Y-1865

A TWO-AXIS MACHINE TOOL CONTROL WITH A DEDICATED MINICOMPUTER

T. L. Williams
J. H. Burkhardt, Jr
C. M. Lay
A. E. Stephens

UNION CARBIDE

OAK RIDGE Y-12 PLANT
OAK RIDGE, TENNESSEE

prepared for the U.S. ATOMIC ENERGY COMMISSION
under U.S. GOVERNMENT *Contract W-7405 eng 26*

DISCLAIMER

Reference to a company or product name does not imply approval or recommendation of the product by Union Carbide Corporation or the U.S. Atomic Energy Commission to the exclusion of others that may meet specifications.

Printed in the United States of America. Available from
National Technical Information Service
U.S. Department of Commerce
5285 Port Royal Road, Springfield, Virginia 22151
Price: Printed Copy $3.00; Microfiche $0.95

UC-38

Y-1865

A TWO-AXIS MACHINE TOOL CONTROL WITH A DEDICATED MINICOMPUTER

T. L. Williams
J. H. Burkhardt, Jr
C. M. Lay
A. E. Stephens

Oak Ridge Y-12 Plant

P.O. Box Y, Oak Ridge, Tennessee 37830

Date Issued - February 9, 1973

Prepared for the U.S. Atomic Energy Commission
Under U.S. Government Contract W-7405-eng-26

THIS PAGE

WAS INTENTIONALLY

LEFT BLANK

ABSTRACT

A control system for a two-axis turning machine was developed using a minicomputer. This endeavor was initiated in order to determine the assets and liabilities of stored part description, open-loop digital servos, and the overall concept of computerized numerical control (CNC) of a machine tool.

The resulting system displayed machining accuracies similar to standard commercial numerical control units which use finer pulse resolution and have analog servos with closed-loop feedback. The CNC system also exhibits better reliability and long-term position repeatability than the standard NC systems.

CONTENTS

SUMMARY

A two-axis, continuous-path, machine-tool control system was designed around a small, general-purpose digital computer. Unique features of the system include:

1. Elimination of the tape reader by way of storing sections of the part description in the computer memory and refreshing of the memory from an inexpensive magnetic-tape recorder between machining operations.

2. Precise contour machining using open-loop, digital servo motors.

3. Generation of command signals using a combination of computer software, computer hardware, and interface hardware.

4. A post processor to generate mathematically defined surfaces using the minicomputer with 4K of 12-bit words.

Production data gathered over an extended period reveal accuracy and reliability factors comparable with identical machines using commercially available numerical control units with a position resolution three times finer. The open-loop CNC also exhibits better long-term position repeatability.

INTRODUCTION

During 1969, a literature search was made on adaptively controlled machine tools and computer-augmented machine-tool systems. This investigation was made in an effort to determine if a contribution could be made to state-of-the-art development in adaptively controlled machining. As the search progressed it became apparent that "adaptive control" as technically defined by the machining industry (automatic optimization of metal removal rates) was not of prime importance to the Oak Ridge Y-12 Plant's[a] machining operations. Because of particular requirements, improvements in such items as accuracy, surface finish, and system reliability received much of the developmental attention. The search also revealed that if significant improvements in accuracy were to be achieved, the computer would most probably be the central element to be used through such techniques as dynamic-error compensation, surface-finish optimization through dynamic variance of feedrates and spindle speeds, and self-training techniques (correcting errors by monitoring past part inspection data). Another area that generated interest is "on-machine gaging".

Since much of the literature pointed to computer-assisted machine-tool systems (known as direct numerical control - DNC) an endeavor was initiated that would place a computer in direct control of a precision contouring machine tool. This approach was taken in order to gain some insight into the computer's assets and liabilities when used to command a machine tool. The system that was put into operation offers several unique features including such items as:

1. Interpolation performed by a combination of computer hardware and software, and interface hardware.

2. Open-loop digital control through use of electrohydraulic pulse motors for slide movement.

3. System ability to use any pulse value. (Present system has a pulse value of 65.1041666.. microinches of slide movement.)

4. Elimination of reading the part-description data from tape during the machining cycle.

5. Lead-screw error compensation with software.

The CNC unit is in regular production along with a group of commercially available NC systems with phase-analog servos. These controllers are used on machine tools that are identical to that of the CNC and have a pulse value that has a resolution that is three times finer (20 microinches). Inspection data reveal that parts machined by way of the CNC offer contour accuracies comparable to the standard NC. The CNC is not as prone to "run away" as the NC, and indications are that the CNC is more reliable. The CNC also displays better long-term repeatability than standard NCs.

(a) Operated by the Union Carbide Corporation's Nuclear Division for the US Atomic Energy Commission.

A TWO-AXIS MACHINE TOOL CONTROL SYSTEM

PRINCIPAL COMPONENTS

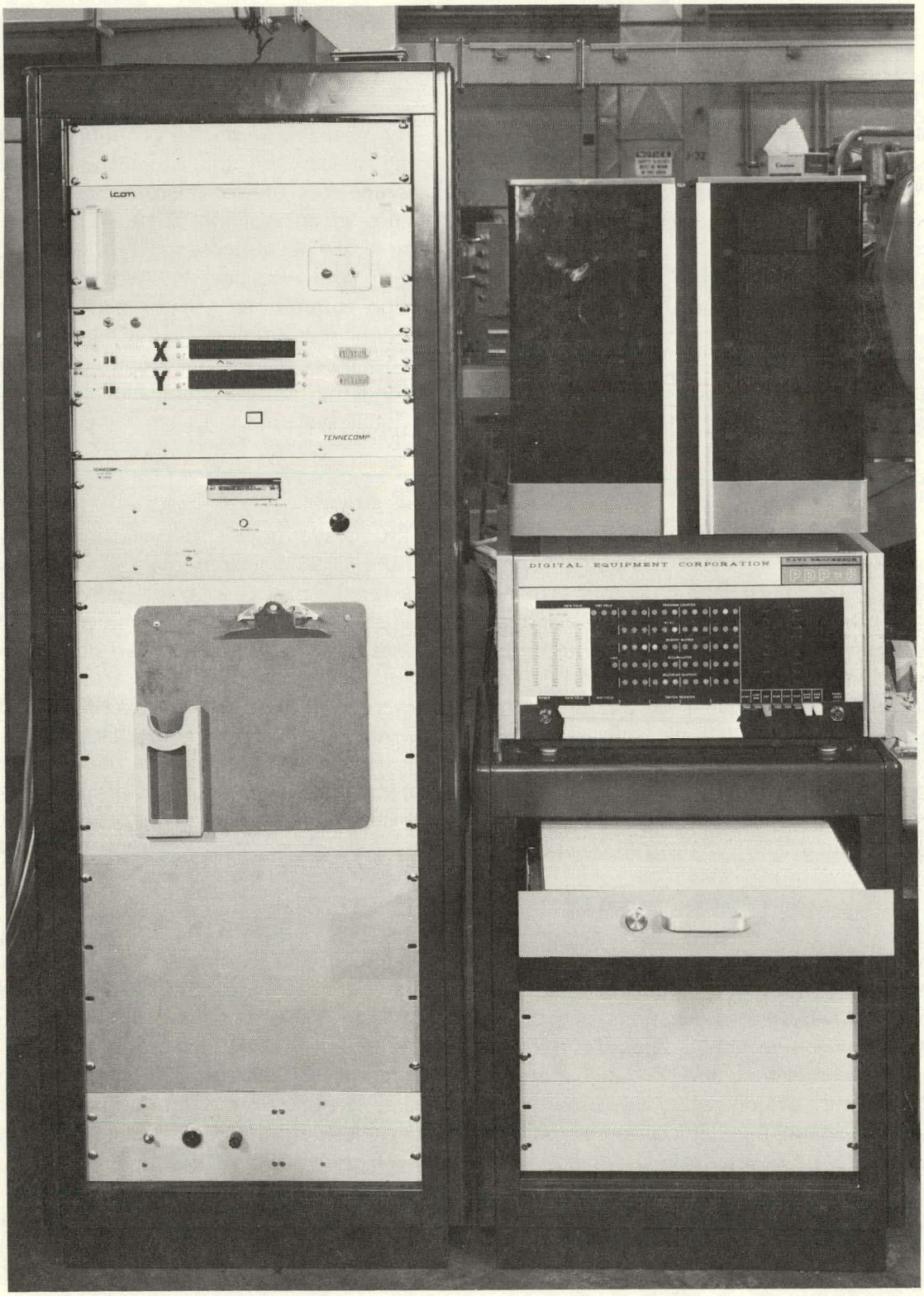
The CNC, seen in Figures 1 and 2, does not by any means represent an optimum design. Because of the uncertainties involved in attaining success on the project, no major mechanical modifications could be made to the machine tool, and no major capital outlays were allowed. These restrictions necessitated the use of an available Digital Equipment Corporation (DEC) PDP-8 computer (which at that time had been succeeded by the PDP-8S, PDP-8I, and PDP-8L) and a cartridge-type magnetic-tape recorder that was previously purchased for another project. Because of production commitments, gear changes in the machine tool were not allowed. With the electrohydraulic pulse motors (EHPM) and gearing that was supplied with the machine, slide movement per pulse was 65.104166.. microinches. This odd pulse value was responsible for complicating the entire system and eliminating some desirable features that would provide a more reliable system.

Digital Computer

The computer[1] is a single-address, fixed-word-length, parallel-transfer device using a 12-bit word and 2's complement arithmetic. Cycle time of the 4,096-word random access memory is 1.5 microseconds. The computer provides storage for the part description being actively used as well as controlling a cartridge-type, magnetic-tape recorder where the entire part description resides. The computer performs such functions as searching the part description for sequence numbers, monitoring slide error, providing correction when the error exceeds one half the pulse value (32.5320833.. microinches), and generating part description data for mathematically defined surfaces from teletypewriter input or from coordinates that are stripped from APT part description tapes. The computer also acts as part of the interpolator. Its data-break facilities are used to increment two memory locations which act as counters for accumulating the number of command pulses generated by the interface feed-rate circuits.

Automatic Loader-Magnetic Tape Recorder

One of the prime considerations during initial development was to eliminate the paper-tape reader and machining from part-description data stored in the computer's memory. However, because of limitations in the core memory size it was necessary to store the part description on magnetic tape. A four-track tape-recorder system (Tennecomp TP-1346 automatic loader[2] and TP-1351 magnetic tape recorder[3]), designed for use with the PDP-8 family of computers, was used for this purpose. Part description data are broken into four convenient parts if necessary and recorded on tape. Before any machining operation occurs, its part description is read into memory from which all machine movements originate. Storing the contents of a tape track in memory requires that a "boot loader subroutine" be inserted into the computer which requires several manual switch operations. This process is eliminated by the automatic loader. By selecting the tape track and depressing the "load" button, the "boot loader" is automatically inserted in memory, the selected track read into memory, and checks are made for errors. If none occur, the computer enters the "machine control program" from which all automatic machining functions are derived.



139591

Figure 1. MACHINE TOOL CONTROL SYSTEM.

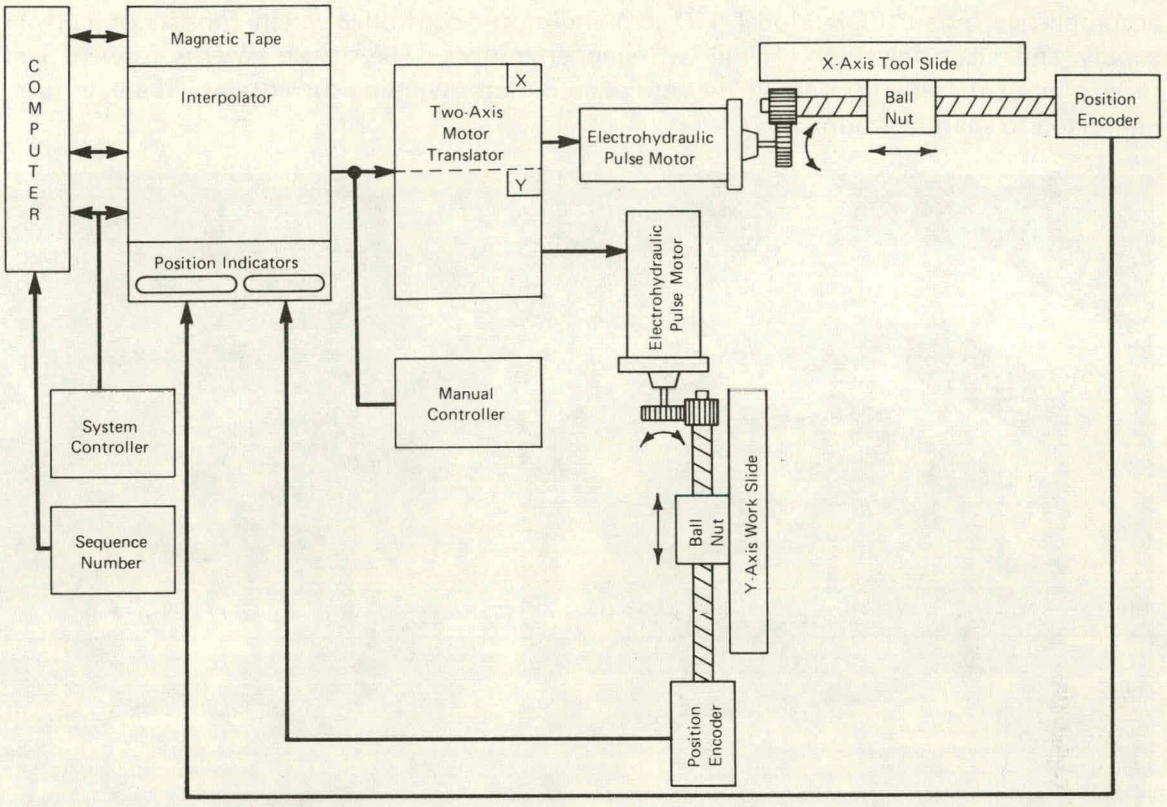


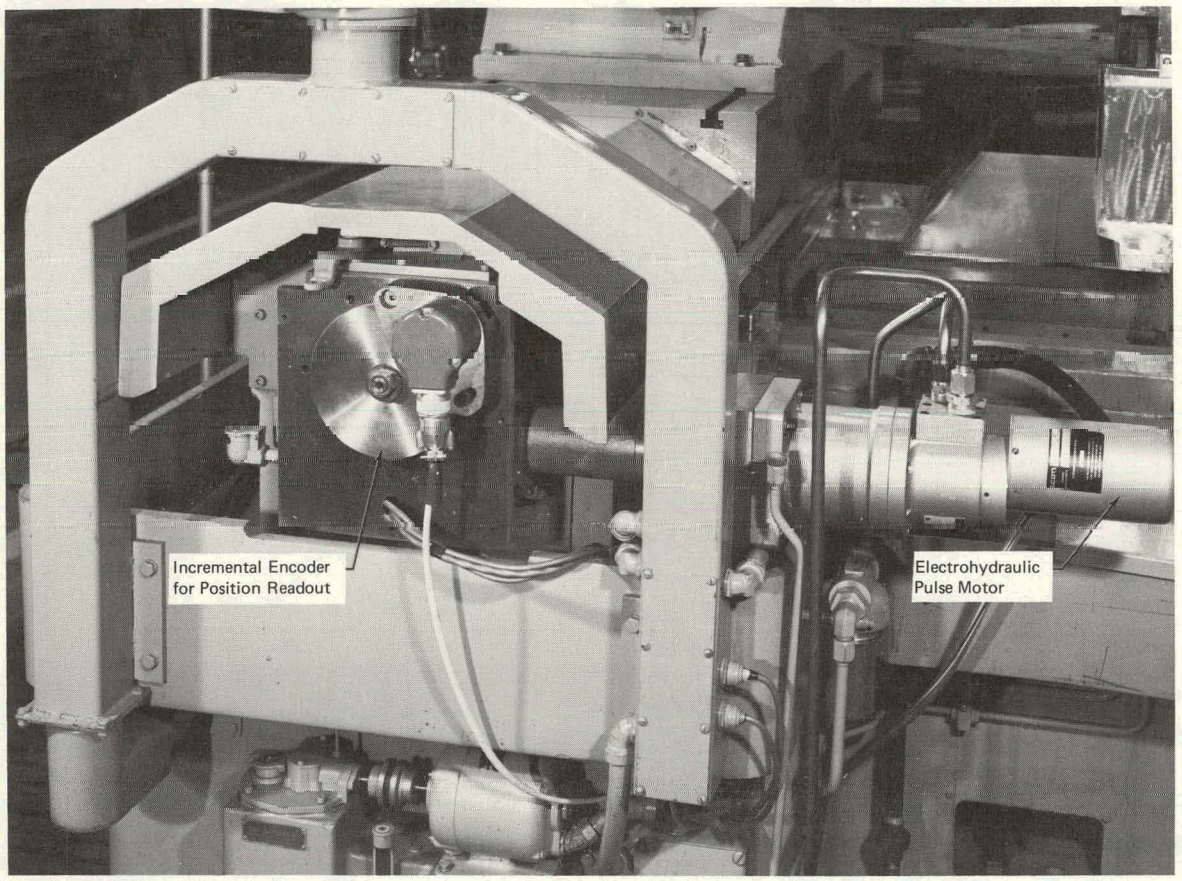
Figure 2. PRINCIPAL COMPONENTS OF THE CONTROL SYSTEM.

Electrohydraulic Pulse Motor and Motor Translator

Another important consideration in the design was to keep each part as simple as possible while retaining or improving the characteristics of equivalent commercial NCs. The phase-analog servos that are normally used for precision positioning are quite complex and unforgiving in that they will quickly respond to any input signal, whether it is correct or incorrect. With these factors in mind, electrohydraulic pulse motors (EHPM)[4,5] were used to provide slide movement (Figure 3). Basically an ultra-high power stepping motor, the device combines a high-speed hydraulic torque amplifier with a small electric pulse motor. Each step of the small electric motor is transformed to 1.5 degrees of rotation at the output shaft. With the twelve-to-one gear ratio in the machine tool and a four-turns-per-inch lead screw pitch, slide movement per step is 65.104166.. microinches. Logically, the machine gears would be changed to provide an even pulse value such as a 20, 50, or 100-microinch movement per pulse. This alteration was not permitted, however, because gearing changes require major machine modifications which would have to be restored to the original condition if the CNC project was not successful. This odd pulse value contributed to the complexity of the system as well as eliminating use of a scheme for error checking, making the system less reliable.

Rotation of the electric stepper motor is obtained by translating command pulses from the interpolator to properly phased currents in the five motor windings. This action is

accomplished by an ICON Model 410 stepping motor controller which consists of a power supply and electronics for driving two stepper motors. The device accepts forward and reverse input pulses and converts these to phased motor winding currents which are, in turn, converted to shaft rotation.



139592

Figure 3. ELECTROHYDRAULIC PULSE MOTOR USED FOR DRIVING BOTH AXES.

Precision Turning Machine

In order to properly evaluate the CNC, it was necessary to have access to a high-precision production machine. Fortunately, an Ex-Cell-O 922 turning machine, awaiting installation of a control system, was made available. The machine has 22 inches of travel on each of its two axes and a positioning accuracy of ± 200 microinches when driven by phase-analog servos. It also has a fluid-cooled ball bearing, hydraulically driven spindle. Normally, lead screw errors are corrected by a cam that rotates the servo's resolver feedback device. Since the CNC has no feedback, a different arrangement was necessitated. The correction cam was removed and slide errors were measured using a laser interferometer. These errors were then used to offset positioning data during generation of the part description. The only machine modifications were the installation of an adaptor plate for the EHPMs and an encoder assembly (Figure 3) for monitoring the slide position.

Computer/Machine Tool Interface

Interpolator - It is desirable to understand the interpolator's operational theory before further discussion. A simplified version of the two-axis interpolator is illustrated in Figure 4.

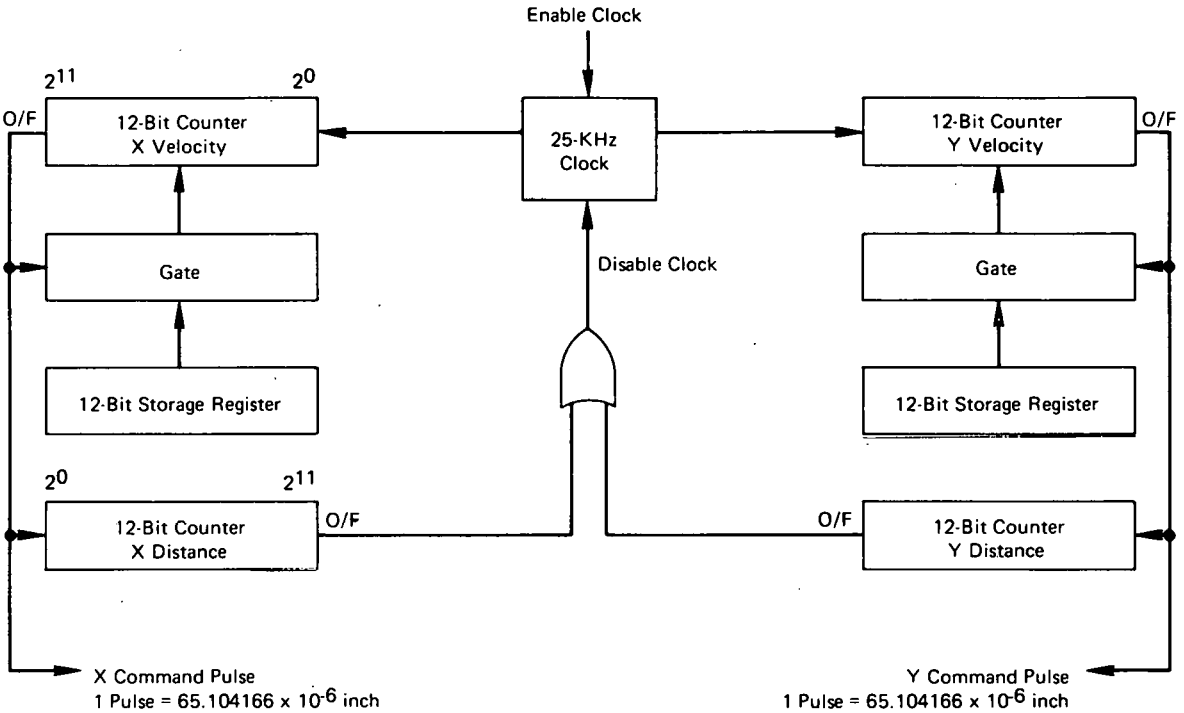


Figure 4. PRINCIPAL COMPONENTS OF THE SIMPLIFIED INTERPOLATOR.

Assume that it is being used to move the machine's tool over the path outlined in Figure 5 and that one command pulse is equal to 65.104166.. microinches of slide movement. The change in X and Y (△X and △Y) may be defined in terms of motor pulses as follows:

△X = △Y = Distance ÷ Pulse Resolution, or (1)

△X = △Y = 0.1 inch ÷ 65.104166.. microinches/pulse, or

△X = △Y = 1,536 pulses.

Feedrate may also be redefined as follows:

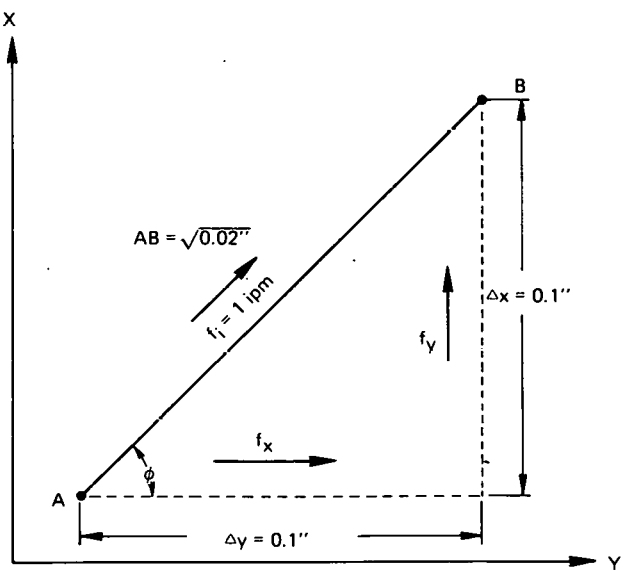


Figure 5. REPRESENTATIVE TOOL PATH.

$$\phi = \tan^{-1} \Delta X/\Delta Y = 45^{o}, \text{ and}$$

$$f = f_i \text{ in/min} \div [(60 \text{ sec/min})(65.104166.. \text{ microinches/pulse})], \text{ or} \quad (2)$$

$$f = 1 \div [60 \times 65.104166.. \times 10^{-6}] \text{ pulses/sec (pps), or}$$

$$f = 256 \text{ pps;}$$

$$f_y = f \cos \phi, \text{ or} \quad (3)$$

$$f_y = 256\ (0.7071), \text{ or}$$

$$f_y = 181.0176 \text{ pps;}$$

$$f_x = f \sin \phi, \text{ or} \quad (4)$$

$$f_x = 256\ (0.7071), \text{ or}$$

$$f_x = 181.0176 \text{ pps.}$$

If overflow pulses occur from each of the velocity counters (Figure 5) at the rate defined by f_y and f_x (181.0176 pps), the tool will follow Path AB. This rate can be accomplished by first loading a negative number into each velocity counter and incrementing the counters each time a pulse is generated by an oscillator whose frequency is constant. Assuming that the correct number was loaded into the velocity counter, an overflow will occur at the most significant counter bit when the counter contents reach zero. If the same negative number is reloaded each time the overflow occurs, command pulses will be generated at the proper rate with fixed spacing. It was previously determined that overflows were needed at a rate of 181.0176 pps. If this number is rounded, no significant feedrate change will occur. At 181 pps, an overflow is needed every:

$$T_{OF} = 1/f, \text{ where} \quad (5)$$

$$1/f = 1/181 = 5.53 \times 10^{-3} \text{ second.}$$

To determine the correct number that must be loaded into the velocity counter, T_{OF} is divided by the time between clock pulses (t_c), giving the exact number of clock pulses between velocity counter overflows, or:

$$N - T_{OF} \div t_c, \text{ or} \quad (6)$$

$$N = 5.53 \times 10^{-3} \text{ sec} \div 40 \times 10^{-6} \text{ sec, or}$$

$$N = 138.3.$$

Because the counters are binary, no number less than one can be resolved, but this limitation is insignificant in that it represents a worst-case T_{OF} change of 40 microseconds. By

rounding and recalculating, the time between overflow pulses is: $t_x = t_y = 138 \times 40 \times 10^{-6} = 5.2 \times 10^{-3}$ second, giving a feed rate of:

$$f_x = f_y = 1 \div t_x = 1 \div 5.2 \times 10^{-3} = 181.3 \text{ pulses per second.} \quad (7)$$

The desired feedrate is generated by initially loading the binary complmenet of N [138(10)] in each velocity counter and storage register and enabling the clock (Figure 5). Each time an overflow occurs at the counter's most significant bit, each slide is moved 65.104166.. microinches and -138(10) is reloaded into the velocity counters. Now the tool is forced to follow the path defined by AB as required, but the interpolator must detect when the tool arrives at B. It was previously calculated that B is attained after 1,536 command pulses are generated. The destination is detected by the same counter concept where the complemented binary equivalent of 1,536 is loaded into a distance counter which is incremented once for each command pulse (Figure 5). Actually, these counters are memory locations inside the computer that are controlled by a data break circuit which also generates a computer interrupt and "stop clock" signal when either memory location overflows. This action signifies that Line AB is complete and the interpolator needs new data.

A block diagram of the complete interpolator is presented in Figure 6, with detailed schematics provided in Appendix A. Circuits that were not previously discussed are the repeat counter, error corrector, accelerator-decelerator, and sign control.

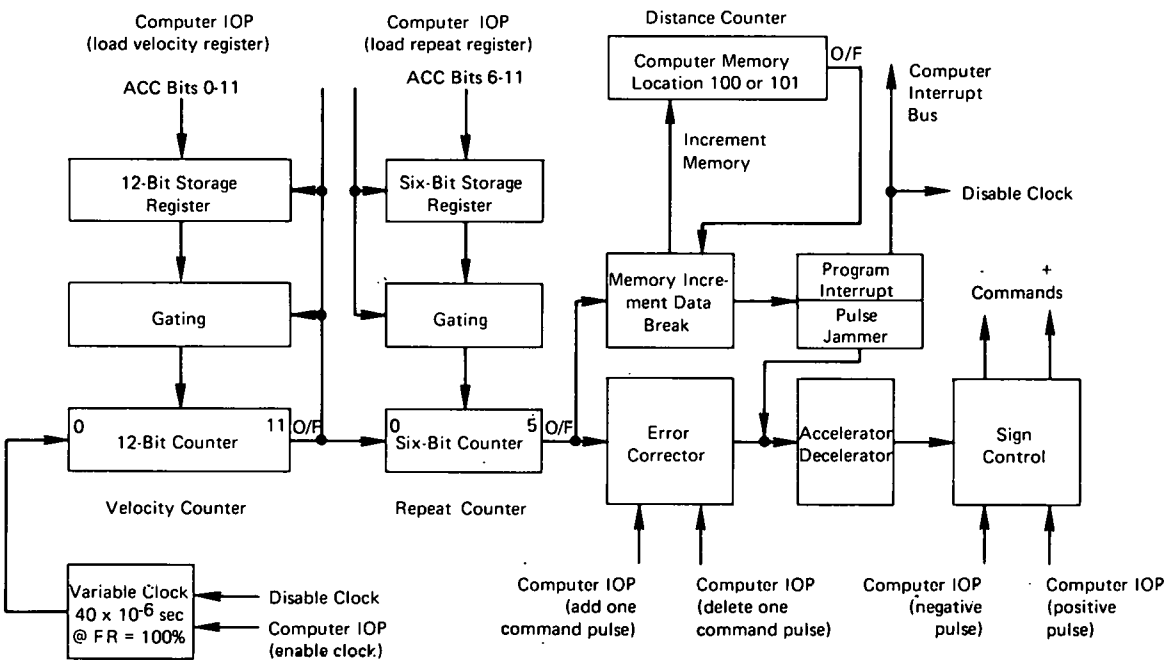


Figure 6. PRINCIPAL COMPONENTS OF THE INTERPOLATOR.

Repeat Counter - With the 12-bit resolution of the velocity counter (VC), it is impossible to describe tool paths with slopes less than 490×10^{-6} degree and feed-

rates less than 0.024 inch per minute. This range was extended by adding a 6-bit repeat counter (RC) which withholds command pulses from the VC until a predetermined number of overflows have occurred. This procedure is accomplished by loading the negative number of repeats into the RC and allowing the overflows from the VC to increment the RC. When it reaches zero, an overflow occurs that is used as the command pulse. With this arrangement, feedrates as low as 380 microinches per minute may be described with tool-path slopes of 7.7×10^{-6} degree.

Error Corrector and Pulse Jammer - Very few tool-path slopes come out with even numbers as in the previous example. In most cases the slope that is generated will always have an error. For example, Path AB of Figure 7 is described as follows:

Using Equation 1:

$$X = OB = 0.025 \div 65.104166.. \times 10^{-6} = 384 \text{ pulses, and}$$

$$Y = AO = 0.06 \div 65.104166.. \times 10^{-6} = 921.6 \text{ pulses.}$$

Using Equation 2:

$$f = 0.21 \div (60)(65.104166.. \times 10^{-6}) = 53.76 \text{ pps.}$$

Using Equation 3:

$$f_y = 53.76 \cos \phi = 49.6258 \text{ pps.}$$

Using Equation 4:

$$f_x = 53.76 \sin \phi = 20.674 \text{ pps.}$$

Now the number of clock pulses are found (number that is loaded into each velocity counter).

Using Equations 5 and 6:

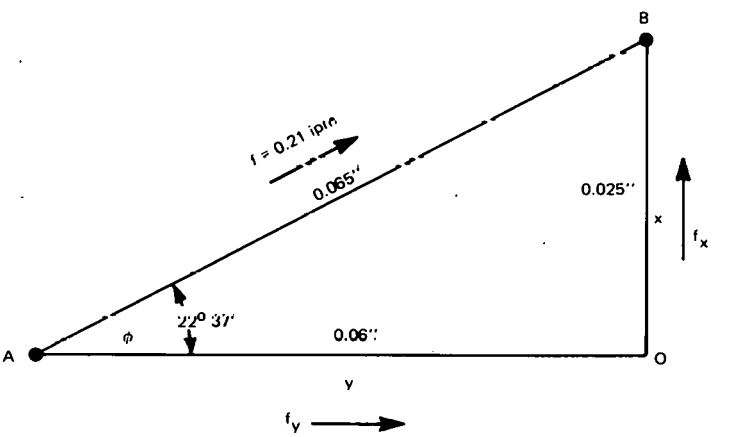


Figure 7. TYPICAL TOOL PATH.

$$N_x = T_{OF} \div t_c = 1/f_x \div t_c = \frac{1}{(f_x)(t_c)} ,$$

$$N_x = 1 \div (20.674)(40 \times 10^{-6}) = 1{,}209.25 \text{ clock pulses.}$$

$$N_y = 1/f_y \div t_c = (1/49.6258) \div 40 \times 10^{-6} ,$$

$$N_y = 503.77 \text{ clock pulses.}$$

The smallest number (N_y) representing the fastest feedrate, is rounded, and N_x is recalculated using the same slope:

$$N_x = N_y \tan (90 - \phi) = 504 \tan (90^o - 22^o\ 37'), \text{ or}$$

$$N_x = 504 \tan 67^o\ 27', \text{ or}$$

$$N_x = 1{,}209.8 \text{ clock pulses.}$$

The complemented binary equivalent of 504 is loaded into the Y velocity counter; and, after 1,209.8 is rounded to 1,210, its complemented binary equivalent is loaded in the X velocity counter. Now each time an X command pulse is generated, there will be a timing error of 0.2 clock pulse or 8 microseconds. This error is continuously monitored by the computer's "operate program" which signals the error corrector (Appendix A, Figure A-1) to either add or withhold a command pulse when the distance exceeds one half the command pulse (32.5520833.. x 10^{-6} inch). Another problem that arises is that the last command pulse from each axis should occur simultaneously. This never occurs, with the interpolation concept used, except for very special cases such as the 45-degree tool path of the first example. Simultaneous pulses are forced by the pulse jammer which monitors each interpolator output; and, upon a program interrupt (generated when either axis' distance counter reaches zero), inserts a pulse in the axis where one is not present.

Previously, when Y was calculated using Equation 8, the result was 921.6 pulses. Obviously the system cannot output 0.6 pulse, but this error is eliminated by moving the Y slide 922 pulses and adding the -0.4 pulse error to the next data block when the part description is being generated by the post processor for all cases except where the block is repeated. In this case, the operate program monitors this error each time the block is repeated and when it is equal to one-half pulse (32.5520833.. x 10^{-6} inch) makes a correction. The residual error that is present on the last block repeat is accounted for by the post processor.

Memory Increment Data Break and Program Interrupt - With this circuit (Appendix A, Figure A-4), the contents of a specified location in memory is read into the memory buffer, incremented by 1, and rewritten at the same memory location. If the content of this register becomes 0 as a result of the addition, an overflow pulse will be transmitted to the interface electronics. A more detailed analysis of data breaks may be found by referring to the computer's handbook.[1] Actually, the device serves as the two 12-bit binary distance counters whose function was

described earlier. In operation, the counters are loaded with the complement of the number of command pulses to be generated by each interpolator. As soon as either counter overflows, a program interrupt disables the variable clock (Appendix A, Figure A-1), preventing further generation of command pulses, and signals the computer to load a new block of part description data.

Accelerator-Decelerator - It is possible to have an instantaneous 8 KHz pulse train at the velocity and repeat counter output. If this pulse train were fed directly to the motor translator, the electric stepper motor would fail to respond (maximum startup pulse rate is 2,000 pps). Instead, these pulses are processed by the circuit (shown in Figure 8 and Appendix A, Figures A-5 and A-6) which outputs these command pulses in a manner that will properly start and stop the motors. The circuit consists of a 10-bit up/down counter, a 10-bit digital-to-analog converter (DAC), a 10-bit gate for sensing zero in the up/down counter, and a voltage-to-frequency converter (VFC) for generating command pulses. Command pulses from the velocity-repeat counter are fed to the "up" input of the up/down counter. The first input pulse sets the counter to "one" which enables the VFC via the 10-bit gate. Since the DAC output is approximately 0 volt, the resulting output frequency is 1.8 KHz. This pulse train is fed to the sign controller which routes the signal to the proper translator input and also to the "down" side of the counter. Assuming that the input frequency is 8 KHz, several more input pulses will have arrived while the one pulse was emitted by the VFC. These pulses add to the contents of the counter, increasing the DAC's output voltage which also increases the VFC's output frequency. Now the command pulses are coming at a faster rate than before, accelerating the motor and decrementing the counter at a faster rate. This process continues until the VFC's output frequency matches the input, exponentially accelerating the motors. As the motors are accelerated, pulses are withheld from the accelerators (the number of withheld pulses is proportional to the input frequency). The exact number of these pulses is stored in the counter. When the input pulse train stops, the motors must be decelerated. Since the counter contains the withheld pulses, the VFC is enabled. Its next output pulse moves the motors and decrements the counter, reducing the DAC's output voltage and VFC frequency. This process continues, decelerating the motors until the contents of the counter are zero. At this time the number of command pulses used during deceleration is equal to the number that were withheld during acceleration. With this arrangement, the accelerator is not employed until the input pulse rate exceeds 1,800 pulses per second. Any rate lower than this will enable the VFC, but it will output a pulse (decrementing the counter whose contents return to zero, disabling the VFC) before the next input pulse arrives.

This circuit provides adequate performance as long as machining pulse rates remain below 1.8 kc (7 inches per minute) where the accelerator-decelerator is only used for rapid traverse. When machining at rates above 1,800 Hz, problems are encountered because of nonlinearity and stability problems in the VFC. These difficulties could be eliminated by using a better VFC with facilities for precise calibration for matching the response of both axes.

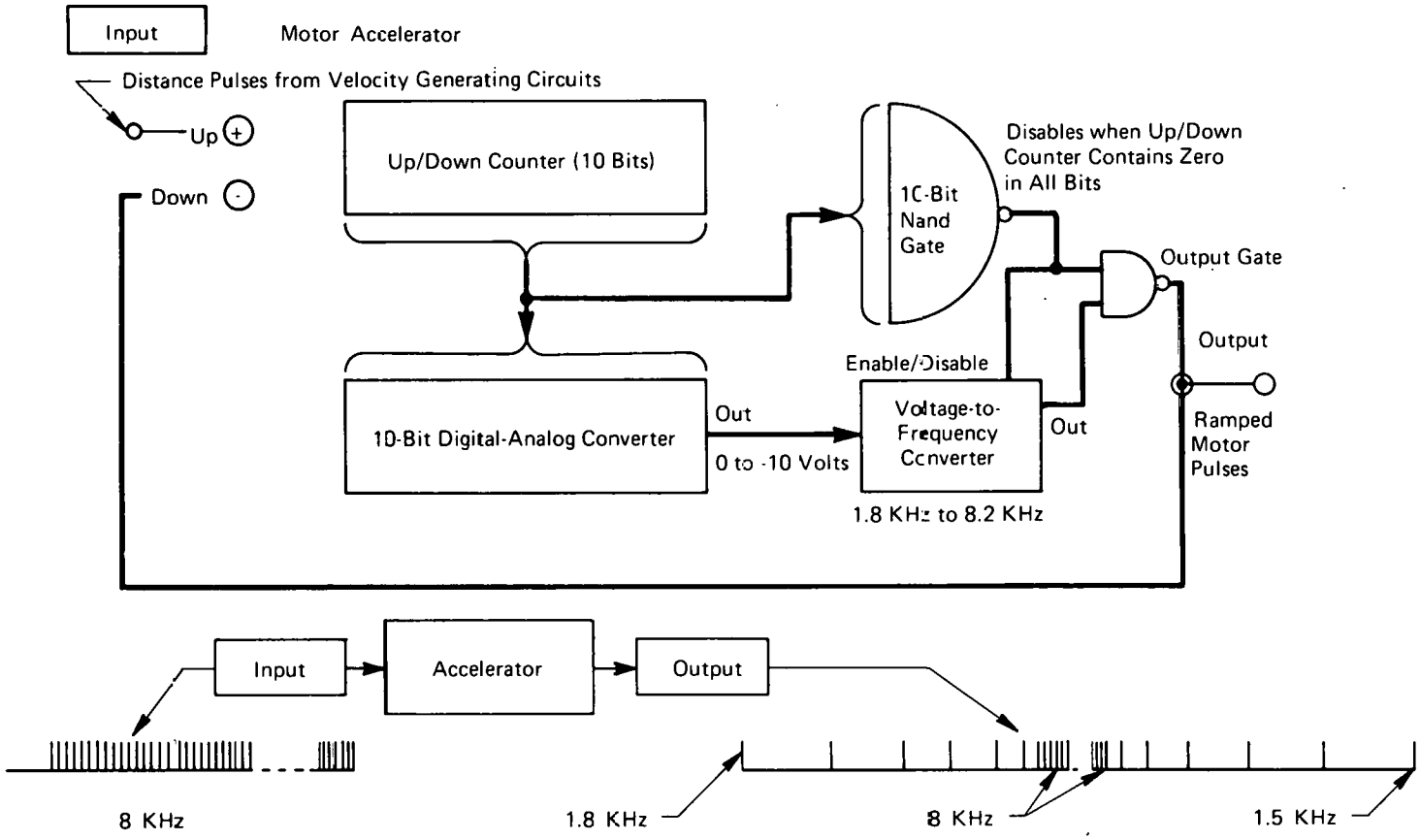


Figure 8. SLIDE ACCELERATOR.

Sign Control - Each block of part description data (Figure 9) contains a sign bit for each axis. These items are used to control sign flip-flops (Figure 10 and Appendix A, Figure A-1) which enable gates that allow command pulses through to the proper translator input. The flip-flop's state is controlled by computer input/output pulses (IOTs) that are generated when a block of data is being loaded into the interpolator (Appendix B, Table B-1).

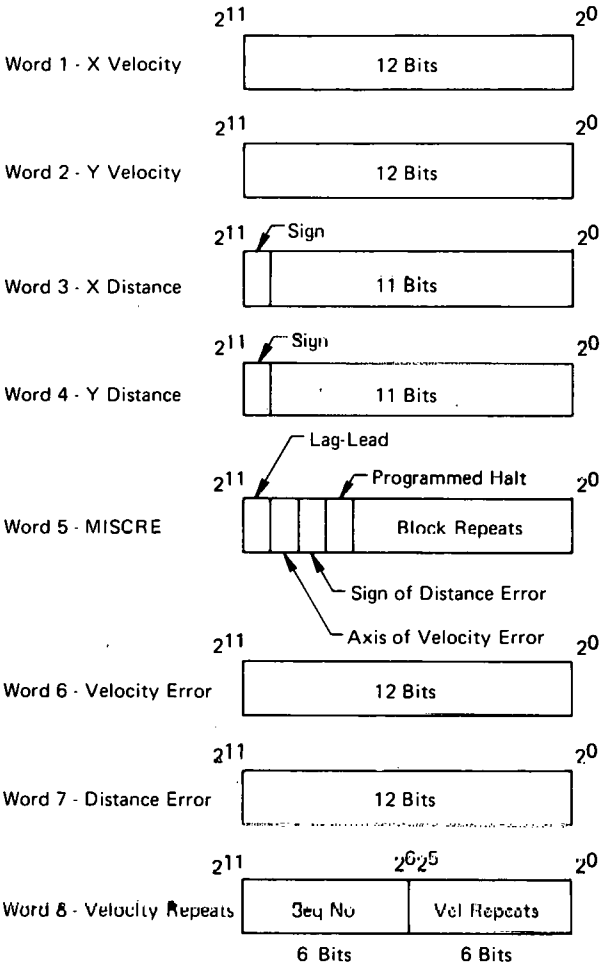


Figure 9. PART DESCRIPTION DATA.

Interpolator Operation - Assuming that the system has just started operating from a previous block of data, the operate program has retrieved the next block of information and inserted it into temporary storage. The program now strips out velocity repeats (Figure 9 shows a data block), block repeats (storing them for later use), and checks to determine if this is the last block of part description. If not, the sign bits are examined and used to modify the operate program to generate the correct sign IOTs when the block is loaded into the interpolator. The operate program then turns its attention to the information being currently processed, monitoring velocity and distance errors, and making the required corrections. When an interpolator distance counter overflows (signifying completion of the block), an interrupt is generated that forces the program flow to an interpolator load subroutine and jams a command pulse on

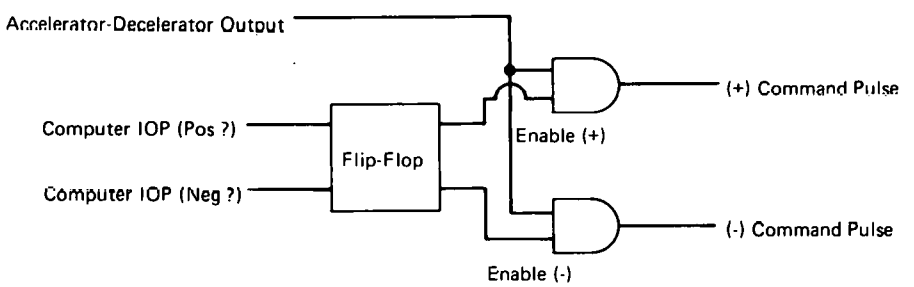


Figure 10. SIGN CONTROL.

the other interpolator output (insures that both axes reach the destination simultaneously). The subroutine checks each distance counter content for zero. If this is not the case (signifying an incorrect data block or missed command pulse) the computer halts; otherwise, the information in temporary storage is loaded into the interpolator. X and Y velocities are loaded into the velocity counter storage, X and Y distances are loaded into the distance counters, velocity repeats are loaded into the repeat storage registers, error constants (velocity error, axis of velocity and distance error, whether it lags or leads, sign of distance error) are prepared for proper error correction, the variable clock is enabled (starts interpolation), and the next block of data is loaded into temporary storage.

Position Indicators - Slide position is monitored by gearing pulse encoders[6] to the lead screws in a manner such that each pulse represents 50 microinches of slide movement (Figure 11). These signals are counted on a bidirectional counter[7] with the first decade modified to operate as a binary counter with a readout of 0 and 5. The position indicators are not to be confused with a feedback system. As stated, it is only used to visually show the position of a slide.

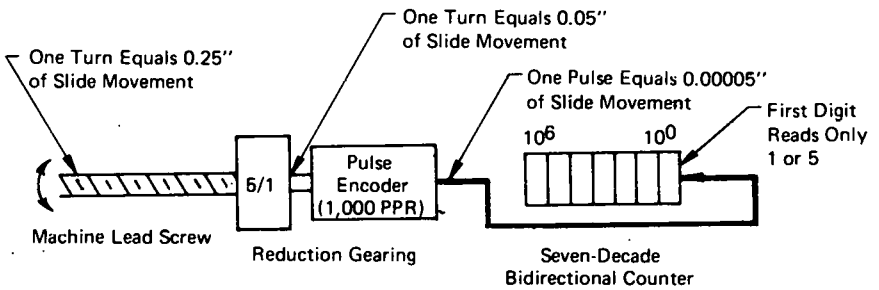


Figure 11. POSITION INDICATOR.

Sequence Number Search - During machining operations it is sometimes necessary to search out sections of the part description and start at these points. This search is accomplished by setting aside a location in each data block (Figure 9) for storage of a "sequence number". If the program control is to start at this point, the number is set in the "seq no" decade switches on the control panel, the "start cycle" is depressed, and then "load seq". This operation generates a signal that forces the computer to enter a subroutine that finds the number's location in the part description. Upon entry, the contents of the switches are read into the computer where the decade number is converted to binary. This conversion is necessary in order to compare the switch contents with each block sequence number (sequence numbers are stored in binary in order to conserve memory). The subroutine searches the part description. When the number is found, operate program constants are changed so that this part description location will be the initial data block. Schematics of the interface electronics that pertain to the sequence number input are provided in Appendix A, Figures A-8 and A-9.

System Controller - Normally, the computer is started by manually loading the initial binary address of the operate program into the switch register and successively depressing the "load address" and "start" switches. Nontechnical personnel can easily make mistakes during this

procedure which requires three separate manual operations. The controller (Appendix A, Figure A-9) electronically reproduces these operations when the "cycle start" switch is depressed. The controller also generates the "continue" function which again initiates the system operation after a programmed halt. Switches are also available to manually start and stop the interpolator's variable clock during a machining operation.

Manual Slide Control - Very often it is necessary to move the machine slide over long distances at high feedrates. These movements are achieved by injecting pulses from a variable clock in the slide accelerator input. The clock rate is selected from a switch (Appendix A, Figure A-8) that changes the clock's frequency determining capacitor. Slide direction also originates from this switch where a second deck is used to control the state of the sign control flip-flop of Figure 10. Manual feedrates vary from about 0.04 to 25 inches per minute.

When machining, it is necessary to offset the machine's slides to compensate for tool wear, part deflection, and other errors that are invariably introduced during manufacture of a part. This adjustment requires that an absolute number of control pulses (determined by offset) be fed to the axis on which the offset is desired. Generally, offsets are 0.0001, 0.0005, 0.001, 0.01, and 0.1 inch. With the pulse value used, these increments are not readily available. Table 1 lists the offsets provided by the circuits shown in Appendix A, Figure A-7. Command pulses are generated by a clock that is also used to increment an 11-bit counter. Five sets of gates (one for each offset) monitor the counter outputs and provide a clock disable (shutting of flow of pulses) after the selected number of control pulses is outputted.

Table 1

AVAILABLE OFFSETS

Offset (inch)	Number of Pulses
0.000065	1
0.000521	8
0.000977	15
0.010026	154
0.1	1,536

System Software

Post Processor - The post processor is actually a part description program for use on the machine tool computer. It is used to calculate and output part description data in a format that conforms to the system requirements. Specifically, this software reduces the part contour to straight-line segments while maintaining a specified contour accuracy. Each segment is defined by the data block shown in Figure 9. The program will accept information from either teletypewriter input or from a standard EIA NC tape with coding for either straight line or arcs. Since most of this particular machine's parts contain coordinate-defined contours, the normal operation involves the use of APT-generated EIA NC tape as an input medium. The processor strips all appropriate information from the tape (eg, X and Y coordinates, feedrates, sequence numbers, and M codes) and from these data generates a block of data for each line segment described on the tape. Words 1 and 2 of the block are loaded into the velocity-generating circuits for establishing proper feedrates; Words 3 and 4 contain the exact number and sign of the pulses to be outputted on each axis and are loaded into Memory Locations 100(8) and 101(8). Word 5 contains the number of times the block is repeated, bits for executing a programmed halt, and error definition bits that are used to correct for velocity and distance errors. Words 6 and 7 contain binary numbers that are equivalent to the error being produced. As each error takes place, these numbers are added to themselves. This addition is monitored by the operate program and, when an overflow occurs (signifying a one-pulse

error), corrective measures are taken. Word 8 contains the sequence number if there happens to be one as well as the number of velocity repeats. When present, this number is loaded into the interface velocity repeat counters to acquire the desired feedrates. A more detailed analysis of the post processor is given in Appendix B.

Operate Program - This program is one that resides in the computer memory during all machining operations, and consists of various subroutines that control the previously discussed hardware as well as performs the necessary corrections that keep the tool path within 1/2 pulse of the desired position. The subroutines perform such functions as retrieving the prescribed block of part description information from memory and inserting it into temporary storage, stripping error information from the data block and modifying its own subroutines so that the proper adjustments are made, loading the velocity counters from the data block, running tests to determine if the system is operating correctly, loading the software distance registers from the data block, controlling the direction of both axes, and executing block repeats when required. Further information on the operate program is given in Appendix B.

CONCLUSIONS AND RECOMMENDATIONS

The CNC system has been in full operation approximately two years. The system continues to display contouring accuracies comparable to identical machines with standard NC controllers having 20 microinches of control resolution, the CNC is less prone to require maintenance because of its simplicity through elimination of the tape reader and use of the open-loop digital servo system. The controller also offers better long-term repeatability because there are no critical gain adjustments that change with time and temperature. Another important factor that should be evaluated is the reluctance of the CNC to cause machine "run away" where the slides move at an extremely fast rate because of malfunctions on the control system. This problem is minimized by the failure of the motors to respond to high pulse rates and the fact that no analog control signals are present. Considering the age of some system components and the fact that this was a developmental efforts, these results are highly encouraging. A more detailed description of the system operation is presented in Appendix C.

REFERENCES

(1) *Small Computer Handbook C-800*; Digital Equipment Corporation, Maynard, Massachusetts (1966-1967).

(2) *Installation and Operating Instructions, Tennecomp TP-1346 Automatic Loader*; Tennecomp, Inc, Oak Ridge, Tennessee.

(3) *Installation and Operating Instructions for the PDP Family-of-Eight Computers, Tennecomp TP-1351 Magnetic Tape Storage Unit*; Tennecomp, Inc, Oak Ridge, Tennessee.

(4) *Fujitsu Electrohydraulic Pulse Motors*; ICON Corporation, Cambridge, Massachusetts.

(5) *Instruction Manual for the Model 410-22*; ICON Corporation, Cambridge, Massachusetts.

(6) *DRC Model 29 Optical Shaft-Angle Encoder*; Dynamics Research Corporation, Stoneham, Massachusetts.

(7) *Model 2400 Bi-Directional Multi-Function Totalizer*; Alec, Inc, Houston, Texas.

ACKNOWLEDGEMENTS

The authors wish to thank F. W. Jones of the Y-12 Fabrication Engineering Department, Fabrication Division, for the administrative support that made this endeavor possible, and F. B. McDonald and G. P. McGhee of the Fabrication Division for their assistance in supplying the machine and manpower necessary to implement this endeavor.

APPENDIX A

SCHEMATIC DIAGRAMS

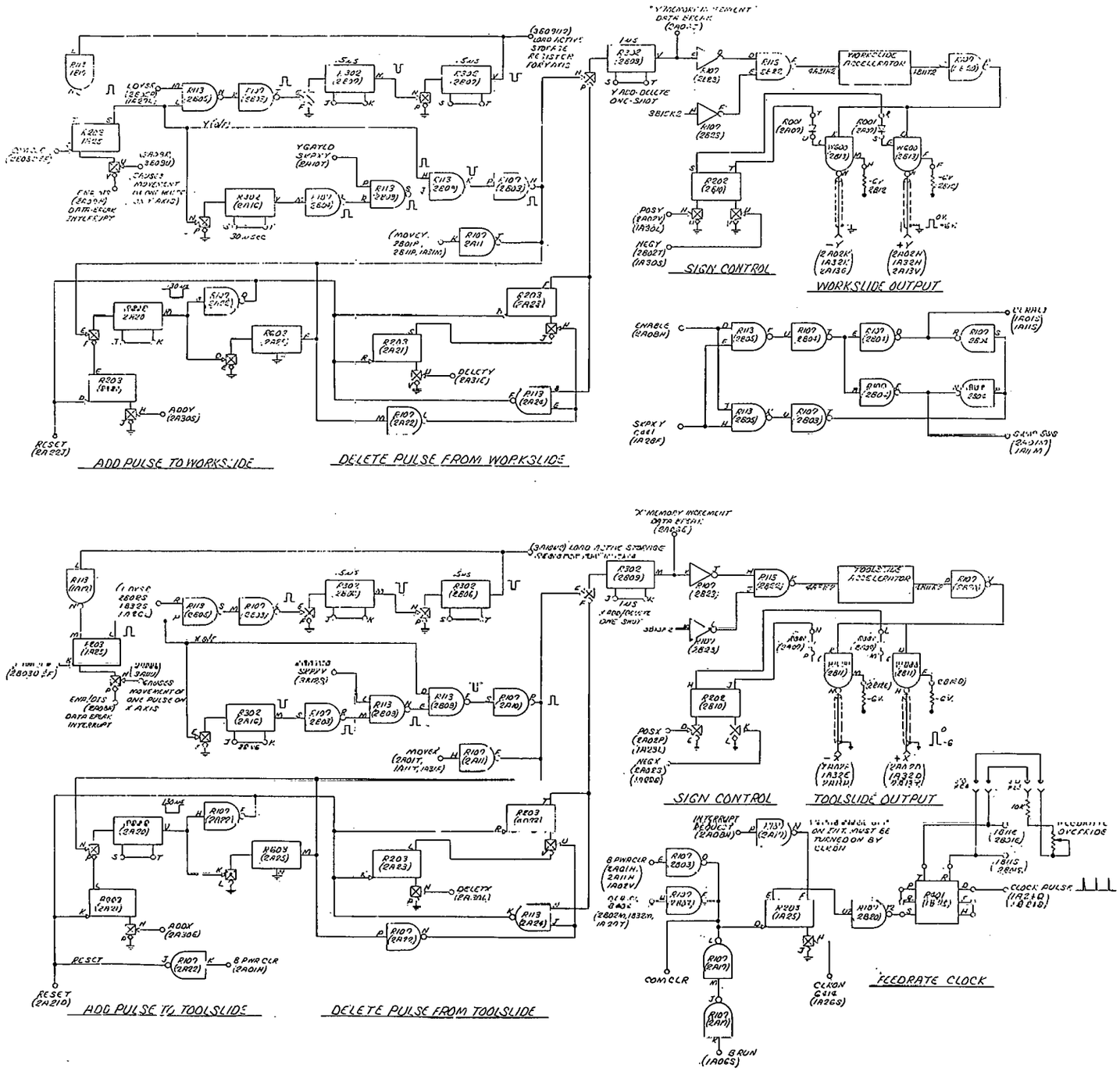


Figure A-1. MACHINE TOOL CONTROL. (Translator Controller)

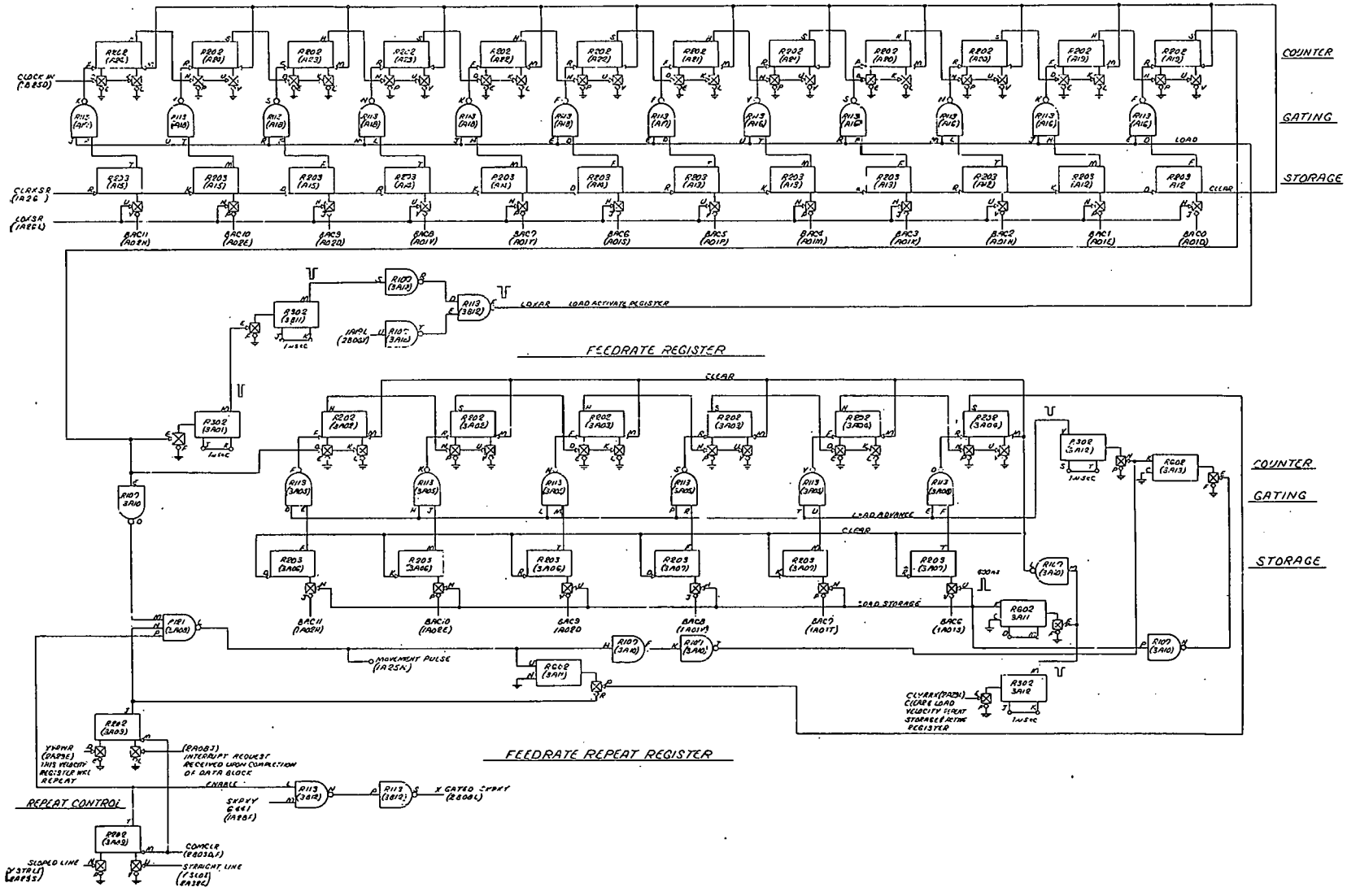


Figure A-2. MACHINE TOOL CONTROL. (Tool Slide Pulse Generator)

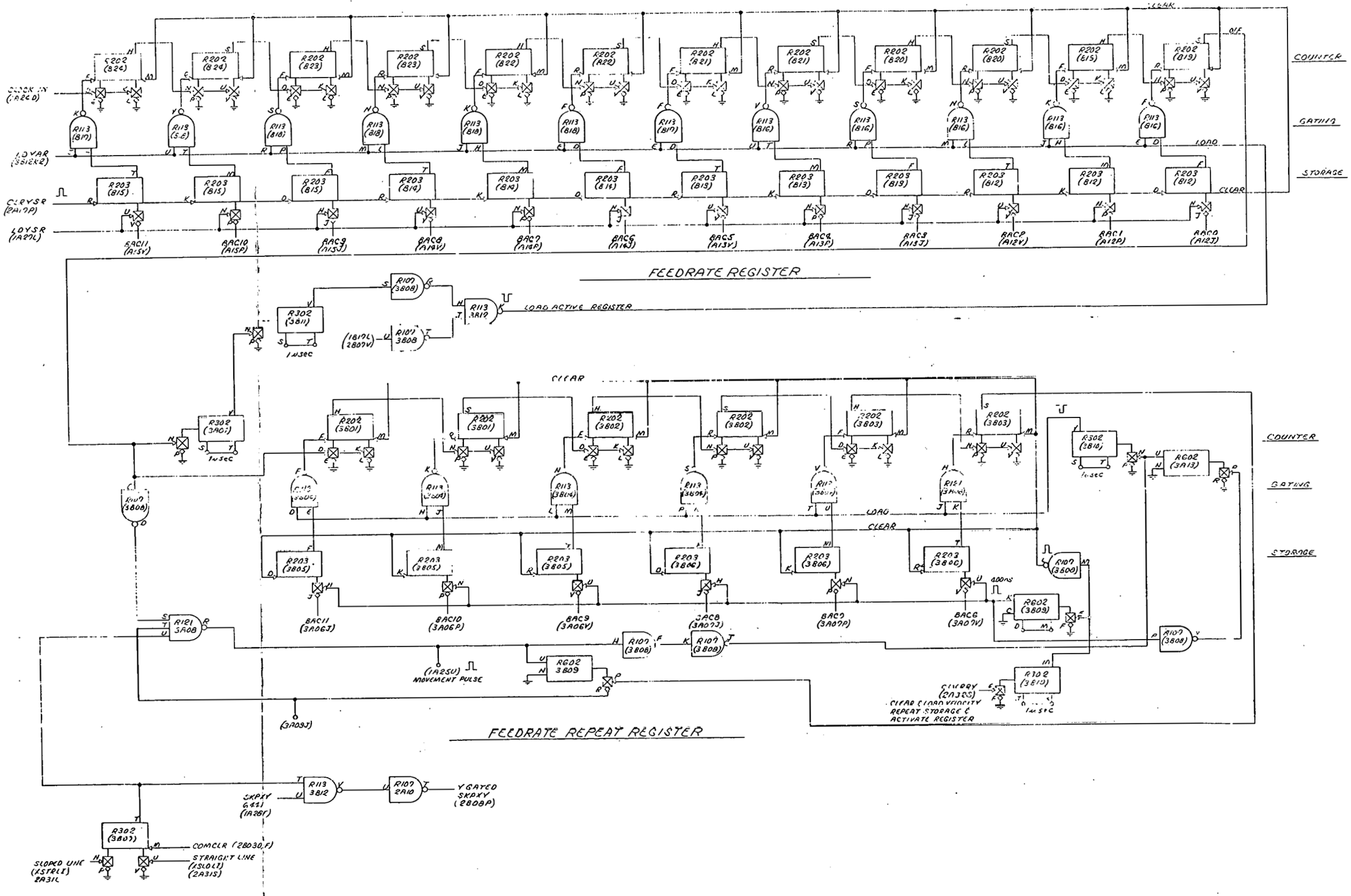
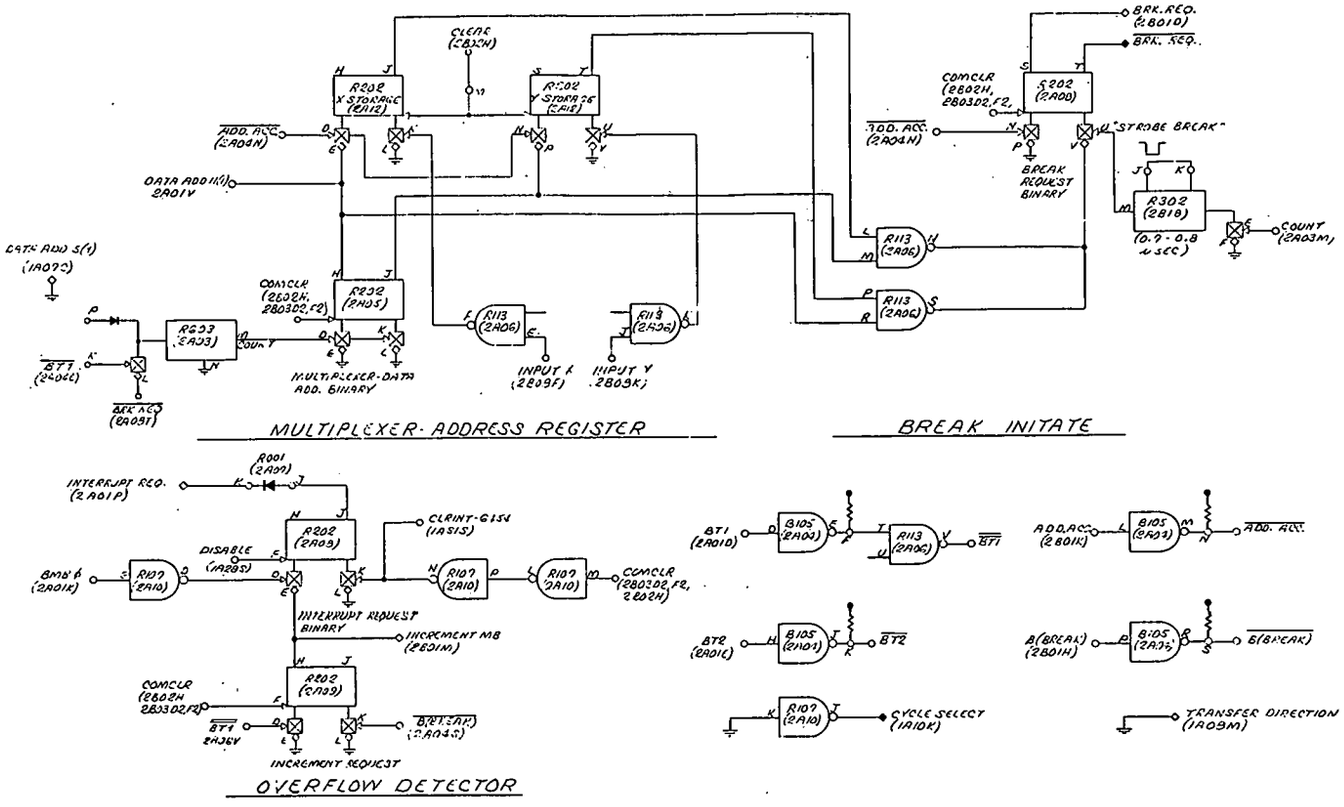
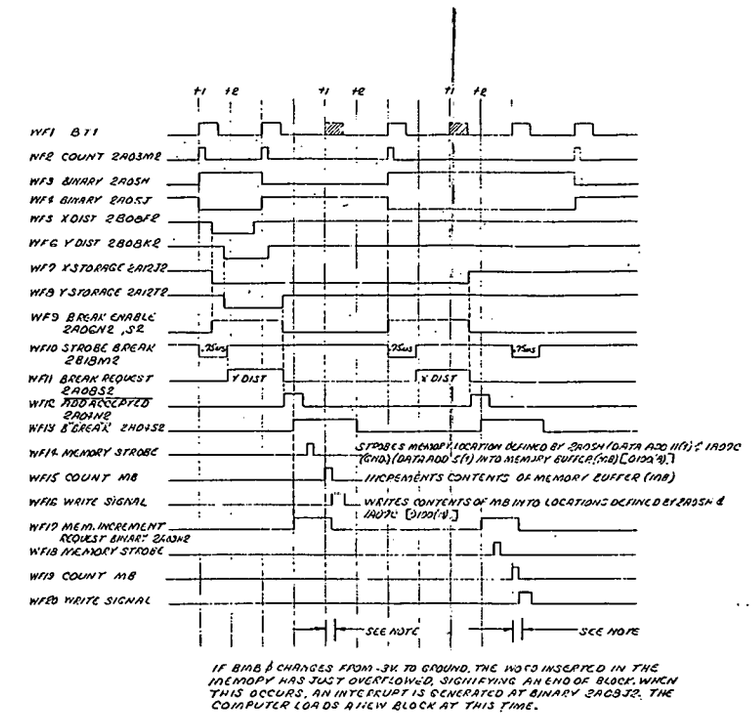


Figure A-3. MACHINE TOOL CONTROL. (Work Slide Pulse Generator)





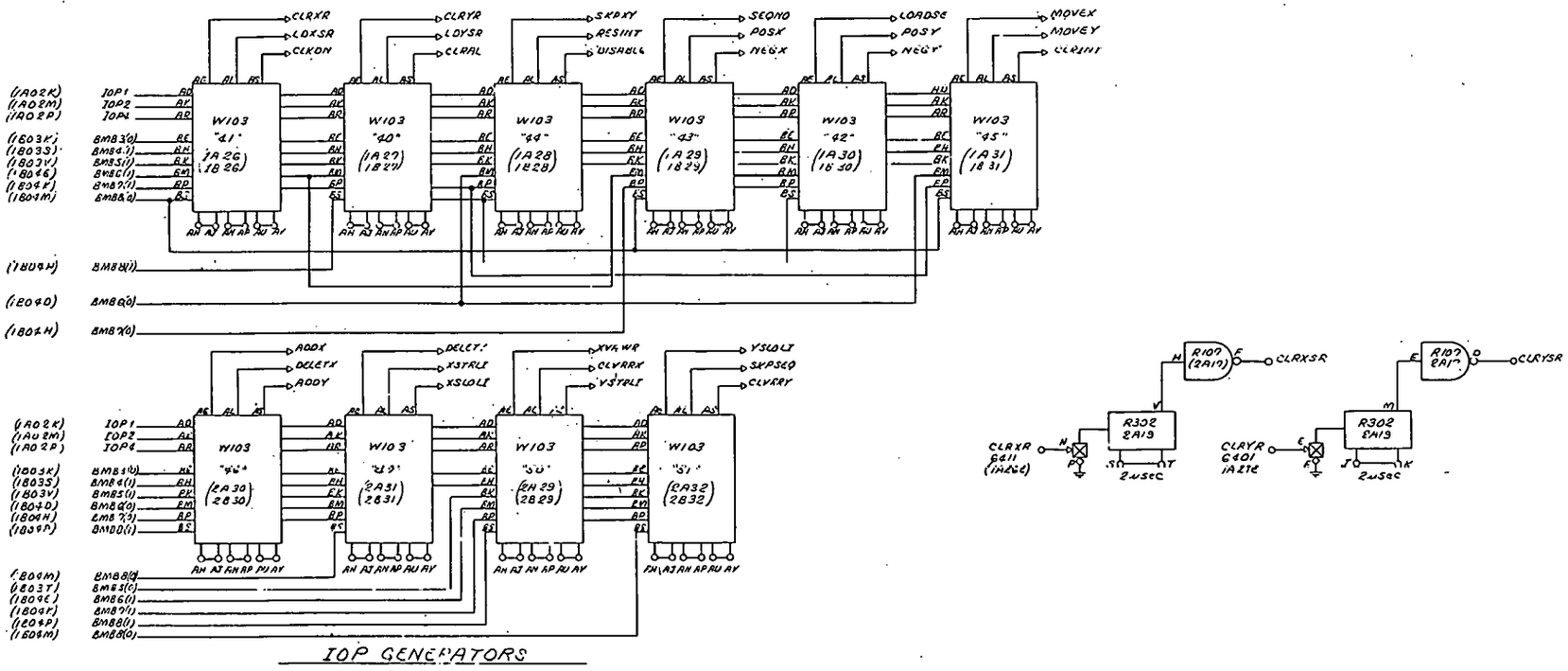


Figure A-4. MACHINE TOOL CONTROL. (Miscellaneous Logic)

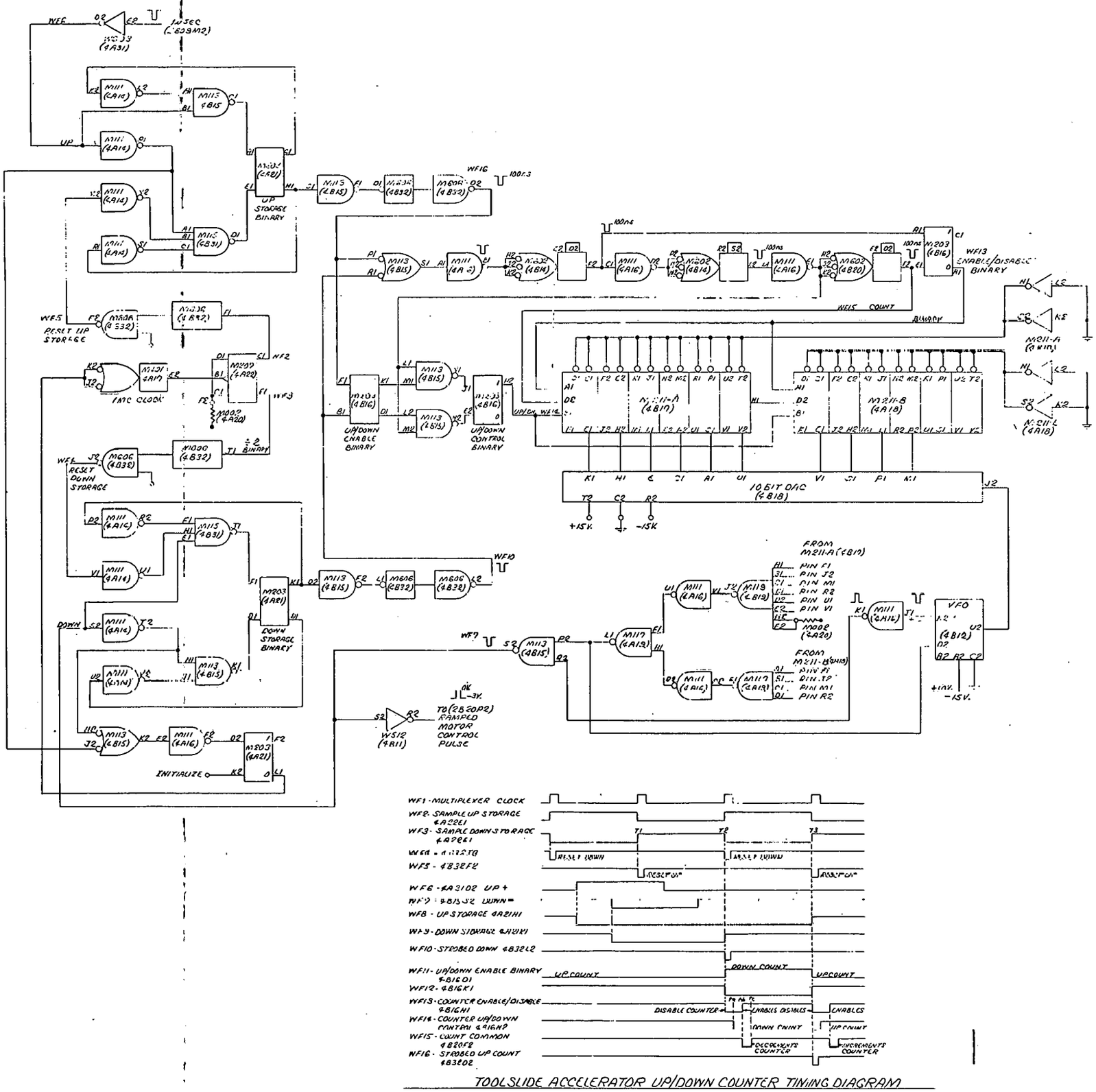


Figure A-5. MACHINE TOOL CONTROL. (Tool Slide Accelerator)

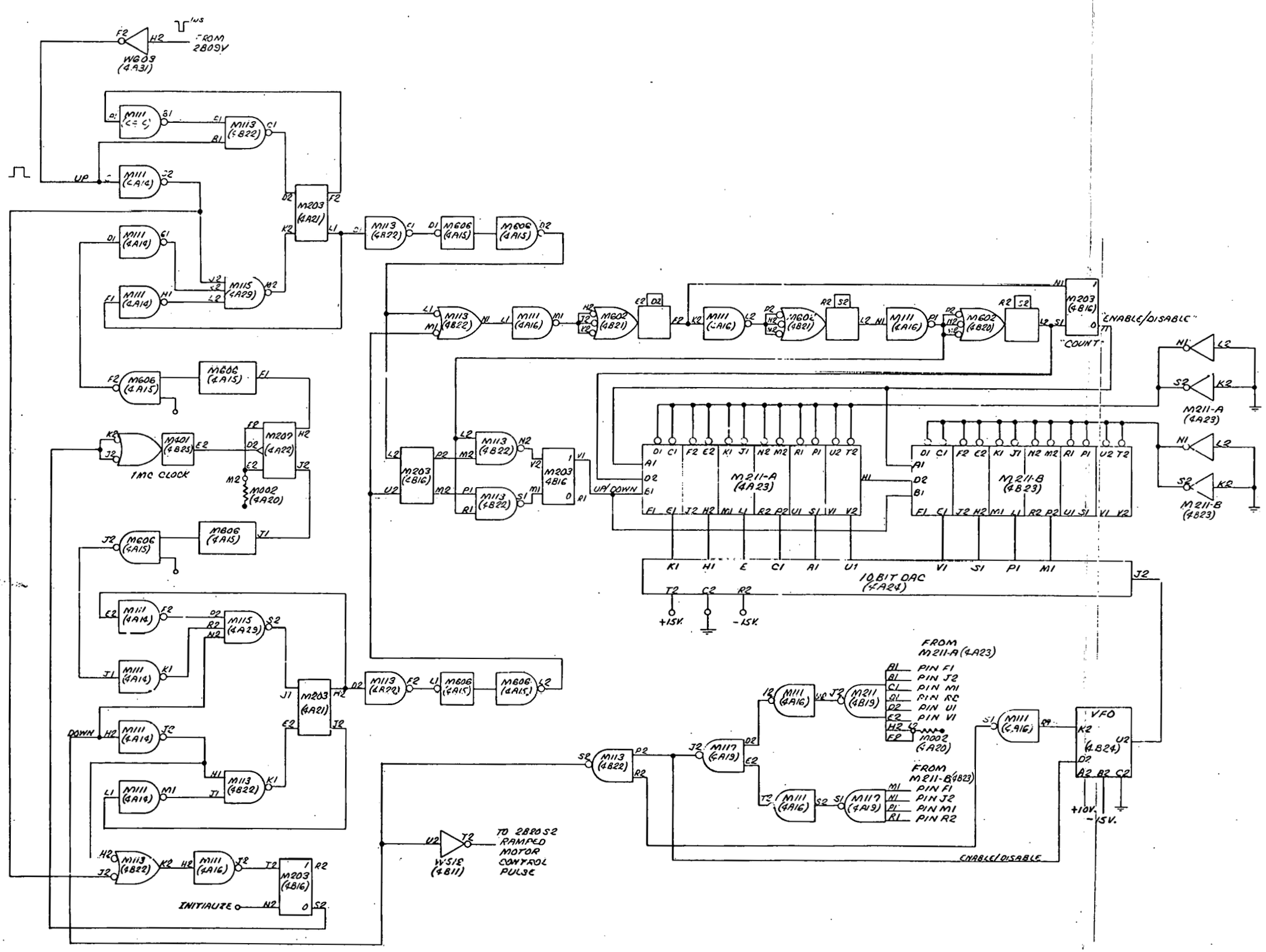


Figure A-6. MACHINE TOOL CONTROL (Work Slide Accelerator)

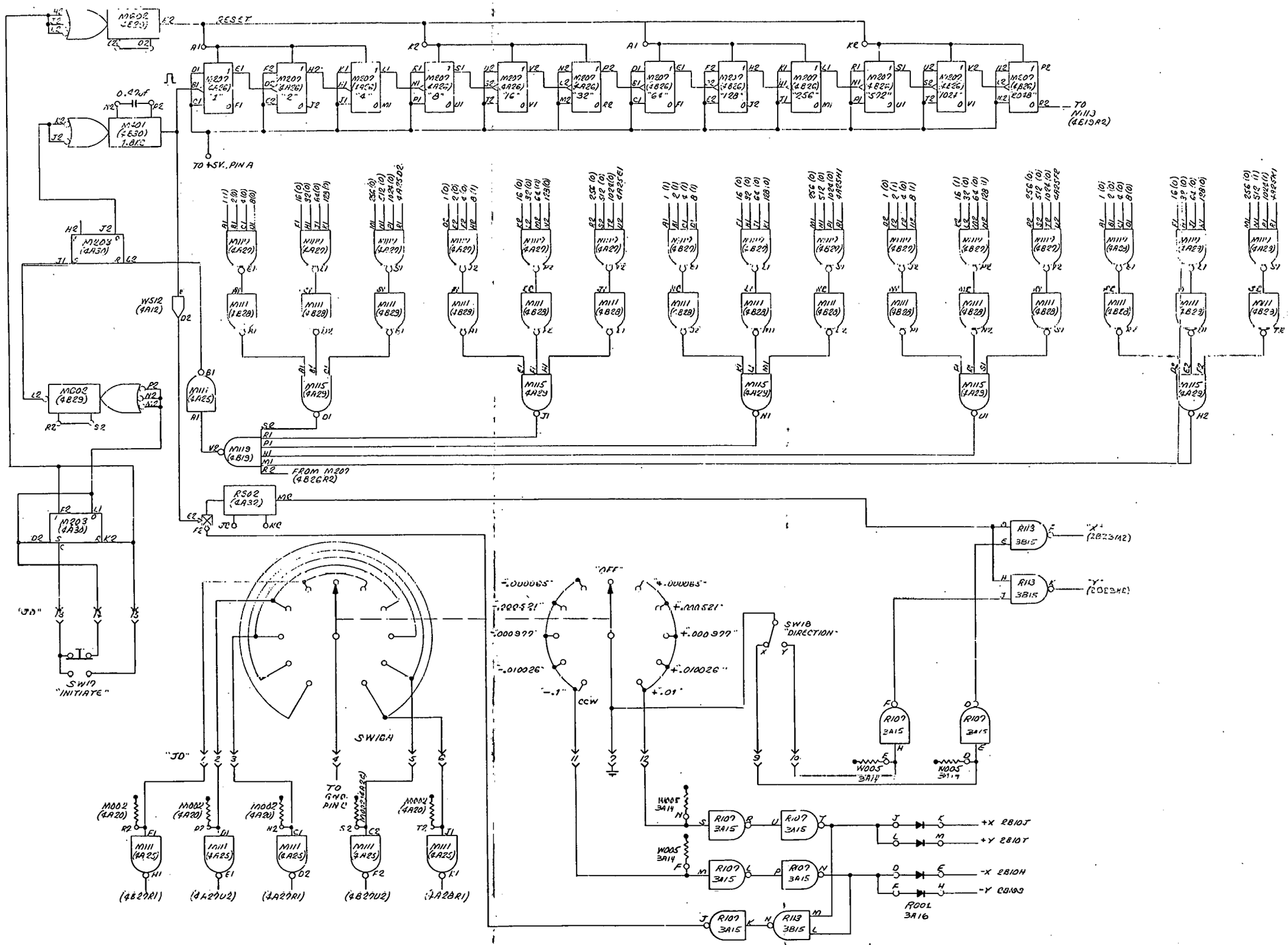


Figure A-7. MACHINE TOOL CONTROL. (Pi Offset)

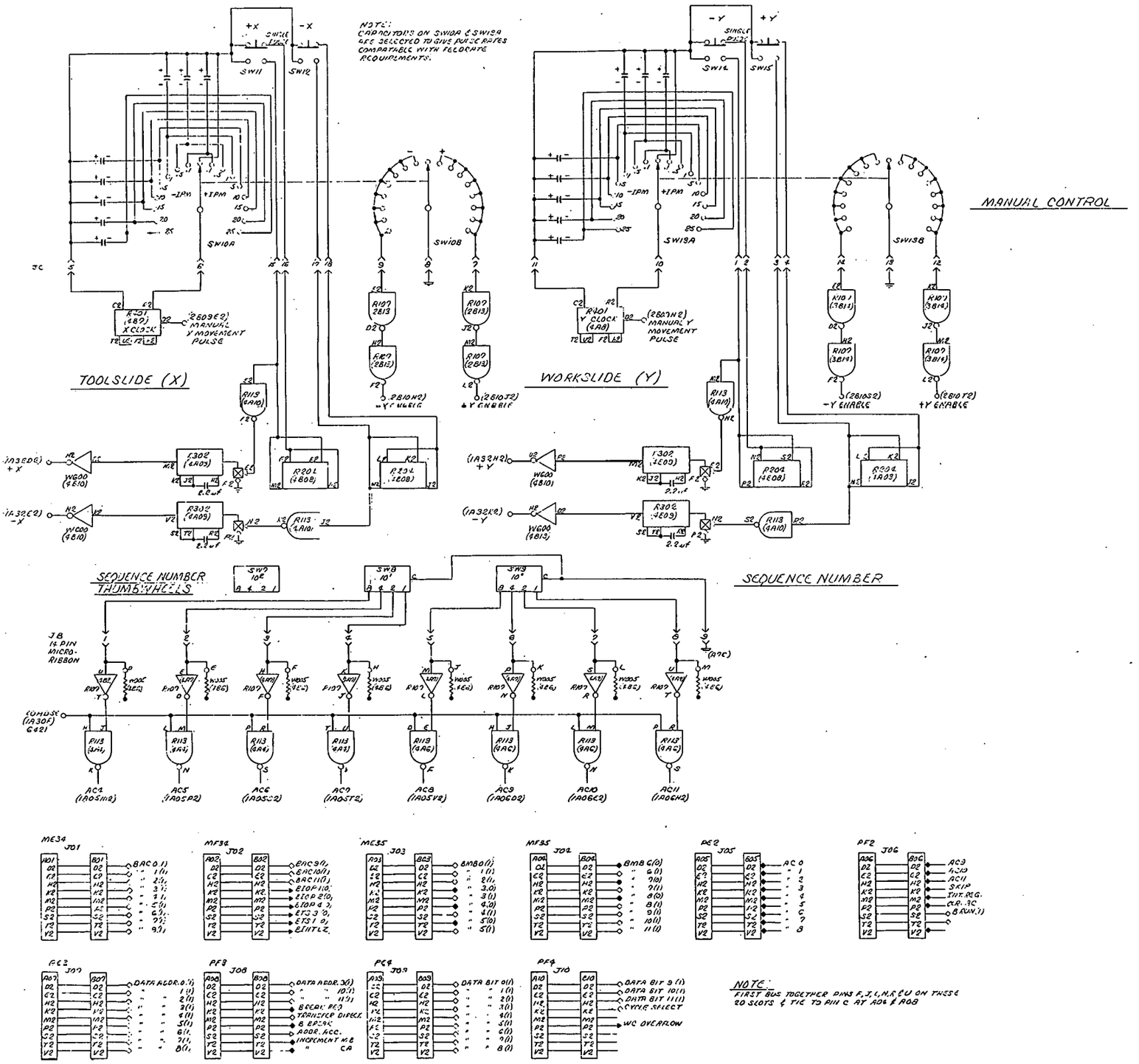


Figure A-8. MACHINE TOOL CONTROL. (Computer Input-Output Bus)

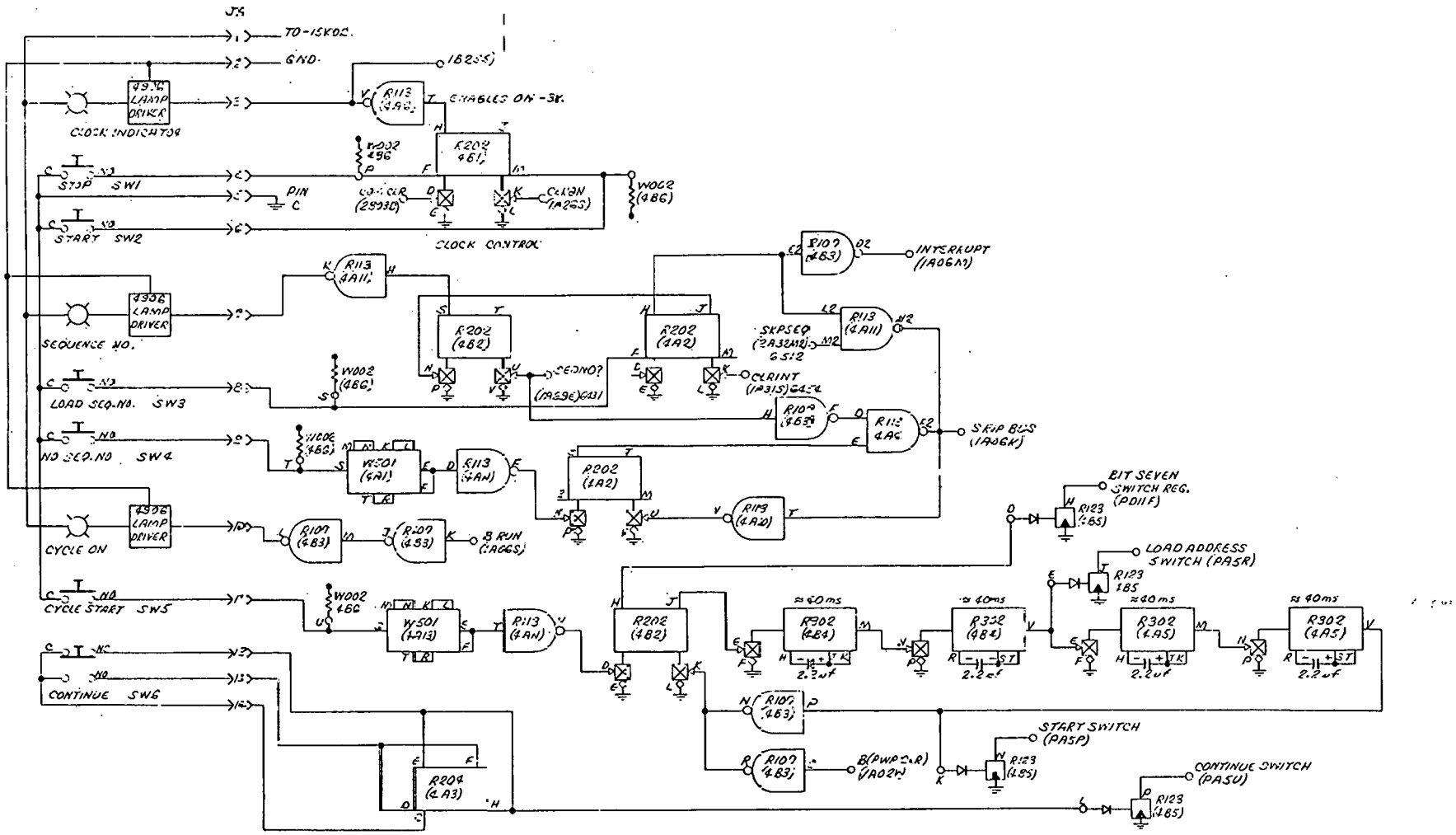


Figure A-9. MACHINE TOOL CONTROL. (Computer Control and Cables)

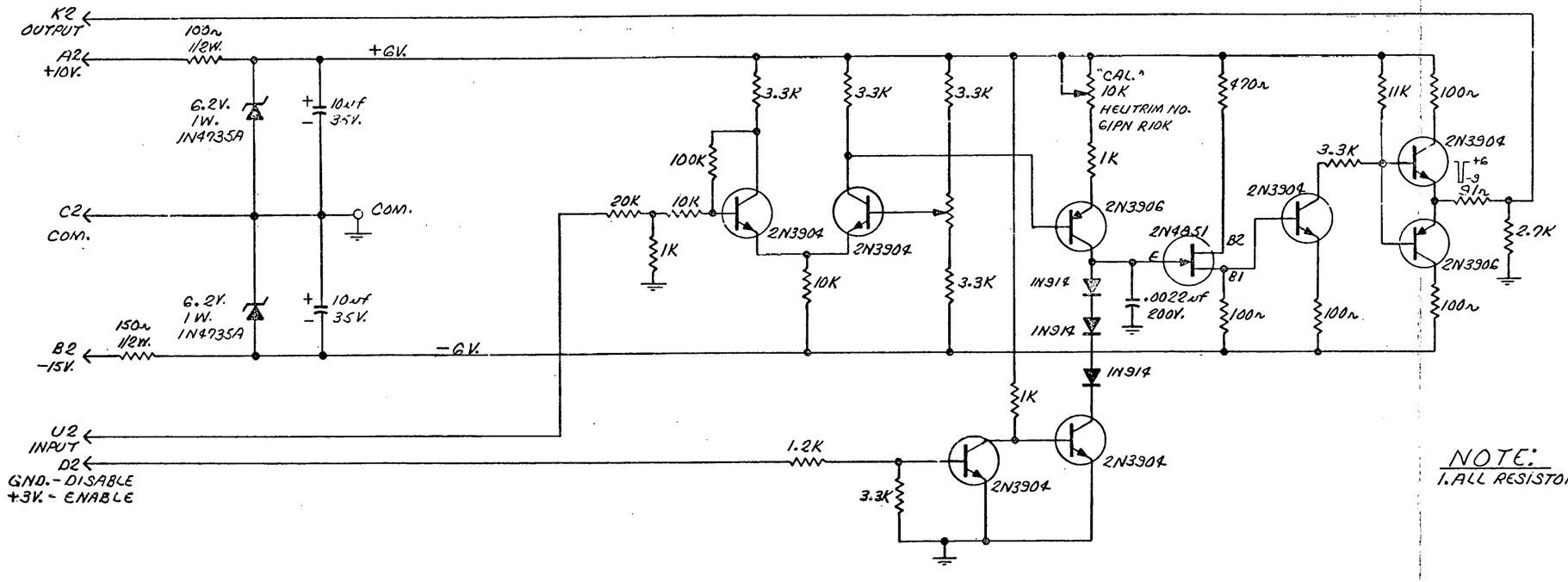


Figure A-10. MACHINE TOOL CONTROL. (MTC VFO Schematic)

NOTE:
1. ALL RESISTORS ARE 1/4W., 5% UNLESS OTHERWISE NOTED.

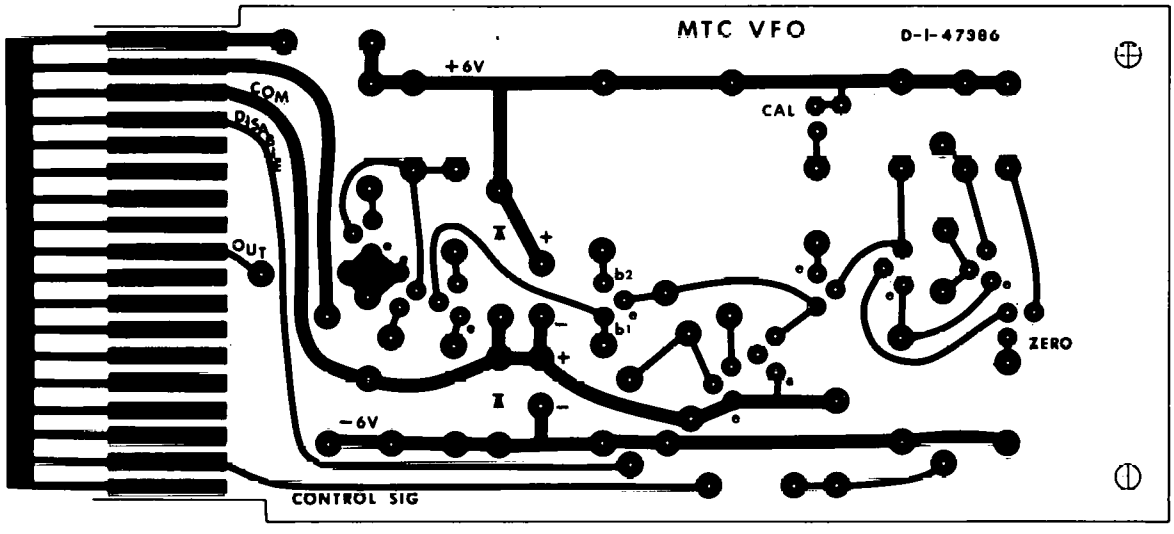


Figure A-11. MACHINE TOOL CONTROL. (MTC VFO PC Layout)

Row	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
A	J01	J02	J03	J04	J05	J06	J07	J08	J09	J10	RACK JUMPER	R203	R203	R203	R203	R113
B	J01	J02	J03	J04	J05	J06	J07	J08	J09	J10	RACK JUMPER	R203	R203	R203	R205	R113

Row	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
A	R113	R113	R202	R202	R202	R202	R202	R202	R203	W103	W103	W103	W103	W103	W103	RACK JUMPER
B	R113	R113	R202	R202	R202	R202	R202	R202	K401	W103	W103	W103	W103	W103	W103	RACK JUMPER

RACK NO. 1

Row	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
A	RACK JUMPER	RACK JUMPER	R603	B105	R202	K113	R001	R202	R202	R107	R107	R207	SP.	SP	SP.	R302
B	RACK JUMPER	RACK JUMPER	R107	R107	R113	K302	R302	R113	R302	R202	W600	W920	W600	R401	R902	W501

Row	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
A	R117	W005	R302	R902	R203	R107	R203	R113	R603	R902	SP.	SP.	W103	W103	W103	W103
B	W720	R302	W005	R107	R001	R113	R107	SP.	SP.	SP.	SP.	SP.	W103	W103	W103	W103

RACK NO. 2

Row	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
A	R302	R202	R202	R202	R113	R203	R203	R121	R207	R107	R602	R302	R602	W005	R107	R001
B	R202	R202	R202	R113	R203	R203	R202	R107	R602	R302	R302	R113	R107	R107	R113	R302

Row	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
A	SP.	"	"	"	"	"	"	"	"	"	"	"	"	"	"	"
B	SP.	"	"	"	"	"	"	"	"	"	"	"	"	"	"	"

RACK NO. 3

+15V. REG. -15V. REG.	+10V. REG. -15V. REG.

POWER SUPPLY RACK (MTD. BEHIND RACK NO.3)

Row	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
A	W501	R202	R204	R113	R302	R113	R107	R401	R302	R113	R113	W512	W501	M111	M606	M111
B	R202	R202	R107	R302	R123	W005	R401	R204	R302	W600	W512	VFO	W600	M602	M113	M203

Row	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
A	M401	M211	M117	M002	M203	M207	M211	DAC 12 BIT	M111	M207	M117	M117	M115	M203	W603	R302
B	M211	DAC	M119	M602	M602	M113	M211	VFO	M401	M207	M117	M111	M602	M401	M115	M606

RACK NO. 4

+5V. REG.	+10V. REG. -15V. REG.

POWER SUPPLY RACK (MTD. BEHIND RACK NO. 4)

Figure A-12. MACHINE TOOL CONTROL. (MTC Module Assignment)

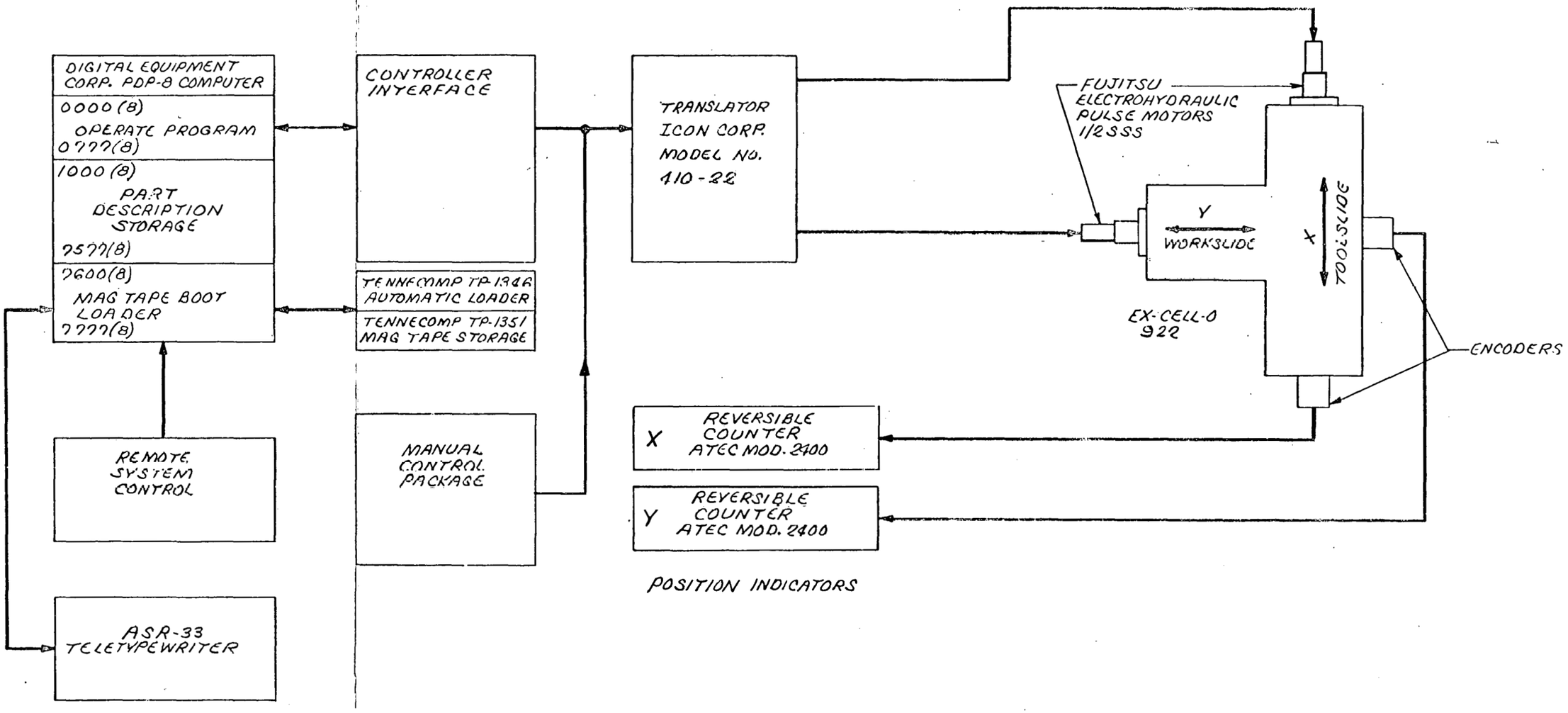
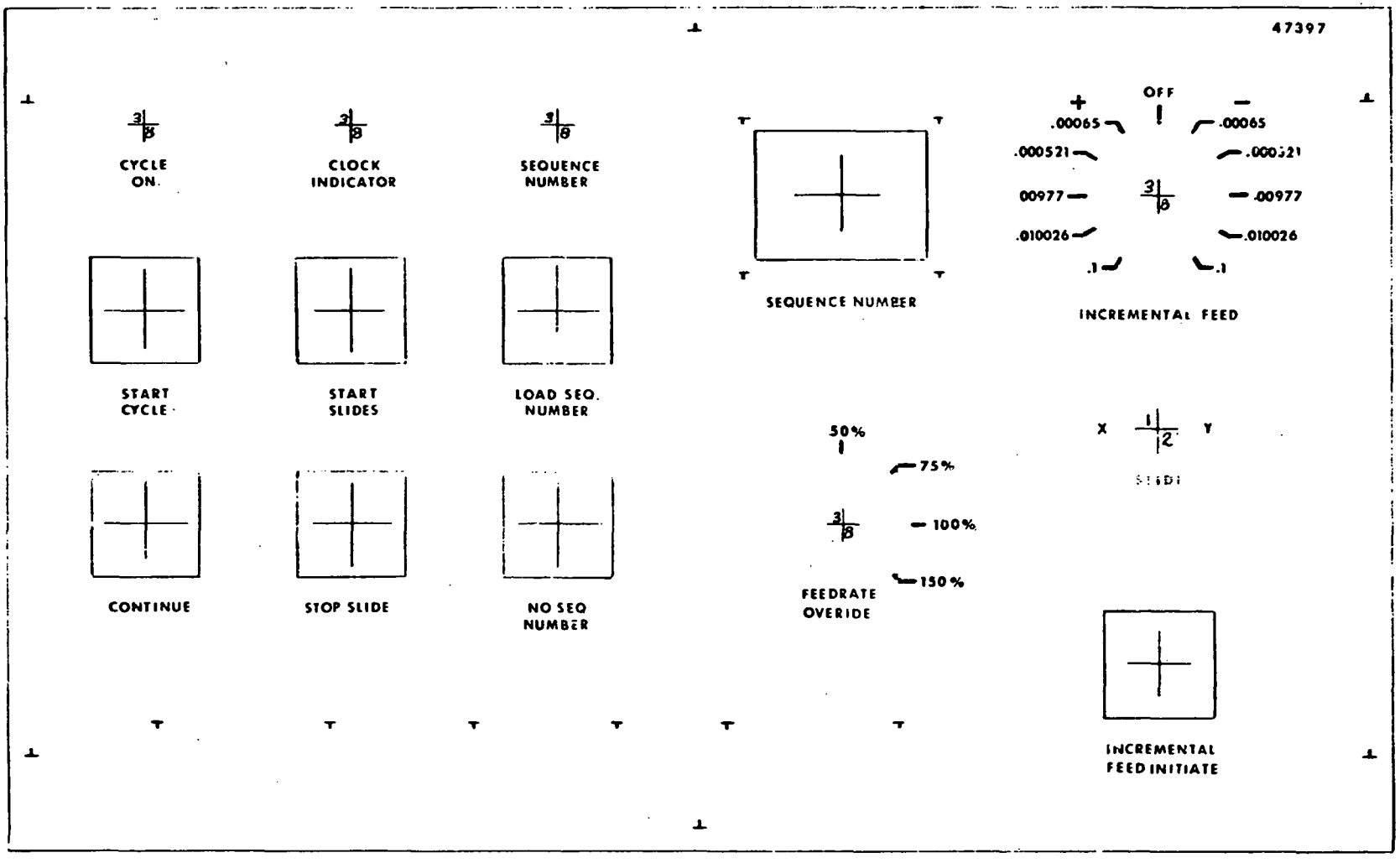


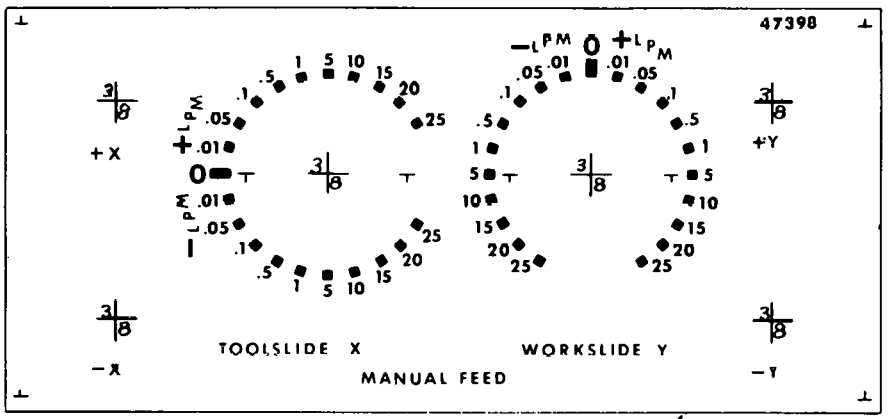
Figure A-13. MACHINE TOOL CONTROL. (MTC System Block Diagram)



T NO. 30 (.1285) DRILL, 10 HOLES

⊥ NO. 16 (.177) DRILL, 6 HOLES

Figure A-14. MACHINE TOOL CONTROL. (MTC Control Panel)



⊤ NO. 30 (.1285) DRILL, 4 HOLES
⊥ NO. 25 (.1495) DRILL, 4 HOLES

1/8 STOCK

Figure A-15. MACHINE TOOL CONTROL. (MTC Manual Feed Panel)

APPENDIX B

COMPUTER SOFTWARE

Machine Tool Operate Program

This program is initiated by depressing the "cycle start" switch on the control panel. Electronics in the interface load 0200(8) in the computer's switch register and starts the program flow at this point. Computer IOT commands (refer to Table B-1 for all IOT functions) CLRINT and CLRAL clear the electronics in the interface, and the interrupt bus is enabled to allow input of a sequence number if necessary. Program flow jumps to Subroutine SEQ at 0020(8) where IOT SEQNO? and JMP-1 make the sequence number inquiry by lighting the "seq no" lamp on the control panel. If no number is desired, "no sequence number" is depressed at the control panel; and, on the next SEQNO?, a skip signal is generated which causes the program to miss the JMP-1 command. The accumulator is set to 0777(8) (memory location prior to start of the part description program) and program flow jumps back to 0205(8) where the interrupt bus and translator controller are disabled. ML 0777(8) is then loaded in ML 0017(8). If a sequence number is needed, the desired number is loaded in the three decade switches and "load sequence" is depressed. This operation causes a program interrupt, forcing program flow to Location 0000(8). With the present system it is possible to have two different interrupts (overflow from distance counter and loading of sequence number). The routine at this point determines the interrupt originator by first issuing SKPXY IOT. An interrupt from a distance counter will cause the program flow to go to Location 0122(8), otherwise a SKPSEQ IOT is issued. Since the sequence number request was the interrupt originator, program flow moves to subroutine SEQUEN at ML 0025(8) where the BCD sequence number is loaded into the computer's accumulator by LOADSE IOT. If this number happens to be zero, the accumulator is loaded with 777(8) and stored in 0017(8). Any number other than zero is converted from binary coded decimal to binary by the subroutine located between 0030(8) and 0053(8). This action is accomplished by adding the binary number for 10 [0012(8)] to itself and using the negated 10^1 decade as the counter which defines the number of additions. After this step is completed, 10^0 is added to the result, giving a binary number equal to the BCD sequence number. Use of binary sequence numbers was necessitated because of storage efficiency. The subroutine located at Memory Locations 0054(8) through 0071(8) searches the part description to determine where this number is stored. Each block of part description data has eight 12-bit words with Bits 0 through 5 of Word 8 set aside for storage of a sequence number. This subroutine searches the complete part description; and, upon finding the sequence number in question, sets the accumulator equal to the memory location where the number is stored. When part description data are retrieved, the following memory location is the beginning of the next data block.

After the starting point of data retrieval is set in the accumulator, program flow moves to 0205(8) where the interrupt bus is turned off, pulse generator circuits are disabled through IOT DISABL, and the accumulator is stored in ML 0017(8) (defines the starting location of the part description). Hardware velocity counters are cleared through IOTs CLRXR and CLRYR and software Distance Registers 100(8) and 101(8) are set equal to zero. The program jumps to Location 730(8) where the first block of information is retrieved from

Table B-1

COMPUTER INPUT-OUTPUT TRANSFERS

Phenomic	OCTAL Coded Word	Function
CLRXR	6411	Clears Toolslide Feedrate Register.
LDXSR	6412	Loads Toolslide Feedrate Register from the accumulator.
CLKON	6414	Enables Feedrate Clock.
CLRYR	6401	Clears Workslide Feedrate Register.
LDYSR	6402	Loads Workslide Feedrate Register from the accumulator.
CLRAL	6404	Disables or clears all interface electronics.
SKPXY	6441	Inserts extra pulse at end of data block and is used to interrogate interrupt electronics for end-of-block interrupt.
RESINT	6442	Not used.
DISABL	6444	Disables Interrupt Request Binary.
SEQNO?	6431	Used to turn on "sequence request" light and to generate a skip command if no sequence number is desired.
POSX	6432	Sets toolslide Sign Control Binary for positive axis movement.
NEGX	6434	Sets workslide Sign Control Binary for negative axis movement.
LOADSE	6421	Loads accumulator with contents of BCD sequence number switches.
POSY	6422	Sets workslide Sign Control Binary for positive axis movement.
NEGY	6424	Sets workslide Sign Control Binary for negative axis movement.
MOEVX	6451	Causes a movement of one pulse on the toolslide axis.
MOVEY	6452	Causes a movement of one pulse on the workslide axis.
CLRINT	6464	Resets Interrupt Request Binary.
ADDX	6461	Adds one pulse to the toolslide axis.
DELETX	6462	Deletes one pulse from the toolslide axis.
ADDY	6464	Adds one pulse to the workslide axis.
DELETY	6471	Deletes one pulse from the workslide axis.
XSTRLI	6472	Disables the Workslide Pulse Generator.
XSLOLI	6474	Enables the Workslide Feedrate Generator.
XVRWR	6501	Enables both Feedrate Generators.
CLVRRX	6502	Loads the Toolslide Feedrate Repeat Register with the contents of the accumulator.
YSTRLI	6504	Disables the Toolslide Pulse Generator.
YSLOLI	6511	Enables the Toolslide Pulse Generator.
SKPSEQ	6512	Used to interrogate the interrupt electronics for an interrupt generated by depressing the Load Sequence Switch.
CLVRRY	6514	Loads the Workslide Feedrate Repeat Register with the contents of the accumulator.

the part description. This subroutine immediately moves the program flow to GETDAT [522(8)] which is used for data removal during the successive program operation. Auto-index register 0017(8) is used to retrieve the first block of data, storing the first seven words in temporary storage at ML 105(8) through 0113(8). Program flow now moves to 362(8) where the sequence number is stripped from the last word of the block and the result is stored in 0114(8). Velocity repeats are stripped from 111(8) and stored in 322(8) for later use. Each time a data block is inserted in temporary storage, checks are made at ML

545(8) and 546(8) to determine if this is the last block of part description. If this happens to be the case, a subroutine at 154(8) changes the interrupt routine so that program flow goes to 160(8) instead of 122(8) at the completion of the data block being processed. When this change occurs, both feedrate registers are disabled by IOT CLRAL and the interrupt routine is restored to its original state. If the check at 546(8) reveals that this is not the end of the program, subroutine GETDIR [747(8)] is entered where the sign bit of each distance is used to set ML 360(8)-XDIR and 361(8)-YDIR to IOT commands that are used to set sign control FFs when this block of part description is being loaded into the interpolator. After the sign bit is removed, both numbers defining distance are complemented and restored in MLs 105(8) and 106(8) by subroutine COMDIS located at 142(8). MISCRE 111(8) (Table C-1) contains a bit that is set if a programmed halt is necessary. Subroutine TEMHAL [561(8)] strips this bit and modifies subroutine SKIP [122(8)] so that a halt occurs after the present block is processed. After the velocity remainder is stored in 77(8), program flow jumps back to the memory location after the point where GETDAT was called [217(8)]. Here, the interrupt electronics in the interface is cleared by IOT CLRINT, and both velocity storage registers are loaded from MLs 107(8) and 110(8). This part of the operate program [0200(8) through 225(8)] is only used when machine operation begins. Velocity registers are loaded by SKIP [122(8)] during the remainder of the part description program. After this operation is accomplished, program control jumps to 227(8).

At 227(8), the software distance counters are checked to determine if they contain zero. Since this is the first block of data, this will be the case, but later checks are very important. From the design concept it is seen that these two registers must both contain zero if the proper number of pulses were outputted on each axis. Any number other than zero in these registers indicates that something is amiss and program control jumps to EMERST [120(8)] where machine operation is halted. When both registers contain zero, IOT sign commands stored in 360(8) and 361(8) are used to set sign control FFs, and software distance registers are loaded with the contents of 107(8) and 110(8).

Subroutine LINE [400(8)] is entered and outputs IOTs that disable the axis not in use when a straight line move is to be made. This step is necessary because, at the completion of the line, a pulse would automatically be inserted on the axis not in use by IOT SKPXY, MISCRE is interrogated at 417(8), and constants in VELERR [272(8)] are set to compensate for velocity errors by monitoring Bits 0 and 1 (Table C-1). After MGETDI [717(8)] sets ML 355(8) equal to the initial distance of the axis defined by Bit 1, Bit 2 of MISCRE is removed and used to set SINDIS [711(8)] for later use when distance errors are being corrected by subroutine [634(8)]. If there is a velocity repeat stored in 114(8), the number is now loaded into velocity repeat registers by IOTs CLVRRY and CLVRRX. These registers are then enabled by IOT XVRWR at subroutine GETVEP [337(8)].

Subroutine SETDUP [600(8)] monitors DIST [716(8)] for determining the axis on which there will be a distance error. DIST was set during subroutine LINE. By observing this action, along with SINDIS [711(8)], EXPPU inside subroutine DISSUM is modified to correct for any distance errors that may occur during a block repeat.

All hardware and software are now ready to start processing the data block stored in 105(8) - 114(8). Interpolation is initiated by enabling the variable clock with IOT CLKON at ML 252(8). If there are block repeats, REPEAT [351(8)] is incremented, and subroutine

DISSUM [634(8)] is entered where DSTMDR [ML 113(8)] (that defines the distance error accumulated for each block repeat) is added to itself. If an overflow occurs (signifying an error of one pulse), EXTPU will either add or delete one pulse from the axis in error. Subroutine VELERR is now entered for correcting the velocity errors. The last block repeat is made with REPEAT equal to zero, but with FLAG [357(8)] equal to 7777. This situation again forces a program flow to DISSUM for the final distance correction; but, before VELERR is entered, subroutine GETDAT [522(8)] stores the next block of part description data in 105(8) through 114(8) for immediate use after the last block repeat. If no block repeats are seen at 0253(8), program flow jumps to 262(8) where a check of FLAG reveals that it is zero. This action forces the program to miss DISSUM, ACCERR 710(8) is set to 4000(8), and the next block of part description data is loaded in 105(8) through 114(8) by subroutine GETDAT at 522(8). Analysis of GETDAT was covered in earlier parts of this section. After the new block of part description data is stored, VELERR [272(8)] is entered for appropriate error corrections.

As mentioned earlier, during the generation of a sloped line, an error exists on one axis each time a movement pulse is generated. This error is accumulative; and, if no corrections were made, would generally cause the control system to halt after the block of data is processed. Subroutine VELERR [272(8)] monitors the axis in error and outputs a correction signal each time the error accumulates to 65.104166.. microinches. Accumulated error is first set to 4000(8). This step is taken to force a correction signal after one-half pulse error is present, keeping the tool paths within one-half pulse (32.552083.. microinches) at all times. Next, a check is made to determine if the tool path is a straight line (either X or Y distance equal to zero). If such is the case, no error correction is necessary, and the program flow jumps to 333(8) where the next toolslide velocity is loaded into the computer's accumulator and a loop is entered where the computer waits for a program interrupt, signifying the end of the present data block. It is possible to have accumulated more than one distance pulse after the straight-line check is made. This condition is a result of the time necessary to perform the program functions after the clock was enabled at 252(8). These pulses are monitored by comparing the original contents of the distance register in question (stored in MLASDI) with the present contents and storing the complement of the result in COUNT [354(8)]. STRMDR (velocity error for one pulse) is added to ACRMDR [352(8)] each time COUNT is incremented until it reaches zero. If an overflow occurs into the link during the addition to ACRMDR, a correction pulse is outputted at EXTPUL [317(8)] which either adds or deletes a pulse from the axis being monitored. When COUNT reaches zero, MLASDI is updated to equal the contents of the present distance register defined by DISTAN [353(8)]. MLASDI is continuously compared to this distance register; and, as each pulse arrives, STRMDR is added to ACRMDR. Each time an overflow occurs, a correction is made to the appropriate axis. When this register gets within one pulse of its destination, no more axis corrections are made. At this time the next tool slide velocity is inserted into the accumulator and the machine waits for a PI which occurs when either axis reaches its destination.

An interrupt from either axis forces program flow to memory location 0000 where IOT SKPXY inserts a pulse on either axis if a distance pulse is not present and forces program flow to ML 122(8). Before the interrupt occurred, the toolslide velocity (XCLOCK) stored in 105(8) was inserted into the accumulator. IOT CLXR at 122(8) loads this number into the toolslide velocity storage register. The workslide velocity [YCLOCK ml 106(8)] is next

loaded in the workslide velocity storage registers by IOT CLYR, and the interrupt electronics is cleared by IOT CLRINT, allowing the system to process another interrupt from software distance register overflows. The velocity error RMD stored in 112(8) is transferred to STRMDR [77(8)]; and, if there is no programmed halt, program flow moves to 227(8) to repeat the previously discussed operations. If the present block has a 1 in the Bit 3 of MISCRE, ML 132 will contain a CLRAL statement instead of the JMP 1 CHK 2 that forced a jump to ML 227(8). This IOT turns off all control electronics and the program flow halts at ML 133(8). When it is desired to continue operation, the operator depresses the "continue" switch. ML 132(8) is reset to JMP 1 CHK2 and the program flow moves to ML 227(8).

The operations just described are repeated for each data block. The final part description data block contains all zeros. When sensed during the GETDAT subroutine, ML 0003(8) is modified to cause the program flow to jump to ML 0160(8) after the previous block of data is processed. Here, CLRAL IOT disables all hardware in the interface, ML 0003(8) is returned to its original state (JMP SKIP), and the program flow halts. At this time the machining operation stored in memory can be repeated, or a new operation may be loaded into memory from the magnetic tape unit.

Post Processor

A part description program for use on a minicomputer was devised to calculate and output the machine tool contouring information. Specifically, this software reduces the part contour to straight-line segments while maintaining a specified contour accuracy. For a selected machine feedrate, the program punches a paper tape containing the toolslide and workslide incremental distances and velocities for cutting a part contour on the computer-controlled two-axis lathe. In addition, tool-path data may be entered for reduction by means of teletypewriter or a high-speed tape reader. The data must follow a prescribed format in each instance. In the case of the teletypewriter input, only straight lines and arcs of constant radius may be described for conversion. For a straight line, the information format is shown in Figure B-1.

I	X_I	Y_I			
L	X_1	Y_1	SFM	PITCH	First Line

I	Initialize identification symbol.
L	Line identification symbol.
X_I	Initial value of X.
Y_I	Initial value of Y.
X_1	First value for X.
Y_1	First value for Y.
SFM	Surface velocity in inches per minute.
PITCH	Cutting pitch in inches per revolution.

Note: Underlined information is typed in response to inquiries from the computer.

Figure B-1. TELETYPEWRITER INPUT TO THE POST PROCESSOR FOR A STRAIGHT-LINE PATH.

Arcs are generated by entering the data outlined in Figure B-2. When it is necessary to use standard EIA NC tapes (Specification RS-274B) as the input medium, the arc-generation software is replaced by subroutines that strip all necessary information from the tapes. The input format is summarized in Figure B-3. After this information is accepted, the tape is read, and a part description is generated from the X and Y coordinates that are punched on the NC tape.

I	X_I	Y_I		
A	+R	θ	SFM	PITCH

I	Initialize identification symbol.
A	Arc identification symbol.
R	Radius of arc with sign noting direction.
θ	Angle subtended by arc.
SFM	Surface velocity in inches per minute.
PITCH	Cutting pitch in inches per revolution.

Figure B-2. TELETYPEWRITER INPUT TO THE POST PROCESSOR FOR AN ARC PATH.

I	X_I	Y_I	
A	SFM	PITCH	

I	Initialize identification symbol.
A	Stripper identification symbol.
SFM	Surface velocity in inches per minute.
PITCH	Cutting pitch in inches per revolution.

Figure B-3. POST PROCESSOR INPUT MODIFICATIONS FOR STANDARD ELECTRONICS INDUSTRIAL ASSOCIATION (EIA) DATA.

For brevity, only the teletypewriter input of a straight line and its conversion to usable data is discussed. Figure B-4 illustrates the coordinate system used and shows the desired line segment. Location 0 is the tool touch-up point which is used as a reference for the initial calculations. Referring to the program flow chart, the program is started in Octal Location 1000. After flags are set and storage registers are cleared, a leader is punched on the output tape. A number is then punched that is used as the starting location for data storage in the operator program. The teletypewriter input routine is entered, and data are entered in the specified sequence. For the straight line of Figure B-4, an "I" is typed to signify that the initial coordinates are to follow. Values for X_1 and Y_1 are entered; and, after a carriage return, an "L" is typed to identify the line calculation subroutine. For the following discussion, assume the line to X_1 and Y_1 is already processed, and an "L" is typed to identify the next line. In the line routine, the LINST flag is set to "1" to signify that the routine has been entered. The operator then types in the values for X2, Y2, the desired machining surface velocity, and pitch. Now the program begins its calculations.

First, the coordinates are adjusted for machine lead screw error, X2 and Y2 are designated as goals, and ΔX and ΔY are calculated in the LEFCHK routine. The first calculations for a contour require no information on previous conditions. If ΔX is zero, FLAGGY is set equal to 4000; if ΔY is zero, FLAGGY is set equal to 1000. The ratio of ΔX to ΔY is the SLOPE.

For line segments inside a one-inch radius, the resultant surface velocity is set equal to the surface velocity for a one-inch radius. If part of the line lies within one inch of the axis of rotation and machining is toward this axis, the program first does calculations for that part of the line lying outside the one-inch radius. On the other hand, if machining away from the axis of rotation, the program first does calculations for the portion of the line inside the one-inch radius. Otherwise, velocity calculations are made with the stipulation that the velocity at the midpoint of the segment differs from the velocity at either endpoint by no more than five percent. If calculations indicate more than a five-percent variation, the line segment to be described by this data block is shortened.

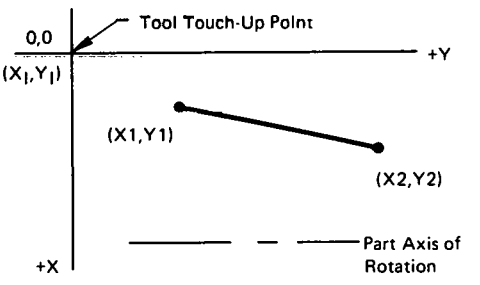


Figure B-4. COORDINATE SYSTEM WITH A TYPICAL LINE.

Both velocities are transformed into clock pulses as follows: The number of X clock pulses (XPULS) occurring at a spacing TIME (period of feedrate clock) required to move the X slide the proper velocity (VELOCX) using a distance per motor pulse INC (65.104166.. microinches) is:

$$XPULS = \frac{INC}{(VELOCX)(TIME)}.$$

If either distance is zero, the corresponding velocity pulses are set to 2047_{10}. XPULS and YPULS are then converted to fixed-point numbers and stored as XCLOCK and YCLOCK.

The total number of X-axis motor pulses needed for this data block is found by dividing $\triangle X$ by the motor-pulse resolution. Y-axis motor pulses are determined by the same method. These results are made positive if necessary, and the original sign is saved. These calculations are made in the DISCAL routine. The longest distance is now divided by the register length (2047_{10}), and the result is placed in a temporary storage register. If this result is greater than one, it is converted to a fixed point number, made positive, reduced by one, and stored in RPTS as the number of repeats. If the number of repeats exceed 511, the program stops; otherwise, RPTS is placed in Bits 4 - 12 of MISC with Bits 0 - 3 clear.

The new coordinate X2, Y2 is found, and the number of motor pulses for the shorter distance is calculated. The number of motor pulses for the long axis is set to 2047_{10} if there were distance repeats, and the fixed-point values for the number of X and Y motor pulses are stored in XDBREG and YDBREG. The shorter axis may have a fractional pulse remainder, which is the difference between the calculated pulses required and the rounded value stored for use in the controller-data-break register. This number is adjusted after each block repeat to insure that the error is always less than or equal to one-half pulse. This adjustment is accomplished through a number (DSTMDR) which is acquired when the remainder is multiplied by 4094_{10} and the result is fixed, made positive if negative, and stored in DSTMDR for use by the operator program. If the remainder was negative, 1000_8 is added to MISC so that corrections will be made in the proper direction.

In preparation for velocity calculations, the velocity-repeat register, VPTS, is cleared, then the polarity of XDBREG and YDBREG is checked. If negative, they are made positive, and Bit 0 is set to "1". If X1 and X2 are less than one inch from the axis of rotation, the resultant velocity is calculated using a one-inch radius; otherwise, the resultant velocity is found by using the mean radius of the line segment. This velocity is made positive if negative and stored as IPS. The X and Y components of velocity are now calculated and stored as VELOCX and VELOCY. These values are checked to determine which one requires the least number of clock pulses. The fixed-point value of clock pulses for this axis is adjusted if necessary to assure that the difference between the fixed-point value and real-floating-point value is less than or equal to one-half clock pulse. This value in conjunction with DX/DY is used to determine the number of clock pulses for the remaining axis. The resulting number is converted to fixed point and also adjusted if necessary to assure that the difference between the fixed-point value and real-floating-point value is less than or equal to one-half pulse. The two fixed-point results are XCLOCK and YCLOCK.

The error on the axis requiring the greater number of clock pulses is made positive if it is negative. When positive, 2000_8 is added to MISC to signify that this axis is leading velocity wise. Now the program converts the velocity error to a fixed-point number that represents the error generated for each velocity pulse. With no velocity repeats, this number is stored as RMDR and is used by the operator program to correct for this type of error.

In the preceding discussion concerning velocity calculations, it was assumed that the largest number of clock pulses required was less than or equal to 3777_8, the maximum usable register length. If this was not the case, both velocity registers would be scaled until the largest one was less than or equal to 3777_8. This step would be accomplished by halving the number of clock pulses for each axis until the condition was satisfied. If the number of clock pulses is scaled, there will be velocity repeats. That is, the countdown of the velocity registers during machining must be repeated to obtain the correct slide velocities. Once the clock pulses have been scaled, the velocity remainder is calculated as described earlier. If the velocity repeats are greater than 77_8, the program stops; otherwise, the velocity remainder is multiplied by the number of velocity repeats, converted to a fixed-point number, and stored as RMDR. The velocity repeats are converted to a fixed-point number, complemented, incremented, and stored as VPTS.

The stored control data [XCLOCK, YCLOCK, XDBREG, YDBREG, the sum (MISC + OPTSTO), RMDR, DSTMDR, and VPTS] are punched on paper tape. This list comprises a complete data block which describes either part or all of the line between X1 and X2. If all of the requested segment could not be described by the first block of data, the coordinates X2, Y2 become the new X1, Y1 with the original destination as X2, Y2. The distance error is summed and stored in the proper register and the calculating cycle is repeated. This procedure continues until the requested line is completely described, at which time the program enters the teletypewriter input routine for further commands.

APPENDIX C

SYSTEM OPERATION

Actual system operation is initiated when power is turned on. At this time the computer automatically outputs a "power clear" signal. This signal initializes the electronics, eliminating spurious signals that would cause slide movements. The operator starts the control system by selecting the proper track of the magnetic tape storage unit (MTSU) and depressing the automatic loader "load" switch which inserts a bootstrap loader program into memory. This program is entered automatically and loads the selected track into the computer's memory. As the program is loaded, a check is made to ensure that the tape is read correctly. If an error occurs, the reading process is repeated. Each magnetic tape track contains an operate program plus a section of part description. The operate program resides in Memory Locations 0000(8) through 0777(8), with the part description in 1000(8) through 7577(8). Each track contains one or more complete operations. In other words, the tool is at a known point at the end of each operation. This condition is necessary because of the inability to change tape tracks during the machining process.

After a track (one or more machining passes, governed by the memory available for part description storage) is loaded, the operator depresses the "cycle start" switch on the manual control panel. This action initiates a computer-controller sequence which electronically sets the proper bits in the computer "switch register", gives an automatic "load address" signal, and starts the program flow at the address defined by the switch register [0200(8)]. The computer program clears the interrupt circuitry through IOT CLRINT along with other electronics to eliminate spurious output signals, enables the computer interrupt bus, and moves to a sequence number inquiry subroutine. Here, a loop is entered that generates IOT "SEQNO?", which illuminates a light on the operator control panel that notifies the operator a sequence number is desired. The loop is exited by a SKIP signal that originates from the computer command SEQNO? after being enabled by depressing the "no sequence number" switch or through a Program Interrupt (PI) which is generated when the operator depresses the "load sequence" switch. In either event, a number is placed in Memory Location 0017(8) that defines the block of part description data where the machining operation is to begin. Then the position control electronics are disabled by IOT DISABL. This step is necessary because the following operation of clearing the velocity storage registers could otherwise cause a distance pulse on either axis. The software Distance Registers 100(8) and 101(8) are cleared, and the block of part description data defined by Memory Location (ML) 17(8) is inserted in the temporary storage. After the data (see Table C-1) are inserted in the temporary storage, the sign of each coordinate is determined, and an IOT command is inserted in ML 360(8) and 367(8). These IOTs will be used as a direction command when this block of data is implemented. Computer command CLRINT clears the interrupt circuits which, in turn, enables the interpolators. Now, any overflow from the velocity generating circuits will cause a slide movement of one pulse. The operate program now loads both velocity storage registers with information stored in 105(8) and 106(8) and checks the software distance registers to insure that they are zero. Since this is the first block, initial software operations have already cleared these registers. As successive data blocks are processed, these tests check for proper system operation. If the correct information was inserted in the distance registers and the proper number of distance pulses outputted, these registers [100(8) and 101(8)] will equal zero at the end of each data block.

Table C-1

PART DESCRIPTION DATA BLOCK

105(8)	Word 1	-	XCLOCK	-	A negative binary number that is read into the toolslide feedrate register.
106(8)	Word 2	-	YCLOCK	-	A negative binary number that is loaded into the workslide feedrate register.
107(8)	Word 3	-	XDISTA	-	A positive number with sign (Bit 0) that defines the number of pulses to be outputted on the toolslide.
110(8)	Word 4	-	YDISTA	-	A positive number with sign (Bit 0) that defines the number of pulses to be outputted on the workslide.
					4 11
111(8)	Word 5	-	MISCRE	-	0 1 2 3
			Bit 0	-	Velocity Error Sign Lag = 0 Lead = 1
			Bit 1	-	Axis with Velocity Error Y = 0 X = 1
			Bit 2	-	Distance Remainder Sign + = 0 - = 1
			Bit 3	-	If Set (1), machine halts after completion of present data block.
			Bit 4-11	-	Number of times this block will be repeated.
112(8)	Word 6	-	RMD	-	Binary number equal to value of error incurred for each distance pulse [7777(8) = 65.104166.. microinches].
113(8)	Word 7	-	DSTMDR	-	Binary number equal to value of error incurred after each block repeat [7777(8) = 65.104166.. microinches].
114(8)	Word 8	-	VELREP	-	Bits 0 - 5 contain a binary sequence number Bits 6 - 11 contain the number of times each feed rate register must repeat for each distance pulse.

Any other number will cause the machine to halt. If all is well at this point in the operate program, the sign commands previously stored are implemented, setting direction control FFs, and software distance registers are loaded with data from the temporary storage.

As stated previously, it is nearly impossible to exactly define any random length line. Each time a block of data is processed, an error occurs. This error is corrected by the part description processor for all cases except when there is one or more block repeats. In this case, the operate program must keep a running account of this distance error and correct either axis when the error reaches one pulse. Information necessary to accomplish this correction is stored in each block of data. A subroutine strips this information from temporary storage (MISCRE and DSTMDR of Table C-1) and stores it for use by correcting a routine which is entered after a block repeat.

Velocity is derived from two 12-bit counters with 6-bit repeat registers. Their essential resolution is determined by the variable clock frequency (Appendix A, Figure A-1) and is centered around 40 microseconds. Here, again, an error is generated because of resolution restrictions, but now there is an error on one axis each time a control pulse is outputted. This error must be monitored and corrected each time it accumulates to one pulse or 65.104166.. microinches. Information concerning this error is stored in MISCRE and RMD. Subroutine SETDUP removes these data and stores them for use by subroutine VELERR where the necessary corrections are made.

At this point in the operate program, both velocity registers are loaded, both distance registers are loaded, and all appropriate information has been stripped from the temporarily stored block of part description. The control system is now ready to start the slide movement which is accomplished by enabling the variable clock that drives both velocity counters. If this particular data block contains block repeats, an error summation is performed by DISSUM and a distance correction is made if necessary. The operate program also determines if this is the last repeat of the block or if there are no repeats. If the latter is the case, the next block of information is retrieved from memory and inserted in the temporary storage [105(8) - 114(8)]. After this step is accomplished, the program flow moves to VELERR where a running account of velocity error is kept and appropriate corrections are made. Control remains in this mode until a PI occurs which signifies that the block of part description information was processed. A more detailed account of each subroutine is contained in the description of the operate program in this section.

When the variable clock was enabled, both the workslide and toolslide velocity counters were loaded with the complement of numbers that will give overflows (command pulses) at the desired pulse rates. Each software distance register was loaded with the complement of the exact number of command pulses. Direction flip-flops were also set that would route the command pulses to the appropriate motor-translator inputs. Clock pulses from the variable clock cause both velocity counters to count down. When either register reaches zero, an overflow occurs and is fed to the translator controller as it reloads the register, routes a command pulse to the proper output through the sign control binary, and feeds an increment command to the data break circuit inputs. The command pulse causes the axis in question to move 65.104166.. microinches and increments either ML 100(8) or 101(8) through the memory increment data break. This operation continues until the tool path defined by the data block is traversed. At this time, either Location 100(8) or 101(8) will change from 7777(8) to 0 during the increment process inside the computer. This action is monitored by the overflow detector of Appendix A, Figure A-4. When this act occurs, a program interrupt forces the operate program to go to ML 0 after the data break is completed, turns the variable clock off, and immediately disables the interpolators. This disabling action prevents spurious signals generated by either pulse generator to be fed to the motor translator. As stated in the design concept, the last pulse from each axis should occur simultaneously. Except in very special cases, such as a 45-degree line, this condition is impossible. Assuming that the program interrupt was generated by the X axis, the Y axis still has one pulse to output. Since both axes are disabled after an interrupt, no pulse occurs. This possibility is circumvented by computer command SKPXY at ML 0001(8) which inserts a command pulse on either axis if no signal arrived from its velocity counter.

Summarizing, the system configuration finds the tool at the end of a path defined by the previous block, the program flow is at 0001(8), and a program interrupt has just occurred. The operate program now determines the interrupt source. Since this interrupt was caused by an end-of-block signal, the program flow jumps to ML 122(8). New feed rate control numbers are read into external X and Y velocity storage registers, the interrupt circuits are cleared through CC CLRINT, and the previous operations are repeated for this and each successive block of part description information until the last block of data is encountered. At this time, the system is disabled and remains in this state until the operator initiates another operation.

DISTRIBUTION

Atomic Energy Commission - ALO

Vespe, V. C.

Atomic Energy Commission - ORO

Hickman, H. D.
Zachry, D. S., Jr

Lawrence Livermore Laboratory

Eckard, R.
Jensen, J. A.
Spies, R. J.
Tech Inf Div

Los Alamos Scientific Laboratory

Stack, F. E.

Oak Ridge Gaseous Diffusion Plant

McLaren, R. A.
Wilcox, W. J., Jr
Winkel, R. A.

Oak Ridge National Laboratory

Adams, R. K.
Borkowski, C. J.

Oak Ridge Y-12 Plant

Alvey, H. E.
Bernander, N. K.
Bowers, G. L.
Burditt, R. B.
Burkhardt, J. H., Jr (5)
Burkhart, L. E.
Butturini, W. G.
Conley, C. E.
Denny, A. (2)
Foulk, D. L.
Gritzner, V. B.
Groppe, W. A.
Hemphill, L. F.
Hensley, C. E.
Jones, F. W.
Kahl, K. G.
Keith, Alvin
Kite, H. T.
Lay, C. M. (5)
Miskell, R. V.
Mitchel, G. W.
Noey, J. L.
Oliphant, G. W.
Perry, A. E.
Read, A. M.
Smelcher, O.
Smith, D. N.
Smith, H. F., Jr
Smith, J. H.
Smith, R. D.
Stephens, A. E. (5)
Stephens, W. E.
Stoner, H. H.
Thompson, C. H.
Tilson, F. V.
Trotter, T. C.
Tuel, E. F.
Tunnell, H. A.
Weathersby, W. E.
Webber, T. R.
Williams, T. L. (5)
Wright, C. C.
Yaggi, W. J.
Y-12 Central Files (45)
Y-12 Central Files (master copy)

Y-12 Central Files (route)
Y-12 Central Files (Y-12RC)

Paducah Gaseous Diffusion Plant

Millican, R.

Sandia - Albuquerque

Gardner, W. A.
Mail Serv Sec

Sandia - Livermore

Davies, L. E.
Library

In addition, this report is distributed in accordance with the category UC-38, **Engineering and Equipment**, as given in the *USAEC Standard Distribution Lists for Unclassified Scientific and Technical Reports*, TID-4500.