415

THE APPLICATION OF COMPUTER AIDED DRAFTING TO MECHANICAL ENGINEERING DESIGN

by

David Thompson. BSc(Eng) A.C.G.I

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

Department of Mechanical Engineering
Imperial College
London
SW7 2AZ

August 1982

ABSTRACT

Rapid advances in computer technology throughout the last decade have brought small, powerful, and above all, cheap computers within the reach of all but the smallest business. As a result of this revolution the computer is no longer confined to its more traditional role of centralised data processing and is being exploited in many hitherto unforseen or uneconomic applications. One such area of rapid growth is that of computer aided drafting.

The thesis outlines the history of computer aided drafting from the early days of batch processing on mainframe computers, where data was normally prepared off line and submitted for subsequent analysis, to the modern interactive systems based on a minicomputer, where the operator communicates directly with the computer in a question and answer manner. A survey of both the current equipment and techniques employed and the implications of the introduction of this technology for business and employees alike is included.

Details are given of a fully 3 dimensional interactive graphics system developed for use in mechanical engineering design. This system, known as 'TIGER', whilst owing some features to early 3 dimensional animation or 2 dimensional engineering systems developed at Imperial College, combines these with many new features presenting a unique approach to the problems of entering, viewing and manipulating 3 dimensional information in a way consistent with modern engineering drafting requirements.

CONTENTS PAGE

ACKNOWLEDGEMENTS

I should like to thank everyone who has assisted me with this project, in particular:

Dr C.B. Besant and Dr A. Jebb of Imperial College for making it all possible.

Mr. A.J. Eagles of Imperial College for much valuable technical advice and support.

The engineers and draftsmen of Balfour Beatty Engineering for their support and suggestions.

1.0 INTRODUCTION

Imperial College has been well known for its works in the various aspects of computer aided design and draughting for a number of years, and pioneered the use of PDP 8 and PDP 11 computers in the early 1970's. A number of projects have been completed in association with varying branches of industry; site layout, shipbuilding and animation being typical examples. In several cases the work has been taken up by the associated company and further developed into a workable commercial format.

Almost without exception the above projects were of a two dimensional nature, although some success was achieved with '2½' dimensions, where the third space co-ordinate is constrained to lie on a number of fixed planes. The first fully three dimensional system was developed by Yi(14) for use in animation, enabling much of the tedious drawing work associated with complex cartoon movements to be performed by the computer. It was recognised that a three dimensional capability would have applications in general mechanical engineering draughting, the automatic production of isometrics being a typical example. Although a number of commercially developed graphics systems were becoming available their three dimensional capabilities were generally minimal or limited to one particular application, such as pipe routing, and their price rendered them beyond the reach of most small businesses. This project was undertaken with the specific aim of developing a small but powerful three dimensional computer aided draughting system of general applicability that required only a minicomputer to run, and was sufficiently flexible to enable it to be tailored to a number of different applications with the minimum of effort. It was further intended that a number of parallel projects should investigate the use of this system for finite element stress analysis, automatic production of tapes for numerically controlled machine tools etc. The three dimensional animation system developed by Yi was used as a basis, but required substantial modifications, amounting to almost a complete rewrite, to build in the facilities needed for engineering use.

Some two years into the project Imperial College was approached by Balfour Beatty Engineering Ltd. with a view to installing computer aided draughting facilities in their drawing office for the production of a variety of different types of engineering drawing. Development work subsequently continued at both Imperial College, and at Balfour Beatty when the necessary equipment had been obtained.

As a consequence of this, and the recent advances in computer technology, it was recognised that the exisiting graphics software, being of a single user format running under the DOS operating system, would be limiting in the commercial environment. To obviate this, conversion to multiuser format under the RSX-11M operating system was undertaken as part of the development process.

The following chapters outline the work carried out and the principles and conventions adopted, concluding with a number of practical examples from the drawing office. Also included is a survey of the hardware currently available, the reasons behind its introduction and some of the problems that may be encountered.

Throughout this thesis the graphics system is referred to as 'TIGER', being an acronym for Three-dimensional Interactive Graphics Routines.

2.0 THE ROLE OF COMPUTER AIDED DRAFTING IN THE DRAWING OFFICE

Computer aided drafting has been described as "The use of a computer based system for translating concepts or sketches into drawings suitable for use in manufacture, or for storing drawings or parts of drawings in a data bank to be available on call for modification, incorporation into a revised drawing, or as input to subsequent manufacturing processes". As such, a destinction exists between computer aided drafting and computer aided design, which is often not fully appreciated. A computer aided drafting system is structured to enable a designer or draftsman to produce a drawing more quickly and efficiently than is possible using conventional techniques, but has no built in design logical and is dependent in the skill of the designer to ensure that correct information is entered. A computer aided design system will typically have a level of design logic built in, and be able to choose between various options automatically, dependent on a number of pieces of key information provided by the operator. A typical example is the inclusion of a valve in a pipeline. The computer aided drafting system relies on the draftsman selecting the appropriate valve by some external process and telling the computer which to use. A computer aided design system will be able to select a standard valve automatically if provided with the necessary information, such as pressure, temperature, fluid etc.

The following chapter outlines the historical processes through which computer adided drafting has passed to reach its current level of development, and concentrates on a number of the benefits, problems and pitfalls that may be associated with the introduction of this technology in the modern drawing office. No attempt has been made to describe the ever increasing range of hardware and software that are available, a discussion of the former being the basis of chapter 3.

2.1 The History of Computer Aided Drafting

The last two decades have heralded a revolution in computer technology, in many respects equal in importance to the industry revolution of the nineteenth century. The computer benefits man in two distinct areas, the ability to perform complex calculations at great speed and the capacity to store large quantities of information for almost instantaneous recall. The exploitation of these different qualities is largely responsible for the distinction between commercial and scientific computing.

Computing techniques have become well established in the commercial world, and are frequently fully integrated into normal office procedures. For example, it is no longer a surprise, or a novelty, to receive a computer printed bank statement. In this application the storage, retrieval and modification of enormous quantities of customer records is of paramount importance, the requirement for calculating ability being very minimal. The computer enables the bank to streamline its operating procedures and provide a better service to its customers.

The use of computers in the scientific environment tends to be less well known, but this is partly a consequence of the lack of personal contact between the general public and such computers. Their principal function tends to be performing complex calculations on relatively small amounts of data, which often vary so much as to make the establishment of a data bank an impractical proposition. A typical example is the prediction of the performance of an internal combustion engine for a range of different design parameters, based on a number of well established laws and formula. It has long been recognised that such applications may place special demands on a computer system, not least being the method of data input and output. Whereas in most commercial applications printed output is normally all that is necessary, scientists and engineers traditionally represent results by means of graphs or drawings. This requirement led to the development of graphical display devices and subsequently graphical input devices, or digitisers. Computer graphics was born.

It is generally accepted that the first important advance in computer graphics took place at the Massachusetts Institute of Technology in 1963 with the development of a system known as "Sketchpad"(22). The system consisted essentially of a cathode-ray oscilloscope driven by a Lincoln TX2 computer, enabling graphical information to be displayed on the screen and then manipulated by a light pen. The system was interactive in nature, in that there was a continuous exchange of information between the computer and the operator, and this technique has become known by the name "Interactive Graphics". This is in sharp contrast to the "batch" mode of operation where all the information needed by a program is pre-prepared, then the program run and the output produced with no intervention from the operator. Early interactive graphics systems tended to be very expensive, as does most new technology at its outset, and necessitated considerable computing power. They were therefore only adopted in major industries, such as aviation, with high design costs.

Few major advances were made in these areas until the early 1970's when rapid technological changes brought about a swift advance in computer aided drafting techniques, many being based on the new breed of minicomputer that was both powerful and inexpensive. Whereas most work had previously been based on research projects, a number of companies emerged that both developed and marketed "turnkey" graphics system to meet specific user requirements. The number of such companies has continued to grow, although a few market leaders have envolved whose names have become synonymous with computer aided drafting and design. Today, new systems appear almost weekly in the journals, as does new hardware. A subjective appraisal of the software is almost impossible since it tends to be diverse and frequently tailored to particular applications or hardware. Furthermore, for obvious reasons, companies tend to be unwilling to divulge the details of their own systems, and information contained in typical advertising leaflets is generally scanty and unspecific. An appraisal of hardware is somewhat easier, and the following chapter is devoted to such a survey.

2.2 The Need for Computer Aided Drafting

There is a considerable conceptual jump from the early graphics system capable of little more than displaying results pictorially to the modern interactive drafting system with sophisticated input and output techniques. This transistion has come about to fulfill demands made by industry, who see computer aided drafting benefiting them in a number of areas and for a number of different reasons.

By combining man and machine in this manner the better qualities of each may be exploited. The computer is logical and can be relied upon to make consistent choices, hold large quanties of data for long periods without degredation, and make calculations without error. Man is essentially illogical in that a choice is often made intuitively, past experience may lead him to choose what would otherwise appear the wrong course of action. Furthermore, mans capacity for holding large quantities of information is poor and likelyhood of error is high. An ideal computer aided drafing system would therefore be programmed to perform all the logical functions based on a series of instructions given by the user. This should result in a problem solving "team" capable of performing more efficiently than either the man or computer alone.

The demand for CAD is naturally financially motivated and stems from a need to use resources in the most efficient way possible. This is particularly true of recent years when labour costs have escallated rapidly, whereas computer costs have reduced, so making a computer aided drafing system a more attractive proposition. Many companies are taking on such systems in the hope of reducing their drawing office staff, or simply enabling the current staff to handle more work. The latter is particularly important in an expanding company because there is currently a real shortage of skilled drawing office staff, and computer aided drafting may be the only solution.

Productivity increases are difficult to quantify since they depend very much on the type of work being undertaken. Claims vary from 2:1 to about 30:1, but it is generally accepted that between 2 and 5:1 is about average. Higher values are achieved when drawings are very repetitious, or use large quantities of standard components. The design of a multi-storey building is a typical example, where the basic floor plan need only be drawn once, then duplicated for each floor before any minor variations are added.

The ability of a computer aided drafting system to use a central store of data is frequently important, and has a number of benefits. It is very easy to impose uniform design standards, and drawing quality will be consistent irrespective of the draftsman who produces it. A typical drawing will contain large quantities of text, and the way in which this is added can alter the overall appearance of the drawing. With computer aided drafting a uniform quality is guaranteed since the character set is generated internally by the computer. This concept extends further to the use of standard parts since the draftsman will call them back from a library and they are certain to appear in the same form on any drawing. Furthermore, updates may be performed automatically on a whole series of drawings if a particular component is changed, and in any case modifications to a single drawing may be quickly and simply applied, sometimes taking only minutes as compared with days for the manual equivalent. Whenever a drawing is updated a new 'master' copy may be run off on a plotter, so resulting in a perfect copy which shows no sign of correction and, unlike drawings that have been on a drawing board for some weeks, is perfectly clean. This can be important in that it impresses present and potential customers and may lead to the company gaining further work.

As well as extracting information from a data bank with which to build up a drawing, a computer aided drafting system is capable of creating a second data bank in parallel with the drawing, which normally takes the form of a parts list, material schedule etc. A simple illustration of this concept is the layout of an electronic circuit.

Symbols for resistors, capacitors etc. are taken from a library, and the number and values used stored as the drawing is built up. When complete the computer can quickly and simple output a list of all components used for that particular circuit, which may be used as a basis for ordering, assembly etc. This can be very important at the tender stage of a contract to enable the company to produce its tender in a very short time and at minimum cost.

The above may give the impression that computer aided drafting is the answer to all drawing office problems, but regrettably this is still very far from the truth. The following section outlines some of the considerations that must be made when installing such equipment, and a few of the problems that may be encountered.

2.3 Choosing a Suitable System

The use of computer aided drafting and design equipment in the U.K. is still somewhat patchy, and by no means as widespread as in other industrial nations such as the USA, Japan and West Germany. This can be partly attributed to their much higher labour costs, and concentration on high technology industries, but is also due to a lack of awareness in the UK, and the traditional British reserve hampering the introduction of this new technology. At the current stage of development the problems of choosing a suitable system are considerable, even for those experienced in the field, and for a new company entering the market for the first time can be extremely daunting.

A great deal of mythology exists about the capabilities of drafting systems in general, partly as a result of well rehearsed demonstrations and televisions "tricks" which make it all look very easy and gloss over the hours of preparation that have gone into a demonstration lasting a few minutes, and frequently the phenomenal cost of the equipment involved. Problems will generally be experienced if these are used as the basis on which a system is chosen, since demonstrations only outline what the system can do, not what it can't, and the requirements of the average drawing office will soon make the latter only too obvious. Detailed specifications of commercially available systems are difficult to obtain, and frequently the only way of getting past "sales talk" is to speak directly to other companies who already have the system in operation, which again may prove difficult if they are in direct competition.

When choosing a computer aided drafting system for a particular application it is extremely important that the potential user identifies clearly in his own mind exactly what the system will be required to do before attempting to match this to a particular system. Frequently insufficient forethought is given and users find themselves with a system incapable of performing the function they require, but very good at performing functions that are not needed. There is a strong temptation to buy a very sophisticated system, but frequently this will have capabilities beyond those required, add unnecessary complication to its operation, and eventually be difficult to justify economically. The best approach is to buy just what is needed and use it to maximimum efficiency. Since the requirements of a company are generally very personal it must be accepted from the outset that no system is likely to be absolutely ideal, and a certain amount of tailoring will be required to meet specific requirements. With large commercially supplied systems this may be difficult to achieve and become very expensive, since the user will normally be obliged to ask the supplier to perform the necessary modificiations due to the unavailability of source code. An alternative approach is to develop an entire system "in-house", perhaps in co-operation with a research organisation. This guarantees that it will meet the companys exact requirements, providing an adequate specification is initially drawn up, but obviously results in a significant development time between initial conception and productive use. Assuming this delay is tolerable the resultant system should more closely match the company's needs, and be far simpler to modify in the future to incorporate any new features that may be required. In some fields this will be the only course of action. Currently a large number of 2 and "2½" dimensional systems exist, but little work has been done in full 3 dimensions. Claims have been made for the capabilities of a number of American systems, but to a large extent these are still unsubstantiated. It is to fill this gap that this project was undertaken.

Suitable software must be matched by compatible hardware, and the choice of such equipment can be bewildering. Technology is advancing so rapidly that a large range of very similar equipment is avilable, and whatever choice is made it is likely to be obsolete almost before it is installed. An effective system must be based on an adequately powerful computer. A number of the processes performed, such as three dimensional rotations, require considerable computing power if they are to be completed within a reasonable time. An important requirement of any interactive graphics system is that it must have a rapid response. Although a slow response may be tolerated by the novice user, it will prove a source of annoyance to the experienced operator and will cause frustration and a loss of enthusiasm in using the equipment. The design of a "user friendly" workstation is very important to the smooth running of the system. The layout should be such that each item of equipment is easily visible and any keyboards or digitisers accessible without undue movement of the operator. An ideal arrangement appears to be to mount the graphics screen above the digitiser, directly in front of the operator. However, problems result from this arrangement if a large digitiser is employed and it is to be used at a steep angle.

In order to justify the capital cost of such a system some productivity increases or manpower saving are obviously necessary. As previously stated, it is generally accepted that productivity factors of between 2 and 5:1 can be expected on average, depending on the type of drawing being produced. It is therefore a relatively simple matter to calculate break even points depending on the usage and original cost. Estimates vary, but it has been suggested that up to 5 years may be required, and in drawing offices of less than about 20 staff this situation may never be reached. A possible solution is the use of a system on a bureau basis, although this is seldom satisfactory due to speed of response and the limited transmission rates along post office lines.

Whatever system is finally chosen it must integrate easily within the existing drawing office procedures. It is foolish to imagine that this can be done without any changes, but these should be minimal in order to avoid upsetting the smooth running of existing company practises. The following section is devoted to a number of the problems that may be encountered in these circumstances.

2.4 The Effects on the Workforce

The success of any new technology is dependent on not only its capabilities but also its acceptance by those who will be involved in its use. Computerisation tends to be an emotive issue and many people feel it will ultimately lead to loss of jobs and the disappearance of traditional trades. A number of union/management disputes have already arisen over this, notable computerised type setting in the printing trade, and if these are to be avoided careful planning and forethough is necessary. Needless to say, union and management views on this subject tend to differ and a workable compromise must be achieved if maximum benefits are to be reaped by both sides.

Unions see a number of potential dangers to their members, most of which apply equally to any new technology as to computer aided drafting. They fear a continuation of the fragmentation of skills that began in the 1930's, such as the job of a designer being broken down into that of stress man, metallurgist, draftsman etc. This, it is felt, leads to lower job satisfaction since no one man can see the job through from start to finish. It is feared that as workers became more specialised they become akin to a part in a machine, and will eventually be scrapped when that machine becomes obsolete. Furthermore, the useful working life of staff is likely to be short, with a high burn up rate which the use of computer techniques will bring. This can be attributed to the reduction in the amount of mundane reference type work performed by a draftsman, and hence the higher proportion of his time spent on the decision making process. It is feared that only a small age bracket will be suitable, and once beyond that bracket staff will suffer a career de-escalation with a resultant lowering of status, and that traditional skills such as tool setting will disappear in the wake of numerically controlled machine tools. To their advantage, it is recognised that computers represent high capital investments and are extremely vulnerable to strike action by a small number of people. A number of trade unionists are aware that other nations, such as Japan and the USA, are using computer technology to great effect, the Japanese motor industry being one example, and if UK industry, and hence jobs, is to survive we must maintain competitiveness by superseding older labour intensive processes.

Employers look upon new technology in the broader spectrum of its benefits to the company rather than the effect on individual employees, although all but a few take the latter into consideration. Recent price trends in computer technology have allowed companies to move away from equipment with a high cost to user ratio, so reducing the pressure to utilise the equipment to the maximum to recover the initial investment. It is now recognised that staff must have time to think since excessive pressure will only result in unnecessary errors. It has been suggested that staff often welcome the introduction of new technology because much of the tedious work is eliminated, leaving greater time for more satisfying work. By sharing one workstation between several operators, maximum use may be made of the equipment without undue stress on staff, who can spend short periods on the equipment and longer periods preparing further work elsewhere. Frequently operators may make helpful comments about how the use of the equipment may be improved. Some employers have suggested that, far from leading to undue specialisation, quite the reserve is true. By using the special capabilities of the machine, such as finite element stress analysis, engineers or draftsmen with an all round knowledge can perform very specialised functions which they would otherwise find impossible. This leads to a broadening of knowledge and a greater overall understanding of the processes involved. With this in mind universities and other training establishments are tending towards less specialised engineering courses including theory, practical experience and a knowledge of the humanities, with the specific aim of training engineers to have a broader outlook on engineering and life in general.

3. COMPUTER AIDED DRAFTING HARDWARE

Throughout this thesis frequent mention is made of a number of pieces of computer aided drafting equipment on the assumption that their function is understood. This chapter describes the principal items of equipment and the current state-of-the-art in their development, starting with the basic computer which is an essential item in any system. Naturally, the combination of equipment employed for a particular system will depend on the users requirements, machine compatibility etc, and no attempt has been made to suggest a 'best' combination.

3.1 The Basic Computer

The word computer is often misapplied to refer to a complete computer system, which can be broken down further and considered as a number of discrete, and frequently interchangeable, units.

a) The Central Processor

The heart of any computer is the 'central processor unit', or CPU as it is generally known. This is responsible for performing not only the basic computation processes of addition, subtraction etc., but acts as a controller for all the other pieces of equipment, or peripherals. An important part of the CPU is the core storage which is used to hold all programs and data while they are executing, and has until recently been a very expensive item. Recent advances have enabled computers to be fitted with large quantities of core for very modest prices. The amount of core available is generally described in terms of 'k', which in this context refers to 1024 words or bytes. For example 96kW is 96 x 1024 words. The definition of a word varies from computer to computer but is normally a fixed number of bits, or binary digits, these being the smallest unit of storage available, capable of taking only two values each, either 0 or 1. A word will typically be composed of 8, 12, 16, 32 or more bits, so enabling a word to assume 2^n different values, where n is the number of bits per word.

b) Bulk Storage Devices

An essential quality of any computer is the ability to store large quantities of data, comprising not only the programs to control the computer, or 'software' as they are generally known, but also any databases that may be set up, such as the drawings produced by an interactive graphics system. The quantities involved preclude permanent storage in core and necessitate the use of some form of bulk storage device.

The most common bulk storage device in use today is the magnetic disk. The principle of operation is a cross between the gramophone record and the tape recorder, being a flat rotating disk covered in magnetic material that is accessed by either a moving head that traverses its surface radially or by a number of such heads at fixed locations. The variety of disks available is considerable, but the principles of most are similar. More conventional designs employ stiff disks that are manufactured to very close tolerances, recent designs employ 'floppy' disks, which rely on aerodynamic effects to keep them correctly placed in relation to the heads. The great advantage of disk storage is that it provides random access to data, enabling information on any part of the disk to be accessed about instantaneouly by moving the head to the appropriate location, and very fast data transfer rates, typical access times being of the order of 50ms and transfer rates approaching 1MB/second (1MB = 1024 x 1KB).

An alternative form of storage is magnetic tape, which behaves very much in the same way as a domestic tape recorder. The capacity of tapes for storing data is large, comparable to medium sized disks, but the access times and transfer rates do not compare with disks. A magnetic tape is a sequential device, so in order to access information at any point on the tape it is first necessary to read past all preceding information, which on a large tape can take several minutes. When the information is found data transfer rates can be of the order of 120KB/second. Offset against the disadvantages of sequential access is the comparitive cost of magnetic tape, sometimes as much as one twentieth of the cost of a similar capacity disk, and its size, which is considerably smaller than such a disk. Their comparitive merits and drawbacks mean that disks are generally used for current information to which frequent and rapid access is essential, whereas tapes are used for archive or backup storage to which access is rarely required.

Recent years have seen the introduction of cassette tapes in the computer industry. In general their capacity is very limited and data reliability less certain then the above devices. They play an important role at a personal level, enabling programmers to store and carry their programs with them, but have made little impact as mass storage devices, despite their very low cost.

Other storage devices include paper tape and cards, but these tend to be very bulky and unsuitable for storing either large quantities of data or that to which frequent or rapid access is desired. These media are frequently used for off-line, i.e. remote from the computer, preparation of programs or data due to the ease with which they may be modified by hand. Their use in interactive graphics is very limited, and are increasingly losing popularity in favour of on line data preparation and editing directly from a terminal connected to the computer.

c) Terminals

Modern computer systems enable large numbers of terminals to be connected to the same computer and give the appearance to each user that the computer is dedicated to him alone. This is made possible by the speed of modern computers and operating systems using principles known as time or resource sharing, allowing users access to all the necessary resources on a rotation basis. Terminals are generally of two basic types, hard copy or VDU. The former are similar in operation to a modern electric typewriter and produce hard copy output on computer paper. VDU's, or 'Visual Display Units' produce no hard copy output but display all their information on a screen similar to a conventional television. This form of terminal has the advantage that running costs are low and very high display rates possible, up to 960 characters per second being typical. This will often be more important than the hard copy output, although frequently modern VDU's have an output that may be connected to a fast printer if a copy is needed.

d). <u>Printers</u>

The conventional hard copy output device is the line printer, normally capable of printing lines of up to 137 columns at rates up to 1000 lines/minute or more. This is achieved by having a print head for every character position on a line and printing the whole line is one operation. The type of printer employed will depend on the volume of output that may be handled. Interactive graphics systems invariably require small quantities of hard copy output and a character printer may be adequate, which behaves like a fast terminal and prints each character individually. This is normally achieved by either a dot matrix head which constructs letters from discrete dots, a golf ball head as found on many modern typewriters, or a daisywheel head, where the letters are mounted on a rotating head formed into a pattern similar to a daisy flower and are pressed on the paper individually by some form of hammer mechanism.

3.2 Special Equipment for Interactive Graphics

The above outlines a number of peripheral devices that may be found in any computer installation, and indeed play an important role in a computer graphics system. In this application, as in any specialist field, a number of additional peripheral devices have evolved to provide the specialist functions required for graphical input and display.

a) Graphical Displays

Perhaps the most common specialist device is the graphics display, being essentially a terminal type device with the capability of displaying not only alphanumerics but graphical information such as lines, circles etc. These devices may or may not have a keyboard attached to enable feedback from the user to the computer. Graphics displays can be broadly categorised into two different classes, refresh and storage display.

i) Refresh Displays

A refresh display is similar in operation to a domestic television in that the screen is composed of a low persistance phosphor which must be refreshed repeatedly in order to maintain a visible image. This must be carried out sufficiently frequently to avoid flicker becoming obvious to the observer, a process which should be carried out locally to avoid excess load on the host computer. A typical device of this type will contain a microprocessor and enough storage to buffer a entire picture locally. Once the picture is transmitted to the display by the computer and placed in local storage the microprocessor is responsible for monitoring the necessary display procedure. The principle of storage and display may be either the conventional television type raster scan format, where the picture is composed entirely of dots scanned in horizontal lines, or a discrete line type format where each line on display is stored and then displayed from start to end in one operation by a beam capable of drawing in any direction rather than just horizontally as in the former method. This has the advantage that less processing of a line is needed than that required to break it down into a raster type format, but suffers from the disadvantage that the database stored varies in size according to the amount of data displayed and the speed of display is proportional to the amount of data, neither of which is true for raster scan format. In either case modern refresh displays still tend to be 'black and white' devices, being capable of generating either darkness or one level of light, although colour displays are rapidly gaining popularity with reducing prices. Exhaustive test have been carried out by many terminal manufactures, which apply equally to graphics and alphanumeric displays, to determine the best colour for the screen for maximum visibility and minimum operator fatigue. Most have come down in favour of green, particuarly for graphics displays, although white is still popular. Feedback from the user via such a device is normally through either a conventional keyboard, a light pen used to point at specific items on the screen, or a joystick to control movement and rotation of an internally stored picture.

ii) Storage Displays

A second category of graphics screen is the direct view storage tube, which as its name suggests 'burns in' the image on the screen, which then remains on display without the need for it to be constantly refreshed. The principle of the device is similar to a standard electrostatic cathode ray tube except than an addition grid electrode and flood gun are incorporated. The grid electrode may form a part of the screen surface, so resulting in a bistable phosphor. The image is created by secondary emission due to election bombardment. The grid electrode is negatively charged before display commences, but when the beam of electrons trace out the picture this becomes positively charged at any point traversed by the beam. This has the effect of accelerating electrons from the flood gun through onto the screen, causing the phosphor to glow and the picture to become visible. The picture will remain held until erased, which is achieved by recharging the grid electrode to its all negative state.

Storage displays have the advantage that no processing is necessary to maintain the image, and their operation is considerably simpler than an equivalent refresh display. Greater resolution can normally be achieved with storage devices, but selective erasive of a part of a drawing is difficult, a process which can be readily achieved with a refresh type display. Until recently the market has been dominated by storage devices, due to their price and functional advantages over refresh displays, but modern advances have made refresh displays both more economically viable and capable of giving the resolution and picture quality required. Colour screens are becoming more readily available that will undoubtedly add a new dimension to future graphics systems.

b) Digitisers and Tablets

A second group of peripherals intended specifically for graphics comprises digitisers and tablets, of which the latter is normally a small, low resolution version of the former. The digitiser is the equivalent of the draughtmans drawing board and is used by him to communicate positional information to the computer. Drawing is performed, not with a pen or pencil, but with a device known as either a puck, cursor, stylus, or a number of other names. The name 'stylus' has been adopted throughout this thesis to avoid confusion with the cursor displayed on the graphics screen, of which a full description is included in chapter 5. The position of the stylus on the digitiser is registered extremely accurately, normally by means of some mechanism in the digitiser surface detecting an electric field emitted by a coil in the stylus. Early digitisers were mechanical devices employing a moving carriage driven by servo motors that followed the stylus as it was moved. The position could be determined by means of optical encoders fitted on each of the two axes. These devices were frequently capable of receiving data from the computer and functioning in a plotting mode as an output device. Modern digitisers tend to be of solid state construction with a grid of wires embedded in the surface. Suitable electronics enable the position to be determined to great accuracy by interpolation, even though the spacing between wires may be as much as 25mm. A typical resolution is ± 0.005mm. The stylus will normally be fitted with a number of push buttons to enable the user to register points. The digitiser will output both the button pressed and the location. In an interactive system this will normally be fed straight to the computer, but where the situation demands data can be recorded off line by connecting the digitiser directly to a mag-tape unit, card punch, or any other such device.

c) Plotters

The ultimate aim of any graphics system is to produce a hard copy drawing that may then be used in much the same way as its hand drawn counterpart. Excluding a photographic copy of the graphics screen, which is normally too small and of too poor a resolution for anything but proof checking, the normal method of producing such a drawing is on some form of mechanical plotter. The types currently available may be broardly categorised into 3 different groups, each with its own merits and shortcomings.

i) Drum Plotters

The drum plotter is the oldest of the three types, and is still the most common in general use. This consists of a drum driven by a stepper motor over which is fed a continuous roll of paper. The paper is normally perforated along the edges in a similar fashion to conventional computer listing paper, engaging in sprockets at each end of the drum, enabling it to be moved precisely in either a forward or backward direction. A pen holder carrying up to four pens is mounted on a gantry above, and parallel to the axis of, the drum. A second motor drives the pen holder along the gantry, which when combined with the movement of the paper allows the pens to be placed over any point on the resultant drawing. The pens are moved into, and out of, contact with the paper by D.C. solenoids.

This type of plotter is popular for several reasons, not least that it is the cheapest of the currently available units. A high degree of unattended operation is possible since a large roll of paper is employed, allowing every drawing to be plotted consequitively with little or no operator intervention, with the exception of periodic cleaning and replacing or refilling of pens. Modern drum plotters will accept a number of different pen types allowing not only different colours or line thicknesses to be used, but also different paper finishes which have particular pen requirements. The great disadvantage with a drum plotter is that it cannot accept pre-printed sheets, and hence all title blocks etc. must be drawn on every drawing by the plotter. This is not only time consuming, but in many applications, such as consultancy, the range of different formats may be very large, and clients may demand that drawings be done on their own paper. A lesser disadvantage of the drum plotter is that where many small drawings are to be produced paper wasteage may be considerable, since no matter what size the drawing the user is compelled to use the full width of the roll. A solution to these problems may be found in the following design.

ii) Flat Bed Plotters

This type of plotter takes its name from the design, incorporating a large horizontal bed onto which paper is placed and over which the pen moves. Like the drum plotter a pen holder moves along a gantry to provide movement on one axis, but in the flat bed plotter the paper remains stationary and the gantry moves over it to achieve movement in the opposite axis. Paper is loaded onto the plotter in individual sheets, which may be of any size up to the maximum area of the bed, which is normally AØ, and held firmly in place by either electrostatic charge or suction through small holes in the bed. This has the advantage that it allows the user to load preprinted sheets, but requires more operator attention since a new sheet must be loaded manually for every drawing, as opposed to the automatic wind forward achieved with drum type plotters. Typical flat bed plotters are normally more expensive than their drum counterparts due to the greater complexity of operation, perhaps the biggest problem being the need to overcome the considerable inertia of the moving gantry to achieve satisfactory results at the beginning and end of lines.

iii) Electrostatic Plotters

The electrostatic plotter is based on a completely different principal from the aforementioned designs. Both the drum and flat bed plotters are based on a moving pen principle, but electrostatic plotters are of a raster design, producing images on sensitised paper in a similar fashion to a television picture. Every line is broken down into a series of dots at the appropriate locations on the paper by software, which are printed by moving the paper under a light bar. This principle has a number of advantages and disadvantages compared with the conventional plotters.

The most obvious advantages is the speed at which the plot is produced. The time taken for a pen plotter to produce a drawing is directly proportional to the complexity of the drawing, whereas the electrostatic plotter will produce a drawing of any complexity at the same speed, this being governed by the rate at which the paper passes under the print bar. Since plotting can often be a bottleneck in a busy drawing office this may be an important consideration, outweighing the small software overhead necessary to preprocess a drawing into raster type format. Regretably, electrostatic plotters do suffer from a number of drawbacks. Special sensitive paper must be used, so precluding the use of preprinted sheets and limiting the number of different surface finishes available. On top of this the special paper can be very expensive, as is the plotter itself, the disadvantages of which are fairly obvious. A good electrostatic plotter will have a resolution of approximately 200 dots/inch, which although acceptable for less demanding work can be inadequate for engineering applications and results in very obvious 'stepping' of lines, particularly those very nearly parallel to one of the axes.

Furthermore, although different line thicknesses can be achieved, the technology to produce colour plots is still being developed, which for applications such as mapping can be an overwhelming disadvantage.

All three types of plotter are normally available with two different modes of operation, referred to as 'on line' and 'off line'. For on line operation the plotter is connected directly to the computer and receives the information to the plotted under the control of software running on that computer. For off line operation the plotter is independant of the computer and reads the data that is to be plotted from a bulk storage device, normally a magtape unit, connected directly to it. The tape that is read must obviously have been produced by the appropriate software on the computer at some earlier time. The choice of which system is used is governed by a particular companys requirements, but in general the on line mode is gaining popularity, due chiefly to the increasing power of computers to handle plotting in a background mode, and the extra capital expenditure required with off line operation to provide an addition tape unit. Furthermore, off line plotting requires a higher level of supervision to ensure that the correct tapes are loaded and unloaded from both the computer and the plotter.

3.3 The Equipment Used For TIGER

Much of the early development work was carried out under the single user operating system at Imperial College, and then under the RSX-11M multi-user operating system both at the college and at Balfour Beatty Engineering Ltd. in Sidcup, Kent. Where possible compatible equipment was used at both sites in order to keep machine dependancy problems to a minimum. The principal differences were due both to advancing technology; the Balfour Beatty PDP11/60 was a more recent machine than the PDP11/45 at the College, and the need for increased power and storage capacity in the commercial environment. In addition to a number of standard peripherals, such as magnetic tape and a line printer, both machines were equipped with a flat bed plotter and graphics workstations.

Much research into plotter design was carried out at Imperial College, and hence during this project three different flat bed plotters, designed and built at the College, were available at differing times. Appendix A contains a description of the interfacing of the second of these plotters, which was subsequently replaced by one built as a separate project. The design of the plotter described was subsequently sold to Computer Instrumentation Ltd. and developed into the commercially available 4/74 with on-board microprocessor control. It is this plotter that is currently in use at Balfour Beatty.

The style of workstation has changed considerably throughout the duration of the project, and again reflects changes in technology. Early work at Imperial College was carried out on a mechanical DMAC digitiser and Tektronix 611 storage screen, both of which were subsequently replaced with a Talos solid state digitiser and Tektronix 4014 graphics screen respectively. This configuration necessitated using the same screen for both messages and the drawing, and as part of the multi-user conversion a simple alphanumeric terminal was added to the workstation to receive typed input and issue system messages and prompts. Two workstations of this style were initially installed at Balfour Beatty, and later two more advanced stations employing raster graphics screens with dual graphics/alphanumeric capability. Originally each device of a user station was connected to the host computer by a separate line, which resulted in a number of operational problems. To overcome these local microprocessors were developed to run at each workstation. A full description of these devices and the software to control them is included in chapter 4.

4.0 THE TIGER SYSTEM CONCEPTS AND STRUCTURE

The following chapter is intended to give an insight into the basic structure of the graphics system including data handling, task swapping, and 3-D input and display. Each function is described in general terms and in most cases specific programming details have been deemed to be beyond the scope of this document. Whilst describing individual concepts this chapter does not attempt to explain how they are combined together to form complete graphics modules. This aspect is covered in some detail in the succeeding chapter.

4.1 The Operating System

Much of the development work for TIGER was under the DOS/BATCH operating system, this being a single user executive supporting up to 32K words of core. This imposed severe limitations, not least that only one operator could use the machine at any time, and in a production environment it precluded simultaneous production and development work. In the later stages of the project the graphics system was converted from its largely obsolete single user form to allow multi-user operation under the RSX-11M operating system, which it was hoped would obviate the above problems and in general result in much greater flexibility.

Graphics systems of this type are well suited to multi-user operation, since a large part of the time is spent waiting for operator action, and so imposing practically no load on the computer. It had been found necessary to enhance the DOS operating system substantially to provide the facilities needed, but RSX-11M, being a more sophisticated executive, has so far proved sufficiently comprehensive to be used with very little modification, and none that result in any noticeable change for other users, who may be performing normal program development work. This thesis is in general written on the basis of the multi-user operating system, although where significant differences in concept exist between it and the single user system the latter is described to illustrate these differences.

4.2 The Overlay System

The software for a large graphics system is obviously very extensive, but generally is modular in structure and can easily be subdivided into individual sections of code performing specific functions. This property is used to advantage in the TIGER system since core restrictions limit the amount of code that may be in memory at any one time. The solution to the problem is significantly different under the two operating systems used during the project, and both will be described to illustrate this difference.

4.2.1 The Single User Overlay System

Under the DOS executive the user is limited to 32K words of core, and a true overlaying system is necessary where only the currently active module is in core, the others residing on a bulk storage device, such as a disk, until they are needed. The overlay system simply provides a means of loading the appropriate module into core and starting its execution.

By definition, when a new overlay is loaded from disk the overlay currently in core is overwritten and thus completely lost. In order to communicate information between overlays it is therefore necessary to have some reserved area of core which is not affected by this overlaying procedure, and that can be accessed by any overlay if necessary. This area typically contains relevant system information such as scale factors, grid sizes etc. This facility was provided by the standard DOS overlay system, but regrettably this had already been proved to be too slow and limited for graphics applications, operating on a tree type structure where the next overlay to be executed must always be prespecified when the task is build and it is not possible to jump from one overlay to another further down the tree, or one on a different branch. Since the sequence of operation of the graphics system is controlled at random from a menu area it is important to be able to jump between overlays in any order or direction and with sufficient speed so as not to produce irritating delays for the user. A faster and more versatile overlaying system was written by Hamlyn [(9)] to operate with version 7 DOS, but even this had shortcomings in that core useage was inefficient and overlay building needed a certain amount of care. This was rewritten during the course of this project to improve these shortcomings and to make the overlay librarian behave more like a standard DOS utility program. For example, the command format was rationalised to conform with DOS conventions, additional commands incorporated to expand the directory management capabilities to include deletion and more comprehensive listing options, and I/O modified in order to allow its use from any UIC. A comprehensive error reporting system was incorporated, which was absent in earlier versions, and by use of larger areas of core packing times, to remove wasted space between overlays in the library, were reduced by the order of 95%. The system provides for a resident area at the top of core which is unaffected by the overlaying process, and an overlay area occupying the rest of available core. Overlays may be written in Fortran IV, Macro 11 or any other language for which a compiler is available. The overlay librarian (OVAL) then turns the load module produced by the DOS linker into a core image and stores it on disk in a large contiguous file known as the overlay library, recording its position and length in a directory to enable fast and efficient recall. When the graphics system calls for a different overlay, which it references by number, the location of the overlay in this file is determined from the index, the overlay copied into core and execution commenced. Overlay swapping is controlled by a set of Fortran callable subroutines, where the overlay number is passed as a simple numeric arguement, and so can have any value at execution time, i.e. any overlay can call any other overlay, or even itself if so desired. The system maintains an overlay execution stack on a 'last in - first out' principle on which overlays can be queued for execution, a return from an overlay causing the top entry on the stack to be removed and executed. The stack contains not only the overlay number but also a second variable, the entry point, which can be used by the overlay to divide itself into logical dependant or independant segments.

A core map for the single user TIGER system is shown overleaf. In some instances the overlays use the "free core" area as additional buffer space outside of their own limits, but this operation is transparent to the user and is generally to speed up disk transfers by reading large blocks of data at one time rather than many smaller blocks. Furthermore, this is a function of individual overlays rather then the overlay system, and therefore does not merit description at this stage.

The directory structure for the overlay file is shown in figure 4.2 and a typical overlay library directory as produced by the librarian 'OVAL' in figure 4.3. Under the latest system the overlay library directory was of fixed length, allowing a maximum of 128 overlays to be defined.

4.2.2 <u>The Multi-user Overlay System</u>

The philosophy behind the multi-user overlay system was that, from a programmers point of view, it must behave much as its DOS predecessor in order to avoid extra complications in making the change from single to multiuser operation. To this end the concept of an overlay execution stack containing a first-in-last-out queue of tasks awaiting execution was maintained, with control over the flow of execution being by means of Fortran callable routines as before. Indeed, from an applications programmers point of view the differences between the two overlay systems are almost non-existant. The operation of the multi-user system is, however, radically different from its predecessor.

The RSX-11M executive is not limited to 32K words of core, since by address mapping, using the memory management hardware, it is possible to address up to 128K words of core in the PDP 11/45 and as much as 1960K on the larger machines. The RSX-11M executive is generally in control of the placing of a specific section of code, or 'task', in core at a location that is currently unoccupied, since it is possible

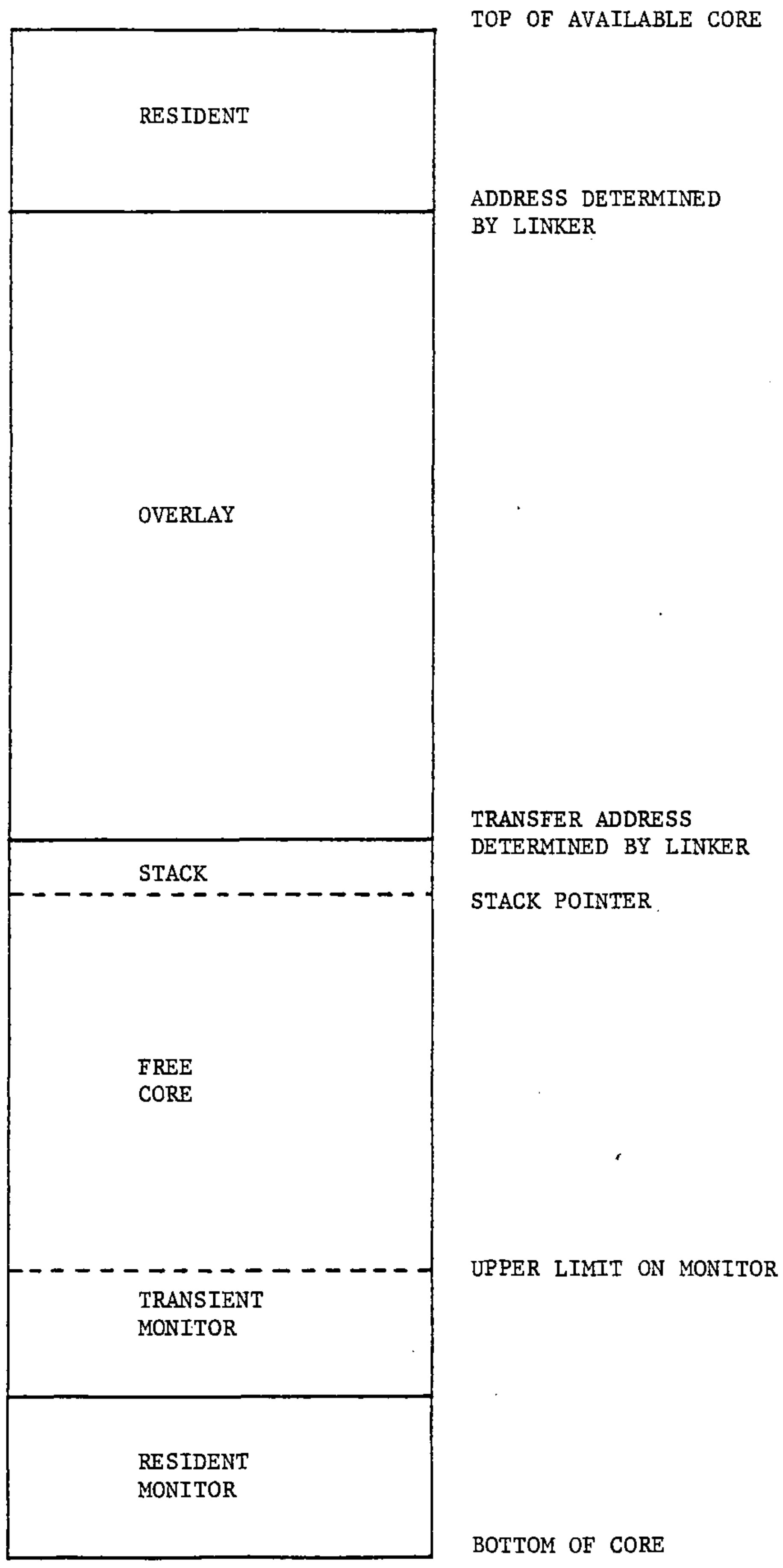


Figure 4.1 DOS Overlay Core Layout

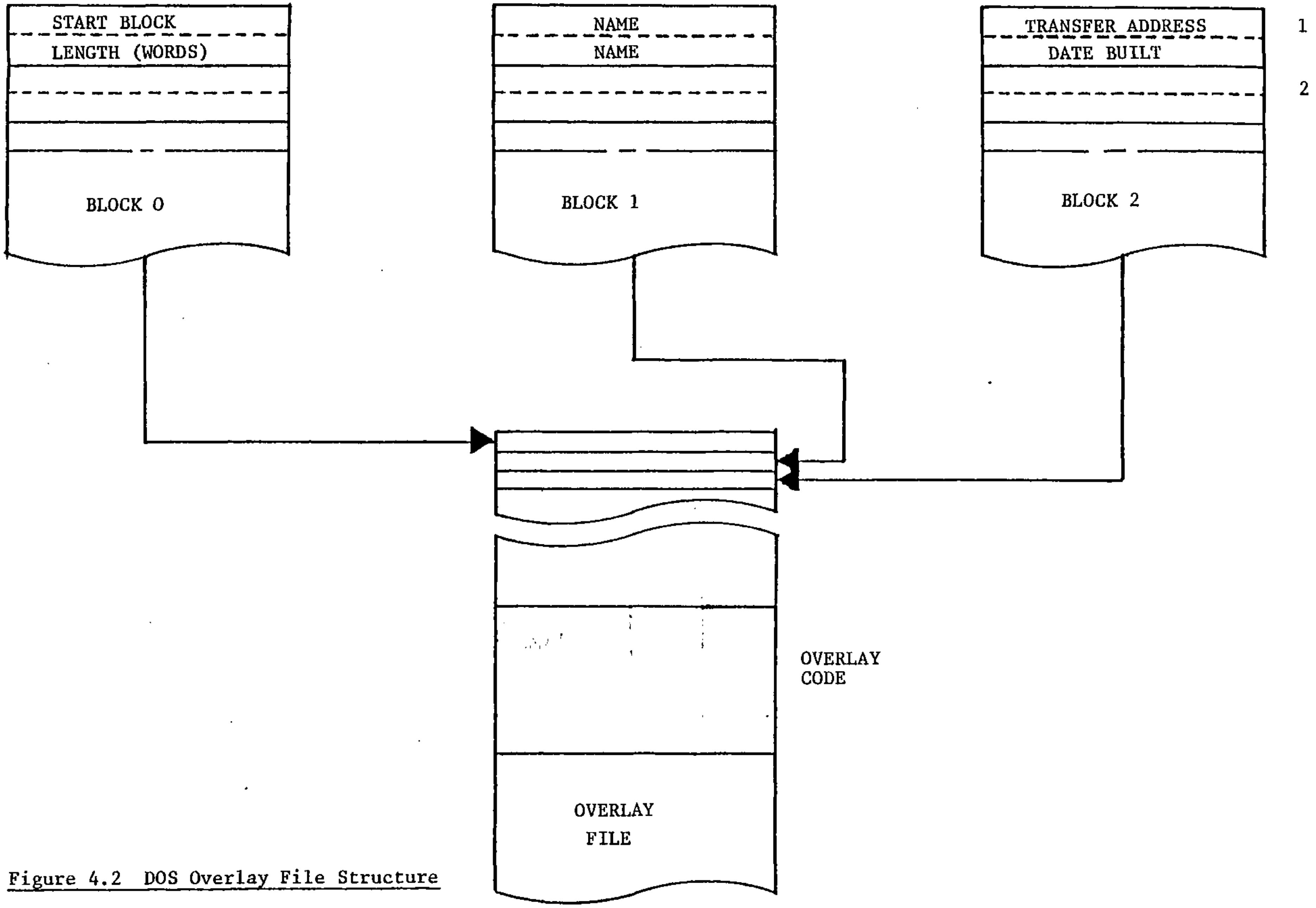


Figure 4.2 DOS Overlay File Structure

```
OVERLAY LIBRARY DIRECTORY 02-MAR-81 16:06:16

OVERLAY   OVERLAY    START     OVERLAY LENGTH    TRANSFER      LAST
NUMBER    NAME       BLOCK      WORDS/BLOCKS      ADDRESS   MODIFICATION
=======   =======    =====     ==============    ========   ============

     1    SETUP        868       5021      20     105006     05-JUN-80
     2    DIGOV        578       8634      34      66714     05-JUN-80
     3    ERROV         51        435       2     126732     17-JUL-80
     4    DEBUG        670       5010      20     105034     17-JUL-80
     5    OVFIL       1120      13449      53      44056     05-JUN-80
     6    DISALL       629      10482      41      57534     05-JUN-80
     7    LINED        300       4964      20     105170     05-JUN-80
     8    WINDOV       293       1800       7     121504     17-JUL-80
     9    RECOVR        24        275       2     127452     09-MAY-80
    10    JSTICK       991       2899      12     115252     09-MAY-80
    11    REVDAT       201        860       4      43000     09-MAY-80
    12    TIDY          22        283       2     127412     17-JUN-80
    13    CURFIT       383       7649      30      72616     09-MAY-80
    14    DOUBLE       413       2420      10     117150     09-MAY-80
    15    DVIEW        287       1348       6     123310     09-MAY-80
    16    LINTRA       690       3398      14     113304     09-MAY-80
    17    REVOLV       446       3129      13     114336     11-MAY-80
    18    CONDIG       459       4310      17     107644     11-MAY-80
    19    SYMED        476      10644      42      57030     05-JUN-80
    21    BUFRES       518       1999       8     120662     09-MAY-80
    22    GRDSUR       100       2518      10     116726     25-APR-80
    23    PRPFIL       110       5454      22     103346     25-APR-80
    24    MAC3D       1175       6290      25     100054     11-MAY-80
    25    PLTOV1       362       5313      21     103716     09-MAY-80
    26    PLTOV2       537      10302      41      60324     09-MAY-80
    27    TRANSF       132       3202      13     114176     25-APR-80
    28    PEXWIN       322       2544      10     116560     11-MAY-80
    29    DISCON        97        582       3     126304     09-MAY-80
    30    MANEDT       332       4074      16     110574     11-MAY-80
    31    MOVPT        205       3207      13     114102     11-MAY-80
    32    CIWRAP        78       1641       7     122280     25-APR-80
    33    MACSET        85       1433       6     123036     11-MAY-80
    34    MACFIL       914       4918      20     105344     11-MAY-80



   101    RMODE        270       4305      17     107656     09-MAY-80
   102    POLY        1058       6406      26      77504     09-MAY-80
   103    ALPNUM       888       6449      26      77356     09-MAY-80
   104    CIRC3D       218       7152      28      74560     09-MAY-80
   105    ARC3D       1200       7524      30      73210     11-MAY-80
   106    SPHERE       423       5785      23     102036     11-MAY-80
   107    ANOTAT       612       4181      17     110246     11-MAY-80
   108    DIMEN1       704       7330      29      74014     11-MAY-80
   109    DIMEN2      1230       8565      34      67146     11-MAY-80
   110    ELIPSE       934       6519      26      77142     11-MAY-80
   111    LINEM        320        321       2     127316     09-MAY-80

FREE BLOCKS:   314
FREE FILES:     61
```

Figure 4.3 Section of a typical overlay directory listing

for a number of users to have tasks executing simultaneously. By considering each of the overlays, as they were called under DOS, as a separate task, it is possible to instruct the RSX executive to load and begin executing them, so eliminating much of the work that had to be done locally by the DOS overlay system. In order to communicate data between tasks the resident common area described in the previous section must also be catered for. Furthermore, unlike the DOS system, there may be more than one user working simultaneously, so necessitating a resident area to be allocated for each user. This is achieved as follows. When a user logs on to the graphics system a dynamic common region is automatically created for him in core and filled with the default resident parameters. In order to access this throughout the drawing session each task automatically includes it in its address space by mapping to it when it is run. These regions are given the name COMnnn, where nnn is the number of the user station to which it applies. The executive knows this region simply by an identification code, and this code is passed between tasks to enable each to map to the correct region. All tasks in the multi-user system must run with a unique name, and since it is possible that two or more user stations may be executing the same task simultaneously it is necessary to identify the tasks with a name unique to the user station to which they apply. This problem exists not only for the graphics system, but also for normal RSX utility functions such as the Fortran compiler. The executive normally overcomes this by a process known as "spawning". The parent task is installed, that is made known to the executive, with a task name such as ...F4P (the Fortran compiler), but when run an offspring task is produced identical to the parent but with a task name of F4PTnn, where the last 3 characters reflect the terminal number from which it is run, e.g. F4PT13 is a copy of the compiler run from terminal number 13. A similar concept is used for the graphics system where each individual graphics task is identified by a name of four characters, such as MONT, but run under the spawned name of MONTnn where nn reflects the user station number.

The overlay librarian described in the previous section in its DOS format now has a completely different function. In general, it is necessary for a task to be installed for the RSX executive to spawn a copy for different user stations. However, it was realised that the number of graphics tasks would be very large and to have each of these installed at all times would be wasteful on core, since each installed task occupies a small area of core with an allocation known as a 'Task Control Block', giving essential information such as the task name, priority, location on disk etc. When a task is to be included in the graphics system the task librarian (TSL) now installs the task, copies its Task Control Block into a library created on disk, and then removes the task again. When a request is made for the task to be run this directory entry is quickly and simply copied into the system task directory, (an executive list of currently known tasks) effectively installing the task, the name modified to reflect the user station requesting it, and execution commenced. By setting the appropriate flag in the task control block the executive can be instructed to remove the task automatically on its exit so avoiding the need for further action by the graphics software. The task library still contains a correlation between each task control block and a user specified task number to allow tasks to be called by number as in the single user graphics system, but no longer contains the actual task images, which reside in the individual files created by the task builder. The structure of the library file is illustrated in figure 4.4.

Every task is responsible for the loading and execution of the next task as in the DOS system, although under DOS the code to achieve this was included in every task, since it was simple and small. However, under RSX the code is neither small nor simple and requires access to executive functions not normally available to standard tasks, and so the process is achieved by an entirely separate task, the 'loader' task, which is permanently resident in core and is shared by every user station. The loader task, when first run, makes a copy of the task library in a condensed format for its own use. At Imperial

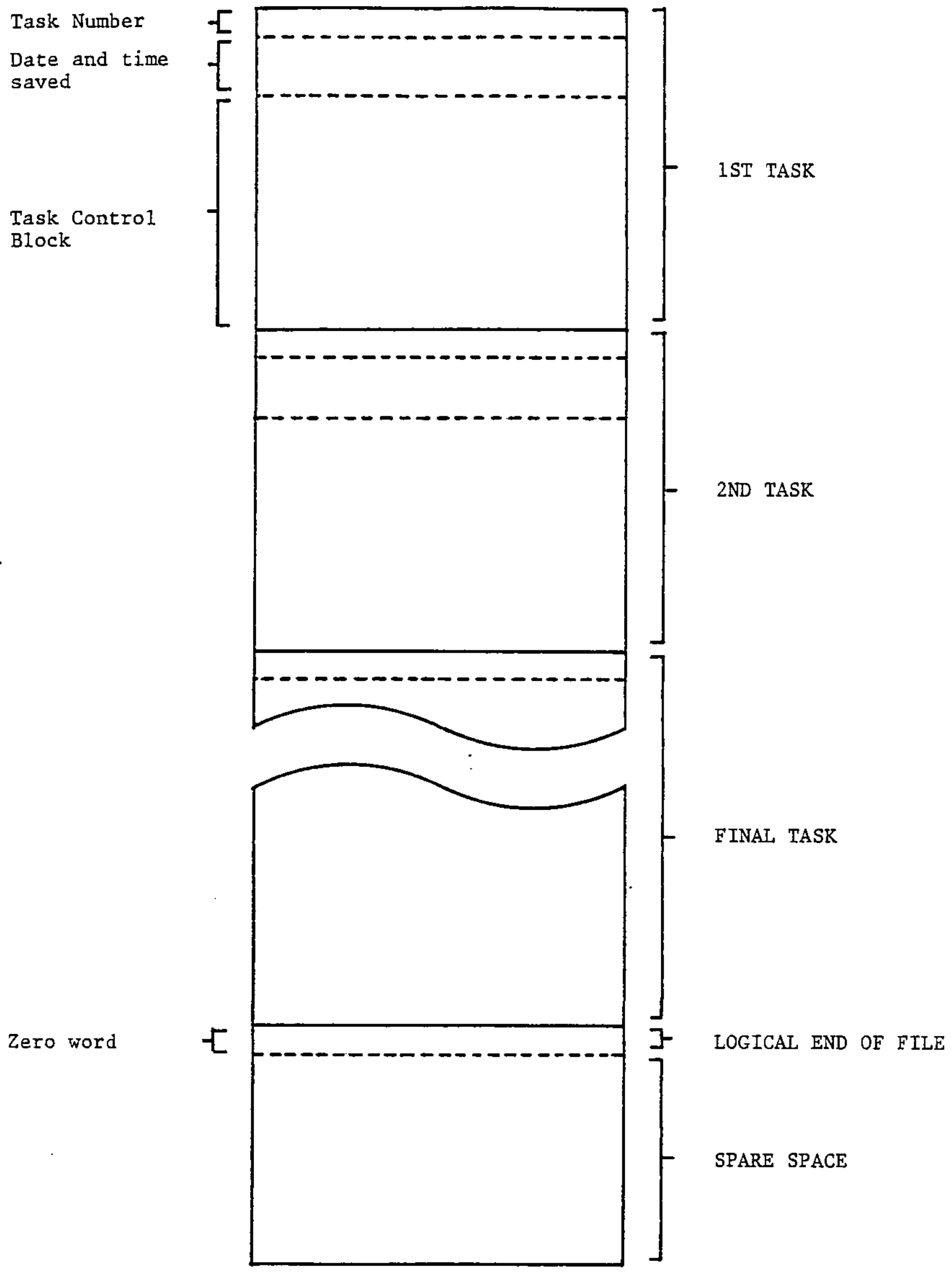


Fig. 4.4 The Multiuser task library format

College this is a disk file, but at Balfour Beatty it is kept within bulk core for very much faster access. When a task is required to be run the following sequence of operations is performed.

a) The initiating task ensures the loader is active and sends it the following information:-

   1) The number of the task to be executed.
   2) The user station which requires it.
   3) The identification of that user stations resident area, as assigned by the executive.

b) The loader task then:-

   1) Completes any operation it may already be performing for another user.
   2) Receives the above information.
   3) Locates the appropriate Task Control Block from its library and enters it in the system task directory, modifying the name to suit the requesting user station.
   4) Sets the new task running.
   5) Sends the new task the identification of its resident area.

c) The new task then:-

   1) Receives the resident identification and maps to the correct region in order to achieve access to the common parameters.
   2) Identifies the appropriate terminal numbers corresponding to this user station and assigns them for communicating with the operator.
   3) Continues to perform the required function.

A number of significant improvements have been achieved by this new concept which may be used to advantage by the applications programmer.

Since tasks no longer share core space with one another, it is quite possible for more than one task to be active for a given user station at any one time. This enables, for example, a routine designed to analyse data input from the digitiser to run concurrently with further input, resulting in improved performance for the user and more efficient use of machine time. The programmer must take care that this is not taken to excess to avoid overloading the system, although it is unlikely that more than two tasks will ever run concurrently, and if a task is requested that is already running the loader will automatically wait for it to exit before continuing.

The limitation of 128 tasks in the library imposed by the directory structure of the DOS system has now been extended to 32767, the maximum positive integer available, by use of an infinitely expandable directory, which is considered adequate for all foreseeable applications. Since the task library no longer contains task images its size has been radically reduced, and the problems previously encountered with one large contiguous file eliminated. The extension of allowable task numbers makes it possible to allocate blocks of numbers for different applications, so keeping a tighter check on tasks, the numbers being allocated to indicate the general function of each task.

Since each section of code is a task in its own right it may have all the attributes of any stand alone task, e.g. different tasks may run at different priorities, have different access rights etc.

Perhaps the greatest benefit is that this system, unlike the DOS system, contains very few non standard functions while providing considerably better facilities.

4.3 Graphics Databases

The selection of a suitable database for a graphics system is fundamental to the general performance and flexibility of that system, and a bad decision at this point will have far reaching consequences at a later date. Much research has been done into this subject, not only at Imperial College but also for many other graphics systems in both academic and commercial environments. The selection of a database for this project was largely governed by existing software available at the college which it was hoped to incorporate, and expand on, to produce a new and powerful three dimensional graphics system. Much of the preliminary research into suitable database structure was conducted by Hamlyn (9) and McLintock (10), who chiefly considered consequtive point storage and separate point/interconnecting line storage, and finally decided on the former. This approach was followed by Yi (14) in the early 3-D graphics system, and since this project was intended to follow on from his work it was thought best to continue with this approach. There is no doubt that different areas of CAD would benefit from different data structures, e.g. the ideal structure for a finite-element system would differ considerably from that for a pipe routing system. However, since this was intended as a general system it was felt that the data structure to be adopted was adequately efficient and versatile to have applications, with only minor modifications, in many different fields of CAD, and furthermore was simple enough for the applications programmer to understand readily. There is no doubt that a more complex data structure may enable more efficient and faster manipulation, but offset against this must be the inherent greater complexity of applications programs, which results in longer program development periods and greater possibility of errors. It is probable that in commercial applications the additional development costs would outweigh any small gains in productivity from a slightly faster system. It is argueable that a slight delay in response may be tolerable in that it allows the operator 'thinking time' to plan his next action, or simply to have a short break of concentration. Under the chosen system the data is defined essentially as a sequence of fixed length records, which may be considered to have four components, X, Y, Z and I. In the simplest application X, Y, Z defines a 3-D space coordinate and I indicates the significance of that coordinate; in more complicated applications these may contain information identifying a particular symbol, possibly its size and type, and groups of such records may be used to identify more complex graphics items such as macros. The concept of symbols and macros is discussed in chapter 5.

The selection of a list type data structure in no way governs the way the structure is used in the resulting graphics system, it simply provides the capability of storing data records consecutively within a file. The handling and implementation of this structure are two important areas of further consideration and are discussed in Section 4.5 and Chapter 5 respectively.

4.4 Graphics Work Files

The ability to handle large quantities of data quickly and efficiently are essential qualities of any CAD system irrespective of the data structure adopted. It has been estimated by one engineering company that an average drawing consists of approximately 40,000 vectors, and it is obvious that if this data is to be handled interactively a very efficient file handling capability must be included in order to avoid excess use of CPU time or irritatingly long delays for the operator. The requirements of a CAD filing system fall into two basic categories, namely workfiles, for short term storage, and permanent files for long term storage of complete drawings. The latter category may be handled by conventional filing facilities, although additional routines may be added to compliment these. A description of their use is included in Chapter 5. The former workfiles, however, are best handled independantly.

Workfiles may be defined as those files with which the user interacts directly during the construction of a drawing. This will include his workspace, the area in which the complete drawing is being built up, and any associated 'buffer' files, typically used for very brief storage of data for manipulation prior to adding to the drawing. It is an unfortunate, but well known, fact that the file handling capabilities of the Fortran language, particularly under the PDP11 DOS system, are poor, being confined chiefly to sequential access of records at a rate which is completely inacceptable for an interactive graphics system. This is largely a consequence of the Fortran ability to handle variable length records, and a file system where each block of a file does not necessary physically follow the previous one, which contains a pointer to it. It is therefore impossible to locate a particular record without reading through every preceeding record. True random access is therefore impossible, and records close to the end of a large file may take a considerable time to access. The concept of random access, the ability to address records out of sequence, is very important for graphics applications since it is often necessary to modify sections of the data without the need to read the entire database, as would be true with a sequential access file. True random access is generally only possible with contiguous files, ie. those where successive blocks are physically and numerically adjacent on the disk, enabling the location of any block to be deduced from the location of the first, or any other, block. A measure of random access was available with this type of file using standard PDP 11 DOS and RSX Fortran routines, although this was not considered adequate for the current requirements. No matter what data structure is adopted for the 3-D drawing information this must ultimately be grouped into disk units known as blocks, of 256 words each, and transferred to a file for storage. The filing process can therefore be considered to comprise two stages, the compacting of drawing records into blocks and the transfer of these blocks to disk storage. For convenience these processes have been kept separate since under some circumstances it may prove more efficient to use the latter process independantly, such as when copying from one file to another. The routines described on the following pages are general enough to handle all block I/O for the graphics workfiles, but are also employed by the specific graphical data handling routines described in section 4.5 for the transfer of blocked records to or from disk.

4.4.1 The Single User System

The method employed in the single user TIGER system was similar to that described by Hamlyn (9), where up to 14 preallocated contiguous files were available for direct access by specially written assembler routines. A number of significant improvements were made on the aforementioned system, perhaps the most important being that the user could specify, at run time, the disk and UIC under which the files reside, thus enabling more than one user to maintain completedly independant files on the same disk pack, or the pack to be placed in any drive in the case of a large multi-drive installation. By setting up, within the resident area, an index of the physical location of each file at log on time it was possible to compute very quickly the location of any particular block of data, and so greatly enhancing speed of operation. A severe limitation of contiguous files is that they must be preallocated before use, and unlike linked files are very difficult to expand should they prove too small. This limitation was accepted, but necessitated the user making a reasonable guess as to the final size of drawing in order that a sufficiently large file could be preallocated. An 'automatic' solution to this problem was never developed for the DOS system.

4.4.2 The multi-user System

As with the overlay system, it was essential to devise a file handling system that appears to the applications programmer to behave very similarly to the DOS predecessor, in order to avoid excessive problems when converting from one system to the other. The file handling characteristics of the RSX executive are completely different from DOS, and in particular the method of direct access to blocks within a file employed under DOS is not allowed by RSX. Under the latter system it is necessary to open the file by name, access the required blocks by their block number within the file rather than their position on disk, and ultimately to close the file when all data transfers are complete. This process is inevitably somewhat slower, but cannot be avoided without major changes to RSX itself, which would be contrary to the basic phylosophy of keeping the executive as standard as possible.

The multiuser system brings with it the further problem that a set of workfiles must be made available for every user currently logged onto the system, and these must have unique names. Furthermore, it is advantageous to have both private workfiles, containing information relevant only to a particular user, and public workfiles, generally of a read-only nature, containing information relevant to all users, such as character set specifications or menu layout data. The method devised caters for up to 10 private files and any number of public files, the former being identified by numbers from 1 to 10 and the latter by larger numbers. The limit of 10 files will adequately cover expansion in the immediate future, but should it prove a limitation it can be increased with a simple software modification. Based on the user station and file numbers a file name is devised as follows:-

a) Private files:

```
TIGuuunnn.DAT
```

where uuu = user station number in decimal
      nnn = file number in decimal (1 to 10) inclusive)

e.g TIG003002.DAT is private file number 2 for user station 3.

b) Public files:

```
TIGCOMnnn.DAT
```

where nnn = the file number in decimal (11 or more)

e.g TIGCOM014.DAT is public file number 14.

In a multi-user environment where several disk drives and work stations are available it is often necessary to allow different users access to different disks, particularly if they are working on different projects, which may be stored on seperate packs. A facility exists whereby the operator can assign drives to workstations. Private files will be assumed to be on this assigned device, whereas public files are always assumed to reside on the system disk.

When a user logs on the system automatically sets up the number of private files that will be required. An initial check is made to see if any private files are left from a previous session and if so these are deleted. This will not normally be the case since the logging off procedure automatically deletes private files, but will result if the previous session was terminated prematurely. This procedure ensures that the disk does not become filled with redundant files. Under normal circumstances a set of files will be allocated with an initial default length that has previously been decided by the system manager. If this allocation is insufficient the RSX executive automatically extends them when they become full, a feature that was unavailable under the DOS system. The system records information about these, and any public files found, in an area of the resident common, enabling the various graphics tasks to access the files quickly by a method known as "opening by file identification". This essentially enables the executive to open the file immediately without the need to search the directory for its name and is therefore a more speedy and efficient process, very important since it is anticipated that the files will be opened and closed frequently during the average digitising session. In order for the graphics tasks to read and write these files a number of Fortran callable subroutines are available, with identical names and arguements to those under the DOS system. An additional feature, implemented by an optional argument, allows asynchronous data transfers to be requested, whereby the calling program will continue to execute while the data transfer is taking place. Execution can be blocked at any time by calling a special subroutine which will force a wait until data transfer is complete. By omitting the optional argument from the subroutine call control will not be returned to the caller until the transfer is complete, so emulating precisely the operation of the routines in the single user system. Since files must be logically opened under RSX a check is always made to ensure that the file needed is open before data transfer is attempted. It is impractical, from space considerations, to keep all possible files open at all times, so a limit of 4 concurrently opened files has been set, this number being decided on from previous experience. If 4 files are already open when a read or write request is made necessitating the opening of a fifth file the least recently used of the 4 is closed to enable the new file to be opened in its place. This process is handled internally within the subroutines and is transparent to the calling routine. Programmers should, however, be aware of this mechanism if more than 4 files are in use to ensure that maximum efficiency is achieved with the minimum of file opening/closing overhead. It is also essential to ensure that a task closes all open files before it exits to avoid leaving them in a 'locked' state, a problem that did not exist under the DOS system. A special subroutine is available for this, since a file will not normally be closed once open unless space is needed as explained above. This subroutine is called automatically as part of the task exit procedure.

4.5 The 3-D Database Handler

The database handler, as outlined in the previous section, is designed to provide a means of blocking and deblocking 3-D data records into convenient units for storage on disk, and to enable the programmer to have a standard method of accessing these records, via subroutine calls, that can be employed in all routines. This frees the casual programmer from the necessity to understand in any great depth the complexities of the data structure.

Early graphics systems did not employ a database handler as such, and it was necessary for the programmer to block and deblock records for himself, which, together with a number of pointers, were stored in Fortran type common blocks within the resident area. This procedure had a number of disadvantages, not least that it was clumsy to program and necessitated the programmer understanding every feature of the database. An early attempt to overcome this was made by Yi(14) with a crude, but effective, database handler for the early 3-D systems that allowed the programmer access to the records via subroutine calls. This handler was written in assembly language and had the further advantages that it was considerably more efficient than the manual Fortran method, and that the bit and word handling capibilities of assembly language made it possible to introduce more complex and efficient data structures. From this early routine was evolved the full 3-D database handler currently employed by the graphics system.

It has already been shown that the basic 3-D record consists of 4 items, an 'I' code indicating the significance of the record and 3 values, X, Y, Z, which are normally a space coordinate. This logically leads to a record length of 7 words, 2 for each of X, Y, and Z, these being floating point values, and 1 for the I code. This represented an immediate saving in the DOS system over the manual handling of records since the Fortran compiler normally allocated 2 words to an integer value such as I but only used one of them! By handling records in assembler this problem is eliminated and much wasted space saved.

The basic phylosophy of the database handler was that it should, by subroutine call alone, allow the programmer to enter or extract records from the database and set or retrieve pointers at which input or output is to take place in order that true random access would be possible. The advantage of this approach can be seen below where (a) indicates the existing manual code required, and (b) the calls necessary to the database handling routines

a) Existing method without database handler:-

```
      COMMON/BUFFER/IB(32), XB(32), YB(32), ZB(32)
      COMMON/FILHND/NR1, IDP

      NR1 = NREC/32                 ! get block number
      IDP = NREC-NR1*32             ! get record within block
      CALL RAREAD (1,NR1,IB)        ! read the block
      I = IB (IDP)                  ! extract correct record
      X = XB (IDP)
      Y = YB (IDP)
      Z = ZB (IDP)
      IDP = IDP + 1                 ! advance to next
      IF (IDP.LE.32) GOTO100        ! continue if same block
      IDP = 1                       ! reset pointer
      CALL RAREAD (1,NR1,IB)        ! read new block

  100 ...
```

b) The same process with the database handler

```
      CALL SETOP (1, NREC)          ! set output pointer
      CALL GETR (1,I,X,Y,Z)         ! get the record
```

In the latter case the database handler takes care of deducing block and record number and ensuring that the correct block of data is always available in core. Furthermore, by elimination of the wasted integer word imposed by the former DOS process packing of the records has been improved from 32 per block to 36, this resulting in more efficient use of file space. The above calls to the database handling routines represent only two of a large number allowing the programmer full control over the storage and retrieval of records. A 'hidden' benefit of the database handler is that all I/O requests are double buffered and handled asynchronously. In the former example the subroutine call to read a block of data must wait while the actual transfer from disk takes place. This can introduce a significant delay on slow disks or a heavily loaded system. The database handler overcomes this to some extent by using two buffers in rotation. While information is being extracted from one buffer the other buffer is being read asynchronously from the file and vice-versa. A similar process also exists for input of records to a file. This implies that every input or output operation required two buffers in core of 256 words each. Obviously a limit must be imposed on the number of such operations that may be carried out concurrently to keep task size to a minimum. A practical limit, determined by experience, has been set at 4 pairs, although this is not to say that 4 pairs of buffers are always available. Buffer space is allocated at task build time within a specially named program section, the size of which must be determined by the programmer dependant upon the particular code employed. If too small a run time error message will advise the programmer of the mistake, if too large the task will simply be extended beyond the size required with no detrimental effects on its operation. The upper limit of 4 pairs is imposed by the need to preallocate pointers and flags within the body of the database handler, which cannot easily be modified at task build. The number of such pointers is small and it is not considered that the amount of space wasted when less than 4 buffer pairs are used is of any great significance. More than 4 concurrent I/O requests may be handled by coding that 'endfiles' i.e. deallocates buffer pairs so that they may be reused, but this must be done by the programmer and can result in considerable degredation in performance.

In addition to the I, X, Y, Z values it has been found necessary to associate further parameters with each record. For example, if a record indicates a line joining two points it must also indicate with which pen that line must be drawn on the plotter and whether it is a solid, dotted or other type of line.

These additional parameters are stored in Fortran type common blocks, one for the codes that are currently being used for input, and a second for the latest parameters that have been decoded on output. This approach has been used because it is more efficient that incorporating them as subroutine arguments, and frequently the programmer need not be concerned with them, or may require to transfer them between tasks, which the resident common block approach facilitates. The graphics system employes five such parameters which are manipulated by the database handler using two different methods, masking and indexing. The principal of each is as follows:-

a) Masked Parameters

A number of parameters change frequently throughout the database and it is convenient for their values to be included in every record. At first sight this may seem impossible without changing the record length, but by limiting the range of the parameters and the I code it is possible to encode them all within the I code word. Half of the word is reserved for I codes, giving a range from 0 to 255, which was judged adequate for all applications. The remaining parameters are encoded in the other 8 bits of the word as in figure 4.5. The use to which these parameters are put is explained in chapter 5. With the exception of the I code the database handler stores the parameter with a base of 0 but returns to the caller a parameter with base 1. For example, pen number is stored as a value from 0 to 3, but is seen by the programmer as a value from 1 to 4. The three additional parameters are encoded into a 'mask' which is added into the I word of every record before it is placed in the database. When records are subsequently extracted from the database the masked information is stripped from the I code and decoded into its separate components.

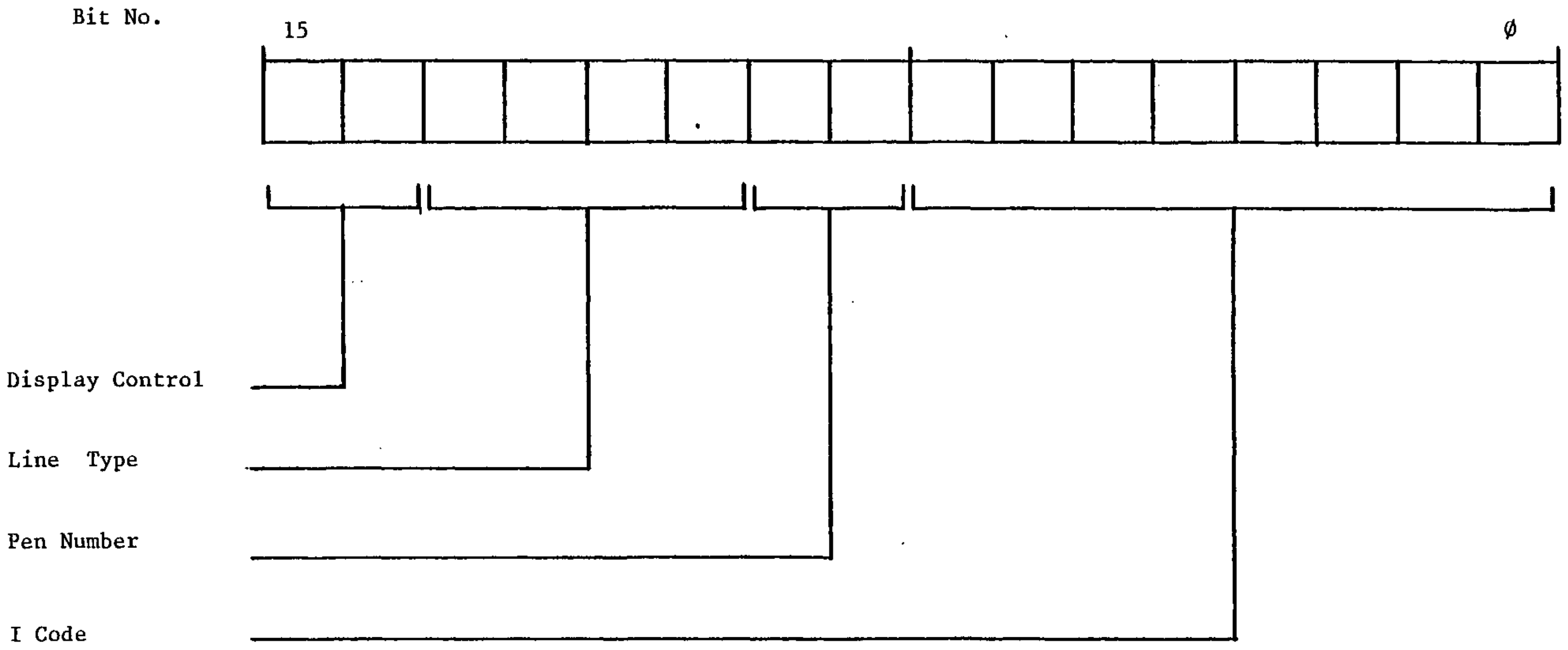


Figure 4.5 The I Word Structure

This is a feature that was only made possible by the advent of the database handler since it involves bit manipulations that were more or less impossible with Fortran routines. Indexed parameters were, however, tried with old systems with some level of success, but the following section describes the first full implementation of this principle.

b) Indexed Parameters

The above approach works well for parameters such as pen number that have limited values or change frequently. However, for parameters that must have larger ranges or that change more infrequently an indexing approach is more flexible. This method does not associate the value of a parameter with every record in the database, but identifies at which record the parameter changes and its new value. In the current implementation a whole word is allocated to the parameter, allowing a range of some 65,000 different values. By consulting the index, which is maintained in strict record number order, it is very easy to determine the value of a parameter for a particular record. Indexed parameters have one further great advantage over the masked parameters. Since the index indicates every change of parameter it is a simple matter to jump through the workspace picking up, for example, only records that have a particular value associated with them, as opposed to the masked parameters where there is no choice but to decode every record to find its value. For this reason the parameters that have been indexed in this application are those that are generally used for search type operations, namely the overlay and select status of the record, which are explained fully within chapter 5. In a process where the database is large but only a small number of records are of interest this technique can result in greatly increase speed of operation over the masked approach and reduces computing required dramatically.

The database handler manipulates the indexes in a way completely transparent to the calling task. The indexes are maintained in separate storage areas from the main body of the database, at Balfour Beatty these are within the bulk core for speed of access, but at Imperial College they reside in a disk file emulating the bulk core. This approach adds extra complication to operations such as saving and recalling drawings, but the benefits were considered to outweigh the disadvantages.

In general the database handler has been found to cater for all requirements and functions very satisfactorily, without being too complicated for the novice programmer to understand. Sufficient flexibility has been incorporated to enable minor specification changes to be made with little or no impact on applications programs.

4.6 Communication between Users and the Computer

An essential quality of an interactive graphics system is the ability for the user to communicate with the computer, and vice-versa, in a clear and coherent manner. To this end a number of standard interfaces and rules have been developed to ensure that every function issues prompts in a standard format and requests information from the user in a way familiar to him.

A number of lines of communication exit between the computer and the user. The computer can communicate with the user either graphically, on the graphics screen, or verbally by means of prompts issued to the alphanumeric terminal. In this application is has been considered essential to keep the graphics screen clear of all but purely graphically data, and as such it does no more than display the results of functions rather than assisting in the input of their raw data from the user. Prompt messages to the alphanumeric terminal are, however, very important in assisting the user in the operation of the system, and the following standard has been adopted in an attempt to rationalise them.

4.6.1 Prompt Messages

A prompt in this context is defined as any message that imparts information to the user without necessarily expecting further input from him, although in the majority of instances this will be the case. In order to differentiate betwen prompts they are always output with a single blank line between them and the previous prompt, and an appropriate number of tabs, equivalent to 8 columns, precede them, depending on their significance. A function, when initially requested, will normally identify itself with a message in capital letters starting in column 1, i.e. with no preceeding tabs, and then output further information in both upper and lower case letters starting at the first tab stop. In some instances during the servicing of the function information messages may be output by the basic graphics system which will normally start at the second tab stop to differentiate them from the messages output by the function itself. All such messages, or groups of messages if more than one is output consequentively, are optionally preceeded by a single bell character, causing a buzz or bleep tone from terminals suitably equipped, to draw the users attention to the message. For the benefit of the more experienced user, or where the workstation is situated such that the tones may be a nuisance, this feature can optionally be disabled and enabled by the user at any time during a drawing session. In order to distinguish between messages that simply impart information and those that ask specifically for some action from the user the latter are preceeded by a single asterisk character in the first column. The applications programmer need never be concerned with inserting the correct number of tabs within messages since this function is handled automatically by the system depending on a value set by him within the resident area. He must, of course, decide which messages should be preceeded by the '*' character since the system is unable to decide automatically which message are purely information and which asks specific questions.

4.6.2 User input on Alphanumeric Terminal

In response to a prompt the user will normally perform some operation, either on the digitiser or the alphanumeric terminal. The former case will be dealt with in a later section. Input from the terminal can be categorised into 4 classes, numeric, textual, confirmation or choice from a list. Each form of input will issue a further prompt, in addition to any specific information output as described above, to indicate to the user what type of input is required. These further prompts are output automatically and reflect the constraints and defaults that apply to the current request. These 'automatic' prompts obey the same laws of tab stops described above, so enabling the user to determine simply at what level his input is being made. The following standard forms of input have been adopted.

a) Numeric Input

A large number of functions require the input of numeric data, e.g. to specify the default radius for a circle or set a scale factor. The format of the default prompt, in this case with one tab stop, is illustrated below:

```
*    Enter the required scale factor
     Range 1.0 to 100.0, default = 2.0
     <NUM>:
```

The initial line is output as a specific prompt in the manner described in the previous section. The second and last line are generated by the system depending on the constraints or defaults applied by the programmer. Other options include ranges up to or down to a certain value with no limit at the opposite end of the range, and no default reply, which is normally selected simply by answering with a carriage return alone to the prompt on the 3rd line, which in this case indicates that a number is required by means of the string 'NUM'. The system is so designed that should the user be unsure of what is required he may enter "?" on its own, which results in a small amount of information being output and the prompt repeated. This is normally a condensed version of the information that is available in the user manual. Should the user enter a number that is out of the allowed range an error is reported and the prompt repeated until a legal response is received. The operation of this form of input, as with the other 3, is asynchronous, and it is possible for the system to be performing some other function while the user is entering his or her response. This can result in what appears to the user to be a faster response to commands and results in more efficient use of computer resources.

b) <u>Textual Input</u>

Textual input is in many ways the simplest format since it involves no processing and there are no limitations, other than maximum string length, that have to be obeyed. The format of the default prompt is as shown below.

```
*    Please enter the drawing title
     <TEXT>:
```

As in the previous example the initial line is output as an explicit prompt whereas the second line is generated automatically by the text input routine. The routine will accept up to a programmer specified maximum number of characters in much the same way as in example (a). The user may enter '?' to obtain help, and again the input is handled asynchronously. There are no error conditions possible within this routine, the only constraint being the maximum character count, and input will normally be truncated automatically to that length.

c) <u>Confirmation</u>

In many cases the user is asked whether or not some action is to be taken, or whether some assumption is correct. In this case he or she must respond 'Yes' or 'No' on the alphanumeric keyboard, both of which may be abbreviated to their first letter. An example of confirmation is as below:

```
*    Do you want to continue? [Y/N]:
```

The programmer must supply the basic question, but the "? [Y/N]" string is appended automatically by the system. The users response appears immediately to the right of the question. As with the previous input modes the user may enter a question mark for information, and the facility exists for the program to continue on another function while the user is entering his decision. Any response other than 'Yes' or 'No' is considered illegal and the prompt repeated until the correct input is received.

d) Choice from a List

The user will frequently be required to choose one option from a list of options. The final type of input provides this facility. A typical example may be:

```
*    What type of valve to you require?
     1 - Gate
     2 - Ball
     3 - Butterfly
     ?
```

The number of choices is dependent on the particular application and has no practical upper limit. The user is expected to enter the number corresponding to the required choice, e.g. to select a ball valve in the above example he would enter the number 2 on the alphanumeric terminal. The number entered is checked for legality and the prompt repeated if it is found to be out of range. As with the previous options the choice is made asynchronously, and the user may enter a question mark for help with this particular question.

- xxx -

The users second method of inputing information to the computer is via the digitiser, usually as a result of a prompt issued on the alphanumeric terminal. The use of the digitiser for issuing commands differs significantly from the terminal and is described below.

4.6.3 User Input from the Digitiser

The data that is output by the digitiser to the computer is somewhat more limited than that from the terminal. Upon pressing a button on

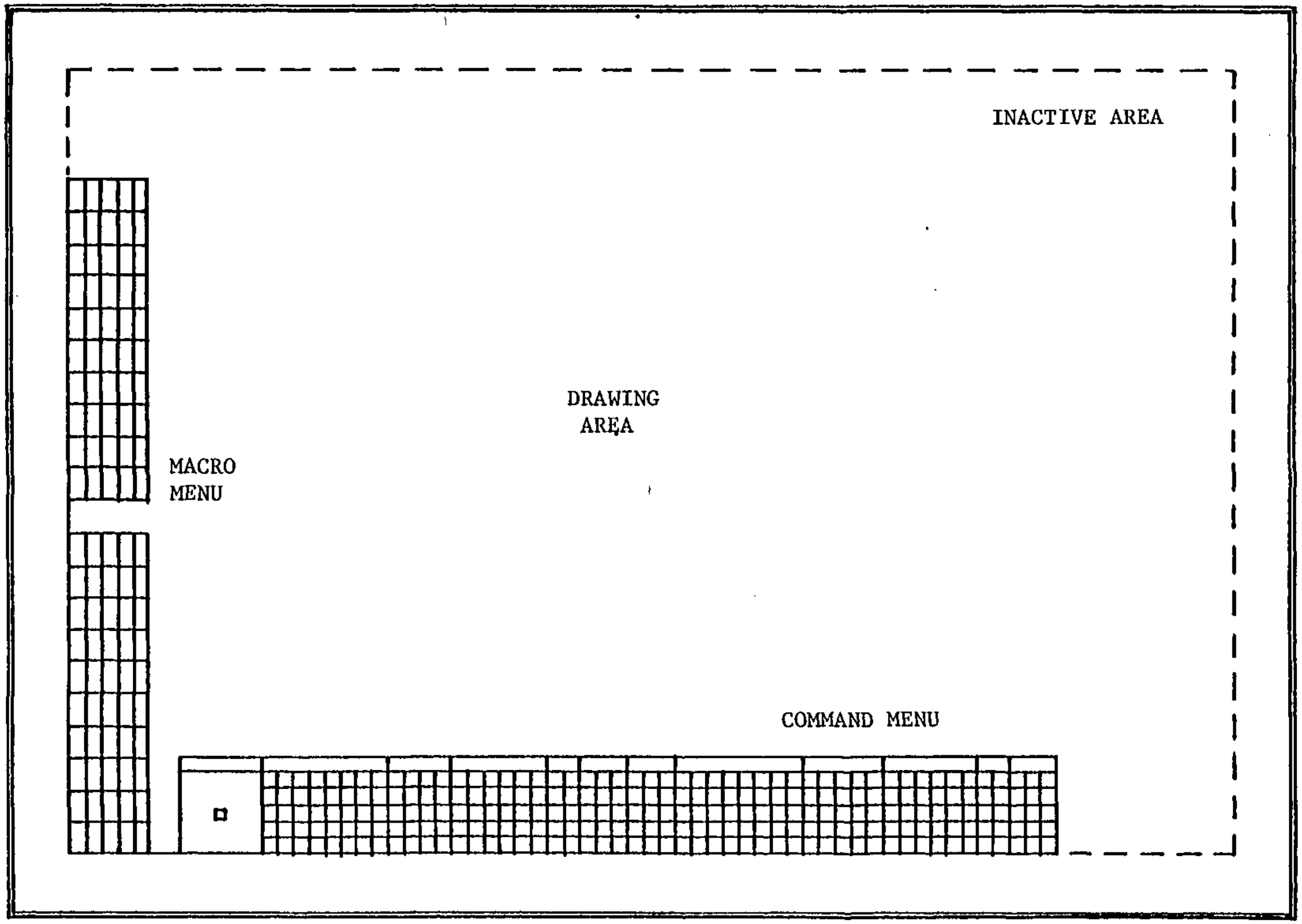


Figure 4.6 Layout of the Digitiser

the digitiser the computer receives that button number and the current position of the stylus on the digitiser. The way in which that information is interpreted will depend greatly upon the current function being performed and the position on the digitiser. In the TIGER graphics system the digitiser is logically divided into 3 areas as shown in figure 4.6, of which the largest is the drawing area. The exact definition of these areas is purely a software function and may be varied to suit a particular application or digitiser.

The drawing area corresponds approximately to the draughtsmans drawing board. Over this the draughtsman may place drawings or rough sketches to be digitised. A detailed description of the use to which this area is put is contained in section 4.7, since in the 3-D system it may be further notionally divided into different views or projections.

The menu area serves an important function in that it constitutes the users principal means of issueing commands from the digitiser. The concept of menu operation has been known for some time and is now exploited almost universally in systems incorporating a digitiser or tablet, the latter generally being a term applied to a very small digitiser. The menu is divided into a large number of squares, each of which is assigned a function within the system that is activated simply by digitising over that square. A typical menu may consist of several hundred squares, the current layout in the TIGER system incorporating 250, although this may be varied to suit operational requirements with very little effort. Early graphics systems at Imperial College employed a menu of 2cm squares in 30 rows of 10 columns situated along the left hand edge of the digitiser. This arrangement had originally been selected because the graphics screen stood to the left of the digitiser and it was advantageous for the user to be as close to this as possible, particularly in view of the very small screen available. However, in practice this arrangement revealed some operational difficulties, not least that to work on a large drawing the user must stand to the right of the menu which places him on impractical distance from the screen. It was felt that for physically small users the top menu squares could prove difficult to reach, and since the majority of users would be right handed it was illogical to have the menu to the left of the user. After a considerable amount of trial and error the layout shown in figure 4.6 was selected. In this arrangement the menu squares lie horizontally along the bottom of the digitiser in 5 rows of 50 squares each. To save space the squares have been reduced to 1.5cm, which proved adequately large to contain a short verbal description of the function assigned to them. Within these 250 squares commands are typically grouped according to function, for example editors, or symbol generators. This grouping is purely for operator convenience since the software assigns no particular significance to any square. A section of a typical menu is shown in figure 4.7 to illustrate the grouping. It is evident that in a large graphics system there will be many more than 250 commands, and in order to accomodate this requirement a system of menu pages has been developed. Each page, numbered upwards from 1, consists of 250 squares, of which page 1 is the default. If a command is needed that is on another page the user simply replaces the menu on the digitiser with the appropriate page and tells the computer by means of a menu command which page has been selected. From that point onwards the commands on the new page will be available. This is achieved by storing a file describing each page in detail. For convenience the programmer sees this as a text file where each line describes one menu command in the form of square number, servicing task number with its entry point, priority, and the text that appears in the square, all separated by commas. This is processed by the system into a machine readable binary file which can be accessed quicker and saves space, the current page being copied by the graphics software into bulk core for very fast access. The text file is also used as input to a special system program that plots the menus on the flat bed plotter. The top left hand square on each menu is dedicated to the function enabling the current menu to be changed so that it may be located rapidly and ensures that the user cannot become 'stuck' on one menu without the ability to change to another. The menu priority system was developed because it was found that a

GRIDS		INFO REQUESTS		LINES AND PENS			SELECT FUNCTIONS			GEOMETRIC FIGURES (SYMBOLS)						
ENABLE X grid	DISABLE X grid	HELP explain menu commands	output CURRENT SYSTEM PARAM'S	change to PEN 1	change to LINE TYPE 1	define LINE TYPE structure	select WITHIN A BOUNDARY	select 2-D data		CIRCLE centre & default radius	SEMI CIRC centre & default radius	MINOR ARC centre & default radius	MAJOR ARC centre & default radius		POLYGON on one side	RECT'GLE diagonal
ENABLE Y grid	DISABLE Y grid			change to PEN 2	change to LINE TYPE 2		select ACTIVE OVERLAYS	select 3-D data		CIRCLE centre & circumf point	SEMI CIRC centre & circumf point	MINOR ARC centre & circumf point	MAJOR ARC centre & circumf point		POLYGON centre & vertex	RECT'GLE centre & corner
ENABLE Z grid	DISABLE Z grid			change to PEN 3	change to LINE TYPE 3		select PASSIVE OVERLAYS			CIRCLE 3-point mode	SEMI CIRC 3 point mode	MINOR ARC 3 point mode	MAJOR ARC 3 point mode			RECT'GLE centre & dims
				change to PEN 4	change to LINE TYPE 4		select SPECIFIC OVERLAYS			CIRCLE centre & specify radius	SEMI CIRC diameter mode	MINOR ARC chord and height	MINOR ARC centre specify radius		ELLIPSE centre mode	RECT'GLE corner & dims
FULL OR BOUNDING node display	SET UP GRIDS				change to OTHER LINE TYPE	MACRO default line type	select ALL DATA		CANCEL SELECT status	set DEFAULT CIRCLE radius	set DEFAULT FILLET RADIUS	set DEFAULT ARC RADIUS			ELLIPSE maj axis mode	

Figure 4.7 Section of a typical command menu

number of commands could be mutually destructive. A large number of functions require the user to input information, such as the location of a symbol, from the digitiser. In early systems it was easy for the user to either accidentally or deliberately call up a further menu command that interfered destructively with that currently being executed. To obviate this problem a system of priorities was developed. When a function is being performed the menu is locked to prohibit the user of any function of a lower priority. By judicious assignment of priorities it is possible to arrange that only non destructive functions may be performed until the current function is complete.

The digitising stylus has on it some 12 buttons at Imperial College and 16 at Balfour Beatty which can normally be considered an extension of the basic menu. These buttons perform functions that are very commonly used and which it is convenient to have available 'at the users fingertips'. The normal assignment of buttons is described in some detail in chapter 5. Under some circumstances this assignment may vary to suit particular operational requirements. In early Imperial College systems the buttons were frequently used in obscure combinations to input data such as numeric valves. The introduction of the alphanumeric terminal has rendered this function obsolete.

The third and final area of the digitiser consists of the macro menu, which is located along the left hand edge of the digitiser. The theory and operation of the macro facility is very important to the efficient use of a computer aided draughting system and as a result the important role it plays is necessarily complicated in operation. A full description of the macro functions implemented in the TIGER graphics system is included in chapter 5.

4.7 3-Dimensional Input Techniques

In order to achieve the input of true 3-D data from a 2-D digitiser some special technique must be devised. At present a workable 3-D digitiser is still a thing of the future, although experiments have been done with an acoustic digitiser capable of sensing not only the position of the pen in X, Y coordinates but also its height above the digitising surface. A number of techniques have been devised to allow full 3-D information to be constructed from one or more 2-D representations of an object, most of which have been tested at some time at Imperial College. The following methods have proved to be workable.

4.7.1 Photogrammetry

This technique was originally devised for use in the shipbuilding industry, and was developed at Imperial College by Dodd for a similar application. The process consists of constructing 3-D data from a pair of stereo photographs of an object, and although high accuracy is possible it requires complex and expensive equipment and in many cases is not practical. It is not entirely suitable for the design process since either the actual subject or an accurate model is needed to obtain the photographs in the first place, and implementing amendments can be both a difficult and time consuming process. Since it is generally accepted that the greatest advantages of CAD over manual methods lie in the area of design and the ability to include modification quickly this input method was not considered to be worthy of further investigation within the bounds of this project.

4.7.2 Contouring

The representation of a 3rd dimension on a flat surface by the use of contours is a principle well known to most people, perhaps the most obvious example being that of Ordnance Survey and similar maps. It is a fairly simple process to input 3-D data from a contour map since, by definition, along the line of a contour one dimension remains constant, thus if this has been predefined by the operator it is a straightforward process for the system to generate true 3-D coordinates by following the contour lines with a digitiser. However, if the data so produced is to be used for engineering it must first be processed into a more useable form, generally by interpolation and curve fitting to produce surfaces. Unfortunately, in many branches of engineering this representation is unsuitable, for example digitising a 3-dimensional pipe run with all its ancilliary equipment is almost impossible.

This, and many other applications, requires the ability to locate accurately discrete 3 dimensional points in space, a process for which contouring is far from suited.

Since the engineer is familiar with orthographic representations of solids the following method, which fulfills the above criterion, has been developed as the principal method of input to the TIGER system.

4.7.3 Orthographic Representation

The conventional representation of a 3-D object is by means of 2 or more orthogonal projections or an isometric projection where each projection contains sufficient information such that when they are considered together the object is fully defined. It is obviously advantageous if this method can be used for 3-D input via a digitiser since it relies on principles already known to the draughtsman and should therefore be readily understandable.

The method devised for the TIGER system allows the user to notionally divide the drawing area into up to 16 different projections, although in practice it is rare that more than 4 will be needed. Each of these areas may be defined as either an isometric, perspective or orthogonal projection. Of these it is normally only practical to allow digitising over orthogonal views, although a number of functions, such as finding an existing point, may be performed with pictorial type projections. Each view is entirely separate from all others and may take different origins, axes and scale factors. The default assignment made available at log on is as shown overleaf. It is recognised that this corresponds directly with neither 1st nor 3rd angle projection but is a compromise that has proved to be convenient to use since the X-Y elevation, which is generally the most significant view, is located at the bottom left within easy reach of the operator from a sitting position. The user must generally reset the views to suit his current drawing. An obvious requirement of this is that the selected views are sufficient to provide a full 3-D definition of the object. Once the necessary views have been established the method of 3-D input is straightforward and is based on picking up the same point in two different orthogonal views. When digitising over any one view it is only possible to define, by the position of the stylus relative to the origin of the view, and the appropriate scale factors appertaining to that view, two of the 3 space coordinates. The 3rd coordinate is therefore defaulted to the last valve it was given. In order to change this value the user must digitise in another view which defines it explicited, whereafter reverting to the original view enables digitising to continue with the new 3rd, or 'trailing', value. A special button on the stylus allows this dummy point in the second view to be digitised without entering it into the database, since it is merely intended to indicate a change of the trailing value. The operation of this button is discussed in chapter 5.

At any time during the drawing process the user may disable a view so that it does not appear on the graphics screen or the final plot. This process can be reversed simply by re-enabling the view. Alternatively, the view may be deleted entirely, and cannot be recalled without fully redefining it. New views may be defined over any area of the digitiser that is not currently assigned to another view, and normally occupy a rectangular area of any shape that suits the application. View boundaries are indicated on the graphics screen

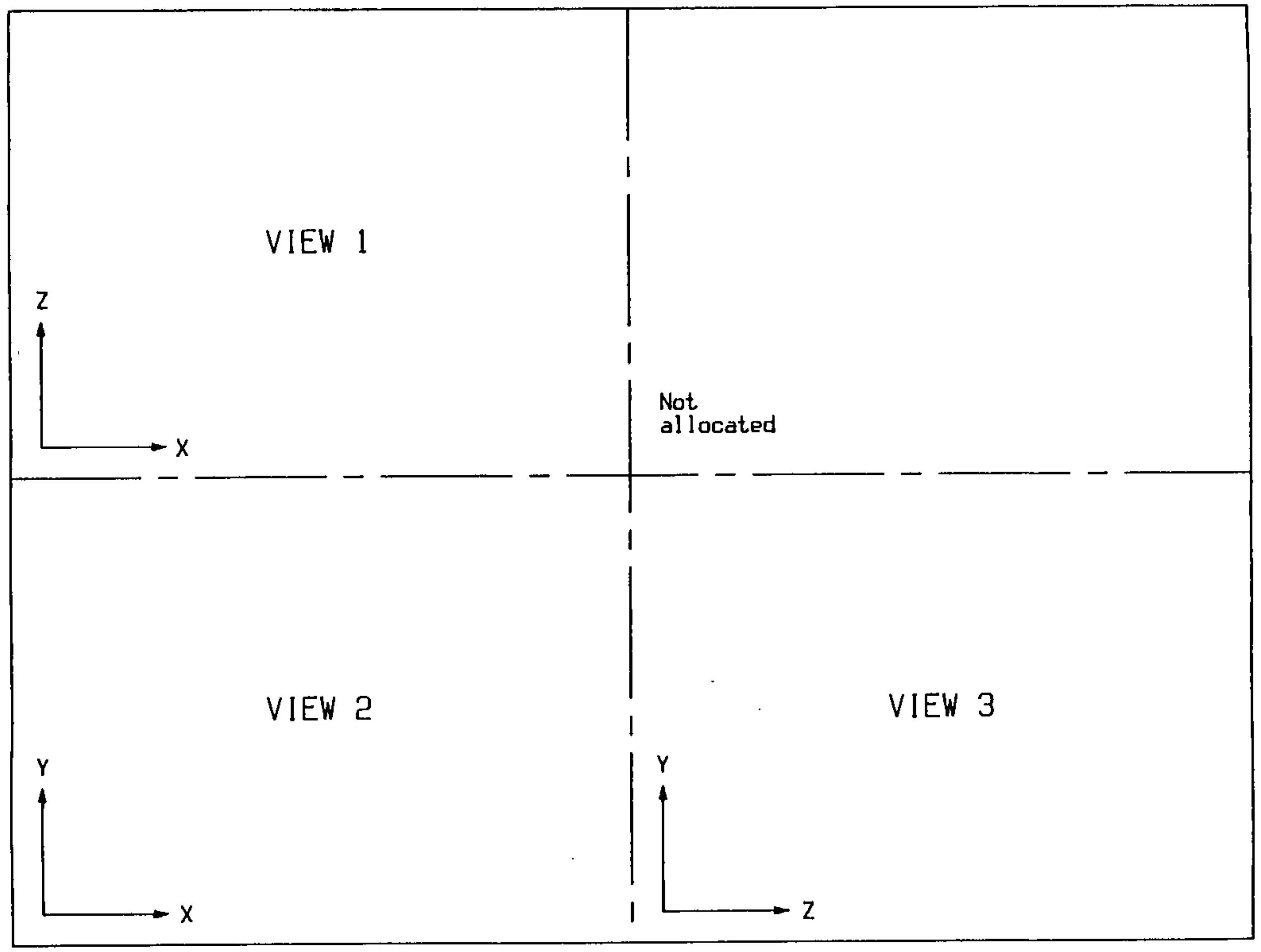


Figure 4.8 Default view allocations

by dashed lines, but under normal circumstances these will not appear on the final drawing.

A special mode of operation is possible where only one orthogonal type view is used. This may be optionally selected by the user and is known as 2-D mode, since it is impossible to input 3-D data with only one view. This is a pseudo mode since the full 3-D database is still used, but by implementing this mode a number of functions can be simplified. A full description of the differences between the modes is included in the following chapter.

Early graphics systems often relied on the manual input of coordinates from devices such as card readers or terminals. Although this method is slow and not strictly interactive in nature it was felt that there may be a case for including such a facility within the TIGER system, perhaps to establish exact datum points within a drawing, or to input points that may be beyond the range of the digitiser. This function is implement via a button on the stylus and is described in detail in chapter 5.

The above methods have been used with considerable success throughout the duration of the project and are now proving very satisfactory in a production environment.

4.8 3-Dimensional Viewing Techniques

In many ways the techniques available for displaying three dimensional information are simply the opposite of those discussed in the previous section with regard to the input of such data. The arguments applied for and against each technique apply equally for the display process and the input process. For example, contouring was dismissed as a feasible method for the input of engineering data, and similarly the display of such data, which may comprise pipelines, valves etc, is not suited to a contour type representation.

It has previously been stated that the drawing area of the digitiser may be divided into a number of views of an object, and that the position of the digitising stylus is monitored on the graphics screen by means of a cross hair cursor. In this way the screen acts as a 'window' to display what appears to the user to lie over a particular area of the digitiser. If the display of the drawing on the screen, the cursor monitoring the position, and the users rough sketches on the digitiser are to relate to each other in any meaningful manner it follows that the views generated on the screen must be identical to those current in use for inputting the drawing from the digitiser. This principal has the further advantage that earlier and well tried 2-D display techniques may be adopted since the display process may be considered in two stages, initially the mapping of a 3-D point into its projections on the digitiser and then the display of this 2-D representation on the graphics screen. This latter process is dependant on the current display window in use, and is discussed at some length in chapter 5.

The principals involved in the above mapping process depend on the type of view to be generated. The system currently handles three principal types, orthogonal, perspective and isometric, although it is hoped to expand this in the future to include features such as cross sections. The principles involved in the 3 methods are as follows.

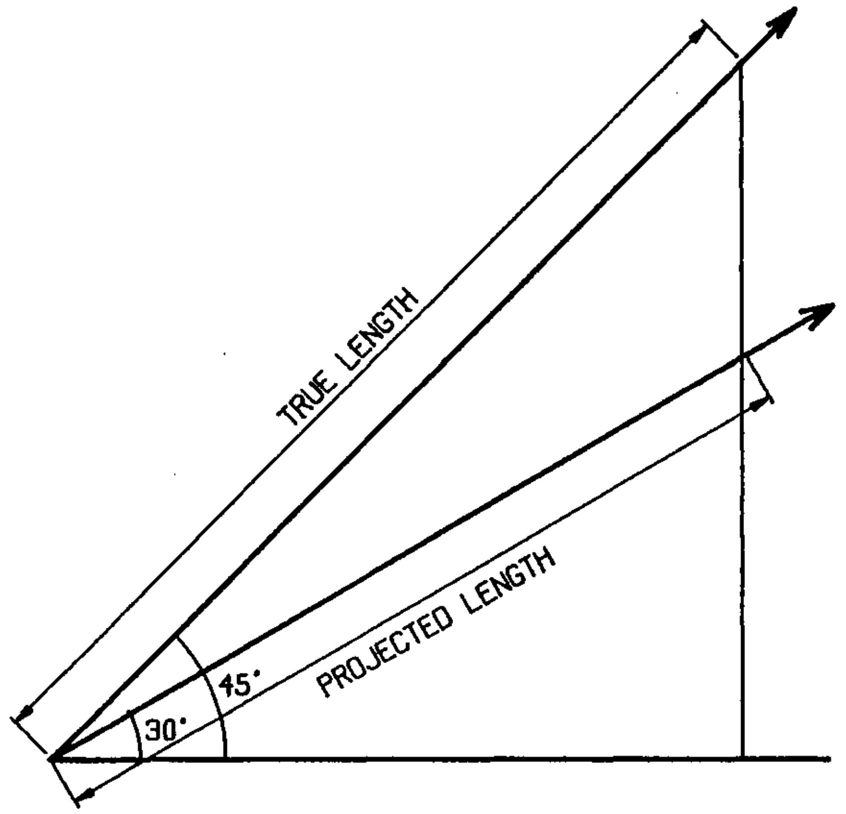
4.8.1 Orthogonal Projections

The orthogonal projection is the simplest to generate since by definition it is a view obtained by looking parallel to one of the three orthogonal major axes. In this way, one of the three space coordinates, depending on which projection is being generated, may be effectively ignored and a simple 2-D process used on the remaining values. In many ways this makes an orthogonal view a 'pseudo' view since an object is never seen this way with no perspective type effects to give an indication of depth, but it is of course an accepted drawing standard and must therefore be included.

4.8.2 Isometric Projection

The isometric projection is a pictorial type of projection giving, in one view, an indication of distances along all three major axes. The view obtained is slightly distorted compared to the perspective view we are used to in everyday life since distant objects appear no smaller than those much closer to the apparent position of the observer. The isometric projection plays an important role in engineering and may be used in applications where a clearer representation of an object is achieved, particularly if those interpreting the drawing are not skilled draughtsmen. For this reason isometrics are often used 'on site' or in other applications where they must be read and interpreted by construction staff who are making or assembling the indicated items.

The graphics system is capable of generating isometric projections automatically within a number of constraints for previously entered 3-D information. Figure 4.9 shows diagramatically the default isometric scale used to give a sense of realistic depth to the projection. The user may have up to 2 concurrent isometric projects specified and is free to define in which direction each axis is to appear and the location of the origin, i.e. the projection of the point (0, 0, 0), to show the desired portion of the drawing.



$$\text{Isometric scale} = \frac{\text{Projected length}}{\text{True length}} = \frac{\sqrt{2}}{\sqrt{3}}$$

Figure 4.9 Isometric scale factors

In manual draughting an isometric scale such as that shown in figure 4.9 would normally be adopted if special isometric paper already marked with a scaled grid were not available. The draughtsman would draw up a small scale and obtain lengths with a pair of dividers. The process can be represented mathematically as follows:-

Assuming	(X,Y,Z)	is the 3-D point to be projected
	(Xo Yo)	is the isometric origin outlined above
	(Xp Yp)	is the projection of point (X,Y,Z)
	S	is the isometric scale

In the diagram overleaf it can be shown that:-

$$S = \sqrt{2}/\sqrt{3}$$

and then by conventional triangle geometry that:-

$$Xp = Xo + (X \times S \times COS\ (30^{\circ})\ ) - (Y \times S \times COS\ (30^{\circ})\ )$$

but $S \times COS\ (30^{\circ}) = \sqrt{2}/\sqrt{3} \times \sqrt{3}/2$
$= 1/\sqrt{2}$

therefore $Xp = Xo + \frac{1}{\sqrt{2}}\ (X-Y)$

similarly $Yp = Yo + Z + \frac{1}{\sqrt{6}}\ (X + Y)$

The above equations enable the projection of any point to be determined since for a given view the isometric origin is fixed.

4.8.3 Perspective Projection

The perspective projection of an object is somewhat more complex than the isometric projection since it is based on the laws of optics, generating a visually pleasing but non scale view of the subject. The mathematics governing this particular projection are complex, a full description of which are given by Yi(14). Indeed, the method of perspective projection used by the TIGER system is based on that described by Yi, chief differences being in the specification and display of a particular view, the transformation algorithms being essentially identical. Different views of an object are generated by moving the viewing position around the object in 3-D space and varying the distance from the object. This is achieved by two different methods, either specifying the various parameters manually to generate an exact view, or by using the joystick function to pan quickly around the object and generate rapid but less exact views. The function of the joystick mode is controlled from a small area of the menu and is designed to emulate the hardware joystick found in some large, and generally very expensive, refresh displays. By moving the digitising stylus up and down on the joystick area the user may increase or decrease the tilt angle of the view, by moving sideways the pan angle is similarly modified. On a refresh display unit the object would appear to rotate on the screen using this method, but this is obviously not possible with a storage unit. In order to be able to keep track of the current view the system displays a small set of axes in the bottom left hand corner of the view which may be redisplayed rapidly enough in non store mode to appear to move as the direction of view is changed. When the axes show the desired angle of view the user presses the appropriate button to instruct the system to generate the projection.

The display is arranged so that the line of sight passes through the centre of the view at what appears to be right angles to the surface of the screen.

It is recognised that in engineering applications perspective projects have little value except to give an overall impression of the appearance of some object or objects to the observer. By their very nature they obey no simple laws of scale and have no place where an accurate scaled drawing is required.

4.8.4 The Display File

It is not strictly necessary for a graphics system to support a display file, since all information is contained in its basic form as part of the 3-D database. Most early systems at Imperial College had no display file processor at all, the exception being Yi(14) who introduced a crude processor, for perspective projections only, into the original 3-D graphics system, although it was not implemented for more than simple redisplay of the workspace. However, a fully operational display file can have many uses which include:-

a) Finding existing data

All graphics systems should include the ability to return to an existing point or line by instructing the computer to search the current database for it. In order to locate the required point the user is normally expected to digitise a point that appears to be close to it on the display. The relationship between a point and its projection in a 2-D system is simple and a display file is of little advantage. However, with more complex mapping processes, such as those used in 3-D display, the relationship between a point and its projection may be somewhat less obvious. Although it is still possible to search the 3-D database and compute the projection of each point for comparison with the user supplied reference point, the process is considerably faster if the projection is already know, as it would be in a display file. On the assumption that the display file has some correlation with the 3-D data file it is then possible to refer back to the main database and retrieve the appropriate 3-D information. Furthermore, the use of a display file greatly reduces the amount of data that has to be searched since it only contains references to data that is currently visible on the screen. On a large drawing only a small part will typically be on display at any given time, and this is the only section of the database that need be searched, whereas without a display file it is necessary to check the entire 3-D database since there is no way of predicting which data may be relevant. The time penalty for such a process can be very considerable.

b) <u>Redisplay of modified data</u>

Many graphics processes, such as editing, result in small changes being made to the database which must subsequently be reflected in the display. It is extremely time consuming to have to recalculate the display of the whole database for one or two minor changes and it is convenient to modify both 3-D data and the display file in parallel, thus enabling redisplay to be handled more efficiently and considerably faster. This problem is naturally only relevant when using storage display units, since as on a refresh type display lines can be physically deleted from the screen, so obviating the need for complete redisplay.

The structure of a display file is in many ways dependant on the hardware for which it must function. The modern breed of 'intelligent' terminals incorporating a microprocessor capable of performing many display processes locally benefits from a different structure from that employed for a terminal entirely dependant on the host computer for its processing power. The former may even be capable of storing and modifying locally the display file, as a response to some user operation or a prompt from the CPU, and redisplaying the modified file without intervention from the computer. In the latter case the terminal will have no storage capability locally and the manipulation and redisplay of the display file must be handled by the computer. In a time sharing environment the former 'decentralised' approach is obviously advantageous since the load on the computer is significantly reduced.

For the majority of this project the display available at Imperial College was the Tektronic 611 storage unit, which is of the latter type, indeed it is one of the simplest devices available, being a purely analogue device dependant on external signals for both character and vector generation. At Imperial College this was achieved by use of suitable CAMAC interface modules. The display file developed consisted essentially of instructions for these modules with an indication of their origin with respect to the 3-D data file.

The later implementation of this for the Tektronix 4014 graphics displays was modifed to suit the different data requirements, this unit being on a standard terminal interface and expecting to receive data encoded as ASCII character strings. The modifications make the display file database somewhat larger and the display process somewhat slower due to the necessity to store data at a higher level, requiring decoding into the appropriate character strings before being despatched to the screen. In either case the method allows cross referencing between the display and data files in either direction, although this must naturally be handled entirely by the computer. In order to reduce processing time a method was devised whereby it is not necessary for the computer to scan and rewrite the display file every time a modification is made to the database. A 'modification table' of up to 10 changes is maintained and referenced by the display processor when constructing the final drawing. A change to the database simply necessitates making an entry in this table. In the unlikely event of this table becoming full the system automatically regenerates the display file to incorporate all the modifications and clears the table for new entries to be made. Any graphic operation which results in the drawing being redisplayed from its 3-D database, such as a rotation or shift, automatically regenerates the display file and clears the modification table. It has been found that this generally occurs quite frequently and the modification table rarely becomes full. However, if this does occur the latest display file processor is capable of regenerating the file in parallel with the users continued work, making use of the multi-tasking ability of the RSX-11M executive. In the single user system this was not possible and such an eventuality necessitated the user waiting while the new file was generated. All standard system routines that modify or add to the 3-D database generate the appropriate display file entries in parallel, a process that is completely transparent to the user. Applications programs should include this feature although failure to do so is not serious since the system regularly tests the status of the display file and will automatically update it if necessary, in parallel with the users current operation. In general this should not be necessary because the display file, although complicated in structure, is accessed by a suite of Fortran callable subroutines that may be readily incorporated in any program.

The TIGER system uses this file for many of its editing features and search algorithms, resulting in considerable time savings. Redisplay time after operations such as editing has been considerably reduced, and recomputation required cut dramatically.

4.9 The Workstation Microprocessors

The design of the workstations, consisting of an alphanumeric terminal, graphics terminal and digitiser, presented a number of problems in the multi-user system for which a solution was needed before effective operation was possible.

The principle by which the software functioned in the single user system was that the digitiser continuously transmitted its position to the computer, which then reprocessed this and sent it to the graphics screen in the form of a cross hair cursor, so enabling the user to monitor his position with respect to the drawing. It soon became evident that this arrangement was not satisfactory under the multi-user system. Not only did a number of digitisers sending information continuously to the computer cause a heavy I/O overhead, but also the reprocessing of this information and retransmittal, in parallel with any other tasks that were running, resulted in a very poor flickery cursor and a slow response when a button was pressed. Perhaps most annoying was not that response was slow but that it was unpredictable, depending heavily on the loading of the computer from second to second.

The three workstation devices were separately connected to the computer on individual lines, thus multiple workstations required a considerable number of ports in the computer. The software simply treated each line as a separate terminal and addressed each individually, the terminal number of the digitiser and graphics screen being deduced from the number of the alphanumeric terminal. This of course meant that each user station must obey a certain numbering convention, which could prove awkward for future expansion. Problems were experienced with user stations located some distance from the computer, since the cost and inconvenience of routing three cables to each is considerable. Furthermore, it effectively precluded any possibility of installing remote user stations connected to the computer via dial-up lines through the telephone network, since this would require three lines and three modems at each end, which would scarcely be a workable arrangement.

When a system is subject to development over a number of years it is obvious that, with advances in technology, any new user stations purchased should use the latest equipment to obtain optimum performance. This presents the host computer with an enormous problem when faced with a number of different types of workstation, and would necessitate considerable software effort to enable it to handle these in parallel, and could result in a considerable rewriting effort whenever a new piece of equipment is purchased. It is therefore extremely desireable that every workstation presents a common interface to the computer, with any differences being handled at the workstation. This feature enables, for example, fully equipped workstations with large digitisers and screens to be run in parallel with 'desktop' workstations consisting of a small tablet and display, costing considerably less and being ideal for simple editing functions.

The time taken to display a complex drawing on the screen is considerable and constitutes one of the major bottlenecks in the drafting process. Although the problem is alleviated by display files and selectively erasible screens it can still result in a considerable amount of non-productive time. This is due almost entirely to the speed at which data can be transmitted to the workstation, normal serial terminal interfaces run to an effective maximum of 960 characters per second, and rather less over dial-up telephone lines. Very much higher rates can be achieved with parallel interfaces, but these limit the distance between the user station and the computer and cannot be used over a dial-up line. To minimise display times it is therefore desireable to reduce to a minimum the number of characters transmitted. Early systems were heavy on data transmission since operations such as drawing a circle were performed in the host, which transmitted it as a series of short straight lines. It is obviously desireable if this can be condensed to the minimum, ie. a centre point and radius in this case for transmission, and expanded locally at the workstation. This has the dual effect of substantially reducing the amount of data transmitted and relieving the host computer of the processing necessary to construct the circle. This principle can then be applied to other regular and geometric shapes, such as conic sections in general, text, grid nodes etc..

The obvious solution to all these problems is to incorperate a local microprocessor at each workstation dedicated to the task of running that workstation and performing all the necessary local functions. The search for a suitable microprocessor must be based on a number of considerations, principally the cost and capability. On the cost side it must be remembered that one complete microprocessor is required for each workstation and in order to keep the cost to a realistic level in comparison with the rest of the equipment and its expected life, it should cost no more than a few thousand pounds. Functionally, the microprocessor must be fast enough to cope with data at the maximum rate the host can transmit, with some spare capacity to enable local computations to be performed, such as the drawing of arcs, without having to block further transmissions and so neutralise the performance increase. The processor selected for this function was the now well eastablished Zilog Z80A, a 4MHz 8 bit chip. The configurations used at Balfour Beatty and Imperial College differ considerably, the former being a commercially available package based on the S100 bus system, and the latter being built entirely in the workshops. However, they are sufficiently compatible that the same program runs on both with only minor modifications to the I/O functions to cater for different interrupt handling. Each system consists of the following essential components:

1) Z80A CPU based processor
2) 4Kb RAM
3) Up to 32Kb EPROM
4) 4 x serial interfaces

Program development was carried out on a similar but larger system with floppy disk drives for program storage, larger RAM for testing and other programming functions, and an EPROM programming card to store the final program in EPROM for use in the workstations, which start automatically on power up. The four serial ports are used for communication with the host computer and each of the workstation devices.

4.9.1 The Microprocessor Software

The microprocessor had two distinct modes of operation, non-graphics and graphics. In the former mode, which is entered on power up and whenever the workstation is not in use for drafting, any characters received from the host are sent directly to the alphanumeric terminal and vice-versa. The graphics screen and digitiser are ignored. This gives the user the impression that the alphanumeric terminal is connected directly to the host and allows logging on, program development etc., to be carried out as at any normal terminal. Graphics mode is entered as a result of a special character sequence received from the host.

Graphics mode is active when drafting is in progress. In this mode the microprocessor expects to receive one of three types of data from the host:

a) Characters for the alphanumeric terminal
b) Characters for the graphics screen
c) Requests for a function and associated data

In order that the microprocessor can discriminate between valid characters and unsolicited information, such as system messages, each block of text is started and terminated by ASCII STX and ETX characters respectively. In this way characters received between ETX and STX are known to be unexpected and are sent directly to the alphanumeric terminal, except when the microprocessor has been told by the host to block such information, when they are discarded. Any expected characters are routed directly to the appropriate device, the routing being controlled by functions requested by the host. A function is identified by a 'function request character', currently set to ENQ since this is not otherwise used by the RSX-11M operating system, and a second character defining the exact function. This may be followed by further data characters depending on the function. Typical functions are:-

1) Redirect subsequent output to graphics screen
2) Draw a circular arc
3) Erase graphics screen
4) Start monitoring digitiser
   etc...

There are currently some 40 of the available 96 functions implemented.

In addition to sending characters from the input stream to the correct device the program optionally receives characters from the alpha-numeric terminal and transmits them to the host, when requested to do so. At other times the terminal is blocked to avoid the user typing unexpected characters that could be misinterpretted. Furthermore, when input has been requested the characters typed are buffered locally within the microprocessor, until a line terminator is entered, and processed locally to remove backspaces etc. so that the idio-syncracies of a particular operating system, for example whether it expects DEL or BS to rub out the last character typed, are transparent to the user.

A major function of the microprocessor is to monitor the digitiser and display a cross hair cursor on the screen, according to the current window parameters sent to it by the host. This function greatly reduces the load on the host since it is no longer showered with data and simply waits to receive a reply from the microprocessor. This reply is either in the form of a coordinate or a menu or macro square and area number, together with the button that has been pressed, enabling the host to take the desired action. The rate at which the microprocessor can receive information from the digitiser and draw a cursor is far greater than was possible when this was handled by the host, resulting in a very much smoother cursor and a faster response when a button is pressed.

All I/O processing in the microprocessor is handled asynchronously by interrupts, thus giving the maximum possible throughput by eliminating polling loops. Characters are taken from, or put into, buffers by the interrupt service routines, and moved about by the basic background loop which executed continuously except when a function is being performed. Most I/O requires very small buffers due to the relative speeds of transmission and processing, but a large input buffer is maintained for characters received from the host to avoid delays when the microprocessor is not in a position to process received characters, such as when a lengthy function is being executed. Buffer overflow is avoided by the use of standard X-ON/X-OFF protocol.

By use of microprocessors the workstation configuration can be made almost transparent to the host computer. A large part of this is achieved by using the functions. For example, the host may issue an 'erase graphics screen' command which is universal to every user station. Within the microprocessor, however, the function will be handled differently depending on the terminal type. Under some circumstances it is necessary for the host to know the basic user station configuration. For example, after an item has been deleted from the drawing database by an editor it is necessary to remove it from the screen. This is easily achieved on terminals with selective erasure capability, but on storage screens it is necessary to erase the entire screen and redisplay what remains of the drawing. This fundamentally different approach must be handled by the host computer. To achieve this the microprocessor sends out, on receipt of the 'enter graphics mode' command, a string of identification characters which are read and stored by the host. These can be referenced by any program and contain all the information necessary for making decisions such as the above.

5.0 THE DRAFTING PROCESS

Whilst the previous chapter was intended to outline some of the basic principals and concepts underlying the graphics system, this chapter is designed to give an insight into how they are combined together to form program units capable of performing the many and varied functions required of a comprehensive graphics system. Although programming examples may be included it is not intended to give a detailed description of the programs, but merely to outline the facilities that are available.

5.1 The Log On Procedure

Before the user can have access to the graphics system it is first necessary to log into the RSX-11M operating system in the normal manner from the alphanumeric terminal at the desired user station, necessitating all users to be familiar with the standard RSX login procedure. The possibility of modifying RSX to allow access to graphics without the need to log on in the conventional manner was considered. Eventually it was decided not to implement such a change because it would make the RSX-11M executive non standard, which was both against the basic philosophy of avoiding such changes that could lead to future problems, and furthermore with the rapid growth of computing in industry it was felt that the draughtsmen may have need of the computer for other functions and would therefore be familiar with the necessary procedure.

The graphics is initiated by issueing the command 'LOG' on the alphanumeric terminal. A number of set up procedures are then performed, as described in chapter 4, the principal function being to determine the current user station number, set up the users random access files and initialize the Tektronix display. A number of parameters must then be supplied by the user to identify himself and the job he is about to perform. The majority of this information is necessary for the accounting system employed at Balfour Beatty and is of no relevance to the correct functioning of the graphics. It has been included in both the Balfour Beatty and Imperial College systems for compatibility only. The information that is required is as follows:-

1. Users Name - up to 14 characters
2. Contract number - the job to which the session is to be charged
3. The drawing number - up to 22 characters
4. Users time sheet number)

5.   Cost code section number) Balfour Beatty Accounting
) information
6.   Cost code activity number)

In the early multiuser system a prompt was issued requesting the user to enter manually each of the above pieces of information every time a new session was started. This is a lengthy process, particularly since draughtsmen are not normally experienced typists, and generally involves them in entering non variant information, such as their name. An automatic method of entering such information is therefore desirable. The following method was devised to overcome this problem.

When the LOG command is issued the users directory is searched for a file called GRAPHICS.LOG, which if found should contain the default information required. If not found a prompt is issued for each piece of information individually. The basic structure of this file is 6 separate lines containing the default information in the above order. If there is no default the line must be replaced with a single '?' character, indicating that the user is to be prompted for that piece of information. Comments may be inserted in the file either on a line of their own or following a piece of information providing they are preceded by an exclamation mark. In the following example the user would be prompted for the contract number and the drawing number.

```
!         An example log on file showing
!         How defaults are established
!
THOMPSON  ! users name
?         ! prompt for contract number
?         ! prompt for drawing number
123       ! timesheet number
10        ! section number
15        ! and activity number
```

It is important that this file contains nothing but true defaults because the user does not have the option of overriding any parameters that are explicitly defined.

In most circumstances there will be some information that is always non variant, and some that is non variant for each particular job. The user may create any number of additional log on files with names of his own choice to include this second class of information. If, for example, the user were to create a file called POWER.LOG he can ask for the information in this file to be considered simply by issuing the logon command 'LOG POWER'. For each piece of information the logon routine first searches file POWER.LOG. If it is undefined, i.e. replaced by a '?', the default file GRAPHICS.LOG is tried, and if this fails to yield the information a prompt is issued for the user to enter it manually from the terminal.

The drawing on which the user is to work is identified uniquely by the drawing number, which is actually a character string of up to 22 characters and not simply a number; the name merely reflecting a convention used by Balfour Beatty. When the log on information is complete the system checks to see if the drawing number matches one that has already been saved (See section 5.10) and if so automatically recalls that drawing. If not the user is informed that a new drawing has been assumed and the full default parameters established. In either case when this operation is completed the log on procedure is finished and the system enters a neutral state known as 'Background Mode'.

5.2 Background Mode

The idle state of the graphics system is normally referred to as 'background mode', and is the state entered whenever a function has been completed and further instructions are required from the user. The handling of background mode in the multi user system is significantly different from the earlier single user system, due principally to the need to conserve core in the former case, but from the user's point of view there appears little difference in operation. The principal function of background mode is to monitor the digitiser and check to see if a button has been pressed. For the benefit of the user a non store cursor is displayed on the graphics screen, when over the drawing area of the digitiser, indicating the current position of the stylus in relation to the drawing. In the single user system a number of messages were also displayed, but in order to remain consistent with the policy outlined in chapter 4 of keeping the graphics screen free of all but graphical information this feature was dropped from the multi user system in favour of prompts output to the alphanumeric terminal, which was not previously available.

Two different modes of cursor display are available, standard mode where the positional relationship between the cursor and the drawing are maintained irrespective of the display window, i.e. the part of the drawing that is currently being displayed, and non scale mode where the cursor always traverses the width of the screen when moved the width of the digitiser irrespective of the size of display window. The latter function is very useful when working freehand with very small items since the accuracy can be greatly improved. In standard mode very small movements of the stylus result in large movements of the cursor and accurate positioning is difficult. The cursor is held on the screen by the local microprocessor, as described in chapter 4, so relieving the host computer of much of the work performed by background mode in the single user system.

Background mode will normally remain in control until the user presses one of the buttons on the stylus. In the single user system the logic to process the request was included within the background overlay, the size of which was unimportant, but in order to ensure that the minimum of core is occupied by the multi-user system when idle control is immediately passed to a second task to check the button and coordinates that have been output.

Due to the multi-tasking capability of RSX-11M it has been possible to introduce a very useful feature to background mode. When a coordinate has been digitised and control passed to another task to process the information, background mode does not exit, but passes into a semi dominant state, monitoring the current activity every few seconds. This function is intended to provide a number of features, not least the ability to detect errors that cause tasks to exit prematurely and would otherwise terminate the graphics entirely. By monitoring which tasks are running, background mode can quickly detect such a failure and take remedial action. It is anticipated that this feature may be expanded to include regular backup of drawing and other sensitive data to guard again loss of work caused by any type of failure. It was felt that the benefits of this facility far outweigh the small penalty on core usage introduced by keeping background mode running at all times. A small change to the task loader was necessary to incorporate this function, which is transparent to the general programmer who still queues background mode as any other task. When a failure is detected background mode automatically starts up a special error recovery task, which resets sensitive system parameters, checks and unlocks files that have not be properly closed, and returns control to the user in the best possible state. A full description of error processing is contained in section 5.11.

If the digitised coordinate is found to be over the menu or symbol areas of the digitiser all buttons, except buttons A and B, are treated similarly and envoke the appropriate menu command. Over the drawing area each button has a specific function, and the user must be careful to press the correct one. These functions are as follows.

5.2.1 Button Assignments

a) Button 0 - Verify current position

This button is used simply to mark the users current position on the graphics screen, and the projection of a 3-D point that would be defined if one were entered at this position. The use of this button may not appear immediately obvious but it becomes important where corrections such as control mode or grid roundoff are in operation since it reflects the point after correction and not as the user sees the cursor on the screen. The function of these facilities is explained fully in a subsequent section of this chapter. The user may check, for example, that the point will be rounded to the correct grid node before entering it, which is particularly important if the density of nodes dictates that only the boundary nodes are displayed.

b) Button 1 - Input 3-D coordinate

By pressing button 1 the user instructs the system to read the current digitiser position and convert this to a true 3-D coordinate using the principles described in chapter 4. This coordinate is normally stored in the workspace as the next sequential entry, and the line joining it to the last entered point is displayed on the graphics screen. In some cases this coordinate will be passed to another routine for further processing, typically as input data for symbol generation. This straightforward method of digitising single coordinates is the simplest method of data input to the TIGER system.

c) Button 2 - Break Line

Button 2 functions in exactly the same way as button 1 except that it records the point with the 'pen up' attribute rather than 'pen down'. This implies that the point is the start of a new line rather than a continuation of the previous, and no vector is drawn to it from the last point.

d) Button 3 - Define Trailing Coordinate

The method of 3-D input from 2 or more orthogonal views has already been explained. This consists essentially of picking up the same point in two views in order to define it fully in 3 dimensions, since each view is only capable of specifying 2 of the 3 space coordinates directly. The coordinate that is not directly specified depends on the view over which the digitiser stylus lies, e.g. Z in an X-Y projection, and is known as the trailing coordinate. In order to modify its value the user presses button 3, whereupon the system outputs the prompt message 'Set Trailing Point' and reverts to background mode. The next point input from the digitiser, normally with button 1, is used to update the stored trailing position, i.e. if the point is over an X-Z view a trailing X and Z value will be defined for subsequent 3-D points that may digitised. Further input over an X-Y view will define X and Y explicitly, and the new trailing Z will be used to complete the coordinate.

e) Button 4 - Window Selection

Button 4 differs from the majority of buttons in that it implements a function more akin to the menu type of command and causes control to pass from background mode to a second service task. This is because the function of the button is too complex to be handled locally, but being a very common feature is located at the user's finger tips for convenience.

The principle of display windows has already been outlined in chapter 4. In essence they provide the user with the means to 'zoom in' on certain areas of drawing to examine them in more detail. Typically a large drawing will have a number of areas of interest and it is convenient for the user to have the ability to store different display windows covering each of these areas, with the option of changing from one to the other without the need to redefine each every time it is required. The existing system did not allow for the storage of more than one window definition, and was very inconvenient to use, particularly when working from freehand sketches where the new window to be displayed does not necessarily overlay the current one, i.e. where the user cannot see the information that he or she is trying to zoom in onto. The method developed for the TIGER system allows up to 15 windows to be defined and stored under one of the stylus buttons. Having initiated the window function with button 4 the user is asked to press one of the 16 buttons corresponding to the desired window. Only 15 windows are allowed because button C functions in its standard 'cancel' mode, as explained later. The window assigned to the chosen button is then recalled and the drawing redisplayed.

The use of button 4 implies that the desired windows have previously been set up. At log on each window defaults to a standard area of the digitiser, but these definitions may be changed by menu command. The use of standard windows is encouraged because it enables draftsmen who may be unfamiliar with a drawing to find their way around it, and has functional advantages in connection with the display file. A window is essentially an imaginary rectangle over an area of the drawing indicating which part is to appear on the screen. This rectangle has a fixed horizontal to vertical ratio of 4:3, corresponding with the shape of the standard display screens, and is defined by digitising each of the bottom corners, although since the rectangle must lie horizontally only the x value of the second corner is significant.

f) <u>Button 5 - Enable or Disable Grids</u>

A grid facility exists whereby the user may set up a pattern of grid nodes covering all or part of the drawing to constrain the input of points to coordinates that lie at grid nodes only. The facility is extremely useful when, for example, laying out pipe supports at equal and known intervals along a pipe run. The graphics system allows grids to be defined independantly along each of the 3 orthogonal axes; a complete description of the definition and use of grids may be found in section 5.2.2. It is convenient to reserve button 5 to enable or disable grid round off depending on the current state. If button 5 is pressed when grids are disabled they become enabled on the axes for which they have been defined, and conversely if they are currently enabled the action of button 5 is to disable all grids. In either case a message is output to inform the user of the current state of grids and indicate on which axes, if any, they are currently enabled. Whenever a drawing is completely redisplayed on the screen the grid nodes are only drawn if grids are currently enabled. This provides a means of keeping the screen free of nodes if not desired, such as when taking a hard copy. Whenever grids are enabled, either on mass with button 5 or individually from the menu, a check is made to see if grid nodes are currently on display, and if not they are displayed automatically. This obviates the need for the user to redisplay the drawing again in order to bring up the nodes.

g) Button 6 - Enable or disable control mode

Control mode provides the means for constraining points to lie in predetermined directions from the last point that was digitised, and is explained fully in section 5.2.2. Button 6 is a flip-flop type switch that either enables or disables control mode, depending on its current state. A message is output to the alphanumeric terminal to inform the user of the new state and the control angle that is currently in force.

h) Button 7 - Find Point Mode

It is obviously of great importance for the user to be able to place a point exactly coincident with a previously entered point in order that, for example, a boundary made be closed precisely. With normal freehand digitising it is extremely difficult to achieve the required accuracy, due chiefly to the poor resolution of the graphics screen, and a small error in the position that may be undetectable at this stage will result in an unsightly and intolerable error on the final plot. The problem is further compounded if the graphics database is to be processed to yield, for example, a control tape for an NC mill, where the error will be reproduced on the final object. It is therefore essential that the graphics system includes a means of locating previously entered points or lines and returning exactly to them. The former facility is provided by this button.

Button 7 functions in the same way as button 1 except that, having decoded the position, it passes control to a separate task to perform the find operation. The find facility is based on the display file, and it is therefore not possible to find a point that is currently not displayed on the screen in any projection. The search will locate the point that appears to lie closest to the digitised point. This is done without reference to the database and is purely two dimensional in nature, thus two points that may in effect be widely spaced in 3 dimensions will be considered close together for the search procedure if they appear close together on the screen. The user must make judicious use of the various views currently available to enable a point to be found quickly and accurately, since a point may be located from any view in which it is displayed, although some may be unsuitable if it appears close to, or coincident with, another point. This method of searching has a number of advantages over existing routines that were based on searching the entire database. The latter process is wasteful in that it is impossible to discriminate between data that is on display and that which is not, thus lengthening the search procedure considerably. This shortcoming was tolerable in 2-D systems since there was a simple relationship between the database and the information on display, but the addition of the 3rd space co-ordinate has introduced more complex display techniques necessary to display the same data in different views, and would add considerably to the computation necessary for a find type facility. By considering projections of a point it is, as previously indicated, not necessary for the actual point and the search point to be close in 3-D space, so obviating the need for the user to manually change the trailing point, using button 3, to bring the points close together again if the trailing value has changed. Furthermore, the find facility can be very useful for returning to a previously defined trailing value quickly and simply without the need to use button 3. Find point mode locates the closest point in the display file and by a cross reference process can immediately access the correct point in the drawings database and return its true 3-D value. Thereafter the coordinate is treated as if had been entered manually with button 1.

i) Button 8 - Find Line

Button 8 is similar in operation to button 7 except that instead of locating an exact point the system computes the line that appears closest to the user specified search point and places the new point exactly on that line at the closest position to the search point. The principles involved are similar to the previous form of find except that the point returned will not normally be one that already exists, but rather one that is computed from the end coordinates of the appropriate line.

j) Button 9 - Insert Fillet

This button allows the user to insert a fillet between two straight lines, providing either buttons 1 or 2 are used on either side of the 9 to define the lines. Two modes of operation are currently possible, default or computed radius, depending on the number of times button 9 is pressed. In the former case when button 9 is used once its location is taken as the intersection point of the two lines and a fillet of default radius constructed, the lines being truncated appropriately. The default radius may be reset by menu command. If the button is used twice in succession the lines ending and starting at the two points respectively are calculated, and the smaller radius fillet inserted that is tangential to the lines at one of the points digitised.

Button 9 is different from most buttons in that its function is not executed until after the next button has been pressed, defining the end of the second line.

k) <u>Button A - Floating Assignment</u>

Certain sequences of operations may result in the need to envoke a particular menu command a number of times, and for the users convenience button A has been reserved so that it may be 'programmed' to emulate any of the menu or macro commands. In order to assign a particular function to the button it is merely necessary for the user to place the stylus over the required square and press button A. Thereafter, pressing the button over the drawing area will envoke that command. The assignment remains in force until the user reassigns the button by specifically digitising over a different command with button A. This is true for both menu commands and macro commands, thus enabling a frequently used macro to be recalled quickly, even if it is on a different page from the current menu or macro page. For a fuller description of macro handling refer to section 5.4.

l) <u>Button B - Floating Assignment</u>

This button behaves in exactly the same way as button A except that it has an initial default setting of 'Break Line'. This function differs from button 2 in that it indicates that the line is to be broken at the next point without actually entering that point. This is important if, for example, it is desired to 'Find' the next point, as button 2 will not allow this. The definition of the button may be changed to any command if the user so desires.

m) <u>Button C - Cancel Operation</u>

This button provides the ability to prematurely terminate any operation, normally as the result of the user accidentally envoking the wrong command or entering bad information. A general tidy up of the system is performed and background mode reenvoked. Unlike the other button assignments, which as described normally only apply in background mode, button C is universably applicable and can be used to cancel most operations. The user must become aware that some operations are irreversible and once started cannot be prematurely terminated. In this case button C will not be effective and some later remedial action will normally be necessary, such as the use of one of the editors described in section 5.5.

n) <u>Button D - Unassigned</u>

Button D has yet to be assigned a function.

o) <u>Button E - Numeric Entry</u>

The input of accurate lengths from the digitiser is reliant on the user working from an accurate drawing or envoking one of the round off facilities, such as grids, described herein. In many cases it may be more convenient for the user to enter the coordinate numerically if an exact space coordinate is known or does not readily suit one of the correction methods. Button E provides this facility. On pressing the button the user is prompted on the alphanumeric terminal to enter the X, Y and Z values of the required coordinate, and define whether this is to be an absolute position or an offset from the previously entered point, the latter being particularly useful for entering lines of known length. The point so defined is then displayed as if it had been digitised normally. This method of entry is very useful in conjunction with button 3 for defining a precise trailing coordinate, particularly in '2½-D' type applications where the 3rd coordinate is constrained to lie on a number of fixed planes. Furthermore, coordinates may be entered that lie outside the current drawing area.

p) Button F - Finish Input

This button is used to indicate the end of input to 'open ended' functions, such as duplication, that accept a variable number of points. The action of this button will normally be established by such a task, and is otherwise ignored.

5.2.2 Coordinate Corrections

Whenever a coordinate is entered it is subjected to a number of tests and corrections before it is finally despatched to the requesting task or placed in the database. These are normally constraints specified by the user to ensure that accurate data results from a fairly roughly digitised drawing. The system currently provides the following corrections.

a) Drawing Skew

Skew correction is particularly important when digitising from an existing drawing. It is obviously necessary to ensure that the axes on the drawing coincide with the axes as the digitiser, particularly on a 3-D drawing where positional relationship between projections is important, and to do this manually would involve very careful positioning of the existing drawing to ensure that it lies absolutely horizontal. The skew function obviates the need for this by allowing the user to digitise a line on the drawing which is to be considered horizontal, a function which must be initiated manually from the menu. Thereafter each point that is entered is modified according to this specification so that it will coincide correctly with the existing drawing. This is essentially a 2 dimensional correction, being applied to the coordinate output from the workstation rather than the computed 3 dimensional coordinate. The remaining corrections are applied during or after this computation process.

b) Control Mode

In many drafting applications objects will consist of lines either exactly horizontal or vertical, or at known angles to one of the axes. For example, a pipe run will typically consist of horizontal and vertical pipes jointed by 90 degree elbows. In order to facilitate the quick and accurate drawing of such lines a control mode is provided that constrains points to lie on a line at a multiple of a previously specified angle from the last point that was entered. In the above example the angle would be 90^o, which constrains the point to lie in only one of 6 mutually perpendicular directions in 3-D space. Whenever a point is digitised control mode, if active, determines to which of the allowable directions it best approximates, and rotates the new point about the previous to bring it exactly onto this direction. This means that the point as displayed on the graphics screen will no longer necessarily coincide with the position of the cursor, and the user should use button 0 to verify the corrected position before entering the point if in any doubt. In the single user system control correction was also applied to the cursor, but this has been dropped from the multiuser system because of the difficulty of implementing it in the microprocessors. The user may change the control angle at any time during the drawing process, even when control mode is active, whereupon the new specification becomes applicable immediately.

c) Grid Roundoff

The grid facility enables the user to define a pattern of 'nodes' in 3-D space to which any coordinate entered will be rounded. Unlike earlier systems, where the grid was of a two dimensional nature relating to the position on the digitiser rather than the final coordinate, the advent of the three dimensional system necessitated the introduction of a truely 3-D grid facility, since the 2-D function cannot correctly take account of the relative position and scale factors applying to different views. Early systems would only handle uniform grids, i.e. one where the distance between every node is identical, and covering the whole of the drawing area. The current implementation will handle uniform or notional grids, the latter consisting of non equispaced nodes with separations specified by the user, covering only the required part of the drawing. Furthermore, different grid spacing, types and limits may be specified for each of the three major axes, and roundoff enabled independantly on each axis. A number of menu commands exist to define a suitable grid pattern and are of an interogative nature requiring user input on the alphanumeric terminal and the digitiser. Grids may be enabled on specific axes by menu command, or globally enabled and disabled by use of button 5. Whenever any grids are enabled and the drawing redisplayed on the graphics screen the projection of the nodes is marked with a small '+' symbol for the benefit of the user. This does not constitute part of the display file and is not used for the roundoff procedure, which is based on the 3-D coordinate.

Experience showed that with small grid spacing and large windows the density of nodes displayed was often so great that the drawing was practically obscured, and a significant delay was introduced while these nodes were displayed. To overcome this, two different modes of grid display were developed, standard display and boundary display, which may be optionally selected by menu command. Standard mode provides the more familiar display of every node, except when the density exceeds a predefined value, when boundary modes is implemented automatically. In boundary mode only the outer nodes of the grid on each of the edges are displayed, or the last nodes to fall within the window, depending on the current display window setting. This effectively overcomes the two disadvantages of standard mode outlined above while maintaining the advantages of grids, since non-displayed nodes can easily be located by lining up the arms of the cross hair cursor on the screen with the edge nodes.

As with control mode, grid roundoff is not reflected in the cursor displayed on the graphics screen, but its effect may be checked using button 0. Control correction and grid roundoff are mutally exclusive functions since each is destructive to the effect of the other. Only one may therefore be enabled at any time, and an attempt to enable both is reported as an error and ignored.

5.2.3 Processing the Coordinate

Whatever combination of corrections are active or buttons are pressed the ultimate aim of background mode is to generate a three dimensional coordinate that is needed as data for some process. In the simplest case where the user is joining points together with straight lines the processing of the point will be handled locally. In more complex applications, such as generating a circle based on digitised coordinates, the point will be passed to the circle generating task to use as basic data for computing the required orientation and radius.

In either of the above cases one or more points will eventually be entered in the database, the new information displayed on the graphics screen, and the display file updated to suit. In chapter 4 the basic structure of the database records was outlined. The normal record consists principally of a three dimensional coordinate, X, Y, Z, and an I code to indicate the meaning of that record. Table 5.1 indicates the principal I codes used by the current graphics system. In general all tasks that examine the workspace only take account of records with I codes they are programmed to recognise, and ignore any others. In this way the specification of further values, which may be in the range 0 to 255, will not affect the performance of existing routines. The full record also contains the additional indexed and masked parameters described in chapter 4. In the current implementation these are as follows.

I CODE	DESCRIPTION OF ASSOCIATED RECORD
1	The start of a line in either 2 or 3 dimensions. Plotter/display moves to this point. or A positional reference point if within a symbol block.
2	The end point of a line in either 2 or 3 dimensions. Plotter/display moves to this point with pen lowered.
3	The first record and low limits of a symbol block. May be 2 or 3 dimensional depending on the symbol.
4	The last record and high limits of a symbol block.
5	An identification record within a symbol block bearing numerical data only.
6	A text record containing 12 encoded ASCII characters.
7	A general record of unspecified format.
13	The first record and low limits of a macro block. May be 2 or 3 dimensional depending on the macro.
14	The last record and high limits of a macro block.
15	An identification record within a macro block.
16	A text node within a macro - for future development.
255	A null record, to be ignored by all functions.

Figure 5.1. The I codes currently used by TIGER

a) Pen number

Most modern plotters have 4 pens available which may be assigned different colours or line thicknesses depending on the particular application. The former assignment is useful for mapwork, but for engineering applications different thicknesses are more useful since engineering drawings are rarely coloured and if so would be difficult to duplicate by the means currently available. Whenever a record is entered in the database the current pen number is associated with it, which may have a value of 1 to 4. At log on the default pen is number 1, but the user may readily change the current pen by use of the appropriate menu command. The new assignment remains in force until it is physically changed again by the user. The effect of different pen numbers cannot be observed on the graphics screen since this is incapable of generating different colours or a full range of line thicknesses. The result will not therefore be observed until a final plot is obtained on the flat bed plotter. A number of graphics functions automatically employ a fixed pen number irrespective of the pen currently being used for input. For example, blockfilling is always done with per 4, by convention the thickest, for efficiency. This operation is transparent to the user, and does not affect the pen number that has been specified by him for input. The current convention for pen thickness is as follows, all pens being black unless otherwise specified.

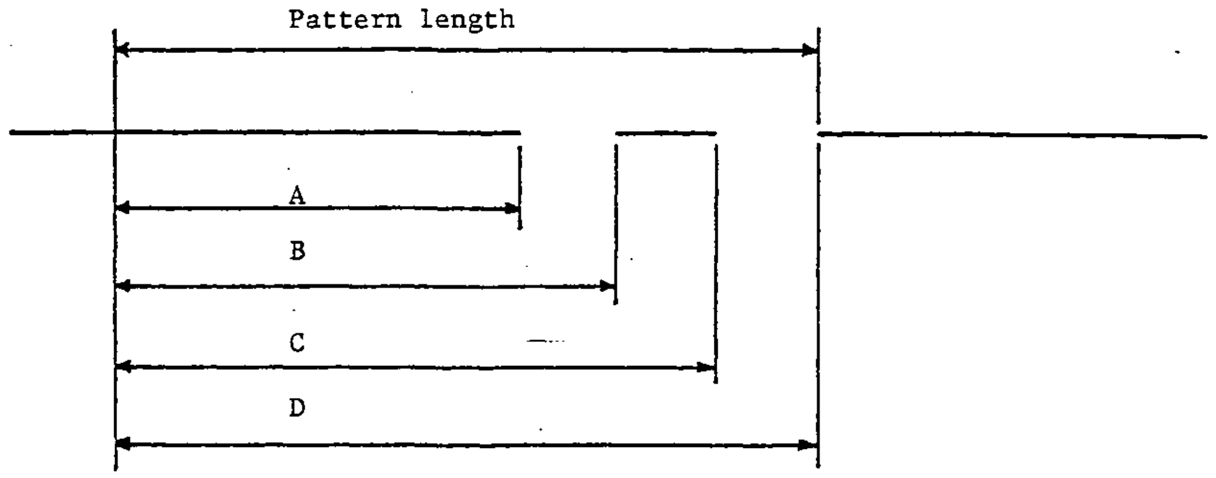
Pen Number	Thickness
1	0.3mm
2	0.5mm
3	0.7mm
4	1.0mm

b) Line type

A drawing may be composed of up to 16 different types of line, the simplest of which is a solid line, the other 15 being different combinations of marks and spaces to form line types such as dotted, dashed or chain dotted. Indeed the above 4 types comprise the defaults available at log on time, of which solid lines are used unless otherwise specified. Each line type is a repeated pattern of two marks and two spaces of varying lengths. The user may assign any of the remaining unspecified line types to form special patterns applicable to a particular drawing. Figure 5.2 shows the method of defining this pattern. By menu command the user may alter the line type used for further input until altered again. Unlike the above example with pen number the effect of different line types may be immediately observed on the graphics screen. A line type is continuous from the start of a line to the last point, irrespective of the number or spacing of intermediate points. In this way a pleasing effect is obtained and constructions such as dotted circles are possible when circular arcs are treated as a large number of short straight lines.

c) Display control

The display control parameter is unique in that it can not be set by the user, who need never know of its existance. The function of the parameter is to indicate to the system whether, boardly, the record is true three dimensional data or two dimensional information such as annotation. The distinction is important for operations such as display since 3-D data will be projected into all available views whereas 2-D data is assumed to obey no view characteristics and is displayed directly. This is also necessary for operations such as scaling or rotation to ensure that the correct transformation is applied. In general a three dimensional record will have X, Y and Z significant whereas a two dimensional record will only have X and Y significant, and in the latter case these will be apparent digitiser coordinates rather than a space coordinate.



LINE TYPE NUMBER	A%	B%	C%	D%	OVERALL LENGTH (MILLIMETRES)	DESCRIPTION
1	100	0	0	0	undefined	Continuous
2	25	50	75	100	7.0	Dotted
3	25	50	75	100	15.0	Dashed
4	70	80	90	100	30.0	Chain dotted

Figure 5.2. Line Type Structure and Convention

d) Input Overlay

The overlaying process provides the user with the ability to logically divide a drawing into up to 128 different sections or overlays. The name of the process is derived from its most frequent use, namely overlaying a common section of a drawing with a number of additional but independent pieces of information. A typical example may be site layout in the building industry. The common overlay would be the basic site plan, over which could be overlayed the various services such as electricity, drinking water etc. By including different combination of overlays a number of different drawings may be produced from one basic master. In the TIGER system the user may specify any of the 128 different overlays for current input, and any combination of overlays as 'active' or 'passive'. Active overlays are those that are currently of interest to the user and are available for display, manipulation etc. Passive overlays are those deemed not to be of relevance, and although part of the drawing database are not eligible for modifications etc. unless changed by the user to an active state. The setting of active and passive overlays is by menu command, the default at log on being all active with input on overlay number 1. Whenever a record is entered into the database its overlay is also considered. This is the first of the 'indexed' parameters referred to in section 4.5, and as such the database handler only makes note of its changes, from which can be deduced its value at any particular record. The indexing process means that in theory the overlay number could have any of some 65,000 different values, but it was decided that, for programming and user convenience, 128 would be adequate.

e) Select Status

The select status of a record indicates to a large number of processes whether or not that record has been marked for some sort of operation, for example rotation or scaling. The select status can only have three values, either 0 for not selected or 1 or 2 if selected, the distinction being between temporary and retained selected status, which are described fully in section 5.6. It has been stored as an indexed parameter for speed of operation to enable functions to jump through the drawing acting on selected data only, which can be achieved more efficiently with indexed parameters. The select status of a record differs from the other associated parameters since it is always entered with a value of 0 and generally modified at a later time as this becomes necessary. The remaining parameters are stored with values that are rarely changed.

In most cases the point digitised will not be entered in the database directly by background mode but passed to a second task to use as input data, the most common being one of the symbol generators.

5.3 Symbol Modes

With the facilities provided by background mode alone the user is limited to the production of drawings consisting solely of straight lines between specified points. It is obvious that this is inadequate for most engineering drawings which require more complex constructions such as circles or polygons. To this end the graphics system incorporates a number of 'symbol' modes which allow the user to input these items based on a number of pieces of key information, such as the centre point and radius of a circle. In general symbol modes handle geometric constructions that are of a known mathematical definition, each of which always conforms to certain rules. For example, a circle in 2-D space is completely defined by its centre point and radius. A typical symbol will be constructed from points digitised by the user and additional information, such as the number of sides on a polygon, obtained via the alphanumeric terminal. The symbol task may freely communicate with the alphanumeric terminal to issue the appropriate prompts and input the user responses, but access to the digitiser must be achieved via background mode. This ensures that the full facilities of background mode, such as grid roundoff, are available to the user even when entering points to construct a symbol, and further ensures that core utilisation is efficient. A simple process exists to allow a symbol, or any other task that needs digitiser input, to receive the data entered on the digitiser. The initial entry to the symbol task is by menu command, when the task will normally identify its function on the alphanumeric terminal. Before exiting the task places itself in the task execution queue with a different entry point from its menu entry, sets a flag within the resident area, and reactivates background mode. When the user enters a point this is placed in a common block within the resident area, irrespective of whether it is to be processed locally or not. Background mode then tests the state of the above flag and if set activates the top task in the execution queue. This will be the requesting symbol task, which can find the coordinate entered and the button number from the resident area where it was placed by background mode. The process may be repeated any number of times to input the required number of coordinates. When enough data is gathered the symbol task can construct the required symbol, reverting finally to background mode when it has been stored and displayed on the graphics screen. The data that is entered in the workspace is the minimum necessary to fully define the symbol so that it may be redisplayed, plotted etc. at some later time. This data is formed into a 'symbol block', being a string of records of predefined format of which the first and last, carrying I codes of 3 and 4 respectively, define the upper and lower limits in space of the symbol, and the second is an identification record indicating the length of the symbol block and the number of the task responsible for creating it. These pieces of information are essential for other tasks within the system to correctly identify and process symbols and so must obey a fixed format. Other records within the symbol block may be unique to a particular symbol and will normally be skipped by other tasks, which can use the length of the block stored in the second record to jump to or past the last record as necessary.

It was considered essential that general graphics routines should not need to know the formats of each symbol block for correct processing of data, and to this end each symbol task must include the capability to perform a number of operations on its own symbol block where necessary. This condition ensures that the addition of further symbols to the system does not necessitate modifications to large numbers of tasks to handle the new symbol. Of the functions to be handled by the symbol task the most common, after the initial entry in the database, are redisplay on the graphics screen or output to the plotter. Each special function is identified by a unique entry point to the task, which must be the same for all symbol tasks. The number of functions to be performed on a symbol block is large but in general they are not related and it has been found easy and efficient on core usage to overlay the various functions within the symbol task in the way shown in figure 5.5. Each overlay is essentially a subroutine that is called by the root segment to perform the desired function.

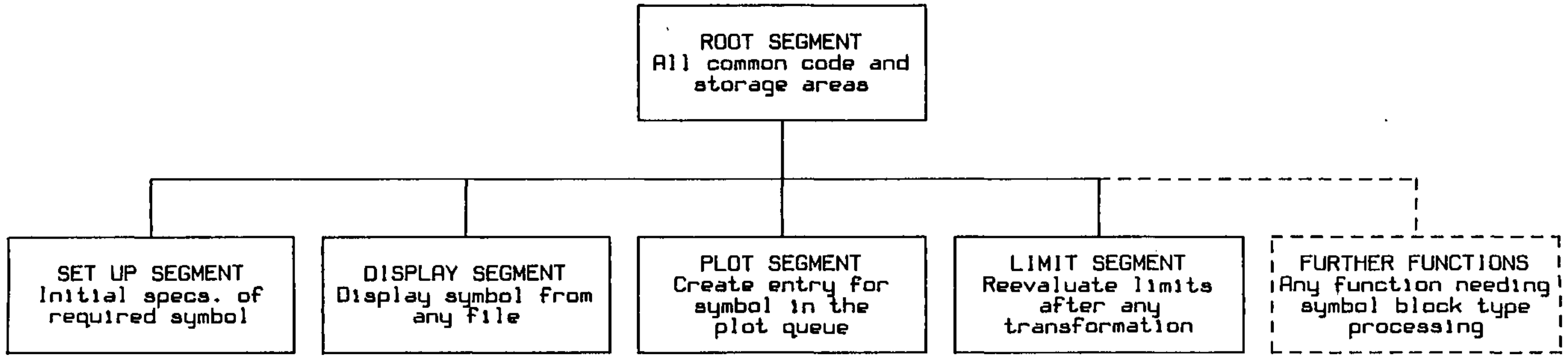

ROOT SEGMENT
All common code and storage areas
SET UP SEGMENT
Initial specs. of required symbol
DISPLAY SEGMENT
Display symbol from any file
PLOT SEGMENT
Create entry for symbol in the plot queue
LIMIT SEGMENT
Reevaluate limits after any transformation
FURTHER FUNCTIONS
Any function needing symbol block type processing

Fig 5.3 Symbol task overlay structure

In some ways this is a retrograde step from the latest single user system where graphics tasks, for example the redisplay routine, contained the logic to process all symbols locally, although the number of different symbols was rather less than is envisaged in the final multiuser system. Its predecessor functioned on a similar principal to that described above, but suffered, as does the current system, from slow operation due to the necessity to swap between tasks to process each symbol as it is encountered in the workspace. For the reasons outlined above it was found necessary to revert to the older principal, but a compromise has been reached in that graphics tasks may be built to process 'known' and 'unknown' symbols. A known symbol is handled within the body of a task simply by including the appropriate overlay segment from the symbol task within that task, whereas unknown symbols still cause a swap to the symbol task for processing. This allows the advantages of local processing to be combined with the advantages of symbol task processing and allows further symbols to be added to the system with no changes to graphics tasks, although for efficiency further symbols can be fairly readily made known to the relevant tasks at some later date with no change to the code of the symbol tasks and minimal change to the processing task. Perhaps the greatest improvement over the separate task operation has been achieved in plotting, since the plot data is written to a file which is later spooled to the plotter as described in section 5.9. This necessitated each task opening and closing the plot file on entry and exit and resulted in considerable processing times to generate a plot file of a drawing containing many symbols. Tests have shown that the local processing method reduces the time required by up to 75% on a typical drawing.

5.4 Macro Handling

In practice, engineering drawings normally contain features that are neither simple lines nor pure symbols, but a combination of these items which are generally referred to as macros. A typical example of this is the pump shown in figure 5.4 which is a combination of rectangles, straight lines and arcs. Furthermore these may appear frequently on one, or a number of different, drawings and it is convenient to enter the item once only and then repeat it at will elsewhere. The macro facility enables the user to do this, and is the heart of an efficient graphics system. The user may build up libraries of standard macros that are in common use and quickly and simply recall them onto the current drawing. The creation and use of a macro may be considered as two separate processes, as outlined below.

5.4.1 Creating a Macro

The standard method of creating and using macros is to enter those needed and store than in the macro library before commencing the drawing. At the initial implementation of the graphics system this is obviously a large task since all macros that are needed will have to be entered, but very soon a comprehensive library of macros will build up and the user will frequently find that all but a few of the macros required for a particular drawing are already in a library.

When creating macros in this manner a special menu square places the system in 'macro-creation' mode. This command initially checks to ensure that the workspace is empty, since all data in the workspace will be overwritten by the macro, and if not the user is given the option of continuing or aborting the process. If all is well a number of macro parameters and pointers are stored and the system reverts to background mode to allow the user to digitise the new macro. The full range of graphics facilities are available for this process and the macro may contain any combination of graphical items, including other

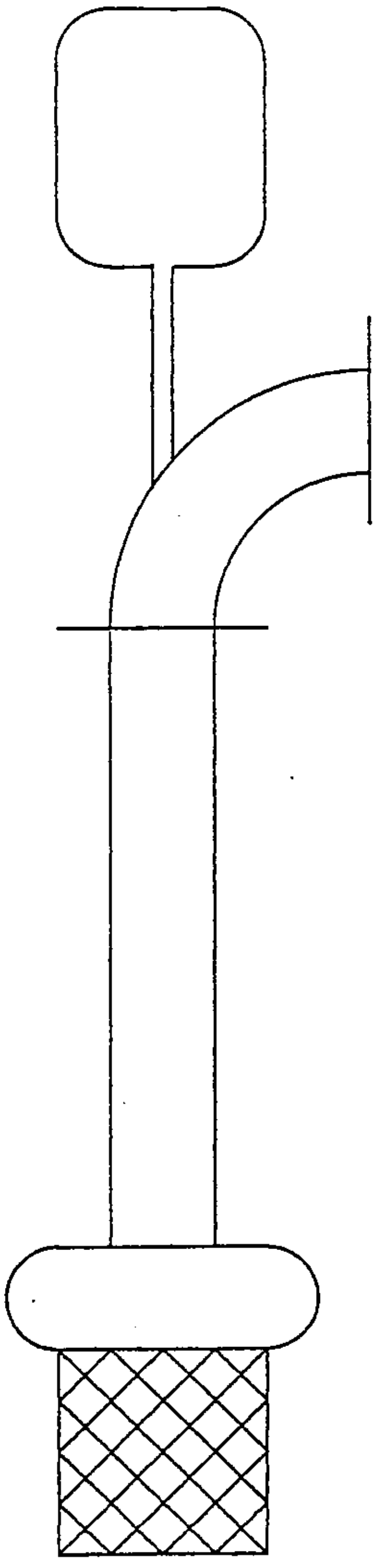


Figure 5.4 A typical macro

existing macros. When digitising is completed the user indicates this to the system by issuing a second menu command which initiates the macro saving procedure. This requests the user to input three more points, which are passed to the macro task by the method outlined in the previous section. These three points are known as 'positional reference points' and are used when the macro is recalled onto a drawing to rotate and scale it to the correct position and size. The user should endeavour to choose 3 points on the macro that are of particular significance, such as points at which connections are made to other items. For example, if the macro is a valve symbol it would be convenient to choose the centre of the flanges as two of the reference points which enable it to be conveniently placed in a pipe run when recalled. Three reference points are required, as opposed to the more familiar two used with existing 2 dimensional systems, since the macro must be placed in 3 dimensional space, for which 3 points are needed to define a unique plane. In addition to the reference points the user must supply a code name by which the macro is to be known. This may be any alphanumeric string of up to 6 characters, which serves as an identifier allowing the macro to be recalled by this name, and will ultimately have significance when a full material take-off facility is incorporated. The macro library is logically divided into 'pages' of 50 macros. This is necessary since a square on the macro area of the digitiser is allocated to each macro, and in a large system it is necessary to incorporate a pageing system of the type used for the menu to allow access to every macro. This feature is discussed more fully in the following section. When creating a macro the user must decide on which page and in which square the macro is to reside, and enter the two numbers as part of the macro termination procedure. This square must be currently unoccupied, and if not the user must physically delete the existing macro before the new macro is stored. This procedure ensures some measure of safety against accidentally overwritting an existing macro, although since it was found inconvenient to bar users from deleting macros altogether the procedure entrusts the user not to delete any but the correct macros.

Each macro is stored in a separate file on the pseudo device GD:, where the file name is derived from the 6 character identification code specified by the user. In addition an index file is maintained which records all the macros currently available on the system and a number of pieces of information about them, such as when they were stored, last accessed etc. The macro file is essentially a direct copy of the workspace, plus the reference points specified during the saving procedure. In the current implementation the index file serves very little useful purpose other than relating macros to their corresponding menu square, but has been included both for compatibility with the drawing saving procedure outlined in section 5.10 and for future expansion, particularly in the field of material take off, where it is anticipated that a number of parameters may have to be stored for each macro, which will most conveniently be located in the index file.

However carefully the drawing is preplanned the user will occasionally discover that an essential macro is not available when needed, or that it is convenient to make a part of the drawing into a macro for later use. To cater for this situation the macro storing routines incorporate a retrospective creation procedure based on selected data, as described in section 5.6. At any time during drawing creation the user may select an area of the drawing and convert it into a macro, the saving procedure being similar to that previously described, except that it is initiated with a different menu command, and requires data to have been previously selected. This procedure does not affect the drawing as it exists in the workspace and can be safely used at any time, and as often as required, during the input of a drawing.

5.4.2 Recalling a Macro

The storage of a macro is, largely through necessity, a lengthy and somewhat tedious process, but may be excused since this a once and for all operation that is carried out only during the initial definition of a macro. In contrast, the recalling process may be carried out many hundreds or thousands of times and it was considered essential that this should be kept as simple as possible.

In order to identify which macro is required the user has two options. Figure 4.6 shows the layout of the digitiser, and the position of the two macro menus along the left hand edge. Each of these represents one of the macro pages previously described, two areas being available in order to give a greater freedom as to which macros are currently available. A common situation would be to have one of the pages allocated to standard company macros for a particular type of drawing, and the other allocated to the uses specific macros for the current drawing. For a macro to be available from the menu the page to which is has been allocated must occupy one of these two positions, page changes being accomplished by menu command. The first, and most common, method of recalling a macro is simply to digitise over the square on the menu to which it has been assigned. The system uses the square and page numbers as pointers in the macro index file to obtain the macro code word, and hence the name of the file in which it is stored. The process is simple for the user to appreciate and has the added advantage that the code word need not be known, the illustration drawn within the box being the means by which the correct macro is chosen. Furthermore, if the macro is required repeatedly, button A may be assigned to that square and so enable it to be recalled without further reference to the menu.

An alternative method of recall is to request the macro by its code name. The process is initiated by a standard menu command which requests the user to enter the code on the alphanumeric terminal. After an initial check in the directory to ensure that the codeword is valid, access can be made directly by the system to the correct macro storage file. From this point on the process is identical irrespective of the method used to initally identify the macro.

In order to specify the location and size of the macro when it is added to the current drawing the user will be expected to enter one or more points from the digitiser, in a similar way to those required for the positioning of a symbol. In the default mode three points will be requested that specify the position of the reference points stored within the macro file. The macro is scaled and rotated so that the first two reference points coincide directly with the two points entered by the user, and the third point lies in the correct plane to the correct side of the line between the first two points. It is fairly obvious that all three points cannot necessarily be directly matched up, unless the triangles defined by the users points and the stored reference points are similar, which would normally be no more than a happy coincidence. The process of aligning the first two points and using the 3rd for space orientation will always function correctly no matter what points are entered unless they are colinear, in which case the user is asked to re-enter them.

When using the full procedure above the macro is automatically scaled and rotated before being added to the workspace. In many cases, particularly where the same macro appears repeatedly on a drawing, the scale factor with respect to the stored macro, or the orientation, are always the same, and it is convenient to previously define these parameters and hence save time and effort when the macro is to be recalled. For example, if the orientation is known, only two points need be entered to specify the scale of the macro, or if both orientation and scale are known the macro may be placed with a single point. The process is convenient but, particularly in the case of default scale factor, requires the user to have a fuller knowledge of the size at which the macros were originally entered. The necessary parameters may be entered or changed at any time during the drawing process, and the use of either default being enabled or disabled at will and independent from one another. Depending on the current defaults the user will be prompted for the required number of points when recalling all future macros.

When the macro is added to the workspace a number of the parameters associated with it are modified. For example, the overlay on which it was digitised is no longer relevant, and is indeed not stored in the macro file, the macro always being placed on the current input overlay. Certain parameters, such as the display control, are preserved, but the current display control is always left set to the same value after the macro as it was before. The handling of line type and pen number is more complex, and merits further discussion. The line type specified in a macro is normally preserved when it is recalled, but at creation time the user may specify a macro default line type for which the current line type is always substituted on recall. This feature is important and allows a macro that has been defined in, for example, solid lines, to be reproduced in dotted lines if the system is in dotted line mode when recalled, but preserve features of the macro that must be of a fixed line type. A similar procedure applies for pen number.

The macro is stored in the drawing in a macro block similar to the symbol blocks discussed in the previous section, where the first and last records indicate the limits of the macro, and the second record contains identification information. Since the macro may contain other macros or symbols the software must scan the entire macro after it is recalled and revaluate the limits on these 'nested' items in order that normal graphics functions will handle them correctly. In general a macro is considered a single entity and may only be processed in its entirety.

To complete the macro facility a number of 'maintenance' type functions are available that allow the user to perform operations such as deleting old macros, or producing a directory of the available macros on the alphanumeric terminal or line printer, each of these functions being made available by self explanatory menu commands.

5.5 Editors

A comprehensive editing facility is essential to an effective graphics system, enabling the user to both correct errors as they occur and to make modifications to drawings as design specifications are changed.

The TIGER system incorporates a number of editors, each of which is designed to handle specific data types. The previous sections of this chapter have outlined three distinct classes of data, simple lines, symbols and macros. A separate editor is included for each of these types. Although an all encompassing editing facility is quite possible it has been found convenient to segregate the separate functions, since each editor can perform its function much faster and more efficiently by ignoring all but a particular data type. These editors are designed to behave in a similar manner to each other, in order that the user may learn one sequence of operations that is applicable to each mode, once having chosen the correct editor from the menu. The three editors each have two different modes of operations, last entry and random search mode.

Last entry mode is specifically designed to be used as part of the drafting operation to correct errors as they occur. The user will normally be aware immediately if an error has occurred, perhaps the wrong choice of a macro, and this mode of operation provides a quick and efficient method of removing the error without the user needing to identify specifically the problem area. The editor simply searches the drawing backwards from the end of information until the correct data type is encountered, which is then removed. The operation is very fast if requested immediately because only a small part of the database need be searched, but is obviously unsuitable for deleting data from anywhere but the end of the drawing.

Random search mode enables the appropriate data type to be located from anywhere within the database and be deleted. The user is requested to digitise a point which lies close to the data of interest as it is projected in any of the views that are currently available. The editor uses this search pointer and locates the data that appears closest to that point, offering it up for deletion. On completion of the search the located data is displayed on the graphics screen for verification, and the user asked to press a button to indicate whether this is the correct data and that it is to be deleted, whether it is incorrect and the search procedure is to be repeated, or simply to verify again. Random search modes loop internally until specifically exited by the user issuing the appropriate button command.

The above procedures may not be appropriate to some editing processes, particularly where a change of specification has rendered a large part of a drawing obsolete. It is obviously convenient to be able to delete this section on mass without considering how it is composed. A general editing function is available, based on selected data as described in section 5.6. The editor simply deletes from the drawing any data that is currently marked as selected, irrespective of its type and composition. Not only is this a very powerful editing facility, but it is extremely quick due to the way in which data is removed from the drawing when it is deleted.

In earlier graphics systems the editors physically removed the deleted data from the workspace, which normally resulted in the complete database being rewritten, a lengthy and time consuming process if a big drawing is being created. The current editors do not remove the data, but simply mark it as deleted for removal at a later time, which will normally be done automatically by any other function that has need to rewrite the database, with no extra time penalty. To achieve this the data is transferred to drawing overlay zero, which is not normally available to the user and is always considered passive by the system. The overlay, being an indexed parameter, may be changed without reference to the main drawing file, since it is kept in a separate bulk core file, for which access times are almost negligable. This method has the added advantage that the editing process is not irreversible and under some circumstances deleted data may be recovered before it is lost entirely.

The above editors are designed principally to remove data from a drawing. A further set of editors exist to modify certain parameters associated with the data and do not result in any deletion. As has previously been stated, the data is stored with a number of additional associated parameters, principally the drawing overlay, pen number and line type. The values that are associated with a particular data type are those that are currently in force when the item is originally added to the database, and it is quite possible to accidentally enter an item with the wrong associated value. An obvious solution is to delete the item and redraw it, but under some circumstances, particularly where the item is complex and the error is not realised quickly, it is better to modify the incorrect parameter in the existing data. The facility exists to modify any of the above 3 parameters independantly, and is based on the user selecting data with the data selector described in section 5.6. As a typical example, changing the pen with which the drawing is to be produced, the user has the option of changing every pen number to one specific pen, or selectively changing one pen number to another, all within data that has previously been selected. This form of editing is slower than the deletion process described above since it involves rewriting sections of the database, and furthermore is irreversible, except by a complementary edit, since the original contents of the modified parameter are lost.

One further editor exists, enabling any values within the database to be modified, using a fairly conventional interactive type process from the alphanumeric terminal. The use of this facility involves in depth understanding of the data structures used, and is therefore limited to programmers use, since by its very nature the user is not intended to know, or need to know, such details. The facility is useful for examining suspect areas, or patching problems that have arisen, but generally is rather slow in operation, requiring a command to change every parameter of every record, and is therefore of limited use. Furthermore, in the wrong hands it is very easy to corrupt the entire database beyond recovery.

The editing process will normally result in a change to the drawing, which can only be seen by the user when the drawing is redisplayed on the graphics screen. It has been found from experience that it is not always necessary for the drawing to be redisplayed immediately, since the user should be aware of the changes that have to be made. Furthermore, the redisplay process is very time consuming on a large drawing and progress can be very slow if this is performed frequently. To overcome this the system supports either automatic or manual redisplay modes, which may be selected by menu command. In automatic mode, which is active by default, the drawing is always redisplayed when a change is made. In manual mode no redisplay every occurs unless the user physically requests one, either explicitly or by implication, such as when changing the display window. The user is free to swap between modes at any time during the drawing process.

5.6 The Data Selector

All early development systems at Imperial College suffered from the inability to apply any graphical functions to anything but the whole database, or at best a predefined set of overlays that were currently marked as active. This meant that drawings needed very considerable forethought in order to split them into workable sections before they were input, and any unexpected changes could lead to difficulties. It was soon realised that in the commercial environment this would prove a disasterous burden and severely limit one of the systems most useful capabilities, that to apply modifications very efficiently to existing drawings. To overcome this the data selector described herein was developed.

The data selector allows the user to identify specific parts of a drawing that are to be made available for consideration, irrespective of any other designations such as overlay, line type etc. In this way the user has complete flexibility over which parts of a drawing are affected by any of the graphical functions, since almost without exception these will operate only on data that has been previously selected. This is very fast since the select status of the drawing is the second of the indexed parameters described in section 4.5, and hence the time taken for any operation is roughly proportional to the amount of data that is selected and not to the overall size of the drawing. Graphical functions relying on selected data will normally expect the user to have previously selected any data of interest and will exit with an error message if this is not the case.

Data that has been selected remains so until it is physically deselected by the user or by one of the graphical functions. A convention has been adopted whereby any function that operates on selected data will, by default, automatically deselect it after the operation is complete. This is important due to the cumulative operation of the selector, and if not the case could lead to the user applying subsequent functions on data which is no longer of interest.

For the experienced user this default may be overridden, whereby any selected data will remain so until specifically deselected from the menu. This can be very important where more than one function is necessary on the same data. For example, if a section of the drawing is to be duplicated at a different angle this is best performed as two operations, duplication at the same angle as the original followed by a rotation to required angle. It is obviously advantageous for the select status to be retained for the second operation, which would not be the case in the default mode of operation.

For maximum flexibility the selector provides a number of different methods for the user to identify the required data, and functions in either select or de-select mode. The latter has been found by experience to be important since it enables the selection of awkward shaped or overlapping sections, allowing initial selection of a large area, then deselection of smaller sections within it. In addition the operation of the data selector is cumulative, as previously mentioned, allowing a number of different operations to be performed to build up the required selected data if this is not possible using one method alone. The methods currently provided for identifying data are as follows. Each description assumes that select mode, as opposed to de-select mode, is currently in operation.

a) Select Within a Digitised Boundary

This option allows the user most flexibility and is by far the most powerful method for selecting data at random. The user simply digitises a boundary composed of up to 30 straight lines to surround the data of interest in any of the views that are currently defined. The selector then checks which data appears to fall within this boundary and marks it as selected. A certain amount of care must be exercised by the user when defining a boundary that intersects data items. The operation is fairly predictable when the boundary intersects simple lines, but may be less obvious when considering macros and symbols. In both cases these items form a complete entity and so must be either selected in their entirety or not selected at all. The current implementation of the data selector will select the entire macro or symbol if it lies principally within the boundary and vice-versa. If in doubt the user can subsequently use the 'displayed selected data' command to verify the action that has been performed.

b) Select Specific Overlays

If effective use if made of the overlaying capability it will often be found that the information of interest is located on one or more known overlays. This option allows the user to specify that certain overlays are to be selected. The overlays of interest are specified numerically via the alphanumeric terminal. Overlays are selected irrespective of whether they are currently marked as active or passive, although in the latter case a warning message is issued to the user, since it is not normal for passive overlays to be affected by graphics functions. This is particularly important since select status takes precedence over overlay, either active or passive, and routines that manipulate selected data do not check on the overlay status, and so could produce unseen effects on passive overlays which will not be displayed on the graphics screen and hence may go unnoticed. This function, in conjunction with the general editor, is particularly useful for deleting whole overlays from a drawing.

c) Select All Active Overlays

A user is normally only aware of active overlays, so in the majority of cases this will appear as if it comprises the whole drawing, when in fact passive overlays may exist that are not affected. This option should normally be employed when it is necessary to select the whole drawing in order to avoid affecting passive overlays, and so producing the aforementioned spurious effects.

d) Select All Passive Overlays

This option is included for completeness and is complementary to the above option. In general it is very dangerous and should be used with great care, normally only for editing purposes.

e) Select Entire Drawing

This operation will select the entire drawing irrespective of any other designations and should be used with great care since passive overlays will be selected and may be affected by future graphics functions. This function is included for completeness and has limited use. A typical example may be to move the entire drawing to fit a different size or shape of paper from that for which it was originally intended.

f) Select Particular Line, Symbol or Macro

This section comprises three separate options which may be requested individually by the user but which are grouped together in this explanation since their functions are alike. Their mode of operation is similar to the random search editor functions described in section 5.5, expecting the user to digitise near any projection of the items of interest. Unlike the editors the user may enter up to 50 different search points before any processing is attempted, using button F to terminate point input in the standard manner. A search is then made for the closest item, either line, symbol or macro depending on the mode, to each digitised points and that item marked as selected. The operation is very fast since only one pass is made through the database, and in conjunction with the general editor may provide a faster method of deleting a number of items from the drawing than repeated use of the standard editors.

g) Select Last Line, Symbol or Macro

As under item f) this section describes three separate options since their functions are similar. Frequently the last item entered will be the one of interest for further processing, and these options allow the user a quick and simple method of selecting that item. For example, if the same macro appears a number of times on a drawing it is faster to recall the macro once, select it as the last item, then duplicate it in the other positions, since this requires less input from the user and less processing than recalling the macro separately for each new location.

5.7 Data Transformations

Discussion so far has centred around the problems of entering and deleting data from a drawing. Another class of operation altogether is data transformation, the modification of existing drawing data without entering or deleting any information. A typical example of this process would be the moving of one item of equipment to a different location. The data describing that item is modified in the workspace to reflect the new location.

All data transformations are applied to selected data only, enabling the user to discriminate between those items that are of interest and the remainder of the drawing that is to remain unaffected. Data transformation routines always check to ensure that data has been preselected before attempting their operation to avoid needless computation, and issue an error message to the user if no selected data is found. These operations can normally be carried out very quickly by jumping through the drawing picking out items that have been selected without considering the remainder of the drawing.

The TIGER system incorporates a large number of transformations, and no attempt will be made to describe each in detail. However, almost without exception every transformation is based on either a rotation, scaling, shifting, or a combination of the three. An understanding of the mechanism behind each of these transformations is adequate to enable the principles of more complex transformations to be deduced.

5.7.1 Rotation

The principles of rotation on a 2 dimensional surface are simple to deduce because there is effectively only one axis about which this rotation can take place, that is the axis perpendicular to the surface. From simple geometry it can be shown that, in matrix notation:-

$$\begin{bmatrix} \mathrm{Cos}\ (\theta) & \mathrm{Sin}\ (\theta) \\ -\mathrm{Sin}\ (\theta) & \mathrm{Cos}\ (\theta) \end{bmatrix} \times \begin{bmatrix} X_1 \\ Y_1 \end{bmatrix} = \begin{bmatrix} X_2 \\ Y_2 \end{bmatrix}$$

or alternatively $\quad TX_R = X_T$

Where θ = angle of rotation

X_R or (X_1,Y_1) = the original point
X_T or (X_2,Y_2) = the new location of the point
T = transformation matrix

Furthermore it can be shown that compound transformations may be condensed into one transformation by multiplying together the transformation matrices, and the order in which this is done is not significant.

In a similar way rotations in 3-D space may be described in matrix notation, although with inherent complications since there are now three axes about which rotation can take place. It is fairly evident that a rotation cannot simply be defined by specifying a rotation about individual axes, since the order in which those rotations are applied affect the final result. It is therefore necessary to define a convention for the order and direction of rotation before any three dimensional transformation can be considered. The system adopted is the conventional right handed set of othogonal axes as shown in figure 5.5. If the X-Y plane is considered to lie flat in front of the observer with the Y axis vertical, the Z axis is always positive in the direction towards the observer. Angles of rotation are in the 'right handed corkscrew' convention, i.e. when looking along any axis in a positive direction from the origin clockwise rotations are defined by positive angles. In order to avoid confusions, compound rotations are always applied about the X, Y and Z axes in that order, each of which may be defined separately by the simple two dimensional transformation described above.

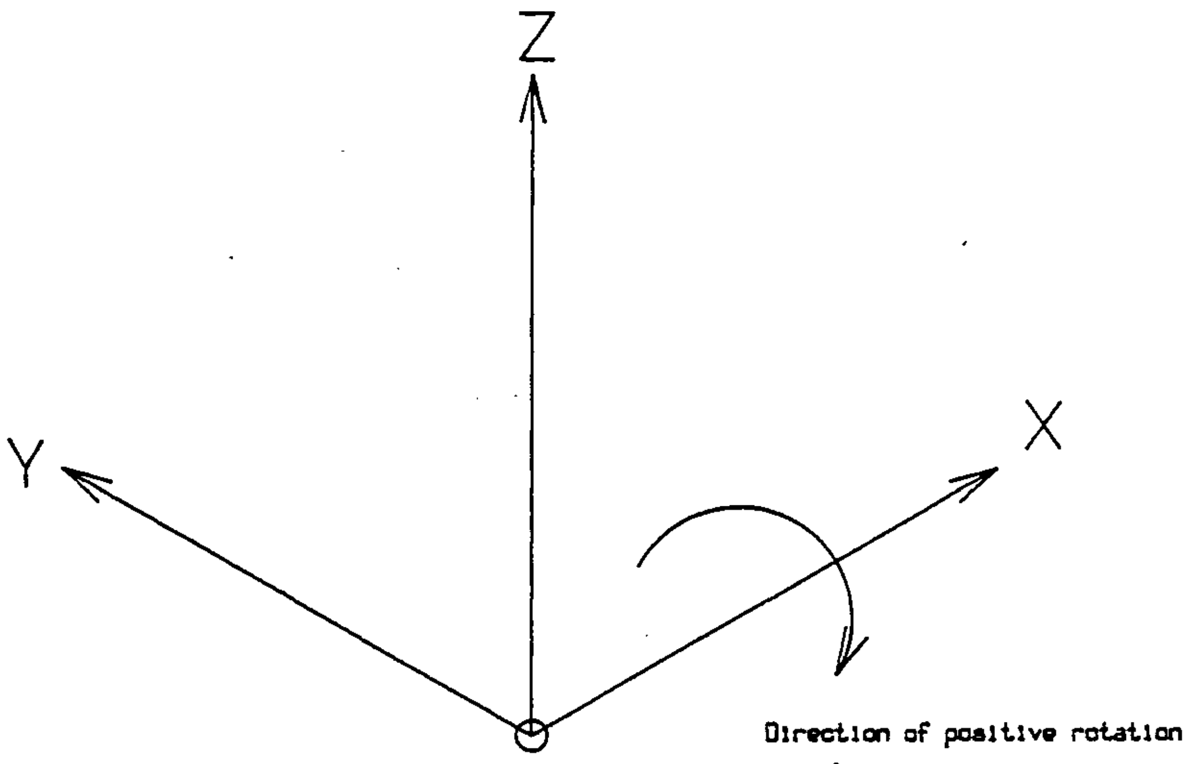


Figure 5.5 Conventional right handed orthogonal axis representation

When applying any transformation to the drawing data a number of problems may be encountered. Little problem exists with simple straight line data that has been input from the digitiser, but a number of problems may be encountered with symbols and macros. As described in Chapter 4, the first and last record of both of these items contain their low and high limits in space, i.e., they define the bounding cuboid. It is fairly evident that if the object is rotated it is not adequate to rotate these limits, since they will not result in the corners of the new bounding cuboid. It is therefore necessary to reevaluate the limits based on the new orientation of this object. This function was described under section 5.3, and may be achieved either by calling the appropriate symbol task, or by including the symbol tasks re-evaluation overlay within the body of the transformation routine. The latter process is normally adopted due to its greater speed and efficiency. Furthermore a symbol block may contain both positional data and numeric data. For example, a block defining a polygon contains co-ordinates defining its position, and a numeric value defining the number of sides. It is important that the transformation is applied only to the positional information, and not to any other types. This is achieved by allocating standard I codes that are recognised by all graphics routines. For example, a positional record must bear an I code of 1 or 2, and a data record one of 5. In general, transformation routines ignore codes of greater than 3 and so do not corrupt data records. Furthermore, in order that symbol blocks can be manipulated by routines that do not know their exact structure they must define both their location and orientation by means of 3-D co-ordinates alone, so that when a rotation is performed any angles that may apply to that symbol block will be correctly updated.

The TIGER system supplies a number of standard Fortran callable subroutines to perform rotations in 3 dimensions, so freeing the casual programmer to a large extent from the need to worry about the conventions outlined above.

5.7.2 Translation

Translation is the simplest transformation and carries with it very few of the problems outlined under the above section. Since the orientation of data is not changed the limits on symbols and macros may be updated in exactly the same way as normal positional records, i.e. by adding the appropriate shifts on each axis. There is therefore no need to include the logic to reevaluate them completely or request the symbol task to do so. Similar arguments apply to the records within a symbol block and the I codes associated with them.

5.7.3 Scaling

The last of the three basic transformations is scaling, which, although slightly more complex than translation, is still simple in concept. Scaling, or the changing of an objects size, can be defined by a scale factor and a reference point about which the object is to be scaled. The need for this point may not be immediately apparent, but is necessary if the location of the object is to be determined correctly after the transformation has been applied. If, for example, the coordinates of an object are multiplied by 2, the scale factor, the object will not only finish up twice its original size, but also twice the distance from the origin of co-ordinates. The final location of the object can be controlled exactly by specifying a reference point which will lie at the same position both before and after the transformation. This point is sometimes referred to as a relative origin. Providing the conventions outlined in section 5.7.1 are obeyed the operation can normally be carried out correctly without recourse to the symbol task itself since the limits will remain valid if scaled, and correctly assigned I codes should allow the scaling routine to apply the correct changes to the data records.

5.8 Two Dimensional Operations

Although a fully three dimensional capability was the principal aim of the project it is recognised that a large number of drawings produced by most companies are of a purely two dimensional nature, such as flow diagrams, and even fully 3-D drawings contain large quantities of essentially two dimensional information in the form of annotation. A number of special facilities exist to provide the necessary functions.

5.8.1 2-D Mode

In effect, there is no reason why a two dimensional drawing cannot be produced using the full three dimensional system with only one view enabled and the third coordinate being ignored. In practice, this can lead to confusion and burden the user with unnecessary complications and requests to perform redundant operations. For example, to input a polygon the user must digitise the centre and one vertex point. In two dimensions this is adequate information to fully define the location, but in three dimensions a third point is necessary to obtain an orientation in space, since two points will only define an infinite family of planes containing their interconnecting line. To overcome this a pseudo mode, known as 2-D mode, may be selected at the users discretion. This is not a true mode since the full 3-D database is still used, the complications involved in using a condensed form of the database being considered prohibitive. In effect, implementing 2-D mode merely modifies the views defined, to cover the entire digitiser with one X-Y view, and sets a flag within the resident area to indicate that 2-D mode is in operation. This flag is tested by a large number of the tasks, and slightly different messages output or operations performed depending on its setting. Additionally, some functions, such as the use of button 3 in background mode (to set the trailing coordinate), are prohibited. With 2-D mode in operation the user is completely unaware of the 3-D capability of the system.

5.8.2 Annotation

As indicated above annotation is a two dimensioned function common to both three and two dimensional drawings. Most annotation consists of special symbols such as arrows, dimensions and text, but may in general consist of most graphical items, these being distinguished from normal drawing data by the setting of the display control flag described in section 5.2.3. The system currently provides a number of special annotation methods as outlined below.

a) Arrows and Pointers

Arrow heads and pointers figure prominently on most engineering drawings and a quick and simple method of entering them is essential. At first glance it may appear that this can be achieved using the macro facility, but this is impractical since they must vary in shape, having differently proportioned heads, different length tails, etc. This is therefore achieved by treating arrows as symbols and entering them according to preset parameters and points digitised by the user. Figure 5.6 shows the parameters that may be varied and a selection of typical arrows that result.

b) Dimensions

Many types of engineering drawings carry dimensions and it is essential that they can be added without the need to draw them from their constituent parts each time. Due to their variations in shape and form they are constructed in a similar manner to the above arrows from a number of predefined parameters and user digitised points. On orthogonal views dimensions are constructed between two digitised points with the double headed arrow running through a third digitised point, either vertically, horizontally, or parallel to the line between the first two points. In isometric projections the dimensions run parallel to one of the orthogonal axes, and appear to be in a major plane specified by the user. By using the find facility the

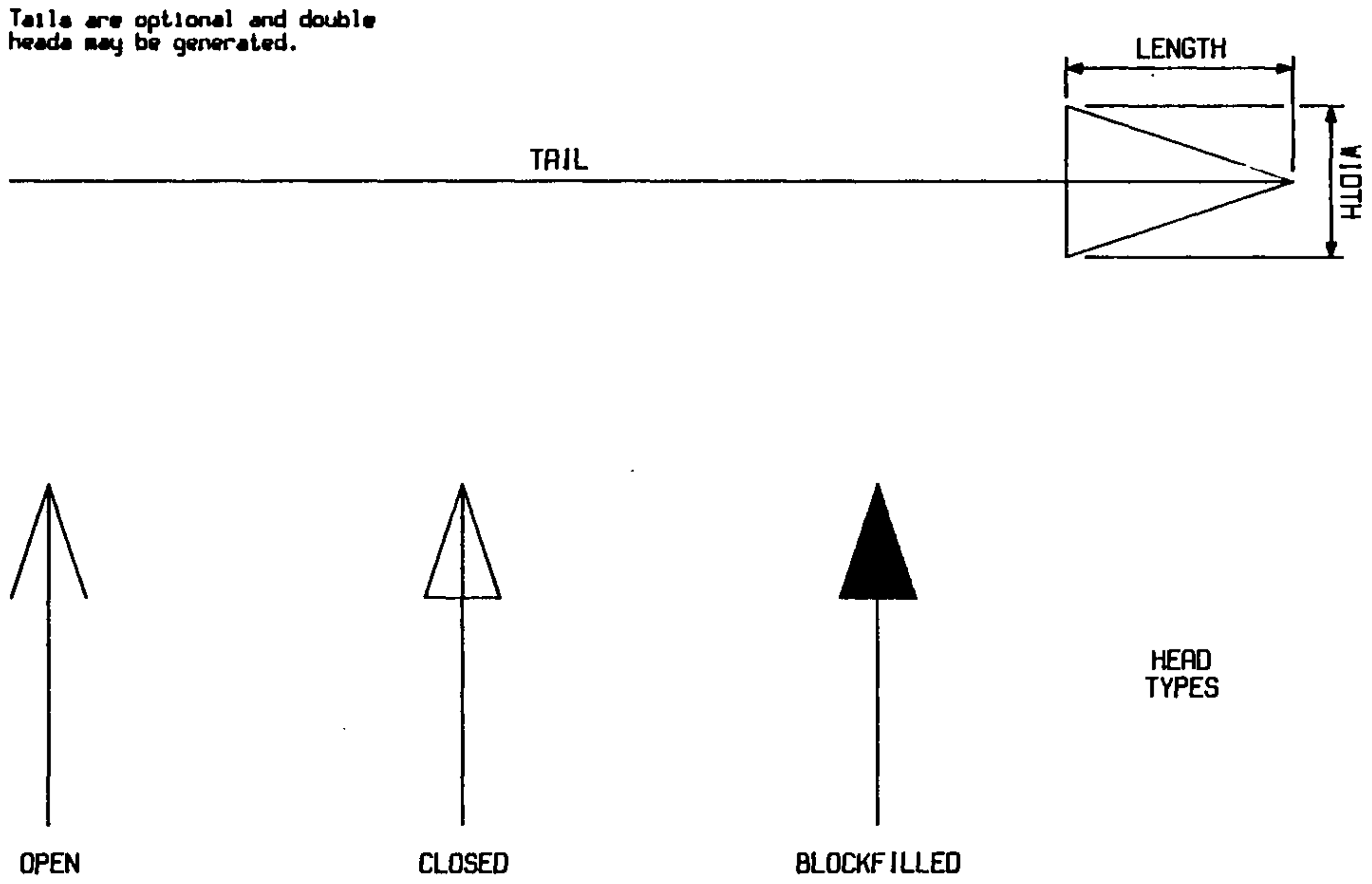


Figure 5.6 Arrow formats

first two points may be made coincident with any existing points on the drawing to give accurate dimensions. The user may optionally allow the system to compute the correct dimension from the co-ordinates, or enter the value to be used manually, which is important if the drawing is not to an accurate scale, which is often the case. Figure 5.7 shows some typical examples of dimensioning.

c) Text

Text is probably the most common type of annotation and appears in some form on every drawing produced, but differs from other forms of annotation in that it has no graphical content. A number of different parameters exist to define the format of text, namely the character height and aspect ratio, character set or font, and the direction in which it is written. The conventional method of entering text was to define the above parameters manually, which were stored in the common area and remained until changed again, then request either right, left or centre justified text from the menu. This prompted the user to digitise a reference point to indicate the required position on the drawing, then type the text on the alphanumeric terminal. The above process suffers from a number of drawbacks, and a more efficient method was sought. Firstly, the user is obliged to turn repeatedly from the digitiser to the terminal and back when entering multiple blocks of text, and secondly the speed of operation is entirely dependant on the draftsman's skill on the typewriter keyboard. Obviously it is unreasonable to expect all users to be proficient typists, and the time required to enter large quantities of text can be quite considerable. An obvious solution to both these problems is to have all the necessary text and commands in a file that the user can call up from the graphics workstation. It is then merely necessary to digitise the location of each text block as it is offered up. Furthermore, this file may be entered from any standard computer terminal and by anybody, so the process does not monopolise a workstation, and can easily be done by someone of secretarial grade who is a proficient typist. To this end the following specification was devised.

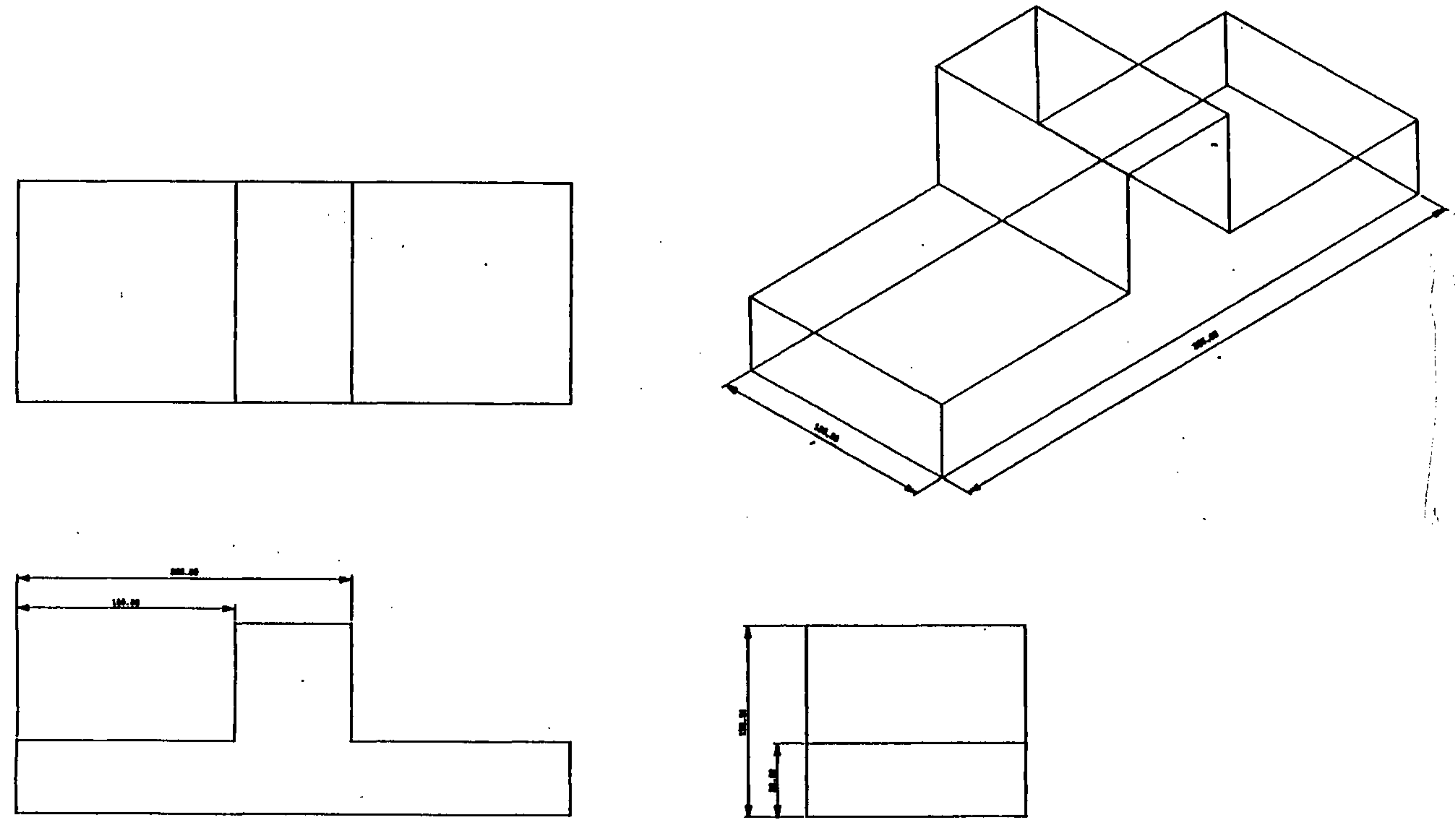


Figure 5.7 An example of automatic dimensioning

The instruction file consists of blocks of text interspersed with commands to set the various parameters, text blocks being logically seperated by one or more commands. Any line that starts with an '*' character is assumed to be a command, any other line part of a text block (except lines starting '**' which are taken as a text block starting with a single '*' character). The character following the '*' indicates the exact command, and any characters following the next space are taken as a qualifier, either a text string or numeric values depending on the command. Any characters between the command identifying character and the space are ignored. For example the command:

```
*HEIGHT 3.5     or     *H 3.5
```

resets the default character height to 3.5mm. In this implementation the following commands are recognised, the examples showing typical values where applicable:

*A 1.2	set aspect ratio
*C	switch to centre justification
*D 90.0	set text angle, degrees
*F	set character font
*H 3.5	set text height, millimetres
*L	switch to left justification
*P 2	change to specified pen
*R	switch to right justification

Although text blocks may be logically separated by any of the above commands it has been made a convention that either C, L or R is used since this bears greatest similarity to the manual method and is least likely to cause confusion. In addition, these commands may be suffixed with a repeat count following a / character which is applied to the next text block encountered. Furthermore, any '#' character encountered in the text block is replaced with the current value of the repeat index. Legal forms for repeat commands are:-

*C/5	repeat next block 5 times
*C/2,9	repeat 8 times with index values from 2 to 9 inclusive
*C/3,15,2	repeat with index values from 3 to 15 increasing by 2 each block
*C/B,Q	repeat block and substitute characters B, C, D up to Q in successive blocks

In character string substitution each '#' is replaced by the character. For numeric substitution a single '#' is substituted for the decimal value of the number and the string expanded accordingly. A block of consecutive '#'s is substituted for one number with leading spaces if necessary and leftmost digits omitted if the space is inadequate. The following examples illustrate this process.

```
Command file                 Resultant text blocks

*C/3
MC#                          MC1
MM###                        MM  1

                             MC2
                             MM  2

                             MC3
                             MM  3


*C/9,11
TYPE # VALVE                 TYPE 9 VALVE

                             TYPE 10 VALVE

                             TYPE 11 VALVE
```

```
*C/A,C
SIZE # PAPER                    SIZE A PAPER

                                SIZE B PAPER

                                SIZE C PAPER
```

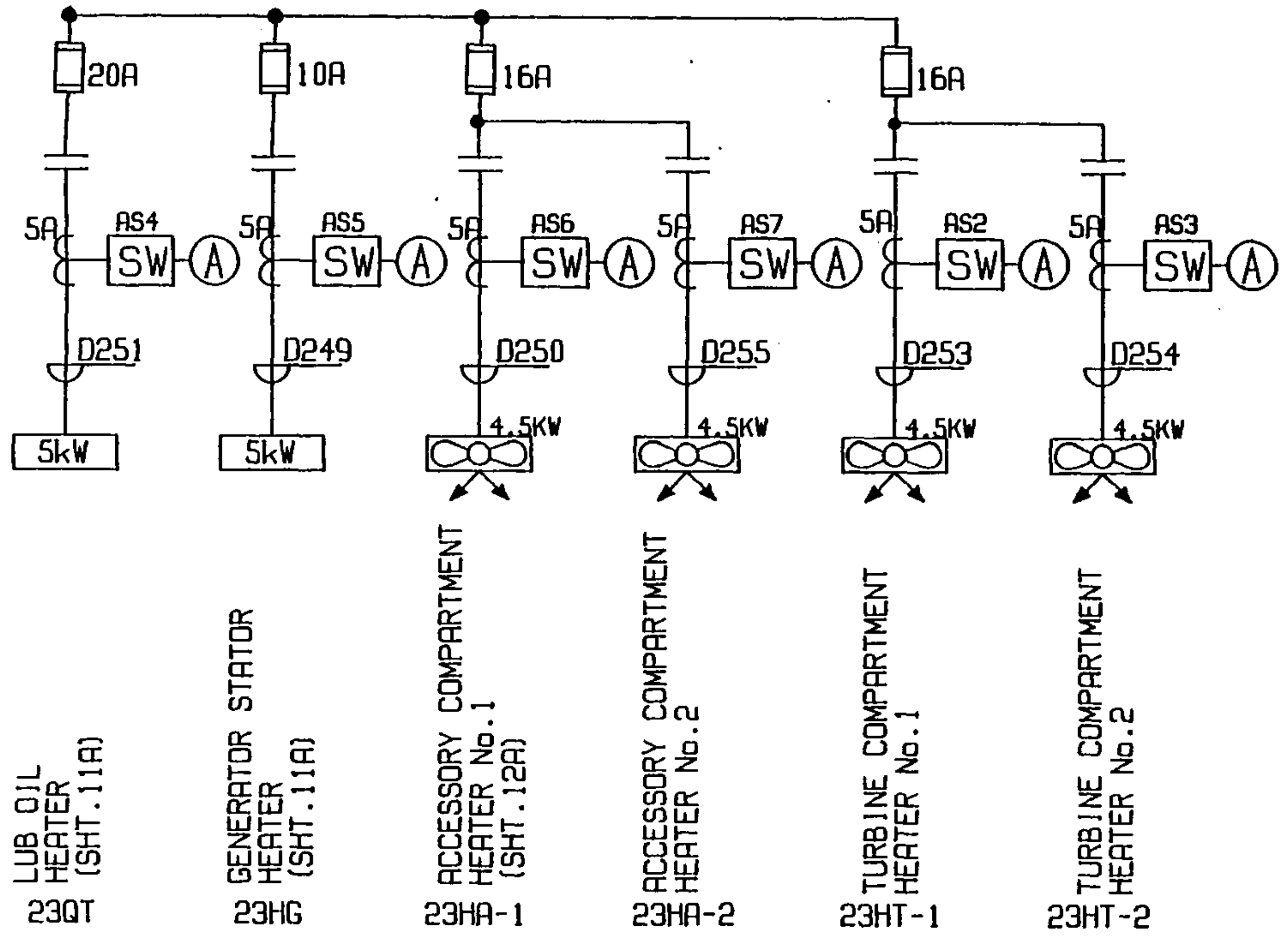
Figure 5.8 illustrates a simple drawing and an example of the command file used to insert the text.

When the user calls for a text file from the workstation the system first processes this file into a direct access scratch file. While reading the file this preprocessor recognises a line starting with an '@' character as a request to start reading another file, the name of which follows the '@' symbol. Output continues from this file until logical end of file is detected, when it resumes after the '@' line in the previous file. It is the scratch file that is used and modified by the graphics system during the entry of text from the digitiser. Throughout this process the buttons on the stylus obey their normal background mode assignments, except button 2 which skips the current block on offer and does not enter it on the drawing, and button C which terminates the process entirely in an orderly manner. After the last block the system checks whether any blocks have been skipped and optionally returns to them.

The format in which the text is stored in the database is identical to the blocks created by the manual method, which is still available if required, and so may be edited, shifted, etc., in exactly the same way.

5.8.3 Transformations Involving Annotation

On a fully three dimensional drawing where true three dimensional information and annotation exist together a certain amount of care has to be exercised to ensure that both are handled logically when certain


20A
10A
16A
16A
5A
AS4
SW
A
5A
AS5
SW
A
5A
AS6
SW
A
5A
AS7
SW
A
5A
AS2
SW
A
5A
AS3
SW
A
D251
D249
D250
D255
D253
D254
5kW
5kW
4.5KW
4.5KW
4.5KW
4.5KW
LUB OIL HEATER (SHT.11A)
GENERATOR STATOR HEATER (SHT.11A)
ACCESSORY COMPARTMENT HEATER No.1 (SHT.12A)
ACCESSORY COMPARTMENT HEATER No.2
TURBINE COMPARTMENT HEATER No.1
TURBINE COMPARTMENT HEATER No.2
23QT
23HG
23HA-1
23HA-2
23HT-1
23HT-2

```
*D 0.0
*A 1.0
*H 3.0
*L
20A
*L
10A
*L/2
16A
*R/6
5A
*H 2.5
*C/2,7
AS#
*H 4.0
*C/5
SW
*C/5
A
*H 3.0
*L/249,255
D#
*C/2
5kW
*H 2.5
*L/4
4.5kW
*D 90.0
*H 3.0
*L
LUB OIL
HEATER
(SHT.11A)
*L
GENERATOR STATOR
HEATER
(SHT.11A)


etc...
```

Figure 5.8 Example of text input from a file

graphics functions are used. The process of shifting an item on a drawing from one location to another is a good example of this. The user identifies the items to be shifted by using one of the select functions, enters the three dimensional shifts required, and the item is subsequently shifted in 3-D space. This will result in it assuming different projections in any views that are currently being displayed. It is possible that the user will have already annotated one view with, perhaps, dimensions and a title, and it is obviously desireable that this information is shifted on equivalent amount to match up with the new projection. This is achieved automatically providing the user ensures that this annotation has been selected along with the 3-D item. In this example, wherever selected annotation is encountered the software determines over which view it lies, and computes, for that view, an apparent 2-D shift equivalent to the 3-D shift that is being applied to the main drawing data. By applying this shift to the annotation its positional relationship to the main drawing is maintained. Obviously, if annotation has been added to more than one projection of the item this will have to be selected in every appropriate view in order that it is all shifted correctly. This may be achieved by multiple use of the select function, taking advantage of its cumulative mode of operation.

5.9 The Plotting Process

At both Balfour Beatty and Imperial College flat bed type plotters, as described in Chapter 3, have been employed, in the former case the CIL 4/74, and at the latter models that were developed at the college. Indeed a small part of this project was to develop the software to drive a flat bed plotter on line to a PDP 11, and is described in Appendix A. This plotter suffered from the disadvantage that continuous monitoring by the computer was necessary, which would only be practical under the single user operating sysem. For multi-user operation a different approach is necessary, and to this end a microprocessor controlled plotter was developed as a separate project. Work is currently in progress to integrate this with the TIGER system to enable its use on line under the RSX-11M operating system.

For reasons explained later the plotting process is best split into two distinct stages.

5.9.1 Requesting a Plot

A plot of the current drawing may be requested at any time during the drawing process, although users should be aware that plotting is a time consuming and hence expensive operation that should only be carried out when absolutely necessary, normally when a drawing, or modifications to a drawing, are complete and a new 'master' copy is required.

The plotting process is initiated by menu command allows the drawing that is currently in the workspace to be plotted. If it is necessary to plot a drawing that is currently on file that drawing must first be recalled to the workspace. This may be seen as a limitation, and is indeed the subject of further thought, but is currently necessary if the user is to be able to provide the information that is needed.

In a large drawing office the draughtsman is unlikely to be able to supervise the plotting of his drawing, so at this stage he must provide all the information that is necessary for the operator to load the correct paper type and size, pens etc. The initial stage of requesting a plot is a short question and answer session on the alphanumeric terminal, during which the draughtsman is able to specify these parameters. He will then be expected to digitise a point on the drawing that will correspond to the bottom left hand corner of the paper on the final plot. By specifying this point during the plotting process rather than at the start of the drawing the maximum flexibility is achieved to cope with changes of specification or layout that have occurred. This feature may be used in conjunction with variations in the plotting scale factor to adapt drawings to fit different sizes of paper. When the user has supplied all the necessary information and confirmed that all questions have been answered to his satisfaction the system processes the drawing and produces a file of the commands necessary for the plotter. The user may then continue drawing since this file is handled independantly by the second stage of the plotting process. An attempt has been made in the graphics software to use what has become known as 'Calcomp Compatible' subroutine calls to generate the plot commands. Most plotter manufacturers distribute a library of such subroutines tailored to their particular machine, and the system can therefore be made to generate plot files for different machines merely by building the plotting tasks to access the required set of subroutines with little or no changes to the main code.

5.9.2 The Plot File Despooler

The on-line operation of the plotter under the single user operating system meant that no further work could be done while the plotter was operating. This is obviously a grossly inefficient method which had to be avoided for the multi-user system. The multi-tasking capability of RSX-11M enables the plotter to run on-line in parallel with any other tasks, which overcomes the above problem but introduces a number of others not previously encountered. There is no way of predicting if and when a user will request a plot, and it is possible for this to happen while the plotter is still outputting a previous drawing. It is therefore necessary to introduce a queuing system where plot requests are honoured only when the plotter becomes free. This principle is well known in most computer applications as applied to line printer output, and is generally known as 'spooling' (Shared Peripheral Operation On Line). The system developed for plotting is of necessity slightly more complex than that for a printer because a certain amount of operator intervention is necessary, and information is not only passed to the plotter but also back from the plotter to the computer. The system that has been developed is as follows.

When the user requests a plot the information that would normally be sent to the plotter is instead placed in a file and a reference to that file placed in a master plot queue index file. Each entry in the index file contains not only this reference to the associated plot file, but information about the size and type of paper requested, and all other parameters specified by the user when the plot was requested. An independant despooling task is run by the operator to examine this index file and plot any drawings that have been queued, when the plotter next becomes free. The despooler task normally takes entries from the queue on a 'first-in-first-out' basis, but the operator may override this if need be, perhaps when an important drawing some way down the queue is required urgently. Similarly, entries may be deleted from the queue, information about entries in the queue obtained etc. At the end of each plot the operator must confirm whether the drawing is satisfactory before the entry is deleted from the queue. This is necessary because it is possible for one of the pens to have run out of ink unexpectedly, or some other problem to have arisen which necessitates the plot being repeated. In the normal mode of operation the despooler will automatically inform the operator of the pens and paper to be used for the next plot and then wait until told the plotter is ready before starting the plot. The process is repeated until the plot queue becomes empty, or the operator specifically asks for it to be terminated. Since the queue is held on disk it is possible for drawings to be waiting overnight when the machine is powered down, and the queue does not get lost in the event of a sudden failure of any description. Different types of plotter can be accomodated with very little change to the despooler, since the actual plot data is generated as part of the graphics systems, and the despooler merely reads the commands and sends them to the plotter without attempting any interpretation.

5.10 Long Term Drawing Storage

In designing a filing system by which drawings may be stored in either a complete or semi complete form a number of factors must be taken into consideration. Not least, the file must contain sufficient information to fully describe the drawing, which in a system such as this includes not only the basic 3-D database, but also the relevant views, scale factors etc. that are in operation. Furthermore, in a system that is undergoing constant development it is important that the structure adopted should be able to handle future expansion or changes without existing drawings becoming incompatible and so unretrievable.

It was recognised that in the production system it would be necessary to have two levels of filing, a fast access but essentially temporary system for drawings that are undergoing modification, and a slower access 'archive' type system for drawings that have been finished but must be stored for future reference. In theory there is no need to make any distinction between the two requirements, but for economic and space considerations it is convenient to use two different approaches.

5.10.1 Fast Access Semi-Permanent Files

High speed access to files at random necessitates the use of disk storage, since the only other convenient bulk storage medium, magtape, functions on a sequential access principle and can be very slow at retrieving information close to the end of the tape.

In the current system a drawing is identified by its Drawing Number which, contary to its title, may consist of any character string of up 22 characters in length, this being two more than the maximum anticipated. It is this number that the draughtsman will use to recall the required drawing from storage. In order to locate the drawing quickly an index file separate from the main drawing files is maintained which contains general information about the drawing such as when it was saved and by whom, how many times it has been accessed etc. In addition the index file contains a correlation between the drawing number and an internal sequence number used to identify the main drawing file, the name of which is derived from this number. During the recall process the system can quickly find the correct file from the index and read it without reference to any other drawing files.

The drawing file itself contains a number of pieces of information about the drawing which are decoded on its recall. Starting from the first block all the relevant system information about scale factors etc. is stored, with pointers to indicate appropriate locations. This is followed by a copy of the indexed parameters and finally the basic 3-D database. The overall size of the file is obviously dependant on the complexity of the drawing.

A number of facilities are available to the user in addition to the basic saving and retrieval of drawings. These include the output of a directory listing showing all the files currently in store and the ability to delete selected drawings. The latter process has a built in protection feature in that a drawing may only be deleted if the current users name matches the name recorded when the drawing was saved. For obvious reasons this may be overridden by users with appropriate privilege status. An index file is stored on a data pack and will only indicate which drawings are available on that particular pack. In this way different packs may be used for different projects and users only have access to the drawings appertaining to their particular project. An operator function allows differing disk drives to be assigned to each user station depending on the packs available or required.

This approach differs considerably from earlier solutions to the problem in a number of ways. The indexing capability has been expanded to allow identification by drawing number as opposed to a simple serial number from 1 upwards and further information about ownership and access has been included. In the early systems all drawings and the index were kept in the same file, the index occupying the first few blocks and indicating at which block within the file each drawing started and ended. This method is limited in that the index is of a fixed length and can only contain a predefined number of entries, and it was necessary to preallocate the file to a given size, this limiting the number and size of drawings that could be saved. These limitations were considered unacceptable within the commercial environment, therefore necessitating the development of the current approach whereby any number of drawings of any size may be handled, the only limit being the space available on the pack. Furthermore, the new system has distinct advantages over the old when archiving becomes necessary, as outlined below.

5.10.2 Archive Storage

The archiving procedure is intended for drawings which have been completed and may not require access for a considerable length of time. To keep such drawings on disk would obviously be advantageous since they could be accessed quickly, but there are a number of reasons why this is not the best solution. A disk pack may be physically very large, and storage becomes a problem when considerable numbers are involved. Furthermore, disk packs are very expensive and it does not make economic sense to tie up large numbers of packs in this way. A far better solution is to use magnetic tapes since these are very much cheaper than disk packs, a cost per byte ratio may be as much as 20:1, and they are considerably smaller than the average disk pack. They do however suffer from a number of limitations. Unlike a disk a magnetic tape is a sequential device and it is necessary to read right through a tape to access files some way from the start. The time penalty for this is obvious. Furthermore, it is very easy to delete or add files on disk, but on tape deletion is impossible and files must always be added to the end of the information currently on the tape. It is therefore obvious that when a drawing is to be archived the user should be certain that it will not be updated again shortly.

The archiving routine functions in much the same way as the operation previously explained for short term shortage, with the exception of the index file handling. It is not possible to keep the index file on the tape in the same way it is kept on disk because it is subject to continuous updates and extension, which cannot be accommodated on tape. Instead, the index files for all tapes are kept on the system disk and additionally contain the label of the tape on which the drawing is kept. Using this principle the system can very quickly determine which tape must be loaded to retrieve a particular drawing, the file format for which is identical to that used on disk. An alternative approach would have been to add futher information to each drawing file and scrap the index file. This would result in an inherently slower system where it would undoubtedly be quickest to maintain a manual indexing system. Most companies maintain a register of all drawings currently in their possession, the conventional approach being a card index system. There is ample scope for computerising this procedure and the archiving system represents the first stage of such an operation. The computer has on file a list of all drawing currently stored, and it is a fairly straightforward matter to expand this to include all the information and features necessary for a fully computerised drawing registry. This development is discussed in greater detail in Chapter 7.

Unlike the disk filing system, the archiving procedure is purely an operator function and cannot be initiated from a user station. This is considered essential since a considerable amount of operator intervention in loading the correct tape is required and the process may take some time to complete, particularly if a tape drive is not currently free. Furthermore, a decision as to whether a drawing is complete and to be archived will probably not be made by the draughtsman but by an engineer who will generally not be familiar with the operation of the graphics system.

5.10.3 Other Filing Systems

Magnetic filing systems such as those previously described are renowned for compatability problems when transferring information from one computer to another, particularly one from a different manufacturer. They are, however, very efficient in that packing density is high and large quantities of information can be accommodated, and it is not feasible to consider storing drawings as a matter of course on any other medium. The graphics system will allow a drawing database to both input and output in card image format, albeit very large and inefficient, on any device the user choses, making it possible to transfer data on, say, paper tape, with a better likelyhood of compatability. The use of this facility is to be discouraged at all costs!

5.11 Error Processing and Recovery

A function that was all too frequently overlooked in early development systems was the detection and correction of errors of all types. Error recovery was of little consequence in experimental situations which could normally be restarted by an operator experienced with the software and able to correct manually any unexpected situations, such as locked files, that may result, and in any case it was rare that valuable data would be lost. It was recognised that in the commercial environment the users would be draughtsmen with little or no computer knowledge who would be unable to attempt such corrections and furthermore would be handling valuable data, so a comprehensive and automatic recovery system must be incorporated. Problems normally result from two eventualities, operator errors or system failures. The way in which such situations are handled differ considerably from one another.

5.11.1 Operator Errors

These error conditions normally result from the user doing the wrong thing at the wrong time, and are normally detected by the appropriate graphics tasks and remedial action taken locally. As such they may be categorised as 'soft' errors; ones that have no harmful effect on the system and that can be easily overcome. A complex graphics system has hundreds of such situations, and for the convenience of the programmer a special error message issuing task is available, allowing common messages to be shared by different tasks and obviating the need for each task to contain error messages locally. The error task is normally run in parallel with the task requesting the message since it is small and speed of response is important. Each error message is originally entered by the user, along with the number by which it is to be known, in a text file containing all the error messages and grouped purely for the programmer convenience. This file may contain comment lines starting with a ';' character which are ignored by the system. Before this file can be used by the error message task it is preprocessed by a special utility routine into a direct access binary file with each message occupying one record, the record number corresponding exactly to the message number. In this way the error message processor can quickly obtain the correct message and output it on the alphanumeric terminal, appending a bell character to attract the user attention. Four special characters are currently recognised as part of each message that cause the processor to substitute values into the message before it is output. If values are to be substituted they must be passed to the task as part of the mapping process described in section 4.2.2. Each time a substitution character is encountered the next of the values passed is used for substitution. The characters currently recognised are:-

# -	Convert the next value to a signed decimal number and substitute in the message it this point.
$ -	Convert the next value to an unsigned octal number and substitute in the message at this point.
% -	Take the next value, treat it as a RADIX-50 character string and substitute the 3 characters in the message at this point. (RADIX-50 is a DEC subset of the ASCII character set allowing 3 characters to be stored in one word rather than the normal 2).

In addition, a message may be terminated with an '@n' string, where 'n' is another legal message number, causing that message to be automatically output after the current message. A common use of this facility is to append a message such as 'DO NOT PROCEED FURTHER WITHOUT ADVICE' to a number of the more serious error condition messages. For example, an error message is stored as:-

```
OPTION NO. # NOT AVAILABLE, USE #@50
```

If the values passed are 7 and 3 respectively, the following messages will result on the users terminal

OPTION NO. 7 NOT AVAILABLE, USE 3	-	original message
THIS OPTION WILL BE ADDED SHORTLY	-	message No. 50.

The error message task makes no attempt to effect error recovery, which is left up to the task which initially detected it.

5.11.2 System Failures

A system failure will normally result from a program error that causes an unexpected crash, or some form of hardware failure. It is obviously hoped that these will be minimal, but the former must be expected all the time new functions are being developed, and in both cases the best recovery possible must be attempted in order to preserve valuable work. The structure of the graphics system dictates that every task must request that the next task in the excution queue be run before it exits in order that control may be passed between tasks. Obviously if a task exits unexpectedly for some reason this process will not occur and the graphics system will effectively cease to function. In early systems the user had no option but to log back on and try again, but this situation is now recoverable due to the action of background mode. Under section 5.2 it was explained that background mode does not exit, even when control is passed to another task, but instead passes into a state where every 5 seconds it checks that the graphics is still functioning correctly. Basically this involves checking that some task, other than itself, is still running for this user station. If no other tasks are found to be running a crash of some sort has obviously occurred and the system crash recovery task is started by background mode. This task performs a number of functions such as ensuring that none of the random access files has been left in a locked state, which results from a task opening the file to write into it and exiting or crashing without closing the file properly. A large number of system parameters and flags, such as the symbol mode flag, are reset to their correct state, the task execution queue is cleared and the workspace checked for integrity to ensure that the indices have not been lost and the input pointer is left set at the correct position, i.e. the current logical end of file. In the majority if cases a recovery is possible and control is returned to the user in normal background mode. If not, the system is shut down and the user invited to log on again. It is anticipated that in a future version where periodic automatic backup of the workspace is implemented the user will be given the option of returning to the state that existed at the last backup if a full recovery is not possible.

In order to assist program development every failure of this type generates a crash dump containing a full octal dump of the resident common area and a number of pieces of information about other aspects of the state of the system when the crash occurred. This dump is written into a special file which may be listed later if required. In addition a message is sent to the console terminal, normally assigned to the terminal in the computer room, reporting the crash in order that computer staff are aware of the occurrance.

5.12 Help Mode

Although a comprehensive user manual is currently in preparation which expains every function in detail it is often useful to have 'on line' help which is available immediately and without the need to search through the manual, particularly in instances where the basics of a command are known, but the user has forgotten one or two details. This function is fulfilled by help mode, which like most other modes is initiated by menu command. When in help mode the user is expected to digitise over a menu command for which assistance is required. A short explanation of that command is then displayed on the alphanumeric terminal. This will not be as comprehensive as that in the manual, but contains all the main points. Help mode remains active until the user explicitly cancels it by pressing button C. Until this is done all buttons are passed directly to 'help' and no other commands can be issued.

The data for help mode must have previously been entered by the programmer into an appropriately named file, based on the task number. These files are current called 'Tnnnnn.TXT', where nnnnn is the task number without leading zeros. The text in the file is subdivided into sections by lines starting with an *, where each section corresponds to an entry point of the task. The appropriate section is read from the file and displayed on the graphics screen for each help request. Any data preceeding the first * is assumed to be title information and is output for every entry point. This is illustrated in the following example:-

```
This is a title which is always output
*1
This is the information about entry point 1
*2
This is the information about entry point 2
*3
This is about entry point 3
etc.
```

5.13 Logging Off

In order to terminate a graphics session the user must log off, so releasing the workstation and the resources used for the next person. The log off procedure performs a number of internal functions, such as deleting any temporary files that have been allocated during the drawing process. This effectively destroys the drawing that is currently held in the workspace, and to avoid any accidental loss of data the log off procedure reminds the user of this fact and asks for confirmation. If the user indicates that drawing is to continue background mode is restored, otherwise the system is shut down. The graphics system maintains an accounts file containing information about every drawing session, the information being added to this file at log off time, since one entry is the duration of the session which obviously cannot be entered at log on time, when the other information is established.

When log off is complete the shut down task exits without starting any further tasks. In order that this is not interpreted as a system crash by background mode a special flag in the resident area is set to indicate that the user has actually logged off. This flag is detected by background mode, and any other task which may be running in parallel, in order that they too may exit correctly.

6.0 TYPICAL EXAMPLES OF PRODUCTION DRAWINGS

The following sections illustrate examples of typical drawings that have been produced using the TIGER graphics system. In all but one case, section 6.6, these are genuine production drawings, although all have had their titles altered or removed to protect the anonimity of the respective clients.

It is regretted that the detail is difficult to discern on some examples, but this is as a consequence of reducing them from either A1 or A0 size to fit A4 pages, and is not a shortcoming on the original drawings.

A number of examples are of a purely 2 dimensional nature, or not of a truely mechanical engineering discipline. No apology is made for their inclusion since they represent an important part of the development and use of TIGER, and are based on many of the concepts evolved for the 3 dimensional mechanical engineering work.

6.1 Power Station Switchgear

This example is included partly for historical reasons in that it was the first production drawing to be completed. As such it is not one of the neatest drawings available and the hours spent drawing it are best forgotten! It does however show a good degree of repetition, and brought to light the need for a 'text input from a file' facility, which was subsequently incorperated.

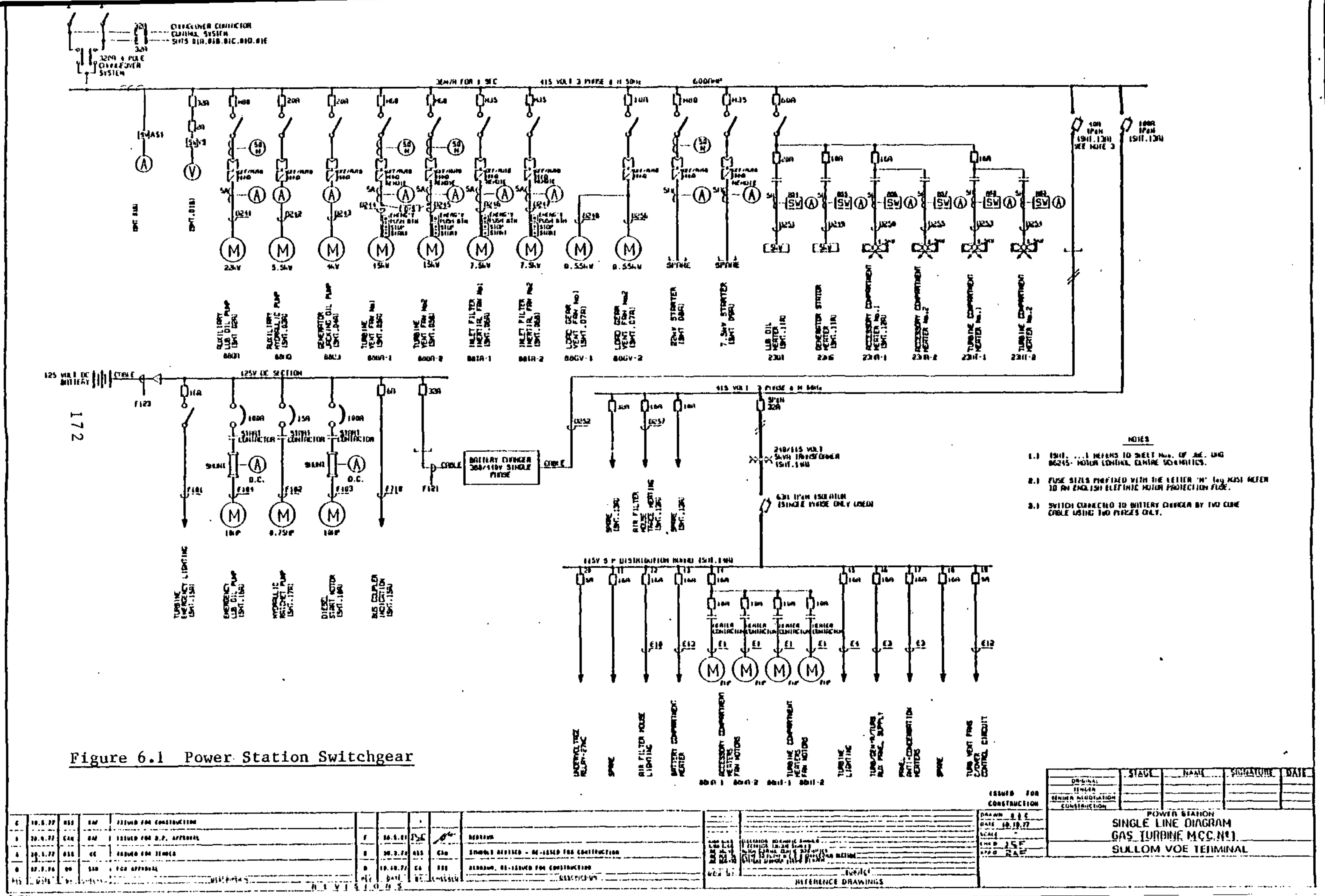

415 VOLT 3 PHASE & N 50Hz
125V DC SECTION
125 VOLT DC BATTERY
BATTERY CHARGER
CABLE
415 VOLT 3 PHASE & N 50Hz
240/115 VOLT 5kVA TRANSFORMER
115V S P DISTRIBUTION BOARD
SPARE
NOTES
1.) (SHT. ....) REFERS TO SHEET Nos. OF JAE. DWG 86245- MOTOR CONTROL CENTRE SCHEMATICS.
2.) FUSE SIZES PREFIXED WITH THE LETTER 'H' (eg H35) REFER TO AN ENGLISH ELECTRIC MOTOR PROTECTION FUSE.
3.) SWITCH CONNECTED TO BATTERY CHARGER BY TWO CORE CABLE USING TWO PHASES ONLY.
STAGE
NAME
SIGNATURE
DATE
ORIGINAL
TENDER
TENDER NEGOTIATION
CONSTRUCTION
ISSUED FOR CONSTRUCTION
POWER STATION
SINGLE LINE DIAGRAM
GAS TURBINE MCC.Nº1
SULLOM VOE TERMINAL
REFERENCE DRAWINGS
REVISIONS

Figure 6.1 Power Station Switchgear

6.2 A Town Plan

Figure 6.2(i) shows a map of a town in the middle east and was used as the basis of a suite of drawings, each depicting one of a number of different services, such as raw water supply and sewage. This illustrates well the use of the overlay system. The basic map was drawn on one overlay, and each individual service on a separate overlay, of which figure 6.2(ii) is an example. By activating specific combinations of overlays a number of composite drawings were produced.

Had it not been for the need to produce such a suite of drawings this particular drawing would not have lent itself well to CAD. The drawing is very irregular and totally non-repetitive, so denying the draftsman the opportunity to capitalise on the facilities to duplicate common sections, work to a grid etc.. It is estimated that the time taken to produce the master map was very little different from the manual method, but the need to produce a number of variations made the use of CAD a worthwhile proposition.

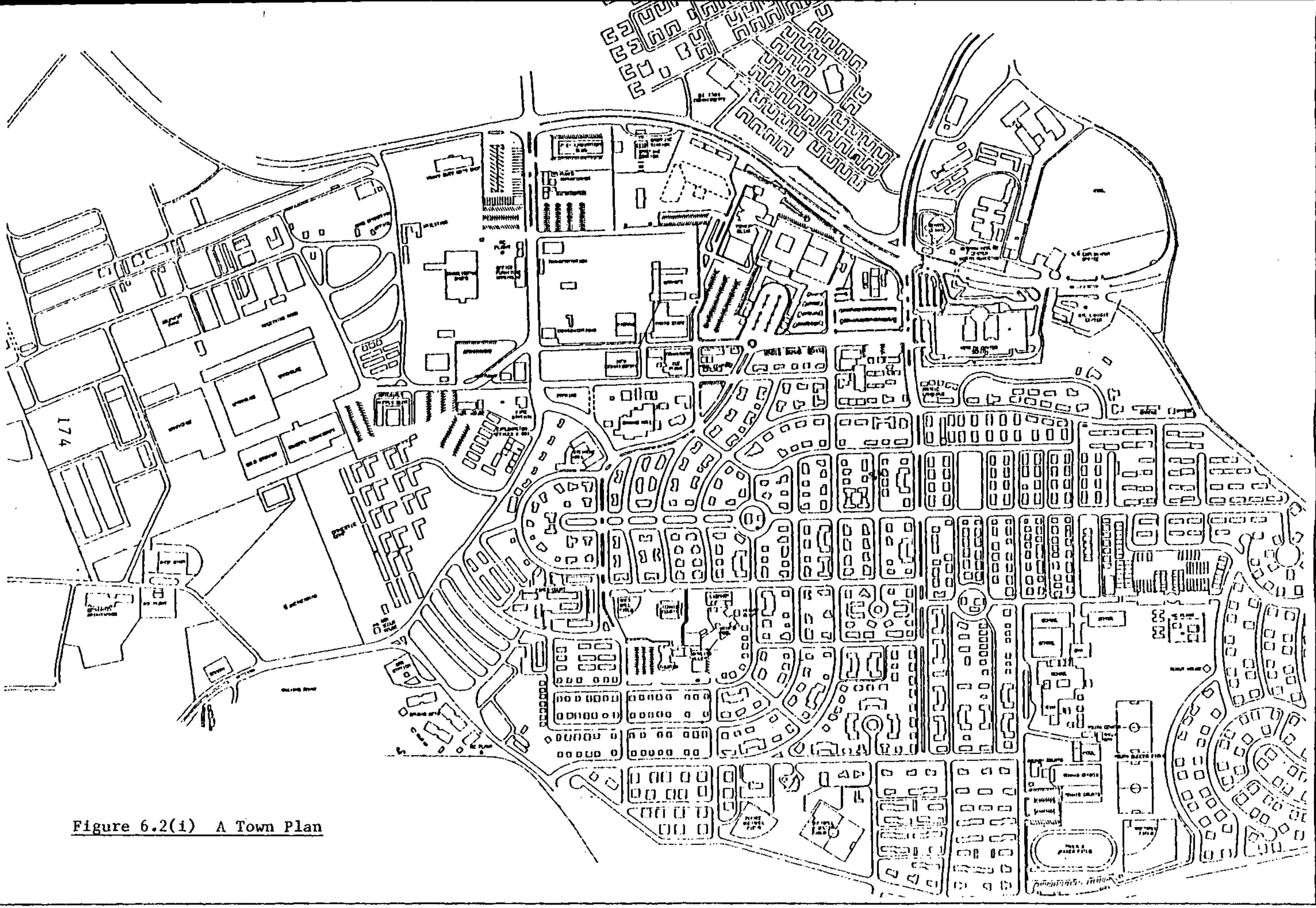


Figure 6.2(i) A Town Plan

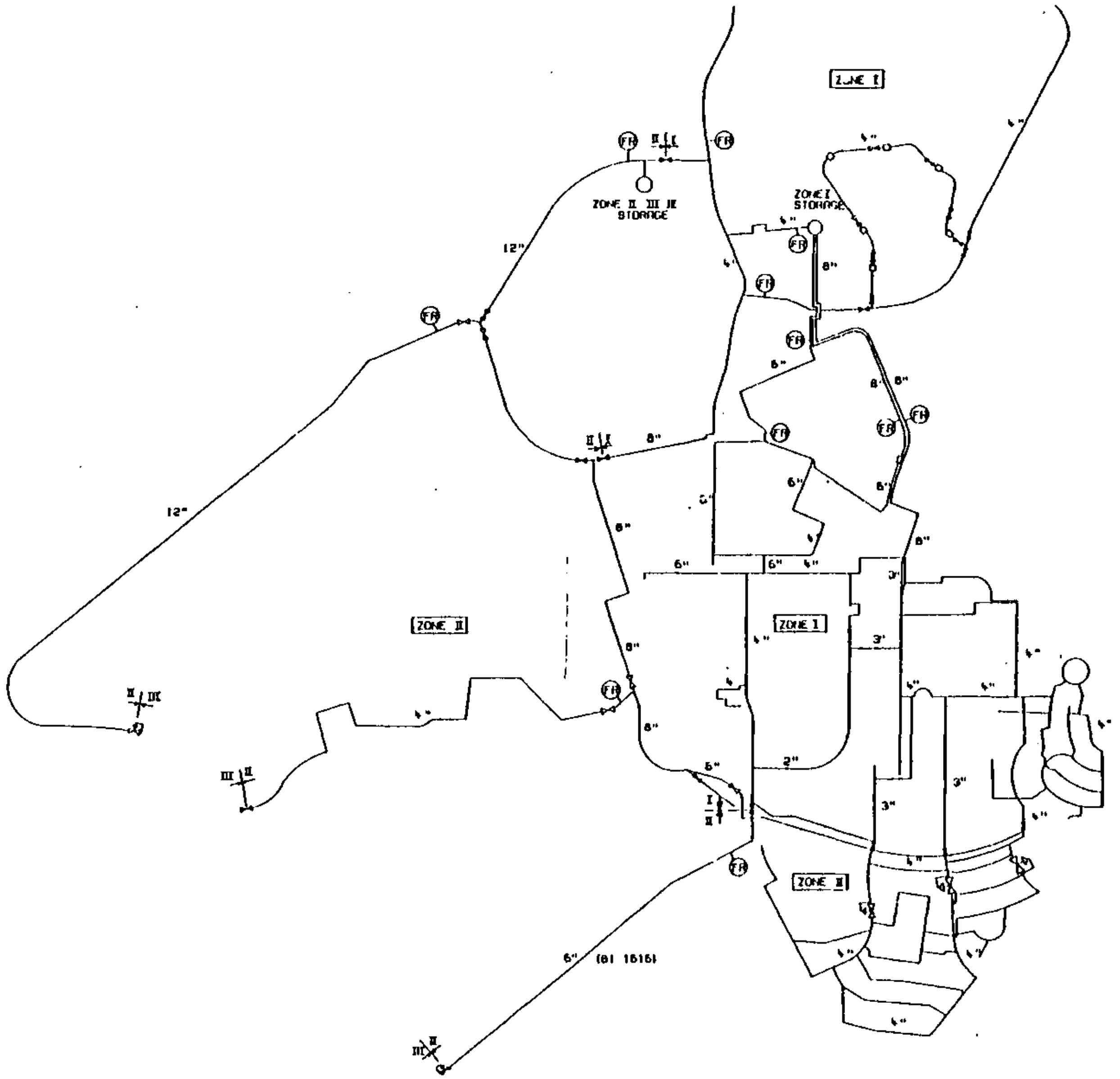


Figure 6.2(ii) A Typical Service Overlay

6.3 Isolator Schematic

This electrical diagram shows the isolator switchgear and indicators for the 275KV Bus in a power substation. This drawing is extremely well suited to CAD in that there is a large amount of repetition and use of standard macros, and furthermore it is one of a suite of similar drawings containing many common features. Use has been made of the grid facility to aid a neat and regular layout

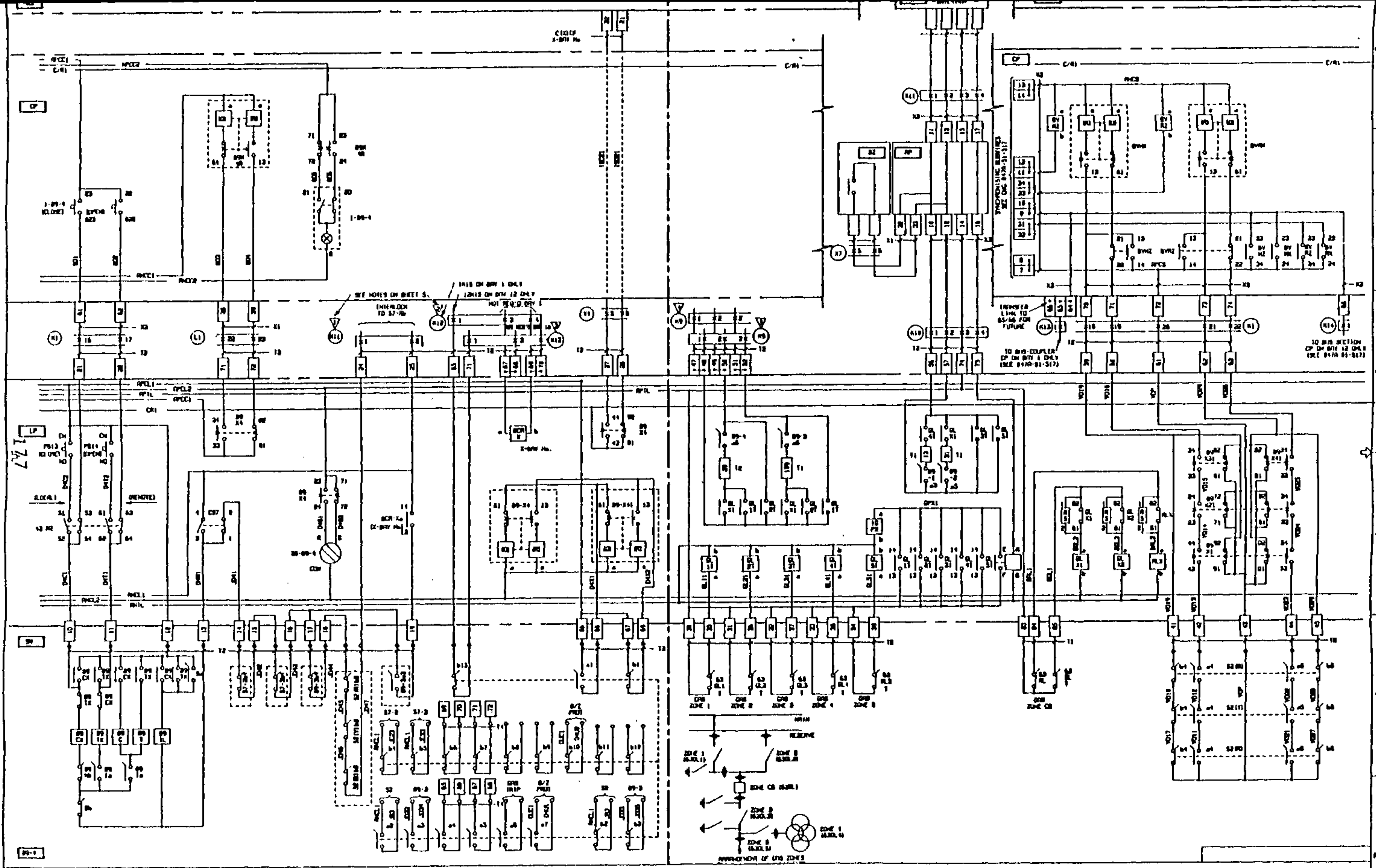
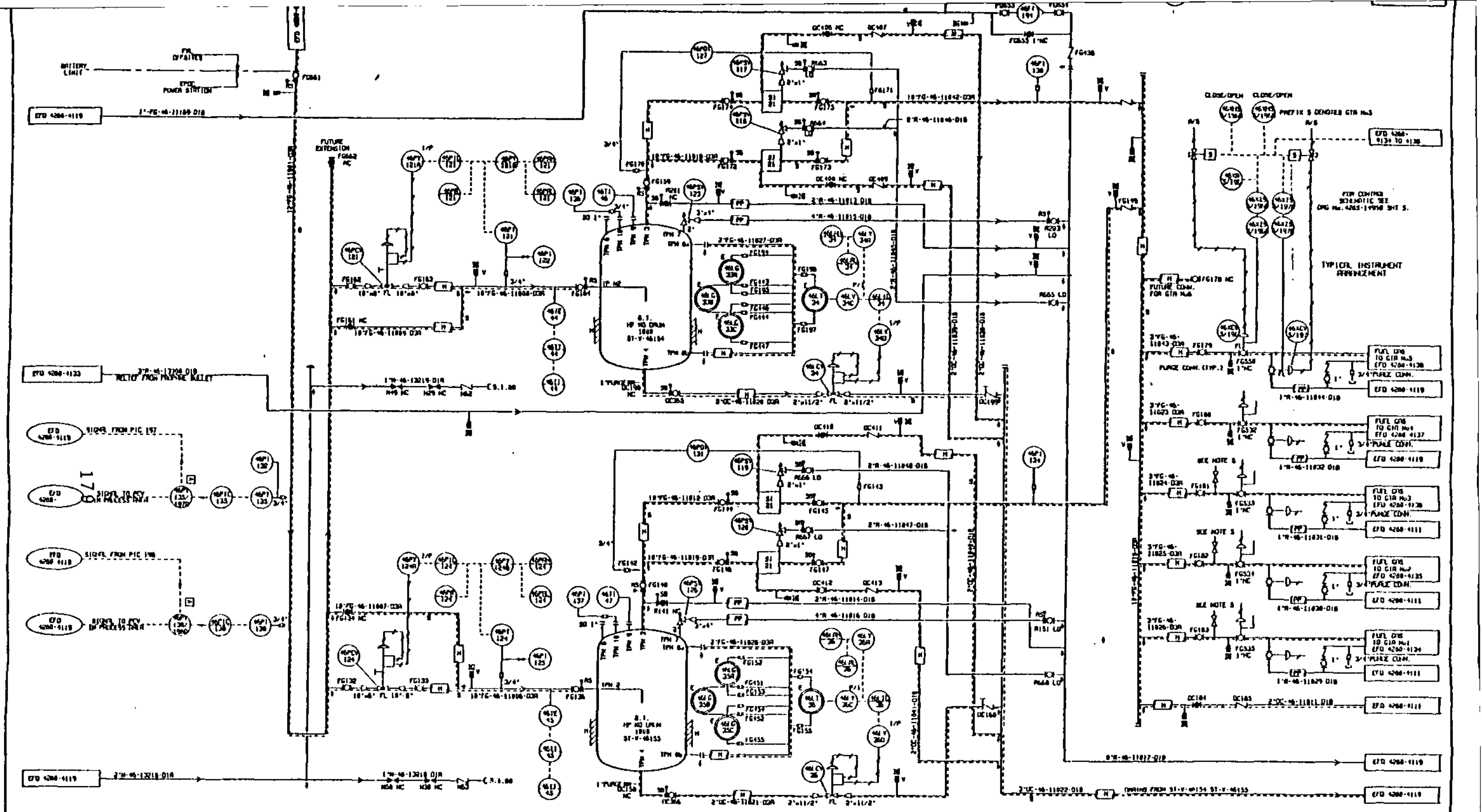


Figure 6.3 Isolator Schematic

6.4 Fuel Gas Flow Diagram

This example is a true mechanical engineering drawing, albeit of a 2 dimensional nature, and is an EFD (Engineering Flow Diagram) illustrating distribution of high pressure fuel gas in a power station. It is fairly repetitive, notice the centre section showing two tanks in an almost identical configuration, and the bottom right showing five similar continuations on other sheets. This drawing also contains a high proportion of dotted lines, which are very much quicker and neater when drawn by the computer than any draftsman could hope to achieve. Differing line thicknesses have been incorperated, making use of all four pens, although when reduced from the original A1 drawing this becomes difficult to discern.



17

Figure 6.4 Fuel Gas Flow Diagram

NOTES

1. ON PIPING ARRANGEMENTS WHERE SPEC 03B IS INDICATED THIS SHOULD READ AS 03A.
2. ALL GAS SERVICE VALVES TO NILE STANDARD SPECIFICATION 03E 300LB CLASS 01A 150LB CLASS
✱ HYDRAULIC TEST DRAIN & VENT VALVES TO BE LOCKED CLOSED & BLIND FLANGE FITTED ON OUTLET SIDE
4. FOR DESIGNATION OF LINE SYMBOLS FG TO DC & R REFER TO SCHEDULE DRG No 4200-4150 SHEETS 1 & 2.
5. FROM GAS COMPRESSOR SEC F 4.L10 EFD

ALTERNATE GAS COMPRESSOR CONN. NOTE SUPPLIED FROM FG LINES TO GTR No's 4 & 5

4200 0764 PROCESS FLOWSHEET FUEL GAS SYSTEM
4200 2054 H.P KNOCK OUT DRUMS ST-V 46154/46155
4200 4057 P.E.D FUEL GAS DISTRIBUTION

ENGINEERING FLOW DIAGRAM
HIGH PRESSURE FUEL GAS SYSTEM

6.5 Piping Isometric

This drawing is typical of thousands of its type and illustrates in isometric projection a short section of a pipe network. This type of drawing is frequently used on site during the construction process.

Although this drawing is truely 3 dimensional each element of it is shown in 2 dimensions; the pipe is shown by centre line only and valves are drawn flat. This is not a shortcoming of the system, which is capable of solid type illustrations, but merely a convention adopted by this, and many other, companies. Furthermore, it is not to an exact scale, some of the longer piperuns having been truncated to make a more convenient shape drawing. This is common practice, but does mean that any form of automatic dimensioning or material take off are difficult to implement. This example shows three essentially similar branches, enabling the draftsman to use duplication to good effect.

Drawings such as this are essentially the only type of 3 dimensional drawing currently in everyday use within Balfour Beatty, and are therefore an important consideration when configuring a graphics system. It is anticipated that experience will lead to greater use of 3-D in visualising piperuns, checking for clashes etc., and for obtaining overall views of an installation for nothing more than aesthetic reasons.

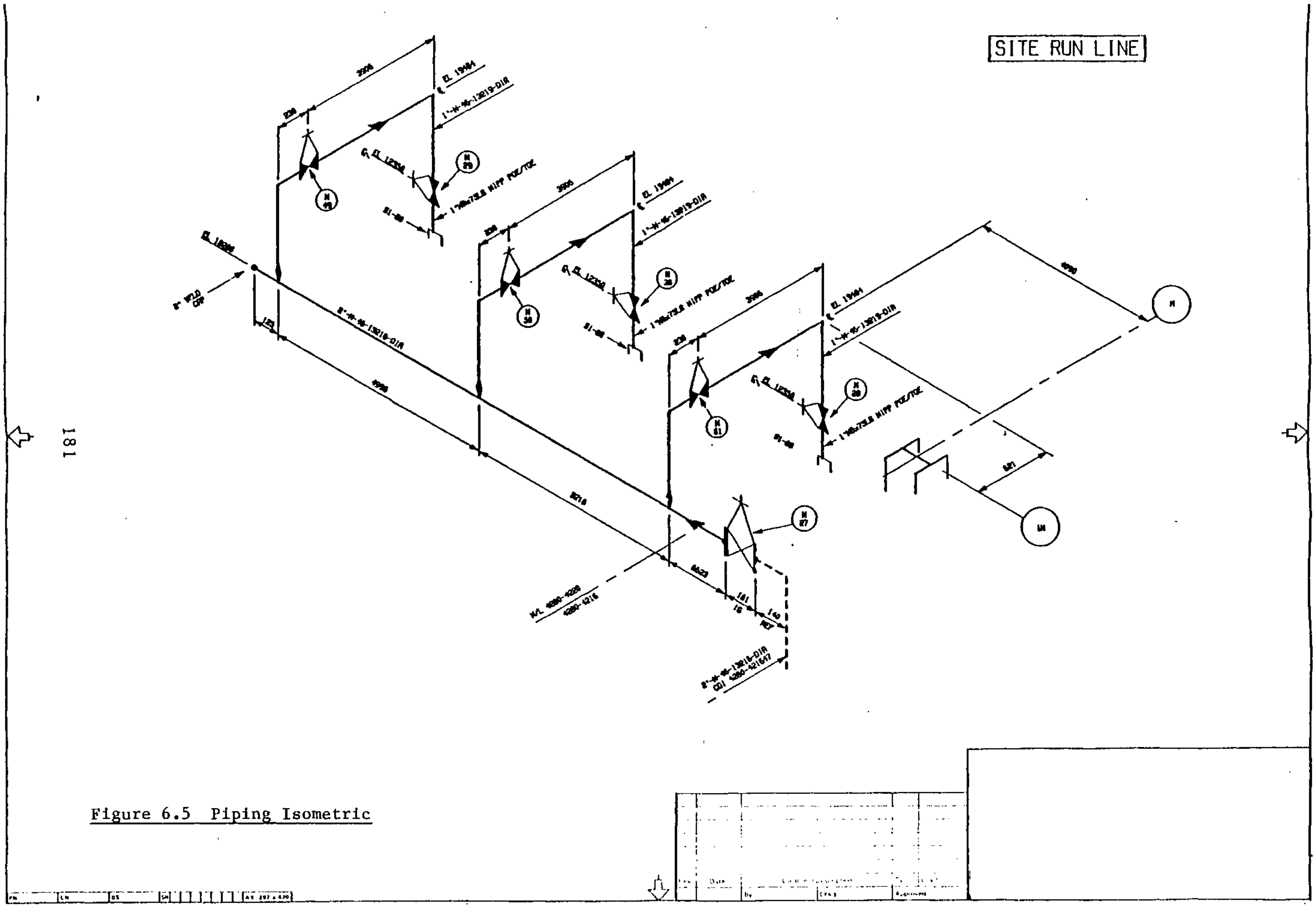


Figure 6.5 Piping Isometric

6.6 Pedestal Bearing

This example does not show a production drawing from Balfour Beatty since it illustrates a subject outside their line of work.

This is a drawing of the components of a bearing in an exploded form, and differs from the previous example in that it shows a solid object, albeit in transparent form with no hidden line elimination. This drawing illustrates well how a number of different views of an object can be generated from the same 3 dimensional data, and how the addition of 2 dimensional information, such as text and dimensions, turns these projections into a complete engineering drawing. Furthermore, it shows how different views may be allocated to give the desired layout. The elevation is built up of four different views, each showing part of the assembly, to enable their relative positions to be different from the isometric projection, the latter also being shown at three quarters scale.

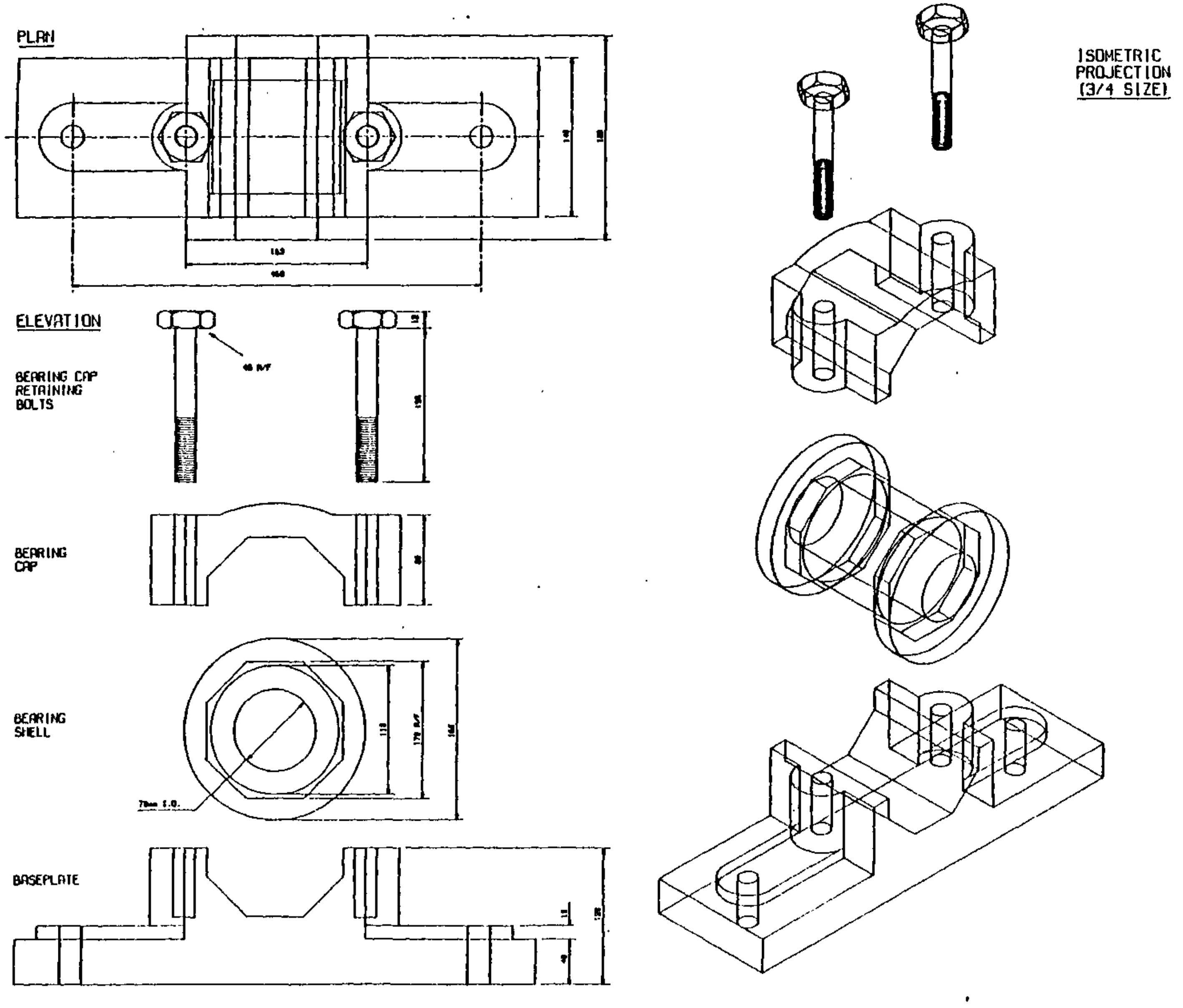


Figure 6.6 Pedestal Bearing

ISSUE AND REVISION | SCALE: | REF. DRAWING No. | DRAWING NUMBER | ENGINEER. APPROVED CHECKED DRAWN | TITLE | No. ISSUE AND REVISION DATE | PLAN NORTH

7.0 CONCLUSIONS

The preceeding chapters describe in overall terms the work that has been carried out in developing the TIGER graphics system for use at Balfour Beatty Engineering, and outline some examples of its use in the drawing office. This chapter examines some of the experience gained during this process and makes suggestions for further development.

7.1 Implementing TIGER in the Drawing Office

A great deal is seen and heard nowadays about the problems of introducing new technology into traditional industries. This falls into two broard categories; the unsuitability of existing techniques necessitating adaptation to the new system, and resistance to change by the workforce and unions concerned. It may therefore be surprising that implementing CAD in the drawing office at Balfour Beatty was a remarkably smooth, if not always straightforward, process. It is felt that there were a number of factors that brought this about.

When the decision was made to develop a system 'in house' rather than buy in a complete package it was recognised that a long lead in time would be required between the initial installation of the equipment and its full productive use. In fact this amounted to a period of over a year, during which the conversion from single user operation and the necessary enhancements were made. As a result the equipment was familiar to the draftsmen and had become an accepted part of the drawing office long before they were required to use it. Many had expressed interest, and some scepticism, during the development, and when it came to the time for them to start using it most were keen to try out this new 'toy' for themselves. In the early days they were encouraged to play with it, find its strengths and weaknesses, and make suggestions as to how it could be improved, bearing in mind that it was still in the early stage of development and many of the more advanced features were as yet unavailable. It was interesting to note the different reactions of the draftsmen.

In general, the older draftsmen were more sceptical and inclined to criticise things that could not, or so appeared at first sight, be done. Inevitably there were a number of problems with the software that sometimes led to drawing corruptions, and it was these same draftsmen who were the first to give up and go back to the manual drawing board under such circumstances. Perhaps through a lack of understanding of computers it did not seem obvious to them that if an operation failed once due to a software problem, the same thing would happen again if they repeated that operation. Frequently the complaints began "I've tried it five times and ...". At the other extreme it was very frustrating to be asked about a problem which it transpired had been present for some long time and nobody had bothered to report. Perhaps naturally, younger draftsmen tended to be more enthusiastic and adaptive, and it was from these that the majority of the feedback was achieved. This proved very useful in providing a function tailored to the way in which draftsmen work, rather than the was engineers and programmers think they work! Training of new users was always a problem due to the limited staffing level of the computer department, but it was generally found that there was a good level of communication between draftsmen, and after a basic introductory course most could become familiar with the facilities by word of mouth and from the users manual.

Whilst there are advantages in developing a system in the above manner there are also a number of operational disadvantages. Perhaps the most serious is that programs tend to be used long before they can be fully tested because testing time for new, and often incompatible, programs is limited to the hours when no productive work is in progress, which contrary to this requirement it is desireable to minimise. Further, when the specification of a function has changed, probably resulting in different prompts and input, it is alarming to the draftsmen if this is suddenly introduced without sufficient warning, and doubly so if it proves to be less than 100% reliable. The way in which these problems are normally overcome is to 'fix' the software at some date and release this for production, then perform development on a separate system, introducing new changes all together, with appropriate documentation, as a completely new release, and repeat the process. The way in which TIGER is structured does not lend itself well to multiple versions on the same processor, but since it is anticipated that as a future development a second computer may become available it is not intended to undertake the substantial effort needed to change this.

There are a number of peripheral operations associated with the CAD system that do not fall directly into the drafting category, such as looking after the plotter, keeping pens clean etc.. There is a good case for employing an operator specifically for these types of jobs, but in general the work load has not justified this. A system that has been shown to work well is if the individual draftsmen are each provided with a set of pens, and supervise the plotting of their own drawings. Initially there is a strong temptation to stand and watch the plotter, but once the novelty has worn off the normal drafting process is not unduely affected, since the plotter needs very little supervision. Most drawings are produced in ink, and pens must be carefully looked after to ensure that they run smoothly and do not become blocked. Using communal pens it was found that they were often left for long periods in the plotter, during which time they dried out, presenting the next user with the task of washing them through. Since no draftsmen like washing pens the problem was simply solved by allocating private pens, which they make every effort to keep running smoothly.

The working environment has been found to be very important to the users acceptance of the system. The early workstations, based on storage graphics screens, needed to be kept in fairly subdued lighting, with the screens facing away from any direct sources of light, particularly windows. In a modern office block this can be difficult to achieve, and purpose designed internal rooms were constructed with dark walls and switchable low lighting. However, they have not proved particularly successful because of the level of lighting, which makes it difficult to see anything other than the screens, and a general feeling of claustrophobia. Furthermore, no special ventilation was installed and they tend to become hot and stuffy. With this in mind the later workstations were specifically chosen so that they can be used in the normal office environment. This necessitated brighter screens, which come automatically with the raster technology, and the elimination of bells and buzzes that earlier stations produces to attract the operators attention, this change being for the benefit of other staff in the office. This design of workstation has been well accepted and is generally used in preference to the older type. They have proved easier and faster to use and perhaps most importantly in this context do not take the draftsman out of his natural environment.

To conclude, although CAD can not yet be seen as a panacea for all drafting problems it certainly can, in specific areas, greatly improve flexibility and productivity. The path followed throughout this project has led to a system that is both flexible and powerful, having applications in many areas of engineering. It is anticipated that development will continue over the coming years, taking advantage of new equipment and techniques as they become available, towards a system tailored for every aspect of drafting within Balfour Beatty.

7.2 Suggestions for further work

The scope for further work on a project such as this is almost limitless, since modifications and enhancements can always be found to adapt existing facilities to meet new requirements and develop new functions as the need arises. This is particularly true in an environment such as that at Balfour Beatty, where consultancy work demands that the company obeys numerous different standards as imposed by their differing clients. Under these circumstances it is very difficult to predict what will be required in the future, but the following paragraphs outline some features that it is hoped may soon be incorperated.

7.2.1 Continuous backup of drawing

When making extensive and lengthy modifications to a drawing it is essential that time is not wasted by having to repeat the operation in the event of either the loss of the drawing through some sort of software or hardware failure, or a serious mistake on the part of the operator. Although a comprehensive error recovery procedure has been incorperated there are still a number of circumstances, normally through hardware failure, when the drawing can be lost or irrecoverably corrupted, and of course operator mistakes are totally unpredicatable.

Under the present system it is the users responsibility to save his work whenever he considers it necessary, in order that as much as possible can be recovered if the drawing does become lost. It is felt that some form of automatic saving and eventual recovery, if necessary, would be of advantage and relieve the user of the need to perform this operation. This could easily be incorperated as part of the background mode semi-dormant state, which would initiate automatic backup at predetermined intervals, perhaps every five minutes or after a certain number of operator actions. The log on procedure could then check for the presence of an automatic backup file, indicating that the previous session was terminated prematurely, and offer the user the option of recovery from that file. The logging off procedure would delete any backup files so that the next log on would be executed normally. This could be accomplished fairly simply as an extension of the manual procedure described in section 5.10.

7.2.2 Command chaining

Under some circumstances the user will be required to perform the same sequence of operations a number of times and it would be useful if some form of command string could be constructed to do this automatically. A typical example might be the recall of a macro, addition of some text, and the duplication of the result the required number of times. This type of operation would be performed when adding resistors of different values to circuit diagrams or valves with different pressures or flow ratings to a pipework diagram. To achieve this a number of menu and keyboard operations must be performed, and if it were possible to specify these, and a repeat count, in some form of command language, the operation would become much simpler and quicker. An early attempt was made to provide such a function in the original single user system, but it was cumbersome and relied on the user performing a dummy run almost blind to program the sequence. It was considered too difficult to use and so was omitted from the later multi-user version.

7.2.3 Re-ordering plot data

The operation of the plot despooler, as outlined in section 5.9.2, leads to a number of problems when multi-pen drawings are being produced, particularly with liquid ink type pens.

The drawing is currently reproduced in exactly the order in which the data is stored, which corresponds approximately to the order in which it was originally digitised. This often involves frequent changes of pen, which in itself presents no problem to a multi-pen plotter, but suffers from the less obvious problem that the a pen tends to dry out when not in use and often fails to write for the first few millimetres, or at all, when called back into use. With the current arrangement it is almost impossible for an operator to predict when a pen will be required, which frequently leads to complex plots being ruined.

It is proposed to introduce a plot despooler which sorts the drawing and plots all occurrances of each pen together, starting with pen 1 and working through. Before each pen is used the spooler will either test the pen automatically, probably in the margin of the sheet, or pause to allow an operator to test the pen manually. In either case the plot will not be restarted until the operator has said that the pen is ready. Even with this system it will still be possible for a plot not to be perfect; the pen may run out of ink, or a mark on the paper may stop it from writing. To obviate this the despooler will ask if any pens are to be replotted before the plot is finally terminated. It is hoped that this will lead to a much lower rejection rate, and perhaps surprisingly require less supervision than the current system. At present the plotter is supervised continuously so that it may be stopped manually if a pen fails, the problem corrected, and the plot restarted. It is hoped that the new system will be able to run unsupervised for the duration of each pen, and hence free the operator for other tasks. Experience has shown that most drawings use one pen almost exculsively, and hence the operator will be required for short concentrated periods rather than every few minutes, which further helps avoid continuous supervision. For drawings with pens that do not require attention, such as ballpoints, it will be possible to instruct the despooler to run through the whole plot without stopping.

7.2.4 Integrated drawing registry

A large engineering company will hold many thousands of drawings and maintain an index, still generally on individual cards, to record all the pertinant details about these drawings. Not only does this take up a large amount of space and require considerable maintenance, but is also a process that can be computerised fairly easily. In its simplest form this could consist of almost a direct conversion from the manual method where updates are made, and information extracted, by operators at conventional terminals. When integrated with a computer graphics system the possibilities become far greater.

The graphics system currently maintains an index on every disk of the drawings that are being held on that particular disk. This could be expanded, as an initial step, into a master index on one disk, containing entries for every drawing in a similar form but with an indication of the disk or tape on which they are held. It is then a small step up to a fully computerised drawing registry, the difference between this and the standard registry being that the drawings are also held by the computer instead of the more conventional microfilm. This would enable drawings to be vetted on a terminal before a print is requested, and enable engineers to skim through drawings with far greater ease than is allowed by the current methods. Only when the correct drawing has been identified would a copy be requested. This would normally be produced on a plotter, and instead of the somewhat poor quality often achieved by normal printing techniques would be a new master of the highest quality possible. Furthermore, an integrated system would give project leaders, and similar staff, the ability to obtain up to the minute information on the state of any drawings, whether they are complete, what revisions have been made etc..

A fundamental requirement for this system to function efficiently is a fully computerised drawing office, since all the time manual and computer produced drawing are handled side by side the full advantages of the system would be difficult to realise. A graphics system capable of handling every type of drawing produced is therefore a mandatory requirement. Even so, a long lead in time is likely since the registry will still be holding large numbers of drawings produced before the computer became available. It would undoubtedly prove too time consuming and expensive to convert all these existing drawings to computer format, unless an automatic digitiser of the form outlined in Appendix B becomes available in a very sophisticated form. It is therefore anticipated that a fully integrated drawing registry will not be developed as part of the TIGER graphics system in the very near future.

7.2.5 Material take-off capability

A large percentage of engineering drawings describe equipment, buildings etc., and are used as the basis of a 'shopping list' of items that must be purchased in order that they may be constructed. The process of extracting this information from the relevant drawings is known as 'material take off', or MTO for short.

MTO is a lengthy and tedious process that is well suited to computerisation. Since every item of equipment that has been placed on the drawing is already known to the computer it is quite possible to expand the information stored to include relevant MTO details such as material type, supplier etc.. The major problem with MTO is the very comprehensive database that is necessary to hold all the information about every item. For example, a typical ball valve will have a different specification depending on a number of parameters such as pipeline diameter, pressure rating, temperature, fluid carried etc.. A system must be devised that is both efficient and comprehensive, able to locate correct information for every item of equipment. MTO is largely an I/O bound process, with much searching through databases and relatively little computation. Until recently the cost of the storage needed to hold these large databases could be prohibitive. Recent advances, particularly winchester drives, have dramatically reduced these costs and made MTO a much more feasible proposition on minicomputers.

The integration of a material take off facility into the TIGER graphics system is currently being investigated as a separate project.

7.2.6 Colour Displays

A problem that has been encountered regularly by draftsmen using the TIGER system is that the monochrome display cannot satisfactorily indicate a number of the differences between pieces of a drawing, such as the pen number with which they are to be drawn, the current select status, overlay etc.. This problem is particularly acute when the drawing has been ammended by different draftsmen, and frequently the easiest method of identifying pen requirements is to produce a rough plot, or to go through a series of displays, which is both time consuming and still leaves no adequate record. These, and a number of similar problems, could be resolved by the use of colour, which until recently has been both technically undeveloped and expensive, particulary for a small system. Recent advances, mainly in terms of resolution, have made them a more practical proposition, and they are becoming more widely available at prices that compare favourably with monochrome displays. It is very unlikely that Balfour Beatty will buy any further workstations based solely on monochrome displays.

The way in which colour is used will probably vary considerably depending on the application. It is anticipated that, for example, civil drawings may be required to highlight different service overlays over a basic map, whereas piping drawings may require to highlight different component types in different colours, irrespective of the overlay. It is obvious that, whilst adding vast new scope to the system, a new series of problems, clashes of interest etc. will be encountered. Only by formal proposals and discussions between the various interested parties will a satisfactory compromise be achieved. The structure of the sofware is sufficiently flexible to allow a colour facility to be added with very little effort, and still remain compatible with existing drawings, although depending on the conventions chosen they may require small patches to bring up to full colour standard. It is anticipated that, through suitable programming of the workstation microprocessors, colour and monochrome workstations could be run simultaneously.

REFERENCES

1) C. B. Besant et al — A FULLY INTERACTIVE COMPUTER AIDED DESIGN SYSTEM.
C.A.D Volume 4 no. 5 October 1972

2) C. B. Besant — THE USE OF CADMAC SYSTEMS IN GENERAL DRAFTING
Department of Mechanical Engineering
Imperial College, London, England.

3) B. Gott, R. Tilbrook — COMPUTER AIDED DRAFTING: THE NATURE OF THE PROBLEM AND THE EQUIPMENT USED.
C.A.D. Centre, Cambridge, England.

4) C. Yi, C. B. Besant, A. Jebb — THREE DIMENSIONAL DIGITISING TECHNIQUES FOR COMPUTER AIDED ANIMATION.
Paper presented at CAD 76
Imperial College, London, England.

5) C. B. Besant, A. Jebb — MECHANICAL ENGINEERING DRAFTING USING CADMAC
Imperial College, July 1974.
Imperial College, London, England.

6) W. M. Newman, R. F. Sproull — PRINCIPLES OF INTERACTIVE COMPUTER GRAPHICS
McGraw Hill 1973.

7) Anon — CADMAC - A MODULAR INSTRUMENTATION SYSTEM FOR DATA HANDLING
Euraton EUR 4100e March 1969.

8) R. Grindley — DEVELOPMENT AND APPLICATION OF A LOW COST CAD SYSTEM IN MECHANICAL ENGINEERING.
PhD Thesis July 1973
Imperial College, London, England.

9) A. Hamlyn — THE APPLICATION OF CAD TECHNIQUES TO BUILDING ENGINEERING DESIGN.
PhD Thesis July 1974
Imperial College, London, England.

10) P. M. McLintock — THE APPLICATIONS OF CAD IN STRUCTURAL ENGINEERING ANALYSIS
PhD Thesis November 1974
Imperial College, London, England.

11) A. W. Nutbourne, R. B. Morris — A CUBIC SPLINE PACKAGE
Cambridge University Engineering Dept. 1972.
Cambridge, England.

12) C. Weitzman — CURRENT AND FUTURE GRAPHICS DISPLAYS FOR MILITARY SYSTEMS
State Development Corperation
Santa Monica, California, USA.

13)	Anon	DIGITAL EQUIPMENT CORPERATION PDP 11/RSX-11M/DOS/BATCH Manuals
14)	C. Yi	THE USE OF COMPUTER AIDED DESIGN TECHNIQUES IN DYNAMIC GRAPHICAL SIMULATION PhD Thesis March 1976 Imperial College, London, England.
15)	D. Thompson	A PROPOSED CAD SYSTEM FOR ENGINEERING AND POWER DEVELOPMENT CONSULTANTS LTD. Imperial College, March 1978 Imperial College, London, England.
16)	M. Cooley	COMPUTER AIDED DESIGN - ITS NATURE AND IMPLICATIONS TASS Head Office, Richmond, Surrey.
17)	Anon	COMPUTER GRAPHICS Quarterly report of Siggraph-ACM Vol II No. 3 Fall 1977
18)	P. Huckle	THE APPLICATION OF CAD TECHNIQUES TO SITE LAYOUT AND PLANNING PhD Thesis, Imperial College, Dec 1975. Imperial College, London, England.
19)	F. Ghassemi	COMPUTER AIDED DESIGN TECHNIQUES IN DATA PROCESSING FOR FINITE ELEMENT ANALYSIS. PhD Thesis, Imperial College, July 1978. Imperial College, London, England.
20)	Cabinet Office	COMPUTER AIDED DESIGN AND MANUFACTURE HMSO Report January 1980
21)	Various	NUMEROUS ARTICLES AND REPORTS IN THE COMPUTING AND ENGINEERING PRESS
22)	I.I. Sutherland	SKETCHPAD: A MAN-MACHINE GRAPHICAL COMMUNICATIONS SYSTEM Proc. SJCC 1963.

APPPENDIX A - THE FLATBED PLOTTER

A.1 Introduction

The excellent workshop facilities at Imperial College have enabled the CAD section to construct several pieces of equipment in prototype form, two of which have been flatbed plotters. The original model, built some years ago, gave valuable service for many years on equipment used by Video Animation Ltd., and has only recently been replaced by them with a commercial plotter, ironically the production version of the college's second model completed in late 1975 and installed on the department's CAD equipment. The second plotter was basically an improved version of the original, as described by Hamlyn(9), position sensing being by means of Moire fringe optical encoders and drive by printed armature motors similar to those used in electric cooling fans for cars. The original method of driving the plotter was adapted for the new model. This relied on an error control module in the systems 'CAMAC'(7) interface rack sending pulses to the motors to generate the correct pen movements. This system was found in practise to be somewhat limited, ramping of the drives at the start and end of a line had to be software controlled, and insufficient speed was available. To overcome these problems a totally new control system was developed where the computer had much more control over all the drive parameters. A servo system on each axis controlled the movement of the pen holder and the carriage, the speed of the motors being controlled simply by varying the voltages applied, the drivers required to generate the correct direction being calculated by the computer in digital form. These were passed to the plotter via the CAMAC interface and interpreted with D/A converters to give the required voltages. In addition the computer could read directly the current position of the pen, enabling instant modifications to the drive to be applied in order to correct any errors in position as they occur. This method involved considerably more software control than before, although results were encouraging and definite improvements in speed and line quality achieved. A block diagram of the control system is shown in Figure A-1.

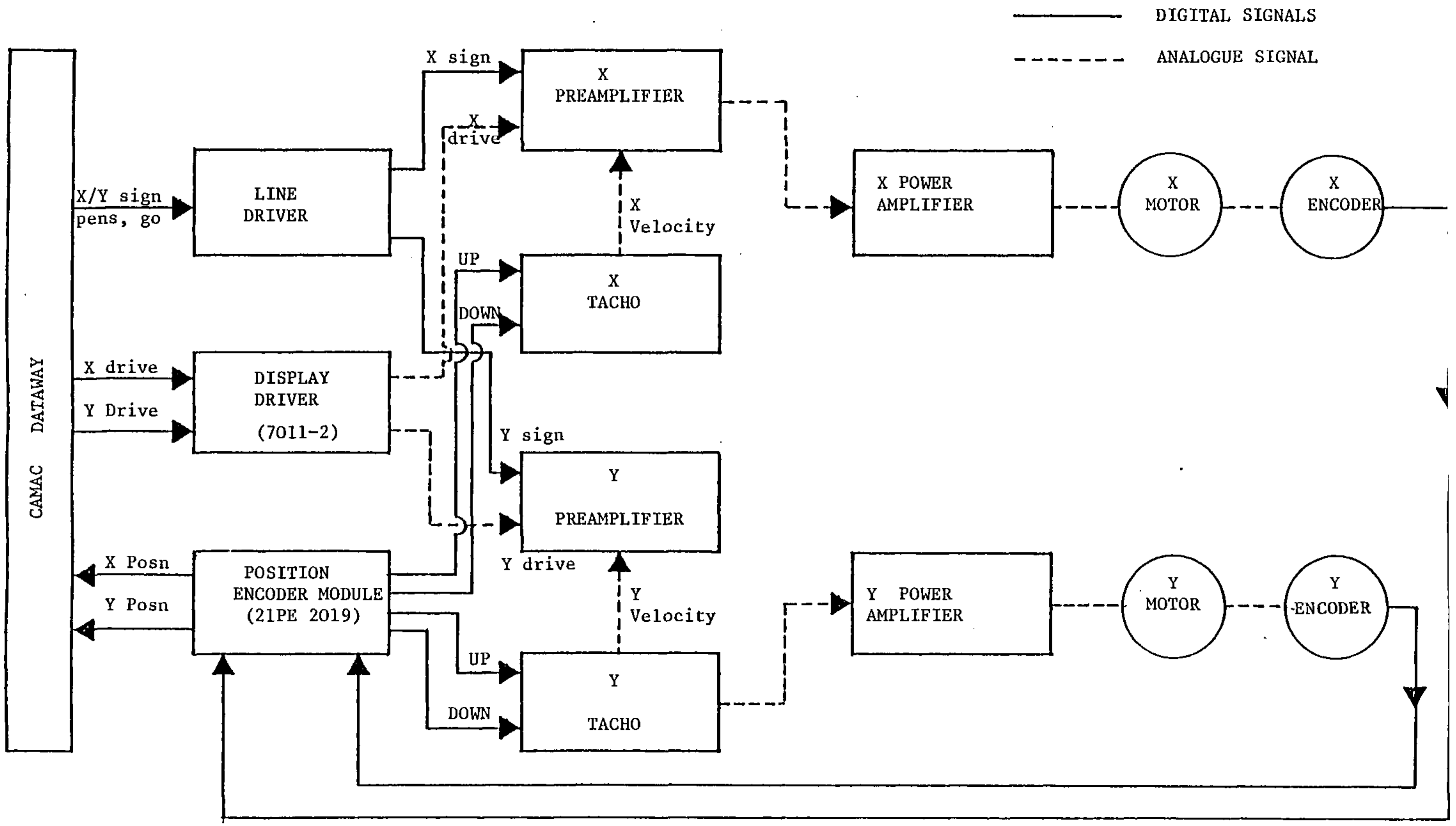


Figure A.1 Block Diagram of Plotter Control System

A.2 Theory and Operation of Software

The movement of the pen in any direction can be resolved into two axial components, which for the sake of convenience can be considered to have a magnitude and a sign. The magnitude is directly proportional to the required voltage across the motor and the sign indicates the direction in which to apply that voltage. Any given vector will have a dominant direction of movement, e.g. if the magnitude of dy/dx is greater than unity, y is dominant. It is convenient to normalise the drives sent to the motors in order that the dominant axis always receives maximum drive, the subordinate axis being sent a proportion of this drive to achieve the desired direction. Up to 1024 magnitudes of subordinate drive are program selectable, the maximum value (1023 on a 0 - 1023 scale) corresponding to the dominant drive. By suitable manipulation of signs and dominance a typical line may be represented as in Figure A-2 overleaf where $0 < \theta < 45$ degrees.

In this example the x axis is dominant and so will receive full drive, of 1023 units. By similar triangles it can be seen that the subordinate, y axis, drive must be:-

$$\text{Y drive} = (y_2 - y_1)/(x_2 - x_1) \times 1023 \text{ units} \qquad (1)$$

Using this technique it is fairly obvious that the actual speed of the pen is dependent on the value of θ, being the vector sum of the axial velocities and hence dependent on the magnitude of the subordinate drive. In order to achieve a constant pen speed and thus maintain consistant line quality on all lines the software computes an overall speed factor to be applied to both axes. This is expressed as:-

$$f = 1023 \; / \; (\text{Dsub}^2 + 1023^2) \qquad (2)$$

where:- Dsub = subordinate drive from equation (1)
f = factor in range 0 to $\sqrt{2}$.

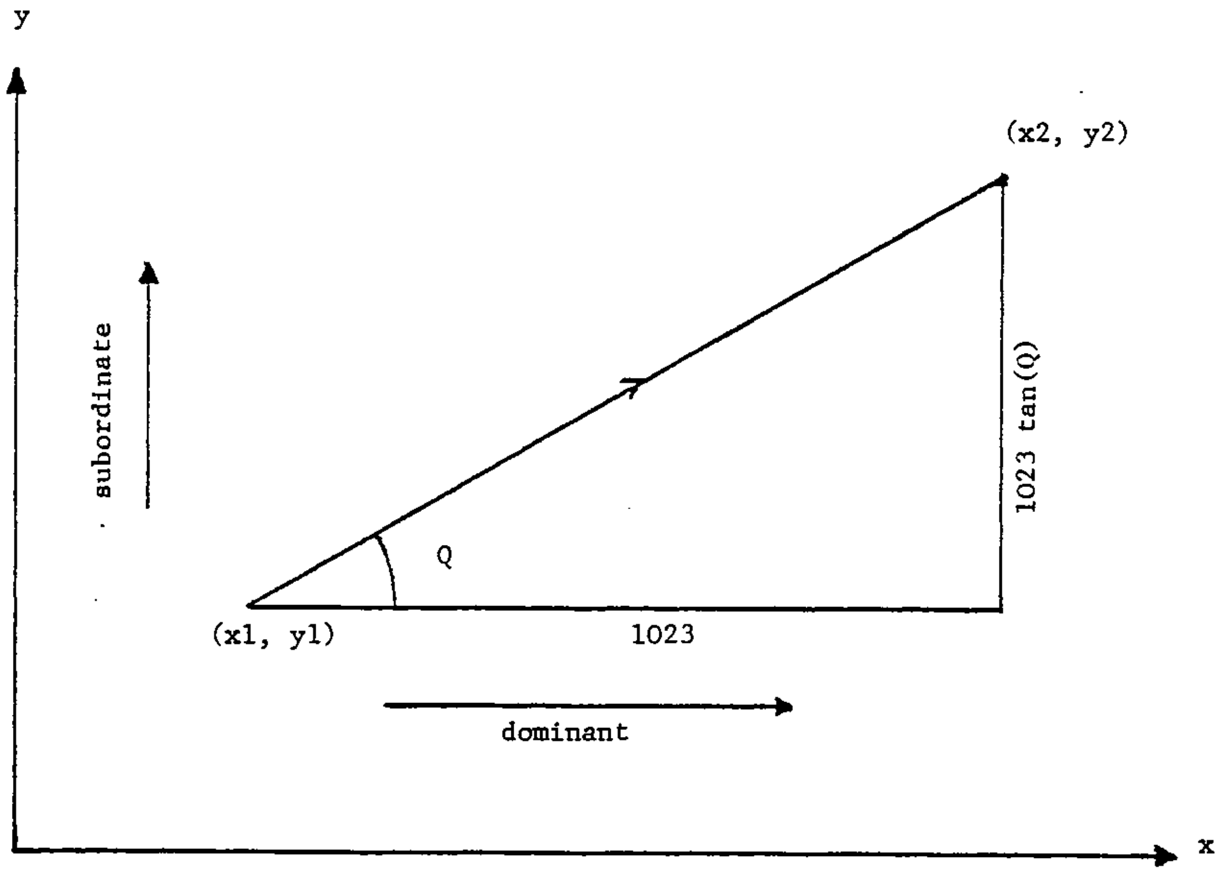


Figure A.2 Dominant and Subordinate drives

Unfortunately, due to non-linearities in the circuits and the finite number of drive magnitudes available it is not possible to simply send the initial angle and expect the plotter to follow the line accurately for its whole length, so the computer constantly monitored the position of the plotter and, at predetermined intervals, applied modifications to the drives in order to keep the pen on the required line. Varying intervals between position checks were tried, but 256 encoder counts (about 1.4mm) on the dominant axis appeared to give the best results. The error in the subordinate axis is a direct indication of the change required in the subordinate drive. Consider the case depicted by Figure A-3.

The plotter should follow the straight line from A to B, but at the first check is found to be at position C, with a subordinate error of E_S. It is evident that the computer must calculate two subordinate drives, namely the drive to bring the pen back onto the correct line at position D and the drive actually required to keep the pen on the line once the error has been corrected.

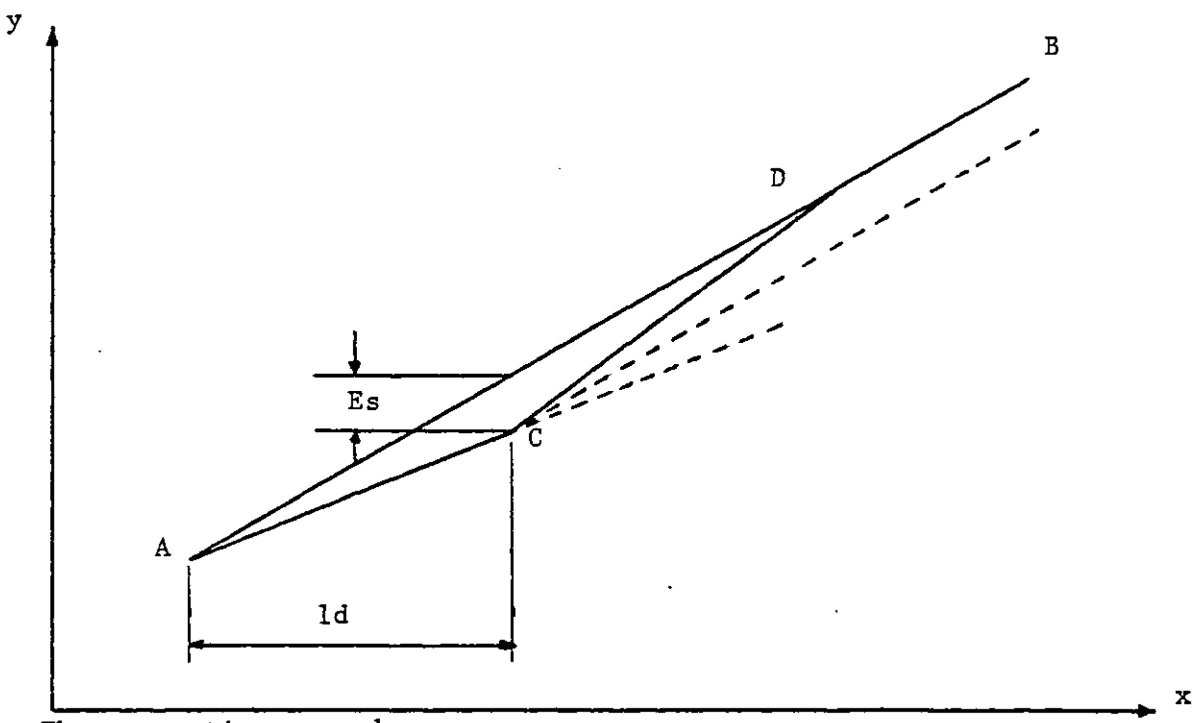
Suppose:-

D_R = 'correct' subordinate drive.
D_S = subordinate drive to correct error in position.
E_S = subordinate error.
l_d = dominant distance between checks.

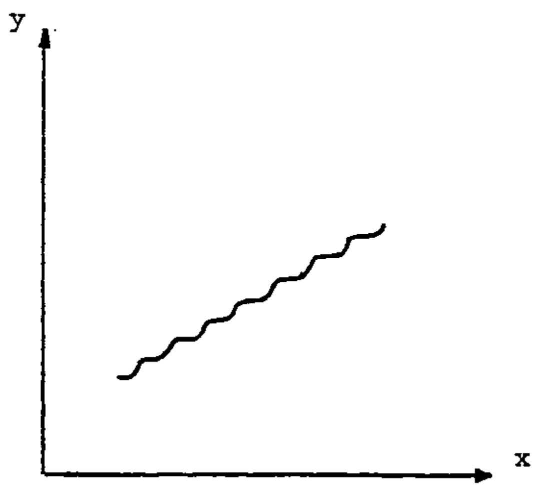
By applying the relationships:-

$$D_R = D_R + (E_f \times f_1)$$
$$D_S = D_R + (E_f \times f_2)$$

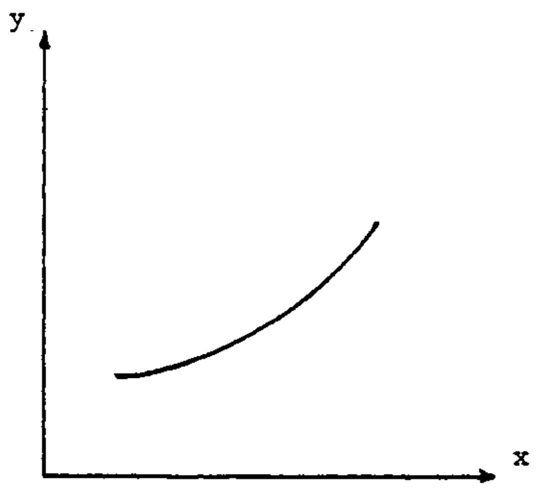
where f_1 and f_2 are constants determined by experiment, the system effectively converges to the required subordinate drive as $E_f \rightarrow 0$. It was found that for l_d = 256 counts, f_1 and f_2 should have values of approximately 1.0 each, larger values lead to overagressive correction



The correction procedure



Effect of over correction



Effect of under correction

Figure A.3 Correction of Positional Errors

and fishtailing, lower values to a gentle curve rather than a straight line.

The acceleration and deceleration of the plotter are important factors in obtaining good line quality. In this system acceleration was controlled by hardware ramping of the drive voltages and not directly by the software. A similar system existed for deceleration towards the end of a line, although the decision at what point to remove the drive in order to stop the pen correctly at the end of the line was made by software. It was found that this distance depended on the speed, the length and the direction of the line and was difficult to compute accurately. A special, very slow, drive was introduced and the main drive regulated to stop the pen just before the end of the line, thus enabling the slow drive to guide the pen to the final correct position. This drive was so slow that any deceleration could be ignored, and the period for which it was applied so short that it was hardly apparent to the observer.

In early systems much trouble was experienced in decelerating correctly at the end of short lines where maximum velocity was never achieved. A set of tests on the plotter revealed the ideal deceleration characteristics shown in Figures A.4 and A.5.

For lines longer than 1mm the deceleration length remained constant, since the plotter always reaches maximum speed. The early problem was immediately apparent since a linear relationship had been assumed between A and C, thus resulting in too little deceleration and the pen being brought to a violent halt at the end of the line. A new system was introduced where section BC was considered linear and values for section AB were tabulated and interpolated by the software. Different values were necessary for the two axes since differences in inertias resulted in different ideal conditions depending on the dominant direction.

```
TEST CALIBRATION ROUTINE, LENGTHS IN PLOTTER INCREMENTS
-------------------------------------------------------

AXIAL TEST IN X

LINE        START        END         START       END
LENGTH      SLOW         SLOW        ERROR       ERROR
------      -----        ----        -----       -----

  400.0       207         255         0.0         1.0
  800.0       528         545        -2.0        -1.0
 1200.0       827         835        -1.0        -3.0
 1600.0      1128        1142        -1.0         0.0
 2000.0      1448        1446        -2.0        -2.0
 2400.0      1762        1750         6.0         2.0
 2800.0      2073        2063         1.0         0.0
 3200.0      2378        2365        -2.0         2.0
 3600.0      2675        2670         1.0        -1.0
 4000.0      2972        2960         2.0         1.0
 4400.0      3238        3235         0.0         0.0
 4800.0      3523        3496        -2.0        -1.0
 5200.0      3797        3753         0.0         0.0
 5600.0      4026        4009        -2.0        -1.0
 6000.0      4287        4253         0.0         1.0
 6400.0      4506        4483        -1.0        -1.0
 6800.0      4688        4672         0.0        -1.0
 7200.0      4852        4837         0.0         1.0
 7600.0      4971        4988         0.0         0.0
 8000.0      5115        5098        -1.0        -2.0
 8400.0      5209        5204        -2.0         0.0
 8800.0      5275        5314        -2.0        -3.0
 9200.0      5327        5375        -2.0         0.0
 9600.0      5388        5427        -1.0        -2.0
10000.0      5458        5480        -3.0        -2.0
10400.0      5487        5518        -1.0        -2.0
10800.0      5503        5556        -1.0        -2.0
11200.0      5551        5583        -1.0        -1.0
11600.0      5536        5601        -1.0        -1.0
12000.0      5579        5611         0.0        -1.0
12400.0      5560        5620        -1.0         0.0
12800.0      5562        5624        -1.0        -1.0
13200.0      5546        5637         1.0        -2.0
13600.0      5564        5652        -1.0        -2.0
14000.0      5588        5660        -2.0        -1.0
```

Figure A-4(i) Optimum decelleration lengths - gantry movement only

```
TEST CALIBRATION ROUTINE, LENGTHS IN PLOTTER INCREMENTS
-------------------------------------------------------

AXIAL TEST IN Y

LINE        START        END         START        END
LENGTH      SLOW         SLOW        ERROR        ERROR
------      -----        ----        -----        -----

  400.0      214          415        -1.0          2.0
  800.0      473          624         0.0          2.0
 1200.0      793          917         0.0          2.0
 1600.0     1066         1213         0.0          3.0
 2000.0     1374         1487         0.0          3.0
 2400.0     1642         1723         0.0          2.0
 2800.0     1919         2063        -2.0          2.0
 3200.0     2232         2326         0.0          2.0
 3600.0     2504         2564        -1.0          2.0
 4000.0     2821         2840        -1.0          3.0
 4400.0     3074         3102         1.0          0.0
 4800.0     3361         3359         1.0         -4.0
 5200.0     3570         3594         1.0          0.0
 5600.0     3821         3813         0.0          0.0
 6000.0     4047         4019        -1.0         -1.0
 6400.0     4230         4209         0.0          0.0
 6800.0     4442         4380         0.0          0.0
 7200.0     4591         4531         1.0          0.0
 7600.0     4784         4670         0.0         -1.0
 8000.0     4859         4789        -1.0         -1.0
 8400.0     4988         6326         1.0          4.0
 8800.0     5117         6170        -1.0          4.0
 9200.0     5250         6098         1.0          4.0
 9600.0     5300         6034         1.0          4.0
10000.0     5375         5974        -2.0          3.0
10400.0     5425         5856         2.0          4.0
10800.0     5475         5786        -2.0          4.0
11200.0     5501         5653         0.0          3.0
11600.0     5510         5653         0.0          3.0
12000.0     5513         5607         1.0          3.0
12400.0     5595         5584        -1.0          2.0
12800.0     5519         5552         0.0          2.0
13200.0     5547         5532         0.0          2.0
13600.0     5516         5512         2.0          2.0
14000.0     5579         5487         0.0          2.0
```

Figure A-4(ii) Optimum decelleration lengths - per carriage movement only

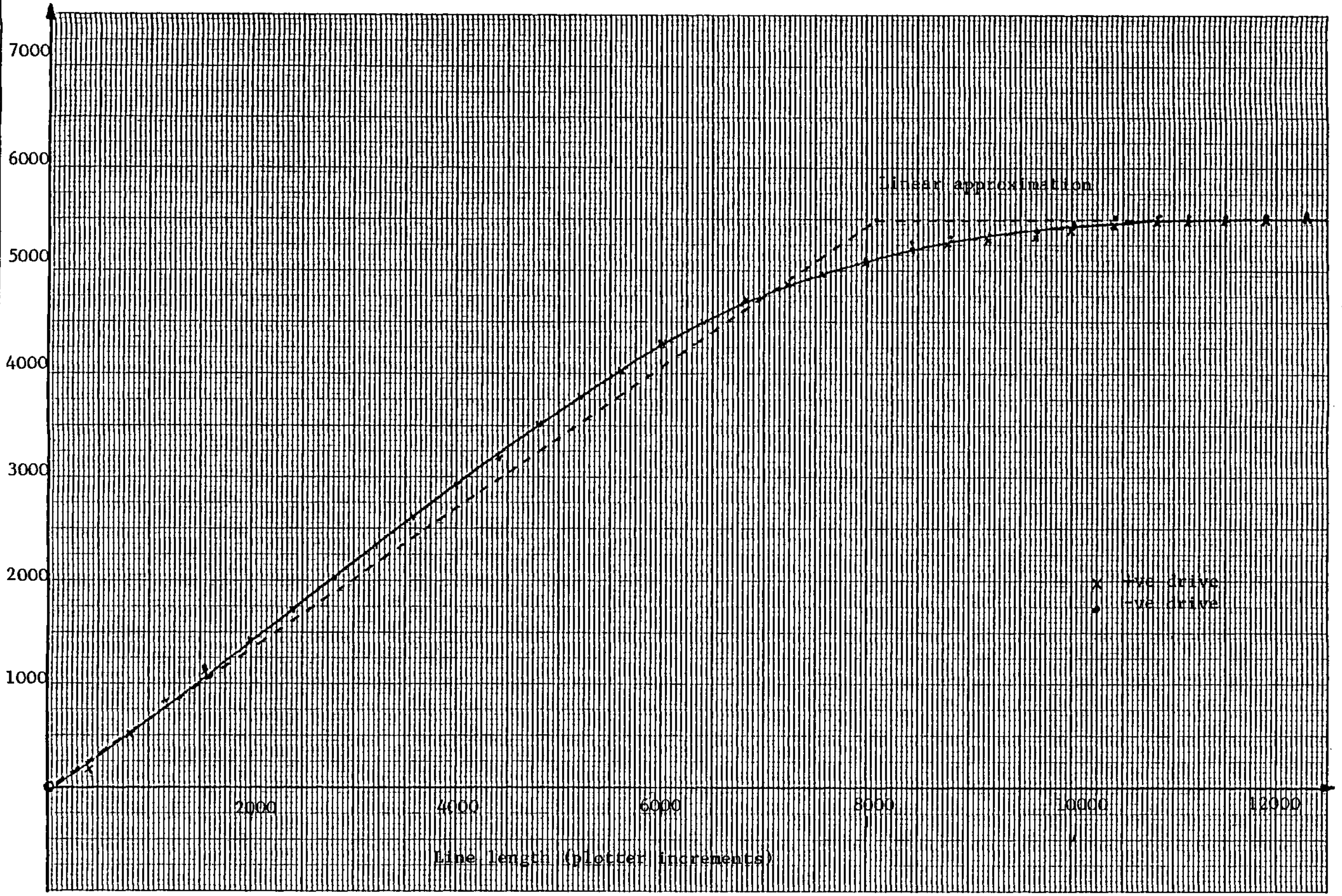


Figure A.5(i) Optimum decelleration length - Gantry movement only

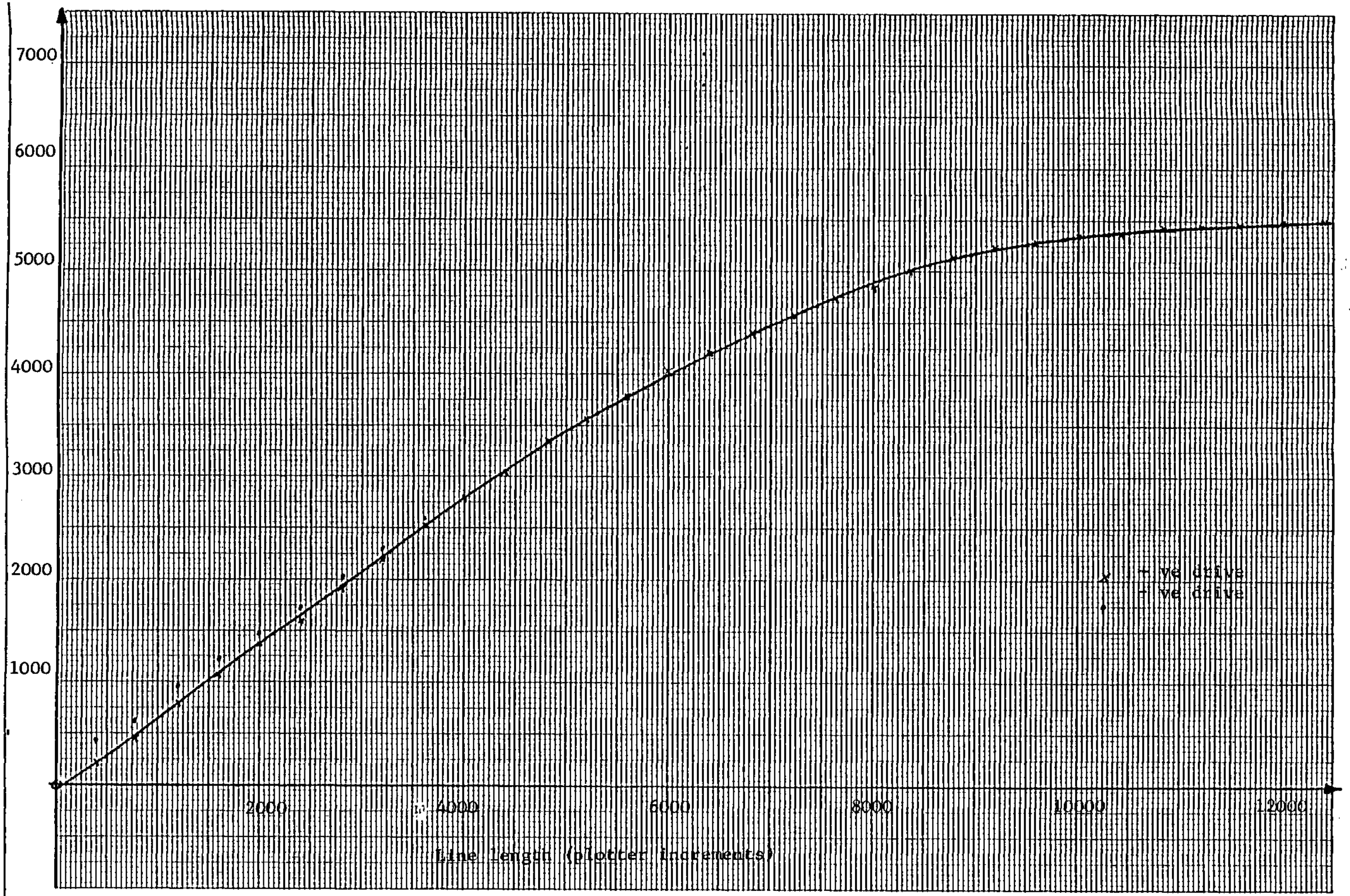


Figure A.5 (ii) Optimum decelleration length - Pen carriage movement only

At the start of every line the pen condition for the line was compared with the current condition. If different the new condition was sent and a short pause generated to give time for pen movement. It was found that without this delay the plotter could move several millimetres in the time taken for the pen to move up or down, thus result in incorrect starting conditions on some lines.

For ease of use several keyboard interrupts were incorporated as follows:

(a) Control C - Abort Plot

Enables the user to terminate a plot prematurely. The plot stops immediately the interrupt is issued, even in the middle of a line.

(b) Control W - Wait At End of Line

It is often necessary to stop the plotter temporarily during a plot to make small adjustments such as changing the pen. This interrupt instructs the plotter to wait at the end of the line currently being drawn.

(c) Control R - Recommence Plotting

Cancels the above effect.

An overall manual speed control was included since different types of pen and drawing surfaces require different plotting speeds. For example, Pentel ball pens will plot much faster than liquid ink pens. The user informs the software of the speed selected by appropriate settings of the switch register on the computer's front console.

A.3 <u>The Interface - Bit Allocation</u>

The plotter is interfaced to the computer through the CAMAC rack using the following SEN modules.

(a) <u>7061</u>

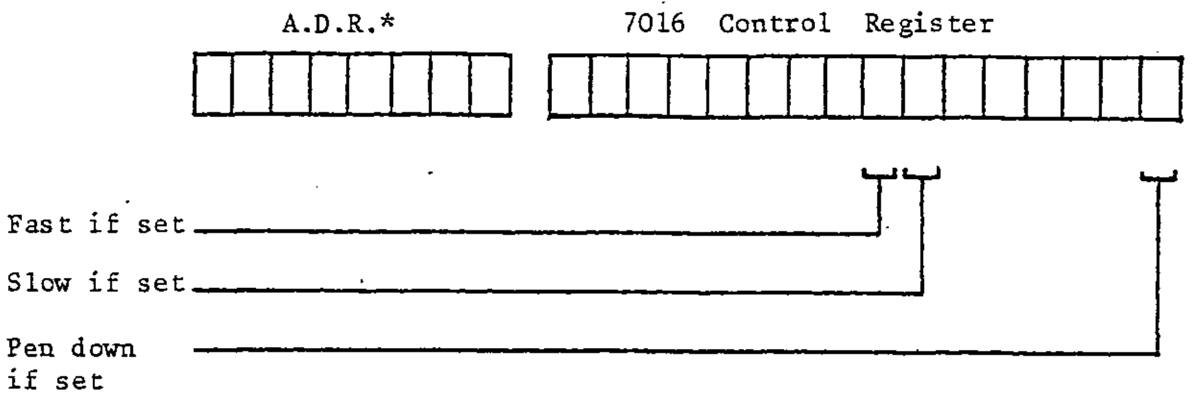
Controls stop/start and pen commands. See Figure A-6 for bit allocation.

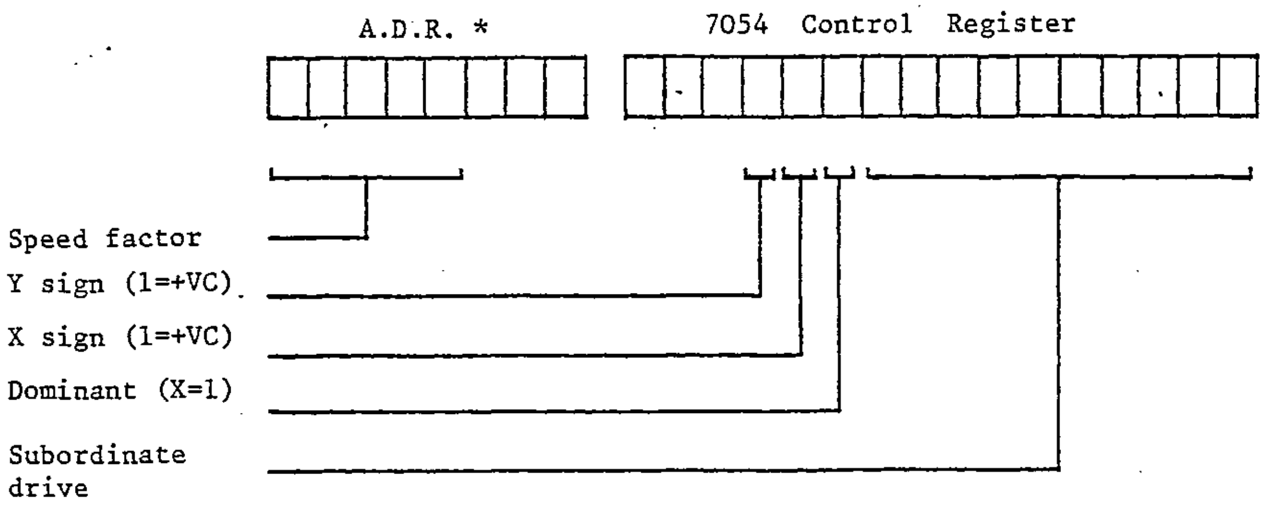
(b) <u>7054</u>

Controls remaining drive functions. See Figure A-6 for bit allocations.

(c) <u>21PE2019</u>

Dual incremental position encoder module. The plotter position is read from sub-addresses 0 and 1 for x and y respectively as 24 bit integers and converted into floating point numbers using the PDP 11/45 floating point hardware.





* Auxiliary data register

<u>Figure A.6 Interface Control Bit Allocations</u>

A.4 Suggestions for Further Development

The plotter has been seen to work well for general draughting purposes, although there are several areas where improvements may be possible.

(a) Arc Generation

The system demands that any arcs be considered as a series of very short straight lines, which the software does not distinguish from any other line. This results in rather jerky and slow movement around arcs since the plotter stops at the end of every line. It should be possible to regulate the drives to guide the pen around a smooth curve without stopping. An algorithm for this was tested by Ghassemi(19) with the original hardware and seen to work well, although through lack of time this was not implemented on the later system.

(b) Multiple Pen Capability

Although the software has the capability of handling up to 4 different pens the hardware currently offers only one pen. The original prototype unit which offered 2 pens was found to be unsatisfactory and was replaced by a single unit provided by CIL, who developed the plotter commercially. This appears to function somewhat better and it is hoped to obtain a multiple pen unit in the near future.

(c) Microprocessor Control

The recent rapid growth in the power of microprocessors accompanied by falling prices makes their use in peripheral devices such as the plotter more and more feasible both technically and economically.

The software developed for the plotter places a very heavy load on the computer since it is necessary to monitor the position of the pen carriage continually. This did not prove a serious problem on the single user DOS operating system but will be a severe burden if the plotter is to function on the RSX-11m multiuser system. It is anticipated that a new version of the plotter will be developed with an onboard microprocessor to perform this function and communicate with the PDP11 only to receive coordinates and return status. This task was considered too large to undertake as part of the current project and is being investigated by other students.

APPPENDIX B - AN AUTOMATIC DIGITISER

B.1 Introduction

With the advent of interactive computer aided design the digitising of large numbers of drawings by a manual tracing process has proved a considerable problem, being both time consuming and tedious. In many cases the process is performed purely for setting up data banks of parts already drawn manually, or the copying of existing drawings onto CAD equipment for archiving or analysis (as the result of this new technology being introduced into the drawing office). The present method of manual digitising using a pen or stylus to trace over the drawing effectively necessitates completely redrawing every detail, and a more efficient method has long been sought. An optical digitiser incorporating television technology may prove the solution to this problem. The system to be described was developed at Imperial College in association with Lloyds Register of shipping, and is a refinement of a prototype system also developed at the college. It is not intended to describe the earlier system in detail, but significant differences are noted as they occur.

The television camera processes the image as a dot matrix of 1024 x 575 points, and at each point records the darkness on a grey scale from 0 to 15. The hardware encodes this into 4 binary digits with 4 such samples being stored in one PDP 11 word and a complete picture in 147456 words. This data is presented to the computer through a CAMAC(7) interface and stored on disk by a suitable program for subsequent analysis.

B.2 The Hardware

The basic principle of standard 625 line interlaced television format is shown in Figure B-1. A frame is constructed from two interlaced fields of 312½ lines each. The scan starts at an even field in the top left hand corner, which the system uses as its origin for x,y coordinates. Each line is scanned from left to right under the control of the line timebase, the return to the next line being a rapid flyback. The first 22½ lines are blank, video data actually starting at the centre of line 23, and finishing at the end of line 310. At the end of a complete field of 312½ lines flyback to the top left hand corner occurs and the odd scan starts. In this pass video data starts at line 336 and ends at the centre of line 623. At the end of line 625 frame flyback again occurs to scan the next even field. In addition to the 25 blank lines inserted between video data in successive frames the first 1.5μs and last 1.5μs of each line are also blanked, these being included to allow for flyback and settling time of the horizontal and vertical time base amplifiers.

It was necessary to use a relatively low data transfer rate into the computer in order to make use of inexpensive components. The system operates by checking the amplitude of the video signal once per scan line, converting this to a 4 bit number and storing it in a shift register. The process starts from the left hand edge of the image (x=0) and during two field passes each video line is sampled to produce levels for that particular x. The sample point is then incremented to the next x position and the process repeated across the complete picture.

The current system is capable of 1024 horizontal locations, thus giving a resolution somewhat better than that obtainable from a good television camera (about 800). Each scan line takes 64us, of which 52us contain useful video data, and an integrated circuit from Ferranti designed for closed circuit television systems provides all the synchronisation and blanking pulses necessary. A block diagram of the equipment is shown in Figure B-2.

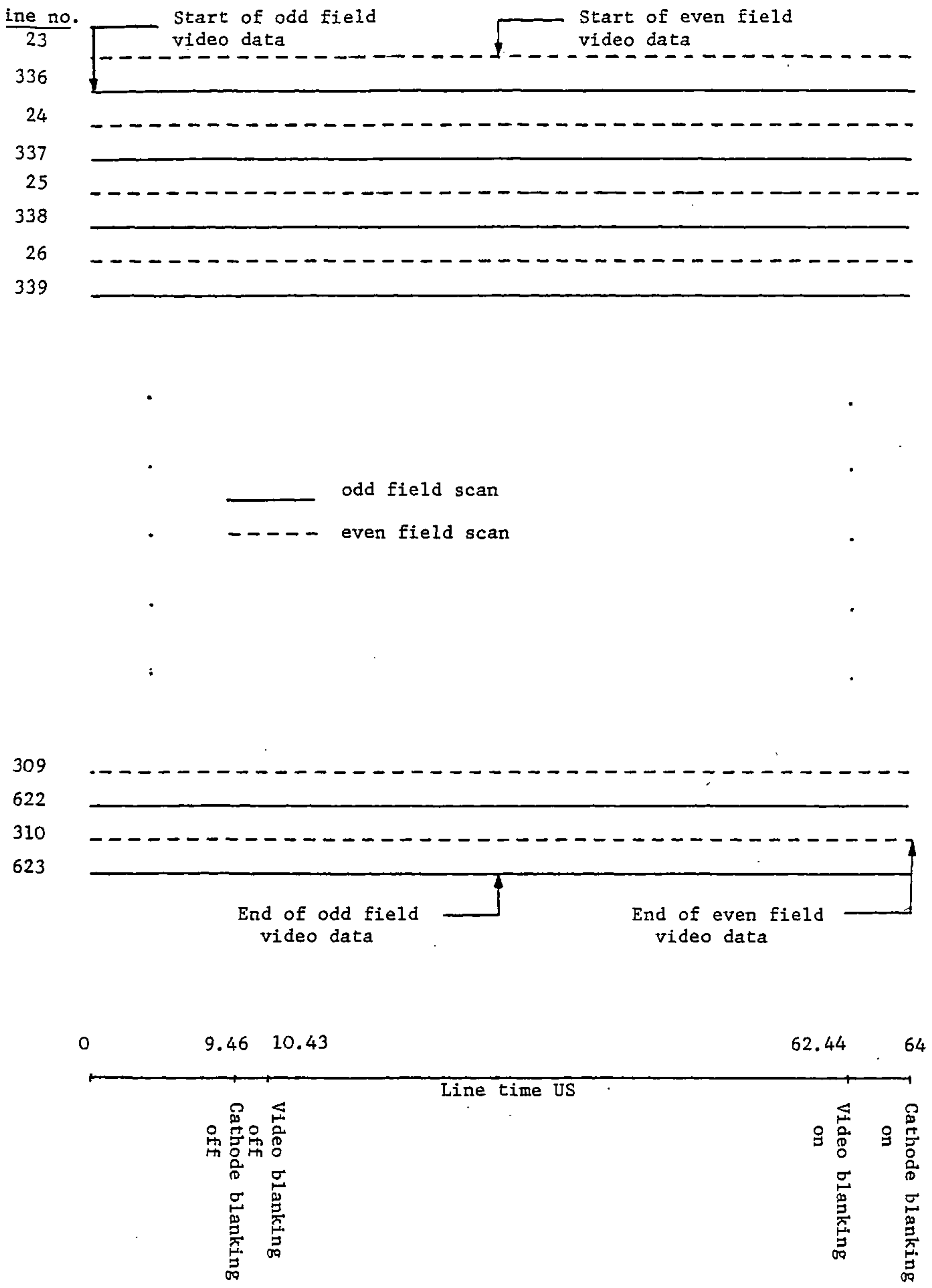


Figure B.1 Principle of Interlaced Television

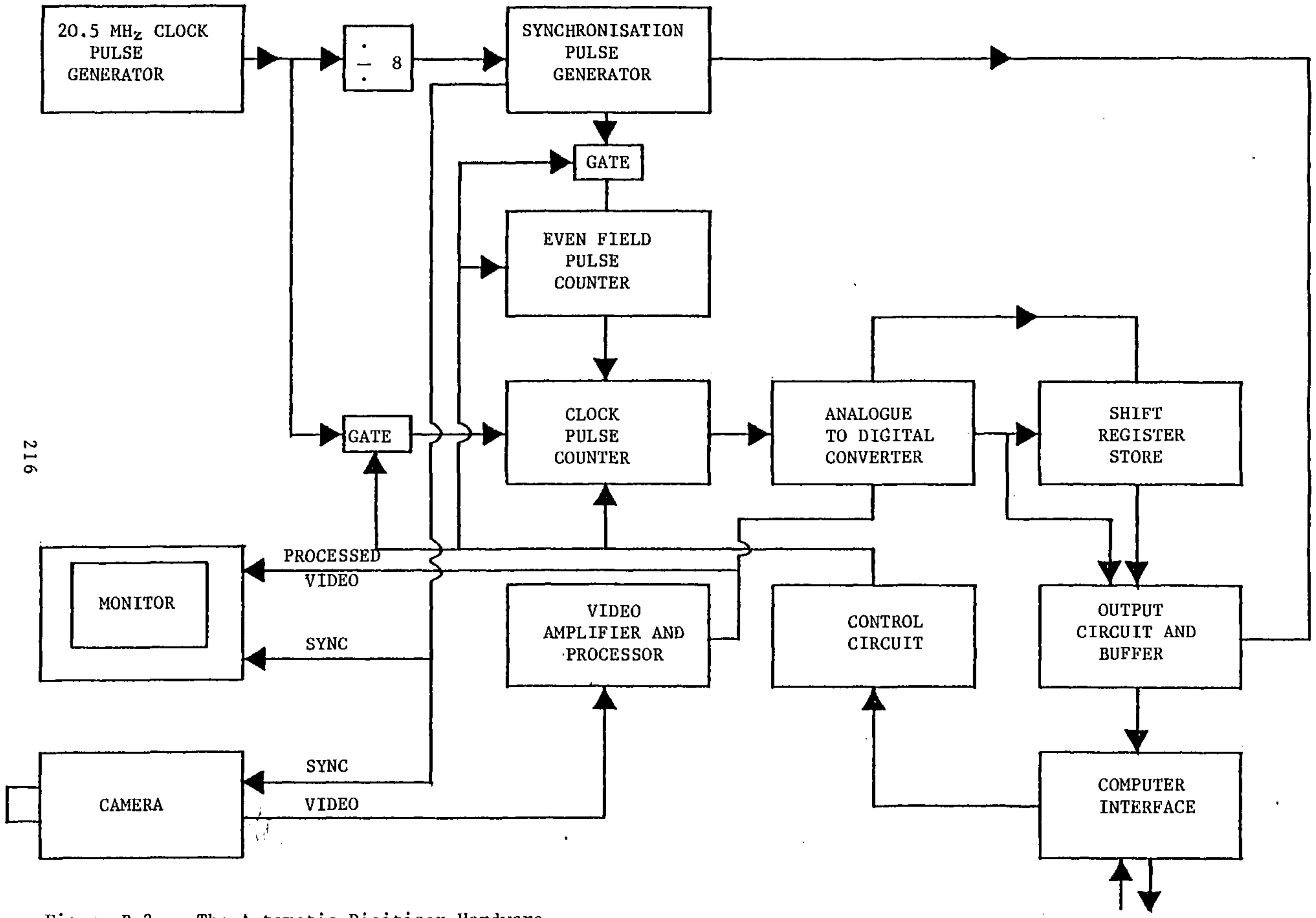


Figure B.2 The Automatic Digitiser Hardware

To commence digitising the computer sends a 'clear' signal which resets the even field pulse counter and a clock pulse counter to correspond to the upper left hand corner of the picture.

The computer then sends a 'continue' signal and sampling commences, the data being stored in a shift register. At the end of the even field this contains 288 values corresponding to the 288 useful lines of video in that field. On the odd field data is passed directly to the output circuits, interleaved with data being moved from the shift register, and presented to the computer as a series of 144 16 bit words, a read signal being sent to the computer each time a new word is available. This procedure is repeated for 128 x values, when the cycle stops to allow the computer to transfer the data to disk. When this is done another 'continue' signal is generated and a further 128 x values samples, this process being carried out 8 times to complete the picture. The entire process takes approximately 45 seconds, after which time the system halts. The software is responsible for detecting the breaks in data transmittal and the final area of data.

B.3 The Software

The development work carried out as part of this project was under the single user DOS operating system only. Due to hardware limitations outlined later it was impossible to function correctly under the RSX-11M multi-user system. It is understood that further development work by Lloyds Register will shortly enable the system to function correctly under the latter operating system.

The very large amounts of data presented by the hardware for each frame are impossible to buffer entirely in core on a computer the size of the PDP using the DOS operating system where addressing is limited to 32K words of core. It is therefore necessary to maintain a smaller but reuseable core buffer and write its contents to disk whenever it becomes full. This introduces a small time penalty since the video scan must be made to wait while data is being transferred, eight such operations being necessary to form a complete frame. To minimise the delay it was decided to ignore conventional file structures on the data disk and simply dump data in consecutive blocks. Using this method each drawing occupies 576 blocks (of 256 words each), thus enabling 8 'files' to be stored on one RK05 disk pack. Data consists of an (x,y) location and its associated grey level. In the prototype model it was necessary to decode the x,y position which were each returned in 10 bits, deinterlace the data and retrieve the grey level from the data using software. The required location in the buffer, which takes the form of a bitmap, was then calculated and the data stored. This was a time consuming process and some problems were encountered in keeping up with the rate of data output from the hardware. It was soon realised that, since the information is always presented in a specific order, it is possible to calculate the x,y coordinate from the previous position, providing no data has been missed. In the later model this was taken one stage further; the hardware decodes the position and grey level, packs four values into one word and presents the whole word to the computer, which has nothing more to do other than to place it in the next space in its core buffer and transfer to disk when the buffer is full. The order of storage in the bitmap is as shown overleaf.

In order to retrieve data from disk a subroutine was written which, given a file number and an x,y location, would return the grey level in the range Ø to 15.

The Tektronix 611 storage tube on the system can be used to view the data directly since it is basically a raster type device itself and it is therefore possible to send dots to correspond with locations in the database. It is not, however, possible to view the data in all grey levels simultaneously since the Tektronix is simply a black or white device. The most useful method of viewing was found to be to set a threshold above which all data is displayed.

In raster form the data is of little use for anything except direct viewing as above, and to be of value to the engineer it must be converted into a more familiar form, i.e. into lines and points. This task is currently being undertaken by Lloyds Register on their own CAD equipment with considerable success.

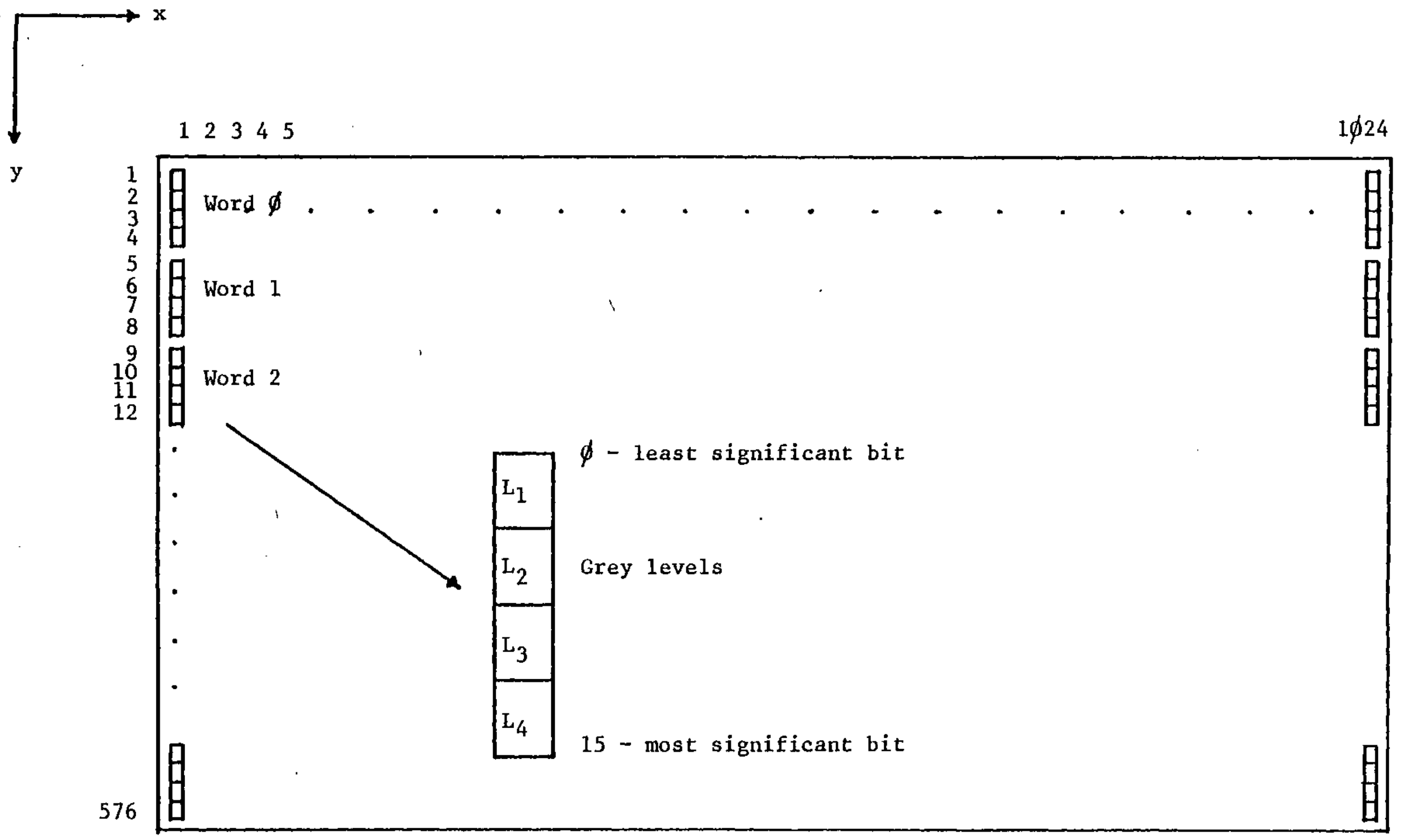
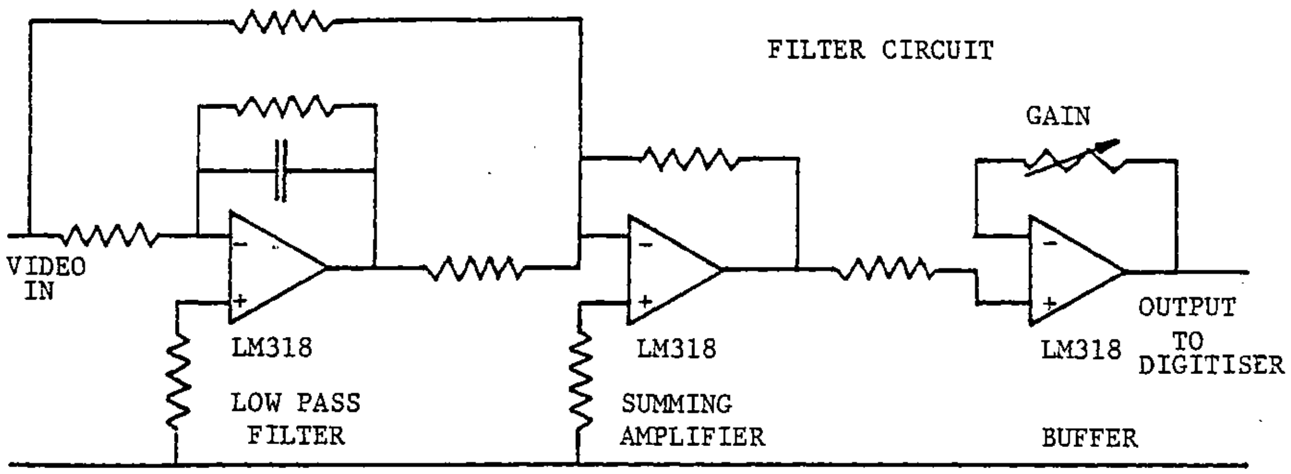


Figure B.3 Bitmap Storage Layout

B.4 Future Development

An imminent development is to be the installation of a DMA (direct memory access) interface to replace the current CAMAC interface. With this arrangement the hardware will be responsible for placing the data in the core buffer, the only responsibility taken by the program being to allocate the buffer and copy from it to disk. This is of particular importance if the system is to function satisfactorily under the RSX-11M operating system since the rate of data transmission means there is serious danger of some being missed, unless the processor is dedicated to this task, which cannot be practically guaranteed with a multi-user executive.

Tests have shown the system to work well if the artwork is of very high contrast, i.e. dense black lines on a uniform white background. Unfortunately, most drawings do not come up to this standard, lines being poorly defined and backgrounds being varying shades of grey or even blue depending on the printing or reproduction technique. In this case it was found difficult to process the data using software, and some attempts at processing the video signal in hardware were made. To this end the simple circuit of Figure B-4 was tried. This consists of 3 operational amplifiers, the first of which acts as a low pass filter and removes all rapid transitions from the signal. This is then subtracted from the original signal in the second, summing, amplifier, resulting in a signal in which the rapid transitions are enhanced and the background variations cancelled out. The final amplifier acts simply as a buffer and allows gain control. This effect is shown diagramatically in Figure B-4. The system obviously performs better on narrow lines and has proved reasonably effective in improving the quality of data digitised from many engineering drawings which basically consist of such lines. Further work is in progress to optimise this circuit and it is hoped to develop a system to digitise drawings in which 'thick' lines play a significant part.



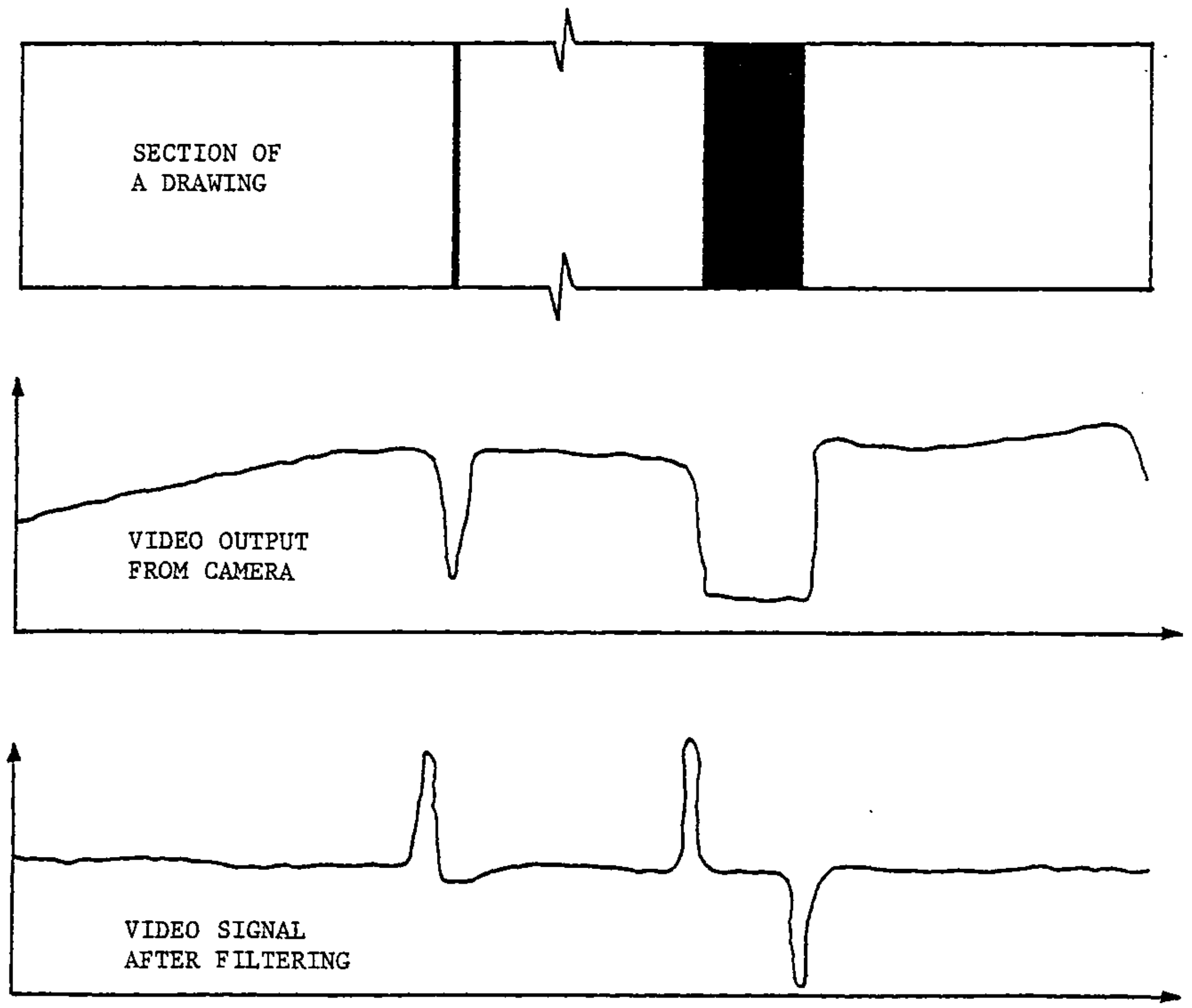


Figure B.4 Enhancing the Video Signal