THE APPLICATION OF COMPUTER AIDED DESIGN

IN

STRUCTURAL ENGINEERING ANALYSIS

by

Peter Miles McClintock, B.Sc. (Eng.), A.C.G.I

November 1974

A thesis submitted for the degree of Doctor of Philosophy of the University of London and for the Diploma of Imperial College.

Mechanical Engineering Department,
Imperial College,
London, S.W.7.

Abstract

The thesis describes work carried out at Imperial College of Science and Technology on the development of an interactive general computer aided draughting/design system, based on a minicomputer with a digitising table, storage tube and plotting table connected on-line.

The system consists of modular programs providing facilities for data input, display, filing and plotting. It is designed not only to produce drawings but also to act as a base for application programs in the fields of engineering and design.

In particular, a structural analysis system 'Stasys' has been developed to enable the construction of an idealised model of buildings and structures, which can be represented by spaceframe (line) and rectangular plate finite elements, and for which the model can be constructed from plans of the different floors or levels of the building or structure.

The data thus generated, may either be output to magnetic tape to form input data for analysis programs on a 'mainframe' computer, or the stiffness matrix that describes the force-displacement properties of the structure may be assembled and solved on the minicomputer. Where this is possible large time savings result. The size of the problem that can be solved is limited only by the quantity of backing store available.

ACKNOWLEDGEMENTS

The author is indepted to Scott Wilson Kirkpatrick & Partners for sponsorship throughout this project, to Dr. C.B. Besant who supervised the work at Imperial College, and to all colleagues who offered both advice and criticism.

Thanks are due particularly to Mr. C.Y. Hsiung for his guidance through the finite element theory, and last, but not least, to my wife who typed the thesis.

LIST OF CONTENTS

NOTATION

u	Displacement
F	External force
k	Stiffness coefficient
k^A	Superscript A refers to element A
$[\ ]$	Matrix
$\{\ \}$	Vector
$[K]^A_L$	Subscript L refers to local coordinate system
$[K]^A_G$	Subscript G refers to global coordinate system
S	Stress matrix
e	Strain
O	Stress
A	Cross sectional area
E	Youngs modules
v, ν	Poisson's ratio

CHAPTER ONE
INTRODUCTION

1.1 Computers in Design.

In the modern world there are many factors which are causing increasing complexity in the design of man made objects: the need to conserve raw materials by minimising the quantity of required material in a design, which can only be achieved by improved analysis and optimisation; the advent of new technologies demanding higher precision; the growth in size of projects in order to maximise economies of scale.

There are also pressures for designs to be carried out more quickly. It follows that any device which enables man to design more efficiently is of great value.

The enormous potential of computers for use in an organisational capacity has already been demonstrated and exploited in all major industrial concerns. They are to be found controlling accounts, inventory and production schedules. However, the use of computers in design, other than as calculating machines, has not yet been recognised to the same extent. This is because in the organisational role, the media of information passed between man and machine is alphanumeric - characters and numbers. Devices enabling this type of data transfer were easily developed from existing technology in the typewriter industry. In the design process the information media is graphic, most commonly engineering drawings, and only recently have graphic terminals become widely available.

Moreover, the design process is difficult to cost and the economic advantages (or disadvantages) of making expensive computers available to the design office will probably never be fully estimated. It has been mathematical problems that have driven the designer to use the computer, often in order to use techniques of analysis that were not humanly possible.

Some of these analysis programs require extremely large amounts of data and now the designer would like to find some way in which to reduce the burden of data preparation. The computer with recently developed peripherals is able to aid the designer in that function, and also in other areas such as information retrieval and draughting.

The price of computers continues to drop dramatically, even in a period of inflation, and at the same time their reliability, flexibility and scope increase. It therefore seems likely that the use of computers in design will become ever more prevalent.

Although computer hardware is observed to be falling in price, the same is certainly not true of software. The increase in software cost is almost certain to continue for some time at a high rate. This is not only because it is actually costing more to produce but also because in the past the true cost of software development has been hidden from the user by the supplier lumping it together with the cost of the hardware.

Now that this price has been appreciated, software has started to be separately priced.

These factors have meant that the true price of software is higher than generally realised and because of this, it is important that the software written to drive the graphic terminals in the design office be capable of supporting a wide range of tasks. It becomes necessary to definethe role that the computer system should fill and then decide what type of system is best able to fulfill that role.

It is possible to identify five major and time consuming areas in the design process:-

(a) Retrieval and extraction of information

(b) Analysis of a design and assessment of its technical performance

(c) Implementation and communication of design changes

(d) Production of drawings

(e) Estimating the cost of a design and the quantities of materials it requires

From these, we deduce that the computer system in the design office ought to support the following activities:-

(a) The maintenance of a data base which may be accessed by engineers working in different fields on the same project.

(b) The addition to, or editing of, the data base in a graphic mode.

(c) Providing access to and supporting a wide range of application programs.

(d) The production of working drawings.

(e) The support of programs for scheduling and accounting.

The maintenance of a data base implies the existence of mass storage within the computer system. The storage medium should provide fast-access to any part of the current project data base for editing, addition and extraction of information.

The system must possess graphic input and output devices and sufficient computing power to carry out analysis programs. Above all, the system must respond quickly to any request made by the user, and, design is a task that involves careful thought and decision on the part of the designer, it should not be so expensive that he is over concerned with getting finished as quickly as possible.

1.2 Finite Element Analysis

It was stated in section 1.1 that designers have started to use computers to carry out techniques of analysis that were not humanly possible. The finite element method is one such example.

Finite element techniques have become increasingly important in the last ten years for carrying out stress analysis on many types of structure. Most of the early work was carried out in the aircraft industry where more accurate methods of analysis were required for the increasingly complex airframes being developed. It was particularly fortunate that at this time digital computers were available within the industry. Interest soon spread to other fields of engineering, and in mechanical and civil engineering, developement and use of the method continues at an accelerating rate.

The finite element method has drastically cut the time required for accurate analysis and has given rise to a tremendous increase in scope to the designer of complex structures.

The technique is a generalisation of standard structural analysis procedures. It permits their extension so that displacements and stresses can be calculated in two and three dimensional structures by the same techniques used for ordinary frame structures.

The basic concept is that every structure may be considered to be an assemblage of individual structural components or elements interconnected at a finite number of points. It is the finite character of the structural connectivity which makes possible solution by simultaneous algebraic equations and which distinguishes a structural system from a problem in continuum mechanics.

It must be realised that the approximation involved in the use of the method is essentially physical. The assemblage of elements is

substituted for the continuum. There need be no mathematical approximation in the solution of the substitute system. This is an important difference between finite element and finite difference methods. The structural idealisation is obtained by dividing the original continuum into segments of appropriate sizes and shapes, all the material properties being retained in the individual elements. The capacity for treating arbitrary material properties and boundary conditions is one of the principal attributes of the finite element method.

At present by far the longest steps in the analysis process are the input and output of data, the actual c.p.u.* time needed is unlikely to exceed ten minutes even on very complex problems.

By contrast, dividing the structure into elements and translating the graphic finite element data, which is normally presented in the form of drawing, into numerical data may take several days. The division of the structure is critical and should be carried out by the engineer as the approximation is likely to be the most important source of error in the analysis.

Clearly, a graphic based computer system would ease this situation.

---

*Central Processing Unit.

1.3 Alternative Computer Systems.

In section 1.1, it was proposed that the computer in design should be capable of supporting several activities and must therefore have sepcific attributes. In this section, a brief survey of available computer systems and commercially produced peripherals is given and the extent to which they are able to support these activities and possess the relevant attributes is discussed.

A single computer system is built from many components which may roughly be divided into four categories:-

(a) Storage devices

(b) Central processors

(c) Input/output devices

(d) Operating systems

The way in which these components are linked together defines the system architecture.

(a) Storage devices.

1) Core storage or main memory.

A computer program consists of a large number of individual instructions which are executed one by one to operate on data. An individual instruction may require only a few microseconds to complete its function, and this operation time must include the time needed to transfer the instruction from memory to the processor and to interpret and execute the instruction, including extracting the required data from memory and storing the result. Thus, it can be seen that the memory must allow for the storage of a large number of instructions and data, which must be available as required, at high speed.

Access to any word is by direct address (ie. each word location has a numbered address). Access times are typically 900nanoseconds, or in the case of semi-conductor memory, can be as low as 300nanoseconds.

Because of the high price of this type of store, it is not economic to consider storing all data and programs in such a medium. The main memory is used to hold current data and instructions, and other data and program segments are called in from backing store as required.

Typical cost per byte is 10 pence, but this is still falling.

2) Disk storage.

A typical magnetic disk store consists of a number of rotating disks each coated upon both curfaces with a magnetizable material. Information is written to, or read from, the disk by a series of arms, one for each disk surface. Each arm contains one, or several read/write heads, and the arms can be positioned above certain areas of the corresponding disks according to instructions received from the central processor. In the case of fixed head disks, a series of reading heads cover the whole radius of the disk so that no movement of the heads is necessary.

Typical cost per byte for a moving head disk would be .25 pence, while the cost for a fixed head disk per byte would be roughly four times this. In general, moving head disks are constructed so that the disks can be changed. In this way, data on a disk pack can be permanently saved.

Access time canbe measured in tens of milliseconds.

3) Magnetic drum storage.

Similar to the magnetic disk but with a drum. Performance and price are similar to those of a fixed head disk. Both disks and drums are suitable in direct access applications and are a compromise to obtain, at an economic price, certian of the qualities exhibited by magnetic core memories.

4) Magnetic tape storage.

Magnetic tape is a serial store; data is recorded as individual characters along a length of tape and processing can only be achieved

by sorting records into a sequence so that each can be examined in turn.

It is mainly used for the storage of complete sets of data or jobs that can be serially read onto disk or drum, or into main memory.

A typical tape can hold eight megabytes of information. When in operation, the cost of storage per byte is approximately .065 pence. This includes the cost of the tape transport and controller. When used for offline storage the cost is only .0001 pence per byte, much cheaper than storing information in normal writing on paper.

(b) Central processors.

The central processor is the nerve centre of any digital computer system, since it coordinates and controls the activities of all the other units and performs all the arithmetic and logical processes applied to the data. Certain arithmetic operations such as multiplication can either be programmed or carried out by hardware. To carry out operations by hardware is much quicker than doing the equivalent operation by software but is much more expensive. There is a trade-off between the power and speed of the processor, and the cost. Different types of application require more or less c.p.u.* effort. Graphic design requires relatively little computation while analysis will require much more.

Processors used to be the most expensive part of a computer system but this is no longer so and the price of processors continues to drop. Naked mini-computers may now be purchased for only £1,000.

(c) Input/output devices.

1) Card input

A very commonly used method for feeding programs and data into a computer. They provide a convenient way of storing programs and have

---

*Central Processing Unit.

the advantage of the data being visible. However, to produce data cards in any number is tedious, and there are liable to be numerous data errors. This will result in more than one run of a program being necessary, in order to eliminate the errors. Large stacks of cards are cumbersome, liable to disorder and wasteful of office space.

2) Paper-tape input/output.

The use of paper-tape is becoming rare because like cards there are physical handling problems for large programs and large quantities of data. It is often used for transferring data or programs between machines and also has the advantage of data being visible.

3) Teletype.

This is a two way communication device that is connected directly with the computer. The operator may enter data directly into the system by typing on the keyboard and the computer may reply by causing a message to be printed directly at the inquiry terminal. One of the factors here is that if the operator makes a mistake in keying the inquiry or data the error may be detected by the computer and a request for its connection immediately issued. Typical cost £800.

4) Visual display unit.

This is similar in operation and use to the teletype but instead of information being typed onto paper, it is displayed on a screen. This has the advantage of being much faster and this is particularly useful in program development, where the operator often wishes to view a section of program, or information retrieval. The disadvantage is that no hard copy is produced for the later reference. The V.D.U.* is cheaper to run since it requires no paper. Typical cost £800.

---

*Visual Display Unit.

5) Line printer.

The line printer is capable of producing printed output at high speed ranging from 300 to 2,000 lines of 160 characters each per minute. Fast line printers are found to consume enormous quantities of paper. They can be programmed to print on preprinted stationery and are valuable for producing schedules, invoice and statements. Typical cost is from £1,000. upwards.

6) Refresh graphic display.

It is both an input and an output device. The refresh C.R.T.* when used with a light pen provides the operator with a means of directly inputting graphic data. It usually includes a keyboard.

Most graphic input/output devices are more expensive to run than those dealing only with alphanumerics. A refresh C.R.T. is expensive because it requires a display processor to drive it. The whole picture must also be stored in core which ties up part of the supporting computer systems capability. There is also a limit on the amount of information that can be displayed before the picture starts flickering. Cost of the typical terminal including display processor and minicomputer is £6,000.

7) Storage tube.

This is often operated in conjunction with a cursor, which enables the operator to specify position on the screen, and a keyboard. All that is written to the face of the screen is stored until the whole screen face is erased. There is therefore no limit to the complexity of the picture which can be displayed and does not have to be held in main memory.

However, selective erasure is impossible and the display is best visible in subdued light.

---

*Cathode Ray Tube.

A hard copy device is available which simply puts an image of the screen onto special paper.

Typical cost £4,000. for tube, keyboard and cursor; £2,000. for the hard copy unit.

8) Digitiser.

The device enables the cartesian coordinates of a point in a plane to be transmitted to the computer. It consists of a table and a pencil with sensing apparatus that monitors the position of the pencil on the table.

One considerable advantage of this device is that it can be the same size as standard working drawings. These can be stuck to the table and the operator may work from them. The device is accurate to within .1mm. Typical cost £7,000.

9) Data tablet.

This is similar to the digitiser but generally smaller, of lower accuracy and cheaper. See figure 1.1.

10) Plotter.

A variety of plotting devices exist. There are two distinct types: Drum plotters and flat-bed plotters. In drum plotter, the paper is drawn round a drum. The drum revolves to give movement in the x-coordinate while a pen unit mounted on a gantry lying along the drum provides y-movement.

In a flat-bed plotter, the paper remains still while the gantry moves.

Another plotting device is microfilm plotter. A photograph is taken of a high resolution C.R.T. display. This is many times faster than normal plotting devices but also more expensive. They are however, well worth it if the work load is sufficient.

(d) Operating systems.

An operating system may be defined as those procedures which control

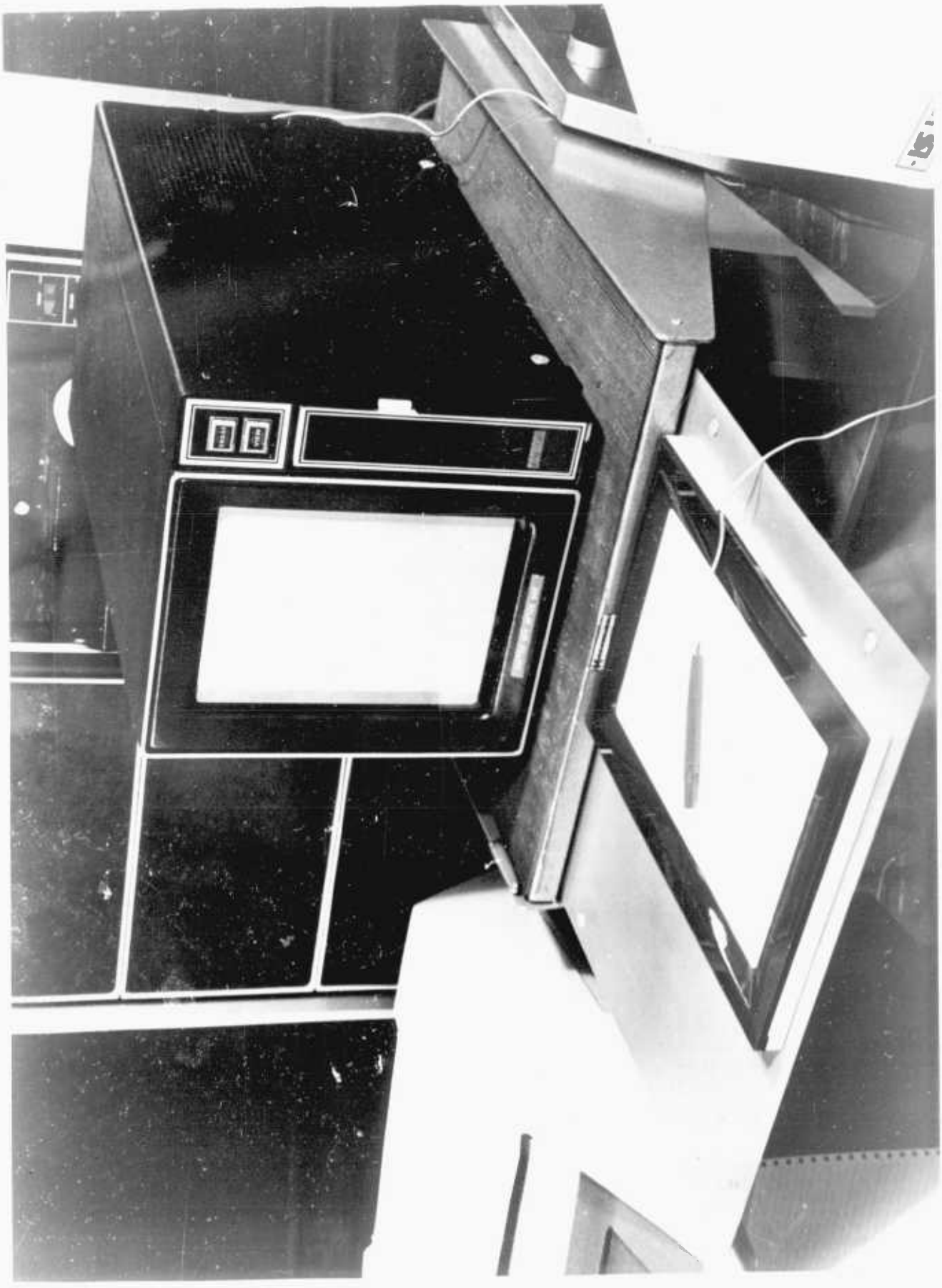


Figure 1.1 Small data tablet and storage tube

the resources within a computer system. Operating systems generally become more complex as the size of the installation increases. The operating system will determine in what processing mode the computer is to be used. Common modes are:-

1) Batch processing mode.

In this mode, jobs are prepared and then input as a single unit, usually in the form of punched cards, to the computer. At some later time, the job is processed and the results output. The time between input and output is termed 'turn around time' and may range from 10 minutes to one or more weeks. Average time is a few hours. There is no interaction between the user and machine, indeed, the user may never even see the machine.

Batch processing mode is certainly unsuitable for the editing or interrogation of a data base and for graphic design.

2) Remote terminal processing.

A multi-access system is one that allows a number of remote terminals to have interactive access to the central computer resource. The remote terminals may range from teletypes to complete satellite systems. It appears to each user that he has the total computer resources to himself. By time sharing it is possible to give a response time of a few seconds to each terminal because human intervention is comparatively such a slow process.

The operator of the remote terminal is able to conduct his work in conversational mode.

This type of operation is clearly suitable for the design application. Because of the large quantities of data that are generated in the design process, the terminal must have sufficient data handling capabilities.

One way to achieve this is to use a minicomputer system as the terminal device. This also allows the user to perform data preparation tasks such as editing and debugging program modules without having to pay for 'connect' and processing time which would result if this was done by the central computer.

Such a terminal is called intelligent.

3) On-line processing.

In on-line processing, the operator is permanently connected to the computer resource. It is mainly used when a high degree of interaction is required. It is clearly capable of supporting the design task but is too expensive to be considered where the central computer is to be capable of handling large analysis programs.

The best alternative appears to be a remote intelligent terminal connected to a powerful central computer.

The computing power of the minicomputer has increased enormously over five years, so that they are now capable of handling most of the processing required by the five activities defined in section 1.1. If such a proposition is true, it is possible to largely dispense with the control computer and to use the intelligent terminal in a 'stand alone' situation.

Minicomputers first came into operation in 1962 and since their introduction, their success has been remarkable.[1] Equally remarkable has been their increase in power.

The trend towards installing minicomputers is a move in the opposite direction to large integrated program systems, maintained by central organisations and accessible from terminals. Because of their increase in power, the definition of a minicomputer is becoming obscure, but it is

generally accepted that it is a type of machine that can be offered in a viable configuration with a core in the range of perhaps 8 to 32Kilowords. Some machines of this type can be extended to a core size of 128Kilowords or more, and they can be equipped with a wide range of peripheral devices such as disk and magnetic tape storage, visual display units, plotters and high speed printers as described, so that in their most sophisticated configurations they can compare with some 'large' computers.

Their significance is that, in their smaller configurations, they provide the opportunity for small firms to have their independent in-house computer. This is particularly important for firms that are interested in systems with interactive input and output through graphic display units, because the delays and costs involved in conducting interactive graphics remotely from the computer are prohibitive in most commercial applications. Thus mincomputers are essential to firms involved in graphics, but otherwise, they should not be installed unless a thorough study has demonstrated that it would not be cheaper to have a terminal linked to a remote central computer. In this comparison, it should be remembered that a minicomputer in-house necessitates sufficient staff who can program for it, and who understand its operating system.

This leads directly to an increasing tendency to study the economics of the entire design and construction process very carefully to decide which areas are likely to yield the best financial return on the cost of applying computers to them. One aim of this project is to test the capability of a minicomputer in solving quite large and complex problems. It is necessary to establish which type of programs can be successfully and economically run on a minicomputer.

CHAPTER 2

A General Computer Aided Draughting System

2.1 Aims of the system.

There was a desire to provide a computer aided design system which would be useful in many fields. The area in which all fields came most nearly together was in graphics and draughting, and it was therefore thought that a general draughting system would provide the best base for any computer aided design system in a particular application.

The system should provide: a means of graphic data input and output including the production of drawings, an operating or user action control system and an interface for applications programs, which could be used for design work in Civil, Mechanical and Electrical engineering. Applications in mind were:-

Data preparation for structural analysis

Building design

Mechanical engineering component design

Printed circuit layout design

The software should comprise an executive and a set of program modules, each module to perform one or more functions. In this way users would select which modules they required for their particular application, and, a library of modules could be gradually built up continually adding to the power of the system without causing old versions to become incompatible.

Once developed and tested, system modules should only be updated by system programmers, application programmers should not be allowed to modify the system modules to their own ends.

In particular modification of the data structure which would make existing data incompatible should be avoided at all costs. The reason for these restrictions was to ensure that persons working on different problems but on the same project would be able to utilise the same database. Thus data initially generated by an architect doing building design could subsequently be used by the structural engineer to generate data for analysis programs and to produce detailed reinforcement drawings.

Lastly, the data structure should be as simple as possible to allow easy interaction with analysis programs. This was considered more important than trying to obtain the last bit of speed out of the system by the use of complex data structures as it was not anticipated that speed of operation would be a limiting factor.

2.2 Hardware.

Previous development had been carried out at Imperial College using a PDP 8/E* minicomputer, with 8K 12 bit words of core storage and dual Dectape providing 500K of slow backing storage, connected on-line to a D-MAC digitiser and Tectronix 611 storage screen.

The main factor limiting the use of this system was its slowness of operation and difficulties in programming. All programs were written in assembly language for the sake of efficiency, and because the level of Fortran that the system could support was too low to be substantially useful. Since Dectapes were the only form of backing store, and these have an access time of several seconds, the system had to be core resident. This made expansion of the system extremely difficult.

The system on which the work by the author was carried out was based around a PDP 11/45* which had considerably more power (see Fig. 2.1). The main components of the system which was largely supplied by C.E.C.** are now described.

a) Processor.

PDP 11/45 with floating point hardware for increased performance. The performance of the PDP 11/45 is compared with that of the PDP 8/E and CDC 6400 in figure 2.2.

b) Storage.

1) Main memory is 16Kilowords of 16bits each, this is expandible up to 124Kilowords.

---

*Manufactured by Digital Equipment Corporation.

**Computer Equipment Company.

Figure 2.1 The system hardware

Operation	PDP 8/I 12 bits	PDP 11/45 16 bits	CDC 6400 60 bits
Memory cycle time	1.5	.9	.5
Add	3.0	.3	1.0
Multiply (fixed point)	360.	3.3	1.0
Divide (fixed point)	460.	7.0	1.0
Floating multiply	-	4.55 - 6.55 6.55 - 11.95*	11.4
Floating divide	-	4.55 - 9.95 6.55 - 18.35*	11.4

* double precision - 64 bits.

Figure 2.2 Comparison of instruction execution times. (micro seconds).

2) Fast backing store is provided by a 1.2 million word moving head disk utilising removable disk cartridges so that each user is able to maintain security over his own data and programs by removing his disk after use.

3) Bulk storage was added in the form of a Magnetic tape unit* in August 1974. This was required not only for storage purposes but also to provide a medium for data transfer between the system and other computer complexes.

c) Input and output devices.

1) An LA 30 DECwriter was used as the system console. This prints characters quietly at 15 characters per second but can be up-graded to 30 characters per second.

2) A high speed paper tape reader and punch capable of reading characters per second and punching characters per second.

3) Serial line printer** with a maximum speed of 180 characters per second. This is almost essential for the listing of results and is extremely useful for obtaining listings of programs during developement, as this is a slow process using the DECwriter.

4) D-MAC digitising/plotting table. The table has a working area of 1.5 by 1.0 metres and consists of two flat surfaces, one above the other. The upper surface, made of toughened glass and upon which the operator may lay drawings and sketches, is used for digitising, the lower surface is used as a flatbed plotter. Between the two layers a gantry is driven by a servo-motor in the x-direction, and mounted on the gantry is a carriage which is also driven by a servo-motor in the y-direction. By a combination of gantry and carriage movement, it is possible to position the carriage anywhere within the table surfaces.

---

*Manufactured by Kennedy & Company.

**Logabax LX180.

The carriage contains a sensing coil and movement of the carriage and gantry is detected by Moire fringe shaft encoders, driven by steel wires attached to the gantry and carriage. A free pencil (Fig. 2.3) containing a coil, through which is passed a 400Hz signal, is held by the operator or above the digitising surface. The sensing coils in the carriage detect x and y positional errors between the carriage and the free pencil, and amplified signals are sent to the servo-motors to drive the carriage towards the free pencil. The movement of the carriage is monitored by the encoders and the outputs are fed to digital counters in the CAMAC interface.

When any one of the eight buttons on the free pencil is pressed the computer reads the current x and y coordinates of the table and records which button was pressed.

In the plotting mode the servo-motors are driven by signals from the computer. A pen unit mounted on the carriage is also under computer control, and raises and lowers the pen.

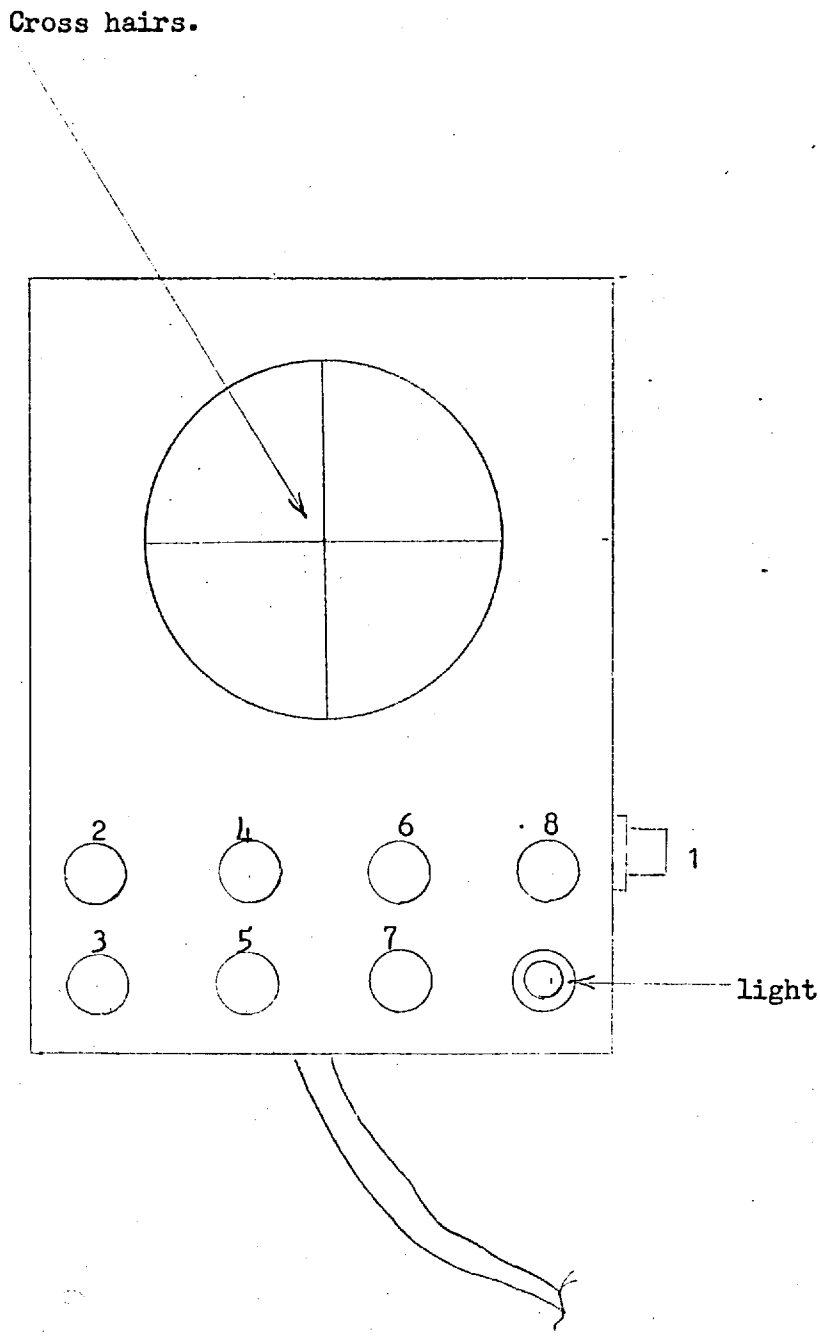
The digitising serves as the main input device for the system.

5) Tectronix 611 storage tube driven by S.E.N.* vector and character generators mounted in the CAMAC crate provides visual display. This has three intensity levels and by correct setting it is possible to arrange that the lowest level be non storing. This is used to display a tracking cross or cursor continually monitoring the position of the digitiser without storing it on the screen.

6) A flat bed plotter with working area 1 square metre developed at Imperial College is also connected on line to the system (Fig.2.4). This was driven by D.C. printed circuit motors via a stereo power amplifier providing a great deal more power than was available in the

---

*S.E.N. Electroniqua Limited.



Numbers refer to function buttons.

Figure 2.3 The digitising pencil

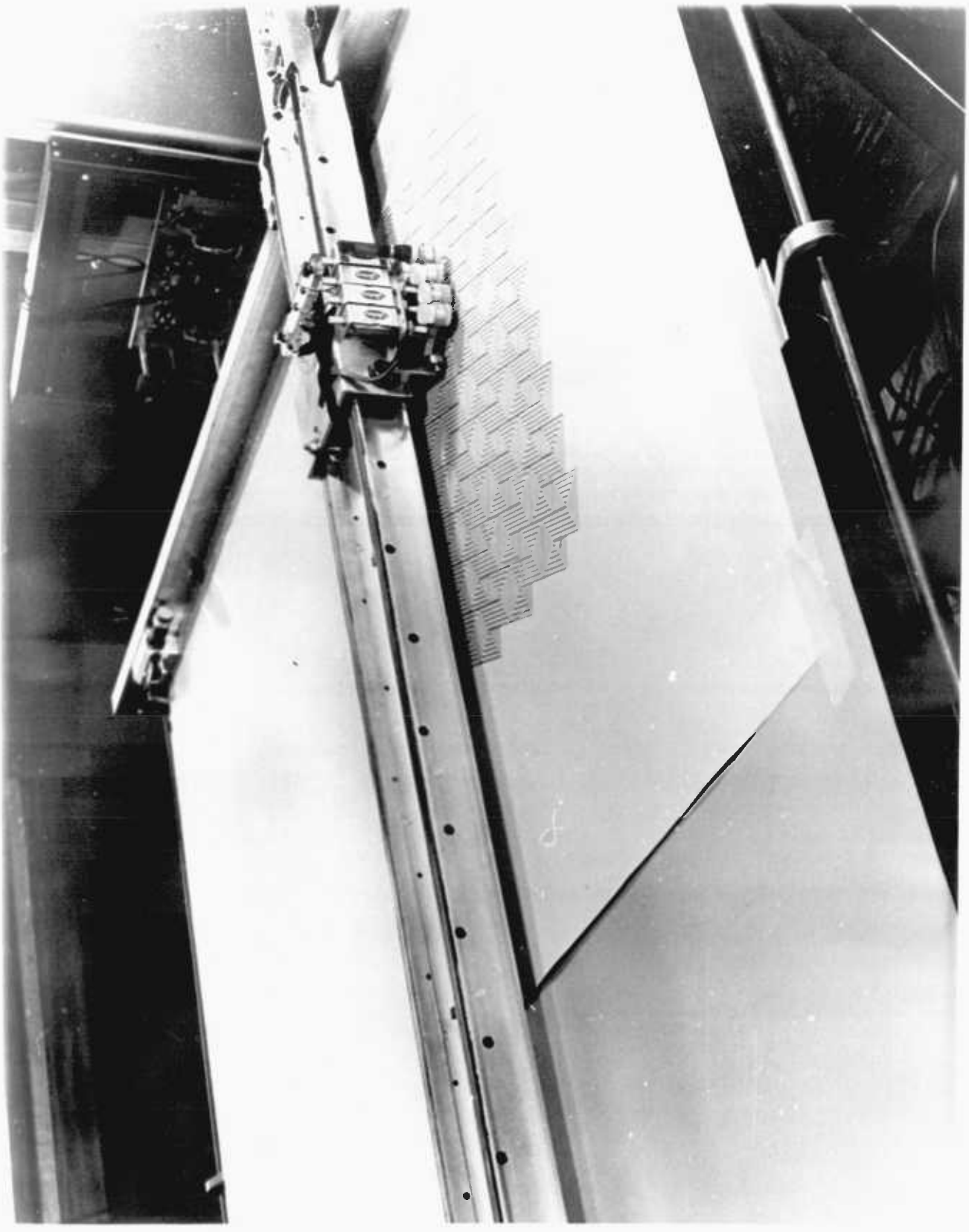


Figure 2.4 The flat bed plotter

digitiser/plotter. The gantry and carriage were also designed to be as light and rigid as possible. Special rubber wheels were used to reduce noise and proved so effective that the whole plotter is virtually inaudible. The average plotting speed is 10 inches per second and is as much limited by the flow of ink from the pens used as by the drive mechanism.

d) CAMAC interface.

The digitiser/plotter, storage tube display, line printer and D.C. plotter are all interfaced to the PDP 11/45 through a CAMAC[2] interface.

CAMAC provides a common standard interface (the CAMAC dataway) into which hardware handling modules can be plugged. An interfacing module, the CAMAC dataway controller, links CAMAC to the computer. This dataway controller must be designed for the computer being used but the hardware handling modules are all independent. These modules and the controller, in the form of printed circuit boards, are plugged into a CAMAC crate, which normally has space for 24 boards.

The following modules are present in the system:-

a) To interface digitising table.

1) Interrupt request register (EKCO 7013)

- Handles the operation of the eight function buttons on the pencil.

2) Dual incremental encoder module (S.E.N. 2019)

- Monitors the carriage and gantry positions.

3) Digitising table interface (D-MAC CS101).

- Turns pencil light on and off.
- Raises and lowers pen unit on the carriage.
- Switches between plot and digitise mode.

b) b) To interface Tectronix 611 storage tube.

1) Storage display driver (S.E.N. SDD 2015)

- Controls operation of the tube.

2) Vector generator (S.E.N. VG 2028)

- Provides ability to display linear vectors with a minimum number of instructions. Vectors may be any length and in any direction.

3) Character generator (S.E.N. CG 2018)

- Generates ASCII characters at any position on the screen with a choice of two character sizes.

4) Display driver (S.E.N. DD 2012)

- Accepts X/Y increment/decrement signals from the vector generator to display vectors on the screen.

c) To interface line printer (7065).

1) Peripheral Driver (7065)

- General purpose output module.

d) To interface the D.C. flat-bed plotter.

1) Digitising table interface (DMAC CS101)

- Raises and lowers pen unit.

2) Dual error module

- Monitors the difference between where the table should be and where it is.

The hardware is shown schematically in figure 2.5.

It should be noted that CAMAC was intended for use in situations where very many peripherals were to be interfaced to the computer. The controller which interfaces CAMAC to the computer is capable of controlling up to eight crates, each with twenty-four modules. In this application only a few peripherals need be interfaced and it is now possible to obtain interfaces from D.E.C. and other suppliers, which plug directly into the PDP 11/45, for all the peripherals mentioned. This has become cheaper and more efficient than using CAMAC.

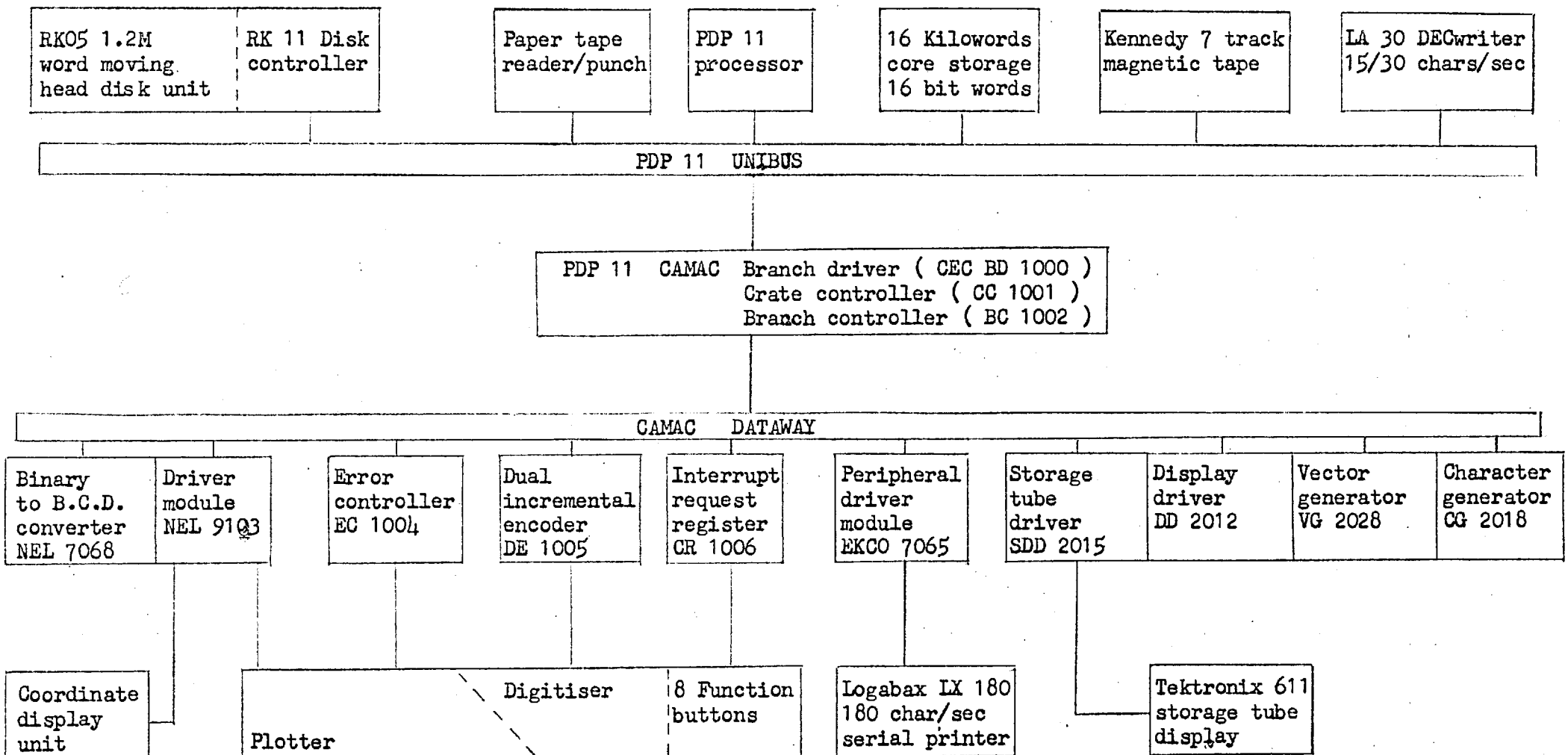


Figure 2.5 Schematic diagram of the system hardware

2.3 Software.

2.3.1 Introduction

The General Computer Aided Draughting System (GCADS) operates under the standard PDP 11 Disk Operating System, known as DOS, which is supplied by D.E.C.. This occupies the lowest 3.5Kilowords of core depending on the number of device handlers currently in use. Although GCADS does not use all the facilities available under DOS they are all retained because to remove them would take significant programming effort and might place severe restrictions on the operations of future applications programs.

The remaining core area, 12.5Kilowords, is divided into a resident and overlay area.

The resident area contains a small executive, some very commonly used subroutines and common areas.

All system and user programs are loaded into the overlay area from the moving head disk unit. A map of core usage is shown in figure 2.6.

The method of calling overlays provided by DOS was not considered to be of sufficient speed. To overcome this a special high speed overlay system was written. The operation of this is described in the next section.

The majority of system programs are written in Fortran IV. Subroutines only have been written in PAL 11 assembly language and only where either, the program effectively interfaces to hardware and assembly language must be used, or where not to do so would result in grossly inefficient operation in terms of execution time and required core storage.

The user controls the system from the digitiser. The digitising area is divided into two: the drawing area and the menu area (see Fig. 2.7). The menu is a 10 x 30 matrix of 20mm squares occupying the leftmost 200mm of the table. When a point on the table is digitised,

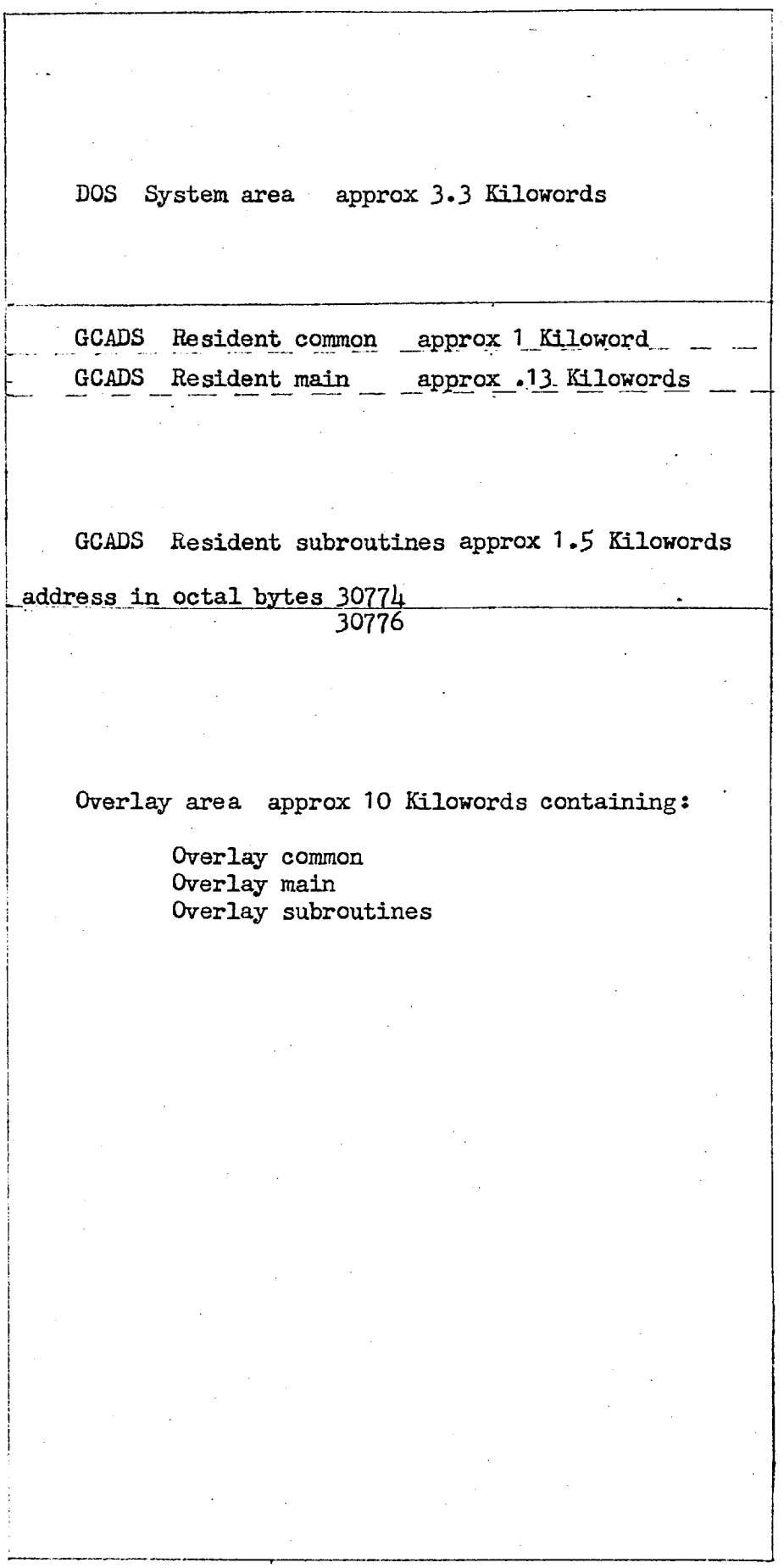


Figure 2.6 Core utilisation

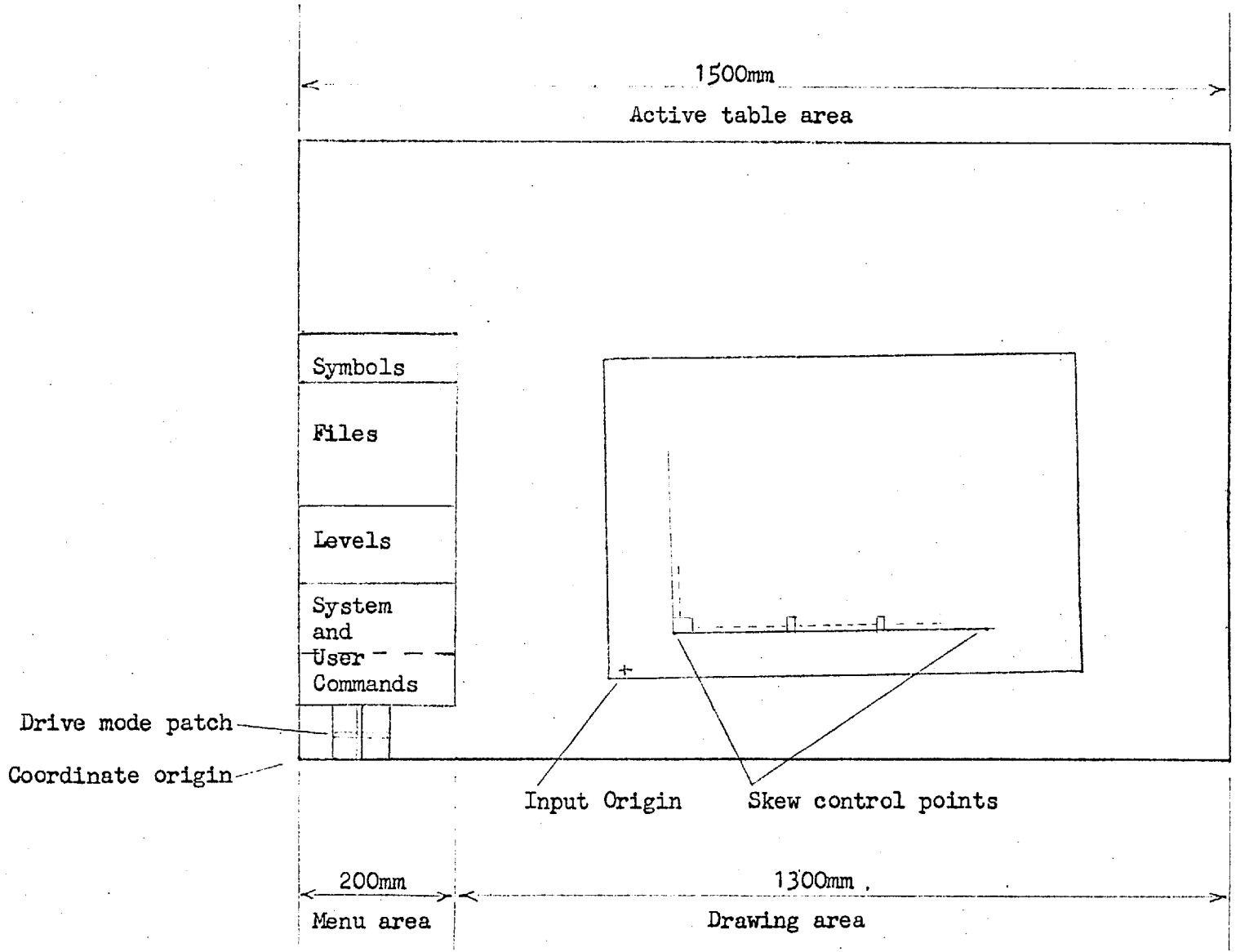


Figure 2.7 Use of the digitising surface.

the UOM determines from the coordinates of the point whether it is within the menu area. If so, it is interpreted as a command by the computer.

In this way the user can issue 'menu commands' without having to continually turn to the system console.

The software comprises a set of modules and utilities some of which are subroutines and some overlays. These are described in section 2.3.3. The UOM forms the nerve centre of the system and has responsibility for responding to menu commands and calling the correct module or utility.

The main functions that the system supports are:

- The input of straight line data (line mode).
- The input of special symbols including text, circles, dimensions and arcs (symbol mode).
- The sectioning of data into levels.
- The filing of data.
- The use of data files as sub-pictures or macros.
- The editing of all types of data.
- The plotting of all data.

During the input of data, several aids are available to the user:

Windowing- a specified area of the table is magnified to fill the display screen.

Control 90- the coordinates are constrained to be at an angle,which is a multiple of 90 degrees, to the last point digitised.

Control 15- the coordinates are constrained to be at an angle, which is a multiple of 15 degrees, to the last point digitised.

Drive mode - Instead of coordinates being calculated from the position of the free pencil on the digitiser, the user is able to increment the coordinates in, ones tens hundredths or thousands of millimetres, in x or y.

Find - The user is able to set the value of the coordinates to those of previously entered.

The programs for the modules and point utilities are described in section 2.3.3 and their operation in section 2.4.

2.3.2 Program communication and arrangement

Each time an overlay is called and loaded into core the previous contents of the overlay area are destroyed. In order to communicate between overlays three methods are employed:

1) The storing of information in the resident Common areas.

2) The storing of information in a disk file which is later accessed by another overlay.

3) The storing of information on the overlay execution stack.

The Common areas in the resident area of core are set out and their function explained in Appendix A-1. They contain; a set of system parameters, some space for applications programs parameters, 256 words of buffer space for the system and 512 words of buffer space for the application programs. A real variable occupies two words.

Two subroutines STORCM(N) and RESTCM(N) are used to store and retrieve the Common user buffer area to and from a contiguous random access file CADMAC.RAO on the disk unit. The argument N defines the block number in the file at which the writing and reading is to start. These subroutines enable the user to conveniently store and retrieve data between overlays. The system itself by saving the common user buffer area before use and retrieving it afterwards is able to utilise the Common user buffer area without affecting applications programs.

Several of the system overlays perform utility functions such as filing and display. The applications programs may wish to call such a utility virtually as a subroutine. In order to facilitate this an overlay execution stack has been created to work in conjunction with the high speed overlay system mentioned in section 2.3.1.. This is now described in detail.

The load module of each overlay is assigned an overlay number and is written into a contiguous random access file CADMAC.OVL on the moving head disk unit. At the front of CADMAC.OVL is a directory containing each overlay name, number, start record and length that has been written into CADMAC.OVL. When GCADS is run the directory, excluding the overlay names, is read into the resident core area.

Overlays are called into core by a statement CALL OVLINK (N) where N is determined by:

N = (Overlay Number - 1) x 25

Overlays may also be called by a statement CALL STACK (N, ARG) where N is determined as before and ARG is used to carry information to the overlay defined by N. Both N and ARG are placed on the overlay execution stack which operates on a Last In First Out principle. A statement CALL OVRETN causes the overlay defined by the value of N at the top of the stack to be called into core and the variable ISUB in common area SUBOV (see Appendix A-1) is set to the value of ARG.

ARG can be used either to transfer an item of data to the overlay called, or where the called overlay performs more than one function or contains more than one entry point, to define which function is required.

By testing the value of ISUB as the first executable statement of an overlay an individual segment can be selected for execution.

If the calling overlay stacks itself before stacking another overlay then control will be returned to it after the called overlay has been executed. This process is illustrated in figure 28. Therefore overlay segments may be called in a similar manner to subroutines but with two important differences:

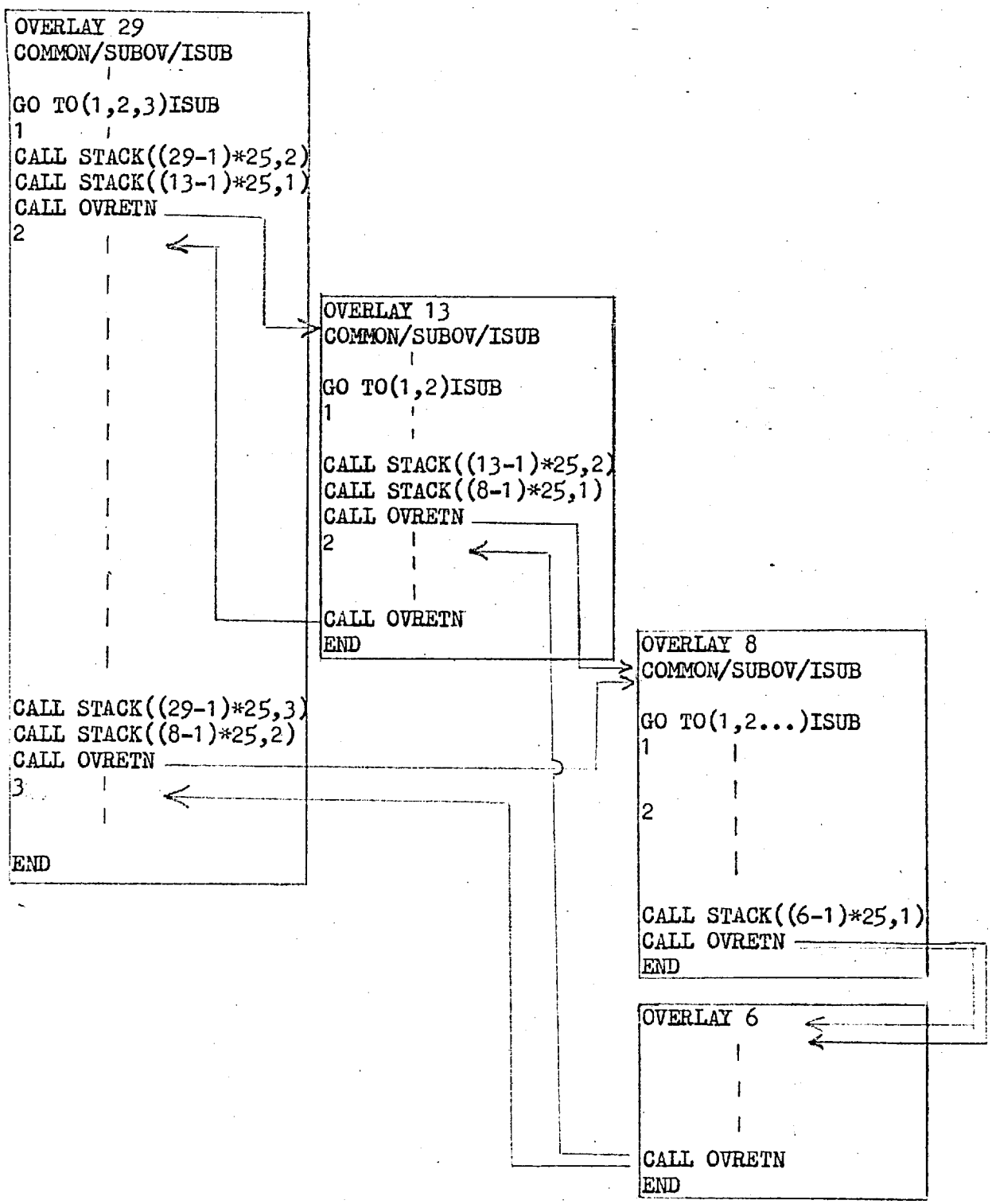


Figure 2.8 Example of overlay 'stacking'.

1) The arguments to the overlay segment called must be transferred by one of the methods previously outlined for communication between overlays.

2) All variables in the calling overlay that are required after such a call must be saved either in resident Common or on the disk unit.

Since each disk access will require on average 90 milliseconds it is preferable to store data for transfer or which is to be saved in resident Common. The resident Common storage locations thus have a high value placed upon them and are only used where essential. It happens that some parameters may only be used within a particular string of overlays and other parameters are only used within a different string of overlays, in such a case the same Common loacations will be used for the two sets of parameters. i.e. The system can be split into subsystems and the role of some of the variables in resident Common may change between subsystems.

The overlay execution stack may be cleared by a statement CALL CLRSTK If a statement CALL OVRETN is executed and the overlay execution stack is clear then the resident executive will call a special overlay, the User Operations Monitor (UOM) , which forms the heart of the system (see figure 2.9).

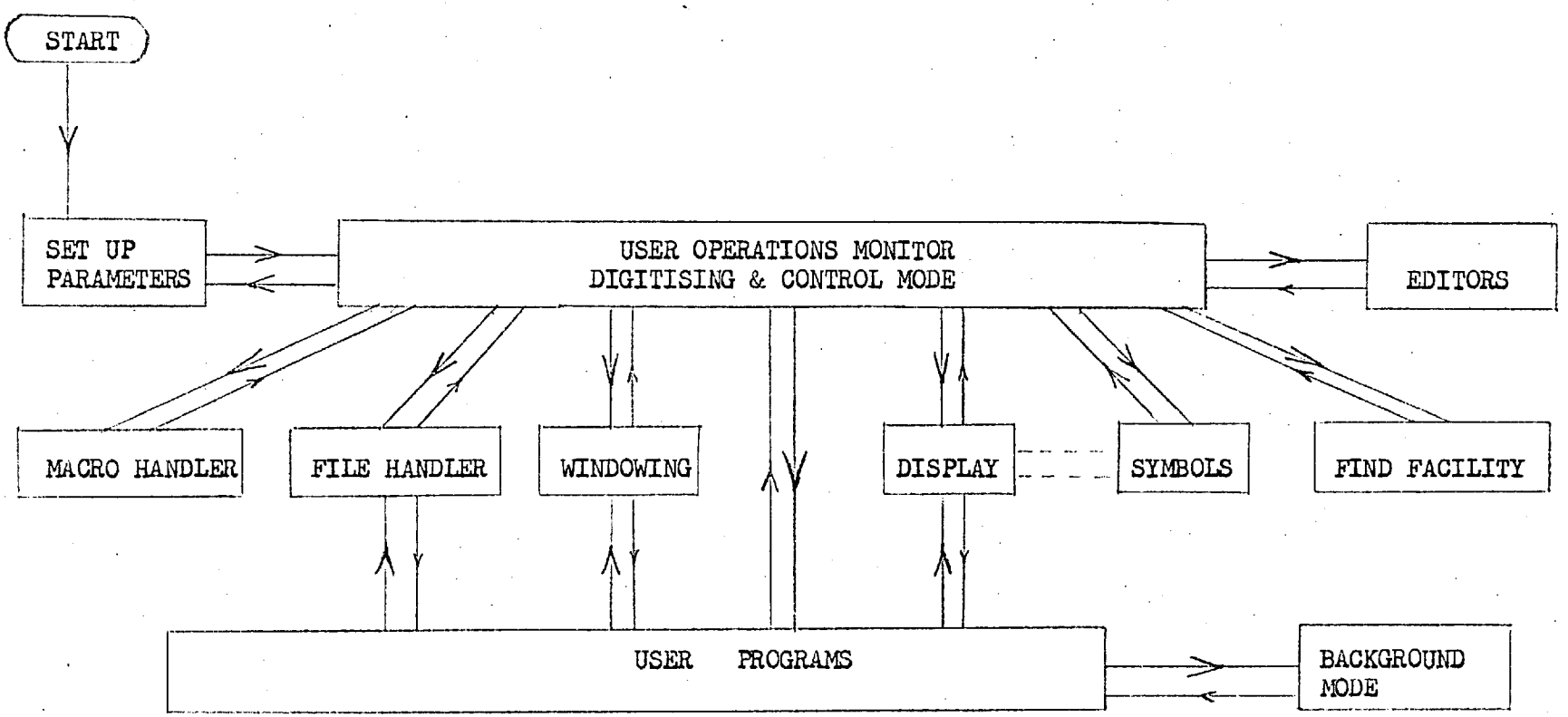


Figure 2.9 Block diagram of main links and communications in GCADS

2.3.3 System modules

a) Setup

On first entry to the system the resident executive calls the setup module. This is used to initialise system parameters. Any of these can later be reset by menu command.

These parameters are:

1) Table origin - the point from which all table coordinates will be measured.

2) Input (Drawing) origin - the point from which all data coordinates will be measured.

3) Skew - defines the angle between the drawing and table axes.

4) Input scale -

5) Output scale - data is plotted at a scale given by Input scale divided by output scale.

6) Alphanumeric size - defines the size of text that would be plotted if an output scale of unity was used.

7) Grid factor - coordinates are rounded to the nearest multiple of the grid factor.

The setup module also initialises the flag that indicates the sub-picture processor status (see sub-picture handler), and initialises the random access work files (see file handler).

b) User Operations Monitor.

After the setup module the UOM is automatically called. This is the real centre of the system and contains the background loop, drive mode, the interrupt handler, the menu command handler, the level handler

and symbol handler.

1) Background loop

When running GCADS the program that is most often under execution is the background loop. It performs the following:

- Reads the table coordinates.
- Converts them to real drawing coordinates.
- Rounds coordinates if a grid factoris set.
- Applies CONTROL* and TRAILING origin adjustments.
- Displays the current input level number and any messages set in the Common MESAGE area.
- Displays the cartesian and polar coordinates of the pencil relative to the last point digitised (TRAILING ORIGIN MODE) or to the drawing origin (ABSOLUTE ORIGIN MODE).
- Displays a tracking cross or cursor representing the current position of the digitising pencil.
- Looks for a pencil button interrupt.

Application programs may display up to five, nineteen character messages in the background loop by setting them in the last 25 variables in Common area MESAGE. The first 27 variables are reserved for system messages.

2) Interrupt handler.

When the operator presses one of the eight pencil buttons it is detected by the bacground loop and control passes to the interrupt handler. Buttons 2,5,6,7 are serviced by the UOM.

If button 1 is pressed, control is passed to the menu handler if the interrupt occurred in the menu area. If not and the system is in line mode (see symbol handler) it is serviced by the UOM. If the system is not

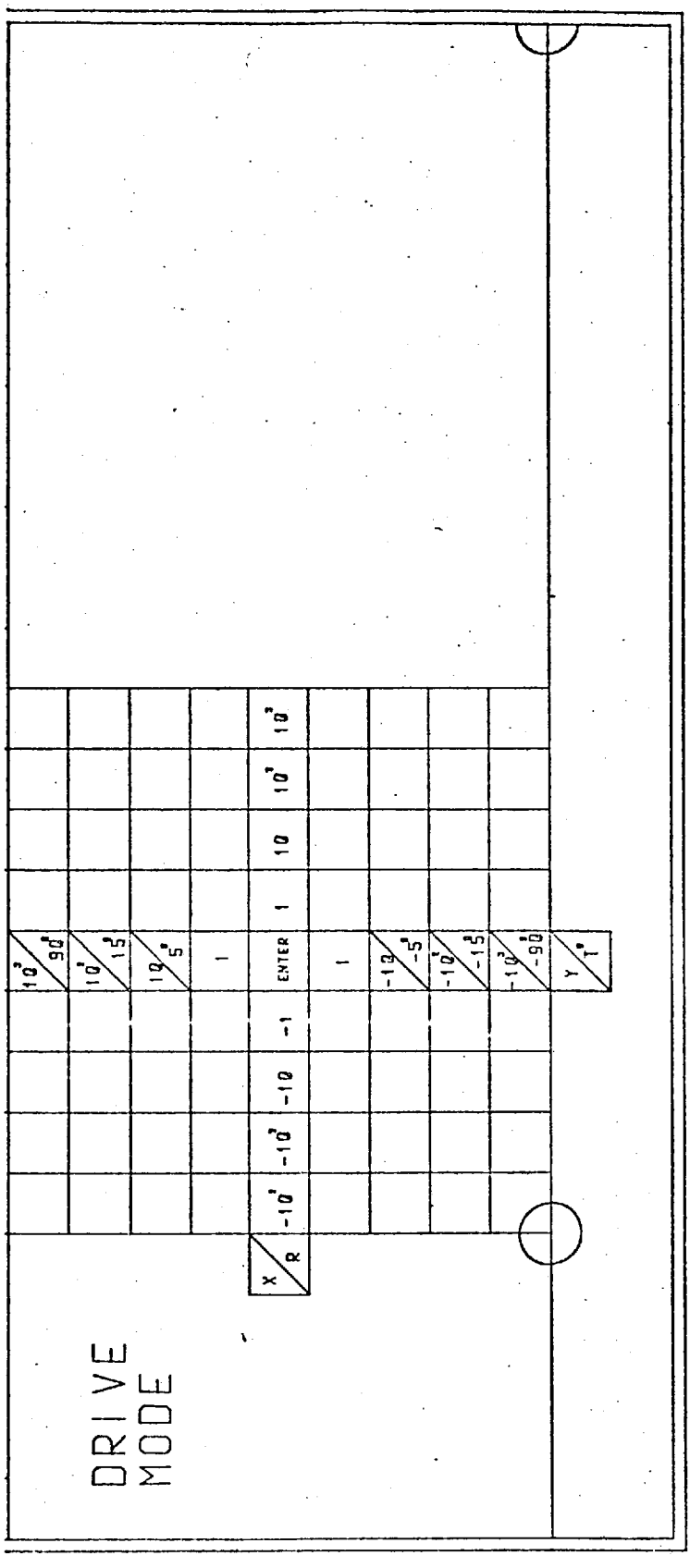


Figure 2.10 The drive mode patch.

in line mode, a CALL OVRETN is executed and the next stacked overlay will be called. This will be a symbol overlay.

Button 3 transfers control to the sub picture processor.

Button 4 transfers control to the 'window' overlay.

Button 8 is used to switch in and out of 'drive mode'.

When in drive mode only, the drive mode patch (Fig. 2.10) is recognised by the background loop.

3) Drive Mode.

In drive mode, the cursor is driven from the last point digitised in discrete intervals either in cartesian or polar coordinates. A 90mm square area of the digitising table at the bottom left corner of the table is used to control both the movement and position of the cursor and the value of the drawing coordinates. This enables accurate put of exact dimensions.

4) Menu handler.

The menu is divided into four sections (Figs. 2.11-2.14).

Commands	-	100	squares
Levels	-	60	squares
Files	-	100	squares
Symbols	-	40	squares

The Command area of the menu is arbitrarily divided into two sections: System Command area and user Command area.

The software makes no distinction between the system and user commands.

The menu square number is used to calculate which number overlay should be called. In order that the position of particular command squares

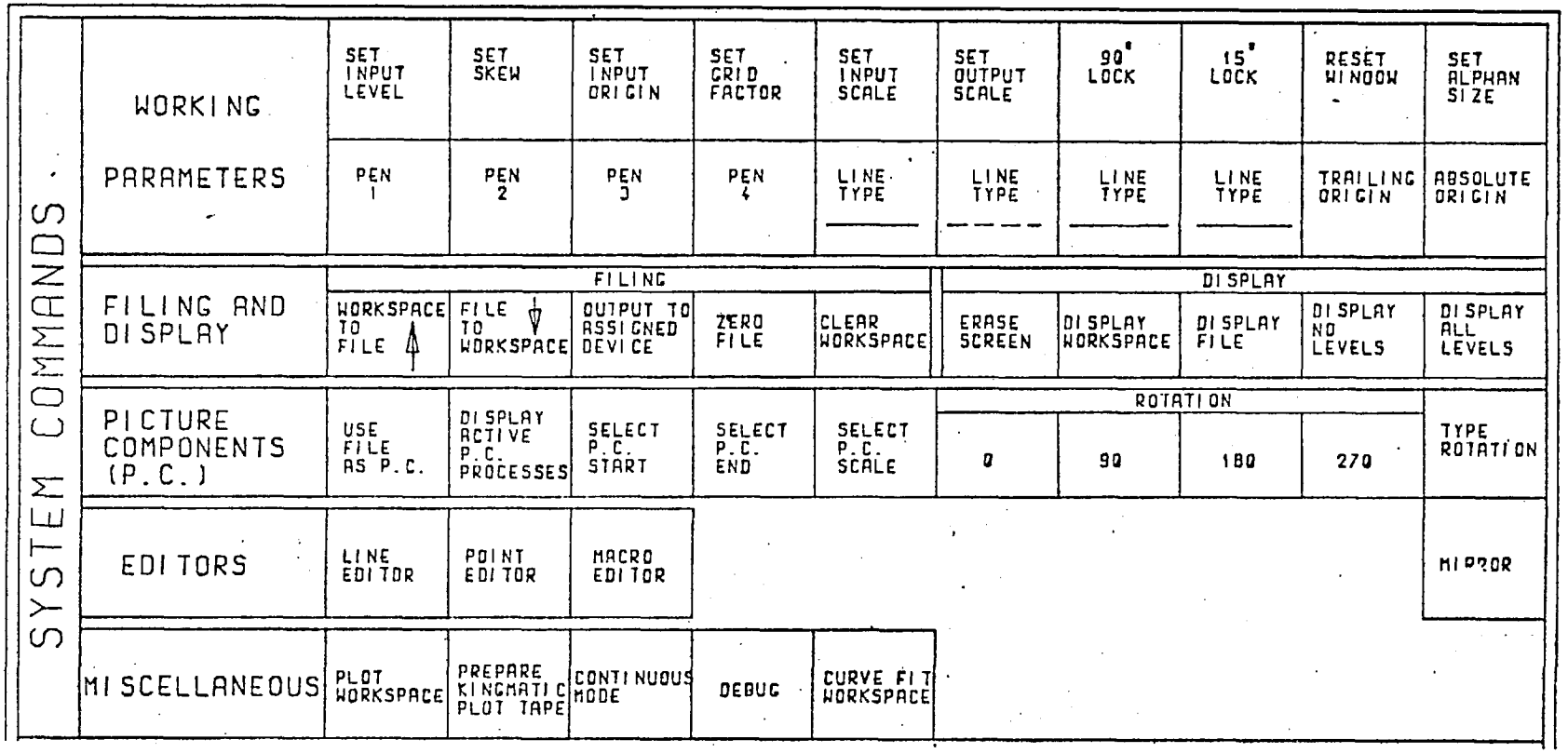


Figure 2.11a GCADS menu system commands.

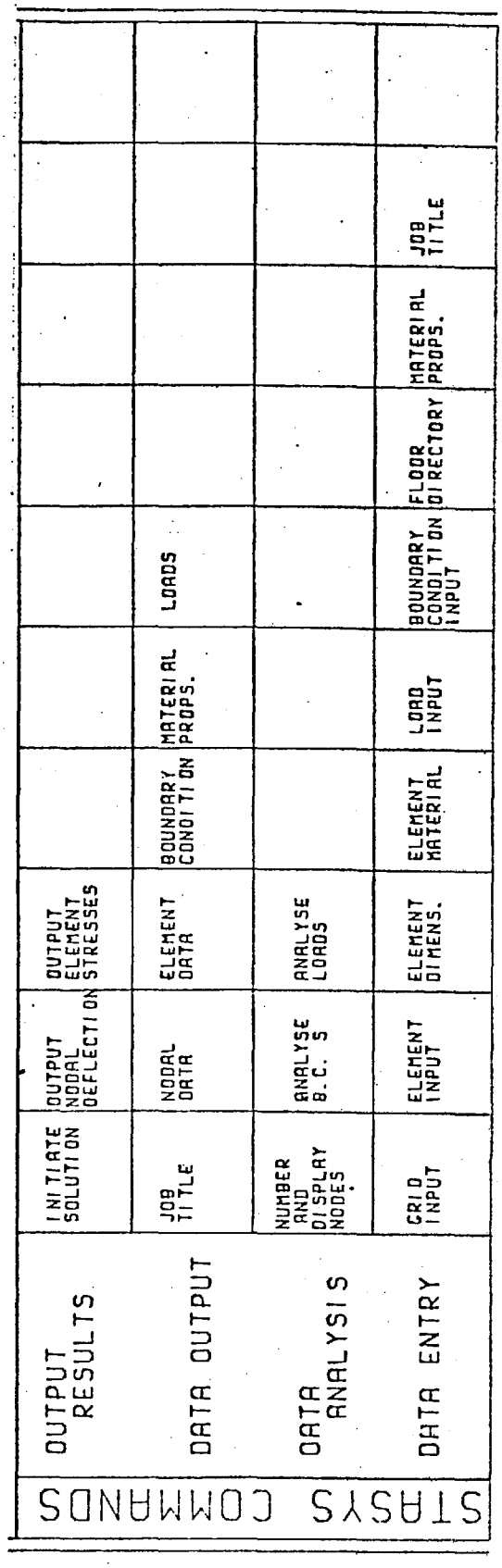


Figure 2.11b GCADS menu user commands.

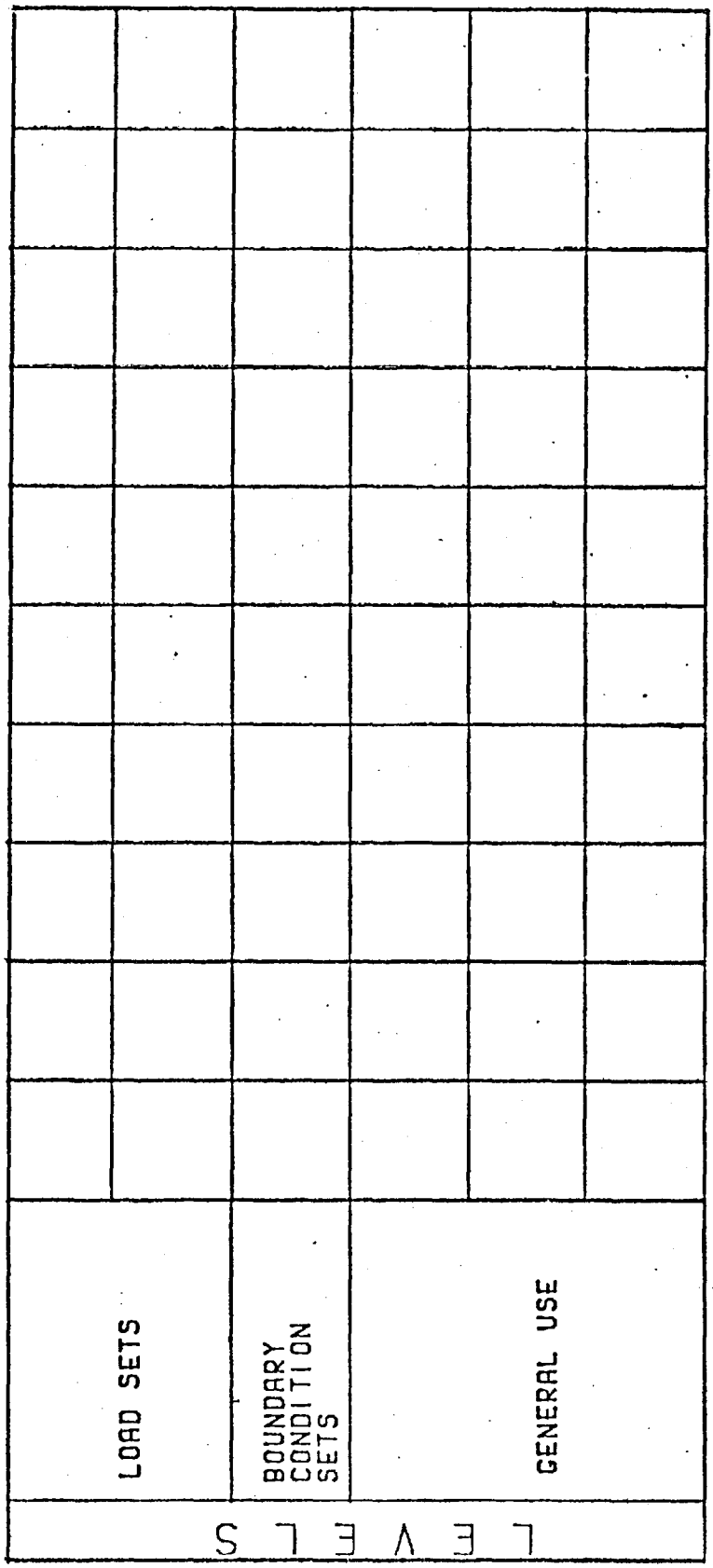


Figure 2.12 GCADS menu levels.

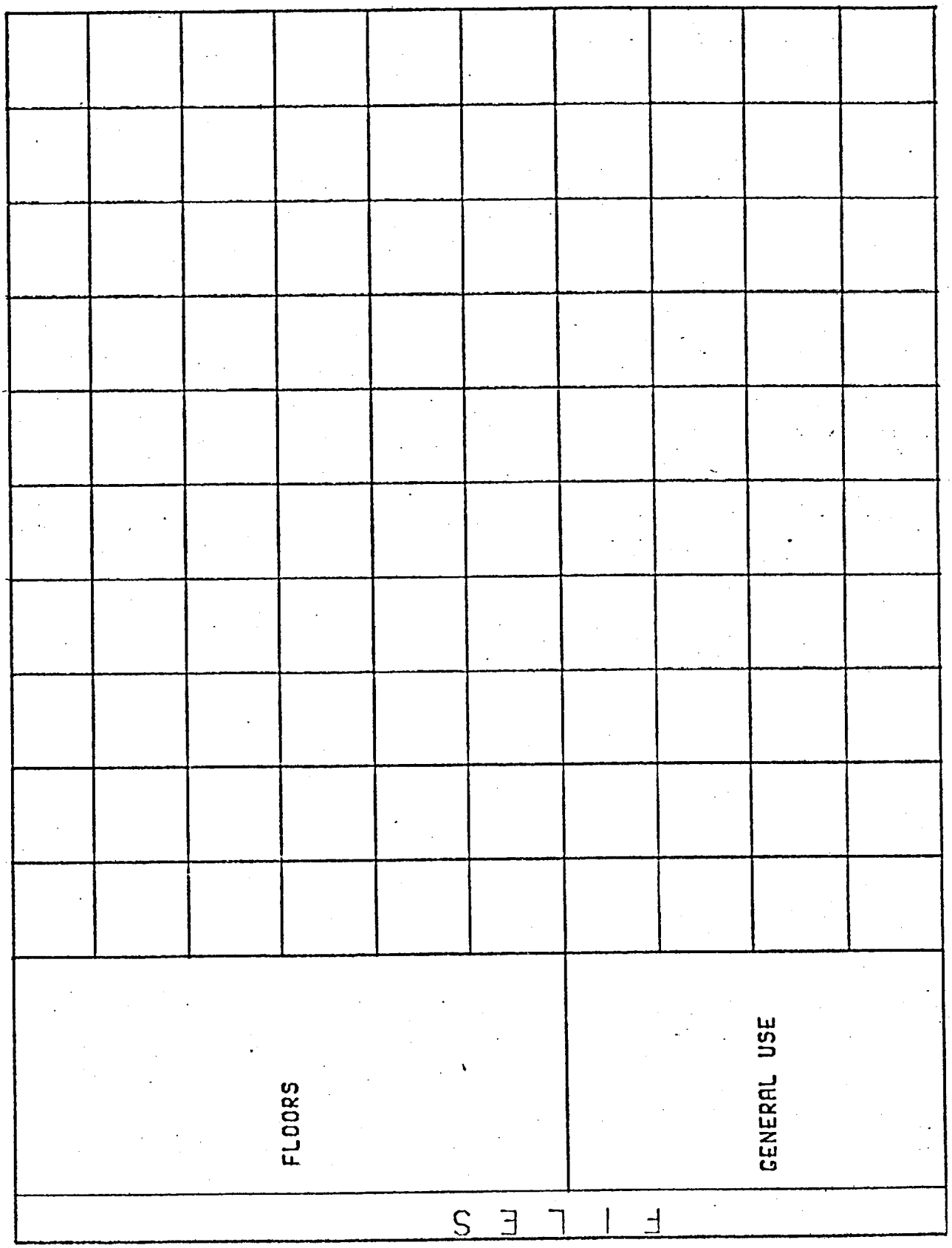


Figure 2.13 GCADS menu files.

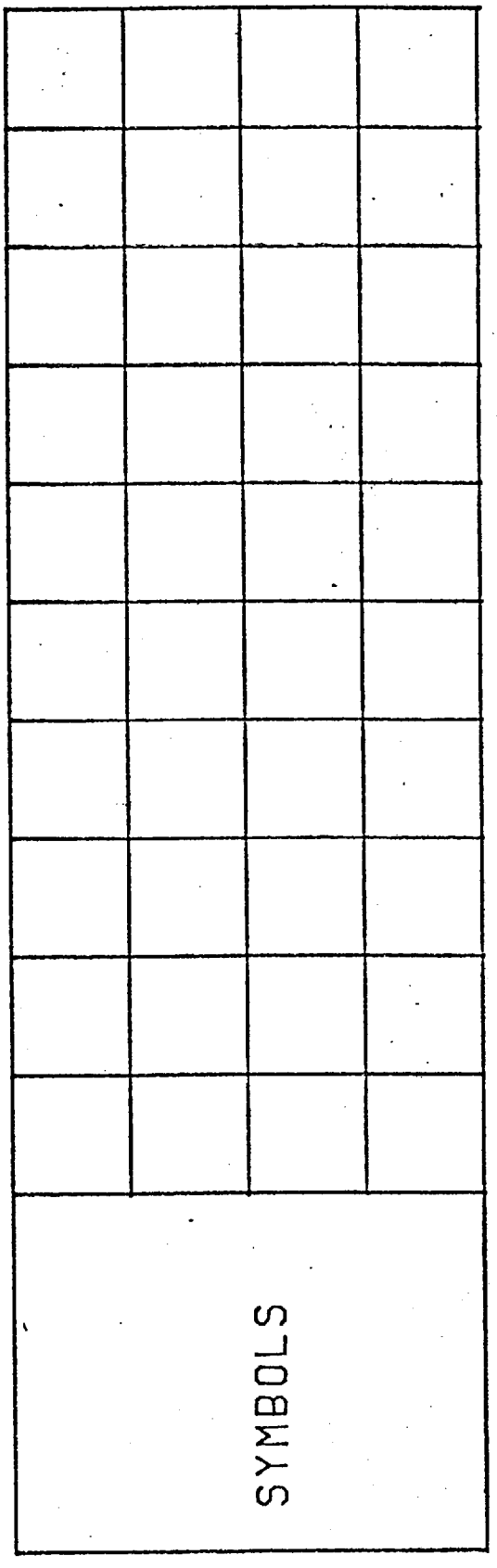


Figure 2.14 GCADS menu symbols

may be re-arranged at will without having to change many programs, a mapping system is used. This is shown schematically in figure 2.15. When a level, file or symbol square is digitised, the appropriate handler is brought into operation.

5) Level handler.

Levels are used to section data. Data may be input under 160 different levels, although only 60 different levels can be directly set from the menu.

Level markers are entered into the data stream by menu command. All data between a level marker 'n' and the next level marker is said to be on level n.

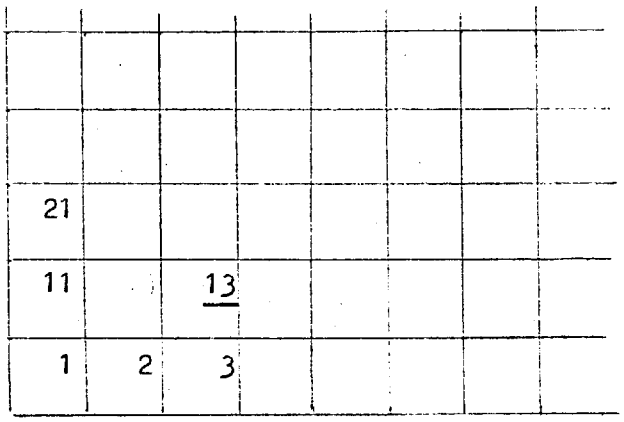
The level handler contains a 160 bit array - one bit corresponding to each level. If the bit corresponding to a particular level is set, then the data on that level is said to be active. If the bit corresponding to a particular level is not set, then the data on that level is said to be passive. Passive data is invisible to the user (ie. it will not be displayed or edited, but it will be filed).

Data levels can be declared active or passive by menu command (see section 2.4).

6) Symbol handler.

Symbols are used to describe graphic data items when geometry can conveniently be defined by only, a few points, and dimensional data (eg. circles, ellipses, rectangles). They are program generated and are used because:-

a) The item is best described mathematically and its dimensions may be required to be variable, e.g. circle.



Menu command squares on table.

```
STACK:   .BYTE  1.,6.,7.,8., ........
         .BYTE  3.,4.,5., .......
         .BYTE
           :
           :
           :
```

Points to position in MENSEL

Menu map in subroutine MENMAP

Overlay number - 1

```
STACK:   .BYTE  3., 9., 99.,23.,100., ......
         .BYTE
           :
           :
           :
```

Fixed menu map in software in subroutine MENSEL.

Figure 2.15 Menu mapping. Function stated in menu command square 13 is executed by overlay number 101.

b) To digitise the item each time it was required would be tedious and to use a macro would produce an excessive amount of data.

c) The item is used so often that writing a special program to generate it is justified, e.g. rectangle.

When the user digitises the first point of the symbol, the UOM, recognising that symbol mode is set, jumps to a statement CALL OVRETN.

Since the first segment of the relative symbol overlay is at the top of the overlay execution stack, it is called into core. This stores the point in common/symbol/ (see appendix A-1), stacks the next segment of symbol overlay and then returns control to the UOM.

This process is repeated until the last point of the symbol is collected, at which time the segment of the overlay that displays the complete symbol is called. Finally, control is handed back to the UOM.

At any time during the construction or entry of symbol points, the operator may return control to the UOM and re-enter 'line mode'.

c) File handler.

For reasons of speed the only type of file that the system directly uses are random access files on the moving head disk unit. There are two ways in which these are used: as working files or as mass storage files. A total of sixteen files may be defined and if two disk units are available the files may be arbitrarily spread between them.

The files that the system is to use are listed in the FILE.DAT which is created on disk unit zero. This is read by the SETUP module which then looks up all the files listed to check their validity and length. Each file is referred to by number in the programs, the number of a file being determined by its position in the list in FILE.DAT.

Data is written to a working file by a statement:

CALL RAWRIT (IFN, IREC, ARR, NW)

and read by a statement:

CALL RAREAD (IFN, IREC, ARR, NW)

where:

IFN - is the random access file number.

IREC - is the record number to be written to or read.

ARR - is the array in core from which data is to be read or written.

NW - is the number of words for transfer.

Each file is divided into 256 word records, if the number of variables for transfer is not specified then 256 words (128 single precision real or integer variables) will be transferred by default.

A mass storage random access file may contain many distinct sets of data each pointed to by an index at the front of the file. This index is initialised by a special overlay called by menu command. The number of entries in the index is specified at the time of initialisation.

The file handler allows data transfer between work files and mass storage files. GCADS uses only one random access file for mass storage, and two working files. CADMAC.MS1 is used to hold a set of 128 storage files. CADMAC.RA1 is used to hold all data as it is input and is known as the 'workspace', CADMAC.RA2 is used by the macro processor.

Four operations are carried out by the GCADS file handler:-

- Transfer of data from a working file to a transfer file
- Addition of data to a working file from a storage file
- The display of a file
- Deletion of a file

Three variables in resident common are used to transfer arguments to the file handler:-

- IONDIR (Common FILHND) defines which operation is to be performed.
- MNUM (Common MENU) defines the mass storage file number.
- ISUB (Common SUBOV) defines the random access file to be used.

IONDIR is set by digitising one of four menu commands and MNUM by pointing to one of hundred file squares on the menu. Thus each file square represents a mass storage file.

The file handler is a powerful tool for use by applications programs.

d) Display & Plotting Module.

This module displays on the storage screen or plots on the flat bed plotter data in the workspace. If any symbols are present in the data, then the symbol overlay display segments are called. This can considerably reduce the speed of the display. The scale of display is determined by parameters set by the 'windowing' module.

e) Window module.

Window parameters define the area of the digitising table which is to be displayed or plotted.

f) Macro Processing Module.

Any of 100 storage files represented by the file squares on the menu can be manipulated by the macro processor and added to the workspace or another storage file. Data that have been added to the workspace in this way are referred to as macros. Macros may be rotated, mirrored, and scaled and start and end points may be selected. The transformation that the macro processor will apply to a file is defined by a set of status flags in common area MACRO. These define:

a) The angle through which the macro will be rotated.

b) Whether a start point will be selected. When the macro is added to the workspace the start point will be positioned at the trailing origin. The other points in the macro will be translated accordingly.

c) Whether and end point will be selected. When the macro is added to the workspace the trailing origin will be set to the position of the end point.

d) Whether the macro should be scaled.

Only data on levels that are active will be processed by the macro process. Data on passive levels is discarded.

Macros enable easy production of similar parts without having to

digitise each part separately. By creating a library of the commonly used components in the 100 storage files the production of working drawings may be considerably speeded.

g) Editors.

There are three editors: the point editor, the line editor and macro/symbol editor. Each is a separate overlay called by menu command. Their operation is described in section 2.4.

h) Peripherals Input/Output Module.

Allows the input or output of data to or from the workspace, from or to any data set. The data set can be on disk, paper tape or magnetic tape. A permanent record of the workspace can be achieved.

2.3.4 Data Storage

Much research has been done on the efficiency and speed of in-core data structures and list processing.[4,5] However in GCADS the data is stored, both on disk and in-core. The designer of an in-core data structure is concerned to minimise the number of memory references and the number of memory locations required to solve a particular problem. A memory reference takes approximately one microsecond, by comparison, a disk reference may take on average seventy milliseconds - seventy thousand microseconds. Clearly with a disk as the main storage medium the data structure must primarily be planned to minimise the number of disk references. Furthermore, disk storage is much cheaper than core storage and therefore less emphasis is placed on trying to minimise the amount of storage space required by the problem and more emphasis on maximising the speed of operation.

It is evident that the larger the quantity of core available for data storage, in future referred to as buffer space, the smaller will be the number of disk references required to process a given quantity of data, and the faster will be solution. On the other hand any core used for storing data can not be used for storing program instructions. A compromise is necessary.

Different objectives can be achieved by varying different factors. The following objectives are listed in order of importance:

1) Maximise speed.
   - Minimise disk accesses.
   - Maximise buffer space.
   - Allow the storage of redundant data to reduce operations.
2) Reduce storage cost
   - Minimise core storage capacity
   - Store minimum data.
3) Enable easy expansion and user interface.
   - Keep it simple.

An obvious conflict exists over the amount of core space that should be allocated for storing data.

The efficiency of a particular type of data structure depends on the quantity of data which must be handled, just as the method chosen for the manufacture of a component depends on the number of components to be produced. It is difficult to quantify just how much data a computer aided draughting system will be required to handle. Assuming that a typical drawing consists of 2000 lines meeting at 700 points.

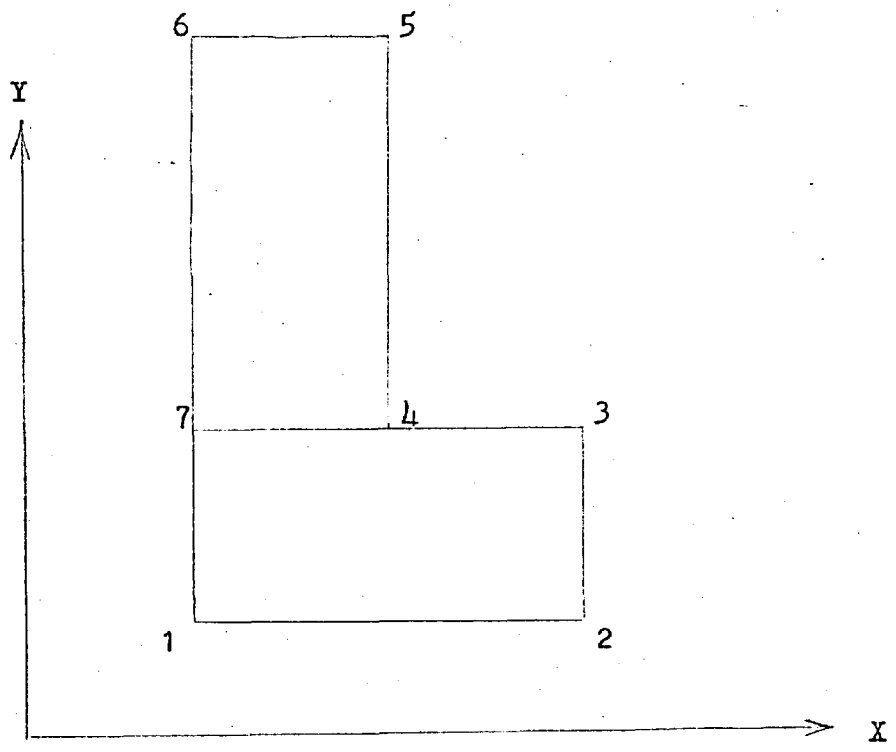
Two basic storage alternatives present themselves:

1) A list of points defining a series of continuous straight lines with a marker to indicates a break in the series.
2) Separate lists of point coordinates and coordinate connections.

These two structures are illustrated in figure 2.16 and compared in figure 2.17. It can be seen that although the single list requires more storage space it can be displayed much faster than a system using one list of coordinate connections and another of the coordinates.

A single list structure has therefore been adopted in GCADS. In order to make fortran programming and user interface even easier an integer marker is associated with each pair of coordinates in the list. Point coordinates are stored as single precision real variables in units of tenths of a millimetre.

The integer marker is referred to as an I code and a list of the I codes used by the system is given in figure 2.18. The chosen buffer size is 128 variables (256 words) for easy disk handling. The 128 variable data block is divided into a 40 integer variable array, two 40 real variable arrays and 8 free real variables. Since the two 40 real variable arrays most commonly hold X and Y coordinate data the three 40 variable arrays are referred to as I, X and Y.



List structure

X	Y
100.	100.
300.	100.
300.	200.
200.	200.
200.	400.
100.	400.
100.	200.
100.	100.
marker	
100.	200.
200.	200.

Coordinate connections	
1	2
2	3
3	4
4	5
5	6
6	7
7	1
7	4

Coordinates X	Y	N
100.	100.	1
300.	100.	2
300.	200.	3
200.	200.	4
200.	400.	5
100.	400.	6
100.	200.	7

Figure 2.16 Two approaches to data storage

		List		Connections & coordinates	
		Best	Worst	Best	Worst
STORAGE	Coordinate pairs	2001	4000	700	700
	Markers & pointers	0	1998	4000	4000
	Total locations	4002	9998	5400	5400
DISK ACCESSES FOR DISPLAY	1 buffer 100 locs.	41	100		
	2 buffers 50 locs.			108	4080
	Total* display units.	45	110	113.5	4085.5
	1 buffer 400 locs.	11	25		
	2 buffers 200 locs.			27	4020
	Total display units.	15	35	32.5	4025.5

*One display unit is approximately 70 milliseconds.
Time to process each location is taken as .001 display unit.

Figure 2.17 Comparison of two types of storage structure.

The possible contents of the I, X and Y triplets are shown in figure 2.19.

One method of reducing the volume of data is to store repeated sequences only once. Repeated sequences of data, referred to as macros in the following discussion, could be stored in a separate list and referenced by a special I code in the data list accompanied by parameters defining the scale, rotation and translation at which they should be displayed. The use of such a system has disadvantages, namely:

1) Editing the contents of a macro after it has been positioned on a drawing becomes difficult.
2) The display routine has to transform the macro data and this would increase display time.
3) Searching for a point within a macro requires that the same transformatio be carried out to calculate the position of the points within it thus increasing the time required to find a point.

The macros are therefore stored in their full form in the data list in GCADS. In some applications where the macros are clearly defined and will not require editing it may be useful for the application programmer to develop such a system.

Macros can contain any type of data and macros and symbols may be nested in the same manner that DO loops may be nested in Fortran. This fact is exploited by the macro editor (section 2.4.6).

A typical piece of data is given and explained in figure 2.20.
It was found that when large quantities of data needed to be displayed, the display time ran into many seconds, particularly when many symbols were included in the data. This was found to be most irritating to the user when the data was windowed and only part of it needed to be displayed on the screen. Since all the data had to be processed to discover whether it would be on or off the screen there were periods when nothing appeared to be

Code	Significance
0	End of data
1	Start of line
2	Point on line
7	Start of new data level
9	Null data (produced by deleting data)
11	As 1 (produced by line editor)
13	Start of a symbol
14	Point used to define the symbol
15	Data defining bounding rectangle of symbol
16	Last data of symbol
17	Start of macro
18	End of macro
19	Non coordinate data
20	Pause. Display is suspended until a pencil button interrupt is detected.
21	Start of new line type
22	Start of new pen

Figure 2.18 Icodes used by GCADS

Icode	Xcode	Ycode
1	X coordinate	Y coordinate
2	X coordinate	Y coordinate
7	Level number	-
9	-	-
11	X coordinate	Y coordinate
13	Symbol number	-
14	X coordinate	Y coordinate
15	X coordinate	Y coordinate
16	Non coordinate data	Non coordinate data
17	Macro (File) number	-
18	-	-
19	Non coordinate data	Non coordinate data
20	-	-
21	Line type 1-4	-
22	Pen number	-

Figure 2.19 Contents of X and Y-codes in GCADS

Icode	Xcode	Ycode	
7	60.	0.	Set default level 60
7	1.	0.	Set level 1
21	1.	0.	Set solid line type
22	1.	0.	Set pen number 1
17	1.	0.	Start of macro created from file 1
1	100.	100.	
2	600.	100.	
2	600.	200.	
2	100.	200.	
2	100.	100.	
18	0.	0.	
7	2.	0.	Set level 2
13	3.	0.	Start of circle symbol number 3
14	50.	150.	Centre point of circle
14	100.	150.	Point on circumference of circle
15	50.	150.	Lower left corner of bounding rectangle
15	100.	200.	Top right corner of bounding rectangle
16	0.	0.	End of circle symbol
13	3.	0.	Start of second circle symbol
14	650.	150.	
14	600.	150.	
15	600.	100.	
15	700.	200.	
16	0.	0.	End of second circle symbol
0	0.	0.	End of data

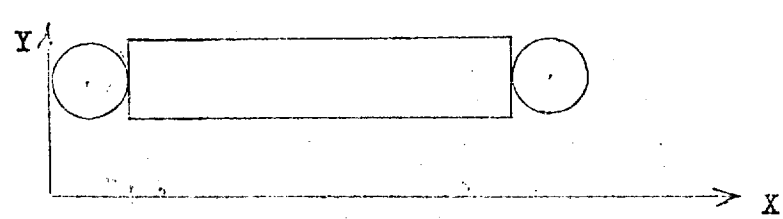


Figure 2.20 Example of GCADS data

happening. To improve this situation four of the eight free variables in each data block have been used to store a bounding rectangle of the data within the data block. Each symbol also contains a bounding rectangle. The display overlay checks the bounding rectangle of each block and if the rectangle does not overlap the display area the block of data is not processed. Similarly, if the bounding rectangle of a symbol is not in the display area the symbol processing overlay is not called.

The introduction of bounding rectangles has considerably speeded display but required considerable programming effort. Symbol bounding rectangles not only have to be calculated at the time of input but also have to be revised after editing. The bounding rectangle of each data block is recalculated every time the data in the workspace is transferred to a semi permanent mass storage file.

This experience has sustained the belief that maintaining a simple data structure is most important, particularly where application programs must be able to use the same database. However speed of operation must be maintained in an interactive system if operation is not to become tedious and the introduction of bounding rectangles has proved worthwhile. It should be mentioned that data added without bounding rectangles will be displayed so that they can be ignored by the applications programmer who does not want the added complication.

2.4 Operation of GCADS.

2.4.1. Setup.

GCADS operates under DOS V008[3]. To run the GCADS system the user must be logged in under User Identification Code (UIC) 2,2. The system is loaded by typing RU GCADS. The resident executive is loaded into core and immediately calls the first SET UP overlay.

On starting the system, the message 'ZERO TABLE' is displayed on the screen. The user must respond by digitising the lower left hand corner of the menu. The point digitised defines the absolute table origin - the point from which the pencil position will be measured - and if it is not at the lower left corner of the menu, menu commands will not be correctly interpreted.

After the table is zeroed, the following message is displayed:

1) USE ALL DEFAULT VALUES.

   SKEW - HORIZONTAL

   INPUT ORIGIN - B.R. MENU

   GRID FACTOR - 0.

   INPUT SCALE - 1.

   OUTPUT SCALE - 1.

   ALPHAN SIZE - 3mm.

2) SET ALL PARAMETERS.

This gives the user the option to use the default values of the working parameters by pressing button 1; or to set them individually by pressing button 2.

If the user chooses to set all the parameters, then the following sequence is entered:

1) Message SET INPUT ORIGIN (Screen)

   Action Digitise a point below and to the left of all the input data. If the data is on a drawing the point should

be marked for future reference.

Message ERROR - INPUT ORIGIN ON MENU (Screen)

This will only be displayed when the point digitised as the input origin is over the menu area. The error message is displayed for a fixed period and will then revert to SET INPUT ORIGIN.

2) Message DIGITISE HORIZONTAL LINE FOR SKEW CONTROL FIRST POINT (Screen)

Action Digitise left hand end of a line parallel to the X-axis of the data.

Message DIGITISE SECOND SKEW CONTROL POINT (Screen)

Action Digitise right end of skew control line.

The data x-axis will be defined as the line passing through the input origin and paralell to the line defined by the two skew control points. All data will be transformed accordingly. The skew points should be clearly marked on the drawing to be digitised. If at one time a quantity of data is taken from a drawing and subsequently more data is required, the two sets of data will be exactly compatible, regardless of the position of the drawing on the table, so long as the same input origin, skew control points, and input scale are used.

Message ILLEGAL SKEW (Screen)

Only displayed when the left hand skew point is to the right of the right hand skew point. After a fixed time the message reverts to DIGITISE SKEW CONTROL POINTS - SKEW ORIGIN.

3) Message ----.-- GRID FACTOR (MM) (DECwriter)

All drawing coordinates are rounded to the nearest multiple of the grid factor.

Action Type the grid factor required in the format shown. The grid factor is measured in millimetres.

4) Message ----.-- INPUT SCALE (DECwriter)

Action Type the scale of the data to be input.

5) Message ----.-- OUTPUT SCALE (DECwriter)

The output scale defines the size at which data will be output. An output scale of unity means that data will be plotted at the same scale as that at which it was input.

The input scale divided by the output scale gives the scale at which data will be plotted.

Action Type the scale at which data is to be output.

6) Message ----.-- ALPHANUMERIC SIZE (MM) (DECwriter)

The size at which text would be plotted if the output scale is set to unity.

Action Type the required text size in millimetres.

After accepting the default values or on completion of the above sequence the User Operations Monitor is called into core. When the digitising pencil is over the drawing area a tracking cursor is displayed on the screen, a message stating the current input level is observed at the top right corner of the screen and at the lower right half of the screen the cartesian and polar coordinates of the pencil relative to a trailing origin are displayed. In the absence of any other messages, the user is free to use any of the menu commands or to start digitising data.

The SET UP overlays will have set some default values without consulting the user. These may be changed by the menu commands listed below:

1) 90^{O}LOCK
15^{O} LOCK
In CONTROL MODE lines are constrained to be at angles which are a multiple of 90^{o} or 15^{o} to the x-axis depending on whether 90^{o} LOCK or 15^{o} LOCK has been selected. 90^{o} LOCK is set by default.

2) TRAILING ORIGIN
ABSOLUTE ORIGIN
The cartesian and polar coordinates displayed on the screen may be relative to the last point digitised (trailing origin), or relative to the drawing input origin (absolute origin). Trailing origin mode is set by default.

3) LINE TYPE
There are four line types available and these are set by four menu commands. The type of line that each menu command will set is displayed in each menu command square. A solid line is set by default.

4) PEN
Four pen numbers may be set from four menu commands. These pen numbers correspond to the four pen units on the flat bed plotter and enable different line thickness or colours to be used during plotting.
Pen unit 1 is set by default.

5) SET INPUT LEVEL
The level at which data is being added to a drawing is defined by digitising the SET INPUT LEVEL command followed by a level menu square. It is usually essential for the user to maintain a discipline over the input levels of various types of data. It is therefore wise to use this command after a CLEAR WORKSPACE which sets the input level to the default level 60.

2.4.2 Data Input.

Data input takes place either in line mode or in one of the symbol modes. The current input mode is displayed in the top right corner of the screen where most system messages are to be found. If no mode is specified then the system is in line mode. When in an input mode and whilst the pencil is over the drawing area of the table (not over the menu area) a tracking cross is displayed on the screen and the cartesian and polar coordinates of the pencil. The user is able to issue any of the menu commands while in an input mode.

In line mode all digitised points are connected by lines. A point is normally digitised by pressing button 1 on the pencil a line is drawn between the last point digitised (the trailing origin) and the current point. The new point becomes the new trailing origin and when in trailing origin mode the cartesian coordinates will be set to zero at the new point.

In order to break the series of lines produced by digitising several points using button 1, a button 2 should be pressed. After a button 2 no line will be drawn between the current trailing origin and the next point digitised. Thus button 2 is used to 'break' a line. If a button 2 was the last button pressed then a message 'LINE BROKEN' will be displayed in the top right corner of the screen.

A symbol mode is entered by digitising the appropriate symbol mode menu square. A message will indicate which mode is in use and a further message will inform the operator of the significance of the next point to be digitised (eg. the centre of a circle or the start point of a text string etc.).

To aid the data input and construction several utilities are available and can be called by pressing a pencil button:

1) Windowing.

Upon first entry to User Operations Monitor the screen represents the whole drawing area of the digitising table. Since the screen is only one twelfth the size of the table, the data is displayed at a much smaller scale than that at which it is input. Thus details are not clearly visible. To overcome this problem, the user can select an area of the table which he wishes to view at a larger scale. The procedure is:

a) Press button 4.

A message 'LEFT HAND WINDOW POINT' is displayed.

b) Digitise the bottom left hand corner of an imaginary square that will cover the desired area of the table (Button 1).

A message 'RIGHT HAND WINDOW POINT' is displayed.

c) Move the pencil from left to right across the table towards the bottom right corner of the imaginary rectangle. As this is done a rectangle is displayed on the screen, when this is seen to enclose the required area, digitise the right hand window point. The data within the rectangle will be displayed on the full area of the screen.

The menu command 'RESET WINDOW' is used to return to the situation where the screen represents the whole drawing area of the digitising table.

2) Angular Control.

In control mode lines are constrained to be at angles which are a multiple of 90° or 15° to the x-axis depending on whether 90° LOCK or 15° LOCK was the last 'LOCK' menu command selected. Control is switched on and off by successive presses of button 6. When control mode is set

a message is displayed with the other system messages in the top right corner of the screen, either 'CONTROL 90 MODE' or 'CONTROL 15 MODE'.

3) Finding.

The 'Find' facilities of button 7 makes it possible to join new data with existing data and to position the trailing origin accurately at a known position.

Pressing button 7 causes the following messages to be displayed on the screen:

Button 1 Searches for a point.

Button 2 Puts cursor on a line

Button 3 Finds intersection of two lines.

If the user wishes to join a line to an existing point or to find a point for placing a macro, etc. then the cursor is moved close to the point and button 1 pressed. If there is no point within a 5mm square centred on the pencil position, the message 'NO NEAR POINT' is displayed on the screen for a short time.

If it is wished to find a position on an existing line, then the cursor is moved to the required position and button 2 pressed. This may be done in or out of 'CONTROL MODE' as desired. Finding a position on a line with the 'LINE BROKEN' message on the screen causes the point on the line found to be perpendicular from the point digitised. However, finding a position on a line while drawing a line causes the point to be found to be the intersection of the two lines.

The third button enables the user to find the intersection of the two lines when there is no previously defined point at that intersection.

The following summarises the main uses of 'Find' facility:

a) To ensure that data joins up correctly.

b) By using a break line before finding, to position the trailing origin at a known point in order to:-

1) Use it as a starting point when entering drive mode.

2) Measure distances.

3) Use the position as an origin for a macro.

4) Drive Mode.

Drive mode allows the user to specify the location of points with precision. In this mode the coordinates are driven in millimetre increments from the trailing origin by positioning the pencil over the 'drive patch' on the menu and pressing button 1.

Drive mode is switched into and out of by successive presses of button 8. On entry to drive mode, the user selects X-Y (cartesian) or R-THETA (polar) drive mode by pressing button 1 or 2. When in drive mode, a message 'DRIVE MODE' is displayed in the top left corner of the screen, and the cursor jumps to the position of the trailing origin. It is no longer related to the position of the pencil on the table.

The drive mode patch, positioned in the bottom left corner of the table, is a 9 by 9 matrix of 10mm squares. When the pencil is positioned over the central square points may be entered by the use of buttons 1 and 2. If the pencil is positioned over any other square, the coordinates will be incremented by an amount determined by the position of the square relative to the central square, every time button 1 is pressed.

In this way, when in cartesian drive mode, the x and y coordinates can be incremented in ones, tens, hundreds and thousands of millimetres, and when in polar drive mode the radius is again incremented in ones, tens, hundreds and thousands of millimetres while the angle may be incremented in ones, fives, fifteens and nineties of degrees.

Of the two remaining buttons, whose sue has not yet been described, button 3 is used to hand control to the macro processor and button 5 has a floating assignment. Because it takes much longer to move the pencil

to a menu command square than to press a button it was decided to give the user the facility of choosing which menu command he would most like to be able to call merely by pressing button 5. Button 5 is set up so that it is equivalent to the command 'LINE MODE'. Thus when the user wishes to return from a symbol mode to line mode, he needs only press button 5.

To assign button 5 to another menu command, position the pencil over the required menu command and press button 5. Button 5 is now equivalent to this menu command.

The last facility, an important factor for the operator, is the division of data into different levels. The operator can select one of sixty different levels under which to input data, by digitising the 'SET INPUT LEVEL' menu command followed by a level menu square. The operator is able to specify the levels to be active or dormant. Data under an active level will be displayed and processed by the system. In general, only filing operations will be carried out on data under dormant levels. Thus by storing data under different levels and by declaring them active or dormant, the operator is able to:

a) Reduce display and processing time.

b) Reduce editing time.

c) Obtain clear views of different categories of data (i.e. may distinguish between water pipes and electrical cables).

To make the best use of this facility, it is advised that the user sit down and think about the best way to split his particular data into levels. Much time can be saved by classifying data under different levels before starting a job.

A message in the top right corner of the screen will tell the operator which level number is the current input level. After a 'CLEAR WORKSPACE' command, the input level will be set to level 60. It is important to

remember to reset it.

An additional input mode is 'background mode'. In this mode it is a calling program that is in control. To the user it appears similar to line input mode, he is able to window, and change the position of the trailing origin by pressing button 1, finding (button 7) and driving. It is used when the calling program wants the user to define a point (e.g. the macro editor calls it to request the user to define the point to which he would like a macro to be moved).

When the user has set the trailing origin to the required position, he indicates his satisfaction by pressing button 5.

In this mode there may be instructive messages, but always the message:

B5 ENTERS POINT

will be displayed.

This mode is also useful in applications programming and is discussed further in Chapter 3.

2.4.3 Filing and display.

All data entered in one of the input modes described in section 2.4.2 is automatically buffered into a random access file called the 'workspace'.

On entry to the system the workspace is empty. Data left in the workspace at the time a run is terminated will be lost. The operator is able to clear the workspace by giving the 'CLEAR WORKSPACE' command. The menu command 'ERASE SCREEN' clears the screen of any information which has previously been stored on it and the workspace is displayed by digitising the 'DISPLAY WORKSPACE' command.

The data in the workspace may be stored under different levels. To display all levels of data the user should digitise the menu command 'DISPLAY ALL LEVELS'. To display only selected levels of data, the user should digitise the command 'DISPLAY NO LEVELS' followed by the level squares of the levels that the user wishes to be displayed.

The data contained in the workspace may be filed in any of 100 semi-permanent mass storage files. These are represented by the 100 file menu squares. There are four menu commands associated with these files: After digitising one of these menu commands, the user additionally digitises a file square to indicate which file is to be operated on. The four commands are:

1) WORKSPACE TO FILE — The file pointed to is deleted and replaced by the contents of the workspace. A message 'CONTINUE' is displayed in the top left corner of the screen after execution is complete.

2) FILE TO WORKSPACE — The contents of the file pointed to are ADDED to the contents of the workspace. The workspace is then displayed. If the file is empty, a message 'FILE EMPTY' is displayed in the top left corner of the screen.

3) DISPLAY FILE — The contents of the file pointed to are displayed in a brief form. (Symbols are not displayed).

4) DELETE FILE — The contents of the file pointed to are deleted.

The user is able to create a permanent record of data by transferring it from the workspace to any assigned device or file. The procedure is:

a) Digitise menu command 'OUTPUT TO ASSIGNED DEVICE', a message will be displayed on the screen:

B1 OUTPUT

B2 INPUT

where B1 and B2 refer to the buttons 1 and 2. After one of these is selected, a message 'A003 nnnnnn ' will be written on the keyboard. This is issued by DOS because it does not know which device is to be used in the data transfer. The user must now assign the required device or file as logical device 3. The method for doing this is described in the 'DOS MONITOR HANDBOOK' reference 3.

After the data transfer is complete, control is returned to the User Operations Monitor.

2.4.4 Macros.

Any of the 100 semi-permanent mass storage files may be manipulated and added to the workspace. This enables similar items, components or sections of data to be created in the workspace only once, filed and subsequently reproduced with very little effort. Groups of data which have been entered into the workspace in this way are called 'macros'.

As with levels, it is advisable for the user to study his problem and to decide in advance which components shall be used as macros.

The operations which can be performed on a semi-permanent mass storage file by the macro processor are:

1) Translation
2) Rotation
3) Scaling
4) Mirroring or handing

It is also possible to define the start and end point of a macro. When the macro is added to the workspace, the start point will be placed in the position defined by the trailing origin and the trailing origin will be set to the coordinate value of the end point of the macro. A set of status flags define which operations the macro processor will perform when it is called into action. These are set by the following menu commands:

1) ROTATION 0°, 90°, 180°, 270° — Sets the clockwise angle through which the macro will be rotated as either 0°, 90°, 180°, or 270°.

2) TYPE ROTATION — If the angle required is not any of 0°, 90°, 180°, 270° then the 'TYPE ROTATION' command is used and the angle is typed in on the DECwriter.

3) MIRROR — The macro will be mirrored about a line through its start point and parallel to the y-axis.

4) SELECT SCALE — When the macro is processed, the user will be asked to type in the scale required on the DECwriter.

5) SELECT START / SELECT END — The user will be asked to define the start/end of the macro at the time it is processed.

The 'SET SCALE', 'SELECT START', 'SELECT END' and 'MIRROR' menu commands act as switches. The first time the command is digitised the relevant status flag is set, the next it is unset, and so on. If the 'SELECT START' flag is not set the start point of a macro will be set to the value of the first data point in the macro. Similarly, if the 'SELECT END' point flag is not set, the end point of a macro will also be set to the value of the first data point in the macro.

The menu command 'DISPLAY ACTIVE PROCESSES' causes a display on the screen which indicates which status flags are set and therefore which operations will be performed when the macro processor is next used. The display shows a flag representing a macro. The flag appears rotated and mirrored according to the condition of the status flags. Messages show what the user will have to set at the time a macro is processed:

SELECT START

SELECT END

SET SCALE

A macro is called by digitising the menu command 'USE FILE AS MACRO' followed by the appropriate file square. If the 'SET SCALE' flag is set a message is displayed requesting the operator to type the required scale on the DECwriter. If either the 'SELECT START' or 'SELECT END' flags are

set the macro is scaled to a convenient size and displayed on the screen with the following message:

B7 SELECT START (or END) POINT

B8 CONTINUE

The user 'finds' the start or end point of the macro. If the user is unable to find the correct point, he may use button 8 to allow the processor to continue and the macro will be processed using the first data point in the macro as the start (or end) point.

When processing is complete, the macro is displayed on the screen in the orientation and position that it would be placed if it were to be directly added to the workspace. However, the user has the option of either accepting the macro as it is, or of repositioning it, or of storing the processed macro in a semi permanent mass storage file. At this stage the system appears to be in line input mode but a message is displayed:

B5 TO ENTER POINT

In this mode the user is able to use the find, drive and window facilities of line input mode but no point is entered until button 5 is pressed when the macro will be added to the workspace with its start point at the current trailing origin. If a button one is given over a file square, then the macro is filed. If a point is entered over the drawing area, the macro is added to the workspace and displayed. The system reverts to input mode.

The last macro that was processed may be called by pressing button 3 whilst in an input mode. The system will enter the state just described except that there is no option to file the macro.

The most efficient way to use macros is to ensure: that the first data point in the file to be used as a macro is the point required as

the start point; the trailing origin is at the point at which the macro is to start; that the correct rotation angle is set; that no other status flags are set. Then give the command USE FILE AS MACRO . The macro will be processed and displayed on the screen in the desired position. Press button 5 to enter the current value of the trailing origin, the macro will be added to the workspace and displayed.

If an end point is selected, the user may add a chain of macros to the workspace by pressing buttons 3 and 5 repeatedly. The macros will be added start point to endpoint, start point to end point, etc.

2.4.5 Editors.

There are three editors: point editor, line editor and the macro and symbol editor.

1) Point Editor.

The point editor is used to move points on a drawing. The editor is entered by the menu command 'POINT EDITOR'.

The operator has the following button options:

Button 1 Define point

Button 2 Define point to be joined to a line

Button 4 To window

Button 5 Application function

Button 8 To exit

Button 1 is used to define the point to be moved. The point is located by putting the cursor near it and pressing button 1. When the point has been defined the operator has the following options:-

Button 1 Locate point

Button 4 To window

Button 5 Application function

Button 6 CONTROL switch

Button 7 To FIND a point

Button 8 To exit

The new position of the moved point is defined either by digitising it using button 1 or by FINDing another point. The latter method is used to move one point to coincide with another.

If the point to be moved is defined using button 2 the operator is asked to define a line to which the point is to be moved. This he does by FINDing the two ends of the line. The point is moved to the line in such a way that the distance moved is a minimum (i.e. at right angles to the line).

2) Line Editor.

The line editor called by the menu command 'LINE EDITOR' is used to delete straight lines. On entry to the program the message:

PRESS ANY BUTTON WHEN NEAR LINE TO BE DELETED

is displayed. When this is done the program searches for a line near the point at which a button was pressed. When a line is found which is within the tolerance set in the program, it is continually displayed with the messages:

B1 EXIT
B2 ADVANCE SEARCH
B3 DELETE LINE

If the line displayed is the line required, the user deletes it by pressing button 3. He may then exit (button 1) or look for a new line. This is done either by moving the pencil so that the cursor becomes closer to the desired line or if this fails by pressing button 2.

It is important for the user to understand that having found a line the editor only searches for other lines within the buffer which contains the found line until a button 2 is pressed when it will read through the file until another line is found. The program is arranged in this way in the interests of speed for it often happens that where several lines are to be deleted, they are all contained within the same buffer. By moving the position of the pencil around these lines will very quickly be found once the one of the series has been located.

On exit from the program the next screen is erased and the workspace redisplayed.

3) Macro Editor.

The macro and symbol editor is called by the menu command 'MACRO EDITOR'. It allows rotation, translation or deletion of any macro, nest of macros or symbol.

The macro editor is called by the menu command 'MACRO EDITOR' and on entry a series of messages are displayed:

B1 DEFINE POINT
B4 WINDOW
B8 EXIT

When the program is displaying these messages, it is said to be in state 1. The operator defines a point within a macro or a real data point within a symbol (because a symbol is program generated some of the points displayed on the screen may not actually exist) by placing the pencil near the point and pressing button 1. When the point is found the program moves to what will be referred to as state 2. If the point found is within a macro a box is continually displayed around the macro, if within a symbol the symbol is displayed only once and the following messages are displayed:

B1 ADVANCE SEARCH
B2 INCREASE MACRO
B3 DELETE
B4 SELECT ROTATION
B5 REDISPLAY SYMBOL
B6 MOVE AND ROTATE
B7 MOVE
B8 RESTART

It may happen that more than one macro or symbol contain the same point, in this case the first macro or symbol encountered in the workspace is displayed. If this is not the one required, the operator may select the next one by pressing button 1.

Where a macro is made from other macros and symbols, the macro or symbol displayed may form only part of the macro that the operator wishes to edit. This is because macros can be nested in a manner similar to DO loops in Fortran and the smallest loop containing the found point is displayed.

The operator can request the contents of the next outer loop to be displayed by pressing button 2 and the next loop out again by another button 2. If no outer loop is found (i.e. there is no containing macro) the program returns to state 1. If button 3 is pressed the displayed macro or symbol is deleted and the program returns to state 1.

If the operator wishes to rotate the found macro or symbol button 4 enables him to set up the rotation angle. The following messages are displayed:

B1 TYPE ANGLE
B2 0°
B3 90°
B4 180°
B5 270°

These buttons define the rotation angle. If button 1 is pressed the operator must enter the angle via the DECwriter. After the angle has been set the program returns to state 2.

If the operator forgets which symbol was found, he may redisplay it by pressing button 5.

When a button 6 or 7 is pressed, the program calls the background mode and the user must enter the position to which the macro or symbol is to be moved. During this move, the point found within the macro is used as the locating point. If a button 6 was used to initiate the move the macro or symbol will be rotated through the angle set by the use of button 4. The box or symbol is redisplayed in the new position and the program returns to state 2.

A button 8 returns the program to state 1.

2.5 Assessment of GCADS

GCADS has proved itself to be a useful tool for the production of drawings of many types from simple flow charts and diagrams to full General Arrangements.[14] However, it has shown its greatest potential as a vehicle for applications programs.[15]

It has been used as a base for programs in: building design; structural analysis; the layout, scheduling and costing of modular store fittings; and for producing animated film sequences.

Inevitably there are criticisms of the present system:

1) The system suffers somewhat as a result of growth. It is almost impossible to plan every detail of something radically new. During the implementation of any such plan design changes are made as unforseen snags appear, or, new ideas are concieved and the system is stretched to embody them. Thus some parts of the programs have become less than elegant and a certain amount of restructuring, particularly a careful look at the common areas would tidy the software.

2) The system is too hardware dependent. In an industry in which such rapid progress is being made in harware development, software that depends on particular hardware is a very perishable commodity. In general where GCADS software interacts directly with harware the interaction is confined to a few subroutines. However the system is centred round the digitiser as the main means of data input. A large amount of the software reflects this and it would take significant effort in reprogramming to replace the digitiser/storage tube combination with a different device. It would be pleasing if the system were able to accept data from a variety of devices such as teletypes, cathode ray tubes with light pens, digitisers or storage tube terminals.

Different applications can best utilise different input/output devices.

The digitising table/storage tube combination is excellent for taking data from existing drawings and sketches. Editing the same data might be more easily achieved using a CRT graphics terminal. Thus in a time sharing system with several man/machine interaction stations the devices used in each station might range from a simple keyboard or alphanumeric display to a graphic CRT terminal or digitiser and storage tube combination.

This would allow better use of the equipment and reduce the average cost of each terminal.

The cost effectiveness of the draughting system depends very much on the type of drawing to be produced. Where the drawings contain many common features a library of components can be established. In such a case the time savings and cost savings can be very considerable.

Where the drawings have only a few similar components the use of a draughting system may only be justified if the acquired data is not only used for producing drawings but also other purposes eg. analysis, schedules, costs etc..

It is certain that the policy of maintaining a simple and open data base in order to allow the addition of applications packages was correct. It is also true that a series of system concepts have been developed which are independent of all the harware used and which may certainly be employed on future hardware.

GCADS not only adequately fulfills the role for which it was designed but the role itself is extremely important for the development of applications programs in Computer Aided Design.

CHAPTER THREE

Using GCADS for Application Programs

When GCADS was written, specific sockets were left free for use by applications programs. The facilities available to the application programmer are:-

- Overlaying system and overlay calling by menu command.
- Buffer and parameter space in the resident common area.
- Ability to obtain data from input devices and to file and display data by 'stacking' GCADS utility overlays andby calling utility subroutines.

1) Overlaying system

a) Using the overlay builder.

The overlay builder OVAL can handle up to 128 overlays. Each overlay may be up to ten kilowords in length and is referred to by number within the programs. GCADS occupies most of the lowest 30 overlay numbers. The overlays are built by OVAL into a random access file CADMAC.OVL on UIC 2,2 on the moving head disk unit. In the following dialogue messages printed by the computer are underlined.

The overlay builder is run by typing the command:-

RU OVAL

the computer responds with:-

OVERLAY BUILDER V001
*

the user may then build overlays into the random access file using a command of the form:-

* OVERLAYN/OV:N,OVERLAYM/OV:M,.....

where OVERLAYN and OVERLAYM are the load modules for overlay numbers N and M.

The user may obtain a listing of the current overlays and overlay numbers contained in CADMAC.OVL by the command:

*DV:/LI

where DV: is the device on which the listing is to be produced. To zero CADMAC.OVL the

*/ZE

command is used.

Overlays are program called from CADMAC.OVL by subroutine calls to subroutine OVLINK and STACK as described in section 2.3.2.

b) Adding a user menu command.

The call for a user overlay may either be initiated from within another user overlay or by a menu command.

Clearly a users program must use at least one menu command to call the first user overlay. Menu command squares one through forty are reserved for user commands. The way in which menu command squares are mapped is described in section 2.3.3. The steps that a user programmer must perform to link an overlay to a menu command square are as follows:

1) Link the overlay main program with its subroutines to a bottom limit of 30776.

2) Select an overlay number OVNUM.

3) Select a free user command menu square and note its position in the menu MNUM.

4) In the byte map in subroutine MENSEL select an unused location and edit in the value of (OVNUM - 1). Note the position of the chosen location MAPNUM within the byte map.

5) The byte map in subroutine MENMAP represents the first 80 menu command squares. In the location MNUM in the map edit in the value MAPNUM.

6) Link GCADS overlay DIGOV which contains the menu handler with the

new versions of MENSEL and MENMAP and the GCADS library of subroutines to a bottom limit of 30776.

7) Run the overlay builder OVAL and build DIGOV as overlay 4 and the user overlay as overlay OVNUM.

c) Adding a symbol

It is convenient at this stage to describe the linking of symbols to menu symbol squares. It will be recalled that a symbol is a graphic item whose display is program generated. There are two segments of program concerned with each symbol: the first is to collect and store the data points which define the dimensions and location of the symbol - the symbol entry segment; the second to generate and display the symbol upon request - the symbol display segment. These segments of program need not be within the same overlay, in fact the speed of display is greatly improved if commonly used symbol display segments are added to the GCADS display overlay DISALL - overlay number 6.

The process of linking a symbol entry segment to a symbol menu square is very similar to that of linking a user menu command except that subroutines MENSEL and MENMAP are replaced by subroutines SYMGO and SYMAP.

1) Link the overlay main program containing the symbol entry segment with its subroutines to a bottom limit of 30776.

2) Let the overlay number be OVNUM and the segment number be SEGNUM.

3) Select a free symbol menu square SQNUM

4) In the word map in SYMGO select an unused location N and edit in the value of (OVNUM - 1)*25. In the Nth + 1 location in the byte map in SYMGO edit in the value of SEGNUM.

5) In the byte map in SYMAP in location SQNUM edit in the value of N.

6) Relink DIGOV with the new versions of SYMGO and SYMAP as before.

7) Run the overlay builder to build DIGOV as overlay 4 and the overlay

containing the symbol entry segment as overlay number OVNUM.

A symbol display segment is called by a call to subroutine DISSYM(N), which contains word and byte maps similar to those in SYMGO. These must be updated as follows :

1) OVNUM is the overlay number containing the symbol display segment in segment number SEGNUM.

2) In the Nth position in the word map in DISSYM edit in the value (OVNUM-1)*25 and in the Nth + 1 position in the byte map edit in the value of SEGNUM.

3) Relink and build all overlays that call DISSYM. This will certainly include the display overlay DISALL and the macro editor MACRO4.

2) Buffer and parameter space in the resident common area.

There are three common areas reserved for user overlays:

```
COMMON/GENRL/ SYS(2),USERP(8)
COMMON/MESAGE/ SYSTEM(27),USERM(5,5)
COMMON/USER/ USERB1(128),USERB2(128)
```

where

USERP is to allow the user to store parameters particular to his problem.

USERM is displayed as five lines of messages by the system when it is in an input mode. Four characters may be stored in each variable, the last character in a message must be a zero. Therefore 5 lines of 19 characters are available. These are used to let user overlays receive data through the system data input facilities and yet allow the user program to send messages or instructions to the operator during data input.

USERB1 and USERB2 are intended to be used as buffer space and each corresponds to one physical disk record in length. USERB1 and USERB2 can be saved and restored to and from the disk unit by calling STORCM(N) and

RESTCM(N) where N is the disk record number in random access file CADMAC.RAO at which the data transfer will commence.

In practice it has been found that there is a lack of parameter space free for the user. In this case the user programmer may choose to use some of the locations in USERM for storing parameters rather than messages.

3) Obtaining data points by 'stacking' 'background mode'.

It frequently happens that an application program requires coordinate data from the digitising table. The application programmer can obtain the data and allow the operator use of the draughting and construction aids usually available in an input mode by stacking BACKOV the background overlay.

BACKOV stores the messages:

B5 ENTERS POINT
B3 EXIT

in common area MESAGE. and calls the background loop. The operator is able to find, drive window etc. but the position of the trailing origin will not be returned to the application overlay until the operator presses button 5.

If button 3 is pressed BACKOV sets MNUM in common area MENU to -1 and returns control to the application overlay.

The current position of the trailing origin i.e. the last point entered by the operator is returned in XT and YT in common area PARAM.

The advantages gained from using BACKOV are:

- The operator is always presented with a similar set of draughting aids and button functions.
- The application programmer is saved from a lot of programming.
- Unnecessary duplication of programs is avoided and the application programmer is able to pack much more useful work into each overlay.

CHAPTER FOUR

STASYS - STructural Analysis SYStem

4.1 Introduction

Methods of structural analysis can be divided into two groups (see figure 4.1) analytical methods and numerical methods. The limitations imposed by analytical methods is well known; Only in the simplest of cases are closed-form solutions feasible. Approximate solutions can be found for some simple structures, but in general for complex structures analytical methods cannot be used with any degree of accuracy and one has to resort to numerical methods. The numerical methods of structural analysis can be divided into two groups:

1) Numerical solutions of differential equations for displacements or stresses.

2) Matrix methods based on discrete-element idealisation.

In the first type the equations of elasticity are solved for a particular structural configuration either by finite difference or by direct numerical integration. In this approach the analysis is based on a mathematical approximation of differential equations. Practical limitations restrict the application of these methods to simple structures. Also solutions cannot be found for general structural configurations.

In the second method the structure is first idealised into an assembly of discrete structural elements with assumed form of displacement or stress distributions, and the complete solution is then obtained by combining these individual approximate stress or displacement distributions in a manner which satisfies the force-equilibrium and displacement compatibility at the junctions of these elements. Methods based on this approach are suitable

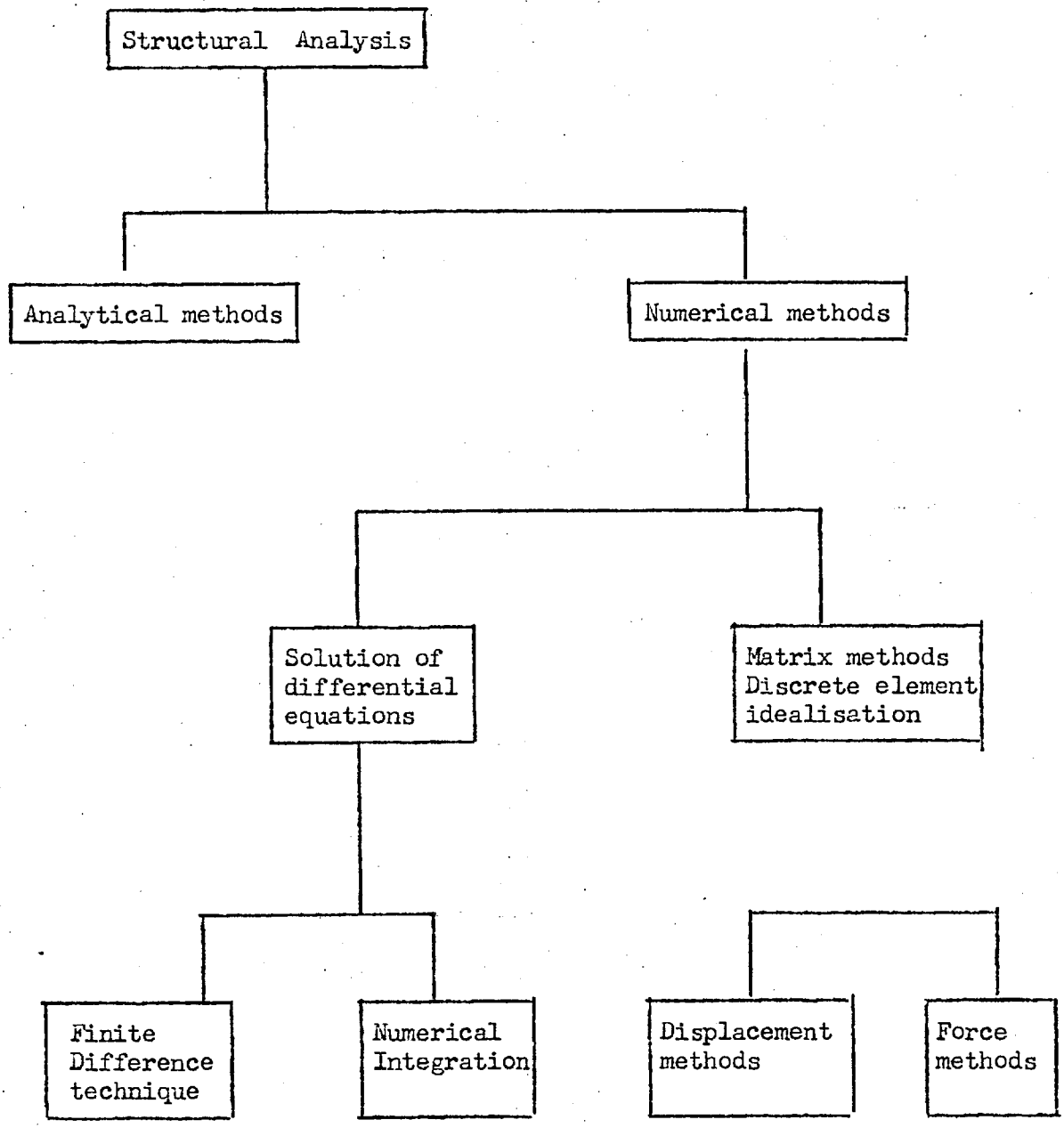


Figure 4.1 Methods of structural analysis.

for the analysis of complex structures. These methods involve an appreciable quantity of linear algebra and the use of matrix algebra is convenient. The formulation of the analysis in matrix algebra is ideally suited for the solution on the digital computer.

The finite element method of analysis has been widely adopted in many fields of engineering. The object under analysis is represented by an assemblage of components or elements interconnected at a finite number of points. It is the finite character of the structural connectivity which makes possible solution by simultaneous equations. The approximation involved in the use of the method is essentially physical. There need be no mathematical approximation in the solution of the substitute system.

It is reasonable to suppose that the larger the number of elements used to represent the original structure, the more accurate the representation and hence the analysis will be. This has been the experience in practice and most structures have to be represented by a considerable number of elements.

Data defining the position of each nodal point, the material properties of each element, the element interconnections and loading and constraint conditions must be generated. It is the effort required in generating and validating this data as well as the computing cost which must be balanced against the probable gains obtained by carrying out the analysis.

In the aero industry minimum weight design is absolutely crucial and yet rigorous safety standards have to be maintained. It is also an industry where the same design may be used for a considerable number of production units. The cost of analysis is therefore spread over a considerable number of units. For these reasons a great deal of the developement of the finite element method was carried out in the aircraft and aerospace industry.

However the comprehensive finite element analysis packages such as NASTRAN that have emerged are extremely large. They can only be run on large computers and are expensive to uge.

In the building industry for aesthetic reasons and because site conditions may demand it, a particular building design may only be used once. Optimum weight design is only important in so far as it may save in material costs. Thus it is relatively important to keep the total cost of the building design process as low as possible. Design cost may be as much as one tenth construction cost. With the present high cost of finite element analysis it is not generally considered to be justified unless particular factors are involved:

- The design is radically new and its behaviour unknown.
- Exceptional and varied loading conditions are anticipated.
- The design is to be used for a number of buildings so the cost of analysis will be spread and the benefits multiplied.

There also exists the problem that codes of practice may not allow full exploitation of the knowledge gained from the analysis.

Computing costs are falling and have fallen over the last years, to the extent that data preparation may cost more than the actual computer analysis. It is in the area of data preparation that this project is aimed.

The desire is not only to greatly reduce the data preparation cost for finite element analysis of buildings but also to determine the extent to which the analysis might be carried out on minicomputers that a Civil Engineering Consultancy would be able to purchase.

If such a system could be produced the tool of finite element analysis would be placed in the hands of every engineer for use on many of the problems that were not previously analysed accurately because to do so would have been too expensive and time consuming.

The system was initially to contain only line and rectangular plate elements because much modern building can be idealised by the use of

these alone. A further constraint that the elements should lie in a vertical or horizontal plane was also added to facilitate data input.

The system makes use of, and relies on, the fact that buildings can be considered as a series of layers and that data will frequently be available on general arrangement drawings at different levels in the buildig. building.

There are four stages in the analysis process:

1) Data input
2) Data collation and data analysis
3) Structural analysis
4) Output of results

The user creates graphic data files of each floor of the building describing the structural elements and loading and boundary conditions.

When the user is satisfied that the graphic representation is accurate the data analysis programs are initiated. If these issue no error or diagnostic messages then the user proceeds to solution and subsequently to the output of results.

It is important to realise that because there are data collation and analysis programs between the graphic data which is input by the user and the data that the structural analysis programs operate on, the user has a great deal of freedom. He may not only use all the utilities and facilities provided by GCADS but may also set up the graphic model of the structure in any order.

4.2 Finite element theory

An elastic structure or continuum may be represented by many discrete components or elements interconnected at a finite number of nodal points situated on the element boundaries.[6,7] The displacements of these nodal points will be the basic unknown parameters of the problem.

A set of functions is chosen to define uniquely the state of displacement within each finite element in terms of its nodal displacements. Thus the displacement functions defune uniquely the state of strain within an element in terms of the nodal displacements. These strains, together with any initial strains and the constitutive properties of the material define the state of stress throughout the element.

A system of forces concentrated at the nodes and equilibrating the boundary stresses and any distributed loads is determined, resulting in a stiffness relationship of the form:

$$\{F\}^A = [K]^A \{U\}^A + \{F\}^A_p + \{F\}^A_{\varepsilon 0}$$

where: $\{F\}^A$ represents the nodal forces on element A,

$[K]^A$ is the element stiffness matrix,

$\{U\}^A$ represents the nodal displacements of element A

$\{F\}^A_p$ represents the nodal forces required to balance any distributed loads acting on the element

and $\{F\}^A_{\varepsilon 0}$ represents the nodal forces required to balance any initial strains such as may be caused by temperature change, if the nodes are not subject to any displacement.

It is not always easy to ensure that the chosen displacement functions will satisfy the requirement of displacement continuity between adjacent elements. Thus, the compatibility condition on such lines may be violated. By concentrating equvalent forces at the nodes, equilibrium conditions are

satisfied in the overall sense only. Local violation of equilibrium conditions within each element will usually occur.

The choice of element shape and of the form of the displacement functions will determine the accuracy of the finite element model.

It is also possible to define the stresses or internal reactions at any specified point or points of the element in terms of the nodal displacements:

$$\{\sigma\}^A = [S]^A\{U\}^A + \{\sigma\}_p^A + \{\sigma\}_{\varepsilon 0}^A$$

where: $[S]^A$ is the element stress matrix

and $\{\sigma\}^A$ represents the nodal stresses for element A.

The last two terms are the stresses due to the distributed element loads and initial stresses when no nodal displacement occurs.

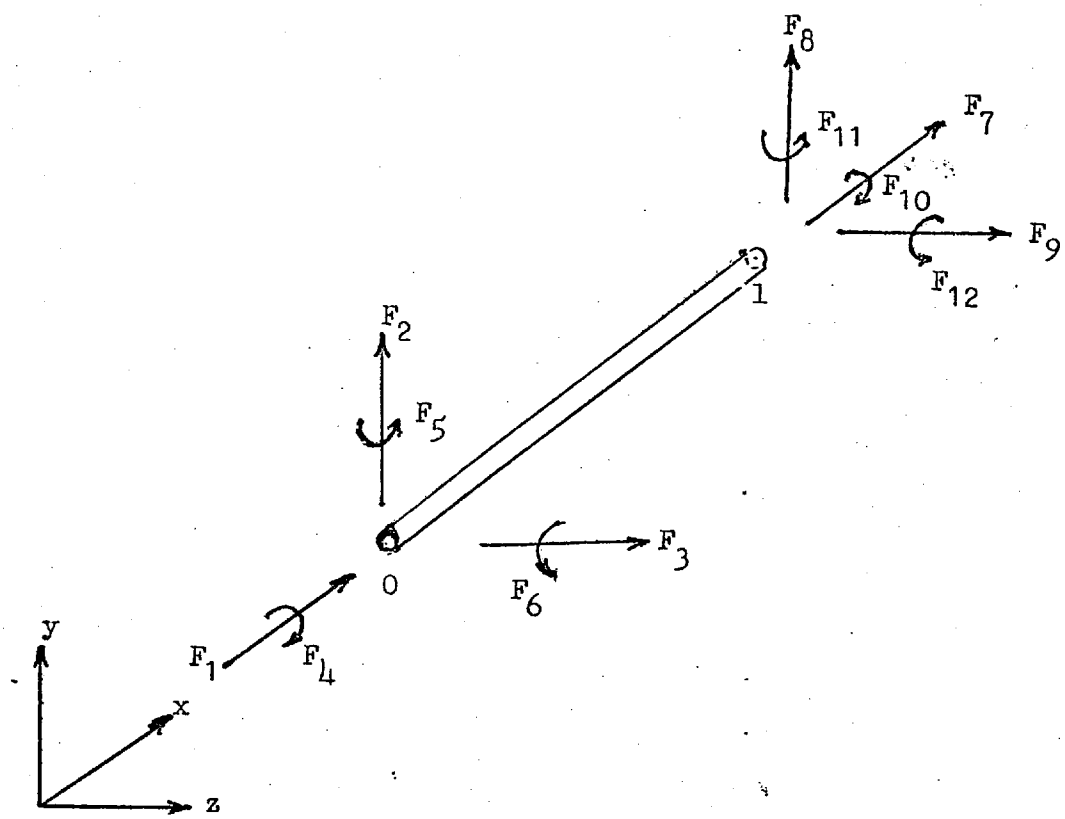
In STASYS it is assumed that there will be no initial strains and temperature effects are not included. Line and rectangular elements are used although it is hoped that other elements will be added to the system later. In any case each nodal point in the structure is allowed six degrees of freedom.

The stiffness matrix for the line element can be derived directly from the differential equations for beam displacements in engineering beam theory.

4.2.1 Derivation of stiffness matrix for line element.

The derivation is for a slender straight bar of uniform cross section capable of resisting axial forces, bending and twisting moments.

The forces acting on the element are F_1 to F_{12} and the corresponding displacements U_1 to U_{12}.



1) Axial forces F_1 and F_7

$F_1 = -(\frac{du}{dx})\ EA$ governing differential equation.

hence $F_1 x = -UEA + C_1$ and assuming that the left hand end of the element has displacement U_1 while $U_7 = 0$ at $x = 1$ then: $C_1 = F_1 1$ and:

$F_1 = \frac{EA}{1} U_1$ also $F_1 = -F_7$ from equilibrium.

2) Twisting moments F_4 and F_{10}

Differential equation for twist 0 on the element:

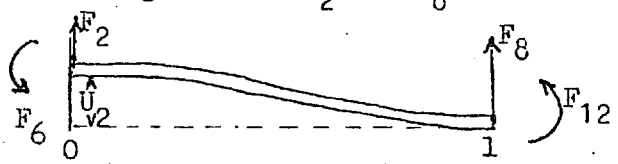
$$F_4 = -GJ \frac{d\theta}{dx}$$

hence $F_4 x = -GJ\theta + C_1$ Let $\theta = 0$ at $x = 1$

then $C_1 = F_4 l$ Also $\theta = U_4$ at $x = 0$

so $F_4 = \frac{GJ}{l} U_4$ and $F_{10} = -F_4$ from equilibrium.

3) Shearing forces F_2 and F_8



The lateral deflection w on the beam subjected to shearing forces and associated moments is assumed to be due only to bending strains.

$$EI = \frac{d^2w}{dx^2} = F_2 x - F_6 \qquad (3.1)$$

$$EI_z w = \frac{F_2 x^3}{6} - \frac{F_6 x^2}{2} + C_1 x + C_2 \qquad (3.2)$$

$\frac{dw}{dx} = 0$ at $x = 0$ and $x = 1$, and $w = 0$ at $x = 1$

hence $$EI_z w = \frac{F_2 x^3}{6} - \frac{F_6 x^2}{2} + \frac{l^3 F_2}{12}$$

where $F_6 = \frac{F_2 l}{2}$ also from equilibrium:

$$F_8 = -F_2 \quad \text{and} \quad F_{12} = -F_6 + F_2 l$$

Now at $x = 0$, $w = U_2$ and from 3.2 $U_2 = \frac{l^3 F_2}{12EI_z}$

therefore $F_2 = (12EI_z/l^3)\, U_2$ ..... $k_{2,2}$

$F_6 = (6EI_z/l^2)\, U_2$ ..... $k_{6,2}$

$F_3 = (12EI_z/l^3)\, U_2$ ..... $k_{8,2}$

$F_{12} = (6EI_z/l^2)\, U_2$ ..... $k_{12,2}$

Similarly if the other end of the beam is displaced:

$k_{8,8} = k_{2,2}$

and $k_{12,8} = k_{6,2}$

4) Bending moments F_6 and F_{12}

To obtain the stiffness coefficients associated with the rotations U_6 and U_{12} the beam is subjected to bending moments and shear forces.

Deflections can be obtained from equation 3.2. The constants C_1 and C_2 must be revaluated for the new boundary conditions:

$U_6 = 0$ at $x = 0$, $x = 1$ and $\frac{du}{dx} = 0$ at $x = 1$

Then the equation becomes: $EI_z U_6 = \frac{F_2}{6}(x^3 - l^3 x) + \frac{F_6}{2}(lx - x^2)$

and $F_2 = \frac{6F_6}{4l}$

The remaining forces can be determined from equilibrium and together with the boundary conditions:

$\frac{du}{dx} = U_6$ at $x = 0$

to give: $F_6 = 4 \frac{EI_z}{l} U_6$ ..... $k_{6,6}$

$F_3 = (6EI_z/l^2)\, U_6$ ..... $k_{8,6}$

$$F_{12} = (2EI_z/l)\, U_6 \qquad \cdots\cdots\cdots\ k_{12,6}$$

and by symmetry $k_{12,12} = k_{6,6}$

5) Shearing forces F_3 and F_9

These are exactly similar to the shear forces F_2 and F_8 except that the bending moments F_5 and F_{11} associated with F_3 and F_9 are in an opposite sense to F_6 and F_{12} associated with F_2 and F_8.

Hence:

$$k_{3,3} = -k_{2,2}$$

$$k_{5,3} = -k_{6,2}$$

$$k_{9,3} = -k_{8,2}$$

$$k_{11,3} = k_{12,2}$$

$$k_{9,9} = k_{8,8}$$

$$k_{11,9} = k_{12,8}$$

Obviously it is I_y that is used in these expressions not I_z.

6) Bending moments F_5 and F_{11}

The same applies as for the shear forces and:

$$k_{5,5} = k_{6,6}$$

$$k_{9,5} = k_{8,6}$$

$$k_{11,5} = k_{12,6}.$$

If all the stiffness coefficients are collected and put into matrix form then the complete stiffness matrix for the line element is formed as shown in figure 4.2.

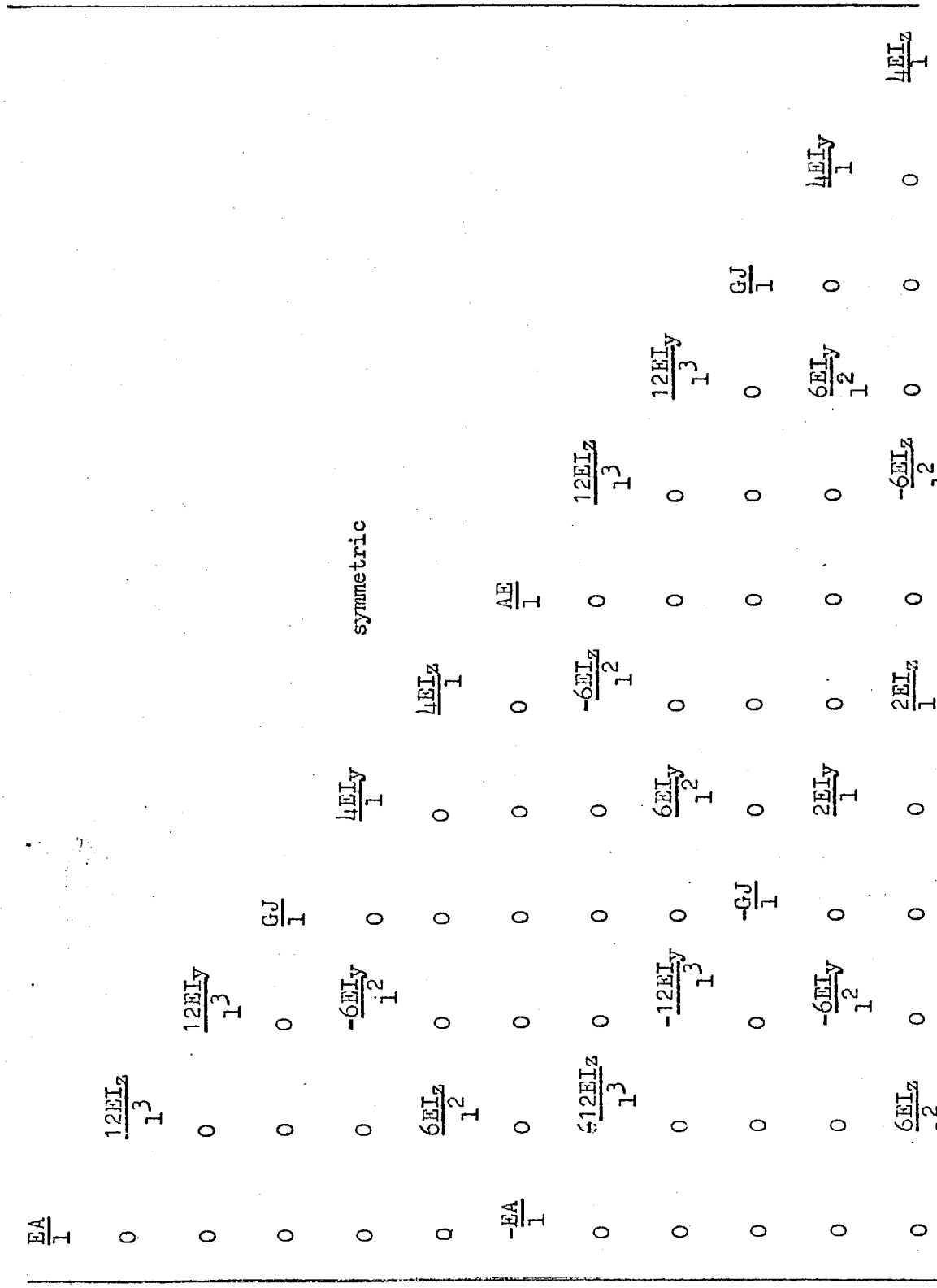
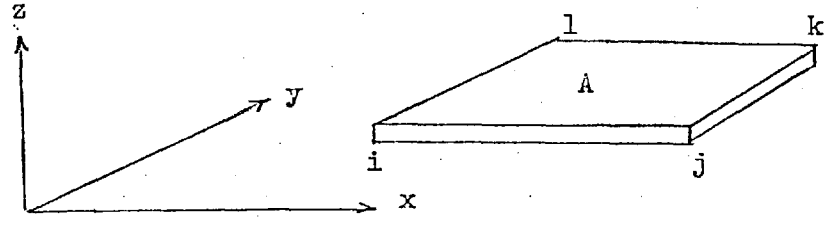


Figure 4.2 Line element stiffness matrix

4.2.2 Formulation of rectangular plate element stiffness matrix.

In the small deflection theory of thin plates the transverse (normal) deflections 'w' are uncoupled from the in-plane deflections u and v. Consequently the stiffness matrices for the in-plane and transverse deflections are also uncoupled and they can be calculated independently.

Consider a typical element A.



In order to derive the stiffness matrix for the transverse deflection w three degrees of freedom have to be considered at each point. Two rotations about perpendicular axes x and y within the plane of the plate and the lateral deflection w. We denote these displacements:

$$u_i = \begin{vmatrix} \theta_{xi} \\ \theta_{yi} \\ w_i \end{vmatrix} = \begin{vmatrix} \frac{\partial w}{\partial y_i} \\ \frac{\partial w}{\partial x_i} \\ w_i \end{vmatrix}$$

We refer to the displacements of the element as:

$$\{U\}^A = \begin{vmatrix} u_i \\ u_j \\ u_k \\ u_l \end{vmatrix}$$

Similarly at every nodal point forces F_i may be assumed to exist. Each consists of three components:

$$F_i = \begin{vmatrix} M_{xi} \\ M_{yi} \\ F_{zi} \end{vmatrix} \qquad \{F\}^A = \begin{vmatrix} F_i \\ F_j \\ F_k \\ F_l \end{vmatrix}$$

The primary interest is usually in the internally distributed moments of the elements. The solution of the problem hinges on determining the element stiffness matrix $[K]^A$ and the matrix relating the internal moments to the element displacements $[S]^A$.

The lateral deflection w may be represented by a polynomial in x and y . Since three degrees of freedom exist at each of the four nodes, twelve undetermined constants may be used.

A suitable expression is:

$$w = A_1 + A_2x + A_3y + A_4x^2 + A_5xy + A_6y^2 + A_7x^3 + A_8x^2y + A_9xy^2 + A_{10}y^3 + A_{11}x^3y + A_{12}xy^3$$

from which we obtain:

$$\theta_x = \frac{w}{y} = A_3 + A_5x + 2A_6y + A_8x^2 + 2A_9xy + 3A_{10}y^2 + 3A_{11}x^3 + 3A_{12}xy^2$$

$$\theta_y = \frac{w}{x} = A_2 + 2A_4x + A_5y + 3A_7x^2 + 2A_8xy + A_9y^2 + 3A_{11}x^2y + A_{12}y^3$$

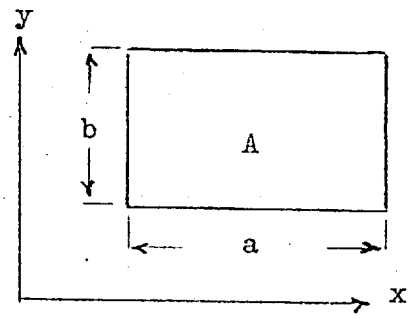
Substituting as follows:

$x_i = 0 \qquad y_i = 0$

$x_j = a \qquad y_j = 0$

$x_k = 0 \qquad y_k = b$

$x_l = a \qquad y_l = b$



we have: $\{U\}^A = [C]\{A\}$

The curvature and twist at any point in the plate can now be determined in terms of the twelve constants.

$$\chi = \begin{vmatrix} -\frac{\partial^2 w}{\partial x^2} \\ -\frac{\partial^2 w}{\partial y^2} \\ 2\frac{\partial^2 w}{\partial x\, y} \end{vmatrix} = [B]\{A\} = [B][C]^{-1}\{U\}^A$$

From the theory of plates[8].

$$M_x = -D\left(\frac{\partial^2 w}{x^2} + \frac{\partial^2 w}{y^2}\right)$$

$$M_y = -D\left(\frac{\partial^2 w}{\partial y^2} + \frac{\partial^2 w}{\partial x^2}\right)$$

$$M_{xy} = \tfrac{1}{2}(1 - \nu) D \frac{\partial^2 w}{\partial x \partial y}$$

where $D = \dfrac{Eh^3}{12(1 - \nu^2)}$

hence we have:

$$\{M\} = [D]\chi$$

$$[D] = \frac{Eh^3}{12(1 - \nu^2)} \begin{vmatrix} 1 & \nu & 0 \\ \nu & 1 & 0 \\ 0 & 0 & \frac{1}{2}(1-\nu) \end{vmatrix}$$

and

$$-\frac{\partial^2 w}{\partial y^2} = -(2A_4 + 6A_7 x + 2A_8 y + 6A_{11} xy)$$

$$-\frac{\partial^2 w}{\partial y^2} = -(2A_6 + 2A_9 x + 6A_{10} y + 6A_{12} xy)$$

$$\frac{\partial^2 w}{\partial x \partial y} = A_5 + 2A_8 x + 2A_9 y + 3A_{11} x^2 + 3A_{12} y^2$$

[B] is shown in figure 4.3.

$$
[B] = \begin{vmatrix}
0 & 0 & 0 & -2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & -2 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & -2 & 0 & 0 & 6a & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & -2 & 0 & 0 & 2a & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 2a & 0 & 0 & 3a^2 & 0 \\
0 & 0 & 0 & -2 & 0 & 0 & 0 & 2b & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & -2 & 0 & 0 & 0 & 6b & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 2b & 0 & 0 & 3b^2 \\
0 & 0 & 0 & -2 & 0 & 6a & 0 & 2b & 0 & 0 & 6ab & 0 \\
0 & 0 & 0 & 0 & 0 & -2 & 0 & 0 & 2a & 6b & 0 & 6ab \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 2a & 2b & 0 & 3a^2 & 3b^2
\end{vmatrix}
$$

Figure 4.3

During any displacement the external work done must be equal to the internal work.

If a displacement is such that it is unity in the direction of a selected external force and zero in the direction of all other forces, the internal work will be the same as the value of this selected force.

Writing δu^A as equal to I the identity matrix then the external work may be represented in matrix form as:

$$W_{ext} = (\delta u^A)^T \{F\}^A = I \{F\}^A = \{F\}^A$$

To each of these displacements corresponds an equal internal work done by the moments:

$$W_{int} = \iint (\delta \chi)^T \{M\} dxdy$$

where

$$\delta \chi = [B][C]^{-1} (\delta u^A) = [B][C]^{-1}$$

Substituting for $\delta \chi$ and M and equating internal and external work results in:

$$\{F\}^A = \iint ([B][C]^{-1})^T [D][B][C]^{-1} \{U\}^A dxdy$$

$$= \left[([C]^{-1})^T \left\{ \iint [B]^T [D] [B] \, dxdy \right\} [C]^{-1} \right] \{U\}^A = [K]^A \{U\}^A$$

from

$$\chi = [B][C]^{-1} \{U\}^A$$

and

$$M = ([D][B][C]^{-1}) \{U\}^A$$

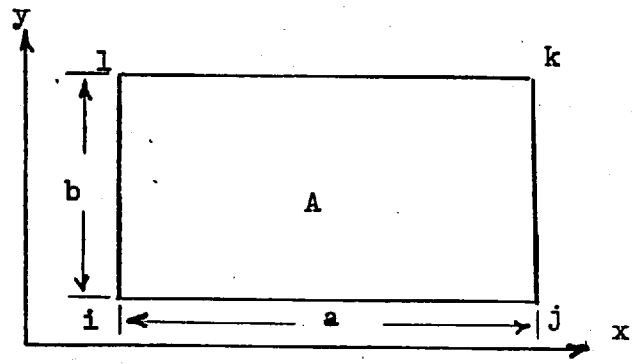
we have

$$M = [S]^A \{U\}^A$$

The matrix $[F]^A$ is given in figure 4.6 together with the terms which make up the in-plane stiffnesses.

Rectangular plate element in-plane forces

Consider a typical element A.



Two degrees of freedom have to be considered at each point. These are in-plane deflections along the axes x and y and are denoted as:

$$u_i = \begin{vmatrix} u_x \\ u_y \end{vmatrix}$$

We refer to the displacements of the element as:

$$U^A = \begin{vmatrix} u_i \\ u_j \\ u_k \\ u_l \end{vmatrix} = \begin{vmatrix} u_1 \\ \vdots \\ u_8 \end{vmatrix}$$

Similarly at every nodal point forces F_i may be assumed to exist. Each consists of two components:

$$F_i = \begin{vmatrix} F_x \\ F_y \end{vmatrix} \qquad \text{and} \qquad \{F\}^A = \begin{vmatrix} F_i \\ F_j \\ F_k \\ F_l \end{vmatrix}$$

We assume that the edge displacement function is linear and a suitable function is:

$$u_x = C_1 p + C_2 pq + C_3 q + C_4$$

$$u_y = C_5 p + C_6 pq + C_7 q + C_8$$

where

$$p = x/a \qquad \text{and} \qquad q = y/b$$

As in the case of plate bending the eight arbitrary constants can be determined from the known displacements in the x and y directions at the four corners of the rectangle. Hence we can obtain:

$$u_x = (1-p)(1-q)u_1 + (1-p)qu_3 + pqu_5 + p(1-q)u_7$$

$$u_y = (1-p)(1-q)u_2 + (1-p)qu_4 + pqu_6 + p(1-q)u_8$$

Examination of these two equations shows that the distribution of the u_x and u_y displacements along any edge is linear and that it depends only on the element displacements of the two corner points defining the particular edge. Thus the assumed form of displacement distribution ensures that the compatibility of displacements on the boundaries of adjacent elements are satisfied.

Noting that:

$$e_{xx} = \frac{\partial u_x}{\partial x} = \frac{1}{a}\frac{\partial u_x}{\partial p}$$

$$e_{yy} = \frac{\partial u_y}{\partial y} = \frac{1}{b}\frac{\partial u_y}{\partial q}$$

$$e_{xy} = \frac{\partial u_x}{\partial y} + \frac{\partial u_y}{\partial x} = \frac{1}{b}\frac{\partial u_x}{\partial q} + \frac{1}{a}\frac{\partial u_y}{\partial p}$$

We find that the total strain-displacement relationship for the element becomes:

$$\begin{vmatrix} e_{xx} \\ e_{yy} \\ e_{xy} \end{vmatrix} = \begin{vmatrix} -\frac{(1-q)}{a} & 0 & -\frac{q}{a} & 0 & \frac{q}{a} & 0 & \frac{1-q}{a} & 0 \\ 0 & -\frac{(1-p)}{b} & 0 & \frac{1-p}{b} & 0 & \frac{p}{b} & 0 & -\frac{p}{b} \\ -\frac{(1-p)}{b} & -\frac{(1-q)}{a} & \frac{1-p}{b} & -\frac{q}{a} & \frac{p}{b} & \frac{q}{a} & -\frac{p}{b} & \frac{1-q}{a} \end{vmatrix} \begin{vmatrix} u_1 \\ \vdots \\ u_8 \end{vmatrix}$$

$$\begin{vmatrix} \sigma_{xx} \\ \sigma_{yy} \\ \sigma_{xy} \end{vmatrix} = \frac{E}{(1-\nu^2)} \begin{vmatrix} -\frac{(1-q)}{a} & -\frac{\nu(1-p)}{b} & -\frac{q}{a} & \frac{\nu(1-p)}{b} & \frac{q}{a} & \frac{\nu p}{b} & \frac{1-q}{a} & -\frac{p}{b} \\ -\nu\frac{(1-q)}{a} & -\frac{(1-p)}{b} & -\nu\frac{q}{a} & \frac{1-p}{b} & \nu\frac{q}{a} & \frac{p}{b} & \nu\frac{(1-q)}{a} & -\frac{p}{b} \\ -\frac{(1-\nu)(1-p)}{2b} & -\frac{(1-\nu)(1-q)}{2a} & \frac{(1-\nu)(1-p)}{2b} & -\frac{(1-\nu)q}{2a} & \frac{(1-\nu)p}{2b} & \frac{(1-\nu)q}{2a} & -\frac{(1-\nu)p}{2b} & \frac{(1-\nu)(1-q)}{2a} \end{vmatrix} \begin{vmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \\ u_7 \\ u_8 \end{vmatrix}$$

Stress matrix S^A for rectangular plate element under in-plane forces

Figure 4.4

$$
\begin{vmatrix}
4B + 2(1-\nu)B^{-1} & & & & & & & & & & & \\
\frac{3}{2}(1+\nu) & 4B^{-1} + 2(1-\nu)B & & & & & & & & & & \\
2B - 2(1-\nu)B^{-1} & -\frac{3}{2}(1-3\nu) & 4B + 2(1-\nu)B^{-1} & & & & & & & & & \\
\frac{3}{2}(1-3\nu) & -4B^{-1} + (1-\nu)B & -\frac{3}{2}(1+\nu) & 4B^{-1} + 2(1-\nu)B & & & & & & & & \\
-2B - (1-\nu)B^{-1} & -\frac{3}{2}(1+\nu) & -4B + (1-\nu)B^{-1} & -\frac{3}{2}(1-3\nu) & 4B + 2(1-\nu)B^{-1} & & & & & & & \\
-\frac{3}{2}(1+\nu) & -2B^{-1} - (1-\nu)B & \frac{3}{2}(1-3\nu) & 2B^{-1} - 2(1-\nu)B & \frac{3}{2}(1+\nu) & 4B^{-1} + 2(1-\nu)B & & & & & & \\
-4B + (1-\nu)B^{-1} & \frac{3}{2}(1-3\nu) & -2B - (1-\nu)B^{-1} & \frac{3}{2}(1+\nu) & 2B - 2(1-\nu)B^{-1} & -\frac{3}{2}(1-3\nu) & 4B + 2(1-\nu)B^{-1} & & & & & \\
-\frac{3}{2}(1-3\nu) & 2B^{-1} - 2(1-\nu)B & \frac{3}{2}(1+\nu) & -2B^{-1} - (1-\nu)B & \frac{3}{2}(1-3\nu) & -4B^{-1} + (1-\nu)B & -\frac{3}{2}(1+\nu) & 4B^{-1} + 2(1-\nu)B & & & &
\end{vmatrix}
$$

S Y M M E T R I C

Figure 4.5 Stiffness matrix for rectangular plate element - in-plane stiffnesses.

The following three pages give the complete stiffness matrix for the rectangular plate element. The matrix is partitioned:

$$\left|\begin{array}{c|c} K_{1,1} & K_{2,1}^{T} \\ \hline K_{2,1} & K_{2,2} \end{array}\right|$$

The order of displacements is:

1 - u_{xi}	7 - u_{xj}	13 - u_{xk}	19 - u_{xl}	in-plane
2 - u_{yi}	8 - u_{yj}	14 - u_{yk}	20 - u_{yl}	
3 - u_{zi}	9 - u_{zj}	15 - u_{zk}	21 - u_{zl}	bending
4 - θ_{xi}	10 - θ_{xj}	16 - θ_{xk}	22 - θ_{xl}	
5 - θ_{yi}	11 - θ_{yj}	17 - θ_{yk}	23 - θ_{yl}	
6 - θ_{zi}	12 - θ_{zj}	18 - θ_{zk}	24 - θ_{zl}	

Multiplier for in-plane terms:

$$\frac{E.t}{12(1-\nu^2)}$$

Multiplier for bending terms:

$$\frac{E.t^3}{12(1-\nu^2)a.b}$$

where t is the thickness of the element.

Figure 4.6

Stiffness matrix for rectangular plate element.

$K_{1,1}$

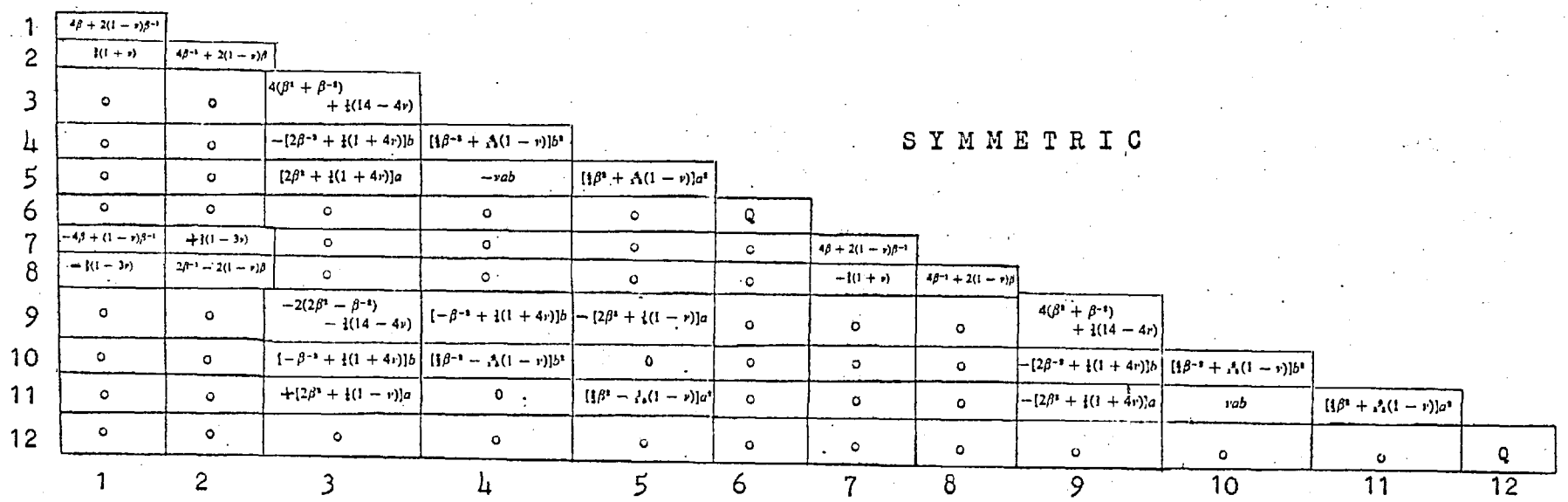


Figure 4.6a

$$K_{2,1} = K_{1,2}$$

	1	2	3	4	5	6	7	8	9	10	11	12
13	$-2\beta-(1-\nu)\beta^{-1}$	$-\frac{1}{2}(1+\nu)$	0	0	0	0	$2\beta-2(1-\nu)\beta^{-1}$	$+\frac{1}{2}(1-3\nu)$	0	0	0	0
14	$-\frac{1}{2}(1+\nu)$	$-2\beta^{-1}-(1-\nu)\beta$	0	0	0	0	$-\frac{1}{2}(1-3\nu)$	$-4\beta^{-1}+(1-\nu)\beta$	0	0	0	0
15	0	0	$-2(\beta^3+\beta^{-3})$ $+\frac{1}{2}(14-4\nu)$	$[-\beta^{-3}+\frac{1}{2}(1-\nu)]b$	$[\beta^3-\frac{1}{2}(1-\nu)]a$	0	0	0	$2(\beta^3-2\beta^{-3})$ $-\frac{1}{2}(14-4\nu)$	$-[2\beta^{-3}+\frac{1}{2}(1-\nu)]b$	$[\beta^3-\frac{1}{2}(1+4\nu)]a$	0
16	0	0	$[\beta^{-3}-\frac{1}{2}(1-\nu)]b$	$[\frac{1}{2}\beta^{-3}+\frac{1}{16}(1-\nu)]b^3$	0	0	0	0	$[2\beta^{-3}+\frac{1}{2}(1-\nu)]b$	$[\frac{1}{2}\beta^{-3}-\frac{1}{16}(1-\nu)]b^3$	0	0
17	0	0	$[-\beta^3+\frac{1}{2}(1-\nu)]a$	0	$[\frac{1}{2}\beta^3+\frac{1}{16}(1-\nu)]a^3$	0	0	0	$[\beta^3-\frac{1}{2}(1+4\nu)]a$	0	$[\frac{1}{2}\beta^3-\frac{5}{16}(1-\nu)]a^3$	0
18	0	0	0	0	0	0	0	0	0	0	0	0
19	$2\beta-2(1-\nu)\beta^{-1}$	$-\frac{1}{2}(1-3\nu)$	0	0	0	0	$-2\beta-(1-\nu)\beta^{-1}$	$\frac{1}{2}(1+\nu)$	0	0	0	0
20	$\frac{1}{2}(1-3\nu)$	$-4\beta^{-1}+(1-\nu)\beta$					$\frac{1}{2}(1+\nu)$	$-2\beta^{-1}-(1-\nu)\beta$				
21	0	0	$2(\beta^3-2\beta^{-3})$ $-\frac{1}{2}(14-4\nu)$	$-[2\beta^{-3}+\frac{1}{2}(1-\nu)]b$	$[-\beta^3+\frac{1}{2}(1+4\nu)]a$	0	0	0	$-2(\beta^3+\beta^{-3})$ $+\frac{1}{2}(14-4\nu)$	$-[\beta^{-3}-\frac{1}{2}(1-\nu)]b$	$-[\beta^3-\frac{1}{2}(1-\nu)]a$	0
22	0	0	$[2\beta^{-3}+\frac{1}{2}(1-\nu)]b$	$[\frac{1}{2}\beta^{-3}-\frac{1}{16}(1-\nu)]b^3$	0	0	0	0	$[\beta^{-3}-\frac{1}{2}(1-\nu)]b$	$[\frac{1}{2}\beta^{-3}+\frac{1}{16}(1-\nu)]b^3$	0	0
23	0	0	$[-\beta^3+\frac{1}{2}(1+4\nu)]a$	0	$[\frac{1}{2}\beta^3-\frac{5}{16}(1-\nu)]a^3$	0	0	0	$[\beta^3-\frac{1}{2}(1-\nu)]a$	0	$[\frac{1}{2}\beta^3+\frac{1}{16}(1-\nu)]a^3$	0
24	0	0	0	0	0	0	0	0	0	0	0	0

Figure 4.6b

$K_{2,2}$

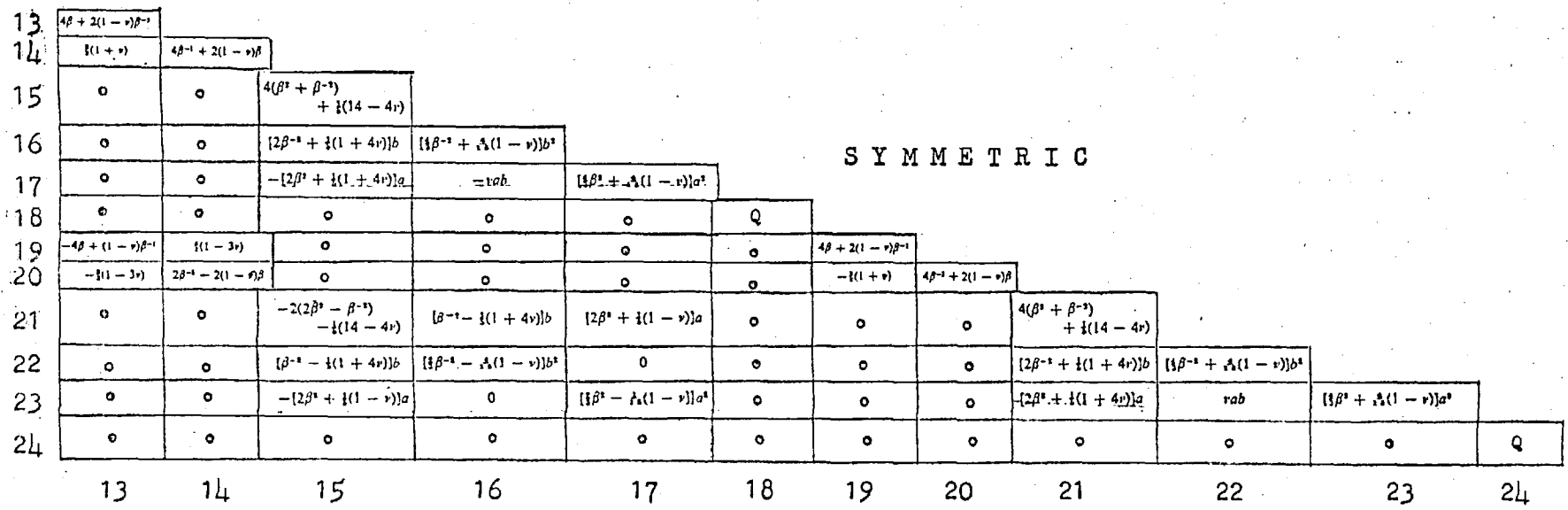


Figure 4.6c

4.2.3 Matrix transformation of coordinates.

The stiffness matrix for each element is defined in terms of a set of local axes. The structure is defined in terms of a set of global axes. Before the individual element stiffness matrices can be added together to form the stiffness matrix which describes the complete structure they must be transformed to the global coordinate system.

If [T] is defined as the matrix which transforms a vector from local to global coordinates then:

$$\{F\}_G = [T]\{F\}_L \qquad \text{and} \qquad \{U\}_G = [T]\{U\}_L$$

and from:

$\{F\}_G = [K]^A_G\{U\}_G$ and substituting we have,

$[T]\{F\}_L = [K]^A_G[T]\{U\}_L$ and since the transformation matrix is orthogonal,

$\{F\}_L = [T]^T[K]_G[T]\{U\}_L$ and since also $\{F\}_L = [K]^A_L\{U\}_L$

then $$[K]^A_L = [T]^T[K]^A_G[T]$$

and $$[K]^A_G = [T][K]^A_L[T]^T$$

In general there will be a separate transformation matrix for each element in the structure. [T] is of block diagonal form and each diagonal block is a 3 * 3 submatrix [R].

In STASYS line elements are used to represent beams and columns whilst rectangular plate elements are used to represent slabs and shear walls. As mentioned in the introduction constraints are placed on the orientation of these components within the structure. The components are shown in their general orientation in figure 4.7. The rotations that must be performed are now described:

1) Wall element

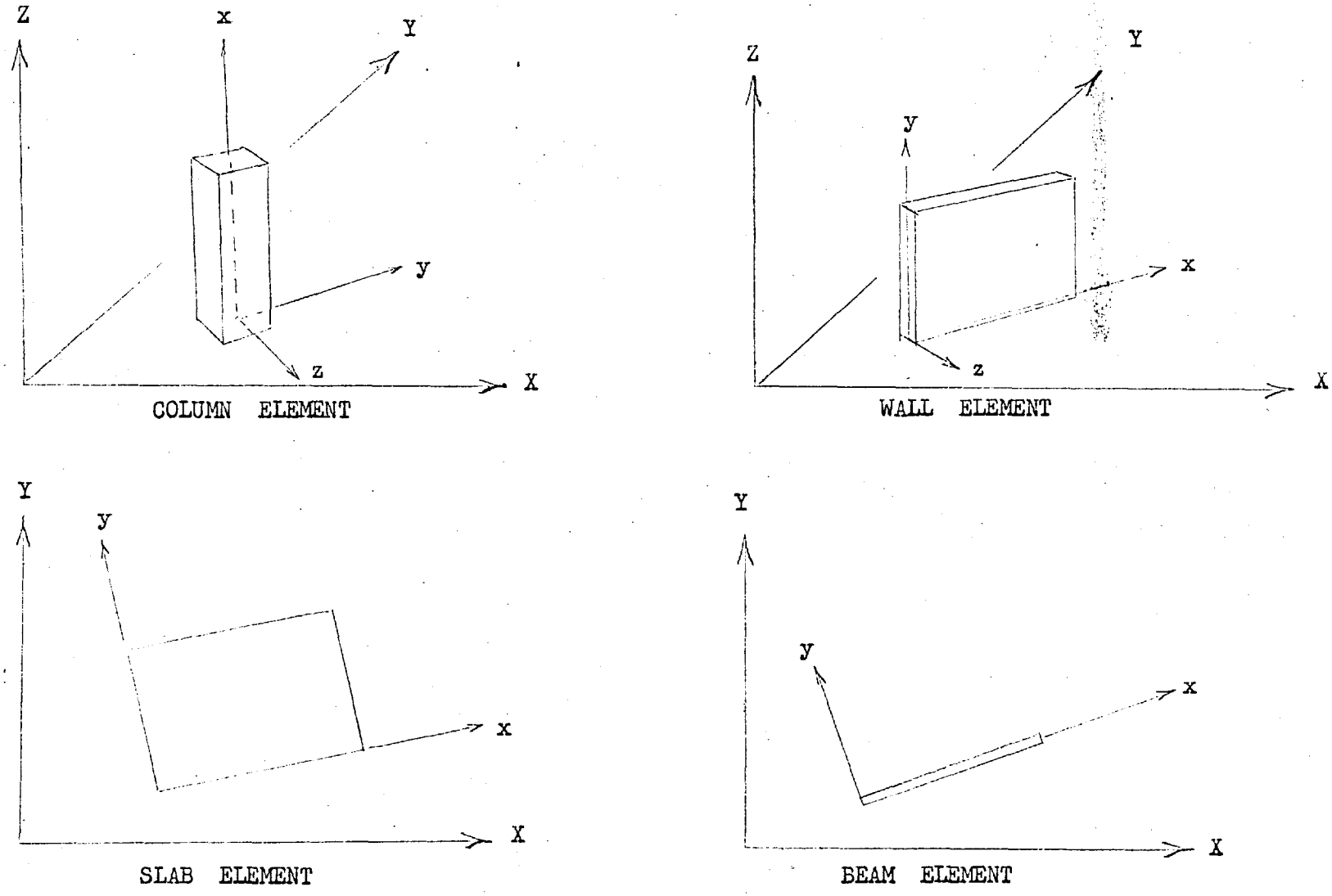


Figure 4.7 Components in their general orientation

a) Rotate about global Z to line up local x and global X.

b) Rotate about global X through 90°.

$$\begin{vmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{vmatrix} \begin{vmatrix} \cos & \sin & 0 \\ -\sin & \cos & 0 \\ 0 & 0 & 1 \end{vmatrix} = \begin{vmatrix} \cos & \sin & 0 \\ 0 & 0 & 1 \\ \sin & -\cos & 0 \end{vmatrix}$$

2) Column element

a) Rotate about global Z so that the orientation point lines up with global X.

b) Rotate about global Y through 90°.

$$\begin{vmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ -1 & 0 & 0 \end{vmatrix} \begin{vmatrix} \cos & \sin & 0 \\ -\sin & \cos & 0 \\ 0 & 0 & 1 \end{vmatrix} = \begin{vmatrix} 0 & 0 & 1 \\ -\sin & \cos & 0 \\ -\cos & -\sin & 0 \end{vmatrix}$$

3) Beam and slab elements

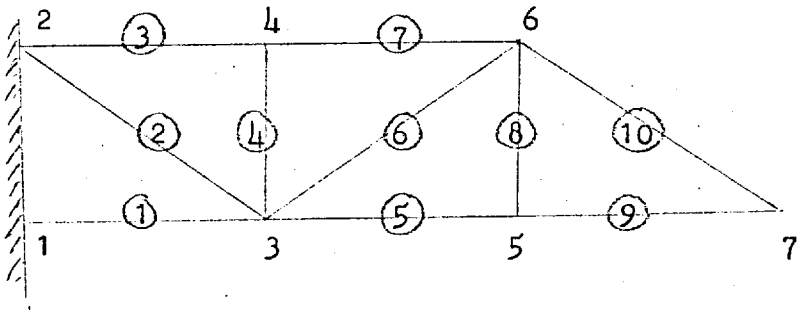
a) Rotate about global Z to line local x with global X.

$$\begin{vmatrix} \cos & \sin & 0 \\ \sin & \cos & 0 \\ 0 & 0 & 1 \end{vmatrix}$$

where 'cos' and 'sin' stand for the sine and cosine of the appropriate angle.

After the element stiffness matrices have been transformed to global coordinates they can be directly added to form the global stiffness matrix.

The assembly process is shown in figure 4.8. The result is always a symmetric banded matrix.



	1	2
1	0	0
2	0	0

+

	2	3
2	0	0
3	0	0

+

	2	4
2	0	0
4	0	0

+ .....

① ② ③ Element stiffness matrices

	1	2	3	4	5	6	7
1	0	0	0				
2	0	0		0			
3	0		0	0	0	0	
4		0	0	0		0	
5			0		0	0	0
6			0	0	0	0	0
7					0	0	0

Stiffness matrix for complete structure

Figure 4.8 Assembly of complete stiffness matrix

4.2.4 Solution of equations

The solution of the matrix equation:

$$[F]_G = [K]_G [U]_G$$

forms the heart of the analysis process. The aim is to solve for U_G the nodal displacements of the structure. For a large analysis problem there will be thousands of equations and even for small problems there are likely to be hundreds of equations. The fact that $[K]_G$ is symmetric and banded reduces the number of coefficients that must be stored but the storage problem, especially on a minicomputer, remains acute.

For a tightly banded system of equations Gaussian elimination is a fast and accurate method of solution. The core storage of a minicomputer is not capable of containing the stiffness matrix, and it is necessary to store the matrix on a fast random access peripheral storage device.

Several methods for performing the solution with only part of the matrix K_G in core at any one moment have been devised, but, usually for a given core capacity the size of problem that can be solved is still limited by the bandwidth of the system of equations. The solution used in STASYS is based on an equation solver written by Cantin[9] and developed for use on PDP 11 minicomputers by Grindley[10].

The Gaussian elimination method has been described in many texts , and usually operates on matrices or systems of equations in which the coefficients are single terms. However in the method presented by Cantin the matrix is divided into blocks of n by n and the Gaussian elimination is carried out on the blocks. Gaussian elimination requires the reciprocal of the diagonal terms of the matrix to produce multipliers. When operating with blocks of the matrix as terms, the inverse of the diagonal blocks of the matrix is used. The method only requires three blocks of the matrix and three equivalent load vectors to be in core at any one time and thus the size of

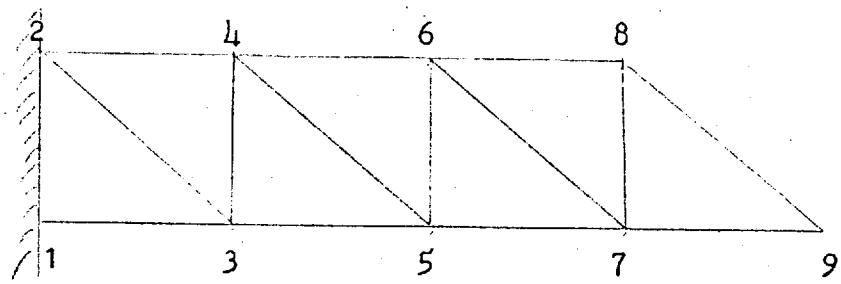
problem that can be solved is only limited by the quantity of backing store that is available. Because of the relatively slow speed of disk accesses the solution becomes faster as the number of blocks needed to store the matrix is reduced. The size of the n by n blocks is therefore chosen to be as large as possible given a fixed amount of core storage.

The solution is carried out in double precision, that is each variable is represented by eight bytes (four PDP 11 words). The disk is physically divided into 256 word records (64 double precision variables) and it is only possible to start a random access read or write at the beginning of each record.

$[K]_G$ is partitioned into n by n blocks and it is convenient and efficient if n is a multiple of the number of degrees of freedom at each node, in this case six, and if n^2 is a multiple of the number of variables per disk record, in this case sixty-four. Thus the lowest convenient value for n in STASYS is twenty-four. If this value were adopted each block of the matrix would occupy exactly nine disk records. Unfortunately it was discovered that the 16 kilowords of core available was not sufficient. STASYS therefore uses a block size of eighteen by eighteen which requires five and one sixteenth disk records. Six records are used for each block, fifteen sixteenths of the sixth disk block is wasted.

Figure 4.9 shows how the matrix is divided into blocks, the smaller squares representing the six by six submatrices at each node and the heavier squares representing the eighteen by eighteen blocks in which the matrix is stored.

STASYS runs on a PDP 11/45 with a single RK05 disk pack which has a capacity of 1.2 million words. If the whole disk unit is used for storing $[K]_G$ then a problem with 1500 degrees of freedom, a half band width of 190 and 10 load vectors can be solved. In practice the 1.2 million word disk is also used for storing program and other data, and only half the disk is



	1	2	3	4	5	6	7	8	9
1	0	0	0						
2	0	0	0	0					
3	0	0	0	0	0				
4		0	0	0	0	0			
5			0	0	0	0	0		
6				0	0	0	0	0	
7					0	0	0	0	0
8						0	0	0	0
9							0	0	0

Figure 4.9 Block division of complete stiffness matrix.

available for storage of the stiffness matrix. However the price of disk storage per byte has already dropped by a factor of four since the present equipment was purchased. New minicomputer systems will therefore have larger disk storage capacities and it may become solution time rather than storage space that becomes the practical limiting factor.

4.2.5 Inclusion of boundary conditions

In general it is possible to modify terms in the overall stiffness matrix and load vectors in order to allow for specified displacements. The simplest method of doing this is to multiply the diagonal term in the stiffness matrix corresponding to the specified displacement by a large number, and to replace the load terms by the new diagonal term times the specified displacement. In the special case where the specified displacement is zero the load term will also become zero and it is this case that is allowed for in STASYS.

However as we have seen the method of solution involves partitioning the matrix into blocks and the diagonal blocks must be inverted. The effect of having terms of excessively large value on the diagonal within a block is to cause ill conditioning during the block invertion.

Zinciewicz suggests an approach in which a zero is written in every term of the row and column corresponding to the fixed degree of freedom and the diagonal term is replaced by unity. This also causes ill conditioning because the diagonal term is now excessively small compared with the others.

There is however no need to replace the diagonal term by unity since if the terms in its row and column are all zero its value can have no effect on the solution. In fact if the diagonal term is left alone and in the general case the load terms are replaced by the diagonal term times the specified displacement the desired result can be achieved without causing any ill conditioning.

4.3 Data input

Data input in STASYS falls into seven categories:

- Element nodal point coordinates
- Element dimensions
- Element material set number
- Sets of material properties
- Floor heights
- Nodal constraints
- Loads

The data input programs are interactive and use many of the facilities of the draughting system described in chapter two.

In the design of STASYS effort has been made to keep the system flexible. The operator is able to choose the order in which he wishes to prepare data. Similar parts of a structure may be quickly reproduced by using the 'sub-picture' processor and the three editors of the draughting system. Forty of the semi-permanent mass storage files are reserved for the users own requirements and sixty for storing the graphical representation of each layer or floor of the structure.

Data can either be taken from existing General Arrangement drawings, floor by floor, or the designer may create a model from sketches of a proposed building. Whichever the starting point, the mode of operation is as follows:

a) A grid is digitised or constructed which is common to all floors of the structure. This is normal building design practice and provides a useful framework for positioning elements and other data. It is not essential for the analysis.

b) The positions of the line and rectangular elements which represent columns, walls, beams and slabs on the current floor are digitised. The points are entered in 'background' mode described in section 2.4.2 and

and discussed in chapter three.

The elements are entered into the 'workspace' as 'symbols' and may be edited and manipulated by the macro editor. Initially they are set to have zero dimensions, and the symbol display programs display them using values calculated from their coordinate values and default dimensions stored in data statements within the programs.

c) Dimensions are assigned to the elements. Dimensions which are defined by the coordinate positions of the nodal points may be omitted. The user is able to set up six sets of dimensions corresponding to buttons one to six on the pencil. Each set consists of the local x, local y and local z dimensions as set out in figures 4.10 to 13. By 'finding' an element and pressing a button the operator enters the dimensions into the element symbol in the workspace. It is found that since many elements have the same dimensions it is possible to enter the dimensions of all the elements without too much resetting of the dimensional sets. The same philosophy is used during the entry of loads.

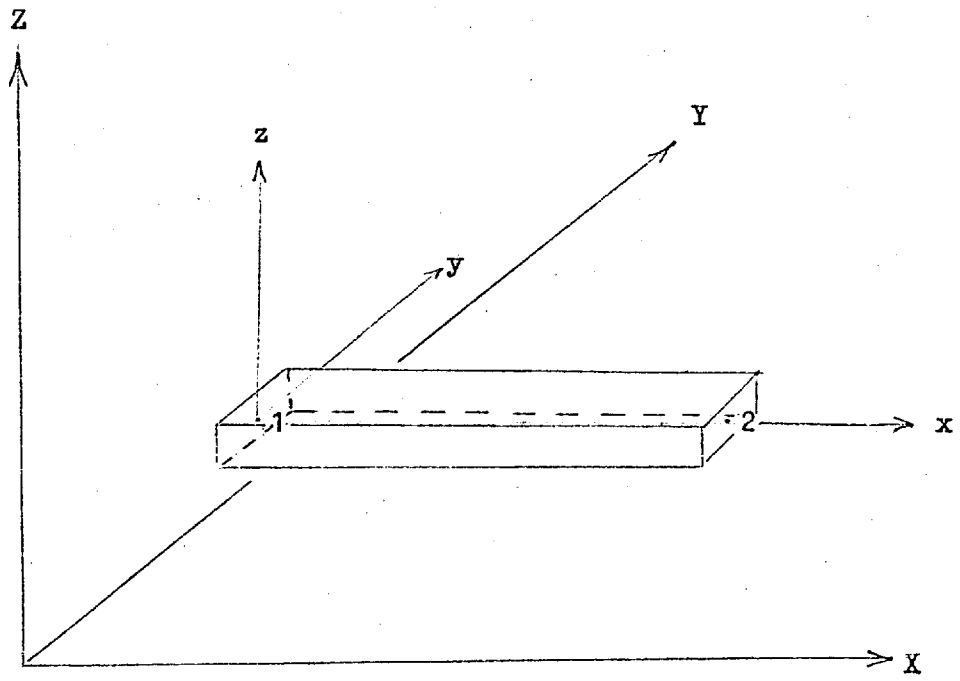
d) Material properties are assigned to each element. One set of material properties is defined to be allocated by default, i.e. the properties of this set are used in the analysis if no other set of properties is assigned to an element. In order to minimise the work of assigning material sets the default properties are chosen to be those which apply to the largest number of elements. Other material sets are entered via the keyboard as and when required during the assignment process. Once a set of material properties has been entered in this way it remains available for use until deleted or overwritten by the user. Up to forty-two sets are allowed.

e) Boundary conditions are input. The user selects the type of restraint required as follows:

Button 1 - zero displacement in global X

Button 2 - zero displacement in global Y

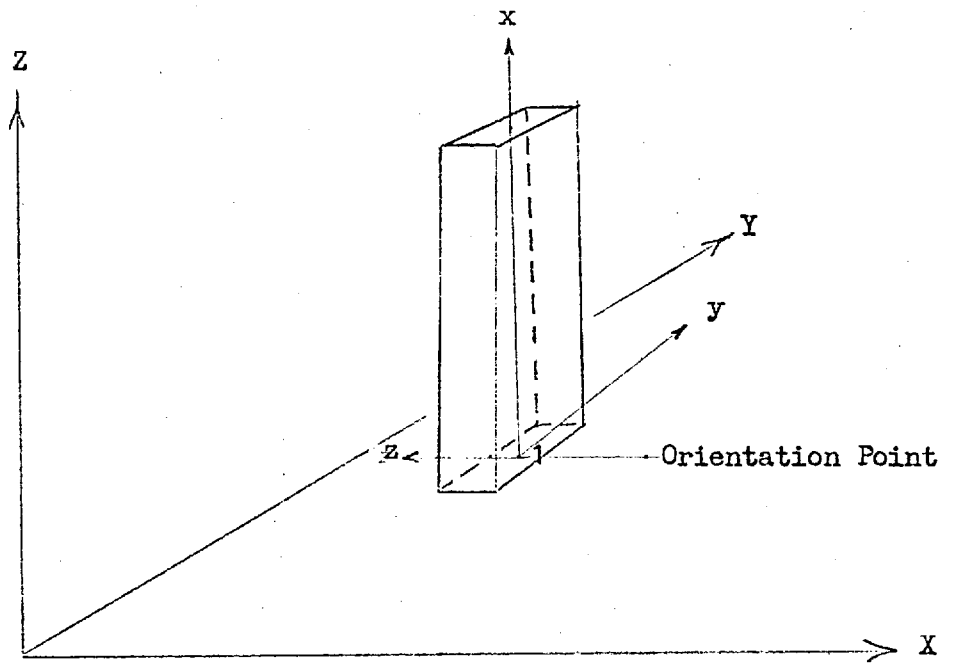
BEAM ELEMENT.



ICODE	X-POINT	Y-POINT
13	SYMBOL NUMBER	ELEMENT NUMBER
14	X-POINT ONE	Y-POINT ONE
14	X-POINT TWO	Y-POINT TWO
19	LENGTH	WIDTH
16	DEPTH	MATERIAL NUMBER

Figure 4.10

COLUMN ELEMENT.



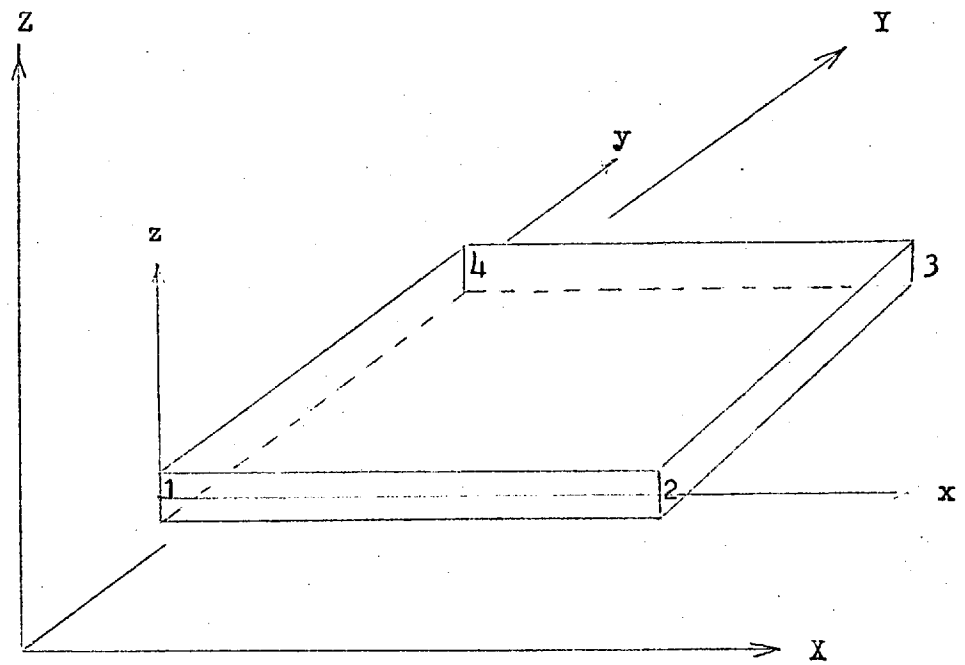
ICODE	X-POINT	Y-POINT
13	SYMBOL NUMBER	ELEMENT NUMBER
14	X-ORIGIN	Y-ORIGIN
14	POINT TO DEFINE ORIENTATION ABOUT Z-AXIS	
19	HEIGHT	BREADTH*
16	WIDTH**	MATERIAL NUMBER

* BREADTH: The distance across the column in the direction of the line joining the Origin to the Orientation Point.

**WIDTH: The distance across the column in the direction at right angles to the line joining the Origin to the Orientation Point.

Figure 4.11

SLAB ELEMENT.



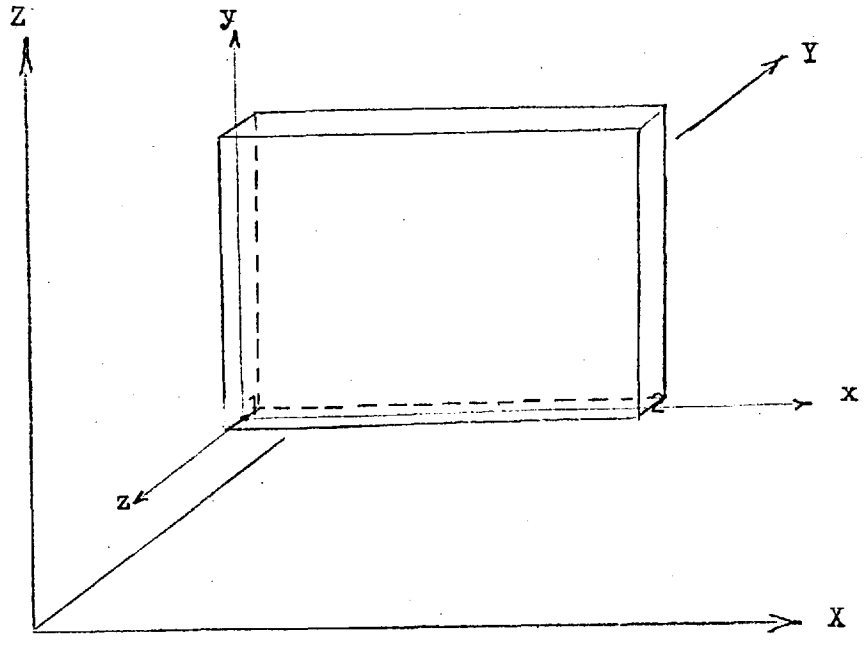
ICODE	X-POINT	Y-POINT
13	SYMBOL NUMBER	ELEMENT NUMBER
14	X-POINT ONE	Y-POINT ONE
14	X-POINT TWO	Y-POINT TWO
14	X-POINT THREE	Y-POINT THREE
14	X-POINT FOUR	Y-POINT FOUR
19	BREADTH*	WIDTH**
16	THICKNESS	MATERIAL NUMBER

*BREADTH: Distance between Point One and Point Two.

**WIDTH: Distance between Point One and Point Four.

Figure 4.12

WALL ELEMENT.



ICODE	X-POINT	Y-POINT
13	SYMBOL NUMBER	ELEMENT NUMBER
14	X-POINT ONE	Y-POINT ONE
14	X-POINT TWO	Y-POINT TWO
19	LENGTH	HEIGHT
16	THICKNESS	MATERIAL NUMBER

Figure 4.13

Button 3 - zero displacement in global Z

Button 4 - zero rotation about global X axis

Button 5 - zero rotation about global Y axis

Button 6 - zero rotation about global Z axis

Button 7 - zero displacement in all degrees of freedom.

He is then able to select the nodal points to which the constraint applies using background mode.

More than one set of boundary conditions can be stored. This is achieved by storing the sets under different 'levels'. Ten levels are reserved for boundary condition data.

The boundary conditions are displayed schematically as shown in figure 4.1

f) Loads are input. Three types of load can be input:

- Point loads
- Uniformly distributed line loads
- Uniformly distributed area loads

Moments are applied as two point loads producing the required moment. In practice there are no external moments applied to building structures.

The loads may be applied by entering the positions of the loaded points, lines or areas in 'background' mode. When defining an area load a rectangle is digitised. Any slabs within the rectangle will have the load imposed on them. As with the input of element dimensions six load vectors can be set up and selected from the pencil buttons. For a point load the value of a vector is interpreted in kilo-newtons, for a line load in kilo-newtons per metre and for an are load in kilo-newtons per square metre. The loads are displayed schematically as shown in figure 4.14.

As with boundary conditions loads can be stored under different levels. In the analysis up to four combinations of different load sets may be included. The way in which the levels and semi-permanent mass storage files are allocated in STASYS is shown in figure 4.15.

Display of boundary condition symbols.

X Y Z
AX AY AZ * nodal point fixed in all degrees of freedom.

X * nodal point - no movement in X axis.

X Y
AZ * nodal point - no movement in X axis
- no movement in Y axis
- no rotation about Z axis

Display of load symbols.

> load acts in positive X direction

∧ load acts in positive Y direction

O load acts in positive Z direction

< load acts in negative X direction

∨ load acts in negative Y direction

■ load acts in negative Z direction

>150.00 point load positive X, 150 kN

150.00
■ ■ ■ ■ ■ line load negative Z, 150.00 kN/M

■ ■ ■ ■ ■ area load negative Z, ·75 kN/M^2

■ ■ ■ ■ ■

■ ■ ■ ■ ■

·75
■ ■ ■ ■ ■

■ ■ ■ ■ ■

■ ■ ■ ■ ■

Figure 4.14

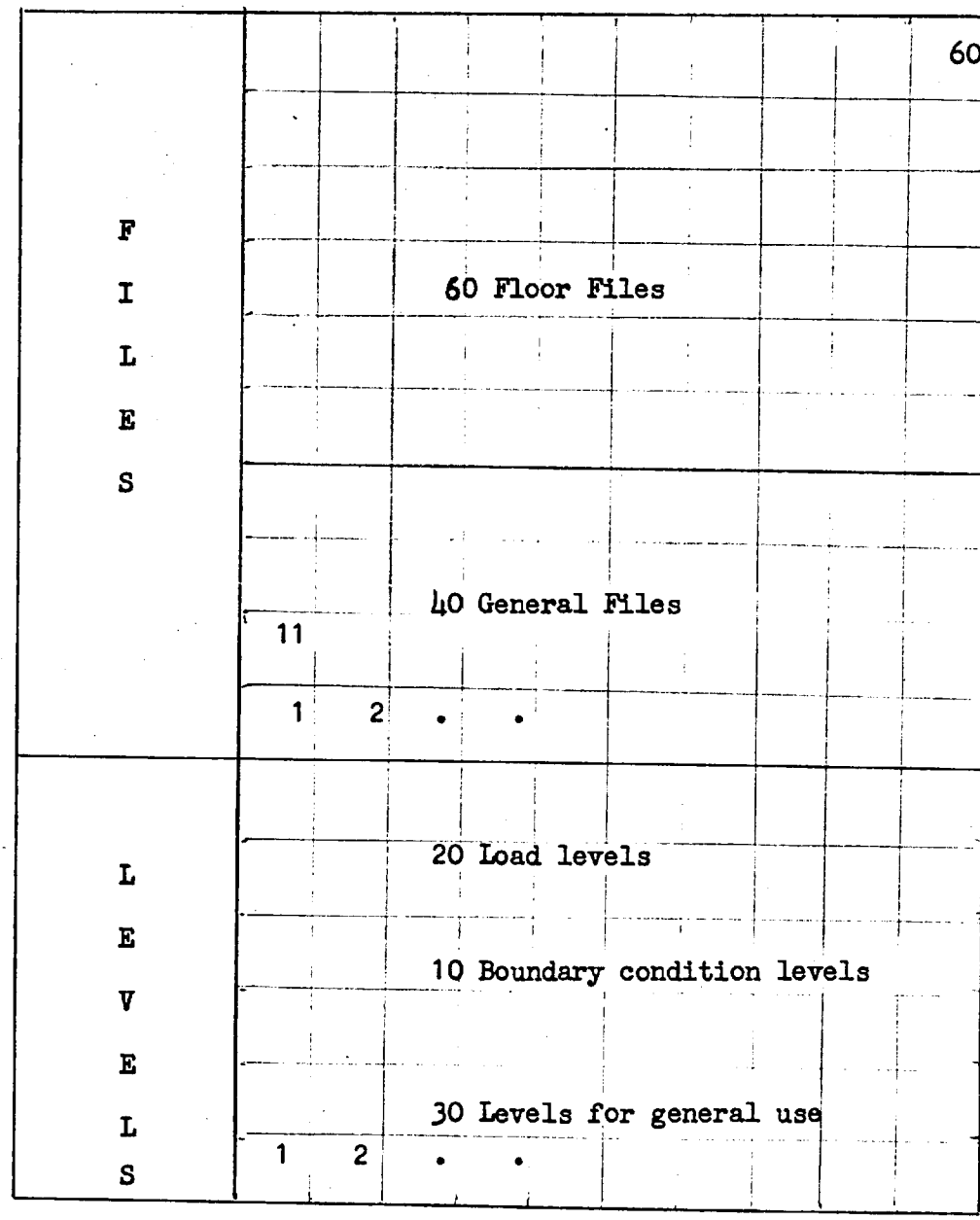


Figure 4.15 Allocation of levels and files by STASYS.

g) When a complete model of a floor has been built in the workspace, it is filed under the appropriate floor file. A typical view of a floor is given in figure 4.16.

At this stage it is convenient to enter the level of the floor in the floor directory. The height of each floor is entered in metres above some datum point. A list of floors and floor heights is displayed on the screen by request.

After the graphic data for every floor of the structure has been produced and filed the data is analysed and reformed for the analysis process. To avoid confusion the processing of the data into a suitable form is called data collation rather than data analysis.

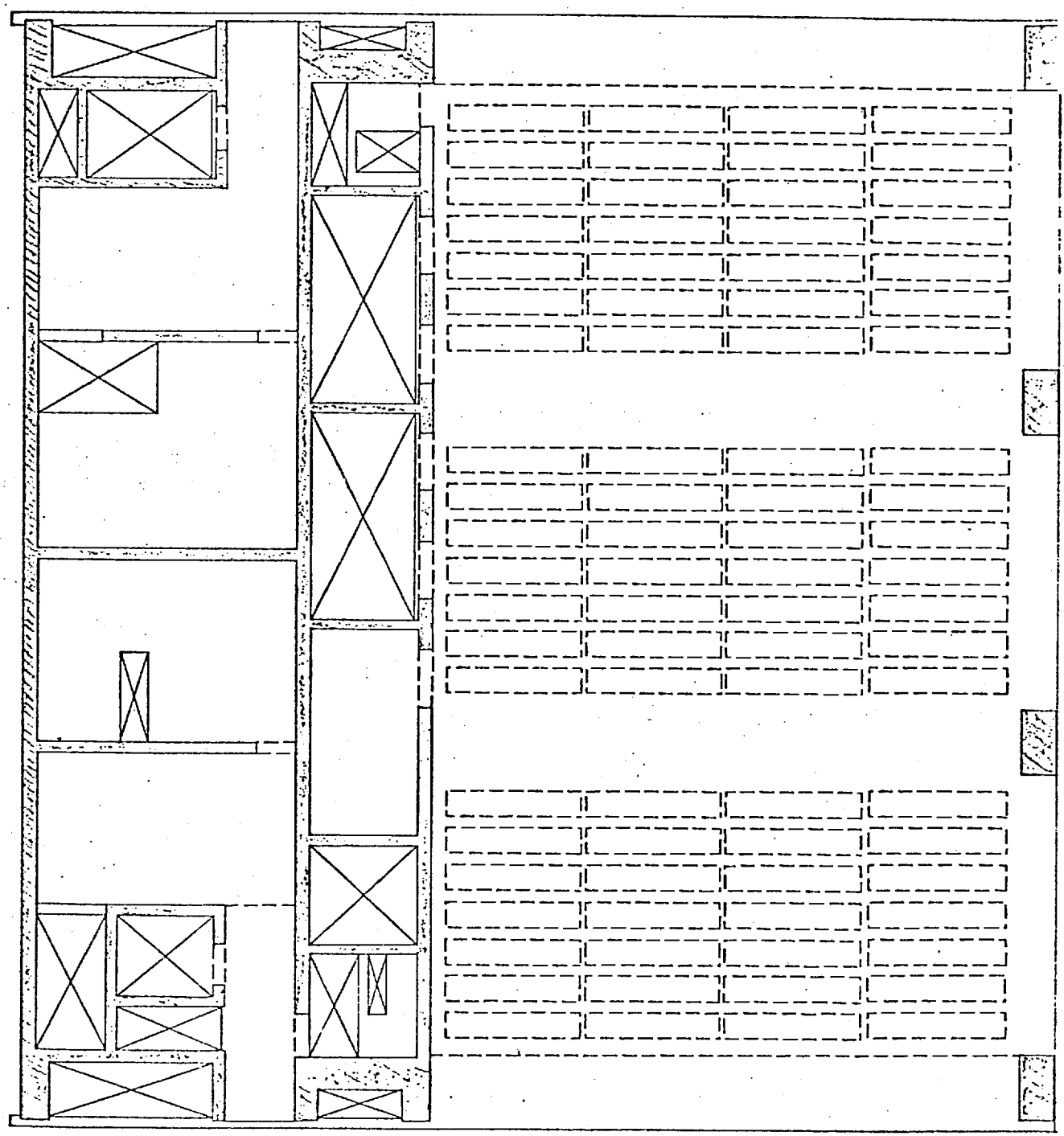
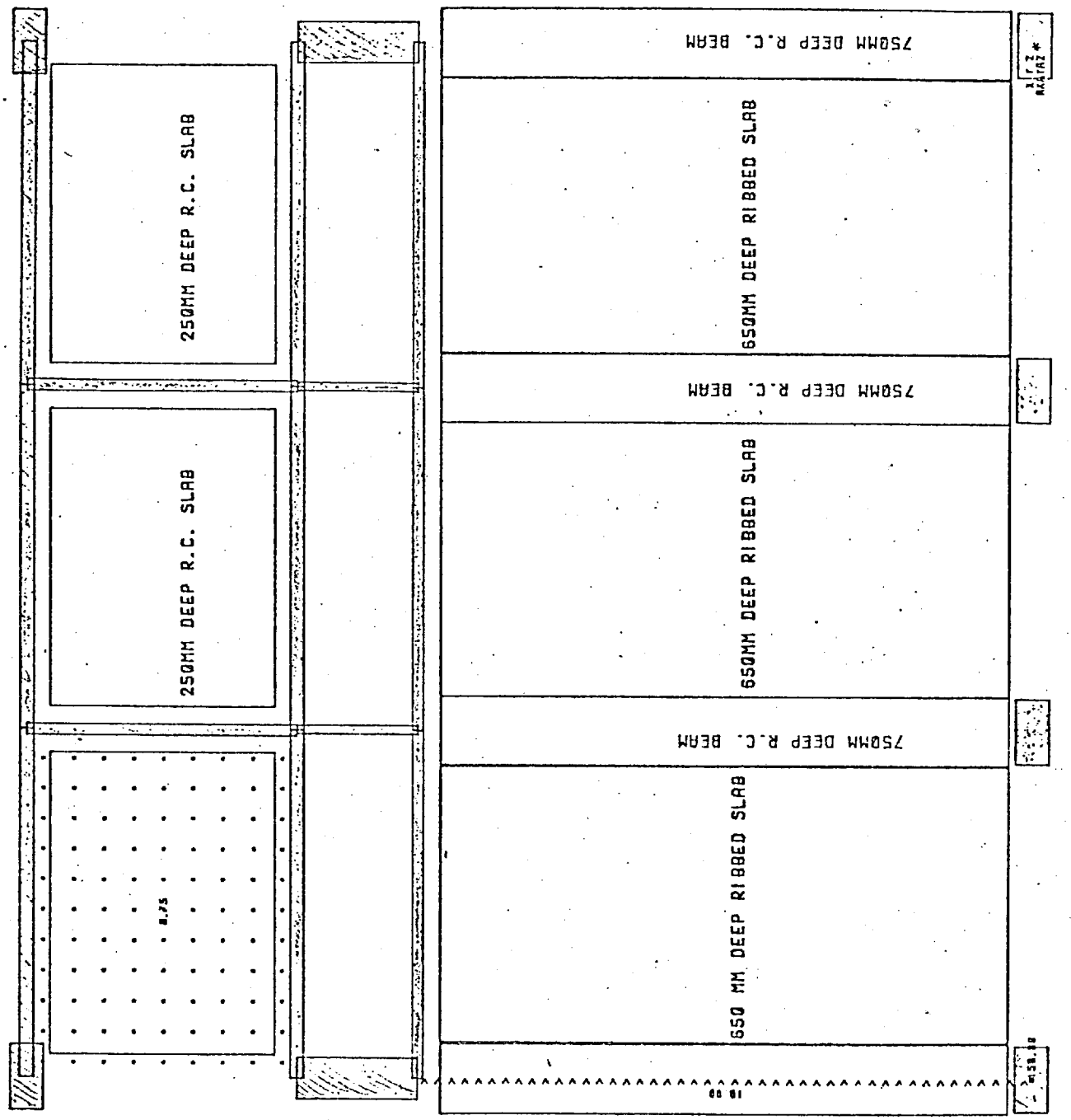


Figure 4.16a Typical plan of building floor.



Figurev .16b View of idealised floor

4.4 Data collation

Data collation can be divided into five stages:

- Sort nodal points, create file of points
- Sort boundary conditions, create file of constraints for each floor.
- Analyse loads, create file of nodal loads
- Data output if analysis on mainframe
- Element and load assembly if analysis on minicomputer

During the first three processes the graphic data for each floor is analysed and packed into three random access files. The filing is carried out by file handlers like the one described in chapter two. The fourth process, data output, is only performed if the solution of the equations is to be carried out on a different computer or if a hard copy of the problem data is required. If the solution is to be performed on the minicomputer then the element stiffness matrices and load vectors are formed and assembled. The last two processes use the data set up by the first three.

A brief summary of what each stage involves is given:

1) Sort nodal points.

Data for the floor under consideration is transferred by the operator to the workspace. Points defining the ends of line elements and the corners of rectangular plate elements are stripped out and stored in a temporary workfile. If two points are within 100mm of each other they are considered as the same point.

The points are ordered by a simple bubble sort. The operator indicates whether should be ordered primarily according to their X or Y coordinate values. Points having a primary coordinate value difference of less than the 'grid factor' are ordered according to the value of their secondary coordinates.

The ordered points are displayed on the screen with their nodal numbers so that the user can visually check the result. Finally the points are filed in CADMAC.RA3 by the file handler.

2) Sort boundary conditions.

The user selects which levels of boundary conditions he wishes to use for the analysis. Points are stripped from the boundary condition symbols in the workspace along with the nodal constraint(s).

Each constraint is given a value according to the algorithm:

$$\text{Value} = 2^{(n-1)}$$

where n is the number of the constrained degree of freedom as defined by the number of the pencil buttons during the boundary condition input.

Where more than one constraint occurs at the same point the values are added. The total value is stored with the point coordinates. It is a simple matter to decode the total values during load assembly.

The points with constraint values are filed in CADMAC.RA5 by the relevant file handler.

3) Analyse loads.

Data analysis of point loads is simple, for uniformly distributed line and area loads a search has to be carried out to determine which beam and slab components of the structure bear the load. Loads are then apportioned to the nodes of these beams and slabs. If it is found that more than 25% of the applied load has not been distributed a diagnostic is printed. This does not necessarily mean that the program is in error because it may be that an area load has been applied over an area that has voids in it.

Any nodes with external loads acting on them are filed in CADMAC.RA9.

Loads due to the self weight of the structural elements are calculated and assembled during the formation and assembly of the element stiffness

matrices.

4) Data output.

The following data is output to a specified device, usually the line printer or magnetic tape unit.

- Nodal coordinates
- Element connections and material numbers
- Sets of material properties
- Nodal constraints
- Loads

5) Assembly of element stiffness matrices and load vectors.

This stage includes the formation, transformation and assembly of the stiffness matrices for each element, as well as the formation and assembly of the loading data.

The outline of the process involved can best be conveyed by a flow diagram figure 4.17.

Because of the way boundary conditions have been included, it is necessary to zero the loads where a degree of freedom has been removed. This is carried out after the assembly is complete. It should be noted that the relevant rows and columns of stiffness coefficients are zeroed in the element stiffness matrices before they are assembled. This saves having to do a lot of reading and writing to the disk to put them in after the matrix has been assembled and is stored on disk.

There is little technical difficulty involved in the assembly process, but it is a lengthy process. The time for assembly of each line element is approximately 1.5 seconds, and for each rectangular plate element 2 seconds. A twenty floor building represented by thirty elements per floor therefore takes approximately twenty minutes to assemble.

Assembly proceeds floor by floor through the structure and each floor plan is displayed prior to assembly enabling progress to be monitored.

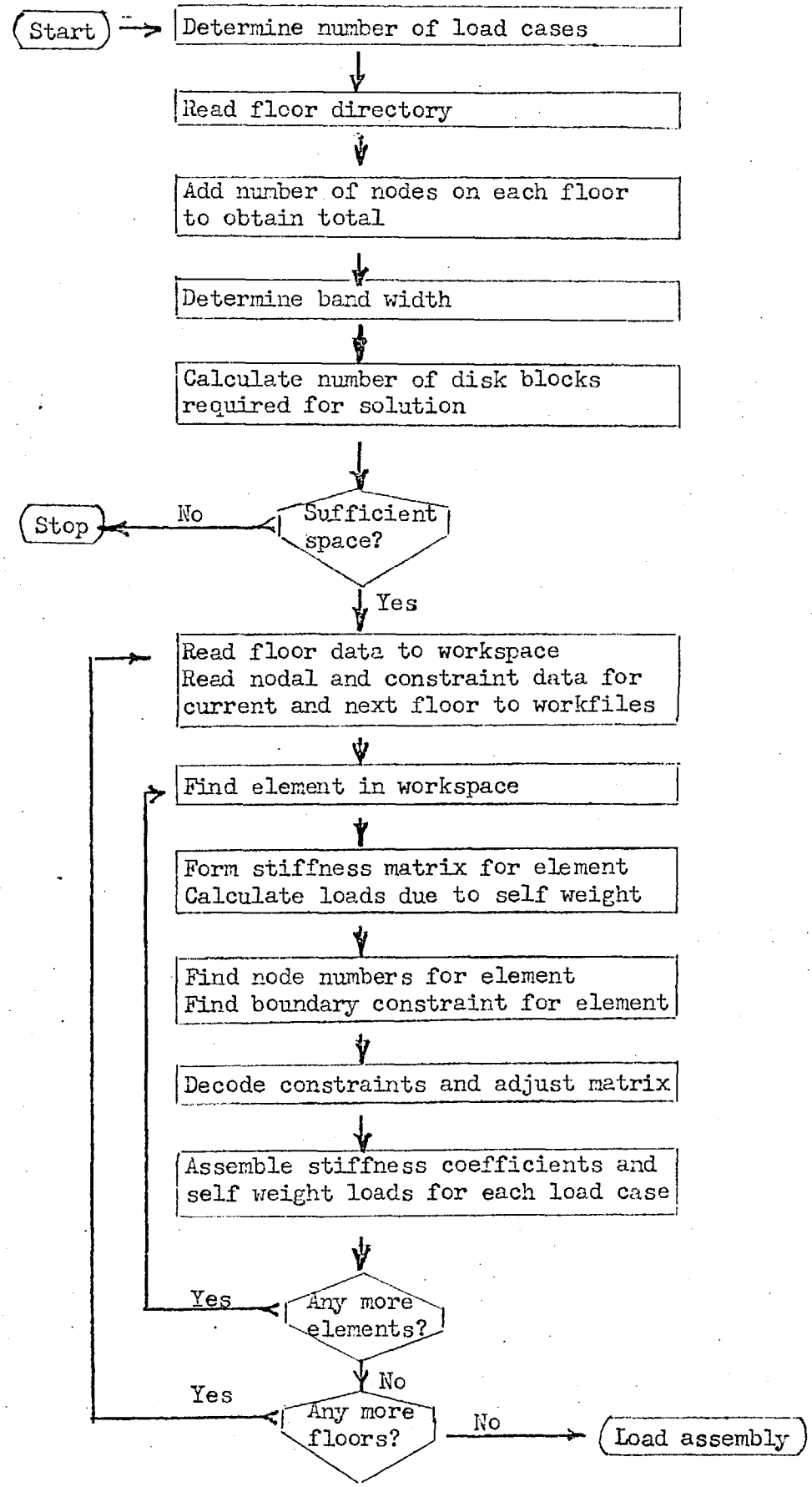


Figure 4.17 The assembly process.

4.5 Solution

The method used for solution has been described in section 4.2.4. Condition numbers are calculated during the inversion of the diagonal blocks of the assembled stiffness matrix. These are listed on the keyboard or line printer as each block is inverted. The calculated displacements are written back to the disk unit over the original load vectors.

Times for solution are plotted in figure 4.18.

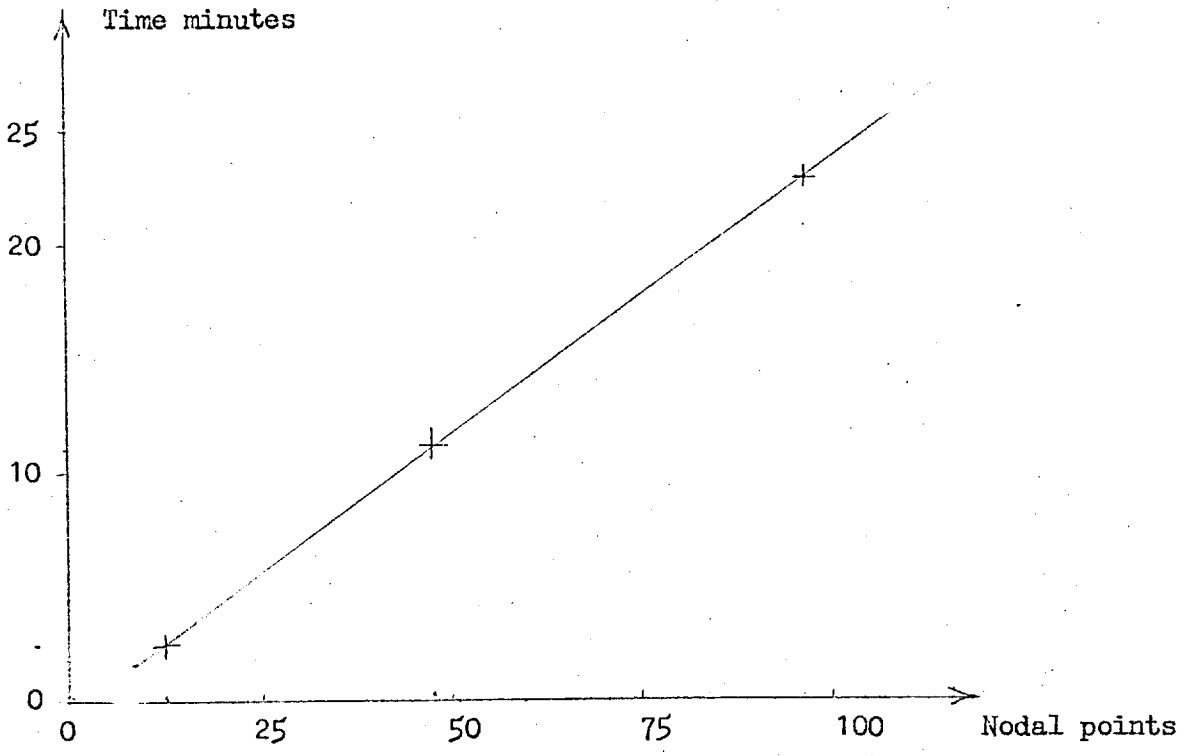


Figure 4.18 Time for solution against nodal points. Each of the three sets of equations has a band width of 12 nodal points. Six degrees of freedom at each nodal point.

It can be seen from figure 4.18 that the time required for solution rises linearly as the number of nodal points in the problem increases, so long as the band width of the problems remains the same. In general the band width of problems increases as their size and figure 4.18 is therefore optimistic.

4.6 Output of results

Three methods are used to communicate the result to the engineer:-

a) List of global displacements.

b) Three dimensional display of displaced structure.

c) Stresses of selected elements.

A list of global displacements for each node may be output to any device.

The user is able to get a much more immediate and clear view of the deformations of the structure by displaying them in three dimensions.

Instead of a full schematic view of all the elements making up the structure, a view of lines joining the nodal points is displayed for two reasons:-

1) An adequate hidden line removal package for use on a minicomputer has not yet been developed.

2) In the absence of 1), a full view of the structure is generally too complex for the human eye to appreciate.

A set of lines joining nodal points to their neighbours is generated to represent the structure. These can be displayed before and after solution. When displayed after solution, the operator is asked to specify a factor by which all displacements shall be multiplied in order to make them visible. If necessary, the original structure may be displayed as well as the displaced structure, in order to provide a frame of reference.

The three dimensional viewing package allows the operator to set the following parameters:-

Spherical or flat projection plane.

Centre of view in cartesian coordinates.

Viewpoint in polar coordinates.

Distance of the projection plane from the viewpoint.

Position of the projection on the display screen.

Figure 4.19 shows a view of a simple frame in its original and displaced states.

The third option allows the user to select a particular element in the structure and obtain a print out of the stresses in that element.

The operator calls the appropriate floor file to the workspace. He is then able to 'find' the required element(s) and request a list of the stresses.

If a rectangular element is selected, the stresses due to in-plane forces and bending moments are calculated separately for the centre and each corner of the element.

For line elements, stresses are calculated at each end and the centre of the element.

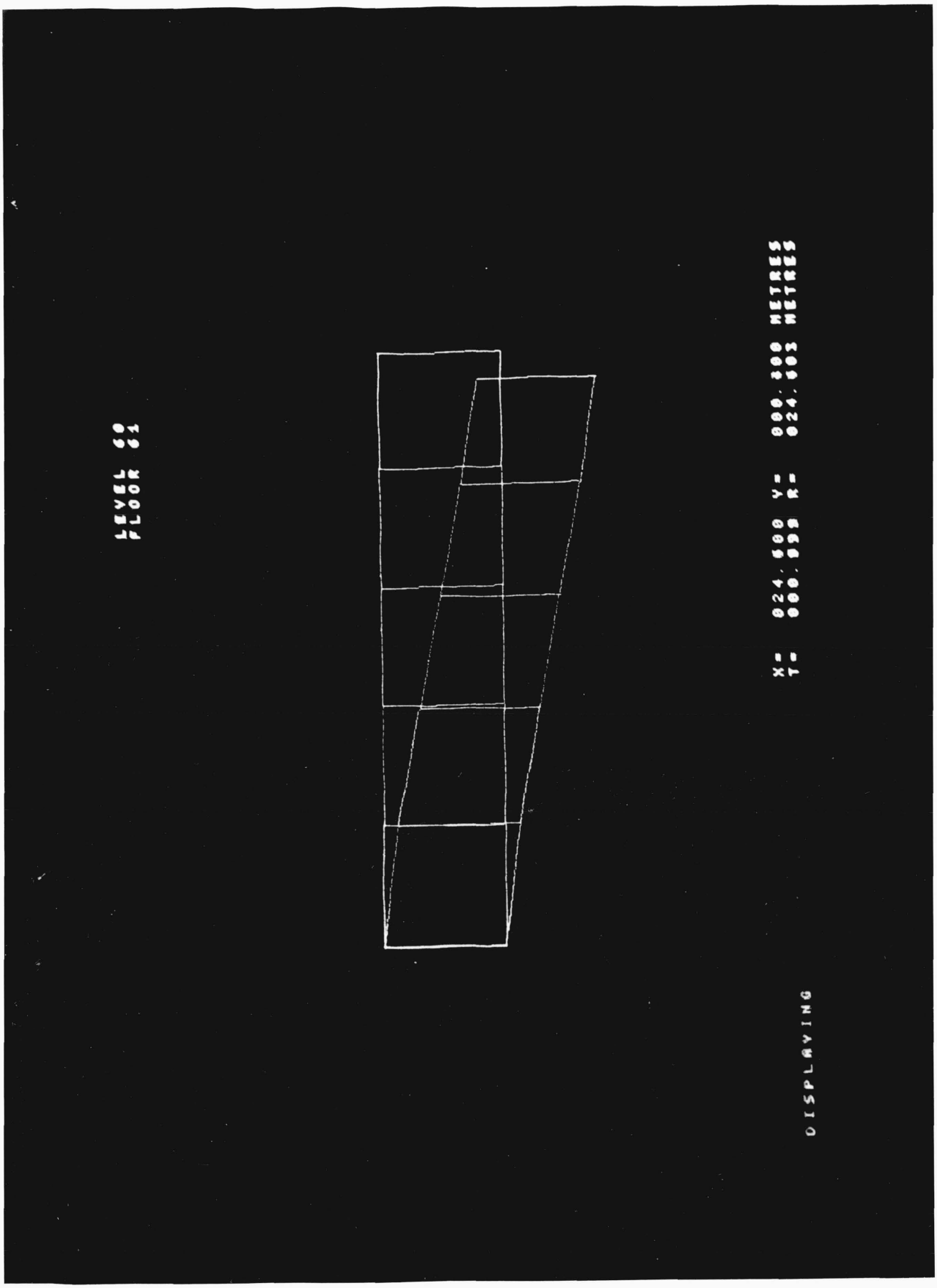


Figure 4.19 A simple frame in its original and displaced states.

4.7 An illustrated example of the use of STASYS

This section is intended to draw together all that has gone before and to give the reader a better feel of the physical reality of operating STASYS. It therefore contains only a flow chart outlining the sequence of operations (figure 4.20) and a series of photographs of the display screen during the analysis process.

The example is an eight floor building with a central core represented by rectangular plate elements and columns spaced round the perimeter of the building connected to the core by beams. The building is represented by a total of 200 elements joined at 96 nodes.

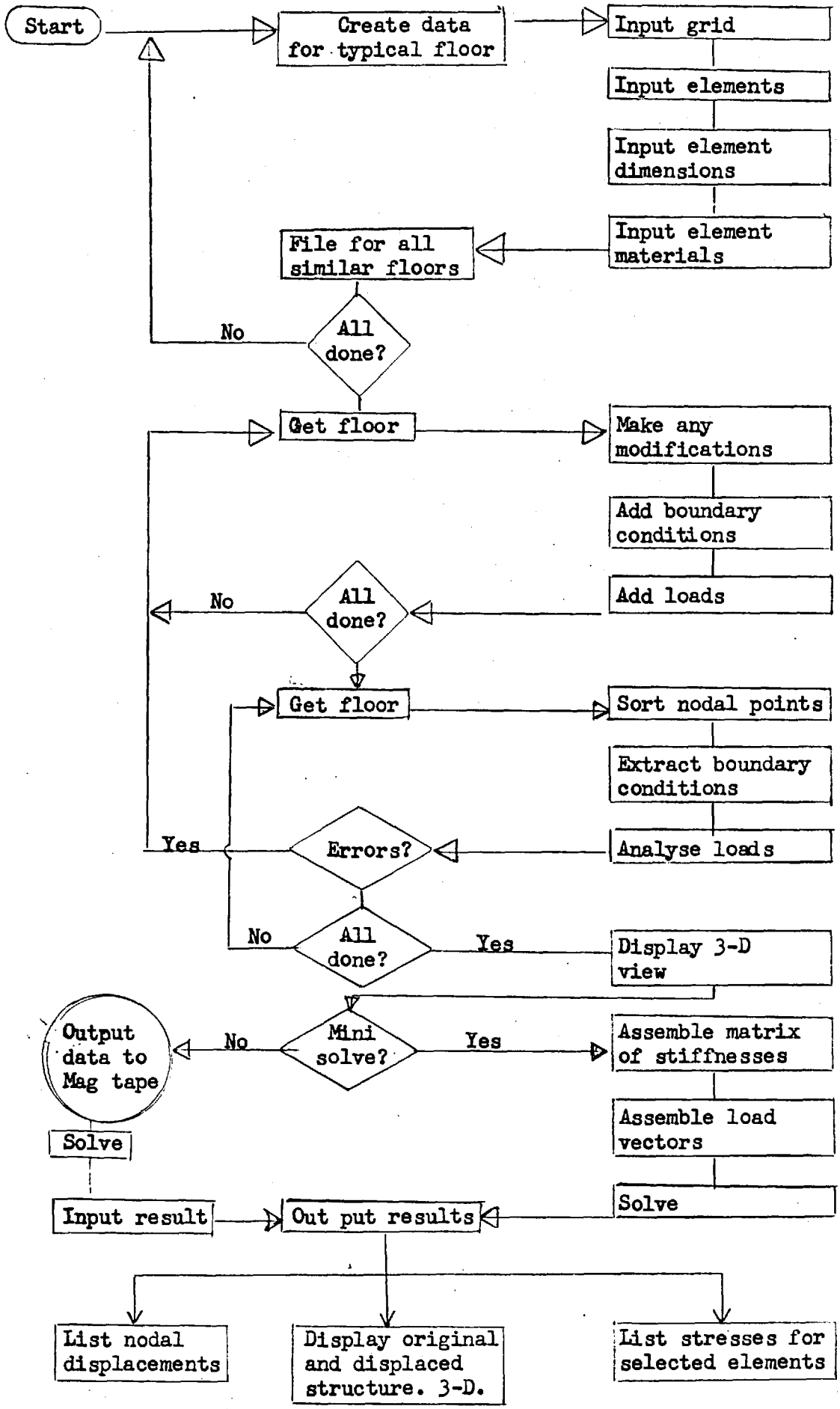


Figure 4.20 Operation of STASYS

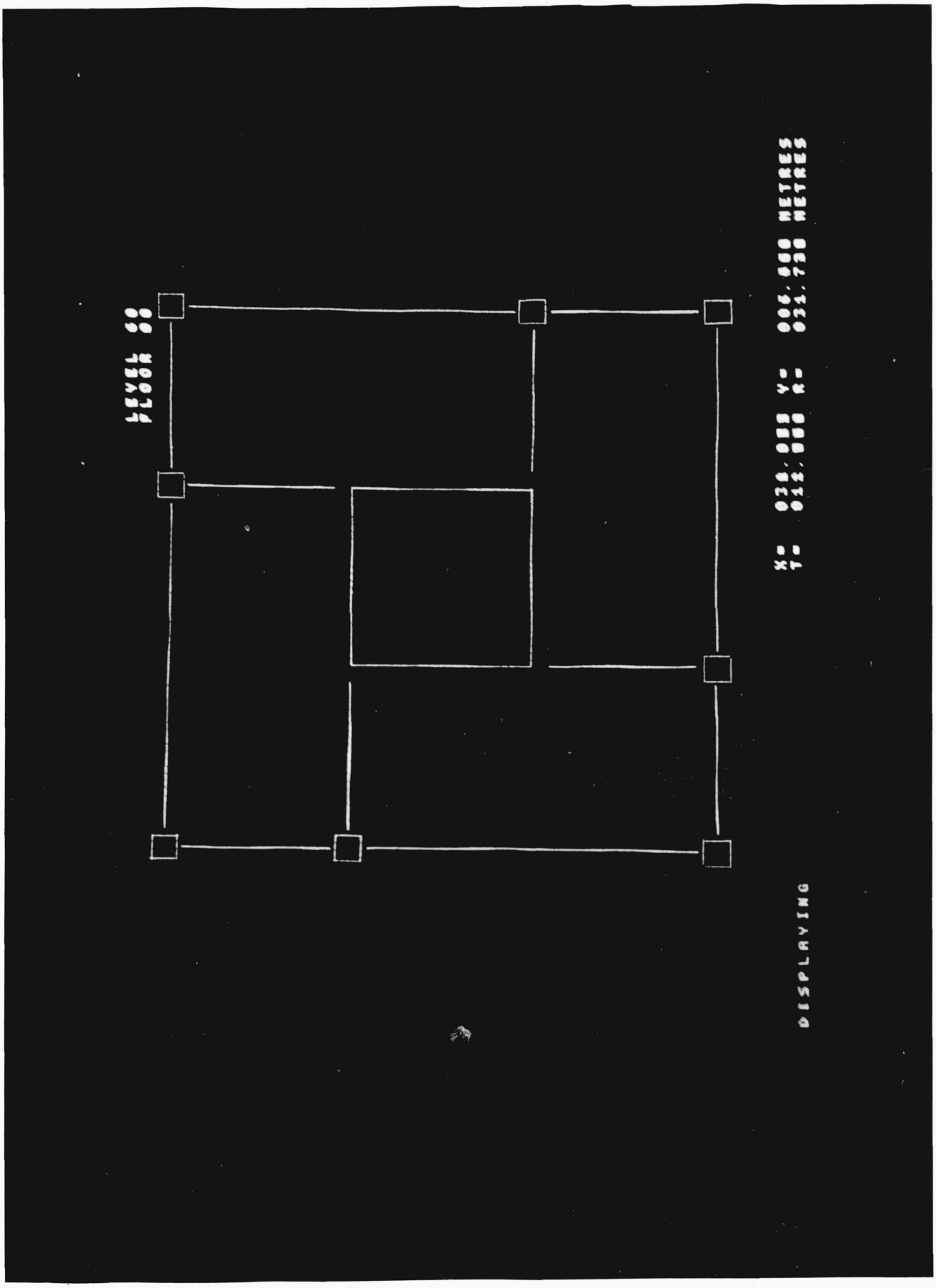


Figure 4.21 Schematic view of a typical floor (dimensionless elements).

Figure 4.22 Adding boundary conditions - elements now dimensioned.

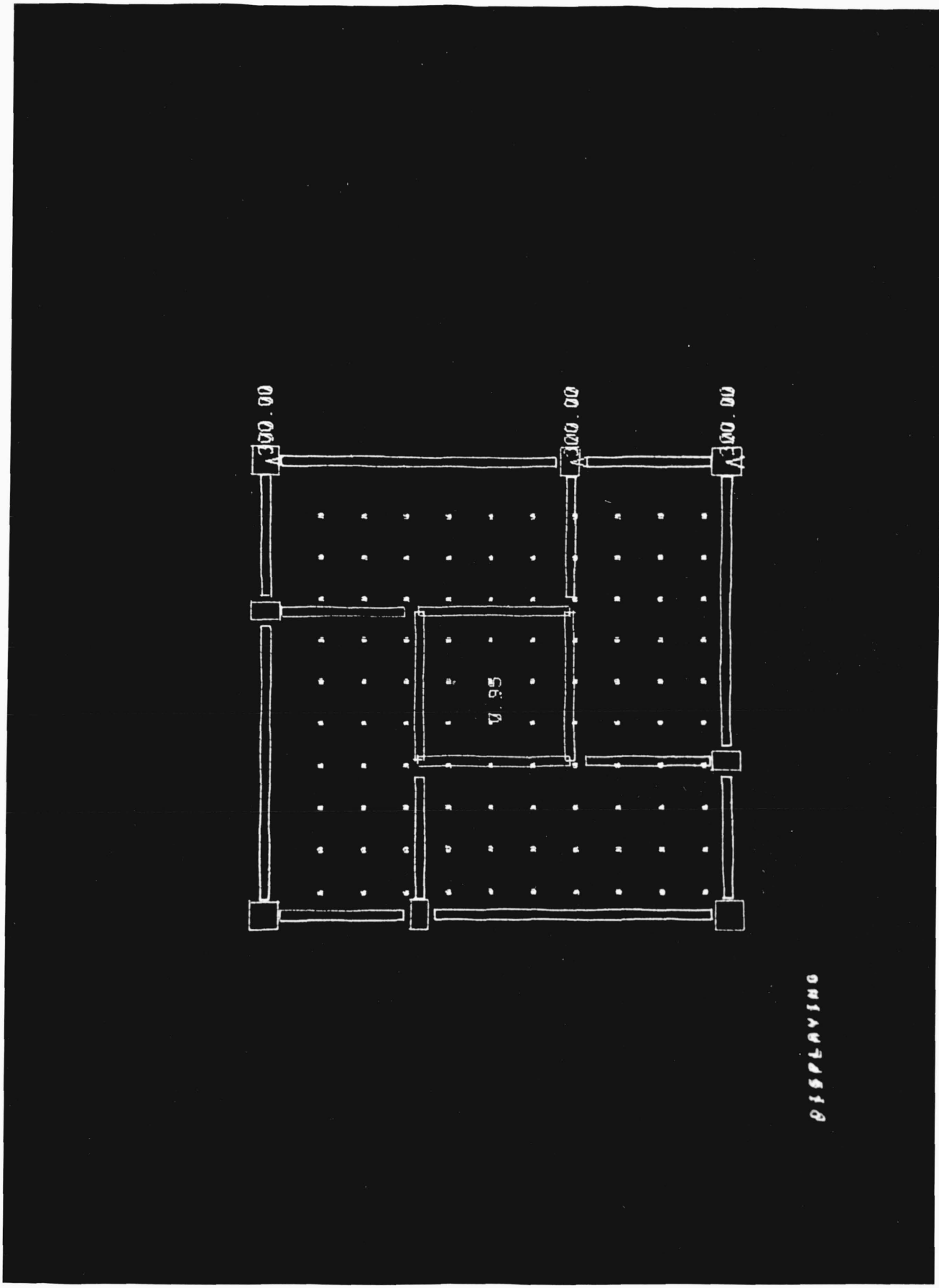


Figure 4.23 An area load of .95 KN/m^2 and three point loads of 300 KN.

```
                    01 TO CONTINUE

FLOOR DIRECTORY

FLOOR 01  1    METRES   012 NOD PNTS
FLOOR 02  7    METRES   012 NOD PNTS
FLOOR 03 13    METRES   012 NOD PNTS
FLOOR 04 19    METRES   012 NOD PNTS
FLOOR 05 25    METRES   012 NOD PNTS
FLOOR 06 31    METRES   012 NOD PNTS
FLOOR 07 37    METRES   012 NOD PNTS
FLOOR 08 43    METRES   012 NOD PNTS
```

Figure 4.24 The floor directory indicating the height of each floor and the number of nodal points on each floor.

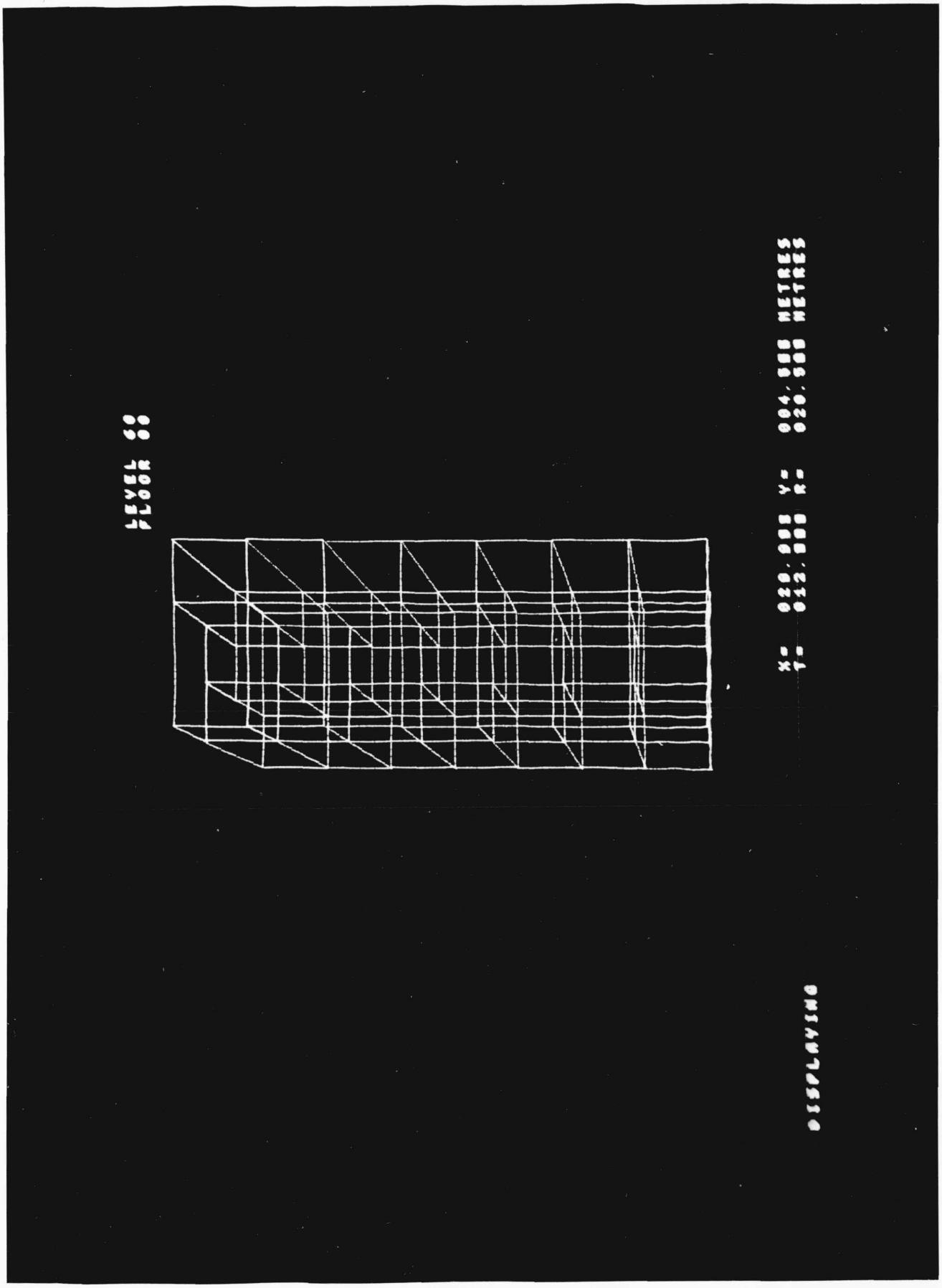


Figure 4.25 Lines joining the nodal points of the undisplaced structure.

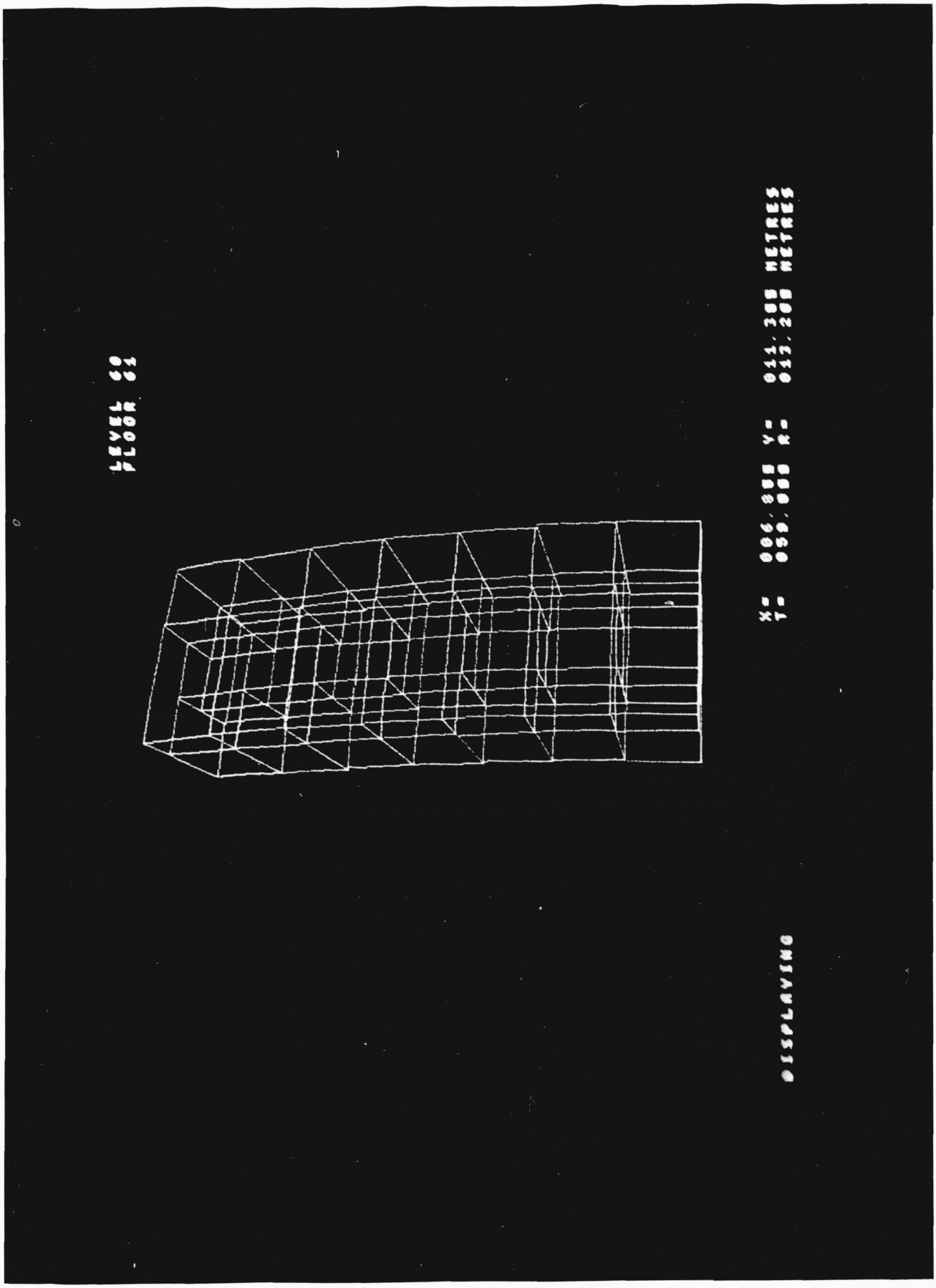


Figure 4.26 The displaced structure.

4.8 Assessment of STASYS

In successfully carrying through a finite element analysis there are three main points at which errors could occur or be generated due to programming:

Errors in data interpretation and manipulation

Incorrect stating of the element stiffness matrices

Inaccuracy in solving the set of simultaneous equations

Eliminating errors in data interpretation is achieved by displaying the data and allowing the operator to visually check it. Eliminating errors during the data collation stage has been achieved by listing the data at many stages of the process and manually checking that it is in the correct format and right magnitude, for a wide variety of data sets.

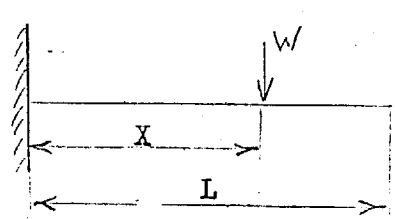
To establish the validity of the code which generates the element stiffness matrices two different approaches have been adopted:

1) All the terms of the element stiffness matrix have been hand calculated for a particular element and compared with the terms produced by the code.

2) Simple structures for which analytical solutions can be easily obtained have been idealised by finite elements. The results obtained by solving the problems on the computer for a variety of load cases compare to the analytical solutions to an accuracy of at least two significant figures and usually more.

For example in simple elastic theory the end deflection 'w' of a cantilever of length 'L' due to a point load at a distance 'X' from the fixed end is given by:

$$w = \frac{WX^3}{3EI} + \frac{WX^2}{2EI}(L - X)$$



The cantilever may be represented either by a series of plate or line elements. In both cases the computer analysis has produced equivalent results although the stiffness matrices involved have been different.

Accuracy of the method of solution is more difficult to establish. The Gaussian elimination procedure used in STASYS does not perform partial pivoting. Some texts written on the general solution of simultaneous equations suggest that such a procedure is necessary.[11,12] However, it is not reasonable to consider the equations which represent a physical structure as being completely general since they possess specific attributes. They are always symmetric and banded. Furthermore, the diagonal terms of the matrix of equations are always similar to or greater in magnitude than the other terms in the matrix. Livesley[13] states that a working precision of at least ten significant digits for all computer operations avoids any serious trouble due to ill-conditioning when using the finite element method of analysis on practical structures. In STASYS the assembly and solution of the matrix of equations is carried out in double precision on the PDP 11/40 i.e., floating point values have a range of from $.14 \times 10^{-38}$ to 1.7×10^{38} with a significance of sixteen decimal digits.

Although the time taken to perform the solution is considerable, it is still much quicker and more convenient to perform the solution on the minicomputer. The illustrated example in section 4.7 was set up, solved and photographed in a single two hour session. The time for solution was twenty-three minutes. By conventional means the preparation of the data cards alone would have taken us long. If one adds the time required for a typical 'turn round' on a batch system and further remembers that there is a strong likelyhood of data errors in the first run, one can appreciate the convenience of STASYS.

CHAPTER FIVE

Conclusions and future development

A graphics capability has been developed on a minicomputer system. By providing data input, display and handling facilities GCADS allows the operator to create,modify, store and plot many types of drawings and helps to eliminate most of the repetitious and tedious tasks involved in the production of drawings by conventional means. To increase the efficiency of production of particular types of drawing further modules may be added to GCADS.

These graphic and data handling facilities may be used and built upon to provide graphic aid in a wide variety of applications. Because the applications programs use the same graphic database it is possible, where these applications programs represent a series of stages in the design of the same project, to spread the cost of data input for those items of data that are common.

In particular programs have been added to enable fast generation of data for the finite element analysis of structures which can be conveniently viewed as a series of layers or floors. The set of equations which describes the stiffness of the structure under investigation may be assembled and solved on the same minicomputer.

The work has clearly demonstrated that a minicomputer system has the power to set up and solve problems of a complex nature. Because the system is one order of magnitude cheaper than typical commercial systems it is economic to use it to solve these problems although the actual time required for solution is much longer.

The low cost enables companies that were not previously able to purchase their own computer facility to do so. The low running cost allows engineers and designers to use powerful computer and graphic aids in solving their everyday problems.

In the field of structural analysis there are a huge number of programs already written to deal with a wide range of problems. At present, they are underutilised because of data preparation difficulties, or because in order to use them it is necessary to wade through a vast manual, or because the computing facilities available are not convenient. Undoubtably, the transfer of these programs onto a system such as GCADS would do much to encourage their use and development.

The maximum size of problem that can be solved on a minicomputer system depends on two factors. The first is the length of time that one considers is reasonable or economic and the second is the quantity of backing store in the system. In STASYS the time that a particular problem takes to solve depends on the amount of main memory available and the speed of data transfer between main memory and the backing storage device.

Cheaper memory, and, faster and larger disk units for the same price, can now be purchased.

If at the same time, proper advantage is taken of the technique of substructuring the size of problem that could be solved on a minicomputer would be many, at least ten, times larger than is presently possible.

In the technique of substructuring, the structure is partitioned, usually along physical boundaries, into smaller units - substructures. If the stiffness properties of each substructure are determined, the substructure can be treated as complex structural elements, and the matrix displacement method of structural analysis can be formulated for the partitioned structure. Once the displacements on substructure boundaries have been found, each substructure can then be analysed separately under known substructure-boundary displacements. The two advantages of substructuring are:-

a) The size of matrix that has to be handled and solved at any one time is greatly reduced.

b) If the structure is carefully partitioned, groups of the sub-structures will be the same. In this case, a reduction in data storage and generation is possible and the stiffness properties of the substructures need only be derived for one of each group.

Although substructuring involves the solution of a larger number of sets of equations, the sets are smaller, and because of this fact and also the economies mentioned in b), the actual time for solution of a problem could be reduced. In such a case, substructuring would have an additional advantage over the analysis of structures as single entities, even where this is possible.

Using these ideas it is possible to work towards a firm proposal for a new or upgraded minicomputer system. This would be able to carry out draughting operations and really useful sized stress analyses. As an extra bonus a COBOL compiler has just been announced by DEC which will allow commercial administrative tasks to be run on the same system.

It is not possible to justify further expenditure if only one person has access to the system at a time. A time sharing system, either RSX 11D or RSX 11M, must be used. The RSX 11D operating system requires 20K words of memory, plus an additional kiloword for each extra peripheral. Postulating that a four terminal system would keep one processor constantly busy, and allowing a 10K core segment for each terminal, 64K words of memory would be the memory required. The COBOL compiler which requires at least 48K to run effectively would be run with the system dedicated to it.

For the stress analysis programs the more core available the faster solutions can be obtained, and again dedicated running would be preferable for large jobs but not essential for development. Using the substructuring technique a maximum of six blocks of matrix need be in core at once. With 64k words a matrix block size of 48 could be sustained. Moving from a block size of 8 to 18 produced an increase in speed of 2.5 times. Moving from a block size of 18 to 48 is expected to result in a similar gain.

A hypothetical structure which is idealised using 1000 nodes and which can be divided into 10 substructures of 5 different types would require the invertion of 6 different sets of 600 equations. The expected solution time would be 72 minutes for all sets using the present disk units.

The temporary backing storage required can be calculated as being approximately 3.6 megawords. At least a further 1 megaword would be needed for program and other data storage. The minimum requirement is therefore 4.6 megawords.

To make best use of the draughting facility a library of standard items would be created. In time this would expand to several megawords but could be sectioned and archived on magnetic tape.

For design and analysis applications a data base containing information on properties of materials, tables of design parameters, data on previous jobs completed and data on the programs available for use would be built. The main part of this would need to be on disk for interrogation.

There are two disk systems that can be considered, each consists of a controller capable of handling up to eight transports. The smaller of these offers 5 megawords/transport, the larger 20 megawords/transport. The cost of the controller and the first disk transport for the smaller system is approximately £6000 and for the larger £8000. Subsequent transports would cost about £3000 and £4000 respectively. For a commercial environment the larger system is deemed necessary but for a university the smaller system would be satisfactory for at least two years.

The composition of the four terminals depends on the work load and the type of work. A digitising table is the best device for extracting data from existing drawings, and one digitiser is essential. Editing of data can be performed more easily by using a data pad/storage screen combination or a refresh tube and light pen.

If a large amount of program development is likely then two of the terminals would only have to be Visual Display Units with keyboards.

If the smaller disk system was chosen with two transports, the cost of upgrading the Imperial College system to the outlined specification would be about £20,000. The cost per terminal would then be £15,000.

Assuming depreciation over a four year period the cost per year for each terminal is similar to the cost of employing a single member of staff.

There can be little doubt that once the software for these systems has been fully developed they will create large savings in the time required to design and schedule projects and large savings in the overall design cost.

References

1. 'A practical guide to minicomputer applications', edited by F. F. Coury IEEE Press, New York, 1972.

2. 'CAMAC - a modern instrumentation system for data handling', Euratom, 1964, EUR4100E.

3. 'DOS Monitor Handbook'

4. Knuth, D., 'The art of computer programming', Vol. 1.,'Fundamental algorithms', Addison-Wesley.

5. Olds, W., 'Interactive modification of aircraft wiring diagrams', Presented at Conference on 'Data structures', Little Hall, Cambridge, 4-5th Sept., 1973.

6. Zienciewicz, O. C., 'The finite element method in Engineering science', McGraw Hill, 1971.

7. Przemieniecki, J. S., 'Theory of matrix structural analysis', McGraw Hill, 1968.

8. Timoshenko, S. P., & Goodier, J. N., 'Theory of elasticity', McGraw Hill, 1970.

9. Cantin, G., 'An equation solver of very large capacity', Int. J. for Num. Meth. in Engng., Vol. 3, 379-388, 1971.

10. Grindley, R. E., 'The development and application of a low cost computer aided design system in Mechanical engineering', Ph.D. thesis University of London, 1973.

11. Wilkinson, J. H., 'Rounding errors in algebraic processes', National Physical Lab., HMSO, 1963.

12. Albasiny, E. L., 'Error in digital solution of linear problems', paper in 'Error in digital computation', Vol. 1., Wiley, 1965.

13. Livesley, R. K., 'Matrix methods of structural analysis', Pergammon Press, 1964.

14. Besant, C. B., et al, 'The use of CADMAC systems in general draughting', Presented at Conference on 'Computer aided draughting systems', St. John's College, Cambridge, 2-4th April 1973.

15. McClintock, P. M., et al, 'STASYS - STructural Analysis SYStem an application in CAD', Presented at Conference on Computers in Engineering and Building design (CAD 74), Imperial College, London, 25-27th Sept. 1974.

Appendix A-1

GCADS use of Common areas

a) COMMON/BUFFER/

IB(40)	
XB(40)	Buffer for workspace data
YB(40)	

b) COMMON/FILHND/

IONDIR	Used to indicate which filing operation is to be executed: 1 Workspace to file 2 File to workspace 3 Display file 4 Use file as macro
NR1	Points to the next disk record of the workspace
NR2,NR3,NR4	Used as record pointers for other random access files
IDP	Points to the next I,X,Y triplet in the workspace buffer to be processed.
IPEN	Indicates line broken (1) or unbroken (2).
XT,YT	Trailing origin
XTR,YTR	Current position of the digitising pencil.

c) COMMON/GENRL/

GEN(10)	Reserved for application program parameters

d) COMMON/MACRO/

MAC(9)	Macro processor status flags
XFLAG(4),YFLAG(4)	Flag to display condition of the macro status flags.
COST,SINT	Angle of rotation to be applied to macros.

e) COMMON/MENU/

MNUM	Last menu square number used

MTYPE	Indicates which menu section 1 Commands 2 Levels 3 Files 4 Symbols
NC	Indicates control mode 1 Control 15^{o} 2 Control 90^{o} 3 No control
NCORD	1 Coordinates displayed indicate position from trailing origin. 0 Coordinates displayed indicate distance from absolute origin.
NCON	Indicates which control mode was selected from the menu: 1 Control 15^{o} 2 Control 90^{o}

f) COMMON/MESSAGE/

ANSIZ	Alphanumeric text display size
LEVEL	Current input level
MESS(5,5)	Five messages to be displayed by the background loop used by the system
MESS(5,5)	Five messages to be used by the application programs

g) COMMON/PARAM/

FIPX,FIPY	Absolute origin of x,y coordinates
CANG,SANG	Cosine and sin of input skew angle
FISCL	Input scale
GR	Grid factor
XWNDO,YWNDO	Window origin
OPSCL	Output scale

h) COMMON/PLOTER/

IPLPAR(4)	Plotting control parameters for speed,acceleration and accuracy
ICPEN(5)	Information on pen unit status
ICLINE(5)	Data on line type

i) COMMON/SUBOV/

ISUB	Points to overlay segment

j) COMMON/SYMBOL/

ISFLAG	0 Line mode
	1 Symbol mode, new symbol not allowed
	2 Symbol mode, new symbol may be selected
NSYM	Symbol mode number
JOKE	Symbol point counter
XCP,YCP	Position of pencil
XSYM(15),YSYM(15)	Storage for symbol points.

k) COMMON/USER/

BUF1(128),BUF2(128) Buffer space for use by application programs.

Appendix A-2

GCADS OVERLAYS

OVERLAY 1 SETUP0

FUNCTION: Sets up table origin,data input origin, and skew control. Called either by menu command or on first entry to the system. Exits to digitising and control mode or to SETUP2 3.

OVERLAYS CALLED

SETUP2 3

SUBROUTINES CALLED

Camac	Curcon	Erscn
Filsrt	Getint	Ovlink
Ovretn	Plmsg	Plots
PmsgO	Setflg	Stord

OVERLAY 2 SETSYM

FUNCTION: Sets up messages for symbols - to be displayed in background m mode. Called by a symbol overlay exits to a symbol overlay.

OVERLAYS CALLED

None

SUBROUTINES CALLED

Ovretn	Setmes

OVERLAY 3 SETUP2

FUNCTION: Sets input and output scales, and grid factor. Called from SETUP0 1 or from menu command, exits to digitising and control mode.

OVERLAYS CALLED

None

SUBROUTINES CALLED

Ovlink Ovretn

OVERLAY 4 DIGOV

FUNCTION: Digitising and control mode. Controls data input and services all menu commands. The centre of the system. Initially entered from SETUP2 3. It regains control by default.

OVERLAYS CALLED

All overlays called by menu command.

SUBROUTINES CALLED

Backgd	Camac	Clrlev
Erscn	Levset	Menmap
Mensel	Ovretn	Ovlink
Plotsc	Setlev	Stack
Stord	Symap	Symgo

OVERLAY 5 ERROV

FUNCTION: Issues error messages on the DECwriter. Called by many overlays. Normally exits to digitising and control mode.

OVERLAYS CALLED

None

SUBROUTINES CALLED

Ovretn

OVERLAY 6 DISALL

FUNCTION: Displays the workspace and other work files in the same format. Calls the symbol display segments. Called by menu command or by other overlays.

OVERLAYS CALLED

Any symbol overlay.

SUBROUTINES CALLED

Camac	Cvt	Dbeam*
Dcol*	Discrs*	Dissym
Dslab*	Dwall*	Erscn
Levtst	Plmsg	PmsgO
Ovretn	Plotsc	Raread
Rawrit	Restcm	Screen
Stack	Stor7	Storcm

OVERLAY 7 OVFIL

FUNCTION: Writes or reads contents of a workfile to or from the semi-permanent files (CADMAC.MS1). Also displays files in abbreviated form. i.e. only lines displayed no symbols. Called by menu command. Exits to digitising and control mode.

OVERLAYS CALLED

MACRO2 16

SUBROUTINES CALLED

Camac	Closms	Deltms
Levtst	Openms	Ovlink
Ovretn	Plmsg	Plot
PmsgO	Raread	Rawrit
Stack	Stord	Readms
Screen	Writms	

OVERLAY 8 GRWNDO

FUNCTION: Sets window parameters. Called by menu command.

OVERLAYS CALLED

None

SUBROUTINES CALLED

Curcon	Getint	Plmsg
Pmsgo	Ovretn	

OVERLAY 9 EDITOV

FUNCTION: Line editor. Called by menu command. Exits to DISALL.

OVERLAYS CALLED

DISALL 6

SUBROUTINES CALLED

Camac	Curccn	Getint
Ovlink	Plmsg	Plotw1
PmsgO	Raread	Rawrit
Screen	Sdist	Windrw

OVERLAY 10 PEROP

FUNCTION: Peripheral input/output. Allows data to be transferred to and from the workspace from and to any device. Called by menu command. Exits to digitising and control mode.

OVERLAYS CALLED

None

SUBROUTINES CALLED

Asgn	Erscn	Getint
Ovretn	Plmsg	PmsgO
Raread	Rawrit	Stack
Stord		

OVERLAY 11 FINDOV

FUNCTION: Finds a point in the workspace. Called from DIGOV or from a symbol overlay. Returns to calling overlay.

OVERLAYS CALLED

None

SUBROUTINES CALLED

Camac	Findy	Ovretn
Plotsc	Stack	Stord
SymgO		

OVERLAY 12 DEBUG

FUNCTION: Writes out contents of random access files in a choice of formats. Used for debugging. Called by menu command.

OVERLAYS CALLED

None

SUBROUTINES CALLED

Raread	Ovretn

OVERLAY 13 BACKOV

FUNCTION: Overlay to return data points to application overlays. Also used by the Macro processor MACRO3 and the macro editor MACRO4 .

OVERLAYS CALLED

GRWNDO 8

SUBROUTINES CALLED

Backgd	Find	Ovretn
Stack	Storcm	Restcm

OVERLAY 15 MOVPT

FUNCTION: Point editor. Allows points in the workspace to be moved. Called by menu command. Exits to DISALL.

OVERLAYS CALLED

DISALL 6

SUBROUTINES CALLED

Backgd	Curcon	Find
Getint	Ovlink	Plmsg
Plotw1	PmsgO	Raread
Rawrit	Stack	

OVERLAY 16 MACRO2

FUNCTION: First of two macro processing overlays. Reads data from specified mass storage to workfile CADMAC.RA2 rejecting data on passive levels. Also gets scale from user if appropriate macro status flag is set. Called by menu command. Exits to MACRO3 17.

OVERLAYS CALLED

DISALL 6

MACRO3 17

SUBROUTINES CALLED

Levtst	Openms	Ovretn
Plmsg	PmsgO	Rawrit
Readms	Stack	Storcm

OVERLAY 17 MACRO3

FUNCTION: Rotates, scales and translates the data stored in workfile 2 by MACRO2. Allows the user to place the macro in the workspace. Called by menu command and by pencil button 3. Exits to digitising and control mode.

OVERLAYS CALLED

BACKOV 13

DISALL 6

SUBROUTINES CALLED

Erscn	Getint	Ovretn
Plmsg	PmsgO	Raread
Rawrit	Restcm	Stack

OVERLAY 18 CONDIG

FUNCTION: Allows the user to digitise a series of points merely by moving the pencil. Points either entered at time intervals or according to the distance moved. Called by menu command. Exits to digitising and control mode.

OVERLAYS CALLED

None

SUBROUTINES CALLED

Curcon	Getint	Ovretn
Plmsg	PmsgO	Xback

OVERLAY 19 PAPFIT

FUNCTION: Fits data in workspace to particular paper size for plotting.

OVERLAYS CALLED

DISALL 6

SUBROUTINES CALLED

Erscn	Levtst	Ovretn
Raread	Rawrit	Stack

OVERLAY 20 MACRO

FUNCTION: Sets macro processor status flags. Called by several menu commands. Exits to digitising and control mode.

OVERLAYS CALLED

None

SUBROUTINES CALLED

Erscn	Camac	Getint
Ovretn	Plmsg	PmsgO
Plotw1	Setflg	

OVERLAY 21 SYM002

FUNCTION: Handles the following symbols: Arcs,circles,fillets rectangles.

OVERLAYS CALLED

DIGOV 4

SUBROUTINES CALLED

Boxsym	Circle	Disarc
Midarc	Ovretn	Plotsc
Stack	Setmes	Stord

OVERLAY 22 SYM003

FUNCTION: Dimensioning symbol.

OVERLAYS CALLED

DIGOV 4

SUBROUTINES CALLED

Boxsym	Dimen	Ovretn
Stack	Setmes	Stord

OVERLAY 23 SYM004

FUNCTION: Text or alphanumeric symbol overlay.

OVERLAYS CALLED

DIGOV 4

SUBROUTINES CALLED

Boxsym	Ovretn	Stack
Setmes	Symbl	Stord
Tstext		

OVERLAY 24 TIDYOV

FUNCTION: Sorts garbage out when data is transferred from workspace to semi permanent mass storage file. Called from OVFIL.

OVERLAYS CALLED

None

SUBROUTINES CALLED

Ovretn	Raread	Rawrit
Stord.		

OVERLAY 29 MACRO4

FUNCTION: Macro and symbol editor. Operator can rotate, translate and delete nests of macros or symbols. Called by menu command.

OVERLAYS CALLED

BACKOV 13

MACRO 20

All symbol overlays

GRWNDO 8

SUBROUTINES CALLED

Curcon	Disbox	Dissym
Getint	Finds	Ovretn
Plmsg	Pmsg0	Raread
Rawrit	Restcm	Stack
Storcm		

Appendix A-3

FORTRAN subroutines in the GCADS library

Length	Name	Subroutines called	Function
1463	BACKGD	CAMAC CAMRTB CORDIS DISPCS DSKEW GRID GETINT PLMSG PMSGO SCREEN TFLOOR TLEV MENAN	Main background loop.
1072	BOXSYM	STORD	Calculates bounding rectangles for symbols and stores them.
634	CIRCLE	PLOTSC	Draws arcs and circles.
316	CDISP	CAMAC DLOT MASK	Displays coordinates on the coordinate display unit.
480	CORDIS	CDISP PLACE PMSGO	Displays coordinates on screen.
233	CAMRTB	CAMAC DROT	Read the table coordinates.
142	CRPLOT	PLOTSC	Generates the vectors that make up characters. Sends them to the display.
195	CURCON	CAMRTB CORDIS DISPCS DSKEW SCREEN	Displays the cursor on the screen.

Length	Name	Subroutines called	Function
219	CLOSMS	IREAD RAREAD RAWRIT WORTRAN	Closes mass storage file.
59	DSKEW	-	Deskews x,y coordinates.
943	DIMEN	NUMBER PLOTSC	Automatically dimensions between two points.
175	DISBOX	CAMAC PLOTW1 SCREEN	Displays box round macro on the screen.
168	DISPCS	CURSND DSPNST DSPSTR	Displays a cross hair on the screen.
338	DISARC	CIRCLE	Displays arcs.
95	DELTMS	BITCLR IREAD IWRIT	Deletes a mass storage file.
57	ERSCN	-	Erases the screen.
129	FILSRT	INITF SETFIL	Reads and inits random access files on the disk unit, which are to be used by GCADS.
506	FIND	LEVTST RAREAD RAWRIT PLMSG PMSGO	Finds a point in the workspace. Called from MOVTP.

Length	Name	Subroutines called	Function
727	FINDS	CURCON DISPCS LEVTST PLMSG PMSGO RAREAD SCREEN	Finds a point in the workspace and returns its position within the workspace. Called by MACROFOR.
565	FINDY	as FIND	As FIND but checks bounding rectangles. Called from FINDOV.
89	FLENTH	-	Function to calculate the length of a line.
164	GRID	-	Rounds coordinates to a multiple of the grid factor.
164	GETINT	CAMAC CVT MASK	Looks for pencil button interrupt.
529	LOOKMS	CLOSMS RAREAD RAWRIT SUBMS	Searches mass storage file for free space.
463	MACMES	CURCON GETINT PLMSG PMSGO	Displays messages for macro editor.
203	MENAN	-	Determines menu section and menu square number of a point.

Length	Name	Subroutines Called	Function
301	MIDARC	-	Calculates spare point in an arc.
477	NUMBER	SYML	Displays number on the screen.
320	OPENMS	RAREAD SUBMS	Opens a mass storage file.
46	PLOTSC	PLOT SCREEN	To call plot and screen.
46	PLOTWS	PLOTW1 SCREEN	As PLOTSC but screen only PLOT.
164	PLOTS	CAMAC PLOTW1	Initialises display system.
1073	PLOTW1	CAMAC CAMRTB PLOTD WINDRW	Plots on the screen.
185	READMS	PLMSG PMSGO RAREAD SUBMS	Reads data from mass storage file.
87	SETFLG	-	Sets up flag to indicate macro processor status.
131	SYMBL	CHAR	Displays symbols on the screen, or for plotting.
101	SDIST	-	Calculates distance from midpoint of line to another point.

Length	Name	Subroutines called	Function
183	SELECT	GETINT FIND CURCON PLMSG PMSGO	Enables selection of start and end points in macros. Called from MACRO3.
227	SETMES	-	Sets symbols messages.
264	STORD	RAWRIT	Stores I,X,Y triplet in workspace. Calculates bounding rectangles.
65	SCREEN	-	Converts coordinates to screen size.
249	STOR7	PACK RAWRIT	Stores data in format for kingmatic plot in random access file 7.
1130	SPLINE	-	Calculates coefficients of cubic spline for curve fitting.
320	WRITMS	LOOKMS RAWRIT SUBMS	Writes to mass storage file.
439	WINDWB	-	Scissoring routine.
441	WINDOW	-	As above but for plotter.
251	XBACK	CAMAC CURCON GETINT PLOTSC STORD	Background loop for continuous digitising.

Assembly Language Subroutines in the GCADS Library

Length*	Name	Subroutines Called	Function
14	ASGN	-	Requests that device be assigned as logical device 3.
226	CAMAC	-	CAMAC interface handler.
734	CHAR	-	Defines and generates characters.
212	CURSND	-	Displays cursor.
24	CVT	-	Vit conversion routine for GETINT.
14	DEPACK	-	Opposite of PACK.
243	DISSYM	-	Stacks display overlay segment, for symbols.
20	DROT	-	Bit conversion routine for CORDIS.
20	DLOT	-	Bit conversion routine for CAMRTB.
112	DSPMDE	-	Entry points DSPNST and DSPSTR to set screen into non-store and store mode.

*Length in Octal bytes.

Length	Name	Subroutines called	Function
226	LEVEL	LEVCLR LEVSET LEVTST SETLEV CLRLEV	Handles level operations.
12	MASK	-	Routine to do a logical mask.
164	MENMAP	-	Allows menu commands squares to be re-located.
200	MENSEL	-	Selects overlays after menu command.
1536	OVLAY	OVLINK OVINIT OVRETN STACK CLRSTK	Overlay handler.
14	PACK	-	Packs second integer into second word of first integer.
166	PLACE	PLMSG	Actually displays coordinates on screen.
152	PLMSG	READY	Plots messages on the screen.
154	PMSGO	READY	Sets position and size at which messages to be plotted on screen.
774	PLOTD	-	Subroutine to drive flat bed plotter.
704	RACSES	RASET RALOOK RAREAD RAWRIT	High speed random access file handler.

Length	Name	Subroutines called	Function
1110	READFR	-	Free format input routine.
120	RLOOK	-	Looks up length of mass storage file.
104	STRCON	RESTCM RAREAD RAWRIT	Stores and re-stores common user area to random access file zero.
316	SUBMS	BITTEST BITCLR BITSET IREAD WORTRN ZERO	Subroutines for mass storage handler.
26	SYMAP	-	Symbol mapping routine.
261	SYMGO	CLRSTK STACK OVRETN	Directs control to symbol segments.
136	TANGLE	-	-
56	TLEV	-	Converts integer number to ASCII.
76	TSTEXT	-	Sorts text strings.

Appendix B - 1

STASYS use of Common areas

a) COMMON/GENRL/

DUMMY(6)	Not used
IFLN	Floor number
IBASE	Number of nodal points up to current floor
IELEM	Number of current element

b) COMMON/MESAGE/

DUMMY(32)	Used for GCADS messages
MM	Number of blocks per row of stiffness matrix
NN	Number of blocks per column of stiffness matrix
NTRKC	Number of column vectors per row of stiffness matrix
NS	Size of matrix blocks
NDF	Number of degrees of freedom at each node
IELT	Current element type
WPN	Self weight at each node of the current element
LDC	Number of load cases in solution
LDNUM	Current load case number
HT	Next floor level minus current floor level
XOFF	
YOFF	Three dimensional centre of view
ZOFF	
XO	
YO	Origin of picture plane
OVER	
UP	Three dimensional viewpoint in polar coordinates
R	
PP	Distance of picture plane from viewpoint
NP	Type of projection plane 1 Flat 2 Spherical

Appendix B-2

STASYS OVERLAYS

OVERLAY 31. GRIDIN

FUNCTION: Grid points or grid arrays are entered in the workspace and displayed on the screen. Called from menu command.

OVERLAYS CALLED

BACKOV 13

SUBROUTINES CALLED

BUTNUT	Discrs	Getint
Ovretn	Plmsg	PmsgØ
Pltcrs	Screen	Stack
Stord		

OVERLAY 32. CBSIN

FUNCTION: Beam, column, slab and wall elements are entered in the workspace. Space is left for dimensions and material number. Called from menu command. Exits to digitising and control mode.

OVERLAYS CALLED

BACKOV 13

DISALL 6

SUBROUTINES CALLED

Dissym	Getint	Ovretn
Plmsg	PmsgØ	Restcm
Stack	Storcm	Stord

OVERLAY 33. CBSIZ

FUNCTION: Dimensions are entered into element data beads that are in the workspace. The elements are displayed with their new dimensions. Called from menu command. Exits to digitising and control mode.

OVERLAYS CALLED

GRWNDO 8

SUBROUTINES CALLED

Advanc	Butnut	Gurcon
Dbeam	Dcol	Dslab
Dwall	Findsy	Getint
Ovretn	Plmsg	PmsgØ
Raread	Rawrit	Restcm
Stack	Storcm	

OVERLAY 34. MATIN

FUNCTION: A material number is entered into the element data beads that are in the workspace. New sets of material properties may be written to records 20 & 21 of CADMAC.RAO/. Sets are listed.

OVERLAYS CALLED

SUBROUTINES CALLED

Advanc	Butnut	Curcon
Dissym	Findsy	Getint
Ovretn	Plmsg	PmsgØ
Raread	Rawrit	Restcm
Stack	Storcm	

OVERLAY 35. LODIN

FUNCTION: Load symbols entered into the workspace and displayed on the screen. Called from menu command. Exits to digitising and control mode.

OVERLAYS CALLED

BACKOV 13

LODSYM 45

SUBROUTINES CALLED

Butnut	Dissym	Ovretn
Plmsg	PmsgØ	Recsrt
Restcm	Getint	Stack
Storcm	Stord	

OVERLAY 36 BCIN

FUNCTION: Boundary condition symbols are entered into the workspace and displayed on the screen. Called from menu command. Exits to digitising and control mode.

OVERLAYS CALLED

BACKOV 13

BCSYM 46

SUBROUTINES CALLED

Dissym	Getint	Ovretn
Plmsg	PmsgØ	Stack
Stord		

OVERLAY 37. FDREC

FUNCTION: Handles all entries and/or deletions to floor directory. Called from menu command. Exits to digitising and control mode.

OVERLAYS CALLED

None

SUBROUTINES CALLED

Butnut	Erscn	Getint
Ovretn	Plmsg	PmsgØ
Raread	Rawrit	Tfloor
Tnum		

OVERLAY 38. LODEX

FUNCTION: Analyses load symbols in workspace and creates a load file for the floor. Called by menu command. Exits to FILOV9 overlay 50.

OVERLAYS CALLED

FILOV9 50

SUBROUTINES CALLED

Addlod	Isita	Isitl
Levsqu	Levtst	Ovretn
Raread	Rawrit	Stack

OVERLAY 44. SORTOV

FUNCTION: Sorts a file of x,y points into order according to priority set by user. Called by NODEX.

NO OVERLAYS CALLED

SUBROUTINES CALLED

Ovretn Raread Rawrit

Tswap

OVERLAY 45. LODSYM

FUNCTION: Displays load symbols. Called from DISALL via Dissym when displaying workspace also from LODIN at time of entry.

NO OVERLAYS CALLED

SUBROUTINES CALLED

Number Symbl Ovretn

OVERLAY 46. BCSYM

FUNCTION: Displays boundary condition symbols in workspace. Called from DISALL via Dissym when displaying workspace and from BCIN at time of entry.

NO OVERLAYS CALLED

SUBROUTINES CALLED

Symbl Ovretn

OVERLAY 48. PNSHOW

FUNCTION: Puts point numbers as text symbols into workspace. Called from NODEX.

NO OVERLAYS CALLED

SUBROUTINES CALLED

Ovretn Raread Storno

Tnum

OVERLAY 49. FILOV5

FUNCTION: Files to and from Random Access Files to Mass Storage File CADMAC.RA5 (Boundary condition file.

NO OVERLAYS CALLED

SUBROUTINES CALLED

Camac	Closms	Deltms
Openms	Ovretn	Ovlink
Plmsg	PmsgØ	Raread
Rawrit	Readms	Stack
Stord	Writms	

OVERLAY 50. FILOV9

FUNCTION: Files to and from Random Access Files to Mass Storage File CADMAC.RA9 (Load file).

NO OVERLAYS CALLED

SUBROUTINES CALLED

Camac	Closms	Deltms
Openms	Ovretn	Ovlink
Plmsg	PmsgØ	Raread
Rawrit	Readms	Stack
Stord	Writms	

OVERLAY 51. NODEX

FUNCTION: Extracts nodal points from workspace and puts them into CADMAC.RA4. Duplicate points are eliminated. Called from menu command. Exits through SORTOV, PNSHOW and DISALL to digitising and control mode.

OVERLAYS CALLED

SORTOV	44
PNSHOW	48
DISALL	6

SUBROUTINES CALLED

Levtst	Ovretn	Plmsg
Pmsgø	Raread	Rawrit
Stack		

OVERLAY 52 BCEX

FUNCTION: Extracts boundary conditions from workspace. Boundary conditions at the same point are added together in such a way that they can be decoded later. The extract is filed in CADMAC.RA4 and then into CADMAC.RA5 (Boundary condition File). Called from menu command.

OVERLAYS CALLED

FILOV5

SUBROUTINES CALLED

Levtst	Ovretn	Raread
Rawrit		

OVERLAY 53. CRMS

FUNCTION: Initialises mass storage file defined by user, currently CADMAC.MS1, .RA3, .RA5 and .RA7. Called from menu command.

NO OVERLAYS CALLED

SUBROUTINES CALLED

Bitset	Iwrit	Ralook
Raset	Rawrit	Wortrn
Zero		

OVERLAY 54. FILOV3

FUNCTION: Files to and from Random Access File to Mass Storage File CADMAC.RA3 (Nodal coordinate file).

NO OVERLAYS CALLED

SUBROUTINES CALLED

Camac	Closms	Deltms
Openms	Ovretn	Ovlink
Plmsg	Pmsgø	Raread
Rawrit	Readms	Stack
Stord	Writms	

OVERLAY 55. ELPRNT

FUNCTION: Lists elements from FLOOR FILES (CADMAC.MS1) to device 3. Called from menu command.

OVERLAYS CALLED

FILOV3 54

SUBROUTINES CALLED

Levtst	Ovretn	Raread
Rawrit	Stack	

OVERLAY 56 NDPRNT

FUNCTION: Out puts nodal points from Nodal Coordinate File (CADMAC.RA3) to device three. Called from menu command.

OVERLAYS CALLED

FILOV3 54

SUBROUTINES CALLED

Ovretn	Raread	Rawrit
Stack		

OVERLAY 61. P3DAT

FUNCTION: Prepares three dimensional data for view of structural frame before and after displacements. Working only from nodal points to maintain clarity.

OVERLAYS CALLED

FILOV3 54

SUBROUTINES CALLED

Butnut	Finlin	Getint
Ovretn	Plmsg	PmsgØ
Raread	Rawrit	Restcm
Stack		

OVERLAY 62. DSTR1

FUNCTION: Allows user to select element in workspace for which stresses will be calculated and printed by DSTR2. Called by menu command. Exits to digitising and control mode.

OVERLAYS CALLED

FILOV3 54

DSTR2 66

SUBROUTINES CALLED

Advanc	Curcon	Dissym
Findsy	Getint	Ovretn
Plmsg	PmsgØ	Raread
Rawrit	Restcm	

OVERLAY 63. DISP3

FUNCTION: Displays three dimensional data set up by P3DAT 61. Called by menu command. Exits to digitising and control mode.

OVERLAYS CALLED

DISALL 6

SUBROUTINES CALLED

Butnut	Getint	Getp
Ovretn	Plmsg	PmsgØ
Raread	Stack	Stord

OVERLAY 64. ENTPAR

FUNCTION: Allows the operator to reset the solution parameters. Useful after a system failure. Called by menu command. Exits to digitising and control mode.

OVERLAYS CALLED

None

SUBROUTINES CALLED

Butnut	Zero

OVERLAY 65. LISTND

FUNCTION: Lists nodal deflections. Called by menu command. Exits to digitising and control mode.

OVERLAYS CALLED

None

SUBROUTINES CALLED

Butnut Ndisp

OVERLAY 66 DSTR2

FUNCTION: Calculates stresses in elements from global displacements. Called by DSTR1. Exits to DSTR1.

OVERLAYS CALLED

None

SUBROUTINES CALLED

Bstres	Lstres	Ndisp
Nodlk	Pstres	Stack

OVERLAY 81. SOLCON.

FUNCTION: Controls assembly of stiffness matrices. Reads through each floor with graphic data for elements. Calls STELM 82 to form elements stiffness matrices and DUSEMB 83 to assemble them. Called by menu command. Exits to LDSEMB 86.

OVERLAYS CALLED

FILOV3	54
FILOV5	49
OVFIL	7
STELM	82
LDSEM	86

SUBROUTINES CALLED

Levtst	Ovretn	Ralook
Raread	Rawrit	Restcm

OVERLAY 82. STELM

FUNCTIONS: Forms element stiffness matrices in terms of local coordinates and transforms them to global coordinates. Exits to DUSEMB 83.

OVERLAYS CALLED

DUSEMB	83

SUBROUTINES CALLED

Bark	Ovretn	Plak
Raread	Rawrit	Stack
Swap		

OVERLAY 83. DUSEMB

FUNCTION: Puts boundary conditions into element stiffness matrices. Assembles element stiffness matrices to form global stiffness matrix. Also assembles loads due to self weight called by STELM 82. Exits to SOLCON 81.

OVERLAYS CALLED

SOLCON 81

SUBROUTINES CALLED

Bclk	Inrec	Nodlk
Ovretn	Raread	Rodisk
Wrdisk		

OVERLAY 86. LDSEMB

FUNCTION: Assembles load vectors. Called by SOLCON 81 or by menu command. Exits to digitising and control mode.

OVERLAYS CALLED

FILOV3 54

FILOV9 50

SUBROUTINES CALLED

Asload	Getint	Ovretn
Plmsg	PmsgØ	Raread
Rawrit	Restcm	

OVERLAY 87. SOLVE

FUNCTION: Solves for displacements. Uses a Gaussian elimination method of solution operating on blocks of the matrix. Called by menu command. Exits to digitising and control mode.

OVERLAYS CALLED

None

SUBROUTINES CALLED

Exdisk	Mult	Ovretn
Syminv	Rddisk	Wrdisk

OVERLAY 88. ZERLD

FUNCTION: Zeros LOAD VECTORS where boundary conditions demand it. Called by menu command. Exits to digitising and control mode.

OVERLAYS CALLED

FILOV3	54
FILOV5	49

SUBROUTINES CALLED

Gnum	Ovretn	Raread
Restcm	Stack	

Appendix B-3

STASYS SUBROUTINES

Length	Name	Subroutines Called	Function
187	ADDP	RAWRIT	Adds XYZ, XYZ coordinates to file CADMAC.R11.
506	ADDLOD	RAREAD RAWRIT	Adds loads to CADMAC.RA4.
	ADVANC	RAREAD RAWRIT	Advances through workspace.
511	ASLOAD	INREC WRDISK RDDISK RAREAD	Assembles loads into global load vectors.
873	BARK	ERMES	Forms line element stiffness matrix.
337	BCLK	RAREAD	Finds boundary conditions for given node.
1289	BSTRES	-	Calculates stresses due to bending in plate elements.
	BUTNUT		Allows entry of numbers by pencil buttons.
457	CADDP	ADDP RDDISK INREC	Looks up displacements of nodal points and calculates their new positions.
229	DBEAM		Displays beam element.
183	DCOL	PLTREC	Displays column element.

Length	Name	Subroutines Called	Function
346	DSLAB	PLTREC	Displays slab element.
226	DWALL	PLTREC	Displays wall element.
	ERMES	-	Writes an error message to DECwriter.
	EXDISK	see WRDISK	see WRDISK
734	FINDSY	RAREAD PLMSG PMSGØ	Finds a symbol in the workspace.
739	FINLIN	ADDP CADDP RAREAD	Finds nearest nodes along positive coordinate axes to given node.
	GETP	RAREAD	Gets XYZ coordinate triplets from CADMAC.R11.
460	GNUM	RAREAD WRDISK RDDISK INREC	Given boundary conditions' coordinates, it determines the node number and zeros appropriate load vectors.
	INREC	-	Function to calculate position of item within a group which is filed in blocks.
698	ISITA	ADDLOD	Checks to see if slab is within area load.
564	ISITL	ADDLOD	Checks to see if beam is within line load.
	LEVS	PLMSG PMSGØ	Asks user if he set the correct levels.

Length	Name	Subroutines Called	Function
	LSTRESS	-	Calculates stresses in line elements from local displacements.
	MULT	-	Multiplies square matrices.
	NDISP	WRDISK RDDISK INREC	Gets global displacements for a given node.
468	NODLK	RAREAD	Given element coordinates, it looks up nodal numbers.
	PLTCRS*	-	Displays a cross.
	PLTREC	PLOTW1	Displays a rectangle.
729	PSTRES	-	Calculates stresses in plate elements due to in-plane displacements.
	PLAK	SWAP ERMES	
	RDDISK		see WRDISK
	RECSRT	SWAP	To sort four corners of rectangle into anticlockwise order starting from bottom left corner. Checks that points make a reasonable rectangle.
871	RWRW	RAREAD RAWRIT INREC	Called by RDDISK/WRDISK to set up calls to RAREAD and RAWRIT, to transfer blocks of matrix and load vectors.

*Indicates Assembly Language.

Length	Name	Subroutines Called	Function
197	STORNO	STORD	Stores numbers as text symbols in workspace.
	SWAP	-	Swaps a pair of real variables.
902	SYMINV	-	Performs symmetrical invertion of blocks of matrix.
	TFLOOR*	-	Converts integer number to ASCIL. Sets message FLOOR - nnn, where nnn is number.
	TNUM*	-	Converts integer number to ASCIL representation Truncates two leading zeros. Leaves three digits.
	TSWAP	-	Swaps pairs of I,X,Y triplets if necessary.
34	WRDISK*	RWRW	Reads and writes blocks of matrix to and from disk.
	ZERO	-	Zeros an array of real variables.

Appendix C

Use of random access files by STASYS and GCADS

CADMAC.RA0 Used by GCADS up to record 10

- Records 20 & 21 - material property sets (STASYS)
- Records 22 & 23 - floor directory (STASYS)
- Record 25 - solution parameters (STASYS)
- Records 14 & 15 - element dimensions (STASYS)
- Records 16 & 17 - load vectors (STASYS)
- Records 34 to 38 - element stiffness matrix (STASYS)

CADMAC.RA1 Workspace (GCADS)

CADMAC.RA2 Used by macro processor (GCADS)

CADMAC.RA3 Dynamically divided into 60 mass storage files for the nodal coordinates of each floor. (STASYS)

CADMAC.RA4 General working file (STASYS)

CADMAC.RA5 Dynamically divided into 60 mass storage files for boundary condition data. (STASYS)

CADMAC.RA7 Redundant

CADMAC.RA8 General working file (STASYS)

CADMAC.RA9 Dynamically divided into 60 mass storage files for load data for each floor. (STASYS)

CADMAC.R10 Redundant

CADMAC.R11 Used for storing three-dimensional data describing nodal point positions before and after analysis. (STASYS)

CADMAC.R12 Used to store global stiffness matrix and load vectors. (STASYS)

CADMAC.MS1 Dynamically divided into 128 mass storage files for GCADS. Only 100 may be accessed by menu command. STASYS reserves 60 of these for graphic data of each floor.

Reprinted from **COMPUTER AIDED DESIGN** Printed in England

The use of CADMAC systems in general draughting

C. B. Besant, A. Hamlyn*, A. Jebb and P. McClintock†
Department of Mechanical Engineering, Imperial College, London, England.

CADMAC is a stand-alone fully interactive computer aided design system. It is based on a mini-computer and a range of peripherals, suitable for particular applications and environments. The system is complete with the software necessary for application.

Draughting is of fundamental importance in most applications in the engineering and architectural fields. Many of the applications require computation to be performed during and on completion of the design process. Compiling and formatting the required data for computation on a computer can be a difficult and a time-consuming task. Also, constructing the actual drawing to a required standard of quality can be tedious.

A collaborative programme of work was undertaken at Imperial College in conjunction with John Laing Design Associates, Computer Equipment Company – D-mac Ltd, and Scott Wilson Kirkpatrick & Partners to develop methods and software for using CADMAC systems in the general draughting field. A description is given of these techniques together with their application to a real problem.

A detailed description is also given of the CADMAC configuration used for this work together with a description of configurations suitable for production environments.

Finally, the economics of utilising computer-based systems for general draughting are also discussed.

1 INTRODUCTION

Computer draughting is a special and important part of c.a.d. Many firms and some universities are now examining techniques for speeding up the draughting process by removing the more tedious parts of draughting and providing aids for the draughtsman. Drawing is, and will be, for many years, the main method of communication between engineers. Modern computer-produced drawings are of a high standard with information often presented with great clarity.

Once a design is in the computer it may be manipulated in a number of ways and the results recorded on a plotter. For example, if the design is in the computer in three-dimensional form then it may be rotated through any angle and a drawing produced showing any view. C.a.d. techniques have often resulted in a better drawing presentation. For example, pipe network layouts are often drawn as a line diagram in 'isometric' form, using symbols for dimensions and data. The symbols are then identified on a print-out from a teletype or lineprinter and the listing is attached to the drawing. By this means a large amount of information can clearly be presented on one drawing.

The biggest advantage in computer draughting is the ability to modify and update computer-held information. It is here that the largest cost savings can often be made in draughting. Computer drawings have often been considered as being just a by-product of an aim to get into the computer store all the product information. However, with over 100 000 draughtsmen presently employed in industry in the UK and with the need to use drawing offices more efficiently, much effort is now being put into computer draughting in its own right.

Five years ago c.a.d. was in the province of large computers but in the past three years, the availability and falling costs of the new minicomputers have greatly increased the possibilities of bringing c.a.d. techniques to a wide range of users.

A research group in the department of Mechanical Engineering at Imperial College, committed itself in 1969 to the development of a c.a.d. system based around a minicomputer, such that the system could be operated in a stand-alone mode or as a satellite to a large computer. A system primarily for engineers was the main target and so a practical communication system around the minicomputer was developed. The complete system now known as CADMAC, is being marketed by Computer Equipment Company Ltd. which is a new sister company of D-mac Ltd. who collaborated with the College on the development.

A description of the CADMAC system is given in this paper together with a discussion on its use in two application areas, namely architecture and structural engineering.

2 THE CADMAC SYSTEM

2.1 Peripherals and software

CADMAC has been designed to give users the widest possible choice of peripherals and software depending on the application and environment. At the lowest end of the range is a digitizer or a tablet connected in an online mode to a desk-top minicomputer of 4 k of 16-bit or 12-bit word capacity with a teletype. Such a system can be used to digitize sketches with the computer aiding this process. It is common to operate the system with a menu situated to one side of the tablet or digitizer area. The menu consists of an area divided into a number of squares. Digitizing a point within any one of these squares, causes a routine to be executed within the computer. Digitizing an instruction (LINE) allows the user to input just the end-points of a straight line and the computer will store the appropriate

* Sponsored by John Laing Design Associates.
† Sponsored by Scott Wilson Kirkpatrick & Partners.

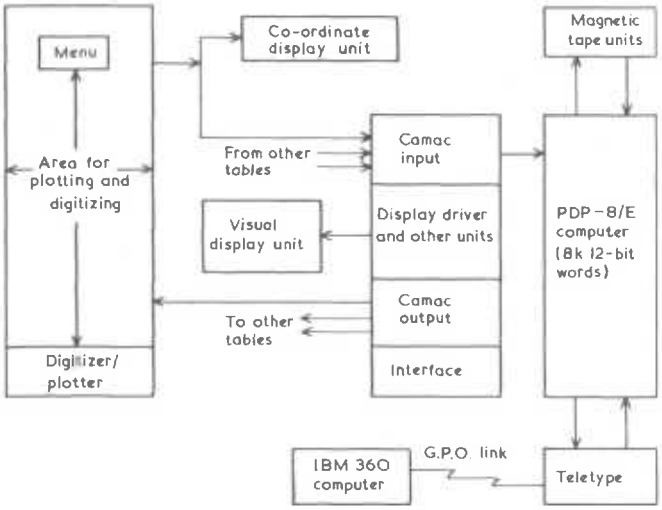


Fig.1. Diagram showing peripheral equipment around a minicomputer in the CADMAC-8 system.

straight line between the points. The working scale is simply chosen by digitizing an instruction (SCALE) and typing in the desired scale on the teletype. There are many other functions which are available to aid the user that allow him to format the design in the computer and produce an output in the form of punched paper tapes via the teletype.

The paper tape can then be taken to a larger computer if any complex calculations are required in the design; such as stress analysis, heat transfer or cost analysis. A paper tape may also be fed to an offline plotter where a drawing can be produced.

These small c.a.d. systems have many other uses besides creating a drawing in computer format. They can be used for template construction, the computer working out areas and cutting lengths; graph analysis where the computer can be programmed to handle interpolation, integration of areas and editing.

2.2 Work station for the designer

The smaller CADMAC systems are of limited use to the design engineer since a display or plotting device is not included. However, the CADMAC-8 system which is in the middle of the range is a complete work-station for the designer. The system configuration is shown in Figure 1 and consists of a combined input/output table which can be used as a digitizer and plotter and operates in an interactive online mode with the minicomputer. A storage tube is an integral part of the system which can be controlled from the digitizer, a joystick or teletype. The computer is a Digital Equipment Corporation PDP-8/E with 8 k of core and magnetic tapes or disc backing store.

While the CADMAC-8 system can perform all the draughting requirements of a designer, it has limitations when it comes to performing complex analysis calculations. This problem has been overcome in a number of ways depending on the environment and application. One method of increasing the computing power has been to link the system with an ITT/IBM 360 time-sharing computer via a modem using a GPO telephone line as with a teletype on the CAD Centre's multi-access system. This type of system has proved very successful on a joint research programme with John Laing Ltd where the CADMAC-8 general draughting programs have been used in conjunction with the Laing costing suite, Laingwall, which were run on the ITT systems. The user controls the whole operation from the CADMAC work-station and, for example, a building such as an office block can be designed, casted and general arrangement drawings produced in less than a day. Laing are finding that the CADMAC system approach is suitable for use by both their architects and engineers.

The second approach at increasing the power of the system has been to substitute the more powerful PDP-11 computer for the PDP-8 to give the CADMAC-11 system. A typical system now in use at Imperial College is shown in Figure 2 with its general layout being similar to that given in Figure 1. The computer in this system has 12 k of core, a 1.2 million word disc and 2 magnetic tape units. Extensive use is made of the Fortran 4 capability with the PDP-11 disc operating software which is an attraction for engineers who are familiar with Fortran. The system is capable of handling the general draughting work, being faster and easier to use than the smaller CADMAC-8 system. It can also be used for applications programs.

One typical research contract between the College and Scott Wilson Kirkpatrick & Partners involves the use of CADMAC-11 in the field of finite element stress analysis on structures. The system is ideal for complex data preparation for such calculations and for the presentation of results in a graphical form. The computer is relatively small for finite element stress type problems and has involved new techniques for obtaining solutions. There is a growing trend in using this type of computer for analysis work as well as for online control applications since its capital cost is relatively low, it is small in physical size and can be used in almost any environment. In fact, two CADMAC systems at Imperial College work quite happily in engineering laboratory conditions.

2.3 Tool for design engineers and architects

It is appropriate to discuss some of the detailed features of the CADMAC system since it was designed at the College in close collaboration with industry. More importantly, it was designed with the aim of being a practical tool for design engineers. Engineers usually generate their ideas in the form of rough sketches. A system was therefore envisaged which would either help the designer to turn the

Fig.2. A CADMAC-11 system at Imperial College being used in the input mode.

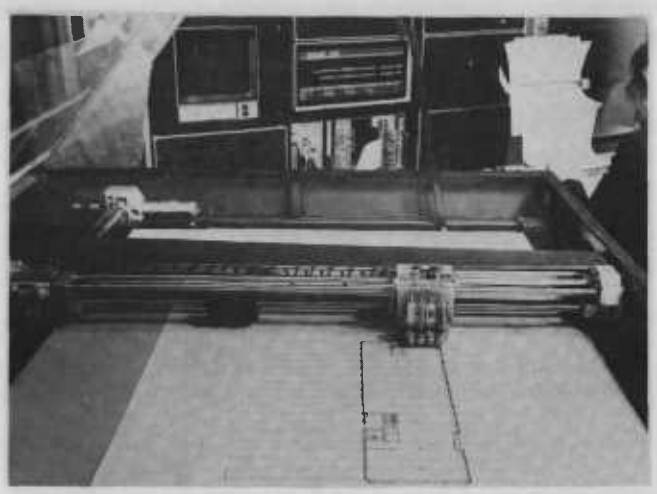


Fig.3. A CADMAC table being used in the plotting mode.

rough sketches into the desired final form and in computer format or even a system which would assist the designer in the forming of the initial design. A digitizing table working interactively with a computer seemed to be a more practical device for inputting data to the computer than the light-pen, keyboard method in conjunction with a display screen.

However, a display was considered to be an essential feature of the system so that data fed into the computer could be immediately verified. Also, the display is essential for speedy editing of information stored in the computer configuration. Another essential feature considered to be desirable was a plotter so that good quality engineering drawings could be produced.

A table was therefore designed and built at the College in collaboration with D-mac Ltd for performing both the digitizing and plotting functions.

The table contains two working surfaces, one beneath the other. The top surface is of toughened glass and is the input digitizing surface. Beneath this surface is a follower mechanism which consists of a carriage, containing sensing coils, running in the y direction on the top of a gantry moving in the x direction. The carriage and gantry, shown in Figure 3, both run on linear bearings which consist of ground stainless steel guides and standard roller bearings. The carriage and gantry are each driven by separate servo-motors. Positional sensing in x and y is effected by two moiré fringe shaft encoders which are driven via separate stainless steel wires attached to the gantry and carriage respectively.

The pencil used on the digitizing surface consists either of a cross-hairline or ballpoint pen arrangement, both of which contain a coil through which a 400 Hz current is passed. The magnetic field set up by the digitizing pencil, is sensed by the coils on the carriage and a signal is passed via amplifiers to the servo-motors causing the carriage to be driven to a position directly beneath the digitizing pencil.

The second table surface is of laminated wood covered with formica and a p.v.c. backing sheet, and is situated below the moving gantry. This surface is the output part of the table where a hardcopy of a drawing may be produced on paper. A set of four output pens is attached to the side of the carriage. The output pens may be either ballpoint or ink type. The movement of each pen is controlled by a solenoid which in turn is controlled from the computer interface.

The two table surfaces are contained within a thin rectangular box structure which is mounted on a hydraulic pedestal arrangement. The glass table top may be opened to obtain access to the plotter.

A number of different table sizes are now in production up to a maximum of 150 cm x 100 cm working area (as shown in Figure 2).

2.4 Menu card

As with most CADMAC systems a menu is attached to the right-hand side of the digitizing surface (Figure 2). Teletype operations are minimized and the system is controlled with the digitizing pencil using the menu. This method seems suitable for engineers and architects who tend to do much of their thinking with a pencil in their hand. The system is easy to use and no programming experience is necessary.

Figure 4 shows a typical menu suitable for general draughting work. Figure 4(a) shows the overall configuration of the menu which is divided into a number of major areas. The main area is the system commands which are shown in detail in Figure 4(b). System commands make up the basic control functions in the general draughting software. User commands are for specific programs such as stress analysis and heat transfer, which can be used in conjunction with the data created by using the system commands. Data may be stored under the file section or under levels which is a sub-division of Files. Standard Components, known as macros, may be stored on tape or disc under Files which are used for speedy assembly of designs. The area name Symbols is used for computer generated items such as circles, arcs or any other shape which can be easily calculated.

2.5 System costs

Approximate costs of CADMAC systems are shown in Table 1 but it should be remembered that being modular in concept, both in hardware and software, the system can take many different forms. In an effort to produce a low cost system, some customers of the system are now requesting multiple table configurations around one computer.

Table 1. Approximate costs of a few CADMAC configurations

CADMAC configuration	Approximate cost
Tablet with 4 k desk top mini computer complete with software.	£600
Digitizer online to PDP-8/E computer with 4 k core and magnetic tape unit. Also included is a v.d.u. and complete operating software.	£16 000
CADMAC-8 System consisting of 1 m square input/output table, v.d.u. and PDP-8/E computer with 8 k of core and magnetic tape or disc unit. The System includes a general draughting and operating software package.	£27 000
CADMAC-11 System consisting of 1.5 m x 1 m input/output table, v.d.u. and PDP-11/40 computer with 16 k core, 1.2 M word disc and high speed paper tape reader/punch. The System includes a comprehensive general draughting software package.	£40 000

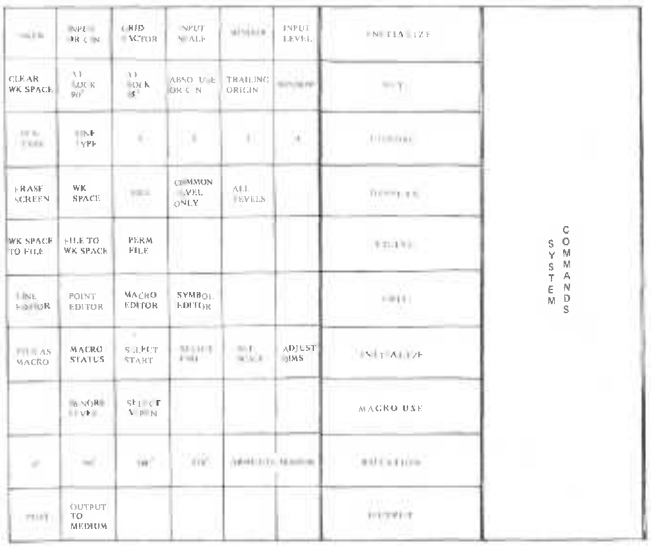


Fig.4(a). The Menu System used with the CADMAC general draughting software.

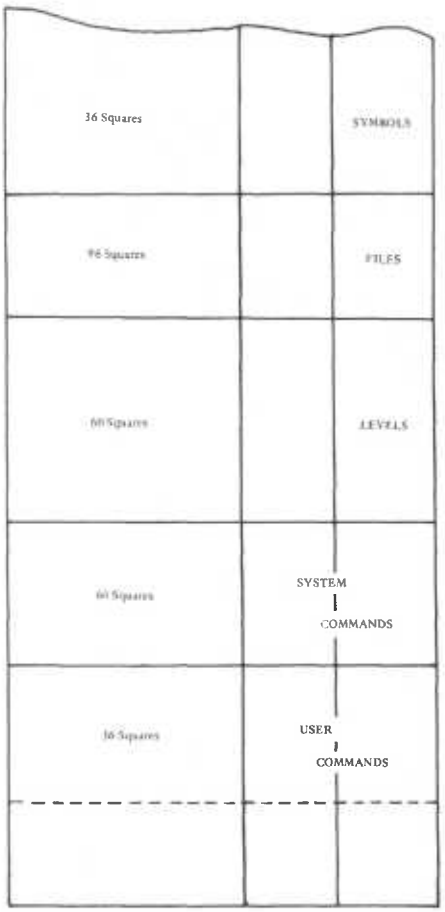


Fig.4(b). The System Commands in the Menu of the general draughting software.

3 CADMAC APPLIED TO DRAUGHTING IN ARCHITECTURE

It is common practice for architects to produce rough sketches of building layouts based on a grid. Converting these rough sketches into working drawings is time-consuming, tedious and costly. It involves the employment of detail draughtsmen which can result in inflexibility and problems when workloads fluctuate. In order to test the application of CADMAC to the present problem, some rough architect's sketches of a hospital scheme were converted into working drawings using the system.

There were two sketches used in the test, one sketch, drawing number SK 10 is a general layout of a wing of the hospital and is shown in Figure 5. The other sketch, drawing number SK 9, gives the detail of two rooms in the general layout and is shown in Figure 6. The overall method of digitizing the sketches was to first create a set of macros (standard items) consisting of doors, windows, walls and even a complete room with its contents. A system of guidelines was then digitized and macros were assembled, using these lines to form the two drawings which were subsequently produced in hardcopy form on the plotter. The details of this procedure are now described.

Sketch SK 9, giving the detail of rooms, was attached to the digitizing surface of the table. The system program was then called via the teletype which resulted in the initiation of the CADMAC set-up procedure. The following requests with their corresponding answers which are all performed via the teletype, are given below:

1. SET TABLE ORIGIN – A point marked by a cross in the bottom left-hand corner on the table is digitized. This procedure sets the table origin to co-ordinate (o, o) so that all points on the table will have positive co-ordinates.
2. SET DRAWING ORIGIN – A point on the drawing is digitized to fix its origin.
3. SET SKEW – Two points are digitized on an axis of the drawing. This ensures that any skew in the axes of the drawing with that of the axes of the table is automatically removed by the computer as each point is digitized.
4. INPUT SCALE– The required scale for digitizing is typed in on the teletype.
5. GRID – The required grid factor is entered via the teletype.

A data level is then specified by digitizing (button 1) an appropriate LEVEL square on the menu. The LEVEL system allows a complex drawing to be divided into discrete sets of data such as, walls, columns and service ducts and these are all stored separately under LEVEL squares. A number of levels are reserved for storage of construction lines, and grid lines which may be suppressed when the final drawing is produced. The LEVELS may subsequently be combined into a single file under a FILE square. The system is then ready for accepting data.

The first task in digitizing sketch SK 9 is the creation of a macro databank consisting of windows, doors, wall panels and room furniture. Each macro is stored under a FILE square and a rough sketch of the macro drawn in the appropriate square for identification.

The actual digitizing procedure used to create a macro is quite straightforward with little further requirement for digitizing menu squares until the completion of a macro.

The normal digitizing mode is LINE MODE. In this mode, button 2 is used to break a line and button 1 is used

to specify the start and subsequent points on the line. All points are joined by straight lines as they are digitized. Other digitizing modes include arc mode, circle mode, dimension mode and tangential arc mode.

It has been found very difficult to produce good quality drawings using a freehand digitizing process. A number of digitizing aids have therefore been built into the system, forming a close analogy to a set-square and ruler system:

Pressing button 6 will switch into operation a 15° or 90° lock facility. When in operation this constrains all lines to be at multiples of 15° to the arcs. When a point is digitized the co-ordinate display unit zeros on the point. This enables lengths of lines to be measured. Measuring lengths manually, however, can be a time-consuming process especially when no grid round-off is being used, and so a drive mode is incorporated by which the cursor may be driven around the screen in set incremental distances by a menu patch. This permits very fast specification of line length.

Verification of digitized data is performed by viewing the display during the digitizing process. When an error is located, the area in question is first magnified on the screen. This function is performed by calling the WINDOW instruction, button 4, followed by digitizing two points on either side of the area where the error is located. The display screen is then automatically erased and the enlarged area drawn on the screen such that the two digitized points lie on the extremities of the screen. If an error in digitizing is

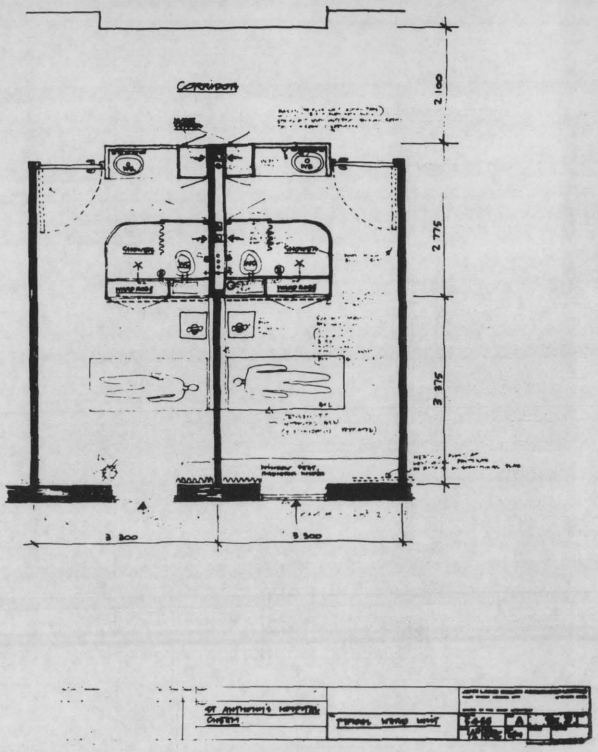


Fig.6. Rough sketch SK 9. Room detail of a typical ward – St. Anthony's Hospital, Cheam.

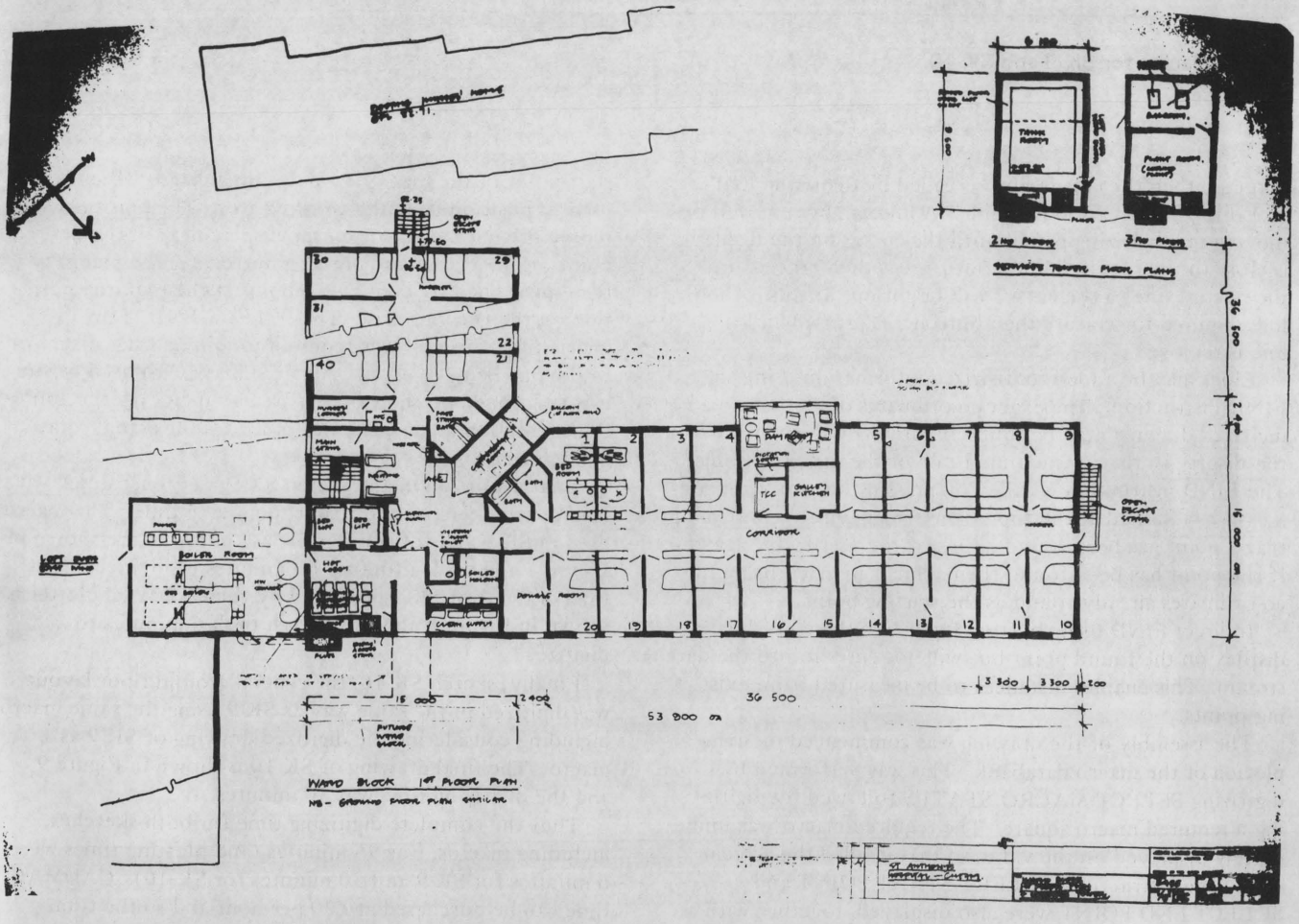


Fig.5. Rough sketch SK 10. Layout of a typical ward unit – St. Anthony's Hospital, Cheam, London.

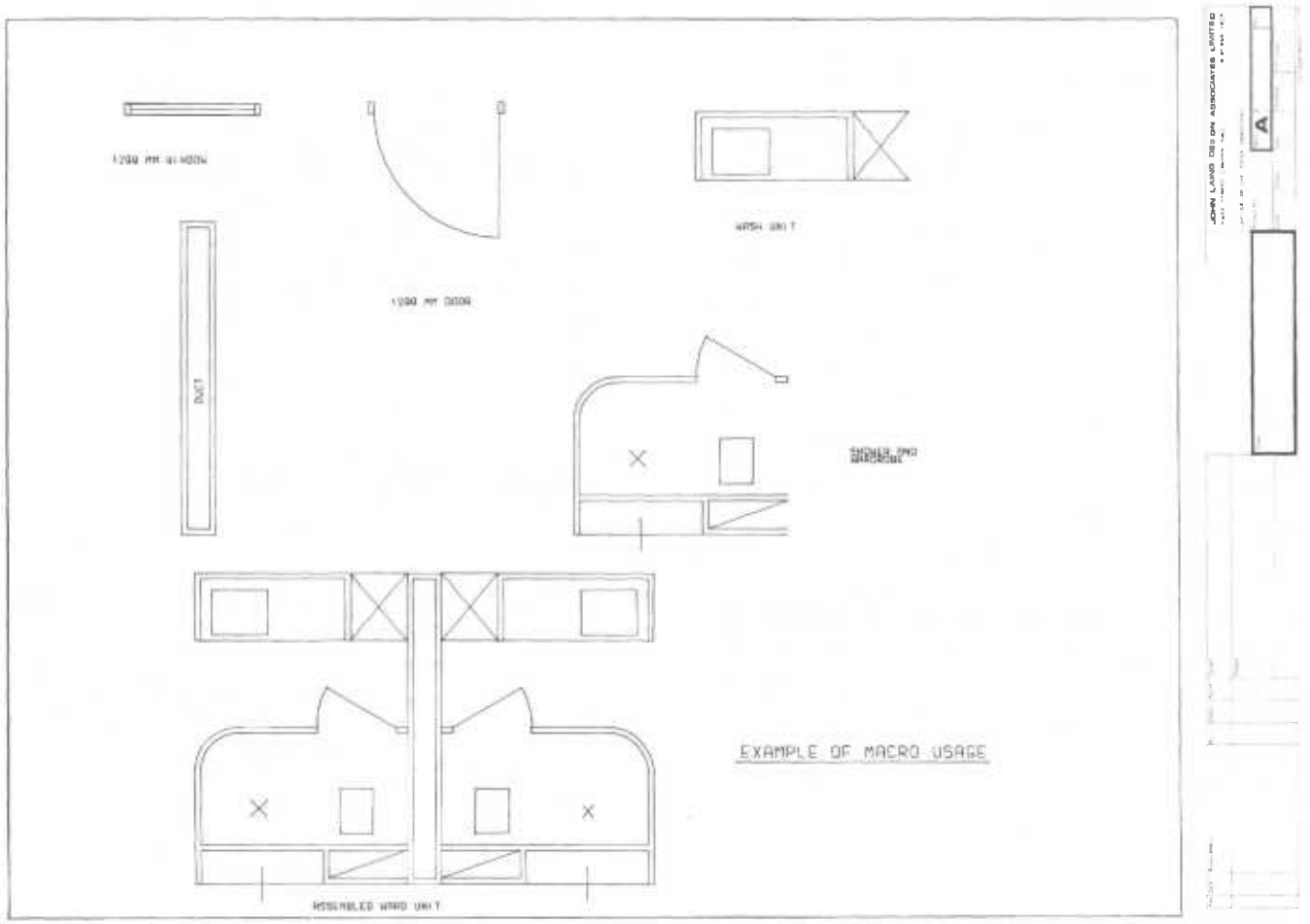


Fig.7. Macros for SK 9 and SK 10.

detected then the edit facility is called by digitizing EDIT.

Editing information is simple. A line can be removed by moving the digitizing pencil until the cursor on the display is close to the required line. Button 1 is then pressed and the nearest line to the cursor will bright-up. If this is the line required for erasure then button 3 is pressed and the line is removed.

Lines may be added to digitized information using the FIND instruction. The exact co-ordinates of a point on a digitized drawing may be found by driving the cursor on the display so that it is within 4 mm of the required point. The FIND instruction is called by pressing button 7 and a message is flashed at the top of the screen, either confirming that a point has been found or asking the user to try again. If the point has been found then a line is inserted using the co-ordinates already found as the starting point.

Indirect FIND using button 5 will zero the co-ordinate display on the found point but will not enter it into the data stream. This enables distances to be measured from existing points.

The assembly of the drawing was commenced on completion of the macro databank. This was performed by digitizing SELECT MACRO STATUS followed by digitizing a required macro square. The required macro was immediately displayed on the v.d.u. so that it filled the screen area. The instructions SELECT START POINT and SELECT END POINT were also displayed, together with a flag which indicated the orientation of the macro. The start-point of the macro is a point on a macro which is used to position it on the working area. The point is selected by driving the cursor on the display near to the desired point on the macro and pressing button 1. The macro is then positioned by digitizing a point at the required position on the working area. The SELECT END POINT instruction is used if it is required to join another macro on to the one being selected. A macro may be rotated before it is positioned on the working area by digitizing the appropriate rotation angle. The flag indicates the actual rotation of the macro.

A large portion of the sketch SK 9 was constructed with macros which permitted a short digitizing time. The macros used in SK 9 and for subsequent use in SK 10 are shown in Figure 7 and took a total of 60 minutes to digitize. The final drawing of SK 9 produced by the CADMAC plotter is shown in Figure 8 and this sketch took 6 minutes to digitize.

Finally, sketch SK 10, the general ground floor layout, was digitized in the same way as SK 9 using the same macros, including considering the digitized drawing of SK 9 as a macro. The final drawing of SK 10 is shown in Figure 9 and the digitizing time was 30 minutes.

Thus the complete digitizing time for both sketches, including macros, was 96 minutes (and plotting times were 6 minutes for SK 9 and 30 minutes for SK 10). CADMAC time can be purchased at £20 per hour and so the total cost for the work was £31.

4 APPLICATION OF CADMAC TO DRAUGHTING IN STRUCTURAL ENGINEERING

One of the many important aspects of structural engineering is the detailing of reinforced-concrete. There has been an enormous expansion in the number of reinforced-concrete structures over the past ten years and this has created a shortage of experienced detailers who can cope with the increasing demand for working drawings, The detail working drawings tended to become more complicated because of the increasing refinements in methods of design and also by the lack of experienced detailers.

A characteristic of reinforced-concrete has been the large variations in design of any one number which would satisfy the criteria laid down by the architects and engineers. With the structural engineers wishing to standardize the reinforced-concrete design procedures in order to increase the efficiency of construction work, the Concrete Society and Institution of Structural Engineers produced a report [1] which is intended to be used in conjunction with BS 1478 [2] with a view to satisfying the aims of the industry.

The work reported in this section is solely concerned with the draughting processes in reinforced-concrete design and reference [1] was taken as a basis in applying CADMAC to this specific problem.

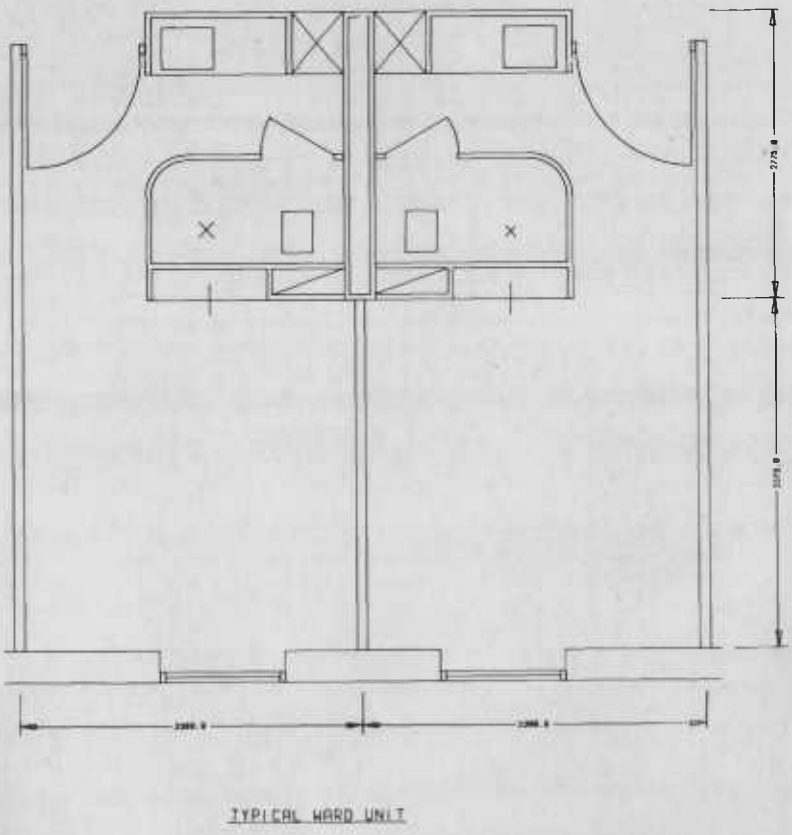
The work has been in progress for eighteen months on producing drawings and bar schedules as an automatic process from the completion of the design, but the aim here is to show that CADMAC can create the shapes and symbols used in reinforced-concrete detailing and produce a finished drawing.

A completed structural engineering general arrangement drawing was selected for the test. The drawing consisted of the structural outline and dimensions. Also shown were the reinforcing bars.

The system set-up procedure was similar to that described in Section 3 with an input scale set at 1:100. The data in the drawing was filed under a number of Levels which are now described.

LEVEL 1 was the basic grid for the overall drawing. This was created by setting a grid factor of 5.5 m and digitizing only the required grid lines.

LEVEL 2 contained the structural outline. This was



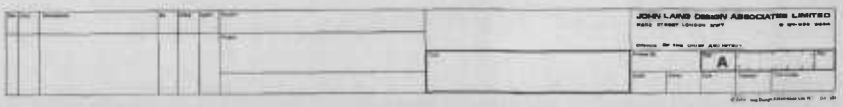
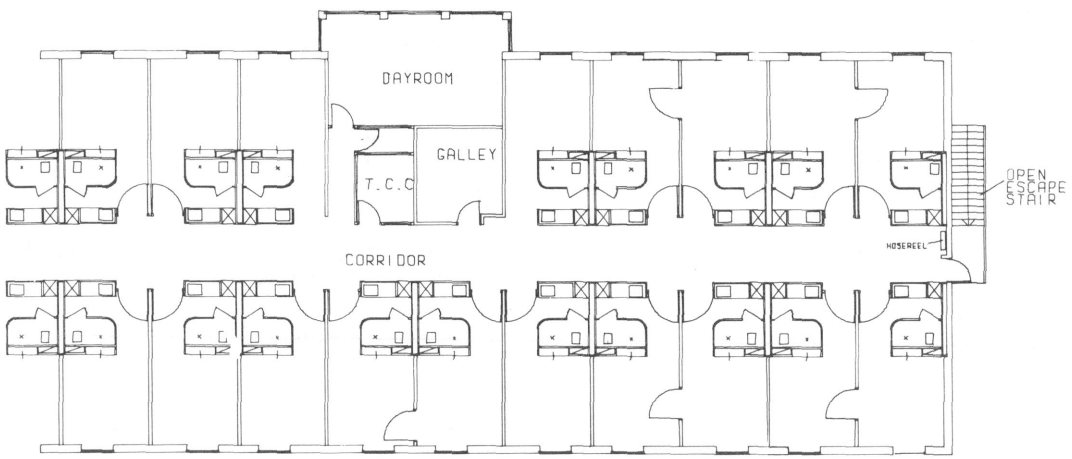


Fig.8. Final drawing of SK 9 – Room detail.

Fig.9. Final drawing of SK 10 – Ward layout.



WARD BLOCK 1ST FLOOR PLAN

N.B. - GROUND FLOOR PLAN SIMILAR

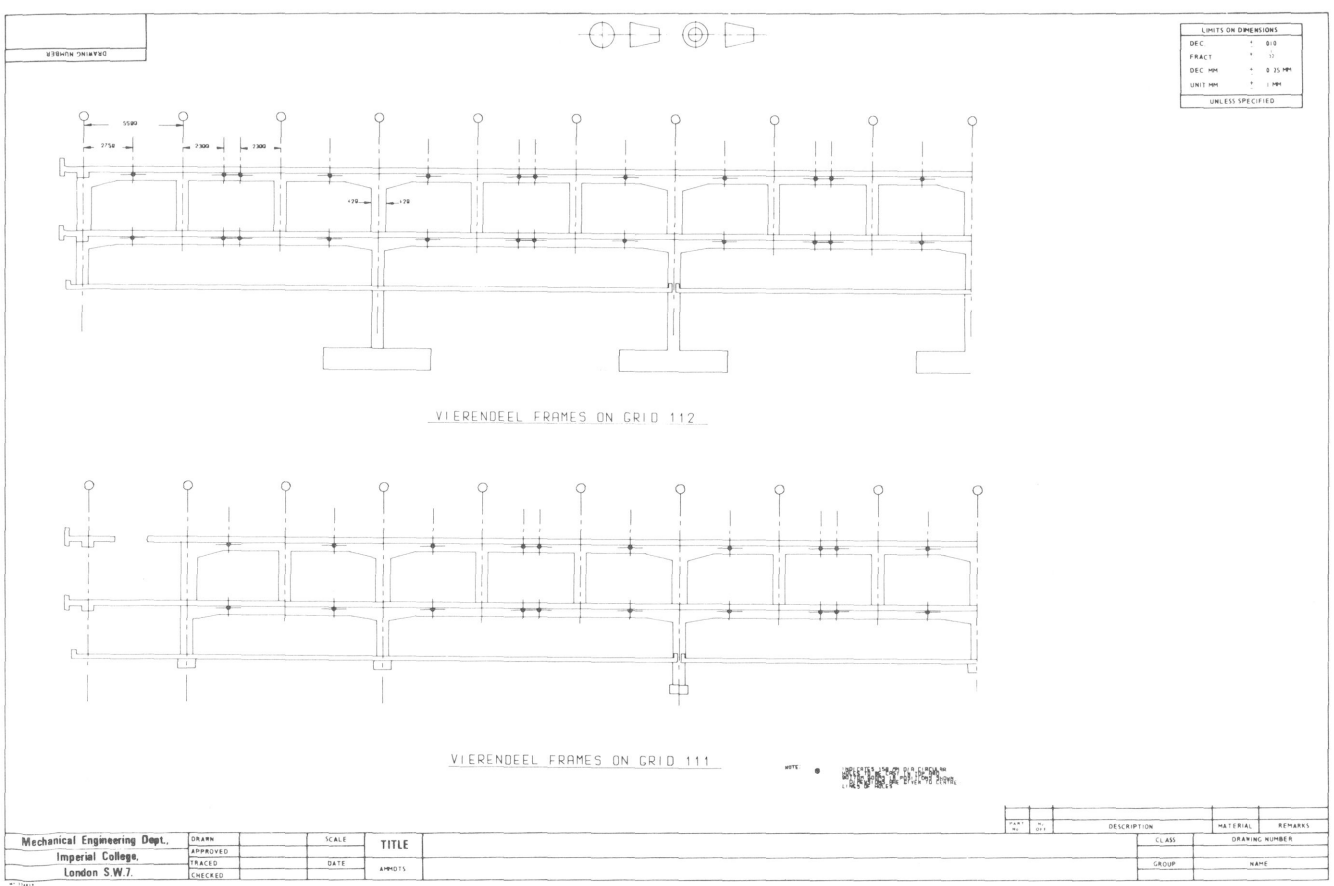


Fig.10. General arrangement of reinforced-concrete frame.

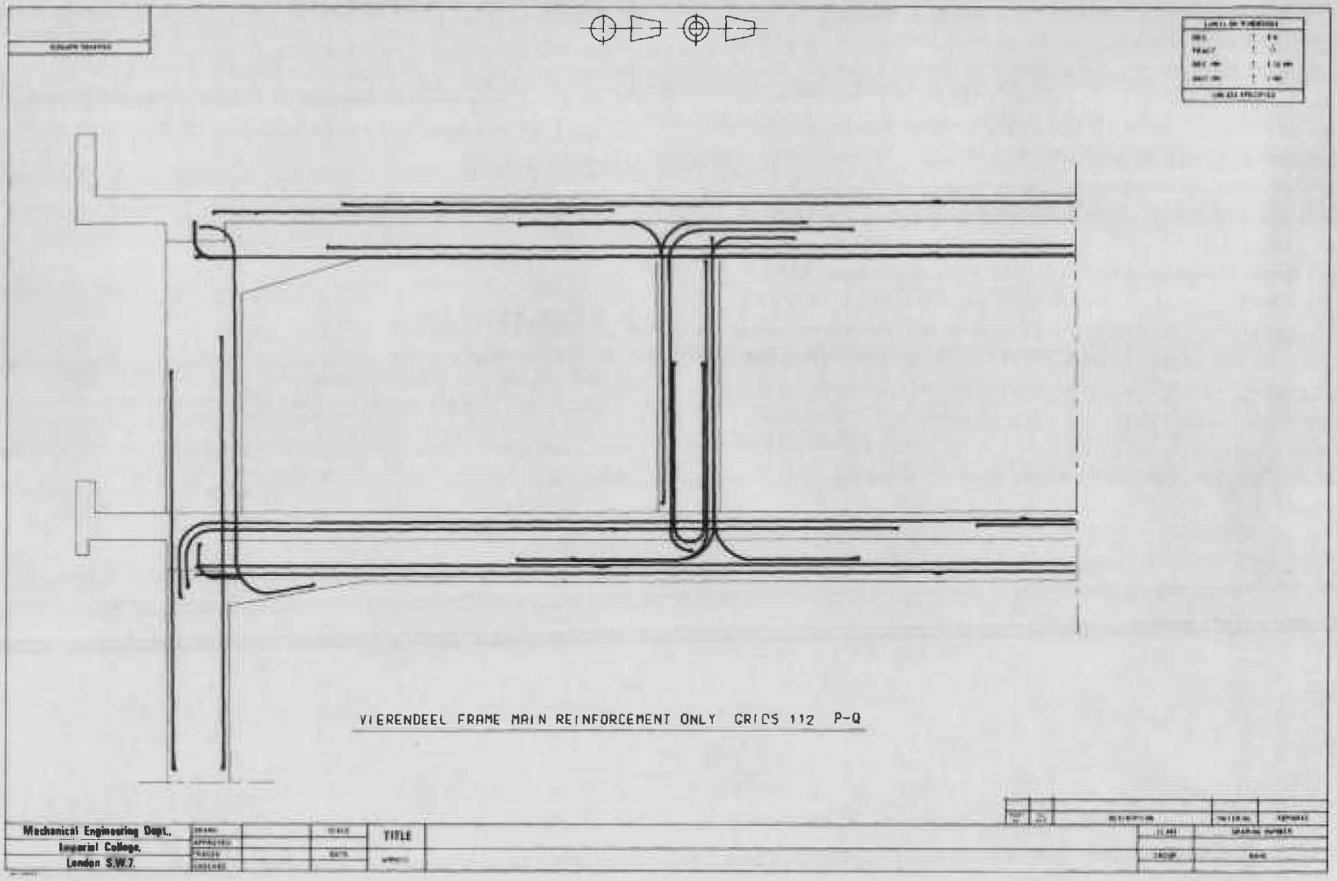


Fig.11. Typical reinforced-concrete detail drawing.

digitized with a grid of 5 mm. All lines were set down to the absolute accuracy of the dimensions by utilising the TRAILING ORIGIN mode in conjunction with the co-ordinate display unit. Considerable use was also made of the CONTROL 90 facility.

LEVEL 3 contained all the alphanumerics. Text strings were set up on the teletype and placed on the work area by digitizing ALPHANUMERICS followed by digitizing the start point of the string. Dimensions were entered using an automatic dimensioning system. This is performed by digitizing the instruction DIMENSION followed by digitizing the two points where the dimension is required. The FIND facility is used in the determination of these points. When the second point has been found and digitized, the computer calculates the distance between the points and inserts the correct dimension. A plot of the general assembly is shown in Figure 10.

A section of the general assembly was 'windowed' to provide an outline required for a detail drawing. A new file was created for the detail drawing which was also divided into a number of levels defined below:

1. LEVEL 1 and LEVEL 2 – Windowed Levels 1 and 2 are of the general assembly.
2. LEVEL 3 – Outlines of section.
3. LEVEL 4 – Main reinforcement in elevation.
4. LEVEL 5 – Main bars in section.
5. LEVEL 6 – Links in section.
6. LEVEL 7 – Links in elevation.
7. LEVEL 8 – Alphanumerics, service hole symbols, arrows and section flags.

The detail drawing is shown in Figure 11. For this drawing considerable use was made of the fillet mode in producing the reinforcement bars in LEVEL 5 and the links in LEVEL 6. Special symbols were created for arrows, section flags and shaded circular sections. Macros were made of groups of arrows, circular sections, repeated alphanumeric strings and section outlines.

The time taken to digitize and plot the general assembly was 210 minutes and 65 minutes for the detail drawing. Thus the cost of the work was £91.

5 CONCLUSIONS

The CADMAC system proved to be most suitable for the present applications being easy to use, easy to check for errors and easy to correct mistakes. In most of the draughting work attempted so far, the most time-consuming part is the creation of the macro databank. In many applications, once a relatively small set of macros have been created, a number of users can be satisfied, each user perhaps only having to build a few extra macros in order to complete his task.

Thus, if the macros had already been built for the architect's work described in Section 3, the general layout drawing could have been completed in 60 minutes including plotting, and the detail drawing in 12 minutes. A draughtsman would have taken 960 minutes to complete the same work by standard drawing methods and the drawings would then have to be traced. The cost savings amount to about 50%, but more significantly the time saving is 888 minutes.

When one considers that other programs, such as those providing cost analysis, can be linked with the draughting programs then enormous economic benefits can accrue in the use of CADMAC. In fact a building cost program was run in conjunction with the draughting program in an architectural application and once the draughting had been completed, the cost analysis could then proceed automatically. This approach has considerable advantages in the field of preparing designs and estimates for tenders.

Other draughting applications such as in structural engineering, printed circuit layouts and mask generation, mechanical engineering and textile design are all showing considerable promise on CADMAC. Systems geared for large-scale production type environments are under consideration. CADMAC will take on the form of multiple table configurations in a time-sharing mode for this type of work and the cost per user can then be reduced.

6 ACKNOWLEDGEMENTS

The authors wish to express their thanks to all those in Computer Equipment Company and the Computer Aided Design Unit at Imperial College who have helped and made this work possible.

7 REFERENCES

1. 'The detailing of reinforced concrete'. Report of the Joint Committee of the Concrete Society and the Institution of Structural Engineers. Technical Report TRCS 2, (May 1968).
2. 'Specification for bending dimensions and scheduling of bars for the reinforcement of concrete.' British Standard 4466, (1969).

The use of computer aided design techniques in printed circuit layouts

C. B. BESANT*, A. HAMLYN† and P. M. McLINTOCK‡
(Department of Mechanical Engineering, Imperial College of Science & Technology, Exhibition Road, London, SW7, England)

Modern printed circuits are manufactured using a mask. This mask can be prepared by directing a light beam around a photographic plate to describe the required circuit. An etching process is then used to transfer the circuit from the photographic plate (the mask) to the circuit board.

A description is given of the techniques used to transfer a circuit design into a form suitable for input to a computer and thence into a data tape for use in controlling a high accuracy plotter, where the actual mask can be created using a light head. The work was performed using the CADMAC computer aided design system to formulate the data tape. This tape was subsequently used with a high accuracy Kingmatic plotter.

(Received on 19th February 1973)

1. PRINTED CIRCUIT MASK GENERATION

The initial requirements for p.c. mask generation involved the need for a total system which would allow a designer or user to digitize a circuit layout from a drawing on temperature stable material; produce a drawing of the digitized layout for checking purposes and lastly to produce a magnetic or punched paper tape for use on a high accuracy plotter system for final mask generation.

It was further required that the system used in the digitizing process should incorporate some interactive design features so that the designer might be assisted during the early stages of a design. This necessitates the use of a c.r.t. within the system so that the user can view immediately any information as it is digitized. The use of an online computer is implicit in such a system and it is required that this computer should be programmed in such a way that the user may build up a databank of macros (standard items, such as integrated circuits) which can be retrieved and displayed when required and then inserted into the desired position on the design. The user should thus be able to perform the following basic functions on the macros—shift, scale-change and rotation.

A number of other features were also thought to be desirable, such as the ability to 'window' certain areas on the table and then enlarge them on the c.r.t.: to reproduce macros and to use computer generated symbols such as arcs and circles.

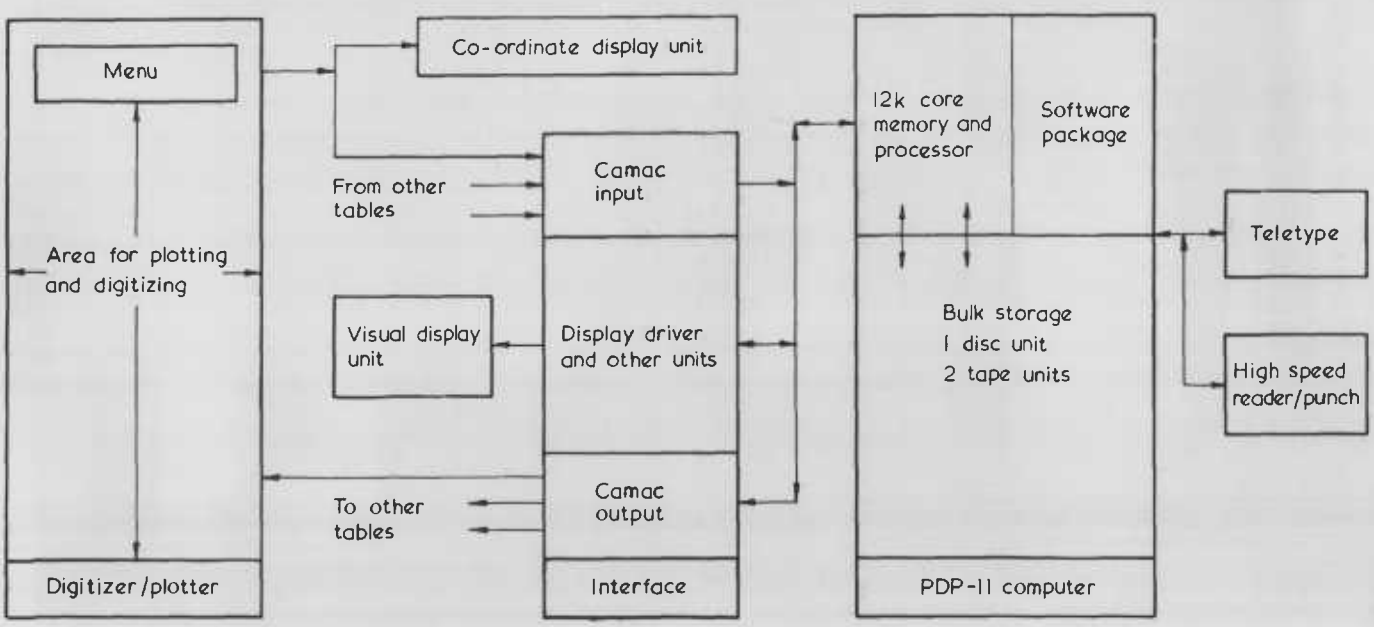
2. CADMAC – AN INTERACTIVE COMPUTER AIDED DESIGN SYSTEM

Only a brief description of CADMAC is given here as full details of the system and general drafting software can be found in Reference 2.

The configuration of the CADMAC system used for the present work is shown in diagrammatical form in Figure 1. It consists of a PDP-11/20 computer with 12k of core

* Technical Director, Computer Equipment Co. Ltd., High Wycombe, Bucks.
† Sponsored by John Laing Design Associates Ltd.
‡ Sponsored by Scott, Wilson Kirkpatrick & Partners.

FIGURE 1. The CADMAC-11 system using a PDP-11/20.



memory, two DECtape units, each unit being capable of storing 200k of 16 bit words and a 1.2 megaword disc. The computer also has a high speed paper tape reader/punch and an ASR 33 teletype.

The computer is connected in an on-line mode to a CADMAC combined digitizing/plotting table via a Camac universal interface. The table has two surfaces, the top one being a glass digitizing surface. The glass top may be opened to reveal the lower plotting surface. The principle of operation is based on the D-mac servo-follower mechanism which can be controlled by a digitizing pen on the top surface when in digitizing mode. The computer controls the servo-follower mechanism when in plotting mode and a set of four pens attached to the moving gantry in the table performs the actual plotting.

The c.r.t. which is incorporated into the system is a Tektronix 611 storage tube display. A co-ordinate display unit is also built into the system which can be used to display the x, y co-ordinates on the table or other information such as length of lines. An overall view of the CADMAC-11 system at Imperial College, London is given in Figure 2.

3. USE OF CADMAC IN P.C. LAYOUTS

The software used for the present application is essentially the general drafting software described in Reference 2. This software is modular in concept and consists of a set of seven system modules which are used to supervise the general operation of CADMAC such as the initial system setup or user data-handling. The system modules are designed such that user application program modules can be 'plugged' into the software system.

The user modules can be called into operation by using the menu system which consists of a set of squares 2 cm x 2 cm positioned on one side of the digitizing surface. When a point is digitized within a square, a certain user application program is called into core and started. The menu used for the present work is shown in Figures 3 and 4.

Using the menu system is very simple, for example a circle can be inserted by digitizing the symbol CIRCLE followed by digitizing the centre point and a point at the correct radius. The circle is displayed on the c.r.t. as soon as the last point has been digitized. The meaning of each menu square is self-explanatory from the identification in the square. If, however, the reader is in doubt as to the meaning of a particular instruction, an explanation is given in Reference 2.

The digitizing pen also carries buttons which control the user modules most frequently required. This eases the user's general operational task by reducing the need for frequent access to the menu area. The buttons also exercise control within user modules since options may be flashed to the operator via the screen.

FIGURE 2. A view of the CADMAC-11 system at Imperial College, London.

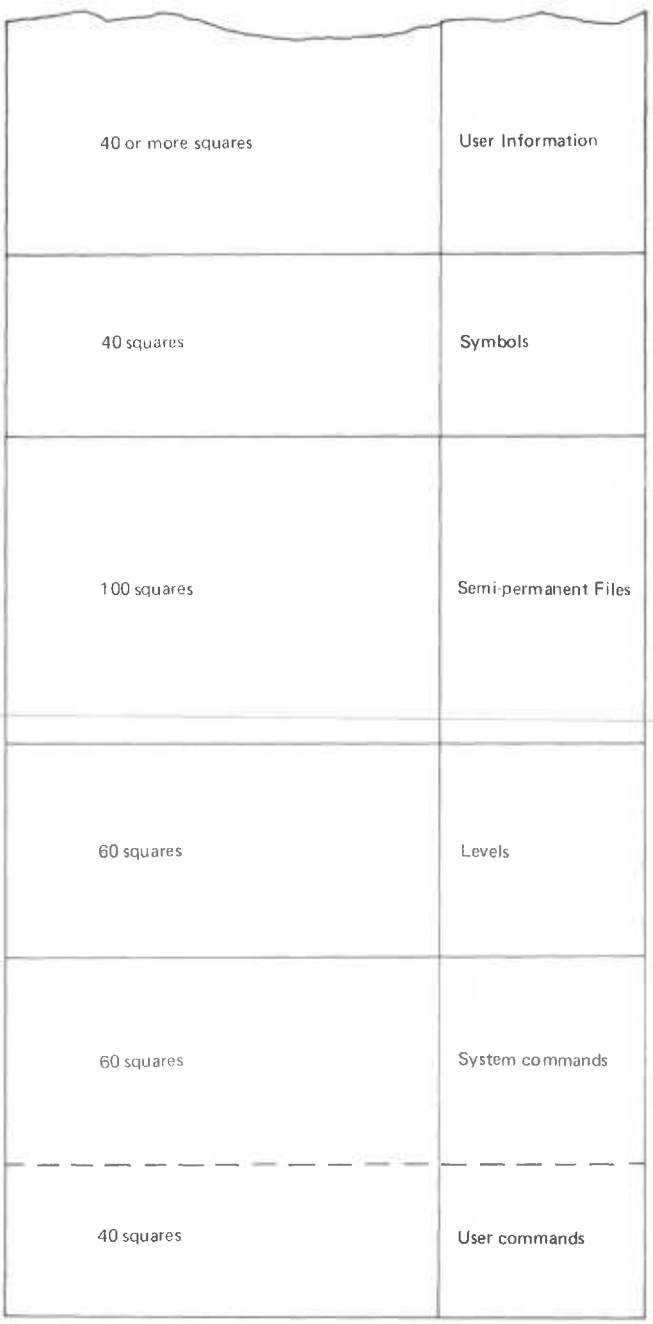


FIGURE 3. The menu system used with the CADMAC general drafting software

Some of the existing user modules in the CADMAC general drafting system are particularly relevant to p.c. mask work. For example the LEVEL system for structuring user data is useful for double-sided p.c. boards. The LEVEL system effectively allows the user to work in three dimensions. Furthermore, it allows a designer to break down a complex design or drawing into a series of simplified drawings which can be superimposed at any time to form the final design.

The MACRO facility is also extremely useful to the p.c. board designer. Standard items (MACROS) such as components, pins, pads, etc may be digitized and stored in the computer for subsequent use in creating the main design. The standard items may be stored in the MACRO bank as single units or multiple units in a cell structure. The MACROS may be used with considerable versatility. They may be scaled, chained, rotated, erased and even their internal dimensions may be changed.

4. *DIGITIZING PROCEDURE AND PRODUCTION OF DATA CHECKS*

The initial requirement for using CADMAC in p.c. board manufacture was to digitize a p.c. layout from a large scale drawing on temperature stable material, and then to produce data checking drawings together with a tape which could be used to control a high accuracy plotter for the generation of the p.c. mask.

The design used in the present test was for a Camac module, namely a clock pulse generator. The design was drawn at a scale of 4:1 on temperature stable material. The circuit on one face of the board had been drawn in red and on the other face in blue. The components were in black.

The design was placed on the working area of the digitizing surface of the table and secured with adhesive tape. The system was then initiated via the teletype. The computer responded by printing out a series of requests on the teletype so that the system could be set-up for the digitizing operation.

The first request was SET TABLE ORIGIN which was answered by digitizing a point marked by a cross in the bottom lefthand corner on the digitizing surface. This procedure sets the table origin to co-ordinate (0, 0), so that all points on the table will have positive co-ordinates.

The second request was SET DRAWING ORIGIN which was answered by digitizing a point on the drawing to fix its origin.

The third request was SET SKEW followed by the digitizing of two points on an axis of the drawing. This ensured that any skew in the axes of the drawing with that of the axes of the table is automatically removed by the computer as each point is digitized. The skew correction is important when data is used for either plotting the check drawings or generating the final mask. The reason for this is that most lines on a drawing are parallel with either the x or y axis and such lines can be plotted with greater speed and accuracy than diagonal lines.

The final request was INPUT SCALE which was answered by typing the required input scale for digitizing. The scale was set to 1:1 by typing 1.00. The LEVEL 1 square was then digitized and a drawing number written on the square. The system was then ready for accepting data.

The p.c. board layout had been drawn on the stable material in two colours, red lines representing one side of the board and blue the other. The p.c. design was divided into five parts for data storage and these five parts were to be stored under five level squares. LEVEL 1 was intended

CLEAR CURRENT FILE	DISPLAY CURRENT FILE	DUMP TO ASSIGNED PERIPH-ERAL	DUMP TO FILE	ADD FROM FILE	DISPLAY FILE	USE FILE AS MACRO	PLOT FILE			FILING	SYSTEM COMMANDS
SET INPUT LEVEL	OUTPUT ONLY COMMON LEVEL	OUTPUT ALL LEVELS								LEVEL HANDLING	
TRAILING ORIGIN	TABLE ORIGIN	CONTROL 90°	CONTROL 15°	ERASE SCREEN	RESET INPUT SCALE	RESET INPUT ORIGIN	RESET GRID FACTOR	RESET SKEW	RESET WINOOW	CONTROL FUNCTIONS	
LINE EDITOR	POINT EDITOR	MACRO EOITOR								EDITING	
MACRO STATUS INFORM-ATION	HAND MACRO	ANY ANGLE ROTATION	0°	90°	180°	270°	SELECT START POINT	SELECT END POINT	SELECT ALPHA-NUMERICS	MACRO STATUS	
IGNORE LEVEL	ADJUST DIMENS-IONS	CHAIN MACROS	SET SCALE							COMMANDS	
DEFINE AREAS	DISPLAY AREAS									USER COMMANDS	

FIGURE 4. The system commands in the menu of the general drafting software

for the main red circuit lines of uniform thickness, LEVEL 2 was for parts of the red circuit which were either wide lines, areas or tags. LEVEL 3 and LEVEL 4 were for the blue circuit corresponding to the functions of LEVELS 1 and 2 respectively. LEVEL 5 was to be used for all the MACROS (standard components).

The actual digitizing of the design was quite straightforward with little further requirement for digitizing menu squares until the completion of a level. The following procedure was adopted during digitizing: for vertical and horizontal lines, button 6, CONTROL 90 was pressed followed by button 1 at the beginning and end point of a line. Further end points could be digitized if the next line was joined to the first. When it was required to digitize a completely new line, button 2, BREAKLINE was pressed and the previous procedure repeated. The CONTROL 90 facility ensured that both horizontal and vertical lines were digitized such that they were accurately parallel to their respective axes. For lines at angles, the CONTROL facility was omitted.

The procedure adopted for the LEVEL 2 and 4 drawings, involving areas rather than lines, was to digitize the perimeter of each area.

The component layout which was designated as LEVEL 5 was generated from a series of MACROS. These individual MACROS are shown in Figure 5, and they are very simple to build and use. The building process consists of digitizing the actual MACRO and when this is complete it is stored in the MACRO store by digitizing a blank MACRO square on the menu and writing in a description in the square.

The MACROS are used by digitizing the appropriate square followed by digitizing a point on the drawing where the MACRO is required. Prompting information is given via the display to assist the designer in using a particular MACRO.

Verification of digitized data was performed in the first instance by viewing the display during the digitizing process; and in the second instance by producing a drawing on the plotting surface at the completion of digitizing a level. This procedure was adopted for each level and the check plot overlayed on the original design at the digitizing surface. Where an error was located, the area in question was first magnified on the screen. This function is performed by calling the WINDOW instruction, button 4, and then digitizing two points on either side of the area where the error might be. The display screen is then automatically erased and the enlarged area drawn on the screen such that the two digitized points lie on the extremities of the screen. If any error in digitizing is detected then the edit facility is called by digitizing EDIT.

Editing information is simple. A line can be removed by driving the cursor on the display and moving the digitizing pen until it is close to the required line. Button 5 is then pressed and the nearest line to the cursor will be lit up. If this is the line required for erasure then button 1 is pressed and the line is removed.

Lines may be added to digitized information using the FIND instruction. The exact co-ordinates of a point on a digitized drawing may be found by driving the cursor on the display so that it is within 1 mm of the required point. The FIND instruction is called by pressing button 7 and a message is flashed at the top of the screen either confirming that the point has been found or asking the user to try again. If successful then a line is inserted using the co-ordinates already found as the starting point.

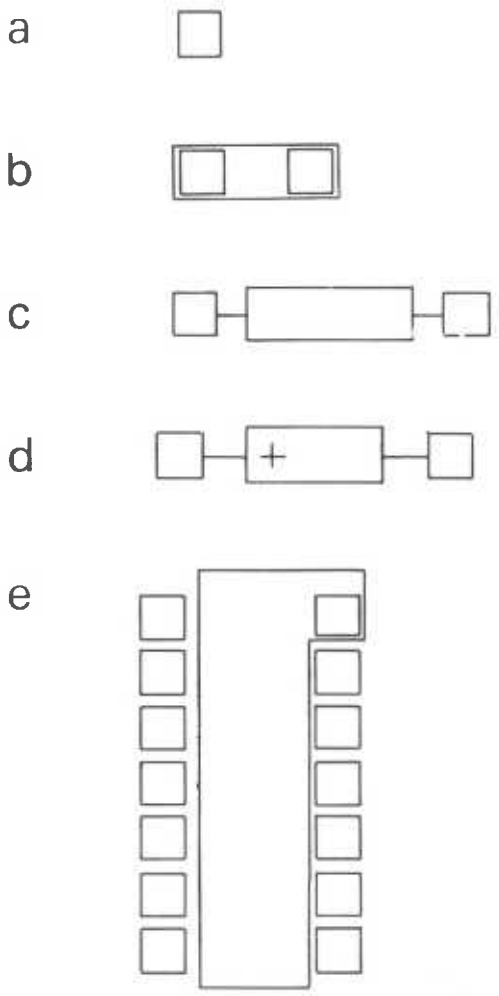


FIGURE 5. MACROS used in the p.c. design: (a) a pin; (b) capacitor; (c) resistor; (d) diode and (e) integrated circuit

The editing and find facilities are both easy and quick to use and are capable of dealing with complete MACROS as well as lines. The user is constantly assisted during this type of work with prompting instructions on the screen.

Finally, LEVELS 1 and 2 were combined and a paper tape output produced for the Kingmatic plotter, with the correct 1:4 scale reduction factor. The same procedure was adopted for LEVELS 3 and 4. LEVELS 2 and 4 carried further instructions so that enclosed areas would be completely blanked out when plotted.

Example plots of LEVELS 1 and 5 using CADMAC are shown in Figures 6a and 6b respectively.

5. MASK GENERATION

Printed circuit mask generation is performed on a high accuracy plotter such as the Kingmatic Model 1215.

The Kingmatic at Imperial College did not have a photo-exposure device and so the mask generation was simulated, with an accurate plot of the circuit on paper. The software required to operate and drive a photo-exposure device on mask generation is very little different from the production of an accurate plot with a pen.

A plot of the 'mask' for one side of the p.c. board, consisting of LEVELS 1 and 2, is shown in Figure 7.

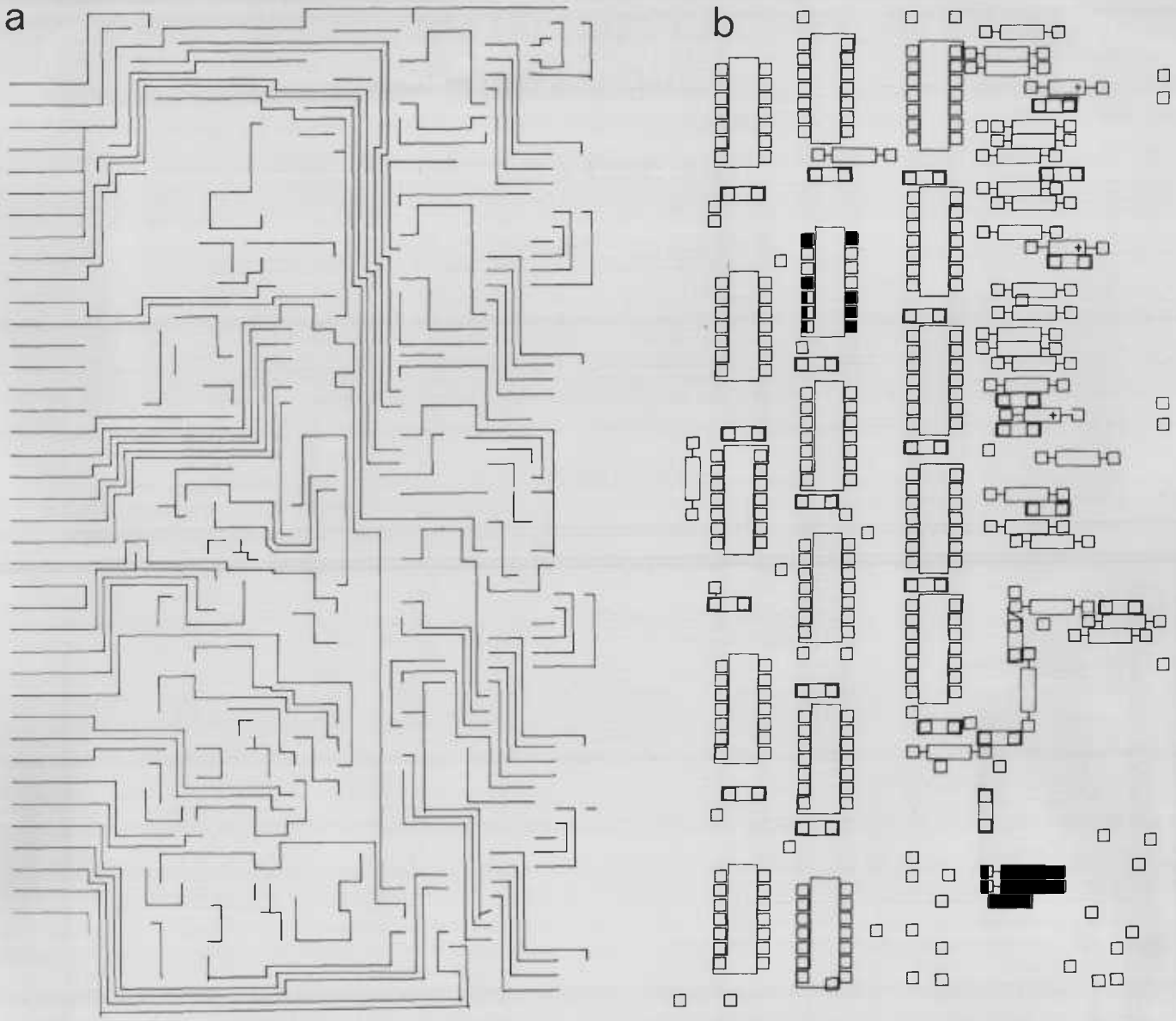


FIGURE 6. (a) CADMAC plot of LEVEL 1 – red lines on design. (b) CADMAC plot of LEVEL 5 providing the layout of MACROS

6. DISCUSSION

The CADMAC system proved to be most suitable for the present application, being easy to use, easy to check and correct for errors. The total digitizing time for the complete p.c. layout was 2.5 hours, and the time for data verification, including check plots, was 1.05 hours. The CADMAC system of similar configuration may be hired from a bureau at £25 per hour and thus the cost of producing correct tapes for driving a mask generating device was less than £90. A Kingmatic plotter may be used at a bureau for £35 per hour and for the present two masks this would have cost £60. Thus the complete task of producing the masks was about £150.

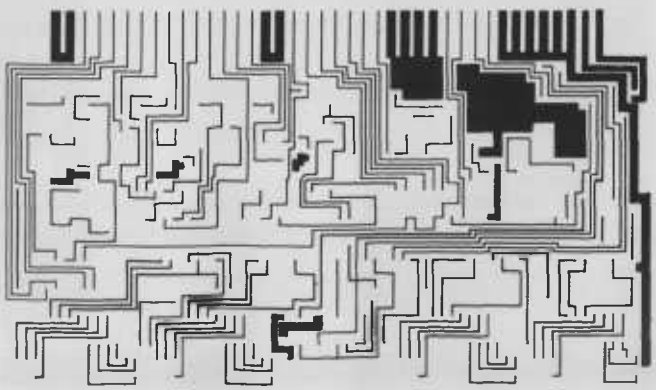


FIGURE 7. Kingmatic plot of LEVELS 1 and 2 – the simulated mask for one side of the p.c. board

It should be stressed that the CADMAC system could have been more effectively used in the complete task by using it during the actual initial design stage. This could result in further savings in time and cost.

REFERENCES

1 Kingmatic Model 1215 Draughting Table, *N.C. System News* No 2 (1972) Kongsberg, Norway.

2 Besant, C. B., Jebb, A. et al, 'CADMAC-11 – A fully interactive computer aided design system' *Comput. Aided Des.* Vol 4 No 5 (October 1972), pp 239–246.

Automated module placement and wire routeing according to a structured biplanar scheme in printed boards

G. ALIA, G. FROSINI and P. MAESTRINI
(Istituto di Elaborazione dell'Informazione, Consiglio Nazionale delle Ricerche, Pisa, Italy)

Heuristic procedures are proposed for solving the module placement and the wire routeing problems in printed boards with the primary objective of reducing the area covered by the conductor paths. A structured biplanar scheme has been adopted, into which the modules are placed on the nodes of an array and the conductor paths are realized by means of horizontal and vertical connections, printed on the different sides of the card and electrically connected via holes. The proposed method is well suited for implementation on a small computer.

(Received on 30th January 1973)

1. INTRODUCTION

This paper deals with the problem of the design automation of logical circuits. Specifically, the two problems considered are the placement of the modules on a printed card and the laying out of the wire among the module terminals.

A biplanar scheme has been fixed, into which the modules are placed in the nodes of a matrix, and the conductor paths are decomposed in horizontal (i.e. parallel to the rows) and vertical segments. Horizontal and vertical segments are printed on different sides of the card.

Heuristic procedures for solving the above mentioned problems are proposed, whose primary object is the reduction of the total area covered by the conductor paths and, indirectly, the total conductor path length. These procedures require very limited resources for implementation and are relatively fast, when compared to the alternative approaches discussed in the literature. The major results in the whole field of logical design automation are discussed in a survey paper[1] and in a number of specific papers dealing with the placement problem[2,3], the wire list determination[4–7] and the wire layout definition[8–10].

The reduced complexity of the approach proposed in this paper originates from:

1. The assumption of a fixed scheme for positioning and connecting the modules.
2. The decomposition of the classic placement and wiring problems into several cascaded subproblems.

Although by applying such constraints the degree of optimality of the result is generally smaller than would otherwise be possible, nevertheless in most cases the quality of the result is satisfactory and adequate in view of the limited cost of the method.

2. GENERAL DESCRIPTION OF THE WIRING LAYOUT SCHEME

It is assumed that the circuit to be realized consists of a number of interconnected multiterminal integrated modules that are to be placed on a printed card. The problem of decomposing a large network into minimally interconnected subcircuits, such that each subcircuit is realizable by a printed card, has been discussed in the literature and is not reconsidered here.

In the proposed scheme, the modules are placed on a printed card according to an array configuration, as shown in Figure 1. The modules occupy the nodes of the array (the number of the modules is assumed not to exceed the number of the nodes). For ease of discussion, assume that the terminals of each module are vertically aligned and that their number is the same for each module. The connections are realized by means of horizontal and vertical segments, printed on the first and the second faces of the card, respectively. Horizontal and vertical segments of the same conductor path are connected via holes (plated-through). The horizontal or vertical segments are localized in the strips between adjacent pairs of rows or columns. Different strips are allowed to have different widths, according to the number of segments they include.

Any conductor path between terminals of modules assigned to the same column (e.g. the one denoted by (a) in Figure 1) consists of a vertical segment connected to the proper terminals by means of horizontal connections, which are called 'terminal extensions'. The vertical segments are placed in the vertical strip on the right side of the column under consideration.

Any conductor path connecting modules assigned to different columns (e.g. the one denoted by (b) in Figure 1) consists of a horizontal segment and two or more vertical segments, each vertical segment connecting the subset of modules belonging to the same column. Each horizontal segment is realized in a horizontal strip between two appropriate adjacent rows.

Subcircuits realized in different cards communicate through the connector. In each card, the connector terminals are assumed to be aligned on the top side of the card, parallel to the rows. Any connection between a module and the connector is realized by means of a vertical segment.

It is easily seen that any arbitrary set of connections is always realizable according to the scheme in Figure 1, no matter how the modules are positioned on the card. In the proposed scheme the wire layout problem always has a solution, without any wire crossing, while the success of the classical approaches, e.g. the Lee's[8] or Akers's[9] algorithms, is not guaranteed. However, the degree of optimality of the solution to the wire layout problem is dependent upon the positioning of the modules.