MICRO-COMPUTER CONTROLLED CUTTING TABLE

DESIGN AND IMPLEMENTATION OF A MICRO-COMPUTER CONTROLLED CUTTING TABLE

by

ALFRED NORMAN ZEUNER

A thesis

Submitted to the School of Graduate Studies
in Partial Fulfillment of the Requirements
for the Degree
Master of Engineering Physics

McMaster University
May 1982

MASTER'S DEGREE (1982) McMASTER UNIVERSITY

ENGINEERING PHYSICS Hamilton, Ontario

TITLE: Design and Implementation of a Micro-computer controlled Cutting Table

AUTHOR: Alfred Norman Zeuner, B.Eng. Phys. (McMaster University)

SUPERVISOR: Dr. C. K. Campbell

NUMBER OF PAGES: viii, 80

ABSTRACT

This thesis deals with the design and implementation of a microcomputer controlled cutting table used to prepare large scale patterns for Surface Acoustic Wave (SAW) devices. This automated cutting table simplifies the creation of many SAW patterns and makes previously unattainable patterns possible. The design is extremely versatile and can cut as well as straight lines in any direction in two dimensions.

A commercially available, 44 inch square, cutting table was automated by mounting stepping motors on each of the two axis. The motion of the metors is controlled by an SDK-85 micro-computer and several peripherals.

A full description of the hardware, software and a successful experimental test is included.

TABLE OF CONTENTS

SOFTWARE

ACKNOWLEDGEMENTS

I would like to express by gratitude to all those people who helped make this project a success. First of all I would like to thank Dr. C.K. Campbell whose idea this was in the first place, and who patiently waited while I stumbled around for months before I got the thing to work. Nick Slater, whose been my friend and lab partner through much of my university career, helped a great deal with ideas and software. He was truly instrumental in the success of this project. Mark Usik, our lab's technical expert, helped me out a lot in the beginning with the hardware aspects of the project.

But most of all, I must thank my parents. They have provided me with a home, and lots of moral support throughout the many years of my education. Behind this successful man is a great family, my challenge now is to justify your contribution. Thank You.

CHAPTER 1

1.1 Introduction

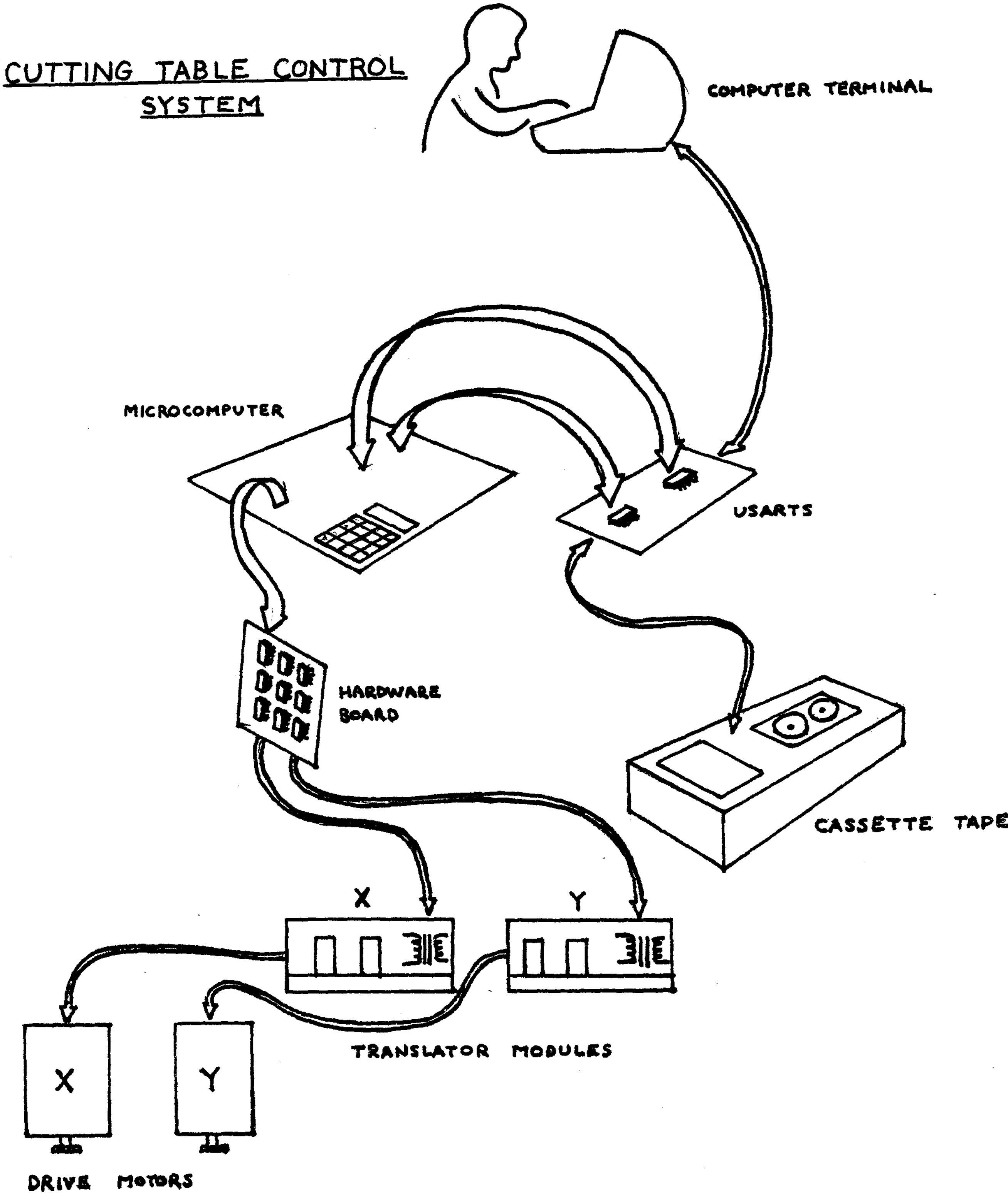
This thesis deals with the design and implementation of a microcomputer controlled cutting table used to prepare large scale patterns for Surface Acoustic Wave (SAW) devices. This automated cutting table simplifies the creation of many SAW patterns and makes previously unattainable patterns possible. The diesign is extremely versatile and can cut curves as well as straight lines in any direction in two dimensions.

A commercially available, 44 inch square, cutting table was automated by mounting stepping motors on each of the two axis. Patterns are cut into two layered plastic sheeting, trade named, "Rubylith". Once the outline of the pattern is cut the thin, red coloured, top layer is peeled off the transparent, underlying plastic where ever desired.

Since the operating characteristics of SAW devices is dependent on the geometry of the finger pattern, accuracy is important. The gearing on the table determines the overall accuracy to be .001 inch, (.0254 mm). The patterns are then normally photo-reduced 100 to 200 times, so the final product's geometry is accurate to the order of 1000 Angstroms.

FIG. 1

CUTTING TABLE CONTROL SYSTEM



1.2 Overall Discription of Operation

Figure 1 (preceeding page) is a schematic of the entire automated cutting table system.

The operator must first create a data file on the CYBER 170 (McMaster's mainframe computer) describing the pattern to be cut. The data file must simply consist of a listing of the x-y co-ordinates that the cutting table will join by straight lines, in "connect the dots" fashion. The knife position, up or down, is included as a third co-ordinate. Curves are realized simply by numerous straight lines in succession. Obviously the most expedient method of creating this data file, especially if curved lines are to be cut, is by way of a Fortran program. Once the data file is loaded, an interface program converts this "point to point" data into the cutting code used by the micro-computer system and loads it onto the micro RAM in 1000 byte chunks. Once the micro-computer's RAM is full it unloads the data onto the cassette tape deck and then signals the CYBER 170 to send the next chunk of data. After all the data has been loaded onto the cassette tape the actual cutting routine can begin, the CYBER system is no longer needed so the terminal may be shut off.

The cutting routine is started by simply executing the appropriate micro program. This program retrieves the data from the cassette tape in 1000 byte chunks and using the data, pulses the x and y stepping motors. The cutting program automatically accelerates and decelerates the motors to compensate for the momentum of the table's axis. The system will run completely automatically until the entire pattern is complete (often 3 or 4 hours depending on the size and complexity of the pattern).

Most of the remainder of this thesis deals with the design of each of the sub-systems, working back from the stepping motors and their translator modules, through the hardware that interfaces them to the micro-computer. After the micro software is discussed and documented the operation of the cassette deck and its interface circuitry is shown. Finally the thesis talks about the CYBER interface and an example of a successful creation of a rather complicated SAW pattern is given.

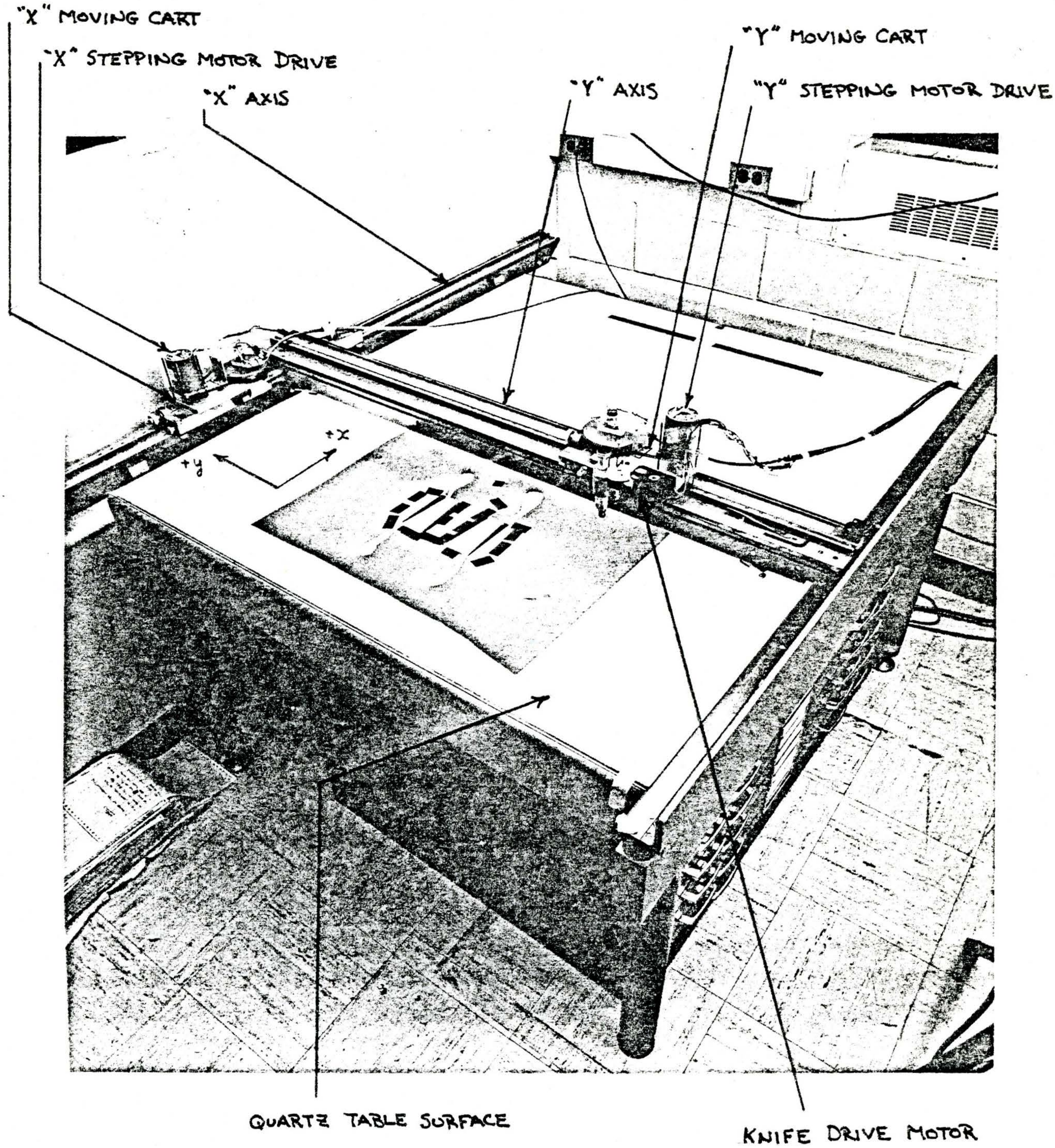


FIGURE 2 THE CUTTING TABLE

CHAPTER 2

2.1 Cutting Table Mechanics

Figure 2 on the previous page shows a top view of the cutting table and its drive motors. The knife or cutting point is carried on the "y" moving cart. The "y" moving cart runs along the "y" axis which in turn is fastened to the "x" moving cart.

2.1.1 Stepping Motors

Each moving cart is driven by a "Slo-Syn" synchronous stepping motor. Figure 3, below, is a schematic of one of the motors:

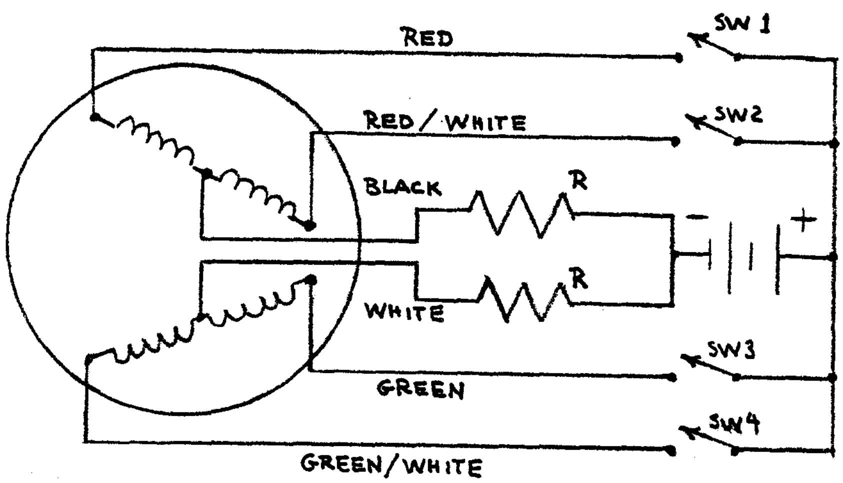


FIG. 3 STEPPING MOTOR

The motors operate by energizing one or two of the windings at a time following a specific input sequence. The motor shaft advances 200 steps per revolution (1.8^{o} per step) when a four step input sequence (full step mode) is used and 400 steps per revolution (0.9° per step) when the eight step input sequence (half step) is used.

Eight step input sequence:

Winding:	1	2	3	4
Step 1	energized	off	energized	off
2	energized	off	off	off
3	energized	off	off	energized
4	off	off	off	energized
5	off	energized	off	energized
6	off	energized	off	off
7	off	energized	energized	off
8	off	off	energized	off

The above sequence is repeated 50 times to make one complete revolution. The windings can be energized in this sequence or in the reverse order to make the motors run the other direction. The four step input sequence is the same as the eight step one if all the even steps are skipped, ie. step sequence 1, 3, 5, 7, 1, 3, etc. In full step mode the motors can operate up to 25 revolutions per second, only about 12 rev/s in half step.

2.1.2 Motor Mounting

The cutting table used for this project started life as a manually operated, imperial measurement table. It was delivered in 1966 before the advent of micro-computers and the introduction of metrification. The cutting edge was simply moved around on the "Rubylith" by the operator physically pushing either the "x" or "y" carts along their respective axis. Each axis had a measuring tape fastened to it to measure inches, each cart mounted a circular vernier scale to indicate fractions of inches. This vernier scale was driven by a gear which ran on a track mounted on each axis. The combination of track, gear and vernier scale has a specified accuracy of 1/1000 ths of an inch.

To automate the cutting table the vernier scale was replaced by a large reduction gear which is chain driven by a stepping motor. Figure 4 below is a simplified diagram of the motor mountings:

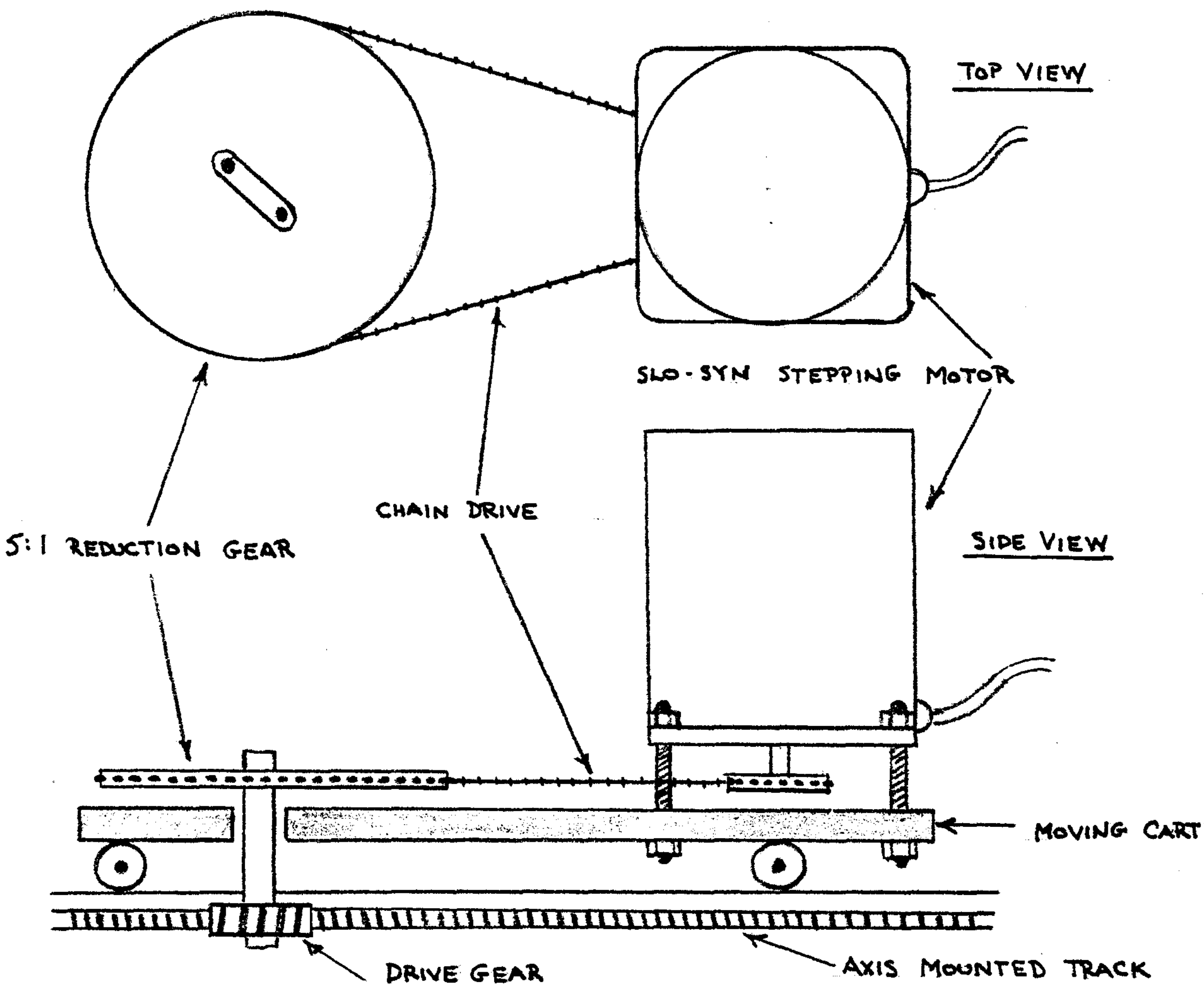


FIG. 4 MOTOR MOUNTING

A reduction ratio of 5 to 1 was chosen to make full use of the rated accuracy of the table. If the motors are operated in full step mode there are 1000 steps to the inch, in half step mode; 2000 steps/inch.

2.1.3 <u>Translator Modules</u>

Each motor winding draws almost 4 Amps so a translator module is supplied to each motor to handle the load. Also, as the name implies, it will "translate" computer commands into the desired stepping sequence, thereby allowing several operating modes.

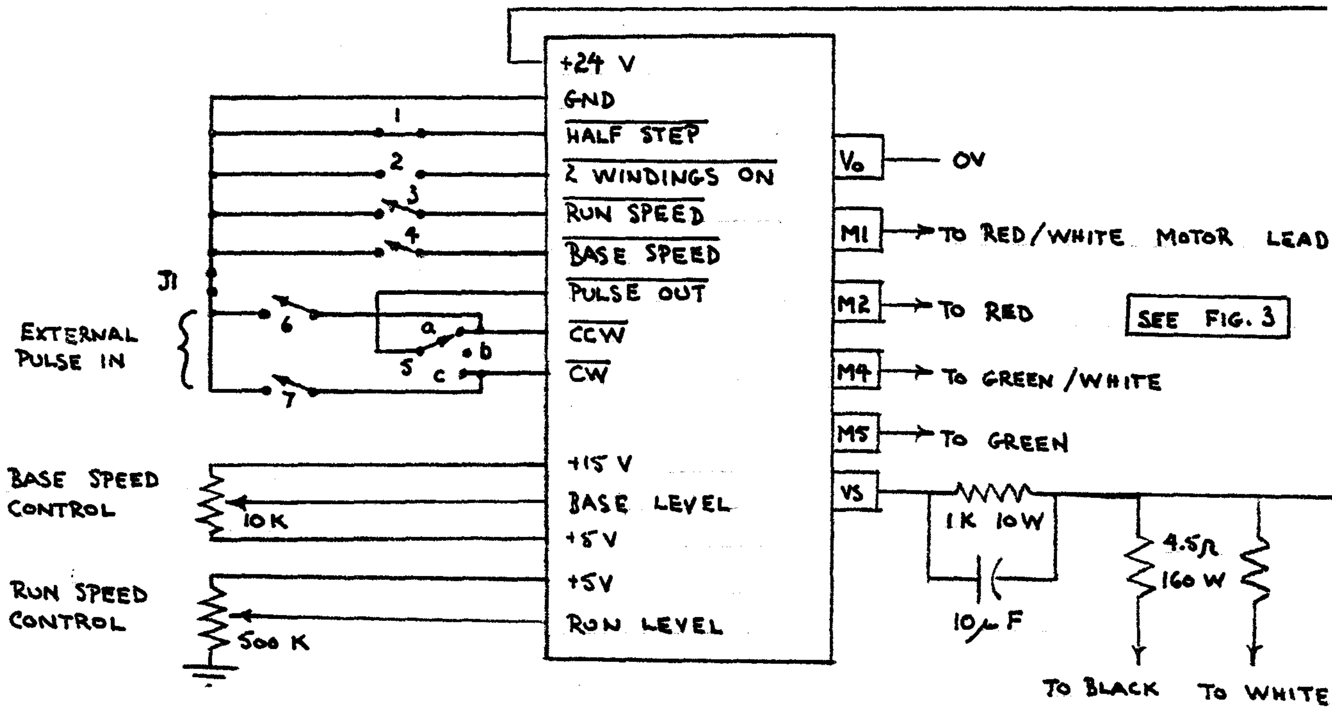


FIG. 5 STM TRANSLATOR MODULE

The speed and direction of the stepping motors is controlled by openning and closing the appropriate switches on the translator module. A counterclockwise step is realized by grounding the CCW input (closing switch 6), a clockwise step by grounding the CW input (closing switch 7). After the switch has been closed for 10 micro-seconds it can be opened and closed again to trigger another step. Either of these two switches may be toggled like this at a rate of up to 6000 Hz, which will run the motor at 15 rev/second in the half step mode. Care must be taken not to ground both the

CW and CCW inputs simultaneously as this will energize three, perhaps four motor windings. Not only will this confuse the stepping motor but it will overload the translator module.

This translator module has two internal oscillators that can be used to operate the motor. The "base speed" oscillator (selected by closing switch 4), will cause a low frequency output on the PULSE OUT pin. This will drive the motor at slow speed in the direction selected by switch 5. The motor can be run faster by switching to "run speed". The "run speed" oscillator has the additional feature of gradually accelerating/ or decelerating the motor to/or from the operating speed.

Base speed, run speed and the acceration constant can be adjusted by potentiometers on the translator module. These constants must be set giving consideration to the ineria of the load the respective motors must handle, and the torque of the motors themselves.

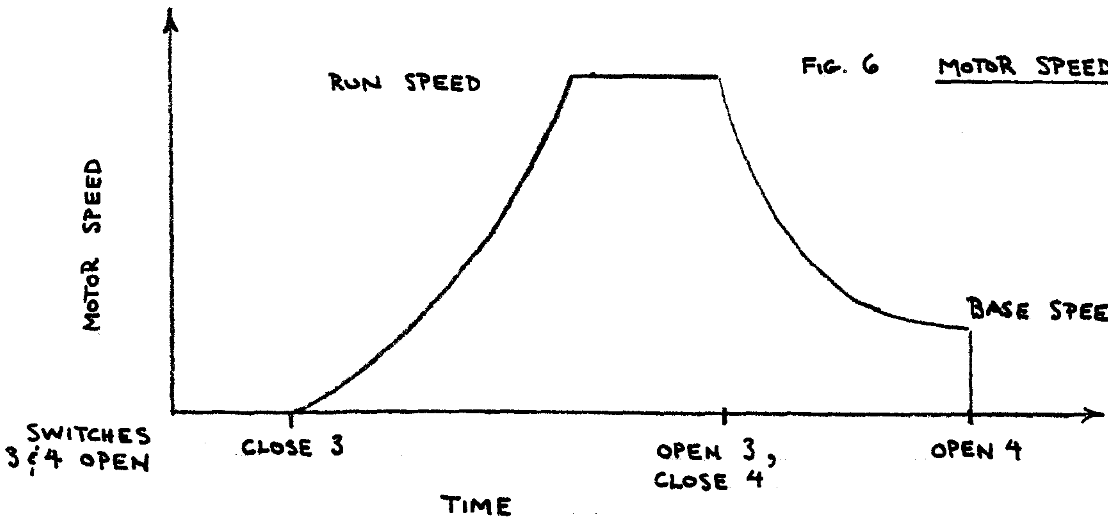


FIG. 6 MOTOR SPEED CONTROL

Before starting the motors the direction of rotation is chosen (switch 5), a micro-second, or so, later the run switch, 3, can be closed. The motor will then accelerate to the operating or "run" speed. At no time can the acceleration be such that the inertia of the motor load exceeds the motor's ability to handle it. Looking back at figure 2, it is obvious that the "x" motor has the larger load so <u>both</u> "x" and "y" motors must be adjusted for the heavier load. The reason for this is explained later in the text. A couple of thousand steps, about one inch, before the motor is to stop the translator is switched to base speed and the motor will decelerate to the slower speed (figure 6). Finally once the motor has stepped far enough, the end of the cut has been reached, switch 4 is opened and the motor abruptly stops. Here again the base speed must be chosen such that the moving carts can be brought to a halt within one step (1/2000 ths of an inch).

2.1.4 Knife Drive Motor

Since the knife must be able to cut in all directions a blade is of no use, so a pointed buret is used instead. Several tips were experimented with. A very small diameter crystal tip is probably the best but are very expensive and turned out to be virtually impossible to obtain. Sharpening the stainless steel shaft of an old, broken, buret proved successful but necessitated numerous painstaking sharpenings. Finally an ordinary sewing pin was epoxied onto the flat end of the old buret. The pin has a very narrow diameter tip and makes a good quality cut, and best of all, costs about one cent compared to $ 200 for a crystal tip. When the pin's point wears out it can simply be pulled out of the epoxy and a new one inserted.

The quartz table top is quite brittle so care must be taken to lower the knife point gently. The point itself can also be expected to survive longer if this precaution is taken.

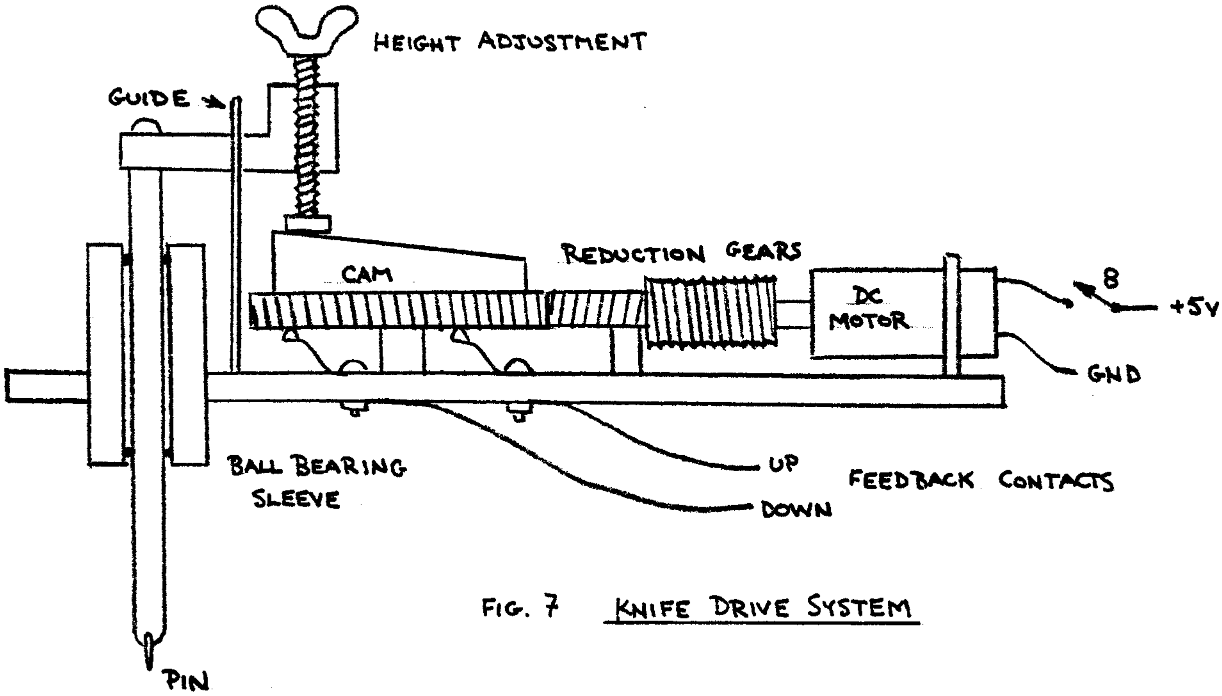


FIG. 7 KNIFE DRIVE SYSTEM

The above figure shows a cross-sectional view of the knife drive system. Basically the knife rides on a cam which can be slowly rotated by a small DC motor. The position of the knife is fedback by the feedback contacts shown. If the feedback contacts disagree with the required position of the knife, switch 8 is closed, starting up the motor and rotating the cam until the knife is in the right position.

2.2 Hardware Control Circuit

The translator module and the knife drive is controlled by opening and closing switches. These "switches" are implemented using 2N2222A switching transistors as shown in figure 8 on page 13.

To operate the motors on one of the internal oscillators either the CW or CCW (inputs 5a or 5c, figure 8) must be set HI, care must be taken not to allow both to be set HI toget-

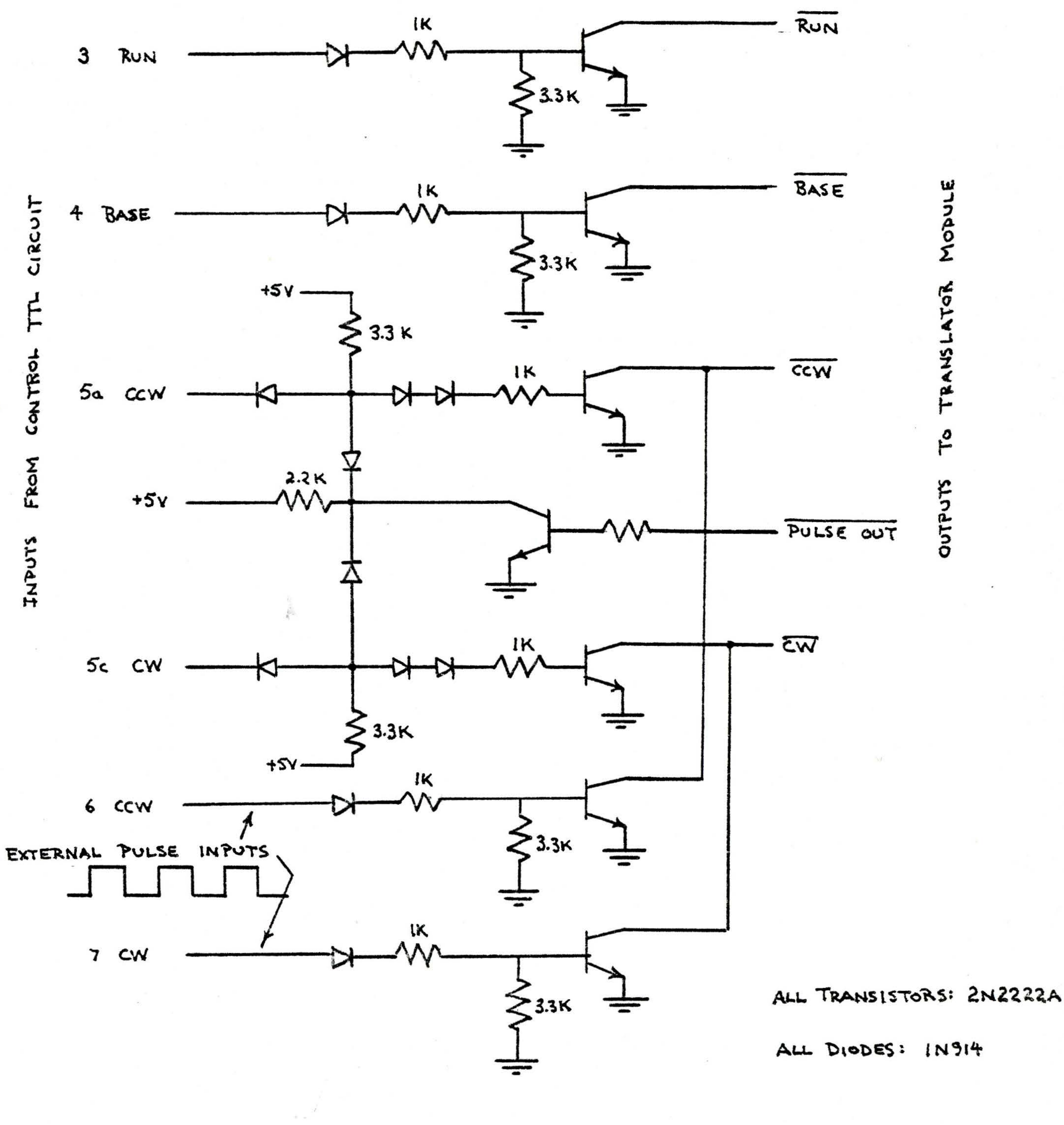


FIG 8 BUFFER SWITCHING CIRCUIT

her. BASE or RUN is set logical HI after the direction of rotation has been selected. When using an external oscillator the pulsed digital signal must be applied either to input 6 or 7, depending on the desired direction of rotation, all other inputs must be LO.

At this point a micro-computer could directly operate the drive motors, however; the computational time required was dramatically reduced by building the small TTL circuit shown in figure 9 on the next page. This circuit will always select one of the motors to run on the translator's internal oscillator, henceforth known as the "Master" motor, while the other must be operated by external pulses supplied by the micro-computer, this one is refered to as the "Slave" motor.

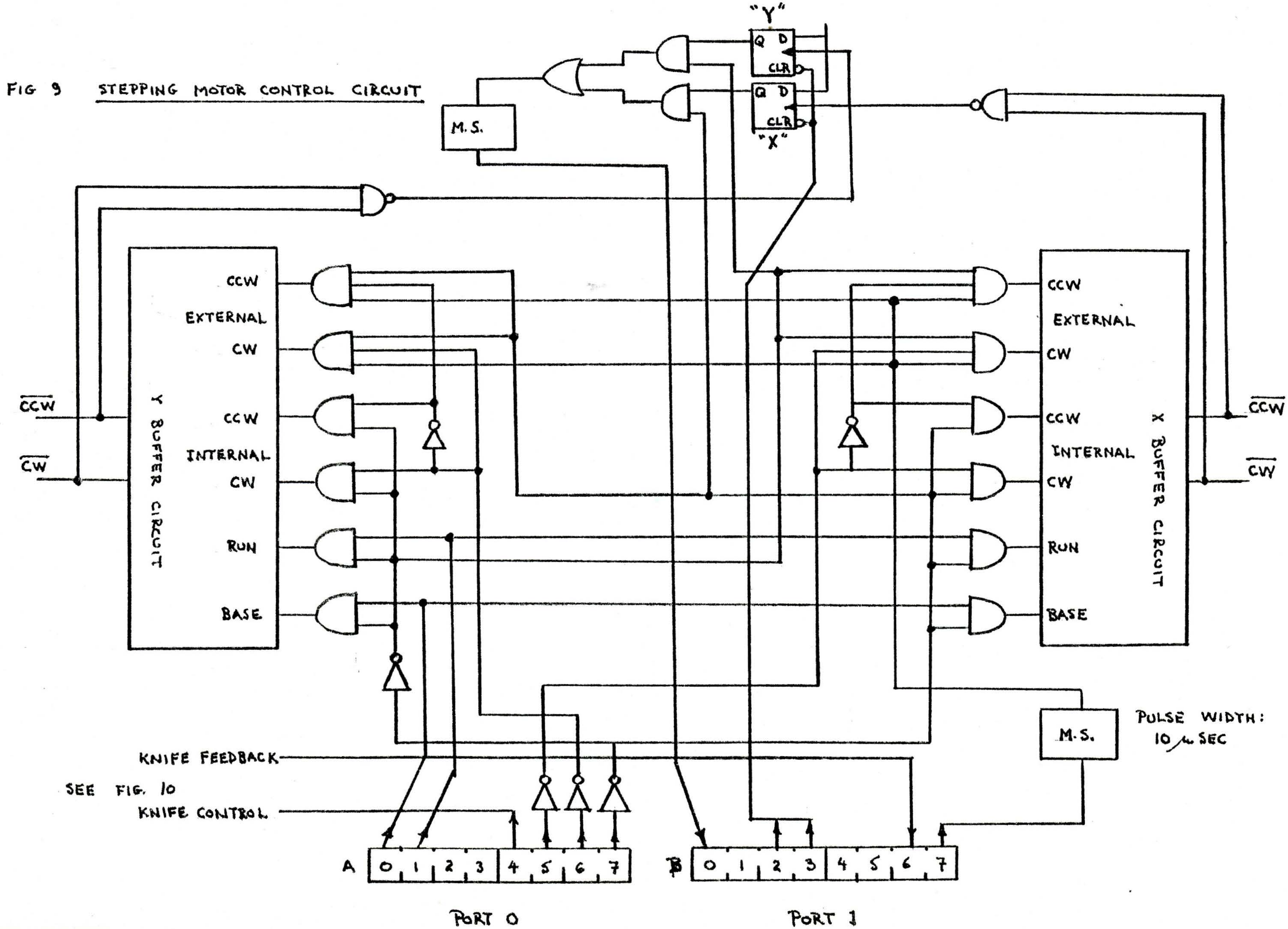
The micro-computer's ports "0" and "1" control the stepping motors via the TTL circuit. The port's bit designation is as follows:

Port 0								Port 1							
A7	A6	A5	A4	A3	A2	A1	A0	B7	B6	B5	B4	B3	B2	B1	B0

A7	1	"x" motor = master, "y" motor = slave
	0	"y" motor = master, "x" motor = slave
A6	1	"y" motor counterclockwise
	0	"y" motor clockwise
A5	1	"x" motor counterclockwise
	0	"x" motor clockwise
A4	1	knife down
	0	knife up
A3,A2		"don't care"
A1,A0	00	stop master motor
	01	master motor at "base" speed
	10	master motor at "run" speed
	11	same as "10"

Bits A0 to A7 define the operating modes of the motors and are set by the micro-computer to start the motors running, and are appropriately called the "control word". Once set,

FIG 9 STEPPING MOTOR CONTROL CIRCUIT



the control word does not change until the motors are to slow down, stop, or change direction. On the other hand bits B0 to B7 constantly interact with the motors. B0 goes HI after each motor step, once this step is recognized and counted by the micro-computer, it is acknowledged by setting bits B2 and B3 HI for a short time (about 3 micro-seconds). This allows the recognition of the next step. The micro-computer steps the slave motor by pulsing port B7. The position of the knife is fedback through B6; LO indicates that it is not in position, HI means that it is.

2.2.1 Knife Control Circuit

Figure 10 shows how the position of the knife is controlled:

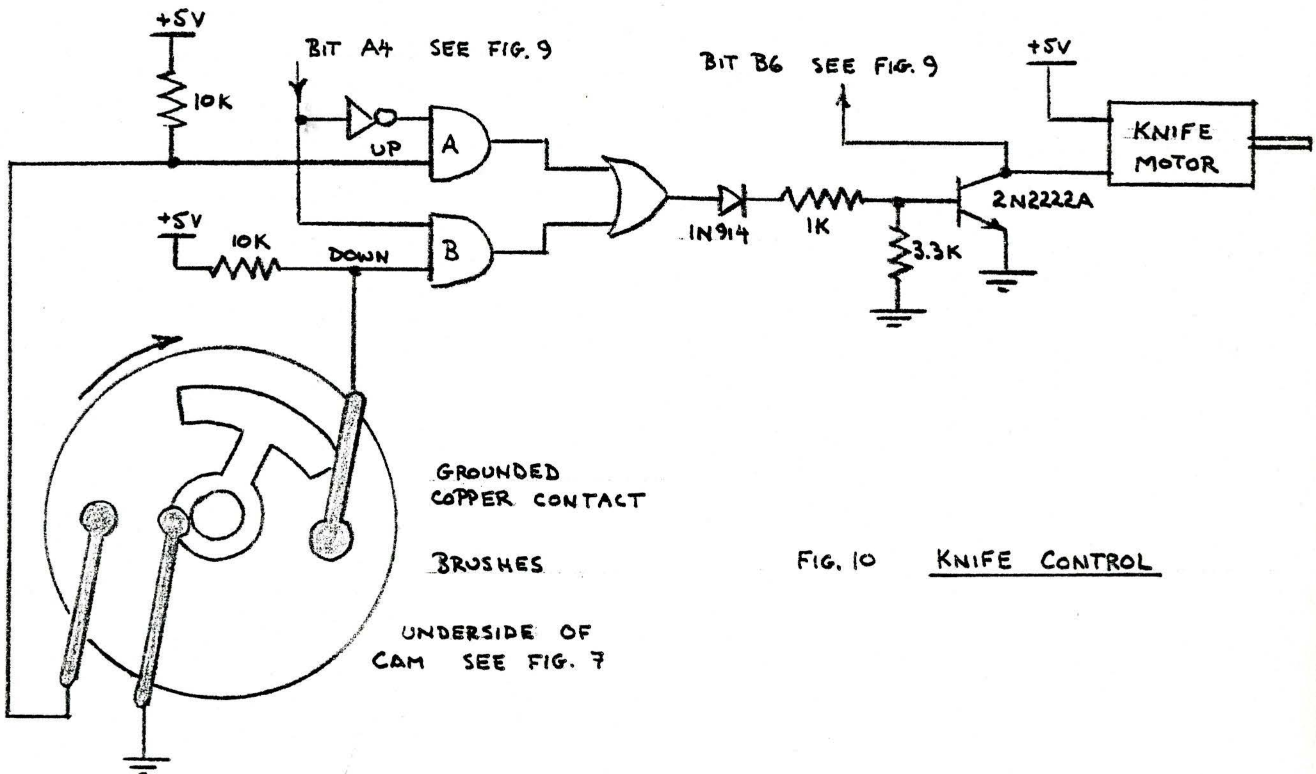


FIG. 10 KNIFE CONTROL

If the micro-computer wants the knife up it sets bit A4 LO, the output of the AND gate "A" will be HI which will start the knife motor. The cam will rotate until the "UP" feedback contact is grounded, AND gate "A" will then go LO, stopping the knife motor. To lower the knife, bit A4 is set HI and the motor will run until the "DOWN" contact stops it.

CHAPTER 3

3.1 Micro-computer Software

Cuts can be made in any direction by first choosing the master and slave motors and their direction of rotation. The master motor is then started and the number of steps it moves is counted by the software program. The slave motor is moved with pulses triggered by the micro-computer according to the slope of the desired cut. For example, if a slope of 3:1 is desired the micro-computer counts three master motor steps then triggers one pulse to the slave motor, it counts three more master motor steps, then sends another pulse to the slave motor, and so on. More complicated slopes such as 5:2 can also be cut, first three steps are counted, then a pulse to the slave motor, then two master motor steps are counted, and another pulse to the slave motor. This sequence is then repeated until the end of the cut.

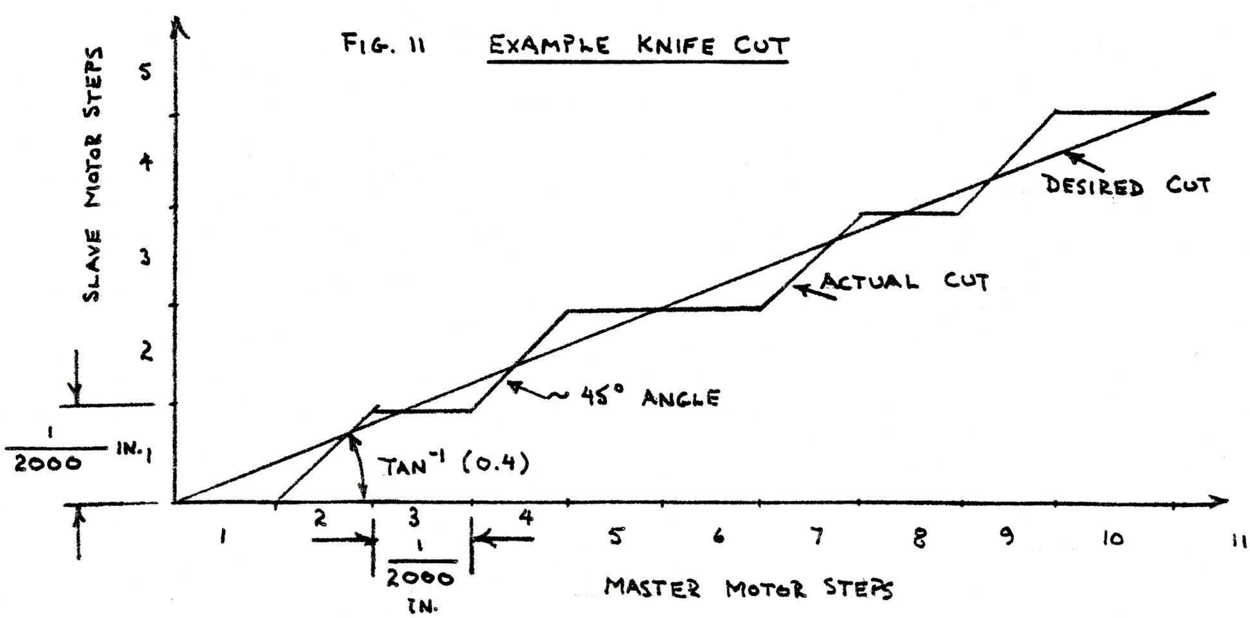
The above routine is accomplished by adding the slope of the desired cut to a register after each master motor step, then pulsing the slave motor every time an overflow is detected. This register is always initialized to 0.5 before every cut to ensure that the actual cut is as close as possible to the desired cut, see figure 11. This algorithm can be demonstrated using the 5:2 slope example from before. The following example, is shown both in decimal, for clarity, and in hexidecimal, the number system the micro-computer actually operates in.

	Decimal	Hexidecimal
Slope	.4000	6666
Initialize Reg.	.5000	8000

Master motor steps	Accumulated total			Slave motor steps
1	.9000	E666		
2	.3000	4CCC	;overflow	1
3	.7000	B332		
4	.1000	1998	;overflow	2
5	.5000	7FFE		
6	.9000	E664		
7	.3000	4CCA	;overflow	3
8	.7000	B330		
9	.1000	1996	;overflow	4
10	.5000	7FFC		
.	.	.		
.	.	.		
.	.	.		
etc	etc	etc		

Figure 11 below, shows how this algorithm is actually realized on the cutting table:

FIG. 11 EXAMPLE KNIFE CUT



The motors move on the rising edge of the oscillator's pulse and since it only takes a few micro-seconds to feedback the master motor pulse and send the slave pulse, both master and slave motors move virtually simultaneously, thus the 45^{o} angles shown in figure 11.

Obviously the largest value that can be entered as the slope is FFFF hex. This corresponds very closely to a 45^{o} slope (within 0.0055^{o}, 360/65,536). To cover all the possible angles the control digit (the upper nibble of the control word) must be chosen as shown in figure 12:

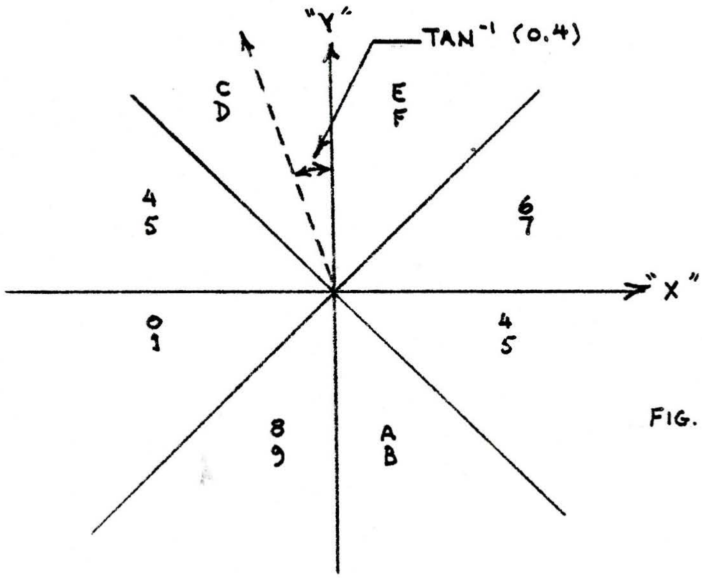


FIG. 12 CONTROL DIGIT SELECTION

If we wish to make the cut shown by the dotted line, a slope of 0.4 is used and the control digit "C" (1100) or "D" (1101) is selected. Actually to make a cut D must be used as the knife will be in the lowered position, C will raise the knife above the surface of the table. In all cases odd numbered control digits lower the knife, even ones raise the point and are used if we want to move the knife without cutting the Rubylith. The orientation of the "x" and "y" axis on the actual cutting table is shown in figure 2.

3.2 Data RAM Organization

In order to successfully complete a cut the micro-computer must be able to access data describing the slope, length and direction of the cut. The slope is a double precision value (16 bits long) ranging from 0000 hex to FFFF hex (0^o to 44.9945^o). The length of the cut is also specified by a double precision value ranging from 0000 hex (0 steps or 0 inches) to FFFF hex (65,535 steps or 32.768 in.). Note from figure 13 that the micro-computer, counts the steps taken by the master motor only, therefore the actual length of the cut will be the distence travelled by the master motor multiplied by the secant of the slope.

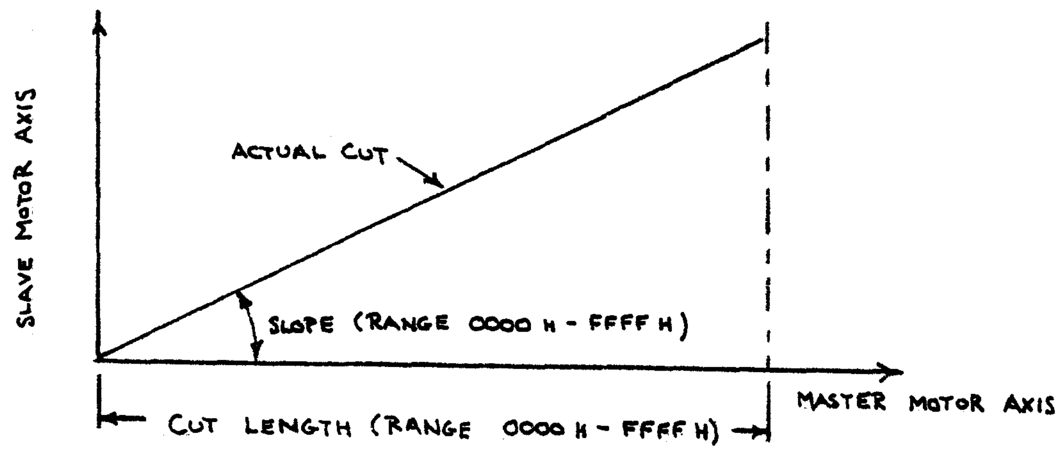


FIG 13 CUTTING DATA FORMAT

Finally the control word must be specified, as explained in Section 2.2 the control word describes the modes and directions of the motors. It also specifies the speed of the master motor and the position of the knife. Bits A2 and A3 are "don't cares" as far as the hardware control circuit is concerned, however they are used within the software. Bit A2 is 1 if the cut is part of a curve, thus telling the micro-computer to use the curve cutting program, if A2 is zero a straight line is to be cut and the appropriate program will be used. Bit A3 indicates the last word of data in the data

RAM. If A3 is 1, more data must be sought from the cassette tape and cutting must be suspended until the data transfer can be completed. Each cut requires five, 8 bit words of data and there are 1024 RAM locations (locations 5000 to 53FF) for data storage, therefore 204 cuts can be stored. Actually only 200 cuts are stored in RAM to use a more convenient figure.

Location:	Data:	Comments:
5000	most sig. 8 bits of slope	first cut
01	least sig. 8 bits of slope	
02	most sig. 8 bits of length	
03	least sig. 8 bits of length	
04	control word, bit A3 = 0	
05	.	second cut
06	.	
07	.	
08	.	
09	.	
.	.	
.	.	
.	.	
53E7	most sig. 8 bits of slope	200 th cut
E8	least sig. 8 bits of slope	
E9	most sig. 8 bits of length	
EA	least sig. 8 bits of length	
EB	control word, bit A3 = 1	

3.3 Loading Data from Data RAM into Micro-computer Registers

Before commencing each cut a short subroutine, "Load Data", moves the required five words of data from the data RAM into the micro-computer's registers, as follows:

Register pair	D-E	←	Slope of the cut
	B-C	←	Length of the cut
Register	A	←	Control word
Locations	30F0 & 30F1	←	Location of data for the next cut

The subroutine "Load Data" ignores cuts with a length of zero steps and will simply load the data for the next cut instead.

Subroutine: "LOAD DATA"

```
4A00  2A F0 30   LHLD 30F0 and F1   ;load H-L with loc. of data
  03  23         INX H-L            ;loc. of first word of data
  04  56         MOV D,M            ;most sig. word of slope to D
  05  23         INX H-L            ;loc of 2nd word of data
  06  5E         MOV E,M            ;least sig. word of slope to E
  07  23         INX H-L            ;
  08  46         MOV B,M            ;most sig. word of length to B
  09  23         INX H-L            ;
  0A  4E         MOV C,M            ;least sig. word of length to C
  0B  23         INX H-L            ;
  0C  78         MOV A,B            ;
  0D  FE 00      CPI $00            ;most sig. word of length = 0?
  0F  C2 1B 4A   JNZ 4A1B           ;
  12  79         MOV A,C            ;
  13  FE 00      CPI $00            ;least sig. word of length = 0?
  15  C2 1B 4A   JNZ 4A1B           ;
  18  C3 03 4A   JMP 4A03           ;load next data if length = 0
  1B  7E         MOV A,M            ;control word to register A
  1C  22 F0 30   SHLD 30F0 and F1   ;save location of next data
  1F  C9         RET
```

3.4 Running the Motors

Basically there are two types of cuts which must be made; straight cuts and curved cuts. Essentially a curved cut is just a series of short straight cuts whose slopes are equal to the tangent of the curve at various intervals, as in figure 15:

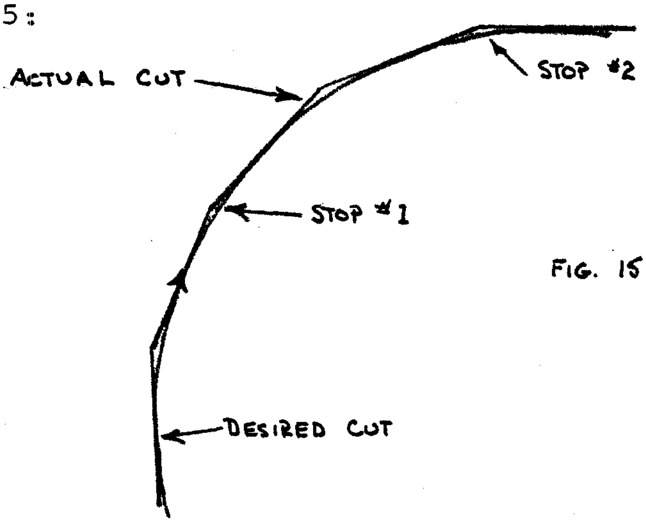


FIG. 15 CURVE CUTTING APPROXIMATION

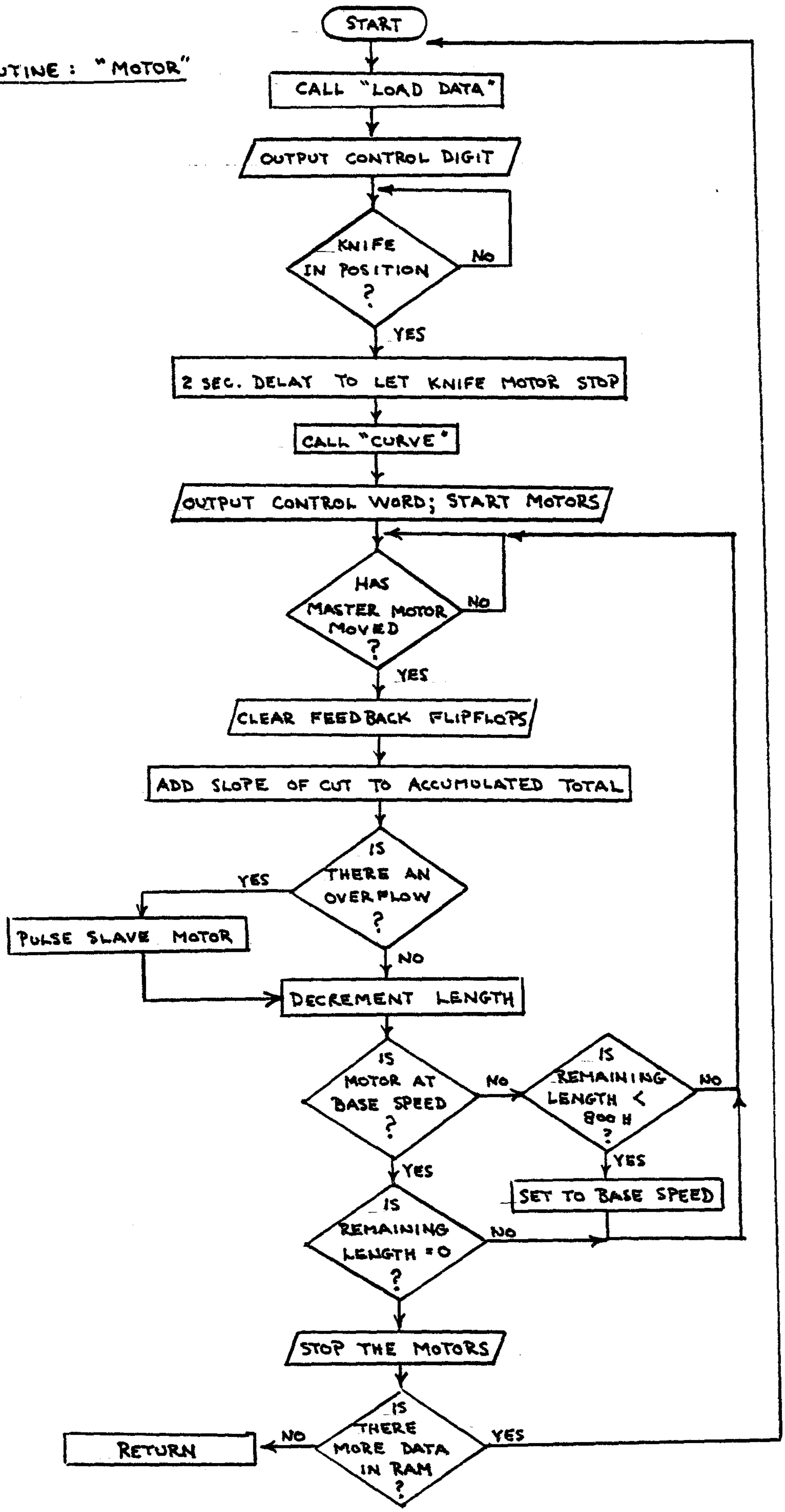
When the micro-computer is making a straight cut, it accelerates the motors to "run" speed, cuts along at this speed until there is only 2000 steps left where it decelerates to base speed. By the time the end of the cut is reached the motors are moving slowly enough that they can be stopped instaneously. Once the motors are stopped the data for the next line is retrieved from RAM, the necessary adjustments are made to the knife position and the translator modules and the process may start again for the next cut. A flowchart, describing subroutine "Motor", shows how this is implemented in software is shown on the next page.

If the subroutine "Motor" was used to cut all the short segments that make up a curve, the motors would stop every time the slope of the cut changed slightly. This would cause a very slow and jerky operation. Instead, the subroutine "Curve" is used (flowchart on page 26). This subroutine does not slow down or stop the motors after each cut, rather it simply quickly calls up the next set of data and proceeds immediately with the execution of the next cut. However, even during the cutting of curves the motors must stop if the motors must change modes or direction. If the tangent to the curve passes through the slopes 45°, 135°, 225°, or 315° the motors must stop to exchange which one is the slave and which the master motor (stop 1, figure 15). At slopes 0°, 90°, 180°, or 270° one of the motors must change direction (stop 2, figure 15). If the curvature of the cut is so small that the slope change between successive segments is greater than 20°, the motors must also slow down and stop to avoid overloading the torque capability of the motors.

The micro-computer decides whether to use subroutine

FIG 16

SUBROUTINE : "MOTOR"



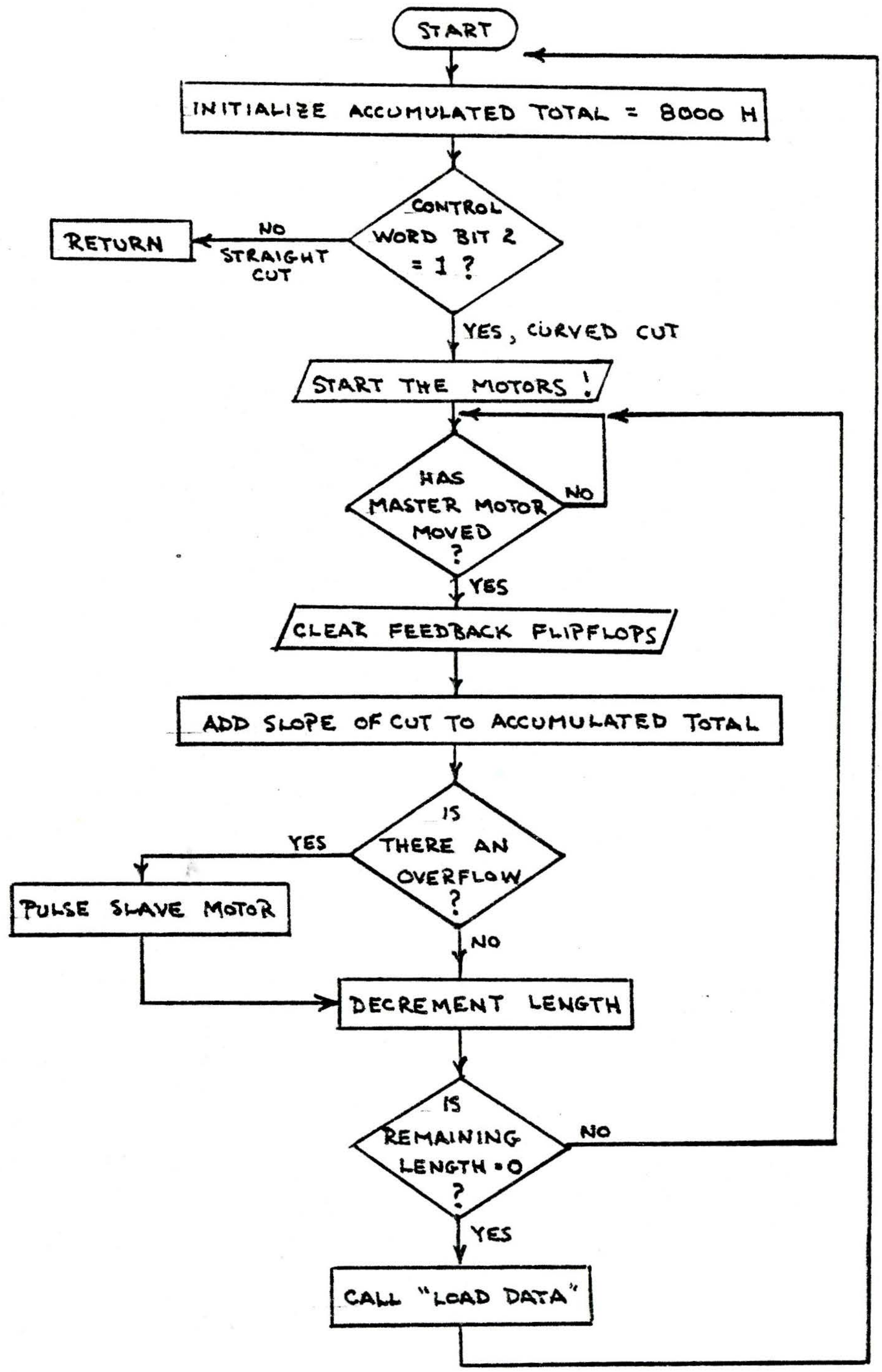


FIG. 17 SUBROUTINE "CURVE"

"Motor" or "Curve" by testing bit 2 of the control word. If bit 2 is a "1" the cut is a segment of a curve and the motors are not to stop after the completion of the segment, subroutine "Curve" is used. If bit 2 is "0" a straight cut is required and the motors are to decelerate and stop at the end of the cut, use subroutine "Motor".

The last cut described in the data RAM will always use the "Motor" routine, so after the execution of this subroutine bit 3 of the control word is tested to see if the last available data has indeed been used. If the test returns a "0" there is still more data, which is loaded and the subroutine loops around to the beginning to make the next cut. If a "1" is found the routine is suspended (returns to the mainline program) until new data is loaded from cassette tape.

Subroutine "MOTOR"

```
4AFA   21 FF 4F    LXI H-L 4FFF         ;
  FD   22 F4 30    SHLD 30F4            ;save location of 1st data
4B00   22 F0 30    SHLD 30F0            ;   "        "       "   "     "
  03   CD 00 4A    CALL "LOAD DATA"     ;load reg. with cut data
  06   E6 F0       ANI $F0              ;mask lower byte control word
  08   D3 00       OUT PORT 00          ;output control digit, set
                                         modes and direction of motors
  0A   DB 01       IN PORT 01           ;
  0C   07          RLC                  ;
  0D   07          RLC                  ;check bit B6, port 01
  0E   D2 0A 4B    JNC 4B0A             ;jump if knife not stopped
  11   CD E4 4B    CALL "DELAY"         ;
  14   CD E4 4B    CALL "DELAY"         ;wait 2 seconds
  17   CD 90 4B    CALL "CURVE"         ;check if this cut is the
                                         start of a curve; yes, use
                                         "CURVE"; no, return
  1A   7E          MOV A,M              ;control word to reg. A
  1B   D3 00       OUT PORT 00          ;start motors
  1D   3E 0C       MVI A, $0C           ;
  1F   D3 01       OUT PORT 01          ;enable feedback flipflops
  21   DB 01       IN PORT 01           ;
  23   0F          RRC                  ;check bit B1, port 01
  24   D2 21 4B    JNC 4B21             ;wait for master motor to move
```

```
4A27  3E 00       MVI A, $00        ;
  29  D3 01       OUT PORT 01       ;clear feedback flipflops
  2B  2A F2 30    LHLD 30F2 and F3  ;accumulated total to H-L
  2E  19          DAD D-E           ;add slope to accumulated total
  2F  22 F2 30    SHLD 30F2 and F3  ;store new accumulated total
  32  D2 3C 4B    JNC 4B3C          ;jump if no overflow
  35  3E 80       MVI A, $80        ;
  37  D3 01       OUT PORT 01       ;pulse slave motor
  39  00 00 00    NOP NOP NOP       ;
  3C  0B          DCX B-C           ;decrement length
  3D  2A F0 30    LHLD 30F0 and F1  ;
  40  7E          MOV A,M           ;control word to reg. A
  41  0F          RRC               ;test bit 0 of control word
  42  DA 53 4B    JC 4B53           ;jump if at base speed
  45  78          MOV A,B           ;
  46  FE 07       CPI $07           ;
  48  D2 1D 4B    JNC 4B1D          ;length > 7FF H, continue
                                     at run speed
  4B  7E          MOV A,M           ;control word to reg. A
  4C  3D          DEC A             ;
  4D  D3 00       OUT PORT 00       ;slow to base speed
  4F  77          MOV M,A           ;store new control word
  50  C3 1D 4B    JMP 4B1D          ;continue at base speed
  53  78          MOV A,B           ;
  54  FE 00       CPI $00           ;
  56  C2 1D 4B    JNZ 4B1D          ;jump if length not 0 yet
  59  79          MOV A,C           ;
  5A  FE 00       CPI $00           ;
  5C  C2 1D 4B    JNZ 4B1D          ;jump if length not 0 yet
  5F  7E          MOV A,M           ;
  60  3D          DEC A             ;
  61  D3 00       OUT PORT 00       ;length now 0, stop motors
  63  CD 00 54    CALL LOC 5400     ;this is a jump to RAM to
                                     allow testing of new pro-
                                     gram additions
  66  2A F0 30    LHLD 30F0 and F1  ;
  69  7E          MOV A,M           ;control word to reg. A
  6A  0F          RRC               ;
  6B  0F          RRC               ;
  6C  0F          RRC               ;
  6D  0F          RRC               ;
  6E  D2 03 4B    JNC 4B03          ;continue if there is more
                                     data in the data RAM
  71  C9          RET               ;no more data, return
```

Subroutine: "CURVE"

```
4B90  21 00 80   LXI H-L $8000       ;
  93  22 F2 30   SHLD 30F2 and F3    ;initialize accumulated
                                      total to 8000 hex
  96  2A F0 30   LHLD 30F2 and F3    ;
  99  7E         MOV A,M             ;control word to reg. A
  9A  0F         RRC                 ;
  9B  0F         RRC                 ;
  9C  0F         RRC                 ;check bit 2 of control word
  9D  D0         RNC                 ;return if bit 2 is zero
  9E  7E         MOV A,M             ;
  9F  D3 00      OUT PORT 00         ;start motors
  A1  3E 0C      MVI A, $0C          ;
  A3  D3 01      OUT PORT 01         ;enable feedback flipflops
  A5  DB 01      IN PORT 01          ;
  A7  0F         RRC                 ;check bit B0 port 1
  A8  D2 A5 4B   JNC 4BA5            ;wait for master motor to move
  AB  3E 00      MOV A, $00          ;
  AD  D3 01      OUT PORT 01         ;clear feedback flipflops
  AF  2A F2 30   LHLD 30F2 and F3    ;accumulated total to H-L
  B2  19         DAD D-E             ;add slope to accumulated total
  B3  22 F2 30   SHLD 30F2 and F3    ;store new accumulated total
  B6  D2 C0 4B   JNC 4BC0            ;jump if no overflow
  B9  3E 80      MVI A, $80          ;
  BB  D3 01      OUT PORT 01         ;pulse slave motor
  BD  00 00 00   NOP NOP NOP         ;
  C0  0B         DCX B-C             ;decrement length
  C1  78         MOV A,B             ;
  C2  FE 00      CPI $00             ;
  C4  C2 A1 4B   JNZ 4BA1            ;jump if length not 0 yet
  C7  79         MOV A,C             ;
  C8  FE 00      CPI $00             ;
  CA  C2 A1 4B   JNZ 4BA1            ;jump if length not 0 yet
  CD  CD 00 4A   CALL "LOAD DATA"    ;load registers with next
                                      cut data
  D0  C3 90 4B   JMP 4B90            ;jump to start and cut next
                                      segment in curve
```

Subroutine: "DELAY"

This subroutine uses a monitor routine called "DELAY" at location 05F1 hex which simply decrements register pair

D-E until it detects zero, then returns. For details on this routine see "SDK - 85 User's Manual, Appendix A, page 36.

Subroutine: "DELAY"

```
4BE4  EB         XCHG            ;save D-E in H-L
  E5  11 FF FF   LXI D-E $FFFF   ;set delay time, about 1 sec
  E8  CD F1 05   CALL "DELAY"    ;call monitor routine DELAY
  EB  EB         XCHG            ;return D-E from H-L
  EC C9          RET             ;1 sec delay complete
```

3.5 Adjusting the Data RAM

The previous section carefully explained how the lower four bits of the control word tell the micro-computer how fast the motors should run, and when and when not to stop the motors. After a block of cut data is loaded from the cassette tape and before the cuts are executed a software subroutine, "Adjust", adjusts the lower nibble of all the control words according to the following set of rules. This ensures correct motor operation once cutting starts.

1. Bit 2 is set to "1" (segment part of a curve) if:

   a. The control digit for the succeeding segment is identical to the control digit for this line, this ensures that neither motor has to change direction or exchange master and slave status, nor does the knife have to move.

   AND

   b. The change in slope does not exceed 2000 hex (about 6°), this ensures that the slave motor does not have to change speed too abruptly causing inertia problems with the table's axis arms.

   AND

   c. This segment is NOT the last cut for which there is data available in the data RAM.

   Otherwise bit 2 must be set to 0 (straight cut).

2. Bit 1 is set to "1" (motors to run at run speed) if:

   a. Bit 0, which sets the motors to base speed, is "0". This condition, although good practice, is not absolutely necessary as the motors will run at run speed if both bits 0 and 1 are set to "1", see page 14.

   AND

   b. There are more than 7FF hex steps in the cut, ie. the cut is longer than one inch.

   OR

   c. There are less than 800 hex steps in the cut but bit 2 is equal to "1" AND there are more than 7FF hex steps before the motors must stop. For example, when cutting curves consisting of segments less than 800 hex steps (one inch) long the motors are to run at run speed until there are less than 800 hex steps left before the control digit must change and the motors stop.

Otherwise bit 1 is set to "0" AND bit 0, which sets the motors to base speed, is set to "1".

Subroutine: "ADJUST"

```
4A23  01 00 50    LXI B-C 5000     ;set B-C to 1st data RAM loc.
  26  03          INX B-C          ;
  27  03          INX B-C          ;
  28  03          INX B-C          ;
  29  03          INX B-C          ;control word loc. in B-C
  2A  0A          LDAX B-C         ;control word into reg. A
  2B  E6 08       ANI $08          ;mask all but bit 3
  2D  FE 08       CPI $08          ;
  2F  CA 36 4A    JZ 4A36          ;jump if last control word
  32  03          INX B-C          ;
  33  C3 26 4A    JMP 4A36         ;continue until last con-
                                    trol word found
  36  79          MOV A,C          ;
  37  32 F6 30    STA 30F6         ;
  3A  78          MOV A,B          ;
  3B  32 F7 30    STA 30F7         ;store location of control word
  3E  0A          LDAX B-C         ;
  3F  32 F8 30    STA 30F8         ;store control word
  42  0B          DCX B-C          ;
```

```
4A43  0A          LDAX B-C           ;
  44  5F          MOV E,A            ;
  45  0B          DCX B-C            ;
  46  0A          LDAX B-C           ;
  47  57          MOV D,A            ;load length in D-E
  48  0B          DCX B-C            ;
  49  0B          DCX B-C            ;
  4A  0A          LDAX B-C           ;
  4B  32 F9 30    STA 30F9           ;store upper byte of slope
  4E  7A          MOV A,D            ;
  4F  FE 07       CPI $07            ;check if length > 800 hex
  51  2A F6 30    LHLD 30F6 and F7   ;control word loc. to H-L
  54  3A F8 30    LDA 30F8           ;control word to reg. A
  57  DA 60 4A    JC 4A60            ;jump if length < 800 hex
  5A  F6 02       ORI $02            ;adjust control word for
                                      run speed
  5C  77          MOV M,A            ;adjusted control word
                                      to data RAM
  5D  C3 63 4A    JMP 4A63           ;
  60  F6 01       ORI $01            ;adjust control word for
                                      base speed
  62  77          MOV M,A            ;adjusted control word to
                                      data RAM
  63  79          MOV A,C            ;
  64  FE 00       CPI $00            ;
  66  C2 6F 4A    JNZ 4A6F           ;
  69  78          MOV A,B            ;
  6A  FE 50       CPI $50            ;
  6C  C8          RNC                ;return if adjustments done
  6D  00 00       NOP NOP            ;
  6F  21 F8 30    LXI H-L 30F8       ;
  72  7E          MOV A,M            ;control word to reg. A
  73  E6 F0       ANI $F0            ;mask lower nibble
  75  77          MOV M,A            ;
  76  0B          DCX B-C            ;
  77  0A          LDAX B-C           ;preceeding control word
                                      in reg. A
  78  BE          CMP M              ;compare the 2 successive
                                      control words
  79  C2 36 4A    JNZ 4A36           ;jump if control words
                                      not identical
  7C  79          MOV A,C            ;
  7D  32 F6 30    STA 30F6           ;
  80  78          MOV A,B            ;
```

```
4A81  32 F7 30    STA 30F7         ;store loc. of control word
  84  0A          LDAX B-C         ;
  85  32 F8 30    STA 30F8         ;store control word
  88  EB          XCHG             ;length into H-L
  89  0B          DCX B-C          ;
  8A  0A          LDAX B-C         ;
  8B  5F          MOV E,A          ;
  8C  0B          DCX B-C          ;
  8D  0A          LDAX B-C         ;
  8E  57          MOV D,A          ;preceeding length to D-E
  8F  19          DAD D-E          ;add this length to total
                                    length in H-L
  90  22 FA 30    SHLD 30FA        ;store total length
  93  0B          DCX B-C          ;loc. of lower byte of slope
  94  0B          DCX B-C          ;loc. of upper byte of slope
  95  21 F9 30    LXI H-L 30F9     ;
  98  0A          LDAX B-C         ;
  99  32 FC 30    STA 30FC         ;
  9C  96          SUB M            ;
  9D  D2 A5 4A    JNC 4AA5         ;
  A0  7E          MOV A,M          ;
  A1  21 FC 30    LXI H-L 30FC     ;
  A4  96          SUB M            ;
  A5  FE 20       CPI $20          ;
  A7  0A          LDAX B-C         ;
  A8  32 F9 30    STA 30F9         ;
  AB  D2 4E 4A    JNC 4A4E         ;two successive slopes are
                                    compared, if slopes dif-
                                    fer by more than 2000 hex,
                                    jump
  AE  2A FA 30    LDA 30FA         ;
  B1  EB          XCHG             ;load total length in D-E
  B2  7A          MOV A,D          ;
  B3  FE 07       CPI $ 07         ;
  B5  2A F6 30    LHLD 30F6        ;loc. of control word in H-L
  B8  3A F8 30    LDA 30F8         ;control word in reg. A
  BB  DA C4 4A    JC 4AC4          ;jump if length < 800 Hex
  BE  F6 06       ORI $06          ;set bit 2 = "1", run speed
  C0  77          MOV M,A          ;adjusted control word
                                    to data RAM
  C1  C3 63 4A    JMP 4A63         ;
  C4  F6 05       ORI $05          ;set bit 2 = "0", base speed
  C6  77          MOV M,A          ;adjusted control word
                                    to data RAM
  C7  C3 63 4A    JMP 4A63         ;
```

CHAPTER 4

As already mentioned there is only space available in RAM to load the data for 200 cuts. Several of the masks that have been constructed have required in excess of 200 cuts, curves especially use up a lot of data. A cassette tape is used to store the data before transfering it in 200 cut chunks to the data RAM. The cassette tape, being a magnetic memory, is also useful to permanently store masks to be used over and over.

This chapter discusses the micro-computer/cassette interface and the associated software. John Metselaar did a great deal of work in this area, which he explains in his thesis, "Micro-computer Interfacing, Design, and Operation". He built an interfacing circuit, called PRO-3, which is used in its original form in this thesis. This circuit will be discussed briefly in this chapter, for completeness, however the details of the construction and operation are quite ably explained in Mr. Metselaar's thesis. The software, on the otherhand, was changed completely and is carefully covered in this chapter.

4.1 The "PHI" Cassette Deck

A PHI-DECK cassette recorder is used to store the cutting data. This is a digitally controlled cassette tape recorder which is TTL compatible. It has the four standard

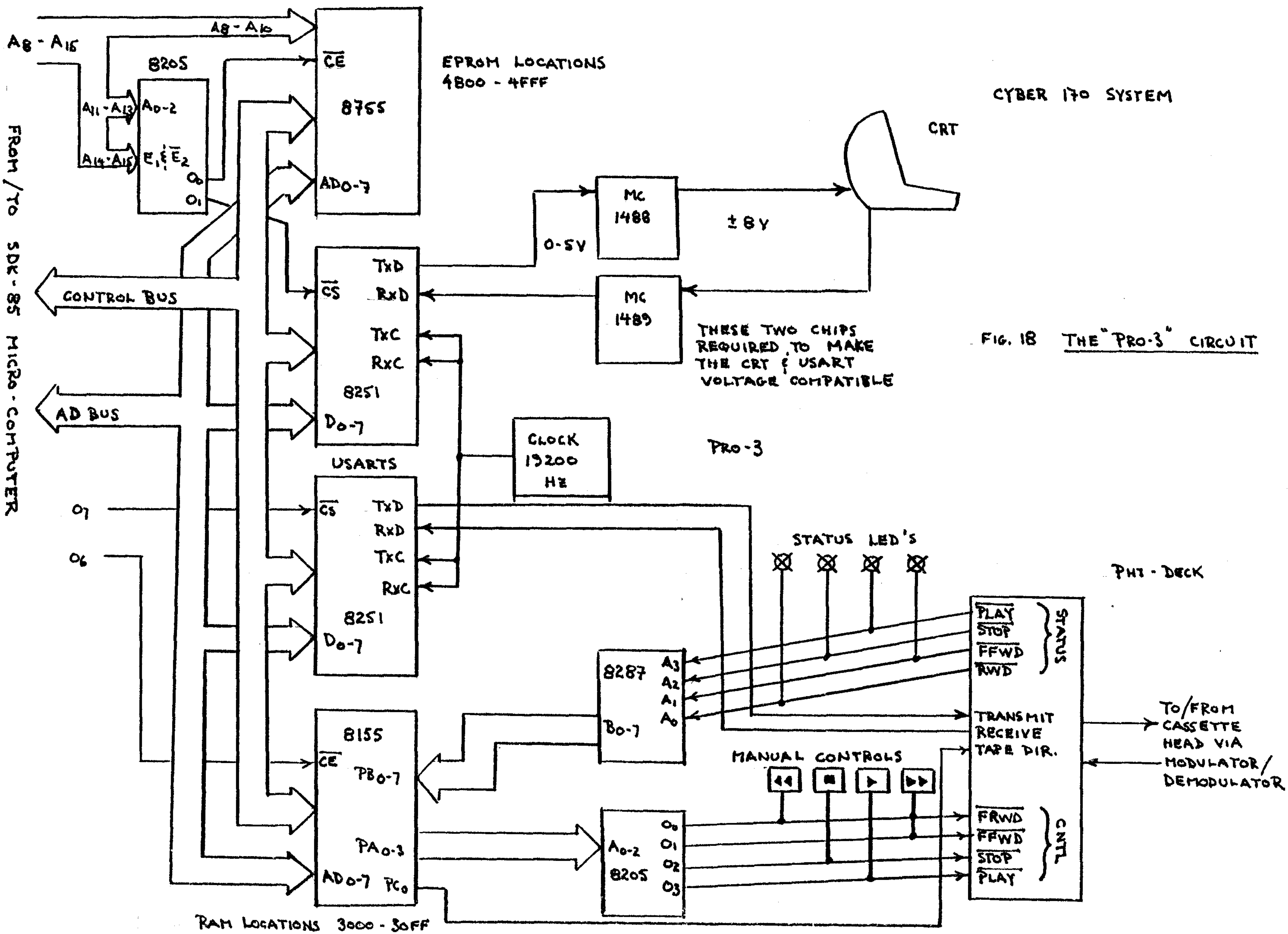


FIG. 18 THE "PRO-3" CIRCUIT

controls which are self explanatory: Play, Stop, Fast Forward, Fast Rewind. All of these controls are triggered by and inverse, 1 micro-second, pulse which can be sent from the micro-computer via an output port or from a manual control panel mounted on the PHI-DECK. See figure 18, on the preceeding page. There are also four associated status LEDs which indicate whether the tape deck is on "play", or "stopped", or "rewinding", or "fast forwarding". There are also the same status indicators on the PRO-3 circuit and the PRO-3 feeds these status signals back to the micr-computer via some input ports. A capstan clock is also provided which operates a digital counter on the control panel and is used to help locate data blocks on the tape. This counter is not hooked up to the micro-computer and can only be used manually.

The actual recording and playing back from the tape deck is accomplished via a frequency shift keying circuit already provided with the PHI-DECK. In order to record data on the tape the "record/playback" switch must be in the record position and the deck must be in the "play" mode. A serial string of logical "1's" and "0's" applied to the receive input to the tape deck will be modulated such that a "1" will be recorded on tape as a 6 KHz signal and a "0" as a 5 KHz signal:

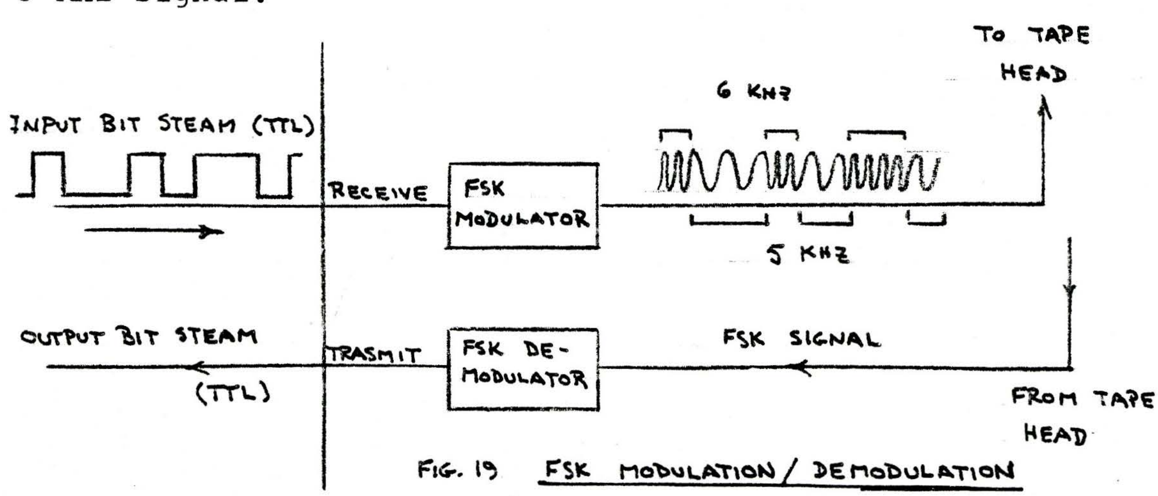


FIG. 19 FSK MODULATION/DEMODULATION

To get the data back off the tape, the tape is simply rewound to the start of the data, the record/playback toggle is flipped to "playback", and the deck set to "play". The FSK signal on the tape will be picked up by the tape head and routed through the demodulator. The demodulator will re-create the inputed bit stream and output it along the "transmit" line.

There are two more controls on the tape deck, both of them potentiometers: tape gain and tape speed. The tape speed adjustment is self explanatory, however care must be taken to playback data at close to the same speed at which it was recorded. Playing back at a different speed will change the frequencies of the FSK signal and it will alter the bit rate at the output. Both the demodulator and the "serial to parallel data receiver" (the USART) can tolerate a certain amount of deviation, but to be on the safe side care must be taken not to adjust the speed control. The "gain" adjustment is normally set to half-way between the lower and upper extremes.

4.2 The USART 8251

USART is an acronym for the INTEL Universal Synchronous/ Asynchronous Receiver/Transmitter and provides the interface between the micro-computer and the PHI-DECK. It also provides part of the interface between the micro-computer and the CRT terminal, which will be covered in the next section.

Data is loaded onto the magnetic tape, and is subsequently received from the tape, in a serial format. The micro-computer however processes the data in 8 bit parallel, the USART's purpose is to make the conversion and to frame each byte so that the data can be sorted out upon recovery from the cassette tape.

Serial data can be framed in two ways: it can be framed by time, meaning that a synchronizing clock signal must accompany the bits and bytes coming off the tape. This clock signal shifts the data into the receiving buffer counting the pulses. Once the count reaches the word length the word is shifted to the micro-computer and the next word is loaded from tape. This is synchronous mode. The PHI-DECK has no way of outputting a synchronous clock with the data so the USART must be used in an asynchronous mode. In this mode each word or byte gets a start and stop bit added to it before it is loaded onto the tape; bit framing. When retrieving the data the USART looks for the start bit and then clocks in the number of expected bits using a receiver generated clock pulse, see figure 18. The USART then looks for the stop bit followed by the next start bit, at which time the clock is re-aligned with the data and the next word is clocked in.

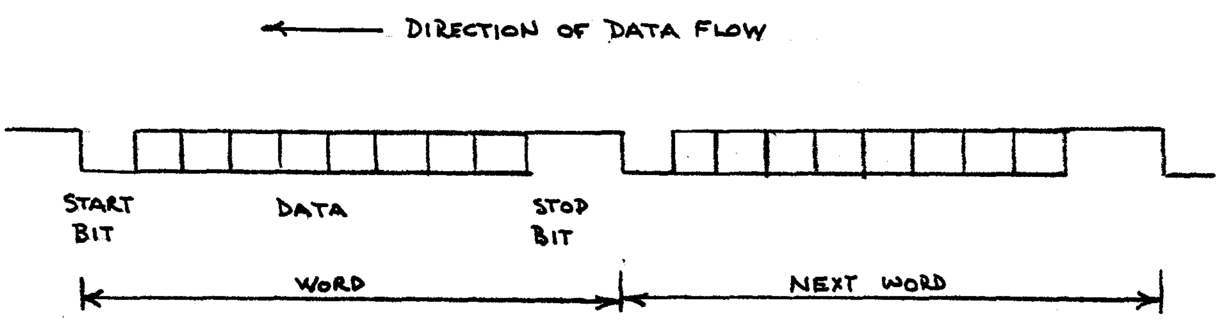


FIG 20 BIT FRAMING

Figure 20 shows how the data is loaded onto the cassette tape. As long as the micro-computer is not sending any data the USART will continuously output "l's" onto the tape. As soon as the USART gets a word to record, a start bit is sent, a single "0", this is followed by the word (which can be 4 to 8 bits long depending on how the USART was initialized), finally a stop bit is sent (this can be one, one and one half, or two "l's"). During the stop bit the USART flags the micro-computer so that the next word can be sent.

In the mode chosen for this project the receiver clock runs at 64 times the expected bit (or baud) rate. Since the CYBER 170 terminal operates at 300 baud and it was convenient to use the same clock for the terminal USART and the tape deck USART, the clock was set to 19,200 Hz.

Before the system can be set to playback data the tape must be rewound such that there is a string of "l's" at least eleven bits long before the data starts, this will ensure that the first "0" found is a start bit. As soon as the input line goes LO the USART counts 32 clock cycles and checks the input line again. If the input line has gone HI in the meantime the USART assumes it detected a noise signal and goes on searching for the start bit. If the input line is still LO a valid start bit has been found! 64 more clock cycles are counted off and the input line is tested. Here the USART will find the center of the first bit in the serial sequence, this bit is clocked serially into a register. 64 more cycles are counted and the second bit is clocked into the register. This process continues until the whole word is clocked in, 8 bits in this case. Finally 64 more clock cycles are counted and the USART checks for a "1", the stop bit. If there is a "1" the micro-computer has 2½ bit times to fetch the data before the USART finds the next start bit and loads the next word in sequence. If the USART finds a "0" instead of a stop bit something is wrong and an error flag is set. As can be seen as long as the clock rate is reasonably close to 64 times the baud rate (within about 6%) the USART will accurately receive the data from the tape and relay it to the micro-computer.

The micro-computer communicates with the USART in two ways. It has the one register which the USART shifts the data through and is addressed like any RAM location. As noted in Appendix 1 (Memory Map) the cassette deck USART occupies memory locations 3800 to 3FFF, any one of these addresses may be used to access the USART shift register, however, it is always refered to using location 3800, all the rest are redundancies. Before the USART can do anything it must receive a Mode Instruction and a Command Instruction, both of these are sent to the USART via an OUT PORT instruction from the micro-computer, namely port 38, see Appendix 2 (Port Map). In order that the USART knows whether it is receiving a mode or command instruction the USART is first reset then the mode instruction is sent, followed by the command instruction. Any subsequent instructions are assumed to be command unless the command instruction specifies the next instruction is a mode instruction or the USART is reset again.

Mode Instruction Format: D7 D6 D5 D4 D3 D2 D1 D0

D1 D0	Baud rate factor	00	synchronous mode
		01	asynchronous 1 times clock
		10	" 16 " "
		11	" 64 " "

D3 D2	Character Length	00	5 bits	
		01	6 bits	
		10	7 bits	
		11	8 bits	
D5 D4	Parity Control	X0	no parity	
		01	odd parity	
		11	even parity	
D7 D6	Framing Control	00	not valid	only valid for asynchonous operation
		01	1 stop bit	
		10	1½ stop bit	
		11	2 stop bits	

As already discussed the USART must operate in asynchronous mode with a clock rate 64 times the baud rate, there are 8 bits per word, no parity bit is to be sent or sought, and two "1's" are to be used as the stop bit. From the above format the mode instruction will be 11001111 binary or CF hex. The following short micro-program can be used to set the mode after the USART has been reset (it can be reset by hitting the "reset" key on the micro-computer).

```
3E CF          MVI A, $CF          ;load reg A with mode
D3 38          OUT PORT 38         ;mode to USART
```

Once the mode instruction has been sent the USART requires the command instruction, in the following format:

Command Instruction Format: D7 D6 D5 D4 D3 D2 D1 D0

D0	Transmit enable	0	disable
		1	enable
D1	Data terminal ready	0	ready
		1	not ready
D2	Receive enable	0	disable
		1	enable
D3	Break character	0	normal operation
		1	transmit "0's"
D4	Reset error flags	0	normal operation
		1	reset error flags to "0"

D5	Request to send	0	send to terminal
		1	do not send to terminal
D6	Internal reset	0	normal operation
		1	next instruction is "mode"

D7 Hunt mode (not used in asynchronous operation)

The command instructions are sent in an identical way as the mode instructions.

All that must be done to begin transmitting the data to be recorded on tape after the mode has been set is to send the command word, 01 hex to port 38. A short delay is necessary here to ensure the tape starts with a string of "1's", after which a data word can be sent to location 3800 to be loaded on tape. In order that the micro-computer knows when the USART is finished sending the word and is ready for the next one, it must check the USART's status register. This is done by reading the port associated with the USART, in this case, port 38.

USART Status Register Format: D7 D6 D5 D4 D3 D2 D1 D0

D0	Transmitter status	0	not ready
		1	ready
D1	Receiver status	0	not ready
		1	ready
D2	Transmitter register	0	not empty
		1	empty
D3	Parity error (not used)		
D4	Over run error (not used)		
D5	Framing error (not used)		
D6	Synch detect (synchronous operation only, not used)		
D7	Data set status	0	data not ready
		1	data ready

Once the micro-computer has sent a word to the USART it will keep reading in port 38 until it detects that the "transmitter empty" flag has been sent, telling the micro-computer to supply the next word to be recorded.

In order that the data on the tape can be identified and found amoung all the other data on the tape, each set of data must have an identifiable start sequence, a unique label, and an end sequence so the micro-computer knows when it has received all of the data.

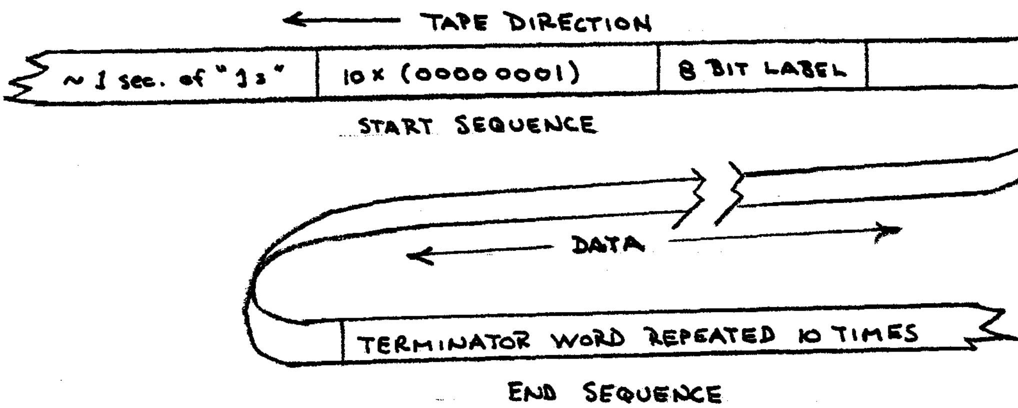


FIG. 21 CASSETTE TAPE FORMAT

There are two kinds of terminator words: As previously noted the data RAM can only hold 1000 words (for 200 cuts) so the data for a mask must be appropriately divided up into 200 cut chunks. Each of these 200 cut chunks are terminated with an ascii "E" (45 hex) except for the very last chunk which is terminated with ascii "T" (54 hex). If the micro-computer does not specify a terminator, 00 hex will automatically be used once 1000 words have been sent.

4.3 Data Recording Programs

This section discusses the micro-computer subroutines required for recording data onto the cassette tape. The data to be recorded must be first loaded into RAM and the first RAM location must be specified in register pair H-L. The subroutine "Record" assumes that the mode instruction has already been set. Register pair D-E must be loaded with the location of the terminator character or the last available RAM location.

Subroutine: "RECORD"

```
484C  3E 03     MVI A,$03        ;
  4E  CD F5 48  CALL "OUT"       ;this outputs 03 on port
                                  31 which set PHI-DECK to play
  51  3E 01     MVI A, $01       ;
  53  D3 38     OUT PORT 38      ;USART to "transmit"
  55  CD DA 49  CALL "WAIT"      ;record 300 "l's"
  58  CD 7F 49  CALL " CAS TX"   ;send 00 hex to tape
  5B  0E 0A     MVI C, $0A       ;
  5D  3E 01     MVI A, $01       ;
  5F  CD 7F 49  CALL "CAS TX"    ;send 01 hex to tape
  62  0D        DCR C            ;
  63  C2 5D 48  JNZ 485D         ;record 10 times 01 hex
  66  3A FF 30  LDA 30FF         ;get label stored at 30FF
  69  CD 7F 49  CALL "CAS TX"    ;store label on tape
  6C  7E        MOV A,M          ;get data from RAM
  6D  CD 7F 49  CALL "CAS TX"    ;record data
  70  23        INX H-L          ;point to next data loc.
  71  CD E9 48  CALL " DCMPR"    ;check for terminator
  74  C2 6C 48  JNZ 486C         ;continue recording
  77  0E 0A     MVI C, $0A       ;
  79  7E        MOV A,M          ;get terminator
  7A  CD 7F 49  CALL "CAS TX"    ;
  7D  0D        DCR C            ;
  7E  C2 79 48  JNZ 4879         ;record terminator 10 times
  81  3E 02     MVI A, $02       ;
  83  CD F5 48  CALL "OUT"       ;stop tape deck
  86  C9        RET              ;
```

The subroutine "Out" controls the operation of the PHI-DECK via port A on one of the 8155 I/O chips (port 31), see figure 18, page 35.

```
00 hex sets the tape deck to fast forward
01                           fast rewind
02                           stop
03                           play
```

One of the above codes must be loaded into register A before the following subroutine is run:

Subroutine: OUT

```
48F5  D3 31      OUT PORT 31      ;send start of tape deck
                                   control pulse
  F7  D5         PUSH D-E         ;
  F8  11 00 40   LXI D-E $4000    ;set for 25 msec delay
  FB  CD F1 05   CALL "DELAY"     ;call monitor program
  FE  D1         POP D-E          ;
  FF  3D         DCR A            ;reg. A = FF
4900  D3 31      OUT PORT 31      ;end control pulse
  02  C9         RET              ;cassette deck in re-
                                   quested mode
```

The subroutine "Wait" is simply a one second delay, used in "Record" to run off a stretch of tape before commencing to record data. The USART will automatically record about 300 "1's" during this period.

Subroutine: WAIT

```
49DA  D5         PUSH D-E         ;
  DB  11 FF FF   LXI D-E $FFFF    ;
  DE  CD F1 05   CALL "DELAY"     ;
  E1  D1         POP D-E          ;
  E2  C9         RET              ;delay complete
```

"CAS TX" sends the word in the accumulator to the USART to be recorded onto the cassette tape.

Subroutine: CAS TX

```
497F  32 00 38   STA 3800         ;reg. A to USART
  82  DB 38      IN PORT 38       ;check status register
  84  E6 04      ANI $04          ;mask all but transmitter
                                   empty flag
  86  00         NOP              ;
  87  CA 82 49   JZ 4982          ;wait until transmitter
                                   is empty
  8A  C9         RET              ;word recorded
```

"DCMPR" compares the register pairs D-E and H-L. If they are equal the zero flag is set, otherwize it is reset. This subroutine is used in "Record" to detect the end of the data (end of buffer).

Subroutine: DCMPR

```
48E9  C5          PUSH B-C          ;
  EA  43          MOV B,E           ;
  EB  7D          MOV A,L           ;
  EC  B8          CMP B             ;CY = 1 if L< E
  ED  C1          POP B-C           ;
  EE  D8          RC                ;return if E not = L
  EF  C5          PUSH B-C          ;
  F0  42          MOV B,D           ;
  F1  7C          MOV A,H           ;
  F2  B8          CMP B             ;CY = 1 if H< D
  F3  C1          POP B-C           ;
  F4  C9          RET               ;
```

4.4 Playing Back the Recorded Data

Before running the "Playback" subroutine the operator must ensure the record/playback switch is in the "playback" position or the data on the tape will be replaced by garbage. The subroutine begins by searching for a string of ten 01 hex, when it finds such a string the start of the data has been found. The next word read from the tape is the label, this label is displayed in the data field display of the micro-computer. "Playback" compares this label to the label the user has requested and stored in RAM location 30FF. If the labels match, the subroutine commences to load the data from the tape into data RAM, starting at the location specified in the register pair H-L. "Playback" is complete when the terminator string is found and the tape deck is stopped.

Subroutine: PLAYBACK

```
4887  3E 03       MVI A, $03        ;
  89  CD F5 48    CALL "OUT"        ;start the tape deck
```

```
488C  3E 04      MVI A, $04        ;
  8E  D3 38      OUT PORT 38       ;set USART to receive
  90  0E 0A      MVI C, $0A        ;
  92  CD 8B 49   CALL "CAS RX"     ;input word from tape   -
  95  3D         DCR A             ;
  96  C2 90 48   JNZ 4890          ;jump if inputed word not 01
  99  0D         DCR C             ;
  9A  C2 92 48   JNZ 4892          ;find string of ten 01 s
  9D  CD 9B 49   CALL "CAS RX"     ;get label
  A0  4F         MOV C,A           ;
  A1  CD 1F 49   CALL "DISPLAY"    ;display the label
  A4  3A FF 30   LDA 30FF          ;get desired label
  A7  B9         CMP C             ;compare desired label with
                                    label from tape
  A8  C2 90 48   JNZ 4890          ;try again if wrong label
  AB  2B         DCX H-L           ;start location of RAM
  AC  06 FF      MVI B, $FF        ;
  AE  0E 00      MVI C, $00        ;
  B0  CD E9 48   CALL "DCMPR"      ;look for end of data
  B3  23         INX H-L           ;increment pointer
  B4  CA E3 48   JZ 48E3           ;end found terminate data
                                    transfer
  B7  CD 8B 49   CALL "CAS RX"     ;get data from tape
  BA  77         MOV M,A           ;store data in data RAM
  BB  CA CC 48   JZ 48CC           ;jump if terminator 00 found
  BE  FE 54      CPI $54           ;
  C0  CA CC 48   JZ 48CC           ;jump if terminator 54 found
  C3  7E         MOV A,M           ;
  C4  FE 45      CPI $45           ;
  C6  CA CC 48   JZ 48CC           ;jump if terminator 45 found
  C9  C3 AC 48   JMP 48AC          ;no terminator get next data
  CC  7E         MOV A,M           ;get terminator
  CD  B8         CMP B             ;
  CE  47         MOV B,A           ;load terminator into B
  CF  CA D4 48   JZ 48D4           ;jump if terminator same
                                    as that in reg. B
  D2  0E 00      MVI C, $00        ;the terminator found was
                                    not part of terminator se-
                                    quence, must be data, re-
                                    set counter.
  D4  0C         INC C             ;count number of termination
                                    characters found
  D5  79         MOV A,C           ;
  D6  FE 0A      CPI $0A           ;
  D8  C2 B0 48   JNZ 48B0          ;continue until 10 terminators
```

```
48DB   7D          MOV A,L         ;
  DC   D6 09       SUI $09         ;
  DE   6F          MOV L,A         ;
  DF   7C          MOV A,H         ;
  E0   DE 00       SBI $00         ;
  E2   67          MOV H,A         ;once terminator sequence
                                    is found the last 9 RAM
                                    locations only repeat the
                                    termination character, so
                                    subtract them from pointer
  E3   3E 02       MVI A, $02      ;
  E5   CD F5 48    CALL "OUT"      ;stop the tape recorder
  E8   C9          RET             ;
```

Note: "Playback" sets the register pair D-E to the last available RAM location, so that if a terminator is not found only as much data is read in as there is RAM to load it into.

"CAS RX" accepts data that has been received by the USART into register A of the micro-computer.

Subroutine: CAS RX

```
498B   DB 38       IN PORT 38      ;check USART status reg.
  8D   E6 02       ANI $02         ;mask all but receiver flag
  8F   00          NOP             ;
  90   CA 8B 49    JZ 498B         ;wait until receiver is full
  93   3A 00 38    LDA 3800        ;load word into reg. A
  96   C9          RET             ;
```

4.4.1 The Micro-computer Display Field

The micro-computer display has six digits, four for the address and two for data. Each digit is a seven segment LED display with a decimal point.

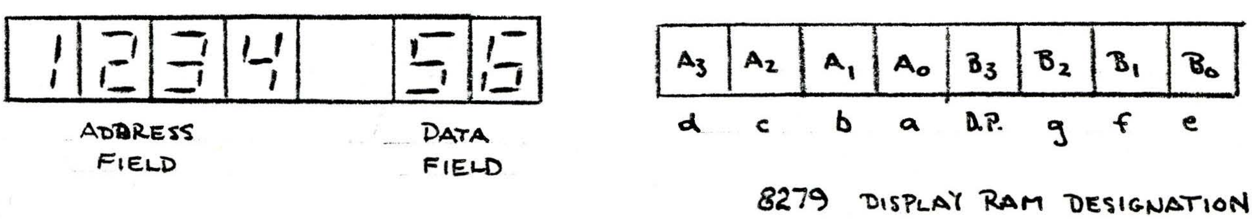


FIG 22 MICRO-COMPUTER DISPLAY FORMAT

The display fields are controlled by a Programmable Keyboard/Display Interface I.C. (an Intel 8279 was used). Although the chip occupies memory locations 1800 to 1FFF (see the Memory Map, Appendix A) only two of them are useful. The micro-computer can read the keyboard or write to the display via location 1800, it can also read the status word or write to the command register via location 1900. Figure 22 shows how digits can be displayed, for example; to display a "7", segments a, b, and c must be activated. To do this the micro-computer outputs 10001111 binary or 8F hex to location 1800. The simplest way of obtaining the display code is by using a look-up table. Such a look-up table is already supplied in the monitor programs supplied with the micro-computer. Locations 0384 through 0393 contain the codes for the characters: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, and F respectively.

The 8279 can store these codes in one of 16 RAM locations each one controlling a single digit. The micro-computer used has only 6 display digits, see figure 22, so only locations 0 to 5 inclusive are used. A command word must be sent via location 1900 to instruct the 8279 as to which RAM location the micro-computer wishes to write to or read from, The MCS-85 User's Manual pages 5-43 to 5-53 details all of the possible commands. The 8279 has the additional feature of auto-incrementing, meaning that successive write (or read) instructions will load the codes in successive RAM locations without the need of repeatedly sending new commands. The command 1001AAAA binary will send the first code to location AAAA binary, the next code to AAAA+1, etc.

The following subroutine will display a word in register C on the data field display in hexidecimal.

Subroutine: DISPLAY

```
491F  3E 94       MVI A, $94          ;
  21  32 00 19    STA 1900            ;set control register to
                                       write to the data field
  24  79          MOV A,C             ;get word to be displayed
  25  0F          RRC                 ;
  26  0F          RRC                 ;
  27  0F          RRC                 ;
  28  0F          RRC                 ;exchange nibbles
  29  CD 2D 49    CALL "LOC 492D"     ;display left nibble
  2C  79          MOV A,C             ;
492D  E6 0F       ANI $0F             ;mask left nibble
  2F  C6 84       ADI $84             ;add 84 hex to right nib-
                                       ble to get lower byte of
                                       look-up table location
  31  E5          PUSD H-L            ;
  32  26 03       MVI H, $03          ;upper byte of look-up
                                       table location
  34  6F          MOV L,A             ;lower byte in reg. L
  35  7E          MOV A,M             ;get digit code into reg. A
  36  2F          CMA                 ;
  37  32 00 18    STA 1800            ;display digit
  3A  E1          POP H-L             ;
  3B  C9          RET                 ;
```

CHAPTER 5

As can be imagined computing the run lengths, the slopes, and the control digits and loading them all onto tape manually would be a very time consuming and tedious job, especially when masks could literally consist of thousands of individual cuts. This job can be done much more efficiently using a computer program on McMaster's CYBER 170 system. This, of course, necessitates an interface between the CYBER CRT terminal and the micro-computer. Again Mr. John Metselaar's PRO-3 circuit was used but new software was written to be compatible with the rest of the systems, and is the primary topic of this chapter.

5.1 The Micro-computer/CRT Terminal Interface

There are only three connections between the CRT terminal and the PRO-3 circuit. Two of them are shown back on figure 18, one wire is used to transmit serial data from the terminal to the micro-computer, in the other the data flows in the reverse direction. There is also a signal ground which is not shown. One unfortunate aspect of this communications link is that the serial data ports on the terminal are not TTL compatible. This problem is overcome by inserting an MC 1488 Quad Line Driver, and an MC 1489 Quad Line Receiver as shown in figure 18. Specification sheets for these I.C.s can be found in John Metselaar's thesis. Because the 1488 Line Driver must produce signals at a different voltage than the regular 0V and 5V TTL levels it must have a +/- 8 volt supply.

The micro-computer can now communicate with the terminal in exactly the same way it communicates with the PHI-DECK, through the USART assigned to the terminal. Mode and Command instructions are passed through port 40, and data is received or transmitted via memory location 4000. Chapter 4 has already dealt with the 8251 USART.

The CYBER 170 system transmits and receives serial data using Ascii code. The signal received from the terminal also include an odd parity bit preceeding the 7 bit Ascii code. The start bit is the same as for the tape deck, a single "0", but the stop bit, a "1", is only one bit wide instead of two. Finally the micro-computer must conform to the 300 baud rate that the terminal operates in. The CRT USART will operate in the asynchronous mode with the clock rate set to 64 times the baud rate, thus set at 19,200 Hz. To comply with these specifications the mode instruction must be set to 01011011 binary, 5B hex (asynchronous with a 64 times clock rate, 7 bit characters, odd parity, and one stop bit), see section on USART control in chapter 4.

5.2 Loading Data from the CRT onto the Data RAM

The micro-computer operates using 8 bit words consisting of two packed hexidecimal digits while the CRT sends data in Ascii code. Ascii 0 to 9 is 30 to 39 hex so the required digits can be obtained by simply masking the first three bits of the Ascii code. Ascii A to F however is 41 to 46 hex so the conversion isn't quite so simple. To get around this problem Ascii characters J to O (4A to 4F hex) were used to represent the hexidecimal digits A to F. The data is then stored in RAM by accepting the characters two at a time, loading the first character received in the most significant nibble of the byte and the second one in the least significant nibble.

The subroutine "Input" loads data from the CRT into the micro-computer's data RAM starting at the location specified in register pair H-L. It assumes that the mode instruction has already been sent to the USART.

Subroutine: INPUT

```
4997  3E 04      MVI A, $04       ;
  99  D3 40      OUT PORT 40      ;set USART to receive
  9B  3A 00 40   LDA 4000         ;clear the USART
  9E  E5         PUSH H-L         ;
  9F  0E FF      MVI C, $FF       ;use reg. C as flag:
                                   00 is 1st ascii character
                                   01 is 2nd ascii character
  A1  CD 73 49   CALL "CRT RX"    ;Ascii character to reg. A
  A4  06 30      MVI B, $30       ;
  A6  B8         CMP B            ;
  A7  DA A1 49   JC 49A1          ;ignore Ascii less than 30
  AA  06 58      MVI B, $58       ;
  AC  B8         CMP B            ;
  AD  D2 A1 49   JC 49A1          ;ignore Ascii greater than 58
  B0  06 45      MVI B, $45       ;
  B2  B8         CMP B            ;
  B3  CA D5 49   JZ 49D5          ;jump if terminator "E"
                                   (Ascii 45) is found
  B6  06 54      MVI B, $54       ;
  B8  B8         CMP B            ;
  B9  CA D5 49   JZ 49D5          ;jump if terminator "T"
                                   (Ascii 54) is found
  BC  E6 0F      ANI $0F          ;mask left nibble
  BE  0C         INR C            ;increment flag
  BF  C2 CA 49   JNZ 49CA         ;jump if 2nd character
  C2  0F         RRC              ;
  C3  0F         RRC              ;
  C4  0F         RRC              ;
  C5  0F         RRC              ;1st character, swap nibbles
  C6  77         MOV M,A          ;store left nibble
  C7  C3 A1 49   JMP 49A1         ;jump to get 2nd character
```

```
49CA  B6          ORA,M          ;combine nibbles from 1st
                                  and 2nd characters
  CB  77          MOV M,A        ;store data word
  CC  CD E9 48    CALL "DCMPR"   ;check for end of memory
  CF  23          INX H-L        ;increment RAM pointer
  D0  C2 9F 49    JNZ 499F       ;continue accepting data
  D3  97          SUB A,A        ;clear reg. A
  D4  2B          DCX H-L        ;point to last RAM location
  D5  77          MOV M,A        ;store terminator
  D6  EB          XCHG           ;store last memory location
                                  in D-E, for use in RECORD
  D7  E1          POP H-L        ;recover start location
  D8  0C          INR C          ;
  D9  C9          RET            ;
```

The subroutine "Input" uses a short subroutine to receive Ascii encoded data from the CRT terminal. The character is loaded into register A.

Subroutine; CRT RX

```
4973  DB 40       IN PORT 40     ;get status word
  75  E6 02       ANI $02        ;mask all but "receiver ready"
  77  00          NOP            ;
  78  CA 73 49    JZ 4973        ;loop until receiver ready
  7B  3A 00 40    LDA 4000       ;load character into reg. A
  7E  C9          RET            ;
```

5.3 Sending Characters to the CRT Terminal

To send characters to the CRT all that needs to be done is to set the USART to transmit and output the appropriate Ascii code via location 4000. This subroutine outputs "GO"; characters "G" (Ascii 47 hex), "O" (4F hex), carriage return (0D hex), and line feed (0A hex). This "GO" is used, as will be seen in the next chapter, to indicate to the CYBER 170 that the micro-computer is ready to accept data from the terminal.

Subroutine: GO

```
4906  3E 01       MVI A, $01        ;
  08  D3 40       OUT PORT 40       ;set USART to transmit
  0A  3E 47       MVI A, $47        ;Ascii for letter "G"
  0C  CD E3 49    CALL"CRT TX"      ;send character
  0F  3E 4F       MVI A, $4F        ;letter "O"
  11  CD E3 49    CALL "CRT TX"     ;
  14  3E 0D       MVI A, $0D        ;carriage return
  16  CD E3 49    CALL "CRT TX"     ;
  19  3E 0A       MVI A, $0A        ;line feed
  1B  CD E3 49    CALL "CRT TX"     ;
  1E  C9          RET               ;"GO" sent
```

Subroutine: CRT TX

```
49E3  32 00 40    STA 4000          ;reg. A to USART
  E6  DB 40       IN PORT 40        ;get status word
  E8  E6 04       ANI $04           ;mask all but "trans-
                                     mitter empty" flag
  EA  00          NOP               ;
  EB  CA E6 49    JZ 49E6           ;wait until USART is
                                     finished sending
  EE  C9          RET               ;
```

CHAPTER 6

A Fortran program was developed on the McMaster CYBER 170 system to enable the user to interact with the cutting table's micro-computer via a CRT terminal. This greatly simplifies the operating proceedure and provides a method for error detection so masks that are larger than the 44 inch square cutting table are not attempted.

6.1 The Conversion Program, "Xmit"

This Fortran program allows the user to enter data, either manually via the terminal or from a previously stored data file, in point to point form. As already mentioned the control system can really only cut straight lines, curves are realized by a succession of short line segments. The user must simply provide a series of x-y co-ordinate points which the cutting table will then cut in "connect-the-dots" fashion. In addition to the two cartesian points a third parameter must be added to control the knife, this parameter, called "P", is 0.0 if the knife is to be up, or 1.0 if the knife is to be down (actually cutting). The very last data point entered must have "P" equal to -1.0, this directs that the knife be up and also indicates the end of the data. The data must be in the form (x,y,P) and x and y must be entered in inches ranging from 0.0 to 42.0 (the maximum size of the table). The run length registers hold only

16 binary bits, 65,535 steps, which works out to about 32 inches, so the program automatically divides cuts longer than 32 inches into two pieces.

Data Format:

x	y	P
XX.XXXX	XX.XXXX	0.0 or 1.0
XX.XXXX	XX.XXXX	0.0 or 1.0
XX.XXXX	XX.XXXX	0.0 or 1.0
.	.	.
.	.	.
.	.	.
XX.XXXX	XX.XXXX	0.0 or 1.0
XX.XXXX	XX.XXXX	-1.0

XX.XXXX is a decimal number in inches ranging from 0.0 to 42.0, four significant decimal places.

Fortran Program: XMIT

```
      PROGRAM XMIT(INPUT,OUTPUT,DATAF,TAPE1=DATAF,TAPE5=
      INPUT,TAPE6=OUTPUT)
      REAL MOVE(4000,3)
      INTEGER BUFSIZE, ITS(9),HEX(16),ECL,ECB,DATA HEX/
      1H0,1H1,1H2,1H3,1H4,1H5,1H6,1H7,1H8,1H9,1HJ,1HK,1HL,
      1HM,1HN,1HO/DATA EOL,EOB/1HP,1HX/
      REWIND 1

      ;The above part of the program simply initializes
       the tapes and dimensions the variables.

    5 WRITE(6,10)
   10 FORMAT(10X,"ENTER POINTS FROM DATA FILE(1) OR
         TERMINAL(5)?")
         READ(5,*)INPDEV
         IF(INPDEV.NE.1.AND.INPDEV.NE.5) GO TO 5

         ;the user is asked whether he/she wishes to enter
          the data points manually from the terminal (5) or
          from a data file (1).  If anything other than 1 or
          5 is entered the question will be repeated.

         XOFSET=0.0
         YOFSET=0.0
         XSCALE=2000.0
         YSCALE=2000.0
```

;Initialize the parameters to be used to convert the user's scale to the cutting table scale. XOFSET and YOFSET indicate the difference between the user's origin and the origin used by the cutting table. Both are set to zero. XSCALE and YSCALE indicate the ratios of scales between the user and the stepping motors, the user works in inches while the motors operate in 1/2000 th inch steps. If the user wishes to work in metric, say in centimeters, the scale factors would be changed to 787.4.

```
00220       WRITE(6,20)
00230    20 FORMAT(10X,"ENTER DATA IN FORM (X,Y,P), WITH X, Y
            IN INCHES",/,10X,"P=0 (MOVE), P=1 (CUT), P=-1 (END
            OF DATA)",/,10X,"X,Y DATA MUST BE IN RANGE OF 0-42
            INCHES")
```

;The user is informed how to enter the data and the limitations of the cutting table.

```
00240       N=0
00250    30 N=N+1
00260       READ(INPDEV,*)X,Y,P
00270       IF(X.LT.0.0.OR.X.GT.42.0.OR.Y.LT.O.O.OR.Y.GT.42.0)
            GO TO 40
00280       IF(P.NE.-1.0.AND.P.NE.1.0.AND.P.NE.0.0) GO TO 40
00290       MOVE(N,1) = X
00300       MOVE(N,2) = Y
00310       MOVE(N,3) = P
00320       IF(P.EQ.-1.0) GO TO 60
00330       GO TO 30
```

;The data is read into the dimensioned matrix MOVE. Data can be read from a data file from TAPE 1 or from the terminal, TAPE 5. MOVE has been dimensioned to accept a maximum of 4000 cuts, if more are required line 00110 must be altered.

```
00340    40 WRITE(6,50)N
00350    50 FORMAT(10X,"ERROR IN INPUT DATA,RE-ENTER POINT",I6)
00360       N=N-1
00370       IF(INPDEV.EQ.1)STOP
00380       GO TO 30
```

;If an error was detected in loading the data the program will print an error message and ask that the incorrect data be re-entered.

```
60 WRITE(6,70)N
70 FORMAT(10X,I6," POINTS ENTERED")

;The number of points entered is displayed

   DO 90 I=1,N
   MOVE(I,1)=(MOVE(I,1)-XOFSET)*XSCALE
   MOVE(I,2)=(MOVE(I,2)-YOFSET)*YSCALE
90 CONTINUE

;The data in the matrix MOVE is re-scaled to the
 cutting table units

   NM1=N-1
   DO 100 I=1,NM1
   J=I+1
   DELTAX=MOVE(J,1)-MOVE(I,1)
   DELTAY=MOVE(J,2)-MOVE(I,2)
   P=MOVE(J,3)
   MOVE(I,3)=0.0
   IF(ABS(DELTAX).LT.ABS(DELTAY))MOVE(I,3)=MOVE(I,3)+8.0
   IF(DELTAY.GE.0.0)MOVE(I,3)=MOVE(I,3)+4.0
   IF(DELTAX.GE.0.0)MOVE(I,3)=MOVE(I,3)+2.0
   IF(P.EQ.1.0)MOVE(I,3)=MOVE(I,3)+1.0
   MOVE(I,1)=ABS(DELTAX)
   MOVE(I,2)=ABS(DELTAY)
100 CONTINUE

   ;The data is all converted from point to point
    data to the data format that the micro-computer
    uses: namely, the control word is generated and
    the absolute data is converted to relative data

   BUFSIZE=200
   K=0

   ;Because of the limited RAM on the micro-computer
    buffersizes are limited to 200 data sets

   DO 130 I=1,NM1
   IF(K.EQ.BUFSIZE)K=0
   J=1
   SLOPE=0.0
   IF(MOVE(I,1).EQ.0.0.OR.MOVE(I,2).EQ.0.0)GO TO 13
   SLOPE=MOVE(I,1)/MOVE(I,2)
   IF(SLOPE.EQ.1.0)SLOPE=.9999999999
   IF(SLOPE.GT.1.0)SLOPE=1.0/SLOPE
```

```
00690  13 IF(MOVE(I,1).LT.MOVE(I,2))J=2
00700     ZSCALE = XSCALE
00710     IF(J.EQ.2.0)ZSCALE=YSCALE
00720     ZOFSET=XOFSET
00730     IF(J.EQ.2.0)ZOFSET=YOFSET
00740     LCHK1=1
00750     IF(MOVE(I,J).GT.(32.0-ZOFSET)*ZSCALE)LCHK1=2
00760     IF(MOVE(I,J).GT.(32.0-ZOFSET)*ZSCALE)MOVE(I,J)=
          MOVE(I,J)/2.0
```

;The conversion to the micro-computer data format is completed. The relative point to point moves are converted to lengths and slopes. Also cuts that are longer than 32 inches but shorter than 44 inches are divided in half.

```
00770     DIGITS(5)=INT(MOVE(I,J)/4096.0)
00780     DIGITS(6)=INT(MOVE(I,J)/256.0)-DIGITS(5)*16
00790     DIGITS(7)=INT(MOVE(I,J)/16.0)-DIGITS(6)*16-DIGITS(5)*256
00800     DIGITS(8)=INT(MOVE(I,J))-DIGITS(7)*16-DIGITS(6)*256
          -DIGITS(5)*4096
00810     DIGITS(1)=INT(SLOPE*16.0)
00820     DIGITS(2)=INT(SLOPE*256.0)-DIGITS(1)*16
00830     DIGITS(3)=INT(SLOPE*4096.0)-DIGITS(2)*16-DIGITS(1)*256
00840     DIGITS(4)=INT(SLOPE*65536.0)-DIGITS(3)*16-DIGITS(2)
          *256-DIGITS(1)*4096
00850     DIGITS(9)=MOVE(I,3)
```

;Here the conversion from decimal to hexidecimal is made. Now all the "DIGITS" contain values between 0 and 15.

```
00860     DO 110 L=1,9
00870     DIGITS(L)=HEX(DIGITS(L)+1)
00880 110 CONTINUE
```

;The values in DIGITS are converted to the characters used by the micro-computer, namely: 0,1,2,3,4,5,6,7,8 9,J,K,L,M,N, and O. The look-up table in line 00120 is used.

```
00890     IF(K.NE.0.0) GO TO 115
00900 125 WRITE(6,126)
00910     READ(5,126)SIGNL
00920     IF(SIGNL.NE."GO") GO TO 125
```

;Wait until the micro-computer is ready for data, ie. it sends a "GO", before continuing to output data.

```
127 FORMAT(A2)
126 FORMAT("WAITING FOR A "GO" SIGNAL FROM MICRO-COM-
    PUTER")
115 DO 116 LCHK2=1,LCHK1
    K=K+1
    IF((I.NE.NM1.AND.K.LT.BUFSIZE)WRITE(6,120)DIGITS,EOL
116 IF(I.EQ.NM1.OR.K.GE.BUFSIZE)WRITE(6,120)DIGITS,EOB
120 FORMAT(1X,10A1)

;Once a go is received from the micro-computer the terminal
 commences to send the data to the data RAM

    IF(K.GT.BUFSIZE.AND.I.NE.NM1)WRITE(6,122)
    IF(I.EQ.NM1)WRITE(6,121)
121 FORMAT(" T")
122 FORMAT(" E")

;At the end of each block of data, consisting of 200
 "cuts" each a termination character is sent.  "E" if
 there is still more data to be transfered, or "T" if
 all the data is transfered.

130 CONTINUE
    END
```

CHAPTER 7

All the individual components of the micro-computer controlled cutting table have now been discussed, all that remains is to show how all these components interact and how the whole system works. There are three basic steps to cutting a mask, first a data file of the data points which are to be connected in "connect-the-dots" fashion must be made up. The simplest way to do this is by a user Fortran program. There are no particular rules on how to create such a program as long as a data file of the correct format results. An example of such a program is given in chapter 8. If the user wishes to enter the points manually step one may be omitted. Secondly the data is converted and loaded onto magnetic tape, finally the data is recovered from the tape and the mask is cut.

7.1 Loading the Data onto Magnetic Tape

The following mainline program "CRT to Tape" receives the data from the terminal and loads it onto the cassette tape.

Mainline: CRT TO TAPE

```
4800  31 C2 20    LXI SP 20C2        ;initialize stack pointer
  03  CD 3C 49    CALL INITIALIZE    ;initialize the ports and
                                      set the modes of the USARTs
```

```
4806  CD 06 49    CALL "GO"          ;inform the CRT that the
                                      micro-computer is ready
  09  21 00 50    LXI H-L $5000      ;set starting location of
                                      data RAM to 5000
  0C  11 FF 53    LXI D-E $53FF      ;set end location of data RAM
  0F  CD 97 49    CALL "INPUT"       ;load data from CRT into data
                                      RAM, the CRT will send no
                                      more than 200 cuts (1000
                                      words) before sending a
                                      termination character.
  12  C2 25 48    JNZ 4825           ;data has been sent, jump
                                      to error display if an
                                      odd number of characters
                                      were sent.
  15  CD 4C 48    CALL "RECORD"      ;load data onto tape
  18  7E          MOV A,M            ;get termination character
  19  FE 54       CPI $54            ;compare to Ascii "T"
  1B  C2 06 48    JNZ 4806           ;there is still more data
                                      jump back to get next block
  1E  3E 40       MVI A, $40         ;
  20  D3 38       OUT PORT 38        ;recording complete, reset
                                      PHI-DECK USART
  22  D3 40       OUT PORT 40        ;reset CRT USART
  24  CF          RST 1              ;return to monitor
```

Subroutine: ERROR

```
4825  0E E0       MVI C, $E0         ;set error code
  27  CD 1F 49    CALL "DISPLAY"     ;display error code
  2A  76          HLT                ;wait for user to do some-
                                      thing about the error.
                                      This error indicates that
                                      a character was missed or
                                      an inadmissible one was
                                      sent.  Normally the problem
                                      is noise on the data line
                                      caused by strong electrical
                                      interference, ie. electric
                                      machinery nearby.  Remove
                                      the source of the noise and
                                      start over.
```

Subroutine: INITIALIZE

```
493C  3E FF       MVI A, $FF         ;
```

```
493E  D3 02      OUT PORT 02      ;port 00 set to output
  40  3E BC      MVI A, $BC       ;bits 0,1, and 6 of port 01
                                   set to input, rest are output
  42  D3 03      OUT PORT 03      ;
  44  3E 0F      MVI A, $0F       ;
  46  D3 20      OUT PORT 20      ;set ports 21,22 and 23
                                   to output
  48  3E 0F      MVI A, $0F       ;
  4A  D3 28      OUT PORT 28      ;set ports 29,2A, and 2B
                                   to output.
  4C  3E 0D      MVI A, $0D       ;
  4E  D3 30      OUT PORT 30      ;PHI-DECK control ports,
                                   ports 31 and 33 set to
                                   output, 32 to input
  50  97         SUB A,A          ;reg. A = 00
  51  D3 00      OUT PORT 00      ;clear port 00, this ensures
                                   the motors are stopped.
  53  D3 01      OUT PORT 01      ;clear port 01
  55  D3 33      OUT PORT 33      ;set tape direction to forward
  57  3E 02      MVI A, $02       ;
  59  CD F5 48   CALL "OUT"       ;ensure tape deck is stopped
  5C  3E 01      MVI A, $01       ;
  5E  D3 33      OUT PORT 33      ;end tape direction pulse
  60  3E CF      MVI A, $CF       ;
  62  D3 38      OUT PORT 38      ;set PHI-DECK USART:  8 bit
                                   word, asynch 64, no parity,
                                   2 stop bits.
  64  3E 5B      MVI A, $5B       ;
  66  D3 40      OUT PORT 40      ;set CRT USART:  7 bit word,
                                   asynch 64, odd parity, 1
                                   stop bit.
  68  C9         RET              ;
```

7.2 Cutting the Mask

The following mainline program, "Cut", recovers the data from the magnetic tape, one 200 cut block at a time, and runs the motors accordingly. The program will continue to run the motors without any user interaction until all the data is used up.

Mainline: CUT

```
482B  31 C2 20   LXI SP 20C2      ;initialize stack pointer
```

```
482E  CD 3C 49   CALL "INITIALIZE" ;initialize ports and USARTs
  31  21 00 50   LXI H-L $5000     ;set start of data RAM
  34  11 FF 53   LXI D-E $53FF     ;set end of data RAM
  37  CD 87 48   CALL "PLAYBACK"   ;load data from tape to RAM
  3A  E5         PUSH H-L          ;save terminator location
  3B  CD 20 4A   CALL "ADJUST"     ;adjust control digits to
                                   correct motor speeds
  3E  CD 3C 4A   CALL "MOTOR"      ;run the motors until the
                                    data in RAM runs out
  41  E1         POP H-L           ;recover terminator loc.
  42  00         NOP               ;
  43  7E         MOV A,M           ;get terminator character
  44  FE 45      CPI $45           ;compare to Ascii "E"
  46  CA 31 48   JZ 4831           ;mask not complete, continue
  49  C3 1E 48   JMP 481E          ;mask complete, jump to the
                                    reset routine, page 63
```

7.3 Operating Instructions

Step 1: If the data points are to be entered manually skip this step, otherwise create a data file of all the data points required in the mask. See section 6.1.

Step 2: Convert the data file to the format used by the micro-computer and load it onto the cassette tape:

a. Turn everything on, except the power to the translator modules. The translator mudule ports must be intialized before the modules are powered up or the motors will run randomly.

b. Connect the CRT terminal to McMaster CYBER 170 system via the acoustic coupler (modem), dial 331.

c. Log on.

d. Localize the data file (if there is one) and the program XMIT:

```
/GET,XMIT
/GET,"data file name"
```

e. Run the program XMIT using the following route:

```
/FTN,XMIT,L=0,GO,PL=9999
```

This ensures that the datafile is not terminated after only 1000 lines as would happen normally.

f. Choose between manual (if no data file has been localized) or data file entry. Once loading is complete the number of points entered will be displayed.

g. Push "reset" on the micro-computer.

h. Enter the data label in RAM location 30FF:

"SBST MEM" "3" "0" "F" "F" "NEXT" "XX" "NEXT"

XX is any two digit hexidecimal code word.

i. Ensure the PHI-DECK is rewound to a blank section of the tape, make sure there is enough tape for all of the data (about ½ minute per 200 cut block)

j. Execute the data loading program:

"EXEC" "GO" "4" "8" "0" "0" "EXEC"

200 cut blocks of data will now be read first onto RAM then from there onto the tape. The recording will continue until all of the data has been transfered to the tape. The CRT will display the data in real time while it is loaded into RAM.

k. The completion of loading is indicated by a "T" following the last data line, log off CYBER.

l. Wait for the tape deck to stop. All the data required to cut the mask is now on tape, it takes 3 to 4 minutes for each block of 200 cuts to be loaded onto tape.

Step 3: Cutting the Mask:

a. Position the knife appropriately on the Rubylith.

b. Ensure that all the gears are engaged on the moving carts.

c. Load instruction "C9" in location 5400

"SBST MEM" "5" "4" "0" "0" "NEXT" "C" "9" "NEXT"

d. Rewind the tape to the start of the data. Switch toggle to "Playback".

e. Enter the data label as in Step 2h.

f. Execute the cutting program:

"EXEC" "GO" "4" "8" "2" "B" "EXEC"

g. Power up the translator modules.

The tape deck will start searching for the data identified by the label in 30FF. As the labels are found they are displayed on the micro-computer's data field display. The micro-computer will not start reading data until the correct label is found. Once the first data block is read in, the tape deck will stop and the motors start. Once all the data is used up in the RAM the motors will stop and more data read in from the tape deck. The entire operation will continue completely automatically until the mask is completed. Large masks can take several hours to complete, depending on what speed the translator modules are set to. Care must be taken not to set them too fast, as the micro-computer will start missing steps, especially if curves are being cut. Running the motors too fast will also cause the x axis to bounce a bit when the end of a cut is reached. Running the "y" motor fast and the "x" slower does not help when the "x" motor is slave to the "y" motor and 45° angles are being cut. Both motors will run at the faster speed. Both motors should be set to the same speed.

h. When the mask is complete the micro-computer will reset itself and display "8085" in the address field.

i. Power down, ensuring that the translator modules are shut off before the micro-computer. Without anything controlling the translator modules the motors are likely to start up on their own and ruin the mask.

j. Peel the mask!

CHAPTER 8

Mr. Nick Slater used the cutting table to cut masks for SAW devices he was designing. His design is detailed in his recently published thesis: Design of Wide Band, Linear Phase Surface Acoustic Wave Filters. The fingers of his device are slightly curved in order to realize a uniform response across the pass band. Curved fingers have only been obtainable since the automation of the cutting table.

Figure 23 shows a picture of a mask that was cut on the cutting table for Nick Slater's thesis. The fingers are slightly curved outward at the bottom. The data for the mask consisted of over 1500 points generated by the program on pages 70 and 71. The mask was done by first cutting all the lower fingers from right to left then cutting the upper fingers inbetween from left to right. Upon close inspection it can be seen that some of the fingers are not perfectly centered, but the error is only about .02 to .05 inches. Considering that this error was accumulated over about 2,200 inches of cutting (4.4 million steps), without any feedback, this system works very well! More accurate masks can be obtained by alternately cutting the upper and lower fingers. This was done with some other masks and there was no detectable error, all of the fingers were perfectly spaced.

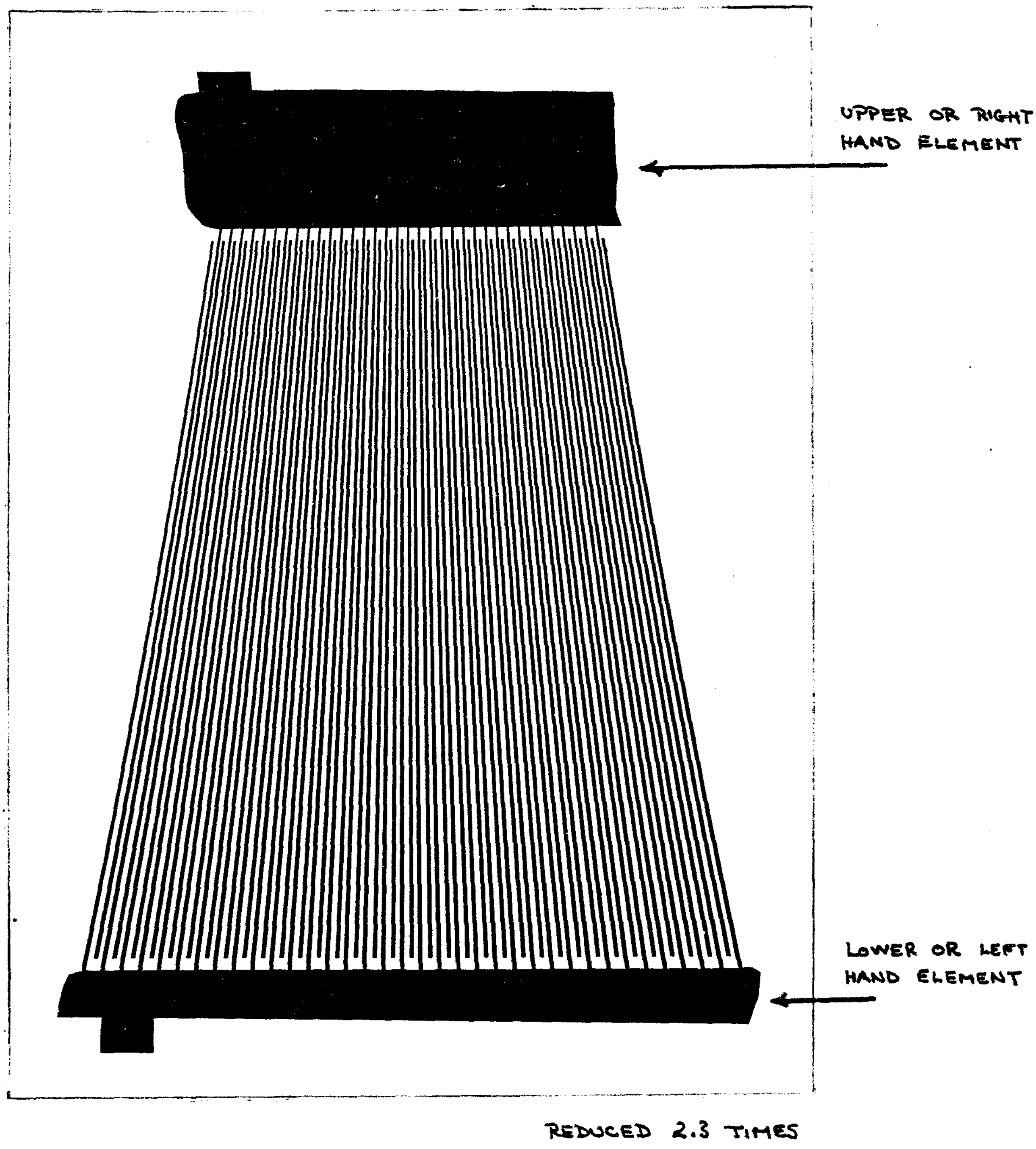


FIG. 23 NICK SLATER'S TRANSDUCER FOR HIS SAW DEVICE

```
      PROGRAM MASK3(INPUT,OUTPUT,DATA4,TAPE1=DATA4,
      TAPE5=INPUT,TAPE6=OUTPUT)
      REWIND1

      ;This program calculates the co-ordinates for a
       curved finger SAW filter mask and outputs them to
       tape1.  Fingers are symmetric about the y axis.
       Digitization follows cutting proceedure:
         1.  Starting point of transducer is output
         2.  Lower side of left hand element of finger
             pair is output, using the subroutine "Curve".
         3.  Right hand edge of finger is output.
         4.  Upper side of left hand element is output.
         5.  Starting point for next finger pair is output.
         6.  Proceedure repeats for all left hand elements.
         7.  Proceedure is executed in reverse for right
             hand elements.
         8.  Proceedure repeats for output transducer.
         9.  Original start position is output, along
             with an "End of Data" (EOD) mark.

      X=0.0
      Y=0.0
      PU=0.0
      PD=1.0
      WRITE(1,1)X,Y,PU
    1 FORMAT(3(3X,F8.5))

      ;All the controlling parameters are initialized.

      SCALE=1.0
      GAP=SCALE*0.3
      ETA=0.5
      NO=35
      W=SCALE*24.0
      BW=0.5
      FC=70.0E+06
      V=SCALE*3158.0*39.37*9.5*20.0
      NSTRP=10.0
      XOFSET=SCALE*1.0
      YOFSET=SCALE*13.0
      SEP=SCALE*6.0
      K=0

      ;The dimensions of all of the parts of the mask
       are initialized.

    5 DELTF=BW*FC
```

```
      DELTX=(W+GAP)/FLOAT(NSTRP)
      FLO=FC-DELTF/2.0
      C1=V/FLO*(0.5*FLOAT(NO)+0.125)+YOFSET
      C2=0.5*V*W/DELTF
      C3=FLO*W/DELTF-XOFSET
      X=XOFSET
      Y=YOFSET-SCALE
      WRITE(1,1)X,Y,PU

      ;Transducer digitization is set up

      N=NO
   10 X=X-DELTX

      ;Digitise left hand elements,starting at the bottom.

      SIGN=1.0
      CALL CURVE(X,Y,N,SIGN,C1,C2,C3,DELTX,NSTRP)
      X=X+DELTX

      ;Lower side of finger.

      SIGN=-1.0
      CALL CURVE(X,Y,N,SIGN,C1,C2,C3,DELTX,NSTRP)
      N=N-2
      IF(N.GE.-NO) GO TO 10
      X=X+GAP+W+GAP
      WRITE(1,1)X,Y,PU

      ;Upper side of finger.

      N=-(NO-1)
   20 X=X+DELTX

      ;Digitise right hand elements,starting at the top.

      SIGN=-1.0
      CALL CURVE(X,Y,N,SIGN,C1,C2,C3,DELTX,NSTRP)
      X=X-DELTX

      ;Upper side of finger.

      SIGN=1.0
      CALL CURVE(X,Y,N,SIGN,C1,C2,C3,DELTX,NSTRP)
      N=N+2
      IF(N.LE.(NO-1))GO TO 20
      IF(K.NE.0)GO TO 30

      ;Lower side of finger.

      X=XOFSET
      Y=YOFSET
      WRITE(1,1)X,Y,PU
```

```
      Y=YOFSET-SEP
      WRITE(1,1)X,Y,PD
      NO=11
      XOFSET=SCALE*1.0
      YOFSET=SCALE*1.0
      K=1
      GO TO 5

      ;Digitise output transducer.

   30 X=0.0
      Y=0.0
      WRITE(1,1)X,Y,EOD
      END

      SUBROUTINE CURVE(X,Y,N,SIGN,C1,C2,C3,DELTX,NSTRP)

      ;Draws a curve controlled by C1,C2, and C3 which
       reflect changes in center frequency, bandwidth,
       aperture, number of fingers, and velocity of sound.

      ;(X,Y) starting position for the finger.
      ;"SIGN" controls the direction of the cut:
        =1.0 cut from left to right
        =-1.0 cut from right to left
      ;"NSTRP" indicates the number of linear segments
        used to approximate the curve.
      ;"DELTX" gives the change in X in inches correspond-
       ing to each linear segment. ((aperature+gap)/NSTRP)

      PD=1.0
      M=NSTRP+1
      DO 15 I=1,M
      X=X+DELTX*SIGN
      Y=C1-(FLOAT(N)+SIGN/4.0)*C2/(X+C3)
      WRITE(1,1)X,Y,PD
   15 CONTINUE
      RETURN
    1 FORMAT(3(3X,F8.5))
      END
```

This program is explained in much more detail in Nick Slater's thesis. It is only included here to provide a typical example of what can be done on the automated cutting table.

CHAPTER 9

The micro-computer controlled cutting table is now fully operational and works sufficiently well to enable Dr. C.K. Campbell's lab to experiment with SAW device configurations previously unobtainable. A phenomenal time saving is also realized when cutting more conventional SAW patterns, a two week job can now be completed in a day of developing and de-bugging a data file program and another couple of hours for the automated cutting table to cut the mask.

Nonetheless, there are numerous improvements that could be made to the system:

1. Presently there is no feedback at all. The micro-computer cuts one line after another knowing only the relative data for that particular line. A feedback system would make the entire proceedure more accurate provided the feedback is very precise; about plus or minus .0005 in. Mr. Mark Usik studied the feedback problem in a fourth year thesis using synchro feedback motors, these motors proved too inaccurate to be of any use. He suggested that linear transducers, although very expensive, would probably be the best solution should accuracies greater than presently obtainable ever be required.
2. Operating the motors in the fully automatic mode, using the CYBER system to create data files, is fine for creating SAW masks. SAW devices are periodic in design and lend themselves easily to computer design. Cutting a mask for an

electronic circuit board would prove very difficult, the data file would probably have to be created manually. It would be much easier if the motors could be operated directly by the user via a keyboard or toggle switches. The operator would also need a display to tell him the position of the motors while they were running. Such a system could likely be added to the present hardware with only some software additions. The present micro-computer keyboard and display could be used.

3. The system occasionally makes a fatal error and cuts a line incorrectly. Normally this is caused by a very minor error in anyone of the thousands of data bits. Often the mask is only slightly damaged and could be saved if there was a method by which the next correct data can be found and the motors restarted. Presently there is no way of restarting the cutting programs once a mistake has been discovered. This means scraping the old mask and starting over. This sort of catastrophic failure doesn't happen too often, however, some sort of save-the-mask routine would definitely be an asset to the overall system.

4. The first time the user gets to see what his mask looks like is when it is actually completely cut on the cutting table. If some software error was made in the creation of the data file or in the loading of the magnetic tape it would only be noticed after the cutting table has spent two or three hours creating a useless mask. The addition of a plotter would enable the data to be quickly verified before the cutting table motors are set into operation. A plotter would also produce a handy hardcopy of the mask without the expense and delay of having the mask photographed.

5. A few operator oriented programs such as a cassette tape "bootstrap" would enable the user to interact more readily with the various components of the system. This would facilitate experimentation in any of the above mentioned improvements.

APPENDIX 1

An SDK-85 micro-computer was used in this project. The operational theory and all the circuit diagrams are provided in detail in Intel's MCS-85 User's Manual. All of the monitor programs provided with the micro-computer are also given in this manual. This project and John Metselaar's made numerous additions to the micro-computer, expanding its memory and I/O capability. Appendix 1 and Appendix 2 show just what memory and I/O is available on this micro-computer.

Table A1: Memory Map

Locations	Device	Comments
0000-07FF	8355 (ROM)	;Monitor programs
0800-17FF	---	;Not connected, available for the addition of more I.C.s.
1800-1FFF	8279	;No memory available, locations 1800 and 1900 address the display and keyboard, all the remainder are redundancies of these two addresses.
2000-20FF	8155 (RAM)	;Monitor programs reserve locations 20C2-20FF, the rest is free for any use.
2100-27FF		;Redundancies for locations 2000-20FF.
2800-28FF	8155 (RAM)	;Free for any use.
2900-2FFF		;Redundancies for locations 2800-28FF.
3000-30FF	8155 (RAM)	;30D0-30FF reserved for cutting table control programs, the rest is free for any use.

Locations	Device	Comments
3100-37FF		;Redundancies for locations 3000-30FF.
3800-3FFF	8251 (USART) (PHI-DECK)	;3800 is used to address the USART register, all of the rest of the locations are redundancies of 3800.
4000-47FF	8251 (USART) (CRT Terminal)	;4000 is used to address the USART register, all the rest of the locations are redundancies of 4000.
5000-53FF	8185 (RAM)	;This is the data RAM to hold mask information.
5400-57FF	8185 (RAM)	;This RAM is free for any use, often used for development of new micro software.
5800-FFFF	---	;Not connected to anything.

APPENDIX 2

Table A2: Port Map

Port	Bits	Device	Comments
00	8	8355 port A	;Input/Output port used to control the stepping motors.
01	8	8355 port B	;Same as port 00
02		8355 C/S	;Command/Status register for port 00
03		8355 C/S	;Command/Status for port 01
04			;Port 00 redundancy.
05			;Port 01 redundancy.
06			;Port 02 redundancy.
07			;Port 03 redundancy.
20		8155 C/S	;Command/Status register for ports 21, 22, and 23.
21	8	8155 port A	;Used in Mark Usik's feedback system.
22	8	8155 port B	;Mark Usik's feedback system.
23	6	8155 port C	;Mark Usik's feedback system.
28		8155 C/S	;Command/Status register for ports 29, 2A, and 2B.
29	8	8155 port A	;Mark Usik's feedback system.
2A	8	8155 port B	;Mark Usik's feedback system.
2B	6	8155 port C	;Mark Usik's feedback system.
30		8155 C/S	;Command/Status register for ports 31, 32, and 33.
31	8	8155 port A	;Used to control the functions of the cassette deck.
32	8	8155 port B	;Used to feedback the status of the cassette deck.

Port	Bits	Device	Comments
33	6	8155 port C	;Used to control the direction of the tape deck.
38		8251 (USART)	;This port gives access to the control word of the PHI-DECK USART.
40		8251 (USART)	;This port gives access to the control word of the CRT USART.
48	8	8755 port A	;Not used.
49	8	8755 port B	;Not used.
4A		8755 C/S	;Command/Status for port 48.
4B		8755 C/S	;Command/Status for port 49.
4C			;Redundancy for port 48.
4D			;Redundancy for port 49.
4E			;Redundancy for port 4A.
4F			;Redundancy for port 4B.

REFERENCES

1. Intel, COMPONENT DATA CATALOG, Literature Department Intel Corporation, Santa Clara, California, 1980.
2. Intel, MCS-85 USER'S MANUAL, Literature Department Intel Corporation, Santa Clara, California, 1977.
3. Metselaar, MICRO-COMPUTER INTERFACING, DESIGN, AND OPERATION, McMaster University Thesis, Hamilton, Ontario, 1979.
4. Slater, DESIGN OF WIDE BAND,LINEAR PHASE, SURFACE ACOUSTIC WAVE FILTERS, McMaster University Thesis, Hamilton, Ontario, 1982.
5. Superior Electric, DESIGN ENGINEER'S GUIDE TO DC STEPPING MOTORS, Superior Electric Company, USA, 1976.
6. Usik, SYNCHRONOUS MOTOR FEEDBACK DESIGN, McMaster University Thesis, Hamilton, Ontario, 1982.