THE DESIGN ASPECTS OF A MICROCOMPUTER CONTROLLED NC MACHINE SYSTEM

by

Peter-Benson Uchechukwu Achi (B.Sc., M.Sc.)

A thesis submitted for the degree of Doctor of Philosophy of the University of Aston in Birmingham

February 1981

THE DESIGN ASPECTS OF A MICROCOMPUTER CONTROLLED NC MACHINE SYSTEM

Peter Uchechukwu Achi

A thesis submitted for the degree of Ph.D of the University of Aston in Birmingham. February 1981.

SUMMARY

Computer numerically controlled (CNC) machine tools are proving their value by helping their users to remain competitive. The advent of microprocessors and thus microcomputers should have encouraged the smaller firms using numerically controlled (NC) machine tools to join the CNC trend but they are often prevented from doing so mainly owing to the cost of the special engineering involved in a dedicated microprocessor machine system. One of the ways by which CNC can be introduced would be to acquire a microcomputer the time of which could be shared between the control of an existing NC machine via 'large scale integrated (LSI) units and possibly the execution of some management functions. This thesis describes the design of an economic control system for a milling machine in $2\frac{1}{2}$ (or 3) axes with stepping motors controlled from a microcomputer in open loop. On the existing machine the X-axis has an hydraulic motor and the Y- (and Z-) axes have hydraulic cylinders and rams. Each axis is actuated by a stepping motor in a servo loop. The axis servos thus formed are shown to be linear, the velocity following error varying proportionally with the stepping rates. The whole CNC system can be built in-house because the high cost of special engineering can be considerably reduced by the use of programmable LSI's in the microcomputer interface which output interpolation pulses to the stepping motors for accelerations, decelerations and programmed feedrates. They count the pulses and provide interrupts to the 16-bit microcomputer. Machines with direct stepping motor drives can also use these interfaces. The microcomputer operating system is generated on floppy diskettes (DOS) which also contains the control file. Assembly language subroutine calls are included in the main program which is in FORTRAN. The use of a high level language for the microcomputer enables the programming of the milling machine to be by manual data inputs (MDI) in computer question-and-answer mode. Chapters 6 and 7 are devoted to recent developments which are evidence of the interest in NC LSI's and are shown to be complementary to this study.

Key Words: NC
Stepping Motors
Hydraulic Servo
Microcomputer
LSI

The Design Aspects of a Microcomputer Controlled NC Machine System

P.B.U. Achi

A thesis submitted for the degree of Ph.D
of the University of Aston in Birmingham.

February 1981

DECLARATION

I, the undersigned, hereby declare that no part of the work described in this thesis was done in collaboration, and that the work has not been submitted for any other award.

P.B.U. Achi

List of Abbreviations*

A/D	Analogue to Digital
BRM	Binary rate multiple
CNC	Computer numerical control
CPU	Central processing unit
D/A	Digital to analogue
DDA	Digital differential analyser
DOS	Disc operating system
EROM	Erasable read-only memory
"HPESYS"	The microcomputer controlled NC system (using the "HPE" machine)
H/W	Hardware
IC	Integrated circuit
I/O	Input/Output
KPPS	Kilo pulses per second
LSI	Large scale integrated circuit
M/C	Machine
MDI	Manual data input
MPS	Multi-processor system
MR	Machine resolution
NC	Numerical control
PIA	Peripheral interface adapter
PPI	Programmable peripheral interface
PPS	Pulses per second
PRM	Pulse rate multiplier
PROM	Programmable read-only memory
ROM	Read-only memory
SM	Stepping motor
S/W	Software
μP	Microprocessor
μC	Microcomputer
VIA	Versatile interface adapter

* Electronic terms and appropriate descriptive names are used wherever necessary in this study.

THE DESIGN ASPECTS OF A MICROCOMPUTER CONTROLLED NC MACHINE SYSTEM

CONTENTS

CHAPTER 1

INTRODUCTION

Numerically controlled machines first became commercially available in the early 1950's in the United States. Since then NC Systems have been concerned with the control of the sequence of operations, the cutting speed, the axis movements, the points in the process at which various operations take place and the axis feedrates and the slide positions, all of which determine the final shape of the component. In full NC systems these variables are automatically controlled after the machine has been fed with numerical instructions by way of an input medium. Input of these media evolved from the punched card to the magnetic tape and this would be described as the second generation conventional NC system. A typical example is the Ferranti MK IV system of the early 1960's. One of its users for many years was Stevens (12) who recalled that it was 'slow business' because the profile data had to be written, punched on an old 'FED-1 Code 5-channel' tape and posted to Ferranti in Scotland where a computer was used to generate the final part programs on magnetic tapes. The problems that this involved led to the replacement of the magnetic tapes with the one-inch 8-hole paper tapes. By the early 1970's they had become widely used all over the world. The paper tapes were definite improvements on the magnetic types because they were cheaper and more significantly, they didn't require the computer (still an expensive unit in the late sixties) for their preparation. The hardwired controller has to read a block of data from the tape, decode it and execute the instructions before it can fetch the next block.

The low speed of the NC tape reader was a definite limitation and the lack of flexibility of the hardwired controller requiring the programmer to obey a defined order of data coding was another. Often the skill required of the programmer was not just that of a production technologist but necessarily that of a mathematician. All these combined to make the paper tape NC only marginally economic for small batches which often constitute the operation region of smaller firms. The delay involved in tape proving and editing, and the extra cost of the limited number of skilled programmers presented a challenge even to the bigger firms to recover their outlay in as short a time as possible. Often the cost of the controller hardware was comparable with the cost of the machine it was to control. The lack of flexibility in this controller encouraged the bigger firms to introduce the computer as an integral part of a numerically controlled machine system. The Sundstrand Omnicontrol system based on IBM 7 series computer was installed by General Motors in 1968 and became the first of such industrial systems. It was an expensive system and could be afforded only by the big companies.

By the early 1970's the cost of computers had fallen considerably with the evolution of the mini computer. Control system manufacturers could then justify the economy of integrating the minicomputer in their range of NC systems. Computer numerical control (CNC) had evolved and the fixed hardware machine tool controller was virtually out of date.

The updating of the existing control systems came by way of BTR (Behind the Tape Reader). In such systems the existing controllers are retained to carry out interpolation, feed and the usual machine

related and miscellaneous functions while the minicomputer handles the part program input, interpretation and distribution NC functions. This system was uneconomic if dedicated to one machine alone. Therefore to make it cost effective one minicomputer was interfaced to several NC machines. Further development saw most of the NC functions including interpolation taken over by the minicomputer but the cost of hardware and software for this direct numerical control (DNC) was still too high for the average company. The introduction of the microprocessor and thus the birth of the microcomputer later in the 1970's enabled the cost of electronic hardware to drop dramatically. The medium-sized companies could then afford CNC and manufacturers of NC systems were encouraged to invest in microprocessor-based products.

The microprocessor has the arithmetic ability to handle considerable quantities of data in real time, more or less like an ordinary computer. It has the extra advantage of drastically reduced physical dimensions and costs. Given good hardware and software support, the microprocessor has the flexibility of the ordinary minicomputer in NC systems. With the low hardware cost, microprocessor-based controls have the potential to keep their users competitive. Some microprocessor based control system packages for specific NC machines are being made by the NC industries. Usually each is dedicated to the control of a machine which is often too expensive thanks to the highly specialised engineering required now and in the foreseeable future to implement micro-based systems. Page (9) shares this view when he said recently that the very high level of integration of microprocessors and their related components being inherently complex, requires considerable expertise for implementation even for those well versed in electronic technology. The same is even more true of the software which usually constitutes the principal part of the total micro-system costs.

The result of all these considerations is that even the medium-sized companies are fighting shy of micro-based CNC. The smaller firms are again being left out altogether just as they were during the early evolution of NC machines. It is evident that the necessary stimulus for further drop in the cost of micro-systems is the inclusion of the lower rungs of business in their use. For the machine tool user this involvement has to be as economic as possible. This study gives a solution by way of microcomputer controlled NC machine systems in which the NC functions are handled with a microcomputer which is supported in its interpretation and feedrate functions by a cheap programmable LSI in its interface. This LSI will then be responsible for generating interpolated feed pulses to drive stepping motors which can be configured in various ways to position the machine axes. The machine tool used in this study is of the three axes milling type made by High Precision Equipment (HPE) Company. The initial control system was of the problematic Ferranti MK IV type mentioned earlier (12).

The milling machine is driven in the X-axis by a hydraulic motor and in the Y- and Z- axes by hydraulic rams. Because of the magnetic tape problems associated with the earlier Ferranti system, (12) this machine had not been much used and formed a good basis for the new control system philosophy described in this thesis. Although small electric stepping motors are used as the control actuators in this study, high power direct electric stepping (or DC motors) could be driven from the micro-computer with this design. With the ever increasing processing speeds and memory integration associated with microprocessors and the availability of high level languages, such systems could incorporate management functions. This makes it even more cost effective for any firm. With the evolution of the low cost microcomputer interest in

the compatible and equally low cost stepping motor CNC systems is likely to increase. Some of the papers in the literature survey provide evidence of this.

CHAPTER 2

LITERATURE SURVEY AND ANALYSIS

2.0 The Advent of minicomputer numerical control for machine tool systems

By the early 1970's the price of minicomputers had fallen sufficiently to justify the design of an NC system around such equipment. This introduced flexibility in the control of the basic NC functions and high speed access storage by way of semi-conductor memories, all of which were lacking in the hardwired controller.

In DNC systems one minicomputer usually controls more than one NC machine and this makes the system more cost-effective. The tape reading problems are solved by the allocation of memory buffers to service the machine interface so that more than one block of data can be read ahead of the present machining operation. Feedrates can now be accurately and continuously monitored, thus eliminating the dwell marks attendant on piece-meal data block transfers especially noticeable during contouring operations. The miscellaneous functions, like on/off collant, tool changes and offsets are rapidly controlled digitally from the computer. Part program editing can now be done on the machine and the paper tape can be restricted to program loading during the initialisation of a batch and provided enough memory is available (2.5m of tape = 1 k store) the whole part program could be resident in memory for the particular batch and this includes the sub-routines necessary for canned cycles.

Tipton and Roberts (14) describe the M.T.I.R.A 6-axes CNC system which is based on Digital Equipment Corporation's PDP 11/05 minicomputer, a 16-bit word machine. With 4K core store, the full NC functions are performed through a General Purpose Interface of sixteen 16-bit Read-only and Write-only registers which are able to retain information (in latches) until updated. By way of subroutines on input tapes the software is modularised, thus making it general purpose in nature (bearing other machine tools in mind). The tape reader is interrupt driven to load six blocks at a time into the core to keep it continuously refreshed. At a data sampling frequency of 200HZ the real-time clock interrupts invoke the servo servicing routines which control the DC motor drives in a feedback loop closed within the minicomputer. Circular interpolation is achieved by approximating arcs to chords and within the stated sampling frequency the incremental size of the chords would determine the feedrate.

A CNC system using the Hewlett Packard 2100 minicomputer is described by Carter (15). The minicomputer has a cycle time of 1 μsec and with an 8K core the NC drive servicing routine is interrupt driven at periods of 10msec (by a real-time clock) to provide interpolation, servo and other machine dependent controls (like job clamping). As in the M.T.I.R.A. system the feedrate control is tied up with the real-time clock interrupt periods. The system operates in a loop closed inside the Hewlett Packard Computer. Carter correctly pointed out that the general weakness of general purpose minicomputers is that they are word-oriented so that the ability to manipulate and address individual bits for single bit machine-dependent operations take more memory than they otherwise would. This increases the cost of memory for a given application, especially in direct memory access systems.

Maclean, Bruce and Davies (16) describe a modular CNC system based on MINIC 1 minicomputer, an 8-bit word machine 8K core of which is required to implement the NEL $2\frac{1}{2}$D contouring system. They pointed out that the vertical integration modular structure of the hardware and software units (from control cabinet, tape reader, control programs to the servo interface) makes the system suitable for future retrofit at any level. The DC motor drives are within the loops closed within the minicomputer. A hardware switching logic controlled by software interpolation generates pulses to drive the motors with a resolution of 2.5 μm.

The chosen position sampling period, fixed by interrupt generating down counters gives the feedrate control.

Crossley and McCartney (17) in tracing the historical development of DNC from its inception in 1968 to date gave one of the most lucid explanations of 'Deep' DNC, BTR-Level DNC and time-sharing micro-BTR systems. Figures 2.1, 2.2, 2.3 and 2.4 show that the basic difference between these is the position of the DNC minicomputer in relation to the interpolation function.

They cited the Sundstrand Omnicontrol system based on IBM 360 as typical of a Deep DNC system. This system is the earliest installed and involves two computers, one a remote general purpose main frame computer and the other the actual DNC computer. The main-frame computer is accessed by a time-sharing link to generate the part programs. The DNC computer handles program storage, its distribution, and machine control functions. The machine control function is taken care of by the interpolator which replaces the conventional NC controller.

The pulses generated by the interpolator drive the multi-axes at a feedrate dependent on the rate of pulse generation. The memory requirement for part programs is usually 1K per NC machine connected and the Omnicontrol could take up to 15 machines.

They rightly described the BTR-level DNC as an answer to the high cost of Deep DNC. Instead of scrapping the existing conventional NC controller it is retained (fig. 2.4) to take care of the usual NC functions including interpolation. The time-shared mainframe computer could be eliminated and a local part program input device (paper tape) included. The BTR (Behind-the-Tape-Reader) unit is added to accept data from the computer and transform this to the format suitable for a given NC controller. The buffer between the mini computer and the BTR 'black box' ensures that data is continuously available for the machine controller during contouring operations. Eight parallel data channels and four control lines are used for data transfer and peripheral device control respectively, in the 'INDEX BTR' system which is based on IBM series 7 computer. It is a 16-bit minicomputer and required 10K memory for part programs. They correctly observed that although more companies could afford systems like the 'INDEX', the cost of software was too high in the early 1970's when they were in use.

The third system discussed by Crossley and McCartney (17) is the 'Salford/Plessey' Multi-task BTR DNC system. It is similar to the BTR-level DNC already described except that it offers more peripheral controls like graphics (visual display units) and a distinct operator terminal. It was based on the DEC PDP-11/45 which controls the machine through serial asynchronous line interfaces. Five NC machines could be controlled using hard disc-based software of FORTRAN IV and

Macro-Assembler. When it is interrupt-driven only 20% of computer time is utilised.

Finally they described a BTR system that would incorporate a microprocessor (fig. 2.5.1). The system (still under development) would include a standard minicomputer which only communicates with the microprocessor on a time-sharing basis. The data buffer is now transfered to the microprocessor which can communicate with several I/O devices through its own data bus. This microcomputer which can be used as a 'stand alone' unit determines the performance of the NC machine which it controls via a BTR black box which in turn communicates with the existing NC machine controller for interpolation, feedrate control, tool offsets and other machine-related functions. This system would give room for future expansion resulting in a modular multi-microcomputer based BTR NC system.

The author observes that it could be more economic to eliminate the BTR concept altogether and configure this multi-system described by Crossley and McCartney with a collection of microcomputer controlled NC machine systems each of which could resemble the subject of this study. By time-sharing the resources (memory and computing power) of a minicomputer the memory and processing requirements from each microcomputer system would be reduced. Operator terminals would also exist at each microcomputer offering the choice of using the facilities of the minicomputer or dispensing with it.

Shumsheruddin (18) describes a NOVA 2/4 minicomputer-controlled jig grinder. The X-Y positioning table is driven by two electric stepping motors. The peripherals include paper tape reader, teletype,

manual data input keyboard, a real time clock and an interpolator. The coarse interpolation is performed by the minicomputer and a digital-differential-analyser-based external interpolator completes the fine interpolation to an accuracy of 0.001mm. For linear interpolation the path is divided into equal increments the maximum size of which is within the bit-size range of the fine interpolator. The feedrate in this case would be derived from the size of the increments and the sampling period. For circular interpolation a look-up table which dispenses with sines and cosines is used. Although this takes up much extra memory, the author considers it a good choice because it frees the processor from any time-consuming real-time calculations thus making future system expansion a feasible proposition. No mention was made of the stepping motor controls in the system except that the interpolated axes increments are passed through A/D convertors before being transmitted to the axes servos. He reckoned that this system would be more cost effective than even a microprocessor-based similar system from the point of view of the extensive software support existing for minicomputers and the minimal external hardware needed to expand the system to multi-machine control.

The computer numerical control systems so far discussed have one thing in common: the minicomputer as the executor of input and output NC functions. There are several other similar systems described in journals and manufacturer's literature up to the late 1970's. Those described, were chosen because they can be seen as typical examples of minicomputer systems up to the emergence of the microprocessor. By the time the second generation 8-bit microprocessors became commercially available many of the basic problems of building an NC system around a general-purpose minicomputer had been solved and proved in practice.

Minicomputers were capable of rather more data-processing functions than were actually needed for basic machine tool NC functions. They were economic only for more expensive multi-axis applications like machining centres. The cost effectiveness required of the less complex machine tool applications were therefore provided first by the 8-bit microprocessors and now by the 16-bit and bit-slice microcomputers. For an understanding of the capabilities and limitations of these new CNC systems it is necessary to make a brief review of the on-going developments in microprocessor architecture.

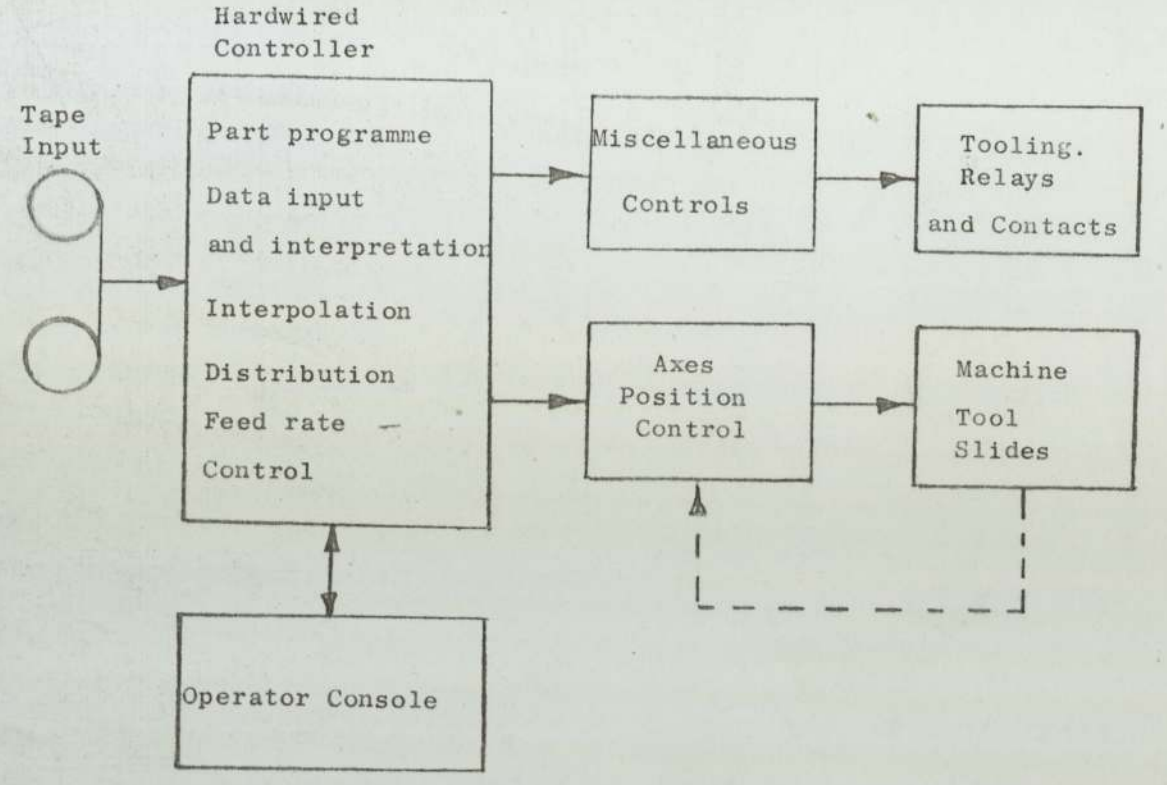


FIG 2.0 Conventional NC System

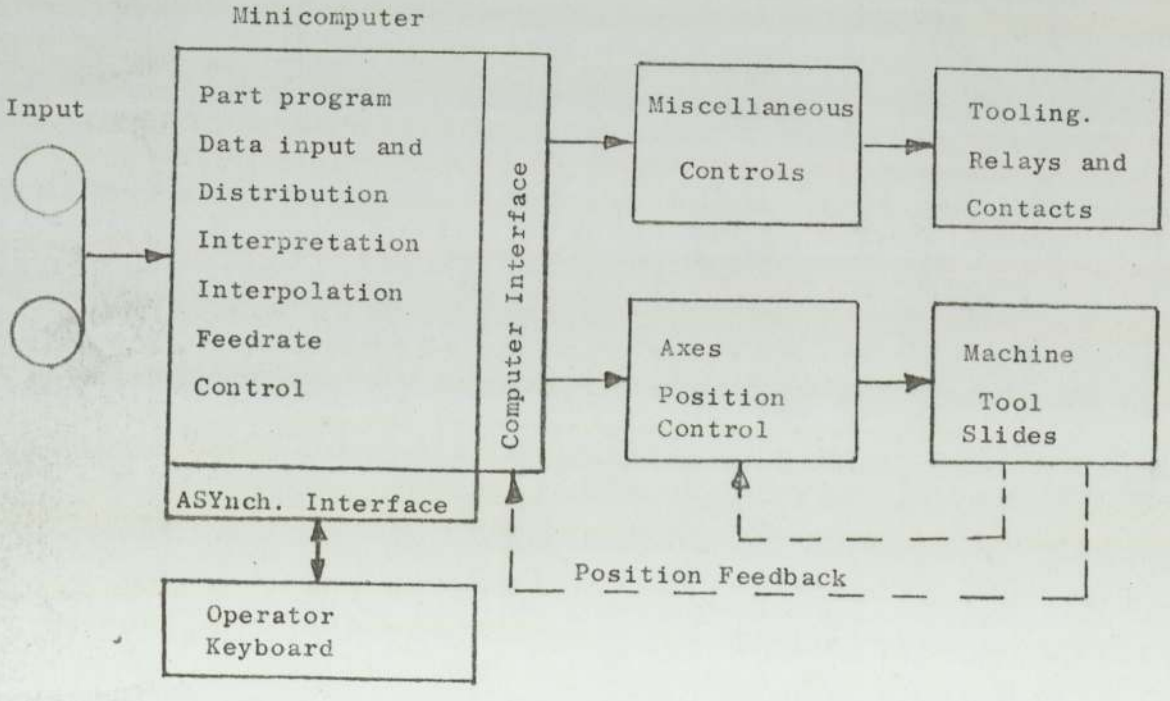


FIG 2.1 Computer Numerical Control System

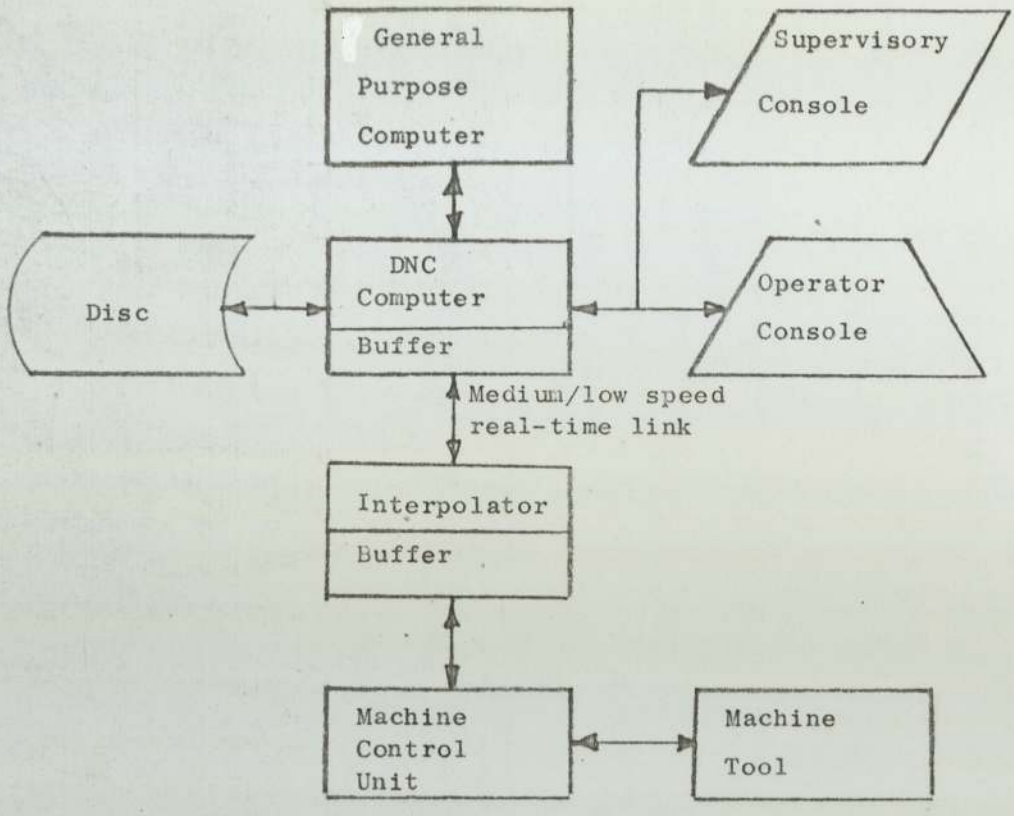


FIG 2.2 DEEP DNC SYSTEM

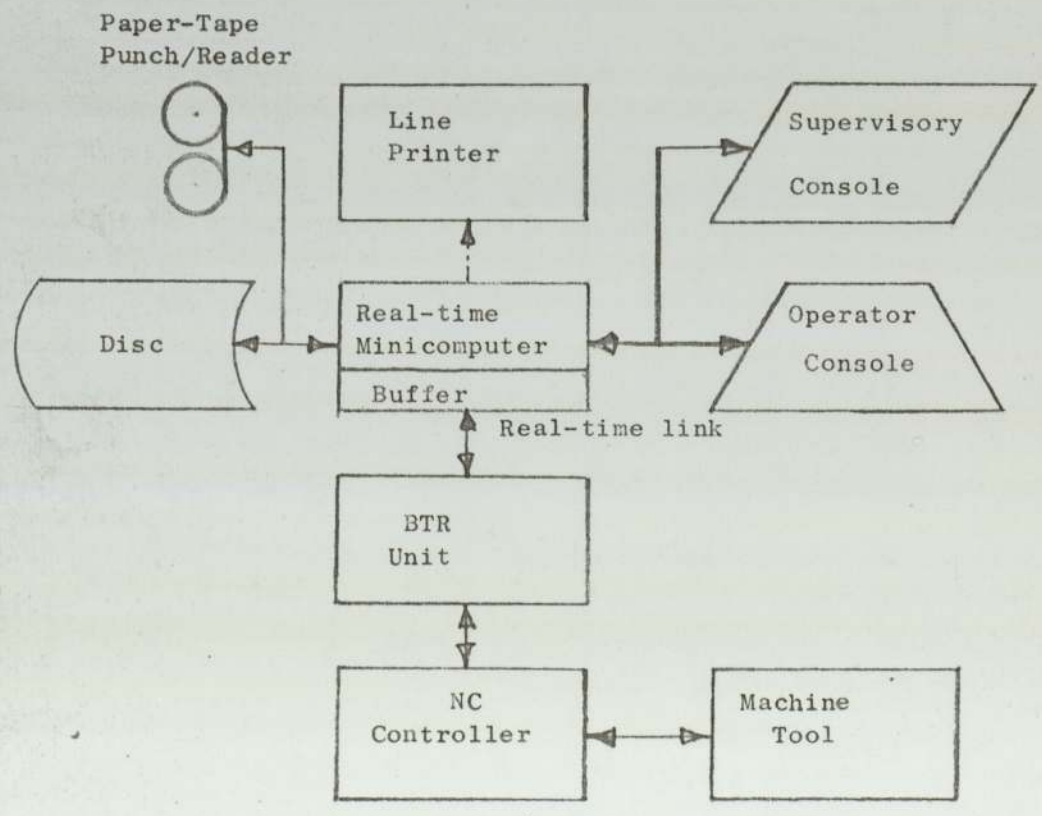


FIG. 2.3 BTR - Level DNC

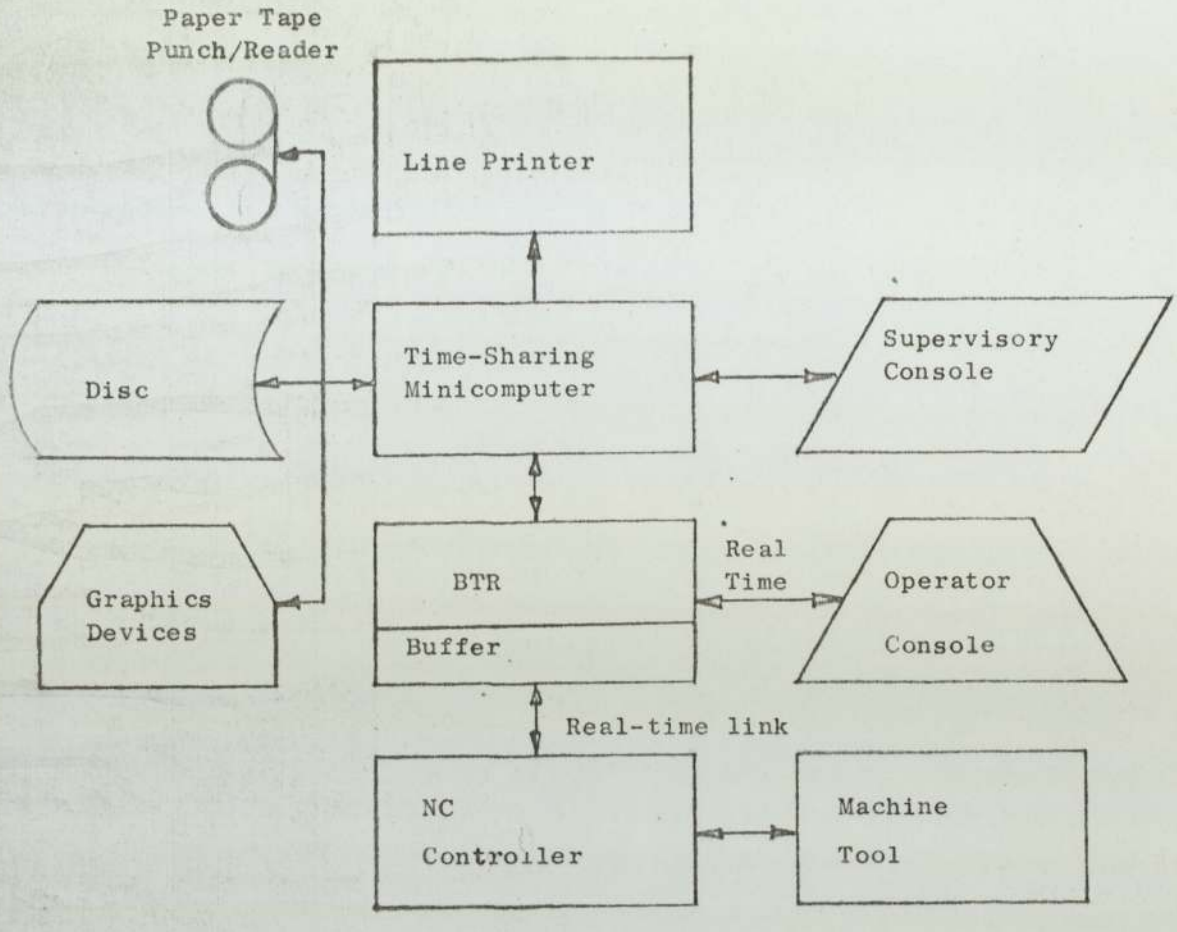


FIG 2.4 Time-Sharing Micro-BTR System

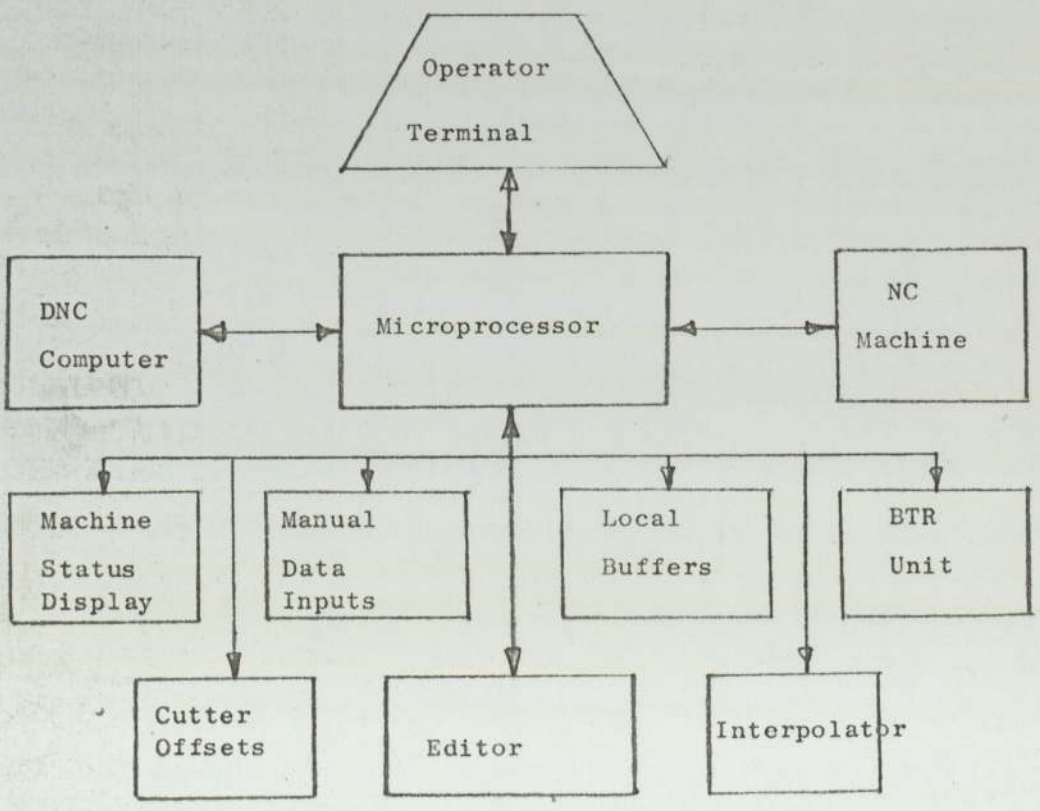


FIG. 2.5.1 MICROPROCESSOR-BASED BTR-LEVEL DNC SYSTEM

2.1 The Introduction of microcomputers

The microprocessor is a programmable LSI which has arithmetic and logical abilities. When supplied with input/output peripheral elements and internal and/or external memory it becomes a microcomputer.

2.1.1 Classes of microprocessors

The Electrical Research Association (19) describes three constraints in the design of a microprocessor, namely, the number of logic elements (gates) which can be integrated in a single μP chip, the maximum number of pins which one chip can accommodate and the optimum speed at which the gates can be switched during arithmetic and logical operations. These constraints influenced the evolution of μP's within the context of their chip count (for system implementation) and the bit lengths (4 bits or its multiples). The single chip μP's (fig. 2.5) differ from the rest on account of the absence of internal memory as an integral part of the μP. The necessary RAM, PROM or ROM have to be supplied as external chips. The more common examples of this group are the Intel 8080 and the Motorola 6800. With the exception of Intel 8085, the subsequent generations of these μP's and the single chip microcomputers contain some internal memory (fig. 2.6).

When μP's are supplied with the necessary system peripherals like I/O lines and clock generators they become microcomputers. Traditionally μP's and μC's are classified according to their bit lengths. The earliest μP's were the 4-bit types like Intel 4004 which was a pioneer μP. It came as a 16-pin chip and required dual power supplies (-10V, + 5V) and could obey 45 instructions. It required a minimum of three chips, the

4004μP, the Intel 4001 8-bit Rom chip and the clock generator to be effective as a simple system. With separate address and data buses, the 4-bit μP requires to multiplex the 12-bit ROM address lines onto its 4-bit addressing logic. This is a complex process and results in slow memory cycle times of 10.8 μsec and even slower instruction execution times of about 80μS. The Rockwell PPs-4 system based on PN10660 μP has a similar performance but its cycle time was quoted as 5 μS.

These requirements show clearly the effect of bit sizes on μP performances and why subsequent generations of μP's have increased bit-lengths to 8 and 16; or even 12, 24 and 32. The Intel 8080 introduced in 1973 was an 8-bit device and a marked improvement of the earlier versions but even so, it requires three power supplies (±5V and +12V) but can obey 78 instructions. The Motorola MC 6800 is very much similar to 8080, except that it requires only one +5V power supply and is capable of accepting 72 instructions. Both use 2 MHZ clocks.

Intel 8048 and 8748 8-bit μC's (fig. 2.6) which appeared later are not compatible with the earlier 8080 μP, ERA (19). Other 8-bit μP's include Zilog Z80, Fairchild F8, National Semiconductor SC/MP and Signetics 2650.

The 8-bit microprocessors could perform only 8-bit arithmetic operations fast enough for high speed real-time applications. Their accompanying 8-bit registers (e.g. for instruction addresses) also have a limited addressing range. The 16-bit μPs were therefore introduced as major improvements on the 8-bit devices. They have faster arithmetic capabilities and larger address space (memory). Also they offer better data resolutions when applied to control systems.

The Texas Instrument 9900, the Ferranti F100-L, the Intel 8086 and the earlier Zilog Z8000 are typical 16-bit microprocessors and described by ERA (19) as the principal market competitors.

Speaking of the Ferranti F100-L, Dixon (21) noted that one of the necessary requirements for a μP to be able to interact effectively with LSI's and other peripheral interfacing is that an asynchronons data/address I/O bus must be adopted. This enables LSI's of speeds different from that of the memory to use the same bus. He mentioned that certain designers opt for the off-chip implementation of maximum possible processor logic to include multi-Accumulator sets and this conflicts with the desire for asynchronism. The reason is that for such a system to be effective it requires a synchronous memory bus with separate data and address lines. The author notes that the micro NOVA microprocessor MN601 (described in chapter 4) is typical of this problematic synchronous operation. It was introduced in 1976, a year ahead of F100-1 as a 16-bit device and the earliest microprocessor venture by Data General Corporation. The logic design around MN601 resembles that of the earlier NOVA line of minicomputers. It has registers (including four accumulators) implemented within the μP and thus leaving little space for on-chip integration. Popa (20) rightly singled out the micro NOVA as a μP which does not float its buses for use by peripherals for direct memory accesses. Forth and Kidd (83) described a complex system which uses the Intel 8085 to configure data and sense registers (D.S.R.) to overcome the problem of timing arising from the synchronous construction of NOVA processors. The author observes that this system may be justified for NOVA mini-computers (as discussed by Forth & Kidd) but is unrealistic for the micro NOVA because it would amount to a total redesign of Data General's microprocessor! Indeed the micro NOVA had been redesigned (before

March 1979) in order to be competitive (11). The earlier micro NOVA microcomputer was used for this study only because it was available to the author at the beginning of this work. (The micro NOVA is described in chapter 4).

The latest generation of microprocessors are the bit-slice devices each of which consists of a parallel array of 4-bit μP's (slices) The ultimate bit length is the number of μP's multiplied by four. Their instruction sets may be user-determined using microprogramming which requires a high degree of programming expertise. Wright (24) describes bit-slice μP's as more versatile, faster and more flexible than equivalent bit-microprocessors.

2.1.2 Selection of microprocessors or microcomputers

Microprocessors in the same bit class have different characteristics dependent on the architectural features, the instruction set and the manufacturer's supporting hardware and documentations available.

2.1.2.1 Architectural features

Greenfield (22) grouped most of the μP characteristics under architectural features consisting of:

(1) Computing ability - this measures the effectiveness by which the hardware can perform basic computational functions. The power of the instruction set, e.g. the accuracy in the data word, the word size, maximum number of executable instructions per word and functions available in the instruction set.

(2) Programmability - Evaluates the efficiency by which the μP can be programmed. This is determined by the S/W support available (Assemblers, Cross Assemblers, Editors), high-level language options and whether the S/W resides in ROM's inside the microprocessor or elsewhere.

(3) The microcomputer Development System - This comprises the H/W Support modules and chip sets for interfacing purposes and for control of I/O and memory operations. Prototyping tools (development aids for the users systems) like I/O interface boards, A/D and D/A convertors, ROM, PROM, RAM are placed under this heading also.

Among the requirements for suitable architectural features, Greenfield (ibid) mentioned direct memory access (DMA) as necessary for asynchronous data transfers to and from memory without processor intervention.

The Electrical Research Association (19) splits the selection criteria into ten headings.

(1) Cost of the Microcomputer - Microprocessors account for less than 10% of the total component costs so that the design can be based on an 'overpowerful microprocessor' which is likely to be necessary for future system development. For microcomputers the cost should be assessed within the context of the criteria below.

(2) Performance - the processing speed of microcomputers depends on factors like the clock rate, the bit length, the register configurations, the efficiency of the instruction set, the

memory cycle time, the programming language and available peripherals. Using benchmark tests (standard application programs) relative speeds of processors can be approximately determined. It is recommended that the application throughput required should not exceed 50% of that which the microcomputer can handle. This is even more important for applications which have facilities for high level languages.

(3) Flexibility - Availability of high level language in the system S/W and/or efficient assembler.

(4) Programming Ease - S/W compatibility between the same family of μP's and clear documentation. The range of S/W support currently available suggests that microcomputers should be chosen from manufacturers with established and reputed level of support S/W.

(5) Manufacturers general supports - This includes the field services rendered to the user. Important among these services is the readiness to provide answers to technical questions.

(6) Reliability - They also stated that the integrated circuit technology may influence the possible occurence of 'pattern sensitivity'. This occurs in memories in which the repeated writing of a location may cause the adjacent locations to lose their data integrity. Indeed the functioning of the μP itself may be influenced by this. The installation and its environment (e.g. soldered joints) also affect reliability and the greater the number of chips soldered, the greater the chances of installation faults developing.

(7) Second Source Supplier - Many microprocessors have more than one manufacturer and this is an advantage.

(8) Power Consumption - power consumption is put at 0.5 to 1 W per single chip μP, the cmos μP's requiring the least power. The use of multiple-level power supplies is a disadvantage because they are expensive.

(9) Range of Complimentary H/W - The availability of a good range of compatible chips (e.g. memories) to support the μP is an advantage.

(10) Special Requirements - Certain applications may place constraints on physical size and weight.

Other research workers (e.g. Prasaad (23)) have rightly placed more emphasis on the characteristics of the electronic architecture. The quality of the Accumulators, index registers and data manipulation registers (e.g. operand registers) which serve the CPU is very important. An efficient interrupt hardware scheme requires asynchronous operation of events during interrupts. This is essential because on interrupts, the μP requires time to store the status registers and to recognise the interrupt request existing (by reading the memory regions where the system interrupt vector address is stored in a table) before it services the device requesting the interrupt.

2.1.3 Microcomputer programming with high-level languages

Several papers have been written on the use of high-level languages for programming μP's. Nearly all the writers agree that this is an area with potential time and cost cutting benefits in micro system applications. Teschler (25) predicted that by the mid 1980's more than half of all microcomputer time will be spent running programs in high-level languages. High level languages are structured to tell the μC what to do rather than how to do it. Thus the programmer is free to concentrate on the logical flow of his instructions and thus his productivity is enhanced. In an assembly coding method, one line of instruction usually contains one executable instruction while in high-level language it would translate to several instructions.

Colin (26) observed appropriately that unlike assembly language programming, high-level language codes are not machine dependent and so are inherently flexible. One of the problems in their use is the inefficiency with which compilers translate them into machine codes. Colin acknowledges this limitation but justifies their use in preference to hand-coded assembler by noting that it is the efficiency with which any given compiler is written that determines whether the object code is inferior or not to the hand-coded assembler. A more general solution to this problem has been predicted by Teschler (ibid) when he suggested that eventually μC's may avoid high-level language inefficiencies by incorporating compiler programs within the computer circuits (e.g. ROMS). These computers will then accept only the high-level language programs instead of the low-level assembly codes now in use. Thus, line by line the μC's can make highly efficient translations from high-level language codes to machine codes, thus avoiding problems, associated with the assembly code, generated by conventional translating programs.

Posa (27) says that every line of assembly language, according to studies, costs about $10 to write and this is comparable with the cost of certain μP's! This will not be surprising when one considers the editing and 'debugging' procedures before a line is accepted. Therefore Posa considers that high-level language will be the way to go with arguments similar to those already discussed.

Hicks (28) discusses 'FORTH' a high-level language which simplifies the inclusion of assembly language instructions for I/O control in time-critical applications. It uses codes already defined in a dictionary of macro-instructions which form part of the operating system. The user is even allowed to define his own macros which FORTH will automatically accept as part of the existing dictionary. This system is said to be currently available for Intel 8086, TMS 9900 and Data General's NOVA and Eclipse.

Caudil P (29) and Matto's P G (30) also advocate the use of high-level languages. Caudil added correctly that assembly language can be restricted to programming machine-dependent features like interrupt-driven I/O devices for which timing is critical.

NEL (31) has announced the development of their CAPE (Computer Aided Planning and Estimating) program for micro-CNC Users. It is a manual data input system which allows parameters like diameter or depth of holes, shape of material to be removed in turning operations or volume to be removed in milling to be entered through a teletype VDU keyboard. A data file holding the appropriate information on the user's machine tool, workpiece material and normal machining practices is then accessed. Values from these are used for the necessary

calculations, like machining times, set-up times, total batch times and cost. The system is based on FORTRAN and has been implemented on an 8-bit μC with the system generated on floppy disks.

Some writers have commented on the memory penalties and real time constraints presented by the use of high level languages. The memory problem is rapidly losing significance because the degree of memory integration is dramatically increasing while prices are fast dropping. Also programs can be carefully organized and proven to take advantage of the ever-increasing μP processing speeds to overcome timing problems in real time applications.

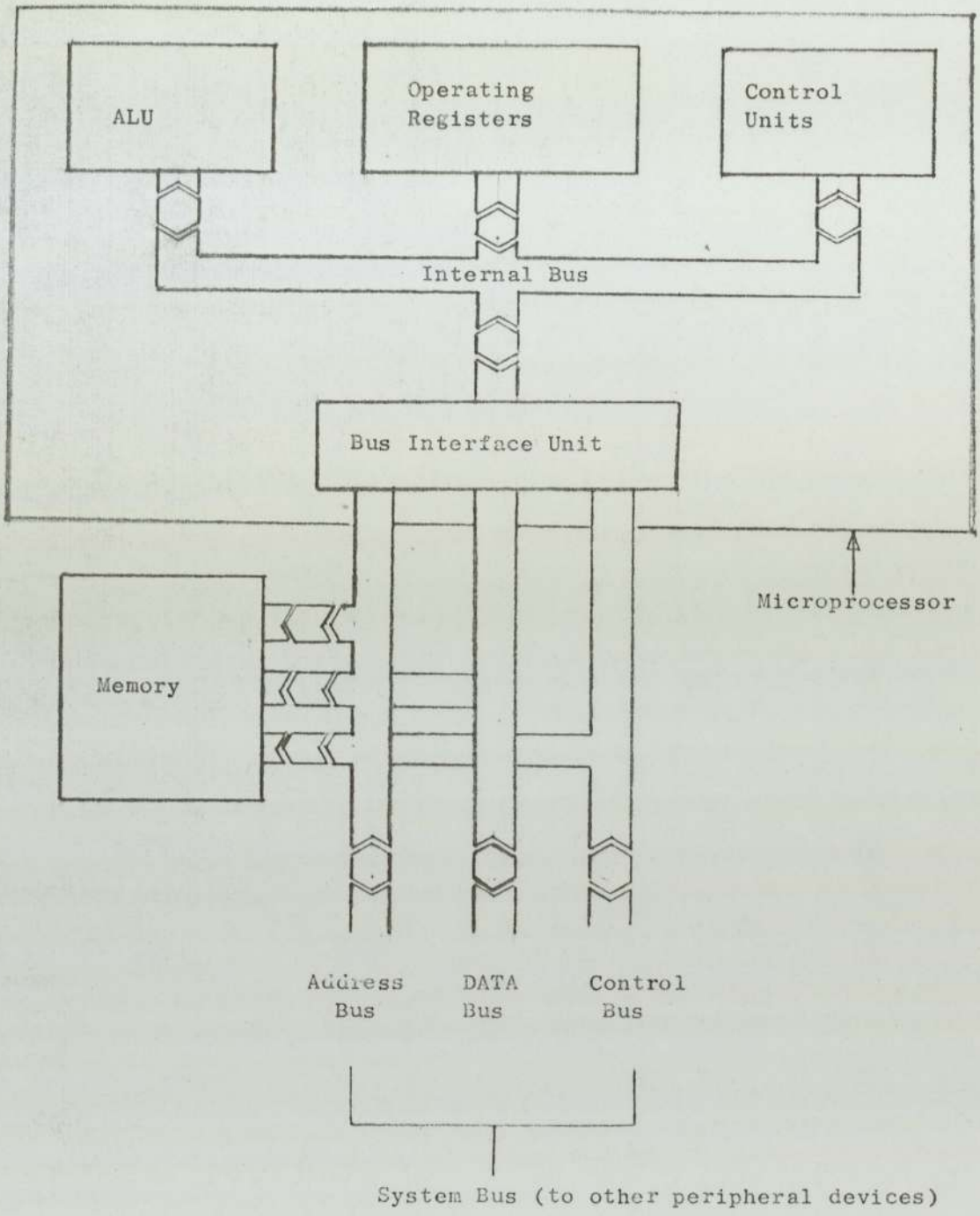


FIG 2.5 Single chip Microprocessor and System Bus

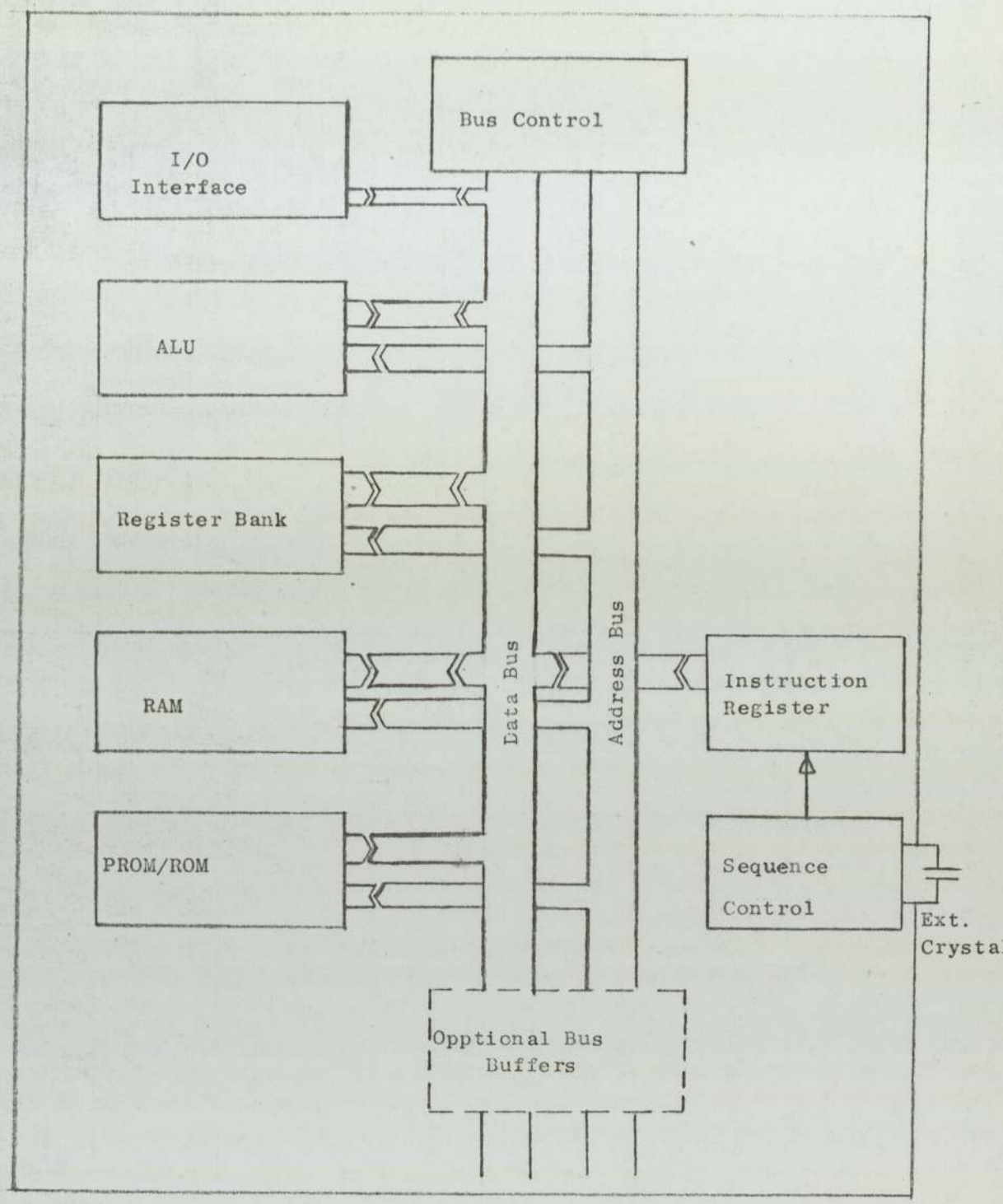


FIG 2.6 Single Chip Microcomputer

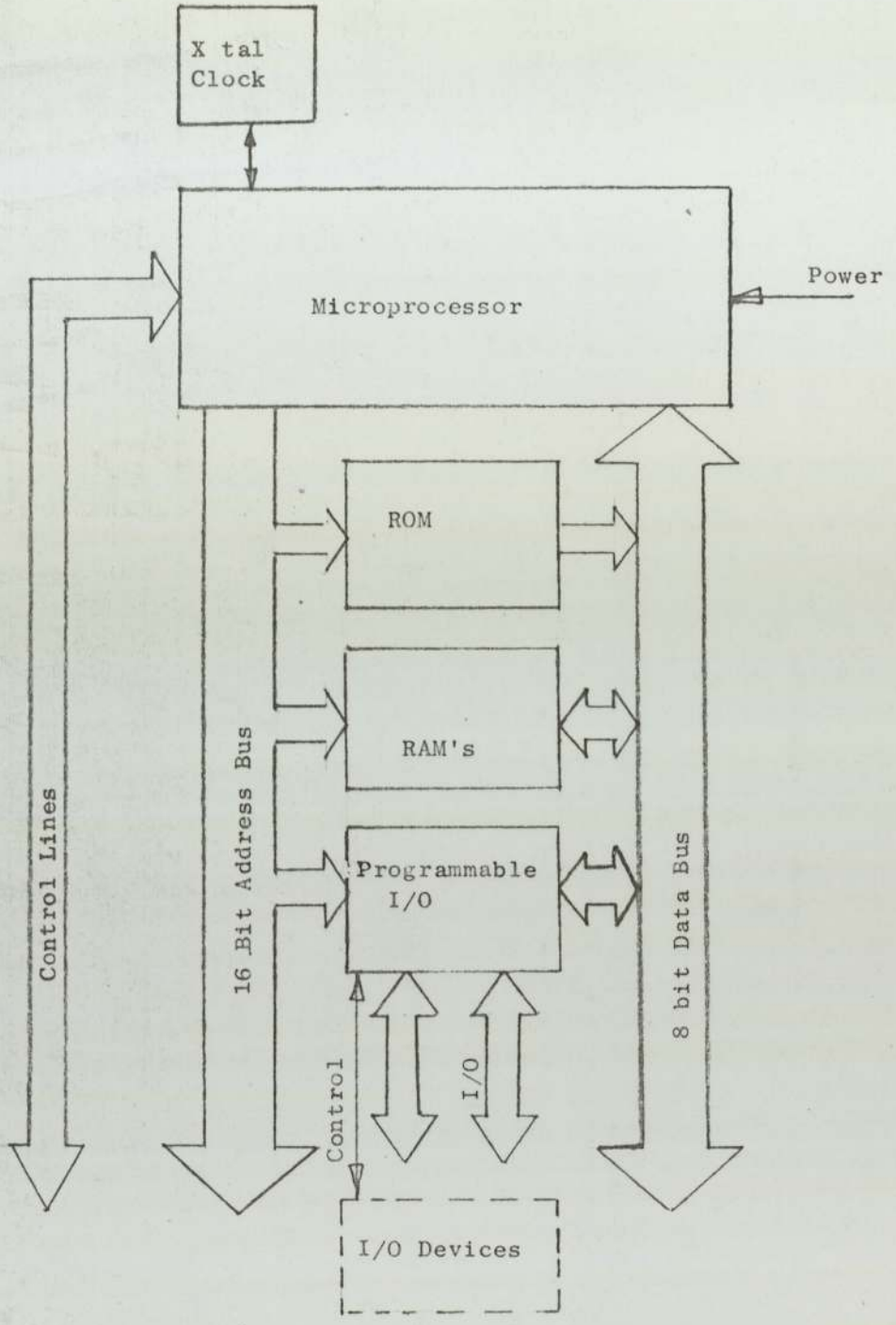


FIG 2.7 Standard Microcomputer System

2.2 Microprocessor-Based Control Systems

2.2.1 The design of a microprocessor-based system

Bottari (32) has given a summary of the procedures to design a microprocessor system. It starts with a detailed analysis of the requirements for the control system, including the functions required of the μC by the system. A survey of available μP and μC literatures with reference to the market is next carried out. He recommends the use of a flow chart in a high level language to put more detailed analysis of the processor functions, including calculations, into perspective. Using the selection criteria already discussed in section 2.1.2 a short list of competing μP's or μC's could then be drawn. A block diagram of the system with features like I/O line requirements is drawn. The comparative instruction execution times and bus features may play a major part in the final selection.

For μP's with the same bit length, speed may be a decisive factor (other criteria having been taken care of). For sampled data control loops, Bottari recommends that the cycle frequency be at least two to five times the highest frequency of the signal being sampled. Within the sampling interval all the calculation and I/O functions must be executed by the μP. The benchmark tests for instruction sets, mentioned earlier, would be useful at the final stages of selection. (The μP manufacturers co-operation is necessary here).

More detailed study of the processor architecture would then follow. Tseng (33) states that the difference between micro's currently in the market lies not so much in the cost or stated performance as in the driving softwares. These include both the

system and user S/W. Therefore knowledge of both Assembly language and the relevant high level language is essential. The application program can then be written and the prototype developed with the help of the available development aids supplied by the manufacturer.

The Electrical Research Association (19) also made suggestions on design procedures following the selection of the μP. These are summarised in sequence as follows: 1. Problem Analysis, 2. Project planning, 3. Hardware design, 4. Software design, 5. Programming, 6, Hardware testing, 7. Software testing, 8. Prototype Evaluation, and 9. Documentation of the design stages.

2.2.2 Microprocessor-Based CNC Systems

The first part of this chapter dealt with typical minicomputer-based NC Systems which were implemented before microprocessors became fully established, from the mid-1970's to the close of the decade. Since the mid-Seventies several μP-based NC systems have been implemented. Some are based around a microprocessor while others are multi-microprocessor systems.

In tracing the history of NC and CNC systems Becker (36) points out that μC systems which only represented a minute percentage of existing control systems will gain more and more ground due to the development of more powerful 16-bit μP's. Supporting the use of LSI's, he mentions that they enable the user to transfer some functions of a μC - based control back to hardware while still maintaining contact through their interrupt - request capabilities. One thus reduces the computer load and gains the opportunity of employing a small computer, ie a μC which then handles the input, interpretation, calculation and distribution functions.

Microprocessor CNC systems can be subdivided into two general groups (37):

1. Fully flexible μP CNC: These have their control systems implemented in read-write memory which therefore can be updated or modified at any time without changing the control hardware. They usually have programmable interfaces, the resolutions of which are in the range of 0.001mm under continuous control. Often these are multi-processor CNC systems.

2. Fixed Feature μP CNC: Here the systems softwares are fixed in ROM which can be altered only by replacing the ROM chip. (If PROMS or EROMS are used then the chip can be reprogrammed). Several CNC equipment manufacturers make this class of μP-based CNC because the high cost of system program development can then be shared between a large number of systems to be marketed. They are usually dedicated to a machine and often only such features as are included by the manufacturer will be available to the user. They are usually single microprocessor CNC systems with paper tape inputs, or more recently MDI's or floppy disks. The interpolation and feedrate control NC functions are implemented in the S/W which is PROM-resident. These control systems become effective immediately on power-on. Examples of this class of μP-based CNC systems are Philips 6600, Plessey RUSC, Siemens-Fanuc System 5. Table 2.0 gives a general classification of some present CNC's

For part programming, MDI techniques can be used for the Plessey RUSC and Philips 6600. Instead of the usual paper tape, cassette recorders are used to store part programs for future use. For operation they are loaded into read/write memories from which they are run. Since they are resident in read/write memories, the part programs can be edited on the machine.

Savage (38) described a low cost μP-based two axis system equipped with loader/editor software which offers full edit facilities at a CRT keyboard. The sampling frequency for feedrate and position control is 100HZ. The position following error in the Closed loop is fed to the D/A converter which control the d.c. axis drives. The time lag in the position control loop is 100μSec, and this is the delay between the

receipt of the feedback signal and the transmission of the new following error to the D/A converter. Details about the method of interpolation and the rest of the system were not given.

Hinduja, et al (39) describes a μP-based interactive system for an NC Lathe which an operator can program by way of the selection of integrated macros which are located by means of hardware press-button switches. Each macro selected (TURN GROOVE, THREAD, DRILL) is supplied with the necessary arguments (variables). Macros can be repeated whenever the operation which they define recurs in the machining sequence. Calculations of data for repetitive operations like depths of cut in turning feedrates and spindle speeds are automatically made by the system. The procedures for these calculations were not given but the paper mentioned that technological information such as workpiece material and tooling are provided by the operator at the initialisation. Geometric features (like TAPER, FILLET, etc) can also be selected from the system.which is based on an 8-bit μP with 48K memory and backed up with two 80K floppy diskettes. Details about the control of the lathe on which the system was implemented were not given. Further analysis of this award-winning project (Hinduja et al) is made in reference 40. The requirements and virtues of MDI systems were discussed. Less skilled programmers are needed to operate MDI systems because specialist areas like identification of company-chosen cutting technology and editing procedures are integrated in the control system. This paper finally mentioned that the system has been developed as a retrofit to an existing Hydro NC540 lathe on which it works in a BTR mode, thus allowing the machine to be operated conventionally.

On MDI systems Tipton (41) cited examples of μP-based keyboard-programmed NC systems. These range from systems which incorporate

RS232C serial I/O connectors to cassette recorders of part programes to systems which contain machining subroutines or macros selectable via the keyboard during the actual machining operation. Feedrates can now be directly programmed in the actual units (mm/min or inch/min) without the need to know the conventional G-Codes characteristic of tape inputs. He correctly observed that the developments of MDI and of work-surface programming are pushing up the threshold at which Computer assisted programming languages become necessary for a part program. For storage purposes the part program which is now resident in memory can be punched or recorded onto tapes (or even diskettes) for future use.

Among the CNC systems offering MDI as extras are those based on multi-microprocessor designs. Typical examples are the Siemens Sinumerik-T range of CNC's and McDonnel Douglas Actrion III CNC system. These are systems with modular S/W and H/W design using two or more µPs. Typically, one µP controls the data handling (the input of data, the selection of the mode of operation, etc), while the second microprocessor calculates the cutter path (interpolation) and the third controls the axis drivers. More µP's could be used for controls like digital read out (DRO).

The McDonnel Douglas Actrion III CNC system contains three µP's whose functions are as described above. The tape reader and control keyboard (operator panel) are interfaced to the first µP (24). The 2nd µC is a five-axis interpolator which computes all axis motions as a function of time and so can be linked to a position sensor on the main spindle to provide for functions like thread cutting. The third µC controls the position of the feed drive and optimises the contour. All the µC's are of the 16-bit class. The feed drives are by d.c. motors and the position loop is closed inside the third µC. The assignment of

a calculated number of position increments to a sampling period determines the feedrate in the axes.

Becker (36) gave a preview of the present Siemens multi-microprocessor systems. The features of this system are also described in reference (42). The three 16-bit μC's perform functions fairly similar to those described for the Actrion III. These modern CNC systems from Siemens use bubble memory which has more capacity for storage integration and have the more significant advantage of being recoverable in the event of power failure without needing battery back-up. Up to 128K of memory is available for a single μC stand-alone system or 256K for the whole multi-processor system. With this level of memory, there is enough capacity to enable the user to write his own special subroutines into the system and they would be resident in memory.

Brussel H V, and Simons J (34) describe a multiprocessor system involving a minicomputer and three microcomputers. Each μC is provided with its own PROM and RAM modules. The minicomputer calculates the data blocks using the DDA algorithm and outputs them to the slave μP's RAM. The μP's receive data relating to complete line segments through a direct memory access (DMA) bus or buffered I/O transfers. Intel 8080 was used to implement the timing of the loop in the DDA algorithm. The timed pulse generation sequence thus achieved is used to control the feedrates on the axes with another microcomputer. The position feedback loop is closed inside this μC which continuously compares the digital inputs from the encoder with the commanded incremental position from the interpolator. Like the system described by Savage (38) the resulting position error is sampled every 10ms and entered into the μC. This sampled position error is converted to proportional analogue voltage which becomes the velocity command to the d.c. motor axis drives.

Prasaad (23) and Popa (20) described the design of a multi-microcomputer NC system with stepping motor drives. The microcomputer used was Motorola's MC6800. The system includes a PDP11/10 minicomputer which handles the input, task scheduling, data interpretation and distribution NC functions. Prasaad concentrated on a single stepping motor control with MC6800 with a view to its inclusion in a multiprocessor system. Like the system discussed in the last paragraph the feedrate control is achieved with a sampling period of 10ms. The interpolation is subdivided into 'coarse' and 'fine', implemented with separate μPs. The drive is in open loop but the fine interpolation μP counts the pulses sent to the stepping motors and gives these the highest priority in the interrupt configuration. Popa carried forward the work described by Prasaad by designing a direct memory access (DMA) bus which will be used to implement the multicomputer NC system.

2.3.3 Advantages and Disadvantages of Multi-Microcomputer Systems

The greater processing power and overall flexibility inherent in modular designs are undoubted advantages of multi-computer CNC systems over the single microcomputer types. They are claimed to have better reliability than single processor systems because the redundancy which is usually inbuilt into such systems can come into use in the case of any part breaking down and it offers more room for future expansion. At the current stage of development it is still too early to be sure of this.

Their relative complexity and high development costs are among the important disadvantages. In a discussion with the author a Siemens engineer estimated that it would take about 12 months for 15 experts to

implement such a system. 'The principal difficulty is not so much in the H/W design', he said 'as in the programming of the interacting μP's to ensure a co-ordinated CNC system.' Another problem mentioned by Wright (24) is that multi-computer systems can suffer from protracted interprocessor faults in which the fault appearing in one processor could be caused by another despite the fact that the processors are arranged in parallel.

The indications, therefore, are that both single and multi processor CNC's will continue to develop side by side. This is seen to be quite probable when one remembers that multi-bit microprocessors (possibly in the form of integral bit-slices) with minicomputer-like processing power are already appearing in the market. One area of application which may take advantage of this development offering low cost control systems is the stepping motor CNC, especially those in open loop.

Table 2.2 Classification of some available NC systems

Manufacturer	µP-CNC		MDI	Hardwired	Minicomputer
	Flexible	Fixed	Systems	NC	computer CNC
Anilam Electronics (UK)		Commando Crusader	MDI 20		
ASEA Ltd	NUCON 400				
Bosch Ltd (NC Division)	5M				5
Cincinnati Milacron				8D	Acramatic CNC
International GEC (N.Y)	1050	1050L 1050HL	1050HL	550	
Giddings & Lewis Sabs Ltd			Numeriset	Numeripoint	CNC800
Herbert Machine Tools Ltd					Batchmatic
Kongsberge Ltd					CNC200
Philips Machine Tools Control		6600			
Plessey NC Ltd		RUSC		1100	7300
Posidata Ltd		CNC1800 CNC2800			
Siemens Ltd	7	5	SPRINT T SPRINT M		550C
Westinghouse Elec. 2560					New World CNC
SMT Pullmax (UK) Ltd					CNC220

2.3 Computer Control of stepping motors in NC systems

2.3.0 Introduction

By the late 1960's the advantages of digital processor control of stepping motors had been recognised. It was evident that the incremental nature of the motion of stepping motors was highly suited to digital computer control. Before the advent of the microcomputer in the mid-1970s, minicomputers were being used to implement CNC systems employing stepping motors.

2.3.1 Minicomputer Controlled stepping Motor Systems

Fredriksen (43) discussed various ways by which direct computer control of stepping motors could be economically realised in closed loop. In this closed loop control the computer samples the feedback of the motion of the motor shaft to ensure that the last step commanded has been executed before the next step pulse is issued. The interval between these has to be carefully controlled to prevent what some writers refer to as 'hunting', the excessive oscillation that causes resonance instability in the drive. Opto-sensitive disc encoders are used to provide the feedback of the motor shaft position. Fredriksen discussed the use of minor and major loops in his control schemes, the minor loop feeds the encoder outputs back to the drive logics consisting of the last step memory, the pulse sequence and position translator and the power switching circuits. The major loop takes the feedback to a position-error command counter which is directly controlled by the minicomputer. The computer also directly provides the motion pulse patterns, thus creating a complete digital servomechanism.

He rightly pointed out that closed loop control of stepping motors uses much of the available processor time and so even the minicomputer requires some external interface H/W to reduce demands on it. Accelerations and Decelerations were software controlled.

Fredriksens (ibid) closed loop does not contain the machine slide which is the member being positioned and so possible lead screw and slide errors are not taken into account. Loss and Frisch (44) used TRAV-A-DIAL encoders which are friction-driven by the machine table to feedback absolute position through a 20-bit TED digital read-out to the controlling PDP8/E minicomputer. The sampling period between consecutive steps was fixed at 5ms which gave a maximum traverse speed of 12 in/min. This low speed was a result of the required computation time which included the conversion of the BCD read out to pure binary code for computer storage (done with a special subroutine)

Akgerman (45) who was an earlier colleague of Loss and Frisch also used a PDP 8/E minicomputer in a different setting to control a Bridgeport MK II milling machine. The machine was equipped on all three axes with Sigma's electric stepping motors controlled with bipolar drives. An APT-like language called 'FOCAL' was complemented with Assembly language to constitute the control S/W. The input was by paper tape and by using linear approximations for circular arcs, the blocks of which were stored in circular buffers, the interpolation technique would be applied to all motions. A programmable real-time clock was used to drive the buffered I/O's and in this way the feedrate was controlled. It must be noted that the control program involves the use of sines and cosines which are known to be time consuming routines in any language.

Plas and Blommaert (7) also used a PDP 8 minicomputer, this time to implement the control of electro-hydraulic stepping motors. Incremental interpolation data to H/W interpolators was supplied by the minicomputer at a sampling period of 10ms.. The torque/speed curve of the stepping motors was used to calculate data for an exponential acceleration scheme which was implemented by H/W under computer control. The scheme was based on an exponential distribution in which either of two time constants could be used depending on the range of slew speeds required. With a resolution of 0.0005 mm, speeds up to 5m/min were reported. Although no mention was made of the effect of friction (if any) in the electro hydraulic system, they reported appropropriately that the hydraulic motor following linear error was taken account of in circular interpolations.

Andrew J P (46) reports briefly on a computer-controlled engraving machine incorporating stepping motor drives. The S/W was said to include a MDI system but no further details about the control were given except that the system helped to reduce production time per part by up to 75%. Compared with the conventional system this was said to yield a cost saving of 50% per 40-hour week.

2.3.2 MicroComputer Control of Stepping Motors in NC Systems

2.3.2.1 Introduction

One of the purposes for using stepping motor drives for CNC is to provide an effective machine control system at low cost. The use of a minicomputer for this control has been described correctly by Cavey (47) as 'extremely effective but very expensive'. Therefore, the

purpose is best achieved if a microcomputer is used to control the stepping motor drives. The flexibility and economy offered by the developments in μP technology has widened the horizons for stepping motor control and its subsequent improvements. This explains the growing interest in stepping motor CNC.

2.5.2.2 Microcomputer-Controlled Stepping Motor Machine Systems

Cavey (ibid) applied a microprocessor to control two slo-syn stepping motors for an X-Y plotter. Automatic accelerations and decelerations were provided by an existing H/W indexer module for the motor drives, which included manual selection of starting and maximum speeds. In this way he simplified the S/W for this system which he said would otherwise cost thousands of pounds and take many months to be implemented 'in-house'. He estimated the cost of typical hardware (including lead screw) at £1500 per axis, excluding special control circuitry. There were no further details about the μP and how the interpolations and feedrates were controlled but he added that his design could be used for higher power and more complex industrial CNC systems at the estimated cost.

Weston and Charles (48) applied a TMS 9900 μP to control stepping motor drives for electron-beam welding and cutter grinding. Interpolations were done by direct calculation of co-ordinates of subsequent positions along the contour. H/W support at the interface was provided by decade rate multipliers (DRM) which function as interrupt timers for the microcomputer. These interrupts are counted and the results are used to update the X and Y positions in a fixed proportion per sampling period. The co-ordinate increments per interrupt and the DRM-driving clock frequency are the basic values used to determine the feed in each

axis. They quoted a loop execution time of 1MS for calculations, interrupt servicing and I/O functions.

Latham and Pelc (49) also employed the TMS9900 μC to control the stepping motors driving the positioning table of a glass grinder. Two binary rate multipliers (BRM's) are loaded with cosine and sine values of the interpolation angle to provide the pulse rates for the X and Y motors respectively. One H/W dividing unit is interposed between the BRM and each motor to help to smooth the irregular pulse pattern typical of BRM outputs. The synchronisation of the axes is achieved by deriving the pulses from the same basic clock. Every output to the motors is used to interrupt the μC which then updates the position count. A separate BRM is programmed to provide the accelerations and decelerations.

A 'TOS' centre lathe has been retrofitted with stepping motor drives, the control of which was initially developed with a DECLAB (PDP11/40) minicomputer. The author, Roberts (50), intends the final system to be microcomputer controlled. He rightly noted that this will provide the necessary flexibility at low cost and that the absence of feedbacks greatly simplifies the control system which can them be dedicated to storing and issuing optimised pulse data to the stepping motors. The original system S/W is based on DOS with Macro-Assembler Subroutines requiring up to 16K of memory for various lathe operations. The required subroutine is swapped between diskette and memory; and management functions like machining time, total time and cost can be included. There was no further information about the feed drive and speed control except that the carriage became the X-axis and the slide the Y-axis in the new drive system.

Wells (51) describes a closed loop microcomputer-controlled one-axis stepping motor drive. The μC used was Intel 8080 which controlled the axis feed by timing the spacing of pulses fedback by optical encoders. The acceleration to this speed was controlled by a value of delay time which is 75% of the time required for the last step. Thus the next delay time is predicted from the present step and when it elapses the motor phase is switched via the drive logic board. The deceleration algorithm involves the skipping of one pulse and the fixing of the delay value until the maximum starting speed is reached when the delay is set to zero.

Al-Anbuky and Swadi (52) describe the retrofit of an existing NC drilling machine with the Intel 8080 μC and stepping motors. The μC sends a sequence of NC data blocks (position, direction, speed) to an external bank of registers which then generate feed forward pulses to drive the stepping motors with a resolution of 0.1mm point to point. Except for stating that the acceleration/deceleration scheme is exponential, no details are given about the rest of the control, especially the registers and how they control feedrates.

Prasaad (23), (mentioned earlier), describes a stepping motor controller based on the MC 6800 μC involving a PDP11 minicomputer which handled the program and data interpretation and distribution to the μC. The μC is interfaced to a programmable interface adapter, (PIA), MC 6820 and BRM's which provide the feed forward pulses to the motor logic board. After each pulse the PIA interrupts the μC which then times the output of the next pulse (with counters) to achieve the open loop acceleration/deceleration scheme developed earlier by Rahmen (8). Although Prasaad's system was implemented on one axis, he suggested the use of algorithms (like Bresenham's) to achieve the necessary two-axis linear interpolation

in a multi-computer control system. The design of this system was the subject of another thesis by Popa (20) who provided some theoretical analyses which are useful for deciding the selection of the various existing interpolation algorithms and feedrate control strategies. He also illustrated the multicomputer bus timing considerations which led to his recommendation of the direct memory access (DMA) bus structure for the multicomputer NC. The requirement in men, equipment and time necessary for such systems (refered to in section 2.3.3) to reach only the preliminary development stages is highlighted again by the studies described above.

Another multicomputer NC system which includes a minicomputer, a μC and stepping motors is described by Weck et al (53). Multicomputer NC is outside the scope of this study but has been mentioned for the sake of completeness.

Lafreniere (54) rightly recommended the use of acceleration/deceleration look up tables to achieve an 'optimum' open loop control of stepping motors using a μC. His system was implemented with an Intel 8080A μC and two four phase superior electric stepping motors which were the drives for film camera positioning mechanism. The acceleration/deceleration table contains the time delay counts before the next pulse is output to the stepping motors. These counts (integers) are pre-calculated using an exponential distribution which is considered the 'optimum' acceleration/deceleration curve. When the minimum count is reached it is kept constant for the slewing speed until the beginning of deceleration when the table is read in reverse order. To use this technique a special area in memory is reserved for dumping the table.

Crossley and Unsworth (55) used the MC 6800 μC and its development modules to perform the NC functions for a draughting system for the verification of NC tapes employed in the manufacture of laser cut die boards. The axis of the draughting machine are driven by stepping motors under the control of the μC which generates the pulses and sets the directions of motion. The general control program is permanently resident in ROM, while the RAM is used for part program inputs. The μC interface has 16 bi-directional I/O ports which are used for on/off controls on the stepping motor and the machine. The motor is controlled in a step by step algorithm and a counter is used to record the number of steps required. Circular interpolation is based on the simplified version (Appendix A) of the general conic equation:

$$f(X,Y) = Ax^2 + 2HXY+By^2+2FX+2GY+C = 0.$$

By working in 24-bit precision the computations are able to provide accuracies in the region of ± 1 part/million. In order to reduce the control S/W development time, the assembler codes were cross-assembled on a PDP11/45 minicomputer to produce the object file which is then punched on tape and later read into the μC memory.

The growing interest in the low cost microcomputer-stepping motor NC systems is currently demonstrated by the popularity of the Bridgeport's Series 1 CNC (56) milling machine. The 3 axes of this machine are driven by 3 superior electric stepping motors (through 2/1 reduction gearboxes and lead screws) under the control of a single microcomputer with tape or MDI input options. The control includes PROM-resident linear and circular interpolation routines and 8 canned Z-axes cycles. At a total standard price of just under £20,000 it has been described as 'popular' (ref 37, p. 166) and the 'finest that money can buy' (ref 57, p. 37).

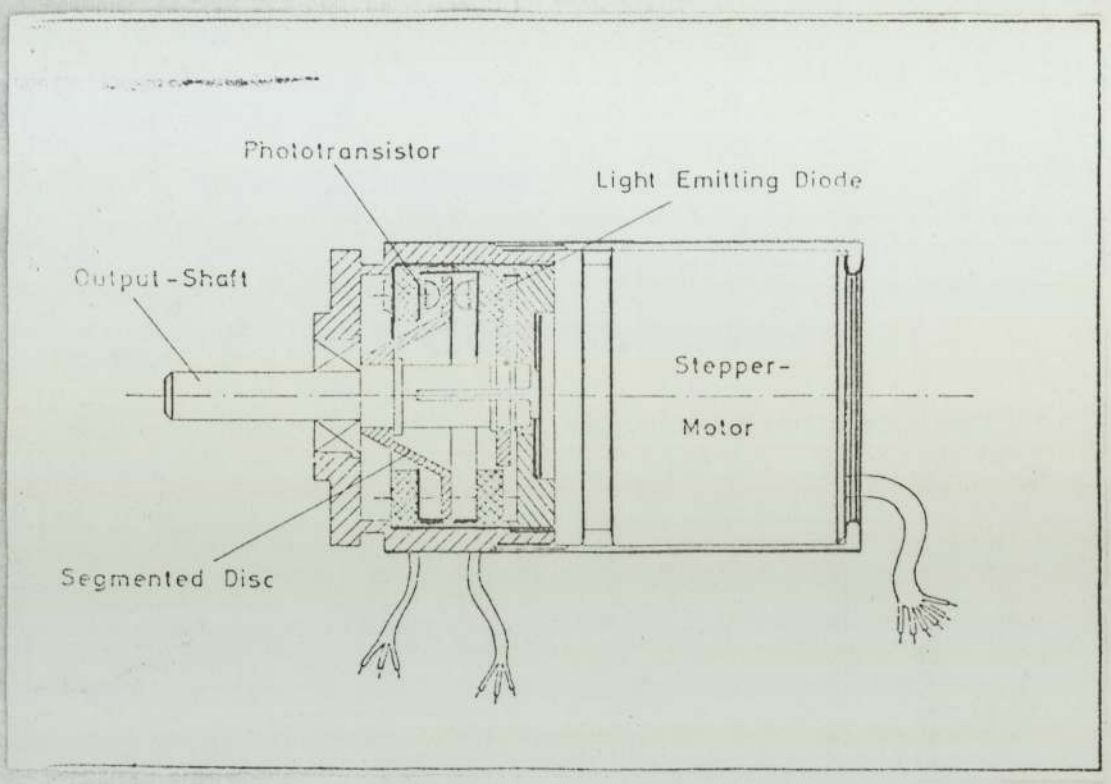


FIG 2.8 SM Shaft Encoder Assembly

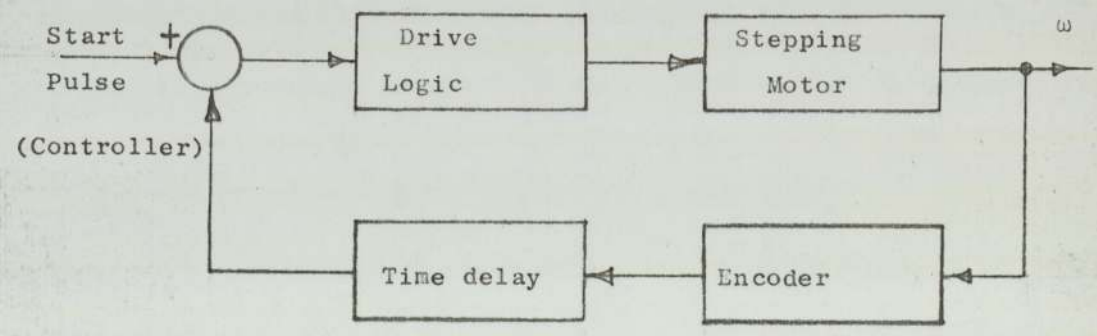


Fig. 2.9 Closed Loop SM Drive with Shaft Encoder

2.4 Stepping Motor Control Characteristics

2.4.1 Introduction

The potential advantages of using stepping motor control systems are the relative ease of design implementation, the control flexibility, the low general costs including maitenance and the high positioning accuracies with direct feedrate control involving few system components. The extent to which these benefits are realised depends very much on the quality and type of control strategy and its overall installation. Closed loop control of stepping motors improves the precision and reliability of the system but negates the other potential advantages which are easily achieved in the open loop.

2.4.2 Closed and open loop controls

In machine tool applications it is often necessary that an axis be moved as fast as possible to a new position so that the next machining cycle can start. Depending on the traverse capabilities of the machine drive, it is often necessary to accelerate to some optimum top speeds and also to decelerate to a low speed and finally stop with the cutter in fresh air. For an electric stepping motor, closed loop control enables it to be accelerated to slewing speeds without the possibility of losing steps or stalling. This results from the digital servomechanism-type of operation (fig. 2.9) of the motor shaft encoder (fig. 2.8) giving the feedback for closed loop control. The photo-transistors (photo-electric sensors) deliver pulses to the translator in such a way that the motor can generate maximum torque.

The inclusion of a feedback not only complicates the system by introducing more precision components and increasing possible sources of error, it reduces the general control flexibility by imposing a dependence on feedback information. This makes the system less competitive with alternative drives like DC motors.

To some degree, the benefits of closed loop control can be achieved in open loop if the torque/speed characteristics of the motor are taken into account in the initial control design. The improved performance of current drive controls can provide a reliable and sufficiently powerful actuation for CNC machine tools. The absence of feedback loops considerably simplifies the overall control system and the existing capacity can therefore be dedicated to the storage and generation of optimised pulse signals to the stepping motors.

In this study emphasis will be given principally to the open loop control strategy and the literature survey will be biased towards programmable control aspects like acceleration/deceleration and related aspects such as oscillations and damping. For further information on other areas of stepping motor design and control the author would recommend the following references: (58), (59), (60) all of which were found to be informative. References (59) and (60) are proceedings of annual conferences dealing with stepping motors while (58) is a book, edited by Kuo, containing some papers from earlier conferences and including that of reference (59).

2.4.3 Resonance oscillations and damping

Russel and Pickup (61) discuss the occurrence of resonant oscillation of stepping motors at pulse rates above the natural frequency, ω_o. They found appropriately that the resonant frequency increases with decrease of rotor inertia in a non-linear relationship and that the mode of excitation affects the frequency at which resonance sets it. They estimated the frequency band at which resonance occurs to be between 10 and 20 pulses per sec (PPS). The non-linear shape of the motor torque/angle curve also determines the occurrence of sub-harmonic resonance and that for variable reluctance motors the five-phase types exhibit less tendency to sub-harmonic resonances (e.g. at $\omega = 2\omega_o$) than the four-phase variety. In another paper (62), the same researchers suggested ways to correct the instability caused by resonance. This is by using a frequency modulation of the command pulses by way of signals which are proportional to the velocity-oscillation during the unstable operation. The signal is provided by a tacho-generator attached to the motor shaft and so is proportional to the instantaneous speed. The oscillating component superimposed on the d.c. voltage during unstable operation is filtered and amplified and then fed back, after phase-shifting, to the voltage-controlled oscillator input, and thus the clock pulses are frequency-modulated. They claim that this technique is more effective than other mechanical damping methods used in open loop but the inclusion of the tachogenerator feedback and associated circuitry adds to further control complication. Besides this, the technique can only be employed effectively with voltage-controlled signal generators, and microcomputer controlled pulse rates are often not of this class.

Pawletko (63) describes various control techniques and their effects on damping motor oscillations. One of these is the 'bang-bang' operation which involves switching two adjacent phases at slightly different times so that currents in them overlap. Another method is using the half step mode at start and during accelerations and transfering to a full step mode at slewing speeds. He gave further insight into closed loop control involving μP's and in this he pointed out that support H/W especially for accelerations would quicken traverse times in short distance positioning.

Leenhouts (64) discusses the use of mini- or micro- stepping motors and drives to achieve almost infinite resolutions and reduced resonance effects resulting from the smaller excitation amplitudes involved.

The operation of micro-stepping drives follows from the derivation of the torque contributions from, say, two phases, A and B:

$$T = T_A + T_B \qquad 2.2$$

where T = total torque

T_A = torque from phase A

T_B = torque from phase B

(A and B being offset by 90^o)

$$T_A = T_M K_A \text{ Sin } \theta \qquad 2.3$$

$$T_B = T_m K_B \text{ Cos } \theta \qquad 2.4$$

where K = ratio of rated and actual winding currents

θ = rotor angle in elect. degs.

$T_M = \text{Max}^m$ Static Torque

$-1 < k < 1$

$\therefore$ From (2.2) $\quad T = T_M (K_B \cos \theta - K_A \sin \theta) \qquad 2.5$

Putting $K_A = \text{Cos } \alpha$ and $K_B = \text{Sin } \alpha$

(where $o < \alpha < 360$),

from (2.5)

$$T = - T_M.\text{Sin } (\theta-\alpha) \qquad 2.6$$

Microstepping is now a matter of changing the values of α by small equal increments. This requires the control of the phase winding currents by incrementing α in the following equations:

$$I_A = I_R \text{ Cos } \alpha \qquad 2.7$$

$$I_B = I_R \text{ Sin } \alpha \qquad 2.8$$

where I_A and I_B are the phase currents of A and B and I_R is their max^m rated value based on single phase switching mode.

Incrementing 'α' alters the value of the stable zero-position torque (holding torque) by a small amount and so causes a microstep. He gives the equation for deriving the number of pulses per microstepping cycle from the natural steps/rev as:

$$M_C = \frac{4.M}{N}.P \qquad 2.9$$

where M = desired no of microsteps/rev.

N = no of natural steps/rev, (e.g. 200 steps/rev)

P = an integer of minimum value = 1

Mc = reqd. no of pulses to complete the micro-stepping cycle.

The value of P is the next integer which makes Mc an integer also. Leenhout reported considerable improvements with regard to the onset of resonance, oscillation amplitudes, position errors and general motor/drive stability through his microstepping scheme.

Hughes et al (65) and Beling (66) all agree that bipolar drives are better in the production of torque available than unipolar drives for high power applications and that the two-phase mode gives better damping but at the expense of a reduction in torque available.

Beling (ibid) further discusses the effects of a control scheme in which a high voltage is supplied at the start of the pulse and maintained until the current reaches a predetermined value, when the phase is switched off allowing the current to decay. He points out that this implies a time modulation of variable voltage and so enables the drive to achieve efficient high start/stop stepping rates. On multi-step drives (i.e. $\frac{1}{2}$, $\frac{1}{4}$ 1/8 steps), he concluded that shaft oscillations were reduced and so machine tool chatter or lack of straightness in linear X, Y plots are considerably reduced too. His results agree with the author's ($\frac{1}{2}$ step mode) and Leenhout's microstepping control strategies but it is obvious that multi- or micro-stepping would involve extra drive complexity. (It is interesting, however, to note that he had achieved a 2HP output with a motor 4.2" Dia x 10.6" length using 1/8 step mode).

Maginot and Oliver (6) agree with the above findings but comment rightly that not only will the multi-step circuit drive for four-phase motors be complex, it is also difficult to achieve critical damping using the technique. They state that the delayed-last-step electronic

damping scheme could be applied to multi-phase motors to achieve critical damping but that the system would become sensitive to load changes. They suggest that viscous inertia dampers can be fitted to the motor shaft but that these add inertia to the whole system and so tend to slow the response to command signals (accelerations).

Kordik (68) agrees with the disadvantages of a viscous inertia damper but adds that there are important advantages, namely:

(i) It does not effect step accuracy
(ii) It has no effect on the maximum slew speed
(iii) It modifies the resonance point.

The author's experience agrees with Kordik's and it would be added that the adverse effects of viscous inertia can be considerably reduced by operating at multi-steps (higher torque available) and at the same time being ready to tolerate some degree of oscillation. Indeed, oscillations are rarely completely damped out.

Kuo and Singh (69) review the existing state of the art in damping and state that the addition of inertia to the shaft is known to improve the torque-speed characteristics in steady state operation.

2.4.4 Torque, Acceleration/Deceleration

The rate of acceleration which a motor is capable of following depends on the torque which is available to it at the instantaneous speed. The non-linear nature of the torque-speed characteristics (fig. 2.10) makes it difficult to predict at any given instant the

appropriate accelerating rate. Often the manufacturer's curve is only an approximation of the no-load operating condition. Lawrenson et. al.(71) and Justice (67) were concerned mainly with the prediction of starting and stopping rates and the static torque for given phase currents and shaft angles.

Justice (ibid) developed a model of a bipolar and bi-level circuit which takes the non-linear torque-speed characteristics into account to predict what torque is available at a given phase current. Lawrenson et al. (ibid) also suggest bi-level voltage as one of the methods to boost the starting and stopping rates and report an increase of over 200%.

Acanley (72) concludes in his thesis that the half step mode produces maximum torque at all speeds and the highest motor and system efficiencies are obtained for one-phase -on operation, while two-phase -on produces the lowest efficiencies.

Jufer (73), Frus and Kuo (74) suggest the achievement of fast accelerations and decelerations by feeding back the phase current before it enters the motor. They all agree that this gives significantly improved results, provided that the level of current can be sensed currectly before it is fed back to the drive but the method of using this technique must be established for individual drive systems.

Maginot and Oliver (6) reviewed

(a) linear, ramp acceleration/deceleration

(b) digital or rate multiplier (e.g. BRM)

(c) exponential ramp up and inverse exponential ramp down, acceleration schemes.

They recommend scheme (c) (figure 2.11) as the optimum control strategy in open loop - in that it provides fastest acceleration/ deceleration.

Singh and Kuo (70), however, applied the alternative scheme for accelerations and decelerations in their minicomputer controled printer system by using a predetermined linear acceleration ramp in 50 steps to reach 7000 steps/sec in 15 msec. This was done in a closed loop mode with bilevel voltages of 28V and 48V and 24Ω suppression resistors. Indeed the linear scheme is effective for top speeds less than 7KHZ.

Plas and Blomnaert (7) split the torque-speed curve into zones and integrated the zones by allocating arbitrary time constants to the corresponding acceleration and deceleration exponential distributions. For speeds under 11KHZ they applied a time constant of T = 0.03S and for speeds up to 16KHZ they used T = 0.49S. This was implemented in H/W under the control of a minicomputer.

Rahmen (8) split the acceleration, slew and deceleration optimum curve into arbitrary breakpoints, the intervals of which are reasonably approximated by linear segments. The author considers this to be realistic especially from the point of view of computer control in high speed applications over 7KHZ. Like Plas and Blomnaert (ibid), Prasaad (23) used H/W consisting mainly of a BRM and separate counter sets under computer control to implement Rahmen's approximations on one axis.

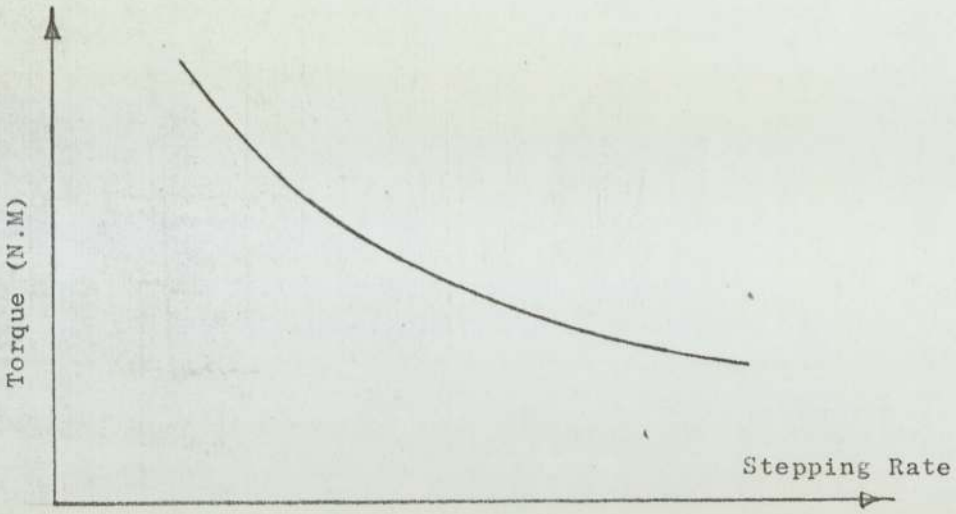
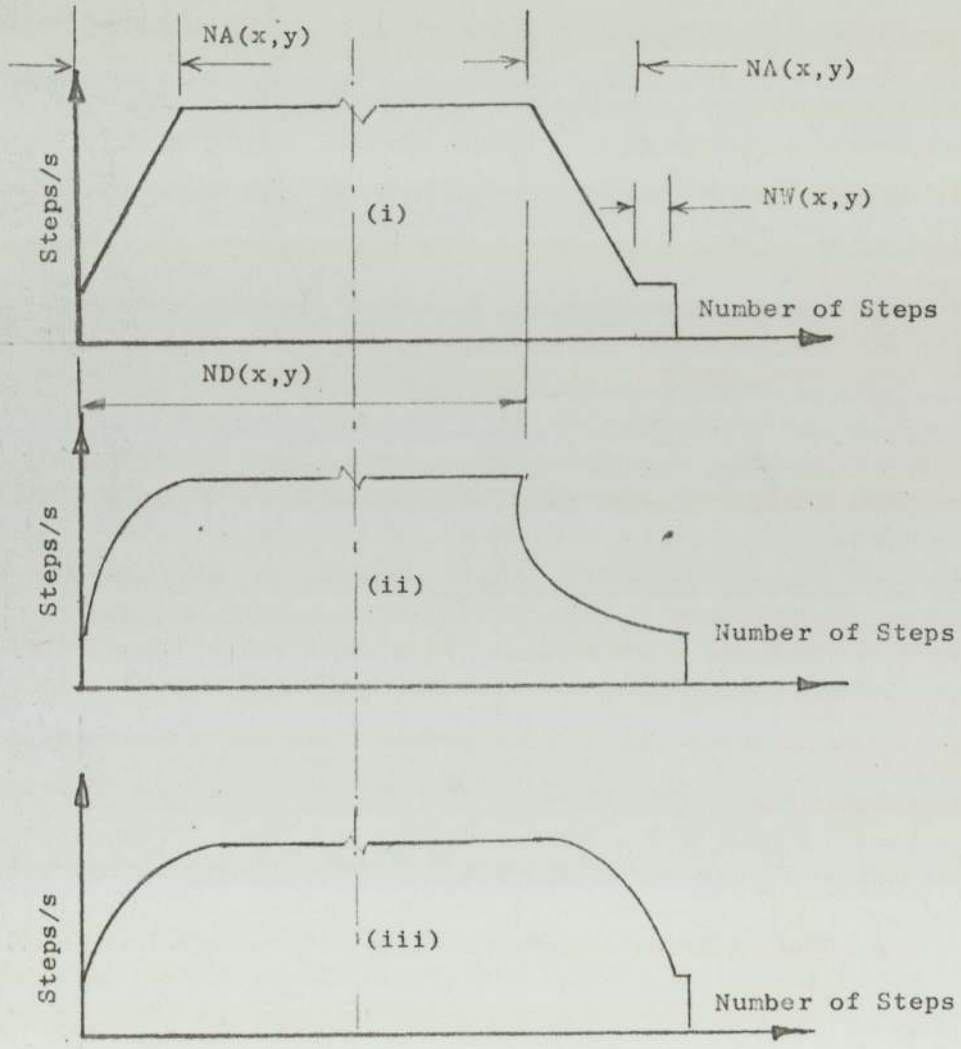


FIG 2.10 Torque-Speed Characteristic Curve of SM's



(i) Linear Acceleration/deceleratiom
(ii) Exponential Ramp up and down
(iii) Exponential ramp up and Inverse Exp. ramp down

FIG 2.11 SM Acceleration/Deceleration Schemes

2.5 The Electrohydraulic Stepping Motors and Design Considerations

The most important limitation of stepping motors in the low power which they are capable of producing. This problem is considerably solved in either of two ways:

(a) Construction of larger and more powerful stepping motors.

(b) Designing electrohydraulic stepping motors.

In electrohydraulic stepping motors a small electric stepping motor provides the actuation for a mechanical rotary-to-linear translator or a potentiometer with complementary terminal voltages. The mechanical or electrical signal set-up moves a servo-valve spool admitting hydraulic pressure into a hydraulic motor or cylinder. The unit is often simply purchased off-the-shelf as a package or, in the case of this study, coupled in-house with the functioning units which already exist. In both cases the operation is that of a torque-converter in which only a small torque is required from the stepping motor to overcome a small back pressure with the compact type of potentiometer friction in this electrically-coupled variety.

Bell et. al. (75,76,77) have given an extensive coverage of the state of the art in design with electrohydraulic stepping motors. In reference (75) and (77) they summarise the existing systems and published research in the area. On the basis of the general hydraulic-servo transfer function:

$$\frac{\theta o(s)}{\theta i(s)} = \frac{\omega_n^2 K_v / D_m}{S(S^2 + 2\varepsilon\omega_n S + \omega_n^2)} \qquad 2.10$$

a constant following error proportional to the stepping motor velocity and inversely proportioned to the hydraulic flow gain, K_v/D_m, is established at the loop summing point. They mentioned appropriately that with a stationary command the following error is zero but in practice a small valve overlap error exists. The author would add also that some error due to hydraulic motor friction and friction in the spool does exist too. The possible effects of this (and its correction) were not included in the paper. However, the researchers highlighted the important design considerations for stable and efficient functioning of the system (also ref. 76). These should include

(a) the acceleration/deceleration of the stepping motor

(b) the maximum slew speed which the hydraulic servo system can follow stably

(c) the maximum following error, which affects (b)

(d) the maximum hydraulic motor torque available

(e) the overall stiffness of the connected members

(f) the ability of the system to tolerate inertia loads.

In the design considerations the interaction of (a) → (f) will determine such NC requirements as resolutions, feed and traverse rates and dynamic performance. The precise design and manufacture of items like leadscrews and gearboxes determine to what extent error-sources arising from backlash and wind up are overcome.

Plas and Blommaert (7) point out the end-of-segment error in circular interpolation which results from the velocity following error and mentioned that it can be corrected in computer control.

Bajwa (78) was in tune with Bell et. al (75), though his analysis mainly featured the overall closed and open loop controls and the determinant for system loop gain.

In the area of electrohydraulic control systems for machine tools Ertongur (92) points out servo-valve hysteresis as being the main cause of the low frequency oscillations ('tweaks') in the drive and the variation of flow gain for small current inputs at the lower drive speeds. He also rightly blames coulomb friction and stiction in the hydraulic drives for contributing to the non-linearity in the lower speed end of the control system drive. He recommends appropriately the tightening of the specifications on maximum acceptable width of valve hysteresis by the manufacturers and confirmed that damping with 'by-pass' flow leakage in the valve block has considerable effect in reducing the oscillations. This is among the techniques used to stabilise the hydraulic motor and cylinder drives on the HPE machine used for this study (Chapter 3)

Further discussions on some general stepping motor control system design aspects will be made in the next section.

2.6 The selection of Electric Stepping Motors and the design Procedures

As has been stated one of the principal advantages of a stepping motor is that it can position and locate its output shaft without having to rely on feedbacks or transducers. Its ability to do this and extend it to external loads depends not only on the electronic controls (discussed earlier) but also on the capabilities of the motor and the effectiveness of the mechanical members. Section 2.5 summarised the design considerations for electrohydraulic stepping motor applications. With the exception of the hydraulic-servo aspects, those considerations also apply to direct drive electric stepping motor systems. The main difference is in the stepping motor torque or power requirements. The electric stepping motor must be able to provide all the necessary power to overcome friction and the range of loads for which the design is created. Although the required torque can be magnified by suitable gearing, the power is reduced in the ratio of efficiencies of the drive-transfer members. Therefore these members (gears, leadscrew and nut) should be carefully designed. Anti-backlash gearboxes and preloaded recirculating ballscrews and nut are recommended.

Often the 'quality' of the stated torques can be cross-checked with the Torque/inertia ratio. These can be above 20×10^3 for very good motors. It is necessary to assess the acceleration potential by comparing the shapes of the torque-speed characteristics under varying load conditions. The number of phases and the basic step angle must be considered. For high power applications (e.g. NC machine tools) multi-phase (5 or more phases) V.R. stepping motors are usually suitable.

Kordik and Senica (79) devote their paper to the comprehensive essentials for stepping motor selection. The paper included discussions for such simple features like dimensions, weight and environmental effects (temperature and heat dissipation). Considerations for accuracy, overshoots, damping and the effects of friction regarding the choice of step angle are also included. Other motor characteristics like torque, accelerations, load effects and maximum speeds are also analysed. The reasons for gearing, torque or inertia matching are fully considered.

Torque matching is necessary to enable the motor to develop enough output torque to drive its loads

$$T_O = T_A . N_G \qquad 2.11$$

T_A = available torque

T_o = output Torque

N_G = gear ratio

Load inertia must be matched to motor rotor inertia because the torque available to a stepping motor for a range of speeds is variable due to its inductance. They suggested (load inertia/rotor inertia) ratios under 10:1 (ideally 3:1) as tolerable depending on the nature of the torque/speed curve.

Superior Electric (80) provides another in-depth treatment of the subject of selection and design. They rightly included electronic drive logic capabilities. The sort of information the designer should expect from the manufacturers of the motor and the drive are listed. The author recommends this booklet to those contemplating designs with

stepping motors as a preliminary study.

Some details about the mechanical component (gears and couplings) design constitute the theme of a paper by Stevens (81). The gear ratio determines the machine resolution and have an effect on oscillation and damping. He was right to suggest multi-stepping (e.g. ½ steps) as a means not only to improve machine resoltuions but also to achieve low oscillations. He confirmed that the torque/inertia ratio of the stepping motor determines its acceleration capability but did not volunteer a figure on this ratio. He announced the availability of ministep logic drives capable of between ½ and 1/30 steps.

2.7 Interpolation and Feedrate Control in Stepping Motor CNC

The interpolation, feedrate and other control procedures in some existing SM computer control systems have been described in section 2.3.0 to 2.3.2. The control of interpolation and feedrate in these systems can be discussed broadly under three headings, namely:

(i) Integration Methods

(ii) Search-step Methods

(iii) Direct Calculation Methods

(i) The Integration Methods

These techniques include the pulse rate multipliers which are implemented as BRM integrated circuits and also the digital differential analysers (DDA's)

(a) The BRM technique'

In the BRM circuit, (fig. 2.13) a pulse frequency, f_o, is fed to the input of the BRM (SN7497) of bit length, n_o. Another source of acceleration pulses of frequency, f_{acc}, is input to an acceleration counter pre-loaded with a number, m_v which represents the number of pulses emerging at the BRM output for every set of (2^{n_o}) pulses from f_o. This means that m_v output-pulses can always be produced in a period defined by M input pulses, where $M = (2^{n_o})$. The output pulses are generated at times which are multiples of the period defined by f_o. The acceleration counter triggers the instants at which the variable number m_v, is loaded into the BRM and thus the number of output pulses for a

constant value, M, of input (f_o) pulses can be varied to give accelerations, decelerations or constant feedrates to the stepping motor.

The problem with the BRM is that the output pulses are not evenly spaced. To solve this problem another counter is interposed between the BRM output and the motor logic board to smooth (to some extent) the stepping pulses.

Therefore the resulting stepping rate at any instant is:

$$f = \frac{f_o}{(2^{n_o})} \cdot \frac{m_v}{N} \qquad 2.12$$

where f = stepping rate

N = number preset in the down counter.

Fig. 2.14 shows the BRM arrangement employed by Latham and Pelc (40) section 2.3.2.2.

(b) The digital differential Analyser (DDA)

DDA's are available today in IC form and consist mainly of Accumulators (R) and summing (integrand) registers (I). Such a device seeks to implement the flow shown in the schematic diagram, fig. 2.15a. The DDA is an accumulator scaler which is operated at an input pulse rate while the registers are loaded serially to obtain the desired proportions of the input pulse. The ouputs are the overflows (most

significant bits) of the integrand registers. The additions to the X and Y registers are controlled by the sequence of pulses which represent elemental changes in each axes. By cross-coupling the X and Y flows (fig. 2.15a) the changes in these axes are transformed into mutual differentials and so are used to generate the parametric equations of a circle:

$$X = \gamma \text{ Cos } \omega t.$$

$$Y = \gamma \text{ Sin } \omega t. \qquad ; \text{ where } \gamma = \text{radius}$$

Fig. 2.15b is a flow loop for S/W implementation of the DDA. Shumsherruddin (18, 88) describes CNC systems which employ the DDA circuitry and Richards (87) gives a theoretical appraisal of DDA's used in computer systems.

(ii) The Search-step Methods

These are step by step control techniques which have the advantage of requiring only additions, subtractions and shifting μC operations in the recursive (S/W) interpolation loop. They have the disadvantage of requiring the μC to be dedicated almost entirely to the interpolation process. The Bresenham algorithm (89) is the best example of the search-step method. It is applied for linear interpolation by issuing a pulse to the major (faster) axis and then to the minor (slower) axis only if the later step helps to reduce the error from the specified linear path.

In fig. 2.16 the line AB can be represented as:

$$Y = \left(\frac{m}{n}\right) . x \qquad 2.14$$

where $(m \leq n)$.

A discriminant, d_{i+1}, which determines whether a move along X and/or Y is performed next is formed thus:

$$d_{i+1} = n - m \qquad 2.15$$

The sign of the discriminant indicates the axis of motion, the subscript (i+1) indicates the initial movement.

The value of d is updated (addition or subtraction for Y and X respectively) after each move.

The sign of d determines the direction of the next move and the loop is executed until the path is completed. This technique has been applied by Martin and Dalzel (97) in their μP-based interface to a digital plotter which is controlled via a D/A converter (instead of stepping motors). They used the MC6800 μP to program a PIA LSI which then controls the D/A converter. Pen velocities of up to 5000 mm/s were reported using this μP.

The advantage of the use of the step-by-step methods in SM μP systems is that minimum H/W is required. However they have the disadvantage of increased S/W costs and a general limitation of reduced speed and excessive processor time used, especially when the SM is operated in a closed loop.

(iii) Direct Calculation Methods

The approximation of circular arcs with linear segments, as used in the MTIRA System (90) is an example of this technique. The length, L,

of a given part program path segment (linear) is approximated as follows:

$$L = /\max((x_f-x_o), (y_f-y_o)/ + \tfrac{1}{2}/ \min((x_f-x_o), (y_f-y_o))/ \qquad 2.16$$

where x_o, y_o = starting point coordinates of the segment.

x_f, y_f = end point coordinates

In the MTIRA system, the programmed feedrate, F, would then be implemented in terms of machine resolution (MR) units per path sampling period. For a sampling frequency, f_s, and machine resolution, R_m, the number of MR units per period is given by (equivalent to SM system)

$$R = \frac{F}{60} \cdot \frac{1}{f_s} \cdot R_m = \frac{F.R_m}{60 f_s} \qquad 2.17$$

where F = steps/min along the resultant path

R_m = in/step

f_s = HZ

Then the equivalent number of MR units per sampling period in each axis, R_x and R_y will be calculated to give the corresponding feedrates in the axes F_x and F_y respectively. Generally

$$F_x = \frac{x_f-x_o}{L} \cdot F \qquad 2.18a$$

$$F_y = \frac{y_f-y_o}{L} \cdot F \qquad 2.18b$$

For circular interpolation, the incremented segment coordinates can be calculated using the Bergren equation (91), equations B.4 and B.7, Appendix B.

The sine-cosine table described by de Barros ref (86) is another example of a calculation method for circular interpolation control. The incremental coordinates along the circular arc are calculated using the tabulated values of the sines and cosines of the interpolation angle,

$$x = \gamma \text{Cos } \theta$$

$$y = \gamma \text{Sin } \theta \qquad 2.19$$

2.7.1 Positioning servomechanisms in CNC Systems (figs 2.17—2.19)

NC positioning servomechanisms are of two main types:
(i) hard servos and (ii) soft servos

(ii) Soft servomechanisms are characterised by low position loop gains, K_p, and higher inner velocity loop gains, V_L

(i) Hard servomechanisms. Unlike the soft servo the hard servos possess high position loop gains and lower velocity loop gains. With their high position loop gains the hard servos have a fast transient response to position commands. The electrohydraulic servo drives (chapter 3) are characterised by fast transient responses, being typical hard servomechanisms. The transfer functions for the electrohydraulic drives used in this study are analysed in chapter 3.

Popa (20) concludes that a digital servo with sampled position loop is preferable to the continuous position loop owing to the increased flexibility with which interpolation and feedrate algorithms can be implemented by S/W. Although this discussion is outside the scope of this study, an analysis of the transfer function of a sampled position

loop servo developed by Middleditch (96) for the Westingshouse CNC is shown below:

The closed loop transfer function (fig. 2.19) is given by:

$$G_{(s)} = \frac{\theta_o(s)}{\theta_i(s)} = \frac{k_p k_v / (1+k_v)}{1 + \dfrac{k_p k_v / (1+k_v)}{s\,(1 + ST)}} \qquad 2.20$$

where K_p = equivalent open loop position gain

K_v = open loop velocity gain

T = velocity loop time constant.

(the value of T is dependent on the electronic delays in the velocity controller, the axis driver and the response time of the tacho-generator, (fig. 2.18).

If the transient position following error = $e_p(t)$

Then,

$$e_p(s) = \theta_i(s) - \theta_o(s) \qquad 2.21$$

From 2.20, $\theta_o(s) = G_{(s)} \cdot \theta_i(s)$

$\therefore$ In 2.21, $e_p(s) = \theta_i(s) - G_{(s)} \cdot \theta_i(s)$

ie, $$e_p(s) = \theta_i(s)\,(1 - G_{(s)}) \qquad 2.22$$

During acceleration the transient feedrate, F, represents a position ramp input signal (especially in a SM system).

Therefore $\theta_i(t) = F(t)$ 2.23

ie. $\theta_i(s) = \frac{F}{S^2}$ 2.24

So that eqn. 2.22 becomes

$$e_p(s) = \frac{F}{S^2}\left(1 - \frac{k_p k_v / (1+k_v)}{s(1+sT) + k_p k_v / (1+k_v)}\right)$$

$$= \frac{F}{S^2}\left(\frac{S(1+sT) + k_p k_v/(1+k_v) - k_p k_v/(1+k_v)}{S(1+sT) + k_p k_v/(1+k_v)}\right)$$

$$= \frac{F}{S^2}\left(\frac{S(1+ST)}{S(1+sT) + k_p k_v/(1+k_v)}\right) \qquad 2.25$$

Applying the Final Value Theorem to the transient value of position following error, the steady state value becomes

$$e_p = \lim_{S \to 0}(e_p(s)) \qquad 2.26$$

From 2.25 and 2.26

$$e_p = \frac{F}{k_p k_v / (1+k_v)} \qquad 2.27$$

For the soft servomechanisms (Middleditch (96)), k_v has a high value

Therefore, $k_v \approx (1+k_v)$

From 2.27, $e_p \approx \frac{F}{k_p}$ 2.28

The maximum following error is obtained at the programmed feedrate F_o.

Therefore $e_{p_{max}} = \frac{F_o}{k_p}$ 2.29

From the value in 2.29, Middleditch rightly concludes that the steady state position error of the sampled system is independent of the sampling rate and is equivalent to that of the continuous position control system. Hence, any under- or overshoot in the controlled contour is entirely dependent on the speed of position response in the control system H/W.

For closed loop μC NC systems the value of $e_{p_{max}}$ in equation 2.29 determines the bit length of the counter for feeding the digital position informtion periodically back to the μC. At every sampling period the digital value of the postion following error is tranfered to an up/down counter of bit-length, n.

$$e_{p_{max}} = \frac{F_o}{k_p} = 2^{n-1} R_m \qquad 2.30$$

where R_m = machine resolution unit (in/step). The counter output controls a D/A converter the output of which is a proportional d.c. voltage. This voltage controls the speed of the d.c. motor or the pulse frequency of a voltage controlled oscillator (VCO) which drives a stepping motor.

This study, however, is concerned with the open loop drive of SM's using μC programmable LSI's (chapters 5—7) to position the new electrohydraulic feed drives fitted to the milling machine. In the next section a discussion of the use of programmable LSI's will be made.

2.8 The Use of Programmable LSI's in μC Systems

Besides the μP LSI, other programmable LSI's available today can be placed into two broad headings: 'general' and 'special purpose'. As the term implies the 'special' purpose LSI's are designed for stipulated applications while the general purpose LSI's are designed for more flexibility in I/O control systems.

Custom LSI's are the best contemporary examples of the special purpose devices. According to ERA (19), Custom LSI's become very competitive at production quantities between 40,000 and 100,000 but it is pointed out that this range is an artificial break-even criterion, largely imposed by manufacturers and 'the availability of design effort'. Page (9) puts the break-even point between Custom LSI and a μP-based system in the range of 10,000 - 20,000 and estimates that this figure should remain approximately constant as micro-technology develops because μP's are LSI's themselves. In ref (82) it is stated that an annual production volume of 35,000 units may be required to make the custom LSI cost-effective. It went on to say that a custom LSI is to be prefered to a μP where the function is I/O intensive. A typical example of this would be open loop interpolation pulse generation to a stepping motor for which the use of a dedicated μP would afford more control flexibility but much reduced speeds and increased S/W development costs.

Usually the combination of a μC and an LSI to form a CNC system offers a suitable compromise solution. Weston and Charles (48) employed a TMS9900 μC to control SM's via a general purpose LSI (PIA) which was only serviced periodically in the sampled data CNC for cutter

grinding. Martin and Dalzel (97), section 2.7, also used the μP (MC6800) and PIA combination in their graph plotter. Prassad (23) and Popa (20) include the PIA, MC6280, with their MC6800 μC in the design of their multicomputer system using BRMS for their SM drives. The interpolation control, however, is implemented in computer S/W with the attendant extra development costs. Young and Brignel (95) state that they combined the speed offered by LSI H/W with the flexibility of the μP in their 'fast programmable interface convertor'. They employed the 16-bit CP1600 μC supported by interface LSI's, PIC1640 and IOB1680, which they described as 'microprocessors in their own right'.

Indeed the two main philosophies in the design of μP-based NC systems are those of minimal H/W and the autonomous interface μC. The first approach involves an intensive S/W implementation of the system and a reduced number of H/W components required. Since such a system does not usually involve the use of interface LSI's it has low variable production costs (once the S/W is developed) and the cost of H/W development is virtually constant from one unit to the other. While offering flexibility in design modifications it has the important handicap of excessive cost of S/W development and reduced processing speeds.

The second design approach trades the increase in H/W by way of LSI's for the decrease in the more important S/W costs. Although it has less flexibility in design modification, it offers a quicker and cheaper solution to a CNC design problem. The author agrees with Young and Brignel (ibid) in their assertion that a compromise solution is to combine the power of the interface set with the programmability of the μP.

In the introduction to this study (chapter 1), mention was made of Page (9) advocating the use of custom LSI's as an avenue to reduce the costs and high expertise necessary to implement a μP-based system. He states further that custom LSI's are offering a serious challenge to μP's especially in high volume production. Chapters 5→7 describe three different LSI's and their variety of virtues and problems, and how they can be used to generate interpolation and feedrate pulses to drive SM's in an open loop general purpose μC system. It is true that this class of control systems does not have as high a performance as the more complex (S/W closed loop) and of course more expensive systems. The objective here is a cost-effective system for a small firm using a general purpose μC. The next chapter describes the design philosophy of the new feed-drive mechanisms on the machine.

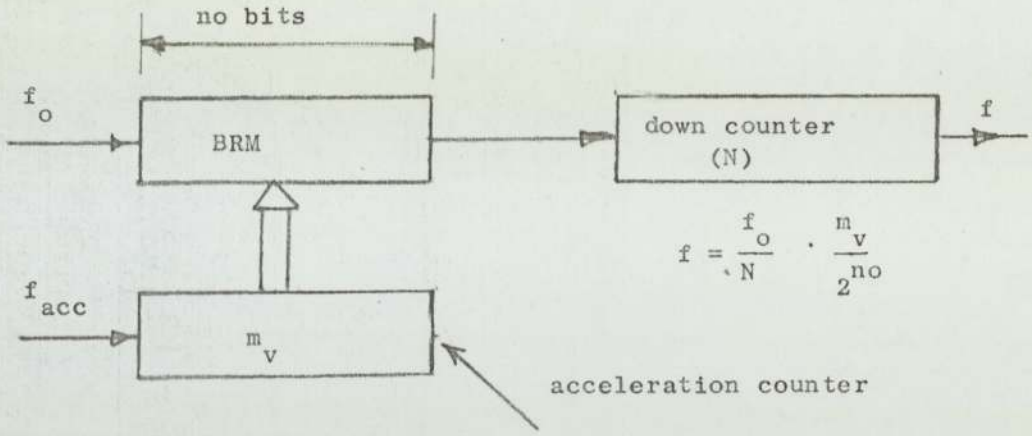


FIG 2.13 ONE Axis Acceleration BRM

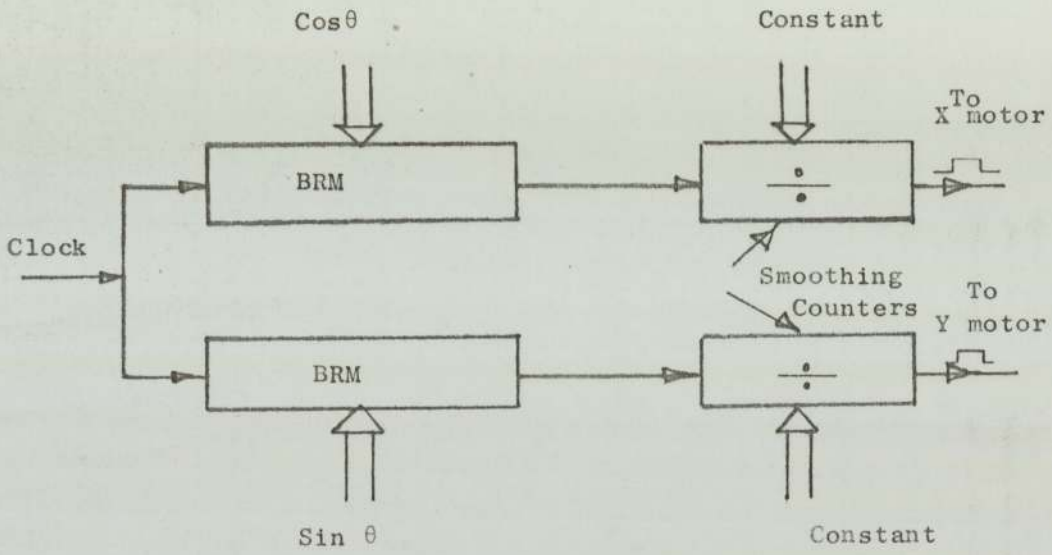


FIG 2.14 BRM's Loaded with Sines and Cosines

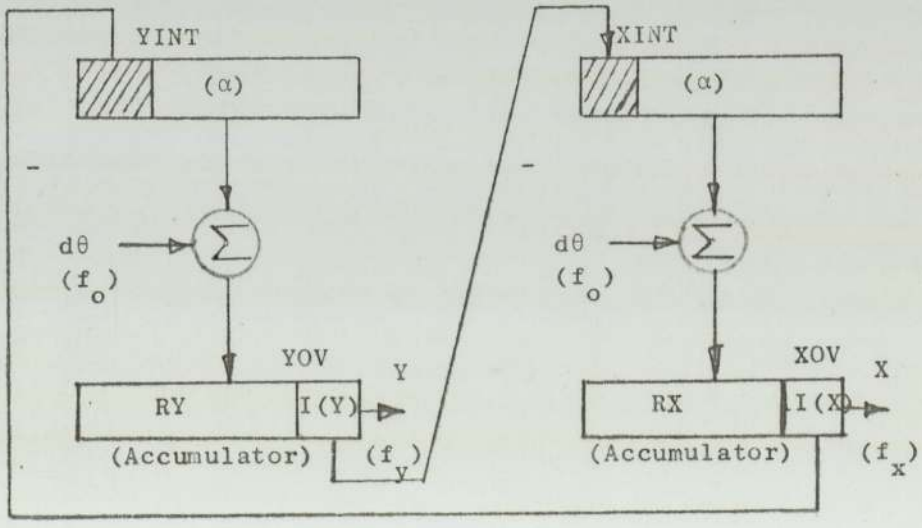


FIG 2.15a DDA for Circular Interpolation

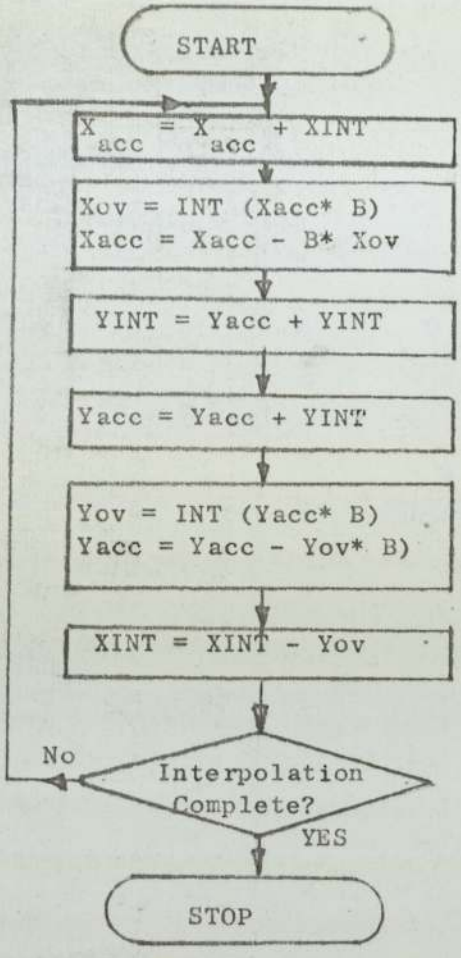


FIG 2.15b DDA Algorith Flowchart

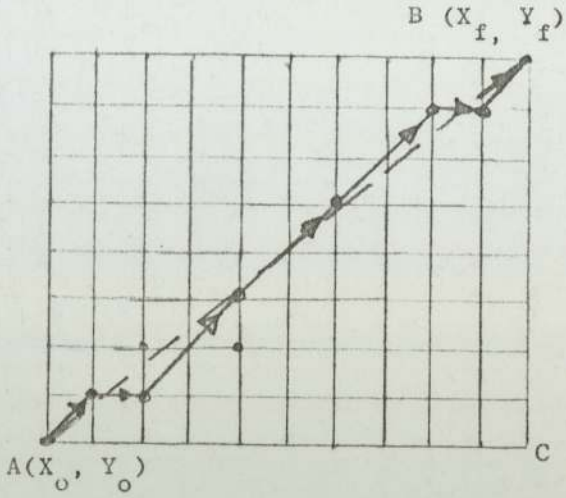


FIG 2.16 Bresenham Algorith

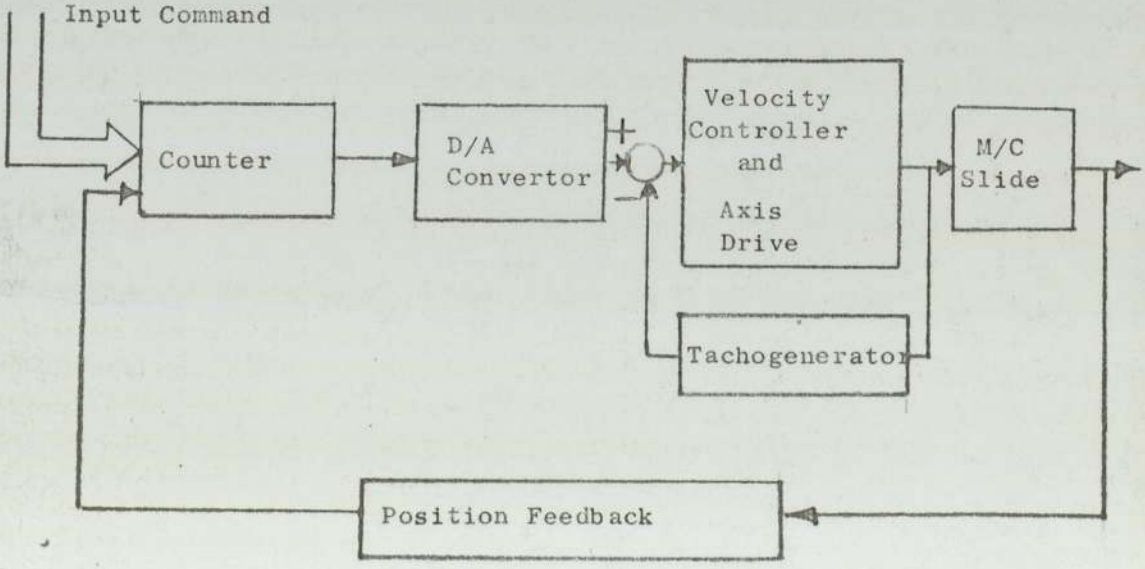


FIG. 2.17 Control System with Hardware Closed Loop

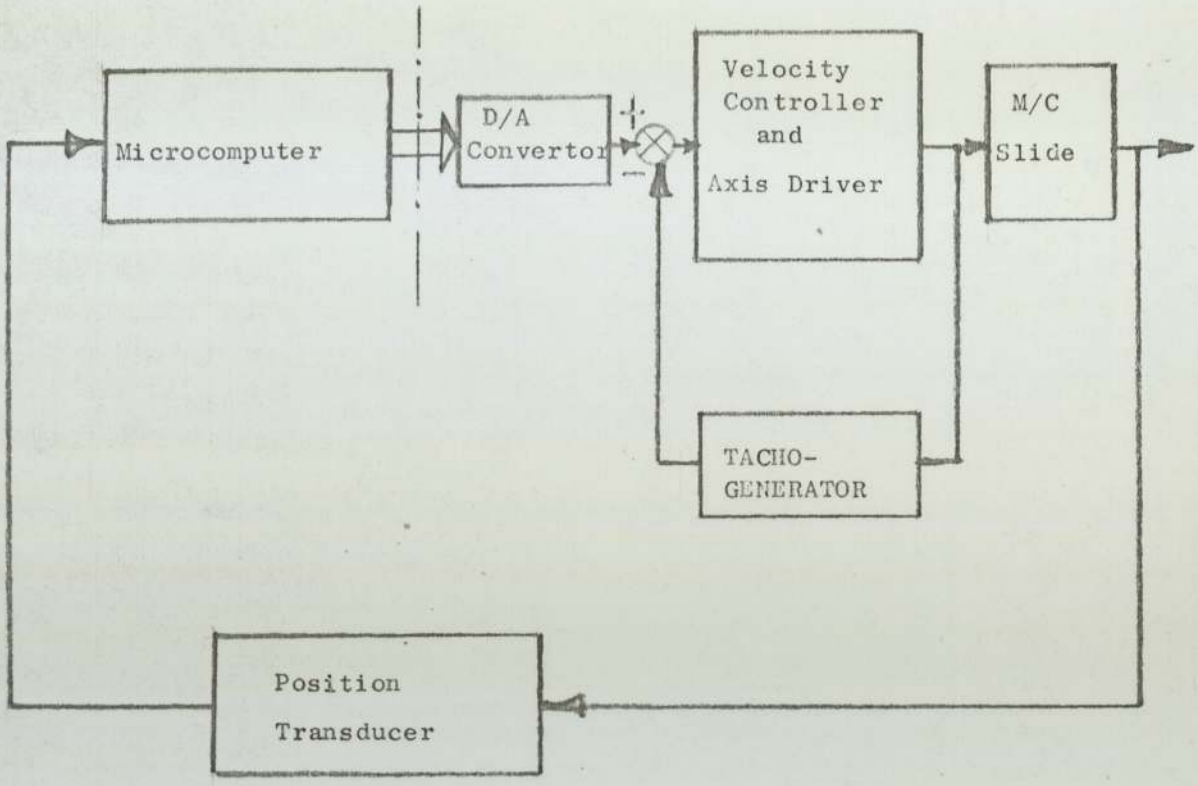


FIG. 2.18 Control System with position loop closed inside computer

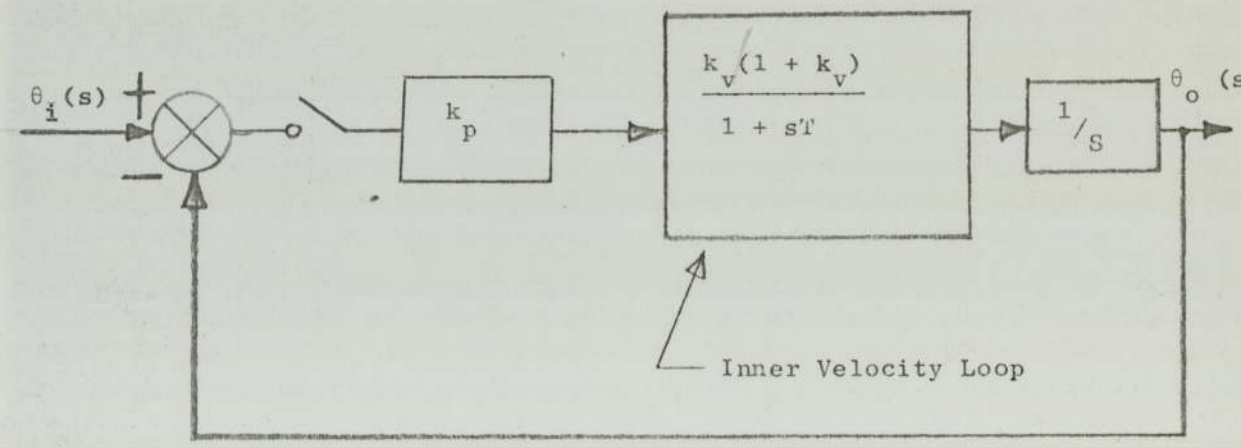


FIG 2.19 Sampled position Data closed loop

(Transfer function elements for fig 2.18)

CHAPTER 3

THE ELECTROMECHANICAL (STEPPING MOTOR - MACHINE) SERVO SYSTEM

3.0 Introduction

Thousands of NC machines have become out-dated because of the development of more versatile and economic control systems. In this age of inflation and stiff competition in the manufacturing industry, the small- or even medium-sized companies often cannot afford to scrap these machines which in themselves could remain productive. Therefore, as was mentioned in Chapter 1, the redesigning or updating of these NC machines with retrofits has become a growing industry in its own right.

Any redesigning proceedures should include reassessments of the precision and adequacy of the various members (gearboxes, leadscrews) on the existing machine. Because the NC machine is an expensive item of equipment it is often employed in double-shift operations and so it is possible that certain mechanical members may have to be replaced. This is not necessarily the case always and the machine used in this study was found to be in good order.

3.1 The existing machine

This is an 'High Precision Equipment' (HPE) milling machine which was fitted with the Ferranti 'Mark IV' 3-axes control system. Magnetic tapes recorded by a computer were the sources of signal generation and their cost was rarely justifiable for small batches and frequent 'one-offs'. This, together with the added complication of the time taken for the preparation of tapes by outside bodies (already discussed in Chapter 1) meant that the machine had not been much used since its purchase in 1964.

The design of the machine is based on a fixed bed layout with vertical milling head. The spindle and slide movements are hydraulically powered. A fixed displacement pump supplies the circuit for all three axis movements (fig. 3.1), and a variable displacement pump drives the spindle providing a means of speed control.

The X-axis motion is achieved by an hydraulic motor driving a recirculating ball-screw through a 50:1 reduction gearbox. This traverse axis of 48 inches is the longest and its slideways are integral with the fixed machine table. These slideways carry the spindle assembly and Z and Y axis rams on a traverse beam structure. The Z ram forms an integral part of the quill assembly and the Y axis slide (carrying the Z-axis) is a cast block assembly having a cylinder housing for traverse movements.

The Y axis motion is on Dexter linear recirculating ball bearings on the top and bottom surfaces of the cast beam unit. The quill mounting casting incorporates the Y cylinder housing which holds a centrally mounted piston rod attached to the frame at both ends. Flow of

oil to the cylinder produces the motion of the quill and cutter head assembly. Total traverse is 12 inches.

The cartridge-type spindle-quill assembly fits into a bored location in the quill housing and is rotated by a hydraulic motor through a stepped V-drive. The rotation of the cutter is obtained through a splined upper extension of the spindle rotated by a sleeve dowelled to the V-pulley on the top. Vertical traverse of the Z-axis (6 ins. max) is obtained by admitting hydraulic oil to the appropriate side of the Z-axis cylinder (which is fixed to the quill) and this moves the quill splines axially within the sleeve. The spindle rotates within the quill assembly and is carried in high precision bearings. The whole assembly is fixed to the cutterhead casting which is tenoned to the Y axis slide.

Each axis is actuated by metering oil through a Moog Valve. The velocity in the axis is proportional to the rate of oil flow which is controlled by the driving signals from the principal control actuator (these will be the stepping motors in the new systems). The maximum traverse in each axis is limited by the available hydraulic power to under 15in/min.

3.1.1 The mechanical components and the new design

The leadscrew is axially located by thrust bearings at both ends, just beyond the X axis 48 inches working range. At the time of assembly the screw is pre-tensioned to about 500 lbf (227kgf) and this corresponds to a 0.0006 in (0.0152 mm) extension.

The 50:1 reduction gearbox ensures that the torque available is sufficient to start motion from rest and is less than 20 lb in. The connection between the leadscrew and the gearbox shaft is a flexible disc-coupling which permits a limited degree of axial misalignment between them without any loss of rotational movement. With these specifications the original control system could offer co-ordinate accuracies of ±0.0005 in (±0.0127 mm) over the 48 inch working range.

The same accuracies were also stated for the ram drives in the other two axes. The mechanical elements, spindle drive and quill, slideways, gearing and leadscrew assembly had all suffered negligible wear and so it was justifiable to fit a new control system to extend the machines period of use. No changes were made on the specifications of the existing mechanical components. It is feasible to design the new economic control system with small electric stepping motors acting only as control actuators for speed and position using the existing hydraulic power. (A user who wishes to boost the power on the axes by using a high power electric stepping motor should review the specifications of the mechanical members).

3.2 The New Control Servo System of the X-axis

Figures 3.2a and 3.2b show the electromechanical servo loops for the X-axis. The manual control, servo amplifier, machine interface and stepping motor drive logic boards are included in the 'drive and control circuits' block (refered to in section 3.6). The manual control unit is necessary for tests to determine the accuracy of the leadscrew/gearbox assemblies and the machine resolution and its dynamic behaviour.

The rotary potentiometer which is driven directly by the small electric stepping motor (section 3.5) is a very precise unit with 10kΩ total resistance and a high linearity of ±0.2%. The miniature gearbox in the feedback path is an anti-backlash precision unit and provides a reduction of 8:1. In the earlier control system, position sensing in all axes was by diffraction grating and scanning motor as demanded by the original Ferranti system. In the case of the X-axis there was an additional velocity signal from a tachogenerator at the hydraulic motor shaft. Spring-loaded anti-backlash spur gears were used to connect this hydraulic motor shaft to the tachogenerator shaft. In the new control system this tachogenerator was removed and the shaft retained to provide the feedback of the hydraulic motor motion to the miniature gearbox. The current amplifier is shown in fig. 3.3 and is identical for X, Y and Z axes.

During this study the building of the hardware components on the X and Y axes was completed, installed and tested with a manual control 'black box' built by the author earlier, (84) . The following sections will describe the design and testing procedures.

3.2.1 The calculations of the steady state stability characteristics (figs 3.2a & b)

Symbols used:

ω_h = hydraulic natural frequency

ξ = damping coefficent

R = miniature gearbox reduction

K_q = servovalve flow gain

d_m = hydraulic motor displacement

θ_i = input angle

θ_o = output angle

G_1 = potentiometer 'gain'

G_2 = servo amplifier gain

G_3 = servo valve - hydraulic motor (or cylinder) gain

V = voltage

$\alpha = 1/\omega_h^2$

$\beta = 2\xi/\omega_h$

ω = rotational speed of stepping motor

K_L = system leakage coefficient

f = viscous friction coefficient

From fig. 3.2b the overall transfer function can be shown to be:

$$\frac{\theta o}{\theta i} = \frac{G_1 G_2 K}{\alpha S^3 + \beta S^2 + S + G_1 G_2 K} \qquad 3.1$$

where $K = \left(\frac{K_q \cdot d_m}{d_m^2 + K_L f}\right)$ 3.2

(The values of the basic constants are given in table 3.0).

Equations (3.1) and (3.2) serve as a basis for selecting maximum possible overall loop gain ($G_1 G_2 K$) in sec^{-1}. Knowing the available miniature gear reduction, R, the rated flow gain of the servo valve, and the values of d_m and $K_L f$ from motor and valve manufacuturers specifications respectively, K can be calculated. For the precision rotary (360) potentiometer $\pm$15V are connected to the terminals. Therefore $G_1 = \frac{30}{2\pi}$ (V/rad)

It can be shown also by Rouths stability criterion that

$$(G_1 G_2 K)_{max} = 2\varepsilon\omega_h \quad (= \frac{\beta}{\alpha}) \qquad 3.3$$

from fig 3.2b and equations (3.1) and 3.2),
(ω_h is deduced from the Physical specifications of the hydraulic motor). With this maximum value of overall loop gain (ie. $G_1 G_2 K$) the maximum setting of the amplifier gain was found and hence the maximum stepping motor speed. These were found to be 37mA/V and 375 rev/min.

The maximum possible amplifier setting was used so as to give the fastest system response. This is desirable:

(1) to achieve some motions in the shortest time period consistent with the pulse rate.

(ii) to minimise the lag between the electrical stepping motor and the hydraulic actuators. This reduces the velocity following error and would avoid the significant end of segment errors in circular interpolation mentioned by Plas and Blommaert (7).

The 375 rev/min of the stepping motor gives the maximum axis traverse and is equal to that originally specified for the HPE machine, 15in/min.

Table 3.0

List of constants for x and y axes

Constant	Units	x	y
Hydraulic natural frequency	rad/s	422	494
Damping coefficient	-	0.32	0.17
Miniature gearbox reduction	-	-8	+5
Servo valve flow gain	in^3/s mA	0.5625	0.3125
Hydraulic motor displacement	in^3/rad	.044	-
Potentiometer 'gain' for x,y	(v/rad; & (v/in, resp.	4.9	30.
Servo amplifier gain	mA/V	37	107
Voltage	V	$\pm$ 15	$\pm$ 15
System leakage coefficient	$(in^3/s)/(lb/in^2)$	6.35×10^{-4}	1.36×10^{-2}
Viscous friction coefficient	(lbf/rad/s; & (lbf/in/s.	0.19	30
Leadscrew pitch	in/rev	0.25	0.20
Mass carried by y axis	lb		625
System pressure	lbf/in^2	1750	1750
Axis travel length	in	48	12
y cylinder cross-sectional area	in^2	-	6
Volume of cylinder	in^3	-	36
Bulk modulus of oil	lbf/in^2	2×10^5	2×10^5
Machine resolution	in/step	.0001	.0001

3.3 The Steady state driving error between the Stepping Motors and the hydraulic following members

The flow through the servo valve is directly proportional to the lag between the electric stepping motor and the hydraulic drive. This lag or driving error can be measured as a certain number of steps or driving pulses.

Treating the step inputs as unit ramp functions it can be shown by the Final Value Theorem that the error

$$E(s) = \frac{S\left(\frac{\omega}{S^2}\right)}{1 + \frac{G_1 G_2 K}{S(\alpha S^2 + \beta S + 1)}} \qquad \text{Symbols as in 3.2.1} \qquad 3.4$$

$$= \frac{\frac{\omega}{S(G_1 G_2 K)}}{S + \frac{}{S(\alpha S^2 + \beta S + 1)}} \qquad 3.5$$

Therefore, in the limit

$$\lim_{S \to 0} E(S) = \frac{\omega}{G_1 G_2 K} \qquad 3.6$$

Where ω = velocity at input = maximum necessary stepping motor speed of 375 rev/min = 39.27 rad/s. The value of $G_1 G_2 K$ (for axis, X) has been found to be = 273 S^{-1}. Therefore the maximum driving error is established as $E_{max} = \frac{39.27 \text{ (rad/s)}}{273 \ (S^{-1})}$

$\therefore$ E_{max} = 0.14385 rad at steady state

For a half step angle of 0.9 , the maximum (traverse) steady state error is equivalent to

$$\frac{(0.14385 \text{rad})}{\frac{0.9}{180}\pi} \quad \text{half steps (pulses)}$$

$$= 9.15 \text{ steps} \simeq 9 \text{ steps (pulses)}$$

This is a path error of 0.0009 in (machine resolution = 0.0001 in/step) and this is at traverse speed. It follows from equation 3.6 that this error will be proportionally less for normal machining feedrates. The equivalent values for Y and Z axes were found to be comparable with that of the X-axes.

As would be expected of a Type 1 servo system, the steady state velocity following error is finite and increased servo amplifier gains corresponding to higher rated servo systems gains would render it negligible. A change of this nature would make it possible for the stepping motor to start and stop at higher speeds with perfect control. This possibility was not pursued as it would have required the purchase of higher rated valves and a larger pump and electric motor.

The loop is closed, as already explained in section 3.2, through the miniature gear reducer driven from the original tachogenerator shaft to the potentiometer. This has the mechanical capability of 360^{o} rotation and is an essential feature to avoid damage should any malfunction occur. It also permits axis traverse by servo valve offset without the use of the stepping motor.

3.3.1 The effect of the stiffness of attachments to the closed loop servo structure

Although the influence of the combined elasticity of the various mechanical attachments in the servo loop have not been quantitatively considered in the calculations of the maximum loop gain and the resulting maximum following error, it is appreciated that there is an effect on the dynamic performance of the servo system. The material and size of connecting items like shafts, couplings and gears determine the overall stiffness of the mechanical unit.

In the servo loop, let

F_o = hydraulic stiffness of the servo valve and hydraulic motor assembly

ω_o = the corresponding hydraulic natural frequency

M = total mass of moving attachments in the servo drive

It can be shown that generally $\omega_o = \left(\frac{F_o}{M}\right)^{\frac{1}{2}}$.

Then,

$$\frac{1}{\omega_o^2} = \frac{M}{F_o} \qquad 3.7$$

If F is the stiffness of the mechanical attachments, and ω the overall system natural frequency, then

$$\frac{1}{\omega^2} = \frac{M}{F_o} + \frac{M}{F} \qquad 3.8$$

From (3.7) and (3.8)

$$\frac{1}{\omega} = \frac{1}{\omega_o}(1+F_o/F)^{\frac{1}{2}} \qquad 3.9$$

$$\omega = \omega_o \cdot 1/(1+F_o/F)^{\frac{1}{2}} \qquad 3.10$$

Equation (3.10) shows that the natural frequency can be significantly reduced if the ratio $\frac{F_o}{F}$ is high. Equation (3.3) shows that the maximum system gain is proportional to the natural frequency. Therefore, a high value of (F_o/F) can lower the value of the system gain, increase the response time and the resulting velocity following error, ultimately affects the accuracy of NC contouring operations.

The effect of F_o/F should be considered from the initial design stage when the material and size of components are selected. The original machine components were well conceived and with the added items minimum compliance was introduced, e.g.silver steel for the feedback shaft and high tensile steel precision ballscrews for the X and Y axes, assembled with minimum length couplings.

It has been correctly pointed out by Ertongur (92) that apart from the adverse effects of valve hysteresis on electrohydraulic system stability, backlash in the system also encourages the low frequency oscillations ('tweak') which cause system instability. Therefore, in the present system the gearboxes used are the anti backlash types and the leadscrew are of the preloaded re-circulating ball nut variety. All these help to increase the accuracy and stability of the whole servo system.

3.4 The New Control Servo System of the Y (and Z) axis

Figures 3.4a and 3.4b are the electromechanical servo loops for the Y-axis and figure 3.5 shows the servo loop for the Z-axis. The drive and control block is as described for the X-axis. The stepping motors in all the axes are identical (section 3.5).

For the Y (and Z) axes the miniature gear boxes have specifications identical to that in the X axis except that they offer a reduction of 5:1 of the input (stepping motor) speed. The outputs of these gear reducers drive the ballscrews and nuts to offer a machine resolution of 0.0001 in/step. In both cases a precision linear potentiometer provides the summing point of the servo valve controlled axis-positioning cylinder and the necessary feedback signals. The error voltage is provided by the wiper arm of the potentiometer the terminals of which are connected to ±15V complementary voltages.

3.4.1 Calculations of the steady state stability characteristics

The symbols listed under section 3.2.1 also apply but the flow gains relate to the cylinder-servo valve assembly. From figure 3.4b the overall transfer function can be derived:

$$\frac{y_o}{y_i} = \frac{G_1 G_2 k}{S\left(\frac{1}{\omega_h^2} S^2 + \frac{2\xi}{\omega_h} S + 1\right) + G_1 G_2 k} \qquad 3.11$$

where $k = \dfrac{K_q . A}{A^2 + K_L f}$

and A = exposed cross-sectional area of piston.

The value of the maximum overall loop gain, (G_1G_2K), and hence the maximum amplifier gain for optimum stable operation are calculated following procedures similar to those of the X axis.

i.e. $2\xi\omega_h = (G_1G_2K)_{max}$ 3.3

It can be shown, also that the hydraulic natural frequency of the cylinder assembly, ω_h, is given by

$$\omega_h = \left\{\frac{2B(A^2 + K_L f)}{VM}\right\}^{\frac{1}{2}} \qquad 3.12$$

where B = Bulk modulus of the oil

M = Mass of load positioned by the cylinder

V = $\frac{1}{2}$ (volume of oil in the cylinder)

The necessary design parameters are therefore found for Y and Z, as for X axis.

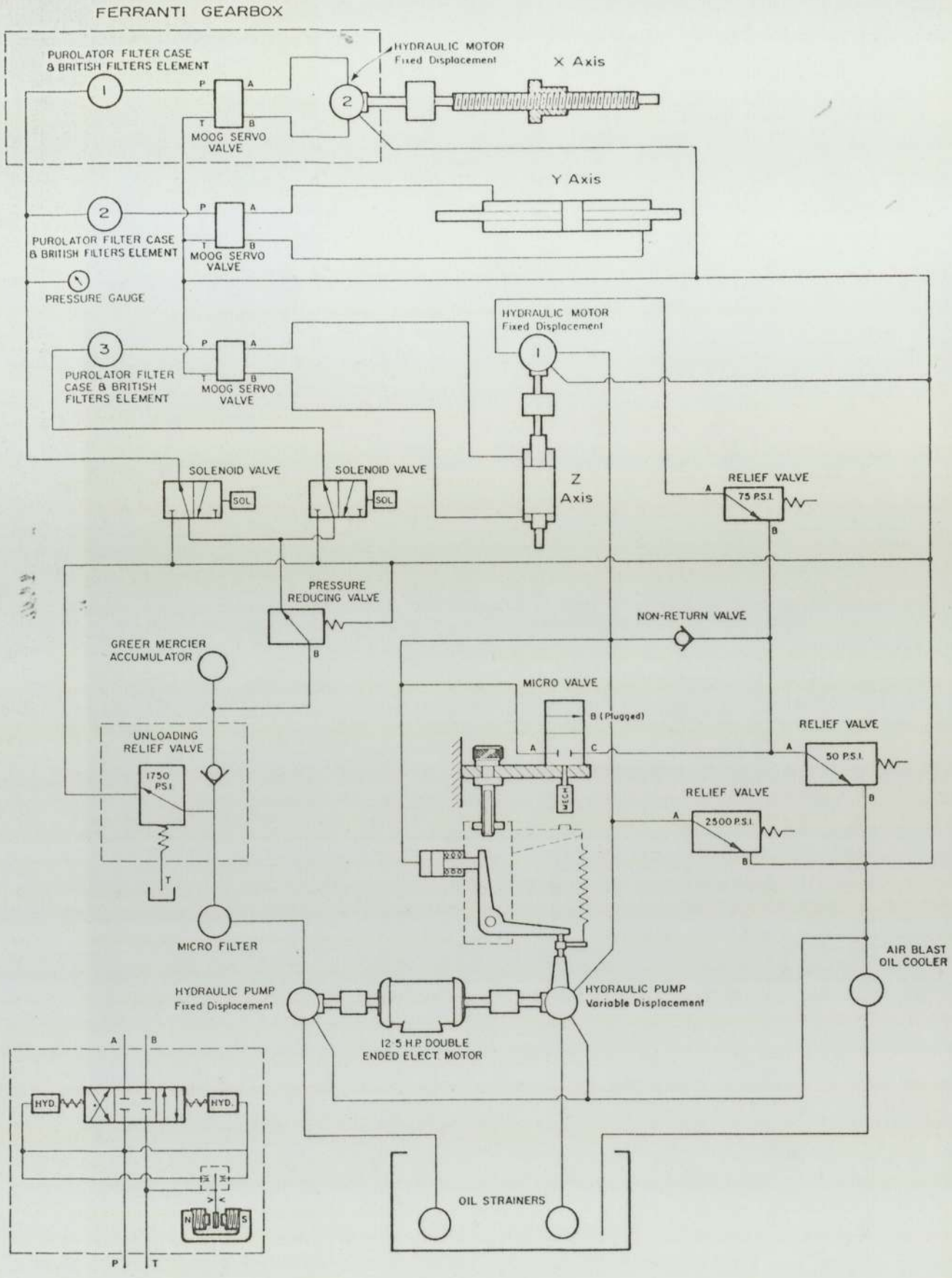
The stability of electrohydraulic drives used for axis positioning in machine tools can pose problems arising from the effects of Coulomb friction, stiction and valve hysteresis. This had been the subject of several papers in the past and the author would recommend Parnaby (85), part of the annual conference of the 'British hydro-mechanical Research Association' and the exposition by Ertongur (92).

On the existing miling machine the traverse on the Y and Z axes were limited to 12 and 6 inches respectively and control by-pass leakages are incorporated in the valve-cylinder assemblies by the manufacturers. These help to provide a value of damping coefficient, ξ, to ensure stable operation and yet maintain adequate response.

The use of pressure feedbacks between both sides of the piston may also be considered if instability remains a problem. Honeywell Ltd. Illinois, makes solid-state pressure transducers ranging from simple micro-switch-types to those suitable for higher pressure applications.

Most modern machine tools do not make use of hydrualic drives because of their inferior efficiences in comparison with d.c. motors or direct electric stepping motors. However, companies like 'Bridgport' still manufacture automatic hydraulic copying machines, and hydraulic power is often used also in several machining centres for tool indexing or Robot actuation. Research in the application of hydraulic drives and actuation will continue.

The following sections and chapters concerning stepping motors are relevant to systems using hydraulic and direct drives.



Details of Servo Valve.

THE HYDRAULIC SYSTEM

Figure 3·1

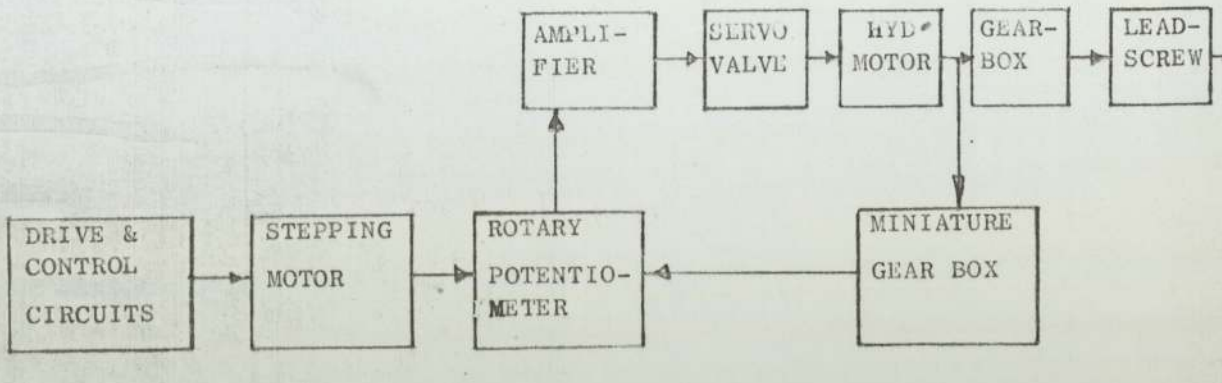


FIG 3.2a X-Axis New Servo System

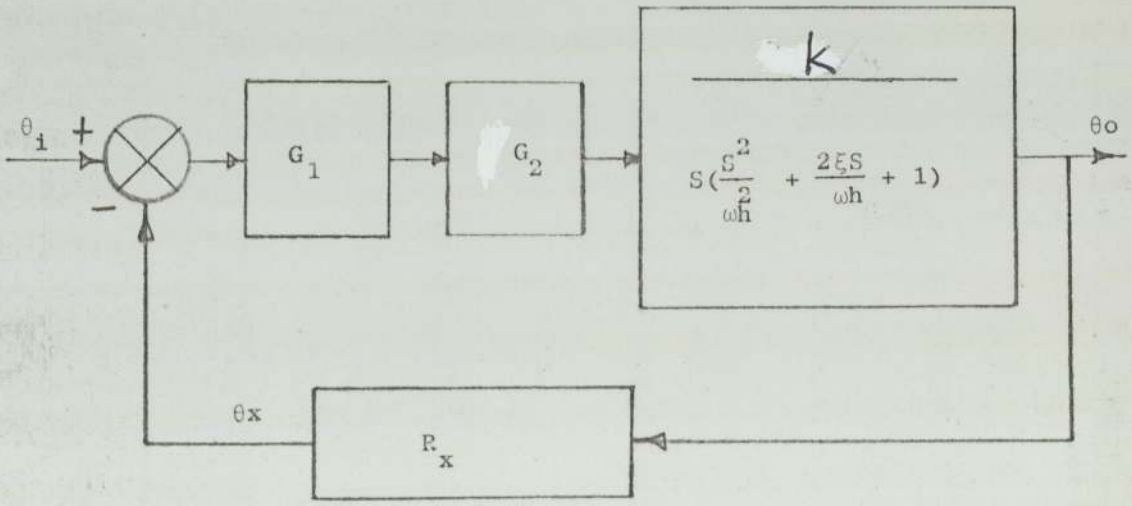


FIG 3.2b The X-Axis Servo Loop

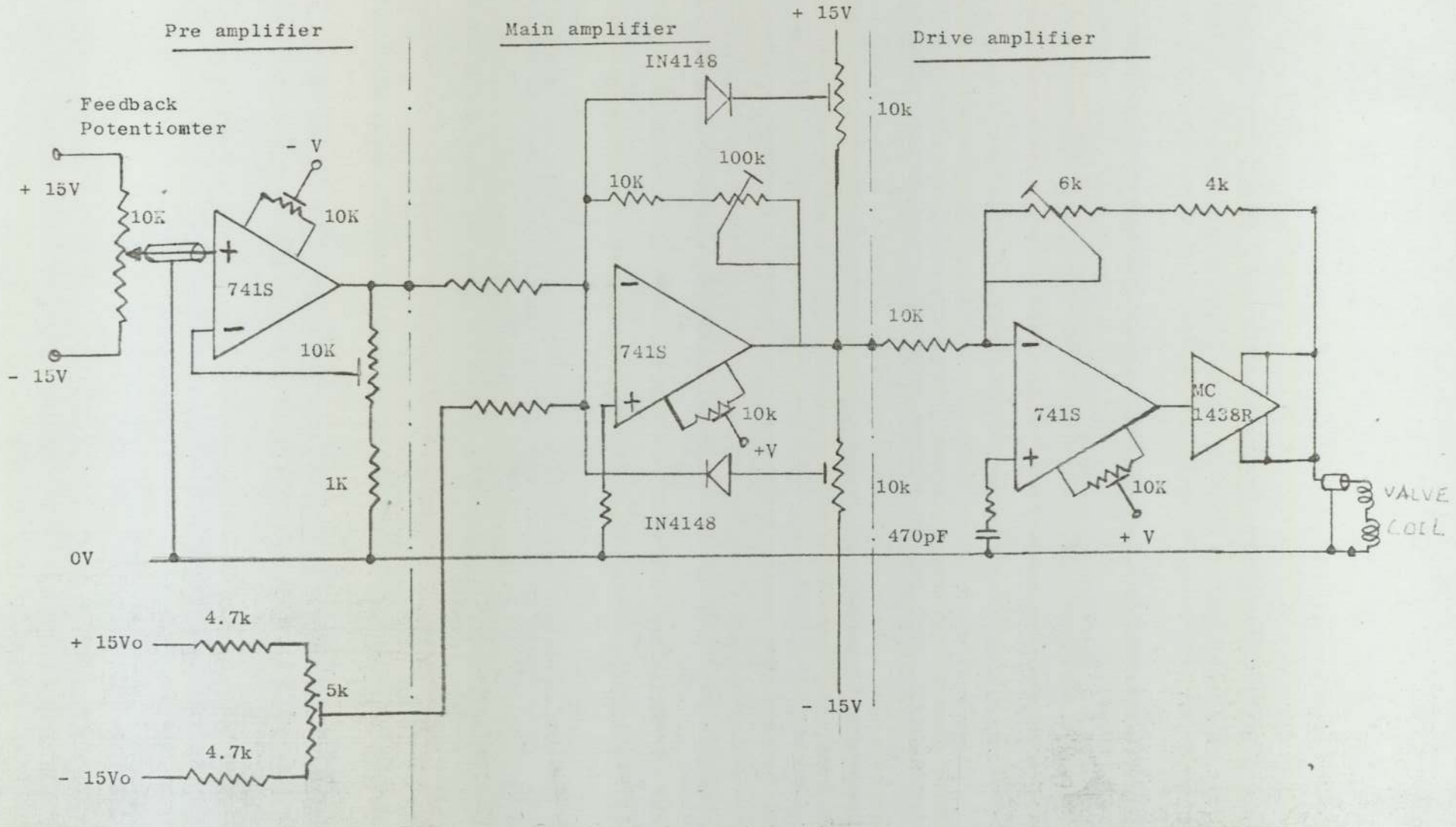


FIG 3.3 Servo Amplifier

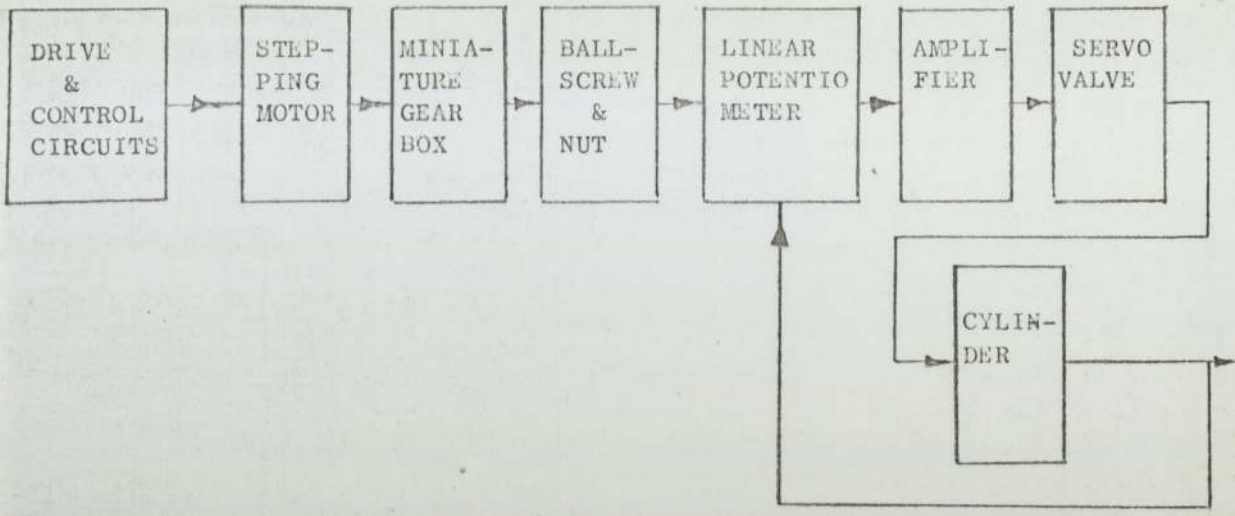


FIG 3.4a Y-Axis New Servo System

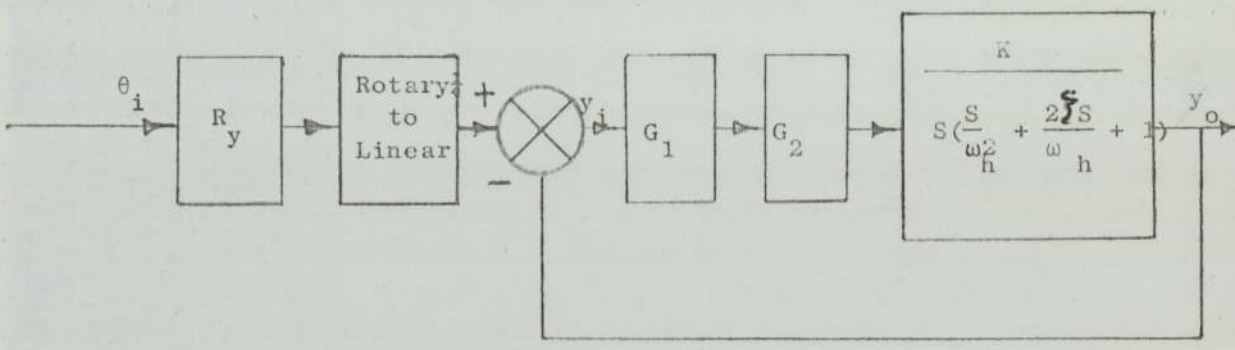


FIG 3.4b The Y-Axis servo Loop

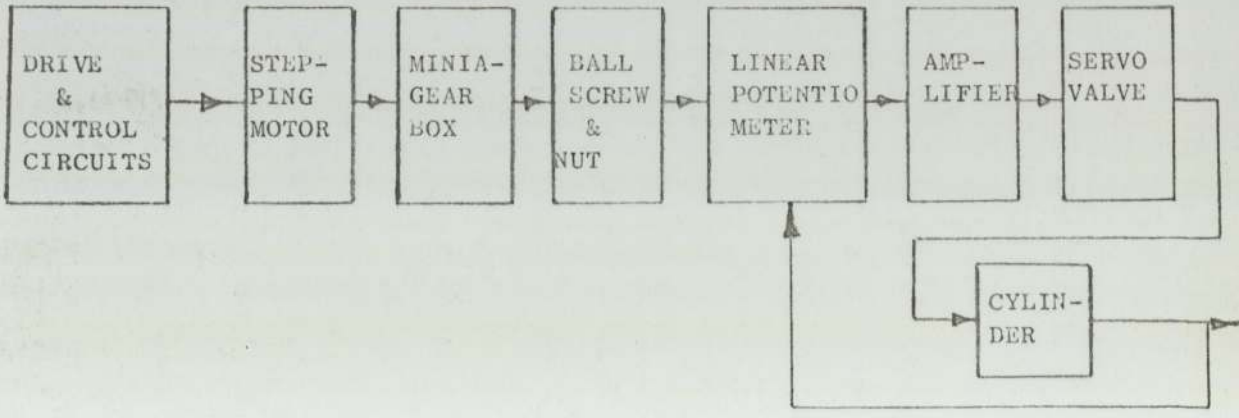


FIG 3.5 Z-Axis Servo Loop

3.5 The System requirements and the selection of the Stepping Motor

Usually the torque requirements in a CNC system constitute the crucial parameter in the mechanical design. The fundamental characteristic of a stepping motor is its torque-speed curve, figs. 2.10 and 3.6. The general design procedures, selection and control of electric and electroyhydraulic SM's are described in sections 2.4 - 2.6. For the purposes of the new control system for the HPE machine small electric SM's are required to perform the main control actuations for the electroyhydraulic servos on the axes. The torque requirements of each motor are only those to overcome friction in the servo potentiometers and miniature gear reducers and to accelerate the inertias at the inputs of the servo loops, figs. 3.2a and 3.4a.

In the X-axis the small rotary potentiometer requires a torque of 0.0002 N.M only. The Y-axis torque requirements are for the gear reducer, the ballscrew and nut, and the equivalent torques to overcome friction in the linear potentiometer and the Y-servo base plate. Although this torque is higher than that required for the X-axis, both have similar range of values and the SM's selected will be identical, each having torque available in excess of the axis requirements.

Another requirement (discussed in section 2.6) is an high value of the torque/inertia ratio because this roughly measures the ability of the motor to achieve fast accelerations, desirable in NC systems.

$$T = CI\alpha$$

$$\alpha = T/IC \qquad 3.13$$

where, T = Holding torque (gm. cm)

I = inertia (gm. cm^2)

α = acceleration (steps/s^2)

C = Convertion constant

Obviously equation 3.13 serves only as a general guide to SM selection since the torque available decreases with increase of stepping rate.

The small stepping motors selected are Moore Reeds 23MS108 (originally 22MS110) the specifications of which are shown below.

3.5.1 Specifications of 23SM108

Type	Hybrid
Holding torque	2600 gm.cm
step angle (half step)	0.9^o
Rotor Inertia	70 $gm.cm^2$
Number of phases	4
Required DC Voltage	24V
Current per phase	2.75A
Resistance per phase	0.6 Ω
* Maximum starting frequency	2200 steps/s
Power	200W
Step Error	5%

*In half step mode.

The torque/inertia ratio, R_{TI}, for these motors is:

$$R_{TI} = \frac{\text{Holding torque}}{\text{Rotor Inertia}} \qquad 3.14$$

i.e. $R_{TI} = \frac{2600X980}{70} \simeq 36 \times 10^3$, which indicates that the acceleration capability is very good.

In the HPE system the constraints on the maximum allowable acceleration are imposed by the electrohydraulic servo and the need to avoid shock loads on the machine system. The stepping motors have the capability of higher accelerations. However since this is a low speed application, having much higher torque available (fig. 3.6) than is demanded by the system a basic linear acceleration scheme suffices:

$$\alpha = \frac{\omega_F - \omega_o}{t} \qquad 3.15$$

where α = acceleration (steps/s^2)

ω_F = desired final speed (steps/s)

ω_o = starting speed (steps/s)

t = duration of acceleration (s)

In higher speed direct electric SM applications, where the torque available varies critically, it would be recommended to employ the schemes developed by Maginot and Oliver (6), or more practically, Rahmen (8), all of which are discussed in section 2.4.4.

In this application the maximum speed in each axis is 15 in/min and for the designed machine resolution of .0001 in/step, the maximum required pulse rate is:

$$\frac{15}{.0001} \times \frac{1}{60} \text{ (steps/s)} = \frac{150000}{60} = 2.5\text{KHZ}$$

3.6 Tests on the Servo drives of the X and Y axes

In order to test the systems a manual control 'black box' was created to generate discrete positional and velocity commands to the stepping motors on the axes. Further details of this electronic circuitry are contained in Achi (84). It includes counters and display units, fig. 3.7, together with the motor drive logic modules (SM translators). The drive logic gives the options of full step (1.8) or half step (0.9) operation. Eventually the half stepping mode was to be adopted after confirmation of the advantages discussed in section 2.4 and 2.6. Bench tests on the SM alone had shown the presence of oscillations in the unit at low speeds but no corrective action was taken at this stage as the effect of the rest of the drive system might well improve matters. It was sufficient to adjust the servo gain to an interim value for an optimum response to a unit step input. A Viscous inertia damper was later found to be effective in reducing the oscillations.

3.6.1 Measurements on the Servo Systems

(i) Resonance: The signal traces in fig. 3.8 confirm the effect of the damper in reducing the oscillation amplitudes of the driving velocity following error. The advantages of a viscous inertia damper investigated by Kordik (68) are also true for this design which uses the SM as actuator for an hydraulic servo. Similarly the oscillation reducing effects of operating a direct drive electric SM in the half step mode as described by Acarnley (72) was confirmed in the tests in this system. The primary oscillation occurs at the stepping frequency and the secondary effect arises from the servo system itself (fig. 3.8).

(ii) Machine Resolution: The design machine resolution of 0.0001 in/step was confirmed during the manual, and later, the microcomputer control tests. The anti-backlash gearing and the preloaded ballscrew units were completely effective in maintaining this resolution for motions in both directions.

(iii) Feedrate: This was found to bear a linear relationship with the pulse rate supplied to the motor. Measurements of the traverse speed, the servo valve current, the velocity following error (voltage) at the potentiometer and the equivalent step following error were made at various stepping rates for the X and Y axes. These measurements were repeated later during μC control. Figs. 3.9a → 3.9g show the various graphs of these test results on the servo system. The linear relationships confirm that the system is suitable for computer control (figs. 9.6a and 9.6b).

The brief deviation from linearity in the region of the origin (fig. 3.9a→g) is a result of internal friction in the servo drives. It was found to be more pronounced in the X than in the Y axis as the graphs confirm.

Although the SM's themselves are capable of a maximum starting frequency of 2KHZ, it was found that the maximum starting frequencies to which the X and Y axes could respond accurately were about 1000and just over 800 pulses/s respectively. For this reason it was decided (for μC control) to fix the maximum starting frequency in each axis at 500pps.

3.7 The effect of dither oscillator and stepping mode on the static friction in stepping motor servo drives

Usually in electrohydraulic drives it is necessary to include some dither oscillation with the signal to the servo valve. This helps to overcome the effect of static friction in the drive and so improves the systems resolution and linearity. In this application tests showed that the velocity following error oscillations were increased when the recommended level of dither signal was added (signal traces in figs. 3.8b → 3.8c). Reduction in the dither signal amplitude rendered it ineffective and indeed equivalent dither already existed by way of the SM oscillations themselves, even from the first step. Therefore the external dither signal was removed.

Beling (66) points out that the fractional stepping mode (1/2, 1/4, or 1/8 step modes) affects the degreee to which a direct drive electric SM is able to overcome friction in the drive. He describes the fractional step modes as having variable 'stiffness' per step. Although he did not specify what 'stiffness' might mean exactly in this context, the finding from figs. 3.9a(i), 3.9c(i) and 3.9g(i) in this study suggests that the mode of operation affects the linearity of the servo system at start up. The minimal non-linearities around the origin are more pronounced in the half than the full step drive mode.

As was mentioned above, using an external dither signal (of the proper amplitude) to correct the non-linearities in the half

step mode encourages resonance of the system following error. (The usual dither signal and SM natural frequencies of oscillation are in the region of 100HZ). The inherent dither-like oscillation present in the SM stepping operation has higher amplitudes in the full than the half step mode and so more easily enables the SM to overcome the frictions in the drive (referred to by Beling (ibid)) and thus improves the system linearity.

However Beling observed correctly that fractional step drives in reducing resonance, help to avoid machine tool chatter and lack of straightness in X, Y plots. So he recommends the use of fractional step drives in all speed ranges. The smooth X, Y traces under the microcomputer control of the HPE M/C (figs. 9.6a and 9.6b) are evidence of the effectiveness of operating in the half step mode.

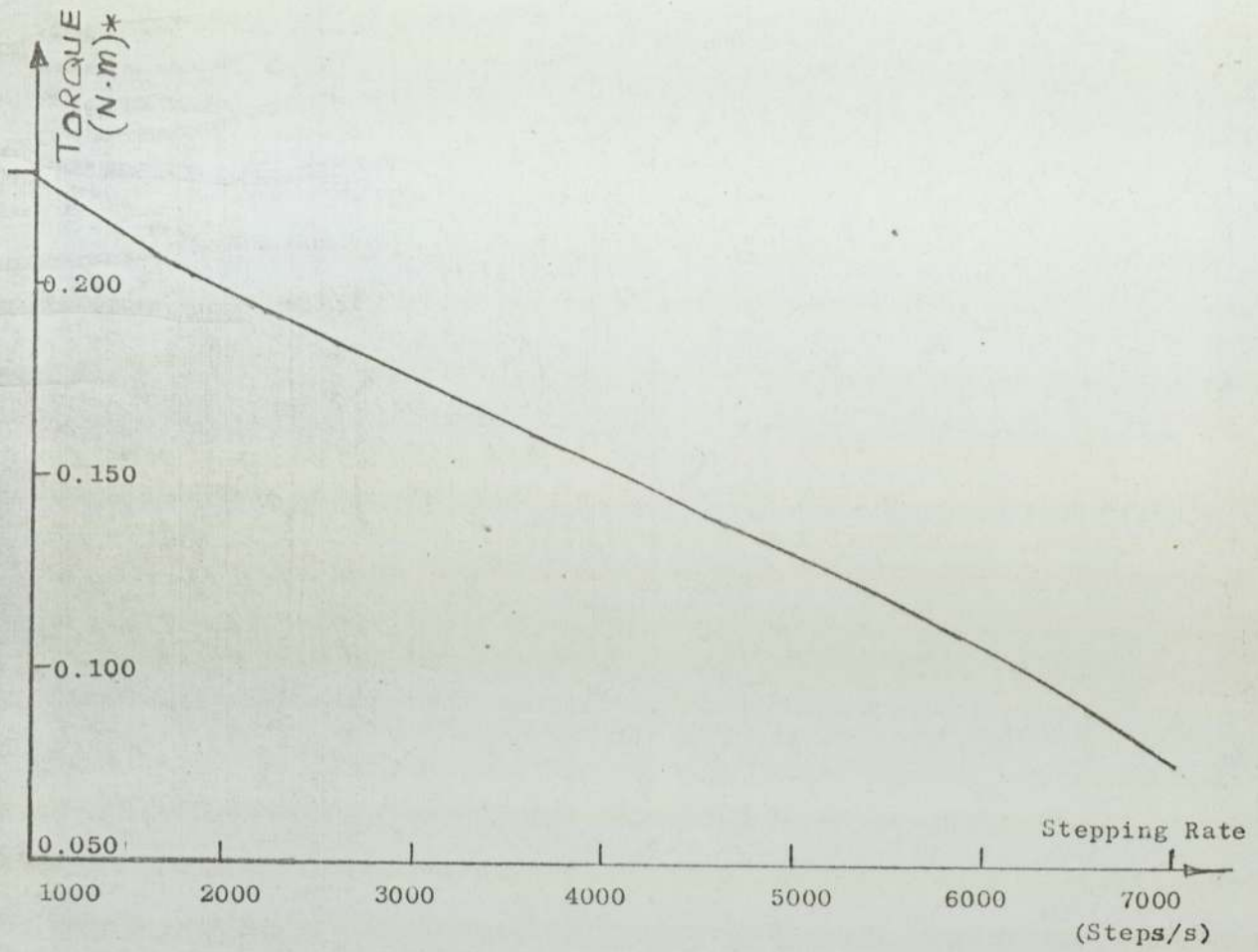


*FIG 3.6 The Stepping Motor (22MSIIO) Torque/Speed Curve

* Half-step mode, 0.9° step angle

Below the stepping rate of 1000 (steps/s) torque available is is greater than o.2N.m for this small stepping motor.

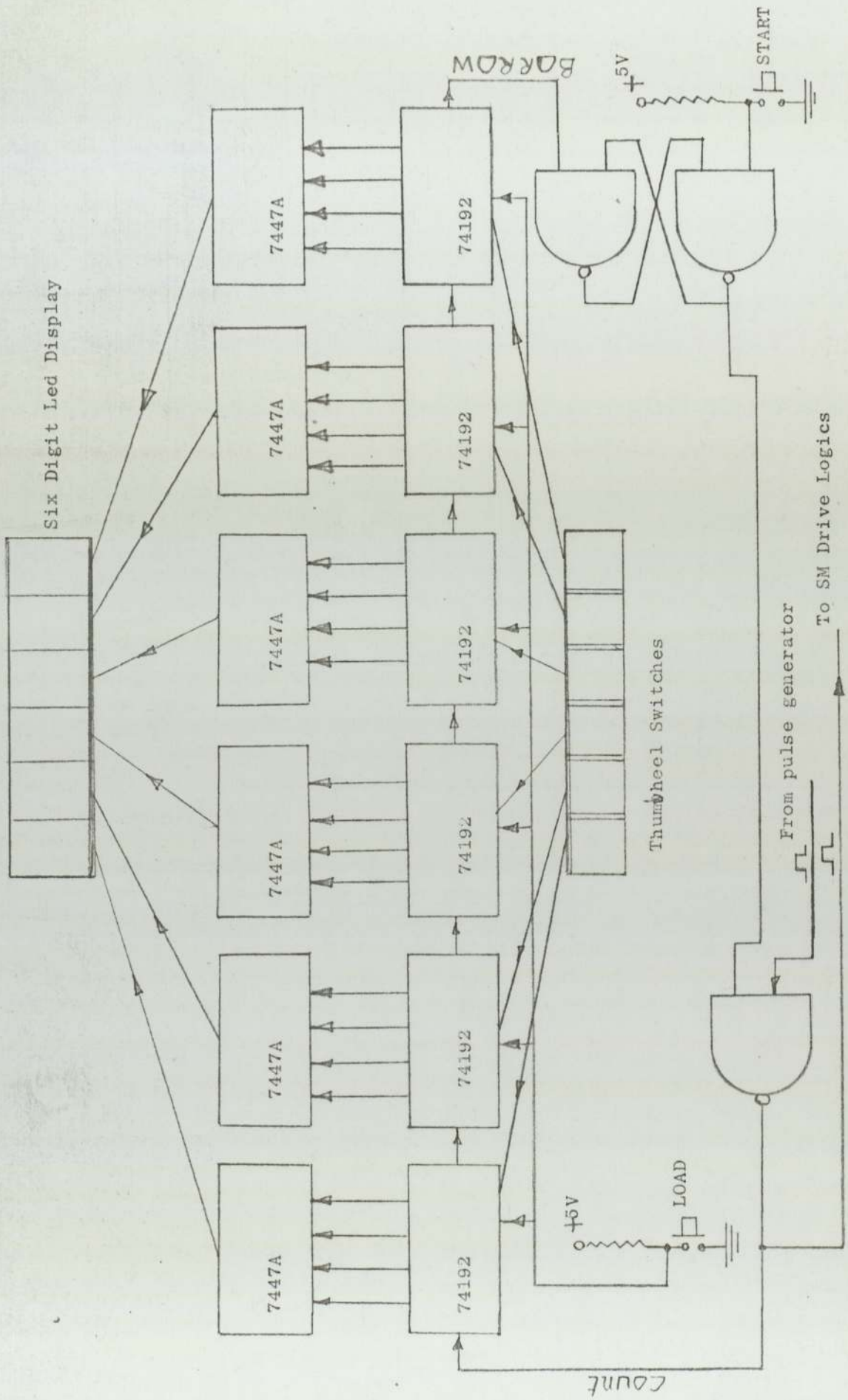
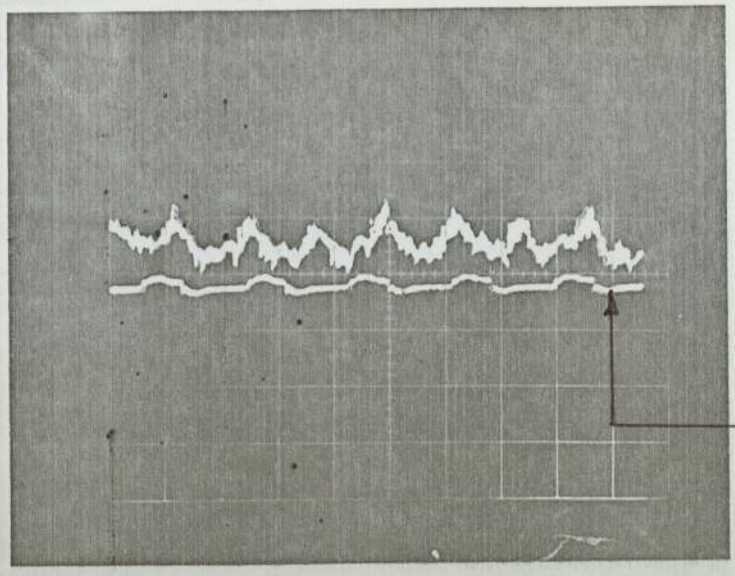
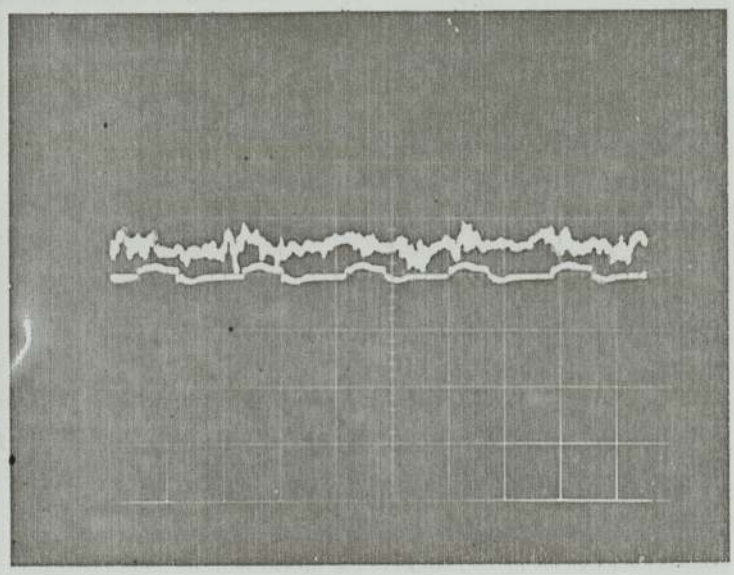


FIG 3.7 Manual Control Circuitry for the Stepping Motors

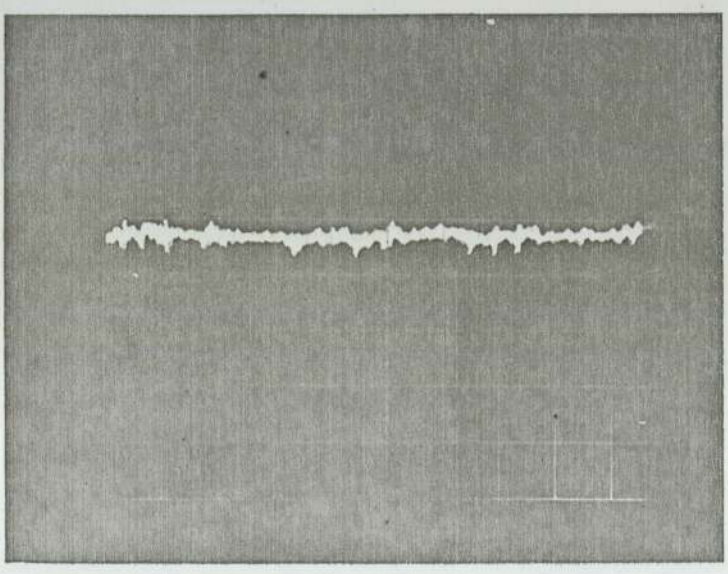


(Vertical scale: 500mV/C
horizontal scale: 2m5/CM
speed: 500 pps.

(a) With dither and no damper



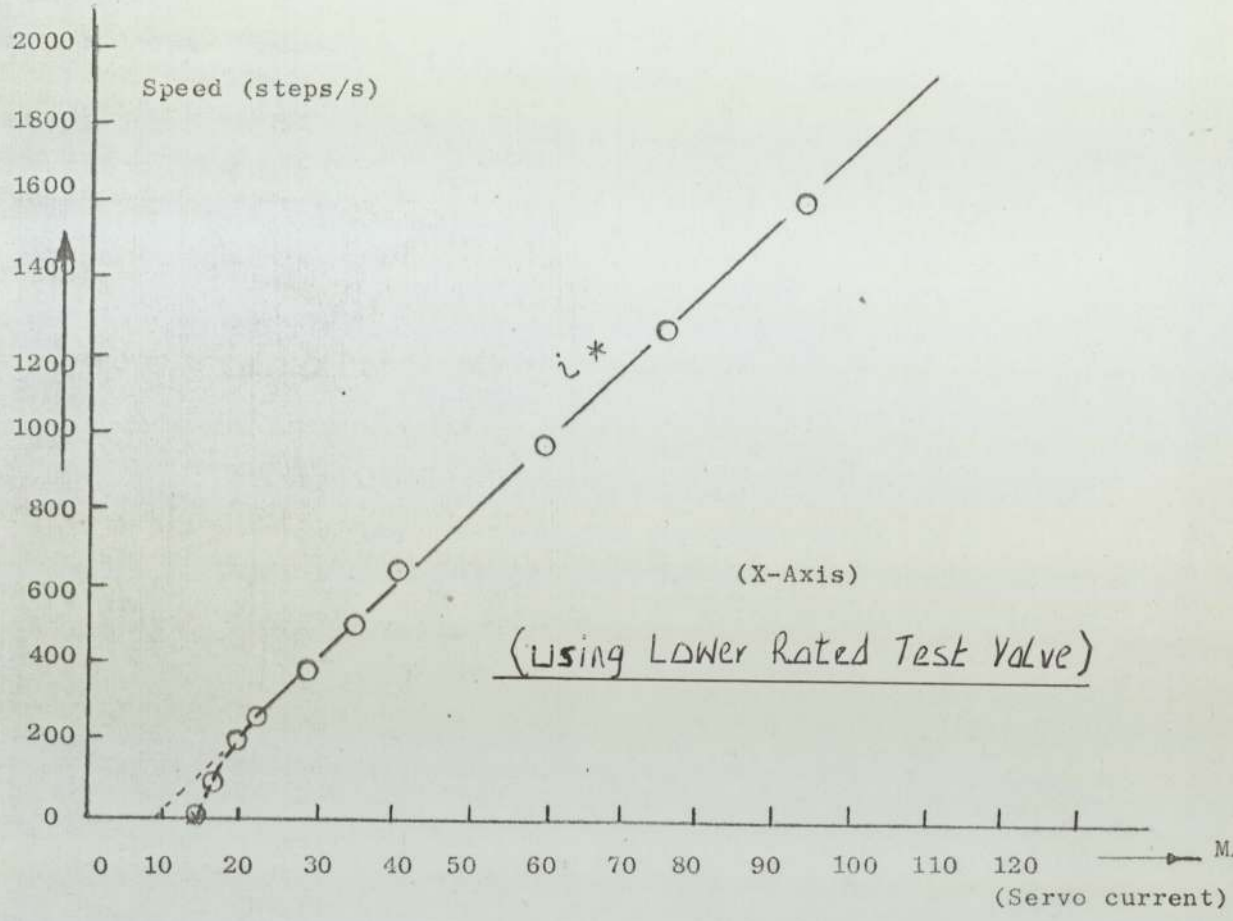
(b) With dither and damper



(c) With damper and no dither

Fig 3.8 The Servo Voltage

Effect of dither Oscillator on the steppin motor hydraulic servo system



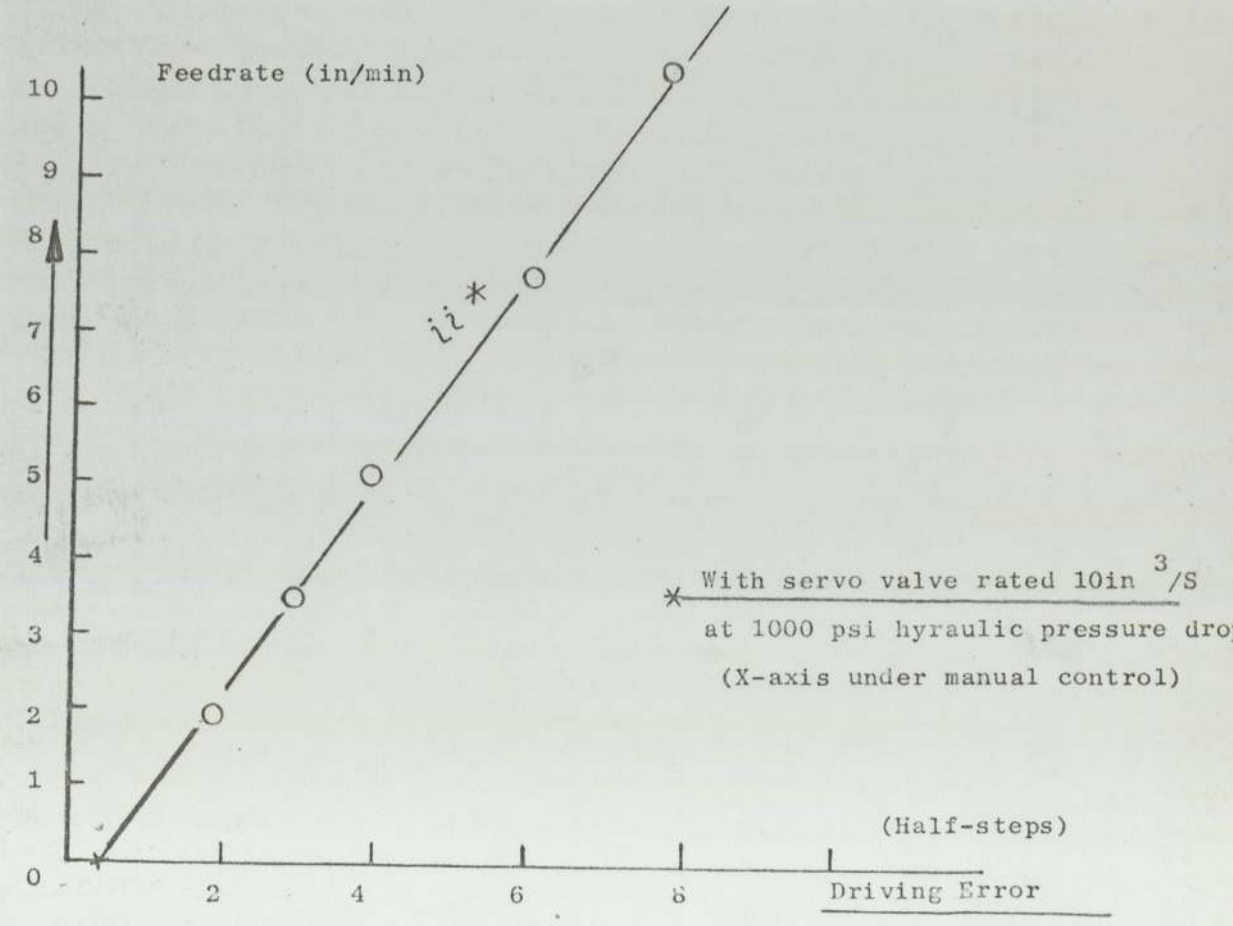
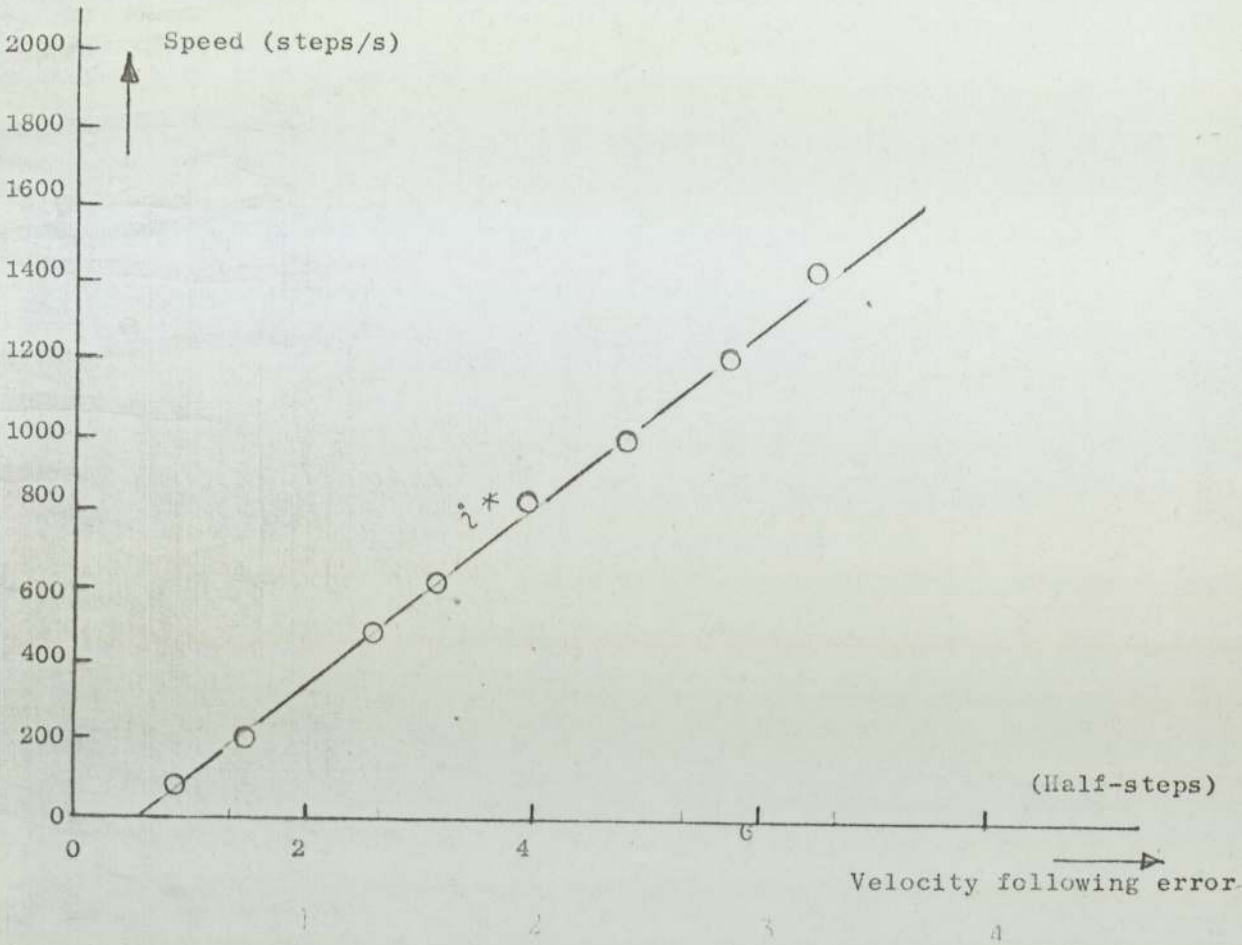


FIG 3.9a Graphs of the Stepping Motor Servo System on the milling M/C



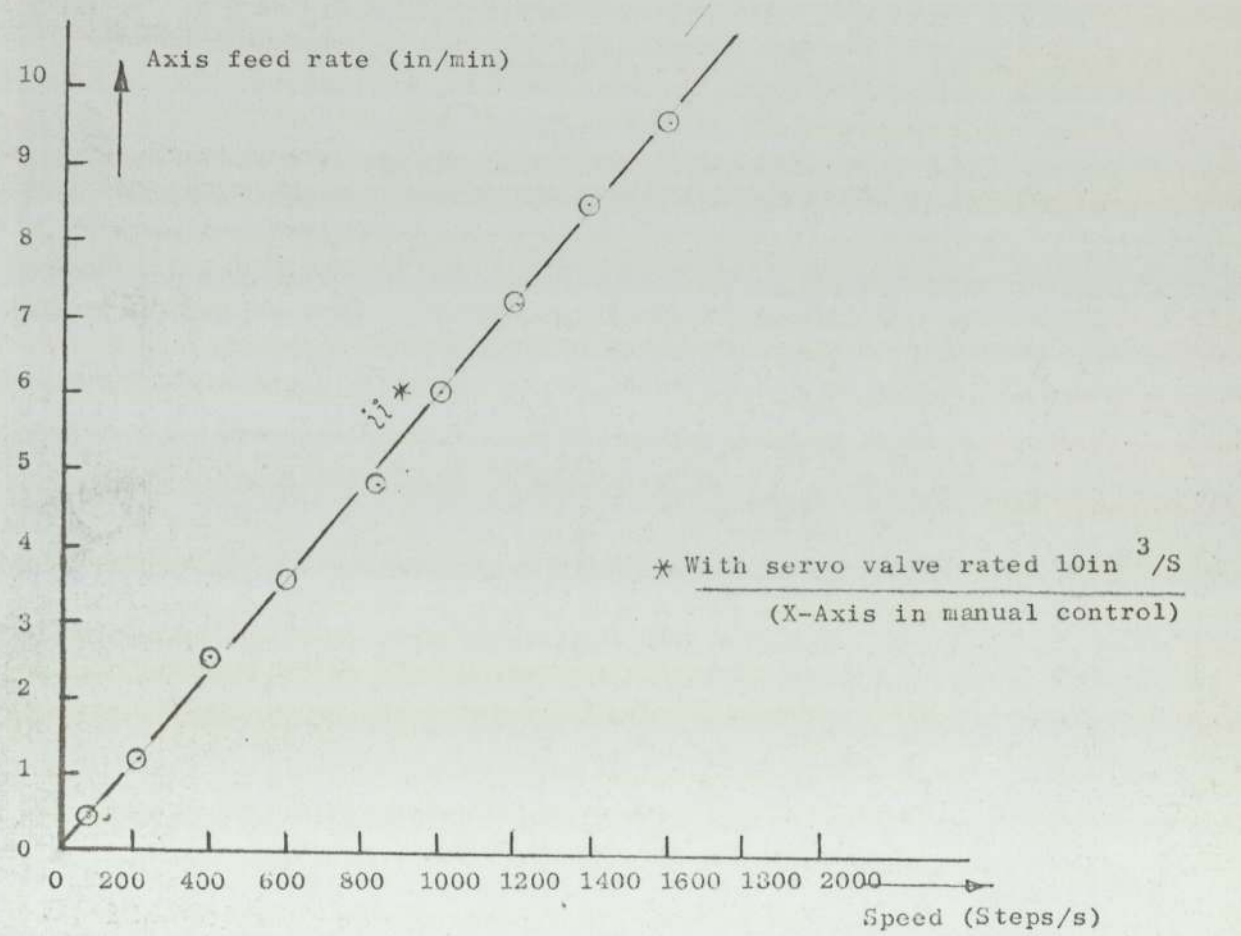
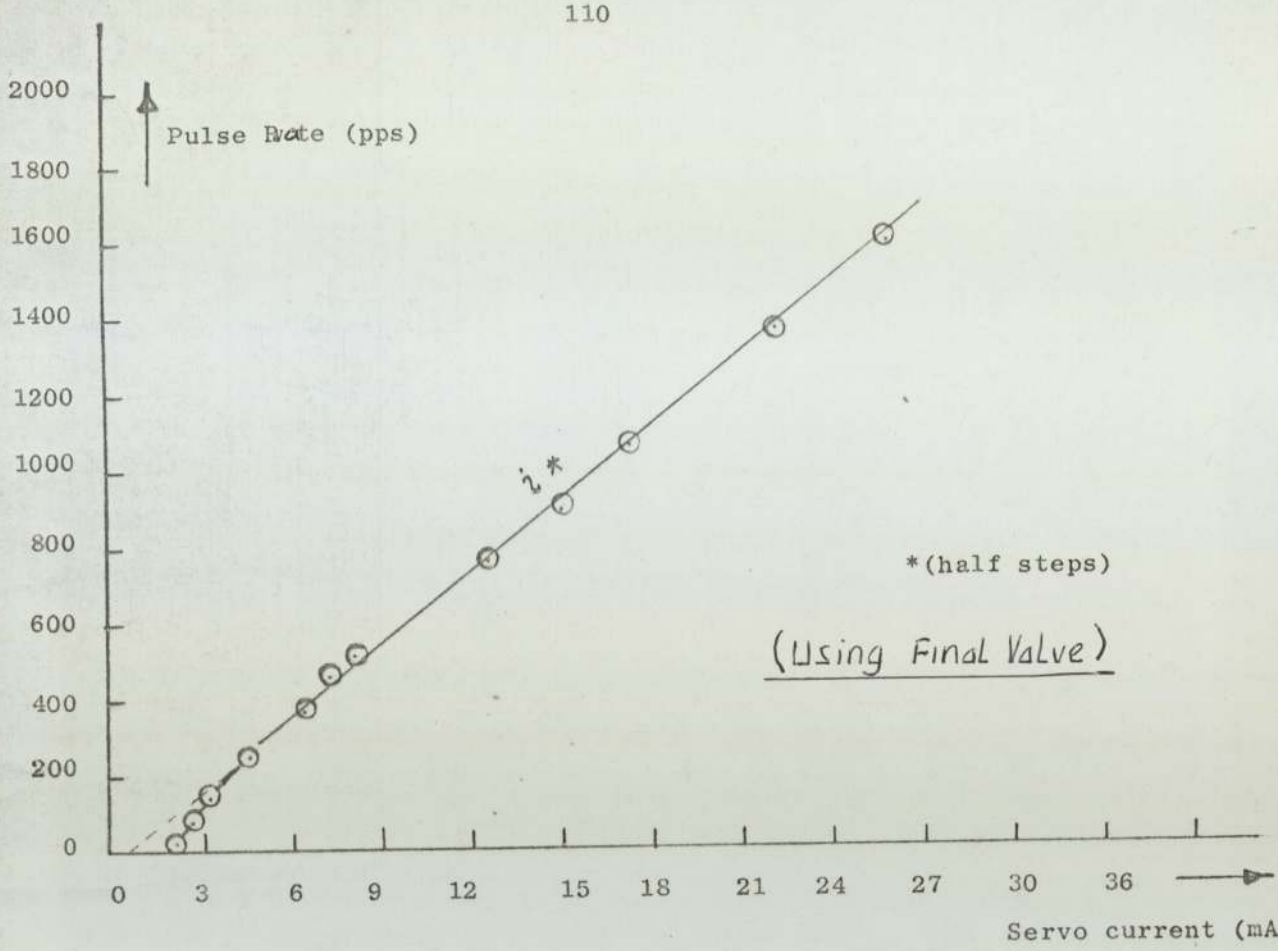


FIG 3.9b Graphs of the Stepping Motor Servo System on the machine



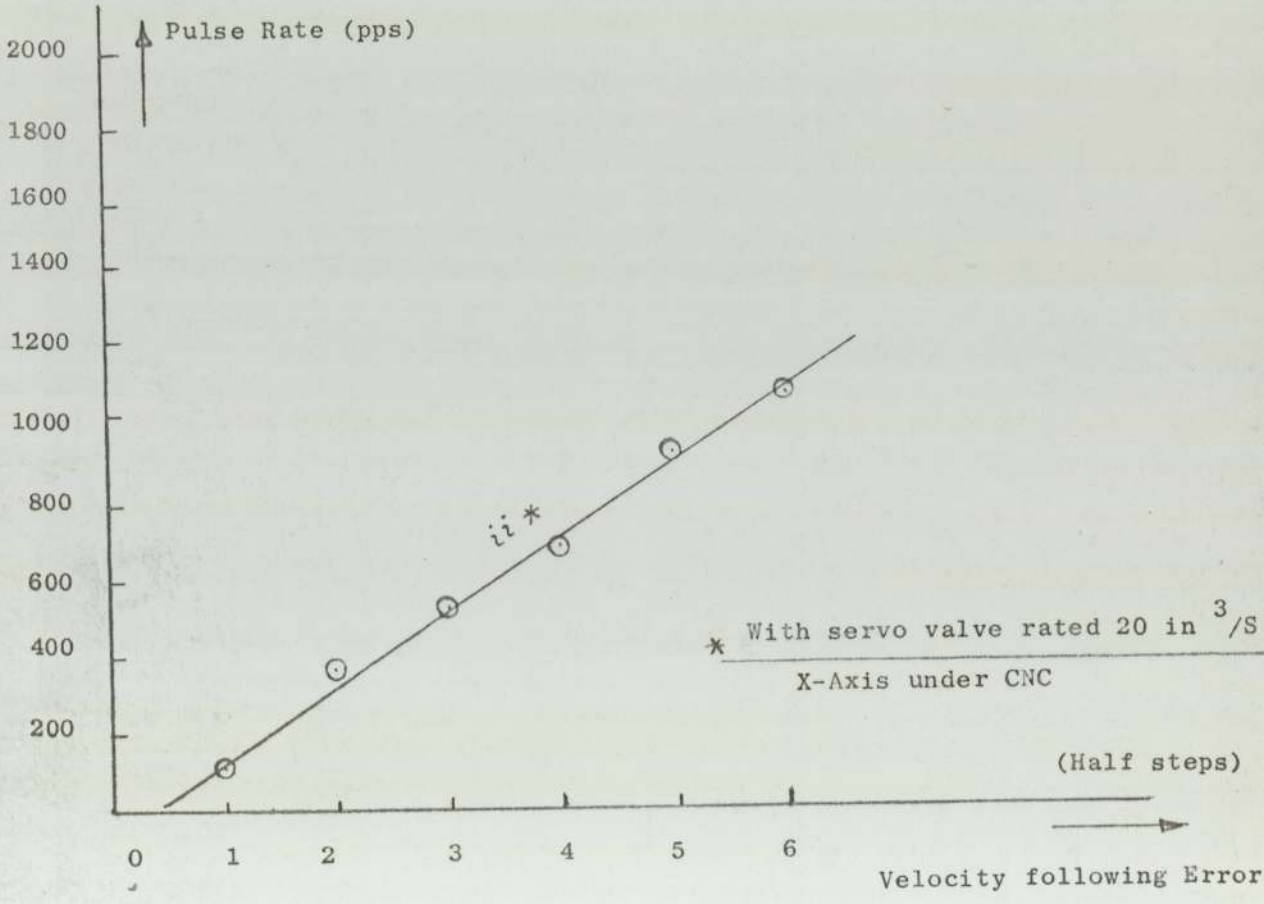
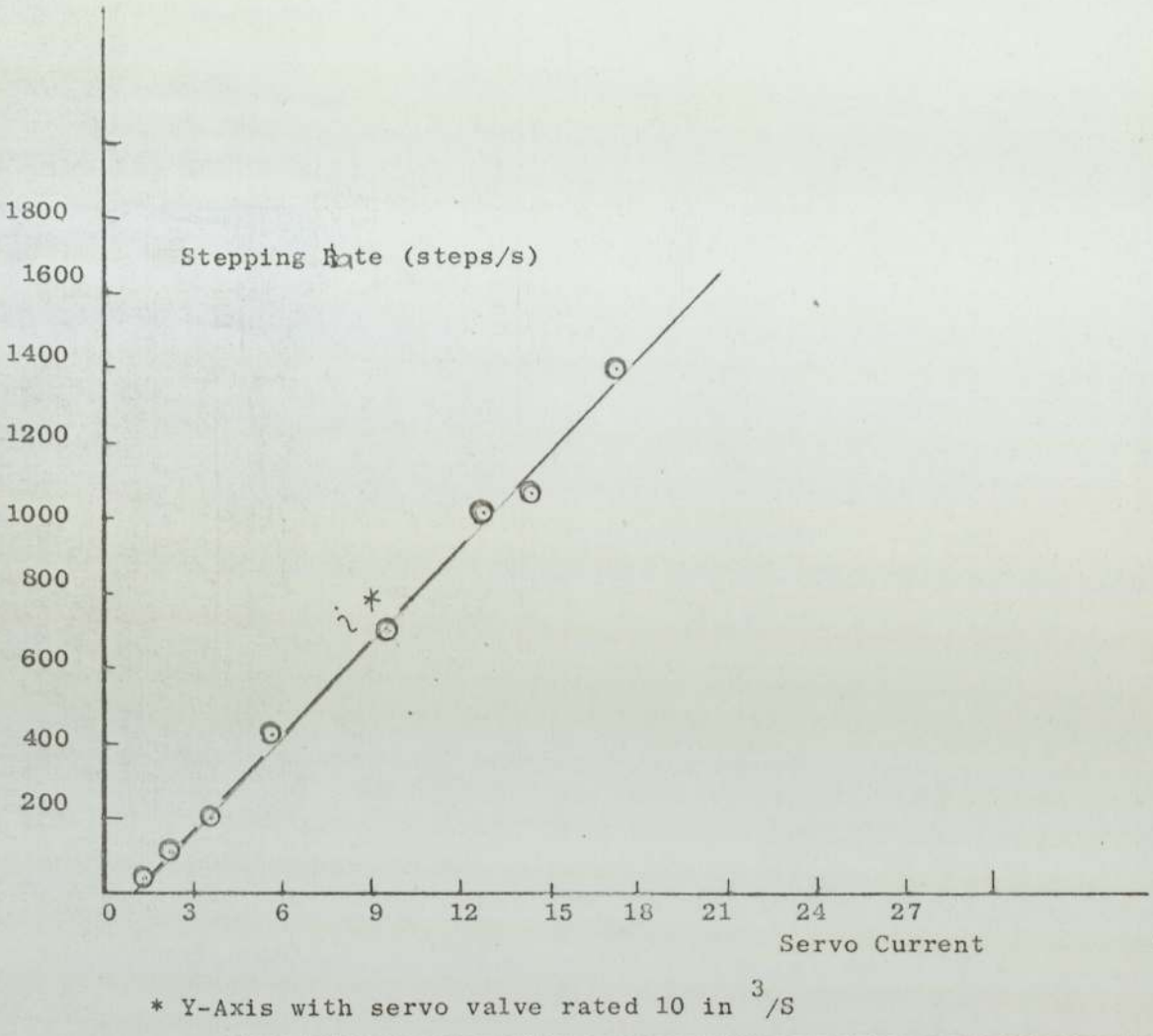


FIG 3.9c X-Axis stepping motor hydraulic servo system on the machine Under Computer Control



* Y-Axis with servo valve rated 10 in ^{3}/S

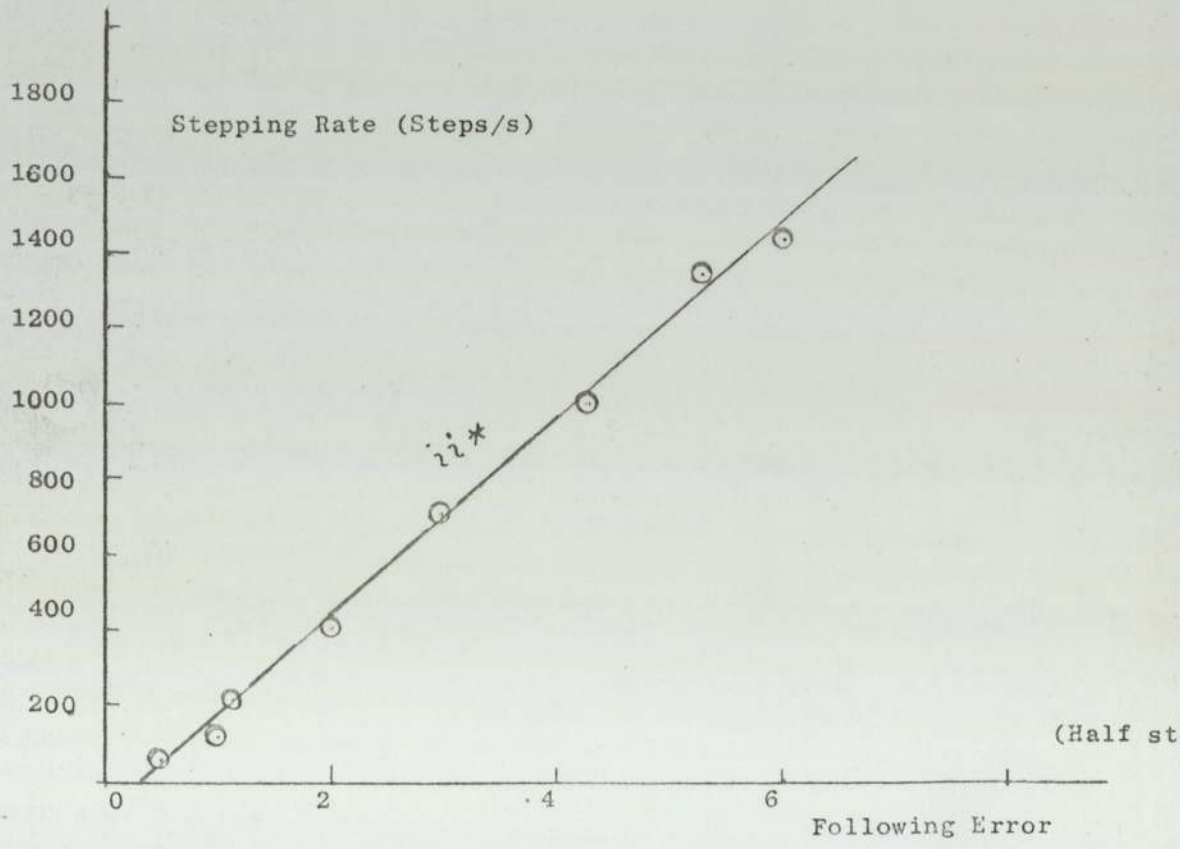


FIG 3.9e Stepping motor hydraulic Servo System (Y-Axis)

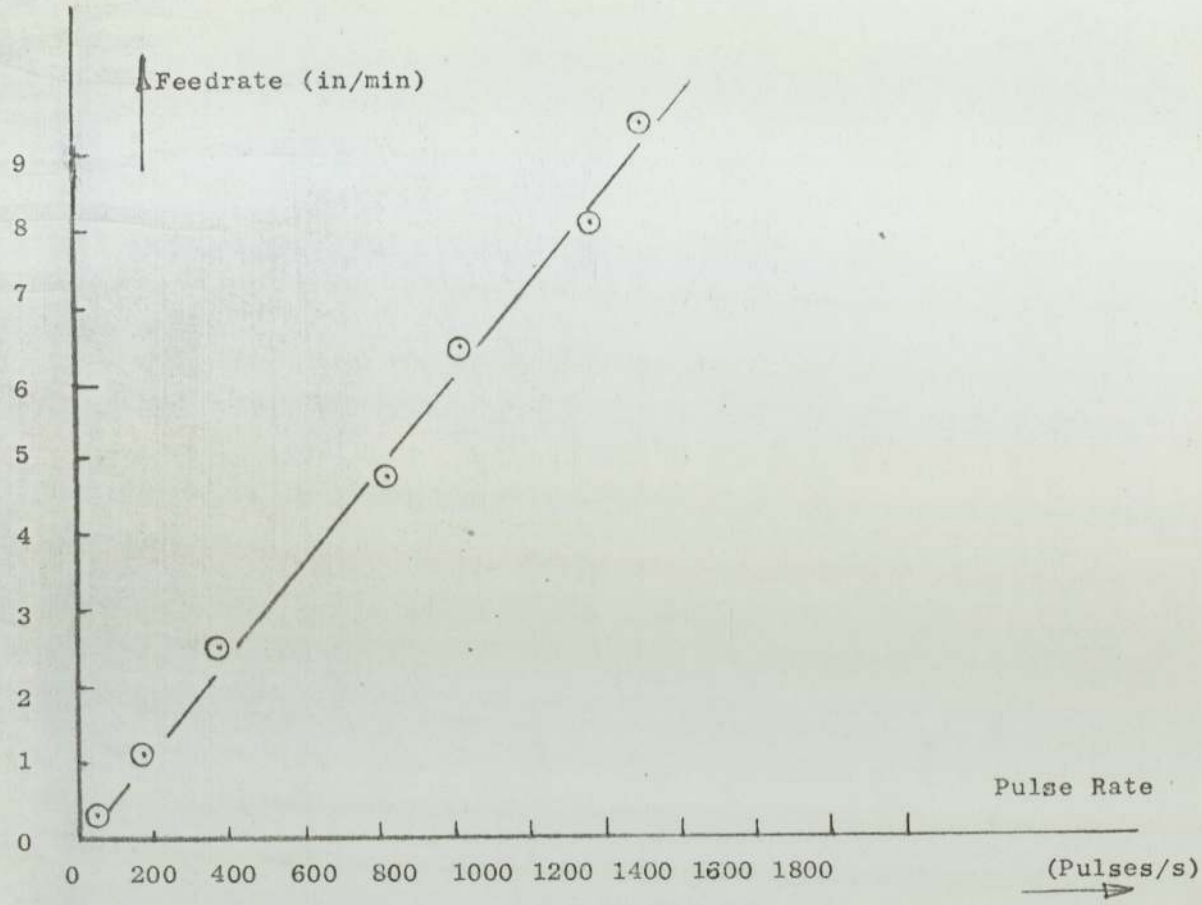


FIG 3.9f(i) Stepping Rate Relationship with Feedrate in Y-Axis Under µComputer Control

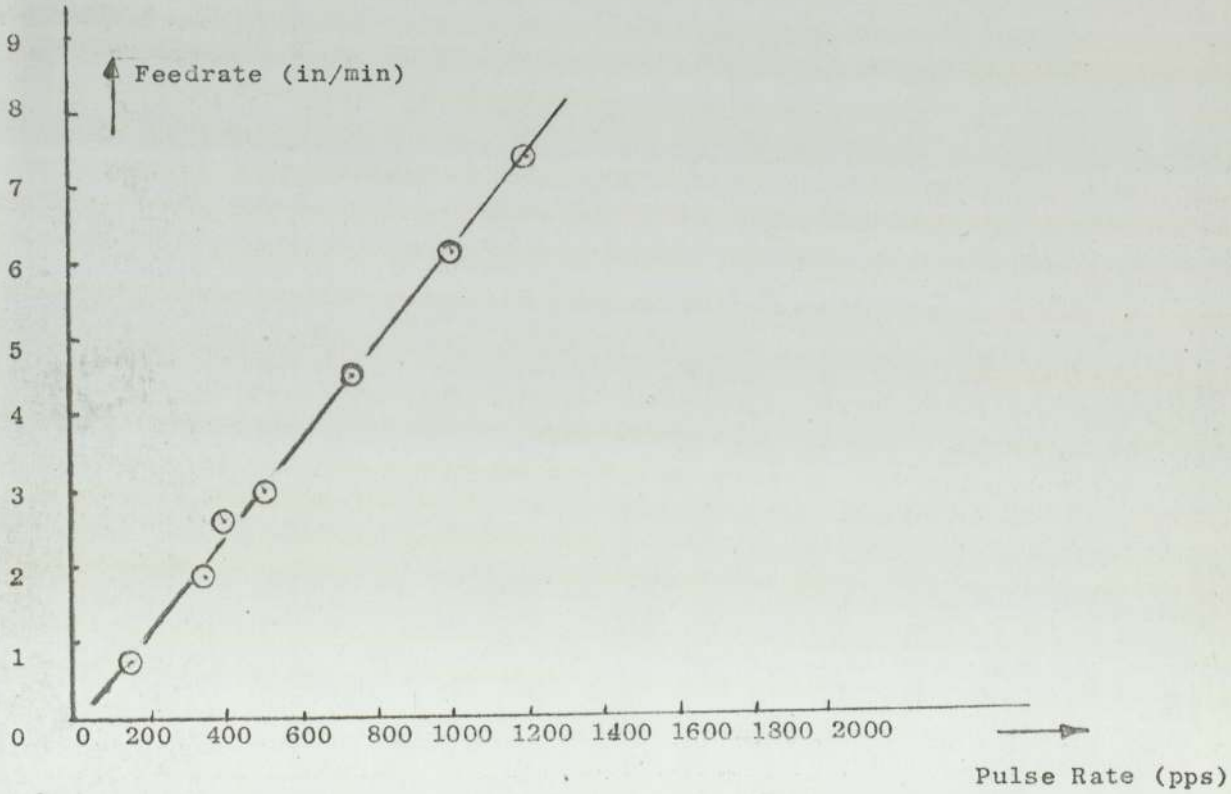
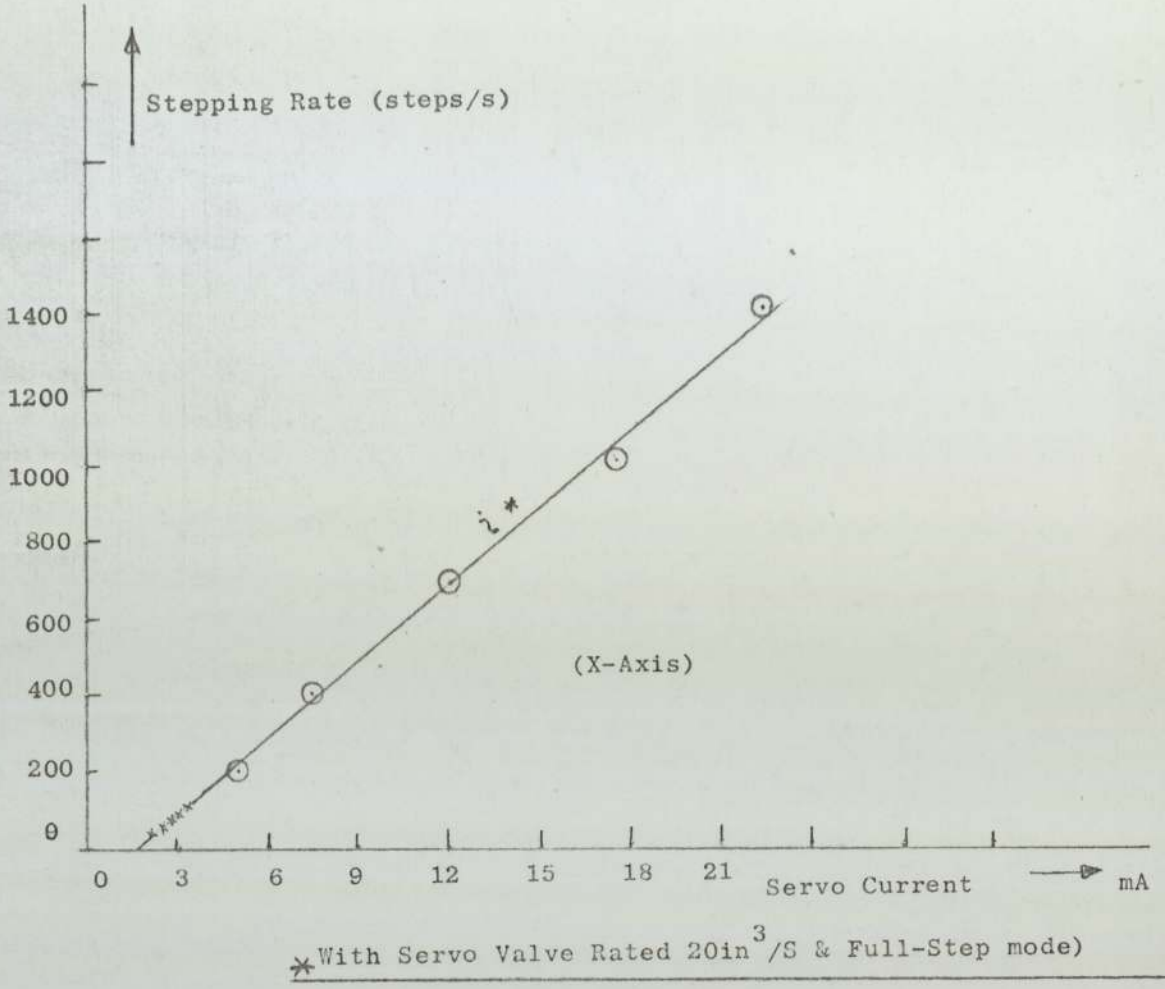


FIG 3.9f(ii) Stepping Rate Relationship with Feedrate In X-Axis under µComputer Control



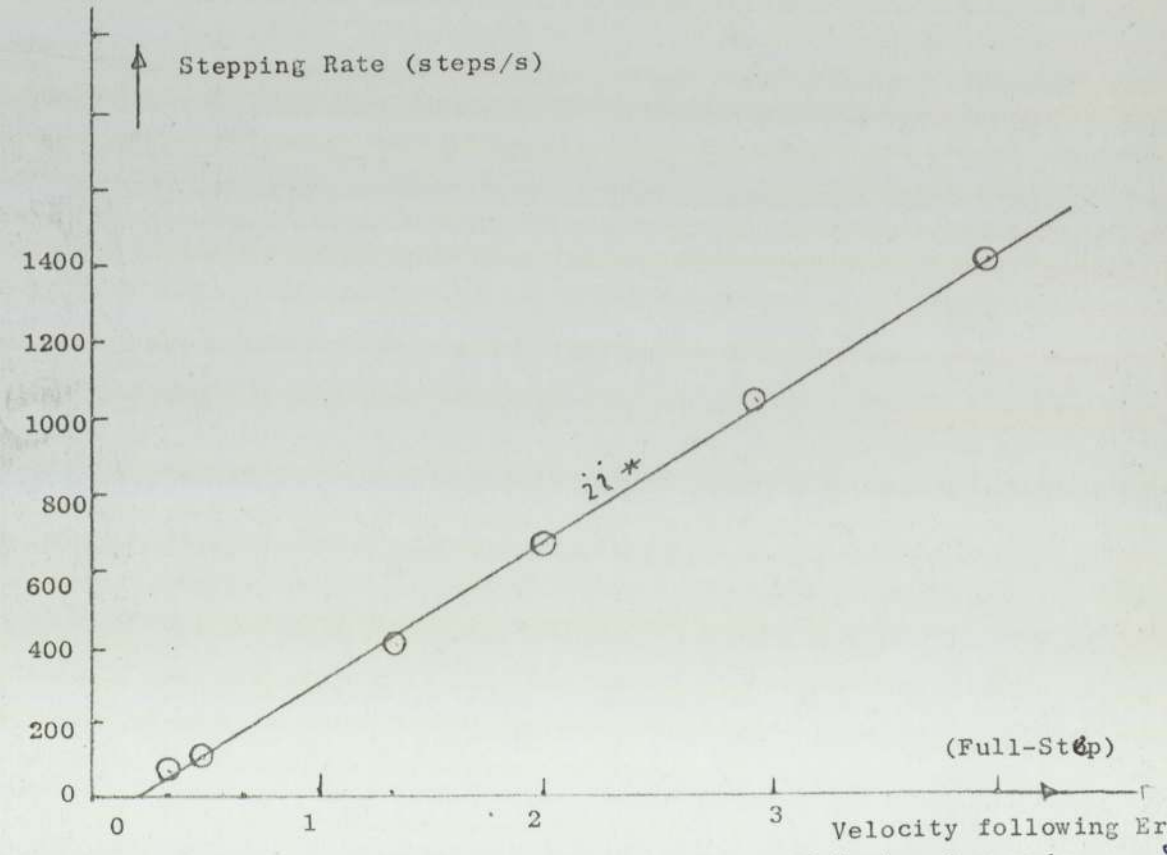


FIG 3.9g Servo System using the 'full step' stepping mode

CHAPTER 4

THE MICROCOMPUTER SYSTEM - HARDWARE FEATURES

4.0 Introduction

The available microcomputer system used in this study is the first generation microNOVA having Data General Corporation's MN601 16-bit microprocessor as its CPU (fig. 4.0). This μP is a silicon-gate N-channel (NMOS) LSI supported by three other external chip sets:

the MN603 I/O controller (IOC) modules

the MN606 4K dynamic RAM modules

the System Buffer Elements (SBE's) or 'Transceivers'

The design of the microNOVA architecture is based on separate memory and I/O buses (fig. 4.1). This prevents devices, other than the μP (CPU), from having direct access to the memory modules and so precludes actual direct memory access (DMA) for I/O data operations. Data from memory must pass via the μP (CPU) before it is transmitted by the latter to the device(s) requesting it. This type of operation makes it necessary for the μP to synchronize every data channel request by a peripheral or an I/O device before the data is transfered. Such a synchronization process is referred to as 'data channel break'. Thus the asynchronous DMA bus which is desirable and often recommended is non-existent and this may justify some of the criticisms of the microNOVA mentioned in section 2.1.2.1. It is admitted that with this system any malfunctions that occur may often take a long time to diagnose precisely. The multi-bus system (fig. 4.1) through which data has to pass may render the latter susceptible to corruption at some stage and processing speeds are affected by both the μP and μC architectures. The microNOVA was used to implement

the design in this study mainly because it happened to be available at the beginning of the study and because of the existing economic situation it could not be replaced by a later generation microcomputer. In fact the microNOVA and its μP have since been redesigned and named MP/200 and MN/602 respectively. The new versions are said to offer more than three times better performance (11). Table 4.0 below gives a brief performance comparison between them. The architectural modifications in the new microNOVA are not included in the reference.

*Table 4.0 Performance Comparison with the redesigned microNOVA

Performance Aspects		MN601	MN602
Data channel throughput	(megabyte/s)	0.3	3.7
16-bit addition	(μS)	2.4	0.8
16-bit multiplication	(μS)	41.3	4.9

*Values for MN602 (the redesigned μP) are from ref. (11)

Despite its architectural problems the microNOVA has the benefits of some established Data General S/W. It is by no means the most suitable for applications to (NC machine) control. The criteria for selecting a μC have been summarised earlier in section 2.1.2, but in this instance the microNOVA was already available and even with its limitations provided a means of researching the necessary techniques. The μC control aspects for NC LSI's as described in chapter 5, (and later in chapters 6 & 7), are general for other 16 bit μC's. Most of these have standard bus structures which can be floated for direct memory access, a feature often desirable for NC applications.

4.1 The Microprocessor MN601

The microprocessor (fig. 4.0) is the central processing unit in a 40-pin package. It contains the I/O control logic and the 'arithmetic and logical unit' (ALU) which handles data processing. Instead of on-chip memory, multiple registers (Table 4.1) are separately incorporated within the chip and these are employed to transfer data between the external memory and the μP via the external bus interfaces or 'transceivers'. The μP is responsible for the execution of instructions, program interrupts and data channel 'breaks'. It carries out these functions by transmitting and receiving data and control signals via 30 of its pins. The registers are used for temporary storage of data transferred via the external pins to or from the 'world' outside the μP LSI.

The necessary power supplies are (fig. 4.0):
VBB = -5V (pin 1), VCC = + 5V (pin 17), VGG = + 15V (pin 7) and VDD = - 15V (pin 38). The average power dissipation within the microprocessor is 1W. The lines MB0 - MB15 (pins 9 - 25, except 17) represent the 16 bits which are connected to the memory bus via two octal memory bus transceivers, each of which is 8 - bits wide. MB0 is the most significant bit and MB15 is the least significant bit and except for bit 0, the other 15 bits are grouped in threes to form the octal equivalent of the 16-bit address, data and instruction fields. The MB(0 - 15) pins can transmit logic 1's (+5 V) or 0's (0V) and each can sink or source up to ±2MA.

4.1.1 The general organization of the microprocessor

Table 4.1 The 13 Registers used by the µP

Register	Size
Accumulator 0	16 bits
Accumulator 1	16 bits
Accumulator 2	16 bits
Accumulator 3	16 bits
Stack pointer	15 bits
Frame pointer	15 bits
Program counter	15 bits
Carry bit	1 bit
Interrupt enable bit	1 bit
'Real-time clock' enable bit	1 bit
Stack-overflow request bit (flag)	1 bit
Real-time clock request bit	1 bit
Refresh address counter	6 bits

The registers handle control, addresses and data for transfers via the µP pins. Thirty of the forty pins are used by the µP for information transfers with external circuity in the µC system. Of the remaining ten pins seven are power supply inputs and the remaining three are not connected. Table 4.2 shows how the thirty pins used for external communications are functionally divided.

4.1.2 The position of the registers and their effect on performance

The four 16-bit accumulators in the microNOVA μP are used as general purpose registers (ie for instructions, memory addressing, I/O and for arithmetic operands). Two of the three 15-bit registers (frame and stack pointers) are employed for memory stack indexing and manipulations

Table 4.2 The functional organisation (fig. 4.0)

Functional description	Pin designation
Two μP driving clocks	alpha 1, 3 ($\alpha_{1,3}$)
	alpha 2, 4($\alpha_{2,4}$)
For control of memory	PG
transfers to/from μP	SAEG
	WEG
Memory address/data pins	MB0_MB15
I/O data control pins	I/O CLOCK
	I/O DATA 1
	I/O DATA 2
	I/O INPUT
I/O interrupt request pins	EXT INT
	DCH INT
Mode Control pins	CLAMP
	HALT
	PAUSE

whenever required in a program execution, while the third is the sequential program counter and can address up to 2^{15} (ie 32,768) memory locations. The 6-bit refresh address counter-register is used by the μP to index any of the sixty four 512-word dynamic RAM locations. The refresh operations are consequent on the use of dynamic RAMS (in micro NOVA), the locations of which must be refreshed to maintain their contents.

The integration of the 13 registers within the μP considerably reduces the space available for further integration necessary for high processing speeds. Because of this the microNOVA has to compromise mainly at the expense of processing speed. (The instruction repertoire resembles that of the original NOVA 3 minicomputer). In contrast, according to ERA (19) the 'Fairchild 8" μP has its memory address and program counter registers implemented in a ROM chip, and the TMS 9900 μP with sixteen 16-bit registers has all these implemented in the main memory. This is done to save area within the μP for the extra integration and makes the implementation of a powerful instruction set possible. Thirteen of the 16 registers in TMS9900 are used as general purpose registers (accumulators). The microNOVA, however, does have the advantage of Data General's established S/W support and would be a more competitive system given a standard bus architecture.

4.1.3 Memory control and address/data transfer

The control of communication between the μP and the external memory is via the PG, SAEG, WEG and MB0 - MB15 pins. Table 4.3 shows the applications of these pins. The clock cycles $\alpha_{1,3}$ and $\alpha_{2,4}$ are used to pre-charge the address/data lines MB0 - MB15 whether the μP is

performing a memory operation or not. This pre-charging is a result of the μP's requirement that none of the address/data pins should be at a low logic level during an α_2 clock cycle.

Table 4.3 The operation of the memory pins

Pin Name	Function	Direction
PG	Initiates memory operation	From μP
SAEG	Indicates data transfer to μP	From μP
WEG	Indicates data transfer from μP	From μP
MBO - MB15	Carry address and data	To/From μP

The WEG pin goes low and the SAEG pin goes high for the μP to 'Read' data from memory and the logic levels on these two pins are interchanged for the μP to 'write' data to a memory location. All these operations are timed by the 120 ns period multiples of the two alpha clocks giving an equal cycle time of 960 ns for both Read and Write memory operations which are performed in a 16-bit parallel fashion.

4.1.4 The I/O transfer control procedures and performance

The two I/O data pins, table 4.4, control the communication between the μP and peripherals. These pins are used to synchronize program-interrupt and data channel requests from peripherals. During a request-enable operation a 2-bit data called a 'request-enable code' is transmitted via the I/O data pins to achieve the synchronization of peripheral interrupt requests. During a data-channel-acknowledge operation the two data pins transmit another 2-bit code to signify the

beginning of a 'data-channel break' by the μP. It is only during these 'breaks' that the μP performs the synchronisation and later the time-critical 2-bit serial transfer of I/O data.

Table 4.4 I/O control and Data Pins

Pin designation	Function	Transfer Direction
I/O CLOCK	Information transfer synchronization	To and from μP
I/O DATA1	Carries control and data	To and from μP
I/O DATA2	Carries control and data	To and from μP
I/O INPUT	Indicates direction of transfer	From μP

Whenever the μP executes an I/O instruction following a 16-bit I/O program command it transmits a 2-bit code via the two data pins (DATA1 and DATA2) to specify a programmed I/O operation that a peripheral is to perform. During an I/O data-out operation, the 2-bit code along with a 16-bit data word are transmitted via the I/O data pins in a two-bit serial form. Also during an I/O data-in operation, the 16-bit data from the peripheral device is two-bit serially transferred in about 20 I/O clock cycles. The DATA1 pin serially transfers the high order byte (bits 0-7) and the DATA2 pin simultaneously but serially transfers the 8 bits (bits 8-15) of the low order byte of the word. The microNOVA board provides the drivers and control buffers for a 2-bit wide differential, serial I/O bus by which the μP performs its data transfers with the μC system peripherals and user-designed I/O interfaces connected to the IOC. This sequential 2-bit data transfer further explains the need for synchronism in the microNOVA 2-bit I/O bus where time requirements are over 7.2 μS per 16-bit word on inputs and 5.8 μS per 16-bit word on outputs. This architecture (figs. 4.1, 4.2) presents timing problems for fast I/O operations and in

this study the use of a programmable LSI (eg. chapter 5) in the μC interface helps (among other things) to reduce the problems (fig. 5.12). The author recommends μP's with asychronous bus interfaces (preferably 16-bits) for NC applications where DMA is highly desirable and for the LSI's discussed in chapters 5 and 6 (figs. 5.10, 6.3).

Dixon (21) correctly states that multiple register integration within the μP often limits the number of pins available for I/O and so necessitates the handling of I/O devices by a separate serial bus. This considerably limits both processing performance and configuration flexibility. Such limitations were confirmed when using this microNOVA. The present generation of it is redesigned (as mentioned in section 4.0) and offers at least three times better performance.

4.1.5 The operation of microNOVA Data Channel and Peripheral Control

The data channel is the 16-bit bidirectional peripheral of the serial 2-bit I/O bus after the IOC at the microNOVA back panel (fig. 4.1) which contains the general purpose interface I/O bus (G.P.I.O.B.). To request data channel transfer an external device causes the DCHINT line (connected via GPIOB to μP) to go low. At the beginning of every memory cycle, the μP synchronises any requests that are being made. Periodically during instruction execution, the μP pauses to service all previously synchronised requests, the servicing priority being dependent on the relative positions of the requesting devices on the I/O bus. That device which is physically closest on the bus is serviced first followed by the next closest device until all interrupt requests (DCHINT = 0) are serviced. The sequences of operation during which the μP transfers data between a memory location and an external device on the data channel take place during 'data channel breaks'.

They are performed at the request of the peripheral devices and allow the latter to transfer data to or from memory via the μP. The μP responds to interrupts by activating the data-channel-acknowledge line. In an I/O data-in operation (DATIA) the device transmits sixteen bits of data to the μP via the 2-bit serial bus and the two I/O data pins (29 and 30). This is a Read operation. In the Write (DATOA) 16 bits of data are transfered from the μP via its two I/O data pins and the 2-bit serial bus to the IOC before it reaches the external device.

This serial transfer architecture of the microNOVA again proved problematic in this activity. Therefore the system was implemented on the 16-bit buffered digital I/O interface (like the data channel, this is controlled by the IOC). It is appreciated that most μP's are generally designed on the basis of a generalised bus structure to which peripherals and memory can be attached at will (fig. 4.2a). The stepping motor controlling LSI interfaces (eg. Figs. 5.10, 6.3 and 7.3) described in the subsequent chapters can be implemented more easily with these modern microcomputers.

4.1.6 The microNOVA instruction operation

Table 4.5 shows the 16-bit instruction field of the microNOVA μP.

Table 4.5 MN601 Bit field

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
←															→
More significant												Less significant			

Information is transferred between the μP registers, the memory address/data pins and the I/O data pins during the execution of an instruction. Also arithmetic, logical and shift operations are performed by the μP using these registers (the accumulators). The 16-bits in the instruction field (with or without an accumulator) contain '1's' or '0's' (binary digits) depending on the decoded value of the instruction. In transmitting addresses or data between the μP and memory, the MB0-MB15 pins correspond to bits 0-15 of the address or data instruction. In memory reference instructions, the CPU fetches an instruction by reading a memory location and loading the contents of the location in to the instruction register. The program counter (PC) content is the address of the memory location from which the current instruction is read and so the PC register guides the μP in the program flow.

Often during the execution of some instructions, the μP calculates an indexed address by using the contents of bits 6-15 in the 16-bit instruction field. Bits 6 and 7 contain the indexed address (ie the accumulator which contains the relative address). The contents of bits 8-15 (displacement field) are added to the contents of the accumulator which has been indexed to get the final address, e.g.

```
LDA 1, @3, NX
```

This increments (@) the address indexed by accumulator 3 by 1 to fetch the data, NX, into accumulator 1. This sort of instruction is often employed to transfer data from a Fortran IV stack to Assembly level. In assembly level programming the memonics included in the instruction set are set up to instruct the μP while data and addresses are written in octal.

The μP performs arithmetic and logic operations using two accumulators and shift operations using one accumulator. It also has hardware 'multiply' and 'divide' facilities and these have execution times of 41.3 μs and 59 μs respectively.

Together with a generalised bus structure, the availability of hardware 'multiply and divide' is also necessary for a microcomputer controlled NC machine system. This makes it much less time consuming for the μC to undertake recursive calculations involving multiplications and divisions especially for feed rate control during circular interpolations using such algorithms like the MTIRA'S (90) linear approximation of arcs.

4.2 The memory modules and operation

As stated in section 4.0, the microNOVA memory is made of packages of 4K dynamic RAM'S, MN606's (fig. 4.0b). Each RAM is a 20-pin unit with three voltage inputs (-5V, + 15V, 0V), a clock input (P) a chip select (CS), two data direction (D/O, D/I) a write enable (WE) and 12 address pins (A0-A11). Being dynamic, the RAM's are refreshed by the μP periodically in groups of 512 words giving a total of 64 groups.

When P goes high a memory operation is started and A0-A11 are sampled by the μP to determine their logic levels. The level at CS at this time determines whether the operation is a Read or a Write (CS = 0, for Read and CS = 1, for Write). For a Read the D/O pin should go low while D/I goes high and vice versa for a Write operation. As the write operation is initiaited the WE pin goes high thus enabling the action.

When P rises the chip select pin (CS) goes high and this initiates a refresh operation which is finally accomplished by the falling edge of the signal on WE. The memory operation is concluded when P goes low finally. The way in which the μP control and addresses the memory has been discussed in section 4.1.2. Since a 15-bit program counter-register is used the μP can address up to 2^{15} (=32768) locations and so this earlier microNOVA is expandable up to 32K. Reference (11) states that the redesigned microNOVA can support up to 64K words of memory.

4.3 The MicroNOVA I/O Controller (IOC)

Like the µP, the I/O controller (IOC) is a 40-pin LSI (fig. 4.0c) in which two I/O data pins (corresponding to those described under section 4.1 for the µP) are used for the transfer of two-bit serial data to or from the µP. Data transfers with peripheral devices are handled with the 16 bits, DO-D15, of the IOC. The 2 clock pins are designated as PHASE A and B. Table 4.6 shows the general functions of the I/O pins. Generally, the IOC is the interface between the external devices and the 2-bit serial bus. The protocols followed by the IOC in the transfer of data and control to/from the µP is identical to that followed by the µP in communcation with the peripherals. During an I/O data-out operation (from µP), the two I/O data pins (in IOC) each receive one bit at a time until all sixteen bits are transferred.

For the conversion of the 2-bit serial I/O µP data to the 16-bit parallel data path of the IOC, a dual eight-bit I/O shift register (IOSR) is used on the IOC chip. The I/O CLOCK, I/O DATA1 and I/O DATA2 and the transfer direction-selecting I/O-INPUT pins function in a manner similar to that described for the µP (section 4.1.4). The PHASE A and B clocks originate from the µP and are derived from the µP-driving BMCLOCK and $\overline{\text{BMCLOCK}}$. Both of these clocks are also derived from the external 8.33 MHZ master clock. This facilitates the request and data synchronisation performed by the µP during I/O transfers involving the IOC and the data channel.

The sequential 2-bit serial data transfer to or from the IOC is identical to that described for the µP operation in section 4.1.3. The I/O DATA1 serially transfers the high order eight bits and the I/O

DATA2 serially transfers the low order eight bits. As was mentioned in the previous sections of this chapter this I/O architecture requires time-critical processes on every data bit before it reaches its final destination and so the I/O bus has to be synchronous.

Table 4.6 IOC pin functions

Pin Designation	Character	Functions
PHASE A	Clock Pins	Receiving timing information
PHASE B		from μP
I/O CLOCK	I/O data	Data transfer via
I/O DATA1	pins	the IOC I/O transceiver
I/O DATA2		to the 2-bit serial bus
I/O INPUT		
INTP	Priority pins	Receiving data channel
DCHP		priority signals from μP
FSTROBE	Function code	Transmit control information
F0-F3	pins	to IOC components
D0-D15	Data pins	peripheral data transfers
BUSY, DONE	busy/done	Transmitting and receiving
INT, INTR	request pins	control pulses
DCH SYNC	Data channel	Receiving data channel
DCHR	request pins	requests by synchronization

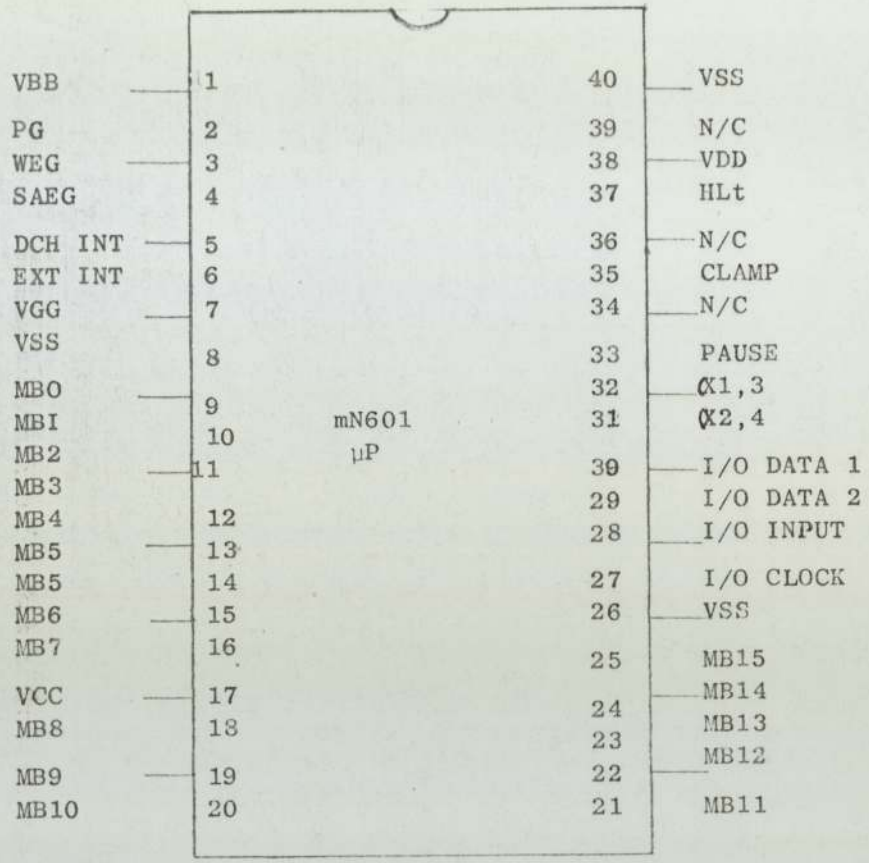


FIG 4.0a The MN601 microprocessor LSI

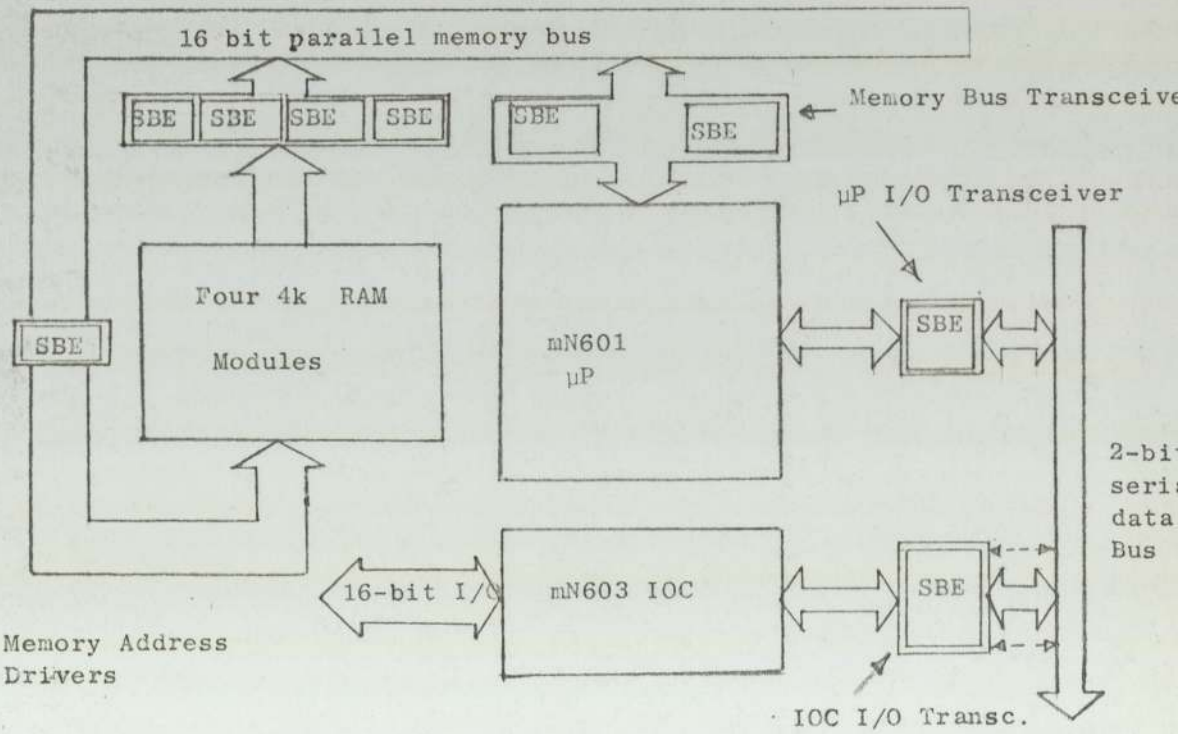


FIG 4.1 microNOVA Microcomputer Architecture

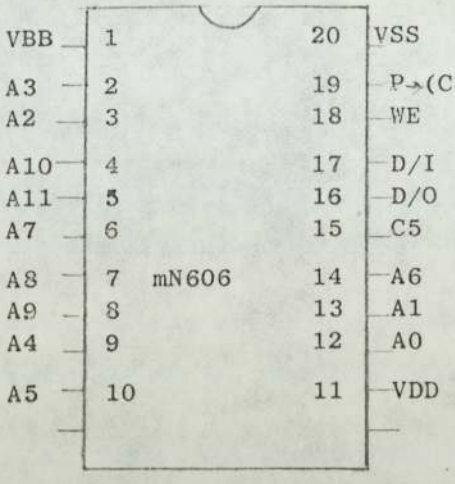


FIG 4.0 b

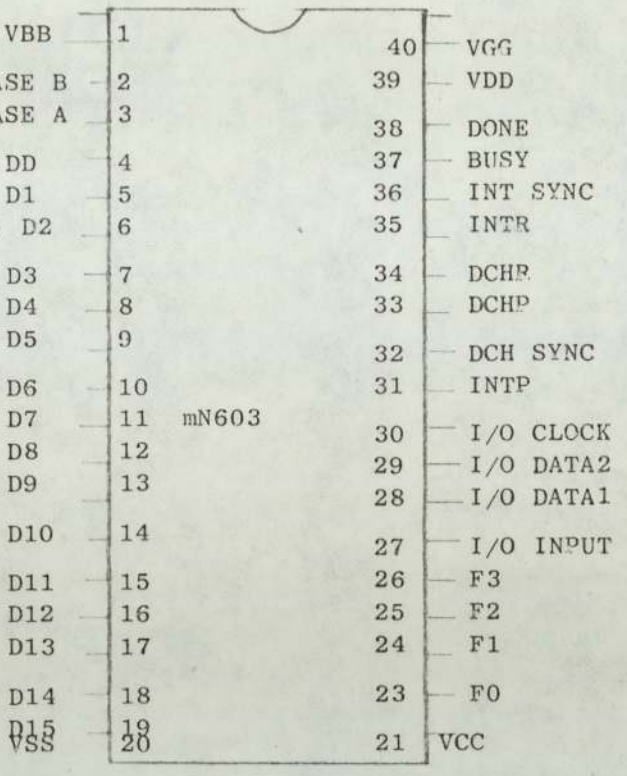


FIG 4.0C

4-K RAM and I/O Controller Chips

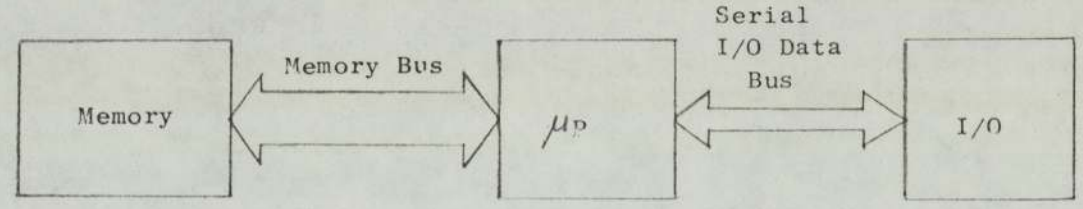


FIG 4.2b Parallel Memory and Serial I/O Data Buses

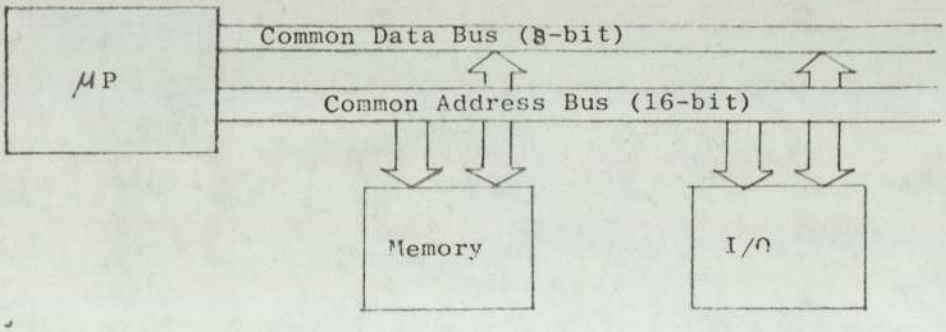


FIG 4.2a Standard Parallel Bit Buses

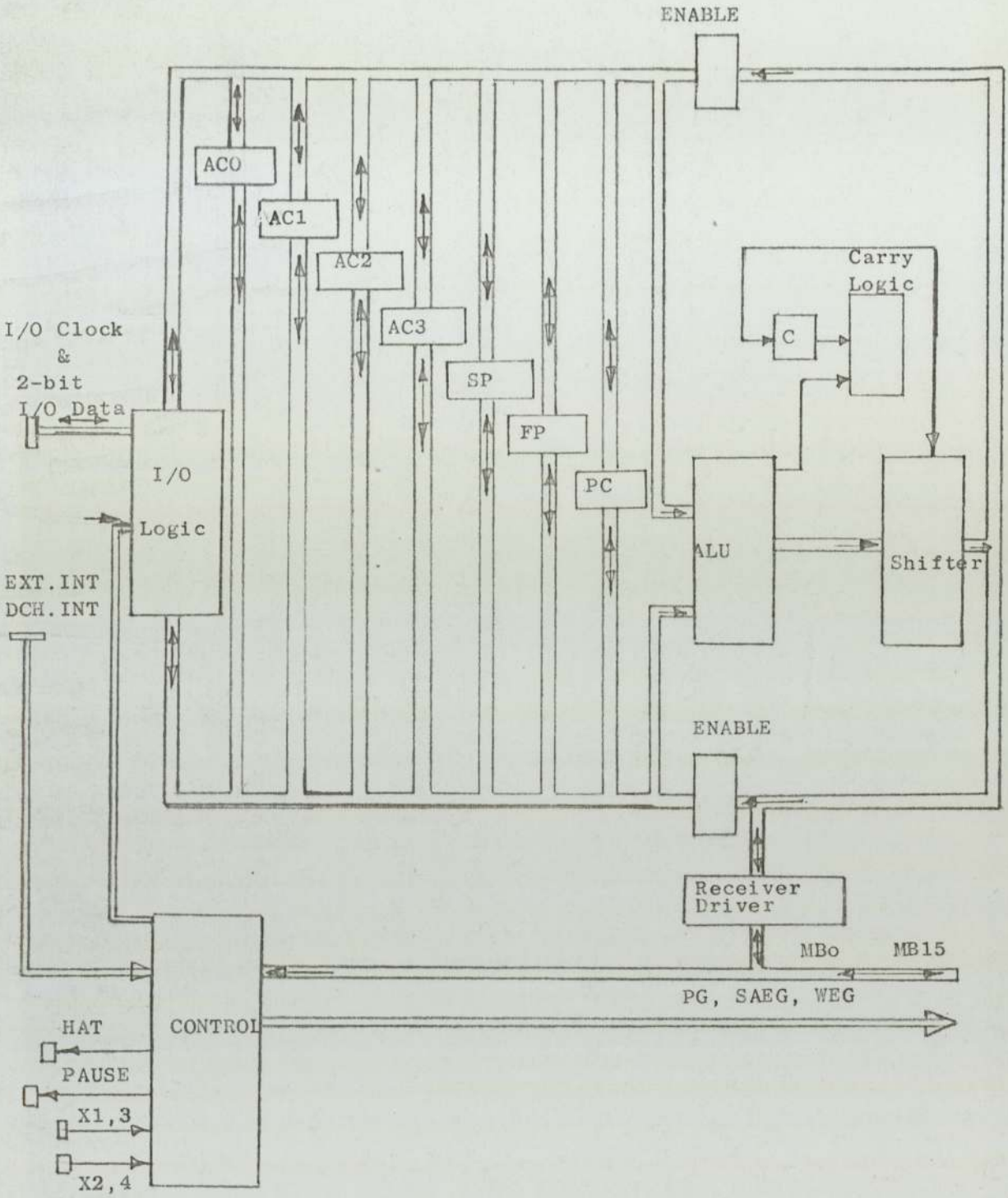


FIG 4.3 The MN601 and Register Organisation

4.3.1 The control of the functions of the I/O controller by the Microprocessor

On executing the appropriate I/O S/W instruction the μP controls external devices via the four function pins available on the IOC. Table 4.7 shows the 16 function code pins and their decoded logics representing the relevant I/O commands issued in the control programs.

Table 4.7 Function codes of the microNOVA IOC

F0	F1	F2	F3	Name	S/W command	Remark
0	0	0	0	DATOA	DOA	Output via A Buffer
0	0	0	1	DATIA	DIA	Input via A Buffer
0	0	1	0	DATOB	DOB	Output via B Buffer
0	0	1	1	DATIB	DIB	Input via B Buffer
0	1	0	0	DATOC	DOC	Output via C Buffer
0	1	0	1	DATIC	DIC	Input via C Buffer
0	1	1	0	STRT	START	Initialise/Start
0	1	1	1	CLR	CLR	Clear/Reset
1	0	0	0	*IOPLS	IOPLS	I/O Pulse
1	0	0	1	IORST	IORST	I/O Reset
1	0	1	0	MSKO	MSKO	Mask Out
1	0	1	1	DCHA	DCHA	Data Channel Acknow.
1	1	0	0	DCHI	DCHI	Data Channel Input
1	1	0	1	DCHO	DCHO	Data Channel Ouput
1	1	1	0	WCEZ	WCEZ	Word Count Equal Zero
1	1	1	1	NOP	NOP	No Output Pulse

*Asserted by the IOC for everyI/O non-skip instruction

The operations performed by the IOC as indicated on the function code pins during each 'F cycle' is determined by the time interval during which the FSTROBE pin is in the low state. The F(0 - 3) codes transmit addresses to the external devices like data channel and digital I/O interface buffers, A-C.

e.g. LDA 2 NY
DOB 2 DIO

This program command instructs the μP to output the number NY from accumulator 2 to the B buffer in the digital I/O interface.

Using the data channel for I/O operations requires no specific command from the program (they occur at similar speeds to those through other I/O interfaces) but the transference is preceded by several program commands which acknowledge interrupts from external devices priodically. When to acknowledge the interrupts is determined by the μP and this happens during microNOVA's periodic 'data channel breaks'. For external devices to use the data channel facility they must be able to transfer 16-bit word addresses, D(0-15)L, to the memory locations containing the required 16-bit, D(0-15)H, data.

4.3.2 The Busy/Done and Interrupt Requests

Fig. 4.4 illustrates the relation between BUSY, DONE, INT SYNC and INTR pins. The RQENB signal goes high to enable I/O devices to interrupt the μP during which period the INTR (interrupt request) line goes low. When a data channel device makes an interrupt request via the IOC, the DCHR pin (table 4.6) goes low. This is preceded by the assertion of the DCH SYNC line by the external

device connected to the data channel. After the data channel acknowledge (DCHA) codes have been transmitted the memory address and later the data content are transfered via the D(0-15)L and D(0-15)H. The design of the μC is such that interrupt servicing begins with that device (identified with a device code) which is physically closest to the μP on the I/O bus using the polling technique (checking the DONE flag of every device) unless the program interrupt service routine stipulates the priority for interrupt servicing.

Every device connected to the I/O bus has a device number and mnemonic (e.g the digital I/O interface is called 'DIO' with device code 42) and 5 as a priority mask bit. Using the 'DOAS' command the device is enabled or disabled to interrupt the CPU by writing a '0' or a '1' respectively into the priority mask bit. When the conditions exist for a device to interrupt the μP, that device places a logic 1 onto the INTR line fig. 4.4. The μP tests the INTR line during every memory cycle. If a request exists the μP resets its ION (interrupt on) flag and thus disallows further interrupts, saves the contents of the program counter in memory location '0' and performs an unconditional jump to location 1 in which the address of the interrupt servicing routine should be placed. Then the rest of interrupt servicing is performed in accordance with the instructions contained in the program service routine.

In this study the digital I/O interface was employed to program SY6522 (general purpose LSI) to generate timed pulses to the SM's (fig. 5.12). On count down the LSI's interrupt the μ C for further programming.

Figs 5.10 and 6.3 show more suitable bus connections using a more standard μC (16-bit parallel address and 8-bit parallel data buses).

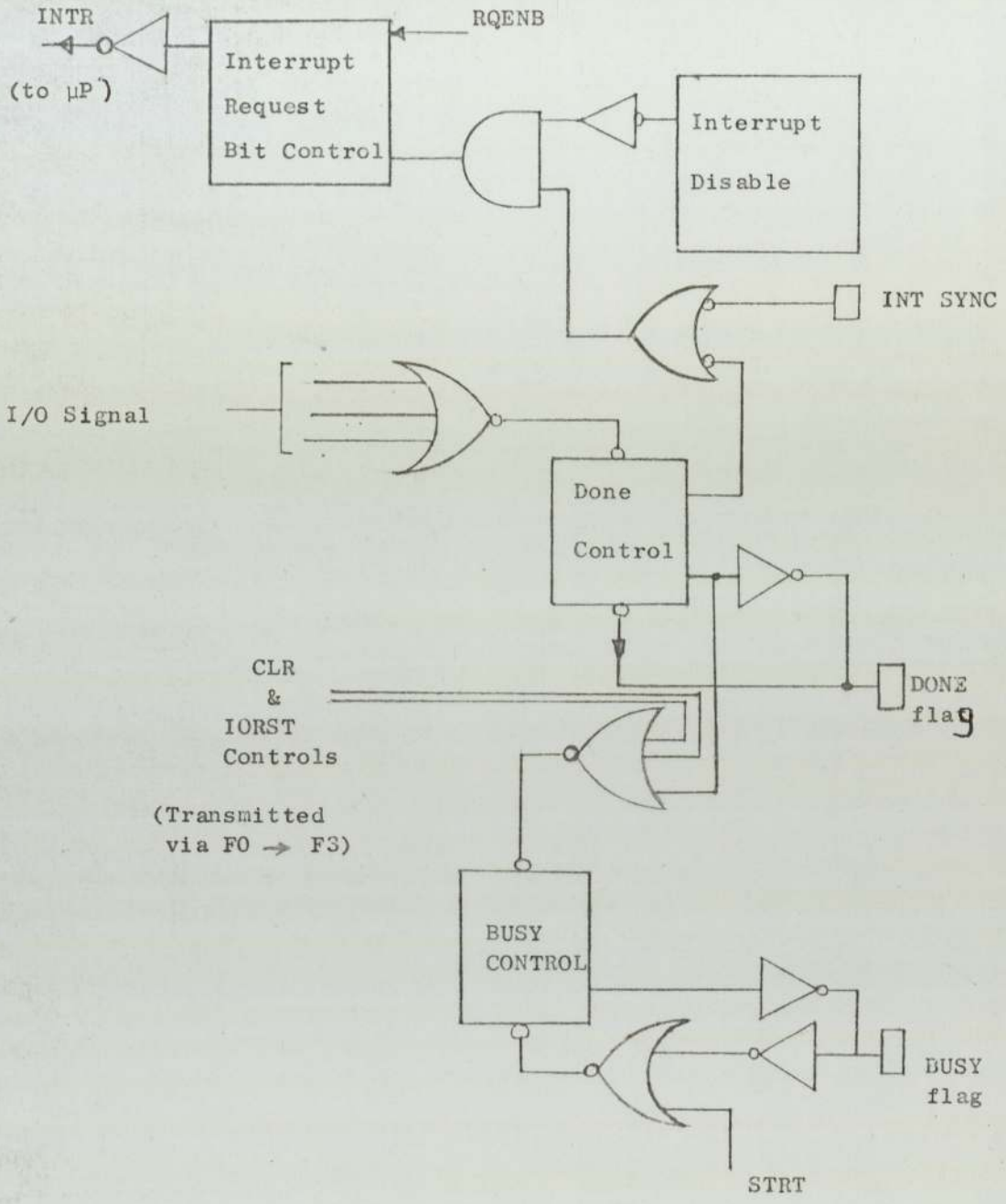


FIG 4.4 IOC Pin Interactions During Interrupts

CHAPTER 5

THE USE OF SY6522 LSI FOR INTERPOLATION FEEDRATE, POSITION AND MISCELLANEOUS CONTROLS

5.0 Introduction

A reasonably cost-effective microcomputer controlled NC (stepping motor) system for a small firm should require minimum possible time and expertise to develop and when in operation allow the user a good degree of general access to the µC. It should be built around a general purpose µC with good S/W development aids and support (section 2.1). Unlike the more expensive systems described by Plas and Blommaert (7), Prasaad (23) Popa (20) and some others described in section 2.0-2.3, the system described in this chapter employs powerful 16-bit interval timers included in a 40-pin CMOS general purpose µP-I/O LSI. The LSI, which is the main concern of this chapter is Synertek's SY6522, Versatile Interface Adapter (VIA), figs 5.1 and 5.2. This device contains the pair of accurate interval timers, a serial-to-parallel and parallel-to-serial shift register and input data latching on its peripheral ports. It has handshaking capabilities to facilitiate control of bi-directional data transfers between the LSI and peripheral devices on a machine interface.

SY6522 can control devices through two 8-bit bi-directional ports, PA and PB. Each of these lines can be programmed to act either as an input or output. Several peripheral I/O lines can be controlled directly from the interval timers for generating programmable frequency square waves and/or for counting externally generated pulses. These features enhance the µC programmability of this LSI to aid in the control of stepping motor NC systems economically.

During this study one SY6522 was assigned to control each axis (X,Y) of the NC milling machine. Chapters 6 and 7 describe feasible modifications of this system with two other μC programmable LSI's (recently developed) to further improve the control capabilities.

Unlike BRM outputs (section 2.7) which are irregular, the integrated interval timer, T1, generates precisely spaced and interpolated pulses to the stepping motor drive logic board. Moreover, the system with SY6522 has the added bonus of another interval timer, T2, which can be programmed to count the interpolated pulses and to generate interrupts to the microcomputer. In order to facilitate control of the many powerful features of this LSI, the internal registers have been organised into an interrupt flag register, an interrupt enable register and a pair of function control registers. In all, this LSI contains 16 registers which can be accessed by the μC via the SY6522/microcomputer bi-directional bus.

This system has been tested with the microNOVA (Chapter 4). The input (MDI) data, their interpretations and the calculations for linear interpolation and feedrate control are handled in FORTRAN (control initialisation, Chapter 9) which then transfers the variables to the low level control in Assembly Language (section 8.7). At this level SY6522 LSI is programmed to perform some of the functions described in this chapter.

5.1 The LSI (SY6522) Buses and Control Lines

This section describes the access lines which interface SY6522 to the system μC. Data transfers between the LSI and the μC take place only while the 'Phase Two Clock', ϕ_2, is high. ϕ_2 also acts as the time base for the various timers, shift registers (SR), peripheral control registers (PCR) and anxiliary control registers (ACR). For better synchronisation, the ϕ_2 clock is derived from the μC master clock. The ϕ_2 clock used in this study has been scaled down to a period of 2.4 μs. (The minimum period recommended for SY6522 ϕ_2 clock is 1 μs).

5.1.1 Chip Select lines (CS1, CS2)

As the name implies, these lines are used to select or address the appropriate SY6522 chip in the μC NC system. The selected internal register in this LSI is accessed when CS1 is high and CS2 is low, the former being the one used in this study, ie. pin 24 (fig. 5.1), the latter (pin 23) is connected permanently to ground.

5.1.2 The Register Select Lines

The four register select lines (RS0, RS1, RS2, RS3) are connected to the μC address bus lines to allow the processor to select the appropriate internal SY6522 registers. The sixteen internal registers are individually defined by the 16 possible decoded combinations of the four register lines, pins 35 to 38 (figs 5.1, 5.2). These registers and their functions are described in table 5.0 below.

*Table 5.0 Organisation of SY6522 Registers

REGISTER ADDRESS					
RS3	RS2	RS1	RS0	REGISTER NAME	FUNCTIONS**
0	0	0	0	IRB,ORB	I/O bit and handshake
0	0	0	1	IRA,ORA	Controls for PB and PA Ports
0	0	1	0	DDRB	Data Direction Control for PB ports
0	0	1	1	DDRA	Data Direction Control for PA ports
0	1	0	0	TILL	Write Low Order Latch of T1
0	1	0	1	TICH	Write High Order Counter of T1
0	1	1	0	TILL	Write Low Order Latch T1
0	1	1	1	TILH	Write High Order Latch T1
1	0	0	0	T2LL	Write into Low Order Latch T2
				T2CL	Write into Low Order Counter T2
1	0	0	1	T2CH	Write into High Order Counter T2
1	0	1	0	SR	Shift Register Address
1	0	1	1	ACR	Auxiliary Control Register Addresses
1	1	0	0	PCR	Peripheral Control Register Addresses
1	1	0	1	IFR	Interrupt Flag Register Address
1	1	1	0	IER	Interrupt Enable Register Address
1	1	1	1	ORA	No Effect on PA handshake

* 1 2.4V

0 0.4V

** Tables 5.2 and 5.3 give further details of the control (including the 'Read') of the two interval timers, T1, T2.

5.1.3 The µC Control of SY6522 Data and Address Buses

After a particular register has been selected via RS0-RS3, data is transfered to it by the µC through the 8-bit bi-directional Data bus corresponding to pins 27-33 (ie. DB7-DB0), figs. 5.1 and 5.10. The direction of data transfer is controlled by the read/write (R/W) lines.

If R/W is low and the chip is selected, data is transfered out of the μC into the addressed SY6522 register (ie write operation). On the other hand the μC reads data from a selected SY6522 chip and register by setting the R/W line high. Figures 5.3 and 5.4 show the necessary Write and Read timing considerations. For example, in fig. 5.3, when the chip is selected with R/W low and phase two clock is high ($\phi_2 = 1$), the data on the data bus is transfered into the addressed SY6522 register for a Write operation.

At power-on the reset input, externally derived from a '555' timer circuit clears all internal registers to logic 0 (except T1, T2 and SR). This forces all the peripheral interface lines (PA's and PB's) to the input state, disables the timers T1 and T2, the shift register and the other I/O ports. Interrupts from the chip are also disabled and the internal drivers will remain in the high-impedence state except when the chip is selected (ie CS1 = 1 and CS2 = 0). The internal chip access control contains the necessary logics within SY6522 to detect the chip select condition and to decode the register select inputs to allow accessing the desired internal registers. In addition, the R/W and ϕ_2 signals are employed to control the timing and direction of data transfers between the μC and SY6522. During a Write operation, the μC first transfers the data into an input data register latch when $\phi_2 = 1$, then the next ϕ_2 cycle finally transfers this data into the selected internal register. When the μC reads the LSI, data is transfered from the desired internal register directly onto the Data Bus during ϕ_2 cycle.

5.2 Control of External devices through SY6522 peripheral interface

The buses and control lines which are used to drive peripheral devices under the control of the internal SY6522 registers are the following:-

(a) the peripheral A Port (PA0-PA7 ie pins 2-9)

(b) the peripheral A control lines (CA1 and CA2, ie pins 40,39)

(c) the peripheral B Port (PB0-PB7 ie pins 10-17)

(d) the peripheral B control lines (CB1 and CB2, ie pins 18, 19)

Each of the peripheral lines is equivalent to one standard TTL I/O load.

(a) and (c): The peripheral A and B ports consist of 8 lines each. These lines can be individually programmed by the µC to act as I/O under the control of the Data Direction registers (DDRA and DDRB, Table 5.0). The polarity of the output pins is controlled by output registers (ORA, ORB) and input data is latched into a chosen internal register under the control of the CA1, CB1 lines. The different operational modes in which the PA, PB lines function are initialised into the appropriate internal control registers, (Table 5.1) by the µC. In addition, the polarity of the PB7 (pin 17) output signal is controlled by one of the internal 16-bit interval timers (T1) while the second 16-bit timer (T2) is programmed to count pulses input to the PB6 pin (ie pin 16). Section 5.4 gives further details of the control registers by which PB7 and PB6 are programmed to generate and count the pulses sent to each stepping motor.

(b) and (d): The CA1 line controls the latching of data on the peripheral A port input lines (PA0-PA7) and CB1 can be programmed to act as a serial output of the pattern loaded into the 8-bit shift register (SR) (if this

feature is desired, eg. to drive axis indicator lights) In addition CA1 and CB1 can be initialised (table 5.1) via the peripheral control register (PCR), Table 5.0, to monitor the state of the machine being controlled. They perform this function by acting as intermediaries of the interrupts between the machine and the μC in features like machine axis servo state 'watch dogs' (eg. for protection of the Y-axis servo linear potentiometer, section 3.4)

Table 5.1 Initialisation of the Peripheral Control Register

BIT NO (DB7-DB0)	7 6 5	4	3 2 1	0
FUNCTION	CB2 CONTROL	CB1 CONTROL	CA2 CONTROL	CA1 CONTROL

CA1 Control: When the μC writes a logic '0' into bit 0 (ie DB0, pin 33, table 5.1) of the peripheral control register, then CA1 is initialised (prepared) for the setting of its interrupt flag when a signal from the machine (connected to pin CA1) changes state from high to low. If a logic '1' has been intially written into Bit 0 (DB0) by the μC the interrupt would be set when the NC machine sends a voltage (up to 2.4V) signal to pin CA1

CB1 Control: This operates in exactly the same manner as CA1, described above, except that as Table 5.1 indicates, Bit no 4 (DB4) now becomes the initialising bit.

The interrupt so generated from the machine axis to CB1 (or CA1) is transmitted internally from SY6522 to the microcomputer. The μC reads the SY6522 Interrupt Flag Register (Tables 5.0 and 5.2) thus identifying the source of the interrupt and so services the machine.

After this, the interrupt is cleared by Reading the ORB (for CB1) or ORA (for CA1) registers using addresses '0001' or '0000' (table 5.0) respectively.

5.2.1 Read and Write Handshaking between the μC and SY6522

The SY6522 LSI allows very positive control of data transfers between the μC and the system peripheral devices through the operation of 'handshake' lines. These machine interface devices can be the controls of any of the machine miscellaneous functions (job clamping, coolant control, tool setting) and the SY6522 registers themselves. The internal organisation of the peripheral control ines, CA1, CA2 and CB1, CB2, within the LSI is such that CA1 and CA2 lines can handshake data on both read and write operations. Therefore it is proper to employ the port B lines (CB1, CB2) for the axis monitoring (malfunction watch-dog) operation discussed under CB1 control in section 5.2, while the Port A lines (CA1 and CA2) control the handshaking operations (fig. 5.6) on both read and write from and to PA devices respectively. Figure 5.6 illustrates possible inclusion of the standard IEEE 488 (ref. 100) microcomputer bus interface which provides control line ('Date Available' and 'Data Accepted') compatibility with SY6522 CA1, CA2 lines.

Read Handshake: In μC controlled machine application a 'job-in-position' (job clamped) signalling peripheral device will generate 'Date Available' to indicate to the μC via CA1 (pin 40) that valid data is present on the peripheral port. This signal normally interrupts the μC which then reads the data causing the generation of a 'Data Taken' signal which can be used to reset the peripheral device. In SY6522, automatic 'read' handshaking is possible only on these CA1 and CA2 lines. The CA1 interrupt

input pin (section 5.2) accepts the 'Data Available' signal and CA2 generates the 'Data Taken' signal. The 'Data Available' signal sets an internal SY6522 flag which interrupts the μC (or which is polled under S/W control). The 'Data Taken' signal can be either high level logic which is set low by the μC or a pulse.

Write Handshake

The procedures for handshaking data from the μC to a peripheral device (via the data bus and PA lines) are similar to those described for the Read Handshake. The difference is that this time the μC would generate the 'Data Available' signal (via SY6522) to the peripheral device which then respond with the 'Data Taken' signal to the μC.

The use of single bits out of any of the 8 PA lines is simplified by the fact that each of the peripheral pins is controlled by a bit in the output register (ORA, Table 5.0) and an Input register (IRA, Table 5.0). The bits of these registers are controlled by writing '1' or '0' (via DB0-DB7) to the register (IRA or ORA), thus controlling the coresponding pins in PA0-PA7 at the peripheral interface to the external devices. A '1' in the output register causes the corresponding PA pin to go high and a '0' causes the pin to go low. It should be noted that if data were written into output register bits corresponding to pins which had been initialised by the program to act as inputs only, these pins would not be affected. Reading a peripheral port causes the contents of the Input Register (IRA) to be transferred onto the μC/ SY6522 Data Bus. This register will always reflect the data on the PA pins. Figures 5.7a and 5.7b show the timing diagrams for read and write handshaking respectively.

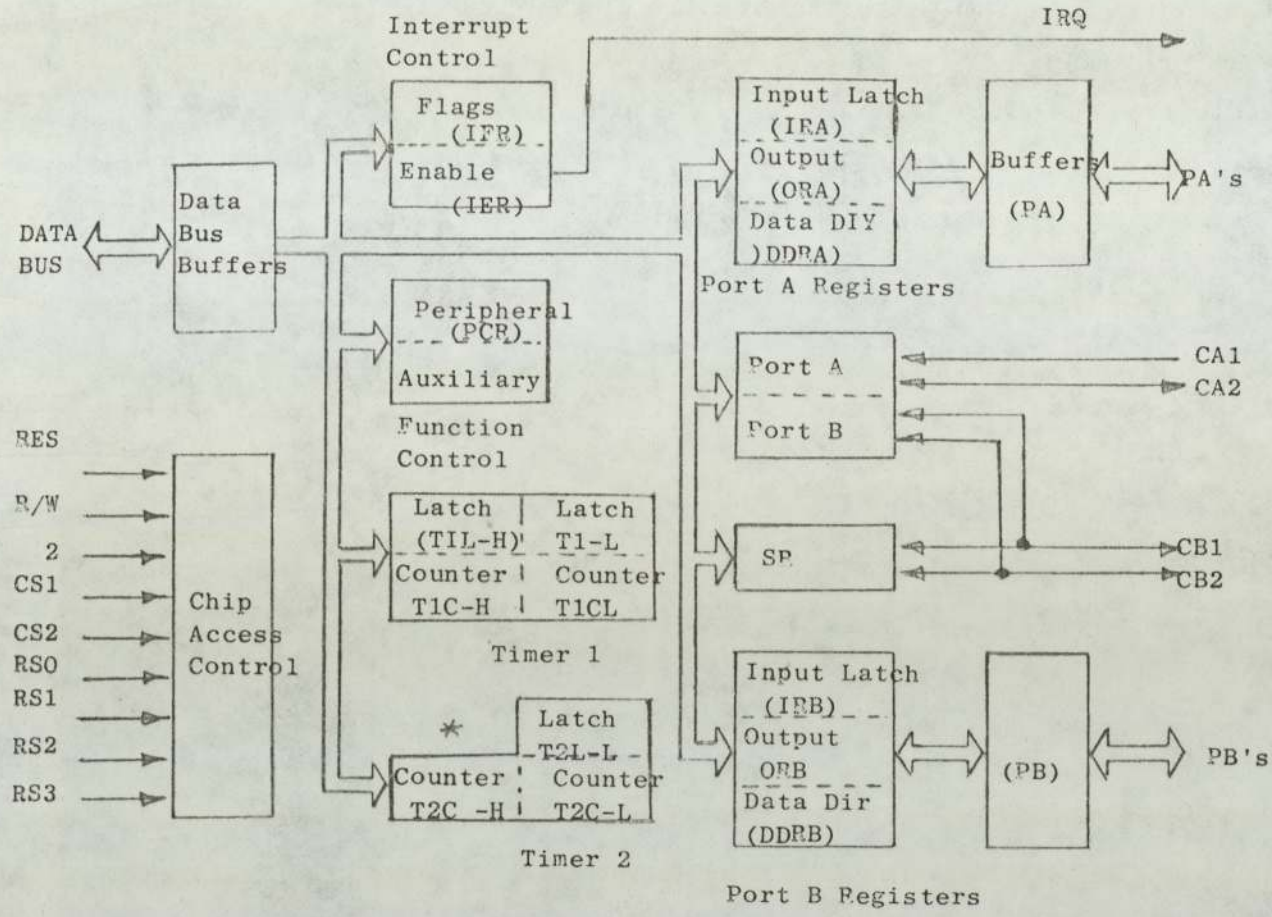


FIG 5.1 The SY6522 Function Block Diagram

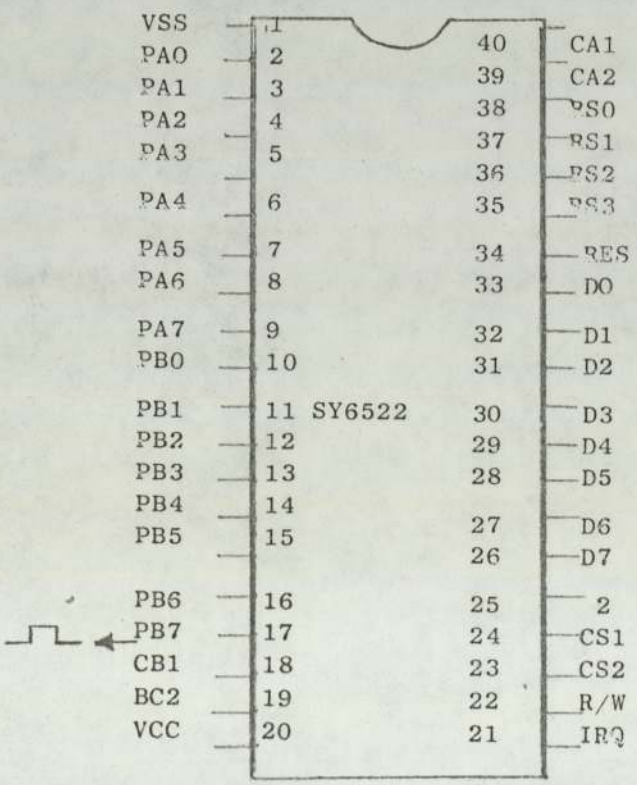
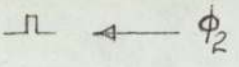


FIG 5.2 SY6522 LSI



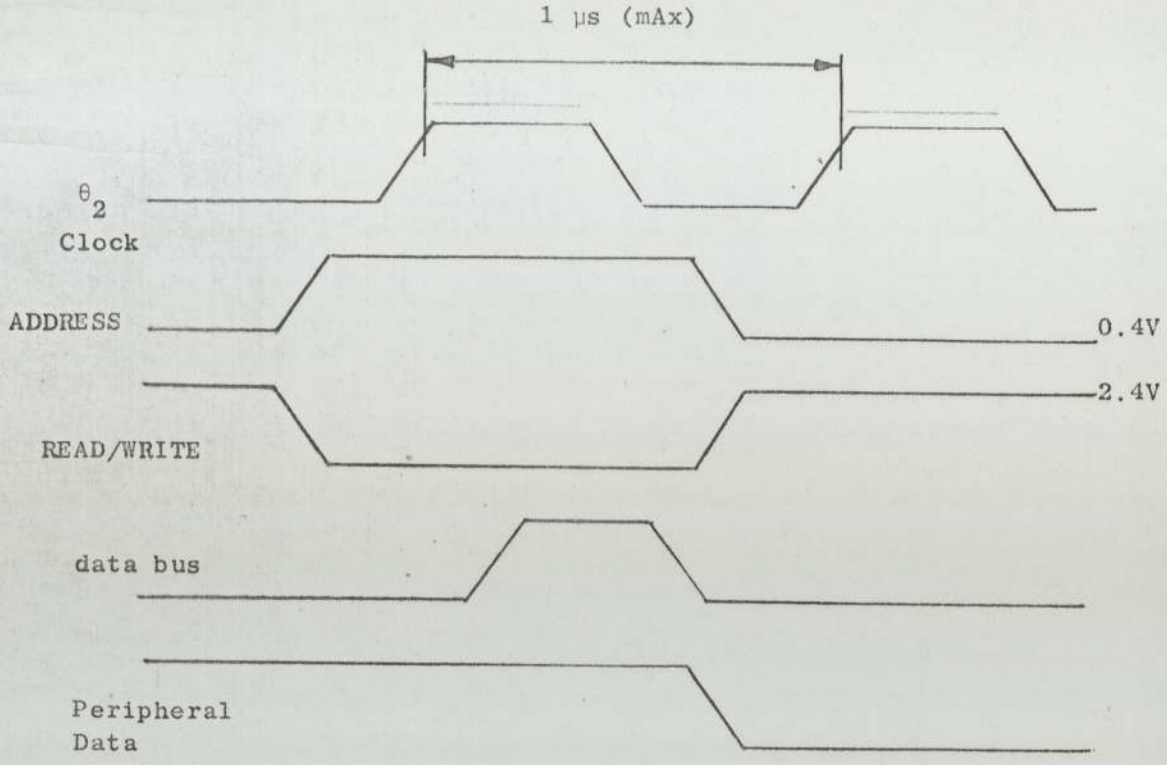


FIG 5.3 SY6522 Write Timing Diagram

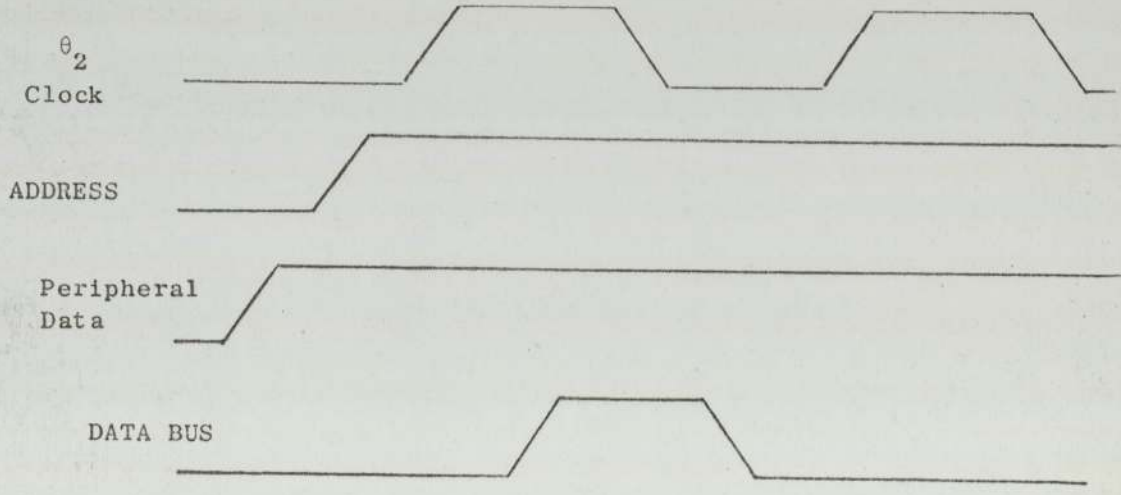


FIG 5.4 SY6522 Read Timing Diagram

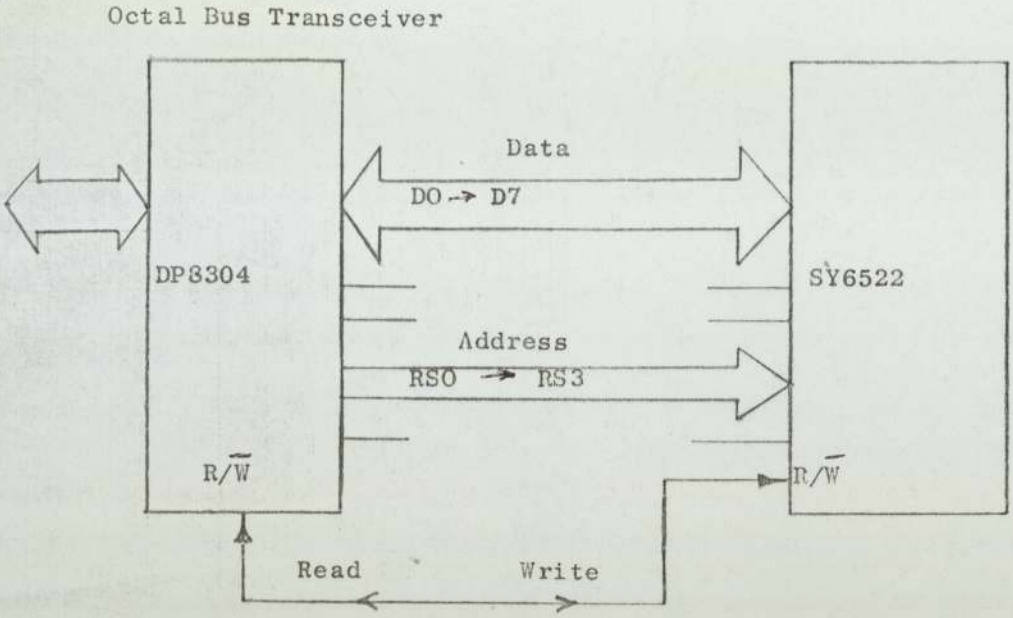


FIG 5.5 SY6522 Read/write control

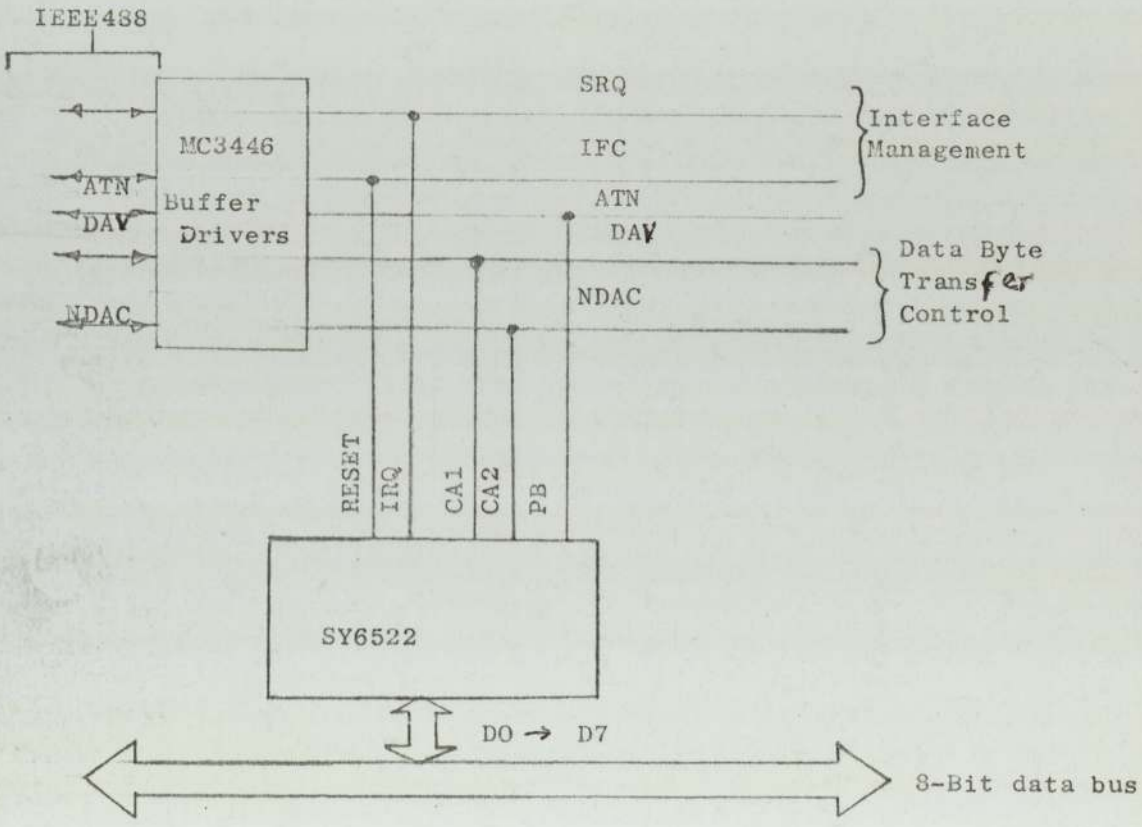


FIG 5.6 SY6522 and IEEE488

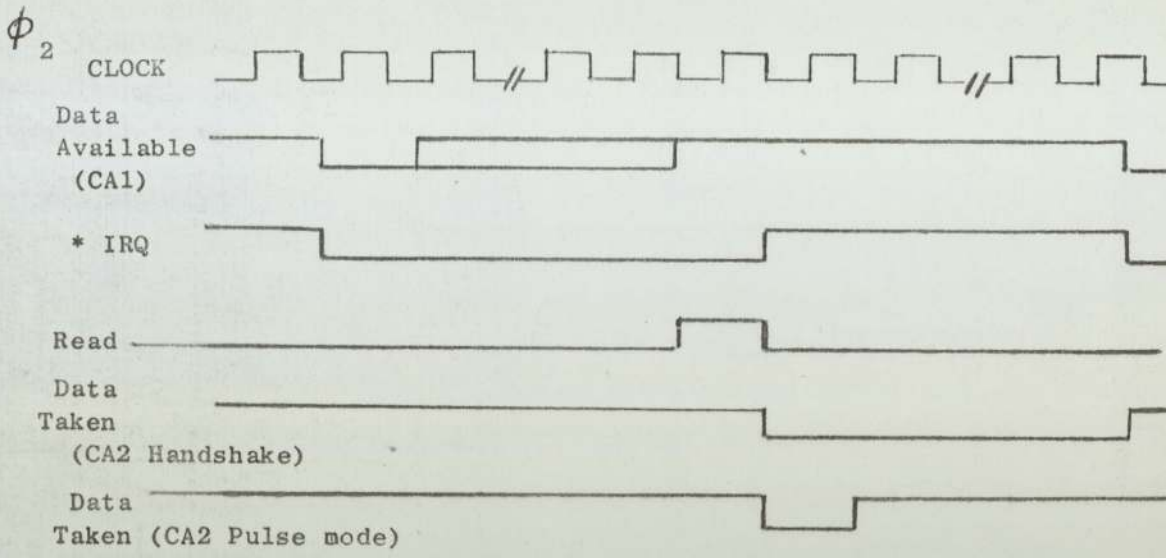


FIG 5.7a SY6522 Read handshake, timing sequence

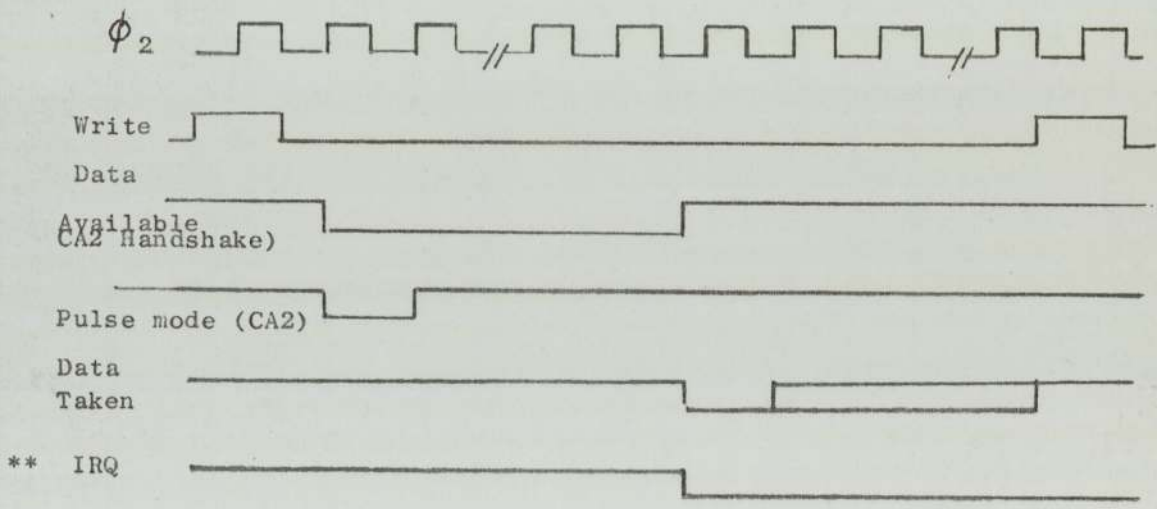


FIG 5.7b SY6522 Write Handshake timing sequence

* SIGNALS "DATA AVAILABLE" TO μC

** SIGNALS "DATA TAKEN" TO μC

5.3 The organisation and control of Interrupt within SY6522 LSI

Table 5.2 shows the internal organisation of the interrupt system of the LSI. The Interrupt Flag Register (IFR) is 8 bits wide and the 8 bits are allocated to the 8 registers (table 5.2). When conditions exist for any of the function registers or timers to generate an interrupt, bit 7 of the IFR (IRQ) goes low. The logic function to determine the state of the IRQ output is given in equation 5.0. IRQ ultimately interrupts the μC.

$$IRQ = IFR6 \times IER6 + IFR5 \times IER5 + IFR4 \times IER4 + IFR3 \times IER3 + IFR2 \times IER2 + IFR1 \times IER1 + IFR0 \times IER0 \qquad 5.0$$

where, x = logic AND operation

\+ = logic OR operation

IER = Interrupt Enable Register

For each interrupt flag in IFR, there is a corresponding bit in the Interrupt Enable Register. The μC can set or clear selected bits in the IER and this simplifies the control of individual interrupts without affecting the others. To clear a selected interrupt, the μC addresses the IER (Table 5.0), ie RS0-RS3 = 0111, and with bit 7 of the data on the data bus (D0-D7) equal to 0, each 1 in bit 6-0 clears the corresponding IER after a Write operation. Each 0 in bits 6-0 does not affect the corresponding bit. Equally the μC can set up an interrupt within the LSI by placing a 1 in bit 7 of IER and writing a 1 into the appropriate function bit. In the usual

case where an interrupt automatically emerges as a result of function timing within the LSI the μ C can read the contents of both the IFR or IER. By giving the appropriate address and on the R/W line going high, the contents of the selected interrupt register will appear on the D0-D7 data bus. By polling this data within an accumulator (using a shifting or masking operation) the exact source of the interrupt can be detected and so serviced.

Table 5.2 Organisation of SY6522 Interrupts*

	REGISTER BIT							
REGISTER NAME	7	6	5	4	3	2	1	0
Interrupt Flag Register (IFR)	IRQ	T1	T2	CB1	CB2	SR	CA1	CA2
Interrupt Enable Register (IER)	Control	T1	T2	CB1	CB2	SR	CA1	CA2

*The CA and CB lines are controlled by the PA output registers (ORA) and the PB output registers (ORB) respectively.

The T1 and T2 timer interrupts are set by the time-out of either of them (if enabled) and cleared by the reading or writing of the low order counter or the high order counter respectively.

5.4 The organisation and control of the interval timer used for interpolation and Feedrate control

As was mentioned in section 5.0 SY6522 has the special feature of being equipped with two internal 16-bit timers, T1 and T2. Lo and Lu (98) describe a different digital programmable frequency multiplier which is functionally similar to the BRM technique already described in section 2.7 and used by Prasaad (23). These are possible solutions to the control of a stepping motor but SY6522 has the extra benefit that it generates evenly spaced and interpolated pulses and still retains its functions as a general purpose μC I/O LSI. Within the same chip (fig. 5.1) the square waves, programmed by T1 are counted by T2, thus reducing the 'chip count'. Table 5.3 shows the bit positions of the Auxiliary Control Register (ACR) which controls the functions of the two interval timers.

Table 5.3 The Auxiliary Control Register and the Organisation of T1, T2

Bit No	7	6	5	4	3	2	1	0
Function	T1 Control		T2 Control	Shift Register Control			PB Latch Enable	PA Latch Enable

5.5 The program initialisation of the stepping pulse frequency generation modes and feedrate control

There are two modes in which T1 can be programmed to operate, namely (i) the 'one-shot' mode and (ii) the 'free-running' mode.

(i) The one-shot mode (fig. 5.8a)

The μC calculates a number which will be loaded into T1 to determine the period of each pulse. The one-shot mode is initialised by placing a '1' in bit 7 of the ACR (ie ACR7 = 1) and placing a '0' in ACR6. In this mode T1 is programmed to produce a single negative pulse on PB7 peripheral pin and a single interrupt on each occasion the high order counter of T1 is written by the μC. PB7 then acts as the output of a single programmable width pulse. At the end of the pulse the T1 interrupt flag is set to request the μC for further control data. The T1 counter will start decrementing the earlier number but no external pulse is generated until the μC intervenes again. The μC can read the current contents of T1 counter to determine the time since the last time-out. This feature can be employed for the step-to-step SM control technqiue discussed in section 2.7 if the μC is powerful enough for such an interpolation control. As was mentioned earlier, it uses up too much computer time and involves more S/W costs.

(ii) The free-running mode (fig. 5.8b)

This is initialised with ACR7 = 1 and ACR6 = 1. In this mode T1 outputs a continuous series of evenly spaced pulses and interrupts (if enabled). The frequency of these square waves emerging from PB7 is not affected by the response time of the μC to any interrupts in the system because the timer automatically transfers the number in its latches into its 16-bit counter to determine the duration of the next pulse. The period of any subsequent pulse can be altered easily by the μC which loads a new data number in the latches without disturbing the construction of the preceding pulse. Acceleration and deceleration of the SM is achieved by the μC responding to the interrupts following each pulse with new data for the T1 latches. Thus it is possible to construct a chosen pattern of feedrate control. At constant feedrate the μC is free to see to other CNC functions until interrupted by Timer, T2, (PB6 input) which acts as the 16-bit counter of the PB7 pulses. The μC can make occasional checks on the pulse frequencies when using the SY6522's for two-axis control and thus monitor the stepping progress.

5.6 Pulse counting for interpolation and position control using Timer, T2

T2 is a 16-bit counter for the pulses fed to the PB6 input. In this design a predetermined position data is loaded into T2 by the microcomputer. T2 then decrements at the rate of the interpolated pulses until time-out after which it generates an interrupt (IRQ) to request attention from the μC (section 4.3.2). By writing a fresh position data into T2 it is enabled to generate another interrupt on the next count down.

In the 16-bit μC one of the bits is used as a sign bit during linear interpolation calculations. Therefore, when working in single precision the maximum value of integer number involved in the calculations is:

$$(2^{16-1}-1) = 32762$$

If N (x, y) is the position value in the x or y axes, then (fig 2.11(i))

$$N(x, y) = NA(x, y) + ND(x, y) + NW(x, y) \qquad 5.1$$

where NA (x, y) = No of steps during acceleration or deceleration

ND (x, y) = No of steps up to the beginning of deceleration

NW (x, y) = No of steps of constant speed (dwell) to stop

If N (x, y) $\leq$ 32700, then the μC first loads the number ND (x, y) into T2 to be decremented at the pulse frequency (stepping rate). When T2 reaches zero, IRQ goes low and thus the μC is interrupted and immediately loads the remainder of N (x, y), ie NA (x, y) + NW (x, y) into T2 and commences programming T1 for deceleration as described in section 5.5(ii). On reaching the constant frequency (equal or less than the maximum starting frequency for the servo being driven) it dwells at this speed until the next T2 interrupt when the pulse train is cut off.

If N (x, y) $>$ 32700, then the number to be loaded into T2 is scaled down:

$$NS\ (x, y) = \frac{N\ (x, y) - NR\ (x, y)}{I(s)} \qquad 5.3$$

where NS (x, y) = No to be transfered to T2 during every load operation up to the beginning of deceleration.

$$NR\ (x, y) = NA\ (x, y) + NW\ (x, y) \qquad 5.4$$

= No of steps from deceleration to stop.

I(s) = Integer no of equal position data values tranfered to T2 during linear interpolation.

The value of NS (x, y) is such that

$$1 \leq NS\ (x, y) \leq 32700$$

After each T2 interrupt is generated, the μC decrements I(s) by one until I(s) reaches zero then, NR (x, y) is tranfered to T2 for the final position to be attained.

The μC timing requirements are based on the half period of each pulse cycle, ie. the minimum pulse width (at the faster axis speed). If this pulse width is T_{T1MIN} then

$$t_{op} \leqslant T_{T1MIN} \qquad 5.5a$$

where t_{op} = total time required for every interrupt response and data transfer operation ie.,

$$t_{op} = t_{IRQ} + t_{DSZ} + t_{LT2} \quad (\mu S) \qquad 5.5b$$

where

t_{IRQ} = time requird by μC to service IRQ

t_{DSZ} = time required to decrement a memory location (containing I(s)).

t_{LT2} = time required to transfer position data to timer, T2.

Although the SY6522 general purpose I/O LSI* is suitable for timing and linear interpolation it is appreciated that even with a powerful μC system circular interpolation would still involve some time-consuming recursive calculations. In the next chapter an LSI, KM3701, which is capable of generating linear, circular and other interpolations when controlled from a μC with normal bus architecture and supported by SY6522 is discussed (fig. 6.3).

*Fisher and Jensen (100) have described the inclusion of one SY6522 (VIA) to provide the timing and I/O latches for the IEEE488 bus data lines in the PET desk top microcomputer system (not built for machine tool control).

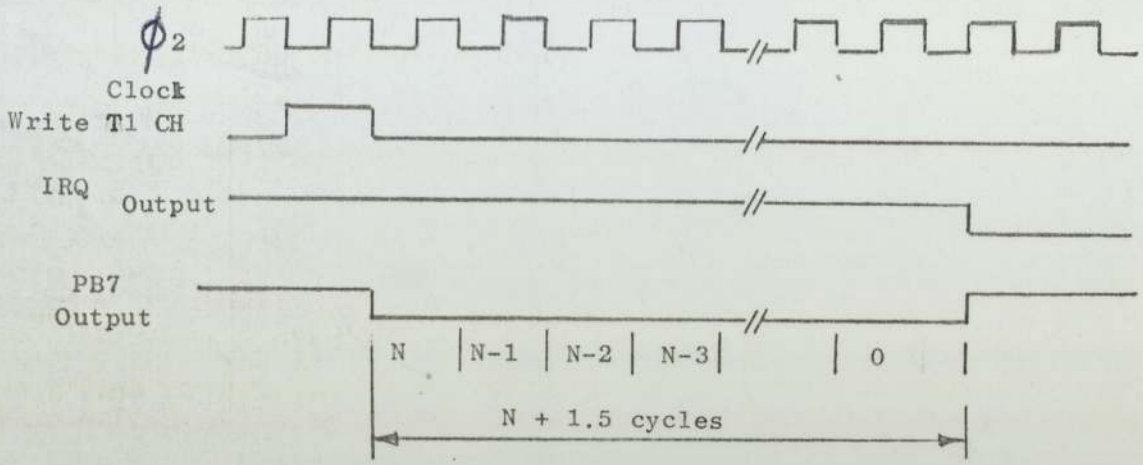


FIG 5.8a SY6522 Interval Timer 'one-shot' mode

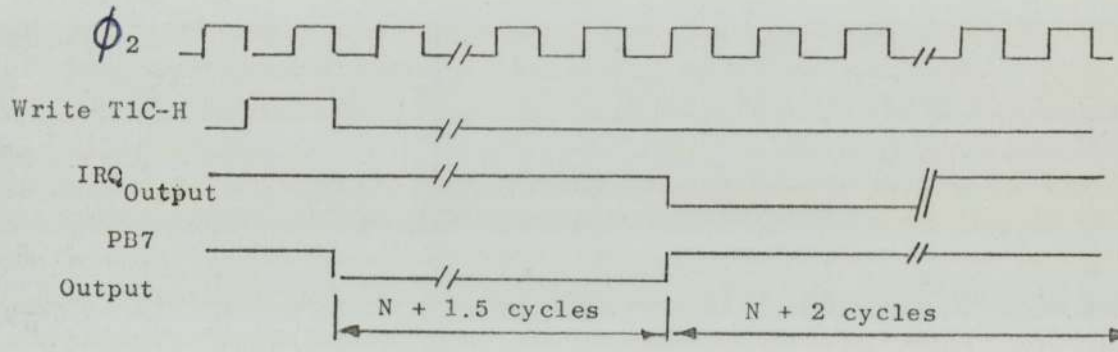


FIG 5.8b SY6522 Timer 1 - 'Free running mode'

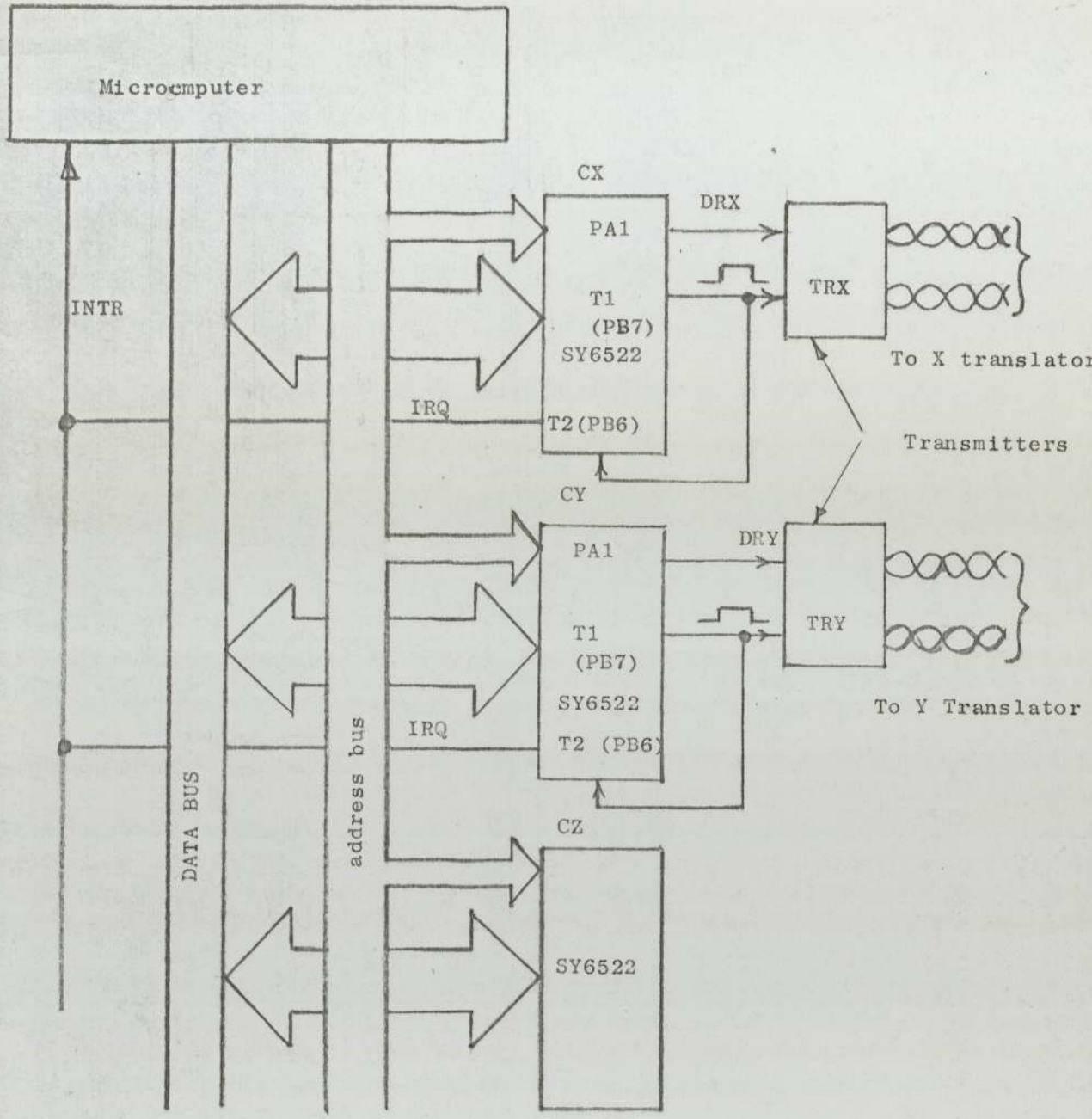


FIG 5.10 Microcomputer LSI Interface (SY6522) Block Diagram

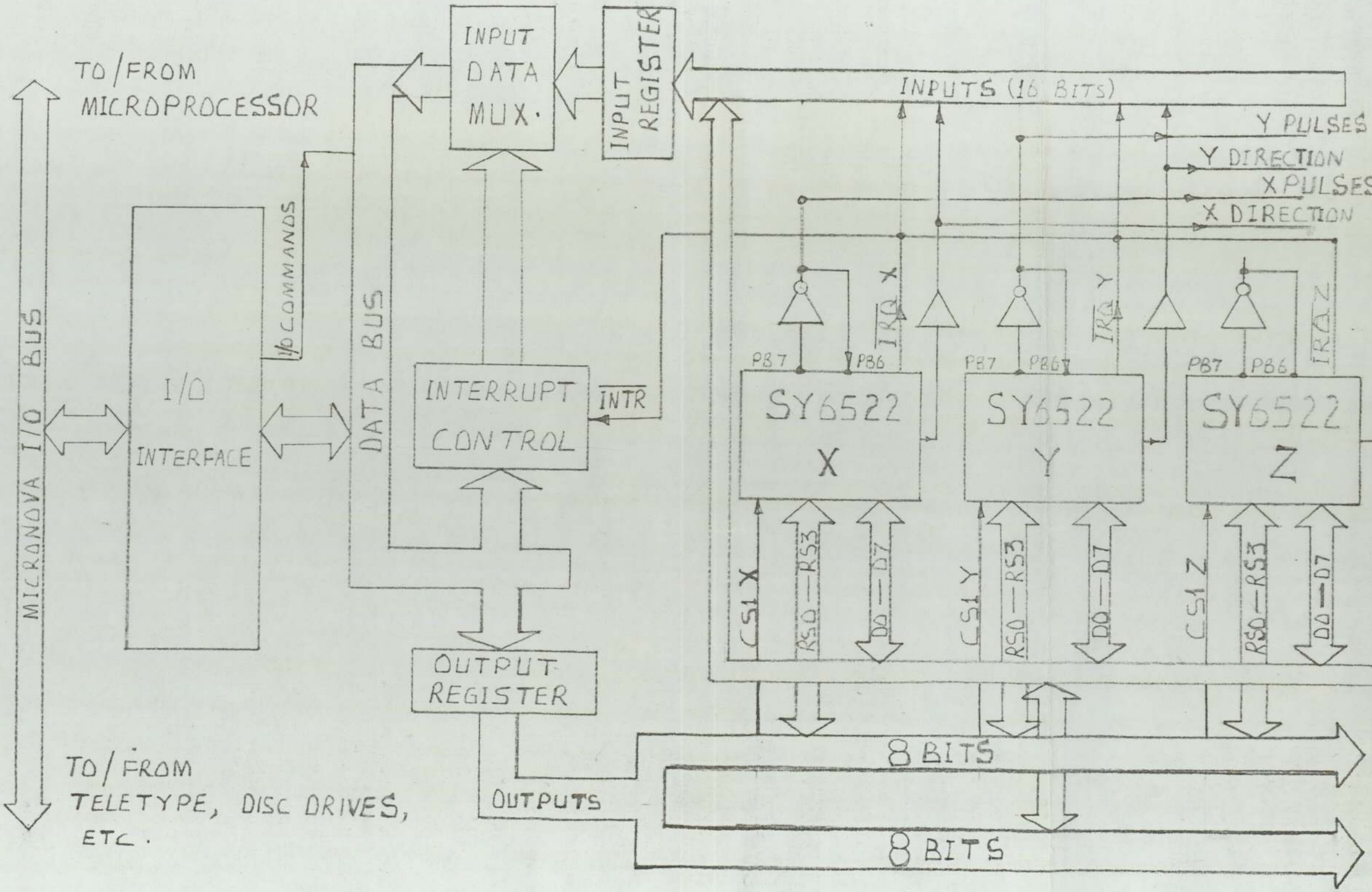


FIG. 5.12 SY6522 AT MICRONOVA I/O INTERFACE

CHAPTER 6

A MICROCOMPUTER PROGRAMMABLE INTERFACE WITH A MULTI - NC FUNCTION LSI (2)

6.1 Introduction

The advantage of the SY6522 unit in the microcomputer interface for interpolation and feedrate control of machine axes are its two powerful interval timers (T1 and T2) and its requirements of fewer supporting IC's for full implementation. Above all, it means economy of cost in the whole stepping motor control system. However, like other microcomputer-based coarse curvilinear interpolation systems it requires more real time computing work in the micropressor.

Since this study seeks to encourage conventional NC users to go micro-CNC it is necessary to introduce other LSI's which could be integrated with SY6522 to reduce these real time demands on the microcomputer and thus facilitate the inclusion of management calculations (if desired). A multi-processor system (a mini and μP's is a feasible solution for bigger firms but is too expensive for the smaller ones to acquire or implement (9). The alternative solution would be to achieve some of the benefits of MPS by way of a microcomputer and modules of multi-function LSI's. The two LSI's which are described in this chapter have the capability to generate circular, parabolic, exponential and logarithmic interpolation pulses under instructions from a microcomputer. They require less programming and on-line computing effort and so less memory may be needed to implement a CNC MDI system. These devices are KM3701 and KM3702, which were very

recently announced by Toko Inc. (W. Germany). One KM3701 generates two-axis pulses to two KM3702 LSI's to drive two dc motors. This chapter will concentrate on the principal module which is KM3701 and how it can be used to drive stepping motors.

6.2 KM3701 LSI for Numerical Control (Figs 6.1, 6.2, 6.3)

This was not built specifically for stepping motor control but like SY6522, it is another multi-Register Input/Output LSI which is capable of generating constructed interpolation pulses for two axes following instructions from the microcomputer CPU. The control file could also be generated on a diskette (DOS) as an MDI system as described for the SY6522. The microprocessor provides all the data relating to the start, profile and end of the interpolation. Unlike the DDA and some algebraic calculation circular interpolations, the KM3701 provides linear, circular, parabolic, exponential and logarithmic interpolations selectable by the microprocessor via the MDI control file. These profiles can be generated without further help from the μP because KM3701 contains enough circuitry to perform internal calculations and distribute the resulting FP pulses to both DX and DY pins (fig. 6.2) according to the profile commanded by the microcomputer. It is this reduced demand on the μC, its memory and the programmer that is the virtue of KM3701 LSI. It ignores the FP pulses during the internal calculations and automatically gates them through afterwards to activate the DX and DY outputs. On completing the pulse distribution the INT line goes low indicating the end of interpolation and thus interrupts the μP which then can provide data for the next job. It is possible for the μP to read and display the contents of a chosen register at all times except during internal calculations which KM3701 indicates with the READY flag set low.
The axis position data registers are 24 bits wide and are organised as three 8-bit bytes, the lower, middle and most significant bytes. The maximum pulse rate for linear interpolation is 100KHZ and 50KHZ for others. The system uses a 1MHZ clock (and 10 clock pulses are required internally to construct a linear interpolation pulse and

20 clock pulses for others). Fig. 6.2 is a block diagram of Toko's 3701 function organisation.

6.2.1 Pin Assignments and Functions

Signal Abreviation	Pin No	I/O	Description	
Vcc	28	-	+ 5V	
Vss	14	-	Ground	
CS	1	I	Device Select	μP Interface
R/W	2	I	Read Write Line	
OD	3	I	Output Disable	
AO-A2	6-4	I	Address of Registers	
DO-D7	15, 13-17	I/O	Data Bus (open drain)	
CP	18	I	Clock Pulses	
BUSY	22	O	Busy Line	
RESET	17	I	Reset	
FP	19	I	Feed Pulses	
DX	24, 25	O	Interpolation Pulse	
DY	26, 27	O	Outputs	
INT	23	O	Interrupt Line	

6.3 Feedrate Control (Figs 6.4, 6.5)

The designer of the microcomputer controlled NC system has a choice of two proceedures. Both have to involve the FP pin for the input motion pulses. One approach constructs the feedrate pulses before input to FP and the other after outputs DX, DY. In the first method the µC generates the major axis feedrate pulses (the absolute feedrate being selected via MDI) to the FP pin. In fact KM3701 operates in this way, basing the interpolation on the pulses fed to the major axis. One of the ways by which the µC can be made to do this is to program a real time clock (if available). The output of which form the FP pulses or a trigger for pulse construction. In that case the µC can more easily monitor the stepping progress for acceleration, decelaration and position. However, it will be virtually tied down if it has to service every pulse sent to the interface throughout the machining operation. This problem can be solved with a BRM and counter arrangement to construct the FP pulses. A better solution, (discussed in chapter 5) is to include one SY6522 to generate only the major axis feedrate pulses leaving its interpolations to KM3701, (fig. 6.3)

The second approach involves the use of two programmable timers, one each for the DX and DY outputs respectively. These are programmed (like counters) during acceleration and deceleration and left constant for the normal feedrate. The snag here is that since KM3701 has already done the interpolation and counted the pulses, one would never attain the desired position unless the initial position data from the µC is magnified to take account of the division by the counters. It will be noted also that at constant feedrate both timers must contain the same number so as not to alter the interpolation pattern.

The maximum pulse rate (whatever method is employed) will be obtained when SY6522 or each of the two timers is loaded with one.

For a 0.9 half step angle motor and a machine resolution of 0.0001 (as in HPESYS), the maximum traverse rate will be:

$$F_{max} = (.0001)(60)(f_o) \text{ in/min} \qquad 6.1$$

Where $f_o = FP_{max}$ = 100KPPS = max. pulse rate in the major axis for linear motion, (the author is not aware of any stepping motor available to achieve this macro-stepping rate at this time).

However, from 6.1, F_{max} = 600 in/min

In the second method, the minimum feedrate will be obtained when the n - bits of the timers are fully set.

A 16-bit interval timer will output a feedrate pulse after $2^{16} - 1$ FP pulses. The minimum feedrate will then be (linear interpolation)

$$F_{min} = \frac{F_{max}}{(2^{16} - 1)} = \frac{600}{65355} = .0092 \text{ in/min}$$

For other interpolations, $F_{min} = \frac{50KPPS}{100KPPS} \times .0092 = .0046$ (in/min)

For best performance in the μC control it is recommended that the CP clock be derived from the microprocessor master clock in which case another pulse rate divider has to be interposed between the master clock and the CP clock input to limit the CP rate to 1 MHZ.

6.4 Functional Descriptions of KM3701

This section gives the summary of the functional organisation of the various address, data, control and interpolation status registers.

6.4.1 The microprocessor interface

Section 6.2.1 shows which pins are directly interfaced with the μC bus. Details of the operation of the registers controlled by these pins now follow.

6.4.2 Internal Register Address Assignments

	A2 A1 A0	Read Mode (CS=0, R/W=1, OD=0)	Write Mode (CS=0, R/W=0, OD=1)
Data Register	0 0 0	Least significant byte	Least significant Byte
	0 0 1	Mid significant byte	Mid significant byte
	0 1 0	Most significant byte	Most significant byte
	0 1 1		Register Control word
	1 0 0	Control word	Control Word
	1 0 1	Status Word	

6.4.2.1 Data Register

This 24-bit data register is a KM3701 'Receptionist' because it simply acts as a temporary buffer between the μC data bus and the LSI's internal interpolation registers. The next command transfers this data to its appropriate Register. The data inputs which are axis coordinates can be resolved into 24 binary bits each by the 16-bit μP instructed to operate in double prevision. Obviously there is some memory penalty

to be paid but if the axes positioning spans or the machine resolution do not justify it, single precision (16-bit words) are acceptable, (this leaves only the most significant byte empty).

Register Part	D7	D6	D5	D4	D3	D2	D1	D0
Least significant	B7	B6	B5	B4	B3	B2	B1	B0
Mid significant	B15	B14	B13	B12	B11	B10	B9	B8
Most significant	B23	B22	B21	B20	B19	B18	B17	B16

The binary number is accompanied by a sign bit (2's complement) which will later enable the KM3701 to decide which of $\pm$(DX, DY) to select as output. Alternatively the program could be structured along the lines of the DIRECTION SUBROUTINE of the HPESYS in chapter 9. In that case only positive values of the co-ordinates are transferred to the registers, the direction together with other machine control functions being controlled directly by the µC via its digital I/O parts or peripheral adapters (PIA's) if the digital I/O option is not part of the µP system.

6.4.2.2 Register Control Word

This operation mode is selected as shown under 6.4.2. Ideally KM3701 is dedicated to only two axes but it is possible to operate in $2\frac{1}{2}$ axes. (This can be done by using the design in Figure 6.3.)

X, Y, U, V symbols are defined by the following interpolation equations (figs 6.6, 6.7, 6.8, 6.9 and 6.10 respectively)

$$Y = \frac{V_o}{U_o} X \text{ (linear interpolation)} \qquad 6.6$$

$$(X - U_o)^2 + (Y - V_o)^2 = U_o^2 + V_o \text{ (Circular)} \qquad 6.7$$

$$2 V_o Y = - (X - U_o)^2 + U_o^2 \text{ (Parabolic)}$$

Axis of symmetry parallel to Y - axis 6.8

$$Y = V_o (1 - e^{-x/U_o}) \text{ (Exponential)} \qquad 6.9$$

$$Y = -V_o \ln (1 - \frac{x}{U_o}) \text{ (logarithmic)} \qquad 6.10$$

Where X and Y are the end point co-ordinates and U and V are the co-ordinates at the centre of the path (for a circle this is the centre and for a line the midpoint co-ordinates).

The control S/W has to be structured to calculate those co-ordinates. The part programmer may base his co-ordinates on a fixed or shifting origin, the latter requiring less μP calculation and memory. As with SY6522, four bits inside KM3701 are decoded and used to address the various co-ordinate registers. This is done by setting D4 → D7 low and assigning D3 → D0 to the internal register bits RS3 → RS0 respectively. Then RS3 → RS0 are decoded appropriately to address X, Y, U, V.

D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	0	RS3	RS2	RS1	RS0

The co-ordinate is first latched into the 'reception' from the microcomputer bus, then the proper register (X, Y, U, or V) is addressed in the next instruction. This effects the transfers below.

RS3	RS2	RS1	RS0	Address of Interpolation Register
0	0	0	0	No Operation
1	0	0	0	No Operation
1	0	0	1	Transfer 'Reception' to X Register
1	0	1	0	Transfer 'Reception' to Y Register
1	0	1	1	Transfer 'Reception' to U Register
1	1	0	0	Transfer 'Reception' to V Register
1	1	1	1	Initialization of interpolation

'Reception' is the 24-bit data register content as mentioned earlier. The 24-bits are simultaneously transferred in every case. For μC display purposes, any of the transferred data can be recalled to 'Reception' and be read back by the μP. The bit assignments for this are as shown below.

RS3	RS2	RS1	RS0	Command
0	0	0	1	Recall X to 'Reception'
0	0	1	0	Recall Y to 'Reception'
0	0	1	1	Recall U to 'Reception'
0	1	0	0	Recall V to 'Reception'

After the data transfer from 'Reception', the Register Control Word, RS3 → RS0, is cleared automatically to the no operation state. (without further intervention from the μP).

6.4.2.3 Control Word

This is also first addressed via $A_0 \rightarrow A_3$ (section 6.4.2). The various interpolation contours are selected through the Control Word mode.

D7	D6	D5	D4	D3	D2	D1	D0
RUN	PR	EY	EX	DR	DW	DV	DU

where DU, DV DW bits are used to select contouring functions (including linear) for interpolation and 'DR' is used to select rotating direction for curves ie. clockwise or anticlockwise. When 'PR' bit is set high the FP pulse distribution rate is reduced by 1.4th of its original value (if necessary). The 'RUN' bit when high enables the selected interpolation to commence. The EX, EY bits are used to program the selected quadrant end points in circular interpolation as shown below (can be applied to the HPESYS which is selected by MDI).

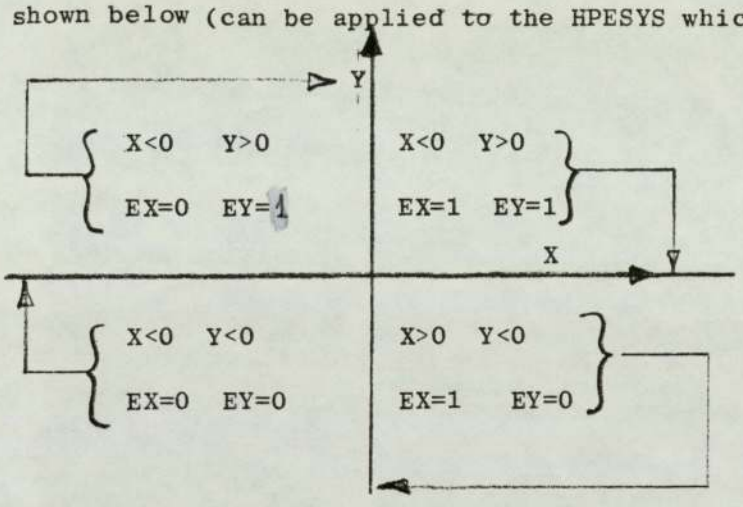


Illustration with circular interpolation

The combinations of DW, DV, DU for selecting the interpolation contours are shown below:

D2	D1	D0	
DW	DV	DU	Interpolation Functions
0	0	0	Linear
0	0	1	Logarithmic
0	1	0	Exponential
0	1	1	
1	0	0	
1	0	1	Parabolic (symmetrical about Y axis)
1	1	0	Parabolic (symmetrical about X axis)
1	1	1	Circular

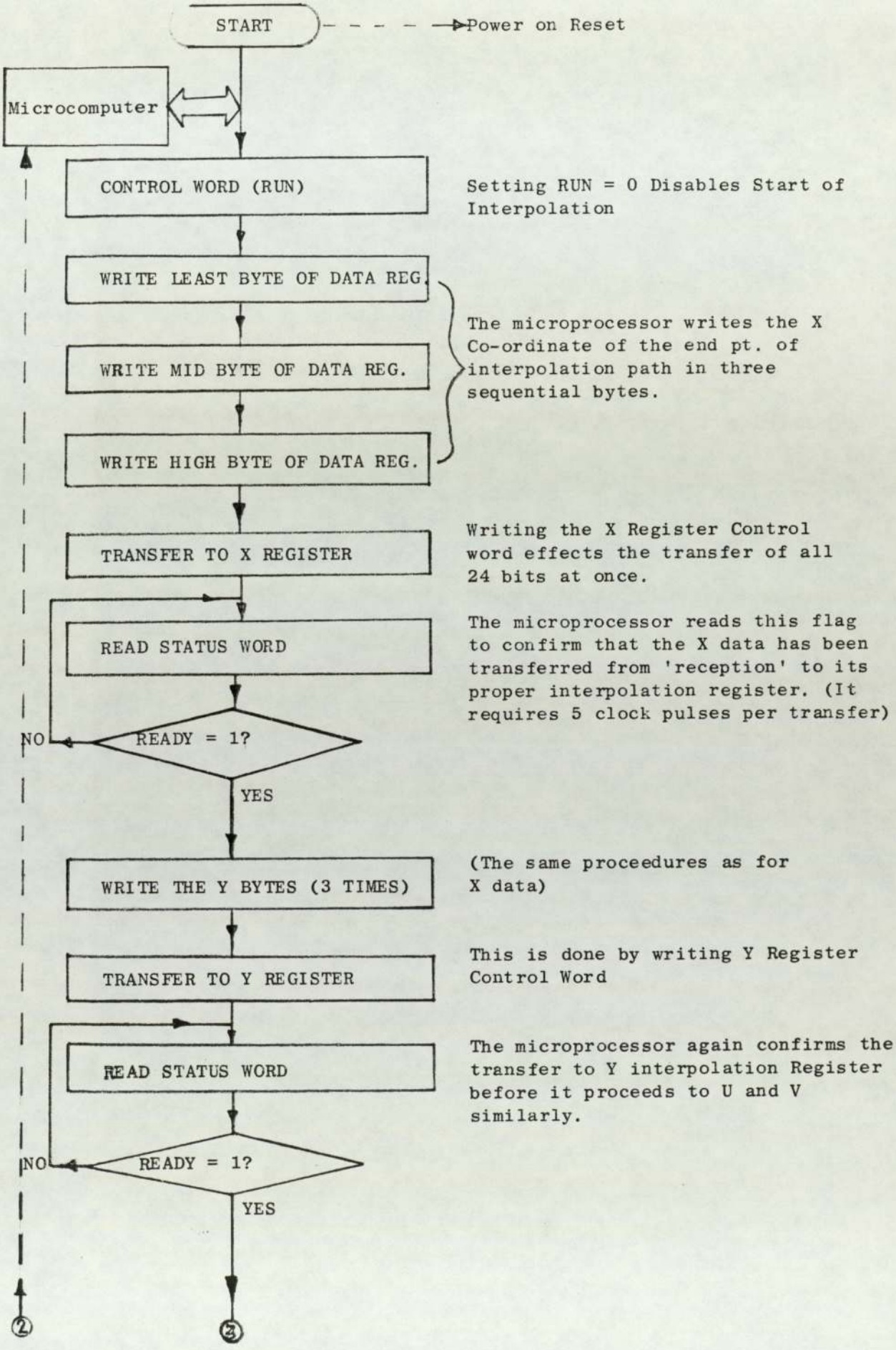
6.4.2.4 Status Word Control

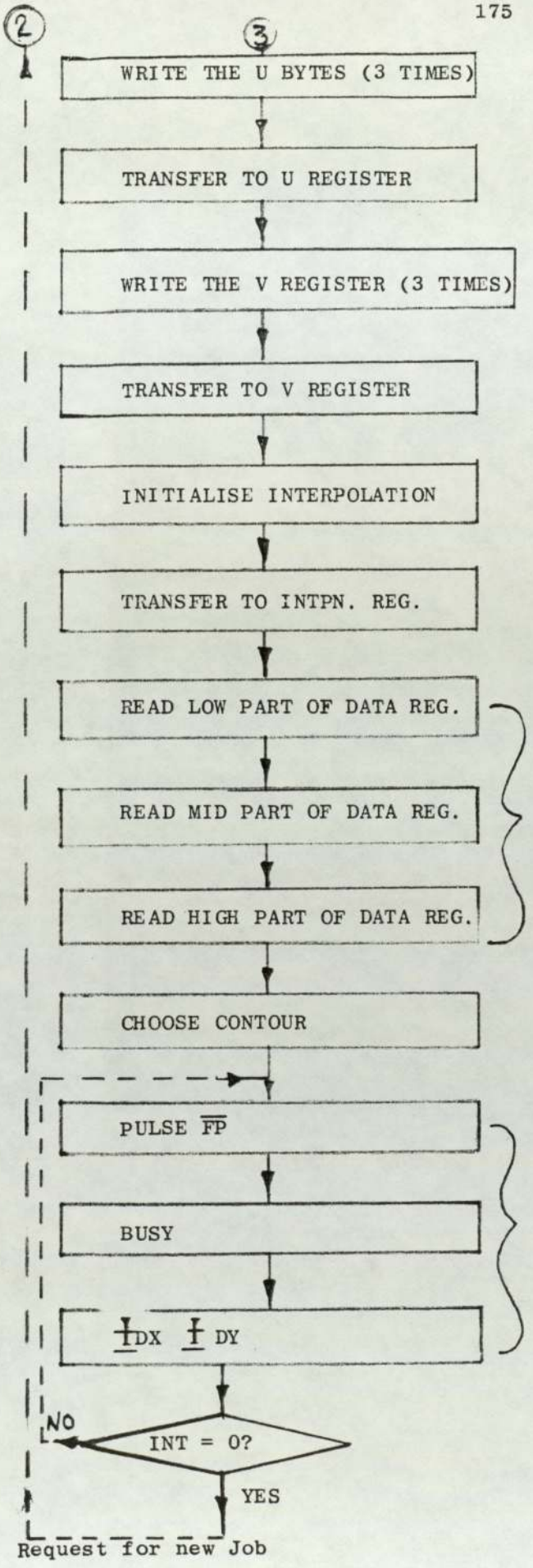
D7	D6	D5	D4	D3	D2	D1	D0
READY	FEND	ZONE SET	M	SV	SU	ZYR	ZXR

The status words are used to flag the state of the registers for the μP Read and display

READY	Flag to show that the data register is ready for Read/Write
FEND	Flag to show end of interpolation
ZONE SET	Flag to indicate coincidence of the sign bits of U, V with the sign bits of end pt. quadrant parameter, EX, EY (can be ignored).
SV	Sign flag for V Register
SU	Sign flag for U Register
ZYR	Zero flag for Y Register
ZXR	Zero flag for X Register

6.4.7 Summary of Program flowchart





(As in X & Y)

By writing the U REGISTER CONTROL Word (Confirmation is made afterwards as usual)

Write V REG. CONTROL WORD; (Confirm transfer)

Write the appropriate Control Word.

Write the Control Word for Interpolation register. Then confirm transfer as before.

By writing '01', '02', '03', '04', the µP re-transfers X, Y, U, V respectively to the 'reception' to cross check their validity. (X, Y, U, V remain unchanged in their respective interpolation registers).

Write the appropriate control word. and start interpolation with RUN = 1.

The microcomputer does not intervene in this loop. The KM3701 accepts the FP pulses and distributes them according to the selected contour until INT = 0

6.5 A Choice of Feedrate Control from Acceleration and Deceleration Considerations

KM3701 has no facility for accelerations and decelerations From the control program flowchart just described (6.4.7) it is evident that after the input of pulses to the FP pin the microcomputer has no further control of the feedrate and the pulses reaching the motors until the KM3701 generates the end of interpolation interrupt. Therefore in order to simplify the design the μC will not control the acceleration and deceleration of the motors after the pulses leave the DX and DY outputs these being dependent on the rate at which the FP input is pulsed. Therefore the second approach to feedrate control discussed in section 6.3 is ruled out, leaving us with the first approach. The microcomputer will then programm the interval timer preceeding the FP pin (fig. 6.3) such as to configure the desired acceleration of the whole machine. Also considerations for the deceleration lead to another complication which is discussed in the next section.

6.5.1 End of Acceleration and beginning of Deceleration

Since the microcomputer controls the feedrate it has the necesary parameters to detect the end of acceleration and beginning of deceleration. The end of acceleration is deduced in the same way which had been described for SY6522. In the present aspect, however, the microcomputer calculates only for the major axis acceleration instead of for individual stepping motors. Therefore the ratio of the axes positions has to be used to establish the proper acceleration/deceleration rate such that non of the motors is over demanded in this respect. As for SY6522, when the acceleration number becomes equal or less than the constant speed number

(to be loaded into the internal timer) the latter is loaded instead and left constant in the latch.

To detect the beginning of deceleration two alternatives exist. Whichever choice will be made depends on the calculated value of the total number of steps (in X or Y motor) which will mark the beginning of deceleration (this has been fully discussed for SY6522). One choice is for the μC to count up to this number and then to commence loading the former acceleration values in a reverse order. For the outdated micro NOVA which the author had used to program SY6522 (for the purpose being discussed, one would run into timing problems at faster speeds by this technique. For the new microNOVA (announced last year (11)) and the other more current microcomputers this choice is feasible provided the counting is given the highest priority (except for malfunction interrupts) to other control proceedures. A top speed of 20 KHZ means a period of 50μSec within which the microprocessor should receive the interrupt, increment the memory location, update the count, do shifting operations, perform the calculation (unless a look-up table is availabe) for the first deceleration number and to output it.

If this is not practicable, then the second choice will be an external H/W by way of a counter which should then generate the interrupt to notify the microprocessor to start deceleration. This counter could be the timer T2 of SY6522 if its timer T1 is already being used for feedrate control.

6.6 Operation of KM3701

6.6.1 DC characteristics

VCC = + 5.0V 5%

Parameter	Symbol	Min	Max	Unit	Test Condition
Low Level input voltage	VIL	-0.3	0.6	V	
High Level input voltage	VIH	3.0	Vcc + 0.5	V	
Low Level output voltage	VOL		0.4	V	IOL = 2 mA
High Level output voltage	VOH	2.4		V	IOH = 200 µA
Low Level output voltage	VOL	0.4		V	IOL = 5.5 mA (see note 2)
Output current	IOFF		I10	µA	Vout = 0 Vcc
Input leak current	IIL		I10	µA	VIN = 0 Vcc
Power supply current	ICC		50	mA	

Note 1: Clock signal BUSY, INT, DX, DY

Note 2: D7 - D0: Open drain output

6.7 Examples of Part Program Geometries

Linear Interpolation: $(Y = \frac{Vo}{Uo} X)$

End Point (X, Y) = P (6,4)

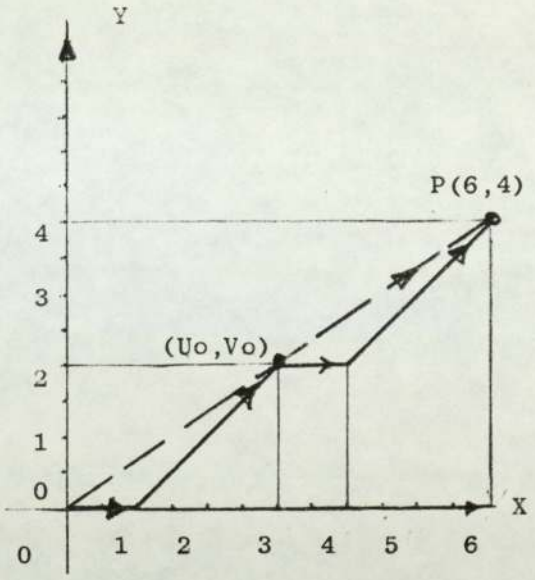
Y data = 000006 (decimal Units)

U data = 000003 (decimal Units)

Y data = 000004 (decimal units)

V data = 000002 (decimal Units)

(Starting point is always (0,0))•



Interpolation Initialisation

Interpolation Initialisation Control Word = '80' (hexadecimal)

$\overline{FP}$ pulse no	1	2	3	4	5	6	INT
Axis sent to	+DX	+DX,+DY	+DX,+DY	+DX	+DX,DY	+DX,DY	EXIT

Cicular Interpolation: $(X-Uo)^2 + (Y-Vo)^2 = Uo^2 + Vo^2$

End point (X, Y)= R (5,5); here radius = 5.

The centre of the profile always (Uo, Vo).

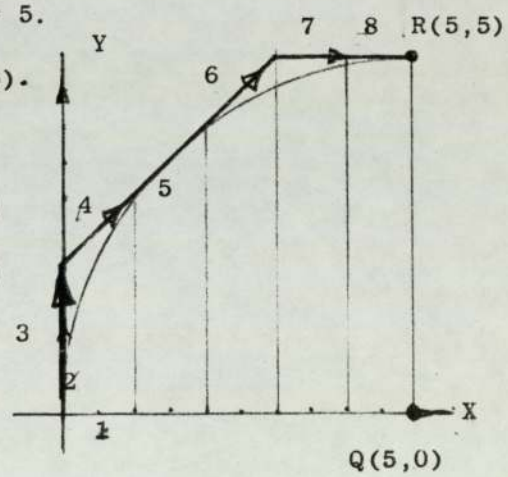
Here centre is Q (5, 0)•

X data = 000005 Units (hexadecimal)

Y data = 000005 Units (hexadecimal)

U data = 000005 Units } Centre

V data = 000000 Units } Centre



Interpolation Initialisation Control word = 'B7' (dexadecimal)

(SU = 1, SV = 1⟶ CW rotation)

Pulse Distribution

$\overline{FP}$ No.	1	2	3	4	5	6	7	8	9
Axis	*	+DY	+DY	+DX,+DY	+DX,+DY	+DX,+DY	+DX	+DX	*INT

* Indicates that no pulse is output yet (as part of KM3701 interpolation strategy).

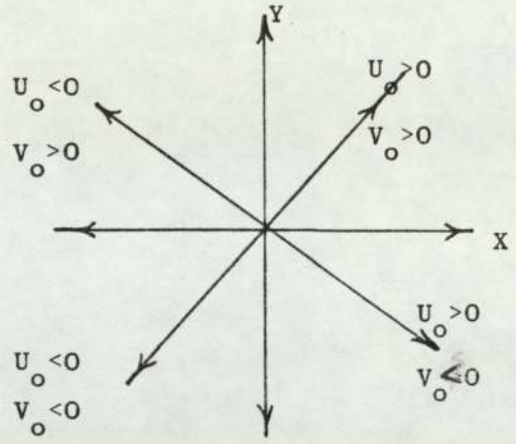
6.8 The KM3702 for Control of DC Motors

This is a 40-pin LSI which is complementary to KM3701 in the drive of DC motors, (one KM3702 for each motor) fig. 6.4)). Under instructions from the microprocessor it monitors the difference between two kinds of input pulses (one being the interpolated pulses, the other the clock pulses). Output pulses proportional to this difference are then generated and input to a D/A converter the resolution of which is 16-bits. The output of the D/A converter is amplified and used to drive the DC motor. Like the microcomputer-controlled DC motor system described by Tal and Persson(1) the feed back pulses are from shaft encoders coupled to the DC motors. Tal and Persson's system send feedback pulses to the μC which does the counting with the attendant timing problems at pulse rates over 15KHZ. This sytem has no such problems because the counting is done by two 24-bit synchronous up/down counters inside KM3702. One is a command counter and the other an error counter the contents of which can be read and displayed by the microcomputer. The programming of KM3702 is similar in many respects to that of KM3701, already described, and so will not be discussed further here.

6.9 Conclusions

The limitation of this system is the fixed range of profiles for which it can be used but it can also be observed that these are the more common component contours. In fact most CNC machine tools have only linear and circular interpolations.

In the introduction to this study mention was made of Colin Page (9) who stated that programmable LSI's will continue to offer significant competition with μP's in dedicated systems. KM3701/3702 constitute one of the more recent examples. Smaller firms (especially the sub-contract types) can use these LSI's for fast in-house design of their microcomputer controlled NC machine system. It leaves abundant real time on the μC to incorporate some management calculations depending on how much memory is available. With the rapidly increasing memory integration and speed of acces, on-line management functions like production control, machining times and tool life calculations can be done during metal cutting using a μC. Although the control could only be in $2\frac{1}{2}$ axes, the cost effectiveness espcially with well engineered open loop stepping motor controls and the reduced expertise to implement the system could encourage firms with less than £100,000 turnover to join the micro-CNC trend, a step to more complex CNC.

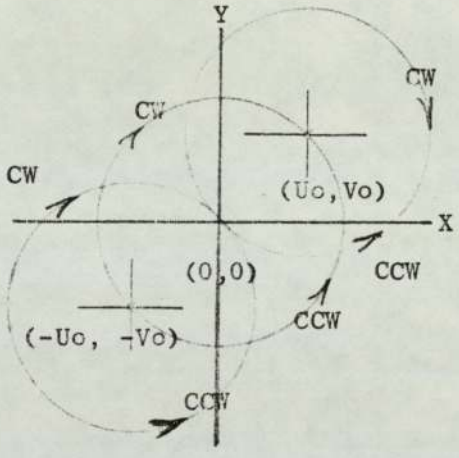


$$Y = \frac{Vo}{Uo} X$$

(Uo,Vo) = midpoint of line .

Directions are defined by Uo, Vo combinations .

Fig. 6.6 Linear Interpolation

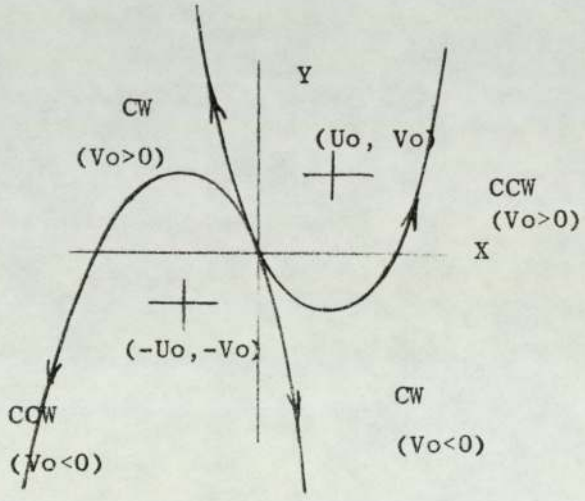


$$(X - Uo)^2+(Y-Vo)^2=Uo^2+Vo^2$$

1st Quadrant: Ex=1, EY=1
2nd Quadrant: Ex=0,EY=1
3rd Quadrant: Ex=0, EY=0
4th Quadrant: Ex=1, EY=0
Start = (0,0) Always .

(With end point (X,Y), EX and EY determine the arc) .

Fig. 6.7 Circular Interpolation



$$2Vo = -(X-Uo)^2 + Uo^2$$

* Axis parallel to Y

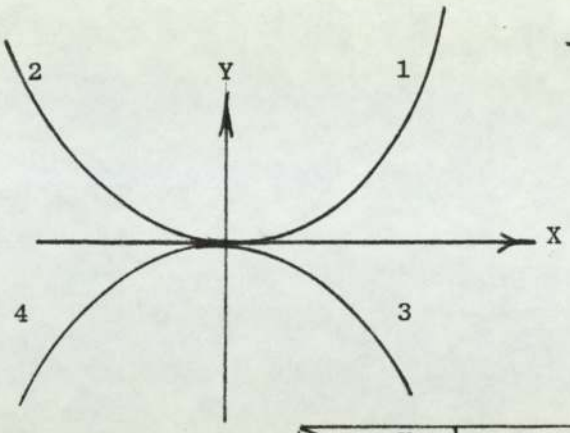
Convex Y upwards: EY = 0

Convex Y downwards: = 1

Fig. 6.8 Parabolic Interpolation

* Axis parallel to X: Convex X leftward: Ex = 0
Convex X rightward: Ex = 1

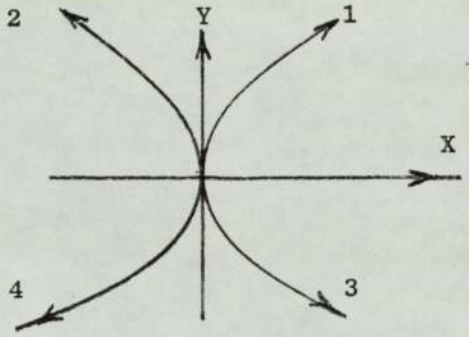
$$Y = Vo(1-e^{-x/Uo})$$



Vo \ Uo	<0	>0
>0	Curve 1	Curve 2
<0	Curve 3	Curve 4

Fig. 6.9 Exponential Interpolation

$$Y = -Voln(1-\frac{X}{Uo})$$



Vo \ Uo	<0	>0
<0	Curve 1	Curve 2
>0	Curve 3	Curve 4

Fig. 6.10 Logarithmic Interpolation

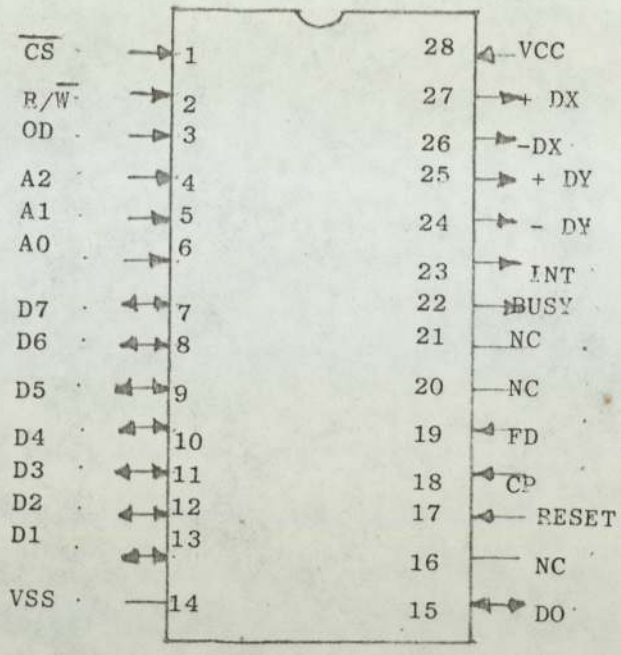


FIG 6.1 KM3701 LSI for NC

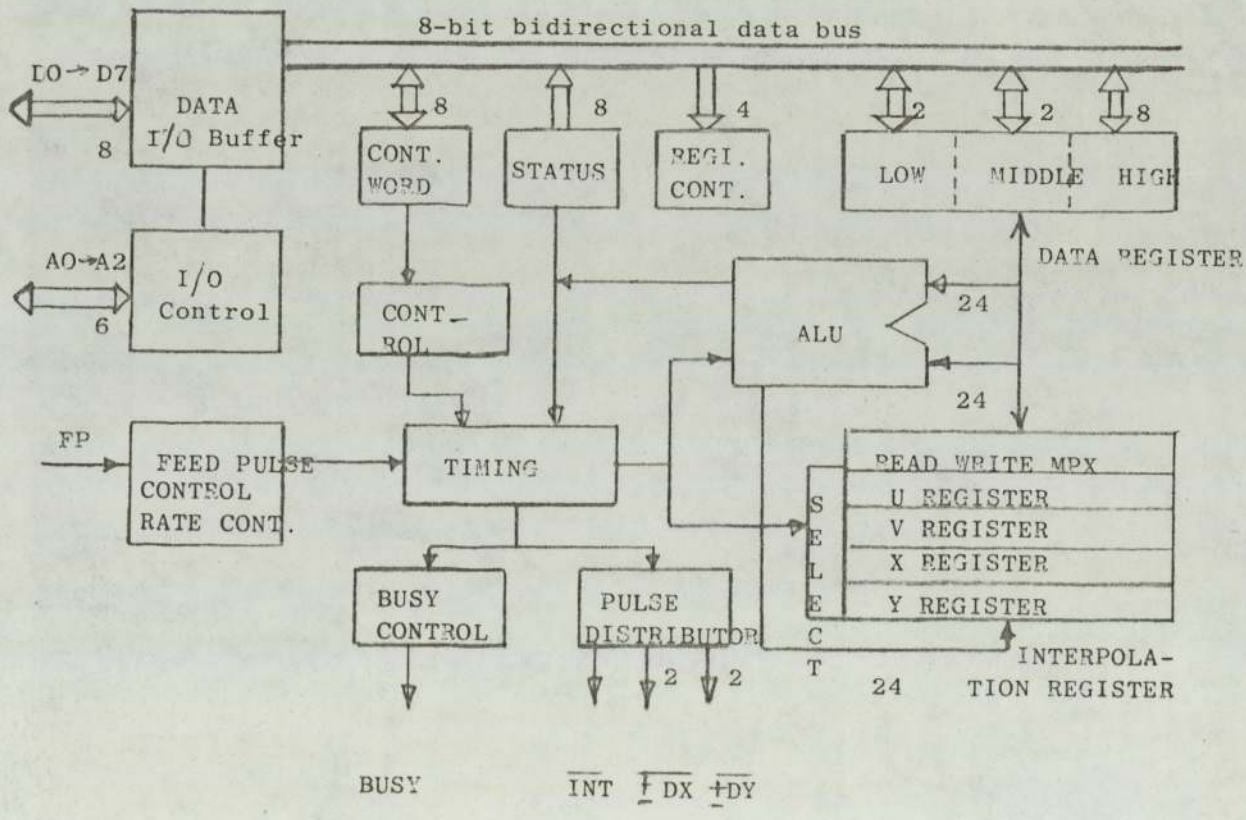


FIG 6.2 KM3701 LSI Block Diagram

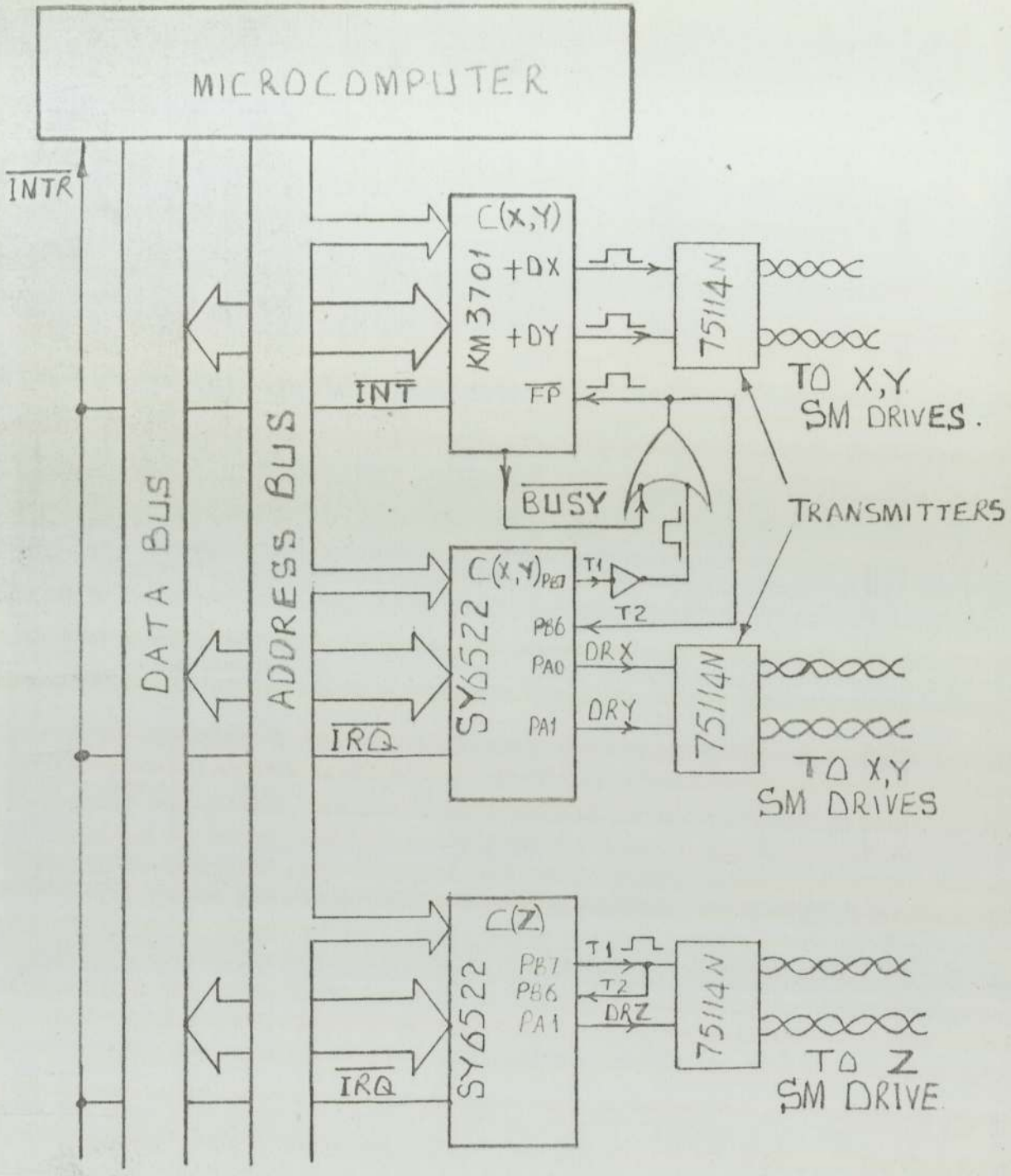


FIG. 6.3 MICROMPUTER PROGRAMMABLE NC INTERPOLATION LSI INTERFACE

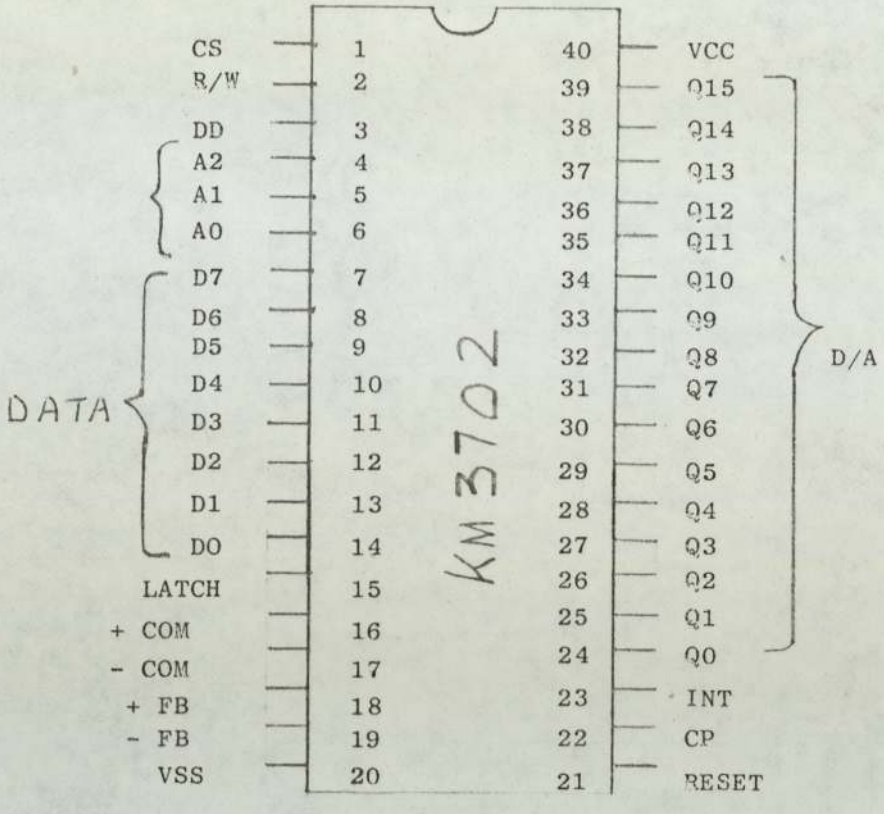


FIG 6.4(i) KM3702 LSI for DC Motor

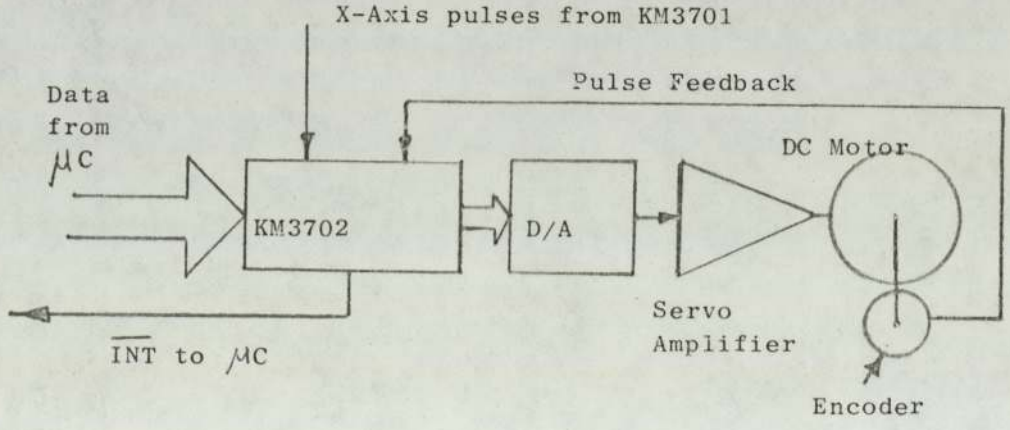


FIG 6.4(ii) KM3702 for Control of DC Motors

CHAPTER 7

CY500: STORED PROGRAM STEPPING MOTOR CONTROLLER (3,4)

7.1 Introduction

The last chapter dealt with two LSI's which could be used with SY6522 to develop a multifunction NC system controlled by a single microcomputer. This system (with Koto's LSI's) is limited to $2\frac{1}{2}$ axes control. It can be designed to control either stepping or DC motors but the programming procedures have to be repeated for every contour in a part program. This chapter will describe the CY500 a controller designed specifically for stepping motors and for which total program re-initialisation may not be necessary bacuase it has a program buffer and this can store high level language-type commands. This controller, recently announced by Cybernetics Micro Systems (4), is a 40-pin device (fig. 7.1) using a standard 5 volt power supply and is microcomputer-programmable, with asynchronous parallel TTL I/O lines. It will easily interface with any standard µC data bus. There are 22 high-level instructions which are teletype (ASCII/Decimal) or ROM programmable in a stand-alone mode (fig. 7.2), or microprocessor-programmable in a µC system (fig. 7.4). This description will concentrate on the microcomputer control aspects.

7.2 Standard Features (fig. 7.1)

These are the other design features of CY500.

Binary communication

Half step/full step operation

Absolute/relative positioning modes

Programmable output line

Two interrupt request lines

Hardware/software direction control

Hardware/software start/stop

Abort capability

single/multi step instructions

Ramp-Up/Slow/Ramp Down modes

Triggered operation

'DO-WHILE' command

'WAIT UNTIL' command

Several synchronous I/O's

7.2.1 Operation

WR pin 1 is the write strobe for the μC (figs. 7.1 & 7.4). The data lines (DB7-DB0) are μC bus programmable in parallel mode to drive a four-phase (ϕ_1 - ϕ_4) stepping motor every step of which starts simultaneously with a pulse output (5 μsec width) at pin 35. The interrupt request 1 (INT REQ 1), (pin 37) indicates the completion of motion and is used to interrupt the host μC at the end of the stepping. The TOGGLE (pin 36) changes level with each step executed and is used as a timing signal. The TRIGGER pin 30 is used for single step (jog) operation under computer control. When it is low the 'GO' command starts a multi-step operation. If the μC sets it high before the 'GO' command, the controller waits

until the pin is low again when only one step is taken. A software 'JOG' command is also provided. The CONTROL pin 34 is a user programmable pin. Issuing the 'BITSET' command (sect. 7.2.2) sets the control line (pin 34) and the 'CLEARBIT' command will reset this output pin. The designer can include these commands in the program sequences for features like ON/OFF coolant when a μP is being used.

For smooth data transfers from the asynchronous μC data bus, pin 37 (BUSY/READY) provides a handshaking line. With this feature the μC will wait until the READY state is signalled before commands or data inputs are made to the CY500. If the standard bus interface IEE 488 (ref.100) is used, its handshaking lines will be connected to this pin (CY500 acknowledges DATA TAKEN by changing READY to BUSY). The ASCII/BINARY pin 33 can be manually controlled to select ASCII-decimal information from tele-type or binary for that from a μC controlled system. The mode select input pin is used to set this. (BASIC programmers who can input commands to CY500 in ASCII-decimal willfind this feature especially useful). For program entry and run controls internally, pins 31 and 32 respectively, are used. PROG (pin 31), when low indicates that programs are in the entry mode and no execution can occur. RUN (pin 32) which also serves as INT REQ 2 is automatically set low during program execution and prevents further entry of commands. The microcomputer can monitor pin 31 and be interrupted via pin 32 to indicate completion of program execution and readiness for a new command entry. For the external start/stop of stepping motors or pulse generation, pin 28 (XSS/TILL) is used. It can also be used to exit CY500 from a program loop. For some emergencies, the μC can be programmed to do this or an operator can decide to do so for other reasons. For direction control by software, the + and - signs are used for clockwise and anticlockwise rotations respectively. If for any reasons the operator decides on

external direction control, he enters an 'L' command (section 7.2.2) at the appropriate point in the program sequence so that the computer may display a request for direction setting.

The default mode is the software direction control when CY500 establishes internally the direction of switching of the motor phases, $\phi_1 - \phi_4$. In a similar manner external jog control is possible if the programmer places the command 'J' in place of the GO command, 'G' (section 7.2.2). The Start/Stop pin (28) will then be used to start and stop the stepping sequence by the operator of the NC machine.

Pins 2 and 3 (XTAL1, 2) are inputs for the external Crystal clock which is 6 MHZ standard. Other crystal clock frequencies can be used but they will require a modification of the rate parameter, r, issued in 'R' command to fix the stepping rate using a non-linear rate table.

It is possible for the μC to read back the parallel data on the bus during the input mode by strobing the RD line (Pin 8). (The data on the bus must remain valid until the trailing edge of the RD strobe occurs).

Another stop option is provided at pin 6, (ABORT). If the ABORT pin is set low during a stepping sequence, that particular sequence is aborted and the 'motion complete' line (INT REQ 1) goes low so that if the program is still in execution the next instruction is fetched from the buffer store. Finally, the RESET line is pin 4 which acts as a power-up reset of CY500.

7.2.2 CY500 Command Summary (3)

The capacity of CY500 program buffer is 18 byte - combinations of any of the following commands:-

Table 7.0

ASCII	BYTES	NAME	INTERPRETATION
A	1	ATHHOME	Set current location equal to absolute zero
B	1	BITSET	Set the programmable (control) output line high
C	1	CLEARBIT	Set the programmable (control) ouput line low
D	1	DOITNOW	Begin Program execution
E	1	ENTER	Program entry mode
F, f	2	FACTOR	Declare rate divisor factor (1-255)
G	1	GOSTEP	Start stepping operation
H	1	HALFSTEP	Set operation for halfstep mode
I	1	INITIALISE	Clear ϕ_1 - ϕ_4 lines, ready for command (no change in distance or rate parameters)
J	1	JOB	Set external START/STOP control mode
L	1	LEFTRIGHT	Set external direction control mode
N, n	3	NUMBER	Set distance ie no of steps to be taken
O	1	ONESTEP	Take one step immediately
P, p	3	POSITION	Declare target position (absolute)
Q*	*	QUIT	Quit programming and re-enter command mode
R, r	2	RATE	Set rate parameter, r
S, S	2	SLOPE	Set ACCEL/DECEL Ramp rate (1-255)
T	1	LOOP TIL	Loop 'TIL' external START/STOP line is high
U	1	UNTIL	Wait until pin 38 (program starter) is high
+	1	CW	Set direction clockwise
-	1	CCW	Set direction counter clockwise
ϕ	1	COMMAND	Exit from programm execution to a command execution

* Except this all commands terminate with 'RETURN' character

7.2.2.1 Binary Equivalents of the COMMANDS (Binary Data Mode)

In microprocessor control of the CY500, it is advisable that the 22 high-level language commands be coded in Assembly language while more complex calculations can be made in a high-level language available with the micromputer (as is described in chap. 9 in HPESYS). The machine language coding introduces a snag if a set of binary data happens to coincide with the binary decoding of the 'Q' (QUIT) command. In this case the program execution will be halted! To avoid this, the number of bytes of data to be sent to CY500 is specified initially and a count is updated after each byte until the data/command format is complete.

Table 7.1 Commands and associated Data Counts

COMMAND	BINARY CODE	DATA COUNT	DATA 1	DATA 2
A	0100 0001	0		
B	0100 0010	0		
C	0100 0011	0		
D	0100 0100	*		
E	0100 0101	0		
Ff	0100 0110 b7 b0	1	FACTOR	
G	0100 0111	0		
H	0100 1000	0		
I	0100 1001	0		
J	0100 1010	0		
L	0100 1100	0		
Nn	0100 1110 a7 a0 b7 b0	2	No of Steps	
O	0100 1111	0		
Pp	0101 0000 a7 a0 b7 b0	2		Target Positions

Q	0101 0001	0	
R,r	0101 0010 b7 b0	1	Rate
S,s	0101 0011 a7 a0	1	Slope
T	0101 0100	*	
U	0101 0101	*	
+	0010 1011	0	
-	0010 1101	0	
	0000 0000	-	

* Not used in binary mode

(In microNOVA assembly coding the binary numbers would be expressed as octal). In the binary mode of CY500 operation the 'E' and 'Q' commands may be used as CLEARBIT AND BITSET respectively for pin 31 (the PROG pin). It is required that the 'Q' be followed by the 'E' command in such a program. The 'B' and 'C' commands can also set and clear pin 34 (control output line) but unlike the 'Q' and 'E' commands they can come in any order making them suitable for the miscellaneous command mentioned in 7.2.1.

7.3 Parameter Decelerations

The F, R, S, N and P commands must be followed by the appropriate data value as shown in the Table 7.0.

R parameter command (2 bytes)

This is the rate command which specifies the stepping rate in the multiple stepping operation mode (as opposed to single stepping mode). The r value ranges form 1 to 254 corresponding to stepping rates of 49 steps/sec to 3730 steps/sec respectively (for 6MHZ standard clock).

The governing equations are:

$$\text{Steps/sec} = \frac{12500}{(256 - \gamma)} \qquad \text{for } 0 < \gamma \leq 200 \qquad 7.1$$

$$\text{Steps/sec} = \frac{12500}{(257.344 - \gamma)} \qquad \text{for } \gamma > 200 \qquad 7.2$$

The command has the form: R r

(The constants in eqns. 7.1 & 7.2 are fixed in LSI H/W).
It can be appreciated that the graphs of eqns. (7.1) or (7.2) are non-linear and that microprocessor calculation time would be required in a multi-axis drive, since the r values cannot be directly scaled from one another. Further considerations of this will be given in the scheme for acceleration/deceleration and feedrate control. From Equ. 7.2, the stepping rate ranges from 50 to 3730 steps/sec using a 6MHZ clock. If a clock frequency, fc, is used, the stepping rate is modified by the ratio (fc/6MHZ).

F Factor command (2 bytes)

If the value of r is as calculated in eqn. (7.1) or (7.2), then the rate divisor factor, f, will be unity i.e. F 1 will be commanded, and when r equals zero, the lowest speed results (i.e. 48.8 steps/s). However for applications where speeds lower than this are desirable (e.g. Dirk's map-displaying μP system (5)) the value of f is greater than 1 and f is used to reduce (divide) the rate specified by r. The command is F f where $1 < f < 255$. Generally speaking for machine tool CNC f would equal 1.

N Command: Nn (3 bytes)

n = number of steps from the present position, where, $0 < n < 64K$, in the relative mode. The n value is stored as two bytes (table 7.1) in the program buffer.

P Command: Pp (3 bytes)

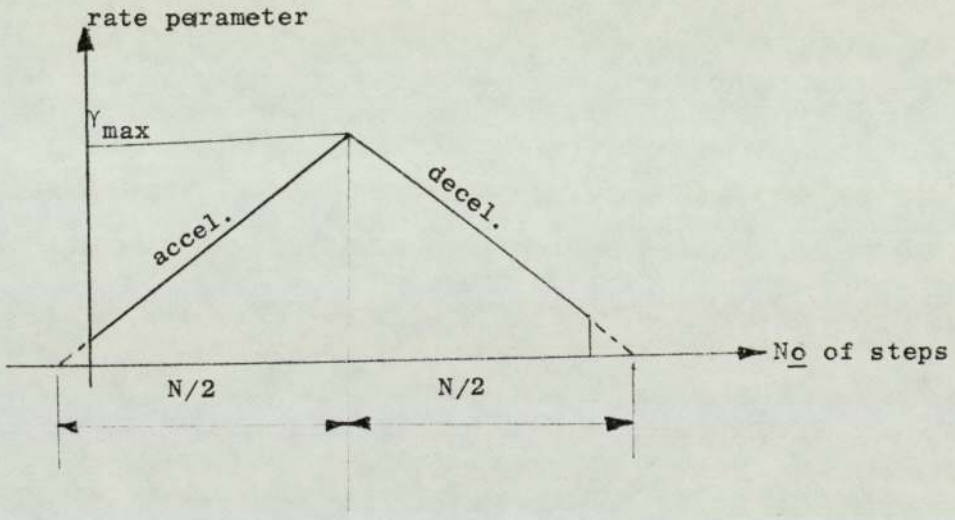
This is the 'POSITION' command which fixes the absolute mode of operation. The value of P is the target position from absolute zero. The ATHOME command 'A' can also be used to define the absolute zero.

7.3.1 S Command - Acceleration/Deceleration

This is the SLOPE value which specifies the acceleration and the deceleration for the command total number of steps, n. It must be given for stepping rates above the maximum starting frequency of the stepping motors to prevent the CY500 jumping instantaneously to the rate specified

by r. The command is S s where 'S' is the increase in rate parameter r, after each step is taken from rest. The initial value of 'r' is forced down to 1 automatically by CY500 and from there 'r' is incremented (acceleration) by 's' each step until the commanded value of 'r' is reached when the speed becomes constant. (From equation 7.1 it is evident that the starting speed is 49 steps/sec whenever 'S' is specified) - CY500 also calculates the number of steps to be taken out of the 'N' value before it changes 'S' to '-S' and thus starts deceleration until the target position is reached.

7.3.1.1 Constraints



In the above diagram if r_{max} = max rate parameter commanded and N is the commanded member of steps to be taken, then the ramping characteristics of CY500 imposes a limit on the value of r. Since the 'r' value increases by 'S' for every step, then $\frac{r_{max}}{S} < \frac{N}{2}$, $(r_{min} > 0)$ 7.3

also since $0 < r < 255$,

$$s < 255 - r \qquad 7.4$$

7.4 Multiple Axes Acceleration, Deceleration and Feedrate Using CY500

7.4.1 Snags in the use of CY500 rate parameter

Investigations by Maginot and Oliver (6) identified that exponential ramp up and inverse exponential ramp down types of acceleration and deceleration control (fig. 7.6a) constitute the ideal strategy for the control of stepping motors. Plas and Blommaert (7) demonstrated this mathematically with the torque/speed characteristic curve of stepping motors. In their CNC system they split this ideal curve into zones defined by arbitary time constants which they fixed by stepping rate ranges. Rahmen (8) approximated the ideal curve by straight lines defined by frequency breakpoints. This solution can be recommended as being ideally practical for microcomputer control in systems capable of high stepping rates.

The step/rate parameter curve of CY500, (fig. 7.6b) is a relatively poor approximation of the ideal solution. The result is that there could be instability at the top end of the accelerations and decelerations when operating at speeds equivalent to frequencies considerably higher than the maximum starting frequency. (The CY500 internal ramping scheme could be tolerated only at lower slewing speeds).

A second snag is the likley profile error which would be introduced during the acceleration and deceleration phases in linear interpolation. This arises from the non-linear relationship between the rate parameter, γ, and the stepping rate (steps/s). If there were no acceleration and deceleration phases the exact values of γ which would yield the stepping speed relationships in both axes can be precisely calculated from equation 7.1 or 7.2 and there would be neglible path error.

Because of these limitations, the author cannot fully recommend the use of the internal ramping facility for multi-axis NC. The next section suggests an approach to solving this problem.

7.4.2 Variable Rate Control

The problems discussed in the last section can be solved by making use of the external start/stop mode incorporated in the design of CY500. It has been stated in section 7.2.1 that the TRIGGER pin 30 can be used for a kind of 'jogging' operation. A single step would be taken every time the microcomputer toggles it from high to low (fig. 7.8) and this means that a programmed pulsing of pin 30 will output a controlled pulse pattern at the motor phases. An alternative method would be to use the 'JOG' command in place of the 'GO' command and to place the CY500 in the external start/stop mode. In this case the μC can start or stop the outputting of pulses to the motors by controlling pin 28 (XSS/TILL pin). These two possible solutions are summarised in the table below.

	XSS/TILL (Pin 28)	TRIGGER (pin 30)
Case 1	Step when low	Always low
Case 2	Always low	Step when low

In case 1 the μC could dedicate a control bit to the TRIGGER pin which is held low while configured pulses (e.g. accel/decel) are input to (XSS/TILL) pin. If the second choice is made (case 2) pin 28 is held low by the control bit while the ramping signals are sent to the TRIGGER pin.

When the CY500 is used in either of these modes the duration of each output pulse would be fixed by the programmed rate parameter while their interspacing is controlled by the processor. The ability of the LSI to do internal counting would be retained provided the 'N' command is given. If this is not required the microcomputer can stop the motor externally. The disadvantage is that the μC has to do more work by providing for acceleration and decelerations as well as position and feedrate controls while the CY500's ramping feature remains redundant. It should be remembered, however, that the CY500 is also the stepping motor four phase controller and (fig. 7.5) SY6522's could be employed to relieve the μC of the counting as well as help in the ramping of the motors. Even then there would still be capacity to assist in the miscellaneous functions mentioned in section 5.2.

7.4.3 Using several CY500's in a Common Data Bus

It has been mentioned earlier that CY500 can provide handshaking facilities with a standard interface bus (e.g. IEEE488, ref. 100). Its BUSY/READY and WR lines can be used to configure 'Talker' and 'Listener' relationships between CY500's and the host μC via this data bus (fig. 7.3).

Fig. 7.4 shows three CY500's (ref. 3) interfaced to a μC via an 8-bit I/O data bus. The WR lines are used for 'chip select' or address for the CY500 being programmed.

7.4.4 Interfacing to CY500 with SY6522

It is possible to interface the CY500 to a µP via programmable I/O devices such as the SY6522 (chapter 5). In fig. 7.5 the PA bus (SY6522) is connected to the CY500 data bus (DB0-DB7). The suggestions (section 7.4.2) for reducing the problem of the non-linear rate parameter could be implemented by generating the acceleration pulses from PB7 (SY6522) to the XSS/TILL pin (CY500). PB6 may be employed for counting pulses for the purposes of timing accelerations and decelerations. In a multi-CY500 system one SY6522 would then act as the µP I/O LSI for one CY500.

7.5 Conclusion

This chapter (like chapter 6) has been included in this study to demonstrate further the trends in the design of special purpose LSI's and where they may fit into the NC industry. That they are being made today is evidence that the market for them exists, following the discussions in section 2.8.

The CY500 is still in its infancy as the manufacturers are reported to be adding important features and making further improvements. The author would suggest that any future development of the CY500 by its designers should include the SY6522 programmable interval timing flexibility described in chapter 5. Given these improvements, the future CY500 benefits greatly by its special features that it is also the SM logic driver as well as co-processor. Then applications designers of microcomputer controlled NC machine systems would have another processor-like LSI to enable them to achieve some of the benefits of a multi-processor system.

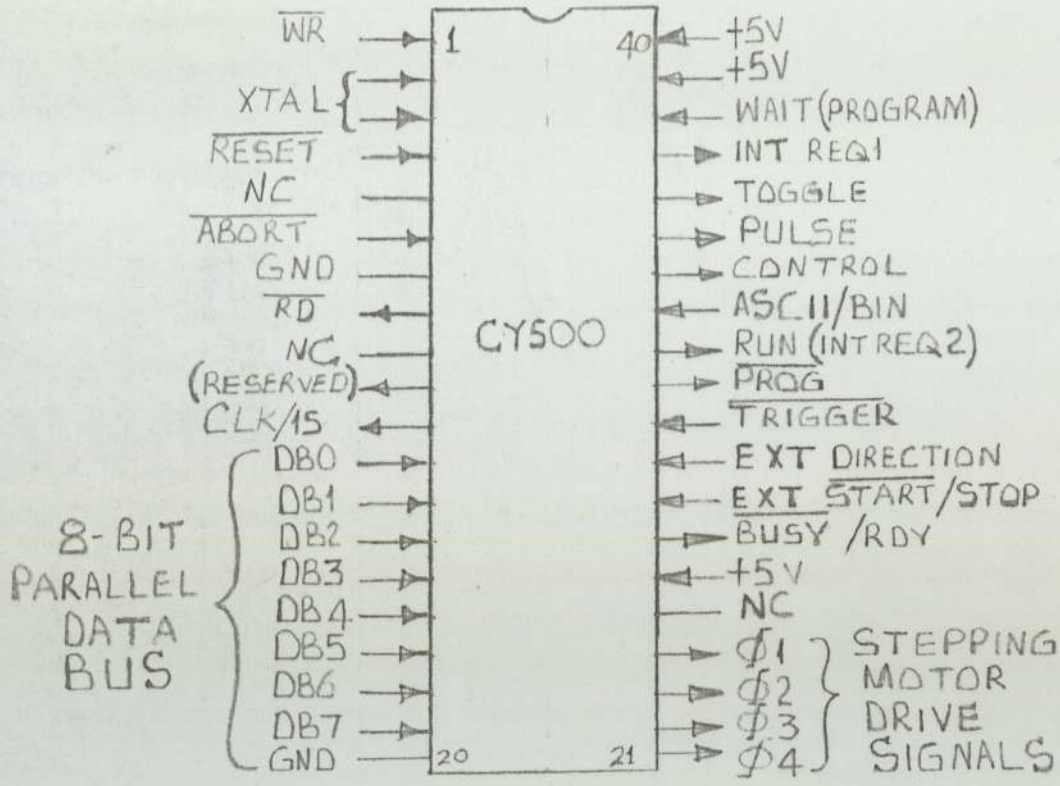


FIG 7·1 STORED PROGRAMME SM CONTROLLER

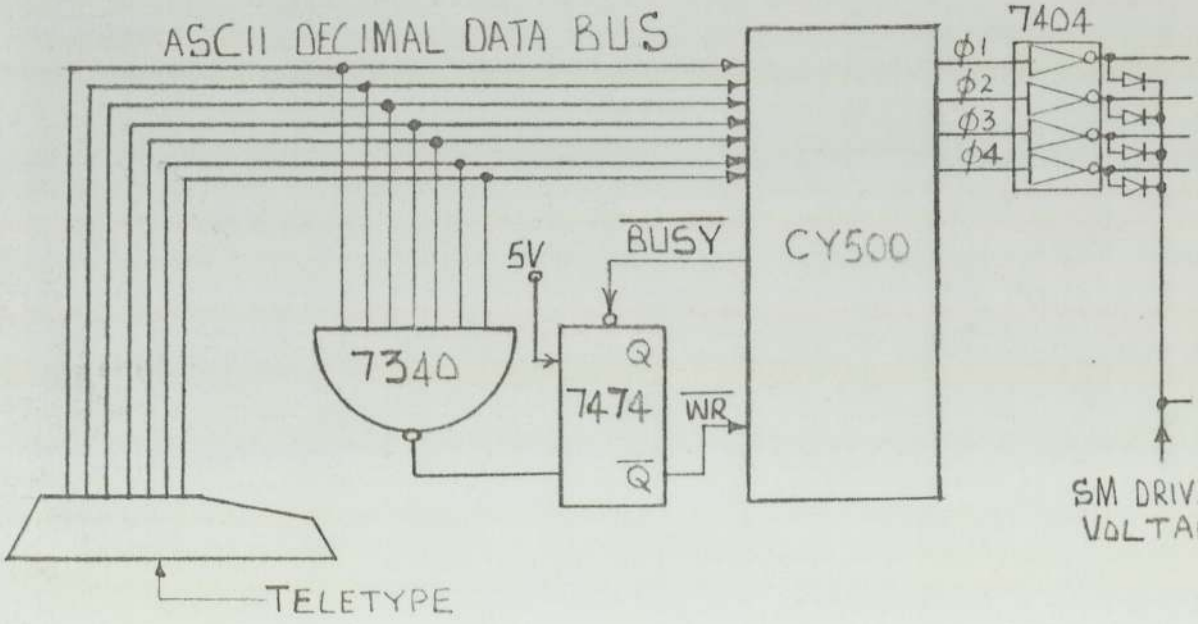


FIG 7·2 PROTOTYPE DEVELOPMENT FOR A SIMPLE SYSTEM

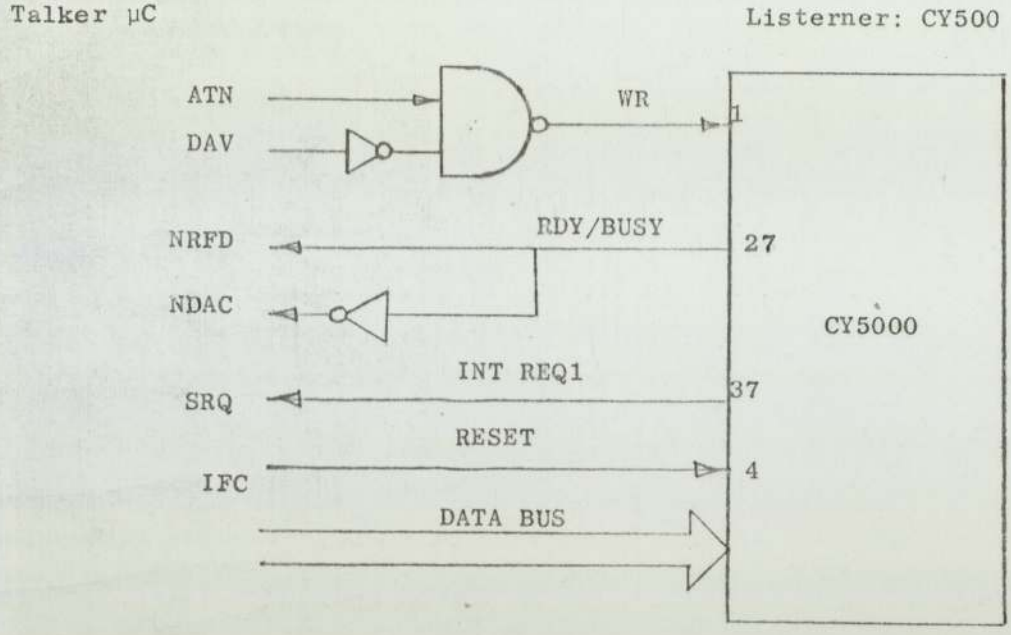


FIG 7.3 CY500 Interface with IEEE488

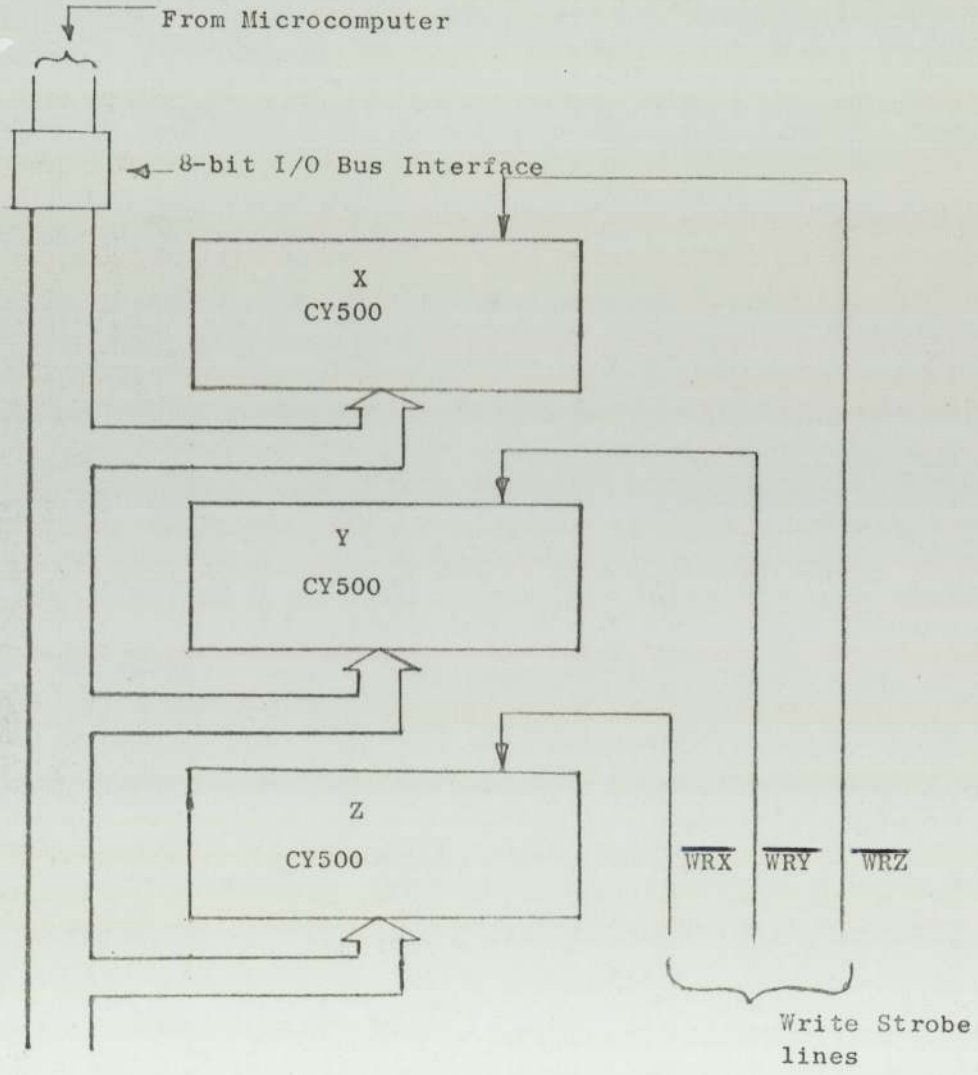


FIG 7.4 3-Axes CY500 System with $\overline{WR}$ used for chip select

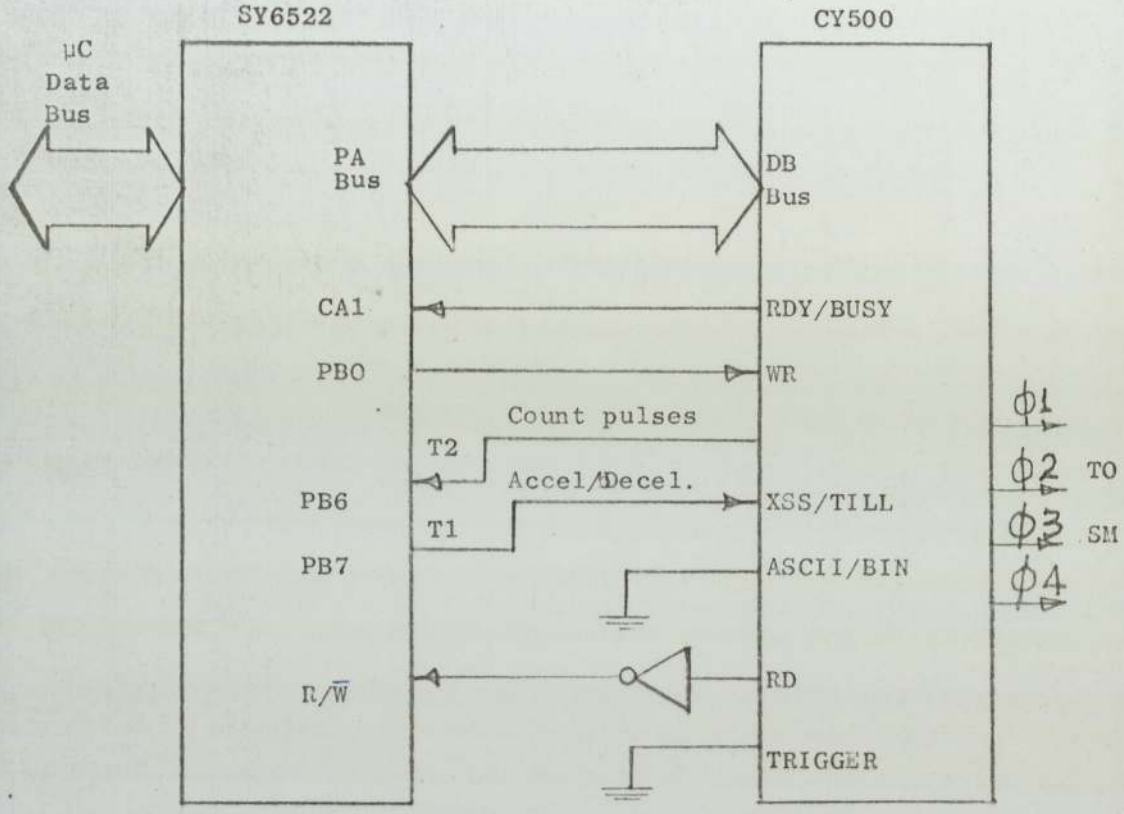


FIG 7.5 CY500 Interface with SY6522

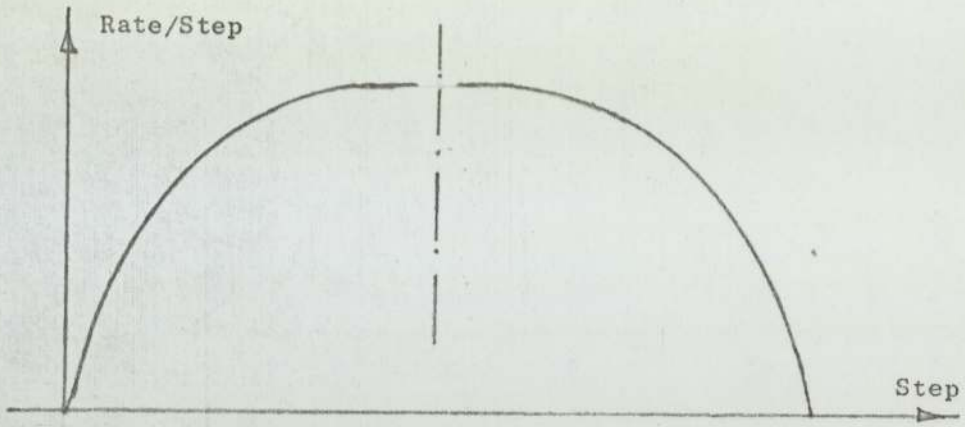


*FIG 7.6a Ideal aceleration curve

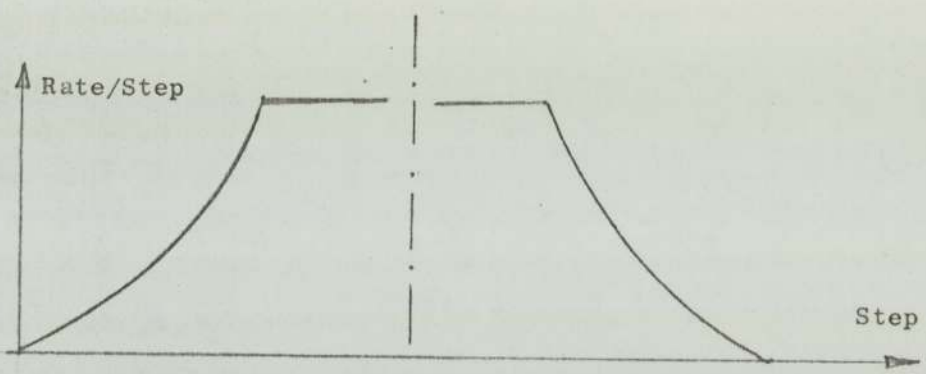


*Fig 7.6b Actual Acceleration Curve

*(The Frequency/Time curves are similar in shape, though their dimensions differ)

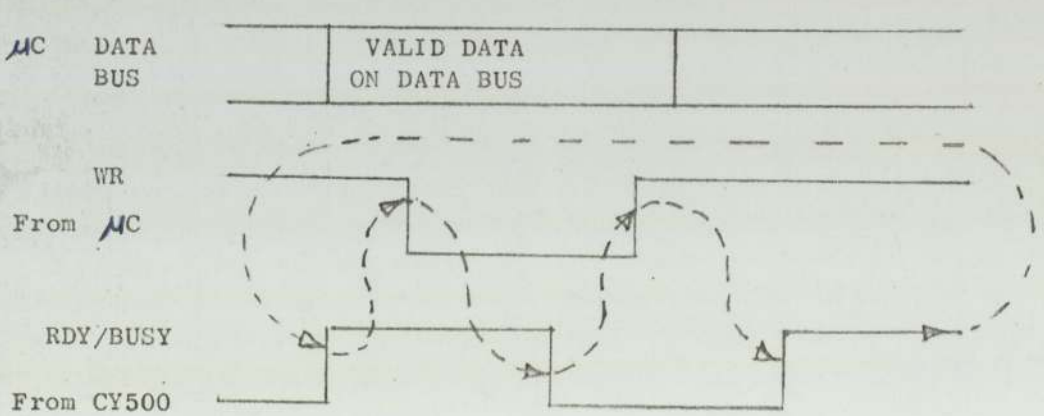


FIG 7.4.3 Handshaking protocol for CY500 in parallel data mode

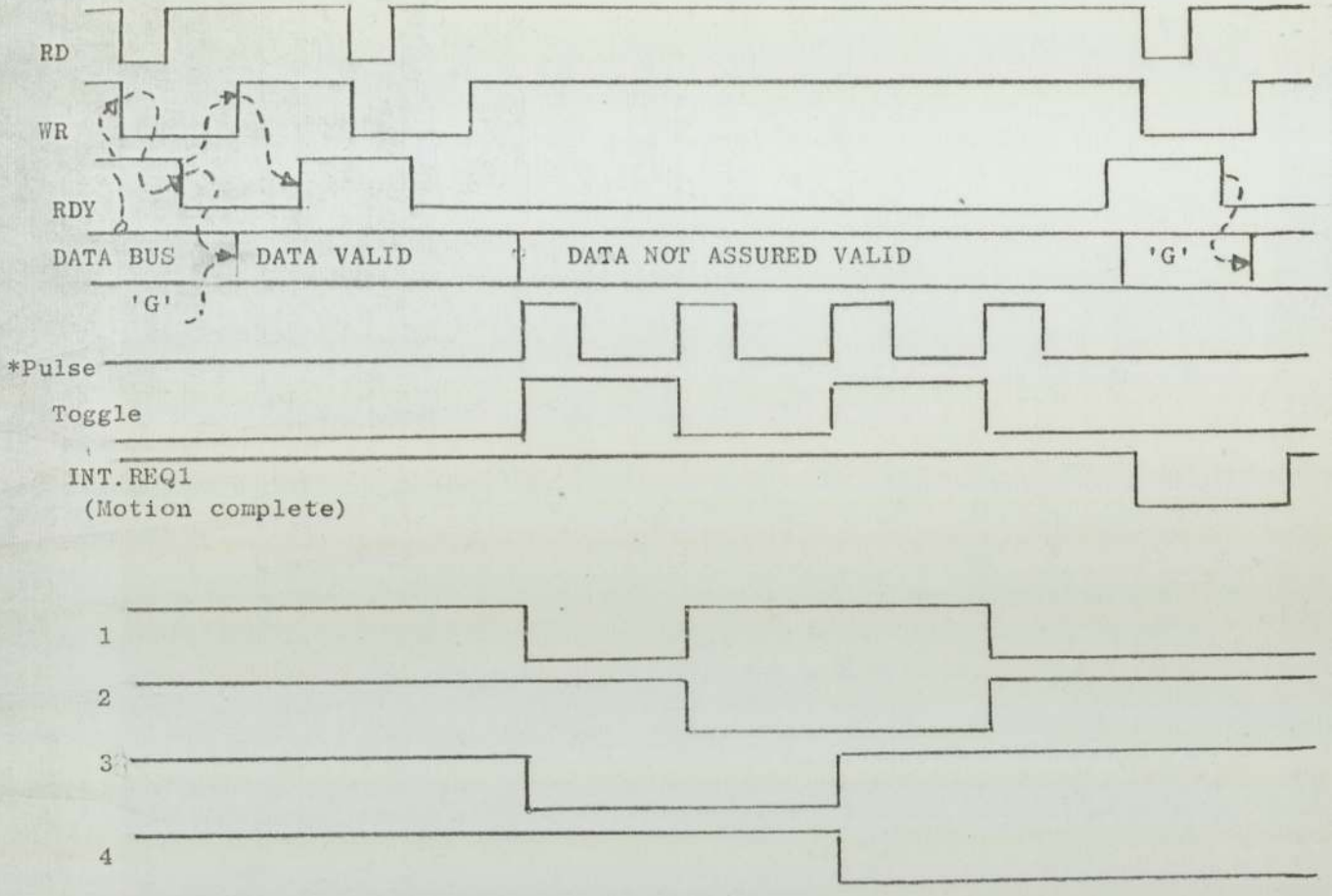


FIG 7.7 Timing Diagram for WR, RD and the 'G' command

* Here only four steps are commanded

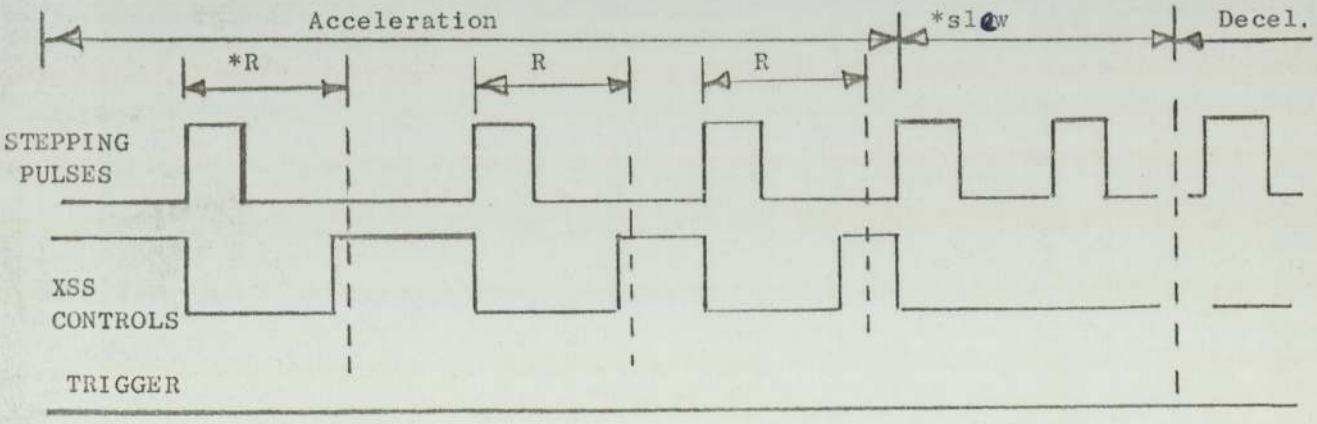


FIG 7.8 Possible accleration, slow, decceleration control thorugh XSS pin from external source (eg SY6522)

CHAPTER 8

THE MICROCOMPUTER SYSTEM SOFTWARE FEATURES - THE DISC OPERATING SYSTEM, DOS

8.0 Introduction

It was mentioned in section 4.0 that the microNOVA is supported by the established S/W development aids from Data General Corporation. The general purpose nature of the μC is enhanced mainly by the fact that its S/W is based on floppy diskettes i.e. disc operating system (DOS). Each of the standard diskettes used in the microNOVA has a capacity of 157K words and having dual diskette drives the diskette back-up capacity is doubled. The S/W's supported by DOS include compilers for FORTRAN, Interpreter for BASIC, Assemblers (Macro and Extended), Text Editor, S/W debugging aids and other general or special purpose DOS files.

Each of the DOS softwares is called a 'file' and the user selects the files appropriate for his application and generates these on his own diskettes. Since diskette space is limited, the file selection process has to be judiciously carried out, otherwise 'file space' will be exhaused by the inclusion of files unnecessary for the designed application. There are certain files, however, which have to be included no matter what application is envisaged. Among these are the files constituting an integral part of theDOS system and essential for communicating with it. The three principal ways to interface the application system with DOS are:

(a) through the console using DOS's 'Command Line Interpreter' (CLI), i.e. by software.

(b) through DOS system calls in a program

(c) via task calls at the program level.

Of these the Command Line Interpreter (CLI) is the most important being interfaced with the system console and providing the 'echo' to the operator during communications between the microcomputer and the console. The system and task calls are inserted as program instructions, if necessary when using the DOS, e.g. for I/O's.

8.1 The Command Line Interpreter

The CLI is the system utility program which accepts command lines from the operator console and translates them into Commands to the DOS. Being the interface between the input medium and the DOS, the CLI is used to create, organise and maintain the application programs (i.e. files). Through the CLI other DOS files can be accessed whenever required for program editing and other development proceedures. After μC initialisation, or DOS bootstrap (start up), or after the execution of a program, the DOS hands command back to the CLI which then indicates the readiness for further instructions by outputting the 'message prompt', 'R' followed by an automatic carriage return at the console. The next command issued from the console must be recognised by the CLI, if not an appropriate error message is displayed. The CLI can be returned to the console (interrupting from the console) by depressing the standard console keys CTRL and A or CTRL and C. The DOS responds with the message 'INT' (interrupted) and returns the control fully to the CLI which outputs the 'R' (Ready). Some typical CLI commands are described in section 8.3.

8.2 The generation of DOS on user diskettes for CNC

By generating his own system, the designer can tailor DOS for his own application H/W. The S/W that enables him to do this is the system generation file called 'SYSGEN'. Before using this file however, the floppy diskettes have to be processed with certain preliminary DOS files supplied with the computer. Firstly, at least three diskettes are required and these must be formatted with a special floppy disc formatting program called by 'FPY FRMT'. This is followed by initialising the diskettes with 'DOSINIT' (DOS initialisation program) which re-checks the patterns installed by the formatting program before being written itself on to the diskettes from μC memory. It is advisable to initialise as many diskettes as are available and use the system 'COPY' command to make copies of the master system diskettes supplied by the manufacturer. After this, the master diskettes should be stored away in conditions comparable with those advised by the diskettes suppliers. If this is done the S/W on the master diskettes should not be corrupted and will be serviceable should other copies be required in future.

One of the master diskettes is a 'DOS Starter', and its copy becomes the designer's 'DOS starter' diskette. This is used together with one of the diskettes initialised earlier (and installed with BOOT.SV). to generate the tailored system using 'SYSGEN'. The CLI files (CLI.SV and CLI.OL) and DOS 'utility' programs are installed on the user system by SYSGEN. The 'UTILITIES' are such S/W as Extended Assemblers (ASM.RB, ASM.SV), Macro Assemblers (MAC) Relocatable Loaders (RLDR.OL, RLDR.SV), system managing files (SYS.DR, SYS.LB), Text Editor (EDIT), etc. It is important to choose carefully between the two Assemblers, Extended and Macro-Assemblers, whereas the latter has more powerful

features, the former is faster and more suitable for CNC. The RLDR, EDIT and SYS files (among others) must be included.

The SYSGEN displays questions offering choices of available DOS facilities for the related hardware configurations which the designer may have included in his application. (Further information on the proceedures for generating a tailored DOS system is covered in Data General's manual no. 0903-000222).

During the system generation, SYSGEN requests the name the user chooses to give his new system. This is the only name (pass word) with which access may be gained to the system for future use. Before the user can attempt writing preliminary programs, he should become familiar with the basic system commands. Many of these are the CLI commands.

8.3 The Command Line Interpreter System Commands

The CLI and DOS system commands are crucial to programming the microcomputer. Table 8.0 gives examples of some of the more common ones.

*Table 8.0 Examples of some CLI and System Commands

CLI COMMAND	SYSTEM COMMAND	INTERPRETATION FOR MICROCOMPUTER
DIR	.DIR	Open and/or Initialise a file director or disc drive
CDIR	.CDIR	Create a new file director on disc
CREATE	.CREA	Create a general file (for one program)
GRAND	.CRAND	Create a general file (for one program)
DELETE	.DELE	Delete an existing file (and program)
RENAME	.RENAM	Rename and existing file
INIT	.INIT	Initialise a directory or device
LINK	.LINK	Create a link entry to a file in another directory
UNLINK	.ULNK	Delete a link entry
CLEAR	_	Set a file's use count to zero
CHATR	.CHATR	Remove the protecting attributes of a file
RELEASE	.RLSE	Release a directory or diskette unit from the system or 'unbookstrap' master diskette.

*There are several other commands and these are described in manuals usually supplied with the microcomputer and DOS systems.

The microNOVA has certain attributes which can be attached to filenames to prevent them being deleted in error; e.g. C1.SR, where C1 is the filename and .SR is the attribute. The system automatically imparts attributes to 'open' files especially in the event that the system 'crashes' (possibly due to H/W malfunction) when the file is in the EDIT or DEBUG buffers. Afterwards the file cannot be accessed again for editing, reading, writing or any pre-execution processing until it is formally 'closed' by the "CHATR' command. Unless a programmer knows about this he may struggle for days (with the microNOVA) and still fail to process his programs because the error messages from the system are often misleading.

Unlike the commands from the CLI (and console) level the system commands (from program level) are always preceded by a dot (Table 8.0). Before a program can be written the file which will contain it has to be created and one of the ways to do this is via the CRAND command. Thus an empty file is created on diskette and the end of the process is indicated by the usual 'R' prompt e.g.

Command:	CRAND C1
DOS Response:	R

(Using the existing microNOVA system these DOS operations were found to be rather slow).

8.4 The DOS organisation in memory

The DOS executive, being the framework of the microcomputer operating system, must be resident in memory before any program or system command can be processed. The executive handles 'overlay' and buffer management, the processing of system calls and interrupts, and the maintenance operations on files and their programs (CLI and system command executions).

*Table 8.1 illustrates the memory organisation using the DOS system. For the microNOVA the minimum size of memory to support an application based on DOS is 16K.

* Table 8.1 DOS-Memory Organisation

Address	Residents
	System Buffers
16K	
	Resident DOS
	NREL address (User programs)
	Overlay Nodes
	NREL (User programs)
	Overlay Directory
	TCB pool
	ZREL (User Page Zero)
16_8	DOS page ZERO

In the memory the lowest 16_8 (octal base) locations are the DOS 'page Zero'. These are reserved for various program and system interrupt entry points to the second area of DOS which is located at the top of memory. The highest portion of DOS is a group of memory-resident system buffers used to receive system 'overlays' and files for buffered I/O transfers during programe execution.

*K = 1024 words.

The 'page Zero' for application programs (ZREL address) starts at physical address 16_8 which is usually the user stack pointer (USP) and this extends to location 377_8. Within this area, addresses 20_8 - 27_8 are 'autoincrementing' locations and these are so-called because the data they contain increment by 1 automatically following a memory operation by the microprocessor. On the other hand addresses 30_8 - 37_8 are auto-decrementing locations in the DOS memory map. Therefore these locations can be employed as indirect address pointers during memory reference instructions (if required).

When the application program uses DOS for I/O transfers the required number of channels through which these transfers are made are specified in the 'User Status Table' (UST) which starts at address 400_8. After the UST portion, come the 'Task Control Blocks' (TCB's) which store such task state information as the state of active accumulators and the carry bit (in such accumulatrs) while the program execution progresses. Occasionally a programmer may employ certain codes arranged in blocks called overlays to support the root program. The executable root program is a 'save' file which is memory-resident and can call in data from overlays whenever needed (if the application requires the use of such overlays). When used, the overlay nodes (block segments) and directories occupy the areas shown in Table 8.1

The Relocatable Loader, RLDR, (Section 8.2) utility S/W produces the executable 'save' file by loading the necessary program modules (subroutines) called by the main program. (Explained further in section 8.6). The NREL 'pseudo-operator' instructs the Assembler to arrange the codes and data in the program for execution in normal relocatable (NREL) memory (Table 8.1) in which absolute addresses do

not need to be specified (an absolute address is specified thus: '.LOC ADDRESS'). In a save file loaded with NREL, the starting (lowest address) and the ending (highest address) locations are called 'ZMAX' and 'NMAX' respectively. Using the minimum memory (16K) configuration, the size of NMAX must not exceed 8K words otherwise the program cannot be loaded because that part of DOS occupying the top of the memory requires up to 8K locations (minimum). However, the μC system memory is expandable to 32K (for the earlier μP) and 64K (for the redesigned μP).

8.4.1 The Organisation of DOS files on diskettes

Diskette files are organised in two ways: 'Randomly' or 'Contiguously'?

Random files

Each microNOVA DOS diskette is initialised with a maximum of 608 blocks. Random files are such files which occupy block positions indicated by a 'File Index' corresponding to sequential number, 0-n, where n is an integer (not necessarily equal to 608). A random file may occupy several blocks but has one index number depending on where it stands in the diskette file organisation. These file indexes guide the μP when it is transfering data between the memory and the diskette blocks.

Contiguous files

Contiguously organised files are those having blocks which can be accessed without the need for the File Index because the blocks are located sequentially in an unbroken series of diskette block addresses.

Unlike random files, contiguously arranged files cannot be expanded or reduced in size. When transferring data between memory and diskettes all that the μC needs to do (via DOS) is to access the first block address (or the file name) and then the relative block number within the file. The advantage of the contiguous files over the random types is that they permit faster block access. Their disadvantage is that they are fixed in size. It is recommended that overlay files be contiguously positioned when used for a given root program.

8.5 Programming the microcomputer

Having generated a working DOS system, and having become acquainted with the overall system and commands, it is necessary to refresh one's knowledge of the chosen programming languages. For high level language the microNOVA offers both BASIC and FORTRAN. Two versions of a FORTRAN IV compiler, 8K and 16K, are available. The 16K compiler is considerably more powerful than the 8K version but obviously it requires more than the 16K of system memory for its application.

For CNC applications it is also necessary to understand the system's Assembly language which is faster for low level machine control than the existing high level languages. However it is much more tedious. In the HPESYS the Assembly language is employed for programming the interface LSI, while the program data is transferred from Fortran level calculations of coarse interpolations and feedrates (discussed in chapter 9) using the 8K compiler. The benefits of using a high level language have been discussed in section 2.1.3.

8.5.1 Programming the microcomputer in FORTRAN

To use Fortran IV for programming the μC, four files are required on the diskette: the three Fortran file S/W's (FORT. SV, FIVNS8.SV, FORT.LB) and the Extended Assembler (ASM.SV) which the compiler requires to produce the binary file. The FIVNS8.SV is the 8k compiler mentioned above. (The more powerful 16K compiler is called FIV.SV).

Program Time: Having created a random file, C1, using the CRAND command, Fortran program statements are transferred to it from the keyboard, after The command:

```
XFER $ TTI C1 ↵
```

in which $TTI is the device name of the keyboard. The Fortran file is closed with CTRL and Z command at the end of the program input.

Compile Time: To compile the program, the command

FORT C1 ↵

is given, and at the end of the compilation, the message 'PROGRAM IS RELOCATABLE' is displayed, followed by the 'R' prompt. If any syntactical errors exist, the 'R' is not displayed until the appropriate error codes have been printed. The Text Editor (EDIT) is then used to correct the program which is later recompiled. This process goes on until the 'R' prompt appears without error messages preceding it. The new file produced by the compiler is a relocatable binary file, C1.RB.

Load Time: The binary file is transformed into an executable 'save' file using the RLDR:

RLDR C1, C2, C3... FORT.LB↵

where C2, C3 .... are other FORTRAN modules (subroutines) called by the main program C1 (The subroutines must have been compiled separately earlier by FORT). The 'FORT.LB' are the Fortran libraries grouped together under FIV. The result of the RLDR operation is a save file, C1.SV.

Error messages are also displayed during the RLDR stage or 'load' time. If these occur the program is sent back to the edit stage and the whole process is repeated until the program is successfully loaded.

Execution Time: The execution is started with the command:

C1 ↵

(followed by carriage return)

If the program does not execute correctly, it is referred to the editing stage and after recompilation, it is reloaded and executed, until it runs correctly.

It is not necessary to delete the -RB or -SV files before the editing process is repeated because DOS suequentially rechecks and corrects the earlier -RB and -SV files as the appropriate function progresses.

More detailed information on the Data General Fortran IV commands is given in the FORTRAN IV User's Manual No. 093-000053.

8.5.2 Programming the microcomputer in Assembly Language

The FORTRAN compilers and Relocatable Loader translate the FORTRAN source files into machine-level codes or binary numbers which physically direct the microprocessor. The Assembler also translates the Assembly language instructions into the binary codes which the μP recognises. Unlike the FORTRAN commands, each of which yields several binary-coded machine instructions after loading, each Assembly command line translates to one binary-coded machine instruction. Although the general Assembly programming procedures for microcomputers are similar, the instruction sets and their related mnemonies are usually different. Therefore every instruction set has to be fairly mastered before a working program can be created to suit a particular application.

With the microNOVA each assembly instruction is translated by the Assembler (Extended Assembler, ASM, or macroAssembler, MAC) into a 16-bit binary field with components as described in section 4.1.6.

The assembly language files are created with the CRAND command, as for FORTRAN. The programs are entered one line per instruction after the initial XFER command has been entered via the keyboard.

Assembler Stage

The -RB file is produced with the Assembler command (Extended Assembler):

```
ASM  F ↵
```

where F is the source program. The usual editing procedure is carried out if the assembler displays syntactical errors.

Loading Stage:

The final executable save file is produced by using the relocatable loader, thus:

```
RLDR F ↵
```

In the Assembly Language program loops are achieved with the jump to subroutine (JSR) instruction. On executing the JSR instruction the µP stores the current contents of the program counter into Accumulator 3 automatically and so the content of this Accumulator is stored (saved) in a memory location specified in octal on entry to the sub-routine. After executing the subroutine the µP is returned to the last calling position by the instruction:

```
JMP 0, 3 ↵
```

and the µP expects Accumulator 3 to have been restored before this instruction.

The various system calls (e.g. Table 8.0) are used only at the Assembly program level and (one call, such as those in Table 8.0) is used to get the memory location counter to a chosen address.

In order to implement the required procedures using the limited capacities of the existing microNOVA a modular design of FORTRAN and Assembly subroutines was used (section 8.6 and fig. 9.0). This required about 8K of memory. High-level language programming of microcomputers is being continuously developed and at the moment certain problems are to be expected. Among these may be the critical timing involved for CNC applications and the requirement of extra memory facilities. As was mentioned in section 2.1.3, the degree of integration of memory is fast increasing while the cost is dropping rapidly and so the extra memory requirement cannot be a problem in future. When the routines are structured in modules called by a main program, it is possible to perform the initialisation calculations in the high level language and transfer control to the Assembler, to program a supporting NC LSI (or otherwise) interfaced to the μC. The degree to which any timing problems becomes significant will depend partially on the power of the μC, and the function capabilities of the LSI (or other devices) at the NC interface. With the system described in chapter 6 less timing problems would be expected and the available μC time can be more easily employed for management functions.

The problem in the effective use of both high and low level language for μC NC is the linking of these languages for efficient data transfers. This is the subject of the next section.

8.6 Using FORTRAN and Assembly languages in one main CNC file

The interposing of assembly instructions between FORTRAN level commands can be done by declaring each assembly line (entry) with an 'A' but with the microNOVA this has serious problems with an elaborate control program requiring loops. Such loops cannot be called to or from the assembly lines because they are indistinguishable from one another, each having the 'A' at the positions usually reserved for a subroutine or module name.

In the HPESYS however, this difficulty is overcome by organising the Assembly level modules into one file which is separately assembled and then loaded with the FORTRAN main program during load time using the RLDR command:

RLDR C1, C2, C3,..., F, FORT.LB↵

Where C2, C3, ... are FORTRAN subroutines and F is the assembly file called by the main FORTRAN program (Fig 9.0).

The main problem is encountered with the FORTRAN-Assembly interface through which calculated control variables are transferred between the two program levels. This problem is solved in the following ways:

(i) The FORTRAN subroutines that calculate the variables are written thus:

SUBROUTINE INTPL (X_1, Y_1, X_2, Y_2, ...,X_n, Y_n)

where X_1, Y_1, X_2, Y_2, ...,X_n, Y_n are the control variables to be calculated by INTPL. These variables are then referred to as the subroutine arguments.

(ii) In the main program these subroutines are 'called' with the arguments in exactly the same format: CALL INTPL (X_1, Y_1, X_2, Y_2, ...,X_n, Y_n). During the compile time, the command given is:

FORT C $TTO/L ↵

Where C is the main program making all the subroutine calls, $TTO/L instructs the DOS to output the compiled program listings on the teleprinter output. From the listings the relative locations of the subroutine arguments (variables) on the FORTRAN memory stack are noted. Usually they are listed relative to a general stack variable, 'v'. The first named arguments in the called subroutine are listed as in table 8.2:

Table 8.2 Examples of FORTRAN stack variable listing at Compile Time

```
 V = -167          ; - 167 is the relative location of the
                     first FORTRAN stack variable.
X1   V + 0
Y1   V + 1
X2   V + 2
Y2   V + 3
etc.
```

(iii) In the Assembly file, those FORTRAN subroutines which send data to the assembler are declared as external entries (i.e. the external modules containing the required variables), thus

.EXTN DRN, INTPL

Here (in HPESYS) 'DRN' is the direction subroutine and 'INTPL' is the interpolation and feedrate control subroutine all of which transmit data to the assembler.

(iv) After declaring the assembly file as normal relocatable with .NREL 'pseudo-op' a calculation using the format in Table 8.2 follows in the assembly program:

X_1 = -167

Y_1 X1+1

X_2 Y1+1

Y_2 X2+1

etc.

The command, 'JSR @ .FARL', is then given. .FARL is one of the programs in the FORTRAN Library file (FORT.LB) and is used to save the contents of all Accumulators at memory locations chosen by the System. In the program these accumulators are then employed to transfer the variables X1, Y1, etc, to memory locations chosen (by the designer) in the Assembly level program. From these memory addresses they are transferred according to the appropriate sequences to program the LSI's at the interfaces to give the required interpolation and feedrate.

To exit from the Assembly level (to the FORTRAN) another FORT.LB subroutine, .FRET is invoked with

'JSR @ .FRET'

This automatically restores the initial state of the accumulators (reverse of the .FARL functions).

Finally a 'JMP 0,3' command returns the control to the FORTRAN level at the end of the interpolation. The '3' in the command refers to accumulator 3 which has the function of registering the contents of the FORTRAN stack pointer at the time control is transferred to the FORTRAN-ASSEMBLER interface.

As was discussed in section 2.1.3 the use of high level languages in programming μC's is bound to increase by the day because this is a certain way to reduce the cost of S/W. Already many μC systems (e.g. TMS990) include a high level language among their available S/W's.

CHAPTER 9

PROGRAM DESIGN FOR DIRECTION, INTERPOLATION AND FEEDRATE CONTROLS USING MANUAL DATA INPUTS (MDI)

9.0 Introduction

The new HPE milling machine control system (HPESYS) is designed on the basis of simple manual data inputs of position, feedrate and other machine (or management) functions. As was mentioned in section 8.6 the general control file contains separate modules (subroutines) in both FORTRAN and assembly languages. At the FORTRAN level NC data is input manually from the keyboard in response to questions displayed by the μC at 'run-time'. Some of these questions are related to (1) the memory requirements for the part program, (2) the axis position co-ordinates, (3) the component batch size, (4) the required feedrate. The μC then uses the information to calculate the number of steps to be taken by each stepping motor and the interpolated data (numbers) to be written later into the LSI's (SY6522 and also applicable to CY500, chapters 5 and 7 respectively) to control the interpolation and feedrate. For KM3701 (chapter 6) it would not be necessary to calculate the number of steps for each axis, and in this case only the appropriate co-ordinates and the feedrate data would be transferred to the Assembly level.

At this level, the appropriate register control words together with the related NC data are sequentially written into the LSI's. To increase the reliability of the open loop system, the μC reads back the interpolation control data periodically to confirm its validity and

thus monitor the stepping progress. As was discussed in section 2.8, the speed afforded by H/W (LSI's) combines with the flexibility and reduced programming costs of a suitable microcomputer to yield a powerful, versatile and economic NC system within the reach of small firms. While the LSI's are executing the machine control, the μC is free to attend to other calculations to be made, possibly some for management purposes.

The programming procedures for KM3701 and CY500 LSI's have been described in chapters 6 and 7 respectively and further details may not be given here, (these chips are still being improved by the manufacturers). For SY6522 LSI also, details of the registers and peripheral control procedures have been described in chapter 5 but further details of the programming of the internal timers for interpolation, acceleration and feedrate control will be discussed in this chapter. Unlike the μC programmable custom LSI's in chapters 6 and 7, the SY6522 is a general purpose I/O LSI which qualifies for inclusion in the μC controlled stepping motor (SM) NC system by virtue of its two powerful interval timers, T1 and T2.

9.1 The manual data input (MDI) technique used for the microcomputer

Fig. 9.0 shows the flowchart of the main program. In the systems FORTRAN compiler, conversational I/O is permitted from keyboard in accordance with the 'Dimensioned' 'ACCEPT' and 'TYPE' statements programmed in the correct format. As was stated in section 9.0 the part program data is typed in during runtime in response to questions displayed by the μC.

To conform with the S/W requirements, the general format and necessary numbers of the part program variables are introduced to the system with the 'DIMENSION' statement. This enables it to allocate enough memory (stack) locations in advance for the maximum number of part program variables expected. An example of the keyboard communication and resulting calculations is shown below.

```
F1
 NUMBER OF INTERPOLATIONS EQUALS N, N=?

       N= 6
X1 ,X2 ,X3 ..........,XN = ?

.5,1.,.5,-.5,-1.,-.5
Y1 ,Y2 ,Y3 ..........,YN = ?

-.866,0.,.866,.866,0.,-.866
       NO. OF COMPONENTS IN THIS BATCH = NC,  NC= ?

       NC =3
       FEEDRATE = F (IN/MIN),  F= ?

       F=1.5
       IDX    IDY   =

       2       0
       X      Y

    0.500000E  0    0.866000E  0
       IAX      IAY

    1666     962
       NX       NY      NXY

    5000    8660       0
       IFX    IFY    IX     IY

     125     216   B 4167    2405
       IDX    IDY   =
```

*(Fig 9.6a is the resulting shape from this part program).

9.2 Direction and Interpolation Subroutines

The axis co-ordinates are programmed with their proper signs at the MDI stage. Then as the direction subroutine is called by the main program, the co-ordinates together with their signs are transfered and from these the direction variables (1's and/or 0's) are determined. Fig. 9.1 shows the flowchart for direction control.

9.2.1 Linear Interpolation and Feedrate Control

As described in chapter 5, the timer, T1 of each SY6522 is programmed to generate shaped interpolated pulses to each SM. T1 is programmed with numbers calculated in the interpolation subroutine (fig. 9.2). The ϕ_2 clock which forms the SY6522 time base has a fixed period of 4.8 μS, i.e. a frequency of $208333\frac{1}{3}$ HZ.

From fig. 9.3 a

$$R = (X^2+Y^2)^{\frac{1}{2}} \qquad 9.1$$

If F = part program feedrate (in/min),

$$FN = \frac{F}{60R_m} \quad \text{(steps/s)} \qquad 9.2$$

where FN = Resultant feedrate (steps/S)

Rm = machine resolution (.0001 in/step)

Then,

$$FX = \frac{X}{R} \cdot FN \qquad 9.3$$

$$FY = \frac{Y}{R} \cdot FN \qquad 9.4$$

Where FX and FY are step rates (pulse rates) in the X and Y axis respectively.

The numbers to be programmed into the T1 registers (Chapter 5) to generate the calculated pulse rates are:

$$IX = \frac{208333.333}{FX} \qquad 9.5a$$

$$IY = \frac{208333.333}{FY} \qquad 9.5b$$

Where IX and IY are the nearest whole number values for X and Y respectively.

9.2.2 Acceleration/Deceleration Control

In section 3.6.1(iii) it was stated that the hydraulic servo system on the HPE machine could respond accurately to starting speeds up to about 833 and 1000 steps/S in the Y and X respectively. (Each SM has a no-load maximum starting speed of over 2000 steps/S in the half step mode). For this reason, it was decided that whenever the value of either FX or FY exceeded 500 steps/S, the acceleration routine would use initial acceleration numbers, AX and AY corresponding to a maximum starting speed of 500 pulses/S in the faster axis. The initial acceleration number (for X or Y) is from eqn. 9.5.

$$A(X, Y) = \frac{208333.333}{500} = 416.66 = 417 \text{ (constant)}$$

The ratio between X and Y is used to calculate a corresponding number (larger) for the slower axis.

As was mentioned in section 3.5.1 the capability of the hydraulic drives on the axes is such that the range of speeds is limited to 2.5KHZ maximum and so single linear acceleration and deceleration schemes (eqn. 3.15) are justified for this particular application. An acceleration of 10HZ/step (eqn. 9.7) is easily within the capability of the axis servos (and less than 1/10th of the capability of the SM's). Therefore after each step the pulse rate is increased by 10HZ in the faster axis by loading the corresponding number (eqn. 9.11) into the T1 latch after the last pulse generated has gone low, making use of the 'free running mode' described in chapter 5. The number in the latch is automatically transferred to the interval timer to determine the duration of the next pulse cycle.

When the current number, A(X, Y), calculated by the MC in the assembly level control (or read from an acceleration table, ref (54)) is such that:

$$A(X, Y) < I(X, Y), \quad \text{(for X or Y)}$$

Then IX and IY are finally loaded into the T1 latches to generate the interpolated pulse rates.

The detection of the beginning of deceleration has been discussed in chapter 5, equns. 5.1 - 5.5, and makes use of similar routines.

9.2.2.1 Determination of a linear acceleration equation

From Fig. 9.3b. the general linear equation is:

$$FA(X, Y) = mNA(X, Y) + F(X_o, Y_o) \qquad 9.6$$

where m = slope of the linear acceleration

NA(X, Y) = no of steps covered in X or Y before the variable stepping rate FA(X, Y) is achieved.

$F(X_o, Y_o)$ = starting stepping rate in X or Y.

$$m = \frac{F(X, Y) - F(X_o, Y_o)}{NA(X_F, Y_F)} = \text{constant acceleration slope.} \qquad 9.7$$

It follows that if $NA(X_F, Y_F)$ is defined as the number of steps to be taken to reach the constant programmed stepping rate in X or Y, $NA(X_F, Y_F)$ approximately equals $\frac{F(X, Y) - F(X_o, Y_o)}{10}$, i.e. m = 10.

To construct the program for this application it was decided to adopt

$$NA(X_F, Y_F) = \frac{F(X, Y)_{max} - F(X_o, Y_o)}{10} \qquad 9.8$$

Where $F(X, Y)_{max}$ = the faster axis stepping rate, FX or FY.

In this way m could be predetermined and from eqn. 9.6 the T1 load numbers are also determined since they are inversely proportioned to the stepping rates:-

$$FA(X, Y) = \frac{208333.33}{IA(X, Y)} = m\,NA(X, Y) + F(X_o, Y_o) \qquad 9.9$$

$$\therefore \quad IA(X, Y) = \frac{208333.33}{mNA(X, Y) + F(X_o, Y_o)} \qquad 9.10$$

Generally, eqn. 9.10 is equivalent to:

$$n_x = \frac{f_o}{mCx + Co} \qquad 9.11$$

Where n_x = required acceleration number, IA(X, Y)

$f_o = \phi_2$ clock frequency, 208333.33HZ

Co = starting axis frequency, $F(X_o, Y_o)$

C_x = m x No. of steps already covered in X or Y.

For systems involving higher speed ranges, the calculations of eqn. 9.11 using the μC in real-time may be avoided by precalculating the values and storing them in an acceleration table. This can be later read in reverse order for deceleration.

These routines are incorporated with the procedures of chapter 5 to maintain continuity of control of position (eqn. 5.1 - 5.5).

9.3 Assembly level control

The technique used to transfer the NC variables between the microcomputer high and low level (assembly) languages has been described in section 8.6.

Depending on whether the number to be transfered to the LSI register is greater or less and 2^8-1 (= 255), the µC transfers it to the LSI one 8-bit byte at a time via D0 - D7 (figs. 5.12 and 5.2). This corresponds to bits 0 - 7 in the µC and is done by first 'masking' off the high order byte of the number and then 'swapping' the bytes. The appropriate register address and chip select word (µC bits 8-15) is then added (parallel bit addition using an accumulator) to the swapped byte. The µC I/O strobe is then employed to control the R/W line (via a monostable) for the eventual transfer of the whole 16-Bit address/data word to the LSI register (Table 9.0 below)

Table 9.0 Parallel Address/data transfer to LSI by µC

Bits	0 - 7	8 - 11	12 - 14	15
FUNCTION	Data Byte	Register Address	Chip Select (X, Y, Z)	-

During a 'Read' operation to confirm the content of a selected register, the address and chip select bits are appropriately placed on the bus and with the R/W = 1, the register would respond by placing its contents on the vacant bits 0 - 7 on the µC address/data bus.

9.4 Approaches to Circular Interpolation

If the LSI described in Chapter 6 is employed, the circular (or any of the other four interpolations) are chosen by programming the appropriate control word after the necessary NC variables have been transfered. The interpolation problem is thus considerably reduced. In fact the LSI (KM 3701) can be looked upon as a microprocessor dedicated to NC interpolations (being itself μC programmable).

On the other hand if CY500 (Chapter 7) is employed either the Bergren algorithm of Appendix B and ref. (91) or sine-cosine tables can be used to determine the incremental x, y co-ordinates. Then the μC H/W multiply and divide capability could be used to implement the recursive routines for feedrate control using the MTIRA algorithm ref. (90) and this too would require a fast μC.

Although the SY6522 is very suitable for linear interpolations (having the capability to be programmed to generate and count continuous and evenly spaced pusles), its other capabilities are similar to those of a general purpose microprocessor I/O LSI. It would involve recursive calculations to enable it to control a constant feedrate during circular interpolation and that would require a very powerful microcomputer. Therefore it is recommended that this LSI be applied for the peripheral controls (described in Chapter 5) and the pulse generation for the KM 3701 (Chapter 6) which can then be programmed more easily for NC interpolations using a standard μC.

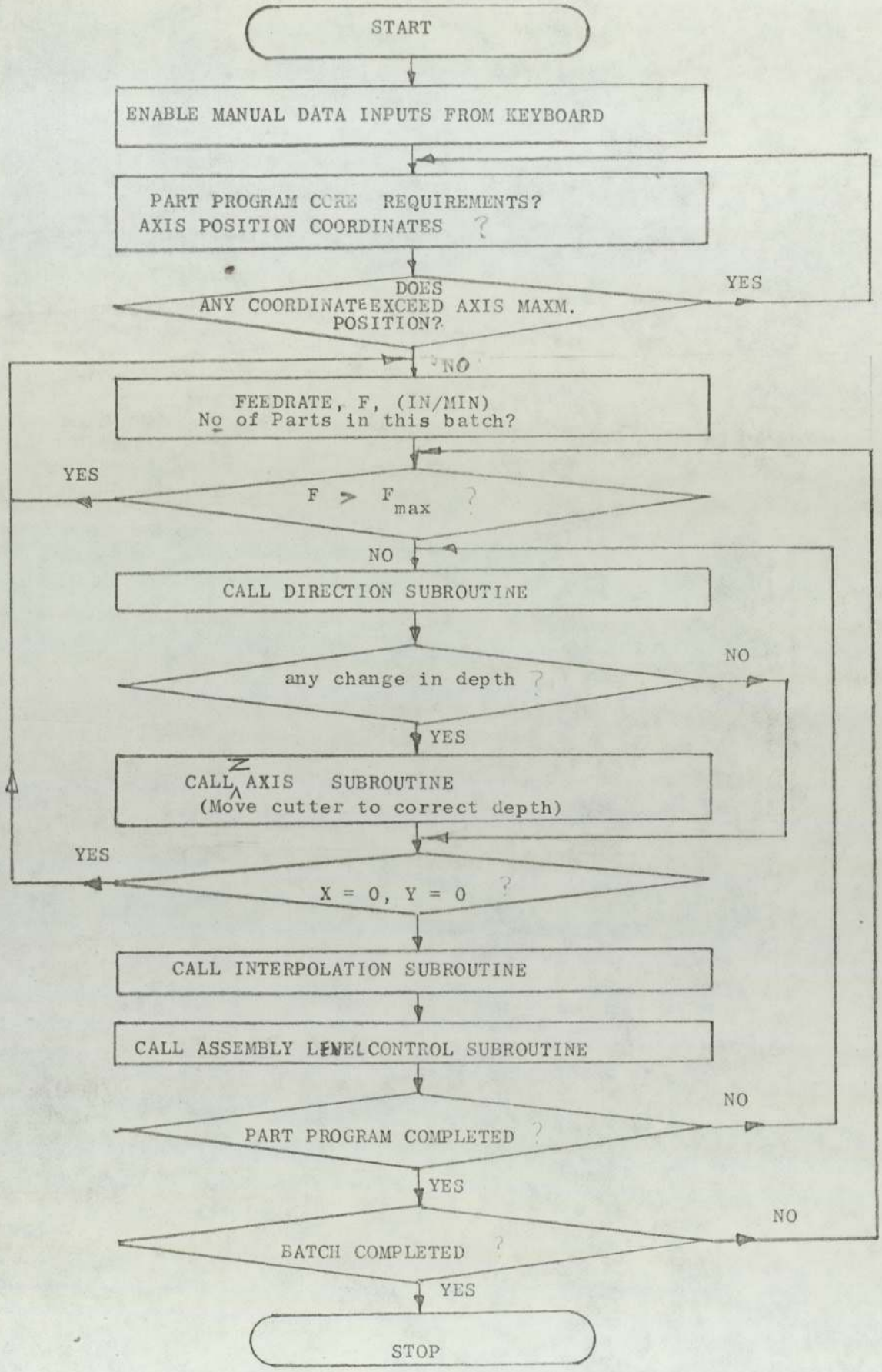


FIG 9.0 HPESYS Main Program flowchart

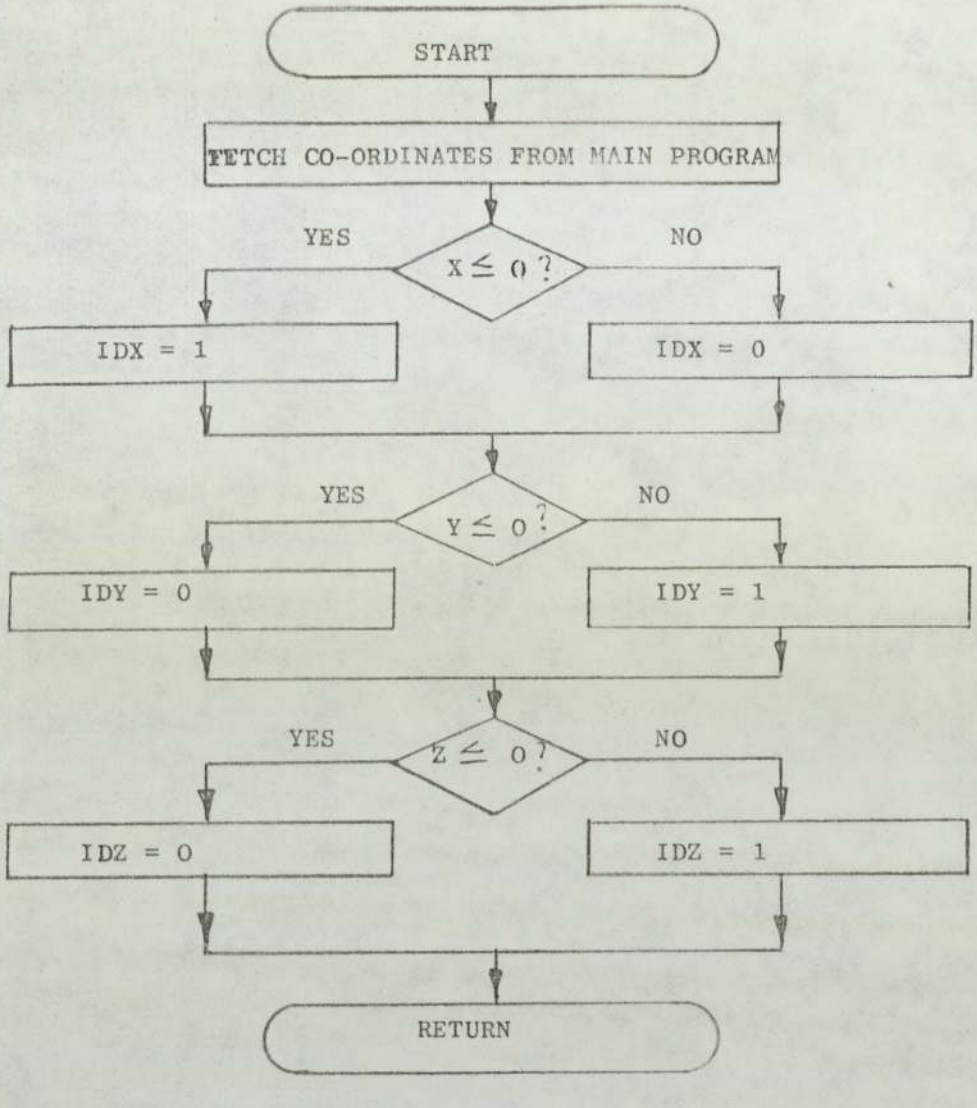


FIG 9.1 Direction Subroutine Flowchart

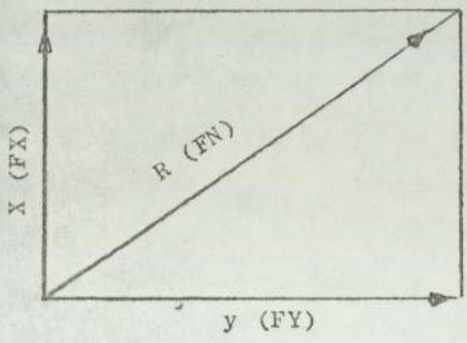


FIG 9.3a Feed Rate

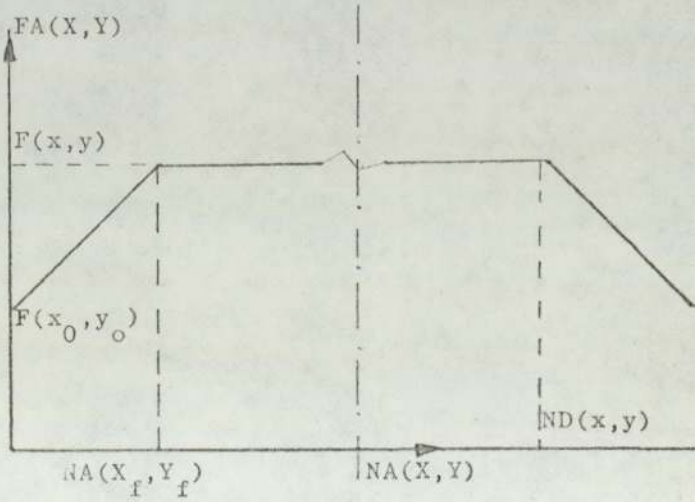


FIG 9.3b Acceleration in Lower speed application

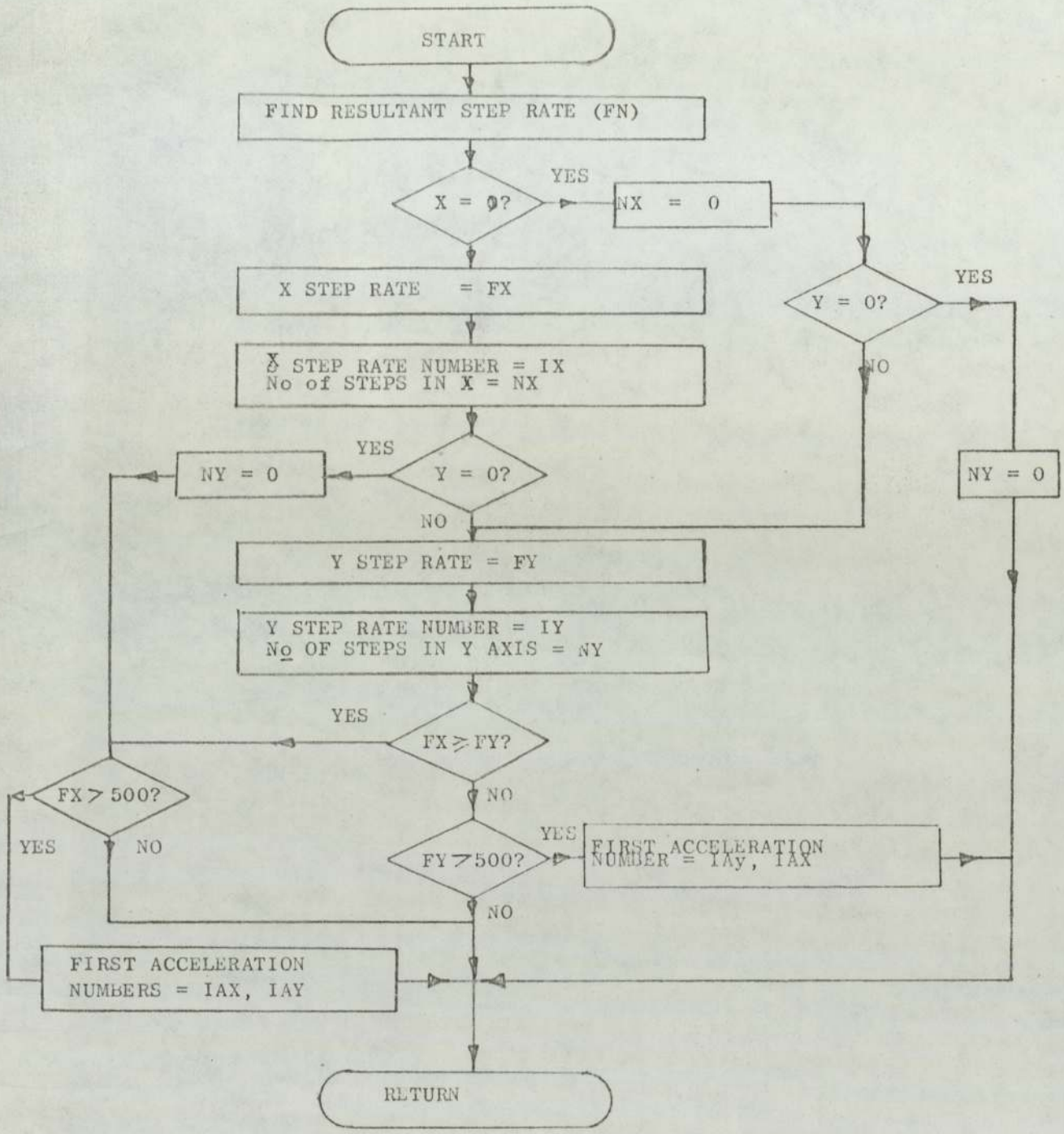


FIG 9.2 Linear Interpolation and Feedrate Control Flowchart (Using SY6522)

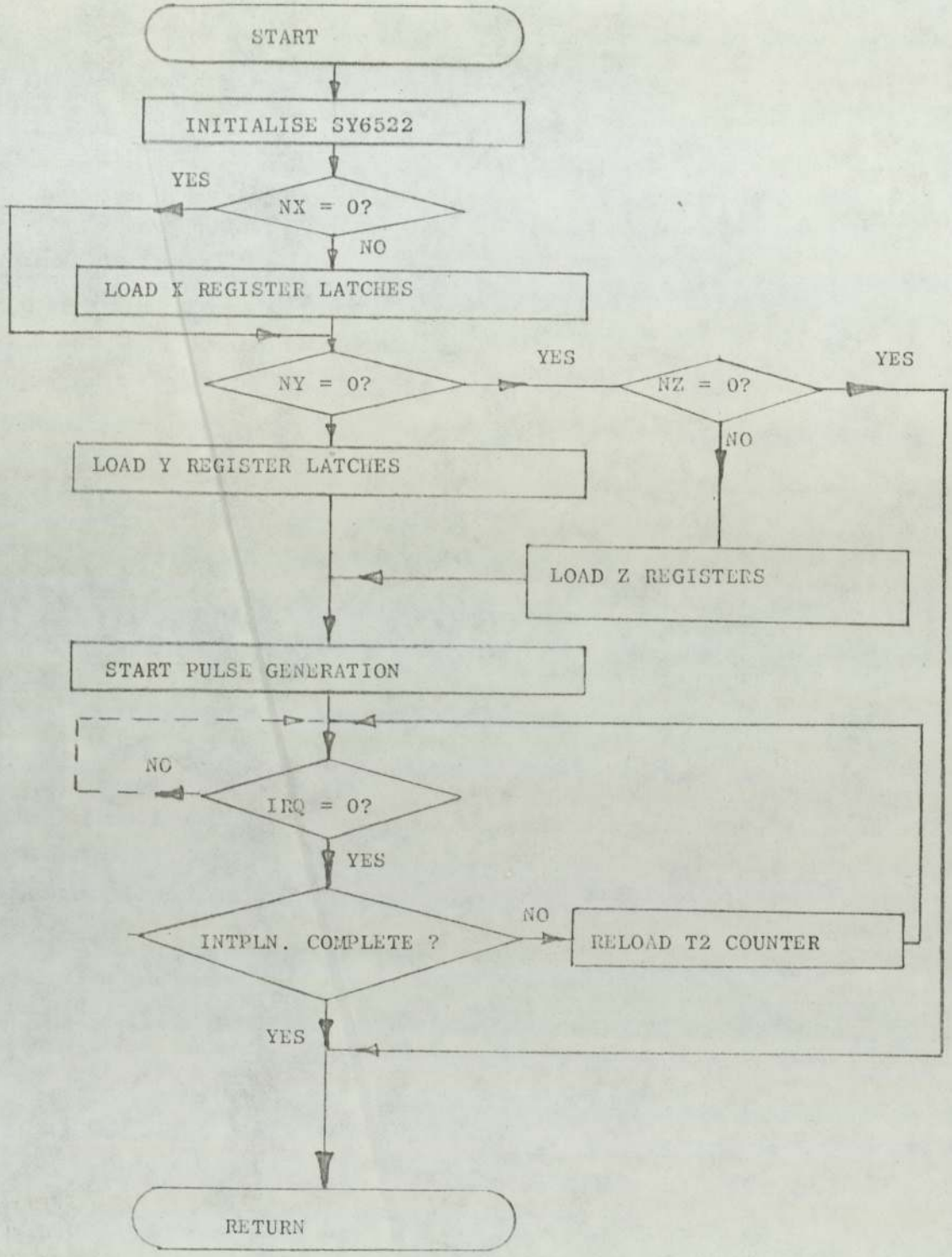
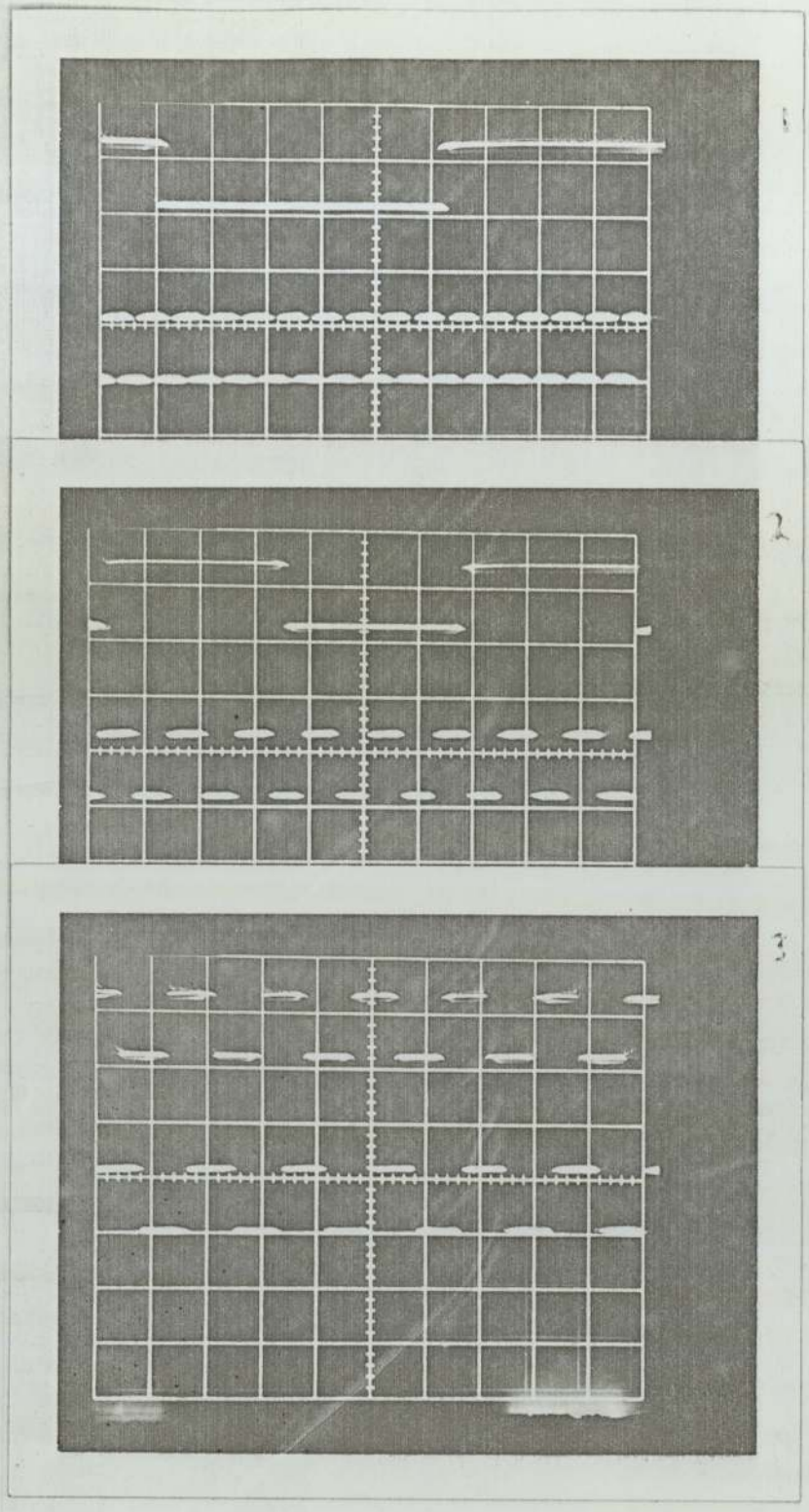


FIG 9.4 SY6522 LSI 'Assembly" Control Subroutine



X = .3552"
Y = 3.116"
F = 3.5 in/min
FX = 66HZ
FY = 579HZ

X = 3.155"
Y = 11.5
F = 2.5 in/min
FX = 110HZ
FY = 401HZ

X = -8.5"
Y = 8.5"
F = 5 in/min
FX = 589HZ
FY = 539HZ

FIG 9.5 Examples of Pulse Generation Using SY6522 under Microcomputer Control

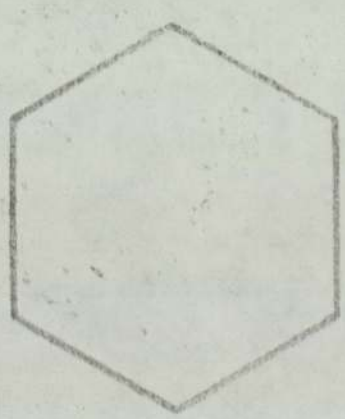


*Fig 9.6a
Shape traced under the
microcomputer (HPE) NC

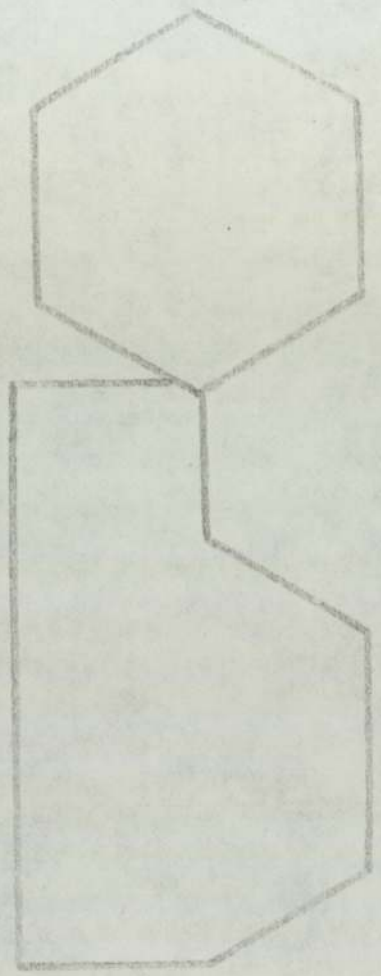


Fig 9.6b
Pencil-traced shape made by the milling machine under microcomputer control

CHAPTER 10

CONCLUSIONS

As a result of this study the following conclusions can be drawn:

10.1 The milling machine - stepping motor servo system

(1) For existing NC milling machines like the HPE, small electric stepping motors can be coupled with rotary and linear potentiometers to form effective actuators for the hydraulic motor and cylinder servo drives respectively.

(2) In the servo mechanism formed it is confirmed that the steady state velocity following error varies linearly with the stepping rate and the feedrate. This error is well within the specified requirements of machine accuracy.

(3) Analysis confirms that the overall stiffness of the members connected together to form the servo systems affects the response characteristics. Increasing the stiffness improves the dynamic performance of the servo drives.

(4) The dither signals usually employed in machine tool electrohydraulic drives to improve their performance does not have the desired effects (at recommended levels) on drives actuated by small electric stepping motors.

(5) The inherent oscillations of the stepping motors already have dither - like effects contributing to the linearisation of the electrohydraulic drives even from the first stepping motion.

(6) Beling's (66) finding that operating a direct-drive electric stepping motor in fractional step modes decreases the capability to overcome friction in the system is also true in a stepping motor actuated electrohydraulic drive. In this case there is a brief deviation from linearity at start up using the half step but this is not so evident in the full step mode.

(7) Viscous inertia dampers are effective in reducing the stepping motor oscillations to satisfactory levels, especially in the hydraulic motor drive application.

(8) Operating in the half step mode can reduce the tendency of the servomechanisms on the axes to respond to the small cycle oscillations ("tweaks") in electrohydraulic drives investigated by Ertongur (92).

10.2 The microcomputer controlled NC system

(9) Manual data inputs of NC data to a microcomputer can be simplified by using a high level language and disc operating system (DOS).

(10) Using the disc operating system, the development costs of the microcomputer numerical control software can be considerably reduced by applying modules of the high level programs for processing the data and generating NC variables to low level programs for the control of an interface.

(11) The interface can be a programmable LSI which generates signals for the numerical control of electric stepping motors. Some of these NC LSI's ('processors' in their own right) have appeared in the market recently and one of them, KM3701, (chapter 6) can be programmed to generate five different NC contours, including linear and circular.

(12) The SY6522 LSI (chapter 5) can be programmed from a microcomputer to generate signals for accelerations, feedrates, position, direction and miscellaneous controls on the NC machine.

(13) The evenly spaced pulses generated with SY6522 interval timers contribute to the smooth performance of the stepping motors under microcomputer control.

(14) The CNC signals transmitted from an LSI interface can control electrohydraulic drives (hydraulic motor(s) and/or cylinder(s)) on a milling machine via the small electric stepping motors.

(15) By using a general purpose microcomputer with good software support (DOS), the cost of NC program development is considerably reduced and so the added cost of LSI's at the interface becomes more justified.

(16) The advantage of an LSI supported microcomputer NC machine system is that it combines the speed offered by hardware with the flexibility afforded by the microprocessor to form a powerful and versatile control system.

The reduced expertise required to implement the hardware and software for such systems makes it possible to develop the system in-house, thus giving the opportunity for a small firm to employ the extra time on the microcomputer for some management calculations if required. Dual use in this way will help justify the cost.

CHAPTER 11

RECOMMENDATIONS FOR FURTHER WORK

(1) For the HPE class of milling machines, the Z axis stepping motor servo system will have to be implemented following the proceedures described for the other two axes. The spindle speed control and miscellaneous functions controls (coolant on/off, job clamping, tool setting) should be instituted on the machine and connected to the peripheral interface (chapters 5 and 6) using a more powerful microcomputer with a standard bit - parallel I/O and memory bus architecture. The circular interpolation would be implemented and the path error analysis carried out.

(2) It will be necessary for the open loop UC control system to establish a datum (postion reference point) from which all co-ordinates are measured. In the event of failure or an emergency stop the tool can be returned to it.

(3) Using an NC LSI like the KM3701 (chapter 6) ample time will be available on the powerful microcomputer to justify the inclusion of subroutines that calculate and print out details of the machining time, total time, tool consumption and total cost at the completion of each component. It is expected that data on machine hour rate, tooling and power would be incorporated in the general control file.

(4) It is necessary to increase research into more efficient and economic use of high-level languages for microcomputer controlled NC machine systems.

(5) It will be beneficial also to design systems employing more powerful direct drive electric stepping motors for NC. These will take advantage of the increasing power of microcomputers and NC LSI's now appearing in the market to offer very economic microcomputer controlled NC machine systems. (The current popularity of the Bridgeport CNC series I and II (P. 46) is a clear indication of the bright future for stepping motor NC).

ACKNOWLEDGEMENTS

The author wishes to thank N.B. Williams for his supervision of this study and for the pains he took to read through the manuscripts. Thanks are also due to the Head of Department of Production Technology and Production Management, Professor R.H. Thornley, for providing laboratory facilities and everybody in the department who helped in one way or the other, especially Mr Eric Langton who advised the author on the electronics.

The help of all those friends in the University of Aston who made the efforts put in this study fruitful is greatly appreciated. The author is indebted to the Vice-Chancellor of the Universigy of Nigeria for providing the sponsorship for this work.

The authors wife, little daughter and family also merit mention for the patience they showed during the difficult times encountered during the course of this study.

Finally the author expresses his gratitudes to Mrs Maureen Creighton for her efficient typing of the manuscripts.

REFERENCES

1. Tal and Persson, E.K. 'Microprocessor - controlled incremental motion servo system'. Proc. of the Annual Symposium: Incremental Motion Control Systems and Devices. Department of Electrical Engineering, University of Illinois, U.S.A 1977.

2. LSI's for Numerical Control: KM3701, KM3702. Technical note from Toko Inc. W. Germany 1980.

3. 'CY500, stored program stepper Motor Controller' preliminary manual from Cybernetic Microsystems, U.S.A 1980.

4. 'Byte' magazine. February 1980. 'A stored program stepping motor controller'.

5. Dirk, J. 'Stepping motor control by angle-sensor monitoring in an automatic map display'. The University of Illinois Symposium, 1978.

6. Maginot, J. and Oliver, W. 'Step Motor Drive Circuitry and Open Loop Control'. The University of Illinois Symposium, ref (59), 1974.

7. Plas, J and Blommaert, J. 'A Stepping Motor Drive Assembly especially designed for CNC systems'. 13th Machine Tool Design and Research (MTDR) Conference University of Birmingham 1972.

8. Rahmen, M.F. 'An experimental study of the performance of V.R. Stepping Motors' Ph.D Thesis. U.M.I.S.T., 1978.

9. Page, C. 'How to put a micro in you product'. Production Engineer, October 1980.

10. Goldsbrough, P.F. 'Microcomputer interfacing with 8255 PPI'. Howard & Sons Inc. Indiana, 1977.

11. 'Micro NOVA's Get Redesigned'. Microcomputer and Systems. International Electronics. March 1979.

12. Ferranti MK II and MK IV. British Numerical Control Society (BNCS). October 1979.

13. 'Are Retrofits the answer to 'obsolete' NC?' Production Engineer April 1979.

14. Tipton, H & Roberts, J.I. 'The M.T.I.R.A. CNC system'. 14th MTDR U.M.I.S.T. September 1973.

15. Carter, W.A. 'Computer Numerical Control Software' 14th MTDR U.M.I.S.T. September 1973.

16. Maclean, R.S., Bruce, J.W. Davies, B. 'Modular Concept in CNC' 13th M.T.D.R. Birmingham, 1972.

17. Crossley, T.R. and McCartney, D. 'A decade of NC'. Annals of CIRP, Vol. 27/1/78.

18. Shumsheruddin, A.A. 'Computer controlled profile jig grinding' 17th M.T.D.R. Birmingham September 1976.

19. The Electrical Research Association (ERA). 'The Engineering of microprocessor systems'. Pergamon Press, 1979.

20. Popa, F. 'The design and implementation of a multi-computer numerical control' Ph.D Thesis. U.M.I.S.T. 1979.

21. Dixon, K.B. 'Microprocessors and the F100-L' Chips with Everything: Proc. of the Interlab Symposium. London, June 1977.

22. Greenfield, S.E. 'The Architecture of microcomputers' Publ. Winthrop computer systems. Massachusetts, 1980.

23. Prasaad, A. 'The design and development of a microprocessor based controller of stepping motor drives' Ph.D Thesis. U.M.I.S.T. 1979.

24. Wright, J. 'Micros: firming up the software' Machinery and Production. 13th June 1979.

25. Teschler, L. 'Coming: new generations of microcomputers'. Machine design. March 22, 1979, Page 108.

26. Colin, A.J.T., 'Implementation of high level programs' The microprocessor and its applications. Edited by D. Aspinal. Cambridge Univ. Press. 1978.

27. Posa, J.G. 'Programming in high level languages'. International Electronics. January 18, 1979.

28. Hicks, M.S. 'Forth - dictionary-like organised high level language for micros'. International Electronics. March 15, 1979.

29. Caudil, P. 'High level programming with assembly included'. International Electronics. February 1, 1979.

30. Mattos, P.G. 'Compiling a real-time high-level language for microprocessors'. Conference on μ P's in automation and communications. University of Kent at Canterbury September 1978.

31. 'Micro-planning program' BNCS News. December 1979.

32. Bottari, W. 'How to design single chip microcomputer into systems' Control Engineering. May 1979.

33. The Production Engineer. P. 27 September 1980.

34. Brussels, H.V. Simons, J and Peters, J. 'Microprocessor in Hierarchical control system'. Annals of the CIRP Vol. 27/1/78.

35. Stute, G. and Worm, H. 'Principle and example of a modular multiprocessor NC system' 19th M.T.D.R. U.M.I.S.T. September 1978.

36. Becker, H. 'Development and present level of numerical control'. 19th M.T.D.R. U.M.I.S.T. 1978.

37. 'Chipping a way towards total control capability'. Metal Working Production. April 1979.

38. Savage, R. 'The influence of microprocessors on the evolution of numerical control'. 16th M.T.D.R. U.M.I.S.T. 1975.

39. Hinduja, S., Satine L, Vale, G., Boon, J. 'Interactive programming systems for NC lathes'. BNCS-80: Towards integrated manufacturing. Fourth Annual Technical Conference of the BNCS. Hotel Metropole Birmingham. April 1980.

40. Numerical Engineering. P. 27. April 1980.
'Going back to operator control?'

41. Tipton, H. 'Microprocessor - based systems with shop floor data input' BNCS-80: Towards integrated manufacturing conference, Birmingham. April 1980.

42. Siemens, Sinumerik Sprint 8T, 8M, 6T & 6M. Technical data manuals: Issues 132 - 349 - 10793.

43. Fredriksen, T.R. 'Direct digital processors control of stepping motors'. IBM journal of research and development. Vol. II 1967.

44. Loss, A.M and Frisch, J. 'Closed loop control of a DNC milling machine'. 15th. MTDR. Birmingham September, 1974.

45. Akgerman, N. 'BCLCNC - A computer numerical control system'. 15th MTDR. September 1974, Birmingham.

46. Andrew, J.P. 'The stepping motor as a machine tool actuator'. Electronics plus power. Journal of IEE February 1972.

47. Cavey, T. 'Simple digital control of mechanical movement - with stepping motors'. Control and instrumentation: March 1979.

48. Weston, R.H. and Charles G.P. 'A microprocessor controller to provide low cost numerical control'. 19th MTDR. UMIST. 1978.

49. Latham, V. and Pelc, S.F. 'The use of microprocessors in glass grinding'. Conference on microprocessors in automation and communication, University of Kent at Canterbury. 19th February 1978.

50. Roberts, A.M. 'An open loop CNC lathe with optimised turning parameters'. 19th MTDR. 1978, UMIST.

51. Wells, B.H. 'Microprocessor control of step motors'. Proc. of the 5th annual symposium on 'Incremental motion control systems and devices. University of Illinois. May 1976.

52. Al-Anbuky, A.H. and Swadi, G.A. 'Microcomputer control of a general purpose NC drill for printed circuit boards. 20th MTDR. Birmingham, September 1979.

53. Weck, M. Gogrewe, U. and Ernst, D. 'Numerical controlled dressing of profile grinding wheels' MTDR, September 1979, Birmingham.

54. Lafreniere B.C. 'Interactive microprocessor-controlled step motor acceleration optimisation'. In cremental motion control systems and devices symposium. Illinois, 1978.

55. Crossley, T.R. and Unsworth, R.E. 'Design of a microprocessor-based draughting system'. Annals of the CIRP Vol. 29/1/80.

56. Bridgeport series I CNC. Technical manual no. DEK 3M/4/79.

57. 'Bridgeport Millers: The finest money can buy'. Numerical Engineering. October 1980.

58. Theory and applications of step motors by B.C. Kuo. West Publ. Co. New York, 1974.

59. Proceedings of the Annual Symposium, incremental motion control systems and devices. Department of Electrical Engineering, University of Illinois, U.S.A, 1973, 1974, 1975, 1976.

60. International Conference on stepping motors and systems. Department of Electrical and Electronic Engineering, University of Leeds, England, 1974, 1975, 1976.

61. Russel, A.P. and Pickup, I.E.D. 'Subharmonic resonance in stepping motors. Proc. IEE. Vol. 126, No. 2, February 1979.

62. Russel, A.P. and Pickup I.E.D. 'Development of a stabilisation scheme for the suppression of a parametric instability in stepping motors'. Proc. IEE. Vol. 126, Nol. 4, April 1979.

63. Pawletko, J.P. 'Advances in step motor controls. Leeds Conference (ref. 60) 1976.

64. Leehouts, A.C. 'Techniques for microstepping control of step motors'. Control Engineering. March 1975.

65. Hughes, A. Acarnley, P.P. Lawrenson, P.J. 'Effects of operating mode on torque - speed characteristics of V.R. motors'. Leeds Conference (ref. 60) 1976.

66. Beling, T.E. 'Some effects of drive systems on stepping motor resonance'. Leeds Conference (ref. 60) 1976.

67. Justice, M. 'Simulation and control of a stepping motor'. Ph.D Thesis. University of Aston in Birmingham, 1979.

68. Kordik, K.S. 'The step motor - what it is and does'. Illinois Symposium (ref. 59) 1974.

69. Kuo, B.C. and Singh, G. 'Damping methods for step motors'. Illinois symposium (ref. 59) 1973.

70. Kuo, B.C. and Singh, G. 'A computer design and case study of a step motor printer system (ref. 59), 1974.

71. Lawrenson, P.J. Hughes A and Acarnley, P.P. 'Improvement and prediction of open-loop starting/stopping rates of stepping motors'. Proc. of IEE. Vol. 124. 1977.

72. Acarley, P.P. 'Analysis and improvement of the steady state performance of variable - reluctance stepping motors'. Ph.D Thesis. University of Leeds. 1977.

73. Jufer, M. 'Self-synchronisation of stepping motors'. Leeds Conference (ref. 60), 1976.

74. Frus, J.R. and Kuo, B.C. 'A closed loop control of stepping motors using wave form detection'. Leeds Conference, (ref. 60) 1976.

75. Bell, R., Lowth, A.C., Shelley, R.B. 'The use of stepping motors in numerically controlled machine tools - a summary of the current state of development'. Int. journal of machine tool design and research. Vol. 10, 1970.

76. Bell, R., Lowth, A.C., Shelley, R.B. 'The application of stepping motors to machine tools'. Machinery Publ. Co. Brighton 1970.

77. Jackson, J.F. and Bell, R. 'An electrohydraulic stepping motor for NC machine tools'. Advances in machine tool design and research, Pergamon Press, 1971.

78. Bajwa, M. 'Open loop control by using electrohydraulic motors'. Illinois symposium (ref. 59) 1973.

79. Kordik, K.S. and Senica, K.M. 'Step motor selection'. Illinois Conference (ref. 59) 1974.

80. Superior Electric 'Design engineers guide to DC stepping motors'. Manual no. SE-L 8753. U.S.A.

81. Stevens, H.G. 'Matching stepping motor characteristics to the mechanical load'. Control and instrumentatiion. March 1979.

82. 'Motor selection' - motors and motor control. Machine design. May 17, 1979.

83. Forth, L. and Kidd, P.A. 'Peripheral interfacing using front-end microprocessors'. Conference on microprocessors in automation and communications. University of Kent, September 1978.

84. Achi, P.B.U. 'Design of stepping motor controls for HPE numerically controlled machine tool'. MSc dissertation 1978. University of Aston in Birmingham.

85. Parnaby, J. 'Instability due to static and coulomb friction in autonomous control systems'. Proc. of 'The British hydromechanics research association, Cranfield, Bedford, 1971.

86. de Barros, P.M.R. 'The design and performance of a modular CNC system'. Ph.D Thesis. UMIST 1976.

87. Richards, R.K. 'Electronic and digital systems'. P.467. Publ. John Wiley, 1966.

88. Shumsheruddin, A.A. 'A new generation of continuous path NC systems'. Proc. of 9th MTDR, September 1968, Birmingham.

89. Bresenham, J.E. 'Algorithm for computer control of a digital plotter'. IBM systems journal, Vol. 4, no. 1, 1965.

90. 'MTIRA CNC system software Part III' by MTIRA, Macclesfield. (Described in ref. (20)).

91. Bergren, C. 'A simple algorithm for circular interpolation'. Control Engineering Vol. 18, no. 9, September 1971.

92. Ertongur, N.A. 'Investigation into the instability in an electrohydraulic control system for machine tools'. M.Sc Thesis (submitted under Ph.D regulations), University of Birmingham, 1966.

93. Data General Ltd. 'Micro NOVA packaged systems' no. 021-781-1.

94. Data General Ltd. MN 601: micro NOVA manuals, 1976.

95. Young, R. and Brignel, J.E. 'The case for a simple, fast programmable interface converter'. Proc. of the Conference on programmable instruments, National Physical Laboratory, Middlesex, November 1977.

96. Middleditch, A.E. 'The design of a digital computer contouring control system'. Ph.D Thesis. Carnegie Institute of Technology and Mellon Institute of Science, Pittsburgh, Pa. U.S.A 1972.

97. Martin, J.D. and Dalzel, D.T. 'An Economic Intelligent plotter'. Proc. of the Conference on Programmable Iestruments. National Physical Laboratory, Middlesex November 1977.

98. Lo, H.Y. and Lu, J.H. 'A simple design for a digital programmable frequency multiplier'. Int. Jnrl. of Electronics 1979. Vol. 46, No. 5.

100. Fisher, E. and Jensen, C.W. 'PET and the IEEE488 bus (GPIB)'. Publ. Osborne/McGraw-Hill Corp. California. P.37-43,(1980).

APPENDIX A

Circular Interpolation Algorithm Used by Crossley and Unsworth

The general cartesion equation of a conic is given by

$$f(x, y) = Ax^2 + 2Hxy + By^2 + 2Fx + Gy + C = 0 \qquad A.1$$

For circular interpolation, suppose the starting point is $(x = x_o, y = y_o)$ and the increments in X and Y axes are $\Delta x, \Delta y = 0$ or ± 1. then after m moves the new positions are

$$x = x_o + x_m \qquad A.2$$

$$y = y_o + y_m \qquad A.3$$

Equation A.1 becomes

$$A (x + x_m)^2 + 2H(x_o + x_m) (y_o + y_m) + B(y_o + y_m)^2 +$$

$$2F (x_o + x_m) + 2G (y_o + y_m) = \Delta m \qquad A.4$$

Where Δm is the associated error in the function f (x, y) after the move.

In the next move the function becomes

$$A (x_o + x_m + \Delta x)^2 + 2H(x_o + x_m + \Delta x) (y_o + y_m + \Delta y) + B(y_o + y_m + \Delta y)^2$$

$$+ 2F (x_o + x_m + \Delta x) + 2G (y_o + y_m + \Delta y) = \Delta_{m+\Delta} \qquad A.5$$

Subtracting 3 from 4 yields

$$2 (Ax_o + Ax_m + Hy_o + Hy_m + 2F) (\Delta x) + 2(by_o + By_m + Hx_o + Hx_m + 2G) (\Delta y)$$

$$+ A(\Delta x)^2 + B(\Delta y)^2 + 2H \times \Delta y = \Delta_{m+\Delta} = \text{error} \qquad A.6$$

$$\text{i.e. } Rx\ \Delta x + Ry\ \Delta y + A(\Delta x)^2 + B(\Delta y)^2 + 2H\ x\ y = \Delta = \text{Error} \qquad A.7$$

Where $Rx = 2 (Ax_o + Ax_m + Hy_o + Hy_m + F)$

$$Ry = 2\ (By_o + By_m + Hx_o + Hx_m + G)$$

The S/W algorithm tends to reduce the error (Δ) by searching the three relevant motions (within a quandrant) out of eight possible directions (fig. A.1) to determine Δx, Δy. After each move the values of Rx and Ry are updated and the interpolation continues until the circular arc is complete.

APPENDIX B

The BERGREN Equation

The Bergren formular determines the co-ordinate increments (fig. B.1)

$$\Delta x_i = x_{i+1} - x_i \text{ and } \Delta y_i = y_{i+1} - y_i \qquad \text{B.1}$$

$$\begin{aligned} x_{i+1} &= x_i \cos\theta - y_i \sin\theta \\ y_{i+1} &= x_i \sin\theta + y_i \cos\theta \end{aligned} \qquad \text{B.2}$$

For small values of interpolation angle, θ,

$$\begin{aligned} \sin\theta &\simeq \theta \quad \text{(radians)} \\ \cos\theta &\simeq 1 - \theta^2/2 \end{aligned} \qquad \text{B.3}$$

From B.2 and B.3

$$\begin{aligned} \Delta x_i &= =x_1(\theta^2/2) - y_1\theta \\ \Delta y_i &= x_i\,\theta - y_i(\theta^2/2) \end{aligned} \qquad \text{B.4}$$

The multiplications in B.4 can be transformed into normal μC shift operations by assigning inverse powers of 2 to the values of θ:

$$\theta = \tfrac{1}{2}^m \qquad \text{B.5}$$

where m = no. of μC right shift. In fig. B.2, the circular arcs defined by θ are approximated by linear segments with imposed chordal error, E_c. In fig. B.2, it is geometrically true that

$$\cos\theta/2 \simeq (R - E_c)/R \qquad \text{B.6}$$

Where R = radius of the circle.

Substituting eqn. B.3 and B.5 into eqn. B.6

$$E_c \simeq (R\theta^2/8) = R/(2^{2m+3}) \qquad B.7$$

Therefore the value of m can be determined from a chosen chordal error. The μC shift the value in B.7 right until the inequality holds and then the value of m is applied in eqn. B.5 and used to calculate the new co-ordinates in eqn. B.4.

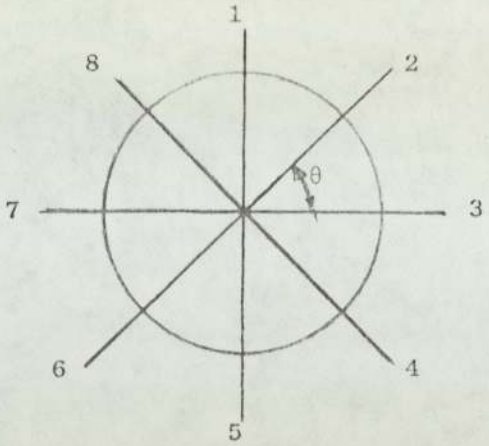


Fig A.1 Circular Interpolation move directions for $\theta = \pi/4$

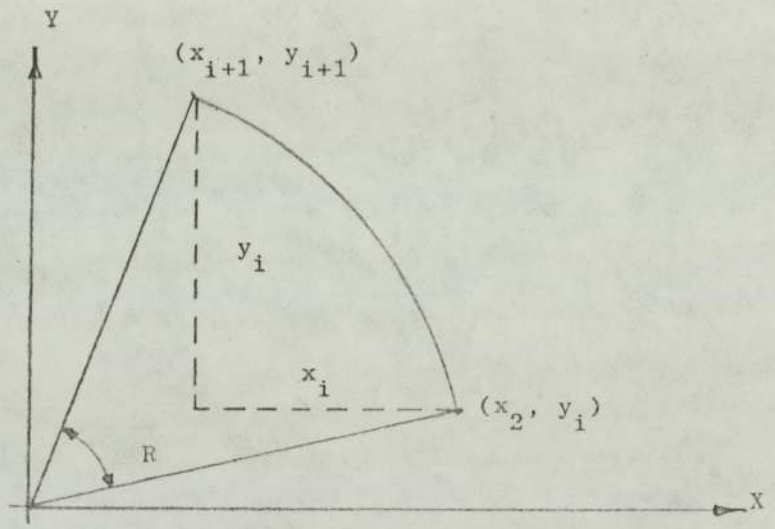


Fig B.1 Geometry for Bergren Algorithm

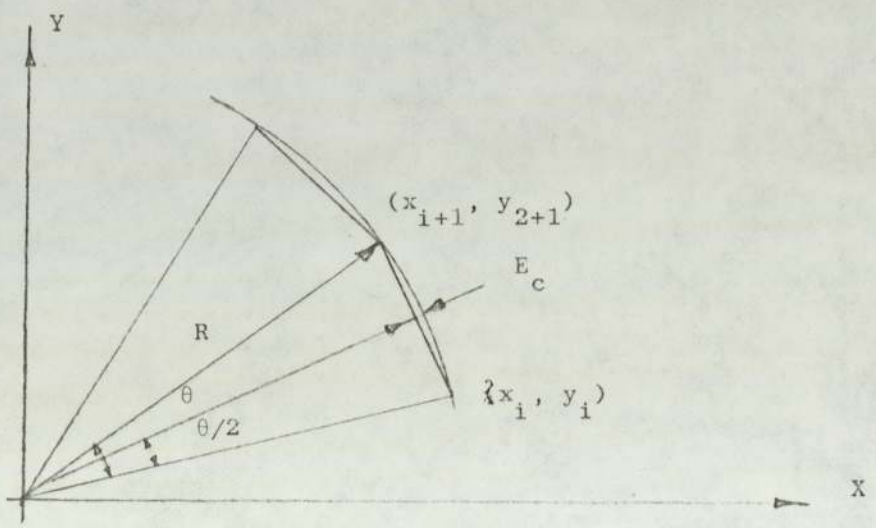


FIG B2 Arcs approximated with segments defined by a chordal error

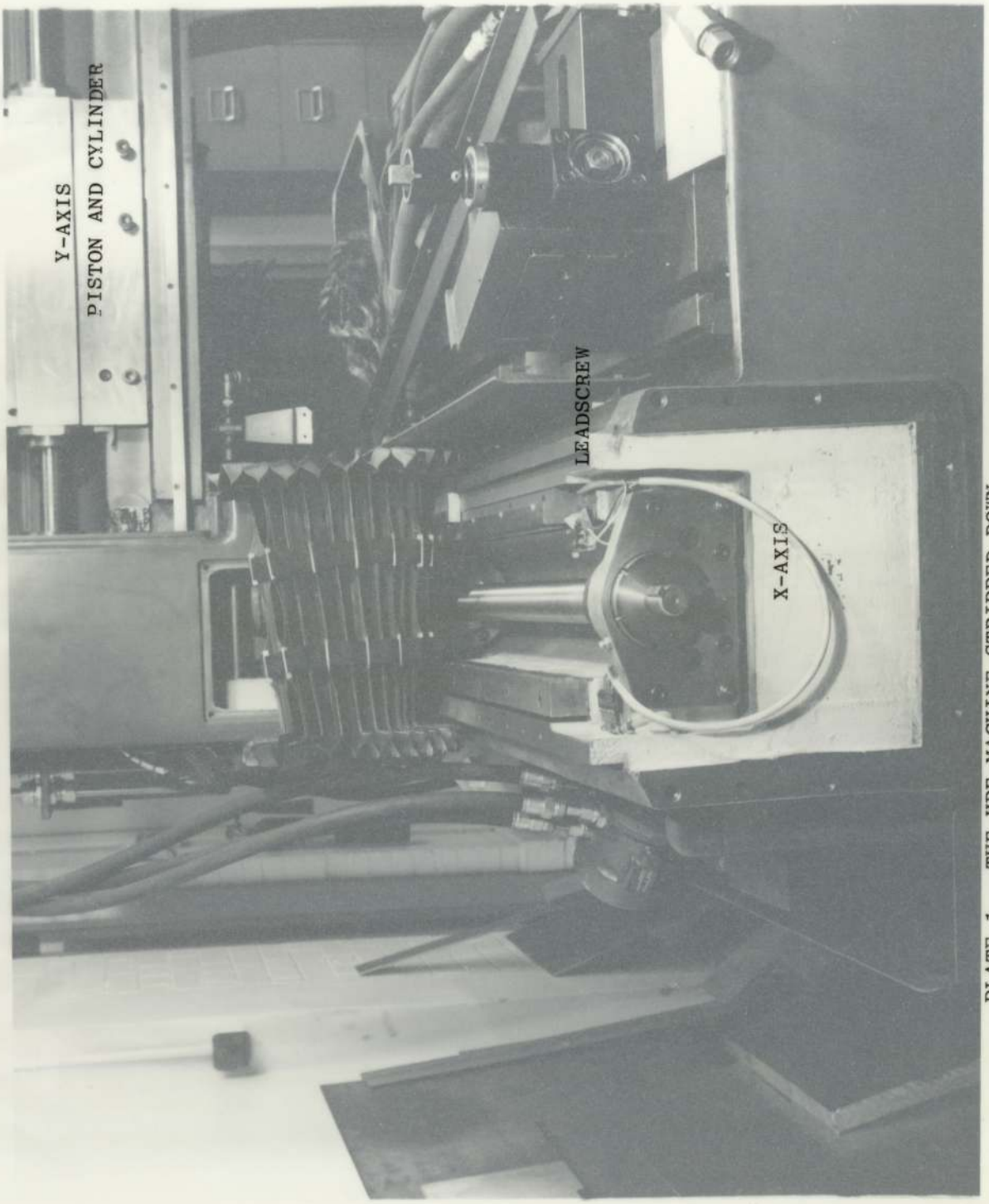
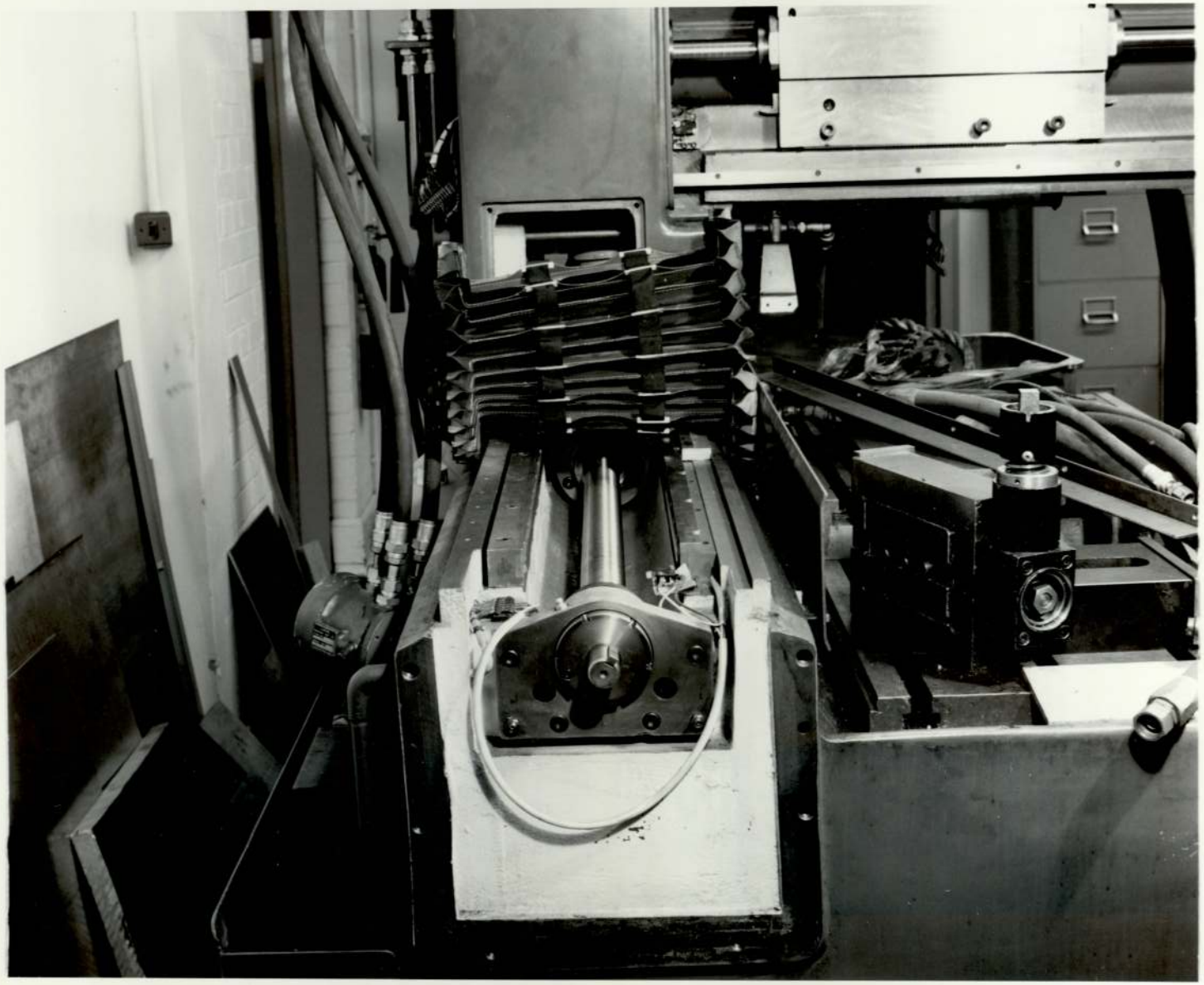


PLATE 1 THE HPE MACHINE STRIPPED DOWN



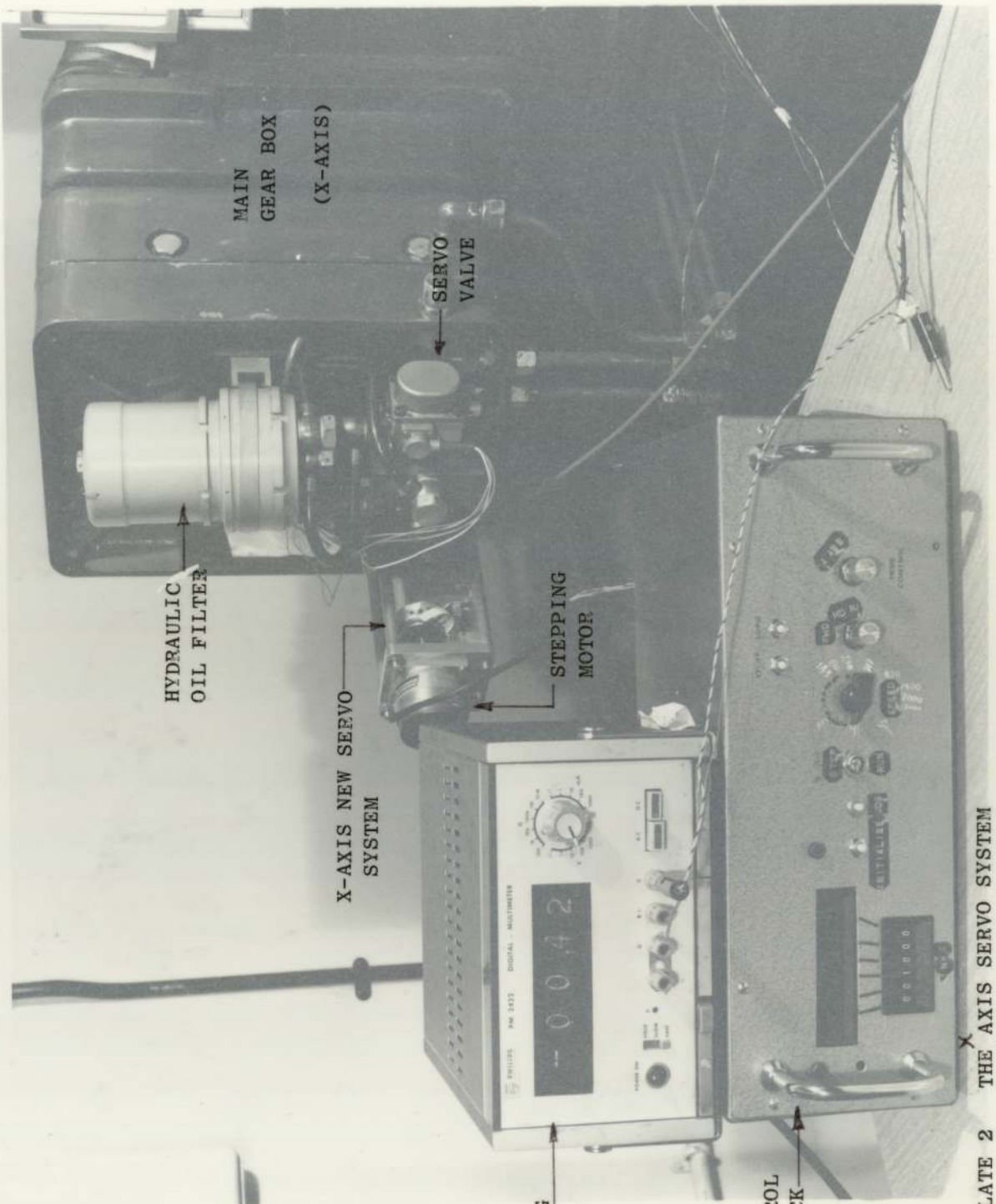


PLATE 2 THE X-AXIS SERVO SYSTEM


PHILIPS PM 2422 DIGITAL MULTIMETER
0042
INITIALISE
JOG
RUN
STEP
SPEED
FWD
REV
MODE CONTROL

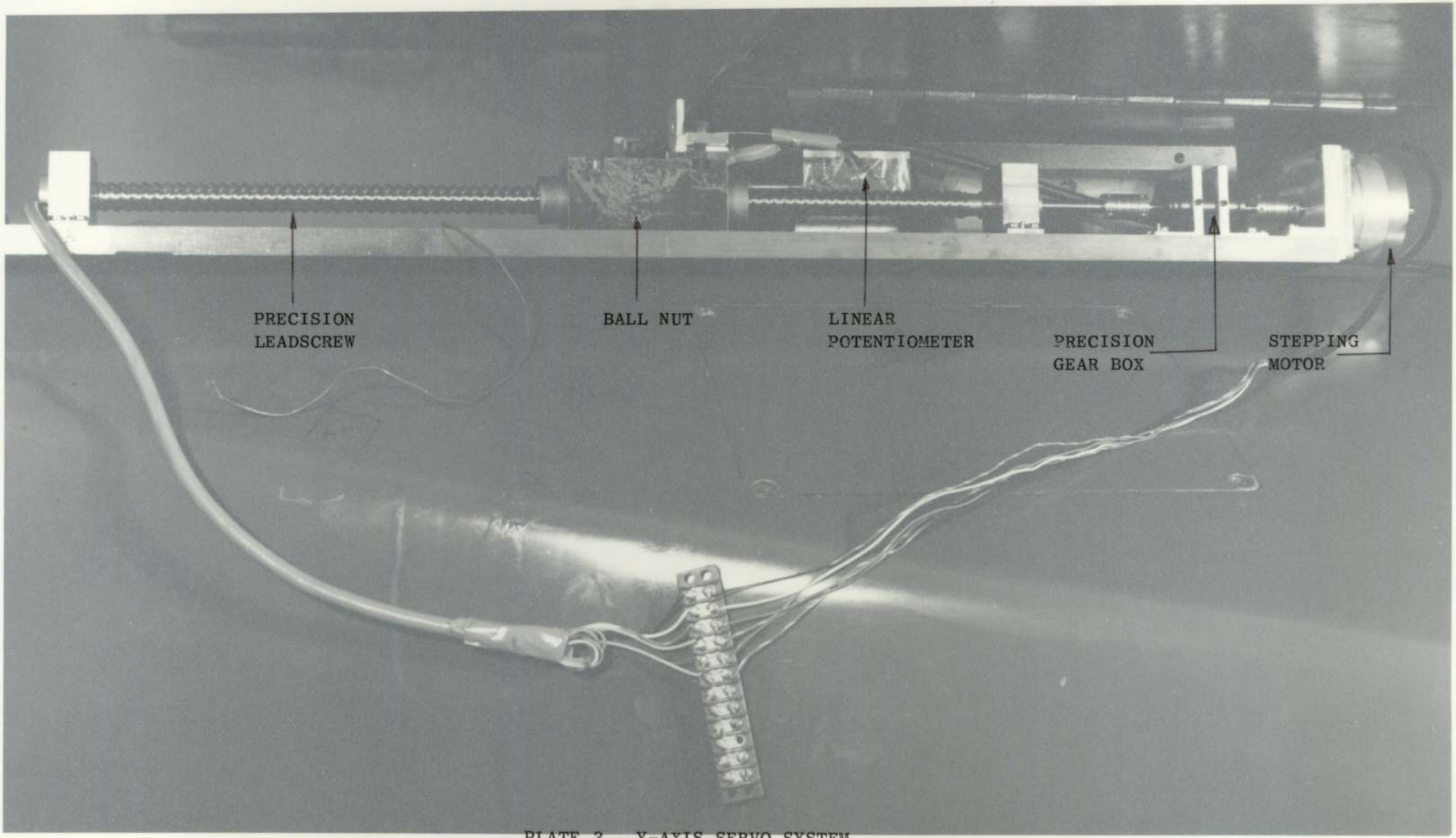
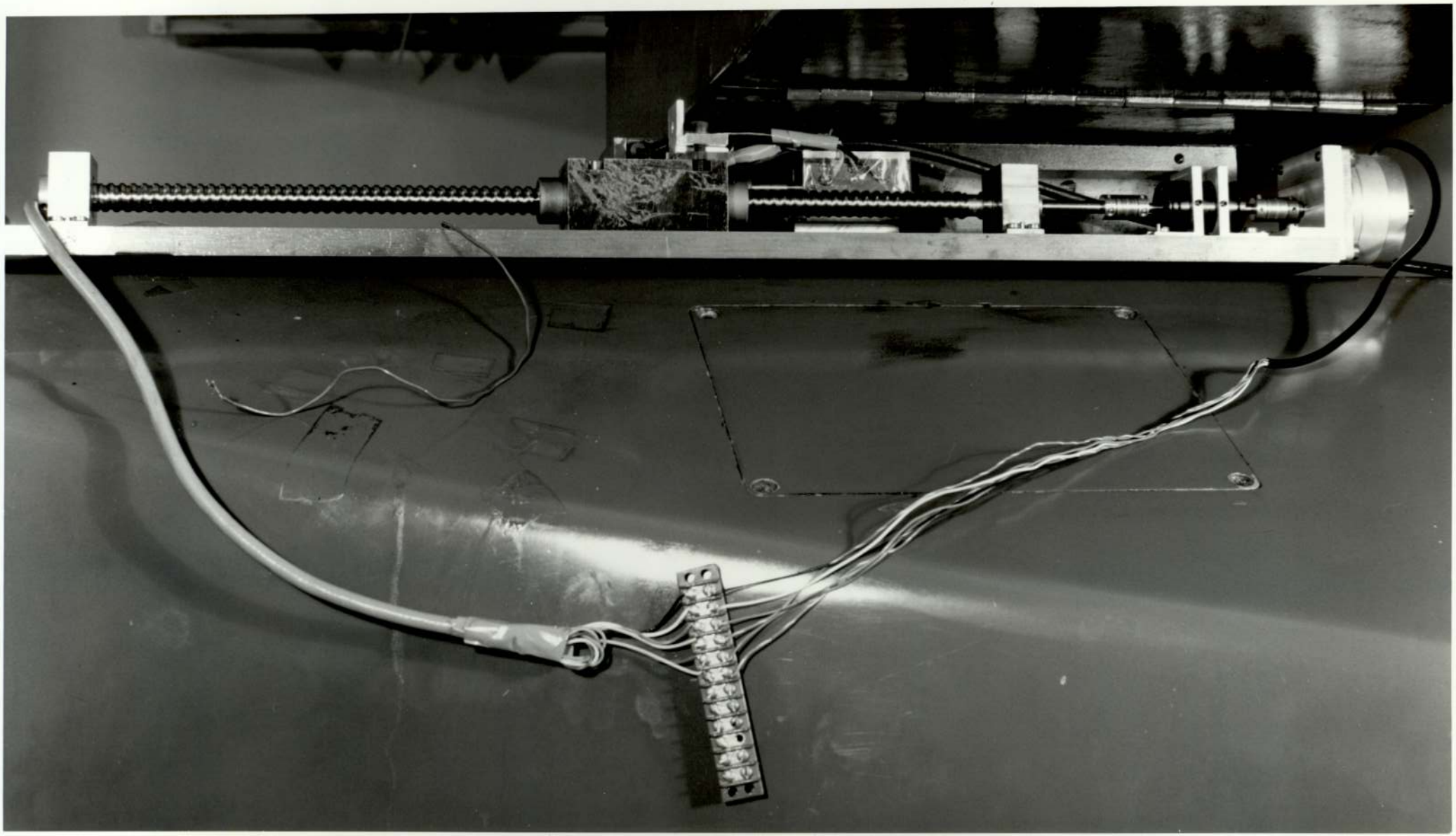


PLATE 3 Y-AXIS SERVO SYSTEM



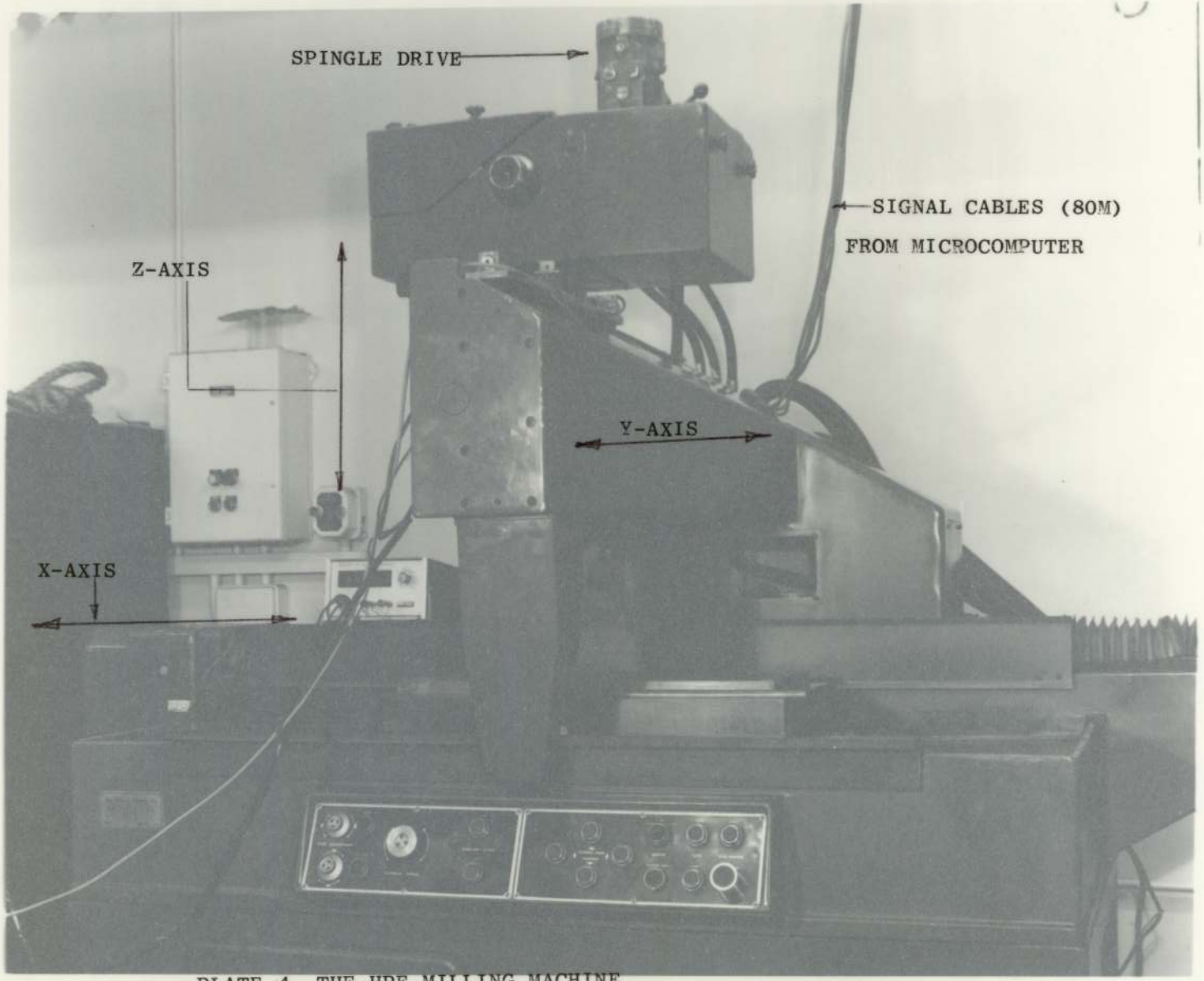
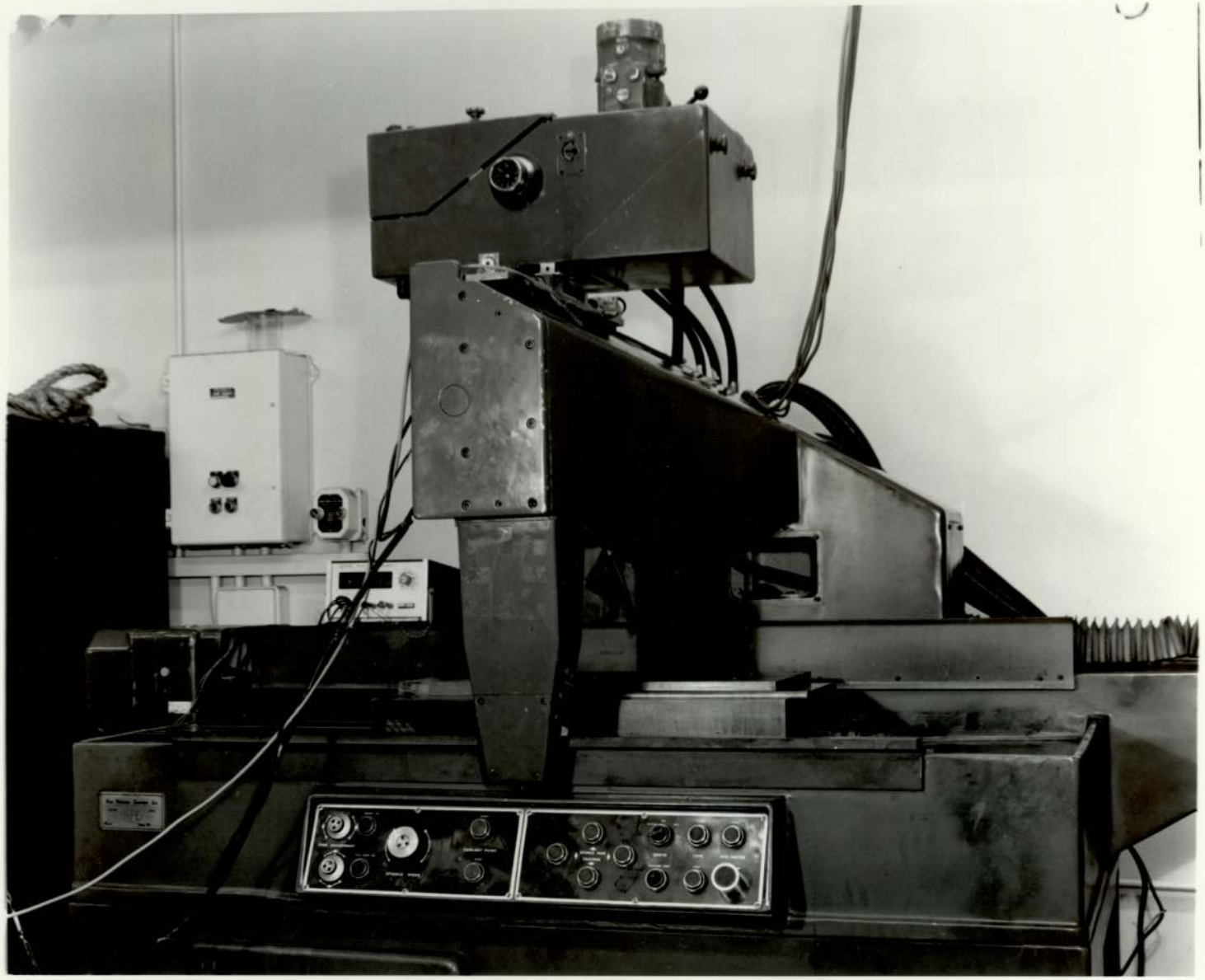


PLATE 4 THE HPE MILLING MACHINE



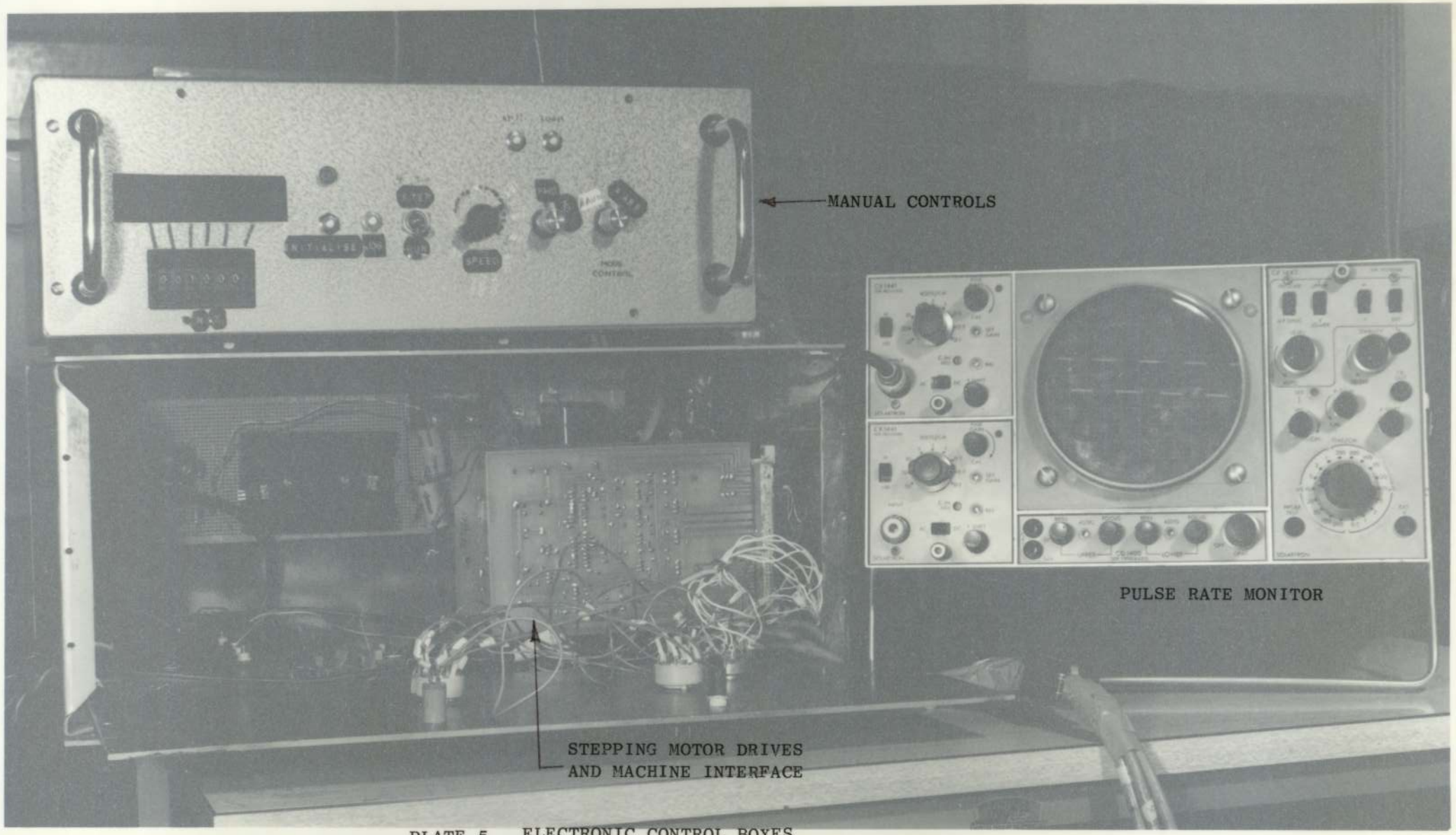
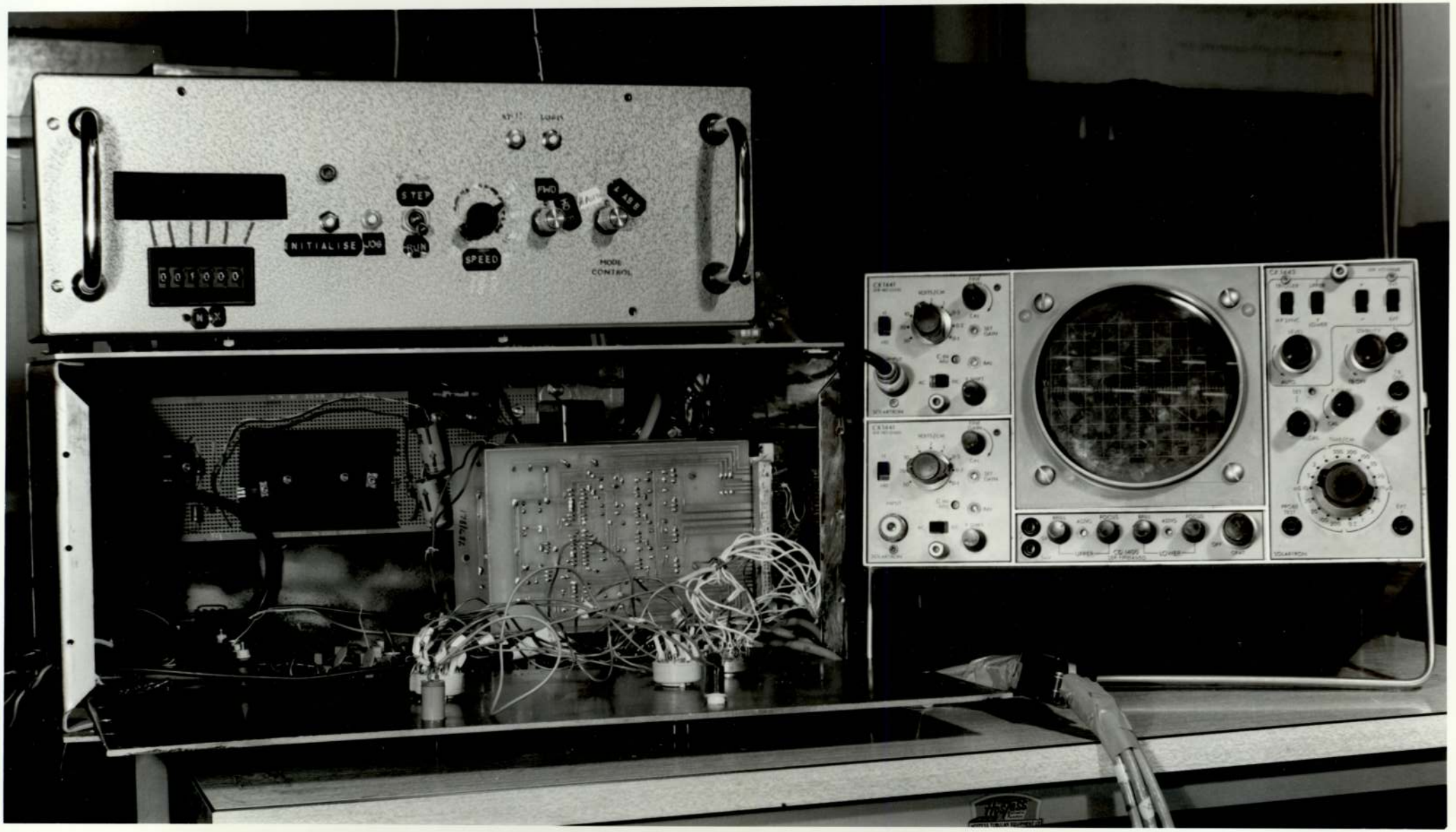


PLATE 5 ELECTRONIC CONTROL BOXES


INITIALISE
STEP
RUN
SPEED
MODE CONTROL

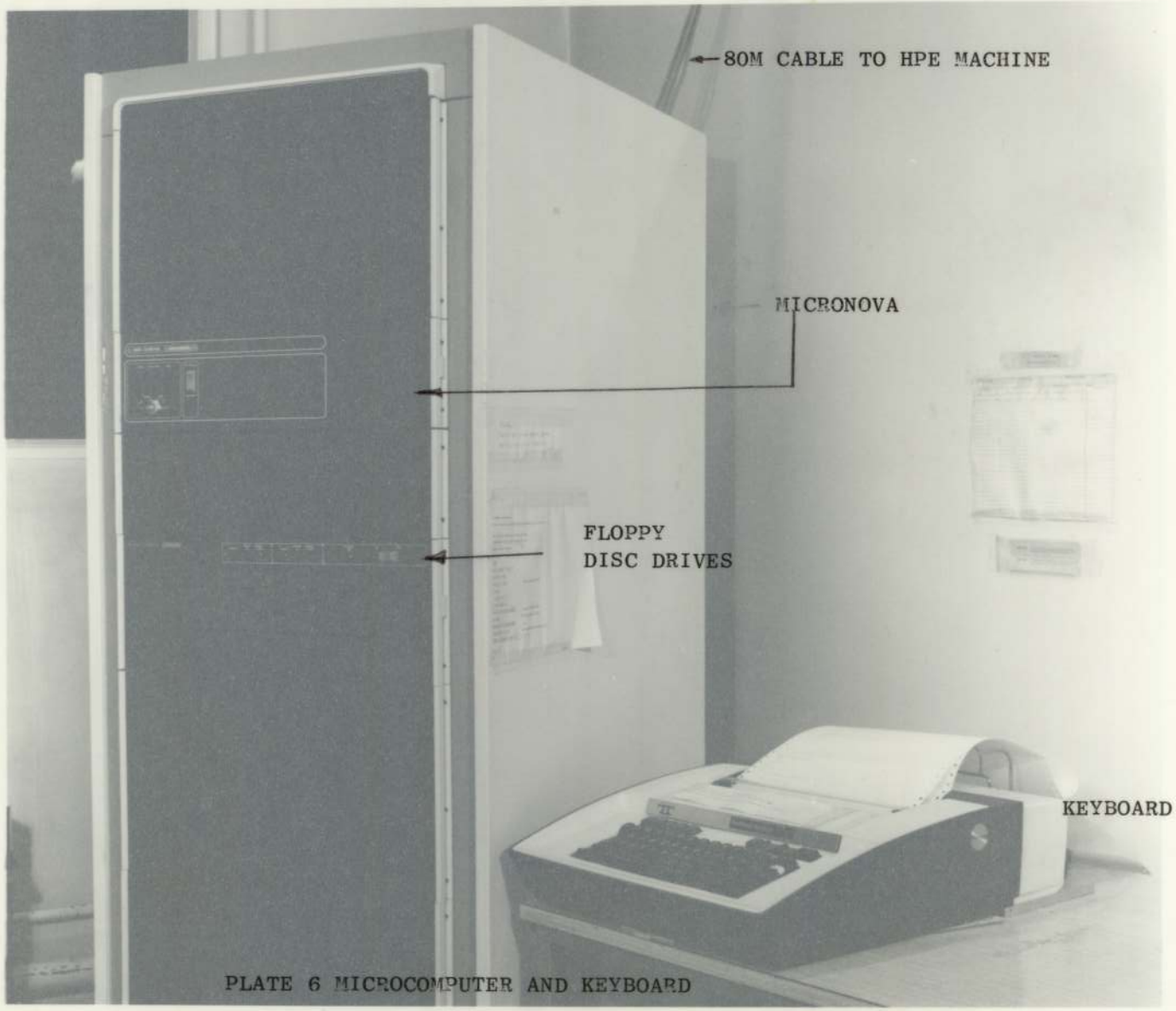


PLATE 6 MICROCOMPUTER AND KEYBOARD