition, data format conversion is done "transparently" in the background. This infomation flow system is affordable by small to medium size companies because no additional customised format converters are required to enable data exchange among various computer systems.

3. SPECIFIC LINKS

3.1 Introduction

The success of Computer Integrated Manufacturing (CIM) relies on an efficient information flow system. The information flow system must be able to provide accurate, up-to-date and efficient bidirectional flow of information among the computer systems of a company (Inglis, 1993; Ang, 1986; Leng, 1986). As discussed in Section 1.2, there are three ways to link systems together for information exchange, namely turnkey systems, specific format translation and neutral data format. Turnkey systems are usually too expensive for medium and small size companies and are not covered any further in this thesis. The methodology of specific format translation is examined in this Chapter while data integration using neutral data format is discussed in Chapter 4.

Even small companies frequently have a variety of multi-vendor computing equipment and integration of such equipment inevitably leads to a diversity in communication hardware and software (See Section 2.1). As was also mentioned, traditional information exchange systems consist of specific links for the particular computer systems involved in the information exchange process. The links are specific in that particular communication hardware and software are used for the linkage and if any of the involved computer systems are modified or replaced, both communication hardware and software must be changed.

This Chapter, which covers the linking between AutoCAD and IMPCON (an MRP-II package), and which is then expanded to include other manufacturing systems to form a CIM system, is important to highlight the difficulties of linking various systems using specific format translation technique. The following Chapter covers the generic linking methodology which was developed from the understanding gained in these specific links.

3.2 AutoCAD to IMPCON

This project began with a request from a computer company to link AutoCAD (Hill, 1987; Miller, 1988) with an MRP-II package called IMPCON, which is a complete inventory management and production control system.

One aim of the project was to transfer the Bill-Of-Material (BOM) from AutoCAD to IMPCON electronically and to give the designers access to part and process information held in the MRP-II package but without the ability to alter the data.

The initial approach was to use the specific translation approach to get to know the systems with the aim of developing a more widely useable package later.

3.2.1 Data To Be Transferred Between the Two Systems

A brief study identified that the BOM data to be transferred from AutoCAD (Thomas, 1988) to IMPCON consists of part code, part description, quantity, release date, measure unit of part as well as the parent code and description .

When the designer has finished the design of a new or modified component, the part list data, with the BOM data can then be transferred to the IMPCON system. However, in practice, the BOM created by the designer (Design BOM) is frequently modified to get a Manufacturing BOM, which is the BOM used by the MRP-II package. To deal with this situation, it was agreed that there should be two BOMs; Design BOM and a Manufacturing BOM which could be created by manufacturing personnel after modifying the Design BOM.

Differences between Design BOM and Manufacturing BOM can include :

- part codes used in Manufacturing BOM may be different as the coding system in manufacturing area may not be the same as in Design area
- release date of part may be altered based on a more realistic date for manufacturing areas
- materials may be substituted depending on availability

3.2.2 Procedures for the Transfer of Design BOM

The procedure for the transfer of the Design BOM from AutoCAD to IMPCON are as follows :

Step 1 : The person at the AutoCAD terminal creates a part- list containing data of all the parts on the drawing. A special program, written in AutoLISP (AutoLISP Reference Manual, 1989), a built-in macro language for AutoCAD users, is then run and the format of the part-list appears on the screen with all agreed data fields, such as Item No (maximum 4 numerical digits), Part Code (maximum 14 characters), Description (maximum 30 characters), Quantity (maximum 7 numerical digits before decimal and no more than 4 numerical digits after decimal), From Date (no more and no less than 6 numerical digits) and Measure Unit (maximum 6 characters).

As the IMPCON package requires strictly formatted input fields, the correct number of numerical digits or characters must be inserted. In this respect, the custom made program must be able to monitor the input field length so that the correct entry is guaranteed.

The designer is prompted to input the parent code and description of the part followed by all other data. The data is then stored as a text file ready to be transferred to the IMPCON terminal. A third-party communication program, such as Kermit, can be used to send the file to the IMPCON system (Sectiion 2.5).

Step 2 : After the Manufacturing Department is notified that the Design BOM is available at the IMPCON terminal, the Design BOM file is called up by the manufacturing personnel for it's input or modifications to data, such as Part Type ("R" for raw material item, "M" for manufactured part, "P" for purchased part , "X" for phantom part).

Step 3 : When the changes are completed, a specially-written program is invoked to insert the data into the appropriate database locations within IMPCON.

3.2.3 Programs for Data Transfer

Three programs were written in AutoLISP, to facilitate the transfer of data between the two systems. A brief description of each follows.

1. Balloon.lsp - This program allows the user to identify all parts on an assembly drawing by creating balloons or circles with labels on the drawing. Size and position of the balloons can be varied, and each balloon has a number inserted automatically to identify the part. It should be noted that this project was based on AutoCAD Release 9 and with the latest AutoCAD Release13, this function is a built-in command and therefore this subroutine is no longer required.

2. Dplist.lsp - This program allows the user to vary the position and size of the part-list to suit the drawing. Additional fields can be added to those pre-arranged fields which are required by IMPCON and the user is prompted for the data of each part. All data can be edited and when completed the part-list is stored as a text file with a file name assigned by the user.

3. Editplist.lsp - This program allows the user to edit the data in the part-list, delete or add new items, or restore the part-list previously stored in “Dplist.lsp”.

3.2.4 Procedure to Use the Programs

The procedure to use the programs are as follows.

a. To create a part-list or a BOM, the three AutoLISP programs and the drawing can be loaded into the current directory of AutoCAD by typing the following after the command prompt of the AutoCAD drawing editor.

```
command : (load "<filename>")
```

To save time this loading procedure can be eliminated by putting all three programs into a text file, e.g. acad.lsp which will be automatically loaded to the memory space allocated for AutoLISP programs when the drawing editor is called.

b. To make balloons, type “mb” at the command prompt.

The user is then prompted to indicate the position and size of the balloon as well as item number, and the balloons is automatically drawn on the screen. To edit the balloon, type

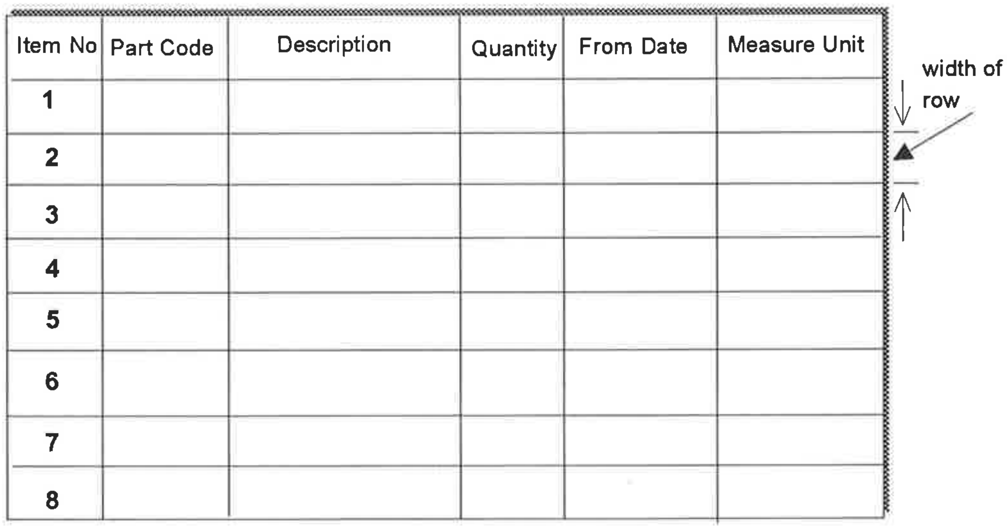
```
command : eb  (meaning erase balloon)
```

c. When all the parts are labelled, the part-list is created by typing

```
command : dplist
```

The data fields contained in the part-list are shown in Fig. 3.1. As explained in Section 3.2.2, Step 1, the number of digits allowed for each data field is fixed so the width of each data field is also fixed. The user is prompted to specify the position of the part-list, e.g. right lower corner or left lower corner of drawing, width of row (See Fig. 3.1) and number of items on the part-list e.g. 8. With this information, the part-list box is created on the drawing and the user is prompted to insert the details of the data fields of individual parts. Editing is allowed after the completion of data input.

All the data is stored in a text file ready to be sent to IMPCON. If changes are required to the part-list, type "editplist" after the command prompt. The format of the input is checked by a sub-routine in the program when the user has finished the input of one data item e.g item no, part code.



Item No	Part Code	Description	Quantity	From Date	Measure Unit
1					
2					
3					
4					
5					
6					
7					
8					

Figure 3.1 : Creation of Part-list Box

3.2.5 Communication Software

To transfer the file to the IMPCON system, a number of communication programs such as Kermit (Kermit User Guide, 1988) or Norton Commander Version 4 can be used. To view the IMPCON data on AutoCAD, a package such as PC Anywhere, X-Windows Emulator can be used to "remotely access" programs which are located on a "remote" computer.

The connection between AutoCAD and IMPCON is a simple RS-series connection between two computer systems. However, the communication connection is more complex when more computer systems are involved in the data exchange process.

3.3 Data Exchange in a Full CIM System

The essential elements of a CIM system is a central database and bidirectional information flow between all the systems. As shown in Fig. 3.2, the manufacturing subsystems for a full

CIM system are divided into three groups, namely Marketing, Design/Planning and Manufacturing. The Marketing group consists of two subsystems, namely Sales/Marketing and Purchase. These are involved in the data exchange with MRP-II system, which belongs to the Design/Planning group, which in turn contains three manufacturing subsystems, CAD, MRP-II and Computer Aided Process Planning (CAPP). Design and planning data is shared among these subsystems as well as the subsystems of the Manufacturing group which are CAM, Automatic Storage and Retrieval System (AS/RS), Automatic Testing Equipment (ATS) and Flexible Manufacturing System(FMS). The data associated with the three activities are :

(a) Marketing

The input and output information for this group are listed in Table 3.1.

Table 3.1 Input/Output Information for Marketing Group

Line No	Output Information	Source System	Destination System
1	Data for setting master production schedule	Sales/Marketing	MRP-II
2	Actual production data for study and analysis	MRP-II	Sales/ Marketing
3	Place orders	Purchasing	MRP-II
4	Delivery Progress, vendor quality	MRP-II	Purchasing

Note : **Line No** refers to the number of the arrow lines shown in Fig. 3.2.

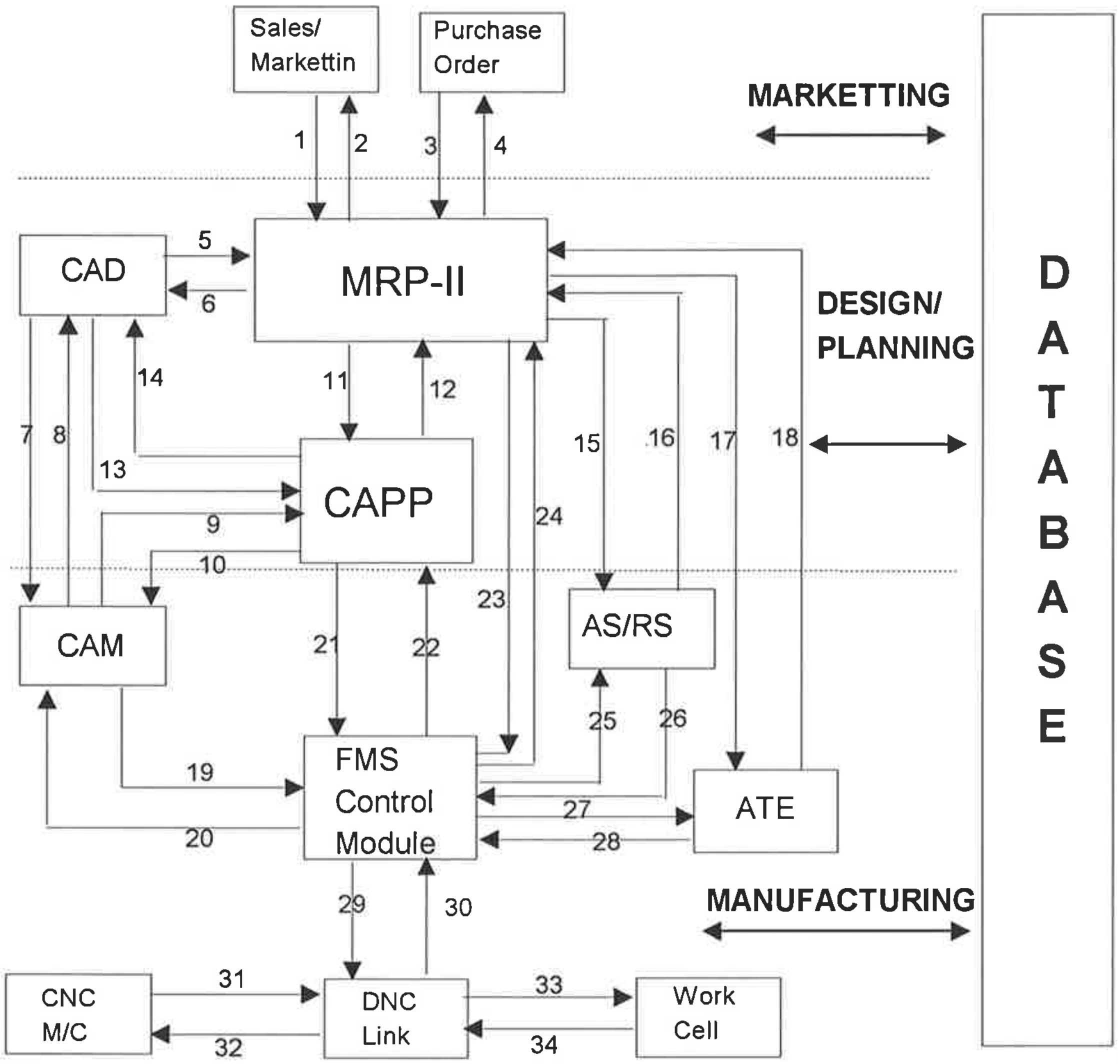


Figure 3.2 : Information Exchange among Manufacturing Systems

(b) Design/Planning

The input and output information for this group are given in Table 3.2.

Table 3.2 Input/Output Information for Design/Planning Group

Line No	**Output Information**	**Source System**	**Destination System**
5	Bill of Material	CAD	MRP-II
6	Inventory records, cost information	MRP-II	CAD
7	Geometric data of design	CAD	CAM
8	Information feedback	CAM	CAD
9	Feedback status and problems found	CAM	CAPP
10	Process data	CAPP	CAM
11	Product information/ cost data	MRP-II	CAPP
12	Process planning summary	CAPP	MRP-II
13	Geometric data	CAD	CAPP
14	Status feedback	CAPP	CAD
15	Component data retrieval, lists of parts to be drawn	MRP-II	AS/RS
16	Inventory situation	AS/RS	MRP-II
17	Retrieval of quality analysis	MRP-II	ATE
18	Quality/measurement results	ATE	MRP-II

(c) Manufacturing

The input and output information for this group are shown in Table 3.3.

Table 3.3 Input/Output Information for Manufacturing Group

Line No	**Output Information**	**Source System**	**Destination System**
19	Status feedback	CAM	FMS C. M.
20	NC programming data	FMS C.M.	CAM
21	Process operations information	CAPP	FMS C.M.
22	Data enquires	FMS C.M.	CAPP
23	Production schedule for FMS	MRP-II	FMS C.M.
24	Orders completed	FMS C.M.	MRP-II
25	Storage and material handling monitoring	FMS C.M.	AS/RS
26	Feedback data as requested	AS/RS	FMS C.M.
27	Quality/dimensional checking	FMS C.M.	ATE
28	Results feedback	ATE	FMS C.M.
29	Monitor machining sequence	FMS C.M.	DNC Link
30	Report on machining status	DNC Link	FMS C.M.
31	Feedback current status	CNC	DNC Link
32	Download programs	DNC Link	CNC
33	Download programs	DNC Link	Work Cell
34	Feedback current status	Work Cell	DNC Link

Note : C.M. = Control Module

3.4 Information Flow System for Specific Linking of Systems

As discussed in Section 2.4, computers running on various operating systems such as DOS/Windows, UNIX and Macintosh can be networked for file transfer using well developed commercial networking products. In addition, security of files e.g. restriction of access of certain files by unauthorized users, can also be achieved (Nance, 1992). However, this networking of computer systems using commercial networking products, or Networked Computer Systems (NCS), is inadequate to achieve the data exchange for a system shown in Fig. 3.2. NCS ensures the transfer of files among computers based on some handshaking rules (protocols) but it does not ensure the transfer of selected data such as the BOM or tool data, from one system to the other. Consequently a specially designed Information Flow System (IFS) is required, which ensures that the destination system can "read" the data. In other words, apart from the "physical" communication connection between systems, an efficient Information Flow System must also be able to deliver "selected" data and "reformatted" data to ensure "seamless" integration with the destination system. In the AutoCAD to IMPCON example, the AutoLISP programs are used for the purpose of "selecting" the BOM data and converting the format to suit the IMPCON system (IMPCON Manual, 1988).

3.5 Communication Link Components

As discussed in Section 2.4, a number of firms such as IBM, Microsoft, Novell and SUN Microsystems have developed networking products which can handle file transfer among various computing operating systems e.g. DOS/Windows, UNIX, OS/2. However, these networking products are not able to connect to CNC machines, robots and Automatic Guided Vehicles (AGVs) because most of these do not use any of the common operating systems such as DOS/Windows or UNIX. Instead, manufacturers of such equipment, e.g. Fanuc, have developed their own simplified "data interpretation/management system" for interfacing the electronic part of the equipment to the mechanical part. However, it should be noted that some of the latest "hi-tech" manufacturing equipment is equipped with the common operating system such as UNIX and DOS/Windows. For example, the Selective Laser Sintering Machine Model 2000 runs on UNIX while Digitbotic Non-Contact Laser

Scanning Machine and Laminated Object Manufacturing Machine Model LOMM 2030 run on Windows NT. But for those machines which do not support a common Operating System (OS), the following special communication links are necessary.

3.5.1 Communication Links for the Formation of a Full CIM System

As discussed in Section 2.1, the special communication links used in industry can be catorgorised into five types, (RS-series connection, Machine Interface Terminal, Secondary Adaptor, Host Adaptor and Network Interface Module) which can be used to connect different computerised manufacturing equipment such as robot, CNC machines, Automatic Guided Vehicle (AGV) etc. to a supervisory computer as shown in Fig. 3.3. The supervisory computer normally runs under UNIX or DOS/Windows platforms and therefore it can be connected to the common networking products such as Netware or Windows NT. With this networking capability, the supervisory computer can be connected to other subsystems, such as CAD, CAM and MRP-II, which support the common networking operating systems. With this approach, the manufacturing equipment such as robots, CNC machines, AGVs are integrated with other computer systems, such as CAD, CAM, MRP-II, indirectly using these special designed communication link products to form a full CIM system.

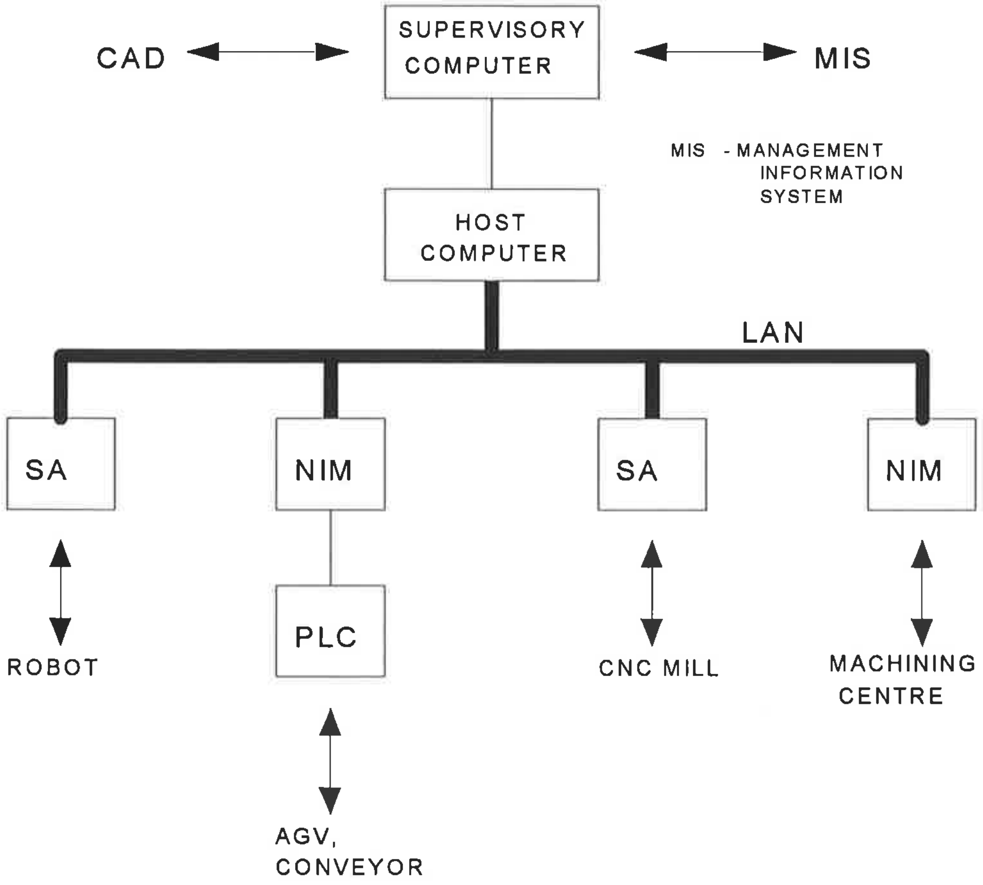


Figure 3.3 : Computer Communications via LAN

3.6 Summary

The project related to the link-up of AutoCAD and IMPCON for BOM transfer demonstrates the requirements for specific link among computer systems. The communication connection is more complex when more computer systems need to be linked together for data exchange. Essentially, there are two prerequisites for specific link of computer systems. The first is the communication link hardware and software for the specific equipment, and the second is related to accurate identification and reformatting of data. An efficient information flow system must be able to ensure that required data is transferred and “accepted” by the destination system.

Most of the traditional computerised manufacturing equipment do not support the common OS such as UNIX and DOS/Windows because they have their own computer system which interfaces with the mechanical parts of the equipment. This means that the commercial networking products such as Novell’s Netware cannot be used to connect them together. However, some industrial communication link components can be used to link this equipment to a supervisory computer which monitors the data communication processes among the computerised manufacturing equipment. Becasue supervisory computers run on common OS, they can be connected to a network and so connecting the computerised manufacturing equipment “indirectly” to other computer systems to form a full CIM system.

4. GENERIC LINKS

4.1 Introduction

In this thesis, the term generic link means the links between computers, for the purpose of information exchange, which are independent of the standards, hardwares, software and operating systems being used by those systems. In this Chapter, the development of a generic link between any number of systems is discussed. Firstly, the requirements for a generic link between CAD and MRP-II systems are examined and this is expanded to include other computer systems. From this analysis, a generic, cost-effective information exchange system, which can be used with any computer, operating system or package, is proposed. The components of this information flow system are the Generic Data Extraction System (GDES), the Generic DataBase Management System (GDBMS) and the Generic Format Translator (GFT). These systems, together with the methodology for their design and implementation are covered in this Chapter.

4.2 Generic Link of CAD and MRP-II Systems

As discussed in Chapter 3, the link between AutoCAD and MRP-II was accomplished with the development of specific data conversion programs to reformat the data from AutoCAD to suit the IMPCON system. This is a specific approach as the programs are designed for AutoCAD and IMPCON systems only. To overcome the problems when either system is changed, the neutral format approach was tested.

4.2.1 Neutral Data File

A neutral format is defined as the format which is independent of any specific system standards, and which acts as an "agent" to connect computer systems which cannot normally communicate with each other. It should be noted that neutral format is the "acceptable" and "consistent" format within an organisation or a group of organisations and the format may differ from other organisations which may set their own formats. In GNSIFS, the neutral format developed to achieve data exchange is unique and only supported by those programs developed based on the GNSIFS methodology (Section 4.6). In order to illustrate how a neutral format file is used by other systems to retrieve specific data, an example is taken from the SYSTEM.INI file, which contains all the Windows system settings and can be considered a neutral format file based on Windows standards. An extract of the SYSTEM.INI file is shown below :

Table 4.1 Extract of SYSTEM.INI file

```
------------------------------
[Network]
winnet=wfwnet/00025100
multinet=netware3
UserName=HENRY LAU
Workgroup=ATES
ComputerName=TOSHIBA_NBK
FileSharing=Yes
PrintSharing=Yes
LogonDisconnected=Yes
EnableSharing=Yes
Comment=USER
AutoLogon=Yes
StartMessaging=No
LoadNetDDE=Yes
LMLogon=0
Priority=200

[Drivers]
----------------------
----------------------
```

Once Windows is started, it looks for the SYSTEM.INI file and another file called WIN.INI for the start-up settings of the Windows environment. Since these files are in neutral formats, Windows knows where to find the required information. For example, if Windows needs to know what Network Operating System (NOS) is being used, it looks for the heading "[Network]" and then find the item "multinet". The text immediate after the "= " sign is the data needed and in this case it is "netware3". Since SYSTEM.INI file is in neutral format, the settings can still be used even if they are copied to another computer. Experienced system engineers with knowledge of the format can put data into SYSTEM.INI manually for a specific purpose e.g. change the name of Workgroup from the existing "ATES" (See Table 4.1) to "Computing" just by changing the text string. It should be noted that SYSTEM.INI is a text file with neutral format designed by Microsoft for the details of system settings. Nevertheless, this text file is important when a Non-Microsoft or Non-Windows programs need to know the Windows systems settings to ensure that the program can run smoothly. For example, a DOS program, such as AutoCAD, needs to know the type of drivers used in the existing computer so that the settings of the AutoCAD program can be adjusted to achieve optimum performance. The required information can be found under the heading of "Drivers" from the SYSTEM.INI file.

Another neutral format, used in manufacturing, is Product Data Exchange Specification (PDES) which provides geometric data related to a product. In order to explain how PDES works, an extract of a PDES file is as shown in Table 4.2.

Table 4.2 An Extract of PDES File

```
@1=CARTESIAN_THREE_COORDINATE(        0.00000,      0.00000,       0.00000);
@2=CARTESIAN_THREE_COORDINATE(        3.25000,      0.00000,       0.00000);
@3=LINE_SEGMENT(#1,#2);
@4=CARTESIAN_THREE_COORDINATE(        3.25000,      0.00000        0.00000);
@5=CARTESIAN_THREE_COORDINATE(        3.25000,      3.25000,       0.00000);
@6=LINE_SEGMENT(#4,#5);
@7=CARTESIAN_THREE_COORDINATE(        3.25000,      3.25000        0.00000);
@8=CARTESIAN_THREE_COORDINATE(        0.00000,      3.25000,       0.00000);
@9=LINE_SEGMENT(#7,#8)
@10=CARTESIAN_THREE_COORDINATE(       0.00000,      4.78000,       0.00000);
@11=CARTESIAN_THREE_COORDINATE(       0.00000,      0.00000,       0.00000;
@12=CARTESIAN_THREE_COORDINATE(       0.00000,      0.00000,       2.78000);
@13=CARTESIAN_THREE_COORDINATE(       10.2500,      0.00000,       0.00000);
@14=B_SPLINE_SURFACE(.F.,.F.,F.,.F.,2,2,#10,#11,#12,#13,3,3,3,3,3,3,1,1,1,1,1,1);
---------------------------------------------------------------------------------
---------------------------------------------------------------------------------
---------------------------------------------------------------------------------
```

The line numbers are specified at the beginning of each lines with the notation "@". Lines 1 & 2 indicate the position of the coordinates; line 3 shows that a line is drawn joining the points 1 & 2 specified by the coordinates in line 1 & 2. Similarly, lines 4 & 5 and 7 & 8 give the coordinates for the next two lines. Line 14 specifies the construction of the spline surface based on coordinates indicated in lines 10 to 13. A PDES file can be "read" by all computer systems because the geometric details of the "entities", such as line, circle etc, are in a "neutral" format and other systems know where to look for the required information. For example, a CAM package requires the geometric data regarding the spline surface for developing tool path for machining. Provided that this CAM package supports PDES format, the keyword "B_SPLINE_SURFACE" is searched from the file, which is found in line 14. Based on the data from that line, the geometric data can be obtained from lines 10 to 13 (#10, #11, #12, #13). Other data about the spline surface can be obtained from the text string stored within the brackets in line 14.

4.2.2 CAD Systems

A brief study (Zockel & Lau, 1990) of the features of CAD systems shows that alphanumeric data can be extracted from the CAD drawing in one of two ways:

- attributes which can be extracted from the drawing by built-in functions as long as they are designated as attributes before being selected by the built-in commands and
- application programmes/macros, which are special programmes written by the user to extract the necessary data and put it in the particular format for the receiving system.

The CAD systems investigated were AutoCAD (Version 9.0), CADDSMAN (Version 2.14) and CATIA (Version 3.0):

(a) AutoCAD - In AutoCAD attributes can be extracted from a drawing in any of three formats: (1) Blank Delimited Format (BDF), (2) Comma Delimited Format (CDF) and (3) Drawing Interchange File (DXF) which is AutoCAD's own format. Most of the common commercial packages such as Lotus, DBase accept BDF or CDF formats but not DXF (See Section 2.10).

AutoCAD also has a built-in language called AutoLISP which is not a system programming language such as Pascal or Fortran because many of the functions in the AutoLISP language can only be carried out by AutoCAD. However, by using some specific commands in AutoLISP, e.g. "getdist" which returns the value of the distance between two points from the drawing, the user is able to access the drawing files (.DWG files), extract the required data and put them into the desired format. However, with the latest release of AutoCAD Release 13, "C" programming language is chosen to replace AutoLISP although AutoCAD Release 13 still supports AutoLISP programs which have been widely used by AutoCAD programmers in the past years.

(b) CADDSMAN - CADDSMAN, which runs on PCs or Workstations, has a built-in function called **Tag** by which any alphanumeric information assigned to any entity can be identified. The maximum number of characters for a **Tag** name can be selected when the attributes or **Tag** file is created.

Tag file and **Tag** names can be recorded in a text file that can be printed in the graphics display area by using the **List Tag** and **List File** commands. However, a further program is necessary to reformat the information to make it suitable for another system.

Apart from the **Tag** functions, CADDSMAN also has a built-in language called Application Programming Language (APL). APL combines mathematical and graphical command routines which enable complex interactive manipulation of the CADDSMAN software database information. APL is a Pascal-like language with special features (CADDSMAN Reference Manual, 1989).

(c) CATIA - CATIA, which runs mostly on mainframe computers, has a powerful REPORTER function which is used to process the attributes of the component and to provide a parts lists. The user can define the format and content of a report by responding to the computer prompts. However, a simple conversion program may still be necessary for the conversion to other system.

CATIA has a built-in language called Interface User Access (IUA) which is designed to provide access to the attributes of elements in CATIA models. One of the main features of this IUA language is that it can run external programs written in Fortran (CATIA is written in Fortran) whilst it is still within the CATIA environment. This can complement the capabilities of IUA since a Fortran program can be written to extract the required data and arrange it in the necessary format (CATIA Reference Manual, 1988).

4.2.3 MRP-II Systems

The type and format of information required by MRP-II systems vary substantially, as do the field sizes for different parts (Table 4.3). Some of the information, such as schedules, costs and capacity, are generated on the system by direct input while information such as bill of materials, inventory status, work in progress and order status are generated on other systems.

Table 4.3 Field Sizes of MRP-II Systems

Field	IMPCON	MENTAT	BPCS
Part code	14	14	15
Description	30	40	30
Part type	1	1	1
Unit of measure	6	2	2
Std unit cost	7.4	6.3	9.4

(a) IMPCON - IMPCON expects information from the keyboard in a specified form in response to specific prompts, i.e. the number of characters as well as the alphanumeric form is critical. Hence a special program has to be written to take a string of selected information and place it into the appropriate fields. Another program would have to be written to reverse the procedure.

(b) BPCS - BPCS has a CAD interface module which allows the transfer of BOM data from PC-based CAD packages to BPCS in DXF format (BPCS, 1988). As DXF format is AutoCAD's own format, which is not available in most other CAD systems, this method is somewhat restrictive. To overcome this problem it is possible to either write a program to convert the BOM data to DXF format or to convert the data from the IGES file to DXF format. Most CAD packages support IGES and commercial conversion programmes are available to convert IGES to DXF.

(c) MENTAT - MENTAT is the MRP-II package marketed by Hewlett-Packard. It has similar capabilities to BPCS in that it has software which enables the user to select text strings on the screen from a text file and then put them into the databases of MENTAT. The benefit of this programme is that the data does not have to be in a fixed format while the disadvantage is that the data must be selected one by one (MENTAT/MFG, 1988).

It would not be difficult to modify the program to pick the required data automatically and rearrange it in a neutral format. This would ensure an efficient and accurate transfer of data.

4.2.4 Establishment of Generic Link between CAD & MRP-II Systems

From the above discussion it is clear that the exchange of information between systems in most cases means that special programs have to be written for each combination of systems unless a generic link is developed (Fig. 4.1). The data is first extracted from the database of CAD and put in an ASCII file which is converted to another ASCII file with a neutral format. This file with neutral format must be converted to a file with the MRP-II system format before it can be stored in the database of the MRP-II system. As long as all conversions go via the neutral format, any CAD system can communicate with any MRP-II system.

A two way process can be achieved with the same way (Fig. 4.2) as long as the neutral file is supported by both sides.

4.2.5 Generic Link of Systems to Form a Complete CIM

The generic link for CAD and MRP-II systems could be expanded to include other manufacturing subsystems to achieve data transmission among all the subsystems to form a complete CIM. However, a more logical strategy is to set up a common database, as is required in CIM, with a Data Base Management System (DBMS) and a standard language such as Structured Query Language (SQL). As shown in Fig. 1.2, the data sent to the

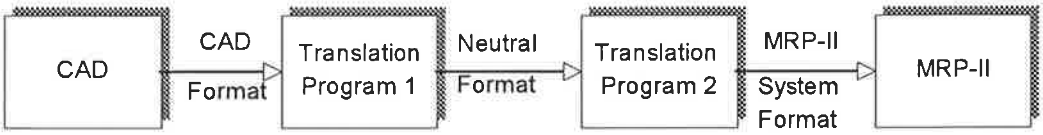


Figure 4.1 : Generic Link of CAD & MRP-II Systems

common database is converted to a neutral format and the data sent to a particular system from the database is converted from the neutral format to the system format. In this way each computer system only has to have one translation program, which includes the translations from system format to neutral format and from neutral format to system format, to handle the input/output of information since it only communicates with the database.

The next problem is the cost of the database which is heavily dependent of the amount and type of information which must be stored. It is proposed that geometric information can all be held in the CAD system and only viewed at other locations. This would eliminate the need for geometric information in the database, reducing its complexity and memory requirement. As discussed in Section 1.2.2, only "selected" alphanumeric data is stored in common database so that data can be accessed more easily. Having decided that the data to be stored in the database is only selected alphanumeric information, it is further possible to divide the information into two groups; Group A dealing with fixed format information such as costs, process times and machinery availability while Group B contains information relating to the products. The reason for dividing the data into two groups is that as group A belongs to fixed format data, it can be handled by a relational DBMS which can be developed easily while Group B data needs to be handled by a specially-designed DBMS (Details covered in Section 4.5) .

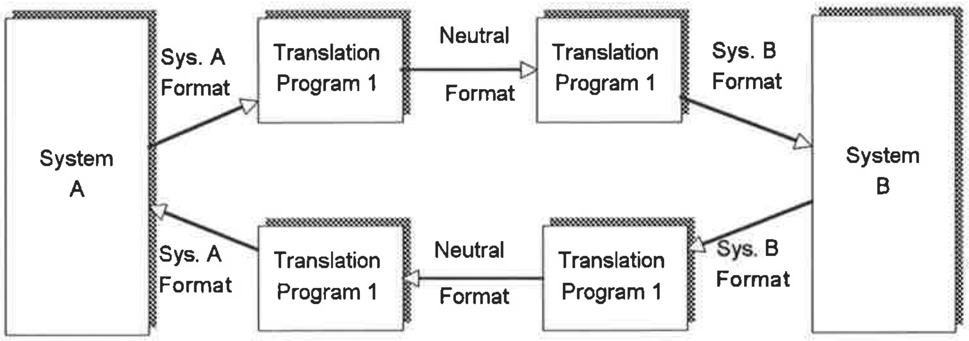


Figure 4.2 : Generic Link Between Two Systems

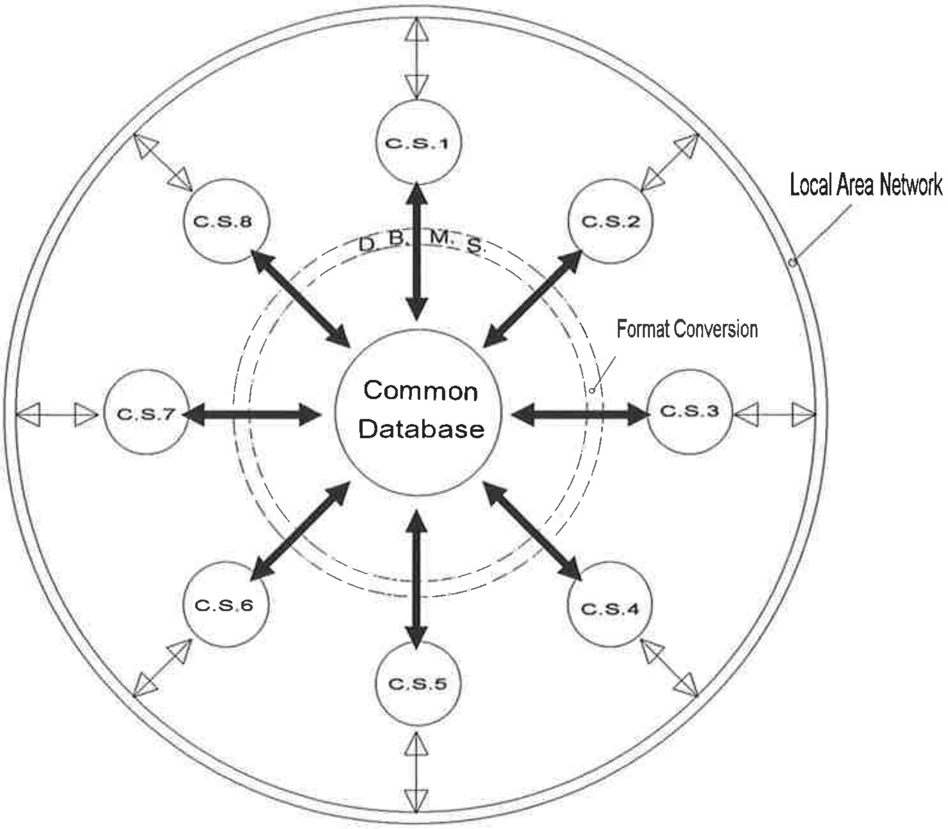
As explained in Section 2.7, the neutral data file used in this generic link can use the established standards, such as PDES, or the self-designed formats such as the one designed by Motorola (Millet, 1990). Once the conversion programs from system formats to neutral formats for various computer systems are developed, this generic system can deal with any future change of computer systems in a more flexible way. For example, the change from AutoCAD to CADDSMAN just requires the development of two conversion programs; one for the conversion of CADDSMAN system datafiles to neutral format files and the other for the conversion of neutral format files to CADDSMAN system format. However, the conversion programs, which are the essential components of this type of generic link, are costly to produce. A more cost effective information flow system, which is able to perform format translation “transparently”, is proposed in the following context.

4.3 Proposed Generic Non Standard-specific Information Flow System (GNSIFS)

Fig. 4.3 is a diagrammatic representation of the proposed information flow system. The generic system operates within the shell of a DataBase Management System (DBMS) which acts like a computer operating system and controls the data storage, data retrieval, data search and other data manipulation operations, and enables the efficient linking of the various computer systems for data transfer.

The alphanumeric data which has been identified for common use is firstly selected and then extracted from datafiles belonging to various computer systems e.g. C.S.1, C.S.3 etc. The format of the datafiles is immaterial because the user selects the data to be transferred from existing ASCII files and the data is converted to a neutral format before being sent to the common database. The main feature of the neutral format is that it is "readable" by all other computer systems. For example, if the Bill Of Material (BOM) is extracted from a CAD system datafile and put in a neutral format, other system, e.g. MRP-II, is able to read the data and, with the help of a data retrieval subroutine, it can retrieve all or part of the information from the file. The common database stores the data (Group B data) as a string for each manufactured component (machine base, bearing bush etc.) and the string contains information relating to a component such as part name, material, process information, manufacturing cost data etc. Because the relevant data of a component is grouped together, the time for data searching is reduced which makes the retrieval of the more efficient. The information can be displayed on a screen to enable others to view the relevant data of a component.

Apart from component data (Group B data), the common database also stores fixed format data (Group A data) such as store inventory, labour costs, machine availability and overhead rates etc. As mentioned in the previous context, fixed format data is handled by a relational DBMS. The details about the development of this DBMS is covered in Section 4.5.



D.B.M.S. -> DataBase Management System

C.S. -> Computer System

Figure 4.3 : Diagrammatic Representation of GNSIFS

4.3.1 Specification of Input and Output Information

Before the proposed Generic Non Standard-specific Information Flow System (GNSIFS) can be used, the personnel of the company have to specify the input and output data items required for data exchange within their company. A summary of likely items are shown in Table 4.4 (and Fig. 3.2 in graphical form) required for data exchange. For instance, does the BOM need to be transferred from AutoCAD to IMPCON or does the designer working at the CAD system need the tool data from the CAM system ?

Table 4.4 Input and Output Information for Various Systems

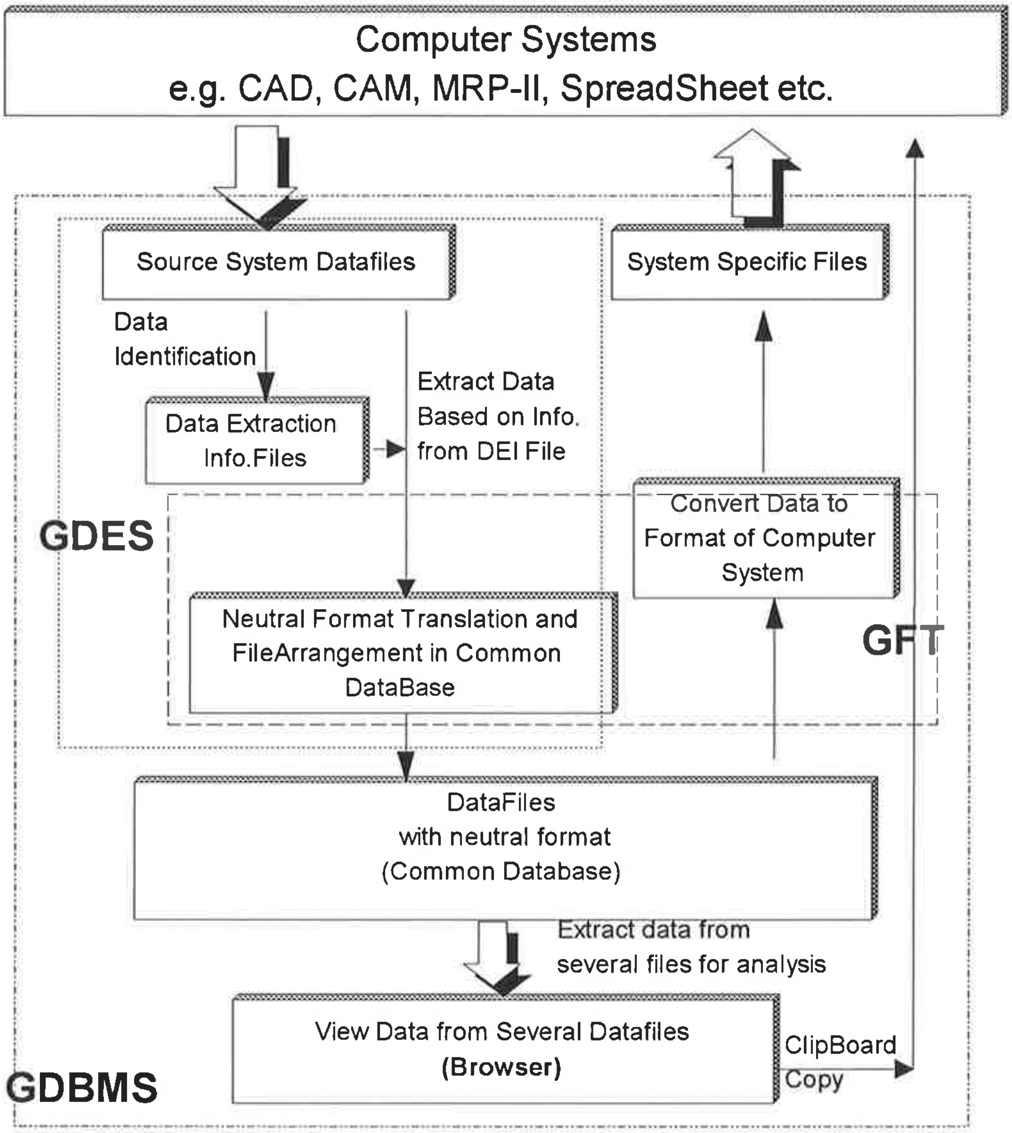
COMPUTER SYSTEM	INPUT	OUTPUT
MRP-II	bill of material, production status feedback, inventory situation,	actual production data, , inventory data, cost info., product data
CAD	component stock data, cost info. design specifications, material info.	bill of material, parts list, quantity, geometric data
CAM	dimensional data, process data, NC programming data	CAPP and FMS control centre, G codes data, tooling list, tool type, tool size
CAPP	product info., cost data, geometric data, tools data, machine availability data	routing, process operations info., process data
ATE	quality analysis data, areas to be checked, checking algorithms data	quality checking results, overall quality analysis
STORE/ INVENTORY	plan of incoming goods, storage and material handling monitoring	inventory situation, forecast utilisation of storage area
FMS Control Centre	process operations info., quality checking results, numerical control data	machining programmes, monitoring machining data

When the data items required for data exchange have been determined, the user can identify the alphanumeric information directly from the data files. For example, if a BOM is required for data exchange, the AutoCAD attribute text file, which contains the BOM and other data, will be called up by GNSIFS. The user then marks the BOM data from the attribute text file and the GNSIFS is then able to perform data exchange. BOM data is transferred to a common database in a neutral format which can then be retrieved from the database by other computer system with the data format converted to the required system format. All these data movement operations are monitored and controlled by the DBMS. In practice, the format conversion is done “transparently” without the knowledge of users.

The functional structure of the proposed GNSIFS (shown as in Fig. 4.4) consists of three main components which are Generic Data Extraction System (GDES), Generic DataBase Management System (GDBMS) and Generic Format Translator (GFT). The details of these three systems are covered in the following sections.

4.4 Generic Data Extraction System

The main function of GDES is the extraction of data from the system datafiles and transfer them to the component-related datafiles in the common database. It is generic in the sense that it allows the user to specify the data to be extracted directly from the system datafiles without the need to take into consideration the formats used in the datafiles.


Computer Systems
e.g. CAD, CAM, MRP-II, SpreadSheet etc.
Source System Datafiles
System Specific Files
Data Identification
Data Extraction Info.Files
Extract Data Based on Info. from DEI File
GDES
Convert Data to Format of Computer System
GFT
Neutral Format Translation and FileArrangement in Common DataBase
DataFiles with neutral format (Common Database)
Extract data from several files for analysis
View Data from Several Datafiles (Browser)
ClipBoard Copy
GDBMS

GDBMS - Generic DataBase Managment System
GFT - Generic Format Translator
GDES - Generic Data Extraction System

Figure 4.4 : Functional Structure of GNSIFS

4.4.1 Types of Files Used in Data Extraction Process

In order to have a clear understanding of how the data extraction process works, the three types of files (Fig. 4.5), which are the main files used in the process, are described as below :

(a) System datafile - This is the text file generated by the source computer system and it contains setup information and technical information related to the source computer system. One example is the datafile generated by the computer system of the Sintering Station Model 2000 (Table 6.2) when a new component is produced by this machine. This datafile contains the technical information related to the production of the component, such as temperatures at various stages, and the dimension tolerances. This datafile can be quite large and in the case of the system datafile for the Sintering Station Model 2000, the datafile can be up to the size of 20 A4 pages. In practice, only a few lines of relevant data is required, such as the warm-up temperature and dimension tolerances, and therefore it is necessary to extract only the relevant data from this system datafile.

(b) Data Extraction Information File (DEI File) - This file contains information to determine the exact location and length of the text to be extracted and is created when user identifies the text to be extracted from the source system datafile. An example of this file is shown in Fig. 4.9 which shows three text-lines identifying three pieces of information to be extracted. In the first line, “x-coordinate of Part-1” is the label (a meaningful name) given to the data,

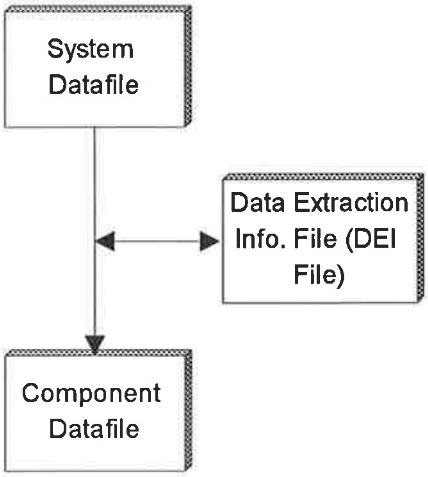


Figure 4.5 Files Used in Data Extraction Process

"15" is the Type of Extraction (explained in Section 4.4.2.4) which indicates the group of "check-boxes" (boxes in a dialogue box for user to select as desired) specified when user identifies the text, "part-1-min_bounding_box" is the Reference Text (Section 4.4.2) in the source system datafile and is essential to "pinpoint" the location of the data to be extracted, the others are the parameters to ensure the exact locations of the data to be extracted. With this information, the exact text to be extracted can be identified correctly.

(c) Component-related Datafile (CD) - The is the file storing data extracted from the source system datafile based on the information provided by DEI File (Fig. 4.5). The characteristic of this file is that most of the data is related to a particular manufactured component (Group B data) such as tool data, quality data or cost data etc. It has the file extension of "dfl" (.dfl) e.g. bearing.dfl, for easy identification. This file can be regarded as the "destination" file since the extracted data is finally stored in this file. Group A data does not require CD because the lengths of data strings (record lengths) for this group are fixed and a relational DBMS developed from Access or DBase can handle this data easily (Section 4.5).

4.4.2 Logical Sequence of Data Extraction Process

The sequence of the data extraction process is :

(a) Identify data in system datafiles

(b) Record data extraction information in DEI file

(c) Specify files for data extraction process (the three files as shown in Fig. 4.5)

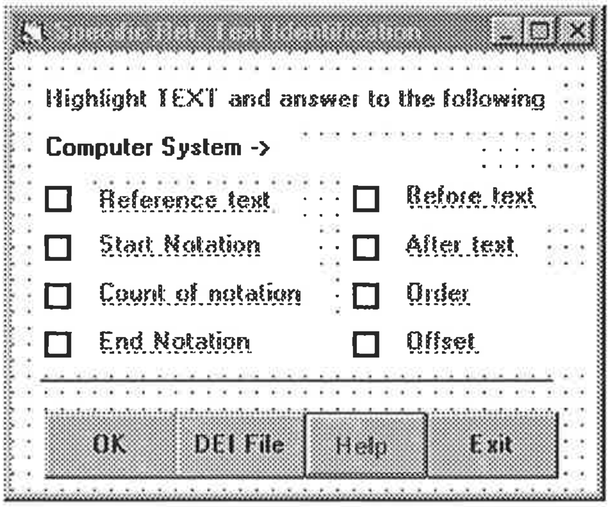
(d) Extract data using DEI file

(e) Store extracted data in common database

4.4.2.1 Identify Data in System Datafiles

Data identification is mainly concerned with the formation of DEI file which records the user's selections when identifying the text in the system datafile directly from the computer screen. In the data identification process the user must specify the system datafile (source) which contains data and the component-related datafile (destination) which is located in the common database. For example, if the Bill of Material (BOM) of a manufactured component named "bearing" is to be stored in the common database, the system datafile which contains BOM of that manufactured component needs to be specified. Assume that in this case AutoCAD is the CAD system and a file called "bearing.txt", which contains the BOM of the component "bearing", has been created using the **Attribute** function. The user "opens" the "bearing.txt" file and because the data is alphanumeric the user can read the data. The required data is identified and the user can assign a label name to the data. This label name is useful as it gives some indication about the function and meaning of the data. A data identification dialogue box (a form on the screen to allow the user to input data in response to the messages, check boxes or questions displayed on it) is provided by GDES to allow the user to specify the exact position of text to be extracted (Fig. 4.6). There are eight check boxes in this dialogue box the functions of which are :

(1) Reference Text - The specification of the Reference Text is an essential step in data identification. This specified Reference Text must be present in the system datafiles for all the components otherwise the data identification process will



Highlight TEXT and answer to the following

Computer System ->

□ Reference text □ Before text

□ Start Notation □ After text

□ Count of notation □ Order

□ End Notation □ Offset

Figure 4.6 : Dialogue Box for Data Identification

fail. In a system datafile, the data are either system values or reference text. Table 4.9 illustrates a portion of a system datafile extracted from the computer system of a Rapid Prototyping Machine. It can be seen from the content of the file that each line contains a Reference Text followed by the system values. For example, the Reference Text of the third line of Table 4.9 is "part-1-min_bounding_box" and the associated values are "-115.316000 -11226.11 0.01600". This whole line means that the minimum boundary of the box which holds part number one has the x,y and z coordinates of "-5.316000", "-11226.11" and "0.01600" respectively. The first check box of the dialogue box allows the user to identify the Reference Text directly from the system datafile by highlighting the Reference Text. However, since the Reference Text, e.g. "part-1-min_bounding_box", may not have any meaning for non-technical personnels, the system user can assign other label names, e.g. "boundary coordinates of part 1", which is more meaningful when it appears on a spreadsheet or a table.

(2) Start Notation - In the previous example, there are three separate values associated with the Reference Text. In fact, it is common that there are more than one system value associated with the Reference Text. For example, there may be more than 20 values associated with the Reference Text "size of cutting tools" in a SmartCAM system datafile. In order to identify which one of the associated values is required, a technique has been devised which relates the

required system value to its "notation sign". In the third line of Table 4.9, there is a blank space separating the Reference Text and the first associated value i.e. “-115.316000”. In other computer systems this notation can be a comma or a slash (backward or forward). In fact, as shown in Fig. 4.7, the user can select any of the four notation signs, characters or an “end of line” or can select any “other” convenient character. The user identifies the "starting notation sign" which in this examlple are the same for all three values i.e. a blank space. Therefore another checkbox selections (“Count of Notation” checkbox) is needed to specify exactly which one of the associated values is to be extracted.

(3) Count of Notation - In the third line of Table 4.9, the system value "-115.316000" occurs after the FIRST space after the Reference Text "part-1-min_bounding_box" whereas the value "0.01600" is after the THIRD space. Therefore, the counts of notation for "-115.316000" and "0.01600" are 1 and 3 respectively.

(4) End Notation - The same procedure as that for the “start” notation is used for the “end” notation. The “end” of the first value "-115.31600" is a space and that for the third value "0.01600" is an "end of line". When all the selections have been made, the information is stored in DEI File after the **OK** button is clicked.

(5) Before Text/After Text - In some system datafiles, the same Reference Text may appear more than once in different sections of the file. For example, a CAM system datafile may use the Reference Text "Cutting conditions" or "Cutting speed limits" in headings for several cutting tools. This may cause confusion during data extraction process. Consequently, a "Before Text" and "After Text" feature has been created to allow the user to specify value either before or after the Reference Text. For example in the SYSTEM.INI file extract (Table 4.1), the "EnablingSharing" has the associated value YES. This specifies that sharing of files is allowed in the Workgroup connection network. However, the Reference Text "EnablingSharing" may appear in other headings such as "Network 1". In order to specify that it is for the Workgroup users, the "Before Text" and "After Text" parameters are specified as DRIVERS and NETWORK respectively. This ensures that the Reference Text is after the NETWORK heading and before the DRIVERS heading.

(6) Order - Another way to specify which Reference Text is relevant for desired system value is to specify the "order" of occurrence. If the order is one, it means it is the first occurrence of the specified reference text. If order is specified as two, it means it is the second occurrence and so on. If there is any confusion when selecting the correct Reference Text, this parameter is disabled and the "Before Text/After Text" parameter alone is used to locate the exact position of the Reference Text.

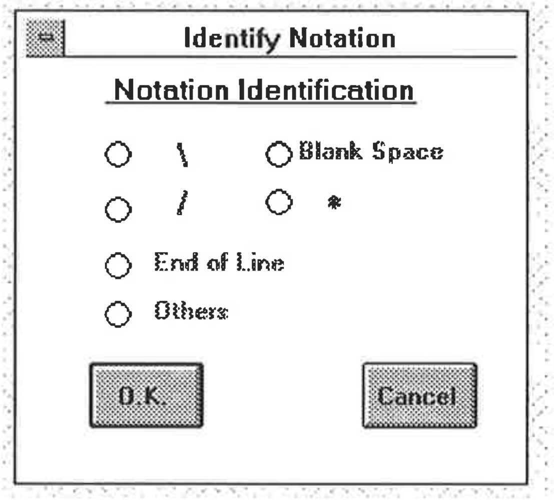


Figure 4.7 : Select Notation Signs

(7) Offset - Yet another way to specify the location of the value relative to the Reference Text is the number of characters between the "last" of the Reference Text and the "first" of the value. For example the value "-115.31600" is 2 spaces away from the Reference Text and the second value "-11226.11" is 14 character spaces away from the same Reference Text. These are called offset values. This item is useful for those system datafiles which present the associated values in a tabular form such as SmartCAM datafiles where the tool data is in tabular form.

4.4.2.1.1 Variable Reference Text

The above data identification process cannot be used when the data is arranged in the form of a part-list as shown in Fig. 4.8 because the Reference Texts for the component "Base Assembly" are the part codes i.e."1739", "23456", "2341", "678" while the Reference Texts for component “Top Assembly” are also the part codes with different values i.e.“2452”, “17456”, “2551”, “8898”. It can be seen that part codes of one component are different from those of others and therefore they cannot be used as the Reference Texts.

This problem has been overcome by defining two levels of Reference Text.. As shown in Fig. 4.8, the part codes of the two component i.e. Base Assembly and Top Assembly are different but the **Text String** “Part List” remains unchanged for all the component part-lists and therefore it can be chosen as the Reference Text. From this unchanged Reference Text, i.e. “Part List”, the part codes i.e. “1739”, “23456”, “2341” and “678” can be identified as these part codes are now the associate values of the Reference Text “Part List”. Once these part codes are identified, they can be used as “second level” Reference Texts for the identification of their associated values e.g. “Bush”, “10” and “Brass” are now the associate values of the “second level” Reference Text which is “1739”. This means that instead of one DEI file, there are two DEI Files created, one for the identification of part codes with “Part List” as Reference Text, the other for identification of the associated values of the part codes which are now the “second level” Reference Texts.. This is referred to “Variable Reference Text Data Identification” because the Reference Text i.e. the part codes change from component to component.

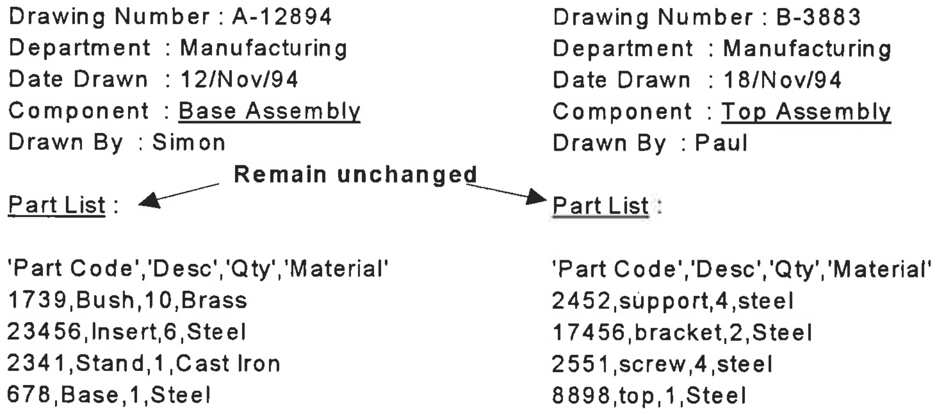
The steps for the data identification with variable Reference Texts are summarised below :

(1) Specify "Part List" as Reference Text, identify the part codes as associated values of this Reference Text with the eight check boxes as shown on the data identification dialogue box (Fig. 4.6).

(2) The first DEI File is created which can be used for the extraction of part codes at a later stage.

(3) Specify the existing part codes (i.e.1739 ......)as Reference Texts and identify the Description, Qty and Material as associated values.

(4) The second DEI file is created but linked to the first so that it can use the extracted values from the first DEI file as Reference Texts for the identification of data in the part-list such as Description, Qty and Material.



Remain unchanged

Drawing Number : A-12894
Department : Manufacturing
Date Drawn : 12/Nov/94
Component : Base Assembly
Drawn By : Simon

Part List :

'Part Code','Desc','Qty','Material'
1739,Bush,10,Brass
23456,Insert,6,Steel
2341,Stand,1,Cast Iron
678,Base,1,Steel

Drawing Number : B-3883
Department : Manufacturing
Date Drawn : 18/Nov/94
Component : Top Assembly
Drawn By : Paul

Part List :

'Part Code','Desc','Qty','Material'
2452,support,4,steel
17456,bracket,2,Steel
2551,screw,4,steel
8898,top,1,Steel

Figure 4.8 : Part-list Data for Two Components

4.4.2.2 Record Data Extraction Information in DEI file

It is important that the input and response by the user during the process of data identification is recorded in a file so that the data recorded can be used in the subsequent data extraction process. For example, if the three associated values of the third line of Table 4.9 are to be extracted, 3 label names must be assigned to these values. The first value i.e. "-115.316000" is labelled "x-coordinate of Part-1"; the second value i.e. "-11226.11" is labelled "y-coordinate of Part-1" and the third value i.e. "0.01600" is labelled "z-coordinate of Part-1".

Table 4.5 shows the users input from the data identification dialogue box for "x-coordinate of Part-1"

Table 4.5 User Input for "x-coordinate of Part-1"

Check Box Title	Check Box No.	User Input
Reference Text	1	part-1-min_bounding_box
Start Notation	2	space i.e. Chr$(32)
Count of Notation	3	1
End Notation	4	space i.e. Chr$(32)

It should be noted that, as shown in Fig. 4.6, the label name is not included in the list of check boxes as the dialogue box is mainly concerned with the exact positions of relevant data to be extracted. Instead the user is prompted to enter the label name when the **OK** button of the dialogue box is pressed. In this case, the label name of the first associated value is assigned as "x-coordinate of Part-1".

The user's inputs for the second associated value are shown in Table 4.6 :

Table 4.6 User Input for "y-coordinate of Part-1"

Check Box Title	**Check Box No.**	**User Input**
Reference Text	1	part-1-min_bounding_box
Start Notation	2	space i.e. Chr$(32)
Count of Notation	3	2
End Notation	4	space i.e. Chr$(32)

The label name of the second associated value is assigned as "y-coordinate of Part-1".

For the third associated value, the user's inputs are as shown in Table 4.7 :

Table 4.7 User Input for "z-coordinate of Part-1"

Check Box Title	**Check Box No.**	**User Input**
Reference Text	1	part-1-min_bounding_box
Start Notation	2	space i.e. Chr$(32)
Count of Notation	3	3
End Notation	4	End of Line i.e. Chr$(13)

The label name of the third associated value is assigned as "z-coordinate of Part-1".

It is important that the DEI file has a record of all the check boxes selected by the user. The technique developed for this purpose is to assign a binary code to each box in Table 4.8 and to sum them.

Table 4.8 Binary Codes for Selections

Check Box Title	Check Box Number	Binary Code
Reference Text	1	1
Start Notation	2	2
Count of Notation	3	4
End Notation	4	8
Before Text	5	16
After Text	6	32
Order	7	64
Offset	8	128

For example, if check boxes of numbers 1,2,3,4 are selected, the sum of binary codes for these boxes is 15 (1+2+4+8). If check boxes of numbers 1,2,3,5 are selected, the sum of binary codes for these boxes is 23 (1+2+4+16). This number, "15" or "23", is called the "Type of Extraction" and it indicates which check boxes have been selected and once the Type of Extraction is specified, all the parameters required for data extraction, are known.

Table 4.9 An Extract of System Datafile

```
part-1-flags DOBOVI binary millimeters
part-1-offset 0.111111 0.228 5.335
part-1-min_bounding_box -115.316000 -11226.11 0.01600
part-1-max_bounding_box 115.316000 1166.230400 32.639000
part-2-STL_filename /usr2/sls2k/geometry/stl/11-12-93/multi_cap.11
part-1-part_name multi-cap.bo
part-1-fill_laser_power 0.000145 22.00023
part-1-fill_scan_count 0.000000 2
part-1-slicer_fill_first 0.000000 1
part-1-STL_filename /usr2/sls2k/geometry/stl/11-12-93/multi_insert.11
part-1-sorted_part_name /usr2/sls2k/geometry/stl/bld11-12-99/multi_cap.10
part-3-STL_filename /usr2/sls2k/geometry/stl/11-12-93/multi_link.11
part-1-outline_laser_power 0.000131 22.00024
part-1-outline_scan_count 0.00014 22.000225
```

In the previous example, concerned with the extraction of the three associated values in the third line of Table 4.9, the Type of Extraction is 15 i.e.(1+2+4+8). The data stored in the DEI file for the extraction of these 3 values is shown in Fig. 4.9. In the first line, the string "x-coordinate of Part-1" is the label name defined by the user; "15" is the Type of Extraction; "part-1-min_bounding_box" is the Reference Text; "chr$(32)" is the space specified in Start Notation check box; "1" is the "Count of Notation"; "Chr$(32)" is the space specified for End Notation. The second and third lines are the same except for the Count of Notation which are "2" and "3" respectively and the End Notation is a "carriage return" i.e. notation "Chr$(13)" instead of a space.

The procedure for data extraction with variable Reference Texts (described in Section 4.4.2.1.1) is as follows :

(1) From the information in the first DEI File, which takes "Part List" as the Reference Text, extract the four part codes i.e. "1739", "23456", "2341" and "678".

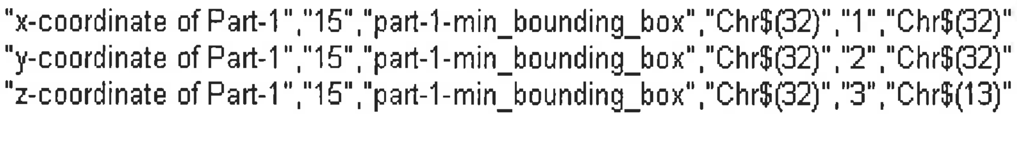
(2) Store the values of the part codes in a temporary location. These values will be used as the "second level" Reference Texts for the extraction of data related to part codes.

(3) From the information in the second DEI file, extract the associate values for each Reference Texts, e.g. for Reference Text "1739", the values are "Bush, 10, Brass" and so on until all the required information is specified.

(4) DEI File closes automatically with a "job completion" message displayed on the screen when all the information is utilised.

4.4.2.3 Specify Files for Data Extraction Process

The previous sections (4.4.2.1 and 4.4.2.2) are concerned with data identification from system datafiles and how the Reference Text, Type of Extraction and relevant parameters are stored in the DEI file. As shown in Figure 4.5, the three files needed for the data extraction process are the system datafile, the DEI file and the component-related datafile. The form shown in Fig. 4.10 is designed for the selection



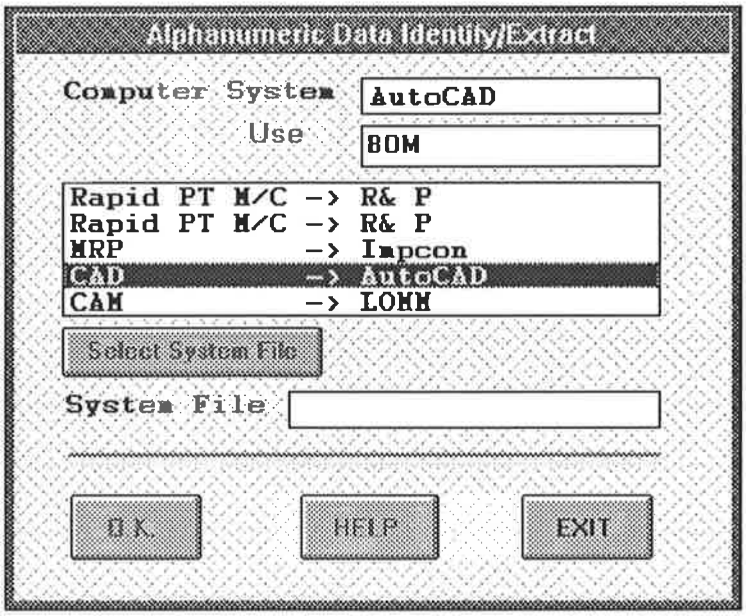
```
"x-coordinate of Part-1","15","part-1-min_bounding_box","Chr$(32)","1","Chr$(32)"
"y-coordinate of Part-1","15","part-1-min_bounding_box","Chr$(32)","2","Chr$(32)"
"z-coordinate of Part-1","15","part-1-min_bounding_box","Chr$(32)","3","Chr$(13)"
```

Figure 4.9 : An Extract of DEI File

of the relevant files for data extraction and the steps are :

(1) The user selects the computer package from the list box which contains all the existing computer packages. If there are more than one computer packages under one type of computer system, there will be more than one DEI file for that computer system. For example, under the computer system Computer Aided Design (CAD), several packages can exist, such as AutoCAD, CADDSMAN and QuikDraw, and each requires different DEI files. However, there may be more than one DEI Files for one package. For example, in AutoCAD, there may be one DEI File for "Extraction of BOM" and the other DEI File for "Extracting key dimensions" The **USE** textbox is designed to indicate the specific purpose of using the extracted data whether it is for "extracting tool data" or "extracting technical setting data". In Fig. 4.10, the **USE** texbox has the input text "BOM", which means the DEI File is used to extract the BOM data from a CAD drawing and in this case it is an AutoCAD drawing. Therefore, the DEI File is linked to the computer package e.g. AutoCAD and the USE e.g. "BOM" which means "for extraction of BOM data".

(2) Click on **Select System File** to select the source system datafile which contains the source data to be extracted.



Computer System AutoCAD

Use BOM

```
Rapid PT M/C  -> R& P
Rapid PT M/C  -> R& P
MRP           -> Impcon
CAD           -> AutoCAD
CAM           -> LOMM
```

System File

Figure 4.10 : Data Identify/Extract Form

(3) With the DEI File and source system datafile specified, data extraction can start. After data extraction, the user then specifies the component-related datafile which stores the extracted data (covered in Section 5.3).

4.4.2.4 Extract Data Using DEI File

The actual data extraction is carried out with the information in the DEI File. For example, if the user would like to transfer the three associated values from the system datafile of "R & P" (Rapid Prototyping Machine) computer system, e.g. "c:\data\r&p.txt", to the component-related file, e.g. "c:\data\shafts.dfl", the steps are as follows :

(a) Determine the Type of Extraction which in this case is "15".

(b) Analyse the binary code and determine the exact group of check boxes involved. "15" is the summation of "1", "2", "4", "8" so that boxes "1", "2", "3" and "4" must have information.

(c) Read the Reference Text and characters based on check boxes involved. In this example, the check boxes involved are "Reference Text", "Start Notation", "Count of Notation", "End Notation". According to the DEI file in Fig. 4.9, the "Reference Text" is "part-1-min_bouning_box", the character for "Start Notation" is "Chr$(32)" which is a space, "Count of Notation" is "1" and "End Notation" is "Chr$(32)" which is also a space. This means that the data to be extracted starts after the FIRST SPACE with reference to the text "part-1-min_bounding_box" and ends before the next "SPACE". A subroutine, written specifically for this Type of Extraction, is then used to extract the data (-115.316000) from the file "c:\data\r&p.txt". Obviously different subroutines are required for different Types of Extractions.

4.4.2.5 Put the Extracted Data into the Component-related Datafile

Having extracted the data it must be transferred to the component-related datafile which has the extension "**.**dfl", e.g. "c:\data\shafts.dfl". The function **FileCopy** provided by Visual Basic can be used to copy data from the Source file to the Destination file i.e. "c:\data\shafts.dfl". Most programming tools e.g Basic, "C" have this file-copying function provided though with different function (command) names. However, the conversion of the copied text to the neutral format requires a neutral format translator, which is a program written in Visual Basic (Microsoft Corporation, 1993a,b,c,d), and is covered in Section 4.6. An extract of the file "c:\data\shaft.dfl" is shown in Table 4.10.

Table 4.10 An Extract of Component-related Datafile

```
### Rapid PT M/C -> R & P -> For Technical Data
"x-coordinate of Part-1","-115.316000"
"y-coordinate of Part-1","-11226.11"
"z-coordinate of Part-1","0.01600"
```

4.4.3 Methodology of Generic Data Extraction

While the data extraction process, using the information provided by DEI file, is discussed in Section 4.4.2.4, the following explains the methodology for the necessary subroutines (illustrated in Fig. 4.11).

1. Each line of the DEI file provides information for the extraction of one single piece of information. For example, the first line of DEI file (Fig. 4.9), is used to extract the first associated value of Reference Text "part-1-min_bounding_box" , which is "-115.316000". The second line and third line are for the second and third values respectively. Each line contains four groups of data viz "Label Name", "Type of Extraction", "Reference Text", and "Parameters". In a subroutine, the extraction of this line can be achieved by one line of code. For example,

in Visual Basic the following statement extracts one line of text from a file (#1) to a variable named TextLine (Microsoft Corporation, 1993).

Line Input #1, TextLine

2. The data string extracted from the DEI file needs to be separated in order to identify the four groups of data, and then stored in an ARRAY in the same order as in the DEI file. Once the retrieved data is stored, the specific group of data is available by referring to the index of that group of data in the ARRAY. It should be noted that the data in the retrieved text line is separated by comma (comma delimited format). As shown in the pseudocodes of the subroutine (see below), a command FIND is used to locate the position of comma in the text line and a "Do While" loop is used to ensure that all the data in the text line is retrieved before ending.

The pseudocodes, which are independent of the syntax (grammer) and semantic (wordings) of <u>any</u> particular programming language and are therefore a generic way to demonstrate the structure of a subroutine, are listed below :

```
Rem : Subroutine - Retrieve Text from DEI File
Rem : Initialize the First Comma Position
      N = 1
Rem : The Following Do Loop Will Continue
Rem : Until the End of Line is Encountered
Do While Not End of Line
      Find "N"th Position of Comma in Line
      Retrieve Text before Comma
      Put Retrieved Text to "Nth" element of Array
      Rem : Increment to the next comma
      N = N + 1
End Do
```

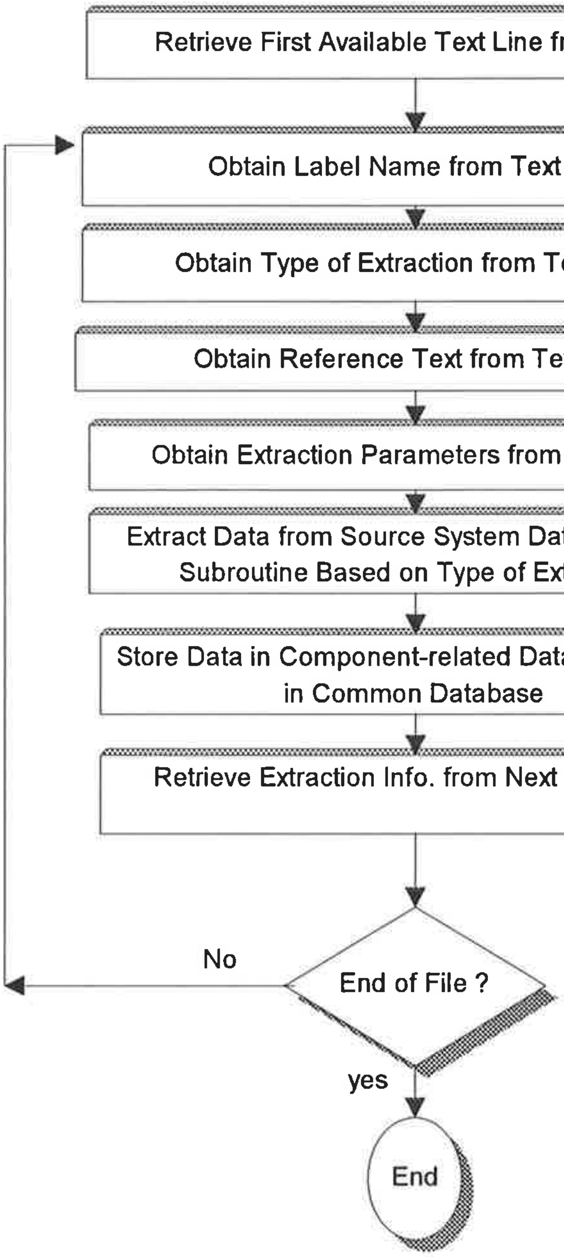

Retrieve First Available Text Line from DEIF
Obtain Label Name from Text Line
Obtain Type of Extraction from Text Line
Obtain Reference Text from TextLine
Obtain Extraction Parameters from Text Line
Extract Data from Source System Datafile Using Subroutine Based on Type of Extraction
Store Data in Component-related Datafile located in Common Database
Retrieve Extraction Info. from Next Text Line
No
End of File ?
yes
End

Figure 4.11 : Data Extraction Using DEI File

3. As explained in Section 4.4.2, the Type of Extraction determines the group of Notation parameters used for the extraction of a particular piece of information. There are eight possible selections as shown in the data identification dialogue box (Fig. 4.6), and so in theory, there are 256 (2 to the power 8) combinations. The subroutines vary slightly according to the parameters involved although the basic concept of the structure of these subroutines are the same. However, in most cases, only about 20 of the 256 subroutines are used because the parameters for specifying the exact positions of Reference Texts such as "Before Text/After Text", "Order" and Offset" are unlikely to have two parameters selected in the data selection process, otherwise ambiguity arises.

An example is taken to illustrate one particular Type of Extraction. Based on the programming structure of the subroutine written for this example, other subroutines can be created to handle other Types of Extraction using the same method. For example, the user has made a selection of check boxes "Reference Text", "Star Notation", "Count of Notation", "End Notation", "Before Text". The sum of binary codes of this selection is 31 (1+2+4+8+16). The pseudocodes for Type of Extraction "31" follow:

```
Rem : Subroutine - Data Extraction Based on Specified Type
Rem :                  of Extraction
Set X = Array Storing Four Groups of Data
Get Value of Reference Text from X
Find  Location of Reference Text in Source File
Set S.I.N. = Starting Identification Notation
Find the First S.I.N.
Find the Nth Count of S.I.N.
Find the Ending Identification Notation (E.I.N)
Find Exact Location of Before_String
If Location of Before_String < E.I.N Then
        Rem : Position of E.I.N. is After
        Rem : Before_String and That Does Not Comply
        Rem : with Conditions Specified
        Exit Subroutine
Else
        Retrieve Text (Nth S.I.N. to E.I.N.)
End If
```

The extracted text is stored into the component-related datafile which resides in the common database. Data from the various source computer systems are stored under the “Heading” of the involved computer system. Table 4.11 is an example of the component-related datafile “c:\data\shaft.dfl” which includes data for thee computer system of a Rapid Prototyping Machine and the AutoCAD package.

Table 4.11 An Extract of File “c:\data\shaft.dfl”

```
### Rapid PT M/C -> SLS2000 -> Technical Data
"warmup_height","11"
"build_height","211"
"left_feed_heater_set_point","109"
### CAD -> AutoCAD -> BOM
"1739","Bush"
"1739","10"
"1739","Brass"
"23456","Insert"
"23456","6"
"23456","Steel"
"2341","Stand"
"2341","1"
"2341","Cast Iron"
"678","Base"
"678","1"
"678","Steel"
```

As shown in Table 4.11, the Type of Computer Systems involved are “Rapid PT M/C” and “CAD”; the exact Name of the Computer Systems (packages) are “SLS2000” and “AutoCAD” while “Technical Data ” and “BOM” indicate the USE (Section 4.4.2.3) of the data. With this format, other systems know the exact location of relevant data.

4.5 Generic Database Management System

The Generic DataBase Management System (GDBMS) is one of the three key components of GNSIFS. Such a database system must be flexible in order to cope with data transfer among different combinations of computing systems. To get a better understanding of the development of database management systems and the reasons why a special designed DBMS is necessary for GNSIFS, database systems development was studied in some depth. The description of GDBMS and the methodology for its development are dealt with in this Section.

4.5.1 Database Models Appropriate for Proposed GDBMS

The functions of DBMS and its various types of data models including hierarchical, network, relational, free format, multi-user distributed structures and object-oriented are covered in Section 2.8. The proposed GDBMS is mainly concerned with the efficient transfer of alphanumeric data among computer systems and a relational database model "looks" appropriate for this task (it is shown in the later context that a flexible database system is also required) because :

- alphanumeric data transfer is reliable
- any data fields can be searched for quickly
- relational DBMS development tools are available
- it is well developed data model

Analysis of Database Development Tools on the market indicates that most of the popular development tools are for relational databases. Browning (1993) states that the popular relational database development packages on the market include DBase IV, Advanced Revelation, DataEase, KnowledgeMan, Microsoft Access, Microsoft FoxPro, Paradox, R:Base and SuperBase. Most of these packages have the built-in capabilities to create customised database structures which greatly reduces the time to develop the database management system. The main feature of this package is that the user has to define the lengths of fields as this allows the software to produce records of fixed lengths so that these records can be accessed very quickly with the help of index files. An index file contains the specific fields of the database followed by the record number of the record which contains that specific field. Since the field data of index file is sorted in order, the search for data is fast (Simpson, 1990).

As discussed in Section 4.2.5, there are two groups of information in the common database, one dealing with fixed information (Group A) such as costs, overhead rates, process times and machine availability while the other group (Group B) contains "flexible" information relating to the products, such as Bill of Materials, tool data, quality data etc.

For the fixed information of Group A data, the Relational DataBase Management System (RDBMS) is ideal since the length of fields can be easily assigned, e.g. the scrap rate field should not need more than 2 digits before decimal and 2 digits after decimal. The relational database development tools are used to create the database system for this group.

However, the relational database development tools are not appropriate for Group B data for which the length is not easily pre-determined. For example, the tool data for one component can contain 5 to 10 pages of text information while other components may have no tool data. A DBMS for this group must therefore have the flexibility to allow lengths from zero to unlimited characters. Consequently, a Flexible DataBase Management System (FDBMS) with a free format capability is required (Fig. 4.12). The main limitation of such a free format database is the time to locate a particular field data because the search must be done character by character. However, if the data in Group B is related to components, then, as long as the location of component string is available, the information can still be retrieved relatively quickly.

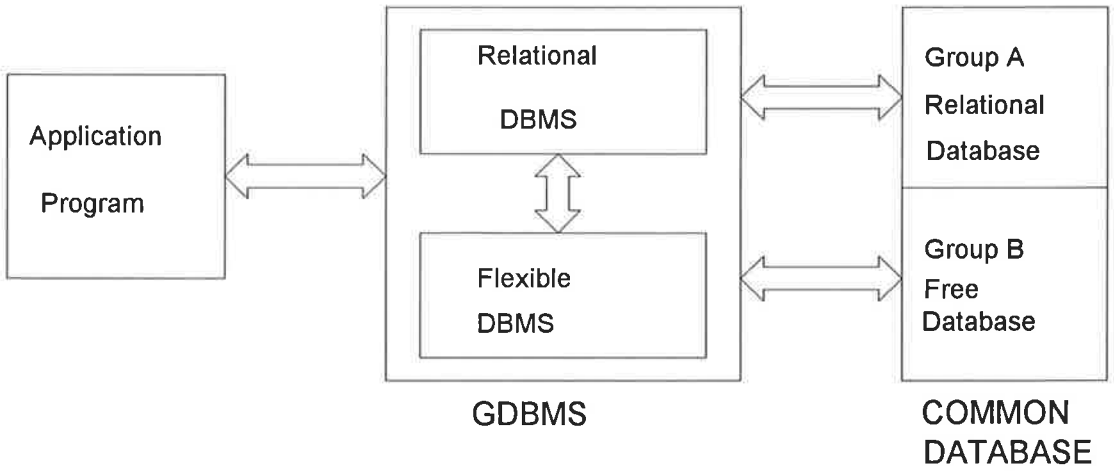


Figure 4.12 : A Proposed Generic DBMS

The features of the FDBMS must include :

- no limit on size of data fields;
- size of data fields to be variable during running time i.e. users can alter the size of data fields when actually running the program;
- number of fields to be variable during running time;
- search of records (component alphanumeric string) is done based on a unique key field, normally the component number.

4.5.2 Methodology for Developing RDBMS for Group A Data

The development steps of a RDBMS for group A data are shown as below :

(i) Determine the data fields to be included in the database system. Assign field names and field sizes to these fields, e.g. assembly line scrap rate, 3 digits before decimal and 2 digits after decimal.

(ii) Select an appropriate RDBMS development tool e.g. DBase IV, Microsoft Access, or Paradox etc. It should be noted that Microsoft Access is both a database development tool and with its own databases though it is limited to relational data model (Microsoft Corporation, 1994b).

(iii) With the use of the development tools, create data structure for the Group A data. For example in DBase IV or Microsoft Access the user "opens" the window for creating database tables following by entering the data fields, e.g. labor cost/hour, overhead rate, and the data sizes and format, e.g. date format, currency format etc. The in-built functions of these tools will then create the database table accordingly.

(iv) With the customisation features of the selected software, build a customised user interface to manipulate the Group A data. Most of the commercialised DBMS development tools come with the customisation or automation features (In Microsoft Access, Access Macros and Access Basic are the tools for customisation. In Paradox, programming tool, i.e. ObjectPAL, can be used to customise and automate database applications). The latest versions of database package, e.g. Dbase V and Microsoft Access 2.0, have "user-friendly" capabilities for creating these customised programs by adopting "familiar" user interface features such as "menu-driven functions", "pop-up dialogue boxes".

4.5.3 Concept of Flexible DataBase Management System

The concepts for Flexible DataBase Management System (FDBMS) is shown in Fig. 4.13. The database contains files which store the component strings of data extracted from the system files of various computer systems e.g. CAD, CAM, MRP-II etc. This database system is flexible because component strings can be added or removed while running the program and each file contains the alphanumeric string of one component. The size of each file is variable depending on the size of the data to be stored in the file.

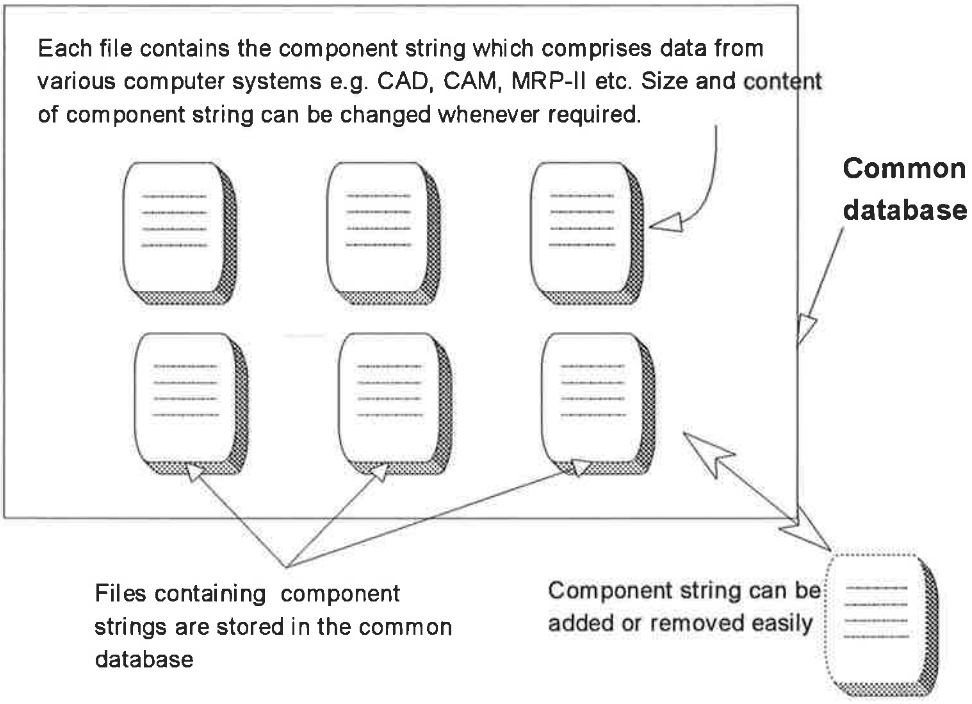

Each file contains the component string which comprises data from various computer systems e.g. CAD, CAM, MRP-II etc. Size and content of component string can be changed whenever required.
Common database
Files containing component strings are stored in the common database
Component string can be added or removed easily

Figure 4.13 : Concept of FDBMS

4.5.4 Methodology for Developing a FDBMS

The methodology for developing the FDBMS is illustrated in Fig. 4.14. In the central part files containing the alphanumeric string of components, are created or removed. However, during the process of creating these files the format of the data in the file must be taken into account as well.

The block at the left side provides information about the format of data and this information is used to change the format if required. Each datafile contains the component string which includes data from various systems. The size and number of these data strings can be varied and changed at any time.

The block at the right hand side illustrates the feature of this database system which allows variable data strings from various computer systems to be stored in the datafile. The detailed descriptions of the blocks are presented in the following sections.

4.5.4.1 Create Component-related Datafile in Common Database

The methodology to create the component-related datafile is illustrated in Fig. 4.15. Whenever a new component has been created the user is asked if a new datafile needs to be created. If the user does not want to create one, the process of creating the datafile will end. However, if the response is positive, a datafile will be created as a text file using any

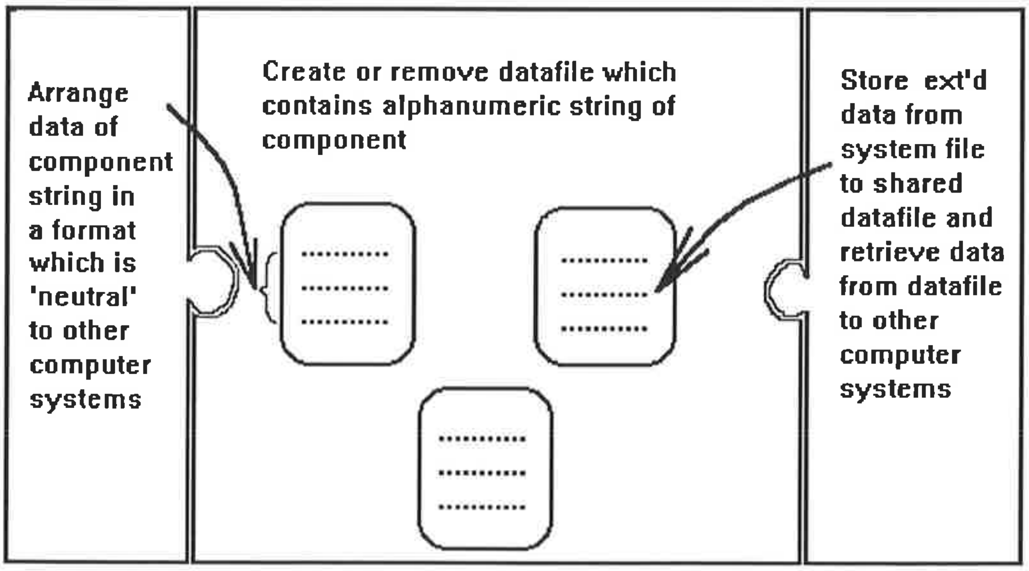

Arrange data of component string in a format which is 'neutral' to other computer systems
Create or remove datafile which contains alphanumeric string of component
Store ext'd data from system file to shared datafile and retrieve data from datafile to other computer systems

Figure 4.14 : Schematic Diagram of Methodology for FDBMS

programming language such as "C", Pascal or Visual Basic. Each file stores data associated with one particular component and the filename assigned to the file should be related to the component number. In the GNSIFS, the datafiles which store component strings (component-related datafiles) should have filenames with the part code of component as the name before the extension and with the extension "dfl" e.g. "12345.dfl". The filename and hence the component part code is limited to 8 characters. The subroutine to create the filename in Visual Basic is described below with pseudocodes to illustrate the structure of the computer program.

```
Rem : Subroutine - Creation of Component-related Datafile
Create an empty text (sequential) file for read and write
Store the part code of the component to Variable_A
Assign filename = Variable_A + ".dfl"
If filename exists
        Overwrite or quit ?
End if
Close file
```

The component data string associated with each computer system is to be placed under the label of the concerned computer system. For example, there are four computer systems to be linked for data transfer. These four computer systems including a Computer Aided Design (AutoCAD), a Computer Aided Manufacturing (SmartCAM), a Manufacturing

Resource Planning package (IMPCON) and a SpreadSheet Package (Excel). After the component-related text file is created, the labels indicating these four computer systems must be printed on the file. Data from any one of these four computer systems will then be stored under the label of the associated computer system. In order to ensure that the existing four computer systems, and all other computer systems which are added later, recognise the data format of this file, a format which is "neutral" to all the computer systems must be developed. Since the list of computer systems is stored in another text file, another subroutine needs to be written to extract the names of the computer system from the text file and write it to the datafile. The pseudocodes of the subroutine are:

```
Rem : Subroutine - Write Computer System Names to Datafile
Open text file storing list of computers for reading
Point to the beginning of the file
While not end of file
    Get name of computer system
    Write name to the datafile
    Point to the next line
End while
Close text file
```

4.5.4.2 Store/Retrieve Data from Datafile

Once the datafile is created and the labels of the computer systems printed, the file is ready to store information from any computer system. It is important to identify the source of the incoming data, e.g. from AutoCAD or Excel, so that it can be traced to the computer system from which the data is extracted. The incoming data consists of the labels for the Reference Texts and the assoiciated values of them (Section 4.4.2). The data is then written into the datafile in neutral format form. As the datafile is in free text format, the size of the data is variable. It can be one line or 100 pages, the limitation being the computer software used to create this file. Normally the file size is limited in terms of Megabytes.

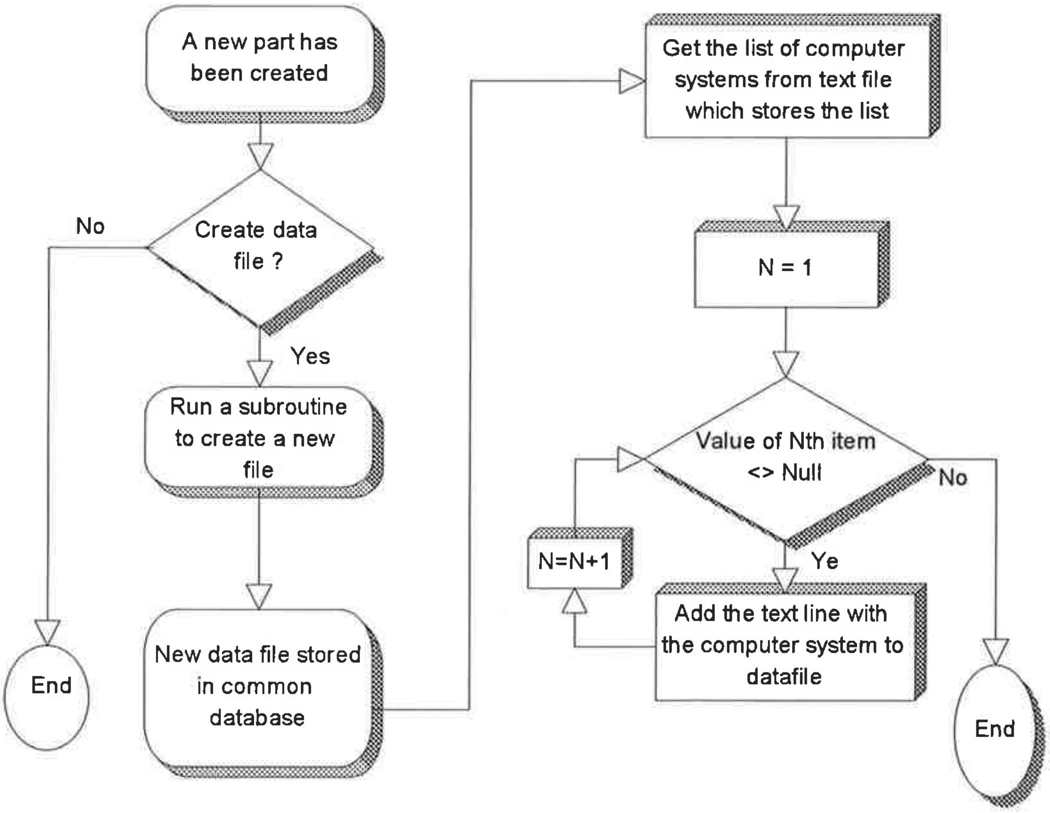

A new part has been created
Create data file ?
No
Yes
End
Run a subroutine to create a new file
New data file stored in common database
Get the list of computer systems from text file which stores the list
N = 1
Value of Nth item <> Null
No
Ye
N=N+1
Add the text line with the computer system to datafile
End

Figure 4.15 : Creating a New Datafile in the Database

Fig. 4.16 illustrates the flow chart for data storage and retrieval. The pseudocodes of the data storage subroutine are:

```
Rem : Subroutine - Store Data to Datafile
Rem : C.S. = Computer System
Create a variable to store the name of C.S.
Put the name of C.S. in variable
Seek C.S. name in database from line starting with "###"
Rem : remove previous data belong to the C.S.
If data exists below this line and the next C.S. line
        remove it
End if
Store data below the C.S. label line
```

To retrieve a particular group of data from the datafile (component-related datafile), e.g. to retrieve the "Technical Data" of "SLS2000" from component-related datafile "c:\data\shaft.dfl" as shown in Table 4.11, the associated computer system i.e. "SLS2000" is located first. Once the location is known, the database system is able to retrieve the data which belongs to the associated computer system i.e. "warmup_height ....". The data retrieved contains two groups, the labels of the Reference Texts i.e. "warmup_height", "build_height" and " left_feed_heater_set_point." and the associated values i.e. "11, 211,109". The FDBMS is able to distinguish these two groups because of the neutral

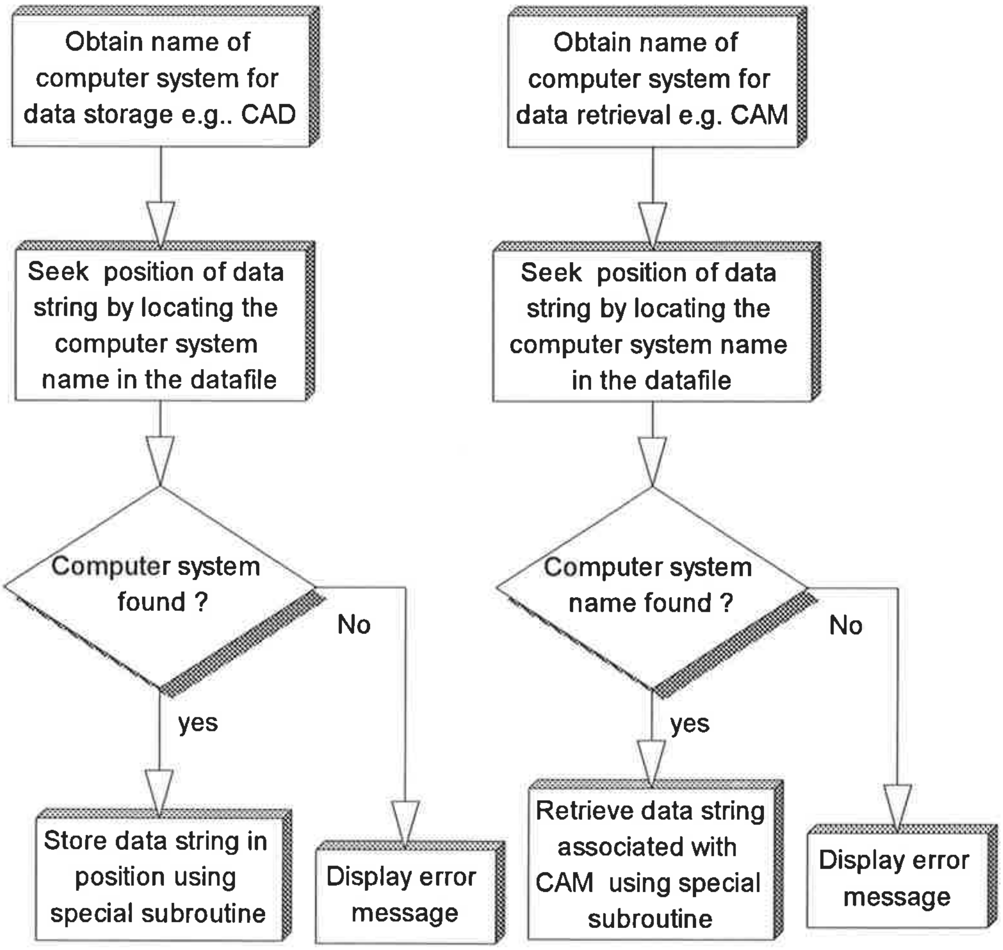

Obtain name of computer system for data storage e.g.. CAD
Seek position of data string by locating the computer system name in the datafile
Computer system found ?
No
yes
Store data string in position using special subroutine
Display error message
Obtain name of computer system for data retrieval e.g. CAM
Seek position of data string by locating the computer system name in the datafile
Computer system name found ?
No
yes
Retrieve data string associated with CAM using special subroutine
Display error message

Figure 4.16 : Flow Chart of Methodology for Data Storage and Retrieval

format and the fact that a Comma is used as a "separator". Fig. 4.16 shows the flow chart for data retrieval and pseudocodes for the data retrieval subroutine are :

```
Rem : Subroutine - Retrieve Data from Datafile
Create a variable to store the name of C.S.
Put the name of C.S. in varible
Seek C.S. name in database from line starts with "###"
Go to the next line
While the line does NOT start with "###"
        Retrieve data
        Separate data into parameter group and value group
        Send data to specified locations
        Go to next line
End while
```

4.6 Generic Format Translator

The third component of GNSIFS is the Generic Format Translator (GFT) which converts the format of the data from one type to another. As shown in the functional structure diagram (Fig. 4.4), GFT plays the important role of reformatting the data before being sent to other computer system.

4.6.1 Types of Data Formats

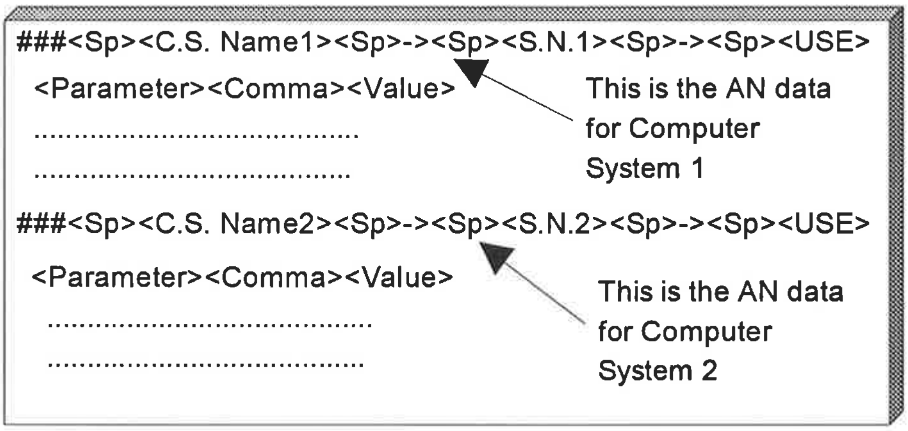
There are a number of established data formats used in text files.. These formats include CDF, SDF and BDF which have been explained in Section 2.10. Most of the common packages, such as Access, DBase IV, Excel, support one or more of these formats but it is possible that some packages require a different format and GFT must also deal with this.

4.6.2 Functions of GFT

As discussed in Section 2.7, the problem with the existing data exchange methodology, using established standards such as PDES and IGES is the need to develop specific conversion programs to ensure efficient transfer of data between systems. In GNSIFS, the format translation operations are handled by GFT which has built-in programs for data conversion. Basically, GFT automatically converts the extracted data to a neutral format before being stored in the common (central) database. In addition, GFT converts the neutral format to the format which is required by the receiving computer system . In case the receiving system requires a format other than the common text formats, such as CDF, SDF and BDF, the user is able to define the specific format required, since GFT enables the user to create specific format converters without the need of writing any codes.

4.6.2.1 Automatic Conversion to Neutral Format

During the data extraction process the data extracted is automatically converted to a neutral format before being put into the common database. Fig. 4.17 illustrates the data structure of the neutral format. The alphanumeric data from various computer systems is grouped together in this text file.



###<Sp><C.S. Name1><Sp>-><Sp><S.N.1><Sp>-><Sp><USE>
<Parameter><Comma><Value>
....
....

This is the AN data for Computer System 1

###<Sp><C.S. Name2><Sp>-><Sp><S.N.2><Sp>-><Sp><USE>
<Parameter><Comma><Value>
....
....

This is the AN data for Computer System 2

Note :
-> Start of text portion which contains the data for the associated computer system
Sp -> Blank Space
S.N. -> Software Name
AN -> Alphanumeric
C.S. -> Computer System

Figure 4.17 : Format of Data in Component-related Datafile

The example for the CAM system is :

```
### CAM -> SmartCAM -> Tool Size
```

Data for this computer system is placed below this label line. As seen in Fig.4.17, the label related to the parameter is followed by a Comma and then the value. The data for SmartCAM might be :

tool number, 4

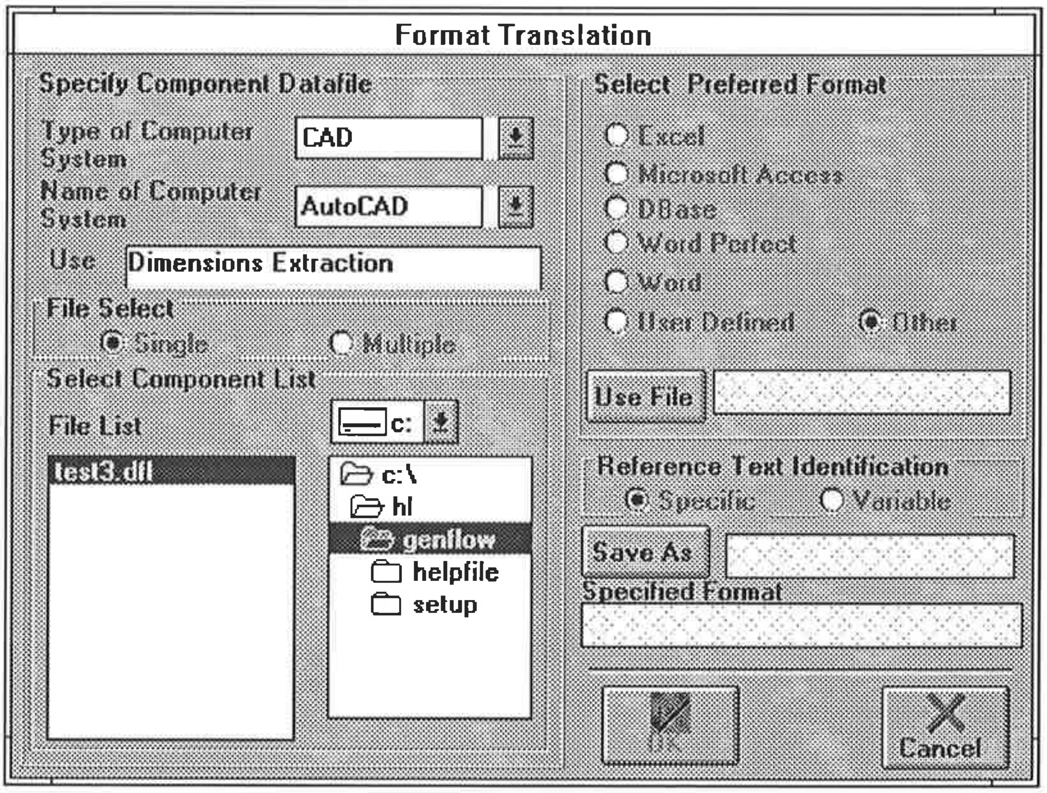
tool size for milling,5.45

tool size for cutting, 6.78

....

4.6.2.2 Conversion of Data to System Format

As shown in Fig. 4.18, the other function of GFT is the conversion of data from neutral format to system data format. The user specifies where the data is to be sent. As mentioned in the previous context, packages such as Excel, Access, Word, WordPerfect support one or more of the established formats, including CDF, SDF and BDF. If the data of a component-related datafile, e.g. "c:\data\shaft.dfl" (Table 4.11), which is in neutral format, is to be transferred to a package, e.g. Excel, which supports BDF, an "internal"



Format Translation

Specify Component Datafile

Type of Computer System: CAD

Name of Computer System: AutoCAD

Use: Dimensions Extraction

File Select: ● Single ○ Multiple

Select Component List

File List: test3.dfl

c:

c:\
hl
genflow
helpfile
setup

Select Preferred Format: ○ Excel ○ Microsoft Access ○ DBase ○ Word Perfect ○ Word ○ User Defined ● Other

Reference Text Identification: ● Specific ○ Variable

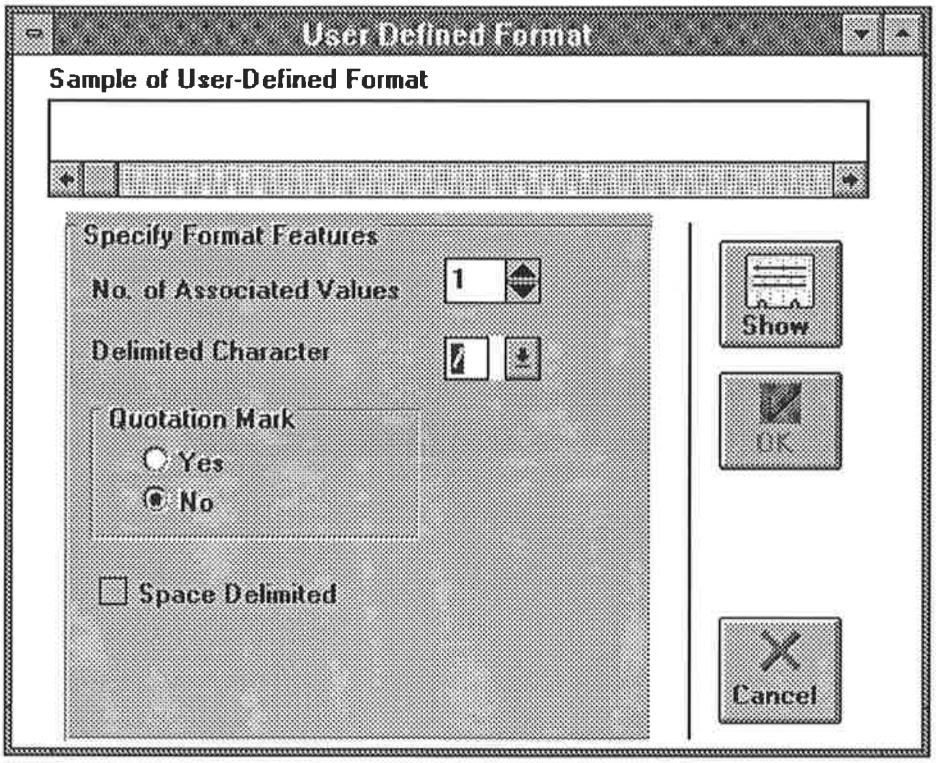
Specified Format

Figure 4.18 : Specification of Preferred Format

conversion program is required to convert the data e.g. the "Techical Data" for computer system "SLS2000" such as "warmup_height ......", from neutral format to the system format which is, in this case, BDF. Similar conversion programs have to be written and become part of GFT, for other established formats e.g. SDF and CDF. If the receiving computer system requires a special format, the user can define this format by clicking on the **User Defined** option on the form. As shown in Fig. 4.19, the user can define special types of data format, such as Backward or Forward Slash ("/","\"), Ampersand ("&"), Percent ("%") or any other character. In addition, the user can also specify whether or not quotation marks are required for the labels. Once the format is defined, the user can assign a file which stores the information of this user-defined format so that can be used again just by specifiying the filename. Fig. 4.18 is repeated in Chapter 5 with more explanation of its functions.

4.6.3 Methodology for Creating Generic Format Translator

The GFT consists of several subroutines which have been developed using Visual Basic programming language. The two main subroutines are described in the following sections.



User Defined Format

Sample of User-Defined Format

Specify Format Features

No. of Associated Values: 1

Delimited Character: /

Quotation Mark

○ Yes

● No

□ Space Delimited

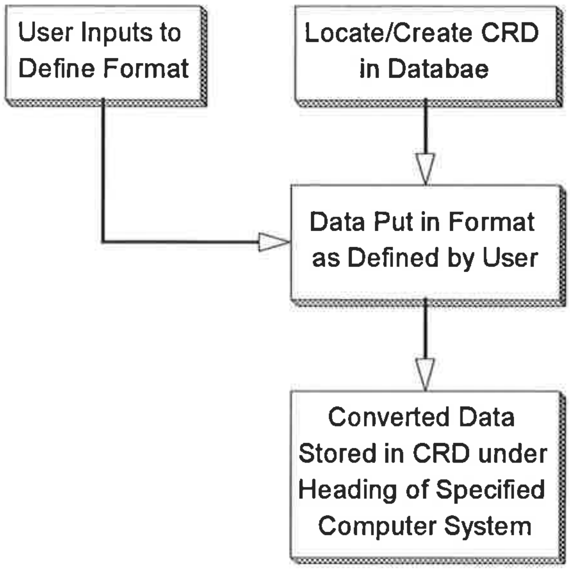
Figure 4.19 : Format Defined by User

4.6.3.1 Creating Automatic Translator to Neutral Format

The subroutine for this neutral format translator allows the user to specify the type and name of the computer system involved and the name of the component related datafile. If such a datafile already exists, the user has the option of using it or to create another one. Once the datafile is present, GFT looks for the label of the specified computer system and stores the extracted data under this heading. The pseudocodes to create this neutral format translator are :

```
Rem : Subroutine - Neutral Format Conversion
Find Component Related Datafile (CRD) in Common Database
```

```
If  CRD Not Found  Then
          Create One
End If
Rem : C.S. = Computer System
Find  the Portion belongs to the Specified Type and Name of  C.S. in File
If  Specified Type and Name of  C.S. Not Found Then
          Create One
End If
Set N =1
Repeat
          Put the  "N" th Label Name under the Heading of the Selected C.S.
          Put  a Comma notation sign after Label Name
          Put the Associated Value after Comma sign
          Insert Double Quotation Signs to Label and Associate Value
          Set N = N + 1
Until All Extracted Data Put in CRD
```


User Inputs to Define Format
Locate/Create CRD in Databae
Data Put in Format as Defined by User
Converted Data Stored in CRD under Heading of Specified Computer System

CRD -> Component Related Datafile

Figure 4.20 : Flow Chart for an User-defined Format Translator

4.6.4 Methodology for Creating User-defined Format Translator

The user-defined format translator consists of another two subroutines; one is for capturing inputs by the user and the other is a "user-defined format translator". In the first subroutine, the user defines the type of format, which is stored in a temporary memory location in the computer while the second subroutine deals with the format conversion and puts the converted data in the component related datafile (Fig. 4.20). The pseudocodes for creating these subroutines are :

Subroutine One

```
Rem : Subroutine - Capturing User Inputs
Input the Type of Delimited Character
Input the Number of Associate Values
Input whether quotation mark is required or Not
If  Quotation Mark is required Then
        Input Single or Double
Endif
Store all the Input Values by User in Memory
```

Subroutine Two

```
Rem : Subroutine - User-defined Format Translator
Find Component Related Datafile (CRD) in Common Database
If  CRD Not Found  Then
        Create One
End If
Rem : C.S. = Computer System
Find  the Portion belongs to the Specified Type and Name of  C.S. in File
If  Specified Type and Name of  C.S. Not Found Then
        Create One
End If
Set N =1
```

```
Repeat
        Put the "N" th Label Name under the Heading of the Selected C.S.
        Put a User-defined Delimited Character after Label Name
        Put the Associated Value after User-defined Delimited Character
        Set $N = N + 1$
Until All Extracted Data Put in CRD
If Quotation Mark is Required Then
        Insert Quotation Marks as Defined by User
Endif
```

4.7 Summary

The development of a generic link between any number of computer systems is discussed in this Chapter. Firstly, the generic link between CAD and MRP-II systems is discussed and this link is expanded to include other computer systems. A generic, information system called Generic Non Standard-specific Information Flow System (GNSIFS), which is able to handle computer systems running with various operating systems, is proposed to achieve generic links to computer systems, without the need to develop specific format translators. It consists of three main components, namely Generic Data Extraction System (GDES), Generic DataBase Management System (GDBMS) and Generic Format Translator (GFT).

The two main functions of GDES are (i) to identify data to be extracted from system datafiles easily and accurately, and (ii) to extract the data from the system datafiles and store them in component-related common datafiles. Data Extraction Information (DEI) file, which stores information regarding the details of parameters required for data extraction, plays an important role in the development of the methodology for creating GDES. During the process of data identification, GDES keeps records of the input by the user and stores the information in DEI file. The information stored in DEI file is later used in the data extraction process.

The Generic DataBase Management System (GDBMS) consists of two units, namely the Relational DataBase Managment System (RDBMS) for Group A or fixed general data and Flexible DataBase Management System (FDBMS) for group B or component-related data. The RDBMS can be developed by selecting an appropriate commercial relational database while FDBMS requires specific design to handle the Group B data.

The task of the Generic Format Translator (GFT) is to perform translation of data from a specific format to a neutral format or vice versa. Therefore additional systems can always be added with minimal effect on existing systems.

5. GENFLOW - THE APPLICATION PROGRAM

5.1 Introduction

The GNSIFS concept aims to achieve a cost effective, non-standard-specific, information flow system. To realise such a system requires a generic methodology which provides the steps, methods and techniques to design and implement it. The methodology to create the proposed information flow system is described in Chapter 4. With the methodology available, the proposed information flow system can be designed and implemented provided that the required tools such as computer languages and third party macros are available (See Fig. 5.1) . GenFlow is written in Visual Basic and "C" and has six main parts, namely Data Group, Data Transfer,

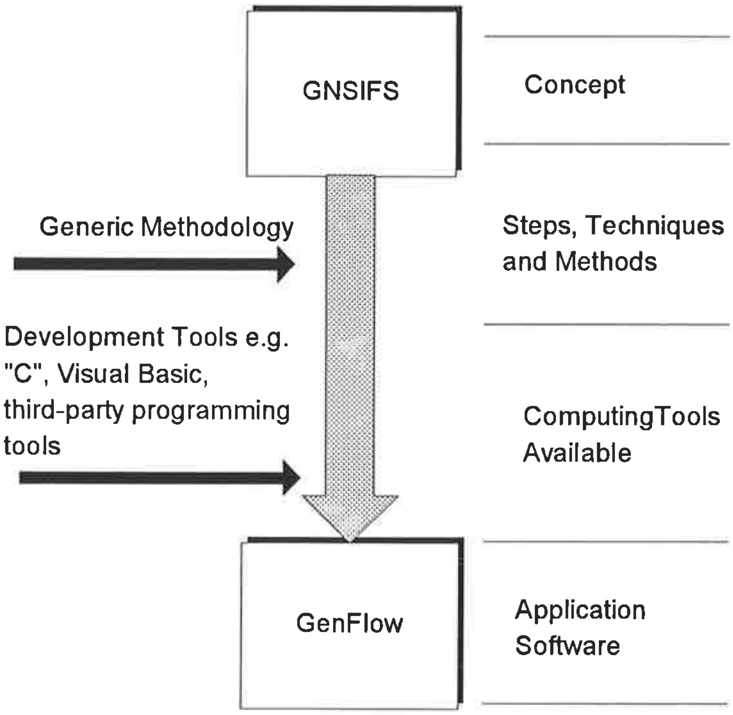

GNSIFS
Concept
Generic Methodology
Steps, Techniques and Methods
Development Tools e.g. "C", Visual Basic, third-party programming tools
ComputingTools Available
GenFlow
Application Software

Figure 5.1 : The Evolution of GenFlow

Format Translator, External Program, Browser and Help. In this Chapter, the design and features of GenFlow are explained.

5.2 General Description of GenFlow

Figure 5.2 shows the functional structure of GenFlow which is designed to achieve data transfer among application programs, such as AutoCAD, SmartCAM, Excel, ComputerVision, Access etc., which may be running in different operating systems such as DOS/Windows and UNIX. Data to be transferred can be in any text format and the user can identify it by just pointing and clicking. Genflow extracts and converts the data into a neutral format automatically so that it can be shared by other systems. The common database, which stores the extracted data, can be put in the server of the network so that other systems can access it easily.

The hardware requirements for running this program are :

- 386 CPU with at least 33 MHz clock speed
- Windows 3.1 or better
- 4 Mb RAM or more
- Hard disk with at least 40 Mb storage space or a network server with 50 Mb for the common database

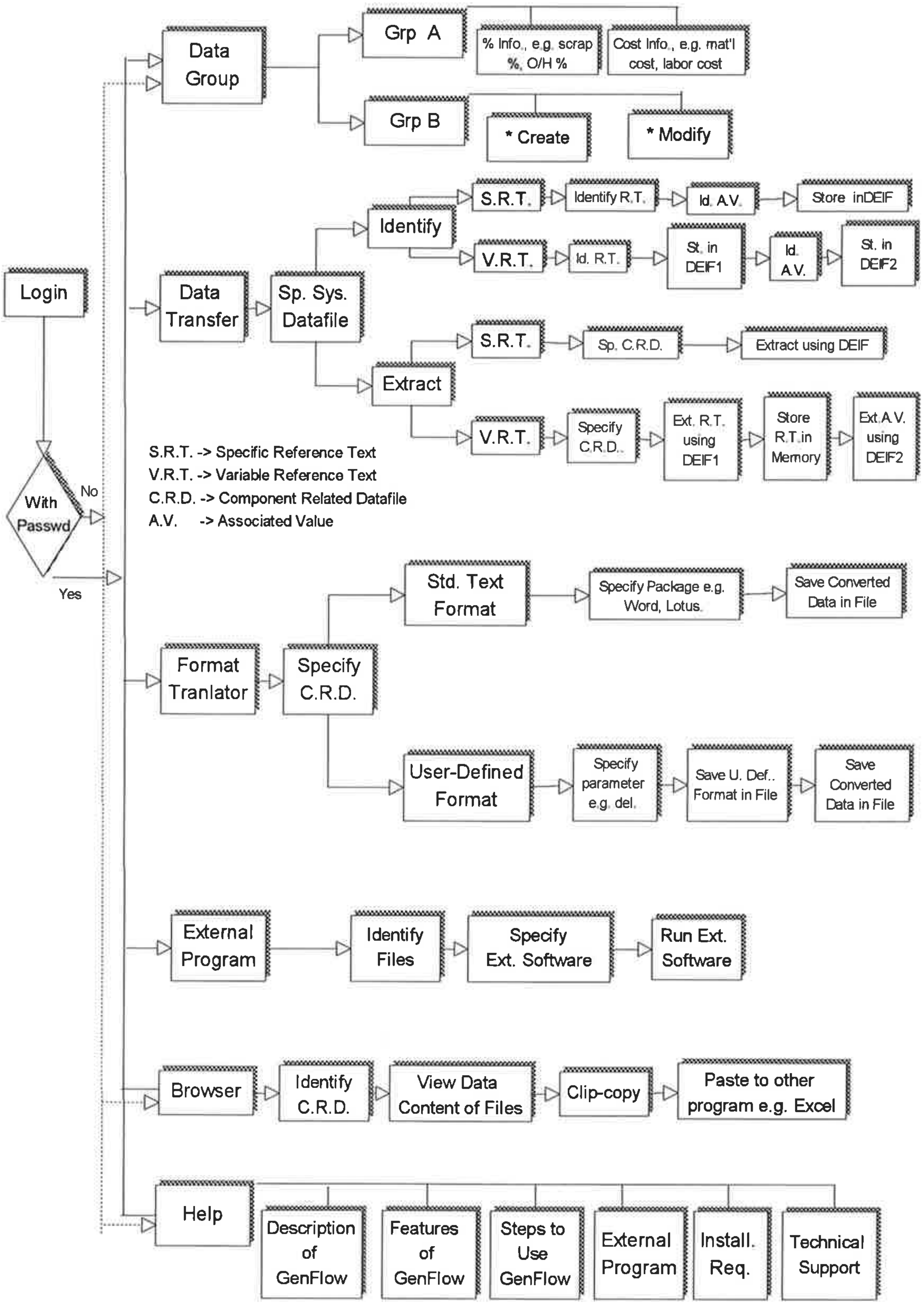


Figure 5.2 : Functional Structure of GenFlow

A comprehensive on-line help system, incorporated with table of contents, search and glossary commands, comes with the program and the users can learn the program in a familiar Windows graphical interface environment. Features of hypertext and hypergraphic, which allow the user to click on the text or graphic, are incorporated in the program as shown in the Help Function (Fig. 5.10 & Fig. 5.11).

5.3 Steps in Using GenFlow

The steps in using GenFlow are covered in the following context. Source codes of the functions of the program are listed in Appendix A.

5.3.1 Login

To use GenFlow, the user must login with or without a password. Those logging-in without a password can only view the data in the common database such as Group A of Data Group, Brower and Help while those with a password can edit or modify the data in the common database and access all the six functions.

After logging in, the main menu of GenFlow appears on the screen (Fig. 5.3).

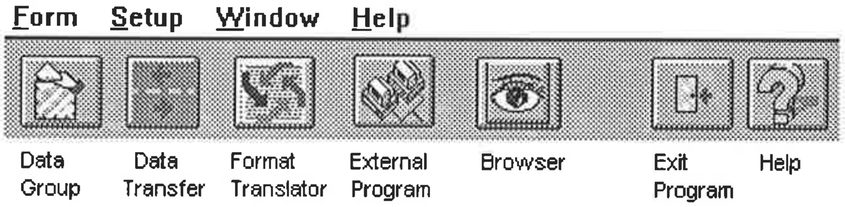
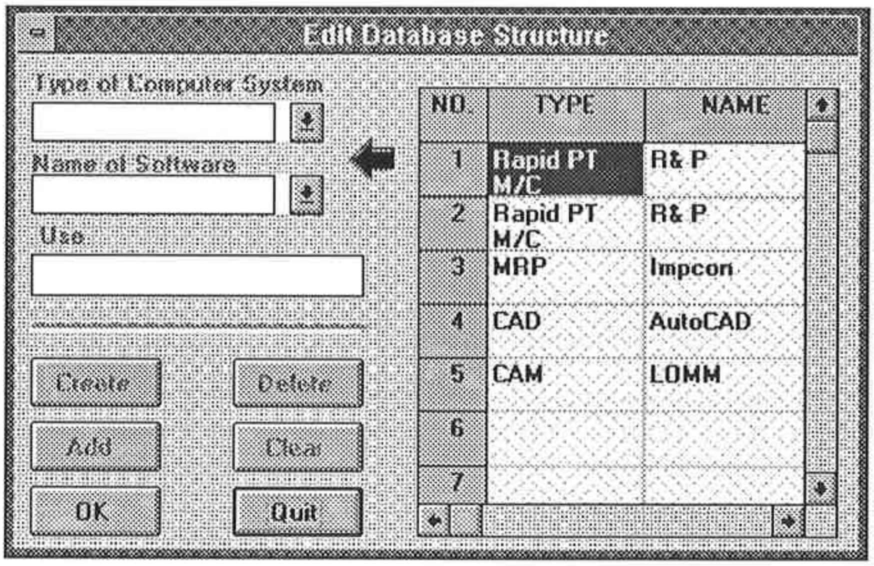


Figure 5.3 : Main Menu of GenFlow

5.3.2 Data Group

To set up the system, users must specify the combination of computer systems. The steps are :

(1) Click on the icon and this opens the form shown in Fig.5.4.



Edit Database Structure

NO.	TYPE	NAME
1	Rapid PT M/C	R& P
2	Rapid PT M/C	R& P
3	MRP	Impcon
4	CAD	AutoCAD
5	CAM	LOMM
6		
7		

Figure 5.4 : Edit Database Structure Form

(2) Input the information in the three text boxes namely “Type of Computer Systems”, “Name of Software” and “Use”. As discussed in Section 4.4.2.3, the **USE** textbox is designed to indicate the specific purpose of using the extracted data.

(3) GenFlow checks whether the file for the database structure exists or not. If the file does not exist, the **Create** button is enabled otherwise

the **Add** button (add to the existing list of computer systems) is enabled. Click on the enabled button to put the information in the database structure. The added system will be shown in the Grid (the table at the right side of the form)

(4) If a computer system is to be deleted, highlight that computer system on the Grid and click on **Delete** button.

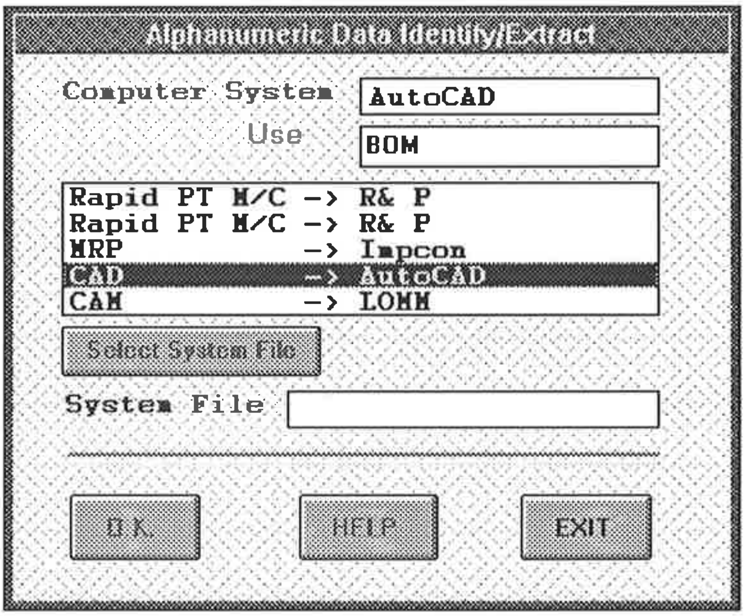
(4) Click on the **Save** button to save the settings and exit the form.

Note : The term "computer system" is explained in Section 1, normally it is referred to a package, e.g. AutoCAD, however, when a machine, e.g. "SLS Model 2000" Rapid Prototying Machine is included, it refers to the computer control unit (hardware and software) of the machine.

5.3.3 Data Transfer

To invoke the data identification and data extraction of GDES, click the second icon on the Tool Bar of the Main Form.

The window as shown in Figure 4.10 is displayed (Fig. 4.10 is repeated below). This window of Fig. 4.10 allows the selection of the system datafile which contains the source data. The steps are :



Computer System: AutoCAD

Use: BOM

```
Rapid PT M/C -> R& P
Rapid PT M/C -> R& P
MRP          -> Impcon
CAD          -> AutoCAD
CAM          -> LOMM
```

System File:

Repeat of Figure 4.10 : Data Identify/Extract Form

(1) Select the computer system in the list box which is in the middle of Fig.4.10. The selected system is shown in the two text boxes namely “Computer System” and “Use”.

(2) Press the **Select System File** button and select the appropriate source datafile.

(3) Click **OK** button to save and exit the window.

5.3.3.1 Data Identification

A new menu bar appears when the **OK** button of Fig. 4.10 is clicked, as shown in Fig. 5.5. Click on the **Data** menu and select **Data Identifcation**; the user is then asked to select either **Specific Reference Text** and **Variable Reference Text** (Specific

Reference Text is the "normal" data identification using one DEI file while Variable Reference Text is the data identification using two DEI file as explained in Section 4.4)

Figure 5.5 : Data Identification Menu Bar

If **Specific Reference Text** is selected, the form (Fig. 4.6) is displayed. Proceed as follows :

(1) Select the options from the eight check boxes. The text to be extracted can be identified based on the selection.

(2) Click on the **OK** button to continue.

(3) The parameters specified by the user are recorded in DEI file which will be used in the subsequent data extraction process. If a DEI file for the same computer system already exists, the user has the choice of either overwriting or appending the data to the existing one.

If **Varible Reference Text** is selected, the form as shown in Fig. 5.6 is displayed. The steps for using this form are:

(1) Select the options from the eight check boxes to identify the Reference Text which remains unchanged e.g. the text string "Part List" as shown in Fig. 4.8 is chosen as the Reference Text since it remains unchanged

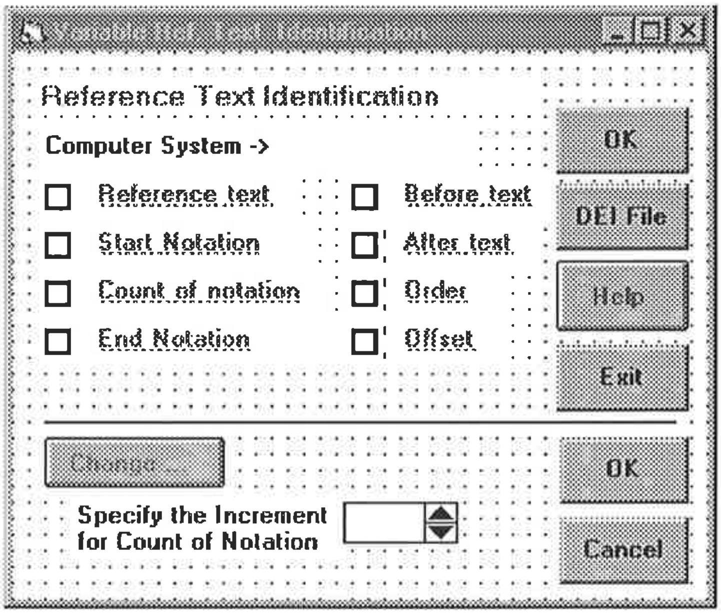


Figure 5.6 : Variable Reference Text Identification

for different component-related datafiles though the part codes e.g. "1739" etc for different components are different.

(2) Identify the associated values of this Reference Text, e.g. "1739", "23456" etc, as shown in Fig. 4.8.

(3) Click on the **OK** button to create first DEI file.

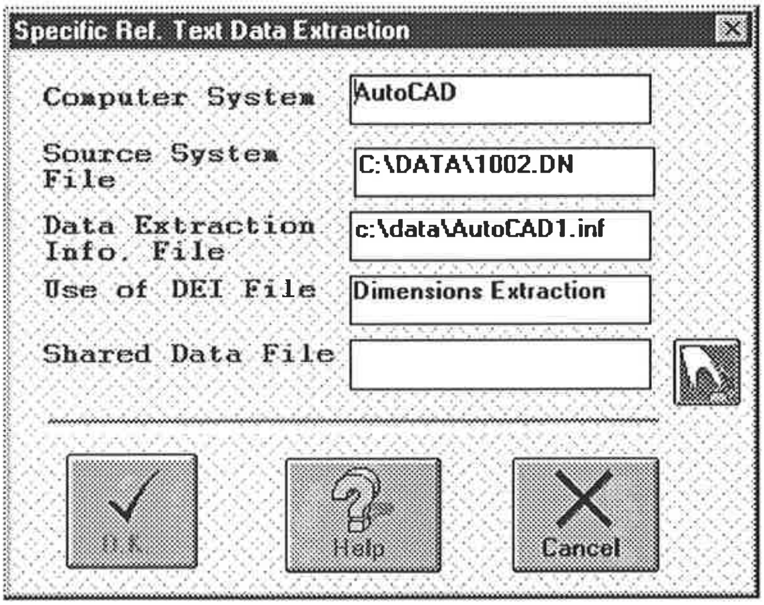
(3) The associated values of the first DEI file, e.g. "1739", "23456" etc, are now the "second level" Reference Texts which are used to identify their associates values. ("1739" is the "second level" Reference Text for "Bush", "10" and "Brass" (Fig. 4.8))

(4) Click on the **OK** button for creation of second DEI file.

5.3.3.2 Data Extraction

Click on the **Data** menu of the menu bar (Fig. 5.5) and select **Data Extraction**; the user is then asked to select either **Specific Reference Text** or **Variable Reference Text**.

If it is a "normal" data extraction using <u>one</u> DEI file, select **Specific Reference Text** and the form is displayed as shown in Fig. 5.7.



Specific Ref. Text Data Extraction

Computer System: AutoCAD

Source System File: C:\DATA\1002.DN

Data Extraction Info. File: c:\data\AutoCAD1.inf

Use of DEI File: Dimensions Extraction

Shared Data File:

Figure 5.7 :Data Extraction Form

Unlike data identification, this form is used for both Specific and Variable Reference Text extraction. GenFlow selects the DEI files accordingly (one DEI file for Specific

Reference Text and two DEI files for Variable Reference Text). The steps for data extraction are:

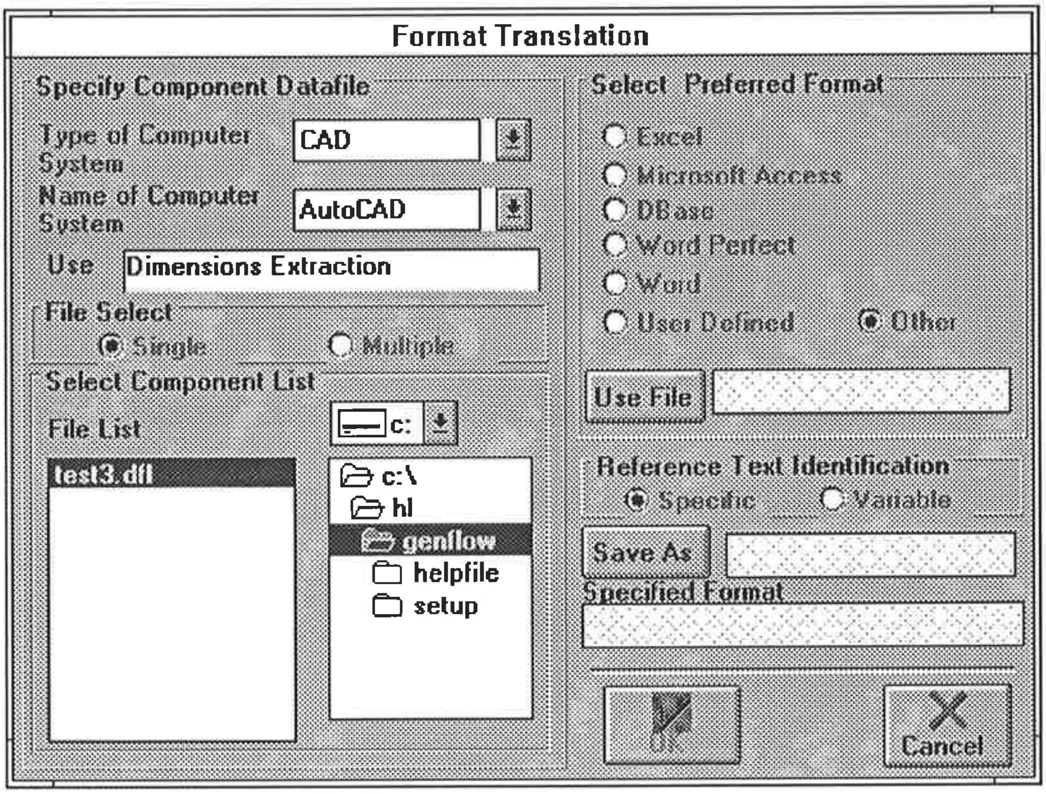
(1) Insert the name of computer system, the source datafile and the DEI file. If the user has specified this information previously, it will already appear in Fig. 5.7. Click on the icon next to the text box of **Shared Data File** to specify the file (component-related datafile) where the extracted data will be stored.

(2) GenFlow performs the data extraction task based on the information provided by DEI file, which has been specified by the user (the data extraction process is explained in Section 4.4.2). The extracted data is stored in the component-related datafile and if it is a Variable Reference Text data extraction, the associated values extracted, based on the first DEI file, are used as the "second level" Reference Texts for the extraction of their associated values.

(3) Select **Save** from the **File** menu to save the extracted data to the datafile.

5.3.4 Format Translator

This function is designed for the conversion of data formats to ensure that data is in the correct format before it is sent to the receiving system. The details of format translation is described in Section 4.6. To use the Format Translator function, click the icon on the tool bar of the main menu.



Format Translation

Specify Component Datafile

Type of Computer System: CAD

Name of Computer System: AutoCAD

Use: Dimensions Extraction

File Select: ● Single ○ Multiple

Select Component List

File List: test3.dfl

c: — c:\ hl genflow helpfile setup

Select Preferred Format: ○ Excel ○ Microsoft Access ○ DBase ○ Word Perfect ○ Word ○ User Defined ● Other

Reference Text Identification: ● Specific ○ Variable

Specified Format

Repeat of Figure 4.18 : Specification of Preferred Format

The form as shown in Fig. 4.18 is displayed (Figure is repeated here). The steps for using this function are:

(1) Fill in the three text boxes namely the "Type of Computer System", "Name of Computer System" and "Use". The user can make a selection from the drop-down lists which are next to the text boxes of "Type of Computer System" and "Name of Computer System".

(2) Select the datafiles for format translation from the "Files List" box. It can be a single or a multiple files selection. The keyboard commands for multiple select are the same as the Windows standard keyboard commands used in File Manager. Holding down the **Ctrl** key while

clicking on files will select/deselect the file/files, while holding down the **Shift** key will select/deselect the list of files in between.

(3) Select the format to be converted. The user can select the package to which the data is to be sent, e.g. Word, WordPerfect, or define a specified format by clicking on the option button to select the preference. If the **User-defined format** option is selected, Fig.4.19 appears where the user can define the desired format.

(4) Click on the **OK** button to accept and save the user-defined format.

5.3.5 External Program

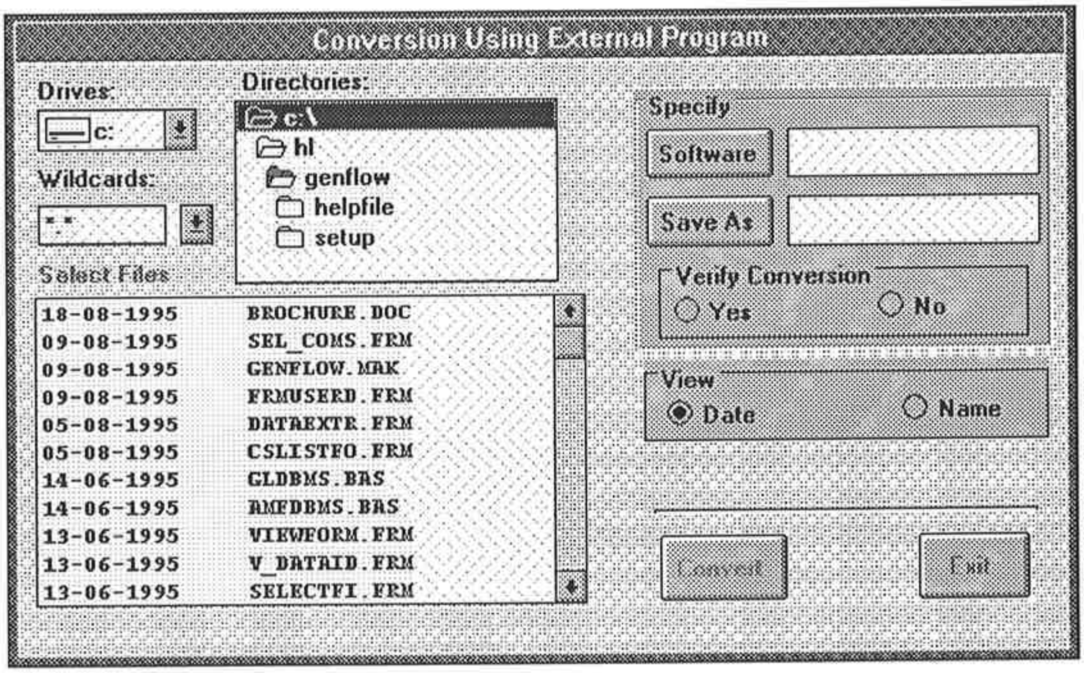
This function is designed to facilitate the use of external programs to perform tasks within the GenFlow environment. An example is the external program which converts the UNIX text to DOS text. It should be noted that all the computing platform such as UNIX, DOS/Windows and Macintosh support the ASCII text format which is a "cross-platform" standard. However, when a file with ASCII text is transferred from a UNIX system to a DOS system, the text may fall into different locations although all the ASCII text is still there. This is because some of the "non-printable" characters, such as carriage return & line feed, are not the same for DOS and UNIX. To overcome this a small program called UNIX2DOS is needed.

After clicking on the icon, the form with title "Conversion Using External Program" is displayed as shown in Fig. 5.8.

The steps for using this function are:

(1) Select the file/files which will be used with the external program. Again, this can be a single or multiple select option as for the Data Transfer function. The files in the list can be sorted based on dates or on the names of files by clicking on the **Date** button or **Name** button respectively.

(2) Specify the external conversion program to be used by clicking on the **Software** button, after which, a Dialogue Box for selection of files is displayed. The external program selected will appear in the text box next to the **Software** button.



Drives: c:

Wildcards: *.*

Directories:
c:\
hl
genflow
helpfile
setup

Select Files

18-08-1995	BROCHURE.DOC
09-08-1995	SEL_COMS.FRM
09-08-1995	GENFLOW.MAK
09-08-1995	FRMUSERD.FRM
05-08-1995	DATAEXTR.FRM
05-08-1995	CSLISTFO.FRM
14-06-1995	GLDBMS.BAS
14-06-1995	AMFDBMS.BAS
13-06-1995	VIEWFORM.FRM
13-06-1995	V_DATAID.FRM
13-06-1995	SELECTFI.FRM

Verify Conversion ○ Yes ○ No

View ● Date ○ Name

Figure 5.8 : Using External Program

(3) Specify the destination filenames of files after running with the external program by selecting **Save As** and typing the filename/filenames. For multiple selection of files, the filenames to be specified must use a "wildcard" e.g. *.txt. With this notations, the files will retain the filename string before the extension but the extension string will become ".txt". For example, filename "testing1.unx" will be changed to "testing1.txt".

(4) The user can verify the conversion of every file by clicking on the **Yes** option button. If **No** button is selected, the conversion will be performed to all the files which have been selected by the user in step (1).

(5) When all the parameters have been entered, the **Convert** button is enabled and by clicking on this button the conversion is performed.

5.3.6 Browser

This function allows the datafiles to be viewed in a tabular form. This is useful when the results of several datafiles need to be compared. In addition, the data displayed in the Grid, which is a term used to describe the form which has rows and columns like a Spreadsheet, can be copied and pasted to another Spreadsheet for further analysis.

When the Browser icon is clicked, Fig. 5.9 appears. The steps for using this function are :

(1) Click on the Arrow Down button to select the type of computer system, e.g. CAD, and the name of computer system, e.g. AutoCAD, from the drop-down list. When the computer system have been specified, the strings for **Use** and **DEI file** will automatically be entered.

(2) Specify whether it is for Specific or Variable Reference Texts by clicking on **Specific** or **Variable** respectively. This is important as the Variable Reference Text ensures that the tabular form e.g. a part list is displayed accurately while the Specific Referene Text ensures that the correct associated value is attached to the Reference Text.

(3) Click on the **Select** button to select files which can be single or multiple file.

(4) If **View** is selected to browse through the data from the selected files, the data is shown in a Grid form. The user can select part of the data by "point and drag" and once the data is selected the **ClipCopy** button is enabled and the data is copied to a Clipboard. Once the data is stored in Clipboard it can be pasted to other programs such as Excel or Word.

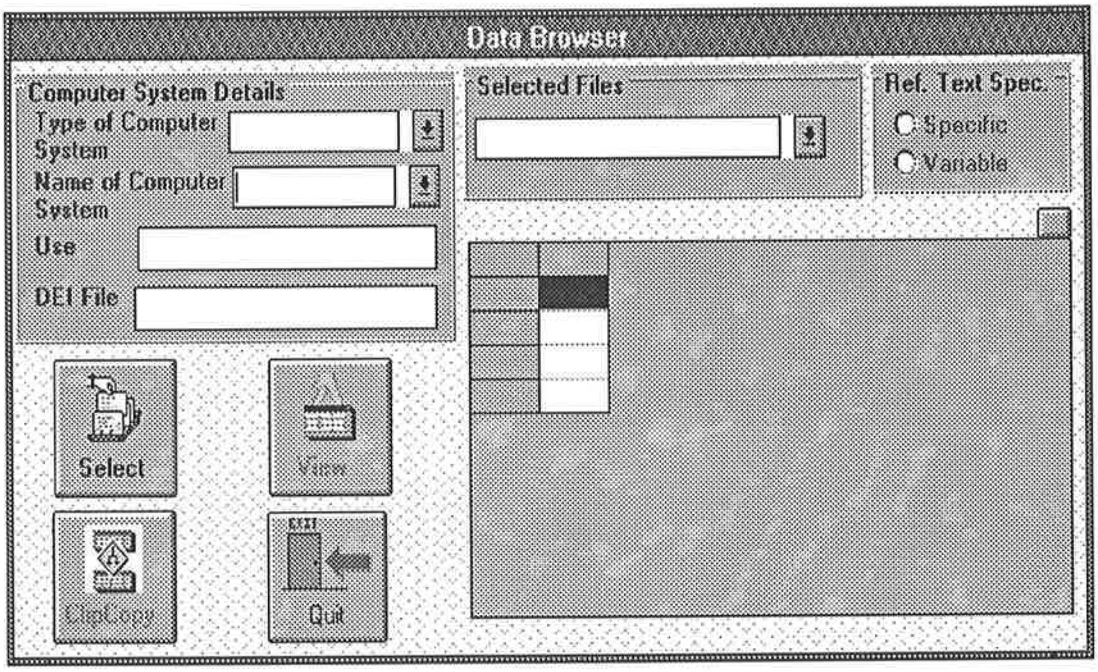


Figure 5.9 : Browser Function

5.3.7 Help

This is the "on-line" help with the "look and feel" of a standard Windows help system. This function was developed using a proprietary software called HDK. The manual of this package (HDK Manual, 1994) provides detailed information regarding the creation of a Help file without any programming. To invoke Help function, click the Help icon on the Tool Bar.

5.3.7.1 Table of Contents

The Table of Contents of this Help function is shown in Fig. 5.10. All the topics in the Table of Contents have the "hypertext jump' capability which means that when they are clicked on, another page of information, related to the clicked topic, will appear. The topics are described below :

(a) Brief description of GenFlow - covers the general functions of GenFlow, its general capabilities to enable data transfer among computer systems.

(b) Features of GenFlow - covers the general features of GenFlow including its ability to deal with data exchange between computer systems running on different operating systems.

(c) Steps to use GenFlow - This provides a step-by-step guide for using the GenFlow program. The steps included are:

- Create/modify combination of computer systems - This provides the steps to create and/or modify the current combination of computer systems which are stored as the database structure of GenFlow. For example, if a new CAD package is added the procedures stipulated can be followed to include the newly-added system to the database structure.

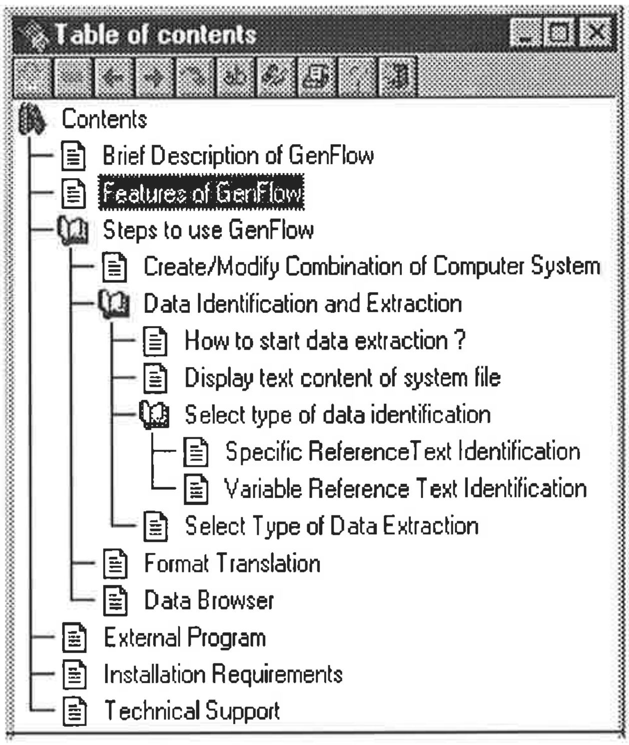


Table of contents

- Contents
  - Brief Description of GenFlow
  - Features of GenFlow
  - Steps to use GenFlow
    - Create/Modify Combination of Computer System
    - Data Identification and Extraction
      - How to start data extraction ?
      - Display text content of system file
      - Select type of data identification
        - Specific ReferenceText Identification
        - Variable Reference Text Identification
      - Select Type of Data Extraction
    - Format Translation
    - Data Browser
  - External Program
  - Installation Requirements
  - Technical Support

Figure 5.10 : Help Function

- Data identification and extraction - This topic explains the step-by-step procedures to carry out the data identification and data extraction processes. It includes the procedures to start data identification and data extraction, the files need to be specified in the process, the differences for Specific and Variable Reference Text Identification or Extraction and the steps to create the DEI files.
- Format translation - This topic explains the conversion of formats of datafiles to the format supported by the receiving system. Hypergraphics, which are graphics to view more related information,

are created to explain the functions of the the buttons and icons on the forms.

- Data Browser - This provides information concerned with the use of the data browser function to view data from datafiles in the common database.

(d) External program - This topic explains the use of external program such as UNIX2DOS which can be run within the GenFlow program.

(e) Installation requirements - This provides information related to the hardware requirements for using GenFlow.

(f) Technical Support - This explains the support provided for users who have problems in using the program.

5.3.7.2 General Windows Help Features

The GenFlow Help function has the "familiar features" of Windows Help functions as shown in Fig. 5.11.

These features include the system commands such as Search, and Glossary. The Search command enables the user to key in the first few characters of the keyword to be searched and the related keywords which start with the same characters will be displayed. For example, if the character "sp" is entered, the related topics such as "specify", "specific", "specifications" will appear. Glossary is like a small dictionary with the words arranged in alphabetical order. By clicking on a word a page of explanations appears. The details related to the features of these commands can be found in the built-in Help System for most of the Windows 3.x packages.

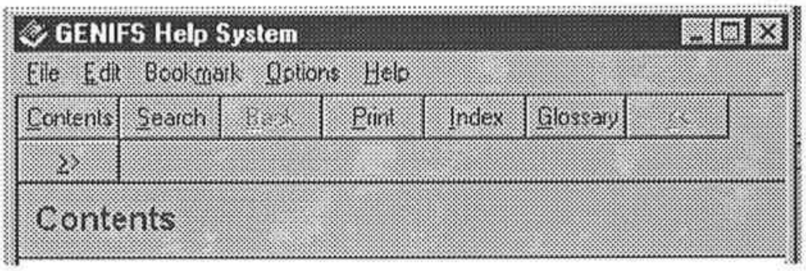

GENIFS Help System
File Edit Bookmark Options Help
Contents Search Back Print Index Glossary
>>
Contents

Figure 5.11 : GenFlow Help System

5.4 Summary

GenFlow was developed, based on the concept of GNSIFS, using Visual Basic, "C" and other third party programs. It is a program designed for the efficient data exchange among a group of computer systems which may be running on various operating systems. It runs within the Windows environment and the minimum hardware requirements are 386 CPU running at 33 MHz and 4 MegaBytes of RAM. GenFlow has the "look and feel" of a commercial Windows program with Menu Bar and Tool Bar. It is a program designed to be "user-friendly".

6. TEST RESULTS AND DISCUSSIONS

6.1 Introduction

In order to prove that the GNSIFS concept works, it was tested in three local organisations in South Australia, viz. Advanced Manufacturing Facility of South Australian Centre For Manufacturing (SACFM), a South Australian Architectural Firm called N.S. Architects, and Gerard Industries, an electrical components manufacturer. This Chapter describes the tasks carried out in the three companies followed by a discussion on the ramifications of using this program in industry.

6.2 Evaluation of GenFlow in SACFM

GenFlow was installed in June, 1994 at the Advanced Manufacturing Facility (AMF) of the South Australian Centre For Manufacturing (SACFM), which provides a rapid prototyping facility for the manufacturing industry. The AMF operates

- a Sintering Laser Station (SLS) rapid prototyping machine, model Sintering Station 2000, which runs on UNIX,
- a laminated object rapid prototyping machine, model LOMM 2030, which runs on Windows NT, and
- a digitbotic, non-contacts, laser scanning, coordinate measuring machine, which also runs on Windows NT.

6.2.1 Aims of GenFlow Evaluation in SACFM

Rapid prototyping machines can produce a working model directly from a CAD model in a few hours. The SLS, uses a laser to fuse layers of wax or nylon while in the LOMM layers of self-adhesive paper are laser cut and stuck together to form the object. The Laser Scanning Machine is able to check dimensions accurately and quickly without physical contact with components.

The quality of components produced by SLS and LOMM are affected by a number of factors including temperature, pressure and process time. The optimum settings for the production of components can only be determined by "trial and error" and in order to improve the quality of components, specific information in the database of these machines must be accessed and analysed. The datafiles are in text form and can be more than 20 pages long. However, only relevant data, which may be just a few lines of text, is required for analysis. To find and identify the required information is time-consuming and the machines cannot be used during this analysis. In this respect, it is important that GenFlow can easily and quickly retrieve the "selected" technical data and transfer them to other systems for further analysis.

In this application, GenFlow is required to:

(i) extract "selected" technical data from the three machines electronically,

(ii) transfer the extracted data to Excel for graph plotting and analysis, and

(iii) store the relevant technical data extracted from the three machines together with key dimensions of components, which comes from the

CAD system, and data analysis results from Excel, to a common database.

6.2.2 Procedure

The aim is to link the computer systems of the rapid prototyping machines, laser scanning machine, ComputerVision CAD and other packages together for data exchange (Fig. 6.1). Table 6.1 shows the input/output information for these systems.

Table 6.1 Input/output Information for the Computer Systems at SACFM

Name of Computer Systems	Input Information	Output Information	Operating System
Sintering Station 2000	Data Analysis Results	Technical Data	UNIX
LOMM 2030	Data Analysis Results	Technical Data	Windows NT
Laser Scanning Machine	Data Analysis Results	Data Analysis Results	Windows NT
ComputerVision Cadds 5	None	Key Dimensions of Components	UNIX
Excel 5.0	Technical Information, Quality Results	Data Analysis Results	Windows 3.x
Dimensional Checking Machine	None	Dimensional Checking Results	DOS

As can be seen from the table the systems have a range of operating systems such as UNIX, DOS/Windows and Windows NT. These are all common operating systems and a commercial network operating system, such as Netware, is able to handle the "networking" of these computer systems. As explained in Section 2.4, "networking" ensures the exchange of files but it does not deal with the exchange of a specific part of the file i.e. data. In this respect, GenFlow is required to enable the exchange of data among these computer systems. The schematic diagram of the integration of the systems is as shown in Fig. 6.1.

The system datafiles of the three machines are all in text form and examples of these are shown in Table 6.2.

The datafiles of the LOMM 2030 and the Laser Scanning machine have similar formats with the "=" sign in between the Reference Text and the data; e.g. "Scan Time", the Reference Text, is separated by "=" from the data, which is "00:59:08". However, the format adopted by Sintering Station differs from the other two machines in that a Blank Delimited Format (BDF) is used, e.g. "part-1-segment_sort_tolerance" is separated from the data "0.012700" by a Space.

The data format of datafiles from Sintering Station 2000 also adopts the separator "/" as shown in Table 6.2 and the text line with this separator is :

part-1-STL_filename /usr2/sls2k/geometry/stl/parts/_stl/nylon-warmup.bo22

For the format shown above, the Reference Text is "part-1-STL_filename " which indicates it is the first part (component) to be manufactured by the machine. A number of associate values follow, including "usr2", "sls2k"...etc. However, the important associate value is "nylon-warmup.bo22" which indicates the material, i.e. nylon, and

Table 6.2 Datafiles of the Machines at SACFM

Data format for Sintering Station 2000 is shown as below :

```
part-1-offset 0.000000 0.000000 0.000000
part-1-min_bounding_box -122.901437 -123.321709 0.000000
part-1-max_bounding_box 118.259599 122.995438 5.000000
part-1-parallel_segment_tolerance 0.000000
part-1-segment_sort_tolerance 0.012700
part-1-coplanarity_tolerance 0.000000
part-1-exposure_tolerance 0.000000
part-1-how_placed 2
part-1-STL_filename /usr2/sls2k/geometry/stl/parts/_stl/nylon-warmup.bo22
part-1-job_filename
.......................................................
```

Data format for LOMM 2030 is shown as below :

```
[System_Parameters]
version= 1.19.00
system_model= 20007
drive_type= GALIL
machine_type= 2030
preslicemode= 0
roundcorners= 1
numhatchregion= 1
burnoutmode= 0
xplatform= 16.000000
yplatform= 11.350000
.....................................
```

Data format for Laser Scanning Machine is shown as below :

```
[ScanStatus]
Scan Status=Complete
Scan Type=Adaptive
Scan Time=00:59:08
Est Scan Time=00:59:00
Section Num=100
Total Section Num=100
Section Z Value=103.34
Section Z Top=103.01
Num Section Points=32
.......................................
```

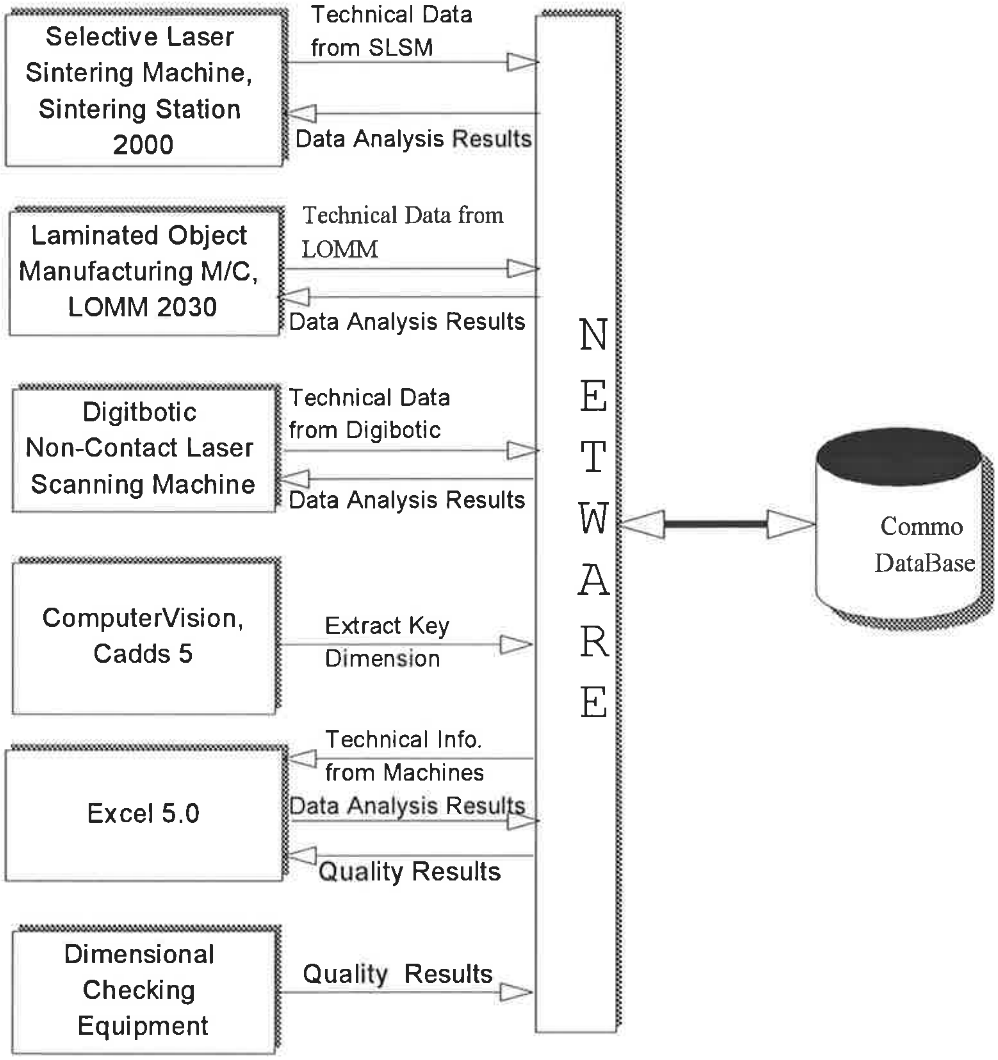


Figure 6.1 : Integration of Computer Systems in SACFM

the warm up condition. GenFlow enables the user to identify and extract the important data from these datafiles. The steps for using GenFlow in SACFM are described as follows.

(a) Install GenFlow on a PC which is linked to the machines on which data is stored.

(b) Run GenFlow and create the database structure which lists the combination of computer systems involved.

(c) Open the datafiles which contain the important data and identify the data to be extracted (See Section 4.4 for details of data extraction processes).

(d) Create a Data Extraction Information (DEI) file which stores information related to details of the types of data to be extracted. With the creation of the DEI File, extraction of data of the same type can be done without the need to go through the data identification process again (See Section 4.4).

(e) Create or select the component-related datafile, which resides in the common database and stores the important data extracted from the system datafiles of the three machines. The data stored in the component-related datafile is in the form of a neutral format and Table 6.3 shows the content of the datafile called “gearbox.dfl”.

Table 6.3 Content of “gearbox.dfl” in Common Database

```
### Rapid PT M/C -> Sintering Station 200 -> For Technical Data
"M-date","1993:12:14:15:40:11"
"Build height","259.384411"
"part-heater","198.000011"
"part-1-STL_filename" , “nylon-warmup.bo22”
### LOMM -> LOMM 2030 -> Technical Data
"xplatform","16"
"yplatform","11.35"
### Scanners -> Laser Scanning Machine -> Technical Data
“Scan Type”,"Adaptive”
“Scan Time”,"00:59:08"
“Est Scan Time”,"00:59:00"
“Section Num”,"100"
```

Note : The quote marks are always present to enable easy location of data during data transfer process

(f) Invoke the Browser function to view the content and perform copy-and-paste command to transfer the data (stored in component-related datafile in a neutral format) to Excel or another Spreadsheet programs such as DBase IV for graph plotting (See Section 5.3.6).

6.2.3 Results of GenFlow Evaluation at SACFM

The results and feedbacks of the evaluation are summarised below :

- The electronic data extraction from the system datafiles saves time and reduces human error during transfer of data. About one hour is saved for each component compared with the manual data extraction. Normally, one batch of components are built each day and that means a saving of at least one hour/day.
- The capability of exporting data from the component-related datafile to other software, such as Excel, allows the analysis of production and quality problems in a more effective way.
- Machine utilisation is increased due to the fast data collection and analysis.

It was requested that the existing company database program in use at SACFM called "sacfm.mak", which manages information related to customers, jobs and schedule details, should be able to view the data from the common database without invoking the GenFlow program. To respond to this request, a small program, which works in conjunction with “sacfm.mak”, was written to enable the retrieval of data from the common database to “sacfm.mak”. This is possible because the component-related datafiles are in neutral format and therefore the data is “readable” by “sacfm.mak”.

A brief cost analysis in this application shows that while GenFlow costs around $1500 the saving of just the 1 hr/day is valued at $15000 based on $60/hr and 250days/yr.

6.3 Evaluation of GenFlow in N.S. Architects

N.S. Architects is an architectural firm with 7 staff. During the past five years, most of their operations have been computerised to become an efficient and competitive company in the architectural area. The computerisation consists of :

- The drafting work, which was previously done on drawing boards, is now done on a 486 PC using AutoCADLT, which is a “simplified” Windows version of AutoCAD.
- Cordell Estimator, is used to estimate the cost of building material and building jobs (User Guide, 1995). This package runs under DOS.
- A customised database program, which is used to manage quotations, project schedules and accounting data, was developed from database package Microsoft Access 2.0.

6.3.1 Aims of GenFlow Evaluation in N.S. Architects

The aims of this test are :

- set up a common database in the hard drive of a 486 PC under the directory with name "c:\data" so that component-related datafiles with information shared by other systems are to be stored under this directory (hard drive of any PC with free disk space of at lease 50 Mbytes can be assigned as the location of common database).
- extract key dimensions from the AutoCAD drawing and store this information in a common database.
- extract the estimated costs data from Cordell Estimator and also store them in the common database.
- transfer the cost data, scheduling and accounting data form the database program "ns.mdb" to the common database.
- load cost data into an Excel spreadsheet to give cost estimates and to produce graphs and forecasts.
- call up the AutoCAD dimensional data, which is stored in common database, from Browser to view the dimensionsl variations of various projects for comparison purpose.
- produce quotations with the imported data from the common database using Word 6.0.

6.3.2 Method of GenFlow Evaluation in N.S. Architects

There are three IBM-Compatible Personal Computers (PCs) in the company running DOS/Windows programs. Five packages are used in the company and Table 6.4 shows their input/output information.

Table 6.4 Input/output Information of Computer Systems at N.S. Architects

Name of Computer System	Input Information	Output Information	Operating Platform
AutoCAD LT	Cost Data	Dimensional Data	Windows 3.x
Cordell Estimator	None	Cost Data	DOS
Access 2.0	Cost Data, Schedule	None	Windows 3.x
Excel 5.0	Cost Data	Accounting Data	Windows 3.x
Word 6.0	Accounting Data	Accounting Statements	Windows 3.x

As all the packages run under Dos/Windows, Windows For Workgroup (WFW) was the appropriate networking program (Microsoft Corporation, 1994). The manuals with WFW provide detailed information about the linking procedures. WFW is a "peer-to-peer" networking system which means that there is no separate to serve as a common database. In view of this, the common database was located in a new directory e.g. "c:\data", of the hard drive of the PC with the largest storage capacity. Fig. 6.2 shows the schematic diagram of the integration of the five packages via Windows For Workgroup (WFW). The steps to prepare the network for data exchange are :

(a) Install GenFlow on all the three PCs so that viewing of information and data transfer operations can be performed on any of the machines in the firm.

(b) Extract key dimensions from AutoCAD drawing using the built-in function **Attribute** or by writing a program in AutoLISP (Zockel & Lau; 1990).

(c) Run GenFlow and create the common database structure which lists the computer systems involved as AutoCAD, Cordell Estimator, Access 2.0, Excel 5.0 and Word 6.0.

(d) Identify and extract data to be stored in the common database. An extract for one of the datafile is shown below :

```
### Estimator -> Cordell -> cost estimates
"Carpentry","2584.71"
"Joinery","5000"
“Electrical", "2001"
"Total Building Cost","234444"
### CAD -> AutoCAD LT-> Dimensions Extraction
“width of master-room","3.34"
“length of master-room","4.45"
“height of ceiling","3.7"
“width of garage","2.8"
“length of garage","3.4"
“width of room1","2.7"
...............................................
```

(e) With the help of GenFlow, view data using the Browser function and paste the data to Excel (or other Spreadsheet program such as Paradox) for further processing and plotting (See Section 5.3).

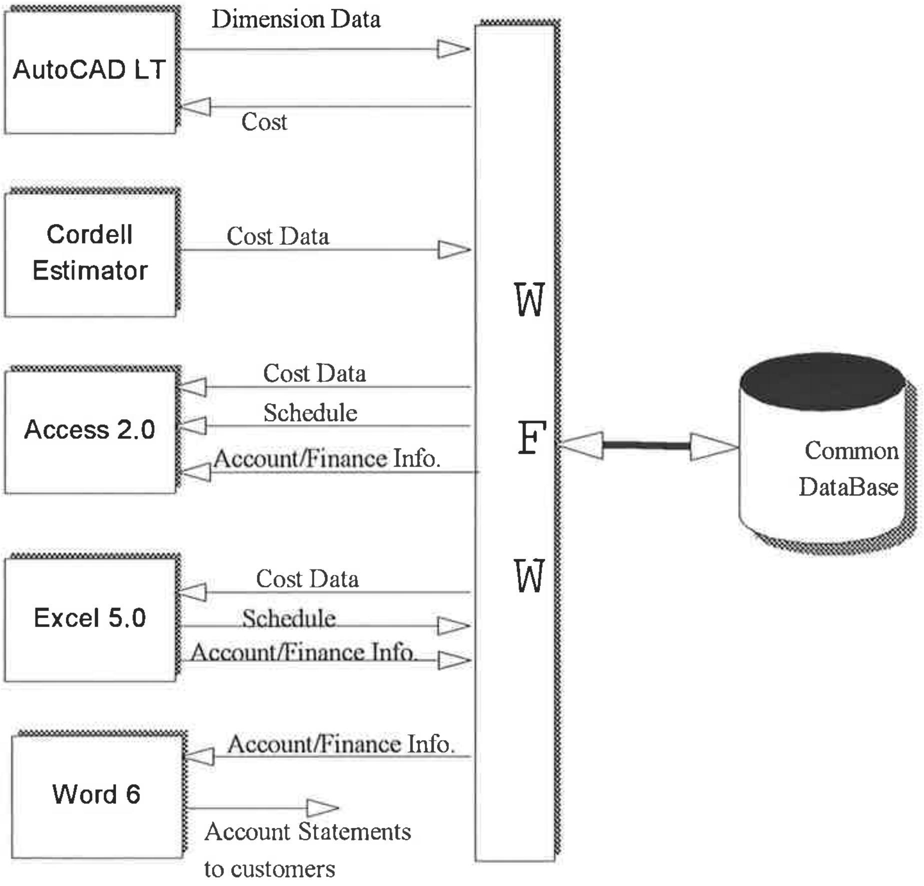


Figure 6.2 : Integration of Computer Systems in N.S. Architect

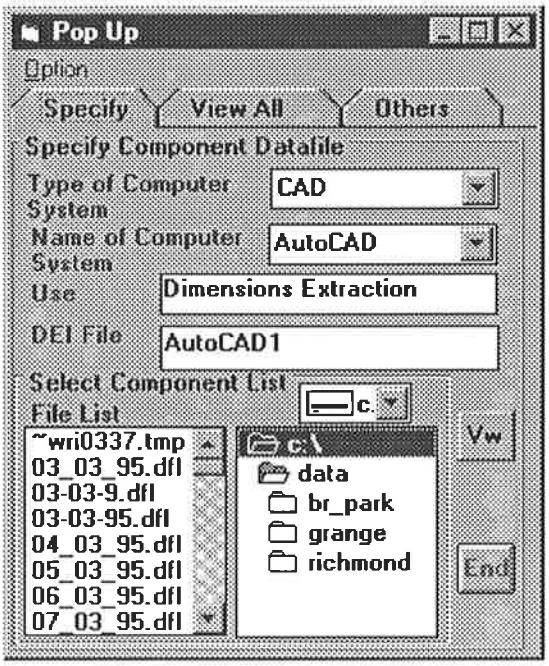
6.3.3 Results of GenFlow Evaluation in N.S. Architects

N.S. Architects have been using GenFlow since June, 1995, with the following results :

- The electronic extraction of key dimensions from AutoCAD drawing saves time and eliminates human errors.
- The information, such as cost data, key dimensions and scheduling, which has to be collected from various programs, can now be viewed together on one screen. This shortens the lead time for building project development.

- Quotations to customers can now incorporate electronically-transferred data from the Cordell Estimator via the common database. This shortens the time to complete a quote from about twenty-two hours to two hours i.e. a saving of 20 hours. There are roughly 100 quotes to make every year and that means a saving of 2000 hours. At an hourly rate of $70 this means an annual saving of $140,000 without considering the hidden benefits of more acurate and better quality quotations.
- The analysis of data imported from various systems into Excel can provide numerical or graphical comparisons of building costs for different projects and different materials.

The firm suggested that the designer should be able to view and enter data without the need to run GenFlow. This is important when designer wants to enter data to the datafiles in the common database during the design work within AutoCAD. Although GenFlow enables the user to extract text data from the drawing, the designer may prefer to type in the data directly during the design process without invoking GenFlow. Consequently, a program was developed which when run, appears on top of the AutoCAD drawing (Fig. 6.3) to enable the designer to enter data as desired. To run the program a "shortcut key" e.g. <Alt-Y> is used for simplicity. In addition, this window can be made to stay "always on top" of the AutoCAD drawing by selecting **Always on Top** under the **Option** menu of Fig. 6.3. The program was written in Visual Basic and the user can specify the computer systems and the datafiles from the



Specify Component Datafile

Type of Computer System: CAD

Name of Computer System: AutoCAD

Use: Dimensions Extraction

DEI File: AutoCAD1

Select Component List

File List

~wri0337.tmp
03_03_95.dfl
03-03-9.dfl
03-03-95.dfl
04_03_95.dfl
05_03_95.dfl
06_03_95.dfl
07_03_95.dfl

c:

c:\
data
br_park
grange
richmond

Figure 6.3 : Pop Up Window to Access Database

built-in functions. This feature is also useful when the designer wants to view the cost data from the Estimator in order to make a decision on which material is to be used.

6.4 Using of GenFlow at Gerard Industries

Gerard Industries is a manufacturing company with about 1400 employees producing electrical switches and other electrical products. The company has an IBM main-frame with terminals in various departments. However, this main-frame system is considered to be obsolete as are most of the programs running on this system. Also, the operating system is an IBM proprietary system which cannot run DOS or Windows programs. There are about 10 PCs in the company with no network connection.

6.4.1 Quality Problems at Gerard Industries

The main concern of the company is quality control. There are several hundred types of brass strips with different dimensions and shapes for various products, and the accuracy of these strips is critical to the functionality of the products. The operators check the dimensions with gauges and record the results manually for later collection by the QC department. The dimensions of gauges are checked regularly with a Co-ordinate Measuring Machine (CMM) and the results recorded in a text file. As there are hundreds of gauges, and all these operations were carried out manually, mistakes occur frequently.

6.4.2 Three Stages of Implementation for Quality Improvements

To tackle these quality problems, a three stage proposal was suggested.

First Stage To ensure the accuracy of gauges a Data Recording System (DRS), which is a database program developed from Access or DBase, can be used to record the calibration data from the CMM. GenFlow can link this program with Excel or other software applications to help analyse the data and provide a detailed record of the calibration data. Only one PC is required for this stage. If a PC is already available, the investment required is minimal.

Second Stage If the first stage is successful, the installation of Netware should be considered, to greatly enhance the communication capability within the plant. A second Data Recording System (DRS) can be used to keep records of the dimensional checks on the shopfloor. GenFlow can link the data to other programs for data analysis. The installation of Netware is the biggest investment in this phase amounting to about $6000 for 50 users. Cost of 30 PCs will be around $90000 based on $3000/PC. The total investment will be approximately $96000.

Third Stage With the Netware connection, it is possible to link additional computer systems such as design, production and store through GenFlow and QC personnel will be able to obtain updated information from other departmental areas quickly and accurately.

The complete implementation of these three stages provides the data exchange of the combination of computer systems as shown in Fig. 6.4, and Table 6.5 shows the input and output information for individual computer systems. However, due to the tight budgetary control of the company, Gerard Industries only agreed to implement the first stage; the second and third stages were put on hold until further funding becomes available.

6.4.3 Procedures for Implementing Stage One

The procedures are summarised below :

(a) A DRS was developed using Visual Basic with a built-in database management routine to keep data of the gauges checking/calibration results.

(b) Installed GenFlow on a PC in the QC Department.

(c) GenFlow was used to extract the data from the CMM file and transfer the data to the DRS. This greatly enhanced the efficiency for the dimensional control of gauges.

(d) Transfer extracted data to Excel or Lotus 1-2-3 for data analysis and graph plotting.

Table 6.5 Input/output Information for the Computer System

Name of Computer System	**Input Information**	**Output Information**	**Operating Platform**
CMM Machine	None	Guage Calibration Data	Manufacturer's Own System
AutoCAD	None	BOM, Dim. data	DOS
Data Recording System (DRS)	Dimensional Data	Technical Data	Windows 3.x
Excel 5.0	Technical Data	Data Analysis Results	Windows 3.x
Word 6.0	None	Reports	Windows 3.x

6.4.4 Feedback for Using GenFlow at Gerard Industries

- The accuracy of gauges can be controlled more efficiently and any dimensional problems with gauges, once detected by CMM , is flagged on the DRS so that prompt corrective action can be taken.
- The possibility of transferring data to Excel for graph plotting and data analysis greatly enhances the efficiency for quality control of components.

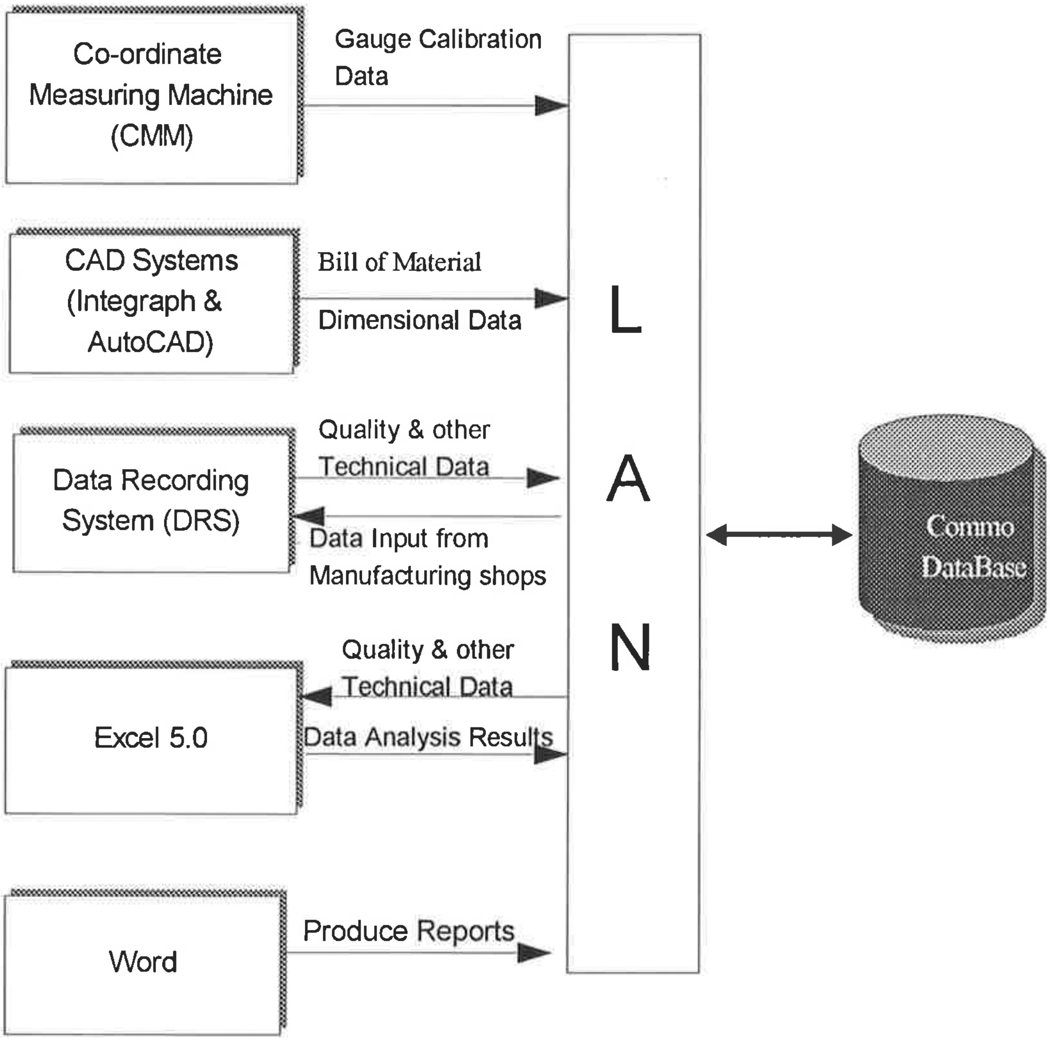

Co-ordinate Measuring Machine (CMM)
Gauge Calibration Data
CAD Systems (Integraph & AutoCAD)
Bill of Material
Dimensional Data
Data Recording System (DRS)
Quality & other Technical Data
Data Input from Manufacturing shops
Excel 5.0
Quality & other Technical Data
Data Analysis Results
Word
Produce Reports
LAN
Commo DataBase

Figure 6.4 : Integration of Computer Systems in Gerard Industries

6.5 Cost Saving in Developing Conversion Programs

Apart from the benefits as mentioned previously, the use of GenFlow eliminates the need to create specific conversion programs between any two systems. This is important for companies such as Motorola (See Section 2.7) which has developed specific conversion programs to link all the computer systems.

Basically, only two conversion programs are required for any particular system, i.e. from system format to neutral format and from neutral format to system format (See Section 4.2). Once these two programs are developed, the system can link to any other system provided that the other system has the two conversion programs as well. For example, if there are four systems, then eight (number of systems x 2) conversion programs are required. However, if each system has to be linked with all others in both directions then the number of conversion programs required are 12 (number of systems x (number of systems-1)) i.e. four conversion programs are saved. This number increases as more systems are involved. It should be noted that savings are realised only when more than three systems (packages) are used.

Due to the complexity of conversion programs, professional programmers are usually required to develop these programs. This is not the case with GenFlow since the format conversion in GenFlow is done "transparently" in a generic way using the GFT so that no additional programming is necessary (See Section 4.6). Table 6.6 shows the basic savings incurred when GenFlow is used comparing with the traditional format conversion technique.

The calculation of saving is based on :

(i) Estimated time to develop a conversion program between any two systems is 5 hours

(ii) Cost for a programmer is estimated at $100/hr, i.e. $1000 for bidirectional data exchange between any two systems

(iii) Cost of a site licence for GenFlow program is estimated at $1500

Table 6.6 Savings in Developing Conversion Programs

Number of Systems	Cost of Individual Programs	GenFlow Program	Saving
2	1000	1500	-500
3	3000	1500	+1500
4	6000	1500	+4500
5	10000	1500	+8500
6	15000	1500	+13500

6.6 Summary and Discussions

The tests of GenFlow in the three organisations including SACFM, N.S. Architects and Gerard Industries were most useful in evaluating the capabilities, uses and limitations of GNSIFS. The results indicate that GenFlow can play an important role in bringing together existing computer systems used in real companies to achieve data exchange among these systems, regardless of the operating systems.

The Advanced Manufacturing Facility of SACFM is equipped with advanced manufacturing machine tools and the test results with this organisation indicate that GenFlow is able to work with these advanced machines for two reasons; firstly, the advanced machines use the popular operating systems like Windows NT and UNIX; and secondly, the computer systems of these machines produce detailed technical data in text form from which GenFlow can efficiently extract data.

At N.S. Architects, GenFlow is able to transfer data between all the existing computer systems without the need for any hardware or software change or modification. Test results found in N.S. Architects suggest that GenFlow can be used without significant capital outlay and can substantially improve productivity.

In a large company like Gerard Industries with departments in various locations, it is important to achieve detection of defective components at the earliest moment so that corrective action can be taken. GenFlow was useful to link departments and enables faster data communication between them. Although only stage one was implemented, the results indicate that GenFlow can be used to extract data from computer system, such as CMM, and transfer them to other program, e.g. Excel, for analysis. There is no doubt that if the computers "are networked", as stipulated in Stage Two, the whole operation will be more efficient.

In general, the evaluation results show that GenFlow, which is estimated at $1500 is affordable and cost effective even in small companies. The fact that it can be "configured" to suit existing groups of computer systems in a company means that no computer hardware or software need to be modified or replaced.

7. CONCLUSIONS AND FURTHER RESEARCH

7.1 Conclusions

GNSIFS is an information management philosophy, aimed at achieving data exchange among various computer systems in a cost-effective way. With the generic methodology and the necessary tools, an information flow system can be designed and implemented to realise the viability of the concept. The establishment of a generic methodology is necessary because the tools, e.g. computer languages and third-party macros, available in the market change rapidly. Results and feedbacks from three organisation indicate that GNSIFS can be introduced easily and can contribute to the productivity and quality output of a company. The main conclusions are as follows.

7.1.1 The Neutral Format is an Essential Element of Efficient and Economical Data Exchange

The use of neutral format is important because it is "system-independent" (See Section 1.2) and only two conversion programs are required for any particular system, i.e. from system format to neutral format and from neutral format to system format. Once these two programs are developed, the system can link to any other system provided that the other system also has the two conversion programs. GNSIFS performs the neutral format conversion "transparently" and therefore no conversion programs need to be developed. However, companies can develop conversion programs for the computer systems for particular purpose.

7.1.2 The Source Files for All Data Must be Available in ASCII for GNSIFS to Work

The pre-requisite for efficient data exchange among computer systems, which are running in different operating systems such as UNIX, DOS/Windows, OS/2 and Macintosh, is that the data is in ASCII format. This is the only constraint of GNSIFS although ASCII is a "cross-platform" standard and therefore it is supported by all operating systems.

7.1.3 The Generic Non Standard-specific Concept is Viable

The concept of achieving data exchange by constraining the data to be in neutral text format does work with all operating systems as all operating systems support ASCII text format. The generic nature of the system is cost effective as it allows the user to "configure" the information flow system to suit existing computer systems of the company without the need to modify or replace existing hardware and software.

7.1.4 Additional Programming

Apart from the capability to link proprietary packages such as AutoCAD and SmartCAM for data exchange, GenFlow can also integrate customised databases such as those developed with Access and Visual Basic. However, special programs may be required depending on the customised databases. This was the reason for special programs to be written for the Visual Basic customised database "sacfm.mak" at SACFM (Section 6.2.3) and the Access customised database "ns.mdb" at N.S. Architects (Section 6.3.1).

7.1.5 Improve Productivity through Better Information Control

Apart from the non-tangible benefits achieved by using GenFlow, such as fast and accurate data collection and analysis and improved productivity, quantifiable cost benefits can also be achieved through better information control.

7.1.6 No Special Computer Skill needed to Use the System

GenFlow is an "user-friendly" program and it can be used by staff without specialised computer knowledge. Once it is "configured" by system engineer, it can be used by other staff with minimal training.

7. 2 Further Research

The area which requires further research is the use of Knowledge-Based System (KBS) to complement the performance of GNSIFS. The implementation of GNSIFS requires the correct selection of data which is to be stored in the common database. A KBS should be able to be designed to help the user set up GNSIFS more easily and guide the user to select the necessary data.

The other area of further research is related to the enhancement of the GenFlow package as below :

- Most of the icons can be re-designed so that it is easier for the user to understand their functions like the icons of some commercial packages such as Excel and Word. In

GenFlow, the icons were selected from the icon libraries provided by Visual Basic and third party programming tools. Study related to the artistic and graphical design of the icons are necessary so that the unique functions of the icons are "represented" by the pictures on them.

- More "bullet-proof" features, which protect the program from being corrupted or having a "clash" when the user keys a wrong button or click on a wrong spot, can be added to GenFlow. This requires "large scale" site testing by different users of different industrial and commercial fields so that reliable and realistic results can be collected to determine the details of the "bullet-proof" requirements.
- The new Windows feature set by Microsoft, e.g OLE 2, can be added to the program. With the OLE 2 capability, the user is able to select all the functions and commands of a program, e.g. Excel, even when the spreadsheet table is within GenFlow environment. For example, if the user calls up the Browser function to view the table which contains data retrieved from the common database, OLE 2 enables the user to perform calculation and plot graph directly from the table, while staying in GenFlow, i.e. without jumping to Excel.

8. REFERENCES

Ahmed, Shamin; Wong, Albert; Sriram, Duvvuru; Logcher, Robert. 1992. Object-oriented database management systems for engineering; a comparison.. *Journal of Object-oriented programming*, June, v5, P.27.

Ang, Michael. 1986. Computer Integrated Manufacturing -- the Reality. *Seminar on Advances in Automated Manufacturing Techniques.* Singapore.

AutoLISP Reference Manual. 1989. Release 9.

Babb, Michael. 1985. Communications Protocol Converter Allows Networking of Incompatible Factory Devices. *Control Engineering*, September. P.149

Barrett, Glen. 1993. Working Through the Data Exchange Maze. *Technical Computing*. The ACADS Journal. No. 79.

Barry, John. 1992. The world of object-oriented DBMSs. *DBMS*, June,,v5,p.53.

Bell, David & Grimson, Jane. 1992. *Distributed Database Systems*. Addison-Wesley Computer Science.

Biegel, Richard A & Pendharkar, Sumant S. 1994. Data Exchange - Spreadsheets and Databases. *Data Bases Advisor,* 60-67, May.

Bohn, J.H. 1996. *Introduction to Rapid Prototyping (ISO-10303 STEP).* Randolph Hall.

BPCS for Manufacturing, 1987.

Browning, Dave. Relational Databases - New Blood New Power. 1993. *PC Magazine*, May,

Buddenhagen, David. Designing a Production System Database. *Synergy '86 Conference Proceedings*, California.

CAD Conferencing. 1993. *Technical Computing*. The ACADS Journal, No77, May, p.24-27.

CAD/CAM planning guide. 1988. Machine design, October, P.36- 52.

CADDSMAN reference manual.1989.

CATIA reference manual. 1988.

Chandler, J.R. & Hobson, D. 1985. Connecting for FMS -- experiences in the building of an educational system. *Computer-aided Engineering Journal*, February P.13-20.

Citidata, The Corporate Resource. 1994. Citidata Pty. Ltd.

Cole, J.E. 1989. Design-to-manufacture -- an integrated information system. *Computer-Aided Engineering Journal*, October.

Coombes, Grahan. 1987. An Integrated Data Management Environment for CIM. *Autofact '87 Conference Proceedings*, Michigan.

CorelDraw users Manual. 1993.

Data, C.J. 1981. *An Introduction to DataBase Systems.* Addison- Wesley Publishing Company.

Day, J.D. & Zimmermann, H. 1983. The OSI Reference Model. *Proceedings of IEEE*, 71.

DBase IV language reference. 1991. Ashton-Tate.

Derus, Kenneth . 1986. Integrated Database: A Necessary Component of CIM. *Synergy '86 Conference Proceedings*, California.

Dodge, John. 1993. CA's Windows-based tool melds cross-database queries, reports. *PC Week*, 1-2, November.

Douglas, Bruce. 1992. Data Exchange Standards. *Technical computing.* The ACADS Journal, No 71,12-14.

ECRC. 1996. (http://www.ecrc.ctc.com/techhub/inter).

EDS Unigraphics. 1995. EDS (http://www.ug.eds.com/ug/translator/step.html).

Everitt, P.B. 1988. Product Data Exchange Specification: RAMP's Unique Implementation of the PDES Specification. *SME Technical Paper*, MS88-660.

Fairall, J. 1992. Editorial. *Technical computing.* The ACADS Journal, No 71,

Ferme W.D. 1993. CIM in Australia (1,2) : Ten Years On. *Australian Conference on Manufacturing Engineeering.* P115-126.

Flack, Russell. 1985. MRP-II & JIT - Which is Better Value ? *SME Technical Paper*.

Fritz, Dan. 1987. *Use Your Personal Computer for Direct Numerical Control.* Owner Suburban Machinery, Inc. Ohio.

Froyd, S.G. 1988. Connecting NC Machines to MAP. *Proceedings of Enterprise '88*. Maryland.

Gavron, Jacquelyn & Lefkowitz, Larry. 1994. The Big Three Suite Success. *Windows Sources Australia,* 66-71, June.

Glen, Ron. 1992. The great new database debate. *Canadian Datasystems*, April, p.23.

Green, J.H. 1988. *Local area network - a user's guide for business professionals*. Scott, Foresman and Company, Glenview, Illinois, London, Computer Book Centre Pte Ltd, Singapore.

Greenbaum, Joshua M. 1993. SAP's R/3 Software Suite: Architecturing Open, Integrated Software for Distributed Enterprises. *Open Information System,* 3-16, July.

Harris,T. R. 1985. Local area networking: what can we do until MAP is completed, and how to complement it ? *SME technical paper*, MS85-1065.

Hill, C. 1987. *The AutoCAD Productivity Book*. NC, Ventana.

House, Bill. 1991. Comparing the greats (comparing database management systems. *Data Based Advisor*, p.170.

HP in Manufacturing. 1990. Hewlett Packard.

IBM Application System/400 - connectivity. 1992. IBM.

IMPCON manual, 1988.

Inglis, A.E. 1993. Implementing CIM in a Networked PC Environment. *Australian Conference on Manufacturing Engineeering*. P127-134..

ISO. 1996. Welcome to ISO Online. (http://www.iso.ch/)

Kermit User Guide. 1988. Trustees of Columbia University in the City of New York.

Kesslet, G. & Shepard, S. 1994. *A Primer On Internet an TCP/IP Tools*. Hill Associates, Inc.

Khoshafian, Setrag. 1991. Modeling with Onject-oriented databases. *AI Expert*, Oct, p.26.

Lau, H. & Zockel, M. 1992. An Economical Database Management System for CIM. *1992 International Conference on Manufacturing Automation*, Hong Kong in October.

Lau, H. & Zockel, M. 1994. A Generic Non Standard-specific Rationalised Information Flow and Storage System for Computer-based Manufacturing. *Proceedings for Industrial Automation '94 Conference*, Singapore.

Leng, L.K. 1986. Computer integrated manufacturing: towards manufacturing excellence. *Seminar on Advances in Automated Manufacturing Techniques*, Singapore.

Leonard, Michel. 1992. *Database Design Theory*. MacMillan Education Ltd. London.

Lim, K. 1986. Computer integrated manufacturing -- the reality. *Seminar on Advances in Automated Manufacturing Techniques*, Singapore.

Lloyd, Alan & Rozmus, Greg. 1991. *OSI Application Essentials*. TAFE, NSW.

Loomis, Mary. 1992. Object versioning (ODBMS). *Journal of Object-oriented programming*, Jan, p.40.

Luca, A.D. 1993. A Step by Step Effective Approach to Computer Integrated Enterprise. *Australian Conference on Manufacturing Engineeering*. P109-113.

MAP information packet. 1986. General Motors.

Markowitz, V.M. 1994. Safe Referential Integrity and Null Constraint Structures in Relational Databases. *Information systems.*

MENTAT/MFG an overview,1988.

Microsoft Corporation. 1993a. *Professional Features Book2.* Microsoft Visual Basic.

Microsoft Corporation. 1993b. *Language Reference.* Microsoft Visual Basic.

Microsoft Corporation. 1993c. *Programmer's Guide.* Microsoft Visual Basic.

Microsoft Corporation. 1993d. *Professional Features Book1.* Microsoft Visual Basic.

Microsoft Corporation. 1994. Windows for Workgroup Manual.

Midrange Systems. 1993. Data Extraction, 57, November.

Miller, A.R. 1988. *The ABC's of AutoCAD*, Synbex.

Millett, B.C. 1990. Neutral Manufacturing Data: The Gateway to CIM. *1990 Pacific Conference on Manufacturing Proceeding*, Vol 1, 534-541.

Mo, John. 1993. On the Aspects of Robot Control through RS-232 Interface. *Proceedings of 1993 Australian Conference on Manufacturing Engineering*. Adelaide.

Mult, H,C. & Hans, I. 1991. Advanced Manufacturing to Achieve World Competitiveness. *Paper for Workshops Manufacturing Australia*, July.

Nance, Barry. 1992. *Introduction to Networking*. QUE.

Navathe, Shamkant B. 1992. Evolution of data modeling for databases. *Communications of the ACM*, Sept, V35, p.112-124.

Newton, M.J. & Hurst, K.S. 1988. CAD data communications -- an update. *Computer-Aided Engineering Journal*, February.

Nicoletti, G.M. 1987. Computer-integrated manufacturing (CIM), manufacturing automation protocol (MAP), and automated mass finishing systems. *SME technical paper*, MS87-158,

Nicoletti,G.M. 1986. CIM: LAN communications, protocol standards, and real-time control. *Automach Australia Conference proceedings*, May 26-29, Sydney, Australia

Norton User Guide, 1993.

Olson, Randy & Bahrololoomi, Ali. 1988. MAP Broad Specification. *MAP/TOP Users Group Summary. Vol. 3.* California.

Price, Gwyn & Gray, Alec. 1994. *Object Oriented Databases & their Application to Historical Data.* University of Wales College of Cardiff.

Quisumbing, Gary & Le, He. 1994. *The Open System Interconnection (OSI) Architecture*. Centre for the New Engineer at George Mason University.

Raker, D. & Rice, H. 1990. *Inside AutoCAD*, CA, New Riders.

Richardson, Douglas. 1987. Factory Control Using MAP. *Proceedings of CIMTECH '87*. California.

Rui, A., Weston, R., Gascoigne, J., Hodgson, A., Sumpter, C. 1988. Automating information transfer in manufacturing systems. *Computer-Aided Engineering Journal*, June.

Sangoma. 1995. (http://www.sangoma.com/index.html)

Sargent, P.M. 1990. Materials Data Interchange for Component Manufacture. *Engineering with Computers*, Vol. 6, 237-247.

Schumner,M. 1988. CNC/DNC Communication with MAP. *Enterprise '88 Conference*, June, Baltimore, Maryland.

Simpson, Alan. 1990. *Understanding dBase IV 1.1*. Sybex.

Snodgrass, B. Neil. 1984. Templates for an Integrated Common Database. *AUTOFACT 6 Conference Proceedings*, California.

Stamper, David & Price, Wilson. 1990. *Database Design & Management - An Applied Approach*, Mcgraw-Hill.

Strehlo, Kevin. 1991. The OODBMS Cutting Edge. *DBMS*, Nov, p.8.

Symcoxs, G & Adey, R. 1986. MAP and its affect on European manufacturing. *Automach Australia Conference proceedings*, May 26-29, Syndey, Australia.

Thomas, R. M. 1988. *Advanced Techniques in AutoCAD*. Synbex.

User Guide. 1995. *Cordell Estimator*. Australia.

Venditto, Gus. 1994. Novell Plans to Rule Windows. *Windows Sources Australia,* 17-18, June.

Zockel, M. & Lau, H. 1990. Development of a generic link between CAD and MRP-II systems. *Pacific Conference on Manufacturing Proceeding*, Vol 1, 512-518.

Windows Sources Australia, May, 1995.

APPENDIX A

A.1 Source Codes for LOGIN Function

```
Declare Function GetFocus Lib "User" () As Integer
Declare Function GetWindowLong Lib "User" (ByVal Hwnd%, ByVal nIndex%) As
Long
Declare Function SetWindowLong Lib "User" (ByVal Hwnd%, ByVal nIndex%,
ByVal dwNewLong&) As Long
Declare Function SendMessage Lib "User" (ByVal Hwnd%, ByVal wMsg%, ByVal
wParam%, ByVal lParam&) As Long
Declare Function SetSysModalWindow Lib "User" (ByVal Hwnd%) As Integer

Const WM_USER = &H400
Const EM_SETPASSWORDCHAR = WM_USER + 28
Const ES_PASSWORD = &H20
Const GWL_STYLE = -16

Sub Cancel_Click ()
      Unload passlockform
      accesslevel = "low"
      mdimenu.Show
      'datagrp.Show
End Sub

Sub cmdOK_Click ()
   Static Count
   Dim curindex As Long
   Dim s As String
   Dim strsearch As String
   Dim wordlen_str As String
   Dim endquote_str As String

   On Error GoTo error_passlock
   decipher "c:\data\rootpass.txt", 111
   Open "c:\data\rootpass.txt" For Input As #1
   Input #1, a$, b$
   Close
   RootPassword$ = Trim$(a$)
   NormalPassword$ = Trim$(b$)
   cipher "c:\data\rootpass.txt", 111

   Select Case text1.Text
      Case RootPassword$
        securityform.Show

      Case NormalPassword$
        'password = text1.Text
```

```
            Unload passlockform
            accesslevel = "high"
            mdimenu.Show
            'datagrp.Show

        Case Else


           Count = Count + 1
            If Count > 3 Then
              MsgBox "Wrong password! Sorry, but you've tried too many times!", 16
              Unload passlockform
            Else
              MsgBox "Wrong password! Try again, but remember, you" + Chr$(13) +
                  Chr$(10) + "only have" + Str$(3 - Count) + " tries left!", 48
              text1.SetFocus
              text1.Text = ""
              text1.SelStart = 0
            End If

      End Select
   Exit Sub

   error_passlock:
      MsgBox errormsg$(errornumber%), 48, "Error"
      Resume end_passlock
   end_passlock:

   End Sub

   Sub exitbtn_Click ()
      End
   End Sub

   Sub Form_Load ()
      cur_form = "passlock"
      Dim Hwnd As Integer     ' window handle for the control
      Dim StyleFlags As Long  ' window style for the control

      ' we have to show form before setting focus on text box
      Show

      ' set focus on the text box
      text1.SetFocus

      ' get text box's hWnd
      Hwnd = GetFocus()

      ' get current style flags
```

```
    StyleFlags = GetWindowLong(Hwnd, GWL_STYLE)

    ' set the password style
    StyleFlags = StyleFlags Or ES_PASSWORD

    ' and change the style flags
    StyleFlags = SetWindowLong(Hwnd, GWL_STYLE, StyleFlags)

    ' send message indicating character to print (Chr$(42)=*)
    StyleFlags = SendMessage(Hwnd, EM_SETPASSWORDCHAR, 42, 0&)

    ' lock this form as a system modal window
    'I = SetSysModalWindow(PassLockForm.Hwnd)

    'Open "c:\data\rootpass.txt" For Input As #1
    'Input #1, a$, b$, c$, d$, e$, f$, g$, h$, i$, j$, k$
    'Close


    check_depathfile


End Sub



Sub Text1_Change ()
    ValidateOkButton text1, 1
End Sub

Sub ValidateOkButton (currentcontrol As Control, boxcount As Integer)

    Dim count_value As Integer
    Dim BoxText As String
    ReDim Ok_Flag(boxcount)   As Integer
    Dim index

    BoxText = currentcontrol.Text
    index = Val(currentcontrol.Tag)
    If BoxText <> "" Then
        Ok_Flag(index) = True
    Else
        Ok_Flag(index) = False
    End If

    For count_value = 1 To boxcount
      If Ok_Flag(count_value) = False Then
        cmdOk.Enabled = False
        Exit Sub
        End If
    Next count_value
    cmdOk.Enabled = True

End Sub
```

A.2 Source Codes for Data Group Function

```
Dim activerow As Integer
Dim selgridtext As String

Sub addbtn_Click ()
    Dim filenum As Integer
    filenum = FreeFile

    checkname = Dir$(defapath + "dbstruct.txt")
    If checkname = "" Then
        MsgBox defapath + "dbstruct.txt" + " NOT FOUND ! Use Create Button.", 48,
"Error"
        Exit Sub
    End If

    Open defapath + "dbstruct.txt" For Input As #filenum
    filesize = LOF(filenum)

    linesfromfile$ = Input$(filesize - 2, filenum)
    newline1$ = Chr(34) + Trim$(typecsystem.Text) + Chr(34) + ","
    newline2$ = Chr(34) + Trim$(softwarename.Text) + Chr(34) + ","
    newline3$ = Chr(34) + Trim$(txtUse.Text) + Chr(34) + ","
    InfoFN$ = AssignInfoFN(Trim$(softwarename.Text))'Assign a DEIF Name with no
        extension
    newline4$ = Chr(34) + InfoFN$ + Chr(34)
    newline$ = newline1$ + newline2$ + newline3$ + newline4$ + Chr(13) + Chr(10)
    totallines = linesfromfile$ + newline$
    Close #filenum

    filenum = FreeFile
    Open defapath + "dbstruct.txt" For Output As #filenum
     Print #filenum, totallines
    Close #filenum

    dbstructgrid.Row = activerow

    dbstructgrid.Col = 1
    dbstructgrid.FixedAlignment(0) = 2
    dbstructgrid.Text = typecsystem.Text

    dbstructgrid.Col = 2
    dbstructgrid.FixedAlignment(1) = 2
    dbstructgrid.Text = softwarename.Text
    activerow = activerow + 1

    clear_combos
End Sub

Sub clear_combos ()
    typecsystem.Text = ""
    softwarename.Text = ""
    txtUse.Text = ""
End Sub

Sub clear_griddisplay ()

    no_of_systems = numberlines(defapath + "dbstruct.txt")
    activerow = 1

    Do Until checkcount% = no_of_systems

        dbstructgrid.Row = activerow

        dbstructgrid.Col = 1
        dbstructgrid.FixedAlignment(0) = 2
        dbstructgrid.Text = ""

        dbstructgrid.Col = 2
        dbstructgrid.FixedAlignment(1) = 2
        dbstructgrid.Text = ""
        activerow = activerow + 1

        checkcount% = checkcount% + 1
    Loop
    Close


End Sub

Sub Clearbtn_Click ()
    typecsystem.Text = ""
    softwarename.Text = ""
    txtUse.Text = ""
End Sub

Sub cmdOK_Click ()
    no_of_systems = numberlines(defapath + "dbstruct.txt")
    Unload editform
    cslistform.Show
End Sub
```

```
Sub createbtn_Click ()
  Dim filenum As Integer
  filenum = FreeFile

  Open defapath + "dbstruct.txt" For Append As #filenum

   newline1$ = Chr(34) + Trim$(typecsystem.Text) + Chr(34) + ","
   newline2$ = Chr(34) + Trim$(softwarename.Text) + Chr(34) + ","
   newline3$ = Chr(34) + Trim$(txtUse.Text) + Chr(34) + ","
   InfoFN$ = AssignInfoFN_1st(Trim$(softwarename.Text))'Assign a DEIF Name
      with no extension
   newline4$ = Chr(34) + InfoFN$ + Chr(34)
   newline$ = newline1$ + newline2$ + newline3$ + newline4$ + Chr(13) + Chr(10)
   Print #filenum, newline$
  Close #filenum

  activerow = 1
  dbstructgrid.Row = activerow

  dbstructgrid.Col = 1
  dbstructgrid.FixedAlignment(0) = 2
  dbstructgrid.Text = typecsystem.Text

  dbstructgrid.Col = 2
  dbstructgrid.FixedAlignment(1) = 2
  dbstructgrid.Text = softwarename.Text
  activerow = activerow + 1

  clear_combos
  createbtn.Enabled = False
  addbtn.Enabled = True


End Sub

Sub cscombo_Click ()
End Sub

Sub dbstructgrid_Click ()
   selgridtext = dbstructgrid.Text
End Sub

Sub del_ODOA ()

  ' delete the last chr$(10) and chr$(13)
  ' of a file

  Open defapath + "dbstruct.txt" For Input As #1
  filesize = LOF(1)
```

```
   linesfromfile$ = Input$(filesize - 4, 1)
   totallines$ = linesfromfile$
   Close #1

   Open defapath + "dbstruct.txt" For Output As #1
    Print #1, totallines$
   Close #1

End Sub

Sub Deletebtn_Click ()

   clear_griddisplay
   typecsystemvar = Trim$(typecsystem.Text)
   softwarevar$ = Trim$(softwarename.Text)
   usevar$ = Trim$(txtUse.Text)
   strsearch$ = Chr(34) + typecsystemvar + Chr(34) + "," + Chr(34) + softwarevar$ +
         Chr(34) + "," + Chr(34) + usevar$ + Chr(34)
   del_line defapath + "dbstruct.txt", strsearch$
   regen_griddata
   del_ODOA     'delete last chr(10) &chr(13)

   clear_combos 'clear text of combos


End Sub

Sub EnterRecord_Click ()
End Sub

Sub FindRecord_Click ()
End Sub

Sub Form_Load ()

    dbstructgrid.Col = 0
    For checkcount% = 0 To 29
       dbstructgrid.RowHeight(checkcount%) = 2300 / 6
       If checkcount% > 0 Then
          dbstructgrid.Row = checkcount%
          dbstructgrid.FixedAlignment(0) = 2
          dbstructgrid.Text = Str$(checkcount%)
       End If
    Next checkcount%

     dbstructgrid.ColWidth(0) = 1000 / 2
     dbstructgrid.ColWidth(1) = 2300 / 2
     dbstructgrid.ColWidth(2) = 2700 / 2
    dbstructgrid.Row = 0

    dbstructgrid.Col = 0
    dbstructgrid.FixedAlignment(0) = 2
    dbstructgrid.Text = "NO."

    dbstructgrid.Col = 1
    dbstructgrid.FixedAlignment(1) = 2
    dbstructgrid.Text = "TYPE"

    dbstructgrid.Col = 2
    dbstructgrid.FixedAlignment(2) = 2
    dbstructgrid.Text = "NAME"

    If selected = "modify" Then
      createbtn.Enabled = False
      'addbtn.Enabled = True
      modify_dbstruct   '************* subroutine
    End If

    If selected = "create" Then
      createbtn.Enabled = True
      addbtn.Enabled = False
    End If

    typecsystem.AddItem "CAD"
    typecsystem.AddItem "CAM"
    typecsystem.AddItem "CAD/CAM"
    typecsystem.AddItem "MRP"
    typecsystem.AddItem "MRP-II"
    typecsystem.AddItem "CAPP"
    typecsystem.AddItem "AS/RS"
    typecsystem.AddItem "ATE"
    typecsystem.AddItem "CACE"
    typecsystem.AddItem "Rapid PT M/C"

    softwarename.AddItem "ComputerVision"
    softwarename.AddItem "AutoCAD"
    softwarename.AddItem "SmartCAM"
    softwarename.AddItem "Costimator"
    softwarename.AddItem "R & P"


End Sub

Sub Form_Resize ()
  left = 200
```

```
    top = 200
End Sub

Sub leftarrow_Click ()

    rowst% = dbstructgrid.SelStartRow
    'MsgBox Str$(rowst%)
    no_of_systems = numberlines(defapath + "dbstruct.txt")
    Open defapath + "dbstruct.txt" For Input As #1
    Seek #1, 1
    checkcount% = 0

    Do Until checkcount% = no_of_systems

        checkcount% = checkcount% + 1
        Input #1, typecsvar$, namecsvar$, usevar$, InfoFN$

        If checkcount% = rowst% Then
            typecsystem.Text = typecsvar$
            softwarename.Text = namecsvar$
            txtUse.Text = usevar$
            Close
            Exit Sub
        End If

    Loop
    Close


End Sub

Sub modify_dbstruct ()
    no_of_systems = numberlines(defapath + "dbstruct.txt")
    Open defapath + "dbstruct.txt" For Input As #1
    Seek #1, 1
    checkcount% = 0
    activerow = 1

    Do Until checkcount% = no_of_systems
        'MsgBox Str$(checkcount)
        Input #1, typecsvar$, namecsvar$, usevar$, InfoFN$

        dbstructgrid.Row = activerow

        dbstructgrid.Col = 1
        dbstructgrid.FixedAlignment(0) = 2
        dbstructgrid.Text = typecsvar$

        dbstructgrid.Col = 2
        dbstructgrid.FixedAlignment(1) = 2
        dbstructgrid.Text = namecsvar$
        activerow = activerow + 1

        checkcount% = checkcount% + 1
    Loop
    Close

End Sub

Sub NextRecord_Click ()
End Sub


Sub okbtn_Click ()
End Sub

Sub Quit_Click ()
    Unload editform
    EnableToolButton
End Sub

Sub regen_griddata ()
    no_of_systems = numberlines(defapath + "dbstruct.txt")
    Open defapath + "dbstruct.txt" For Input As #1
    Seek #1, 1
    checkcount% = 0
    activerow = 1

    Do Until checkcount% = no_of_systems
        'MsgBox Str$(checkcount)
        Input #1, typecsvar$, namecsvar$, usevar$, InfoFN$

        dbstructgrid.Row = activerow

        dbstructgrid.Col = 1
        dbstructgrid.FixedAlignment(0) = 2
        dbstructgrid.Text = typecsvar$

        dbstructgrid.Col = 2
        dbstructgrid.FixedAlignment(1) = 2
        dbstructgrid.Text = namecsvar$
        activerow = activerow + 1

        checkcount% = checkcount% + 1
    Loop
    Close

End Sub
```

```
Sub softwarename_Change ()
   ValidateOkButton softwarename
End Sub

Sub softwarename_Click ()
   ValidateOkButton softwarename
End Sub

Sub txtUse_Change ()
   ValidateOkButton txtUse
End Sub

Sub typecsystem_Change ()
   ValidateOkButton typecsystem
End Sub

Sub typecsystem_Click ()
   ValidateOkButton typecsystem
End Sub

Sub ValidateOkButton (currentcontrol As Control)

   Const boxcount = 3
   Static ok_flag(boxcount)    As Integer
   Dim count_value As Integer
   Dim BoxText As String
   Dim index

   clearbtn.Enabled = False
   BoxText = currentcontrol
   index = Val(currentcontrol.Tag)
   If BoxText <> "" Then
       ok_flag(index) = True
   Else
       ok_flag(index) = False
   End If

   If ok_flag(1) = True Or ok_flag(2) = True Or ok_flag(3) = True Then
      clearbtn.Enabled = True
   End If

   For count_value = 1 To boxcount
      If ok_flag(count_value) = False Then
         'cmdOk.Enabled = False
         addbtn.Enabled = False
         deletebtn.Enabled = False
         Exit Sub
      End If
   Next count_value
   'cmdOk.Enabled = True
   addbtn.Enabled = True
   deletebtn.Enabled = True

End Sub
```

A.3 Source Codes for Specific Data Identification Function

```
Dim entryS As String
Dim namecsystem_check As String

Sub a_textCHK_Click ()
   If a_textchk.Value = 1 Then
     okbtn.Enabled = True
     a_text_entry.Show
     from_form = "a_text_entry"
   End If
End Sub

Sub a_textPU_Click ()
   nl$ = Chr$(13) + Chr$(10)
   msg1$ = "This is the text which should have the" + nl$
   msg2$ = "extracted text after it."
   MsgBox msg1$ + msg2$, 64, "HINT"

End Sub

Sub b_textCHK_Click ()
   If b_textchk.Value = 1 Then
     okbtn.Enabled = True
     b_text_entry.Show
     from_form = "b_text_entry"
   End If
End Sub

Sub b_textPU_Click ()
   nl$ = Chr$(13) + Chr$(10)
   msg1$ = "This is the text which should have the extracted" + nl$
   msg2$ = "text before it."
   MsgBox msg1$ + msg2$, 64, "HINT"

End Sub

Sub c_o_notationCHK_Click ()
  If c_o_notationchk.Value = 1 Then
    okbtn.Enabled = True
    c_o_notation_entry.Show
  End If
End Sub
```

```
Sub c_o_notationPU_Click ()
    nl$ = Chr$(13) + Chr$(10)
    msg1$ = "To specify the count of notation before " + nl$
    msg2$ = "data to be extracted."
    MsgBox msg1$ + msg2$, 64, "HINT"

End Sub

Function cal_accum_value () As Integer
  Dim acc_value As Integer
  acc_value = 0
  If r_datachk.Value = 1 Then
     acc_value = 1
  End If
  If i_notationchk.Value = 1 Then
     acc_value = acc_value + 2
  End If
  If c_o_notationchk.Value = 1 Then
     acc_value = acc_value + 4
  End If
  If n_f_endingchk.Value = 1 Then
     acc_value = acc_value + 8
  ElseIf n_f_endingchk.Enabled = False Then     ' for automatic count
     acc_value = acc_value + 8
  End If

  If b_textchk.Value = 1 Then
     acc_value = acc_value + 16
  End If
  If a_textchk.Value = 1 Then
     acc_value = acc_value + 32
  End If
  If orderchk.Value = 1 Then
     acc_value = acc_value + 64
  End If
  If offsetchk.Value = 1 Then
     acc_value = acc_value + 128
  End If

  cal_accum_value = acc_value
End Function

Sub cancelbtn_Click ()
End Sub

Sub Check1_Click ()

End Sub
```

```
Sub clear_CHKvalues ()
   r_datachk.Value = 0
   i_notationchk.Value = 0
   c_o_notationchk.Value = 0
   n_f_endingchk.Value = 0
   b_textchk.Value = 0
   a_textchk.Value = 0
   orderchk.Value = 0
   offsetchk.Value = 0
   entrystring = ""
End Sub

Sub cmdDEIF_Click ()
   Dim filenum As Integer
   filenum = FreeFile

   'namecsystem_check = namecsystem
   'namecsystem_check = compact$(namecsystem_check)
   'len_namecsystem% = Len(namecsystem_check)
   'If len_namecsystem% > 8 Then
     'namecsystem_check = Left$(namecsystem_check, 8)
   'End If

   file_name = defapath + info_filename + ".inf"  '******

   checkname = Dir$(file_name)
   If checkname = "" Then
      MsgBox file_name + " NOT FOUND !", 48, "Error"
      Exit Sub
   End If

   Open file_name For Input As #filenum
     filesize = LOF(filenum)
     linesfromfile$ = Input$(filesize - 2, filenum)
   Close #filenum
   If Len(linesfromfile$) > 500 Then
     linesfromfile$ = Mid$(linesfromfile$, 1, 500)
   End If
   MsgBox linesfromfile$, 64, "DEI File -> " & file_name
End Sub

Sub cmdExit_Click ()
   Unload sp_dataidentify
   EnableToolButton
End Sub

Sub CSCmd1_Click ()
```

```
    If Not Help_File_In_Path() Then
       Text = "GenFlow Help File not found in your path." + Chr$(10) + Chr$(13)
       Text = Text + "Windows searches your PATH environment variable for help
              files, "
       Text = Text + "so you need to copy GenFlow.HLP to a directory included in
              your "
       Text = Text + "PATH if you wish to obtain help while running GenFlow."
       MsgBox Text, 48, "GenFlow Help File not available"
       HelpFilePath = "c:\hl\genflow\helpfile\genflow.hlp"
    End If

    R = WinHelp(sp_dataidentify.hWnd, HelpFilePath, HELP_CONTEXT, 10077)

End Sub

Sub Form_Load ()
   computersystem.Text = namecsystem
End Sub

Sub Form_Resize ()
   top = 200
   Left = 4750
End Sub

Sub Form_Unload (Cancel As Integer)

   editor.identifycmd.Enabled = True
   editor.extractcmd.Enabled = True
End Sub

Function getentryS () As String
  Dim entryS As String
  entryS = ""
  If r_datachk.Value = 1 Then
    entryS = Chr$(34) + r_data_entryS + Chr(34)
  End If
  If i_notationchk.Value = 1 Then
    entryS = entryS + "," + Chr$(34) + i_notation_entryS + Chr(34)
  End If
  If c_o_notationchk.Value = 1 Then
    entryS = entryS + "," + Chr$(34) + c_o_notation_entryS + Chr(34)
  End If
  If n_f_endingchk.Value = 1 Then
    entryS = entryS + "," + Chr$(34) + n_f_ending_entryS + Chr(34)
  End If
  If n_f_endingchk.Enabled = False Then  ' for automatic count
    entryS = entryS + "," + Chr$(34) + n_f_ending_entryS + Chr(34)
  End If
  If b_textchk.Value = 1 Then
    entryS = entryS + "," + Chr$(34) + beforetext_entryS + Chr(34)
  End If
  If a_textchk.Value = 1 Then
    entryS = entryS + "," + Chr$(34) + aftertext_entryS + Chr(34)
  End If
  If orderchk.Value = 1 Then
    entryS = entryS + "," + Chr$(34) + order_entryS + Chr(34)
  End If
  If offsetchk.Value = 1 Then
    entryS = entryS + "," + Chr$(34) + offset_entryS + Chr(34)
  End If

  getentryS = entryS

End Function

Sub i_notationchk_Click ()
  If i_notationchk.Value = 1 Then
    okbtn.Enabled = True
    notation_type = "iden_sign"
    i_notation_entry.Show
    i_notation_entry.Caption = "Identify Notation"
  End If
End Sub

Sub i_notationPU_Click ()
    nl$ = Chr$(13) + Chr$(10)
    msg1$ = "To identify the notation or sign which can be helped" + nl$
    msg2$ = " to specify the location of data to be extracted."
    MsgBox msg1$ + msg2$, 64, "HINT"


End Sub

Sub n_f_endingchk_Click ()
  If n_f_endingchk.Value = 1 Then
      okbtn.Enabled = True
      notation_type = "iden_end"
      i_notation_entry.Show
      i_notation_entry.Caption = "Identify Notation for Ending"
  End If
End Sub

Sub n_f_endingPU_Click ()
    nl$ = Chr$(13) + Chr$(10)
    msg1$ = "To identify the notation which signals" + nl$
    msg2$ = "the ending of the data to be extracted."
    MsgBox msg1$ + msg2$, 64, "HINT"
```

```
End Sub

Sub offsetCHK_Click ()
  If offsetchk.Value = 1 Then
    okbtn.Enabled = True
    offset_entry.Show
  End If
End Sub

Sub offsetPU_Click ()
  nl$ = Chr$(13) + Chr$(10)
  msg1$ = "This is referring to the offset of the " + nl$
  msg2$ = "reference data to the extracted data."
  MsgBox msg1$ + msg2$, 64, "HINT"

End Sub

Sub okbtn_Click ()
  Dim filenum As Integer

  okbtn.Enabled = False

  labelname = InputBox$("Enter label for selected data :", "Enter Label Name")
  ' if the response is not Cancel
  If labelname = "" Then
    response% = MsgBox("Take Reference Text as Label Name ?", 4)
  End If

  If response% = 6 Then    'Yes
        labelname = r_data_entryS
  ElseIf response% = 7 Then    'No
        Exit Sub
  End If


  filenum = FreeFile
  iden_value = cal_accum_value() '******* calculate selected value
  If r_data_entryS = "" Then
   MsgBox "Reference Data must have an entry !", 64
   Exit Sub
  End If
  'namecsystem_check = namecsystem
  'namecsystem_check = compact$(namecsystem_check)
  'len_namecsystem% = Len(namecsystem_check)
  'If len_namecsystem% > 8 Then
    'namecsystem_check = Left$(namecsystem_check, 8)
  'End If
```

```
   file_name = defapath + info_filename + ".inf"  '******
   there% = exist%(file_name)
   entrystring = getentryS() '************** Get full string of entry
   If there% = -1 Then
     'Unload sp_dataidentify
     overwrite_add.Show
   Else
      Select Case MsgBox("No Extract File Exist, Create One ?", 35)
        Case 2  'Cancel
          Unload sp_dataidentify
          Exit Sub
        Case 6   'Yes
          GoSub data_to_new_file
        Case 7   'No
          Unload sp_dataidentify
          Exit Sub
      End Select
   End If

   n_f_endingchk.Enabled = True
   r_datachk.Value = 0
   i_notationchk.Value = 0
   c_o_notationchk.Value = 0
   n_f_endingchk.Value = 0
   b_textchk.Value = 0
   a_textchk.Value = 0
   orderchk.Value = 0
   offsetchk.Value = 0

   Exit Sub

data_to_new_file:
   Open file_name For Output As #filenum
     newline1$ = Chr(34) + RTrim$(labelname) + Chr(34) + ","
     newline2$ = Chr(34) + LTrim$(Str$(iden_value)) + Chr(34) + ","
     newline3$ = entrystring
     newline$ = newline1$ + newline2$ + newline3$ + Chr(13) + Chr(10)
     Print #filenum, newline$
   Close #filenum
   clear_CHKvalues
   MsgBox newline$

 Return

End Sub

Sub orderCHK_Click ()
  If orderchk.Value = 1 Then
```

```
      okbtn.Enabled = True
      order_entry.Show
   End If
End Sub

Sub orderPU_Click ()
   nl$ = Chr$(13) + Chr$(10)
   msg1$ = "This refers to the reference data. If the reference" + nl$
   msg2$ = "data is not unique, this will indicate which one."
   MsgBox msg1$ + msg2$, 64, "HINT"

End Sub

Sub r_datachk_Click ()
   If r_datachk.Value = 1 Then
      okbtn.Enabled = True
      r_data_entry.Show
      from_form = "r_data_entry"
   End If
End Sub

Sub r_datahint_Click ()
End Sub

Sub r_dataPU_Click ()
   nl$ = Chr$(13) + Chr$(10)
   msg1$ = "This is the text which is related to the" + nl$
   msg2$ = "data to be extracted from the data file."
   MsgBox msg1$ + msg2$, 64, "HINT"

End Sub
```

A.4 Source Codes for Variable Data Identification Function

```
Dim namecsystem_check As String

Sub a_textCHK_Click ()
   If a_textchk.Value = 1 Then
      okbtn.Enabled = True
      a_text_entry.Show
      from_form = "a_text_entry"
   End If
End Sub

Sub a_textPU_Click ()
   nl$ = Chr$(13) + Chr$(10)
   msg1$ = "This is the text which should have the" + nl$
   msg2$ = "extracted text after it."
   MsgBox msg1$ + msg2$, 64, "HINT"
```

```
End Sub

Sub b_textCHK_Click ()
   If b_textchk.Value = 1 Then
     okbtn.Enabled = True
     b_text_entry.Show
     from_form = "b_text_entry"
   End If
End Sub

Sub b_textPU_Click ()
   nl$ = Chr$(13) + Chr$(10)
   msg1$ = "This is the text which should have the extracted" + nl$
   msg2$ = "text before it."
   MsgBox msg1$ + msg2$, 64, "HINT"

End Sub

Sub c_o_notationCHK_Click ()
  If c_o_notationchk.Value = 1 Then
    okbtn.Enabled = True
    c_o_notation_entry.Show
  End If
End Sub

Sub c_o_notationPU_Click ()
   nl$ = Chr$(13) + Chr$(10)
   msg1$ = "To specify the count of notation before " + nl$
   msg2$ = "data to be extracted."
   MsgBox msg1$ + msg2$, 64, "HINT"

End Sub

Function cal_accum_value () As Integer
  Dim acc_value As Integer
  acc_value = 0
  If r_datachk.Value = 1 Then
    acc_value = 1
  End If
  If i_notationchk.Value = 1 Then
    acc_value = acc_value + 2
  End If
  If c_o_notationchk.Value = 1 Then
    acc_value = acc_value + 4
  End If
  If n_f_endingchk.Value = 1 Then
    acc_value = acc_value + 8
  ElseIf n_f_endingchk.Enabled = False Then    ' for automatic count
```

```
    acc_value = acc_value + 8
  End If

  If b_textchk.Value = 1 Then
    acc_value = acc_value + 16
  End If
  If a_textchk.Value = 1 Then
    acc_value = acc_value + 32
  End If
  If orderchk.Value = 1 Then
    acc_value = acc_value + 64
  End If
  If offsetchk.Value = 1 Then
    acc_value = acc_value + 128
  End If

  cal_accum_value = acc_value
End Function

Sub cancelbtn_Click ()

End Sub

Sub clear_CHKvalues ()
   r_datachk.Value = 0
   i_notationchk.Value = 0
   c_o_notationchk.Value = 0
   n_f_endingchk.Value = 0
   b_textchk.Value = 0
   a_textchk.Value = 0
   orderchk.Value = 0
   offsetchk.Value = 0
   entrystring = ""
End Sub

Sub cmdCancel_Click ()
   Unload v_dataidentify
   EnableToolButton
End Sub

Sub cmdChange_Click ()
   lblIncrement.Visible = True
   txtIncrement.Visible = True
   spnIncrement.Visible = True
   cmdOK.Visible = True
   cmdCancel.Visible = True
End Sub

Sub cmdDEIF_Click ()
```

```
    Dim filenum As Integer
    filenum = FreeFile

    'namecsystem_check = namecsystem
    'namecsystem_check = compact$(namecsystem_check)
    'len_namecsystem% = Len(namecsystem_check)
    'If len_namecsystem% > 8 Then
      'namecsystem_check = Left$(namecsystem_check, 8)
    'End If              '

    If lblTitle.Caption = "Reference Text Identification" Then
        file_name = defapath + info_filename + ".rti"'**************
    ElseIf lblTitle.Caption = "Associated Values Identification" Then
        file_name = defapath + info_filename + ".avi"'**************
    End If

    checkname = Dir$(file_name)
    If checkname = "" Then
        MsgBox file_name + " NOT FOUND !", 48, "Error"
        Exit Sub
    End If

    Open file_name For Input As #filenum
      filesize = LOF(filenum)
      linesfromfile = Input$(filesize - 2, filenum)
    Close #filenum
    'MsgBox linesfromfile$, 64, "DEI File -> " & file_name
    frmDEIFile.Show
End Sub

Sub cmdExit_Click ()
    editor.identifycmd.Enabled = True
    editor.extractcmd.Enabled = True
        Unload v_dataidentify
    EnableToolButton

End Sub

Sub cmdOK_Click ()

    mousepointer = 11
    For i% = 1 To 99
        oldtext$ = Chr$(34) + Trim$(c_o_notation_entryS) + Chr$(34)
            c_o_notation_entryS = Trim$(Str(Val(c_o_notation_entryS) +
            Val(txtIncrement)))
        newtext$ = Chr$(34) + c_o_notation_entryS + Chr$(34)
        entrystring = replaceS(entrystring, oldtext$, newtext$)
        append_to_inf_file
    Next i%
```

```
   MsgBox entrystring

   lblTitle.Caption = "Associated Values Identification"
   lblTitle.FontSize = 9.75
   lblTitle.ForeColor = &HC0&
   cmdChange.Enabled = False

   lblIncrement.Visible = False
   txtIncrement.Visible = False
   spnIncrement.Visible = False
   cmdOK.Visible = False
   cmdCancel.Visible = False

   r_datachk.Value = 0
   i_notationchk.Value = 0
   c_o_notationchk.Value = 0
   n_f_endingchk.Value = 0
   b_textchk.Value = 0
   a_textchk.Value = 0
   orderchk.Value = 0
   offsetchk.Value = 0
   mousepointer = 0


End Sub

Sub CSCmd1_Click ()
      If Not Help_File_In_Path() Then
         Text = "GenFlow Help File not found in your path." + Chr$(10) + Chr$(13)
         Text = Text + "Windows searches your PATH environment variable for help
                files, "
         Text = Text + "so you need to copy GenFlow.HLP to a directory included in
                your "
         Text = Text + "PATH if you wish to obtain help while running GenFlow."
         MsgBox Text, 48, "GenFlow Help File not available"
         HelpFilePath = "c:\hl\genflow\helpfile\genflow.hlp"
      End If

      R = WinHelp(v_dataidentify.hWnd, HelpFilePath, HELP_CONTEXT, 10078)

End Sub

Sub Form_Load ()
   computersystem.Text = namecsystem
End Sub

Sub Form_Resize ()
   top = 200
   Left = 4000
```

```
    width = 5235
    height = 4440
End Sub

Sub Form_Unload (Cancel As Integer)
    editor.identifycmd.Enabled = True
    editor.extractcmd.Enabled = True

End Sub

Function getentryS () As String
  Dim entryS As String
  entryS = ""
  If r_datachk.Value = 1 Then
     entryS = Chr$(34) + r_data_entryS + Chr(34)
  End If
  If i_notationchk.Value = 1 Then
     entryS = entryS + "," + Chr$(34) + i_notation_entryS + Chr(34)
  End If
  If c_o_notationchk.Value = 1 Then
     entryS = entryS + "," + Chr$(34) + Trim$(c_o_notation_entryS) + Chr(34)
  End If
  If n_f_endingchk.Value = 1 Then
     entryS = entryS + "," + Chr$(34) + n_f_ending_entryS + Chr(34)
  End If
  If b_textchk.Value = 1 Then
     entryS = entryS + "," + Chr$(34) + beforetext_entryS + Chr(34)
  End If
  If a_textchk.Value = 1 Then
     entryS = entryS + "," + Chr$(34) + aftertext_entryS + Chr(34)
  End If
  If orderchk.Value = 1 Then
     entryS = entryS + "," + Chr$(34) + order_entryS + Chr(34)
  End If
  If offsetchk.Value = 1 Then
     entryS = entryS + "," + Chr$(34) + offset_entryS + Chr(34)
  End If

  getentryS = entryS

End Function

Sub i_notationchk_Click ()
  If i_notationchk.Value = 1 Then
    okbtn.Enabled = True
    notation_type = "iden_sign"
    i_notation_entry.Show
    i_notation_entry.Caption = "Identify Notation"
  End If
End Sub

Sub i_notationPU_Click ()
   nl$ = Chr$(13) + Chr$(10)
   msg1$ = "To identify the notation or sign which can be helped" + nl$
   msg2$ = " to specify the location of data to be extracted."
   MsgBox msg1$ + msg2$, 64, "HINT"


End Sub

Sub n_f_endingchk_Click ()
  If n_f_endingchk.Value = 1 Then
     okbtn.Enabled = True
     notation_type = "iden_end"
     i_notation_entry.Show
     i_notation_entry.Caption = "Identify Notation for Ending"
  End If
End Sub

Sub n_f_endingPU_Click ()
   nl$ = Chr$(13) + Chr$(10)
   msg1$ = "To identify the notation which signals" + nl$
   msg2$ = "the ending of the data to be extracted."
   MsgBox msg1$ + msg2$, 64, "HINT"


End Sub

Sub offsetCHK_Click ()
   If offsetchk.Value = 1 Then
      okbtn.Enabled = True
      offset_entry.Show
   End If
End Sub

Sub offsetPU_Click ()
   nl$ = Chr$(13) + Chr$(10)
   msg1$ = "This is referring to the offset of the " + nl$
   msg2$ = "reference data to the extracted data."
   MsgBox msg1$ + msg2$, 64, "HINT"

End Sub

Sub okbtn_Click ()
   Dim filenum As Integer
   cmdChange.Enabled = True
   okbtn.Enabled = False
   labelname = InputBox$("Enter label for selected data :", "Enter Label Name")
```

```
' if the response is not Cancel
If labelname = "" Then
    response% = MsgBox("Take Reference Text as Label Name ?", 4)
End If

If response% = 6 Then    'Yes
        labelname = r_data_entryS
ElseIf response% = 7 Then    'No
        Exit Sub
End If

cmdChange.Enabled = True  ' enable the command for change
filenum = FreeFile
iden_value = cal_accum_value() '******* calculate selected value
If r_data_entryS = "" Then
  MsgBox "Reference Data must have an entry !", 64
  Exit Sub
End If
'namecsystem_check = namecsystem
'namecsystem_check = compact$(namecsystem_check)
'len_namecsystem% = Len(namecsystem_check)
'If len_namecsystem% > 8 Then
   'namecsystem_check = Left$(namecsystem_check, 8)
'End If

If lblTitle.Caption = "Reference Text Identification" Then
    file_name = defapath + info_filename + ".rti"'**************
ElseIf lblTitle.Caption = "Associated Values Identification" Then
    file_name = defapath + info_filename + ".avi"'**************
End If

there% = exist%(file_name)
entrystring = getentryS() '************** Get full string of entry
If there% = -1 Then
   'Unload v_dataidentify
   overwrite_add.Show
Else
    Select Case MsgBox("No Extract File Exist, Create One ?", 35)
      Case 2  'Cancel
         Unload v_dataidentify
         Exit Sub
      Case 6   'Yes
         GoSub data_to_new_file
      Case 7   'No
         Unload v_dataidentify
         Exit Sub
    End Select
End If
n_f_endingchk.Enabled = True
```

```
    r_datachk.Value = 0
    i_notationchk.Value = 0
    c_o_notationchk.Value = 0
    n_f_endingchk.Value = 0
    b_textchk.Value = 0
    a_textchk.Value = 0
    orderchk.Value = 0
    offsetchk.Value = 0
    routeflag = True

    Exit Sub

data_to_new_file:
    Open file_name For Output As #filenum
      newline1$ = Chr(34) + RTrim$(labelname) + Chr(34) + ","
      newline2$ = Chr(34) + LTrim$(Str$(iden_value)) + Chr(34) + ","
      newline3$ = entrystring
      newline$ = newline1$ + newline2$ + newline3$ + Chr(13) + Chr(10)
      Print #filenum, newline$
    Close #filenum
    clear_CHKvalues
    MsgBox newline$
    cmdChange.Enabled = True
Return
End Sub

Sub orderCHK_Click ()
   If orderchk.Value = 1 Then
      okbtn.Enabled = True
      order_entry.Show
   End If
End Sub

Sub orderPU_Click ()
    nl$ = Chr$(13) + Chr$(10)
    msg1$ = "This refers to the reference data. If the reference" + nl$
    msg2$ = "data is not unique, this will indicate which one."
    MsgBox msg1$ + msg2$, 64, "HINT"

End Sub

Sub r_datachk_Click ()
    If r_datachk.Value = 1 Then
       okbtn.Enabled = True
       r_data_entry.Show
       from_form = "r_data_entry"
    End If
End Sub
```

```
Sub r_datahint_Click ()
End Sub

Sub r_dataPU_Click ()
   nl$ = Chr$(13) + Chr$(10)
   msg1$ = "This is the text which is related to the" + nl$
   msg2$ = "data to be extracted from the data file."
   MsgBox msg1$ + msg2$, 64, "HINT"

End Sub

Sub spnIncrement_SpinDown ()
   If Val(txtIncrement) < 1 Then Exit Sub
   txtIncrement = Val(txtIncrement) - 1

End Sub

Sub spnIncrement_SpinUp ()
   txtIncrement = Val(txtIncrement) + 1
End Sub
```

A.5 Source Codes for External Conversion Function

```
Const TEXTFLAG = 0
Const FILEFLAG = 1
Const DIRFLAG = 2

Dim SelectFlag As Integer
'Const true = -1
'Const false = 0
Const modal = 1
Const wildcard$ = "*.*"

Dim FileSelected As Integer
Dim aboutstr As String
Dim f_name As String



Sub cancelbtn_Click ()
  Unload getfile
  get_fullfilepath.Show
End Sub

Sub cmdConvert_Click ()
   On Error GoTo errorhandler


   If txtSaveAs.Text = "" Then
   MsgBox "No destination file name !", 48, "Error"
   Exit Sub
  End If

  If txtSoftWare.Text = "" Then
   MsgBox "No software name !", 48, "Error"
   Exit Sub
  End If

  For fcnt% = 0 To filelst.ListCount - 1   '************ Main routine
     If filelst.Selected(fcnt%) Then
       'MsgBox filelst.List(fcnt%)
       'totalConverted$ = ""
       GoSub ModifyFilename
       'MsgBox FSelected$
       If optYes.Value = True Then GoSub VerifyConversion
       GoSub DoConvert
     End If
   Next fcnt%
  Exit Sub

VerifyConversion:

  Select Case MsgBox("Convert file " + Fselected$ + " ?", 35, "")
   Case 2  'cancel
     Exit Sub
   Case 6  'yes
   Case 7  'no
     'whereis.Show
     Exit Sub
  End Select

Return


DoConvert:
   messagebox.Visible = True
   messagebox = "Converting " + Fselected$ + " ..."
   convertfile$ = txtSoftWare + " " + Fselected$ + " " + txtSaveAs.Text
   x = Shell(convertfile$, 1)
   Do While GetModuleUsage(x) > 0  ' Loop until program finished
    DoEvents
   Loop


   messagebox = ""
   messagebox.Visible = False
Return
```

```
ModifyFilename:
   If dateopt.Value = True Then
      f_name = LTrim$(RTrim$(Mid$(filelst.List(fcnt%), 14)))
      GoSub AddDirPath
   ElseIf nameopt.Value = True Then
      f_name = RTrim$(Left$(filelst.List(fcnt%), 12))
      GoSub AddDirPath
   End If
Return

AddDirPath:
   If Right$(dir1.Path, 1) = "\" Then
      Fselected$ = dir1.Path + f_name
   Else
      Fselected$ = dir1.Path + "\" + f_name
   End If
Return

errorhandler:

    MsgBox "Error encountered in Convert Operation ! ", 48
    Exit Sub


End Sub

Sub cmdExit_Click ()
   Unload getfile
   EnableToolButton
End Sub

Sub cmdSaveAs_Click ()
   cmdialog1.Filter = "All Files |*.*"
   cmdialog1.Action = 2
   txtSaveAs = cmdialog1.Filename

End Sub

Sub cmdSoftware_Click ()
   cmdialog1.Filter = "All Files |*.*"
   cmdialog1.Action = 1
   txtSoftWare = cmdialog1.Filename

End Sub

Sub Combo1_Change ()
   updatefileslst
   ValidateOkButton filelst
End Sub
```

```
Sub Combo1_Click ()
   updatefileslst
   ValidateOkButton filelst
End Sub

Sub Command1_Click ()
  On Error GoTo errortrap

  If SelectFlag = TEXTFLAG Then
    'file1.FileName = Text1.Text
    If FileSelected = True Then
      On Error GoTo 0
      Unload getfile
      Exit Sub
    End If
    'dir1.Path = file1.Path
  ElseIf SelectFlag = DIRFLAG Then
    dir1.Path = dir1.List(dir1.ListIndex)
    Dir1_Change
  Else
    If Right$(dir1.Path, 1) = "\" Then
      'fullfilepath = dir1.Path + Text1.Text
    Else
      'fullfilepath = dir1.Path + "\" + Text1.Text
    End If
    FileSelected = True
    Unload getfile
  End If
  Exit Sub

errortrap:
  Beep
  Resume Next
End Sub

Sub Command2_Click ()
  Unload getfile
End Sub

Sub convertcmd_Click ()

 Dim x As Integer


 'SendKeys "%{ }{Down 3}{Enter}", True
 For i = 0 To filelst.ListCount - 1

  curfilename = filelst.List(i)
```

```
   If Right$(curfilename, 1) <> "*" Then
     'filetobeconverted.Text = curfilename
     'messagebox.Visible = True
     'messagebox.Text = "Converting " + curfilename + " ..."
     'On Error GoTo err_trap
     'convertfile$ = txtSoftWare.Text + " " + curfilename + " " + txtSaveAs.Text
     'x = Shell(convertfile$, 1)
     'setuserattr filelst.List(i)
     'MsgBox Str$(getuserattr%(filelst.List(i)))
   End If
   'MsgBox "wait"
  Next i
'end Sub
 'SendKeys "%{ }Enter}", True
  messagebox.Visible = False
  Exit Sub

err_trap:

    message$ = errormsg$(errornumber%)
    MsgBox message$, 48
    messagebox.Visible = False
    Exit Sub


End Sub

Sub dateopt_Click ()
   If dateopt.Value = True Then
      updatefileslst
   End If
End Sub

Sub Dir1_Change ()
  FillLabel1
  'file1.FileName = dir1.Path + "\" + file1.Pattern
  Drive1.Drive = dir1.Path
  'Text1.Text = file1.Pattern
  SelectFlag = DIRFLAG
  updatefileslst
End Sub

Sub Dir1_Click ()
  SelectFlag = DIRFLAG
End Sub

Sub Drive1_Change ()
  dir1.Path = Drive1.Drive
  'Text1.Text = file1.Pattern
  SelectFlag = DIRFLAG
End Sub

Sub exitbtn_Click ()
   Unload getfile
End Sub

Sub File1_Click ()
  'Text1.Text = file1.FileName
  SelectFlag = FILEFLAG
End Sub

Sub filelst_Click ()
   ValidateOkButton filelst
   If dateopt.Value = True Then
     f_name = LTrim$(RTrim$(Mid$(filelst.Text, 14)))
   ElseIf nameopt.Value = True Then
        f_name = RTrim$(Left$(filelst.Text, 12))
   End If

End Sub

Sub filetobeconvertedbtn_Click ()



End Sub

Sub FillLabel1 ()
  'Label1.Caption = dir1.Path
  'If Label1.Width > 2055 Then
    'a$ = Left$(dir1.Path, 3)
    'b$ = Mid$(dir1.Path, 4)
    'Do While InStr(b$, "\")
      'b$ = Mid$(b$, InStr(b$, "\") + 1)
    'Loop
    'Label1.Caption = a$ + "...\" + b$
  'End If
End Sub

Sub Form_Load ()
  cur_form = "getfile"
  getfile.Left = (Screen.Width - getfile.Width) / 2
  getfile.Top = (Screen.Height - getfile.Height) / 2
  combo1.AddItem "*.exe"
  combo1.AddItem "*.nc"
  combo1.AddItem "*.txt"
  combo1.AddItem "*.bat"
  combo1.AddItem "*.*"
  combo1.Text = "*.*"
  'txtSoftWare.Text = "c:\unix2dos"

  dateopt.Value = True

  'If fullfilepath <> "" Then
    'Tmp$ = fullfilepath
    'Do Until Right$(Tmp$, 1) = "\"
      'Tmp$ = Left$(Tmp$, Len(Tmp$) - 1)
    'Loop
    'Tmp$ = Tmp$ + wildcard$
    'file1.FileName = Tmp$
    'dir1.Path = file1.Path
  'End If
  'file1.path = "a:\"
  'file1.Pattern = wildcard$
  'FillLabel1
  'Text1.Text = file1.Pattern
  'SelectFlag = DIRFLAG
  'fileselected = False
  'updatefileslst

End Sub

Sub Form_Resize ()
  'Text1.SetFocus
  Left = 100
  Top = 100
  Height = 4920
  Width = 8205
End Sub

Sub getfilebtn_Click ()
   Static word(15) As String
   Dim numwords As Integer
   Dim i As Integer, k As Integer
   Dim s As String
   Dim addfiles As Integer
   Dim f As String

   wildcards$ = combo1.Text
   updatefileslst

End Sub

Sub groupconvertcmd_Click ()

End Sub
```

```
Sub nameopt_Click ()
   If nameopt.Value = True Then
      updatefileslst
   End If

End Sub

Sub okbtn_Click ()
  If SelectFlag = TEXTFLAG Then
    'fullfilepath = file1.Path + "\" + file1.FileName
  Else
    If Right$(dir1.Path, 1) = "\" Then
      fullfilepath = dir1.Path + f_name
    Else
      fullfilepath = dir1.Path + "\" + f_name
    End If
  End If
  FileSelected = True
  If selflag = "selsoft" Then

   get_fullfilepath.sel_software.Caption = fullfilepath
  End If

  If selflag = "selfile" Then

   get_fullfilepath.sel_file.Caption = fullfilepath
  End If
    'MsgBox fullfilepath
    'msgbox selflag
   get_fullfilepath.Show
  Unload getfile

End Sub

Sub selectcmd_Click ()
  groupselect = True
  'whereis.Show
End Sub

Sub softwarebtn_Click ()


    If Right$(dir1.Path, 1) = "\" Then
      txtSoftWare.Text = dir1.Path + f_name
    Else
      txtSoftWare.Text = dir1.Path + "\" + f_name
    End If

End Sub
```

```
Sub Text1_Change ()
  SelectFlag = TEXTFLAG
End Sub

Sub txtSaveAs_Change ()
    ValidateOkButton txtSaveAs
End Sub

Sub txtSoftware_Change ()
   ValidateOkButton txtSoftWare
End Sub

Sub updatefileslst ()
   Const ascending_order = 0
   Dim filecount As Integer
   Dim i As Integer
   Dim dirstr As String


  dirstr = dir1.List(-1)
  If Right$(dirstr, 1) <> "\" Then dirstr = dirstr + "\"
  dirstr = dirstr + combo1.Text

  filecount = fcount%(dirstr)
  'MsgBox "filecount " + Str$(filecount)
  Do While filelst.ListCount
    filelst.RemoveItem 0
  Loop
  If filecount > 0 Then

     If dateopt.Value = True Then
       ReDim f_arrayT(1 To filecount) As fullinfo
       readfileX dirstr, dirsize&, f_arrayT(1)
       For i = 1 To filecount
         str_edate2num$ = LTrim$(Str$(date2num%(f_arrayT(i).filedate)))
         'MsgBox f_arrayT(i).filedate
         f_arrayT(i).filedate = addfrontzero2string(str_edate2num$, 5)
         'sortstr f_arrayT(1), filecount, 0
         'f_arrayT(i).filedate=enum2date$(val(f_arrayT(i).filedate))
         filelst.AddItem f_arrayT(i).filedate + "    " + RTrim$(f_arrayT(i).basename) +
                "." + f_arrayT(i).extname
       Next i

       For i = 1 To filecount
         datenum_str$ = Left$(filelst.List(0), 5)
         withoutdate_str$ = Mid$(filelst.List(0), 7)
         dat$ = enum2date$(Val(datenum_str$))
         filelst.RemoveItem 0
```

```
        filelst.AddItem dat$ + withoutdate_str$, filecount - i
      Next i

    End If

    If nameopt.Value = True Then
      ReDim f_array(0 To filecount) As String
      For i = 1 To filecount
         f_array(i) = Space$(37)
      Next

      f_array(0) = dirstr
      readfileI f_array(0)

      sortstr f_array(1), filecount, 0
      For i = 1 To filecount
           filelst.AddItem f_array(i)
      Next i
    End If

  End If


End Sub

Sub ValidateOkButton (currentcontrol As Control)

   Const boxcount = 3
   Static Ok_Flag(boxcount)    As Integer
   Dim count_value As Integer
   Dim BoxText As String

   Dim index

   BoxText = currentcontrol.Text
   index = Val(currentcontrol.Tag)
   If BoxText <> "" Then
      Ok_Flag(index) = True
   Else
      Ok_Flag(index) = False
   End If

   For count_value = 1 To boxcount
     If Ok_Flag(count_value) = False Then
        cmdConvert.Enabled = False
        Exit Sub
     End If
   Next count_value
   cmdConvert.Enabled = True
```

```
End Sub
```

A.6 Source Codes for Browser Function

```
Sub cmbNameComSys_Change ()
   ValidateViewButton cmbNameComSys
End Sub

Sub cmbNameComSys_Click ()
   ValidateViewButton cmbNameComSys
   Dim filenum As Integer
   filenum = FreeFile
   no_of_systems = numberlines(defapath + "dbstruct.txt")

   List_index% = cmbNameComSys.ListIndex 'Get the selected item index
   List_index% = List_index% + 1

   'Get the content of Use from file
   Open defapath + "dbstruct.txt" For Input As #filenum
   Seek #filenum, 1
   checkcount% = 0

   Do Until checkcount% = no_of_systems

      checkcount% = checkcount% + 1
      Input #filenum, typecsvar$, namecsvar$, usevar$, InfoFN$

      If checkcount% = List_index% Then
         cmbTypeComSys = typecsvar$
         lblUse = usevar$
         txtDEIFile = InfoFN$
         Close #filenum
         Exit Sub
      End If

   Loop
   Close #filenum
End Sub

Sub cmbSelectedFiles_Change ()
    ValidateViewButton cmbSelectedFiles
End Sub

Sub cmbSelectedFiles_Click ()
   ValidateViewButton cmbSelectedFiles
End Sub

Sub cmbTypeComSys_Change ()
```

```
   ValidateViewButton cmbTypeComSys
End Sub

Sub cmbTypeComSys_Click ()
   ValidateViewButton cmbTypeComSys
   Dim filenum As Integer
   filenum = FreeFile
   no_of_systems = numberlines(defapath + "dbstruct.txt")

   List_index% = cmbTypeComSys.ListIndex 'Get the selected item index
   List_index% = List_index% + 1

   'Get the content of Use from file
   Open defapath + "dbstruct.txt" For Input As #filenum
   Seek #filenum, 1
   checkcount% = 0

   Do Until checkcount% = no_of_systems

      checkcount% = checkcount% + 1
      Input #1, typecsvar$, namecsvar$, usevar$, InfoFN$

      If checkcount% = List_index% Then
         cmbNameComSys = namecsvar$
         lblUse = usevar$
         txtDEIFile = InfoFN$
         Close #filenum
         Exit Sub
      End If

   Loop
   Close #filenum



End Sub

Sub cmdClipCopy_Click ()
   Dim ClipText
   Dim CopyText
   Dim NC
   Dim NR
   Dim Count
   Dim colstrt
   Dim colend
   Dim rowstrt
   Dim rowend
   Dim HeadTxt
   ' Initialize new column (NC) & row (NR) variables
```

```
    NC = Chr$(9)
    NR = Chr$(13) & Chr$(10)
    ' Initialize variables for rows/cols selected
    mousepointer = 11
    colstrt = viewdata_grid.SelStartCol
    colend = viewdata_grid.SelEndCol
    rowstrt = viewdata_grid.SelStartRow
    rowend = viewdata_grid.SelEndRow
    ' Initialize variable that gets input from grid
    ClipText = viewdata_grid.Clip
    ' Initialize variable that holds output to the clipboard
    CopyText = ""
    ' Header row = current row
    viewdata_grid.Row = 0
    ' For each cell in header row
    For Count = colstrt To colend
       viewdata_grid.Col = Count
       ' Define heading text
       HeadTxt = viewdata_grid.Text
       ' Copy column header to output variable
       CopyText = CopyText & NC & HeadTxt
    Next Count

    CopyText = CopyText & NR
    ' Add 1st row header to output variable
    viewdata_grid.Col = 0
    viewdata_grid.Row = rowstrt
    CopyText = CopyText & viewdata_grid.Text & NC
    ' Copy text from input variable to output variable
    For Count = 1 To Len(ClipText)
       ' If current character isn't carriage return
       If Mid$(ClipText, Count, 1) <> Chr$(13) Then
          ' Concat character to output variable
          CopyText = CopyText & Mid$(ClipText, Count, 1)
       ' If current character is carriage return
       Else
          ' Concat carriage return/linefeed to output variable
          viewdata_grid.Row = viewdata_grid.Row + 1
          CopyText = CopyText & NR & viewdata_grid.Text & NC
       End If
    Next Count
    ' Copy contents of output variable to clipboard
    Clipboard.SetText CopyText
    cmdClipCopy.Enabled = False
    mousepointer = 0

End Sub

Sub cmdMax_Click ()
```

```
   If browser.optSpecific.Value = False And browser.optVariable = False Then
      MsgBox "Need to specify <Specific> or <Variable>", 48, "Message"
      Exit Sub
   End If
   frmMax.Show
End Sub

Sub cmdQuit_Click ()
     If formexists("selectfiles") Then
          Unload SelectFiles
     End If
     mdimenu.cmdToolbar(4).Enabled = True
     Unload browser
End Sub

Sub cmdSelect_Click ()
   clearlist cmbSelectedFiles
   SelectFiles.Show
End Sub

Sub cmdView_Click ()

  Dim j As Integer
  Dim k As Integer

   On Error GoTo errorhandler

   If optSpecific.Value = False And optVariable = False Then
      MsgBox "Need to specify <Specific> or <Variable>", 48, "Message"
      Exit Sub
   End If

   DEIFName$ = txtDEIFile
   If optSpecific.Value = True Then
      DEIFName$ = defapath + Trim$(txtDEIFile) + ".inf"
   ElseIf optVariable.Value = True Then
      DEIFName$ = defapath + Trim$(txtDEIFile) + ".rti"
   End If

  no_of_lines = numberlines(DEIFName$)
  If no_of_lines = False Then Exit Sub

  For fcnt% = 0 To cmbSelectedFiles.ListCount - 1  'Get files in List
       FilePicked$ = cmbSelectedFiles.List(fcnt%)
         MsgBox FilePicked$
         GoSub ExtDataFromFile
         If fcnt% = 0 Then
            GoSub ViewGridValue1  'The  data  from first file
         Else
```

```
            GoSub ViewGridOtherValue
         End If

   Next fcnt%

  cmdView.Enabled = False
  Exit Sub


ViewGridValue1:
    viewdata_grid.Cols = 2'initialise number of columns
   'MsgBox "RowHeight  " & viewdata_grid.RowHeight(1)
   viewdata_grid.Height = viewdata_grid.RowHeight(1) * (no_of_lines + 1) + 300
   If viewdata_grid.Height > 3225 Then
      viewdata_grid.Height = 3225
   End If

   viewdata_grid.ColWidth(0) = 1500
   viewdata_grid.ColWidth(1) = 1200

   viewdata_grid.Rows = no_of_lines + 1
   viewdata_grid.Row = 0
   viewdata_grid.Col = fcnt%
   viewdata_grid.FixedAlignment(0) = 2
   viewdata_grid.Text = "LABEL NAME"
   viewdata_grid.Col = 1
   viewdata_grid.FixedAlignment(1) = 2
   viewdata_grid.Text = SelectedFiles(1)

   k = -1
   s$ = DataExtracted$
   r$ = SearchText$
   For i% = 1 To no_of_lines
        viewdata_grid.Row = i%
        viewdata_grid.Col = 0
        j = k + 2
        viewdata_grid.Text = sdata_extracted_comb(s$, r$, Chr$(34), j, Chr$(34))
        viewdata_grid.Col = 1
        k = j + 2
        viewdata_grid.Text = sdata_extracted_comb(s$, r$, Chr$(34), k, Chr$(34))

   Next i%
Return

ViewGridOtherValue:
   k = -1
   s$ = DataExtracted$
   r$ = SearchText$
   viewdata_grid.Cols = fcnt% + 2 'Set number of columns to be used
   viewdata_grid.ColWidth(fcnt% + 1) = 1000 'Set width of added column
   viewdata_grid.Col = fcnt% + 1  'Set Current column
   viewdata_grid.FixedAlignment(fcnt% + 1) = 2
   viewdata_grid.Row = 0
   viewdata_grid.Text = SelectedFiles(fcnt% + 1)

   For i% = 1 To no_of_lines
       viewdata_grid.Row = i%
       k = k + 4
       viewdata_grid.Text = sdata_extracted_comb(s$, r$, Chr$(34), k, Chr$(34))
   Next i%

Return

ExtDataFromFile:
   file_name = FilePicked$
   SearchText$ = Trim$(cmbNameComSys) & " -> " & Trim$(lblUse)
   namecsystem = Trim$(cmbNameComSys)'Put into Global variable
   use_deif = Trim$(lblUse)
   DataExtracted$ = extdata_f_dfl(SearchText$)
   MsgBox DataExtracted$
Return

DisplayInGrid:
Return


errorhandler:

    MsgBox "Error encountered during operation ! ", 48
    Exit Sub


End Sub

Sub Form_Load ()
   cmbSelectedFiles.Text = ""
   no_of_systems = numberlines(defapath + "dbstruct.txt")
   getcomsyslist  'get the computer sys. list from dbstruct.txt
   For i = 1 To no_of_systems
      cmbTypeComSys.AddItem comsys(i)
      cmbNameComSys.AddItem softname(i)
   Next i

End Sub

Sub Form_Resize ()
   left = 150
   top = 100
```

```
  width = 8880
  Height = 4605
End Sub

Sub lblUse_Change ()
    ValidateViewButton lblUse
End Sub

Sub lblUse_Click ()
   ValidateViewButton lblUse
End Sub

Sub txtDEIFile_Change ()
    ValidateViewButton txtDEIFile
End Sub

Sub txtDEIFile_Click ()
   ValidateViewButton txtDEIFile
End Sub

Sub ValidateViewButton (currentcontrol As Control)

   Const boxcount = 5
   Static Ok_Flag(boxcount)    As Integer
   Dim count_value As Integer
   Dim BoxText As String

   Dim index

   BoxText = currentcontrol
   index = Val(currentcontrol.Tag)
   If BoxText <> "" Then
       Ok_Flag(index) = True
   Else
       Ok_Flag(index) = False
   End If

   For count_value = 1 To boxcount
      If Ok_Flag(count_value) = False Then
         cmdView.Enabled = False
         Exit Sub
      End If
   Next count_value
   cmdView.Enabled = True

End Sub

Sub viewdata_grid_Click ()
   colstrt = viewdata_grid.SelStartCol
```

```
    colend = viewdata_grid.SelEndCol
    rowstrt = viewdata_grid.SelStartRow
    rowend = viewdata_grid.SelEndRow

    If (colend - colstrt) <> 0 Or (rowend - rowstrt) <> 0 Then
        cmdClipCopy.Enabled = True
    Else
        cmdClipCopy.Enabled = False
    End If

End Sub

Sub viewdata_grid_MouseUp (Button As Integer, Shift As Integer, X As Single, Y As
Single)
    colstrt = viewdata_grid.SelStartCol
    colend = viewdata_grid.SelEndCol
    rowstrt = viewdata_grid.SelStartRow
    rowend = viewdata_grid.SelEndRow

    If (colend - colstrt) <> 0 Or (rowend - rowstrt) <> 0 Then
        cmdClipCopy.Enabled = True
    Else
        cmdClipCopy.Enabled = False
    End If

End Sub
```