NOAA Technical Memorandum NWS WR-148

A REAL-TIME RADAR INTERFACE FOR AFOS

Mark Mathewson
Western Region Headquarters
Scientific Services Division
Salt Lake City, Utah
January 1980

UNITED STATES
DEPARTMENT OF COMMERCE
Juanita M. Kreps, Secretary

NATIONAL OCEANIC AND
ATMOSPHERIC ADMINISTRATION
Richard A. Frank, Administrator

National Weather
Service
Richard E. Hallgren, Director

This Technical Memorandum has been reviewed and is approved for publication by Scientific Services Division, Western Region.

L. W. Snellman, Chief
Scientific Services Division
Western Region Headquarters
Salt Lake City, Utah

TABLE OF CONTENTS

FIGURES

APPENDIX

A REAL-TIME RADAR INTERFACE FOR AFOS

Mark Mathewson
Scientific Services Divison

I. INTRODUCTION

Real-time radar data is very important in flash flood and severe thunderstorms situations. Without a computer monitoring the radar, the forecaster has to waste valuable time in either going to look at the radar or calling the radar operator on the phone. At some places, the forecaster just doesn't have time to check the radar and therefore, the forecasts are not as reliable as they could be. A prototype computer radar system has been developed to test the feasibility of an inexpensive real-time radar tool.

This technical memorandum describes the software interface between a WSR-74C weather radar and AFOS. By using a Nova312 mini-computer as the interface device, real-time displays can be made available to forecasters on AFOS.

The hardware interface between the WSR-74C radar and Nova312 has been described in detail. Schematics and waveform diagrams have been included. The software description is also supplemented by flowcharts and tables.

The appendix contains program listings, MDB interface board descriptions and schematics, and examples of AFOS-radar output.

II. HARDWARE

A. Hardware Description - General

Real-time radar displays can be made for AFOS using almost any Data General mini-computer with 16K of memory with the addition of a special radar interface board. For this experiment, a Nova312 mini-computer equipped with 16K of memory and dual floppy disk drives was used successfully. The radar interface board contains all the circuitry necessary to transfer the radar data, radar status indicators, and antenna position from the WSR-74C radar to the Nova series computer. The circuit resides on one Data General compatible general input/output board. The board was purchased with the register and data channel options from MDB Systems Inc. for an approximate price of $800. Some additional circuitry was necessary to interface the radar to the MDB board. The technical specifications and schematics for the MDB board have been included in the appendix.

About $50 of components were added to the board to interface the radar. A radar simulator circuit was also constructed on the board to provide

FIGURE 1 RADAR - AFOS INTERFACE

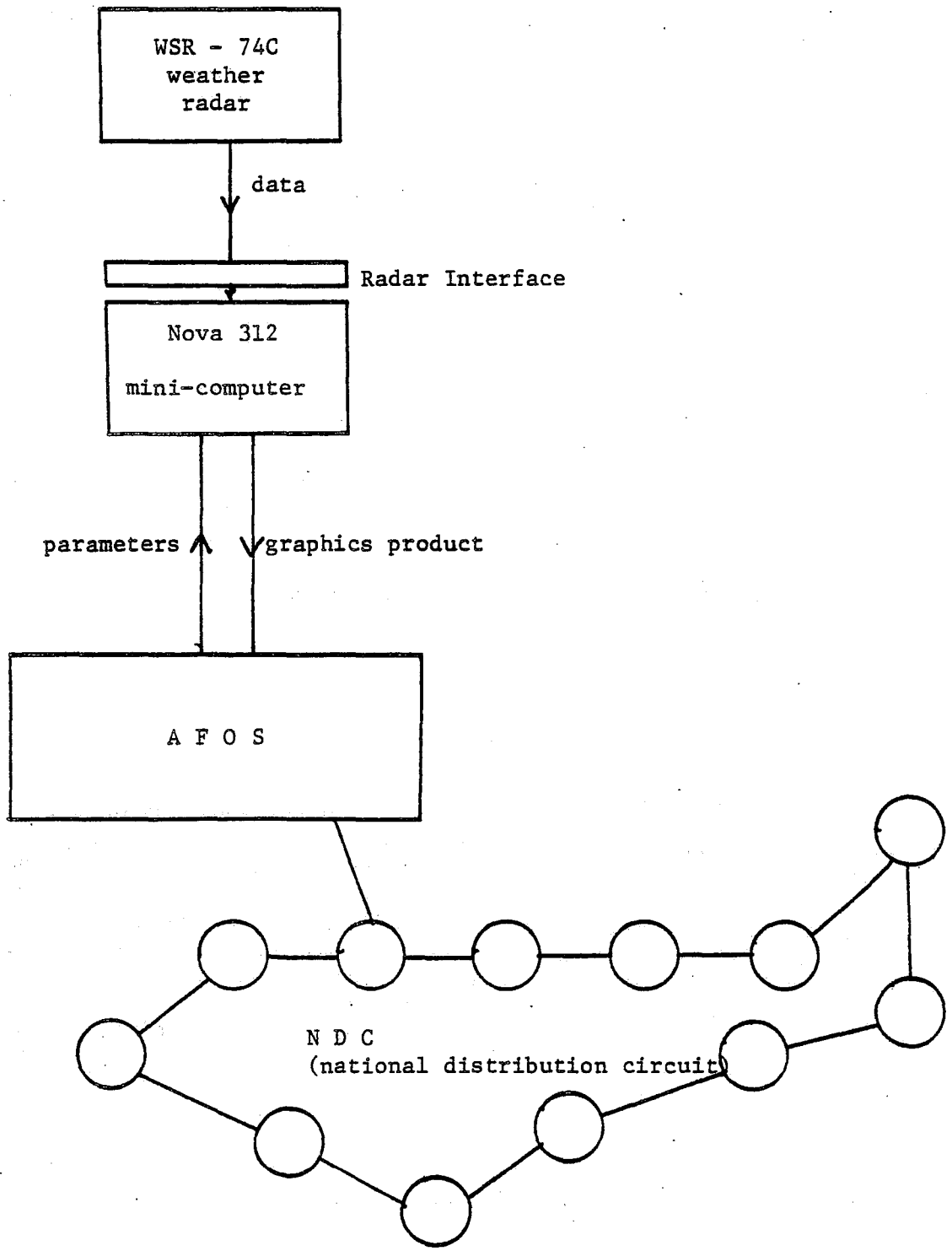


FIGURE 2 -- RADAR COMPUTER HARDWARE

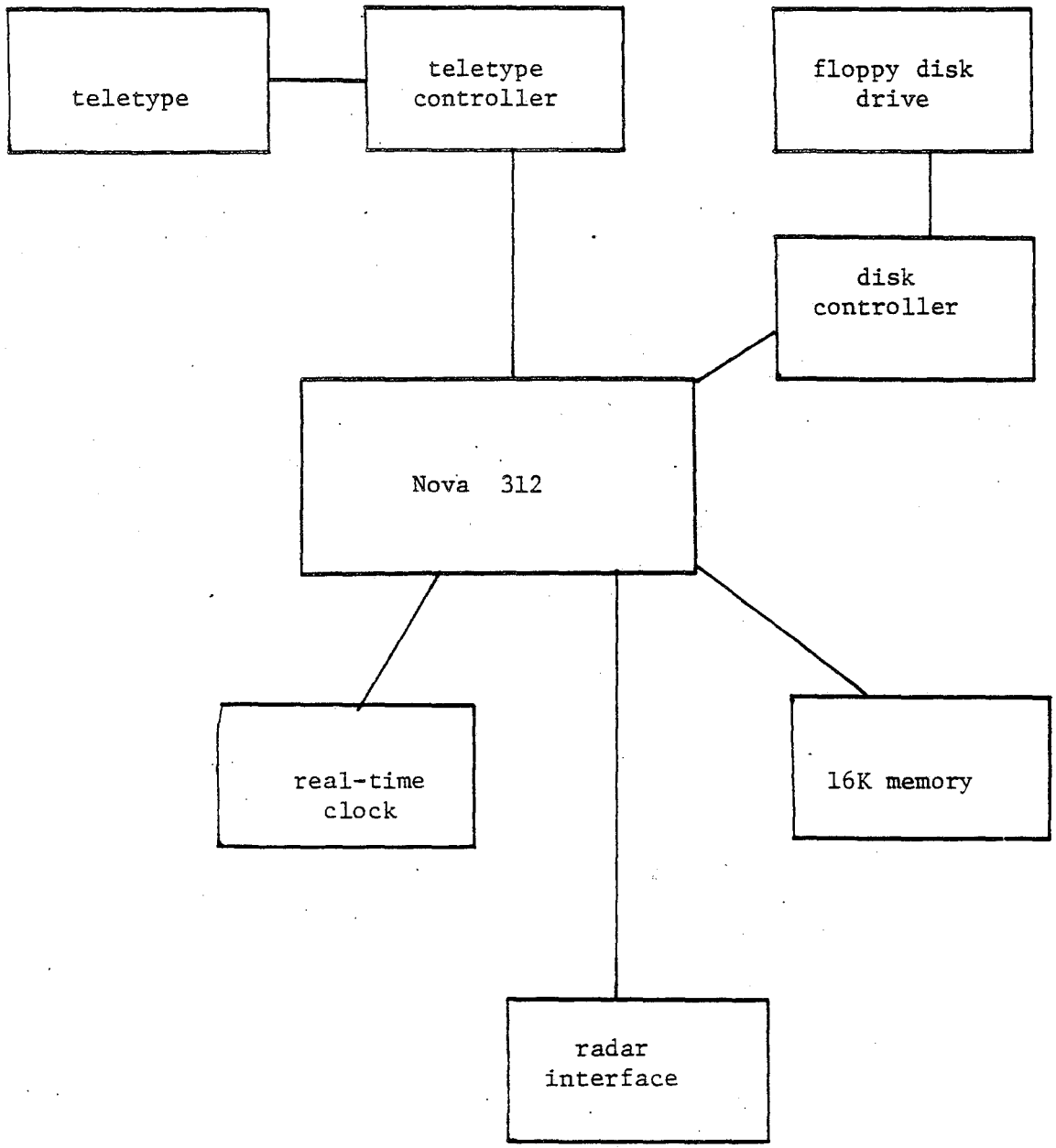


FIGURE 3 RADAR INTERFACE BOARD INSTRUCTION SET

Device Code = 25 octal
Mask Out Bit = 1

Input/Output Instructions:

DIA	reads azimuth
DIB	reads status and elevation
DIC	reads memory address register
DOA	loads memory address register
DOB	loads word count register
DOC	loads output register bit 15=0 xmit interrupts disabled bit 15=1 xmit interrupts enabled

bit values for DIA and DIB:

bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
azm.	-	-	200	100	80	40	20	10	8	4	2	1	.8	.4	.2	.1
elev.	-	-	-	-	-	40	20	10	8	4	2	1	.8	.4	.2	.1
stat	a	b	c	d	e	-	-	-	-	-	-	-	-	-	-	-

where: a=Test Mode Monitor
b=IF Attenuator Monitor
c=Range Interval Monitor
d=Time Sample Monitor
e=STC Monitor

Command Instructions:

S	starts DMA transfer
C	clears interrupt request and stops DMA transfer
P	disables xmit interrupts

test signals during the hardware/software development stage. Radar intensity data is transferred using the processor-independent direct-memory-access method and the antenna and status information is transferred by the conventional processor-controlled input/output routines.

Two 50-pin ribbon connectors are available on the board and have been wired for radar input and simulator output. By installing a jumper cable between the two connectors, the system can be tested using the simulator.

B. WSR-74C Radar Signals

The Enterprise WSR-74C radar incorporates a digital video integrator processor (DVIP) which converts the radar return (from the form of video) into a digitized format compatible with computer systems. The DVIP incorporates a intricate scheme to average the data both over range and azimuth to effectively reduce the noise to better the 2db. This integrated output is available as a eight-bit word at a connector on the rear panel of the radar.

The DVIP also has several control signals available to the user. One of these is the data ready line. It informs the computer that the data appearing on the data lines is valid. The DVIP is capable of operating with either range gates of one or two kilometers (one or two kilometer resolution) selectable by a front panel switch. Hence, new data will appear every one or two kilometers on the data lines and the data ready line will inform the user when to read the data. The DVIP outputs data between the ranges of 21 kilometers and 450 kilometers.

The master sync line is also available from the radar, also called the transmit pulse, it synchronizes the radar system timers which convert echo return time into range. It can also be used to indicate the beginning of a new data scan. The transmit pulse rate is approximately 259 hz.

Other control signals from the DVIP are the switch monitors: IF Attenuators, STC, Time Sample, Test Mode, and Range Interval. These lines simply indicate the positioning of the switches on the radar.

The antenna position signals are derived from two separate but identical units; the azimuth from the PPI section, and the elevation from the RHI section. Each unit contains a synchro-to-digital converter which senses the antenna position by using the synchro lines and outputs the results as a 14-bit BCD word.

In summary, the following signals are available for an external user:

8 data lines
data ready line
master sync line (xmit)
STC monitor
Range Interval monitor
Test Mode monitor

FIGURE 4 WSR-74C SIGNALS

Bit values for BCD antenna information

Bit	Value	Bit	Value
0	.1 deg.	7	8 deg.
1	.2 deg.	8	10 deg.
2	.4 deg.	9	20 deg.
3	.8 deg.	10	40 deg.
4	1.0 deg.	11	80 deg.
5	2.0 deg.	12	100 deg.
6	4.0 deg.	13	200 deg.

Signal levels for monitor functions

Signal	Signal at OV	Signal at 5V
STC	Off	On
Time Sample	15	31
Test Mode	On	Off
Range Interval	2 km.	1 km.
IF Attenuators	Off	On
Data	0	1
Antenna Pos.	0	1

Signal pin connections on rear panel connector (1J2 WSR74-C)

Pin	Signal	Pin	Signal	Pin	Signal
A	shield gnd	V	elevation 100	p	data return 3
B	azimuth .1	W	elevation 80	q	data 3
C	azimuth .2	X	elevation 40	r	data return 4
D	azimuth .4	Y	elevation 20	s	data 4
E	azimuth .8	Z	elevation 10	t	data return
F	azimuth 1	a	elevation 8	u	data 5
G	azimuth 2	b	elevation 4	v	data return 6
H	azimuth 4	c	elevation 2	w	data return 6
J	azimuth 8	d	elevation 1	x	data return 7
K	azimuth 10	e	elevation .8	y	data 7
L	azimuth 20	f	eelvation .4	z	data 8
M	azimuth 40	g	elevation .2	aa	data return 8
N	azimuth 80	h	elevation .1	bb	data ready out
P	azimuth 100	hh	converter busy - elev	cc	time sample
R	azimuth 200	j	data return 1	dd	range interv
S	azimuth data gnd	k	data 1 (1st)	ee	stc monitor
T	elevation data ret.	m	data return 2	ff	test mode monitor
U	elevation 200	n	data 2	I	IF attn bypass
				gg	converter busy azimuth

Time Sample monitor
IF Attenuator monitor
14 Azimuth position lines
14 Elevation position lines

All signals except for the xmit line have TTL open-collector outputs and require pull-up resistors. The xmit line is not TTL and requires voltage reduction to interface with TTL logic.

C. Radar Simulator

A radar simulator was constructed on the radar interface board to assist in hardware/software development. The simulator consists of a series of integrated circuit timers used to duplicate the radar output signals from the WSR-74C radar.

The simulator consists of two parts: the antenna signal generator and the data signal generator. The antenna portion simulates only the azimuth portion; however, the azimuth and elevation lines are paralled for test purposes. The heart of the system is the Sylvania 978 dual timer integrated circuit. The 978 is a very versatile timing chip capable of providing monostable (one-shot) and astable (free-running) pulse trains ranging from periods of nanoseconds to minutes.

The 978 for the antenna simulator is connected in an astable configuration providing a continuous stream of pulses at a frequency of 180 hz. The square wave output is input to a string of four BCD (binary-coded decimal) counters which have an ultimate capacity of 9999. However, a four-input AND gate is connected such that the counters will reset to zero whenever 3600 is reached. (3600=360 degrees) Each counter functions as either the .1, 1., 10, or 100 degree digit. The frequency of 180 hz. is equivalent to 18 degrees/second antenna rotation rate, or three rpm which is the standard radar rotation rate.

The other portion of the simulator provides the data signals. It uses the same type of timer (Sylvania 978) as the antenna simulator. Three timers are used, two in the astable mode and one in the monostable mode. The timers are labeled the xmit, delay, and data ready timers.

The xmit astable in conjunction with an inverter produces a xmit pulse which duplicates the WSR-74C master synchronization pulse. It has a repetition frequency of approximately 250 hz which is similar to the radar PRF (pulse repetition frequency) of 259 hz.

The data ready astable produces the data ready pulses which also is very similar to the radar data ready pulse. The cycle time for this timer is 6.6 microseconds. In addition, this timer drives an 8-bit counter used for providing a test ramp of data.

The monostable delay timer is triggered by the xmit astable 250 times per second and forces the data ready pusles to stop for 120 microseconds. This delay accounts for the absence of data for the first 20 kilometers.

FIGURE 5 WSR-74C RADAR WAVEFORMS

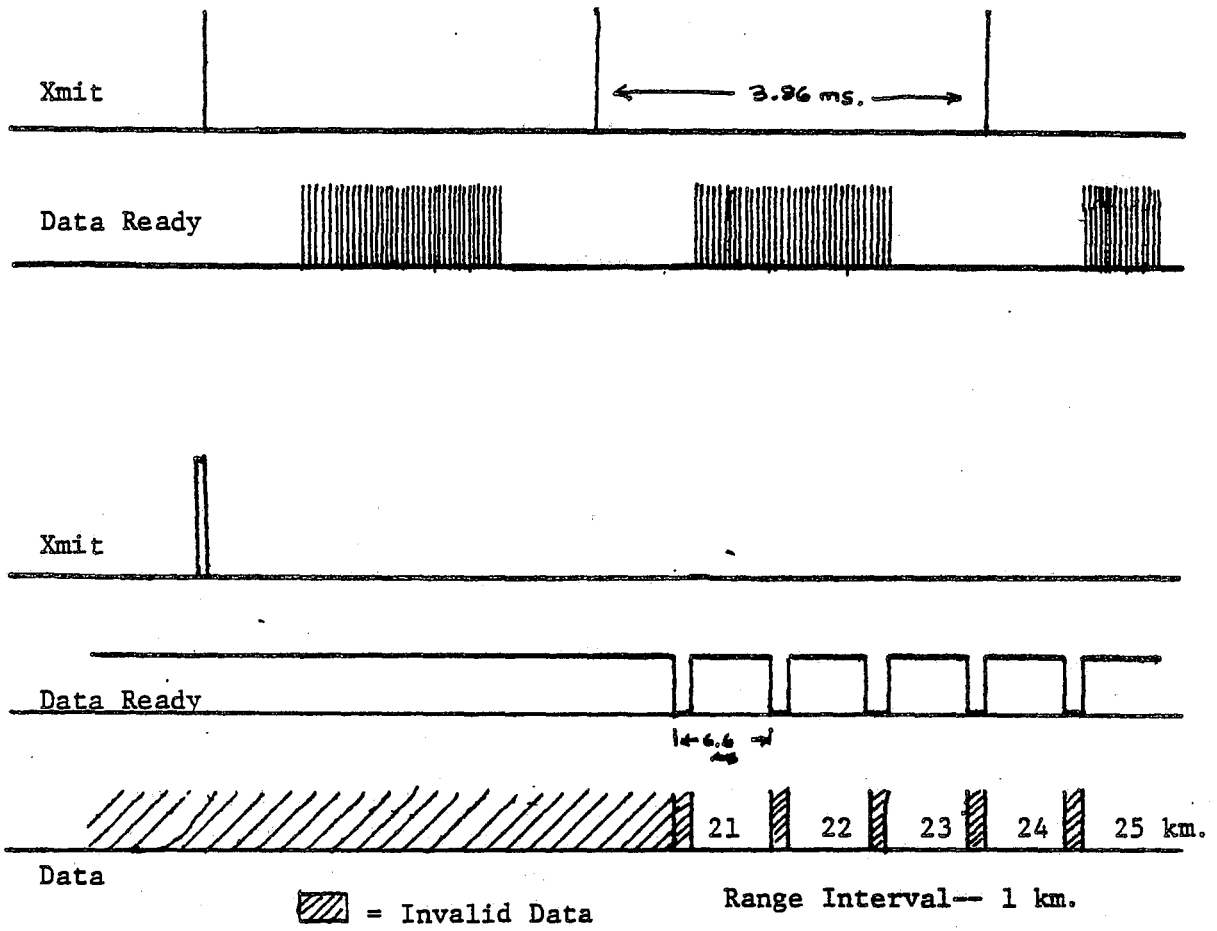
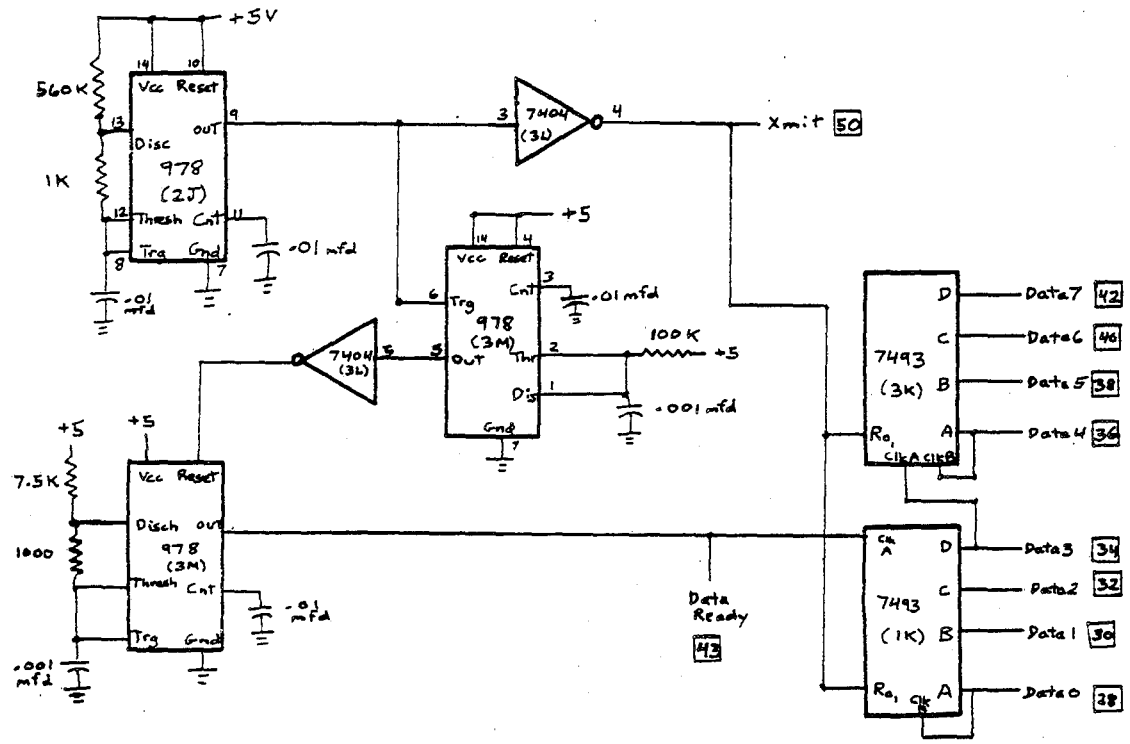


FIGURE 6 RADAR SIMULATOR - DATA PORTION



Data Simulator Waveforms

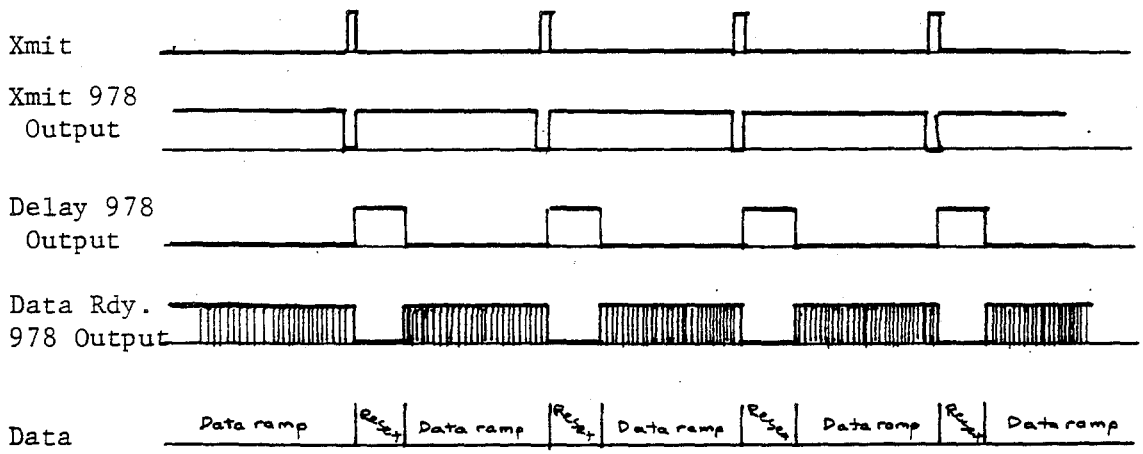
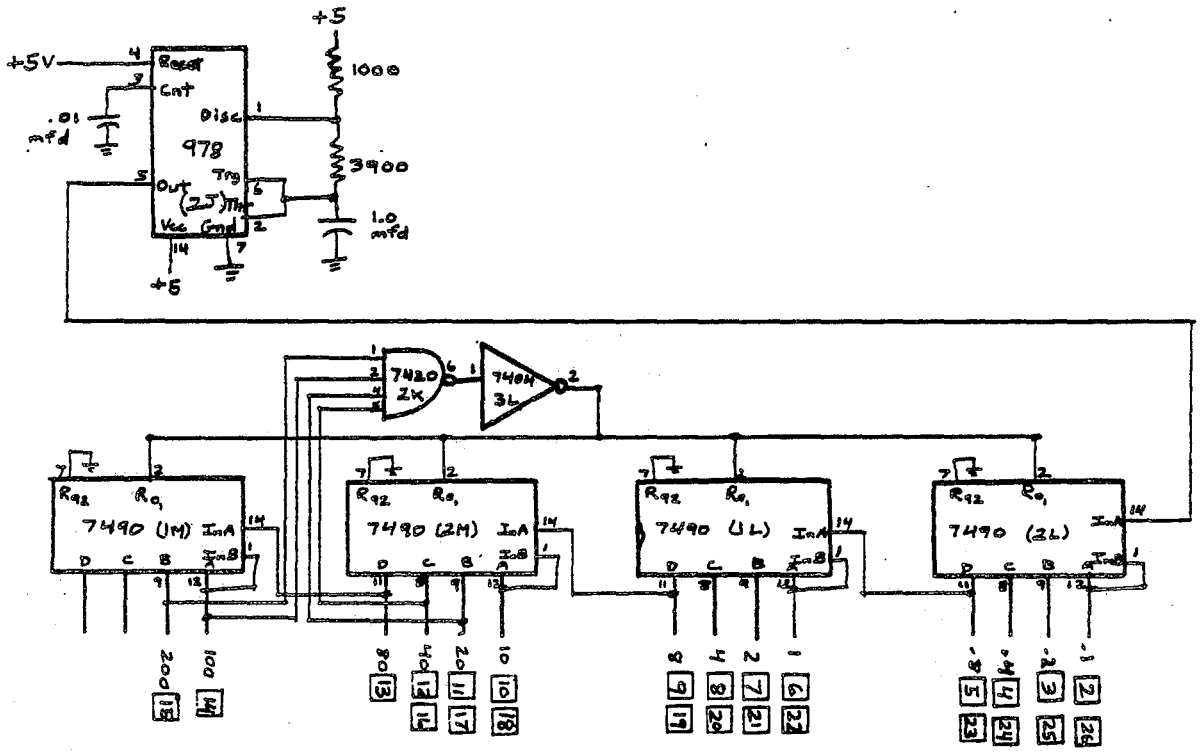
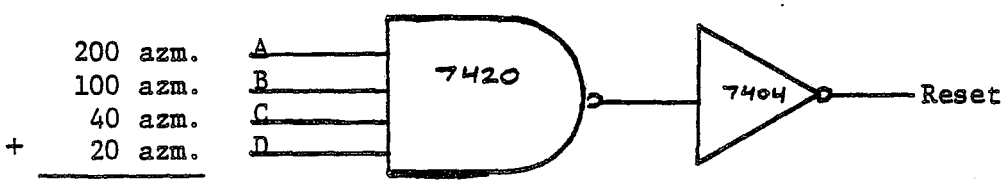
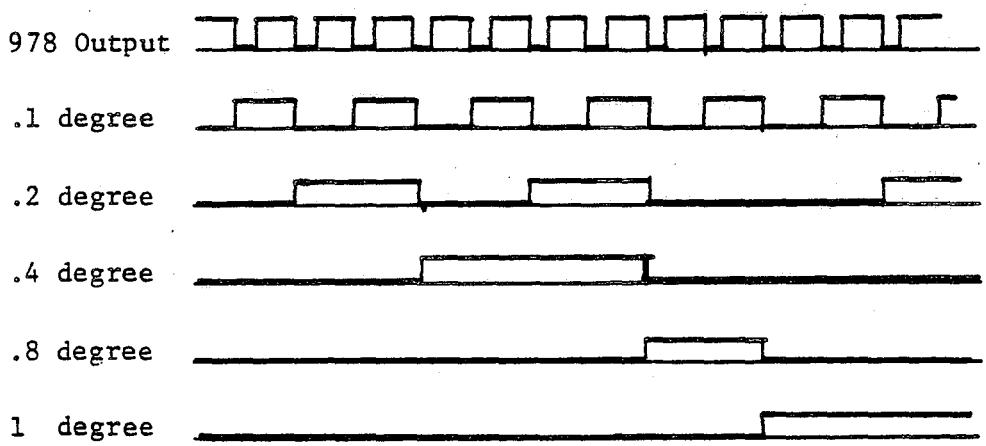


FIGURE 7 RADAR ANTENNA SIMULATOR





Reset Circuit for 3600 to zero
(lines A,B,C,and D must be high for reset)



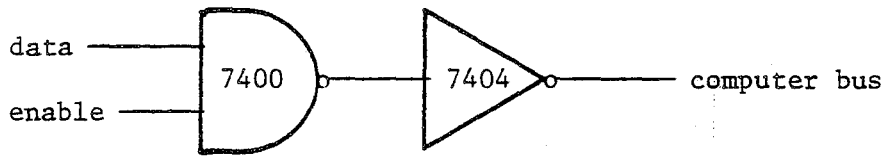
Antenna Simulator Waveforms

The combination of the three timers and the eight-bit timer with the antenna simulator provide all the necessary signals for hardware/software program development without the need of an operational WSR-74C radar.

A 50-pin ribbon connector is installed on the interface board allowing the simulator signals to be connected to the computer via a ribbon jumper cable.

D. Radar Interface - Antenna and Status

The antenna position and status information are read into the computer in the conventional processor-controlled mode by issuing input/output instructions. The interface board responds to the command DIA ac,25 (ac=accumulator 0, 1, 2, or 3) to obtain the azimuth information and to the command DIB ac,25 to obtain the elevation and status information. Each input line from the radar is connected to a 610 ohm pull-up resistor and one input of a two-input NAND gate. The second input to the NAND gate (called the enable input) is connected to the MDB board control line DIA (data in-register A) or DIB (data in-register B).



When the enable input goes high (becomes active) by executing the appropriate instruction, the data on the radar line is put on the computer bus and stored in one of the processor accumulators.

Sixteen two-input NAND gates form a 16-bit register and two registers are used; one for the azimuth information and one for the elevation and status information.

E. Radar Interface - Interrupts

The interrupt control on the interface board has been wired to allow for two different events to cause an interrupt. The first is a transmit interrupt (XMIT) and the second is a DMA transfer finished (DONE) interrupt.

For either event, an interrupt request is generated when pin X9 on the interface board goes high. This signal is latched and an interrupt is generated when the computer is ready. The DONE interrupt request is generated when the MDB Done flip-flop is set. In the radar design, the Done flip-flop can only be set when the DMA transfer has completed. The DONE interrupt is used to inform the collection program that data has been received and processing may begin.

The XMIT interrupt request is generated when a transmit signal has been received and XMIT interrupt enable is active. The transmit signal is always being received at 259 hz when the radar is operating. Therefore, XMIT interrupts can only be generated by enabling them by setting bit 15 to one in the output register. The instruction DOC ac,25 can be used to enable

FIGURE 8 AZIMUTH, ELEVATION, AND STATUS INTERFACE

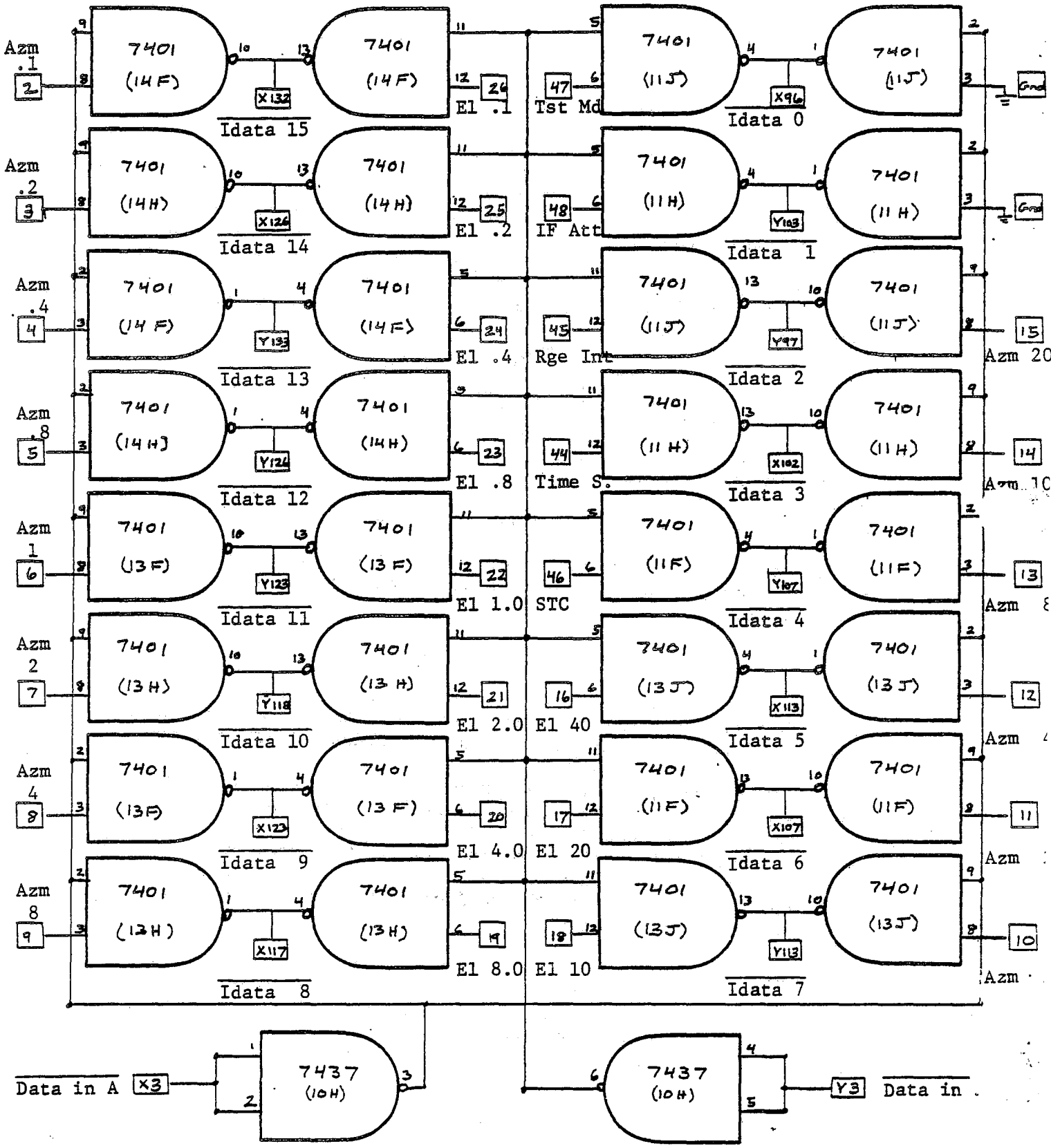
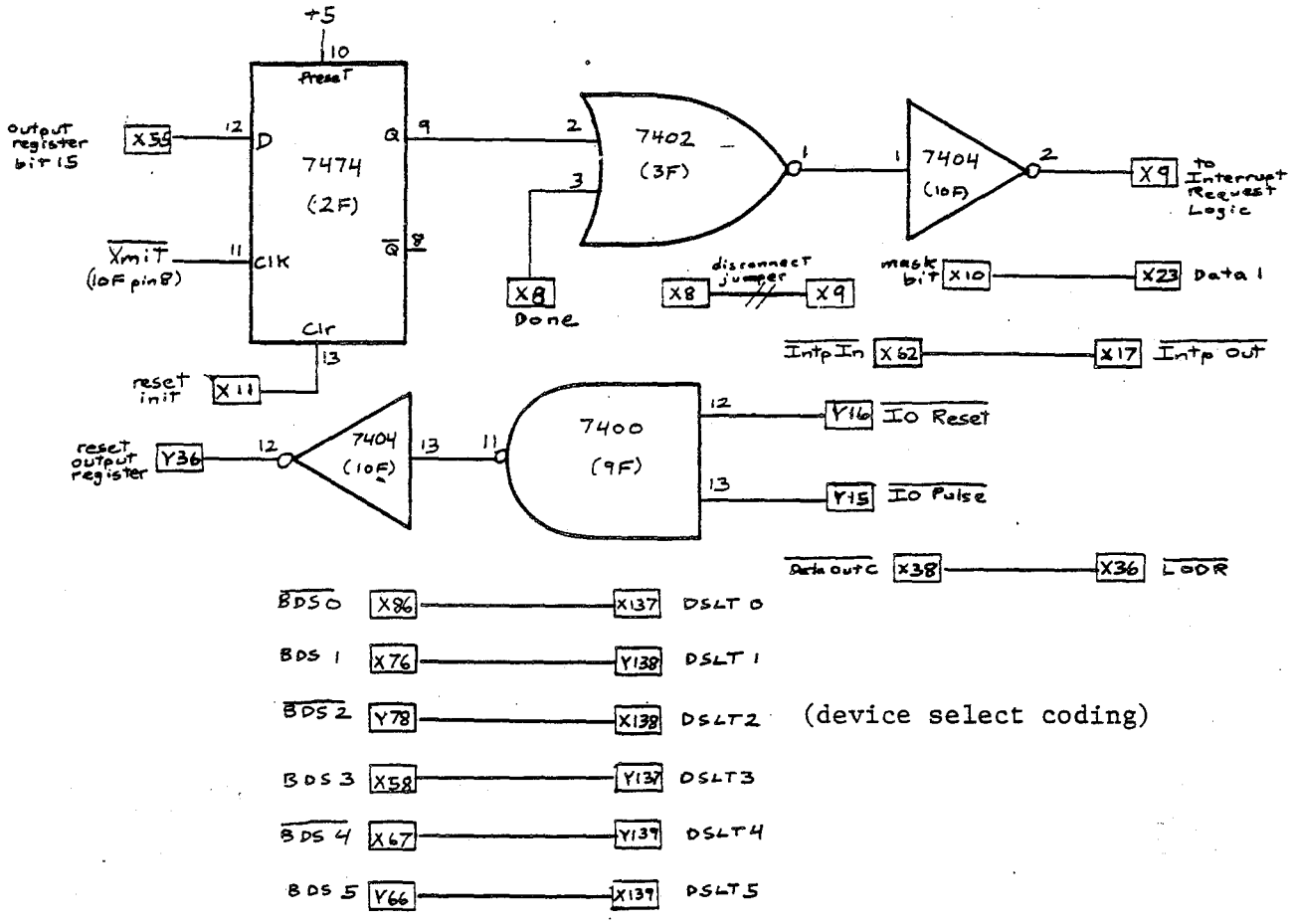
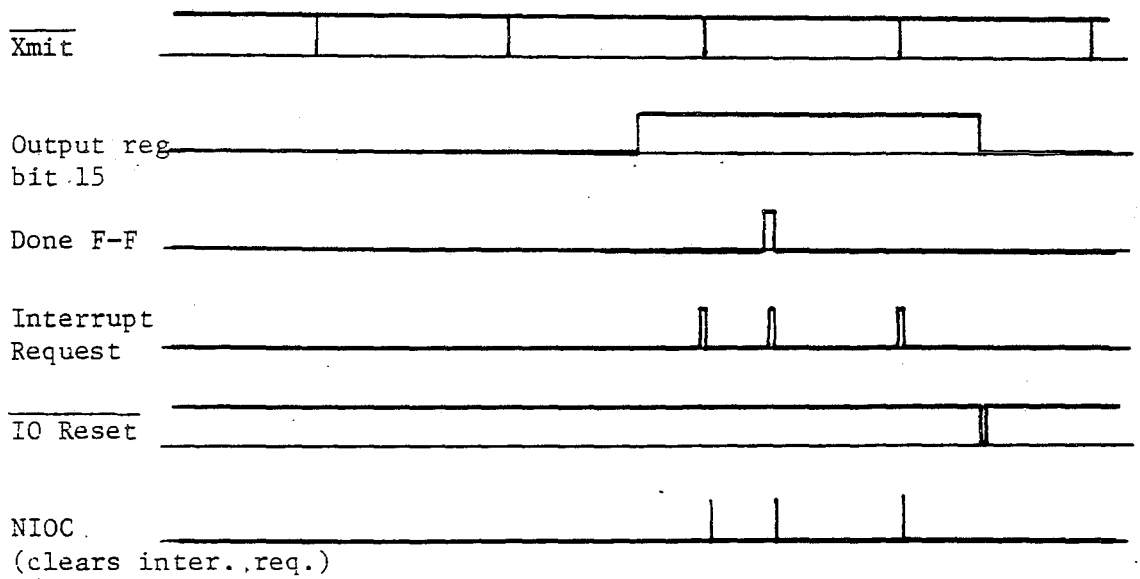


FIGURE 9 INTERRUPT REQUEST LOGIC



(device select coding)



interrupts and either a RESET or NIOP can be used to disable XMIT interrupts. The XMIT interrupt is used to synchronize the radar collection program.

F. Radar Interface - Byte Packing Logic

The Nova312 computer works on 16-bit data words and the radar uses 8-bit data words. In order to maximize the efficiency of the computer, it is necessary to incorporate byte packing logic. The logic reads in two 8-bit words from the radar and stores them as one 16-bit word. The 8-bit inputs are connected in parallel to both the low and high order bytes of the MDB input register. Hence, by activating either the load low (Y130) or load high (X110) line, either byte can be loaded with the 8-bit radar data. The radar generated data ready line clocks a J-K flip-flop to alternatively load each byte. The XMIT pulse resets the flip-flop to assure correct loading of the first byte. The two-input NAND gates alternatively activate the load lines when data is ready to be transferred. The CLK DSYNC (X135) signal is generated at the $\overline{Q}$ output of the flip-flop. This signal requests a DMA data transfer (data channel request) after every two bytes are loaded.

G. Radar Interface - Data Channel

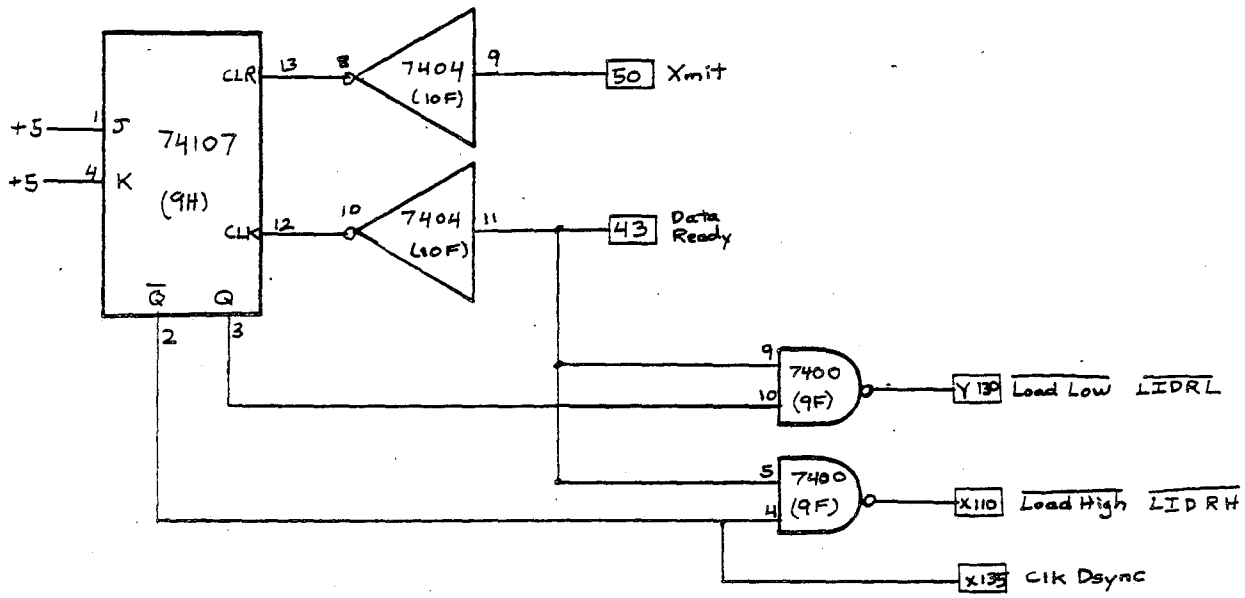
The data channel mode is used to transfer the radar intensity data to the computer memory without any processor intervention. The radar data arrives at the interface board every 6.6 microseconds making it impossible to sample data by the conventional means of input/output instructions. To initiate a data channel transfer, the programmer must load the word count register with the number of words to be transferred, the memory address register with the starting location of the transfer, and set the start flag. The data is automatically transferred word-by-word as the CLK DSYNC (X135) line is pulsed.

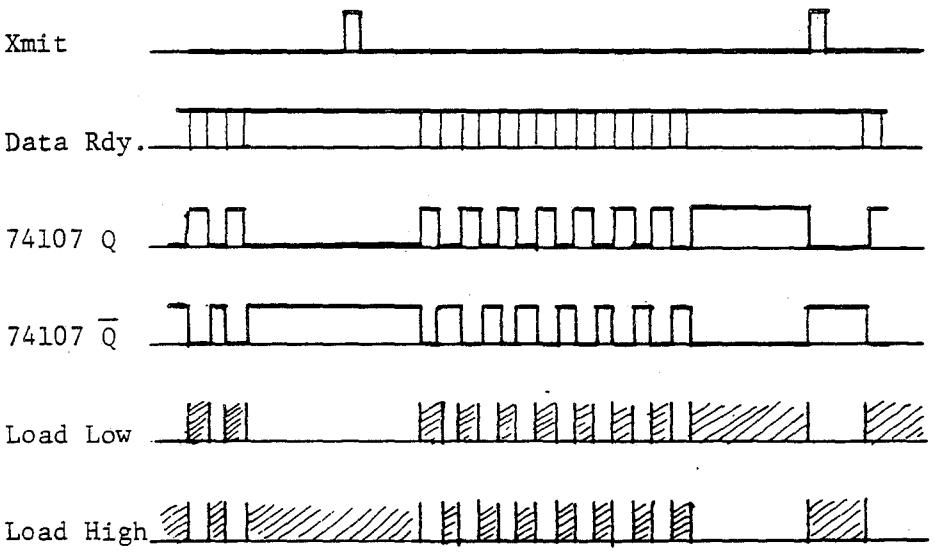
The memory address register is wired to increment by one after each word transfer. Hence, the data is written into sequential memory locations as the transfer progresses.

The word counter register is also wired to increment after each transfer. A zero count detector is used to detect when the register is equal to zero (transfer is finished). The zero count detector terminates all further data channel requests by clearing the BUSY and setting the DONE flip-flops. This action also generates an interrupt request.

Data channel transfer requests are only honored when the BUSY flip-flop is set. It can be set by sending the "S" (Start) instruction. Termination of data channel requests occur when the word count register reaches zero.

FIGURE 10 BYTE PACKING LOGIC





▨ = load data

FIGURE 11 DATA CHANNEL CIRCUIT

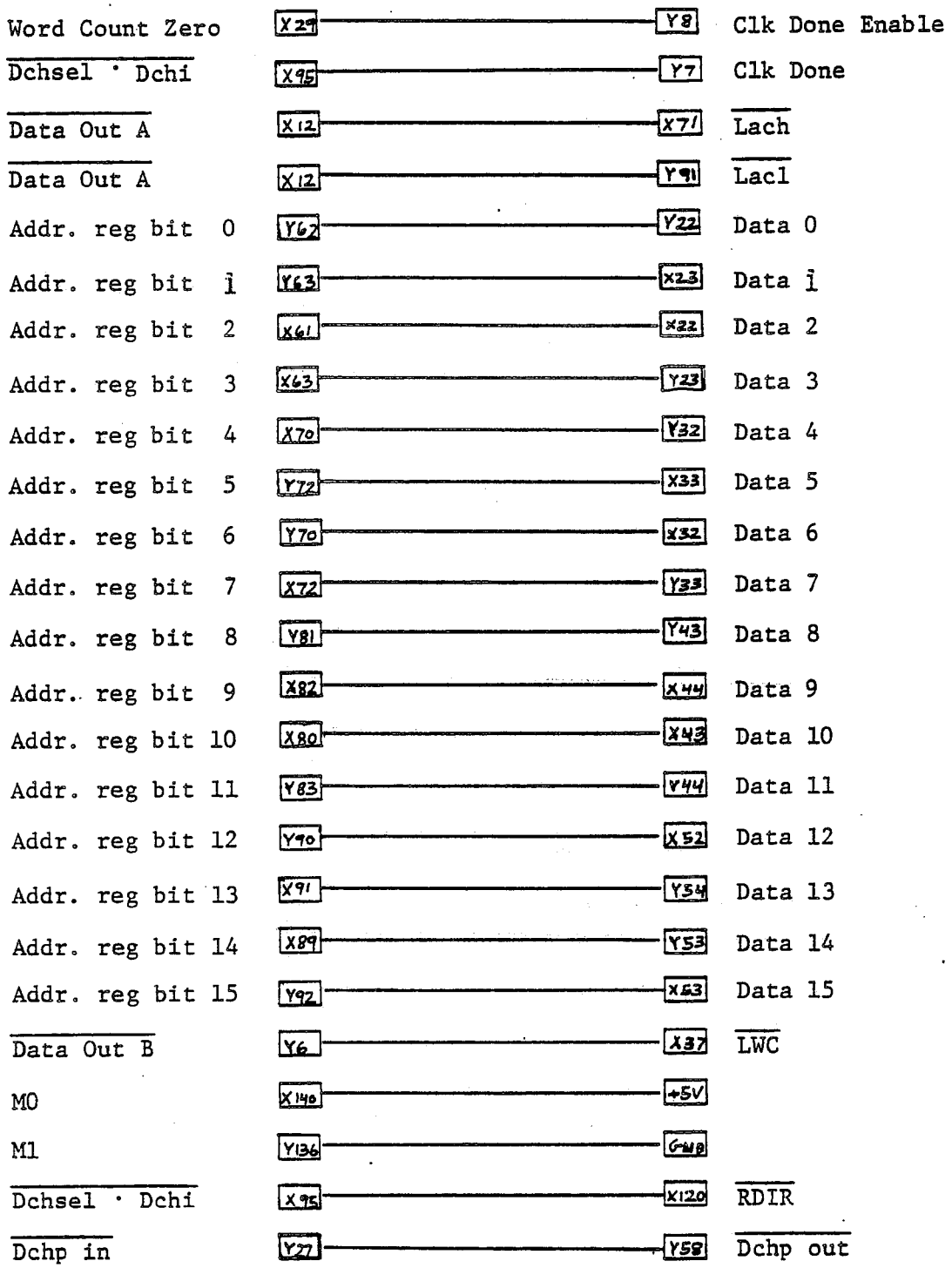
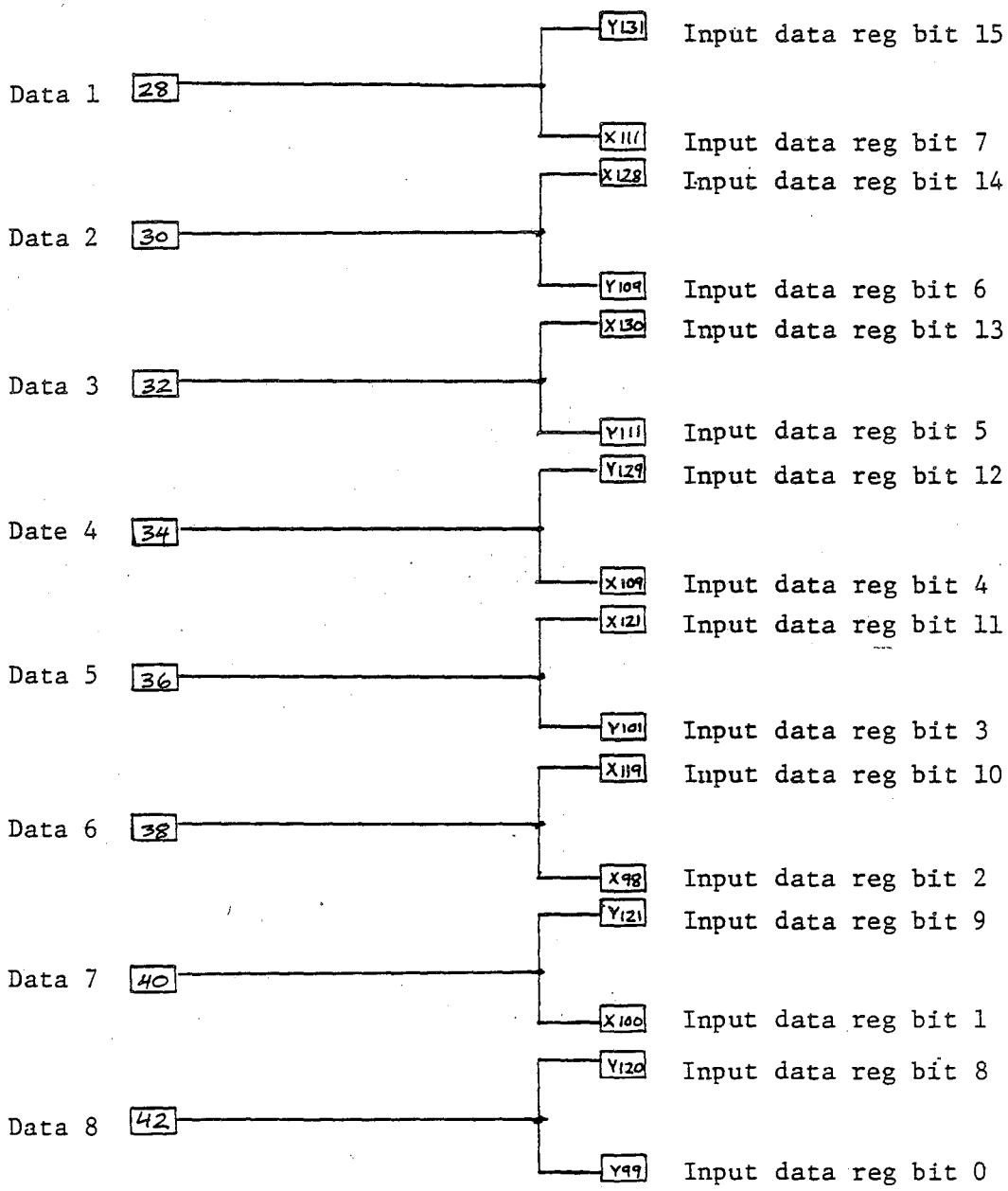


FIGURE 12 INPUT DATA REGISTER CIRCUIT



Note: Radar data 1 is LSB (least significant bit)
DCC idata 15 is LSB

III. SOFTWARE

A. Software - Overview

A series of DGC Assembler and DGC Fortran IV programs have been developed to collect radar data, process the data, and to display the data. In the present version, the programs produce a file of AFOS compatible graphic instructions that display radar intensities on a grid using alpha-numeric characters. A county map background centered around the radar site is also displayed. The various programs are summarized below:

1. Radarlog - An assembler program which collects the raw radar data in polar coordinates and stores it on floppy disk.

2. Grid - A Fortran program which converts the collected radar data into rectangular coordinates and stores it in a grid format. The grid is stored as four quadrants on floppy disk.

3. Graph - A Fortran program which converts the gridded data into AFOS graphics commands and stores them on floppy disk.

4. Parameter - A Fortran program which produces a parameter file. The user has control over items such as resolution, range blanking, threshold levels, etc. This program is only run when parameters must be changed.

5. Control - A Fortran control program which automatically calls the collection, grid, and graph programs at a parameter specified interval.

6. Mapmake - A Fortran program designed to be run on a IBM 360/168 computer. The program produces a file of coordinates for the county map background. A second Fortran program runs on the DGC and converts the coordinates into graphic instructions.

In operational status, the operator would run the control program. The system would automatically schedule the observations and run the grid and graph programs. The control program also sends the final product to AFOS by means of a teletype line.

B. Software - Radarlog

The radarlog program's main task is to collect one revolution of radar data and store it on floppy disk. The data is stored in polar coordinates in file radardata. File radardata is a contiguous file containing 180 blocks. The program is organized in three separate sections: variable declaration and initialization, program code, and data buffers. The variable declaration and initialization portion contain the buffer pointers, constants, and collection parameters. The data buffer section contain the 4096 word data buffer area.

FIGURE 13 RADARDATA FILE FORMAT - VERSION 1.0

Each block has the same format:
File is 180 blocks in length.

Location in block Octal	Decimal	Parameter
0	0	Azimuth (BCD) in tenths
1	1	Status--Elevation (BCD)
2	2	Month
3	3	Day
4	4	Hour
5	5	Minute
6	6	0
7	7	0
10	8	Data--Bin 0,1
11	9	Data--Bin 2,3
12	10	Data--Bin 4,5
.	.	.
.	.	.
.	.	.
163	115	Data--Last Bin
164	116	0
165	117	0
.	.	.
.	.	.
177	127	0

Second portion of block repeats same format as first portion.

Azimuth:	-	-	200	100	80	40	20	10	8	4	2	1	.8	.4	.2	.1
Bit # :	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Elev :	-	-	-	-	-	40	20	10	8	4	2	1	.8	.4	.2	.1
Status :	a	b	c	d	e											

a=test mode monitor 0=on,1=off
b=if attenuators 0=off,1=on
c=range interval 0=2km,1=1km
d=time sample 0=15,1=31
e=stc monitor 0=off,1=on

The radarlog program starts executing at label "START". Refer to the flow charts for this program in the text and the program listing in the appendix. The present time and date is obtained from system calls and stored in memory. The stack pointer and frame pointer are initialized, and the xmit interrupts disabled. The radardata file is opened on channel three and the disk heads are positioned at the start of file. Certain collection variables are initialized and all buffers are loaded with the date and time header.

Since the radar interrupt board was not sysgened and would be impossible to do such, the program next executes the .IDEF command to inform the system to accept radar interrupts. No radar interrupts occur at the present time since xmit interrupts have been disabled (output register bit 15 is not set). The program next waits in a loop labeled "AZO" waiting until the antenna azimuth (obtained via a DIA 0,25 command) is equal to the start collection azimuth (which is zero degrees). At this point, the program enables XMIT radar interrupts to occur by setting bit 15 of the output register with a DOC 0,25 command. The remainder of the collection program is timed by interrupts. The main portion is put into a short routine labeled "WAIT". Usually this routine simply jumps to itself and does nothing. However, branches can be taken to a disk write routine or to the exit routine depending on the contents of variable Flag. The variable Flag is controlled by the interrupt scheme.

On radar interrupt, the system branches to the radar service routine labeled "RADAR". The done flag on the interface board is interrogated by a SKPDN 25 instruction to determine whether the interrupt is caused by a XMIT or DONE condition.

In most cases, the interrupt was caused by a XMIT pulse. In this case, the service routine jumps to label "XMIT" and servicing continues. The present azimuth is read in and compared to the next azimuth desired. If the present azimuth is less than the desired observation azimuth, the interrupt scheme exits. If not, the memory address is calculated and loaded, the word counter is loaded, and the DMA transfer is begun. During the DMA transfer,the azimuth, status, and elevation are stored in the correct data buffer, and the service routine exits. When the DMA transfer is finished, the DONE flag is activated and an interrupt requested.

If the interrupt was caused by a DONE condition, the DMA process has finished. The program jumps to label "DONE" and servicing commences. The next desired azimuth is determined and converted to BCD for comparison purposes. If the next azimuth is greater than 360 degrees, the interrupt scheme exits to the "FINIl" routine. Every sixteenth DONE interrupt causes the service routine to exit to the write-data-to-disk routine labeled "CONT" and "WRT".

Each data scan requires a data buffer of 128 words or 1/2 disk block. The disk write routine writes eight blocks at a time consisting of sixteen data scans (every 16 degrees). By writing such a large block of data, system overhead is substantially reduced. The "WRT" routine determines

FIGURE 14 RADARLOG FLOW CHART

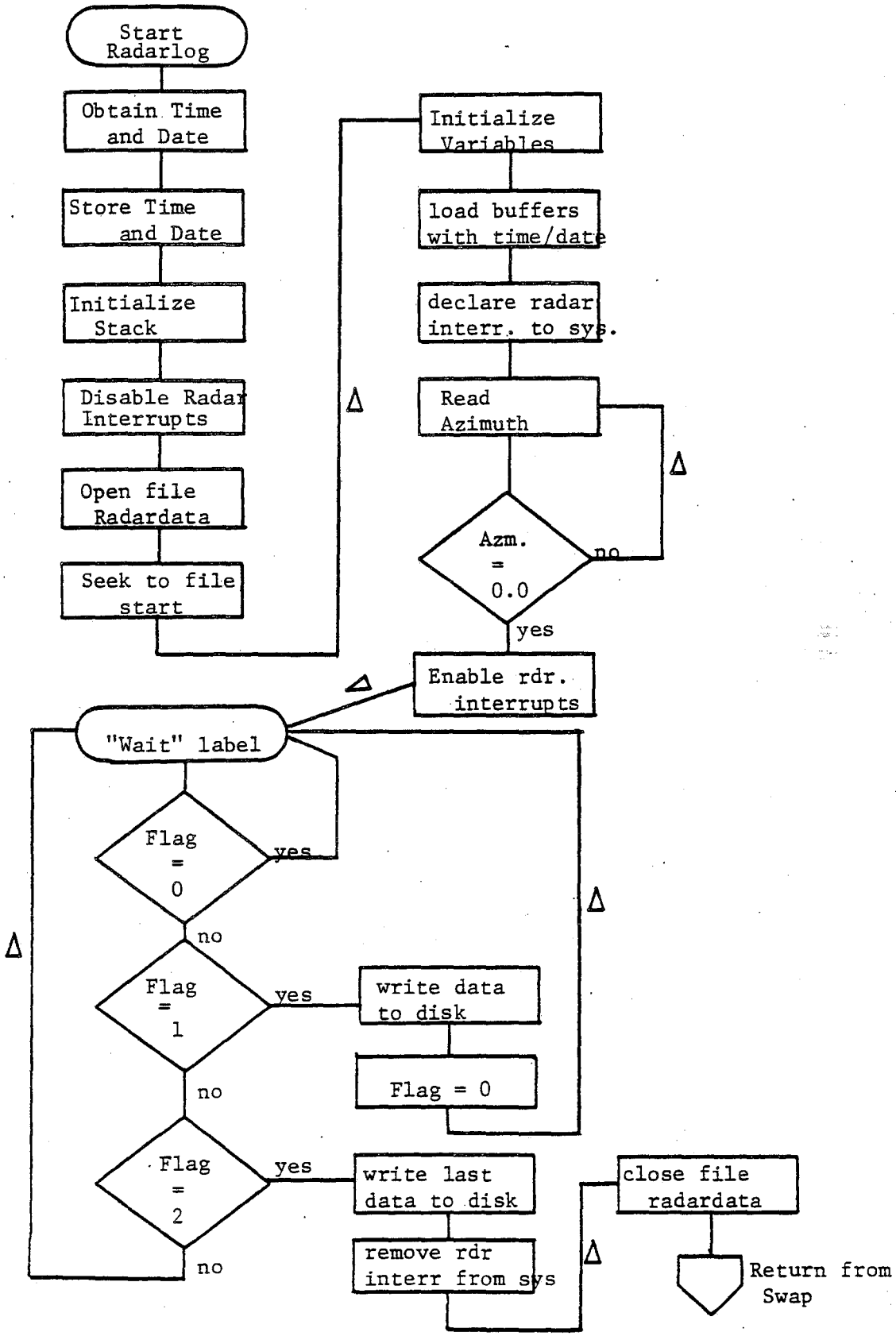
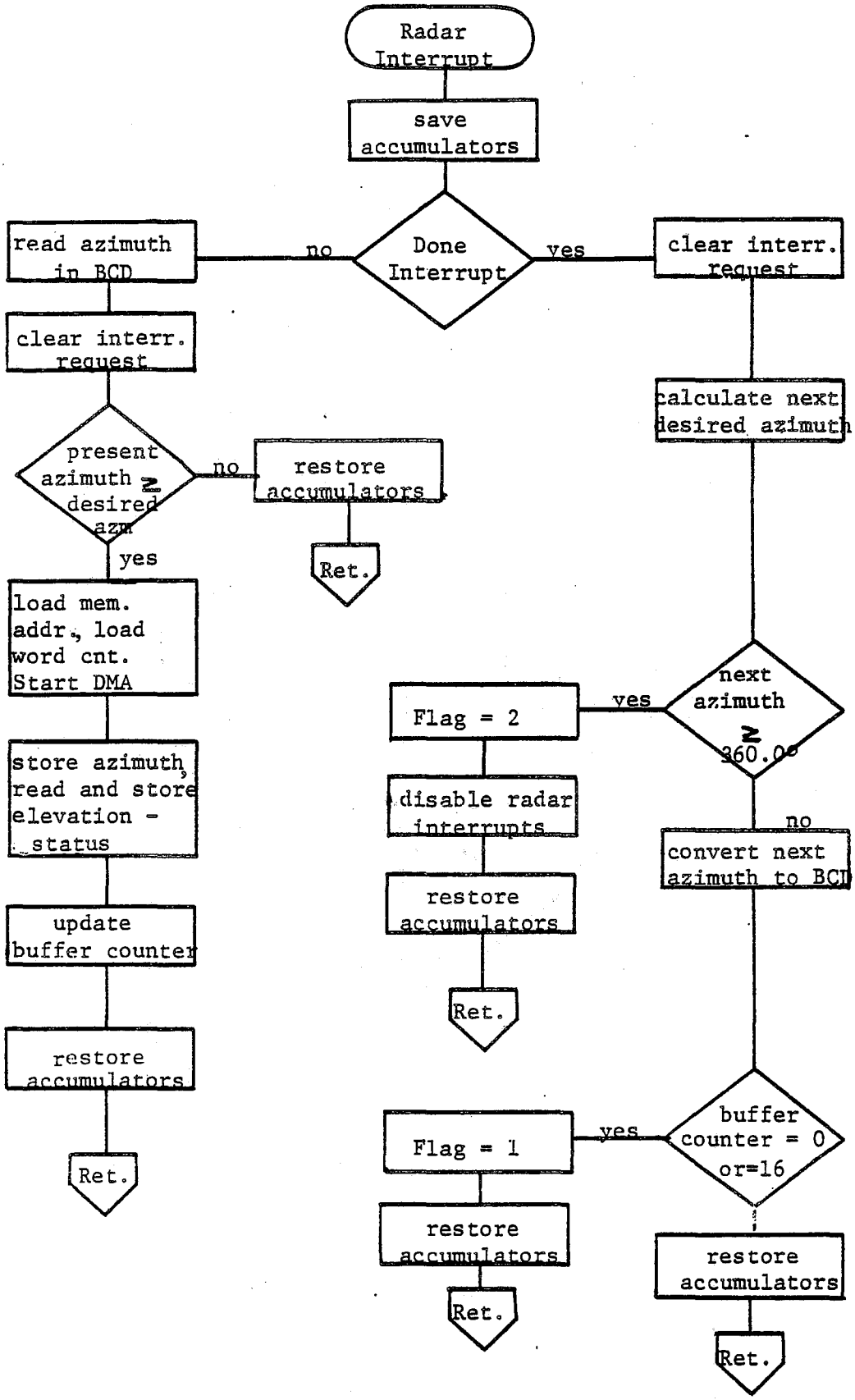


FIGURE 15 RADARLOG INTERRUPT SCHEME



determines which group of 16 buffers to write to disk, executes the write command, and then returns to the "WAIT" routine.

The finish routine is executed when a complete 360 degrees of data have been taken. The last partially filled buffer is then written to disk, the radar interrupts disabled, the data file is closed, and the system performs a return from swap.

C. Software - Grid

The Grid program is used to convert the data collected using the radarlog program into a gridded data file. The grid is centered around the radar site and consists of a 110 X 110 array with a resolution specified by the parameter data file. The array is stored in a file named GRID in four quadrants, each consisting of a 55 X 55 array. The main limitation on the array size is caused by available memory which is the primary reason why the grid is divided into quadrants.

The program begins by initializing certain variables and reading in other variables from the stored data file. The grid array file is opened and is zeroed. The radardata file is opened and data processing commences.

One block of data (two data scans) are read into memory, the azimuth determined and the sin and cosine are calculated. The quadrant is determined for that particular azimuth and if the quadrant is not memory-resident, the quadrant change subroutine is called. Quadchg simply writes the memory-resident quadrant to the grid file and reads the desired quadrant into memory.

Processing continues by checking the status bits to determine the range interval. If the range interval is one, the range for each sequential data byte is updated by one kilometer. If the interval is two, the data bytes are updated in increments of two kilometers. The range interval is determined at collection time by a front panel switch on the WSR-74C DVIP.

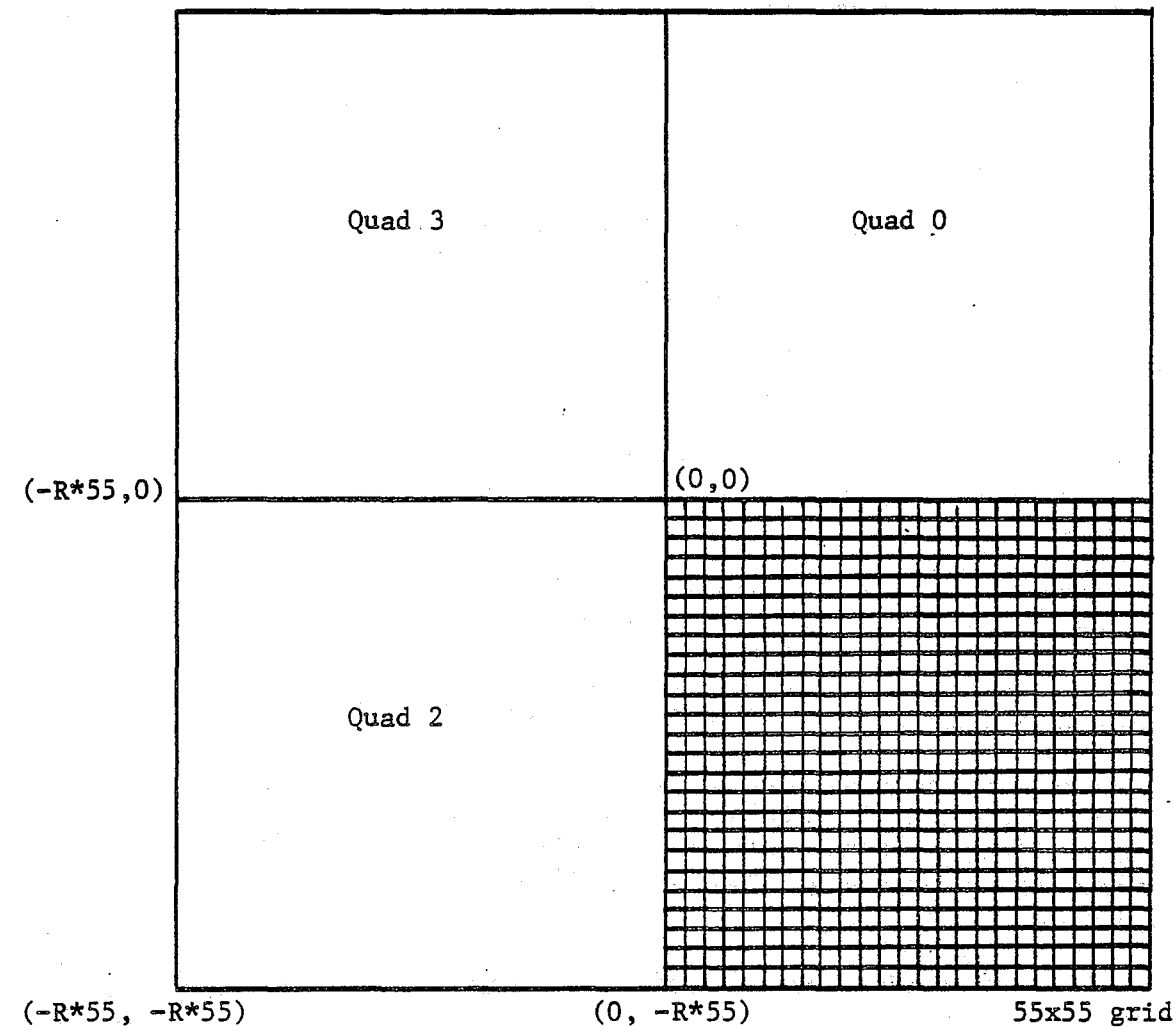
The program next converts every data byte (in radar DB/2) into an appropriate grid element is filled. If the calculated array element falls outside the grid bounds, it is ignored. Also, if the calculated level is less than the previous level in a particular element, the element is not updated.

After the program has finished executing, a grid file (48 contiguous blocks) has been created with maximum intensity information for each grid box. It is compatible with the graphics generation package incorporated in the GRAPH program.

D. Software - Graph

The Graph program converts the information contained in the parameter file and the gridded data file to an AFOS-compatible graphics product.

FIGURE 16 GRID QUADRANTS



R = resolution

FIGURE 17 GRID PROGRAM FLOW CHART

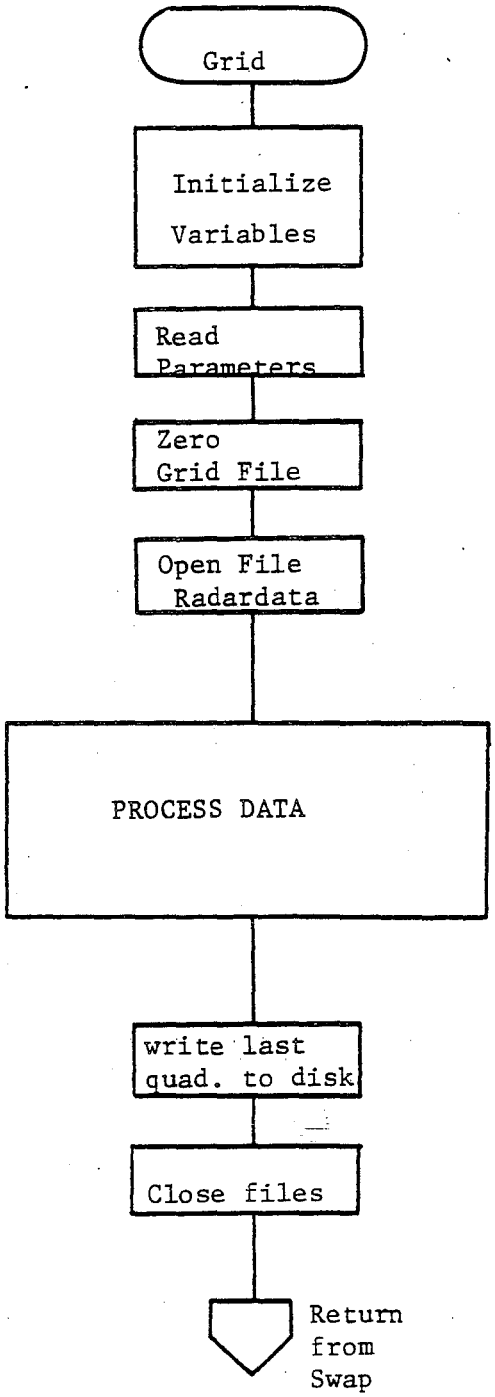
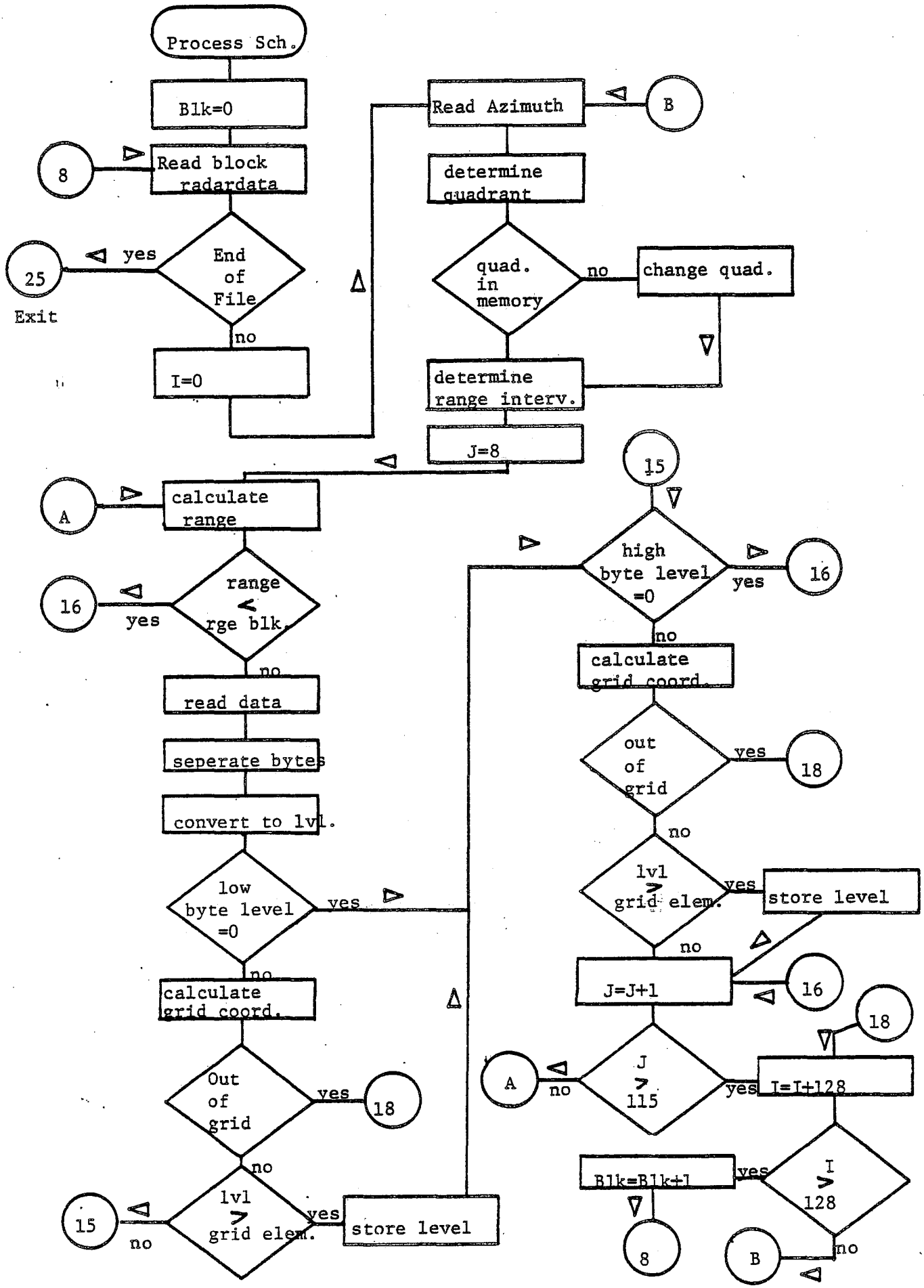


FIGURE 18 GRID PROCESSING DATA SCHEME



The program incorporates a block data subprogram which contained the pre-assembled AFOS graphics instructions. This approach reduces the memory requirements for graphics routines and decreases execution time by several fold.

The main program begins by opening and reading several variables from the radardata and parameter files. Some of the variables are converted into ascii characters (using subroutine decasc) and stored in the pre-assembled array. The second portion ot the program processes the intensity data stored in the grid file. Variables that control the appearance of the resultant product, such as scaling and centering, are initialized. The grid file is opened and the intensity data is processed.

Each quadrant is read into memory from the grid file and the array is searched for non-zero elements. If a non-zero level is detected, the AFOS screen coordinates are calculated. If the coordinates lie on the screen, it is plotted by adding the appropriate four-word instruction set to the graphics array.

When the graphics array reaches a length of 256 words, it is written onto disk in file "NMCGPHRDR" and the array element count is reset.

Processing continues until all four quadrants have been analyzed. The program then exits using a return from swap command.

The graphics file has been assembled in UTF (Universal Transmission Format). The only important requirements of UTF is that every 204k byte (end of transmission) be replaced by a 20k,14k sequence (delete, form feed), and every 20k byte (delete) be replaced by a 20k, 20k sequence. The pre-assembled block data code has these changes already incorporated in it; however, in the generation of the intensity code, the 204k or 20k byte can appear in the X or Y screen coordinate instruction. To greatly simplify matters, several lines of code have been added to increment the coordinate by one if the 203k or 20k byte appear.

AFOS requires a certain protocol for its graphics instructions. Refer to the tables which details the format of the block data pre-format. The first portion describes the AFOS data base name and is called the communication header. The graphics product definition comes next and describes the coordinate system used. It states the maximum x and maximum y screen coordinates. The block data subprogram contains code for drawing the box around the screen. This is accomplished by using the relative vector instruction. The alphanumerics are written by using the alphanumerics instruction.

Details on the code and subroutines can be obtained from the program listings for making the map backgrounds. These programs were used in version 1.0 of the graph program and were eliminated for efficiency sake.

The program terminates when all grid data has been analyzed and the AFOS graphics file is complete.

FIGURE 19 GRAPH PROGRAM FLOW CHART

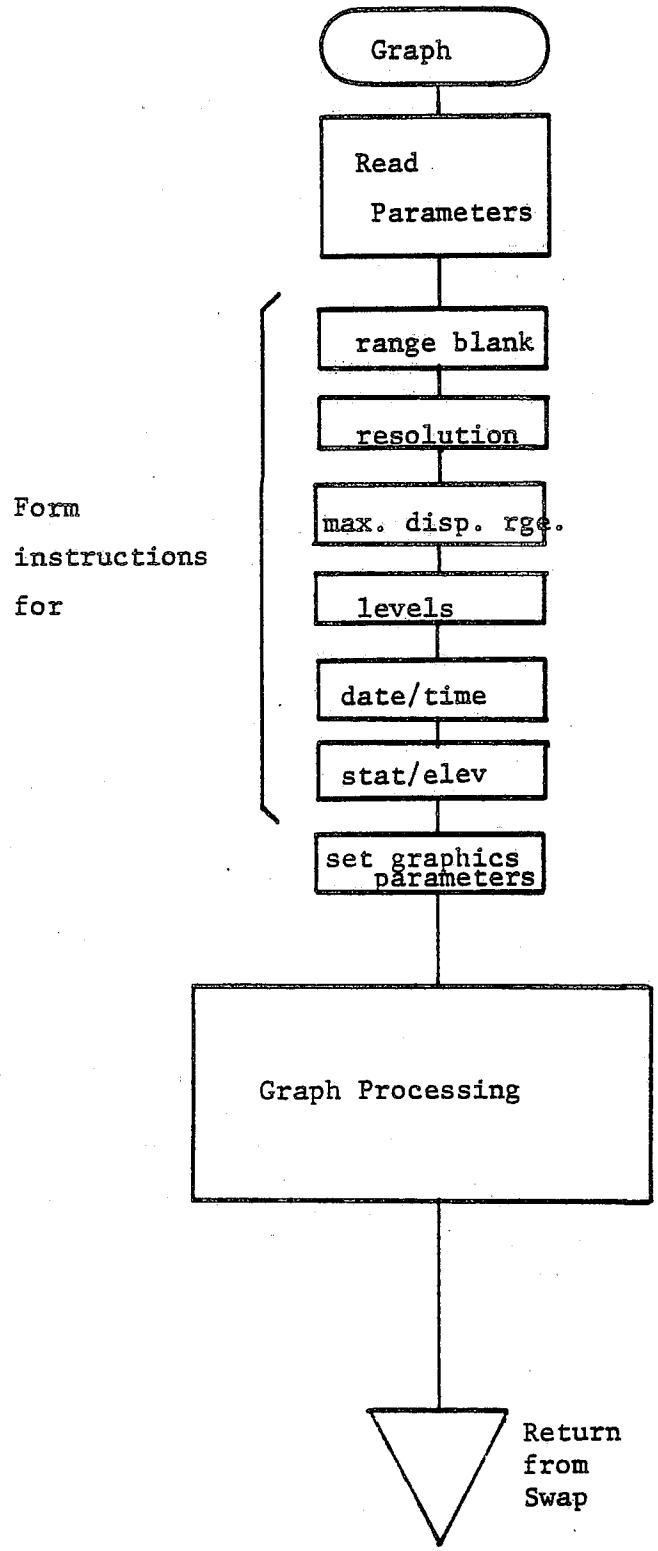


FIGURE 20 GRAPH PROCESSING FLOW

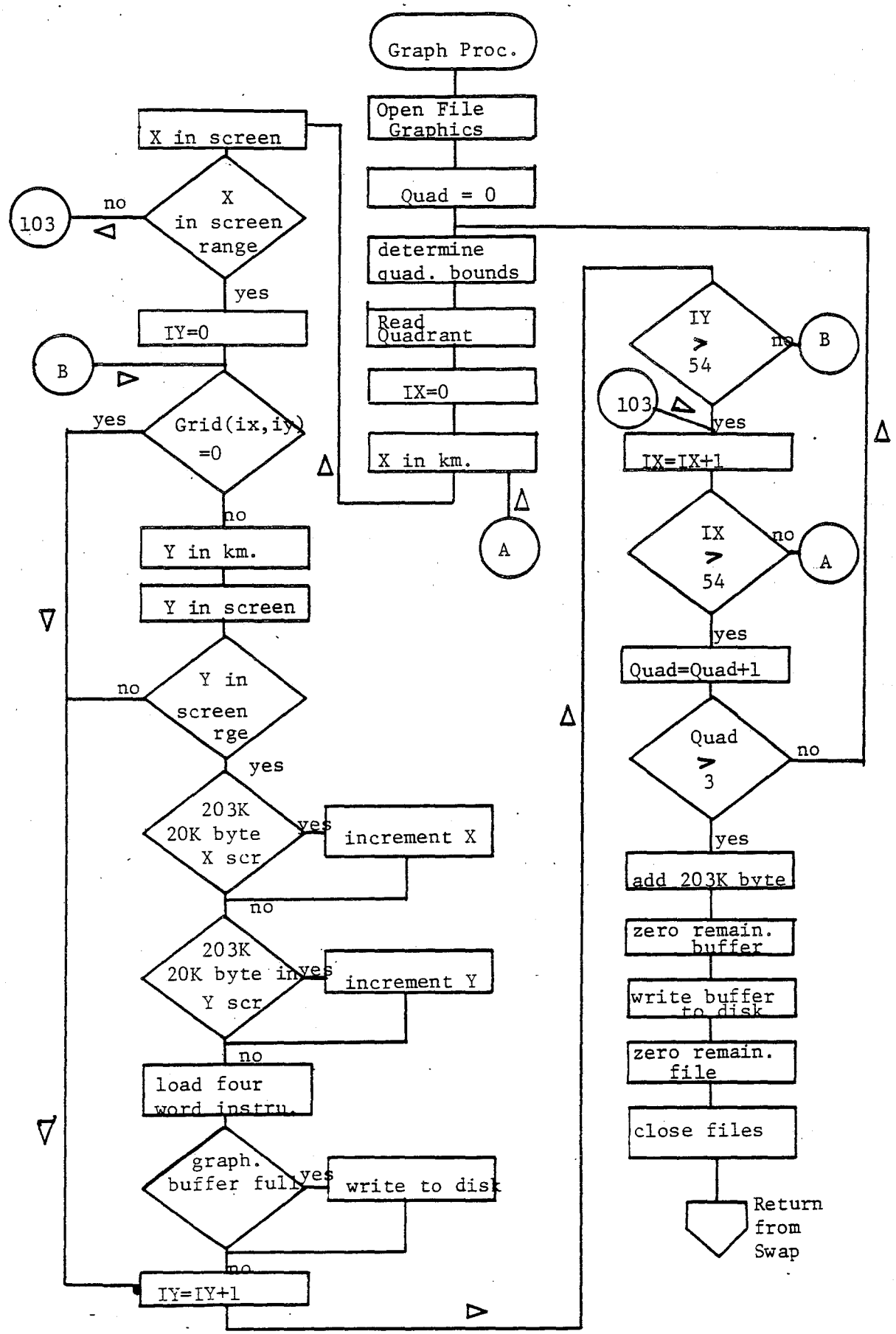


FIGURE 21 GRAPH BLOCK DATA SUBPROGRAM BREAKDOWN

```
"NMCGPHRDR000"     Communications Header--AFOS data base name
-1,-1,2400k
140400k            Graphics Product Definiation
0                       geography scale (not used)
10000k                  maximum i coordinate
6000k                   maximum j coordinate
0                       day-month-year (not used)
0                       time-pdc (not used)
141400k            Relative Vector Instruction (draw box around frame)
0                       i coordinate start
0                       j coordinate start
8                       8 words follow (4 vector pairs)
0                       delta x (vector 1)
3070                    delta y (vector 1)
4090                    delta x (vector 2)
0                       delta y (vector 2)
0                       delta x (vector 3)
-3070                   delta y (vector 3)   (bit 15,14,13 set)
-4090                   delta x (vector 4)   (bit 15,14,13 set)
0                       delta y (vector 4)
142400K            Text Writing Instruction--"RADAR"
143504K                 i coordinate start with large letters
5524k                   j coordinate start
"RADAR "                title
142400k            Text Writing Instruction--Date
62k                     i coordinate start
1130k                   j coordinate start
"DATE= xxx xx    "      text
142400k            Text Writing Instruction--Time
62k                     i coordinate start
1034k                   j coordinate start
"TIME= xxxxZ     "      text
142400k            Text Writing Instruction--Elevation
62k                     i coordinate start
740k                    j coordinate start
"ELEVATION=00.0  "      text
142400k            Text Writing Instruction--Range Blanking
62k                     i coordinate start
644k                    j coordinate start
"RANGE BLANK= 000 "     text
142400k            Text Writing Instruction--Resolution
62k                     i coordinate start
454k                    j coordinate start
"RESOLUTION= 00   "     text
142400k            Text Writing Instruction--Max. Display Range
62k                     i coordinate start
454k                    j coordinate start
"MAX DISP RGE=000 "     text
```

```
142400k          Text Writing Instruction--Time Sample
62k                   i coordinate start
360k                  j coordinate start
"TIME SAMPLE=00  "    text
142400k          Text Writing Instruction--IF Attenuators
62k                   i coordinate start
264k                  j coordinate start
"IF ATTN=OFF     "    text
142400k          Text Writing Instruction--Level one
62k                   i coordinate start
36k                   j coordinate start
"LV1=X.XX"            text
142400k          Text Writing Instruction--Level two
702k                  i coordinate start
36k                   j coordinate start
"LV2=X.XX"            text
142400k          Text Writing Instruction--Level three
1522k                 i coordinate start
36k                   j coordinate start
"LV3=X.XX"            text
142400k          Text Writing Instruction--Level four
2342k                 i coordinate start
36k                   j coordinate start
"LV4=X.XX"            text
142400k          Text Writing Instruction--Level five
3162k                 i coordinate start
36k                   j coordinate start
"LV5=X.XX"            text
142400k          Text Writing Instruction--Level six
4002k                 i coordinate start
36k                   j coordinate start
"LV6=X.XX"            text
142400k          Text Writing Instruction--Level seven
4622k                 i coordinate start
36k                   j coordinate start
"LV7=X.XX"            text
142400k          Text Writing Instruction--Level eight
5442k                 i coordinate start
36k                   j coordinate start
"LV8=X.XX"            text
142400k          Text Writing Instruction--Level nine
6262k                 i coordinate start
36k                   j coordinate start
"LV9=X.XX"            text
```

68 words of zero follow in the 256 word block data buffer

E. Software - Map Backgrounds

The data for the map backgrounds have been obtained from the DIMECO data base. This data base contains a county boundary file produced by the Census Use Study for its computer graphics research. The file consists of 46,159 records on a standard-labeled magnetic tape. Each record contains one latitude-longitude and one latitude-longitude change pair. Using coordinate conversions, the data can be converted to AFOS screen coordinates. Two programs have been written; the first runs on an IBM 360/168 and reads the mag tape. It sorts out the important coordinates and punches these on to cards. The second program runs on a DGC computer and converts the coordinates into AFOS graphics using specially adapted routines written by Jim Fors and Alexander MacDonald of Western Region Headquarters.

Program listings for all the necessary conversions from the DIMECO data base to AFOS graphics have been included in the appendix. The graphics routines originally were used in the graph program version 1.0.

F. Software - Parameter

The user of the radar software can specify certain parameters to tailor the resultant graphics product to specific requirements. Some of the parameters need only be set once (at the time of installation); however, others may be changed quite frequently. Below is a list of the various changes that can be made:

1. Observation Interval - for automatic soliciting of radar data by the control program.

2. Range Blanking - used to blank out areas of ground clutter to a specified distance.

3. Resolution - determines the grid box size and details of echoes.

4. Radar Threshold - sets the level output from the DVIP when a level of 0.01"/hr precipitation rate is observed.

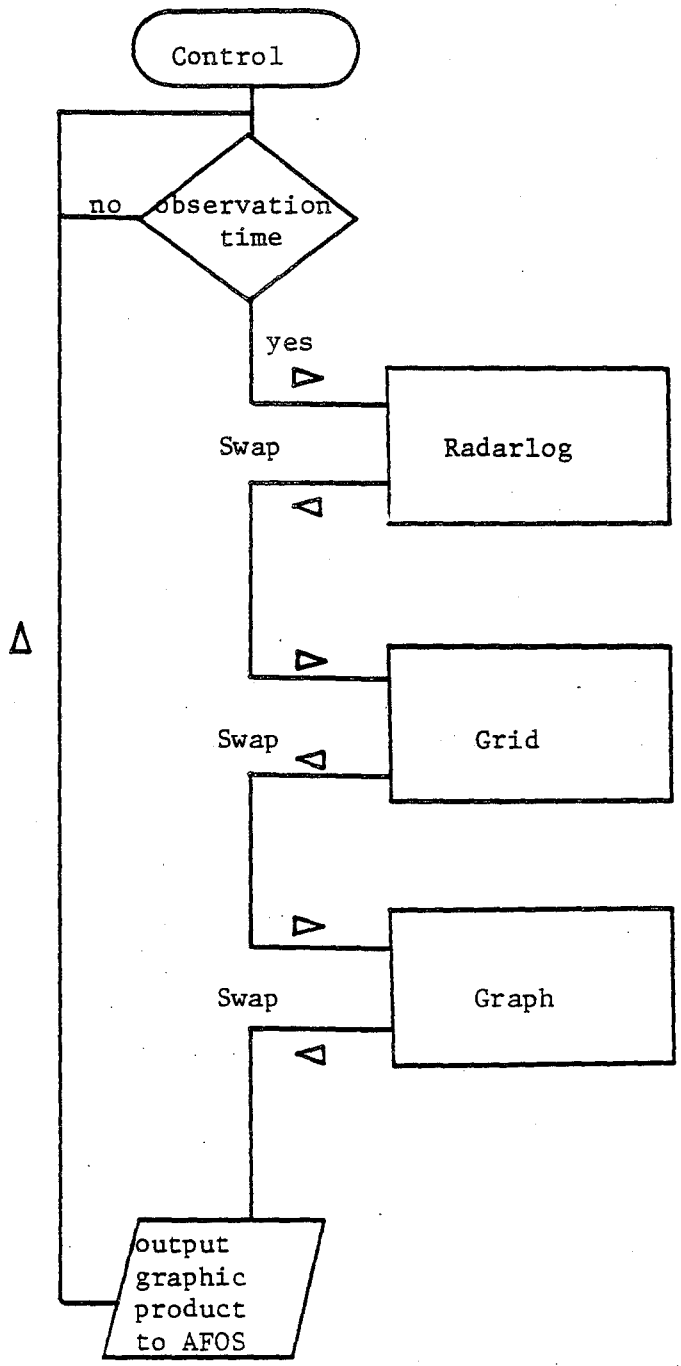
5. Thresholds - threshold levels for up to nine levels can be set for detailed analysis of precipitation rates.

The parameter program simply asks the user for various parameters, stores these in an array, and stores the array on to disk. The parameter file is read by the grid and graph programs for variable initialization.

G. Software - Control

The control program was designed for automatic running of the radar programs. The user simply types "control" and the radar program will automatically by scheduled at the appropriate time. Time is checked by using the Fortran subroutine Fgtim. At observation time, the program swaps

FIGURE 22 CONTROL PROGRAM FLOW CHART



to the radarlog, grid, and then to the graph program. When the AFOS graphics file is complete, the control program sends the file to AFOS using the $TTO line.

IV. IMPROVEMENTS AND RECOMMENDATIONS

Several changes could be made to improve the quality of the AFOS radar map. One of these, ground clutter rejection, would be extremely valuable to the forecaster.

A. Ground Clutter Rejection

Several methods are available for ground clutter rejection. The easiest method blanks out all areas of ground clutter. The major problem is that if precipitation exists over the ground clutter, it would not appear in the display.

The second method involves the subtraction of ground clutter echoes. A data file can be created on a non-precipitation day. When precipitation is detected, the non-precipitation is subtracted resulting in a precipitation-only display. This sounds easy; however, ground-clutter returns do not always have the same intensity or location. When the elevation angle is changed, ground-clutter returns can also change. It may be possible to construct a file containing the maximum echo return over all elevations. This method would eliminate most ground returns.

The third method of ground clutter rejection is almost 100% effective; however, it is also the most expensive. The raw video output from the radar is sampled with a high-speed analog-to-digital converter. The data is compared scan to scan and the variance of a target is calculated. If the variance is high, it can be assumed that the echo is precipitation; if the variance is small, the echo is probably ground clutter.

B. Regional Maps

At the present time, only a county map centered around the radar site is available. It would be quite practical to produce a regional map containing several radar sites. A composite map would be more useful to the forecaster than the single map since one would not have to be bothered by checking each map individually; however, that option would still exist. The scale could incorporate two or three states with a resolution of about ten kilometers. For finer details, the forecaster would have to consult the individual maps.

C. Precipitation Totals

The intensity information gathered at each radar site could be integrated over time to provide rainfall totals over periods of several hours. This could alert forecasters of heavy rainfall amounts caused by stationary cells. Other products might include rainfall areas and amounts for river basin forecasting.

D. Alarm Products

The AFOS alarm/alert system lends itself admirably to the radar data project. Alarms could automatically be actuated if echoes were to be observed over a certain intensity or if precipitation totals exceed a preset value. This would reduce the amount of time the forecaster would need studying the radar maps.

E. Parameter Change

A procedure combined with the message composition feature on AFOS would allow the forecaster to run the parameter change program from AFOS. For remote sites where the radar and AFOS are not co-located, this would greatly reduce the workload of the forecaster. All that is required to implement this option is the capability to do multi-tasking on the Nova312. An example of message pre-format has been included. The forecaster simply fills in the parameters.

F. Operating System

The current operating system running on the Nova312 is DOS (Diskette Operating System). DOS has been designed to simulate RDOS (Real-time Disk Operating System) and therefore has quite useful features. The major problem with DOS is that it is too slow. For example, when a program name is typed on the keyboard, it can take about thirty seconds before the program starts to execute. It has been recommended to me that programs be developed under DOS but run under RTOS (Real-time Operating System). RTOS is much faster, takes less core, and is as versatile as RDOS. Multi-tasking implemented on RTOS which would allow many innovative products.

DOS is now running with 16K of memory. The DOS system only leave 8K of memory to be used for program usage. This has put a tight lid on program development. Programs can be written to run in 8K provided they use the disk for temporary storage and are split into small sections. The group of radar programs would run much quicker (70% improvement) if memory was available to combine the programs into one unit. Presently, the group of radar programs require five minutes to run, half which DOS is loading or unloading programs from core. It is recommended that more memory be purchased and a different operating system be used.

G. Elevation

For an additional $1000 in hardware, a digital-to-synchro converter can easily be installed to control the radar antenna elevation from the Nova312. This would allow the calculation of height maps, vertically integrated liquid water content maps, and a slew of many more useful products.

FIGURE 23 RADAR PARAMETER CHANGE PROCEDURE

```
WRHMCP008
WOUS20 KWRH 252034

            RADAR PARAMETER CHANGE PROCEDURE

OBSERVATION INTERVAL  (MIN)    [015]
RESOLUTION (KM)                [004]
GROUND CLUTTER BLANKING (KM)   [020]



THRESHOLD LEVELS    [01] [10] [20] [30] [40] [50] [60] [70] [80]
       REFER TO RAINFALL RATE CHART FOR EQUILIVANT LEVELS.
       FILL UNUSED LEVELS WITH ZEROES.



                                POSITION CURSOR HERE--PRESS ENTER------[ ] [ ]
                                                                 PAGE 01
```

APPENDIX

RADARLOG

```
;*****************************************************************
;       RADAR LOGGING PROGRAM
;
;  VERSION 1.0   JULY 8,1979   MARK MATHEWSON--SSD WRH NWS
;*****************************************************************


; RADAR LOGGING PROGRAM GATHERS RAW RADAR DATA USING DMA AND STORES
; IN FILE DP1:RADARDATA.  REFER TO DOCUMENTATION FOR COMPLETE
; DETAILS ON FORMAT USED.

        .TITL RADARLOG
        .ENT AZNXT,AZINC,AZBCD,BUFC,WORD,HOUR,DAY,MIN
        .ENT MONTH,N1,CO,FI,STKP,N3,N32,N3600,N16,N128
        .ENT N8,N25,BLK,ERR,DCTP,N8S,FILENM,BUFP,BUF0P
        .ENT BU16P,CNTMAX,PTT,COUNT,DIVR,STK,START,OPENF
        .ENT SEEK,L1,SINT,AZ0,WAIT,CONT,FINI1,WRT,RADAR
        .ENT DONE,R,RC,RF,XMIT,BINBCD,L2,ONE,NXT,ERROR
        .ENT BUF0,BUF1,BUF2,BUF3,BUF4,BUF5,BUF6,BUF7
        .ENT BUF8,BUF9,BUF10,BUF11,BUF12,BUF13,BUF14,BUF15
        .ENT BUF16,BUF17,BUF18,BUF19,BUF20,BUF21,BUF22
        .ENT BUF23,BUF24,BUF25,BUF26,BUF27,BUF28,BUF29
        .ENT BUF30,BUF31,DCTR,FLAG
        .TXTM 1


        .EXTN .UIEX
        .DIAC PSHA=61401
        .DIAC POPA=61601
        .DUSR SAV=62401
        .DIAC MTSP=61001
        .DIAC MTFP=60001
        .DUSR RET=62601

; DATA WORDS STORAGE AREA
        .ZREL
AZNXT:  0       ;NEXT AZIMUTH (BINARY)
AZINC:  10.     ;AZIMUTH INCREMENT (1/10TH DEGREES)
AZBCD:  1       ;NEXT AZIMUTH (BCD)
BUFC:   0       ;BUFFER COUNTER
WORD:   -107.   ;WORD COUNTER FOR DMA
HOUR:   0       ;HOUR
DAY:    0       ;DAY
MIN:    0       ;MINUTE
MONTH:  0       ;MONTH
N1:     1       ;NUMBER ONE
CO:     CONT    ;CONTINUE ROUTINE ADDRESS
FI:     FINI1   ;FINISH ROUTINE ADDRESS
STKP:   STK-1   ;STACK POINTER
N3:     3
N32:    32.
N3600:  3600.
N8S:    2000    ;8. SWAPPED BYTES
N16:    16.
N128:   128.
FLAG:   0
N8:     8.
N25:    25
```

```
BLK:    0           ;DISK FILE RELATIVE BLOCK NUMBER
ERR:    ERROR       ;ERROR ROUTINE POINTER
DCTP:   DCTR        ;RADAR DCT POINTER
NBS:    4000        ;8. SWAPPED BYTES
FILENM:             .+1*2   ;POINTER TO ASCII FILENAME
        .TXT "DP1:RADARDATA"    ;RADARDATA FILE
ERR1:   .+1*2       ;POINTER TO ERROR MESSAGE
        .TXT " R1:RADARDATA PGM ERROR"
LOGFIL:             .+1*2   ;POINTER TO ERROR LOG FILE NAME
        .TXT "DP1:ERR1.LOG"
BUFP:   BUF0P       ;BUFFER POINTER
BUF0P:  BUF0        ;BUFFER POINTERS
        BUF1
        BUF2
        BUF3
        BUF4
        BUF5
        BUF6
        BUF7
        BUF8
        BUF9
        BUF10
        BUF11
        BUF12
        BUF13
        BUF14
        BUF15
BU16P:  BUF16
        BUF17
        BUF18
        BUF19
        BUF20
        BUF21
        BUF22
        BUF23
        BUF24
        BUF25
        BUF26
        BUF27
        BUF28
        BUF29
        BUF30
        BUF31
CNTMAX:             16.     ;MAX COUNTER FOR BINBCD ROUTINE
PTT:    DIVR-1      ;ADDRESS OF DIVIDE CONSTANTS BINBCD ROUTINE
COUNT:  0           ;COUNTER FOR BINBCD ROUTINE
DIVR:   8000.       ;DIVIDE CONSTANTS BINBCD ROUTINE
        4000.
        2000.
        1000.
        800.
        400.
        200.
        100.
        80.
        40.
        20.
        10.
        8.
        4.
        2.
        1.
;STACK AREA
STK:    .BLK 100.
```

```
;MAIN LOGGING PROGRAM--STARTUP ROUTINE
        .NRLL
START:  .SYSTM  ;GET TIME AND DATE AND STORE INTO MEMORY
        .GDAY
        HALT    ;NEVER GETS HERE
        STA 1.MONTH     ;STORE MONTH
        STA 0,DAY       ;STORE DAY
        .SYSTM  ;GET TIME
        .GTOD
        HALT    ;NEVER GETS HERE
        STA 2,HOUR      ;STORE HOUR
        STA 1,MIN       ;STORE MINUTE
        LDA 0,STKP      ;SET STACK UP
        MTFP 0          ;SET FRAME POINTER
        MTSP 0          ;SET STACK POINTER
        SUB 0,0         ;CLEAR AC0
        DOC 0,25        ;DISABLE XMIT INTERR, SET TTO OUTPUT
        NIOC 25         ;CLEAR DONE INTERRUPTS RADAR

;OPEN RADARDATA FILE--CHANNEL 3
OPENF:  LDA 0,FILENM    ;LOAD POINTER TO FILENAME
        SUB 1,1         ;KEEP SYSTEM ATTRIBUTES FOR FILE
        .SYSTM          ;OPEN FILE
        .OPEN 3         ;ON CHANNEL 3
        JMP @ERR        ;ERROR--GO TO ERROR ROUTINE

;POSITION DISK HEADS TO START OF FILE (USE READ COMMAND)
SEEK:   LDA 0,BUF0P     ;LOAD MEMORY ADDRESS FOR READ
        LDA 1,BLK       ;LOAD BLOCK NUMBER (0)
        LDA 2,N1        ;AC2=1  READ 1 BLOCK
        MOVS 2,2        ;BLK CNT IN LEFT BYTE
        .SYSTM          ;EXECUTE READ BLOCK COMMAND
        .RDB 3          ;CHANNEL 3
        JMP @ERR        ;ERROR--GO TO ERROR ROUTINE

;INITILIZE VARIABLES
        SUB 0,0         ;AC0=0
        STA 0,BUFC      ;BUFC=0
        STA 0,AZNXT     ;AZNXT=0
;       STA 0,AZBCD     ;AZBCD=0

;LOAD ALL BUFFERS WITH TIME AND DATE HEADER
        LDA 2,BUF0P     ;BUFFER POINTER
        LDA 1,N32       ;AC1=32.
        NEG 1,1         ;AC1=-32. COUNTER
L1:     LDA 0,MONTH     ;GET MONTH INTO AC0
        STA 0,3,2       ;STORE IT
        LDA 0,DAY       ;GET DAY INTO AC0
        STA 0,3,2       ;STORE IT
        LDA 0,HOUR      ;LOAD HOUR INTO AC0
        STA 0,4,2       ;STORE IT
        LDA 0,MIN       ;LOAD MINUTE INTO AC0
        STA 0,5,2       ;STORE IT
        LDA 0,N128      ;LOAD 128. INTO AC0
        ADD 0,2         ;UPDATE POINTER TO NEXT BUFFER
        INC 1,1,SZR     ;ALL BUFFERS FULL, SKIP
        JMP L1

;TELL SYSTEM THAT RADAR INTERRUPTS EXIST
SINT:   LDA 0,N25       ;LOAD RADAR DEVICE CODE (25 OCTAL)
        LDA 1,DCTP      ;LOAD ADDRESS OF DEVICE DCT
        .SYSTM          ;IDENTIFY THE RADAR INTERRUPT
        .IDEF
        JMP @ERR        ;ERROR--GO TO ERROR ROUTINE
```

```
;WAIT FOR ANTENNA AZIMUTH TO EQUAL 0.0
AZ0:    DIA 0,25        ;READ AZIMUTH IN BCD
        LDA 1,AZBCD     ;LOAD START AZIMUTH
        SUB # 0,1,SZR   ;SKIP IF AZM=START AZM
        JMP .-3         ;TRY AGAIN

;START RADAR INTERRUPTS
        LDA 0,N1        ;AC0=1
        DOC 0,25        ;ENABLE XMIT INTERRUPTS

;WAIT ROUTINE
WAIT:   LDA 0,FLAG              ;WAIT ROUTINE
        NEG 0,0,SNR     ;GET NEGATIVE OF FLAG
        JMP WAIT        ;FLAG=0 WAIT ROUTINE
        INC 0,0,SNR     ;INCREMENT AC0
        JMP CONT        ;FLAG=1, CONTINUE ROUTINE
        INC 0,0,SNR     ;INCREMENT AC0
        JMP FINI1       ;FLAG=2, FINISH ROUTINE
        JMP WAIT        ;OTHER, WAIT ROUTINE


;CONTINUE ROUTINE
CONT:   JSR WRT         ;WRITE DISK AND SEEK
        SUB 0,0         ;CLEAR AC0
        STA 0,FLAG      ;CLEAR FLAG
        JMP WAIT        ;GO TO WAIT ROUTINE

;FINISH ROUTINE
FINI1: LDA 0,BU16P      ;SET BUFFER 16
        STA 0,BUFC
        LDA 1,N4S       ;SET 4 BLOCK WRITE
        STA 1,N8S
        JSR WRT         ;WRITE DISK AND SEEK
        LDA 0,N25       ;REMOVE RADAR INTERR HAND FROM SYSTM
        .SYSTM
        .IRMV
        JMP @ERR        ;ERROR--GO TO ERROR ROUTINE
        .SYSTM          ;CLOSE RADARDATA FILE
        .CLOSE 3
        JMP @ERR        ;ERROR--GO TO ERROR ROUTINE
        LDA 1,N3600     ;TO BE SURE OF RESEHEDULING
        .SYSTM          ;RETURN FROM PROGRAM SWAP
        .RTN
        JMP @ERR        ;ERROR--GO TO ERROR ROUTINE
        HALT            ;NEVER GETS HERE


;WRITE ROUTINE
WRT:    SAV             ;SAVE REGISTERS
        LDA 1,BUFC      ;LOAD BUFFER COUNTER INTO AC0
        MOV 1,1,SZR     ;DETERMINE MEM ADDR FOR WRITE
        JMP .+3         ;MUST BE BUFC=0 OR BUFC=16.
        LDA 0,BU16P     ;MEM ADDR START=BUF16P
        JMP .+2
        LDA 0,BUF0P     ;MEM ADDR START=BUF0P
        LDA 1,BLK       ;LOAD BLOCK NUMBER INTO AC1
        LDA 2,N8S       ;BLOCK COUNT 8
        .SYSTM          ;WRITE TO DISK
        .WRB 3          ;CHANNEL 3
        JMP @ERR        ;ERROR--GO TO ERROR ROUTINE
        LDA 2,N8        ;AC2=8.
        ADD 2,1         ;INCREMENT BLK NUMBER BY 8. FOR NEXT TIME
        STA 1,BLK       ;STORE UPDATED BLOCK NUMBER
        RET             ;RETURN FROM SUBROUTINE
```

```
;       DIA 0,25        ;READ IN AZIMUTH
        STA 0,0,3       ;STORE AZIMUTH
        DIB 0,25        ;READ IN STATUS-ELEVATION
        STA 0,1,3       ;STORE STATUS-ELEVATION
        LDA 0,BUFC      ;UPDATE BUFFER COUNTER
        INC 0,0         ;INCREMENT BY ONE
        LDA 1,N32       ;AC1=32.
        AND # 1,0,SZR   ;BUFC>=32, THEN SET BUFC=0
        SUB 0,0         ;AC0=0
        STA 0,BUFC      ;STORE UPDATED VALUE OF BUFC
        JMP R           ;NORMAL RETURN


;BINARY TO BCD CONVERTER ROUTINE (AC0---AC0)
BINBCD:         PSHA 1          ;SAVE REGISTERS
        PSHA 2
        MOVL 3,3
        PSHA 3
        LDA 2,PTT       ;LOAD POINTER TO DIVIDE CONSTANTS
        STA 2,21        ;STORE IN AUTO-INC LOCATION
        LDA 2,CNTMAX    ;INITILIZE COUNTER
        STA 2,COUNT
        SUB 2,2         ;CLEAR AC2
L2:     LDA 1,@21       ;LOAD DIVIDE CONSTANT
        ADCZ # 0,1,SNC  ;RESULT>=0,SKIP
        JMP ONE
        MOVZL 2,2       ;MOVE 0 INTO RIGHTMOST BIT
        JMP NXT         ;MORE TO DO
ONE:    SUB 1,0         ;STORE NEW VALUE OF AC0
        MOVOL 2,2       ;STORE 1 INTO RIGHTMOST BIT
NXT:    DSZ COUNT       ;DECREMENT COUNTER
        JMP L2          ;NOT DONE YET
        MOV 2,0         ;MOVE RESULT TO AC0
        POPA 3          ;RESTORE REGISTERS
        MOVR 3,3
        POPA 2
        POPA 1
        JMP 0,3         ;RETURN


;ERROR HANDLER
ERROR:  SUB 1,1         ;OPEN ERROR FILE
        LDA 0,LOGFIL    ;ERROR FILE NAME
        .SYSTM          ;APPEND THE FILE
        .APPEND 2       ;ON CHANNEL 2
        JMP .+4         ;ERROR, EXIT PROGRAM
        LDA 0,ERR1      ;LOAD POINTER TO ERROR MESSAGE
        .WRL 2          ;WRITE LINE
        JMP .+1         ;ERROR,EXIT PROGRAM
        LDA 1,N3600     ;TO BE SURE OF RESCHEDULING
        .SYSTM          ;CLOSE ALL FILES
        .RESET
        JMP .+1
        .SYSTM          ;RETURN FROM SVMP
        .RTN
        HALT
        HALT

;DEVICE CONTROL TABLE FOR RADAR (DCT)
DCTR:   0       ;IGNORES THIS WORD
        177777  ;INTERRUPT MASK (NO OTHER INTERRUPTS ALLOWED)
        RADAR   ;SERVICE ROUTINE ADDRESS LABEL
```

```
;RADAR DATA BUFFERS
BUF0:   .BLK 128.
BUF1:   .BLK 128.
BUF2:   .BLK 128.
BUF3:   .BLK 128.
BUF4:   .BLK 128.
BUF5:   .BLK 128.
BUF6:   .BLK 128.
BUF7:   .BLK 128.
BUF8:   .BLK 128.
BUF9:   .BLK 128.
BUF10:  .BLK 128.
BUF11:  .BLK 128.
BUF12:  .BLK 128.
BUF13:  .BLK 128.
BUF14:  .BLK 128.
BUF15:  .BLK 128.
BUF16:  .BLK 128.
BUF17:  .BLK 128.
BUF18:  .BLK 128.
BUF19:  .BLK 128.
BUF20:  .BLK 128.
BUF21:  .BLK 128.
BUF22:  .BLK 128.
BUF23:  .BLK 128.
BUF24:  .BLK 128.
BUF25:  .BLK 128.
BUF26:  .BLK 128.
BUF27:  .BLK 128.
BUF28:  .BLK 128.
BUF29:  .BLK 128.
BUF30:  .BLK 128.
BUF31:  .BLK 128.

        .END START
```

GRID

```
C*************************************************************************
C       CONVERSION PROGRAM FROM RAW RADAR DATA TO GRID ARRAY
C
C       VERSION 2.0  JULY 24,1979  MARK MATHEWSON--SSD WRH NWS
C  V2.0 REPLACES PORTION OF COMPRESS PGM (V1.0) AND ALL OF V1.0
C  GRID PGM.  OTHER PORTION OF V1.0 COMPRESS PGM IS IN V2.0
C  TITLE PGM.
C
C  GRID (V2.0) READ RAW RADAR DATA FROM FILE DP1:RADARDATA AND PUTS
C  IT DIRECTLY INTO GRID ARRAY WHICH IS STORED IN FOUR QUADRANTS ONTO
C  DISK. FILE NAMED GRID.
C*************************************************************************
        COMPILER NOSTACK
        EXTERNAL QUADCHG,BCDRAD
        COMMON/DTA/GRID,DUM
C VARIABLE DECLARATION AND INITIALIZATION
C PROGRAM IS SET FOR A 55 X 55 GRID ARRAY WITH FOUR SEPERATE
C QUADRANTS.
        INTEGER GRID(0:54,0:54),DUM(100),LEVEL(0:255),RGEINT
        INTEGER QUAD,RESOL,RGBLK,INB(0:255),BMSK,SMSK,BLK,RGMSK
        BMSK=377K          ;BYTE MASK
        SMSK=3777K         ;STATUS-ELEV MSK
        RGMSK=20000K       ;RANGE INTERV MASK
        BLK=0              ;BLOCK # OF RADARDATA FILE
        QUAD=4             ;QUADRANT
C READ DESIRED RESOLUTION, RANGE BLANKING, AND LEVELS FROM PARAMETER FILE
        CALL OPEN(1,"DP1:PARAMETER",2,IER)       ;OPEN FILE
        CALL RDBLK(1,0,LEVEL,1,IER)      ;READ PARAMETERS
        RESOL=LEVEL(0)   ;RESOLUTION
        RGBLK=LEVEL(1)   ;RANGE BLANKING
        CALL RDBLK(1,1,LEVEL(2),1,IER)      ;READ LEVELS
        CALL CLOSE(1,ERR)                   ;CLOSE PARAMETER FILE
C OPEN THE GRID FILE
        CALL OPEN(3,"DP1:GRID",3,IER)    ;OPEN GRID FILE
        DO I I=0,54
        DO I II=0,54
1       GRID(I,II)=0
        DO 2 I=0,36,12
2       CALL WRBLK(3,I,GRID,12,IER)
C OPEN RADAR DATA FILE
        CALL OPEN(2,"DP1:RADARDATA",2,IER)
C PROCESS RADAR DATA
C
```

```
C READ BLOCK OF DAT
8       CALL RDBLK(2,BLK,INB,1,IER)      ;READ BLOCK OF RADAR DATA
        IF(IER.EQ.9) GO TO 25   ;END OF FILE
        DO 18 I=0,120,120       ;TWO/SETS PER BLK
C DETERMINE AZIMUTH
        CALL BCDRAD(INB(I),IAZM,RAD)     ;IAZM=.1 DEG,RAD=RADIANS
        S=SIN(RAD)      ;SIN OF ANGLE
        C=COS(RAD)      ;COSINE OF ANGLE
C DETERMINE QUADRANT
        J=IAZM/900      ;900=90 DEG.
        IF(J.NE.QUAD) CALL QUADCHG(J,QUAD,IXBOT,IYBOT,RESOL) ;CHANGE QUAD
C DETERMINE RANGE INTERVAL
        RGEINT=IAND(RGMSK,IND(I+1))
        IF(RGEINT.EQ.0) RGEINT=2
        IF(RGEINT.NE.2) RGEINT=1
C DATA CONVERSION
        DO 16 J=8,115,1         ;J=POSITION IN BUFER
C OBTAIN RANGE
        RL=19.+RGEINT*(J-7)*2   ;RANGE LOW BYTE
        RH=RL+RGEINT            ;RANGE HIGH BYTE
C CHECK RANGE BLANKING
        IF(RL.LT.RGBLK) GO TO 16        ;WITHIN RANGE BLANKING RANGE.
C READ DATA
        I1=INB(J+I)     ;I1=DATA
C SEPERATE DATA INTO TWO BYTES
        I2=IAND(I1,BMSK)        ;LOW BYTE
        I3=ISHFT(I1,-8)         ;HIGH BYTE
C CONVERT DATA INTO LEVELS
        I2=LEVEL(I2)
        I3=LEVEL(I3)
C PROCESS LOW BYTE
        IF(I2.EQ.0) GO TO 15    ;LEVEL0 , SKIP PROCESSING
        IX=((RL*S)-IXBOT)/RESOL ;X GRID COORD.
        IY=((RL*C)-IYBOT)/RESOL ;Y GRID COORD.
        IF(IX.GT.54.OR.IY.GT.54) GO TO 18
        IF(IX.LT.0.OR.IY.LT.0) GO TO 18
        IF(GRID(IX,IY).LT.I2) GRID(IX,IY)=I2    ;PUT MAX VALUE INTO GRID
C PROCESS HIGH BYTE
15      IF(I3.EQ.0) GO TO 16    ;LEVEL0, SKIP PROCESSING
        IX=((RH*S)-IXBOT)/RESOL ;X GRID COORD.
        IY=((RH*C)-IYBOT)/RESOL ;Y GRID COORD.
        IF(IX.GT.54.OR.IY.GT.54) GO TO 18       ;OUT OF GRID
        IF(IX.LT.0.OR.IY.LT.0) GO TO 16
        IF(GRID(IX,IY).GT.13) GRID(IX,IY)=I3    ;PUT MAX VALUE INTO GRID
16      CONTINUE
18      CONTINUE
C INCREMENT BLOCK NUMBER BY ONE
        BLK=BLK+1
        GO TO 8         ;PROCESS NEXT BLOCK
C END OF DATA SET
25      J=QUAD-1        ;CHANGE QUAD FOR FORCE WRITING OF LAST QUAD
        CALL QUADCHG(J,QUAD,IXBOT,IYBOT,RESOL)  ;QUAD CHANGE
        CALL CLOSE(2,IER)       ;CLOSE RADAR DATA FILE
        CALL CLOSE(3,IER)       ;CLOSE GRID FILE
C RETURN FROM SWAP
        CALL BACK
        END
```

```
C*****************************************************************
C       QUADRANT CHANGE PROGRAM
C
C       VERSION 2.0  JULY 24,1979  MARK MATHEWSON--SSD WRH NWS
C*****************************************************************
C READS AND WRITES QUADRANTS FOR GRID V2.0 PGM. QUADRANTS STORED
C IN FILE DP1:GRID AND ARE CONTIGIOUS (12BLKS EACH).
        COMPILER NOSTACK
        SUBROUTINE QUADCHG (NEWQ,QUAD,IXBOT,IYBOT,RESOL)
        COMMON/DTA/GRID,DUM
        INTEGER GRID(0:54,0:54),DUM(100),Y(0:3),X(0:3)
        INTEGER QUAD,RESOL,INB(0:255)
        IF(NEWQ.EQ.QUAD) RETURN ;SAME QUAD WANTED
C ASSUMES GRID FILE IS OPENED ON CHANNEL 3
C WRITE OLD QUADRANT TO DISK FILE
        J=QUAD*12
        CALL WRBLK(3,J,GRID,12,IER)
C READ IN NEW QUADRANT
        QUAD=NEWQ
        J=QUAD*12
        CALL RDBLK(3,J,GRID,12,IER)
C SETUP XBOT AND YBOT
        X(0)=0
        X(1)=0
        X(2)=-RESOL*55
        X(3)=-RESOL*55
        Y(0)=0
        Y(1)=-RESOL*55
        Y(2)=-RESOL*55
        Y(3)=0
        IXBOT=X(QUAD)
        IYBOT=Y(QUAD)
        RETURN
        END
C*****************************************************************
C       CONVERTS BCD ANTENNA POSITION TO DECIMAL AND RADIANS
C
C       VERSION 1.0  JULY 9,1979   MARK MATHEWSON--SSD WRH NWS
C       VERSION 2.0  JULY 24,1979  MARK MATHEWSON--SSD WRH NWS
C        --NO MODIFICATIONS--USED FOR GRAPH V2.0 PGM.
C*****************************************************************
        COMPILER NOSTACK
        SUBROUTINE BCDRAD (IBCD,IDEC,RII)
C IBCD=INPUT VALUE, IDEC=OUTPUT DECIMAL VALUE, RII=RADIANS OUTPUT
        COMMON/CONVE/CONV,IBI
        INTEGER CONV(0:15),IBI(0:15)
        IDEC=0          ;RESULT INITILIZE TO ZERO
C CONVERT TO DECIMAL
        DO 3 I=0,15
        IF(IAND(IBI(I),IBCD).NE.0) IDEC=IDEC+CONV(I)
3       CONTINUE
        RII=FLOAT(IDEC)/572.957
        RETURN
        END
```

```
C*******************************************************************
C       GRIDBLOCK DLOCK DATA AREA FOR GRID PROGRAM
C
C       VERSION 2.0  JULY 24,1979  MARK MATHEWSON--SSD WRH NWS
C*******************************************************************
        COMPILER NOSTACK
        BLOCK DATA
        COMMON/CONVE/CONV,IBI
        INTEGER CONV(0:15),IBI(0:15)
        DATA CONV/8000,4000,2000,1000,800,400,200,100,80,40,20,10,8,4,2,1/
        DATA IBI/100000K,40000K,20000K,10000K,4000K,2000K,1000K,400K,200K,
     1100K,40K,20K,10K,4K,2K,1K/
        END
```

GRAPH

```
C*************************************************************************
C       CONVERSION PROGRAM FROM GRID ARRAY TO GRAPHICS
C
C    VERSION 2.1  JULY 28,1979   MARK MATHEWSON--SSD WRH NWS
C       --SIMILIAR TO V2.0; BUT DOES NOT USE GRAPHICS SUBROUTINES
C          SINCE ONLY TEXT IS OUTPUT.
C*************************************************************************
        COMPILER NOSTACK
        EXTERNAL DECASC,BCDRAD,STATUS
        COMMON/DTA/IGA,MTH
        COMMON/GD/GRID,DUMM
        INTEGER IGA(0:255),SPC(0:255),GRID(0:54,0:54),MTH(24)
        INTEGER SMSK,RESOL,QUAD,X(0:3),Y(0:3),BMSK,DUMM(100)
        EQUIVALENCE (GRID(0,0),SPC(0))
        SMSK=3777K      ;STATUS-ELEVATION MASK
        BMSK=377K       ;BYTE MASK
C OBTAIN PARAMETERS
        CALL OPEN(2,"DP1:PARAMETER",2,IER)      ;OPEN PARAMETER FILE
        CALL RDBLK(2,0,SPC,1,IER)       ;READ PARAMETERS
C RANGE BLANKING
        IGA(76)=IGA(76)+SPC(1)/100      ;100S.
        CALL DECASC(MOD(SPC(1),100),IGA(77))    ;10S,1S
C RESOLUTION
        RESOL=SPC(2)
        CALL DECASC(RESOL,IGA(87))      ;WRITE IT
C MAXIMUM DISPLAY RANGE
        I=RESOL*55      ;GRID SIZE=55
        IGA(98)=IGA(98)+I/100   ;100S.
        CALL DECASC(MOD(I,100),IGA(99)) ;10S,1S
C LEVELS
        DO 2 I=3,11
        IF(I.NE.3.AND.SPC(I).LE.SPC(I-1)) GO TO 10      ;EXIT
        IRT=100.*(10.**((SPC(I)-53.02)/32.))    ;RATE .01"/HR
        K=(I-3)*7+127   ;POSITION IN ARRAY
        IGA(K)=ISHFT(IRT/100+60K,8)+56K ;INCHES AND .
2       CALL DECASC(MOD(IRT,100),IGA(K+1))      ;.01,.1
10      CALL CLOSE(2,IER)       ;CLOSE PARAMETER FILE
C DATE-TIME GROUP
        CALL OPEN(2,"DP1:RADARDATA",2,IER)      ;OPEN RADARDATA FILE
        CALL RDBLK(2,0,SPC,1,IER)       ;READ IN FIRST BLOCK
        IGA(40)=MTH(SPC(2)*2-1) ;WRITE MONTH
        IGA(41)=MTH(SPC(2)*2)
        CALL DECASC(SPC(3),IGA(42))     ;WRITE DAY
        CALL DECASC(SPC(4),IGA(51))     ;WRITE HOUR
        CALL DECASC(SPC(5),IGA(52))     ;WRITE MINUTE
C STATUS-ELEVATION
        I=SPC(1)        ;ELEVATION-STATUS WORD
        CALL STATUS(I,I1,I5,I2,I3,I4)   ;SEPERATE STATUS BITS
        CALL DECASC(I3,IGA(109))        ;WRITE TIME SAMPLE
        IF(I1.EQ.0) GO TO 11    ;IF ATTN OFF
        IGA(110)="ON"
        IGA(119)="  "
11      I=IAND(I,SMSK)  ;ELEVATION ALONE
        CALL BCDRAD(I,JJ,RAD)   ;CONVERT TO DECIMAL
        J=JJ/10
        CALL DECASC(J,IGA(64))  ;ELEVATION TENS,UNITS
        IGA(65)=ISHFT(56K,8)+MOD(JJ,10)+60K     ;TENTHS, ELEVATION
        CALL CLOSE(2,IER)       ;CLOSE RADAR FILE
C
```

```
C
C  PART II OF GRAPHICS PROGRAM--PROCESS INTENSITY DATA
C SET UP PARAMETERS
      IBLK=0  ;BLOCK # FOR GRAPHICS OUTPUT
      IPT=188 ;WORD POINTER IN GRAPHICS ARRAY (IGA)
      IXCEN=4096/2     ;X CENTER OF SCREEN
      IYCEN=3072/2     ;Y CENTER OF SCREEN
      MAP=230          ;MAP SCALE RANGE
      SCALE=IYCEN/MAP ;SCALE=PTS/KM
C INSERTED FOR LAS VEGAS OFFSET MAP
      IXCEN=IXCEN+600 ;600=OFFSET FOR 1.0 DEG LONG.
      X(0)=0           ;XBOTTOM Y BOTTOM FOR QUADRANTS
      X(1)=0
      X(2)=-55*RESOL  ;55=GRID SIZE
      X(3)=-55*RESOL
      Y(0)=0
      Y(1)=-55*RESOL
      Y(2)=-55*RESOL
      Y(3)=0
C OPEN GRID FILE
      CALL OPEN(2,"DP1:GRID",2,IER)
C OPEN GRAPHICS OUTPUT FILE
      CALL OPEN(1,"DP1:NMCGPHRDR",2,IER)
C PROCESS INTENSITY DATA
      DO 102 QUAD=0,3
      IXBOT=X(QUAD)
      IYBOT=Y(QUAD)    ;BOTTOM OF QUADRANT
      J=QUAD*12        ;RECORD (BLOCK) START
      CALL RDBLK(2,J,GRID,12,IER)      ;READ IN QUADRANT
      DO 103 IX=0,54  ;PROCESS X COORDINATES
      XKM=IX*RESOL+IXBOT       ;XKM POSITION
      IXSCR=IXCEN+XKM*SCALE    ;X SCREEN COORDINATE
      IF(IXSCR.GT.4090.OR.IXSCR.LT.0) GO TO 103       ;SKIP ALL IY
      DO 104 IY=0,54  ;PROCESS Y COORDINATES
      IF(GRID(IX,IY).EQ.0) GO TO 104  ;LEVEL 0, SKIP CALCULATIONS
      YKM=IY*RESOL+IYBOT       ;YKM POSITION
      IYSCR=IYCEN+YKM*SCALE    ;Y SCREEN COORDINATE
      IF(IYSCR.GT.3070.OR.IYSCR.LT.0) GO TO 104       ;OUT OF GRID
C WRITE TO GRAPHICS ARRAY
C INSURE NO 203K OR 20K APPEAR IN IX,IY STRING. ADD 1 IF APPEARS:
      I=IAND(IXSCR,BMSK)       ;GET LOWER BYTE
      IF(I.EQ.20K.OR.I.EQ.203K) IXSCR=IXSCR+1 ;INCREMENT BY ONE
      I=IAND(IYSCR,BMSK)       ;GET LOWER BYTE
      IF(I.EQ.20K.OR.I.EQ.203K) IYSCR=IYSCR+1 ;INCREMENT BY ONE
      IGA(IPT)=142400K         ;TEXT COMMAND FOR AFOS
      IGA(IPT+1)=IXSCR         ;IX COORDINATE
      IGA(IPT+2)=IYSCR         ;IY COORDINATE
      IGA(IPT+3)=ISHFT(GRID(IX,IY)+60K,8)+40K ;LEVEL TO ASCII
      IPT=IPT+4        ;UPDATE BUFFER POINTER
      IF(IPT.NE.256) GO TO 104         ;BUFFER NOT FULL
C BUFFER FULL, WRITE IT
      CALL WRBLK(1,IBLK,IGA,1,IER)     ;WRITE GRAPHICS BUFFER
      IF(IER.NE.1) GO TO 446  ;ERROR-
      IPT=0    ;RESET BUFFER COUNTERS AND POINTERS
      IBLK=IBLK+1
104   CONTINUE
103   CONTINUE
102   CONTINUE
```

```
C END OF DATA,
C ADD 203K BYTE TO END, ZERO REMAINING BUFFER AND DISK FILE
      IGA(IPT)=203K    ;ADD ENDING BYTE
      IPT=IPT+1
      DO 444 I=IPT,255
444   IGA(I)=0         ;ZERO REMAINING BUFFER
      CALL WRBLK(1,IBLK,IGA,1,IER)     ;WRITE BUFFER
      DO 445 I=0,255   ;ZERO ENTIRE BUFFER
445   IGA(I)=0
      IBLK=IBLK+1
446   CALL WRBLK(1,IBLK,IGA,1,IER)     ;WRITE ALL ZEROS TO FILL FILE
      IF(IER.EQ.9) GO TO 450  ;FINISHED
      IBLK=IBLK+1
      GO TO 446        ;MORE TO DO.
C FINISH PROGRAM
450   CALL CLOSE(1,IER)
      CALL CLOSE(2,IER)
      CALL BACK        ;RETURN FROM SWAP
      END
C*************************************************************
C       CONVERTS BCD ANTENNA POSITION TO DECIMAL AND RADIANS
C
C       VERSION 1.0  JULY 9,1979   MARK MATHEWSON--SSD WRH NWS
C       VERSION 2.0  JULY 24,1979  MARK MATHEWSON--SSD WRH NWS
C        --NO MODIFICATIONS--USED FOR GRID V2.0 PGM.
C*************************************************************
      COMPILER NOSTACK
      SUBROUTINE BCDRAD (IBCD,IDEC,RII)
C IBCD=INPUT VALUE, IDEC=OUTPUT DECIMAL VALUE, RII=RADIANS OUTPUT
      COMMON/CONVE/CONV,IBI
      INTEGER CONV(0:15),IBI(0:15)
      IDEC=0           ;RESULT INITILIZE TO ZERO
C CONVERT TO DECIMAL
      DO 3 I=0,15
      IF(IAND(IBI(I),IBCD).NE.0) IDEC=IDEC+CONV(I)
3     CONTINUE
      RII=FLOAT(IDEC)/572.957
      RETURN
      END
```

```
C*******************************************************************
C       BLOCK DATA FOR GRAPH PROGRAM
C
C  VERSION 2.1  JULY 28,1979  MARK MATHEWSON--SSD WRH NWS
C       --REPLACES V2.0, CONTAINS ACTUAL GRAPHICS DATA PRE-ASSEMBLED
C         IN AFOS FORMAT.  ELIMINATES NEED FOR GRAPHICS SUBROUTINES.
C         UTF NOT NEEDED SINCE TEXT IS ONLY OUTPUT.
C*******************************************************************
        COMPILER NOSTACK
        BLOCK DATA
        COMMON/DTA/IGA,MTH
        COMMON/CONVE/CONV,IBI
        INTEGER CONV(0:15),IBI(0:15)

        INTEGER IGA(0:255),MTH(24)
        DATA MTH/'JAN FEB MAR APR MAY JUN JUL AUG SEP OCT NOV DEC '/
        DATA IGA/'NMCGPHRDR000',2*177777K,2400K,140400K,0,10920K,
     114K,2*0,303K,3*0,4000K,13K,177017K,175000K,0,24K,1020K,10906K,
     20,142400K,143504K,5524K,'RADAR ',142400K,62K,1130K,
     3'DATE= XXX XX     ',142400K,62K,1034K,'TIME= XXXXZ     ',142400K,
     462K,740K,'ELEVATION=00.0  ',142400K,62K,644K,'RANGE BLANK= 000',
     5142400K,62K,550K,'RESOLUTION= 00  ',142400K,62K,454K,
     6'MAX DISP RGE=000',142400K,62K,360K,'TIME SAMPLE=00  ',142400K,62K,
     7264K,'IF ATTN=OFF      ',142400K,62K,36K,'LV1=X.XX',142400K,702K,
     836K,'LV2=X.XX',142400K,1522K,36K,'LV3=X.XX',142400K,2342K,36K,
     9'LV4 X.XX',142400K,3162K,36K,'LV5=X.XX',142400K,4002K,36K,
     1'LV6=X.XX',142400K,4622K,36K,'LV7=X.XX',142400K,5442K,36K,
     2'LV8 X.XX',142400K,6262K,36K,'LV9=X.XX      ',203K,67*0/
        DATA CONV/8000,4000,2000,1000,800,400,200,100,80,40,20,10,8,4,2,1/
        DATA IBI/100000K,40000K,20000K,10000K,4000K,2000K,1000K,400K,200K,
     1100K,40K,20K,10K,4K,2K,1K/
        END
```

```
C*************************************************************
C       CONVERSION FROM TWO-DIGIT INTEGER TO TWO-BYTE ASCII
C
C       VERSION 2.0  JULY 26,1979  MARK MATHEWSON--SSD WRH NWS
C*************************************************************
        COMPILER NOSTACK
        SUBROUTINE DECASC(I,IASOU)
C  I=INPUT VALUE, IASOU=OUTPUT WORD IN ASCII
        IASOU=ISHFT(I/10+48,8)+48+MOD(I,10)
        RETURN
        END



C*************************************************************
C       CONVERT ELEVATION-STATUS WORD TO STATUS INDICATORS
C
C       VERSION 1.0  JULY 5,1979  MARK MATHEWSON--SSD WRH NWS
C       VERSION 2.0  JULY 26,1979 MARK MATHEWSON--SSD WRH NWS
C        --NO MODIFICATIONS FROM V1.0
C*************************************************************
        COMPILER NOSTACK
        SUBROUTINE STATUS (IN,ATT,RGEINT,STC,TMSAM,TEST)
        INTEGER RGEINT,STC,TMSAM,TEST,ATT
C  "IN" IS ELEVATION STATUS WORD INPUT
C  INITIALIZE STATUS INDICATORS
        MATT=400000K
        MRGE=200000K
        MSTC=400K
        MTMSAM=10000K
        MTST=100000K
        ATT=0             ;IF ATTENUATORS
        RGEINT=2          ;RANGE INTERVAL
        STC=0             ;STC MONITOR
        TMSAM=15          ;TIME SAMPLE MONITOR
        TEST=1            ;TEST MODE MONITOR
C  CHECK STATUS BITS
        I=IAND(IN,MATT)
        IF(I.NE.0) ATT=1
        I=IAND(IN,MRGE)
        IF(I.NE.0) RGEINT=1
        I=IAND(IN,MSTC)
        IF(I.NE.0) STC=1
        I=IAND(IN,MTMSAM)
        IF(I.NE.0) TMSAM=31
        I=IAND(IN,MTST)
        IF(I.NE.0) TEST=0
        RETURN
        END
```

PARAMETER

```
C*******************************************************************
C       RADAR PARAMETER CHANGE PROGRAM
C
C   VERSION 1.0   JULY 8,1979    MARK MATHEWSON--SSD WRH NWS
C   VERSION 2.0   JULY 28,1979   MARK MATHEWSON--SSD WRH NWS
C      --MODIFICATIONS--BLK#1 CONTAINS LEVEL ARRAY FOR V2.0 GRID
C        PGM. ELIMINATED REQUESTS FOR CERTAIN PARAMETERS UNUSED
C        IN PRESENT VERSION.
C
C*******************************************************************
C
C
C PROGRAM ASKS USER FOR VARIOUS RADAR LOGGING AND DISPLAY
C PARAMETERS AND THEN STURES THOSE PARAMETERS INTO FILE
C DP1:PARAMETER.  THIS IS NOT THE SAME ROUTINE THAT ALLOWS
C USERS TO CHANGE PARAMETERS FROM AFOS.  THIS ROUTINE ALLOWS
C MORE CHANGES TO BE MADE.
C
      INTEGER P(0:255),LEVEL(0:255)
C OPEN PARAMETER FILE
      CALL OPEN (1,"DP1:PARAMETER",2,IER)
      IF(IER.EQ.1) GO TO 1
      TYPE "PARAMETER FILE OPEN FAIL. IER=",IER
      STOP
C OBSERVATION INTERVAL (RANGE 5 TO 60 MINUTES)
1     ACCEPT "OBSERVATION INTERVAL (MIN) ",P(0)
      IF(P(0).LE.60.OR.P(0).GE.5) GO TO 2
      TYPE "OBS INT OUT OF BOUNDS. RANGE 5-60 MIN."
      GO TO 1
C RANGE BLANKING (ANY RANGE)
2     ACCEPT "RANGE BLANKING (KM) ",P(1)
      IF(P(1).GE.0) GO TO 3
      TYPE "RANGE BLANKING MUST BE >= 0"
      GO TO 2
C RESOLUTION DESIRED (RANGE 1 TO 20 KM)
3     ACCEPT "RESOLUTION (KM) ",P(2)
      IF(P(2).GE.1.AND.P(2).LE.20) GO TO 6
      TYPE "RESOLUTION OUT OF BOUNDS (RANGE 1-20 KM)"
      GO TO 3
C RADAR THRESHOLD
6     ACCEPT "RADAR CALIBRATION THRESHOLD (0-200)      ",P(19)
      IF(P(19).GE.0.OR.P(19).LE.200) GO TO 8
      TYPE "RADAR CAL THRESH OUT OF BOUNDS (0-200)"
      GO TO 6
```

```
C LEVELS FOR THRESHOLDS
8       DO 14 I=3,11
14      P(I)=0
        WRITE(10,15)
15      FORMAT(' ','THRESHOLD LEVELS..ENTER ZERO TO STOP')
        DO 16 I=1,9
        WRITE(10,21) I
21      FORMAT(' ENTER THRESHOLD FOR LEVEL',I2,5X,Z)
        J=I+2
        ACCEPT  P(J)
        IF(P(J).LE.0.OR.P(J).GT.250) GO TO 20
16      CONTINUE
C WRITE NEW PARAMETERS TO DISK
20      CALL CLOSE(10,IER)
        CALL WRBLK (1,0,P,1,IER)
        IF(IER.NE.1) TYPE "PARAMETER WRITE ERROR. IER=",IER
C PRODUCE TABLE OF LEVELS FOR USE IN GRID V2.0 PGM.
C FIND MAXIMUM LEVEL DESIRED
        MAX=0
        DO 500 I=3,11
        IF(P(I).LE.MAX) GO TO 543
        JK=I-2
500     MAX=P(I)
C INITILIZE LEVELS TO MAX LEVELS
543     DO 409 I=0,255
409     LEVEL(I)=JK
C FORM TABLE TO CONVERT DATA TO LEVELS
        PT=P(3)
          I1=0
          DO 502 J=1,JK
          I2=P(J+2)+PT
          DO 504 I=I1,I2
          LEVEL(I)=J-1
504       CONTINUE
502       I1=I2
        CALL WRBLK(1,1,LEVEL,1,IER)
C CLOSE ALL FILES AND EXIT
        CALL RESET
        STOP
        END
```

CONTROL

```
C*****************************************************************
C       RADAR-AFOS CONTROL PROGRAM
C
C       VERSION 2.0  JULY 28,1979  MARK MATHEWSON--SSD WRH NWS
C
C RADAR CONTROL PROGRAM:
C       1. READS PARAMETER FILE TO DETERMINE OBSERVATION INTERVAL
C       2. SCHEDULES RADAR OBSERVATION, GRID, AND GRAPH PROGRAM
C       3. NO ERROR CHECKING IS INCORPORATED FOR VALID DATA.
C       4. SENDS FINISHED PRODUCT TO AFOS BY $TTO LINE
C*****************************************************************
        COMPILER NOSTACK
        INTEGER IJ(0:255)
C START-PROGRAM--INSURE DIRECTORY DP1 IS INITED
        CALL INIT ("DP1",0,IER)
C START SEQUENCE
C READ IN OBSERVATION INTERVAL
1       CALL OPEN(1,"DP1:PARAMETER",2,IER)
        IF(IER.NE.1) GO TO 1000
        CALL RDBLK(1,0,IJ,1,IER)
        IF(IER.NE.1) GO TO 1000
        INTERV=IJ(0)     ;OBS INTERVAL
        CALL CLOSE(1,IER)
        IF(IER.NE.1) GO TO 1000
C GET PRESENT TIME, CHECK FOR OBS TIME
2       CALL FGTIM(IHR,IMIN,ISEC)        ;PRESENT TIME
        IF(MOD(IMIN,INTERV).EQ.0) GO TO 10       ;OBS TIME
        GO TO 2          ;WAIT TILL OBSERVATION TIME
C OBSERVATION TIME
10      CALL SWAP ("DP0:RADARLOG.SV",IER)        ;GET DATA
        IF(IER.NE.1) GO TO 1000
        CALL SWAP ("DP0:GRID.SV",IER)    ;GRID THE DATA
        IF(IER.NE.1) GO TO 1000
        CALL SWAP ("DP0:GRAPH.SV",IER)   ;GRAPH THE DATA
        IF(IER.NE.1) GO TO 1000
C SEND THE DATA TO $TTO
        CALL OPEN (1,"DP1:NMCGPHRDR",2,IER)
        IF(IER.NE.1) GO TO 1000
        CALL OPEN (10,"$TTO",2,IER,64)
        IF(IER.NE.1) GO TO 1000
        IBLK=0  ;BLOCK COUNTER
11      CALL RDBLK(1,IBLK,IJ,1,IER)
        IF(IER.EQ.9) GO TO 50
        IF(IER.NE.1) GO TO 1000
C CHECK DATA FOR ALL ZEROS (END OF DATA)
        J=0
        DO 12 I=0,255
12      IF(IJ(I).NE.0) J=J+1
        IF(J.EQ.0) GO TO 50      ;END OF TRANSFER
        CALL WRITR(10,0,IJ,8,IER)        ;TRANSFER DATA.
        IF(IER.NE.1) GO TO 1000
        IBLK=IBLK+1      ;INCREMENT BLOCK COUNT
        GO TO 11         ;TRANSFER NEXT BLOCK
C END OF SEQUENCE
50      CALL CLOSE(10,IER)       ;CLOSE $TTO
        CALL CLOSE(1,IER)        ;CLOSE NMCGPHRDR FILE
        GO TO 1          ;RESTART SEQUENCE
C
C ERROR
1000    STOP     ;EXIT PROGRAM
        END
```

IBM 360/168
Map extraction
Program

```
      IMPLICIT REAL*8 (A-Z)
      REAL*8 LON1,LAT1,LON2,LAT2
      INTEGER II,REG1,I,REG2,ST1,ST2,X1SC,Y1SC,X2SCD,Y2SCD
      INTEGER IXCEN,IYCEN
      INTEGER CNT(20)/20*0/
C PARAMETERS
C SCALE IS RADAR RANGE MAXIMUM IN KM
      SCALE=450.
C FULL IS FULL SCALE PIXEL SIZE
      FULL=3072.
C IXCEN AND IYCEN ARE CENTER PIXEL LOCATIONS
      IXCEN=2048
      IYCEN=1536
C RADLAT,RADLON RADAR LATITUDE AND LONGITUDE
      RADLAT=34.056/57.2957795
      RADLON=118.4475/57.2957795
      RMAX=SCALE*1.3
      A=FULL/SCALE/2.
C READ DATA
1     READ(1,2,END=100) REG1,ST1,REG2,ST2,LON1,LAT1,LON2,LAT2
2     FORMAT(12X,2(2I2,3X),4F6.4)
      CNT(REG1)=CNT(REG1)+1
      CALL DIRDIS (DIR,DIST,RADLAT,RADLON,LAT1,LON1)
      IF(DIST.GT.RMAX) CALL DIRDIS(DIR1,DIST1,RADLAT,RADLON,LAT2,LON2)
      IF(DIST.GT.RMAX.AND.DIST1.GT.RMAX) GO TO 1
      II=ST1-ST2
C NEXT STATEMENT FOR ONLY STATES
C     IF(II.EQ.0) GO TO 1
      X1KM=DIST*DSIN(DIR)
      Y1KM=DIST*DCOS(DIR)
      X1SC=IXCEN+IDINT(X1KM*A)
      Y1SC=IYCEN+IDINT(Y1KM*A)
      CALL DIRDIS(DIR,DIST,LAT1,LON1,LAT2,LON2)
      X2KM=DIST*DSIN(DIR)
      Y2KM=DIST*DCOS(DIR)
      X2SCD=IDINT(X2KM*A)
      Y2SCD=IDINT(Y2KM*A)
      WRITE(17,3) II,X1SC,Y1SC,X2SCD,Y2SCD,X1KM,Y1KM,X2KM,Y2KM
3     FORMAT(' ',I3,2X,4(I4,1X),4F10.1)
      GO TO 1
100    DO 101 I=1,10
101   WRITE(6,102) I,CNT(I)
102   FORMAT(' COUNT FOR REGION ',I5,' IS ',I6)
      STOP
      END
```

```
SUBROUTINE DIRDIS(DIR,DIST,LATS,LONS,LATD,LOND)
IMPLICIT REAL*8 (A-Z)
B=1.57079632700-LATS
A=LONS-LOND
C=1.57079632700-LATD
TMPP=DCOS(B)*DCOS(C)+DSIN(B)*DSIN(C)*DCOS(A)
IF(TMPP.GT.1.0D00) TMPP=1.0D00
IF(TMPP.LT.-1.0D00) TMPP=-1.0D00
DIST=DARCOS(TMPP)
TMPP=(DCOS(C)-DCOS(DIST)*DCOS(B))/(DSIN(DIST)*DSIN(B))
IF(TMPP.GT.1.0D00) TMPP=1.0D00
IF(TMPP.LT.-1.0D00) TMPP=-1.0D00
DIR=DARCOS(TMPP)
DIST=DIST*6.3598315D03
IF(DSIN(A).LT.0.0D00) DIR=4.0D00*1.57079636327D00-DIR
RETURN
END
```

```
C*************************************************************************
C  PRODUCE AFOS GRAPHICS PRODUCT ---COUNTY MAP BACKGROUND
C
C  ASSUMES INPUT DATA IS FROM PSU MAP EXTRACTION PROGRAM
C  AND IS IN FILE INDATA
C
C  VERSION 1.0  JULY, 1979  MARK MATHEWSON--SSD WRH NWS
C*************************************************************************
      EXTERNAL GPD1,COMMH,UTF,DELTAVEC
      COMMON/DTA/IGA,ICOUNT,IREC
      INTEGER IGA(500)
      CALL OPEN(3,"INDATA",2,IER,80)  ;OPEN FILE
      IGA(1)='WR'
      IGA(2)='HG'  ;COMMS HEADER
      IGA(3)='PH'
      IGA(4)='80'
      IGA(5)='80'
      IGA(6)='00'
      CALL COMMH
      CALL GPD1
12    READ(3,2,END=100) (IGA(J),J=1,4)
2     FORMAT(6X,4(I4,1X))
      CALL DELTAVEC
      GO TO 12
100   CALL CLOSE (3,IER)
      CALL UTF
      STOP
      END
```

```
C*************************************************************************
C       COMM HEADER SUBROUTINE FOR AFOS GRAPHICS PACKAGE
C
C       VERSION 1.0   JULY 14,1979  MARK MATHEWSON--SSD WRH NWS
C
C  [illegible] WRITTEN BY ALEXA[illegible] MCPS-COMPILER AND JIM FORS
C  OF [illegible] PROGRAMS WRITTEN FOR [illegible].
C  [illegible] WRITES FILE NAME [illegible] AND OPENS
C  OUTPUT FILE [illegible]
C--D [illegible] MUST BE PROVIDED IN IGA(1) TO IGA(5)
C*************************************************************************
      COMPILER MODTRACK
      SUBROUTINE COMMH
      COMMON/DTA/IGA,ICOUNT,IREC
      INTEGER IGA(500)
      IGA(7)=300608
      IGA(8)=-1
C OPEN OUTPUT FILE (DP1:AFOSGPH.DT)
      CALL DFILW("DP1:AFOSGPH.DT",IER)
      CALL CFILW("DP1:AFOSGPH.DT",2,IER)
      CALL OPEN(1,"DP1:AFOSGPH.DT",2,IER,2)
      CALL WRITR(1,IREC,IGA,8,IER)
      IREC=IREC+8
      RETURN
      END
```

```
C*************************************************************
C     TEXT WRITING SUBROUTINE FOR AFOS GRAPHICS
C
C  VERSION 1.0  JULY 14,1979  MARK MATHEWSON--SSD WRH NWS
C
C ADAPTED FROM ROUTINES WRITTEN BY ALEXANDER MACDONALD AND
C JIM FORS OF SSD WRH NWS.  THESE ROUTINES WRITTEN FOR RUNNING
C ON NOVA312.
C
C TEXT WRITING ROUTINE WRITES TEXT IN THREE SIZES SPECIFIED BY
C ISIZE=0,1,2. IX AND IY SPECIFY STARTING COORDINATES. IJ=#WORDS
C*************************************************************
      COMPILER NOSTACK
      SUBROUTINE TEXT(IJ,ISIZE,IX,IY)
      COMMON/DTA/IGA,ICOUNT,IREC
      INTEGER ZZ(3),IGA(500)  ; ZZ IS TEMPORARY ARRAY
      ICOUNT=IJ
      ZZ(1)=142400K    ;TEXT ASCII INSTRUCTION
      ZZ(2)=IX         ;IX START COORDINATE
      ZZ(3)=IY         ;IY START COORDINATE
      IF(ISIZE.EQ.1.OR.ISIZE.EQ.3) CALL ISET(ZZ(2),14)
      IF(ISIZE.EQ.2.OR.ISIZE.EQ.3) CALL ISET(ZZ(2),15)
      CALL WRITR(1,IREC,ZZ,3,IER)
      IREC=IREC+3
C WRITE TEXT INTO BUFFER
      CALL WRITR(1,IREC,IGA,ICOUNT,IER)
      IREC=IREC+ICOUNT         ;UPDATE RECORD COUNTER
      RETURN
      END
C*************************************************************
C    DELTAVEC DRAWS A SINGLE VECTOR FOR THE AFOS GRAPHICS PACKAGE
C
C    VERSION 1.0  JULY 14,1979  MARK MATHEWSON--SD WRH NWS
C
C DELTAVEC DRAWS A SINGLE VECTOR GIVEN THE STARTING COORDINATES
C AND THE RELATIVE X AND RELATIVE Y CHANGES.
C--WRITTEN FOR THE NOVA312.
C IGA(1)=X START,IGA(2)=Y START,IGA(3)=DELTA X,IGA(4)=DELTA Y
C*************************************************************
      COMPILER NOSTACK
      SUBROUTINE DELTAVEC
      COMMON/DTA/IGA,ICOUNT,IREC
      INTEGER I(6),IGA(500)
      MSK=17777K
C WRITE INSTRUCTION SET
      I(1)=141400K     ;VECTOR RELATIVE INSTRUCTION
      I(2)=IGA(1)
      I(3)=IGA(2)
      I(4)=2
      I(5)=IGA(3)      ;DELTA X
      I(6)=IGA(4)      ;DELTA Y
      DO 1 JJ=5,6
      I(JJ)=IAND(I(JJ),MSK)
1     CONTINUE
      CALL WRITR(1,IREC,I,6,IER)
      IREC=IREC+6
      RETURN
      END
```

```
C*************************************************************************
C       GRAPHIC PRODUCT DEFINITION FOR AFOS GRAPHIC PACKAGE
C
C       VERSION 1.0  JULY 14,1979  MARK MATHEWSON--SSD WRH NWS
C
C ADAPTED FROM ROUTINES WRITTEN BY ALEXANDER MACDONALD AND
C JIM FORS OF SSD WRH NWS.  THIS ROUTINE WRITTEN TO RUN ON
C NOVA312.
C
C THIS SUBROUTINE DEFINES GRAPHIC PRODUCT AND DRAWS BOX
C AROUND BOUNDARY.
C*************************************************************************
        COMPILER NOSTACK
        SUBROUTINE GPD1
        COMMON/DTA/IGA,ICOUNT,IREC
        INTEGER IGA(500)
C PRODUCE GRAPHIC PRODUCT DEF.
        IGA(1)=2400K     ;NECESSARY FOR UNKNOWN REASON
        IGA(2)=140400K   ;GRAPH PROD DEFIN.
        IGA(3)=0         ;GEOGRAPHY SCALE
        IGA(4)=10000K    ;MAXIMUM I COORDINATE
        IGA(5)=6000K     ;MAXIMUM J COORDINATE
        IGA(6)=0         ;DAY-MONTH-YEAR
        IGA(7)=0         ;TIME-PDC
C PRODUCE BOX AROUND SCREEN
        IGA(8)=141400K   ;RELATIVE VECTOR INSTRUCTION
        IGA(9)=0         ;I COORDINATE START
        IGA(10)=0        ;J COORDINATE START
        IGA(11)=8        ;8 WORDS FOLLOW
        IGA(12)=0        ;DRAW UP
        IGA(13)=3070     ;IY
        IGA(14)=4090     ;IX
        IGA(15)=0        ;IY
        IGA(16)=0        ;DRAW DOWN
        IGA(17)=-3070
        IGA(18)=-4090
        IGA(19)=0
C CLEAR NECESSARY BITS FOR CORRECT INSTRUCTIONS
        DO 1 JJ=17,18
        CALL ICLR(IGA(JJ),15)
        CALL ICLR(IGA(JJ),14)
        CALL ICLR(IGA(JJ),13)
1       CONTINUE
C WRITE BUFFER TO DISK
        CALL WRITR(1,IREC,IGA,19,IER)
        IREC=IREC+19
        RETURN
        END
```

```
C*********************************************************************
C     MOD BLOCKDATA SUBPROGRAM
C
C     VERSION 1.0   JULY 14,1979   MARK MATHEWSON--SSD WRH NWS
C
C  ADAPTED FROM GRAPHICS PROGRAMS WRITTEN BY ALEXANDER MACDONALD
C AND JIM FORS OF SSD WRH NWS.  THESE PROGRAMS MADE TO RUN ON
C NOVA312.
C*********************************************************************
      COMPILER NOSTACK
      BLOCK DATA
      COMMON/DTA/IGA,ICOUNT,IREC
      INTEGER IGA(500),ICOUNT,IREC
C IGA=INPUT DATA BUFFER, ICOUNT=VALID DATA COUNT, IREC=#REC IN OUTPUT FILE
      DATA IGA/500*0/
      DATA ICOUNT/0/
      DATA IREC/0/
      END
```

MDB INTERFACE BOARD

General Purpose Interface Board for Data General or similar type computers

BASIC BOARD

Includes BUSY/DONE logic, interrupt request, acknowledge and mask logic. All data lines are buffered with gates to allow multiplexing data to the computer by "OR" tying input data at the input of the buffer gates rather than on the computer data line bus.

Multiple Device Controllers—The printed circuit logic is used for *one* device controller but all device select (address) lines and all control signals from the computer are buffered and accessible via wirewrap posts so that other device controllers can be mechanized on the wirewrap socket portion of the board. This includes the capability of routing the interrupt and data channel priority lines to the extra device controllers. The GPI/O manufactured by Data General board does not allow for a convenient method of mechanizing extra device controllers on its board.

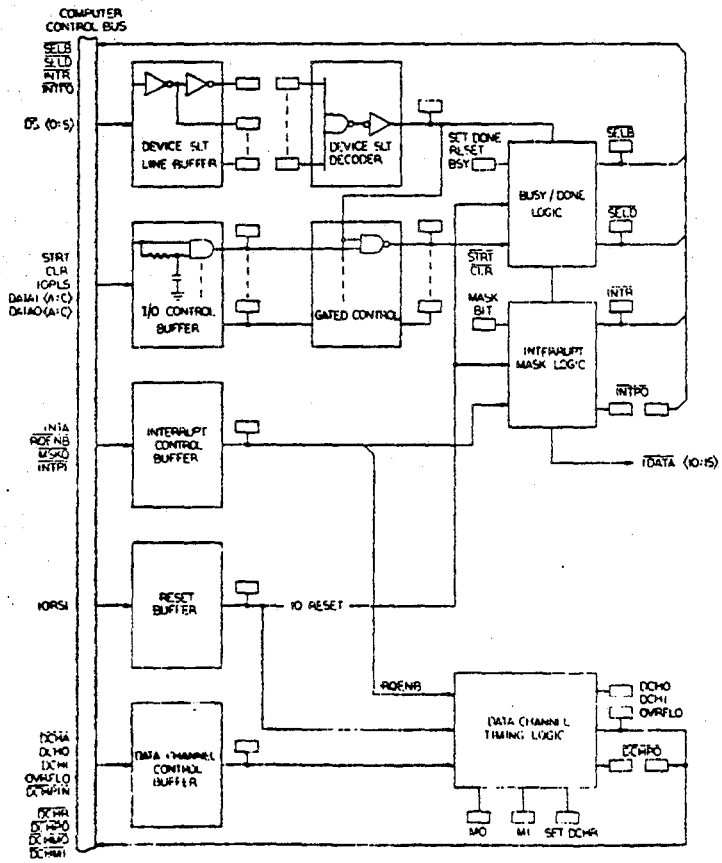
I/O DATA REGISTERS

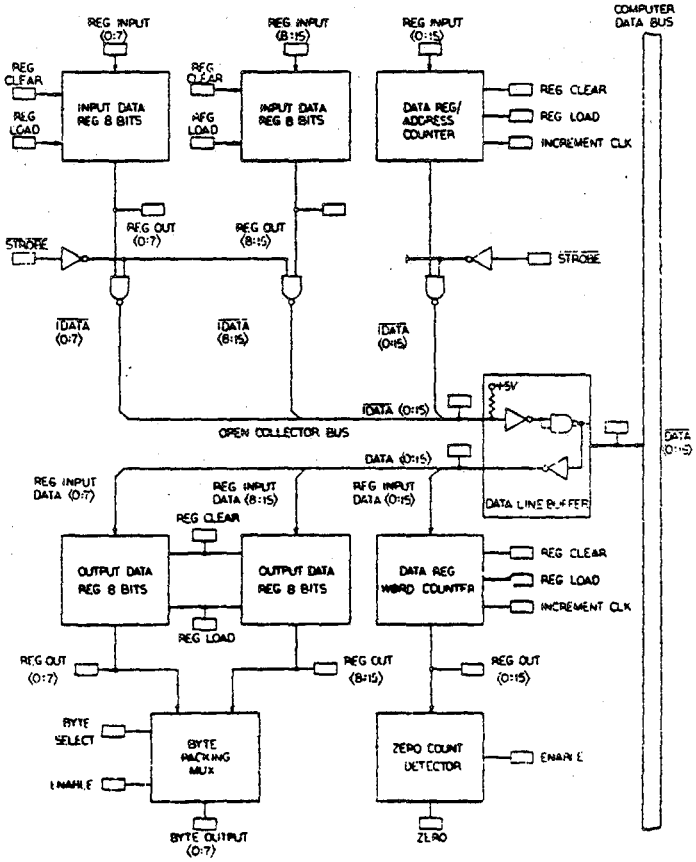
There are two general 16 bit I/O registers—one for input and one for output. The input data register may be loaded with a 16 bit word or each byte may be loaded separately for byte unpacking. The output data register is loaded with a 16 bit word from the computer and may be read by the user as a 16 bit word or may be read as two 8 bit bytes using the byte packing multiplexer included on the printed circuit logic.

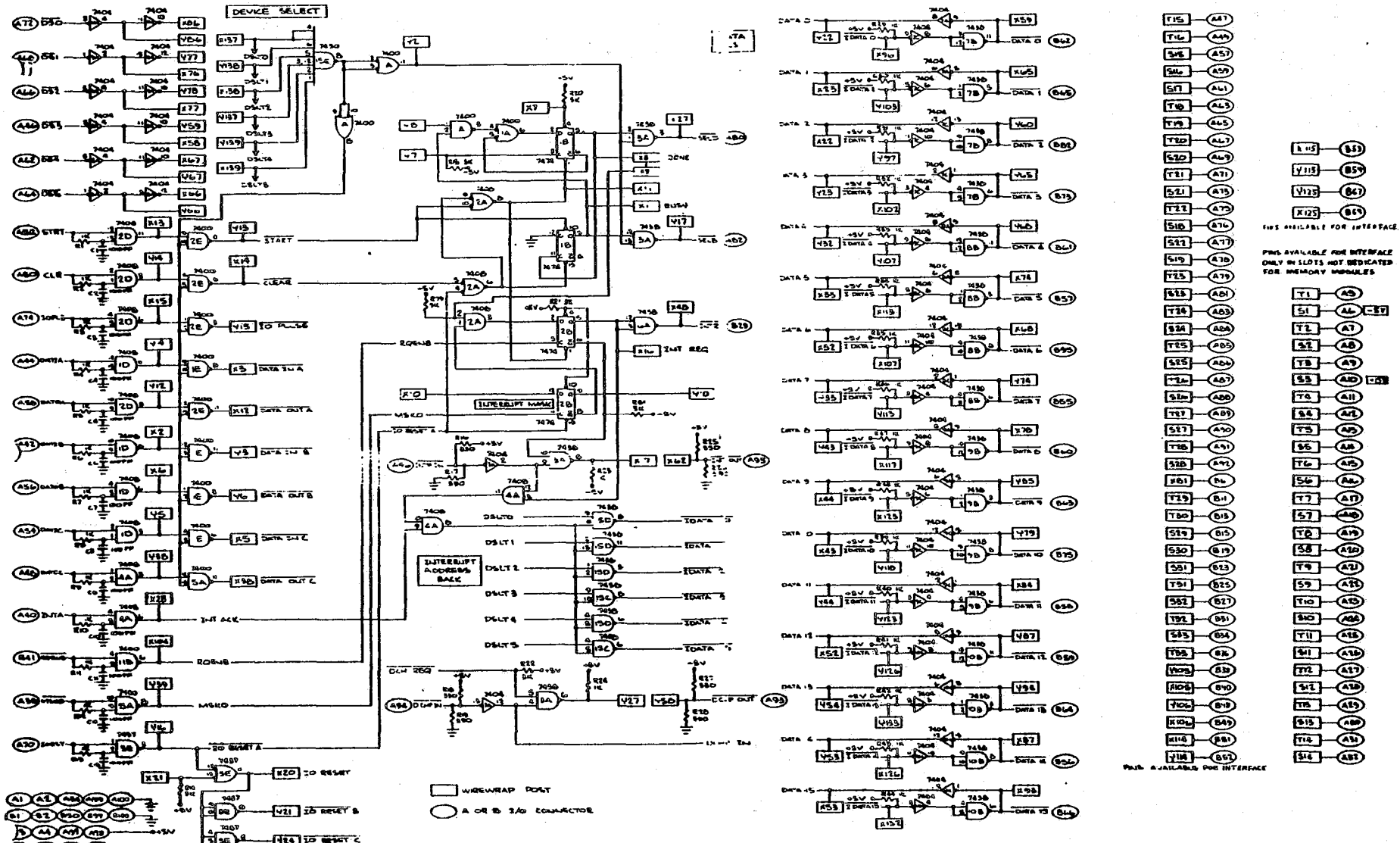
There are two additional 16 bit registers that may be used as a general input data register and a general output data register or may be employed as an address counter register and a word counter register when the user is interfacing to the computers data channel. The word counter register is also connected to a zero word count detection circuit for terminating the block data channel transfer.

DATA CHANNEL CONNECTION

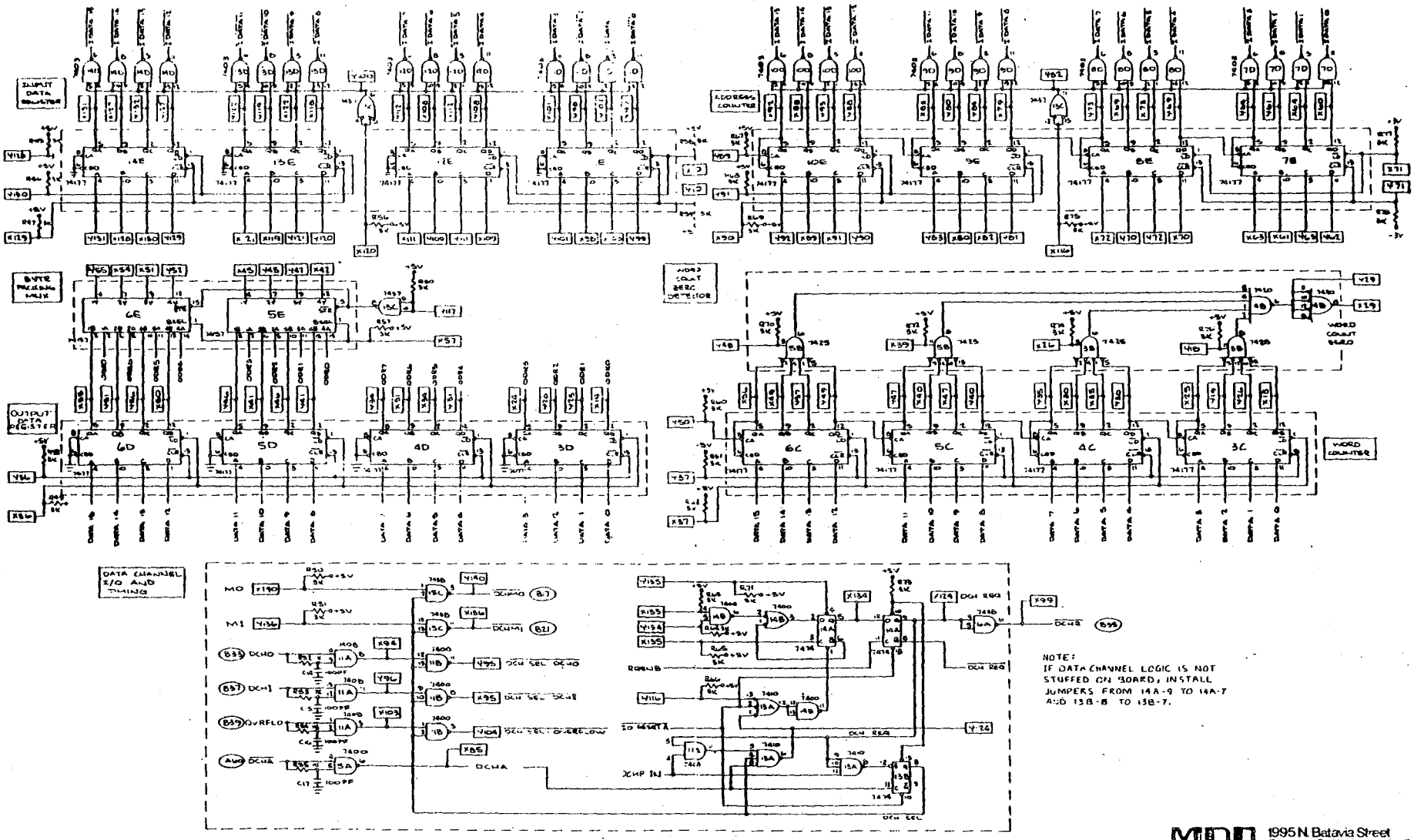
Includes data channel synchronization and request and acknowledge logic. Data channel control signals are buffered and accessible by wirewrap posts so that multiple device controllers may be mechanized for data channel interface.






DEVICE SELECT
START
CLEAR
INT REQ
INT ACK
RQBUS
MSKO
BUSY
DONE
INTERRUPT MASK
INTERRUPT ADDRESS BACK
WIREWRAP POST
A OR B I/O CONNECTOR

-66-



MDB SYSTEMS INC.
1995 N. Batavia Street
Orange, California 92665
714-998-6900
TWX 910-593-1339

TECHNICAL DESCRIPTION
GENERAL-PURPOSE INTERFACE BOARD

INTRODUCTION

This technical description gives a general description of the MDB Systems General Purpose Interface Board (GPIB), and provides information for configuring and using the GPIB.

The GPIB contains basic interface logic for communicating with the I/O bus of a Data General computer (or other computers emulating a Data General computer), plus facilities for building user-designed and furnished device interface logic.

The GPIB offers the user an economical means of building a computer I/O bus-to-device interface without the need to build the bus interface common to all device controllers.

Approximately one-half of the module area provides wirewrap and printed circuit facilities for installing a large number of DIP devices and discrete components forming the device interface. The other half of the module contains logic to interface the user-built logic with the Data General I/O bus.

This technical description assumes that the user is familiar with the Data General I/O bus interface and has a general knowledge of device controllers working with the bus.

PHYSICAL DESCRIPTION

The GPIB is a 15-inch-square printed circuit board built to plug into any Data General computer or expansion chassis. Even with user-built logic, it requires only a single slot position in the chassis.

The rear half of the module contains I/O bus interface logic. Data, address, interrupt priority, and other control lines that may be used to interface with user-built logic appear at two rows of wirewrap pins (rows X and Y) at the center of the module.

The remaining half of the module contains positions for as many as 105 fourteen- or sixteen-pin DIP devices making up user-built logic. Device positions are arranged in columns numbered 1 through 15, and seven rows designated F, H, J, K, L, M, and N, respectively.

Columns (except columns 4, 8, and 12) accommodate standard 16-pin DIP devices with pins on 0.3-inch centers; except in row N, where columns 1, 2, 3, 13, 14, and 15 accommodate only 14-pin devices.

Sixteen-pin positions have pin 8 etched to the ground plane on the component side of the board, and have pin 16 etched to the +5V (Vcc) bus on the solder side. On the solder side an etch conductor connects pins 7 and 8 together, accommodating 14-pin devices with standard ground and Vcc pin assignments. Wherever a position is used for a 16-pin device, the etch between pins 7 and 8 must be cut.

When a device has non-standard power and ground pin assignments, the power and ground etch must be cut as required before the device is installed.

The six 14-pin device locations (row N, columns 1, 2, 3, 13, 14, and 15) have pin 7 connected to the ground plane, and pin 14 connected to the Vcc plane.

Between rows are pads in which the user may install ceramic decoupling capacitors with leads spaced 0.3 inch apart. Pairs of pads spaced 0.7 inch apart, at both sides of the module, accommodate axial-lead tantalum decoupling capacitors.

Columns 4, 8, and 12 have pads at 0.1-inch intervals, through all rows, to accommodate DIP devices having pin spacing of 0.3, 0.4, or 0.6 inch. Pads are etched to ground and to the Vcc plane to connect pin 8 of a 16-pin device to ground, and connect pin 16 to Vcc. Using other than 16-pin devices, these connections may need to be cut. Using 14-pin devices connect pin 7 to the ground plane.

The two rows of wirewrap pins that interconnect I/O bus logic and user-built logic are designated X and Y, with pins in each row numbered 1 through 140. Some of these pins are connected to unassigned fingers on the module's B connector.

Near the A and B connectors are two rows of pins (rows S and T), numbered 1 through 33. These pins are connected to unassigned pins on A and B connectors. The user may use these connections to connect the GPIB to the peripheral device. The user must, in this case, provide a cable between the chassis backplane and the device. Table 1 lists wirewrap pins in rows X, Y, S, and T, and the backplane connector pins to which each is connected.

At the front edge of the module are pad patterns to accommodate two ribbon cable connectors (P1 and P2) having 20, 26, 34, 40, or 50 pins. Refer to the MDB price list for ordering information.

The user will, generally, install wirewrap pins on the wirewrap area of the module. However, the user may order the Wirewrap Option (MDB-4043) which furnishes wirewrap pins installed in all DIP device positions. Pins are mounted on the component side of the board to minimize space requirements in the chassis.

Another option (MDB-4044) furnishes both installed wirewrap pins and low-profile DIP sockets in all device positions. Unless otherwise ordered, 16-pin sockets are installed in all positions except the six 14-pin positions in row N, where 14-pin sockets are installed.

An optional Board Cover (MDB-1024) may be furnished to protect wirewrap wire from interference by the module mounted in the next-higher slot in the chassis.

Table 1. Pins Available for Device Interface Connection

Wirewrap Pin	Backplane Pin	Wirewrap Pin	Backplane Pin	Wirewrap Pin	Backplane Pin
T15	A47	S23	A81	S31	B23
T16	A49	T24	A83	T31	B25
S15	A57	S24	A84	S32	B27
S16	A59	T25	A85	T32	B31
S17	A61	S25	A86	S33	B34
T18	A63	T26	A87	T33	B36
T19	A65	S26	A88	Y105	B38
T20	A67	T27	A89	X105	B40
S20	A69	S27	A90	Y106	B48
T21	A71	T28	A91	X106	B49
S21	A73	S28	A92	X114	B51
T22	A75	X81	B6	Y114	B52
S18	A76	T29	B11	X115	B53
S22	A77	T30	B13	Y115	B54
S19	A78	S29	B15	Y125	B67
T23	A79	S30	B19	X125	B69
The following pins are available only at slots not dedicated for memory modules, and also in the MDB Expansion Chassis MDB-NEC-01.					
T1	*A5	S6	A16	T11	*A25
T2	A7	T7	A17	S11	*A26
S2	*A8	S7	A18	T12	A27
T3	*A9	T8	A19	S12	A28
T4	A11	S8	A20	T13	A29
S4	*A12	T9	A21	S13	A30
T5	*A13	S9	A22	T14	A31
S5	*A14	T10	*A23	S14	A32
T6	A15	S10	*A24		
*These pins are available on NOVA 3 computer, slot 4 and above. S1 is connected to A6 which is -5V on backplane. S3 is connected to A10 which is +15V on backplane.					

FUNCTIONAL DESCRIPTION

Figure 1 shows the general organization of the I/O bus interface logic available on the GPIB module.

The basic configuration (MDB-4040) provides buffers for the device address, data status, and control lines. Data may be multiplexed to the computer through OR lines at buffer gates, instead of on the computer data bus. Address, data, status, and control lines present only a single TTL load at the bus.

The basic configuration also provides complete address decoding and Busy/Done logic; as well as interrupt request, mask, and priority logic.

The Register Option (MDB-4041) provides four 16-bit registers. The Input Data Register may be loaded with lower-order and higher-order bytes, with the I/O bus reading a 16-bit word from the register. The Output Data Register receives a 16-bit word from the I/O bus, and presents two 8-bit bytes to a byte multiplexer from which the device interface may select either byte.

Two additional registers included in the Register Option may be used simply as input and output registers, or as counters if the Data Channel Option (MDB-4042) is included on the module.

In data channel operations, one register (input) may be loaded from the bus with a starting address, and incremented with subsequent input data transfer cycles. The output register may be used as a word counter, loaded from the bus with the number of words to be transferred, and counted down with subsequent output data cycles. A zero count detector is provided to terminate block transfers.

The Data Channel Option provides all logic required to generate data channel requests, arbitrate priority, and buffer control signals to and from the bus.

Even though the GPIB is designed to serve a single device, all device address lines and control signals at the bus are buffered and accessible so that additional device controllers can be built, as space permits, on the wirewrap area of the board. Note that the Data General GPI/O board will not conveniently accommodate more than a single controller.

LOGIC UTILIZATION

The following paragraphs describe the manner in which logic shown in figure 1 operates, and explain how the logic may be utilized by the user.

BASIC CONFIGURATION

Logic included in the basic GPIB is shown in sheet 1 of the logic diagram included in this manual. All logic terms are asserted true in the high state (+5V) except when the term appears with a bar over it. The bar term is asserted true in the low state (zero volts).

Data Bus Drivers and Receivers

Data on the bidirectional data lines ($\overline{\text{DATA0}}$ - $\overline{\text{DATA15}}$) is received by inverting gates which provide outputs DATA0 through DATA15. Note that DATA0 is the *most*-significant bit. These outputs appear at wirewrap pins for use in user-built logic, and appear at inputs of the (optional) output data registers.

The I/O data bus is driven by open-collector bus drivers which are driven by inverters. Inputs to the inverters ($\overline{\text{IDATA0}}$ - $\overline{\text{IDATA15}}$) may originate at wirewrap pins in the user-built logic, and at outputs of the (optional) input data registers. Open-collector or tri-state drivers should be used to drive the $\overline{\text{IDATAn}}$ lines from user-built logic.

Table 2 lists the data lines and wirewrap pins at which each is terminated in pin rows X and Y.

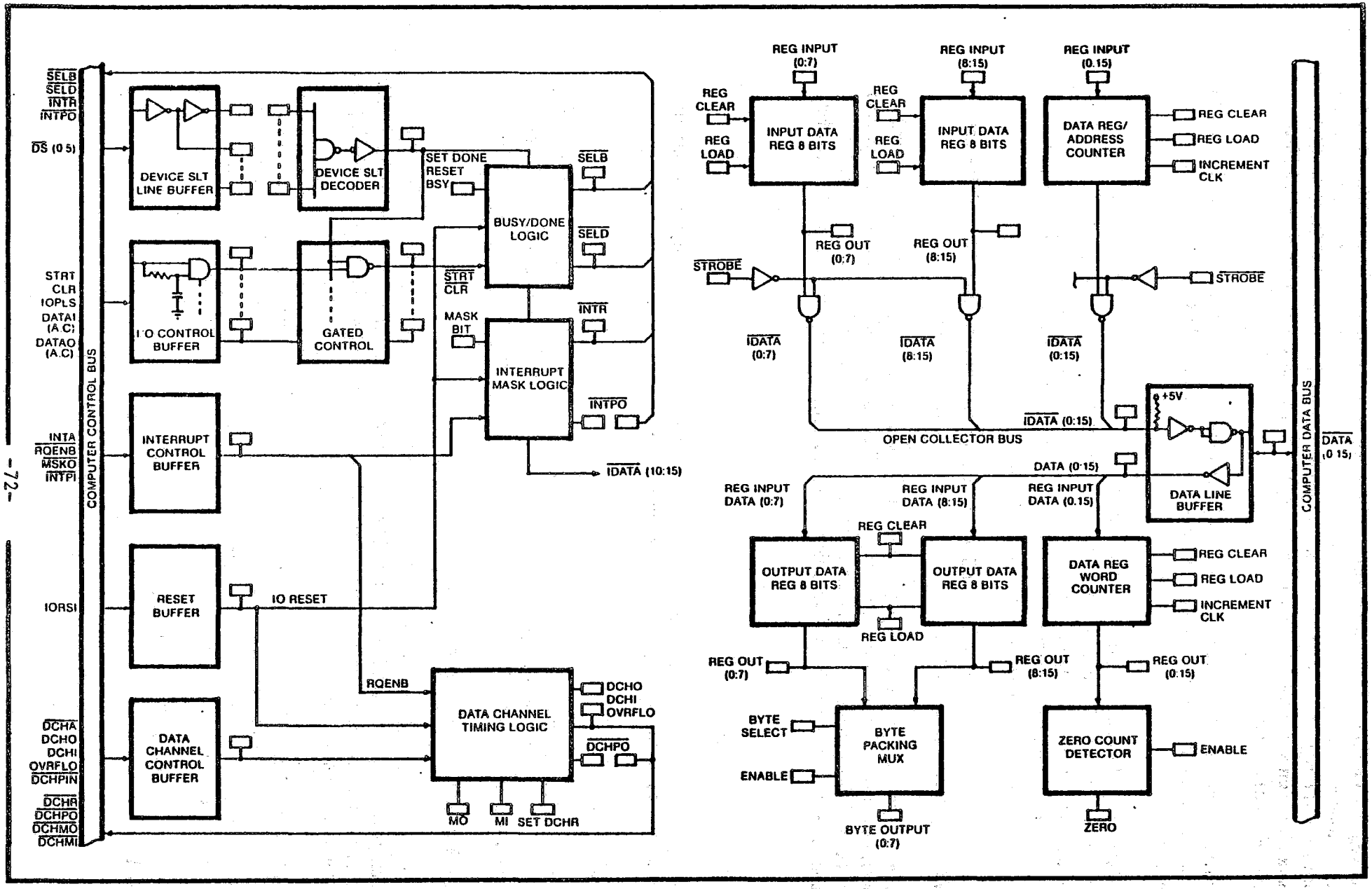


Figure 1. General-Purpose Interface Board, Block Diagram

Table 2. Input/Output Data Bus Pin Assignments

Computer Data Bus	Pin Number	GPIB Output Bus	Pin Number	GPIB Input Bus	Pin Number
$\overline{\text{DATA0}}$/B62	X59	DATA0	Y22	$\overline{\text{IDATA0}}$	X96
$\overline{\text{DATA1}}$/B65	X65	DATA1	X23	$\overline{\text{IDATA1}}$	Y103
$\overline{\text{DATA2}}$/B82	Y60	DATA2	X22	$\overline{\text{IDATA2}}$	Y97
$\overline{\text{DATA3}}$/B73	Y65	DATA3	Y23	$\overline{\text{IDATA3}}$	X102
$\overline{\text{DATA4}}$/B61	Y68	DATA4	Y32	$\overline{\text{IDATA4}}$	Y107
$\overline{\text{DATA5}}$/B57	X74	DATA5	X33	$\overline{\text{IDATA5}}$	X113
$\overline{\text{DATA6}}$/B95	X68	DATA6	X32	$\overline{\text{IDATA6}}$	X107
$\overline{\text{DATA7}}$/B55	Y74	DATA7	Y33	$\overline{\text{IDATA7}}$	Y113
$\overline{\text{DATA8}}$/B60	X78	DATA8	Y43	$\overline{\text{IDATA8}}$	X117
$\overline{\text{DATA9}}$/B63	Y85	DATA9	X44	$\overline{\text{IDATA9}}$	X123
$\overline{\text{DATA10}}$/B75	Y79	DATA10	X43	$\overline{\text{IDATA10}}$	Y118
$\overline{\text{DATA11}}$/B58	X84	DATA11	Y44	$\overline{\text{IDATA11}}$	Y123
$\overline{\text{DATA12}}$/B59	Y87	DATA12	X52	$\overline{\text{IDATA12}}$	Y126
$\overline{\text{DATA13}}$/B64	Y94	DATA13	Y54	$\overline{\text{IDATA13}}$	Y133
$\overline{\text{DATA14}}$/B56	X87	DATA14	Y53	$\overline{\text{IDATA14}}$	X126
$\overline{\text{DATA15}}$/B66	X93	DATA15	X53	$\overline{\text{IDATA15}}$	X132

Device Select Logic

The I/O bus device select lines ($\overline{\text{DS0}}$ - $\overline{\text{DS5}}$) are received by dual inverters which generate terms BDS0 through BDS5, and their complements (BDS0 is the *most*-significant line). These terms are connected to wirewrap pins for user access.

Device select terms DSLT0 through DSLT5 are inputs to the device address decoder and must be wired from wirewrap pins to appropriate BDSn or $\overline{\text{BDSn}}$ lines. These lines are available at wirewrap pins in rows X and Y as listed in table 3. This table also lists pins at which the device select bus lines are available.

Table 3. Device Select Line Pin Assignments

Computer Device Select Line	GPIB BDS Line	Pin Number	GPIB BDS Line	Pin Number	GPIB Device Select Line	Pin Number
$\overline{\text{DS0}}$/A72	BDS0	Y86	$\overline{\text{BDS0}}$	X86	DSLT0	X137
$\overline{\text{DS1}}$/A68	BDS1	X76	$\overline{\text{BDS1}}$	Y77	DSLT1	Y138
$\overline{\text{DS2}}$/A66	BDS2	X77	$\overline{\text{BDS2}}$	Y78	DSLT2	X138
$\overline{\text{DS3}}$/A46	BDS3	X58	$\overline{\text{BDS3}}$	Y59	DSLT3	Y137
$\overline{\text{DS4}}$/A62	BDS4	Y67	$\overline{\text{BDS4}}$	X67	DSLT4	Y139
$\overline{\text{DS5}}$/A64	BDS5	Y66	$\overline{\text{BDS5}}$	X66	DSLT5	X139

For example, the following connections are required to encode the address 62_8:

X137 to Y86 (DSLT0 to BDS0)
Y138 to X76 (DSLT1 to BDS1)
X138 to Y78 (DSLT2 to $\overline{\text{BDS2}}$)
Y137 to Y59 (DSLT3 to $\overline{\text{BDS3}}$)
Y139 to Y67 (DSLT4 to BDS4)
X139 to X66 (DSLT5 to $\overline{\text{BDS5}}$)

Be sure to connect lines bit-to-bit, in the correct order, because the DSLTn lines also form the device address returned to the computer to acknowledge an interrupt.

When the selected address appears on the lines, the term DEV SLT becomes true at wirewrap pin Y2. This term gates the contents of Busy and Done flip-flops to the computer, and gates a number of control signals received on the bus to internal logic.

Interface Control Signals

The basic GPIB/computer interface has 15 control lines controlled by the computer, and five lines controlled by the GPIB.

Computer generated control signals are received by buffer gates with outputs accessible at wirewrap pins. These terminations may be used in implementing additional controllers in the wirewrap area of the module. Device-oriented lines (gated by DEV SLT) control Busy/Done logic, load output registers, read input registers, or perform various functions in the user-built logic.

The five control signals originating at the GPIB are driven by open-collector gates.

Computer-generated control signals are listed and defined in table 4. Control signals originating at the GPIB are listed and defined in table 5.

Table 4. Computer Output Control Line Pin Assignments

Computer Control Line	Buffered Control Line	Pin	Device-Selected Control Line	Pin	Description
STRT/A52	BSTRT	X13	$\overline{\text{START}}$	Y13	Sets Busy, clears Done & INT REQ.
CLR/A50	BCLR	Y14	$\overline{\text{CLEAR}}$	X14	Clears Busy, Done, & INT REQ.
IOPLS/A74	BIOP	X15	$\overline{\text{IO PULSE}}$	Y15	User signal. May be used for user-defined control pulse.
DATIA/A44	BDIA	Y4	$\overline{\text{DATA IN A}}$	X3	User signal. May be connected to read Input Data Register or Address Counter Register.

Table 4. Computer Output Control Line Pin Assignments (cont'd)

Computer Control Line	Buffered Control Line	Pin	Device-Selected Control Line	Pin	Description
DATOA/A58	BDOA	Y12	$\overline{\text{DATA OUT A}}$	X12	User signal. May be connected to load Output Data Register or Word Counter Register.
DATIB/A42	BDIB	X2	$\overline{\text{DATA IN B}}$	Y3	Same as $\overline{\text{DATA IN A}}$.
DATOB/A56	BDOB	X6	$\overline{\text{DATA OUT B}}$	Y6	Same as $\overline{\text{DATA OUT A}}$.
DATIC/A54	BDIC	Y5	$\overline{\text{DATA IN C}}$	X5	Same as $\overline{\text{DATA IN A}}$.
DATOC/A48	BDOC	Y38	$\overline{\text{DATA OUT C}}$	X38	Same as $\overline{\text{DATA OUT A}}$.
INTA/A40	INT ACK	X28			Reads device address in GPIB when $\overline{\text{INTPIN}}$ and INT REQ are both true.
$\overline{\text{RQENB}}$/B41	RQENB	X104			Synchronizes INT REQ and DCH REQ to computer.
$\overline{\text{MSKO}}$/A38	MSKO	Y39			Masks out INT REQ when mask bit is on a data line.
IORST/A70	$\overline{\text{IO RESET A}}$	Y16			Resets all GPIB control logic.
$\overline{\text{INTPIN}}$/A96					Determines priority of GPIB interrupt request.
$\overline{\text{DCHPIN}}$/A94					Determines priority of GPIB data channel request.

Table 5. GPIB Output Control Line Pin Assignments

Computer Control Line	Pin	GPIB Source	Pin	Description
$\overline{\text{SELD}}$/A80	X27	Done Flip-flop	X8, X9	Asserted when Done is set and DEV SLT is true.
$\overline{\text{SELB}}$/A82	Y17	Busy Flip-flop	X1	Asserted when Busy is set and DEV SLT is true.
$\overline{\text{INTR}}$/B29	X48	INT REQ	X16	Asserted when INT REQ flip-flop is set.
$\overline{\text{INTPOUT}}$ /A95	X62	INTP IN and $\overline{\text{INT REQ}}$	--	Connect X62 to X17. This allows $\overline{\text{INTP IN}}$ to propagate to $\overline{\text{INTP OUT}}$ line when GPIB has not set INT REQ.
$\overline{\text{DCHPOUT}}$ /A93	Y58	DCHP IN and $\overline{\text{DCH REQ}}$	—	Connect Y58 to Y27. This allows $\overline{\text{DCHP IN}}$ to propagate to $\overline{\text{DCHP OUT}}$ line when GPIB has not set DCH REQ.

Busy/Done and Interrupt Logic

The basic GPIB provides the standard mechanization of Busy/Done and interrupt logic generally used to control device controllers on the computer I/O bus.

Busy is set by the computer-generated STRT pulse and usually indicates the start of the controller operation. The GPIB remains busy until user-built logic completes its operation and sets Done, clearing the Busy flip-flop. Busy is also cleared by a computer-generated CLR or IORST pulse.

The Done flip-flop may be set by user-built logic in either of two ways:

a. By a negative-going pulse at wirewrap pin X7 ($\overline{\text{SET DNE}}$), which direct-sets the Done flip-flop; or

b. By a clock signal at pin Y7 (CLK DNE) with pin Y8 (CLK DNE EN) held high. Note that in this method Done can be set only while Busy is high.

The computer may monitor Done on the $\overline{\text{SELD}}$ line. If an interrupt is enabled, the logic will set an interrupt request ($\overline{\text{INTR}}$) on the next rise of RQENB after Done is set.

The INT REQ flip-flop is usually set after Done is set, but there are two alternate methods, as follows:

a. Disconnect Done (Pin X8) from pin X9 and control INT REQ with another signal at X9. Ground X9 to permanently disable interrupt requests.

b. Assign a mask bit in software to appear on a data line (DATA0 - DATA15), and connect the bit to pin X10 (MSK BIT). The computer can then disable interrupt requests using a Mask Out ($\overline{\text{MSKO}}$) instruction to clock the assigned bit into the Mask flip-flop.

The INT REQ flip-flop may be reset by STRT, CLR, or IORST. The Mask flip-flop is cleared to the "enable" state by IROST.

The computer responds to $\overline{\text{INTR}}$ with an Interrupt Acknowledge instruction which asserts INTA (along with DS0 through DS5) at the interface. The GPIB responds to INTA by returning the device address (provided by the user-wired DSn lines to the device select decoder) on the data lines if (1) INT REQ is set, and (2) the GPIB has priority ($\overline{\text{INTP IN}}$ is asserted).

If INT REQ is not set, $\overline{\text{INTP IN}}$ is propagated to wirewrap pin X17. If another controller is built on the GPIB, it will have the next-lower priority. Therefore, connect X17 to the $\overline{\text{INTP IN}}$ terminal of that logic. If the controller having next-lower priority is not on the GPIB, connect X17 to X62 ($\overline{\text{INTP OUT}}$) to be transmitted to the next controller on the serial priority chain.

If the Data Channel option is **not** used, connect pin Y27 to pin Y58 to propagate DCHP IN to the DCHP OUT line. If the Data Channel option is to be used, refer to *Data Channel Option.*

I/O RESET

IORST at the interface is inverted to form $\overline{\text{IO RESET A}}$, which resets all control flip-flops in the basic GPIB logic. $\overline{\text{IO RESET A}}$ also produces IO RESET, $\overline{\text{IO RESET B}}$, and $\overline{\text{IO RESET C}}$, which may be used in user-built logic.

A reset signal generated in user-built logic may be connected to wirewrap pin X21 and ORed with $\overline{\text{IO RESET A}}$ to also generate IO RESET, $\overline{\text{IO RESET B}}$, and $\overline{\text{IO RESET C}}$.

REGISTER OPTION

Logic for the Register Option is shown on sheet 2 of the logic diagram included in this manual. This option furnishes four 16-bit registers, as follows:

- **Input Data Register.** This is the normal register for receiving data from user-built logic for transfer to the computer.

- **Input Data/Address Counter Register.** This may be used either as another input data register, or as an address counter with the Data Channel Option.

- **Output Data Register.** This is the normal register for receiving data from the computer for transfer to user-built logic on the GPIB.

- **Output Data/Word Counter Register.** This may be used as another output data register, or as a word counter with the Data Channel Option.

Input Data Register

The 16 inputs to this register are connected from wirewrap posts. Each half of the register may be loaded separately, with LIDRL (pin Y130) loading the eight least-significant bits, and LIDRH (pin X110) loading the most-significant eight bits. To load the register with a 16-bit word, connect pins X110 and Y130 together.

To pack byte-oriented data, connect the byte data both to high-order and low-order halves of the register, and load the halves alternately.

Assert RIDRL (pin X129) to reset the low-order half of the register, or assert RIDRH (pin Y110) to reset the high-order half. Tie those pins together to reset the entire register.

Register outputs appear at wirewrap posts for access to user-built logic, and to gates that drive the GPIB input data bus. When $\overline{\text{RDIDR}}$ (pin X120) is asserted, the contents of the register are strobed to the bus. $\overline{\text{RDIDR}}$ is normally connected to $\overline{\text{DATA IN A}}$ (pin X3), $\overline{\text{DATA IN B}}$ (pin Y3), or $\overline{\text{DATA IN C}}$ (pin X5), depending on software format.

Table 6 lists wirewrap pin assignments for signals related to the Input Data Register.

Table 6. Input Data Register Pin Assignments

Read Control $\overline{\text{RDIDR}}$ (X120)					
High Order Byte			Low Order Byte		
Load Control $\overline{\text{LIDRH}}$ (X110) Reset Control $\overline{\text{RIDRH}}$ (Y110)			Load Control $\overline{\text{LIDRL}}$ (Y130) Reset Control $\overline{\text{RIDRL}}$ (X129)		
Data Input	Data Output	GPIB Input Data Bus	Data Input	Data Output	GPIB Input Data Bus
Y99	X97	$\overline{\text{IDATA0}}$ (MSB)	Y120	X118	$\overline{\text{IDATA8}}$
X100	Y102	$\overline{\text{IDATA1}}$	Y121	X122	$\overline{\text{IDATA9}}$
X98	Y98	$\overline{\text{IDATA2}}$	X119	Y119	$\overline{\text{IDATA10}}$
Y101	X101	$\overline{\text{IDATA3}}$	X121	Y122	$\overline{\text{IDATA11}}$
X109	Y108	$\overline{\text{IDATA4}}$	Y129	Y127	$\overline{\text{IDATA12}}$
Y111	X112	$\overline{\text{IDATA5}}$	X130	Y132	$\overline{\text{IDATA13}}$
Y109	X108	$\overline{\text{IDATA6}}$	X128	X127	$\overline{\text{IDATA14}}$
X111	Y112	$\overline{\text{IDATA7}}$	Y131	X131	$\overline{\text{IDATA15}}$ (LSB)

Output Data Register

The 16 inputs to this register originate at the I/O data bus. Register outputs are connected to wirewrap pins and to inputs of a byte-packing multiplexer.

The register is loaded by asserting $\overline{\text{LODR}}$ (pin X36) and is reset by asserting $\overline{\text{RODR}}$ (pin Y36). Typically, $\overline{\text{LODR}}$ is connected to $\overline{\text{DATA OUT A}}$ (pin X12), $\overline{\text{DATA OUT B}}$ (pin Y6), or $\overline{\text{DATA OUT C}}$ (pin X38), depending on software format.

The byte multiplexer selects either the high-order byte, or the low-order byte, at register outputs and presents the selected byte to wirewrap pins for use by user-built logic. The

byte is selected by BSEL (pin X57). The high-order byte is selected when BSEL is low, and the low-order byte is selected when BSEL is high.

Multiplexer outputs may be disabled by connecting STR (pin Y117) low.

Table 7 lists wirewrap pin assignments for signals related to the Output Data Register.

Table 7. Output Data Register Pin Assignments

Load Control $\overline{\text{LODR}}$ (X36) Reset Control $\overline{\text{RODR}}$ (Y36)				Multiplexer Control BSEL (X57) Low = DATA0 - DATA7 High = DATA8 - DATA15 Multiplexer Disable STR (Y117)
Data In	**Data Out**	**Data In**	**Data Out**	**Multiplexer Out**
DATA0 (MSB)	X19	DATA8	Y41	X42
DATA1	Y25	DATA9	X46	Y42
DATA2	Y20	DATA10	X41	Y45
DATA3	X24	DATA11	Y46	X45
DATA4	Y31	DATA12	X50	Y52
DATA5	X34	DATA13	Y56	X51
DATA6	X31	DATA14	Y51	X54
DATA7	Y34	DATA15 (LSB)	X55	Y55

Input Data/Address Register

This 16-bit register may be utilized as a second input register, operating in the same manner as the Input Data Register described previously. The low-order byte is loaded by asserting $\overline{\text{LACL}}$ (pin Y91), and is reset by asserting $\overline{\text{RACL}}$ (pin X90). The high-order byte is loaded by asserting $\overline{\text{LACH}}$ (pin X71), and reset by asserting $\overline{\text{RACH}}$ (pin Y71). The register is read by asserting $\overline{\text{RDAC}}$ (pin X116).

This register may also be connected as a binary ripple counter, incremented by the negative-going transition to clock input CAC (pin Y89).

Used with the Data Channel Option, the register is used as a memory address counter. The user must connect the data inputs to the register, from the data bus. The register is loaded with a starting address on the data bus (DATA0 - DATA15) by an asserted command $\overline{\text{DATA OUT A}}$, $\overline{\text{DATA OUT B}}$, or $\overline{\text{DATA OUT C}}$, depending on software format. The register is incremented by CAC, provided by the inverse of the term $\overline{\text{DCHSEL•DCHA}}$ (pin Y124) from user-built logic, updating the address for each data channel cycle.

The address counter register is read to the bus by $\overline{\text{RDAC}}$ (connected to $\overline{\text{DCHSEL•DCHA}}$ pin Y124) during each data channel cycle.

Table 8 lists pin assignments for signals related to this register.

Table 8. Input Data/Address Register Pin Assignments

Read Control $\overline{RDAC}$ (X116)			Clock Input CAC (Y89)		
High-order Byte			Low-order Byte		
Load Control $\overline{LACH}$ (X71) Reset Control $\overline{RACH}$ (Y71)			Load Control $\overline{LACL}$ (Y91) Reset Control $\overline{RACL}$ (X90)		
Data Input	Data Output	GPIB Input Data Bus	Data Input	Data Output	GPIB Input Data Bus
Y62	X60	$\overline{IDATA0}$ (MSB)	Y81	X79	$\overline{IDATA8}$
Y63	X64	$\overline{IDATA1}$	X82	Y84	$\overline{IDATA9}$
X61	Y61	$\overline{IDATA2}$	X80	Y80	$\overline{IDATA10}$
X63	Y64	$\overline{IDATA3}$	Y83	X83	$\overline{IDATA11}$
X70	Y69	$\overline{IDATA4}$	Y90	Y88	$\overline{IDATA12}$
Y72	X73	$\overline{IDATA5}$	X91	Y93	$\overline{IDATA13}$
Y70	X69	$\overline{IDATA6}$	X89	X88	$\overline{IDATA14}$
X72	Y73	$\overline{IDATA7}$	Y92	X92	$\overline{IDATA15}$ (LSB)

Output Data/Word Counter Register

This 16-bit register may be utilized as a second output register, operating in the same manner as the Output Data Register described previously. The register is loaded from the data bus (DATA0 - DATA15) by asserting $\overline{LWC}$ (pin X37), and is reset by asserting $\overline{RWC}$ (pin Y37). Register outputs appear at wirewrap pins for use by user-built logic and (with the Data Channel Option) at a word count zero detector.

The register may be operated as a binary ripple counter, incrementing the counter with the negative transition at the clock input CWC (pin Y50).

Used as a word counter in the Data Channel Option, the register is loaded from the bus with the twos-complement of the number of words to be transferred in the block. Subsequent CWC pulses increment the counter with each data transfer, until the count reaches zero to flag the end of the block transfer. In this application the CWC pin may be connected in user-built logic to the term DCHSEL•DCHA (term $\overline{DCHSEL•DCHA}$ is available at wirewrap pin Y124).

The word counter detector output WCZERO (pin X29) is asserted when the word count reaches zero. The term $\overline{WCZERO}$ is also available at pin Y29.

The maximum word count may be preset by strapping disable signals $\overline{WCDISA}$, $\overline{WCDISB}$, $\overline{WCDISC}$, and $\overline{WCDISD}$ as follows:

Max. Count	$\overline{WCDISA}$ (Y18)	$\overline{WCDISB}$ (X26)	$\overline{WCDISC}$ (X39)	$\overline{WCDISD}$ (Y48)
65,536	open	open	open	open
4,096	ground	open	open	open
256	ground	ground	open	open
16	ground	ground	ground	open

Note that because WCZERO is generated by decoding a ripple counter, that term may have spikes at clock intervals. A suggested method of ending a block transfer is to connect WCZERO to CLK DNE EN (pin Y8), and to connect CLK DNE (pin Y7) to either DCH $\overline{\text{SEL•DCHO}}$ (pin Y95) or $\overline{\text{DCH SEL•DCHI}}$ (pin X95).

Table 9 lists pin assignments for signals related to this register.

Table 9. Output Data/Word Counter Register Pin Assignments

Load Control $\overline{\text{LWC}}$ (X37) Reset Control $\overline{\text{RWC}}$ (Y37) Clock Input CWC (Y50)			
Data Input	**Data Output**	**Data Input**	**Data Output**
DATA0 (MSB)	X18	DATA8	Y40
DATA1	Y26	DATA9	X47
DATA2	Y19	DATA10	X40
DATA3	X25	DATA11	Y47
DATA4	Y30	DATA12	Y49
DATA5	X35	DATA13	Y57
DATA6	X30	DATA14	X49
DATA7	Y35	DATA15 (LSB)	X56

DATA CHANNEL OPTION

Logic for the Data Channel Option is shown on sheet 2 of the logic diagram included in this manual. Logic for this function permits the GPIB to control block transfer of data on the computer's data channel. Logic includes interface buffers, data channel control, request and acknowledge logic, and address and word counters with word count zero-detection logic (refer to *Register Option*).

The five data channel control lines originating at the computer (table 10) are buffered, with buffer outputs available at wirewrap pins for connection to a second user-built device controller on the GPIB.

Table 11 lists and defines the four data channel control lines originating at the GPIB.

Table 10. Computer-Generated Data Channel Control Lines, Pin Assignments

Computer Control Line	Buffered Control Line	Pin	DCH-Selected Control Line	Pin	Description
$\overline{\text{DCHA}}$/A60	DCHA	X85	$\overline{\text{DCHSEL}}$• $\overline{\text{DCHA}}$	Y124	Returns memory address to computer during data channel cycle. May be used to read Address Counter by connecting to $\overline{\text{RDAC}}$ (X116).

Table 10. Computer-Generated Data Channel Control Lines, Pin Assignments (cont'd)

Computer Control Line	Buffered Control Line	Pin	DCH-Selected Control Line	Pin	Description
DCHO/B33	BDCHO	X94	$\overline{\text{DCHSEL}}$• $\overline{\text{DCHO}}$	Y95	Use to access computer output data on data channel. To load Output Data Register, connect to $\overline{\text{LODR}}$ (X36).
DCHI/B37	BDCHI	Y96	$\overline{\text{DCHSEL}}$• $\overline{\text{DCHI}}$	X95	Puts memory data on bus to be read by computer during data channel input cycle. To read Input Data Register, connect to $\overline{\text{RDIR}}$ (X120).
OVRFLO/B39	BOVRFLO	X103	$\overline{\text{DCHSEL}}$• $\overline{\text{OVERFLOW}}$	Y104	Generated by certain model computers to indicate data overflow during data channel increment or add-to-memory cycle.
$\overline{\text{DCHP IN}}$/A94					Determines priority of GPIB data channel request.

Table 11. GPIB-Generated Data Channel Control Lines, Pin Assignments

Computer Control Line	GPIB Source	Pin	Description
$\overline{\text{DCHR}}$ B35, X99	DCH REQ flip-flop	X124	Requests transfer to or from memory on data channel.
$\overline{\text{DCHM0}}$ B17, Y140	M0	X140	Asserted during data channel cycle to allow computer to determine the type of memory operation to be performed. User connects M0 and M1 as follows:
$\overline{\text{DCHMI}}$ B21, X136	MI	Y136	**M0 M1 Data Channel Operation** 0 0 Data Out (read memory) 0 1 Increment Memory 1 0 Data In (write to memory) 1 1 Add to Memory Note that increment and add-to-memory are valid operations only on certain models of computer.
$\overline{\text{DCHP OUT}}$ A93, Y58	DCHP IN and $\overline{\text{DCH REQ}}$		Connect Y27 to Y58. This allows GPIB to propagate $\overline{\text{DCHP IN}}$ to $\overline{\text{DCHP OUT}}$ when GPIB has not set DCH REQ.

The user initiates a data channel request by setting the DCH SYNC (X134) flip-flop in either of two ways, as follows:

a. Direct-set DCH SYNC with a negative-going pulse at pin Y135 ($\overline{\text{SET SYNC}}$), or

b. Set DCH SYNC with a clock at pin X135 (CLK DSYNC), with pins X133 and Y134 (DSYNC EN1 and DSYNC EN2, respectively) both held high.

For repetitive data channel requests, $\overline{\text{SET SYNC}}$ may be held low while the data channel requests are asserted. $\overline{\text{SET SYNC}}$ must be raised before, or during, the last DCHA signal required for the repetitive data channel operations.

DCH SYNC is normally reset with the DCH-selected DCHA pulse or IORST. DCH REQ (pin X124) is set by the positive-going transition of the next RQENB pulse after DCH SYNC is set. This asserts $\overline{\text{DCHR}}$ at the computer bus. DCH REQ is reset by the next RQENB pulse after DCH SYNC is reset, or by IORST.

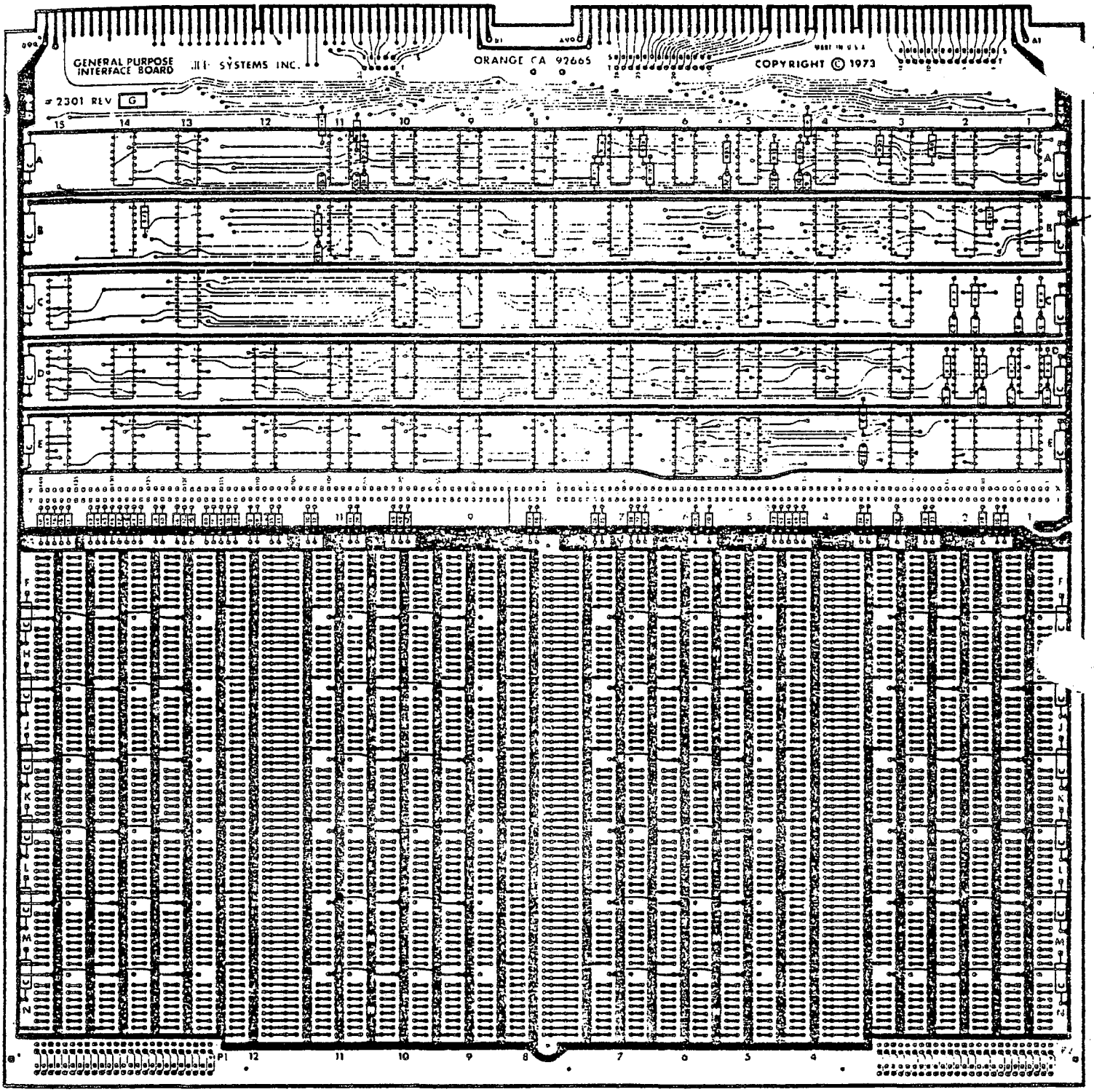
$\overline{\text{DCHP IN}}$ is propagated to pin Y27 if $\overline{\text{DCH REQ}}$ is not set. The user may connect pin Y27 to user-built logic, or connect it to pin Y58 ($\overline{\text{DCHP OUT}}$) for transmission to the next controller on the I/O bus.

MAINTENANCE

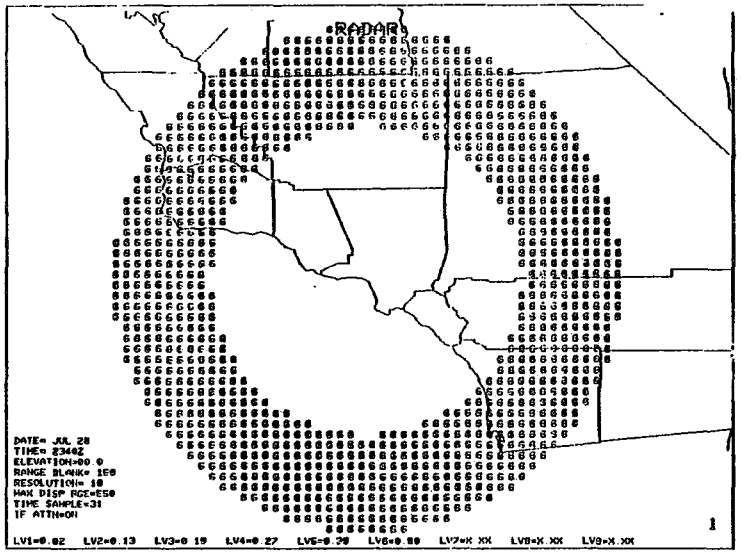
Refer to the logic and assembly diagrams in this manual, and to appropriate Data General manuals for the NOVA computer.

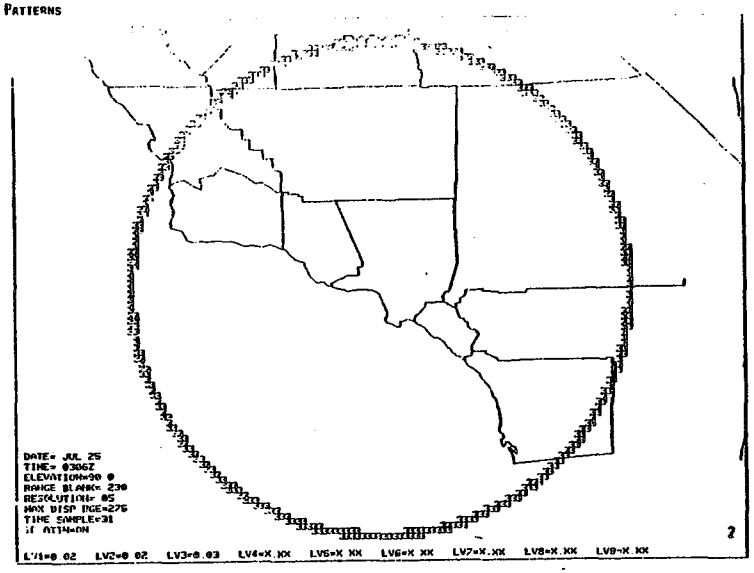
Repair the module using appropriate skills, techniques, and materials. Contact MDB Systems' Customer Service Department for assistance, if necessary.

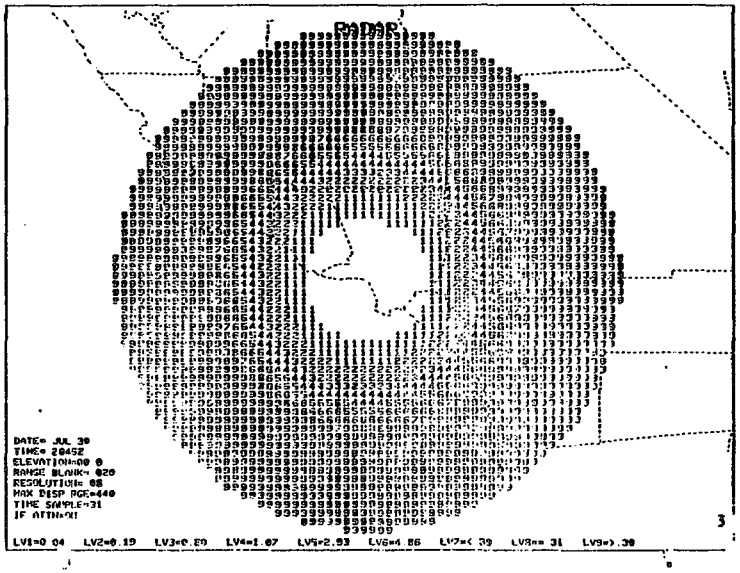
Do not return the user-configured module to MDB Systems without prior permission of MDB Systems.

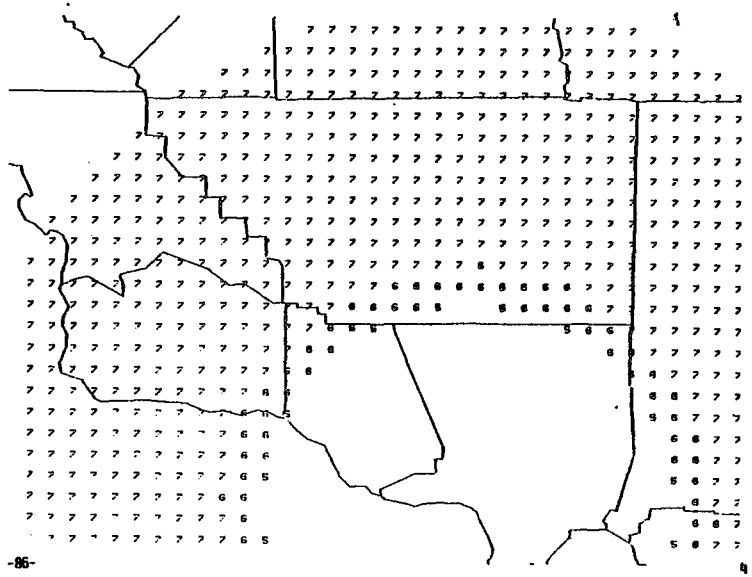

GENERAL PURPOSE
INTERFACE BOARD
JEF SYSTEMS INC.
ORANGE CA 92665
COPYRIGHT © 1973
2301 REV G

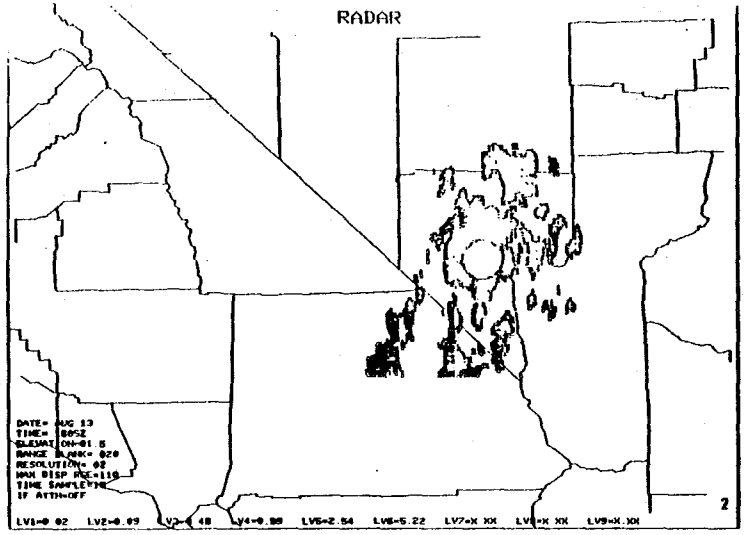
AFOS - RADAR OUTPUT EXAMPLES

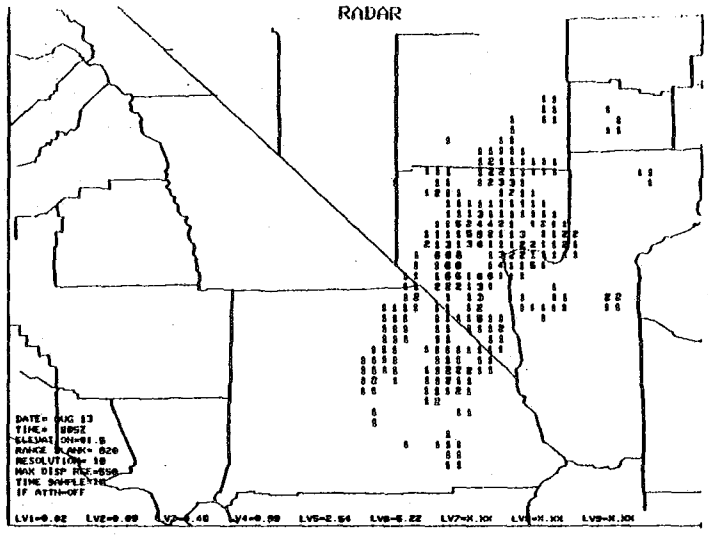

RADAR
DATE= JUL 28
TIME= 2340Z
ELEVATION=00 0
RANGE BLANK= 1E0
RESOLUTION= 10
MAX DISP RGE=E50
TIME SAMPLE=31
IF ATTN=ON
LV1=0.02 LV2=0.13 LV3=0.19 LV4=0.27 LV5=0.38 LV6=0.80 LV7=X.XX LV8=X.XX LV9=X.XX
1

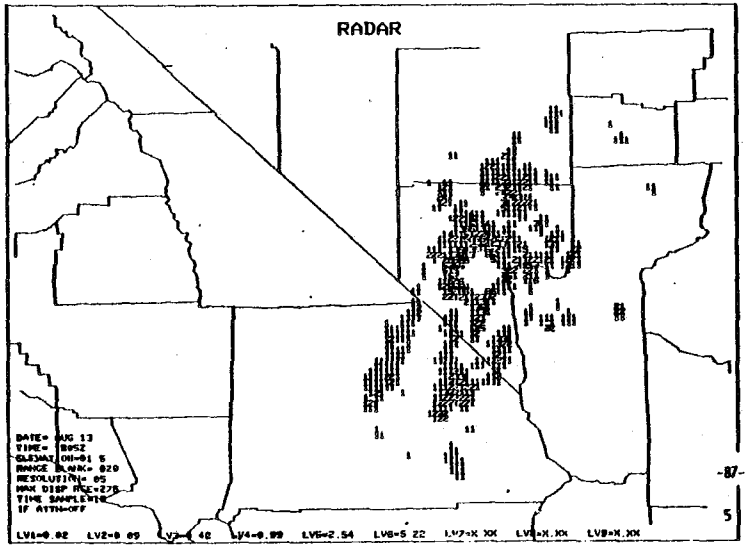

DATE= JUL 25
TIME= 0306Z
ELEVATION=90 0
RANGE BLANK= 230
RESOLUTION= 05
MAX DISP RGE=275
TIME SAMPLE=31
IF ATTN=ON
LV1=0.02 LV2=0.02 LV3=0.03 LV4=X.XX LV5=X.XX LV6=X.XX LV7=X.XX LV8=X.XX LV9=X.XX
2

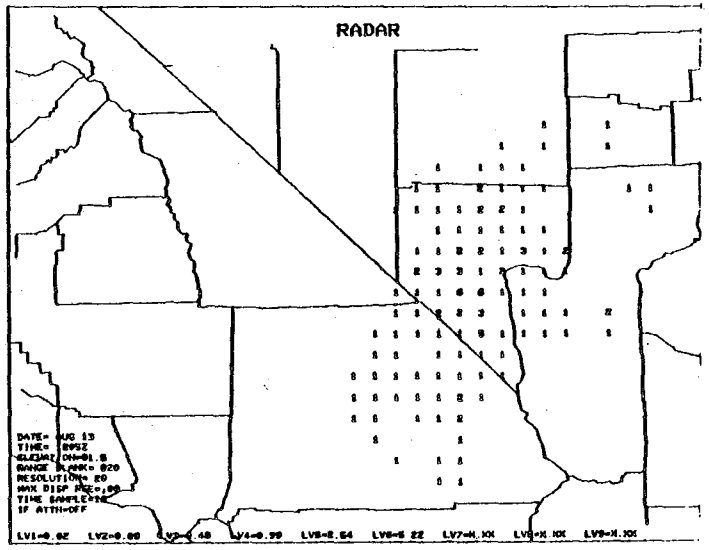

RADAR
DATE= JUL 30
TIME= 2045Z
ELEVATION=00 0
RANGE BLANK= 020
RESOLUTION= 08
MAX DISP RGE=440
TIME SAMPLE=31
IF ATTN=ON
LV1=0.04 LV2=0.19 LV3=0.29 LV4=1.07 LV5=2.93 LV6=4.86 LV7=< 30 LV8=> 31 LV9=> 30
3

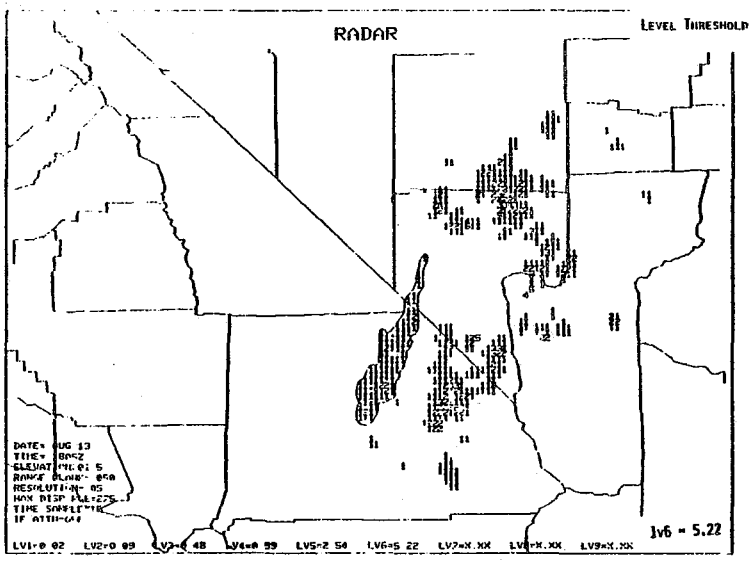

4


RADAR
2

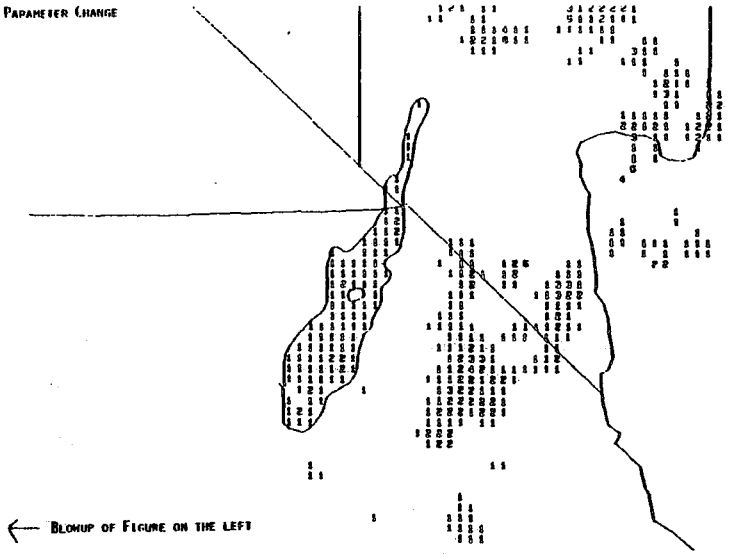

RADAR


RADAR
-87-
5

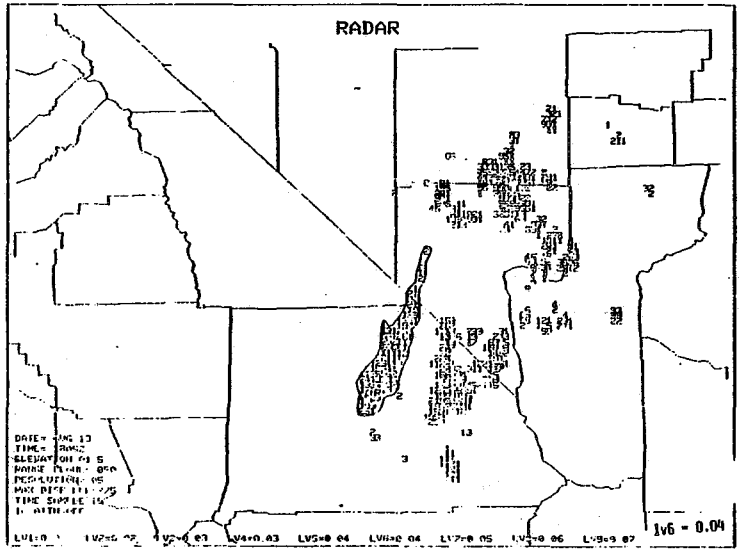

RADAR

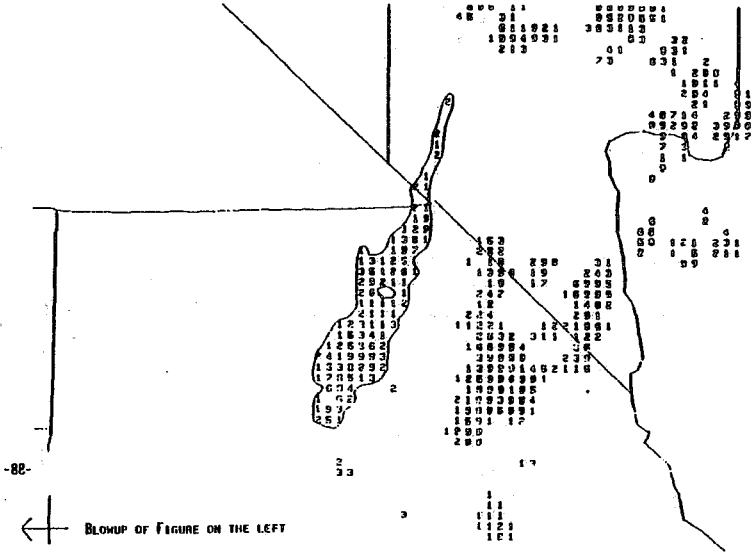


LEVEL THRESHOLD PARAMETER CHANGE

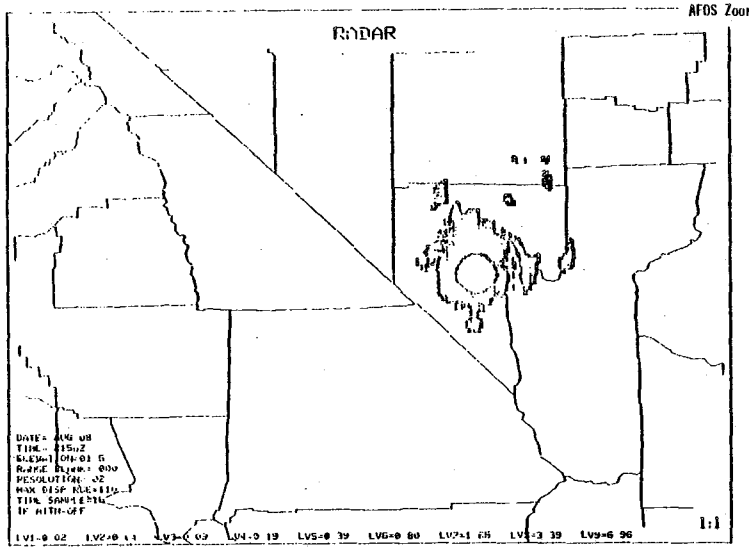


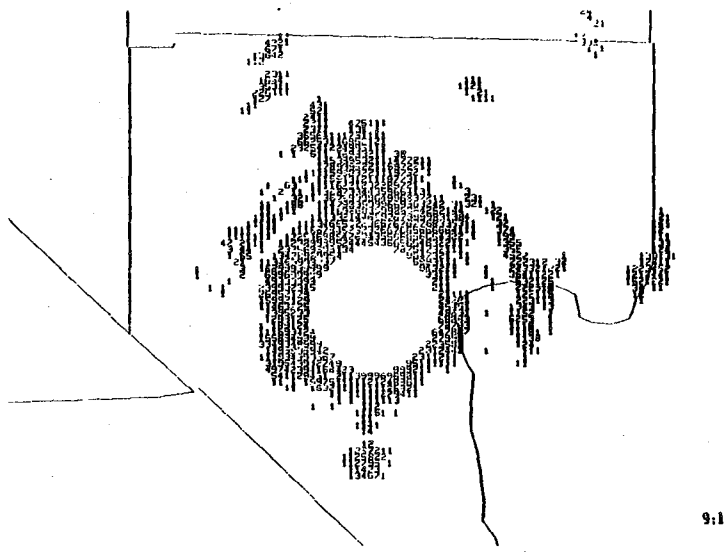
← BLOWUP OF FIGURE ON THE LEFT

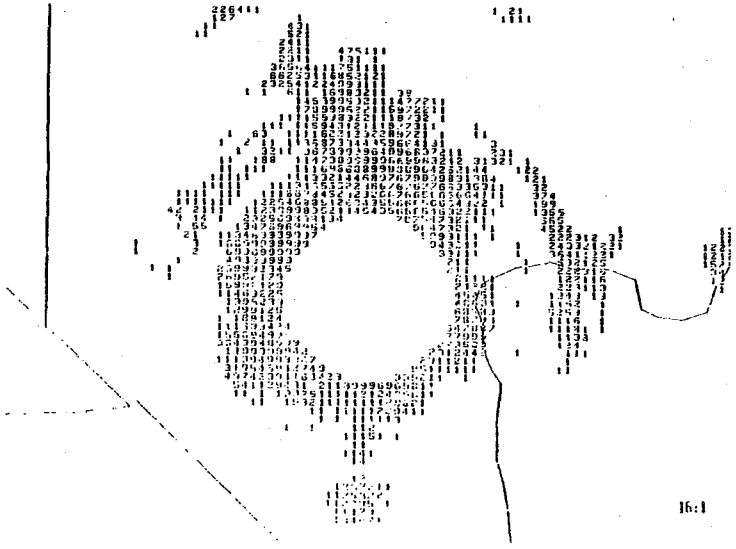


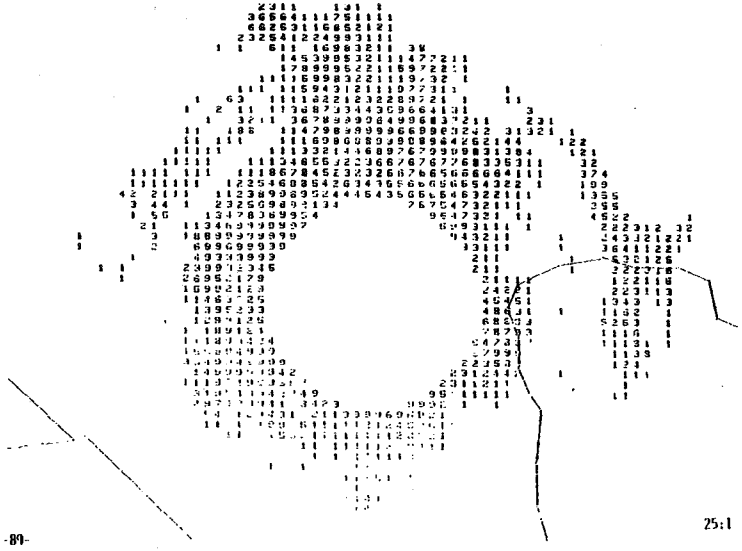


← BLOWUP OF FIGURE ON THE LEFT

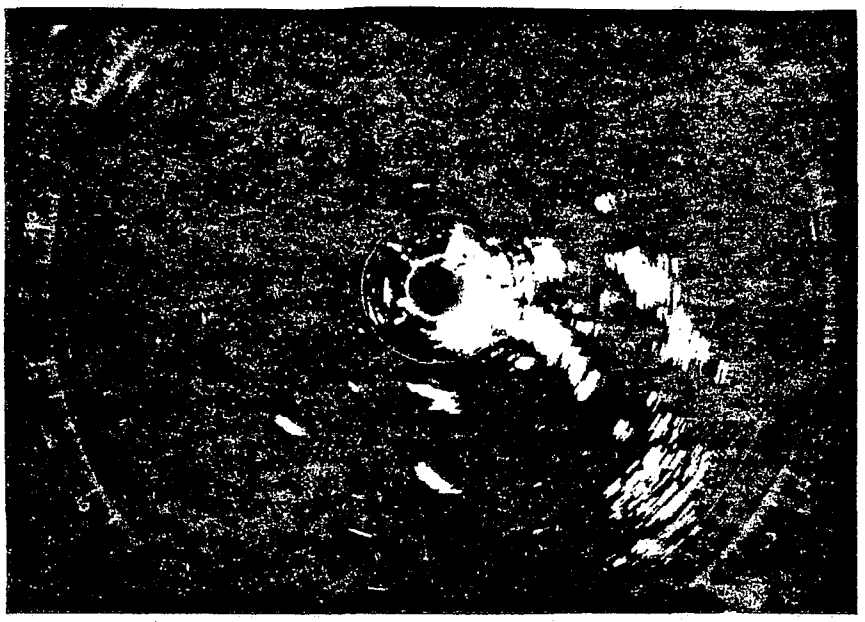

RADAR
1:1


9:1


16:1


25:1

LAX Ground Clutter



SW-5C radar screen for August 3, 1979, 3764, 250 am, range

NOAA Technical Memoranda NWSWR: (Continued)

92 Smoke Management in the Willamette Valley. Earl M. Bates, May 1974. (COM-74-11277/AS)
93 An Operational Evaluation of 500-mb Type Regression Equations. Alexander E. MacDonald, June 1974. (COM-74-11407/AS)
94 Conditional Probability of Visibility Less than One-Half Mile in Radiation Fog at Fresno, California. John D. Thomas, August 1974. (COM-74-11355/AS)
96 Map Type Precipitation Probabilities for the Western Region. Glenn E. Rasch and Alexander E. MacDonald, February 1975. (COM-75-10428/AS)
97 Eastern Pacific Cut-off Low of April 21-28, 1974. William J. Alder and George R. Miller, January 1976. (PB-250-711/AS)
98 Study on a Significant Precipitation Episode in Western United States. Ira S. Brenner, April 1976. (COM-75-10719/AS)
99 A Study of Flash Flood Susceptibility--A Basin in Southern Arizona. Gerald Williams, August 1975. (COM-75-11360/AS)
102 A Set of Rules for Forecasting Temperatures in Napa and Sonoma Counties. Wesley L. Tuff, October 1975. (PB-246-902/AS)
103 Application of the National Weather Service Flash-Flood Program in the Western Region. Gerald Williams, January 1976. (PB-253-055/AS)
104 Objective Aids for Forecasting Minimum Temperatures at Reno, Nevada, During the Summer Months. Christopher D. Hill, January 1976. (PB-252-866/AS)
105 Forecasting the Mono Wind. Charles P. Ruscha, Jr., February 1976. (PB-254-650)
106 Use of MOS Forecast Parameters in Temperature Forecasting. John C. Plankinton, Jr., March 1976. (PB-254-649)
107 Map Types as Aids in Using MOS PoPs in Western United States. Ira S. Brenner, August 1976. (PB-259-594)
108 Other Kinds of Wind Shear. Christopher D. Hill, August 1976. (PB-260-437/AS)
109 Forecasting North Winds in the Upper Sacramento Valley and Adjoining Forests. Christopher E. Fontana, Sept. 1976. (PB-273-677/AS)
110 Cool Inflow as a Weakening Influence on Eastern Pacific Tropical Cyclones. William J. Denney, November 1976. (PB-264-655/AS)
112 The MAN/MOS Program. Alexander E. MacDonald, February 1977. (PB-265-941/AS)
113 Winter Season Minimum Temperature Formula for Bakersfield, California, Using Multiple Regression. Michael J. Oard, February 1977. (PB-276-694/AS)
114 Tropical Cyclone Kathleen. James R. Fors, February 1977. (PB-273-676/AS)
116 A Study of Wind Gusts on Lake Mead. Bradley Colman, April 1977. (PB-268-847)
117 The Relative Frequency of Cumulonimbus Clouds at the Nevada Test Site as a Function of K-value. R. F. Quiring, April 1977. (PB-272-831)
118 Moisture Distribution Modification by Upward Vertical Motion. Ira S. Brenner, April 1977. (PB-268-740)
119 Relative Frequency of Occurrence of Warm Season Echo Activity as a Function of Stability Indices Computed from the Yucca Flat, Nevada, Rawinsonde. Darryl Randerson, June 1977. (PB-271-290/AS)
121 Climatological Prediction of Cumulonimbus Clouds in the Vicinity of the Yucca Flat Weather Station. R. F. Quiring, June 1977. (PB-271-704/AS)
122 A Method for Transforming Temperature Distribution to Normality. Morris S. Webb, Jr., June 1977. (PB-271-742/AS)
124 Statistical Guidance for Prediction of Eastern North Pacific Tropical Cyclone Motion - Part I. Charles J. Neumann and Preston W. Leftwich, August 1977. (PB-272-661)
125 Statistical Guidance on the Prediction of Eastern North Pacific Tropical Cyclone Motion - Part II. Preston W. Leftwich and Charles J. Neumann, August 1977. (PB-275-155/AS)
127 Development of a Probability Equation for Winter-Type Precipitation Patterns in Great Falls, Montana. Kenneth S. Mielke, February 1978. (PB-281-237/AS)
128 Hand Calculator Program to Compute Parcel Thermal Dynamics. Dan Gudgel, April 1978. (PB-283-080/AS)
129 Fire Whirls. David W. Coons, May 1978. (PB-285-866/AS)
130 Flash-Flood Procedure. Ralph C. Hatch and Gerald Williams, May 1978. (PB-286-014/AS)
131 Automated Fire-Weather Forecasts. Mark A. Mollner and David E. Olsen, September 1978. (PB-289-916/AS)

132 Estimates of the Effects of Terrain Blocking on the Los Angeles WSR-74C Weather Radar. R. G. Pappas, R. Y. Lee, and B. W. Finke, October 1978. (PB289767/AS)
133 Spectral Techniques in Ocean Wave Forecasting. John A. Jannuzzi, October 1978. (PB291517/AS)
134 Solar Radiation. John A. Jannuzzi, November 1978. (PB291195/AS)
135 Application of a Spectrum Analyzer in Forecasting Ocean Swell in Southern California Coastal Waters. Lawrence P. Kierulff, January 1979. (PB292716/AS)
136 Basic Hydrologic Principles. Thomas L. Dietrich, January 1979. (PB292247/AS)
137 LFM 24-Hour Prediction of Eastern Pacific Cyclones Refined by Satellite Images. John R. Zimmerman and Charles P. Ruscha, Jr., January 1979. (PB294324/AS)
138 A Simple Analysis/Diagnosis System for Real Time Evaluation of Vertical Motion. Scott Hoflick and James R. Fors, February 1979. (PB294218/AS)
139 Aids for Forecasting Minimum Temperature in the Wenatchee Frost District. Robert S. Robinson, April 1979.
140 Influence of Cloudiness on Summertime Temperatures in the Eastern Washington Fire Weather District. James Holcomb, April 1979.
141 Comparison of LFM and MFM Precipitation Guidance for Nevada During Season. Christopher Hill, April 1979.
142 The Usefulness of Data from Mountaintop Fire Lookout Stations in Determining Atmospheric Stability. Jonathan W. Corey April 1979.
143 The Depth of the Marine Layer at San Diego as Related to Subsequent Cool Season Precipitation Episodes in Arizona. Ira S. Brenner, May 1979.
144 Arizona Cool Season Climatological Surface Wind and Pressure Gradient Study. Ira S. Brenner, May 1979.
145 On the Use of Solar Radiation and Temperature Models to Estimate the Snap Bean Maturity Date in the Willamette Valley. Earl M. Bates, August 1979
146 The Barr Experiment. Morris S. Webb, October 1979.
147 Occurrence and Distribution of Flash Floods in the Western Region. Thomas L. Dietrich, December 1979.
148 A Real-Time Radar Interface for AFOS. Mark Mathewson, January 1980.

NOAA SCIENTIFIC AND TECHNICAL PUBLICATIONS

NOAA, the *National Oceanic and Atmospheric Administration*, was established as part of the Department of Commerce on October 3, 1970. The mission responsibilities of NOAA are to monitor and predict the state of the solid Earth, the oceans and their living resources, the atmosphere, and the space environment of the Earth, and to assess the socioeconomic impact of natural and technological changes in the environment.

The six Major Line Components of NOAA regularly produce various types of scientific and technical information in the following kinds of publications:

PROFESSIONAL PAPERS — Important definitive research results, major techniques, and special investigations.

TECHNICAL REPORTS—Journal quality with extensive details, mathematical developments, or data listings.

TECHNICAL MEMORANDUMS — Reports of preliminary, partial, or negative research or technology results, interim instructions, and the like.

CONTRACT AND GRANT REPORTS—Reports prepared by contractors or grantees under NOAA sponsorship.

TECHNICAL SERVICE PUBLICATIONS—These are publications containing data, observations, instructions, etc. A partial listing: Data serials; Prediction and outlook periodicals; Technical manuals, training papers, planning reports, and information serials; and Miscellaneous technical publications.

ATLAS—Analyzed data generally presented in the form of maps showing distribution of rainfall, chemical and physical conditions of oceans and atmosphere, distribution of fishes and marine mammals, ionospheric conditions, etc.

Information on availability of NOAA publications can be obtained from:

ENVIRONMENTAL SCIENCE INFORMATION CENTER
ENVIRONMENTAL DATA SERVICE
NATIONAL OCEANIC AND ATMOSPHERIC ADMINISTRATION
U.S. DEPARTMENT OF COMMERCE

3300 Whitehaven Street, N.W.
Washington, D.C. 20235