

AD-A119 133

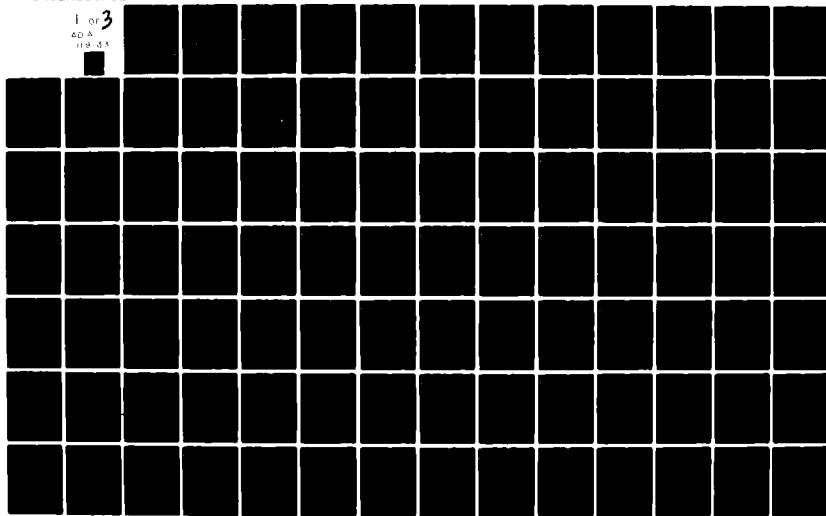
LITTON SYSTEMS INC VAN NUYS CALIF DATA SYSTEMS DIV
CONTROL DISPLAY UNIT PROGRAM.(U)
1978

F/G 9/2

UNCLASSIFIED

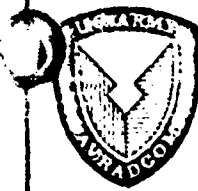
NL

1 of 3
ADA
119 133



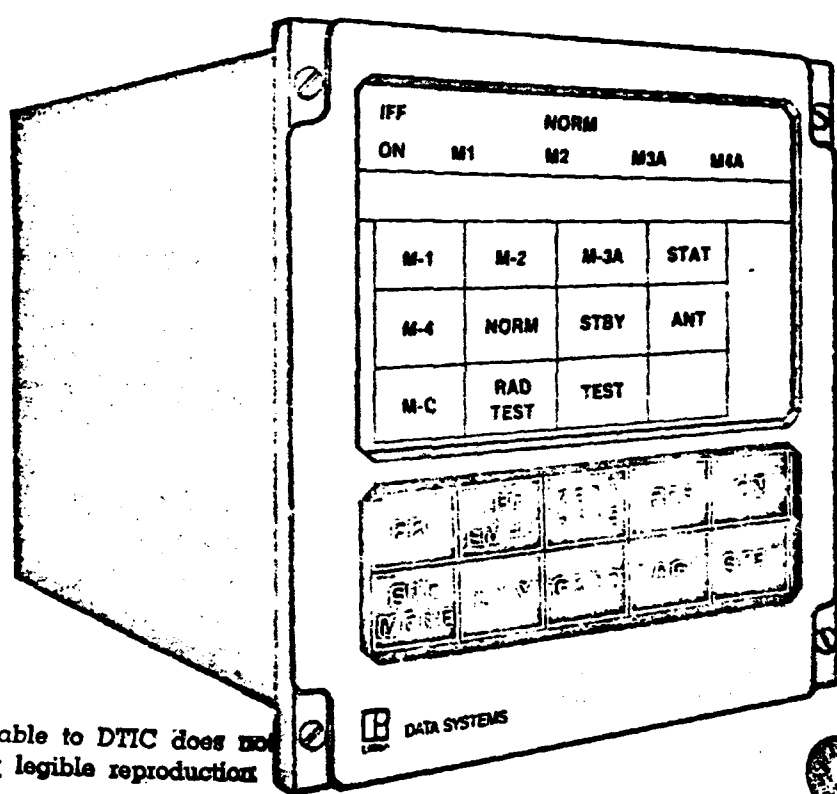
AD A119133

DTIC FILE COPY



CONTROL DISPLAY UNIT PROGRAM FINAL REPORT

CLIN 004, CDRL C002



Copy available to DTIC does not permit fully legible reproduction

1978

DTIC
SELECTED
AUG 19 1982

Prepared for
U. S. Army Electronics Command
Fort Monmouth, New Jersey 07703

Presented by
Litton Data Systems
8000 Woodley Avenue
Van Nuys, California 91409

This document has been approved for public release and sale; its distribution is unlimited.

DATA SYSTEMS

12027-1A

82 08 18 032

DISCLAIMER NOTICE

**THIS DOCUMENT IS BEST QUALITY
PRACTICABLE. THE COPY FURNISHED
TO DTIC CONTAINED A SIGNIFICANT
NUMBER OF PAGES WHICH DO NOT
REPRODUCE LEGIBLY.**

TABLE OF CONTENTS

INTRODUCTION

SECTION 1

DESIGN DETAILS

SCHEMATICS, CARDS

SCHEMATICS, DISPLAY

PANEL MEMBRANCE SWITCH

BILL OF MATERIALS

SECTION 2

SOFTWARE DESCRIPTION

SECTION 3

REPORT OF SIGNIFICANT CHANGES TO DESIGN REVIEW BASELINE

APPENDIX

TASK REPORT

RESULTS OF "STATE OF ART" REVIEW

FUNCTION FLOW AND ALLOCATION

DESCRIPTION OF ALTERNATE DESIGNS

DESCRIPTION OF PROPOSED FINAL
APPROACH

APPLICATION NOTE FOR MULTIPLEX

TERMINAL UNIT (MTI-110)

FL-88 on file

A	23	CP
---	----	----



INTRODUCTION

This information contained within this document satisfies the requirement for a final report of the Control Display Unit demonstration unit. The final report describes a Control Display Unit which, in the course of evolving, changed from a breadboard configured unit capable of physical and electrical interface to one of suitcase configuration, self-contained with dummy responses as required for presentation purposes.

It is the intent of this report to describe, by the provision of all documentation generated during the contract period, the physical and electrical elements of the suitcase Control Display Unit.

A

SECTION 1 DESIGN DETAILS

Schematics, Cards

Schematics, Display

Panel Membrane Switch

Bill of Materials

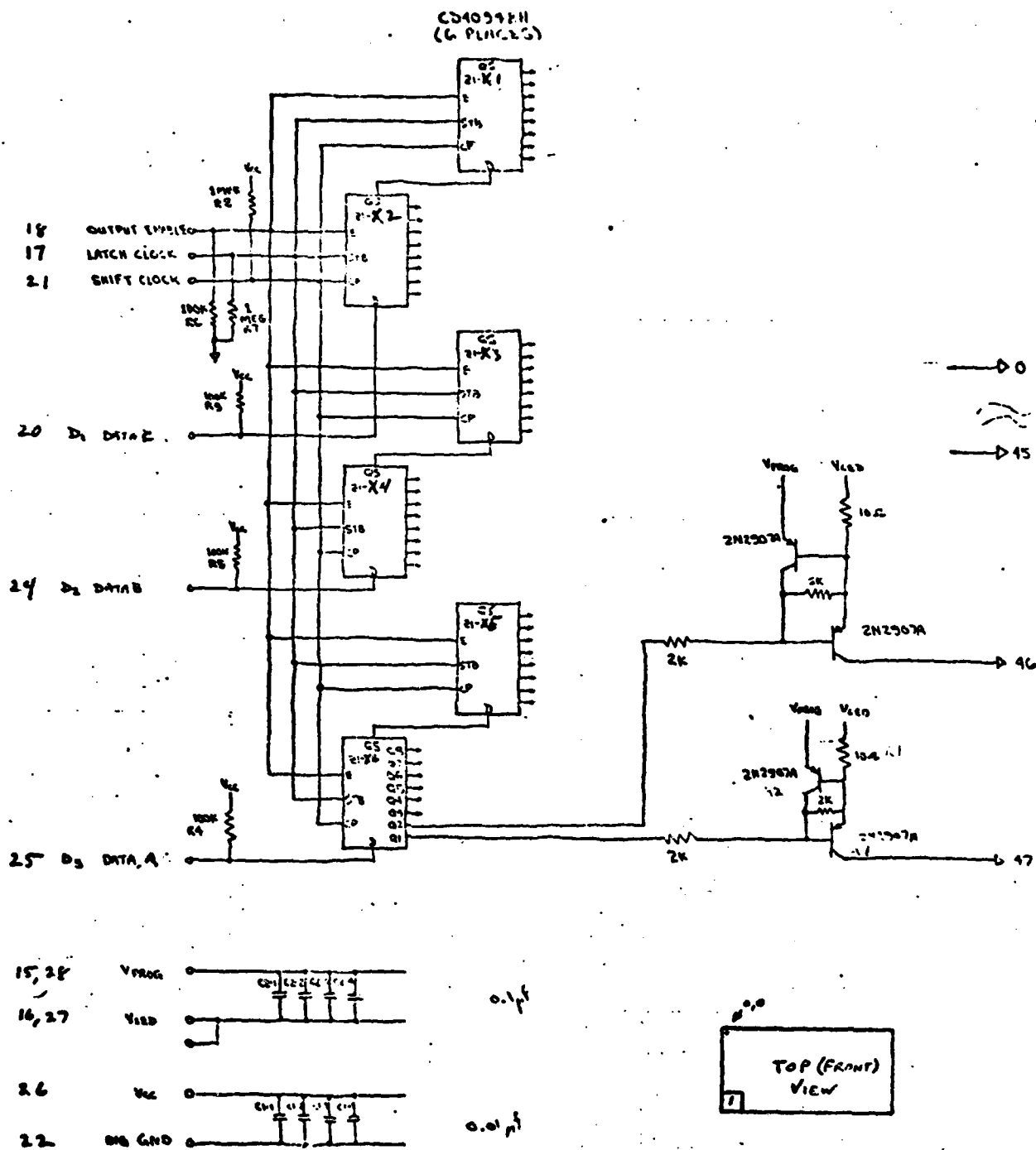
CDU

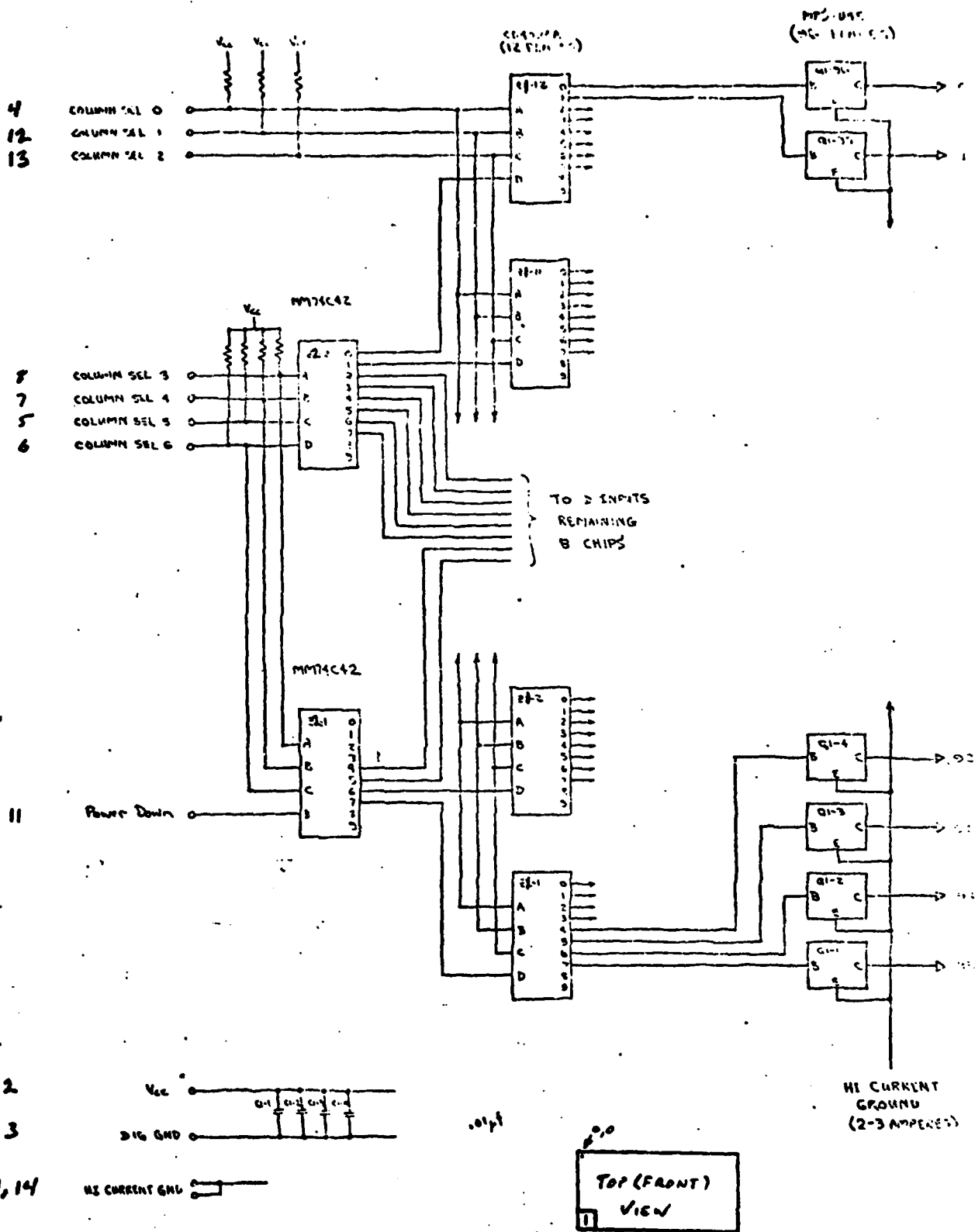
SCHEMATICS

C

DISPLAY

MODULE SCHEMATIC





NOTES:

ALL RESISTORS 1 MEGOHM

CDU FRONT

PANEL MEMBRANE

SWITCH

C

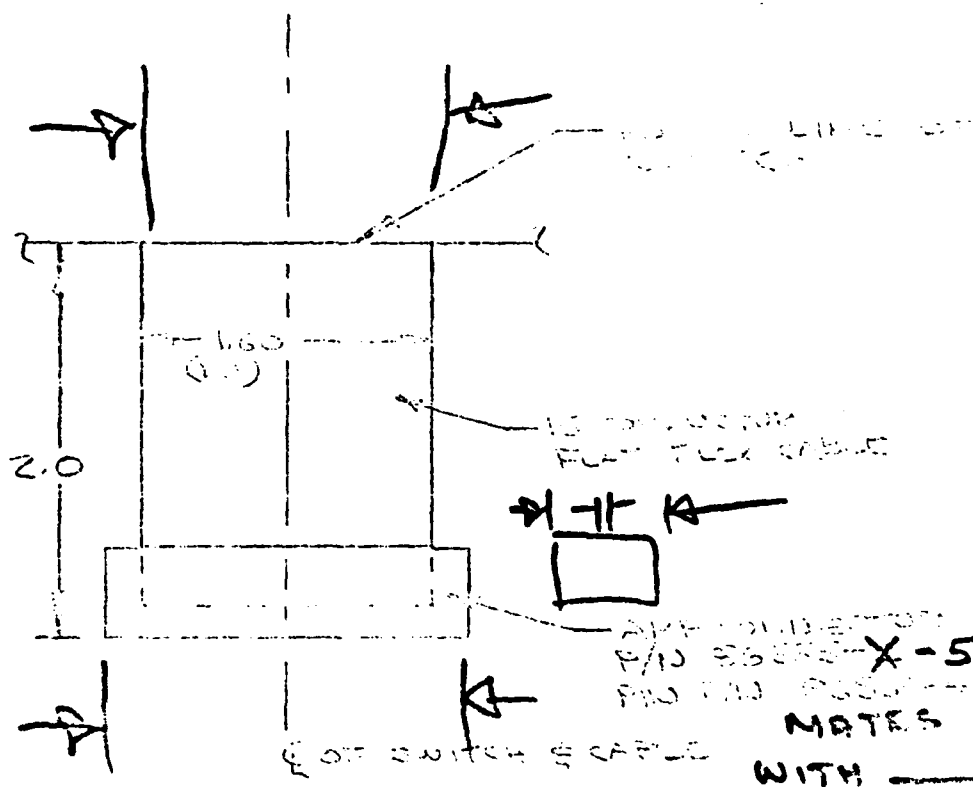
1. The first of these is the fact that the
The first of these is the fact that the

[illegible]

3. ^{ALL} BAND RADII —

4. MTL & GEN Practices PIR
AMP DWS -

5.



FRACTIONS & 1/G_s, ANGLES ± 6° 30' SURFACE TEXTURE

1961

§ 44 T.C.

100

111

•CAL-1

1. 1. 1.

10

NOV 1964

192A

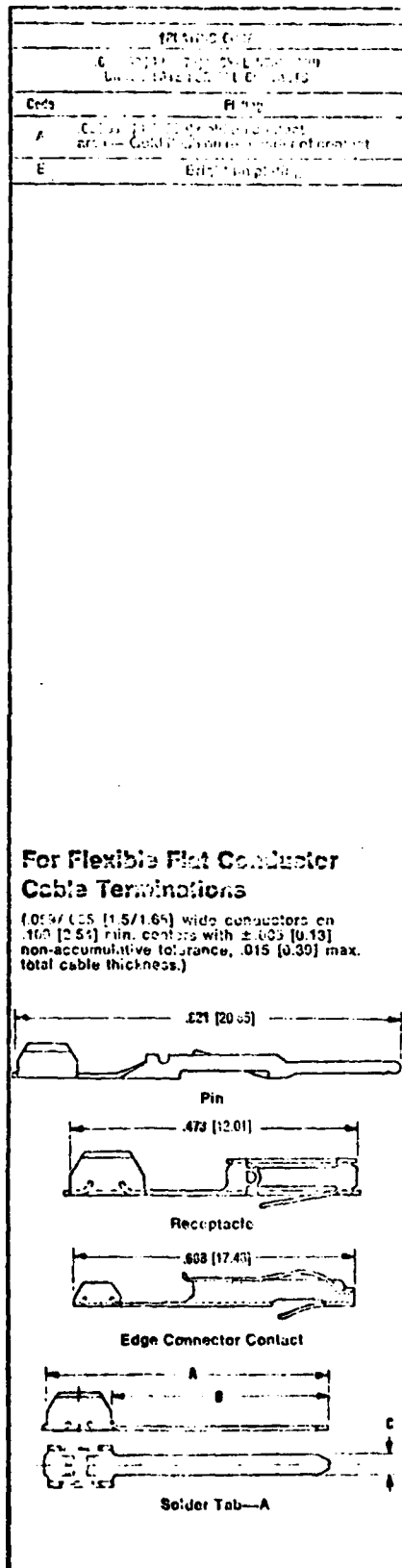
ALIP Incorporated
Hawthorne, N.J.

OVERLAY SWITCH

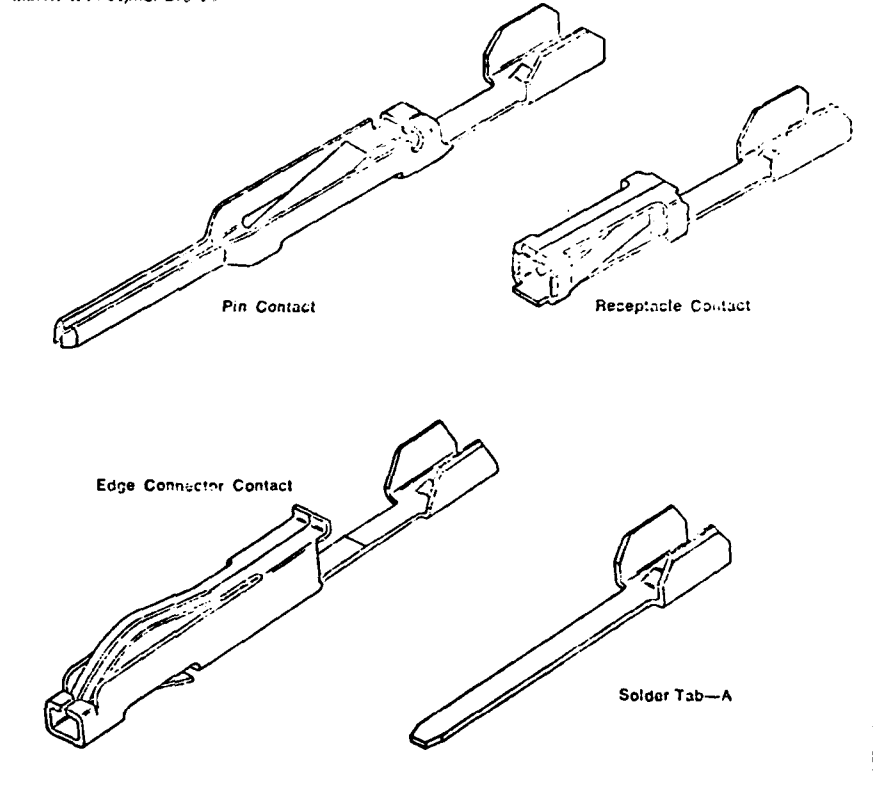
Controls

Dimensions:

1. All dimensions in inches and millimeters. Values in brackets are metric equivalents.
2. Contact and dimensions in inches over millimeters.



Materials: Phosphor Bronze



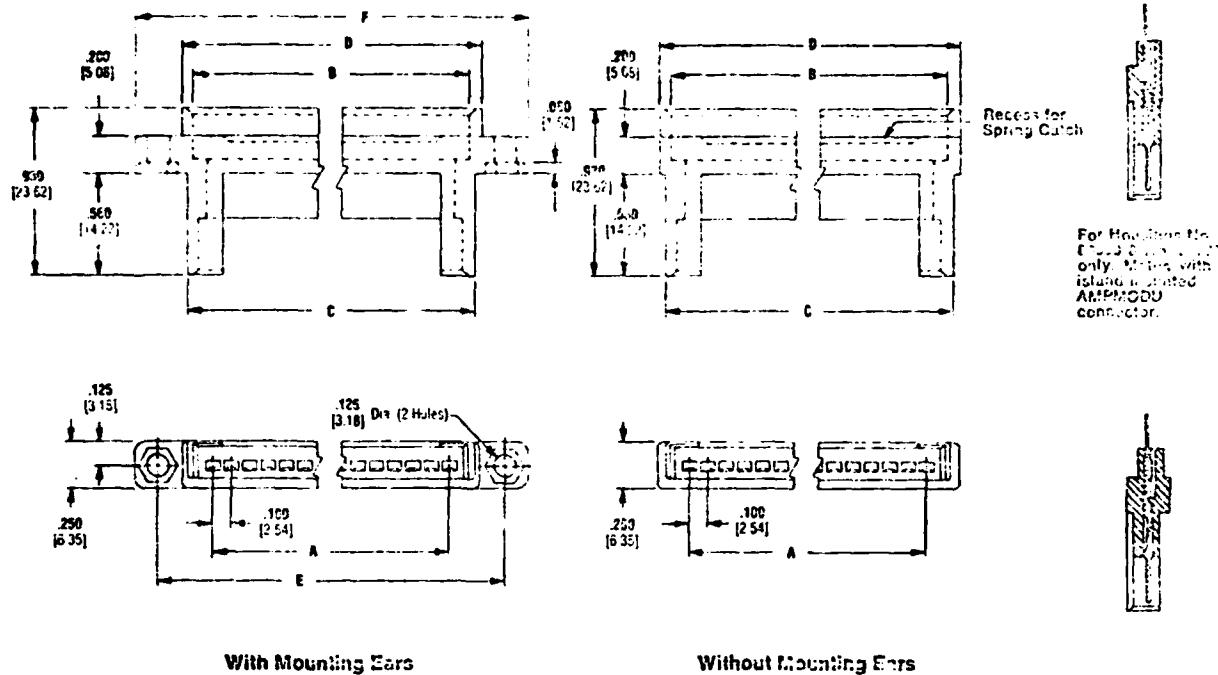
Type of Contact	Contact	Substrate Material	Location
PIN	A	65/35-8	Bottom
RECEPTACLE	A	65/35-8	Bottom
EDGE CONNECTOR CONTACT	A	65/42-4	Bottom

Type of Contact	A	Dimensions	Plating	Substrate Material	Location
SOLDER TAB—A (not for use in housings)					

Single Row Pin Housing

Dimensioning:

1. All dimensions are in inches and millimeters.
2. Conversion factor: 1 inch = 25.4 millimeters.



With Mounting Ears

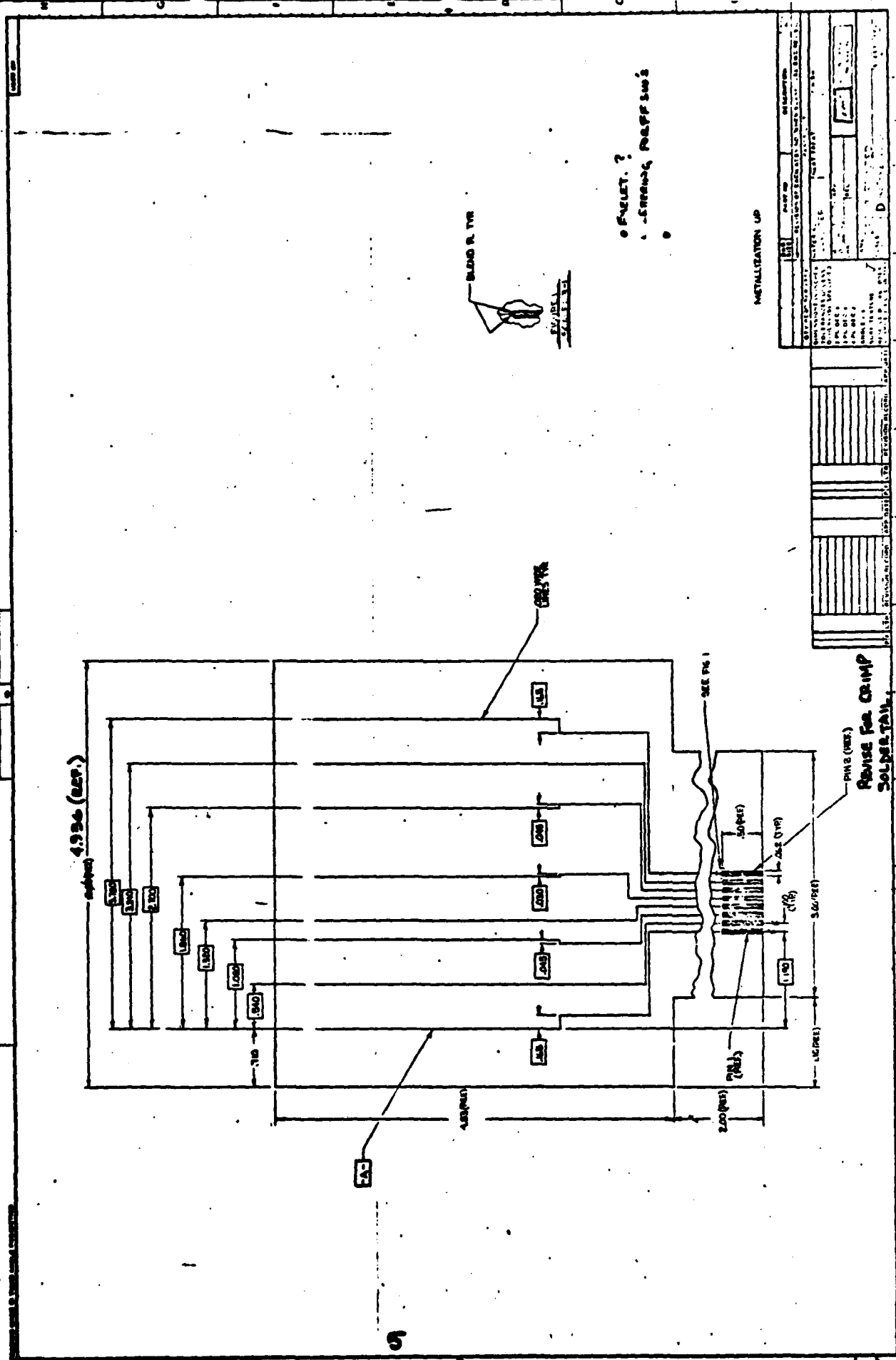
Without Mounting Ears

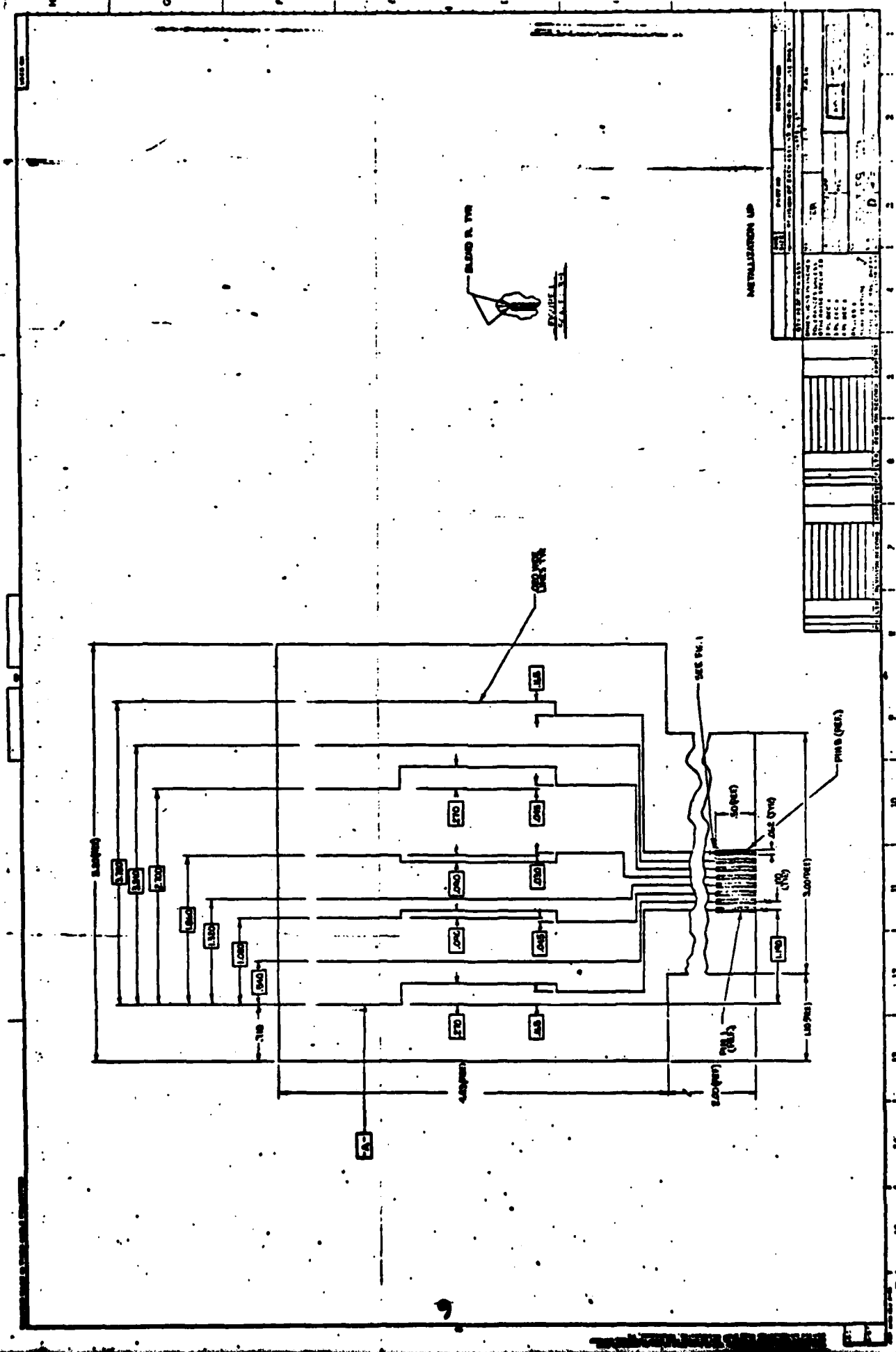
Housing Material: Black Glass Filled Nylon

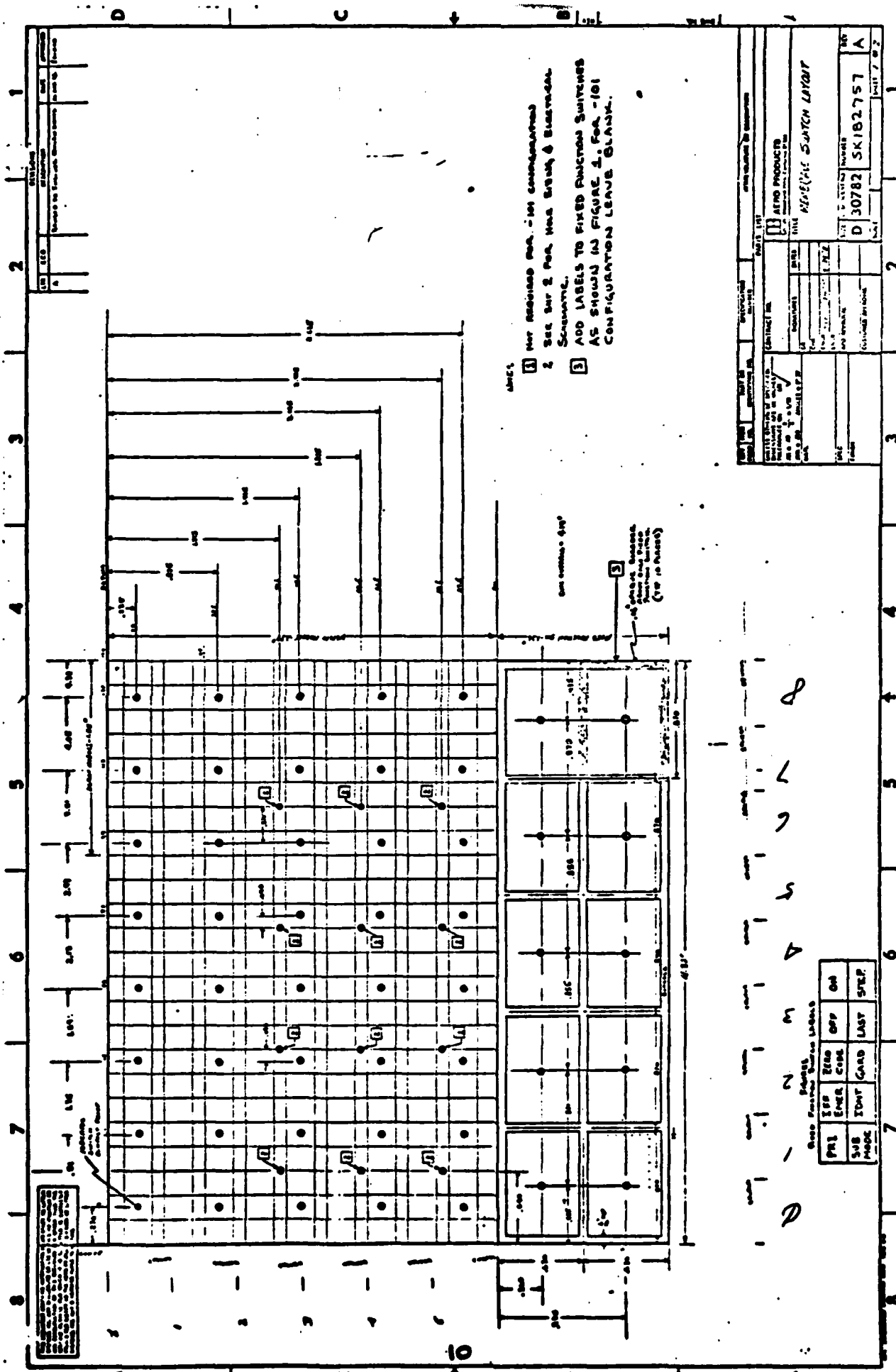
No. of Positions	Dimensions						Housing for Single Cable or Round Wire with Mounting Ears	Housing for Double (Chaining Cable) Cable with Mounting Ears	Housing for Single Cable or Round Wire without Mounting Ears	Housing for Double (Chaining Cable) Cable without Mounting Ears	Extraction Tool**
	A	B	C	D	E	F					
9	.630 20.32	1.036 26.05	1.080 27.43	1.150 29.21	1.410 35.81	1.660 42.16	86562-3	86777-1	86555-2	86776-1	91047-1
12*	1.100 27.94	1.326 33.69	1.383 35.05	1.459 36.83	—	—	—	—	86560-2	86778-1	1-01047-1
18	1.700 43.18	1.928 49.12	1.980 50.29	2.050 52.07	2.310 58.67	2.560 65.02	86562-9	86777-9	86555-5	86776-9	1-01047-1
19*	1.800 45.72	2.028 51.45	2.080 52.83	2.150 54.61	—	—	—	—	86560-1	86778-3	91047-2
19	1.800 45.72	2.028 51.45	2.080 52.83	2.150 54.61	2.410 61.21	2.660 67.56	86562-4	86777-3	86555-1	86776-3	91047-2
20	2.000 50.80	2.226 56.74	2.280 58.12	2.350 59.68	2.610 66.28	2.860 72.93	86562-2	86777-5	86555-3	86776-5	91047-3
33	3.200 81.28	3.426 87.02	3.480 88.39	3.550 90.17	3.810 96.77	4.060 103.12	86567-1	86777-7	86555-4	86776-7	91047-4

*Housings to mate with island mounted AMPMODU Connectors

**Extraction Tool Part No. 91092-1 for Round Wire Contact



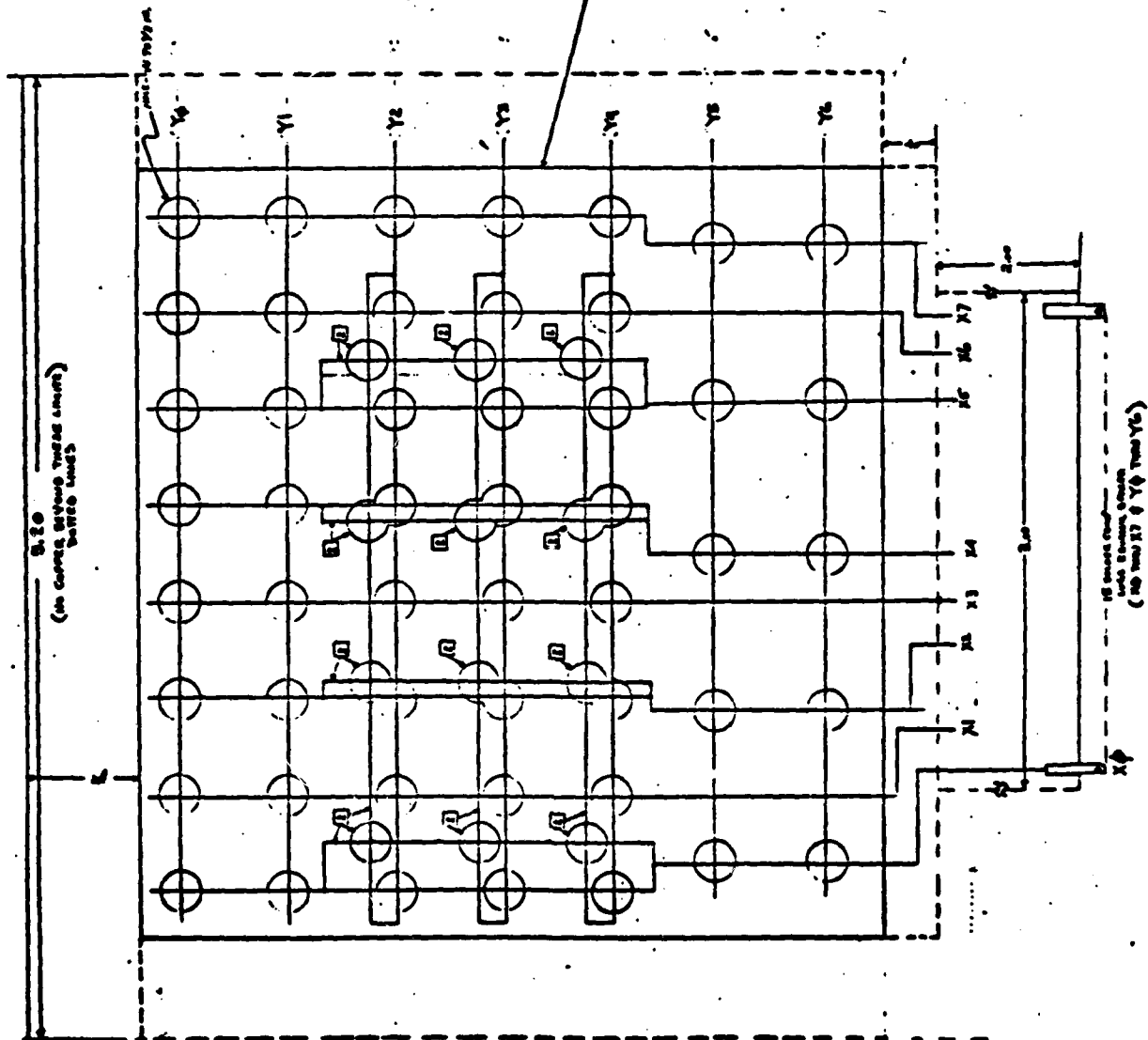




- 1 ☒ 1) NOT RECORDED FOR - IN COMPLETION
- 2 ☒ 2) See Bar 2 For Non Entry & Electrical Square.
- 3 ☒ 3) ADD LABELS TO FIXED FUNCTION SWITCHES AS SHOWN IN FIGURE 1. FOR - (01) CONFIGURATION LEAVE BLANK.

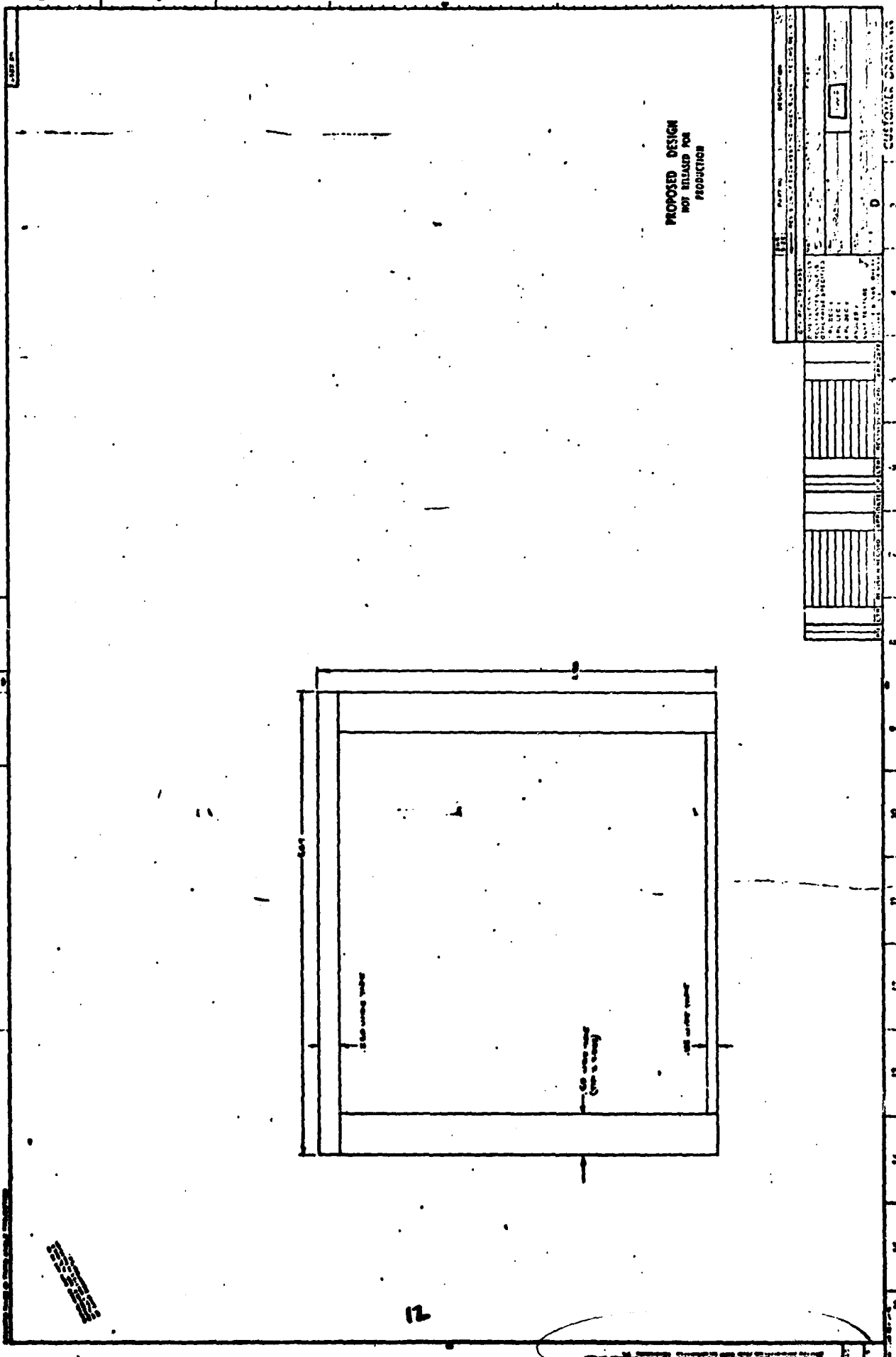
[illegible]

RACE FACTORS SURVEY LABELS			
PAI	ISS EMER	ZERO CODE	OFF OH
SUB MADE	TDOT	CARD	LAST STEP.



- Notes:
1. All dimensions are in inches.
 2. All dimensions are to center of hole unless otherwise noted.
 3. All dimensions are to bottom of hole unless otherwise noted.
 4. All dimensions are to center of hole unless otherwise noted.

MEMORIAL JOURNAL ENTRY
SK102757

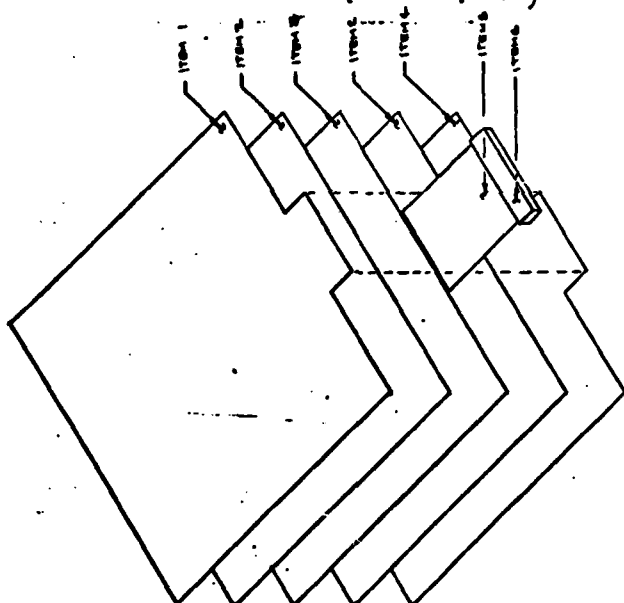


PROPOSED DESIGN
NOT INTENDED FOR
PRODUCTION

REV	DATE	BY	CHKD	APPD
1	10/10/00	J. J. J.	J. J. J.	J. J. J.
2	10/10/00	J. J. J.	J. J. J.	J. J. J.
3	10/10/00	J. J. J.	J. J. J.	J. J. J.
4	10/10/00	J. J. J.	J. J. J.	J. J. J.
5	10/10/00	J. J. J.	J. J. J.	J. J. J.
6	10/10/00	J. J. J.	J. J. J.	J. J. J.
7	10/10/00	J. J. J.	J. J. J.	J. J. J.
8	10/10/00	J. J. J.	J. J. J.	J. J. J.
9	10/10/00	J. J. J.	J. J. J.	J. J. J.
10	10/10/00	J. J. J.	J. J. J.	J. J. J.
11	10/10/00	J. J. J.	J. J. J.	J. J. J.
12	10/10/00	J. J. J.	J. J. J.	J. J. J.
13	10/10/00	J. J. J.	J. J. J.	J. J. J.
14	10/10/00	J. J. J.	J. J. J.	J. J. J.
15	10/10/00	J. J. J.	J. J. J.	J. J. J.
16	10/10/00	J. J. J.	J. J. J.	J. J. J.
17	10/10/00	J. J. J.	J. J. J.	J. J. J.
18	10/10/00	J. J. J.	J. J. J.	J. J. J.
19	10/10/00	J. J. J.	J. J. J.	J. J. J.
20	10/10/00	J. J. J.	J. J. J.	J. J. J.

CUSTOMER DRAWING

13



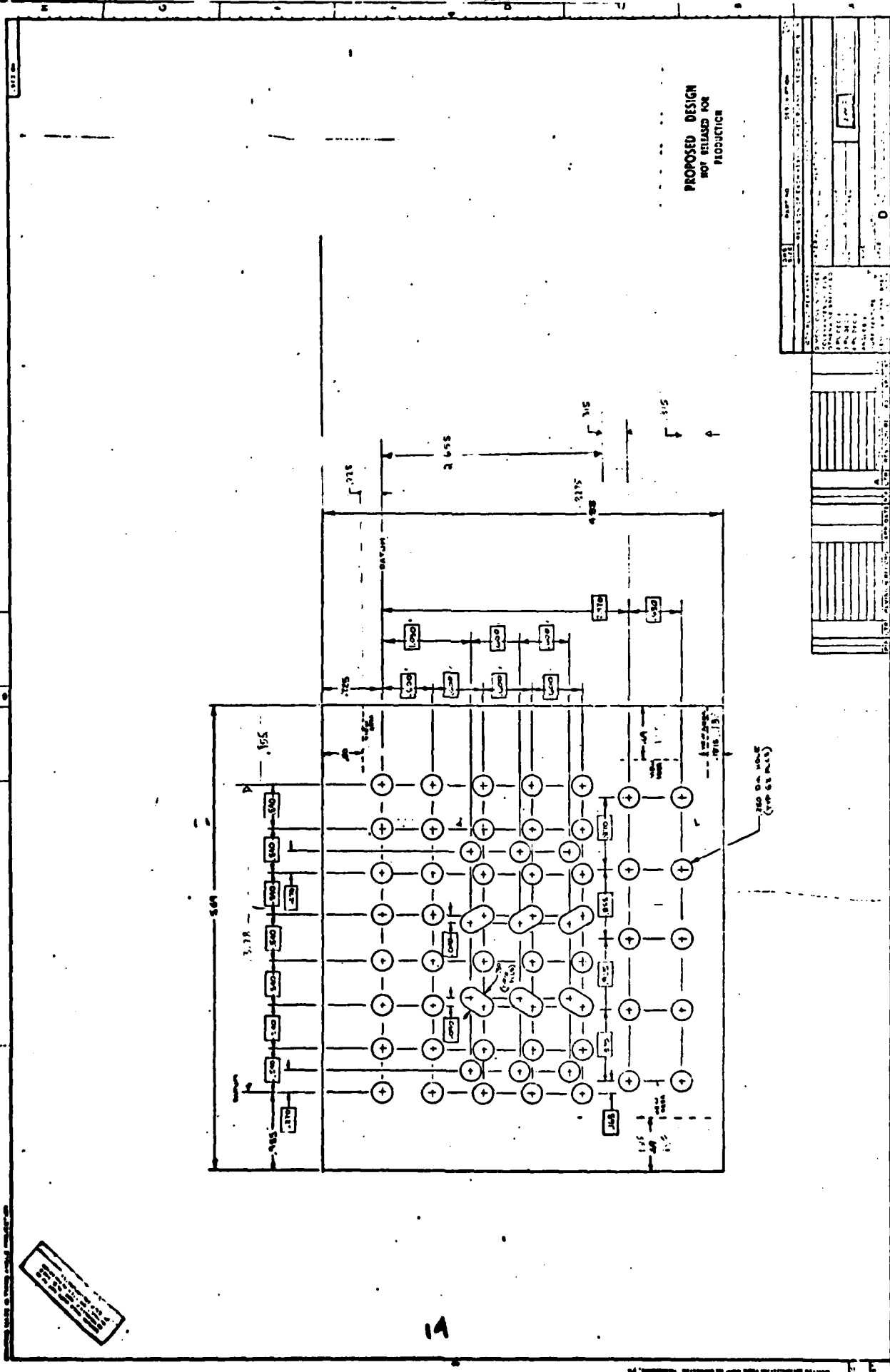
TAPE BONDING TOGETHER

PROPOSED DESIGN
NOT READY FOR
PRODUCTION

ITEM	DESCRIPTION	QTY	UNIT	DATE
1	ITEM 1	1	EA	10/1/74
2	ITEM 2	1	EA	10/1/74
3	ITEM 3	1	EA	10/1/74
4	ITEM 4	1	EA	10/1/74
5	ITEM 5	1	EA	10/1/74
6	ITEM 6	1	EA	10/1/74
7	ITEM 7	1	EA	10/1/74
8	ITEM 8	1	EA	10/1/74
9	ITEM 9	1	EA	10/1/74
10	ITEM 10	1	EA	10/1/74
11	ITEM 11	1	EA	10/1/74
12	ITEM 12	1	EA	10/1/74
13	ITEM 13	1	EA	10/1/74
14	ITEM 14	1	EA	10/1/74
15	ITEM 15	1	EA	10/1/74
16	ITEM 16	1	EA	10/1/74
17	ITEM 17	1	EA	10/1/74
18	ITEM 18	1	EA	10/1/74
19	ITEM 19	1	EA	10/1/74
20	ITEM 20	1	EA	10/1/74

ITEM	DESCRIPTION	QTY	UNIT	DATE
1	ITEM 1	1	EA	10/1/74
2	ITEM 2	1	EA	10/1/74
3	ITEM 3	1	EA	10/1/74
4	ITEM 4	1	EA	10/1/74
5	ITEM 5	1	EA	10/1/74
6	ITEM 6	1	EA	10/1/74
7	ITEM 7	1	EA	10/1/74
8	ITEM 8	1	EA	10/1/74
9	ITEM 9	1	EA	10/1/74
10	ITEM 10	1	EA	10/1/74
11	ITEM 11	1	EA	10/1/74
12	ITEM 12	1	EA	10/1/74
13	ITEM 13	1	EA	10/1/74
14	ITEM 14	1	EA	10/1/74
15	ITEM 15	1	EA	10/1/74
16	ITEM 16	1	EA	10/1/74
17	ITEM 17	1	EA	10/1/74
18	ITEM 18	1	EA	10/1/74
19	ITEM 19	1	EA	10/1/74
20	ITEM 20	1	EA	10/1/74

CUSTOMER DRAWING

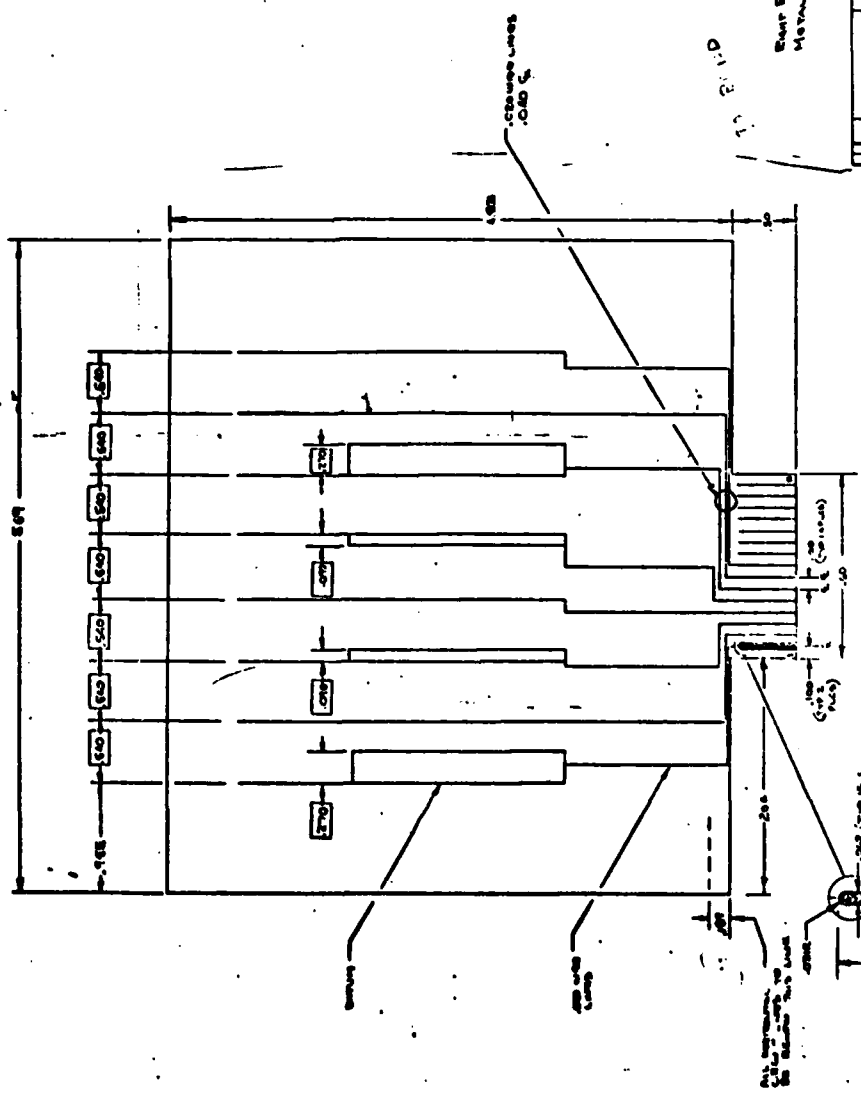


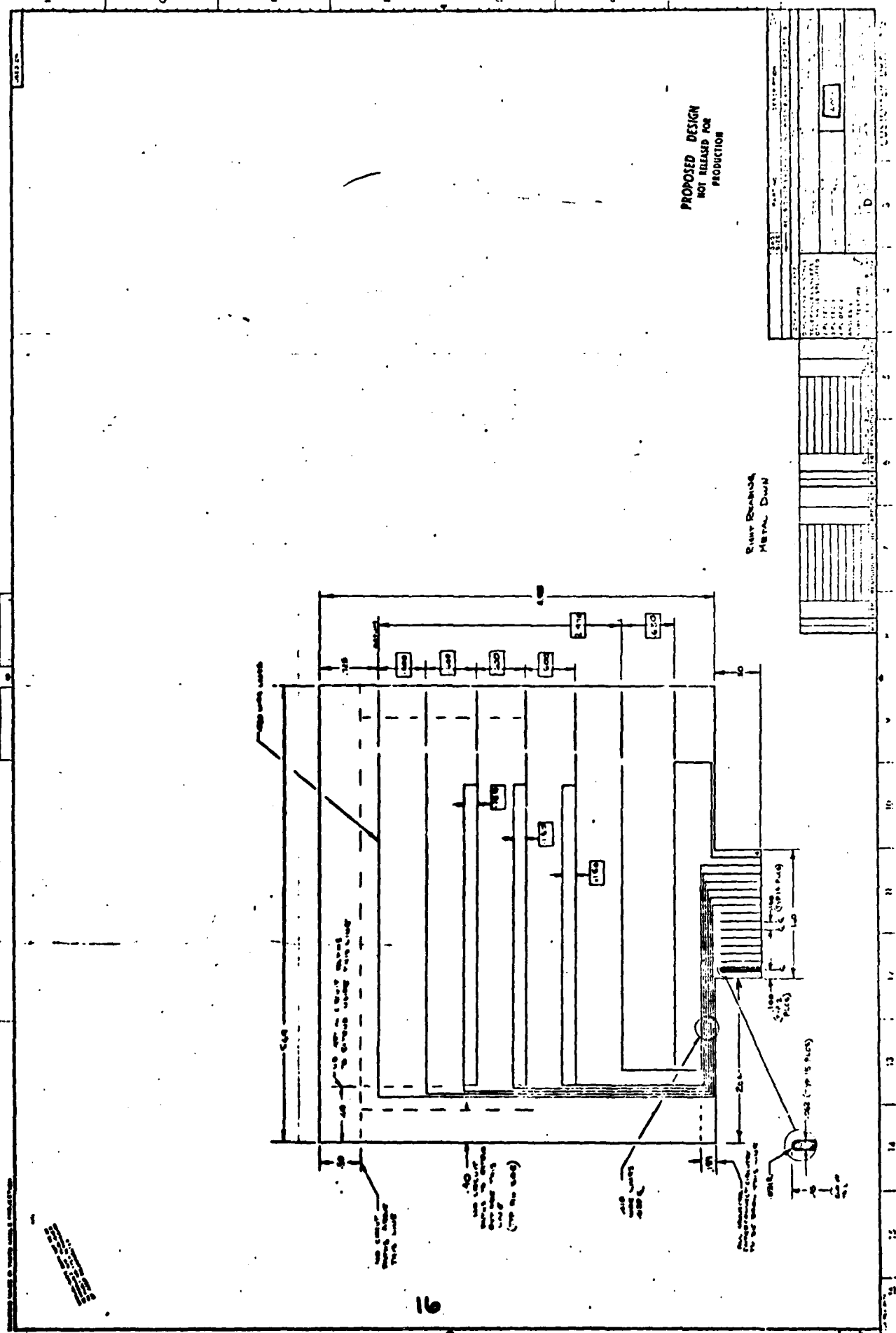
10/27/76

PROPOSED DESIGN
NOT RELEASED FOR
PRODUCTION

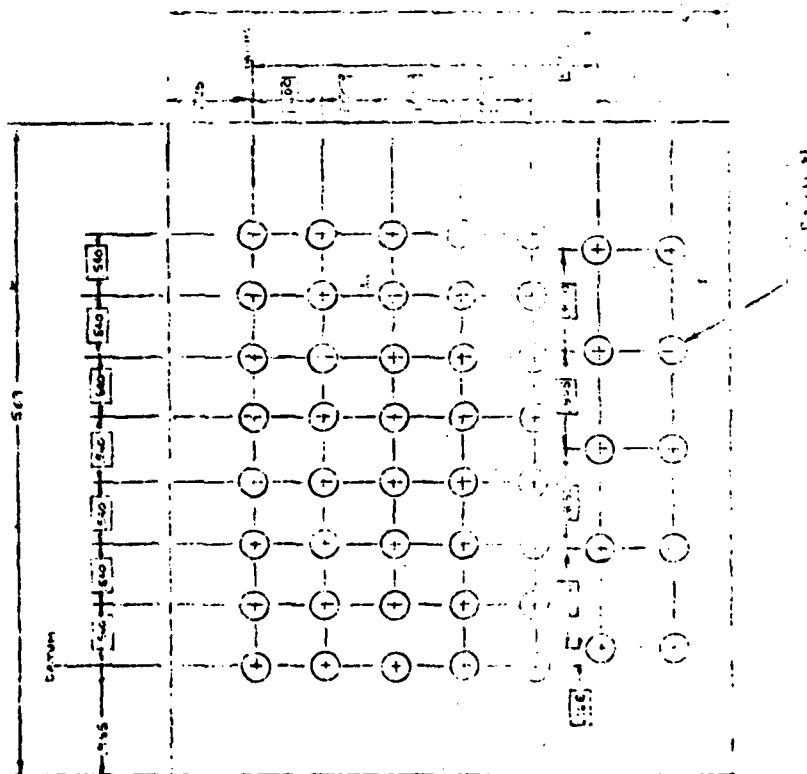
DATE	10/27/76
BY	10/27/76
CHKD BY	10/27/76
APPROVED BY	10/27/76
REVISIONS	
1	10/27/76
2	10/27/76
3	10/27/76
4	10/27/76
5	10/27/76
6	10/27/76
7	10/27/76
8	10/27/76
9	10/27/76
10	10/27/76
11	10/27/76
12	10/27/76
13	10/27/76
14	10/27/76
15	10/27/76
16	10/27/76
17	10/27/76
18	10/27/76
19	10/27/76
20	10/27/76
21	10/27/76
22	10/27/76
23	10/27/76
24	10/27/76
25	10/27/76
26	10/27/76
27	10/27/76
28	10/27/76
29	10/27/76
30	10/27/76
31	10/27/76
32	10/27/76
33	10/27/76
34	10/27/76
35	10/27/76
36	10/27/76
37	10/27/76
38	10/27/76
39	10/27/76
40	10/27/76
41	10/27/76
42	10/27/76
43	10/27/76
44	10/27/76
45	10/27/76
46	10/27/76
47	10/27/76
48	10/27/76
49	10/27/76
50	10/27/76
51	10/27/76
52	10/27/76
53	10/27/76
54	10/27/76
55	10/27/76
56	10/27/76
57	10/27/76
58	10/27/76
59	10/27/76
60	10/27/76
61	10/27/76
62	10/27/76
63	10/27/76
64	10/27/76
65	10/27/76
66	10/27/76
67	10/27/76
68	10/27/76
69	10/27/76
70	10/27/76
71	10/27/76
72	10/27/76
73	10/27/76
74	10/27/76
75	10/27/76
76	10/27/76
77	10/27/76
78	10/27/76
79	10/27/76
80	10/27/76
81	10/27/76
82	10/27/76
83	10/27/76
84	10/27/76
85	10/27/76
86	10/27/76
87	10/27/76
88	10/27/76
89	10/27/76
90	10/27/76
91	10/27/76
92	10/27/76
93	10/27/76
94	10/27/76
95	10/27/76
96	10/27/76
97	10/27/76
98	10/27/76
99	10/27/76
100	10/27/76

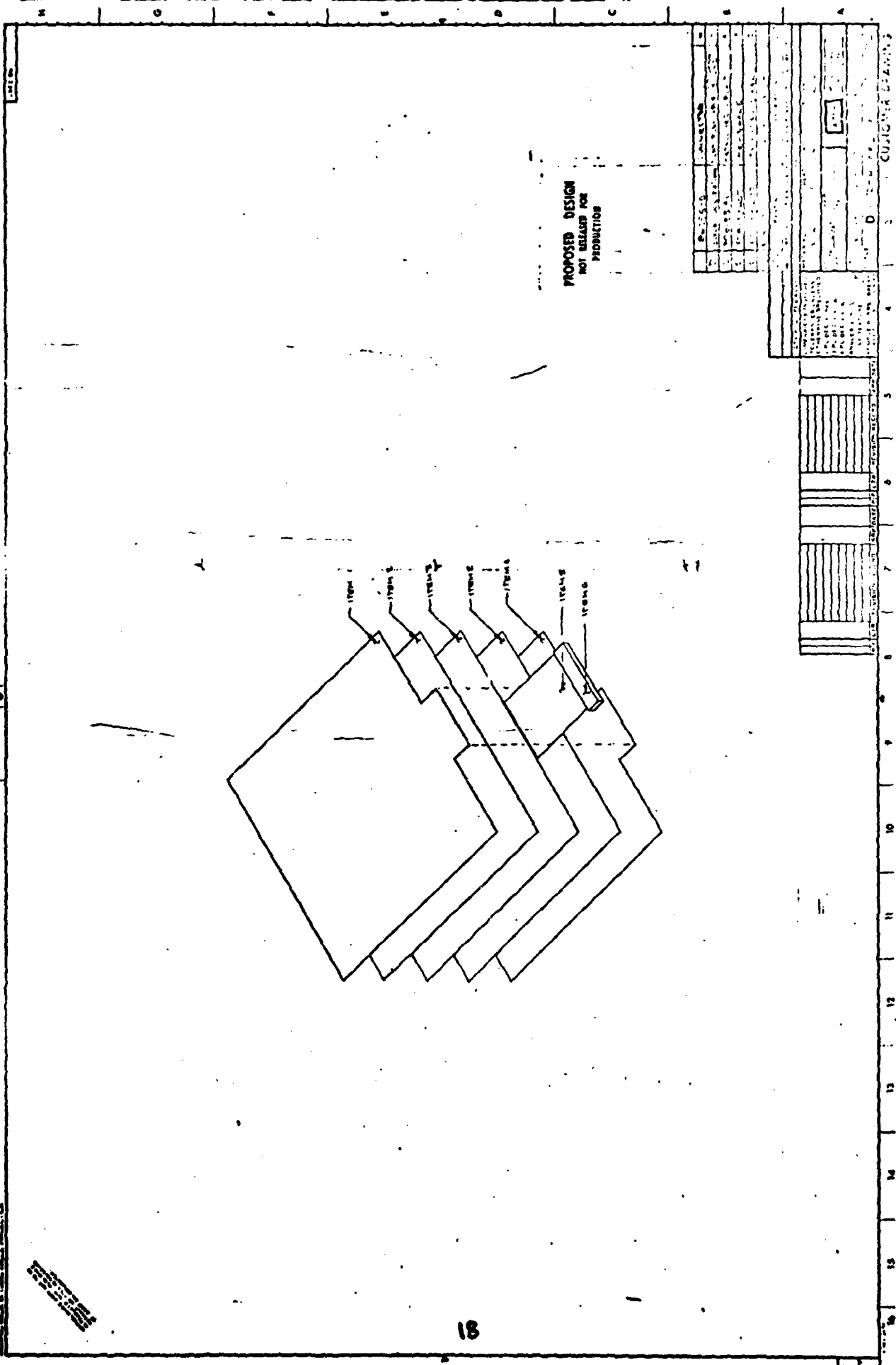
CUSTOMER DRAWING





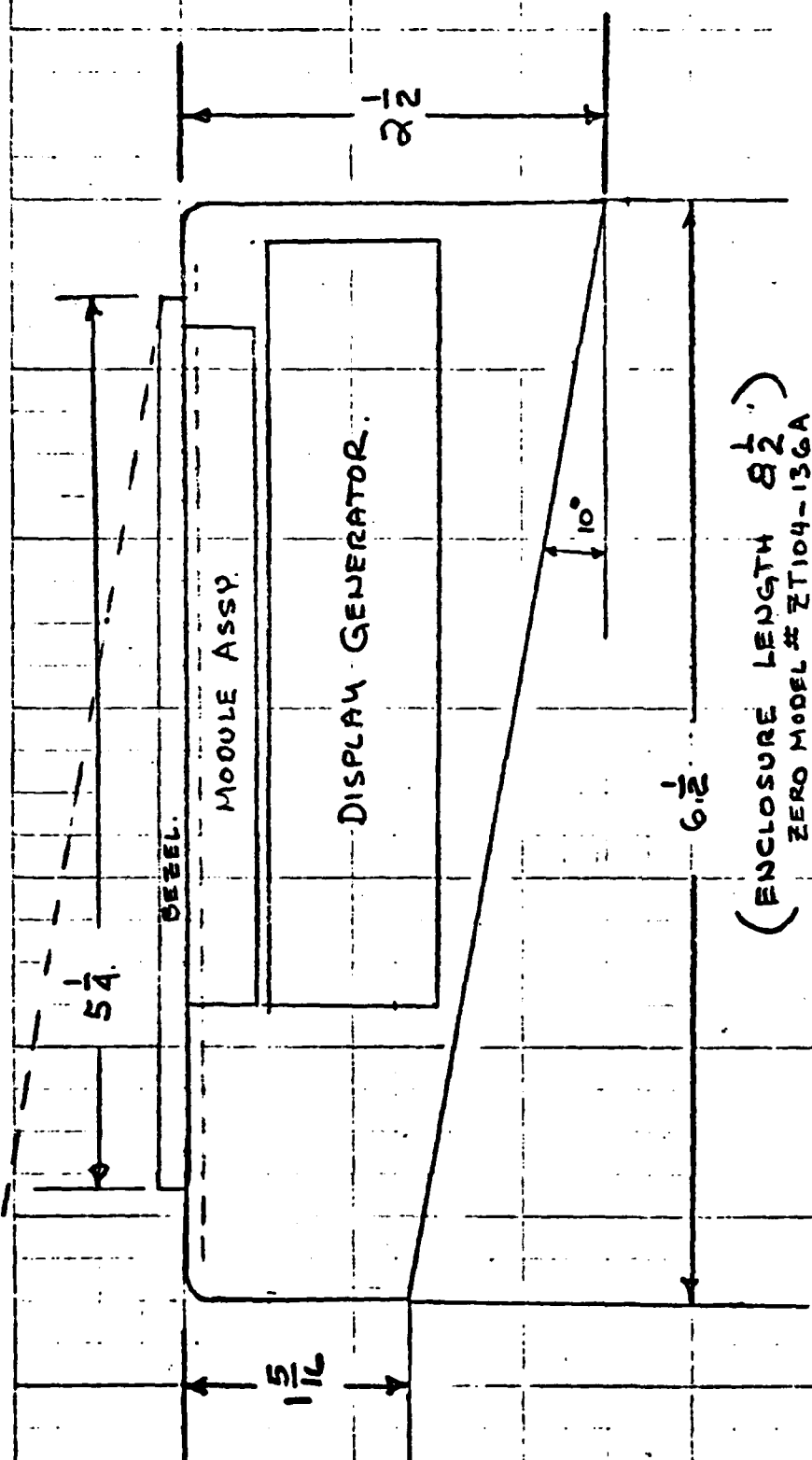
Prop. in Prop.
 207 11/11/50
 1.000000





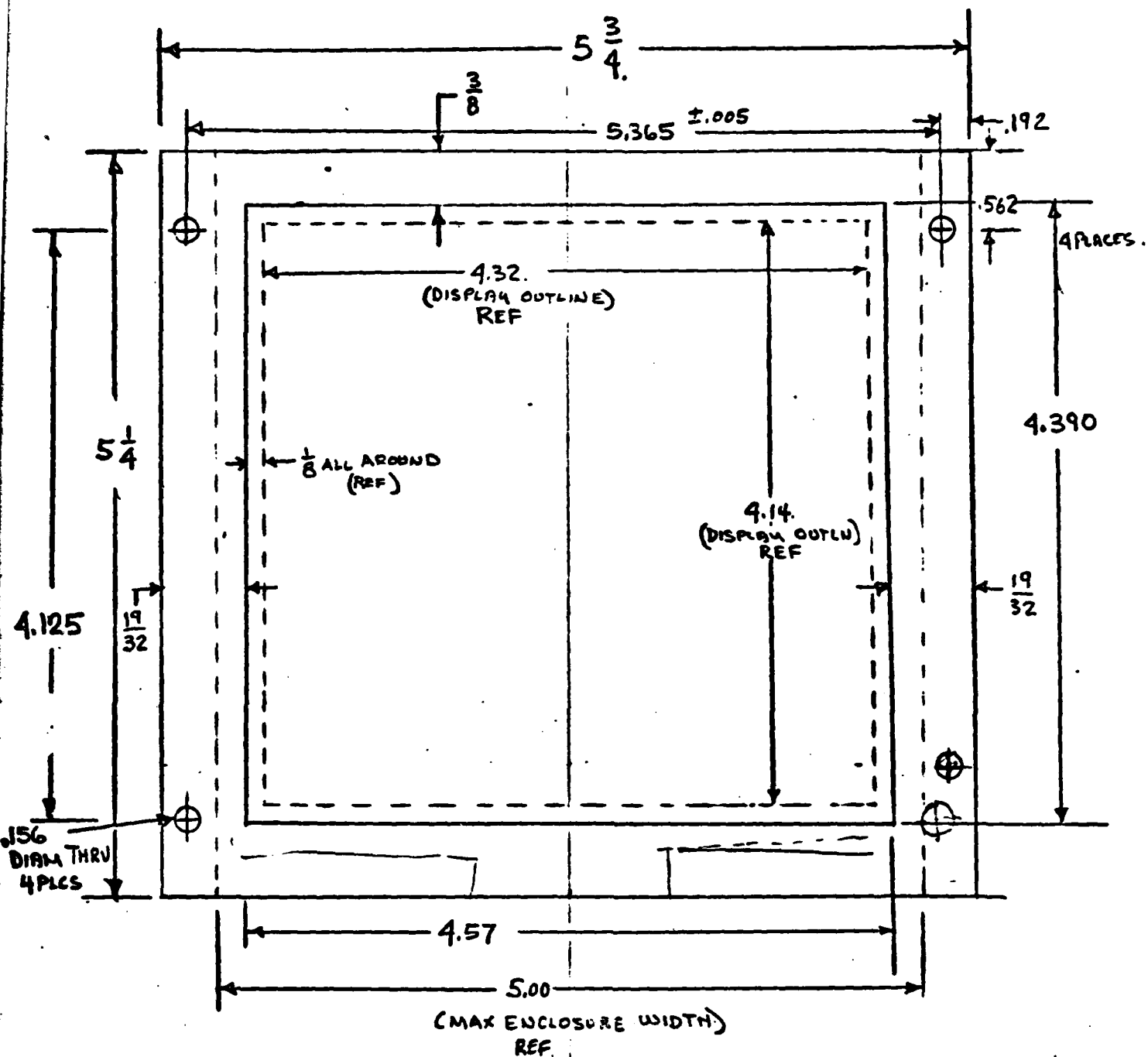
DISPLAY ASSEMBLY

MECHANICAL DESIGN



DISPLAY ASSY LAYOUT.

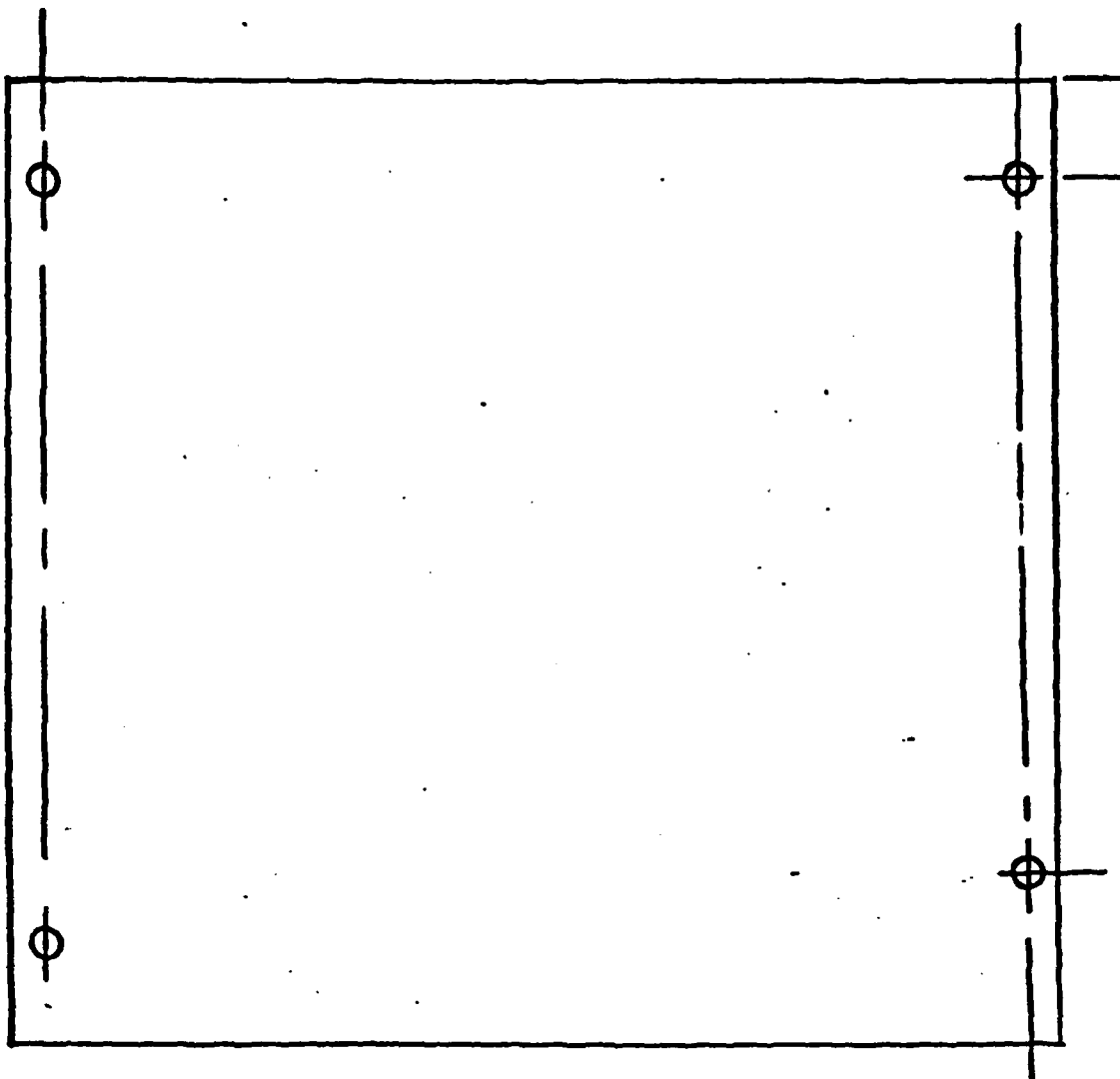
ZAVATO



NOTES:

1. MTL: $\frac{1}{16}$ 24 ST ALUM. WITH.
BLK ANODIZE FINISH.
2. REMOVE SHARP EDGES & BURRS.

BEZEL FOR CDU.



CLEAR PLASTIC MEMBRANE SWITCH
SUPPORT.

$\frac{1}{8}$ " OR $\frac{1}{16}$ " CLEAR HOMALITE 911

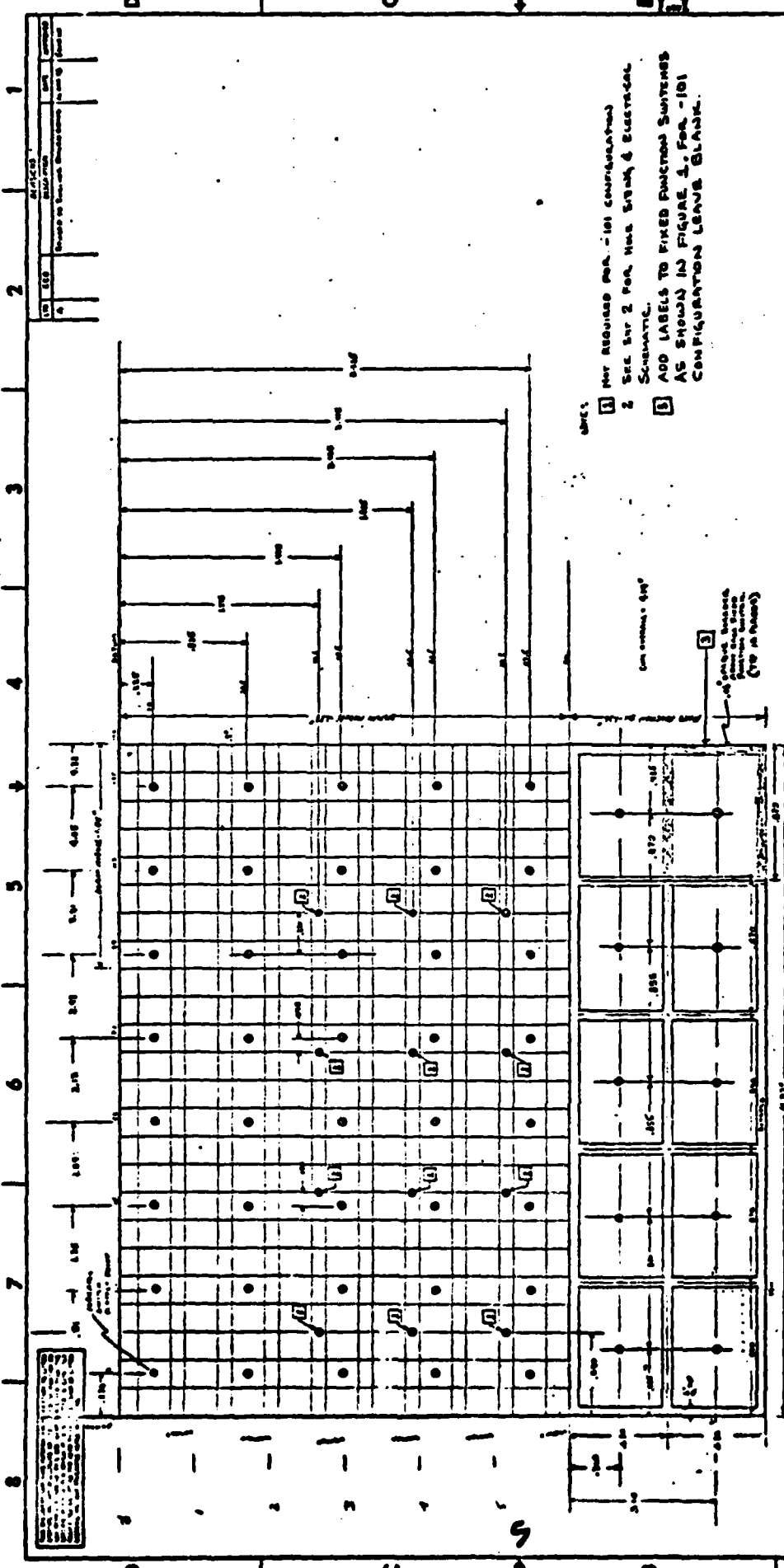
11 BROOKSIDE DRIVE
WILMINGTON, DEL.
302-652-3686.

OR

DICK WEST. (213) - 636-0377.

4

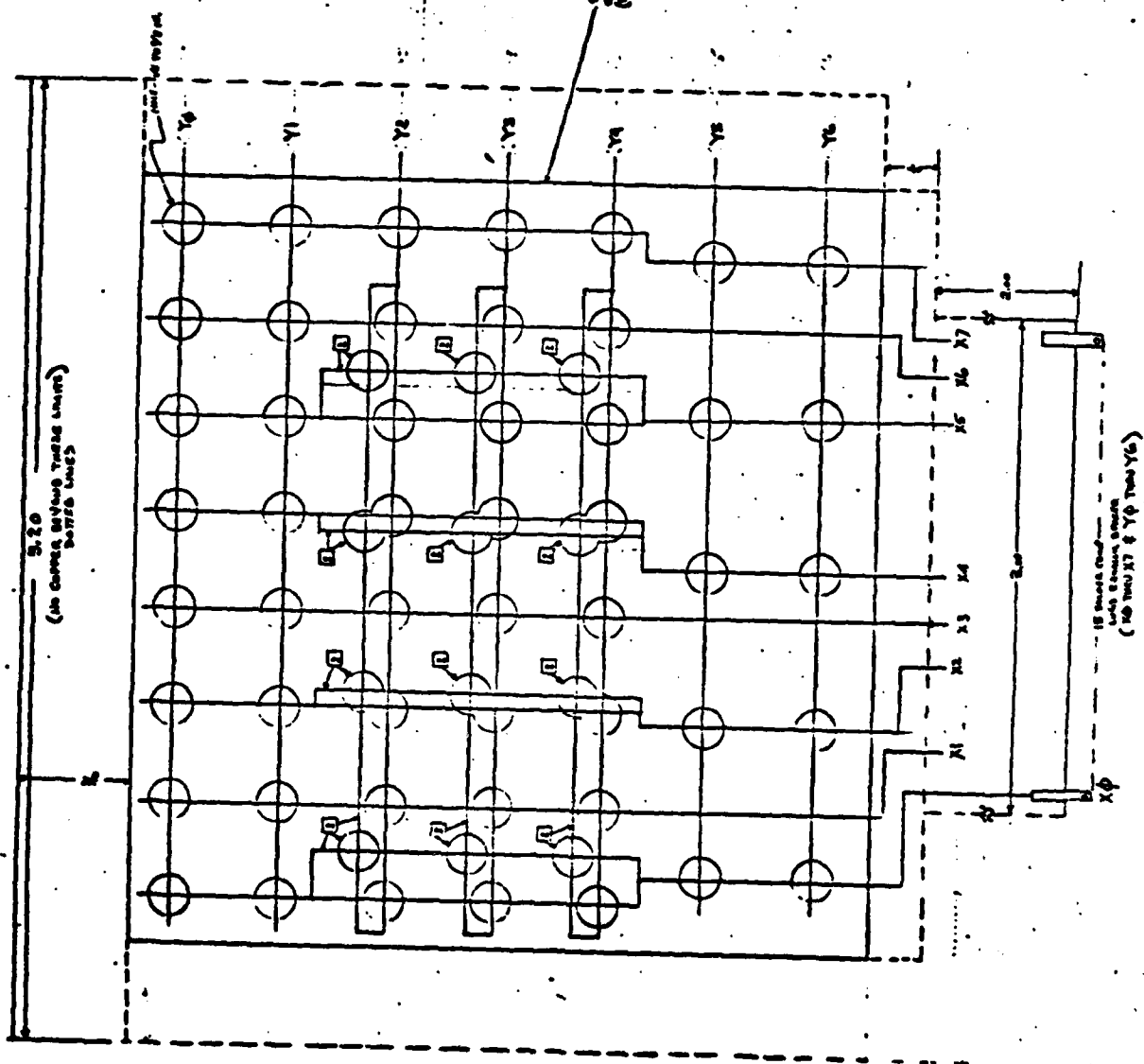
ZAVATTO.



- NOTES
- 1. NOT REQUIRED FOR -101 CONFIGURATION
 - 2. SEE FIG 2 FOR WIRE TRUNK & ELECTRICAL SCHEMATIC
 - 3. ADD LABELS TO FIXED FUNCTION SWITCHES AS SHOWN IN FIGURE 3, FOR -101 CONFIGURATION LEAVE BLANK.

DATE		REV		BY		CHKD	
10/10/50		1		J. H. H.		J. H. H.	
PROJECT NO.				CONTRACT NO.			
100-101				100-101			
TITLE				TITLE			
WIRE TRUNK & ELECTRICAL SCHEMATIC				WIRE TRUNK & ELECTRICAL SCHEMATIC			
DRAWN BY				CHECKED BY			
J. H. H.				J. H. H.			
DATE				DATE			
10/10/50				10/10/50			
SCALE				SCALE			
1:1				1:1			
SHEET NO.				SHEET NO.			
1				1			
TOTAL SHEETS				TOTAL SHEETS			
1				1			
PROJECT NAME				PROJECT NAME			
100-101				100-101			
DRAWN BY				DRAWN BY			
J. H. H.				J. H. H.			
DATE				DATE			
10/10/50				10/10/50			
SCALE				SCALE			
1:1				1:1			
SHEET NO.				SHEET NO.			
1				1			
TOTAL SHEETS				TOTAL SHEETS			
1				1			

REL	SW	END	OFF	ON
101	101	101	101	101
102	102	102	102	102
103	103	103	103	103
104	104	104	104	104
105	105	105	105	105
106	106	106	106	106
107	107	107	107	107
108	108	108	108	108
109	109	109	109	109
110	110	110	110	110



Notes:

1. GIVE MEASUREMENTS OF 2.00" (2.00" MIN.) (2.00" MIN.)
2. GIVE MEASUREMENTS OF 2.00" (2.00" MIN.) (2.00" MIN.)
3. GIVE MEASUREMENTS OF 2.00" (2.00" MIN.) (2.00" MIN.)
4. ALL TERMINALS FROM BOTTOM

MEMORIAL PHOTO COPY
SK182757

2 of 2

POWER

SYSTEM

BILL OF

MATERIALS

FRONT PAIR
ABM NO. EC211 CD1

SHEET / OF

ADVANCE BILL OF MATERIAL

LINE NO.	(D) LITTON DWG (M) MILITARY	(S) LITTON SPEC (C) MFG NO. PART NUMBER	PART NAME OR DESCRIPTION	MANUFACTURER	QTY PER ASSY.	CHG LTR	FOR PLANNING USE
1			DISPLAY MODULE	LITTON DSD	3		
2			MEMBRANE SW. ASSY	LITTON DSD	1		
3			CONNECTOR, 50-PIN				
4							
5							
6							
7							
8							
9							
10							
11							
12							
13							
14							
15							
16							
17							
18							
19							

CHGLTR	CHGLTR	ADVANCE BILL OF MATERIAL
ABM NO. <u>CPU CHKD H3</u> <u>ECONOMY</u>		SHEET <u>1</u> OF <u>1</u>

LINE NO.	(D) LITTON DWG (M) MILITARY	PART NUMBER (S) LITTON SPEC (C) MFG NO.	PART NAME OR DESCRIPTION	MANUFACTURER	QTY PER ASSY.	CHG LTR	FOR PLANNING USE
1		80255	CPU (Z1)	INTEL	1		40-PIN
2		80257	DMA CONTROLLER (Z2)	INTEL	1		40-PIN
3		80212	8-BIT I/O PORT (Z3, Z7, Z9, Z10)		4		30-PIN
4		74LS258	SELECTOR (Z4)		1		16-PIN
5		80259	INTERRUPT CONTROLLER (Z5)		1		28-PIN
6		80253	TIMER (Z6)		1		14-PIN
7		54LS045	TRI-STATE BUFFER (Z8)		1		22-PIN
8		54LS70	D-TYPE F/F. (Z11)		1		14-PIN
9		54LS02	2-INPUT AND (Z12, Z13)		2		14-PIN
10			CRYSTAL				
11							
12							
13							
14							
15							
16							
17							
18							
19							
20							



AERIAL PRODUCTS

21050 Burbank Boulevard, Woodland Hills, California 91364

ADVANCE BILL OF MATERIAL

VENTURIST CARD NO. 12
ADM NO. 100-31 CDA

SHEET 1 OF 1

LINE NO.	CHG LTH		PART NUMBER	PART NAME OR DESCRIPTION	MANUFACTURER	QTY PER ASSY.	CHG LTH	FOR PLANNING USE
	(D) LITTON DWG	(M) MILITARY						
1	2716			2K X 8 (IVE ROA (21-24)		4		24-P110
2	2114L			1K X 4 R112A (25-212)		8		18-P110
3	54LS138			DC000012-3708 (213214)		2		16-P110
4								
5								
6								
7								
8								
9								
10								
11								
12								
13								
14								
15								
16								
17								
18								
19								

LITTON

AER PRODUCTS

21050 Bu. Blvd., Culver, Woodland Hills, California 91364

CHG LTR	CHG LTR

ADVANCE BILL OF MATERIAL

ABM NO. 10 CARD 45,
ECOM CDU

SHEET 1 OF 1

LINE NO.	(D) LITTON DWG (M) MILITARY	(S) LITTON SPEC (C) MFG NO. PART NUMBER	PART NAME OR DESCRIPTION	MANUFACTURER	QTY PER ASSY.	CHG LTR	FOR PLANNING USE
1		54LS138			1		16-PIN
2		8255			2		40-PIN
3							
4							
5							
6							
7							
8							
9							
10							
11							
12							
13							
14							
15							
16							
17							
18							
19							

NO CARD AS

INTER REQUIREMENTS

TOTALS

PER CHIP

QTY 51

CHIP

54291138

1 6.8

8255

2 120

TOTAL CURRENT (M11)

247

TOTAL POWER (WATTS)

1.2

MEMORY CARD #14

POWER REQUIREMENTS

PER CHIP TOTALS

CHIP	QTY	SV	PER CHIP	SV	TOTALS
2716	4	40		160	
2114L	8	80		640	
5425/38	2	6.3		12.6	
TOTAL CURRENT (mA)				813	
POWER (WATTS)				4.07	

CPU-A3

POWER REQUIREMENTS

PER CHIP (mA)

TOTALS (mA)

CHIP	QTY	5V	5V	5V
8085	1	170		170
8257	1	120		120
8212	4	90		360
74LS258	1	12		12
8250	1	85		85
8253	1	104		104
74LS245	1	64		64
54LS76	1	400		400
54LS02	2	28		56

TOTAL CURRENT (mA)

822

TOTAL POWER (WATTAGE)

441

DISPATCH CONTROLLER A2

POWER REQUIREMENTS

CHIP	QTY	5V	8V	PER CHIP (MA)	5V	8V	TOTALS (MA)
5445163	4	19			76		
5445151	1	6.0			6		
P21144-2	6	9.0			54.0		
54451363A	2	12			24		
DS78L12	3		6.0			18	
DS78L12	1	6.0			6.0		
544510	1	1.8			1.8		
5445100	3	2.4			7.2		
5445108	2	4.4			8.8		
5445102	3	2.8			8.4		
5445101	1	3.6			3.6		

TOTAL CURRENT (MA)

682.18

POWER (WATTS)

3.4 .142

FRONT PANEL, A1

POWER REQUIREMENTS

CHIP	QTY	PER CHIP (MA)	TOTALS (MA)
------	-----	---------------	-------------

DISPLAY MODULE

3 750

2.250

TOTAL CURRENT (MA)

2.250

POWER (WATTS)

18.0

CHIP

QTY

PER CHIP (MA)

TOTALS (MA)

8V

SECTION 2

SOFTWARE DESCRIPTION

29 OCTOBER 1979

C

ISIS-II LINER V2.1 WAS INVOKED BY:

LINK :F1:ARSPAT.OBJ,:F1:CDU.OBJ,:F1:INPUT.OBJ,:F1:ABSDIS.OBJ,:F1:COM.OBJ, &
:F1:VHF.OBJ,:F1:UHF.OBJ,:F1:ADF.OBJ,:F1:CNV.OBJ,:F1:ABSHWI.OBJ, &
:F1:IFF.OBJ, &
:F1:FILBUF.OBJ,:F1:GRID.OBJ,:F1:COUDAT.OBJ,:F1:FONTPA.OBJ &
TO :F1:CDU.LNK &
MAP PRINT(:F1:CDU.MAP)

LINK MAP FOR :F1:CDU.LNK(CDU)

SEGMENT INFORMATION:

START STOP LENGTH REL NAME

		31BFH	B	CODE
		342H	B	DATA
		7EH	B	STACK
3H	1AH	18H	A	ABSOLUTE
20H	22H	3H	A	ABSOLUTE
24H	26H	3H	A	ABSOLUTE

INPUT MODULES INCLUDED:

:F1:ARSPAT.OBJ(MAIN)
:F1:CDU.OBJ(CDU)
:F1:INPUT.OBJ(IN)
:F1:ABSDIS.OBJ(DISP)
:F1:COM.OBJ(COM)
:F1:VHF.OBJ(VHF)
:F1:UHF.OBJ(UHF)
:F1:ADF.OBJ(ADF)
:F1:CNV.OBJ(CNV)
:F1:ABSHWI.OBJ(HWIO)
:F1:IFF.OBJ(IFF)
:F1:FILBUF.OBJ(FIL)
:F1:GRID.OBJ(GRID)
:F1:COUDAT.OBJ(DAT)
:F1:FONTPA.OBJ(FONTPA)

UNRESOLVED EXTERNAL NAMES:

@P0094
@P0014
@P0034
DNVTIMETO00
@P0029
@P0090
@P0027
RIBLFB
@P0095
@P0023
@P0098
@P0104
@P0091
@P0105
@P0070
@P0028
RIBLFA
DISPLAYPRESENTPOSITION
DISPLAYACTIVEMAPT
@P0069
INITONV
DNVBEARING
DNVCURMODE
DNVSCALE

ISIS-II LINKER V2.1 WAS INVOKED BY:

LINK :F1:CDU.LNK, &
:F1:DNV.OBJ,:F1:DNV1.OBJ,:F1:DNV2.OBJ,:F1:DNV3.OBJ,:F1:DNV4.OBJ,&
:F1:DNV5.OBJ,:F1:DNV.DAT.OBJ, &
PLM80.LIB,SYSTEM.LIB &
TO :F1:DNV.LNK MAP PRINT(:F1:DNV.MAP)

LINK MAP FOR :F1:DNV.LNK(DNV)

SEGMENT INFORMATION:

START	STOP	LENGTH	REL	NAME
	482BH	B		CODE
	FC3H	B		DATA
	A6H	B		STACK
3H	1AH	18H	A	ABSOLUTE
20H	22H	3H	A	ABSOLUTE
24H	26H	3H	A	ABSOLUTE

INPUT MODULES INCLUDED:

:F1:CDU.LNK(CDU)
:F1:DNV.OBJ(DNV)
:F1:DNV1.OBJ(DNV1)
:F1:DNV2.OBJ(DNV2)
:F1:DNV3.OBJ(DNV3)
:F1:DNV4.OBJ(DNV4)
:F1:DNV5.OBJ(DNV5)
:F1:DNV.DAT.OBJ(DAT)
PLM80.LIB(EP0014)
PLM80.LIB(EP0022)
PLM80.LIB(EP0025)
PLM80.LIB(EP0029)
PLM80.LIB(EP0034)
PLM80.LIB(EP0048)
PLM80.LIB(EP0069)
PLM80.LIB(EP0084)
PLM80.LIB(EP0089)
PLM80.LIB(EP0091)
PLM80.LIB(EP0094)
PLM80.LIB(EP0098)
PLM80.LIB(EP0103)
PLM80.LIB(EP0105)

ISIS-II LOCATER V2.1 INVOKED BY:

- LOCATE :F1:DNV.LNK ORDER(CODE,STACK,DATA,MEMORY) &

** CODE(23H) STACK(6000H) DATA(6030H) STACKSIZE(30H) &

** RESTARTO &

** PRINT(:F1:DNV.LOC) MAP

MEMORY MAP OF MODULE DNV

READ FROM FILE :F1:DNV.LNK

WRITTEN TO FILE :F1:DNV

MODULE START ADDRESS 0028H

START STOP LENGTH REL NAME

0H	2H	3H	A	ABSOLUTE
3H	1AH	18H	A	ABSOLUTE
20H	22H	3H	A	ABSOLUTE
24H	26H	3H	A	ABSOLUTE
28H	4852H	482BH	B	CODE
6000H	602FH	30H	B	STACK
6030H	6FF2H	FC3H	B	DATA
6FF3H	F6BFH	86CDH	B	MEMORY

DECLARE ROW0 LITERALLY '80H',
ROW1 LITERALLY '81H',
ROW2 LITERALLY '82H',
ROW3 LITERALLY '83H',
ROW4 LITERALLY '84H',
ROW5 LITERALLY '85H',
ROW6 LITERALLY '86H',
ROW7 LITERALLY '87H',
ROW8 LITERALLY '88H',

COL1 LITERALLY '0A1H',
COL2 LITERALLY '0A2H',
COL3 LITERALLY '0A3H',
COL4 LITERALLY '0A4H',
COL5 LITERALLY '0A5H',
COL6 LITERALLY '0A6H',
COL7 LITERALLY '0A7H',
COL8 LITERALLY '0A8H',
COL9 LITERALLY '0A9H',
COL10 LITERALLY '0AAH',
COL11 LITERALLY '0ABH',
COL12 LITERALLY '0ACH',
COL13 LITERALLY '0ADH',
COL14 LITERALLY '0AEH',
COL15 LITERALLY '0AFH',
COL16 LITERALLY '0B0H',
COL17 LITERALLY '0B1H',
COL18 LITERALLY '0B2H',
COL19 LITERALLY '0B3H',
COL20 LITERALLY '0B4H',
COL21 LITERALLY '0B5H',
COL22 LITERALLY '0B6H',
COL23 LITERALLY '0B7H',

DECLARE TRUE LITERALLY 'OFFH',
FALSE LITERALLY '0',
FOREVER LITERALLY 'WHILE 1',
SWITCH LITERALLY '40H',
CNTL#DIGIT LITERALLY '81H',
NOCLEAR#CNTL#DIGIT LITERALLY '0C1H',
NOCLEAR#DIGIT LITERALLY '41H',
DIGIT LITERALLY '1',
OCTAL LITERALLY '2',
NORTH#SOUTH LITERALLY '3',
EAST#WEST LITERALLY '4',
DN#ALPHA LITERALLY '5',
CLEAR#SW LITERALLY '450',
ENTER#SW LITERALLY '470',
SQL#OFF#SW LITERALLY '400',
OFF#SW LITERALLY '530',
SUB#MODE#SW LITERALLY '600',
LAST#SW LITERALLY '630',
STEP#SW LITERALLY '640',
CQPT LITERALLY '0',
TGT LITERALLY '1',
BEG#SYN LITERALLY '22H',

ISIS-II PL/M-80 V3.0 COMPILATION OF MODULE MAIN
OBJECT MODULE PLACED IN :F1:ABSMAT.OBJ
COMPILER INVOKED BY: PLM80 :F1:ABSMAT.SRC DATE(22DEC78)

```
          $TITLE('MAIN')
1      MAIN: DO:
2 1      DECLARE FOREVER LITERALLY 'WHILE 1':

3 1      ADD$SW0: PROCEDURE(SW0) EXTERNAL;
4 2      DECLARE SW0 BYTE;
5 2      END;
6 1      INIT$HARDWARE: PROCEDURE EXTERNAL;
7 2      END;
8 1      INIT$CDU: PROCEDURE EXTERNAL;
9 2      END;
10 1     CDU: PROCEDURE EXTERNAL;
11 2     END;
12 1     PROCESS$SWITCH: PROCEDURE EXTERNAL;
13 2     END;

14 1     CALL INIT$HARDWARE;
15 1     CALL INIT$CDU;
16 1     DO FOREVER;
17 2         CALL CDU;
18 2         CALL PROCESS$SWITCH;
19 2     END;

20 1     END; /* MAIN */
```

MODULE INFORMATION:

CODE AREA SIZE = 0014H 200
VARIABLE AREA SIZE = 0000H 00
MAXIMUM STACK SIZE = 0002H 20
25 LINES READ
0 PROGRAM ERROR(S)

END OF PL/M-80 COMPILATION

ISIS-II PL/M-80 V3.1 COMPILATION OF MODULE CDU
OBJECT MODULE PLACED IN :F1:CDU.OBJ
COMPILER INVOKED BY: PLM80 :F1:CDU.SRC DATE(27OCT79) DEBUG

```
          $TITLE('CDU')
          /* CLEAR, INIT$CDU, SUSPEND, RESTART, LIMIT$TEST */
1          CDU: DO:
          $NOLIST INCLUDE(:F1:CDULIT.SRC)
4 1        DECLARE MORZ LITERALLY 'OFZH',
          VERT LITERALLY 'OFDH';

5 1        READ: PROCEDURE(ICB$PTR) EXTERNAL;
6 2        DECLARE ICB$PTR ADDRESS;
7 2        END;
8 1        CLEAR: PROCEDURE(ROW) EXTERNAL;
9 2        DECLARE ROW BYTE;
10 2       END;
11 1       INSERT: PROCEDURE(NCHAR, SOURCE$PTR, DEST$PTR, DELM$CH, DELM$MASK) EXTERNAL;
12 2       DECLARE (NCHAR, DELM$CH) BYTE, (SOURCE$PTR, DEST$PTR, DELM$MASK) ADDRESS;
13 2       END;
14 1       UPDATE$LINE: PROCEDURE(ROW, COL, NUM$CH, PTR) EXTERNAL;
15 2       DECLARE (ROW, COL, NUM$CH) BYTE, PTR ADDRESS;
16 2       END;
17 1       DISPLAY: PROCEDURE(ICB$PTR) EXTERNAL;
18 2       DECLARE ICB$PTR ADDRESS;
19 2       END;
20 1       GRID: PROCEDURE(PTR) EXTERNAL;
21 2       DECLARE PTR ADDRESS;
22 2       END;
23 1       UPDATE$SCREEN: PROCEDURE EXTERNAL;
24 2       END;
25 1       INITIALIZE$MQ: PROCEDURE EXTERNAL;
26 2       END;
27 1       SCREEN$INTENSITY: PROCEDURE(LEVEL) EXTERNAL;
28 2       DECLARE LEVEL BYTE;
29 2       END;
30 1       CLEAR$LINE: PROCEDURE(LINE$NUM) EXTERNAL;
31 2       DECLARE LINE$NUM BYTE;
32 2       END;
33 1       INIT$VHF: PROCEDURE EXTERNAL;
34 2       END;
35 1       INIT$UHF: PROCEDURE EXTERNAL;
36 2       END;
37 1       INIT$ADF: PROCEDURE EXTERNAL;
38 2       END;
39 1       INIT$CNV: PROCEDURE EXTERNAL;
40 2       END;
41 1       INIT$IFF: PROCEDURE EXTERNAL;
42 2       END;
43 1       INIT$DNV: PROCEDURE EXTERNAL;
44 2       END;
45 1       VHF$SUBMODE: PROCEDURE EXTERNAL;
46 2       END;
47 1       DNV$SUBMODE: PROCEDURE EXTERNAL;
48 2       END;
```

```

49 1  UHF$SUBMODE: PROCEDURE EXTERNAL;
50 2  END;
51 1  IFF$SUBMODE: PROCEDURE EXTERNAL;
52 2  END;
53 1  ADF$SUBMODE: PROCEDURE EXTERNAL;
54 2  END;
55 1  CNV$SUBMODE: PROCEDURE EXTERNAL;
56 2  END;
57 1  DISPLAY$ACTIVE$HAYPT: PROCEDURE EXTERNAL;
58 2  END;
59 1  DISPLAY$PRESENT$POSITION: PROCEDURE(ROW) EXTERNAL;
60 2  DECLARE ROW BYTE;
61 2  END;

/* EXTERNAL VARIABLES */
62 1  DECLARE
    CURRENT$DISP(9) STRUCTURE(CHAR(24) BYTE) EXTERNAL,
    SM$INDEX BYTE EXTERNAL,
    SMV BYTE EXTERNAL,
    WHF$STATUS BYTE EXTERNAL,
    WHF$ACTIVE$CHAN BYTE EXTERNAL,
    WHF$FREQ(10) STRUCTURE(DIGITS(4) BYTE) EXTERNAL,
    UHF$STATUS BYTE EXTERNAL,
    UHF$ACTIVE$CHAN BYTE EXTERNAL,
    UHF$FREQ(10) STRUCTURE(DIGITS(6) BYTE) EXTERNAL,
    ADF$STATUS BYTE EXTERNAL,
    ADF$ACTIVE$CHAN BYTE EXTERNAL,
    ADF$FREQ(10) STRUCTURE(DIGITS(4) BYTE) EXTERNAL,
    CNV$STATUS BYTE EXTERNAL,
    CNV$ACTIVE$CHAN BYTE EXTERNAL,
    CNV$FREQ(10) STRUCTURE(DIGITS(5) BYTE) EXTERNAL,
    IFF$STATUS BYTE EXTERNAL,
    IFF$MODE BYTE EXTERNAL,
    DNV$STATUS BYTE EXTERNAL,
    DNV$RANGE BYTE EXTERNAL,
    DNV$BEARING BYTE EXTERNAL,
    DNV$TIME$TO$GO BYTE EXTERNAL;

63 1  DECLARE OFF$ON$TEXT(6) BYTE PUBLIC DATA('OFFON ');
64 1  DECLARE TEMP$BUF(24) BYTE PUBLIC;

/* BLANK LINE PLACED IN LOW CORE TO SAVE A FEW BYTES OF PROM */
65 1  DECLARE BLANK$LINE(24) BYTE PUBLIC AT(3)
    DATA(' ');

66 1  CLEAR$TEMP$BUF: PROCEDURE PUBLIC;
67 2  CALL MOVE(24,.BLANK$LINE,.TEMP$BUF);
68 2  END;
69 1  DECLARE PRI$ENABLE$LIST STRUCTURE(
    SM$MASK(7) BYTE,
    SM$VALUE(4) BYTE) DATA(
    80H,80H,80H,80H,0,10H,0,
    01H,23H,45H,66H);

70 1  DECLARE PRI$ICB STRUCTURE(
    MODE BYTE,
    NL$SCH BYTE,
    DEL$SCH BYTE,

```



```

      *EJECT
80  1  DECLARE CDU$STACK(40) ADDRESS,
      CDU$SPV ADDRESS,
      SYS$SPV ADDRESS;

81  1  INIT$CDU: PROCEDURE PUBLIC;

82  2      DISABLE;
83  2      CDU$STACK(LENGTH(CDU$STACK)-1) = .CDU$TOP;
84  2      CDU$SPV = .CDU$STACK(LENGTH(CDU$STACK)-1);
85  2      CALL INITIALIZE$CDU;
86  2      CALL INIT$VHF;
87  2      CALL INIT$UHF;
88  2      CALL INIT$ADF;
89  2      CALL INIT$CNV;
90  2      CALL INIT$IFF;
91  2      CALL INIT$DNV;
92  2      ENABLE;
93  2  END; /* OF INIT$CDU */

94  1  SUSPEND: PROCEDURE PUBLIC;

95  2      DISABLE;
96  2      CDU$SPV = STACKPTR;
97  2      STACKPTR = SYS$SPV;
98  2      ENABLE;
99  2  END;

100 1  RESTART: PROCEDURE PUBLIC;

101 2      DISABLE;
102 2      CDU$STACK(LENGTH(CDU$STACK) - 1) = .CDU$TOP;
103 2      CDU$SPV = .CDU$STACK(LENGTH(CDU$STACK)-1);
104 2      STACKPTR = SYS$SPV;
105 2      ENABLE;
106 2  END;
```

```
REJECT
107 1  ERROR: PROCEDURE(CODE) PUBLIC;
108 2  DECLARE CODE BYTE;

109 2  DECLARE STEP$ENABLE STRUCTURE(
      SW$MASK(7) BYTE,
      SW$VALUE(1) BYTE) CTA(
      0,0,0,0,4,0,0, /* CLEAR$SW */
      00H);
110 2  DECLARE ERROR$ICB STRUCTURE(
      MODE BYTE,
      NUM$CH BYTE,
      DEL$CH BYTE,
      DEL$MASK ADDRESS,
      ECHO$ROW BYTE,
      ECHO$COL BYTE,
      SW$ENABLE ADDRESS) DATA(SWITCH,0,0,0,0,0,0,STEP$ENABLE);
111 2  DECLARE MSG0(*) BYTE DATA(ROW2,'INVALID ENTRY',0),
      MSG1(*) BYTE DATA(ROW2,'NO UNALLOCATED TARGETS',0),
      MSG2(*) BYTE DATA(ROW2,'NO DATA FOR THIS CKPT',0);
112 2  DECLARE MSG$ADD(3) ADDRESS DATA(.MSG0,.MSG1,.MSG2);

113 2  CALL MOVE(24,.CURRENT$DISP(2),.TEMP$BUF); /*SAVE CURRENT DISPLAY ROW 2 */
114 2  CALL CLEAR$LINE(2);
115 2  CALL DISPLAY(MSG$ADD(CODE));
116 2  CALL READ(.ERROR$ICB);
117 2  CALL CLEAR$LINE(2);
118 2  CALL UPDATE$LINE(2,0,24,.TEMP$BUF); /* RESTORE DISPLAY ROW 2 */
119 2  END; /* OF ERROR */
```

```

SELECT
120 1  LIMIT$TEST: PROCEDURE(BUF$PTR,NUM$DIGITS,MIN$PTR,MAX$PTR) BYTE PUBLIC;
/* THIS PROCEDURE PERFORMS A LEXICOGRAPHICAL COMPARISON OF A TEXT
   STRING WITH TWO OTHER STRINGS TO DETERMINE IF THE UNKNOWN STRING
   IS WITHIN THE BOUNDS SPECIFIED BY THE OTHER TWO.
   THE DIGITS ARE EXAMINED FROM LEFT TO RIGHT.
   IT WAS WRITTEN TO COMPARE ASCII DIGIT STRINGS WITH MAX AND MIN STRINGS.
   A VALUE OF 1 IS RETURNED IF:
       MIN$STRING <= STRING <= MAX$STRING
   AND A VALUE OF 0 OTHERWISE. */
121 2  DECLARE NUM$DIGITS BYTE,
      (BUF$PTR,MIN$PTR,MAX$PTR) ADDRESS;
122 2  DECLARE (N,I) BYTE,
      (BUF BASED BUF$PTR)(1) BYTE,
      (MIN BASED MIN$PTR)(1) BYTE,
      (MAX BASED MAX$PTR)(1) BYTE;

123 2  IF (BUF(0) < MIN(0)) OR (BUF(0) > MAX(0)) THEN RETURN 0; /* FAIL */
125 2  IF NUM$DIGITS > 2 THEN N = NUM$DIGITS - 2;
127 2  ELSE N = 0;
128 2  DO I = 0 TO N;
129 3  IF MIN(I) > MAX(I) THEN DO;
131 4  IF (MIN(I) < BUF(I)) AND (BUF(I) < MAX(I)) THEN RETURN 1; /* OK */
133 4  IF BUF(I) = MIN(I) THEN DO;
135 5  IF BUF(I+1) < MIN(I+1) THEN RETURN 0; /* FAIL */
137 5  IF BUF(I+1) > MIN(I+1) THEN RETURN 1; /* OK */
139 5  END;
140 4  ELSE DO;
141 5  IF BUF(I) = MAX(I) THEN DO;
143 6  IF BUF(I+1) > MAX(I+1) THEN RETURN 0; /* FAIL */
145 6  IF BUF(I+1) < MAX(I+1) THEN RETURN 1; /* OK */
147 6  END;
148 5  END; /* OF ELSE DO */
149 4  END; /* OF IF MIN(I) > MAX(I) THEN DO */
150 3  ELSE DO;
151 4  IF BUF(I) > MIN(I) THEN RETURN 0; /* FAIL */
153 4  IF (BUF(I+1) < MIN(I+1)) OR (BUF(I+1) > MAX(I+1))
      THEN RETURN 0; /* FAIL */
155 4  END;
156 3  END; /* OF DO I */
157 2  RETURN 1; /* OK */

158 2  END; /* OF LIMIT$TEST */

```

```

      SELECT
159 1  DISPLAY%CDU%STATUS: PROCEDURE;

160 2  FORMAT%FREQ:PROCEDURE(CHAN,STATUS,FREQ%PTR,NDIGITS,ROW,COL,MASK);
161 3  DECLARE (CHAN,STATUS,NDIGITS,ROW,COL) BYTE;
      (FREQ%PTR,MASK) ADDRESS;

162 3  CALL CLEAR%TEMP%BUF;
163 3  IF (STATUS AND 20H) > 0 THEN TEMP%BUF(0) = '0';
165 3  CALL INSERT(NDIGITS,FREQ%PTR+CHAN*NDIGITS,TEMP%BUF(1),',',MASK);
166 3  CALL UPDATE%LINE(ROW,COL,NDIGITS+2,TEMP%BUF);
167 3  END;

168 2  CALL FORMAT%FREQ(VHF%ACTIVE%CHAN,VHF%STATUS,,VHF%FREQ,4,0,3,2000H);
169 2  CALL FORMAT%FREQ(UHF%ACTIVE%CHAN,UHF%STATUS,,UHF%FREQ,6,2,3,1000H);
170 2  CALL FORMAT%FREQ(ADF%ACTIVE%CHAN,ADF%STATUS,,ADF%FREQ,4,4,3,0);
171 2  CALL FORMAT%FREQ(CNV%ACTIVE%CHAN,CNV%STATUS,,CNV%FREQ,5,6,3,1000H);

/* DISPLAY IFF STATUS LINE */
172 2  CALL CLEAR%TEMP%BUF;
173 2  IF (IFF%STATUS AND 20H) > 0 THEN TEMP%BUF(0) = '0';
175 2  IF IFF%STATUS THEN TEMP%BUF(1) = '1';
177 2  IF (IFF%STATUS AND 2) > 0 THEN TEMP%BUF(2) = '2';
179 2  IF (IFF%STATUS AND 4) > 0 THEN DO;
181 3  TEMP%BUF(3) = '3';
182 3  TEMP%BUF(4) = 'A';
183 3  END;
184 2  IF (IFF%STATUS AND 10H) > 0 THEN TEMP%BUF(5) = 'C';
186 2  IF (IFF%STATUS AND 8) > 0 THEN DO;
188 3  TEMP%BUF(6) = '4';
189 3  IF IFF%MODE THEN TEMP%BUF(7) = 'B';
191 3  ELSE TEMP%BUF(7) = 'A';
192 3  END;
193 2  CALL UPDATE%LINE(6,15,8,TEMP%BUF);

/* DISPLAY CNV STATUS */
194 2  CALL DISPLAY%ACTIVE%WAYPT;
195 2  IF SHR(DNV%STATUS,4) THEN CALL UPDATE%LINE(0,15,1,('*'));
197 2  CALL INSERT(4,DNV%RANGE,TEMP%BUF,',',1000H);
198 2  CALL UPDATE%LINE(1,17,5,TEMP%BUF);
199 2  CALL UPDATE%LINE(2,19,3,DNV%BEARING);
200 2  CALL UPDATE%LINE(3,19,3,CNV%TIME%TO%GO);
201 2  CALL DISPLAY%PRESENT%POSITION(8);
202 2  END; /* OF DISPLAY%CDU%STATUS */

```

```

      $EJECT
203 1    CDU$ON$OFF: PROCEDURE:

204 2          IF SMV = OFF$SM THEN CALL SCREEN$INTENSITY(0);
206 2          ELSE CALL SCREEN$INTENSITY(50H);
207 2    END;

208 1    CDU: PROCEDURE PUBLIC:

209 2          DISABLE;
210 2          SYS$SPV = STACKPTR;
211 2          STACKPTR = CDU$SPV;
212 2          ENABLE;
213 2    END;

214 1    CDU$TOP: PROCEDURE:

      /* GENERATE PRIMARY TABLEAU */
215 2          CALL CLEAR(0);
216 2          CALL DISPLAY(,PRI$TABLEAU);
217 2          CALL DISPLAY(CDU$STATUS);
218 2          CALL READ(,PRI$(CB));
219 2          DO CASE SM$INDEX:
220 3              CALL WHF$SUBMODE;
221 3              CALL DMV$SUBMODE;
222 3              CALL UHF$SUBMODE;
223 3              CALL ADF$SUBMODE;
224 3              CALL CNV$SUBMODE;
225 3              CALL IFF$SUBMODE;
226 3              CALL CDU$ON$OFF;
227 3          END; /* OF DO CASE */

      /* TERMINATE CDU PROCESSING */
228 2          DISABLE;
229 2          CDU$STACK(LENGTH(CDU$STACK)-1) = .CDU$TOP;
230 2          CDU$SPV = .CDU$STACK(LENGTH(CDU$STACK)-1);
231 2          STACKPTR = SYS$SPV;
232 2          ENABLE;
233 2    END; /* OF CDU$TOP */

234 1    END; /* OF CDU: DO */

```

MODULE INFORMATION:

```

CODE AREA SIZE   = 0610H 15600
VARIABLE AREA SIZE = 007FH 1270
MAXIMUM STACK SIZE = 000EH 140
408 LINES READ
0 PROGRAM ERROR(S)

```

END OF PL/M-80 COMPILATION

ISIS-II PL/M-80 V3.0 COMPILATION OF MODULE IN
 OBJECT MODULE PLACED IN :F1:INPUT.OBJ
 COMPILER INVOKED BY: PLM80 :F1:INPUT.SRC DATE(21JUNE79) DEBUG

```

      $TITLE('INPUT')
1      IN: DO;
      $NOLIST INCLUDE(:F1:COLU.IT.SRC)
4      1      DECLARE PRIMARY$M LITERALLY '500';

5      1      SUSPEND: PROCEDURE EXTERNAL;
6      2      END;
7      1      RESTART: PROCEDURE EXTERNAL;
8      2      END;
9      1      UPDATE$SCREEN: PROCEDURE EXTERNAL;
10     2      END;
11     1      UPDATE$LINE: PROCEDURE(ROW,COL,NCHAR,TEXT$PTR) EXTERNAL;
12     2      DECLARE (ROW,COL,NCHAR) BYTE, TEXT$PTR ADDRESS;
13     2      END;

14     1      DECLARE ICB$PTR ADDRESS,
           ICB BASED ICB$PTR STRUCTURE(
           MODE BYTE, /* INPUT MODE */
           NUM$CH BYTE, /*NUMBER OF CHS TO BE INPUT */
           DELM$CH BYTE, /* DELIMETER CHARACTER */
           DELM$MASK ADDRESS, /* DELIMETER MASK */
           ECHO$ROW BYTE,
           ECHO$COL BYTE,
           SM$ENABLE$PTR ADDRESS);

           /* MISCELLANEOUS VARIABLES */
15     1      DECLARE
           IN$MODE BYTE,
           INPUT$ENABLE BYTE,
           FIELD$SIZE BYTE,
           DISP$LOC BYTE,
           DISP$MASK ADDRESS,
           IN$DELM$MASK ADDRESS,
           CH$COUNT BYTE,
           DISP$BUF(16) BYTE,
           CLEAR$STATE BYTE;

           /* GLOBAL VARIABLES */
16     1      DECLARE
           IN$BUF(16) BYTE PUBLIC,
           DATA$ENTERED BYTE PUBLIC,
           SM$INDEX BYTE PUBLIC,
           SMV BYTE PUBLIC;

           /* INPUT QUEUE VARIABLES */
17     1      DECLARE SM$QUEUE BYTE;

18     1      ADD$SMO: PROCEDURE(TEMP) PUBLIC;
           /* ADD NEW ENTRY TO INPUT QUEUE.
           THIS PROCEDURE IS CALLED FROM FOREGROUND.

```

```
IF THE QUEUE IS FULL THE ENTRY IS IGNORED. */
19 2  DECLARE TEMP BYTE;

/* NOTE: INPUT PROCEDURES MODIFIED TO REDUCE QUEUE LENGTH TO ONE ENTRY
THE INPUT QUEUE IS ENABLED ONLY AFTER THE BACKGROUND PROCESSING HAS
BEEN COMPLETED AND THE KEYBOARD IS POLLED. THIS PREVENTS SPURIOUS ENTRIES. */

20 2  IF (SW$QUEUE = OFFH) AND INPUT$ENABLE THEN SW$QUEUE = TEMP;

22 2  END; /* OF ADD$SWQ */

23 1  REMOVE$SWQ: PROCEDURE BYTE;
/* REMOVE ENTRY FROM INPUT QUEUE.
A VALUE OF OFFH IS RETURNED IF THE QUEUE IS EMPTY.
THIS PROCEDURE IS CALLED FROM THE BACKGROUND THEREFOR INTERRUPTS
MUST BE DISABLED WHEN MODIFYING THE QUEUE VARIABLES. */
24 2  DECLARE SWQ$TEMP BYTE;

25 2  DISABLE;
26 2  IF SW$QUEUE < OFFH THEN DO;
28 3  SWQ$TEMP = SW$QUEUE;
29 3  SW$QUEUE = OFFH;
30 3  ENABLE;
31 3  RETURN(SWQ$TEMP);
32 3  END;
33 2  ENABLE;
34 2  RETURN(OFFH);
35 2  END; /* OF REMOVE$SWQ */

36 1  INITIALIZE$SWQ: PROCEDURE PUBLIC;

37 2  DISABLE;
38 2  SW$QUEUE = OFFH;
39 2  ENABLE;
40 2  END;
```

```

$EJECT
41 1  CLEAR$INPUT$FIELD: PROCEDURE;
      /* CLEAR THE DISPLAY BUFFER, INSERT THE DELIMITER CHARACTER(S) AND
      COMPUTE THE SIZE OF THE INPUT FIELD (THE NUMBER OF DIGITS PLUS
      THE NUMBER OF DELIMITER CHARACTERS.) */
42 2  DECLARE (I,N) BYTE;

43 2      CH$COUNT = 0;
44 2      DISP$LOC = 0;
45 2      DISP$MASK = 8000H;
46 2      CLEAR$STATE = TRUE;
47 2      FIELD$SIZE = 0;

      /* THE TEST ON NUM$CH WAS ADDED TO ALLOW CLEAR$INPUT$FIELD TO
      BE INVOKED WHEN THE INPUT MODE IS EQUAL TO ZERO (SWITCH). */
48 2      IF ICB.NUM$CH > 0 THEN N = ICB.NUM$CH - 1;
50 2          ELSE N = 0;
51 2      DO I = 0 TO N;
52 3          IF (DISP$MASK AND IN$DELM$MASK) > 0 THEN DO;
54 4              DISP$BUF(FIELD$SIZE) = ICB.DELM$CH;
55 4              FIELD$SIZE = FIELD$SIZE + 1;
56 4          END;
57 3          DISP$MASK = SHR(DISP$MASK,1);
58 3          IN$BUF(I) = '0';
59 3          DISP$BUF(FIELD$SIZE) = '?';
60 3          FIELD$SIZE = FIELD$SIZE + 1;
61 3          END; /* OF DO I */
62 2      DISP$MASK = 8000H;

63 2      END; /* OF CLEAR$INPUT$FIELD */

```

```
SELECT
44 1 READ: PROCEDURE(ICB$ADD) PUBLIC:
65 2 DECLARE ICB$ADD ADDRESS:
    /* INPUT MODE CONTROL VARIABLE FORMAT:
       ICB.MODE AND OFH = 0 == SWITCH
                1 == DIGIT
                2 == OCTAL DIGIT
       IF BIT 7 = 1 THEN READ RETURNS AFTER ICB.NUM$CH DIGITS HAVE BEEN
       ENTERED (ENTER$CH NOT REQUIRED).
       IF BIT 6 = 1 THEN THE INPUT FIELD IS NOT CLEARED UNTIL AFTER THE
       FIRST CHARACTER HAS BEEN ENTERED.
    */

46 2 ICB$PTR = ICB$ADD:
47 2 IN$MODE = ICB.MODE:
48 2 DATA$ENTERED = FALSE:
49 2 IN$DEL$MASK = ICB.DEL$MASK:
70 2 CALL CLEAR$INPUT$FIELD:
71 2 IF (IN$MODE AND 40H) = 0 THEN
72 2     CALL UPDATE$LINE(ICB.ECHO$ROW,ICB.ECHO$COL,FIELD$SIZE,,DISP$BUF):
73 2 CALL UPDATE$SCREEN:
74 2 INPUT$ENABLE = TRUE: /* RESET BY INPUT PROCEDURES */
75 2 DO WHILE INPUT$ENABLE:
76 3     CALL SUSPEND: /* WAIT FOR INPUT */
77 3     END:

78 2 END: /* OF READ */
```

```
REJECT
79 1  PROCESS%SWITCH: PROCEDURE PUBLIC;
80 2  DECLARE ENABLE%LIST%PTR ADDRESS,
      ENABLE%LIST BASED ENABLE%LIST%PTR STRUCTURE(
        SW%MASK(7) BYTE,
        SW%VALUE(1) BYTE),
        MASK BYTE,
        (I,J,N,ROW,COL,CHAR) BYTE;

81 2  PROCESS%CH: PROCEDURE;

82 3  ADD%CH: PROCEDURE(CH);
83 4  DECLARE CH BYTE;
      /* ADD A NEW CHARACTER TO THE INPUT BUFFER AND THE DISPLAY BUFFER
      (AVOIDING DELIMITER CHARACTERS). */

84 4  CLEAR%STATE = TRUE;
85 4  IN%BUF(CH%COUNT) = CH;
86 4  CH%COUNT = CH%COUNT + 1;
87 4  IF (DISP%MASK AND IN%DELM%MASK) > 0 THEN DO;
88 5  DISP%LOC = DISP%LOC + 1;
89 5  END;
90 5  DISP%BUF(DISP%LOC) = CH;
91 4  DISP%LOC = DISP%LOC + 1;
92 4  DISP%MASK = SHR(DISP%MASK,1);
93 4  CALL UPDATE%LINE(1CB,ECHO%ROW,1CB,ECHO%COL,FIELD%SIZE,,DISP%BUF);
94 4

95 4  END; /* OF ADD%CH */
```

```

REJECT
96 3  REMOVE$CH: PROCEDURE;
    /* INVOKED IF CH = C(CLEAR).
    IF CLEAR$STATE = TRUE (THE INITIAL STATE) THE LAST CHARACTER ENTERED
    IS DELETED AND CLEAR$STATE SET FALSE.
    IF CLEAR$STATE = FALSE THEN THE ENTIRE INPUT FIELD IS DELETED. */

97 4      IF CH$COUNT > 0 THEN DO;
99 5          IF CLEAR$STATE THEN DO;
101 6              CLEAR$STATE = FALSE; /* CLEAR ONE CHARACTER */
102 6              CH$COUNT = CH$COUNT - 1;
103 6              DISP$LOC = DISP$LOC - 1;
104 6              DISP$BUF(DISP$LOC) = '?';
105 6              DISP$MASK = SHL(DISP$MASK,1);
106 6              IF (DISP$MASK AND ICB.DEL$MASK) > 0
                  THEN DISP$LOC = DISP$LOC - 1;
108 6              END; /* OF IF CLEAR$STATE */
109 5          ELSE DO;
110 6              CALL CLEAR$INPUT$FIELD; /* CLEAR THE ENTIRE FIELD */
111 6              CLEAR$STATE = TRUE;
112 6              END;
113 5              CALL UPDATE$LINE(ICB.ECHO$ROW,ICB.ECHO$COL,FIELD$SIZE,,DISP$BUF);
114 5              END;

115 4      END; /* OF REMOVE */

/* BEGIN PROCESS$CH CODE */

116 3      IF CHAR = CLEAR$SW THEN DO;
118 4          CALL REMOVE$CH;
119 4          CALL UPDATE$SCREEN;
120 4          RETURN;
121 4          END;
122 3      IF CHAR = ENTER$SW THEN DO;
124 4          INPUT$ENABLE = FALSE;
125 4          RETURN;
126 4          END;
127 3      DATA$ENTERED = TRUE; /* FIRST CHARACTER RECEIVED */
128 3      IF CH$COUNT < ICB.NUM$CH THEN DO;
130 4          CALL ADD$CH(CHAR); /* ADD THE NEW CHARACTER */
131 4          CALL UPDATE$SCREEN;
132 4          END;
133 3      IF ((IN$MODE AND 80H) > 0) AND (CH$COUNT = ICB.NUM$CH)
          THEN INPUT$ENABLE = FALSE;

135 3      END; /* OF PROCESS$CH */

```

```

REJECT
136 2  DECODE#DIGIT$SW: PROCEDURE BYTE;
      /* THIS PROCEDURE DECODES SWITCHES REPRESENTING A 9-DIGIT KEYBOARD.
      THE MAPPING IS:
      SWITCH VALUES  DIGITS
      15(8) - 17(8)   1 - 3
      25(8) - 27(8)   4 - 6
      35(8) - 37(8)   7 - 9
      46(8)           0
      45(8)           C
      47(8)           E
      */
137 3  DECLARE (R,C) BYTE;

138 3      C = SWV AND 7;
139 3      R = SHR(SWV,3) - 1;
140 3      IF (R < 4) AND (C > 4) THEN DO;
142 4          C = C - 4 + R*3;
143 4          IF C = 11 THEN C = 0;
145 4          RETURN (C + 30H);
146 4      END;
147 3      ELSE RETURN (OFFH);

148 3  END; /* OF DECODE#DIGIT$SW */

149 2  DECLARE CH BYTE;
150 2  DECODE#OCTAL$SW: PROCEDURE BYTE;

151 3      CH = DECODE#DIGIT$SW;
152 3      IF CH > '7' THEN RETURN (OFFH);
154 3          ELSE RETURN (CH);
155 3  END;

156 2  DECODE#INV$ALPHA$SW: PROCEDURE BYTE;

157 3      IF (SWV > 440) OR (SWV < 100) THEN RETURN(OFFH);
159 3      CH = SWV + 39H;
160 3      IF (CH = 'I') OR (CH = 'O') THEN RETURN(OFFH);
162 3      RETURN CH;
163 3  END; /* OF DECODE#ALPHA$SW */

164 2  DECODE#EAST$WEST$SW: PROCEDURE BYTE;

165 3      IF SWV = 250 THEN RETURN 'E';
167 3      IF SWV = 270 THEN RETURN 'W';
169 3      RETURN OFFH;
170 3  END; /* OF DECODE#EAST$WEST$SW */

171 2  DECODE#NORTH$SOUTH$SW: PROCEDURE BYTE;

172 3      IF SWV = 160 THEN RETURN 'N';
174 3      IF SWV = 360 THEN RETURN 'S';
176 3      RETURN OFFH;
177 3  END; /* OF DECODE#NORTH$SOUTH$SW */

```

REJECT

/* BEGIN PROCESS\$SWITCH CODE */

```

178 2      IF NOT INPUT$ENABLE THEN RETURN;
180 2      SWV = REMOVE$SW; /*GET NEXT CH FROM INPUT QUEUE */
181 2      IF SWV = OFFH THEN RETURN; /* INPUT QUEUE EMPTY */
183 2      IF IN$MODE = SWITCH THEN DO;
185 3          IF (SWV = CLEAR$SW) OR (SWV = ENTER$SW) THEN CHAR = SWV;
              ELSE
187 3              DO CASE (IN$MODE AND OFFH) - 1;
188 4                  CHAR = DECODE$DIGIT$SW;
189 4                  CHAR = DECODE$OCTAL$SW;
190 4                  CHAR = DECODE$NORTH$SOUTH$SW;
191 4                  CHAR = DECODE$EAST$WEST$SW;
192 4                  CHAR = DECODE$DIV$ALPHA$SW;
193 4                  END;
194 3              IF CHAR = OFFH THEN DO;
196 4                  CALL PROCESS$CH;
197 4                  RETURN;
198 4                  END;
199 3              END; /* OF IF IN$MODE ...DO; */

/* INPUT MODE IS NOT "DIGIT" OR
IS "DIGIT" AND DIGIT SWITCH WAS NOT PRESSED.
TEST FOR SPECIAL FUNCTION KEY. */
200 2      I = SWV - PRIMARY$SW;

201 2      IF I = 0 THEN CALL RESTART;

/* THE FOLLOWING CODE IS BYPASSED TO REDUCE MEMORY REQUIREMENTS
IF I < 6 THEN DO;
    DO CASE I;
        CALL RESTART;
        CALL IFF$EMER;
        CALL ZERO$CODE;
        GO TO A;
        GO TO A;
        GO TO A;
        CALL RPLY;
    END;
    RETURN;
A:
    END;
*/

/* INPUT IS NOT A SPECIAL FUNCTION SWITCH SO IT MUST BE AN
ENABLED SWITCH. DETERMINE RELATIVE INDEX. */
203 2      ENABLE$LIST$PTR = ICB,SW$ENABLE$PTR;
204 2      ROW = SHR(SWV,3);
205 2      COL = SWV AND 7;
206 2      IF COL > 0 THEN MASK = SHR(80H,COL);
208 2          ELSE MASK = 80H;
209 2      SW$INDEX = 0;
210 2      IF (MASK AND ENABLE$LIST,SW$MASK(ROW)) > 0 THEN DO;
/* DETERMINE THE INDEX VALUE OF THIS SWITCH BY COUNTING THE NUMBER OF
1'S IN THE MASK WORDS PRECEDING THE MASK FOR THIS ROW. */

```

```

212 3      IF ROW > 0 THEN DO:
214 4          N = ROW - 1:
215 4          DO I = 0 TO N:
216 5              MASK = ENABLE$LIST.SW$MASK(I):
217 5              DO WHILE MASK > 0:
218 6                  IF (MASK AND 80H) > 0 THEN SW$INDEX = SW$INDEX + 1:
220 6                  MASK = SHL(MASK,1):
221 6                  END:
222 5              END: /* OF DO I */
223 4          END: /* OF IF ROW > 0 */

/* COUNT THE NUMBER OF 1'S IN THE MASK WORD FOR THIS ROW */
224 3      MASK = ENABLE$LIST.SW$MASK(ROW):
225 3      IF COL > 0 THEN DO:
227 4          N = COL - 1:
228 4          DO I = 0 TO N:
229 5              IF (MASK AND 80H) > 0 THEN SW$INDEX = SW$INDEX + 1:
231 5              MASK = SHL(MASK,1):
232 5              END: /* OF DO I */
233 4          END: /* OF IF COL > 0 */

234 3      I = SW$INDEX/2: /*SW VALUE NIBBLES ARE PACKED 2 PER BYTE.*/
235 3      IF SW$INDEX THEN S$INDEX = ENABLE$LIST.SW$VALUE(I) AND 0FH:
237 3          ELSE SW$INDEX = SHR(ENABLE$LIST.SW$VALUE(I),4):
238 3      INPUT$ENABLE = FALSE:
239 3      END: /* OF IF (MASK AND ENABLE$LIST... */

240 2      END: /* OF PROCESS$SWITCH */

241 1      END: /* OF IN: DO */

```

MODULE INFORMATION:

```

CODE AREA SIZE   = 053BH 1339D
VARIABLE AREA SIZE = 0043H 67D
MAXIMUM STACK SIZE = 000AH 10D
419 LINES READ
0 PROGRAM ERROR(S)

```

END OF PL/M-80 COMPILATION

ISIS-II PL/M-80 V3.0 COMPILATION OF MODULE DISP
 OBJECT MODULE PLACED IN :F1:ABSDIS.OBJ
 COMPILER INVOKED BY: PLM80 :F1:ABSDIS.SRC DATE(21JUNE79) DEBUG

```

1      $TITLE('DISP')
      DISP: DO;
      /* CDU DISPLAY PROCEDURES */

2 1    DECLARE TRUE LITERALLY 'OFFH',
      FALSE LITERALLY '0';
3 1    DECLARE RM$BUFA(1) BYTE EXTERNAL,
      RM$BUFB(1) BYTE EXTERNAL;
4 1    DECLARE BLANK$LINE(24) BYTE EXTERNAL;
5 1    UPDATE$RM: PROCEDURE EXTERNAL;
6 2    END;
7 1    FILL$BUF: PROCEDURE(ROW,COL,BUF$ADD,REF$BUF$ADD,NCHAR,MODE) EXTERNAL;
8 2    DECLARE (ROW,COL,NCHAR,MODE) BYTE, (BUF$ADD,REF$BUF$ADD) ADDRESS;
9 2    END;

      /* THE FOLLOWING ARRAY CONTAINS THE ASCII CHARACTERS APPEARING ON THE
      NINE ROWS OF THE DISPLAY. */
10 1   DECLARE CURRENT$DISP(9) STRUCTURE(CHAR(24) BYTE) PUBLIC;

11 1   UPDATE$SCREEN: PROCEDURE PUBLIC;
12 2   CALL UPDATE$RM; /*INITIATE DMA TRANSFER TO REFRESH MEMORY */
13 2   END;

14 1   UPDATE$LINE: PROCEDURE(ROW,COL,NCHAR,TEXT$PTR) PUBLIC;
      /* THE INPUT MUST BE ASCII TEXT */

15 2   DECLARE ROW BYTE, /* ROW NUMBER */
      COL BYTE, /* STARTING COLUMN NUMBER */
      NCHAR BYTE, /* NUMBER OF CHS TO BE UPDATED */
      TEXT$PTR ADDRESS; /* TEXT STRING STARTING ADDRESS */
16 2   DECLARE CUR$PTR ADDRESS;
17 2   DECLARE (TEXT BASED TEXT$PTR)(1) BYTE,
      (CHAR BASED CUR$PTR)(1) BYTE,
      I BYTE;

18 2   CUR$PTR = .CURRENT$DISP(ROW).CHAR(COL);
19 2   CALL FILL$BUF(ROW,COL,TEXT$PTR,CUR$PTR,NCHAR,TRUE);
20 2   NCHAR = NCHAR - 1;
21 2   DO I = 0 TO NCHAR;
22 3   IF TEXT(I) <> ' ' THEN CHAR(I) = TEXT(I);
24 3   END;

25 2   END; /* OF UPDATE$LINE */

26 1   CLEAR: PROCEDURE(ROW) PUBLIC;

```

/* CLEARS BOTH THE CURRENT DISPLAY BUFFER AND THE REFRESH MEMORY
BUFFERS FROM THE SPECIFIED ROW TO THE LAST ROW (ROW 9).
NOTE: DNV WAYPOINT DATA OVERLAYS RM\$BUFB. */

```
27 2  DECLARE ROW BYTE;
28 2  DECLARE (K,N) ADDRESS;

29 2      IF ROW = 0 THEN K = 0;
31 2          ELSE K = 64 + DOUBLE(ROW)*160;
32 2      N = 1535 - K;
33 2      RM$BUFA(K) = 0;
34 2      CALL MOVE(N, RM$BUFA(K), RM$BUFA(K+1));
35 2      DO N = K TO 1535;
36 3          RM$BUFB(N) = RM$BUFB(N) AND OFOH;
37 3      END;
38 2      CURRENT$DISP(ROW), CHAR(0) = ' ';
39 2      CALL MOVE(215 - ROW*24, CURRENT$DISP(ROW),
                .CURRENT$DISP(ROW), CHAR(1));
40 2  END;

41 1  CLEAR$LINE: PROCEDURE(LINE$NUM) PUBLIC;
42 2  DECLARE LINE$NUM BYTE;

43 2      CALL FILL$BUF(LINE$NUM, 0, BLANK$LINE, CURRENT$DISP(LINE$NUM), 24, FALSE);
44 2      CALL MOVE(24, BLANK$LINE, CURRENT$DISP(LINE$NUM));
45 2  END;
```

```

      REJECT
46 1  DISPLAY: PROCEDURE(INPTR) PUBLIC;
47 2  DECLARE INPTR ADDRESS;
      /* UNPACK INPUT TEXT STRING AND FILL CURRENT DISPLAY BUFFER AND
      THE REFRESH MEMORY BUFFER.
      THE INPUT TEXT STRING IS ENCODED IN THE FOLLOWING FORMAT:
      1. BIT 7 = 0, BITS 0 - 6 IS AN ASCII CHARACTER.
      2. BIT 7 = 1, BITS 5,6: KEY, BITS 0 - 4: N
      KEY = 0 => ROW := N
      KEY = 1 => COL := N
      KEY = 2 => REPEAT THE NEXT CHARACTER N TIMES.
      KEY = 3 => NOT ASSIGNED.
      NOTE: THE FIRST CHARACTER IN ANY SEQUENCE MUST BE A ROW SPECIFIER.
      THE LAST CHARACTER IN ANY SEQUENCE MUST BE A ZERO TERMINATOR. */

48 2  DECLARE LINE(24) BYTE;
      (ROW,COL,IX,N,KEY,CH) BYTE;
      (IN BASED INPTR)(1) BYTE;

49 2  NEW$ROW: PROCEDURE;
50 3  DECLARE DEST$PTR ADDRESS;
      (DEST BASED DEST$PTR)(1) BYTE;
      I BYTE;

51 3  DEST$PTR = .CURRENT$DISP(ROW);
52 3  CALL FILL$BUF(ROW,0,LINE,DEST$PTR,24,TRUE);
53 3  DO I = 0 TO 23;
54 4  IF LINE(I) = ' ' THEN DO;
55 5  DEST(I) = LINE(I); /* STORE CH IN CURRENT DISP BUFFER */
56 5  LINE(I) = ' '; /* BLANK LINE FOR NEXT TIME AROUND */
57 5  END;
58 5  END;
59 4  END;
60 3  ROW = N;
61 3  COL = 0;
62 3  END;

63 2  REPEAT: PROCEDURE;
64 3  DECLARE I BYTE;

65 3  IF (N = 0) OR (N > 24) THEN RETURN;
67 3  IF N + COL > 23 THEN N = 23 - COL;
69 3  IX = IX + 1;
70 3  CH = IN(IX); /* GET THE CHARACTER TO BE REPEATED */
71 3  N = N - 1;
72 3  DO I = 0 TO N;
73 4  LINE(COL) = CH;
74 4  COL = COL + 1;
75 4  END;
76 3  END;

```

```

REJECT
/* BEGIN DISPLAY CODE */
77 2  LINE(0) = ' '; /* CLEAR THE LINE BUFFER */
78 2  CALL MOVE(23,LINE,LINE(1));
79 2  ROW = IN(0) AND 1FH;
80 2  CH = IN(1);
81 2  IX = 1;
82 2  COL = 0;
83 2  DO WHILE CH > 0;
84 3  IF (CH AND 80H) = 0 THEN DO;
86 4  LINE(COL) = CH; /* INSERT NEXT ASCII CHARACTER */
87 4  COL = COL + 1;
88 4  END;
89 3  ELSE DO;
90 4  N = CH AND 1FH; /* PROCESS CONTROL CHARACTER */
91 4  KEY = SHR(CH,5) AND 3;
92 4  DO CASE KEY;
93 5  CALL NEWROW; /*ROW SPECIFIER ENCOUNTERED */
94 5  COL = N; /*COLUMN SPECIFIER ENCOUNTERED */
95 5  CALL REPEAT; /*REPEAT SPECIFIER ENCOUNTERED */
96 5  1 /* NOT ASSIGNED */
97 5  END; /* OF DO CASE */
98 4  END; /* OF ELSE DO */
99 3  IX = IX + 1;
100 3  CH = IN(IX);
101 3  END; /* OF DO WHILE */
102 2  CALL NEWROW; /* PROCESS LAST ROW */

103 2  END; /* OF DISPLAY */

```

```

      @EJECT
104 1  INSERT: PROCEDURE(NCHAR, SOURCE$PTR, DEST$PTR, DEL$MCH, DEL$MASK)
      PUBLIC;
      /* COPY CHARACTERS FROM A SOURCE BUFFER TO A DESTINATION BUFFER AND
      INSERT A DELETER CHARACTER UNDER CONTROL OF A MASK. THE PRIMARY
      USE OF THIS PROCEDURE IS IN FORMATTING DIGIT STRINGS FOR PRESENTATION
      ON THE DISPLAY. NOTE: THE MASK CAN SPAN ONLY 16 BITS BUT THE NUMBER
      OF CHARACTERS THAT MAY BE MOVED IS LIMITED ONLY BY THE SIZE OF
      THE BYTE VARIABLES INVOLVED. */
105 2  DECLARE NCHAR BYTE, /*NUMBER OF CHARACTERS IN THE SOURCE BUFFER*/
      SOURCE$PTR ADDRESS,
      DEST$PTR ADDRESS,
      DEL$MCH BYTE,
      DEL$MASK ADDRESS;
106 2  DECLARE (SOURCE BASED SOURCE$PTR)(1) BYTE,
      (DEST BASED DEST$PTR)(1) BYTE,
      (I,LOC) BYTE;

107 2      LOC = 0;
108 2      I = 0;
109 2      DO WHILE I < NCHAR;
110 3          IF (DEL$MASK AND 8000H) > 0 THEN DEST(LOC) = DEL$MCH;
112 3          ELSE DO;
113 4              DEST(LOC) = SOURCE(I);
114 4              I = I + 1;
115 4          END;
116 3          DEL$MASK = SHL(DEL$MASK,1);
117 3          LOC = LOC + 1;
118 3          END;
119 2      END; /* OF INSERT */

120 1      END; /* OF DISP: DO */

```

MODULE INFORMATION:

```

CODE AREA SIZE   = 03DAH   986D
VARIABLE AREA SIZE = 0114H   276D
MAXIMUM STACK SIZE = 000CH   12D
191 LINES READ
0 PROGRAM ERROR(S)

```

END OF PL/M-80 COMPILATION

ISIS-II PL/M-80 V3.1 COMPILATION OF MODULE COM
 OBJECT MODULE PLACED IN :F1:COM.OBJ
 COMPILER INVOKED BY: PLM80 :F1:COM.SRC DATE(30OCT79) DEBUG

```

1      $TITLE('COM')
      COM: DO:
      $NOLIST INCLUDE(:F1:CDULIT.SRC)
4 1    ERROR: PROCEDURE(CODE) EXTERNAL;
5 2    DECLARE CODE BYTE;
6 2    END;
7 1    CLEAR: PROCEDURE(START$ROW) EXTERNAL;
8 2    DECLARE START$ROW BYTE;
9 2    END;
10 1   GRID: PROCEDURE(POINTER) EXTERNAL;
11 2   DECLARE POINTER ADDRESS;
12 2   END;
13 1   DISPLAY: PROCEDURE(IN$PTR) EXTERNAL;
14 2   DECLARE IN$PTR ADDRESS;
15 2   END;
16 1   READ: PROCEDURE(ICB$PTR) EXTERNAL;
17 2   DECLARE ICB$PTR ADDRESS;
18 2   END;
19 1   UPDATE$LINE: PROCEDURE(ROW,COL,NCHAR,TEXT$PTR) EXTERNAL;
20 2   DECLARE (ROW,COL,NCHAR) BYTE, TEXT$PTR ADDRESS;
21 2   END;
22 1   CLEAR$TEMP$BUF: PROCEDURE EXTERNAL;
23 2   END;
24 1   INSERT: PROCEDURE(NCHAR,SOURCE$PTR,DEST$PTR,DELM$CH,DELM$MASK) EXTERNAL;
25 2   DECLARE (NCHAR,DELM$CH) BYTE, (SOURCE$PTR,DEST$PTR,DELM$MASK) ADDRESS;
26 2   END;
27 1   UPDATE$SCREEN: PROCEDURE EXTERNAL;
28 2   END;
29 1   DISPLAY$DIGIT$KB: PROCEDURE EXTERNAL;
30 2   END;
31 1   LIMIT$TEST: PROCEDURE(BUF$PTR,NUM$DIGITS,MIN$PTR,MAX$PTR) BYTE EXTERNAL;
32 2   DECLARE (BUF$PTR,MIN$PTR,MAX$PTR) ADDRESS,
      NUM$DIGITS BYTE;
33 2   END;

      /* LOCAL VARIABLES WHOSE VALUES ARE OBTAINED FROM THE CURRENT
      CONTROL BLOCK (CNTRL) OR COMPUTED FROM VALUES THEREIN. */
34 1   DECLARE ACTIVE$CHAN BYTE,
      LAST$ACTIVE$CHAN BYTE,
      FREQ$BASE$ADD ADDRESS,
      FREQ$SIZE BYTE,
      FREQ$SIZE$PLUS$ONE BYTE,
      INSERT$MASK ADDRESS,
      FREQ$MIN$ADD ADDRESS,
      FREQ$MAX$ADD ADDRESS,
      MIN$FREQ$ICB$ADD ADDRESS;

      /* EXTERNAL VARIABLES */
35 1   DECLARE
      BLANK$LINE(24) BYTE EXTERNAL,
      IN$BUF(16) BYTE EXTERNAL,

```

```

DATA$ENTERED BYTE EXTERNAL,
SMV BYTE EXTERNAL,
TEMP$BUF(24) BYTE EXTERNAL,
OFF$ON$TEXT(6) BYTE EXTERNAL,
SW$INDEX BYTE EXTERNAL;

```

```

36 1 DECLARE CNTL$ADD ADDRESS;
37 1 DECLARE CNTL BASED CNTL$ADD STRUCTURE(
    LABEL$PTR ADDRESS,
    ACTIVE$CHAN$PTR ADDRESS,
    LAST$ACTIVE$CHAN$PTR ADDRESS,
    FREQ$PTR ADDRESS,
    SIZE BYTE,
    MASK ADDRESS,
    MIN$PTR ADDRESS,
    MAX$PTR ADDRESS,
    MAN$FREQ$ICB$PTR ADDRESS);
38 1 DECLARE SUBMODE$LAST$ENABLE STRUCTURE(
    SM$MASK(7) BYTE,
    SM$VALUE(1) BYTE) PUBLIC DATA(0,0,0,0,0,90H,0);
39 1 DECLARE SUBMODE$LAST$STEP$ENABLE STRUCTURE(
    SM$MASK(7) BYTE,
    SM$VALUE(1) BYTE) PUBLIC DATA(0,0,0,0,0,98H,01H);
40 1 DECLARE DIGIT$SEL$ICB STRUCTURE(
    MODE BYTE,
    NUM$CH BYTE,
    DEL$MCH BYTE,
    DEL$M$MASK ADDRESS,
    ECHO$ROW BYTE,
    ECHO$COL BYTE,
    SM$ENABLE ADDRESS)
    DATA(NOCLEAR$CNTL$DIGIT,1,0,0,2,3,.SUBMODE$LAST$STEP$ENABLE);
41 1 DECLARE PAGE0$SM$ENABLE STRUCTURE(
    SM$MASK(7) BYTE,
    SM$VALUE(11) BYTE) DATA(
    0,0E0H,0E0H,0E0H,0,0,98H,
    00H,01H,11H,22H,23H,33H,44H,45H,55H,67H,80H);
42 1 DECLARE PAGE1$SM$ENABLE STRUCTURE(
    SM$MASK(7) BYTE,
    SM$VALUE(8) BYTE) DATA(
    0,0E0H,0E0H,0E0H,0,0,98H,
    00H,01H,11H,22H,23H,33H,45H,60H);
/* THIS IS INCLUDED TO INSURE INITIALIZATION OF RAM STORAGE OF
STAT$PAGE$ICB. */
43 1 DECLARE INITIAL$STAT$PAGE$ICB STRUCTURE(
    MODE BYTE,
    NUM$CH BYTE,
    DEL$MCH BYTE,
    DEL$M$MASK ADDRESS,
    ECHO$ROW BYTE,
    ECHO$COL BYTE,
    SM$ENABLE ADDRESS) DATA(SWITCH,0,0,0,0,0,.PAGE0$SM$ENABLE);
44 1 DECLARE STAT$PAGE$ICB STRUCTURE(
    MODE BYTE,
    NUM$CH BYTE,
    DEL$MCH BYTE,
    DEL$M$MASK ADDRESS,

```

PL/M-80 COMPILER COM

ECHOSRON BYTE,
ECHOSCOL BYTE,
SUSPENSIBLE ADDRESS);

```

SELECT
45 1  INITIALIZE%COM: PROCEDURE(CNTL%PTR) PUBLIC:
    /* THIS PROCEDURE MUST BE INVOKED PRIOR TO ANY OF THE FOLLOWING
       PROCEDURES. */
46 2  DECLARE CNTL%PTR ADDRESS:

47 2      CALL MOVE(9, INITIAL$STAT$PAGE$ICB, STAT$PAGE$ICB);
48 2      CNTL$ADD = CNTL%PTR;
49 2      CALL MOVE(1, CNTL$ACTIVE$CHAN$PTR, ACTIVE$CHAN);
50 2      CALL MOVE(1, CNTL$LAST$ACTIVE$CHAN$PTR, LAST$ACTIVE$CHAN);
51 2      FREQ$BASE$ADD = CNTL$FREQ$PTR;
52 2      FREQ$SIZE = CNTL$SIZE;
53 2      FREQ$SIZE$PLUSONE = FREQ$SIZE + 1;
54 2      INSERT$MASK = CNTL$MASK;
55 2      FREQ$MIN$ADD = CNTL$MIN$PTR;
56 2      FREQ$MAX$ADD = CNTL$MAX$PTR;
57 2      NAN$FREQ$ICB$ADD = CNTL$NAN$FREQ$ICB$PTR;

58 2  END;

59 1  FREQ$ADD: PROCEDURE(CHAN) ADDRESS:
60 2  DECLARE CHAN BYTE;
    /* COMPUTE OFFSET INTO FREQ ARRAY. */

61 2      RETURN(FREQ$BASE$ADD + CHAN * FREQ$SIZE);
62 2  END;

63 1  DISPLAY$CHAN$FREQ: PROCEDURE(CHAN$IX) PUBLIC:
64 2  DECLARE CHAN$IX BYTE;

65 2      CALL CLEAR$TEMP$BUF;
66 2      CALL MOVE(7, ('CH- F-'), TEMP$BUF);
67 2      CALL INSERT(FREQ$SIZE, FREQ$ADD(CHAN$IX), TEMP$BUF(7), '.', INSERT$MASK);
68 2      TEMP$BUF(3) = CHAN$IX + 30H;
69 2      CALL UPDATE$LINE(2, 0, 15, TEMP$BUF);

70 2  END; /* OF DISPLAY$ACTIVE$CHAN */

71 1  UPDATE$FREQ: PROCEDURE(CHAN$IX) BYTE:
72 2  DECLARE CHAN$IX BYTE;

73 2      IF LIMIT$TEST(1, IN$BUF, FREQ$SIZE, FREQ$MIN$ADD, FREQ$MAX$ADD)
        THEN DO:
75 3          CALL MOVE(FREQ$SIZE, IN$BUF, FREQ$ADD(CHAN$IX));
76 3          RETURN TRUE;
77 3          END;
78 2      ELSE DO:
79 3          CALL ERROR(0);
80 3          RETURN FALSE;
81 3          END;

82 2  END; /* OF UPDATE$FREQ */

83 1  LAST$FREQ: PROCEDURE:

```

84 2 ACTIVECHAN = LAST\$ACTIVECHAN;
 85 2 CALL MOVE(1, LAST\$ACTIVECHAN, CNTL, ACTIVECHAN\$PTR);
 86 2 CALL DISPLAY\$CHAN\$FREQ(ACTIVECHAN);
 87 2 END; /* OF LAST\$FREQ */

LAST
CALL MOVE(1, LAST\$ACTIVECHAN, CNTL, ACTIVECHAN\$PTR);
CALL DISPLAY\$CHAN\$FREQ(ACTIVECHAN);

```
SELECT
88 1  CHANSEL: PROCEDURE PUBLIC;
89 2  DECLARE I BYTE;
90 2  DECLARE CHANSEL$TABLEAU(*) BYTE DATA(
      ROW2,'CH-?',
      ROW3,COL1,'CHAN',
      ROW4,COL1,'SEL',0);

91 2  CALL CLEAR(2);
92 2  CALL DISPLAY$DIGIT$KB;
93 2  CALL DISPLAY(.CHANSEL$TABLEAU);
94 2  I = OFFH;
95 2  DO FOREVER;
96 3  CALL READ(.DIGIT$SEL$ICB);
97 3  IF DATA$ENTERED THEN DO;
98 4  I = IN$BUF(0) - 30H;
100 4  CALL DISPLAY$CHAN$FREQ(I);
101 4  END;
102 3  ELSE IF (SMV = ENTER$SW) AND (I < OFFH) THEN DO;
104 4  LAST$ACTIVE$CHAN = ACTIVE$CHAN;
105 4  CALL MOVE(1..LAST$ACTIVE$CHAN,CNTL..LAST$ACTIVE$CHAN$PTR);
106 4  ACTIVE$CHAN = I;
107 4  CALL MOVE(1..ACTIVE$CHAN,CNTL..ACTIVE$CHAN$PTR);
108 4  RETURN;
109 4  END;
      IF SMV = SUBMODE$SW THEN RETURN;
      IF SMV = CLEAR$SW THEN DO;
112 3  CALL UPDATE$LINE(2,7,FREQ$SIZE$PLUS$ONE,.BLANK$LINE);
114 4  I = OFFH;
115 4  END;
116 4  END; /* OF DO FOREVER */
117 3  END; /* OF CHANSEL */
118 2
```

```

      $EJECT
119 1  MAN$FREQ: PROCEDURE(CHAN) PUBLIC;
120 2  DECLARE CHAN BYTE;
121 2  DECLARE MAN$FREQ$TABLEAU(*) BYTE DATA(
      ROW3,COL6,'MAN',
      ROW4,COL6,'FREQ',0);

122 2  CALL CLEAR(2);
123 2  CALL DISPLAY$DIGIT$KB;
124 2  CALL DISPLAY(.MAN$FREQ$TABLEAU);
125 2  CALL DISPLAY$CHAN$FREQ(CHAN);
126 2  DO FOREVER;
127 3  CALL READ(MAN$FREQ$ICB$ADD);
128 3  IF DATA$ENTERED AND (SWV = ENTER$SW) THEN DO;
130 4  IF UPDATE$FREQ(CHAN) THEN RETURN;
132 4  END;
133 3  ELSE IF SWV = LAST$SW THEN CALL LAST$FREQ;
      IF SWV = SUBMODE$SW THEN RETURN;
137 3  END;

138 2  END; /* OF MAN$FREQ */

139 1  PRST$CHAN: PROCEDURE PUBLIC;
140 2  DECLARE PRST$CHAN$TABLEAU(*) BYTE DATA(
      ROW2,'CH-?',COL5,'F-',
      ROW3,COL11,'PRST',
      ROW4,COL11,'CHAN',0);
141 2  DECLARE I BYTE;

142 2  CALL CLEAR(2);
143 2  CALL DISPLAY$DIGIT$KB;
144 2  CALL DISPLAY(.PRST$CHAN$TABLEAU);
145 2  CALL READ(.DIGIT$SEL$ICB);
146 2  IF DATA$ENTERED THEN DO;
148 3  I = IN$BUF(0) - 30H;
149 3  DO WHILE I < 10;
150 4  CALL DISPLAY$CHAN$FREQ(I);
151 4  CALL READ(MAN$FREQ$ICB$ADD);
152 4  IF DATA$ENTERED AND (SWV = ENTER$SW) THEN DO;
154 5  IF UPDATE$FREQ(I) THEN I = I + 1;
156 5  END;
157 4  ELSE DO;
158 5  IF SWV = SUBMODE$SW THEN RETURN;
160 5  IF SWV = LAST$SW THEN CALL LAST$FREQ;
162 5  IF SWV = STEP$SW THEN I = I + 1;
164 5  END;
165 4  END; /* OF DO WHILE */
166 3  END; /* OF DATA$ENTERED */
167 2  END; /* OF PRST$CHAN */

```

```

SELECT
168 1  STAT$PAGE: PROCEDURE PUBLIC;

169 2  GENERATE$STAT$HEADER: PROCEDURE(PAGE);
170 3  DECLARE PAGE BYTE;

171 3      CALL CLEAR(0);
172 3      CALL CLEAR$TEMP$BUF;
173 3      CALL MOVE(3,CNTL.LABEL$PTR,,TEMP$BUF);
174 3      CALL MOVE(16,('CHAN STAT /2'),,TEMP$BUF(4));
175 3      TEMP$BUF(17) = PAGE + 31H;
176 3      CALL UPDATE$LINE(0,4,20,,TEMP$BUF);
177 3  END: /* OF GENERATE$STAT$HEADER */


178 2  PROCESS$PAGE: PROCEDURE(PAGE$NUM,SW$ENABLE$PTR);
179 3  DECLARE PAGE$NUM BYTE;
      I BYTE;
      SW$ENABLE$PTR ADDRESS;
180 3  DECLARE PAGE$MAP(2) STRUCTURE(LIST(6) BYTE) DATA(
      0,3,1,4,2,5, /* PAGE 1 */
      6,9,7,0FFH,8,0FFH); /* PAGE 2 */
181 3  DECLARE SW$IX$MAP(2) STRUCTURE(LIST(6) BYTE) DATA(
      0,3,1,4,2,5, /* PAGE 1 */
      6,9,7,8,0,0); /* PAGE 2 */
182 3  DECLARE MAX$SW$IX(2) BYTE DATA(6,4);


183 3  DISPLAY$STAT$LINE: PROCEDURE(LINE$NUM);
184 4  DECLARE LINE$NUM BYTE;
185 4  DECLARE CHAN BYTE;

186 4      CALL CLEAR$TEMP$BUF;
      /* FORMAT LEFT HALF OF LINE */
187 4      CHAN = PAGE$MAP(PAGE$NUM).LIST(LINE$NUM*2);
188 4      TEMP$BUF(0) = CHAN + 30H;
189 4      IF CHAN = ACTIVE$CHAN THEN TEMP$BUF(1) = '*';
191 4      CALL INSERT(FREQ$SIZE,FREQ$ADD(CHAN),,TEMP$BUF(2),',',INSERT$MASK);
      /* FORMAT RIGHT HALF OF LINE */
192 4      CHAN = PAGE$MAP(PAGE$NUM).LIST(LINE$NUM*2 + 1);
193 4      IF CHAN < 0FFH THEN DO;
195 5          TEMP$BUF(11) = CHAN + 30H;
196 5          IF CHAN = ACTIVE$CHAN THEN TEMP$BUF(12) = '*';
198 5          CALL INSERT(FREQ$SIZE,FREQ$ADD(CHAN),,TEMP$BUF(13),',',INSERT$MASK);
199 5          END;
200 4      CALL UPDATE$LINE(LINE$NUM*2+2,1,21,,TEMP$BUF);
201 4  END: /* OF DISPLAY$STAT$LINE */


/* BEGIN PROCESS$PAGE CODE.
PAGE$NUM AND SW$ENABLE$PTR ARE PASSED AS ARGUMENTS. */

202 3  STAT$PAGE$ICB.SW$ENABLE = SW$ENABLE$PTR;
203 3  DO FOREVER;
204 4      CALL GENERATE$STAT$HEADER(PAGE$NUM);
205 4      DO I = 0 TO 2;
206 5          CALL DISPLAY$STAT$LINE(I);

```

```

207 5      END;
208 4      CALL READ(.STAT$PAGE$ICB);
209 4      IF SW$INDEX < MAX$SW$IX(PAGE$NUM) THEN DO;
211 5          CALL MAN$FREQ(SW$IX$MAP(PAGE$NUM),LIST(SW$INDEX));
212 5      END;
213 4      ELSE IF SWV = SUBMODE$SW THEN RETURN;
217 4      IF SWV = STEP$SW THEN RETURN;
219 4      IF SWV = LAST$SW THEN CALL LAST$FREQ;
220 3      END; /* OF DO FOREVER */

```

< IF SWV = LIST\$SW THEN DO;
 CALL LAST\$FREQ;
 CALL SWAP(.VHF\$ACTIVE\$CHAN,
 .LAST\$VHF\$ACTIVE\$CHAN);

```

/* BEGIN STAT$PAGE CODE */
221 2      DO FOREVER;
222 3          CALL PROCESS$PAGE(0,.,PAGE$SW$ENABLE);
223 3          IF SWV = SUBMODE$SW THEN RETURN;
225 3          CALL PROCESS$PAGE(1,.,PAGE1$SW$ENABLE);
226 3          IF SWV = SUBMODE$SW THEN RETURN;
228 3      END;
229 2      END; /* OF STAT$PAGE */

```

/x LAST x/

```

230 1      SWAP: PROCEDURE(A$PTR, B$PTR) PUBLIC;
231 2      DECLARE (A$PTR, B$PTR) ADDRESS;
232 2      DECLARE A BASED A$PTR BYTE,
          B BASED B$PTR BYTE,
          T BYTE;

```

```

233 2      T = A;
234 2      A = B;
235 2      B = T;
236 2      END; /* OF SWAP */
237 1      END; /* OF COM */

```

MODULE INFORMATION:

```

CODE AREA SIZE   = 0577H 1399D
VARIABLE AREA SIZE = 0021H 45D
MAXIMUM STACK SIZE = 0010H 16D
409 LINES READ
0 PROGRAM ERROR(S)

```

END OF PL/M-80 COMPILATION


```

38 1  DECLARE I BYTE;
39 1  DECLARE MIN$FREQ(4) BYTE DATA('3000');
      MAX$FREQ(4) BYTE DATA('7595');

40 1  INIT$VHF: PROCEDURE PUBLIC;

41 2      DO I = 0 TO 9;
42 3          CALL MOVE(4,MIN$FREQ,VHF$FREQ(I));
43 3      END;
44 2      VHF$ACTIVE$CHAN = 0;
45 2      LAST$VHF$ACTIVE$CHAN = 0;
46 2      VHF$STATUS = 0;
47 2  END;

48 1  VHF$SUBMODE: PROCEDURE PUBLIC;

49 2  DECLARE MAN$FREQ$ICB STRUCTURE(
      MODE BYTE,
      NUM$CH BYTE,
      DELM$CH BYTE,
      DELM$MASK ADDRESS,
      ECHO$RON BYTE,
      ECHO$COL BYTE,
      SM$ENABLE ADDRESS)
      DATA(NOCLEAR$DIGIT,4,'.',2000H,2,7,SUBMODE$LAST$STEP$ENABLE);

50 2  DISPLAY$STATUS: PROCEDURE;
51 3  DECLARE MODE$TEXT(7) STRUCTURE(CHS(6) BYTE) DATA(
      ' ', 'T/R', 'T/R+G', 'HOM', 'RETRAN', ' ', 'FAIL ');

52 3      I = SHR(VHF$STATUS,5) AND 1;
53 3      CALL UPDAT$LINE(1,0,3,OFF$ON$TEXT(3*1));
54 3      I = VHF$STATUS AND OFH;
55 3      IF I = 5 THEN I = I + (SHR(VHF$STATUS,4) AND 1);
56 3      CALL UPDAT$LINE(1,5,6,MODE$TEXT(I));
57 3  END; /* OF DISPLAY$STATUS */

58 2  UPDATE$MODE: PROCEDURE;
59 3  DECLARE SAM$LIST(6) BYTE DATA(30Q,32Q,34Q,35Q,36Q,44Q);

60 3      I = 0;
61 3      DO WHILE (SAM < SAM$LIST(I)) AND (I < 6);
62 4          I = I + 1;
63 4      END;
64 3      IF I < 6 THEN I = I + 1;
65 3      VHF$STATUS = (VHF$STATUS AND OFH) + I;
66 3  END;

67 2  ONOFF: PROCEDURE;

68 3      IF SAM < OFF$M
        THEN VHF$STATUS = VHF$STATUS AND OFFH; /* OFF */
        ELSE VHF$STATUS = VHF$STATUS OR 3WH; /* ON */
69 3      CALL DISPLAY$STATUS;
70 3  END;

```

```

$EJECT
75 2  DECLARE VHF$SUBMODE$TABLEAU(*) BYTE DATA(
      ROW0,'FM ',
      ROW3,'CHAN MAN PRST STAT',
      ROW4,'SEL',COL6,'FREQ',COL11,'CHAN PAGE',
      ROW5,'T/R',COL6,'T/R+ HOM',COL17,'RE',
      ROW6,COL6,'GARD',COL16,'TRAN',
      ROW7,COL11,'TEST',0);
76 2  DECLARE VHF$SW$ENABLE STRUCTURE(
      SW$MASK(7) BYTE,
      SW$VALUE(8) BYTE) DATA(
      0,0,0EEH,0AEH,8,18H,10H, /* MASK DATA */
      00H,12H,33H,44H,44H,55H,60H); /* VALUE DATA */
77 2  DECLARE VHF$ICB STRUCTURE(
      MODE BYTE,
      NUM$CH BYTE,
      DELM$CH BYTE,
      DELM$MASK ADDRESS,
      ECHO$ROW BYTE,
      ECHO$COL BYTE,
      SW$ENABLE ADDRESS) DATA(
      SWITCH,0,0,0,0,0,.,VHF$SW$ENABLE);

78 2  DECLARE VHF$LABEL(3) BYTE AT(.,VHF$SUBMODE$TABLEAU(1));
79 2  DECLARE VHF$CNTL STRUCTURE(
      LABEL$PTR ADDRESS,
      ACTIVE$CHAN$PTR ADDRESS,
      LAST$ACTIVE$CHAN$PTR ADDRESS,
      FREQ$PTR ADDRESS,
      FREQ$SIZE BYTE,
      INSERT$MASK ADDRESS,
      MIN$PTR ADDRESS,
      MAX$PTR ADDRESS,
      MAN$FREQ$ICB$PTR ADDRESS) DATA(
      .VHF$LABEL,.,VHF$ACTIVE$CHAN,.,LAST$VHF$ACTIVE$CHAN,.,VHF$FREQ,4,2000H,
      .MIN$FREQ,.,MAX$FREQ,.,MAN$FREQ$ICB);

```

```

      SELECT
80  2      CALL INITIALIZE$COM(.VHF$CNTL);
81  2      DO FOREVER;
82  3          CALL CLEAR(0);
83  3          CALL GRID(.SUBMODE$GRID);
84  3          CALL DISPLAY(.VHF$SEQUENCE$TABLEAU);
85  3          CALL DISPLAY$CHAN$FREQ(VHF$ACTIVE$CHAN);
86  3          CALL DISPLAY$STATUS;
87  3          CALL READ(.VHF$ICB);
88  3          DO CASE SW$INDEX;
89  4              CALL CHAN$SEL;
90  4              CALL MAN$FREQ(VHF$ACTIVE$CHAN);
91  4              CALL PRST$CHAN;
92  4              CALL STAT$PAGE;
93  4              CALL UPDATE$MODE;
94  4              CALL ON$OFF;
95  4              CALL SWAP(.VHF$ACTIVE$CHAN, .LAST$VHF$ACTIVE$CHAN); /* LAST */
96  4              END; /* OF DO CASE */
97  3          END; /* OF DO FOREVER */

98  2      END; /* OF VHF$SUBMODE */

99  1      END; /* OF VHF */
```

MODULE INFORMATION:

```
CODE AREA SIZE   = 0247H   5830
VARIABLE AREA SIZE = 0001H   10
MAXIMUM STACK SIZE = 000AH  100
224 LINES READ
0 PROGRAM ERROR(S)
```

END OF PL/M-80 COMPILATION

ISIS-II PL/M-80 V3.1 COMPILATION OF MODULE UNF
 OBJECT MODULE PLACED IN :F1:UNF.OBJ
 COMPILER INVOKED BY: PLM80 :F1:UNF.SRC DATE (30OCT79) DEBUG

```

1      $TITLE('UNF')
      UNF: DO:
      $NOLIST INCLUDE(:F1:CDULIT.SRC)
4 1    INITIALIZE$COM: PROCEDURE(CNTL$PTR) EXTERNAL:
5 2    DECLARE CNTL$PTR ADDRESS:
6 2    END:
7 1    CHAN$SEL: PROCEDURE EXTERNAL:
8 2    END:
9 1    MAN$FREQ: PROCEDURE(CHAN) EXTERNAL:
10 2   DECLARE CHAN BYTE:
11 2   END:
12 1   PRST$CHAN: PROCEDURE EXTERNAL:
13 2   END:
14 1   STAT$PAGE: PROCEDURE EXTERNAL:
15 2   END:
16 1   DISPLAY$CHAN$FREQ: PROCEDURE(CHAN) EXTERNAL:
17 2   DECLARE CHAN BYTE:
18 2   END:
19 1   CLEAR: PROCEDURE(START$ROW) EXTERNAL:
20 2   DECLARE START$ROW BYTE:
21 2   END:
22 1   GRID: PROCEDURE(POINTER) EXTERNAL:
23 2   DECLARE POINTER ADDRESS:
24 2   END:
25 1   DISPLAY: PROCEDURE(IN$PTR) EXTERNAL:
26 2   DECLARE IN$PTR ADDRESS:
27 2   END:
28 1   READ: PROCEDURE(ICB$PTR) EXTERNAL:
29 2   DECLARE ICB$PTR ADDRESS:
30 2   END:
31 1   UPDATE$LINE: PROCEDURE(ROW,COL,NCHAR,TEXT$PTR) EXTERNAL:
32 2   DECLARE (ROW,COL,NCHAR) BYTE, TEXT$PTR ADDRESS:
33 2   END:
34 1   SWAP: PROCEDURE(A, B) EXTERNAL:
35 2   DECLARE (A, B) ADDRESS:
36 2   END:

      /* EXTERNAL VARIABLES */
37 1   DECLARE
      OFF$ON$TEXT(16) BYTE EXTERNAL,
      BLANK$LINE(24) BYTE EXTERNAL,
      SWM BYTE EXTERNAL,
      SWM$INDEX BYTE EXTERNAL,
      UNF$ACTIVE$CHAN BYTE EXTERNAL,
      SUBMODE$LAST$STEP$ENABLE ADDRESS EXTERNAL,
      LAST$UNF$ACTIVE$CHAN BYTE EXTERNAL,
      UNF$STATUS BYTE EXTERNAL,
      UNF$FREQ(10) STRUCTURE(DIGITS(6) BYTE) EXTERNAL,
      SUBMODE$GRID ADDRESS EXTERNAL:

```

```

38 1  DECLARE I BYTE;
39 1  DECLARE MIN$FREQ(6) BYTE DATA('225000'),
      MAX$FREQ(6) BYTE DATA('399975');

40 1  INIT$UHF: PROCEDURE PUBLIC;

41 2      DO I = 0 TO 9;
42 3          CALL MOVE(6, MIN$FREQ, UHF$FREQ(I));
43 3      END;
44 2      UHF$STATUS = 0;
45 2      UHF$ACTIVE$CHAN = 0;
46 2      LAST$UHF$ACTIVE$CHAN = 0;
47 2  END;

48 1  UHF$SUBMODE: PROCEDURE PUBLIC;

49 2  DECLARE MAN$FREQ$ICB STRUCTURE(
      MODE BYTE,
      NUM$CH BYTE,
      DEL$M$CH BYTE,
      DEL$M$MASK ADDRESS,
      ECHO$ROW BYTE,
      ECHO$COL BYTE,
      SW$ENABLE ADDRESS) DATA(
      NO$CLEAR$DIGIT, 6, '.', '1000H, 2, 7, SUBMODE$LAST$STEP$ENABLE);

50 2  DISPLAY$STATUS: PROCEDURE;
51 3  DECLARE MODE$TEXT(5) STRUCTURE(CH$ (5) BYTE) DATA(
      ' ', 'T/R', 'T/R+G', 'ADF', 'GARD');

52 3      CALL UPDATE$LINE(1, 0, 3, OFF$CH$TEXT(3*(SHR(UHF$STATUS, 5) AND 1)));
53 3      CALL UPDATE$LINE(1, 5, 5, MODE$TEXT(UHF$STATUS AND 0FH));
54 3      IF (UHF$STATUS AND 10H) > 0
55 4          THEN CALL UPDATE$LINE(0, 5, 3, ('SQL'));
56 3          ELSE CALL UPDATE$LINE(0, 5, 3, BLANK$LINE);
57 3  END; /* OF DISPLAY$STATUS */

58 2  UPDATE$MODE: PROCEDURE;
59 3  DECLARE SW$LIST(5) BYTE DATA(30Q, 32Q, 34Q, 35Q, 36Q);

60 3      I = 0;
61 3      DO WHILE (SW < SW$LIST(I)) AND (I < 5);
62 4          I = I + 1;
63 4      END;
64 3      IF I < 4 THEN I = I + 1;
66 3      UHF$STATUS = (UHF$STATUS AND 0FH) + I;
67 3  END;

68 2  SOL$ON$OFF: PROCEDURE;
69 3      IF SW = SOL$OFF$SW
70 4          THEN UHF$STATUS = UHF$STATUS AND 0EFH;
71 3      ELSE UHF$STATUS = UHF$STATUS OR 10H;

```

```

72 3      END;

73 2      TEST$TONE: PROCEDURE;
74 3      ;
75 3      END;

76 2      ON$OFF: PROCEDURE;
77 3      IF SW = OFF$SW
          THEN UHF$STATUS = UHF$STATUS AND 0DFH; /* OFF */
          ELSE UHF$STATUS = UHF$STATUS OR 20H; /* ON */
80 3      END;

81 2      DECLARE UHF$SUBMODE$TABLEAU(*) BYTE DATA(
          ROW0,'UHF',
          ROW3,'CHAN HAN PRST STAT',
          ROW4,'SEL',COL6,'FREQ',COL11,'CHAN PAGE',
          ROW5,'T/R',COL6,'T/R+ ADF',COL16,'GARD',
          ROW6,COL6,'GARD',
          ROW7,'SQL',COL6,'SQL',COL11,'TEST',
          ROW8,'OFF',COL6,'ON',COL11,'TONE',0);

82 2      DECLARE UHF$SW$ENABLE STRUCTURE(
          SW$MASK(7) BYTE,
          SW$VALUE(9) BYTE) DATA(
          0,0,0E2H,0A2H,0A8H,12H,10H, /* MASK DATA */
          00H,12H,33H,44H,44H,45H,56H,77H,80H);

83 2      DECLARE UHF$ICB STRUCTURE(
          MODE BYTE,
          NUM$CH BYTE,
          DEL$CH BYTE,
          DEL$MASK ADDRESS,
          ECHO$ROW BYTE,
          ECHO$COL BYTE,
          SW$ENABLE ADDRESS) DATA(
          SWITCH,0,0,0,0,0,0,UHF$SW$ENABLE);

84 2      DECLARE UHF$LABEL(3) BYTE AT((UHF$SUBMODE$TABLEAU(1)));
85 2      DECLARE UHF$CNTRL STRUCTURE(
          LABEL$PTR ADDRESS,
          ACTIVE$CHAN$PTR ADDRESS,
          LAST$ACTIVE$CHAN$PTR ADDRESS,
          FREQ$PTR ADDRESS,
          FREQ$SIZE BYTE,
          INSERT$MASK ADDRESS,
          MIN$PTR ADDRESS,
          MAX$PTR ADDRESS,
          MAN$FREQ$ICB$PTR ADDRESS) DATA(
          UHF$LABEL,UHF$ACTIVE$CHAN,LAST$UHF$ACTIVE$CHAN,UHF$FREQ,6,1000H,
          MIN$FREQ,MAX$FREQ,MAN$FREQ$ICB);

```

```
SEJECT
86 2      CALL INITIALIZE$COM(.UHF$CNTL);
87 2      DO FOREVER;
88 3          CALL CLEAR(0);
89 3          CALL GRID(.SUBMODE$GRID);
90 3          CALL DISPLAY(.UHF$SUBMODE$TABLEAU);
91 3          CALL DISPLAY$CHAN$FREQ(UHF$ACTIVE$CHAN);
92 3          CALL DISPLAY$STATUS;
93 3          CALL READ(.UHF$ICB);
94 3          DO CASE SW$INDEX;
95 4              CALL CHAN$SEL;
96 4              CALL MAIN$FREQ(UHF$ACTIVE$CHAN);
97 4              CALL PRST$CHAN;
98 4              CALL STAT$PAGE;
99 4              CALL UPDATE$MODE; /* T/R, T/R+G, ADF, GARD */
100 4          CALL SOL$ON$OFF;
101 4          CALL TEST$TONE;
102 4          CALL ON$OFF;
103 4          CALL SNAP(.UHF$ACTIVE$CHAN, .LAST$UHF$ACTIVE$CHAN); /* LAST */
104 4          END; /* OF DO CASE */
105 3      END; /* OF DO FOREVER */

106 2      END; /* OF UHF$SUBMODE */

107 1      END; /* OF UHF */
```

MODULE INFORMATION:

```
CODE AREA SIZE   = 0296H   646D
VARIABLE AREA SIZE = 0001H   1D
MAXIMUM STACK SIZE = 0008H   8D
239 LINES READ
0 PROGRAM ERROR(S)
```

END OF PL/M-80 COMPILATION

ISIS-II PL/M-80 V3.1 COMPILATION OF MODULE ADF
 OBJECT MODULE PLACED IN :F1:ADF.OBJ
 COMPILER INVOKED BY: PLM80 :F1:ADF.SRC DATE(30OCT79) DEBUG

```

1      $TITLE('ADF')
      ADF: DO;
      $NOLIST INCLUDE(:F1:CDXLIT.SRC)
4 1    INITIALIZE$COM: PROCEDURE(CNTL$PTR) EXTERNAL;
5 2    DECLARE CNTL$PTR ADDRESS;
6 2    END;
7 1    CHAN$SEL: PROCEDURE EXTERNAL;
8 2    END;
9 1    HAN$FREQ: PROCEDURE(CHAN) EXTERNAL;
10 2   DECLARE CHAN BYTE;
11 2   END;
12 1   PRST$CHAN: PROCEDURE EXTERNAL;
13 2   END;
14 1   STAT$PAGE: PROCEDURE EXTERNAL;
15 2   END;
16 1   DISPLAY$CHAN$FREQ: PROCEDURE(CHAN) EXTERNAL;
17 2   DECLARE CHAN BYTE;
18 2   END;
19 1   CLEAR: PROCEDURE(START$ROW) EXTERNAL;
20 2   DECLARE START$ROW BYTE;
21 2   END;
22 1   CLEAR$TEMP$BUF: PROCEDURE EXTERNAL;
23 2   END;
24 1   GRID: PROCEDURE(POINTER) EXTERNAL;
25 2   DECLARE POINTER ADDRESS;
26 2   END;
27 1   DISPLAY: PROCEDURE(IN$PTR) EXTERNAL;
28 2   DECLARE IN$PTR ADDRESS;
29 2   END;
30 1   READ: PROCEDURE(ICB$PTR) EXTERNAL;
31 2   DECLARE ICB$PTR ADDRESS;
32 2   END;
33 1   UPDATE$LINE: PROCEDURE(ROW,COL,NCHAR,TEXT$PTR) EXTERNAL;
34 2   DECLARE (ROW,COL,NCHAR) BYTE, TEXT$PTR ADDRESS;
35 2   END;
36 1   SWAP: PROCEDURE(A, B) EXTERNAL;
37 2   DECLARE (A, B) ADDRESS;
38 2   END;

/* EXTERNAL VARIABLES */
39 1   DECLARE
      OFF$ON$TEXT(6) BYTE EXTERNAL,
      BLANK$LINE(24) BYTE EXTERNAL,
      TEMP$BUF(24) BYTE EXTERNAL,
      SUBNODE$LAST$STEP$ENABLE ADDRESS EXTERNAL,
      SWM BYTE EXTERNAL,
      SWINDEX BYTE EXTERNAL,
      ADF$ACTIVE$CHAN BYTE EXTERNAL,
      LAST$ADF$ACTIVE$CHAN BYTE EXTERNAL,
      ADF$STATUS BYTE EXTERNAL,
      ADF$FREQ(10) STRUCTURE(DIGITS(4) BYTE) EXTERNAL,

```

```

SUBMODE%GRID ADDRESS EXTERNAL;

40 1  DECLARE CM$M LITERALLY '420';
41 1  DECLARE I BYTE;
42 1  DECLARE MIN$FREQ(4) BYTE DATA('0190'),
      MAX$FREQ(4) BYTE DATA('1750');

43 1  INIT$ADF: PROCEDURE PUBLIC;

44 2      DO I = 0 TO 9;
45 3          CALL MOVE(4,MIN$FREQ,ADF$FREQ(I));
46 3      END;
47 2      ADF$STATUS = 0;
48 2      ADF$ACTIVE$CHAN = 0;
49 2      LAST$ADF$ACTIVE$CHAN = 0;
50 2  END;

51 1  ADF$SUBMODE: PROCEDURE PUBLIC;

52 2  DECLARE MAN$FREQ$ICB STRUCTURE(
      MODE BYTE,
      NUM$CH BYTE,
      DEL$CH BYTE,
      DEL$MASK ADDRESS,
      ECHO$ROW BYTE,
      ECHO$COL BYTE,
      SW$ENABLE ADDRESS) DATA(
      NO$CLEAR$DIGIT,4,0,0,2,7,.,SUBMODE$LAST$STEP$ENABLE);

53 2  DISPLAY$STATUS: PROCEDURE;
54 3  DECLARE MODE$TEST$LABEL(3) STRUCTURE(TEXT(4) BYTE)
      DATA('VCE CH '),
      MODE1$LABEL(3) STRUCTURE(TEXT(8) BYTE)
      DATA('RCYR  AUTO ADFMAN ADF ');

55 3      CALL UPDATE$LINE(0,5,4,.,MODE$TEST$LABEL(ADF$STATUS AND 3));
56 3      CALL CLEAR$TEMP$BUF;
57 3      IF (ADF$STATUS AND 20H) > 0 THEN I = 3;
58 3          ELSE I = 0;
59 3      CALL MOVE(3,.,OFF$CH$TEXT(I),.,TEMP$BUF);
60 3      CALL MOVE(8,.,MODE1$LABEL(SHR(ADF$STATUS,2) AND 3),.,TEMP$BUF(5));
61 3      CALL UPDATE$LINE(1,0,13,.,TEMP$BUF);
62 3  END; /* OF DISPLAY$STATUS */

64 2  UPDATE$MODE1: PROCEDURE;
65 3  DECLARE SW$LIST(4) BYTE DATA(300,310,320,340),
      VAL$LIST(4) BYTE DATA(0,0,4,8);

66 3      I = 0;
67 3      DO WHILE (SW < SW$LIST(I)) AND (I < 4);
68 4          I = I + 1;
69 4      END;
70 3      ADF$STATUS = (ADF$STATUS AND (F3H) + VAL$LIST(I);

```

```

71 3      END;

72 2      UPDATE$MODE$A: PROCEDURE;
73 3          ADF$STATUS = ADF$STATUS AND OFCH;
74 3          IF SMV = CW$SW THEN ADF$STATUS = ADF$STATUS + 1;
76 3      END;

77 2      TEST: PROCEDURE;
78 3          ADF$STATUS = (ADF$STATUS AND OFCH) + 2;
79 3      END;

80 2      ON$OFF: PROCEDURE;
81 3          IF SMV = OFF$SW
82 3              THEN ADF$STATUS = ADF$STATUS AND ODFH; /* OFF */
83 3              ELSE ADF$STATUS = ADF$STATUS OR 20H; /* ON */
84 3      END;

85 2      DECLARE ADF$SUBMODE$TABLEAU(*) BYTE DATA(
          ROM0,'ADF',
          ROM3,'CHAN MAN',COL11,'PRST STAT',
          ROM4,'SEL',COL6,'FREQ',COL11,'CHAN PAGE',
          ROM5,'RCVR AUTO MAN',
          ROM6,COL6,'ADF',COL11,'ADF',
          ROM8,'VCE',COL7,'CW',COL11,'TEST',0);

86 2      DECLARE ADF$SW$ENABLE STRUCTURE(
          SW$MASK(7) BYTE,
          SW$VALUE(9) BYTE) DATA(
          0,0,0EEH,0ESH,0ESH,10H,10H,
          00H,12H,33H,44H,44H,55H,56H,77H,80H);

87 2      DECLARE ADF$ICB STRUCTURE(
          MODE BYTE,
          NUM$CH BYTE,
          DEL$CH BYTE,
          DEL$MASK ADDRESS,
          ECHO$ROW BYTE,
          ECHO$COL BYTE,
          SW$ENABLE ADDRESS) DATA(
          SWITCH,0,0,0,0,0,ADF$SW$ENABLE);

88 2      DECLARE ADF$LABEL(3) BYTE AT(ADF$SUBMODE$TABLEAU(1));
89 2      DECLARE ADF$CNTL STRUCTURE(
          LABEL$PTR ADDRESS,
          ACTIVE$CHAN$PTR ADDRESS,
          LAST$ACTIVE$CHAN$PTR ADDRESS,
          FREQ$PTR ADDRESS,
          FREQ$SIZE BYTE,
          INSERT$MASK ADDRESS,
          MIN$PTR ADDRESS,
          MAX$PTR ADDRESS,
          MAN$FREQ$ICB$PTR ADDRESS) DATA(
          .ADF$LABEL,.ADF$ACTIVE$CHAN,.LAST$ADF$ACTIVE$CHAN,.ADF$FREQ,4,0,
          .MIN$FREQ,.MAX$FREQ,.MAN$FREQ$ICB);

```

```

      SELECT
90  4      CALL INITIALIZE$COM(.ADF$CNTL);
91  2      DO FOREVER;
92  3          CALL CLEAR(0);
93  3          CALL GRID(.SUBMODE$GRID);
94  3          CALL DISPLAY(.ADF$SUBMODE$TABLEAU);
95  3          CALL DISPLAY$CHAN$FREQ(ADF$ACTIVE$CHAN);
96  3          CALL DISPLAY$STATUS;
97  3          CALL READ(.ADF$ICB);
98  3          DO CASE SW$INDEX;
99  4              CALL CHAN$SEL;
100 4              CALL MAN$FREQ(ADF$ACTIVE$CHAN);
101 4              CALL PRST$CHAN;
102 4              CALL STAT$PAGE;
103 4              CALL UPDATE$MODE1;
104 4              CALL UPDATE$MODE0;
105 4              CALL TEST;
106 4              CALL ON$OFF;
107 4              CALL SNAP(.ADF$ACTIVE$CHAN, .LAST$ADF$ACTIVE$CHAN); /* LAST */
108 4              END; /* OF DO CASE */
109 3          END; /* OF DO FOREVER */

110 2      END; /* OF ADF$SUBMODE */

111 1      END; /* OF ADF */
```

MODULE INFORMATION:

```
CODE AREA SIZE    = 028BH    651D
VARIABLE AREA SIZE = 0001H    1D
MAXIMUM STACK SIZE = 0008H    8D
244 LINES READ
0 PROGRAM ERROR(S)
```

END OF PL/M-80 COMPILATION

ISIS-II PL/M-80 V3.1 COMPILATION OF MODULE CNV
 OBJECT MODULE PLACED IN :F1:CNV.OBJ
 COMPILER INVOKED BY: PLM80 :F1:CNV.SRC DATE(30OCT79) DEBUG

```

1      $TITLE('CNV')
      CNV: DO:
      $NOLIST INCLUDE(:F1:COLLIT.SRC)
4 1    INITIALIZE$COM: PROCEDURE(CNTL$PTR) EXTERNAL;
5 2    DECLARE CNTL$PTR ADDRESS;
6 2    END;
7 1    CHAN$SEL: PROCEDURE EXTERNAL;
8 2    END;
9 1    MAN$FREQ: PROCEDURE(CHAN) EXTERNAL;
10 2   DECLARE CHAN BYTE;
11 2   END;
12 1   PRST$CHAN: PROCEDURE EXTERNAL;
13 2   END;
14 1   STAT$PAGE: PROCEDURE EXTERNAL;
15 2   END;
16 1   DISPLAY$CHAN$FREQ: PROCEDURE(CHAN) EXTERNAL;
17 2   DECLARE CHAN BYTE;
18 2   END;
19 1   CLEAR: PROCEDURE(START$ROW) EXTERNAL;
20 2   DECLARE START$ROW BYTE;
21 2   END;
22 1   CLEAR$TEMP$BUF: PROCEDURE EXTERNAL;
23 2   END;
24 1   GRID: PROCEDURE(POINTER) EXTERNAL;
25 2   DECLARE POINTER ADDRESS;
26 2   END;
27 1   DISPLAY: PROCEDURE(IN$PTR) EXTERNAL;
28 2   DECLARE IN$PTR ADDRESS;
29 2   END;
30 1   READ: PROCEDURE(ICB$PTR) EXTERNAL;
31 2   DECLARE ICB$PTR ADDRESS;
32 2   END;
33 1   UPDATE$LINE: PROCEDURE(ROW,COL,NCHAR,TEXT$PTR) EXTERNAL;
34 2   DECLARE (ROW,COL,NCHAR) BYTE, TEXT$PTR ADDRESS;
35 2   END;
36 1   SWAP: PROCEDURE(A, B) EXTERNAL;
37 2   DECLARE (A, B) ADDRESS;
38 2   END;

/* EXTERNAL VARIABLES */
39 1   DECLARE
      OFF$ON$TEXT(6) BYTE EXTERNAL,
      BLANK$LINE(24) BYTE EXTERNAL,
      TEMP$BUF(24) BYTE EXTERNAL,
      SWM BYTE EXTERNAL,
      SUBMODE$LAST$STEP$ENABLE ADDRESS EXTERNAL,
      SWINDEX BYTE EXTERNAL,
      CNV$ACTIVE$CHAN BYTE EXTERNAL,
      LAST$CNV$ACTIVE$CHAN BYTE EXTERNAL,
      CNV$STATUS BYTE EXTERNAL,
      CNV$MB$VOL BYTE EXTERNAL,

```

```

CNV$NAV$VOL BYTE EXTERNAL,
CNV$FREQ(10) STRUCTURE(DIGITS(5) BYTE) EXTERNAL,
SUBNODE$GRID ADDRESS EXTERNAL;

40 1  DECLARE VOL$INC$SW LITERALLY '320',
      MB$HIGH$SW LITERALLY '350';

41 1  DECLARE I BYTE;
42 1  DECLARE MIN$FREQ(5) BYTE DATA('10800'),
      MAX$FREQ(5) BYTE DATA('11750');
43 1  DECLARE VOL$DATA$PTR ADDRESS,
      VOL BASED VOL$DATA$PTR BYTE;

44 1  INIT$CNV: PROCEDURE PUBLIC;

45 2      DO I = 0 TO 9;
46 3          CALL MOVE(5, MIN$FREQ, CNV$FREQ(I));
47 3      END;
48 2      CNV$STATUS = 0;
49 2      CNV$ACTIVE$CHAN = 0;
50 2      LAST$CNV$ACTIVE$CHAN = 0;
51 2      VOL$DATA$PTR = ,CNV$MB$VOL;
52 2      CNV$MB$VOL = 0;
53 2      CNV$NAV$VOL = 0;
54 2  END;

55 1  CNV$SUBMODE: PROCEDURE PUBLIC;

56 2  DECLARE MAN$FREQ$ICB STRUCTURE(
      MODE BYTE,
      NUM$CH BYTE,
      DEL$M$CH BYTE,
      DEL$M$MASK ADDRESS,
      ECHO$ROW BYTE,
      ECHO$COL BYTE,
      SW$ENABLE ADDRESS) DATA(
      NO$CLEAR$DIGIT, 5, ',', 1000H, 2, 7, , SUBMODE$LAST$STEP$ENABLE);

57 2  VOL$ADJUST: PROCEDURE;
58 3      IF SWV = VOL$INC$SW THEN VOL = VOL + 1;
60 3                      ELSE VOL = VOL - 1;
61 3  END; /* OF VOL$ADJUST */

62 2  MB$HIGH$LOW: PROCEDURE;

63 3      IF SWV = MB$HIGH$SW THEN CNV$STATUS = CNV$STATUS OR 1;
65 3                      ELSE CNV$STATUS = CNV$STATUS AND 0FH;
66 3  END; /* OF MB$HIGH$LOW */

67 2  CNV$OFF: PROCEDURE;
68 3      IF SWV = OFF$SW
      THEN CNV$STATUS = CNV$STATUS AND 0DF; /* OFF */

```

AD-A119 133

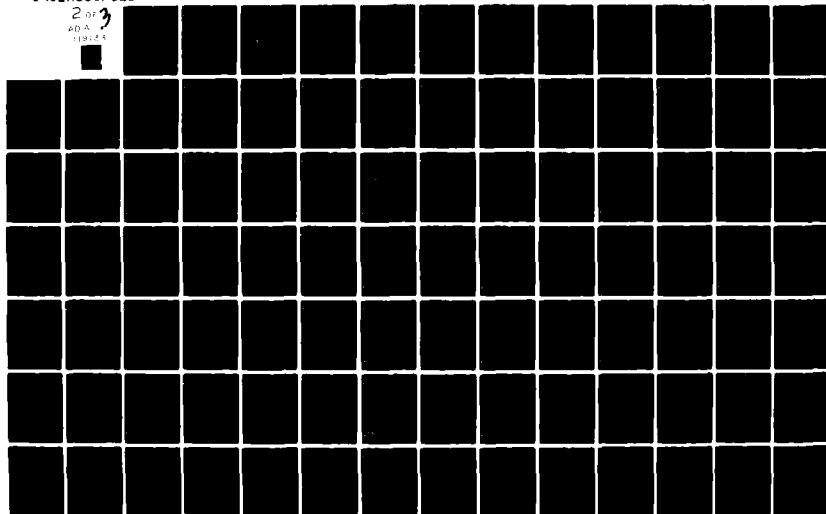
LITTON SYSTEMS INC. VAN NUYS CALIF DATA SYSTEMS DIV
CONTROL DISPLAY UNIT PROGRAM.(U)
1978

F/G 9/2

UNCLASSIFIED

NL

2 of 3
ADA
119-133



```

70 3      ELSE CNV%STATUS = CNV%STATUS OR 20H;    /* ON */
71 3      END;

72 2      DISPLAY%STATUS: PROCEDURE;
73 3      DECLARE I BYTE;
74 3      DECLARE MB$HILO$TEXT(2) STRUCTURE(CH(5) BYTE) DATA('MB-LO','MB-HI');

75 3      IF (CNV%STATUS AND 20H) > 0 THEN I = 3;
76 3      ELSE I = 0;
77 3      CALL UPDATE%LINE(1,0,3,OFF$ON$TEXT(I));
78 3      CALL UPDATE%LINE(1,19,5,MB$HILO$TEXT(CNV%STATUS AND 1));
79 3      END;
80 3      END;

81 2      DECLARE CNV%SUBMODE$TABLEAU(*) BYTE DATA(
      R0M0,'CNV',
      R0M3,'CHAN MAN',COL11,'PRST STAT',
      R0M4,'SEL',COL6,'FREQ',COL11,'CHAN PAGE',
      R0M5,COL2,'MB',COL6,'INC',COL17,'MB',
      R0M6,'VOL',COL6,'VOL',COL17,'HI',
      R0M7,'NAV',COL6,'DEC',COL17,'MB',
      R0M8,'VOL',COL6,'VOL',COL11,'TEST LOW',0);

82 2      DECLARE CNV%SMENABLE STRUCTURE(
      SM$MASK(7) BYTE,
      SM$VALUE(8) BYTE) DATA(
      0,0,0EEH,0A4H,0ACH,13H,10H,
      00H,12H,33H,45H,67H,58H,69H,9AH);

83 2      DECLARE CNV%ICB STRUCTURE(
      MODE BYTE,
      N$M$MCH BYTE,
      DEL$M$MCH BYTE,
      DEL$M$MASK ADDRESS,
      ECHO$ROW BYTE,
      ECHO$COL BYTE,
      SM$ENABLE ADDRESS) DATA(SWITCH,0,0,0,0,0,CNV%SMENABLE);

84 2      DECLARE CNV$LABEL(3) BYTE AT(1,CNV%SUBMODE$TABLEAU(1));
85 2      DECLARE CNV$CNTRL STRUCTURE(
      LABEL$PTR ADDRESS,
      ACTIVE$CHAN$PTR ADDRESS,
      LAST$ACTIVE$CHAN$PTR ADDRESS,
      FREQ$PTR ADDRESS,
      FREQ$SIZE BYTE,
      INSERT$MASK ADDRESS,
      MIN$PTR ADDRESS,
      MAX$PTR ADDRESS,
      MIN$FREQ$ICB$PTR ADDRESS) DATA(
      .CNV$LABEL,.CNV$ACTIVE$CHAN,.LAST$CNV$ACTIVE$CHAN,.CNV$FREQ,5,1000H,
      .MIN$FREQ,.MAX$FREQ,.MAN$FREQ$ICB);

```

```

      REJECT
06 2      CALL INITIALIZE$CON(.CNV$CNTL);
07 2      DO FOREVER:
08 3          CALL CLEAR(0);
09 3          CALL GRID(.SUBMODE$GRID);
10 3          CALL DISPLAY(.CNV$SUBMODE$TABLEAU);
11 3          CALL DISPLAY$CH$FREQ(CNV$ACTIVE$CHAN);
12 3          CALL DISPLAY$STATUS;
13 3          CALL READ(.CNV$ICB);
14 3          DO CASE SW$INDEX:
15 4              CALL CHANSEL;
16 4              CALL MAN$FREQ(CNV$ACTIVE$CHAN);
17 4              CALL PRST$CHAN;
18 4              CALL STAT$PAGE;
19 4              VOL$DATA$PTR = .CNV$MB$VOL;           /* MB VOL */
100 4          CALL VOL$ADJUST;                          /* INC VOL, DEC VOL */
101 4          CALL MB$HIGH$LOW;                          /* MB HIGH, MB LOW */
102 4          VOL$DATA$PTR = .CNV$NAV$VOL;               /* NAV VOL */
103 4          ;                                           /* TEST */
104 4          CALL ON$OFF;
105 4          CALL SWAP(.CNV$ACTIVE$CHAN, .LAST$CNV$ACTIVE$CHAN); /* LAST */
106 4          END; /* OF DO CASE */
107 3          END; /* OF DO FOREVER */

108 2      END; /* OF CNV$SUBMODE */

109 1      END; /* OF CNV */

```

MODULE INFORMATION:

```

CODE AREA SIZE      = 023FH   5750
VARIABLE AREA SIZE  = 0004H   40
MAXIMUM STACK SIZE  = 0008H   80
241 LINES READ
0 PROGRAM ERROR(S)

```

END OF PL/M-60 COMPILATION

ISIS-II PL/M-80 V3.0 COMPILATION OF MODULE DAT
 OBJECT MODULE PLACED IN :F1:CD8AT.OBJ
 COMPILER INVOKED BY: PLM80 :F1:CD8AT.SRC DATE(19DEC78) 0808

```

      $TITLE('CD8AT')
1      DAT: DAT
2 1      DECLARE
          WFACTIVESCHN BYTE PUBLIC,
          LASTWFACTIVESCHN BYTE PUBLIC,
          WFOSTATUS BYTE PUBLIC,
          WFOFIBER(10) STRUCTURE(DIGITS(4) BYTE) PUBLIC;

3 1      DECLARE WFACTIVESCHN BYTE PUBLIC,
          LASTWFACTIVESCHN BYTE PUBLIC,
          WFOSTATUS BYTE PUBLIC,
          WFOFIBER(10) STRUCTURE(DIGITS(4) BYTE) PUBLIC;

4 1      DECLARE WFACTIVESCHN BYTE PUBLIC,
          LASTWFACTIVESCHN BYTE PUBLIC,
          WFOSTATUS BYTE PUBLIC,
          WFOFIBER(10) STRUCTURE(DIGITS(4) BYTE) PUBLIC;

5 1      DECLARE WFACTIVESCHN BYTE PUBLIC,
          LASTWFACTIVESCHN BYTE PUBLIC,
          WFOSTATUS BYTE PUBLIC,
          WFOFIBER(10) STRUCTURE(DIGITS(4) BYTE) PUBLIC;

6 1      DECLARE WFACTIVESCHN BYTE PUBLIC,
          LASTWFACTIVESCHN BYTE PUBLIC,
          WFOSTATUS BYTE PUBLIC,
          WFOFIBER(10) STRUCTURE(DIGITS(4) BYTE) PUBLIC;

7 1      DAT: /> DAT </

```

MODULE INFORMATION:

CODE AREA SIZE = 0000H 00
 VARIABLE AREA SIZE = 0009H 217D
 MAXIMUM STACK SIZE = 0000H 00
 34 LINES READ
 0 PROGRAM ERROR(S)

END OF PL/M-80 COMPILATION

ISIS-II PL/M-80 V3.1 COMPILATION OF MODULE FONTDA
 OBJECT MODULE PLACED IN :F1:FONTDA.OBJ
 COMPILER INVOKED BY: PLM80 :F1:FONTDA.SRC DATE(27OCT79) DEBUG

```

1      $TITLE('FONTDA')
2 1      FONTDA: DO:
      DECLARE CH$FONT$DATA(320) BYTE PUBLIC DATA(
        0,0,0,0,0, /% SP %/
        0,0,5FH,0,0, /% ! %/
        6,9,9,6,0, /% DEG SYM %/
        0,7FH,3EH,1CH,8, /% SOLID RIGHT ARROW %/
        0,0CH,2AH,7FH,2AH, /% $ %/
        0,63H,13H,8H,64H, /% Z %/
        0,36H,49H,56H,20H, /% & %/
        0,0,0,7,0, /% ' %/
        0,0,1CH,22H,41H, /% ( %/
        0,0,41H,22H,1CH, /% ) %/
        2AH,1CH,3EH,1CH,2AH, /% * %/
        8,8,3EH,8,8, /% + %/
        0,40H,30H,0,0, /% , %/
        8,8,8,8,8, /% - %/
        0,60H,60H,0,0, /% . %/
        60H,10H,8,4,3, /% / %/
        3EH,51H,49H,45H,3EH, /% 0 %/
        44H,42H,7FH,40H,40H, /% 1 %/
        72H,49H,49H,49H,46H, /% 2 %/
        22H,41H,49H,49H,36H, /% 3 %/
        18H,14H,12H,7FH,10H, /% 4 %/
        27H,43H,43H,43H,39H, /% 5 %/
        3CH,44H,49H,49H,30H, /% 6 %/
        61H,11H, 9, 5, 3, /% 7 %/
        36H,49H,49H,49H,36H, /% 8 %/
        46H,49H,49H,29H,1EH, /% 9 %/
        0,36H,36H,0,0, /% : %/
        0,46H,36H,0,0, /% ; %/
        8,14H,22H,41H,0, /% < %/
        14H,14H,14H,14H,14H, /% = %/
        0,41H,22H,14H,8, /% > %/
        2,1,51H,9H,6, /% ? %/
        3EH,41H,50H,55H,2EH, /% @ %/
        7CH,12H,11H,12H,7CH, /% A %/
        7FH,49H,49H,49H,36H, /% B %/
        3EH,41H,41H,41H,22H, /% C %/
        7FH,41H,41H,22H,1CH, /% D %/
        7FH,49H,49H,49H,41H, /% E %/
        7FH,9,9,9,1, /% F %/
        3EH,41H,51H,51H,72H, /% G %/
        7FH,8,8,8,7FH, /% H %/
        0,41H,7FH,41H,0, /% I %/
        20H,40H,41H,3FH,1, /% J %/
        7FH,8,14H,22H,41H, /% K %/
        7FH,40H,40H,40H,40H, /% L %/
        7FH,2,0CH,2,7FH, /% M %/
        7FH,6,0CH,18H,7FH, /% N %/
        3EH,41H,41H,41H,3EH, /% O %/

```

```
7FH,9,9,9,6, /% P %/  
3EH,41H,51H,21H,5EH, /% Q %/  
7FH,9,19H,29H,46H, /% R %/  
22H,45H,45H,51H,22H, /% S %/  
1,1,7FH,1,1, /% T %/  
3FH,40H,40H,40H,3FH, /% U %/  
7,18H,60H,18H,7, /% V %/  
7FH,20H,1CH,20H,7FH, /% W %/  
43H,14H,8,14H,63H, /% X %/  
3,4,78H,4,3, /% Y %/  
41H,51H,49H,45H,43H, /% Z %/  
))
```

3 1 END: /% OF COLDAT %/

MODULE INFORMATION:

```
CODE AREA SIZE = 0140H 3200  
VARIABLE AREA SIZE = 0000H 00  
MAXIMUM STACK SIZE = 0000H 00  
65 LINES READ  
0 PROGRAM ERROR(S)
```

END OF PL/M-80 COMPILATION

ISIS-II PL/M-80 V3.1 COMPILATION OF MODULE IFF
 OBJECT MODULE PLACED IN :F1:IFF.OBJ
 COMPILER INVOKED BY: PL/M80 :F1:IFF.SRC DATE(29OCT79) DEBUG

```

      $TITLE('IFF')
1      IFF: DO:
      $NLIST INCLUDE(:F1:CDULIT.SRC)
4      1      ERROR: PROCEDURE(CODE) EXTERNAL;
5      2      DECLARE CODE BYTE;
6      2      END;
7      1      CLEARLINE: PROCEDURE(LINE) EXTERNAL;
8      2      DECLARE LINE BYTE;
9      2      END;
10     1      CLEAR: PROCEDURE(START$ROW) EXTERNAL;
11     2      DECLARE START$ROW BYTE;
12     2      END;
13     1      CLEAR$TEMP$BUF: PROCEDURE EXTERNAL;
14     2      END;
15     1      GRID: PROCEDURE(POINTER) EXTERNAL;
16     2      DECLARE POINTER ADDRESS;
17     2      END;
18     1      DISPLAY: PROCEDURE(IN$PTR) EXTERNAL;
19     2      DECLARE IN$PTR ADDRESS;
20     2      END;
21     1      READ: PROCEDURE(ICB$PTR) EXTERNAL;
22     2      DECLARE ICB$PTR ADDRESS;
23     2      END;
24     1      LIMIT$TEST: PROCEDURE(BUF$PTR,NUM$DIGITS,MIN$PTR,MAX$PTR) BYTE EXTERNAL;
25     2      DECLARE NUM$DIGITS BYTE, (BUF$PTR,MIN$PTR,MAX$PTR) ADDRESS;
26     2      END;
27     1      UPDATE$LINE: PROCEDURE(ROW,COL,NCHAR,TEXT$PTR) EXTERNAL;
28     2      DECLARE (ROW,COL,NCHAR) BYTE, TEXT$PTR ADDRESS;
29     2      END;
30     1      RAD$TEST: PROCEDURE;
31     2      ;
32     2      END;

33     1      DECLARE NORM$SW LITERALLY '320';

/* EXTERNAL VARIABLES */
34     1      DECLARE
      OFF$ON$TEXT(2) STRUCTURE(CHAR(3) BYTE) EXTERNAL,
      BLANK$LINE(24) BYTE EXTERNAL,
      TEMP$BUF(24) BYTE EXTERNAL,
      SW BYTE EXTERNAL,
      SW$INDEX BYTE EXTERNAL,
      IN$BUF(16) BYTE EXTERNAL,
      DATA$ENTERED BYTE EXTERNAL,
      DIGIT$KB$GRID ADDRESS EXTERNAL,
      OCTAL$KB$TABLEAU ADDRESS EXTERNAL,
      IFF$STATUS BYTE EXTERNAL,
      IFF$MISC$CODE(2) BYTE EXTERNAL,
      IFF$MISC$CODE(4) BYTE EXTERNAL,
      IFF$MISC$CODE(4) BYTE EXTERNAL,
      IFF$MISC$MODE BYTE EXTERNAL,

```

IFF#ANTENNA#POSITION BYTE EXTERNAL.
SUBMODE#GRID ADDRESS EXTERNAL;

```

35 1  DECLARE ON#OFF#SUB#MODE#ENABLE STRUCTURE(
      SW#MASK(7) BYTE,
      SW#VALUE(2) BYTE) DATA(
      0,0,0,0,0,18H,80H, /* OFF, ON, SUBMODE */
      01H,20H);
36 1  DECLARE GARD#ENABLE STRUCTURE(
      SW#MASK(7) BYTE) DATA(0,0,0,0,0,0,0A0H); /* VALUE NOT IMPORTANT */
37 1  DECLARE GARD#ICB STRUCTURE(
      MODE BYTE,
      MUP#CH BYTE,
      DEL#MCH BYTE,
      DEL#MASK ADDRESS,
      ECHO#ROW BYTE,
      ECHO#COL BYTE,
      SUB#ENABLE ADDRESS) DATA(SWITCH,0,0,0,0,0,.,GARD#ENABLE);

38 1  INIT#IFF: PROCEDURE PUBLIC;
      /* INITIALIZATION OF THE CODE DATA BELOW REQUIRES THAT THE DATA BE
      LOCATED CONTIGUOUSLY AND THAT MI#CODE APPEAR FIRST IN THE LIST
      (HAVE THE LOWEST ADDRESS). */

39 2      IFF#STATUS = 0;
40 2      IFF#SUB#MODE = 0;
41 2      IFF#ANTENNA#POSITION = 0;
42 2      IFF#MI#CODE(0) = '0';
43 2      CALL MOVE(9,.,IFF#MI#CODE,.,IFF#MI#CODE + 1);
44 2  END;

45 1  INDEX: PROCEDURE(VALUE,BIT#LOC,MASK) BYTE;
46 2  DECLARE(VALUE,BIT#LOC,MASK) BYTE;

47 2      RETURN(SHR(VALUE,BIT#LOC) AND MASK);
48 2  END;

49 1  DECLARE I BYTE;

50 1  IFF#ON#OFF: PROCEDURE;

51 2  IFF#OFF: PROCEDURE;

52 3      CALL DISPLAY(1,ROW2, 'OFF GARD#',0);
53 3      CALL READ(.,GARD#ICB);
54 3      IF SW = SUB#MODE#SW THEN RETURN;
55 3      CALL CLEAR#LINE(2);
56 3      IFF#STATUS = IFF#STATUS AND 00FH;
57 3      END; /* IF IFF#OFF */

58 2      IF SW = OFF#SW THEN CALL IFF#OFF;
59 2      ELSE IFF#STATUS = IFF#STATUS OR 20H;
60 2  END;

```

```

63 1  NORMSTBY: PROCEDURE:
64 2    IF SMV = NORMISM THEN IFFSTATUS = IFFSTATUS AND OBFM:
65 2      ELSE IFFSTATUS = IFFSTATUS OR O40H:
66 2
67 2  END:
68 1  TEST: PROCEDURE:
69 2  DECLARE TEST$TABLEAU(*) BYTE DATA(
      ROW0,COL8,'IFF TEST',
      ROW4,' M-1',COL6,'M-2',COL11,'M-3A',
      ROW6,' M-4',
      ROW8,' M-C',0):
70 2  DECLARE TEST$ENABLE STRUCTURE(
      SW$MASK(7) BYTE,
      SW$VALUE(3) BYTE) DATA(
      0,0,0A0H,80H,80H,0,80H,
      01H,23H,45H):
71 2  DECLARE TEST$ICB STRUCTURE(
      MODE BYTE,
      NUM$CH BYTE,
      DEL$MCH BYTE,
      DEL$MASK ADDRESS,
      ECHO$ROW BYTE,
      ECHO$COL BYTE,
      SW$ENABLE ADDRESS) DATA(SWITCH,0,0,0,0,0,TEST$ENABLE):

72 2  CALL CLEAR(0):
73 2  IF IFF$MCHMODE THEN TEMP$BUF(0) = 'B':
74 2      ELSE TEMP$BUF(0) = 'A':
75 2
76 2  CALL UPDAT$LINE(6,4,1,TEMP$BUF):
77 2  CALL CLEAR$TEMP$BUF:
78 2  CALL MOVE(8,TEST$TABLEAU + 2,TEMP$BUF):
79 2  CALL MOVE(6,('PASSED'),TEMP$BUF(11)):
80 2  CALL DISPLAY(.TEST$TABLEAU):
81 2  CALL GRID(.SUBMODE$GRID):
82 2  DO FOREVER:
83 3    CALL READ(.TEST$ICB):
84 3    IF SMV = SUBMODE$SM THEN RETURN:
85 3    IF SW$INDEX = 4
86 3      THEN TEMP$BUF(9) = 'C':
87 3      ELSE TEMP$BUF(9) = SW$INDEX + 31H:
88 3    CALL UPDAT$LINE(2,0,17,TEMP$BUF):
89 3
90 3  END: /* OF DO FOREVER */
91 2  END: /* OF TEST */

92 1  DIS$PLAY$SUBMODE$STATUS$LINE: PROCEDURE:
93 2  DECLARE BUF$PTR ADDRESS,
      (BUF BASED BUF$PTR)(1) BYTE,
      MASK BYTE:
94 2  DECLARE NORM$STBY$TEXT(2) STRUCTURE(CHAR(4) BYTE) DATA('NORM$STBY'):

95 2  CALL CLEAR$LINE(1):
96 2  CALL UPDAT$LINE(0,10,4,NORM$STBY$TEXT(INDEX(1,IFFSTATUS,6,1))):
97 2  BUF$PTR = TEMP$BUF + 4:
98 2  CALL CLEAR$TEMP$BUF:

```

```
99 2      CALL MOVE(3,OFFROW$TEXT(INDEX(IFF$STATUS,5,1)),TEMP$BUF);
100 2      MASK = IFF$STATUS;
101 2      DO I = 0 TO 4;
102 3          IF MASK THEN DO; /* INSERT MASK TEXT FOR 'ON' UNITS */
104 4              BUF(0) = 'M';
105 4              BUF(1) = I + 31H;
106 4              END;
107 3          BUF$PTR = BUF$PTR + 4;
108 3          MASK = SHR(MASK,1);
109 3          END; /* OF DO I */
110 2      IF (IFF$STATUS AND 4) > 0 THEN TEMP$BUF(14) = 'A';
112 2      IF (IFF$STATUS AND 8) > 0 THEN DO;
114 3          IF IFF$MODE THEN TEMP$BUF(18) = 'B';
116 3          ELSE TEMP$BUF(18) = 'A';
117 3          END;
118 2      IF (IFF$STATUS AND 10H) > 0 THEN TEMP$BUF(21) = 'C';
120 2      CALL UPDATERLINE(1,0,22,TEMP$BUF);
121 2      END; /* OF DISPLAY$SUBNODE$STATUS$LINE */
```

```

SELECT
122 1  M4$SELECT: PROCEDURE;
123 2  DECLARE M4$TABLEAU(*) BYTE DATA(
      ROW2,'M-4 SELECT',
      ROW3,COL7,'A',COL13,'B',COL16,'HOLD',
      ROW5,COL6,'AUD',COL11,'AUD',COL16,'OUT',
      ROW6,'M-4',COL11,'LITE',0);
124 2  DECLARE M4$ENABLE STRUCTURE(
      SM$MASK(7) BYTE,
      SM$VALUE(5) BYTE) DATA(
      0,0,2CH,2CH,0,18H,80H,
      01H,23H,45H,67H,80H);
125 2  DECLARE M4$ICB STRUCTURE(
      MODE BYTE,
      NUM$CH BYTE,
      DEL$M$CH BYTE,
      DEL$M$MASK ADDRESS,
      ECHO$ROW BYTE,
      ECHO$COL BYTE,
      SM$ENABLE ADDRESS) DATA(SWITCH,0,0,0,0,0,0,M4$ENABLE);
126 2  DECLARE HOLD$TEXT(*) BYTE DATA('HOLD');
127 2  DISPLAYM4$STATUS: PROCEDURE;
128 3  DECLARE AUD$LITE$TEXT(3) STRUCTURE(CHAR(8) BYTE)
      DATA(' ','AUD ','AUD LITE');
129 3  CALL CLEAR$LINE(1);
130 3  CALL CLEAR$TEMP$BUF;
131 3  IF (IFF$M4$MODE AND 2) > 0 THEN
132 3  CALL MOVE(4,HOLD$TEXT,TEMP$BUF(20));
133 3  CALL MOVE(8,AUD$LITE$TEXT(INDEX(IFF$M4$MODE,2,3)),TEMP$BUF(6));
134 3  CALL MOVE(3,OFF$ON$TEXT(INDEX(IFF$STATUS,3,1)),TEMP$BUF);
135 3  CALL UPDATE$LINE(1,0,24,TEMP$BUF);
136 3  IF IFF$M4$MODE THEN TEMP$BUF(0) = 'B';
137 3  ELSE TEMP$BUF(0) = 'A';
138 3  CALL UPDATE$LINE(2,3,1,TEMP$BUF);
139 3  CALL UPDATE$LINE(2,3,1,TEMP$BUF);
140 3  END;

/* BEGIN M4$SELECT CODE */
141 2  CALL CLEAR(1);
142 2  CALL GRID(.SUBMODE$GRID);
143 2  CALL DISPLAY(.M4$TABLEAU);
144 2  DO FOREVER;
145 3  CALL DISPLAYM4$STATUS;
146 3  CALL READ(.M4$ICB);
147 3  DO CASE SM$INDEX;
148 4  IFF$M4$MODE = IFF$M4$MODE AND OFEH; /* A */
149 4  IFF$M4$MODE = IFF$M4$MODE OR 1; /* B */
150 4  IFF$M4$MODE = (IFF$M4$MODE AND OFCH) OR
      ((NOT IFF$M4$MODE) AND 2); /* HOLD */
151 4  IFF$M4$MODE = (IFF$M4$MODE AND OF3H) OR 4; /* AUD */
152 4  IFF$M4$MODE = (IFF$M4$MODE AND OF3H) OR 8; /* AUD LITE */
153 4  IFF$M4$MODE = IFF$M4$MODE AND OF3H; /* OUT */
154 4  IFF$STATUS = IFF$STATUS AND OF7H; /* OFF */
155 4  IFF$STATUS = IFF$STATUS OR 8; /* ON */
156 4  RETURN; /* SUBMODE */
157 4  END; /* OF DO CASE */

```

PL/M-60 COMPILER IFF

29OCT79 PAGE 6

158 3 END: /* OF DO FOREVER */
159 2 END: /* OF M4\$SELECT */

```
%SELECT  
160 1 DECLARE ANT$TEXT(3) STRUCTURE(CHAR(3) BYTE) DATA('TOPDIVBOT');  
  
161 1 ANT$SELECT: PROCEDURE;  
162 2 DECLARE ANT$TABLEAU(*) BYTE DATA(  
    ROW2,'ANT SELECT',  
    ROW4,' TOP',  
    ROW6,' DIV',  
    ROW8,' BOT',0);  
163 2 DECLARE ANT$ENABLE STRUCTURE(  
    SW$MASK(7) BYTE,  
    SW$VALUE(3) BYTE) DATA(  
    0,0,0COH,0COH,0COH,0COH,0,0SH, /* TOP, DIV, BOT, SUBMODE */  
    0OH,11H,22H);  
164 2 DECLARE ANT$I/CB STRUCTURE(  
    MODE BYTE,  
    NUM$CH BYTE,  
    DELM$CH BYTE,  
    DELM$MASK ADDRESS,  
    ECHO$ROW BYTE,  
    ECHO$COL BYTE,  
    SM$ENABLE ADDRESS) DATA(SWITCH,0,0,0,0,0,.ANT$ENABLE);  
  
165 2 CALL CLEAR(2);  
166 2 CALL DISPLAY(.ANT$TABLEAU);  
167 2 CALL GRID(.SUBMODE$GRID);  
168 2 CALL CLEAR$LINE(1);  
169 2 CALL UPDATE$LINE(1,20,3,.ANT$TEXT(1FF$ANTENNA$POSITION));  
170 2 CALL READ(.ANT$I/CB);  
171 2 IF SW = SUBMODE$SW THEN RETURN;  
173 2 IFF $ANTENNA$POSITION = SW$INDEX;  
174 2 END; /* OF ANT$SELECT */
```

```
REJECT
175 1  MC$Q:OFF: PROCEDURE:
176 2  DECLARE MC$TABLEAU(*) BYTE DATA(
      ROW2,'M-C ON/OFF',0);
177 2  DECLARE MC$ICB STRUCTURE(
      MODE BYTE,
      NUMSCH BYTE,
      DELMSCH BYTE,
      DELMASK ADDRESS,
      ECHO$ROW BYTE,
      ECHO$COL BYTE,
      SIMENABLE ADDRESS) DATA(SWITCH,0,0,0,0,0,ON$OFF$SUBMODE$ENABLE);

178 2  CALL CLEAR(1);
179 2  CALL UPDATE$LINE(0,20,4, BLANK$LINE);
180 2  CALL GRID(.SUBMODE$GRID);
181 2  CALL DISPLAY(.MC$TABLEAU);
182 2  CALL UPDATE$LINE(1,0,3, OFF$ON$TEXT(INDEX(IFF$STATUS,4,1)));
183 2  CALL READ(.MC$ICB);
184 2  IF SMV = SUBMODE$SW THEN RETURN;
186 2  IF SM$INDEX THEN IFF$STATUS = IFF$STATUS OR 10H; /* ON */
188 2  ELSE IFF$STATUS = IFF$STATUS AND 0EFH; /* OFF */
189 2  END; /* OF MC$ON$OFF */
```

```

SELECT
190 1 CODEENTRY: PROCEDURE;
/* THIS PROCEDURE HANDLES DATA ENTRIES FOR M1, M2, AND M3A CODES. THE
SPECIFIC CODE UPDATED IS DETERMINED BY THE VALUE OF SW$INDEX ON ENTRY:
SW$INDEX = 0 - M1
          = 1 - M2
          = 2 - M3
*/
191 2 DECLARE ENTRY$TABLEAU(*) BYTE DATA(
      ROM2,'ENTER CODE',0);

192 2 DECLARE INIT$CODE$ICB STRUCTURE(
      MODE BYTE,
      NUM$CH BYTE,
      DEL$M$CH BYTE,
      DEL$M$MASK ADDRESS,
      ECHO$ROW BYTE,
      ECHO$COL BYTE,
      SW$ENABLE ADDRESS) DATA(OCTAL,0,0,0,2,11,ON$OFF$SUBMODE$ENABLE);

193 2 DECLARE CODE$ICB STRUCTURE(
      MODE BYTE,
      NUM$CH BYTE,
      DEL$M$CH BYTE,
      DEL$M$MASK ADDRESS,
      ECHO$ROW BYTE,
      ECHO$COL BYTE,
      SW$ENABLE ADDRESS);

194 2 DECLARE CODE$TAB(*) BYTE DATA(ROM2,'ENTER CODE',0);
195 2 DECLARE LABEL$TAB(3) STRUCTURE(TEXT(7) BYTE) DATA(
      ROM4,COL1,'M-1',0,
      ROM4,COL6,'M-2',0,
      ROM4,COL11,'M-3A',0);

196 2 DECLARE NUM$DIGITS(3) BYTE DATA(2,4,4),
      DATA$ADD(3) ADDRESS DATA(.IFF$M1$CODE,.IFF$M2$CODE,.IFF$M3$CODE);
197 2 DECLARE MIN$CODE(4) BYTE DATA('0000'),
      MAX$CODE(3) STRUCTURE(DIGITS(4) BYTE) DATA(
      '7300','7777','7777');

198 2 DECLARE GARD$TEXT$TAB(*) BYTE DATA(ROM2,'M2 GUARDED',0);
199 2 DECLARE (IX,N,ON$OFF$MASK) BYTE;

```

```

      SELECT
200  2      IX = SW%INDEX;
201  2      IF IX > 0 THEN ON%OFF%MASK = SHL(1,IX);
203  2          ELSE ON%OFF%MASK = 1;
204  2      N = NUM%DIGITS(IX);
205  2      CALL CLEAR(2);
206  2      CALL GRID(.DIGIT%K%GRID);
207  2      IF IX = 1 THEN DO: /* M2 IS GUARDED */
209  3          CALL DISPLAY(.GARD%TEXT%TAB);
210  3          CAL' READ(.GARD%ICB);
211  3          IF SW% = SUBMODE%SW THEN RETURN;
213  3          END;
214  2      CALL CLEAR%LINE(2);
215  2      CALL DISPLAY(.CODE%TAB);
216  2      CALL DISPLAY(.OCTAL%*B%TABLEAU);
217  2      CALL DISPLAY(.LABEL%TAB(IX));
218  2      CALL MOVE(9,..INIT%CODE%ICB,..CODE%ICB);
219  2      CODE%ICB.NUM%CH = N;
220  2      DO FOREVER;
221  3          CALL DISPLAY%SUBMODE%STATUS%LINE;
222  3          CALL READ(.CODE%ICB);
223  3          IF DATA%ENTERED AND (SW% = ENTER%SW) THEN DO;
225  4              IF LIMIT%TEST(.IN%BUF.N..MIN%CODE..MAX%CODE(IX)) THEN GO TO A;
227  4              CALL ERROR(0); /* ILLEGAL ENTRY */
228  4              END;
          ELSE
229  3              DO CASE SW%INDEX;
230  4                  IFF%STATUS = IFF%STATUS AND (NOT ON%OFF%MASK); /* OFF%SW */
231  4                  IFF%STATUS = IFF%STATUS OR ON%OFF%MASK; /* ON%SW */
232  4                  RETURN; /* SUBMODE%SW */
233  4                  END;
234  3              END; /* OF DO FOREVER */
235  2          A:
              CALL MOVE(N,..IN%BUF.DATA%ADD(IX));
236  2          END; /* OF CODE%ENTRY */

```

```

SELECT
237 1  STAT$PAGE: PROCEDURE:
238 2  DECLARE STAT$TABLEAU(*) BYTE DATA(
      ROW0,COL7,'IFF STATUS',
      ROW4,' M1 -',COL12,'M4-',
      ROW6,' M2 -',COL12,'M-C-',
      ROW8,' M3A-',COL12,'ANT-',0);
239 2  DECLARE STAT$ENABLE STRUCTURE(
      SW$MASK(7) BYTE,
      SW$VALUE(4) BYTE) DATA(
      0,0,88H,88H,88H,0,8CH,
      03H,14H,25H,60H);
240 2  DECLARE STAT$ICB STRUCTURE(
      MODE BYTE,
      NUM$CH BYTE,
      DEL$CH BYTE,
      DEL$MASK ADDRESS,
      ECHO$ROW BYTE,
      ECHO$COL BYTE,
      SW$ENABLE ADDRESS) DATA(SWITCH,0,0,0,0,0,STAT$ENABLE);

241 2  FORMAT$PAGE: PROCEDURE:

/* FORMAT ROW4 */
242 3  CALL CLEAR$TEMP$BUF;
243 3  CALL MOVE(2,IFF$M1$CODE,,TEMP$BUF(7));
244 3  IF IFF$STATUS THEN TEMP$BUF(9) = 'a';
245 3  IF IFF$M4$MODE THEN TEMP$BUF(15) = 'B';
246 3  ELSE TEMP$BUF(15) = 'A';
247 3  IF (IFF$STATUS AND 8) > 0 THEN TEMP$BUF(19) = 'a';
248 3  CALL UPDATENLINE(4,0,22,,TEMP$BUF);
249 3

/* FORMAT ROW6 */
252 3  CALL CLEAR$TEMP$BUF;
253 3  CALL MOVE(4,IFF$M2$CODE,,TEMP$BUF(5));
254 3  IF (IFF$STATUS AND 2) > 0 THEN TEMP$BUF(9) = 'a';
255 3  CALL MOVE(3,,OFF$CH$TEXT(INDEX(1,IFF$STATUS,4,1)),,TEMP$BUF(16));
256 3  IF (IFF$STATUS AND 10H) > 0 THEN TEMP$BUF(19) = 'a';
257 3  CALL UPDATENLINE(6,0,19,,TEMP$BUF);
258 3

/* FORMAT ROW8 */
260 3  CALL MOVE(4,IFF$M3$CODE,,TEMP$BUF(5));
261 3  IF (IFF$STATUS AND 4) > 0 THEN TEMP$BUF(9) = 'a';
262 3  CALL MOVE(3,,ANT$TEXT(1,IFF$ANTENNA$POSITION),,TEMP$BUF(16));
263 3  CALL UPDATENLINE(8,0,19,,TEMP$BUF);
264 3  END: /* OF FORMAT$PAGE */

```

```

      SELECT
266 2      DO FOREVER:
267 3          CALL CLEAR(0);
268 3          CALL DISPLAY(,STAT$TABLEAU);
269 3          CALL FORMAT$PAGE;
270 3          CALL READ(,STAT$ICB);
271 3          CALL CLEAR$LINE(0);
272 3          DO CASE SM$INDEX:
273 4              CALL CODE$ENTRY: /* M-1 */
274 4              CALL CODE$ENTRY: /* M-2 */
275 4              CALL CODE$ENTRY: /* M-3A */
276 4              CALL M$SELECT;
277 4              CALL MC$ON$OFF;
278 4              CALL ANT$SELECT;
279 4              RETURN: /* SUBMODE */
280 4              END;
281 3          END: /* OF DO FOREVER */
282 2      END: /* OF STAT$PAGE */
```

```

      SELECT
283 1      IFF$SUBMODE: PROCEDURE PUBLIC;
284 2      DECLARE IFF$SUBMODE$TABLEAU(*) BYTE DATA(
          ROW0,'IFF',
          ROW1,'M-1',COL6,'M-2',COL11,'M-3A',COL16,'STAT',
          ROW4,'M-4',COL6,'NORM',COL11,'STBY',COL16,'ANT',
          ROW7,COL6,'RAD',
          ROW8,'M-C',COL6,'TEST',COL11,'TEST',0);
285 2      DECLARE IFF$SWENABLE STRUCTURE(
          SW$MASK(7) BYTE,
          SW$VALUE(8) BYTE) DATA(
          0,0,0A0H,0A0H,0A0H,18H,0,
          01H,23H,34H,55H,66H,78H,9AH,0A0H);
286 2      DECLARE IFF$ICB STRUCTURE(
          MODE BYTE,
          NUM$CH BYTE,
          DEL$M$CH BYTE,
          DEL$M$MASK ADDRESS,
          ECHO$ROW BYTE,
          ECHO$COL BYTE,
          SW$ENABLE ADDRESS) DATA(SWITCH,0,0,0,0,0,0,IFF$SWENABLE);

287 2      DO FOREVER;
288 3          CALL CLEAR(0);
289 3          CALL GRID(,SUBMODE$GRID);
290 3          CALL DISPLAY(,IFF$SUBMODE$TABLEAU);
291 3          CALL DISPLAY$SUBMODE$STATUS$LINE;
292 3          CALL READ(,IFF$ICB);
293 3          DO CASE SW$INDEX;
294 4              CALL CODE$ENTRY; /* M-1 */
295 4              CALL CODE$ENTRY; /* M-2 */
296 4              CALL CODE$ENTRY; /* M-3A */
297 4              CALL STAT$PAGE;
298 4              CALL M$SELECT;
299 4              CALL NORM$STBY;
300 4              CALL ANT$SELECT;
301 4              CALL MC$ON$OFF;
302 4              CALL RAD$TEST;
303 4              CALL TEST;
304 4              CALL IFF$ON$OFF;
305 4              END; /* OF DO CASE */
306 3          END; /* OF DO FOREVER */

307 2      END; /* OF IFF$SUBMODE */

308 1      END; /* OF IFF */

```

MODULE INFORMATION:

```

CODE AREA SIZE      = 0931H   23530
VARIABLE AREA SIZE = 0013H    190
MAXIMUM STACK SIZE = 000EH    140
540 LINES READ

```

PL/M-80 COMPILER IFF

29OCT79 PAGE 14

0 PROGRAM ERROR(S)

END OF PL/M-80 COMPILATION

ISIS-II PL/M-80 V3.1 COMPILATION OF MODULE HWIO
 OBJECT MODULE PLACED IN :F1:ABSHWI.OBJ
 COMPILER INVOKED BY: PLM80 :F1:ABSHWI.SRC DATE(27OCT79) DEBUG

```

1          $INTVECTOR(4,20H) TITLE('HWIO')
2 1        HWIO: DO:
2 1        DECLARE RMBUFA(1536) BYTE EXTERNAL,
          RMBUFB(1536) BYTE EXTERNAL,
          DEBOUNCECOUNT BYTE:

3 1        ADDASMO: PROCEDURE(SMV) EXTERNAL:
4 2        DECLARE SMV BYTE:
5 2        END:

/* THESE INTERRUPTS (0 AND 1) ARE GENERATED BY THE INTERRUPT CONTROLLER
(PIN 8259) AND REQUIRE INTERRUPT VECTORS AS SPECIFIED BY THE INTVECTOR
STATEMENT AT THE BEGINNING OF THIS MODULE.
THESE INTERRUPTS ARE DISTINCT FROM THOSE THAT MAY BE GENERATED BY THE
PROCESSOR WHICH REQUIRE INTERRUPT VECTORS STARTING AT LOCATION 0. */

6 1        DEBOUNCE: PROCEDURE INTERRUPT 0:
/* THIS PROCEDURE STARTS PROGRAMMABLE TIMER 0 WHICH GENERATES THE
INTERUPT THAT CAUSES THE MEMBRANE SWITCH TO BE READ. */

7 2        DEBOUNCECOUNT = 3: /* DELAY MULTIPLIER */
8 2        OUTPUT(10H) = 0: /* LSB */
9 2        OUTPUT(10H) = 0: /* MSB 40 MS. DELAY */
10 2       OUTPUT(0) = 20H: /* SIGNAL END OF INTERRUPT */
11 2       END:

12 1       READMEMBRANE$SMV: PROCEDURE INTERRUPT 1:
13 2       DECLARE (MASK,ROW,COLS,Q) BYTE,
          MEM$Q(*) BYTE DATA(0,0,1,0,2,3,0,4),
          SW$BUF(7) BYTE:

14 2       DEBOUNCECOUNT = DEBOUNCECOUNT - 1:
15 2       IF DEBOUNCECOUNT > 0 THEN DO:
17 3       OUTPUT(10H) = 0: /* START THE TIMER AGAIN */
18 3       OUTPUT(10H) = 0:
19 3       OUTPUT(0) = 20H: /* CLEAR THE INTERRUPT */
20 3       RETURN:
21 3       END:

22 2       MASK = 0FEH:
23 2       DO ROW = 0 TO 6: /* INPUT DATA FOR ALL ROWS WHILE */
24 3       OUTPUT(30H) = MASK: /* THE SWITCH IS DEPRESSED. */
25 3       SW$BUF(ROW) = INPUT(31H):
26 3       MASK = ROL(MASK,1):
27 3       END:

28 2       MASK = 1:
29 2       DO ROW = 0 TO 6:
30 3       COLS = NOT SW$BUF(ROW):
31 3       IF COLS > 0 THEN DO: /* IF THIS IS THE RIGHT ROW THEN ... */

```

```

33 4      Q = - 1;
34 4      DO WHILE COLS > 0; /* ... LOCATE THE CORRECT COLUMN */
35 5          Q = Q + 1;
36 5          COLS = SHR(COLS,1);
37 5      END;
38 4      IF ROW > 4 THEN Q = NEW%Q(Q); /* CORRECTION FOR LAST TWO ROWS */
39 4      CALL ADD$SMQ(ROW*8 + Q); /*ADD SM VALUE TO INPUT QUEUE*/
40 4      OUTPUT(13H) = 30H; /* REENABLE THE TIMER */
41 4      OUTPUT(30H) = 0; /* RESET THE MEMBRANE MASK */
42 4      OUTPUT(0) = 20H; /* SIGNAL END OF INTERRUPT */
43 4      RETURN;
44 4      END; /* OF IF COLS > 0 */
45 3      ELSE MASK = SHL(MASK,1); /* TRY THE NEXT ROW */
46 3      END; /* OF DO ROW */
47 3      OUTPUT(13H) = 30H; /* REENABLE TIMER 0 */
48 2      OUTPUT(30H) = 0; /* RESET THE MEMBRANE MASK */
49 2      OUTPUT(0) = 20H;
50 2

51 2      END; /* OF READ$MEMBRANE$SM */


52 1      SCREEN$INTENSITY: PROCEDURE(LEVEL) PUBLIC;
53 2      DECLARE LEVEL BYTE;

54 2          OUTPUT(32H) = LEVEL;
55 2      END;


56 1      INIT$HARDWARE: PROCEDURE PUBLIC;
57 2          /* INITIALIZE VARIOUS HARDWARE DEVICES */

58 2          DISABLE;
59 2          OUTPUT(33H) = 83H;
60 2          OUTPUT(23H) = 80H;

61 2          OUTPUT(13H) = 30H; /* RESET TIMER 0 TO MODE 0 */
62 2          OUTPUT(13H) = 70H; /* RESET TIMER 1 TO MODE 0 */
63 2          OUTPUT(13H) = 0B0H; /* RESET TIMER 2 TO MODE 0 */

64 2          OUTPUT(30H) = 0; /* LOAD MEMBRANE ENABLE MASK */
65 2          CALL SCREEN$INTENSITY(0); /* BLANK THE SCREEN */

66 2          /* THE FOLLOWING VALUES MUST AGREE WITH THE VALUES USED IN THE
67 2          INTVECTOR STATEMENT AT THE BEGINNING OF THIS MODULE. */
68 2          OUTPUT(0) = 36H; /* INITIALIZE INTERRUPT CONTROLLER - ICM1 */
69 2          OUTPUT(1) = 0; /* INTERVAL = 4, LOCATION - ICM2 */
70 2          OUTPUT(1) = 0FCH; /* OCM1 */
71 2          ENABLE;

72 2      END; /* OF INIT$HARDWARE */


73 1      UPDATE$RM: PROCEDURE PUBLIC;
74 2          /* INITIATE DMA TRANSFERS TO LOAD REFRESH MEMORIES */

75 2          DISABLE;

```

```
72 2      OUTPUT(54H) = LOW(.RMBUFA); /* INITIATE DMA 1 */
73 2      OUTPUT(54H) = HIGH(.RMBUFA);
74 2      OUTPUT(55H) = OFFH; /* LSH OF WORD COUNT */
75 2      OUTPUT(55H) = 85H; /* RSH OF WORD COUNT + 80H */
76 2      OUTPUT(56H) = 4CH; /* OUTPUT MODE TO DMA CONTROLLER */
77 2      OUTPUT(22H) = 4CH; /* TOGGLE DMA RQ2 */
78 2      OUTPUT(22H) = 0;
79 2      CALL TIME(6CH); /* SHORT DELAY */
80 2      OUTPUT(56H) = LOW(.RMBUFB);
81 2      OUTPUT(56H) = HIGH(.RMBUFB);
82 2      OUTPUT(57H) = OFFH;
83 2      OUTPUT(57H) = 85H;
84 2      OUTPUT(58H) = 4CH;
85 2      OUTPUT(22H) = 80H; /* TOGGLE DMA RQ3 */
86 2      OUTPUT(22H) = 0;
87 2      ENABLE;

88 2      END; /* OF UPDATE$RM */

89 1      END; /* OF HWIO */
```

MODULE INFORMATION:

```
CODE AREA SIZE      = 0186H   3900
VARIABLE AREA SIZE  = 0000H   130
MAXIMUM STACK SIZE  = 0004H   100
128 LINES READ
0 PROGRAM ERROR(S)
```

END OF PL/M-80 COMPILATION

ISIS-II PL/M-80 V3.0 COMPILATION OF MODULE FIL
 OBJECT MODULE PLACED IN :F1:FILEBUF.OBJ
 COMPILER INVOKED BY: PLM80 :F1:FILEBUF.SRC DATE(8MAR79)

```

      $TITLE('FILBUF')
1      FIL: DO:
2      1      DECLARE
            RM$BUFA(1536) BYTE EXTERNAL, /* 16*6*12 */
            RM$BUFB(1536) BYTE EXTERNAL,
            CM$FONT$DATA(320) BYTE EXTERNAL:
3      1      DECLARE TRUE LITERALLY 'OFFH',
            FALSE LITERALLY '0':

4      1      FILL$BUF: PROCEDURE(ROW,CM$COL,TEXT$PTR,REF$PTR,NCHAR,MODE) PUBLIC:
5      2      DECLARE ROW BYTE, /* ROW NUMBER: 0 - 8 */
            CM$COL BYTE, /* COLUMN ADDRESS OF FIRST CH: 0 - 23 */
            TEXT$PTR ADDRESS, /* TEXT ADDRESS */
            REF$PTR ADDRESS, /* REFERENCE TEXT ADDRESS */
            NCHAR BYTE, /* NUMBER OF CHS IN TEXT STRING */
            MODE BYTE: /* SEE BELOW */

6      2      DECLARE (TEXT BASED TEXT$PTR)(1) BYTE,
            (REF BASED REF$PTR)(1) BYTE,
            (FONT$PTR,REF$FONT$PTR,BUF$PTR,BUF$OFFSET) ADDRESS,
            (FONT BASED FONT$PTR)(1) BYTE,
            (REF$FONT BASED REF$FONT$PTR)(1) BYTE,
            (BUF BASED BUF$PTR)(1) BYTE,
            (RM$COL,SECTOR,SECTOR$COL,MASK,NOTMASK,N,I,J,K,W,LOC,NEW$MASK,
            REF$LOC,R) BYTE:

```

/* THIS PROCEDURE CONVERTS ASCII TEXT STRINGS TO FONT DATA AND UPDATES
 THE REFRESH MEMORY BUFFERS. THE ROW AND COLUMN VALUES DETERMINE THE
 SCREEN COORDINATES OF THE FIRST CHARACTER. EACH NEW CHARACTER IS
 COMPARED WITH THE CHARACTER IN THE CORRESPONDING LOCATION IN THE
 CURRENT DISPLAY BUFFER (HENCE ON THE SCREEN) TO DETERMINE THE ACTION
 NECESSARY TO UPDATE THE REFRESH MEMORY BUFFERS.
 IF MODE = TRUE BLANK INPUT CHARACTERS ARE IGNORED!
 IF MODE = FALSE BLANK INPUT CHARACTERS ARE PROCESSED.
 THE INPUT TEXT STRING MUST BE 24 CHARACTERS OR LESS (ONE LINE). */

```

SELECT
7 2  NEWSECTOR: PROCEDURE:
8 3      MASK = SHL(MASK,1);
9 3      NOTMASK = NOT MASK;
10 3     SECTOR = SECTOR + 1;
11 3     IF SECTOR < 6 THEN BUFAPTR = .RMSBUFA + BUF$OFFSET;
12 3         ELSE BUFAPTR = .RMSBUFB + BUF$OFFSET;
13 3     IF (SECTOR > 3) AND NEW$MASK THEN DO;
14 4         MASK = 1;
15 4         NOTMASK = OFEH;
16 4         NEW$MASK = FALSE;
17 4     END;
18 3     END; /* OF NEWSECTOR */

21 2  UPDATE: PROCEDURE:
22 3      SECTOR$COL = (SECTOR$COL + 1) AND OFH; /* MODULO 16 */
23 3      IF SECTOR$COL = 0 THEN CALL NEWSECTOR;
24 3      ELSE BUFAPTR = BUFAPTR + 1;
25 3  END; /* OF UPDATE */

27 2  ADDNEW$CH: PROCEDURE:
28 3      FONTAPTR = .CH$FONT$DATA + LOC*5;
29 3      REF$FONTAPTR = .CH$FONT$DATA + REF$LOC*5;
30 3      DO J = 0 TO 4; /* PROCESS 5 CHARACTER FRAGMENTS */
31 4          M = FONT(J);
32 4          R = REF$FONT(J);
33 4          DO K = 0 TO 96 BY 16; /* 7 TIMES */
34 5              IF M THEN BUF(K) = BUF(K) OR MASK;
35 5              ELSE IF R THEN BUF(K) = BUF(K) AND NOTMASK;
36 5              M = SHR(M,1);
37 5              R = SHR(R,1);
38 5          END;
39 4          CALL UPDATE;
40 4      END;
41 3      CALL UPDATE; /* SKIP THE SIXTH CHARACTER FRAGMENT */
42 3  END; /* OF ADDNEW$CH */

45 2  REMOVEOLD$CH: PROCEDURE:
46 3      REF$FONTAPTR = .CH$FONT$DATA + REF$LOC*5;
47 3      DO J = 0 TO 4;
48 4          R = REF$FONT(J);
49 4          DO K = 0 TO 96 BY 16;
50 5              IF R THEN BUF(K) = BUF(K) AND NOTMASK;
51 5              R = SHR(R,1);
52 5          END;
53 4          CALL UPDATE;
54 4      END;
55 3      CALL UPDATE;
56 3  END; /* OF REMOVEOLD$CH */

59 2  SKIP: PROCEDURE:
60 3      SECTOR$COL = SECTOR$COL + 6;
61 3      IF SECTOR$COL > 15 THEN DO;
62 4          SECTOR$COL = SECTOR$COL - 16;
63 4          CALL NEWSECTOR;
64 4          BUFAPTR = BUFAPTR + SECTOR$COL;
65 4      END;

```

```

66 3      ELSE BUF$PTR = BUF$PTR + 6;
67 3      END; /* OF SKIP */

/* BEGIN FILBUF CODE */
68 2      IF (ROW > 8) OR (CH$COL > 23) THEN RETURN; /* ERROR */
69 2      RM$COL = CH$COL*6;
70 2      SECTOR = RM$COL/16;
71 2      SECTOR$COL = RM$COL AND 0FH;
72 2      BUF$OFFSET = 64 + COUNTE(ROW)*160;
73 2      IF SECTOR < 6 THEN DO;
74 3          BUF$PTR = RM$COL*6 + BUF$OFFSET + SECTOR$COL;
75 3          MASK = 1;
76 3          IF SECTOR > 0 THEN MASK = SHL(1,SECTOR);
77 3          NOTMASK = NOT MASK;
78 3          NEWMASK = TRUE;
79 3          END;
80 2      ELSE DO; /* SECTOR > 5 */
81 3          BUF$PTR = RM$COL*6 + BUF$OFFSET + SECTOR$COL;
82 3          NEWMASK = FALSE;
83 3          MASK = ROL(80H,SECTOR - 5);
84 3          NOTMASK = NOT MASK;
85 3          END;
86 2      IF (INCHAR + CH$COL) > 24 THEN N = 23 - CH$COL;
87 2      ELSE N = NCHAR - 1;
88 2      DO I = 0 TO N; /* PROCESS EACH CHARACTER */
89 3          LOC = TEXT(I) - 20H;
90 3          REF$LOC = REF(I) - 20H;
91 3          IF LOC = REF$LOC THEN CALL SKIP; /* CH ALREADY IN BUFFER */
92 3          ELSE DO;
93 4              IF LOC > 0 THEN CALL ADD$NEWCH; /* LOC = 0 => BLANK */
94 4              ELSE DO;
95 5                  IF MODE THEN CALL SKIP;
96 5                  ELSE CALL REMOVE$OLD$CH;
97 5                  END;
98 4              END;
99 3          END; /* OF DO I */
100 2      END; /* OF FILL$BUF */
101 1      END; /* OF COU */

```

MODULE INFORMATION:

```

CODE AREA SIZE      = 0326H    806D
VARIABLE AREA SIZE = 001EH     30D
MAXIMUM STACK SIZE = 0006H     6D
136 LINES READ
0 PROGRAM ERROR(S)

```

END OF PL/M-80 COMPILATION

ISIS-II PL/M-80 V3.0 COMPILATION OF MODULE GRID
 OBJECT MODULE PLACED IN :F1:GRID.OBJ
 COMPILER INVOKED BY: PL/M80 :F1:GRID.SRC DATE(12MAR79) DEBUG

```

      $TITLE('GRID')
1      GRID: DO:
2      1      DECLARE RM$BUFA(1) BYTE EXTERNAL,
           RM$BUFB(1) BYTE EXTERNAL:
3      1      DECLARE HORZ LITERALLY 'OFH',
           VERT LITERALLY 'OFVH':

4      1      VERT$VECT: PROCEDURE(X,Y0,Y1):
           /* GENERATE REFRESH MEMORY DATA FOR VERTICAL VECTOR.
           THE GENERATED DATA IS 'ORED' WITH THE CONTENTS OF THE REFRESH MEMORY
           BUFFERS RM$BUFA AND RM$BUFB. */
5      2      DECLARE (X, /* COLUMN ADDRESS: 0 <= X <= 143 */
           Y0, /* STARTING ROW ADDRESS */
           Y1) /* ENDING ROW ADDRESS: 0 <= Y0 <= Y1 <= 95 */
           BYTE:
6      2      DECLARE (MK,L,N) BYTE,
           (K0,K1,J) ADDRESS:

7      2      IF (X > 143) OR (Y0 > Y1) OR (Y1 > 95) THEN RETURN:
9      2      N = SHR(X,4):
10     2      L = X AND OFH:
11     2      K0 = DOUBLE(Y0)*16 + L:
12     2      K1 = DOUBLE(Y1)*16 + L:
13     2      IF N < 6 THEN
14     2          DO:
15     3              MK = ROL(80H,N + 1):
16     3              DO J = K0 TO K1 BY 16:
17     4                  RM$BUFA(J) = RM$BUFA(J) OR MK:
18     4              END:
19     3          END:
           ELSE
20     2          DO:
21     3              MK = ROL(80H,N - 5):
22     3              DO J = K0 TO K1 BY 16:
23     4                  RM$BUFB(J) = RM$BUFB(J) OR MK:
24     4              END:
25     3          END:

26     2      END: /* OF VERT$VECT */

```

```

SELECT
27 1  HORZ%VECT: PROCEDURE(X0,X1,Y);
    /* GENERATE REFRESH MEMORY DATA FOR HORIZONTAL VECTORS.
    THE GENERATED DATA IS 'CEP' WITH THE CONTENTS OF THE REFRESH MEMORY
    BUFFERS RM%BUFA AND RM%ELFB. */
28 2  DECLARE (X0, /* STARTING COLUMN ADDRESS */
    X1, /* ENDING COLUMN ADDRESS: 0 <= X0 <= X1 <= 143 */
    Y) /* ROW ADDRESS: 0 <= Y <= 95 */
    BYTE;

29 2  DECLARE (NO,N1,L0,L1,MK,MK0,MK1,J) BYTE,
    BUF%PTR ADDRESS,
    (BUF BASED BUF%PTR)(1) BYTE;

30 2  PROC%A: PROCEDURE;
    /* GENERATE HORIZONTAL VECTOR RM DATA FOR MEMORY A, SECTORS 0 - 5. */

31 3  MK0 = 1;
32 3  IF NO > 0 THEN MK0 = SHL(1,NO);
34 3  IF NO = N1 THEN
35 3  DO; /* SHORT VECTOR TOTALLY CONTAINED IN ONE SEGMENT */
36 4  DO J = L0 TO L1;
37 5  BUF(J) = BUF(J) OR MK0;
38 5  END;
39 4  RETURN;
40 4  END;

    /* N1 > NO - VECTOR EXTENDS MORE THAN ONE SEGMENT */
41 3  DO J = L0 TO 15; /* LEFT-MOST PORTION */
42 4  BUF(J) = BUF(J) OR MK0;
43 4  END;

44 3  IF N1 > 6 THEN MK1 = 40H;
46 3  ELSE MK1 = SHL(1,N1);

47 3  IF (N1 - NO) > 1 THEN
48 3  DO; /* MIDDLE PORTION */
49 4  MK = MK1 - SHL(MK0,1);
50 4  DO J = 0 TO 15;
51 5  BUF(J) = BUF(J) OR MK;
52 5  END;
53 4  END;

54 3  IF N1 < 6 THEN
55 3  DO J = 0 TO L1; /* RIGHT-MOST PORTION */
56 4  BUF(J) = BUF(J) OR MK1;
57 4  END;

58 3  END; /* OF PROC%A */

```

```
REJECT
59 2  PROC98: PROCEDURE;
    /* GENERATE HORIZONTAL VECTOR RN DATA FOR MEMORY B, SECTORS 6 - 8. */

60 3      MK0 = 1;
61 3      IF NO > 6 THEN MK0 = SHL(1, NO - 6);
63 3      IF NO > 5 THEN
64 3          DO;
65 4              IF NO = N1 THEN
66 4                  DO; /* SHORT VECTOR TOTALLY CONTAINED IN ONE SEGMENT */
67 5                      DO J = L0 TO L1;
68 6                          BUF(J) = BUF(J) OR MK0;
69 6                      END;
70 5                      RETURN;
71 5                      END;
    /* N1 > NO */
72 4          DO J = L0 TO 15; /* INITIAL PORTION OF VECTOR */
73 5              BUF(J) = BUF(J) OR MK0;
74 5              END;
75 4          END;

76 3      MK1 = ROL(80H, N1 - 5);
77 3      IF (N1 - NO) > 1 THEN
78 3          DO;
79 4              MK = MK1 - 1;
80 4              DO J = 0 TO 15; /* MIDDLE PORTION */
81 5                  BUF(J) = BUF(J) OR MK;
82 5                  END;
83 4              END;

84 3      DO J = 0 TO L1; /* END PORTION */
85 4          BUF(J) = BUF(J) OR MK1;
86 4          END;

87 3      END; /* OF PROC98 */
```

```
SEJECT
/* BEGIN HORIZVECT PROCESSING */

88 2      IF (X0 > X1) OR (X1 > 143) OR (Y > 95) THEN RETURN;
90 2      NO = SHR(X0,4); /* STARTING SECTOR INDEX: 0 - 8 */
91 2      N1 = SHR(X1,4); /* ENDING SECTOR INDEX */
92 2      LO = X0 AND 0FH;
93 2      LI = X1 AND 0FH;
94 2      IF NO < 6 THEN
95 2          DO;
96 3          BUF&PTR = .RHS&BUFA + DOUBLE(Y)*16;
97 3          CALL PROC&A;
98 3          END;
99 2      IF N1 > 5 THEN
100 2          DO;
101 3          BUF&PTR = .RHS&BUFB + DOUBLE(Y)*16;
102 3          CALL PROC&B;
103 3          END;

104 2      END; /* OR HORIZVECT */
```

```

SELECT
105 1  GRID: PROCEDURE(LIST*PTR) PUBLIC;
      /* GENERATE RM DATA FOR A GRID DEFINED BY A LIST OF HORIZONTAL
      AND VERTICAL VECTORS. THE LIST HAS THE FOLLOWING FORMAT:
      GRID$LIST := ((DIR, START, END, LOC1, LOC2, ..., LOCN), OFFH)

      1. IF DIR = HORIZ THEN START = X0; END = X1; LOC1 = Y1, 1 <= I <= N.
      THIS DEFINES N HORIZONTAL VECTORS AT COORDINATE POSITION
      (X0,Y1-X1,Y1), (X0,Y2-X1,Y2), ..., (X0,YN-X1,YN).

      2. IF DIR = VERT THEN START = Y0; END = Y1; LOC1 = X1, 1 <= I <= N.
      THIS DEFINES N VERTICAL VECTORS AT COORDINATE POSITIONS
      (X1,Y0-X1,Y1), (X2,Y0-X2,Y1), ..., (XN,Y0-XN,Y1).

      THE INNER SUBLIST MAY BE REPEATED AS MANY TIMES AS NECESSARY.
      THE LIST TERMINATOR (OFFH) IS REQUIRED. */

106 2  DECLARE LIST*PTR ADDRESS,
      (LIST BASED LIST*PTR)(1) BYTE,
      (CH,I,X0,X1,Y0,Y1) BYTE;

107 2  CH = LIST(0);
108 2  I = 0;
109 2  DO WHILE CH < OFFH;
110 3  IF CH = HORIZ THEN
111 3  DO;
112 4  X0 = LIST(I + 1);
113 4  X1 = LIST(I + 2);
114 4  CH = LIST(I + 3);
115 4  I = I + 3;
116 4  DO WHILE CH < OFFH;
117 5  CALL HORIZ$VECT(X0,X1,CH);
118 5  I = I + 1;
119 5  CH = LIST(I);
120 5  END;
121 4  END;
      ELSE
122 3  DO;
123 4  IF CH = VERT THEN
124 4  DO;
125 5  Y0 = LIST(I + 1);
126 5  Y1 = LIST(I + 2);
127 5  CH = LIST(I + 3);
128 5  I = I + 3;
129 5  DO WHILE CH < OFFH;
130 6  CALL VERT$VECT(CH,Y0,Y1);
131 6  I = I + 1;
132 6  CH = LIST(I);
133 6  END;
134 5  END;
135 4  ELSE RETURN; /* ERROR */
136 4  END;
137 3  END; /* OF DO WHILE CH < OFFH */

138 2  END; /* OF GENGRID */

139 1  END; /* OF COUGRD: DO: */

```

PL/M-80 COMPILER GRID

12MAR79 PAGE 6

MODULE INFORMATION:

CODE AREA SIZE	= 0453H	1107D
VARIABLE AREA SIZE	= 0021H	33D
MAXIMUM STACK SIZE	= 0006H	6D
203 LINES READ		
0 PROGRAM ERROR(S)		

END OF PL/M-80 COMPILATION

ISIS-II PL/M-80 V3.1 COMPILATION OF MODULE DNV
 OBJECT MODULE PLACED IN :F1:DNV.OBJ
 COMPILER INVOKED BY: PLM30 :F1:DNV.SRC DATE(270CT79) DEBUG

```

      $TITLE('DNV')
      /* INIT$DNV, DISPLAY$ACTIVE$WAYPT, DISPLAY$PRESENT$POSITION, DNV$SUBNODE, DNV$NAV$STATUS */
1      DNV: DO:
2      1      DECLARE HORZ LITERALLY 'OFCH',
              VERT LITERALLY 'OFCH';
      $MOLIST INCLUDE(:F1:CDULIT.SRC)
5      1      DNV$FLY$TO: PROCEDURE EXTERNAL;
6      2      END;
7      1      DNV$BACKUP$NAV: PROCEDURE EXTERNAL;
8      2      END;
9      1      DNV$WAYPT$STATUS: PROCEDURE(TYPE) EXTERNAL;
10     2      DECLARE TYPE BYTE;
11     2      END;
12     1      DNV$CKPT: PROCEDURE EXTERNAL;
13     2      END;
14     1      FORMAT$WAYPT$COORD: PROCEDURE(WAYPT$ADD) EXTERNAL;
15     2      DECLARE WAYPT$ADD ADDRESS;
16     2      END;
17     1      PACK$WAYPT: PROCEDURE(NBYTES, SOURCE$PTR, DEST$PTR) EXTERNAL;
18     2      DECLARE NBYTES BYTE, (SOURCE$PTR, DEST$PTR) ADDRESS;
19     2      END;
20     1      UNPACK$WAYPT: PROCEDURE(NBYTES, SOURCE$PTR, DEST$PTR) EXTERNAL;
21     2      DECLARE NBYTES BYTE, (SOURCE$PTR, DEST$PTR) ADDRESS;
22     2      END;
23     1      DNV$TGT: PROCEDURE EXTERNAL;
24     2      END;
25     1      DNV$UPDATE: PROCEDURE EXTERNAL;
26     2      END;
27     1      CLEAR: PROCEDURE(START$ROW) EXTERNAL;
28     2      DECLARE START$ROW BYTE;
29     2      END;
30     1      CLEAR$TEMP$BUF: PROCEDURE EXTERNAL;
31     2      END;
32     1      INSERT: PROCEDURE(NCHAR, SOURCE$PTR, DEST$PTR, DEL$CH, DEL$MASK) EXTERNAL;
33     2      DECLARE (NCHAR, DEL$CH) BYTE, (SOURCE$PTR, DEST$PTR, DEL$MASK) ADDRESS;
34     2      END;
35     1      GRID: PROCEDURE(POINTER) EXTERNAL;
36     2      DECLARE POINTER ADDRESS;
37     2      END;
38     1      DISPLAY: PROCEDURE(IN$PTR) EXTERNAL;
39     2      DECLARE IN$PTR ADDRESS;
40     2      END;
41     1      READ: PROCEDURE(ICB$PTR) EXTERNAL;
42     2      DECLARE ICB$PTR ADDRESS;
43     2      END;
44     1      UPDATE$LINE: PROCEDURE(ROW, COL, NCHAR, TEXT$PTR) EXTERNAL;
45     2      DECLARE (ROW, COL, NCHAR) BYTE, TEXT$PTR ADDRESS;
46     2      END;

      /* PUBLIC VARIABLES USED BY DNV PROCEDURES */
47     1      DECLARE ZERO(1) BYTE PUBLIC DATA('000');

```

```

48 1  DECLARE SUBMODE$STEP$SWENABLE STRUCTURE(
      SWMASK(7) BYTE,
      SW$VALUE(1) BYTE) PUBLIC DATA(0,0,0,0,0,0,0,0,0,0);
49 1  DECLARE SUBMODE$SWENABLE STRUCTURE(
      SWMASK(7) BYTE,
      SW$VALUE(1) BYTE) PUBLIC DATA(0,0,0,0,0,0,0,0,0,0);

50 1  DECLARE SUBMODE$ICB STRUCTURE(
      MODE BYTE, NUM$CH BYTE, DELM$CH BYTE, DELM$MASK ADDRESS,
      ECHO$ROW BYTE, ECHO$COL BYTE, SW$ENABLE ADDRESS) DATA(
      SWITCH,0,0,0,0,0,0, SUBMODE$SWENABLE);
51 1  DECLARE SINGLE$DIGIT$ICB STRUCTURE(
      MODE BYTE,
      NUM$CH BYTE,
      DELM$CH BYTE,
      DELM$MASK ADDRESS,
      ECHO$ROW BYTE,
      ECHO$COL BYTE,
      SW$ENABLE ADDRESS) PUBLIC DATA(DIGIT,1,0,0,2,5, SUBMODE$SWENABLE);
52 1  DECLARE SUB$SUBMODE$GRID(*) BYTE PUBLIC DATA(
      HORZ,0,92,22,32,
      HORZ,2,92,52,72,92,
      VERT,32,92,2,32,62,
      VERT,22,92,92,OFFH);

/* EXTERNAL VARIABLES */
53 1  DECLARE
      OFF$ON$TEXT(2) STRUCTURE(CH(3) BYTE) EXTERNAL,
      BLANK$LINE(24) BYTE EXTERNAL,
      TEMP$BUF(24) BYTE EXTERNAL,
      SW$ BYTE EXTERNAL,
      SW$INDEX BYTE EXTERNAL,
      SUBMODE$GRID ADDRESS EXTERNAL,
      DNV$GROUND$SPEED BYTE EXTERNAL,
      DNV$RANGE BYTE EXTERNAL,
      DNV$TRACK$ANGLE$ERROR BYTE EXTERNAL,
      DNV$BEARING BYTE EXTERNAL,
      DNV$TRACK$ANGLE BYTE EXTERNAL,
      DNV$CROSS$TRACK$ANGLE BYTE EXTERNAL,
      DNV$TIME$TO$GO BYTE EXTERNAL,
      DNV$WIND$SPEED(1) BYTE EXTERNAL,
      DNV$WIND$DIR BYTE EXTERNAL,
      DNV$DESTINATION BYTE EXTERNAL,
      DNV$STATUS BYTE EXTERNAL,
      DNV$TEMP$WAYPT STRUCTURE(
        STATUS BYTE,
        UTM$SPHEROID BYTE,
        UTM$ZONE(2) BYTE,
        UTM(11) BYTE,
        LAT(6) BYTE,
        LON(7) BYTE,
        MAGVAR(6) BYTE) EXTERNAL,
      DNV$LAST$WAYPT$PTR ADDRESS EXTERNAL,
      DNV$CKPT$DATA(10) STRUCTURE(DIGITS(68) BYTE) EXTERNAL,
      DNV$GTG$DATA(10) STRUCTURE(DIGITS(68) BYTE) EXTERNAL;

```

```

SELECT
54 1  INIT$DNV: PROCEDURE PUBLIC;
55 2  DECLARE I BYTE;
      ZERO BYTE DATA(0);

56 2      DNV$LAST$WAYPT$PTR = .DNV$CKPT$DATA; /* PRESENT POSITION */
57 2      DNV$STATUS = 0;
58 2      DNV$DESTINATION = 0;
/* CLEAR ALL TARGETS AND CHECKPOINTS; IE, SET WAYPT.STATUS = 0 */
59 2      DO I = 0 TO 9;
60 3          CALL PACK$WAYPT(1,.ZERO,.DNV$CKPT$DATA(I));
61 3          CALL PACK$WAYPT(1,.ZERO,.DNV$TGT$DATA(I));
62 3          END;
63 2      DNV$WIND$SPEED(0) = ' ';
64 2      CALL MOVE(26,.DNV$WIND$SPEED,.DNV$WIND$SPEED(1));
/* INITIALIZE VARIABLES APPEARING ON NAV STATUS TABLEAU */
65 2      CALL MOVE(4,('163'),.DNV$RANGE);
66 2      CALL MOVE(3,('104'),.DNV$TRACK$ANGLE$ERROR);
67 2      CALL MOVE(3,('273'),.DNV$BEARING);
68 2      CALL MOVE(3,('205'),.DNV$CROSS$TRACK$ANGLE);
69 2      CALL MOVE(3,('200'),.DNV$TIME$TO$GO);
/* INITIALIZE PRESENT POSITION */
70 2      DNV$TEMP$WAYPT.STATUS = 6; /* L/L AND UTM DATA AVAILABLE */
71 2      CALL MOVE(13,('N34112N118254'),.DNV$TEMP$WAYPT.LAT);
72 2      DNV$TEMP$WAYPT.UTM$SPHEROID = 11;
73 2      CALL MOVE(13,('01ABC12345678'),.DNV$TEMP$WAYPT.UTM$ZONE);
74 2      CALL PACK$WAYPT(34,.DNV$TEMP$WAYPT,.DNV$CKPT$DATA); /* STORE INTO CKPT(0) */
75 2      END; /* OF INIT$DNV */

76 1  DISPLAY$ACTIVE$WAYPT: PROCEDURE PUBLIC;
77 2  DECLARE WAYPT$DESCRIPTOR(2) STRUCTURE(TEXT(4) BYTE) DATA('CKPT TGT');
78 2  DECLARE I BYTE;

79 2      CALL CLEAR$TEMP$BUF;
80 2      CALL MOVE(4,.WAYPT$DESCRIPTOR(DNV$DESTINATION AND 1),.TEMP$BUF);
81 2      TEMP$BUF(5) = (SHR(DNV$DESTINATION,1) AND 0FH) + 30H;
82 2      CALL UPDATE$LINE(0,18,6,.TEMP$BUF);
83 2      END; /* OF DISPLAY$ACTIVE$WAYPT */

84 1  DISPLAY$PRESENT$POSITION: PROCEDURE(ROW) PUBLIC;
85 2  DECLARE ROW BYTE;

86 2      CALL CLEAR$TEMP$BUF;
87 2      CALL UNPACK$WAYPT(34,.DNV$CKPT$DATA(0),.DNV$TEMP$WAYPT);
88 2      CALL FORMAT$WAYPT$COORD(.DNV$TEMP$WAYPT);
89 2      TEMP$BUF(0),TEMP$BUF(1) = 'P';
90 2      CALL UPDATE$LINE(ROW,0,24,.TEMP$BUF);
91 2      END;

```

```

SELECT
92 1  DNV%SUBMODE: PROCEDURE PUBLIC:

93 2  DECLARE DNV%SUBMODE$TABLEAU(*) BYTE DATA(
      ROM0,'DNV',
      ROM3,' FLY',COL6,'CKPT BKUP NAV',
      ROM4,' TO',COL12,'NAV STAT',
      ROM5,' NEXT',COL16,'CKPT',
      ROM6,' CKPT',COL6,'UTM',COL12,'L/L STAT',
      ROM7,' TGT',COL7,'LP',COL11,'TEST TGT',
      ROM8,COL6,'DATE',COL15,'STAT',0);

94 2  DECLARE DNV%SM$ENABLE STRUCTURE(
      SM$MASK(7) BYTE,
      SM$VALUE(9) BYTE) DATA(
      0,0,0A0H,0E0H,0A0H,10H,0,
      01H,23H,34H,45H,67H,78H,9AH,0BBH,0CDH);

95 2  DECLARE DNV%ICB STRUCTURE(
      MODE BYTE, NUM$CH BYTE, DELM$CH BYTE, DELM$MASK ADDRESS,
      ECHO$ROM BYTE, ECHO$COL BYTE, SM$ENABLE ADDRESS) DATA(
      SWITCH,0,0,0,0,0,0,0,DNV%SM$ENABLE);

96 2  DNV%NAV$STATUS: PROCEDURE:
97 3  DECLARE STATUS$TABLEAU(*) BYTE DATA(
      ROM0,'DNV',COL9,'STATUS',
      ROM4,'RGE',COL9,'GSPD',COL17,'TKE',
      ROM6,'BRG',COL9,'TRK',COL17,'XTKE',
      ROM8,'TTB',COL9,'WIND',COL16,'/',0);

98 3  DECLARE DATA$ADD(9) ADDRESS DATA(
      .DNV$RANGE,
      .DNV$GROUND$SPEED,
      .DNV$TRACK$ANGLE$ERROR,
      .DNV$BEARING,
      .DNV$TRACK$ANGLE,
      .DNV$CROSS$TRACK$ANGLE,
      .DNV$TIME$TO$GO,
      .DNV$WIND$SPEED,
      .DNV$WIND$DIR);

99 3  DECLARE NCHAR(9) BYTE DATA(4,3,3,3,3,3,3,2,3);
/* THE FOLLOWING ARE THE LEFT BYTES OF THE INSERT MASK */
100 3  DECLARE MSB$INSERT$MASK(9) BYTE DATA(10H,0,0,0,0,20H,0,0,0);
101 3  DECLARE MASK ADDRESS,
      MASK$OVERLAY(2) BYTE AT(.MASK);
102 3  DECLARE DEST$LOC(9) ADDRESS DATA(
      .TEMP$BUF+3,.TEMP$BUF+13,.TEMP$BUF+20,
      .TEMP$BUF+3,.TEMP$BUF+12,.TEMP$BUF+20,
      .TEMP$BUF+3,.TEMP$BUF+17,.TEMP$BUF+13);
103 3  DECLARE (I,J,IX) BYTE:

104 3  CALL CLEAR(2);
105 3  CALL DISPLAY(.STATUS$TABLEAU);
106 3  CALL DISPLAY$PRESENT$POSITION(2);
107 3  MASK$OVERLAY(0) = 0;
108 3  DO I = 0 TO 2;

```

```
109 4      CALL CLEAR$TEMP$BUF;
110 4      DO J = 0 TO 2;
111 5          IX = I*3 + J;
112 5          MASK$OVERLAY(1) = MSB$INSERT$MASK(IX);
113 5          CALL INSERT(NCHAR(IX),DATA$ACC(IX),DEST$LOC(IX),
                        ' ',MSB$INSERT$MASK(IX));
114 5          END;
115 4      CALL UPDATE$LINE(I*2+4,0,24,TEMP$BUF);
116 4      END;
117 3      CALL READ(.SUBNODE$ICB); /* WAIT FOR SUBNODE SW ENTRY */

118 3      END; /* OF DNV$WWW$STAT */
```

```

REJECT
/* BEGIN DNV SUBMODE */
119 2    DO FOREVER:
120 3        CALL CLEAR(0);
121 3        CALL UPDATELINE(1,0,3,OFF$ORTEXT(SHR(DNV$STATUS,4)));
122 3        CALL GRID(.SUBMODE$GRID);
123 3        CALL DISPLAY(.DNV$SEQUENCE$TABLEAU);
124 3        CALL DISPLAYACTIVE$MAYPT;
125 3        CALL DISPLAY$PRESENT$POSITION(2);
126 3        CALL READ(.DNV$ICB);
127 3        DO CASE $N$INDEX:
128 4            CALL DNV$FLY$TO;
129 4            CALL DNV$CKPT;
130 4            CALL DNV$BACKUP$NAV;
131 4            CALL DNV$NAV$STATUS;
132 4            DNV$DESTINATION = (DNV$DESTINATION AND 1) OR
                                SHL((SHR(DNV$DESTINATION,1) + 1) MOD 10,1);
133 4            DNV$STATUS = DNV$STATUS OR 1; /* UTH */
134 4            DNV$STATUS = DNV$STATUS AND OFEH; /* L/L */
135 4            CALL DNV$MAYPT$STATUS(CKPT); /* CKPT STATUS */
136 4            CALL DNV$TGT;
137 4            CALL DNV$UPDATE;
138 4            ; /* DNV TEST */
139 4            CALL DNV$MAYPT$STATUS(TGT); /* TGT STATUS */
140 4            DNV$STATUS = DNV$STATUS AND OFEH; /* OFF */
141 4            DNV$STATUS = DNV$STATUS OR 10H; /* ON */
142 4            ENH; /* OF DO CASE */
143 3        ENH; /* OF DO FOREVER */
144 2    ENH; /* OF DNV SUBMODE */

145 1    ENH; /* OF DNV */

```

MODULE INFORMATION:

```

CODE AREA SIZE   = 0464H 11240
VARIABLE AREA SIZE = 0003H 80
MAXIMUM STACK SIZE = 000AH 100
315 LINES READ
0 PROGRAM ERROR(S)

```

END OF PL/M-80 COMPILATION

ISIS-II PL/M-80 V2.0 COMPILATION OF MODULE DNV1
 OBJECT MODULE PLACED IN :F1:DNV1.OBJ
 COMPILER INVOKED BY: PLM80 :F1:DNV1.SRC DATE(9MAR79) DEBUG

```

      $TITLE('DNV1')
      /* DNV%KPT, DNV%TOT, DNV%FLY%TD */
1      DNV1: DO:
      $INCLUDE(:F1:CDULIT.SRC)
4      1      DNV%COORD$DATA$ENTRY: PROCEDURE(WAYPT$PTR,WAYPT$TYPE) BYTE EXTERNAL;
5      2      DECLARE WAYPT$PTR ADDRESS, WAYPT$TYPE BYTE;
6      2      END;
7      1      CLEAR: PROCEDURE(START$ROW) EXTERNAL;
8      2      DECLARE START$ROW BYTE;
9      2      END;
10     1      GRID: PROCEDURE(POINTER) EXTERNAL;
11     2      DECLARE POINTER ADDRESS;
12     2      END;
13     1      DISPLAY: PROCEDURE(IN$PTR) EXTERNAL;
14     2      DECLARE IN$PTR ADDRESS;
15     2      END;
16     1      DISPLAY$DIGIT$KB: PROCEDURE EXTERNAL;
17     2      END;
18     1      READ: PROCEDURE(ICB$PTR) EXTERNAL;
19     2      DECLARE ICB$PTR ADDRESS;
20     2      END;
21     1      UPDATE$LINE: PROCEDURE(ROW,COL,NCHAR,TEXT$PTR) EXTERNAL;
22     2      DECLARE (ROW,COL,NCHAR) BYTE, TEXT$PTR ADDRESS;
23     2      END;
24     1      ERROR: PROCEDURE(CODE) EXTERNAL;
25     2      DECLARE CODE BYTE;
26     2      END;
27     1      PACK$WAYPT: PROCEDURE(NBYTES,SOURCE$PTR,DEST$PTR) EXTERNAL;
28     2      DECLARE NBYTES BYTE, (SOURCE$PTR,DEST$PTR) ADDRESS;
29     2      END;
30     1      UNPACK$WAYPT: PROCEDURE(NBYTES,SOURCE$PTR,DEST$PTR) EXTERNAL;
31     2      DECLARE NBYTES BYTE, (SOURCE$PTR,DEST$PTR) ADDRESS;
32     2      END;

      /* EXTERNAL VARIABLES */
33     1      DECLARE
          BLANK$LINE(24) BYTE EXTERNAL,
          SMV BYTE EXTERNAL,
          SM$INDEX BYTE EXTERNAL,
          DATA$ENTERED BYTE EXTERNAL,
          IN$BUF(1) BYTE EXTERNAL,
          SINGLE$DIGIT$KB ADDRESS EXTERNAL,
          SUBMODE$GRID BYTE EXTERNAL,
          SUB$SUBMODE$GRID BYTE EXTERNAL,
          SUBMODE$SW$ENABLE BYTE EXTERNAL,

          DNV$TEMP$WAYPT STRUCTURE(
              STATUS BYTE,
              UTM(14) BYTE,
              LAT(6) BYTE,
              LON(7) BYTE,

```

MACVAR(6) BYTE) EXTERNAL.

DNV\$DESTINATION BYTE EXTERNAL.

DNV\$LAST\$WAYPT\$PTR ADDRESS EXTERNAL.

DNV\$CKPT\$DATA(10) SINGLE DIGITS(68) BYTE) EXTERNAL.

DNV\$GT\$DATA(10) STRUCTURE(DIGITS(68) BYTE) EXTERNAL.

```

34 1  DNV$CKPT: PROCEDURE PUBLIC;
35 2  DECLARE CKPT$TABLEAU(*) BYTE DATA(
      ROW2,'CKPT',
      ROW3,COL6,'CKPT',0);
36 2  DECLARE CKPT$NUM BYTE;

37 2      CALL CLEAR(2);
38 2      CALL DISPLAY$DIGIT$KB;
39 2      CALL DISPLAY(,CKPT$TABLEAU);
40 2      CALL READ(,SINGLE$DIGIT$ICB);
41 2      IF SW = SUBMODE$SW THEN RETURN;
42 2      IF DATA$ENTERED THEN DO;
43 3          CKPT$NUM = IN$BUF(0) ~ 30H;
44 3          IF DNV$COORD$DATA$ENTRY(,DNV$TEMP$WAYPT,CKPT) = 0 THEN RETURN; /* SUBMODE RETURN */
45 3          CALL PACK$WAYPT(34, ,DNV$TEMP$WAYPT, ,DNV$CKPT$DATA(CKPT$NUM));
46 3          DNV$LAST$WAYPT$PTR = ,DNV$CKPT$DATA(CKPT$NUM);
47 3          END;
48 2  END; /* OF DNV$CKPT */

```

```

SELECT
52 1  DNV$TGT: PROCEDURE PUBLIC;
53 2  DECLARE TGT$TABLEAU(*) BYTE DATA(
      ROW2,'TGT',
      ROW4,'FRZE',
      ROW7,'TGT',0);
54 2  DECLARE FRZE$SUBMODE$SW$ENABLE STRUCTURE(
      SW$MASK(7) BYTE,
      SW$VALUE(1) BYTE) DATA(
      0,0,80H,0,0,0,80H,
      01H);
55 2  DECLARE TGT$ICB STRUCTURE(
      MODE BYTE, NUM$CH BYTE, DELM$CH BYTE, DELM$MASK ADDRESS,
      ECHO$ROW BYTE, ROW$COL BYTE, SW$ENABLE ADDRESS) DATA(
      DIGIT,1,0,0,2,5,.FRZE$SUBMODE$SW$ENABLE);
56 2  DECLARE FRZE$TAB(*) BYTE DATA(ROW2,'FRZE',0);
57 2  DECLARE TGT$ICB2 STRUCTURE(
      MODE BYTE, NUM$CH BYTE, DELM$CH BYTE, DELM$MASK ADDRESS,
      ECHO$ROW BYTE, ECHO$COL BYTE, SW$ENABLE ADDRESS) DATA(
      DIGIT,1,0,0,2,5,.SUBMODE$SW$ENABLE);
58 2  DECLARE TGT$NUM BYTE,
      I BYTE;
59 2  DECLARE FRZE$SW LITERALLY '200';

60 2  A:
      CALL CLEAR(2);
      CALL DISPLAY$DIGITS$B;
      CALL DISPLAY(.TGT$TABLEAU);
      CALL READ(.TGT$ICB);
      IF SW = SUBMODE$SW THEN RETURN;
      IF SW = FRZE$SW THEN GO TO B;
      IF DATA$ENTERED THEN DO:
      70 3    TGT$NUM = IN$BUF(0) - 30H;
      71 3    IF DNV$COORD$DATA$ENTRY(.DNV$TEMP$WAYPT,TGT) = 0 THEN RETURN; /* SUBMODE RETURN */
      73 3    CALL PACK$WAYPT(34,.DNV$TEMP$WAYPT,.DNV$TGT$DATA(TGT$NUM));
      74 3    DNV$LAST$WAYPT$PTR = .DNV$TGT$DATA(TGT$NUM);
      75 3    RETURN;
      76 3    END;
      ELSE DO:
      77 2    CALL ERROR(0);
      78 3    GO TO A;
      79 3    END;
      80 3    END;
      81 2  B:
      /* FRZE SWITCH PRESSED, RECORD PRESENT POSITION. */
      CALL DISPLAY(.FRZE$TAB);
      CALL READ(.TGT$ICB2);
      IF SW = SUBMODE$SW THEN RETURN;
      IF SW = ENTER$SW THEN DO:
      87 3    IF DATA$ENTERED THEN TGT$NUM = IN$BUF(0) - 30H;
      89 3    ELSE DO:
      90 4      DO TGT$NUM = 6 TO 9;
      91 5      CALL UNPACK$WAYPT(1,.DNV$TGT$DATA(TGT$NUM),.DNV$TEMP$WAYPT);
      92 5      IF (.DNV$TEMP$WAYPT.STATUS AND 6) = 0 THEN GO TO C;
      94 5      END;
      95 4      CALL ERROR(1); /* NO UNALLOCATED TARGETS */
      96 4      GO TO A;

```

```
97 4      END:    /* OF ELSE DO */
98 3      C:
      /* COPY PRESENT POSITION TO DESIGNATED TARGET */
      CALL UNPACK$WAYPT(34, DNV$CAPT$DATA(1), DNV$TEMP$WAYPT);
      CALL PACK$WAYPT(34, DNV$TEMP$WAYPT, DNV$GT$DATA(TCISNUM));
      END:
      /* DISPLAY COORDINATES OF TARGET AND TARGET NUMBER */

101 2      END:    /* OF DNV$GT */
```

```

      $EJECT
102 1  DNV$FLY$TO: PROCEDURE PUBLIC;
103 2  DECLARE FLY$TO$TABLEAU(*) BYTE DATA(
      ROW3,' FLY',
      ROW4,' TO',0);
104 2  DECLARE TYPE(2) STRUCTURE(CH(7) BYTE) DATA(
      ROW3,COL6,'CKPT',0,
      ROW7,COL1,' TGT',0);
105 2  DECLARE TOT(*) BYTE DATA(ROW2,' TO',0);
106 2  DECLARE FLY$TO$ENABLE STRUCTURE(
      SW$MASK(7) BYTE,
      SW$VALUE(2) BYTE) DATA(
      0,0,20H,0,80H,0,80H, /$CKPT,TGT,SUBMODE $/
      01H,20H);
107 2  DECLARE FLY$TO$ICB STRUCTURE(
      MODE BYTE, NUM$CH BYTE, DELM$CH BYTE, DELM$MASK ADDRESS,
      ECHO$ROW BYTE, ECHO$COL BYTE, SW$ENABLE ADDRESS) DATA(
      SWITCH,0,0,0,0,0,FLY$TO$ENABLE);

108 2  CALL CLEAR(2);
109 2  CALL DISPLAY(.FLY$TO$TABLEAU);
110 2  CALL DISPLAY(.TYPE(0));
111 2  CALL DISPLAY(.TYPE(1));
112 2  CALL GRID(.SUB$SUBMODE$GRID);
113 2  CALL READ(.FLY$TO$ICB);
114 2  IF SW = SUBMODE$SW THEN RETURN;
116 2  DNV$DESTINATION = SW$INDEX;
117 2  CALL CLEAR(2);
118 2  CALL DISPLAY(.TOT);
119 2  CALL DISPLAY(.FLY$TO$TABLEAU);
120 2  CALL DISPLAY(.TYPE(SW$INDEX));
121 2  CALL DISPLAY$DIGIT$KB;
122 2  CALL READ(.SINGLE$DIGIT$ICB);
123 2  IF DATA$ENTERED AND (SW = ENTER$SW) THEN
124 2  DNV$DESTINATION = DNV$DESTINATION OR SHL(IN$BUF(0)-30H,1);
125 2  END; /* OF DNV$FLY$TO */

126 1  END; /* OF DNV1 */

```

MODULE INFORMATION:

```

CODE AREA SIZE   = 0263H   611D
VARIABLE AREA SIZE = 0003H   3D
MAXIMUM STACK SIZE = 0004H   4D
245 LINES READ
0 PROGRAM ERROR(S)

```

END OF PL/M-80 COMPILATION

ISIS-II PL/M-80 V3.0 COMPILATION OF MODULE DNV2

OBJECT MODULE PLACED IN :F1:DNV2.OBJ

COMPILER INVOKED BY: PLM30 :F1:DNV2.SRC DATE(12MAR79) DEBUG

```

      $TITLE('DNV2')
      /* DISPLAY$WAYPT$TYPE, LAT$LON$DATA$ENTRY, MAGVAR$DATA$ENTRY, DNV$COORD$DATA$ENTRY */
1      DNV2: DO:
      $NOLIST INCLUDE(:F1:CDULIT.SRC)
4      1      CLEAR: PROCEDURE(START$ROW) EXTERNAL;
5      2      DECLARE START$ROW BYTE;
6      2      END;
7      1      ERROR: PROCEDURE(CODE) EXTERNAL;
8      2      DECLARE CODE BYTE;
9      2      END;
10     1      GRID: PROCEDURE(POINTER) EXTERNAL;
11     2      DECLARE POINTER ADDRESS;
12     2      END;
13     1      DISPLAY: PROCEDURE(IN$PTR) EXTERNAL;
14     2      DECLARE IN$PTR ADDRESS;
15     2      END;
16     1      READ: PROCEDURE(ICB$PTR) EXTERNAL;
17     2      DECLARE ICB$PTR ADDRESS;
18     2      END;
19     1      UPDATE$LINE: PROCEDURE(ROW,COL,NCHAR,TEXT$PTR) EXTERNAL;
20     2      DECLARE (ROW,COL,NCHAR) BYTE, TEXT$PTR ADDRESS;
21     2      END;
22     1      LIMIT$TEST: PROCEDURE(BUF$PTR,NCHAR,MIN$PTR,MAX$PTR) BYTE EXTERNAL;
23     2      DECLARE NCHAR BYTE, (BUF$PTR,MIN$PTR,MAX$PTR) ADDRESS;
24     2      END;
25     1      INSERT: PROCEDURE(NCHAR,SOURCE$PTR,DEST$PTR,DELM$CH,DELM$MASK) EXTERNAL;
26     2      DECLARE (NCHAR,DELM$CH) BYTE, (SOURCE$PTR,DEST$PTR,DELM$MASK) ADDRESS;
27     2      END;
28     1      UNPACK$WAYPT: PROCEDURE(NBYTES,SOURCE$PTR,DEST$PTR) EXTERNAL;
29     2      DECLARE NBYTES BYTE, (SOURCE$PTR,DEST$PTR) ADDRESS;
30     2      END;
31     1      DNV$UTM$DATA$ENTRY: PROCEDURE(WAYPT$ADD,WAYPT$TYPE) BYTE EXTERNAL;
32     2      DECLARE WAYPT$ADD ADDRESS, WAYPT$TYPE BYTE;
33     2      END;

      /* EXTERNAL VARIABLES */
34     1      DECLARE
          OFF$ON$TEXT(6) BYTE EXTERNAL,
          BLANK$LINE(24) BYTE EXTERNAL,
          SWV BYTE EXTERNAL,
          SW$INDEX BYTE EXTERNAL,
          DATA$ENTERED BYTE EXTERNAL,
          IN$BUF(1) BYTE EXTERNAL,
          ZERO(1) BYTE EXTERNAL,
          SUBMODE$GRID ADDRESS EXTERNAL,
          SUBMODE$SW$ENABLE ADDRESS EXTERNAL,
          SUBMODE$STEP$SW$ENABLE ADDRESS EXTERNAL,
          DNV$LAST$WAYPT$PTR ADDRESS EXTERNAL,
          DIGIT$KB$GRID BYTE EXTERNAL,
          DIGIT$KB$TABLEAU BYTE EXTERNAL;

```

```

35 1  DECLARE WAYPT$PTR ADDRESS;
      /* UNPACKED WAYPOINT DATA STRUCTURE */
36 1  DECLARE WAYPT BASED WAYPT$PTR STRUCTURE(
      STATUS BYTE,
      UTM$SPHEROID BYTE,
      UTM$ZONE(2) BYTE,
      UTM$ALPHA(3) BYTE,
      UTM$VALUE(8) BYTE,
      LAT(6) BYTE,
      LON(7) BYTE,
      MAGVAR(6) BYTE);

      /* PACKED WAYPOINT DATA STRUCTURE.
      NOTE: THIS DATA OVERLAYS RMBUF. */
37 1  DECLARE LAST$WAYPT BASED DNV$LAST$WAYPT$PTR STRUCTURE(
      STATUS(2) BYTE,
      UTM(28) BYTE,
      LL(26) BYTE,
      MAGVAR(12) BYTE);


38 1  DISPLAY$WAYPT$TYPE: PROCEDURE(TYPE) PUBLIC;
39 2  DECLARE TYPE BYTE,
      I BYTE;
      /* TYPE FORMAT:
      BIT0: 0 - CKPT, 1 - TGT,
      BIT1: 0 - L/L, 1 - UTM */
40 2  DECLARE WPT$ROW(2) BYTE DATA(4,6),
      WPT$COL(2) BYTE DATA(6,2);
41 2  DECLARE WPT$TEXT(2) STRUCTURE(CH(4) BYTE) DATA('CKPTTGT ');
42 2  DECLARE COORD$TEXT(2) STRUCTURE(CH(3) BYTE) DATA('UTML/L');
      COORD$COL(2) BYTE DATA(6,12);

43 2  I = TYPE AND 1;
44 2  CALL UPDATE$LINE(WPT$ROW(I),WPT$COL(I),4..WPT$TEXT(I));
45 2  I = SHR(TYPE,1) AND 1;
46 2  CALL UPDATE$LINE(6,COORD$COL(I),3..COORD$TEXT(I));
47 2  END;

```

```

REJECT
48 1  DECLARE LAT$TABLEAU(*) BYTE DATA(
      ROW1,'ENTER LATITUDE',
      ROW2,COL19,'N',
      ROW6,COL19,'S',
      ROW8,COL16,'C',COL22,'E',0);
49 1  DECLARE LAT$DIR$ICB STRUCTURE(
      MODE BYTE, NUM$CH BYTE, DELM$CH BYTE, DELM$MASK ADDRESS,
      ECHO$ROW BYTE, ECHO$COL BYTE, SW$ENABLE ADDRESS) DATA(
      NORTH$SOUTH,1,0,0,2,0,,SUE$MODE$STEP$SW$ENABLE);
50 1  DECLARE LAT$VAL$ICB STRUCTURE(
      MODE BYTE, NUM$CH BYTE, DELM$CH BYTE, DELM$MASK ADDRESS,
      ECHO$ROW BYTE, ECHO$COL BYTE, SW$ENABLE ADDRESS) DATA(
      DIGIT,5,'',2800H,2,2,,SUE$MODE$STEP$SW$ENABLE);
51 1  DECLARE LONG$TABLEAU(*) BYTE DATA(
      ROW1,COL6,'LONGITUDE',
      ROW4,COL16,'E',COL22,'W',
      ROW8,COL16,'C',COL22,'E',0);
52 1  DECLARE LONG$DIR$ICB STRUCTURE(
      MODE BYTE, NUM$CH BYTE, DELM$CH BYTE, DELM$MASK ADDRESS,
      ECHO$ROW BYTE, ECHO$COL BYTE, SW$ENABLE ADDRESS) DATA(
      EAST$WEST,1,0,0,2,0,,SUE$MODE$STEP$SW$ENABLE);
53 1  DECLARE LONG$VAL$ICB STRUCTURE(
      MODE BYTE, NUM$CH BYTE, DELM$CH BYTE, DELM$MASK ADDRESS,
      ECHO$ROW BYTE, ECHO$COL BYTE, SW$ENABLE ADDRESS) DATA(
      DIGIT,6,'',1400H,2,1,,SUE$MODE$STEP$SW$ENABLE);
54 1  DECLARE LAT$MAX(*) BYTE DATA('900'),
      LONG$MAX(*) BYTE DATA('180');

55 1  GET$DATA: PROCEDURE(CNTL$PTR,DATA$PTR,DATA$TYPE,WAYPT$TYPE);
      /* READ LAT OR LONG DIRECTION AND VALUE ENTRIES. */
56 2  DECLARE CNTL$PTR ADDRESS,
      DATA$PTR ADDRESS,
      DATA$TYPE BYTE, /* 0 - LAT, 1 - LONG */
      WAYPT$TYPE BYTE;
57 2  DECLARE (DAT BASED DATA$PTR)(1) BYTE;
58 2  DECLARE CNTL BASED CNTL$PTR STRUCTURE(
      MAX$ADD ADDRESS,
      TAB$ADD ADDRESS,
      DIR$ICB$ADD ADDRESS,
      VAL$ICB$ADD ADDRESS);
59 2  DECLARE ETAB(*) BYTE DATA(ROW2,COL4,DEG$SYN,COL7,'.',COL9,'"',0);

60 2  CALL CLEAR(2);
61 2  CALL DISPLAY$WAYPT$TYPE(WAYPT$TYPE);
62 2  CALL GRID(.DIGIT$KB$GRID);
63 2  CALL DISPLAY(CNTL.TAB$ADD);
64 2  A:
      CALL READ(CNTL.DIR$ICB$ADD); /* DIRECTION */
65 2  IF (SWV = SUBMODE$SW) OR (SWV = STEP$SW) THEN RETURN;
67 2  IF NOT DATA$ENTERED THEN GO TO A: /* REJECT (CENTER) NOT PRECEDED BY DATA */
69 2  DAT(0) = IN$BUF(0);
70 2  CALL UPDATE$LINE(2,0,1,,IN$BUF); /* PUT DIR BACK ON THE DISPLAY */
71 2  CALL DISPLAY(.DIGIT$KB$TABLEAU);
72 2  CALL DISPLAY(.ETAB);
      /* INPUT DEGREES VALUE */

```

```

73 2      B:
          CALL READ(CNTL.VAL$ICB$ADD);
74 2      IF (SMV = SUBMODE$SW) OR (SMV = STEP$SW) THEN RETURN;
76 2      IF NOT DATA$ENTERED THEN GO TO B;
78 2      IF LIMIT$TEST(,IN$BUF,2+DATA$TYPE,,ZERO,CNTL.MAX$ADD) AND
          LIMIT$TEST(,IN$BUF(2+DATA$TYPE),3,,ZERO,.LAT$MAX) THEN
79 2          CALL MOVE(5+DATA$TYPE,,IN$BUF,DATA$PTR+1);
80 2      ELSE DO;
81 3          CALL ERROR(0); /* INVALID ENTRY */
82 3          GO TO B;
83 3      END;
84 2      END; /* OF GET$DATA */

85 1      LAT$LONG$DATA$ENTRY: PROCEDURE(WAYPT$ADD,WAYPT$TYPE) BYTE;
86 2      DECLARE WAYPT$ADD ADDRESS,
          WAYPT$TYPE ADDRESS; /* 0 - CKPT, 1 - TOT */
87 2      DECLARE CNTL(2) STRUCTURE(
          MAX$ADD ADDRESS,
          TAB$ADD ADDRESS,
          DIR$ICB$ADD ADDRESS,
          DIR$ICB$ADD ADDRESS) DATA(
          .LAT$MAX,.LAT$TABLEAU,.LAT$DIR$ICB,.LAT$VAL$ICB,
          .LON$MAX,.LON$TABLEAU,.LON$DIR$ICB,.LON$VAL$ICB);

88 2      WAYPT$PTR = WAYPT$ADD;
89 2      CALL GET$DATA(,CNTL(0),,WAYPT.LAT,0,WAYPT$TYPE);
90 2      IF SMV = SUBMODE$SW THEN RETURN(0);
92 2      IF SMV = STEP$SW THEN RETURN(80H);

94 2      CALL GET$DATA(,CNTL(1),,WAYPT.LON,1,WAYPT$TYPE);
95 2      IF SMV = SUBMODE$SW THEN RETURN(0);
97 2      IF SMV = STEP$SW THEN RETURN(80H);
99 2      WAYPT.STATUS = WAYPT.STATUS OR 4;
100 2      RETURN(1); /* NORMAL RETURN */
101 2      END; /* OF LAT$LONG$DATA$ENTRY */

```

```

$EJECT
102 1  MAGVAR$DATA$ENTRY: PROCEDURE(MAYPT$ADD,MAYPT$TYPE) BYTE;
103 2  DECLARE MAYPT$ADD ADDRESS,
      MAYPT$TYPE BYTE;

104 2  DECLARE MAGVAR$TABLEAU(*) BYTE DATA(
      ROW1,'ENTER MAG VAR',
      ROW4,COL16,'E',COL22,'W',
      ROW9,COL16,'C',COL22,'E',0);
105 2  DECLARE CNTL STRUCTURE(
      MAX$ADD ADDRESS,
      TAB$ADD ADDRESS,
      DIR$ICB$ADD ADDRESS,
      VAL$ICB$ADD ADDRESS) DATA(.LAT$MAX,.MAGVAR$TABLEAU,.LOW$DIR$ICB,.LAT$VAL$ICB);

106 2  MAYPT$PTR = MAYPT$ADD;
107 2  CALL CLEAR(1);
108 2  CALL DISPLAY$MAYPT$TYPE(MAYPT$TYPE);
109 2  CALL GRID(.DIGIT$KE$GRID);
110 2  CALL DISPLAY(.MAGVAR$TABLEAU);
111 2  CALL GET$DATA(.CNTL,.MAYPT,.MAGVAR,0,MAYPT$TYPE);
112 2  IF SW = SUBNODE$SW THEN RETURN(0);
114 2  IF SW = STEP$SW THEN RETURN(20H);
116 2  MAYPT.STATUS = MAYPT.STATUS OR 8;
117 2  RETURN(1);

118 2  END: /* OF MAGVAR$DATA$ENTRY */

```

```

SELECT
119 1  DNV$COORD$DATA$ENTRY: PROCEDURE(WAYPT$ADD,WAYPT$TYPE) BYTE PUBLIC;
120 2  DECLARE WAYPT$ADD ADDRESS, /* MUST POINT TO AN UNPACKED WAYPOINT DATA BLOCK */
      WAYPT$TYPE BYTE; /* SEE DISPLAY$WAYPT$TYPE FOR FORMAT DESCRIPTION */
121 2  DECLARE COORD$TABLEAU(*) BYTE DATA(
      ROW2,'SELECT COORDINATE SYSTEM',
      ROW6,COL6,'UTM',COL12,'L/L',0);
122 2  DECLARE COORD$SW$ENABLE STRUCTURE(
      SW$MASK(7) BYTE,
      SW$VALUE(2) BYTE) DATA(
      0,0,0,23H,0,0,88H,
      01H,20H);
123 2  DECLARE COORD$ICB STRUCTURE(
      MODE BYTE, NUM$CH BYTE, DEL$CH BYTE, DEL$MASK ADDRESS,
      ECHO$ROW BYTE, ECHO$COL BYTE, SW$ENABLE ADDRESS) DATA(
      SWITCH,0,0,0,0,0,COORD$SW$ENABLE);
124 2  DECLARE RET$FLAG BYTE;

125 2  CALL CLEAR(2);
126 2  CALL DISPLAY(,COORD$TABLEAU);
127 2  CALL DISPLAY$WAYPT$TYPE(WAYPT$TYPE);
128 2  CALL GRID(,SUBMODE$GRID);
129 2  CALL READ(,COORD$ICB);
130 2  IF SW = SUBMODE$SW THEN RETURN(0);
132 2  IF SW = STEP$SW THEN DO;
134 3  RET$FLAG = 00H; /* COPY ALL DATA FROM LAST WAYPOINT */
135 3  GO TO A;
136 3  END;
137 2  WAYPT$TYPE = WAYPT$TYPE + SHL(SW$INDEX,1);
138 2  IF SW$INDEX THEN RET$FLAG = LAT$LONG$DATA$ENTRY(WAYPT$ADD,WAYPT$TYPE);
140 2  ELSE RET$FLAG = DNV$UTM$DATA$ENTRY(WAYPT$ADD,WAYPT$TYPE);
141 2  IF RET$FLAG = 0 THEN RETURN(0);
143 2  RET$FLAG = RET$FLAG OR MAGVAR$DATA$ENTRY(WAYPT$ADD,WAYPT$TYPE);
144 2  IF RET$FLAG = 0 THEN RETURN(0);

/* IN EACH OF THE ABOVE DATA ENTRY PROCEDURES THE STEP SWITCH IS ENABLED
AT CERTAIN POINTS, WHICH IF PRESSED SIGNIFIES THAT DATA FROM AN EARLIER
OR PREVIOUS WAYPOINT IS TO BE COPIED INTO THE ONE CURRENTLY UNDER
CONSIDERATION, THUS SAVING THE OPERATOR A FEW KEYSTROKES.
THE FORMAT OF THE RETURN FLAG IS:
BIT 7 - LAT$LONG$DATA$ENTRY
BIT 6 - UTM$DATA$ENTRY
BIT 5 - MAGVAR$DATA$ENTRY
IF ALL DATA WAS ENTERED NORMALLY THE RETURN FLAG HAS A VALUE OF ONE (1).
IF THE SUBMODE SWITCH WAS PRESSED, THE RETURN FLAG HAS A VALUE OF ZERO (0).
*/

146 2  IF RET$FLAG = 1 THEN RETURN(1);
148 2  A:
      WAYPT$PTR = WAYPT$ADD;
149 2  IF (RET$FLAG AND 4CH) > 0 THEN DO;
151 3  CALL UNPACK$WAYPT(13, LAST$WAYPT, UTM, WAYPT, UTM$SPHEROID);
152 3  WAYPT.STATUS = WAYPT.STATUS OR 2;
153 3  END;
154 2  IF (RET$FLAG AND 8CH) > 0 THEN DO;
156 3  CALL UNPACK$WAYPT(13, LAST$WAYPT, LL, WAYPT, LAT);

```

```
157 3      WAYPT.STATUS = WAYPT.STATUS OR 4;
158 3      END;
159 2      IF (RET$FLAG AND 20H) > 0 THEN DO;
161 3          CALL UNPACK$WAYPT(6,..LAST$WAYPT..MAGVAR..WAYPT.MAGVAR);
162 3          WAYPT.STATUS = WAYPT.STATUS OR 8;
163 3          END;
164 2      RETURN(1);

165 2      END; /* OF COORD */

166 1      END; /* OF DMV2 */
```

MODULE INFORMATION:

```
CODE AREA SIZE   = 044CH 1100D
VARIABLE AREA SIZE = 0015H 21D
MAXIMUM STACK SIZE = 000CH 12D
339 LINES READ
0 PROGRAM ERROR(S)
```

END OF PL/M-80 COMPILATION

ISIS-II PL/M-80 V3.0 COMPILATION OF MODULE DNV3
 OBJECT MODULE PLACED IN :F1:DNV3.OBJ
 COMPILER INVOKED BY: PLM30 :F1:DNV3.SRC DATE(09AR79) DEBUG

```

      $TITLE('DNV3')
      /* DNV$UPDATE */
1      DNV3: DO:
      $NOLIST INCLUDE(:F1:COLLIT.SRC)
4      1      DISPLAY$ACTIVE$WAYPT: PROCEDURE EXTERNAL;
5      2      END;
6      1      DISPLAY$PRESENT$POSITION: PROCEDURE(ROW) EXTERNAL;
7      2      DECLARE ROW BYTE;
8      2      END;
9      1      DNV$COORD$DATA$ENTRY: PROCEDURE(WAYPT$PTR,WAYPT$TYPE) BYTE EXTERNAL;
10     2      DECLARE WAYPT$PTR ADDRESS, WAYPT$TYPE BYTE;
11     2      END;
12     1      CLEAR: PROCEDURE(START$ROW) EXTERNAL;
13     2      DECLARE START$ROW BYTE;
14     2      END;
15     1      ERROR: PROCEDURE(CODE) EXTERNAL;
16     2      DECLARE CODE BYTE;
17     2      END;
18     1      GRID: PROCEDURE(POINTER) EXTERNAL;
19     2      DECLARE POINTER ADDRESS;
20     2      END;
21     1      DISPLAY: PROCEDURE(IN$PTR) EXTERNAL;
22     2      DECLARE IN$PTR ADDRESS;
23     2      END;
24     1      DISPLAY$DIGIT$KB: PROCEDURE EXTERNAL;
25     2      END;
26     1      READ: PROCEDURE(ICB$PTR) EXTERNAL;
27     2      DECLARE ICB$PTR ADDRESS;
28     2      END;
29     1      UPDATE$LINE: PROCEDURE(ROW,COL,NCHAR,TEXT$PTR) EXTERNAL;
30     2      DECLARE (ROW,COL,NCHAR) BYTE, TEXT$PTR ADDRESS;
31     2      END;
32     1      UNPACK$WAYPT: PROCEDURE(NBYTES,SOURCE$PTR,DEST$PTR) EXTERNAL;
33     2      DECLARE NBYTES BYTE, (SOURCE$PTR, DEST$PTR) ADDRESS;
34     2      END;

      /* EXTERNAL VARIABLES */
35     1      DECLARE
          OFF$ON$TEXT(6) BYTE EXTERNAL,
          BLANK$LINE(24) BYTE EXTERNAL,
          DATA$ENTERED BYTE EXTERNAL,
          IN$BUF(1) BYTE EXTERNAL,
          SWI BYTE EXTERNAL,
          SW$INDEX BYTE EXTERNAL,
          SUBMODE$GRID BYTE EXTERNAL,
          SUBMODE$SW$ENABLE BYTE EXTERNAL,
          DNV$CKPT$DATA(10) STRUCTURE(DIGITS(68) BYTE) EXTERNAL,

          DNV$TEMP$WAYPT STRUCTURE(
              STATUS BYTE,

```

```

UTM$SPHEROID BYTE,
UTM$ZONE(2) BYTE,
UTM$ALPHA(3) BYTE,
UTM$VALUE(8) BYTE,
LATITUDE(6) BYTE,
LONGITUDE(7) BYTE,
MAGVAR(6) BYTE) EXTERNAL;

```

```

36 1  DNV$UPDATE: PROCEDURE PUBLIC;
37 2  DECLARE TAB1(*) BYTE DATA(
      ROW0,'DNV',COL9,'UPDATE',
      ROW4,COL6,'CKPT',
      ROW7,COL7,'UP',
      ROW8,COL6,'DATE',0);
38 2  DECLARE ENABLE1 STRUCTURE(
      SW$MASK(7) BYTE,
      SW$VALUE(2) BYTE) DATA(
      0,0,20H,0,20H,0,80H, /* CKPT, UPDATE, SUBMODE */
      01H,20H);
39 2  DECLARE ICB1 STRUCTURE(
      MODE BYTE, NUM$CH BYTE, DELM$CH BYTE, DELM$MASK ADDRESS,
      ECHO$ROW BYTE, ECHO$COL BYTE, SW$ENABLE ADDRESS) DATA(
      SWITCH,0,0,0,0,0,ENABLE);
40 2  DECLARE TAB2(*) BYTE DATA(ROW2,'ENTER CKPT',0);
41 2  DECLARE ICB2 STRUCTURE(
      MODE BYTE, NUM$CH BYTE, DELM$CH BYTE, DELM$MASK ADDRESS,
      ECHO$ROW BYTE, ECHO$COL BYTE, SW$ENABLE ADDRESS) DATA(
      DIGIT,1,0,0,2,11, SUBMODE$SW$ENABLE);
42 2  DECLARE FRZE$TAB(*) BYTE DATA(ROW4,'FRZE',0);
43 2  DECLARE FRZE$ENABLE STRUCTURE(
      SW$MASK(7) BYTE,
      SW$VALUE(1) BYTE) DATA(
      0,0,80H,0,0,0,80H, /* FRZE, SUBMODE */
      01H);
44 2  DECLARE FRZE$ICB STRUCTURE(
      MODE BYTE, NUM$CH BYTE, DELM$CH BYTE, DELM$MASK ADDRESS,
      ECHO$ROW BYTE, ECHO$COL BYTE, SW$ENABLE ADDRESS) DATA(
      SWITCH,0,0,0,0,0,FRZE$ENABLE);
45 2  DECLARE UPDATE$TAB(*) BYTE DATA(
      ROW2,'UPDATE DIST 12.3',
      ROW7,COL7,'UP',
      ROW8,COL6,'DATE',0);
46 2  DECLARE UPDATE$ENABLE STRUCTURE(
      SW$MASK(7) BYTE,
      SW$VALUE(1) BYTE) DATA(
      0,0,0,0,20H,0,80H, /* UPDATE, SUBMODE */
      01H);
47 2  DECLARE UPDATE$ICB STRUCTURE(
      MODE BYTE, NUM$CH BYTE, DELM$CH BYTE, DELM$MASK ADDRESS,
      ECHO$ROW BYTE, ECHO$COL BYTE, SW$ENABLE ADDRESS) DATA(
      SWITCH,0,0,0,0,0,UPDATE$ENABLE);
48 2  DECLARE NEW$FRZE$TAB(*) BYTE DATA(
      ROW4,'FRZE',
      ROW8,'NEW',0);
49 2  DECLARE NEW$FRZE$ENABLE STRUCTURE(
      SW$MASK(7) BYTE,
      SW$VALUE(2) BYTE) DATA(

```

```

0.0.80H.0.80H.0.80H. /* NEW, FRZE, SUBMODE */
01H.20H);
50 2  DECLARE NEW$FRZE$ICB STRUCTURE(
      MODE BYTE, NUP$ICH BYTE, IEL$MCH BYTE, DELM$MASK ADDRESS,
      ECH$1ROW BYTE, ECH$MCL BYTE, SW$ENABLE ADDRESS) DATA(
      SWITCH.0.0.0.0.0.1,NEW$FRZE$ENABLE);
51 2  DECLARE I BYTE;

52 2  TOP:
      CALL CLEAR(0);
53 2  CALL DISPLAY$ACTIVE$WAYPT;
54 2  CALL DISPLAY$PRESENT$POSITION(2);
55 2  CALL DISPLAY(.TAB1);
56 2  CALL GRID(.SUBMODE$GRID);
57 2  CALL READ(.ICB1);
58 2  DO CASE SW$INDEX:
59 3      GO TO A: /* CKPT SWITCH PRESSED */
60 3      GO TO C: /* UPDATE SWITCH PRESSED */
61 3      RETURN: /* SUBMODE */
62 3      END;
63 2  A:
      CALL CLEAR(2);
64 2  CALL DISPLAY(.TAB2); /* "ENTER CKPT" */
65 2  CALL DISPLAY$DIGIT$KB;
66 2  B:
      CALL READ(.ICB2); /* CKPT NUMBER */
67 2  IF SW = SUBMODE$SW THEN RETURN;
68 2  IF NOT DATA$ENTERED THEN DO:
71 3      CALL ERROR(0); /* "INVALID ENTRY" */
72 3      GO TO B;
73 3      END;
74 2  I = IN$BUF(0) - 30H;
75 2  CALL UNPACK$WAYPT(34, DNV$CKPT$DATA(I), DNV$TEMP$WAYPT);
76 2  IF (DNV$TEMP$WAYPT.STATUS AND 6) = 0 THEN DO:
78 3      CALL ERROR(2); /* "NO DATA FOR THIS CKPT" */
79 3      GO TO B;
80 3      END;
      /* CKPT SELECTED, WAIT FOR FRZE */
81 2  CALL CLEAR(2);
82 2  CALL DISPLAY(.FRZE$TAB);
83 2  CALL GRID(.SUBMODE$GRID);
84 2  CALL READ(.FRZE$ICB);
85 2  IF SW = SUBMODE$SW THEN RETURN;
      /* FRZE ENTERED, WAIT FOR UPDATE */
87 2  CALL CLEAR(2);
88 2  CALL DISPLAY(.UPDATE$TAB);
89 2  CALL READ(.UPDATE$ICB);
90 2  RETURN: /* DONE */

91 2  C: /* UPDATE SWITCH PRESSED FIRST */
      CALL CLEAR(2);
92 2  CALL DISPLAY(.NEW$FRZE$TAB);
93 2  CALL READ(.NEW$FRZE$ICB);
94 2  DO CASE SW$INDEX:
95 3      GO TO E: /* FRZE */
96 3      GO TO D: /* NEW */

```

```

97 3      RETURN: /* SUBMODE */
98 3      END:
99 2      B:
/* NEW SWITCH PRESSED, ENTER COORDINATES */
IF DNV$COORD$DATA$ENTRY(.DNV$TEMP$WAYPT,CKPT) = 0 THEN RETURN:
/* DATA ENTERED, WAIT FOR FRZE */
101 2      CALL CLEAR(2);
102 2      CALL DISPLAY(.FRZE$TAB);
103 2      CALL GRID(.SUBMODE$GRID);
104 2      CALL READ(.FRZE$ICB);
105 2      IF SW = SUBMODE$SW THEN RETURN:
106 2      GO TO F;
107 2      E:
/* FRZE ENTERED BEFORE NEW */
IF DNV$COORD$DATA$ENTRY(.DNV$TEMP$WAYPT,CKPT) = 0 THEN RETURN:
110 2      F:
CALL CLEAR(1);
CALL DISPLAY(.UPDATE$TAB);
111 2      CALL GRID(.SUBMODE$GRID);
112 2      CALL READ(.UPDATE$ICB);
113 2      CALL READ(.UPDATE$ICB);
114 2      IF SW$INDEX = 0 THEN GO TO TOP;

116 2      END: /* OF UPDATE */

117 1      END: /* OF DNV3 */

```

MODULE INFORMATION:

```

CODE AREA SIZE   = 0204H   5160
VARIABLE AREA SIZE = 0001H   10
MAXIMUM STACK SIZE = 0004H   40
248 LINES READ
0 PROGRAM ERROR(S)

```

END OF PL/M-80 COMPILATION

ISIS-II PL/M-80 V3.1 COMPILATION OF MODULE DN4
 OBJECT MODULE PLACED IN :F1:DN4.OBJ
 COMPILER INVOKED BY: PLM30 :F1:DN4.SRC DATE(1NOV79) DEBUG

```

      $TITLE('DN4')
      /* FORMAT$WAYPT$COORD, DN$WAYPT$STATUS, DN$BACKUP$NAV */
1      DN4: DO;
      $HOLIST INCLUDE(:F1:CDULIT.SRC)
4      1      DISPLAY$ACTIVE$WAYPT: PROCEDURE EXTERNAL;
5      2      END;
6      1      CLEAR: PROCEDURE(ROW) EXTERNAL;
7      2      DECLARE ROW BYTE;
8      2      END;
9      1      DISPLAY: PROCEDURE(PTR) EXTERNAL;
10     2      DECLARE PTR ADDRESS;
11     2      END;
12     1      UPDATE$LINE: PROCEDURE(ROW,COL,NBYTES,TEXT$PTR) EXTERNAL;
13     2      DECLARE (ROW,COL,NBYTES) BYTE, TEXT$PTR ADDRESS;
14     2      END;
15     1      DISPLAY$DIGIT$KB: PROCEDURE EXTERNAL;
16     2      END;
17     1      GRID: PROCEDURE(PTR) EXTERNAL;
18     2      DECLARE PTR ADDRESS;
19     2      END;
20     1      READ: PROCEDURE(ICB$PTR) EXTERNAL;
21     2      DECLARE ICB$PTR ADDRESS;
22     2      END;
23     1      INSERT: PROCEDURE(NCHAR,SOURCE$PTR,DEST$PTR,DELM$CH,DELM$MASK) EXTERNAL;
24     2      DECLARE (NCHAR,DELM$CH) BYTE, (SOURCE$PTR,DEST$PTR,DELM$MASK) ADDRESS;
25     2      END;
26     1      PACK$WAYPT: PROCEDURE(COUNT,SOURCE$PTR,DEST$PTR) EXTERNAL;
27     2      DECLARE COUNT BYTE, (SOURCE$PTR, DEST$PTR) ADDRESS;
28     2      END;
29     1      UNPACK$WAYPT: PROCEDURE(COUNT,SOURCE$PTR,DEST$PTR) EXTERNAL;
30     2      DECLARE COUNT BYTE, (SOURCE$PTR, DEST$PTR) ADDRESS;
31     2      END;
32     1      CLEAR$TEMP$BUF: PROCEDURE EXTERNAL;
33     2      END;
34     1      DN$COORD$DATA$ENTRY: PROCEDURE(WAYPT$PTR,WAYPT$TYPE) BYTE EXTERNAL;
35     2      DECLARE WAYPT$PTR ADDRESS, WAYPT$TYPE BYTE;
36     2      END;

      /* EXTERNAL VARIABLES */
37     1      DECLARE
          SWV BYTE EXTERNAL,
          SW$INDEX BYTE EXTERNAL,
          IN$BUF(1) BYTE EXTERNAL,
          DATA$ENTERED BYTE EXTERNAL,
          TEMP$BUF(1) BYTE EXTERNAL,
          SUBMODE$GRID BYTE EXTERNAL,
          SUBMODE$WHEN$PLE BYTE EXTERNAL,
          SPHEREID$TEXT(6) STRUCTURE(CH(3) BYTE) EXTERNAL,

          DN$TEMP$WAYPT STRUCTURE(
              STATUS BYTE,

```

UTM\$SPHEROID BYTE,
UTM\$ZONE(2) BYTE,
UTM\$ALPHA(1) BYTE,
UTM\$VALUE(8) BYTE,
LATITUDE(6) BYTE,
LONGITUDE(7) BYTE,
MAGVAR(6) BYTE) EXTERNAL,

DNV\$STATUS BYTE EXTERNAL,
DNV\$DESTINATION BYTE EXTERNAL,
DNV\$CKPT\$DATA BYTE EXTERNAL,
DNV\$GT\$DATA BYTE EXTERNAL,
DNV\$LAST\$WAYPT\$PTR ADDRESS EXTERNAL,
DNV\$WIND\$DIR BYTE EXTERNAL,
DNV\$WIND\$SPEED(3) BYTE EXTERNAL,
DNV\$GROUND\$SPEED(3) BYTE EXTERNAL,
DNV\$TRACK\$ANGLE(3) BYTE EXTERNAL;

38 1 DECLARE WAYPT\$BASE\$ADD ADDRESS,
(PACKED\$WAYPT BASED WAYPT\$BASE\$ADD)(10) STRUCTURE(CH(68) BYTE);

```

$EJECT
39 1  FORMAT$WAYPT$COORD: PROCEDURE(WAYPT$ADD) PUBLIC:
40 2  DECLARE WAYPT$ADD ADDRESS: /* MUST POINT TO AN UNPACKED WAYPOINT DATA BLOCK */
41 2  DECLARE WAYPT BASED WAYPT$ADD STRUCTURE(
        STATUS BYTE,
        UTM$SPHEROID BYTE,
        UTM$ZONE(2) BYTE,
        UTM$ALPHA(3) BYTE,
        UTM$VALUE(8) BYTE,
        LATITUDE(6) BYTE,
        LONGITUDE(7) BYTE,
        MAGVAR(6) BYTE);
42 2  DECLARE ERROR$MSG(*) BYTE DATA('DATA NOT AVAILABLE');
43 2  FORMAT$LAT$LON: PROCEDURE;
44 3      CALL INSERT(6,.WAYPT.LATITUDE,.TEMP$BUF(3),' ',1200H);
45 3      CALL INSERT(7,.WAYPT.LONGITUDE,.TEMP$BUF(14),' ',0900H);
46 3      TEMP$BUF(6),TEMP$BUF(18) = DEG$SYM;
47 3      TEMP$BUF(11),TEMP$BUF(23) = ' ';
48 3  END;
49 2  FORMAT$UTH: PROCEDURE;
50 3      CALL MOVE(3,.SPHEROID$TEXT(WAYPT.UTM$SPHEROID AND OFH),.TEMP$BUF(3));
51 3      CALL INSERT(13,.WAYPT.UTM$ZONE,.TEMP$BUF(7),' ',2210H);
52 3  END;
53 2  IF DNV$STATUS THEN DO;
55 3      IF (WAYPT.STATUS AND 2) > 0 THEN CALL FORMAT$UTH;
57 3      ELSE DO;
58 4          IF (WAYPT.STATUS AND 4) > 0 THEN CALL FORMAT$LAT$LON;
60 4          ELSE CALL MOVE(18,.ERROR$MSG,.TEMP$BUF(3));
61 4          END;
62 3      END;
63 2  ELSE DO;
64 3      IF (WAYPT.STATUS AND 4) > 0 THEN CALL FORMAT$LAT$LON;
66 3      ELSE DO;
67 4          IF (WAYPT.STATUS AND 2) > 0 THEN CALL FORMAT$UTH;
69 4          ELSE CALL MOVE(18,.ERROR$MSG,.TEMP$BUF(3));
70 4          END;
71 3      END;
72 2  END; /* OF FORMAT$WAYPT$COORD */

```

```

$EJECT
73 1  DNV$WAYPT$STATUS: PROCEDURE(TYPE) PUBLIC;
74 2  DECLARE TYPE BYTE; /* 0 == CKPT, 1 == TGT */

75 2  DECLARE STAT$TABLEAU(*) BYTE DATA(
      ROW0,'DNV',
      ROW1,COL22,'/3',0);
76 2  DECLARE TYPE$TABLEAU(2) STRUCTURE(TEXT(7) BYTE) DATA(
      ROW0,COL6,'CKPT',0,
      ROW0,COL6,'TGT',0);
77 2  DECLARE PAGE12$STAT$SW$ENABLE STRUCTURE(
      SW$MASK(7) BYTE,
      SW$ENABLE(2) BYTE) DATA(
      0,80H,80H,80H,80H,0,68H, /* SW = 100,200,300,400,SUBMODE, STEP */
      01H,23H);
78 2  DECLARE PAGE12$STAT$ICB STRUCTURE(
      MODE BYTE, NUM$CH BYTE, DELM$CH BYTE, DELM$MASK ADDRESS,
      ECHO$ROW BYTE, ECHO$COL BYTE, SW$ENABLE ADDRESS) DATA(
      SWITCH,0,0,0,0,0,0,PAGE12$STAT$SW$ENABLE);
79 2  DECLARE PAGE3$STAT$SW$ENABLE STRUCTURE(
      SW$MASK(7) BYTE,
      SW$VALUE(1) BYTE) DATA(
      0,80H,80H,0,0,0,68H, /* SW = 100,200,SUBMODE,STEP */
      01H);
80 2  DECLARE PAGE3$STAT$ICB STRUCTURE(
      MODE BYTE, NUM$CH BYTE, DELM$CH BYTE, DELM$MASK ADDRESS,
      ECHO$ROW BYTE, ECHO$COL BYTE, SW$ENABLE ADDRESS) DATA(
      SWITCH,0,0,0,0,0,0,PAGE3$STAT$SW$ENABLE);
81 2  DECLARE PAGE$SIZE(3) BYTE DATA(3,3,1), /* MINUS ONE */
      WAYPT$NUM BYTE,
      WAYPT$OFFSET BYTE,
      STAR$ENABLE BYTE,
      ACTIVE$WAYPT BYTE,
      (PAGE,1,N) BYTE;

82 2  IF TYPE = CKPT THEN WAYPT$BASE$ADD = .DNV$CKPT$DATA;
84 2  ELSE WAYPT$BASE$ADD = .DNV$TGT$DATA;
85 2  STAR$ENABLE = NOT (TYPE XOR DNV$DESTINATION);
86 2  ACTIVE$WAYPT = SHR(DNV$DESTINATION,1) AND 0FH;
87 2  PAGE = 0;
88 2  WAYPT$OFFSET = 0;
89 2  DO FOREVER;
90 3  CALL CLEAR(0);
91 3  CALL DISPLAY$ACTIVE$WAYPT;
92 3  CALL DISPLAY(.TYPE$TABLEAU(TYPE));
93 3  CALL DISPLAY(.STAT$TABLEAU);
94 3  N = PAGE$SIZE(PAGE);
95 3  TEMP$BUF(0) = PAGE + 31H;
96 3  CALL UPDATE$LINE(1,21,1,TEMP$BUF);
97 3  CALL CLEAR(2);
98 3  DO I = 0 TO N;
99 4  CALL CLEAR$TEMP$BUF;
100 4  WAYPT$NUM = WAYPT$OFFSET + I;
101 4  TEMP$BUF(1) = WAYPT$NUM + 30H;
102 4  IF STAR$ENABLE AND (WAYPT$NUM = ACTIVE$WAYPT) THEN TEMP$BUF(2) = 'B';

```

```
104 4      CALL UNPACK$WAYPT(34,.PACKED$WAYPT(WAYPT$NUM),.DNV$TEMP$WAYPT);
105 4      CALL FORMAT$WAYPT$COORD(.DNV$TEMP$WAYPT);
106 4      CALL UPDATE$LINE(1+2*2,0,24,.TEMP$BUF);
107 4      END;
108 3      IF PAGE < 3 THEN CALL READ(.PAGE12$STAT$ICB);
109 3          ELSE CALL READ(.PAGE3$STAT$ICB);
110 3
111 3      IF SWV = SUBMCIE$SW THEN RETURN;
112 3      IF SWV = STEP$SW THEN DO;
113 4          PAGE = (PAGE + 1) MOD 3;
114 4          WAYPT$OFFSET = PAGE*4;
115 4          END;
116 3      ELSE DO;
117 4          WAYPT$NUM = WAYPT$OFFSET + SW$INDEX;
118 4          IF DNV$COORD$DATA$ENTRY(.DNV$TEMP$WAYPT,TYPE) = 1 THEN DO;
119 5              CALL PACK$WAYPT(34,.DNV$TEMP$WAYPT,.PACKED$WAYPT(WAYPT$NUM));
120 5              DNV$LAST$WAYPT$PTR = .PACKED$WAYPT(WAYPT$NUM);
121 5              END;
122 4          END;
123 3      END; /* OF DO FOREVER */
124 2      END; /* OF DNV$WAYPT$STATUS */
```

```

OBJECT
128 1   DIM$BACKUP$RAW: PROCEDURE PUBLIC;
129 2   DECLARE BACKUP$STAELEJ(4) BYTE DATA(
        ROW0,'DIR',COL9,'BACKUP',
        ROW3,' WIND WIND',COL11,'GSPD',
        ROW4,' DIR',COL6,'SPD',
        ROW5,' TRK',
        ROW6,' ANG',0);
130 2   DECLARE BACKUP$SW$ENABLE STRUCTURE(
        SW$MASK(7) BYTE,
        SW$VALUE(3) BYTE) DATA(
        0,0,0A$H,80H,0,0,80H, /*WIND DIR,WIND SPD,GSPD,TRK,SUBNODE */
        01H,23H,40H);
131 2   DECLARE BACKUP$I$CB STRUCTURE(
        MODE BYTE, NUM$CH BYTE, DELM$CH BYTE, DELM$MASK ADDRESS,
        ECHO$ROW BYTE, ECHO$COL BYTE, SW$ENABLE ADDRESS) DATA(
        SWITCH,0,0,0,0,0,0,,BACKUP$SW$ENABLE);
132 2   DECLARE BACKUP$DATA$(CB(4)) STRUCTURE(
        MODE BYTE, NUM$CH BYTE, DELM$CH BYTE, DELM$MASK ADDRESS,
        ECHO$ROW BYTE, ECHO$COL BYTE, SW$ENABLE ADDRESS) DATA(
        DIGIT,3,0,0,2,9,,SUBMODE$SW$ENABLE, /* WIND DIRECTION */
        DIGIT,2,0,0,2,9,,SUBMODE$SW$ENABLE, /* WIND SPEED */
        DIGIT,3,0,0,2,11,,SUBMODE$SW$ENABLE, /* GROUND SPEED */
        DIGIT,3,0,0,2,10,,SUBMODE$SW$ENABLE); /* TRACK ANGLE */
133 2   DECLARE BACKUP$ENTRY$TAB(4) STRUCTURE(CH(13) BYTE) DATA(
        ROW2,'WIND DIR ',0,
        ROW2,'WIND SPD ',0,
        ROW2,'GROUND SPD ',0,
        ROW2,'TRACK ANGLE',0);
134 2   DECLARE NUM$DIGITS(4) BYTE DATA(3,2,3,3);
135 2   DECLARE DATA$ADD(4) ADDRESS DATA(
        ,DIM$WIND$DIR,
        ,DIM$WIND$SPEED,
        ,DIM$GROUND$SPEED,
        ,DIM$TRACK$ANGLE);

```

```

REJECT
136 2    BACKUP$DATA$ENTRY: PROCEDURE(I);
137 3    DECLARE I BYTE;

138 3    CALL CLEAR(2);
139 3    CALL DISPLAY(.BACKUP$ENTRY$TAB(I));
140 3    CALL DISPLAY$DIGITS$B;
141 3    CALL READ(.BACKUP$DATA$ICB(I));
142 3    IF SW = SUBMODE$SW THEN RETURN;
144 3    IF DATA$ENTERED AND (SW = ENTER$SW) THEN
145 3        CALL MOVE(NUM$DIGITS(I),.IN$BUF,DATA$ADD(I));
146 3    END; /* OF BACKUP$DATA$ENTRY */

147 2    DISPLAY$STAT$LINE: PROCEDURE;
148 3    DECLARE STAT$TABLEAU(4) BYTE DATA(
        ROW1,'W',COL4,'/',COL9,'TRK',COL17,'GSPD',0);
149 3    DECLARE LINE$LOC(4) ADDRESS DATA(
        .TEMP$BUF+1,.TEMP$BUF+5,.TEMP$BUF+21,.TEMP$BUF+12);
150 3    DECLARE I BYTE;

151 3    CALL DISPLAY(.STAT$TABLEAU);
152 3    CALL CLEAR$TEMP$BUF;
153 3    DO I = 0 TO 3;
154 4        CALL MOVE(NUM$DIGITS(I),DATA$ADD(I),LINE$LOC(I));
155 4    END;
156 3    CALL UPDATE$LINE(1,0,24,.TEMP$BUF);

157 3    END; /* OF DISPLAY$STAT$LINE */

/* BEGIN BACKUP CODE */
158 2    DO FOREVER;
159 3        CALL CLEAR(0);
160 3        CALL DISPLAY$ACTIVE$W$M$PT;
161 3        CALL DISPLAY$STAT$LINE;
162 3        CALL DISPLAY(.BACKUP$TABLEAU);
163 3        CALL GRID(.SUBMODE$GRID);
164 3        CALL READ(.BACKUP$ICB);
165 3        IF SW = SUBMODE$SW THEN RETURN;
167 3        CALL BACKUP$DATA$ENTRY(SW$INDEX);
168 3    END; /* OF DO FOREVER */
169 2    END; /* DNV$BACKUP$SW */

170 1    END; /* OF DNV4 */

```

MODULE INFORMATION:

```

CODE AREA SIZE   = 0491H 1169D
VARIABLE AREA SIZE = 000EH 14D
MAXIMUM STACK SIZE = 000CH 12D
333 LINES READ
0 PROGRAM ERROR(S)

```

PL/M-80 COMPILER DNV4

1NOV79 PAGE 8

END OF PL/M-80 COMPILATION

ISIS-II PL/M-80 V2.1 COMPILATION OF MODULE DNV5
 OBJECT MODULE PLACED IN :F1:DNV5.OBJ
 COMPILER INVOKED BY: PLM80 :F1:DNV5.SRC DATE(1NOV79) DEBUG

```

      $TITLE('DNV5')
      /* DNV5UTH$DATA$ENTRY, PACK$WAYPT, UNPACK$WAYPT */
1      DNV5: DO;
2      1      DECLARE HORZ LITERALLY 'OFH',
              VERT LITERALLY 'OFVH';
      $MOLIST INCLUDE(:F1:CDULIT.SRC)
5      1      CLEAR: PROCEDURE(ROW) EXTERNAL;
6      2      DECLARE ROW BYTE;
7      2      END;
8      1      DISPLAY: PROCEDURE(PTR) EXTERNAL;
9      2      DECLARE PTR ADDRESS;
10     2      END;
11     1      UPDATE$LINE: PROCEDURE(ROW,COL,NCHAR,TEXT$PTR) EXTERNAL;
12     2      DECLARE (ROW,COL,NCHAR) BYTE, TEXT$PTR ADDRESS;
13     2      END;
14     1      READ: PROCEDURE(ICB$PTR) EXTERNAL;
15     2      DECLARE ICB$PTR ADDRESS;
16     2      END;
17     1      GRID: PROCEDURE(GRID$PTR) EXTERNAL;
18     2      DECLARE GRID$PTR ADDRESS;
19     2      END;
20     1      DISPLAY$DIGIT$KB: PROCEDURE EXTERNAL;
21     2      END;
22     1      DISPLAY$WAYPT$TYPE: PROCEDURE(TYPE) EXTERNAL;
23     2      DECLARE TYPE BYTE;
24     2      END;
25     1      DISPLAY$ACTIVE$WAYPT: PROCEDURE EXTERNAL;
26     2      END;

      /* EXTERNAL VARIABLES */
27     1      DECLARE
              SHV BYTE EXTERNAL,
              SH$INDEX BYTE EXTERNAL,
              IN$BUF(1) BYTE EXTERNAL,
              DATA$ENTERED BYTE EXTERNAL,
              SUBMODE$SH$ENABLE ADDRESS EXTERNAL,
              SUBMODE$STEP$SH$ENABLE ADDRESS EXTERNAL,
              SUBMODE$GRID ADDRESS EXTERNAL;

28     1      DECLARE SPHEROID$TEXT(6) STRUCTURE(CH(3) BYTE) PUBLIC DATA('CLOEVOL6EB0IN0AU0');

29     1      DNV5UTH$DATA$ENTRY: PROCEDURE(WAYPT$PTR,WAYPT$TYPE) BYTE PUBLIC;
30     2      DECLARE WAYPT$PTR ADDRESS, /* MUST POINT TO AN UNPACKED WAYPT DATA BLOCK */
              WAYPT$TYPE BYTE;

      /* UNPACKED WAYPOINT DATA BLOCK */

```

```

31 2  DECLARE WAYPT BASED WAYPT$PTR STRUCTURE(
        STATUS BYTE,
        UTM$SPHEROID BYTE,
        UTM$ZONE(2) BYTE,
        UTM$ALPHA(3) BYTE,
        UTM$VALUE(8) BYTE,
        LAT(6) BYTE,
        LON(7) BYTE,
        MAGVAR(6) BYTE);
32 2  DECLARE SPHEROID$TABLEAU(*) BYTE DATA(
        ROW0,'DNV',
        ROW2,'SELECT SPHEROID',
        ROW4,COL12,'CL0',COL17,'EVO',
        ROW6,COL12,'CL6',COL17,'BE0',
        ROW8,COL12,'INO',COL17,'AU0',0);
33 2  DECLARE SPHEROID$SW$ENABLE STRUCTURE(
        SW$MASK(7) BYTE,
        SW$VALUE(6) BYTE) DATA(
        0,0,0EH,0EH,0EH,0,8EH, /* CL0,EVO,BE0,INO,AU0,SUBMODE,STEP */
        01H,12H,33H,45H,56H,70H);
34 2  DECLARE SPHEROID$ICB STRUCTURE(
        MODE BYTE, NUM$CH BYTE, DELM$CH BYTE, DELM$MASK ADDRESS,
        ECHO$ROW BYTE, ECHO$COL BYTE, SW$ENABLE ADDRESS) DATA(
        SWITCH,0,0,0,0,0,0,SPHEROID$SW$ENABLE);
35 2  DECLARE ZONE$ICB STRUCTURE(
        MODE BYTE, NUM$CH BYTE, DELM$CH BYTE, DELM$MASK ADDRESS,
        ECHO$ROW BYTE, ECHO$COL BYTE, SW$ENABLE ADDRESS) DATA(
        DIGIT,2,0,0,2,11,SUBMODE$STEP$SW$ENABLE);
36 2  DECLARE ZONE$TABLEAU(*) BYTE DATA(ROW2,'ENTER ZONE',0);
37 2  DECLARE ALPHA$KB$TABLEAU(*) BYTE DATA(
        ROW2,COL1,'A',COL4,'B',COL7,'C',COL10,'D',COL13,'E',COL16,'F',COL19,'G',COL22,'H',
        ROW4,COL4,'J',COL7,'K',COL10,'L',COL13,'M',COL16,'N',COL22,'P',
        ROW6,COL1,'Q',COL4,'R',COL7,'S',COL10,'T',COL13,'U',COL16,'V',COL19,'W',COL22,'X',
        ROW8,COL1,'Y',COL4,'Z',COL16,'C',COL22,'E',0);
38 2  DECLARE ALPHA$KB$GRID(*) BYTE DATA(
        HORZ,4,14,22,32,42,52,62,72,82,92,
        HORZ,22,32,22,32,42,52,62,72,82,92,
        HORZ,40,50,22,32,42,52,62,72,82,92,
        HORZ,58,68,22,32,42,52,62,72,82,92,
        HORZ,76,86,22,32,42,52,62,72,82,92,
        HORZ,94,104,22,32,42,52,62,72,82,92,
        HORZ,112,122,22,32,42,52,62,72,82,92,
        HORZ,130,140,22,32,42,52,62,72,82,92,
        VERT,22,32,4,14,22,32,40,50,58,68,76,86,94,104,112,122,130,140,
        VERT,42,52,4,14,22,32,40,50,58,68,76,86,94,104,112,122,130,140,
        VERT,62,72,4,14,22,32,40,50,58,68,76,86,94,104,112,122,130,140,
        VERT,82,92,4,14,22,32,40,50,58,68,76,86,94,104,112,122,130,140,OFFH);
39 2  DECLARE ALPHA$ICB STRUCTURE(
        MODE BYTE, NUM$CH BYTE, DELM$CH BYTE, DELM$MASK ADDRESS,
        ECHO$ROW BYTE, ECHO$COL BYTE, SW$ENABLE ADDRESS) DATA(
        DNM$ALPHA,3,0,0,1,6,SUBMODE$STEP$SW$ENABLE);
40 2  DECLARE VALUE$TABLEAU(*) BYTE DATA(ROW1,'ENTER VALUE',0);
41 2  DECLARE VALUE$ICB STRUCTURE(
        MODE BYTE, NUM$CH BYTE, DELM$CH BYTE, DELM$MASK ADDRESS,
        ECHO$ROW BYTE, ECHO$COL BYTE, SW$ENABLE ADDRESS) DATA(
        DIGIT,8,'',0800H,2,0,SUBMODE$STEP$SW$ENABLE);

```

```

SELECT
/* UTM SPHEROID SELECTION */
42 2    CALL CLEAR(0);
43 2    CALL DISPLAY$ACTIVE$WAYPT;
44 2    CALL DISPLAY(.SPHEROID$TABLEAU);
45 2    CALL DISPLAY$WAYPT$TYPE(WAYPT$TYPE);
46 2    CALL GRID(.SUBMODE$GRID);
47 2    CALL READ(.SPHEROID$ICB);
48 2    IF SWV = SUBMODE$SW THEN RETURN(0);
50 2    IF SWV = STEP$SW THEN RETURN(40H);
52 2    WAYPT.UTM$SPHEROID = SW$INDEX;

/* UTM ZONE ENTRY */
53 2    CALL UPDATE$LINE(0, 4, 3, .SPHEROID$TEXT(WAYPT.UTM$SPHEROID));
54 2    CALL CLEAR(1);
55 2    CALL DISPLAY(.ZONE$TABLEAU);
56 2    CALL DISPLAY$WAYPT$TYPE(WAYPT$TYPE);
57 2    CALL DISPLAY$DIGIT$KB;
58 2    A:
        CALL READ(.ZONE$ICB);
59 2    IF SWV = SUBMODE$SW THEN RETURN(0);
61 2    IF SWV = STEP$SW THEN RETURN(40H);
63 2    IF DATA$ENTERED AND (SWV = ENTER$SW) THEN DO:
65 3        WAYPT.UTM$ZONE(0) = IN$BUF(0);
66 3        WAYPT.UTM$ZONE(1) = IN$BUF(1);
67 3    END;
68 2    ELSE GO TO A: /* REJECT CENTERD NOT PRECEDED BY DATA */

/* UTM ALPHA ENTRY */
69 2    CALL CLEAR(2);
70 2    CALL UPDATE$LINE(0, 7, 2, .IN$BUF); /* PUT ZONE ON THE SCREEN */
71 2    CALL GRID(.ALPHA$KB$GRID);
72 2    CALL DISPLAY(.ALPHA$KB$TABLEAU);
73 2    B:
        CALL READ(.ALPHA$ICB);
74 2    IF SWV = SUBMODE$SW THEN RETURN(0);
76 2    IF SWV = STEP$SW THEN RETURN(40H);
78 2    IF DATA$ENTERED AND (SWV = ENTER$SW) THEN
79 2        CALL MOVE(3, .IN$BUF, .WAYPT.UTM$ALPHA);
80 2    ELSE GO TO B;

/* UTM VALUE ENTRY */
81 2    CALL UPDATE$LINE(0, 9, 3, .IN$BUF);
82 2    CALL CLEAR(3);
83 2    CALL DISPLAY$DIGIT$KB;
84 2    CALL DISPLAY(.VALUE$TABLEAU);
85 2    CALL DISPLAY$WAYPT$TYPE(WAYPT$TYPE);
86 2    C:
        CALL READ(.VALUE$ICB);
87 2    IF SWV = SUBMODE$SW THEN RETURN(0);
89 2    IF SWV = STEP$SW THEN RETURN(40H);
91 2    IF DATA$ENTERED AND (SWV = ENTER$SW) THEN DO:
93 3        CALL MOVE(8, .IN$BUF, .WAYPT.UTM$VALUE);
94 3        WAYPT.STATUS = WAYPT.STATUS OR 2;
95 3    END;
96 2    ELSE GO TO C;
97 2    RETURN(1); /* NORMAL RETURN - DATA ENTERED */

98 2    END: /* OF UTM$DATA$ENTRY */

```

```

99 1      REJECT
        DECLARE(I,J,M) BYTE;

100 1      PACK$WAYPT: PROCEDURE(COUNT,SOURCE$PTR,DEST$PTR) PUBLIC;
        /* THE DATA FROM THE SOURCE ARRAY ARE SEPARATED INTO NIBBLES AND ARE PLACED IN
        SUCCESSIVE BYTES IN THE DESTINATION ARRAY IN THE LEFTMOST FOUR BITS.
        THIS DATA WILL OVERLAY GRAPHICS DATA IN R$BUF9. */
101 2      DECLARE COUNT BYTE, /* NUMBER OF BYTES TO BE PACKED */
        SOURCE$PTR ADDRESS,
        DEST$PTR ADDRESS;
102 2      DECLARE (SOURCE BASED SOURCE$PTR)(1) BYTE,
        (DEST BASED DEST$PTR)(1) BYTE;

103 2      J = 0;
104 2      COUNT = COUNT - 1;
105 2      DO I = 0 TO COUNT;
106 3          M = SOURCE(I);
107 3          DEST(J) = (DEST(J) AND OFH) OR (M AND OFOH);
108 3          DEST(J+1) = (DEST(J+1) AND OFH) OR SHL(M,4);
109 3          J = J + 2;
110 3          END;
111 2      END; /* OF PACK$WAYPT */

112 1      UNPACK$WAYPT: PROCEDURE(COUNT,SOURCE$PTR,DEST$PTR) PUBLIC;
        /* THE LEFTMOST 4-BIT NIBBLES FROM THE SOURCE ARRAY ARE ASSEMBLED INTO BYTES
        IN THE DESTINATION ARRAY. THE FIRST NIBBLE IS PLACED IN THE LEFT 4 BITS
        OF THE FIRST BYTE. */
113 2      DECLARE COUNT BYTE, /* NUMBER OF BYTES TO BE RETRIEVED */
        SOURCE$PTR ADDRESS,
        DEST$PTR ADDRESS;
114 2      DECLARE (SOURCE BASED SOURCE$PTR)(1) BYTE,
        (DEST BASED DEST$PTR)(1) BYTE;

115 2      J = 0;
116 2      COUNT = COUNT - 1;
117 2      DO I = 0 TO COUNT;
118 3          DEST(I) = (SOURCE(J) AND OFOH) OR SHR(SOURCE(J+1),4);
119 3          J = J + 2;
120 3          END;

121 2      END; /* OF UNPACK$WAYPT */

122 1      END; /* OF DMS: DO; */

```

MODULE INFORMATION:

```

CODE AREA SIZE   = 040BH  1035D
VARIABLE AREA SIZE = 0010H   16D
MAXIMUM STACK SIZE = 0006H    6D
264 LINES READ
0 PROGRAM ERROR(S)

```

END OF PL/M-80 COMPILATION

ISIS-II PL/M-80 V3.1 COMPILATION OF MODULE DAT
 OBJECT MODULE PLACED IN :F1:DNVDAT.LBJ
 COMPILER INVOKED BY: PLM80 :F1:DNVDAT.SRC DATE(200CT79) DEBUG

```

1          $TITLE('DNVDAT')
2 1          DAT: DO:
2 1          DECLARE RMBUFA(1536) BYTE PUBLIC,
              RMBUFB(1536) BYTE PUBLIC;

3 1          DECLARE /* DNVD VARIABLES */
              DNVDDESTINATION BYTE PUBLIC,
              DNVDDISPLAYCOORDMODE BYTE PUBLIC,
              /* DNVD INITIALIZATION REQUIRES THE FOLLOWING ITEMS BE CONTIGUOUS */
              DNVDWINDSPEED(3) BYTE PUBLIC,
              DNVDWINDSDIR(2) BYTE PUBLIC,
              DNVDGROUNDSPD(3) BYTE PUBLIC,
              DNVDWINDRANGE(4) BYTE PUBLIC,
              DNVDTRACKANGLEERROR(3) BYTE PUBLIC,
              DNVDTRACKANGLE(3) BYTE PUBLIC,
              DNVDXROSSTRACKANGLE(3) BYTE PUBLIC,
              DNVDBEARING(3) BYTE PUBLIC,
              DNVDTIMESTOGO(3) BYTE PUBLIC,
              /*****/
              DNVDSTATUS BYTE PUBLIC,
              DNVDLASTWAYPTPTR ADDRESS PUBLIC,
              DNVDTEMPWAYPT(34) BYTE PUBLIC,
              DNVDCKPTDATA(10) STRUCTURE(DIGITS(68) BYTE) PUBLIC AT(.RMBUFB),
              DNVDGTDATA(10) STRUCTURE(DIGITS(68) BYTE) PUBLIC AT(.RMBUFB+680);
4 1          END: /* DAT */

```

MODULE INFORMATION:

```

CODE AREA SIZE      = 0000H      00
VARIABLE AREA SIZE  = 0C42H      3138D
MAXIMUM STACK SIZE  = 0000H      00
25 LINES READ
0 PROGRAM ERROR(S)

```

END OF PL/M-80 COMPILATION

APPENDIX

TASK REPORT

Results of "State of Art" Review

Function Flow and Allocations

Description of Alternate Designs

Description of Proposed Final Approach

RESULTS OF THE
STATE OF THE ART REVIEW



TECHNICAL APPROACH

APPROACH PREDICTED ON TECHNICAL INTENT OF PROJECT

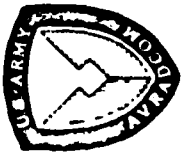
CONSTRAINTS ON DESIGN APPROACH

EXCLUSIVE USE OF PROCESSOR TECHNOLOGY

GENERALIZE APPROACH — BUILDING BLOCKS

CONCEPTUAL DESIGN

MEMORY MAP



APPROACH PREDICTED ON TECHNICAL INTENT OF PROJECT

TECHNICAL INTENT OF PROJECT

- DEMONSTRATE THAT SIX COCKPIT DEVICES MAY BE REPLACED BY ONE
-

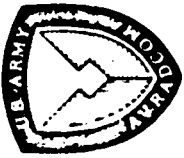
REASONS

- REDUCE THE NUMBER OF DEVICES PILOT MUST MAINTAIN WITHIN HIS COGNIZANCE
- SIMPLIFY OPERATING PROCEDURES
- ALLOW SUMMARY DISPLAY OF ENTIRE SYSTEM STATUS
- OPTIMIZATION OF COCKPIT REAL ESTATE



DATA SYSTEMS

12027-3



CONSTRAINTS ON DESIGN APPROACH

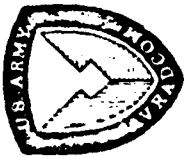
EXISTING BUILDING BLOCKS

- THE 48 x 96 LED MODULES
- MEMBRANE SWITCH
- DUAL IN LINE PACKAGES FOR ELECTRONIC DEVICES

CDU VOLUME ALLOWANCE

EASE OF MODIFICATION

- FUNCTION
- CONFIGURATION



EXCLUSIVE USE OF PROCESSOR TECHNOLOGY

FIRMWARE BASED SYSTEM ALLOWING FUNCTIONAL RECONFIGURATION
WITH NO IMPACT TO HARDWARE

ALGORITHMIC RATHER THAN LOGIC BASED

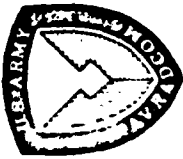
- LOOK UP TABLE RATHER THAN SYMBOL GENERATION
 - ALGORITHM RATHER THAN LINE GENERATOR
 - DISPLAY BUFFER WITH DIRECT MEMORY ACCESS RATHER THAN
ON-THE-FLY DISPLAY GENERATION
-

POSSIBLE FUTURE USE OF COMPUTER CHIPS TO REPLACE CPU AND
SUPPORTING CHIPS



DATA SYSTEMS

12027-5



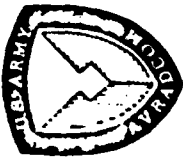
GENERALIZED APPROACH — A GROUP OF BUILDING BLOCKS

FRONT PANEL MAY TAKE ON ANY FORM

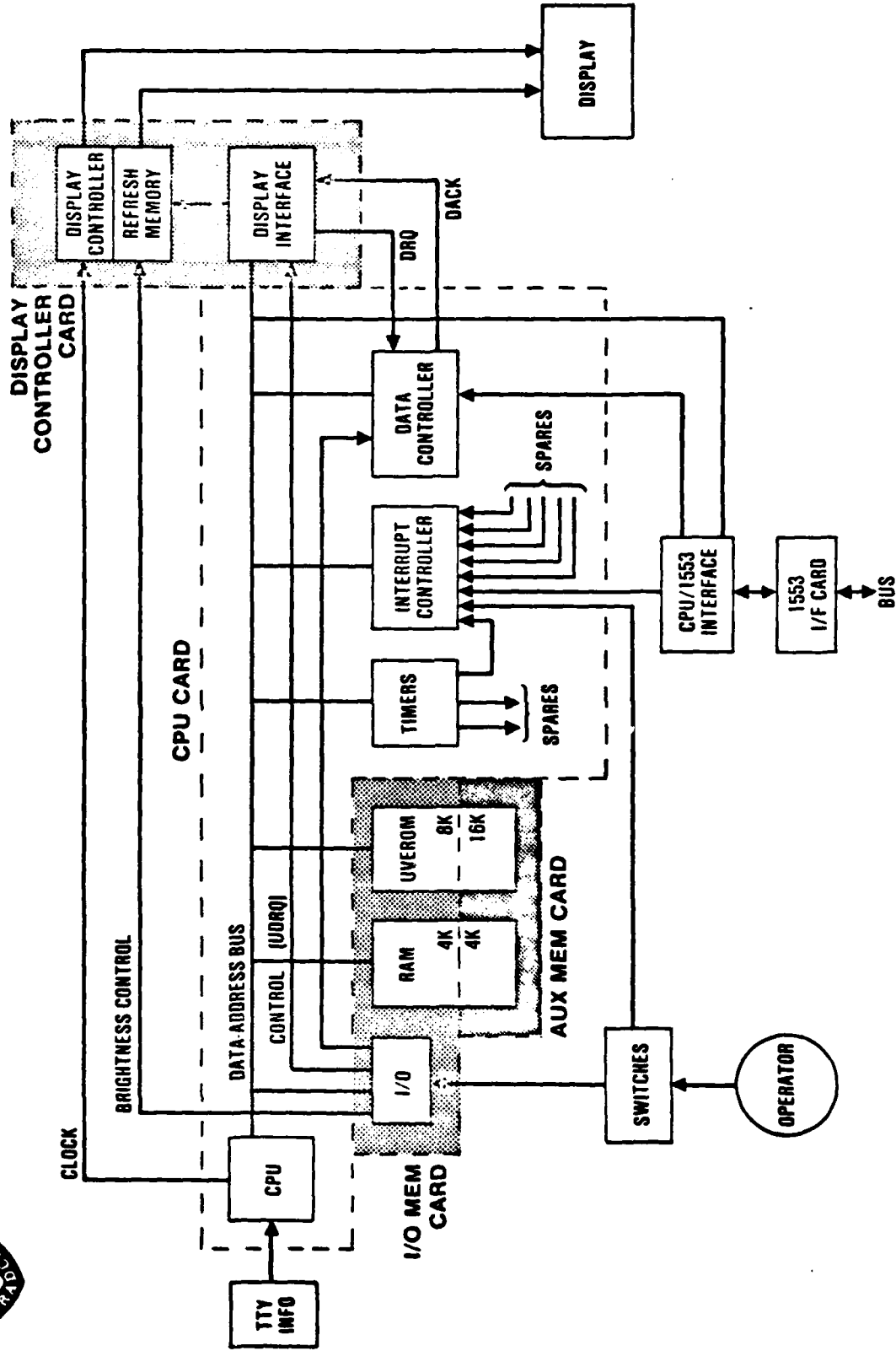
- MAXIMUM 64 SWITCHES
- THREE 48 x 96 LED MODULES
- ALL SWITCHES ARE LOGIC ACTIVATED

SWITCHES MAY TAKE ANY MEANING (SW ASSY CHANGEABLE)

MEMORY EXPANDABLE FROM 12K TO 32K

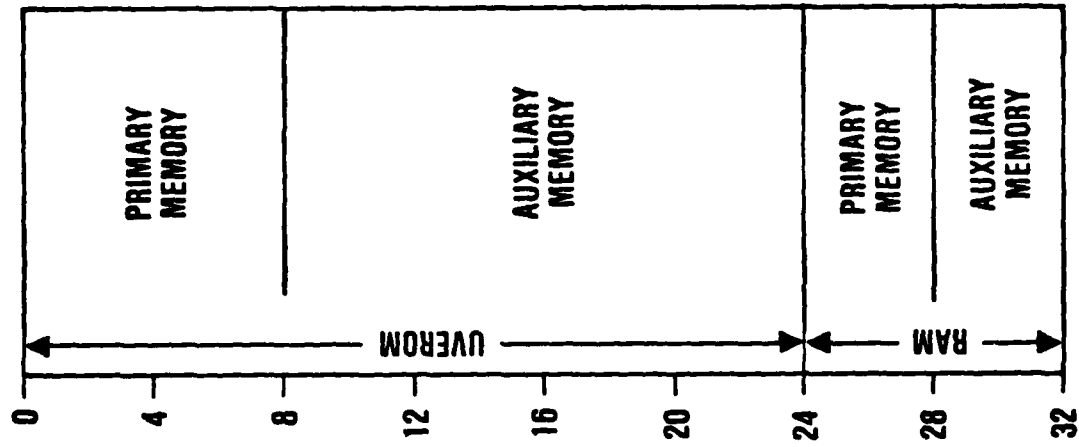


CONCEPTUAL DESIGN





MEMORY MAP

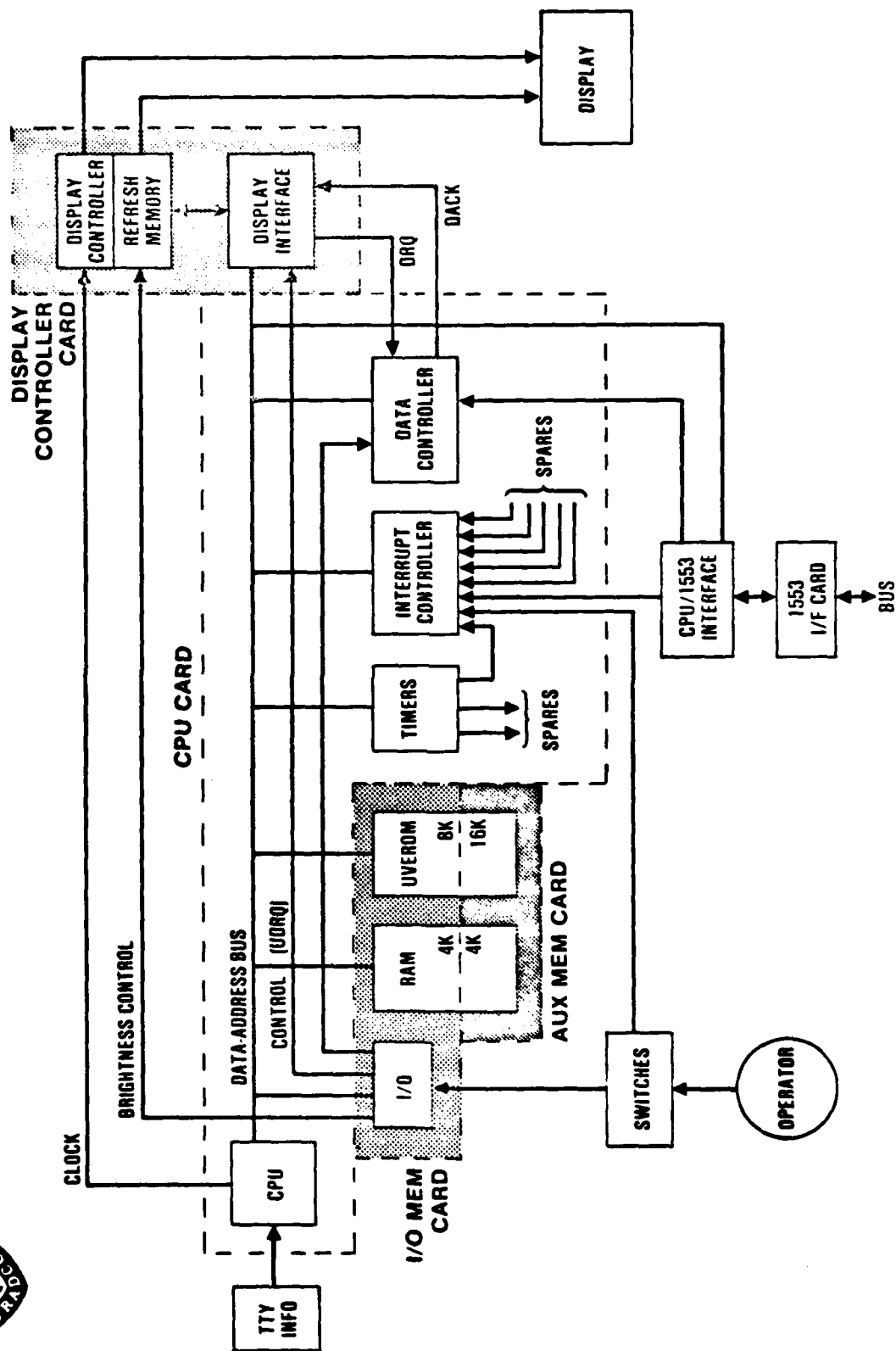


CDU FUNCTIONAL FLOW

AND ALLOCATION



CDU FUNCTIONAL FLOW AND ALLOCATION



DESCRIPTION OF

ALTERNATE DESIGNS

DESCRIPTION OF ALTERNATE DESIGNS

The culmination of current State of Art investigations and incorporation of the resultant technologies has determined the present Design philosophy. Alternate designs have, however, been anticipated in the implementation of certain control functions. These are:

1. Power Control
2. Manual/Automatic Display Brightness Control
3. Display Response Control

The Area of Power Control was not fully designed to an end item configuration due to the requirement for the Control Display Unit to be operational in a portable configuration and utilizing Standard 110V AC as a power source. It is, therefore, anticipated that power control implementation studies should be completed to determine Mechanical/logic implementation from a system view.

Bright/Dim control is an area that, for suitcase demonstrations, was implemented in a straight-forward manner, that is, the use of an octal coded thumb-wheel switch. Studies must be completed to determine light control requirements for cockpit displays so that appropriate control policies may be implemented into future designs..

Display response is less than optimized due to the implementation limitations of Design funding, program implementation time constraints, and customer definition. Therefore, there is left two other optimization avenues that should be considered for final Design. These are: Restructuring of the Display orientation and redesign of existing module design, to adapt the presently used module for this specific application.

DESCRIPTION OF PROPOSED FINAL APPROACH

The requirements identified at this time that will allow transition from a suitcase demonstration unit into an Advanced Engineering Development model are as follows:

A. Studies

- Human Factors
- System Integration
- Program Loading
- System Redundancy
- Test Equipment
- Design to Unit Production Cost Goal
- Power Control Implementation
- Display Brightness Control (where, how)

B. Design

- Appropriate 1553 Interface
- ADM Electro/Mechanical Implementation

C. Fabrication

- Two Units (ADM)

D. Testing

- Electrical Verification - 1 Unit
- Electro/Mechanical - 1 Unit

AN-MTI-001 A

APPLICATION NOTE
FOR
MULTIPLEX TERMINAL UNIT
(MTI-110)

SCI PART NO. 4199000-1

APRIL 1976

Revised February 1977

SCI SYSTEMS, INC.
2600 SOUTH MEMORIAL PARKWAY
HUNTSVILLE, ALABAMA 35802

TABLE OF CONTENTS

	<u>Page</u>
1.0 GENERAL DESCRIPTION	1
1.1 MECHANICAL DESCRIPTION	1
1.2 TIMING DIAGRAMS	6
1.3 INPUTS AND OUTPUTS	6
1.4 POWER REQUIREMENTS	6
2.0 DATA LINK INTERFACE	11
3.0 RECEIVER OPERATION	14
3.1 EXTERNAL CLOCK (E16)	14
3.2 RECEIVER THRESHOLD ADJUST (E20, E30)	14
3.3 RESET (E25)	14
3.4 DATA READY (E54)	15
3.5 RECEIVED SYNC TYPE (E59)	15
3.6 DATA ENABLES (E21, E49)	15
3.7 PATTERN ERROR (E51)	15
3.8 PARITY ERROR (E50)	16
3.9 FLAG AND EOM (E46, E60)	16
3.10 SYNC DETECT (E57)	16
3.11 BUS ACTIVE (E56)	16
3.12 RECEIVED NRZ AND RECEIVED CLOCK (E53, E55)	16
4.0 TRANSMITTER OPERATION	18
4.1 TRANSMITTER A SELECT (E10)	18
4.2 ENABLE SHUTDOWN (E52)	18
4.3 TRANSMITTER TIMEOUT (E40)	18
4.4 SERIAL MODE SELECT (E23)	19
4.5 TRANSMIT COMMAND (Z6)	19

TABLE OF CONTENTS (Continued)

4.6	TRANSMIT SYNC TYPE (E31)	19
4.7	PARALLEL DATA LOAD 1 AND 2 (E17, E48)	20
4.8	TRANSMIT NR Z (E27)	20
4.9	ONE MEGAHERTZ CLOCK (E14)	20
4.10	DATA REQUEST (E28)	21

LIST OF FIGURES

<u>Figure</u>	<u>Title</u>	<u>Page</u>
1	MTI-110 Block Diagram	2
2	Received Data Timing	7
3	Transmitted Data Timing	8
4	Receiver Input External Strapping Options	12
5	MTI-110 Receiver Input Transformer Connections	13
6	Recommended Circuit for Changing Receiver Serial Word Length	17
Plate M954	Multiplex Terminal Unit MTI-110	3
Plate M956	MTI-110 Multiplex Terminal Unit (Disassembled)	4
AN-MTI-001	MTI-110 Outline Drawing	5
Table I	MTI-110 Pin Function List	9
Table II	MTI-110 Non-TTL Pin Functions	10

1.0

GENERAL DESCRIPTION

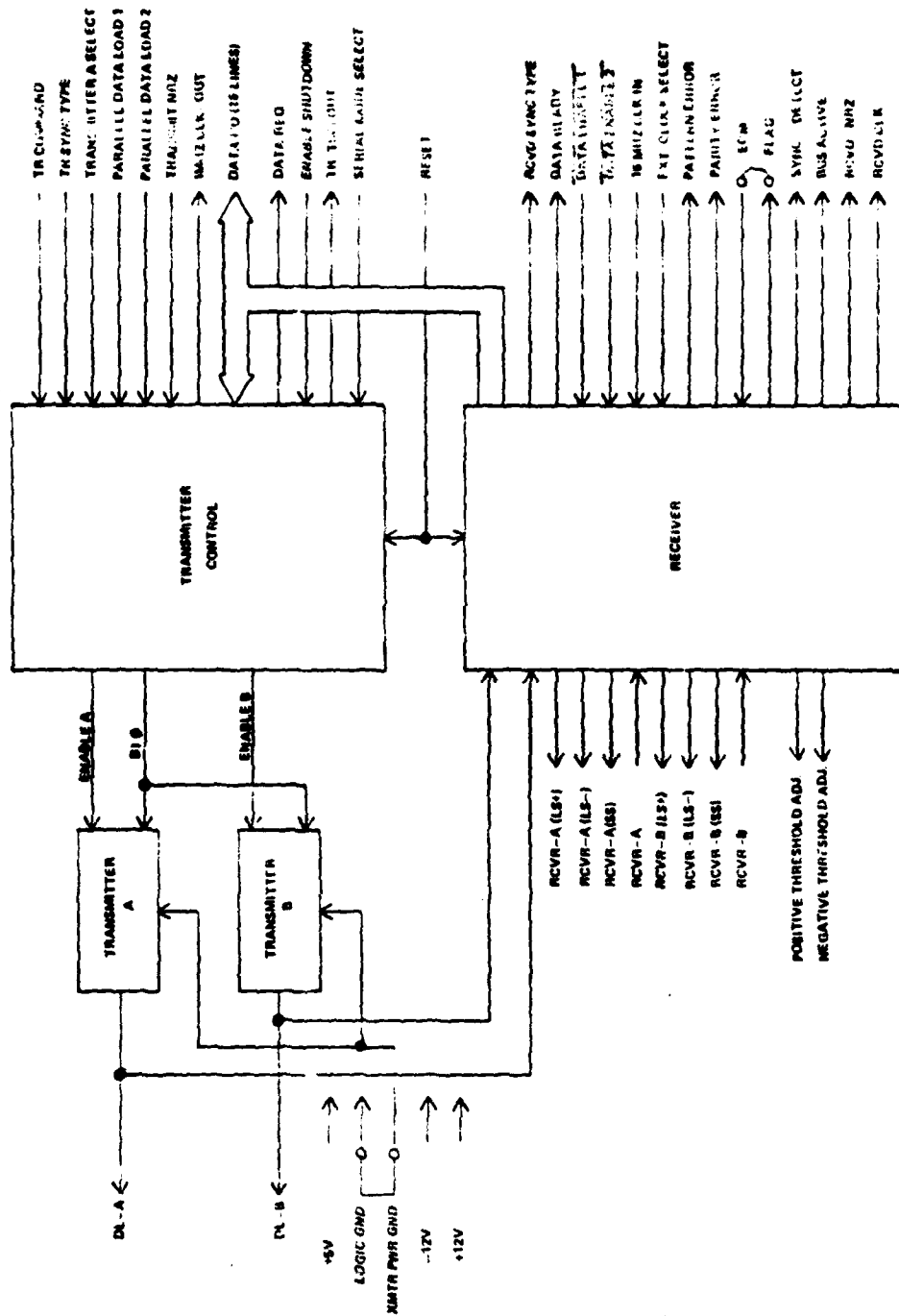
The MTI-110 is a multiplex data bus interface module designed for aircraft subsystems signal transfer. It provides a two way interface between TTL logic and dual redundant twisted pair transmission lines utilizing serial bi-phase data transmission as specified by MIL-STD-1553A and MIL-G-85013. The MTI is capable of operating on either of the two redundant busses on command of the user. A block diagram of the MTI-110 is shown in Figure 1. Data appearing on the data bus is received by the MTI-110 and presented both in serial and in parallel form along with control signals and data validity signals. The receiver parallel output is available as 3-state outputs and can be enabled in 2 - 8 bit bytes or 1 - 16 bit word. The receiver is operated from a 16 MHz clock which is available internally or may be provided externally by the user.

Data to be transmitted on the data bus is accepted by the MTI-110 either in serial or in parallel form and may be transmitted on either of 2 data busses. Although the MTI-110 is designed specifically to meet MIL-STD-1553A which calls for a 16 bit data word plus parity, it can operate in the serial mode with any word length as long as each word begins with a MIL-STD-1553A sync pattern and ends with a parity bit.

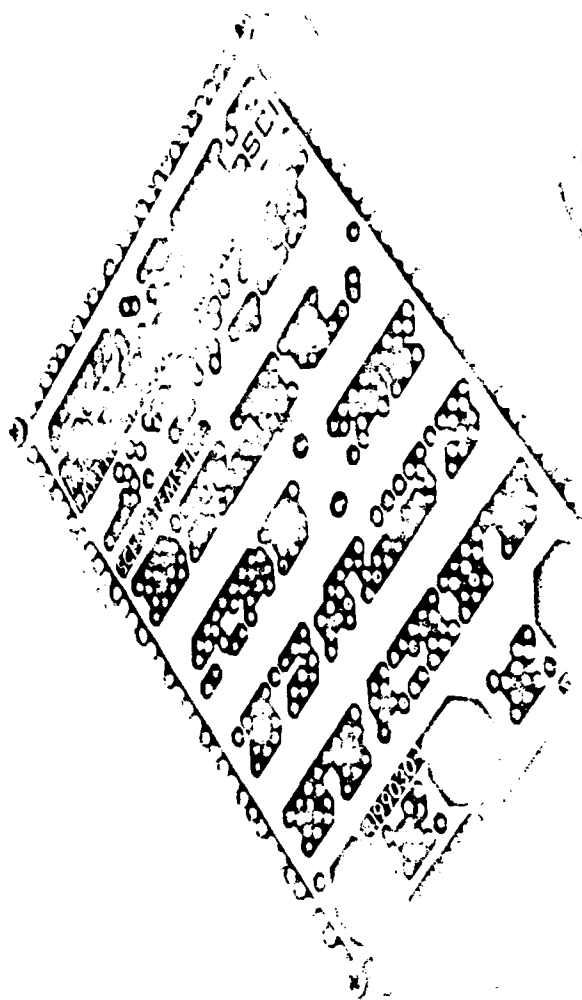
1.1

MECHANICAL DESCRIPTION

The complete MTI-110 module is shown in Plate M954. It consists of two multilayer printed circuit boards held together as a module by the use of screws and spacers. All inter-board connections are by spring-type pin connectors as shown in the disassembled view (Plate M956). Standard flat pack type integrated circuits are used throughout. There are no custom hybrid, LSI or other custom circuits. User connections are by the use of 62 pins arranged in a dual in line pattern. The complete module measures 3 x 4 x 0.375 and weighs 3.5 oz. Mechanical details are shown in Drawing AN-MTI-001.



MTI-110 BLOCK DIAGRAM
FIGURE 1



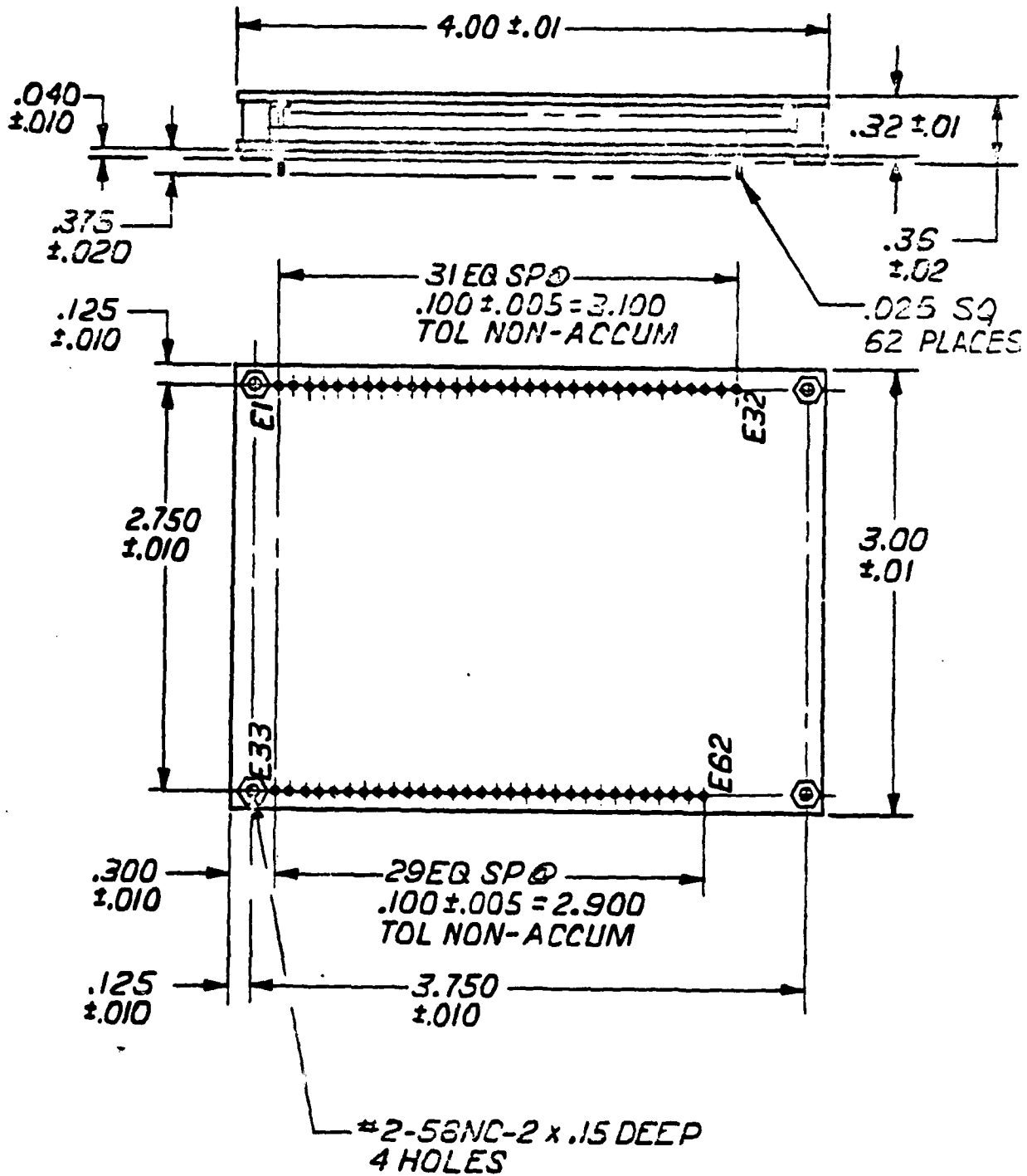
M954

MULTIPLEX TERMINAL UNIT MTI - 110

M956

MTI - 110 MULTIPLEX TERMINAL UNIT (DISASSEMBLED)





CODE IDENT. NO. 17981	DWG. SIZE A	AN-MTI-001	
		OF	REV

1.2 TIMING DIAGRAMS

Detailed relationships among the various interface signals involved in I/O operations are shown in the timing diagrams of Figures 2 and 3. In these diagrams each signal or group of signals is represented by a horizontal line with a raised section. In the case of a control signal that is generated at a specific time to control some particular function, the raised section represents the time that the function is true. For signals that carry binary information such as the data I/O line and the sync type lines, the raised section indicates the time during which that information is held on the input or output line.

1.3 INPUTS AND OUTPUTS

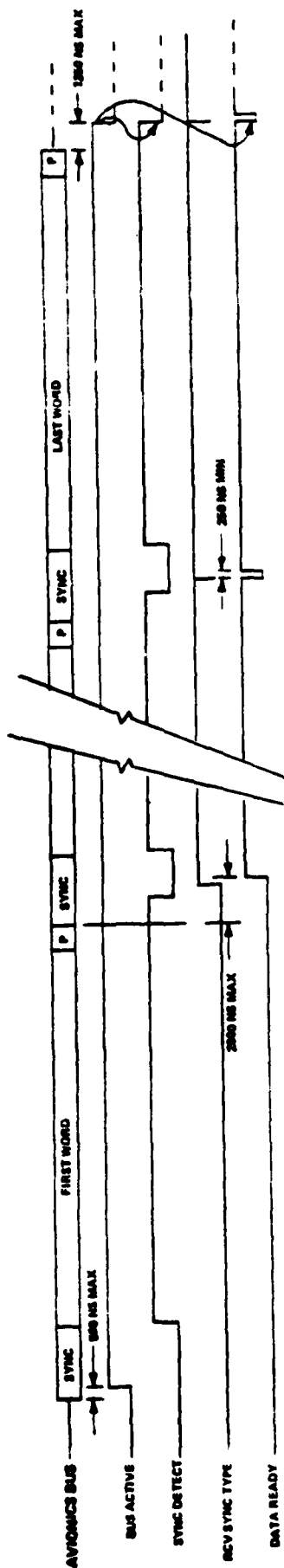
All MTI-110 inputs and outputs are shown in the pin function list (Table I). All inputs and outputs are standard TTL compatible except for those given in Table II. TTL outputs will drive two standard TTL loads minimum and inputs are equivalent to one low power Schottky TTL load, except for the external clock input which is equivalent to 10 low power Schottky TTL loads.

1.4 POWER REQUIREMENTS

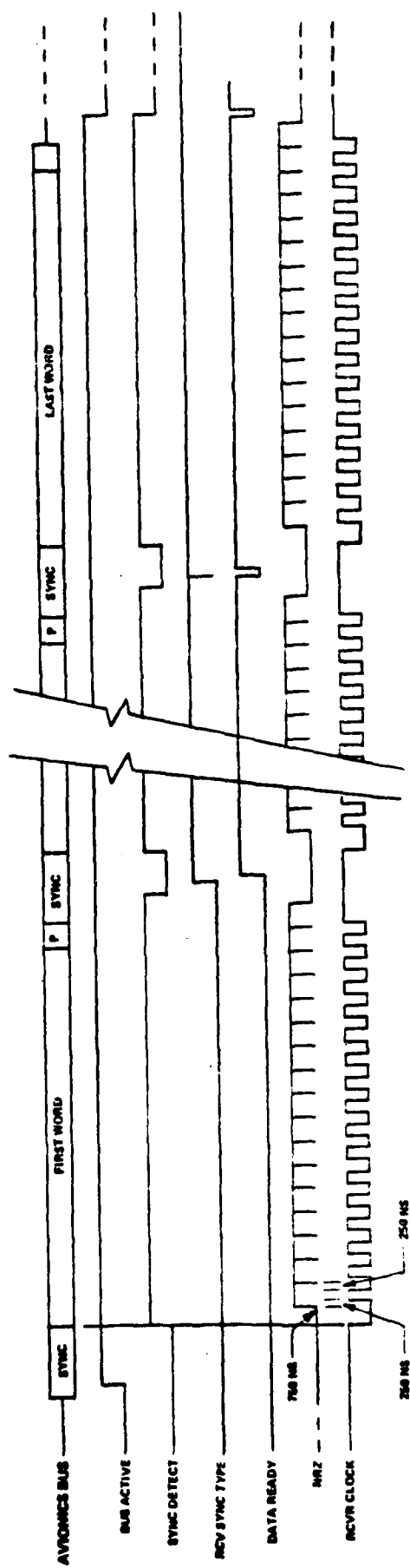
The MTI-110 requires three supply voltages: +5 volts, +12 volts and -12 volts. While the MTI-110 will operate with up to +15 volts, all current drain and signal threshold specifications are based on the use of a +12 volt supply. Current requirements are as follows:

+5 Volts	600 ma max
+12 Volts (non-transmitting)	30 ma max
(transmitting - long stub)	650 ma max
(transmitting - short stub)	300 ma max
-12 Volts	30 ma max

Voltage tolerance is $\pm 5\%$.

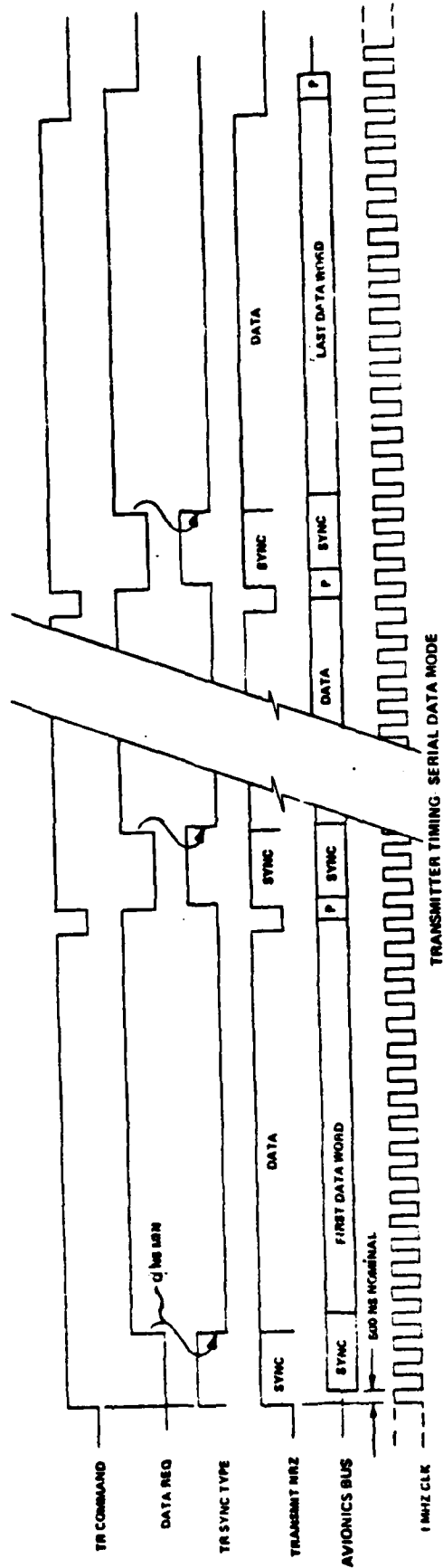
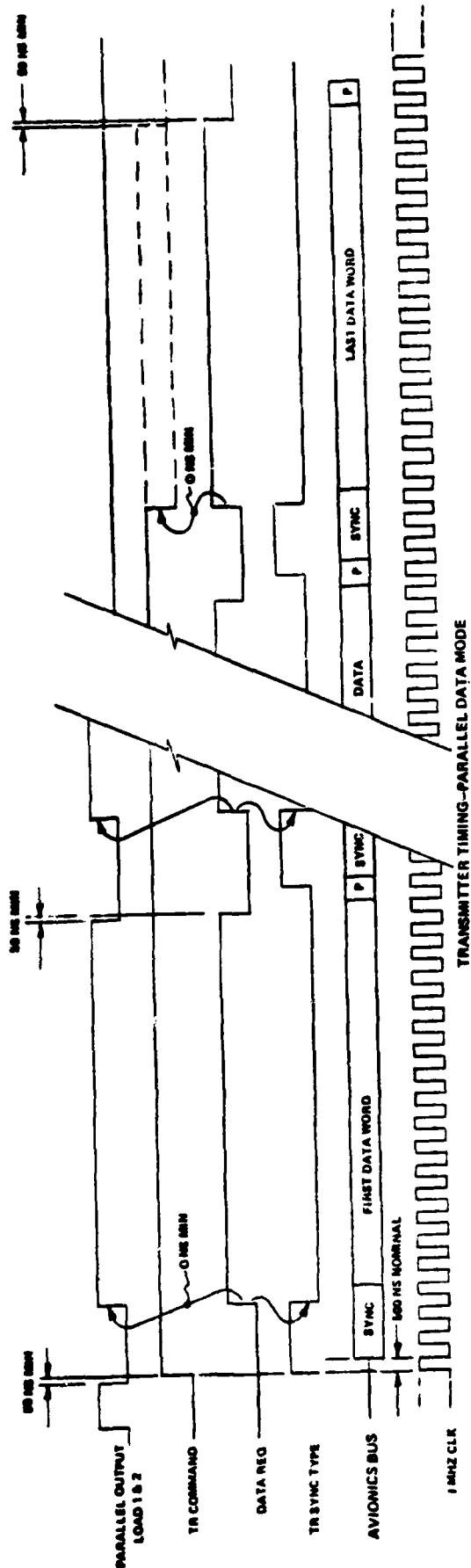


RECEIVED DATA - PARALLEL MODE



RECEIVED DATA - SERIAL MODE

RECEIVED DATA TIMING
FIGURE 2



TRANSMITTER DATA TIMING
FIGURE 3

TABLE I
MTI 110 PIN FUNCTION LIST

<u>Terminal</u>	<u>Function</u>	<u>Terminal</u>	<u>Function</u>
1	data link B (+)	32	+5 volts input
2	data link B (-)	33	data link A (+)
3	receiver input B (LS+)	34	data link A (-)
4	receiver input B (SS)	35	receiver input A (LS+)
5	data I/O 0 (LSB)	36	receiver input A (SS)
6	data I/O 4	37	data I/O 15 (MSB)
7	data I/O 2	38	transmitter power ground
8	data I/O 5	39	data I/O 13
9	data I/O 3	40	transmitter timeout
10	transmitter A select	41	data I/O 10
11	data I/O 1	42	data I/O 8
12	+12 volts input	43	data I/O 11
13	data I/O 6	44	data I/O 9
14	1 MHz clock output	45	data I/O 14
15	data I/O 7	46	flag output
16	external clock input	47	data I/O 12
17	parallel data load 1	48	<u>parallel data load 2</u>
18	external clock select	49	data enable 2
19	-12 volts input	50	parity error output
20	<u>negative threshold adjust</u>	51	pattern error output
21	data enable 1	52	enable shutdown input
22	receiver input B	53	received NRZ output
23	serial mode select	54	data ready output
24	receiver input A	55	received clock output
25	reset input	56	bus active output
26	transmit command input	57	sync detect output
27	transmit NRZ input	58	receiver input A (LS-)
28	data request output	59	received sync type output
29	receiver input B (LS-)	60	end of message (EOM) input
30	positive threshold adjust	61	-----
31	transmit sync type input	62	logic ground

TABLE II

MTI-110 NON-TTL PIN FUNCTIONS

<u>Function</u>	<u>Pin Number</u>
data link B (+)	1
data link B (-)	2
receiver input B (LS+)	3
receiver input B (SS)	4
+12 volts input	12
-12 volts input	19
negative threshold adjust	20
receiver input B	22
receiver input A	24
receiver input B (LS-)	29
positive threshold adjust	30
+5 volts input	32
data link A (+)	33
data link A (-)	34
receiver input A (LS+)	35
receiver input A (SS)	36
transmitter power ground	38
receiver input A (LS-)	58
logic ground	62

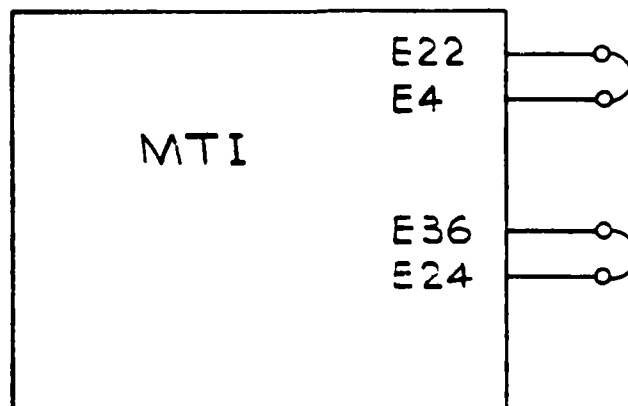
2.0

DATA LINK INTERFACE

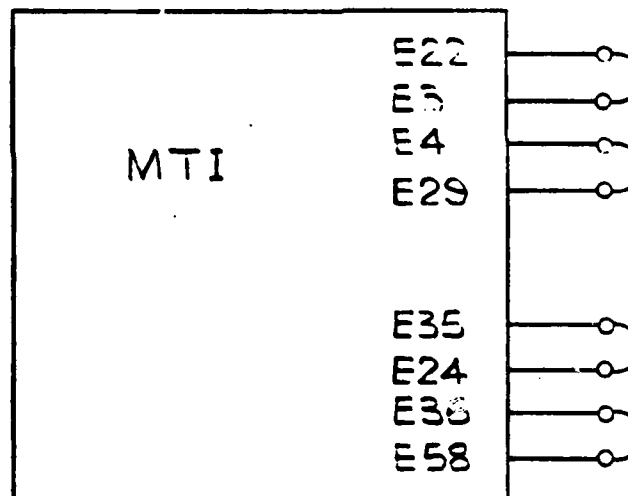
The MTI-110 includes provisions for two MIL-STD-1553A data links. Either or both data links may be connected. Data may appear on only one data link at a time. If only one data link is used the other must be terminated with 70 ohms maximum. The MTI is designed for use in either the long stub or short stub configuration as described by MIL-STD-1553A. In either configuration external jumpers must be used to connect the receiver input as shown in Figure 4.

Figure 5 shows the MTI-110 receiver input transformer with external strapping options and stub configurations shown. The receiver A input is shown in the short stub configuration and receiver B is connected for long stub operation. The two receiver input circuits are identical and may be strapped independently for either configuration. There are three windings on the MTI-110 coupling transformer which are associated with the receiver input. Winding 1 is the input from the data link and windings 2 and 3 provide the input to the receiver. For MIL-STD-1553A short stub operation, winding 3 provides the receiver input and winding 2 is unused as shown in Figure 5A. External isolation resistors must be used as shown.

For long stub operation, windings 2 and 3 are connected in series as shown in Figure 5B and a MIL-STD-1553A data link coupler must be used between the bus and the stub. The data link coupler (SCI Model DLC-10 or equivalent) contains the isolation transformer and proper isolation resistors to meet all the requirements of MIL-STD-1553A for long stub operation. If a different coupling transformer is used, the source impedance of the coupler as it is connected to the MTI must be 50 ohms minimum or overloading of the MTI transmitter will result.



SHORT STUB OPERATION OR LONG
STUB WITH 1:1 COUPLING TRANSFORMER.



LONG STUB OPERATION WITH
3:1 COUPLING TRANSFORMER.

FIGURE 4. RECEIVER INPUT EXTERNAL STRAPPING OPTION.

CODE IDENT NO 17881	DWG SIZE A	
------------------------------	------------------	--

AD-A119 133

LITTON SYSTEMS INC VAN NUYS CALIF DATA SYSTEMS DIV
CONTROL DISPLAY UNIT PROGRAM.(U)
1978

F/6 9/2

UNCLASSIFIED

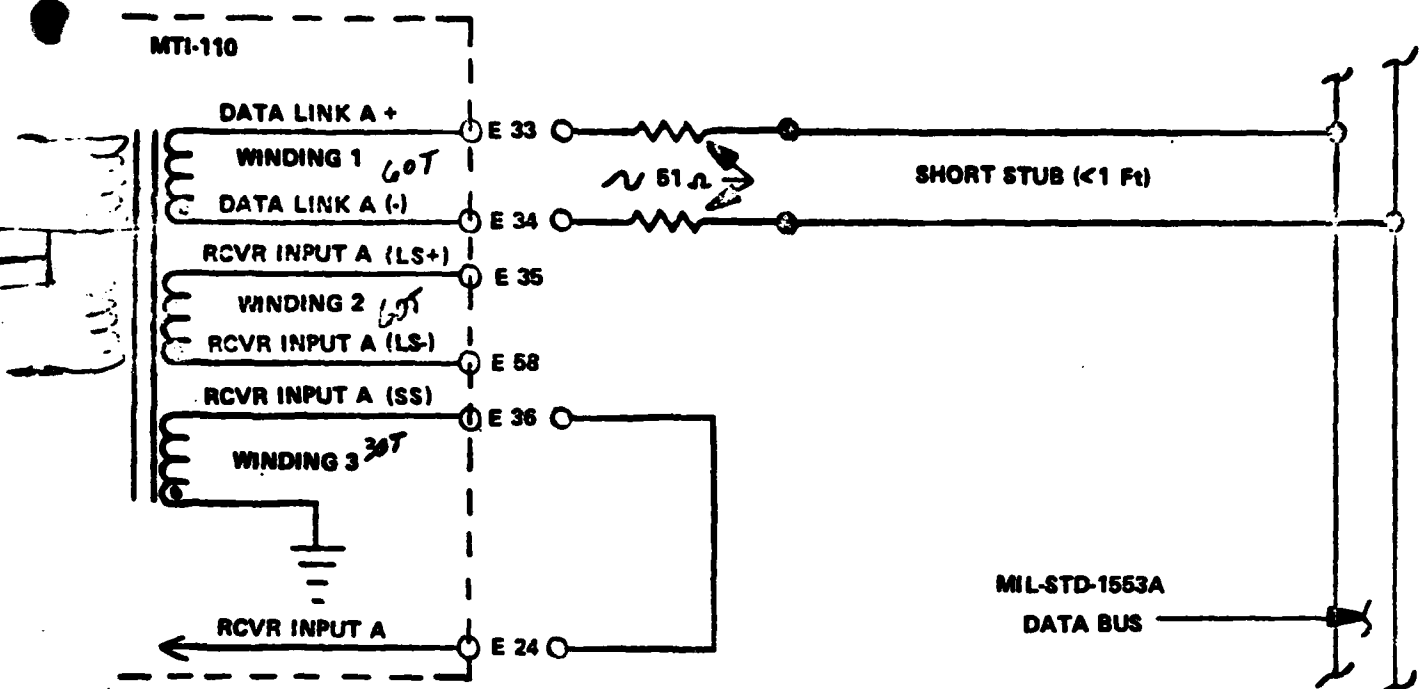
NL

3 OF 3

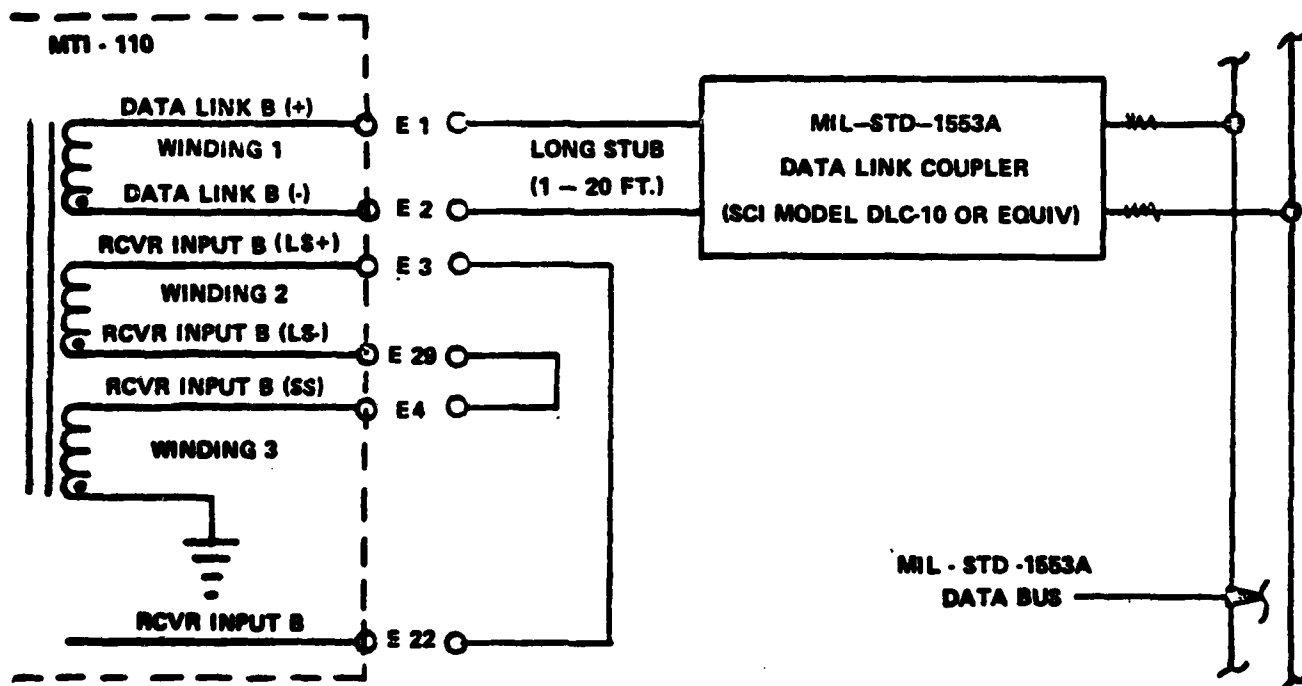
AD A

19132

END
DATE
FILMED
10-82
DTIC



A. SHORT STUB CONFIGURATION (RECEIVER INPUT A SHOWN)



B. LONG STUB CONFIGURATION (RECEIVER INPUT B SHOWN)

FIGURE 5. MTI - 110 RECEIVER INPUT TRANSFORMER CONNECTIONS

3.0 RECEIVER OPERATION

The MTI-110 features an all-digital biphasic receiver design which provides high stability without the requirement for special bench alignment. A complete description of user options and external connections pertaining to receiver operation is provided in this section.

3.1 EXTERNAL CLOCK (E16)

The MTI normally operates from an internal 16 MHZ clock but may be operated from an external 16 MHZ clock at the user's option. This is done by presenting a 16 MHZ TTL level clock to the external clock input (E16) and leaving the external clock select input (E18) open. For internal clock operation the external clock select input must be grounded.

3.2 RECEIVER THRESHOLD ADJUST (E20, E30)

The MTI receiver thresholds are factory set at +0.1V and -0.1V at the detector input, and these factory settings are compatible with MIL-STD-1553A. The threshold points are available for external adjustment and may be adjusted over the range of $\pm 0.075V$ to $\pm 0.2V$ by injecting a current into the threshold adjustment node. The input impedance of each node is 100 ohms.

3.3 RESET (E25)

A logic one on the reset input sets the receiver data ready to zero, turns off the transmitter and initializes the transmitter shutdown circuit. A logic one should be applied momentarily to the reset input when the MTI is being powered up in order to insure proper operation. The reset should remain high until power lines are stable. If the transmitter failsafe timer causes the transmitter to shut down, the reset must be reapplied before the transmitter is operated again. Minimum pulse width on the reset input is 100 nanoseconds.

3.4 DATA READY (E54)

Data ready goes high after a word has been received on the data link. Each time a new word is received on the data link the data ready line goes low for a period greater than 250 nanoseconds and less than one microsecond and then goes high again. Data ready remains high until a new word is received. Data, received sync type, pattern error, and parity error outputs may be read at any time while data ready is high.

3.5 RECEIVED SYNC TYPE (E59)

Received sync type indicates whether the received word was a command word or a data word. Received sync type is low for a command word and high for a data word. Received sync type may be read when data ready is high.

3.6 DATA ENABLES (E21, E49)

The 16 receiver output data lines are three-state. They are enabled in eight bit bytes by data enable 1 and data enable 2. Data enable 1 enables the least significant byte and data enable 2 enables the most significant byte. A low on the enable line enables the output lines. The propagation delay from enable to output is less than 100 nanoseconds.

3.7 PATTERN ERROR (E51)

The pattern error output is high if the received data word contains a Manchester coding violation or an invalid word length. A high output indicates an error. The pattern error output may be read when data ready is high.

3.8 PARITY ERROR (E50)

The parity error output is high if the received data word contains even parity and low for odd parity. The parity error output may be read when data ready is high.

3.9 FLAG AND EOM (E46, E60)

The flag output must be connected to the EOM input for normal MIL-STD-1553A operation. These signals may be used in conjunction with external circuitry to change the receiver word length for serial data output only. Figure 6 shows the external circuitry required to change the receiver word length. Parallel data and parity are available at the outputs of the flip-flops shown in the figure. These outputs are valid while the Q output of FF1 is high.

3.10 SYNC DETECT (E57)

The sync detect output goes high when a valid sync pattern is detected on the data link. The sync detect output goes low again when the first half of a sync pattern is detected or when the data link becomes inactive.

3.11 BUS ACTIVE (E56)

Bus active goes high 250 nanoseconds after a signal appears on the data link and goes low one microsecond after the data link goes inactive.

3.12 RECEIVED NRZ AND RECEIVED CLOCK (E53, E55)

The received NRZ output contains the serial NRZ data and parity. The received clock output is a 17 pulse clock output which is derived from the received biphasic data. The NRZ data bits may be read on the rising edge of the received clock.

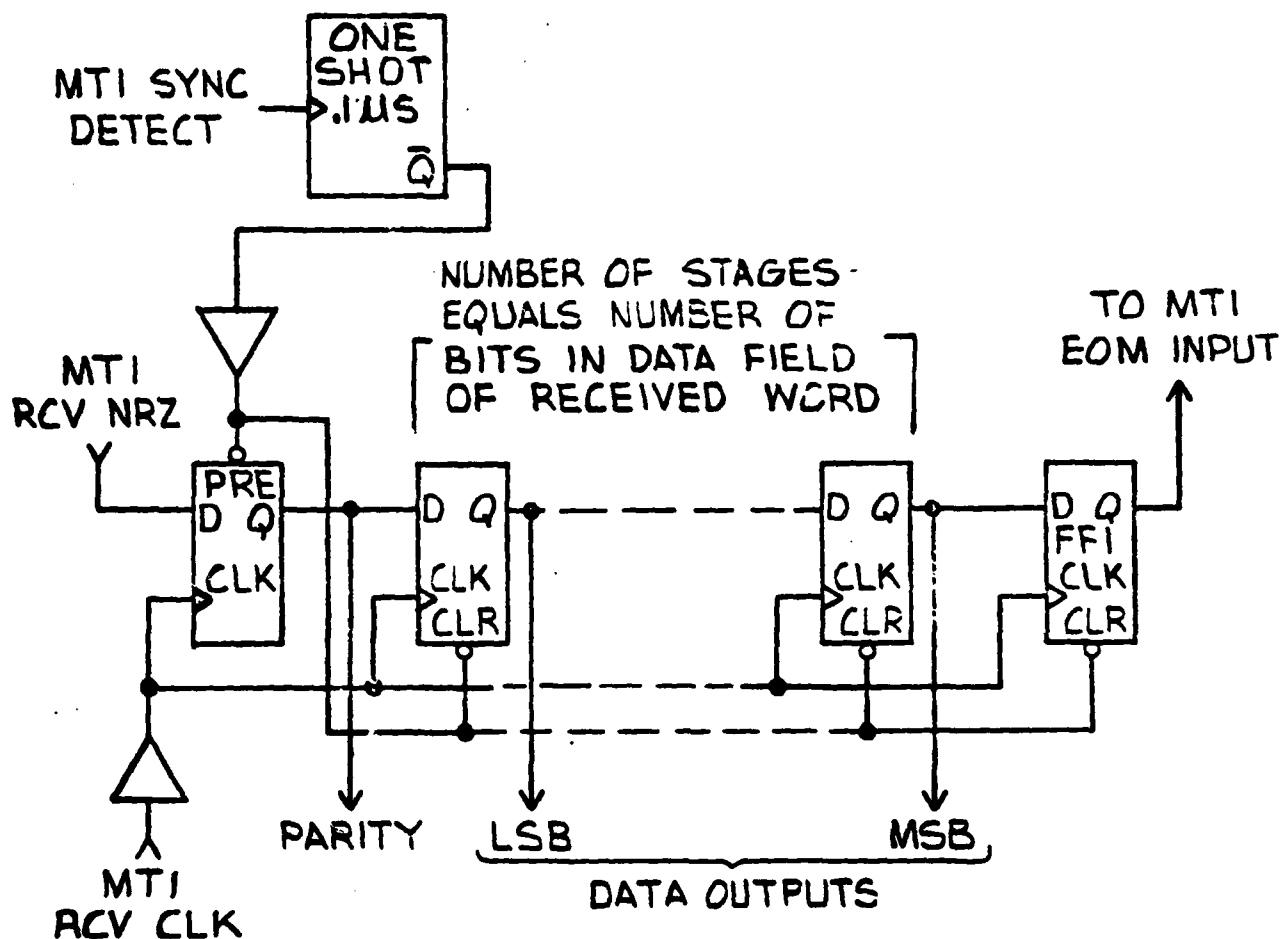


FIGURE 6. RECOMMENDED CIRCUIT FOR CHANGING RECEIVER SERIAL WORD LENGTH

CODE IDENT. NO. 17001	DWG. SIZE A		
		OF	REV

4.0 TRANSMITTER OPERATION

The MTI-110 transmitter allows operation on either of two MIL-STD-1553A data busses. Two complete transformer-coupled interfaces are provided, allowing the MTI to transmit on either bus at the command of the user. A complete description of user options and external connections pertaining to transmitter operation is provided in this section.

4.1 TRANSMITTER A SELECT (E10)

The MTI transmitter is capable of transmitting on either of two data links. When the transmitter A select input is high the data will be transmitted on data link A and when the transmitter A select input is low data will be transmitted on data link B.

4.2 ENABLE SHUTDOWN (E52)

The MTI incorporates a timer which times all transmissions. When the enable shutdown input is high the MTI transmitter will be shutdown automatically if the timer senses a transmission period exceeding 672 microseconds. A reset input must be applied in order to reset the shutdown circuit to allow subsequent transmissions to occur. If the enable shutdown input is low the timer will not shutdown the transmitter.

4.3 TRANSMITTER TIMEOUT (E40)

The transmitter timeout output is a positive pulse which occurs when the transmitter timer senses that the MTI has transmitted continuously for more than 672 microseconds. Minimum pulse width on this output is 100 nanoseconds.

4.4 SERIAL MODE SELECT (E23)

The MTI transmitter will accept data in either serial or parallel form. When the serial mode select input is high the MTI transmitter accepts serial NRZ data from the transmit NRZ input. When the serial mode select input is low the MTI transmitter accepts parallel data from the 16 data I/O lines.

4.5 TRANSMIT COMMAND (E26)

The MTI transmitter is turned on by the transmit command input. A high input turns the transmitter on and a low input turns the transmitter off. For proper operation the transmit command must go high within 0 to 300 nanoseconds after the rising edge of the MTI one megahertz clock output. The transmission begins on the falling edge of the one megahertz clock output. When the transmitter is operating in the parallel data input mode the TR command must remain high until the last word to be transmitted, then it must go low while the MTI data request output is high. The transmission will then be terminated at the end of the current word. When the transmitter is operating in the serial data input mode all transitions of the transmit command must occur within 0 to 300 nanoseconds after the rising edge of the MTI one megahertz clock output. The transmit command is required to go low during the last bit period (parity) of each word. During the last word to be transmitted the TR command goes low at the beginning of the last bit period and remains low. The word length of the transmitted message in the serial data input mode is determined by the transmit command input.

4.6 TRANSMIT SYNC TYPE (E31)

The transmit sync type input determines the polarity of the sync pattern of each word transmitted by the MTI. When the transmit sync type input is high a

command sync pattern will be generated and when it is low a data sync pattern will be generated. The sync type information must be present on the transmit sync type input during the time that the data request output is low.

4.7 PARALLEL DATA LOAD 1 AND 2 (E17, E48)

In the parallel data input mode the data present on the 16 I/O data lines is loaded into the MTI transmitter input register in 8 bit bytes by the parallel data load inputs. Data is loaded on the rising edge of each parallel load input. The data must be present 100 nanoseconds before the rising edge occurs and must remain until 10 nanoseconds after the edge occurs. Parallel data load 1 loads the least significant byte of data and parallel data load 2 loads the most significant byte of data. The first word of a message to be transmitted must be loaded before the transmit command goes high. All succeeding words are loaded during the time when the data request output is high. One word is loaded during each high interval of the data request output.

4.8 TRANSMIT NRZ (E27)

When the MTI transmitter is operating in the serial data input mode the serial data and parity to be transmitted must be present at the transmit NRZ input. The first bit of each data word to be transmitted is taken by the transmitter on the rising edge of the data request output. All succeeding bits of the word are taken on succeeding rising edges of the one megahertz clock output. Each bit of data to be transmitted must be present 100 nanoseconds before the edge on which it is taken and must remain for 10 nanoseconds after the edge occurs.

4.9 ONE MEGAHERTZ CLOCK (E 14)

The one megahertz clock output is provided for the user for input data and control input synchronization. Its use is explained in the appropriate paragraphs.

4.10 DATA REQUEST (E28)

All data must be loaded into the MTI transmitter when the data request is high. Use of the data request output is explained in more detail in the appropriate paragraphs.

1 August 1979

ATTACHMENT A

ENGINEERING DESIGN TEST PLAN

ECOM CDU DISPLAY AND CONTROL FORMATS

The final formats for the ECOM CDU
are detailed in this document.

~~82 08 18 000~~

USE OR DISCLOSURE OF THIS
DATA IS SUBJECT TO THE
RESTRICTION ON THE TITLE
PAGE OF THIS DOCUMENT

Primary Format

The Primary Format for the CDU is shown in Figure 1. This format is both a control and display format in that each piece of equipment displayed in this format can be selected for use by pressing the equipment designator (i.e., [VHF]) which is a switch point and will change the display from the Primary Format display to the Submode Display for the selected equipment.

The status of each piece of equipment will be shown by three methods. Equipments that have not reported a malfunction and are considered operable, will have the frequency they are tuned to displayed in line with the designator in the case of communications equipment and the mode they are in, in the case of DNV and IFF equipment.

If the equipment is active and in use, an asterisk will be displayed between the equipment designator and the frequency or mode as illustrated for the VHF, DNV and IFF in Figure 1.

If the equipment is inoperative by failing its internal test or is not installed, a [FAIL] will be displayed in place of the frequency or mode as shown for ADF in Figure 1. 'NO KIT' will be displayed with the IFF, if a KIT is not installed or is faulty.

In all cases, selection of a radio or other unit, by pressing the equipment designator will display the Submode format for that equipment as follows:

Table I

Select	Display
VHF	Figure 2
UHF	Figure 3
ADF	Figure 4
CNV	Figure 5
DNV	Figure 6
IFF	Figure 7

Ten fixed switches will be provided below the display area of the CDU. These switches will have the following function:

PRI - Activation of this switch will return the display to the Primary Format, Figure 1.

IFF EMER - This switch is guarded and requires a sequence of operation to activate the function as follows: First, press [IFF EMER], then [GARD], and then [IFF EMER] again to select the IFF emergency function.

ZERO CODE - This switch is guarded and requires a sequence of operation to activate the function as follows: First, press [ZERO CODE], then [GARD], and then [ZERO CODE] again, to select this function.

ON - The ON switch is used to turn on the CDU and each equipment on as follows:

- A. With the CDU in the OFF state, activation of [ON] will apply power to the CDU and display the Primary Format, Figure 1.
- B. With the CDU on and the Primary Format displayed, activation of an equipment designator ([VHF] etc.) will display the Submode Format for the equipment as shown in Table 1. Activation of [ON] after activation of equipment designator will then turn the equipment on, with the exception of the IFF. The IFF is put into the [STBY] or [NORM] operation by selection of either respective switch.

OFF - The [OFF] switch is used to turn the CDU and each equipment off as follows: With the CDU on and the Primary Format displayed, selection of an equipment designator will display the equipment Submode Format shown in Table 1 and then selection of [OFF], will turn the equipment off. The displayed asterisk symbol (ON indicator) will be erased from the Primary Format. Each of the equipments, with the exception of the CDU and the IFF, will be turned off in this manner. The CDU and the IFF will be turned off, using the guarded switch function as follows:

- A. **IFF OFF** - Select [IFF] from the Primary Format, (IFF Submode Format will display), activate [OFF], then [GARD], then [OFF]. This will turn the IFF subsystem off.

- B. **CDU OFF** - With the Primary Format displayed, select [OFF], then [GARD], and then [OFF] again to turn the CDU off.

SUB MODE - Activation of **SUB MODE** will return the display to the Submode Format (Table 1), when a subset of the Submode is being displayed. Activation of **SUB MODE** when the Primary Format, or any Submode Format (Table 1) is displayed, will have no effect on the display or equipment.

RPLY - Activation of the [RPLY] switch will send a message to the IFF subsystem via the 1553A bus interface. It will have no effect on the display.

GARD - The [GARD] switch is used as described previously.

LAST - This switch is used in the communications formats and DNV formats. When activated with a VHF or UHF format displayed, it will change the frequency selection from the present active frequency to the prior active frequency. When activated with the DNV Formats, the last Format used will be displayed.

STEP - This switch is active in the preset channel format, TGT Format and Check-point Formats only. Activation of [STEP] will cause the next channel number and frequency to be available for change if needed.

V H F * 4 2 . 7 5					D N V * DNV* NAV				
T / R					CKPT 4				
					RGE 163.5 KM				
					BRG 273				
					TTG 50.5M				
U H F 1 1 8 . 3 7 5					I F F * 1 2 3 A C 4 A				
T / R									
A D F F A I L									
C N V 1 1 0 . 9 5									
P P 1 5 S U P 1 4 1 7 8 4 0 9									
PRI		IFF EMER		ZERO CODE		OFF		ON	
SUB MODE		RPLY		GARD		LAST		STEP	

FIXED
FUNCTIONIC
SWITCHE

Figure 1. Primary Format CNI CDU

CAPABLE OF BLINKING ON FINAL MODEL.

VHF		CH-4		42.75	
ON		T/R			
} LEGEND					
} LINE					
CHAN SEL		MAN FREQ		PRST CHAN	
STAT PAGE					
T/R		T/R+ GARD		HOM	
RE TRAN					
		TEST			

LEGEND
LINE

Figure 2. VHF Submode

ON or OFF will display in second line under VHF for indication of ON/OFF status.

	<u>Switch</u>	<u>Format</u>	
From:	VHF	Figure 1	
To:	[CHAN SEL]	Figure 2. A	
	[MAN FREQ]	Figure 2. B	
	[PRST CHAN]	Figure 2. C	
	[STAT PAGE]	Figure 2. D	
	[T/R]	Figure 2	T/R will display in top line
	[T/R + GARD]	Figure 2	T/R + G will display on second line
	[HOM]	Figure 2	HOM will display in top line
	[RETRAN]	Figure 2	RETRAN will display in top line
	[TEST]	Figure 2	OK or FAIL will display on second line

VHF			CH-4	42.75
ON			T/R	
CH- 7			42.95	1 2 3
CHAN SEL			4	5 6
			7	8 9
			C	Ø E

Figure 2.A. VHF Channel Select Format

	<u>Switch</u>	<u>Format</u>
From:	[CHAN SEL]	Figure 2
To:	[E]	Figure 2

This format is used to select the active channel in the following manner:

1. Selection of a numeric from the keyboard will display that numeric in the legend line following CH-. The associated frequency stored in that channel if any will appear in the legend line.
2. [C] may be used to clear the entry for corrective purposes.
3. Selection of [E] will enter the change indicated in the legend line to the radio and switch the display back to Figure 2 where the selected channel number and its associated frequency are displayed in the first line. Legal entries from this format are 0 through 9. [CHAN SEL] switch is not active in this format.

VHF		CH-4		42.75	
ON		T/R			
F- --.---			1	2	3
	MAN FREQ		4	5	6
			7	8	9
			C	Ø	E

Figure 2.B. Manual Frequency Select Format

	<u>Switch</u>	<u>Format</u>
From:	MAN FREQ [FREQ SEL]	Figure 2
To:	[E]	Figure 2

This format is used to change or enter the active frequency in the following manner:

1. A frequency is entered in the legend line by selecting from the keyboard. Legal entries are from 30.00 to 75.95.
2. For corrective purposes, a single activation of [C] will clear the last entered digit and a double activation will clear all of the entered numerics.
3. Activation of [E] will place the entered frequency into use and return the display to the VHF Submode Format, Figure 2. The [MAN FREQ] switch is not active in this format.

VHF		CH-4		42.75	
ON		T/R			
CH- _ F- _ _ . _ _			1	2	3
		PRST CHAN	4	5	6
			7	8	9
			C	Ø	E

Figure 2. C. Preset Channel Select

Switch

Format

From: [PRST CHAN] Figure 2
To: [E] Figure 2

This format is used to preassign frequencies to channels in the following manner:

1. A channel number 0 - 9 is entered in the legend line by selecting from the keyboard. The associated frequency, if any, will also appear in the legend line.
2. The frequency to be assigned to the channel is next entered by use of the keyboard. Legal entries are from 30.00 to 75.95.
3. For corrective purposes, a single action of [C] will clear the last entered digit and a double activation will clear all of the entered numerics.
4. Activation of [E] will place the channel assignment in memory and return to this page with the next channel digit displaced in the legend line. It will continue to step after each entry. Selecting [PRI] or [SET] will return the display to its respective condition. [MODE]
5. The channel with its assigned frequency will be displayed on the Preset Channel Format, Figure 2. D, when that format is displayed. [PRST CHAN] is not active in this format.

VHF CHAN STATUS					1/2
0	42.73	3	*	42.75	
1		4		31.25	
2		5			

Figure 2. D. Stat Page

	<u>Switch</u>	<u>Format</u>
From:	[STAT PAGE]	Figure 2
To:	0 thru 9	Figure 2.D

This format is primarily a review or status display of the preset channels. An active channel is indicated by the asterisk symbol. A channel can be selected for re-assignment in the following manner:

1. Select channel for re-assignment by pressing on channel legend.
2. Manual frequency select format, Figure 2. B will display with the selected channel and it's associated frequency, if any, displayed in the legend line.
3. A new assignment may be made by entering a new frequency from the keyboard. The channel entry area in the legend line will not be changeable when accessed from the [STAT PAGE] Format.

UHF CHAN STATUS		2/2
6	9	
7		
8		

Figure 2.E. Stat Page

UHF		CH-6	118.315
OFF		T/R	
CHAN SEL	MAN FREQ	PRST CHAN	STAT PAGE
T/R	T/R+ GARD	ADF	GARD
SQL OFF	SQL ON	TEST TONE	

Figure 3. UHF Submode Format

	<u>Switch</u>	<u>Format</u>	
From:	UHF	Figure	
To:	CHAN SEL	Figure 3.A	
	MAN FREQ	Figure 3.B	
	PRST CHAN	Figure 3.C	
	STAT PAGE	Figure 3.D	
	T/R	Figure 3	T/R will display in first line
	T/R + GARD	Figure 3	T/R & GARD will display in first line
	ADF	Figure 3	ADF will display in first line
	GARD	Figure 3	GARD will display in first line
	SQL ON	Figure 3	SQL will display in second line
	SQL OFF	Figure 3	SQL will be erased from second line
	TONE (Momentary)	Figure 3	A tone will be heard in the headset.

UHF		CH-6		118.315	
ON		T/R			
CH- _			1	2	3
CHAN SEL			4	5	6
			7	8	9
			C	Ø	E

Figure 3.A. UHF Channel Select Format

	<u>Switch</u>	<u>Format</u>
From:	[CHAN SEL]	Figure 3
To:	[E]	Figure 3

This format is used to select an active channel in the following manner:

1. Legal entries for this mode are 0 thru 9.
2. [C] may be used to clear the entry for corrective purposes.
3. Selection of [E] will enter the change indicated in the legend line to the radio and switch the display back to Figure 3 where the selected channel number and it's associated frequency is displayed in the first line. [CHAN SEL] switch is not active in this format.
4. Operation in this mode is the same as that shown on page 6, Figure 2.A for VHF channel select.

UHF		CH-6		118.315	
OFF		T/R			
F- - - - -			1	2	3
	MAN FREQ		4	5	6
			7	8	9
			C	Ø	E

Figure 3.B. UHF Frequency Select Format

	<u>Switch</u>	<u>Format</u>
From:	[MAN FREQ]	Figure 3
To:	[E]	Figure 3

This format is used to select frequencies in the following manner.

1. A frequency is entered in the legend line by selecting from the keyboard.
Legal entries are 116.000 to 149.975 .
2. A single activation of [C] will clear the last entered number and a double activation will clear all of the entered numerics for corrective purposes.
3. Activation of [E] will place the entered frequency in use and return the display to the UHF Submode Format, Figure 3. [MAN FREQ] Switch is not active in this format.
4. UHF frequency selection is the same as VHF frequency selection shown on page 8.

UHF		CH-6		118.315	
OFF		T/R			
CH- _		F- _ _ _ _ _		1	2
		PRST CHAN	4	5	6
			7	8	9
			C	Ø	E

Figure 3. C. Preset Channel Format

	<u>Switch</u>	<u>Format</u>
From:	[PRST CHAN]	Figure 3
To:	[E]	Figure 3

This format is used to pre-assign channels to frequencies in the following manner:

1. A channel number 0 - 9 is entered in the legend line by selecting from the keyboard. The associated frequency, if any, will also appear in the legend line.
2. The frequency to be assigned to the channel is next entered by use of the keyboard. Legal entries are from 30.00 to 75.95.
3. For corrective purposes, a single action of [C] will clear the last entered digit and a double activation will clear all of the entered numerics.
4. Activation of [E] will place the channel assignment in memory and return to this page with the next channel digit displaced in the legend line. It will continue to step after each entry. Selecting [PRI] or

SUB
MODE

 will return the display to its respective condition.
5. The channel with its assigned frequency will be displayed on the Preset Channel Format, Figure 2. D, when that format is displayed. [PRST CHAN] is not active in this format.
6. VHF preset channel operations are the same as those used on VHF procedure shown on page 8.

UHF CHAN STATUS			1/2
0		3	113.615
1	112.715	4	
2		5	

Figure 3.D. Stat Page

	<u>Switch</u>	<u>Format</u>
From:	[STAT PAGE]	Figure 3
To:	CH-0 thru CH-9	Figure 3.D

This format is primarily a review or status display of the preset channels. The active channel is indicated by the asterisk symbol. A channel can be selected for re-assignment in the following manner:

1. Select channel for re-assignment by pressing on channel legend.
2. Manual frequency select format, Figure 3.B will display with the selected channel and its assigned frequency, if any, displayed in the legend line.
3. A new assignment may be made by entering a new frequency from the keyboard. The channel entry area in the legend line will not be changeable when accessed from the [STAT PAGE] Format.
4. This status page will be displayed on two pages. Use step and last to sequence.

6	118.315	9
7		
8		

Figure 3.E. Stat Page

ADF		CH-2		25.64
ON		AUTO		VCE
CHAN SEL	MAN FREQ	PRST CHAN	STAT PAGE	
RCVR	AUTO ADF	MAN ADF		
VCE	CW	TEST		

Figure 4. ADF Submode Format

	<u>Switch</u>	<u>Format</u>
From:	ADF	Figure 1
To:	CHAN	Figure 4. A
	MAN FREQ	Figure 4. B
	PRST CHAN	Figure 4. C
	STAT PAGE	Figure 4. D
	RCVR	Figure 4
	AUTO ADF	Figure 4
	MAN ADF	Figure 4
	VCE	Figure 4
	CW	Figure 4
	TEST	Figure 4

Activation of any one of three mutually exclusive switches [RCVR], [AUTO ADF] and [MAN ADF] will display in the second line in the mode display.

Activation of any one of three mutually exclusive switches [VCE], [CW] and [TEST] will display in the second line of the display.

[VCE] allows the ADF to be used as an AM receiver.

[CW] allows the ADF to be used as a CW receiver.

[TEST] will slew the ADF indicator 180°. Test is an automatic 10 second duration.

ADF			CH-2	25.64	
ON			AUTO	VCE	
CH- _			1	2	3
CHAN SEL			4	5	6
			7	8	9
			C	Ø	E

Figure 4. A. Channel Select Format

	<u>Switch</u>	<u>Format</u>
From:	[CHAN SEL]	Figure 4
To:	[E]	Figure 4

This format is used to select an active channel in the following manner:

1. Selection of a numeric from the keyboard will display that numeric on the legend line, or numeric [1] and [9]. The associated frequency stored in that channel will then appear in the legend line.
2. [C] may be used to clear the entry for corrective purposes.
3. Selection of [E] will enter the change indicated in the legend line to the radio and switch the display back to Figure 4 where the selected channel number and its associated frequency is displayed in the first line. Legal entries from this format are 0 thru 9. [CHAN SEL] switch is not active in this format.
4. Procedure to be similar to those shown on pages 6, 11, 16.

ADF ON		AUTO		CH-2	2564 VCE	
F- _ _ _ _				1	2	3
	MAN FREQ		4	5	6	
			7	8	9	
			C	Ø	E	

Figure 4. B. Frequency Select Format

	<u>Switch</u>	<u>Format</u>
From:	MAN FREQ	Figure 4
To:	E	Figure 4

This format is used to select frequencies in the following manner:

1. A frequency is entered in the legend line by selecting from the keyboard. Legal entries are 100 KHZ to 3000 KHZ.
2. A single activation of [C] will clear the last entered number and a double activation will clear all of the entered numerics for corrective purposes.
3. Activation of [E] will place the entered frequency in use and return the display to the ADF Submode Format, Figure 4. [MAN FREQ] switch is not active in this format.

ADF		CH-2		2564	
ON		AUTO		VCE	
CH- __		F- ----		1	2
		PRST CHAN	4	5	6
			7	8	9
			C	Ø	E

Figure 4. C. Preset Channel

	<u>Switch</u>	<u>Format</u>
From:	[PRST CHAN]	Figure 4
To:	E	Figure 4

This format is used to pre-assign channels to frequencies in the following manner:

1. A channel number 0 - 9 is entered in the legend line by selecting from the keyboard. The associated frequency, if any, will also appear in the legend line.
2. The frequency to be assigned to the channel is next entered by use of the keyboard. Legal entries are from 30.00 to 75.95.
3. For corrective purposes, a single action of [C] will clear the last entered digit and a double activation will clear all of the entered numerics.
4. Activation of [E] will place the channel assignment in memory and return to this page with the next channel digit displaced in the legend line. It will continue to step after each entry. Selecting [PRI] or [SUB] will return the display to its respective condition.
5. The channel with its assigned frequency will be displayed on the Preset Channel Format, Figure 2. D, when that format is displayed. [PRST CHAN] is not active in this format.
6. Procedures are similar to those shown on pages 8, 12 & 23.

ADF CHAN STATUS			
0	120	5	
1	150	6	479
2	2250	7	
3		8	1572
4	760	9	

Figure 4.D. Stat Page

	<u>Switch</u>	<u>Format</u>
From:	STAT PAGE	Figure 4
To:	0 thru 9	Figure 4. B

This format is primarily a review or status display of the preset channels. The active channel is indicated by the asterisk symbol. A channel can be selected for re-assignment in the following manner:

1. Select channel for re-assignment by pressing on channel legend.
2. Frequency selection format, Figure 4. B will display with the selected channel and its assigned frequency, if any, displayed in the legend line.
3. A new assignment may be made by entering a new frequency from the keyboard. The channel entry area in the legend line will not be changeable when accessed from the Preset Channels Format.

CNV	CH-4	118.35	
ON		MB HI	
CHAN SEL	MAN FREQ	PRST CHAN	STAT PAGE
MB VOL	↑↑		MB HI
NAV VOL	↓↓	TEST	MB LO

Figure 5. CNV Submode Format

	<u>Switch</u>	<u>Format</u>
From:	CNV	Figure 1
To:	CHAN SEL	Figure 5.A
	MAN FREQ	Figure 5.B
	PRST CHAN	Figure 5.C
	STAT PAGE	Figure 5.D
	MB VOL	Figure 5
	MB HI	Figure 5
	MB LO	Figure 5
	NAV VOL	Figure 5
	TEST	Figure 5

[MB HI] - Activation of [MB HI] will place the receiver in the marker beacon high sensitivity mode and display "MB HI" in the second line of the display.

[MB LO] - Activation of [MB LO] will place the receiver in the marker beacon low sensitivity mode and display "MB LO" in the second line of the display.

[TEST] - External test equipment is needed and the test indications are displayed on external equipment. This is a momentary switch.

[MB VOL] - Selection of [MB VOL] will allow control of the volume by use of the up or down arrow symbol keys.

[NAV VOL] - Selection of [NAV VOL] will allow control of the volume by use of the up or down arrow symbol keys.

CNV			CH-4			118.35		
ON								
CH-__				1	2	3		
CHAN SEL				4	5	6		
				7	8	9		
				C	Ø	E		

Figure 5. A. Channel Select Format

	<u>Switch</u>	<u>Format</u>
From:	CHAN SEL	Figure 5
To:	E	Figure 5

This format is used to select an active channel in the following manner:

1. Selection of a numeric from the keyboard will display that numeric in the legend line following CH-. The associated frequency if stored in that channel will appear in the legend line.
2. [C] may be used to clear the entry for corrective purposes.
3. Selection of [E] will enter the change indicated in the legend line to the radio and switch the display back to Figure 5 where the selected channel number and it's associated frequency is displayed in the first line. Legal entries from this format are 0 through 9. [CHAN SEL] switch is not active in this format.
4. Procedures are similar to those shown on pages 6, 11, 16.

CNV		CH-4		118.35	
ON				MB-HI	
F- _____			1	2	3
	MAN FREQ		4	5	6
			7	8	9
			C	/	E

Figure 5. B. Frequency Select Format

	<u>Switch</u>	<u>Format</u>
From:	MAN FREQ	Figure 5
To:	E	Figure 5

This format is used to select frequencies in the following manner:

1. A frequency is entered in the legend line by selecting from the keyboard. Legal entries are 108.00 MHz to 117.95 MHz.
2. A single activation of [C] will clear the last entered number and a double activation will clear all of the entered numerics for corrective purposes.
3. Activation of [E] will place the entered frequency in use and return the display to the CNV Submode Format, Figure 5. The [MAN FREQ] switch is not active in this form.

CNV		C-4		118.35	
ON				MB HI	
CH-__		F-____		1	2
		PRST CHAN	4	5	6
			7	8	9
			C	Ø	E

Figure 5.C. Preset Channel Format

	<u>Switch</u>	<u>Format</u>
From:	PRST CHAN	Figure 5
To:	NUMERIC + E	Figure 5

This format is used to pre-assign channels to frequencies in the following manner.

1. A channel number 0 - 9 is entered in the legend line by selecting from the keyboard. The associated frequency, if any, will also appear in the legend line.
2. The frequency to be assigned to the channel is next entered by use of the keyboard. Legal entries are from 30.00 to 75.95.
3. For corrective purposes, a single action of [C] will clear the last entered digit and a double activation will clear all of the entered numerics.
4. Activation of [E] will place the channel assignment in memory and return to this page with the next channel digit displaced in the legend line. It will continue to step after each entry. Selecting [PRI] or [SUB] will return the display to its respective condition. [MODE]
5. The channel with its assigned frequency will be displayed on the Preset Channel Format, Figure 2.D, when that format is displayed. [PRST CHAN] is not active in this format.

CNV CHAN STATUS	
0	5
1	6
2	7
3	8
4	9

Figure 5.D. Stat Page

	<u>Switch</u>	<u>Format</u>
From:	[STAT PAGE]	Figure 5
To:	0 thru 9	Figure 5.b

This format is primarily a review or status display of the preset channels. The active channel is indicated by the asterisk symbol. A channel can be selected for re-assignment in the following manner.

1. Select channel for re-assignment by pressing on channel legend.
2. Frequency selection format, Figure 5.B will display with the selected channel and it's assigned frequency, if any, displayed in the legend line.
3. A new assignment may be made by entering a new frequency from the keyboard. The channel entry area in the legend line will not be changeable when accessed from the Preset Channels Format.

THIS PAGE LEFT BLANK ON PURPOSE

THIS PAGE LEFT BLANK ON PURPOSE

DNV		NAV		CKPT 1	
ON					
PP 15 SUP 1417 8409					
FLY TO		CKPT		BKUP	
NAV STAT		NEXT CKPT		UTM	
L/L		CKPT STAT		TGT	
UP DATE		TEST		TGT STAT	

Figure 6. DNV Submode Format

	<u>Switch</u>	<u>Format</u>
From:	DNV	Figure 1.
To:	FLY TO + CKPT	Figure 6A
	or TGT	
	CKPT	Figure 6B
	NAV STAT	Figure 6D
	NEXT CKPT	(See below)
	BKUP NAV	Figure 6F
	CKPT STAT	Figure 6G
	TGT	Figure 6C
	UP DATE	Figure 6H
	TEST	(See below)
	TGT STAT	Figure 6I

Activation of [NEXT CKPT] will increment the FLY TO checkpoint number by one and all data will now be referenced to the new checkpoint number.

Activation of [TEST] will display a momentary message in the legend line of the display. "OK" for no failure and "FAIL" for a failure.

Activation of [L/L] or [UTM] will present information in lat long or UTM coordinates.

DNV		NAV		CKPT 1		
TO ____				1	2	3
FLY TO	CKPT		4	5	6	
			7	8	9	
TGT			C	Ø	E	

Figure 6.A. Fly to CKPT or TGT Format

	Switch	Format
From:	[FLY TO] + [CKPT] or [TGT]	Figure 6
To:	[FLY TO]	No Action
	[CKPT] + Numeric	Figure 6
	or [TGT] + Numeric + E	Figure 6

This format is used to select either a checkpoint or target as the point to compute NAV data related to present position.

1. Selection of [CKPT] will display CKPT after "TO" in the legend line.
2. After Selection of [CKPT] one numeric must be entered from the keyboard and will display after "CKPT".
3. Selection of [E] will enter this data to the DNV System and return the display to Figure 6. The entered numeric will now display in the upper right hand area of the display after "CKPT".
4. Selection of [TGT] + numeric will change the display back to Figure 6 and display "TGT" + numeric in the upper right hand area of the display.

DNV		NAV		CKPT 1		
CKPT ____		COORD ____		1	2	3
	CKPT			4	5	6
	UTM	L/L		7	8	9
				C	Ø	E

Figure 6. B. CKPT Entry Format

	<u>Switch</u>	<u>Format</u>
From:	[CKPT]	Figure 6
To:	[UTM]	Figure 6. B. 1
	[L/L]	Figure 6. B. 2

This format is used to enter the Checkpoint Number and select the Coordinate system for the checkpoint entry. Selection of UTM as L/L will place the CDU in that mode for entry of coordinates. The converse coordinate would be computed by the Doppler NAV unit and be available to the CDU for display.

1. The Checkpoint number is entered via the numeric keyboard and displayed in the legend hhe.
2. [UTM] or [L/L] is selected for the coordinate entry system.
3. Selection of [E] will display Figure 6. B. 1 with the checkpoint + number displayed in the upper right hand corner.

DNV			NAV			CKPT 1		
SEL SPHEROID								
CLO			CKPT			EVO		
CL6			UTM			BEO		
INO						AUO		
						C		E

Figure 6. B. 1. UTM Coordinate Entry Format

	<u>Switch</u>	<u>Format</u>
From:	[E]	Figure 6. B
To:	[CLO]	Figure 6. B. 1. 1
	[EVO]	Figure 6. B. 1. 1
	[CL6]	Figure 6. B. 1. 1
	[BEO]	Figure 6. B. 1. 1
	[INO]	Figure 6. B. 1. 1
	[AUO]	Figure 6. B. 1. 1

This format is used to enter the UTM Spheroid. Selection of any one of the six spheroids will display Figure 6. B. 1. 1 and display the Selected Spheroid in the right hand side of the second line of display.

This Format can be bypassed by the use of [STEP] in the Fixed Switches.

DNV		NAV		CKPT 1		
ENT ZONE ____				1	2	3
	CKPT		4	5	6	
	UTM		7	8	9	
			C	Ø	E	

Figure 6. B. 1. 1. UTM Numeric Entry Format

	<u>Switch</u>	<u>Format</u>
From:	[CLO]	Figure 6. B. 1
	[EVO]	Figure 6. B. 1
	[CL6]	Figure 6. B. 1
	[BEO]	Figure 6. B. 1
	[INO]	Figure 6. B. 1
	[AUO]	Figure 6. B. 1
To:	[E]	Figure 6. B. 1. 2
	[E]	Figure 6. B. 1. 3

This format is used to enter the first two numerics (ZONE) in the UTM Coordinate System.

1. Enter two numerics (Legal entry 1-60). The numeric will display in the legend line.
2. Select [E] to enter the numerics. Alpha entry display Figure 6. B. 1. 2 will display.

DNV 15 SUP		NAV				CKPT 1	
A	B	C	D	E	F	G	H
I	J	K	L	M	N	O	P
Q	R	S	T	U	V	W	X
Y	Z				C		E

Figure 6. B. 1. 2. UTM Alpha entry Format

	<u>Switch</u>	<u>Format</u>
From:	[E]	Figure 6. B. 1. 1
To:	[E]	Figure 6. B. 1. 1

This format is used to enter the area alphas in the UTM Coordinate system. A total of three alphas will be entered from this format. Legal entries are C thru X with I and O omitted.

1. Enter three alphas which will display as shown.
2. Select [E] to enter the alphas. Figure 6. B. 1. 1 will display for entry of the final numerics. (Legal entry 8 digits.)
3. After completion of Numeric entry from Figure 6. B. 1. 1, actuation of E will display Figure 6. B. 1. 3.

DNV		NAV		CKPT 1	
ENT MAG VAR					
	CKPT		E	+	W
			C		E

Figure 6.B.1.3. Mag VAR Entry Format

	<u>Switch</u>	<u>Format</u>
From:	[E]	Figure 6.B.1.1
To:	[E]	Figure 6.B.1.1

This format is used to enter the E or W for magnetic variation of the checkpoint.

1. Enter [E] or [W] for the MAG VAR.
2. Select [E] to enter the data. Figure 6.B.1.4 will display for entry of numerics. --
3. Select [E] from 6.B.1.4 after numerics have been entered will enter the data and display Figure 6, DNV Submode format.
4. Activation [STEP] will place the same MAG VAR if any as inserted for the previous CKPT or TGT. Display will return to Figure 6.B.

DNV		NAV		CKPT 1		
E _ _ . _				1	2	3
	CKPT		4	5	6	
	UTM		7	8	9	
			C	Ø	E	

Figure 6.B.1.4. UTM Numeric Entry Format

DNV		NAV		CKPT 1	
ENT LAT				N	
	CKPT				
		L/L		S	
			C		E

Figure 6.B.2. L/L Coordinate Entry Format

	<u>Switch</u>	<u>Format</u>
From:	[L/L]	Figure 6. B
To:	[E]	Figure 6. B. 2. 1

This format is used to enter the N or S direction for Latitude.

1. Select [N] or [S] from the keyboard.
2. Select [E] to enter the data and display Figure 6. B. 2. 1 for further numeric entry.

DNV		NAV		CKPT 1		
N ____ ° ____ ' ____ "				1	2	3
	CKPT		4	5	6	
		L/L	7	8	9	
			C	Ø	E	

Figure 6.B.2.1. L/L Numeric Entry Format

	<u>Switch</u>	<u>Format</u>
From:	[E]	Figure 6.B.2
To:	[E]	Figure 6.B.2.2

This format is used to enter the numeric for the Latitude.

1. Select Numeric from the keyboard.
2. Select [E] to enter the numeric and display Figure 6.B.2.2 for entry of Longitude direction.

DNV		NAV		CKPT 1	
ENT LON ____° ____' ____"					
	CKPT		E		W
		L/L			
			C		E

Figure 6. B. 2. 2. Longitude Direction Entry Format

	<u>Switch</u>	<u>Format</u>
From:	[E]	Figure 6. B. 2. 1
To:	[E]	Figure 6. B. 2. 1

This format is used to enter the East or West direction of the longitude.

1. Select [E] or [W] for direction from the keyboard.
2. Select [E] to enter the data and display Figure 6. B. 2. 1 for entry of numeric data for longitude.
3. After completion of entry of numeric data, Selection of [E] from Figure 6. B. 2. 1 will display Figure 6. B. 2. 3.

DNV		NAV		CKPT 1	
ENT MAG VAR					
	CKPT		E		W
		L/L			
			C		E

Figure 6. B. 2. 3. CHECKPOINT or TARGET MAG VAR Entry Format

	<u>Switch</u>	<u>Format</u>
From:	[E]	Figure 6. B. 2. 1
To:	[E]	Figure 6. B. 1. 4

This format is used to enter the E or W for Magnetic Variation of the checkpoint.

1. Select [E] or [W] from the keyboard area.
2. Select [E] to enter the data and display Figure 6. B. 2. 1 for entry of three numerics.
3. After entry of three numerics from Figure 6. B. 2. 1, selection of [E] will enter the data and display Figure 6.
4. Activation of [STEP] will place the same MAG VAR as inserted for the previous CKPT or TGT. Display will Return to 6B.

DNV		NAV		TGT 1		
TGT—		COORD—		1	2	3
FRZE				4	5	6
	UTM	L/L		7	8	9
TGT		TGT STAT		C	Ø	E

Figure 6.C. Target Entry Format

	<u>Switch</u>	<u>Format</u>
From:	[TGT]	Figure 6
To:	See Text	

This format will be used to enter target coordinates in the same manner as the checkpoint coordinates were entered. The only difference between Figure 6.B and Figure 6.C is that [TGT] is used in Figure 6.C. The sequence of Target coordinate entry is the same as checkpoint coordinate entry starting at Figure 6.B.

To Freeze a target without presetting a target number, depressing [FRZE] then [E] over desired target point will enter target coordinates in the last entered coordinate system in non use target positions 6-9.

If specific target number is desired, enter [FRZE] + numeric 0-9.

For freeze entries, after depressing [E] the coordinates and the target number of frozen point are displayed on the 2nd line.

If it is desired to change the target number, depressing a numeric and [E] will revise auto selected target number.

DNV	STATUS	TGT 1
PP	N 41° 10.5	W 164° 14. 1
RGE 163.5	63	GSPD 270 TKE 104
BRG 273	TRK 122	XTK 20.5
TTG 200	WND 150/14	

Figure 6. D. Navigation Status Display

	<u>Switch</u>	<u>Format</u>
From:	[NAV STAT]	Figure 6
To:	[SUB]	Figure 6

This page will display all of the flight information in one page. Return to the DNV submode format will be by selection of the fixed function switch [SUB].

DNV			NAV			CKPT 1		
ENT PRES POS					1	2	3	
			4	5	6			
PRES POS	UTM	L/L	7	8	9			
			C	Ø	E			

Figure 6. E. Present Position Entry Format

	<u>Switch</u>	<u>Format</u>
From:	[PRES POS]	Figure 6
To:	See text.	

This format is used to enter the present position in UTM or LAT/LONG coordinates. Selection of UTM or L/L will display the function Figure 6. B. 1 or Figure 6. B. 2 for coordinate entry in the same manner as the checkpoint coordinate entry was performed.

The only difference in the function is that [PRES POS] will display rather than [CKPT].

DNV		BKUP		CKPT 1	
WIND 120/15		TRK 122		GSPD 125	
WIND DIR		WIND SPD		GSPD	
TRK					

Figure 6. F. Backup NAV Format

	<u>Switch</u>	<u>Format</u>
From:	[BKUP NAV]	Figure 6
To:	[WIND DIR]	Figure 6. F. 1
	[WIND SPD]	Figure 6. F. 2
	[GSPD]	Figure 6. F. 3
	[TRK]	Figure 6. F. 4

This format is used to provide estimated information for the Backup mode in D NAV when the Radar is inoperative but the computer is still functioning. Selection of any of the switches will display that format for entry of data.

[STEP] may be used to bypass menus in the back-up mode.

DNV		BKUP		CKPT 1	
WIND 120/15		TRK 122		GSPD 125	
ENT WIND DIR _ _ _ _			1	2	3
WIND DIR			4	5	6
			7	8	9
			C	Ø	E

Figure 6. F. 1. WIND Direction Entry Format

	<u>Switch</u>	<u>Format</u>
From:	[WIND DIR]	Figure 6. F
To:	[E]	Figure 6. F

This format is used to enter estimated Wind direction.

1. Enter Three numerics for Wind direction from the keyboard. The number will display on the legend line as entered.
2. Select [E] to enter the data to the computer. Figure 6. F will display with the Wind direction displayed in the Second line.

DNV		BKUP		CKPT 1	
WIND 120/15		TRK 122		GSPD 125	
ENT WND SPD _ _			1	2	3
	WIND SPD		4	5	6
			7	8	9
			C	Ø	E

Figure 6. F.2. Wind Speed Entry Format

	<u>Switch</u>	<u>Format</u>
From:	[WIND SPD]	Figure 6. F
To:	[E]	Figure 6. F

This format is used to enter an estimated wind speed in the backup mode of operation.

1. Enter a Two digit number for wind speed from the keyboard. The number will appear in the legend line.
2. Selection of [E] will enter the data to the computer and display Figure 6. F with the wind speed displayed in the Second line.

DNV		BKUP		CKPT 1	
WIND 120/15		TRK 122		GSPD 125	
ENT GND SPD _ _ _			1	2	3
		GSPD	4	5	6
			7	8	9
			C	Ø	E

Figure G.F.3. Ground Speed Entry Format

	<u>Switch</u>	<u>Format</u>
From:	[GSPD]	Figure 6. F
To:	[E]	Figure 6. F

This format is used to enter an estimated ground speed in the event of a Radar failure.

1. Enter a three digit number for the estimated ground speed from the numeric keyboard. The number will display in the legend line.
2. Selection of [E] will enter this data to the computer and display Figure 6. F with the ground speed displayed in the second line.

DNV			BKUP			CKPT 1		
WND 120/15			TRK 122			GSPD 125		
ENT TRK ANG _ _ _					1	2	3	
				4	5	6		
TRK				7	8	9		
				C	Ø	E		

Figure 6. F. 4. Track Entry Format

	<u>Switch</u>	<u>Format</u>
From:	[TRK]	Figure 6. F
To:	[E]	Figure 6.F

This format is used to enter an estimated Track in the event of a radar failure.

1. Enter a three digit Track Angle using the numeric Keyboard. The number will display in the legend line.
2. Selection of [E] will enter this data to the computer and display Figure 6. F with the TRK Angle displayed in the Second line.

DNV	CKPT STATUS	CKPT 1 1/2
0	W 164° 12: 1 N 45° 12: 1	3
1		4
2		5

Figure 6. G. Checkpoint Status Display

	<u>Switch</u>	<u>Format</u>
From:	[CKPT STAT]	Figure 6
To:	[CKPI 0] thru [CKPT 9]	Figure 6. B

This format is used to display the status of the checkpoints showing each Checkpoint with its coordinates. A checkpoint can be selected for coordinate change from this format by pressing on the CKPT Legend. Figure 6. B will display with the selected Checkpoint numerical displayed. New coordinates can be entered in the same manner as were used for original checkpoint entries.

This information will appear on two pages 0-5 on page 1/2 & 6-9 on page 2/2.

DNV		UP DATE		CKPT 1	
PP 15SUP 1417 8409				XXXX	
UPDATE DIST _ _ . _			1	2	3
FRZE	CKPT		4	5	6
			7	8	9
	UP DATE		C	Ø	E

Figure 6.H. Update Formats

	<u>Switch</u>	<u>Format</u>
From:	[UP DATE]	Figure 6
To:	[FRZE]	Figure 6.H
	[UP DATE]	Figure 6

This format will be used to update present position from a stored checkpoint. The update can be accomplished in one of the two following ways:

1. Update to Existing Checkpoint

An existing checkpoint that has its coordinates already entered into the system can be used to update present position in the following manner.

- A. Select [UP DATE] prior to arriving at the checkpoint. Update format Figure 6.H will display.

- B. Select [CKPT] and numeric to indicate the update will be made in reference to the checkpoint displayed. If checkpoint displayed is the desired reference point proceed to step C.
- C. Select [FRZE] when the selected checkpoint is flown over. This action will store the coordinates of the checkpoint as computed by the Navigation system and compare them to the previously entered checkpoint coordinates. The difference in Range will be displayed in the legend line with "UPDATE DIST" displayed before the number.
- D. If the operator elects to update the system, [UP DATE] will be selected to accomplish the update and DNV Submode format Figure 6 will display.

2. Update to New Checkpoint

A new checkpoint or terrain feature can be used for update purposes if the coordinates are known.

- A. Select [UP DATE] from Figure 6 prior to reaching the new Terrain feature.
- B. Select [NEW] from Figure 6.H to inform the system that the update will be compared against a new set of coordinates to be entered by the operator.
- C. The coordinates for the New point can be entered before or after overflying the point. When [NEW] is selected, prompting will display in the legend line for the coordinate system that is presently in the system. The operator will select [DATA ENT] to enter L/L or UTM in the previously described manner.
- D. As the New point is overflown selection of [FRZE] computed coordinates of the point for comparison with the entered coordinates. The difference will be displayed in the second line of the display.
- E. The operator can select to update the system by activating [UP DATE]. The display will return to Figure 6.

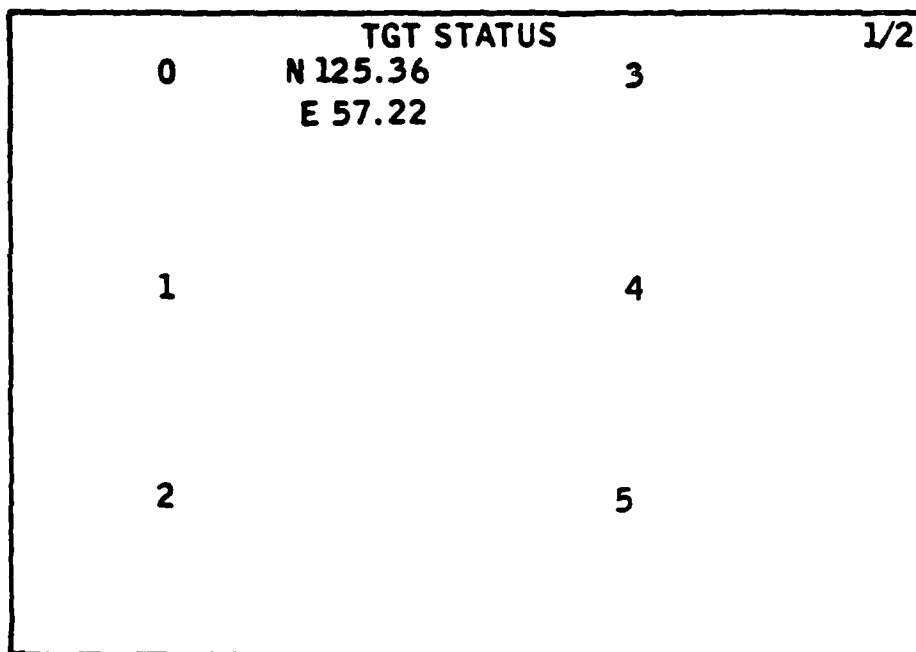


Figure 6.1 Target Status Display

	<u>Switch</u>	<u>Format</u>
From:	[TGT STAT]	Figure 6
To:	[PRI]	Fixed

This display is used to display the status of the Targets. The system will have the capability to store up to 10 targets with the coordinates.

This display will be on two pages. Targets 0-4 on page 1/2 & 5-9 on page 2/2.

IFF		NORM		
ON	M1	M2	M3A	M4A
M-1	M-2	M-3A	STAT	
M-4	NORM	STBY	ANT	
M-C	RAD TEST	TEST		

Figure 7. . IFF Submode

	<u>Switch</u>	<u>Format</u>
From:	IFF	Figure 1
To:	M-1	Figure 7. A
	M-2	Inserted as fixed data in ROM
	M-3A	Figure 7. C
	M-4	Figure 7. D
	ANT	Figure 7. E
	M-C	Figure 7
	NORM	Figure 7
	- STBY	Figure 7
	- RAD TEST	Figure 7
	TEST	Figure 7

"NORM" - Activation of "NORM" will place the IFF in the normal mode of operation and display "NORM" on the top line of the display.

"STBY" - Activation of this switch will place the IFF in the standby mode and display "STBY" in the top line of the display.

"RAD TEST" - Activation of this momentary switch will enable the IFF to test its receive functions with the help of an outside source.

"TEST" - Activation of ["TEST"] and then any of the mode switches M-1 through M-4 will request a test of that IFF function. The test results will be displayed in the second line of the display.

To turn any modes (M-1 to M-4) on, use the following procedure: EX: M-1

Select the mode by pressing [M-1] ; then [ON] MI will appear in the 2nd line.

To turn any mode off, use the following procedure: EX: M-1

Press [M-1] then [OFF] MI will be deleted from the 2nd line.

IFF		NORM				
ON		M1				
ENT CODE ____				1	2	3
M-1				4	5	6
				7		
				C	Ø	E

Figure 7.A. M-1 Code Format

	<u>Switch</u>	<u>Format</u>
From:	[M-1]	Figure 7
To:	[E]	Figure 7

This format is used to enter the M-1 code for the IFF and operator in the following manner:

1. [M-1] CODE will be displayed in the legend line. The code will be entered by selection from the numeric keyboard. Legal entries are from 00 to 73.
2. ["C"] can be used to clear the entry as in previous formats.
3. ["E"] will be used to enter the code and will switch the display back to Figure 7, where M-1 and its code will be displayed in the top line of the display.

IFF			NORM		
ON					
ENT CODE _ _ _ _ _			1	2	3
	M-2		4	5	6
			7		
			C	Ø	E

Figure 7.B. M-2 Code Format

	<u>Switch</u>	<u>Format</u>
From:	(M-2) (GARD)	Figure 7
To:	E	Figure 7

This format will be used to enter the M-2 code which is a guarded code:

1. [M-2] CODE will be displayed in the legend line. The code will be entered by selection from the numeric keyboard. Legal entries are from 0 to 7777.
2. ["C"] can be used to clear the entry for corrective purposes as described previously.
3. ["E"] will enter the M-2 code to the system and return the display to Figure 7.

IFF			NORM		
ON			M3A		
ENT CODE _____			1	2	3
		M-3A	4	5	6
			7		
			C	Ø	E

Figure 7-C. M-3A Code Format

	<u>Switch</u>	<u>Format</u>
From:	M-3A	Figure 7
To:	E	Figure 7

This format is used to enter the M-3A code:

1. [M-3A] CODE will display in the legend line. Variation for the code will be entered by use of the numeric keyboard. Legal entries from 0 to 7777.
2. ["C"] may be used to clear the entry for corrective purposes as described previously.
3. ["E"] will be used to enter the new code to the IFF system and will return the display to Figure 7.

IFF		NORM		HOLD	
ON		M4			
M-4 SELECT					
M-4	A	B	HOLD		
	AUD	AUD LITE	OUT		

Figure 7.D. M-4 Mode Formats

	<u>Switch</u>	<u>Formats</u>
From:	M-4	Figure 7
To:	A	Figure 7
	B	Figure 7
	HOLD	Figure 7
	AUD	Figure 7
	AUD LITE	Figure 7
	OUT	Figure 7

This format is used to select the M-4 mode of operation in the following manner:

1. ["A"] - Selection of this switch will place the IFF in the M-4A mode and display "M-4A" in the top display line.
2. ["B"] - Selection of this switch will place the IFF in the M-4B mode and display "M-4B" in the top display line.
3. ["HOLD"] - Selection of this switch will place the IFF in the hold mode and display "HOLD" in the top display line. Mode 4 code will be retained if power is not turned off for 15 seconds.

4. [AUD] - Selection of this switch will place the IFF in the AUD mode and display "AUD" in the second line display line.
5. [AUD LITE] - Selection of this switch will place the IFF in the audio and light mode and display "AUD LITE" in the second display line.
6. [OUT] - Selection of this switch will turn the AUD or AUD LITE mode off.

IFF		NORM		TOP	} 4A } 4B
ON		M4			
ANT SEL					
TOP					
DIV					
BOT					

Figure 7. E. ANT Mode Format

	<u>Switch</u>	<u>Format</u>
From:	ANT	Figure 7
To:	TOP	Figure 7
	DIV	Figure 7
	BOT	Figure 7

This format is used to select the antenna mode:

1. ["TOP"] - Selection of this switch will place the antenna in the top pattern and display ["TOP"] on the second line of the display in Figure 6.
2. ["DIV"] - Selection of this switch will place the antenna in the divided antenna pattern and display DIV on the second line of the display in Figure 6.
3. ["BOT"] - Selection of this switch will place the antenna in the bottom antenna pattern and display ["BOT"] in the second line of the display in Figure 7.

IFF STATUS			
M1 - 72	*	M4 - A	*
M2 - 3533	*	M-C - ON	*
M3A - 7231	*	ANT - TOP	

Figure 7. F Stat Page

	<u>Switch</u>	<u>Format</u>
From:	[STAT PAGE]	Figure 2
To:	See text.	Figure 2. B

This format is primarily a review of IFF status. Active modes are indicated by the asterisk symbol. Return to any mode may be selected for re-assignment in the following manner:

1. Select mode for re-assignment by pressing on mode legend.

ATE
LMED
-8