INTERFACING AN ENGINE LATHE AND A MICROCOMPUTER

by

BRADLEY A. KRAMER

B.S., Kansas State University, 1980

A MASTER'S THESIS

submitted in partial fulfillment of the
requirements for the degree

MASTER OF SCIENCE

Department of Industrial Engineering

KANSAS STATE UNIVERSITY

Manhattan, Kansas

1981

Approved by:

[illegible]

Major Professor

SPEC
COLL
LD
2668
.T4
1981
K72
c.2

TABLE OF CONTENTS

		Page
LIST OF FIGURES		ii
LIST OF TABLES		iii
ACKNOWLEDGEMENT		iv
CHAPTER 1.	INTRODUCTION	2
1.1	HISTORY OF THE LATHE	2
1.2	HISTORY OF NUMERICAL CONTROL	6
1.3	PROBLEM STATEMENT	9
1.4	PURPOSE	9
1.5	METHOD	9
1.6	EQUIPMENT	10
CHAPTER 2.	DESIGN	13
2.1	GENERAL CONCEPT OF OPERATION	13
2.2	MACHINABILITY DATA	13
2.3	BASIC OPERATION OF THE MMD-1	15
2.4	DECODER CIRCUIT	16
2.5	COUNTER CIRCUIT	22
2.6	INPUT CIRCUIT	30
2.7	MOTOR DRIVE CIRCUIT	34
2.8	SUMMARY OF THE ELECTRONIC CIRCUITRY	39
CHAPTER 3.	CONSTRUCTION	43
CHAPTER 4.	TESTING	46
CHAPTER 5.	PROGRAMMING	49
CHAPTER 6.	RESULTS	62
CHAPTER 7.	FURTHER RESEARCH	64
REFERENCES		65
APPENDICES		
A.	X-MOV SUBROUTINE	68
B.	Z-MOV SUBROUTINE	71
C.	MOVE-CHECK SUBROUTINE	74
D.	SEMICIRCLE SUBROUTINE	77
E.	MAIN PROGRAM	85

LIST OF FIGURES

Figure		Page
1.	Essential Components of a Lathe	3
2.	General Concept of Operation	14
3.	Decoder Circuit	17
4.	Counter Circuit	23
5.	Input Circuit	31
6.	AC Motor Drive Relay Circuit	36
7.	AC Motor Drive Circuit	38
8.	Test Part	47
9.	X-Mov Subroutine Flow Chart	51
10.	Move-Check Subroutine Flow Chart	52
11.	Semicircle Subroutine Flow Chart	55
12.	Generalized Program Flow Chart	60

LIST OF TABLES

Table		Page
1.	Machinability Data for Acrylic Plastic	13
2.	Logic Table for the 74154 IC Chip	19
3.	Logic Table for a 74L04 IC Chip	20
4.	Logic Table for an AND Gate (7408)	22
5.	Logic Table for a 7490 Decade Counter	25
6.	Logic Table for Two Cascaded 7490's	27
7.	Logic Table for an 8212 IC Chip	32
8.	Logic Table for a 7474 IC Chip	39
9.	Microcomputer Codes for Device Selection and Operation	40
10.	BCD Numeral Conversion	53
11.	X and Z Moves for a Semicircular Arc	57

ACKNOWLEDGEMENT

The author expresses his gratitude to his wife for her patience and loving support over the past year. He also wishes to thank his parents and family for their support and encouragement. The help and experienced insight of Mr. Bruce P. Mignano was also much appreciated. In addition, the author is indebted to Professor John E. Biegel for his guidance, suggestions, and good humor throughout the course of this research.

THIS BOOK
CONTAINS
NUMEROUS PAGES
THAT WERE
BOUND WITHOUT
PAGE NUMBERS.

THIS IS AS
RECEIVED FROM
CUSTOMER.

CHAPTER 1

INTRODUCTION

INTRODUCTION

1.1 HISTORY OF THE LATHE

"It is conceded that of all the machines employed by the mechanic to aid him in his work, the lathe holds the honor of having been the first machine tool. From it, in one way or another, all other machine tools have been developed."*

This quote from Oscar E. Perrigo's book, Lathe Design, sums up the importance of the lathe to the development of all machine tools (2).

A lathe is a machine which rotates a workpiece around a central axis against a fixed tool. The essential components of a lathe are the headstock, tailstock, carriage, and frame. Figure 1 shows these components in block form. The headstock is the device which transmits rotation to the workpiece. The tailstock is used to support the end of the workpiece opposite to the headstock. The carriage or cross-slide is the mechanism which supports and moves the cutting tool. It can travel along the Z axis (between the headstock and the tailstock) and it supports the cross-feed which can travel along the X axis (perpendicular to the Z axis or moving into or away from the workpiece). The carriage may also support a compound rest. The compound rest is similar to the cross-feed mechanism except that it has a much shorter length of movement. It is used for making angular cuts and short tapers. The frame is the part of the machine which holds and supports the rest of the lathe. It consists of the bed and legs. The legs support the bed and the bed supports the tool and the work. The bed also absorbs the forces associated with the cutting of the workpiece. These are

* Oscar E. Perrigo, Lathe Design, The Norman W. Henry Publishing Co., New York, NY, 1917

THIS BOOK CONTAINS NUMEROUS PAGES WITH DIAGRAMS THAT ARE CROOKED COMPARED TO THE REST OF THE INFORMATION ON THE PAGE.

THIS IS AS RECEIVED FROM CUSTOMER.

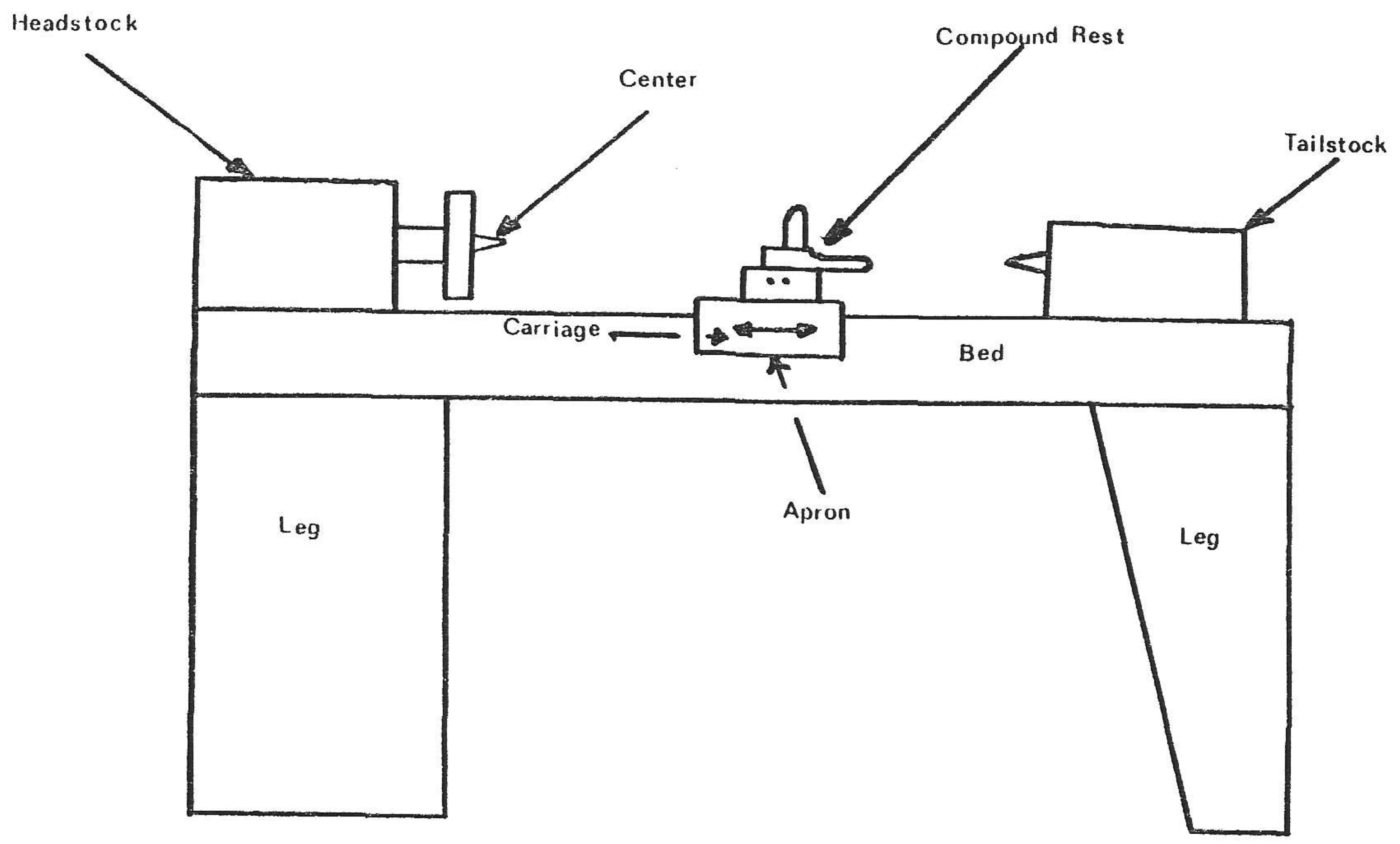


Fig. 1. Essential Components of a Lathe

THIS BOOK CONTAINS NUMEROUS PAGES WITH THE ORIGINAL PRINTING BEING SKEWED DIFFERENTLY FROM THE TOP OF THE PAGE TO THE BOTTOM.

THIS IS AS RECEIVED FROM THE CUSTOMER.

the essential components of an engine lathe.

The lathe's long history began about seven hundred years before Christ when the tree lathe was in use. This lathe was very simple in construction and operation. To use the tree lathe, a craftsman would go into a wooded area and find two trees the proper distance apart and then, with his axe, he would fashion two pointed pieces of wood to serve as the centers for the turning of his workpiece. These centers were either lashed securely to the two trees or driven into the sides of these trees. Another piece of wood was secured across ths span of the two trees to serve as a tool rest. The piece to be turned was then placed between the two centers which means that the piece to be turned runs parallel to the tool rest. A rope, connected at one end to a springy bough or sapling which was above the workpiece, was wound a-round the workpiece and then dropped to the ground. The loose end was tied into a loop for the man's foot. This constituted the power source of the tree lathe. As the foot was brought to the ground, the bough was deflected and the workpiece turned. When the workpiece had stopped turning, the foot was released and the bough sprang back into position readying the piece for turning again. This set-up allowed the workman to make a cut only while he was depressing the rope, but the craftsmen of this era became very adept at this practice and could produce a quality product.

The next major form of the lathe was the pole lathe. This lathe was driven in much the same manner as the tree lathe, but was used in-doors. Replacing the springy sapling or bough was a pole which was attached to the ceiling above the machine and worked in the same manner as the bough or the sapling had worked. This pole was called a "lath" and it has been speculated that this is where the term, "lathe", came from. The biggest difference with this lathe, however, was that the basics of a machine had been developed (2). The pole lathe had a head stock which was used for transmitting the turning of the workpiece from a drive pulley that was connected to the lath cord, an adjustable tail-stock to hold the workpiece between it's centers, an adjustable tool rest, and a wooden bed mounted on legs. This basic lathe was in use as early as 757 A.D. (1).

One variation of the spring pole lathe was called the "bow" or "fiddle bow" lathe. The only difference between the fiddle bow lathe and the spring pole lathe was in the drive of the workpiece. A cord was still wound around a driving pulley, but it was attached to a bow which was pushed back and forth to turn the piece instead of depressing the footloop as was done to power the pole lathe. Some of the smaller bow lathes were operable by one man, but most required an assistant to run the bow to power the lathe.

The next major improvement to the lathe was the introduction of a flywheel to the drive mechanism (2). In using the lathe, it had been noticed that when one had a heavy workpiece, the release of the cutting tool before the power stroke was finished resulted in the continued turning of the workpiece during the return stroke. This led to the introduction of the flywheel and thus, the first continuous turning of a workpiece.

The production of a drive pulley with more than one diameter was a small, but significant improvement (2). Now the mechanic could vary the speed at which the workpiece was turned for a more efficient operation.

Perhaps one of the greatest advances was the introduction of the cross-slide or carriage (1). The control of the tool was taken from the hands of the machinist and put into a mechanical device. The cross-slide was in use as early as 1480 A.D. and consisted of the major elements found in the present day cross-slides. It had the cutting tool, a holding screw for the tool, and a special recess for the tool to keep it from twisting or turning. The tool holder was mounted on a mechanism which incorporated a screw connected to a crank which operated the cross-feed. This was fitted on another device which allowed the entire cross-slide to move along the Z axis (the longitudinal axis). Thus the major elements of the lathe were developed by 1500 A.D.

Some of the improvements that needed to be made, however, included the construction of a heavier bed, the improvement of the accuracy of the lead screw threads, and the application of an external, constant source of power. Henry Maudslay was the man called upon to bring all of these components together into one machine.

It was Maudslay who also recognized the need for precision flat surfaces on which to mount the cross-slide, headstock, and tailstock (1). He recognized the need for a precision lead screw and for spindle bearings and tailstock centers to ensure the precision rotation of the workpiece. Maudslay, by 1825, had combined all of these features into an industrial size lathe which was suitable for turning metal (1). This lathe had all of the essential components of a modern lathe.

1.2 HISTORY OF NUMERICAL CONTROL

Numerical control is a system of control which uses numerical codes as directions for driving a machine. The control medium for these codes can range from punched paper tape and recorded tapes or discs to the direct system control that a computer can maintain.

The first proposal for a numerically controlled machine came in 1949 from the Parsons Company (3). Parsons had a need for a machine that could accurately mill the complex forms of the helicopter blades that they manufactured. Their proposal was to utilize computer routines to operate a milling machine on a *point-to-point* system. This proposal was submitted to the U.S. Air Force.

Eventually, the Air Force sent this project to the Massachusetts Institute of Technology (MIT) and a prototype NC milling machine with three-axis control was developed in 1952 (3). The machine utilized punched paper tape as the control medium and operated by sending electrical pulses to move the machine in the desired direction. The first system was driven by a huge vacuum tube computer which actually took more space than the milling machine itself. But as time went on the control units became and are still becoming smaller and smaller.

This system was widely acclaimed when it came out and by 1957 the first NC systems were being installed in industry (3). The software required to drive these systems was very bulky and inefficient. To get around this problem, Arnold Siegel of MIT proposed an automatic programming system in 1956. The system was to allow the programmer to talk to the computer in simple English-like language. It was to be called APT--Automatically Programmed Tools (3). This was a very important step in the development of computer-aided manufacturing.

In the 1960's, the NC systems became more accurate and easier to operate. At the same time, they were developing in two different directions. The first direction was that of remote control. Remote control means that the tool path is not actually calculated by the NC system but is calculated elsewhere and the machine movements corresponding to this calculated tool path are communicated to the NC system. The second direction of development was that of interpolation at the machine control unit. This concept meant that only the starting and ending points of a segment would be specified by the operator and the machine would interpolate the rest of the tool path.

In the 1970's these two directions developed into the DNC (Direct Numerical Control) and the CNC (Computer Numerical Control) systems (3). With DNC, one computer drives several NC machines. The computer simply transmits the codes to drive the machines directly to the NC system. With CNC, one computer is dedicated to one NC machine. In the CNC system, we may have interaction between the NC machine and the computer. As computers become smaller and cheaper, we will find that these CNC systems will become more attractive.

Some recent research in the area of CNC systems was done in the Department of Industrial Engineering laboratories at Kansas State University by Dr. John E. Biegel and Mr. Bruce P. Mignano (4). They developed, in 1980, a system whereby a microcomputer was used to control a two axis milling machine driven by DC stepper motors.

One of the most important advances of this research was that a milling machine and not an NC milling machine was the recipient of this modification to become a computer controlled system. This was important because at the present time it is not possible to buy an interface system to convert a machine tool to computer control. Presently, if a shop wishes to have an NC machine, it must buy an NC system from a tool manufacturer. But as the microcomputer is inexpensive and the interfaces between a machine tool and a microcomputer become available, every machine shop will have the opportunity of becoming computerized.

1.3 PROBLEM STATEMENT

To design, build, and test an interface between a microcomputer and an engine lathe.

1.4 PURPOSE

To provide an inexpensive alternative to the numerically controlled machining system.

At the present time, it is not possible to purchase a little black box which one may connect up to an engine lathe to convert it into a computer controlled machine. If one wishes to have some sort of numerically controlled system, he must purchase an NC machine from a machine tool manufacturer. It is the objective of this research to use a microcomputer to provide an inexpensive way for the average machine shop to obtain the advantages of computer controlled machinery.

1.5 METHOD

The process of interfacing a lathe and a microcomputer consisted of three phases: 1) Design, 2) Construction, and 3) Testing. The first phase consisted of the design of the electronic circuitry necessary for the interface. The second phase consisted of the actual construction of these circuits and fitting them together with the lathe and the microcomputer. The final phase was to test the feasibility of the system by producing a predetermined part configuration in acrylic plastic.

1.6 EQUIPMENT

The equipment used in this research is described below:

The lathe is manufactured by Sherline Products, Inc.* It has a spindle speed capability ranging from 200 to 2000 rpm. There are two 1/100 HP AC reversible motors mounted to drive the cross-feed and the carriage-feed. The cross-feed motor is geared down to 4 rpm while the carriage-feed motor is geared down to 63 rpm. The lead screws along both axes are cut to 20 threads per inch. The handwheels controlling the screws are divided into 50 graduations. The graduations are spaced so that each graduation represents a .001 inch linear travel along either the X axis (cross-feed or part radius) or the Z axis (carriage-feed). The maximum swing over the carriage is 1.75 inches. The maximum distance between centers is 8 inches. The tailstock spindle can travel 1.5 inches and the cross-slide can travel 2.25 inches. The lathe can be set up in an area that is 2 ft. by 1 ft.

The microcomputer is the MMD-1 manufactured by E & L Instruments, Inc.# This microcomputer utilizes the 8080A microprocessor made by Intel. The RAM (Random Access Memory) for this microcomputer has a capacity of 512 words. An additional 1024 words of RAM was added by expanding the microcomputer with an M1# board. The word size for this system is one byte (eight bits). The M1 board also provides an audio recorder interface which will allow programs to be stored on and retrieved from cassette tapes. Data and the initial program can be entered by the use of a 16 key keyboard. Data may also be brought into the

* Sherline Products, Inc., 170 Navajo Street, San Marcos, CA

# E & L Instruments, Inc., Derby, CT

system or output from the system through the computer's special interface sockets. All programming is in a machine language coded in a base eight or octal numerical system.

CHAPTER 2

DESIGN

DESIGN

2.1 GENERAL CONCEPT OF OPERATION

To provide the interface between a microcomputer and a lathe, it will be necessary to have two way communication between the microcomputer and the lathe. The microcomputer will send directions to the AC reversible motors to turn them on or off and the lathe must communicate it's tool position back to the microcomputer. Electronic circuits are needed, then, to take digital (5VDC) instructions from the program in the microcomputer and use them to drive the AC (110 V) reversible motors. Other circuits will be required to watch the continuous feed along the X and Z axes and communicate the position of the tool (in digital form) to the microcomputer. Figure 2 depicts this relationship in a block diagram.

2.2 MACHINABILITY DATA

The material used in this project was acrylic plastic. The machinability data for this material is listed in Table 1.

Table 1. Machinability Data For Acrylic Plastic (5)

SPEED	50--100 sfps
FEED	.003-.008 in/rev
TOOL	High Speed Steel or Carbide
COOLING	Dry, Air Jet

Using a cutting speed of V sfpm (surface feet per second) and a workpiece diameter of 1 inch, we find the following relationship for spindle speed:

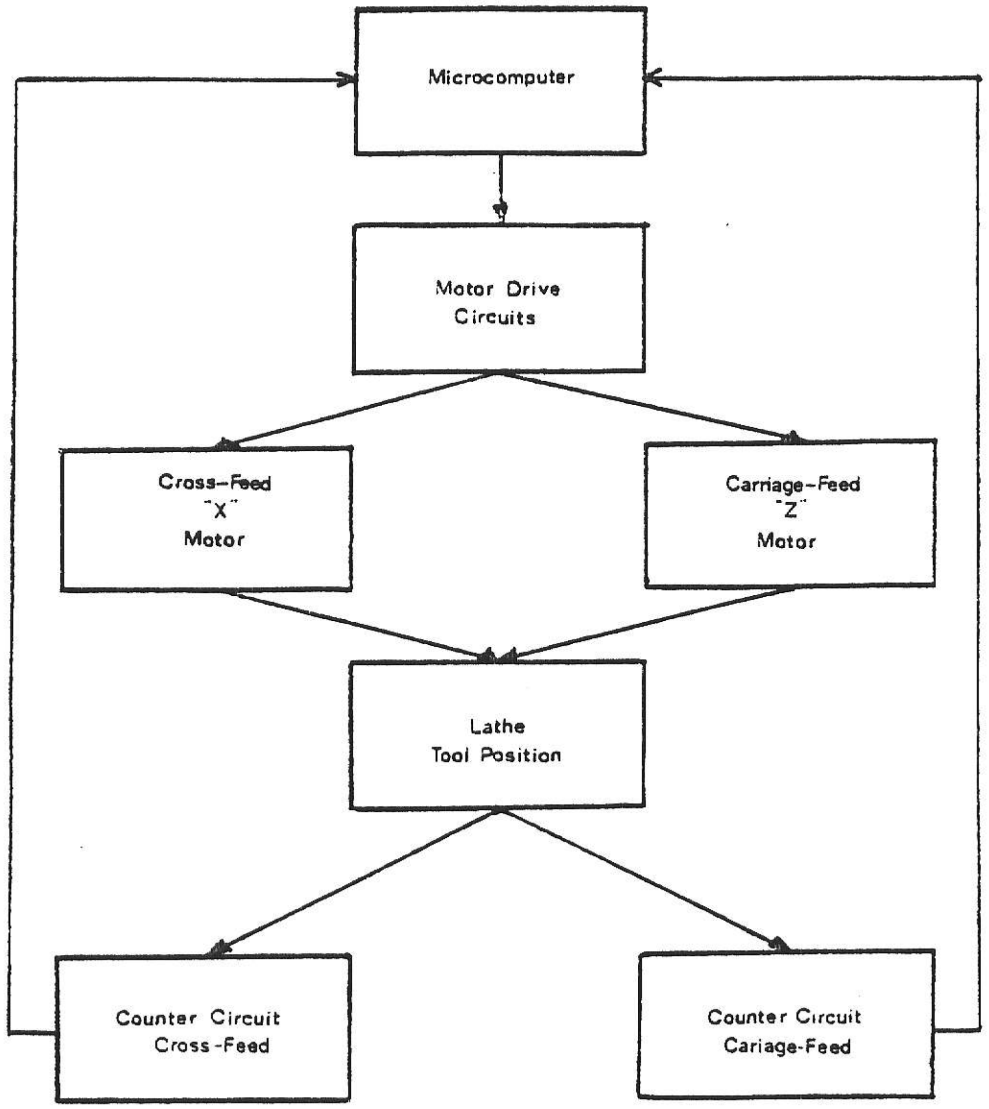


Fig. 2. General Concept of Operation

$N \text{ (rpm)} = (12 * V)/(\pi * D) * (1 \text{ min}/60 \text{ sec})$

Noting that V can range from 50 to 100 sfps for acrylic plastic, the spindle speed range for the lathe can be calculated as:

$N_1 = (12 * 300)/(3.1416 * 1) * (1/60) = 19.2 \text{ rev/sec}$ &

$N_2 = (12 * 600)/(3.1416 * 1) * (1/60) = 38.3 \text{ rev/sec}$

However, the maximum possible spindle speed is 33.3 rps. So the usable spindle speed range is 19.2 to 33.3 rps.

The feed will be determined by the carriage-feed motor speed. The motor runs at 1.05 rps and moves the carriage .05 inches/revolution. Therefore, the feed is .0525 inches/sec (1.05 rps * .05 ipr) along the Z axis. To find the feed in ipr (inches/revolution), we will need to use the spindle speeds calculated above:

$f_1 = .0525 \text{ in/sec} * 1 \text{ sec}/19.2 \text{ rps} = .0027 \text{ ipr}$

$f_2 = .0525 \text{ in/sec} * 1 \text{ sec}/33.3 \text{ rps} = .0016 \text{ ipr}$

Neither one of these feed rates meets the minimum feed rate specified in Table 1. For the purpose of this project, a high quality cut is not needed. The feasibility of the system can be shown with a comparably rough cut. It should be noted that if a high quality cut is necessary, the AC reversible motor which drives the carriage-feed can be connected to a gearbox which will allow it to turn at a faster rate (for N = 19.2 rps, a feed rate of .003 ipr would require that the carriage-feed screw turn at 1.15 rps while a feed rate of .008 ipr requires the carriage-feed screw to turn at 3.07 rps.)

2.3 BASIC OPERATION OF THE MMD-1

Being part of the microelectronic world, the MMD-1 lives completely in digital communication. It recognizes only two different signals:

1) +5 volts and 2) Ground. When a digital circuit encounters a potential of +5 volts (actually 2.4 to 5.5 VDC), it recognizes this as a "1" and when it encounters a ground potential (actually 0.0 to 0.4 VDC), the digital circuit associates this state with a "0". These two signals, a "1" and a "0", constitute the entire language of the MMD-1. To the MMD-1, a device is either <u>ON</u> or it is <u>OFF</u>.

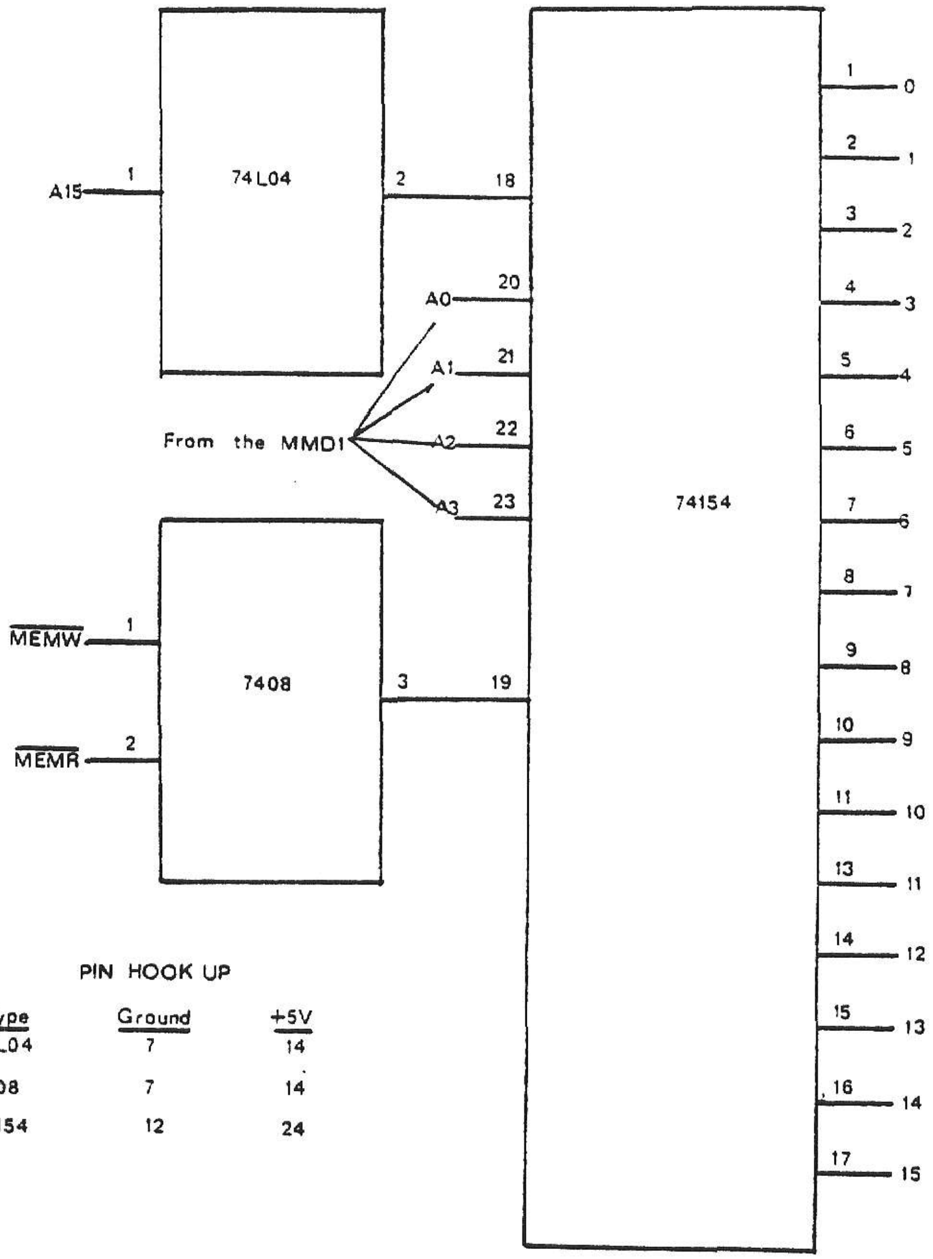
The smallest unit of communication is the bit (binary digit). This is one unit of information, either a "1" or a "0". Most commonly, however, the MMD-1 talks in it's own word system. The words in this system consist of combinations of eight bits. These eight contiguous bits are also known as one byte.

The computer sends all of it's communication along bus lines. These bus lines are paths by which information can be exchanged. One important characteristic of a bus is that only one transfer of information can take place at any one time.

The important busses to this research are the address bus and the bi-directional data bus. The address bus is used to identify memory locations or I/O (input/output) devices and the bi-directional data bus can be used to either input or output information between the microcomputer and the I/O devices.

2.4 DECODER CIRCUIT

To direct the electronic circuitry required to interface the lathe to the microcomputer, the microcomputer must be able to address the different circuits involved and be able to synchronize the actions of these devices and the microcomputer. A block diagram of the circuit designed for this purpose is shown in Figure 3.



PIN HOOK UP

Type	Ground	+5V
74L04	7	14
7408	7	14
74154	12	24

Fig. 3. The Decoder Circuit

The decoder circuit is to decode the low address byte, when it is available on the address bus, and choose a unique I/O (Input/Output) device. Because there are only twelve different I/O device signals needed for this interface, only the four lowest address bits, A_0, A_1, A_2, and A_3, need to be decoded. These four address bits are capable of representing up to sixteen different I/O devices.

To decode these four address bits, a four line to sixteen line decoder is used. The chip capable of serving this purpose is the 74154 IC (Integrated Circuit) chip. This chip requires two device enable signals and the four bits, A_0-A_3. The logic table for the operation of the 74154 is shown in Table 2. As shown in Table 2, the 74154 decodes the four lowest address bits and produces a low pulse at the output port addressed by the microcomputer.

To utilize this decoder circuit, the microcomputer may input or output data by either accumulator I/O or memory mapped I/O. Accumulator I/O utilizes two instructions to get information into or out of the microcomputer and all transfers take place through the accumulator register. In memory mapped I/O, there are thirty-three different instructions to get information into or out of the microcomputer. Also, transfers of data may take place through any of the seven special registers present within the MMD-1. Memory mapped I/O was used in this research.

With memory mapped I/O, each I/O device is addressed as if it were simply a unique memory location within the microcomputer. To distinguish between a memory location and a memory I/O device, the address bus bit A_{15} is used. When A_{15} is high, the microcomputer is talking to a memory location. Whenever it is necessary to talk to an

I/O device, it is called by specifying the low address byte and the high address byte. When the high address is 2XX or 3XX, i.e., A_{15} is a "1", an I/O device is being serviced. The low address byte will be used to identify a unique I/O device. When the high address byte is 0XX or 1XX, a memory location is being serviced (A_{15} is a 0).

Table 2. Logic Table For The 74154 IC Chip

ENABLE 1	ENABLE	INPUTS D	C	B	A	OUTPUTS 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
X*	1	X	X	X	X	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	X	X	X	X	X	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0	0	0	0	0	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0	0	0	0	1	0	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1
0	0	0	0	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1
0	0	0	1	0	0	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1
0	0	0	1	0	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1
0	0	0	1	1	0	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1
0	0	0	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1
0	0	1	0	0	0	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1
0	0	1	0	0	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1
0	0	1	0	1	0	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1
0	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1
0	0	1	1	0	0	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1
0	0	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1
0	0	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1
0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0

* An "X" in the table means that the state of this bit does not effect the output of the device.

Since bit A_{15} is a unique pulse when an I/O device is addressed, it may be used as a device select or a device enable pulse. Whenever the microcomputer is going to talk to the lathe a "1" will be present at the A_{15} socket and the "1" (present only for the device selection) can be used to enable the address decoding circuit.

As noted in Table 2, both enable signals need to be "0"s for the 74154 to function. Therefore, the A_{15} bit must be inverted in order for it to be used as an enable signal for the 74154. This is the reason that the 74L04 is part of the decoder circuit. The 74L04 is a low power input, inverter gate. The logic table for an inverter is shown in Table 3. The only time that a "0" will be available from the 74L04 to

Table 3. Logic Table For the 74L04 IC Chip

INPUT	OUTPUT
1	0
0	1

be used as an enable signal for the 74154 (see Figure 3) is when an I/O device is being addressed, i.e., the high address byte is 2XX or 3XX (A_{15} is a "1"). With bit A_{15} as one enable signal for the 74154, another enable signal is required in order for the decoding circuit to function.

To inform the circuits of the availability of the memory device select code on the address bus, a control signal is generated by the microcomputer at the same time as the number code of the I/O device is available on the address bus. This control signal, then, can be used as a device enable signal to synchronize the appearance of the unique device code on the address bus and the enabling of the device to decode this numerical code.

In memory mapped I/O, the control signals available are the $\overline{\text{MEMW}}$ and the $\overline{\text{MEMR}}$. A $\overline{\text{MEMW}}$ pulse is an electrical pulse generated whenever information is output from the microcomputer to an I/O device and a $\overline{\text{MEMR}}$ pulse is generated whenever data is to be input into the microcomputer. In either case, the pulse generated as a control signal is a "0" or an electrical pulse of ground potential. This control signal can be used directly as an enable for the decoder circuit.

However, there is only one additional enable input needed to enable the 74154 and there are two possible control signals, $\overline{\text{MEMR}}$ and $\overline{\text{MEMW}}$. Either two different decoding circuits are required, one with the $\overline{\text{MEMR}}$ pulse as an enable signal for the decoding of input device codes and the other circuit with the $\overline{\text{MEMW}}$ pulse as an enable signal for the decoding of output device codes, or the two control signals must be combined into one enable signal for a common decoding circuit. The latter choice was taken and so whenever an I/O device is addressed, the 74154 will require a "0" as an enable signal that will be produced from the combination of the $\overline{\text{MEMR}}$ and the $\overline{\text{MEMW}}$ control signals. To reiterate, whenever communication between the computer and an I/O device takes place, a control signal goes to ground potential, at all other times these control signals are "1"s or at +5 volts. Therefore, a device is needed that will give a "0" whenever an I/O device is addressed and one control signal is low and the other control signal is high. This device must also produce a "1" to disable the 74154 whenever both control signals are at the +5V state. The device that is capable of this logic is a 7408 IC chip. This chip is an AND gate and it's logic is shown in

Table 4.

Table 4. Logic Table For An AND Gate (7408)

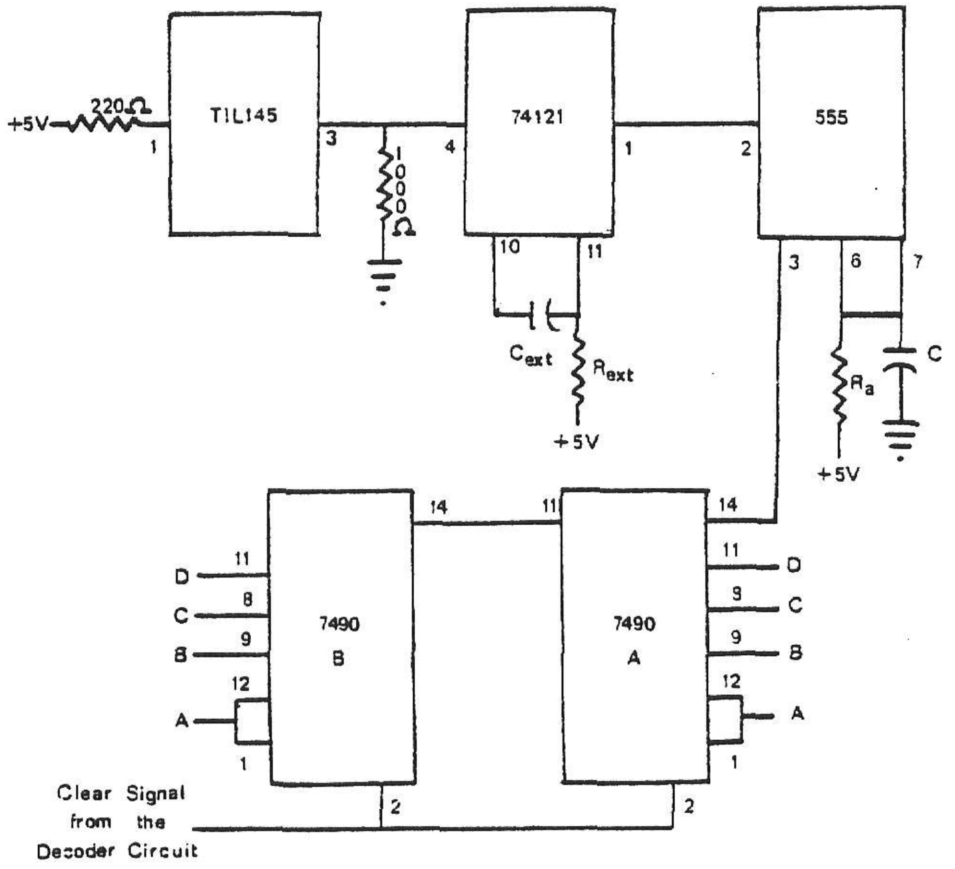
INPUT B	INPUT A	OUTPUT C
0	0	0
0	1	0
1	0	0
1	1	1

To summarize the decoder circuit, there are three signals ($\overline{MEMR}$, $\overline{MEMW}$, and A_{15}) used to enable a 74154 chip that decodes the device select signal. The device select signal consists of the four lowest bits of the address bus and appears on the address bus at the same time as the enabling pulses are generated. The decoder circuit produces, from these signals, a unique low pulse at one of the output ports of the 74154. This unique signal will be used as input to or to enable one of the circuits used in the interface between the lathe and the microcomputer.

2.5 COUNTER CIRCUIT

The counter circuit's purpose is to keep track of the lathe's tool position. This is accomplished by counting the number of increments that the lead screws for both the carriage-feed and the cross-feed have turned. The circuit designed for this purpose is shown in Figure 4.

In order to accomplish this task, evenly spaced holes were drilled around the outer perimeter of a disk attached to the handwheel. There



Pin Hook Up

Type	Gnd	+5V	Misc Pin Connections
TIL145	2	4	1 to +5V thru 220Ω, 3 to gnd thru 1000Ω
74121	7	3,5,14	11 to +5V thru R_{ext} to pin 10 thru C_{ext}
555	1	4,8	6,7 to +5V thru R_a and 7 to gnd thru C
7490	10	5	1 to 12 and 2 to the Decoder Circuit

Fig. 4. Counter Circuit

were 100 holes drilled in the cross-feed handwheel and 50 holes drilled in the carriage-feed handwheel. It is these holes that are counted by the microcomputer.

The spacing of these holes provides a means for precise control. Since both lead screws are cut to 20 threads per inch, each revolution, for either the cross-feed or the carriage-feed handwheels, represents a translation of the respective feed mechanisms of .05 inches. For the Z axis, the carriage-feed axis, the rotation of the handwheel from one hole to the next (i.e., 1/50th of a revolution) will represent a translation of the carriage-feed of .001 inches. For the X axis, the cross-feed axis, the rotation of the handwheel from one hole to the next (i.e., 1/100th of a revolution) will represent a translation of the cross-feed of .0005 inches (which is a .001" reduction in the diameter of the workpiece).

To count the holes in the handwheels, TIL145 opto-sensors were used. The TIL145 operates as a light sensing device. It consists of an LED (Light Emmitting Diode) and a light sensitive transistor. These two components are separated by a space through which an object may be passed. In the closed circuit state, there is no object between the LED and the light sensor. This allows the circuit to be complete and a "1" is the output of the TIL145. When an object comes between the LED and the light sensor, the circuit is broken and the output of the TIL145 is a "0". Therefore, as a hole is viewed by the opto-sensor, the light from the LED will strike the optical sensor and a "1" will be produced by the TIL145 and as the wheel rotates and is between holes, the light from the LED is blocked off from the light sensor and a "0" is produced by the TIL145. The result is that a signal is generated which alternates between a "1" and a "0" as holes come into and go out of view of the TIL145.

The next problem is to count these alternating signals. A 7490 decade counter was used to count the signals produced by the TIL145. The logic table for a 7490 decade counter is shown in Table 5. As the input signal to a 7490 goes from high to low, the counter increments by one. By using the output of the TIL145 as the input signal to a 7490, the holes in a handwheel disk can be counted. Every time a hole rotates out of the view of the TIL145, the input signal to the 7490 will go from a "1" to a "0" and will increment the counter. As seen in Table 5, the counter counts from 0 to 9 and then starts over again at 0. If two 7490's are cascaded together, it is possible to count from 0 to 99 before the count starts over again at 0. Cascading these counters

Table 5. Logic Table For A 7490 Decade Counter

ENABLE	INPUT SIGNAL	OUTPUT				COUNT
		D	C	B	A	
0	X	X	X	X	X	0
1	¯|_	0	0	0	0	0
1	¯|_	0	0	0	1	1
1	¯|_	0	0	1	0	2
1	¯|_	0	0	1	1	3
1	¯|_	0	1	0	0	4
1	¯|_	0	1	0	1	5
1	¯|_	0	1	1	0	6
1	¯|_	0	1	1	1	7
1	¯|_	1	0	0	0	8
1	¯|_	1	0	0	1	9
1	¯|_	0	0	0	0	0

means simply to take one of the 7490's and use it as an input to the other 7490. When the low counter changes from 9 to 0, a signal will be produced at output D (see Table 5) which goes from a "1" to a "0", this signal may be used as the input signal to the second 7490 IC chip. Thus the second 7490 will increment by one for every 10 counts of the first or lowest 7490 (see Table 6).

However, if one should try to hook up these 7490's directly to a TIL145 which is watching a rotating handwheel, he would probably find that he either gets no count at all or a very erratic count. The problem stems from the nature of the signal produced by the TIL145. As was noted earlier, the 7490 increments it's count by one as the input signal changes from a "1" to a "0". In other words, the 7490 is a "negative edge" triggered device and a sharp drop in voltage from +5V to ground is required to produce an increment in the 7490 count. As the handwheel rotates, the TIL145 will change states from a "1" to a "0" and then back from a "0" to a "1", but this change does not take place instantaneously. The edge produced is not a sharp vertical edge, but it is a gradual sloping edge. This sloping edge will not trigger an increment in the 7490. So the output from the TIL145 needs to be "squared" up, i.e., the pulse needs to be made to have a sharp enough edge to trigger the 7490.

To square up the output of the TIL145, two devices are used, a 74121 IC chip and a 555 IC chip. Both of these devices are monostable multivibrators. A monostable multivibrator is a device which has one stable state, but it may be triggered to change state for a predetermined time interval before it returns to it's stable state. The 74121 has the advantage that it may take any length of input pulse and then

Table 6. Logic Table For Two Cascaded 7490's

HIGH 7490 D C B A	COUNT	LOW 7490 D C B A	COUNT
0 0 0 0	0	1 0 0 1	9
0 0 0 1	1	0 0 0 0	0
0 0 0 1	1	0 0 0 1	1
0 0 0 1	1	0 0 1 0	2
0 0 0 1	1	0 0 1 1	3
0 0 0 1	1	0 1 0 0	4
0 0 0 1	1	0 1 0 1	5
0 0 0 1	1	0 1 1 0	6
0 0 0 1	1	0 1 1 1	7
0 0 0 1	1	1 0 0 0	8
0 0 0 1	1	1 0 0 1	9
0 0 1 0	2	0 0 0 0	0
0 0 1 0	2	0 0 0 1	1
0 0 1 0	2	0 0 1 0	2
.	.	.	.
.	.	.	.
.	.	.	.

give an output pulse of predetermined length whereas the 555 requires the input pulse to be of shorter duration than the output pulse. However, the 555 has a superior repeatability of it's output pulse widths as compared to the 74121. Therefore, the 74121 will be used to transform the output of the TIL145 to a pulse which is short enough to drive the 555. The 555 will then produce a precise pulse width which will be, when the handwheel is rotating, an excellent square wave train. It is this square wave train which is used as the input to the

7490 decade counters.

In order to utilize these monostable multivibrators, external resistors and capacitors must be connected to the devices to determine the output pulse time width (t_w). The values for these resistors and capacitors may be calculated by the use of the following equations:

For the 74121:

$t_w = (.693) * R_{ext} * C_{ext}$ for $R_{ext} < 40000$ ohms

For the 555:

$t_w = (1.1) * R_A * C$ for $1000 < R_A < 3.3$ megohms

& $C > 500$ picofarads

To calculate the values for these external components, the timing pulse widths that are necessary must be determined.

For the X axis (cross-feed), it is recalled that there are 100 holes evenly spaced around the handwheel. The lead screw is cut to 20 threads per inch and is driven by an AC reversible motor at 4 rpm. This means that the handwheel will turn one revolution every 15 seconds (4 rev/min * 1 min/60 sec). Assuming that a hole is in view of the TIL145 one half of the time and the other half of the time the metal of the handwheel is in the view of the TIL145, the length of the different pulses can be calculated as being 1/2 of the time spent between holes. In other words, a "1" pulse will be present for .075 seconds (.15/2) and then a "0" pulse will be present for .075 seconds. The output timing width of the 555 should correspond to the width of pulse available from the TIL145 and thus be approximately .075 seconds. The external components for the 555 were chosen to be:

$R_A = 680000$ ohms,

$C_{ext} = 2.2 * 10^{-6}$ farads, &

$t_w = .693 * 10000 * 2.2 * 10^{-6} = .01525$ seconds

Therefore, the pulse generated by the 74121 is shorter in it's duration than that of the 555 and can be used for the input to the 555.

For the Z axis (the carriage-feed axis), there are 50 holes evenly spaced around the handwheel. The lead screw is cut to 20 threads per inch and is driven by an AC reversible motor at 63 rpm (1.05 rps). This means that the handwheel will turn 1 revolution every .9524 seconds (63 rev/min * 1 min/60 sec). Since there are 50 holes/rev, the time between centers of successive holes is .01905 seconds (.9524 sec/rev * 1 rev/50 holes). Assuming that a hole is in view of the TIL145 one-third of the time and the other two-thirds of the time the metal of the handwheel is in view of the TIL145, the length of the high pulse can be calculated as 1/3 of the timing width or .00635 seconds (.01905/3) and the length of the low pulse can be calculated as 2/3 of the timing width pulse or .0127 seconds 2/3 * .01905). Therefore, the output of the 555 should correspond to a pulse of approximate width of .0127 seconds. To provide the required pulse width, the external components of the 555 were chosen to be:

$R_A = 1 * 10^6$ ohms,

$C = 1 * 10^{-6}$ ohms, &

$t_w = 1.1 * 10^6 * 10^{-6} = .011$ seconds.

To get a pulse of shorter duration than .011 seconds, the 74121 is again used. The external components chosen for the 74121 were:

$R_{ext} = 33000$ ohms,

$C_{ext} = 0.47 * 10^{-6}$ farads, &

$t_w = 0.693 * 33000 * 0.47 * 10^{-6} = .01075$ seconds

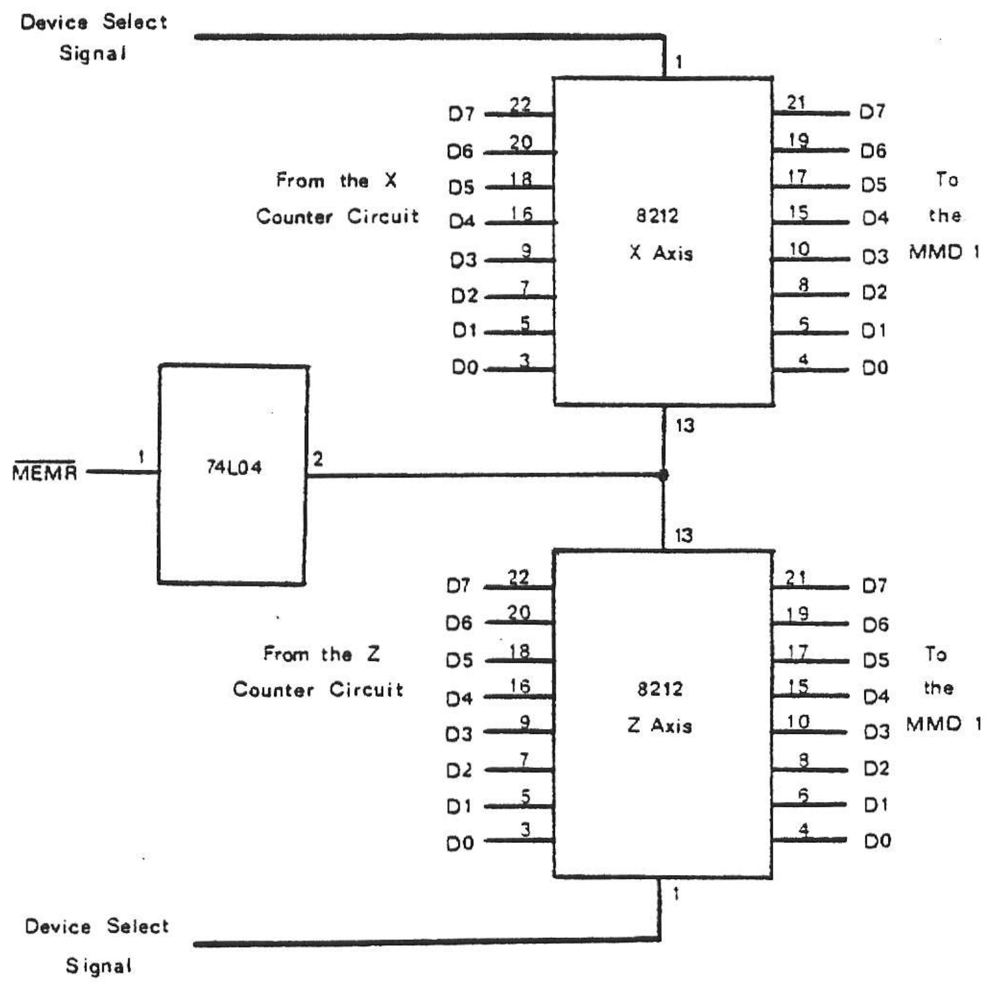
This combination of the two monostable multivibrators was used to drive the counter circuit of the Z axis.

To summarize the counter circuit, a TIL145 provides an alternating pulse which is converted into a very precise square wave train by taking the output of the TIL145 and sending it through a 74121 monostable multivibrator and then sending the output of the 74121 through a 555 monostable multivibrator. The resulting square wave train triggers a pair of 7490 cascaded counters to produce a count of the number of .001 inch increments that have been taken by the lead screws of the feed mechanisms.

2.6 INPUT CIRCUIT

The purpose of the input circuit is to take the counts produced by the counter circuits for both the cross-feed and the carriage-feed and communicate these counts, on demand, to the microcomputer. The input circuit is used to keep track of the lathe's tool position. The circuit designed for this purpose is shown in Figure 5.

The input circuit is built around the 8212 IC chip. The 8212 is a three-state data latch/buffer. It is three-state in the sense that the output has three possible states: 1) a high state, logic "1", output, 2) a low state, logic "0", output, or 3) a high impedance state where the device is essentially disconnected. The 8212 is also a data latch. As a data latch, the output of the 8212 will follow the input signal while the clock input is high, but when the clock input goes low, the data in the 8212 is latched or locked into the output ports until the next high pulse comes to the clock input. The logic table for an 8212 is shown in Table 7. The clock input has been tied to +5V



PIN HOOK UP

Type	Gnd	±5V
8212	2, 12	11, 14, 24
74L04	7	14

Misc. Pin Connections

Pin 13 to $\overline{\text{MEMR}}$ through a 74L04

Fig. 5. Input Circuit

so that the output ports directly follow the data at the input ports when the 8212 is enabled. As shown in Table 7, the device enable pulses for the 8212 need to be such that $\overline{DS_1} \cdot DS_2 = 1$ for the output ports of the 8212 to function. To get $\overline{DS_1} \cdot DS_2 = 1$, one of the device select pulses needs to be a "1" while the other pulse is a "0".

Table 7. Logic Table For An 8212 IC Chip

ENABLE $\overline{DS_1} \cdot DS_2$	INPUT	OUTPUT
0	X	High Impedance State
1	0	0
1	1	1

The 8212 will be used to take the data of the input circuit and put it into the microcomputer. It will follow the data of the counter circuit on the input side, but will be disabled and in the high impedance state on it's output side. This will allow both the input for the cross-feed and the carriage-feed counts to be connected into the bi-directional data bus at the same time. (The bi-directional data bus cannot have two signals on it at one time.) Therefore, the only time that the data bus will even know that anything is connected into it is when one of the 8212's is enabled which will take it out of the high impedance (disconnected) state at the same time as data is to be input to the microcomputer.

To get the two enabling pulses, a "1" and a "0", to the 8212 simultaneously, the purpose of the circuit needs to be examined. As stated earlier, the circuit is simply to input data to the microcomputer. This means that a $\overline{\text{MEMR}}$ instruction will be issued by the microcomputer when it is ready to accept information from the 8212. The $\overline{\text{MEMR}}$ instruction is used as one of the device enable signals for the 8212.

The 8212 must also have a unique signal as one of it's device-select signals in order for the computer to establish direct communication with the 8212 without bothering any of the other circuits. A device-select pulse from the decoder circuit can be used to accomplish this task. It is recalled that the decoder circuit selected a unique device by picking off the address available at the lowest four address bits when the control signals ($\overline{\text{MEMR}}$ or $\overline{\text{MEMW}}$ and bit A_{15}) enabled the decoding circuit. The output of this decoding circuit was a low signal at a unique output port of the decoder circuit. This unique signal from the decoder circuit may be used to inform the input circuit that the microcomputer is ready to accept data from the input circuit.

Both of the signals which have been proposed to be the enable signals for the 8212 are "0" when the input circuit is to be used. However, the 8212 requires one high and one low signal as the enabling pulses. To meet this requirement, one of these signals needs to be inverted. The $\overline{\text{MEMR}}$ was chosen to be inverted by passing it through a 74L04 inverter chip (see Table 3). The reason was to equalize the propagation delays of the two signals. The device select code must go through the decoding circuit before it is available to enable the 8212 while the $\overline{\text{MEMR}}$ signal could be taken directly from the microcomputer. Although the time that it takes for a signal to pass through a 74L04 is minimal, it is also just a very minute difference in the time that the two signals arrive at the 8212 that will keep the 8212 from being enabled and thus, functioning. So the propagation delay of the 74L04 will delay the $\overline{\text{MEMR}}$ signal's availability to the 8212 and make the time that it arrives at the 8212 closer to the time that the device select signal from the decoder circuit arrives at the 8212.

This input circuit is capable of inputing data into the microcomputer on demand.

2.7 MOTOR DRIVE CIRCUIT

The purpose of the motor drive circuit is to take the control directions from the microcomputer and drive the AC reversible motors which are used to move both the cross-feed and the carriage-feed mechanisms. There are two essential components to the motor drive circuit. They are: 1) a device which is capable of taking DC (+5V) commands and driving AC (110V) motors and 2) devices to interpret and hold these digital commands to control the AC motors.

The device which is capable of taking +5VDC input and controlling 110V AC output, is a solid state relay. The relay used in this research was the MX-100 manufactured by the Theta-J Corporation*. A solid state relay is a device which consists of an LED in the input side of the device and a photo transistor in the output side of the device. The output side of the relay has two connections, one of these connections is connected to a 110V supply and the other connection is made to an output device. The input side consists of ports for a +5V connection and a ground connection. When both the +5V pin and the ground pin are properly connected, the input side is complete and the LED lights up,

* Theta-J Corporation, Woburn, MA

otherwise the LED is unlit. Normally, the two output pins are not connected together, that is, when the input side of the relay is not complete and the LED is unlit, then the output side is disconnected. However, when the input side is complete, causing the LED to be lit, the photo transistor sees the light and closes the output side of the relay allowing current to flow between the supply pin and the output pin connected to an external device. When the input circuit is broken, i.e., either the +5V or the ground of the input side is not present, the LED will go off and the photo transistor will break the circuit on the output side and the 110V will not be available to supply an external device. So the solid state relay is capable of using a DC signal and either turn on or off a device which controls 110VAC.

In this research, solid state relays are used to turn a motor on or off and to choose it's rotation, either clockwise or counterclockwise. The connections necessary for the relays to accomplish these tasks are shown in Figure 6. Relay 1 is used to turn the motor either on or off. The other two relays, relays 2 and 3, control whether the motor turns clockwise (CW) or counterclockwise (CCW). These two relays are connected into relay 1 in order to get their 110V supply when relay 1 has been energized and the motor is turned on. Therefore, the only way a motor can work is if 1) relay 1 is enabled and thus, power is available to relays 2 and 3 and 2) either relay 2 or relay 3 is enabled (both cannot be enabled at once for the motor to work). In all cases, the input side of the circuit must be complete in order for the output side (AC) to be complete. In the circuit shown in Figure 6, +5VDC is connected (through a 330 ohm resistor) at all times to the "+" input while the "-" input is connected to ground only when a relay is to be

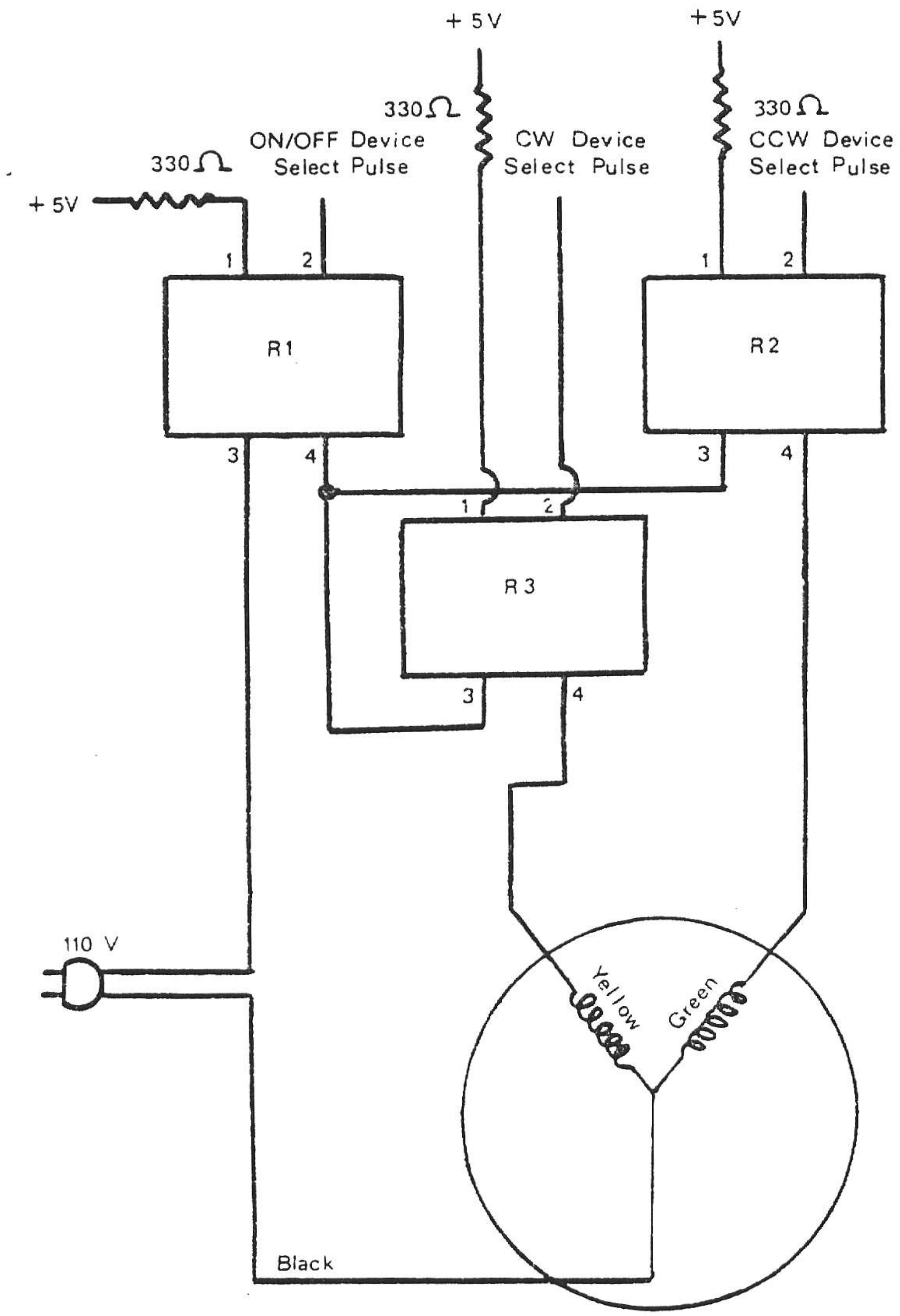


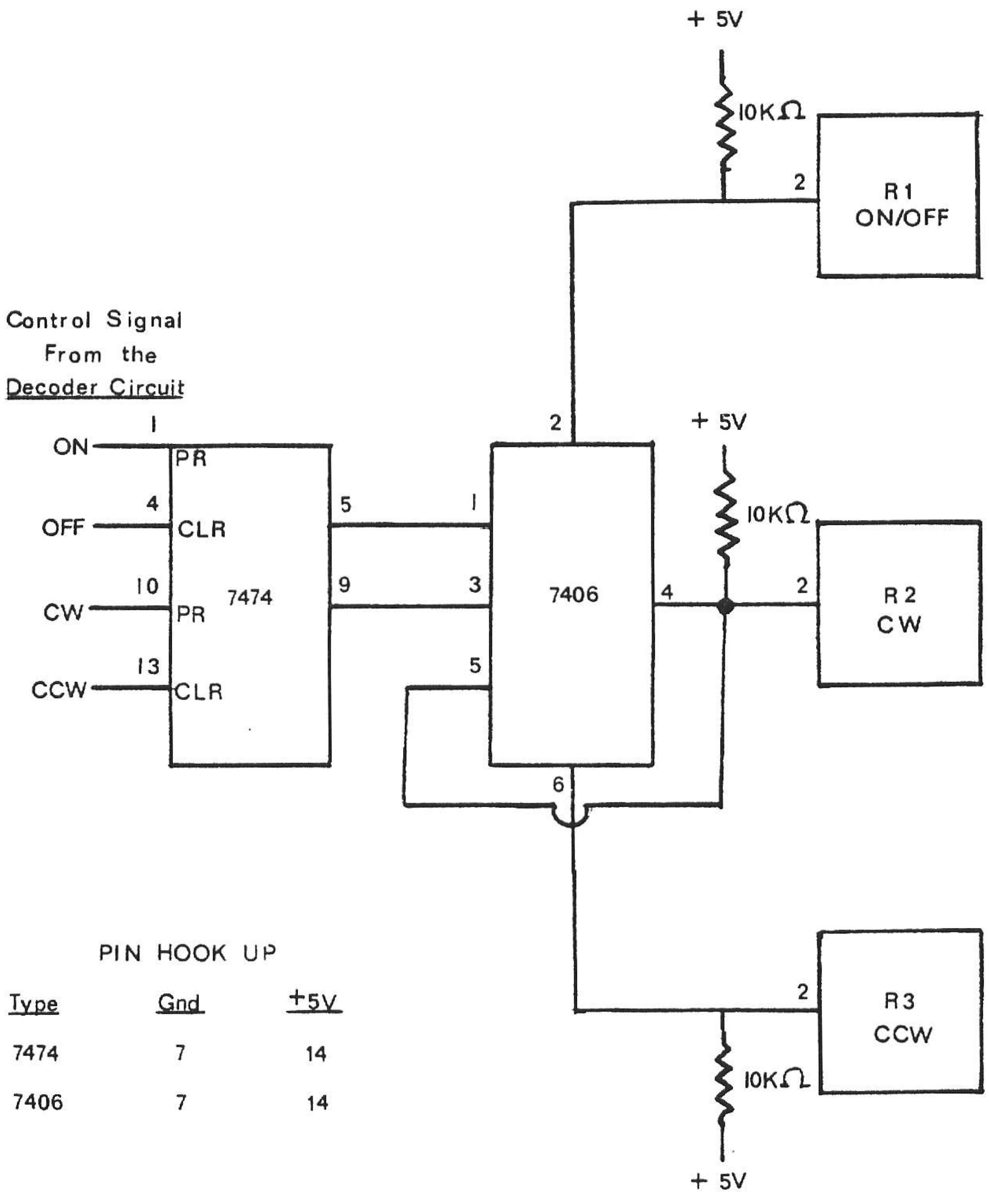
Fig. 6. AC Motor Drive Relay Circuit

enabled (the ground completes the input side of the relay). The ground signal, then, must be a unique signal which will only enable the relay that was intended to be enabled.

A unique signal from the decoder circuit (the output from the 74154) will not be sufficient to enable the relays because these signals are only present for the duration of one clock cycle (the clock of the MMD-1 operates at 750kHz). If one signal was used to enable a relay, the relay would be turned on for only 1.33 microseconds. This signal must be either output over and over to the relays to keep them on or the signal must somehow be latched into a device when this device select signal becomes available. The first option would take a lot of programming time and be inefficient. Therefore, the second option was chosen and the circuit that was designed to latch a signal and thus turn a relay on or off is shown in Figure 7.

The device chosen to latch the input signal to the solid state relays is a 7474 IC chip. This device is actually a dual data latch which incorporates a preset and a clear to each latch. For this project, the 7474 was not actually used as a data latch, but utilized only the preset and the clear functions of the data latch. The logic table for the 7474 IC, as used in this research, is shown in Table 8. This table shows that a signal to the preset input of the 7474 will directly reverse the action of a signal to the clear input. For this application, a signal to the preset input could be used to turn a relay on and then a signal to the clear input could be used to turn it off again. Either way, once the output is set, it will not change again until a signal is given to the input of the 7474 to change that output.

The only problem with this latching device is that the output of



PIN HOOK UP

Type	Gnd	+5V
7474	7	14
7406	7	14

Fig. 7. AC Motor Drive Circuit

Table 8. Logic Table For A 7474 IC chip

PRESET	CLEAR	OUTPUT
0	0	Not an allowed state
0	1	1
1	1	1
1	0	0
1	1	0

the 7474 is not a strong enough pulse to drive the relays. To overcome this problem, the output of the 7474 must be enhanced. A 7406 IC chip was used for this purpose. A 7406 is an inverter like the 74L04 used previously (see Table 3) except that it also boosts the power of it's output. This boost in power is enough so that the output of the 7406 is capable of enabling the relays. The only twist that the 7406 provides to the motor drive circuit is that they do invert the output of the 7474's.

To summarize the motor drive circuit, it is noted that the different signals needed to turn a motor on or off and either clockwise or counterclockwise are provided by the microcomputer. The decoder circuit takes these signals and prepares them to drive the motors. A 7474 takes the output signals of the decoder circuit and latches them so that they are available to drive the motors. The 7406 is used to boost the output power of the 7474 to make it capable of enabling a solid state relay that will turn the AC reversible motors on or off.

2.8 SUMMARY OF THE ELECTRONIC CIRCUITRY

The electronic circuitry designed to make up the hardware part of the interface between an engine lathe and a microcomputer consists of four major component circuits. These component circuits were the decoder circuit, counter circuit, input circuit, and the motor drive circuit.

The decoder circuit was designed to take directions from the address bus of the microcomputer and output one low signal to the port which represents the code given by the microcomputer. The signals are decoded by the decoder circuit and their intended purpose is shown in Table 9.

Table 9. Microcomputer Codes For Device Selection and Operation

OCTAL* CODE	DECIMAL CODE	PURPOSE
000	0	Turn the X axis motor on
001	1	Turn the X axis motor off
002	2	Rotate the X axis motor CW
003	3	Rotate the X axis motor CCW
004	4	Turn the Z axis motor on
005	5	Turn the Z axis motor off
006	6	Rotate the Z axis motor CW
007	7	Rotate the Z axis motor CCW
010	8	Clear the X axis count
011	9	Clear the Z axis count
012	10	Input the X axis count to the microcomputer
013	11	Input the Z axis count to the microcomputer

* This is the low address byte. In all cases the high address byte is 200.

The counter circuit keeps track of the lathe's tool position. To do this, the holes in the disks attached to the handwheels of the feed mechanisms are counted as the handwheels rotated. The count produced by this action can be directly related to the tool's movement and the tool position.

The input circuit takes the counts formulated by the counter circuits and inputs these counts to the microcomputer. A three-state data, latch/buffer which inputs data to the microcomputer only when it is enabled, is used for this purpose.

The motor drive circuit takes the decoded control signal from the decoder circuit and controls the AC reversible motors to drive the cross-feed and carriage-feed mechanisms.

These four circuits constitute the hardware portion of the interface between the engine lathe and a microcomputer. This hardware is coupled with the proper software (programming) to make the interface usable.

CHAPTER 3

CONSTRUCTION

CONSTRUCTION

The construction phase of this research consisted of the building of the electrical circuits described in Chapter 2 and the mounting and interconnection of this circuitry with the microcomputer and the lathe.

The first step was to take the circuits designed in phase one and assemble them in a temporary fashion upon breadboards. These circuits were tested as individual components and when they were found to be satisfactory, they were assembled onto printed circuit (PC) boards. To assemble a PC board, one must solder the required devices into place and solder all of the interconnections between the different devices involved in a designed circuit. There were three PC boards used for this purpose. The first board contained the counter circuit. This circuit was connected into another PC board used to display the counts produced in the counter circuit. The display circuit is not necessary to the interface between the lathe and the microcomputer and was, therefore, omitted from description. For those interested, the display circuit contained four seven-segment displays and the drivers necessary to power these displays. The second PC board contained the input circuit. The third PC board contained both the decoder circuit and the motor drive circuit. The last component of this design was a perforated board which held the relays necessary to drive the AC reversible motors attached to the cross-feed and the carriage-feed mechanisms.

The next step was to connect the four circuits together so that the signals required by one circuit, which were produced by another circuit, would be available for use. The connections between the microcomputer and the lathe were then made. These connections between the lathe and the microcomputer included the connection of the two TIL145's, mounted to straddle the handwheels of the cross-feed and carriage-feed mechanisms, to the counter circuit. They also included the connection of all the circuits to +5V and ground. Finally, the connections to the microcomputer were made. These included connections to the $\overline{\text{MEMR}}$, $\overline{\text{MEMW}}$, A_0,A_1,A_2,A_3,A_{15},D_0,D_1,D_2,D_3,D_4,D_5,D_6, and D_7 output ports of the MMD-1 microcomputer.

Upon completion of this phase, the interface was ready for testing.

CHAPTER 4

TESTING

TESTING

The final phase of this research was the testing of the feasibility of the system. For even if the design seems theoretically sound, it is totally useless unless it may be used to actually produce a predetermined part. Therefore, the test for this system is to make a predesigned part. The part to be produced by this system is shown in Figure 8. This piece is to be made from a 5" rod of acrylic plastic that is 1" in diameter.

The programs necessary to control the interface needed to be written. These programs are described in the next chapter.

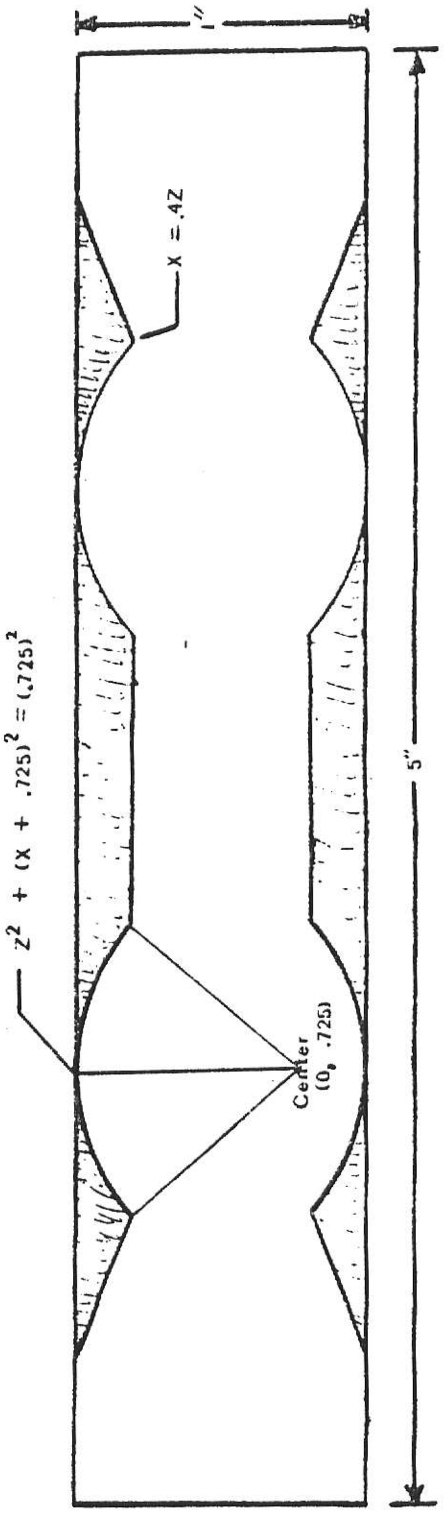


Fig. 8. Test Part

CHAPTER 5

PROGRAMMING

PROGRAMMING

To program the microcomputer to make the test part, machine language must be used. Machine language is the language which the computer uses. It is simply numerical codes which represent binary words which the computer can understand. A detailed summary of the machine language used is given in (16) and the basics of using this machine language in programming the MMD-1 microcomputer are given in (17).

There are five main sections to the program used to machine the part shown in Figure 8. They are the X-Mov subroutine, the Z-Mov subroutine, the Move-Check subroutine, the semicircle subroutine, and the main program which calls these subroutines and controls the overall system.

The X-Mov subroutine, listed in Appendix A, is used to move the cross-feed mechanism in or out by choosing the rotation of the motor which drives the mechanism. The input required in the calling program is the choice of the rotation direction of the motor, the choice of rotation direction of the software brake, and the length (specified as the number of holes to be counted) that the cross-feed should travel. The subroutine takes the count from the counter circuit and compares it to the value set by the calling program. As soon as the proper count has been reached, the program executes a software brake which is used to minimize the coasting of the motor when it has been turned off. The software brake works by giving a short pulse to the motor which will act to turn the motor in the opposite direction from which it has been turning. Once this brake has been executed, the subroutine sends control back to the main program. If the proper count has not been reached, the subroutine sends control back to the main program while the motor is still on. The main program will call the X-Mov subroutine over and over until the proper count has been reached. When the proper count has been reached and the motor is turned off, the main program will continue through it's course. A flow chart for the X-Mov subroutine is shown in Figure 9.

Because the counter circuit was set-up to count in Binary Coded Decimal (BCD), the number of holes which are to be counted will need to be represented in BCD. It is recalled that the number of holes to be counted is part of the input required for the use of the subroutines. Therefore, these counts must be communicated in BCD. Table 10 shows the conversion of decimal numerals to BCD numerals and then the conversion of this number to it's octal representation. This octal number is the one which is used by the main program, subroutines, and the microcomputer.

The Z-Mov subroutine works in exactly the same manner as the X-Mov subroutine. The only difference is that the codes for turning a motor on and off for the Z axis are used instead of the codes for X axis control. This subroutine is listed in Appendix B.

The Move-Check subroutine is used to monitor simultaneous steps along the X and the Z axes. This subroutine is listed in Appendix C and a flow chart of this subroutine is shown in Figure 10. The Move-Check subroutine allows both the X axis and the Z axis motors to be controlled simultaneously. The program monitors the counts for both the X axis and the Z axis and turns the motors off when they have reached their proper counts. When both motors have reached their proper counts and have been turned off, the subroutine transfers con-

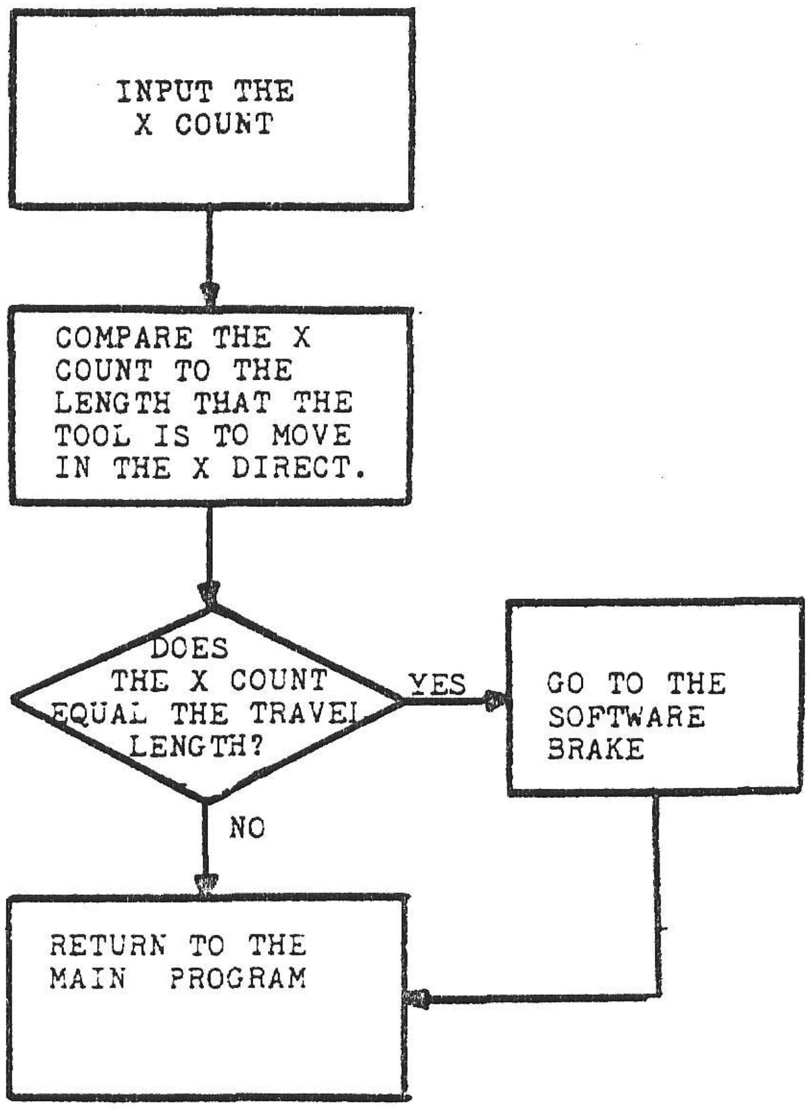

INPUT THE X COUNT
COMPARE THE X COUNT TO THE LENGTH THAT THE TOOL IS TO MOVE IN THE X DIRECT.
DOES THE X COUNT EQUAL THE TRAVEL LENGTH?
YES
GO TO THE SOFTWARE BRAKE
NO
RETURN TO THE MAIN PROGRAM

Fig. 9. X-Mov Subroutine Flow Chart

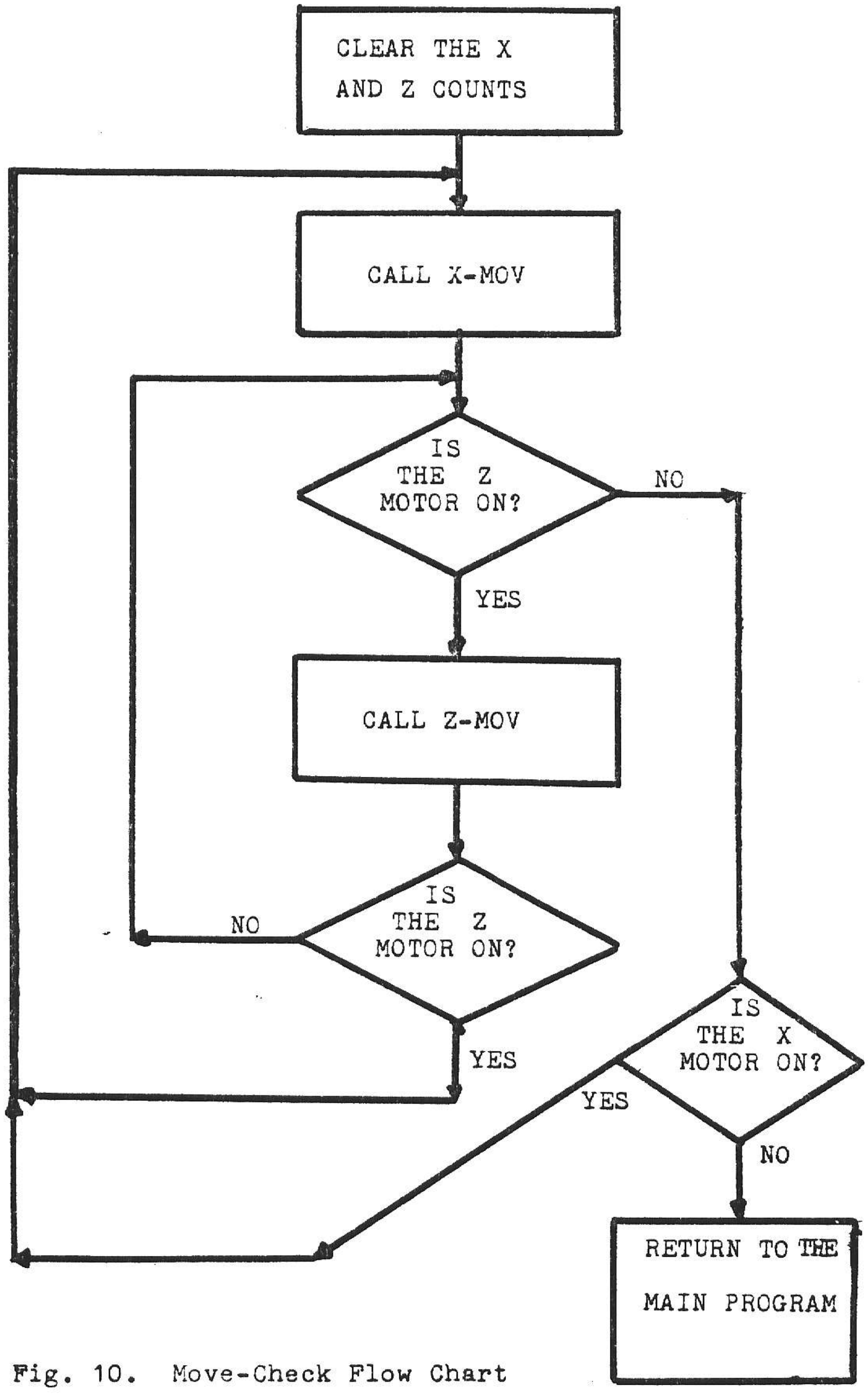


Fig. 10. Move-Check Flow Chart

trol back to the calling program.

Table 10. BCD Numeral Conversion

DECIMAL	BINARY	OCTAL
1	0000 0001	001
2	0000 0010	002
3	0000 0011	003
4	0000 0100	004
5	0000 0101	005
6	0000 0110	006
7	0000 0111	007
8	0000 1000	010
9	0000 1001	011
10	0001 0000	020
11	0001 0001	021
12	0001 0010	022
13	0001 0011	023
14	0001 0100	024
15	0001 0101	025
16	0001 0110	026
17	0001 0111	027
18	0001 1000	030
19	0001 1001	032
20	0010 0000	040

The semicircle subroutine was set-up to allow the semi-circular arc portions of the test piece to be turned. The semicircle subroutine is listed in Appendix D. The input required to use this subroutine is to set the X and the Z motor rotation directions and to set the software brakes for both of the axes. The subroutine works by calling the Move-Check subroutine which uses the X-Mov and the Z-Mov subroutines in order to step around the curve. The semicircle subroutine actually works as a quarter of a semicircular arc in the way that it uses it's data. The data for the number of holes to be counted for the feed mechanisms, as it travels around the first quarter of the semicircular arc, are found in successive memory locations. The program uses the first location as input to the X-Mov subroutine to move along the X axis, the second location to move along the Z axis, the third to move along the X axis and so on. When the subroutine has finished the first quarter of the curve (all of the data has been read), it changes the cross-feed direction (from moving out to moving in) and starts at the last data location (a Z move) and goes backward through the memory addresses moving simultaneously along the X and the Z axes. When the first memory address with data in it is reached, the semicircular arc is complete and control transfers back to the main program. A flow chart of the semicircle subroutine is shown in Figure 11.

The arc of the circle which was machined into the workpiece can be described by the following equation:

$$Z^2 + (X + .725)^2 = (.725)^2$$

With this equation, X was calculated for different values of Z. The difference between successive values of X and of successive values of Z were then found and used as the amount of movement along each axis for each step. The length of each step's movement was approximated by a number of holes to be counted and this count was used as the data input to the semicircle subroutine. Table 11 shows the results of these calculations. The resulting data, the number of holes to be counted, was converted to BCD and then to the octal representation of these BCD numbers and read into memory for the semicircle subroutine to use.

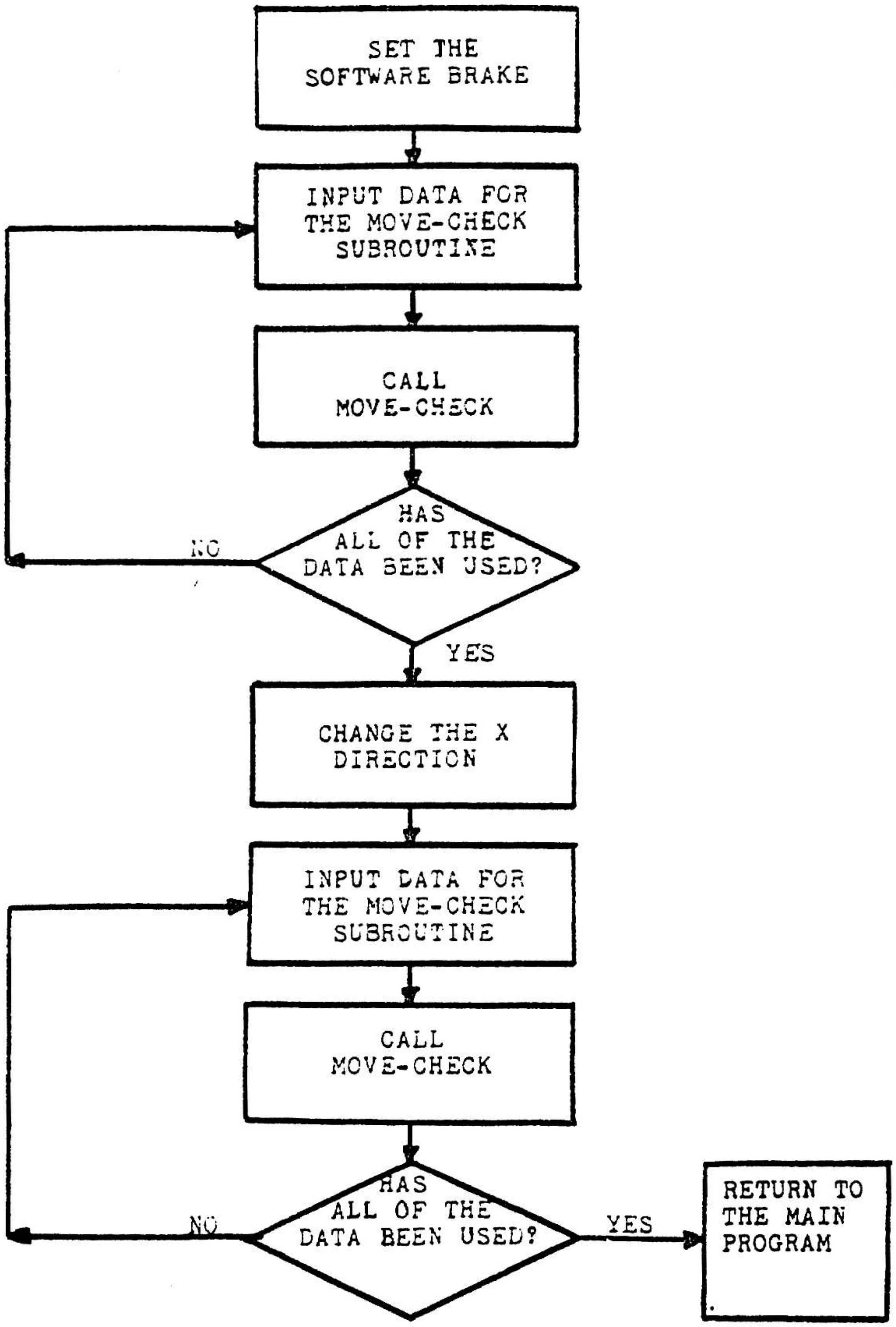

SET THE SOFTWARE BRAKE
INPUT DATA FOR THE MOVE-CHECK SUBROUTINE
CALL MOVE-CHECK
HAS ALL OF THE DATA BEEN USED?
NO
YES
CHANGE THE X DIRECTION
INPUT DATA FOR THE MOVE-CHECK SUBROUTINE
CALL MOVE-CHECK
HAS ALL OF THE DATA BEEN USED?
NO
YES
RETURN TO THE MAIN PROGRAM

Fig. 11. Semicircle Subroutine Flow Chart

This data allows a maximum error along the tool path of .005 inches. Each quarter of the arc, used for this test piece, had 43 steps.

The main program was used to control all of the system in making the test part. This program consists of two parts, the initialization part and the actual machining control part. The initialization was used to take the tool from a known starting point (the tailstock edge of the workpiece) and get it to the position where it could start machining the part. The system control program was used to combine the different subroutines available in order to machine the part. The tool point was controlled to follow the outline of the test part. When it had finished one pass, it would move the cross-feed into the workpiece a small increment, change all of the directions along the Z axis and start the process again. Once the tool gets back to the original position, the microcomputer will increment the cross-feed, change the direction of the Z axis feed mechanism and start the path again. This would go on until the piece had been turned to it's prescribed dimensions. The main program is listed in Appendix E.

The piece to be turned is shown in Figure 8. From this figure, the tool path may be described. The tool path starts with a taper. This taper has a slope of .4 which means that the tool is to move in the X direction 4 units for each 10 units of movement along the Z axis. When the taper is complete, the microcomputer calls the semicircle subroutine and follows it's path. Next, the tool moves 1" straight along the Z axis by calling the Z-Mov subroutine. The semicircle subroutine is then called to cut the second semicircular arc. Finally, another taper is cut which has the same slope as the first taper (.4) except that it is negative. In other words, the same taper is cut by

Table 11. X and Z Moves For A Semicircular Arc

Z	X	DIFF IN Z	DIFF IN X	NO. OF HOLES IN Z	NO. OF HOLES IN X
.50	.2000				
.49	.1907	.01	.0093	10	19
.48	.1817	.01	.0090	10	18
.47	.1730	.01	.0087	10	17
.46	.1646	.01	.0084	10	17
.45	.1566	.01	.0080	10	16
.44	.1488	.01	.0078	10	15
.43	.1413	.01	.0075	10	15
.42	.1340	.01	.0073	10	15
.41	.1271	.01	.0069	10	14
.40	.1203	.01	.0068	10	14
.39	.1138	.01	.0065	10	13
.38	.1076	.01	.0062	10	12
.37	.1015	.01	.0061	10	12
.36	.0957	.01	.0058	10	12
.35	.0901	.01	.0056	10	11
.34	.0847	.01	.0054	10	11
.33	.0795	.01	.0052	10	10
.32	.0744	.01	.0051	10	10
.31	.0690	.01	.0048	10	10
.30	.0650	.01	.0046	10	9
.29	.0605	.01	.0045	10	9
.28	.0563	.01	.0042	10	8
.27	.0522	.01	.0041	10	8
.26	.0482	.01	.0040	10	8
.25	.0445	.01	.0037	10	7
.24	.0409	.01	.0036	10	7
.23	.0375	.01	.0034	10	7
.22	.0342	.01	.0033	10	7

Table 11. (Continued)

Z	X	DIFF IN Z	DIFF IN X	NO. OF HOLES IN Z	NO. OF HOLES IN X
.22	.0342				
.21	.0311	.01	.0031	10	6
.20	.0281	.01	.0030	10	6
.19	.0253	.01	.0028	10	6
.18	.0227	.01	.0026	10	5
.17	.0202	.01	.0025	10	5
.16	.0179	.01	.0023	10	5
.15	.0157	.01	.0022	10	4
.14	.0136	.01	.0021	10	4
.13	.0118	.01	.0018	10	4
.12	.0100	.01	.0018	10	4
.11	.0084	.01	.0016	20	6
.10	.0069	.01	.0015		
.09	.0056	.01	.0013	20	5
.08	.0044	.01	.0012		
.07	.0034	.01	.0010	20	4
.06	.0025	.01	.0009		
.05	.0017	.01	.0008	20	3
.04	.0011	.01	.0006		
.03	.0006	.01	.0005	50	2
.02	.0003	.01	.0003		
.01	.0001	.01	.0002		
.00	.0000	.02	.0001		

starting from the opposite end of the taper. The tool is then moved into the part a small amount and it's path is reversed. This constituted one pass of the part. As stated earlier, this would continue until the part had reached it's predetermined dimensions. To get to the proper dimensions, the program is started with the number of passes to be made and then this number is decremented after each pass. When the pass number has reached 0, the part is finished and the system is shut off.

A generalized flow chart of this process is shown in Figure 12.

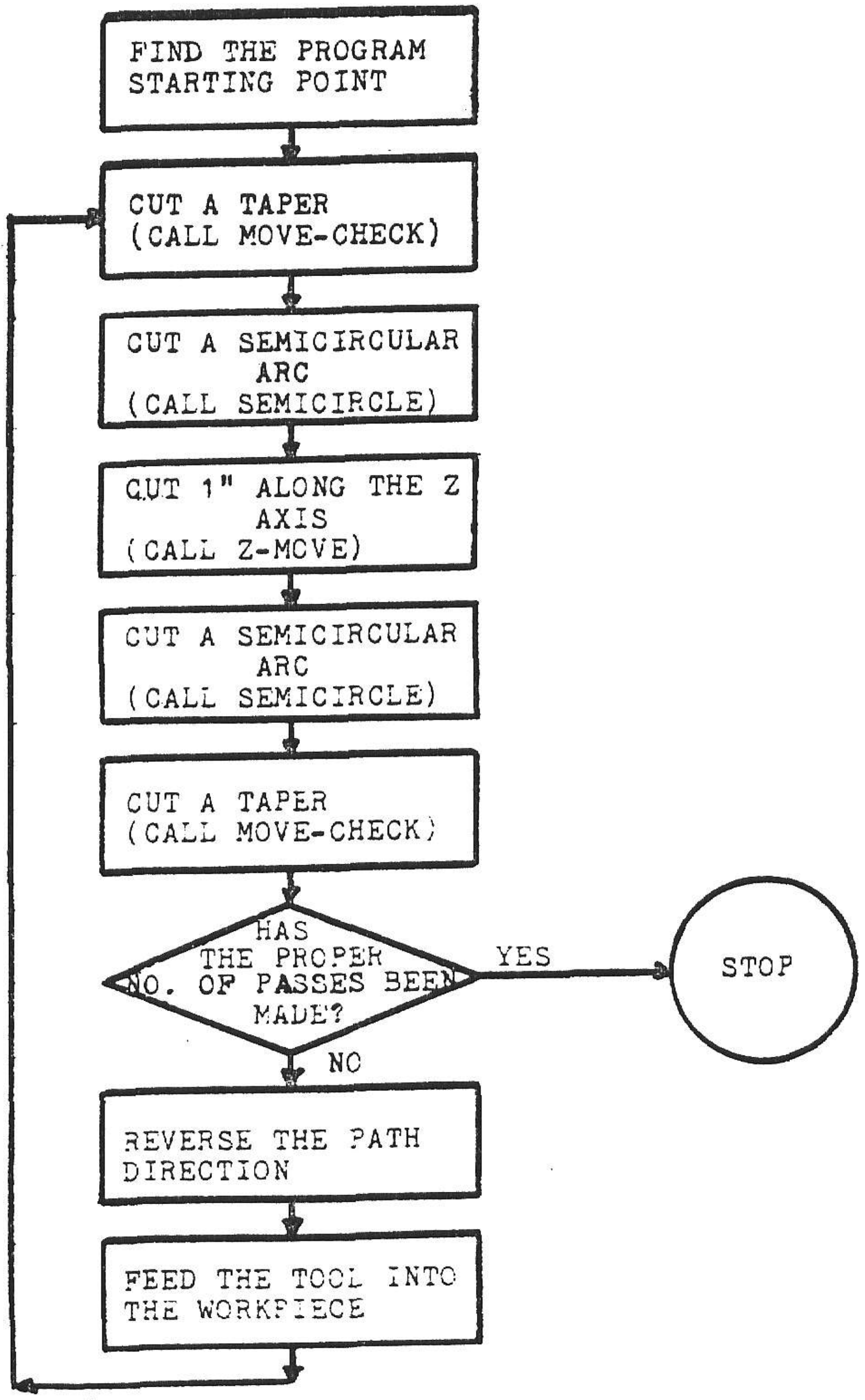

FIND THE PROGRAM STARTING POINT
CUT A TAPER (CALL MOVE-CHECK)
CUT A SEMICIRCULAR ARC (CALL SEMICIRCLE)
CUT 1" ALONG THE Z AXIS (CALL Z-MOVE)
CUT A SEMICIRCULAR ARC (CALL SEMICIRCLE)
CUT A TAPER (CALL MOVE-CHECK)
HAS THE PROPER NO. OF PASSES BEEN MADE?
YES
STOP
NO
REVERSE THE PATH DIRECTION
FEED THE TOOL INTO THE WORKPIECE

Fig. 12. Generalized Program Flow Chart

CHAPTER 6

RESULTS

RESULTS

An engine lathe has been successfully interfaced to a microcomputer. The result is a computer-controlled machining system. This system's capability was demonstrated by turning a piece of acrylic plastic to the predetermined form shown in Figure 8. This test piece took approximately 15 minutes to machine each pass. The same piece would require two set-ups to produce the double tapers on an engine lathe and the curved sections of the piece would be extremely difficult to control by hand. Thus, the microcomputer introduces new capabilities to the lathe, provides for the accurate production of a part, and provides a high degree of repeatability in the turning of several similar parts.

This same interface may be utilized by any engine lathe by merely changing the size of the relays and connecting the interface into the lathe. The cost of this system is perhaps the most significant realization of this project. The total interface, microcomputer and all of the circuitry required, can be obtained for under $1000. Therefore, for a small amount of capital investment, a machine shop can have a computer-controlled lathe which exceeds the capabilities of an NC machine system.

CHAPTER 7

FURTHER RESEARCH

FURTHER RESEARCH

In reviewing the work done on this project, a few ideas come to mind for further research. First of all, variable speed AC reversible motors should be considered for the control of the feed mechanisms. This would allow a continuous surface to be generated and not approximated, as is the case now, by steps along the two axes. The amount of programming required by the test piece was immense and took a very meticulous effort. A higher level computer language should be used by the system to lighten the programming load. Finally, the control of the spindle speed should be done by the microcomputer to increase the capability of the system.

REFERENCES

1. Woodbury, Robert S., Studies in the History of Machine Tools, The MIT Press, Cambridge, Mass., 1960

2. Perrigo, Oscar E., Lathe Design, The Norman W. Henry Publishing Co., New York, NY, 1917

3. Olesten, Nils O., Numerical Control, John Wiley & Sons, New York, NY, 1970

4. Mignano, Bruce P., Converting a Milling Machine to a Two Axis Contouring Machine, A Master's Thesis, Department of Industrial Engineering, Kansas State University, 1980

5. CADCO Engineers, "Machining the Engineering Plastics", Plastics Design and Processing, September 1968

6. Marcy, William M., "Digital Electronics for Microprocessor Applications in Control of Manufacturing Processes", AIIE Transactions, March 1980

7. Berger, Roger W., "Digital Logic and Arithmetic: Background for Interfacing Microcomputers", AIIE Transactions, March 1980

8. Goodrich, Jerry L., "Interfacing Microcomputers for Control and Data Acquisition", AIIE Transactions, June 1980

9. Wolfe, Philip M., and Houston B. Mount, "Data Input/Output Using a Microprocessor System", AIIE Transactions, June 1980

10. Berger, Roger W., "A Microprocessor-based Traffic Light Simulator", AIIE Transactions, September 1980

11. Texas Instruments Engineers, The Optoelectronics Data Book for Design Engineers, 5th Edition, Texas Instruments, Inc., Dallas, TX, 1978

12. Texas Instruments Engineers, The TTL Data Book For Design Engineers, 2nd Edition, Texas Instruments, Inc., Dallas, TX, 1976

13. __________, Motorola HEP Data Book & Selector Guide, Motorola, Inc., Tempe, Arizona, 1978

14. __________, Archer Semiconductor Reference Handbook, Radio Shack, Fort Worth, TX, 1978

15. Smith, Ralph J., Circuits, Devices, & Systems, 3rd Edition, John Wiley & Sons, New York, NY, 1976

16. Larsen, David G., Peter R. Rony, and Johnathon A. Titus, The Bug Book V, E & L Instruments, Inc., Danbury, CT, May 1977

17. Larsen, David G., Peter R. Rony, and Johnathon A. Titus, The Bug Book VI, E & L Instruments, Inc., Danbury, CT, May 1977

APPENDIX A

X-MOV SUBROUTINE

X-MOV SUBROUTINE

LOCATION	OCTAL	DESCRIPTION
030 012	365	Push PSW
3	345	Push H
4	000	NOP
5	366	Clear the CY and the Z flags
6	111	
7	000	NOP
20	000	
1	000	
2	000	
3	072	Input the X count
4	012	
5	200	
6	000	NOP
7	273	Compare the X count to the E register
30	312	If the X count equals the E register
1	052	then jump to the software brake
2	030	
3	322	
4	052	
5	030	
6	366	Clear the CY and the Z flags
7	111	
40	006	MVI B
1	000	
2	341	POP H
3	361	POP PSW
4	311	Return
5	000	NOP
.	.	.
.	.	.
.	.	.
51	000	NOP

LOCATION		OCTAL	DESCRIPTION
030	052	062	Start of the software brake
	3	001	Turn the X motor off
	4	200	
	5	062	Turn the X motor in the opposite direction
	6	003	from which it has been turning
	7	200	
	60	062	Turn the X motor back on
	1	000	
	2	200	
	3	315	Call a 10 msec time delay
	4	127	
	5	000	
	6	006	MVI B
	7	001	
	70	000	NOP
	1	062	Turn X off
	2	001	
	3	200	
	4	341	POP H
	5	361	POP PSW
	6	311	Return

APPENDIX B

Z-MOV SUBROUTINE

Z-MOV SUBROUTINE

LOCATION		OCTAL	DESCRIPTION
030	100	365	Push PSW
	1	000	NOP
	2	345	Push H
	3	366	CLR the CY and Z flags
	4	111	
	5	000	NOP
	6	000	
	7	000	
	10	000	
	1	072	Input the Z count to the microcomputer
	2	013	
	3	200	
	4	000	NOP
	5	000	
	6	272	Compare the Z count to register D
	7	312	If the count is equal to the D register
	20	141	then jump to the software brake
	1	030	
	2	322	
	3	141	
	4	030	
	5	366	CLR the CY and the Z flags
	6	111	
	7	016	MVI C
	30	000	
	1	341	POP H
	2	361	POP PSW
	3	311	Return
	4	000	NOP
	.	.	.
	.	.	.
	.	.	.

LOCATION		OCTAL	DESCRIPTION
030	140	000	NOP
	1	062	Start the software brake
	2	005	Turn the Z motor off
	3	200	
	4	062	Rotate the Z motor in the opposite direction
	5	007	from which it has been turning
	6	200	
	7	062	Turn the Z motor back on
	50	004	
	1	200	
	2	000	NOP
	3	000	
	4	315	Call a 10 msec delay
	5	127	
	6	000	
	7	000	NOP
	60	000	
	1	000	
	2	016	MVI C
	3	001	
	4	062	Turn the Z motor off
	5	005	
	6	200	
	7	341	POP H
	70	361	POP PSW
	1	311	Return

APPENDIX C

MOVE-CHECK SUBROUTINE

MOVE-CHECK SUBROUTINE

LOCATION	OCTAL	DESCRIPTION
034 000	016	Initialize C
1	000	
2	000	NOP
.	.	.
.	.	.
.	.	.
7	000	NOP
10	062	CLR the X count
1	010	
2	200	
3	062	CLR the Z count
4	011	
5	200	
6	062	Turn X on
7	000	
20	200	
1	062	Turn Z on
2	004	
3	200	
4	315	Call X-Mov
5	010	
6	030	
7	247	CLR the CY flag
30	076	MVI A
1	000	
2	271	CMP C
3	332	Jump if the Z motor is turned off
4	053	
5	034	
6	315	Call Z-Mov
7	100	
40	030	

LOCATION	OCTAL	DESCRIPTION
034 041	247	CLR the CY flag
2	076	MVI A
3	000	
4	270	CMP B
5	332	Jump if the X motor is turned off
6	027	
7	034	
50	303	Jump unconditionally
1	024	
2	034	
3	247	CLR the CY flag
4	076	MVI A
5	000	
6	270	CMP B
7	322	Jump if the X motor is on
60	024	
1	034	
2	315	Call the 10 msec delay
3	127	
4	000	
5	311	Return

APPENDIX D

SEMICIRCLE SUBROUTINE

SEMICIRCLE SUBROUTINE

LOCATION	OCTAL	DESCRIPTION
030 200	041	Select the rotation direction for the X
1	062	software brake and store it into the
2	002	X-Mov subroutine
3	042	
4	055	
5	030	
6	041	Select the rotation direction for the Z
7	062	software brake and store it into the
10	007	Z-Mov subroutine
1	042	
2	144	
3	030	
4	041	Place the no. of points used in the arc
5	053	into a memory location
6	000	
7	042	
20	175	
1	030	
2	041	Set the memory location where data may be
3	051	found
4	002	
5	062	Select the X direction
6	003	
7	200	
30	062	Select the Z direction
1	006	
2	200	
3	136	Mov E,M
4	043	INX H
5	126	Mov D,M
6	043	INX H
7	042	LHLD

LOCATION		OCTAL	DESCRIPTION
030	240	223	
	1	030	
	2	062	CLR the X count
	3	010	
	4	200	
	5	062	CLR the Z count
	6	011	
	7	200	
	50	062	Turn X on
	1	000	
	2	200	
	3	062	Turn Z on
	4	004	
	5	200	
	6	006	MVI B
	7	000	
	60	016	MVI C
	1	000	
	2	315	Call Move-Check
	3	000	
	4	034	
	5	041	Load the no. of points in the arc
	6	175	
	7	030	
	70	065	DCR M
	1	302	Jump if the content of M is NZ
	2	222	
	3	030	
	4	041	Select the X software brake rotation
	5	062	
	6	003	
	7	042	
	300	055	
	1	030	
	2	041	Set the no. of points in the arc

LOCATION	OCTAL	DESCRIPTION
030 303	053	
4	000	
5	042	
6	175	
7	030	
10	041	Set the location where the data will
1	176	start at
2	002	
3	062	Set the X direction
4	002	
5	200	
6	062	Set the Z direction
7	006	
20	200	
1	126	Mov D,M
2	053	DCX H
3	136	Mov E,M
4	053	DCX H
5	042	LHLD
6	311	
7	030	
30	062	CLR the X count
1	010	
2	200	
3	062	CLR the Z count
4	011	
5	200	
6	062	Turn X on
7	000	
40	200	
1	062	Turn Z on
2	004	
3	200	
4	006	MVI B

LOCATION	OCTAL	DESCRIPTION
030 345	000	
6	016	MVI C
7	000	
50	315	Call Move-Check
1	000	
2	034	
3	041	Load memory so the number of points
4	175	run can be known
5	030	
6	065	DCR M
7	302	JMP if the contents of M equal 0
60	310	
1	030	
2	041	Set the subroutine up for the next
3	051	time it will be used
4	002	
5	042	
6	223	
7	030	
70	041	
1	176	
2	002	
3	042	
4	311	
5	030	
6	311	Return

The following constitute the data that was used by the semicircle subroutine in making it's steps around the arc:

LOCATION	OCTAL	BCD	REPRESENTATION
002 051	031	19	This is an X data input value
2	020	10	This is a Z data input value
3	030	18	The rest of the data alternate in
4	020	10	this fashion between X and Z
5	027	17	X
6	020	10	Z
7	027	17	X
60	020	10	Z
1	026	16	.
2	020	10	.
3	025	15	.
4	020	10	
5	025	15	
6	020	10	
7	025	15	
70	020	10	
1	024	14	
2	020	10	
3	024	14	
4	020	10	
5	023	13	
6	020	10	
7	022	12	
100	020	10	
1	022	12	
2	020	10	
3	022	12	
4	020	10	
5	021	11	
6	020	10	
7	021	11	

LOCATION	OCTAL	BCD
002 110	020	10
1	020	10
2	020	10
3	020	10
4	020	10
5	020	10
6	020	10
7	011	9
20	020	10
1	011	9
2	020	10
3	010	8
4	020	10
5	010	8
6	020	10
7	010	8
30	020	10
1	007	7
2	020	10
3	007	7
4	020	10
5	007	7
6	020	10
7	007	7
40	020	10
1	006	6
2	020	10
3	006	6
4	020	10
5	006	6
6	020	10
7	005	5
50	020	10
1	005	5

LOCATION		OCTAL	BCD
002	152	020	10
	3	005	5
	4	020	10
	5	004	4
	6	020	10
	7	004	4
	60	020	10
	1	004	4
	2	020	10
	3	004	4
	4	020	10
	5	006	6
	6	040	20
	7	005	5
	70	040	20
	1	004	4
	2	040	20
	3	003	3
	4	040	20
	5	002	2
	6	120	50

APPENDIX E

MAIN PROGRAM

There are two components to the main program. They are the section of program used to set the initial tool position by indexing over from the tailstock edge of the workpiece and the section which generates the tool path for machining the prescribed test piece.

INITIAL TOOL POSITIONING

LOCATION	OCTAL	DESCRIPTION
035 000	041	Set the number of passes to be made
1	010	
2	000	
3	042	
4	300	
5	002	
6	041	Set the X software brake
7	062	
10	002	
1	042	
2	055	
3	030	
4	062	Turn the motor CCW
5	003	
6	200	
7	036	MVI E The input to the X-Mov subroutine
20	120	50 BCD
1	062	Turn X on
2	000	
3	200	
4	315	Call X-Mov
5	010	
6	030	
7	076	MVI A
30	000	
1	270	CMP B

LOCATION		OCTAL	DESCRIPTION
035	032	322	Jump if X is still on
	3	024	
	4	035	
	5	041	Check to see if the correct number of
	6	300	passes have been made
	7	002	
	40	062	CLR X
	1	010	
	2	200	
	3	065	DCR M
	4	302	Jump if the contents of M = 0
	5	014	
	6	035	
	7	000	NOP
	50	041	Set the number of loops to be made for the
	1	012	next section of program
	2	000	
	3	042	
	4	300	
	5	002	
	6	041	Set the Z software brake
	7	062	
	60	007	
	1	042	
	2	144	
	3	030	
	4	062	Turn the Z motor CW
	5	006	
	6	200	
	7	026	MVI D
	70	120	50 BCD
	1	062	Turn Z on
	2	004	
	3	200	

LOCATION	OCTAL	DESCRIPTION
035 074	315	Call Z-Mov
5	100	
6	030	
7	076	MVI A
100	000	
1	271	CMP C
2	322	Jump if Z is still on
3	074	
4	035	
5	041	Check to see if the correct number of
6	300	loops have been made
7	002	
10	062	CLR Z
1	011	
2	200	
3	065	DCR M
4	302	Jump if the contents of M $\neq$ 0
5	064	
6	035	
7	166	Halt

TOOL PATH PROGRAM

LOCATION		OCTAL	DESCRIPTION
031	000	062	Turn X off
	1	001	
	2	200	
	3	062	Turn Z off
	4	005	
	5	200	
	6	062	Clear the X count
	7	010	
	10	200	
	1	062	Clear the Z count
	2	011	
	3	200	
	4	000	NOP
	5	000	.
	.	.	.
	.	.	.
	.	.	.
	046	000	.
	7	000	NOP
	50	041	Set the number of full passes to be made
	1	005	
	2	000	
	3	042	
	4	030	
	5	031	
	6	041	Set the index that determines which way
	7	002	the tool is moving along the Z axis
	60	000	
	1	042	
	2	040	
	3	031	
	4	000	NOP

```
LOCATION    OCTAL   DESCRIPTION
031  064     000    NOP
       .      .     .
       .      .     .   The next section of program cuts a
       .      .     .   taper in the test part
     117     000    NOP
      20     062    Turn X CW
       1     002
       2     200
       3     041    Set the X software brake
       4     062
       5     003
       6     042
       7     055
      30     030
       1     036    MVI E
       2     004    4 BCD
       3     062    Turn X on
       4     000
       5     200
       6     315    Call X-Mov
       7     010
      40     030
       1     076    MVI A
       2     000
       3     270    CMP B
       4     322    Jump if X is still on
       5     136
       6     031
       7     000    NOP
      50     041    Set the number of loops to be run for this
       1     061    section of program
       2     000
       3     042
       4     310
       5     002
```

LOCATION	OCTAL	DESCRIPTION
031 156	062	Turn Z CW
7	006	
60	200	
1	041	Set the Z software brake
2	062	
3	007	
4	042	
5	144	
6	030	
7	062	Turn X CW
70	002	
1	200	
2	041	Set the X software brake
3	062	
4	003	
5	042	
6	055	
7	030	
200	026	MVI D
1	020	10 BCD
2	036	MVI E
3	010	
4	315	Call Move-Check
5	000	
6	034	
7	000	NOP
10	041	LXI H
1	310	
2	002	
3	065	DCR M
4	302	Jump if the contents of M are not equal to 0
5	156	
6	031	
7	000	NOP
20	062	Turn Z CW

LOCATION		OCTAL	DESCRIPTION
031	221	006	
	2	200	
	3	041	Set the Z software brake
	4	062	
	5	007	
	6	042	
	7	144	
	30	030	
	1	062	Turn X CW
	2	002	
	3	200	
	4	041	Set the X software brake
	5	062	
	6	003	
	7	042	
	40	055	
	1	030	
	2	026	MVI D
	3	020	10 BCD
	4	036	MVI E
	5	010	8 BCD
	6	240	CLR the CY flag
	7	315	Call Move-Check
	50	000	
	1	034	
	2	000	NOP
	.	.	.
	.	.	.
	.	.	.
	57	000	NOP
	60	041	Set the X software brake
	1	062	
	2	002	
	3	042	

LOCATION	OCTAL	DESCRIPTION
031 264	201	
5	030	
6	041	Set the Z software brake
7	062	
70	007	
1	042	
2	207	
3	030	
4	041	Set the X direction in the semicircle
5	062	subroutine
6	003	
7	042	
300	225	
1	030	
2	041	Set the Z direction in the semicircle
3	062	subroutine
4	006	
5	042	
6	230	
7	030	
10	041	Set the X software brake for the second
1	062	quarter of the semicircular arc
2	003	
3	042	
4	275	
5	030	
6	041	Set the X direction for the second quarter
7	062	of the semicircular arc
20	002	
1	042	
2	313	
3	030	
4	041	Set the Z rotation for the second quarter
5	062	
6	006	

LOCATION	OCTAL	DESCRIPTION
031 327	042	
30	316	
1	030	
2	000	NOP
3	000	
4	000	
5	315	Call the semicircle subroutine
6	200	
7	030	
40	041	Set the Z software brake
1	062	
2	007	
3	042	
4	144	
5	030	
6	041	Set the number of loops to be made
7	024	
50	000	
1	042	
2	320	
3	002	
4	062	CLR Z
5	011	
6	200	
7	062	Turn Z CW
60	006	
1	200	
2	026	MVI D
3	120	50 BCD
4	062	Turn Z on
5	004	
6	200	
7	315	Call Z-Mov
70	100	
1	030	

LOCATION		OCTAL	DESCRIPTION
031	372	076	MVI A
	3	000	
	4	271	CMP C
	5	322	Jump if Z is still on
	6	367	
	7	031	
032	000	041	Check the number of loops that have been made
	1	320	
	2	002	
	3	065	DCR M
	4	302	Jump if M ≠ 0
	5	354	
	6	031	
	7	000	NOP
	10	315	Call the semicircle subroutine
	1	200	
	2	030	
	3	000	NOP
	4	000	
	5	000	
	6	062	Turn X CCW
	7	003	
	20	200	
	1	000	NOP
	2	000	
	3	041	Set the X software brake
	4	062	
	5	002	
	6	042	
	7	055	
	30	030	
	1	036	MVI E
	2	004	4 BCD
	3	062	Turn X on
	4	000	

LOCATION	OCTAL	DESCRIPTION
032 035	200	
6	315	Call X-Mov
7	010	
40	030	
1	076	MVI A
2	000	
3	270	CMP B
4	322	Jump if X is still on
5	036	
6	032	
7	000	NOP
50	041	Set the number of loops to be run for this
1	061	section of program
2	000	
3	042	
4	310	
5	002	
6	062	Turn Z CW
7	006	
60	200	
1	041	Set the Z software brake
2	062	
3	007	
4	042	
5	144	
6	030	
7	062	Turn X CCW
70	003	
1	200	
2	041	Set the X software brake
3	062	
4	002	
5	042	
6	055	

LOCATION	OCTAL	DESCRIPTION
032 077	030	
100	026	MVI D
1	020	10 BCD
2	036	MVI E
3	010	8 BCD
4	315	Call Move-Check
5	000	
6	034	
7	000	NOP
10	041	LXI H
1	310	
2	002	
3	065	DCR M
4	302	Jump if the proper number of loops have
5	056	not been taken
6	032	
7	000	NOP
20	062	Turn Z CW
1	006	
2	200	
3	041	Set the Z software brake
4	062	
5	007	
6	042	
7	144	
30	030	
1	062	Turn X CCW
2	003	
3	200	
4	041	Set the X software brake
5	062	
6	002	
7	042	
40	055	
1	030	

LOCATION		OCTAL	DESCRIPTION
032	142	026	MVI D
	3	020	10 BCD
	4	036	MVI E
	5	010	8 BCD
	6	240	CLR the CY flag
	7	315	Call Move-Check
	50	000	
	1	034	
	2	000	NOP
	.	.	.
	.	.	.
	.	.	.
	220	000	NOP
	1	041	Load the following two bytes into H and L
	2	040	
	3	031	
	4	065	DCR M
	5	312	If M is 0 jump to this location, otherwise
	6	100	continue
	7	033	
	30	000	NOP
	.	.	.
	.	.	.
	.	.	.
	37	000	NOP

The following section of program will reverse all of the Z moves written into the program thus far in order to bring the tool back along the same tool path that it has just followed. Before it is implemented, however, the tool is moved into the workpiece an amount equal to the depth of the cut that is to be taken.

LOCATION	OCTAL	DESCRIPTION
032 240	041	Change the Z-MOV at the following location
1	062	with the given instructions
2	007	
3	042	
4	156	
5	031	
6	041	Change the next Z-MOV instruction
7	062	
50	006	
1	042	
2	162	
3	031	
4	041	Change the next Z-MOV instruction
5	062	
6	007	
7	042	
60	220	
1	031	
2	041	Change the next Z-MOV instruction
3	062	
4	006	
5	042	
6	224	
7	031	
70	041	Change the next Z-MOV instruction
1	062	
2	006	

LOCATION	OCTAL	DESCRIPTION
032 273	042	
4	267	
5	031	
6	041	Change the next Z-Mov instruction
7	062	
300	007	
1	042	
2	303	
3	031	
4	041	Change the next Z-Mov instruction
5	062	
6	007	
7	042	
10	325	
1	031	
2	041	Change the next Z-Mov instruction
3	062	
4	007	
5	042	
6	357	
7	031	
20	041	Change the next Z-Mov instruction
1	062	
2	006	
3	042	
4	341	
5	031	
6	041	Change the next Z-Mov instruction
7	062	
30	007	
1	042	
2	056	
3	032	
4	041	Change the next Z-Mov instruction
5	062	

LOCATION	OCTAL	DESCRIPTION
032 336	006	
7	042	
40	062	
1	032	
2	041	Change the next Z-Mov location
3	062	
4	007	
5	042	
6	120	
7	032	
50	041	Change the next Z-Mov location
1	062	
2	006	
3	042	
4	124	
5	032	
6	000	NOP
7	000	
60	062	Turn X CW (Feed the tool into the workpiece)
1	002	
2	200	
3	062	Turn X on
4	000	
5	200	
6	041	Set the X software brake
7	062	
70	003	
1	042	
2	055	
3	030	
4	036	MVI E
5	100	40 BCD
6	315	Call X-Mov
7	010	

LOCATION		OCTAL	DESCRIPTION
033	000	030	
	1	076	MVI A
	2	000	
	3	270	CMP B
	4	322	Jump if X is still on
	5	376	
	6	032	
	7	303	Jump back to this location to start the
	10	120	second half of a pass at the workpiece
	1	031	
	2	000	NOP
	3	000	
	.	.	.
	.	.	.
	.	.	.
	077	000	NOP
	100	041	Change the first Z-Mov direction back to
	1	062	it's original direction
	2	006	
	3	042	
	4	156	
	5	031	
	6	041	Change the next Z-Mov location
	7	062	
	10	007	
	1	042	
	2	162	
	3	031	
	4	041	Change the next Z-Mov location
	5	062	
	6	006	
	7	042	
	20	220	
	1	031	

LOCATION	OCTAL	DESCRIPTION
033 122	041	Change the next Z-Mov location
3	062	
4	007	
5	042	
6	224	
7	031	
30	041	Change the next Z-Mov location
1	062	
2	007	
3	042	
4	267	
5	031	
6	041	Change the next Z-Mov location
7	062	
40	006	
1	042	
2	303	
3	031	
4	041	Change the next Z-Mov location
5	062	
6	006	
7	042	
50	325	
1	031	
2	041	Change the next Z-Mov location
3	062	
4	006	
5	042	
6	357	
7	031	
60	041	Change the next Z-Mov location
1	062	
2	007	
3	042	
4	341	

LOCATION	OCTAL	DESCRIPTION
033 165	031	
6	041	Change the next Z-Mov location
7	062	
70	006	
1	042	
2	056	
3	032	
4	041	Change the next Z-Mov location
5	062	
6	007	
7	042	
200	062	
1	032	
2	041	Change the next Z-Mov location
3	062	
4	006	
5	042	
6	120	
7	032	
10	041	Change the next Z-Mov location
1	062	
2	007	
3	042	
4	124	
5	032	
6	041	Load the following two bytes into H and L
7	030	
20	031	
1	065	DCR M
2	302	Unless M is 0, jump to the following
3	260	location
4	033	
5	062	Turn the X motor CCW (this moves the tool
6	003	away from the workpiece after it has been
7	200	completed)

LOCATION	OCTAL	DESCRIPTION
033 230	036	MVI E
1	231	99 BCD
2	062	Turn the X motor on
3	000	
4	200	
5	315	Call X-Mov
6	010	
7	030	
40	076	MVI A
1	000	
2	270	CMP B
3	322	Jump if the X motor is still on
4	235	
5	033	
6	166	Halt
7	000	NOP
50	000	
1	000	
2	000	
3	000	
4	000	
5	000	
6	000	
7	000	
60	062	Turn X CW (Feed into the workpiece)
1	002	
2	200	
3	041	Set the software brake
4	062	
5	003	
6	042	
7	055	
70	030	
1	036	MVI E
2	100	40 BCD

LOCATION	OCTAL	DESCRIPTION
033 273	315	Call X-Mov
4	010	
5	030	
6	303	Jump back to this location (Start a new pass)
7	056	
300	031	

INTERFACING AN ENGINE LATHE AND A MICROCOMPUTER

by

BRADLEY A. KRAMER

B.S., Kansas State University, 1980

AN ABSTRACT OF A MASTER'S THESIS
submitted in partial fulfillment of the
requirements for the degree

MASTER OF SCIENCE

Department of Industrial Engineering
KANSAS STATE UNIVERSITY
Manhattan, Kansas
1981

The purpose of this research is to develop, construct, and test an interface between a microcomputer and an engine lathe. Although numerically controlled machines are available, it is impossible, at the present time, to obtain a package which will convert an ordinary lathe into a computer controlled machine. In 1980, such an interface was built for a milling machine. This system utilized DC stepper motors to control the position of the milling machine table. This research was done in the laboratories of the Department of Industrial Engineering at Kansas State University. The present project will utilize reversible AC motors and thus require different interfacing techniques. The project consists of three major areas, design, construction, and testing. The design phase consists of the development of the necessary circuitry and hardware for the interface. The second phase is to construct and debug the system designed in phase one. Finally, a machine language program is written to machine a specific part. This program tests the feasibility of the system.