CONTROL DISPLAY UNIT
PROGRAM FINAL REPORT

CLIN 004, CDRL C002

Copy available to DTIC does not permit fully legible reproduction

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.

1978

AUG 1 9 1982

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)
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.
SECTION 1 DESIGN DETAILS

Schematics, Cards
Schematics, Display
Panel Membrane Switch
Bill of Materials
DISPLAY

MODULE SCHEMATIC
NOTES:
ALL RESISTORS 1 MEORM

INTERACTIVE DISPLAY
COLUMN DRIVERS

Page 2022
CDU FRONT

PANEL MEMBRANE

SWITCH
## For Flexible Flat Conductor Cable Terminations

For flexible flat conductors on 0.015 [0.38] wire, conductors on 0.015 to 0.030 [0.41] wire, with 2.05 to 0.35 non-accumulative tolerance, 015 [0.30] max. total cable thickness.

### Pin
- **Length:** 0.422 [10.73]

### Receptacle
- **Length:** 0.328 [8.34]

### Edge Connector Contact
- **A:**
- **B:**
- **C:**

### Solder Tab—A
- **A:**
- **B:**
- **C:**

<table>
<thead>
<tr>
<th>Type of Contact</th>
<th>T</th>
<th>E</th>
<th>G</th>
<th>Pin</th>
<th>Description</th>
<th>Evaluation</th>
</tr>
</thead>
<tbody>
<tr>
<td>PIN</td>
<td>0.014</td>
<td>0.013</td>
<td>0.067</td>
<td>E</td>
<td>65-548</td>
<td>6.3/13</td>
</tr>
<tr>
<td>PIN/RECEPTACLE</td>
<td>0.014</td>
<td>0.013</td>
<td>0.067</td>
<td>E</td>
<td>65-548</td>
<td>6.3/13</td>
</tr>
<tr>
<td>EDGE CONNECTOR CONTACT</td>
<td>0.014</td>
<td>0.013</td>
<td>0.067</td>
<td>E</td>
<td>65-424</td>
<td>6.3/13</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Type of Contact</th>
<th>T</th>
<th>E</th>
<th>G</th>
<th>Pin</th>
<th>Description</th>
<th>Evaluation</th>
</tr>
</thead>
<tbody>
<tr>
<td>SOLDER TAB—A (Plastic in housing)</td>
<td>0.014</td>
<td>0.013</td>
<td>0.067</td>
<td>E</td>
<td>65-001</td>
<td>6.3/13</td>
</tr>
<tr>
<td>SOLDER TAB—A (Plastic in housing)</td>
<td>0.014</td>
<td>0.013</td>
<td>0.067</td>
<td>E</td>
<td>65-001</td>
<td>6.3/13</td>
</tr>
<tr>
<td>SOLDER TAB—A (Plastic in housing)</td>
<td>0.014</td>
<td>0.013</td>
<td>0.067</td>
<td>E</td>
<td>65-001</td>
<td>6.3/13</td>
</tr>
</tbody>
</table>
Single Row
Fin Housing

**Dimensions:**
1. All dimensions are in millimeters.
2. Controlling dimension is shown in bold.

**Housing Material:** Black Glass Filled Nylon

<table>
<thead>
<tr>
<th>No. of Positions</th>
<th>Dimensions</th>
<th>Housing Fix</th>
<th>Cover Fix</th>
<th>Reel Fix</th>
<th>Shipping Box</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>A</td>
<td>B</td>
<td>C</td>
<td>D</td>
<td>E</td>
</tr>
<tr>
<td>9</td>
<td>8.30</td>
<td>1.05</td>
<td>1.85</td>
<td>3.11</td>
<td>1.51</td>
</tr>
<tr>
<td>12</td>
<td>2.76</td>
<td>1.22</td>
<td>1.80</td>
<td>2.07</td>
<td>2.29</td>
</tr>
<tr>
<td>18</td>
<td>4.57</td>
<td>1.25</td>
<td>1.80</td>
<td>2.07</td>
<td>-</td>
</tr>
<tr>
<td>22</td>
<td>7.80</td>
<td>1.25</td>
<td>1.80</td>
<td>2.07</td>
<td>-</td>
</tr>
<tr>
<td>33</td>
<td>8.20</td>
<td>1.25</td>
<td>1.80</td>
<td>2.07</td>
<td>-</td>
</tr>
</tbody>
</table>

*Housings to mate with bulk mounted AHV/MGDU Connectors
**Extraction Tool Part No.: 91292-1 for Round Wire Contact
NOTES:
1. MTL: 1/16 24 ST ALUM. WITH.
   BLK ANODIZE FINISH.
2. REMOVE SHARP EDGES & BURRS.

BEZEL FOR CDU.

25 APR 78. ZAVATTO
CLEAR PLASTIC MEMBRANE SWITCH SUPPORT.

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

11 BROOKSIDE DRIVE
WILMINGTON, DEL.
302-652-2686.

OR
DICK WEST (213) - 636-0377.
BILL OF

MATERIALS
<table>
<thead>
<tr>
<th>LINE NO.</th>
<th>(D) LITTON DWG</th>
<th>(E) LITTON SPEC</th>
<th>PART NUMBER</th>
<th>PART NAME OR DESCRIPTION</th>
<th>MANUFACTURER</th>
<th>QTY PER ASSY.</th>
<th>CHG LTR</th>
<th>FOR PLANNING USE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>54L3'163</td>
<td></td>
<td>54L3'163</td>
<td>SYNC. COUNTER (21-24)</td>
<td></td>
<td>4</td>
<td></td>
<td>16-PIN</td>
</tr>
<tr>
<td>2</td>
<td>54L3'151</td>
<td></td>
<td>1</td>
<td>OF 8 SELECTOR (25)</td>
<td></td>
<td>1</td>
<td></td>
<td>16-PIN</td>
</tr>
<tr>
<td>3</td>
<td>1.76M14-2</td>
<td></td>
<td>1K X4 RAM</td>
<td>(26-211)</td>
<td></td>
<td>6</td>
<td></td>
<td>14-PIN</td>
</tr>
<tr>
<td>4</td>
<td>54L5'368A</td>
<td></td>
<td>54L5'368A</td>
<td>HEX BUS DRIVER (212,213)</td>
<td></td>
<td>2</td>
<td></td>
<td>16-PIN</td>
</tr>
<tr>
<td>5</td>
<td>D578'12</td>
<td></td>
<td>D578'12</td>
<td>T2 TO 300 INVERTER (214,215)</td>
<td></td>
<td>4</td>
<td></td>
<td>14-PIN</td>
</tr>
<tr>
<td>6</td>
<td>L-5132</td>
<td></td>
<td></td>
<td>TRIPLE 3-INP NAND</td>
<td>(219)</td>
<td>1</td>
<td></td>
<td>14-PIN</td>
</tr>
<tr>
<td>7</td>
<td>54L5'00</td>
<td></td>
<td>54L5'00</td>
<td>QUAD 2-INPUT NAND (219,221)</td>
<td></td>
<td>3</td>
<td></td>
<td>14-PIN</td>
</tr>
<tr>
<td>8</td>
<td>54L3'08</td>
<td></td>
<td>54L3'08</td>
<td>Q110 2-INPUT AND (220,223)</td>
<td></td>
<td>2</td>
<td></td>
<td>14-PIN</td>
</tr>
<tr>
<td>9</td>
<td>54L5'02</td>
<td></td>
<td>54L5'02</td>
<td>QUAD 2-INPUT NOR (224,226,227)</td>
<td></td>
<td>1</td>
<td></td>
<td>14-PIN</td>
</tr>
<tr>
<td>10</td>
<td>54L5'04</td>
<td></td>
<td>54L5'04</td>
<td>HEX INVERTER (225)</td>
<td></td>
<td>1</td>
<td></td>
<td>14-PIN</td>
</tr>
<tr>
<td>11</td>
<td>54L5'07</td>
<td></td>
<td>54L5'07</td>
<td>MASTER SLIDE / (228)</td>
<td></td>
<td>1</td>
<td></td>
<td>14-PIN</td>
</tr>
</tbody>
</table>
## ADVANCE BILL OF MATERIAL

<table>
<thead>
<tr>
<th>LINE NO.</th>
<th>PART NUMBER</th>
<th>PART NAME OR DESCRIPTION</th>
<th>MANUFACTURER</th>
<th>QTY PER ASSY.</th>
<th>CHG LTR</th>
<th>FOR PLANNING USE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>8085</td>
<td>CPU (Z1)</td>
<td>INTEL</td>
<td>1</td>
<td>40-P11</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>8257</td>
<td>DMA CONTROLLER (Z2)</td>
<td>INTEL</td>
<td>1</td>
<td>40-P11</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>8212</td>
<td>8-BIT I/O PORT (Z3, Z7, Z9, Z10)</td>
<td></td>
<td>4</td>
<td>3V-P11</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>74S258</td>
<td>SELECTOR (Z4)</td>
<td></td>
<td>1</td>
<td>16-P11</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>8259</td>
<td>INTERRUPT CONTROLLER (Z5)</td>
<td></td>
<td>1</td>
<td>2S-P11</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>8252</td>
<td>TIMER (Z6)</td>
<td></td>
<td>1</td>
<td>1W-P11</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>74A154</td>
<td>TRI-STATE BUFFER (Z8)</td>
<td></td>
<td>1</td>
<td>2C-P11</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>74H74</td>
<td>D-THE F/F. (Z11)</td>
<td></td>
<td>1</td>
<td>1W-P11</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>54LS02</td>
<td>2-INPUT AND (Z10, Z13)</td>
<td></td>
<td>2</td>
<td>1W-P11</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td>CRYSTAL</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LINE NO.</td>
<td>PART NUMBER</td>
<td>PART NAME OR DESCRIPTION</td>
<td>MANUFACTURER</td>
<td>QTY PER ASSY.</td>
<td>CHG LTH</td>
<td>FOR PLANNING USES</td>
</tr>
<tr>
<td>---------</td>
<td>-------------</td>
<td>--------------------------</td>
<td>--------------</td>
<td>---------------</td>
<td>---------</td>
<td>-------------------</td>
</tr>
<tr>
<td>1</td>
<td>2716</td>
<td>2K X 8 INCH ROLL (21-24)</td>
<td></td>
<td>4</td>
<td></td>
<td>2V-P10</td>
</tr>
<tr>
<td>2</td>
<td>21144</td>
<td>1K X 4 ROLL (35-312)</td>
<td></td>
<td>3</td>
<td></td>
<td>19-P10</td>
</tr>
<tr>
<td>3</td>
<td>545.138</td>
<td>DECODER - 3708 (118,214)</td>
<td></td>
<td>2</td>
<td></td>
<td>16-P10</td>
</tr>
<tr>
<td>CHIP</td>
<td>QTY</td>
<td>5V</td>
<td>240</td>
<td>6.8</td>
<td>2.0</td>
<td>6.8</td>
</tr>
<tr>
<td>--------</td>
<td>-----</td>
<td>----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5V</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

TOTAL CURRENT (mA)
<table>
<thead>
<tr>
<th>Chip</th>
<th>QTY</th>
<th>SW</th>
<th>&quot;Other Requirements&quot;</th>
<th>Totals (Wt.)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>7</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>8</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>9</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>10</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*Total Pounds (Wt.): 822.441*
<table>
<thead>
<tr>
<th>CHIP</th>
<th>QTY</th>
<th>5V</th>
<th>8V</th>
<th>5V</th>
<th>8V</th>
</tr>
</thead>
<tbody>
<tr>
<td>54LS163</td>
<td>4</td>
<td>1.9</td>
<td></td>
<td>7.6</td>
<td></td>
</tr>
<tr>
<td>54LS151</td>
<td>7</td>
<td>0.6</td>
<td></td>
<td>4.2</td>
<td></td>
</tr>
<tr>
<td>74HC4064</td>
<td>6</td>
<td>0.9</td>
<td></td>
<td>5.4</td>
<td></td>
</tr>
<tr>
<td>54LS244</td>
<td>2</td>
<td>1.2</td>
<td></td>
<td>2.4</td>
<td></td>
</tr>
<tr>
<td>74F74112</td>
<td>2</td>
<td>0.7</td>
<td></td>
<td>1.4</td>
<td></td>
</tr>
<tr>
<td>74F74L12</td>
<td>1</td>
<td>0.6</td>
<td></td>
<td>1.8</td>
<td></td>
</tr>
<tr>
<td>74F74L10</td>
<td>3</td>
<td>2.4</td>
<td></td>
<td>7.2</td>
<td></td>
</tr>
<tr>
<td>54LS7402</td>
<td>3</td>
<td>4.4</td>
<td></td>
<td>8.4</td>
<td></td>
</tr>
<tr>
<td>54LS7402</td>
<td>3</td>
<td>2.8</td>
<td></td>
<td>8.4</td>
<td></td>
</tr>
<tr>
<td>54LS7402</td>
<td>1</td>
<td>3.6</td>
<td></td>
<td>3.6</td>
<td></td>
</tr>
</tbody>
</table>

**TOTAL CURRENT (MA):** 682.18 MA

**Power (Watts):** 3.4414 W
<table>
<thead>
<tr>
<th>FRONT PLATE, AT 8V</th>
<th>CHIPS</th>
<th>QTY</th>
<th>4V</th>
<th>3.3V</th>
<th>POWER (WATS)</th>
<th>TOTAL CURRENT (MA)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>3</td>
<td>700</td>
<td>9.4%</td>
<td></td>
<td>18.0</td>
</tr>
<tr>
<td>DISPLAY MODULE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
SECTION 2

SOFTWARE DESCRIPTION

29 OCTOBER 1979
ISIS-II LINKER v2.1 WAS LINKED BY:
LINK /FI1ABSDIS.OBJ:/FI1INPUT.OBJ:/FI1INPUT.OBJ:/FI1ABSDIS.OBJ:/FI1INPUT.OBJ:

LINK MAP FOR :FI1CU.LNK(CDU)

SEGMENT INFORMATION
START STOP LENGTH REL NAME

3H8F 0 B CODE
3H2H 0 B DATA
7EH 0 B STACK

3H 1H 18H A ABSOLUTE
2H 2H 3H A ABSOLUTE
24H 26H 3H A ABSOLUTE

INPUT MODULES INCLUDED:
/FI1ABSDIS.OBJ(MAIN)
/FI1CU.LNK(CDU)
/FI1INPUT.OBJ(IN)
/FI1ABSDIS.OBJ(FLSP)
/FI1CU.LNK(FLSP)
/FI1INPUT.OBJ(UWF)
/FI1CPU.LNK(UWF)
/FI1ABSF.OBJ(AQF)
/FI1CPU.LNK(AQF)
/FI1ABSMI.OBJ(HWID)
/FI1IFF.OBJ(IFF)
/FI1FILE.BUF.OBJ(FIL)
/FI1GRID.OBJ(GRID)
/FI1CURDAT.OBJ(DAT)
/FI1FONTDA.OBJ(FONTDA)

UNRESOLVED EXTERNAL NAMES:
CP0094
CP0014
CP0034
DNTMETHADO
CP0029
CP0096
CP0027
ARBFB
CP0095
CP0033
CP0098
CP0104
CP0091
CP0105
CP0070
CP0028
ARWFA
DISPLAYPRESENTPOSITION
DISPLAYACTIVEWPPT
CP0089
IDIDMW
DAYBEARING
DAYSUBWPPT
DAY-NULLE
ISIS-II LINTER V2.1 WAS INVOKED BY:

LINK :F1:CDU.LNK &
  :F1:OPTS.OBJ :F1:SV4DAT.OBJ &
  PLNB.LIB :SYSTEM.LIB &
  TO :F1:SV4.LNK MAP PRINT (:F1:SV4.MAP)

LINK MAP FOR :F1:SV4.LNK(DV)

SEGMENT INFORMATION:
  START    STOP LENGTH REL NAME
    4028H        B CODE
    F3CH        B DATA
    A8H        B STACK
    3H  1AH  1BH A ABSOLUTE
    20H  22H  3H A ABSOLUTE
    24H  26H  3H A ABSOLUTE

INPUT MODULES INCLUDED:
  :F1:CDU.LNK(CDU)
  :F1:SV1.OBJ(DV1)
  :F1:SV2.OBJ(DV2)
  :F1:SV3.OBJ(DV3)
  :F1:SV4.OBJ(DV4)
  :F1:SV4DAT.OBJ(DAT)
  PLNB.LIB(EPO012)
  PLNB.LIB(EPO022)
  PLNB.LIB(EPO025)
  PLNB.LIB(EPO029)
  PLNB.LIB(EPO034)
  PLNB.LIB(EPO048)
  PLNB.LIB(EPO069)
  PLNB.LIB(EPO084)
  PLNB.LIB(EPO089)
  PLNB.LIB(EPO091)
  PLNB.LIB(EPO094)
  PLNB.LIB(EPO099)
  PLNB.LIB(EPO103)
  PLNB.LIB(EPO105)
ISIS-II LOCATER V2.1 INVOKED BY:
- LOCATE #2:16W,LNK, ORDER(CODE, STACK, DATA, MEMORY) &
  $$ CODE(35H) STACK(1600H) DATA(16030H) STACKSIZE(30H) &
  $$ RESTART &
  $$ PRINT(ID1:DAV, LCC) MAP

MEMORY MAP OF MODULE DIV
READ FROM FILE :F1:DW.LNK
WRITTEN TO FILE :F1:DIV
MODULE START ADDRESS 0028H

START STOP LENGTH REL. NAME

01H 21H 3H A ABSOLUTE
31H 1AH 1CH A ABSOLUTE
21H 22H 3H A ABSOLUTE
24H 2AH 3H A ABSOLUTE
28H 4853H 4829H B CODE
6000H 602FH 30H B STACK
6030H 6FF2H FC3H B DATA
6FF3H FF0FH BACOH B MEMORY
DECLARE ROAD LITERALLY '0CH';
R0A1 LITERALLY '0IH';
R0A2 LITERALLY '0CH';
R0A3 LITERALLY '0CH';
R0A4 LITERALLY '0SH';
R0A5 LITERALLY '0SH';
R0A6 LITERALLY '0KH';
R0A7 LITERALLY '0SH';
R0A8 LITERALLY '0SH';

COL1 LITERALLY '0AH';
COL2 LITERALLY '0CH';
COL3 LITERALLY '0AH';
COL4 LITERALLY '0AH';
COL5 LITERALLY '0SH';
COL6 LITERALLY '0AH';
COL7 LITERALLY '0AH';
COL8 LITERALLY '0SH';

COL9 LITERALLY '0SH';
COL10 LITERALLY '0AH';
COL11 LITERALLY '0SH';
COL12 LITERALLY '0AH';
COL13 LITERALLY '0AH';
COL14 LITERALLY '0SH';

COL15 LITERALLY '0AH';
COL16 LITERALLY '0SH';
COL17 LITERALLY '0AH';

COL18 LITERALLY '0SH';
COL19 LITERALLY '0SH';

COL20 LITERALLY '0SH';
COL21 LITERALLY '0SH';

COL22 LITERALLY '0SH';

COL23 LITERALLY '0SH';

TRUE LITERALLY '0FFFH';
FALSE LITERALLY '0';
FOREVER LITERALLY 'WHILE 1';
SWITCH LITERALLY '400H';
CNTL40000000 LITERALLY '00H';
HNCLREADCNTL40000000 LITERALLY '0CH';
HNCLREADCNTL40000000 LITERALLY '41H';
DIGIT LITERALLY '1';

OCAL LITERALLY '2';

NORTH LITERALLY '3';

EAST LITERALLY '4';

NORTH LITERALLY '5';

CLEAR LITERALLY '6';
ENTRY LITERALLY '8';

SOUTH LITERALLY '9';

OFFSH LITERALLY '530';
OFFSH LITERALLY '560';

LASTL LITERALLY '630';
STEP LITERALLY '640';

COPY LITERALLY '0';

TOT LITERALLY '1';

BERNSYN LITERALLY '22H';
ISIS-II PLM-80 V0.0 COMPILATION OF MODULE MAIN
OBJECT MODULE PLACED IN: F1:ASMMAIN.OBJ
COMPILER INVOID BY: PLM80 F1:ASMMAIN.SRC DATE: 22DEC78

STITLE('MAIN')
1 MAIN: DD:
2 1 DECLARE FOREVER LITERALLY 'WHILE 1':
3 1 ADDR(SAV): PROCEDURE(SAV) EXTERNAL:
4 2 DECLARE SAV BYTE:
5 2 END:
6 1 INIT(HARDWARE): PROCEDURE EXTERNAL:
7 2 END:
8 1 INIT(CI): PROCEDURE EXTERNAL:
9 2 END:
10 1 CALL: PROCEDURE EXTERNAL:
11 2 END:
12 1 PROCESS(SWITCH): PROCEDURE EXTERNAL:
13 2 END:
14 1 CALL INIT(HARDWARE):
15 1 CALL INIT(CI):
16 1 DO FOREVER:
17 2 CALL (SWITCH):
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 = 0000H 20
25 LINES READ
0 PROGRAM ERROR(S)

END OF PLM-80 COMPILATION
TITLE('CDU')

/* CLEAR, INIT, SCDU, SUSPEND, RESTART, LIMIT, TEST */

CDU DO:

ANDLIST INCLUDE(FL:CDUI.SRC)

DECLARE HORI LITERALLY 'OFDR',

VERT LITERALLY 'OFDR'

READ PROCEDURE(ICMPTR) EXTERNAL

DECLARE ICMPTR ADDRESS;

END;

CLEAR: PROCEDURE(ROW) EXTERNAL

DECLARE ROW BYTE;

END;

INSERT: PROCEDURE(ROW, COLUMN, PTR) EXTERNAL

DECLARE (ROW, COLUMN) BYTE, PTR ADDRESS;

END;

DISPLAY: PROCEDURE(ICMPTR, ROW) EXTERNAL

DECLARE ICMPTR ADDRESS;

END;

GRID: PROCEDURE(PTR) EXTERNAL

DECLARE PTR ADDRESS;

END;

UPDATESCREEN: PROCEDURE EXTERNAL

END;

INITIALIZER: PROCEDURE EXTERNAL

END;

SCREEN INTESE: PROCEDURE(LEVEL) EXTERNAL

DECLARE LEVEL BYTE;

END;

CLEAR: PROCEDURE(LINE) EXTERNAL

DECLARE LINE BYTE;

END;

INITIALIZATION: PROCEDURE EXTERNAL

END;

SCREEN: PROCEDURE EXTERNAL

END;

INITIALIZE: PROCEDURE EXTERNAL

END;

INITIALIZE: PROCEDURE EXTERNAL

END;

WPA SUBNODE: PROCEDURE EXTERNAL

END;

END;

END;

ENDS;

ENDS;

ENDS;

ENDS;

ENDS;

ENDS;

ENDS;

ENDS;

ENDS;

ENDS;

ENDS;

ENDS;

ENDS;

ENDS;

ENDS;

ENDS;

ENDS;

ENDS;

ENDS;

ENDS;

ENDS;
DECLARE CURRENTDISP(9) STRUCTURE(CHAR(24) BYTE) EXTERNAL;
  SWINDEX BYTE EXTERNAL;
  SAV BYTE EXTERNAL;
  WNFLSTATUS BYTE EXTERNAL;
  UNFACTIVECHAN BYTE EXTERNAL;
  UNFFRDF(10) STRUCTURE(DIGITS(4) BYTE) EXTERNAL;
  UNFFRDC(10) STRUCTURE(DIGITS(6) BYTE) EXTERNAL;
  ADFASTATUS BYTE EXTERNAL;
  ADFACTIVECHAN BYTE EXTERNAL;
  ADFFRDF(10) STRUCTURE(DIGITS(4) BYTE) EXTERNAL;
  CENVSTATUS BYTE EXTERNAL;
  CENVACTIVECHAN BYTE EXTERNAL;
  CENVFRDF(10) STRUCTURE(DIGITS(5) BYTE) EXTERNAL;
  EFFSTATUS BYTE EXTERNAL;
  EFFWNAME BYTE EXTERNAL;
  DINVSTATUS BYTE EXTERNAL;
  DINVANCE BYTE EXTERNAL;
  DINVBEARING BYTE EXTERNAL;
  DINVTIME(10) BYTE EXTERNAL;

DECLARE OFFHONSTATE(6) BYTE PUBLIC DATA('OFFON '11
DECLARE TEMPBUF(24) BYTE PUBLIC

DECLARE BLANKLINE(24) BYTE PUBLIC AT(3)
  DATA(' '11

DECLARE PR1ENABLEL1ST STRUCTURE
  SHMAG(7) BYTE,
  SHMVALUE(4) BYTE DATA(0BH,0BH,0BH,0BH,0BH,0BH,0BH,0BH),
  01H,23H,45H,67H)

DECLARE PR1(IO) STRUCTURE
  MODE BYTE,
  N.MACH BYTE,
  D.MACH BYTE,
DECLARE PRISTAELEAU(4) BYTE DATA:
ROM0,'FM',COL12,'DM'.
ROM1,COL13,'RE',COL23,'M'.
ROM2,'WH',COL12,'E'R'.
ROM3,COL13,'TG',COL23,'M'.
ROM4,'AOF'.
ROM5,'DN',COL12,'IFF',01;

DECLARE DIGITKBSGRID(4) BYTE PUBLIC DATA:
HORZ,0,104,22,92.
HORZ,2,104,52,72,92.
HORZ,92,104,42,62,92.
HORZ,110,122,22,32,52,62,72,82,92.
HORZ,126,140,22,32,42,52,62,72,82,92.
VERT,32,92,2,32,62,92.
VERT,22,92,104,110,122,128,140.
VERT,42,52,104,110,122,128,140.
VERT,62,72,104,110,122,128,140.
VERT,82,92,104,110,122,128,140,OFFH;

DECLARE SUBMODESKGRID(4) BYTE PUBLIC DATA:
HORZ,6,143,22,92.
HORZ,2,122,52,72,92.
VERT,32,92,2,32,62,92,122,OFFH;

DECLARE OCTALKBSTABLEAU(4) BYTE PUBLIC DATA:
ROM0,COL16,'1',COL19,'2',COL22,'3'.
ROM4,COL16,'4',COL19,'5',COL22,'6'.
ROM6,COL16,'7',COL19,'8',COL22,'9'.
ROM8,COL16,'0',COL19,'0',COL22,'E'.01;

DECLARE OCTALKBSTABLEAU(4) BYTE PUBLIC DATA:
ROM0,COL16,'1',COL19,'2',COL22,'3'.
ROM4,COL16,'4',COL19,'5',COL22,'6'.
ROM6,COL16,'7'.
ROM8,COL16,'C',COL19,'0',COL22,'E'.01;

DISPLAYDIGITSB: PROCEDURE PUBLIC:
CALL GRID1,DIGITKBSGRID);
CALL DISPLAY1,DIGITKBSTABLEAU);
END
DECLARE COUNTSTACK(40) ADDRESS,
CDU$SPV ADDRESS,
SYSS$SPV ADDRESS;

INITICDU: PROCEDURE PUBLIC:

DISABLE;
COUNTSTACK(LENGTH(COUNTSTACK)-1) = .COUNTTOP;
COUNTSPV = .COUNTSTACK(LENGTH(COUNTSTACK)-1);
CALL INITIALIZE;
CALL INITAVF;
CALL INITADF;
CALL INITAIV;
CALL INITSIF;
ENABLE;
END; /* OF INITICDU */

SUSPEND: PROCEDURE PUBLIC:

DISABLE;
COUNTSPV = STACKPTR;
STACKPTR = SYSSSPV;
ENABLE;
END;

RESTART: PROCEDURE PUBLIC:

DISABLE;
COUNTSTACK(LENGTH(COUNTSTACK) - 1) = .COUNTTOP;
COUNTSPV = .COUNTSTACK(LENGTH(COUNTSTACK)-1);
STACKPTR = SYSSSPV;
ENABLE;
END;
DECLARE CODE BYTE;

DECLARE STEPPABLE STRUCTURE:
  SWSH(7) BYTE,
  SWVALUE(1) BYTE "$A( 
  0.0.0.0.0.0,0.0, /* CLEARSM */
  001);"

DECLARE ERRORSTRUCTURE:
  MODE BYTE,
  NUMMACH BYTE,
  DLMADD ADDR, ADDRESS,
  ECHOMND BYTE,
  ECHOCOL BYTE,
  SHMENABLE ADDRESS) DATA(SWITCH,0.0.0.0.0..STEPENABLE);"

DECLARE MSG0(+) BYTE DATA(R02,"INVALID ENTRY",O),
    MSG1(+) BYTE DATA(R02, "NO UNALLOCATED TARGETS",O),
    MSG2(+) BYTE DATA(R02, "NO DATA FOR THIS CHP",O);"

DECLARE MSGADD(3) ADDRESS DATA(MSG0, MSG1, MSG2);"

CALL MOVE(R24, CURRENTDISPLAY), TEMPBUF); /*SAVE CURRENT DISPLAY ROW 2 */
CALL CLEARLINE(2);"
CALL DISPLAY(MSGADD(1));"
CALL READ$(ERROR1CB);"
CALL CLEARLINE(2);"
CALL UPDATEALINE(2,0.24, TEMPBUF); /*RESTORE DISPLAY ROW 2 */
END1 /* OF ERROR */
LIMITTEST: PROCEDURE(BUFPTR,NUMDIGITS,MINPTR,MAXPTR) 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:
MINSTRING <= STRING <= MAXSTRING
AND A VALUE OF 0 OTHERWISE. */

DECLARE NUMDIGITS BYTE;
(BUFPTR,MINPTR,MAXPTR) ADDRESS;

DECLARE (N+1) BYTE;
(BUF BASED BUFPTR)(1) BYTE;
(MIN BASED MINPTR)(1) BYTE;
(MAX BASED MAXPTR)(1) BYTE:

IF (BUF(1) < MIN(1)) OR (BUF(0) > MAX(0)) THEN RETURN 0; /* FAIL */
IF NUMDIGITS > 2 THEN N = NUMDIGITS - 2;
ELSE N = 0;
DO I = 0 TO N:
IF MIN(I) < MAX(I) THEN DO:
IF (MIN(I) < BUF(I)) AND (BUF(I) < MAX(I)) THEN RETURN 1; /* OK */
IF BUF(I) = MIN(I) THEN DO:
IF BUF(I+1) < MIN(I+1) THEN RETURN 0; /* FAIL */
IF BUF(I+1) > MIN(I+1) THEN RETURN 1; /* OK */
END;
ELSE DO:
IF BUF(I) = MAX(I) THEN DO:
IF BUF(I+1) > MAX(I+1) THEN RETURN 0; /* FAIL */
IF BUF(I+1) < MAX(I+1) THEN RETURN 1; /* OK */
END:
/ OF ELSE DO */
END;
/ OF IF MIN(I) < MAX(I) THEN DO */
ELSE DO:
IF BUF(I) < MIN(I) THEN RETURN 0; /* FAIL */
IF BUF(I+1) > MIN(I+1) THEN RETURN 1; /* OK */
END;
/* OF ELSE DO */
ELSE DO:
IF BUF(I) < MAX(I) THEN RETURN 0; /* FAIL */
IF BUF(I+1) > MAX(I+1) THEN RETURN 1; /* OK */
END;
/ OF ELSE DO */
RETURN 1; /* OK */
END; /* OF LIMITTEST */
REJECT

129 1 DISPLAYCURRENTSTATUS: PROCEDURE;

130 2 FORMATPROCEDURE(CHAN, STATUS, FREQUENCY, NOLOGITS, ROW, COL, MASK);

131 3 DECLARE (CHAN, STATUS, NOLOGITS, ROW, COL) BYTE;
   (FREQUENCY, MASK) ADDRESS;

132 3 CALL CLEARDISPLAYBUF;

133 3 IF (STATUS AND 2H) > 0 THEN TEMPSBUF(0) = 'A';

134 3 CALL INSERTLOGITS(STATUS, TEMPSBUF(0), TEMPSBUF(1), TEMPSBUF(2), MASK);

135 3 CALL UPDATENAME(ROW, COL, NOLOGITS2, TEMPSBUF);

136 3 END;

137 2 CALL FORMATREQ(VHFACTIVECHAN, VHFSTATUS, VHFREQ, 4, 6, 3, 20000H);

138 2 CALL FORMATREQ(UHFACTIVECHAN, UHFSTATUS, UHFREQ, 6, 2, 3, 10000H);

139 2 CALL FORMATREQ(UHFACTIVECHAN, UHFSTATUS, UHFREQ, 4, 4, 2, 01);

140 2 CALL FORMATREQ(UHFACTIVECHAN, UHFSTATUS, UHFREQ, 5, 6, 3, 10000H);

/* DISPLAY IFF STATUS LINE */

141 2 CALL CLEARDISPLAYBUF;

142 2 IF (IFFSTATUS AND 2H) > 0 THEN TEMPSBUF(0) = 'A';

143 2 IF IFFSTATUS THEN TEMPSBUF(1) = '1';

144 2 IF (IFFSTATUS AND 2) > 0 THEN TEMPSBUF(2) = '2';

145 2 IF (IFFSTATUS AND 4) > 0 THEN DO:

146 3 TEMPSBUF(3) = '3';

147 3 TEMPSBUF(4) = '4';

148 3 END;

149 2 IF (IFFSTATUS AND 1H) > 0 THEN TEMPSBUF(5) = 'C';

150 2 IF (IFFSTATUS AND 8) > 0 THEN DO:

151 3 TEMPSBUF(6) = '4';

152 3 IF IFFMSMODE THEN TEMPSBUF(7) = 'B';

153 3 ELSE TEMPSBUF(7) = 'A';

154 3 END;

155 2 CALL UPDATENAME(6, 15, 8, TEMPSBUF);

/* DISPLAY DAV STATUS */

156 2 CALL DISPLAYACTIVE(NAKPT);

157 2 IF SHO(NAVSTATE, 4) THEN CALL UPDATELINE(6, 15, 1, 'A');

158 2 CALL INSERT('A', NAVSTATE, TEMPSBUF, '1', 10000H);

159 2 CALL UPDATELINE(11, 17, 5, TEMPSBUF);

160 2 CALL UPDATELINE(2, 19, 3, NAVSTATE);

161 2 CALL UPDATELINE(3, 19, 3, NAVSTATE);

162 2 CALL DISPLAYPRESENTPOSITION(81);

163 2 END; /* OF DISPLAYCURRENTSTATUS */
REJECT

203 1  COUHONOFF: PROCEDURE;

204 2  IF SW1 = OFF THEN CALL SCREEN: INTENSITY(0);
206 2  ELSE CALL SCREEN: INTENSITY(5); END;

208 1  COUH PROCEDURE PUBLIC:

209 2  DISABLE;
210 2  SYSISP = STACKPTR;
211 2  STACKPTR = COUSPV;
212 2  ENABLE;
213 2  END;

214 1  COUNTOP: PROCEDURE:

/ * GENERATE PRIMARY TABLEAU */
215 2  CALL CLEAR();
216 2  CALL DISPLAY(.PRINTTABLEAU);
217 2  CALL DISPLAY(COUNTOP); STATUS;
218 2  CALL READ(.PRINT); IC;
219 2  DO CASE SW1INDE;
220 3  CALL VHF8SUBMODE;
221 3  CALL IMASUBMODE;
222 3  CALL UNASUBMODE;
223 3  CALL AFASUBMODE;
224 3  CALL OHFASUBMODE;
225 3  CALL IIFASUBMODE;
226 3  CALL COUNTOP;
227 3  END / * OF DO CASE */

/ * TERMINATE CPU PROCESSING */
228 2  DISABLE;
229 2  COUNSTACK(LENGTH(COUNSTACK)-1) = COUNTOP;
230 2  COUSPV = COUNSTACK(LENGTH(COUNSTACK)-1);
231 2  STACKPTR = SYSISP;
232 2  ENABLE;
233 2  END / * OF COUNTOP */

234 1  END / * OF CPU DO */

MODULE INFORMATION:

CODE AREA SIZE = 061BH 15600
VARIA BLE AREA SIZE = 007FH 1270
MAXIMUM STACK SIZE = 000EH 140
400 LINES READ
0 PROGRAM ERROR(S)

END OF PL/M-80 COMPILATION
$TITLE('INPUT')

1 IN: IDI

DECLARE PRIMARY 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 [ROM, COL, CHAR, TEXTP] EXTERNAL
12 2 DECLARE [ROM, COL, CHAR] BYTE; TEXTP ADDRESS
13 2 END

14 1 DECLARE ICMPTR ADDRESS,
   ICMPTR STRUCTURE:
   MODE BYTE, /* INPUT MODE */
   NUMICH BYTE, /* NUMBER OF CHS TO BE INPUT */
   DELICH BYTE, /* DELIMITER CHARACTER */
   DELIMASK ADDRESS, /* DELIMITER MASK */
   ECHICH BYTE,
   SIMECH ADDRESS,
   SIMECH BYTE,
   SIMESIZE BYTE,
   ECHICH BYTE,
   SIMECH ADDRESS,
   SIMESIZE BYTE,
   CHICTRT BYTE,
   DISPLAYBUF ADDRESS
   CLEARSTATE BYTE

15 1 DECLARE
   INMODE BYTE,
   INPUTBUFFER BYTE,
   SENTEZE SIZE BYTE,
   DISPLAYBUF BYTE,
   DISPMASK ADDRESS,
   INDELIMASK ADDRESS,
   CHICOUNT BYTE,
   DISPLAYBUF ADDRESS
   CLEARSTATE BYTE

16 1 DECLARE
   DUR 역시 BYTE PUBLIC,
   DATAENTERED BYTE PUBLIC,
   SIMINDEX BYTE PUBLIC,
   SWH BYTE PUBLIC

17 1 DECLARE SIMQUEUE BYTE

18 1 ADDRESS: 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 (SINQUEUE = OFFH) AND INPUTENABLE THEN SINQUEUE = TEMP;
22 2 END1 /* OF ADDQUEUE */

23 1 REMOVEQUEUE: 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 THEREFORE INTERRUPTS
      MUST BE DISABLED WHEN MODIFYING THE QUEUE VARIABLES. */
24 2 DECLARE SINOQUEUE BYTE:

25 2 DISABLE;
26 2 IF SINQUEUE = OFFH THEN DO:
28 3 SINOQUEUE = SINQUEUE;
29 3 SINQUEUE = OFFH;
30 3 ENABLE;
31 3 RETURN(SINOQUEUE);
32 3 END;
33 2 END;
34 2 RETURN(OFFH);
35 2 END1 /* OF REMOVEQUEUE */

36 1 INITIALIZEQUEUE: PROCEDURE PUBLIC1
37 2 DISABLE;
38 2 SINQUEUE = OFFH;
39 2 ENABLE;
40 2 END;

\
OBJECT
CLEARINPUTFIELD: 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.) */
DECLARE (1,10) BYTE;
   CLEARCOUNT = 0;
   DISPLLOC = 0;
   DISPAMSK = 8000H;
   CLEARSTATE = TRUE;
   FIELDSIZE = 0;
   /* THE TEST ON NUMACH WAS ADDED TO ALLOW CLEARINPUTFIELD TO
   BE INVOKED WHEN THE INPUT MODE IS EQUAL TO ZERO (SWITCH). */
   IF (ICB.NUMACH > 0) THEN N = ICB.NUMACH - 1;
   ELSE N = 01;
   DO I = 0 TO N;
      IF (DISPAMSK AND NUMELMMASK) > 0 THEN DOJ
      DISPHBUF(FIELDSIZE) = ICB.ELMACH
      FIELDSIZE = FIELDSIZE + 11
      END;
      DISPAMSK = SHR(DISPAMSK,11);
      NUMBUF(I) = '0';
      DISPHBUF(FIELDSIZE) = '?'
      FIELDSIZE = FIELDSIZE + 11
      END; /* OF DO I */
   DISPAMSK = 8000H;
   END; /* OF CLEARINPUTFIELD */
SELECT
44 1 READ PROCEDURE (ICBADD) PUBLIC;
45 2 DECLARE ICBADD ADDRESS /
46 3 /* INPUT MODE CONTROL VARIABLE FORMAT:
47 4 ICB.MODE AND OFH = 0 = SWITCH
48 5 1 = DIGIT
49 6 2 = OCTAL DIGIT
50 7 IF BIT 7 = 1 THEN READ RETURNS AFTER ICB.NUMCH DIGITS HAVE BEEN
51 8 ENTERED (ENTERAGAIN NOT REAUSED).
52 9 IF BIT 6 = 1 THEN THE INPUT FIELD IS NOT CLEARED UNTIL AFTER THE
53 10 FIRST CHARACTER HAS BEEN ENTERED.
54 */
55 2 ICBMODE = ICBADD;
56 2 INMODE = ICB.MODE;
57 2 DATAMINPUTED = FALSE;
58 2 INPUTEMASK = ICB.EMASK;
59 2 CALL CLEARINPUT FIELD1
60 2 IF (INMODE AND 0FH) = 0 THEN
61 2 CALL UPDTEPICB (ICB.ECHONMB, ICB.CHONCOL, FIELDMITI,.DISPMoB);
62 2 CALL UPDATESCREEN;
63 2 INPUTENABLE = TRUE; /* RESET BY INPUT PROCEDURES */
64 2 DO WHILE INPUTENABLE;
65 3 CALL SUSPND; /* WAIT FOR INPUT */
66 3 END;
67 2 END; /* OF READ */
PROCESS:SWITCH PROCEDURE PUBLIC:
DECLARE ENABLELISTPTR ADDRESS;
ENABLELIST BASED ENABLELISTPTR STRUCTURE:
SMASK(7) BYTE,
SMVALUE(1) BYTE,
MASK BYTE,
(IJH.ROW.COL.CHAR) BYTE;

PROCESS:CH PROCEDURE:
DECLARE CH BYTE;
/* ADD A NEW CHARACTER TO THE INPUT BUFFER AND THE DISPLAY BUFFER
(AVOIDING DELIMITER CHARACTERS), */
CLEARSTATE = TRUE;
INBUF(CHCOUNT) = CH;
CHCOUNT = CHCOUNT + 1;
IF (DISPMASK AND INDELMASK) > 0 THEN DO:
   DISPLocate = DISPLocate + 11
END
DISPMASK = SHR(DISPMASK, 11)
DISPMASK = SHR(DISPMASK, 11)
CALL UPDATELINE(1CB.ECHORow.1CB.ECHOCOL.FIELDSIZE,DISPSBF)
END //* OF ADDCH */
REMOVACH: PROCEDURE;

/* INVOKED IF CH = CLEAR */
IF CLEARSTATE = TRUE (THE INITIAL STATE) THE LAST CHARACTER ENTERED IS DELETED AND CLEARSTATE SET FALSE.
IF CLEARSTATE = FALSE THEN THE ENTIRE INPUT FIELD IS DELETED. */

97 4 IF CHACCOUNT > 0 THEN DO:
99 5 IF CLEARSTATE THEN DO:
101 6 CLEARSTATE = FALSE; /* CLEAR THE CHARACTER */
102 6 CHACCOUNT = CHACCOUNT - 1;
103 6 DISPLOC = DISPLOC - 11
104 6 DISPBUF(DISPLoc) = "?";
105 6 DISPmask = SHL(DISPmask,11);
106 6 IF (DISPmask AND ICB.DELIMASK) > 0 THEN DISPLOC = DISPLOC - 11
108 6 END; /* OF IF CLEARSTATE */
109 5 ELSE DO:
110 6 CALL CLEARINPUTFIELD; /* CLEAR THE ENTIRE FIELD */
111 6 CLEARSTATE = TRUE;
112 6 END;
113 5 CALL UPDATE LINE1 (ICB.ECHOPR, ICB.ECHOSZ, FIELD.SI ZE, DISPBUF);
114 3 END;
115 4 END; /* OF REMOVE */

/* BEGIN PROCESSACH CODE */

116 3 IF CHAR = CLEARSM THEN DO:
118 4 CALL REMOVACH:
119 4 CALL UPDATE SCREEN;
120 4 RETURN;
121 4 END;
122 3 IF CHAR = ENTERSM THEN DO:
124 4 INPUTENABLE = FALSE;
125 4 RETURN;
126 4 END;
127 3 DATAENTERED = TRUE; /* FIRST CHARACTER RECEIVED */
129 3 IF CHACCOUNT < ICB.NUMCHAR THEN DO:
130 4 CALL ADDCHAR(CHAR); /* ADD THE NEW CHARACTER */
131 4 CALL UPDATE SCREEN;
132 4 END;
133 3 IF ((TERM MODE AND 80H) > 0) AND (CHACCOUNT = ICB.NUMCHAR) THEN INPUTENABLE = FALSE;
135 3 END; /* OF PROCESSACH */
DECLARGE DIGITASM: 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
16(8)    0
46(8)    C
47(8)    E
*/

DECLARE (R,C BYTE);  
C = SW AND 71;  
R = SHR(SW,3) - 1;  
IF (R < 4) AND (C > 4) THEN DO:  
    C = C - 4 + R+1;  
    IF C = 11 THEN C = 0;  
RETURN(C + 30H);END;  
ELSE RETURN(0FFH); END; /* OF DECLARGE DIGITASM */

DECLAR0H BYTE;  
DCODESOCTALASM: PROCEDURE BYTE;  
CH = DECLARGE DIGITASM;  
IF CH = '7' THEN RETURN(0FFH);  
ELSE RETURN(CH);END;

DECLAR0H/ALPHA8H: PROCEDURE BYTE;  
IF (SW > 440) OR (SW < 100) THEN RETURN(0FFH);  
CH = SW + 29H;  
IF (CH = 'I') OR (CH = 'O') THEN RETURN(0FFH);  
RETURN CH;END; /* OF DECLAR0H/ALPHA8H */

DECLAR0H/ESTEASH: PROCEDURE BYTE;  
IF SW = 250 THEN RETURN ('E' );  
IF SW = 270 THEN RETURN ('M');  
RETURN OFFH;  
END; /* OF DECLAR0H/ESTEASH */

DECLAR0H/NORTH/SOUTHASM: PROCEDURE BYTE;  
IF SW = 160 THEN RETURN ('M');  
IF SW = 340 THEN RETURN ('S');  
RETURN OFFH;  
END; /* OF DECLAR0H/NORTH/SOUTHASM */
REJECT

/* BEGIN PROCESS/SWITCH CODE */

178 2 IF NOT INPUTENABLE THEN RETURN;
180 2 SW = REMOVESW;  /*GET NEXT CH FROM INPUT QUEUE */
181 2 IF SW = OFFH THEN RETURN;  /* INPUT QUEUE EMPTY */
183 2 IF INMODE O SWITCH THEN DO:
185 3 IF (SW = CLEARSW) OR (SW = ENTERSW) THEN CHAR = SWV;
ELSE
187 3 DO CASE (INMODE AND OFF) = 11
188 4 CHAR = DECODEXSWISH;
189 4 CHAR = DECODEXSWISH;
190 4 CHAR = DECODEXSWISH;
191 4 CHAR = DECODEXSWISH;
192 4 END;
194 3 IF CHAR OFFH THEN DO:
196 4 CALL PROCESSSWH;
197 4 RETURN;
198 4 END;
199 3 END;  /* OF IF INMODE ...DO: */

/* INPUT MODE IS NOT "DIGIT" OR
IS "DIGIT" AND DIGIT SWITCH WAS NOT PRESSED.
TEST FOR SPECIAL FUNCTION KEY. */

200 2 I = SW - PRIMARYSW;

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 ONEMER;
CALL ZEROQCODEI;
GO TO A1;
GO TO A1;
GO TO A1;
CALL APLYI/;
END;
RETURN;
A1:
END;
*/

/* INPUT IS NOT A SPECIAL FUNCTION SWITCH SO IT MUST BE AN
ENABLED SWITCH. DETERMINE RELATIVE INDEX. */

203 2 ENABLELISTPTR = ICLOBALENABLEPTR;
204 2 ROW = SHR(SAV,31);
205 2 COL = SWA AND 7;
206 2 IF COL > 0 THEN MASK = SHR(80H,COL1);
208 2 ELSE MASK = 80H;
209 2 INDEX = 0;
210 2 IF (MASK AND ENABLELIST,COL,ROW1) > 0 THEN DO:
*/

/* DETERMINE THE INDEX VALUE OF THIS SWITCH BY COUNTING THE NUMBER
OF 1'S IN THE MASK WORDS PRECEEDING THE MASK FOR THIS ROW. */
IF ROW > 0 THEN DO:
   N = ROW - 1;
   DO I = 0 TO N:
      DO WHILE MASK > 0:
         IF (MASK AND 80H) > 0 THEN SWINDEX = SWINDEX + 1;
         MASK = SHR(MASK, 1);
      END:
   END:
END; /* OF IF ROW > 0 */

/* COUNT THE NUMBER OF 1'S IN THE MASK WORD FOR THIS ROW */
IF COL > 0 THEN DO:
   N = COL - 1;
   DO I = 0 TO N:
      IF (MASK AND 80H) > 0 THEN SWINDEX = SWINDEX + 1;
      MASK = SHR(MASK, 1);
   END:
END; /* OF IF COL > 0 */

I = SWINDEX / 2; /* A VALUE NIBBLES ARE PACKED 2 PER BYTE. */
IF SWINDEX THEN SWINDEX = ENABLELIST.SWVALUE(I) AND 0FH
ELSE SWINDEX = SHI(ENABLELIST.SWVALUE(I), 4);
INPUTENABLE = FALSE;
END; /* OF IF MASK AND ENABLELIST... */
END; /* OF PROCESSSWITCH */

END; /* OF IN: DO */

MODULE INFORMATION:

  CODE AREA SIZE   = 038BH   1339D
  VARIABLE AREA SIZE = 0043H   67D
  MAXIMUM STACK SIZE = 0044H   10D
  419 LINES READ
  0 PROGRAM ERROR(S)

END OF PL/M-80 COMPILATION
$TITLE('DISP')

1 DISP: DO;
   /* CPU DISPLAY PROCEDURES */

2 1 DECLARE TRUE LITERALLY 'OFFH',
      FALSE LITERALLY '0';

3 1 DECLARE PHBUF(1) BYTE EXTERNAL;
    RHBUF(1) BYTE EXTERNAL;

4 1 DECLARE BLANKLINE(24) BYTE EXTERNAL;
5 1 UPDATEAR: PROCEDURE EXTERNAL;
6 2 END;
7 1 FILLARF: PROCEDURE(ROW-COL-BUFN-ADDR-REF1-REF2-NCHAR-NODEL) EXTERNAL;
8 2 DECLARE (ROW-COL-NCHAR-NODEL) BYTE, (BUFADD-REF1-REF2ADDR) ADDRESS;
9 2 END;

10 1 DECLARE CURRENT_DISP(9) STRUCTURE(CHAR(24) BYTE) PUBLIC;

11 1 UPDATE_SCREEN: PROCEDURE PUBLIC:
12 2 CALL UPDATEAR; /* INITIATE DATA TRANSFER TO REFRESH MEMORY */
13 2 END;

14 1 UPDATE_LINE: PROCEDURE(ROW-COL-NCHAR-TEXT-ADDR) PUBLIC:
   /* THE INPUT MUST BE ASCII TEXT */
15 2 DECLARE ROW BYTE, /* ROW NUMBER */
    COL BYTE, /* STARTING COLUMN NUMBER */
    NCHAR BYTE, /* NUMBER OF CHAR TO BE UPDATED */
    TEXTPTR ADDRESS, /* TEXT STRING STARTING ADDRESS */
    CURPTR ADDRESS;
16 2 DECLARE CURPTR ADDRESS;
17 2 DECLARE (TEXT BASED TEXT-ADDR(1)) BYTE,
    (CHAR BASED CURPTR(1)) BYTE,
    i BYTE;
18 2 CURPTR = CURRENT_DISP(ROW), CHAR(COL);
19 2 CALL FILLARF(ROW-COL-TEXT-ADDR-CURPTR-NCHAR, TRUE);
20 2 NCHAR = NCHAR - 1;
21 2 DO I = 0 TO NCHAR;
22 3 IF TEXT(I) 0 ' ' THEN CHAR(I) = TEXT(I);
23 3 END;
24 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 (RAM 9).
   note: data waypoint data overlays RAMBUF. */

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 RAMBUF(K) = 0;
34 2 CALL MOVE(N, RAMBUF(K), RAMBUF(K+1))
35 2 DO N = K TO IS35;
36 3 RAMBUF(N) = RAMBUF(N) AND OFOH;
37 3 END;
38 2 CURRENT(1): DISP(ROW).CHAR(O) = ' '; 
39 2 CALL MOVE(215 - ROW+24, CURRENT(1): DISP(ROW),
39 3 CURRENT(1): DISP(ROW).CHAR(1));
40 2 END;

41 1 CLEAR_LIN: PROCEDURE(LINENUM) PUBLIC;
42 2 DECLARE LINENUM BYTE;
43 2 CALL FILLBUF(LINENUM,0,BLANK_LIN,.CURRENT(1):DISP(LINENUM).24,FALSE));
44 2 CALL MOVE(24..BLANK_LIN,.CURRENT(1):DISP(LINENUM));
45 2 END;
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. */

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

NEWROW: PROCEDURE;

DECLARE DESTPTR ADDRESS,
(DEST BASED DESTPTR)(1) BYTE,
I BYTE;

DESTPTR = CURRENT(10)(SP,ROW);
CALL FILLBUF(ROW,0,LINE,DESTPTR,24,TRUE)
DO I = 0 TO 23:
   IF LINE(I) O' ' THEN DO:
      DEST(I) = LINE(I);
      LINE(I) = ' ' /BLANK LINE FOR NEXT TIME AROUND /
   END;
END;
ROW = N:
COL = 01
END;

REPEAT: PROCEDURE:

DECLARE I BYTE;

IF (N = 01) OR (N > 24) THEN RETURN;
IF N + COL > 23 THEN N = 23 - COL;
II = IX + 11
CH = IN(I11); /* GET THE CHARACTER TO BE REPEATED */
N = N - 11
DO I = 0 TO N:
   LINE(COL) = CH
   COL = COL + 11
END;
END;
/\ BEGIN DISPLAY CODE */
77 2 LINE(0) = ' ' /* CLEAR THE LINE BUFFER */
78 2 CALL MOVE(2), LINE(1), LINE(1)
79 2 ROW = IN(0) AND 1FH;
80 2 CH = IN(1);
81 2 IX = IX
82 2 COL = 0;
83 2 DO WHILE CH > 0;
84 3 IF (CH AND 1FH) = 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 IX = CH AND 1FH; /* PROCESS CONTROL CHARACTER */
91 4 KEY = CHR(IX) AND 3;
92 4 DO CASE KEY:
93 5 CALL NEARROW; /*ROW SPECIFIER ENCOUNTERED */
94 5 COL = IX; /*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 + 11
100 3 CH = IN(1);
101 3 END; /* OF DO WHILE */
102 2 CALL NEARROW; /* PROCESS LAST ROW */
103 2 END; /* OF DISPLAY */
OBJECT:

104 1 INSERT: PROCEDURE(NCHAR, SOURCEPTR, DESTAPTR, DELMACH, DELMASK)
PUBLIC;

/* COPY CHARACTERS FROM A SOURCE BUFFER TO A DESTINATION BUFFER AND
INSERT A DELIMITER 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 SCAN 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*/
SOURCEPTR ADDRESS,
DESTAPTR ADDRESS,
DELMACH BYTE,
DELMASK ADDRESS;

106 2 DECLARE (SOURCE BASED SOURCEPTR(1)) BYTE,
(DEST BASED DESTAPTR(1)) BYTE,
(I,LOC) BYTE;

107 2 LOC = 01
108 2 I = 01
109 2 DO WHILE I < NCHAR
110 3 IF (DELMASK AND 8000H) > 0 THEN DEST(LOC) = DELMACH
112 3 ELSE DO;
113 4 DEST(LOC) = SOURCE(I);
114 4 I = I + 1
115 4 ENDI
116 3 DELMASK = SHL(DELMASK,11);
117 3 LOC = LOC + 11;
118 3 ENDI
119 2 ENDI /* OF INSERT */
120 1 ENDI /* OF DISP: DO */

MODULE INFORMATION:

CODE AREA SIZE = 0304H 9860
VARIABLE AREA SIZE = 0114H 2760
MAXIMUM STACK SIZE = 000CH 120
191 LINES READ
0 PROGRAM ERROR(S)

END OF PL/I-80 COMPILATION
ISIS-II PL/I-P-80 V2.1 COMPILATION OF MODULE Card
OBJECT MODULE PLACED IN :F1:COM.OBJ
COMPILER INVOKED BY: PL/11-1=COM.SRC DATE(30OCT79) DEBUG

$TITLE( 'COM' )
1   COM: DO;
2      INCLUDE( F1:CO slut.SRC )
3 1   ERROR: PROCEDURE( CODE ) EXTERNAL;
4 2   DECLARE CODE BYTE;
5 3   END;
6 2   CLEAR: PROCEDURE( STARTROM ) EXTERNAL;
7 2   DECLARE STARTROM BYTE;
8 2   END;
9 1   GRID: PROCEDURE( POINTER ) EXTERNAL;
10 2   DECLARE POINTER ADDRESS;
11 2   END;
12 1   DISPLAY: PROCEDURE( INAPTR ) EXTERNAL;
13 2   DECLARE INAPTR ADDRESS;
14 2   END;
15 1   READ: PROCEDURE( ICAPTR ) EXTERNAL;
16 2   DECLARE ICAPTR ADDRESS;
17 2   END;
18 1   UPDATENLINE: PROCEDURE( ROW, COL, NCHAR, TEXT$PTR ) EXTERNAL;
19 2   DECLARE ( ROW, COL, NCHAR ) BYTE, TEXT$PTR ADDRESS;
20 2   END;
21 2   CLEARTEXTBUF: PROCEDURE EXTERNAL;
22 2   END;
23 1   INSERT: PROCEDURE( NCHAR, SOURCE$PTR, DESTAPTR, DELIMCH, DELMMASK ) EXTERNAL;
24 2   DECLARE ( NCHAR, DELIMCH ) BYTE, ( SOURCE$PTR, DESTAPTR, DELMMASK ) ADDRESS;
25 2   END;
26 1   UPDATESCREEN: PROCEDURE EXTERNAL;
27 2   END;
28 1   DISPLAYDIGITW: PROCEDURE EXTERNAL;
29 2   END;
30 1   LIMITTEST: PROCEDURE( BUFAPTR, NUMDUGITS, MINAPTR, MAXAPTR ) BYTE EXTERNAL;
31 2   DECLARE ( BUFAPTR, NUMDUGITS, MAXAPTR ) ADDRESS,
32        NUMDUGITS BYTE;
33 2   END;
34 1   DECLARE ACTIVCHAN BYTE;
35 1   LASTACTIVCHAN BYTE;
36 1   FREEMBASEADD ADDRESS;
37 1   FREEMSIZE BYTE;
38 1   FREEMSIZEPLUSONE BYTE;
39 1   INSERTMASK ADDRESS;
40 1   FREEMMINADD ADDRESS;
41 1   FREEMMINADD ADDRESS;
42 1   MINIMUMADD ADDRESS;
43 1   MAXIMUMADD ADDRESS;
44 1   NUMFREEADD ADDRESS;
45 1   BLANKLINE(24) BYTE EXTERNAL;
46 1   J IMBUF(16) BYTE EXTERNAL;
47 1   END;
48 1   LOCAL VARIABLES WHOSE VALUES ARE OBTAINED FROM THE CURRENT
49 1   CONTROL BLOCK (CNTL) OR COMPUTED FROM VALUES THEREIN. */
50 1   DECLARE ACTIVECHAN BYTE;
51 1   LASTACTIVECHAN BYTE;
52 1   FREEMBASEADD ADDRESS;
53 1   FREEMSIZE BYTE;
54 1   FREEMSIZEPLUSONE BYTE;
55 1   INSERTMASK ADDRESS;
56 1   FREEMMINADD ADDRESS;
57 1   FREEMMINADD ADDRESS;
58 1   MINIMUMADD ADDRESS;
59 1   MAXIMUMADD ADDRESS;
60 1   NUMFREEADD ADDRESS;
61 1   BLANKLINE(24) BYTE EXTERNAL;
62 1   J IMBUF(16) BYTE EXTERNAL;
63 1   END;
DECLARE ADDRESS;
DECLARE BASD CNTLSADD STRUCTU WU(
  LABELPTR ADDRESS,
  ACTIVEDIANPTR ADDRESS,  
  LASTACTIVECHAPTR ADDRESS,
  PARENTPTR ADDRESS,
  SIZE BYTE,
  MASK ADDRESS,
  KINPTR ADDRESS,
  MASHPTR ADDRESS,
  MANAGED(0,BMPTR ADDRESS));

DECLARE SIDIODELASTSHENABLE STRUCTURE(
  SMASK(7) BYTE,
  SMVALUE(11) BYTE) PUBLIC DATA(0.0.0.0.0.0.0.0.0.0.0.0.1);

DECLARE SIDIODELASTSTEPENABLE STRUCTURE(
  SMASK(7) BYTE,
  SMVALUE(11) BYTE) PUBLIC DATA(0.0.0.0.0.0.0.0.0.0.0.0.1);

DECLARE DIGITSHENABLE STRUCTURE(
  MODE BYTE,
  NUMCH BYTE,
  DELMCH BYTE,
  DELMASH ADDRESS,
  ECHROM BYTE,
  ECHOCOL BYTE,
  SHENABLE ADDRESS)
  DATA(NOCLARCHNLG:S,1.0.1,2,3,3,13,13,13,14,45,55,67,68,00,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
PL/M-80 Compiler COMM

ECHO XOR BYTE;
ECHO COL BYTE;
SHOWABLE ADDRESS;
INITIALIZECOM: PROCEDURE(CNTLAPTR) PUBLIC;
/* THIS PROCEDURE MUST BE INVOKED PRIOR TO ANY OF THE FOLLOWING
PROCEDURES. */
DECLARE CNTLAPTR ADDRESS:
CALL MOVE(9, INITIAL$STATPAGE$CB, STATPAGE$CB);
CNTLADD = CNTLAPTR;
CALL MOVE(1, CNTL, ACTIVE$CHNPTR, ACTIVE$CHAN);
CALL MOVE(1, CNTL, LASTACTIVE$CHNPTR, LASTACTIVE$CHAN);
FREQBASEADD = CNTL, FREQAPTR;
FREQSIZE = CNTL, SIZE;
FREQSIZEPLUS$SIZE = FREQSIZE + 1;
INSERTMASK = CNTL, MASK;
FREQMINADD = CNTL, MINAPTR;
FREQMAXADD = CNTL, MAXAPTR;
FREQICBAD$ADD = CNTL, FREQICBAD$APTR;
END:
FREQADD: PROCEDURE(CHAN) ADDRESS;
DECLARE CHAN BYTE;
/* COMPUTE OFFSET INTO FREQ ARRAY. */
RETURN(FREQBASEADD + CHAN * FREQSIZE);
END:
DISPLAY$FREQ: PROCEDURE(CHAN$II) PUBLIC;
DECLARE CHAN$II BYTE;
CALL CLEARTEMPBUF;
CALL MOVE(7, (CH- F- TEMPBUF);
CALL INSERT(FREQSIZE, FREQADD(CHAN$II), TEMPBUF(7), -INSERTMASK);
TEMPBUF(3) = CHAN$II + 30H;
CALL UPDATELINE(2, 6, 15, TEMPBUF);
END; /* OF DISPLAY$ACTIVE$CHAN */
UPDATE$FREQ: PROCEDURE(CHAN$II) BYTE;
DECLARE CHAN$II BYTE;
IF LIMIT$TEST(F, INBUF, FREQSIZE, FREQMINADD, FREQMAXADD)
THEN DO:
CALL MOVE(FREQSIZE, INBUF, FREQADD(CHAN$II));
RETURN TRUE;
END;
ELSE DO:
CALL ERROR(0);
RETURN FALSE;
END; /* OF UPDATE$FREQ */
LAST$FREQ: PROCEDURE;
04 2  ACTIVECHAN = LASTACTIVECHAN
05 2  CALL MOVE(1,LASTACTIVECHAN,ONTL,ACTIVECHANPTR)
06 2  CALL DISPLAYCHAN(REF(ACTIVECHAN))
07 2  END: /* OF LASTACTIVE */
DECLARE CHASESEL: TABLEAU(8) BYTE DATA;
  READ "CH: ";
  READ "SEL: ";
  READ "CH4SEL( ";

91 2 CALL CLEAR(21);
92 2 CALL DISPLAY(DIGITSEL);
93 2 CALL DISPLAY(CHASESEL,TABLEAU);
94 2 I = OFFH;
95 2 DO FOREVER
96 3 CALL READ(DIGITSEL);1
97 3 IF DATAENTERED THEN DO:
  99 4 I = DIMBUF(0) - 30H;
100 4 CALL DISPLAY(CHASESEL,FREQ(11));
101 4 END;
102 3 ELSE IF (SAY = ENTER) AND (I < OFFH) THEN DO:
104 4 LASTACTIVECHAN = ACTIVECHAN;
105 4 CALL MOVE(LASTACTIVECHAN+1,LASTACTIVECHAN+1);
106 4 ACTIVECHAN = I;
107 4 CALL MOVE(LASTACTIVECHAN+1,LASTACTIVECHAN+1);
108 4 RETURN;
109 4 END;
112 3 IF SAY = SUBMODE THEN RETURN;
113 3 IF SAY = CLEARASH THEN DO:
114 4 CALL UPDATELINE(2.7,FRAMESIZE+LINES,BLANKLINE);
115 4 I = OFFH;
116 4 END;
117 3 END /* OF DO FOREVER */
119 2 END /* OF CHASESEL */
SELECT
119 1 MANAFRE: PROEDURE(CHAN) PUBLIC;
120 2 DECLARE CHAN BYTE;
121 2 DECLARE MANAFRE(CTABLEAU(*)) BYTE DATA;
122 2 CALL CLEAR(2);
123 2 CALL DISPLAYDIGIT3;
124 2 CALL DISPLAY(CMANAFRE(CTABLEAU));
125 2 CALL DISPLAYMANAFRE(CCHAN);
126 2 DO FOREVER;
127 3 CALL READ(MANAFRE(CHANADD));
128 3 IF DATAENTERED AND (SW = ENTERSW) THEN DO;
129 4 IF UPDATERE(FRE(CHAN)) THEN RETURN;
130 4 END;
131 3 ELSE IF SW = LASTSW THEN CALL LASTFREQ;
132 3 IF SW = SUBMODESW THEN RETURN;
133 3 END;
134 2 END; // OF MANAFRE */

139 1 PRSTCHAN: PROCEDURE PUBLIC;
140 2 DECLARE PRSTCHAN(TABLEAU(*)) BYTE DATA;
141 2 DECLARE I BYTE;
142 2 CALL CLEAR(2);
143 2 CALL DISPLAYDIGIT3;
144 2 CALL DISPLAY(,PRSTCHAN(TABLEAU));
145 2 CALL READ(DIGITSEL1CH); 1
146 2 IF DATAENTERED THEN DO;
147 3 I = UNBUF(0) - 30H
148 3 DO WHILE I < 101
149 3 CALL DISPLAYCHANFREQ(II); 1
150 4 CALL READ(MANAFRE(CHANADD)); 1
151 4 IF DATAENTERED AND (SW = ENTERSW) THEN DO;
152 5 IF UPDATEFRE(II) THEN I = I + 1;
153 5 END;
154 4 ELSE DO;
155 5 IF SW = SUBMODESW THEN RETURN;
156 5 IF SW = LASTSW THEN CALL LASTFREQ;
157 5 IF SW = STEP3SW THEN I = I + 1;
158 5 END;
159 4 END; // OF PRSTCHAN */
160 2 END; // OF DATAENTERED */
161 2 END; // OF DATAENTERED */
162 2 END; // OF DATAENTERED */
BEGIN PROCESSPAGE CODE.

PAGEWNUM AND SHENABLEPTR ARE PASSED AS ARGUMENTS.

DECLARE EX(10), EX(2,1, 2, 0), EX(2,1, 2), EX(2,1, 2)

DECLARE PAGEWNUM BYTE.

DECLARE INERTF(12) BYTE DATA;

CALL CLEAR and TEMPHBUF;

CALL MOVE(3, OTH LABELIPTR, TEMPHBUF);

CALL MOVE(16, "CHW START", TEMPHBUF);

THB1 = PAGE + 31H;

CALL UPI DATE.(LINE(10, 4, 20), TEMPHBUF);

£ND /* OF GENERATESTATTHEADER */

CALL GENERATESTATTHEADER(PAGEWNUM);

DO FOREVER:

CALL GENERATESTATTHEADER(PAGEWNUM);

DO I = 0 TO 255;

CALL DISPLAYSTATLINE(11)

/* BEGIN PROCESSPAGE CODE. */
207 5 END;
208 4 CALL READI,STATPAGE(I0);)
209 4 IF SWINDEX CHAN$=XI0(PAGEINDEX) THEN DO;
211 5 CALL MAKER:INDEXI0(PAGEINDEX),LIST(,SWINDEX))
212 5 END;
213 4 ELSE IF SW = SUBLINDEX THEN RETURN;
217 4 IF SW = LASTSW THEN CALL LASTWRED;
219 4 END; /* OF DO FOREVER */
220 3 END; /* OF PROCESS PAGE */

/* BEGIN STATPAGE CODE */
221 2 DO FOREVER:
222 3 CALL PROCESSPAGE(0,...PAGEINDEXENABLE));
223 3 IF SW = SUBLINDEX THEN RETURN;
225 3 CALL PROCESSPAGE(1,...PAGEINDEXENABLE);)
226 3 IF SW = SUBLINDEX THEN RETURN;
228 3 END;
229 2 END; /* OF STATPAGE */

230 1 SNAP: PROCEDURE (AIPTR, BIPTR) PUBLIC;
231 2 DECLARE (AIPTR, BIPTR) ADDRESS;
232 2 DECLARE A BASED AIPTR BYTE;
233 2 B BASED BIPTR BYTE;
234 2 T BYTE;
235 2 A = B;
236 2 B = T;
237 1 END; /* OF SNAP */

MODULE INFORMATION:

CODE AREA SIZE = 0577H 1399D
VARIABLE AREA SIZE = 0023H 45D
MAXIMUM STACK SIZE = 0010H 16D
409 LINES READ
0 PROGRAM ERRORS
END OF PL/M-80 COMPILATION
/* WIF HAS BEEN CHANGED TO READ FE */

WIF: DO;

INLIST INCLUDE INCLUDE LISP.S;

4 1 INITIALIZECON: PROCEDURE (EXTENT, EXTERNAL);
5 2 DECLARE CONTPLTR ADDRESS;
6 2 END;
7 1 CHANGEL: PROCEDURE EXTERNAL;
8 2 END;
9 1 MANVALID: PROCEDURE (CHAN, EXTERNAL);
10 2 DECLARE CHAR BYTE;
11 END;
12 FETCCON: PROCEDURE EXTERNAL;
13 END;
14 #define: PROCEDURE (CHAN, EXTERNAL);

/* EXTERNAL */

DECLARE READ:

DECLARE UPDATE:

DECLARE SWAP:

DECLARE EXTERNAL:

DECLARE EXTERNAL VARIABLES */

DECLARE OFFFROMTEXT A BYTE EXTERNAL;
SGV BYTE EXTERNAL;
SHINDEX BYTE EXTERNAL;
SUBMODELASTSTATEBYTE ADDRESS EXTERNAL;
WIFSTATEBYTE ADDRESS EXTERNAL;
LASTSTATEBYTE ADDRESS EXTERNAL;
WIFSTATEBYTE ADDRESS EXTERNAL;
WIFSTATEBYTE ADDRESS EXTERNAL;
WIFSTATEBYTE ADDRESS EXTERNAL;
SUBMODEBYTE ADDRESS EXTERNAL;
DECLARE I BYTE;
DECLARE MINAFREQ(4) BYTE DATA('3000');
MAXAFREQ(4) BYTE DATA('7575')

INITVHF: PROCEDURE PUBLIC;
DO I = 0 TO 9;
   CALL MOVE(4,MINAFREQ,MAXAFREQ(I));
END;

SEND;
LASTVFACTIVECHN = 01;
VHFSTATUS = 01;
END;

VHFSUBMODE: PROCEDURE PUBLIC;
DECLARE MAXAFREDCNF STRUCTURE(
   MODE BYTE,
   NUMACH BYTE,
   DELNUMACH BYTE,
   DELNUMACH ADDRESS,
   EDCHROM BYTE,
   EDCHROM MS BYTE,
   SIMENABLE ADDRESS)
   DATA(MC444304D,4,'2.2000H.2.7.,SUBMENU-LASTSTEPENABLE)

DISPLAYSTATUS: PROCEDURE;
DECLARE MODETEXT(7) STRUCTURE(ch6) BYTE DATA(
   ' ', 'T'V', 'T/RG', 'HOM', 'RETRY', 'FAIL')
   DATA(1)
   CALL UPDATELINE(1,0.3.,OFFFROMTEXT(341));
   IF I = VHFSTATUS AND OFH
   THEN I = I + (SHR(VHFSTATUS,4) AND 11)
   CALL UPDATELINE(1,5.6.,MODETEXT(I));
END: # OF DISPLAYSTATUS #/

UPDATEMODE: PROCEDURE;
DECLARE SIMA LIST(6) BYTE DATA(3000,3200,3400,3500,3600,3400)
I = 0;
DO WHILE ((I = 0 OR SIMA LIST(I)) AND (I < 61)
   I = I + 1;
END;
IF I < 6 THEN I = I + 1;
VHFSTATUS = VHFSTATUS AND OFH + 1;
END;

ONOFF: PROCEDURE:
IF SIM = OFF THEN
   THEN ONOFF = VHFSTATUS AND OFH; # OFF #/
ELSE SIMA LIST(I) = VHFSTATUS OR DOH; # ON #/
SEND;
CALL DISPLAYS(341);
END;
DECLARE VHF SUBMODE TABLEAU(1) BYTE DATA(
  ROM4, 'FM','
  ROM5, 'CHAN MAN PRST STAT',
  ROM6, 'SEL', COL6, 'FREE', COL11, 'CHAN PAGE',
  ROM7, 'T/R', COL5, 'T/R MH', COL17, 'RE',
  COL6, 'GAP0', COL16, 'TRAN',
  ROM7, COL11, 'TEST', 0)

DECLARE VHF HSHENABLE STRUCTURE
  SHMASK(7) BYTE,
  SHVALUE(6) BYTE DATA(
    O:0,0:0000:0000:0000:0000:0000:0000:0000 /
    #* MASK DATA */

DECLARE VHF IC8 STRUCTURE
  NODE DATA,
  NNUM OF BYTE,
  DENL OF BYTE,
  DELPRT MASK ADDRESS,
  ECHOROW BYTE,
  ENCPRT BYTE,
  SHENABLE ADDRESS) DATA(
  SWITCH,0,0,0,0,0,0,0,0,0,0,VHF HSHENABLE)

DECLARE VHF FREEATL(3) BYTE AT(VHF SUBMODE TABLEAU(1));

DECLARE VHF LACT STRUCTURE
  LACT STRPTR ADDRESS,
  ACTIVECHAN PRT ADDRESS,
  LASTACTIVECHAN PRT ADDRESS,
  PREDPRT ADDRESS,
  PREDSIZE BYTE,
  INSERTS MASK ADDRESS,
  RNM PRT ADDRESS,
  MAXPRT ADDRESS,
  MAXFREQ(CRPRT ADDRESS) DATA(
  ,VHF FREEATL, ,VHF ACTIVECHAN, LASTACTIVECHAN .VHF FREE, 4,20000,
  ,MAXFREQ, ,MAXFREQ, ,MAXFREQ IC8)

DECLARE VHF FREQ STRUCTURE
  FREQ PTR ADDRESS,
  FREQ BYTE,
  MAXPRT ADDRESS,
  MAXFREQ(CRPRT ADDRESS) DATA(
  ,NEW FREE, ,NEW VHF, ,NEW MAXFREQ IC8)

DECLARE VHF NK STRUCTURE
  NK PRT ADDRESS,
  NK BYTE,
  MAXPRT ADDRESS,
  MAXFREQ(CRPRT ADDRESS) DATA(
  ,NEW NK, ,NEW VHF, ,NEW MAXFREQ IC8)
CALL INITIALIZE();
DO FOREVER;
    CALL CLEAR();
    CALL GRID();
    CALL DISPLAY();
    CALL DISPLAYS();
    CALL DISPLAYDATA();
    CALL READ();
    DO CASE SUBMODE;
    CALL CHANS();
    CALL MAINREAD();
    CALL INITCHAN();
    CALL STATUS;
    CALL UPDATED();
    CALL ONOFF();
    CALL SAMP();
    END / OF DO CASE /;
END / OF DO FOREVER /;
END / OF VHF SUBMODE /;

MODULE INFORMATION:
CODE AREA SIZE = 247H 5630
VARIABLE AREA SIZE = 001H 10
MAXIMUM STACK SIZE = 000AH 100
224 LINES READ
0 PROGRAM ERROR(S)

END OF PL/I-80 COMPILATION
TITLE('UNF')

UNF: DO
  WNDLIST INCLUDE(F1:COULIT.SRC)
  HANDLE:
  INITIALIZE:PROCEDURE(INAPTR) EXTERNAL
  DECLARE INAPTR ADDRESS
  CHANNEL: PROCEDURE EXTERNAL
  END
  MANIFEST: PROCEDURE(CHANNEL) EXTERNAL
  DECLARE CHANNEL BYTE
  END
  PITCHMARK: PROCEDURE EXTERNAL
  END
  STATPAGE: PROCEDURE EXTERNAL
  END
  DISPLAY:MANIFEST:PROCEDURE(CHANNEL) EXTERNAL
  DECLARE CHANNEL BYTE
  END
  CLEAR: PROCEDURE(STARTROW) EXTERNAL
  DECLARE STARTROW BYTE
  END
  GRID: PROCEDURE(POINTER) EXTERNAL
  DECLARE POINTER ADDRESS
  END
  DISPLAY:PROCEDURE(INAPTR) EXTERNAL
  DECLARE INAPTR ADDRESS
  END
  READ: PROCEDURE(INAPTR) EXTERNAL
  DECLARE INAPTR ADDRESS
  END
  UPDATE:LINE: PROCEDURE(ROM, COLUMN, X, Y, TEXTAPTR) EXTERNAL
  DECLARE (ROM, COLUMN, X, Y, TEXTAPTR ADDRESS
  END
  MAP: PROCEDURE(A, B) EXTERNAL
  DECLARE (A, B) ADDRESS
  END
  / * EXTERNAL VARIABLES */
  DECLARE
    OFFCOMPILE(6) BYTE EXTERNAL,
    BLANKLINE(24) BYTE EXTERNAL,
    SMY BYTE EXTERNAL,
    SMINDEX BYTE EXTERNAL,
    UNFACTIVATE(ROM BYTE EXTERNAL,
    LASTX(ROM, COLUMN, X, Y, TEXTAPTR ADDRESS EXTERNAL,
    LASTY(ROM, COLUMN, X, Y, TEXTAPTR EXTERNAL,
    LASTSTATUS BYTE EXTERNAL,
    UNFORMAT(10) STRUCTURE(10, DIGITS, 6) BYTE EXTERNAL,
    SUBINDEX(10) ADDRESS EXTERNAL.
PUNI-SO
COMPILER
30OCT79
PAGE 2

DECLARE I BYTE;
DECLARE MINFREQ(6) BYTE DATA('25000');
DECLARE MAXFREQ(6) BYTE DATA('399997');

INITUHF: PROCEDURE PUBLIC;
   DO I = 0 TO 9;
   CALL MOVE(6, MINFREQ, UHF+4FRED(I));
   END;
   UHF$STATUS = 01;
   UHF$ACTIVE$CHAN = 01;
   LASTUHF$ACTIVE$CHAN = 01
END;

DISPLAYUHF: PROCEDURE PUBLIC;
DECLARE MINAFRENS STRUCTURE
  MODE BYTE,
  NUMCH BYTE,
  DELMCH BYTE,
  DELMASK ADDRESS,
  ECHOMOD BYTE,
  ECHODCL BYTE,
  SHOWN$ADDRESS) DATA
  NODECLEARDIGIT(6, "*/1000H, 2.7, SUBMODE, LASTSTEPENABLE);

DISPLAYSTATUS: PROCEDURE;
DECLARE NODETEXT(5) STRUCTURE
  CH(IS) BYTE DATA
  '*/2'H, '*/4H', '*/6H', '*/GH', '*/9H';

CALL UPDATE$LINE(1, 0.3, OFF$CONTEXT(3+SHR(UHF$STATUS(5) AND 1)));
CALL UPDATE$LINE(1, 5.5, MODE$TEXT(UHF$STATUS AND OFH));
IF (UHF$STATUS AND 10H) > 0
  THEN CALL UPDATE$LINE(0, 5.3, ("SOL"));
ELSE CALL UPDATE$LINE(0, 5.3, BLANK$LINE);
END: /* OF DISPLAYSTATUS */

UPDATEMODE: PROCEDURE;
DECLARE SAV$LIST(5) BYTE DATA(301, 320, 340, 350, 360);
I = 01;
DO WHILE (SAV$LIST(I) AND (I < 51))
  I = I + 1;
END:
IF I < 4 THEN I = I + 1;
UHF$STATUS = (UHF$STATUS AND OFH) + 1;
END;

SOL$ON$OFF: PROCEDURE;
IF SAV = SOL$OFF$ON
  THEN UHF$STATUS = UHF$STATUS AND OFH;
ELSE UHF$STATUS = UHF$STATUS OR 10H;
DECLARE UNF#SUBMODE$TABLEAU(* BYTE DATA(
    "UNF",
    "CHAN MAN FRST STAT",
    "CHAN PAGE",
    "SEL COL6 \"FREQ\" COL11 \"CHAN PAGE\",
    "T/R COL6 \"T/R ADJ COL16 \"GARD",
    "COL6 \"GARD",
    "COL11 \"TEST",
    "COL6 \"ON COL11 \"TONE",
    "OFF \"COL6",
    "ON COL11 \"TONE",
    0)
DECLARE UNF#SMANGEABLE STRUCTURE(
    UNMASK(7) BYTE,
    "ON", 0,0,0,0,0,0,0,0,0,0
DECLARE UNF#SMABLE STRUCTURE(
    NODE BYTE,
    "ON", 0,0,0,0,0,0,0,0,0,0
DECLARE UNF#SMABLE STRUCTURE(
    NODE BYTE,
    "ON", 0,0,0,0,0,0,0,0,0,0
DECLARE UNF#SMABLE STRUCTURE(
    NODE BYTE,
    "ON", 0,0,0,0,0,0,0,0,0,0
DECLARE UNF#SMABLE STRUCTURE(
    NODE BYTE,
    "ON", 0,0,0,0,0,0,0,0,0,0
DECLARE UNF#SMABLE STRUCTURE(
    NODE BYTE,
    "ON", 0,0,0,0,0,0,0,0,0,0
DECLARE UNF#SMABLE STRUCTURE(
    NODE BYTE,
    "ON", 0,0,0,0,0,0,0,0,0,0
DECLARE UNF#SMABLE STRUCTURE(
    NODE BYTE,
    "ON", 0,0,0,0,0,0,0,0,0,0
DECLARE UNF#SMABLE STRUCTURE(
    NODE BYTE,
    "ON", 0,0,0,0,0,0,0,0,0,0
DECLARE UNF#SMABLE STRUCTURE(
    NODE BYTE,
    "ON", 0,0,0,0,0,0,0,0,0,0
DECLARE UNF#SMABLE STRUCTURE(
    NODE BYTE,
    "ON", 0,0,0,0,0,0,0,0,0,0
DECLARE UNF#SMABLE STRUCTURE(
    NODE BYTE,
    "ON", 0,0,0,0,0,0,0,0,0,0
DECLARE UNF#SMABLE STRUCTURE(
    NODE BYTE,
    "ON", 0,0,0,0,0,0,0,0,0,0
DECLARE UNF#SMABLE STRUCTURE(
    NODE BYTE,
    "ON", 0,0,0,0,0,0,0,0,0,0
DECLARE UNF#SMABLE STRUCTURE(
    NODE BYTE,
    "ON", 0,0,0,0,0,0,0,0,0,0
DECLARE UNF#SMABLE STRUCTURE(
    NODE BYTE,
    "ON", 0,0,0,0,0,0,0,0,0,0
DECLARE UNF#SMABLE STRUCTURE(
    NODE BYTE,
    "ON", 0,0,0,0,0,0,0,0,0,0
DECLARE UNF#SMABLE STRUCTURE(
    NODE BYTE,
    "ON", 0,0,0,0,0,0,0,0,0,0
DECLARE UNF#SMABLE STRUCTURE(
    NODE BYTE,
    "ON", 0,0,0,0,0,0,0,0,0,0
DECLARE UNF#SMABLE STRUCTURE(
    NODE BYTE,
    "ON", 0,0,0,0,0,0,0,0,0,0
DECLARE UNF#SMABLE STRUCTURE(
    NODE BYTE,
    "ON", 0,0,0,0,0,0,0,0,0,0
DECLARE UNF#SMABLE STRUCTURE(
    NODE BYTE,
    "ON", 0,0,0,0,0,0,0,0,0,0
DECLARE UNF#SMABLE STRUCTURE(
    NODE BYTE,
    "ON", 0,0,0,0,0,0,0,0,0,0
DECLARE UNF#SMABLE STRUCTURE(
    NODE BYTE,
    "ON", 0,0,0,0,0,0,0,0,0,0
DECLARE UNF#SMABLE STRUCTURE(
    NODE BYTE,
    "ON", 0,0,0,0,0,0,0,0,0,0
DECLARE UNF#SMABLE STRUCTURE(
    NODE BYTE,
    "ON", 0,0,0,0,0,0,0,0,0,0
DECLARE UNF#SMABLE STRUCTURE(
    NODE BYTE,
    "ON", 0,0,0,0,0,0,0,0,0,0
DECLARE UNF#SMABLE STRUCTURE(
    NODE BYTE,
    "ON", 0,0,0,0,0,0,0,0,0,0
DECLARE UNF#SMABLE STRUCTURE(
    NODE BYTE,
    "ON", 0,0,0,0,0,0,0,0,0,0
DECLARE UNF#SMABLE STRUCTURE(
    NODE BYTE,
    "ON", 0,0,0,0,0,0,0,0,0,0
DECLARE UNF#SMABLE STRUCTURE(
    NODE BYTE,
    "ON", 0,0,0,0,0,0,0,0,0,0
DECLARE UNF#SMABLE STRUCTURE(
    NODE BYTE,
    "ON", 0,0,0,0,0,0,0,0,0,0
DECLARE UNF#SMABLE STRUCTURE(
    NODE BYTE,
    "ON", 0,0,0,0,0,0,0,0,0,0
DECLARE UNF#SMABLE STRUCTURE(
    NODE BYTE,
    "ON", 0,0,0,0,0,0,0,0,0,0
DECLARE UNF#SMABLE STRUCTURE(
    NODE BYTE,
    "ON", 0,0,0,0,0,0,0,0,0,0
DECLARE UNF#SMABLE STRUCTURE(
    NODE BYTE,
    "ON", 0,0,0,0,0,0,0,0,0,0
DECLARE UNF#SMABLE STRUCTURE(
    NODE BYTE,
    "ON", 0,0,0,0,0,0,0,0,0,0
DECLARE UNF#SMABLE STRUCTURE(
    NODE BYTE,
    "ON", 0,0,0,0,0,0,0,0,0,0
DECLARE UNF#SMABLE STRUCTURE(
    NODE BYTE,
    "ON", 0,0,0,0,0,0,0,0,0,0
DECLARE UNF#SMABLE STRUCTURE(
    NODE BYTE,
    "ON", 0,0,0,0,0,0,0,0,0,0
DECLARE UNF#SMABLE STRUCTURE(
    NODE BYTE,
    "ON", 0,0,0,0,0,0,0,0,0,0
DECLARE UNF#SMABLE STRUCTURE(
    NODE BYTE,
    "ON", 0,0,0,0,0,0,0,0,0,0
DECLARE UNF#SMABLE STRUCTURE(
    NODE BYTE,
    "ON", 0,0,0,0,0,0,0,0,0,0
DECLARE UNF#SMABLE STRUCTURE(
    NODE BYTE,
    "ON", 0,0,0,0,0,0,0,0,0,0
DECLARE UNF#SMABLE STRUCTURE(
    NODE BYTE,
    "ON", 0,0,0,0,0,0,0,0,0,0
DECLARE UNF#SMABLE STRUCTURE(
    NODE BYTE,
    "ON", 0,0,0,0,0,0,0,0,0,0
DECLARE UNF#SMABLE STRUCTURE(
    NODE BYTE,
    "ON", 0,0,0,0,0,0,0,0,0,0
DECLARE UNF#SMABLE STRUCTURE(
    NODE BYTE,
    "ON", 0,0,0,0,0,0,0,0,0,0
DECLARE UNF#SMABLE STRUCTURE(
    NODE BYTE,
    "ON", 0,0,0,0,0,0,0,0,0,0
DECLARE UNF#SMABLE STRUCTURE(
    NODE BYTE,
    "ON", 0,0,0,0,0,0,0,0,0,0
DECLARE UNF#SMABLE STRUCTURE(
    NODE BYTE,
    "ON", 0,0,0,0,0,0,0,0,0,0
DECLARE UNF#SMABLE STRUCTURE(
    NODE BYTE,
    "ON", 0,0,0,0,0,0,0,0,0,0
DECLARE UNF#SMABLE STRUCTURE(
    NODE BYTE,
    "ON", 0,0,0,0,0,0,0,0,0,0
DECLARE UNF#SMABLE STRUCTURE(
    NODE BYTE,
    "ON", 0,0,0,0,0,0,0,0,0,0
DECLARE UNF#SMABLE STRUCTURE(
    NODE BYTE,
    "ON", 0,0,0,0,0,0,0,0,0,0
DECLARE UNF#SMABLE STRUCTURE(
    NODE BYTE,
    "ON", 0,0,0,0,0,0,0,0,0,0
DECLARE UNF#SMABLE STRUCTURE(
    NODE BYTE,
REJECT

96  2 CALL INITIALIZECOM(UHFCONT)
97  2 DO FOREVER
86  3 CALL CLEAR();
87  3 CALL GRID.(SUBMODE(0));
89  3 CALL DISPLAY(UHFSUBMODETABLEAU);
90  3 CALL DISPLAY(UHFCONT(UHFACTIVECHAN));
91  3 CALL DISPLAY(UHFACTIVECHAN);
92  3 CALL DISPLAYSTATUS;
93  3 CALL READ(UHFACTIVECHAN);
94  3 DO CASE SUBMODE;
95  4 CALL CHESS();
96  4 CALL MAN(H bảng(UHFACTIVECHAN));
97  4 CALL PRISTICHAN;
98  4 CALL STATIPAGE;
99  4 CALL UPDATINGMODE / T/R, T/RG, ADF, GARD #/
100  4 CALL SELECTOFF();
101  4 CALL TESTSTONE();
102  4 CALL ONOFF();
103  4 CALL SNMP(UHFACTIVECHAN, LAST(UHFACTIVECHAN)); / LAST #/
104  4 END1 / DO CASE #/
105  3 END1 /# OF DO FOREVER #/
106  2 END1 /# OF UHF #/
107  1 END1 /# OF UHF #/

MODULE INFORMATION:

CODE AREA SIZE  = 0204H  6440
VARIABLE AREA SIZE  = 0001H  10
MAXIMUM STACK SIZE = 0008H  80
239 LINES READ
0 PROGRAM ERROR(S)

END OF PL/M-80 COMPILATION
TITLE('ADF')

1 ADF: DO;

2 INCLUDE(('File:COLLIT.SRC'))

3 INITIALIZECOM: PROCEDURE(CNTL#PTR) EXTERNAL;
4 DECLARE CNTL#PTR ADDRESS;
5 END;
6 CHANSSEL: PROCEDURE EXTERNAL;
7 END;
8 MAINTREY: PROCEDURE(CHAN) EXTERNAL;
9 DECLARE CHAN BYTE;
10 END;
11 POSTCHS: PROCEDURE EXTERNAL;
12 END;
13 STATPAGE: PROCEDURE EXTERNAL;
14 END;
15 DISPLAY: PROCEDURE(CHAN) EXTERNAL;
16 DECLARE CHAN BYTE;
17 END;
18 CLEAR: PROCEDURE(STARDROM) EXTERNAL;
19 DECLARE STARDROM BYTE;
20 END;
21 CLEARSTEPBUF: PROCEDURE EXTERNAL;
22 END;
23 GRID: PROCEDURE(POINTER) EXTERNAL;
24 DECLARE POINTER ADDRESS;
25 END;
26 DISPLAY: PROCEDURE(IN#PTR) EXTERNAL;
27 DECLARE IN#PTR ADDRESS;
28 END;
29 READ: PROCEDURE(IN#PTR) EXTERNAL;
30 DECLARE IN#PTR ADDRESS;
31 END;
32 UPDATELM: PROCEDURE(ROM,COL,NCHAR,TEXT#PTR) EXTERNAL;
33 DECLARE (ROM,COL,NCHAR) BYTE, TEXT#PTR ADDRESS;
34 END;
35 SWAP: PROCEDURE(A, B) EXTERNAL;
36 DECLARE (A, B) ADDRESS;
37 END;
38 /

/* EXTERNAL VARIABLES */

DECLARE

OFFPOINTEXT(16) BYTE EXTERNAL;
BLANKLINE(24) BYTE EXTERNAL;
TEMPBUF(24) BYTE EXTERNAL;
SUBMODESEL#STEPENABLE ADDRESS EXTERNAL;
SIV BYTE EXTERNAL;
SMINDEX BYTE EXTERNAL;
MAFFACTIVE#CHAN BYTE EXTERNAL;
LASTMAFFACTIVE#CHAN BYTE EXTERNAL;
MAFFSTATUS BYTE EXTERNAL;
MAFFPREA(10) STRUCTURE:DIGITS(4) BYTE) EXTERNAL;
SUBPROGRAM ADDRESS EXTERNAL;

40 1 DECLARE CMSW LITERALLY '420';
41 1 DECLARE I BYTE;
42 1 DECLARE MINFRED4(4) BYTE DATA('0190');
    MAXFRED(4) BYTE DATA('1750');
43 1 INITIAL: PROCEDURE PUBLIC;
44 2 DO I = 0 TO 9;
45 3 CALL MOVE(4,MINFRED..ADFRED(1));
46 3 END;
47 2 ADFSTATUS = 01;
48 2 ADFACTIVECHAN = 01;
49 2 LASTADFACTIVECHAN = 01;
50 2 END;

51 1 ADFSUBMODE: PROCEDURE PUBLIC;
52 2 DECLARE MNHFRED4CIO STRUCTURE(
    NODE BYTE,
    NUMCH BYTE,
    DELMON BYTE,
    DEMASK ADDRESS,
    EONCHitY;,
    EDONCOL BYTE,
    SIMENABLE ADDRESS) DATA(
    MNHFRED4CIO,1,0,2,7,,SUBMODE,LASTSTEPENABLE);

53 2 DECLARE MODETESTLABEL(3) STRUCTURE(TEXT(4) BYTE)
    DATA('0.5.4..MODETESTLABEL(ADFSTATUS AND 3)');
    MODELABEL(3) STRUCTURE(TEXT(8) BYTE)
    DATA('RCVR AUTO ADPM BIT')
54 3 CALL UPDATE LINE(0,5,4..MODETESTLABEL(ADFSTATUS AND 3));
55 3 CALL CLEAR routine:;
56 3 IF (ADFSTATUS AND 20H) > 0 THEN I = 3;
57 3 ELSE I = 01;
58 3 CALL MOVE(3..OFFCHANTEXT(1),.TERMBUF);
59 3 CALL MOVE(8..MODELABEL(5H.(ADFSTATUS AND 2) AND 31,.TERMBUF(5));
60 3 CALL UPDATE LINE(1,013,.TERMBUF);
61 3 ENDI /* OF DISPLAY STATUS */

62 2 UPDATEMODE: PROCEDURE;
63 3 DECLARE SWLIST(4) BYTE DATA(200,310,220,340),
    VALIST(4) BYTE DATA(0.0,4.8);
64 3 I = 01;
65 3 DO WHILE (SW = SWLIST(I)) AND (I < 4);
66 4 I = I + 11;
67 4 ENDI;
68 3 ADFSTATUS = (ADFSTATUS AND (34H) + VALIST(1));
PL/M-80 COMPILER ADF

71 3 END;

72 2 UPDATEPROCEDURE:
73 3 ADFSSTATUS = ADFSTATUS AND OFCH;
74 3 IF SMV = CWSH THEN ADFSTATUS = ADFSTATUS + 1;
75 3 ENI;

77 2 TEST: PROCEDURE:
78 3 ADFSTATUS = (ADFSTATUS AND OFCH) + 2;
79 3 ENI;

80 2 ON/OFF: PROCEDURE:
81 3 IF SMV = OFFSH
82 3 THEN ADFSTATUS = ADFSTATUS AND OFCH /* OFF */
83 3 ELSE ADFSTATUS = ADFSTATUS OR OFCH /* ON */
84 3 ENI;

85 2 DECLARE ADF$SUBMODETABLEAU(*) BYTE DATA
   ROAD,'ADF';
   ROM,'CHN MAN',COL1,'PST STAT';
   ROM,'SEL',COL1,'FREQ',COL1,'CHAN PAGE';
   ROMS,'CVR AUTO MAN';
   ROM6,'ADF',COL1,'ADF';
   ROM6,'VCE',COL7,'CM',COL1,'TEST';01;

86 2 DECLARE ADF$SHENABLE STRUCTURE:
   SHMASK(7) BYTE,
   SHVALUE(9) BYTE DATA
   0,0,0,0,0,0,0,0,0,
   00h,12h,33h,44h,55h,66h,77h,88h;

87 2 DECLARE ADF$ICB STRUCTURE:
   MODE BYTE,
   NORMIC BYTE,
   DELIMIC BYTE,
   DELIMASK ADDRESS,
   ECHOMON BYTE,
   ECHONCOL BYTE,
   SHENABLE ADDRESS) DATA
   SWITCH,0,0,0,0,0,0,0,0,0,0

88 2 DECLARE ADF$LABEL(3) BYTE AT.ADF$SUBMODETABLEAU(1111)
89 2 DECLARE ADF$ICNTL STRUCTURE:
   LABELPTR ADDRESS,
   ACTIVECHANPTR ADDRESS,
   LASTACTIVECHANPTR ADDRESS,
   FREQUENCY ADDRESS,
   FREQUENCY BYTE,
   INSERTMASK ADDRESS,
   MINMPTR ADDRESS,
   MAXMPTR ADDRESS,
   MINFREQ(4)(MPTR ADDRESS) DATA
   .ADF$LABEL, .ADF$ACTIVECHAN, .LASTACTIVECHAN, .ADF$FREQ, 4.0,
   ,MINFREQ, .MAXFREQ, .MINFREQ(0)
REJECT
90  CALL INITIALIZECOM(.ADF$CTRL);
91  DO FOREVER
92  CALL CLEAR();
93  CALL GRID(.SUBGENERIC);
94  CALL DISPLAY(.ADF$SUBGENETABLEAU);
95  CALL DISPLAY.MAP(.ADF$ACTIVECHAN);
96  CALL DISPLAY$STATUS;
97  CALL READ(.ADF$CTRL);
98  DO CASE SHUTDOWN;
99  CALL CHANGE;
100  CALL UPDATE(ACTIVECHAN);
101  CALL PASTECHAN;
102  CALL STATUSPAGE;
103  CALL UPDATEMODE1;
104  CALL UPDATEMODE2;
105  CALL TEST;
106  CALL ON/OFF;
107  CALL SNAP(.ADF$ACTIVECHAN, LAST$ACTIVECHAN); /* LAST */
108  END; /* OF DO CASE */
109  END; /* OF DO FOREVER */
110  END; /* OF ADF$SUBMODE */
111  END; /* OF ADF */

MODULE INFORMATION:

CODE AREA SIZE = 028BH  651B
VARIABLE AREA SIZE = 0001H  1D
MAXIMUM STACK SIZE = 0008H  6B
244 LINES READ
0 PROGRAM ERROR(S)

END OF PLAN-80 COMPILED
/* TITLE('CNV') */
1
2
3
4 1 DECLARE CNVTEXT(1) BYTE EXTERNAL;
5 2 DECLARE BLANKLINE(24) BYTE EXTERNAL;
6 2 DECLARE TEMPHBUF(24) BYTE EXTERNAL;
7 2 DECLARE SW_BYTE external;
8 2 DECLARE SUBINDEXBYTE ADDRESS EXTERNAL;
9 2 DECLARE LASTACTIVEBYTE EXTERNAL;
10 2 DECLARE CNVACTIVEBYTE EXTERNAL;
11 2 DECLARE CNVSTATUS BYTE EXTERNAL;
12 2 DECLARE CNVBYTE EXTERNAL;
13 2 DECLARE CNVTEXT(1) BYTE EXTERNAL;
14 2 DECLARE BLANKLINE(24) BYTE EXTERNAL;
15 2 DECLARE TEMPHBUF(24) BYTE EXTERNAL;
16 2 DECLARE SW_BYTE external;
17 2 DECLARE SUBINDEXBYTE ADDRESS EXTERNAL;
18 2 DECLARE LASTACTIVEBYTE EXTERNAL;
19 2 DECLARE CNVACTIVEBYTE EXTERNAL;
20 2 DECLARE CNVSTATUS BYTE EXTERNAL;
21 2 DECLARE CNVBYTE EXTERNAL;
22 2 DECLARE CNVTEXT(1) BYTE EXTERNAL;
23 2 DECLARE BLANKLINE(24) BYTE EXTERNAL;
24 2 DECLARE TEMPHBUF(24) BYTE EXTERNAL;
25 2 DECLARE SW_BYTE external;
26 2 DECLARE SUBINDEXBYTE ADDRESS EXTERNAL;
27 2 DECLARE LASTACTIVEBYTE EXTERNAL;
28 2 DECLARE CNVACTIVEBYTE EXTERNAL;
29 2 DECLARE CNVSTATUS BYTE EXTERNAL;
30 2 DECLARE CNVBYTE EXTERNAL;
31 2 DECLARE CNVTEXT(1) BYTE EXTERNAL;
32 2 DECLARE BLANKLINE(24) BYTE EXTERNAL;
33 2 DECLARE TEMPHBUF(24) BYTE EXTERNAL;
34 2 DECLARE SW_BYTE external;
35 2 DECLARE SUBINDEXBYTE ADDRESS EXTERNAL;
36 2 DECLARE LASTACTIVEBYTE EXTERNAL;
37 2 DECLARE CNVACTIVEBYTE EXTERNAL;
38 2 DECLARE CNVSTATUS BYTE EXTERNAL;
39 2 DECLARE CNVBYTE EXTERNAL;
OVM#1: BYTE EXTERNAL,
OVM#2: (H) STRUCTURE (DIGITS5 BYTE) EXTERNAL,
SVENDER AT ADDRESS EXTERNAL:

40 1 DECLARE VOLUME#5 LITERALLY '325';
MB#HIGH#5 LITERALLY '350';
41 1 DECLARE I BYTE;
42 1 DECLARE MINFRED(5) BYTE DATA('10000');
MB#FRED(5) BYTE DATA('11270');
43 1 DECLARE VOLUME#APTR ADDRESS;
VOL# BASED VOLUME#APTR BYTE;
44 1 INIT#H#5: PROCEDURE PUBLIC;
45 2 DO I = 0 TO 9;
46 3 CALL MOVE(5, MINFRED...OVM#FRED(1));
47 3 END;
48 2 OVM#STATUS = 0;
49 2 OVM#ST#ACTIVE#CH#N = 01;
50 2 LASTOMV#ST#ACTIVE#CH#N = 01;
51 2 VOLUME#APTR = '225#H#5#VOL#';
52 2 OVM##H#5#VOL = 00;
53 2 OVM#ST#5#VOL = 00;
54 2 END;

55 1 OVM#ST#5#H#5: PROCEDURE PUBLIC;
56 2 DECLARE MINFRED(5) STRUCTURE(5
NODE BYTE,
NUM#B#5 BYTE,
DEL#5#N#5 BYTE,
DEL#5#N#5#MASK ADDRESS;
ECHO#5#H#5 BYTE,
ECHO#5#N#5 BYTE,
M#5#E#5#N#5#H#5 ADDRESS) DATA(
NUM#CLE#5#H#5#DIGITS5.5'/1000H.2.7,SVENDER#LAST#STEP#H#5#E#5#');

57 2 VOLUME#AJUST: PROCEDURE:
58 3 IF SW = VOLUME#5 THEN VOL = VOL + 1;
59 3 ELSE VOL = VOL - 1;
60 3 END: /* OF VOLUME#AJUST */
61 2 MB#HIGH#5#LOW: PROCEDURE:
62 3 IF SW = MB#HIGH#5#5 THEN OVM#ST#5 = OVM#ST#5# AND 11;
63 3 ELSE OVM#ST#5 = OVM#ST#5# AND 0F#5;
64 3 END: /* OF MB#HIGH#5#LOW */
65 2 OVM#OFF: PROCEDURE:
66 3 IF SW = OVM#OFF
THEN OVM#ST#5 = OVM#ST#5# AND 00#5:
/* OFF */

70 3 ELSE C@NIVSTATUS = C@NIVSTATUS OR 20H;  // ON */
71 3 END;

72 3 DISPLAYSTATUS; PROCEDURE;
73 3 DECLARE I BYTE;
74 3 DECLARE MBHILITEXT(2) STRUCTURE(CH(5) BYTE) DATA("MB-LO","MB-HI");
75 3 IF (C@NIVSTATUS AND 20H) > 0 THEN I = 3;
76 3 ELSE I = 0;
77 3 CALL UPDATETE.getTextline(1,0,3,...OFFRONTTEXT(I));;
78 3 CALL UPDATETE.getTextline(1,19,5,...MBHILITEXT(C@NIVSTATUS AND I));
79 3 END;

80 2 DECLARE CMINVALUETABLEAU(4) BYTE DATA:

82 2 DECLARE CMIVALUABLEABLEU(4) BYTE DATA:
83 2 SIMASK(7) BYTE,
84 2 SIMVALUE(8) BYTE DATA(
85 2 0,0,0,0,0,0,0,0,CH1,CH1,CH1,CH1,
86 2 CH1,CH1,CH1,CH1,CH1,CH1,CH1,CH1,CH1,
87 2 CH1,CH1,CH1,CH1,CH1,CH1,CH1,CH1,CH1,
88 2 CH1,CH1,CH1,CH1,CH1,CH1,CH1,CH1,CH1,
89 2 CH1,CH1,CH1,CH1,CH1,CH1,CH1,CH1,CH1,
90 2 CH1,CH1,CH1,CH1,CH1,CH1,CH1,CH1,CH1,
91 2 CH1,CH1,CH1,CH1,CH1,CH1,CH1,CH1,CH1,
92 2 CH1,CH1,CH1,CH1,CH1,CH1,CH1,CH1,CH1,
93 2 CH1,CH1,CH1,CH1,CH1,CH1,CH1,CH1,CH1,
94 2 CH1,CH1,CH1,CH1,CH1,CH1,CH1,CH1,CH1,
95 2 CH1,CH1,CH1,CH1,CH1,CH1,CH1,CH1,CH1,
96 2 CH1,CH1,CH1,CH1,CH1,CH1,CH1,CH1,CH1,
97 2 CH1,CH1,CH1,CH1,CH1,CH1,CH1,CH1,CH1,
98 2 CH1,CH1,CH1,CH1,CH1,CH1,CH1,CH1,CH1,
99 2 CH1,CH1,CH1,CH1,CH1,CH1,CH1,CH1,CH1,
100 2 CH1,CH1,CH1,CH1,CH1,CH1,CH1,CH1,CH1,
101 2 CH1,CH1,CH1,CH1,CH1,CH1,CH1,CH1,CH1,
102 2 CH1,CH1,CH1,CH1,CH1,CH1,CH1,CH1,CH1,
103 2 CH1,CH1,CH1,CH1,CH1,CH1,CH1,CH1,CH1,
104 2 CH1,CH1,CH1,CH1,CH1,CH1,CH1,CH1,CH1,
105 2 CH1,CH1,CH1,CH1,CH1,CH1,CH1,CH1,CH1,
106 2 CH1,CH1,CH1,CH1,CH1,CH1,CH1,CH1,CH1,
107 2 CH1,CH1,CH1,CH1,CH1,CH1,CH1,CH1,CH1,
108 2 CH1,CH1,CH1,CH1,CH1,CH1,CH1,CH1,CH1,
109 2 CH1,CH1,CH1,CH1,CH1,CH1,CH1,CH1,CH1,
110 2 CH1,CH1,CH1,CH1,CH1,CH1,CH1,CH1,CH1,
111 2 CH1,CH1,CH1,CH1,CH1,CH1,CH1,CH1,CH1,
112 2 CH1,CH1,CH1,CH1,CH1,CH1,CH1,CH1,CH1,
113 2 CH1,CH1,CH1,CH1,CH1,CH1,CH1,CH1,CH1,
114 2 CH1,CH1,CH1,CH1,CH1,CH1,CH1,CH1,CH1,
MODIFIED PAGE 4

模块信息:

代码区域大小 = 023FH 5750
变量区域大小 = 0004H 4D
最大堆栈大小 = 0008H 6D
241 行读取
0 个程序错误

END OF PL/10 COMPILATION
TITLE("CROUT")

DECLE

VIRFACTIVECHCK BYTE PUBLIC;
LASTINFRACTIVECHCK BYTE PUBLIC;
INFSTATUS BYTE PUBLIC;
INFPOSFD(10) STRUCTURE(DIGITS(4) BYTE PUBLIC)

DECLARE INFFACTIVECHCK BYTE PUBLIC;
LASTINFRACTIVECHCK BYTE PUBLIC;
INFSTATUS BYTE PUBLIC;
INFPOSFD(10) STRUCTURE(DIGITS(4) BYTE PUBLIC)

DECLARE INFFACTIVECHCK BYTE PUBLIC;
LASTINFRACTIVECHCK BYTE PUBLIC;
INFSTATUS BYTE PUBLIC;
INFPOSFD(10) STRUCTURE(DIGITS(4) BYTE PUBLIC)

DECLARE INFFACTIVECHCK BYTE PUBLIC;
LASTINFRACTIVECHCK BYTE PUBLIC;
INFSTATUS BYTE PUBLIC;
INFPOSFD(10) STRUCTURE(DIGITS(4) BYTE PUBLIC)

DECLARE INFFACTIVECHCK BYTE PUBLIC;
LASTINFRACTIVECHCK BYTE PUBLIC;
INFSTATUS BYTE PUBLIC;
INFPOSFD(10) STRUCTURE(DIGITS(4) BYTE PUBLIC)

DECLARE INFFACTIVECHCK BYTE PUBLIC;
INFSCREEN BYTE PUBLIC;
INFSCREENPOS BYTE PUBLIC;
/* DEF INITIALIZATION REQUIRES THAT THE FOLLOWING ITEMS BE CONTIGUOUS. */
INFSCREENCODE(2) BYTE PUBLIC;
INFSCREENCODE(4) BYTE PUBLIC;
INFSCREENCODE(4) BYTE PUBLIC

END /* DEF */

PUBLIC INFORMATION:

CODE AREA SIZE = 00000 0D
VARIABLE AREA SIZE = 00070 217D
MAXIMUM STACK SIZE = 00000 0D
34 LINES READ
0 PROGRAM ERROR(S)

END OF PL/M-80 Compilation
TITLE('FON'TDA')
FONTDA: 0D1
1 2 1 DECLARE CFON'TDATA(220) BYTE PUBLIC DATA
   0.0.0.0.0, */ SP */
   0.0.3FH.0.0, */ ! */
   6.7H.6.0, */ NEG SYN */
   0.7FH.3EH.1CH.8, */ SOLID RIGHT ARROW */
   0.6CH.3BH.7FH.2CH, */ $ */
   0.6CH.130H.6FH, */ Z */
   0.3AH.4FH.5FH.2FH, */ % */
   0.6.0.7.0, */ . */
   0.0.1CH.22H.41H, */ ( */
   0.0.41H.22H.1CH, */ ) */
   2DH.1CH.32H.1CH.2DH, */ = */
   6.6.3EH.9.8, */ + */
   6.6CH.30H.6.0, */ - */
   6.6.6.6.6, */ */
   6.68.60H.0.0, */ 9 */
   60H.19H.6.4.3, */ */
   3EH.51H.49H.45H.3EH, */ 0 */
   46H.42H.7FH.40H.40H, */ 1 */
   72H.49H.49H.49H.46H, */ 2 */
   22H.41H.49H.49H.36H, */ 3 */
   1B1.16H.12H.7FH.1BH, */ 4 */
   27H.43H.45H.45H.39H, */ 5 */
   30H.66H.49H.49H.30H, */ 6 */
   41H.13H.9.5.3, */ 7 */
   34H.49H.49H.49H.34H, */ 8 */
   44H.49H.49H.29H.13H, */ 9 */
   0.36H.36H.0.0, */ 0 */
   0.46H.36H.0.0, */ 1 */
   0.18H.22H.41H.0, */ < */
   1EH.18H.1EH.1EH.1EH, */ > */
   0.41H.22H.14H.8, */ > */
   2.1.51H.8.6, */ ? */
   32H.41H.32H.52H.2EH, */ 0 */
   70H.12H.11H.12H.7CH, */ A */
   7FH.49H.49H.49H.34H, */ B */
   33H.41H.49H.14H.23H, */ C */
   7FH.41H.41H.23H.1CH, */ D */
   7FH.49H.49H.49H.41H, */ E */
   7FH.9.9.9.1, */ F */
   32H.41H.31H.51H.72H, */ 0 */
   7FH.6.8.8.7FH, */ N */
   0.41H.7FH.41H.0, */ 1 */
   20H.40H.41H.3FH.1, */ J */
   7FH.8.14H.22H.41H, */ K */
   7FH.49H.49H.49H.40H, */ L */
   7FH.2.0CH.2.7FH, */ M */
   7FH.6.00H.18H.7FH, */ N */
   32H.41H.41H.3FH, */ 0 */
TITLE('IFF')

IFF: DO:

END

CLEARLINE: PROCEDURE(LINE) EXTERNAL

DECLARE LINE BYTE

END

CLEAR: PROCEDURE(STARTROW) EXTERNAL

DECLARE STARTROW BYTE

END

CLEARNYPHBUF: PROCEDURE EXTERNAL

END

GRID: PROCEDURE(POINTER) EXTERNAL

DECLARE POINTER ADDRESS

END

DISPLAY: PROCEDURE(INPTR) EXTERNAL

DECLARE INPTR ADDRESS

END

READ: PROCEDURE(INCHPTR) EXTERNAL

DECLARE INCHPTR ADDRESS

END

LIMITTEST: PROCEDURE(INCHPTR,NUMDIOITS,INMPTR,MAXMPTR) BYTE EXTERNAL

DECLARE NUMDIOITS BYTE, INCHPTR,INMPTR,MAXMPTR ADDRESS

END

UPDATE: LINE: PROCEDURE(ROM,COL,NO,CHAR,TEXTPTR) EXTERNAL

DECLARE (ROM,COL,NO,CHAR,TEXTPTR ADDRESS

END

READTEST: PROCEDURE

DECLARE NORMSH LITERALLY '32P'

*/

EXTERNAL VARIABLES */

DECLARE

OFFONTEXT(2) STRUCTURE(CHAR(3) BYTE) EXTERNAL,
BLANKLINE(24) BYTE EXTERNAL,
TEMPBUFF(24) BYTE EXTERNAL,
SAV BYTE EXTERNAL,
SHINDX BYTE EXTERNAL,
NUMBUF(16) BYTE EXTERNAL,
DATAENTERED BYTE EXTERNAL,
DIGITWORD ADDR ADDRESS EXTERNAL,
ACTUALHISTORY ADDR ADDRESS EXTERNAL,
ERRSTATUS BYTE EXTERNAL,
IFMSTAT(2) BYTE EXTERNAL,
IFMCHCODE(4) BYTE EXTERNAL,
IFMCHCODE(4) BYTE EXTERNAL,
IFMCHMODE BYTE EXTERNAL,
DECLARE GROUPEABLE STRUCTURE:
SIZE=7, BYTE.
SIZEVALUE[2] BYTE DATA:
O.0.0.0.0, 186.20H. /* OFF, ON, SUBNODE */
O14, 20H11
DECLARE GROUPEABLE STRUCTURE:
SIZE=7, BYTE DATA(O.0.0.0.0-0.O000) /* VALUE NOT IMPORTANT */
DECLARE GROUPEABLE STRUCTURE:
ITEM BYTE,
ITEMBYTE BYTE,
ITEMBYTE BYTE,
ITEMMASK ADDRESS,
ECOMIN BYTE,
ECOMIL BYTE,
SIZEVALUE ADDRESS) DATA(SWITCH-O.O.O.O.O..GROUPEABLE)

INITIAL: PROCEDURE PUBLIC:
/* INITIALIZATION OF THE CODE DATA BELOW REQUIRES THAT THE DATA BE
LOCATED CONTIGUOUSLY AND THAT HICODE APPEAR FIRST IN THE LIST
GAVE THE LOWEST ADDRESS. */
IFSTATUS = 01
IFFHANODE = 01
IFFHANPOSITION = 01
IFFHANCODE(0) = 01
CALL NODE((+, IFFHANCODE, IFFHANCODE + 1))
END

DIRECT: PROCEDURE(VALUE, BITLOC, MASK) BYTE
DECLARE(VALUE, BITLOC, MASK) BYTE
RETURN(SHR(VALUE, BITLOC) AND MASK)
END

DECLARE 1 BYTE

IFFIONOFF: PROCEDURE:

IFFOFF: PROCEDURE:
CALL DISPLAY, (ROA, "OFF GROUPE", 0111
CALL READI, (GROUPE))
IF SUM = SUBCODE THEN RETURN
CALL CLEARLINE(2)
IFFSTATUS = IFFSTATUS AND 00FH
END // OF IFFOFF */

IF SUM = OFF THEN CALL IFFOFF
ELSE IFFSTATUS = IFFSTATUS OR 20H
END
43 1 NORMSTBY: PROCEDURE;
44 2 IF Sind = NORM34 THEN IFFSTATUS = IFFSTATUS AND OSFM;
45 2 ELSE IFFSTATUS = IFFSTATUS OR OSFM;
46 2 END;
47 1 TEST: PROCEDURE;
48 2 DECLARE TESTTABLEAU(+) BYTE DATA;
49 2 ROM, COL.1 = "IFF TEST",
50 2 ROM, +1 = "M-1", COL.6 = "M-2", COL.11 = "M-34",
51 2 ROM, +2 = "M-4",
52 2 ROM, +3 = "M-2", COL.01;
53 2 DECLARE TESTTABLE: STRUCTURE;
54 2 SBMASK(7) BYTE,
55 2 SBVALUE(3) BYTE DATA(0, 0, 000, 000, 000;
56 2 CS:03H-80H1;
57 2 DECLARE TESTTAB STRUCTURE;
58 2 NODE BYTE,
59 2 NUMCH BYTE,
60 2 DLMCH BYTE,
61 2 DLDMASK ADDRESS;
62 2 ECHWVRN BYTE;
63 2 ECHOL COL BYTE;
64 2 SHAINABLE ADDRESS) DATA(SWITCH, 0, 0, 0, 0, 0, 0, 0, TESTABLE13;
65 2 CALL CLEAR(0);
66 2 IF IFFWINDO THEN TEMPHB(0) = "B";
67 2 ELSE TEMPHB(0) = "W";
68 2 CALL UPDATE LINE(6, 4, 1, TEMPHB);
69 2 CALL CLEARTEMPBUF;
70 2 CALL MOVE(0, TESTTABLEAU + 2, TEMPHB);
71 2 CALL MOVE(6, ("PASSED"), TEMPHB(11));
72 2 CALL DISPLAY(TESTTABLEAU);
73 2 CALL GRID(1, SUBNODEGRID1);
74 2 DO FOREVER;
75 3 CALL READ(1, TESTC1);
76 3 IF Sind = SUBNODE THEN RETURN;
77 3 IF SIND = 4 THEN TEMPHB(9) = "C";
78 3 ELSE TEMPHB(9) = SIND + 30H;
79 3 CALL UPDATE LINE(2, 0, 17, TEMPHB);
80 3 END /% OF DO FOREVER 4/;
81 2 END /% OF TEST 4/;
82 1 DISPLAY/ SUICIDES STATUS LINE: PROCEDURE;
83 2 DECLARE BUF PTR ADDRESS;
84 2 BUF BASED BUF PTR(11) BYTE;
85 2 MASK BYTE;
86 2 DECLARE NORMSTBY TEXT(2) STRUCTURE(CHAR(4) BYTE) DATA("NORMSTBY11;
87 2 CALL CLEARLINE(11);
88 2 CALL UPDATE LINE(0, 10, 4, NORMSTBYTEXT(INDEX(1, IFFSTATUS, 4, 11), IFFSTATUS, 4, 11));
89 2 BUF PTR + TEMPHB + 4;
90 2 CALL CLEAR TEMP BUF1;
CALL MOVE(3,OFFزمارة(ÍNDEX(IFR* status,5,1))..TEMPBUF);

DO I = 0 TO 4:
   IF MASK THEN DO! /* INSERT HCI> TEXT FOR 'ON' UNITS */
      BUF(0) = 'A'
      BUF(1) = I + 1\HN
   END;
   BUF PTR = BUF PTR + 4;
   MASK = SHR(MASK,1));
   END /* OF DO I */
   IF (IFR* status AND 4) = 0 THEN TEMPBUF(14) = 'A'
   IF (IFR* status AND 10) = 0 THEN DO;
      IF IFR* noDEN THEN TEMPBUF(18) = 'B'
      ELSE TEMPBUF(18) = 'A'
   END;
   IF (IFR* status AND 10) = 0 THEN TEMPBUF(21) = 'C'
   CALL UPDATE\LINE(1,0;22,TEMPBUF);
   END /* OF DISPLAYSUBRDOTESTATUS\LINE */
select
122 1  
123 2  DECLARE COUNTABLE(1) BYTE DATA(  
        124 2  ROCL, 'N', 4, 'SELECT',  
        ROCL, 'C', 17, 'A', 10, 'COL1', 'B', 14, 'COL16', 'HOLD',  
        ROCL, 'C', 17, 'A', 10, 'COL16', 'OUT',  
        ROCL, 'N', 4, 'COL11', 'LITE', 'O');
125 2  DECLARE M41ICE STRUCTURE(  
        SHAPE MASK(7) BYTE,  
        SHAPE VALUE(3) BYTE DATA(  
                0, 0, 0, 20H, 4, 0, 16H, 0, 20H, 45H, 67H, 89H),
        SHAPE ADDRESS) DATA(SWITCH: 0, 0, 0, 0, 0, LITENABLE)  
        VDATA(1, MD2, 1), VDATA(2, MD2, 1, LITE)  
        VDATA(3, MD2, 1, AUD, 1, LITE);  
126 2  DECLARE HOLD TEXT(* BYTE DATA(HOLD));
127 2  DISPLAY H4 STATUS: PROCEDURE;
128 3  DECLARE AUDIT TEXT(3) STRUCTURE(CHAR(8) BYTE)  
        DATA(  
                0, 0, 0, 0, 'A', 'AUD', 'A', 'AUD LITE');
129 3  CALL CLEAR LINE(11);  
130 3  CALL CLEAR TEXT(1);  
131 3  IF (IFFMACHINE AND 2) = 0 THEN  
132 3  CALL MOVE(4, 'HOLD TEXT', TEMPBUF(20));
133 3  CALL MOVE(4, 1, 'HOLD TEXT', INDEX(1, IFFMACHINE, 2, 31), TEMPBUF(6));
134 3  CALL MOVE(4, 1, 'HOLD TEXT', INDEX(1, IFFMACHINE, 2, 31), TEMPBUF(6));
135 3  CALL UPDATE LINE(1, 0, 0, 0, 'HOLD ARTIFACT', TEMPBUF);  
136 3  IF IFFMACHINE THEN TEMPBUF(10) = 'B';  
137 3  ELSE TEMPBUF(10) = 'A';  
138 3  CALL UPDATE LINE(2, 3, 1, TEMPBUF);  
139 3  END;
140 3  
141 2  /\ BEGIN MA SELECT CODE \\  
142 2  CALL CLEAR(11);  
143 2  CALL GRID1, SUBMODE GRID1;  
144 2  CALL DISPLAY(1, COUNTABLE);  
145 2  DO FOREVER:  
146 3  CALL DISPLAY H4 STATUS;  
147 3  CALL READ(1, R411, (B));  
148 3  DO CASE SW INDEX1;  
149 4  IFFMACHINE = (IFFMACHINE AND OFÉN)  
150 4  OR 11;  
151 4  OR (NOT IFFMACHINE) AND 2);  
152 4  IFFMACHINE = (IFFMACHINE AND OFÉN) OR 4;  
153 4  IFFMACHINE = IFFMACHINE AND OFÉN) OR 8);  
154 4  IFFMACHINE = (IFFMACHINE AND OFÉN) OR 8);  
155 4  IFFMACHINE = (IFFMACHINE AND OFÉN) OR 8);  
156 4  RETURN;  
157 4  END;  
158 4  /\ OF 6 CASE \\
138 3  END  /* OF DO FOREVER */
139 2  ENDO  /* OF R44SELECT */
SELECT

160 1 DECLARE ANTTEXT(3) STRUCTURE(CHAR(3) BYTE) DATA('TOPDIVBOT');

161 2 ANTSELECT: PROCEDURE;

162 2 DECLARE ANTSTABLEAU(3) BYTE DATA( \
    ROM2.'ANT SELECT', \
    ROM4.'TOP', \
    ROM6.'DIV', \
    ROM8.'BOT':01);

163 2 DECLARE ANTIENABLE STRUCTURE(
    SHAMASK(7) BYTE, \
    SHAVALUE(3) BYTE) DATA( \
    0-0.OOH:0.OOH,0.OOH-0.OOH: \* TOP, DIV, BOT, SUBMODE #/ \
    OOH.1DDH:22H);

164 2 DECLARE ANTIICF STRUCTURE(
    NODE BYTE, \
    MUNKH BYTE, \
    DELACKH BYTE, \
    DELMASK ADDRESS, \
    ECHONAK BYTE, \
    ECHOCOL BYTE, \
    ANTIENABLE ADDRESS) DATA(SWITCH.O.O.O.O.O.,ANTIENABLE);

165 2 CALL CLEAR(2));

166 2 CALL DISPLAY(ANTSTABLEAU1);

167 2 CALL GRID(SUBMODEGRID);

168 2 CALL CLEARHLINE1();

169 2 CALL UPDATEHLINE(1.2.3.,ANTTEXT(IFFANTIENABLEPOSITION));

170 2 CALL READ(ANTIICF);

171 2 IF SW = SUBMODESW THEN RETURN;

173 2 IFFANTIENABLEPOSITION = SWINEX;

174 2 END; \* OF ANTSELECT */
NEEET
175 1 NECKOFF: PROCEDURE;
176 2 DECLARE NECSTABLEAU(* BYTE DATA;
177 2 DECLARE NECH Structure:
  NODE BYTE,
  NUMCOL BYTE,
  DELMACH BYTE,
  DELLMASK ADDRESS,
  ECHINR BYTE,
  ECHINCOL BYTE,
  SWENABLE ADDRESS) DATA(SWITCH.O.O.O.O.O.,ON/OFF,SWENABLE);)

178 2 CALL CLEAR(111);
179 2 CALL UPDATEVLINE(0.20.,4..BLANKLINE));
180 2 CALL GRID1(SWGRID);XGRID);
181 2 CALL DISPLAY(NECSTABLEAU);
182 2 CALL UPDATEVLINE(1.0.3..OFFSHOWTEXT(INDEX(NECHSTATUS,4.1.1)));
183 2 CALL READ(NECH(3));
184 2 IF SW = SUBGRID THEN RETURN;
186 2 IF SWINDEX THEN NECHSTATUS = NECHSTATUS OR 10H /* ON */
188 2 ELSE NECHSTATUS = NECHSTATUS AND OEFH /* OFF */
189 2 END; /* OF NECKOFF */
DECLARE ENTRYTABLE(i) BYTE DATA
ROMZ:ENTER_CODE".01"

DECLARE (INITCODE) STRUCTURE
160 BYTE,
161 NORTH BYTE,
162 DELTA BYTE,
163 DELINJM MASK ADDRESS,
164 EDCHROM BYTE,
165 ECHOCOL BYTE,
166 SHAHABLE ADDRESS)) DATA(OCTAL0.0.0.0.2.11.0.HOFFSUBCODEENABLE)011

DECLARE CODE(3) STRUCTURE
168 BYTE,
169 NORTH BYTE,
170 DELTA BYTE,
171 DELINJM MASK ADDRESS,
172 EDCHROM BYTE,
173 ECHOCOL BYTE,
174 SHAHABLE ADDRESS))

DECLARE CODETAB13) BYTE DATA(ROMZ:ENTER_CODE'011

DECLARE LABELTAB(3) STRUCTURE(TEXT(7) BYTE) DATA1
ROMA-COL11/M-1 '0.'
ROMA-COL6/"M-2 '0.'
ROMA-COL11/"M-34'011

DECLARE NUMDIGITS(3) BYTE DATA(2.4.D1.

DECLARE MINCODE(4) BYTE DATA("DPOS")

DECLARE (MAINCODE(3) STRUCTURE(DIGITS(4) BYTE) DATA1
"T)00","7777","7777")

DECLARE (DATA(ROMZ: "M2 GUARDED",011

DECLARE (DATA(ROMZ: "M2 GUARDED") BYTE)
REJECT
200 2 IX = SWINDEX1
201 2 IF IX > 0 THEN CHOFFMASK = SHL(1,IX)
202 2 ELSE ONHOFFMASK = IX
203 2 N = NUMDIGITS(IX)
204 2 CALL CLEAR(2); 205 2 CALL GRID1.DIGITBGRID();
206 2 IF IX = 1 THEN DO: /# M2 IS GUARDED */
207 2 CALL DISPLAY(GAR2(TEITSTAB));
209 3 CALL READ(GART1CB); 210 3 IF SW = SUBMODE#SH THEN RETURN;
211 3 END)
214 2 CALL CLEARLINE(21); 215 2 CALL DISPLAY(CODE4TAB);
216 2 CALL DISPLAY(OCTAL#DATALEA1); 217 2 CALL DISPLAY(LABELTAB(1));
218 2 CALL MOVE19..INIT(CODE4ICB, CODE4ICB);
219 2 CODE4ICB.NUMICH = N;
220 2 DO FOREVER;
221 3 CALL DISPLAY(SUBMODE#STATUSLINE); 222 3 CALL READY(CODE4ICB);
223 3 IF DATAENTERED AND (SW = ENTER#SH) THEN DO:
225 4 IF LIMIT(IX, INBUF..MINCODE..MAXCODE(IX)) THEN GO TO AT
227 4 CALL ERROR(11) /# ILLEGAL ENTITY */
228 4 END;
229 3 ELSE
230 4 IF IFSTATUS = IFSTATUS AND (NOT ONHOFFMASK) /# OFFSH */
231 4 IF IFSTATUS = IFSTATUS OR ONHOFFMASK /# ONSH */
232 4 RETURN; /# SUBMODE#SH */
233 4 END)
234 3 END /# OF DO FOREVER */
235 2 AT
236 2 CALL MOVE19..INBUF..DATAADD(IX));
237 2 END /# OF CODEENTRY */
SELECT
237 1 STATSPAGE: PROCEDURE;
238 2 DECLARE STATSTABLEAU(4) BYTE DATA;
239 2 DECLARE STATENABLE STRUCTURE;
240 2 DECLARE STATISTICS STRUCTURE;
241 2 FORMATPAGE: PROCEDURE;
242 3 /0 FORMAT ROM1 0/:
243 3 CALL CLEARTEMPBUF;
244 3 CALL MOVE(2..IFFM11CODE,.TEMPBUF(7));
245 3 IF IFFSTATUS THEN TEMPHBUF(9) = 'a';
246 3 IF IFFMACRO THEN TEMPHBUF(15) = 'b';
247 3 ELSE TEMPHBUF(15) = 'a';
248 3 IF IFFSTATUS AND 8 > 0 THEN TEMPHBUF(19) = 'e';
249 3 CALL UPDATELINE(4.0.22,.TEMPBUF(1));
250 3 /0 FORMAT ROM2 0/:
251 3 CALL CLEARTEMPBUF;
252 3 CALL MOVE(4..IFFM21CODE,.TEMPBUF(5));
253 3 IF (IFFSTATUS AND 2) > 0 THEN TEMPHBUF(9) = 'a';
254 3 CALL MOVE(13..IFFSTATUSINDEX(4.41))..TEMPBUF(16));
255 3 IF (IFFSTATUS AND 10) > 0 THEN TEMPHBUF(19) = 'e';
256 3 CALL UPDATELINE(6.0.19,.TEMPBUF(1));
257 3 /0 FORMAT ROM3 0/:
258 3 CALL MOVE(4..IFFMACRO,.TEMPBUF(5));
259 3 IF (IFFSTATUS AND 4) > 0 THEN TEMPHBUF(9) = 'a';
260 3 CALL MOVE(3..ANTTEXT(1FFMAXEXTPOSITION),.TEMPBUF(16));
261 3 CALL UPDATELINE(8.0.19,.TEMPBUF(1))
262 3 END; /0 OF FORMATPAGE 0/;
DO FOREVER:
  CALL CLEAR(STATPAGE)
  CALL DISPLAY(STATTABLEAU):
  CALL FORVATFACE:
  CALL READY(STATICAP).
  CALL CLEARLINE(0):
  DO CASE SHTINDEX:
    CALL CODEENTRY /* M-1 */
    CALL CODEENTRY /* M-2 */
    CALL CODEENTRY /* M-36 */
    CALL MASELECT:
    CALL MGEHEL:
    CALL ANISELECT:
    RETURN; /* SUBMODE */
  END:
END /* OF DO FOREVER */
END /* OF STATPAGE */
283 1  
DECLARE IFFSUBMODE(20) BYTE DATA(  
  ROM.+"IFF",  
  ROM.+COL6,"RAD",  
  ROM.+"N-6",COL11,"TEST",COL11,"TEST",01)  
285 2  
DECLARE IFFSUMENABLE STRUCTURE(  
  SHVMSK(7) BYTE,  
  SHVVALUE(8) BYTE DATA(  
  0,0,0,0,0,0,0,0,10H,0,  
  01H,23H,34H,55H,66H,78H,9AH,0A0H)  
286 2  
DECLARE IFFSUM STRUCTURE(  
  NODE BYTE,  
  NUMACH BYTE,  
  DELMACH BYTE,  
  DELMASK ADDRESS,  
  ECHCHROM BYTE,  
  ECHOCOL BYTE,  
  SUMENABLE ADDRESS) DATA(SWITCH,0,0,0,0,0,0,.,IFFSUNENABLE))  
287 2  
DO FOREVER:  
288 3  
CALL CLEAR01;  
289 3  
CALL GRID(SUIMODEGRID);  
290 3  
CALL DISPLAY(IFFSCHNMODEDESTATELEAU);  
291 3  
CALL DISPLAYSUIMODESTATUSLINE;  
292 3  
CALL READ(IFFSUMCH);  
293 3  
ID CASE SUIMODE:  
294 4  
CALL CODEENTRYl /N N-1  
295 4  
CALL CODEENTRYl /N N-2  
296 4  
CALL CODEENTRYl /N N-3A  
297 4  
CALL STATPAGE;  
298 4  
CALL MASELECT;  
299 4  
CALL NORMENTRY;  
300 4  
CALL ANTSELECT;  
301 4  
CALL NOIMOFF;  
302 4  
CALL RASTEPT;  
303 4  
CALL TEST;  
304 4  
CALL IFFSUMOFF;  
305 4  
END / OF DO CASE  
306 3  
END / OF DO FOREVER  
307 2  
END / OF IFFSUMMODE */  
308 1  
END / OF IF */  

MODULE INFORMATION:  
CODE AREA SIZE = 0951H 23530  
VARIABLE AREA SIZE = 0013H 190  
MAXIMUM STACK SIZE = 000EH 140  
948 LINES READ
PL/M-80 COMPILER

0 PROGRAM ERROR(S)

END OF PL/M-80 COMPILETION
ISIS-II PL/M-99 V2.1 COMPILATION OF MODULE HWIO
OBJECT MODULE PLACED IN F:IS/HWIO.L06
COMPILER INVOKED BY PL/M-99 F:IS/HWIO.SRC DATE(27OCT79) DEBUG

*/INTVECTOR(4,20H) TITLE('HWIO')*
1 1 HWIO DO
2 1 DECLARE RAMBUF(1536) BYTE EXTERNAL;
3 1 RAMBUF(1536) BYTE EXTERNAL;
4 1 DECLARE COUNT BYTE;
5 1 END;

/* THESE INTERRUPTS (0 AND 1) ARE GENERATED BY THE INTERRUPT CONTROLLER
   (UP 8229) 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 O. */
6 1 DECLARE: PROCEDURE(SAV) EXTERNAL;
7 1 /* THIS PROCEDURE STARTS PROGRAMMABLE TIMER 0 WHICH GENERATES THE
     INTERRUPT THAT CAUSES THE FINGERED SWITCH TO BE READ. */
8 7 DECLARE COUNT = 31 /* DELAY MULTIPLIER */
9 8 OUT(10H) = 01 /* LSB */
10 8 OUT(11H) = 01 /* MSB */
11 8 OUT(10H) = 01 /* MSB */
13 8 OUT(10H) = 20H /* SIGNAL END OF INTERRUPT */
14 8 END;
16 1 READMEMBRANDH procedure INTERRUPT 11:
17 1 DECLARE (MASK:ROW, COLS,0) BYTE,
18 1 HEMBRANDH BYTE DATA(0,0,1,0,2,3,0,4),
19 1 SAVBUF(7) BYTE
20 2 DECLARE COUNT = COUNT -11:
22 2 IF COUNT > 0 THEN DO:
23 3 OUTPUT(11H) = 01 /* START THE TIMER AGAIN */
24 3 OUTPUT(11H) = 01
25 3 OUTPUT(10H) = 20H /* CLEAR THE INTERRUPT */
26 3 RETURN
28 3 END;
30 2 MASK = 0FH
32 2 DO ROW = 0 TO 63 /* INPUT DATA FOR ALL ROWS WHILE */
34 3 OUTPUT(12H) = MASK /* THE SWITCH IS DEPRESSED */
35 3 SAVBUF(ROW) = INPUT(13H);1
37 3 END
39 2 RETURN
41 2 IF Cols > 0 THEN DO: /* IF THIS IS THE RIGHT ROW THEN */
43 3 MASK = 11
45 3 DO ROW = 0 TO 63
47 3 COLS = NOT SAVBUF(ROW1);1
49 3 IF COLS > 0 THEN DO: /* IF THIS IS THE RIGHT ROW THEN */
P"i-80

COMPILER 11110

270CT79 PAGE 2

32  4  $ = -11
34  4  DO WHILE COLS > 0: /# ... LOCATE THE CORRECT COLUMN # /
35  5  Q = Q + 11
36  5  COLS = SHR(COLS, 11)
37  5  ENI
38  4  IF ROW > 4 THEN Q = NEW(42); /# CORRECTION FOR LAST TWO ROWS # /
39  4  CALL ADD(43641H); Q11: /# ADD '5' VALUE TO INPUT QUEUE # /
40  4  OUTPUT(12H) = 30H; /# REENABLE THE TIMER # /
41  4  OUTPUT(30H) = 01; /# RESTART THE MEMBRANE MASK # /
42  4  OUTPUT(0) = 20H; /# SIGNAL END OF INTERRUPT # /
43  4  RETURN
44  4  END: /# OF IF COLS > 0 # /
45  4  ELSE
46  5  IF (Q = 0) THEN
47  6  ROW = SHAH; /# TRY THE NEXT ROW # /
48  5  END: /# OF DO ROW # /
49  4  END: /# OF DO ROW # /
50  4  END: /# OF IF ROW > 4 # /
51  4  END: /# OF IF COLS > 0 # /
52  1  SCREEN.INTENSITY: PROCEDURE(LEVEL) PUBLIC:
53  2  DECLARE LEVEL BYTE:
54  2  OUTPUT(32H) = LEVEL;
55  2  END:
56  1  INITIALIZE: PROCEDURE PUBLIC:
57  1  /* INITIALIZE VARIOUS HARDWARE DEVICES # /
58  2  DISABLE:
59  2  OUTPUT(32H) = 83H
60  2  OUTPUT(22H) = 80H
61  2  OUTPUT(13H) = 30H; /# RESTART TIMER 0 TO MODE 0 # /
62  2  OUTPUT(13H) = 70H; /# RESTART TIMER 1 TO MODE 0 # /
63  2  OUTPUT(13H) = 080H; /# RESTART TIMER 2 TO MODE 0 # /
64  2  OUTPUT(30H) = 01; /# LOAD MEMBRANE ENABLE MASK # /
65  2  CALL SCREEN.INTENSITY(01); /# BLANK THE SCREEN # /
66  1  /* THE FOLLOWING VALUES MUST AGREE WITH THE VALUES USED IN THE
67  2  INSPECTOR STATEMENT AT THE BEGINNING OF THIS MODULE, # /
68  2  OUTPUT(0) = 36H /# INITIALIZE INTERRUPT CONTROLLER - ICH1 # /
69  2  OUTPUT(1) = 01 /# INTERVAL = 4, LOCATION - ICH2 # /
70  2  DISABLE:
71  2  END: /# OF INITIALIZE # /
72  1  UPDATE: PROCEDURE PUBLIC:
73  1  /* INITIATE DMA TRANSFERS TO LOAD REFRESH MEMORIES # /
74  2  DISABLE:
PL/M-80 COMPILER  N410

72 2 OUTPUT(S4H) = LOW,OP,PA1A; /* INITIATE DMA 1 */
73 2 OUTPUT(S4H) = HIGH,OP,PA1A;  
74 2 OUTPUT(S5H) = 0FH; /* LW OF WORD COUNT */
75 2 OUTPUT(S5H) = 0FH; /* LW OF WORD COUNT */
76 2 OUTPUT(S6H) = 4CH; /* OUTPUT MUX TO DMA CONTROLLER */
77 2 OUTPUT(S2H) = 0FH; /* TOGGLE DMA R02 */
78 2 OUTPUT(S2H) = 01;
79 2 CALL TIME(60H); /* SHORT DELAY */
80 2 OUTPUT(S5H) = LOW,OP,PA1B;  
81 2 OUTPUT(S5H) = HIGH,OP,PA1B;  
82 2 OUTPUT(S6H) = 0FH;  
83 2 OUTPUT(S7H) = 0FH;  
84 2 OUTPUT(S8H) = 4CH;  
85 2 OUTPUT(S2H) = 80H; /* TOGGLE DMA R03 */
86 2 OUTPUT(S2H) = 01;
87 2 ENABLE;
88 2 END; /* OF UPDATEARN */
89 1 END; /* OF N410 */

MODULE INFORMATION:

CODE AREA SIZE = 018AH 3900D
VARIABLE AREA SIZE = 0003H 130
MAXIMUM STACK SIZE = 0004H 100
128 LINES READ
0 PROGRAM ERROR(S)

END OF PL/M-80 COMPILATION
PL/M-80 Compiler  FILEBUF

ISIS-11 PL/M-80 v2.0 Compilation of Module FILEBUF
OBJECT MODULE PLACED IN (FILEBUF.REF)
COMPILER INVOKED BY:  PL/M 200:1FILEBUF.SRC DATE(890779)

#TITLE('FILEBUF')
   FILE DO;
2 1 DECLARE
   FA(1536) BYTE EXTERNAL;  /* 164412 */
   FA(1536) BYTE EXTERNAL;
   FA(1536) BYTE EXTERNAL;
   FA(1536) BYTE EXTERNAL;
3 1 DECLARE TRUE LITERALLY 'OFF',
      LITERALLY '0';

4 1 FILEBUF: PROCEDURE(ROW:CHCOL:TEXTPTR:REFPTR:NCHAR:MODE) PUBLIC;
      DECLARE ROW BYTE,  /* ROW NUMBER: 0 - 8 */
         CHCOL BYTE,  /* COLUMN ADDRESS OF FIRST CH: 0 - 23 */
         TEXTPTR ADDRESS,  /* TEXT ADDRESS */
         REFPTR ADDRESS,  /* REFERENCE TEXT ADDRESS */
         NCHAR BYTE,  /* NUMBER OF CHS IN TEXT STRING */
         MODE BYTE;  /* SEE BELOW */
5 2 DECLARE (TEXT BASED TEXTPTR)(1) BYTE,
         (FONT BASED TEXTPTR)(1) BYTE,
         (FONT/TEXT/REFPTR/BUFPTR/BUF/SCENT)(1) ADDRESS;
      FONT BASED TEXTPTR(1) BYTE,
      (REFPTR BASED REFPTR)(1) BYTE,
      BUFFER BUF PTR)(1) BYTE,
      CHCOL,SECTOR,SECTORCOL,MASK,NOTMASK,N,I,J,K,LOC,NEWMASK,
      REFPTR(1) 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
COMPARSED 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). */
SECTOR
7 2 NEWSECTOR: PROCEDURE:
8 3 MASK = SAYMASK, 11
9 3 NOTMASK = NOT MASK
10 3 SECTOR = SECTOR + 15
11 3 IF SECTOR < 6 THEN BUFAPTR = SAYBUF + BUFOFFSET
12 3 ELSE BUFAPTR = SAYBUF + XBUFOFFSET
13 3 IF (SECTOR > 5) AND NEWMASK THEN DO
14 4 Mask = 11
15 4 NOTMASK = OPENH
16 4 NEWMASK = FALSE
17 4 END
18 3 END / OF NEWSECTOR /
19 2 UPDATE: PROCEDURE:
20 3 SECTORCOL = (SECTORCOL + 1) AND OPENH / MODULO 16 /
21 3 IF SECTORCOL = 0 THEN CALL NEWSECTOR
22 3 ELSE BUFAPTR = BUFAPTR + 11
23 3 END / OF UPDATE /
24 2 ADDREASON: PROCEDURE:
25 3 FONTAPTR = CHAFONTDATA + LOGCS:
26 3 REFPONTRAPTR = CHAFONTDATA + REFLOCS:
27 3 DO J = 0 TO 41 / PROCESS 5 CHARACTER FRAGMENTS /
28 4 H = FONT(J)
29 4 R = REF FONT(J)
30 4 DO K = 0 TO 7 BY 16 / 7 TIMES /
31 5 IF K THEN BUF(K) = BUF(K) OR MASK
32 5 ELSE IF R THEN BUF(K) = BUF(K) AND NOTMASK
33 5 H = SHR(H, 11)
34 5 R = SHR(R, 11)
35 3 END
36 3 CALL UPDATE:
37 3 END
38 3 CALL UPDATE: / SKipt THE SIXTH CHARACTER FRAGMENT /
39 3 END / OF ADDREASON /
40 2 RESENVENLOMACH: PROCEDURE:
41 3 REFPONTAPTR = CHAFONTDATA + REFLOCS:
42 3 DO J = 0 TO 41
43 4 R = REF FONT(J)
44 4 DO K = 0 TO 7 BY 16
45 5 IF R THEN BUF(K) = BUF(K) AND NOTMASK
46 5 R = SHR(R, 11)
47 5 END
48 3 CALL UPDATE:
49 3 END
50 3 CALL UPDATE:
51 3 END / OF RESENVENLOMACH /
52 2 SKIP: PROCEDURE:
53 3 SECTORCOL = SECTORCOL + 61
54 3 IF SECTORCOL > 15 THEN DO
55 4 SECTORCOL = SECTORCOL - 161
56 4 CALL NEWSECTOR:
57 4 BUFAPTR = BUFAPTR + SECTORCOL
58 4 END
66 3 ELSE BUFAPTR = BUFAPTR + 61;
67 3 END; /* OF SKIP */

/ * BEGIN FILBUF CODE */
68 2 IF (ROW > 9) OR (CHCOL > 23) THEN RETURN; /* ERROR */
69 2 RHICOL = CHCOL*61;
70 2 SECTOR = RHICOL/161;
71 2 SECTORCOL = SECTOR AND OFH;
72 2 BUFOFFSET = 64 + EQU(spec*1601);
73 2 IF SECTOR < 6 THEN DO:
74 3 BUFAPTR = BUFAPTR + BUFOFFSET + SECTORCOL;
75 3 MASK = 11;
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:
81 3 BUFAPTR = BUFAPTR + BUFOFFSET + SECTORCOL;
82 3 NEWMASK = FALSE;
83 3 RTSSR = RTSCOL-51;
84 3 NOTMASK = NOT MASK;
85 3 END:
86 2 IF (NCHAR + CHCOL) > 24 THEN N = 23 - CHCOL;
87 2 ELSE N = NCHAR - 11;
88 2 LACH 3 DO I = 0 TO M: /* PROCESS EACH CHARACTER */
89 3 LOC = TEXT(I) - 20H;
90 3 REFLOC = REM(I) - 20H;
91 3 IF LOC = REFLOC THEN CALL SKIP;/* CH ALREADY IN BUFFER */
92 3 ELSE DO:
93 4 IF LOC > 0 THEN CALL ADDANEACH;/* LOC = 0 => BLANK */
94 4 ELSE DO:
95 5 IF N = 0 THEN CALL ADDANEACH;
96 5 ELSE CALL REMVKECH;
97 5 END;
98 4 END;
99 4 END; /* OF DO I */
100 2 END; /* OF FILBUF */
101 1 END; /* OF CP */

MODULE INFORMATION:

CODE AREA SIZE = 0220H 800D
VARIABLE AREA SIZE = 001EH 30D
MAXIMUM STACK SIZE = 0000H 8D
136 LINES READ
0 PROGRAM ERROR(S)

END OF PL/I-80 compilation
TITLE('GRID')
1
GRID: DO;
2 1 DECLARE RMHBUFA(I) BYTE EXTERNAL,
RMHBUF(I) BYTE EXTERNAL;
3 1 DECLARE HORIZ LITERALLY 'OFHM',
VERT LITERALLY 'OFVM';
4 1 VERTVECT: PROCEDURE(I,YO,YI);
/* GENERATE REFRESH MEMORY DATA FOR VERTICAL VECTOR. 
   THE GENERATED DATA IS 'GRID' WITH THE CONTENTS OF THE REFRESH MEMORY 
   BUFFERS RMHBUFA AND RMHBUF. */
5 2 DECLARE (X, /* COLUMN ADDRESS: 0 <= X <= 143 */
YO, /* STARTING ROW ADDRESS */
YI) /* ENDING ROW ADDRESS: 0 <= YO <= YI <= 95 */
BYTE;
6 2 DECLARE (XH,XL,N) BYTE,
(KH,KL,J) ADDRESS;
7 2 IF (X > 143) OR (YO > YI) OR (YI > 95) THEN RETURN
9 2 N = SHR(X,4);
10 2 L = I AND OFH;
11 2 KO = DOUBLE(YO)*16 + L1
12 2 KL = DOUBLE(YI)*16 + L1
13 2 IF N < 6 THEN
14 2 DO1
15 3 MK = ROL(KOH,N + 11)
16 3 DO J = KO TO KL BY 16;
17 4 RMHBUFA(J) = RMHBUFA(J) OR MK
18 4 END;
19 3 END1
20 2 ELSE
21 3 MK = ROL(KO-H,N - 51);
22 3 DO J = KO TO KL BY 16;
23 4 RMHBUF(J) = RMHBUF(J) OR MK
24 4 END;
25 3 END;
26 2 END; /* OF VERTVECT */
PROCEDURE 10.11.Y:

/\ GENERATE REFRESH MEMORY DATA FOR HORIZONTAL VECTORS.

THE GENERATED DATA IS "CROSS" WITH THE CONTENTS OF THE REFRESH MEMORY

BUFFERS PMBF.A:CI. EMBF.B. */

DECLARE (X0. // STARTING COLUMN ADDRESS /*
   Y)  // ENDING COLUMN ADDRESS 0 <= X <= 143 /*
   // ROW ADDRESS 0 <= Y <= 95 /*
   BYTE;)

DECLARE (NO,N1,LO,LI,PK,PKO,PK1,J) BYTE;

BUFAPTR ADDRESS;

(BUF BASED BUFAPTR)(1) BYTE;

PROCMA: PROCEDURE:

/\ GENERATE HORIZONTAL VECTOR IN DATA FOR MEMORY A, SECTORS 0 - 5. */

MK0 = 11

IF NO > 0 THEN MK0 = SHL(1,NO); /*

IF NO = N0 THEN

DO;  /* SHORT VECTOR TOTALLY CONTAINED IN ONE SEGMENT */

DO J = LO TO LI

BUF(J) = BUF(J) OR MK0

END;

RETURN;

END:

/\ NO > NO - VECTOR EXTENDS MORE THAN ONE SEGMENT */

DO J = LO TO LI: /* LEFT-MOST PORTION */

BUF(J) = BUF(J) OR MK0;

END;

IF N1 > 6 THEN MK1 = 40H;

ELSE MK1 = SHL(1,N1);

IF (N1 - NO) > 1 THEN

DO;  /* MIDDLE PORTION */

MK = MK1 - SHL(NO,1); /*

DO J = 0 TO LI

BUF(J) = BUF(J) OR MK;

END;

END;

IF N1 < 6 THEN

DO J = 0 TO LI: /* RIGHT-MOST PORTION */

BUF(J) = BUF(J) OR MK1;

END;

END:  /* OF PROCMA */
PROCIB: PROCEDURE:
/* GENERATE HORIZONTAL VECTOR MN DATA FOR MEMORY B, SECTORS 6 - 8. */

MKO = 11
IF NO > 6 THEN MKO = SHL(I,NO - 6);  
IF NO > 5 THEN
   DO:
       IF NO = NL THEN
           DO: /* SHORT VECTOR TOTALLY CONTAINED IN ONE SEGMENT */
               DO J = 0 TO L1;
               BUF(I) = BUF(I) OR MKO;
           END;
           RETURN;
       END:
       /* NL > NO */
       DO J = L0 TO L1: /* INITIAL PORTION OF VECTOR */
       BUF(I) = BUF(I) OR MKO;
    END;
    END;
MK1 = ROL(80H,NL - 5);
IF (NL - NO) > 1 THEN
    DO:
        MK = MK1 - 11
        DO J = 0 TO L1: /* MIDDLE PORTION */
        BUF(I) = BUF(I) OR MK1;
    END;
    END:
    DO J = 0 TO L11: /* END PORTION */
    BUF(I) = BUF(I) OR MK1;
    END;
    END:
END: /* OF PROCIB */
SELECT
    /* BEGIN HORIZVENT PROCESSING */

88 2 IF (X0 > X1) OR (X1 > 103) OR (Y > 95) THEN RETURN
90 2 XI = SHR(103,4) / STARTING SECTOR INDEX: 0 - 8 */
91 2 X1 = SHR(X1,4) /* ENDING SECTOR INDEX */
92 2 LD = IO AND OPN;
93 2 LI = XI AND OPN;
94 2 IF NO < 6 THEN
95 2 DO:
96 3 BUFAPTR = .RAMBUF + DOUBLE(Y#16)
97 3 CALL PROC(A);
98 3 END:
99 2 IF XI > 5 THEN
100 2 DO:
101 3 BUFAPTR = .RAMBUF + DOUBLE(Y#16)
102 3 CALL PROC(B);
103 3 END:
104 2 END: /* OR HORIZVENT */
SELECT

GRID: PROCEDURE (LISTAPTR) PUBLIC:

/* GENERATE RH DATA FOR A GRID DEFINED BY A LIST OF HORIZONTAL
AND VERTICAL VECTORS. THE LIST HAS THE FOLLOWING FORMAT:
GRIDLIST := (DIR, START, END, LOCI1, LOCI2, ..., LOCN, OFFH)

1. IF DIR = Hori THEN START = XO; END = XI; LOCI = YI. 1 <= I <= N,
   THIS DEFINES N HORIZONTAL VECTORS AT COORDINATE POSITIONS
   (X0,Y1-X1,Y1), (X0,Y2-X1,Y2), ..., (X0,YN-X1,YN).

2. IF DIR = Vert THEN START = YO; END = Y1; LOCI = XI. 1 <= I <= N,
   THIS DEFINES N VERTICAL VECTORS AT COORDINATE POSITIONS
   (XI,Y0-X2,Y1), (XI,Y2-X2,Y1), ..., (XI,YW-Y2,Y1).

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

DECLARE LISTAPTR ADDRESS,
(LIST BASED LISTAPTR)(1) BYTE,
(CH1.X0-X1.Y0.Y1) BYTE:

CH = LIST (CH1);
I = 0;
DO WHILE CH < OFFH;
IF CH = Hori THEN
   DO:
      IO = LIST (I + 1);
      XI = LIST (I + 2);
      CH = LIST (I + 3);
      I = I + 3;
   DD:
   DO WHILE CH < OFFH:
      CALL HORIZVECTOR (X0.X1.CH1);
      I = I + 1;
      CH = LIST (CH1);
   END:
   ELSE:
      ELSE:
      DO:
         IF CH = Vert THEN
            DO:
               YO = LIST (I + 1);
               Y1 = LIST (I + 2);
               CH = LIST (I + 3);
               I = I + 3;
            DD:
            DO WHILE CH < OFFH:
               CALL VERTVECTOR (Y0.Y1.CH1);
               I = I + 1;
               CH = LIST (CH1);
            END:
            ELSE:
               ELSE RETURN: /* ERROR */
            END:
            END:
            DO WHILE CH < OFFH:
               DO:
               /* OF GRID */
            DD:
            ELSE:
               /* OF COUNGRID */
            END:
MODULE INFORMATION

CODE AREA SIZE = 045CH 11070
VARIABLE AREA SIZE = 0021H 320
MAXIMUM STACK SIZE = 0006H 60
203 LINES READ
0 PROGRAM ERROR(S)

END OF PL/11-80 COMPILATION
TITLE("DNAV")
"/ INITIATION, DISPLAYACTIVEWINPT, DISPLAYPRESENTPosition, DMNISUBMODE, DMNNAVSTATUS */
1 2 DECLARE HORIZ LITERALLY "OPEN",
   VERT LITERALLY "OFCH":
INC INCLUDE(FI1CODULIT.SRC)
5 2 INCLUDE(PI1CODULIT.SRC)
7 2 END;
9 2 END;
11 2 END;
13 2 END;
15 2 END;
17 2 END;
19 2 END;
21 2 END;
23 2 END;
25 2 END;
27 2 END;
29 2 END;
31 2 END;
33 2 END;
35 2 END;
37 2 END;
39 2 END;
41 2 END;
43 2 END;
45 2 END;
47 2 END;

/* PUBLIC VARIABLES USED BY DNAV PROCEDURES */
1 DECLARE ESO(1) BYTE PUBLIC DATA(000'11)
DECLARE SUBMODEXICB STRUCTURE(
    SUBMASK(7) BYTES,
    SUBVALUE(11) BYTES) PUBLIC DATA(0.0.0.0.0.0.0.804.0.0.0.0.0.0.0)

DECLARE SUBMODEXICB STRUCTURE(
    NODE BYTE, NUMACH BYTE, DELMACH BYTE, DELMASK ADDRESS,
    ECHORON BYTE, ECKICOL BYTE, SHWENABLE ADDRESS) DATA(0.0.0.0.0.0.0.804.0.0.0.0.0.0.0)

DECLARE SUBMODEXICB STRUCTURE(
    NODE BYTE, NUMACH BYTE, DELMACH BYTE, DELMASK ADDRESS,
    ECHORON BYTE, ECKICOL BYTE, SUBVALUE(11) BYTES) PUBLIC DATA(0.0.0.0.0.0.0.804.0.0.0.0.0.0.0)

DECLARE SUBMODEXICB STRUCTURE(
    NODE BYTE, NUMACH BYTE, DELMACH BYTE, DELMASK ADDRESS,
    ECHORON BYTE, ECKICOL BYTE, SHWENABLE ADDRESS) PUBLIC DATA(0.0.0.0.0.0.0.804.0.0.0.0.0.0.0)

DECLARE SUBMODEXICB STRUCTURE(
    NODE BYTE, NUMACH BYTE, DELMACH BYTE, DELMASK ADDRESS,
    ECHORON BYTE, ECKICOL BYTE, SUBVALUE(11) BYTES) PUBLIC DATA(0.0.0.0.0.0.0.804.0.0.0.0.0.0.0)

DECLARE SUBMODEXICB STRUCTURE(
    NODE BYTE, NUMACH BYTE, DELMACH BYTE, DELMASK ADDRESS,
    ECHORON BYTE, ECKICOL BYTE, SUBVALUE(11) BYTES) PUBLIC DATA(0.0.0.0.0.0.0.804.0.0.0.0.0.0.0)

DECLARE SUBMODEXICB STRUCTURE(
    NODE BYTE, NUMACH BYTE, DELMACH BYTE, DELMASK ADDRESS,
    ECHORON BYTE, ECKICOL BYTE, SUBVALUE(11) BYTES) PUBLIC DATA(0.0.0.0.0.0.0.804.0.0.0.0.0.0.0)

DECLARE SUBMODEXICB STRUCTURE(
    NODE BYTE, NUMACH BYTE, DELMACH BYTE, DELMASK ADDRESS,
    ECHORON BYTE, ECKICOL BYTE, SUBVALUE(11) BYTES) PUBLIC DATA(0.0.0.0.0.0.0.804.0.0.0.0.0.0.0)

DECLARE SUBMODEXICB STRUCTURE(
    NODE BYTE, NUMACH BYTE, DELMACH BYTE, DELMASK ADDRESS,
    ECHORON BYTE, ECKICOL BYTE, SUBVALUE(11) BYTES) PUBLIC DATA(0.0.0.0.0.0.0.804.0.0.0.0.0.0.0)

DECLARE SUBMODEXICB STRUCTURE(
    NODE BYTE, NUMACH BYTE, DELMACH BYTE, DELMASK ADDRESS,
    ECHORON BYTE, ECKICOL BYTE, SUBVALUE(11) BYTES) PUBLIC DATA(0.0.0.0.0.0.0.804.0.0.0.0.0.0.0)

DECLARE SUBMODEXICB STRUCTURE(
    NODE BYTE, NUMACH BYTE, DELMACH BYTE, DELMASK ADDRESS,
    ECHORON BYTE, ECKICOL BYTE, SUBVALUE(11) BYTES) PUBLIC DATA(0.0.0.0.0.0.0.804.0.0.0.0.0.0.0)

DECLARE SUBMODEXICB STRUCTURE(
    NODE BYTE, NUMACH BYTE, DELMACH BYTE, DELMASK ADDRESS,
    ECHORON BYTE, ECKICOL BYTE, SUBVALUE(11) BYTES) PUBLIC DATA(0.0.0.0.0.0.0.804.0.0.0.0.0.0.0)

DECLARE SUBMODEXICB STRUCTURE(
    NODE BYTE, NUMACH BYTE, DELMACH BYTE, DELMASK ADDRESS,
    ECHORON BYTE, ECKICOL BYTE, SUBVALUE(11) BYTES) PUBLIC DATA(0.0.0.0.0.0.0.804.0.0.0.0.0.0.0)

DECLARE SUBMODEXICB STRUCTURE(
    NODE BYTE, NUMACH BY...
SELECT
54 1 INITATION: PROCEDURE PUBLIC;
55 2 DECLARE I BYTE.
   ZERO BYTE DATA(01):
56 2 DMV/LAST/WAYPT/ PTR = .DMACOPT/ DATA; /* PRESENT POSITION */
57 2 DMV/DESTINATION = 01
58 2 DMV/DESTINATION = 01
59 // CLEAR ALL TARGETS AND CHECKPOINTS; IE: SET WAYPT. STATUS = 0 0 */
60 2 DO 1 I = 0 TO 9:
61 3 CALL PACK/WAYPT(1,.ZERO,.DMACOPT/DATA(1))
62 3 CALL PACK/WAYPT(1,.ZERO,.DMACOPT/DATA(1))
63 3 END
64 2 DMV/WIND/SPEED(0) = '1'
65 2 CALL MOVE(26,.DMV/WIND/SPEED,.DMV/WIND/SPEED(1))
66 // INITIALIZE VARIABLES APPEARING ON NAV STATUS TABLEAU */
67 2 CALL MOVE(4,'1635').0VARANGE)
68 2 CALL MOVE(3,'1104').0VITRACK/ANGLEERROR)
69 2 CALL MOVE(5,'273').0VIR/CREMENT)
70 2 CALL MOVE(3,'305').0VITRACK/ANGLE)
71 2 CALL MOVE(3,'200').0VITIME/STORAGE)
72 // INITIALIZE PRESENT POSITION */
73 2 DMV/TEM/P/STATUS = 61 /* L/L AND UTH DATA AVAILABLE */
74 2 CALL MOVE(13,'141134116354').0VIT/EM/P/ARRAY/PTY/LAT)
75 2 DMV/TEM/P/STATUS. UTH/SPHERE/ID = 11
76 2 CALL MOVE(13,'01ABC12345678').0VIT/EM/P/ARRAY/PTY/UTH/ZONE)
77 2 CALL PACK/WAYPT(24,.DMV/TEM/P/ARRAY/PTY/.DMACOPT/DATA(1) /* STORE INTO CP1(0) */
78 2 END /* OF INITATION */

79 1 DISPLAY/ACTIVE/WAYPT: PROCEDURE PUBLIC;
80 2 DECLARE WAYPT/DESCRIPTION(2) STRUCTURE(TEXT(4) BYTE) DATA('CP1 TGT')
81 2 DECLARE I BYTE)
82 2 CALL CLEAR/TEM/P/BUF1
83 2 CALL MOVE(4,WAYPT/ DESCRIPTION( DMV/DESTINATION AND 1), .TEM/P/BUF(1))
84 2 TEM/P/BUF(1) = CHR(DMV/DESTINATION 1) AND OFH) + 3OH
85 2 CALL UPDATE/ LINE('10', 6, .TEM/P/BUF(1)
86 2 END /* SIPLEY/ACTIVE/WAYPT */

91 1 DISPLAY/PRESENT/POSITION: PROCEDURE(ROM) PUBLIC;
92 2 DECLARE ROM BYTE)
93 2 CALL CLEAR/TEM/P/BUF1
94 2 CALL UNPACK/WAYPT(34,.DMACOPT/DATA(01),.DMV/TEM/P/WAYPT)
95 2 CALL FORMAT/WAYPT/COORD(,DMV/TEM/P/WAYPT)
96 2 TEM/P/BUF(10), .TEM/P/BUF(1) = 'P'
97 2 CALL UPDATE/ LINE('04', 0.24, .TEM/P/BUF(1)
98 2 END

99
DECLARE DIVA$SUBNODE: PROCEDURE PUBLIC()

DECLARE DIVA$SUBNODE(DIVNAME) BYTE DATA(
    ROM0, 'DIV',
    ROM1, 'FLY', COL6, 'CPT 90 UP NAV',
    ROM2, 'TO', COL12, 'NAV STAT',
    ROM3, 'NEXT', COL16, 'CPT',
    ROM4, 'CPT', COL12, 'L/L STAT',
    ROM5, 'TG', COL7, 'UP', COL11, 'TEST TGT',
    ROM6, COL9, 'DATE', COL16, 'STAT', 011
)

DECLARE DIVA$SHMENABLE STRUCTURE(
    SHMASK(7) BYTE,
    SHVALUE(9) BYTE) DATA(
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
)

DECLARE DLJIC STRUCTURE:
    NODE BYTE, NUMCHAN BYTE, DELMCH BYTE, DELMMSK ADDRESS,
    ECHVROM BYTE, ECHVCOL BYTE, SHMENABLE ADDRESS) DATA(
    SWITCH, 0, 0, 0, 0, 0, 0, 0, DIVA$SHMENABLE
)

DECLARE DIVA$STATUS: PROCEDURE:

DECLARE STATUSTABLEAU(14) BYTE DATA(
    ROM0, 'DIV', COL9, 'STATUS',
    ROM1, 'FLY', COL12, 'SPD', COL17, 'THC',
    ROM2, 'PRO', COL9, 'TRX', COL17, 'THK',
    ROM3, 'TGY', COL9, 'TRN', COL16, 'TEGR', 011
)

DECLARE DATAADD(9) ADDRESS DATA:
    .DIVRANGE,
    .DIVGROUNDSPEED,
    .DIVITRACKANGLEERROR,
    .DIVITBEARING,
    .DIVITRACKANGLE,
    .DIVITCROSSITRACKANGLE,
    .DIVITLINTOROGO,
    .DIVITWINDSPEED,
    .DIVITWINDCHORD)

DECLARE NOCHAR(9) BYTE DATA(0, 0, 0, 0, 0, 0, 0, 0, 0, 0)
/* THE FOLLOWING ARE THE LEFT BYTES OF THE INSERT MASK */

DECLARE INSERTMASK(9) BYTE DATA(00, 0, 0, 0, 0, 0, 0, 0, 0, 0)

DECLARE MASKADDRESS:
    MASKOVERLAY(2) BYTE AT'I MASK01

DECLARE DESTLOC(9) ADDRESS DATA:
    TEMPBUF+3, TEMPBUF+13, TEMPBUF+20,
    TEMPBUF+3, TEMPBUF+12, TEMPBUF+20,
    TEMPBUF+3, TEMPBUF+17, TEMPBUF+13)

DECLARE (1, 1, 11) BYTE;

CALL CLEAR(21);
CALL DISPLAY_ADDRESSTABLEAU;
CALL DISPLAY_PRESENTPOSITION(211)

MASKOVERLAY(0) = 01
DO I = 0 TO 21
109 4        CALL CLEAR@TEMP@BUF)
110 4        DO J = 0 TO 2;
111 5        N = I X J
112 5        MASKOVERLAY(1) = RES@INSERT@MASK(11)
113 5        CALL INSERT(MCHR(11),DATA@C(11),EST@LOC(11).  
114 5        END)
115 4        CALL UPDATE@LINE(I2+4,0,24.,TEMP@BUF));
116 4        END
117 3        CALL READ(SUB#DE$1C$1);   /* WAIT FOR SUB#DE$ SH ENTRY */
118 3        END   /* OF DRAWAWARESTAT */
SUBJECT

/ * BEGIN DIY SUBCODE */

119 2  DO FOREVER
120 3   CALL CLEAR(0)
121 3   CALL UPDATE_LINE(1.O,0.2,OFF,ON,TEXT(SHR(DIVVSTATUS,4))))
122 3   CALL GRID(4,SUBMODE(0))
123 3   CALL DISPLAY(DIV1,DIV1,DESTABLEAU)
124 3   CALL DISPLAYACTIVE(EMPTY)
125 3   CALL DISPLAYFREEHIDEPOS(2)
126 3   CALL READ(DIV1,0)
127 3   DD CASE SUBMODE:
128 4   CALL DINAFLYTO:
129 4   CALL DIVTO:
130 4   CALL DIVBACKUP:
131 4   CALL DIVNAVDEST:
132 4   DIVDESTINATION = (DIVDESTINATION AND 1) OR
       SHR((SHR(DIVDESTINATION,1) + 1) MOD 10, 1)
133 4   DIVVSTATUS = DIVVSTATUS OR 11 /* ON */
134 4   DIVVSTATUS = DIVVSTATUS AND OFEH /* L/L */
135 4   CALL DIVNAVQPTSTATUS(0QPT) /* QPT STATUS */
136 4   CALL DIVVST:
137 4   CALL DIVVUPDATE:
138 4   /* DIV TEST */
139 4   CALL DIVNAVQPTSTATUS(0QPT) /* QPT STATUS */
140 4   DIVVSTATUS = DIVVSTATUS AND OFEH /* OFF */
141 4   DIVVSTATUS = DIVVSTATUS OR OFEH /* ON */
142 4   END: /* OF DO CASE */
143 3   END: /* OF DO FOREVER */
144 2  END: /* OF DIYSUBCODE */
145 1  END: /* OF DIY */

MODULE INFORMATION:

CODE AREA SIZE = 0664H 1124D
VARIABLE AREA SIZE = 0000H 9D
MAXIMUM STACK SIZE = 0000H 10D
313 LINES READ
0 PROGRAM ERROR(S)

END OF DIY-80 COMPILATION
STITLE('DM1')

/* DM1OPT, DM/STG, DM/FLY/TO */

1 DM1 OPT;

*NEST INCLUDE('C2LI(CLL, SRC)

1 DMACOORD DATA ENTRY; Procedure(WAYPTPTR, WAYPT byte) BYTE EXTERNAL;
2 DECLARE WAYPTPTR ADDRESS, WAYPT byte;
3 END;
4 CLEAR: Procedure(STARTROW) EXTERNAL;
5 DECLARE STARTROW BYTE;
6 END;
7 GRID: Procedure(POINTER) EXTERNAL;
8 DECLARE POINTER ADDRESS;
9 END;
10 DISPLAY: Procedure(INAPTR) EXTERNAL;
11 DECLARE INAPTR ADDRESS;
12 END;
13 READ: Procedure(ICMAPTR) EXTERNAL;
14 DECLARE ICMAPTR ADDRESS;
15 END;
16 UPDATELINE: Procedure(ROM, COL, CHAR, TEXTPTR) EXTERNAL;
17 DECLARE (ROM, COL, CHAR) BYTE, TEXTPTR ADDRESS;
18 END;
19 ERROR: Procedure(CODE) EXTERNAL;
20 DECLARE CODE BYTE;
21 END;
22 PACKWAYPT: Procedure(NBYTES, SOURCEPTR, DESTPTR) EXTERNAL;
23 DECLARE NBYTES BYTE, SOURCEPTR, DESTPTR ADDRESS;
24 END;
25 UNPACKWAYPT: Procedure(NBYTES, SOURCEPTR, DESTPTR) EXTERNAL;
26 DECLARE NBYTES BYTE, SOURCEPTR, DESTPTR ADDRESS;
27 END;

/* EXTERNAL VARIABLES */

DECLARE
BLANKLINE (24) BYTE EXTERNAL,
SWAP BYTE EXTERNAL,
SWINDEX BYTE EXTERNAL,
DATHENTERED BYTE EXTERNAL,
Databuf (1) BYTE EXTERNAL,
SINGLEDIGIT IOC ADDRESS EXTERNAL,
SUBNODEGRID BYTE EXTERNAL,
SUBASLGREDKRD BYTE EXTERNAL,
SUBMODESHENABLE BYTE EXTERNAL,

WAYPTWAYPT STRUCTURE:
STATUS BYTE,
UTM(14) BYTE,
LAT(6) BYTE,
LON(7) BYTE.
PLMN-90 Compiler

PAGE 2

DATAVAR(6) BYTE EXTERNAL.

DYNAMODestination BYTE EXTERNAL.

DYNASTARTWAYPTPTR ADDRESS EXTERNAL.

DYNACLIPDATA1(30) STRUCTURE(DIGITS(60) BYTE) EXTERNAL.

DYNACLIPDATA2(10) STRUCTURE(DIGITS(60) BYTE) EXTERNAL.

34 1 DMACOPT: PROCEDURE PUBLIC;
35 2 DECLARE OPTTABLEAU(*4) BYTE DATA(
36 2 3 RAZ."OPT",
36 2 3 RNOG.COLA."OPT",01;)
36 2 DECLARE OPTNUM BYTE;
37 2 CALL CLEAR(21);
38 2 CALL DISPLAY(DIGITS(30));
39 2 CALL DISPLAY(OPTTABLEAU);?
40 2 CALL READS(SINGLE12DIGITS(30));
41 2 IF SW = SUBMODE THEN RETURN;
42 2 IF DATAENTERED THEN DT;
43 2 OPTNUM = IN$BUF(1) + 30H;
44 3 IF DMACORDDATAENTRY(DMVSTEPWAYPT,OPT) = 0 THEN RETURN; /* SUBMODE RETURN */
45 3 CALL PACKWAYPT(34, DMVSTEPWAYPT, DMACOPTDATA(OPTNUM));
46 3 DMALASTWAYPTPTR = $.DMACOPTDATA(OPTNUM);
47 3 END;
48 2 END /* OF DMACOPT */
SELECT
52 1   DN1&TGT PROCEDURE PUBLIC;
53 2   DECLARE TOT2ABLEAU(5) BYTE DATA (1
55 2   DECLARE FRE2T3A2E1S$ENABLE STRUCTURE (1
55 2   DECLARE TOT2ICB STRUCTURE (1
56 2   DECLARE FRE2TAB1 BYTE DATA (2
57 2   DECLARE TOT2ICB STRUCTURE (1
58 2   DECLARE TOT2NUM BYTE (1
59 2   DECLARE FRE3NK LITERALLY '200';
60 2   A:
61 2   CALL CLEAR(2);
62 2   CALL DISPLAY40DIT3K3B;
63 2   CALL READ1,TOT3ICB();
64 2   IF SAV = SUBNODES# THEN RETURN;
65 2   IF SAV = FRE3NK THEN GO TO B;
66 2   IF DATASEN3TERED THEN DO;
67 3   TOT2NUM = [MVARU(0) - 30H]
68 3   IF IMVROR SубМОД3S#ENT3RY(MVAR3P3WAYPT,TOT2) = 0 THEN RETURN /* SUBMODE RETURN */
69 3   CALL PACK3WAYPT(M3P3, MVAR3P3WAYPT, MVAR3T3TOTALDATA(TOT3NUM));
70 3   IM3VR3LASTWAYPT3PR = .MV3RT3TOTALDATA(TOT3NUM);
71 3   RETURN;
72 3   END;
73 2   ELSE DO;
74 3   CALL ERROR1();
75 3   GO TO A;
76 3   END;
77 2   B:
78 4   /* FREE3K SWITCH PRESSED, RECORD PRESENT POSITION. */
79 4   CALL DISPLAY40DIT3K3B();
80 2   CALL READ1,TOT3ICB();
81 2   IF SAV = SUBNODES# THEN RETURN;
82 2   IF SAV = ENTER3V3N THEN DO;
83 3   IF DATASEN3TERED THEN TOT3NUM = [MVARU(0) - 30H];
84 3   ELSE DO;
85 4   IF TOT3NUM = 5 TO 91
86 4   CALL UNPACK3WAYPT(M3P3, MVAR3T3TOTALDATA(TOT3NUM), MVAR3P3WAYPT);
87 5   IF (MVAR3P3WAYPT.STATUS AND 6) = 0 THEN GO TO C;
88 5   END;
89 4   CALL ERROR1(); /* NO UNALLOCATED TARGETS */
90 4   GO TO A;}
97 4 END: /* OF ELSE DO */
98 3 CI
/* COPY PRESENT POSITION TO DESIGNATED TARGET */
CALL UNPACKWAYPT(34, ENVAWAYPT1, DATA(1), ENATAWAYPT1)
3 3 CALL PACKWAYPT(34, ENVAWAYPT1, DATA(1), ENATAWAYPT1); END;
/* DISPLAY COORDINATES OF TARGET AND TARGET NUMBER */
101 2 END: /* OF DATA */
SELECT
102 1 DMAFLYSTO: PROCEDURE PUBLIC:
103 2 DECLARE FLYSTO-TABLEAU() BYTE DATA(
   ROM0,'FLY',
   ROM4,'TOT',0
):=
104 2 DECLARE TYPE(2) STRUCTURE(CH7 BYTE) DATAAI
   ROM0.CHA:'COPY',0,
   ROM7.CHA;'TOT'0
);=:
105 2 DECLARE TOT(8) BYTE DATA(ROM0,'TOT':011
106 2 DECLARE FLYSTO-TABLELE(X) STRUCTURE:
   SNMARK(7) BYTE,
   SNVALUE(2) BYTE DATA:
   0,0,20H,0,30H,0,80H /'COPY,TOT,SUBNODE@/
   0H,20H
107 2 DECLARE FLYSTO-ICB STRUCTURE:
   NODE BYTE, NIMACH BYTE, DEMACH BYTE, DEMMARK ADDRESS,
   ECCHARON BYTE, ECOSCOL BYTE, SHENABLE ADDRESS) DATA:
   SWITCH(0,0,0,0,0,0,FLYSTOENABLE)
108 2 CALL CLEAR(211)
109 2 CALL DISPLAY.FLYSTO-TABLEAU1
110 2 CALL DISPLAY.TYPE(0));
111 2 CALL DISPLAY.TYPE(1));
112 2 CALL GRID1 SUBSYSTEM(COUNT)
113 2 CALL READ(FLYSTOICB3)
114 2 IF SW = SUBNODEAW THEN RETURN;
116 2 INVDESTINATION = SWINDEX1
117 2 CALL CLEAR(211)
118 2 CALL DISPLAY(TOT):
119 2 CALL DISPLAY.FLYSTO-TABLEAU1
120 2 CALL DISPLAY.TYPE(SWINDEX11)
121 2 CALL DISPLAYDIGITS;
122 2 CALL READ(SINGLEDIGITSICB3)
123 2 IF DATA ENTERED AND (SW = ENTERSUM THEN
124 2 INVDESTINATION = INVDESTINATION OR SHL(INBUF(0)-3OH,11)
125 2 END / OF DMAFLYSTO */
126 1 END / OF DMV */

MODULE INFORMATION:

   CODE AREA SIZE = 0063H 611D
   VARIABLE AREA SIZE = 0033H 2D
   MAXIMUM STACK SIZE = 0004H 4D
   245 LINES READ
   0 PROGRAM ERROR(S)

END OF PL/1-80 COMPILATION
$TITLE("DMZ")$

/* DISPLAY(WAYPTYPE, LATLON, DATAM ENTRY, MDVHAP, DATAM ENTRY, DNV4COORD, DATAM ENTRY )*/
1
DMZ: DO:
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34

DECLARE:
OFFLINE(16 BYTE EXTERNAL),
BLANKLINE(24 BYTE EXTERNAL),
SAV BYTE EXTERNAL,
SWINDEX BYTE EXTERNAL,
DATAENTERED BYTE EXTERNAL,
INBUF(1) BYTE EXTERNAL,
ZERO(1) BYTE EXTERNAL,
SUBMODEGROW ADDRESS EXTERNAL,
SUBMODEGROW ADDRESS EXTERNAL,
SUBMODESTP ADDRESS EXTERNAL,
DATAWANT WAYPTPTR ADDRESS EXTERNAL,
DIGITKOBGROW BYTE EXTERNAL,
DIGITKOBST ISLEA BY T EXTERNAL,
DECLARE WAYPTPTR ACCESS;
/* UNPACKED WAYPOINT DATA STRUCTURE */
DECLARE WAYPT BASED WAYPTPTR STRUCTURE
  STATUS BYTE,
  UTNSPHEROTO BYTE,
  UTNSZONE(2) BYTE,
  UTNSLPHA(3) BYTE,
  UTNSVALUE(8) BYTE,
  LAT(6) BYTE,
  LON(7) BYTE,
  IDAYPTSPTR(35)
WINPR(2)
WAYPOINT
PATA;

/* PACKED WAYPOINT DATA STRUCTURE. */
NOTE: THIS DATA OVERLAYS MBUPB. */
DECLARE LASTWAYPT BASED DMYLASTWAYPTPTR STRUCTURE
  STATUS(2) BYTE,
  UTN(2) BYTE,
  LL(26) BYTE,
  IWAYPTDATA(12) BYTE;

DISPLAYWAYPTTYPE: PROCEDURE(TYPE) PUBLIC;
DECLARE TYPE BYTE.
  I BYTE;
/* TYPE FORMAT: */
  BIT0: 0 = CPT, 1 = TOT,
  BIT1: 0 = L/L, 1 = UTN */
DECLARE MPTROM(2) BYTE DATA(4,6),
  MPTICOL(2) BYTE DATA(6,2)
DECLARE MPTTEXT(2) STRUCTURE(CH(4) BYTE) DATA("CPTTOT ")
DECLARE COORDTABLE(2) STRUCTURE(CH(3) BYTE) DATA("UTNL/L ")
  COORDICOL(2) BYTE DATA(6,12)

I = TYPE AND 11
CALL UPDATEFILL(MPTROM(I),MPTICOL(I),4,MPTTEXT(I));
I = SHR(TYPE,1) AND 11
CALL UPDATEFILL(I,COORDICOL(I),3,COORDTEXT(I));
END
DECLARE LATTABLE(4) BYTE DATA(M1.,"ENTER LATITUDE",.
M2..COL15,\"N",.
M3..COL15,\"S",.
M4..COL16,\"C",COL22,\"E",0)!

DECLARE LATDISP(4) STRUCTURE(.
 NODE BYTE, NUMCH BYTE, DELCH BYTE, DELMASK ADDRESS,
 ECHONOM BYTE, ECHCOL BYTE, SHENABLE ADDRESS DATA.
 NORTHSOUTH,1.0,0.2,0.,SUERE(1,STEP)#SHENABLE)!

DECLARE LATVALS(4) STRUCTURE(.
 NODE BYTE, NUMCH BYTE, DELCH BYTE, DELMASK ADDRESS,
 ECHONOM BYTE, ECHCOL BYTE, SHENABLE ADDRESS DATA.
 DIGIT,5.,-20000.2.2.,SUERE(1,STEP)#SHENABLE)!

DECLARE LOHDISP(4) BYTE DATA(M1..COL15,\"LONITUDE",.
M2..COL16,\"E",COL22,\"W",.
M3..COL16,\"C",COL22,\"E",0)!

DECLARE LOHDIRNS(4) STRUCTURE(.
 NODE BYTE, NUMCH BYTE, DELCH BYTE, DELMASK ADDRESS,
 ECHONOM BYTE, ECHCOL BYTE, SHENABLE ADDRESS DATA.
 EASTWEST,1.0,0.2,0.,SUERE(1,STEP)#SHENABLE)!

DECLARE LOHVALS(4) STRUCTURE(.
 NODE BYTE, NUMCH BYTE, DELCH BYTE, DELMASK ADDRESS,
 ECHONOM BYTE, ECHCOL BYTE, SHENABLE ADDRESS DATA.
 DIGIT,6.,-14000.2.1.,SUERE(1,STEP)#SHENABLE)!

DECLARE LATMAX(4) BYTE DATA(N900),.
 LONMAX(4) BYTE DATA(180)!

GETDATA: PROCEDURE(CTL4PTR.DATAPTR..DATATYPE,WAYPTYPE)!
 /*READ LAT OR LON DIRECTION AND VALUE ENTRIES, */

DECLARE CTL4PTR ADDRESS,
 DATAPTR ADDRESS,
 DATATYPE BYTE, /* 0 - LAT, 1 - LON */
 WAYPTYPE BYTE;

DECLARE (DAT BASED DATAPTR)(1) BYTE;

DECLARE CTL4 BASED CTL4PTR STRUCTURE(.
 MAIN_ADDR ADDRESS,
 TAB_ADDR ADDRESS,
 DIR_ADDR ADDRESS,
 VAL_ADDR ADDRESS)!

DECLARE ETAB(4) BYTE DATA(M02..COL4..COL7,\"..COL9\",\"..C01.7",\"0)!

CALL CLEAR(21)!

CALL DISPLAY(WAYPTYPE(WAYPTYPE))!

CALL GRID(DIGIT=\"..\",GRID1)!

CALL DISPLAY(CTL4.TAB4.ADD1)!

CALL READ(CTL4.DIR4.ADD4)!
 /* DIRECTION */

IF (GAM = SUB) OR (GAM = STEPS) THEN RETURN;

IF NOT DATAENTERED THEN GO TO AT /* REJECT CENTERED NOT PRECEDED BY DATA */

DAT0 = INBUF(0)!

CALL UPDATE.LINE(2.0.1,INBUF)!
 /* PUT DIR BACK ON THE DISPLAY */

CALL DISPLAY(DIGIT=ETABTABLE)!

CALL DISPLAY(ETAB)!
 /* INPUT DEGREES VALUE */
CALL READ(CNTL, VAL(*CHACD));
74 2 IF (SAV = SUBMODE1) OR (SAV = STEPMEM) THEN RETURN
76 2 IF NOT DATAENTERED THEN GO TO B;
78 2 IF LIMITST1.INUSE[2].CATATYPE.*ZERO.CNTL,MAXADD() AND
    LIMITST2.INUSE[2].CATATYPE.*ZERO.LATMAX2 THEN
79 2 CALL NONE(5,CATATYPE.,INUSE,DATATYPE[1]);
80 2 ELSE DO:
81 3 CALL ERROR(011) /* INVALID ENTRY */
82 3 GO TO B;
83 3 END;
84 2 END /* OF GETDATA */

LATLONDATAENTRY: PROCEDURE(NAVTADDD,NAVTADDD) BYTE;
85 2 DECLARE NAVTADDD ADDRESS;
86 2 NAVTADDD ADDRESS: /* 0 = CPT, 1 = TOT */
87 2 DECLARE CNTL[2] STRUCTURE:
    MAXADD ADDRESS,
    TABHADD ADDRESS,
    INUSE[2]CPTADD ADDRESS,
    INUSE[2]TADDD ADDRESS DATA,
    LATMAX1, LATTABLE11, LATM1MIC1, LATMAX1C1,
    LONMAX1, LONTABLE11, LONM1MIC1, LONMAX1C1:

88 2 NAVTADDD = NAVTADDD;
89 2 CALL GETDATA(CNTL[0],NAVTADDD,NAVPT,LAT,NAVPTADDD);11
90 2 IF SAV = SUBMODE1 THEN RETURN(011);
92 2 IF SAV = STEPMEM THEN RETURN(0111);
94 2 CALL GETDATA(CNTL[1],NAVTADDD,NAVPT,LON,NAVPTADDD);11
95 2 IF SAV = SUBMODE1 THEN RETURN(011);
97 2 IF SAV = STEPMEM THEN RETURN(0111);
99 2 NAVPT,STATUS = NAVPT,STATUS OR 41
100 2 RETURN(11) /* NORMAL RETURN */
101 2 END /* OF LATLONDATAENTRY */
PROCEDURE: macroaddress

DECLARE macroaddress ADDRESS;
macroaddress TYPE BYTE;

DECLARE macroaddress(10) BYTE DATA;
ROW<"ENTER MAC V05","ROW<COL16,"E".COL22,"W",
ROW<COL16,"C".COL22,"E".011

DECLARE CTNL STRUCTURE:
macroaddress ADDRESS;
TABADDRESS ADDRESS;
RETURN ADDRESS ADDRESS;
VALUEADDRESS ADDRESS DATA(.LATMAX,.MACROADDRESS,.LONGADDR,.LATVAL,.LATVAL)11

106 2      \(\text{macroaddress} = \text{macroaddress}1\)
107 2      \text{CALL CLEAR(11)}
108 2      \text{CALL DISPLAY(macroaddress(macroaddress))}
109 2      \text{CALL GRID(DIGITKEGRID)11}
110 2      \text{CALL DISPLAY(macroaddress)}
111 2      \text{CALL GETDATA(CTNL, \text{macroaddress}, \text{macroaddress}, \text{macroaddress})}
112 2      \text{IF SAV = SUBMODO THEN RETURN(10)}
113 2      \text{IF SAV = STEPSH THEN RETURN(20H)}
114 2      \text{macroaddress.STATUS = \text{macroaddress}.STATUS OR 01}
115 2      \text{RETURN(11)}
116 2      \text{END} /* OF macroaddressENTRY */
DECLARE WAYPTADD ADDRESS, /\ MUST POINT TO AN UNPACKED WAYPOINT DATA BLOCK */
WAYPTTYPE BYTE; /* SEE DISPLAYWAYPTTYPE FOR FORMAT DESCRIPTION */
DECLARE COORDTABLEAU(*) BYTE DATA;
RENAME 'SELECT COORDINATE SYSTEM',
ROW(COL. 'UTH', 'COL.2', 'L/L', '0');
DECLARE COORDTABLEAU(*) BYTE DATA;
SHARED(1) BYTE;
SHARED(2) BYTE DATA;
0.0,0.28,0.0,0.28,
0.0,2.01);
DECLARE COORDTABLEAU(*) BYTE DATA;
MODE BYTE, NUMCH BYTE, DELMACH BYTE, DELMASK ADDRESS,
ENDERON BYTE, ECHOCOL BYTE, SHAHABLE ADDRESS) DATA(
SWITCH:0.0,0.0,0.0,COORDTABLEAU());
DECLARE RETAFLAG BYTE;
CALL CLEAR(2); CALL DISPLAY(.COORDTABLEAU());
CALL DISPLAY(WAYPTTYPE(WAYPTTYPE));
CALL GRID(SUBMODEGRID());
CALL READ(.COORDTABLEAU());
IF SW = SUBMODESW THEN RETURN(0)
IF SW = STEPSW THEN DO1
RETAFLAG = GEMOH /* COPY ALL DATA FROM LAST WAYPOINT */
GO TO 2;
ENDIF;
WAYPTTYPE + SHL(SWINDEX,1));
IF SWINDEX THEN RETAFLAG = LATHLONDATENTRY(WAYPTADD,WAYPTTYPE);
ELSE RETAFLAG = DMHNUTLONDATENTRY(WAYPTADD,WAYPTTYPE);
IF RETAFLAG = 0 THEN RETURN(1)
IF RETAFLAG = 0 THEN RETURN(1)
IF (RETAFLAG = 1 THEN RETURN(1))

/\ 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. THIS SAVING THE OPERATOR A FEW KEYSWROKES.
THE FORMAT OF THE RETURN FLAG IS:
BIT 7 = LATLONDATENTRY
BIT 6 = UTHLONDATAENTRY
BIT 5 = RHMV DATENTRY
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).
/\
WAYPT.STATUS = WAYPT.STATUS OR 0
END:
IF (RETIFLAG AND 20H) > 0 THEN 001
CALL UNPACK(WAYPT);LACT(WAYPT,MAGVAR..WAYPT,MAGVAR)
WAYPT.STATUS = WAYPT.STATUS OR 81
END:
RETURN(1)

END1 /* OF COORD */

END1 /* OF DW2 */

MODULE INFORMATION:

CODE AREA SIZE = 044CH 11000
VARIABLE AREA SIZE = 0013H 210
MAXIMUM STACK SIZE = 000CH 120
329 LINES READ
0 PROGRAM ERROR(S)

END OF PL/I-80 COMPILATION
PL/I-80 Compiler  IV3

IS8S-11 PL/I-80 V3.0 Compilation of Module IV3
Object Module Placed in FX:OIV3.OBJ
Compiler Invoked By: PL/80 FX:OIV3.SRC DATE(OMAR79) DEBUG

*/TITLE("IV3")
/* IVUPD4TE */
1 1 DM3: DO
MOGLIST INCLUDE(FX.CULIT.SRC)
2 1 DISPLAYACTIVATE: PROCEDURE EXTERNAL;
3 2 ENDI
4 1 DISPLAYPRESENTPRESENTATION: PROCEDURE(ROM) EXTERNAL;
5 2 ENDI
6 1 DECLARE ROM BYTE;
7 2 ENDI
8 1 DMACORRDATAENTRY: PROCEDURE(WAYPTPTR.WAYPTTYPE) BYTE EXTERNAL;
9 2 DECLARE WAYPTPTR ADDRESS, WAYPTTYPE BYTE;
10 2 ENDI
11 1 CLEAR: PROCEDURE(STARTWROM) EXTERNAL;
12 2 DECLARE STARTWROM BYTE;
13 2 ENDI
14 1 ERROR: PROCEDURE(CODE) EXTERNAL;
15 2 DECLARE CODE BYTE;
16 2 ENDI
17 1 GRID: PROCEDURE(POINTER) EXTERNAL;
18 2 DECLARE POINTER ADDRESS;
19 2 ENDI
20 1 DISPLAY: PROCEDURE(INHPTPTR) EXTERNAL;
21 2 DECLARE INHPTPTR ADDRESS;
22 2 ENDI
23 1 DISPLAY: PROCEDURE(INHPTPTR) EXTERNAL;
24 2 DECLARE INHPTPTR ADDRESS;
25 2 ENDI
26 1 READ: PROCEDURE(INCHPTPR) EXTERNAL;
27 2 DECLARE INCHPTPR ADDRESS;
28 2 ENDI
29 1 UPDATE: PROCEDURE(ROM, COL, NCHAR, TEXTPTR) EXTERNAL;
30 2 DECLARE (ROM, COL, NCHAR) BYTE, TEXTPTR ADDRESS;
31 2 ENDI
32 1 UNPACK: WAYPT: PROCEDURE(INBYTES, SOURCEPTR, DESTPTR) EXTERNAL;
33 2 DECLARE INBYTES BYTE, (SOURCEPTR, DESTPTR) ADDRESS;
34 2 ENDI
35 1

*/EXTERNAL VARIABLES */
1 DECLARE
2 OFFLINE(6) BYTE EXTERNAL,
3 BLANKLINE(24) BYTE EXTERNAL,
4 DATAMEIERD BYTE EXTERNAL,
5 DMAMIF(1) BYTE EXTERNAL,
6 SW BYTE EXTERNAL,
7 DMINDEX BYTE EXTERNAL,
8 SUBMODEGRID BYTE EXTERNAL,
9 SUBMODECHENABLE BYTE EXTERNAL,
10 DMAPCTDATA(10) STRUCTURE(DIGITS(48) BYTE) EXTERNAL,
11 DMATYPEPHWAYPT STRUCTURE;
12 STATUS BYTE;
DECLARE TAB1() BYTE DATA(ROM0,'DIV',COL5,'UPDATE'),
ROMX.COL6,'COPT'),
ROM7.COL7,'UP'),
ROM8.COL6,'DATE',01;
DECLARE ENABLE STRUCTURE
SM stI$K(7) BYTE,
SM stI$VALUE(2) BYTE DATA(0,0,20H,0,20H,0,80H, /* COPT, UPDATE, SUBMODE #/ 01H,20H));
DECLARE IC8I STRUCTURE
NODE BYTE, NUMCH BYTE, DELNUM BYTE, DELMASC ADDRESS,
ED ONROM BYTE, ED ONCOL BYTE, SW tNABLE ADDRESS) DATA(SWITCH0,0,0,0,0,0,0, ENABLE));
DECLARE TAB2() BYTE DATA(ROM2,'ENTER COPT',01);
DECLARE IC82 STRUCTURE
NODE BYTE, NUMCH BYTE, DELNUM BYTE, DELMASC ADDRESS,
ED ONROM BYTE, ED ONCOL BYTE, SW tNABLE ADDRESS) DATA(SWITCH0,0,0,0,0,0,0, SUBMODEENABLE));
DECLARE FRZESTAB() BYTE DATA(ROM4,' FREE',01);
DECLARE FRZENABLE STRUCTURE
SM stI$K(7) BYTE,
SM stI$VALUE(1) BYTE DATA(0,0,80H,0,0,80H, /* FREE, SUBMODE #/ 01H));
DECLARE FRIENABLE STRUCTURE
NODE BYTE, NUMCH BYTE, DELNUM BYTE, DELMASC ADDRESS,
ED ONROM BYTE, ED ONCOL BYTE, SW tNABLE ADDRESS) DATA(SWITCH0,0,0,0,0,0, FRIENABLE));
DECLARE UPDATESTAB() BYTE DATA(ROM2,'UPDATE DIST 12.3',
ROM7.COL7,'UP'),
ROM8.COL6,'DATE',01);
DECLARE UPDATEENABLE STRUCTURE
SM stI$K(7) BYTE,
SM stI$VALUE(1) BYTE DATA(0,0,0,0,20H,0,80H, /* UPDATE, SUBMODE #/ 01H));
DECLARE UPDATEC3 STRUCTURE
NODE BYTE, NUMCH BYTE, DELNUM BYTE, DELMASC ADDRESS,
ED ONROM BYTE, ED ONCOL BYTE, SW tNABLE ADDRESS) DATA(SWITCH0,0,0,0,0, UPDATEENABLE));
DECLARE NEWFREETAB() BYTE DATA(ROM4,' FREE'
ROM5,' NEW',01);
DECLARE NEWFREEENABLE STRUCTURE
SM stI$K(7) BYTE,
SM stI$VALUE(2) BYTE DATA(0,0,0,0,0,0,0, /* NEWFREEENABLE));
DECLARE UPDATESTAB() BYTE DATA(ROM2,'UPDATE DIST 12.3',
ROM7.COL7,'UP'),
ROM8.COL6,'DATE',01);
DECLARE UPDATEENABLE STRUCTURE
SM stI$K(7) BYTE,
SM stI$VALUE(1) BYTE DATA(0,0,0,0,20H,0,80H, /* UPDATE, SUBMODE #/ 01H));
DECLARE UPDATEC3 STRUCTURE
NODE BYTE, NUMCH BYTE, DELNUM BYTE, DELMASC ADDRESS,
ED ONROM BYTE, ED ONCOL BYTE, SW tNABLE ADDRESS) DATA(SWITCH0,0,0,0,0, UPDATEENABLE));
DECLARE NEWFREETAB() BYTE DATA(ROM4,' FREE'
ROM5,' NEW',01);
DECLARE NEWFREEENABLE STRUCTURE
SM stI$K(7) BYTE,
SM stI$VALUE(2) BYTE DATA(0,0,0,0,0,0,0, /* NEWFREEENABLE));
DECLARE NEWFRIEND STRUCTURE
MODE BYTE. NAME 8-17E. CODE ADDRESS. EDATION BYTE. EDAT. L. S. U. L. S. U. (ADDRESS) DATA.
SWITCH-0.0.0.0.0.0.0.0.0.0. (NEWFRIEND);}

DECLAR I BYTE;

TOPT:
CALL CLEAR(0):
CALL DISPLAY ACTIVEKEYPT;
CALL DISPLAY PRESENTPOSITION(21):
CALL DISPLAY(TAB1):
CALL GRID(, SUBNODEGRID):
CALL READ. ICB1)
DO CASE SW1 INDEX;
GO TO A1; /* CWT SWITCH PRESSED */
GO TO C1; /* UPDATE SWICTH PRESSED */
RETURN; /* SUBMODE */
END

CALL CLEAR(21):
/* "ENTER CWT" */
CALL DISPLAY(0):/* ""/*

CALL READ. IC62): /* CWT NUMBER */
IF SW = SUBNODE# THEN RETURN:
IF NOT DATA ENTERED THEN DO:
CALL ERROR(011); /* "INVALID ENTRY" */
GO TO 81
END:
I = IDNUM(6) - 30H:
CALL UNPACKKEYPT(34,..DMV,DMV,DMV,DATA(1),..DMVASDMV,DMV);
IF (DMV,DMV,DMV,STATUS AND 6) = 0 THEN DO:
CALL ERROR(231); /* "NO DATA FOR THIS CWT" */
GO TO 81
END:
/* CWT SELECTED. WAIT FOR CWF */
CALL CLEAR(21):
CALL DISPLAY(,FRIENDTAB);
CALL GRID. SUBNODEGRID):
CALL READ. FRIEND;
IF SW = SUBNODE# THEN RETURN:
/* FWF ENTERED. WAIT FOR UPDATE */
CALL CLEAR(21):
CALL DISPLAY(,UPDATETAB):
CALL READ. UPDATE:
RETURN; /* DONE */

CALL CLEAR(21):
/* UPDATE SWITCH PRESSED FIRST */
CALL CLEAR(21):
CALL DISPLAY(NEWFRIENDTAB)
CALL READ. NEWFRIEND:
DO CASE SW1 INDEX:
GO TO E1; /* CWF */
GO TO D1; /* NEW */
97 3 RETURN; /* SUB-KEY * /
98 3 END
99 2 M
100 2 /* NEW SWITCH PRESSED. ENTER COORDINATES */
101 2 IF DVRACORDDATAFILL.LSTEP+LWAPL+OFT) = 0 THEN RETURN;
102 2 /* DATA ENTERED. WAIT FOR FREE */
103 2 CALL CLEAR(2);  
104 2 CALL DISPLAY(FREETEMP);  
105 2 CALL GRID(SUBMODESGRID);  
106 2 CALL REAFL.FRZ2(18);  
107 2 IF SVN = SUBMODE THEN RETURN;
108 2 GO TO F1;
109 2 E1;
110 2 /* FREE ENTERED BEFORE NEW */
111 2 IF DVRACORDDATAFILL.ENTRY.LSTEP+LWAPL+OFT) = 0 THEN RETURN;
112 2 F1;
113 2 CALL CLEAR(1);  
114 2 CALL DISPLAY(UPDATETAB);  
115 2 CALL GRID(SUBMODESGRID);  
116 2 CALL READL.UPDATE(18);  
117 2 IF SVNINDEX = 0 THEN GO TO TOPL;
118 2 END; /* OF UPDATE */
119 1 END; /* OF DVR3 */

MODULE INFORMATION:

CODE AREA SIZE = 0204H 5160
VARIABLE AREA SIZE = 0001H 10
MAXIMUM STACK SIZE = 0004H 40
240 LINES READ
0 PROGRAM ERRORS

END OF PL/M-80 COMPIATION
PL/N-90 COMPILER  INV

ISIS-II PL/N-90 V3.1 COMPILATION OF MODULE DNW
OBJECT MODULE PLACED IN :F10WAV,elope
COMPILER INVOKED BY: PLN000 (F1:01DNW, SRC DATE(INOV79) DEBUG

TITLE("DNW")
/* FORMAT-WAYPT-COORD, DNW-WAYPT-STATUS, DNWBACK сент */
1 DNW: DO;
2   WLIST INCLUDE(F1:COLIT.SRC)
3 1 DISPLAY(WAYPT : PROCEDURE EXTERNAL)
4 2 END;
5 1 CLEAR: PROCEDURE(ROM) EXTERNAL;
6 2 DECLARE ROM BYTE;
7 2 END;
8 1 DISPLAY: PROCEDURE(PTR) EXTERNAL;
9 2 DECLARE PTR ADDRESS;
10 2 END;
11 1 UPDATE: PROCEDURE(ROM,COL,BYTES,EXIST) EXTERNAL;
12 2 DECLARE (ROM,COL,BYTES) BYTE, EXIST ADDRESS;
13 2 END;
14 1 DISPLAY: DIGIT(B) EXTERNAL;
15 2 END;
16 1 GRID: PROCEDURE(PTR) EXTERNAL;
17 2 DECLARE PTR ADDRESS;
18 2 END;
19 1 READ: PROCEDURE(COMP) EXTERNAL;
20 2 DECLARE (COMP) ADDRESS;
21 2 END;
22 1 INSERT: PROCEDURE(NCHAR, SOURCEPTR, DESTPTR, DELMACH, DELMASK) EXTERNAL;
23 2 DECLARE (NCHAR, DELMACH) BYTE, (SOURCEPTR, DESTPTR, DELMASK) ADDRESS;
24 2 END;
25 1 PACK: WAYPT: PROCEDURE(COUNT, SOURCEPTR, DESTPTR) EXTERNAL;
26 2 DECLARE COUNT BYTE, (SOURCEPTR, DESTPTR) ADDRESS;
27 2 END;
28 1 UNPACK: WAYPT: PROCEDURE(COUNT, SOURCEPTR, DESTPTR) EXTERNAL;
29 2 DECLARE COUNT BYTE, (SOURCEPTR, DESTPTR) ADDRESS;
30 2 END;
31 1 CLEAR: WAYPT: PROCEDURE EXTERNAL;
32 2 END;
33 1 INVCOORD: DATAENTRY: PROCEDURE(WAYPTPTR, WAYPTTYPE) BYTE EXTERNAL;
34 2 DECLARE WAYPTPTR ADDRESS, WAYPTTYPE BYTE;
35 2 END;
36 1 /* EXTERNAL VARIABLES */
37 1 DECLARE
38   SNW BYTE EXTERNAL,
39   SINDEX BYTE EXTERNAL,
40   INBUFF(1) BYTE EXTERNAL,
41   DATAENTERED BYTE EXTERNAL,
42   TEMPBUFF(1) BYTE EXTERNAL,
43   SUPMAGBYTE BYTE EXTERNAL,
44   SUPDSEXBYTE BYTE EXTERNAL,
45   SHEROIDEIT(6) STRUCTURE(CH(3) BYTE) EXTERNAL,
46   DNWTEMPWAYPT STRUCTURE(
47   STATUS BYTE,
DECLARE WAYPTBASEADD ADDRESS;
(PACKEDWAYPT BASE WAYPTBASEADD)(10) STRUCTURE(CH(68) BYTE);
DECLARE WAYPT is COORD; /* MUST POINT TO AN UNPACKED WAYPOINT DATA BLOCK */

DECLARE WAYPT CHanged WAYPT coord STRUCTURE:
  STATUS BYTE,
  UTHMPSHEROID BYTE,
  UTMAZONE(2) BYTE,
  UTHMPEA(3) BYTE,
  UTHMVALyE(4) BYTE,
  LATITUDE(6) BYTE,
  LONGITUDE(7) BYTE,
  MAGVAR(8) BYTE;

DECLARE ERRORMED(* BYTE DATA("DATA NOT AVAILABLE")

PROCEDURE: FOTMATLATADON;

CALL INSERT(WAYPT, LATITUDE, TEMP_BUF(3), ,"120044")
CALL INSERT(WAYPT, LONGITUDE, TEMP_BUF(14), ,"090041")
TEMP_BUF(6), TEMP_BUF(18) = DEGASY;
TEMP_BUF(11), TEMP_BUF(22) = "";

END;

PROCEDURE: FORMATMUTH;

CALL MOVE(UTMPSHEROID TEXT (WAYPT, UTHMPSHEROID AND ORH), TEMP_BUF(3));
CALL INSERT(WAYPT, UTMZONE, TEMP_BUF(7), , "221044")
END;

IF (WAYPT STATUS) THEN DO;
  IF (WAYPT STATUS AND 2) > 0 THEN CALL FORMATMUTH
  ELSE DO;
    IF (WAYPT STATUS AND 2) > 0 THEN CALL FORMATLATADON
    ELSE CALL MOVE(, ERRORMSG, TEMP_BUF(3))
    END;
  END;
ELSE DO;
  IF (WAYPT STATUS AND 4) > 0 THEN CALL FORMATLATADON
  ELSE DO;
    IF (WAYPT STATUS AND 4) > 0 THEN CALL FORMATMUTH
    ELSE CALL MOVE(, ERRORMSG, TEMP_BUF(3))
    END;
  END;
END /* OF FOTMATWAYPTCOORD */
REJECT

73 1 DMY=AWAYPTSTATUS: PROCEDURE(TYPE) PUBLIC;
74 2 DECLARE TYPE BYTE; /* X = OPT, I = TGT */

75 2 DECLARE STATSTRU(1) BYTE DATA
RIO:, 'DAY, 'RO:, 'COL=2,'/3', '0);
76 2 DECLARE TYPESTRU(1) STRUCTURE(TEXT(7) BYTE) DATA
RIO:, 'COL=6,'OPT', 'I0,
RIO:, 'COL= 'TGT', '01;

77 2 DECLARE PAGESTATSHENABLE STRUCTURE!
SHMASK(7) BYTE,
SHENABLE(2) BYTE DATA
0, 0BH, 0BH, 0BH, 0BH, 0BH, 0BH, /* SW = 100..300..400..SUBS+3. STEP */
0, 0H, 0H;
78 2 DECLARE PAGESTATSHENABLE STRUCTURE
MODE BYTE, NUMCH BYTE, DELCHN BYTE, DELMARK ADDRESS,
ECHOCHR BYTE, ECHCOL BYTE, SHENABLE ADDRESS) DATA
SWITCH, 0, 0, 0, 0, 0, 0, PAGE12STATSHENABLE;
79 2 DECLARE PAGESTATSHENABLE STRUCTURE
SHMASK(7) BYTE,
SHVALUE(1) BYTE DATA
0, 0BH, 0BH, 0BH, 0BH, /* SW = 100..SUBS+3. STEP */
01H;
80 2 DECLARE PAGESTATSHENABLE STRUCTURE
MODE BYTE, NUMCH BYTE, DELCHN BYTE, DELMARK ADDRESS,
ECHOCHR BYTE, ECHCOL BYTE, SHENABLE ADDRESS) DATA
SWITCH, 0, 0, 0, 0, 0, PAGE12STATSHENABLE;
81 2 DECLARE PAGESIZE(3) BYTE DATA(3, 3, 1) /* HINDS ONE */
AWAYPTNUM BYTE,
AWAYPTOFFSET BYTE,
SHENABLE BYTE,
ACTIVEAWAYPT BYTE,
(PAGE.1,11) BYTE;

82 2 IF TYPE = OPT THEN AWAYPTBASE(4) = .AWAYPTDATA
83 2 ELSE AWAYPTBASE(4) = .AWAYPTDATA
84 2 SHENABLE = NOT (TYPE OR AWAYPTDESTINATION);
85 2 ACTIVEAWAYPT = SHR(AWAYPTDESTINATION, 1) AND C8H;
86 2 PAGE = 01;
87 2 AWAYPTOFFSET = 0;
88 2 DO FOREVER;
89 2 CALL CLEAR(OI);
90 2 CALL DISPLAY(AWAYPT);
91 3 CALL DISPLAY(.STATSTRU(TYPE));
92 3 CALL DISPLAY(.TYPESTRU(TYPE));
93 3 CALL DISPLAY(.STATSTRU(TYPE));
94 3 N = PAGESIZE(PAGE);
95 3 TEMPRBUF(0) = PAGE + 31H;
96 3 CALL UPDATE_LINE(1, 21, 1, TEMPRBUF); 
97 3 CALL CLEAR(2);
98 3 DO I = 0 TO N;
99 4 CALL CLEAR(TEMPBUF);
100 4 AWAYPTNUM = AWAYPTOFFSET + I;
101 4 TEMPRBUF lul = AWAYPTNUM + 30H;
102 4 IF SHENABLE AND (AWAYPTNUM = ACTIVEAWAYPT) THEN TEMPRBUF(2) = 'O'
CALL UNPACKEDWAYPT(34..PACKEDWAYPT(NWAYPTNUM)..INWAYPTWAYPT);
CALL FORMATTEDCOORD(INSTEPNWAYPT);
CALL UPDATELINE((1+2-0.24..TEMPHUF));
END;
108 3 IF PAGE < 3 THEN CALL READ..PAGE(STATICS); ELSE CALL READ..PAGE(STATICS);
111 3 IF SWY ~ SUBPOEM THEN RETURN;
113 3 IF SWY = STEPW THEN DO;
115 4 PAGE = (PAGE + 1) MOD 31;
116 4 NWAYPTOFFSET = PAGE + 1;
117 4 END;
118 3 ELSE DO;
119 4 NWAYPTNUM = NWAYPTOFFSET + SWINDEX;
120 4 IF INWAYPTDATAENTRY(34..INSTEPNWAYPT..PACKEDWAYPT(NWAYPTNUM)) = 1 THEN DO;
122 5 CALL PACKWAYPT(34..INSTEPNWAYPT..PACKEDWAYPT(NWAYPTNUM));
125 5 END;
128 3 END; /* IF DO FOREVER */
127 2 END; /* IF INWAYPTSTATUS */
DECLARE BACKUP$SHENABLE STRUCTURE(  
SHMASK(7) BYTE,  
SHVALUE(3) BYTE) DATA(  
0, 0, 0, 40H, 0, 0, 08H, /WIND DIR,MIND SPD,SPD,TRK,SMODE /O  
OAH, ZEH, 40H)  
DECLARE BACKUP$CB STRUCTURE(  
NODE BYTE, NVMCH BYTE, DEMMAC ADDRESS,  
ECHOLR BYTE, ECHOCOL BYTE, SHENABLE ADDRESS) DATA(  
SWITCH(0, 0, 0, 0, 0, BACKUP$SHENABLE1)  
DECLARE BACKUP$DATA(16, 4) STRUCTURE(  
NODE BYTE, NVMCH BYTE, DEMMAC ADDRESS,  
ECHOLR BYTE, ECHOCOL BYTE, SHENABLE ADDRESS) DATA(  
DIGIT(3), 0, 0, 2, 9, SUBMODESENABLE, /WIND DIRECTION /O  
DIGIT(2), 0, 0, 2, 9, SUBMODESENABLE, /WIND SPEED /O  
DIGIT(3), 0, 0, 2, 11, SUBMODESENABLE, /GROUND SPEED /O  
DIGIT(3), 0, 0, 2, 10, SUBMODESENABLE1, /TRACK ANGLE /O  
DECLARE BACKUP$ENTRY$TAB(4) STRUCTURE(16, 13) BYTE) DATA(  
NODE, ''MIND DIR '', 0,  
NODE, ''MIND SPD '', 0,  
NODE, ''GROUND SPD '', 0,  
NODE, ''TRACK ANGLE '', 0)  
DECLARE NUMDIGITS(4) BYTE DATA(3, 2, 3, 3)  
DECLARE DATAWORD(4) ADDRESS DATA(  
BMA$WIN$D,  
BMA$WIN$D$SPEED,  
BMA$GROUND$SPEED,  
BMA$TRACK$ANGLE)
PL/M-80 COMPILER  DNA

PAGE 7

136 2  BACKUPDATALNTRY: PROCEDURE;
137 3  DECLARE 1 BYTE;
138 3  CALL CLEAR(2);
139 3  CALL DISPLAY, BACKUPDATALNTRY(1);
140 3  CALL DISPLAYDIGITS;
141 3  CALL READ, BACKUPDATALNTRY(1);
142 3  IF SW = SN/DECODE THEN RETURN;
143 3  IF DATAENTERED AND (SW = ENTER#) THEN
144 3  CALL MOVE, BACKUPDATALNTRY(1), DATAADD(1);
145 3  END /\ OF BACKUPDATALNTRY /\;
146 3
147 2  DISPLAYSTATILINE: PROCEDURE;
148 3  DECLARE STATISTABLEAU(4) BYTE DATA(
149 3  "01", "02", "03", "04", "05", "06", "07", "08", "09");
150 3  DECLARE LINELOC(4) ACCESS DATA(
151 3  "TOP", "TOP", "TOP", "TOP");
152 3  DECLARE I BYTE;
153 3  CALL DISPLAY, STATISTABLEAU(1);
154 3  DO I = 1 TO 4;
155 3  CALL MOVE, BACKUPDATALNTRY, DATAADD(1), LINELOC(1);
156 3  END;
157 3  END /\ OF DISPLAYSTATILINE /\;
158 2
159 3  /* BEGIN BACKUP CODE */
160 3  DO FOREVER;
161 3  CALL CLEAR(1000);
162 3  CALL DISPLAYACTIVEENTRY;
163 3  CALL DISPLAYSTATILINE;
164 3  CALL DISPLAY, BACKUPDATALNTRY(1);
165 3  CALL READ, BACKUPDATALNTRY(1);
166 3  IF SW = SN/DECODE THEN RETURN;
167 3  CALL BACKUPDATALNTRY(SW, INDEX);
168 3  END /\ OF DO FOREVER /\;
169 2  END /\ OF DNA /;
170 1

MODULE INFORMATION:

CODE AREA SIZE = 0491H  11670
VARIABLE AREA SIZE = 000EH  140
MAXIMUM STACK SIZE = 000CH  130
333 LINES READ
0 PROGRAM ERRORS)
31 2 DECLARE WAYPT BASED WAYPTMAP STRUCTURE
   STATUS BYTE,
   UTM/SFEROID BYTE,
   UTM/ZONE(2) BYTE,
   UTM/ALPHA(3) BYTE,
   UTM/VALUE(8) BYTE,
   LAT(6) BYTE,
   LON(7) BYTE,
   NAVVAR(6) BYTE
32 2 DECLARE SPHERIODISTABLEAU() BYTE DATA(ROGD: 'DMV');
   ROM/SELECT SPHERID;
   ROMH/COL12: 'CLO', COL17: 'EVC',
   ROM4/COL12: 'CLO', COL17: 'BEO',
   ROM8/COL12: 'INO', COL17: 'AUX', 011
33 2 DECLARE SPHEROID/SHENABLE STRUCTURE
   SPHMAEK(7) BYTE,
   SPHVALU(6) BYTE) DATA
   0,0,0,0,0,0,0,0; # CLO,EVC,BEO,INO,AUX,SHENABLE,STEP 0/
   0,0,0,0,0,0,0,0; # S1H,SH,SH,SH,SH,SH,SH,SH;
34 2 DECLARE SPHERIOD/SHENABLE STRUCTURE
   NODE BYTE, NUMCH BYTE, DELMAC BYTE, DELMAE Mask ADDRESS,
   ECHODR BYTE, ECHO/COL BYTE, SHENABLE ADDRESS) DATA
   SWITCH, 0.0.0.0.0.0, SPHEROID/SHENABLE)
35 2 DECLARE ZONE/SC STRUCTURE
   NODE BYTE, NUMCH BYTE, DELMAC BYTE, DELMAE Mask ADDRESS,
   ECHODR BYTE, ECHO/COL BYTE, SHENABLE ADDRESS) DATA
   DIGIT, 2.0.0.2.11, SPHEROID/SHENABLE)
36 2 DECLARE ZONE/SC TABLEAU() BYTE DATA(ROGD: 'ENTER ZONE', 011)
37 2 DECLARE ALPHABET/SC TABLEAU() BYTE DATA(ROGD: 'ENTRY', 'GM',
38 2 DECLARE ALPHABET/SC STRUCTURE
   NODE BYTE, NUMCH BYTE, DELMAC BYTE, DELMAE Mask ADDRESS,
   ECHODR BYTE, ECHO/COL BYTE, SHENABLE ADDRESS) DATA
   DIGIT, 2.0.0.0.0.0, SPHEROID/SHENABLE)
39 2 DECLARE ALPHABET/SC STRUCTURE
   NODE BYTE, NUMCH BYTE, DELMAC BYTE, DELMAE Mask ADDRESS,
   ECHODR BYTE, ECHO/COL BYTE, SHENABLE ADDRESS) DATA
   DIGIT, 2.0.0.0.0.0, SPHEROID/SHENABLE)
40 2 DECLARE VALUE/SC TABLEAU() BYTE DATA(ROGD: 'ENTER VALUE', 011)
41 2 DECLARE VALUE/SC STRUCTURE
   NODE BYTE, NUMCH BYTE, DELMAC BYTE, DELMAE Mask ADDRESS,
   ECHODR BYTE, ECHO/COL BYTE, SHENABLE ADDRESS) DATA
   DIGIT, 2.0.0.0.0.0, SPHEROID/SHENABLE)
SELECT

/\ UTM SPHEROID SELECTION /\  
42 2 CALL CLEAR(01)
43 2 CALL DISPLAY(SPHEROID)
44 2 CALL DISPLAY(SPHERIOTABLEAU)
45 2 CALL DISPLAY(WAYPTSTYPE(WAYPTSTYPE))
44 2 CALL GRID(SPHERIOD(01))
47 2 CALL READ(SPHERIOD(01))
48 2 IF SW = SUBMODE THEN RETURN(01)
50 2 IF SW = STEPMS THEN RETURN(40H)
52 2 WAYPT.UTMSPHEROID = SYMINDEX;

/\ UTM ZONE ENTRY /\  
53 2 CALL UPDATE(LINE(0,4,3.,SPHERIODTEXT(WAYPT.UTMSPHEROID)))
54 2 CALL CLEAR(11)
55 2 CALL DISPLAY(LZONETABLEAU)
56 2 CALL DISPLAY(WAYPTSTYPE(WAYPTSTYPE))
57 2 CALL DISPLAY(DIGIT(31))
58 2 A:
59 2 CALL READ(LZONETABLEAU)
60 2 IF SW = SUBMODE THEN RETURN(01)
62 2 IF SW = STEPMS THEN RETURN(40H)
63 2 IF DATAENTERED AND (SW = ENTERWSW) THEN DO:
65 3 WAYPT.UTMZONE(0) = INBUF(01)
66 3 WAYPT.UTMZONE(1) = INBUF(11)
67 3 END;
68 2 ELSE GO TO A; /\ REJECT CENTER NOT PRECEDED BY DATA /\  

/\ UTM ALPHA ENTRY /\  
69 2 CALL CLEAR(11)
70 2 CALL UPDATE(LINE(0,7,2.,INBUF)) /\ PUT ZONE ON THE SCREEN /\  
71 2 CALL GRID(ALPHA(51H))
72 2 CALL DISPLAY(ALPHATABLEAU)
73 2 B:
74 2 CALL READ(ALPHA(51H))
75 2 IF SW = SUBMODE THEN RETURN(01)
76 2 IF SW = STEPMS THEN RETURN(40H)
77 2 IF DATAENTERED AND (SW = ENTERWSW) THEN DO:
79 2 CALL MOVE(3.,INBUF.,WAYPT.UTMVALUE)
80 2 ELSE GO TO B; /\ UTM VALUE ENTRY /\  
81 2 CALL UPDATE(LINE(0,9,3.,INBUF))
82 2 CALL CLEAR(11)
83 2 CALL DISPLAY(DIGIT(31H))
84 2 CALL DISPLAY(VALUE(248))
85 2 CALL DISPLAY(WAYPTSTYPE(WAYPTSTYPE))
86 2 C:
87 2 CALL READ(1.VALUE(31H))
88 2 IF SW = SUBMODE THEN RETURN(01)
89 2 IF SW = STEPMS THEN RETURN(40H)
91 2 IF DATAENTERED AND (SW = ENTERWSW) THEN DO:
93 3 CALL MOVE(0.,INBUF.,WAYPT.UTMVALUE)
94 3 WAYPT.STATUS = WAYPT.STATUS OR 21;
95 3 END;
96 2 ELSE GO TO C;
97 2 RETURN(11); /\ NORMAL RETURN - DATA ENTERED /\  
98 2 END; /\ OF UTMZONENTRY /\
DECLARE(E.J.W) BYTE;

PACK\&\#\#\#PT: PROCEDURE(COUNT, SOURCEPTR, DESTPTR) 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 ARSPFB. /
DECLARE COUNT BYTE;  /* NUMBER OF BYTES TO BE PACKED */
SOURCEPTR ADDRESS;
DESTPTR ADDRESS;
DECLARE (SOURCE BASED SOURCEPTR)(1) BYTE;
(DEST BASED DESTPTR)(1) BYTE;
J = 01;
COUNT = COUNT - 11;
DO I = 0 TO COUNT;
W = SOURCE(I);
DEST(J) = (DEST(J) AND OFH) OR (W AND OFH);
DEST(J+1) = (DEST(J+1) AND OFH) OR SHL(W,4);
J = J + 21;
END
END /* OF PACK\&\#\#\#PT */

UNPACK\&\#\#\#PT: PROCEDURE(COUNT, SOURCEPTR, DESTPTR) 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. /
DECLARE COUNT BYTE;  /* NUMBER OF BYTES TO BE RETRIEVED */
SOURCEPTR ADDRESS;
DESTPTR ADDRESS;
DECLARE (SOURCE BASED SOURCEPTR)(1) BYTE;
(DEST BASED DESTPTR)(1) BYTE;
J = 01;
COUNT = COUNT - 11;
DO I = 0 TO COUNT;
DEST(I) = (SOURCE(I) AND OFH) OR SHL(SOURCE(I+4),4);
J = J + 21;
END
END /* OF UNPACK\&\#\#\#PT */

END /* OF DMA5 DO */

MODULE INFORMATION:

CODE AREA SIZE = 040H 1025D
VARILABLE AREA SIZE = 0010H 11D
MAXIMUM STACK SIZE = 0006H 6D
264 LINES READ
0 PROGRAM ERROR(S)

END OF PL/M-80 COMPILATION
TITLE('DWAYAT')

DATA:

1
2 1 DECLARE H8BUF(1536) BYTE PUBLIC;
   8BUF(1536) BYTE PUBLIC;

3 1 DECLARE /* DW VARIABLES */
   DWDESTINATION BYTE PUBLIC;
   DWMODEPLAYMODE BYTE PUBLIC;
   /* DW INITIALIZATION REQUIRES THE FOLLOWING ITEMS TO BE CONTIGUOUS */
   DWHAIRNSPEED(3) BYTE PUBLIC;
   DWMICRONSHED(3) BYTE PUBLIC;
   DWTRACKANGLEERROR(3) BYTE PUBLIC;
   DWTRACKXANGLE(3) BYTE PUBLIC;
   DWMICROXTRACKANGLE(3) BYTE PUBLIC;
   DWREARW(3) BYTE PUBLIC;
   DWMAINTK0(3) BYTE PUBLIC;
   /* */
   DWSTATUS BYTE PUBLIC;
   DWMASTXYPTPTR ADDRESS PUBLIC;
   DWMASTXYPTPTR(34) BYTE PUBLIC;
   DWMOSTXYPTDATA(10) STRUCTURE(DIGITS(68) BYTE) PUBLIC AT(.R8BUF),
   DWMASTXYPTDATA(10) STRUCTURE(DIGITS(68) BYTE) PUBLIC AT(.R8BUF+68011)

4 1 END /* DW */

MODULE INFORMATION:

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

END OF PL/M-80 COMPILATION
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
<table>
<thead>
<tr>
<th>CONSTRAINTS ON DESIGN APPROACH</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXISTING BUILDING BLOCKS</td>
</tr>
<tr>
<td>• THE 48 x 96 LED MODULES</td>
</tr>
<tr>
<td>• MEMBRANE SWITCH</td>
</tr>
<tr>
<td>• DUAL IN LINE PACKAGES FOR ELECTRONIC DEVICES</td>
</tr>
<tr>
<td>CDU VOLUME ALLOWANCE</td>
</tr>
<tr>
<td>EASE OF MODIFICATION</td>
</tr>
<tr>
<td>• FUNCTION</td>
</tr>
<tr>
<td>• CONFIGURATION</td>
</tr>
</tbody>
</table>
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
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
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 straightforward 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

...
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

## 1.0 GENERAL DESCRIPTION

1.1 MECHANICAL DESCRIPTION  
1.2 TIMING DIAGRAMS  
1.3 INPUTS AND OUTPUTS  
1.4 POWER REQUIREMENTS  

## 2.0 DATA LINK INTERFACE

## 3.0 RECEIVER OPERATION

3.1 EXTERNAL CLOCK (E16)  
3.2 RECEIVER THRESHOLD ADJUST (E20, E30)  
3.3 RESET (E25)  
3.4 DATA READY (E54)  
3.5 RECEIVED SYNC TYPE (E59)  
3.6 DATA ENABLES (E21, E49)  
3.7 PATTERN ERROR (E51)  
3.8 PARITY ERROR (E50)  
3.9 FLAG AND ECM (E46, E60)  
3.10 SYNC DETECT (E57)  
3.11 BUS ACTIVE (E56)  
3.12 RECEIVED NRZ AND RECEIVED CLOCK (E53, E55)

## 4.0 TRANSMITTER OPERATION

4.1 TRANSMITTER A SELECT (E10)  
4.2 ENABLE SHUTDOWN (E52)  
4.3 TRANSMITTER TIMEOUT (E40)  
4.4 SERIAL MODE SELECT (E23)  
4.5 TRANSMIT COMMAND (26)
<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.6</td>
<td>TRANSMIT SYNC TYPE (E21)</td>
<td>19</td>
</tr>
<tr>
<td>4.7</td>
<td>PARALLEL DATA LOAD 1 AND 2 (E17, E38)</td>
<td>20</td>
</tr>
<tr>
<td>4.8</td>
<td>TRANSMIT NRZ (E27)</td>
<td>20</td>
</tr>
<tr>
<td>4.9</td>
<td>ONE MEGAHERTZ CLOCK (E14)</td>
<td>20</td>
</tr>
<tr>
<td>4.10</td>
<td>DATA REQUEST (E28)</td>
<td>21</td>
</tr>
<tr>
<td>Figure</td>
<td>Title</td>
<td>Page</td>
</tr>
<tr>
<td>--------</td>
<td>------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>1</td>
<td>MTI-110 Block Diagram</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>Received Data Timing</td>
<td>7</td>
</tr>
<tr>
<td>3</td>
<td>Transmitted Data Timing</td>
<td>8</td>
</tr>
<tr>
<td>4</td>
<td>Receiver Input External Strapping Options</td>
<td>12</td>
</tr>
<tr>
<td>5</td>
<td>MTI-110 Receiver Input Transformer Connections</td>
<td>13</td>
</tr>
<tr>
<td>6</td>
<td>Recommended Circuit for Changing Receiver Serial Word Length</td>
<td>17</td>
</tr>
<tr>
<td>Plate M954</td>
<td>Multiplex Terminal Unit MTI-110</td>
<td>3</td>
</tr>
<tr>
<td>Plate M956</td>
<td>MTI-110 Multiplex Terminal Unit (Disassembled)</td>
<td>4</td>
</tr>
<tr>
<td>AN-MTI-001</td>
<td>MTI-110 Outline Drawing</td>
<td>5</td>
</tr>
<tr>
<td>Table I</td>
<td>MTI-110 Pin Function List</td>
<td>9</td>
</tr>
<tr>
<td>Table II</td>
<td>MTI-110 Non-TTL Pin Functions</td>
<td>10</td>
</tr>
</tbody>
</table>
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-MT-.001.
4.00 ± 0.01

0.40 ± 0.010

0.375 ± 0.020

0.125 ± 0.010

0.300 ± 0.010

2.750 ± 0.010

3.00 ± 0.01

3.750 ± 0.010

≈ 2-56NC-2 x 0.15 DEEP
4 HOLES

31EQ SPØ

0.100 ± 0.005 = 3.100

TOL NON-ACCUM

0.025 SQ 62 PLACES

29EQ SPØ

0.100 ± 0.005 = 2.900

TOL NON-ACCUM

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

17981
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 ±5%.
## TABLE I

### MTI 110 Pin Function List

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

MTI-110 NON-TTL PIN FUNCTIONS

<table>
<thead>
<tr>
<th>Function</th>
<th>Pin Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>data link B (+)</td>
<td>1</td>
</tr>
<tr>
<td>data link B (-)</td>
<td>2</td>
</tr>
<tr>
<td>receiver input B (LS+)</td>
<td>3</td>
</tr>
<tr>
<td>receiver input B (SS)</td>
<td>4</td>
</tr>
<tr>
<td>+12 volts input</td>
<td>12</td>
</tr>
<tr>
<td>-12 volts input</td>
<td>19</td>
</tr>
<tr>
<td>negative threshold adjust</td>
<td>20</td>
</tr>
<tr>
<td>receiver input B</td>
<td>22</td>
</tr>
<tr>
<td>receiver input A</td>
<td>24</td>
</tr>
<tr>
<td>receiver input B (LS-)</td>
<td>29</td>
</tr>
<tr>
<td>positive threshold adjust</td>
<td>30</td>
</tr>
<tr>
<td>+5 volts input</td>
<td>32</td>
</tr>
<tr>
<td>data link A (+)</td>
<td>33</td>
</tr>
<tr>
<td>data link A (-)</td>
<td>34</td>
</tr>
<tr>
<td>receiver input A (LS+)</td>
<td>35</td>
</tr>
<tr>
<td>receiver input A (SS)</td>
<td>36</td>
</tr>
<tr>
<td>transmitter power ground</td>
<td>38</td>
</tr>
<tr>
<td>receiver input A (LS-)</td>
<td>58</td>
</tr>
<tr>
<td>logic ground</td>
<td>62</td>
</tr>
</tbody>
</table>
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
<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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 biphase 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 +0.075V to +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 biphase 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
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 out-
put. 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 com-
mand 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.
ATTACHMENT A

ENGINEERING DESIGN TEST PLAN

ECOM CDU DISPLAY AND CONTROL FORMATS

The final formats for the ECOM CDU are detailed in 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

<table>
<thead>
<tr>
<th>Select</th>
<th>Display</th>
</tr>
</thead>
<tbody>
<tr>
<td>VHF</td>
<td>Figure 2</td>
</tr>
<tr>
<td>UHF</td>
<td>Figure 3</td>
</tr>
<tr>
<td>ADF</td>
<td>Figure 4</td>
</tr>
<tr>
<td>CNV</td>
<td>Figure 5</td>
</tr>
<tr>
<td>DNV</td>
<td>Figure 6</td>
</tr>
<tr>
<td>IFF</td>
<td>Figure 7</td>
</tr>
</tbody>
</table>
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 Checkpoint Formats only. Activation of [STEP] will cause the next channel number and frequency to be available for change if needed.
<table>
<thead>
<tr>
<th>VHF * 42.75</th>
<th>DNV * DNV* NAV</th>
</tr>
</thead>
<tbody>
<tr>
<td>T/R</td>
<td>CKPT 4</td>
</tr>
<tr>
<td></td>
<td>RGE 163.5 KM</td>
</tr>
<tr>
<td></td>
<td>BRG 273</td>
</tr>
<tr>
<td></td>
<td>TTG 50.5M</td>
</tr>
<tr>
<td>UHF 118.375</td>
<td>IFF * 123AC4A</td>
</tr>
<tr>
<td>T/R</td>
<td>ADF FAIL</td>
</tr>
<tr>
<td>CNV 110.95</td>
<td></td>
</tr>
<tr>
<td>PP 15SUP14178409</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PRI</th>
<th>IFF EMER</th>
<th>ZERO CODE</th>
<th>OFF</th>
<th>ON</th>
</tr>
</thead>
<tbody>
<tr>
<td>SUB MODE</td>
<td>RPLY</td>
<td>GARD</td>
<td>LAST</td>
<td>STEP</td>
</tr>
</tbody>
</table>

**Figure 1.** Primary Format CNI CDU

CAPABLE OF BLINKING ON FINAL MODEL.
<table>
<thead>
<tr>
<th>Switch</th>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>From: VHF</td>
<td>Figure 1</td>
</tr>
<tr>
<td>[CHAN SEL]</td>
<td>Figure 2.A</td>
</tr>
<tr>
<td>[MAN FREQ]</td>
<td>Figure 2.B</td>
</tr>
<tr>
<td>[PRST CHAN]</td>
<td>Figure 2.C</td>
</tr>
<tr>
<td>[STAT PAGE]</td>
<td>Figure 2.D</td>
</tr>
<tr>
<td>[T/R]</td>
<td>Figure 2</td>
</tr>
<tr>
<td>[T/R + GARD]</td>
<td>T/R + G will display on second line</td>
</tr>
<tr>
<td>[HOM]</td>
<td>Figure 2</td>
</tr>
<tr>
<td>[RETRAN]</td>
<td>RETRAN will display in top line</td>
</tr>
<tr>
<td>[TEST]</td>
<td>Figure 2</td>
</tr>
<tr>
<td></td>
<td>OK or FAIL will display on second line</td>
</tr>
</tbody>
</table>

ON or OFF will display in second line under VHF for indication of ON/OFF status.
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.
Figure 2.B. Manual Frequency Select Format

Switch Format
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.
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.

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.
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.
Figure 2.E. Stat Page
<table>
<thead>
<tr>
<th>UHF</th>
<th>CH-6</th>
<th>118.315</th>
</tr>
</thead>
<tbody>
<tr>
<td>OFF</td>
<td>T/R</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Switch</th>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>From:</td>
<td>Format</td>
</tr>
<tr>
<td>UHF</td>
<td>Figure 3.A</td>
</tr>
<tr>
<td>CHAN SEL</td>
<td>Figure 3.B</td>
</tr>
<tr>
<td>MAN FREQ</td>
<td>Figure 3.C</td>
</tr>
<tr>
<td>PRST CHAN</td>
<td>Figure 3.D</td>
</tr>
<tr>
<td>STAT PAGE</td>
<td></td>
</tr>
<tr>
<td>T/R</td>
<td>Figure 3</td>
</tr>
<tr>
<td>T/R + GARD</td>
<td></td>
</tr>
<tr>
<td>ADF</td>
<td>Figure 3</td>
</tr>
<tr>
<td>GARD</td>
<td>Figure 3</td>
</tr>
<tr>
<td>SQL ON</td>
<td>Figure 3</td>
</tr>
<tr>
<td>SQL OFF</td>
<td>Figure 3</td>
</tr>
<tr>
<td>TONE</td>
<td>Figure 3</td>
</tr>
<tr>
<td>(Momentary)</td>
<td></td>
</tr>
</tbody>
</table>

**Figure 3. UHF Submode Format**

- T/R will display in first line
- T/R & GARD will display in first line
- ADF will display in first line
- GARD will display in first line
- SQL will display in second line
- SQL will be erased from second line
- A tone will be heard in the headset.
Figure 3.A. UHF Channel Select Format

Switch | Format
---|---
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 its 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.
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.
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. VHF preset channel operations are the same as those used on VHF procedure shown on page 8.
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.
Figure 3.E Stat Page
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.
Figure 4.A. Channel Select Format

<table>
<thead>
<tr>
<th>Switch</th>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>From:</td>
<td>[CHAN SEL]</td>
</tr>
<tr>
<td>To:</td>
<td>[E]</td>
</tr>
</tbody>
</table>

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.
Figure 4. B. Frequency Select Format

Switch | Format
--- | ---
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. \([\text{MAN FREQ}]\) switch is not active in this format.
Figure 4.C. Preset Channel

Switch  Format
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.
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.
<table>
<thead>
<tr>
<th>CHAN SEL</th>
<th>MAN FREQ</th>
<th>PRST CHAN</th>
<th>STAT PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>MB VOL</td>
<td>↑</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NAV VOL</td>
<td>↓</td>
<td>TEST</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>MB HI</th>
<th>MB LO</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hi</td>
<td>Lo</td>
</tr>
</tbody>
</table>

**Figure 5. CNV Submode Format**

**Switch**

**From:** CNV

**To:**

- **CHAN SEL**
- **MAN FREQ**
- **PRST CHAN**
- **STAT PAGE**
- **MB VOL**
- **MB HI**
- **MB LO**
- **NAV VOL**
- **TEST**

<table>
<thead>
<tr>
<th>[MB HI]</th>
<th>Activation of [MB HI] will place the receiver in the marker beacon high sensitivity mode and display &quot;MB Hi&quot; in the second line of the display.</th>
</tr>
</thead>
<tbody>
<tr>
<td>[MB LO]</td>
<td>Activation of [MB LO] will place the receiver in the marker beacon low sensitivity mode and display &quot;MB LO&quot; in the second line of the display.</td>
</tr>
<tr>
<td>[TEST]</td>
<td>External test equipment is needed and the test indications are displayed on external equipment. This is a momentary switch.</td>
</tr>
<tr>
<td>[MB VOL]</td>
<td>Selection of [MB VOL] will allow control of the volume by use of the up or down arrow symbol keys.</td>
</tr>
<tr>
<td>[NAV VOL]</td>
<td>Selection of [NAV VOL] will allow control of the volume by use of the up or down arrow symbol keys.</td>
</tr>
</tbody>
</table>
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.
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.
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.

---

<table>
<thead>
<tr>
<th>CNV</th>
<th>C-4</th>
<th>MB HI</th>
</tr>
</thead>
<tbody>
<tr>
<td>ON</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CH-__</th>
<th>F-<strong>-</strong></th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PRST CHAN</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td></td>
<td>7</td>
<td>8</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td></td>
<td>C</td>
<td>Ø</td>
<td>E</td>
</tr>
</tbody>
</table>

**Figure 5.C. Preset Channel Format**

<table>
<thead>
<tr>
<th>Switch</th>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>From: PRST CHAN</td>
<td>Figure 5</td>
</tr>
<tr>
<td>To: NUMERIC + E</td>
<td>Figure 5</td>
</tr>
</tbody>
</table>

**From:**

**To:**

---

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

23
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 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.
THIS PAGE LEFT BLANK ON PURPOSE
## Figure 6. DNV Submode Format

<table>
<thead>
<tr>
<th>Switch</th>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>From:</td>
<td>DNV</td>
</tr>
<tr>
<td>To:</td>
<td>DNV</td>
</tr>
</tbody>
</table>

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.
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.
Figure 6.B. CKPT Entry Format

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 MHE.
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.

Switch | Format
---|---
[CKPT] | Figure 6
[UTM] | Figure 6.B.1
[L/L] | Figure 6.B.2

From: (CKPT) | To: [UTM] [L/L]

<table>
<thead>
<tr>
<th>DNV</th>
<th>NAV</th>
<th>CKPT 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>CKPT</td>
<td>COORD</td>
<td>1 2 3</td>
</tr>
<tr>
<td>CKPT</td>
<td>L/L</td>
<td>4 5 6</td>
</tr>
<tr>
<td>UTM</td>
<td>L/L</td>
<td>7 8 9</td>
</tr>
<tr>
<td>C</td>
<td>Ø</td>
<td>E</td>
</tr>
</tbody>
</table>
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.
This format is used to enter the first two numerics (ZONE) in the UTM Coordinate System.

1. Enter two numerics (Legal entry 1-40). 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.
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.

<table>
<thead>
<tr>
<th>Switch</th>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>From:</td>
<td>[E]</td>
</tr>
<tr>
<td>Figure 6.B.1.1</td>
<td></td>
</tr>
<tr>
<td>To:</td>
<td>[E]</td>
</tr>
<tr>
<td>Figure 6.B.1.1</td>
<td></td>
</tr>
</tbody>
</table>

Figure 6.B.1.2. UTM Alpha entry Format
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.
<table>
<thead>
<tr>
<th>DNV</th>
<th>NAV</th>
<th>CKPT 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>E__E</td>
<td>1</td>
<td>2  3</td>
</tr>
<tr>
<td>CKPT</td>
<td>4</td>
<td>5  6</td>
</tr>
<tr>
<td>UTM</td>
<td>7</td>
<td>8  9</td>
</tr>
<tr>
<td>C</td>
<td>Ø</td>
<td>E</td>
</tr>
</tbody>
</table>

Figure 6.B.1.4. UTM Numeric Entry Format
Figure 6.B.2. L/L Coordinate Entry Format

Switch | Format
---|---
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.
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.
**Figure 6.B.2.2. Longitude Direction Entry Format**

<table>
<thead>
<tr>
<th>Switch</th>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>From:</td>
<td>[E]</td>
</tr>
<tr>
<td>To:</td>
<td>[E]</td>
</tr>
</tbody>
</table>

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


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

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

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.
### Figure 6.C. Target Entry Format

<table>
<thead>
<tr>
<th>DNV</th>
<th>NAV</th>
<th>TGT 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>TGT—</td>
<td>COORD—</td>
<td>1 2 3</td>
</tr>
<tr>
<td>FRZE</td>
<td></td>
<td>4 5 6</td>
</tr>
<tr>
<td>UTM</td>
<td>L/L</td>
<td>7 8 9</td>
</tr>
<tr>
<td>TGT</td>
<td>TGT STAT</td>
<td>C ∅ E</td>
</tr>
</tbody>
</table>

**Switch**

From: [TGT]

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.
<table>
<thead>
<tr>
<th>Switch</th>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>From:</td>
<td>[NAV STAT] Figure 6</td>
</tr>
<tr>
<td>To:</td>
<td>[SUB] Figure 6</td>
</tr>
</tbody>
</table>

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].
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].
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.
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.
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.
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.
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.
Figure 6.G. Checkpoint Status Display

<table>
<thead>
<tr>
<th>Switch</th>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>From: [CKPT STAT]</td>
<td>Figure 6</td>
</tr>
<tr>
<td>To: [CKPT 0] thru [CKPT 9]</td>
<td>Figure 6.B</td>
</tr>
</tbody>
</table>

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.
Figure 6.H. Update Formats

Switch | Format
--- | ---
From: | Figure 6
[UP DATE] | 
To: | Figure 6. H
[FRZE] | Figure 6
[UP DATE] | 

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, [UPDATE] 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 [UPDATE] 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 [UPDATE]. The display will return to Figure 6.
Figure 6.1 Target Status Display

Switch          Format
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.
<table>
<thead>
<tr>
<th>IFF</th>
<th>NORM</th>
</tr>
</thead>
<tbody>
<tr>
<td>ON</td>
<td>M1</td>
</tr>
</tbody>
</table>

- **M-1**: M-2, M-3A, STAT
- **M-4**: NORM, STBY, ANT
- **M-C**: RAD, TEST, TEST

Figure 7. IFF Submode

<table>
<thead>
<tr>
<th>Switch</th>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>From:</td>
<td>IFF</td>
</tr>
<tr>
<td>To:</td>
<td>M-1</td>
</tr>
<tr>
<td></td>
<td>M-2</td>
</tr>
<tr>
<td></td>
<td>M-3A</td>
</tr>
<tr>
<td></td>
<td>M-4</td>
</tr>
<tr>
<td></td>
<td>ANT</td>
</tr>
<tr>
<td></td>
<td>M-C</td>
</tr>
<tr>
<td></td>
<td>NORM</td>
</tr>
<tr>
<td></td>
<td>STBY</td>
</tr>
<tr>
<td></td>
<td>RAD</td>
</tr>
<tr>
<td></td>
<td>TEST</td>
</tr>
</tbody>
</table>

"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.
### Figure 7.A. M-1 Code Format

<table>
<thead>
<tr>
<th>Switch</th>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>From:</td>
<td>[M-1]</td>
</tr>
<tr>
<td>To:</td>
<td>[E]</td>
</tr>
</tbody>
</table>

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.
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**.

### Figure 7.B. M-2 Code Format

<table>
<thead>
<tr>
<th>IFF</th>
<th>NORM</th>
</tr>
</thead>
<tbody>
<tr>
<td>ON</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ENT CODE</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>M-2</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td></td>
<td>Ø</td>
<td>E</td>
</tr>
</tbody>
</table>

**Switch**  
From: (M-2) (GARD)  
To: E  

**Format**  
Figure 7  
Figure 7
**Figure 7-C. M-3A Code Format**

<table>
<thead>
<tr>
<th>IFF</th>
<th>NORM</th>
</tr>
</thead>
<tbody>
<tr>
<td>ON</td>
<td>M3A</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ENT CODE</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>M-3A</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>Ø</td>
<td>E</td>
<td></td>
</tr>
</tbody>
</table>

**Switch**

- **From:** M-3A
- **To:** E

**Format**

From: Figure 7

To: 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.
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.
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.
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.