# THE DESIGN OF A COMPUTER INTERFACE FOR A RADAR PROCESSOR

by

CARL C. ANDREASEN

B.S. Kansas State University, 1974

A MASTER'S THESIS

submitted in partial fulfillment of the requirements for the degree

MASTERS OF SCIENCE

Department of Electrical Engineering

KANSAS STATE UNIVERSITY

Manhattan, Kansas

1976

Approved by:

Donald R. Hummels

# TABLE OF CONTENTS

Page

# ILLEGIBLE DOCUMENT

THE FOLLOWING DOCUMENT(S) IS OF POOR LEGIBILITY IN THE ORIGINAL

THIS IS THE BEST COPY AVAILABLE

P-474

## LIST OF TABLES

Table

LIST OF FIGURES

Figure

# LIST OF PLATES

Plate

# Chapter 1

The Department of Electrical Engineering at Kansas State University is designing and building a system that will be used to obtain weather data from radar signals. Figure 1.1 shows a block diagram of the system. The system consists of a radar unit, a Digital Processor and the Electrical Engineering Department's Data General Nova 1200 minicomputer. The system components currently available are a surplus APS-81 radar unit, and a Nova 1200 minicomputer with the peripherals indicated in Figure 1.1. The purpose of the Digital Processor is to digitize the video signals coming from the radar unit and transfer the digital data to a mass storage device. A digital interface is required between the Nova and all of its peripherals as shown in Figure 1.1. This thesis covers a design and implementation of a Computer Interface circuit, for the Digital Processor, which controls the movement of the digitized video data to the computer's memory. Before proceeding with a description of the Computer Interface some backround information on how the digitized video signal data is obtained is necessary.

## BACKROUND INFORMATION

The principle of radar involves transmitting a pulse of energy of known magnitude towards an object, and then measuring the amount of the pulse that is reflected and the

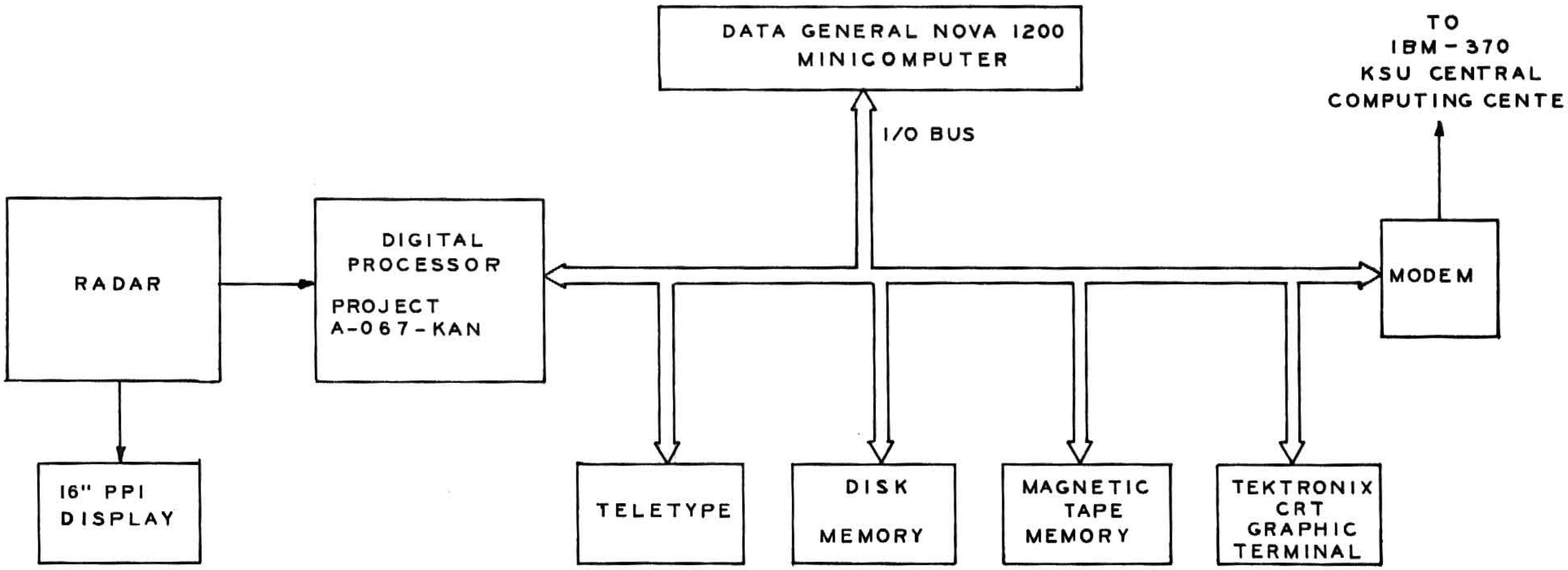



FIGURE 1.1
WEATHER RADAR DATA GATHERING
SYSTEM BLOCK DIAGRAM

time between transmission of the pulse and reception of a reflection, or echo. This time period is used to determine the distance between the transmitter and the object. The amount, or magnitude, of the echo is proportional to the reflectivity and size of the object. The transmitted pulse, as shown in Figure 1.2, is actually a high frequency signal which is amplified and transmitted for a short period of time.

When moisture droplets are of sufficient radius and quantity to fall as rain they will reflect the transmitted signal producing a detectable echo. In this manner an observer will "see" on the display device, a Pulse Position Indicator scope, the echoes received from falling rain. The radar antenna has a beamwidth of 5 degrees horizontally and 10 degrees vertically and is rotated clockwise 360 degrees every four seconds With 800 high frequency pulses transmitted per second the APS-81 will receive unambiguous echos of objects that are less than 100 miles away from the antenna.

The reflected pulses are detected in the radar receiver to produce a video signal for each of the transmitted pulses. The magnitude, or voltage, at any point along the video signal is proportional to the magnitude of the echo from the associated transmitted pulse. There is a direct time to distance relationship in the video signal as shown in Figure 1.3

The antenna starts at true North and revolves 360

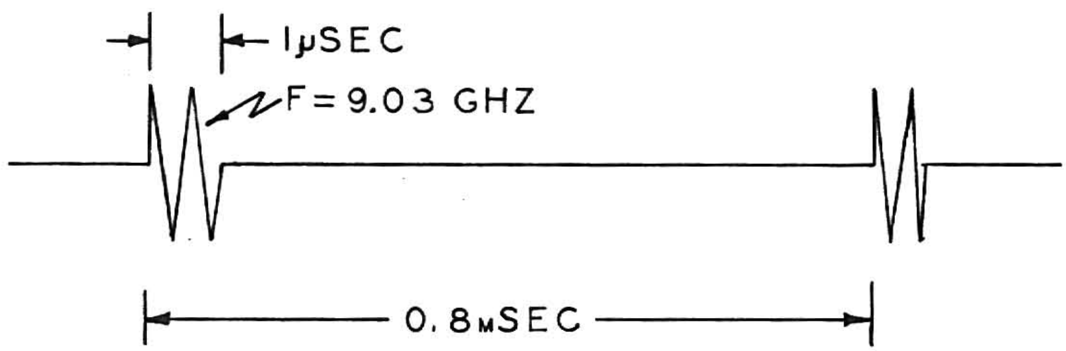



FIGURE 1.2

THE RADAR TRANSMITTER SIGNAL

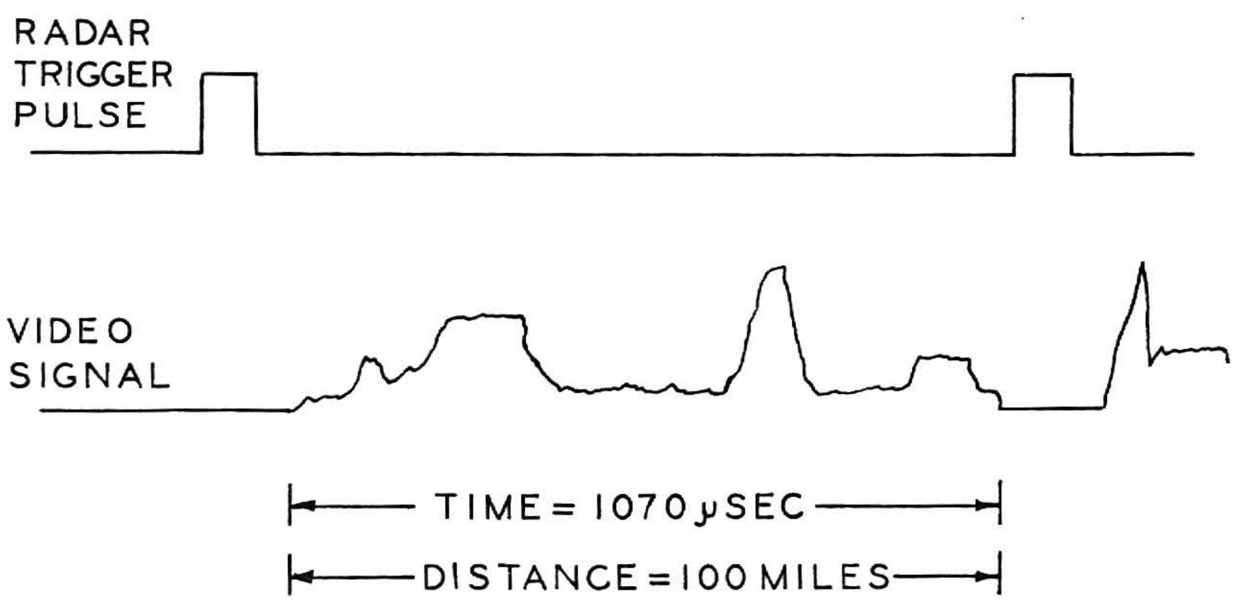



FIGURE 1.3

VIDEO SIGNAL TIME TO
DISTANCE RELATIONSHIP

degrees. The azimuth is defined as the angular distance from true North that the antenna is facing. The azimuth is determined by using a shaft encoder. The shaft encoder has a binary output that is equal to the angle that the encoder's shaft is with respect to a fixed point. The shaft encoder output is decoded in incremental amounts and then the video signals that occur in this incremental amount are integrated. The integral of the magnitude of the video signal over a specified interval and incremental azimuth is the echo intensity for a region, and is called a range bin integral. A range bin is defined geometrically as a sector of an annulus whose side is one-half mile and angle is the incremental azimuth value. Figure 1.4 shows the division of the radar pattern into the range bins or range azimuth cells. With the above backround information a brief description of the operation of the components of the Digital Processor block can now be presented.

## THE DIGITAL PROCESSOR COMPONENTS

The Digital Processor block is composed of three circuits as shown in Figure 1.5. The Logarithmic Video Amplifier is used to compress the large dynamic range video signal into a video signal with a ten step dynamic range. The Integration and Quantization Circuit will integrate the video signal for each of 200 seperate intervals, each interval corresponding to a one-half mile segment of range.

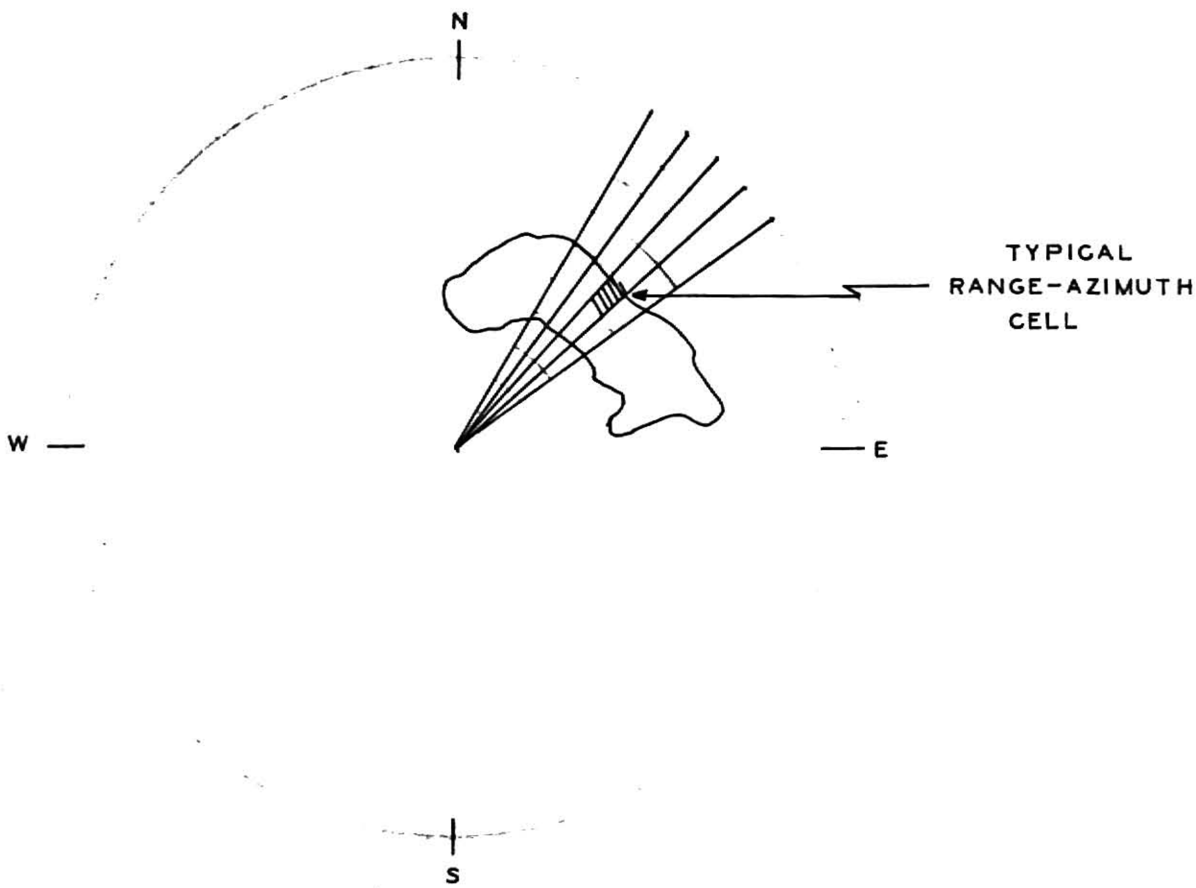



FIGURE 1.5
TYPICAL RANGE-AZIMUTH CELL

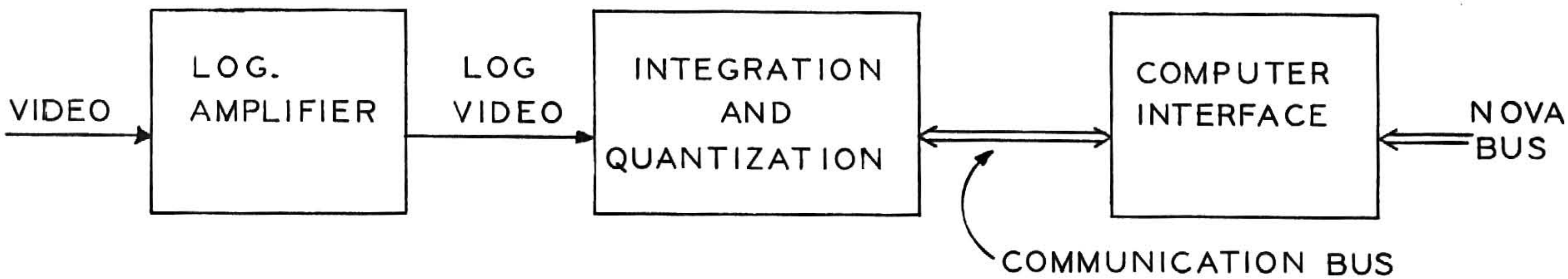



FIGURE I.4

DIGITAL PROCESSOR BLOCK DIAGRAM

Thus the tenth interval is that part of the video signal corresponding to 4.5 to 5.0 miles away from the radar antenna. The time of integration is the time that a point object would be within the antenna beam, during one scan. The Quantization part of the Integration and Quantization circuit converts the voltage on a capacitor, which represents the range bin integral, to a Binary Coded Decimal (BCD) value, the range bin intensity. The Integration and Quantization circuit generated the range bin intensities and now the range bin intensities are passed on to the Computer Interface circuit for eventual transfer to a mass storage device.

Each one-half mile segment of the video signal corresponds to 5.37 microseconds. After the period of integration each range bin intensity will be present on the output of the Integration and Quantization circuit for only 5.37 microseconds. This rate is faster than most mass storage devices can handle. The solution is to collect the data in a computer's memory and then move the data to a mass storage device in whatever format is needed. The sequence of events necessary to obtain and store in the computer's memory all of the range bin intensities for one revolution of the antenna is termed the "data acquisition sequence". The computer used is a Data General Nova 1200 minicomputer with 28K words of memory.

The Computer Interface circuit is needed to control the incremental azimuth over which integration of the video signals is performed and control the movement of the range bin intesities from the Integration and Quantization Circuit to the computer's memory. This thesis covers the design and implementation of the Computer Interface circuit which accomplishes these tasks.

# Chapter 2

This chapter presents a detailed explanation of the internal operation of the Computer Interface Circuit. Figure 2.1 shows the block diagram of the Computer Interface Circuit, hereafter referred to as the interface or interface circuit. Plate I shows the Computer Interface circuit as implemented. The interface is necessary to synchronize the movement of the range bin intensities from the Integration and Quantization circuit to the Nova minicomputer. The interface also decodes the azimuth and provides signals which control the azimuth interval over which the video signals are integrated.

In order to describe the internal operation of the interface a description of the signals provided and required by the interface is required. The first section of this chapter presents a description of the signals required and provided by the Nova minicomputer . The next section presents a description of how the azimuth is decoded, the signals generated and their use. The next two sections provide a description of the signals used to communicate with the Integration and Quantization circuit and a description of the operation of the registers used to temporarily store the range bin intensities respecively. The final section of this chapter provides a description of the circuit which simulates the azimuth and the signals coming from the Integration and Quantization circuit.

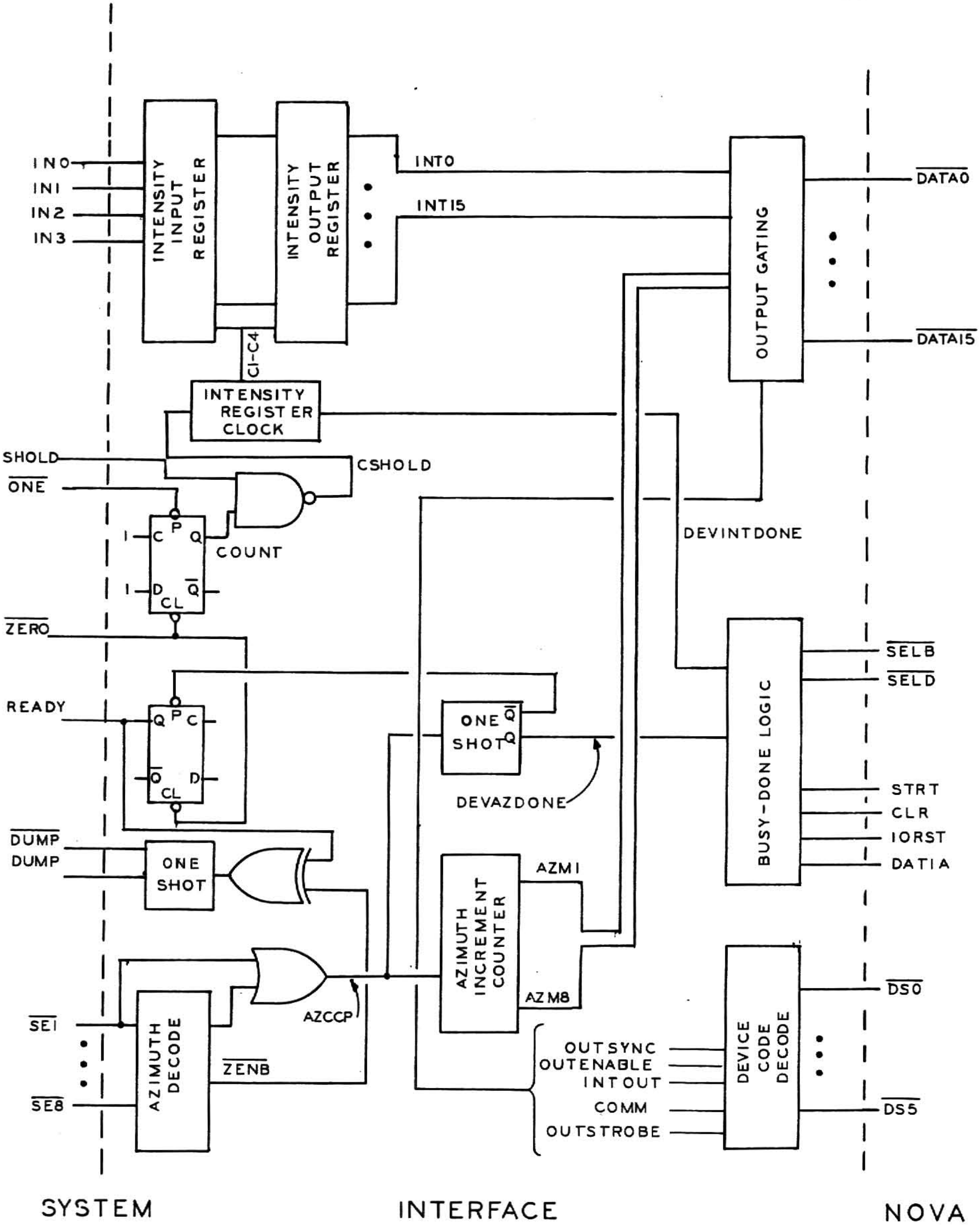
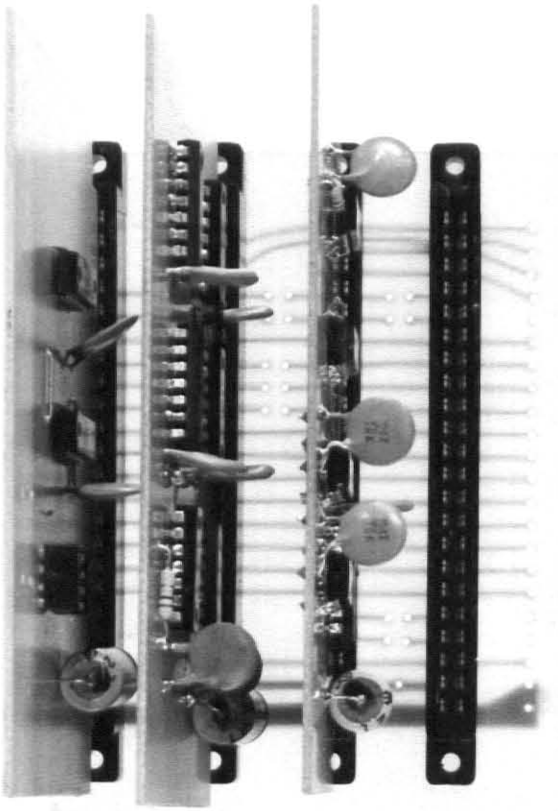


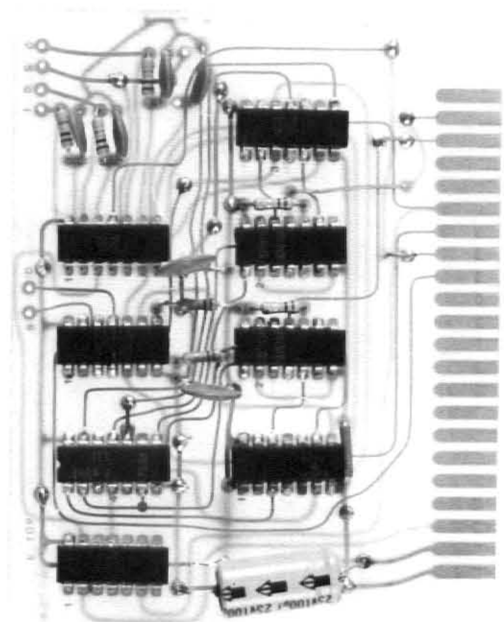

FIGURE 2.1

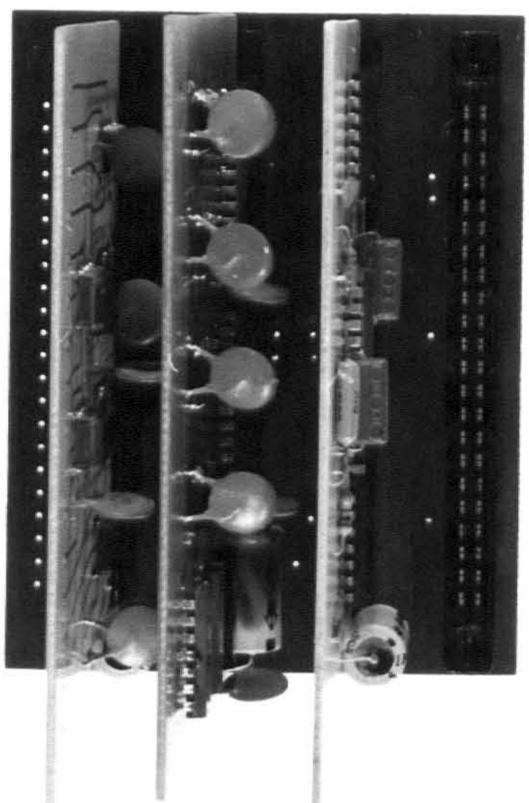
COMPUTER INTERFACE BLOCK DIAGRAM

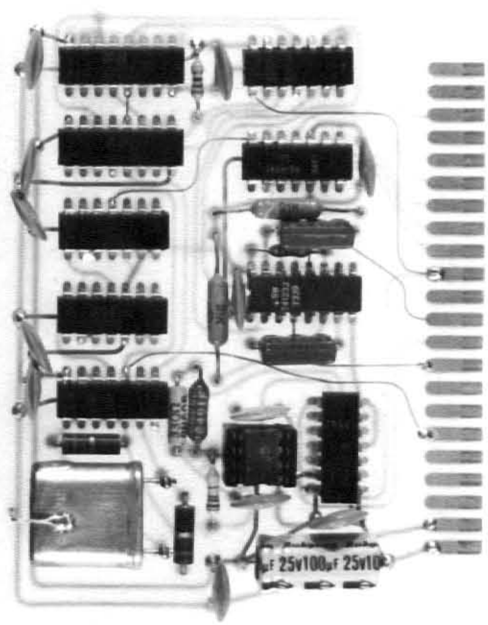
PLATE I

THE COMPUTER INTERFACE CIRCUIT AS IMPLEMENTED









# THE NOVA CHARACTERISTICS

One method for a computer to communicate with its peripheral devices is the use of a group of conductors, called a bus, which runs to every device. The bus carries the necessary electrical signals to allow communication between the computer and peripheral devices. The Data General Nova 1200 (hereafter is referred to as "computer") bus carries to every peripheral:

1. Six device select on lines, $\overline{DS0}$-$\overline{DS5}$, from computer to the devices.
2. Nineteen Control lines from the computer to the devices to synchronize data movement to and from the bus. For example the IORST signal that resets all peripherals.
3. Six control lines from the devices to the computer, such as the Interrupt Request line.
4. Sixteen bidirectional data lines, $\overline{DATA0}$-$\overline{DATA15}$.

Since this bus is shared by all peripherals a separate interface is provided for each peripheral device as illustrated in Figure 2.2. As shown some interfaces can be included within the computer mainframe. For additional peripheral devices the bus can be extended, as in the case of the Computer Interface circuit for the Radar Processor.

When using the external bus the control signals that originate in the computer and drive the peripheral interfaces must be terminated as shown in Figure 2.3. The

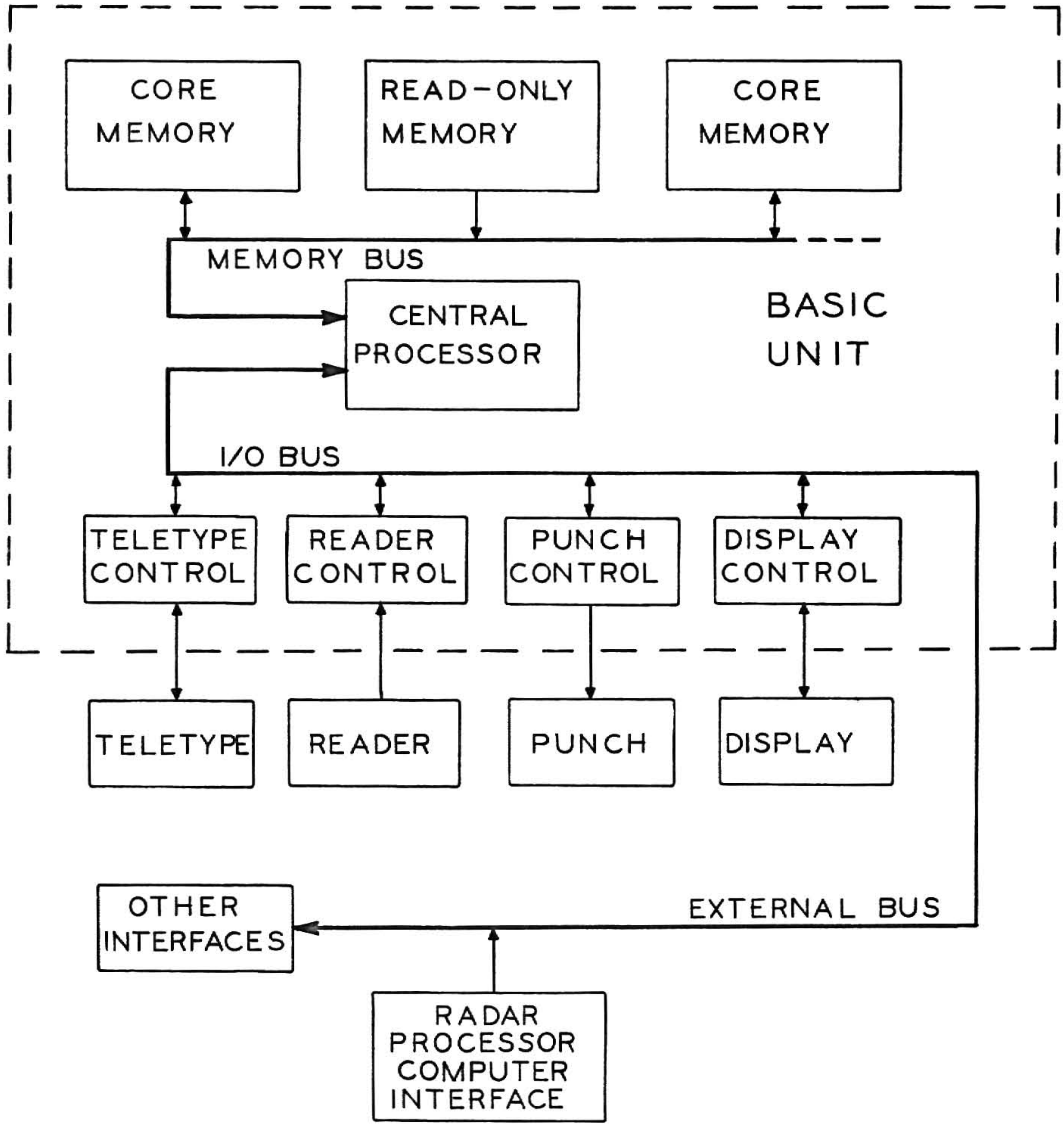



FIGURE 2.2

NOVA 1200 ARCHITECTURE (3)

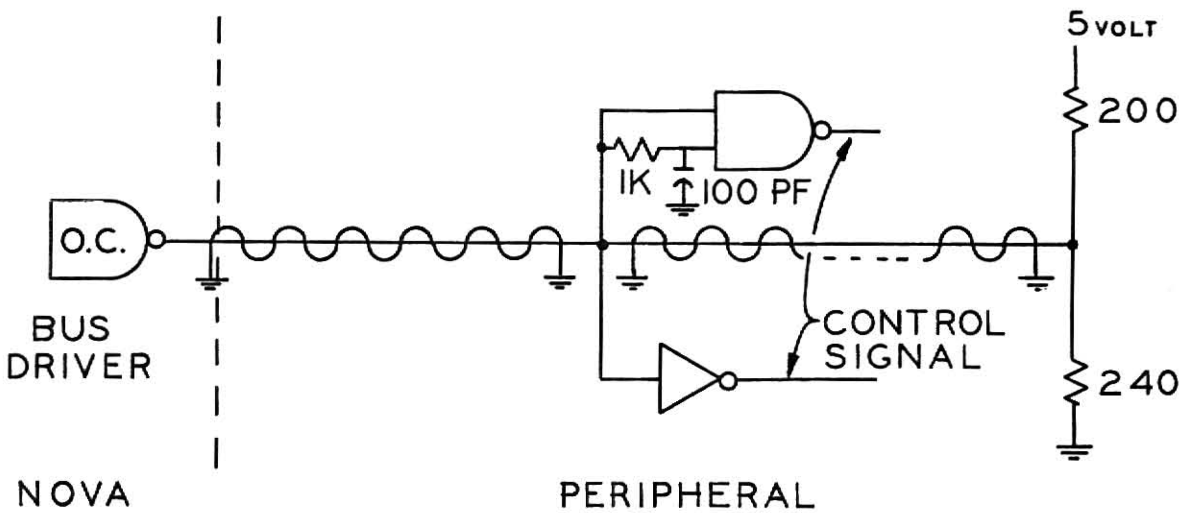



FIGURE 2.3

TERMINATION OF PROCESSOR
TO DEVICE SIGNALS

signal receivers are often an inverter or a nand gate with a filter on the input lines. The voltage divider at the end of each bus line gives the correct termination.

The bidirectional signals, such as the data lines, and signals which come from the peripheral interfaces to the computer, such as the $\overline{\text{SELB}}$ line, must be terminated as shown in Figure 2.4. The bidirectional lines are "active low" in that pulling a line low is equivalent to placing a logical 1 signal on that line. Thus the output transistor of an open collector gate, such as the SN7438, is used for the line transmitter and an inverter is used for a line receiver. Note that all bus lines must be twisted pairs.

Device Selection and Data Transfer

The computer provides a six bit device selection code on the device selection lines $\overline{\text{DS0}}$-$\overline{\text{DS5}}$. These lines are active low and are decoded so that each interface has an enable line that is high when the interface's device code is on the device select lines. The complement of the enable line is used to synchronize the transmission or reception of data when one of the Data In (DATIA, DATIB, DATIC) or Data Out (DATOA, DATOB, DATOC) lines is active respectively. This interface transfers an azimuth value and the range bin intensity word so to distinguish between them, two device select codes are used. One device selection code (63) is used to synchronize the transfer of the output of the

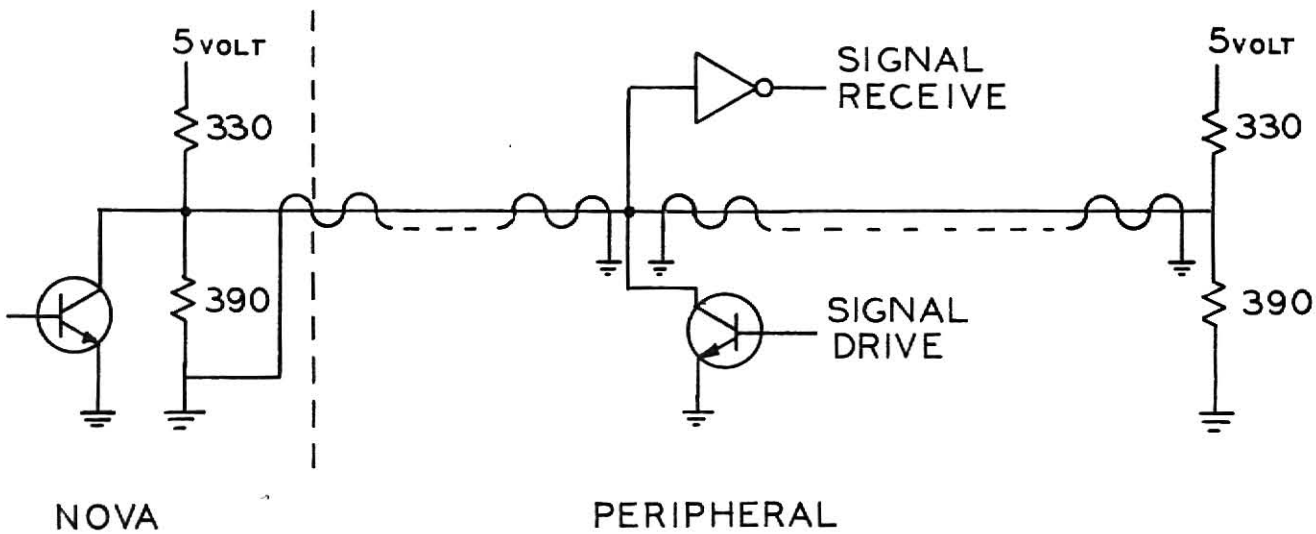



FIGURE 2.4

TERMINATION AND GENERATION OF
BIDIRECTIONAL BUS LINES

Azimuth Increment Counter to the data bus, and the other device selection code (67) is used to tranfer the output of the Range Bin Intensity Output Register to the data bus.

Figure 2.5 shows the logic necessary to generate the enable lines DEVINT and DEVAZ required by this interface.

Figure 2.6 gives the timing diagram for a typical input instruction. This instruction is of the form:

Mnemonic(code) AC,Device Code

where the Mnemonics are:

| | |
|---|---|
| DIA | Read data from the A buffer |
| DIB | Read data from the B buffer |
| DIC | Read data from the C buffer |

Note from the above that with each peripheral there may be up to three buffers from which data may be obtained. The optional code that can be concatenated to the mnemonic will be explained in further detail in the next section of this chapter. AC stands for one of the four accumulators (0,1,2,3), and Device Code is the two digit octal device code placed on the device selection lines.

Execution of the DIA instruction causes the device code to be placed on the device selection lines and the DATIA line to go high. When this condition is decoded by a device interface an enable line is generated which gates the data from the device interface output buffer to the data bus. Figure 2.5 shows the data output enable lines INTOUT and AZOUT that are required to gate the Azimuth Increment Counter or the Range Bin Intensity Output Register to the

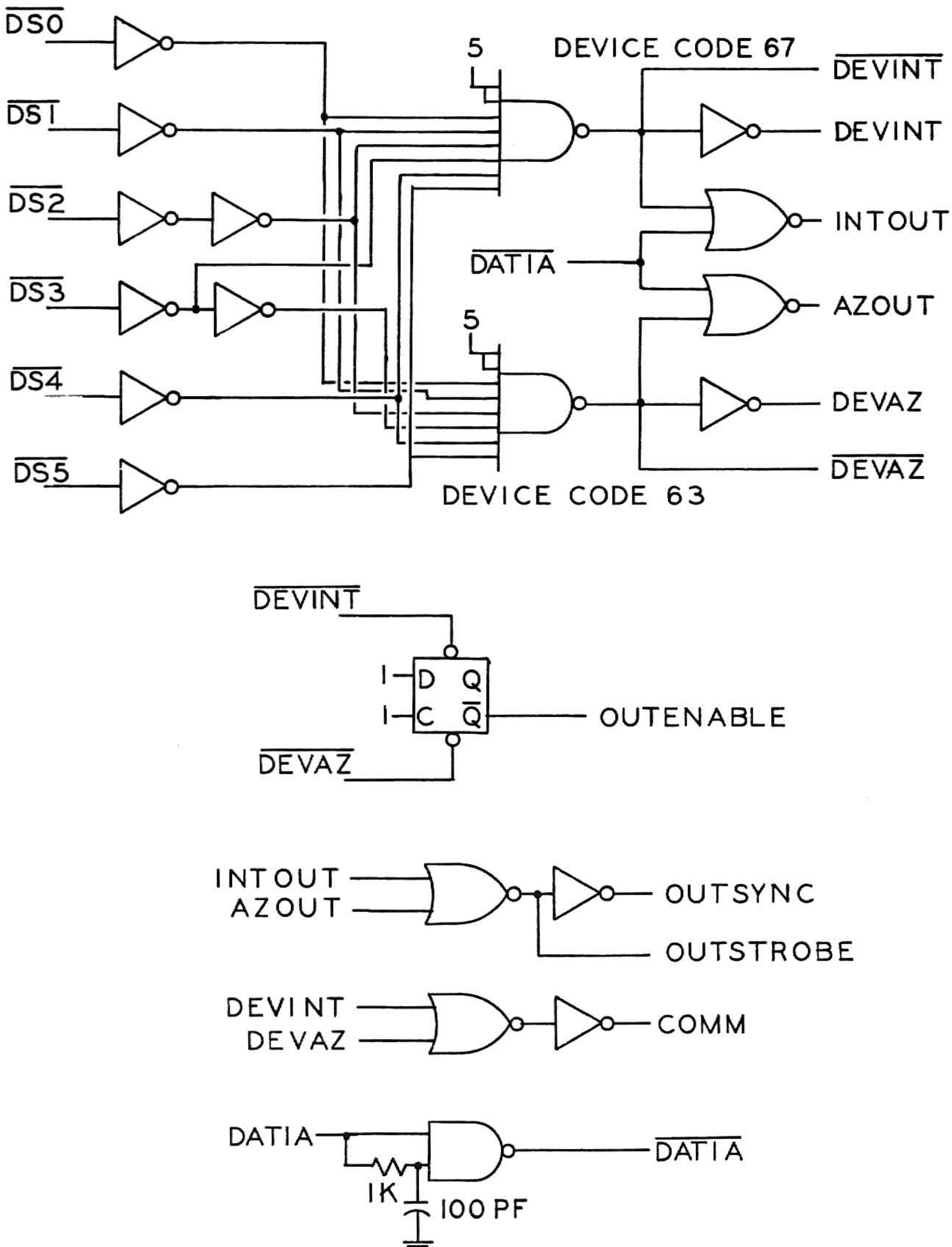



FIGURE 2.5
DEVICE CODE DECODING CIRCUIT

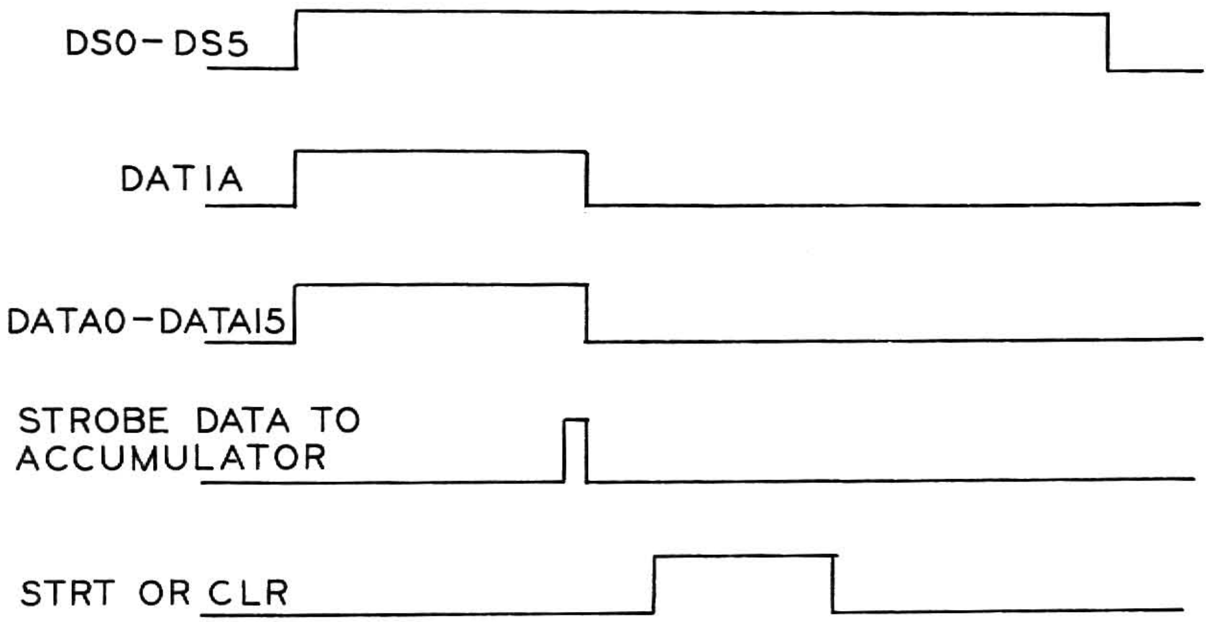



FIGURE 2.6

TIMING OF AN INPUT INSTRUCTION

bus.

Figure 2.7 presents a brute force method of gating the data from the Range Bin Intensity Output Register and the Azimuth Increment Counter to the data bus. A better method is shown in Figure 2.8. This method requires fewer packages and reduces the number of gate inputs that must be driven by the INTOUT signal.

## The Interface Status Circuit

Each peripheral device interface has a four flipflop control network that specifies the state of the device and requests an interrupt by that device (Busy, Done, Interrupt Disable and Interrupt Request). Figure 2.9 shows the flipflop control network suggested. When an input device has data ready to be sent to the computer, the device sets its Done flipflop. The computer will clear the Busy and clear the Done flipflops of an output device when data is placed in the buffer for that device. When the output device is ready for more data it will set its Done flipflop.

The computer can determine when a device is ready for service by two methods. The first method involves periodically checking the Selected Done ($\overline{SELD}$) line which is brought low when the device Done flipflop is set and its device select code is on the device select lines. The execution of a skip-on-done instruction (SKPDN) in the program initiates this check. The second method is for the

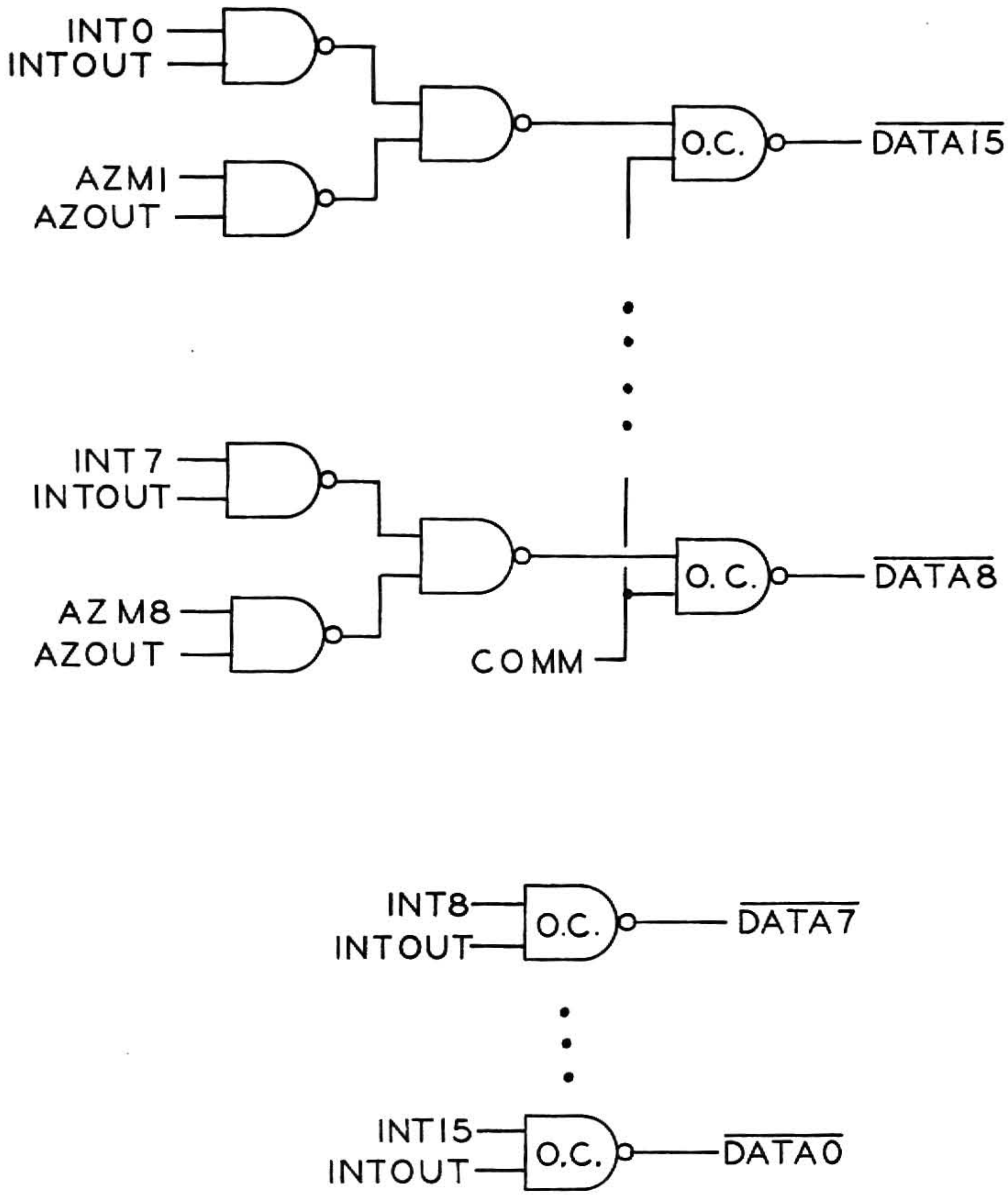



FIGURE 2.7

BRUTE FORCE DATA OUTPUT GATING

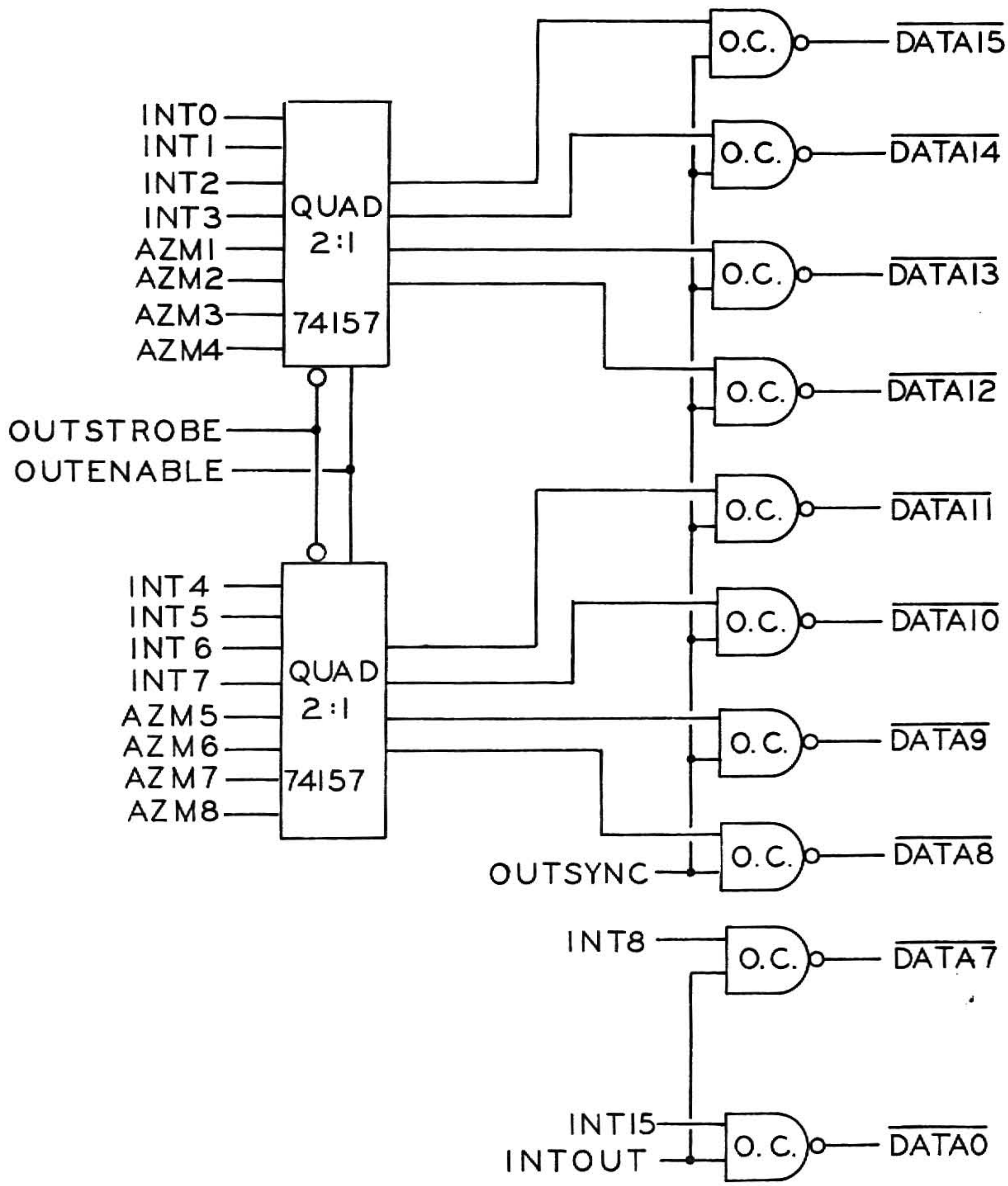



FIGURE 2.8

LSI GATING OF OUTPUT DATA

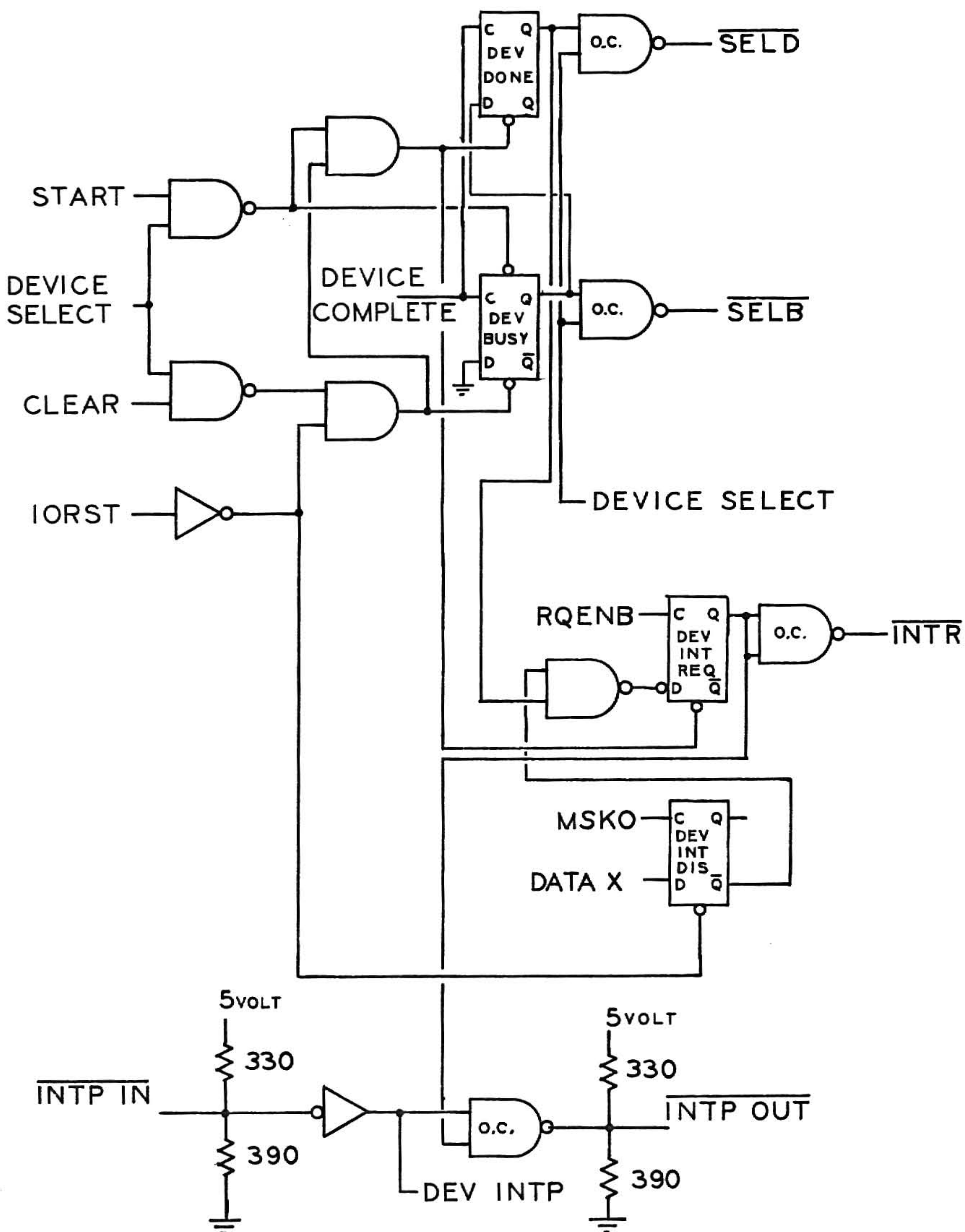



FIGURE 2.9
THE FOUR FLIPFLOP CONTROL NETWORK
FOR PERIPHERAL INTERFACES (3)

device interface to interrupt the computer. During each computer cycle, at a time when the computer is capable of being interrupted, the Request Enable ($\overline{\text{RQENB}}$) line is pulsed. One or more devices needing service will then pull the Interrupt Request ($\overline{\text{INTR}}$) line low if their associated Interrupt Disable flipflop is clear.

When an interrupt occurs the program counter (PC) is stored in location 0 and the computer begins executing instructions at the location whose address is found in location 1. Location 1 contains the address for the routine that services interrupts. The service routine must execute an Interrupt Acknowledge (INTA) instruction to determine the device code of the device closest on the bus that is interrupting. The INTA instruction pulses the $\overline{\text{INTP IN}}$ and INTA lines and the device interrupting, which is closest to the computer places its device code on the data bus lines $\overline{\text{DATA10}}$-$\overline{\text{DATA15}}$. A more detailed description of interrupt servicing can be found in (3,5).

If the interrupt feature of the Nova is used then a short interrupt handler must be written to process interrupts from the interface. The interrupt handler can be broken down into several steps. These steps and their associated execution time in microseconds are:

| | | | |
|---|---|---|---|
| Finish current instruction | | 1.35 | maximum |
| Store Program Counter | | 1.35 | |
| Jump to service routine | | 1.20 | |
| Execute: | INTA 3 | 2.55 | |

Execute: JMP @DISP,3 1.35 + 1.20

Execute program segement to read appropriate data

Execute: INTEN 3.15

The total time to exectue the above, less the time to execute the program segement to read the appropriate data, is 12.15 microseconds.

The minimum time required to read the azimuth increment count is given in Figure 3.4 and is 10.80 microseconds. Thus to handle an interrupt and read the azimuth increment count would take:

12.15 + 10.80 =22.95

microseconds. The time necessary to read the range bin intensity word is also given in Figure 3.4 and is 17.05 microseconds. Thus to handle an interrupt and read the range bin intensity word would take:

12.15 + 17.05 = 29.75

microseconds. The range bin intensity word (4 range bin intensities) must be transfered within 21.48 microseconds. Using the interrupt feature of the Nova this time requriement cannot be satisfied. Therefore the interrupt feature of the Nova cannot be used and need not be implemented in the interface. This simplifies the required implementation of the interface status circuit to that shown in Figure 2.10.

To set the Busy and Done flipflops to known states, an optional code letter can be concatenated to the input/output instruction mnemonic. The code letters are:

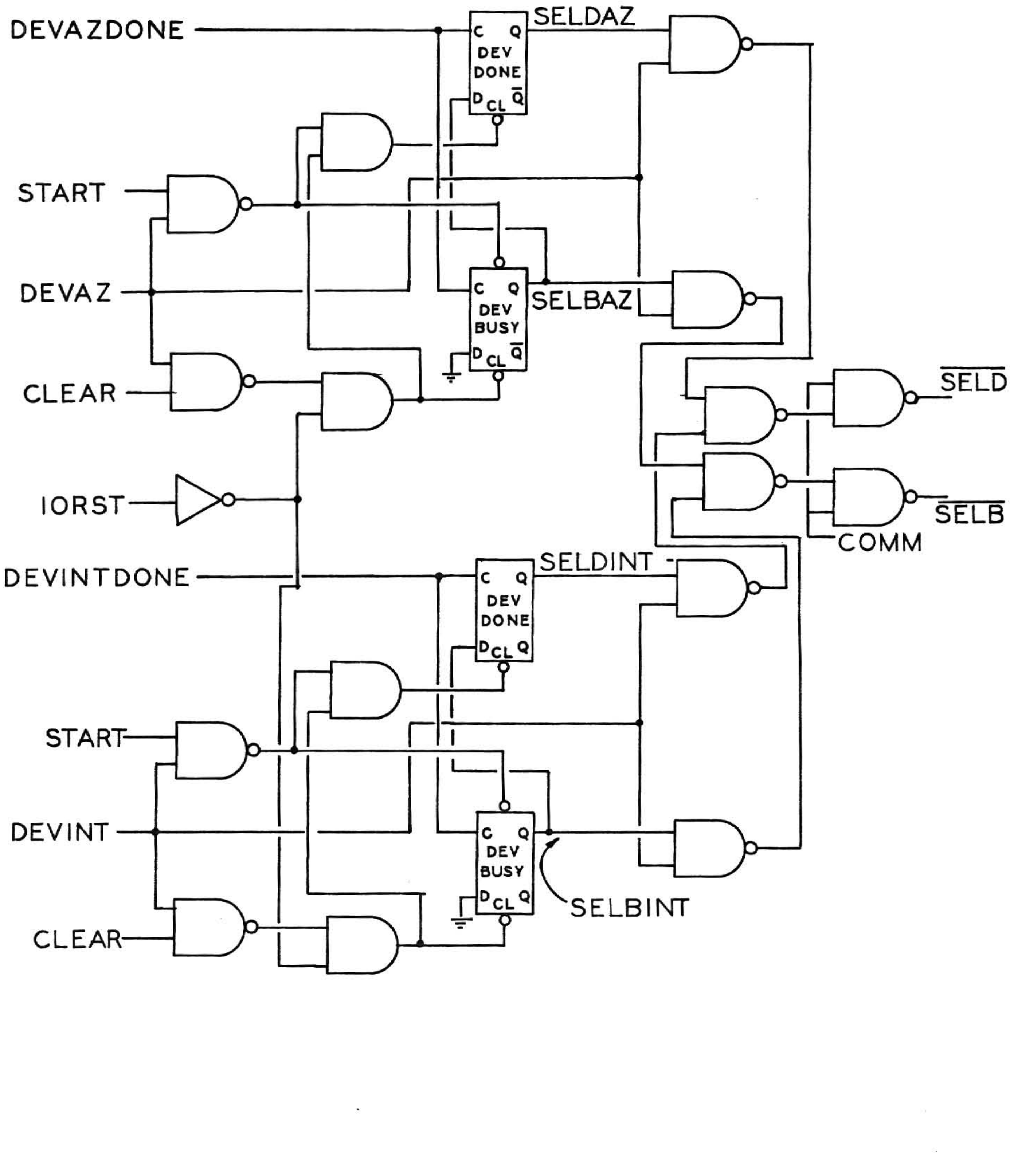



FIGURE 2.10
BUSY-DONE FLIPFLOPS IMPLEMENTED

| | |
|---|---|
| none | Leave Busy and Done as they are |
| S | Set Busy and clear Done |
| C | Clear both Busy and Done |
| P | Special pulse on IOPLS line |

To distinguish which data value, azimuth increment count or range bin intensity word, is available to be transfered it is necessary to have two device codes and two Busy-Done circuit implementations or a status register to indicate which data value is available. To reduce the complexity of the circuit and increase the speed of the control program two device codes are used on this interface.

The signal DEVINTDONE is the clock pulse C4 which moves the Range Bin Intensity Input Register to the Range Bin Intensity Output Register. When this move occurs the data is ready to be transferred to the computer memory so SELDINT is set and the SELD line is pulled low when DEVINT is active.

The signal DEVAZDONE is the output of a monostable which is triggered each time the azimuth increments by 5.625 degrees. The azimuth increment count is in the Azimuth Increment Counter. A new sequence of moving the range bin intensities must start each time DEVAZDONE occurs.

## AZIMUTH INCREMENT COUNTER

The azimuth is determined from decoding the 13 bit binary output of the shaft encoder into two pulses. The first pulse occurs when the azimuth reaches 0 degrees, and the second pulse occurs each 5.625 degree increment. The incremental azimuth will be used to clock a counter whose output will be a binary value equal to the number of 5.625 degree increments.

The shaft encoder makes a full count in 32 revolutions of the radar antenna. The shaft encoder starts numbering with bit 1 and this convention gives the correlation between bit place and degree of revolution shown in Table 2.1. From Table 2.1 it can be seen that only SE1 to SE8 need be decoded to detect 0 degrees. When SE1-SE8 are all low, or $\overline{SE1}$-$\overline{SE8}$ are all high, the antenna is pointing towards North and a data aquisition sequence can now be performed. The condition of zero azimuth decoded and a flipflop is set as shown in Figure 2.11. The lowest order bit of the shaft encoder cycles from low to high to low in 5.625 degrees. The falling edge of $\overline{SE1}$ is used to clock the Azimuth Increment Counter. This allows the azimuth counter to be clocked 2.8125 degrees before the shaft encoder shows a multiple of 5.625 degrees, thus the data is valid when the counter output is read. The Azimuth Increment Counter output will be the binary number of increments of 5.625 degrees. Thus the azimuth data word is decoded to indicate which wedge of the radar's scan the data following will represent.

In four seconds the antenna sweeps 360 degrees and the lower 8 bits of the azimuth shaft encoder counts from $2^0$ to $2^7$ . Thus the time per count is:

$$T=(4\ sec)/(2^7\ count) = 31.25\ millisec/count$$

Table 2.1 Correlation Between Azimuth Angle and Shaft Encoder Bit Position

| Bit Position | Azimuth Angle |
|---|---|
| 13 | 11520 |
| 12 | 5760 |
| 11 | 2880 |
| 10 | 1440 |
| 9 | 720 |
| 8 | 360 |
| 7 | 180 |
| 6 | 90 |
| 5 | 45 |
| 4 | 22.5 |
| 3 | 11.25 |
| 2 | 5.625 |
| 1 | 2.8125 |

Thus, each 5.625 degree increment will take 62.50 milliseconds.

The circuit in Figure 2.11 is the circuit for decoding the azimuth. To obtain a 1 to 0 state transition on the ZENB line all inputs to the 8-input nand gate must be high. This transition sets the Zero flipflop, which is cleared by an IORST instruction. When $\overline{\text{ZENB}}$ is high the complement of AZ1 becomes the clock, AZCCP, to the Azimuth Increment Counter. The timing diagram for the circuit shown in Figure 2.11 is shown in Figure 2.12.

The outputs of the Azimuth Increment Counter are fed to the output gating network and placed on the data bus lines $\overline{\text{DATA0}}$-$\overline{\text{DATA7}}$. The Azimuth Increment Counter is read by executing a DIA instruction with device code 63.

## THE SYNCHRONIZATION SIGNALS

There are several signals required to synchronize transfer of the range bin intensities from the Integration and Quantization circuit to the computer interface. These are: SHOLD, $\overline{\text{ZERO}}$, $\overline{\text{ONE}}$, READY, and DUMP. The first three signals indicate when the intensity range bins are being transferred. The last two signals inform the Integration and Quantization circuit that the computer interface is ready to accept the range bin intensities.

Figure 2.13 shows how the signals $\overline{\text{ZERO}}$, $\overline{\text{ONE}}$, and SHOLD are generated on the Integration and Quantization circuit.

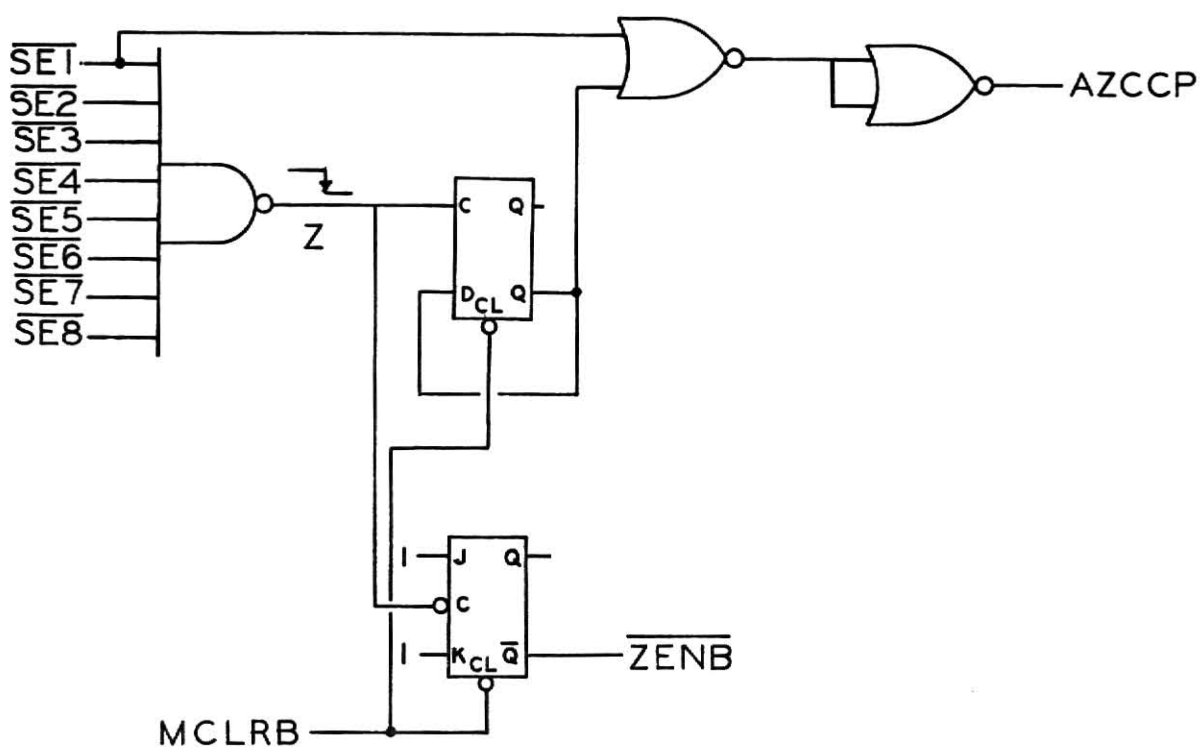



FIGURE 2.11

DECODING THE AZIMUTH

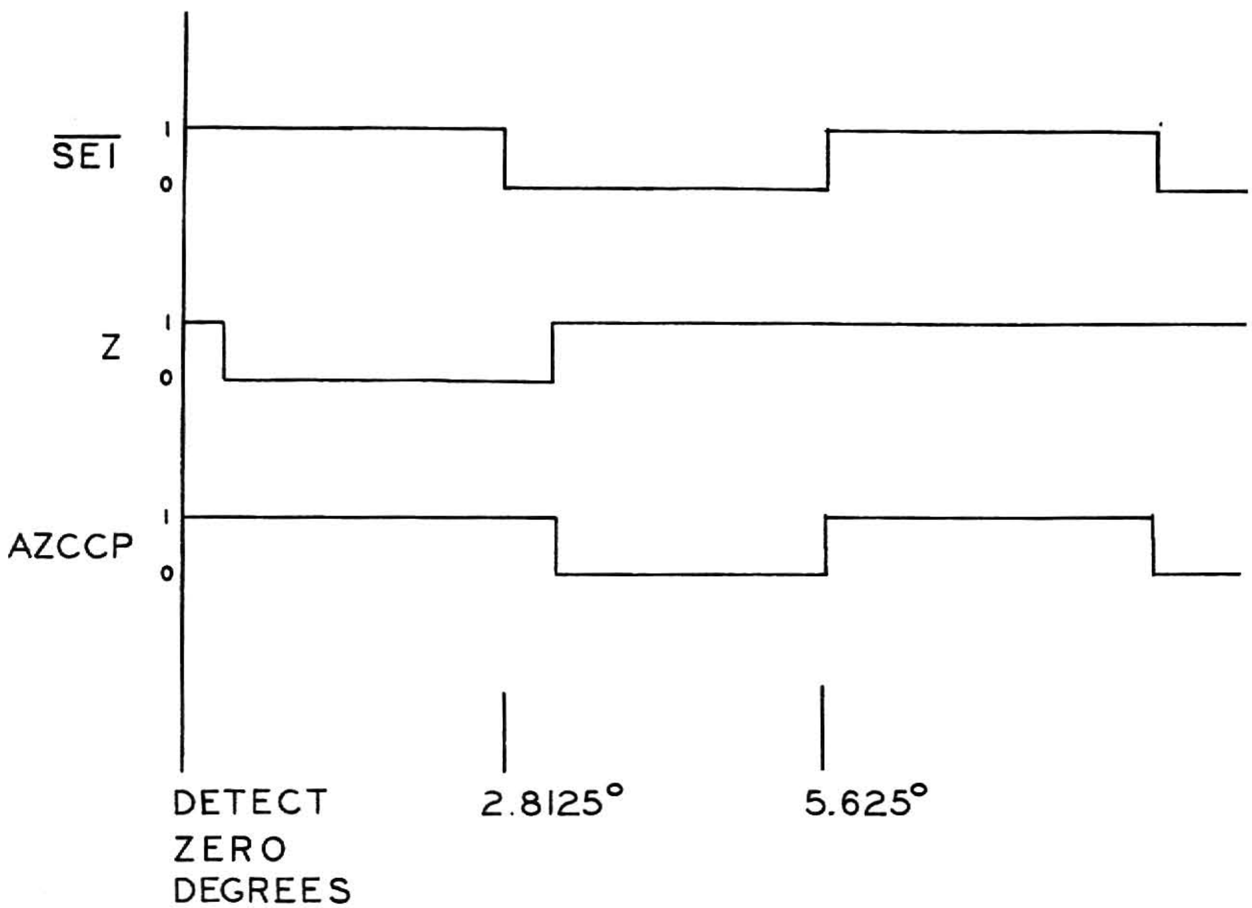



FIGURE 2.12

TIMING DIAGRAM FOR DECODING THE AZIMUTH

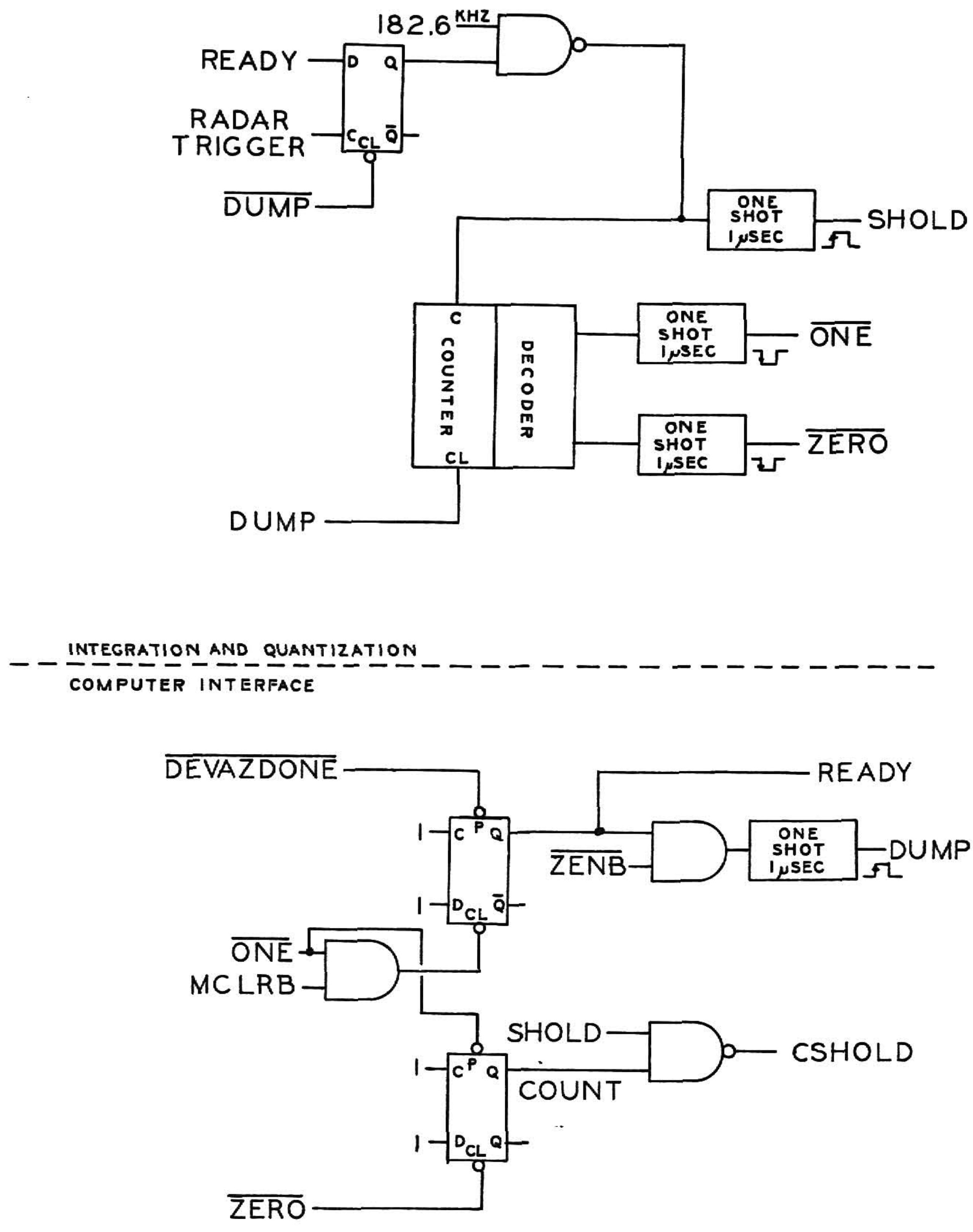



FIGURE 2.13

GENERATION OF THE COMPUTER INTERFACE TO INTEGRATION AND QUANTIZATION CIRCUT SYNCHRONIZATION SIGNALS

Figure 2.11 illustrates how these are used in the computer interface and how READY and DUMP are generated.

The signal $\overline{\text{ONE}}$ is a pulse that goes low for one microsecond when the range bin counter reaches the count of 1. The count of 1 is used because at this time the voltage on the capacitor for range bin 0 has been digitized and its value is present and stable at IN0-IN3. When the range bin count gets to one the pulse $\overline{\text{ONE}}$ sets the count flipflop and the sample and hold signal SHOLD becomes the clock to the Range Bin Intensity Input Register Clock Counter. The counter is advanced by 1 and the data from range bin 0 is clocked into the first four bits of the Range Bin Intensity Input Register.

The signal $\overline{\text{ZERO}}$ is a pulse that goes low for one microsecond when the range bin counter reaches a count of zero. The signal $\overline{\text{ZERO}}$ is used to stop the sequence of movement of range bin intensities into the Range Bin Intensity Input Register by resetting the Count flipflop. When the Count flipflop is reset then SHOLD can no longer clock the Range Bin Intensity Input Register Clock Counter. The $\overline{\text{ZERO}}$ pulse is also used to reset the Ready flipflop.

READY is synchronized with the radar trigger pulse and used as an enable line so that the $\overline{\text{ZERO}}$ and $\overline{\text{ONE}}$ pulse occurs only when READY is high.

The DUMP pulse goes high for one microsecond when a zero azimuth crossing is detected and after the range bin intensities for a azimuth increment have been transfered and stored in the computer's memory. The DUMP pulse is used to reset the READY and Radar Trigger Pulse synchronization flipflops and to discharge the integration capacitors on the Integration and Quantization circuit.

## THE INTENSITY REGISTERS

Two registers are required in order to buffer the range bin intensities and provide enough time to move them to the computer's memory. The Range Bin Intensity Input/Output Register diagram is shown in Figure 2.14. The Range Bin Intensity Input Register is divided into four four-bit nibbles. Each nibble contains one range bin intensity. The range bin intensities come once every 5.37 microseconds, thus moving four of them into a 16 bit register giving computer 21.48 microseconds to read the data into the memory.

The Range Bin Intensity Input Clock Counter is a two bit Gray code counter whose output is decoded to provide four clock pulses as shown in Figure 2.14. The 74107 dual JK flipflop is used for the Gray code counter so that the falling edge of CSHOLD clocks the counter. When SHOLD goes high, CSHOLD goes low, advancing the counter, if READY is high. The rising edge of the clock cycles C1-C4 are used to clock the data into the D flipflops used in the Range Bin Intensity Input Register. When SHOLD goes high, so will , after a few gate delays, one of the clock cycles. The data

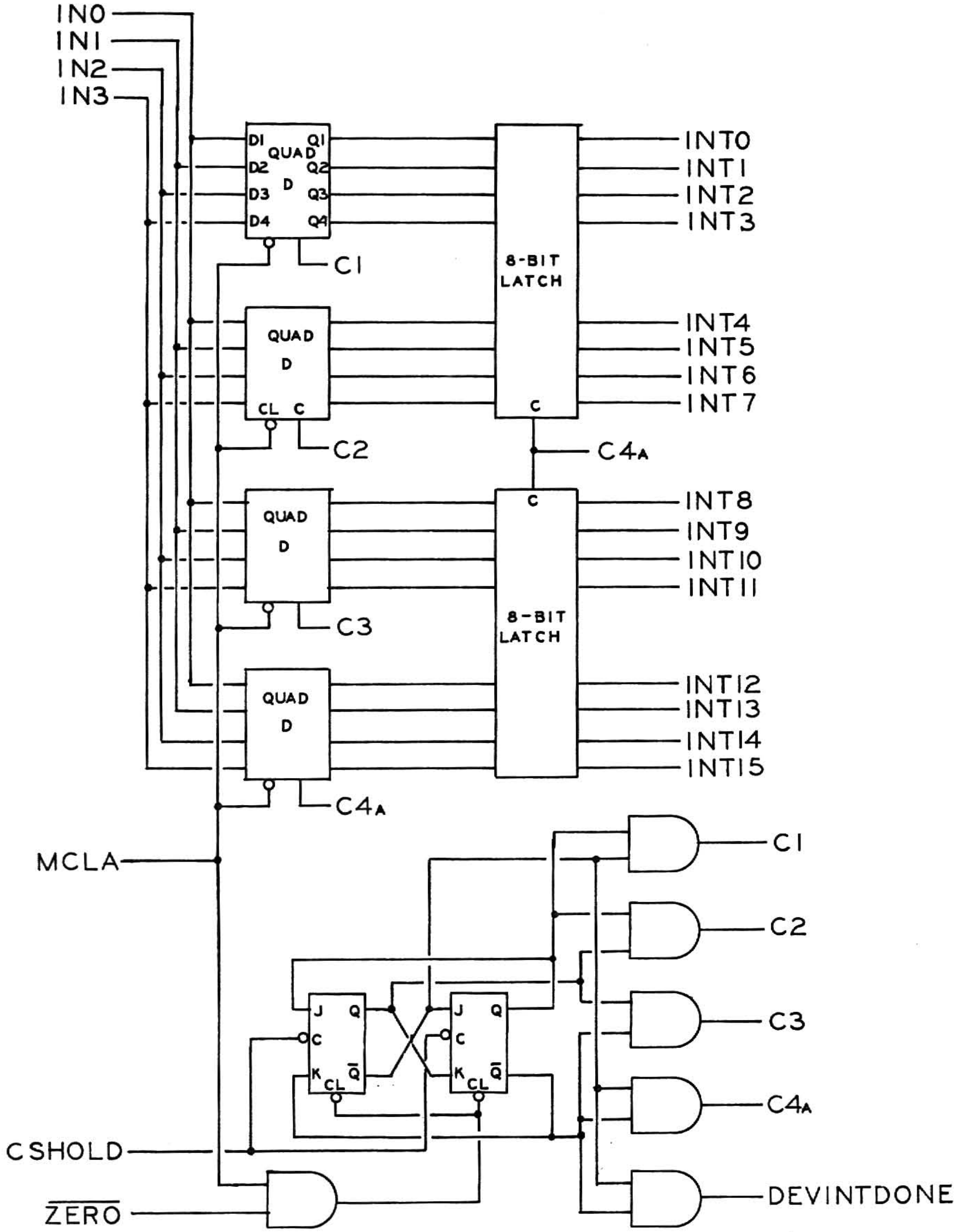



FIGURE 2.14
INTENSITY STORAGE REGISTERS

for a range bin is stable on IN0-IN3 when SHOLD goes high and is the data for the capacitor N-1 where N=count appearing on the Range Bin Counter on the Integration and Quantization Circuit.

Clock phase C4 is used for several purposes. First C4 is used to enable the latches that compose the Range Bin Intensity Output Register. D-type flipflops could be used and a delayed C4 clock phase could clock the data from the Range Bin Intensity Input Register to the Range Bin Intensity Output Register, but use of the latches reduces package count and still maintains data transfer reliability. C4 is also used to set the Done flipflop that is associated with the intensity registers. When this Done, SELDINT, is high the data is ready to be transferred from the Range Bin Intensity Ouput Register. Appendix A contains the circuit diagrams for the Computer Interface as implemented. Also provided is a system timing diagram.

## THE SELF TEST CIRCUITS

To provide a means of testing the interface and control program two self test circuits are provided. Figure 2.15 shows the circuit used to simulate the azimuth. The azimuth consists of an 8 bit binary value and is decoded to provide the proper azimuth increment. The circuit board IAQ-2 (see Appendixes A and B) contains the pseudo-azimuth circuit of Figure 2.15 and is removed when the shaft encoder provides

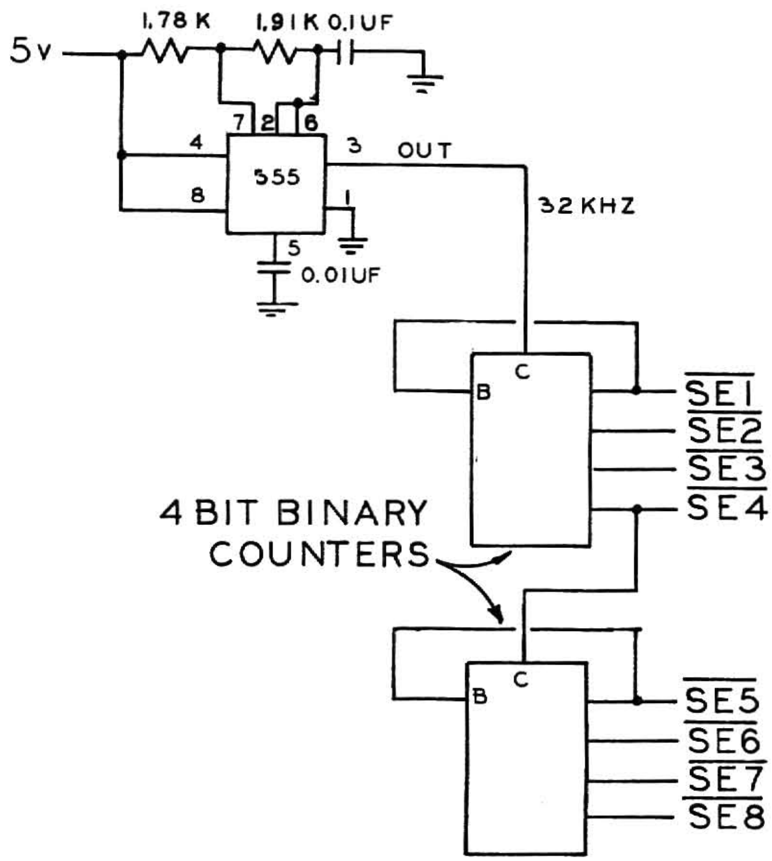



FIGURE 2.15

THE SELF TEST CIRCUIT USED TO SIMULATE THE AZIMUTH the azimuth. The voltage levels of the shaft encoder output are converted to TTL voltage levels when the shaft encoder provides the azimuth signal. The shaft encoder low order bit has a frequency of 32 Hz. and the self test circuit provides a low order bit with a frequency of 31.2 Hz.

Figure 2.16 shows the circuit that simulates the output and synchronization signals of the Integration and Quantization circuit. The synchronization signals $\overline{\text{ONE}}$, $\overline{\text{ZERO}}$, DUMP and READY are described in the section "The Synchronization Signals". The circuit shown in Figure 2.16 is implemented on board IAQ-1. The clock frequency in the Integration and Quantization circuit that is used to select the range bins is 186.2 KHz. The self test circuit uses a 181 KHz. clock to select the range bins. This means that the self test circuit outputs a range bin once every 5.52 microseconds instead of every 5.37 microseconds as provided by the Integration and Quantization circuit.

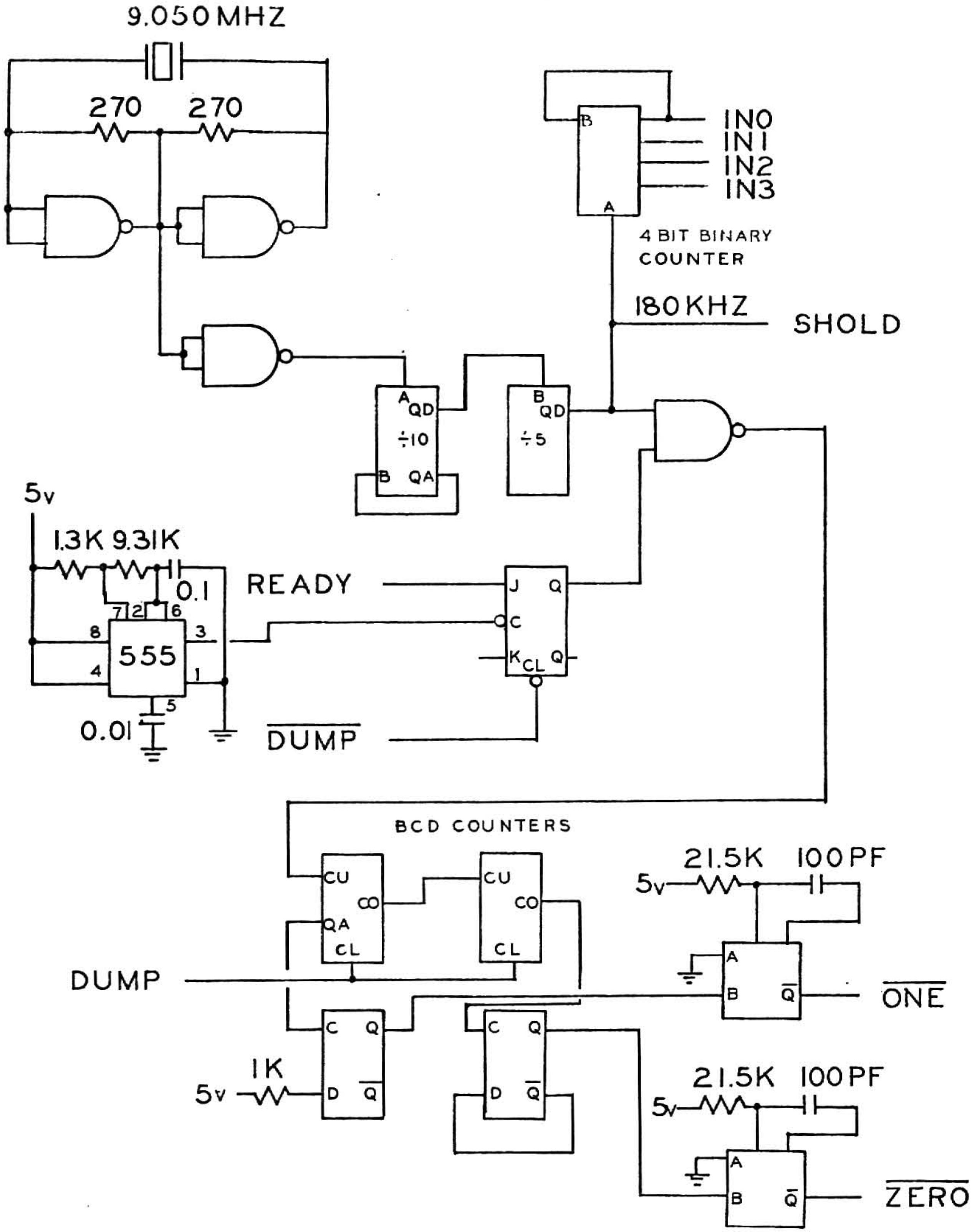



FIGURE 2.15
GENERATION OF THE COMPUTER INTERFACE TO INTEGRATION AND QUANTIZATION CIRCUIT SYNCHRONIZATION SIGNALS

# Chapter 3

Since a computer is used as an intermediate step in moving the data from the interface to the disk there must be a program to control movement of the data from the interface to the computer's memory and then out to the disk. At the present time the Department of Electrical Engineering has two Caelus CD-22 disk drives and a Disk Management Operating System supporting these disk drives. Figure 3.1 is a high level flowchart of the program that moves data from the interface to the disk. With the memory size available on the Nova 1200 it is possible to store in memory all of the range bin words from one scan, and then once the scan in completed, to move the data to the disk. This program is available on the computer system's disk under the name RDCP.BI, Radar Data Collection Program (.BI stands for executable "binary" ). The function of each block of the flowchart shown in Figure 3.1 is discussed in this chapter. A source listing of RDCP is provided in Appendix C.

The term "data acquisition record" will be defined as the 64 azimuth increment counts and their 50 associated range bin intensity words.

## THE INITIALIZATION BLOCK

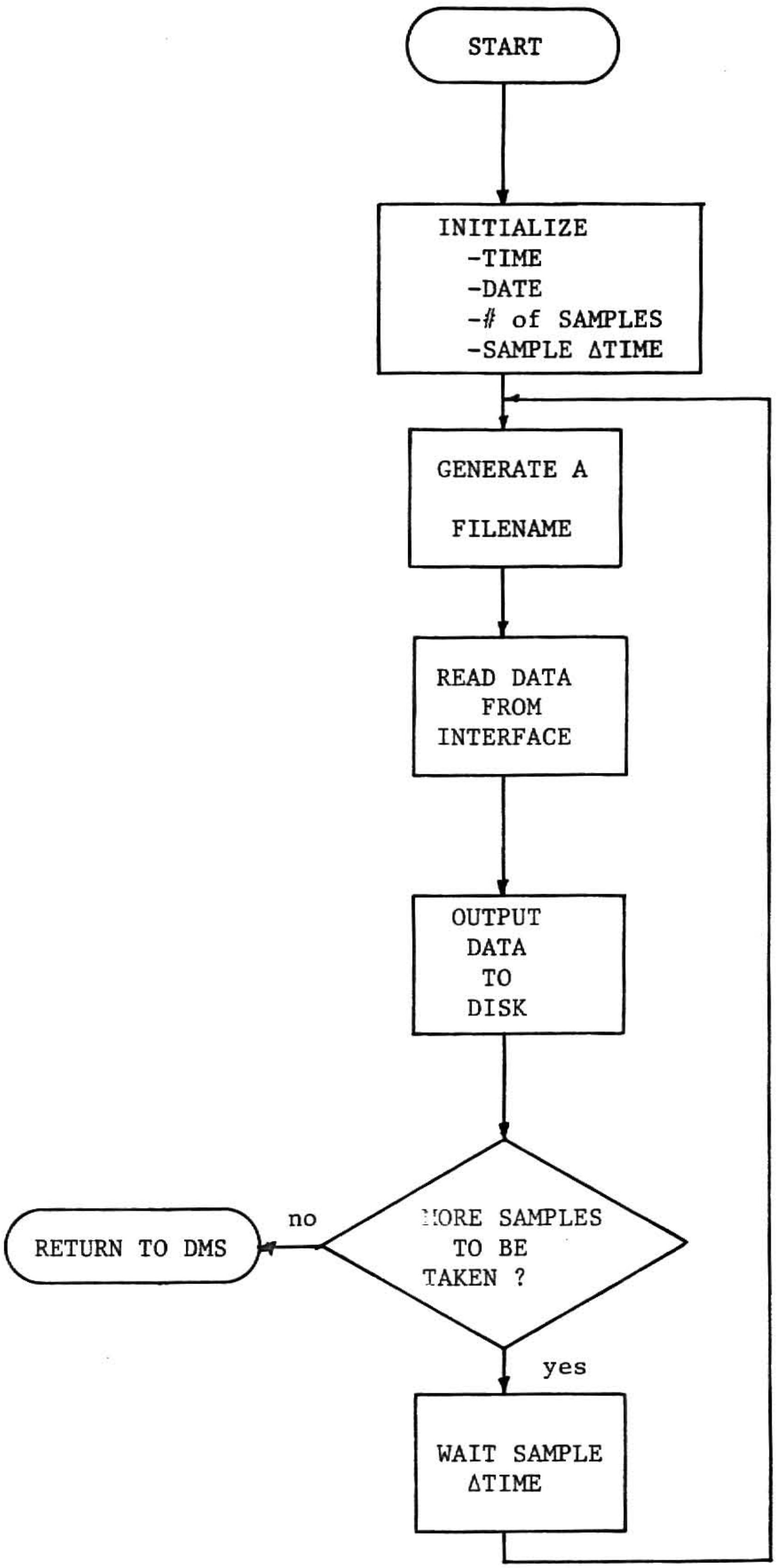



FIGURE 3.1

HIGH LEVEL FLOWCHART OF RADAR INTERFACE CONTROL PROGRAM

Prior to obtaining data from the interface the user is asked to initialize the program parameters. The user is asked the time of day and date so that each data acquisition record can be labeled for easier bookeeping.The user can also specify how many data acquisition sequences he wishes to record and the interval between them. Figure 3.2 shows the flowchart for the initialization block of RDCP.

The program will type out the statement:

TIME OF DAY (H/M/S) :

The user must respond with the time of day in a 24 hour format. Each field (hours, minutes and seconds) must be a two digit integer value. Any invalid responses will result in the question being repeated. A typical response would be:

TIME OF DAY (H/M/S) : 09/30/25

where the "/" symbols will be provided by the program.

After obtaining the time of day the program will type out the statement:

DATE (MONTH/DAY/YEAR) :

The user must respond with the date. The year field consists of only the last two digits of the current calender year, e.g., input 75 for 1975. The month and day responses must be two digit integer values as was required for the time fields. The response for the date February 1, 1975 would be:

DATE (MONTH/DAY/YEAR) : 02/01/75

where the "/" symbols will be provided by the program. Any incorrect response results in the repetition of the

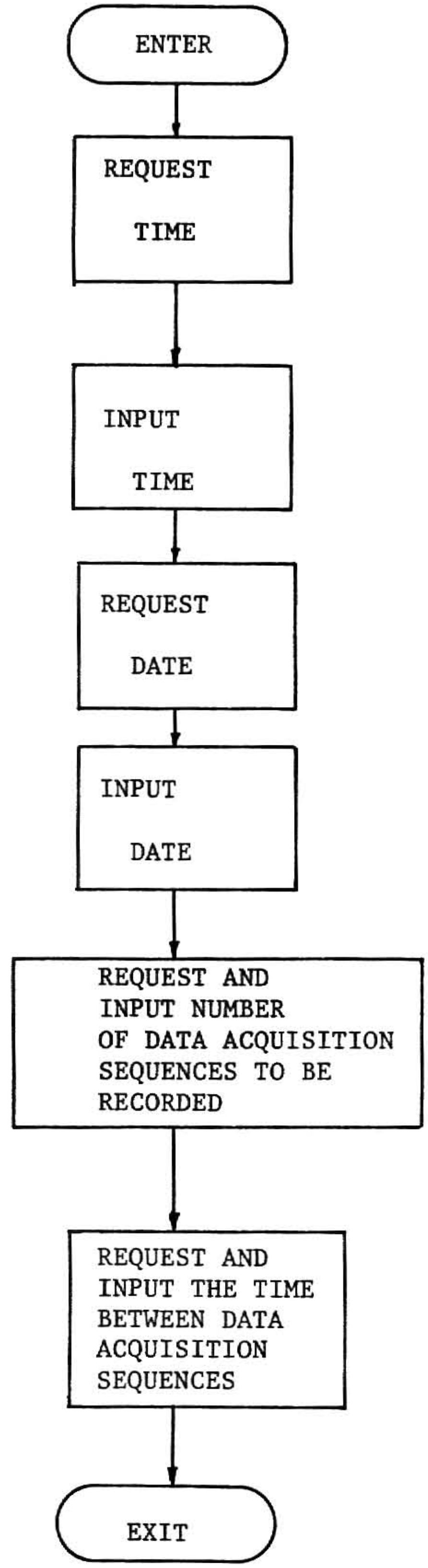



FIGURE 3.2

THE INITIALIZATION BLOCK FLOWCHART

question.

The time of day and date are stored in the first six words of the data acquisition record. The order of storage is:

| Word | Contents |
|---|---|
| 0 | Hours |
| 1 | Minutes |
| 2 | Seconds |
| 3 | Month |
| 4 | Day |
| 5 | Year |

Each word will contain the binary equivalent of the BCD integers from that field. For example the year 1975 would be entered in as 75 and appear in word 5 in binary as : 0000000001001011.

The program is initialized to obtain and store up to 100 data acquisition records. The user responds to the question:

NUMBER OF RECORDS DESIRED :

with the three digit integer number of data acquisition sequences to be recorded. Entry of a non-numeric character or a value greater than 100 is invalid and the question is repeated. The user signals the program that the entire number has been entered into the computer by pressing the RETURN key. The time interval between successive data acquisition sequences can also be specified. The time interval is set by by responding to the question:

DATA ACQUISITION SEQUENCE INTERVAL

TIME (H/M/S) :

with the desired time in a 24 hour format as required for the time of day entry.

Once the program is initialized, space must be provided on the disk for the data from the acquisition records. The next block will create a unique file name for each data acquisition record.

## GENERATE A FILE NAME

To reduce the size of the control program a specific family of file names are used and generated in this block of the control program. The data is stored on the disk and is accessed under the file name RDmn.DA where mn=00 ... 99. There is a unique file name associated with the data from each data acquisition sequence. The family of file names used will allow a maximum of 100 data acquisition sequences to be stored on the disk.

This program is written such that it must be executed under the Disk Management Operating System (DMS). Associated with each file on the disk is a table that is placed in the disk file directory. This table is called the Work File Parameter Table or WFPT. The WFPT contains the file name, the starting sector address, the location in memory where the file is placed, and the file identification. The format of the Work File Parameter Table is shown in Table 3.1.

Table 3.1 Work File Parameter Table Organization

| Word | Contents |
|---|---|
| 0-2 | File name, up to six seven bit ASCII characters packed right to left. For an odd number of characters the last character is left justified. |
| 3 | Starting Sector Address |
| 4 | Lower Core Limit if the file is binary. Sector size if the file is source. |
| 5 | Upper Core Limit if the file is binary. Zero if the file is source or relocatable binary. |
| 6 | Start Address if the file is binary. One if the file is relocatable binary. As initialized if the file is source. |
| 7 | File ID, two ASCII characters packed right to left. |

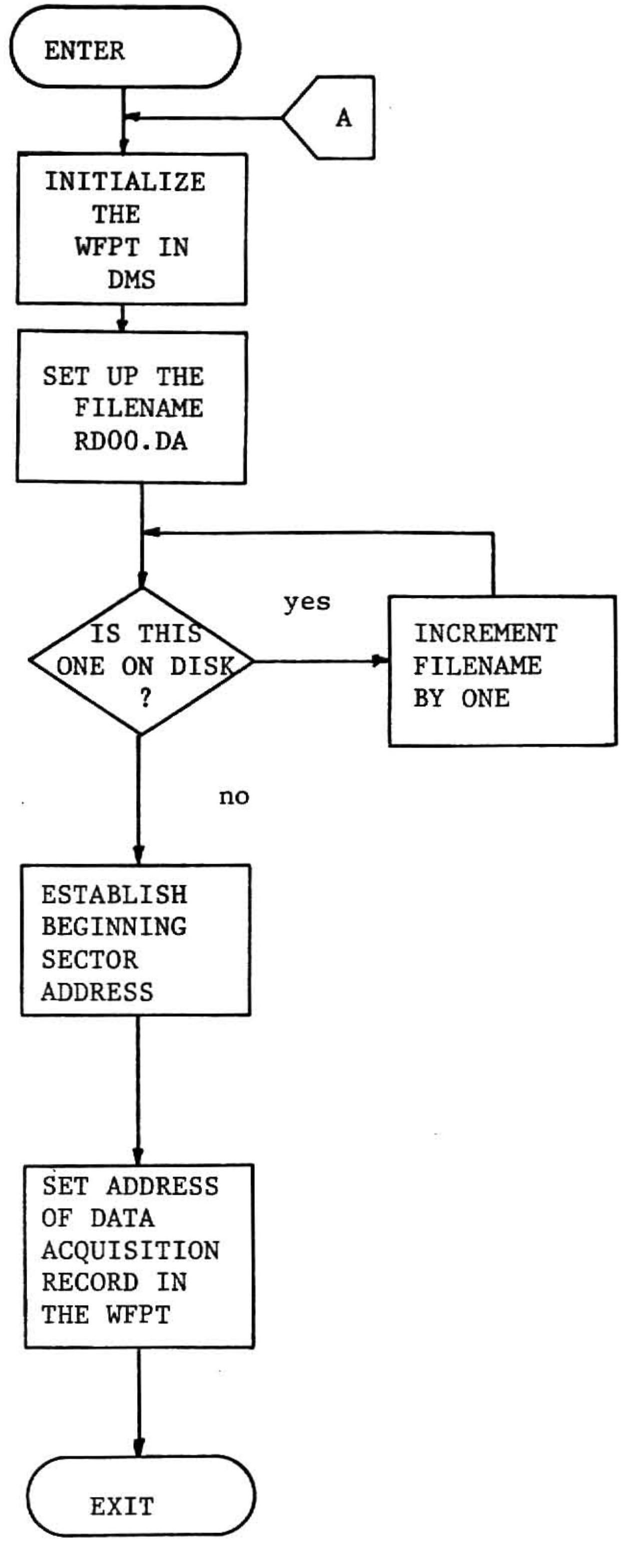



FIGURE 3.3

FLOWCHART FOR DETERMINING FILENAME

Table 3.2 relates the WFPT word to its contents. Using this format the user can read any data acquisition record one disk sector at a time to any desired buffer area. A detailed description of the use of DMS and the resident routines for reading and writing data to the disk can be found in the Caelus Memories, Inc, Disk Management System Manual (1).

## THE DATA ACQUISITION SEQUENCE

After the disk file name is created and the program parameters are initialized the data is moved from the interface to the computer's memory. Execution time in this segment is a critical factor. The interface supplies four data values in one computer word every 21.48 microseconds, thus the program must read the data from the interface, store it in the computer's memory, and be ready to read the next value in this 21.48 microsecond time frame. Figure 3.4 shows the program segment and instruction times for moving the data from the interface to the computer's memory.

Once all of the data is stored in the computer's memory it is then placed on the disk. The next block is the program segment required to accomplish this.

## WRITING DATA TO THE DISK

Figure 3.4 Instructions and Times

For the Transfer Segment of the

Control Program RDCP

```
        LDA      2,COUNT
        COM      2,2
START:  IORST
        NIOS     66
        SKPDN    66        2.55
        JMP      .-1       1.35
        DIAS     0,66      2.55
        STA      0,@TABLE  2.55 + 1.20 + 0.60
```

total time to read and store
azimuth increment count : 10.80 microseconds

```
        NIOS     67        3.55
        SKPDN    67        2.55
        JMP      .-1       1.35
        DIAS     0,67      2.55
        STA      0,@TABLE  2.55 + 1.20 + 0.60
        INC      2,2,SZR   1.35
        JMP      .-6       1.35
```

total time to read and store
range bin intensity word : 17.05 microseconds

Note: all times are given in microseconds

Table 3.2 Contents of the Work File Parameter Table

| Word | Contents |
| --- | --- |
| 0 | Characters 'DR' |
| 1 | Character digits 'nm' |
| 2 | A 1 in bit 0, all other bits 0 |
| 3 | Beginning sector address |
| 4 | Number of sectors used |
| 5 | A 1 in bit 0, all other bits 0 |
| 6 | A 1 in bit 0, all other bits 0 |
| 7 | Characters 'AD' |

The WFPT was set up prior to executing the data acquisition sequence and the data is now in a memory buffer. Table 3.3 shows the organization of the buffer. The first 12 bytes contain the date and time information. Bytes 13 to 6540 contain the azimuth increment counts and range bin intensities for one 360 degree sweep of the radar antenna. Figure 3.5 shows the word structure for the intensity values.

The data is placed on the disk using the sequence of steps shown in the flowchart of Figure 3.6.

The CATAL and DSKW are resident DMS routines and their use is explained in Appendix D of (1). Once the data is on the disk it is left to the user to provide programs that will process it furthur.

## PREPARATION FOR THE NEXT DATA ACQUISITION SEQUENCE

The decision block of the flowchart in Figure 3.1 determines if more data acquisition sequences are to be recorded. The Wait block provides the required interval between the data acquisition sequences. The expanded flowchart for these two blocks is shown in Figure 3.7.

During initialization the number of samples to be taken was entered by the user. The number of samples requested by the user is decremented by one each time a data acquisition sequence has been performed. After all data acquisition sequences have been performed control is returned to the

Table 3.3 ORGANIZATION OF BYTES FOR ONE DATA ACQUISITION SEQUENCE

| AZIMUTH | AZIMUTH BYTES | INTENSITY BYTES |
|---|---|---|
| 5.625 | 12 - 13 | 14 - 113 |
| 11.25 | 114 - 115 | 116 - 215 |
| 16.875 | 216 - 217 | 218 - 317 |
| 22.5 | 318 - 319 | 320 - 419 |
| 23.125 | 420 - 421 | 422 - 521 |
| 33.75 | 522 - 523 | 524 - 623 |
| 39.375 | 624 - 625 | 626 - 725 |
| 45 | 726 - 727 | 728 - 827 |
| 50.625 | 828 - 829 | 830 - 929 |
| 56.25 | 930 - 931 | 932 - 1031 |
| 61.875 | 1032 - 1033 | 1034 - 1133 |
| 67.5 | 1134 - 1135 | 1136 - 1235 |
| 73.125 | 1236 - 1237 | 1238 - 1337 |
| 78.75 | 1338 - 1339 | 1340 - 1439 |
| 84.375 | 1440 - 1441 | 1442 - 1541 |
| 90 | 1542 - 1543 | 1544 - 1643 |
| 95.625 | 1644 - 1645 | 1646 - 1745 |
| 101.25 | 1746 - 1747 | 1748 - 1347 |
| 106.875 | 1848 - 1349 | 1850 - 1949 |
| 112.5 | 1950 - 1951 | 1952 - 2051 |
| 118.125 | 2052 - 2053 | 2054 - 2153 |
| 123.75 | 2154 - 2155 | 2156 - 2255 |
| 129.375 | 2256 - 2257 | 2258 - 2357 |
| 135 | 2358 - 2359 | 2360 - 2459 |
| 140.625 | 2460 - 2461 | 2462 - 2561 |
| 146.25 | 2562 - 2563 | 2564 - 2663 |
| 151.875 | 2664 - 2665 | 2666 - 2765 |
| 157.5 | 2766 - 2767 | 2768 - 2367 |
| 163.125 | 2868 - 2869 | 2870 - 2969 |
| 168.75 | 2970 - 2971 | 2972 - 3071 |
| 174.375 | 3072 - 3073 | 3074 - 3173 |
| 180 | 3174 - 3175 | 3176 - 3275 |
| 185.625 | 3276 - 3277 | 3278 - 3377 |
| 191.25 | 3373 - 3379 | 3380 - 3479 |
| 196.875 | 3480 - 3481 | 3482 - 3581 |
| 202.5 | 3582 - 3583 | 3584 - 3683 |
| 208.125 | 3684 - 3685 | 3686 - 3785 |
| 213.75 | 3786 - 3787 | 3788 - 3887 |
| 219.375 | 3888 - 3839 | 3890 - 3989 |
| 225 | 3990 - 3991 | 3992 - 4091 |
| 230.625 | 4092 - 4093 | 4094 - 4193 |
| 236.25 | 4194 - 4195 | 4196 - 4295 |
| 241.875 | 4296 - 4297 | 4293 - 4397 |
| 247.5 | 4393 - 4399 | 4400 - 4499 |
| 253.125 | 4500 - 4501 | 4502 - 4601 |
| 258.75 | 4602 - 4603 | 4604 - 4703 |
| 264.375 | 4704 - 4705 | 4706 - 4805 |
| 270 | 4806 - 4807 | 4808 - 4907 |
| 275.625 | 4908 - 4909 | 4910 - 5009 |
| 281.25 | 5010 - 5011 | 5012 - 5111 |
| 286.875 | 5112 - 5113 | 5114 - 5213 |
| 292.5 | 5214 - 5215 | 5216 - 5315 |
| 298.125 | 5316 - 5317 | 5318 - 5417 |
| 303.75 | 5418 - 5419 | 5420 - 5519 |
| 309.375 | 5520 - 5521 | 5522 - 5621 |
| 315 | 5622 - 5623 | 5624 - 5723 |
| 320.625 | 5724 - 5725 | 5726 - 5825 |
| 326.25 | 5826 - 5827 | 5828 - 5927 |
| 331.875 | 5928 - 5929 | 5930 - 6029 |
| 337.5 | 6030 - 6031 | 6032 - 6131 |
| 343.125 | 6132 - 6133 | 6134 - 6233 |
| 348.75 | 6234 - 6235 | 6236 - 6335 |
| 354.375 | 6336 - 6337 | 6338 - 6437 |
| 360 | 6438 - 6439 | 6440 - 6539 |

| | | | |
|---|---|---|---|
| INT0 INT1 INT2 INT3 | INT0 INT1 INT2 INT3 | INT0 INT1 INT2 INT3 | INT0 INT1 INT2 INT3 |
| 0 3 | 4 7 | 8 11 | 12 15 |

FIGURE 3.5

STRUCTURE OF THE RANGE BIN INTENSITY WORDS

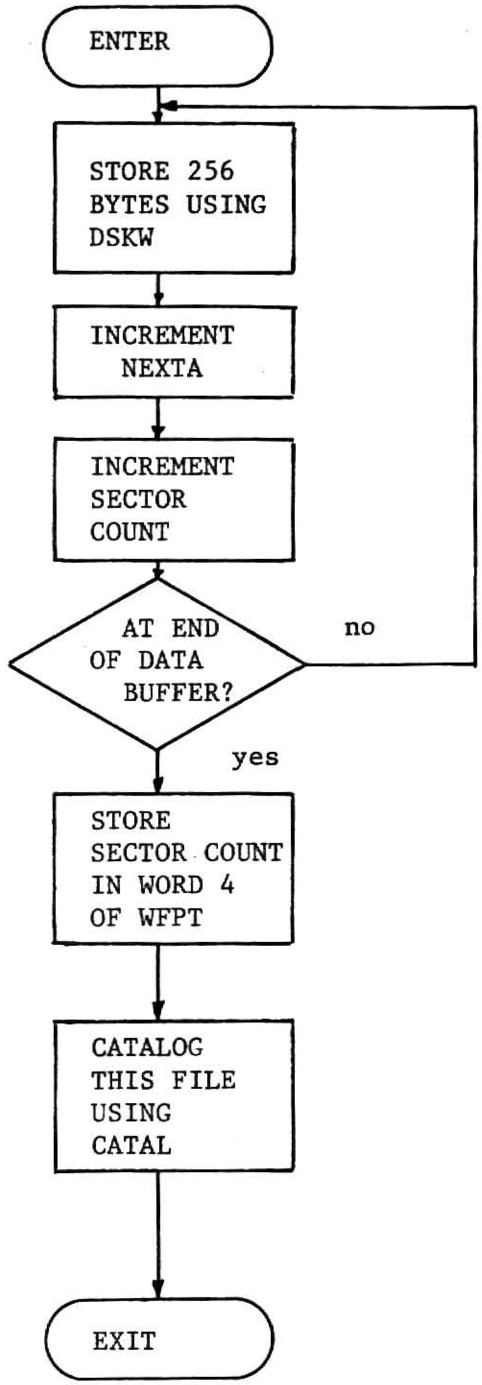



FIGURE 3.6

FLOWCHART FOR MOVING DATA FROM COMPUTER TO DISK

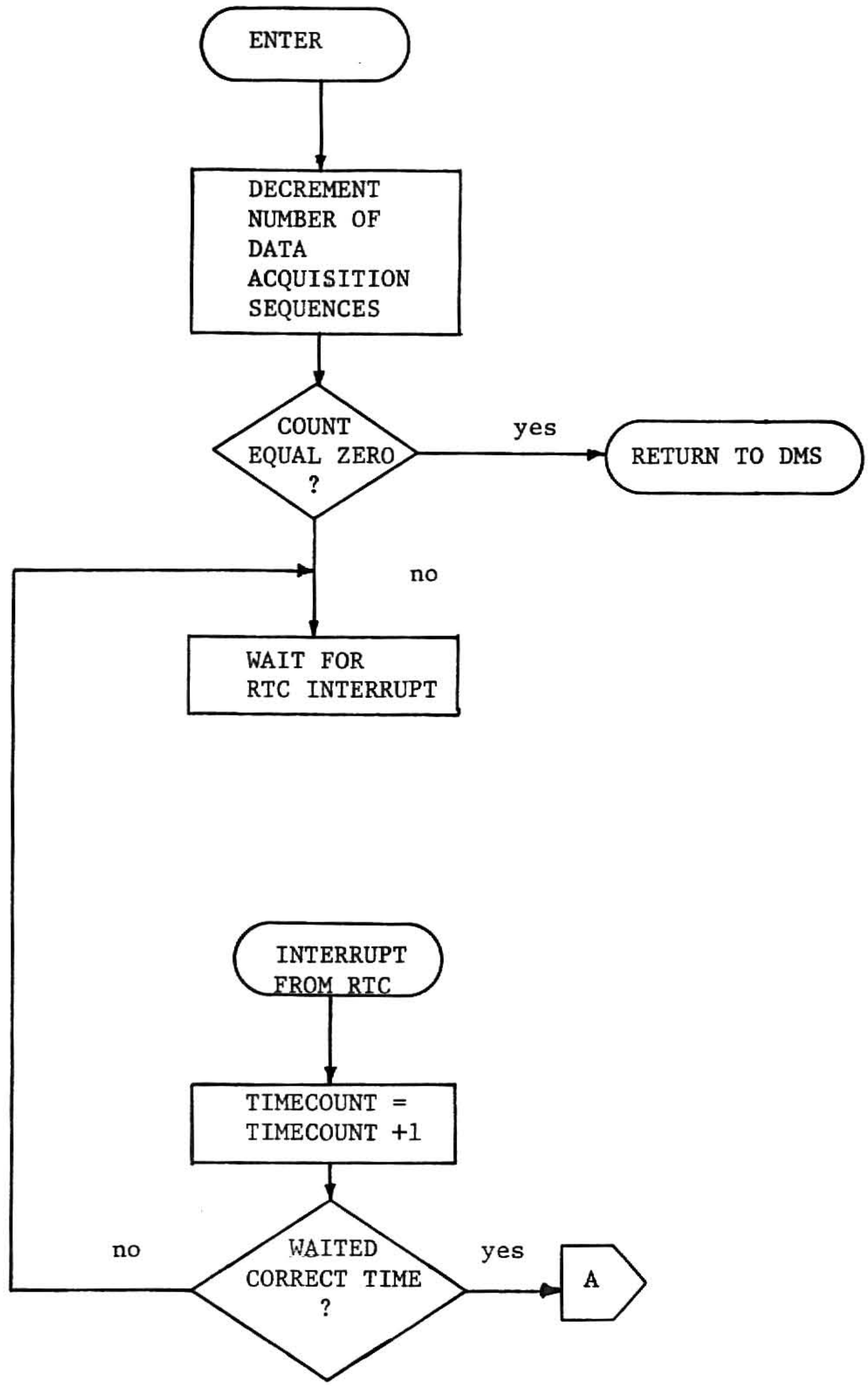



FIGURE 3.7

FLOWCHART FOR DETERMINING NEXT SEQUENCE

Disk Management Operating System.

The Wait block is a loop that provides the time interval specified between successive data acquisition sequences. This loop counts the interrupts from the Real Time Clock (RTC) until the number of interrupts matches the MODULO 60 value of the hour, minutes, and seconds that were entered during initialization.

The time of the next data acquisition sequence is determined by adding the time to execute one data acquisition sequence and the interval time between data acquisition sequences to the time of the previous data acquisition sequence.

# Chapter 4

The data collected in one data acquisition sequence can be converted to precipitation rate and cumulative precipitation information. The cumulative precipitation information can be used in flash flood forcasting. The precipitation data collected using the system should greatly improve the quality of precipitation measurements for this area. These precipitation measurements could be input to a real time watershed modeling program. Analysis of the precipitation data over a period of time should yield better ground water predictions.

The system can also be used to record the movement of particularly large or severe storms. The computer graphics terminal can be used to display the recorded data, thus allowing analysis of such factors as the direction and velocity of movement of a storm. Work in the area of storm identification through pattern recognition is now possible with the data collected using this system.

If the beamwidth of the antenna is changed to a value less than 2.8125 degrees only minor modifications are necessary to allow collection of the range bin intensities every 2.8125 degrees. The Azimuth Interval Counter clock input, AZCCP, must be changed so that the counter will count in 2.8125 degree increments. This may be accomplished by one of two methods.

The first method is to add a ratio gear to the shaft encoder so that the encoder will make a full count in 16 revolutions of the antenna. This modification will not require any circuit modifications as bit 1 of the shaft encoder would represent 1.40625 degree increments of the antenna. Bit 1 is multipled by two to obtain the azimuth increment over which integration is performed.

The second method is to change the clock pulse AZCCP so that the falling and rising edges of AZCCP clock the Azimuth Increment Counter. This circuit modification is shown in Figure 4.1 and would require production of a new circuit board (Board B).

No program modifications are necessary for either of the above hardware modifications. The time between azimuth increments would still be much greater than the time to transfer all of the range bin intensities. The amount of memory used to store the range bin words is increased by a factor of 2. However the program maintains pointers to the beginning and end of the buffer containing the range bin words so all of the data would still be transfered to the disk.

Using the self test circuit, program runs indicate the control program and the interface are in working order.

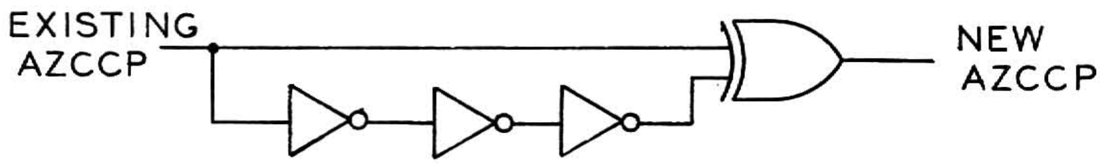



FIGURE 4.1

MODIFICATIONS TO AZCCP

# BIBLIOGRAPHY

1. ——— Disk Management System (DMS) Manual, California, Caelus Memories Inc.

2. Andreasen, Carl C., A Manual for Programming the Data General Nova Minicomputers at the Assembly Language Level Under Three Operating Systems: Stand Alone Operating System, Disk Management Operating System, and Real Time Disk Operating System, (Unpublished Graduate problems paper, Kansas State University, 1975)

3. English, William, How to Use the Nova Computers, Massachusetts, Data General Corporation, 1971.

4. Hummels, Donald R., A Digitized Radar for Precipitation Measurements and Applications to Hydrology, Project Completion Report, Kansas State University, 1975.

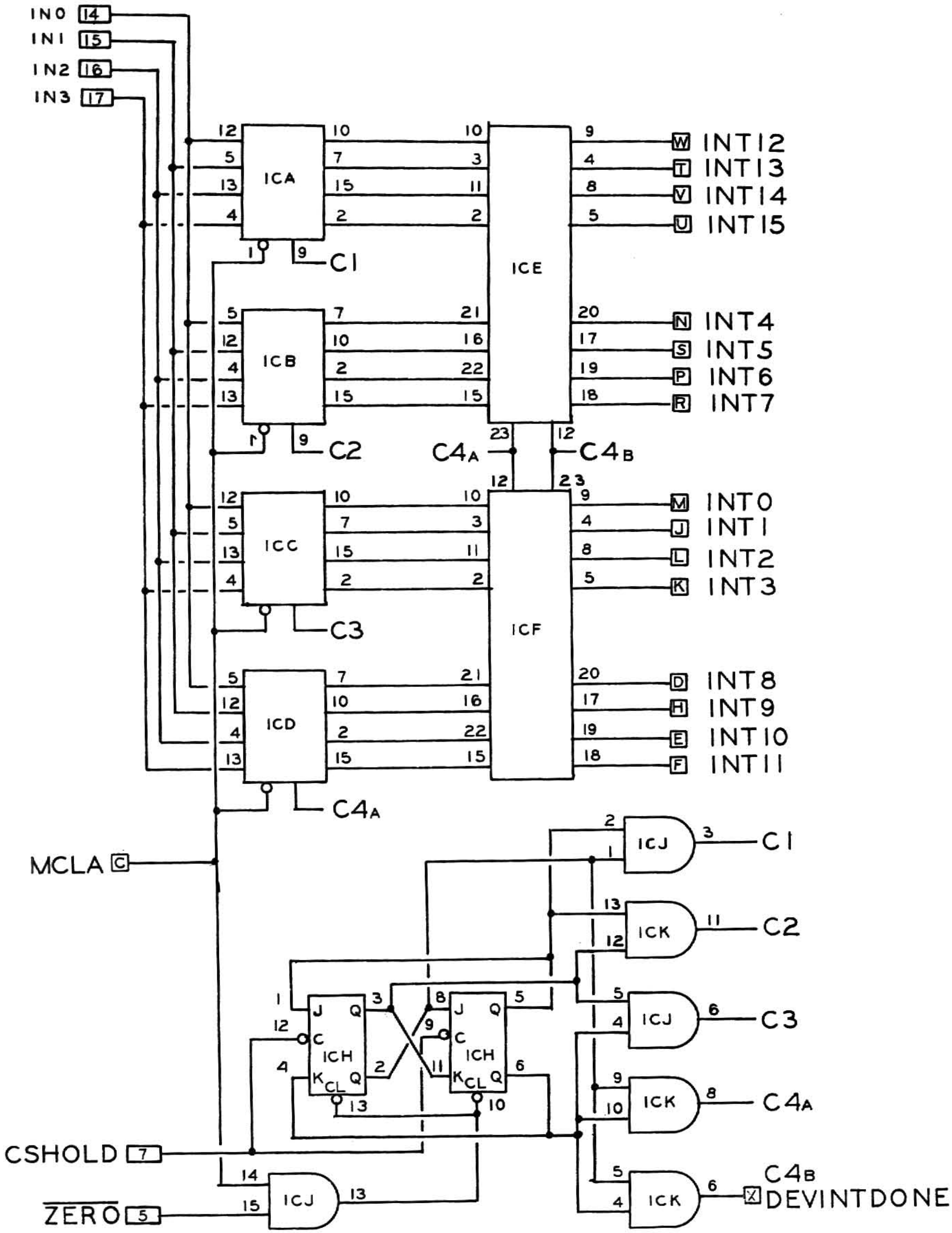
5. Jesse, Richard H., The Design and Implementation of an Interface Between the Nova 1200 and Three Peripherals: An Analog-to-Digital Converter, A Digital-to-Analog Converter and an Incrementtal Tape Transport, (Unpublished Master's Degree Thesis, Kansas State University, 1974)

# APPENDIX A

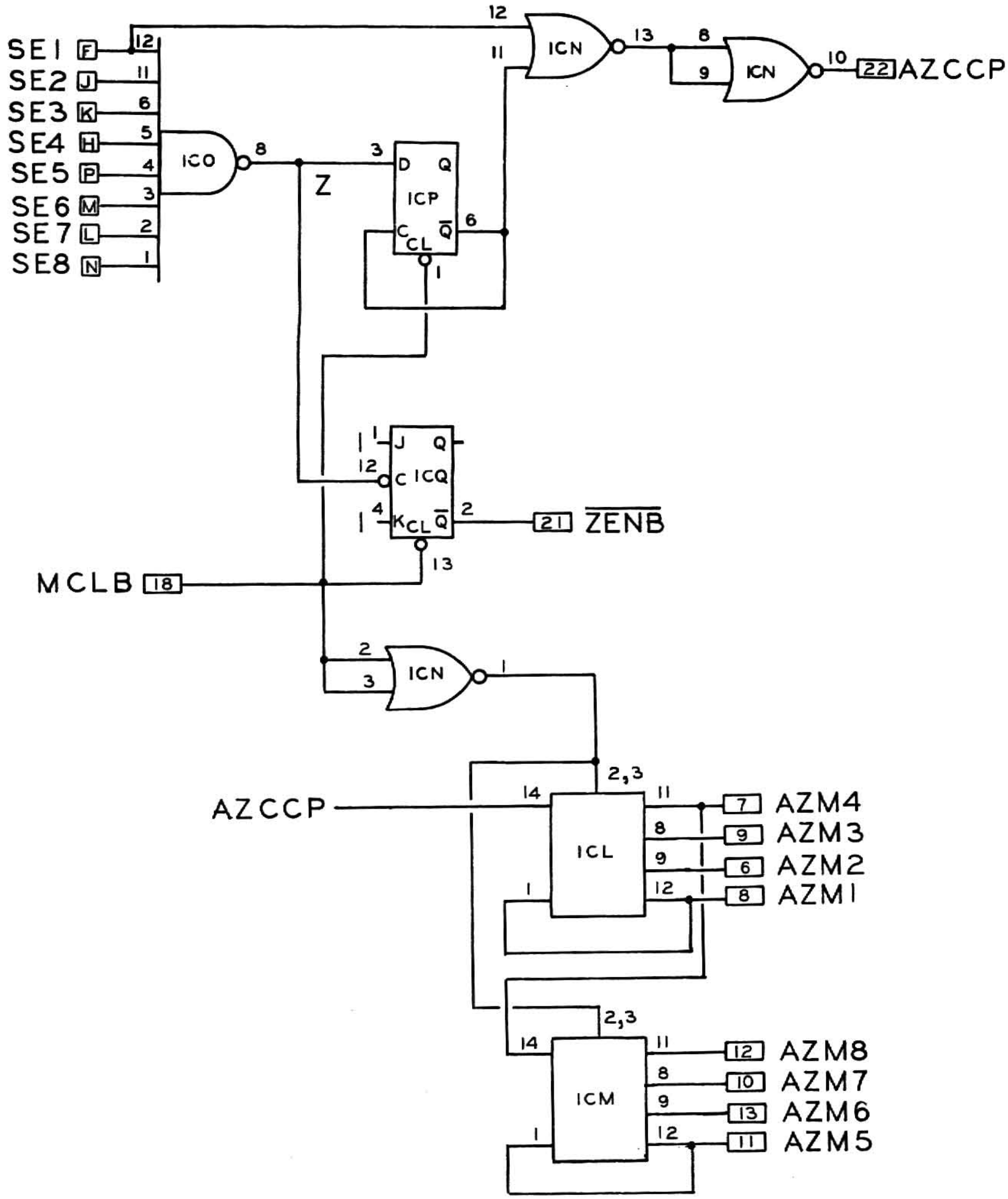
## CIRCUIT DIAGRAMS

Legend:

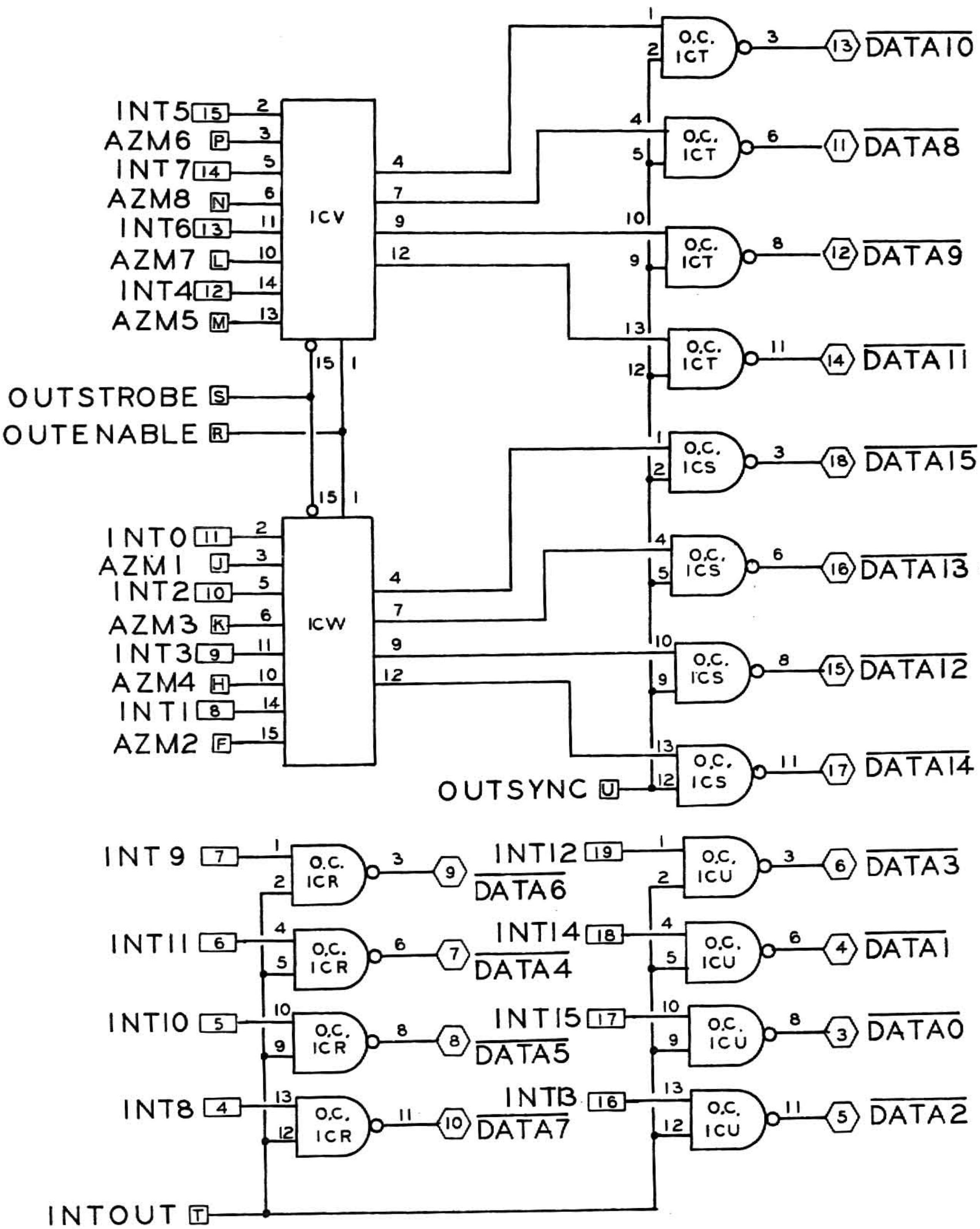
| | |
|---|---|
| 7404 | Integrated Circuit Device Type |
| [5] , [A] | Backplane Pin |
| ⟨14⟩ | Computer Bus Pin |
| Fn | Feedthrough Number n |
| Jn | Jumper Number n |
| | Capacitor |
| | Resistor |


IN0 14
IN1 15
IN2 16
IN3 17
ICA
ICB
ICC
ICD
ICE
ICF
C1
C2
C3
C4A
C4B
INT12
INT13
INT14
INT15
INT4
INT5
INT6
INT7
INT0
INT1
INT2
INT3
INT8
INT9
INT10
INT11
MCLA
ICJ
ICK
ICH
CSHOLD 7
ZERO 5
DEVINTDONE

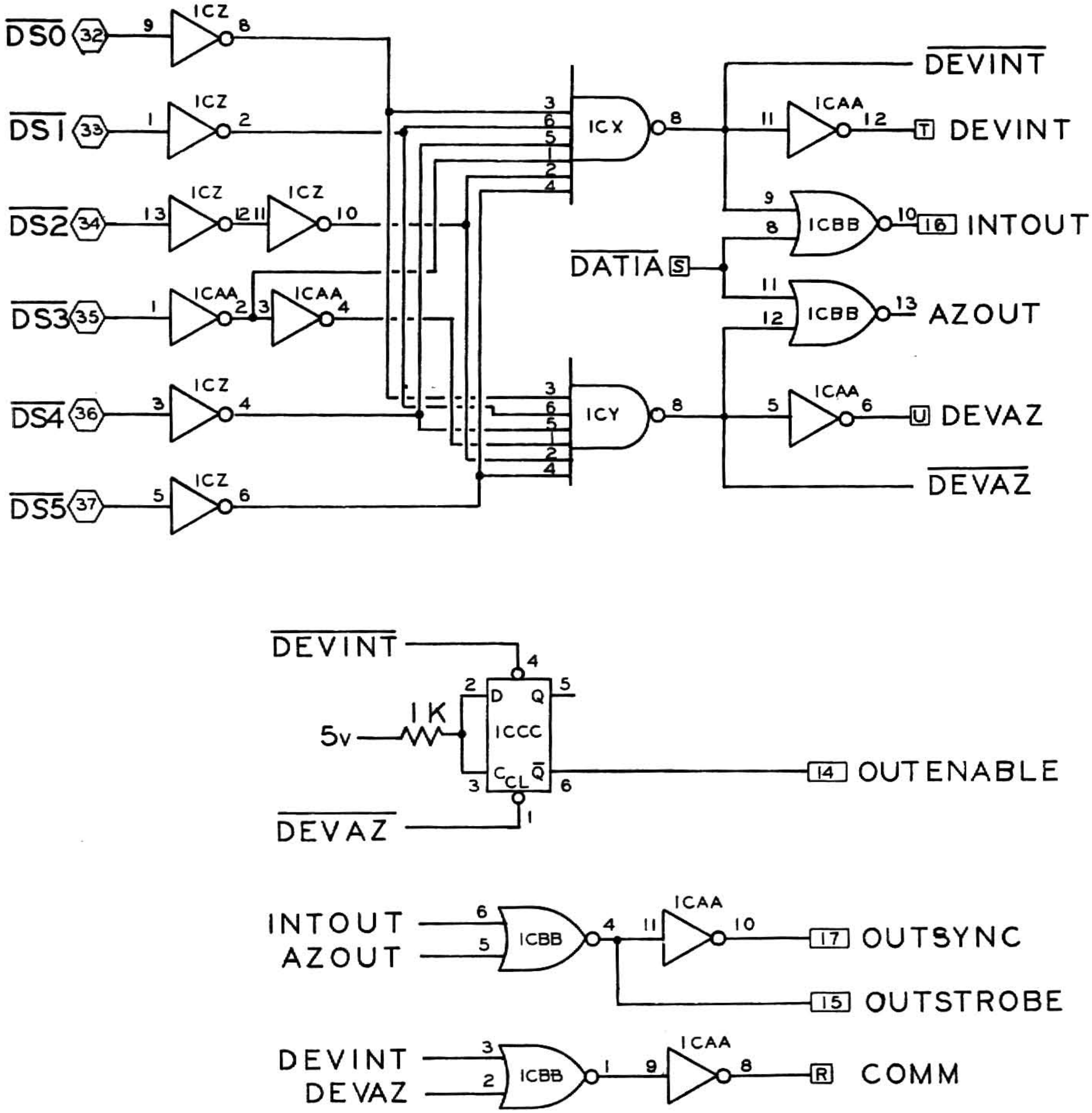
BOARD A


SE1 F
SE2 J
SE3 K
SE4 H
SE5 P
SE6 M
SE7 L
SE8 N
ICO
Z
ICP
ICN
22 AZCCP
ICQ
21 $\overline{ZENB}$
MCLB 18
AZCCP
ICL
7 AZM4
9 AZM3
6 AZM2
8 AZM1
ICM
12 AZM8
10 AZM7
13 AZM6
11 AZM5

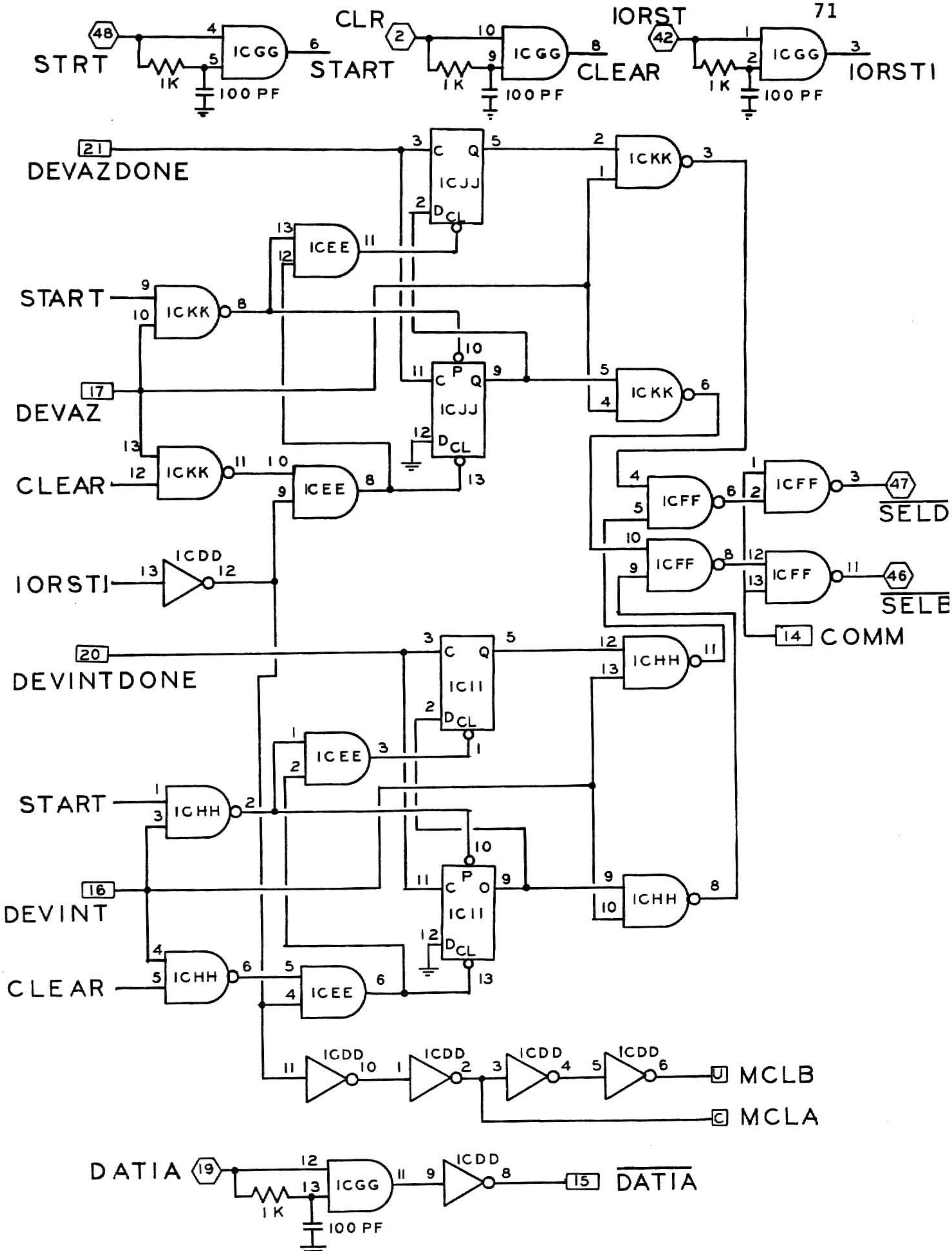
BOARD B


INT5 15
AZM6 P
INT7 14
AZM8 N
INT6 13
AZM7 L
INT4 12
AZM5 M
ICV
OUTSTROBE S
OUTENABLE R
INT0 11
AZM1 J
INT2 10
AZM3 K
INT3 9
AZM4 H
INT1 8
AZM2 F
ICW
O.C. ICT
O.C. ICS
13 DATA10
11 DATA8
12 DATA9
14 DATA11
18 DATA15
16 DATA13
15 DATA12
17 DATA14
OUTSYNC U
INT9 7
INT11 6
INT10 5
INT8 4
O.C. ICR
9 DATA6
7 DATA4
8 DATA5
10 DATA7
INT12 19
INT14 18
INT15 17
INT13 16
O.C. ICU
6 DATA3
4 DATA1
3 DATA0
5 DATA2
INTOUT T

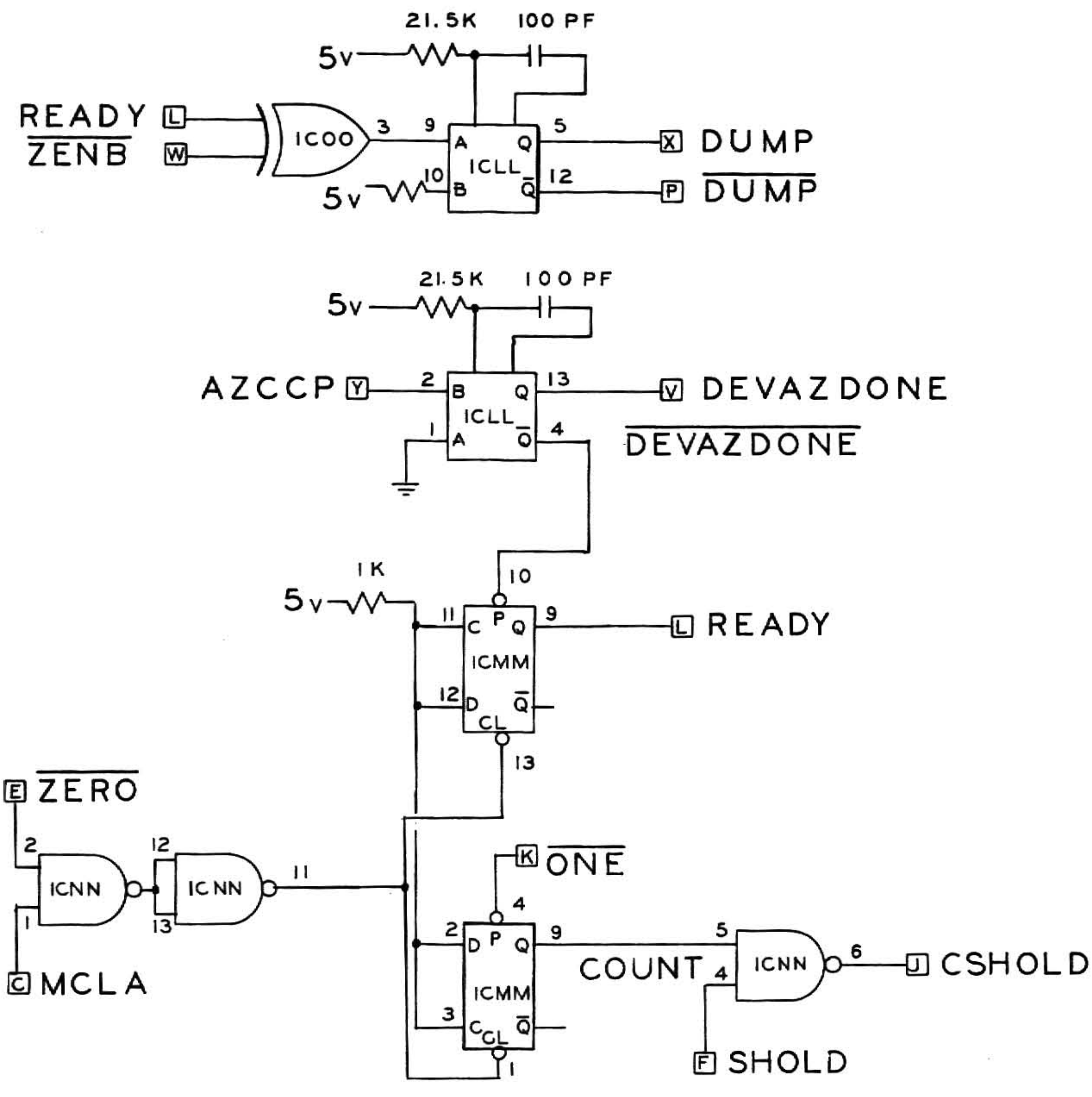
BOARD C


DS0 32
DS1 33
DS2 34
DS3 35
DS4 36
DS5 37
ICZ
ICAA
ICX
ICY
ICBB
DEVINT
T DEVINT
16 INTOUT
DATIA S
AZOUT
U DEVAZ
DEVAZ
DEVINT
5v
1K
D Q
ICCC
$C_{CL}$ $\bar{Q}$
14 OUTENABLE
DEVAZ
INTOUT
AZOUT
ICBB
ICAA
17 OUTSYNC
15 OUTSTROBE
DEVINT
DEVAZ
ICBB
ICAA
R COMM

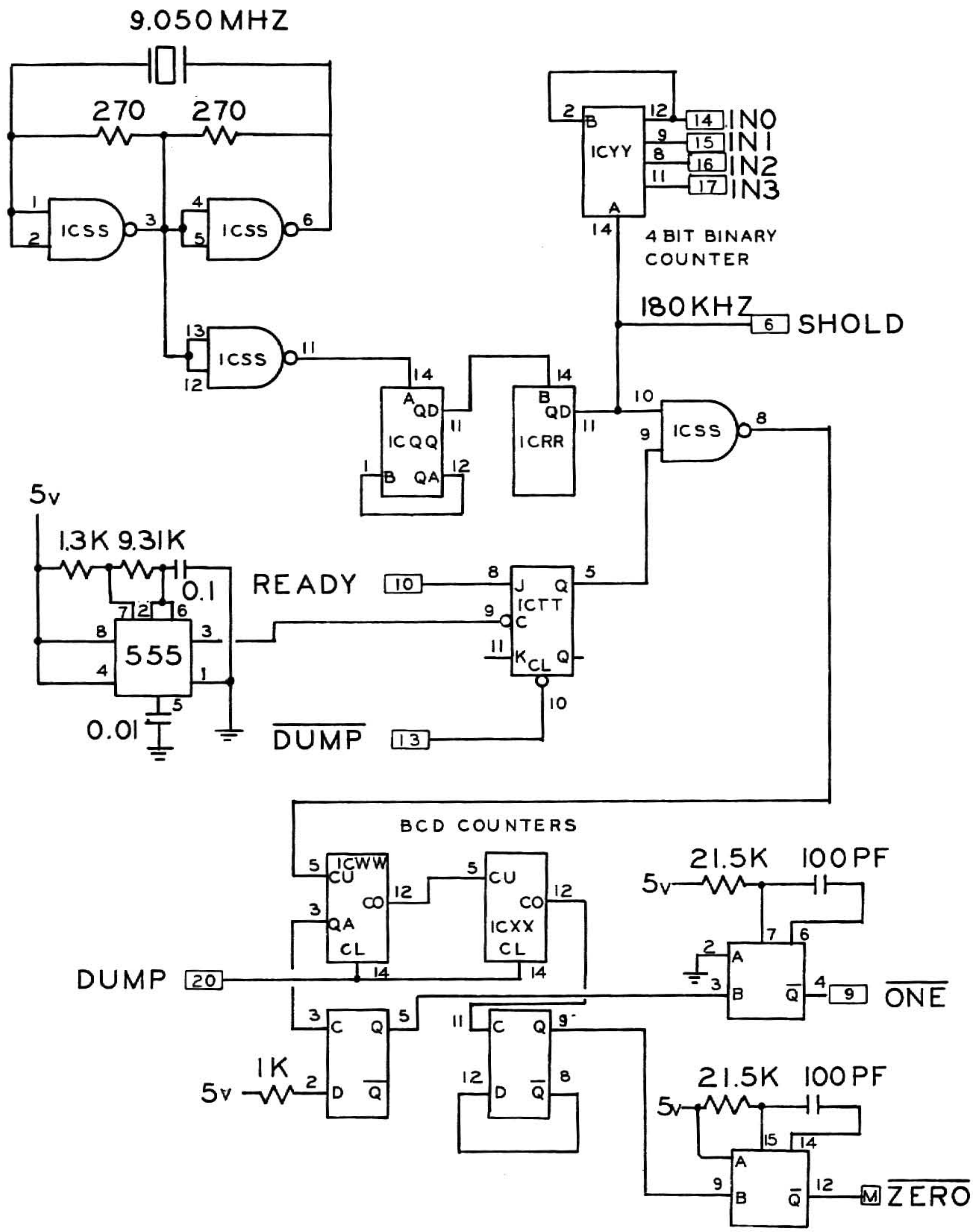
BOARD D


STRT
48
1K
100 PF
ICGG
START
CLR
2
CLEAR
IORST
42
IORSTI
21
DEVAZDONE
ICJJ
ICKK
ICEE
START
17
DEVAZ
CLEAR
ICDD
IORSTI
ICFF
47
SELD
46
SELE
14
COMM
20
DEVINTDONE
IC11
ICHH
16
DEVINT
CLEAR
U
MCLB
C
MCLA
DATIA
19
15
DATIA

BOARD E

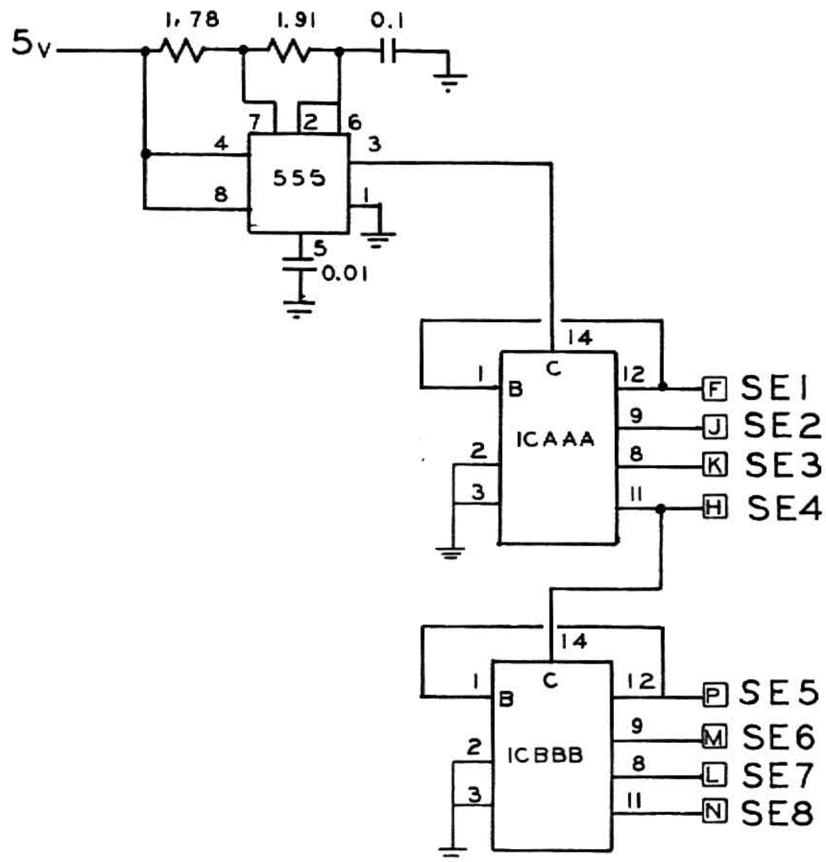

21.5K
100 PF
5v
READY
ZENB
ICOO
ICLL
DUMP
DUMP
5v
21.5K
100 PF
5v
AZCCP
ICLL
DEVAZDONE
DEVAZDONE
1K
5v
ICMM
READY
ZERO
ICNN
ICNN
ONE
ICMM
COUNT
ICNN
CSHOLD
MCLA
SHOLD

BOARD F

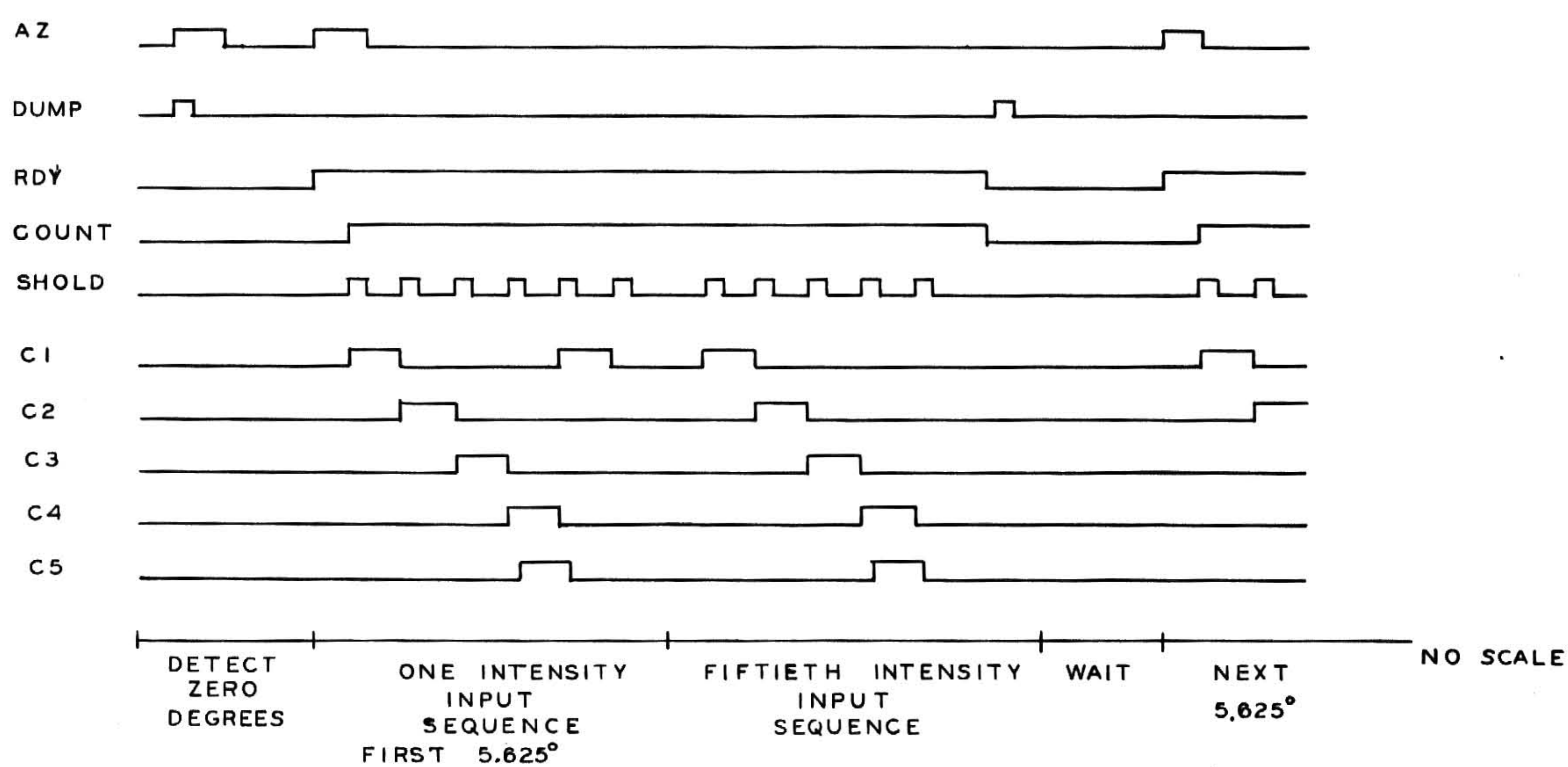


9.050 MHZ
270
270
ICSS
ICSS
ICSS
ICSS
ICYY
IN0
IN1
IN2
IN3
4 BIT BINARY COUNTER
180 KHZ
SHOLD
ICQQ
ICRR
5v
1.3K 9.31K
0.1
555
0.01
READY
ICTT
DUMP
BCD COUNTERS
ICWW
ICXX
21.5K 100PF
ONE
DUMP
1K
21.5K 100PF
ZERO


BOARD IAQ-1

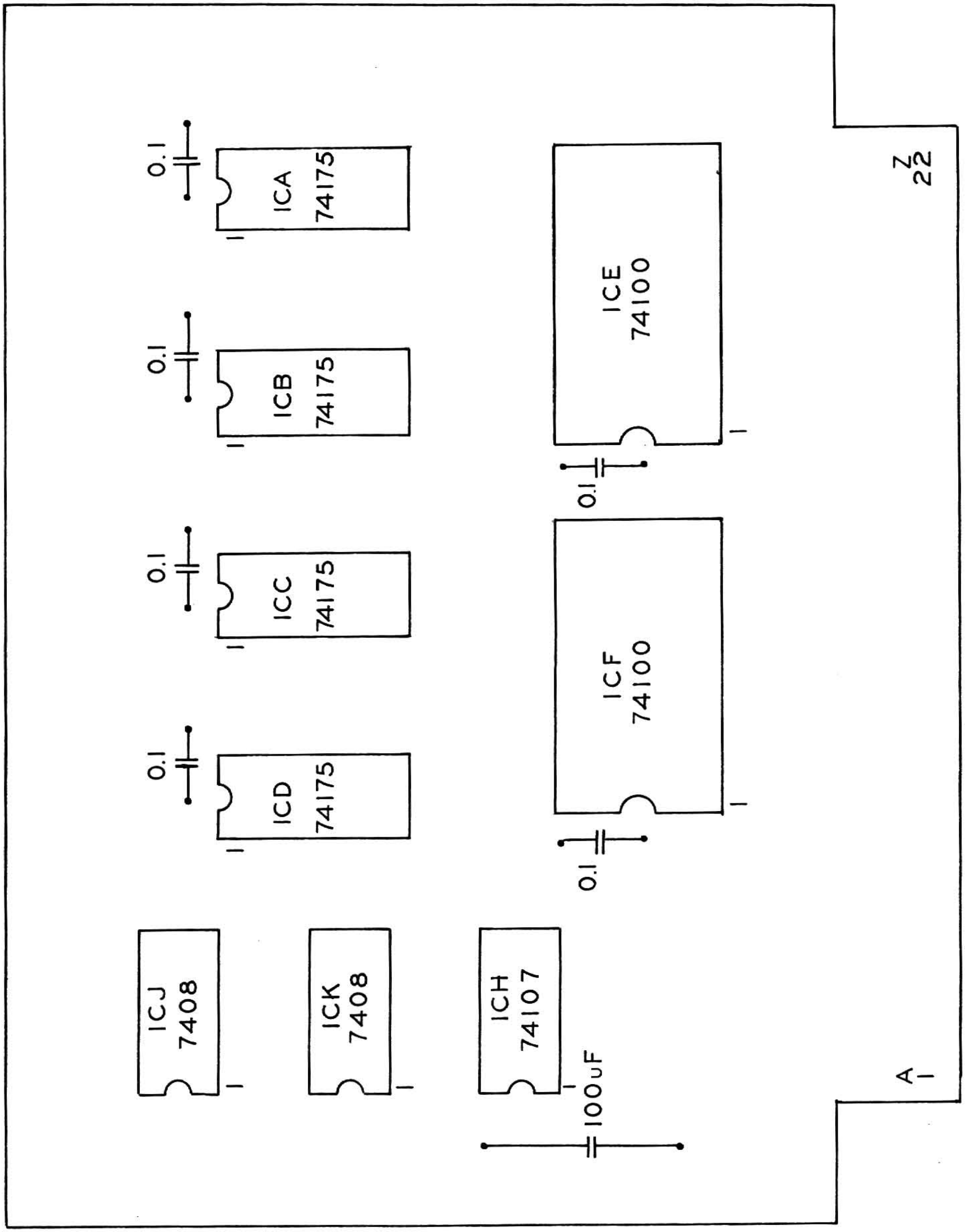

1, 78
1. 91
0. 1
5v
7
2
6
4
3
555
8
1
5
0.01
14
1
C
12
B
F SE1
9
J SE2
ICAAA
2
8
K SE3
3
11
H SE4
14
1
C
12
B
P SE5
9
M SE6
2
ICBBB
8
L SE7
3
11
N SE8

BOARD IAQ-2

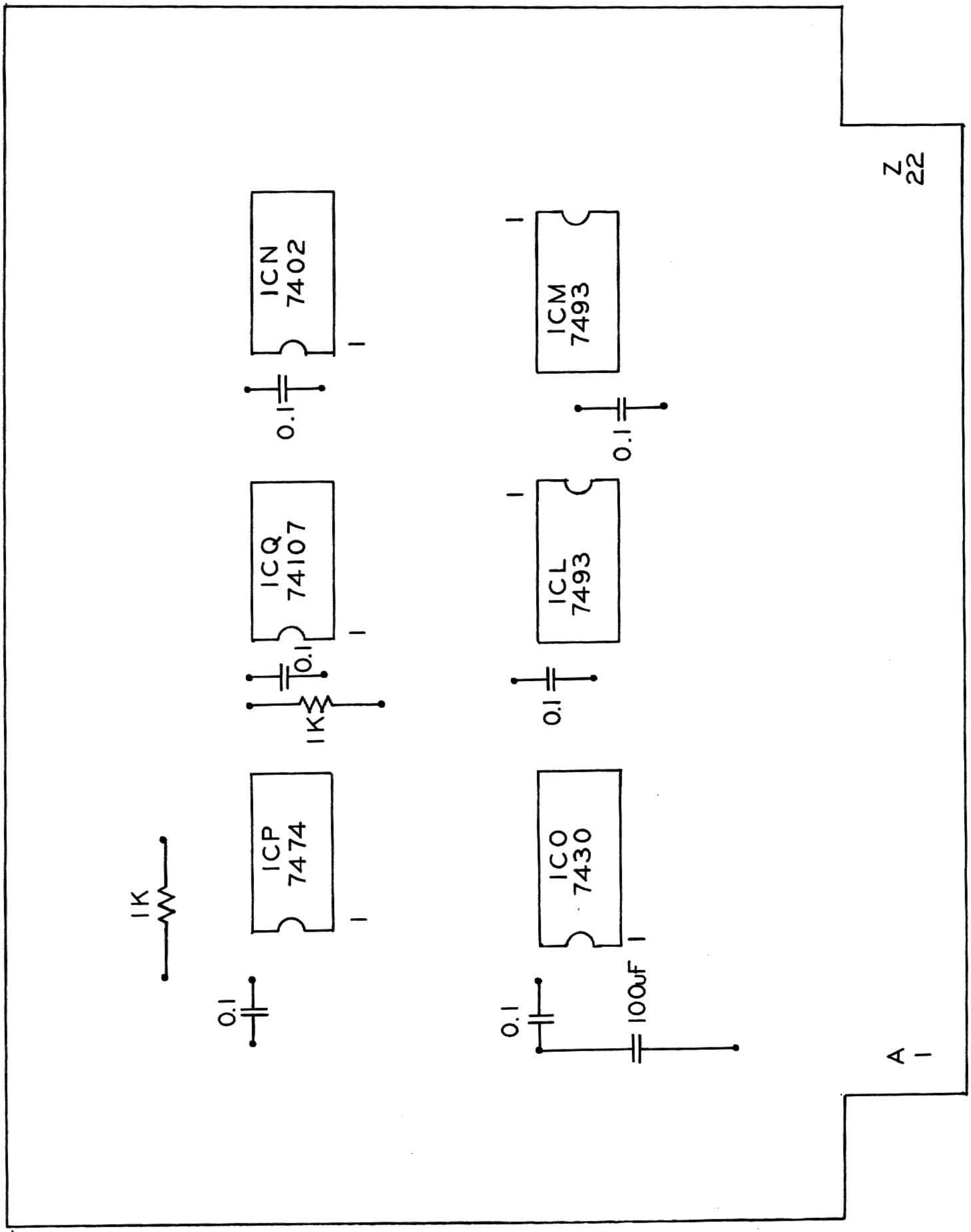

AZ
DUMP
RDY
COUNT
SHOLD
C1
C2
C3
C4
C5
DETECT ZERO DEGREES
ONE INTENSITY INPUT SEQUENCE FIRST 5.625°
FIFTIETH INTENSITY INPUT SEQUENCE
WAIT
NEXT 5.625°
NO SCALE

# APPENDIX B

## COMPONENT PLACEMENT DIAGRAMS

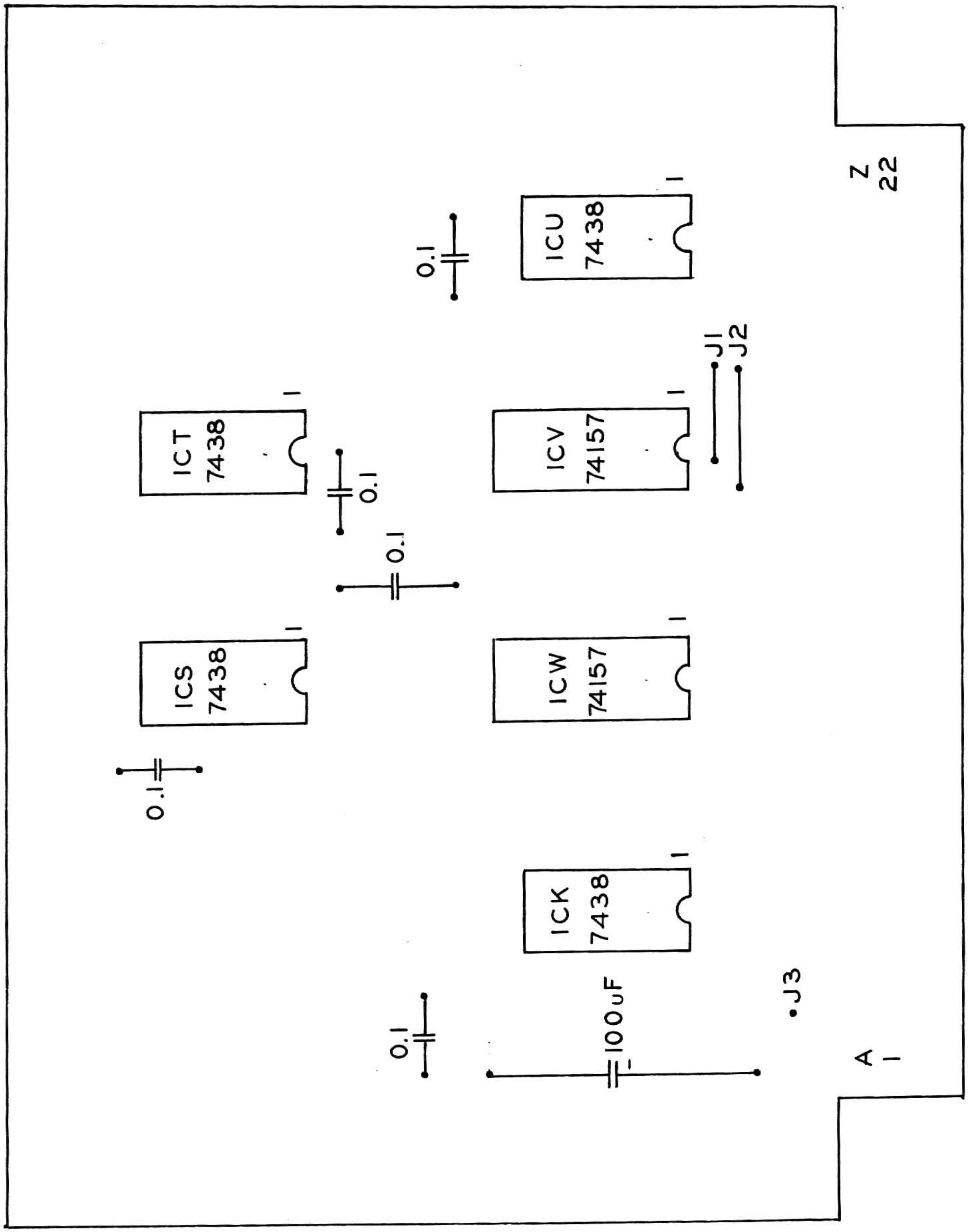

0.1
ICA
74175
ICB
74175
ICC
74175
ICD
74175
ICE
74100
ICF
74100
ICJ
7408
ICK
7408
ICH
74107
100uF
Z
22
A
1

BOARD A

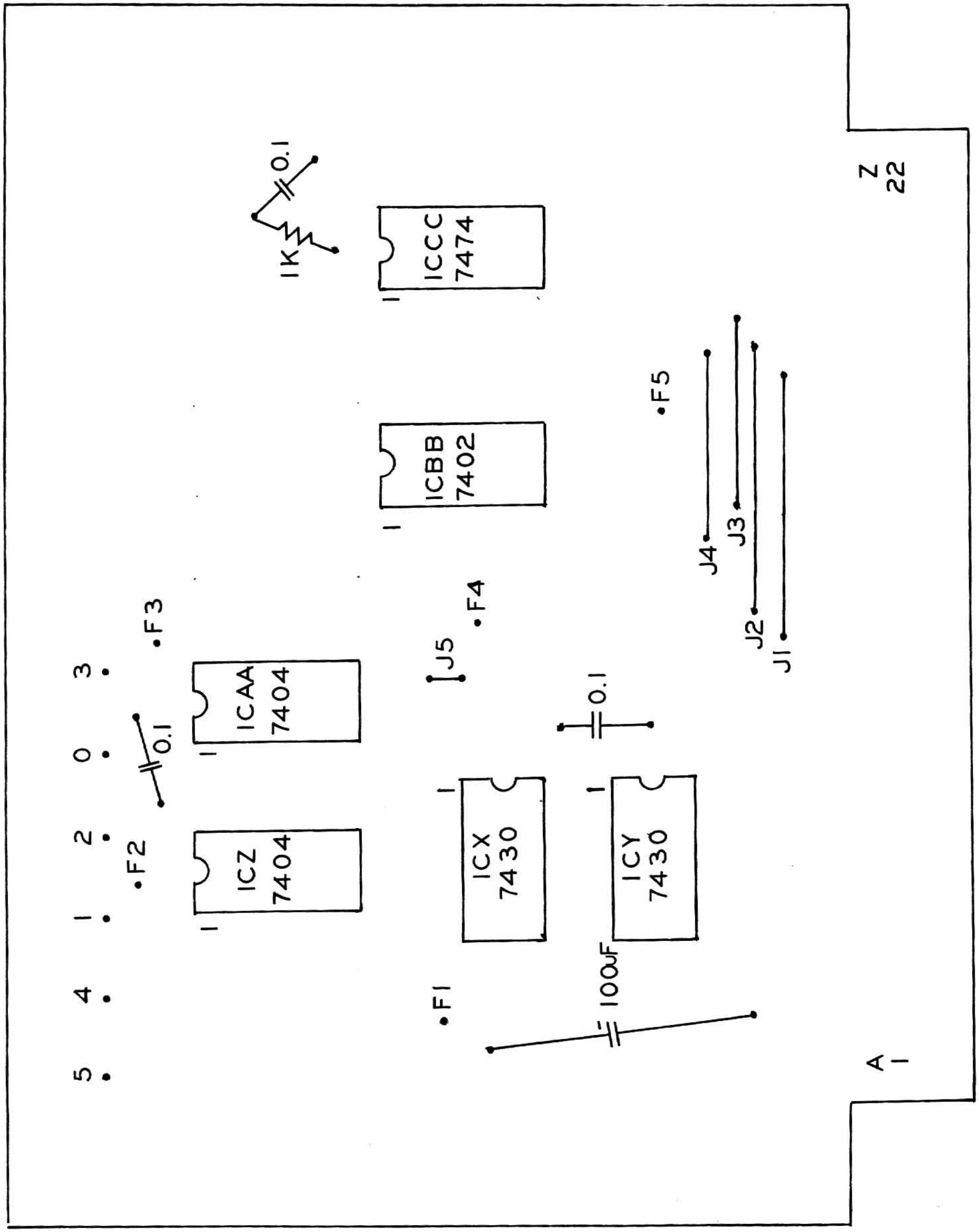


ICN
7402
ICQ
74107
ICP
7474
ICM
7493
ICL
7493
ICO
7430
0.1
0.1
0.1
0.1
0.1
0.1
0.1
1K
1K
100uF
Z
22
A

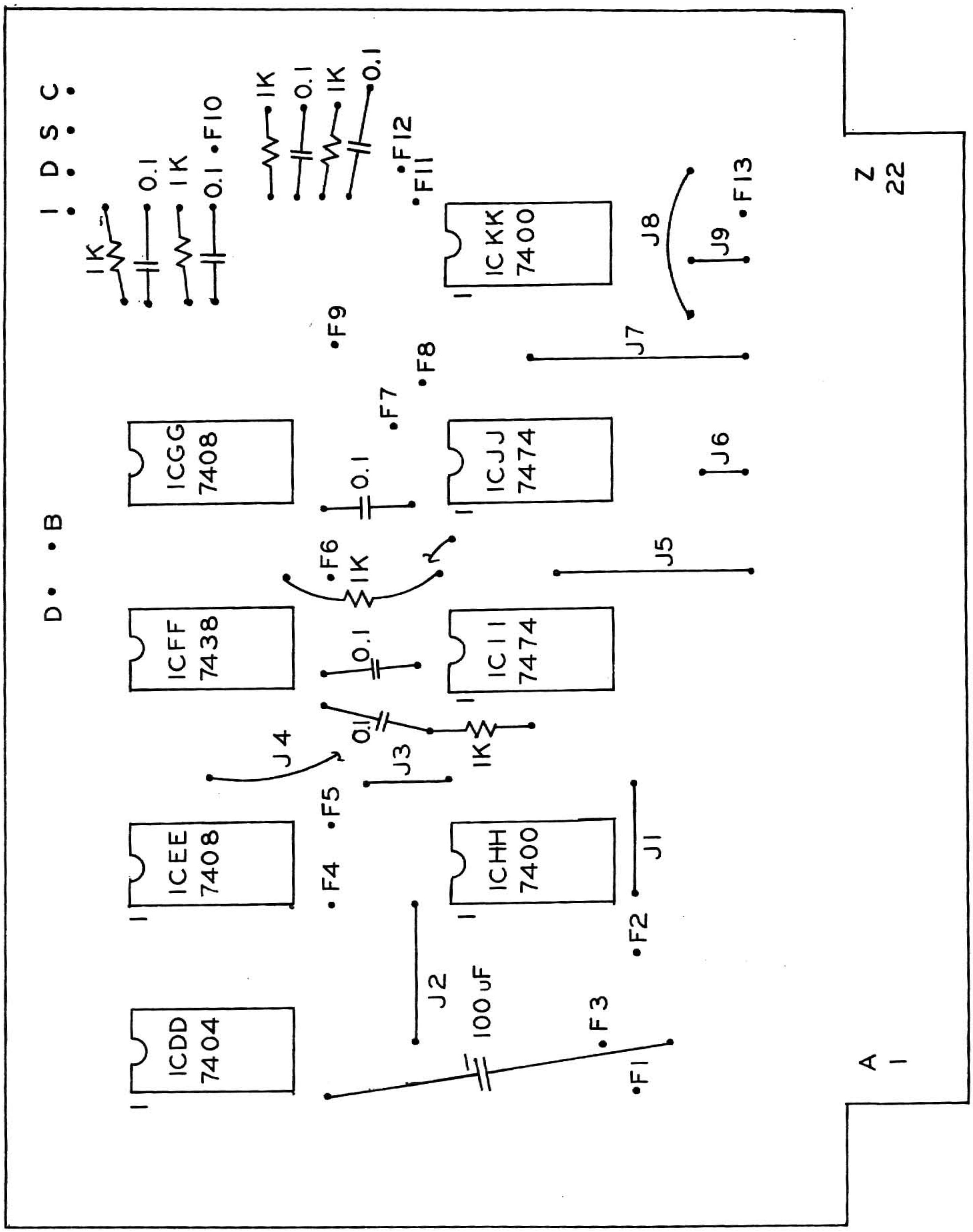

BOARD B


Z 22
ICU 7438
0.1
J1
J2
ICT 7438
0.1
ICV 74157
0.1
ICS 7438
ICW 74157
0.1
ICK 7438
0.1
100uF
J3
A 1

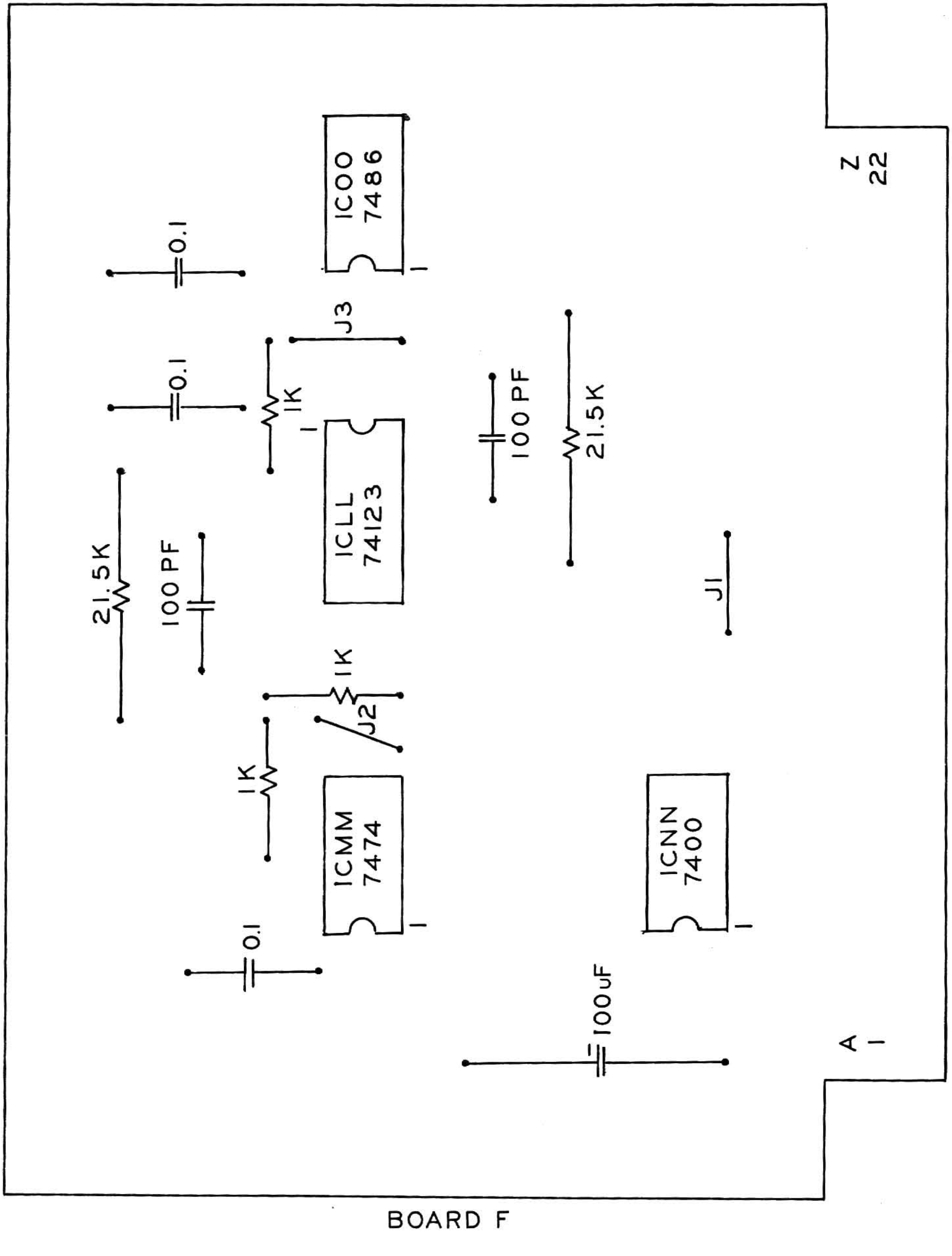
BOARD C


ICCC
7474
ICBB
7402
ICAA
7404
ICZ
7404
ICX
7430
ICY
7430
0.1
1K
0.1
0.1
100uF
F1
F2
F3
F4
F5
J1
J2
J3
J4
J5
Z
22
A
1
3
0
2
1
4
5

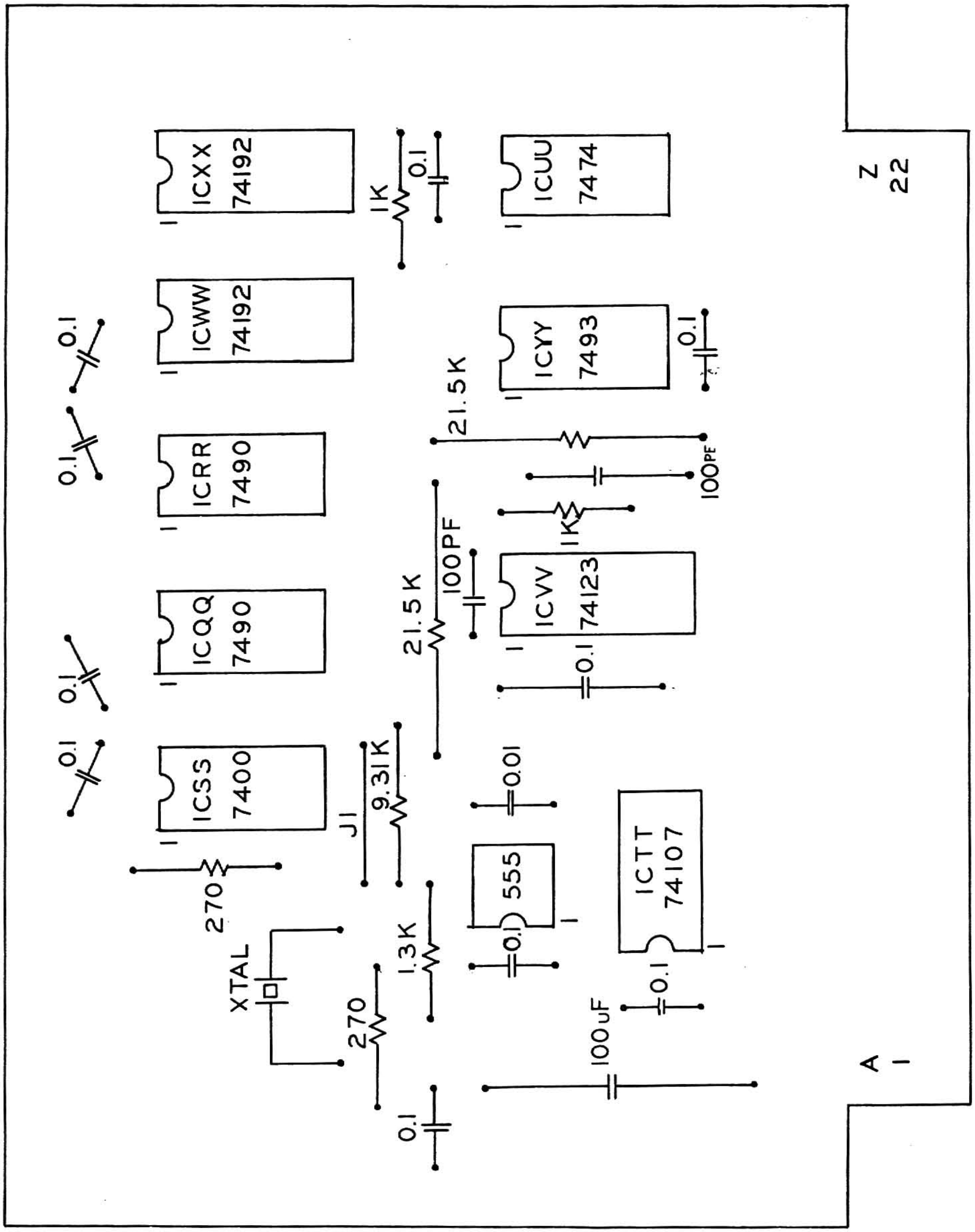
BOARD D


I D S C
0.1
1K
0.1
F10
1K
0.1
1K
0.1
F12
F11
F13
Z
22
ICKK
7400
J8
J9
J7
F9
F8
F7
ICGG
7408
ICJJ
7474
J6
0.1
D
B
F6
1K
J5
ICFF
7438
ICII
7474
0.1
0.1
1K
J4
J3
F5
ICEE
7408
ICHH
7400
J1
F4
F2
J2
100 uF
F3
ICDD
7404
F1
A
1

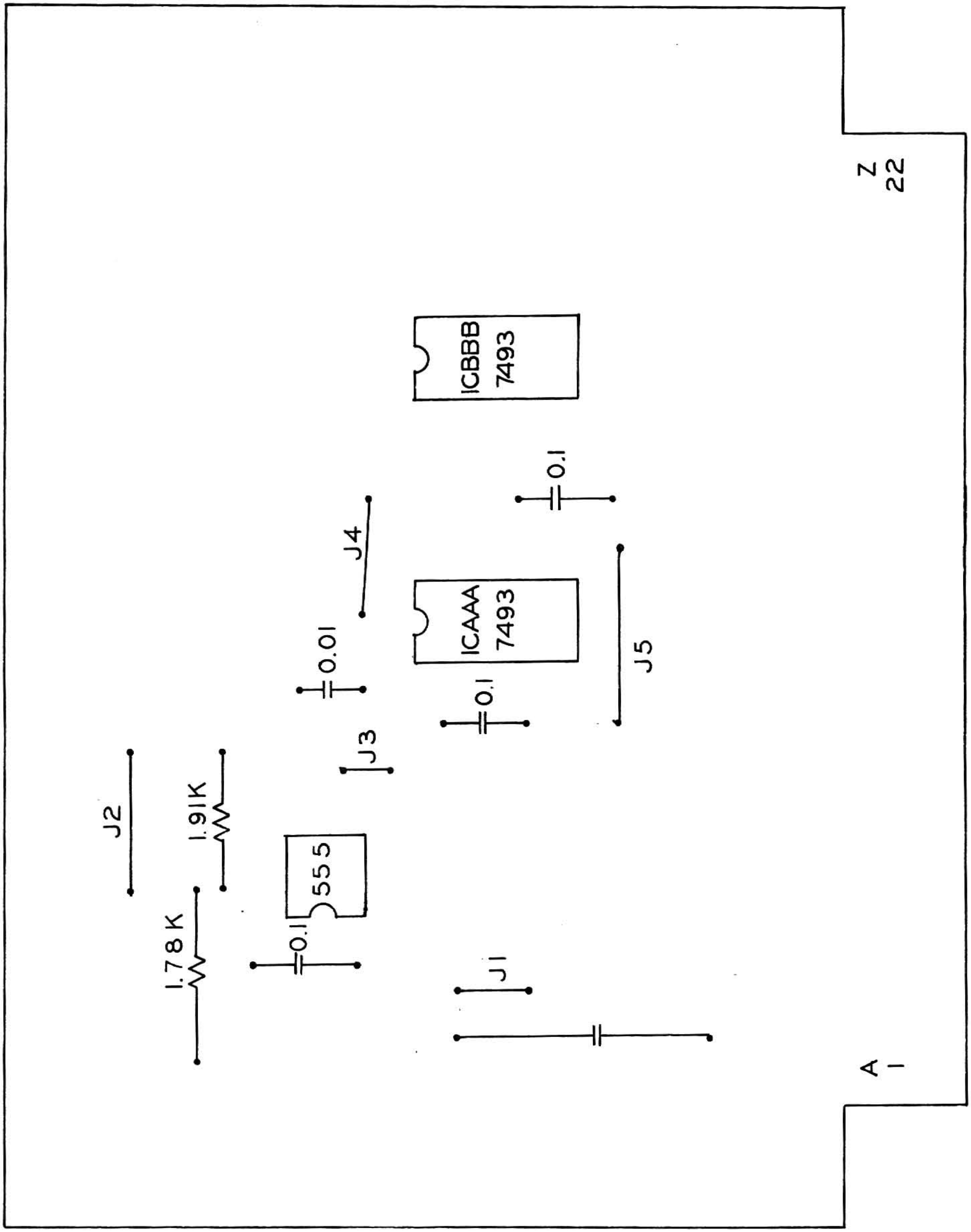
BOARD E


ICOO
7486
ICLL
74123
ICMM
7474
ICNN
7400
0.1
0.1
0.1
J1
J2
J3
1K
1K
1K
100 PF
100 PF
21.5K
21.5K
100uF
Z
22
A
1

BOARD F


ICXX
74192
ICWW
74192
ICRR
7490
ICQQ
7490
ICSS
7400
ICUU
7474
ICYY
7493
ICVV
74123
555
ICTT
74107
XTAL
J1
Z
22
A
1
1K
0.1
21.5K
100PF
9.31K
1.3K
270
0.01
100uF

BOARD IAQ-1


Z
22
ICBBB
7493
0.1
J4
ICAAA
7493
J5
0.01
0.1
J3
J2
1.91K
555
1.78K
0.1
J1
A
1

BOARD IAQ-2

# APPENDIX C

## SOURCE LISTING OF RDCP

RDCP

```
;RADAR DATA COLLECTION PROGRAM
;BY CARL C. ANDREASEN
;JANUARY 20 1976


;RDCP IS THE PROGRAM USED TO CONTROL THE
;DIGITAL COMPUTER INTERFACE OF THE DIGITAL
;RADAR PROCESSOR.
;REFERENCE (2) PRESENTS THE STYLE OF PRGORAMMING
;USED IN THIS PROGRAM.


;CREATE POINTER TABLE ENTRY
        .ENT    DSKR,DELET,RDMS,WORKF
        .ENT    NEXTA,DSKW,CATAL,FETCH
        .EXTD   PT1,TXTOT
        .ZREL
;PROCEDURE TO BUILD ALL NECESSARY LINKAGES TO DMS.

;DISPLACEMENT VALUES FOR RESIDENT DMS REFERENCES

DIVAL:  .+1     ;POINTER TO DISPLACEMENT VALUE TABLE
        -351    ;WORKFILE PARAMETER TABLE
        -325    ;NEXT AVAILABLE SECTOR ADDRESS IN DMS LIBRARY
        -324    ;RETURN ADDRESS TO DMS
        -320    ;FETCH RELOAD SECTOR ADDRESS
        -317    ;FETCH ROUTINE
        -340    ;DELETE ROUTINE
        -235    ;CATALOG ROUTINE
        -130    ;DISK READ ROUTINE
        -126    ;DISK WRITE ROUTINE
        -56     ;TELTYPE PRINT ROUTINE
C40:    40      ;END OF DIVAL TABLE FLAG &
                ;SECTOR ADDRESS OF EXECUTIVE ROUTINE
;RESIDENT DMS POINTERS

WORKF:  0       ;WORKFILE PARAMETER TABLE
NEXTA:  0       ;NEXT AVAILABLE SECTOR ADDRESS IN DMS LIBRARY
PAGEX:  0       ;RETURN ADDRESS TO DMS
LASTP:  0       ;FETCH RELOAD SECTOR ADDRESS
FETCH:  0       ;FETCH ROUTINE
DELET:  0       ;DELETE ROUTINE
CATAL:  0       ;CATALOG ROUTINE
DSKR:   0       ;DISK READ ROUTINE
DSKW:   0       ;DISK WRITE ROUTINE
TTOUT:  0       ;TELETYPE PRINT ROUTINE

RDMS:   57021   ;RETURN ADDRESS TO DMS
        .NREL
;POINTER TABLE PACKING DISPLACEMENT

PTPD=WORKF-DIVAL-1

;THIS ROUTINE BUILDS ALL DMS POINTERS

START:  LDA 2,DIVAL         ;GET POINTER TO DISPLACEMENT
```

```
                        ;VALUE TABLE TABLE
        LDA 0,0,2        ;GET A DISPLACEMENT VALUE
        MOVL# 0,0,SNC    ;IF VALUE IS POSITIVE, MUST BE END
                        ;OF TABLE
        JMP .+5         ;IT WAS POSITIVE,SO POINTER GENERATION
                        ;IS DONE
        ADD 3,0         ;WAS NOT POSITIVE, SO BUILD A POINTER
        STA 0,PTPD,2     ;PACK POINTER IN DMS POINTER TABLE
        INC 2,2         ;UPDATE DISPLACEMENT POINTER
        JMP START+1      ;GET NEXT DISPLACEMENT

;DMS POINTER TABLE HAS BEEN BUILT, NOW SET FETCH RELOAD
;SECTOR TO 40.

        STA @0,LASTP
        JMP     @PT1    ;GO TO PART 1 OF THE PROGRAM

        .END
```

```
;CREATE POINTER TABLE ENTRY
        .ENT    PT1,TABLE,SAMNU,STMEH,STMEL
        .EXTD   DATE,TIME,SAMP,DATAT,STIME,RDMS,FETCH,TXTOT,CRLF
        .EXTD   WORKF,DSKW,NEXTA,CATAL,HOURS,MIN,SEC
        .ZREL
PT1:    MIL1
SAMNU:  0       ;NUMBER OF DATA ACQUISITION RECORDS
STMEH:  0       ;STORAGE OF HIGH ORDER WORD OF SAMPLE TIME
STMEL:  0       ;STORAGE OF LOW ORDER WORD OF SAMPLE TIME
SECTC:  0       ;STORAGE FOR SECTOR COUNT
        .LOC    20
TABLE:  5000    ;ADDRESS OF DATA ACQUISITION RECORD
        .NREL
;INITIALIZE THE POINTER TO THE DATA ACQUISITION RECORD
;DATA BUFFER (DARDB)

MIL1:   LDA     0,MID1          ;LOAD ADDRESS OF DARDB
        STA     0,TABLE         ;STORE ADDRESS OF DARDB IN
                                ;AUTOINCREMENTING LOCATION "TABLE"
        DSZ     TABLE           ;DECREMENT DARDB ADDRESS BY 1
                                ;SINCE AUTOINCREMENTING LOCATION ADDS

                                ;BEFORE STORING DATA.
;OBTAIN THE DATE AND STORE THE DATE DARDB
        JSR     @DATE
;OBTAIN THE TIME AND STORE THE TIME IN DARDB
        JSR     @TIME
;OBTAIN THE NUMBER OF DATA ACQUISITION RECORDS TO BE RECORDED
;THE RESULT BEING STORED IN LOW CORE LOCATION "SAMNU"
        JSR     @SAMP
;OBTAIN THE TIME INTERVAL BETWEEN DATA ACQUISITION SEQUENCES.
        JSR     @STIME
;ESTABLISH A FILE NAME OF THE FORM RDMN.DA
;WHERE MN = 00 ... 99
        LDA     3,WORKF         ;LOAD ADDRESS OF WFPT
        LDA     1,MID2          ;OBTAIN FIRST 2 CHARACTERS OF
                                ;FILE NAME
        STA     1,0,3           ;STORE IN WFPT
        LDA     1,MID3          ;LOAD SECOND 2 CHAR. OF FILE NAME
        STA     1,1,3           ;STORE THEM IN WFPT
MIL3:   SUBZR   0,0             ;SET BIT 0=1
        STA     0,2,3           ;INITIALIZE WORDS 2-6 OF WFPT
        STA     0,3,3
        STA     0,4,3
        STA     0,5,3
        STA     0,6,3
        LDA     1,MID4          ;LOAD DISK FILE ID
        STA     1,7,3           ;STORE DISK FILE ID
;HAVE ESTABLISHED WFPT AT DEFAULT MODE
;NOW CHECK FOR THE NEXT RD##.DA
;WHERE # CAN BE AN INTEGER FROM 0-9
        JSR     @FETCH          ;FOR THE FILE IN THE WFPT, SEE IF
                                ;IS ON THE DISK
        JMP     MIL5            ;IT WAS NOT
        JMP     MIL5            ;IT WAS NOT
        LDA     1,MID3          ;LOAD SECOND 2 CHARACTERS OF
                                ;FILE NAME
```

```
        MOVS    1,1         ;JUSTIFIED RIGHT-LEFT, NOW ARE
                            ;LEFT-RIGHT
        LDA     0,MID5      ;LOAD MASK TO KEEP RIGHT HAND SIDE
        AND     0,1         ;KEEP JUST RHS
        LDA     0,MID6      ;LOAD MASK FOR '9'
        SUB#    0,1,SNR     ;TEST IF CHAR=MASK
        JMP     MIL4        ;IS A 9 SO CHECK OTHER SIDE
        LDA     1,MID3      ;LOAD SECOND 2 CHARACTERS OF FILE NAME
        MOVS    1,1         ;WAS NOT 9 SO INCREMENT
        INC     1,1         ;BY 1 AND STORE THEM IN
        MOVS    1,1
        LDA     3,WORKF     ;LOAD ADDRESS OF WFPT
        STA     1,1,3       ;STORE SECOND 2 CHAR IN WFPT
        STA     1,MID3      ;UPDATE THE SECOND 2 CHARACTERS
        JMP     MIL3        ;GO CHECK IF THIS ONE EXITS
MIL4:   LDA     1,MID3      ;LOAD THE SECOND 2 CHARACTERS
        LDA     0,MID5      ;LOAD MASK TO KEEP RHS
        AND     0,1         ;KEEP JUST RHS
        LDA     0,MID6      ;LOAD MASK OF '9'
        SUB#    0,1,SZR     ;TEST IF MASK=CHAR
        JMP     .+5         ;WAS NOT A 9
        JSR     @CRLF
        JSR     @TXTOT      ;WAS A 9 SO RD99.DA EXISTS
        MIT1
        JSR     @RDMS
        LDA     1,MID3      ;LOAD THE SECOND 2 CHARACTERS
                            ;OF FILE NAME
        INC     1,1         ;SINCE WAS NOT 9, INCREMENT IT
        LDA     0,MID7      ;LOAD MASK TO RESET CHAR '0' IN LHS
        AND     0,1         ;HAVE NEXT SECOND 2 CHARACTERS
                            ;OF FILE NAME
        LDA     3,WORKF     ;LOAD ADDRESS OF WFPT
        STA     1,1,3       ;STORE NEW SECOND 2 CHARACTERS
        STA     1,MID3      ;UPDATE SECOND 2 CHARACTERS
        JMP     MIL3        ;GO SEE IF THE NEW FILE NAME EXISTS
        .ZREL
MID1:   5000    ;ADDRESS OF DANDB
MID2:   042122  ;CHARACTERS 'DR'
MID3:   030060  ;CHARACTERS '00'
MID4:   040504  ;CHARACTERS 'AD'
MID5:   377     ;MASK TO KEEP RHS
MID6:   071     ;CHARACTER '9'
MID7:   030377  ;CHARACTER'0' MASK FOR LHS
MID9:   0
MID10:  -60.    ;RTC COUNTER
MID15:  0       ;TEMPORARY LOCATION
MID16:  0       ;TEMPORARY LOCATION
MID12:  MIL10   ;ADDRESS OF INTERRUPT HANDLER
MID13:  177773  ;INTERRUPT MASK
MID14:  00014   ;RTC DEVICE CODE
MID17:  5       ;FUDGE FACTOR TIME CONSTANT
MID18:  -64.    ;# OF 5.625 DEGREE INCREMENTS IN 360 DEGREES
MID19:  -50.    ;#OF AZIMUTH WORDS TO BE TRANSFERED
MIA1:   MIL3
MIT1:   .TXT    &ALL FILES FROM RD00.DA TO RD99.DA EXIST&
MIT2:   .TXT    &BEGIN SAMPLING&
MIT3:   .TXT    &END SAMPLING&
        .NREL
```

```
;READ THE DATA FROM THE INTERFACE
MIL5:   JSR     @CRLF
        JSR     @TXTOT
        MIT2
        JSR     @CRLF
        JSR     @DATAR
        JSR     @CRLF
        JSR     @TXTOT
        MIT3
;OUTPUT THE DARDB
;"TABLE" POINTS TO END OF DARDB
;"MIL1" POINTS TO BEGINNING OF DARDB
;STORE THE DATA
MIL6:   SUB     2,2         ;CLEAR AC2 FOR SECTOR COUNT
        STA     2,SECTC     ;CLEAR SECTOR COUNT LOCATION
        LDA     1,MID1      ;MOVE BEGINNING POINTER TO USE LOCATIN

        STA     1,MID9
        LDA     3,WORKF     ;LOAD ADDRESS OF WFPT
        LDA     0,@NEXTA    ;LOAD NEXT SECTOR ADDRESS
        STA     0,3,3       ;STORE NEXT SECTOR ADDRESS IN WFPT
;                           AS IT IS STARTING SECTOR FOR THIS FILE
        LDA     0,@NEXTA    ;LOAD NEXT SECTOR ADDRESS
        LDA     1,MID9      ;LOAD START ADDRESS OF DATA
        JSR     @DSKW       ;WRITE THE DATA
        ISZ     SECTC       ;INCREMENT THE SECTOR COUNT
        STA     0,@NEXTA    ;STORE NEXT SECTOR ADDRESS
        STA     1,MID9      ;STORE START ADDRESS OF DATA + 400
        LDA     0,TABLE     ;LOAD MAX CORE USED
        SUBZ#   1,0,SZC     ;SKIP WHEN ALL WRITTEN
        JMP     .-10
        LDA     3,WORKF     ;LOAD ADDRESS OF WFPT
        LDA     2,SECTC     ;LOAD THE SECTOR COUNT
        STA     2,4,3       ;STORE SECTOR COUNT
        SUB     2,2
        STA     2,5,3       ;CLEAR WORD 5 OF WFPT
        JSR     @CATAL      ;CATALOG THIS FILE
;TEST IF MORE DATA ACQUISITION SEQUENCES TO BE RECORDED
MIL7:   DSZ     SAMNU       ;DECREMENT # OF DATA ACQUISITION
                            ;SEQUENCES
;                             TO BE RECORDED.
        JMP     .+2         ;MORE YET TO GO
        JMP     @RDMS       ;RETURN TO OPERATING SYSTEM
;WAIT DESIRED INTERVAL
MIL8:   SUB     2,2         ;CLEAR AC2 FOR A COUNTER
        DOA     2,RTC       ;SET RTC FOR 60 HZ OPERATION
        NIOS    RTC
        LDA     1,MID12     ;LOAD AND STORE
        STA     1,1         ;ADDRESS OF INTERRUPT HANDLER
        LDA     0,STMEH
        NEG     0,0         ;FORMS 2'S COMPLEMENT OF INTERVAL TIME
        STA     0,MID15
        LDA     0,STMEL
        NEG     0,0         ;FORMS 2'S COMPLEMENT OF INTERVAL TIME
        STA     0,MID16
        LDA     1,MID13     ;LOAD INTERRUPT MASK
        DOB     1,CPU       ;EXECUTE MSKO
MIL11:  LDA     0,MID10     ;LOAD -60 DEC. AS # 60 HZ
```

```
                              ;INTERRUPTS TO COUNT
M1L9:     INTEN               ;ENABLE INTERRUPTS
          JMP     .+1
          JMP     .-1         ;WAIT
M1L10:    INTA    2           ;INPUT DEVICE CODE OF
                              ;INTERRUPTING DEVICE
          LDA     1,M1D14     ;LOAD DEVICE MASK
          NIOS    RTC         ;WAKE UP THE RTC AGAIN
          SUB#    2,1,SZR     ;TEST IF DEVICE CODE 13
          JSR     @RDMS
          INC     0,0,SZR     ;INCREMENT COUNTER
          JMP     M1L9        ;NOT ENOUGH
          LDA     2,M1D16     ;LOAD LOW ORDER OF INTERVAL
                              ;TIME IN SECONDS
          LDA     1,M1D15     ;LOAD HIGH ORDER OF INTERVAL
                              ;TIME IN SECONDS
          MOV     2,2,SNR     ;TEST IF LOW ORDER IS ZERO
          JMP     M1L12       ;YES, THEN PROCESS HIGH ORDER
          INC     2,2         ;NO, INCREMENT ONE SECOND
          STA     2,M1D16     ;STORE NEW INTERVAL TIME COUNT
          JMP     M1L11       ;INTERRUPT FOR NEXT SECOND
M1L12:    MOV     1,1,SNR     ;TEST IF HIGH ORDER IS ZERO
          JMP     M1L13       ;YES, THEN BOTH HIGH AND
                              ;LOW ORDER INTERVAL TIME
;                              COUNT IS ZERO, SO INTERVAL TIME
;                              IS UP. WRITE NEXT DARDB
          INC     1,1         ;NO, INCREMENT ONE SECOND
          STA     1,M1D15     ;STORE NEW HIGH ORDER TIME COUNT
          JMP     M1L11       ;CONTINUE FOR NEXT SECOND
M1L13:    SUB     1,1         ;CLEAR AC1
          DOB     1,CPU       ;EXECUTE MSKO INSTRUCTION
          LDA     3,M1D1      ;RESET TABLE POINTER
          STA     3,TABLE
          DSZ     TABLE       ;DECREMENT ADDRESS BY 1
          LDA     0,3,3       ;LOAD HOURS
          LDA     1,HOURS     ;LOAD INTERVAL TIME HOURS
          ADD     0,1
          STA     1,3,3       ;STORE TIMEHOURS + INTERVAL HOURS
          LDA     0,4,3       ;LOAD MINUTES
          LDA     1,MIN       ;LOAD INTERVAL TIME MINUTES
          ADD     0,1
          STA     1,4,3       ;STORE TIME MINUTES + INTERVAL MINUTES
          LDA     0,5,3       ;LOAD SECONDS
          LDA     1,SEC       ;LOAD INTERVAL SECONDS
          ADD     0,1
          LDA     0,M1D17
          ADD     0,1
          STA     1,5,3       ;STORE TIME SECONDS + INTERVAL SECONDS
+ 5
          JMP     @M1A1       ;CONTINUE FOR REST OF DESIRED
                              ; # OF RECORDS
;                 DATA FROM DATA ACQUISITION SEQUENCE
          .END
```

```
        TXTOT
;THIS SUBROUTINE PRINTS A TEXT TO THE TELETYPE
;
;CALLING SEQUENCE:
;        JSR     @TXTOT
;        RETURN
;
;INPUTS:AC3      POINTS TO ADDRESS OF THE TEXT
;
;OUTPUTS:         NONE
;
;CREATE ENTRY TO POINTER TABLE
         .ENT    TXTOT
         .ZREL
TXTOT:   S1L1
         .NREL
;
S1L1:    STA     3,S1D0
         NIOS    TTO
         STA     2,S1D0+1
         STA     1,S1D0+2
         STA     0,S1D0+3
         LDA     1,S1D1      ;LOAD THE MASK
S1L2:    LDA     3,0,3       ;LOAD ADDRESS OF WORD OF TEXT
S1L4:    LDA     0,0,3       ;LOAD A WORD OF TEXT
         AND#    0,1,SNR     ;TEST IF BITS 0-7 ARE 0
         JMP     S1L3
         SKPBZ   TTO
         JMP     .-1
         DOAS    0,TTO
         MOVS    0,0         ;GET THE SECOND HALF
         AND#    0,1,SNR     ;TEST IF BITS 0-7 ARE 0
         JMP     S1L3
         SKPBZ   TTO
         JMP     .-1
         DOAS    0,TTO
         INC     3,3
         JMP     S1L4
S1L3:    LDA     0,S1D0+3
         LDA     1,S1D0+2
         LDA     2,S1D0+1
         ISZ     S1D0        ;INCREMENT PAST POINTER
         LDA     3,S1D0
         JMP     @S1D0
S1D0:    .BLK    4
S1D1:    00377
         .END
```

```
        CHOUT
;THIS SUBROUTINE OUTPUTS A RIGHT JUSTIFIED CHARACTER FROM AC0
;
;CALLING SEQUENCE:
;       JSR     @CHOUT
;       RETURN HERE
;
;INPUTS:AC0     CHARACTER IN BITS 7-15 TO GO OUT
;
;OUTPUTS:        NONE
;
;CREATE ENTRY TO POINTER TABLE
        .ENT    CHOUT
        .ZREL
CHOUT:  S2L1
        .NREL
;
S2L1:   STA     3,S2D0
        STA     2,S2D0+1
        STA     1,S2D0+2
        STA     0,S2D0+3
        SKPBZ   TTO             ;CHECK IF TELETYPE IS PRINTING
        JMP     .-1
        DOAS    0,TTO           ;OUTPUT CHARACTER
        LDA     0,S2D0+3
        LDA     1,S2D0+2
        LDA     2,S2D0+1
        LDA     3,S2D0
        JMP     @S2D0
;DATA
S2D0:   .BLK    4
        .END
```

```
        CREAD
;
;CALLING SEQUENCE:
;       JSR     @CREAD
;       RETURN HERE
;
;INPUTS:NONE
;
;OUTPUTS:          AC0         RESULT OF READ IN BITS 7-15
;
;CREATE ENTRY TO POINTER TABLE
        .ENT    CREAD
        .ZREL
CREAD:  S3L1
        .NREL
;
S3L1:   STA     3,S3D0      ;STORE ENVIRONMENT
        STA     2,S3D0+1
        STA     1,S3D0+2
        NIOS    TTI         ;START THE TELETYPE INPUT
        SKPDN   TTI         ;WAIT FOR A KEYSTROKE
        JMP     .-1
        DIAS    0,TTI       ;BRING IN A CHARACTER FROM TTI
        NIOC    TTI
        LDA     1,S3D1      ;LOAD MASK TO ELIMINATE PARITY BIT
        AND     1,0         ;MASK OFF PARITY
        LDA     2,S3D0+1
        LDA     1,S3D0+2
        JMP     @S3D0       ;AND RETURN
S3D0:   .BLK    3
S3D1:   177
        .END
```

```
        CRLF
;THIS SUBROUTINE OUTPUTS A CR AND LF
;
;CALLING SEQUENCE:
;       JSR     @CRLF
;       RETURN
;
;INPUTS:NONE
;
;OUTPUTS:        NONE
;
;SUPPORT SUBROUTINES
;       DELAY
;
;CREATE POINTER TABLE ENTRY
        .ENT    CRLF
        .EXTD   DELAY
        .ZREL
CRLF:   S8L1
        .NREL
;
S8L1:   STA     3,S8D0
        STA     2,S8D0+1
        STA     1,S8D0+2
        STA     0,S8D0+3
        LDA     1,S8D1          ;LOAD CR AND LF CHARACTERS
        SKPBZ   TTO             ;TEST IF OUTPUT BEING DONE
        JMP     .-1
        DOAS    1,TTO           ;OUTPUT CR
        JSR     @DELAY          ;GIVE A DELAY
        MOVS    1,1
        SKPBZ   TTO
        JMP     .-1
        DOAS    1,TTO           ;OUTPUT THE LF
        LDA     0,S8D0+3
        LDA     1,S8D0+2
        LDA     2,S8D0+1
        LDA     3,S8D0
        JMP     @S8D0
;DATA
S8D0:   .BLK    4
S8D1:   .TXT    '<15><12>'
        .END
```

```
        DELAY
;THIS SUBROUTINE GIVES A SHORT DELAY
;SO THAT THE CARRIAGE CAN RETURN WHEN USING THE TTY
;
;CALLING SEQUENCE
;         JSR     @DELAY
;         RETURN
;
;INPUTS:NONE
;
;OUTPUTS:         NONE
;
;CREATE POINTER TABLE ENTRY
        .ENT    DELAY
        .ZREL
DELAY:  S9L1
        .NREL
;
S9L1:   STA     3,S9D0
        STA     2,S9D0+1
        LDA     2,S9D1
        STA     2,S9D2
        DSZ     S9D2
        JMP     .-1
        LDA     2,S9D0+1
        JMP     @S9D0
S9D0:   .BLK    2
S9D1:   500
S9D2:   0
        .END
```

```
                BELL
;         OUTPUTS THE BELL CHARACTER TO THE TTO
;
;CALLING SEQUENCE
;         JSR     @BELL
;         RETURN
;
;INPUTS:NONE
;
;OUTPUTS:          NONE
;
;SUPPORT SUBROUTINES
;         DELAY
;
;CREATE POINTER TABLE ENTRY
          .ENT    BELL
          .EXTD   DELAY
          .ZREL
;
BELL:     SEL1
          .NREL
SEL1:     STA     3,SEDØ
          STA     2,SEDØ+1
          STA     1,SEDØ+2
          STA     Ø,SEDØ+3
          LDA     Ø,SED1
          DOAS    Ø,TTO
          JSR     @DELAY
          JSR     @DELAY
          JSR     @DELAY
          SKPBZ   TTO
          JMP     .-1
          LDA     Ø,SEDØ+3
          LDA     1,SEDØ+2
          LDA     2,SEDØ+1
          JMP     @SEDØ
SEDØ:     .BLK    4
SED1:     ØØ7
          .END
```

```
        DATE
;
;THIS SUBROUTINE OBTAINS THE DATE IN THE FORMAT
;       MONTH/DAY/YEAR
;WHERE EACH FIELD IS A 2 DIGIT INTEGER VALUE
;AND IS STORED AS A BINARY VALUE.
;AN AUTOINCREMENTING LOCATION "TABLE" MUST
;EXIST. THE DATE WILL BE STORED INDIRECTLY
;THROUGH "TABLE"
;
;CALLING SEQUENCE
;       JSR     @DATE
;       RETRN
;
;SUPPORT SUBROUTINES
;       TXTOT
;       CRLF
;       GETNUM
;       CHOUT
;       ERROR
;
;CREATE POINTER TABLE ENTRY
        .ENT    DATE
        .EXTD   TXTOT,CRLF,ERROR,GETNUM,TABLE,CHOUT
        .ZREL
DATE:   SIL1
        .NREL
;
SIL1:   STA     3,SID0      ;STORE ENVIRONMENT
        STA     2,SID0+1
        STA     1,SID0+2
        STA     0,SID0+3
SIL2:   JSR     @CRLF
        JSR     @TXTOT      ;PRINT QUESTION
        SIT1
        LDA     0,SID2      ;LOAD # OF DIGITS
        LDA     1,SID4      ;LOAD MAXIMUM VALUE FOR MONTH
        JSR     @GETNUM     ;OBTAIN MONTH
        JMP     SIL3        ;ERROR,RETURN HERE
        STA     0,@TABLE    ;STORE MONTH
        LDA     0,SID7      ;LOAD CHACTER "/"
        JSR     @CHOUT
        LDA     0,SID2      ;LOAD # OF DIGITS
        LDA     1,SID6      ;LOAD MAX FOR DAY
        JSR     @GETNUM     ;OBTAIN DAY
        JMP     SIL4        ;ERROR,RETURN HERE
        STA     0,@TABLE    ;STORE DAY
        LDA     0,SID7
        JSR     @CHOUT
        LDA     0,SID2      ;LOAD # OF DIGITS
        LDA     1,SID5      ;LOAD MAXIMUM YEAR VALUE
        JSR     @GETNUM     ;OBTAIN YEAR
        JMP     SIL5        ;ERROR,RETURN HERE
        STA     0,@TABLE    ;STORE YEAR
        LDA     0,SID0+3    ;RESTORE ENVIRONMENT
        LDA     1,SID0+2
        LDA     2,SID0+1
        LDA     3,SID0
```

```
        JMP     @SIDØ
SIL5:   DSZ     TABLE
SIL4:   DSZ     TABLE
SIL3:   JSR     @ERROR
        JMP     SIL2
;DATA
SIDØ:   .BLK    4           ;ENVIRONMENT STORAGE
SID1:   Ø
SID2:   2       ;MAX # OF DIGITS
SID3:   Ø
SID4:   12.     ;MAXIMUM MONTH VALUE
SID5:   99.     ;MAXIMUM YEAR VALUE
SID6:   31.     ;MAXIMUM DAY VALUE
SID7:   Ø57     ;CHARACTER "/"
SIT1:   .TXT    &DATE (MONTH/DAY/YEAR)  :  &
        .END
```

GETNUM

```
;
;THIS SUBROUTINE OBTAINS A BCD NUMBER
;FROM THE TTY. VALUES ARE POSITIVE AND
;UP TO 4 BCD DIGITS IN LENGTH (MAX=9999)
;THEN THE BCD VALUE IS CONVERTED TO BINARY AND RETURNED
;IN AC0
;
;INPUTS:AC0      NUMBER OF DIGITS
;                AC1          MAXIMUM VALUE
;                             MINIMUM VALUE IS 0
;
;OUTPUTS:         AC0          (A BINARY NUMBER EQUAL TO BCD VALUE
;                             READ IN)
;
;CALLING SEQUENCE
;        JSR     @GETNUM
;        RETURN IF ERROR
;        NORMAL RETURN
;
;CREATE POINTER TABLE ENTRY
        .ENT    GETNUM
        .EXTD   CREAD,CHOUT
        .ZREL
GETNUM: SJL1
        .NREL
;
;
SJL1:   STA     3,SJD0      ;STORE ENVIRONMENT
        STA     2,SJD0+1
        STA     1,SJD0+2
        STA     0,SJD0+3
        STA     0,SJD1      ;STORE DIGIT # COUNT
        SUBZ    2,2         ;CLEAR AC2
SJL6:   JSR     @CREAD      ;GET A CHARACTER
        JSR     @CHOUT      ;ECHO IT
        LDA     1,SJD3      ;GET MAX CHAR LIMIT
        SUBZ#   1,0,SZC     ;SKIP IF MAX>WHAT READ IN
        JMP     SJL3
        LDA     1,SJD2      ;GET MINIMUM CHAR LIMIT
        SUBZ#   1,0,SNC     ;SKIP IF MIN <OR= WHAT READ IN
        JMP     SJL3        ;INCORRECT INPUT
        LDA     1,SJD7      ;GET  CHARACTER MASK
        AND     1,0         ;OBTAIN OCTAL EQUIVALENT
                            ;OF THE CHARACTER
        DSZ     SJD1        ;DECREMENT # OF DIGITS
        JMP     SJL2        ;SHIFT LEFT 4 BITS FOR NEXT INPUT
        ADD     0,2         ;WAS LAST DIGIT,SO ADD IT TO SUM
        STA     2,SJD6      ;STORE RESULT IN TEMPORARY LOCATION
        LDA     0,SJD4      ;LOAD COUNTER (# BCD DIGITS MAX)
        STA     0,SJD1      ;STORE IN COUNTER LOCATION
        SUB     2,2         ;CLEAR AC2 FOR RESULT REGISTER
SJL4:   JSR     SJL5        ;GET DIGIT
        JSR     SJL7        ;MULTIPLY SUM BY 10 AND ADD DIGIT
        ISZ     SJD1        ;DONE YET?
        JMP     SJL4        ;NO
        MOVZ    2,0         ;RESULT TO ACC0, CLEAR CARRY
        JMP     SJL8
```

```
SJL5:   LDA     1,SJD4      ;LOAD LOOP COUNTER
        STA     1,SJD5      ;STORE IT FOR USE
        LDA     1,SJD6      ;LOAD THE VALUE TO BE CONVERTED
;                            TO BINARY
        SUB     0,0         ;CLEAR AC0
        MOVZL   1,1         ;SHIFT OUT A BIT
        MOVL    0,0         ;MOVE THE BIT TO AC0
        ISZ     SJD5        ;INCREMENT LOOP COUNTER AND
        JMP     .-3         ;LOOP TILL A DIGIT SHIFTED TO AC0
        STA     1,SJD6      ;STORE CURRENT WORD
        JMP     0,3         ;RETURN
SJL7:   MOVZL   2,1         ;N*2
        MOVZL   1,1         ;N*4
        ADD     2,1         ;N*5
        MOVZL   1,2         ;N*10
        ADD     0,2         ;N*10+AC0
        JMP     0,3         ;RETURN
SJL2:   MOVZL   0,0         ;SHIFT LEFT 4 PLACES
        MOVZL   0,0
        MOVZL   0,0
        MOVZL   0,0
        ADD     0,2         ;FORM RESULT IN AC2
        JMP     SJL6        ;CONTINUE
SJL8:   LDA     1,SJD0+2    ;TEST RESULT AGAINST MAXIMUM
        SUBZ#   0,1,SZC     ;SKIP IF MAX<VALUE
        ISZ     SJD0        ;INCREMENT RETURN ADDRESS.
SJL3:   LDA     3,SJD0
        LDA     2,SJD0+1
        LDA     1,SJD0+2
        JMP     @SJD0                    ;RETURN
SJD0:   .BLK 4
SJD1:   0       ;LOOP COUNTER OF # OF DIGITS READ IN
SJD2:   060     ;MINIMUM CHARACTER
SJD3:   071     ;MAXIMUM CHARACTER
SJD4:   -4      ;COUNTER
SJD5:   0       ;TEMPORARY COUNTER
SJD6:   0       ;TEMPORARY LOCATION
SJD7:   177     ;CHARACTER MASK FOR ASCII TO OCTAL
        .END
```

```
        TIME
;THIS SUBROUTINE OBTAINS THE TIME IN THE FORMAT:
;       HOURS/MINUTES/SECONDS
;EACH FIELD MUST BE A TWO DIGIT INTEGER VALUE AND
;IS STORED AS A BINARY VALUE.
;AN AUTOINCRENTING LOCATION LABELED "TABLE"
;MUST EXST. THE TIME IS STORED INDIRECTLY
;THROUGH THIS LOCATION. "TABLE" MUST CONTAIN THE
;ADDRESS OF WHERE THE TIME IS TO BE STORED.
;
;INPUTS:NONE
;
;OUTPUTS:       NONE
;
;CALLING SEQUENCE
;       JSR     @TIME
;       RETURN
;
;
;SUPPORT SUBROUTINES
;       GETNUM
;       ERROR
;       TXTOT
;       CRLF
;
;
;CREATE POINTER TABLE ENTRY
        .ENT    TIME
        .EXTD   CRLF,TXTOT,GETNUM,ERROR,TABLE
        .ZREL
TIME:   SLL1
        .NREL
;
SLL1:   STA     3,SLD0      ;STORE THE ENVIRONMENT
        STA     2,SLD0+1
        STA     1,SLD0+2
        STA     0,SLD0+3
SLL2:   JSR     @CRLF
        JSR     @TXTOT      ;ASK FOR TIME INPUT
        SLT1
        LDA     0,SLD1      ;LOAD #      OF BCD DIGITS TO BE READ I

        LDA     1,SLD2      ;LOAD MAX OF HOURS
        JSR     @GETNUM     ;OBTAIN THE HOURS
        JMP     SLL3        ;RETURN HERE IF ERROR
        STA     0,@TABLE    ;STORE HOURS
        JSR     @TXTOT
        SLT2
        LDA     0,SLD1      ;LOAD # OF BCD DIGITS TO BE READ IN
        LDA     1,SLD3      ;LOAD MAX OF MINUTES
        JSR     @GETNUM     ;OBTAIN THE MINUTES
        JMP     SLL5        ;RETURN HERE IF ERROR
        STA     0,@TABLE    ;STORE MINUTES
        JSR     @TXTOT
        SLT2
        LDA     0,SLD1      ;LOAD # OF BCD DIGITS TO BE READ IN
        LDA     1,SLD3      ;LOAD MAX OF SECONDS
        JSR     @GETNUM     ;OBTAIN THE SECONDS
```

```
        JMP     SLL4        ;RETURN HERE IF ERROR
        STA     Ø,@TABLE    ;STORE SECONDS
        LDA     3,SLDØ      ;RESTORE ENVIRONMENT
        LDA     2,SLDØ+1
        LDA     1,SLDØ+2
        LDA     Ø,SLDØ+3
        JMP     @SLDØ       ;RETURN
SLL4:   DSZ     TABLE
SLL5:   DSZ     TABLE
SLL3:   JSR     @ERROR
        JMP     SLL2
;DATA
SLDØ:   .BLK    4           ;ENVIRONMENT STORAGE
SLD1:   2                   ;NO OF BCD CHARACTER TO BE OBTAINED
SLD2:   24.                 ;MAX HOURS
SLD3:   59.                 ;MAX SECONDS,MINUTES
SLT1:   .TXT    &TIME (HOURS/MINUTES/SECONDS)  :&
SLT2:   .TXT    */*
        .END
```

```
        STIME
;THIS SUBROUTINE OBTAINS THE TIME BETWEEN
;DATA AQUISITION SEQUENCES DESIRED BY THE
;USER.
;THE TIME IS A DOUBLE PRECISION
;BINARY VALUE. THE MODULO 60 OF THE HR,MIN,SEC ENTERED
;IS FOUND IN LOCATIONS STMEH,STMEL.
;
;INPUTS:NONE
;
;OUTPUTS:        NONE
;
;CALLING SEQUENCE
;        JSR     @STIME
;        RETURN
;
;
;SUPPORT SUBROUTINES
;        TXTOT
;        TIME
;        MULT,MULTA
;        CRLF
;AUTOINCREMENTING LOCATION "TABLE"
;AND LOW CORE LOCATIONS STMEH,STMEL MUST
;BE PROVIDED IN THE POINTER TABLE.
;
;CREATE POINTER TABLE ENTRY
        .ENT    STIME,HOURS,MIN,SEC
        .EXTD   CRLF,TXTOT,TIME,MULT,MULTA
        .EXTD   TABLE,STMEH,STMEL
        .ZREL
STIME:  SML1    ;SUBROUTINE TO READ TIME INTERVAL
HOURS:  Ø       ;STOREAGE FOR INTERVAL HOURS
MIN:    Ø       ;STORAGE FOR INTERVAL MINUTES
SEC:    Ø       ;STORAGE FOR INTERVAL SECONDS
        .NREL
;
SML1:   STA     3,SMDØ      ;STORE ENVIRONMENT
        STA     2,SMDØ+1
        STA     1,SMDØ+2
        STA     Ø,SMDØ+3
        LDA     Ø,TABLE     ;CHANGE TABLE TO
        STA     Ø,SMD1      ;          POINT
        LDA     Ø,SMD2      ;          TO
        STA     Ø,TABLE     ;LOCATIONS IN THIS SUBROUTINE
        JSR     @CRLF
        JSR     @TXTOT      ;OUTPUT MESSAGE
        SMT1
        JSR     @TIME       ;OBTAIN THE TIME
        LDA     Ø,SMD3+2    ;LOAD THE SECONDS
        STA     Ø,SEC
        LDA     1,SMD3+1    ;LOAD THE MINUTES
        STA     1,MIN
        LDA     2,SMD5      ;LOAD MIN MULTIPLIER OF 6Ø
        JSR     @MULTA      ;FORM SEC + (MIN*6Ø)
        MOV     1,Ø         ;MOVE LOW ORDER TO ACØ
        LDA     1,SMD3      ;LOAD HOURS
        STA     1,HOURS
```

```
        LDA     2,SMD4      ;LOAD HOURS MULTIPLIER OF 3600
        JSR     @MULTA      ;FORM (SEC + (MIN*60)) + (HRS*3600)
        STA     0,STMEH     ;STORE HIGH ORDER OF RESULT
        STA     1,STMEL     ;STORE LOW ORDER OF RESULT
        LDA     0,SMD1      ;RESTORE ORIGINAL POINTER IN
        STA     0,TABLE     ;LOCATION "TABLE"
        LDA     3,SMD0      ;RESTORE THE ENVIRONMENT
        LDA     2,SMD0+1
        LDA     1,SMD0+2
        LDA     0,SMD0+3
        JMP     @SMD0       ;RETURN
;DATA
SMD0:   .BLK    4           ;ENVIRONMENT STORAGE
SMD1:   0                   ;TEMPORARY STORAGE
SMD2:   SMD2                ;ADDRESS OF TIME STORAGE BLOCK - 1
SMD3:   .BLK    3           ;TIME STORAGE BLOCK
SMD4:   3600.               ;HOURS MULTIPLIER
SMD5:   60.                 ;MINUTES MULTIPLIER
SMT1:   .TXT    /DATA ACQUISITION SEQUENCE INTERVAL /
        .END
```

```
MULT  (AC1*AC2)
  ;NAME: MULTA   (AC0+(AC1*AC2))
  ;THIS SUBROUTINE MULTIPLIES THE SINGLE PRECISION BINARY
  ;VALUES IN AC1 AND AC2 TOGHETHER. THE RESULT IS RETURNED
  ;WITH HIGH ORDER IN AC0 AND LOW ORDER IN AC1.
  ;
  ;INPUT: AC1     MULTIPLIER
  ;       AC2     MULTIPLICAND
  ;
  ;OUTPUT AC0     HIGH ORDER OF RESULT
  ;       AC1     LOW ORDER OF RESULT
  ;
  ;NOTE THAT ENTRY AT SNL2 WILL
  ;     GIVE AC0 + (AC1*AC2).
  ;
  ;CALLING SEQUENCE
  ;       JSR     @MULT       ; (AC1*AC2)
  ;       RETURN
  ;  OR
  ;       JSR     @MULTA      ;AC0 + (AC1*AC2)
  ;       RETURN
  ;
  ;CREATE POINTER TABLE ENTRY
          .ENT    MULT,MULTA
          .ZREL
  MULT:   SNL1
  MULTA:  SNL2
          .NREL
  ;
  ;
  SNL1:   SUBC    0,0         ;CLEAR AC0, AND DO NOT DISTURB CARRY
  SNL2:   STA     3,SND0      ;STORE RETURN ADDRESS
          LDA     3,SND1      ;LOOP COUNTER INTO AC3
  SNL3:   MOVR    1,1,SNC     ;CHECK NEXT MULTIPLIER BIT
          MOVR    0,0,SKP     ;0,JUST SHIFT
          ADDZR   2,0         ;1,ADD MULTIPLICAND AND SHIFT
          INC     3,3,SZR     ;CHECK FOR 16TH TIME THROUGH
          JMP     SNL3        ;NO,CONTINUE
          MOVCR   1,1         ;YES,SHIFT LAST LOW BIT
          LDA     3,SND0      ;RESTORE AC3
          JMP     @SND0
  ;DATA
  SND0:   0           ;TEMPORARY LOCATION
  SND1:   -20         ;-16 IN DECIMAL, LOOP COUNTER
          .END
```

```
        ERROR
;THIS SUBROUTINE DECREMENTS THE ERROR COUNT
;A MAXIMUM OF N ERRORS CAN OCCUR.
;THE MAXIMUM IS STORED IN LOCATION
;"ERCNT"
;IF ERCNT=Ø THEN RETURN CONTROL TO DMS
;
;INPUTS:NONE
;
;OUTPUTS:         NONE
;
;CALLING SEQUENCE
;        JSR     @ERROR
;        RETURN
;
;SUPPORT SUBROUTINES
;        TXTOT
;        CRLF
;        RDMS
;        BELL
;
;CREATE POINTER TABLE ENTRY
        .ENT    ERROR
        .EXTD   TXTOT,CRLF,RDMS,BELL
        .ZREL
ERROR:  SOL1
ERCNT:  5       ;MAXIMUM NUMBER OF ENTRY ERRORS
        .NREL
;
SOL1:   STA     3,SODØ     ;STORE ENVIRONMENT
        DSZ     ERCNT
        JMP     .+5
        JSR     @CRLF
        JSR     @TXTOT
        SOT1               ;OUTPUT MESSAGE
        JMP     @RDMS      ;RETURN TO DMS
        JSR     @BELL      ;BLOW THE WHISTLE ONCE
        LDA     3,SODØ     ;RESTORE ENVIRONMENT
        JMP     @SODØ      ;RETURN
;DATA
SODØ:   Ø       ;ENVIRONMENT STORAGE
SOT1:   .TXT    /MAXIMUM NUMBER OF ENTRY ERRORS EXCEEDED/
        .END
```

```
        SAMP
;
;CALLING SEQUENCE:
;        JSR     @SAMP
;        RETURN
;
;INPUTS:NONE
;
;OUTPUTS:         NONE
;
;SUPPORT SUBROUTINES
;        CRLF
;        TXTOT
;        GETNUM
;
;CREATE POINTER TABLE ENTRY
        .ENT    SAMP
        .EXTD   CRLF,ERROR,TXTOT,GETNUM,RDMS,SAMNU
        .ZREL
SAMP:   SPL1
        .NREL
;
SPL1:   STA     3,SPDØ      ;STORE ENVIRONMENT
        STA     2,SPDØ+1
        STA     1,SPDØ+2
        STA     Ø,SPDØ+3
        JSR     @CRLF
        JSR     @TXTOT
        SPT1                ;ASK FOR # OF SAMPLES
        LDA     Ø,SPD1      ;LOAD # OF DIGITS
        LDA     1,SPD2      ;LOAD MAXIMUM VALUE
        JSR     @GETNUM     ;OBTAIN #OF SAMPLES
        JMP     SPL2        ;ERROR
        MOV     Ø,1,SNR     ;TEST IF WAS ZERO
        JMP     @RDMS       ;WAS ZERO,SO QUIT
        STA     1,SAMNU     ;STORE # OF SAMPLES
        LDA     3,SPDØ      ;RESTORE ENVIRONMENT
        LDA     2,SPDØ+1
        LDA     1,SPDØ+2
        LDA     Ø,SPDØ+3
        JMP     @SPDØ       ;RETURN
SPL2:   JSR     @ERROR      ;PROCESS ENTRY OR LIMIT ERROR
        JMP     .-2Ø
;DATA
SPDØ:   .BLK    4
SPD1:   3       ;NUMBER OF DIGITS
SPD2:   1ØØ.    ;MAXIMUM INPUT FOR # OF DATA ACQUISITION RECORDS
SPT1:   .TXT    &NUMBER OF SAMPLES DESIRED  :  &
        .END
```

```
;NAME:          DATAR
        ;THIS SUBROUTINE READS THE DATA FROM
        ;THE INTERFACE
                .ENT    DATAR
                .EXTD   TABLE
                .ZREL
        DATAR:  SRL1
                .NREL
        SRL1:   STA     3,SRDØ
                LDA     2,SRD2
                LDA     1,SRD1
                NIOC    67
                NIOC    63
                IORST
                NIOS    63
                SKPDN   63
                JMP     .-1
                DIAS    Ø,63
                STA     Ø,@TABLE
                NIOS    67
                SKPDN   67
                JMP     .-1
                DIAS    Ø,67
                STA     Ø,@TABLE
                INC     2,2,SZR
                JMP     .-5
                LDA     2,SRD2
                INC     1,1,SZR
                JMP     .-15
                JMP     @SRDØ
        SRDØ:   Ø
        SRD1:   -64.
        SRD2:   -5Ø.
                .END
```

## ACKNOWLEDGEMENTS

I would like to take this opportunity to thank the Kansas Water Resources Board which provided the grant under which this project was developed (4). Special thanks also go to my advisor Dr. Donald Hummels, Dr. Myron A. Calhoun, and (the late) Dr. Dale E. Kaufman for their assistance in the design phase of this project.

A special thanks is extended to my wife, Ellen, for her moral support and to Patricia Copeland and William Ryan for their assistance in the production of the printed circuit boards used in this project.

# THE DESIGN OF A COMPUTER INTERFACE FOR A RADAR PROCESSOR

by

CARL C. ANDREASEN

B.S. Kansas State University, 1974

AN ABSTRACT OF A MASTER'S THESIS

submitted in partial fulfillment of the
requirements for the degree

MASTERS OF SCIENCE

Department of Electrical Engineering

KANSAS STATE UNIVERSITY

Manhattan, Kansas

1976

## ABSTRACT

This thesis describes the design, implementation, and use of a special-purpose data acqisition interface which controls the transfer of binary coded decimal values representing the integral of the radar video signals in a sector of an annulus of side one-half mile and angle 5.625 degrees from the radar to the computer. This thesis is also intended to be used as an operating manual and to provide assistance for maintenance. Included is the theory of operation of the interface,a description of the interface signals that are generated from the other parts of the system and a description of how to use the associated interface control program.

A brief discussion is given on future uses of the data collected by this interface.