United States Patent [19]

Martell

[11] **3,887,903**

[45] **June 3, 1975**

[54] **INTERACTIVE MAN-MACHINE METHOD AND SYSTEM FOR GRADING PATTERN PIECES AND FOR PRODUCING AN APPAREL MARKER**

[75] Inventor: **Charles Ronald Martell,** Dallas, Tex.

[73] Assignee: **Camsco, Inc.,** Richardson, Tex.

[22] Filed: **Aug. 29, 1973**

[21] Appl. No.: **392,651**

[52] **U.S. Cl.** ..... **340/172.5;** 340/324 AD

[51] **Int. Cl.** **G06f 3/14**

[58] **Field of Search** ..... 340/172.5, 324 AD; 235/150

[56] **References Cited**

UNITED STATES PATENTS

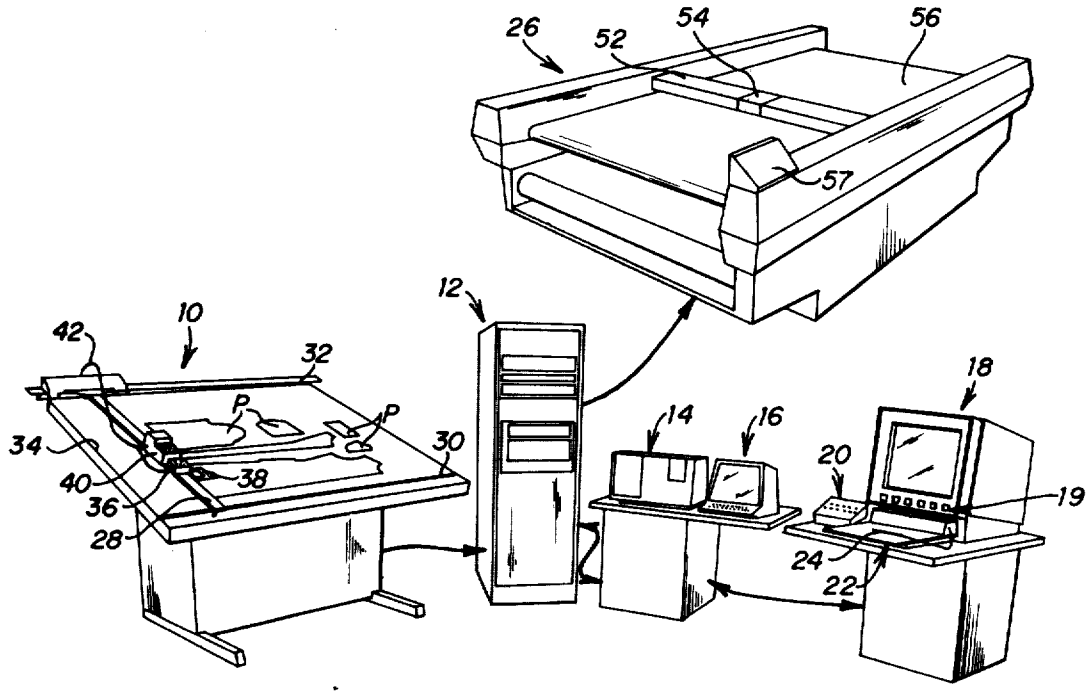
3,391,392	7/1968	Doyle	340/172.5
3,473,157	10/1969	Little et al.	340/172.5
3,596,068	7/1971	Doyle	235/150
3,621,214	11/1971	Romney et al.	340/172.5 X
3,720,948	3/1973	Eichen	340/172.5

Primary Examiner—Harvey E. Springborn
Attorney, Agent, or Firm—Richards, Harris & Medlock

[57] **ABSTRACT**

The specification discloses a technique for laying out an apparel pattern marker by efficiently arranging a plurality of apparel pattern pieces within a predetermined marker area. The system includes a digitizer for inputting digital coordinate data representative of a plurality of apparel pattern pieces. A central controller receives and stores the coordinate data and upon command, applies predetermined grading rules to the pattern pieces. The graded pattern pieces may be displayed upon a cathode ray tube display and checked by the operator. To produce a pattern marker, the operator faces the cathode ray tube display on which is displayed miniature representations of the marker area and the pieces comprising the desired style and sizes. The operator manually operates a stylus on a data tablet which is interconnected with the central controller and the cathode ray tube display. The operator further operates a multi-button function box in conjunction with movement of the stylus, in order to arrange the piece reproductions within the marker area on the cathode ray tube. When the desired miniature marker layout has been produced on the cathode ray tube display, the operator may cause the full sized apparel pattern marker to be automatically plotted. The pattern marker may then be utilized to cut cloth to enable the fabrication of a large number of garments of the desired style and sizes.

30 Claims, 56 Drawing Figures



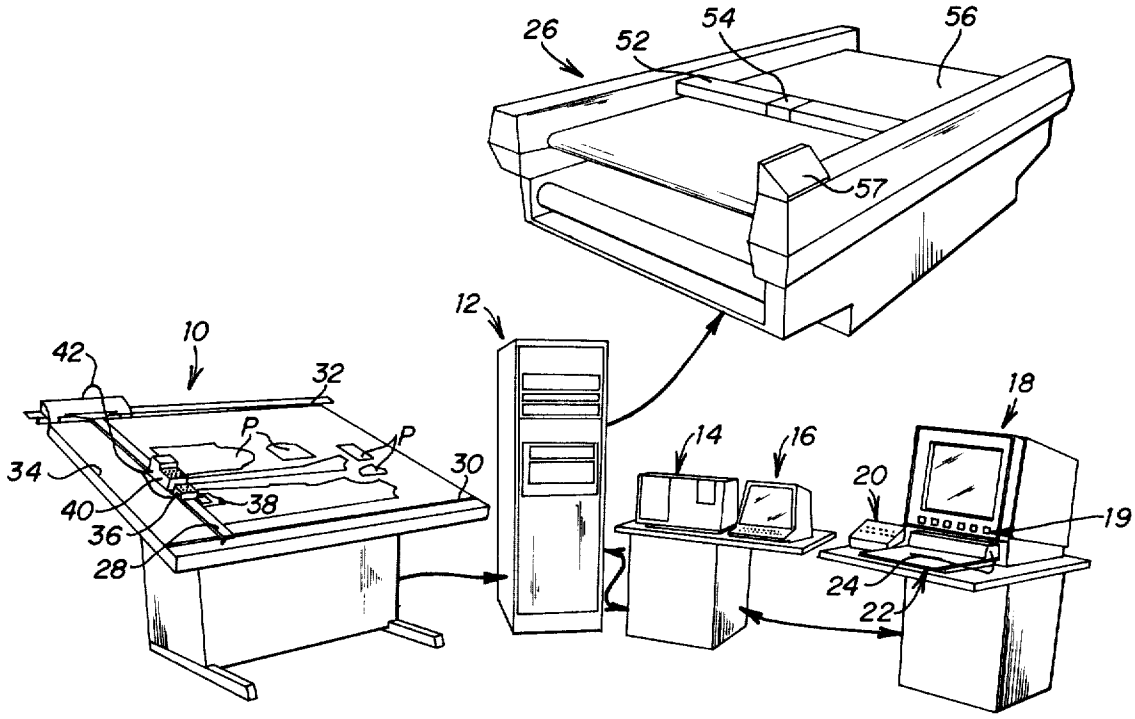

26
52
54
56
57
12
10
42
32
18
P
P
34
30
14
16
20
19
40
38
36
28
24
22

FIG. 1

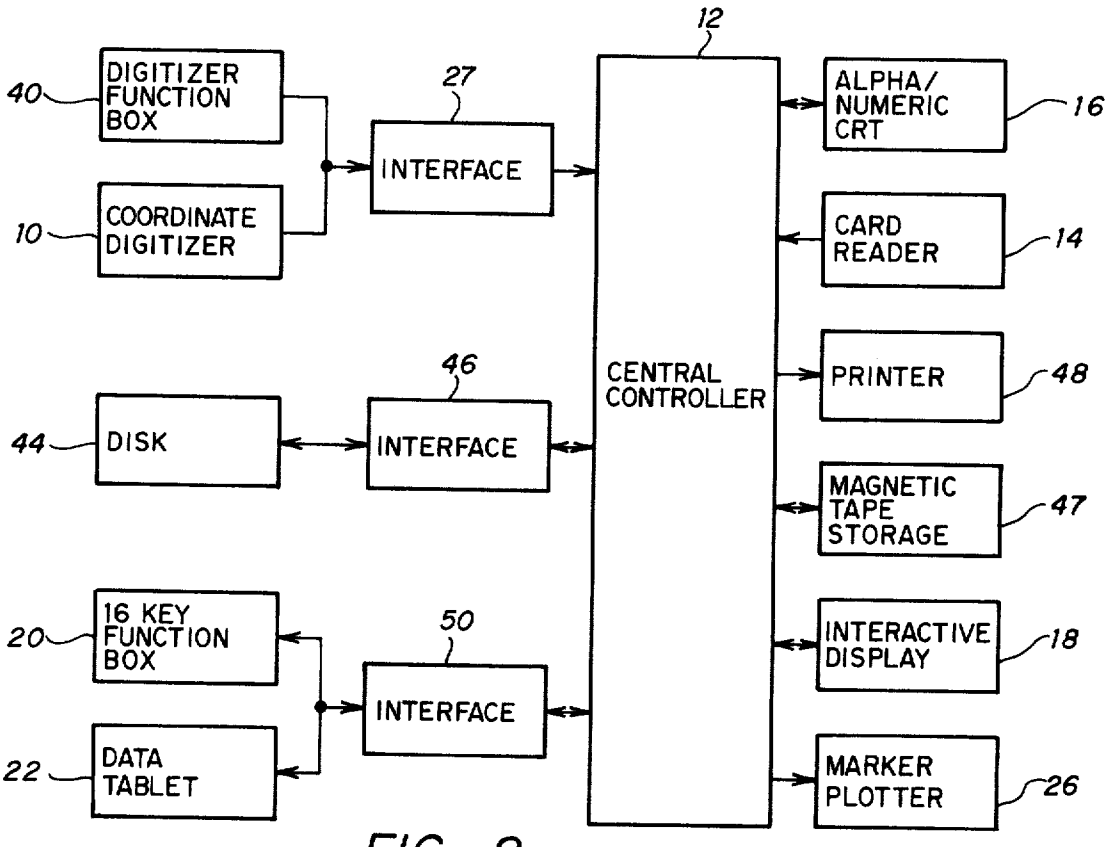

12
40
DIGITIZER FUNCTION BOX
27
INTERFACE
10
COORDINATE DIGITIZER
ALPHA/ NUMERIC CRT
16
CARD READER
14
CENTRAL CONTROLLER
46
44
DISK
INTERFACE
PRINTER
48
MAGNETIC TAPE STORAGE
47
20
16 KEY FUNCTION BOX
50
INTERACTIVE DISPLAY
18
INTERFACE
22
DATA TABLET
MARKER PLOTTER
26

FIG. 2

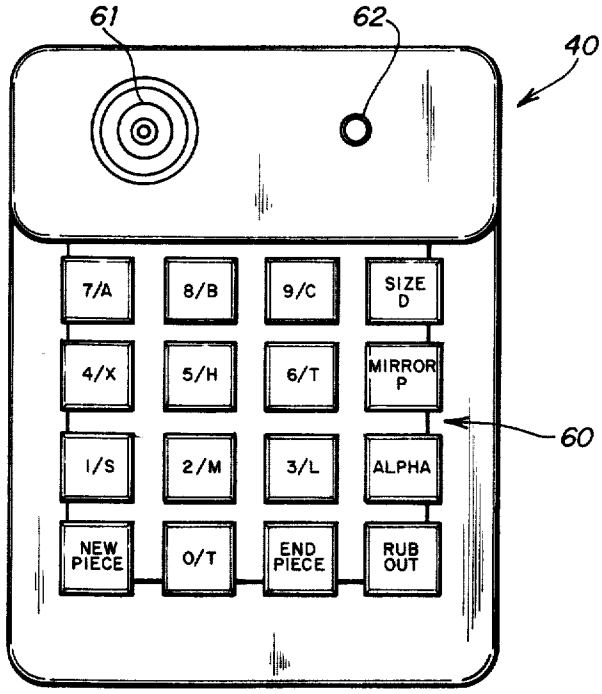

61
62
40
7/A
8/B
9/C
SIZE D
4/X
5/H
6/T
MIRROR P
1/S
2/M
3/L
ALPHA
NEW PIECE
0/T
END PIECE
RUB OUT
60

FIG. 3

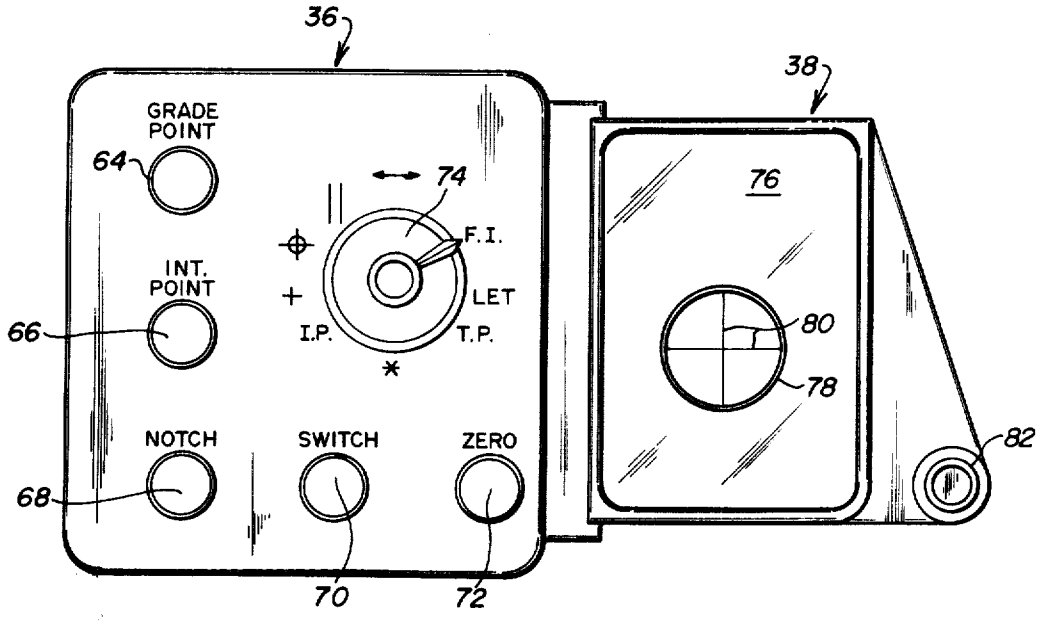

36
38
GRADE POINT
64
74
F.I.
LET
INT. POINT
66
I.P.
T.P.
76
80
78
82
NOTCH
SWITCH
ZERO
68
70
72

FIG. 4

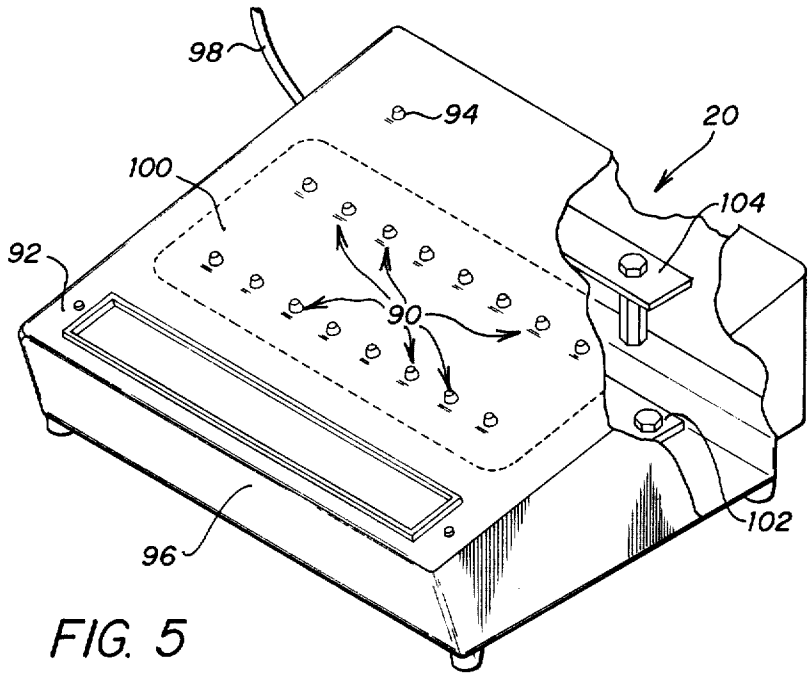


FIG. 5

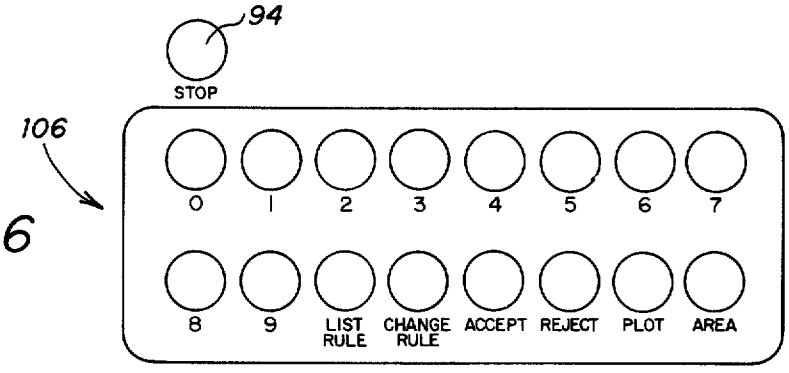


FIG. 6

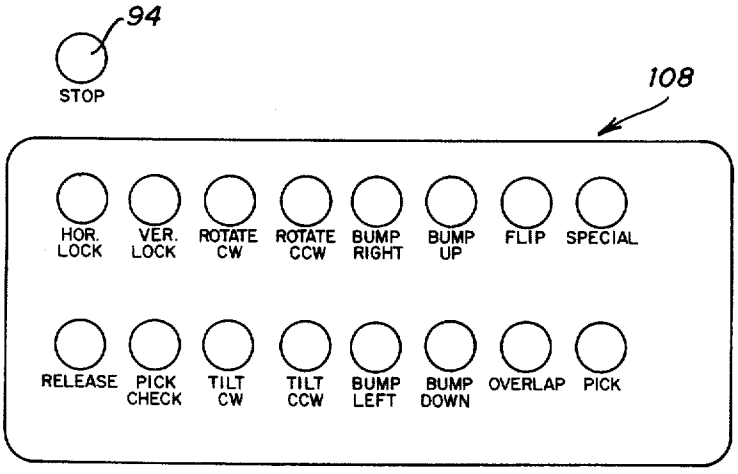


FIG. 7

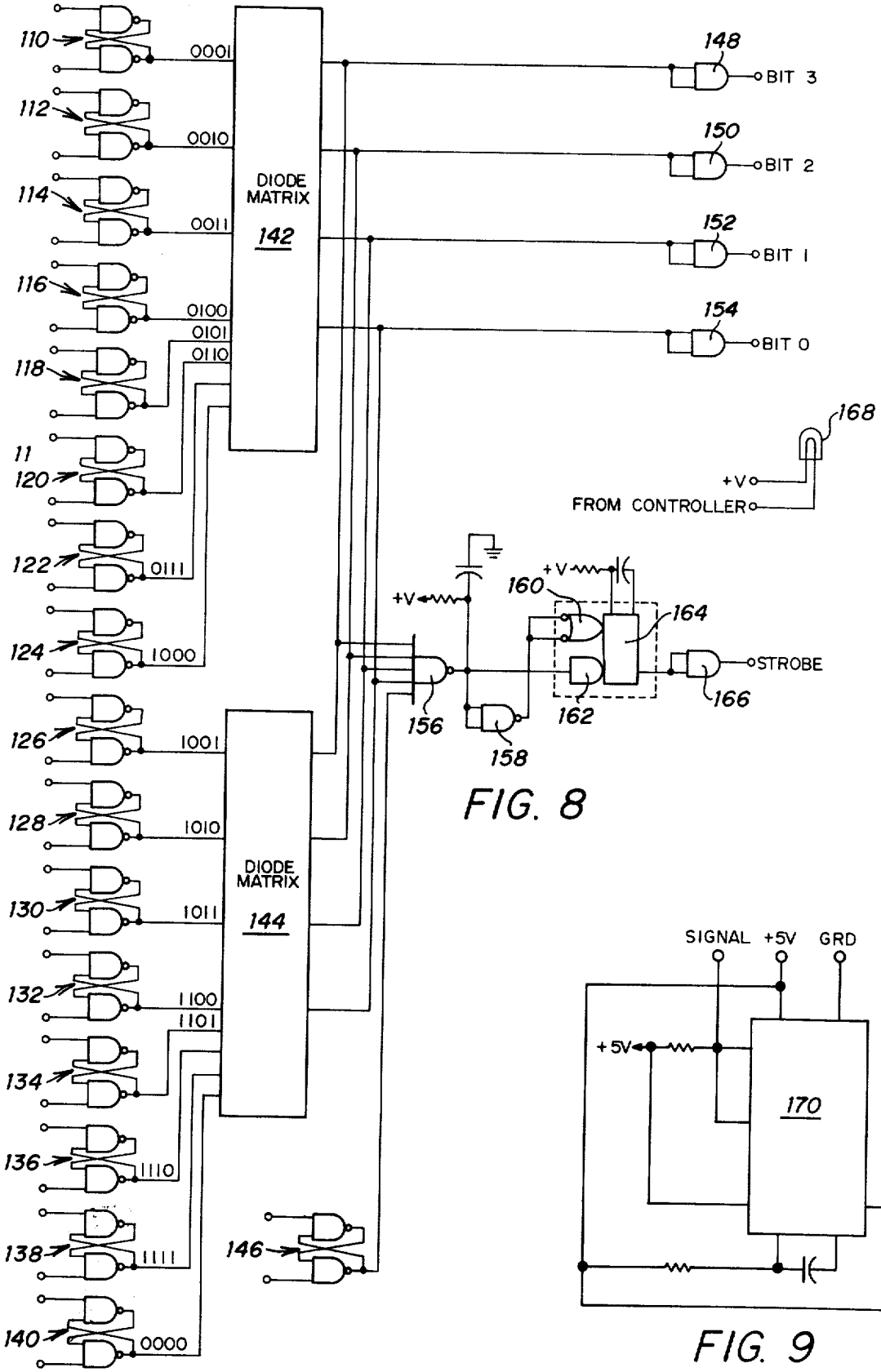

110
112
114
116
118
11
120
122
124
126
128
130
132
134
136
138
140
146
0001
0010
0011
0100
0101
0110
0111
1000
1001
1010
1011
1100
1101
1110
1111
0000
DIODE MATRIX 142
DIODE MATRIX 144
148
150
152
154
BIT 3
BIT 2
BIT 1
BIT 0
168
+V
FROM CONTROLLER
+V
160
164
STROBE
166
156
158
162

FIG. 8

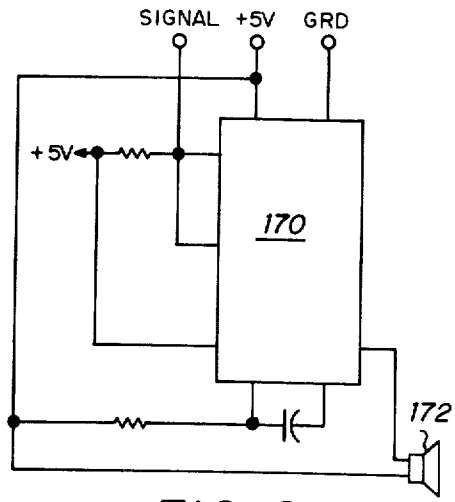

SIGNAL
+5V
GRD
+5V
170
172

FIG. 9

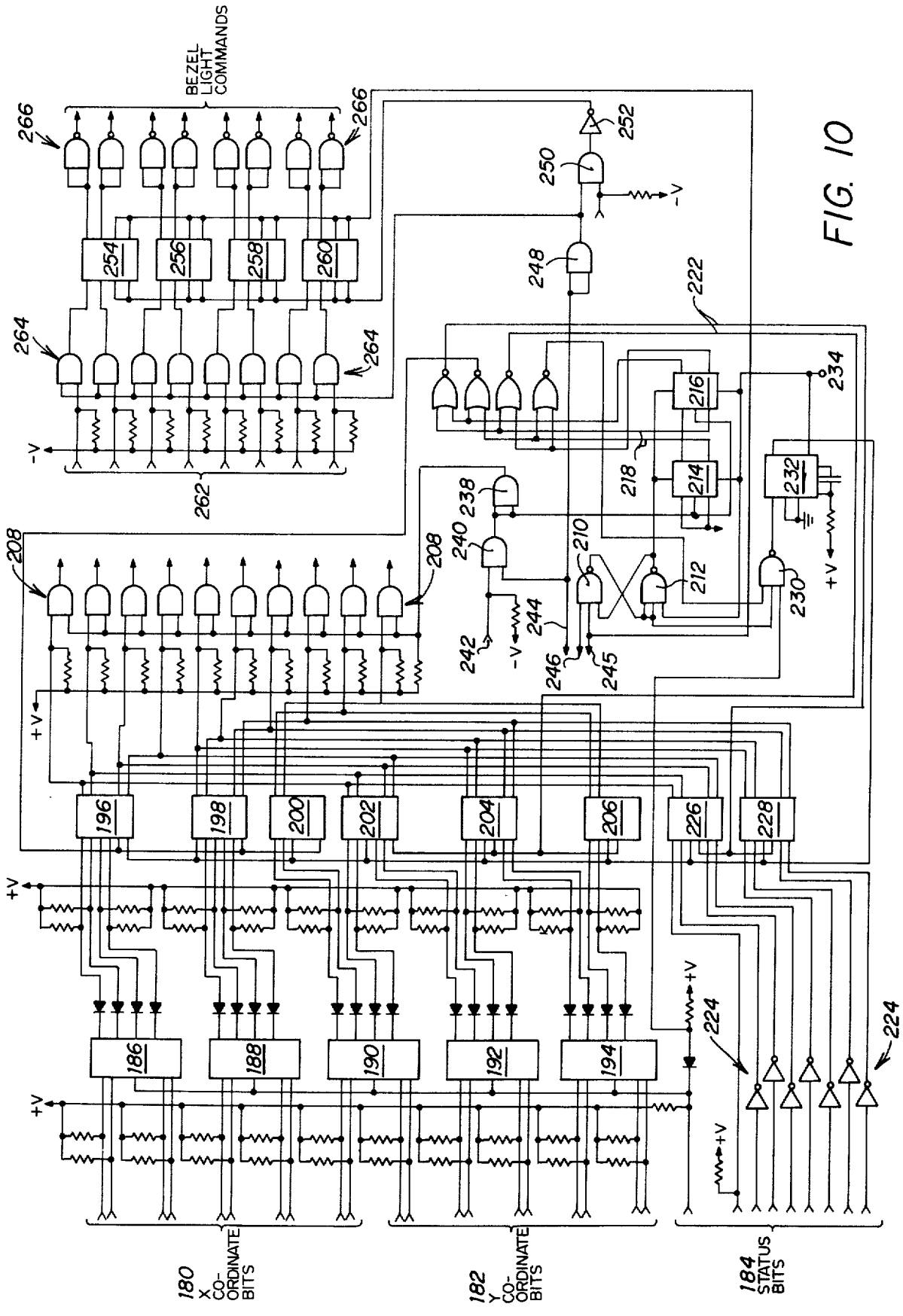

BEZEL LIGHT COMMANDS
266
254
256
258
260
264
262
-V
248
250
252
222
216
234
218
214
232
238
240
208
210
212
230
242
244
246
245
+V
196
198
200
202
204
206
226
228
186
188
190
192
194
224
180 X CO-ORDINATE BITS
182 Y CO-ORDINATE BITS
184 STATUS BITS

FIG. 10

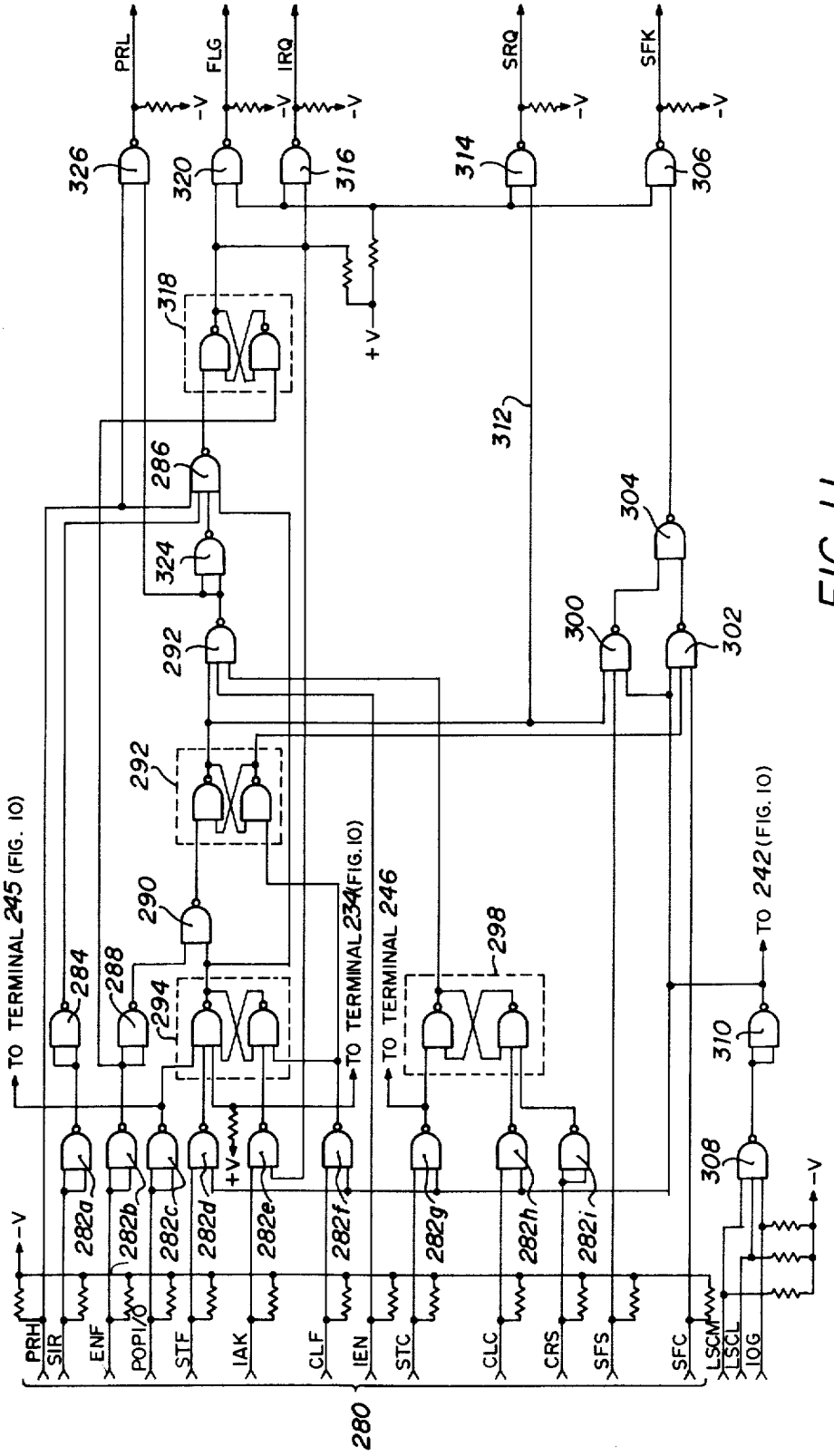

PRL
FLG
IRQ
SRQ
SFK
-V
+V
326
320
316
314
306
318
312
286
324
304
292
300
302
290
284
288
294
298
310
308
280
TO TERMINAL 245 (FIG. 10)
TO TERMINAL 234 (FIG. 10)
TO TERMINAL 246
TO 242 (FIG. 10)
282a
282b
282c
282d
282e
282f
282g
282h
282i
PRH
SIR
ENF
POPIO
STF
IAK
CLF
IEN
STC
CLC
CRS
SFS
SFC
LSCM
LSCL
IOG

FIG. 11

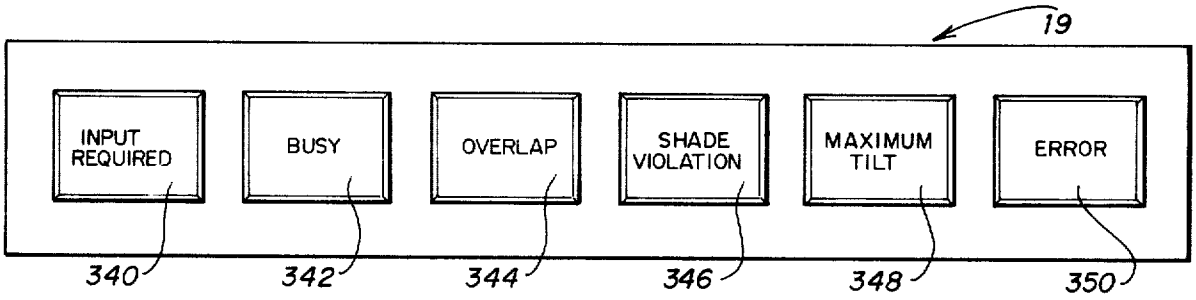

19
INPUT REQUIRED
BUSY
OVERLAP
SHADE VIOLATION
MAXIMUM TILT
ERROR
340
342
344
346
348
350

FIG. 12

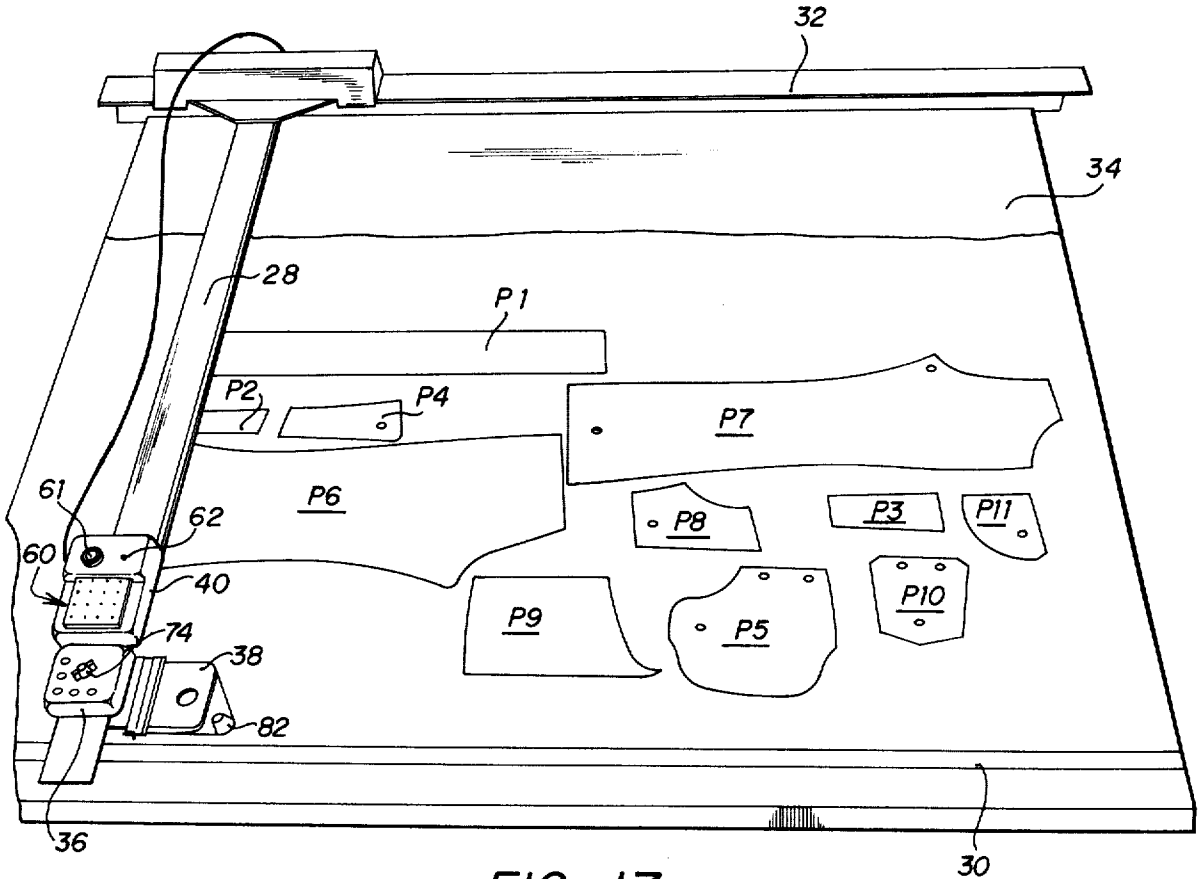

32
34
28
P1
P2
P4
P7
P6
P8
P3
P11
P10
P9
P5
61
62
60
40
74
38
82
36
30

FIG. 17

```
$RULE:
BUILD
RULE NAME 0003200100021
3230

/E
60 40 20 0 -20 -40 -60

999
```

FIG. 13

```
$RULE:
LIST
RULE NAME
```

FIG. 14

```
$RULE:
LIST
RULE NAME 5,12,1,28,16,1
```

FIG. 15

```
$RULE:

*ALL
```

FIG. 16

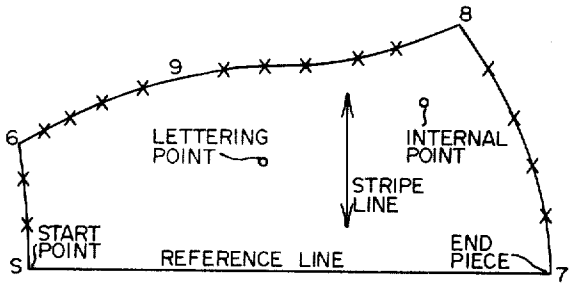


FIG. 18

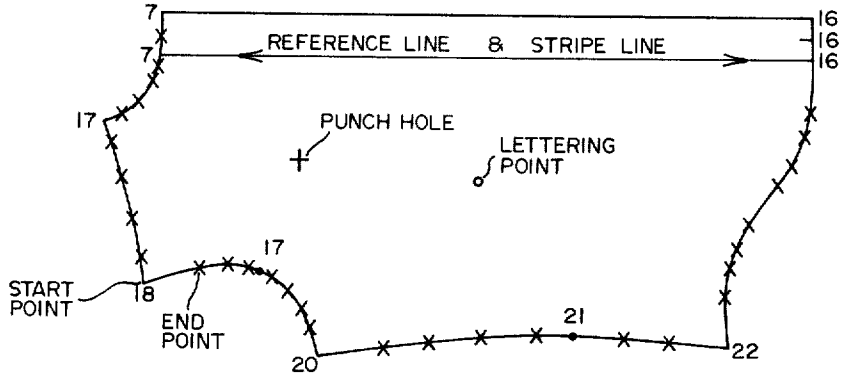


FIG. 19

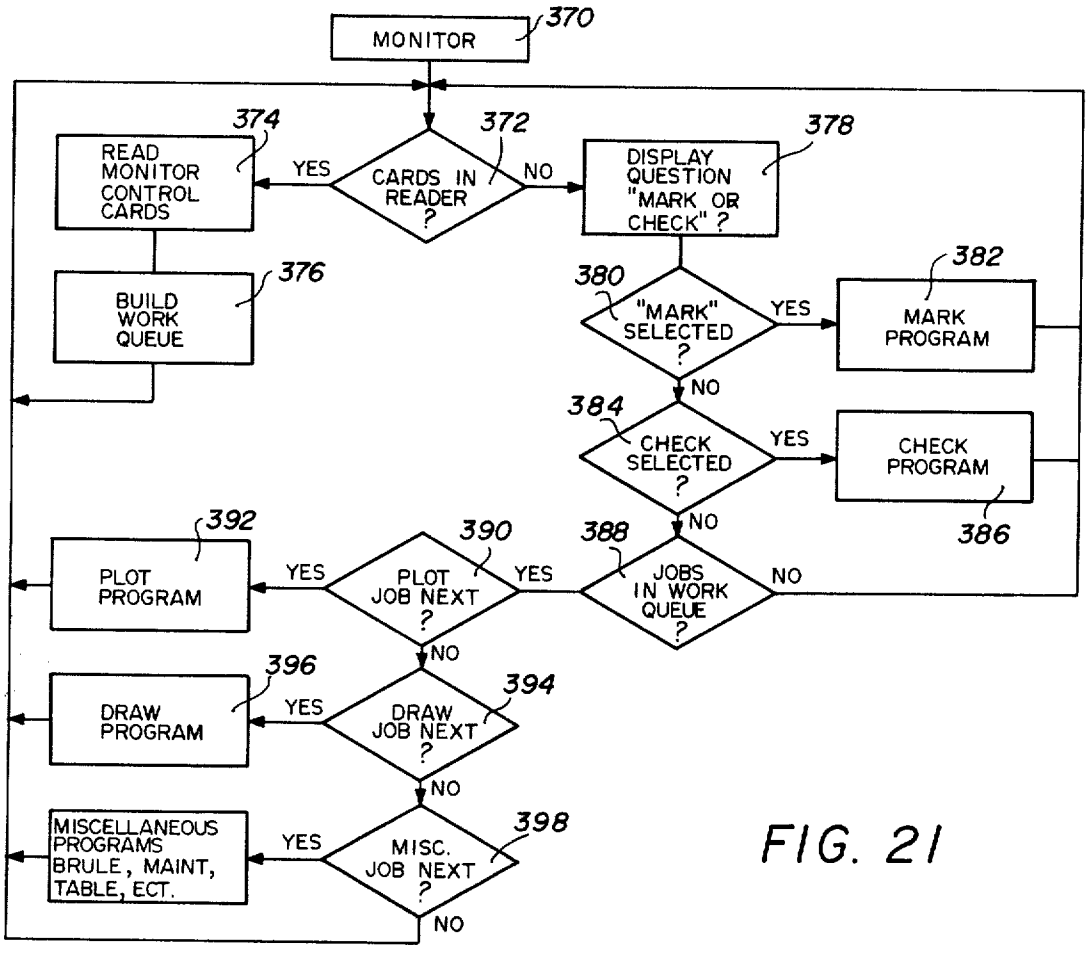


FIG. 21

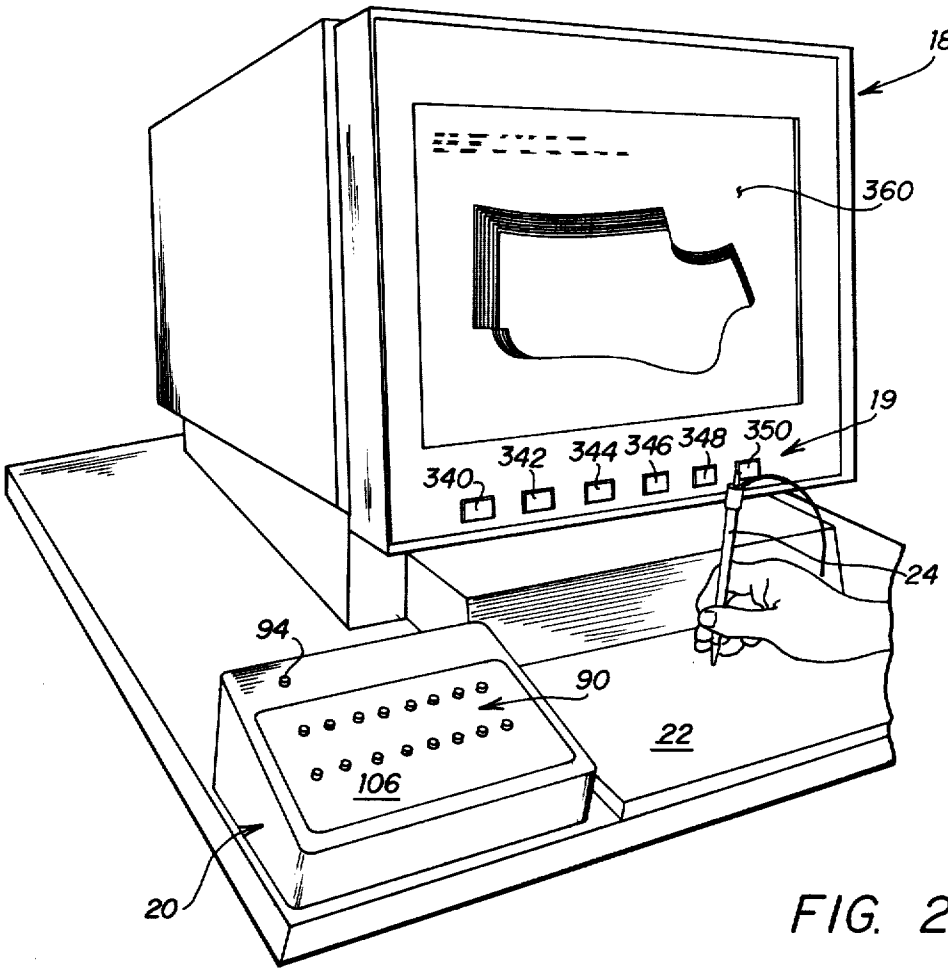

18
360
19
340
342
344
346
348
350
24
94
90
22
106
20

FIG. 20

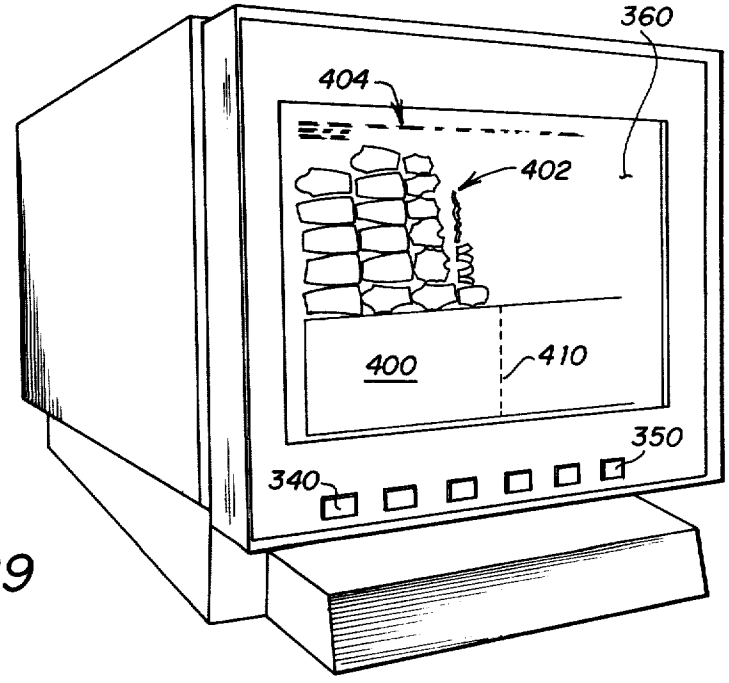

360
404
402
400
410
350
340

FIG. 29

$STYLE: STYLE NAME, RULE TABLE
PIECE # FLIP, BLOCK, STRIPE=GROUP
OPPOSITE, ROTATE, NAP, SHADE, X=VALUE
Y=VALUE, TABLE =n, TILT=n, MSG_1 , MSG_2 , ---------MSG_n

```
$ST:EXAMPSTYLE,CAMSCO1
;0000000011,FLIP,SHADE
;0000000022,SHADE,ROTATE,TABLE=01
;0000000033,FLIP,02
;0000000044,SHADE,TILT=10
;0000000055,BLOCK,STRIPE=1,NAP
;0000000066,FLIP,01
;0000000077,BLOCK,STRIPE=1,NAP
$END:
```

FIG. 22

EFF=%
GOAL=YDS
$MARK: MARKER NAME, WIDTH=VALUE, GOAL=YDS/INCHES, DIMENSION,
SHADE =VALUE, STRIPE=INC/OFFSET, PLAID=VALUE, X=VALUE,
Y=VALUE, FLIP, NAP; STYLE NAME, $SIZE_1$, $SIZE_2$, ----------$SIZE_n$

```
A $MA:EXAMPMARK1,WI=60,GO=3/12,DI;STYLE1,14
B $MA:EXAMPMARK2,WI=59.5,GO=4/24,DI;EXAMPST
    ,122,142,162
C $MA:EXAMPMARK3,WI=60.5,GO=3/0,DI,ST=2/1,PL
    =2,FLIP,NAP;STYLE1,14 ;STYLE2,162
D $MA:MARK1,WI=60.S,EFF=90.0;STYLE1,14,R14
```

FIG. 23

$DRAW: MARKER NAME, ALPHA=VALUE, GAP=VALUE, SCALE=VALUE;
MSG_1 , MSG_2 , ----------MSG_n

```
A $DR:EXAMPMARK,ALPHA=A,GAP=4
B $DR:MK1,AL=A,SC=.5;11
C $DR:MK1
```

FIG. 24

$PLOT: PIECE #, CUT NEST, RULE, TABLE, SCALE=VALUE;
$SIZE_1$, $SIZE_2$, ----------$SIZE_n$

```
A $PL:00000000011,NEST,CAMSCO1,SC=1.0
B $PL:00000000011,NEST,CAMSCO1,SC=0.5
C $PL:00000000011,NEST,CAMSCO1,SC=1.0;10,12
D $PL:00000000011,CAMSCO1
```

FIG. 25

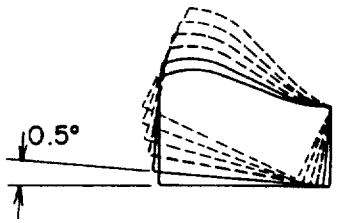


FIG. 26

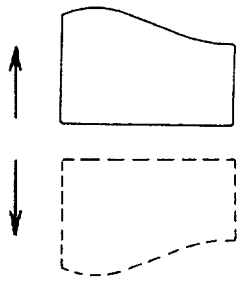


FIG. 27

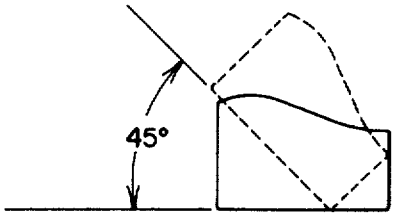


FIG. 28

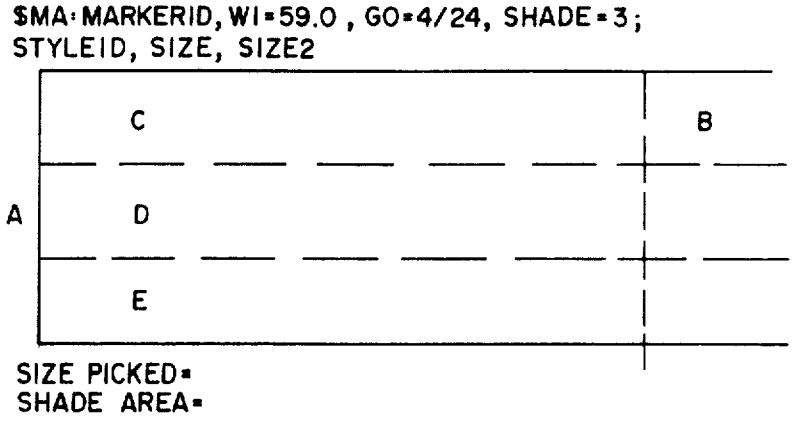


FIG. 32

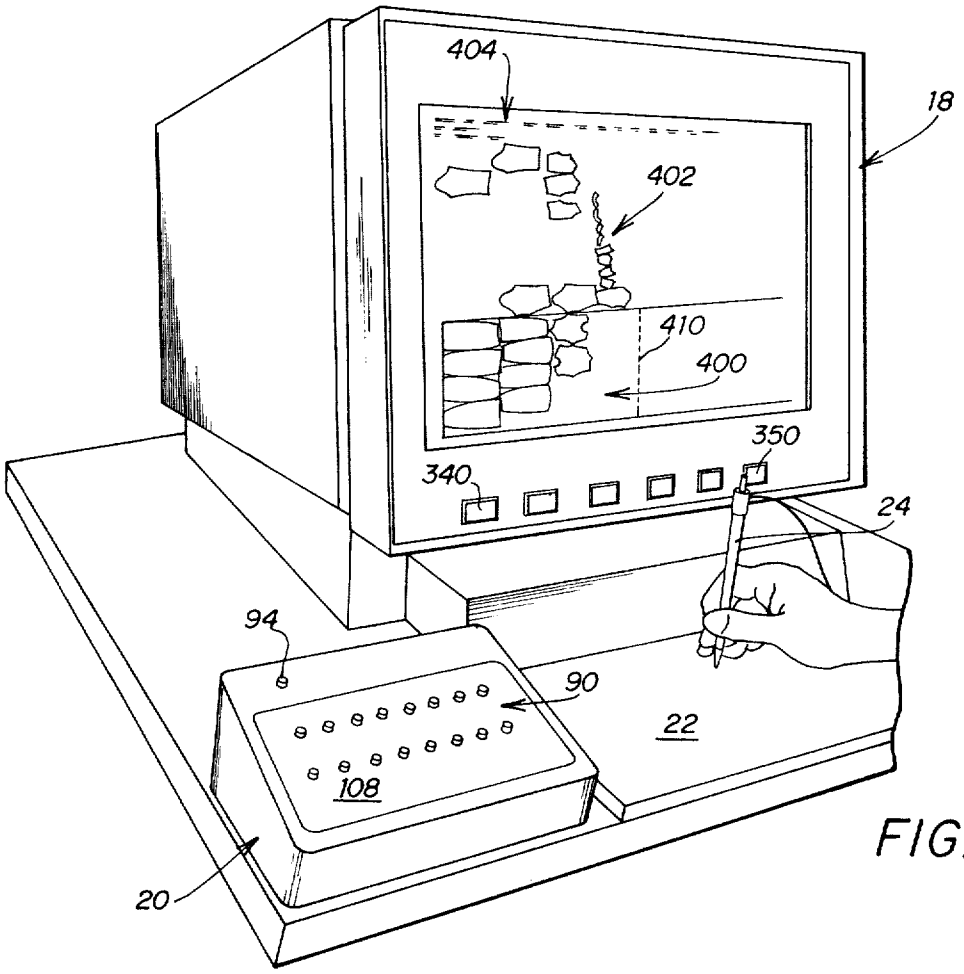


FIG. 30

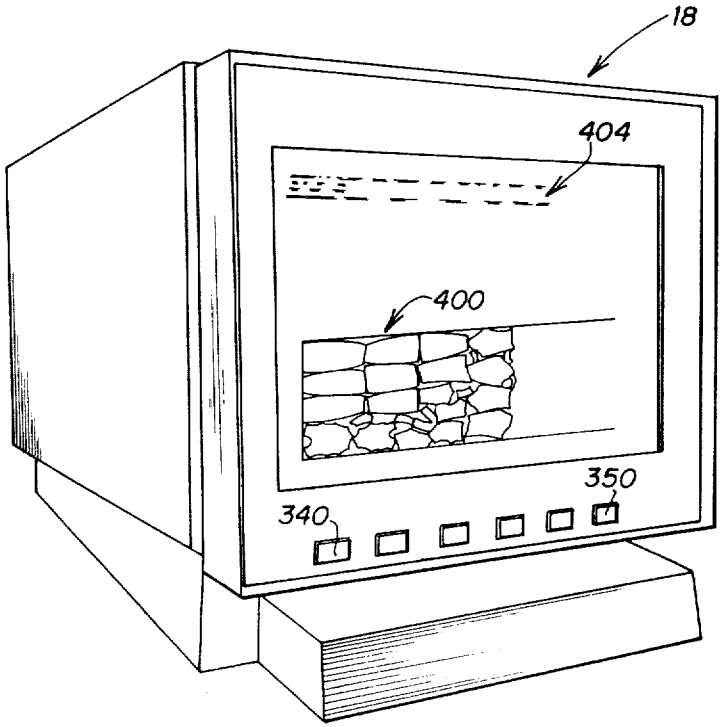


FIG. 31

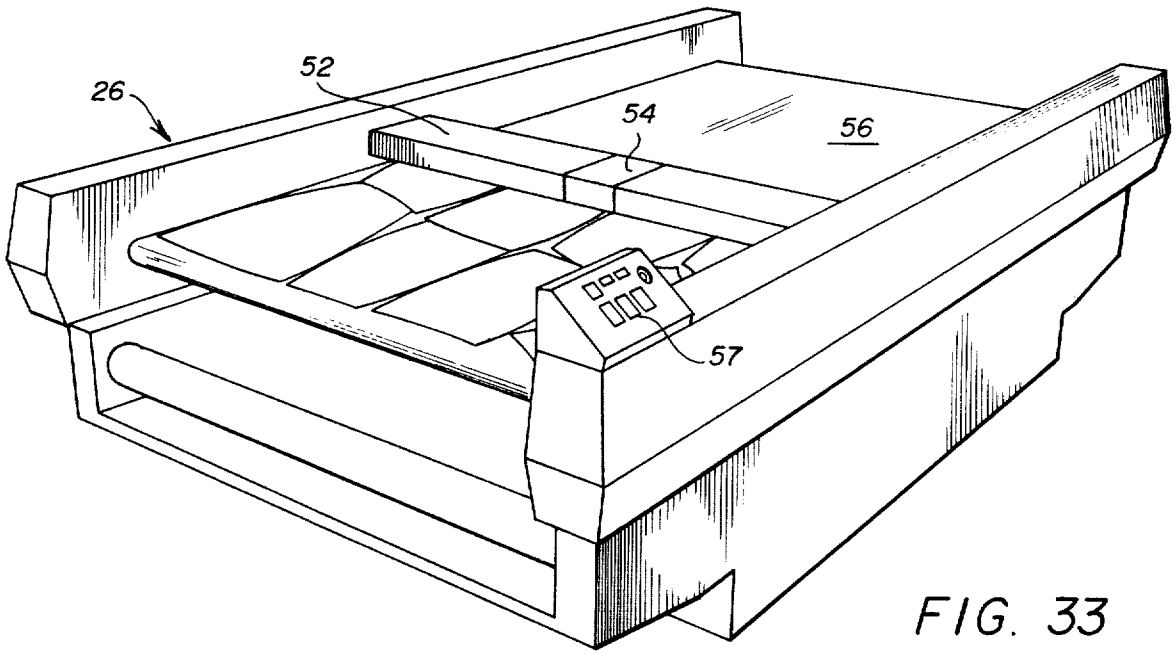

26
52
54
56
57

FIG. 33

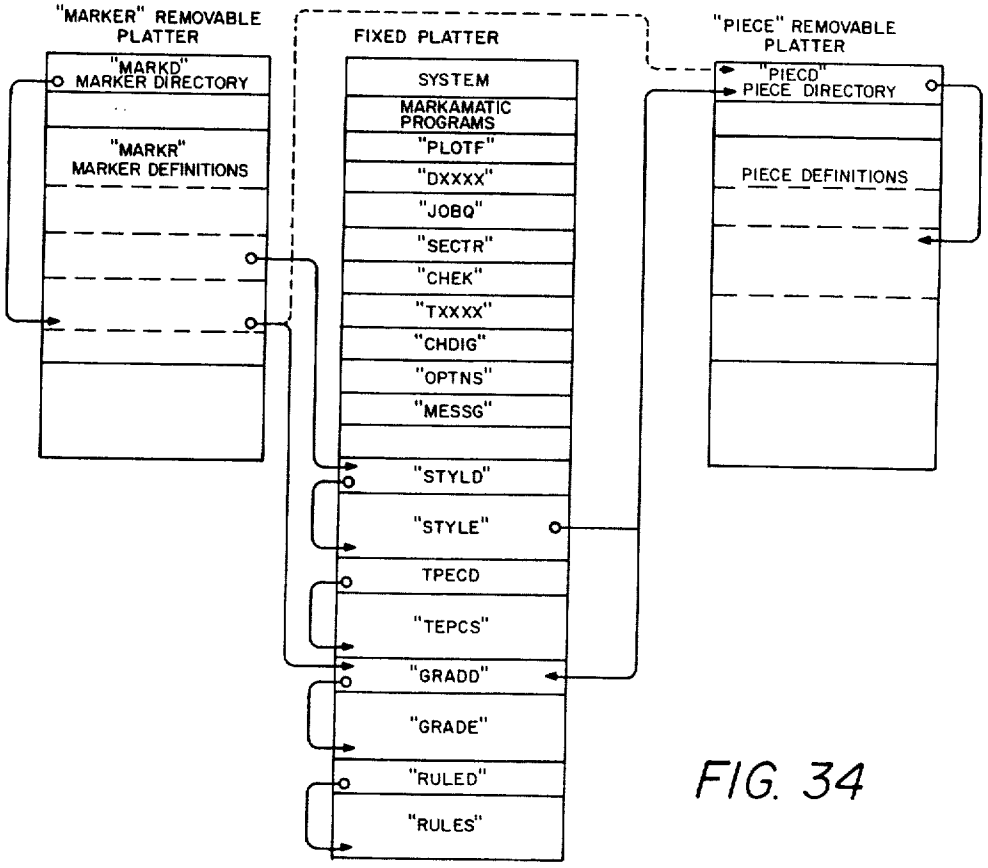

"MARKER" REMOVABLE PLATTER
"MARKD" MARKER DIRECTORY
"MARKR" MARKER DEFINITIONS
FIXED PLATTER
SYSTEM
MARKAMATIC PROGRAMS
"PLOTF"
"DXXXX"
"JOBQ"
"SECTR"
"CHEK"
"TXXXX"
"CHDIG"
"OPTNS"
"MESSG"
"STYLD"
"STYLE"
TPECD
"TEPCS"
"GRADD"
"GRADE"
"RULED"
"RULES"
"PIECE" REMOVABLE PLATTER
"PIECD" PIECE DIRECTORY
PIECE DEFINITIONS

FIG. 34

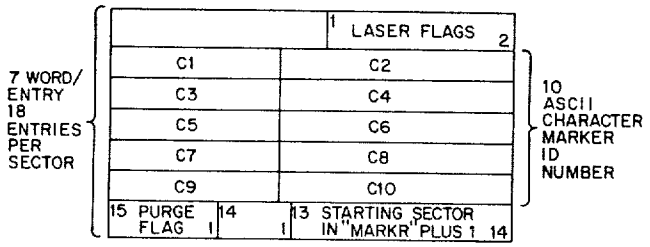


FIG. 35

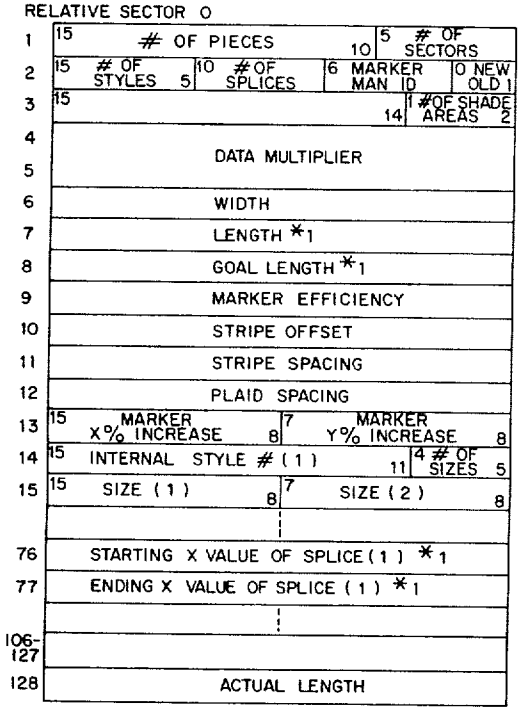


FIG. 36

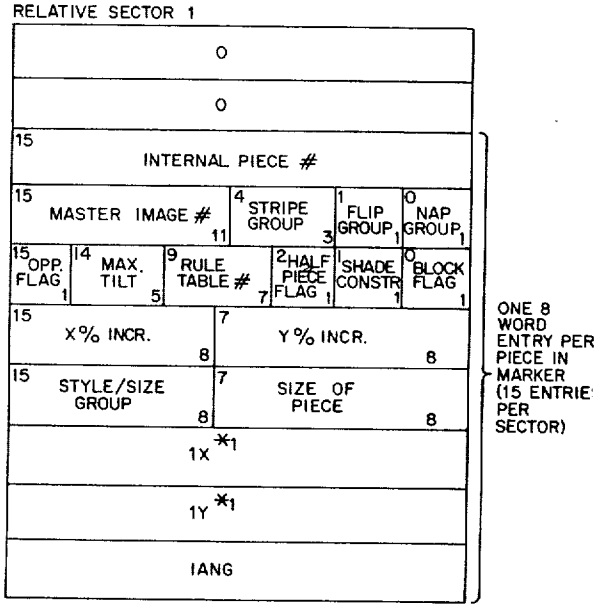


FIG. 37

FIG. 38

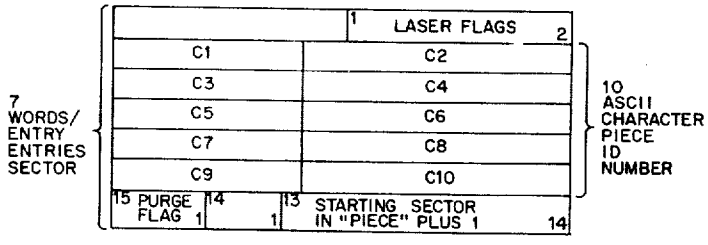
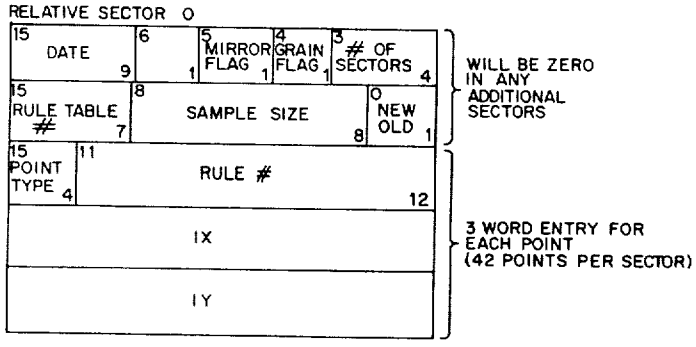


FIG. 39



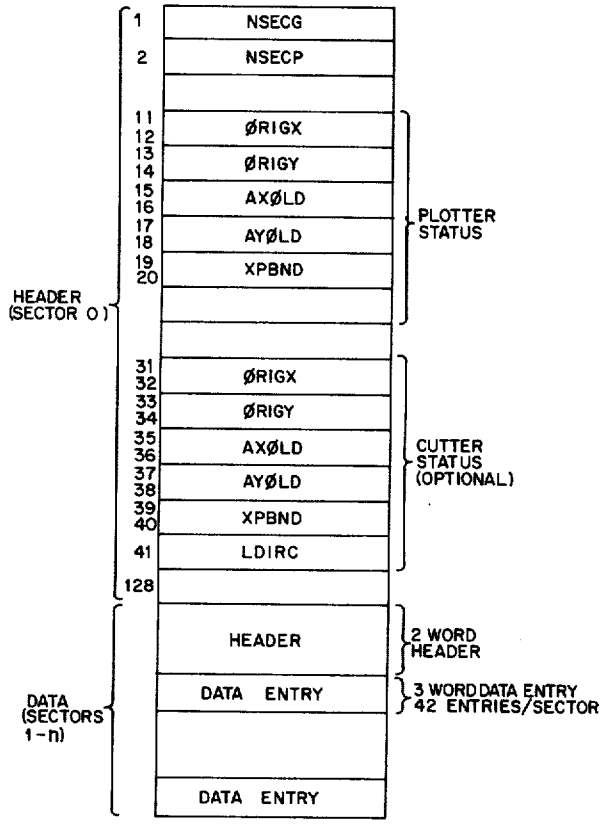


FIG. 40

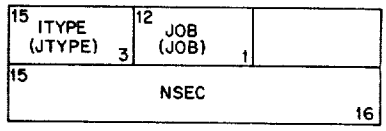


FIG. 41

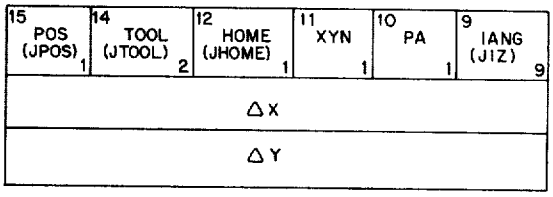


FIG. 42

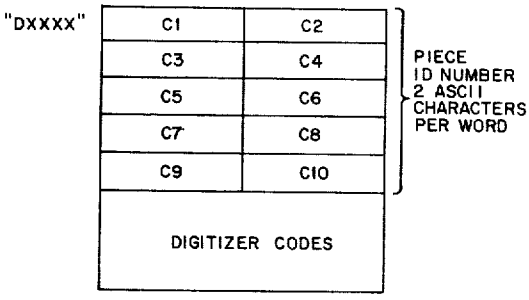


FIG. 43

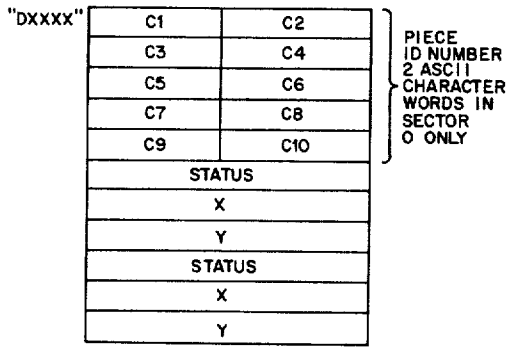


FIG. 44

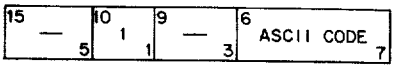


FIG. 45

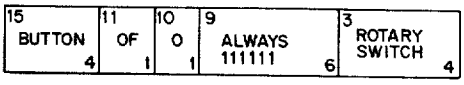


FIG. 46

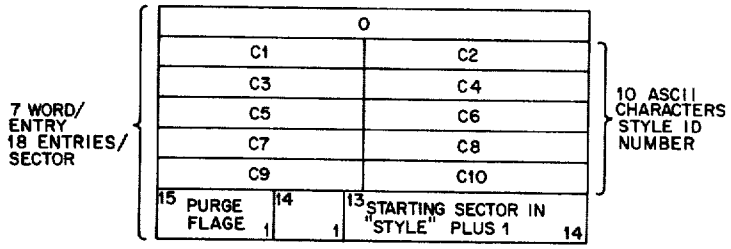


FIG. 47

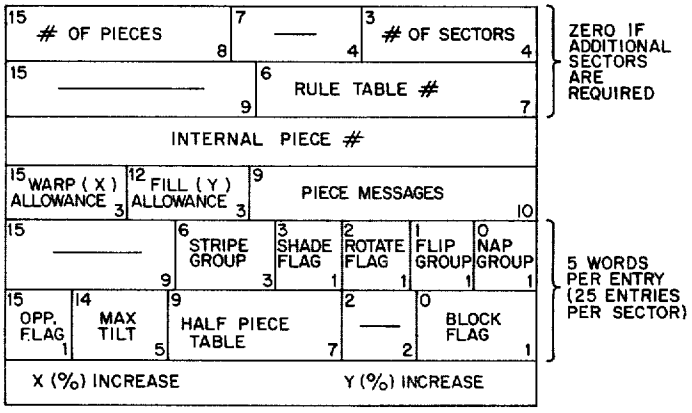


FIG. 48

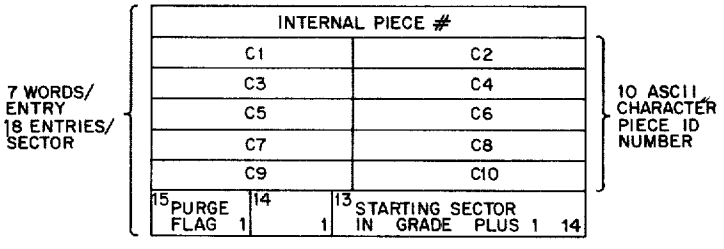


FIG. 49

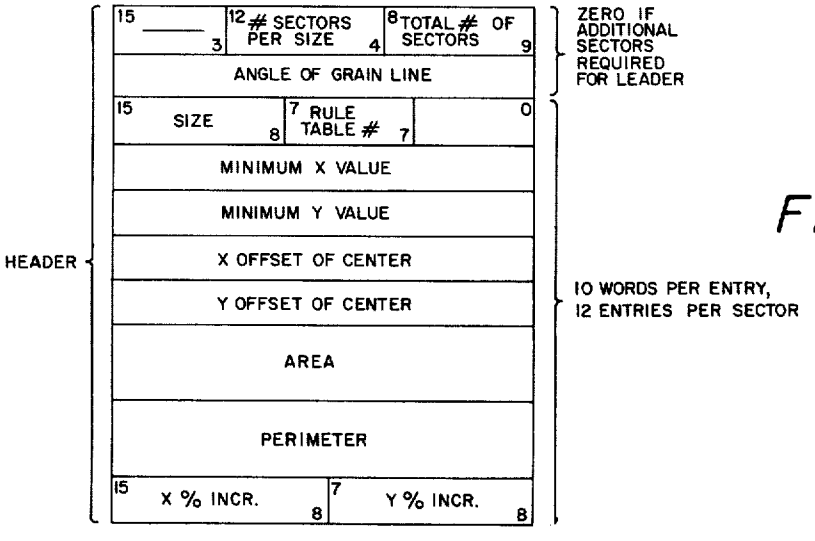


FIG. 50

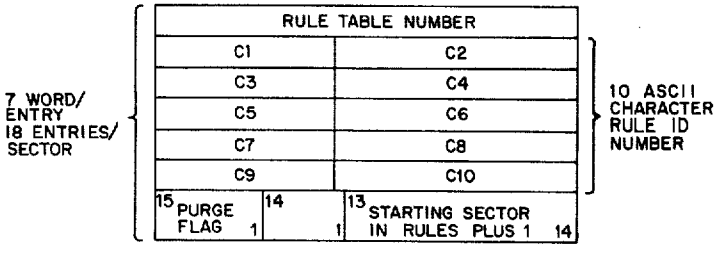


FIG. 51

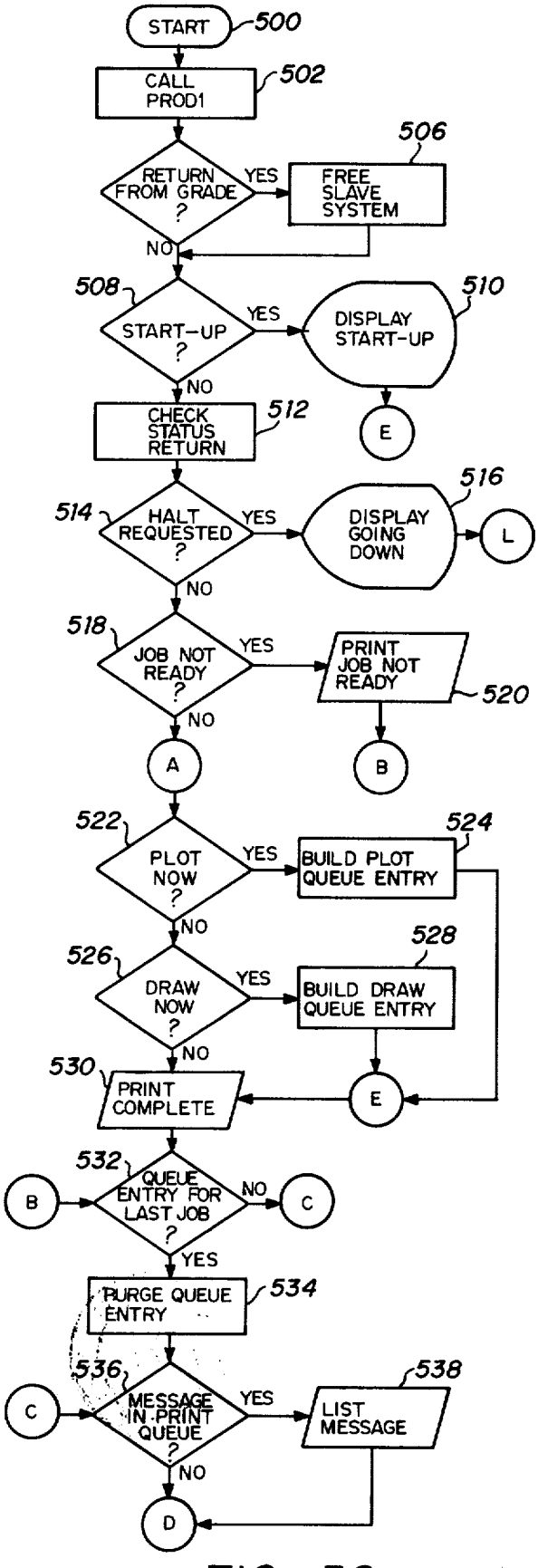

START
500
CALL PROD1
502
RETURN FROM GRADE ?
YES
FREE SLAVE SYSTEM
506
NO
START-UP ?
508
YES
DISPLAY START-UP
510
NO
CHECK STATUS RETURN
512
E
HALT REQUESTED ?
514
YES
DISPLAY GOING DOWN
516
L
NO
JOB NOT READY ?
518
YES
PRINT JOB NOT READY
520
NO
A
B
PLOT NOW ?
522
YES
BUILD PLOT QUEUE ENTRY
524
NO
DRAW NOW ?
526
YES
BUILD DRAW QUEUE ENTRY
528
NO
PRINT COMPLETE
530
E
B
QUEUE ENTRY FOR LAST JOB ?
532
NO
C
YES
PURGE QUEUE ENTRY
534
C
MESSAGE IN PRINT QUEUE ?
536
YES
LIST MESSAGE
538
NO
D

FIG. 52

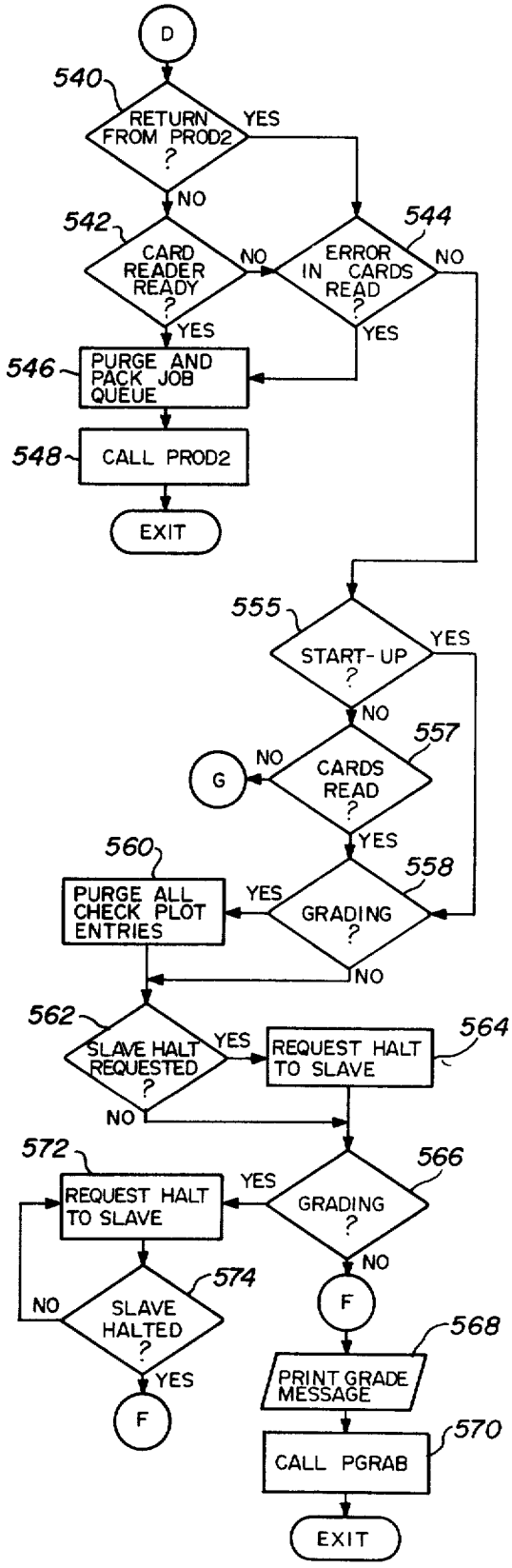

D
RETURN FROM PROD2 ?
540
YES
NO
CARD READER READY ?
542
NO
ERROR IN CARDS READ ?
544
NO
YES
YES
PURGE AND PACK JOB QUEUE
546
CALL PROD2
548
EXIT
START-UP ?
555
YES
NO
CARDS READ ?
557
NO
G
YES
GRADING ?
558
YES
PURGE ALL CHECK PLOT ENTRIES
560
NO
SLAVE HALT REQUESTED ?
562
YES
REQUEST HALT TO SLAVE
564
NO
REQUEST HALT TO SLAVE
572
YES
GRADING ?
566
NO
SLAVE HALTED ?
574
NO
YES
F
F
PRINT GRADE MESSAGE
568
CALL PGRAB
570
EXIT

FIG. 53

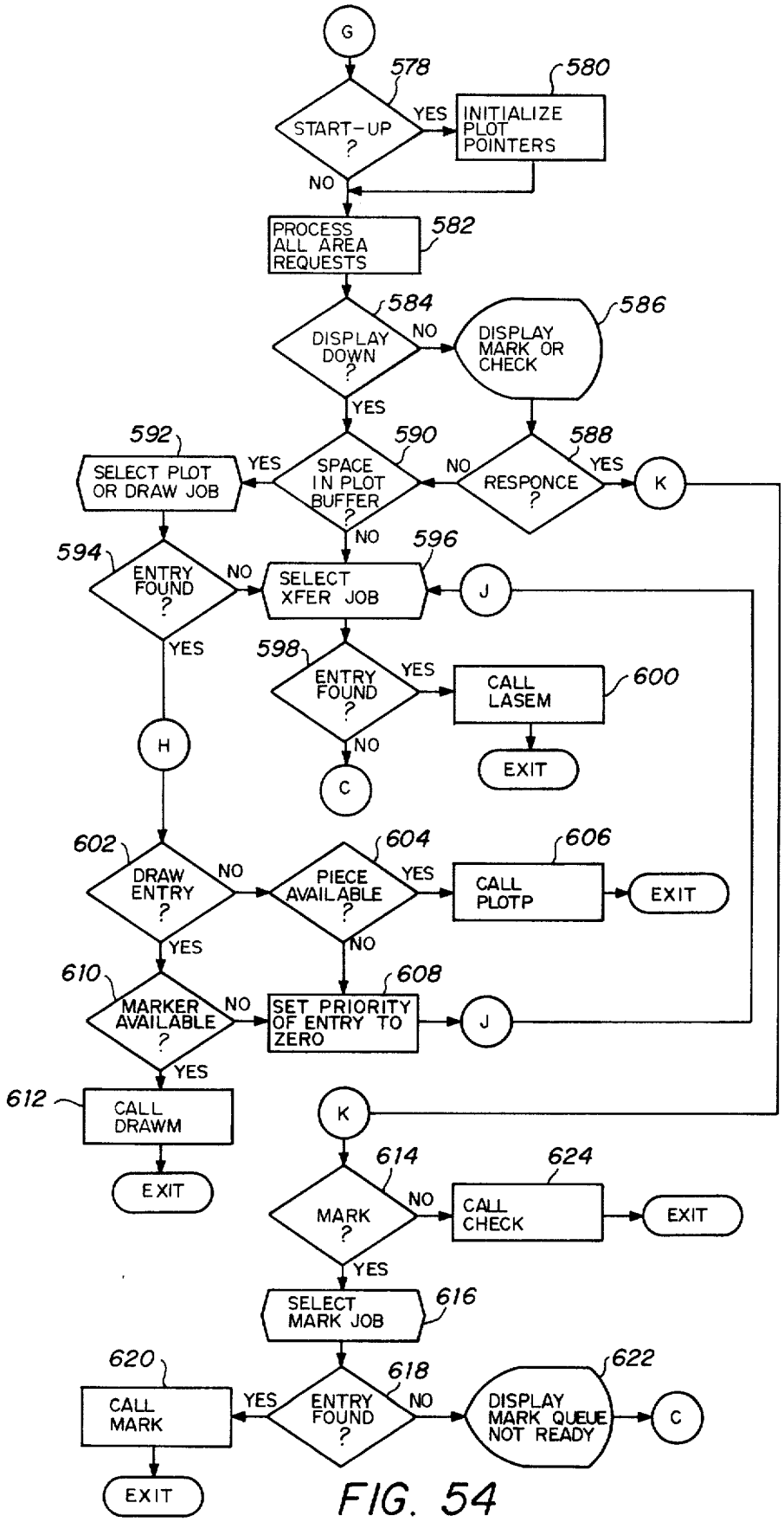

G
578
START-UP ?
YES
580
INITIALIZE PLOT POINTERS
NO
PROCESS ALL AREA REQUESTS
582
584
DISPLAY DOWN ?
NO
586
DISPLAY MARK OR CHECK
YES
592
SELECT PLOT OR DRAW JOB
YES
590
SPACE IN PLOT BUFFER ?
NO
588
RESPONCE ?
YES
K
NO
596
594
ENTRY FOUND ?
NO
SELECT XFER JOB
J
YES
598
ENTRY FOUND ?
YES
CALL LASEM
600
EXIT
NO
H
C
602
DRAW ENTRY ?
NO
604
PIECE AVAILABLE ?
YES
606
CALL PLOTP
EXIT
YES
NO
610
MARKER AVAILABLE ?
NO
608
SET PRIORITY OF ENTRY TO ZERO
J
YES
612
CALL DRAWM
EXIT
K
614
MARK ?
NO
624
CALL CHECK
EXIT
YES
SELECT MARK JOB
616
620
CALL MARK
YES
618
ENTRY FOUND ?
NO
622
DISPLAY MARK QUEUE NOT READY
C
EXIT

FIG. 54

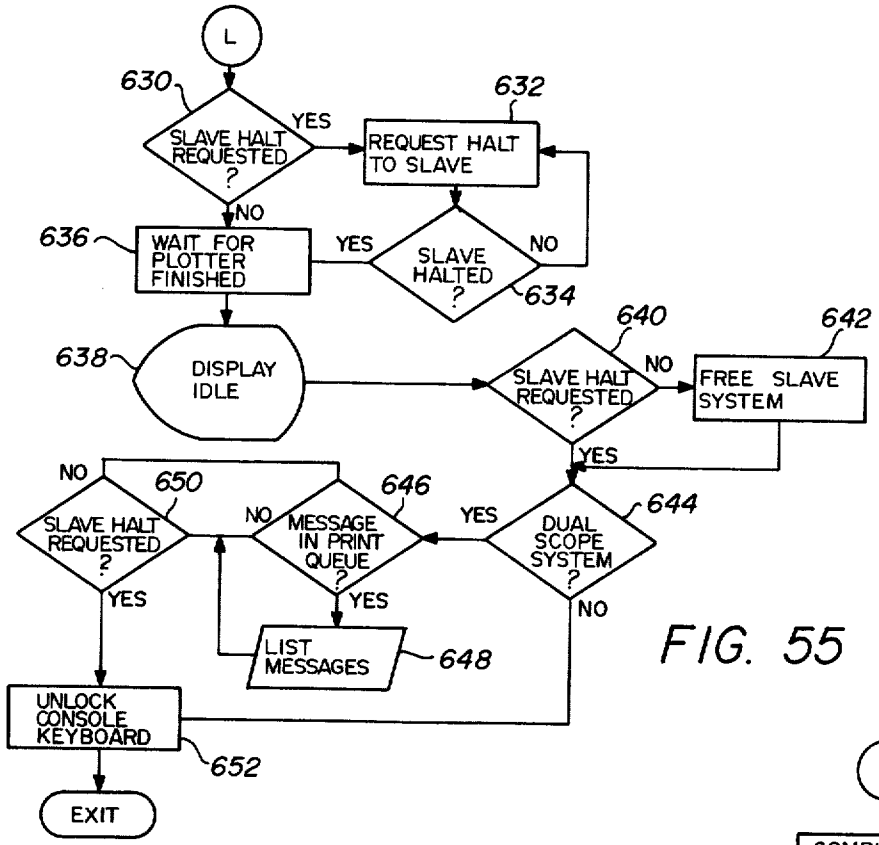

L
630
SLAVE HALT REQUESTED ?
YES
632
REQUEST HALT TO SLAVE
NO
636
WAIT FOR PLOTTER FINISHED
YES
SLAVE HALTED ?
NO
634
638
DISPLAY IDLE
640
SLAVE HALT REQUESTED ?
NO
642
FREE SLAVE SYSTEM
YES
NO
650
SLAVE HALT REQUESTED ?
NO
646
MESSAGE IN PRINT QUEUE ?
YES
644
DUAL SCOPE SYSTEM ?
YES
YES
NO
LIST MESSAGES
648
UNLOCK CONSOLE KEYBOARD
652
EXIT

FIG. 55

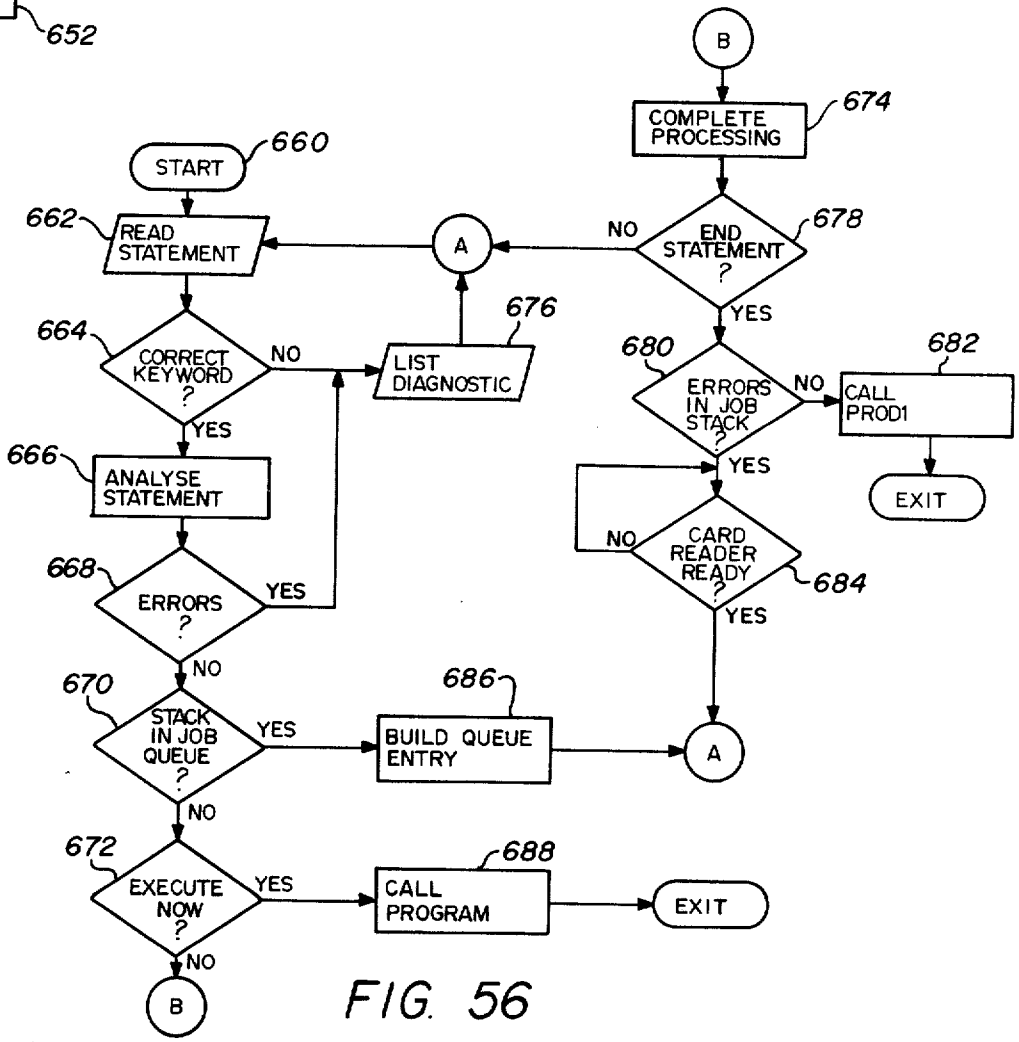

B
674
COMPLETE PROCESSING
START
660
662
READ STATEMENT
A
NO
678
END STATEMENT ?
YES
664
CORRECT KEYWORD ?
NO
676
LIST DIAGNOSTIC
680
ERRORS IN JOB STACK ?
NO
682
CALL PROD1
EXIT
YES
666
ANALYSE STATEMENT
NO
CARD READER READY ?
684
668
ERRORS ?
YES
YES
NO
670
STACK IN JOB QUEUE ?
YES
686
BUILD QUEUE ENTRY
A
NO
672
EXECUTE NOW ?
YES
688
CALL PROGRAM
EXIT
NO
B

FIG. 56

INTERACTIVE MAN-MACHINE METHOD AND SYSTEM FOR GRADING PATTERN PIECES AND FOR PRODUCING AN APPAREL MARKER

FIELD OF THE INVENTION

This invention relates to grading apparel patterns and preparing apparel pattern markers, and more particularly relates to a man-machine interactive method and system for automatically grading a plurality of apparel patterns and for producing efficient apparent pattern markers.

THE PRIOR ART

In the mass production of garments and items of wearing apparel, it has long been the practice to produce model pattern pieces and to manually grade additional pattern sizes from the model pattern pieces. Each pattern piece for each size and style garment desired to be produced is then cut out and manually arranged over a long length of paper called the marker. When an efficient layout of the pattern pieces over the marker is determined, the outline of the pieces are manually traced on the marker. The marker is then placed over a stack of material laid out along a cutting table and a saw is used to cut the multilayers of material along the pattern pieces outlined on the marker.

The above-described prior technique is extremely time consuming and laborious. In addition, unless the personnel accomplishing the grading function are highly skilled, errors can be introduced into the graded pattern pieces. Even with highly trained personnel laying out of the pattern markers, inefficient marker layouts may result, thereby resulting in expensive waste of cloth during the cutting procedure. In addition, the requirement for inventory procedures and storage space for patterns and markers is substantial with prior manual methods. A need has thus arisen for a method and system for quickly and accurately grading apparel patterns and for laying out and automatically plotting pattern markers.

SUMMARY OF THE INVENTION

In accordance with the present invention, a method and system have been developed which substantially eliminates or reduces the problems noted above which have occurred with prior manual pattern grading and marker preparation techniques. The present technique substantially reduces the time required for pattern grading and marking and eliminates the need for physical storage space for paper patterns and markers. The present method and system may be operated by personnel familiar with previous manual grading and marking techniques and specialized personnel having intricate knowledge of electronic data processing are not required.

In accordance with the present invention, a system is provided for laying out an apparel pattern marker wherein a plurality of apparel pattern pieces are to be arranged within a predetermined marker area. The system includes a screen for displaying to an operator scaled miniature representations of the pattern pieces adjacent a scaled miniature representation of the marker area. A stylus is movable by the operator to selectively move the representations of the pattern pieces on the screen to within the representation of the marker area on the screen. By use of the stylus and a function box, the operator efficiently positions each of the pattern pieces within the marker area. When the final marker is prepared on the screen, a plotter is energized to plot out a full size pattern marker corresponding to the miniature arrangement arrived at on the screen by the operator.

In accordance with another aspect of the invention, a system is provided for laying out a plurality of pattern pieces in a pattern marker which includes a digitizer for generating pattern piece coordinate data and grading instructions for the coordinate data. A storage device stores the coordinate data and grading instructions and circuitry is provided for selectively retrieving portions of the coordinate data and grading instructions corresponding to pattern pieces of a preselected style. Structure is provided to apply the grading instructions to the retrieved coordinate data to grade the pattern pieces to preselected sizes. A display screen displays miniature representations of the graded pattern pieces and the marker area. A data tablet includes a stylus movable relative to the data tablet for moving the pattern pieces on the display screen into the displayed marker area. A plotter forms a full size marker corresponding to the final arrangement of pattern pieces on the display screen.

In accordance with another aspect of the invention, a system is provided for grading and checking model pattern pieces and includes a digitizer for generating pattern piece coordinate data and grading instructions for the coordinate data. The coordinate data and grading instructions are stored. The grading instructions are then selectively applied to the coordinate data according to predetermined grade rules. A screen selectively displays nested plots of a plurality of graded sizes of each of the pattern pieces. A data tablet is interconnected with the display and includes a stylus. Circuitry is responsive to movement of the stylus adjacent the data tablet for displaying grading instructions for selected points on the displayed nested plots. A function box has switches operable to provide the grading instructions for selected points on the displayed nested plots. Circuitry is provided for regrading the selected points according to the revised grading instructions.

In accordance with yet another aspect of the invention, a system is provided for preparing an apparel marker. The system includes a digitizer table for supporting pattern pieces to be digitized. A slide bar is laterally movable across the table. A cursor is mounted on the slide rule and is movable along the length of the slide rule for being positioned at points along the pattern pieces. The cursor has switches operable to generate electrical signals representative of the coordinates of the position of the cursor on the table. A function box is movable along the slide bar with the cursor and includes buttons operable to generate grading instruction signals corresponding to the points along the pattern pieces. Circuitry is provided to store representations of the electrical signals and the grading instruction signals.

DESCRIPTION OF THE DRAWINGS

For a more complete understanding of the present invention and for further objects and advantages thereof, reference is now made to the following description taken in conjunction with the accompanying drawings, in which:

FIG. 1 is a perspective view of the present grading and marker preparation system;

FIG. 2 is a block diagram of the basic components of the system shown in FIG. 1;

FIG. 3 is a top view of the cursor function box;

FIG. 4 is a top view of the cursor control box;

FIG. 5 is a perspective view, partially broken away, of the function box utilized with the data tablet;

FIG. 6 is a view of the template used on the function box shown in FIG. 5 during the CHECK program;

FIG. 7 is a top view of the template used on the function box shown in FIG. 5 during the MARK program;

FIG. 8 is an electrical schematic of the cursor function box and the data tablet function box;

FIG. 9 is an electrical schematic for the audible system for the cursor function box;

FIGS. 10 and 11 are electrical schematics of the interface circuitry between the data tablet and the central controller;

FIG. 12 is a view of the bezel lights on the display screen;

FIGS. 13–16 are the coding sheets utilized in the input of Grade Rule data into the system;

FIG. 17 is a perspective view of the digitizer table with a plurality of typical garment pattern pieces laid out thereon;

FIGS. 18 and 19 are representations of pattern pieces digitized according to the present technique;

FIG. 20 is a perspective view of the interactive display console of the invention during the CHECK program;

FIG. 21 is a flow chart of various programs of the invention;

FIG. 22 is a coding sheet for the input of $STYLE: program data;

FIG. 23 is a coding sheet to enable the input of $MARK: information;

FIG. 24 is a coding sheet to enable input of $DRAW: information to the system;

FIG. 25 is a coding sheet enabling the input of PLOT information;

FIG. 26 is an example of tilting of a pattern piece on the display screen;

FIG. 27 is an illustration of flipping a pattern piece on the display screen;

FIG. 28 is an example of rotating a pattern piece on the display screen;

FIG. 29 is a perspective view of the display screen prior to operation of the MARK program;

FIG. 30 is a perspective view of the interactive console system illustrating positioning of pattern pieces within the marker area on the display screen during the MARK program;

FIG. 31 is a perspective view of the interactive console shown in FIG. 30 after the marker has been completed;

FIG. 32 is a diagramatic illustration of shade areas defined on a marker;

FIG. 33 is a perspective view of the plotter according to the present invention when plotting a full size marker;

FIG. 34 illustrates the file organization for the present system;

FIGS. 35–51 illustrate details in the file format for various programs of the system; and

FIGS. 52–56 illustrate flow diagrams for PROD 1 and PROD 2 programs of the invention.

DESCRIPTION OF THE PREFERRED EMBODIMENT

System Components

Referring to FIG. 1, the present system includes a digitizer 10 which inputs digital coordinate data representing pattern pieces and grading instructions into a central processor 12. A card reader 14 is utilized to provide instructions to the central processor 12 and an alpha/numeric cathode ray tube terminal 16 is utilized to input instructions to the central processor 12 and to display data output from the central processor 12. Terminal 16 may alternatively comprise a conventional teleprinter. A cathode ray tube screen display 18 is connected to display digital coordinate data generated from the central processor 12. A function box 20 includes a plurality of button switches thereon for enabling control of various functions of the display screen 18. A data tablet 22 includes a movable stylus 24 which may be utilized to change portions of the display on the screen 18. A plotter 26 is interconnected with the central processor 12 for automatically plotting a full size apparel pattern marker which may be used directly to cut pattern pieces from a stack of material.

FIG. 2 illustrates an electrical block diagram of the interconnection of the components of the present system. The digitizer 10 is interconnected through an interface 27 to the central controller 12. As shown in FIG. 1, the digitizer 10 includes a vertical bar 28 which is laterally movable along parallel guide rails 30 and 32 located along the top and bottom of a table 34. A cursor control box 36 is attached to the bar 28 and includes a cursor aperture member 38 which is centered over the point desired to be digitized. The cursor control box 36 and the cursor aperture member 38 are movable vertically along the bar 28 so that any point laying within the confines of the table 34 may be digitized by the system. A multi-key function box 40 is also mounted on the bar 28 and is interconnected with the control box 36 for movement vertically along the bar 28 and for lateral movement with the bar 28.

Electrical signals generated from the control box 36 and the function box 40 are applied via a flexible conductor 42 and through the interface 27 to the central controller 12 for storage therein. A plurality of pattern pieces designated generally as P are oriented on the table 34 for digitizing in FIG. 1. By placing the cursor aperture member 38 over a desired point on the pattern pieces P, and by proper operation of buttons on the control box 36 and on the function box 40, electrical signals representative of the coordinates of the point and the grading rules for the point are applied through the electrical conductor 42 to the central processor 12. By inputting a plurality of spaced apart points along each of the patterns P, coordinate data representative of the shape of the pattern pieces is stored in the central processor 12.

A number of digitizer systems may be utilized in conjunction with the present system. However, in the preferred embodiment of the invention, the digitizer Model No. 3931A manufactured and sold by Auto-Trol Corp. of Denver, Colo. has been found to operate satisfactorily. Such a digitizer is modified only to the extent that the function box 40 constructed in accordance with the invention, to be later described in detail, is mounted for movement with the cursor control box 36 to enable the operator to input data with ease. The in-

terface **27** comprises interfacing included in the Autotrol Model No. 3913B package. The interconnection of the interface **27** between the digitizer **10** and the central controller is well known in the art and is specified in detail in literature provided with the HT12566B interface module.

The central controller **12** may comprise any suitable electronic data processor operable to provide real time control for the present system and having the desirable storage capacity. In the preferred embodiment, the central controller **12** is operable with a 16 bit digital word size and should have a minimum storage capacity of 24,576 words with a memory cycle time of in the range of 970 nanoseconds.

In the preferred embodiment, the central controller **12** comprises a general purpose digital computer such as the 2100A computer manufactured and sold by Hewlett Packard of Pala Alto, California. Firmware coding for the central controller is stored in bipolar read-only-memory in the floating point hardware option to substantially increase the performance of all floating point instructions and subroutines. For further description of the construction and operation of the Model 2100A Hewlett Packard computer, reference is made to the Hewlett Packard 2100A reference manual which is incorporated herein by reference.

Storage memory for the central controller **12** is provided by a magnetic disk assembly **44** which is connected through an interface **46** with the controller **12.** Additional flexibility in storage may be provided by a magnetic tape unit **47.** The disk **44** may comprise for example the Nashua 4415 Disc Cartridge manufactured and sold by the Nashua Corporation of Nashua, New Hampshire. In addition, the HP7901A cartridge Disk Subsystem manufactured and sold by Hewlett Packard may be utilized in conjunction with the disk **44.** The interface **46** may comprise for example the Hewlett Packard Models 13210-60000 and 13210-60004 interfaces. The magnetic tape storage unit **47** may comprise for example the **7970B** digital magnetic tape units manufactured and sold by Hewlett Packard of Palo Alto, Calif.

Each disk **44** utilized with the system is capable of storing approximately 1.25 million words of data to be used by the central controller **12.** Pattern pieces, marker descriptions and other data are stored on the disk platters. One of the disk platters is permanently attached to the disk drive and cannot be removed or accessed by the user. The other platter is a "removable" platter that is inserted by the user. In practice, the user may find it desirable to procure additional removable platters to store piece and marker descriptions in order to back up his disk files.

The card reader **14** shown in FIG. **1** may comprise for example the 6042 card reader manufactured and sold by Mohawk Data Sciences Corporation of King of Prussia, Pa. The card reader **14** is operated by the operator to input MONITOR control cards, grading rules and other types of data input via conventional computer punch cards, as will be later described.

The alpha/numeric display terminal **16** may comprise any suitable input/output terminal. For example, the Data Point Model No. 3300 or Telray Model 3300 terminal may be utilized with the present system. In place of or in addition to terminal **16,** a teleprinter **48** such as the ASR **33** teletype may be used. The buffered teleprinter input/output **2752A** and the **12531B** interface kit manufactured and sold by Hewlett Packard of Palo Alto, Calif. may be advantageously utilized with the present system.

The cathode ray tube display screen **18** may comprise, for example, the graphic terminal Model 2D2 manufactured and sold by Vector General of Canoga Park, Calif. The data tablet **22** may comprise the Model GT**50** unit manufactured and sold by Computek, Inc. The sixteen key function box **20** will be subsequently described in detail. The function box **20** and data tablet **22** are interconnected with the central controller **12** through an interface **50,** which will be subsequently described.

The marker plotter **26** may comprise, for example, the Series **2000** Modular Automated Drafting System manufactured and sold by Xynetics, Inc. of Canoga Park, Calif. As shown in FIG. **1,** the plotter **26** includes a bar **52** which is movable along the length of the plotting area, along with a writing stylus **54** which is movable along the bar **52.** The bar **52** and the stylus **54** are driven by electrical motors according to X and Y axis control signals generated from the central controller **12.** The HP12566B computer interface module may be connected between the central controller **12** and the marker plotter **26** in a conventional manner. Control of the plotter **26** is afforded by operation of buttons on the control console **57** on the plotter. Movement of the stylus **54** enables the drafting of pattern pieces in the marker area at full size. Patterns are plotted on a long roll of paper **56** which may then be removed and placed on a stack of fabric for cutting a large number of patterns.

FIG. **3** is a top view of the digitizer function box **40.** The function box includes 16 keys **60** which comprise conventional two position pushbutton switches which may be manually operated by the operator. Switches **60** are utilized to input the piece identification number, a new piece flag, and end piece flag, sample size, mirror flag, grade rule number and error corrections into the central controller **12.** The functions for the 16 keys are as follows:

1/S — used to designate either the number 1 or the letter "S" for small, depending upon the state of the alpha button;

0/+ — used to designate the numeral 0 or "+";

2/M — used to designate the numeral 2 or the letter "M" to designate medium;

3/L — used to designate the numeral 3 or to designate the letter "L" for large;

4/X — used to designate the numeral 4 or the letter "X";

5/H — used to designate the numeral 5 or the letter "H";

6/T — used to designate the numeral 6 or the letter "T";

7/A — used to designate the numeral 7 or the letter "A";

8/B — used to designate the numeral 8 or to designate the letter "B";

9/C — used to designate the numeral 9 or the letter "C";

"New Piece" — used to indicate that a new piece is about to be digitized;

"End Piece" — used to indicate that the pattern piece has been completely digitized;

"Rub Out" — used to erase data entered by the pressure of a wrong key;

"Alpha" — depressed when it is desired to input letters or the + sign by depression of the corresponding keys **60**;

"Mirror" — utilized to indicate that the pattern being digitized is a mirror piece; and

"Size/D" — utilized to indicate that the pattern being digitized is the sample size or to indicate the letter "D".

Above the sixteen key **60** is a speaker **61** which emits an audible tone when a key is depressed and the central controller **12** receives the generated signal. An error light **62** is also provided which is illuminated when a gross input error is made. The light **62** remains illuminated until the error is corrected. Detailed description of the electrical circuitry of the function box shown in FIG. **3** will be subsequently described.

FIG. **4** is a top view of the cursor control box **36** and the cursor aperture member **38.** As previously noted, box **36** and member **38** are commercially available as a portion of the Autotrol digitizer system. The cursor control box **36** includes five manually operable button key switches **64–72** and a switch dial **74.** The operator may input grade points, intermediate points and notches by proper operation of the three keys **64, 66** and **68.** The key switch **70** and the switch dial **74** are used in conjunction with one another. The switch dial **74** indicates what data is being input when the switch key **70** is depressed. The switch key **70** and dial **74** are thus utilized to input the grain line, stripe line, reference line, punch holes, internal points, alternate start point, transfer function point, center lettering point, and fixed intermediate point. The zero key **72** is utilized to input a starting point which is utilized in the digitizer maintenance test and in various tests utilizing grade increments. The operator can tell whether or not the keys depressed are functioning properly by a single beep given out by speaker **61** when a key is depressed.

The following is a list of the symbols about the switch dial **74** and their meanings:

Symbol	Notation
↔	Grain line
‖	Stripe line
⊕	Reference line
+	Punch line
I.P.	Internal point
*	Alternate start point
T.P.	Transfer function point
LET.	Center point for lettering
F.I.	Fixed intermediate point

Attached to the right hand side of the box **36** is the cursor aperture member **38** which includes a plate **76** having a circular aperture **78** formed therethrough. Crosshairs **80** extend across apertures **78** and are utilized to sight on the centers of grade points, the ends of lines and the perimeter of the piece. A knob **82** is utilized by the operator to move the cursor aperture member **38** to the desired location.

FIG. **5** illustrates a perspective view of the function box **20** for inputting instructions associated with the operation of the data tablet **22** and the display **18.** The function box **20** includes sixteen pushbutton keys **90** which extend upwardly from a generally planer face **92** of a console housing. A stop button switch **94** is positioned above the button keys **90.** The face **96** of the housing of the function box is adapted to face the operator and electrical signals generated by the function box are transmitted via a conductor **98** to the central controller **12.** The function box **20** may be utilized during grade checking or marker preparation, and each of the checking and grading functions utilizes a different template which fits over the keys **90** and covers the area generally designated by the dotted line **100.** Each template bears different legends, as will be subsequently described. As shown in the broken away portion of the function box **20,** a printed circuit board **102** and an electrical panel **104** are supported by conventional means within the function box. Electrical construction and operation of the function box will be subsequently described in greater detail.

FIG. **6** illustrates the template **106** which is utilized with box **20** to perform grade checking functions. The template **106** comprises a thin plastic sheet having 16 apertures formed therein, each aperture positioned to receive one of the button keys **90.** Depression of the 10 buttons labeled **0–9** enable the rule table number desired to be checked to be entered into the central controller **12.** For example, to enter the rule number table "128", the buttons marked "1", "2" and "8" are sequentially depressed. The button labeled "List Rule" causes the rule table number to be displayed under the list of rule tables on the display **18.** The "Change Rule" button enables the rule number to be changed to another grade rule. The button labeled "Accept" indicates that the graded piece is accepted and re-entered into the central processor. The button labeled "Reject" is utilized to reject the displayed graded pattern. The "Plot" button may be depressed to request the display of a nested plot of the pattern piece in various grades. The button marked "Area" may be depressed to list the area and perimeter of each size of the nested plot. Button **94** marked "Stop" may be operated with various ones of the command buttons to give control back to the CHECK program of the system, so that the commands designated prior to the depression of the stop button may be performed by the system.

FIG. **7** illustrates a template for placing over the buttons **90** on the function box **20** when it is desired to perform marking procedures. Template **108** comprises a plastic sheet having sixteen apertures defined therein for receiving the buttons **90** and **94.** Labels or legends are defined under each of the apertures to enable the accomplishment of various functions in the marking procedure. Depression of the cutton marked "Pick" enables the operator to "Pick Up" a pattern piece displayed on the display **18** so that the pattern may be moved on the display by operation of the cursor. Depression of the "Overlap" button enables the position of the "Picked" pattern piece to be adjusted so that it does not overlap any surrounding pattern pieces on the display screen. The button marked "Bump Down" will be used to adjust the position of a Picked piece by moving it down until it hits another piece or the marker edge. The "Bump Left" button adjusts the position of the Picked piece by moving it left until it hits another piece. The button marked "Tilt CCW" rotates the Picked piece in a counter-clockwise movement is steps of 0.5°. The button marked Tilt CW rotates the Picked piece in a clockwise direction in steps of 0.5°. The Release button allows a Picked pattern piece to be moved off the marker area on the display screen **18.** The "Spe-

cial" buttons turns "Special Functions" on and off on the display screen. The "Flip" button turns the Picked piece over. The "Bump Up" button adjusts a Picked piece by moving it up until it hits another piece or the marker edge. The "Bump Right" button moves the Picked piece right until it hits another piece or the marker length line. The "Rotate CCw" button causes the Picked piece to be rotated in a counter-clockwise direction in steps of 45°. The "Rotate CW" button causes the Picked piece to rotate in a clockwise direction in steps of 45°.

The "Pick Check" button functions the same as the Pick button, except that the desired piece does not jump to the cursor and the piece will not move along with the cursor. The "Vertical Stripe Lock" button adjusts the position of the picked piece on a vertical stripe line so that the stripe line of the picked piece aligns with the closest stripe line on the marker. The "Horizontal Lock" button has the same function as the Vertical Stripe Lock, except that the stripe to be locked on must be horizontal. The "Stop" button **94** is utilized in the checking procedure.

FIG. **8** illustrates a schematic diagram of the circuitry utilized by both the cursor function box **40** and the function box **20.** As each of the function boxes **20** and **40** contain sixteen primary keys, the same electrical circuits may be utilized for both boxes. The circuitry includes 16 pairs **110–140** of cross coupled NAND gates to provide debounce circuitry for the keys. Each pair of cross coupled NAND gates includes two input terminals which are tied to the sixteen pushbutton switches of the function box. Upon depression of a button switch, the output of its respective cross coupled NAND gates goes logically low and when the pushbutton switches are released, the output goes logically high. The output of each of the 16 cross coupled NAND gate pairs are applied to a pair of diode matrixes **142** and **144** comprise, for example, the diode matrix TIDM286 manufactured and sold by Texas Instruments, with a predetermined address code burned into the matrix. As an example of the code, the output of each of the cross coupled NAND gates is illustrated in FIG. **8** with a binary address code at the output thereof. Upon depression of the button switch associated with each of the cross coupled NAND gate pairs, that particular binary code is output from the diode matrixes **142–144.** For use with box **20,** a seventeenth pair of cross coupled NAND gates **146** is provided with input terminals which are connected to the seventeenth pushbutton switch corresponding to the Stop button **94.**

The outputs of the diode matrixes **142** and **144** are tied together and applied to the input of four AND gates **148–154.** The output of gate **148** comprises bit three, the output of gate **150** comprises bit two, output of gate **152** comprises bit one and the output of gate **154** comprises bit zero of the address code output. The four bits from gates **148–154** comprise a binary address code having sixteen positions, each position relating to a different one of the sixteen button switches on the function box. This address code is provided to the central controller which implements the selected function.

The outputs of the diode matrixes **142** and **144** are connected to a multi-input NAND gate **156,** the output of which is applied through a NAND gate **158** to the inputs of an OR gate **160.** The output of OR gate **160** and an AND gate **162** are utilized to control the operation of a one shot **164.** The one shot **164** may comprise, for example, a SN74141 multivibrator. The output of the one shot **164** is applied through an AND gate **166** as a strobe signal for strobing the bits zero-three. A lamp **168** is provided to be energized from a signal from a central controller in order to indicate gross errors in the case of the function box **40.**

FIG. **9** illustrates circuitry for creating an audible sound upon depression of the pushbutton switches of the function box **40.** A one shot **170** receives signals from the central controller and stretches the pulse received therefrom by an amount sufficient to generate an audible sound via a speaker **172.** In operation, depression of the buttons of the function box creates a logic low signal at the output of the associated cross coupled NAND gates, which operate through the diode matrixes **142** and **144** in order to generate the unique address code at the output of gates **148–154.** The central controller **12** recognizes the generated output and if the address is correct, the central controller generates a signal which is applied to the one shot **170** in order to cause an audible sound to be emitted from the speaker **172.** This indicates to the operator that the system is operating satisfactorily. The one shot **170** and speaker **172** are mounted within the function box casing.

FIG. **10** is a schematic diagram of the interface circuitry between the data tablet **22** and the central controller **12.** Operation of the stylus **24** relative to the data tablet **22** causes generation of three types of digital data signals. Ten bits of X coordinate data are applied to the ten terminals **180,** while 10 bits of Y coordinate data are applied at terminals **182.** Status bits are applied to terminals **184** from the data tablet. The X and Y coordinate data bits are supplied to inputs of 4-bit buffer chips **186, 188, 190** and **194.** A suitable buffer chip may comprise the **370CJ** buffer chip. The output of the chips **186–194** are applied through diodes to buffers **196, 198, 200, 202, 204** and **206.** The voltage level of the X and Y coordinate data signals is increased from 5 volts to 12 volts by the buffers **186–194** in order to eliminate low level noise. The logic level of the coordinate data signals is reduced to conventional TTL logic level at the buffers **196–206.** The output of buffers **196-206** are applied to inputs of ten AND gates **208.**

A timing circuit, including cross coupled NAND gates **210** and **212** and interconnected flipflop counters **214** and **216,** operates to generate timing signals for the switching of the coordinate data. Counters **214** and **216** may comprise, for example, a SN7476 counter circuit. The output of the timing circuit is applied via leads **218** to four NOR gates **220.** The outputs of NOR gates **220** comprise a pair of enable lines **222** which selectively enable X and Y data to be alternatively transmitted from the buffers **196–206** through the AND gates **208.** The output of the AND gates **208** comprise ten bits of either X or Y coordinate data or status bits. This ten bit signal is applied to the central controller **12** for operation as will be subsequently described.

The status bits generated from the data tablet **22** are applied through convertors **224** to inputs of buffers **226** and **228.** The output of buffers **226** and **228** are also connected to the AND gates **210** in order to enable the output of status bits. Other portions of the timing circuitry comprise a three input NAND gate **230** which is connected to an input of a multivibrator **232** which op-

erates to generate timing signals for enabling of the status bits. Terminal **234** is connected between counter **216** and multivibrator **232** and is tied to the remaining portion of the interface circuitry shown in FIG. **11.**

An output of counter **214** is applied to inputs of an ANd gate **238,** the output of which is applied to AND gates **208.** Gate **238** is connected through an ANd gate **240** to a terminal **242** which supplies timing signals to portions of the system. In addition, terminals **244** and **246** supply timing signals to portions of the interface circuitry shown in FIG. **11.**

One input of gate **240** is applied through NAND gate **248** and gate **250** and through invertor **252** to inputs of four flipflops **254, 256, 258** and **260.** Gates **248** and **250** and invertor **252** operate as a strobe generator for operation of the bezel lights on the display screen **18,** to be subsequently described. Terminals **262** are connected to receive command signals from the central controller **12** for selective operation of the bezel lights on the display screen **18.** Terminals **262** are connected to inputs of eight ANd gates **264,** the outputs of which are connected to flip-flops **254–260.** The output of flip-flops **254–260** are applied through eight NAND gates **266** which generate signals for illuminating the bezel lights shown in FIG. **12.** Specifically, the designated NAND gate **266** is operated such that its output becomes logic low in order to provide a ground to the bezel lights. The other terminals of the bezel lights are connected to a source of positive voltage such that the lights are illuminated upon the occurrence of the logic low at the output of one of the gates **266.**

FIG. **11** illustrates additional circuitry in the interface circuits connected between the data tablet **22** and the central controller. The circuitry shown in FIG. **11** is primarily flag circuitry for providing priority of I/O data to the central controller.

Referring specifically to FIG. **11,** conventional input signals in Hewett Packard language are input from the central controller to the terminals **280.** The command signals are applied through nine NAND gates **28***a–i.* The output of NAND gate **282***a* is applied through NAND gate **284,** the output of which is applied to a four input NAND gate **286.** The output of NAND gate **282***b* is applied through NAND gates **288** and **290** to an input of a flag flipflop circuit **292.** The output of gates **282***c* and **282***d* are applied to the inputs of a flag buffer flipflop **294.** An output of the flipflop **294** is applied to terminal **234** shown in FIG. **10.** An output of gate **282***e* is also applied to flipflop **294.** The output of gate **282***f* is applied to flipflop **294** and to flip-flop **292.** The IEN input is directly applied to an input of a NAND gate **296** which receives an output from flipflop **292.** An output of gate **282***g* is applied to terminal **246** shown in FIG. **10** and is also applied to an input of a control flipflop **298.** Outputs of gates **282***h* and **282***i* are applied to the flipflop **298.** The output of flipflop **298** is applied to an input of NAND gate **296.** The SSS input is applied directly to an input of a NAND gate **300,** while the SSC input is directly applied as an input of gate **302.** The output of gates **300** and **302** are applied through an NAND gate **304** as an input of NAND **306.** The output of gate **306** comprises the Skip Flap Signal (SSK) which is applied to the data tablet **22.**

LSCM, LSCL and IOG signals are applied through a NAND gate **308** and a NAND gate **310** to terminal **242** shown in FIG. **10** and also to an input of gate **302.** The output of flipflop **292** is applied via lead **312** to an input of a NAND gate **314** which generates the service interrupt request (SRQ). An input of gate **282***e* is applied as an input to a NAND gate **316** which generates the interrupt request (IRQ) signal. The output of NAND gate **286** is applied as an input, along with an input from gate **288** to the IRQ flipflop **318.** The output of flipflop **318** is applied to a NAND gate **320** which generates the flag signal (FLG). The output of NAND gate **296** is applied to inputs of NAND gate **324** and to an input of a NAND gate **326** which generates the priority low (PRL) signal.

FIG. **12** illustrates the light switches **19** which are disposed at the lower portion of the frame, or bezel, of the display screen **18** previously shown in FIG. **1.** The lights **19** include the Input Required light **340** which becomes illuminated when the display screen **18** requests information during marking procedures or when the special functions of the system are on. The Busy light **342** becomes illuminated when the system is performing a marking function and is busy, thereby locking the operator out of the system. The Overlap light **344** becomes illuminated when an overlap pattern piece cannot be positioned in the position desired without overlaps. The Shade Violation button light **346** becomes illuminated when a piece is not placed in its proper shade area, as will be latter described. The Maximum Tilt light **348** becomes illuminated when a pattern piece with a tilt restriction is tilted to that limit. The Error light **350** becomes illuminated when an error occurs in a marking procedure.

The step by step operation of the present system will now be explained in detail. Briefly, a grade rules program is built and is stored in the central processor **12.** Individual pattern pieces are then digitized on the digitizer **10** and the coordinate data and grading instruction data relative to the pattern pieces and stored in the central controller **12.** Nested plots of various sizes of a pattern piece may then be displayed on the display screen **18** and checked by the use of the CHECK program. The style and size of the patterns desired to be marked are then punched into MONITOR punch cards and fed through the card reader **14** into the system. In addition, other operating instructions for the system are input into the system through the card reader **14** through draw statement cards, plot cards, style cards and the like. In the initial use of the system, optional features of the system may be selected by operation of the POPTS program by typing in responses requested by the system into terminal **16.** The desired markers are then constructed on screen **18** by operation of the function box **20** and the stylus **24.** Error messages may be displayed to the operator during operation of the system in case of errors in operation. After preparation of a miniature marker on the display screen **18,** the plotter **26** is operated to provide a full scale reproduction of the desired marker. The marker may then be utilized in the conventional manner and a plurality of layers of material are cut to enable fabrication of a plurality of garments of the desired style and size. Each of the above-noted steps will now be described in detail.

BUILDING GRADE RULE TABLES

Prior to use of the system shown in FIG. **1,** grade rules must be input into the system. These grade rules will be stored and will be applied to digitized points of pattern pieces as specified during the digitizing procedure.

A grade point rule defines the X and Y increments for grading the different sizes at a grade point. There are several ways of determining the X and Y coordinates for a grade point rule. One method of determining these increments is for the grader to determine what they are by hand. The second method is to digitize each size of a grade point on a nested plot. The routine for building a grade rule table with the present system is termed the DRULE program. Below are the operating procedures for DRULE and an example of a DRULE grade rule listing.

1. Place the piece to be used on the digitizer table so that the reference line is in a true horizontal orientation.
2. Type in on the console typewriter **16** the following:
   :JOB cr/lf
   :PR,DRULE cr;lf
3. On the digitizer function box **40,** punch in the rule number. This can be from 1 to 3 digits.
4. Position the cursor aperature member **38** over the grade point of the sample size and push the ZERO button on the control box **36.** At this time the printer will print:

Rule number N			
coordinate points in 32nds		coordinate points in 64ths	
x	y	x	y

5. Position the cursor aperture member **38** over the first point (smallest size) to be graded and push the button labeled NOTCH. The printer **16** will print the relative coordinates in both **64**th and **32**nds.
6. Repeat step **5** until all of the points (smallest to largest size) for that rule have been printed.
7. To do the next rule, go back to step **3.**
8. To stop the program, turn on bit 0 of the central controller **12.**

The console typewriter **16** will print DRULE STOP **0000.**

EXAMPLE OF DRULE GRADE POINT LISTING

Rule Number 2

Coordinate Points in 32nds		Coordinate Points in 64ths	
X	Y	X	Y
−32	0	−64	0
−24	0	−49	0
−16	0	−31	0
−8	0	−16	0
0	0	0	0
8	0	16	0
16	0	32	0
31	0	62	0
46	0	92	0
55	0	111	0
64	0	129	0
73	0	147	1

After the increments (X and Y) have been determined for the grade rules, a Grade Rule Table has to be built and input into the system. To build a grade rule table, the X and Y increments have to be keypunched into IBM computer cards.

In the preparation of the computer cards, the following rules are followed:

1. All X increments and Y increments have to be keypunched into separate cards with the X card first then the y card for each grade rule.
2. Each increments must be separated with one space.
3. If there is insufficient room on one card to punch the increments for all sizes, a slash must be punched in column **80** on the card and the rest of the increments may continue on another card. There is no limit on the number of continuation cards.

The transfer function value used occasionally in the digitizing procedures to increase or decrease the size of a piece has to be put on the rule cards where the value will be applied. After all increments for all sizes have been punched into a card, the transfer value should be punched in. The transfer function value is to be the last value on a rule card. If a transfer value is not put on a card it is assumed to be zero.

The $RULE: statement is used to build and list rule tables. Rule tables are keypunched onto computer cards and stored in the central controller **12** via the $RULE: card. Each rule table must have a unique name (without imbedded blanks) which will be used to reference the table with other commands.

The rule table indicates to the system the precision of the grade increments to use in grading. The following grade increments are allowed:

00032 = 1/32″
00064 = 1/64″
00128 = 1/128′
00256 = 1/256′
00010 = 1 millimeter — for metric grading
00100 = 1/10 millimeter

To build a rule table the card deck structure shown in FIG. **13** must be used. FIG. **13** illustrates an 80 column coding sheet used to layout data to be keypunched onto an 80 column conventional computer card. Each column of the coding sheets corresponds to a column on the computer card. A description of each card follows:

1.	$RULE:	Indicates to the monitor that a rule table is to be built or listed (all or part of it).
2.	OPERATION CARD	Indicates to the rule program the type of operation to perform.
	BUILD	Columns 1–5 (FIG. 13) BUILD A RULE TABLE
3.	NAME CARD	Identifies the rule table, the grading increment to use, the number of sizes and the number of rules in the table. The numbers in columns 18–21 and 22–25 (FIG. 13) are compared to the number of sizes and rules that are actually read. If these numbers are not the same, a diagnostic will occur. If columns 18–21 and 22–25 are left blank this comparison check will not occur.
	RULE NAME	Columns 1–10
	GRADE INCREMENT	Columns 13–17
	NUMBER OF SIZES	Columns 18–21
	NUMBER OF RULES	Columns 22–25
4.	SIZE CARDS	Identifies each size and the order of sizes in the grade rules. Only one size per card is allowed.
	SIZE NAME	Columns 1–6 (FIG. 13) 6 Character size name.
5.	TERMINATOR CARD /E	Terminates the size cards. Columns 1–2 (FIG. 13)
6.	GRADE RULE CARDS	Defines the increments for each size for a given grade point. As shown in FIG. 13, these cards are punched in a free-field format. Free field format means that increments are separated by blank spaces.
7.	999	End of Rule Deck.

To list a specific rule table the card deck structure shown in FIG. **14** may be used. A description of each card follows:

1.	$RULE:	Indicates to the monitor that a rule table is to be built or listed.
2.	LIST	Indicates to the rule program the type of operation to perform.
	LIST	Columns 1-4 (FIG. 14) LIST A RULE TABLE
3.	NAME CARD	Identifies the rule table to be listed.
	RULE NAME	Columns 1-10 (FIG. 14)

To list a specific rule or rules the card deck structure shown in FIG. **15** may be used. A description of each card follows:

1.	$RULE:	Indicates to the monitor that a rule table is to be built or listed.
2.	LIST	Indicates to the rule program the type of operation to perform.
3.	NAME CARD	Indicates the rule numbers of a specific rule table to be listed. Columns 11-80 can be used to specify rule numbers for listing. There can be no imbedded blanks. The first value indicates the number of rules to be listed. Each rule is to be separated by a comma. As many rules can be requested as can be fit into columns 11-80. There are no continuation cards.
	RULE NAME	Columns 1-10
	RULES SPECIFIED	Columns 11-80

To list all rule table names the card deck structure shown in FIG. **16** should be used. A description of each card follows:

1.	$RULE:	Indicates to the monitor that a rule table is to be built or listed.
2.	*ALL	Indicates to the rule program that a list of rule table names should be provided.
	*ALL	Columns 1-4

Digitizing Procedures

When the desired grade rule table has been loaded into the system as previously noted, pattern pieces may be digitized by the digitizer **10** and coordinate data and grading instructions for the pieces may be stored by the central controller **12.**

The operator of the digitizer **10** is responsible for entering all the necessary data required to build a data file for each pattern piece via the digitizer. The data used to build a piece data file consists of:

1. the piece identification number;
2. the sample size of the piece;
3. the mirror indicator if needed;
4. the reference line;
5. all internal piece information concerning the stripe line, grain line, punch holes, internal points, alternate start point, and center point for lettering if they are present; and
6. the grade points, intermediate points, transfer function point, and fixed intermediate points.

The patterns that are to be digitized should have all the information that is required to make that piece useful in the final marker. This information will be on the piece before it reaches the digitizer **10.** The operator should keep two to three styles ahead of the styles required by the marker maker in order to reduce the majority of the unnecessary delays in marking future markers.

As shown in FIG. **17,** the pattern pieces are placed on the digitizer table **34** with the pattern reference line in a horizontal position and the pieces are secured with tape. Although a plurality of pattern pieces is shown on the table **34** in FIG. **17** for clarity of illustration, in practice an operator will usually digitize one pattern piece at a time.

Referring to FIG. **17,** pattern pieces for a pair of pants are illustrated. Piece **P1** is a waist band pattern. Piece P2 is a left fly pattern and piece **P3** is a right fly pattern. Piece **P4** is a pocket facing pattern and piece **P5** is a pocket lining pattern. Pieces **P6** and **P7** are panel patterns. Piece **P8** is a front pocket shaper, piece **P9** is a back pocket shaper, piece **P10** is a back pocket pattern and piece **P11** is a front facing pattern.

After positioning the pattern piece, the operator is now ready to enter the data for the piece. For each piece certain steps must be taken.

In performing these steps, the operator uses the digitizer function box **40** and the cursor **38.** As previously noted, the digitizer function box **40** has 16 keys **60** which are used to input the piece identification number, new piece flag, and end piece flag, sample size, mirror flag, grade rule number, and error corrections. The cursor control box **36** has five keys and a switch dial **74.** The operator can input grade points, intermediate points, and notches with the three leftmost keys on the box **36.** The switch key (bottom center) and the switch dial **74** are used in conjunction with each other. The switch dial **74** indicates what data is being input when the switch key is pressed. This switch key and dial are used to input the grain line, stripe line, reference line, punch holes, internal points, alternate start point, transfer function point, center lettering point, and fixed intermediate point. The zero key is used to input a starting point which is used in the digitizer maintenance test and various tests used to document grade increments. The operator can tell if the buttons depressed are functioning properly by the single beep given from the speaker **61** when the button is depressed.

Operation of the digitizer **10** on a typical pattern piece will now be described. The function box **40** button "New Piece" is depressed. Until the New Piece button is depressed, all data entered is ignored. Depression of the button sets up a new data file. The piece identification number is entered via the function box **40.** Each piece identification number is a 10 digit number. The number is entered by depressing the key **60** on the function box **40** that corresponds with each of the 10 digits. If an identification number of less than 10 digits is input and the operator proceeds to the next step, the error light **62** will be turned on. To correct this error, the operator has to return to the first step. All data input while the light **62** is on will be ignored.

The sample size of the piece is now entered into the function box **40.** The sample size can be no more than size alpha numeric characters long. If more than six characters are input, the first six are accepted and the remainder are ignored. The sample size is the size from which grading will start. This can be input by depressing the Size button and the number buttons corresponding to the digits in the size. If the letters A, B, C, D, H, L, M, P, S, T and/or + are to be input in the sample size, such as for size **32S,** the operator should for

each letter depress the alpha key and then the appropriate letter key.

At this point, the "Rub Out" button on the function box **40** can be used. If the operator depresses a wrong key, he depresses Rub Out; and the wrong data entered will be erased. For example if a 3 were entered, it could be rubbed out and replaced with a 4 by depressing the Rub Out button and the 4 button. If a grade point were entered, it could be erased and replaced with an intermediate point by depressing the Rub Out button and the "Int. Point" button. For example, if a grade point rule number of 136 were entered, it could be replaced with 126 by depressing the Rub Out button twice, the 2 button, and the 6 button. The Rub Out will erase the last entry or may be used repeatedly to erase the last several entries made from either the function box or the cursor.

If a piece is symmetrical about the center piece line, it can be digitized as a mirrored piece. Only one-half of mirrored pieces are digitized, thereby reducing the amount of data input required for the data file. The grading program generates the reflection of the top half and thereby assures a perfectly symmetrical piece. The half being digitized has to be the top half, the reference line has to be the piece center line, and the reference line has to be horizontal. A mirrored piece is digitized using the same steps as a non-mirrored piece except that the Mirror button has to be depressed.

Next, a horizontal reference line has to be entered for each pattern piece. The reference line is used by the system to define the horizontal line (X-axis) used in grading the piece being digitized. This may be input by turning the switch dial **74** to the ⊕ position and positioning the cursor cross hair **80** at the left end of the reference line. When the cross hair is positioned, the Switch button **70** (FIG. **4**) is depressed. Repeat the same procedure at the right end of the reference line. The reference line will determine the orientation of the piece when it is used in a marker, unless there is a stripe or grain line.

If a stripe line or lines are shown on the pattern piece, the switch dial **74** is rotated to the symbol ||. For a horizontal stripe line, the center of the cursor cross hair **80** is positioned at the left end of the stripe line. The Switch button **70** is depressed, and the grade rule number applied to the stripe line is entered through the function box **40.** The procedure is repeated at the right end of the stripe line. For a vertical stripe line, the procedure is started at the bottom end of the stripe and continued to the top end. The first stripe line will determine the orientation of the piece in a marker, unless a grain line is present.

If a grain line is shown, the switch dial **74** is rotated to the symbol ↔. The cursor cross hair **80** is positioned at the left end of the grain line and the Switch button **70** is depressed. The cursor cross hair **80** is moved to the right end and the procedure repeated. The grain line will determine the orientation of the piece. If a grain line is in a piece, the piece will rotate in 180° increments, instead of 45° increments when making a marker with this piece in it.

If punch holes are present, the switch dial **74** is rotated to the symbol +. For each punch hole, the cursor cross hair **80** is positioned at the center of the punch hole and the Switch button **70** is depressed. Next, the rule number to be applied to the punch hole is entered via the function box **40.** If the rule number used for the next punch hole is the same as the rule number previously entered, then it needed not be repeated. This method of omitting rule number is referred to as allowing the rules to be "Modal."

If internal points are present on the pattern piece, rotate the switch dial **74** to the symbol "I.P." Internal points are grouped in pairs and must be digitized as such. Position the cursor cross hair **80** at right or bottom one of the internal points and depress the Switch button **70.** Enter the rule number applied at this point via the function box **40.** Position the cursor cross hair **80** at the other internal point in the pair and depress the Switch button **70.** If necessary, enter the rule number applied to this internal point via the function box **40.**

An alternate start point is sometimes placed on the perimeter of a pattern piece that will be used in a marker. The marker will be cut by some kind of mechanical cutter. The alternate start point allows the machine to move from one starting point to the nearest starting point, or alternate starting point of another piece, in order to save time in cutting. If an alternate start point is shown, rotate the switch dial **74** to the symbol *. Position the cursor cross hair **80** at the alternate start and depress the Switch button **70.** Enter the rule number to be applied at the alternate start point via the function box **40.**

The transfer function point is optional and is used to move a grade point in the sample size to another point before it is graded. The increment used in determining the new position is the extra increment added to a grade rule. This option is useful is grading normal size patterns into children's patterns. The transfer function point is applied by turning the cursor switch dial **74** to "T.P.", depressing the Switch button **70** and applying a grade rule through the function box **40.**

A center point for lettering must be entered for every piece. This point is used to center all lettering inside the piece perimeter. This point is usually placed near the center of the piece. A grade should be applied to a lettering point if that point is to stay in one position. To enter this type point, rotate the switch dial **74** to the symbol "LET". Position the cursor cross hair **80** and depress the Switch button **70.** A rule may be applied to this point, but it is optional.

The pattern piece outline must be entered in a clockwise direction around the piece starting at the point labeled as the start point. The rules are entered following the Modal method previously described, except for the start point. The start point must be followed by a grade rule. Six types of data can be entered in describing the piece outline: a grade point, a transfer function point, an intermediate point, a fixed intermediate point, a notch and the ending point of the piece perimeter.

The grade points are entered by positioning the cursor cross hair **80** at the grade point and depressing the cursor box button "Grade Point." If required, the rule number for the grade point is entered via the function box **40.**

The transfer function point was previously explained. This function is normally used on grade points on the perimeter of the piece. The transfer function point is entered and the rule applying to the point is entered. This will move the grade point to a predetermined point, and the point is then graded.

Intermediate points are digitized between grade points if the section of the outline between these grade points is curved. The digitized intermediate points are

used to define curved lines between grade points. The number of intermediate points entered is left up to the digitizer operator and should be chosen on the basis that for every intermediate point entered, two more points should be placed along side of the intermediate point. More points are needed where the curve of the line is the greatest.

The intermediate point is entered by positioning the cursor cross hair **80** at the point and depressing the cursor button "Int. Point." A grade rule is not to be entered on an intermediate point.

The system has the capability to eliminate any point that is found to be too close to another point. This function is called data suppression.

At times, the grader may wish the intermediate points to be exempt from data suppression. To enter intermediate points exempt from data suppression, the operator has to enter fixed intermediate points. To do this, the switch dial **74** is turned to the symbol "F.I.." The cursor cross hair **80** is aligned on the piece perimeter and the Switch button **70** is depressed. Where an intermediate point would normally have been entered along the perimeter, a fixed intermediate point may be entered.

To enter a notch occurring on the pattern piece, position the cursor cross hair **80** at the output position of the notch. This point is entered as a grade point as previously disclosed, or as an intermediate point. The cursor cross hair **80** is then positioned at the inner position of the notch and the cursor button "Notch" is depressed. The operator then proceeds to the next point to be digitized.

After all data for a piece has been entered correctly, the "End Piece" button on the function box **40** is depressed. This terminates the piece data and any data entered is ignored until digitizing steps are repeated for a new piece. A piece has to be ended after the last grade point or intermediate point.

FIG. **18** shows a mirrored pattern piece in the proper position to be digitized. The reference line is horizontal and a vertical stripe line is provided. Points which are digitized are marked on the drawing by numbers and "X's". The following explains how the piece was digitized:

1. depress New Piece;
2. enter the piece identification;
3. enter the sample size;
4. this is a mirrored piece, so the Mirror button is depressed;
5. enter the stripe line;
6. enter the internal points;
7. enter the lettering point;
8. enter the data concerning the perimeter (points with numbers are grade points, and the X's on the perimeter are intermediate points);
9. end the piece data description.

A piece that is to be digitized as a mirrored piece has to have the body of the piece being digitized above the reference line. FIG. **19** is an illustration of a non-mirrored pattern piece in position to be digitized. The piece was digitized as follows:

1. New Piece button is depressed;
2. piece identification is entered;
3. sample size is entered;
4. reference line is entered;
5. stripe line is entered;
6. punch hole is entered;
7. lettering point is entered;
8. enter the data concerning the perimeter (points with numbers are grade points, and the X's on the perimeter are intermediate points);
9. end the piece data description.

Check Program Procedures

When pattern piece data has been input into the system through the digitizer, the phrase "MARK OR CHECK" will be periodically flashed on the display screen **18.**

The CHECK program is used to verify the accuracy of pieces before they are used in markers. When the question "MARK OR CHECK" displayed on screen **18** is answered by moving the stylus **24** to a predetermined position and depressed, thereby selecting CHECK, control is transferred to the CHECK program which displays a nested (or stacked) grade of the various sizes of a single piece. A typical nested display in the CHECK mode is shown in FIG. **20.** The accuracy of grade rules and digitized data are verified by the check program. While using CHECK, grade rules may be corrected, pieces may be rejected, accepted, or left in the digitizer files for further examination, nested plots can be requested, and the area and perimeter length of pieces can be calculated.

The CHECK program examines the digitizer files, and when piece data is found in a digitizer file, the question

CHECK XXXXXXXXXX?
YES NO
EXIT

is displayed on the display screen **18**, where the ten X's represent the 10-digit piece identification number. If the operator does not wish to check the piece, the stylus **24** is moved until the cursor on screen **18** representing the position of the stylus is positioned over the displayed "NO." The stylus **24** is then activated with a short burst of pressure on the data tablet **22** at the NO position. The next piece identification number will then be displayed on screen **18.** If the stylus **24** is held on the NO with a steady pressure on the data tablet **22** to keep the stylus active, the operator is able to search through each file in quick succession. When the operator reaches the end of the digitizer files, the message

ALL DIGITIZER FILES HAVE BEEN SEARCHED

is displayed and control returns to the MONITOR program. The question MARK OR CHECK is then displayed once again on screen **18.** When CHECK is again called, the digitizer file search begins at the first digitizer file. If the operator wants to get back to the first of the digitizer files without examining all the files, he may activate the stylus on the displayed "EXIT." Control is returned to the MONITOR; and when CHECK is next called, the digitizer file search will begin at the first file.

When a piece is to be checked, the operator should activate the stylus on the displayed "YES." When checking pieces, the CHECK function box template **106** shown in FIG. **6** should be placed over the function box buttons **90,** as shown in FIG. **20.** The template **106**

specifies the functions of the buttons used in the CHECK program, as previously described.

The rule table which is to be applied to the piece must be specified by the operator. The rule tables (up to 128), the piece identification and the message (specify rule table) will be displayed on screen **18** as follows:

PIECE I.D. SPECIFY
1 RULE TABLE NAME
2 RULE TABLE NAME

The operator specifies the rule table by entering the rule number into function box **20.** The rule table number is the number to the left of the rule table name of the screen. Entry into box **20** is done by:

1. Pressing the number buttons on box **20** to indicate the rule table number. Example: Press the button marked 2 for rule table number 2; press the buttons marked 1, 2 and 8 for rule table number 128.
2. Press the button marked "LIST RULE." This will cause the rule table number to be displayed under the list of rule tables. If the number is incorrect, repeat step 1 and step 2. If three X's are displayed, the operator must re-enter the correct rule table number. The X's represent a nonexistent rule table number.
3. Press the STOP button. The piece is then graded and displayed as a nested plot as shown in FIG. **20.**

Optionally, the operator may accept or reject a piece when he is asked to specify the rule table. When the CHECK system asks the operator to specify the rule table, the operator may reject the piece without viewing it on the display by:

1. Pressing the REJECT button.
2. Pressing the STOP button.

If the operator wishes to accept a piece without viewing it on the display, he should:

1. Input the rule table number.
2. Press the LIST button.
3. Press the ACCEPT button.
4. Press the STOP button.

If the operator tries to accept a piece without specifying the rule table number, the CHECK program ignores the STOP button until the rule table is specified.

If the wrong rule table number has been entered, the message

*** SAMPLE SIZE, YY NOT AVAILABLE ENTER CORRECT SAMPLE SIZE OR "EXIT"

is printed on the screen **18.** At this point, the sample size in the digitized data cannot be found in the rule table. The correct sample size for the rule table given must be entered to proceed with the grading of the piece.

If the rule table was entered correctly, the piece will be graded and generated on the display **18** as shown in FIG. **20.** At this time, the information generated in the upper left corner of the display **18** should be noted. The information takes the following form:

PIECE XXXXXXXXXX RULE # ZZ
SIZES IN GRADE
Y Y Y Y Y Y

The first line gives the piece identification number (represented by 10 X's) and RULE #, followed by the rule number (represented by ZZ) of any grade point. The next line indicates that the numbers (represented by Y) below are the sizes in this grade.

In FIG. **20,** a small asterisk **360** (called a cursor) is displayed. When the stylus **24** touches the data tablet **22,** the cursor **360** is displayed at the corresponding point on the display screen **18.** The cursor **360** will follow the stylus **24** when the stylus is within ¼ inch of the data tablet surface. The operator uses this as a guiding tool when checking the pattern piece.

At this time, the grade rules at the grade points may be checked and changed if necessary. If the operator activates the stylus **24** (by applying pressure) at a grade point; a large asterisk will be displayed at the grade point with the grade rule and the type of point at the right of the asterisk. By applying a constant pressure, the system will check each point that has a grade applied to it except a lettering point. When the stylus **24** is lifted up, the system will stop at the last point checked. The points checked are:

1. Grade Point — G.P.
2. Punch Holes — P.H.
3. Stripe Line — S.L.
4. Alternate Start Point — A.S.
5. Internal Point — I.P.
6. Transfer Point — T.P.

If an incorrect grade rule is displayed, the RULE # can be changed to another grade rule by:

1. Make sure the grade rule that is to be changed is displayed, i.e., make sure the proper grade point is selected.
2. Enter the new rule number the same way the rule table number was entered.
3. Press the function box button marked CHANGE RULE and the rule number displayed will change to the desired rule number. If the rule number displayed at this time is wrong, repeat Steps 2 and 3.

At this time other grade rules may be checked and/or changed, or the STOP button **94** may be depressed to regrade the piece using the new rules. More than one rule number may be changed before the piece is regraded.

The operator now has five options as to what can be done with the piece. He may:

1. Return the piece to the digitizer files without utilizing the piece. This is done by depressing the STOP button **94** on function box **20.** This will return the piece to the digitizer files in its original form without any corrected rules.
2. Reject the piece to eliminate the piece from the digitizer files. This can be done by
   a. depressing the button marked REJECT and
   b. depressing the STOP button **94.**
3. Accept the piece. When a piece is accepted, it is removed from the digitizer file and put into a temporary file. When any card is read through the card reader **14** or program MONST is restarted, the piece is transferred from the temporary file to the permanent piece file. A piece is accepted by (a) depressing

the button marked ACCEPT and (b) depressing the STOP button **94.**

4. Request a nested plot of the piece on plotter **26.** All sizes of the piece will then be plotted full size. A plot is generated by (a) depressing the button marked PLOT and (b) depressing the STOP button. The piece on the display will be returned to the digitizer file in its original form. Data concerning the piece, with any rule changes, will be prepared and put in the plot queue for plotting. The piece can be rechecked later.
5. List the area and perimeter of each size of the piece. This can be done by (a) depressing the button marked AREA and (b) depressing the STOP button **94.** The area and perimeter for each size is printed on the system console, and the piece is returned to the digitizer files.

In the last four options the buttons **90** on function box **20** marked ACCEPT, PLOT, REJECT and AREA were used to designate a specific command that the operator wanted performed. The STOP button was used to give control back to the CHECK program so these commands could be performed. After the operator checks a piece, control is returned to the MONITOR, and the MARK OR CHECK? question is displayed once more on screen **18.**

Controller Program Configuration

The program configuration of the central controller **12** is designed to provide the user with complete capability to input a sample pattern piece, grade the pattern piece to various sizes, group the pattern with other pieces to define a style, display the pieces in the style as various pieces on the display console, manipulate the pieces to produce a marker, and plot the marker in full-scale to produce a reproducible marker plot.

A flow chart of the program configuration for the central controller **12** is shown in FIG. **21.** A control program **370** named the MONITOR controls the operation of the system. The MONITOR calls various main programs, and each main program calls various subprograms as required to perform operations requested by the user.

Referring to FIG. **21,** at **372,** MONITOR makes a decision as to whether or not cards are presently being read by the card reader **14.** If the decision is positive, the MONITOR controls cards at **374,** a work queue is built at **376** and the program reiterates back to the decision point **372.**

If the decision at **372** is negative, the question MARK OR CHECK? is displayed at **378** on the display screen **18.** If the word MARK is selected by the user at **380,** the MARK program is executed at **382.** The MARK program displays reduced size reproductions of the pattern pieces on the display screen **18** and allows the operator to manipulate the pattern pieces on the screen to produce the marker. The marking program performs many functions for the operator including detecting overlaps in pieces, adjusting pieces to correct overlap, bumping pieces together, assuring pieces stay on the marker, computing efficiency of the marker and manipulating pieces in general.

If the CHECK program is selected at **384,** the CHECK program is executed at **386.** The CHECK program has been previously described and displays nested or stacked grades of pieces on the display screen **18** and enables the user to check the grade of the pieces. The operator is able to correct incorrect grade rules entered by the digitizer operator at this point.

If neither the MARK or CHECK is selected, a decision is made at **388** as to whether or not jobs are in the work queue. If not, the system reiterates beginning at decision **372.** If jobs are in the work queue, a decision is made at **390** as to whether or not to plot the next job. If so, the PLOT program is executed at **392.** The PLOT program produces nested plots and produces instructions to drive an optional pattern cutter head in order to cut pieces from cardboard, if that option is desired with the system.

If the decision at **390** is negative, a decision is made at **394** as to whether or not a DRAW job is next in the work queue. If the decision is positive, the DRAW program is executed at **396.** The DRAW program produces marker plots on the plotter **26.** A decision is made at **398** as to whether or not miscellaneous jobs are next in the job queue. If so, any or a number of miscellaneous programs such as BRULE, MAINT, TABLE or the like may be executed. The BRULE program enables the building and lifting or rule tables. The MAINT program is a maintenance program which allows the user to list what pieces and markers are available and allows the operator to purge pieces and markers that are no longer required. The remainder of the miscellaneous programs will be subsequently discussed in greater detail.

Monitor Control Cards

Prior to laying out markers on display screen **18,** the operator must feed in MONITOR control cards into the card reader **14.** These cards define the various pattern pieces stored in the central controller **12** which define particular apparel styles.

The overall operation of the present system is controlled by the MONITOR program. The MONITOR reads data cards through the card reader **14** which specify the tasks to be performed by the system. The tasks which the operator can request the system to perform as as follows:

1. markers can be scheduled to be made ($MARK:);
2. markers can be plotted ($DRAW:);
3. pieces can be plotted ($PLOT:);
4. styles can be defined ($STYLE:);
5. grade rule tables can be created and edited ($RULE:); and
6. markers, pieces and styles can be reported and purged ($UTILITY:).

In addition to the basic functions outlined above, additional functions are available to control the operation of the system. These functions enable the operator to perform the following tasks:

1. identify the marker maker by inputting an identification number ($OPERATOR:);
2. modify the sequence of events to be performed by the MONITOR ($PRIORITY:);
3. clear all jobs scheduled to be performed ($CLEAR:);
4. suppress the redundant grading of pieces ($NO-GRADE:);
5. shut down the system in an orderly manner ($HALT:);

6. signify the end of input data ($END:);
7. define messages to be plotted on pieces and/or markers ($MESSAGE:);
8. define shared or "half" piece size groups ($TABLE:); and
9. transfer pieces to a laser cutting system (optional) ($XFER:).

The data cards read by the MONITOR are referred to as MONITOR control cards. Each data card has a "$" in Column 1 followed by the "key word" for the operation to be performed. The key words for the various operations are shown in parenthesis above, e.g., MARK, DRAW, etc. Only the first two characters of each key word are required. A colon immediately follows the key word. Some MONITOR control card formats require additional data after the colon. The formats for each card type will be subsequently shown.

The MONITOR is initiated by typing :PR,MONST on the terminal **16.** After the MONITOR is initialized, the operator will not be able to input MONITOR control commands via the console. This forces the operator to input the MONITOR control commands via the card reader **14,** unless the MONITOR makes a request for information. The MONITOR control cards are read by placing the cards in the card reader **14** face down, with the top edge toward the user and the leading edge (Col. **1**) to the left in the card reader hopper. The card reader should then be activated by depressing the START and RUN buttons. The system will automatically read the cards the print the cards on the system's console as they are read. The $END: card must always be the last card in the MONITOR control deck. When the $END: card is read, the card reader will automatically shut itself off. The MONITOR will then perform the operations requested by the MONITOR control cards.

An example listing of a MONITOR run with one nested plot requested is shown below:

```
    :PR,MONST
1       MARKAMATIC SYSTEM START
        COMPLETED
2       TRANSFER AND GRADE PIECES
        COMPLETED
3       PURGE AND PACK JOB QUEUE
        COMPLETED
4       READ JOB STACK
            1   $CL
            2   $PL:1444444908,RULE1,1.0;38
            3   $EN:
                COMPLETED
5       TRANSFER AND GRADE PIECES
        COMPLETED
6       PLOT 1444444908
        COMPLETED
```

The MONITOR assigns a sequence number to each operation performed, e.g., operation number **2** was to generate plot data for piece 1444444908. In addition, the MONITOR control cards are also assigned numbers when they are printed.

The MONITOR reads and processes the input data cards and builds a "job queue" which consists of a list of the functions to be performed by the system. The tasks scheduled in the job queue are executed by the MONITOR in the sequence in which the MONITOR control cards were input. Therefore, the order of the MONITOR control cards determines the priority assigned to various jobs. For example, if several markers are scheduled, the marker specified in the first $MARK: card will be the first marker presented to the marker maker at the graphics console. If desired, the priority of various jobs can be modified by use of the $PRIORITY: card.

Normally when MONITOR control cards are input, the system will be required to grade pattern pieces stored on the removable platters and transfer accepted pieces to the removable platter. The MONITOR scans all input data cards to determine the number of pieces to be graded and the sizes required for the various pieces. Pieces to be graded are transferred from the removable platter, graded and stored in a temporary file area on the fixed platter. In addition, pieces which have been accepted by the CHECK program are transferred from a temporary storage area on the fixed platter to the permanent storage area on the removable platter. All pieces to be used in a marker have to be in the permanent piece file. Since the grading process requires a considerable amount of time, it is important that the system not be restarted until as many scheduled jobs as possible have been executed. As jobs are executed, they are eliminated from the job queue. After grading is completed, the message MARK OR CHECK is displayed on the display screen **18.**

At this time, the user can select to begin the marking or checking program. The MARK OR CHECK message is displayed for a period of approximately **5** seconds on the screen **18.** If the operator does not begin a marking or checking operation, the MONITOR will examine the job queue and execute the next job in the queue. The jobs in the queue will normally be $PLOT: or $DRAW: operations to generate nested plots of pattern pieces or plots of markers respectively. After the operation is completed or if no operation is in the job queue scheduled for operation, the message MARK OR CHECK is displayed on the screen **18** once more. The MONITOR is halted by entering a $HALT: card to assure an orderly shutdown of the MONITOR.

The layout of the more important MONITOR cards will now be described in detail. In the description of the MONITOR card, an 80 column coding sheet representation will be shown in the figures. The coding sheet is a conventional technique for laying out data to be keypunched into a conventional 80 column computer card. Each column of the coding sheet therefore corresponds to a column of a conventional computer card.

FIG. **22** illustrates the coding sheet for a $STYLE: card. The statement on the coding sheet is utilized to define styles are processed and permanently stored in the file system. Each $STYLE: definition must have a unique name and defines a list of pattern pieces to be included in the particular $STYLE:. As previously noted, a large number of coordinate data defining pattern pieces are stored in the system. Only by use of the $STYLE: card are the pattern pieces defined which comprise a particular style. The $STYLE: definitions are processed by the MONITOR program before the definitions are referenced by a $MARK: statement. Upon inputting of the $STYLE: cards, the central controller **12** leaves the data input from the card reader **14,** and checks to make sure that each pattern piece defined on the cards is digitized and is available.

The modifiers which appear on the coding sheet shown in FIG. **22** are defined as follows:

$STYLE:	Specifies the type of MONITOR request.
Style Name,	The style identifier field specifies the identifier for the style definition. The identifier must be 10 or less alpha/numeric characters and must be a unique name or number.
Rule Table	The rule table name denotes which rule table previously stored in the system to use for the grading process on the pieces in the definition.
;Piece #	The piece list specifies the pieces to be included in the style definition. Each piece description has a set of optional parameters which may be included with each piece reference. The piece identifier must reference a previously stored pattern piece.
,FLIP	This modifier (optional) specifies that this piece and all pieces in the style which have a FLIP parameter will be flipped when the flip function is selected while making a marker.
,BLOCK	This modifier (optional) specifies the piece is to be blocked while using plaid material. The amount of blocking is equal to one stripe spacing in the width and one plaid spacing in the length.
,X=value	This modifier (optional) specifies a percentage X (length) increase or decrease desired for the regular graded piece. (Note: the X= value is in addition to the normal grading rules and applied only to the piece.) The value is a ± integer or ± decimal % increase.
,Y=value	Same as X=value except the increase or decrease is applied in the Y (width) direction.
,OPPOSITE	This modifier (optional) specifies that the opposite or flip (right for left, left for right, etc.) piece is desired and not the original piece.
,ROTATE	This modifier (optional) specifies that the rotated piece (180°) is desired rather than the original piece.
,NAP	This modifier (optional) specifies that this piece and all pieces in the style which have the NAP modifier will be rotated when the rotate function is selected while making the marker.
,TABLE=n	This modifier (optional) permits the user to specify the combination (or pairs) of sizes which require only one piece for a pair of sizes rather than a piece for each size in the marker. The n value specifies the table number to be used for selection of the combination of sixes. The table values are input to the system by using the $TABLE: MONITOR request.
,TILT=n	This modifier (optional) specifies the number of n degrees (XXX.X) allowed for tilting the piece during the marking operation. The number of degrees cannot exceed 30.0. A piece may be rotated past the limit but not tilted past it. The piece will rotate 180°.
,STRIPE=group	This modifier (optional) specifies a group for particular piece matching if the stripe parameter in the marker equals 99. This parameter can only be used on pieces that are to be locked on the horizontal stripe line. 1. Center line Lock: The pieces in the style that are given this group number are to be locked on the center stripe. 2. Equidistance from Centerline: The pieces in the style that are given this group number are to be locked on equidistance from the centerline, whether below or above the line. 3. Align on same stripe: The pieces in the style that are given this group number are to be locked on the same line. Group numbers 4 thru 7 can be given to piece in a style to allow separate groups to be locked on different lines. 4.–7. Same as 3.
,SHADE	This modifier (optional) specifies that the piece must conform to the shading constraint as specified on the $MARK: request.
,msg	This message parameter (optional) is an integer from 1 to 10 and specifies that message n appears on the piece as it is drawn on the marker plotter. The message number refers to one of 10 messages previously loaded by the $MESSAGE: statement.

FIG. 23 illustrates a coding sheet for the $MARK: control card. The $MARK: MONITOR control card specifies the styles and sizes to be marked. Each marker must have a unique name and the marker when completed is stored permanently in the $MARK: file.

FIG. 23 shows four examples of various markers. Example A shown on the coding sheet is a single size 14 marker with a material width of 60 inches, a yardage length goal of 3 yards 12 inches, using the pieces which have been defined by the style definition . . . STYLE1.

Example B is a three size marker using sizes 12½, 14½, 16½ with a material width of 59½ inches, a yardage length goal of 4 yards 24 inches, and the pieces defined by the style definition . . . EXAMPST.

Example C is a multiple style marker using size 14 of the style called STYLE1 and size 16½ of the style called STYLE2. The material width is 60½ inches, the yardage length goal is 3 yards, and the Dimension, Stripe, Plaid, Flip & Nap parameters are used.

Example D is a single style marker with a 14 and a reverse size 14 (piece rotated 180°). The material width is 60½ inches, and the yardage length goal is set at 90 percent efficiency.

The card modifiers for the $MARK: card are as follows:

$MARK:	Requests the generation of a marker.
Marker Name,	The marker identifier field specifies the identifier for the stored marker. The identifier must be 10 or less alpha/numeric characters and must be a unique name or number. Examples: 238721FALL; 28-26-7620; ALINE-2631; 060-32-767.
WIDTH=value,	This modifier specifies the width in inches of the marker. The width parameter may be expressed as decimal value.
GOAL=yds/inches GOAL=yds EFF =%	The modifier specifies a yardage length in yards and inches (GO=yds/inches) or decimal yards (GO=yds). An example of each would be GO=2/1 or GO=2.5. The operator can set the yardage length by specifying percent of utilization or the percent of fallout (waste). To set the yardage length in this manner, the operator would use EFF=% greater than 50% for percent of utilization or EFF=% less the 50% for percent of fallout (waste) allowed. The value can be an integer or decimal value. An examle of each would be EFF=86.5 for percent of utilization and EFF=15.5 for the percent of fallout (waste) allowed. The marker program displays a dashed line on the displayed marker which represents the yardage length the marker should be made in. If the EFF=% less than 50% for percent of fallout is used (Example: EFF=15.5), the yardage length line will be set at 84.5% percent utilization.
DIMENSION,	This modifier (optional) instructs the MARK program to output to the standard output device the area and perimeter of all pattern pieces in the marker definition. This information will be generated on all markers if the DIMENSION bit has been set by CAMSCO.
SHADE-value,	This modifier (optional) defines the number of horizontal shaded areas. The MARK program will take into consideration where pattern pieces of the same group are positioned in the marker layout area. This modifier is an integer value not greater than 4. For further information see Marking Procedures.
STRIPE=spacing/-offset,	This modifier (optional) defines the distance in inches between stripes and the start point from the top edge of the marker. If the value in stripe equals 99, there will be no stripes drawn in the marker. The piece matching group number for ST-99 is given in the style definition's stripe parameter will apply instead. If a stripe parameter is not in the style definition the marker will not have any stripes and the "Stripe Lock" button used in making markers will be useless.
PLAID=value,	This modifier (optional) defines the distance in inches between plaid lines. The parameter value is either an integer or decimal value.
X=value,	This modifier (optional) specifies a percentage of increase or decrease applied to the regular grade for all pieces in the marker. The modifier is an integer or decimal value. Example: X=–7.5 for .75% decrease.
Y=value,	This modifier (optional) specifies a percentage of increase or decrease applied to the regular grade for all pieces in the marker. The modifier is an integer or decimal value.
FLIP,	This modifier (optional) instructs the MARK program that a constraint has been applied to the marking process for the style. All pieces in the style/size group are flipped when the FLIP constraint is applied on the marker request. If the flip override is selected then all pieces in the style which have a flip parameter are flipped as a group.

-Continued

NAP;	This modifier (optional) instructs the MARK program that a constraint has been applied to the marking process for the style. All pieces in the style/size group are rotated when the NAP constraint is applied on the marker request. If the nap override is selected then all pieces in the style which have a nap parameter are rotated as a group.
Style Name,	The particular style definition to use in the marker description — the style definition contains the description of the pieces to include.
$size_1, size_2, \ldots size_n$	The size list specifies the desired sizes from the style to be included in the marker layout. The size parameters must match the available sizes of the style in the rule table. A size is initially rotated if the character "R" preceeds the size parameter (R2830, R10).

Any other styles and their sizes should be entered in the following format (see above modifiers for explanation):

;style name, $size_1$, $size_2$, . . . $size_n$

All styles and their sizes are separated by semicolons (;).

In addition to the above-described $MARK:, the system includes an Old $MARK: card which defines the parameters and pattern pieces required to create a new marker from an old marker. This old $MARK: statement is loaded into the job queue and initiated on the display **18** when requested by the marker operator. The marker request specifies an old marker that is to be displayed. This old $MARK: card is generally similar to the above-described $MARK: card.

FIG. **24** illustrates a coding sheet for the preparation of a $DRAW: MONITOR card. The $DRAW: card schedules the plotting of a previously generated marker by specifying a marker identifier, the scale, the alpha bundle designator, the gap between markers and a list of optional messages to be printed on the marker. FIG. **24** illustrates three examples of a $DRAW: card.

Example A shown in FIG. **24** is a draw card to plot the marker EXAMPMARK at full scale, with at least a four inch gap between this marker and the last thing plotted. Example B is a draw card to plot the marker MK1 at half scale with message number eleven plotted on the marker header. Example C is a draw card to plot marker MK1 at full scale.

Modifiers for the $DRAW: card are as follows:

$DRAW:	Requests the plotting of a marker.
Marker Name,	This modifier specifies the marker identifier for the marker to be plotted.
ALPHA=value	(Optional) The alpha character is used to assign bundle numbers to sizes in a marker section. The alpha character will be added to the size plotted on each piece and advanced to the next alphabetic character on each occurrence of a new size in the marker section. For example, with AL=C for a 2 size dress marker with sizes 10 and 16, size 10 pieces would be labeled 10-C and size 16 pieces would be labeled 16-D. If the alpha value is set to a numeric value, e.g., AL=25, the bundle numbers start with the value specified (3031-25, 3030-26, etc.). The bundle numbers are assigned to sizes in the order specified by the $MARK: card when the marker was originally created.
GAP=value,	(Optional) This modifier specifies the distance the operator would like to allow between the marker specified in the $DRAW: and the last plotted data.
SCALE=value	(Optional) This modifier specifies the scale at which the marker is to be plotted. Typically the scale will always be 1.0 for one to one scaling but a decimal scale can be specified, (.5, etc.).
;msg_1, msg_2, . . . msg_n	(Optional) This message parameter is an integer from 11 to 60 and specifies that the message number given will be plotted on the marker header. The message was the $MESSAGE: statement.

FIG. **25** illustrates a coding sheet for the preparation of a $PLOT: card. The $PLOT: statement makes a request for a nested plot, or a cut or a piece that has been permanently stored on a piece file. The modifiers cut or nest specify which type of "plot" to perform. The nest modifier will create a stacked nested plot on the plotter **26.** The cut modifier will create commands to generate each graded pattern piece on the cutter.

FIG. **25** illustrates four different plot configurations. Examples A and B are plot cards for piece number 0000000011 to be plotted full-scale and half-scale respectively, showing all sizes used in grade rule table CAMSCO1. Example C is to plot only sizes 10 and 12 of piece 0000000011 at full-scale. Example D is to plot a nested plot at full-scale. When the nest-cut parameter and the scale parameter are omitted, they will default to nest and full-scale respectively.

The modifiers for a $PLOT: card are as follows:

$PLOT:	Specifies the type of MONITOR job request.
Piece #,	The modifier specifies the piece identifier of the requested piece. The piece must be on the piece file.
CUT NEST	The cut or nest modifier specifies the type of output required. This is an optional parameter. If this parameter is omitted, it will be defaulted to nest.
RULE TABLE,	This modifier specifies the name of the rule table for grading the piece.
SCALE = value;	This modifier specifies the size of plot or cut desired. This is an optional parameter. If this parameter is omitted, it will be defaulted to 1.0 scale.
$size_1, size_2, \ldots size_n$	This modifier specifies the sizes desired for cutting or nesting. The sizes must correspond to the sizes of the reference rule table. If no sizes are specified in the statement, then all sizes are plotted. If the sample size is the only size specified, the grade rule number for each grade point will be printed on the pattern.

Selection of Optional Features of the System

Prior to initial marker construction with the present system, the user must select the options desired to be used with the system. The POPTS program makes it possible for the user to select the optional features needed to enhance the system. This program presents the questions for the optional features of the system to the user for selection. Most of the questions of this program require a YES or NO response to be typed in on the terminal **16.** Other questions will require a numeric value response. If the user fails to reply with the correct response, this program will inform the user with an ERROR MESSAGE, then repeat the question for the correct response.

The following contains all the optional features, in the way they appear on the terminal **16.** The following also lists the ERROR MESSAGES and a directory to each question with an explanation of the question. At the end of the program, the user can get a report of all the OPTIONS selected by answering the question: "DO YOU WANT OPTION Report?" with a YES response. To use the program, simply type in the following: :RU,-POPTS.

The first option to be chosen is whether or not alpha characters are desired. When making a marker, the pieces required for the marker appear on the display screen **18.** The system checks to see if the user has

specified that the pieces to be displayed have alphabetic characters to identify them by size groupings. Under the POPTS program, when the question: "ALPHA CHARACTERS" appears on the terminal **16**, the following happens:

If the user responds with YES on the terminal **16**, the displayed pieces will have alpha characters identifiers thereon.

If the user responds with NO, the displayed pieces will have no identifing characters.

If the user fails to answer the question with a YES or NO answer, an ERROR MESSAGE will appear, followed with a repeat of the question until the question is answered properly.

Another option comprises whether or not piece tilt is allowed. When making a marker, the efficiency of the marker is determined by PIECE MANIPULATION. After the user has picked the piece, the user can use the function box **20** to TILT the piece clockwise or counter-clockwise in steps of 0.5°, if the ALLOW TILT option is selected. When the question PIECE TILT ALLOWED appears in the terminal **16**, the following happens:

If the user responds with YES on terminal **16**, the picked piece can be tilted within the tilt constraint specified in the style definition by depressing the TILT CW or TILT CCW buttons on the function box **20**. FIG. **26** illustrates tilting of a typical pattern piece in increments of 0.5°.

If the user responds with NO the system will not allow pieces to be tilted by the marker maker.

Another option to be chosen is whether or not piece flip is allowed. When making a marker, the pieces required for the marker appear on the display screen **18**. Once the user has picked the piece, the user can FLIP it by depressing the FLIP button on the function box **20**. Each time the FLIP button is depressed, the piece flips top to bottom, if you have this optional feature selected. When the question PIECE FLIP ALLOWED appears in the POPTS program, the following occurs:

If the user responds with YES on terminal **16**, the picked piece will be flipped top to bottom, as shown in FIG. **27**, by depressing the function box FLIP button.

If the user responds with NO, the system will not allow a piece to be flipped by the marker maker.

Another option is whether or not piece rotation is allowed. When making a marker, the pieces necessary to make the marker appear on the display screen **18**. To manipulate the piece, the user must use the function box **20** to pick and gain control of the piece. Once the user has picked the piece, the user can rotate it clockwise or counter-clockwise in steps of 45° each rotation, if the ROTATE option is selected. When the question PIECE ROTATION ALLOWED appears on terminal **18**, see the following:

If the user responds with YES, the picked piece will be rotated in **45** degree increments each time the function box ROTATE CW or ROTATE CCW buttons are depressed. FIG. **28** illustrates a clockwise rotation of a piece in a 45° increment.

If the user responds with NO, the system will not subsequently allow the pieces to be rotated by the marker maker.

Other options which may be initially chosen during the POPTS Program prior to marking operations is the option PRINT MESSAGE WHEN FLIP OR PACK RUN. This option allows the system user to specify that a record of the use of the FLIP or PACK buttons on the function box **20** be printed out. This record can be utilized for quality control purposes, as it will indicate that pieces were flipped or the PACK function was used in a given marker. This option, and other subsequent options to be described, are chosen in the manner previously described by typing in YES or NO on the terminal **16**.

Another option is PLOT SPLICE MARKS. When making a marker on the system, a splice mark is utilized to indicate an area on the marker where the splicing of the end and beginning of bolts of material can stop. When the option is chosen, when saving a created marker permanently, the message PLACE SPLICE MARKS will appear on the display screen **18**. At that time, the stylus **24** may be moved by the user to indicate where the splice marks are to be placed on the display marker.

Another option is the NO OVERLAP MESSAGE IN PACK AND CHECK, which allows the user to specify that the NO OVERLAP message will appear on a display screen **18** when each piece in the marker is checked for overlaps during the PACK and CHECK functions. When the option is chosen, the program will check all pieces during PACK and CHECK and will display the NO OVERLAP message for each piece if there is no overlap, or will display the OPERATOR MUST CORRECT message on the screen **18** if a piece with overlap is found. The option ARE SIZES TO BE DISPLAYED ON LARGE PIECES enables the user to specify that alpha/numeric sizes are to be displayed on large pieces and that alphabetical characters are to be displayed on all pieces in the marker. When this option is chosen, the size for each piece remains within the outline of the piece even when the piece is being moved into the marker area.

The option REPORT LENGTH IN YARDS AND INCHES enables the user to depress the SPECIAL button on the function box **20** after completion of a marker in order to check the length of the completed marker. In operation, the SPECIAL button on the function box **20** is depressed, and the stylus **24** is operated to place the cursor asterisk **360** on the word LENGTH on the screen **18**. The user then receives a report of the complete length of the marker in either yards or inches, as requested.

The PRINT LENGTH AND EFFICIENCY WHEN MARKER SAVES option allows the operator to have the efficiency (percent of material utilized to the hundreth of a percent) of a marker and the length (in decimal yards or yards and inches as requested) of a marker printed after the marker is completed.

The ROUND LENGTH TO NEXT LARGEST ONE/HALF INCH option allows the user to request that the marker be rounded off to the next largest one/half inch.

The PLOT ALPHA CHARACTERS option checks to find out if the alpha characters (or bundle numbers) are to be plotted on each piece for identification after a marker is completed and saved.

The option DRAW STYLE NUMBERS ON PIECES determines if the style numbers for each piece on the marker are to be plotted on the pieces when the completed marker is drawn.

The PLOT HEAD ON NOTCHES option determines how the notches on the pieces are to be drawn when a completed marker is drawn. This option enables a head

to be drawn on a notch to assist the cutters to make the correct length cut.

The option TICK MARKS TO OUTSIDE determines if the tick mark representing vertical or horizontal stripe lines are to be pointed to the outside of the marker outline or to the inside.

The option DRAW BOX AND CROSS FOR PUNCH HOLES allows a box and a cross to be drawn at the location of a hole.

The option PRINT SIZE ON PIECE checks to see if the sizes are to be placed on each plotted piece when a completed marker is to be plotted.

A large number of other options are available with the present system which enables the user to specify the size of letters desired, which kind of plotter or cutter is to be utilized, enables the grading of pieces in metric dimensions rather than inches, automatically gives a programmers dump if the program develops errors, enables the storing of pieces into other platter disk storage, allows locked markers to achieve better material utilization when cutting and the like.

Marking Procedure

After the necessary pattern pieces have been digitized, the correct MONITOR card input into the system, the POPTS option decisions have been made, the pieces checked and saved, styles can be built by using the $STYLE: card. After the styles have been built, markers can be laid in via the interactive display system shown in FIGS. **1** and **29–31.** The interactive display system includes the display screen **18,** data tablet **22,** stylus (pen) **24** and the **16** key function box **20.** The display screen **18** is used to display pattern pieces to be manipulated when making markers. The data tablet **22** measures the position of the stylus (pen) **24** manipulated by the user and transmits the appropriate location information to the central controller **12.** The controller processes the location information and manipulates the piece on the display accordingly. The **16** key function box **20** is also used in conjunction with the marking procedures to specify functions the operator wishes to perform at various times in marking.

As shown in FIG. **29,** a rectangular marker area **400** will be displayed on the display screen **18** with the pieces **402** of the marker positioned outside of the marker which is outlined on the display. Both the marker area **400** and the pieces **402** are miniature scaled representations of the desired full scale marker. The pieces **402** are automatically arranged on screen **18** in order according to their size. Above the marker area **400** and its pieces **402** are two lines of information **404** concerning the marker and below the marker is one line of information. Below is an example of the information:

MARK I.D.	STYLES	WIDTH=	LENGTH=	EFFI-CIENCY
Size (s) Size Picked	3030 = A	30 = B		

The first line has the marker identification, the style or styles identification and width of the marker. The length and efficiency values of the marker are placed on the first line when they are asked for. The second line contains the sizes of the pieces in the marker and the letter corresponding to that size. A third line will show the size of each piece when a piece is picked. This line is located under the left end of the marker.

To determine where the stylus **24** is in relation to a particular piece on the screen **18,** the operator should note the small cursor (asterisk) **360** on the display screen **18.** This cursor **360** follows the stylus **24** whenever the stylus is close to the data tablet **22** (which is termed a proximity activity) and when the stylus **24** is pressed against the data tablet **22** with a small amount of pressure (this is termed as pressure activity). The proximity activity is used to move a piece **402** or move the cursor **24** to a piece. The pressure activity is used to select a piece **402** for manipulation (also called picking a piece) and to answer YES or NO questions asked by the Monitor.

In the marking mode, template **108** (FIG. **7**) is placed over buttons **90.** To pick a piece **402** for manipulation, the operator must press the pick button on function box **20** and activate the pen **24** on the data tablet **22** at the spot that corresponds with the center of the selected piece. When a piece is picked, the piece will jump on the screen **18** so that the cursor **360** is at the lower right hand corner of the piece. To release a piece from the cursor **360,** the operator can use one of two methods:

1. he may press the pick button, and the piece will be released from the cursor; or
2. the piece will be released from the cursor after some of the 16 functions have been used to manipulate the piece (bump left, bump right, bump down, bump up, overlap and pick).

Once the operator has control of a piece **402,** he may manipulate the piece in several ways with the function box buttons **90.** He may:

1. check and correct the piece for overlap with another piece;
2. bump a piece up, down, right or left;
3. tilt a piece clockwise or counter-clockwise in steps of 0.5°;
4. move the piece out of the marker;
5. flip a piece;
6. rotate a piece clockwise and counter-clockwise in steps of 45°;
7. pick (check) a piece; or
8. lock a piece on vertical and/or horizontal stripes.

With the button **90** marked Special, the operator will be able to:

1. determine the efficiency or the length of a marker;
2. respread pieces inside a marker to their original positions outside the marker or to the positions held when saved as a temporary marker;
3. turn on or off any stripe lines that are in a marker;
4. roll a marker right or left and enlarge or decrease the size of a marker and the pieces in and around it;
5. abort a marker;
6. save a finished marker permanently or temporarily;
7. place splice marks on a marker after saving it permanently;
8. plot a finished marker full-scale;
9. pack the pieces in the marker; and
10. check the pieces on the marker for overlap.

The user then moves the stylus **24** to pick one of the pieces **402** and moves the piece downwardly into the

marker area **400.** The operator then positions the piece in the desired location within the marker area **400** by using the function box **20,** releases the piece and picks up another of the pieces **402** and moves it downwardly into the marker area **400** and positions it.

FIG. **30** illustrates a partially completed marker on screen **18** and illustrates how the various pieces **402** may be rotated in order to interfit between one another. As noted, once each of the pieces **402** is initially placed in the marker area by a stylus **24,** the buttons of the function box **20** are operated by the user to provide final adjustment of the desired position of the pieces. As shown in FIG. **30,** a dotted GOAL LINE **410** is displayed on the display screen **18** to define the desired area on the marker area **400** for receiving the pieces **402.** If the operator is able to place all of the pieces **402** within the area defined by the GOAL LINE **410,** the operator is assured of having formed a successful marker.

FIG. **31** illustrates the display screen **18** with a finalized marker **400** formed thereon. When a suitable marker is finalized on the display screen **18,** the operator can then have a full size replica of the miniature marker display plotted out on the plotter **26.**

An important aspect of the present invention is the large number of functions available for operation of the function box **20.** With the use of the functions, the operator may quickly and easily arrange pattern pieces within the marker area to form the most efficient marker. Detailed description of the various functions available by operation of the buttons of the function box **20** are as follows:

1. PICK

This function box button (shown on template **108** in FIG. **7**) enables the operator to "pick up" a pattern piece so that he can move it along with the "cursor" or perform any other function that involves a picked piece. To pick a piece the operator presses this button, moves the cursor close to the center of the desired pattern piece, and then applies "pressure" to the data tablet pen. At this time the piece will jump so that the cursor is at the lower right corner of the piece. Now when the operator moves the cursor, the picked piece moves along with it. A picked piece can be released from the cursor or pen by pressing the Pick button.

2. OVERLAP

This button adjusts the position of the picked piece so that it does not overlap any surrounding pieces. If an overlap cannot be corrected, then the overlap light **344** (FIG. **12**) at the bottom of the screen **18** will light up. This light **344** will remain on until the piece is repicked.

3. BUMP DOWN

This button is used to adjust the position of a picked piece by moving it down until it hits another piece or the marker edge. If the operator has a piece that has been locked on a horizontal stripe line and he attempts to bump it down from the stripe line, he must answer the question (OVERRIDE STRIPE? YES NO) that will be shown on the display screen **18.** The operator must point to his answer with the cursor **360** and apply pressure. If he replies YES, the piece is moved, otherwise no action is taken.

4. BUMP LEFT

This button adjusts the position of the picked piece by moving it left until it hits another piece. The picked piece must be close to the other pieces (that is within about ½ inch on the display screen). If another piece is not in the way, the picked piece will be moved to the marker edge. If an operator attempts to bump a piece locked on a vertical stripe to the left, the same OVERRIDE STRIPE? YES NO question will apply.

5. TILT CCW

This button rotates the picked piece in a counterclockwise direction in steps of 0.5 degrees. The amount of tilt can be restricted by the tilt parameter in the $STYLE: statement. (Refer to the MONITOR control commands) When a piece has reached its maximum tilt, the tilt maximum light **348** will come on. A piece may be rotated in 180° increments if it has a tilt restriction on it.

6. TILT CW

This button rotates the picked piece in a clockwise direction in steps of 0.5°. The restriction in Tilt CCW applied to TILT CW.

7. RELEASE

After a picked piece has been moved into the marker, it is restricted from being moved off of the marker again. To override this restriction, hold down this button while moving the picked piece off the marker.

8. SPECIAL ON/OFF

This button turns special functions on and off in the display screen. While the special functions are on (displayed on the screen), the Input Required Light **340** will be turned on. When these special functions are turned on, they will appear on a line across the screen **18** above the marker. The operator must activate one of them or turn them off before he can perform any of the aforementioned actions. To turn the special functions off, repress the special button. To activate one of the special functions, the operator moves the cursor **360** to the function word and applied pressure. A piece cannot be picked or manipulated while the special functions are on. The special functions are:

A. EFFICIENCY — This special function computes the efficiency and length of the marker currently on the display screen **18.** Efficiency is computed according to the area of the pieces in the marker.

B. LENGTH — This special function draws a line on the marker that represents the current length of the marker. If the user has requested the option for locked markers, the question LOCKED MARKER? is asked of the operator. The length and efficiency is then computed according to his reply to the question.

C RESPREAD — This special function allows the operator to start over with the pieces spread as they were when he last saved a marker temporarily. Two questions are asked of the operator: RESPREAD? TO VERIFY HIS REQUEST TO START OVER, THE ORIGINAL? to determine how he wishes to start (from the first or from a temporary marker).

D. MATERIAL — This special function draws/removes the stripe and/or plaid description on the marker.

E. PACK — This special function moves all the pieces in the marker down and left so as to assure a close marker except for those pieces on the selvage of the marker. Those pieces will move up and left. Pack will correct overlaps if possible. If it cannot correct overlaps, the message OPERA-

TOR MUST CORRECT will appear on the screen and no action is made by the computer. The overlap light **344** will light up and the piece in overlap will be the picked piece. A message of PACK COMPLETE appears on screen **18** when this function is finished.

F. CHECK — This special function checks every piece in the completed marker for overlap. The cursor asterisk appears in the center of each piece on the screen **18** while that piece is being checked. If an overlap is found, the message OPERATOR MUST CORRECT appears on screen **18**, the overlap light **344** comes on, and the piece in question is automatically the picked piece. If no overlaps are found, the message CHECK COMPLETE appears on the screen.

G. SAVE — This special function allows the operator to save his marker temporarily or permanently.

The following YES or NO questions are asked of the operator:

LOCKED MARKER? — This is asked only if the user has requested the option for locked markers.

TEMPORARY? — A positive reply to this question saves the marker in the temporary storage area. As operator would want to save a temporary marker if he wanted to try and make a better marker than the one he had just finished. After he saves the marker temporarily, he may respread the marker to its original form. If the second marker is better, he may save this one permanently or temporarily. If the second marker is save permanently, the first marker will be lost. If the second marker is saved temporarily, the first marker will be lost, but another marker can be made to better the second marker. For the situation of saving a less efficient marker than the one already saved temporarily; see the next question 'Override Efficiency?.'

OVERRIDE EFFICIENCY? — If the operator is attempting to save a marker which is less efficient than the marker in temporary stores, this question is asked. A positive reply will save the marker despite its efficiency.

FINISHED? — A positive reply to this question saves the marker permanently and returns to the monitor.

PIECE OUT, SAVE? — If a piece is left out of the marker, this message is asked. If the operator answers no, the marker is not saved and the piece or pieces can be put in the marker.

DRAW NOW? — A positive reply to this after the operator gives a positive answer to FINISHED? will cause the plotting of the marker fullscale.

PLACE SPLICE MARKS — An explanation of splice will follow.

H. ABORT — This special function allows the operator to return to the monitor without saving the marker.

I. ← — This special function rolls the marker to the left.

J. → — This special function rolls the marker to the right.

K. ↑ — This special function increases the size of the pieces and marker.

L. ↓ — This special function decreases the size of the pieces and marker.

9. FLIP

This function box button flips (turns the piece over) the picked piece. A group of pieces can have the $STYLE: flip parameter applied to them. By attempting to flip one of these pieces, the questions FLIP OVERRIDE? YES NO must be answered with a positive answer to flip the piece. If one piece is flipped, all of the pieces in the group will flip.

10. BUMP UP

This button functions the same as BUMP DOWN, except the picked piece is adjusted by moving it up until it hits another piece or the marker edge.

11. BUMP RIGHT

This button functions the same as BUMP LEFT except the picked piece is adjusted by moving it to the right until it hits another piece or the length line.

12. ROTATE CCW

This button rotates the picked piece in a counterclockwise direction in steps of 45°. If a nap constraint was applied to the marker, a question is asked of the operator before allowing rotation. The question OVERRIDE NAP? YES NO is shown on the display screen. If the user replies YES all the pieces in the size group and nap group of the picked piece are rotated 180°. If he replies NO, no action is taken.

13. ROTATE CW

This button functions the same as ROTATE CCW, except that the piece is rotated in a clockwise direction in steps of 45°. The restriction concerning the nap restraint in ROTATE CCW applies to ROTATE CW.

14. PICK (CHECK)

This function box button functions the same as the PICK button except that the desired piece does not jump to the cursor and the piece will not move along with the cursor. The piece is, however, a picked piece for functions such as BUMP LEFT and OVERLAP. The operator depresses PICK CHECK and activiates the pen on the proper piece. Now the piece can be manipulated by using the overlap and bump buttons but the piece cannot be moved by the pen. After using PICK CHECK the PICK button will have to be pressed to pick the next piece.

15. VERTICAL STRIPE LOCK

This button adjusts the position of the picked piece on a vertical stripe line so that the stripe line of the picked piece aligns with the closest stripe line on the marker. After pushing this button, the picked piece cannot be moved up or down. To override this restriction, push the RELEASE button. A piece can be moved off a stripe with a positive answer to the question OVERRIDE STRIPE? YES NO whenever he tries to move the piece off a stripe line with the function box buttons.

16. HORIZONTAL STRIPE LOCK

This button has the same function as a vertical stripe lock except the stripes to be locked on must be horizontal.

17. STOP

This button is used in the checking procedures as previously described.

During the marking procedure, several situations may arise which are related to the particular fabric being utilized. The present system is thus capable of handling splice marks, stripe locking procedures and shade parameters.

A splice mark is used to indicate an area in a marker where the splicing of the end of a bolt of material and the beginning of a bolt of material can take place.

When a marker is saved permanently and the user requests the splice mark option, the message PLACE SPLICE MARKS will appear on the screen **18**, and the marker will be zoomed such that all pieces are displayed. The operator moves the cursor **360** to the area where he wishes a splice mark to be made and applied pressure to the data tablet pen. The width of the splice mark will be computed on the following basis:

1. a buffer area of 6 inches either side of the point chosen by the operator is used in the computation;
2. only pieces crossing into this buffer area will be checked;
3. the right-most edge of all the pieces whose right edge falls in the buffer is used as the right edge of the splice mark. The edge may then be moved to the right a fixed distance farther if the user requests; and
4. a similar process is performed for the left edge of the splice mark.

The operator may indicate up to fifteen splice marks. When he has indicated all the splice marks needed, he presses the RELEASE function box button, and the marker is saved permanently.

Any piece in a stripe or plaid marker that has the proper stripe line digitized in it can be locked on a marker stripe line. If a marker has horizontal stripe lines, then the pieces that are to be locked on those lines have to have a horizontal stripe line digitized in them. Pieces with vertical stripe lines can only be locked on vertical stripe lines in a marker. If a piece has both vertical and horizontal stripe lines in it, the piece can be locked on both horizontal and vertical stripe lines on the marker. Unless a piece has a special piece matching stripe parameter (given to it by the $STYLE: statement) and the stripe parameter in the $STYLE: statement is equal to 99, the following information will apply to that piece:

1. a piece will lock on to the nearest horizontal or vertical stripe line according to what orientation the stripe line was digitized in the piece and which stripe lock button was pressed;
2. a piece will lock on the closest horizontal and vertical stripe line, if that piece has a horizontal and a vertical stripe line, and both stripe lock buttons are pressed;
3. a piece can only be bumped off a stripe line with a positive answer to the OVERRIDE STRIPE question or the piece may be repicked;
4. a piece locked on a stripe line can be flipped, tilted, and rotated if desired. The piece stripe line will have to be locked on the marker stripe line; and
5. a piece locked on a horizontal stripe line can be moved or bumped left or right freely. This, also, applies to the up and down movement of a piece locked on a vertical stripe line. If a piece is locked on a vertical and horizontal stripe line, the piece cannot be moved in any direction unless it is repicked or the stripe lock is overridden with a positive answer to the OVERRIDE STRIPE? question.

If pieces in a marker have special piece matching parameters applied to them and the stripe parameter in the marker is equal to a 99, then the following will apply to the stripe locking procedures for these pieces:

1. there will be no stripe lines drawn into the marker;
2. in the case that the piece is in stripe group **1**, the use of the STRIPE LOCK button will cause the piece to align on the center **2** through **7**, the first piece of that group which is placed in the marker will define the stripe line or lines for that group. In order to define this stripe line the piece must be positioned and the STRIPE LOCK button depressed. After this stripe line is defined, all other pieces in that group will be locked onto that stripe line when they are placed in the marker and the STRIPE LOCK button hit. Those pieces in the style which were not assigned to a stripe group will not lock on any stripe lines even if the STRIPE LOCK button is hit. Care should be taken by the user that he place the first piece for groups **2** through **7** exactly where he wants that stripe line to remain for the marker. After fixing a stripe line for a group, the only way to change that stripe line would be to respread the marker and start from the beginning; and
3. once a stripe line or lines have been established, the pieces remaining in the group or groups will automatically jump to the proper or closest stripe line.

Shade areas are used in markers to represent different colors or designs that run the length of the material in a uniform manner. The shade parameter in the $MARK: command specifies the number of shade areas a marker is to be divided into lengthwise, as indicated in FIG. **32**, which shows a marker having three shade areas C, D and E.

Parameters of the marker shown in FIG. **32** are as follows:

A = Marker width of 59.0 inches
B = Goal of 4 yards 24 inches
C = First shade area
D = Second shade area
E = Third shade area

In the example shown in FIG. **32**, the vertical dashed line B is the goal line set in the marker card at 4 yards 24 inches. The horizontal dashed lines and the lines that divide the marker into the three shade areas specified by the marker card. The shade parameter in the style definition is used to indicate which pieces in each size are to be subject to the shade restriction.

As indicated above, only the pieces in each size that are given the shade restriction will be subject to that restriction. The first piece of each shade restricted size group placed in a shade area will determine where all the other pieces of that size group should be placed. Note the two lines of information located below the marking area. The "SIZE PICKED =" line is where the size of each piece will be shown as it is picked. The "SHADE AREA =" line will appear below the SIZE PICKED = when the marker has shade areas in it. After the first piece of each size grup group has the shade re-

striction applied to it) has been placed, the SHADE AREA = line will indicate which shade area all other pieces of that size group should be placed. In the above example shade area A is 1, shade area B is 2 and shade area C is 3. If a piece is not placed in its proper shade area the shade violation light **346** (FIG. **12**) will come on.

FIG. **33** illustrates the operation of the plotter **26** in accordance with commands from the central controller **12**. Under control of the controller **12**, the stylus **54** is moved laterally along the bar **52** and the bar **52** is moved along the length of the roll of paper **56** in order to draw a full size marker identical to the finished marker displayed on screen **18**. When finished, the paper is disengaged from the plotter **26** and is spread out over a plurality of stacks of fabric and cut in a conventional manner.

It will be understood that the marker prepared by the present system will generally comprise marker sections which are placed end to end to form a total marker. As an example, a cutting table used in apparel factories may extend from 70 to 120 feet in length, and fabric may be folded along the table with differing thicknesses. For example, the first portion of fabric on the cutting table may comprise sixty layers of fabric, while subsequent sections of the fabric may comprise one hundred twenty or more layers of fabric. The pattern section produced by the plotter **26** is spread out over the varying desired sections and the fabric is then cut to provide the desired number of pieces of the desired styles.

Storage File Organization

FIG. **34** illustrates the file storage organization for the present system. The more important portions of the file organization will now be described in detail.

Referring to FIG. **34**, the MARKD, Marker Directory is located in a removable storage platter. MARKD is the DOS-M directory file name assigned to the marker directory. The marker directory is a directory used by the system to reference the marker definitions stored in DOS-M file MARKR. The format of MARKD is shown in FIG. **35**. The last two words of each sector are zeroes except for the first sector in the directory. The last word in the first sector will be the starting sector of the last marker definition stored in MARKR. An unused entry will contain zero in the 7th word. Bit **15** of word **7** is set when that entry is to be purged. Laser flags are used when converting the system files to a system using a laser cutter.

The MARKR — Marker Definitions — file shown in FIG. **34** is the DOS-M file which contains all the marker definitions for the markers referenced by MARKD. Each marker definition will begin on a sector boundary and will be formatted as shown in FIGS. **36** and **37**. Definitions of the legends appearing in FIGS. **36** and **37** are as follows:

of pieces — number of pieces in the marker (0–1023)

of sectors — number of sectors used by this marker definition (0–63)

of styles — number of styles in marker (0–31)

of splices — number of splice marks placed in marker (0–15)

Marker man ID — identification number of the person who made the marker (0–63);

New/Old — 0 — marker has not been sent to Lasermatic— 1 — marker has been sent to Lasermatic;

of shade areas — number of divisions made in marker width for shading, less 1 (0–3);

Data Multiplier — floating point scale factor used to convert "marker units" to inches. See note *1 on previous page;

Width — width of marker in inches (format assumed as XX.XX);

Length — length of marker in marker units;

Goal Length — goal set for the marker (in marker units);

Marker Efficiency — efficiency obtained in the marker (format assumed as XX.XX);

Stripe Offset — position of first stripe relative to the top of the marker (format assumed as XX.XX);

Stripe Spacing — distance between stripes (format assumed as XX.XX);

Plaid Spacing — distance between plaid (format assumed as XX.XX);

Marker X% increase — % of increase to apply to entire marker in X direction;

Marker Y% increase — % of increase to apply to entire marker in Y direction;

Internal Style # — style used in the marker. Points to the entry in the style directory (STYLD);

of sizes — number of sizes of this style;

Size — size indicator for each size needed. This must be used in conjunction with the rule table used by the style;

X values of splice mark — placement made for a splice mark in marker units;

Internal Piece # — pointer to the piece in the piece directory (PIECD);

Master Image # — when the piece is a slave due to the half piece flag being set, this is the relative piece number for the master piece;

Stripe Group — special stripe matching by assigning a group number 1–7. 0 indicates regular stripe locking.;

Flip group — 0 — piece not flipped with group, — 1 — piece flipped with group;

Nap group — 0 — piece not rotated 180° with group, — 1 — piece rotated 180° with group;

Opp. flag — 0 — piece is not an opposite, — 1 — piece is an opposite by style definition;

Max. Tilt — rotation limit (**0-31**);

Rule Table # — rule table used by style;

Half Piece Flag — 0 — piece is a master, — 1 — piece is a slave;

Shade Constraint — 0 — piece is not effected by shading, — 1 — piece is checked for shade constraints;

Block flag — 0 — piece is not blocked, — 1 — piece is blocked;

X% increase — % of increase to apply to piece in X direction;

Y% increase — % of increase to apply to piece in Y direction;

Style/Size Group — pointer set up from order of style and sizes stored in header;

Size of Piece — size indicator of size of this piece;

IX - X-coordinate placement of piece in marker units;

IY - Y-coordinate placement of piece in marker units; and

IANG — angle of roatation of the piece. If the magnitude of the angle is more than 360 the piece is to be

flipped after the rotation. Format of angle is assumed as XXX.X.

The PIECD, Piece Directory is located on a removable disk platter and is the DOS-M directory file name assigned to the piece directory. The piece directory is used by the system to reference the piece definitions stored on this disk platter in file, PIECE. This directory contains an entry for each piece definition in the format shown in FIG. **38.**

The last two words of each sector of FIG. **38** are zeros except for the first sector in the directory. The last word in the first sector will be the starting sector of the last piece definition stored in PIECE. An unused entry will contain zero in the 7th word. Bit **15** of word **7** is set when that entry is to be purged. Laser flags are used when converting the present files to a format for a laser cutting system. The PIECE, Piece Definitions, are shown in FIG. **39** and is the DOS-M file which contains all the piece definitions for the pieces referenced in PIECD. The definition of the piece is the digitized data after curve-citting. Each piece definition will start on a sector boundary and will be formatted as shown in FIG. **39.**

Definitions of the legends in FIG. **39** are as follows:

Date — of the year;

Mirror Flag = 0 means piece *not* mirrored about x-axis of the piece, = 1 means piece is mirrored about the x-axis of the piece;

Grain Flag = 0 means piece has no grain line and may be rotated 45°, 90° and/or 180°, = 1 means piece has a grain line and may only be rotated 180°;

of Sectors — number of sectors used to store this piece description (0–15);

Rule Table — rule table number (0–128). This is the rule table number stored in RULED pointing to the table stored in RULES;

Sample Size # — number reference of the dample size as stored in the rule table used (0–255);

New or Old Flag — 0 — piece has not been sent to the Lasermatic system, — 1 — piece has been sent to the Lasermatic system;

Point Type — type of point (0–15)

0 = Grade Point
1 = Intermediate Point
2 = Punch hole Point
3 = Stripe line Point
4 = Notch Point
5 = Grain line Point
6 = Alternate start Point
7 = Internal Point
8 = Transfer function Point
9 = Lettering Point
10 = End of data code
11 = Fixed intermediate Point

Rule # — number of the rule to be applied to the grade point, punch hole point, stripe line point, alternate start point, transfer function point, or lettering point;

IX — X-coordinate of point stored in absolute mode in mils, biased about the center of the piece;

IY — Y-coordinate of point stored in absolute mode in mils, biased about the center of the piece; and

PLOTF — plotter data file — is plotter data generated by programs PLOTP and DRAW. The data is stored in file PLOTF on the fixed disk platter and is organized as shown in FIG. **40.**

The values ϕRIGX and ϕRIGY shown in FIG. **40** are the coordinates for the bottom (or top for the cutter) left corner of a bounding rectangle for nested or cut pieces. When the cutter is moving down the top left corner is used.

The values AXϕLD and AYϕLD are the absolute table coordinates for the center of the bounding rectangle for the previous piece. The value XPBND represents the X limit for the current column of pieces. When a new column of pieces is started on the cutter, the value ORIGX is set to XPBND in order to clear the previous column. The volue LDIRC represents the column direction for the cutter with 1 = up and −1 = down.

Programs PLϕTP and DRAW are responsible for maintaining the status values for the plotter and program PLϕTP is responsible for maintaining the cutter status. At the first of a production run the production monitor initializes the status data.

The values NSECP and NSECG are the "put" and "get" sector numbers for the circular plotter data file. These pointers are also core resident and are stored in locations 36_8 and 37_8 respectively.

Each data sector has a two word header which is organized as shown in FIG. **41.** Referring to FIG. **41,**

ITYPE = Data Type for this Sector

0 = Continuation Sector
1 = Nested Plot
2 = Marker
3 = Cut Pieces
4 = Single Plot

JOB = Job Flag
0 = Old Job
1 = New Job = New Column or New Marker for the Plotter or new Tableful for Cutter.

NSEC = Sector in the PLOTF file in which data for the current job begins.

Each data entry shown in FIG. **40** is a three word entry consisting of a status word and a X and Y data word. The entries are organized as shown in FIG. **42.** Referring to FIG. **42:**

POS = Tool position with 0 = up and 1 = down
TOOL = Tool number with 1 = Pen and 3 = Cutter
HOME = Home Cutter Flag (Cutter Only) with 1 = Home
XYN = Xynetics only Flag
PA = Paper Advance Flag. When set, the X Value represents the paper advance in steps with approximately 71.05 steps required to advance 1 inch.
IANG = Blade angle increment (Cutter Only)
$\Delta X \& \Delta Y$ = Incremental X and Y values in mils.

When the status word = −1, the data for the piece is completed and the ΔX and ΔY values correspond to the first and last sector for the piece respectively.

Referring again to FIG. **34,** the DXXXX, digitizer files (Dual Channel) are approximately 200 files of 5 sectors each. The DOS-M directory names will be DXXXX where XXXX is a number from 0 to 199. The files are written by the digitizer driver and must be stored contiguously on the disk. The files will be set up in the format shown in FIG. **43.**

The DXXXX, Digitizer Files — Single Channel — comprises approximately 200 files of 5 sectors each. The DOS-M directory names will be DXXXX where XXXX is a number from 0 to 199. The files are written

by the digitizer driver and must be stored continuously on the disk. The files will be set up in the format shown in FIG. **44.**

FIG. **45** illustrates the format of the status words for the function box **20** and FIG. **46** illustrates the format of the status words for the cursor. Definitions of the legends of FIGS. **45** and **46** are as follows:

ASCII code — 0 — digit 0
1 — digit 1
2 — digit 2
3 — digit 3
4 — digit 4
5 — digit 5
6 — digit 6
7 — digit 7
8 — digit 8
9 — digit 9
< — size indicator
= — mirror indicator
> — cancel
? — rubout
; — end of piece

Button — 1000 — Grade point cursor button
0100 — Interm. point cursor button
0010 — Notch cursor button
0001 — Switch cursor button

OF — set to 1 when a register overflow occurres

CPU — set to 1 when computer requested reading of registers was made

Scale — 00 — scale of 1
01 — scale of 4
10 — scale of 2

Rotary Switch — (0-9) value on rotary switch at time of reading
0 — grain line
1 — stripe line
2 — reference line
3 — punch hole
4 — internal point
5 — alternate start point
6 — transfer function point
7 — fixed intermediate point
8 — not presently used
9 — not presently used

The Job Queue JOBQ contains the job sequences to be performed by the production monitor. The job queue is stored on the fixed disk platter in file JOBQ. The job queue is 256 sectors long and is divided into two areas. The second is the queue entry which contains the data. The first area is two sectors long with the queue entries running from location **7** to location **255.** The entire queue area is considered as a word sequential file from location **1** to **32766**$_{10}$. Numeric values in the Queue data area fileds which are data values (except counters) are scaled to one hundredths. The Job Queue will accommodate approximately **120** job requests at any one time.

Referring again to FIG. **34,** SECTR is a **128** word sector used for communication between the production monitor and executing programs. The sector is also used for status return from programs which have completed their operation.

The production monitor calls a program with the SECTR sector set to the following condition:

word **1** — address of queue entry request

The called program (MARK, CHECK, PLOT, DRAW, STYLE, etc.) returns control to the production monitor with status parameters.

CHECK — Temporary Storage — shown in FIG. **34** is used by CHECK as the graded data for the piece being checked. At this time its format is like that of file GRADE. The file is also used by MARK V2.0 SPREAD storage for the original marker spread and for the temporarily stored marker. It is also used to save the Internal #, %, Grain Angle, Warp Allowances and Fill Allowance overhead words. The file is also used by PGRAB to pass piece numbers to WSTYL and is used by BRULE to completely build an error-free rule table before transferring it to the RULES file.

Referring to FIG. **34,** OPTNS — Options file — is used by the system to provide option words for each major program in the syste. The option words are set depending on the requirements for each system. The one sector file is labeled OPTNS and is stored on the fixed platter. The option sector is generally set at system generation time and reflects the specific requirements and options for the system.

The following descriptions specify the option word location for representative programs and the bit assignments for each option:

Word 1 — Option Word for Program MARK

Bit 0 — Display alpha character on pieces displayed (1)
1 — Allow "locked markers (1)
2 — Allow Tilt for markers (1)
3 — Allow Flip for markers (1)
4 — Allow Rotate for markers (1)
5 — Print when FLIP or PACK used (1)
6 — Place splice marks (1)
7 — Message "No Overlap" in PACK and CHECK (1)
8 — Sizes to be Displayed on LARGE PIECES (1)
9 — Report Length in Yards (Hundredths) (1) in Yards and Inches (0)
10 — Interration Placement of Small Pieces
11 — Print Length & Efficiency When Marker Saved (1)
12 — Round Length to Next Largest ½ Inch (1)
13 —
14 —
15 — Multi Scope Option Present

Word 2 — Option Word for Program DRAW

Bit 0 — Plot alpha character following size on each piece in the marker (1)
1 — Treat marker as a "locked" marker (1)
2 — Draw style number when labeling pieces (1)
3 — Xynetics Model 1100 scaling (1)
4 — Draw head on notches (1)
5 — Turn stripe and plaid "tick" marks to the outside (1)
6 — Xynetics Model 2000 scissoring and paper slew (1)
7 — Draw box around cross for punch hole (1)
8 — Print size on each piece (1)
9 — Draw marker gridded at ½ units scale (1)
10 — Xynetics Carbon System (1)
11 — Stop X Axis Movement Every 4 Feet (1)
12 — Do Not Grid Piece (1)
13 — Report Length in (Hundredths) of Yards (1) In Inches (0)
14,15 — Size of Symbols on Pieces (0 - ¼ inch)
(1 - ½ inch)
(2 - ¾ inch)
(3 - 1 inch)

Word 4 — Option Word for Grading Program

Bit 0 — Grade Increments are Meteric (1)
1 — Trace Program Flow in PGRAB
2 — Print Number of Pieces to be Graded (1)
3 —
4 —
5 —
6 —
7 —
8 —
9 —
10 —
11 —
12 —
13 —
14,15 — Number of Piece Platters in System (1-0)
(2-1)
(3-2)
(4-3)

Word 5 — Deviation angle for defining a corner
Word 6 — Number of digitizer files
Word 7 —
Word 8 — Pointer to digitizer file used by CHECK. Set to 1 originally.
Word 9 — Default sample size - Example: "3" means use third Rule Table entry when the sample size is not input by the digitizer operator

-Continued

Word 1 — Option Word for Program MARK

Word 10 — Paper width (in Hundredths of inches)
Word 11 — Model 2000 paper advance factor.
Word 15 — 2nd Option Word in DRAWM

Bit 0 — Do detailed Piece Sort (1)
1 — Do Not Draw Selvage Lines on Marker (1)
2 — Force Heading Even When GAP=0 Above Marker (1)
3 — Force Heading Inside the Marker When GAP=0 (1)
4 — Count Corners (1)
5 — Count Notches and Punch Holes (1)
6 —
7 —
8 —
9 —
10 —
11 —
12 —
13 —
14 —
15 —

Referring again to FIG. **34**, MESSG — Marker messages — file contains the marker messages printed on the marker as the marker is drawn on the plotter. Messages **1** thru **10** apply to individual pieces as specified by the style definition. Messages **11** thru **60** are printed in the heading of the marker and are selected by the $DRAW: statement. The file organization consists of **20** sectors with **3** messages per sector.

The STYLD, Style Directory shown in FIG. **34** is the DOS-M directory file name assigned to tye style directory. This directory is used the the system to reference the style definitions stored in the file, STYLE. The format of STYLD is shown in FIG. **47.** The last two words of each sector are zeroes except in the first sector. The last word in the first sector will be the starting sector of the last cycle definition stored in STYLE. An unused entry will contain zero in the 7th word. Bit **15** of word **7** is set when that entry is to be purged.

STYLE, Style Definitions, is the DOS-M file which contains all the style definitions for the styles referenced in STYLD. Each style definition will begin on a sector boundary and will be formatted as shown in FIG. **48.**

The GRADD, Graded Piece Directory, shown in FIG. **34** is the DOS-M directory file name for the graded piece directory. This directory is used to reference the graded piece data stored in file GRADE. The format of GRADD is shown in FIG. **49.** The last two words of each sector are zeroes except in the first sector. The last word in the first sector will be the starting sector of the last graded piece definition in GRADE. An unused entry will contain zero in the 7th word. Bit **15** of word **7** is set when that entry is to be purged.

The GRADE, Graded Piece DEfinition, is the DOS-M file containing all the graded piece definitions for the pieces referenced in GRADD. Each piece in the directory will be stored in the format shown in FIG. **50.**

The RULED, Rule Directory, is the DOS-M directory file name assigned to the rule directory. The rule directory is a directory used by the system to reference the rule table definitions stored in DIS-M file RULES. The format shown in FIG. **51** applied to RULED. RULED will be **5** sectors in length which assures that 90 rule tables may be referenced. The last two words of each sector are zeroes except for the first sector in the directory. The last word in the first sector will be the starting sector of the last rule table definition stored in RULES. An unused entry will contain zero in the 7th word. Bit **15** of word **7** is set when that entry is to be purged.

Formats of the remaining subroutines shown in FIG. **34** will be apparent to one skilled in the art from the previous descriptions.

Program Flow Charts

The complete program listing for accomplishment of the functions of the present system by the central controller **12**, which preferably comprises the **2100A** computer manufactured and sold by Hewlett Packard of Palo Alto, California, will be subsequently presented. However, in order to clearly document the theory of operation of the more important portions of the program, the following description will define several of the more important aspects of the program for operation of the central controller **12.**

WPROD

The purpose of the WPROD program is to monitor and control all operations of the present system. WPROD accepts control statements from the operator, performs all job scheduling and initiates the requested jobs. A job log is maintained listing each job as it is executed and completed and lists the job stack as it is read. The directory of variables of WPROD which is divided into two segments, PROD1 and PROD2 is as follows:

a. Common to PROD1 and PROD2:

IBUF	Contains first word of options file.
ISTAT	Commonly used name for status return from subroutine calls.
LOT(256)	Scratch buffer array.
LOUT	Logical unit number for job log device.
ND	Next available data location.
NER	Number of card errors.
NHALT	Flag for halt or no halt of system.
NJQ	Lock table number for job queue.
NOPAL	Option bit for AL = 1 on draw now.
NOPDI	Option bit for dimension always set.
NOPNO	Current master operator number.
NPGB	Flag for grading or no grading.
NPGIN	Flag for all cards read in or not.
NQ	Next available queue location.
NQD	Next available queue location.
NS0(128)	Buffer to contain SECTR file.
NSTRT	Flag for system start-up or not.

b. PROD1:

IDSJ	Options bit to display job messages or not.
K	Job type to be selected on calling NXTJB.
NA	Address of selected job returned from NXTJB.
NDSD	Flag for master display down or not.
NFS	Lock table number for freeze slave bit.
NJT	Same as variable K.
NRTN	Contains word one of SECTR file.
NSA	Lock table number for slave active bit.
NSHF	Flag to halt slave system or not.
NSTAT	Contains word two of SECTR file.

c. PROD2:

Key(36)	Array containing first two characters of the keywords for all control cards.
NASS	Assignment code for queue entry being built.
NFD(15)	Array used by WGF routine for all card input.
NKEY	Number of control cards in KEY.
NSTMT	Current statement number in reading cards.
NT	Job type for queue entry being built.

FIGS. **52–56** illustrate flow diagrams of the PROD1 and PROD2 programs. After start of the system at **500,** the program PROD1 is called at **502** and a decision is made at **504** as to whether or not to return from grade. If the decision is positive, the slave system is freed at **506** and status is determined at **508** to check for start-up. If start-up is indicated, the initial display MARKAMATIC SYSTEM is displayed on screen **18** at **510.** This status return of the system is check at **512** and decision is made at **514** as whether or not a halt is requested. If the decision is positive, a display that the

system is going down is made at **516.** If the halt is not requested and the job is not ready at **518,** terminal **16** prints out that the job is not ready at **520.**

The decision is made at **522** as whether to plot now and if so, the PLOTQ entry is built at **524.** If the decision is made at **526** to draw now, a DRAWQ entry is built at **528.** A display is made to the operator at **530** that the print is completed and a decision is made at **532** as whether or not a queue entry for the last job is present. If so, the queue entry is purged at **534** and a decision is made at **536** as whether or not a message end print queue is being provided. If so, the messages are listed to the user at **538.**

Referring to FIG. **53,** a decision is made at **540** as to whether or not the system has returned from PROD. A decision is made at **542** as whether or not the card reader is ready and, if no, a decision is made at **544** as whether or not there are errors in the cards which have been read. If the decisions at **542** and **544** are positive, the job queue is purged and packed at **546** and PROD **2** is called at **548.** If there are no errors in the card reader, a decision is made at **550** as whether or not the system is in startup. If not, a decision is made at **557** as whether or not cards have been read and if so, a decision is made at **558** as whether or not grading is necessary. If so, all check plot entries are purged at **560** and a decision is made at **562** as whether or not a slave halt has been requested. If so, a request halt to the slave is generated at **564.** A decision is made as whether or not grading is desired, and if so the grade message is printed at **568** and the program PGRAB is called at **570.** If grading is not desired, a request halt to the slave is generated at **572** and a decision is made at **574** as to whether or not the slave has been halted.

Referring to FIG. **54,** the decision is made at **578** as whether or not the system is in startup. If so, all PLOT pointers are initialized at **580.** If the system is not in startup, all area requests are processed at **582** and a decision is made at **584** as whether or not the display is down. If not, MARK or CHECK is displayed on the screen **18** at **586** and a decision is made at **588** as whether a response has been made. If the display is down or if no response is made, a decision is made at **590** as whether or not to space in the PLOT buffer. If the decision is positive, the PLOT or DRAW JOB is selected at **592** and the decision is made at **594** as whether or not an entry has been found. If there is not space in the plot buffer, the XFER job is selected at **596** and a decision is made at **598** as whether or not an entry has been found. If yes, the LASEM program is called at **600** and the program exits.

A decision is made at **602** as whether or not a DRAW entry is present. If not, a decision is made at **604** as whether or not a piece is available and if so the PLOTP program is called at **606** and the program exits. If no piece is available, the priority of entry is set to zero at **608** and the program reiterates to step **596.** If a marker is available at **610,** the DRAWM program is called at **612** and the program exits.

If response is present at **588,** a decision is made at **614** as whether or not the response to marked. If yes, a mark job is selected at **616** and a decision is made at **618** as whether or not an entry was found. If so, the MARK program is called at **620** and the program exits. If no entry is found, a display is made on screen **18** at **622** that the MARK queue is not ready and the program reiterates to step **536.** If the MARK decision is not made at **614,** the CHECK program is called at **624** and the program exits.

If a halt is requested at **514** (FIG. **52**) and the display going down decision is made at **516,** the decision is made at **630** in FIG. **55** as whether or not a slave halt is requested. If so, the slave is requested to halt to **632** and a decision is made at **634** as whether or not the slave has been halted. If so, the system waits for the plotter to finish at **636** and the display is idle at **638.** A decision is made at **640** as whether or not a slave halt has been requested and, if not, the slave system is freed at **642.** If so, the decision is made at **644** as whether or not a dual scope system is being utilized. If yes, a decision is made at **646** as whether or not messages are in a PRINTQ and if so the messages are listed at **648.** A decision is made at **650** as whether or not a slave halt has been requested and if so the console keyboard is unlocked at **652** and the program stops.

FIG. **56** illustrates the PROD **2** program which is called whenever the card reader is ready. On entry at **660,** the program reads the first statement at **662** and identifies the first two characters of the statement key word. If the keyword is correct, at **664** control is passed to the processor for that control card type within PROD **2.** The statement is analyzed at **666** and a decision is made at **668** as whether or not errors appear. If no, a decision is made at **670** as whether or not there is a stack in the job queue. If not, a decision is made at **672** as whether or not to execute and the system completes processing at **674.** If the keyword is incorrect, the diagnostics are listed at **676.**

A decision is made at **678** as whether or not the statement is ended and if so, a decision is made at **680** as whether or not errors are in the job stack. If not, PROD **1** is called at **682** and the program exits. If errors are found in the job stack, the decision is made at **684** as whether or not the card reader is ready. If so, the system reiterates to read statements in step **662.**

If the stack in the job queue is available, the queue entry is built at **686** and the system reiterates. If it is decided to execute at **672,** the program is called at **688** and the system exits.

Program Check

As previously noted, the purpose of the CHECK system is to check the grading and digitizer codes of pieces recently digitized. The CHECK program included three sections. The section CHKA searches for pieces in the digitizer files, converts the digitizer codes, smoothes the curves and stores it as a temporary piece on the fixed pattern platter. Program CHKB grades the piece to all sizes requested and stores the data into a scratch file CHEK. Program CHKC displays the nested piece and allows the operator to check and/or change digitized data.

The directory of variables and arrays for CHKA is as follows:

JBUF(**128**) —Buffer buffer array for storing one sector of digitizer data.

JBUF(**1536**) — Buffer area for output to temporaty piece file.

XREF(**2**) — Array for storing X coordinates of reference points.

YREF(**2**) — Array for storing Y coordinates of reference points.

IND(**500**) — Array for point type.

X(500) — Array for X coordinate of point type.
Y(500) — Array for Y coordinate of point type.
IRUL(500) — Grade Rule Number.
MSIZ — Sample Size.
IBR — Flag That is Set When a Bad Rule Number of Found.
MXSEC — Maximum # of Sectors per Digitizer File.
IDIG — Digitizer file Name.
IDNO — Piece ID Number.
OPTNS — Options File.
ITPCD — Temporary Piece Directory.
NAMR — Rule Table Name.
MFIL — Temporary Piece File.
MCK — CHKB
LIN — Logical Unit # 1 (TTY).
LOUT — Logical Unit # 6 (Printer).
NGEN — Number of Points to be added between Point Types During Smoothing Process.
NFINI — End of Piece Code.
NGP — Grade Point Code.
NIP — Intermediate Point Code.
NPH — Punch Hole Card.
NSL — Stripe Line Code.
NGL — Grain Line Code
NAP — Alternate Start Point Code.
NNP — Notch Point Code
— Transfer Point Code.
NLP — Lettering Point Code.
NREF — Reference Line Code.
NFI — Fixed Intermediate Point Code.
NSIZ — Size Key Code.
NMIRO — Mirror Key Code.

In execution, CHKA first calls in the options file to get the number of digitzer files on the system. CHKA then looks for the first file that it can fild that is not zero. If they are all zero, DMESS is called to display the message, EXIT.

When the routine finds a file with data, a call is made to DMESS to display the piece ID number and a YES or NO to check. Also EXIT is displayed. DMESS passes back to CHKA which of the three options were taken. If NO is hit, CHKA will search for the next non-zero file and repeat the previously described method. If EXIT is hit, control is returned to the monitor.

When YES is hit, CHKA again calls DMESS to display the list of Rule Table names. A code is passed back by DMESS to indicate whether the operator chose a normal return, accept the piece without display, or reject without display. Passed back with the normal return and accept is the Rule Table number selected by the operator for the piece.

If the piece is rejected without display, this routine will clear out the digitizer file and search for the next nonzero digitizer file. Otherwise, the validity of the Rule Table is checked.

Then the Piece Directory is searched for a duplicate piece ID number. If one is found, an error message is printed. If no duplicate is found, CHKA calls CONVT to change the raw digitizer data to readable codes for the rest of the CHECK program. CHKA then initializes the header information for the display and processes the codes passed back from CONVT. At this point, CHKA leaves a specified number of blank words between point types for smoothing of the data. Nonmirrored pieces are closed and all grade points are checked to see if they are in the specified Rule Table. If not, IBR flag is set to one. TABCL then is called to smooth the curves by filling in the word gaps and repack the data.

CHKA also checks the sample size in the piece data. If no such size is available in the rule table, a message will be sent to the operator to enter the correct sample size.

The last function of CHKA is to make an entry into the Temporary Piece Directory for the piece data to see if there is room. SQUEZ is called to suppress all unnecessary data and then the data is placed into the Temporary Piece File. (See FIG. 4) If the flag IBR = 1, an error message will print and the piece data will not be stored.

At this point control is passed to CHKB where the piece will be graded. The directory of variables and arrays is as follows:

IRLT — Rule Table Name.
ISIZ(42) — Array of Size Names.
NSIZ — # of Sizes.
NAMR — Rule Table File.
IBR 13 Flag Used to Indicate Bad Grade Rule.
IDNO — Digitizer File #.
IDIG — Digitizer File Name.

In execution, CHKB pulls the number of sizes and names to be graded and displayed from the specified Rule Table. The number of sizes graded is limited to 42. The array ISIZ is used to store the size names for the piece. A call to GRADP is then made to grade all sizes in ISIZ. A code of ϕ is passed from GRADP back when the grading was successfully completed. A code of 99 is passed back to CHKB when no end-of-piece code is found. The piece is then purged from the Temporary Piece File. A 98 is passed back to CHKB when a back grade rule is found and IBR is set to 1.

Control is then passed to CHKC to display the graded piece. The directory of variables and arrays for CHKC is as follows:

ISZ(42) — Array of Size Names Graded
XOFF(42) — X coordinates.
YOFF(42) — Y coordinates.
LBUF(9) — Buffer array for plot, with piece ID number and rule table #.
PERIM(42) — Array of perimeters for each size graded.
AREA(42) — Array of areas for each size graded.
IPLTB(5000) — Buffer array for display file.
ICHK — CHEK program.
ICHKB(3) — CHKB program.
LIN — Logical Unit # 1 (TTY).
LOUT — Logical Unit # 6 (Printer).
NSZ — Number of Sizes.
IRLT — Rule Table Name.
NS — Number of Sizes per Sector.
NPTYP — Point Type.
MIF — Flag to Indicate A Mirrored Piece.
ITMP — Variable for Function Box Code.
IFL — Flag to Indicate Whether Change Rule button was hit.
MSCT — Beginning Sector Number of Piece Data.
NRLNO — Grade Rule #.
MFLAG — Flag for Whether in Mirrored Part of Piece.
LISTF — Flag to List Areas.

In execution, routine CHKC reads in scratch file CHEK to get lists of size names, offsets, areas and perimeters as well as minimum X and Y coordinates. The display file is initialized with calls to EN123, SCALE and INDSP. The display is initialized for a piece with calls to ENPEC and DSPLY.

The data is read for each size and every point is set up for drawing on the display with a call to SETDS. All sizes are drawn if there is enough room in the display file but not to exceed 42 sizes.

After the grade piece is shown on the dislay by a call to START, CHKC allows the operator to accept, reject, plot, list grade rule numbers as well as change them. CHKC monitors the function box and the stylus which the operator uses to indicate his choice of action.

A call to FSHIT is made to check to see if any of the function buttons have been hit. If not, a call is made to DTHIT to see if the stylus is in the area of the piece and if pressure is being put on it. CHKC will search through the piece data for the closest point to where pen pressure is sensed. Intermediate points, end of piece, lettering points, notch or grain lines will be ignored. After the closest point type has been found, an asterisk is moved to that location and the type point as well as the grade rule will be displayed. If the pen is left down, CHKC will go through the remainder of the data sequentially to automatically display the next point type and grade rule. In a mirrored piece, when CHKC finds an end of piece code, it will go through the piece data in reversed order and negating the Y coordinate.

If a function box button is hit for Change Rule CHKC will change the rule # of the last grade point listed and store the new information into CHEK and Temporary Piece files. When a stop is hit after the change rule button, a call is made to CHKB to regrade the piece.

If the accept button is depressed, CHKC will check the flag IBR to see if all the grade rules are legal. If IBR is set to 1, an error message will be printed and the piece will not be accepted. If IBR=0, a purge flag will be set in the temporary piece directory.

Once the reject button is hit, a purge flag is also set in the Temporary Piece Directory. When the plot button is depressed, the piece ID number and Rule Table # are written into output buffer LBUF.

After the stop button is depressed, CHKC checks to see if either the accept or reject button was hit previously. A message will be printed as to whether the piece has been accepted, rejected or remains in the digitizer files. If the piece remains in the digitizer files, CHKC returns to program CHKA. If the piece is accepted or rejected, the digitizer file is released.

If plot button has been depressed CHKC will transfer the data in CHEK into Grade and the piece is placed into the plot queue.

Mark Program

As previously noted, the MARK program allows the user to place pieces for the garment into the marker via the screen 18. The directory of variables for the MARK program are as follows:

Variables in COMMON: (First word address column is based on first word of available memory as 0)

FWA	VARIABLE	DESCRIPTION
00316	CS	This variable is the coordinate scale currently used in the display file.
00270	CSMN	This variable contains the minimum value allowed for the coordinate scale.
00265	CSMX	This variable contains the maximum value allowed for the coordinate scale.

Continued

Variables in COMMON: (First word address column is based on first word of available memory as 0)

FWA	VARIABLE	DESCRIPTION
00236	EFF	This variable indicates the efficiency of the marker as percentage of area used.
00157	FACTR	This variable is the scale factor used to convert inches to marker units.
00352	IAREA	Not used.
00432	IBD	Button number on the function box for BUMP DOWN.
00435	IBL	Button number on the function box for BUMP LEFT.
00415	IBLOK	This variable contains the blocked status of the picked piece. If IBLOK = 0, the piece is not blocked. If IBLOK ≠ 0, the piece is blocked.
00434	IBR	Button number on the function box for BUMP RIGHT.
00431	IBU	Button number on the function box for BUMP UP.
00446	ICOM	Number of variables in COMMON to be stored with the display file when a marker is saved temporarily. (EFF through ICOM).
00361	ICPEC	Flag used to indicate if picked piece is not to be tracked.
00375	ICUST	Bit 1 set if FLIP used. Bit 0 set if PACK used. Bit 2 set if a piece is outside of the the marker when marker is saved temporarily or permanently.
00277	IDFSZ	This variable is the maximum size of the display file (Array IDUM).
00147–00153	IDMK(5)	These five fariables are the ASCII identification number for the marker being made.
00154	IDOP	This variable is the marker man identification number.
00447–?	IDUM(?)	The display file and overhead data array.
00306	IDXP	This variable is the X position of the picked piece in display units.
00307	IDYP	This variable is the Y position of the picked piece in display units.
00340	IEDG	This variable is the right most edge of the marker in marker units.
00444	IFL	Button number on the function box for FLIP.
00374	IFLP	Flip by group option for the marker.
00421	ILOOP	Loop counter for tilt and overlap check.
00430	ILPB	Button number on the function box for the SPECIAL button.
00305	IMENU	This variable is the relative position in the display file for the header information.
00337	IMESG	This variable is the relative position in the display file for the message area.
00301	IMGE	This variable is the relative position in the display file for the efficiency value.
00360	IMGIN	This variable is the relative position in the display file for the inches portion of the length.
00422	IMGL	The image number of the last piece picked.
00336	IMGLP	This variable is the relative position in the display file for the special buttons.
00273	IMGP	This variable contains the image number for the piece which is currently picked.
00376	IMGPK	Not used.
00345	IMGPS	This variable is the relative position in the display file for the size of the picked piece.
00302	IMGTC	This variable is the relative position in the display file for the tracking cross.
00300	IMGY	This variable is a relative position in the display file for the yardage value.
00303	IMK01	This variable is the relative position in the display file for the marker outline.
00304	IMD02	This variable is the relative position in the display file for the stripe and plaid lines.
00010–00046	INSTY(31)	These thirty-one variables are the internal style number and the rule table number for each style in the marker. The internal style number is stored in bits 15 thru 5 of the word and the rule table number is stored in 4 thru 0 of the same word.
00372	IOFSET	Flag to "slip" pieces in size/style substitution.

Variables in COMMON (First word address column is based on first word of available memory as 0)

FWA	VARIABLE	DESCRIPTION
00351	IOPTN	Option flags for program MARK.
00433	IOV	Button number on the function box for OVERLAP.
00445	IPK	Button number on the function box for PICK.
00362	IPKC	Button number on the function box for the PICK (CHECK) button.
00441	IRCCW	Button number on the function box for ROTATE COUNTERCLOCKWISE.
00440	IRCW	Button number on the function box for ROTATE CLOCKWISE.
00443	IRE	Button number on the function box for RELEASE.
00416	IROT	The flag indicating if picked piece was rotated 180° in the initial spread.
00362-00370	ISD(7)	The positions of the special stripes on the marker in marker units. If value is set to 9999 the stripe for that group has not been set.
00402	ISHDL	The lower edge of the shade zone for the picked piece in marker units.
00401	ISHDU	The upper edge of the shade zone for the picked piece in marker units.
00162-00233	ISIZ(42)	These forth-two variables are the sizes used in the marker. Bits 14 thru 8 of the word contain a pointer into array INSTY to indicate the style of that size. Bits 7 thru 0 contain the size indicator for that size. Bit 15 is used by MRKA and MRKB only to indicate the size is to be rotated 180 degrees.
00353	ISIZE	Not Used.
00442	ISL	Button number on the function box for STRIPE LOCK.
00325	ISLX	This variable is the plaid repeating factor in marker units.
00326	ISLY	This variable is the stripe repeating factor in marker units.
00341	ISOFF	Stripe offset value in marker units.
00437	ITCCW	Button number on the function box for TILT COUNTERCLOCKWISE.
00436	ITCW	Button number on the function box for TILT CLOCKWISE.
00275	ITYPE	This variable indicates the type of function entered into the function box.
00331	MYL	This variable is the limit for the lower edge of the screen in marker units.
00132-00134	NAMGD(3)	These three variables are the ASCII name for the graded data file.
00123-	NAMMK(3)	These variables are the ASCII name for the marker data file.
00066-	NAMST(3)	These three variables contain the ASCII name for the style data file.
00247	NAP	This variable is a flag indicating the nap constraint. If NAP equals 0, then there is no nap constraint on the pieces. Using ROTATE by group (or NAP by group).
00276	NOIMG	This variable indicates the number of images (pieces) in the marker.
00007	NOSTY	This variable is the number of styles used in the marker.
00155	NPEC	This variable is the number of pieces in the marker.
00272	NRGE	This variable contains the value for the goal line in marker units.
00076	NSCMU	This variable is the starting sector number for the marker data for the old marker.
00156	NSHD	This variable is the number of shade division in the marker.
00161	NSIZ	This variable indicates the number of sizes in the marker.
00417	WNDOA	The number of words in the header for each piece.
00342	SOFF	Stripe offset value in inches.
00240	SX	This variable is the plaid repeating factor in inches.
00242	SY	This variable is the stripe repeating factor in inches.
00310	IXC	This variable is the X position for the center of the screen in marker units.
00312	IXL	This variable is the left edge of the screen in marker units.
00313	IXR	This variable is the right edge of the screen in marker units.
00311	IYC	This variable is the Y position for the center of the screen in marker units.
00315	IYL	This variable is the lower edge of the screen in marker units.

Variables in COMMON (First word address column is based on first word of available memory as 0)

FWA	VARIABLE	DESCRIPTION
00314	IYU	This variable is the lower edge of the screen in marker units.
00126-00131	JBG(4)	These four variables are the field definitions for the graded data file.
00077-00122	JBM(20)	These twenty variables are the field definitions for the marker data.
00135-00146	JBP(10)	These ten variables are the field definitions for the piece data.
00047-00065	JBS(15)	These fifteen variables contain the field definitions used by the style data file.
00424	JFG	Field definition of flip by group flag in flag words.
00322	JFLG	This variable is used in communicating between segments and is especially important in using the correct entry point in segment MRKE.
00425	JNG	Field definition of nap by group flag in flag words.
00427	JRT	Field definition of rotate by size flag in piece flag words.
00425	JSHD	Field definition of shade areas in flag words.
00423	JSTY	Field definition of style number of INSTY.
00262	KAD	This variable is a relative pointer into array IDUM for the beginning of the table of addresses.
00263	KADP	This variable is a relative pointer into array IDUM for the address table entries for the pieces.
00256	KANG	This variable is a relative pointer into array IDUM for the angle of rotation for each piece.
00354	KCSR	Address in the first word address table for the coordinate scale register for the pieces.
00264	KDF	This variable is a pointer into IDUM indicating the beginning of the display file.
00261	KDPT	This variable is a pointer into array IDUM indicating the next available word in the display file.
00251	KFLG	This variable is the relative pointer into array IDUM for a flag word for each piece.
00414	KFLG2	This variable is the relative pointer into the array IDUM for the second flag word of each piece.
00413	KFWAP	This variable is the pointer into the array IDUM to the table address entry for the picked piece.
00252	KINC	This variable is a relative pointer into array IDUM for the stretch percentage for each piece. The stretch percent in the X direction is stored in Bits 15 thru 8, the Y percentage stretch is stored in Bits 7 thru 0 in MRKA & MRKB. Stored on disk then it is area of piece as XXX.X.
00250	KINT	This variable is a relative pointer to array IDUM for the internal piece numbers for the pieces in MRKA & MRKB stored on disk then it is beginning in GRADE for piece.
00373	KPLH	Button number on the function box for the Plaid Lock button.
00357	KPLL	Relative pointer to array IDUM for the plaid line information for each piece.
00411	KPLLV	The plaid line offset for the picked piece.
00356	KPSIZ	Address in the display file for the "picked size" lable.
00274	KREL	This variable is a flag indicating whether the piece is on the marker (equal 1) or is off the marker (equal 0).
00253	KSIZ	This variable contains a pointer into array IDUM for the size indicator for each piece. This size indicator word contains a pointer into array ISIZ in Bits 15 thru 8 of the word. In Bits 7 thru 0 is stored the size indicator.
00344	KSLH	Horizontal stripe line flag for locking.
00347	KSLV	Vertical plaid line flag for locking.
00244	KSTI	This variable is the relative pointer into array IDUM for the stripe line information for each piece.
00412	KSTLV	The stripe line offset for the picked piece.
00350	KTC	Entry number in the first word address

Continued

Variables in COMMON (First word address column is based on first word of available memory as 0)

FWA	VARIABLE	DESCRIPTION
		table for the tracking cursor.
00254	KXMN	This variable is a relative pointer into array IDUM for the left edge of the binding rectangle for each piece.
00405	KXMNV	The left edge of the bounding rectangle for the picked piece.
00257	KXMX	This variable is a relative pointer into array IDUM for the right edge of the binding rectangle for each piece.
00406	KXMXV	The right edge of the bounding rectangle for the picked piece.
00355	KYES	Address in the display file for the YES and NO messages.
00255	KYMN	This variable is a relative pointer into array IDUM for the lower edge of the binding rectangle for each piece.
00407	KYMNV	The lower edge of the bounding rectangle for the picked piece.
00260	KYMX	This variable is a relative pointer into array IDUM for the upper edge of the binding rectangle for each piece.
00410	KYMXV	The upper edge of the bounding rectangle for the picked piece.
00001-00003	LABF(3)	These three variables are no longer used.
00004-00006	LABR(3)	These three variables are used as the three parameters sent to MRKG corresponding to BUMP parameters.
00420	LMTPT	Number of points allowed per piece during the BUMP and OVERLAP computation.
00346	MAXDF	Maximum number of words available in the display file.
00320	MDELX	This variable is the X dimension in marker units for the binding box of the picked piece.
00321	MDELY	This variable is the Y dimension in marker units for the binding box of the picked piece.
00071-00075	MKUP(5)	These five variables contain the ASCII identification number for the marker requested as the old marker or the letters FR which indicates that the user wishes to start with the pieces spread above the marker.
00267	MKWD	This variable contains the width of the marker in marker units.
00323	MLW	This variable is the lower edge of the marker in display units.
00000	MOUT	This variable is the logical unit for the device used for output.
00324	MUP	This variable is the upper edge of the marker in display units.
	T3	Not Used.
00377	TLML	Actual Length in yards.
00327	TOTAL	This variable is the total area of all the pieces in the marker.
00234	WIDTH	This variable is the width of the marker in inches.
00332	XPC	This variable is the stretch percentage to be applied to the entire marker in the X direction.
00245	YARDS	This variable contains the length of the marker in yards.
00334	YPC	This variable is the stretch percentage to be applied to the entire marker in the Y direction.

The processing method of the MARK program includes the following segments:

MRKA This segment initializes field definitions and variables in COMMON to be used throughout program. The job queue is read and this data is placed in COMMON also. The arrays INSTY and ISIZ are set up and the option file word for MARK is read.

MRKB Substitution of styles and sizes is completed. This segment sets up the overhead information for the pieces in the marker. The position of the pieces for the initial spread or the old marker are determined and window dimensions and scaling factors are set accordingly.

MRKC This segment builds the display file, initializes additional variables in COMMON and turns on the display.

MRKD This segment performs the function for roll and zoom. The segment is entered from MRKE when the user requests a roll function or a zoom function. Segment returns to segment MRKE when the user requests a function other than a roll function or a zoom function.

MRKE This segment is the interactive loop which monitors the inputs from the light pen and the function box. This segment performs the tasks requested by the user through these two devices. Due to core limitations, a few of these tasks require calling in an additional segment. MRKD, MRKH, MRKJ and MRKG are examples of these segments which MRKE uses to perform additional tasks.

MRKF This segment has two functions. One is to store the data for the marker into a temporary file on the disk and then return to the main interactive loop. The other function is to have the operator place splice marks and then to save the marker permanently by calling MRKK.

MRKG This segment performs the function for bumping pieces and checking overlap between pieces. This segment and subroutine BUMP are identical except segment MRKG contains some diagnostic print-outs.

MRKH This segment performs the functions, pack or check. Each piece in the marker is checked for overlap and is then bumped to the left and down to assure a more efficient marker. When the function check is used the pieces are only checked for overlap and are not moved.

MRKJ This segment displays the "special" functions and performs those functions.

MRKK This segment formats the marker data to be stored permanently in the marker files. Control is then returned to the monitor.

The Bump Program

As previously noted, this subroutine bumps pieces together or checks for overlap on the screen **18.** Arguments for the program BUMP are as follows:

IND—This parameter is used to indicate which part of the MARK Program has called this BUMP subroutine.

If IND = 1 the subroutine was called from the main loop, MRKE.

If IND = 2 the subroutine was called from the packing segment, MRKH.

IP1 — This variable contains the magnitude of the move returned from this routine.

IP2 —. This parameter returns the direction of the move.

If IP'= 0 no overlap was found by the BUMP subroutine.

If IP2 ==1 an uncorrectable overlap condition exists.

If IP2 = 1 the direction of the move is in the horizontal (X) direction.

If IP2 = 2 move is in the vertical (Y) direction.

The arguments IP1 and IP2 have meaning only when IND = 2.

The piece IMGP is the piece which is bumped or moved out of an overlap condition. When the variable ITYPE (found in COMMON) is equal to IOV (also found in COMMON) the piece is checked for overlap against all other pieces in the marker. When ITYPE is equal to IBU, IBD, IBR or IBL (all found in COMMON) the piece is bumped up, bumped down, bumped to the left or bumped to the rights, respectively. When the subroutine was called from the packing segment, that is, IND = 2, the magnitude and direction of the

movement of the piece is returned via arguments IP1 and IP2 and the piece is not moved on the screen by the subroutine. When the subroutine was called from the main loop, that is, IND = 1, the piece is moved on the screen by the subroutine. The Bezel lights **240–350** are used to inform the user when there is overlap or no overlap and when the subroutine has been completed. The limit on the size of pieces is restricted as noted under the Source & Relocatable Files. When checking for overlap the subroutine attempts to correct this overlap a maximum of ten times before returning to the calling program.

The pattern pieces to be "bumped" (i.e., forced to be tangent) are defined by a closed polystring of points. To assure that the polystring of points is closed, the first and last points must be the same. A further restriction is added that the points must be originally ordered clockwise. If the piece is a "flip", the points would be ordered counterclockwise. All pieces are defined in "marker units" with respect to a common origin.

For each piece, the minimum and maximum values for X and y are used to define a rectangle which completely encloses the piece.

The direction in which the piece is to be moved is indicated by the operator as left (−X), right (+X), up (+Y) or down (−Y). The rectangle enclosing the piece to be "bumped" is then enlarged in the direction of movement. The amount the rectangle is enlarged depends upon the scale at which the operator is working. All pieces which fall within the enlarged rectangle are then examined to determine the offset needed to "bump" the pieces.

The magnitude of the offset must be the minimum distance found between the piece being moved and any surrounding piece in the designated direction. To determine this minimum distance, the following steps are taken:

1. the points of surrounding pieces which fall in the enlarged rectangle are reduced in number such that the only part which needs to be examined are those points or those line segments which fall within that rectangle;
2. the minimum distance from the points to be examined to the piece to be moved is computed. Only the distance with a direction opposite to the direction of move are used;
3. the minimum distance from the piece to be moved to the points to be examined is computed. Only the distances with a direction the same as the direction of movement are used; and
4. the minimum of the two distances found in Steps 2 and 3 is the magnitude of the offset.

The program listing for accomplishment of operation of the present system 2100A Hewlett Packard computer is set forth below in conventional Fortran computer language:

Whereas, the present invention has been described with respect to specific embodiments thereof, it will be understood that various changes and modifications will be suggested to one skilled in the art, and it is intended to encompass such changes and modifications as fall within the scope of the appended claims.

```
PAGE #164   LOISV E      CAMSCO,INC. RICHARDSON,TEXAS S


FTN4
      SUBROUTINE [illegible]55(IX,IY,IDF,IDSPR)
C.....4/27/72
C.....SUB. MAHTELL
C.....REDUCED MARK
      DIMENSION IBLOC(5),IDSPR(1)
      DATA IBLOC/4[illegible],1,1M137B,25M24B/
      JDF=IDF-1
      IDSPR(JDF+1)=4M23B
      IDSPR(JDF+2) = 77764B
      IDSPR(JDF+3)=IX*16
      IDSPR(JDF+4)=IY*16+1
      DO 10   I=1,5
      K=JDF+4+I
   10 IDSPR(K)=IBLOC(I)
      RETURN
      END
      END$


FTN4
      SUBROUTINE DISCH(IX,IY,ISCAL,IIMG,ITEXT,IDUM2,INDX,LMT)
C.....CREATES DISPLAY INSTRUCTIONS TO DISPLAY CHARACTERS
C.....IN ITEXT.  TERMINATES ON CHARACTER, $.
C
C
C
```

```
      DIMENSION ITEXT(40),NSCAL(4),NDC4(2),LCHR(2)
      DIMENSION IDUM2(1)
      DATA LD/4000B/,NMR/22B/,NXR/10B/,NCH/10017B/,
     1     NSCAL/1000B,12.B,140B,160B/,NDC4/12000B,24B/,NTB/1/,
     2     LSHF4/20B/,LSHF6/40B/,LCHR/4004B,100B/
      LRFLG = 0
C.....DETERMINE NUMBER OF CHARACTERS
      DO 10 I=1,40
      IF(ITEXT(I).EQ.LCHR(1)) GO TO 20
      N = (ITEXT(I)/LSHF6)*LSHF8
      N = ITEXT(I) - N
      IF(N.NE.LCHR(2)) GO TO 10
      LRFLG = 1
      GO TO 20
   10 CONTINUE
C.....MORE THAN 80 CHARACTERS
      CALL ERR(5,0)
   20 NNN = INDX+6+I
      IF(NNN.GT.LPT)CALL ERR(4,0)
      IDEPT = INDX
C.....CREATE DISPLAY INSTRUCTIONS
      IDUM2(IDEPT) = LD+NMR
      IDUM2(IDEPT+1) = (IIMG*LSHF8)+NTB
      IDUM2(IDEPT+2) = LD+NXR
      IDUM2(IDEPT+3) = IX*LSHF4
      IDUM2(IDEPT+4) = (IY*LSHF4)+NTB
      IDUM2(IDEPT+5) = NCH+NSCAL(ISCAL)
      IDEPT = IDEPT+6
      M = I-1
      DO 50 I=1,M
      IDUM2(IDEPT) = ITEXT(I)
   50 IDEPT = IDEPT+1
      IF(LRFLG.EQ.0) GO TO 70
      IDUM2(IDEPT) = (ITEXT(I)-LCHR(2))+NDC4(2)
      GO TO 90
   70 IDUM2(IDEPT) = NDC4(1)
   90 INDX = IDEPT + 1
      RETURN
      END
      ENDS


FTN4
      SUBROUTINE DISVL(IX,IY,VAL,NLD,NTD,ISCAL,IMGNO,IDUM2,KNDX,LMT)
C.....ROUTINE TO CREATE DISPLAY INSTRUCTIONS TO
C.....GENERATE A FORMATTED NUMBER.  "NLD" IS THE
C.....NUMBER OF LEADING DIGITS, AND "NTD" IS THE
C.....NUMBER OF TRAILING DIGITS.  IF "NTD" IS
C.....LESS THAN ZERO, THEN THE NUMBER IS INTEGER.
      DIMENSION IDUM2(1)
      DIMENSION NSCAL(4),NDC4(2),NCHAR(15),NUM(15)
      DATA LD/4000B/,NMR/22B/,NXR/10B/,NCH/10017B/,
     1     NSCAL/1000B,12.B,140B,160B/,NDC4/12000B,24B/,
     2     NTB/1B/,LSHF4/20B/,LSHF8/40B/,NUM0/6B/,
     3     NDCPT/56B/,NEG/55B/,NBLK/4B/,NFLG/0/
      IFLG = 0
      NDSP = KNDX
    7 NNX = IABS(NTD)
      IF (NTD.LT.0) NNX = 0
      NTOD = NNX
      VALUE = VAL
      IFIX = 0
      DO 10 I=1,15
   10 NCHAR(I) = NFLG
      IF (NTOD.EQ.0) GO TO 30
      DO 20 I=1,NTOD
```

```
   20 NCHAR(I) = NUM4
   30 INDX = NTDD + 1
      M = NDCPT
      IF(NTD.LT.0) M = NBLK
      NCHAR(INDX) = M
      INDX = INDX + 1
      L = A
C.....STORE ASCII NUMBERS INTO NCHAR IN THE FORMAT
C.....CALLED FOR.
   40 IF (ABS(VALUE).GT.32768.0) L=ABS(VALUE)/10000.0
      A = FLOAT(L)*10000.0
      M = ABS(VALUE)-A
      MMM = IABS(M)
   50 MM = M/10 * 10
      NCHAR(INDX) = NUM(1+M-MM
      IF(INDX.EQ.15) GO TO 100
      INDX = INDX+1
      IF (M.LT.10) GO TO 60
      M = M/10
      GO TO 50
   60 IF(L.EQ.0) GO TO 90
      IF(MMM.GE.1000) GO TO 80
      NEND = 5 - (INDX-1-NTDD)
      DO 70 I=1,NEND
      NCHAR(INDX) = NUM0
   70 INDX = INDX+1
   80 M = L
      L = 0
      GO TO 50
   90 IF (IFTT.EQ.1) GO TO 100
      IF(VALUE.LT.0.0) NCHAR(INDX) = NEG
      IF (NTDD.EQ.0) GO TO 100
      IF (ABS(VALUE).LE.32768.0) GO TO 95
      LL = ABS(VALUE)/10000.0
      VALUE = ABS(VALUE)-(FLOAT(LL)*10000.0)
   95 NN = ABS(VALUE)
      J = NN
      VALUE = (ABS(VALUE)-J)*(10.0**NTDD)
      INDX = 1
      IFTT = 1
      GO TO 40
  100 M = NLD + NTDD + 1
      DO 110 I=1,M
      IF (NCHAR(I).EQ.NFLG) NCHAR(I) = NBLK
  110 CONTINUE
      NX = 0
C.....COUNT NUMBER OF CHARACTERS USED.
      DO 120 I=1,15
      IF (NCHAR(-I+16).EQ.NFLG) GO TO 120
      NX = NX + 1
      NUM(NX) = NCHAR(-I+16)
  120 CONTINUE
  121 NPO = NDSP+5+(NX+1)/2
      IF(NPO.GT.LMT)CALL ERR(4,0)
      IF(ISCAL.EQ.0) GO TO 124
C.....CREATE NEMBER.
      IDUM2(NDSP) = LD+NMR
      IDUM2(NDSP+1) = 1
      IDUM2(NDSP+2) = LD+NXR
      IDUM2(NDSP+3) = IX*LSHF4
      IDUM2(NDSP+4) = (IY*LSHF4)+NTR
      IDUM2(NDSP+5) = NCH+NSCAL(ISCAL)
  124 NDSP = NDSP+6
C.....CREATE INSTRUCTIONS TO DRAW CHARACTERS.
```

```
  125 DO 13M I=1,NX,2
      IF(I.EQ.NX) GO TO 14M
      IDUM2(NDSP) = (NUM(I)*LSHF8)+NUM(I+1)
      NDSP = NDSP + 1
  13M CONTINUE
      IDUM2(NDSP) = NDC4(1)
      NDSP = NDSP+1
      GO TO 15M
  14M IDUM2(NDSP) = (NUM(NX)*LSHF8)+NDC4(2)
      NDSP = NDSP+1
  15M KNOX = NDSP
      RETURN
      END
      ENDS

FTN4
      SUBROUTINE DSPLA (NMSG, NSECS, IANSR)
C
C     NMSG = NUMBER OF MESAGE TO POST
C     IANSR = RESULT OF LIGHT PEN HIT FOR QUESTIONS
C     NSECS = NUMBER OF SECONDS TO HOLD QUESTION
C     IX1,IY1 = COORDINATES FOR HEADING
C     IX2,IY2 = COORDINATES FOR MESAGE
C     DISPLA = DISPLA FILE AREA
C     TABLE = DISPLAY TABLE AREA
C     DSIZE = MAX SIZE OF DISPLA ARRAY
C     NEXTD = NEXT FREE WORD IN DISPLA ARRAY
C     NREF = NUMBER OF TIMES TO REFRESH EACH SECOND
C     EOT = END OF TABLE INDICATOR
C
      INTEGER DSIZE,EOT
      INTEGER TABLE(2),DISPLA(6M),HEADR(11)
C*****************DUMMY DIM. TO LOAD****************
      DIMENSION IDUM(425)
      DIMENSION MESAGE(11,1M)
      DIMENSION IXMIN(2),IXMAX(2),IYMIN(2),IYMAX(2)
      DATA IXMIN/-4M4,423/,IXMAX/-375,115M/
      DATA IYMIN/-9M,-9M/,IYMAX/2M,2M/
      DATA JFLAG /M/
      DATA DSIZE /5M/, LSCAL /4/, EOT /14M4M4B/
      DATA IX1,IY1 /-12M,11M/, IX2,IY2 /-12M,-5M/
      DATA MESAGE
     +     /2H  ,2H  ,2H  ,2H  ,2H I,2HDL,2HE!,2H@ ,2H  ,2H  ,2H  ,
     +      2H  ,2H  ,2H G,2H01,2HNG,2H D,2HDW,2HN!,2H@ ,2H  ,2H  ,
     +      2H  ,2H  ,2H G,2H00,2HD ,2HMO,2HRN,2HIN,2HG@,2H  ,2H  ,
     +      2HMA,2HRK,2H ?,2HDE,2HDE,2H N,2HOT,2H R,2HEA,2HDY,2H!@,
     +      2H  ,2HCH,2HEC,2HK ,2HFI,2HLE,2H E,2HMP,2HTY,2H!@,2H  ,
     +      2H  ,2H M,2HAR,2HK ,2H O,2HR ,2H C,2HHE,2HCK,2H?@,2H  ,
     +      2H  ,2H  ,2H  ,2H  ,2HMA,2HRK,2H?@,2H  ,2H  ,2H  ,2H  ,
     +      2H  ,2H  ,2H  ,2H  ,2HCH,2HEC,2HK?,2H@ ,2H  ,2H  ,2H  ,
     +      11*M,
     +      11*M/
      DATA HEADR
     +     /2H  ,2HMA,2HRK,2HAM,2HAT,2HIC,2H S,2HYS,2HTE,2HM@,2H  /
      DATA LWAIT /37MB/
      DATA MWAIT /2M12M/
      DATA DISPLA/M43M05B,166M1B,M42M14B,077761B,M4M21B,
     +            073M41B, M4M23B, 077760B, 6M0000B, 000001B/
      DATA TABLE /M6MM1B, 14M2M8/
      DATA NREF /4M/
      DATA LMSG /M/
C
C     CLEAN OUR SLATE
C
      IF (NMSG+LMSG.EQ.M) RETURN
```

```
      IF(NMSG.EQ.LMSG.AND.NMSG.NE.6) GO TO 115
      LMSG=NMSG
      CALL STOP
      IF (NMSG.EQ.0) RETURN
      IF(ISSW(10).LT.0)WRITE(1,3333)NMSG,LMSG
 3333 FORMAT(2I9)
C
C     ENTER HEADING INTO DISPLAY FILE, THEN TEXT
C
      NEXTD=11
      CALL DISCH (IX1,IY1,LSCAL,0,HEADR,DISPLA,NEXTD,DSIZE)
C
      IF (NMSG.EQ.11) GO TO 110
      CALL DISCH (IX2,IY2,LSCAL,1,MESAGE(1,NMSG),DISPLA,NEXTD,DSIZE)
      IF(NMSG.NE.6) GO TO 110
      NLWA=NEXTD
      CALL CROSS(0,0,NEXTD,DISPLA)
      NEXTD=NEXTD+10
      JB1=JFLD(1,1)
  110 DISPLA(NEXTD)=EOT
      CALL START (NREF,TABLE,DISPLA)
  115 IF (NSECS.EQ.999) RETURN
      IF (NMSG.LE.5 .OR. NMSG.EQ.11) GO TO 140
C
C     TIMING LOOP FOR LIGHT PEN
C
      DO 120 I=1,NSECS,1
      DO 120 J=1,LWAIT,1
      CALL DTHIT(ISWD,LX,LY)
      IF(ISSW(10).LT.0) WRITE(1,2222)ISWD,LX,LY
 2222 FORMAT(3I6,2I8)
      CALL CROSS(LX,LY,0,DISPLA)
      IF(JLOF(JB1,ISWD).EQ.0) GO TO 120
      DO 125   K=1,2
      IF(IXMIN(K).GT.LX.OR.IXMAX(K).LT.LX) GO TO 125
      IF(IYMIN(K).GT.LY.OR.IYMAX(K).LT.LY) GO TO 125
      GO TO 130
  125 CONTINUE
  120 CONTINUE
C                                        TIMG OUT
      IANSR=0
      RETURN
  130 IANSR=K
      LMSG=0
      CALL STOP
C.....RESTORE DISPLAY FILE
      DISPLA(NLWA)=EOT
      RETURN
C
C     TIMING LOOP FOR MESAGE WITH NO LIGHT PEN RESPONSE
C
  140 DO 150 I=1,NSECS,1
      DO 150 J=1,MWAIT,1
      NMSG=NMSG+1
      NMSG=NMSG-1
  150 CONTINUE
      RETURN
      END
  199 ENDS
```

```
ASMB
       NAM DVRUP,7
       ENT DVRUP
       EXT .ENTR,MPOFF,MPON,EXEC
*
*      THIS ROUTINE INITIALIZES THE DIGITIZER AND PLOTTER DRIVERS
*
DVRUP NOP
       JSB .ENTR
       DEF DVRUP
*
       JSB MPOFF       TURN MEMORY PROTECT OFF
       DEF *+1         RETURN ADDRESS
*
       LDA DCHAN       LOAD DIGITIZER I/O SLOT CONTENTS
       CPA JSB2        DIGITIZER DRIVER ALREADY INITIALIZED?
       JMP IGNOR       YES
*
       JSB IN,FD,I     INITIALIZE F,B,/DIGITIZER
       DEC 16          1ST FILE TRK #
       DEC 0           1ST FILE SECTOR #
       DEC 5           # SECTORS/FILE
       DEC 100         # FILES
*
IGNOR LDA CUDSC        LOAD CURRENT USER DISC SUBCHN.
       STA TEMP        SAVE FOR PASS RESTORE
       CLA             SET FOR SUBCHN 0
       JSB UD          GO TO SUBCHN 0
*
       JSB EXEC        GET PLOTF LOC AND SIZE
       DEF *+4
       DEF D18
       DEF PNAM
       DEF SC
       SZA,RSS         PLOTF THERE?
       JMP EXIT        NO, IGNORE INIT
       STA FILEA       SET LOC
       LDA SC
       STA LENAD       SET SIZE
*
       JSB EXEC        GET CCD LOC
       DEF *+4
       DEF D18
       DEF CNAM
       DEF SC
       SZA,RSS         CCD THERE?
       JMP EXIT        NO, IGNORE INIT
       STA DUM1        SET LOC
       STA DUM2        SET LOC
       STA CCD         SET LOC
*
       JSB EXEC        GET GERB LOC
       DEF *+4
       DEF D18
       DEF GNA1
       DEF SC
       SZA             GERB THERE?
       STA GERB1       YES, SET LOC
       SZA             GERB THERE?
       STA GERB2       YES, SET LOC
*
       JSB EXEC        GET PGM1 LOC
       DEF *+4
       DEF D18
       DEF P1NAM
```

```
        DEF SC
        SZA             PGM1 THERE?
        STA PGM1        YES, SET LOC
*
        JSB EXEC        GET PGM2 LOC
        DEF **4
        DEF D18
        DEF P2NAM
        DEF SC
        SZA             PGM2 THERE?
        STA PGM2        YES, SET LOC
*
        JSB INIT,I      INITIALIZE CORE TO CORE
        DEF LSTE        END OF LIST ADDRESS
        DEF PBLOK       ADDRESS OF PROGRAM BLOCK TABLE
        DEF FILEA       ADDRESS OF FILE TRK/SEC
        DEF LENAD       ADDRESS OF FILE LENGTH
*
*       PATCH LIST
*
        NOP
        ABS AD+515B
        NOP
        ABS AD+516B
        NOP
        ABS AD+517B
        CLB
        ABS AD+5208
        NOP
        ABS AD+542B
        NOP
        ABS AD+543B
LSTE    EQU *
*
EXIT    LDA TEMP        LOAD SAVED USER DISC SUBCHN
        JSB UD          RESTORE
*
        JSB MPON        TURN MEMORY PROTECT ON
        DEF **1         RETURN ADDRESS
        JMP OVRUP,I     RETURN
*
UD      NOP
        CPA CUDSC       ALREADY THERE?
        JMP UD,I        YES, RETURN
        STA SC          SET NEW SUBCH
        JSB EXEC        GO THERE
        DEF **5
        DEF **1
        DEC 23
        DEF SC
        DEF .+1
        JMP UD,I        RETURN
*
*       CORE TO CORE PROGRAM BLOCK TABLE
*
PBLOK   OCT 2####0
GERR1   NOP
        OCT 4#####
GERR2   NOP
        OCT 6#####
PGM1    NOP
        OCT 1#####
PGM2    NOP
        OCT 12####
```

```
DUM1    NOP
        OCT 14000000
DUM2    NOP
        OCT 16000000
CCD     NOP
        OCT 0           END OF TABLE
*
PNAM    ASC 3,PLOTF
CNAM    ASC 3,CCD
GNAM    ASC 3,GERB
P1NAM   ASC 3,PGM1
P2NAM   ASC 3,PGM2
*
TEMP    OCT 0
SC      OCT 0
D16     DEC 16
J5M2    J5B 2B,I
*
FILEA   OCT 0
LENAD   OCT 0
*
IN.FD   OCT 5P14        F.B./DIGITIZER INIT. ENTRY
INIT    ABS 0           CORE TO CORE INIT. ENTRY
*
DCMAN   EQU 13B
.       EQU 53B
CUDSC   EQU 161B
AD      EQU 63B2B
*
        END

FTN4
        SUBROUTINE EXPAN(M,N)
C.....THIS ROUTINE EXPANDS THE DISPLAY FILE BY MOVING ALL ENTRIES
C.....DOWN N WORDS AND ADJUSTING PNTRS AS REQUIRED.
C
C
C
C
        COMMON MOUT,LABF(3),LABR(3),NOSTY,INSTY(31),JBS(15),NAMST(3),
       1 MKUP(5),NSCMD,JBM(22),NAMMK(3),JBG(4),NAMGD(3),JBP(10),
       2 IDMK(5),IDDP,NPEC,NSHD,FACTR,NSIZ,ISIZ(42),WIDTH,EFF,SX,SY,KSTL,
       3 YARDS,NAP,KINT,KFLG,KINC,KSIZ,KXMN,KYMN,KANG,KXMX,KYMX,KOPT,
       4 KAD,KADP,KDF,CSMX,MKWD,CSMN,NRGE,IMGP,KREL,ITYPE,
       5 NOIMG,IDFSZ,IMGY,IMGF,IMGTC,IMK01,IMK02,IMEND,IDXP,IDYP,
       6 IXL,IYC,IXL,IXR,IYU,IYL,CS,MDELX,MDELY,JFLG,MLW,MUP,ISLX,ISLY,
       7 TOTAL,MYL,XPC,YPC,IMGLP,IMESG,IEDG,ISOFF,SOFF,KSLH,
       8 IMGPS,MAXDF,KSLV,KTC,IOPTN,IAREA,ISIZE,KCSR,KYES,KPSIZ,KPLL,
       9 IT1,IT2,IT3,IT4,IT5,IT6,IT7,IT8,IT9,IT10,
       A IOFSET,KPLH,IPLP,ICUST,IMGPK,
       B T1,T2,T3,T4,T5,T6,T7,T8,T9,T10,
       C JSTY,JFG,JNG,JSHD,JRT,
       D ILPH,IHU,IHD,IOV,IHR,IBL,ITCW,ITCCW,
       E IRCW,IRCCW,ISL,IRE,IPL,IPK,ICOM,
       F IDUM(1)
        CALL STOP
        L=KADP+M+1
        N1=IABS(IDUM(L))
        N2 = IDFSZ-1
        DO 10   I=N2,N1,-1
        L=I-N
   10   IDUM(I)=IDUM(L)
C.....COUNT ON STRIPES
        N1=NPEC+2
        DO 20   I=1,N1
```

```
      IF(I.LE.M) GO TO 20
      L=IDUM(KADP+I)
      IF(L.EQ.100000B) GO TO 50
      N2=N
      IF(L.LT.0) N2=-N
      IDUM(KADP+I)=L+N2
   20 CONTINUE
   50 CALL START(40,IDUM(KAD+1),IDUM)
      RETURN
      END
      ENDS

ASMB
      NAM IDST,7
      ENT IDST
      EXT .ENTR
*
*     ROUTINE TO FIND THE MINIMUM DISTANCE BETWEEN A POINT
*     (IX,IY) AND A POLY-STRING OF POINTS, (JXX,JYY).
*
*     REWRITTEN BY RD GORDON 29 SEP 72
*
09999 DEC 9999
IX    BSS 1           C-B-V
IY    BSS 1           C-B-V
JXXAD BSS 1
JYYAD BSS 1
NXYJ  BSS 1           C-B-V
NAD   BSS 1
IFPAD BSS 1
IDST  NOP
      JSB .ENTR
      DEF IX          DEFINE ARGUEMENT ADDRESSES
      LDA 09999
      STA ID          ID = 9999
      LDB NXYJ,I
      ADB .-2
      SSA,INB               GET NXYJ-1 FOR LOOP
      JMP IDST,I            IF NXYJ.LE.1  RETURN
      STB NXYJ
      LDA IX,I        CALL BY VALUE
      STA IX          IS 0.98MS FASTER
      LDA IY,I        SO SET IT UP
      STA IY          NOW
      LDB NAD,I
      CLE,SSA         IF N < 0
      CMB,CME,INB      SET E AND GET IABS(N)
      ADB IFPAD,I     ADD IFP  (IEXP IN B-REG)
      LDA .+1
      SLB             EVEN POWER?
      CMA,INA         NO, GET -1
      SEZ             WAS N> 0?
      CMA,INA         NO, NEGATE A-REG AGAIN
      STA INTS
      LDB IFPAD,I     IS IFP
      SZB,RSS         EQUAL TO 0?
      CMA,INA         YES, GET -INTS
      STA JNTS
      CLA             INITIALIZE
      STA J           ARRAY INDEX
      LDB NAD,I
      SSA
      CMB,INB         GET IABS(N)
      ADB .-3         NN=0, IF IABS(N) < 3
      SSB,RSS
      INA             NN=1, IF IABS(N) >= 3
```

```
        STA JN
        SZA,RSS         NN=2?
        JMP MLOOP           YES, NO SWAP
        LDA IX          NO, REVERSE
        LDB IY          CO-ORDINATES
        STA IY
        STB IX
        LDA JXXAD
        LDB JYYAD
        STA JYYAD
        STB JXXAD
        LDA INTS
        CMA,INA
        STA INTS        IF (NN.EQ.2) INTS=-INTS
MLOOP   LDB JYYAD
        ADB J
        LDA 1,I
        STA JYYJ
        CMA,INA
        ADA IY
        SSA
        CMA,INA
        STA ID1         ID1=IABS(IY-JYY(J))
        INB
        LDA 1,I
        CMA,INA
        STA 1
        ADA IY
        STA ID6         ID6=IY-JYY(J+1)
        ADB JYYJ
        STB ID4         ID4=JYY(J)-JYY(J+1)
        LDB JXXAD
        ADB J
        LDA 1,I
        CMA,INA
        INB
        ADA 1,I
        CMA,INA
        STA ID5         ID5=JXX(J)-JXX(J+1)
        LDA IX
        CMA,INA
        ADA 1,I
        STA ID7         ID7=JXX(J+1)-IX
S130    LDB ID6
        SSB
        CMB,INB
        ADB ID1             ID2=IABS(ID6)+ID1
        LDA ID4
        SSA
        CMA,INA         ID3=IABS(ID4)
        CPA 1
        RSS
        JMP S200        REJECT IF ID2 <> ID3
        ADA 1
        SZA,RSS
        JMP S200        REJECT IF ID2=ID3=0
        LDA ID5
        SZA,RSS
        JMP S130        SPECIAL IF ID5=0
        MPY ID6
        DIV ID4         IDST = ID6*ID5 / ID4
        SOC
        JMP S200
        STA 1           SAVE IN B-REG
        ADA ID7         INT = ID7 + IDST
        SZA,RSS
        JMP S170        SPECIAL IF INT = 0
```

```
      STA INT
      XOR JNTS        IF (ISIGN(1,INT).NE.JNTS)
      SSA                   THEN
      JMP S200                    GO TO 200
      LDA ID7
      SZA,RSS
      JMP S165        REJECT IF ID7 = 0
      XOR ID5
      SSA,RSS
      JMP S165        REJECT IF ISIGN(1,ID7).EQ.ISIGN(1,ID5)
      CMA,INA         & WAS IDST IN IT STILL
      ADA IDST        GET -IABS(IDST)
      LDA ID7
      SSA
      CMA,INA         GET IABS(ID7)
      ADA 1
      SSA
      JMP S165        REJECT IF IABS(IDST).GT.IABS(ID7)
      LDA INT
      SSA
      INA,RSS
      ADA .-1               INT=INT-ISIGN(1,INT)
      SZA,RSS         IF INT = 0
      JMP IDST,I            QUICK EXIT
      STA INT
*
S165  LDA INT
      SSA,RSS         ENTER HERE FORM S180
      CMA,INA         GET -IABS(INT)
      ADA ID
      CMA,SSA,INA,RSS           SKIP IF A.LE.0 AND NEGATE A
      JMP S200
      ADA ID          A := -(ID-IABS(INT))+ID
      STA ID          IF (IABS(INT).LE.ID) ID=IABS(INT)
*
S200  ISZ J
      LDA J
      CPA MXYJ
      RSS
      JMP MLOOP
      LDA ID
      JMP IDST,I
*
S180  LDA ID7                   SPECIAL IF ID7.EQ.0
      SZA                       DO UPDATE W/ ID7
      JMP S165+1
*
S170  LDA ID4                   SPECIAL TEST
      XOR INTS        USED WHEN INT.EQ.0  (OR ID7)
      SSA
      JMP S200        REJECT POINT AND GO ON
      CLA             INT = 0
      JMP IDST,I      QUICK EXIT
*****
*****
ID    BSS 1
ID1   BSS 1
ID4   BSS 1
ID5   BSS 1
ID6   BSS 1
ID7   BSS 1
INT   BSS 1
INTS  BSS 1
JNTS  BSS 1
JXYJ  EQU INT
J     BSS 1
```

```
*****
*****
*****
*****
```

```
FIN4
      SUBROUTINE LITE(M,N)
C.....9/30/72
C.....RON HARTELL
C.....PROGRAM HANK
C.....REMOVES IMAGES FROM DISPLAY
C       M=0 FOR OFF AND M=1 FOR ON
C       N=ADDRESS IN TABLE OF FWA
        COMMON MOUT,LABF(3),LABR(3),NOSTY,INSTY(31),JRS(15),NAMST(3),
     1 NKUP(5),NSCND,JRM(20),NAMMK(3),JRG(4),NAMGD(3),JRP(10),
     2 IDMK(5),IDOP,NPEC,NSHD,FACTR,NSIZ,ISIZ(42),WIDTH,EFF,SX,SY,KSTL,
     3 YAPOS,NAP,KINT,KFLG,KINC,KSIZ,KXMN,KYMN,KANG,KXMX,KYMX,KDPT,
     4 KAD,KADP,KDF,CSHX,NKAD,CSMN,NAGE,IMGP,KREL,ITYPE,
     5 NOIMG,IDFSZ,IMGY,IMGE,IMGTC,IMKO1,IMKO2,IMENU,IOXP,IOYP,
     6 IAC,IYC,IXL,IXR,IYU,IYL,CS,MDELX,MDELY,JFLG,MLW,MUP,ISLX,ISLY,
     7 TOTAL,MYL,XPC,YPC,IMGLP,IMFSG,IEDG,ISOFF,SOFF,KSLN,
     8 IMGPS,MAXDF,KSLV,KTC,IOPTN,IAREA,ISIZE,KCSR,KYES,KPSIZ,KPLL,
     9 IT1,IT2,IT3,IT4,IT5,IT6,IT7,ITA,IT9,IT10,
     A IOFSET,KPLN,IFLP,ICUST,IMGPK,
     B T1,T2,T3,T4,T5,T6,T7,T8,T9,T10,
     C JSTY,JFG,JNG,JSMD,JRT,
     D ILPN,IMU,IMD,IOV,IBR,IBL,ITCW,ITCCW,
     E IRCW,IRCCW,ISL,IRE,IFL,IPK,ICOM,
     F IDUM(1)
        IF(M.GE.0) GO TO 100
C           REMOVE PIECES BY PUTTING THEIR START AT 2K,2K
C           OR AT -2K,2K DEPENDING UPON THE LOCATION
C           OF THE PIECE ORIGIN
        IAD=IDUM(N)
        L=1000000B
        IF(IDUM(IAD+4).GT.0) L=777600B
        IDUM(IAD+1)=L
        IDUM(IAD+2)=777618
        RETURN
  100 IND=IABS(IDUM(N))
        IF(M.EQ.0) GO TO 110
C.....TURN IMAGE ON
        CALL STAT
        IDUM(IND) = IAND(IDUM(IND),77777B)
        IDUM(N) = IND
        RETURN
C.....TURN IMAGE OFF
  110 IDUM(N) = -IND
        IDUM(IND) = IOR(IDUM(IND),100000B)
        RETURN
        END
        ENDS
```

```
      FTN4
      SUBROUTINE LOCAT (DXN,DXP,DYNN,DYP,LSTAT)
      COMMON LOUT,ITYPE,ORIGX,ORIGY,LDIRC,XPBND,AXOLD,AYOLD,
     1    IPNF(3),IPNR(3),XMAX,YMAX,SCAL,XLAST,YLAST,LASTP,LASTA,
     2    LASTT,IPLT(3),JTYPE,JOB,JPOS,JTOOL,JHOME,JIZ,SPACE,IOPTN
     3    ,JPLAT,JH0,JH1,J82,JB3,JH4
     4      ,XYCNST
      MM=-3
      DYN=DYNN
C.....ADJUST THE Y OFFSET TO CLEAR LETTERING IF AT THE TART OF A
C.....NEW COLUMN
      IF (ORIGY.EQ.0 .AND. ITYPE.EQ.2HCH) DYN=DYN-1.0
      IF(ABS(DXN)+DXP+SPACE.LE.XMAX.AND.ABS(DYN)+DYP+SPACE.LE.
     1    YMAX) GO TO 5
      RETURN
    5 XX=ORIGX+ABS(DXN)+DXP+SPACE
      YY=ORIGY+LDIRC*(ABS(DYN)+DYP+SPACE)
      IF(ISSW(15).LT.0)WRITE(LOUT,1020)DXN,DXP,DYN,DYP,ORIGX,ORIGY
      IF(ISSW(15).LT.0)WRITE(LOUT,1020)XX,YY,YMAX
 1020 FORMAT(6F9.3)
C.....CHECK Y DIRECTION
      IF(YY.LT.0..OR.YY.GT.YMAX) GO TO 100
C.....NOW CHECK X DIRECTION
      IF(XX.GT.XMAX) GO TO 200
C.....OK IN X AND Y. UPDATE ORIGINS AND BOUNDS.
      AX=ORIGX+ABS(DXN)
      AY=ORIGY+ABS(DYN)*LDIRC
      IF(ISSW(15).LT.0)WRITE(LOUT,1010)AX,AXOLD,AY,AYOLD,MM
 1010 FORMAT(4F12.3,I6)
      DXX=AX
      DYY=AY
      IF(ITYPE.NE.2HCU.OR.LSTAT.EQ.1) GO TO 50
      DXX=AX-AXOLD
      DYY=AY-AYOLD
   50 CALL PLOT(DXX,DYY,MM)
      AXOLD=AX
      AYOLD=AY
      ORIGY=YY
      IF(XX.GT.XPBND) XPBND=XX
      LSTAT=1
      RETURN
C.....OVERFLOW IN Y. CHECK X IF CUTTER OR XYNETICS
  100 M2000=JLOF(JB1,IOPTN)
      M1100=JLOF(JB2,IOPTN)
      IF(ITYPE.NE.2HCU.AND.M2000.EQ.0.AND.M1100.EQ.0) GO TO 205
      IF(XPBND+ABS(DXN)+DXP.GT.XMAX) GO TO 200
C.....FITS IN X. REVERSE DIRECTION AND SET NEW ORIGINS.
      ORIGX=XPBND
      IF(LDIRC.EQ.1) GO TO 110
C.....MOVING DOWN
      ORIGY=SPACE
      GO TO 120
C.....MOVING UP
  110 ORIGY=YMAX-SPACE
  120 LDIRC=-LDIRC
      GO TO 5
C.....NEW PAGE REQUIRED.
  200 IF(ITYPE.EQ.2HCU.AND.M2000.EQ.0) GO TO 300
      IF(ITYPE.EQ.2HCU.AND.M2000.EQ.1) GO TO 400
      IF(M1100.NE.0) GO TO 500
C.....CHECK ON A MODEL 2000. ADVANCE PAPER AND MOVE THE PEN TO
C.....BOTTOM OF THE NEW COLUMN.
      LSTAT=3
      IX=(XPBND+SPACE)*SCAL*XYCNST
      IY=0
```

```
      CALL STORE(1,ISTAT,IX,IY)
C.....MOVE THE TOOL HOME
  300 CALL PLOT(-AXOLD,-AYOLD,-3)
  305 CALL STORE(3,0,0,0)
      CALL STORE(-1,0,0,0)
  200 ORIGX=SPACE
      ORIGY=SPACE
      LSTAT=1
      XPOS0=0.
      LOINC=1
C.....SET THE NEW JOB FLAG FOR NEXT CALL TO S/R PLOT
      JOB=+40
      IF(ITYPE.NE.2HCH) GO TO 5
      ORIGX=0.
      ORIGY=0.
      XLAST=0.
      YLAST=0.
C.....ADJUST OFFSET TO CLEAR LETTERING FOR SIZES
      OYN=OYN-1.0
      GO TO 5
C.....CUTTING ON A XYNETICS 2000, OFFSET + OR - FIVE FEET
  400 IF(IPAGE.EQ.2) GO TO 410
      IPAGE=1
      XLAST=XLAST+50.0
      GO TO 300
  410 IPAGE=2
      XLAST=XLAST-01.0
      GO TO 300
C.....'CHECK' ON A MODEL 1100
C.....TREAT LIKE GERBER FOR NOW
  500 CALL PLOT(0.0,0.0,-3)
      GO TO 305
      END
      ENDS

FTN4
      SUBROUTINE MESAG (ITY,ITEX,IFLG)
C.....ROUTINE TO DISPLAY A MESSAGE ON SCREEN.  IF ITY=0,
C.....MESSAGE LEFT UP FOR 2 SECONDS.  IF ITY=1, MESSAGE IS
C.....A QUESTION AND ROUTINE GETS A YES/NO ANSWER.  IFLG=0 IS
C.....NO.  IFLG = 1 IS YES.
C
      DIMENSION ITEX(1)
      COMMON NOUT,LABF(3),LABR(3),NOSTY,INSTY(31),JBS(15),NAMST(3),
     1 NKOP(5),NSCMD,JBM(20),NAMMK(3),JBG(4),NAMGD(3),JBP(10),
     2 IOPR(5),IOOP,NPEC,NSHD,FACTR,NSIZ,ISIZ(42),WIDTH,EFF,SX,SY,KSTL,
     3 YARDS,NAP,KINT,KFLG,KINC,KSIZ,KXMN,KYMN,KANG,KXMX,KYMX,KOPT,
     4 KAD,KADP,KOP,CSMX,MKWD,CSMN,NRGE,IMGP,KKEL,ITYPE,
     5 NOIMG,TOPSZ,LAGY,IMGE,IMGTC,IMKO1,IMKO2,IMENU,IOXP,IOYP,
     6 IXC,IYC,IXL,IXP,IYU,IYL,CS,MDELX,MDELY,JFLG,MLW,MUP,ISLX,ISLY,
     7 TOTAL,MYL,XPC,YPC,IMGLP,IMESG,IEOG,ISOFF,SOFF,KSLH,
     8 IMGPS,MAXOP,KSLV,KTC,IOPTN,IAREA,ISIZE,KCSR,KYES,KPSIZ,KPLL,
     9 IMGIN,ICPEC,IT3,IT4,IT5,IT6,IT7,IT8,IT9,IT10,
     A IOFSET,KPLH,IPLP,ICUST,IMGPK,
     B T1,T2,T3,T4,T5,T6,T7,T8,T9,T10,
     C JSTY,JFG,JNG,JSHD,JRT,
     D ILPS,IHU,IHD,IOV,IHP,IBL,ITCW,ITCCW,
     E IRCW,IRCCW,ISL,IRC,IFL,IPK,ICOM,
     F IOOR(1)
      DATA NSECS/1/,NLOOP/261000/
      DATA IEND/1000/,JEND/4000000/
C.....PLACE MESSAGE IN DISPLAY FILE
      M=KAD+IMESG
```

```
      KADJ=IABS(IDUM(M))
      IAD=KADJ+3
      DO 20 I=1,20
      ITMP = ITEX(I)
      IF(IAND(ITMP,177400B).EQ.JEND) GO TO 40
      IF(IAND(ITMP,377B).EQ.IEND) GO TO 30
      IF(I.EQ.20) GO TO 30
      IDUM(IAD+I) = ITMP
   20 CONTINUE
   30 IDUM(IAD+I)=IAND(ITMP,177400B)+408
   40 IF(ITY.EQ.1) GO TO 100
C.....DISPLAY MESSAGE ONLY
      CALL LITE(1,M)
C.....WAIT 2 SECONDS
      DO 50 I=1,NSECS
      DO 50 J=1,NLOOP
      ITY = ITY+1
   50 ITY = ITY-1
      GO TO 200
C.....QUESTION
100   IDUM(KYES)=2HYE
      IDUM(KYES+1)=2HS
      IDUM(KYES+4)=2HNO
      CALL LITE(1,M)
C.....LOOK FOR PICK ON "YES" OR "NO"
  110 CALL LIGHT(1,1)
      CALL MKHIT(4,IX,LY,IFLG)
      CALL LIGHT(1,0)
C.....BLANK MESSAGE AREA AND RETURN
200   CALL LITE (0,M)
      DO 210 I=1,20
  210 IDUM(IAD+I)=2H
      IDUM(KYES)=2H
      IDUM(KYES+1)=2H
      IDUM(KYES+4)=2H
      RETURN
      END
      ENDS


FTN4
      SUBROUTINE MKHIT(IFLG,LX,LY,IMG)
C.....9/27/72
C.....RON MARTELL
C.....PROGRAM MARK
C
C
C
      COMMON MOU1,LABF(3),LABR(3),NOSTY,INSTY(31),JBS(15),NAMST(3),
     1 MKUP(5),NSCMD,JBM(20),NAMMK(3),JBG(4),NAMGD(3),JBP(10),
     2 IDMK(5),IDDP,NPEC,NSHD,FACTR,NSIZ,ISIZ(42),WIDTH,EFF,SX,SY,KSTL,
     3 YMPOS,MAP,KINT,KFLG,KINC,KSIZ,KXMN,KYMN,KANG,KXMX,KYMX,KDPT,
     4 KAD,KADP,KDF,CSHX,MKMD,CSMN,NKGE,IMGP,KKEL,ITYPE,
     5 MOIMG,IOFSZ,IMGY,IMGE,IMGIC,IMKO1,IMKO2,IMENU,IDXP,IDYP,
     6 IXC,IYC,IXL,IXR,IYU,IYL,CS,MDELX,MDELY,JFLG,MLW,MUP,ISLX,ISLY,
     7 TOTAL,NYL,XPC,YPC,IMGLP,IMESG,IEDG,ISOFF,SOFF,KSLH,
     8 IMGPS,MAXOF,KSLV,KTC,IOPTN,IAREA,ISIZE,KCSR,KYES,KPSIZ,KPLL,
     9 IT1,IT2,IT3,IT4,IT5,IT6,IT7,IT8,IT9,IT10,
     A IOFSET,KPLH,IFLP,ICOST,IMGPK,
     B T1,T2,T3,T4,T5,T6,T7,T8,T9,T10,
     C JSTY,JFG,JNG,JSMD,JRT,
     D ILP6,I40,IBD,IOV,IRM,IBL,ITCW,ITCCW,
     E IRCW,IRCCW,ISL,IRE,IFL,IPK,ICOM,
     F IDUM(1)
C.....IFLG= 0, RETURN IMG = IMAGE NUMBER OF PIECE PICKED
```

```
C       1. SET IMG = 3776 FOR PROXIMITY AND = 0 IF NOT.
C       2. SET IMG = LIGHT PEN BUTTON PICKED.
C       4. SET IMG = 1 FOR "YES" AND = 0 FOR "NO".
C.....LX AND LY = COORDINATES OF STYLUS IN MARKER UNITS.
C.....IMG = RETURN ARGUMENT
      DIMENSION IXMIN(12),IXMAX(12)
      DATA IXMIN/-1950,-1474,-1134,-858,-250,
     1 22,352,634,924,1128,1332,1536/
      DATA IXMAX/-1610,-1274,-794,-386,-114,
     1 226,498,838,1092,1296,1500,1744/
      DATA NB/12/
      N=4
      IF(NSIZ.GE.14)N=7
      LYLMT=1500-N
      LYYLMT=1450-N
      LMT=1964
      IMG=0
      JB0=JFLD(0,1)
      JB1=JFLD(1,1)
C.....CHECK STYLUS STATUS AND POSITION.
   10 CALL DTH1T(ISWD,LX,LY)
C.....UPDATE THE CURSOR POSITION.
      LX=MAX0(LX,-LMT)
      LX=MIN0(LX,LMT)
      LY=MAX0(LY,-LMT)
      LY=MIN0(LY,LMT)
      CALL CROSS(LX,LY,KTC,IDUM)
      IF(IFLG.EQ.0) GO TO 20
      IF(IFLG.EQ.2) GO TO 200
      IF(IFLG.EQ.1) GO TO 100
C.....CHECK FOR PRESSURE.
      IF(JLOF(JB1,ISWD).EQ.0) GO TO 10
      GO TO 400
C.....LOOKING FOR A HIT ON A PIECE.
   20 CONTINUE
      IF(JLOF(JB1,ISWD).EQ.0) RETURN
      IDIST=32000
      LX=IRND(FLOAT(LX)/CS)+IXC
      LY=IRND(FLOAT(LY)/CS)+IYC
      DO 30   I=1,NOIMG
C.....IGNORE SLAVE PIECES
      IF(JLOF(JB0(2,I),IDUM(KFLG+I)).EQ.1) GO TO 30
      IF(LX.LT.IDUM(KXMX+I).OR.LX.LT.IDUM(KXMN+I)) GO TO 30
      IF(LY.GT.IDUM(KYMX+I).OR.LY.LT.IDUM(KYMN+I)) GO TO 30
C.....MATCH. MAY NOT BE CLOSEST.
C.....CALCULATE DISTANCE FROM CENTER OF PIECE TO THE STYLUS.
      DX=(FLOAT(IDUM(KXMX+I))+FLOAT(IDUM(KXMN+I)))/2-LX
      DY=(FLOAT(IDUM(KYMX+I))+FLOAT(IDUM(KYMN+I)))/2-LY
      IDIS=SQRT(DY*DX+DY*DY)
      IF(IDIS.GT.IDIST) GO TO 30
      KK=I
      IDIST=IDIS
   30 CONTINUE
C.....DID A VALID PICK OCCUR?
      IF(KK.EQ.0) RETURN
C.....WAIT FOR PRESSURE TO BE RELEASED
   35 CALL DTH1T(ISWD,LXX,LYY)
      IMG=KK
      RETURN
C.....TRACKING A PIECE MODE.
  100 IF(JLOF(JB0,ISWD).EQ.1)IMG=3776
      LX=IRND(FLOAT(LX)/CS)
      LY=IRND(FLOAT(LY)/CS)
      RETURN
```

```
C.....PICKING A LIGHT PEN BUTTON.
  200 IF(JLOF(JB1,ISWD).EQ.0) RETURN
      IF(LY.LT.IYLPB.OR.LY.GT.IYLPB+70) RETURN
      DO 210    I=1,NB
      IF(LX.GT.IXMAX(I).OR.LX.LT.IXMIN(I)) GO TO 210
      IMG=1
  210 CONTINUE
      RETURN
C.....PICKING "YES" OR "NO"
  400 IF(LY.LT.IYYES.OR.LY.GT.IYYES+70) GO TO 10
      IF(LX.GT.-520.AND.LX.LT.-380) IMG=1
      IF(LX.GT.-250.AND.LX.LT.-165) RETURN
      IF(IMG.NE.0) RETURN
      GO TO 10
      END
      ENDS

FTN4
      PROGRAM MRKA,5
C.....DEC. 7, 1972
C.....MARGARET KUNAN
C.....PROGRAM MARK
C.....VERSION 2.0
C.....SEGMENT OF MARK.  TEMPORARILY SET UP TO READ FROM
C.....DATA POINT.  WILL EVENTUALLY GET DATA FROM A QUEUE
      DIMENSION ISCT(3),IFILE(100)
      DIMENSION IBUF(128),IFCT(2),MTAP(3),MK(3),IDST(5)
      COMMON MOUT,LABF(3),LABR(3),NOSTY,INSTY(31),JBS(15),NAMST(3),
     1 NBUF(5),NSCRD,JBM(20),NAMMK(3),JBG(4),NAMGD(3),JBP(10),
     2 IDMA(5),IDDF,NREC,NSHD,FACTR,NSIZ,ISIZ(42),WIDTH,EFF,SX,SY,KSTL,
     3 YABUS,NAP,KINT,KFLG,KINC,KSIZ,KXMN,KYMN,KANG,KXMX,KYMX,KDPT,
     4 KAD,KADP,KDF,CSMX,MKWD,CSMN,NKGE,IMGP,KREL,ITYPE,
     5 NOIMG,IDFSZ,INGY,IMGE,IMGTC,IMK01,IMK02,IMENU,IDXP,IDYP,
     6 IXC,IYC,IXL,IXR,IYU,IYL,CS,MDELX,MDELY,JFLG,MLW,MUP,ISLX,ISLY,
     7 TOTAL,NYL,XPC,YPC,IDGLP,IMESG,IEDG,ISCFF,SOFF,KSLH,
     8 IMGFS,MAXDF,FOLV,KTC,IOPTN,IAREA,ISIZE,KCSR,KYES,KPSIZ,KPLL,
     9 T11,T12,T13,ISD(7),
     A IOFSET,KPLH,IFLP,ICUST,IMGPK,
     B T1,T2,T3,T4,T5,T6,T7,T8,T9,T10,
     C JSTY,JFG,JNG,JSHD,JRT,
     D ILPB,IBU,IED,IUV,IMR,IBL,ITCW,ITCCW,
     E IRCW,IRCCW,ISL,IRE,IFL,IPK,ICOM,
     F IOBU(1)
      EQUIVALENCE (IFCT(1),FACTR)
      NAMMK     = 2HMA
      NAMMK(2) = 2HRK
      NAMMK(3) = 2HR
      ISCT     = 2HSE
      ISCT(2) = 2HCT
      ISCT(3) = 2HR
C.....SET UP FIELD DEFINITIONS FOR MARKER DATA
C.....  JBM(1)  -- NUMBER OF PIECES IN MARKER
C               (2)  -- MARKER MAN IDENT. NUMBER
C               (3)  -- NUMBER OF SECTORS OF DATA
C               (4)  -- NUMBER OF STYLES
C               (5)  -- NUMBER OF SHADE AREAS
C               (6)  -- NEW/OLD FLAG
C               (7)  -- INTERNAL STYLE NUMBER & MASTER IMAGE
C               (8)  -- NUMBER OF SIZES
C               (9)  -- X% AND LEFT HALF WORD
C               (10) -- Y% AND RIGHT HALF WORD
C               (11) -- OPPOSITE FLAG
C               (12) -- RULE TABLE
C               (13) -- STRIPE GROUP
C               (14) -- SHADE CONSTRAINT
```

```
C                (15) -- BLOCK FLAG
C                (16) -- NUMBER OF SPLICE MARKS
C                (17) -- FLIP GROUP
C                (18) -- NAP GROUP
C                (19) -- MAXIMUM TILT
C                (20) -- HALF PIECE FLAG
      JRM    = JFLD(15,10)
      JRM(2) = JFLD(1,6)
      JRM(3) = JFLD(5,6)
      JRM(4) = JFLD(15,5)
      JRM(5) = JFLD(1,2)
      JRM(6) = JFLD(2,1)
      JRM(7) = JFLD(15,11)
      JRM(8) = JFLD(4,5)
      JRM(9) = JFLD(10,8)
      JRM(10) = JFLD(7,6)
      JRM(11) = JFLD(15,1)
      JRM(12) = JFLD(9,7)
      JRM(13) = JFLD(4,3)
      JRM(14) = JFLD(1,1)
      JRM(15) = JFLD(0,1)
      JRM(16) = JFLD(14,4)
      JRM(17) = JFLD(1,1)
      JRM(18) = JFLD(3,1)
      JRM(19) = JFLD(14,5)
      JRM(20) = JFLD(2,1)
      NAMGD    = 2HGR
      NAMGD(2) = 2HAD
      NAMGD(3) = 2HE
C..........SET UP FIELD DEFINITIONS FOR GRADED HEADER
C..........   JRG(1) -- NUMBER OF SIZES GRADED
C                (2) -- NUMBER OF SECTORS OF DATA
C                (3) -- SIZE
C                (4) -- RULE TABLE
      JRG=JFLD(12,4)
      JRG(2)=JFLD(8,3)
      JRG(3) = JFLD(15,8)
      JRG(4) = JFLD(7,7)
C..........SET UP FIELD DEFINITIONS FOR PIECE DATA
C..........   JRP(1)  -- RULE TABLE
C                (2)  -- SAMPLE SIZE
C                (3)  -- MIRROR FLAG
C                (4)  -- GRAIN FLAG
C                (5)  -- NUMBER OF SECTORS OF DATA
C                (6)  -- DATA
C                (7)
C                (8)  -- NEW/OLD FLAG
C                (9)  -- POINT TYPE
C                (10) -- RULE NUMBER
      JRP    = JFLD(15,7)
      JRP(2) = JFLD(8,8)
      JRP(3) = JFLD(5,1)
      JRP(4) = JFLD(4,1)
      JRP(5) = JFLD(3,4)
      JRP(6) = JFLD(15,9)
      JRP(8) = JFLD(9,1)
      JRP(9) = JFLD(15,4)
      JRP(10)= JFLD(11,12)
C..........SET UP FIELD DEFINITIONS FOR JOB QUEUE
      JRST = JFLD(15,1)
      JRAP = JFLD(14,1)
      JOLD = JFLD(13,1)
      JFLP = JFLD(12,1)
      JSRAD = JFLD(11,4)
      JRST = JFLD(7,8)
```

```
      JSTY = JFLD(15,9)
      IOUT = 6
      NAME     = 2HST
      NAME(2) = 2HYL
      NAME(3) = 2HE
C.....SET UP FIELD DEFINITIONS FOR STYLE DATA
C.....    JRS(1)  -- NUMBER OF PIECES
C                 (2)  -- RULE TABLE
C                 (3)  -- NUMBER OF SECTORS OF DATA
C                 (4)  -- OPPOSITE FLAG
C                 (5)  -- ROTATE FLAG
C                 (6)  -- STRIPE GROUP
C                 (7)  -- SHADE FLAG
C                 (8)  -- BLOCK FLAG
C                 (9)  -- YX
C                 (10) -- YX
C                 (11) -- FLIP GROUP
C                 (12) -- NAP GROUP
C                 (13) -- MAXIMUM TILT
C                 (14) -- HALF PIECE TABLE
C                 (15) -- PIECE MESSAGES
      JRS     = JFLD(15,8)
      JRS(2) = JFLD(9,7)
      JRS(3) = JFLD(3,4)
      JRS(4) = JFLD(15,1)
      JRS(5) = JFLD(2,1)
      JRS(6) = JFLD(5,3)
      JRS(7) = JFLD(3,1)
      JRS(8) = JFLD(4,1)
      JRS(9) = JFLD(15,5)
      JRS(10)= JFLD(7,8)
      JRS(11) = JFLD(1,1)
      JRS(12) = JFLD(M,1)
      JRS(13) = JFLD(14,5)
      JRS(14) = JFLD(9,7)
      JRS(15) = JFLD(9,10)
      MK     = 2HMP
      MK(2) = 2HKR
      MK(3) = 2H
C.....DATA READ FROM QUEUE
  500 CALL QDSC(3)
      CALL EXEC(14,RR,MTMP,2,ISCT,0)
C.....GET QUEUE ADDRESS FROM "SECTR"
      IAD = MTMP
C.....GET MARKER OPERATOR ID FROM "SECTR"
      IOMP = MTMP(2)
      IOUT = 6
      IAD = IAD+1
C.....READ MARKER ID
      DO 510 I=1,5
      L = IAD+I-1
  510 CALL MOST(L,IOMK(I))
C.....READ WIDTH
  530 IAD = IAD+5
      CALL MOST(IAD,ITMP)
      IOFSET = 0
C.....MASK OFF SLIDE FLAG
      IF(ITMP.GE.0) GO TO 535
      IOFSET = 1
      ITMP = IAND(ITMP,77777B)
  535 WIDTH = ITMP/100.0
C.....READ GOAL IN TENTHS OF INCHES
      IAD = IAD+1
      CALL MOST(IAD,ITMP)
      GOAL=ITMP/10.0
```

```
C..........READ PRINT, NAP, OLD/NEW, FLIP, # SHADES, #STYLES
      IAD = IAD+1
      CALL RDST (IAD,ITMP)
      JFLG = JLOF(JPRT,ITMP)
      NAP = JLOF(JNAP,ITMP)
      MKUP = 0
      IF(JLOF(JOLD,ITMP).EQ.0) MKUP = 2HFR
      IFLP = JLOF(JFLP,ITMP)
      NSHD = JLOF(JSHAD,ITMP)
      NUSTY = JLOF(JTST,ITMP)
C..........IF "OLD" THEN # STYLES IS # SUBSTITUTIONS
      IF(MKUP.EQ.2HFR) NUMSP = NUSTY
C..........READ X INCREASES
      IAD = IAD+1
      CALL RDST (IAD,ITMP)
      XPI = ITMP/10000.0 + 1.0
      CALL RDST(IAD+1,ITMP)
      YPI = ITMP/10000.0 + 1.0
C..........READ STRIPE OFFSET
      IAD=IAD+2
      CALL RDST(IAD,ITMP)
      SOFF=ITMP/100.0
C..........READ STRIPE
      IAD=IAD+1
      CALL RDST(IAD,ITMP)
      SY = ITMP/100.0
C..........READ PLAID
      IAD = IAD+1
      CALL RDST (IAD,ITMP)
      SX = ITMP/100.0
      IAD = IAD+1
      IF(MKUP.EQ.2HFR) GO TO 600
C..........START WITH OLD MARKER
      DO 540 I=1,5
      L = IAD+I-1
  540 CALL RDST(L,MKUP(I))
      IAD = IAD+5
C..........START FROM OLD MARKER
      DO 550 I=1,5
      L = IAD+I-1
  550 CALL RDST(L,IDMK(I))
C..........READ WIDTH
  555 IAD = IAD+5
      CALL RDST(IAD,ITMP)
      IOFSET = 0
C..........MASK OFF SLIDE FLAG
      IF(ITMP.GE.0) GO TO 565
      IOFSET = 1
      ITMP = IAND(ITMP,7777B)
  565 WIDTH = ITMP/100.0
C..........READ GOAL IN TENTHS OF INCHES
      IAD = IAD+1
      CALL RDST(IAD,ITMP)
      GPI=ITMP/10.0
C..........READ PRINT, NAP, OLD/NEW, FLIP, # SHADES, #STYLES
      IAD = IAD+1
      CALL RDST (IAD,ITMP)
      JFLG = JLOF(JPRT,ITMP)
      NAP = JLOF(JNAP,ITMP)
      MKUP = 0
      IF(JLOF(JOLD,ITMP).EQ.0) MKUP = 2HFR
      IFLP = JLOF(JFLP,ITMP)
      NSHD = JLOF(JSHAD,ITMP)
      NUSTY = JLOF(JTST,ITMP)
```

```
C..........IF "OLD" THEN # STYLES IS # SUBSTITUTIONS
      IF(MARK,EQ.2HFF) NUMFP = NUSTY
C..........READ % INCREASES
      IAD = IAD+1
      CALL NUST (IAD,ITMP)
      APC = ITMP/1000.0 + 1.0
      CALL NUST(IAD+1,ITMP)
      YPC = ITMP/1000.0 + 1.0
C..........READ STRIPE OFFSET
      IAD=IAD+2
      CALL NUST(IAD,ITMP)
      SOFF=ITMP/1000.
C..........READ STRIPE
      IAD=IAD+1
      CALL NUST(IAD,ITMP)
      SY = ITMP/1000.
C..........READ PLAID
      IAD = IAD+1
      CALL NUST (IAD,ITMP)
      SX = ITMP/1000.
      IAD = IAD+1
      IF(MARK.EQ.2HFF) GO TO 600
C..........START WITH OLD MARKER
      DO 540 I=1,5
      L = IAD+I-1
  540 CALL NUST(L,MARK(I))
      IAD = IAD+5
C..........START FROM OLD MARKER
    5 CALL DIREC (IABIN,MARK,ITMP,NSCMU,JSTAT)
      IF(JSTAT.EQ.0) GO TO 70
      GO TO (10,15,20,15,15),JSTAT
C..........ERROR CONDITIONS
C..........BIG PROBLEM
   10 CALL EXIT(1,0)
C..........MARKER NOT FOUND
   15 WRITE(NOUT,1004)
 1004 FORMAT ("*** OLD MARKER NOT AVAILABLE")
      CALL RTN
C..........WRONG PLATTER ON
   20 WRITE(NOUT,1005)
 1005 FORMAT("*** MOUNT MARKER PLATTER")
      CALL RTN
C..........SET UP STYLE DEFINITION AND SIZES
   70 CALL DOSC(1)
      CALL EXEC (14,55,IBUF,128,NAMMK,NSCMU)
C..........SET UP IBUF LENGTH IN VARIABLE "YARDS"
      ITMP = IBUF(128)
      IF(ITMP.EQ.0) ITMP = IBUF(7)
      YARDS = FLOAT(ITMP)
      NUSTY = JLOF(JBUF(4),IBUF(2))
      IF(NUSTY.LE.31) GO TO 72
      WRITE(NOUT,1022)
 1022 FORMAT("*** TOO MANY STYLES")
      CALL RTN
   72 JBS = 1
      JBS = 14
      JBIZ = 1
   75 NBZ = JLOF(JBUF(JBS),IBUF(JBW))
      ITMP = JLOF(JBUF(JBS),IBUF(JBW))
      CALL DIREC(14HSCM,IUST,ITMP,NSCST,ISTAT)
      IF(ISTAT.EQ.0) GO TO 78
C..........ERROR CONDITIONS
      GO TO (15,105,77,15,15),ISTAT
```

```
C..........STYLE NOT FOUND
  100 WRITE(OUT,1125)
 1125 FORMAT("*** STYLE NOT AVAILABLE")
      CALL RTN
C..........DISK PROBLEM
   77 WRITE(OUT,1024)
 1024 FORMAT("*** ERROR IN STYLE DATA")
      CALL RTN
C..........OUTLD INSTY ARRAY  --  INTERNAL STYLE #; RULE TABLE
   75 CALL EXEC (14,05,ITMP,2,NAMST,NSEST)
      IRLT = JLOF(JPS(2),ITMP(2))
      CALL JSTF(ITMP,JSTY,INSTY(JDS))
      CALL JSTF(IRLT,JPS(2),INSTY(JDS))
      IBEG = JDW+1
      IEND = IBEG+(NSZ-1)/2
      ICON = JDS*256
      ICTR = 1
C..........OUTLD ISIZ ARRAY  --  SIZE GROUP; SIZE INDICATOR
      DO 80 I1=IBEG,IEND
      ISIZ(NSIZ) = JLOF(JPM(9),IBUF(I1))+ICON
      NSIZ = NSIZ+1
      IF(ICTR.EQ.NSZ) GO TO 85
      ICTR = ICTR+1
      ISIZ(NSIZ) = JLOF(JPM(10),IBUF(I1))+ICON
      NSIZ = NSIZ+1
   80 ICTR = ICTR+1
   85 JDS = IEND+1
      JDS = JDS+1
      IF(JDS.LE.NOSTY) GO TO 75
      NSIZ = NSIZ-1
C..........USE IBUF FROM 4000 TO 4042 FOR NEW SIZES
  545 DO 547 I=4000,4042
  547 IBUF(I) = 0
C..........USE IBUF FROM 3900 TO 3931 FOR NEW STYLES
      DO 549 I=3900,3931
  549 IBUF(I) = 0
      IF(NOREP.EQ.0) GO TO 800
      DO 870 I=1,NOREP,2
C..........READ STYLE ID
      DO 551 I1=1,5
      L = IAD+I1-1
  551 CALL RDST(L,IDST(I1))
      CALL DIREC(100,IDST,INTST,JTMP,JSTAT)
      IF(JSTAT.EQ.0) GO TO 1552
      GO TO (915,915,77,915,915),JSTAT
C..........CHECK VALIDITY OF STYLE
 1552 DO 552 I1=1,NOSTY
      IF(JLOF(JSTY,INSTY(I1)).EQ.INTST) GO TO 553
  552 CONTINUE
      WRITE(OUT,1125) (IDST(I1),I1=1,5)
 1125 FORMAT ("***STYLE NOT IN MARKER",5A2)
      CALL RTN
C..........STYLE VALID
  553 IBIST = I1
      IRLT = JLOF(JPS(2),INSTY(I1))
C..........READ # OF SIZES
      IAD = IAD+5
      CALL RDST(IAD,NSZ)
      IAD = IAD+1
      DO 555 I1=1,NSZ
C..........READ SIZES
      DO 555 I2=1,3
      L = IAD+I2-1
  555 CALL RDST (L,ITMP(I2))
      IAD = IAD+3
```

```
      CALL SIZEP (1,IPLT,MTMP,ITMP)
      IF(IFLG.P) 554,515,555
  555 IST = ITMP+INTST*256
C.....FIND SIZE IN ISIZ
      DO 557 I2=1,NSIZ
      IF(ISIZ(I2).EQ.IST) GO TO 558
  557 CONTINUE
      WRITE(NOUT,1020) (MTMP(I2),I2=1,3)
 1020 FORMAT ("*** SIZE NOT IN MARKER",3A2)
      CALL RTN
C.....SIZE IS VALID, SAVE INDEX IN IDUM(3800) +
  558 IDUM(3800+I1) = I2
      ISIZ(I2) = -ISIZ(I2)
  559 CONTINUE
C.....READ NEW STYLE ID
      DO 561 I1=1,5
      L = IAL+I1-1
  561 CALL GKST(L,IDST(I1))
      CALL DIREC (IPLOR,IDST,ITMP,JTMP,JSTAT)
      IF(JSTAT.EQ.0) GO TO 562
      GO TO (515,512,77,515,515),JSTAT
C.....STYLE IS VALID, STORE IN IDUM(3900) +
  562 CALL GKSC(K)
      CALL PAEL (14,55,ITMP,2,NAMST,JTMP)
      IPLTN = JLOK(JMS(2),MTMP(2))
      CALL JSTF(ITMP,JSTY,IDUM(3901+INTST))
      CALL JSTF(IPLTN,JMS(2),IDUM(3902+INTST))
C.....READ # OF SIZES
      IAD = IAD+5
      CALL GKSI(IAD,NSZ)
      IAD = IAD+1
      DO 566 I1=1,NSZ
C.....READ SIZES
      DO 564 I2=1,3
      L = IAD+I2-1
  564 CALL GKSI(L,MTMP(I2))
      IAD = IAD+3
      CALL SIZEP(1,IPLTN,MTMP,ITMP)
      IF(ITMP) 565,515,565
  565 ITMP = IDUM(3902+I1)
      IDUM(4000+JTMP) = ITMP+INTST*256
  566 CONTINUE
  570 CONTINUE
      GO TO 500
C.....START FRESH
  600 NSIZ = 0
C.....CHECK MARKER VALIDITY
      CALL DISEC (IPLOR,IDMK,ITMP,JTMP,JSTAT)
      IF(JSTAT.EQ.0) GO TO 520
      GO TO (515,625,525,515,515),JSTAT
C.....DISK PROBLEM
  515 CALL ERR(0,0)
C.....DUPLICATE NAME
  525 WRITE(NOUT,1030)
 1030 FORMAT ("*** DUPLICATE MARKER ID")
      CALL RTN
C.....MARKER FILE NOT FOUND
  625 WRITE(NOUT,1040)
      CALL RTN
  650 DO 651 I=1,NOSTY
C.....READ STYLE ID
      DO 615 I1=1,5
      L = IAD+I1-1
  615 CALL GKSI(L,IDST(I1))
```

```
C..........CHECK STYLE VALIDITY
      CALL DIREC (14,A0,IDST,INTST,NSCST,JSTAT)
      IF(JSTAT.EQ.0) GO TO 615
C..........ERROR CONDITIONS
      GO TO (615,612,77,515,515),JSTAT
  612 WRITE(MOUT,1006)
      CALL RTN
C..........RULE TABLE?
  615 CALL DOSC(0)
C..........BUILD INSTY ARRAY  --  INTERNAL STYLE #; RULE TABLE
      CALL EXEC (14,69,ATMP,2,NAMST,NSCST)
      IRLT = JLOK(JPS(2),ATMP(2))
      CALL JSTF(INTST,JSTY,INSTY(I))
      CALL JSTF(IRLT,JPS(2),INSTY(I))
C..........READ # SIZES
      IA0 = IA0+0
      CALL RWST(IA0,NSZ)
      IA0 = IA0+1
      DO 650 I1=1,NSZ
      NSIZ = NSIZ+1
C..........READ SIZE
      DO 620 I2=1,3
      L = IA0+I2-1
  620 CALL RWST(L,ATMP(I2))
      IA0 = IA0+3
C..........CHECK FOR ROTATE BY SIZE FLAG
      IRT=0
      IF(IAND(ATMP,100000B).EQ.100000B) IRT = 1
      ATMP=IAND(ATMP,77777B)
      CALL SIZEP(1,IRLT,ATMP,ITMP)
      IF(ITMP) 630,515,640
  630 WRITE(MOUT,1012)
 1012 FORMAT("*** SIZE NOT AVAILABLE")
      CALL RTN
C..........BUILD "ISIZ" ARRAY=ROTATE FLAG;STYLE POINTER;SIZE INDICATOR
  640 ISIZ(NSIZ) = ITMP+I*256
      CALL JSTF(IRT,JPRT,ISIZ(NSIZ))
  650 CONTINUE
  800 CALL STOP
      CALL DOSC(M)
C.....READ OPTIONS FROM "OPTNS" FILE
      ISCT(1) = 2HUP
      ISCT(2) = 2HTN
      ISCT(3) = 2HS
      CALL EXEC(14,69,IOPTN,1,ISCT,0)
      IF(ISSW(1).LT.3)WRITE(MOUT,2222)NAMMK,NSCMU
 2222 FORMAT(3A2,I6)
C..........SET UP STRIPE GROUP DEFINITIONS
      ISD(1) = 0
      ISD(2) = 9999
      ISD(3) = 9999
      ISD(4) = 9999
      ISD(5) = 9999
      ISD(6) = 9999
      ISD(7) = 9999
      CALL EXEC(3,MK)
      CALL MARK
      CALL ERRB
      END
      ENDS
```

```
FTN4
      PROGRAM MARK,5
C.....DEC. 6, 1972
C.....MARGARET NOWAK
C.....PROGRAM: MARK
C.....VERSION 2.0
C.....SEGMENT OF MASK THAT POSITIONS THE PIECES AND
C.....PREPARES INFORMATION FOR SETTING UP DISPLAY FILES
      DIMENSION IDUM(521)
      DIMENSION IBUF(128),MK(3),IFLOT(2),INTN(6),NSC(31),NAMK(3)
      COMMON NOUT,LABF(3),LABR(3),NOSTY,INSTY(31),JBS(15),NAMST(3),
     1 IBUF(5),NSCMU,JBM(2),NAMMK(3),JBG(4),NAMGO(3),JBP(10),
     2 IDUM(5),IDUP,NPEC,NSHD,FACTR,NSIZ,ISIZ(42),WIDTH,EFF,SX,SY,KSIL
     3 YAMOS,NAP,KINI,KFLG,KINC,KSIZ,KXMN,KYMN,KANG,KXMX,KYMX,KDPT,
     4 KAD,KADP,KDP,CSMX,MAWD,CSMN,NAGE,IMGP,KREL,ITYPE,
     5 NOIMG,IOFS7,IMGY,IMGE,IMGTC,IMK01,IMK02,IMENU,IDXP,IDYP,
     6 IXC,IYC,IXL,IXM,IYU,IYL,CS,MDELX,MDELY,JFLG,MLW,MUP,ISLX,ISLY,
     7 TOTAL,MYL,XPC,YPC,IMGLP,IMESG,IEDG,ISOFF,SOFF,KSLH,
     8 IMGFS,MAXDF,NSLV,NTL,IOPTN,IAKE,ISIZE,KCSR,KYES,KPSIZ,KPLL,
     9 IT1,IT2,IT3,IT4,IT5,IT6,IT7,IT8,IT9,IT10,
     A IOFSET,KPLM,IFLP,ICOST,IMGPK,
     B T1,T2,T3,T4,T5,T6,T7,T8,T9,T10,
     C JSTY,JFG,JNG,JSHD,JRT,
     D ILPN,IRU,IFD,IUV,IRK,IBL,ITCW,ITCCW,
     E IRCW,IRCCW,ISL,IRE,IPL,IPK,ICOM,
     F IDUM(1)
      EQUIVALENCE (IFLOT,FLOT)
C.....SET MINIMUM AREA FOR PIECES WITH SIZE ON PIECE
      NSIZ = 30,0
      NANG = 0
      IK    = 2HMR
      MK(2) = 2HKC
      MK(3) = 2H
C.....SET UP FIELD DEF. FOR OVERHEAD
      JFG=JFLD(5,1)
      JNG=JFLD(7,1)
      JSHD = JFLD(6,3)
      JRT=JFLD(9,1)
      IF(NBUF.EQ.2HER) GO TO 100
C.....STARTING FROM OLD MARKER
C.....READ IN PIECE POSITION & INFO. FROM MARKER FILE
      CALL GOSC(1)
C.....IF(ISSW(1).LT.4) WRITE(MOUT,2222) NAMMK,NSCMU
C...2222 FORMAT(3A2,I8)
      CALL EXEC (14,05,IBUF,128,NAMMK,NSCMU)
      NPEC = JLOF(JBM   ,IBUF   )
      LSEC = JLOF(JBM(3),IBUF   )
C.....READ DATA MULTIPLIER
      IFLOT    = IBUF(4)
      IFLOT(2) = IBUF(5)
      FACTR = FLOT
C.....SCALE LENGTH
      YAMOS = ((IBUF(126)+32000,0)*FACTR)/36.0
C.....INITIALIZE IDUM POINTERS FOR OVERHEAD
      KINI = 1
      KFLG = KINI+NPEC
      KINC = KFLG+NPEC*2
      KSIZ = KINI+NPEC
      KXMN = KSIZ+NPEC
      KYMN = KXMN+NPEC
      KANG = KYMN+NPEC
      KXMX = KANG+NPEC
      KYMX = KXMX+NPEC
```

```
C..........OPTIONALLY SET UP PLAID & STRIPES
      KPLL = KYMX
      IF(KX.NE.7.4) KPLL = KYMX+NPEC
      KSTL = KPLL
      IF(SY.NE.1.M) KSTL = KPLL+NPEC
      KTOP = KSTL+NPEC
      IDX = 1
      KSEC = 1
   10 CALL EXEC (14,36,IBUF,128,NAMMK,NSCMU+KSEC)
      DO 20 I1=5,125,8
      IDUM(KXMU+IDX) = IBUF(I1+5)
      IDUM(KYMU+IDX) = IBUF(I1+6)
      IDUM(KANG+IDX) = IBUF(I1+7)
      IF(IDX.EQ.NPEC) GO TO 30
   20 IDX = IDX+1
      KSEC = KSEC+1
      IF(KSEC.LT.LSEC) GO TO 10
      WRITE(IOUT,1002)
 1002 FORMAT ("***ERROR IN OLD MARKER DATA")
      CALL PTN
   30 NPEC = IDX
      IF(IDUM(39M1).EQ.0) GO TO 100
C..........SUBSTITUTE STYLES AND SIZES
      DO 35 I1=1,NOSTY
   35 ISTY(I1) = IDUM(39M+I1)
      DO 37 I1=1,NSIZ
   37 ISIZ(I1) = IDUM(40.00+I1)
C..........FRESH START
C..........READ PIECE INFO,FROM STYLE FILE
C..........COUNT TOTAL NUMBER OF PIECES IN MARKER
  100 NPEC = 0
C..........JTP1 IS FIELD DEF. FOR STYLE IN "ISIZ" JUST IN "MRKB"
      JTP1 = JFLD(14,7)
      DO 125 I=1,NOSTY
      NSZ = 0
C..........COUNT # OF SIZES OF STYLE
      DO 101 J=1,NSIZ
      IF(JLOF(JTP),ISIZ(J)).NE.I) GO TO 101
      NSZ = NSZ+1
  101 CONTINUE
      INTS = JLOF(JSTY,ISTY(I))
      CALL DIREC (14,36,INTN,INTS,NSCST,ISTAT)
      IF(ISTAT.EQ.0) GO TO 104
C..........ERROR CONDITIONS
      GO TO(115,102,115,115,102),ISTAT
  102 WRITE(IOUT,1003)
 1003 FORMAT("***STYLE NOT AVAILABLE")
      CALL PTN
C..........SAVE BEGINNING STYLE SECTOR
  104 NSC(I)=NSCST
      CALL READ(14,36,IBUF,1,NAMST,NSCST)
      NPCE = JLOF(JPCS(1),IBUF)
      NPEC = NPEC+NPCE*NSZ
  125 CONTINUE
      IF(IGROUP.EQ.2HFR) GO TO 127
C..........OLD MARKER, CHECK # OF PIECES
      IF(NPEC.EQ.NOPEC) GO TO 129
      WRITE(IOUT,1004)
 1004 FORMAT("*** STYLES IN OLD MARKER DO NOT COMPARE TO NEW STYLES")
      CALL PTN
C..........SET UP IDUM POINTERS AND SET UP PIECE INFO,
  127 KINT = 0
      NFLG = KINT+NPEC
      NINC = NFLG+NPEC*2
```

```
      KSIZ = KINC+NPEC
      KXIN = KSIZ+NPEC
      KYMN = KXMN+NPEC
      KXMX = KXMN+NPEC
      KYMX = KXMX+NPEC
C.....OPTIONALLY SET UP PLAID & STRIPES
      KPLL = KYMX
      IF(SX.NE.0.0) KPLL = KYMX+NPEC
      KSTL = KPLL
      IF(SY.NE.0.0) KSTL = KPLL+NPEC
      KTOP = KSTL+NPEC
C.....SET UP PIECE OVERHEAD
  120 IDX = 1
      DO 130 I=1,NOSTY
      NSCST = NSC(I)
      JDX = 1
      LSEC = JLOF(JBS(2),INSTY(I))
      KSEC = 0
  130 CALL EXEC (14,06,IBUF,128,NAMST,NSCST+KSEC)
      IF(KSEC.NE.0) GO TO 135
      LSEC = JLOF(JBS(3),IBUF)
      LPEC = JLOF(JBS(1),IBUF)
  135 DO 170 I1=0,127,5
      IF(JDX.GT.LPEC) GO TO 200
      JDX = JDX+1
C.....SET UP FLAGS IN "ITMP", "JTMP"
      ITMP = IBUF(I1+3)
      I3 = IBUF(I1+2)
      CALL JSTF(JLOF(JBS(5),I3),JRT,ITMP)
      CALL JSTF(JLOF(JBS(11),I3),JFG,ITMP)
      CALL JSTF(JLOF(JBS(12),I3),JNG,ITMP)
      CALL JSTF(0,JSM0,ITMP)
      CALL JSTF(JLOF(JBS(7),I3),JBM(14),ITMP)
      JTMP = 0
      CALL JSTF(JLOF(JBS(6),I3),JBM(13),JTMP)
      IDX1 = IDX
      DO 140 I3=1,NSIZ
      IF(JLOF(JTMP1,JSIZ(I3)).NE.I) GO TO 140
C.....VALID SIZE
      IDUM(KINC+IDX) = IBUF(I1)
      IDUM(KFLG+IDX) = ITMP
      IDUM(KFLG+NPEC+IDX) = JTMP
C.....ROTATE BY SIZE?
      LTMP = JLOF(JRT,ITMP)
      IF(ISIZ(I3).LT.0) LTMP = 1-LTMP
      CALL JSTF(LTMP,JRT,IDUM(KFLG+IDX))
      IDUM(KINC+IDX) = IBUF(I1+4)
      IDUM(KSIZ+IDX) = IAND(ISIZ(I3),77777B)
      CALL JSTF(I3,JBM(9),IDUM(KSIZ+IDX))
      JDX = IDX+1
  140 CONTINUE
C.....ASSIGN MASTER/SLAVE STATIS
      IF(JLOF(JBS(14),IBUF(I1+3)).EQ.0) GO TO 170
      ITABL = JLOF(JBS(14),IBUF(I1+3))
      CALL HALFS(IDX1,IDX-1,ITABL,IRLT)
  170 CONTINUE
      KSEC = KSEC+1
      IF(KSEC.LE.LSEC) GO TO 130
  180 WRITE(IOUT,1000)
 1000 FORMAT ("***ERROR IN STYLE DATA")
      CALL EXIT
  200 CONTINUE
C.....REMOVE ROTATE FLAG FROM "ISIZ"
      DO 210 I=1,NSIZ
  210 ISIZ(I)=IAND(ISIZ(I),77777B)
```

```
C..........POSITION PIECES
C..........SET UP WINDOW DIMENSIONS
  300 LSIZ = 0
      LPEC = -1
      LINC = 0
      DIM = 0.0
      TOTAL = 0.0
      TOTP = 0.0
      KTMP = KTMP+1
      KTMP1 = KTMP
      CALL DESC(0)
C..........CALCULATE TOTAL AREA AND SET UP DIMENSIONS
      DO 305 I1 = 1,NPEC
C..........SAVE INTERVAL # AND % TO STORE ON DISK
      IDUM(KTMP) = IDUM(KINT+I1)
      IDUM(KTMP+1) = IDUM(KINC+I1)
      IDUM(KTMP+2) = 0
      KTMP = KTMP+3
      MPEC = IDUM(KINT+I1)
C..........MSIZ IS SIZE; RULE TABLE
      MSIZ = JLDF(JBG(13),IDUM(KSIZ+I1))
      ITMP = JLDF(JBG(5),IDUM(KSIZ+I1))
      ITMP = JLDF(JBG(9),ISIZ(ITMP))
      JTMP = 0
      CALL JSTF(JLDF(JBG(2),INSTY(ITMP)),JBG(4),JTMP)
      CALL JSTF(MSIZ,JBG(3),JTMP)
      MSIZ = JTMP
      MINC = IDUM(KINC+I1)
      IF(MPEC.EQ.LPEC) GO TO 320
      LPEC = MPEC
      LSIZ = 0
      LINC = 0
C..........FIND PIECE IN "GRADE"
      INTN(5) = MPEC
      CALL DIREC (30,40,INTN,JTMP,NSEC,JSTAT)
      IF(IBBA(1).LT.0) WRITE(6,9000) MPEC,MSIZ,INTN,JTMP,NSEC,JSTAT
 9000 FORMAT (2X8A)
      IF(JSTAT.EQ.0) GO TO 320
C..........ERROR CONDITIONS
      GO TO (315,317,315,315,315),JSTAT
  315 CALL STOP
      CALL ERR(1,0)
  317 WRITE(6OUT,1000)
 1000 FORMAT ("***PIECE NOT AVAILABLE")
      CALL STOP
  320 IF(MSIZ.EQ.LSIZ.AND.MINC.EQ.LINC) GO TO 339
      CALL EXEC (14,ED,IBUF,128,NANGU,NSEC)
      IF(IBBA(1).LT.0) WRITE(6,9000) IBUF
C..........GET DATA ON MSIZ AND MINC
      LSEC=JLDF(JBG(2),IBUF)
      NS=(12*LSEC)/((12*JLDF(JBG,IBUF))+1)
      NSEC=0
      KS = 1
      LANG = IBUF(2)
      NANG = 0
      IF(MKUP.NE.2HFR) NANG = IDUM(KANG+I1)
C..........NSCP IS # OF SECTORS PER PIECE
C..........MSCP IS BEGINNING SECTOR OF FIRST SIZE PIECE
      NSCP = (LSEC-(NS-1)/12-1)/NS
      MSCP = (NS-1)/12+1+NSEC
      LSEC=MSCP-1
  325 DO 330 I2=3,12,,10
      IF(MSIZ.EQ.IBUF(I2).AND.MINC.EQ.IBUF(I2+9)) GO TO 340
      IF(KS.EQ.NS) GO TO 317
      MSCP = MSCP+NSCP
```

```
  334 KS = KS+1
      KSEC = KSEC+1
      IF(KSEC + NSEC.LE.LSEC)GO TO 335
      WRITE(MOUT,1005)
 1005 FORMAT ("***ERROR IN GRADE DATA")
      CALL PTM
  335 CALL EXEC (14,55,IBUF,128,NAMGD,KSEC+KSEC)
      GO TO 325
C.....SLAVE PIECE?
  339 IF(JLOF(JRT,IDUM(KFLG+I1)),EQ,1) GO TO 357
      NANG = 0
      IF(MKUP.EQ.2HFR) GO TO 350
      NANG = IDUM(KANG+I1)
C.....SLAVE PIECE?
  345 IF(JLOF(JRT,IDUM(KFLG+I1)),EQ,1) GO TO 357
C.....SAVE AREA, PERIMETER, AND DIMENSIONS
      IFLOT    = IBUF(I2+5)
      IFLOT(2) = IBUF(I2+6)
      AREA = FLOT
      IFLOT    = IBUF(I2+7)
      IFLOT(2) = IBUF(I2+8)
      PREM = FLOT
      XMN = IABS(IBUF(I2+1))
      YMN = IABS(IBUF(I2+2))
      LSIZ = MSTZ
      LINC = LINC
      JFP = JLOF(JBS(4),IDUM(KFLG+I1))
C.....ADJUST DIMENSIONS TO INITIAL DISPLAY POSITION
C.....AND SET UP GRAIN ANGLE IN "LANG"
      IF(LANG.EQ.2HFR) CALL GRAIN(MSCP,LANG)
      CALL MAXMN(MSCP,XMN,YMN,NANG,LANG,JFP)
      IF(LSS(1),LT,0) WRITE(6,9001) XMN,YMN,AREA
 9001 FORMAT (3F10.3)
      IF(XPC.EQ.1.0.AND.YPC.EQ.1.0) GO TO 350
C.....ADJUST DIMENSIONS, AREA,& PERIMETER BY STRETCH FACTORS
      XMN = XMN*XPC
      YMN = YMN*YPC
      CALL MAREA(MSCP,AREA,PREM)
  350 TOTAL = TOTAL+ABS(AREA)
      TOTP = TOTP+PREM
C.....SET ANGLE TO 180 IF ROTATE FLAG IS SET
      IF(JLOF(JRT,IDUM(KFLG+I1)),EQ,1) NANG = 1800
C.....STORE BEGINNING SECTOR AND AREA IN OVERHEAD
      IDUM(KINT+I1) = MSCP
      IDUM(KINC+I1) = IRND(ABS(AREA)*10.0)
C.....CHECK FOR SIZE ON PIECE AND SET FLAG.
      IF (JLOF(JFLD(6,1),IOPTN),NE,0.AND.ABS(AREA).GT.ASIZ)
     +CALL JSTF (1,JBM(15),IDUM(KFLG+NPEC+I1))
      IDUM(KTMP-1) = LANG
      DX = XMN*1.414
      DY = YMN*1.414
      IF(ISLX*ISLY.EQ.0.OR.JLOF(JBS(6),IDUM(KFLG+I1)),EQ,0)
     1 GO TO 351
      DX = DX+SX
      DY = DY+SY
      IF(SY.EQ.99.0) DY = DY-SY
C.....ACCUMULATE LARGEST DIMENSION FOR SCALING
  351 DIM = AMAX1(DX,DIM)
      DIM = AMAX1(DY,DIM)
C.....STORE DIMENSIONS TEMPORARILY IN KXMX & KYMX
      IF(MKUP    .NE.2HFR) GO TO 355
      IDUM(KANG+I1) = NANG
      IDUM(KXMX+I1) = XMN/2.0
      IDUM(KYMX+I1) = YMN/2.0
      GO TO 350
  355 IDUM(KXMX+I1) = XMN
```

```
      IDUM(KYMX+I1) = YMX
      GO TO 360
C.....SLAVE PIECE
  357 IDUM(KIMT+I1) = MSLP
      IDUM(KIMC+I1) = M
      IDUM(KIMP+1) = LANG
      IDUM(KXMN+I1) = X
      IDUM(KXMX+I1) = W
      IDUM(KYMN+I1) = H
      IDUM(KYMX+I1) = Y
      IDUM(KYMN+I1) = W
  360 CONTINUE
      IF(JPLG.EQ.1) WRITE(6,1001) TOTAL,TOTP
 1001 FORMAT ("AREA = ",F10.3,2X,"LENGTH = ",F10.3)
C.....COMPUTE SCALE (FACTR), MINIMUM COORDINATE SCALE (CSMN),
C.....MAXIMUM COORDINATE SCALE (CSMX), GOAL LINE (NRGE)
C.....AND MARKER DISCRIPTION IN MARKER UNITS
      DIM = DIM*.002
      DEL=EFF
C.....SET UP SLIDE CONSTANT
      IF(MKUP.NE.2HFR) ISLDE = 1500
C.....USE LENGTH IF IT IS LONGER THAN GOAL
      IF(MKUP.NE.2HFR.AND.YARDS*36.0.GT.EFF) DEL=YARDS*36.0
      EFF=0.0
      IF(MKUP.NE.2HFR) DIM = FACTR*4096.0
      IF(MKUP.EQ.2HFR)FACTR = DIM/4096.0
      CSMX = AMIN1(DIM/WIDTH,1.0)
      MKWD = IFIX(WIDTH/FACTR)
      CSMN = CSMX*.125
      NRGE = IFIX(DEL/FACTR)
      NRGE = -32000+NRGE
      ISLX = SX/FACTR
      ISLY = SY/FACTR
      ISOFF = SOFF/FACTR
      IF(ISSW(1).LT.0) WRITE(6,9001)DIM,EFF,DEL,WIDTH,TOTAL,CSMX,
     1 FACTR,CSMN
      IF(ISSW(1).LT.0) WRITE(6,9002) MKWD,NRGE
 9002 FORMAT(6I8)
      DO 365 I1=1,NPEC
C.....IGNOR SLAVE PIECE
      IF(JLOF(JBM(2),IDUM(KFLG+I1)),EQ.1) GO TO 365
      IF(MKUP.NE.2HFR) GO TO 363
      IDUM(KXMX+I1) = (IDUM(KXMX+I1)/500.0)/FACTR
      IDUM(KYMX+I1) = (IDUM(KYMX+I1)/500.0)/FACTR
  363 IF(ISSW(1).LT.0) WRITE(6,9002)IDUM(KXMX+I1),IDUM(KYMX+I1)
      IF(ISLX*ISLY.EQ.0.OR.JLOF(JBM(15),IDUM(KFLG+I1)),EQ.0)
     1 GO TO 365
C.....INCREASE DIMENSIONS FOR BLOCKING
      IDUM(KXMX+I1) = IDUM(KXMX+I1)+ISLX
      IF(SY.EQ.99.0) GO TO 365
      IDUM(KYMX+I1) = IDUM(KYMX+I1)+ISLY
  365 CONTINUE
      IF(MKUP.EQ.2HFR) GO TO 370
C.....SLIDE PIECES IF IOFSET=1
      IF(IOFSET.EQ.1) GO TO 367
      DO 366 I1=1,NPEC
      IDUM(KXMX+I1) = IDUM(KXMX+I1)+ISLDE
  366 IDUM(KXMN+I1) = IDUM(KXMN+I1)+ISLDE
      NRGE = NRGE+ISLDE
C.....SET COORDINATE SCALE AND INITIAL WINDOW
C.....POSITION FOR DEL MARKER
  367 IXDEL = IABS(-32000-NRGE)
      IXDEL = IXDEL+IXDEL/2
      IYDEL = MAX0(IXDEL,4096,2*MKWD)
      IXC = -32000+IXDEL/2
```

```
      CS = IFIX(2046./FLOAT(IXDEL)*2047.0)
      CS = CS/2047.0
      IYC = FLOAT((IXDEL-MKWD)/2)-256.1/CS+.5
C.....POSITION PIECE BY SETTING KXMN,KYMN,KXMX,KYMX
  370 CALL GPRED
      IF(ISSW(1).LT.0) WRITE(6,9002)(IDUM(KINT+K),IDUM(KFLG+K),
     * IDUM(KFLG+NPEC+K),
     1 IDUM(KSIZ+K),IDUM(KXMN+K),IDUM(KYMN+K),IDUM(KANG+K),
     2 IDUM(KXMX+K),IDUM(KYMX+K),K=1,NPEC)
      IXL = IXL-2048./CS
C.....CALL MRKC TO BUILD DISPLAY FILE
  375 CALL MRKC(0)
C.....STORE ABS(I) FOR SIZES AT END OF IDUM
      IDF3Z=MAXDF-NSIZ*3
      ISAVS=IDF3Z+1
      LSTY = 0
      DO 530 I=1,NSIZ
      MSTY = JLOF(JBS(9),ISIZ(I))
      IF(MSTY.EQ.LSTY) GO TO 530
      IPLT = JLOF(JBS(2),INSTY(MSTY))
      LSTY = MSTY
  530 ISZ=JLOF(JBS(10),ISIZ(I))
      CALL SIZEF(2,IPLT,NAMR,ISZ)
      IDUM(IMGPS)=NAMR
      IDUM(IMGPS+1)=NAMR(2)
      IDUM(IMGPS+2)=NAMR(3)
      IMGPS = IMGPS+3
  550 CONTINUE
      NAMR = 2HCH
      NAMR(2) = 2HE*
      NAMR(3) = 2H
C.....STORE INTERNAL #,% & GRAIN ANGLE ON
C.....DISK TEMPORARILY INTO "CHEK"
      NWDS = NPEC*3
      N = 2*((MAXDF+ICOM)/128)+2
      CALL EXEC(15,55,IDUM(KTMP1),NWDS,NAMR,N)
      CALL EXEC (6,'MK')
      CALL MARK
      CALL MRKC
      END
      ENDS

FTN4
      PROGRAM MRKC,5
C.....DEC. 11, 1972
C.....MARGARET KOWAN
C.....PROGRAM: MARK
C.....VERSION 2.0
C.....SEGMENT OF MARK THAT BUILDS THE DISPLAY FILE
      DIMENSION MK(3),ITMP(6),NAMT(3)
      COMMON MOUT,LABF(3),LABR(3),NOSTY,INSTY(31),JBS(15),NAMST(3),
     1 MKUP(5),NSCMD,JBM(20),NAMMK(3),JBG(4),NAMGD(3),JBP(10),
     2 IDMK(5),IODP,NPEC,NSMD,FACTR,NSIZ,ISIZ(42),WIDTH,EFF,SX,SY,KSTL,
     3 YAMOS,NAP,KINT,KFLG,KINC,KSIZ,KXMN,KYMN,KANG,KXMX,KYMX,KDPT,
     4 KAD,KADP,KDF,CSMX,MPWD,CSMN,NRGE,IMGP,KKEL,ITYPE,
     5 NOIMG,IDFS7,IMGY,IMGE,IMGTC,IMKO1,IMKO2,IMENU,IDXP,IDYP,
     6 IXC,IYC,IXL,IXR,IYU,IYL,CS,MDELX,MDELY,JFLG,MLW,MUP,ISLX,ISLY,
     7 TOTAL,MYL,XPC,YPC,IMGLP,IMESG,IEDG,ISOFF,SOFF,KSLH,
     8 IMGPS,MAXDF,KSLV,KTC,IOPTN,IAREA,ISIZE,KCSR,KYES,KPSIZ,KPLL,
     9 IMGIN,ICPEC,IPKC,IT4,IT5,IT6,IT7,IT8,IT9,IT10,
     A IOFSET,KPLH,IFLP,ICUST,IMGPK,
     B ILPL,ISHDO,ISMDL,T3,T4,T5,T6,T7,T8,T9,ILOOP,INSCP,
     C JSTY,JFG,JNG,JSHD,JRT,
     D ILPB,IMD,IBD,IOV,IBR,IBL,ITCW,ITCCW,
     E IRCW,IRCCW,ISL,IRE,IFL,IPK,ICOM,
     F IDUM(1)
```

```
      DATA MK/2HMR,2HKE,2H  /,NAMT/2HCH,2HEK,2H  /
      ILOOP = 0
      KSL=1
      KFLM = 2
C.....CHECK FOR ROTATE OPTION
      IRC= = 99
      IRCCW = 99
      JB4 = JFLD(4,1)
      IF(JLOF(JB4,IOPTN).EQ.0) GO TO 5
      IRCW = 3
      IRCCW = 4
    5 CONTINUE
      IBW=5
      IBU=6
C.....CHECK OPTION FOR FLIP
      JB3 = JFLD(3,1)
      IFL = 99
      IF(JLOF(JB3,IOPTN).EQ.1) IFL = 7
      ILPB=8
      IKE=9
      IPKC = 10
C.....CHECK OPTION FOR TILT
      ITCW = 99
      ITCCW = 99
      JB2 = JFLD(2,1)
      IF(JLOF(JB2,IOPTN).EQ.0) GO TO 100
      ITCW = 11
      ITCCW = 12
  100 CONTINUE
      IFL=13
      IBU=14
      IDV=15
      IPA=16
      IPGP = 0
      KREL = 0
      ITYPE = 0
      KSLH = 0
      KSLV = 0
      ICPKC = 0
C.....SET SHADE LIMITS TO MARKER LIMITS AS DEFAULT
      ISHDU = MKHD/2
      ISHDL = ISHDU-MKHD
C.....SET FLIP AND PACK BITS TO 0 IN "ICUST"
      ICUST = 0
      IEDG = -32000
      KAD = KSTL+NPEC
      KADP=KAD+8
      KDF = KADP+NPEC+4
      JDIMG = NPEC
      IDEND = 1
      IMSY = 2
      IMGIU=3
      IMGE=4
      IPGPS=5
      IMGLP=6
      IMESG=7
      IMGIC=8
      IMADL = JDIMG+1
      IMADU = JDIMG+2
C.....ZERO OUT DISPLAY FILE AREA
      IBEG = KAD+1
      DO 10 I=IBEG,IDFSZ
   10 IDUM(I) = 0
      IDXP = 0
      IDYP = 0
```

```
C.....INITIAL CALLS FOR "DSPLY"
      CALL NSPAC(2048)
      CALL INDSP (KOPT,IDUM,IDFSZ)
C.....BUILD DISPLAY FILE FOR MENU
      CALL PRGI
C.....END OF TABLE CODE
      IDUM(KADP+NUING+3) = 1B##0B
C.....SET UP SCALING FOR INCHES TO MARKER UNITS
      GIM = FACTR*4.0/0.
      CALL SCALE (0.,0.,GIM,GIM,1)
C.....BUILD DISPLAY FILE FOR CURSOR
      IDUM(KAD+IMGTC)=KOPT
      KTC = KOPT
      CALL CROSS(0.,0.,KTC,IDUM)
      KOPT=KOPT+9
C.....SET CSP FOR PIECES
      KCS=KOPT+1
      IDUM(KOPT)=4B.23B
      IDUM(KOPT+1)=IFIX(2047.0*CS)*16+1
      KOPT=KOPT+2
C.....BUILD DISPLAY FOR PIECES AND OUTLINE
      CALL PIDUM
C.....TURN DISPLAY ON
      CALL START(4B,IDUM(KAD+1),IDUM)
C.....TURN OFF "BUSY" LIGHT
      CALL LIGHT(2,0)
      JFLG = 0
      CALL DUSC(0)
C.....DUMP COMMON TO "CHEK"
      N = (MAXOF+ICOM)/128+1
      CALL EXEC(15,55,EFF,MAXOF+ICOM,NAMT,N)
C.....ZERO FIRST SECTOR OF "CHEK"
      CALL EXEC (15,56,0,1,NAMT,0)
   20 CALL EXEC (6,NR)
      CALL MARK
      CALL MRKE
      END
      ENDS


FTN4
      PROGRAM MRKE,5
C.....1972
C.....MARGARET ROMAN
C.....PROGRAM MARK
C.....DUMMY DIMENSION
      DIMENSION IDUMM(475)
      DIMENSION NR(3),ITEX(9),NAMT(3),ITEX1(7),ITEX2(7)
      DIMENSION ITEX3(3),ITEX4(5),ITEX5(4),ITEX6(4)
      COMMON MOUT,LABF(3),LABR(3),NOSTY,INSTY(31),JBS(15),NAMST(3),
     1 MKUP(5),NSCMD,JBM(20),NAMMK(3),JBG(4),NAMGD(3),JBP(10),
     2 IDMM(5),IDDP,NPLC,NSHD,FACTR,NSIZ,ISIZ(42),WIDTH,EFF,SX,SY,KSTL,
     3 TABDS,NAP,KINT,KFLG,KINC,KSIZ,KXMN,KYMN,KANG,KXMX,KYMX,KDPT,
     4 KAD,KADP,KDF,CSMX,MKWD,CSMN,NMGE,IMGP,KREL,ITYPE,
     5 NOIMG,IDFSZ,IMGY,IMGE,IMGTC,IMKC1,IMKC2,IMENU,IDXP,IDYP,
     6 IXC,IYC,IAL,IXR,IYU,IYL,CS,MDELX,MDELY,JFLG,MLW,MUP,ISLX,ISLY,
     7 TOTAL,MYL,XPC,YPC,IMGLP,IMESG,IEDG,ISOFF,SOFF,KSLH,
     8 IMGPS,MAXOF,KSLV,KTC,IOPTN,IAREA,ISIZE,KCSR,KYES,KPSIZ,KPLL,
     9 IMGIN,ILPLC,IPKL,ISU(7),
     A IDFSFT,KPLP,IFLP,ICUST,IMGPK,
     B TLML,ISMDU,ISPDL,T3,T4,T5,T6,T7,T8,T9,ILOOP,INSCP,
     C JSTY,JFS,JNG,JSHD,JRT,
     D ILPB,IBU,IBU,IDV,IBR,IBL,ITCW,ITCCW,
     E IMCW,IPLCW,ISL,IRE,IFL,IPK,ICOM,
     F TOUCS(1)
```

```
      DATA ITEX/2HOV,2HEK,2HRI,2HDE,5*2H  /
      DATA ITEX1/2HCA,2HN ,2HNO,2HT ,2HLO,2HCK,2H@ /
      DATA ITEX2/2HDA,2HXI,2HMO,2HM ,2HTI,2HLT,2H@ /
      DATA ITEX3/2H N,2HAP,2H?@/,ITEX4/2H S,2HTR,2HIP,2HE?,2H@ /
      DATA ITEX5/2H P,2HLA,2HID,2H?@/,ITEX6/2H F,2HLI,2HP?,2H@ /
      DATA NAMT/2HCH,2HEK,2H  /
      DATA MK/2HMR,2HK ,2H  /
      JM15=JFLD(15,1)
      JM1 = JFLD(1,1)
      JB0 = JFLD(0,1)
C.....LABR(1) = 0 TELLS S/R BUMP IT WAS CALLED FROM MRKE
      LABR = 0
      KSLV=0
      KALP = KAD+IMGLP
      KAD1 = KADP+IMK01
      KAD2 = KADP+IMK02
      IF(IMGP.NE.0) IPUT=JLDF(JRT,IDUM(KFLG+IMGP))
C.....TURN OFF "BUSY" AND "OVERLAP"
      CALL LIGHT(2,0)
      CALL LIGHT(3,0)
C.....RETURN FROM MRKH COMPLETE, THEN RELEASE PICKED PIECE
      IF(IMGP.NE.9999) GO TO 40
      IMGP = 0
      IMGL = 0
      IDUM=0
   40 CALL INDSP(KDPT,IDUM,IDFSZ)
      CALL NSPAC(2,46)
      DIA = FALTR*4096.0
      JPCK =  1
      IRT = IRND(2,0,/CS)
      MYP = MXD/2-IYC
      MYN = MYP-MKWD
      CALL SCALE(2,0,DIM,DIM,1)
C.....READ THE TABLET
  200 CALL MKMIT(JFLK,NDXP,NDYP,IMG)
  205 IF(IMG.EQ.0) GO TO 300
      IF(IMG.LE.NDIMS) GO TO 210
      IF(IMG.EQ.32765.AND.ICPEC.EQ.0) GO TO 250
      GO TO 300
C.....PIECE PICKED
C.....DETERMINE SHADE ZONE
  210 IF((IMGP.NE.0.OR.IMGL.NE.0).AND.NSHD.GT.1) CALL SHADE(IMGL)
      IMGP = IMG
      IPUT = JLDF(JRT,IDUM(KFLG+IMGP))
C.....DISPLAY SIZE OF THE PIECE PICKED
  211 IMGS = (JLDF(JSM(9),IDUM(KSIZ+IMGP))-1)*3+IDFSZ+1
      CALL INSRT(3,IDUM(KPSIZ),IDUM(IMGS))
C.....SET ISMAL TO 99 IF A "SMALL" PIECE
      ISMAL=99
      MXSML=50
      IF(IABS(IDUM(KADP+IMGP+1))-IDUM(KADP+IMGP),GT,MXSML) ISMAL=0
      IDUM=0
C.....DISPLAY SHADE ZONE
      IF(NSHD.LE.1) GO TO 215
      ITMP = JLDF(JSHD,IDUM(KFLG+IMGP))
      IDUM(KPSIZ+4)=2H
      IF(ITMP.NE.4) IDUM(KPSIZ+4) = ITMP+5060B
C.....SET SHADE ZONE LIMITS
      MKU = MKWD/2
      MKU = MKU-MKWD
      IF(ITMP.EQ.0) GO TO 213
      INCSH = MKWD/NSHD
C.....MORE THAN 2 ZONE, USE ASSIGNED ZONE AND ADJACENT ZONES
      IF(NSHD.EQ.2) GO TO 212
      ISHDU = MIN0(MXU,MKU-(INCSH*(ITMP-2)))
```

```
      ISHOL = MAX0(MKD,MKU-(INCSH*(ITMP+1)))
      GO TO 215
C..........2 ZONES ONLY, DON'T ALLOW ADJACENT ZONES
  212 ISHOU = MKU-INCSH*(ITMP-1)
      ISHOL = ISHOU-INCSH
      GO TO 215
C..........USE ENTIRE MARKER AS THE ZONE
  213 ISHOU = MKU
      ISHOL = MKD
C.....CLEAR THE STRIPE LOCK FLAG
  215 KSLH = 0
      KSLV = 0
C.....COMPUTE HEADER OFFSET
      NN=6
      IF(JLOF(JRM(15),IOPTN),NE,0) NN=NN+3
      IF(JLOF(JRM(15),IOOM(KFLG+NPEC+IMGP)),NE,0) NN=NN+4
      CALL VISIB (IMGP,IORGP,IOXP,IOYP)
C.....COMPUTE DEL X AND DEL Y
      MDELX = IOOM(KXMX+IMGP)-IOOM(KXMN+IMGP)
      MDELY = IOOM(KYMX+IMGP)-IOOM(KYMN+IMGP)
C.....SET THE RELEASE FLAG IF PIECE IS IN THE MARKER
      KREL = 0
      IF(IOYP+MDELY.GT.MYP)GO TO 220
      KREL = 1
  220 JMOK=1
      GO TO 270
C..........TRACKING CROSS
C.....DO NOT MOVE IN "X" IF PLAID LOCK FLAG SET
  250 IF(KSLV.EQ.0) IOXP = NOXP
C.....DO NOT MOVE IN Y IF STRIPE LOCK FLAG SET
      IF (KSLH.EQ.0) IOYP = NOYP
  251 IF(IMGP.EQ.0) GO TO 300
C.....UPDATE RELEASE FLAG AND KEEP PIECE ON MARKER
      IF(KREL.EQ.1) GO TO 255
      IF(IOYP+MDELY.GT.MYP)GO TO 200
      KREL = 1
  255 IF(KSLH.EQ.1) GO TO 256
C..........CHECK Y LIMITS
      IF(NOYP.LT.MYN) IOYP=MYN
      IF(NOYP+MDELY.GT.MYP) IOYP=MYP-MDELY
C..........CHECK SHADE ZONE
      IF(NSHD.EQ.1) GO TO 2551
      IYCP = IOYP+MDELY/2+IYC
      IF(ISHOU.GE.IYCP.AND.ISHOL.LE.IYCP) GO TO 2550
C..........VIOLATION, TURN ON "SHADE" LIGHT
      CALL LIGHT(4,1)
      GO TO 2551
C..........TO SHADE ZONE TURN OFF "SHADE" LIGHT
 2550 CALL LIGHT(4,0)
 2551 IF(KSLV.EQ.1) GO TO 265
C..........CHECK X LIMITS
  256 IF(IORGP.EQ.2) GO TO 257
C.....ORIGIN AT LOWER LEFT
      OXLF=FLOAT(IXC)+FLOAT(NOXP)
      IF(OXLF.LT.-32000.0)IOXP = (-32600-IXC)
      IF(NOXP.LT.-LMT.OR.NOXP.GT.LMT) GO TO 280
      NOX=NOXP+MDELX
      IF(NOX.LT.LMT.AND.NOX.GT.-LMT)GO TO 280
      GO TO 265
C.....ORIGIN AT LOWER RIGHT
  257 OXLF=FLOAT(IXC)+FLOAT(NOXP-MDELX)
      IF(OXLF.LT.-32000.0)IOXP=-32600.0+MDELX-IXC
C.....NOW CHECK Y LIMITS
  260 IF(NOYP.GT.LMT) GO TO 280
      OXLF=FLOAT(IXC)+FLOAT(NOXP)
```

```
      IF(KREL.EQ.1) GO TO 254
C.....PIECE ABOVE MARKER
      IF(IORGP.EQ.2) GO TO 261
      IF(DXLF.LT.-32700.0)IOXP=-32700-IXC
      GO TO 254
  261 DXLF = DXLF+FLOAT(MDELX)
      IF(DXLF.LT.-32700.0)IOXP=-32700-IXC+MDELX
  254 IF(NOXP.LT.-LMT.OR.NOXP.GT.LMT) GO TO 280
C.....NOW ADJUST PIECE POSITION
  255 CALL MODI(1,IOXP,IOYP,IORGP,IMGP)
C.....CHECK OVERLAP IF IN "LOOP" FOR SMALL PIECE
  270 IF(ILOOP.NE.0) GO TO 418
      GO TO 300
  274 CALL MODI(2,IOXP,IOYP,IORGP,IMGP)
      IF(IORGP.NE.0) GO TO 270
C.....PIECE IS OFF SCREEN
      CALL LITE(0,KADP+IMGP)
      IMGP = 0
      N=-3
      IF(NSHO.GT.1) N=-5
      CALL INSRT(N,IOOM(KPSIZ),0)
      GO TO 270
C.....READ FUNCTION SWITCHES
  300 CALL FUNCB(JTYPE)
      IF(JTYPE.EQ.0) GO TO 200
      ITYPE = JTYPE
  303 IF(IORGP.NE.0) CALL ORGIN(IOXP,IOYP,0,IMGP)
      IF(ITYPE.EQ.IOV) GO TO 410
      IF(ITYPE.EQ.IHL.OR.ITYPE.EQ.IBO) GO TO 410
      IF(ITYPE.EQ.IHR.OR.ITYPE.EQ.IBO) GO TO 410
      IF(ITYPE.EQ.ITCW.OR.ITYPE.EQ.ITCCW) GO TO 339
      IF(ITYPE.EQ.IRCW.OR.ITYPE.EQ.IRCCW) GO TO 340
      IF(ITYPE.EQ.ISL) GO TO 480
      IF(ITYPE.EQ.KPLH) GO TO 500
      IF(ITYPE.EQ.IPE) GO TO 420
      IF(ITYPE.EQ.IFL) GO TO 430
      IF(ITYPE.EQ.ILPE) GO TO 440
      LY=9999
      IF(ITYPE.EQ.IPK) GO TO 306
      IF(ITYPE.EQ.IPKC) GO TO 320
      GO TO 200
C.....PICK
  306 JPCK = 0
      ICPEC = 0
C.....DETERMINE SHADE ZONE
  307 IF(IMGP.EQ.4.AND.IMGL.EQ.0) GO TO 200
C.....TURN OFF "SHADE" LIGHT
      CALL LIGHT(4,0)
      IF(NSHO.GT.1) CALL SHADE(IMGL)
      IF(IMGP.EQ.0) GO TO 200
      IF(LY.GT.MYP) IOYP = IRND(LY/CS)
      CALL ORGIN(IOXP,IOYP,0,IMGP)
      N=-3
      IF(NSHO.GT.1) N=-5
      CALL INSRT(N,IOOM(KPSIZ),0)
      IMGP=0
      IMGL=0
      GO TO 200
C....."PICK" (CHECK) WITHOUT MOVING,  SET ICPEC FLAG,
  320 ICPEC = 1
      JPCK = 0
      GO TO 307
C.....ROTATE
  340 IF(IMGP.EQ.0) GO TO 200
      IMGL = 450
```

```
      M=INCCW
      IF(JSLX*ISLY.NE.0.AND.JLDF(JBM(15),IDUM(KFLG+IMGP)).EQ.1)
     1 GO TO 200
C.....CHECK NAP CONSTRAINT
      IF(NAP.EQ.0) GO TO 341
      IF(JLDF(JNG,IDUM(KFLG+IMGP)),EQ.0) GO TO 341
      CALL INSRT(3,ITEX(5),ITEX3)
C.....DISPLAY "OVERRIDE NAP?", "YES OR NO"
      CALL MESAG (1,ITEX,ITMP)
      IF(ITMP.EQ.1) CALL ROTAL(IMGP)
      GO TO 200
C.....TILT
  339 IF(IMGP.EQ.0) GO TO 200
      INGL = 5
      M = ITCCW
  341 IF(ITYPE.EQ.M) INGL=-INGL
  342 IANG=KANG+IMGP
      IIXX = KADP+IMGP
      IF(IROT.EQ.0.OR.M.EQ.ITCCW) GO TO 343
      INGL = 1800
  343 NANG = MOD(IDUM(IANG),3600) + INGL
      MMM = 0
      IF(IABS(IDUM(IANG)).GE.3600) MMM=ISIGN(3600,NANG)
      NANG = MOD(NANG,3600)
C.....CHECK FOR MAXIMUM TILT
      MXTLT = JLDF(JBM(19),IDUM(KFLG+IMGP))*10
      IF(MXTLT.EQ.310) GO TO 345
      NXTLT = IABS(MOD(NANG+MMM,3600))
      IF(NXTLT.LE.MXTLT) GO TO 345
      IF(IABS(NXTLT-1800).LE.MXTLT) GO TO 345
      IF(IABS(NXTLT-3600).LE.MXTLT) GO TO 345
C.....EXCEEDS MAXIMUM TILT, TURN ON "MAX. TILT" LIGHT
      CALL LIGHT(5,1)
C.....DELAY WITH LIGHT ON
      DO 3444 I=1,20000
      ITMP = ITMP+1-1
 3444 CONTINUE
C.....TURN OFF "MAX. TILT" LIGHT
      CALL LIGHT(5,0)
      GO TO 200
  345 IDUM(IANG) = NANG+MMM
      CALL DRAWI(IDXP,IDYP,IORGP,IMGP,ISTAT)
      CALL ORGIN(IDXP,IDYP,IORGP,IMGP)
  352 IF(IABS(IDUM(IANG)).LT.3600) GO TO 353
      M = IABS(IDUM(IIXX))
      CALL FLIP(IDUM(M+NN))
      CALL ORGIN(IDXP,IDYP,IORGP+10,IMGP)
      IF(ISLY.EQ.0) GO TO 353
C.....ADJUST STRIPE LINES
      ITMP = KSTL+IMGP
      IF(IDUM(ITMP).EQ.9999) GO TO 353
      IDUM(ITMP) = -IDUM(ITMP)
  353 CALL MODI(2,IDXP,IDYP,IORGP,IMGP)
      CALL LITE(1,IIXX)
      GO TO 251
C.....CHECK OVERLAP OF BUMP UP (SEGMENT CHANGE)
  413 IF(IMGP.EQ.0.AND.IMGL.EQ.0) GO TO 200
      IF(IMGP.NE.0) IMGL=IMGP
      IF(IMGP.EQ.0) IMGP=IMGL
      IF(ITYPE.EQ.IOV) GO TO 412
      IF(ISLX*ISLY.NE.0.AND.JLDF(JBM(15),IDUM(KFLG+IMGP)).EQ.1)GO TO 412
C.....CHECK FOR STRIPE LOCK TO BUMP UP OR DOWN
      IF(ISLY.EQ.0) GO TO 411
      IF(ITYPE.EQ.ISR.OR.ITYPE.EQ.IBL) GO TO 412
      IF(IDUM(KSTL+IMGP).EQ.9999) GO TO 412
```

```
      IF(SY.EQ.99..AND.JLDF(JBM(13),IDUM(KFLG+NPEC+IMGP)).EQ.0)GO TO 412
      CALL INSRT(5,ITEX(5),ITEX4)
      GO TO 413
C.....CHECK FOR PLAID LOCK TO BUMP LEFT OR RIGHT
411   IF(ISLX.EQ.0.OR.ITYPE.EQ.IBD.OR.ITYPE.EQ.IBD) GO TO 412
      IF(IDUM(KPLL+IMGP).EQ.9999) GO TO 412
      CALL INSRT(4,ITEX(5),ITEX5)
C.....REQUEST OVERIDE
413   CALL MESAG(1,ITEX,ITMP)
      IF(ITMP.EQ.0) GO TO 200
  412 ICOST=0
C.....TURN ON "BUSY"
  413 CALL LIGHT(2,1)
      IF(ISSW(15).LT.0) GO TO 415
C.....NOTE,S/R BUMP REQ. TYPE = 15 FOR OVERLAP
  416 N=1
      IF(ILOOP.NE.0) N=-1
      CALL BUMP(N,ITMP,JTMP)
C.....LOOP IF "SMALL" PIECE AND OPTION SET AND OVERLAP FOUND
      IF(JTMP.EQ.0.OR.ISMAL.NE.99.OR.JLDF(JFLD(9,1),IOPTN).EQ.0.
     + OR.ICPEC.EQ.1.OR.ILOOP.GT.10) GO TO 414
      ILOOP=ILOOP+1
      INGL=10*ILOOP
      IF(MOD(ILOOP,2).EQ.0) INGL=-INGL
      GO TO 342
  414 JFLG=0
      IMGP=0
C.....TURN OFF "BUSY" AND "OVERLAP"
      CALL LIGHT(2,0)
      CALL LIGHT(3,0)
      GO TO 300
C.....OVERLAP AND BUMP WITH PRINT OUT
  415 MK(2) = 2HKG
      CALL EXEC (8,MK)
C.....CALL MRKG
      CALL MARK
C.....RELEASE PIECE FROM MARKER OUTLINE, STRIPE LINE OR PLAID LINE
  420 IF(IMGP.EQ.0) GO TO 200
      IF(KSLH+KSLV.EQ.0) GO TO 425
      KSLV = 0
      KSLH = 0
      GO TO 200
  425 KREL = 0
      IDYP = IDYP+2
      ITYPE = 0
      GO TO 265
C.....FLIP PIECE
  430 IF(IMGP.EQ.0) GO TO 200
      IF(ISLX*ISLY.NE.0.AND.JLDF(JBM(15),IDUM(KFLG+IMGP)).EQ.1)
     1 GO TO 200
C.....CHECK FLIP CONSTRAINT
      IF(IFLP.EQ.0) GO TO 432
      IF(JLDF(JFG,IDUM(KFLG+IMGP)).EQ.0) GO TO 432
      CALL INSRT(4,ITEX(5),ITEX6)
C.....DISPLAY "OVERRIDE FLIP?"  "YES OR NO"
      CALL MESAG (1,ITEX,ITMP)
      IF(ITMP.EQ.1) CALL FLPAL(IMGP)
      GO TO 200
C.....SET FLIP BIT IN "ICOST"
  432 CALL JSTF(1,JB1,ICOST)
      IAD = IDUM(KADF+IMGP)
      CALL FLIP(IDUM(IAD+NN))
      IF(ISLY.EQ.0) GO TO 435
C.....ADJUST STRIPE LINES
      ITMP = KSTL+IMGP
```

```
      IF(IDUM(ITMP),EQ,9999)GO TO 435
      IDUM(ITMP) = -IDUM(ITMP)
  435 NANG = IDUM(KANG+IMGP)
      MMM = NANG+1
      IF(IABS(NANG),GE,3600) MMM = -MMM
      IDUM(KANG+IMGP) = NANG+ISIGN(3600,MMM)
      CALL ORGIN(IDXP,IDYP,IORGP+10,IMGP)
C.....SET UP ADDRESS LOCATION FOR CALL TO LITE AT 353
      IIXX = KADP+IMGP
      GO TO 353
C.....LIGHT PEN BUTTONS (SPECIAL ON/OFF)
  440 MK(2) = 2MKJ
      IF(IMGP,NE,0) CALL MODI(2,IDXP,IDYP,IORGP,IMGP)
      IMGP=0
      IMGL=0
      CALL DDSC(0)
      CALL EXEC(8,MK)
      CALL MMKJ
C.....LOCK STRIPE LINE
  464 IF(IMGP,EQ,0) GO TO 200
C.....PIECE HAS STRIPE LINE?
      IF(IDUM(KSTL+IMGP),EQ,9999) GO TO 200
      IF(ISLY,EQ,0) GO TO 200
      IF(SY,EQ,99,0) GO TO 550
C.....LOCK ON STRIPE
      IF(KREL,NE,1) KSLH=0
      ISLP = (IDUM(KYMN+IMGP)+IDUM(KYMX+IMGP))/2+IDUM(KSTL+IMGP)
      ISL1 = MKWD/2
      ISL3 = MKWD/2-MKWD
      ISL2 = ISL1+ISOFF
      IF(ISLP,GE,ISL2,AND,ISLP,LE,ISL1) GO TO 465
      ISL1 = ISL2
  463 ISL2 = ISL1-ISLY
      IF(ISL2,LT,ISL3) ISL2 = ISL3
      IF(ISLP,GE,ISL2,AND,ISLP,LE,ISL1) GO TO 465
      IF(ISL2,EQ,ISL3) GO TO 200
      ISL1 = ISL2
      GO TO 463
  465 ID1 = ISL1-ISLP
      ID2 = ISL2-ISLP
      KSLH = 1
      IF(IABS(ID1),LT,IABS(ID2)) GO TO 467
      IDD = ID2
      GO TO 468
  467 IDD = ID1
  468 IF((IDUM(KYMX+IMGP)+IDD),GT,MKWD/2) IDD=IDD-ISLY
      IF((IDUM(KYMN+IMGP)+IDD),LT,ISL3) IDD=IDD+ISLY
      IDYP = IDYP+IDD
      GO TO 200
C.....LOCK ON PLAID
  500 IF(IMGP,EQ,0) GO TO 200
      IF(ISLX,EQ,0) GO TO 200
      IF(IDUM(KPLL+IMGP),EQ,9999) GO TO 200
      IF(KREL,NE,1) KSLV=0
      ISLP = IDUM(KYMN+IMGP)+MDELX/2+IDUM(KPLL+IMGP)
      ISL1=-32000
      ISL3=32500
  510 ISL2=ISL1+ISLX
C.....MUST USE FLOATING POINT DUE TO SIZE OF INTEGERS
      IF(FLOAT(ISL2),GT,FLOAT(ISL3)) GO TO 200
      IF(FLOAT(ISLP),LE,FLOAT(ISL2),AND,FLOAT(ISLP),GE,
     1  FLOAT(ISL1)) GO TO 520
      ISL1=ISL2
      GO TO 510
```

```
  520 ID1=ISL1-ISLP
      ID2=ISL2-ISLP
      NSLV = 1
      IF(IABS(ID1).LT.IABS(ID2)) GO TO 530
      IDD=ID2
      GO TO 540
  530 IDD=ID1
  540 IF(FLOAT(IDUM(KXMX+IMGP)+IDD).GT.32600.0) IDD=IDD-ISLX
      IF(FLOAT(IDUM(KXMN+IMGP)+IDD).LT.-32600.0) IDD=IDD+ISLX
      IDXP=IDXP+IDD
      GO TO 280
C.....LOCK BY STRIPE GROUP
  550 ISTGP = JLOF(JBM(13),IDUM(KFLG+NPEC+IMGP))
C.....STRIPE GROUP = 0, DON'T LOCK
      IF(ISTGP.EQ.0) GO TO 200
      IF(ISD(ISTGP).NE.9999) GO TO 560
C.....FIRST PIECE PLACED , DETERMINE STRIPE
      ISD(ISTGP) = (IDUM(KYMN+IMGP)+IDUM(KYMX+IMGP))/2+IDUM(KSTL+IMGP)
      KSLM = 1
      GO TO 280
C.....COMPUTE CLOSEST OF TWO STRIPES AND LOCK
  560 ISDP = (IDUM(KYMN+IMGP)+IDUM(KYMX+IMGP))/2+IDUM(KSTL+IMGP)
      ID1 = ISD(ISTGP) -ISDP
      ID2 =-ISD(ISTGP) -ISDP
      IFTT = 0
      IF(IABS(ID1).LT.IABS(ID2)) GO TO 565
  562 IDD = ID2
C.....CHECK FOR MARKER LIMITS
      IF(IFTT.GT.1) GO TO 580
      IFTT = IFTT+1
      IF((IDUM(KYMN+IMGP)+IDD).LT.-(MKWD-MKWD/2)) GO TO 565
      IF((IDUM(KYMX+IMGP)+IDD).GT.MKWD/2) GO TO 565
      GO TO 570
  565 IDD = ID1
C.....CHECK FOR MARKER LIMITS
      IF(IFTT.GT.1) GO TO 580
      IFTT = IFTT+1
      IF((IDUM(KYMN+IMGP)+IDD).LT.-(MKWD-MKWD/2)) GO TO 562
      IF((IDUM(KYMX+IMGP)+IDD).GT.MKWD/2) GO TO 562
C.....UPDATE Y POSITION AND LOCK
  570 IDYP = IDYP+IDD
      KSLM = 1
      GO TO 280
C.....IMPOSIBLE TO LOCK
  580 CALL MESAG(4,ITEX1,ITMP)
      GO TO 200
      END
      END$


FTN4
      PROGRAM MRKJ,5
C.....10/2/72
C.....MARGARET ROWAN
C.....PROGRAM MARK
C.....VERSION 2.0
      DIMENSION MK(3),ITEX(9),NAMT(3),ITEX1(6)
      DIMENSION ITEX2(6),ITEX3(5),ITEX4(4),ITEX5(5),ITEX6(6)
      COMMON MOUT,LABF(3),LABR(3),NOSTY,INSTY(31),JBS(15),NAMST(3),
     1 MKUP(5),NSCMD,JBM(25),NAMMK(3),JMG(4),NAMGD(3),JBP(10),
     2 IDMK(5),IDDP,NPEC,NSHD,FACTR,NSIZ,ISIZ(42),WIDTH,EFF,SX,SY,KSTL,
     3 YARDS,NAP,KINT,KFLG,KINL,KSIZ,KXMN,KYMN,KANG,KXMX,KYMX,KOPT,
     4 KAD,KADP,KOF,CSMX,MKWD,CSMN,NMGE,IMGP,KKEL,ITYPE,
     5 NOIMG,IOFSZ,IMGY,IMGE,IMGTC,IMKO1,IMKO2,IMENU,IDXP,IDYP,
     6 IXC,IYC,IXL,IXR,IYU,IYL,CS,MDELX,MDELY,JFLG,MLW,MUP,ISLX,ISLY,
```

```
     7 TOTAL,MYL,XPC,YPC,IMGLP,IMESG,IEDG,ISOFF,SOFF,KSLH,
     8 IMGPS,MAXOF,KSLV,KTC,IOPTN,IAREA,ISIZE,KCSR,KYES,KPSIZ,KPLL,
     9 IMGIN,ICPEC,IT3,IT4,IT5,IT6,IT7,IT8,IT9,IT10,
     A IOFSET,KPLH,IFLP,ICUST,IMGPK,
     B TEMP,T2,T3,T4,T5,T6,T7,T8,T9,ILOOP,INSCP,
     C JSTY,JFB,JNG,JSH0,JRT,
     D ILPR,IBU,IFD,IOV,IBR,IBL,ITCW,ITCCW,
     E IRCW,IRCCW,ISL,IRE,IFL,IPK,ICOM,
     F IDUM(1)
      DATA ITEX4/2HAR,2HOP,2HT?,2H@ /
      DATA ITEX5/2HF1,2HC1,2HSH,2HED,2H?@/
      DATA ITEX6/2HTE,2HMP,2HOR,2HAR,2HY?,2H@ /
      DATA ITEX3/2HOR,2HIG,2HIN,2HAL,2H?@/
      DATA ITEX2/2HLO,2HCK,2HED,2H M,2HAR,2HKE,2HR?,2H@ /
      DATA ITEX1/2HRE,2H-S,2HPR,2HEA,2HD?,2H@ /
      DATA NAMT/2HCH,2HEK,2H  /
      DATA MK/2HMR,2HK ,2H  /
      JB15 = JFLD(15,1)
      KALP = KA0+IMGLP
      KA01 = KA0P+IMK01
      KA02 = KA0P+IMK02
C.........DISPLAY THE SPECIAL BUTTONS
  442 CALL LITE(1,KALP)
      CALL LIGHT(1,1)
C.........LOOK FOR STYLUS PRESSURE OR A FB HIT ON "SPECIAL"
  440 CALL MAHIT(2,ITMP,JTMP,KTMP)
      IF(KTMP.NE.0) GO TO 445
      CALL PUNCH(ITMP)
      IF(ITMP.NE.ILP6) GO TO 440
      CALL LITE(0,KALP)
      CALL LIGHT(1,0)
      GO TO 200
  445 CALL LITE(0,KALP)
      CALL LIGHT(1,0)
      IF(KTMP.GT.8) GO TO 600
      GO TO (530,530,510,520,470,470,530,650),KTMP
C.........COMPUTE EFFICIENCY AND YARDAGE
  530 RGP = -32600.0
      RGPL = -32600.0
      TOTAL = 0.0
      MKOL = (0KA0/2)
      DO 532 I=1,IOTMG
C.........IGNOR SLAVE PIECE
      IF(JLDF(JRM(2),IDUM(KFLG+I)),EQ.1) GO TO 532
      MOPP = IDUM(KYSX+I)
      MLMP = IDUM(KYSN+I)
      IF(MOPP-1.GT.MKOL) GO TO 532
      DIM = IDUM(KXSX+I)
      TOTAL = TOTAL+FLOAT(IDUM(KINC+I))/10.0
      IF(MOPP.GT.0) RGPL = AMAX1(RGPL,DIM)
      IF(MLMP.LT.0) RGP = AMAX1(RGP,DIM)
  532 CONTINUE
C..... CHECK FOR A LOCKED MARKER
      DIM = AMAX1(RGP,RGPL)
      D=DIM
      JB1=JFLD(1,1)
      IFLG = 0
      IF(JLDF(JB1,IOPTN).EQ.0) GO TO 535
      CALL MESAG(1,ITEX2,IFLG)
      IF(IFLG.EQ.1) DIM = RGPL
  535 IF(IFLG.EQ.0) RGP = DIM
      CALL MESAG(0,2H  ,ITMP)
      YARDS = FACTR*((RGP+DIM)/2.0+32600.0)
```

```
C..........SAVE ACTUAL LENGTH
      TLML = FACTR*(U*32...0)
      TLML = TLML/36.0
      EFF = (TOTAL/(WIDTH*YARDS))*100.0
      YARDS = YARDS/36.0
      IF(EFF.LE...0.2) GO TO 200
      IF(EFF.GT.999.9) EFF=999.9
      IF(YARDS.GT.999.9) YARDS=999.9
      CALL CHGVL(KAD+IMGY,YARDS,IDUM,IDFSZ)
      I=YARDS
      X=(YARDS-FLOAT(I))*36.0
      CALL CHGVL(KAD+IMGIN,X,IDUM,IDFSZ)
      CALL CHGVL(KAD+IMGE,EFF,IDUM,IDFSZ)
      IF(KTMP.EQ.2) GO TO 540
      IF(KTMP.EQ.1) GO TO 442
C..........SAVE
      CALL MESAG(1,ITEX6,ITMP)
      IF(ITMP.EQ.0) GO TO 480
      GO TO 490
C..........PACK OR CHECK
  470 CALL UOSC(0)
      JFLG = KTMP
      IF(JFLG.EQ.5) GO TO 475
C..........SET PACK BIT IN "ICUST"
      CALL JSTF(1,JFLD(0,1),ICUST)
  475 MK(2) = 2HKH
       CALL EXEC(8,MK)
      CALL MRKM
C..........SAVE FINAL
  480 JFLG = 1
      CALL MESAG(0,2H *,ITMP)
      CALL MESAG (1,ITEX5,ITMP)
      IF(ITMP.EQ.0) GO TO 442
  485 MK(2) = 2HKF
      CALL UOSC(0)
      CALL EXEC (8,MK)
      CALL MRKF
C..........SAVE TEMP
  490 JFLG = 0
      GO TO 485
C..........RE SPREAD
  510 CALL MESAG(1,ITEX1,ITMP)
      IF(ITMP.EQ.0) GO TO 442
  515 N = (MAXDF+ICOM)/128+1
      M=N
C.........DISPLAY DUMMY TO ENABLE USER TO "GET OFF" YES OR NO
      CALL MESAG(0,2H *,ITMP)
C.....CHECK FOR ORIGINAL OR TEMPORARILY SAVED MARKER
      CALL MESAG(1,ITEX3,ITMP)
      IF(ITMP.EQ.0) N = M
      CALL UOSC(0)
      CALL STOP
      NWDS = MAXDF+ICOM
      CALL EXEC(14,66,EFF,NWDS,NAMT,N)
C.....CHECK FOR VALID DATA - ASK AGAIN IF INVALID
      IF(NAMT.EQ.0) CALL EXEC (14,66,EFF,NWDS,NAMT,N)
      CALL START(46,IDUM(KAD+1),IDUM)
      GO TO 200
C..........MATERIAL ON/OFF
  520 IF(ISLX.EQ.0.AND.ISLY.EQ.0) GO TO 200
      IF(SY.EQ.99.0.AND.ISLX.EQ.0) GO TO 200
      IAD=JLOF(JB15,IDUM(KA02))
      CALL LITE(IAD,KA02)
      GO TO 200
```

```
C.........LENGTH
  544 IEDG=(XGP+DIR)/2.
      YEDG = (IEDG-IXC)*CS
      IAD = IDUM(KAD1) + 16
      CALL LITE(0,KAD1)
      IF(IEDG.LT.IXL.OR.IEDG.GT.IXR) GO TO 545
      IDUM(IAD) = 1HDH4H
      IDUM(IAD+1) = YEDG*1E+9
      IDUM(IAD+2) = NUP*1E+16
      IDUM(IAD+3) = NLR*1E+14
  543 CALL LITE(1,KAD1)
      GO TO 442
  545 DO 550   I=1,4
      K=IAD+I-1
  550 IDUM(K)=0
      GO TO 543
C.........ROLL OR ZOOM SET ITYPE FOR MRKD
  600 MK(2)=2HKD
      LABF=KSLR
      CALL LITE(1,KALP)
      CALL DUSC(3)
      ITYPE=KTOP
      CALL DUSC(4)
      CALL EXEC (8,MK)
      CALL MRKD
C.........ABORT
  650 CALL MESAG (1,ITEX4,ITMP)
      IF(ITMP.EQ.0) GO TO 200
      CALL STOP
      CALL DUSC(4)
      CALL EXEC (15,66,0,1,NAMT,0)
      CALL RTN
  200 MK(2)=2HKE
      JFLG = 0
      CALL DUSC(4)
      CALL EXEC(8,MK)
      CALL MRKE
      CALL MARK
      END
      ENDS


FTN4
      SUBROUTINE NOTCH(X2,Y2,X1,Y1)
      TK=.03
      D=SQRT(((X2-X1)*(X2-X1))+((Y2-Y1)*(Y2-Y1)))
      CO=(X2-X1)/D
      SI=(Y2-Y1)/D
      IF(D.LT..187)D=.187
      X=X1-CO*TK+SI*TK
      Y=Y1-SI*TK-CO*TK
      CALL PLOT(X,Y,2)
      X=X1+CO*D+SI*TK
      Y=Y1+SI*D-CO*TK
      CALL PLOT(X,Y,2)
      X=X1+CO*D-SI*TK
      Y=Y1+SI*D+CO*TK
      CALL PLOT(X,Y,2)
      X=X1-CO*TK-SI*TK
      Y=Y1-SI*TK+CO*TK
      CALL PLOT(X,Y,2)
      CALL PLOT(X1,Y1,2)
      END
      ENDS
```

```
FTN4
      SUBROUTINE PLOT(X,Y,N)
      COMMON LOUT,ITYPE,ORIGX,ORIGY,LDIRC,XPEND,AXOLD,AYOLD,
     1   IPNF(3),IPUR(3),XMAX,YMAX,SCAL,XLAST,YLAST,LASTP,LASTA,
     2   LASTT,IPLT(3),JTYPE,JOB,JPOS,JTOOL,JHOME,JIZ,SPACE,IOPTN
     3   ,NPLAT,JR0,JR1,JR2,JR3,JR4
     4     ,AYCNST
      DATA PMAX /32,0/
      IF (ISSW(6).LT.0) WRITE (1,1000) X,Y,N
 1000 FORMAT("PLOT",2F6.2,I4)
      IZ=0
      IF(LASTT.NE.3)GO TO 1
C.....CUTTER TOOL. COMPUTE THE BLADE ANGLE.
      CALL BANG(X-XLAST,Y-YLAST,IANG)
      IZ=LASTA-IANG
      LASTA=IANG
    1 CONTINUE
      XMOVE=X-XLAST
      YMOVE=Y-YLAST
      IF (ITYPE.EQ.2HMA) XMOVE=XMOVE*SCAL
      IF (ITYPE.EQ.2HMA) YMOVE=YMOVE*SCAL
C.....CALCULATE THE NUMBER OF MOVES REQUIRED.
      BIG=ABS(XMOVE)
      IF(ABS(YMOVE).GT.BIG) BIG=ABS(YMOVE)
      NM=BIG/PMAX+1
    2 XINC=XMOVE/NM
      YINC=YMOVE/NM
      IX=XINC*1000.
      IY=YINC*1000.
      IF (ITYPE.EQ.2HMA) GO TO 10
      XLAST=XLAST+FLOAT(IX)*.001
      YLAST=YLAST+FLOAT(IY)*.001
      GO TO 20
   10 CONTINUE
      S=1000.0*SCAL
      XLAST=XLAST+FLOAT(IX)/S
      YLAST=YLAST+FLOAT(IY)/S
   20 CONTINUE
C.....INITIALIZE TOOL AS PEN AND ANGLE TO 0
      ISTAT=200000B
      IF (IZ.NE.0) CALL JSTF (IZ,JIZ,ISTAT)
      IF (LASTT.NE.1) ISTAT=ISTAT+400000B
      IZ=0
      IF(N.EQ.2) ISTAT=ISTAT+100000B
C.....HOME THE CUTTER IF NECESSARY
      IF(N.EQ.-3.AND.ITYPE.EQ.2HCU)CALL JSTF(1,JHOME,ISTAT)
      CALL STORE(1,ISTAT,IX,IY)
      NM=NM-1
      IF(NM.LE.0)GO TO 5
      XMOVE=XMOVE-XINC
      YMOVE=YMOVE-YINC
      GO TO 2
    5 IF(N.GT.0) RETURN
C.....INITIALIZE POSITION AND BLADE ANGLE
      XLAST=0.
      YLAST=0.
      LASTA=0.
C.....SET THE NEW JOB FLAG WHEN N=-99
      IF(N.EQ.-99) CALL STORE(2,0,0,0)
      RETURN
      END
      ENDS
```

```
FTN4
      SUBROUTINE RULEN (IRULE,X,Y)
C
C     VERSION 2.0
C
C     RONALD GORDON
C
C     PROGRAM TO DRAW RULE NUMBERS ON NESTED PLOTS WHEN
C     SAMPLE SIZE IS DRAWN.
C
      INTEGER ASCII(2)
      DATA SYMSIZ /0.15/
C
C     LEFT JUSTIFY NUMBER
C
      IF (IRULE.LT.1) RETURN
      JRULE=IRULE
      NDIGIT=3
  100 IF (JRULE.GE.100) GO TO 200
      JRULE=JRULE*10
      NDIGIT=NDIGIT-1
      GO TO 100
C
C     CONVERT TO ASCII AND PLOT
C
  200 CALL CODE
      WRITE (ASCII,300) JRULE
  300 FORMAT(I3)
      CALL SYMB (X,Y,SYMSIZ,ASCII,0.0,NDIGIT)
      CALL PLOT (X,Y,3)
      RETURN
      END
ENDS

ASMB
      NAM SCALE,7
      ENT SCALE
      EXT .ENTR
      EXT FLOAT
      EXT JRND
*
*
* SUBROUTINE SCALE(IX,IY,X,Y,N)
*
* N=1 SET UP SCALE FACTORS FOR DISPLAY
* N=2 SCALE FROM REAL UNITS TO RASTER UNITS
* N=3 SCALE FROM RASTER UNITS TO REAL UNITS
*
* REWRITTEN BY JEANIE LEMONS 03 FEB 73
*
*
IX    DEF *,I
IY    DEF *,I
X     DEF *,I
Y     DEF *,I
N     DEF *,I
*
* ENTRY TO SCALE
*
SCALE JMP *
      JSB .ENTR
      DEF IX
      LDA HERE      ***  GO TO(ONE,TWO,THREE)N
      ADA N,I
      JMP 0,I
```

```
HERE  DEF *
      JMP ONE
      JMP TWO
      JMP THREE
FFTN  DEC .0002442  ***  FFTN IS RSTR/15.0,  RSTR=0.3663
FONE  DEC 1.0
*
* SET UP SCALE FACTORS FOR DISPLAY
*@
ONE   DLD X,I       ***  SCALX=(X/15.0)*RSTR
      FMP FFTN
      DST SCALX
      DLD Y,I       ***  SCALY=(Y/15.0)*RSTR
      FMP FFTN
      DST SCALY
      DLD FONE      ***  SET UP INVERSE OF SCALX AND SCALY TO AVOID
      FDV SCALX     ***  USING FDV BELOW.
      DST SCLXI
      DLD FONE
      FDV SCALY
      DST SCLYI
      LDA IX,I      ***  IOFX=IX
      STA IOFX
      LDA IY,I      ***  IOFY=IY
      STA IOFY
      JMP SCALE,I   ***  RETURN
*
* SCALE FROM REAL UNITS TO RASTER UNITS
*
TWO   DLD X,I       ***  IX=IRND(X/SCALX)+IOFX
      FMP SCLXI
      JSB JRND
      ADA IOFX
      STA IX,I
      DLD Y,I       ***  IY=IRND(Y/SCALY)+IOFY
      FMP SCLYI
      JSB JRND
      ADA IOFY
      STA IY,I
      JMP SCALE,I   ***  RETURN
*
*
SCALX DEC 0.0
SCALY DEC 0.0
SCLXI DEC 0.0
SCLYI DEC 0.0
IOFX  DEF 0
IOFY  DEF 0
*
* RASTER TO REAL CONVERSION
*
THREE LDA IOFX      ***  X=FLOAT(IX-IOFX)*SCALX
      CMA,INA
      ADA IX,I
      JSB FLOAT
      FMP SCALX
      DST X,I
      LDA IOFY      ***  Y=FLOAT(IY-IOFY)*SCALY
      CMA,INA
      ADA IY,I
      JSB FLOAT
      FMP SCALY
      DST Y,I
      JMP SCALE,I   ***  RETURN
      END
      ENDSS
```

```
 FTN4
 SUBROUTINE TOOL(N)
 COMMON LOUT,ITYPE,ORIGX,ORIGY,LDIRC,XPEND,AXOLD,AYOLD,
1    IPNF(3),IPNP(3),XMAX,YMAX,SCAL,XLAST,YLAST,LASTP,LASTA,
2    LASTT,IPLT(5),JTYPE,JOB,JPOS,JTOOL,JHOME,JIZ,SPACE,IOPTN
3    ,NPLAT,JH0,JH1,JH2,JH3,JH4
 IF(N.EQ.LASTT) RETURN
 X=2.0
 IF(ISSW(11).LT.0)X=0.
 IF(N.EQ.1) X=-X
 XLAST=XLAST+X
 LASTT=N
 RETURN
 END
 ENDS


FTN4,M
      SUBROUTINE TRANS(IBUF,IDUM)
      DIMENSION IBUF(1),IDUM(1)
      DO 10   I=1,128
   10 IBUF(I)=IDUM(I)
      IF(ISSW(6).LT.0)WRITE(6,1000)(IBUF(K),K=1,128)
 1000 FORMAT(8O6)
      RETURN
      END
      ENDS

ASMB,R
      NAM IRND,6
      ENT IRND
      EXT JRND,.ENTR
*****
*****
ARG   BSS 1
IRND  NOP
      JSB .ENTR
      DEF ARG
      DLD ARG,I
      JSB JRND
      JMP IRND,I
*****
*****
      END

ASMB,R
      NAM JRND,6
      ENT JRND
      EXT IFIX,.FAD,.FSB
*****
*****
JRND  NOP
      SSA
      JMP **4
      JSB .FAD
      DEF HALF
      JMP **3
      JSB .FSB
      DEF HALF
      JSB IFIX
      JMP JRND,I
*****
*****
HALF  DEC 0.5
*****
*****
      END
```

```
ASMB
       NAM ACCES,7
       ENT ACCES
       EXT .ENTR,EXEC,.STOP
IOPER  NOP             OPERATION ADDRESS, 0=STORE, 1=COMPARE
ICODE  NOP             ACCESS CODE ADDRESS
ACCES  NOP
       JSB .ENTR
       DEF IOPER
*
       LDA ICODE,I     LOAD ACCESS CODE
       LDB IOPER,I     LOAD OPERATION
       SZB             STORE?
       JMP COMPR       NO, COMPARE
       LDB ADDR        LOAD B=STORE ADDRESS
       JSB EXEC        EXEC BASE PAGE STORE
       DEF *+2         RETURN ADDRESS
       DEF RC19        REQUEST CODE ADDRESS
       JMP ACCES,I     RETURN
*
COMPR  CPA ADDR,I      ACCESS CODES MATCH?
       JMP ACCES,I     YES, RETURN
       CLA,INA         NO A = 1
       JSB .STOP       STOP THE WORKS
*
RC19   OCT -19         BASE PAGE STORE REQUEST CODE
ADDR   OCT 51B         ACCESS CODE BASE PAGE STORAGE ADDRESS
*
       END

ASMB,L
       NAM ALLOC,7
       ENT ALLOC
       EXT .ENTR
*
*      ROUTINE USED TO ALLOCATE THE BLOCK OF STORAGE
*      WHICH IS LEFT OVER AFTER LOADING
*
*      CALLING SEQUENCE
*                JSB ALLOC
*                DEF *+3
*                DEF BUFAD
*                DEF BUFSZ
*
*      SIZE RETURNED WILL BE MULTIPLE OF 128.
*
BUFAD  NOP
BUFSZ  NOP
ALLOC  NOP
       JSB .ENTR
       DEF BUFAD
       LDA EXPG+6
       INA
       STA BUFAD,I
       CMA,INA
       ADA UMLKA
       AND MASK
       STA BUFSZ,I
       JMP ALLOC,I
UMLKA  EQU 1777B
EXPG   EQU 1417B
MASK   OCT 177600
*                                   WRITTEN BY RON GORDON
       END
END$
```

```
FTN4
      SUBROUTINE ASTAR(IX,IY,IDF,IDSPB)
C.....WHEN IDF IS GREATER THAN 1, SET UP INSTRUCTIONS FOR
C.....DRAWING AN ASTERISK.  WHEN IDF IS ZERO, UPDATE LOCATION
C.....OF ASTERISK.  WHEN IDF IS 1, TURN OFF THE ASTERISK.
      DIMENSION IDAT(10),IDSPB(5000)
      DATA IDAT/40022B,0,77760B,0,1,40010B,177560B,177121B,10177B,
     1  25024B/
      IF(IDF.LE.1)GO TO 40
      JDF = IDF
      DO 20 I=1,10
   20 IDSPB(JDF+1) = IDAT(I)
   40 IDSPB(4) = -JDF-1
      IDSPB(JDF) = 100000B
      IF(JDF.EQ.0)GO TO 50
      RETURN
   50 IDSPB(JDF+4) = IX*16
      IDSPB(JDF+5) = IY*16 +1
      IDSPB(4) = JDF + 1
      IDSPB(JDF) = 0
      RETURN
      END
      END$


FTN4
      SUBROUTINE BANG(X,Y,IANG)
      DATA RAD/57.2957795131/
      DATA PIP2/1.570796/
      DATA PIP3/4.71236/
      DATA PIP4/6.28318/
      IF(X)2,3,1
    1 IF(Y)10,5,11
    2 IF(Y)13,6,12
    3 IF(Y)8,5,7
    5 ANGLE=0.0
      GO TO 15
    6 ANGLE=180.0
      GO TO 15
    7 ANGLE=90.0
      GO TO 15
    8 ANGLE=270.0
      GO TO 15
   10 W=Y/X
      ANGLE=(PIP4+ATAN(W))*RAD
      GO TO 15
   11 W=Y/X
      ANGLE=ATAN(W)*RAD
      GO TO 15
   12 W=X/Y
      ANGLE=(PIP2-ATAN(W))*RAD
      GO TO 15
   13 W=X/Y
      ANGLE=(PIP3-ATAN(W))*RAD
   15 FANG=ANGLE
      IF(ANGLE.GT.90.0.AND.ANGLE.LE.270.)FANG=ANGLE-180.
      IF(ANGLE.GT.270.0.AND.ANGLE.LE.360.)FANG=ANGLE-360.
      ROUND=.5
      IF(FANG.LT.0)ROUND=-ROUND
      IANG=(FANG/1.8)+ROUND
      RETURN
      END
      END$
```

```
FTN4
      SUBROUTINE REGI
C.....DEC. 7, 1972
C.....MARGARET ROWAN
C.....PROGRAM: MARK
C.....VERSION 2.0
      DIMENSION JDUM(5),ITEX1(5),ITEX2(7),ITEX3(4),ITEX4(7),
     1 ITEX5(5),ITEX6(14),ITEX8(11),ITLP(54),ITMG(20),ISTDF(10)
      DIMENSION ITEXD(6),NAMRL(3)
C.....ROUTINE TO SET UP THE FIRST FOUR IMAGES IN THE DISPLAY
C          FILE.  THESE FORM THE MENU OR HEADING INFORMATION.
C
      COMMON MOUT,LABF(3),LABR(3),NOSTY,INSTY(31),JBS(15),NAMST(3),
     1 MKUP(5),NSCMD,JHM(20),NAMMK(3),JBG(4),NAMGD(3),JBP(10),
     2 IDMK(5),IDDP,NPEC,NSHD,FACTR,NSIZ,ISIZ(42),WIDTH,EFF,SX,SY,KSTL,
     3 YARDS,NAP,KINT,KFLG,KINC,KSIZ,KXMN,KYMN,KANG,KXMX,KYMX,KOPT,
     4 KAD,KADP,KDF,CSMX,MKWD,CSMN,NRGE,IMGP,KREL,ITYPE,
     5 NUIMG,IDFSZ,IMGY,IMGE,IMGTC,IMKO1,IMKO2,IMENU,IDXP,IDYP,
     6 IXC,IYC,IXL,IXR,IYU,IYL,CS,MDELX,MDELY,JFLG,MLW,MUP,ISLX,ISLY,
     7 TOTAL,NYL,XPC,YPC,IMGLP,IMESG,IEDG,ISOFF,SOFF,KSLH,
     8 IMGPS,MAXDF,KSLV,KTC,IOPTN,IAREA,ISIZE,KCSR,KYES,KPSIZ,KPLL,
     9 IMGIN,TOPEC,IT3,IT4,IT5,IT6,IT7,IT8,IT9,IT10,
     A IDFSET,KPLH,IFLF,ICUST,IMGPK,
     B T1,T2,T3,T4,T5,T6,T7,T8,T9,T10,
     C JSTY,JFG,JNG,JSHD,JRT,
     D ILMS,IMU,IMD,IDV,IDR,IHL,ITCW,ITCCW,
     E IHCW,IHCCW,ISL,IRE,IFL,IPK,ICOM,
     F IDUM(1)
      DATA ICM/64/,IX1/-1950/,IY1/1740/,IY2/1670/,IY3/-1950/,
     1    IY4/1530/,IY5/1460/
      DATA ITEX1/2HMA,2HRK,2HER,2H@ ,2H  /
      DATA ITEX2/2HST,2HYL,2HE(,2HS),3*2H  /
      DATA ITEX3/2HWI,2HDT,2HH ,2H= /
      DATA ITEX4/2HLE,2HNG,2HTH,2H= ,2H  ,2H /,2H@ /
      DATA ITEX5/2HSI,2HZE,2H(S,2H) ,2H@ /
      DATA ITEX8/2HEF,2HFI,2HCI,2HEN,2HCY,2H =,3*2H  ,
     1     2H %,2H@ /
      DATA ITEX6/2HSI,2HZE,2H P,2HIC,2HKE,2HD ,2H=@,2H S,2HHA,2HDE,
     1 2H Z,2HON,2HE ,2H=@/
      DATA ITEXD/3*2H  ,2H@ ,2H  ,2H@ /
      DATA ISTDF/400021B,733008,0,777600,0,1,400058,
     1 140201B,400140,777013/
      DATA ITLP/2HEF,2HFI,2HCI,2HEN,2HCY,2*2H  ,
     1          2HLE,2HNG,2HTH,2*2H  ,
     2          2HRE,2H-S,2HPR,2HEA,2HD ,2*2H  ,
     3          2HMA,2HTE,2HRI,2HAL,2*2H  ,
     4          2HPA,2HCK,2*2H  ,
     5          2HCH,2HEC,2HK ,2*2H  ,
     6          2HSA,2HVE,2*2H  ,
     7          2HAR,2HOR,2HT ,2*2H  ,
     8          136400B,2*2H  ,
     9          137400B,2*2H  ,
     A          13400B,2*2H  ,
     B          12400B,2H@ /
      DATA ITMG/19*2H  ,2H @/
      JFLD(2,1)=JFLD(2,1)
      IDUM(KAD+IMENU)=KDF
      AREA=0.
      YARDS=0.
      NSKIP=3
      IF(NOSTY.NE.1) NSKIP = 2
      KOPT=KDF
C          SET PS, NAME, CSR, X, Y, MCP AND INTENSITY
      DO 5 I=1,12
      IDUM(KOPT) = ISTDF(I)
```

```
5       KOPT = KOPT+1
C        DISPLAY "MARKER"
        CALL DISCH(IX1,IY1,1,0,ITEX1,IDUM,KOPT,IDFSZ)
        IDUM(KOPT-1) = 2H
C       DISPLAY MARKER I.D., ELIMINATING LEADING BLANKS
        DO 30 I=1,5
        IF(IDMR(I).EQ.2H  ) GO TO 30
        IDUM(KOPT) = IDMR(I)
        KOPT = KOPT + 1
30      CONTINUE
C       DISPLAY "STYLE(S)"
        DO 100 I=1,NSKIP
        IDUM(KOPT) = 2H
100     KOPT = KOPT + 1
        DO 35 I=1,4
        IDUM(KOPT) = ITEX2(I)
35      KOPT = KOPT + 1
C       DISPLAY STYLE I.D.(S), ELIMINATING LEADING BLANKS
        DO 1M I=1, NOSTY
        IDUM(KOPT) = 2H
        KOPT = KOPT+1
        NSTY = JLOF(JSTY,INSTY(I))
        CALL DIREC (14030B,ITEX1,NSTY,ITMP,ISTAT)
        DO 44 J=1,5
        IF (ITEX1(J).EQ.2H  ) GO TO 40
        IDUM (KOPT) = ITEX1(J)
        KOPT = KOPT + 1
40      CONTINUE
10      CONTINUE
        DO 120 I=1,NSKIP
        IDUM(KOPT) = 2H
120     KOPT = KOPT + 1
C       DISPLAY "WIDTH ="
        DO 70 I=1,4
        IDUM(KOPT) = ITEX3(I)
70      KOPT = KOPT + 1
C       DISPLAY WIDTH VALUE
        CALL DISVL(IX1,IY1,WIDTH,2,3,-1,0,IDUM,KOPT,IDFSZ)
         IX2=650
C       DISPLAY "LENGTH ="
        CALL DISCH (IX2,IY1,1,0,ITEX4,IDUM,KOPT,IDFSZ)
        IXYD=IX2 + ICH*7
C       DISPLAY "EFFICIENCY= %"
        IX3=IX2+ICH*20
        CALL DISCH(IX3,IY1,1,0,ITEX8,IDUM,KOPT,IDFSZ)
        IXEF = IX3 + ICH*12
C       DISPLAY "SIZE(S)"
        CALL DISCH(IX1,IY2,1,0,ITEX5,IDUM,KOPT,IDFSZ)
        KOPT=KOPT-1
        DO 00 I=1,NSIZ
        IDUM(KOPT) = 2H
        KOPT = KOPT + 1
        IF (I.NE.14) GO TO 25
C       ADD A CR, LF
        IDUM(KOPT)=64000B
        KOPT = KOPT + 1
        IY4 = IY4-70
        IY5=IY5-70
25      INDX=IDFSZ+(I-1)*3
        DO 50 J=1,3
        IF(IDUM(INDX+J).EQ.2H  ) GO TO 50
        IDUM(KOPT) = IDUM(INDX+J)
        KOPT = KOPT + 1
50      CONTINUE
        IF(JLOF(J00,IOPTN).EQ.0) GO TO 60
```

```
C.....ADD "=ALPHA"
      IDUM(KOPT)=2H=A+I-1
      KOPT=KOPT+1
60    IDUM(KOPT) = 12[illegible]
      KOPT = KOPT+1
C.....DISPLAY "SIZE OF PICKED PIECE" AND "SHADE ZONE"
      IF(NSHD.GE.1) ITEX6(7) = 364155
      CALL DISCH(IX1,IY3,1,0,ITEX6,IDUM,KOPT,IDFSZ)
      IXYS = IX1+ICH*22
      IDUM(KAD+IMGY) = KOPT
C     DISPLAY LENGTH IN YARDS AND INCHES (YD/IN)
      IYARD = YARDS
      Y = FLOAT(IYARD)
      X = (YARDS-Y)*36.0
      CALL DISVL (IXYD,IY1,Y,3,-1,1,0,IDUM,KOPT,IDFSZ)
      INC=ICH*5
      IDUM(KAD+IMGIN)=KOPT
      CALL DISVL (IXYD+INC,IY1,36.00,3,2,1,0,IDUM,KOPT,IDFSZ)
      CALL CHGVL (KAD+IMGIN,X,IDUM,IDFSZ)
      IDUM(KAD+IMGE) = KOPT
      CALL DISVL (IXEF,IY1,0.0,4,2,1,IMGE,IDUM,KOPT,IDFSZ)
C.....LEAVE ROOM FOR PICKED PIECE SIZE AND PIECE SHADE ZONE
      IDUM(KAD+IMGPS) = KOPT
      IF(NSHD.GE.1)ITEXD(4)=50108
      CALL DISCH(IXYS,IY3,1,0,ITEXD,IDUM,KOPT,IDFSZ)
      KPSIZ = KOPT-4
      IF(NSHD.GE.1)KPSIZ=KPSIZ-2
C          DISPLAY LIGHT PEN BUTTONS
      IDUM(KAD+IMGLP) = KOPT
      CALL DISCH(IX1,IY4,1,0,ITLP,IDUM,KOPT,IDFSZ)
      CALL LITE(0,KAD+IMGLP)
C           DISPLAY AREA FOR MESSAGES
      IDUM(KAD+IMESG)=KOPT
      CALL DISCH (IX1,IY5,1,7,ITMG,IDUM,KOPT,IDFSZ)
C          LEAVE ROOM FOR "YES" AND "NO"
      KOPT = KOPT-1
      DO 140 I=1,NSKIP
      IDUM(KOPT) = 2H
140   KOPT = KOPT + 1
      KYES = KOPT
      DO 145 I=1,6
      IDUM(KOPT) = 2H
145   KOPT = KOPT + 1
      IDUM(KOPT-1) = 2H24B
      CALL LITE(2,KAD+IMESG)
      RETURN
      END
      ENDS

FTN4
      SUBROUTINE BUMP (IND,IP1,IP2)
C.....BUMP PIECES AND CHECK FOR OVERLAP
      DIMENSION MNX(4),LMX(2,30),LMY(2,30),INTPC(30),
     1 IX1(201),IY1(201),IX2(201),IY2(201),IX(4),IY(4)
C********************** DUMMY ARRAY FOR LOADER ***************
C
      DIMENSION ITEX1(6),ITEX2(12),ITEX3(8)
      COMMON MOUT,LABF(3),LABR(3),NOSTY,INSTY(31),JBS(15),NAMST(3),
     1 MKUP(5),NSCMD,JBM(27),NAMMK(3),JBG(4),NAMGD(3),JBP(10),
     2 IDMK(5),IOOP,NPEC,NSHD,FACTR,NSIZ,ISIZ(42),WIDTH,EFF,SX,SY,KSTL,
     3 YARDS,NAP,KINT,KFLG,KINC,KSIZ,KXMN,KYMN,KANG,KXMX,KYMX,KOPT,
     4 KAD,KADF,KOF,LSMX,MKWD,LSMN,NKGE,IMGP,KKEL,ITYPE,
     5 NOIMG,IDFSZ,IMGY,IMGE,IMGTC,IMKG1,IMKG2,IMENU,IOXP,IOYP,
     6 IXC,IYC,IXL,IAR,IYU,IYL,CS,MDELX,MDELY,JFLG,MLW,MUP,ISLX,ISLY,
     7 TOTAL,MYL,XPC,YPC,IMGLP,IMESG,IEDG,ISOFF,SOFF,KSLH,
```

```
     8 IMGPS,MAXDF,KSLV,KTC,IOPTN,IAREA,ISIZE,KCSR,KYES,KPSIZ,KPLL,
     9 IT1,IT2,IT3,IT4,IT5,IT6,IT7,IT8,IT9,IT10,
     A IOFSET,KPLH,IFLP,ICUST,IMGPK,
     B T1,T2,T3,T4,T5,T6,T7,T8,T9,T10,
     C JSTY,JFG,JMG,JSHD,JRT,
     D ILPB,IRD,IMD,IOV,IRR,IBL,ITCW,ITCCW,
     E IRCW,IRCCW,ISL,IRE,IFL,IPK,ICUM,
     F IDUM(1)
      DATA IIEX1/2H N,2HU ,2HOV,2HER,2HLA,2HP@/
      DATA IIEX2/2HTO,2HO ,2HMA,2HNY,2H P,2HIE,2HCE,2HS ,2HIN,2H A,
     1  2HRE,2HAS/
      DATA IIEX3/2HPI,2HEC,2HE ,2HTO,2HO ,2HLA,2HRG,2HE@/
      DATA LMT/201/
      JCTR = 0
      IF(IMGP.EQ.0) RETURN
C.....SEND MASK TO READ "PICK" FOR ABORTING OPERATION
      CALL PSHIT(MMMM,1HW00Y0B)
      MMMM=IND
      IND=IABS(IND)
      JXMN = KXMN+IMGP
      JXMX = KXMX+IMGP
      JYMN = KYMN+IMGP
      JYMX = KYMX+IMGP
      MXP = IEDG
      MXN = -320.0
      IF(IDUM(JXMN).GT.IEDG) MXN = IEDG
      DMXN = FLOAT(MXN)
      DMXP = FLOAT(MXP)
      MYP = MKWD/2
      MYN=-(MKWD-MYP)
      IF(ITYPE.NE.IOV) GO TO 1
      IXP = 0
      IXN = 0
      IYP = 0
      IYN = 0
    1 MNX1 = IDUM(JXMN)
      MXX1 = IDUM(JXMX)
      MNY1 = IDUM(JYMN)
      MXY1 = IDUM(JYMX)
      JCTR = JCTR+1
      IF(JCTR.GT.10) GO TO 920
      MDLX = MXX1-MNX1
      MDLY = MXY1-MNY1
      MNX    = MNX1-1
      MNX(2) = MXX1+1
      MNX(3) = MNY1-1
      MNX(4) = MXY1+1
      IF(ITYPE.NE.IOV) GO TO 2
      IF(MNX1.LE.MXN) IXN = 1
      IF(MXY1.GE.MYP) IYP = 1
      IF(MNY1.LE.MYN) IYN = 1
C.....CHECK FOR BLOCKED PIECE
    2 IF(ISLX+ISLY.NE.0.AND.JLDF(JBM(15),IDUM(KFLG+IMGP)).EQ.1) GO TO 6
      IF(ISLY.EQ.0) GO TO 3
      IF(IDUM(KSTL+IMGP).EQ.9999) GO TO 3
C.....SPECIAL STRIPE GROUP
      IF(SY.EQ.99.0.AND.JLDF(JBM(13),IDUM(KFLG+NPEC+IMGP)).EQ.0) GO TO 3
      IYP = 1
      IYN = 1
      GO TO 6
    3 IF(ISLX.EQ.0) GO TO 6
      IF(IDUM(KPLI+IMGP).EQ.9999) GO TO 6
      IXP=1
      IXN=1
      GO TO 6
```

```
    2 IF(ITYPE.EQ.IBL) IDIR=1
      IF(ITYPE.EQ.IBR) IDIR=2
      IF(ITYPE.EQ.IBD) IDIR=3
      IF(ITYPE.EQ.IBU) IDIR=4
      TMP=FLOAT(MNX(IDIR))+(-1**IDIR)*400.0
      TMP = AMIN1(TMP,32600.0)
      TMP = AMAX1(TMP,-32000.0)
      MNX(IDIR) = TMP
    6 J = 1
      DO 10 I=1,NOING
      IF(I.EQ.IMGP) GO TO 10
C.....IGNOR SLAVE PIECES
      IF(JLDF(JMM(24),IDUM(KFLG+I)).EQ.1) GO TO 10
      MNX2 = IDUM(KXMN+I)-1
      MXX2 = IDUM(KXMX+I)+1
      MNY2 = IDUM(KYMN+I)-1
      MXY2 = IDUM(KYMX+I)+1
      IF(MNY2.GE.MYP) GO TO 10
      IF(MNX2.GE.MNX(2).OR.MXX2.LE.MNX   ) GO TO 10
      IF(MNY2.GE.MNX(4).OR.MXY2.LE.MNX(3)) GO TO 10
      IF(ITYPE.NE.IOV) GO TO 7
      LMX(1,J) = MAX0(MNX   ,MNX2)
      LMX(2,J) = MIN0(MNX(2),MXX2)
      LMY(1,J) = MAX0(MNX(3),MNY2)
      LMY(2,J) = MIN0(MNX(4),MXY2)
    7 INTPC(J) = I
      J = J+1
      IF(J.LE.50) GO TO 10
C.....DISPLAY "TOO MANY PIECES IN AREA"
      CALL MESSG (6,ITEX2,ITMP)
      IF(IND.NE.2) GO TO 900
      IP2 = -1
      RETURN
   10 CONTINUE
      IF(ITYPE.EQ.IOV) GO TO 500
C.....BUMP UP
      IF(J.NE.1) GO TO 435
  400 IF(IND.EQ.2) GO TO 950
      GO TO (410,415,420,425),IDIR
  410 IDUM(JXMN) = MXN
      IDUM(JXMX) = MXN+MDLX
      GO TO 430
C.....BUMP RIGHT TO LIMIT ONLY IF LEFT OF MXP
  415 IF(IDUM(JXMX).GT.MXP) GO TO 430
      IDUM(JXMX) = MXP
      IDUM(JXMN) = MXP-MDLX
      GO TO 430
  420 IDUM(JYMN) = MYN
      IDUM(JYMX) = MYN+MDLY
      GO TO 430
  425 IDUM(JYMX) = MYP
      IDUM(JYMN) = MYP-MDLY
  430 CALL XYSIP (IMGP,IORG,IDXP,IDYP)
      IF(IORG.EQ.0) GO TO 432
      CALL ORGIN(IDXP,IDYP,IORG,IMGP)
      CALL MOVI(1,IDXP,IDYP,IORG,IMGP)
  431 CALL MOVI (1,IDXP,IDYP,0,0)
      IF(ITYPE.EQ.IOS) GO TO 1
      GO TO 900
  432 III = IDUM(KADP+IMGP)
      CALL LITE(0,KADP+IMGP)
      IDXP = 0
      IDYP = 0
      IMGP = 0
      GO TO 431
```

```
  435 NINT = J-1
      IDIST = 9999
      JSTAT = LMT
      CALL GENXY (IMGP,IX1,IY1,IXY1,IFP,JSTAT)
      IF(JSTAT.EQ.1) GO TO 910
      IXY1 = IXY1+1
      IX1(IXY1) = IX1
      IY1(IXY1) = IY1
      DO 450 I=1,NINT
      JSTAT = LMT
      CALL GENXY (INTPC(I),IX2,IY2,IXY2,JFP,JSTAT)
      IF(JSTAT.EQ.1) GO TO 910
      IXY2 = IXY2+1
      IX2(IXY2) = IX2
      IY2(IXY2) = IY2
      JTC = 2
      NBEG2 = 1
      NVAL2 = IXY2
      IFIT=0
  443 IF(NBEG2.LT.IXY2) GO TO 442
      IF(IFIT.EQ.1) GO TO 450
      IFIT=1
      IF(NBEG2.EQ.IXY2) GO TO 450
      NBEG2=NBEG2-IXY2
      NVAL2 = IXY2-NXY2+3
  442 CALL FALL(MMX,MMX(3),IX2(NBEG2),IY2(NBEG2),NVAL2,JX2,NXY2,JTC)
      IF(NXY2.LE.0) GO TO 448
      JX2 = JX2+NBEG2-1
      IF(NBEG2.EQ.1) KX2 = JX2
      JTC = 1
      NBEG2 = JX2+NXY2-1
      NVAL2 = IXY2-NBEG2+1
C.....MINIMUM DISTANCE FROM INTPC TO IMGP
      DO 445 J = 1,NXY2
      JJ = INDEX((JX2+J-1),IXY2)
      ID = IDST(IX2(JJ),IY2(JJ),IX1,IY1,IXY1,IDIR,IFP)
      IF(ID.LT.IDIST) IDIST = ID
  445 CONTINUE
C.....MINIMUM DISTANCE FROM IMGP TO INTPC
      DO 447 J=1,IXY1
      NXY = NXY2
      IF(JX2.LE.IXY2) GO TO 1446
      JX2 = MOD(JX2,IXY2)
      GO TO 446
 1446 IF(NBEG2.LE.IXY2) GO TO 446
      NXY = MOD(NBEG2,IXY2)
      ID = IDST(IX1(J),IY1(J),IX2,IY2,NXY,-IDIR,JFP)
      IF(ID.LT.IDIST) IDIST=ID
      NXY = NXY2-NXY
  446 ID = IDST(IX1(J),IY1(J),IX2(JX2),IY2(JX2),NXY,-IDIR,JFP)
      IF(ID.LT.IDIST) IDIST=ID
  447 CONTINUE
      GO TO 440
  448 IF(NBEG2.EQ.1) GO TO 450
      NXY2 = IXY2-KX2+3
      NBEG2 = IXY2+1
      GO TO 443
  450 CONTINUE
      IF(IDIST.GE.9996) GO TO 405
      IDIST = IDIST*(-1**IDIR)
      IF(IND.EQ.2) GO TO 940
      GO TO (460,465,470,470),IDIR
  460 DXN = FLOAT(IDUM(JXMN))+FLOAT(IDIST)
      IDUM(JXMN) = MAX1(DXN,DMXN)
      IDUM(JXMX) = IDUM(JXMN)+MDLX
      GO TO 434
```

```
C..........BUMP RIGHT TO LENGTH
  465 IF(MXN.EQ.MXP) GO TO 466
      DX = FLOAT(IDUM(JXMX))+FLOAT(IDIST)
      IDUM(JXMX) = MIN1(DXN,DMXP)
      IDUM(JXMN) = IDUM(JXMX) -MDLX
      GO TO 432
  470 IDUM(JYMX) = IDUM(JYMX)+IDIST
      IDUM(JYMN) = IDUM(JYMN)+IDIST
      IF(IDUM(JYMX).GT.MYP) GO TO 425
      IF(IDUM(JYMN).LT.MYN) GO TO 426
      GO TO 432
C..........OVERLAP
  500 IF(J.EQ.1) GO TO 930
C.....CONTINUE ?
      IF(JCTR.EQ.1) GO TO 501
      CALL FSHIT (ITMP,1,0)
      IF(ITMP.EQ.IPP) GO TO 920
  501 NINT=J-1
      JSTAT = LMT
      CALL GENXY (INGP,IX1,IY1,IXY1,IPP,JSTAT)
      IF(JSTAT.EQ.1) GO TO 910
C..........CHECK WHERE POSSIBLE CONFLICT MAY OCCUR FOR A
C..........POSSIBLE INTERSECTION.
      DO 4?? I=1,NINT
      JSTAT = LMT
      CALL GENXY (INTPC(I),IX2,IY2,IXY2,JPP,JSTAT)
      IF(JSTAT.EQ.1) GO TO 910
      NBEG1 = 1
      NVAL1 = IXY1
   20 NBEG2 = 1
      NVAL2 = IXY2
      IF(NBEG1.GE.IXY1) GO TO 400
C..........FIND A STRING OF POINTS IN CONFLICT AREA FOR
C..........PIECE, INGP.
      JTC = 1
      IF(NVAL1.EQ.IXY1) JTC = 2
      CALL FALL(LMX(1,I),LMY(1,I),IX1(NBEG1),IY1(NBEG1),
     1  NVAL1,JX1,NXY1,JTC)
      IF(NXY1.LE.0) GO TO 400
      JX1 = JX1+NBEG1-1
      ITMP = JX1+NXY1-1
      IF(NBEG1.EQ.1.AND.MOD(ITMP,IXY1).GT.1) ITMP=MOD(ITMP,IXY1)
      NBEG1 = ITMP
      NVAL1 = IXY1-NBEG1+1
   25 IF(NBEG2.GE.IXY2) GO TO 20
C..........FIND A STRING OF POINTS IN CONFLICT AREA FOR
C..........PIECE, INTPC(I)
      JTC = 1
      IF(NVAL2.EQ.IXY2) JTC = 2
      CALL FALL(LMX(1,I),LMY(1,I),IX2(NBEG2),IY2(NBEG2),
     1  NVAL2,JX2,NXY2,JTC)
      IF(NXY2.LE.0) GO TO 20
      JX2 = JX2+NBEG2-1
      ITMP = JX2+NXY2-1
      IF(NBEG2.EQ.1.AND.MOD(ITMP,IXY2).GT.1) ITMP=MOD(ITMP,IXY2)
      NBEG2 = ITMP
      NVAL2 = IXY2-NBEG2+1
C..........CHECK FOR INTERSECTION OF LINESEGMENTS
  115 IX    = IX1(JX1)
      IY    = IY1(JX1)
      IFLG = 2
C..........FLAG FOR HORIZONTAL/VERTICAL OVERLAP
      IHV = 0
      IEND = NXY1-1
      IF(NXY1.EQ.IXY1) IEND = NXY1
```

```
      DO 150 II=1,JEND
      III = INDEX(JX1+II,IXY1)
      IX(2) = IX1(III)
      IY(2) = IY1(III)
      IF(IX    .EQ.IX(2).AND.IY    .EQ.IY(2)) GO TO 150
      IX(3) = IX2(JX2)
      IY(3) = IY2(JX2)
      JEND = NXY2-1
      IF(NXY2.EQ.IXY2) JEND = NXY2
      DO 140 JJ=1,JEND
      JJJ = INDEX(JX2+JJ,IXY2)
      IX(4) = IX2(JJJ)
      IY(4) = IY2(JJJ)
      IF(IX(3).EQ.IX(4).AND.IY(3).EQ.IY(4)) GO TO 130
C.....ALWAYS CHECK AFTER TWO INTERSECTIONS FOUND
      IF(IFLG.EQ.2) GO TO 1120
C.....ELIMINATE IMPOSSIBLE CASES DUE TO SEGMENTS BOUNDS
      IMXV = MAX0(IX(3),IX(4))
      IMNV = MIN0(IX(3),IX(4))
      JMXV = MAX0(IY,IY(2))
      JMNV = MIN0(IX,IX(2))
      IF(FLOAT(JMXV).LT.FLOAT(IMNV).OR.FLOAT(JMNV).GT.FLOAT(IMXV))
     + GO TO 130
      IMXV = MAX0(IY(3),IY(4))
      IMNV = MIN0(IY(3),IY(4))
      JMXV = MAX0(IY,IY(2))
      JMNV = MIN0(IY,IY(2))
      IF(JMXV.LT.IMNV.OR.JMNV.GT.IMXV) GO TO 130
 1120 CONTINUE
      CALL INTST(ITFLG,IX,IY,XINT,YINT)
      IF(IFLG.EQ.2.AND.ITFLG.NE.-1) GO TO 160
      IF(ITFLG)152,130,120
C.....CHECK FOR TANGENT POINT
  120 INTRP = 0
      IF(XINT.EQ.FLOAT(IX).AND.YINT.EQ.FLOAT(IY))INTRP=1
      IF(XINT.EQ.FLOAT(IX(2)).AND.YINT.EQ.FLOAT(IY(2)))INTRP=2
      IF(XINT.EQ.FLOAT(IX(3)).AND.YINT.EQ.FLOAT(IY(3)))INTRP=3
      IF(XINT.EQ.FLOAT(IX(4)).AND.YINT.EQ.FLOAT(IY(4)))INTRP=4
      IF(INTRP.EQ.0) GO TO 116
C.....CHECK POINT BEFORE AND AFTER FOR POSITION OF IN OR OUT
      IF(INTRP.LE.2) GO TO 117
C.....POINT ON INTPC(I)
      JXP = IDUM(KXMN+IMGP)-1
      JYP = IDUM(KYMN+IMGP) -1
      INTRP = JJJ+INTRP-4
      KKK = INDEX(INTRP-1,IXY2)
      IN1 = INOUT(IX2(KKK),IY2(KKK),IX1,IY1,IXY1,JXP,JYP)
      KKK = INDEX(INTRP+1,IXY2)
      IN2 = INOUT(IX2(KKK),IY2(KKK),IX1,IY1,IXY1,JXP,JYP)
      IF(IN1.EQ.IN2) GO TO 130
      GO TO 116
C.....POINT ON IMGP
  117 INTRP = III+INTRP-2
      ITMP = INTPC(I)
      JXP = IDUM(KXMN+ITMP)-1
      JYP = IDUM(KYMN+ITMP)-1
      KKK = INDEX(INTRP-1,IXY1)
      IN1 = INOUT(IX1(KKK),IY1(KKK),IX2,IY2,IXY2,JXP,JYP)
      KKK = INDEX(INTRP+1,IXY1)
      IN2 = INOUT(IX1(KKK),IY1(KKK),IX2,IY2,IXY2,JXP,JYP)
      IF(IN1.EQ.IN2) GO TO 130
      GO TO 116
  116 IF(IFLG.EQ.1) GO TO 127
      IF11 = II-1+JX1
      IF12 = JJ+JX2
```

```
      ITX = IRND(XINT)
      ITY = IRND(YINT)
      IFLG = 1
      GO TO 130
  127 IF(IRND(XINT).EQ.ITX.AND.IRND(YINT).EQ.ITY) GO TO 130
      JPT1=II+JX1
      JPT2 = JJ-1+JX2
      IJX = IRND(XINT)
      JJY = IRND(YINT)
      IF(JPT1-IPT1.NE.1.AND.IFP.EQ.JFP) GO TO 129
      IPT2 = IPT2-1
      JPT2 = JPT2+1
  129 ICTR = 0
      IF(JPT2.GT.IPT2) GO TO 131
      ITMP = JPT2
      JPT2 = IPT2
      IPT2 = ITMP
  131 IJPT1 = 1
      IJPT2 = 0
      IF(JPT1-IPT1.LE.1) IJPT1=1
      IF(JPT2-IPT2.LE.1) IJPT2=1
      IFLG = 2
      GO TO 130
C.....FIRST INTERSECTION IS ON TANGENT LINE, SO OMIT
  132 IFLG = MAX0(0,IFLG-1)
  130 IX(3) = IX(4)
      IY(3) = IY(4)
  140 CONTINUE
      IX = IX(2)
      IY = IY(2)
  150 CONTINUE
      IF(IFLG.NE.2) GO TO 25
  160 ICTR = ICTR+1
      IF((IXN*IXP*IYN*IYP).EQ.0.OR.ICTR.LT.10) GO TO 165
C.....PIECE NO LONGER MOVABLE
      GO TO 920
  165 IDIST = 0
      IDIR = 1
      IF(JJX.EQ.IIX.AND.JJY.EQ.IIY) GO TO 25
      IF(IABS(JJX-IIX).GT.IABS(JJY-IIY)) IDIR = 2
      IF(IDIR.EQ.1.AND.(IXP*IXN).EQ.1) GO TO 167
      IF(IDIR.EQ.2.AND.(IYP*IYN).EQ.1) GO TO 169
      GO TO 170
  167 IDIR = 2
      GO TO 170
  169 IDIR = 1
  170 IF(IJPT1.EQ.1.AND.IJPT2.EQ.1) GO TO 25
      IF(IJPT1.EQ.1) GO TO 185
      KBEG = IPT1+1
      KEND = JPT1-1
      KFIT = 0
      DO 180 K=KBEG,KEND
      KK = INDEX(K,IXY1)
      NXY = JPT2-IPT2+1
      IDT = 9999
      IF(JPT2.LE.IXY2) GO TO 175
      NXY = MOD(JPT2,IXY2)
      IDT = JDST(IX1(KK),IY1(KK),IX2,IY2,NXY,IDIR)
      NXY = JPT2-IPT2+2-NXY
      IX2(IXY2+1) = IX2
      IY2(IXY2+1) = IY2
  175 ID = JDST(IX1(KK),IY1(KK),IX2(IPT2),IY2(IPT2),NXY,IDIR)
      IF(IABS(ID).GT.IABS(IDT))ID=IDT
      IF(IABS(ID).EQ.9999) GO TO 180
      IF(IABS(ID).GT.IABS(IDIST)) IDIST=ID
      KFIT = 1
```

```
  180 CONTINUE
      IF(KFTT.EQ.0) IDIST = 9999
  185 IF(IJPT2.EQ.1) GO TO 195
      KBEG = IPT2+1
      KEND = JPT2-1
      KFTT = 0
      DO 190 K=KBEG,KEND
      KK = INDEX(K,IXY2)
      NXY = JPT1-IPT1+1
      IDT = 9999
      IF(JPT1.LE.IXY1) GO TO 187
      NXY = MOD(JPT1,IXY1)
      IDT = JDST(IX2(KK),IY2(KK),IX1,IY1,NXY,IDIR)*(-1)
      NXY = JPT1-IPT1+2-NXY
      IX1(IXY1+1) = IY1
      IY1(IXY1+1) = IY1
  187 ID=JDST(IX2(KK),IY2(KK),IX1(IPT1),IY1(IPT1),NXY,IDIR)*(-1)
      IF(IABS(ID).GT.IABS(IDT))ID=IDT
      IF(IABS(ID).EQ.9999) GO TO 190
      IF(IABS(ID).GT.IABS(IDIST)) IDIST=ID
      KFTT = 1
  190 CONTINUE
      IF(KFTT.EQ.0) IDIST=9999
  195 IF(IDIST.EQ.0) GO TO 200
      IF(IABS(IDIST).EQ.9999) GO TO 250
      IF(INV.EQ.2) GO TO 940
      GO TO (205,305),IDIR
  205 IF(IDIST.GT.0) IXN = 1
      IF(IDIST.LT.0) IXP = 1
      IF(IDIST.GT.0.AND.IXP.EQ.0) GO TO 220
      IF(IDIST.LT.0.AND.IXN.EQ.0) GO TO 220
      GO TO 250
  220 IYP = 0
      IYN = 0
      GO TO 460
  250 IF(INV.EQ.2) GO TO 920
C.........TRY OTHER DIRECTION
      IDIR = 3-IDIR
      INV = 2
      IDIST = 0
      GO TO 170
C.........CHECK REST OF POINTS IN AREA FOR INTERSECTIONS
  260 IF(JPT2.GE.JX2+NXY2-1) GO TO 25
      NXY2 = NXY2-(JPT2-JX2)
      JX2 = INDEX(JPT2,IXY2)
      GO TO 110
  305 IF(IDIST.GT.0) IYN = 1
      IF(IDIST.LT.0) IYP = 1
      IF(IDIST.GT.0.AND.IYP.EQ.0) GO TO 320
      IF(IDIST.LT.0.AND.IYN.EQ.0) GO TO 320
      GO TO 250
  320 IXP = 0
      IXN = 0
      GO TO 470
  400 CONTINUE
      GO TO 930
  900 RETURN
C.........DISPLAY "PIECE TOO LARGE"
  910 CALL MESAG(0,ITEXT,ITMP)
      IP2 = -1
      RETURN
C.........OVERLAP EXISTS
  920 IF(MMM1.LT.0) GO TO 921
```

```
C..........TURN ON "OVERLAP" AND DELAY
      CALL LIGHT (3,1)
      DO 9210 I = 1,2***
      ITMP = ITMP+1-1
 9210 CONTINUE
  921 IP2=-1
      RETURN
C..........NO OVERLAP, RETURN
  930 IP1=0
  935 IP2 = 0
      RETURN
C..........RETURN PARAMETERS FOR PACK
  940 IP1 = IDIST
      IP2 = IDIR
      RETURN
  950 GO TO (951,952,953,954),IDIR
  951 IP1=MXN-IDUM(JXMN)
      GO TO 955
  952 IP1=MXP-IDUM(JXMX)
      GO TO 955
  953 IP1=MYN-IDUM(JYMN)
      GO TO 955
  954 IP1=MYP-IDUM(JYMX)
  955 IP2 = IDIR
      RETURN
      END
      $ENDS

FTN4
      SUBROUTINE CCIM (IQUEU1,IQUEU2,IQUEU3,LIM,IQUEU4,LIM4,IQUEU5,LIM5,
     +                 IPNTR)
C
C     CONTROL CARD INTERPRETER FOR MAINTENANCE BATCH INPUT
C
      LOGICAL ERROR
C**
C**   IQUEU4 MUST BE DIMENSIONED IQUEU4(LIM4,3) BUT BUG IN
C**   VERSION 24177M KILLS US.  KLUGE FOLLOWS.
C**
      DIMENSION IQUEU4(12*,3)
      DIMENSION IQUEU1(1), IQUEU2(1), IQUEU3(1), IPNTR(6,5)
      DIMENSION IQUEU5(LIM5)
      DIMENSION LIM(3)
      DIMENSION ICWRD(2,4), ISUBF(2,9), MOVER(14)
      DATA ICWRD /1HP,1HU, 1HR,1HE, 1HL,1HI, 1HE,1HN/
      DATA ISUBF /1HP,1HI, 1HS,1HT, 1HM,1HA, 1HP,1HS,
     +            1HP,1HM, 1HS,1HM, 1HU,1HN, 1HO,1HF,
     +            1HU,1HI/
      DATA IDOLA, ICOLN, ICOMA, IDOST, IASTR, IACST, IEQAL
     +     /1H$, 1H:, 1H,, 1HD, 1H*, 1HA, 1H=/
      DATA MOVER(1) /0/
      DATA IU6 /6/
      WRITE (IU6,2500)
C
C     SET UP INITIAL VALUES FOR POINTERS
C
      DO 14 J=1,5,1
      DO 14 I=1,6,1
   10 IPNTR(I,J)=0
      DO 20 I=1,3,1
   20 IPNTR(2,I)=LIM(I)+1
      ERROR = .FALSE.
C
C**********COME HERE FOR NEXT COMMAND
C
```

```
  100 CONTINUE
      CALL NGF (MOVER)
      IF (MOVER(2).NE.2) GO TO 110
C                                     SIZE ERROR
  105 CALL NGF (-3)
      WRITE (IO6,1100)
      ERROR=.TRUE.
      GO TO 100
  110 IF (MOVER(3).EQ.IOOLA) GO TO 100
C                                     COMMAND MUST BE FOLLOWED BY :
      IF (MOVER(3).EQ.ICOLN) GO TO 120
      CALL NGF (-3)
      WRITE (IO6,1200)
      ERROR=.TRUE.
      GO TO 100
C                                     LOOK UP COMMAND IN TABLE
  120 DO 130 IC=1,4,1
      IF (MOVER(5).NE.ICARD(1,IC)) GO TO 130
      IF (MOVER(6).EQ.ICARD(2,IC)) GO TO 150
  130 CONTINUE
C                                     BAD COMMAND WORD
      CALL NGF (-3)
      WRITE (IO6,1300)
      ERROR=.TRUE.
      GO TO 100
C                                     PICK UP SUB FIELD
  150 IF (IC.EQ.4) GO TO 700
      CALL NGF (MOVER)
      IF (MOVER(2).EQ.2) GO TO 105
C                                     LOOK UP IN TABLE OF ALL SUBFIELDS
      DO 160 IS=1,9,1
      IF (MOVER(6).NE.ISUBF(2,IS)) GO TO 160
      IF (MOVER(5).EQ.ISUBF(1,IS)) GO TO 170
  160 CONTINUE
C                                     BAD SUB FIELD
      IS=99
C
C     BRANCH TO HANDLE APPROPRIATE COMMAND
C
  170 IF (IC-2) 200,400,600
C
C**********SPURGE:SUBFIELD     COMMAND
C
  200 CONTINUE
C                                     MUST HAVE <PI,>, <ST,>, OR <MA,>
      IF (IS.LE.3) GO TO 210
C                                     SPECIAL CASE FOR <DI,>
      IF (IS.EQ.9) GO TO 350
C                                     MISSING CORRECT SUBFIELD
  205 CALL NGF (-3)
      WRITE (IO6,1500)
      ERROR=.TRUE.
      GO TO 100
  210 IF (MOVER(3).EQ.ICOMA) GO TO 218
C                                     MISSING FIELD
  215 CALL NGF (-3)
      WRITE (IO6,1400)
      ERROR=.TRUE.
      GO TO 100
C
C     START TO PROCESS <IDNET> , <IDENT> , ... <IDENT>
C
  218 CALL NGF (MOVER)
      IF (MOVER(2).EQ.2) GO TO 105
      IF (MOVER(2).EQ.3) GO TO 215
```

```
C                       CHECK FOR <*DATE,>
      IF (MOVER(5).NE.IASTR) GO TO 270
      IF (IS.EQ.1) GO TO 220
C                                     BAD USE OF *
  219 CALL WGF (-3)
      WRITE (IO6,1000)
      ERROR=.TRUE.
      GO TO 100
  220 IF (MOVER(3).EQ.IEQUAL) GO TO 223
      CALL WGF (-3)
      WRITE (IO6,2300)
      ERROR=.TRUE.
      GO TO 100
C                                     CHECK FOR < D >
  223 IF (MOVER(6).NE.IDCST) GO TO 230
      GO TO 219
C                                     ILLEGAL NUMERIC FIELD
  225 CALL WGF (-3)
      WRITE (IO6,1700)
      ERROR=.TRUE.
      GO TO 100
  230 IF (MOVER(4).LT.1 .OR. MOVER(4).GT.1024) GO TO 225
      IPNTR(3,1)=MOVER(4)
      IF (MOVER(3).EQ.ICOMA) GO TO 218
      GO TO 100
C
C     COME HERE TO PROCESS PURGE IDENT, IDENT, ...
C
  270 CONTINUE
C                                     GOT ROOM FOR ONE MORE?
      IF (IPNTR(2,IS)-IPNTR(1,IS)-5 .GE. 0) GO TO 310
      CALL WGF (-3)
      WRITE (IO6,1900)
      ERROR=.TRUE.
      GO TO 100
  310 CALL WPK (MOVER(5),10)
      L=IPNTR(1,IS)+1
      IPNTR(1,IS)=L+4
      IF (IS-2) 311,312,313
  311 DO 301 I=0,4,1
  301 IQUEU1(I+L)=MOVER(I+5)
      GO TO 320
  312 DO 302 I=0,4,1
  302 IQUEU2(I+L)=MOVER(I+5)
      GO TO 320
  313 DO 303 I=0,4,1
  303 IQUEU3(I+L)=MOVER(I+5)
C                                     CHECK FOR MORE COMING 'ROUND DA MOUNT
  320 CONTINUE
      IF (MOVER(3).EQ.ICOMA) GO TO 218
      GO TO 100
C
C     COME HERE TO PARSE $PU:DI,*A
C
  350 CONTINUE
      IF (MOVER(3).NE.ICOMA) GO TO 215
      CALL WGF (MOVER)
      IF (MOVER(2).EQ.2) GO TO 105
      IF (MOVER(2).EQ.3) GO TO 215
      IF (MOVER(5).NE.IASTR) GO TO 215
      IF (MOVER(6).NE.IALST) GO TO 219
      IPNTR(1,5)=1
      GO TO 100
C
```

```
C**********$REPORT:SUBFIELD      COMMAND
C
  400 CONTINUE
C                                       ODD BALL QUEUE IF DIGITIZER RPT
      IF (IS.EQ.9) GO TO 550
C                                       MUST HAVE PI, ST, MA, PS, PM, OR SM
      IF (IS.GT.6) GO TO 205
  410 IF (MOVER(3).NE.ICOMA) GO TO 215
C                                       CHECK REPORT OR XREF
      IF (IS.GT.3) GO TO 510
  415 CALL NGF (MOVER)
      IF (MOVER(2).EQ.3) GO TO 215
      IF (MOVER(2).EQ.2) GO TO 105
C                                       *ALL FIELD
      IF (MOVER(5).NE.IASTR) GO TO 420
      IF (MOVER(6).NE.IACST) GO TO 219
      IPNTR(5,IS)=1
      GO TO 450
C                                       CHECK ON ROOM
  420 IF (IPNTR(2,IS)-IPNTR(1,IS)-5 .GE. 0) GO TO 430
  425 CALL NGF (-3)
      WRITE (IO6,2000)
      ERROR=.TRUE.
      GO TO 100
  430 CALL WPK (MOVER(5),10)
      L=IPNTR(2,IS)-5
      IPNTR(2,IS)=L
      IF (IS-2) 431,432,433
  431 DO 441 I=0,4,1
  441 IQUEU1(I+L)=MOVER(I+5)
      GO TO 450
  432 DO 442 I=0,4,1
  442 IQUEU2(I+L)=MOVER(I+5)
      GO TO 450
  433 DO 443 I=0,4,1
  443 IQUEU3(I+L)=MOVER(I+5)
C                                       CHECK FOR MORE COMING IN
  450 CONTINUE
      IF (MOVER(3).EQ.ICOMA) GO TO 415
      GO TO 100
C
C     STUF FOR PROCESSING XREF STUFF IN QUEUES
C
  510 CONTINUE
      CALL NGF (MOVER)
      IF (MOVER(2).EQ.3) GO TO 215
      IF (MOVER(2).EQ.2) GO TO 105
      IF (MOVER(5).EQ.IASTR) GO TO 219
      IF (LIM4-IPNTR(IS-3,4)-6 .GE.0) GO TO 520
      CALL NGF (-3)
      WRITE (IO6,2000)
      ERROR=.TRUE.
      GO TO 100
  520 CALL WPK (MOVER(5),10)
      L=IPNTR(IS-3,4)+1
      IPNTR(IS-3,4)=L+5
      DO 530 I=0,4,1
  530 IQUEU4(I+L,IS-5)=MOVER(I+5)
      IQUEU4(L+5,IS-3)=-1
      IF (MOVER(3).EQ.ICOMA) GO TO 510
      GO TO 100
C
C     COME HERE TO PROCESS REPORTS FOR THE DIGITIZER QUEUE
C
```

```
  550 CONTINUE
      IF (MOVER(3).NE.ICOMA) GO TO 215
  565 CALL WGF (MOVER)
      IF (MOVER(2).EQ.2) GO TO 105
      IF (MOVER(2).EQ.3) GO TO 215
C                                        LOOK FOR *ALL FIELD
      IF (MOVER(5).NE.IASTK) GO TO 570
      IF (MOVER(6).NE.IACST) GO TO 219
      IPNTR(5,5)=1
      GO TO 580
C                                        LOOP FOR IDENT,IDENT, ...
  570 IF (IPNTR(2,5)+5.GT.LIM5) GO TO 425
      CALL WPK (MOVER(5),10)
      L=IPNTR(2,5)+1
      IPNTR(2,5)=L+4
      DO 575 I=0,4,1
  575 IQUEU5(I+L)=MOVER(I+5)
  580 IF (MOVER(3).EQ.ICOMA) GO TO 565
      GO TO 100
C
C**********$LIST,ON    OR    $LIST,OFF
C
  600 CONTINUE
      IF (IS.EQ.7) GO TO 610
      IF (IS.EQ.6) GO TO 620
      CALL WGF (-3)
      WRITE (IO6,2100)
      ERROR=.TRUE.
      GO TO 100
C                                        TURN IT ON
  610 CALL WGF (-1)
      GO TO 100
C                                        TURN IT OFF
  620 CALL WGF (-2)
      GO TO 100
C
C**********SEND      COMAND
C
  700 IF (.NOT.ERROR) GO TO 710
      WRITE (1,2200)
      PAUSE
  710 WRITE (IO6,2400)
      RETURN
C
 1100 FORMAT("FIELD TOO BIG?")
 1200 FORMAT("MISSING COLON?")
 1300 FORMAT("BAD COMMAND?")
 1400 FORMAT("MISSING FIELD?")
 1500 FORMAT("ILLEGAL FILE?")
 1600 FORMAT("BAD USE FOR *?")
 1700 FORMAT("BAD NUMERIC?")
 1900 FORMAT("TOO MANY NAMES TO PURGE?")
 2000 FORMAT("TOO MANY NAMES TO REPORT?")
 2100 FORMAT("NOT ON OR OFF?")
 2200 FORMAT("ERROR(S) DETECTED.  TO CONTINUE TYPE :GO")
 2300 FORMAT("MISSING =?")
 2400 FORMAT("END BATCH INPUT")
 2500 FORMAT("START BATCH")
      END
      SUBROUTINE WGF (LIST)
      DIMENSION LIST(14)
C
C     LIST - ARRAY OF STUFF GOING IN AND COMING OUT
C     PRNT - TRUE IF LINES ARE TO BE PRINTED WHEN READ
C     ICOL - COLUMN OF CHAR BEING PROCESSED
C     LCOL - START COL NUMBER OF BAD FIELD
```

```
C      LINE = BUFFER FOR LINE INPUT
C      NCHAR = NUMBER OF CHAR BEING SENT TO LIST OUT
C      NUMBR = VALUE OF INTEGER BEING FORMED
C      ISTAT = SET TO 1 FOR ALPHA, RESET TO 0 WHEN DIGITS FOUND
C
       LOGICAL PRNT,SKIP
       DIMENSION LINE(80)
       DATA IB /0/, ICOL /80/, PRNT /.FALSE./, SKIP /.FALSE./
       DATA IO5, IO6 /5,6/
C                                          HP FTN4 BOMBS ON 1H= IN IF STMT
       DATA IEQ /1H=/
C
C      IS THIS A CONTROL FUNCTION?
C
       IF (LIST(1).LT.0) GO TO 500
C                                          NO DO INITIALIZATION
  100  DO 120 I=5,14,1
  120  LIST(I)=1H
       ISTAT=0
       NCHAR=0
       NUMBR=0
C                                          GET A CHARACTER FROM BUFFER
  140  ICOL=ICOL+1
       IF (ICOL.LE.IB) GO TO 150
       READ (IO5,1000) LINE
C                                          TRIM TRAILING BLANKS FOR TTY
       DO 145 IB=80,2,-1
       IF (LINE(IB).NE.1H ) GO TO 147
  145  CONTINUE
  147  IF (PRNT) WRITE (IO6,1000) (LINE(I), I=1,IB,1)
       ICOL=1
  150  IC=LINE(ICOL)
       IF (IC.EQ.1H ) GO TO 140
C                                          SKIP TO $ IF IN ERROR TRAP
       IF (.NOT.SKIP) GO TO 152
       IF (IC.NE.1H$) GO TO 140
       SKIP=.FALSE.
       GO TO 140
  152  NCHAR=NCHAR+1
       IF (NCHAR.GT.11) GO TO 180
       IF (IC.GE.1HA .AND. IC.LE.1HZ) GO TO 170
       IF (IC.GE.1H0 .AND. IC.LE.1H9) GO TO 160
       IF (IC.EQ.1H,) GO TO 155
       IF (IC.EQ.1H$) GO TO 155
       IF (IC.EQ.1H:) GO TO 155
       IF (IC.EQ.IEQ) GO TO 155
       IF (IC.EQ.1H.) GO TO 165
       IF (IC.NE.1H;) GO TO 170
C                                          DELIMETER FOUND
  155  LIST(3)=IC
       IF (NCHAR.GT.1) GO TO 156
       LIST(1)=ICOL
       LIST(2)=3
       RETURN
  156  LIST(1)=NCHAR-1
       LIST(2)=ISTAT
       LIST(4)=NUMBR
       RETURN
C                                          PROCESSING FOR DIGITS
  160  NUMBR=NUMBR*10+(IC-1H0)/256
       GO TO 175
  165  IF (ISTAT.EQ.0) ISTAT=4
       GO TO 175
```

```
C                                       PROCESSING FOR SIMPLE CHARACTERS
  170 ISTAT=1
  175 LIST(NCHAR+4)=IC
      GO TO 140
C                                       SIZE ERROR
  180 LIST(1)=ICOL
      LIST(2)=2
      LIST(3)=1H
      LIST(4)=0
      RETURN
C
C     THIS JUST MIGHT BE A CONTROL FUNCTION CALL - LOOK AND SEE
C
  500 IF (LIST(1).EQ.-1) GO TO 510
      IF (LIST(1).EQ.-2) GO TO 520
      IF (LIST(1).EQ.-3) GO TO 530
C                                       SORRY FOLKS - IS GARBAGE
      GO TO 100
C                                       TURN PRINT ON (=1)
  510 IF (.NOT.PRNT) WRITE (IO6,1000) (LINE(I), I=1,IB,1)
      PRNT=.TRUE.
      RETURN
C                                       TURN PRINT OFF (=2)
  520 PRNT=.FALSE.
      RETURN
C                                       FORCE CARD TO PRINT
  530 IF (.NOT.PRNT) WRITE (IO6,1000) (LINE(I), I=1,IB,1)
      WRITE (IO6,2000) ICOL
      SKIP=.TRUE.
      RETURN
C
 1000 FORMAT(80A1)
 2000 FORMAT("ERROR IN ABOVE STMT, COLUMN",I3)
      END
      SUBROUTINE WPK (LIST,NCHAR)
      DIMENSION LIST(NCHAR)
C
C     LIST - CHARACTERS LEFT JUSTIFIED
C     NCHAR - SIZE OF LIST
C
      MASKL=JFLD(15,8)
      MASKR=JFLD(7,8)
C
      N=1
      L=1
      DO 120 I=2,NCHAR,1
      L=1-L
      IF (L.EQ.0) GO TO 110
      LIST(N)=LIST(I)
      GO TO 120
  110 CALL JSTF (JLOF(MASKL,LIST(I)),MASKR,LIST(N))
      N=N+1
  120 CONTINUE
      RETURN
      END
ENDS


FTN4
      PROGRAM CHECK,3
C.....FEB.20,1975
C.....LYNN KIRSTEN, MARGARET ROWAN
C.....VERSION 2.0
C..........MAIN PROGRAM TO CHECK GRADE AND DIGITIZING OF
C..........PIECES RECENTLY DIGITIZED.
```

```
      DIMENSION MCK(3)
      DIMENSION IDUMM(640)
      COMMON JBP(10),JBG(4) ,IDNO(6),JFLAG,IRL1,IDIG(3),
     +MSIZ(3),XC,YC,BOX,IBR
      MCK(1) = 2HCH
      MCK(2) = 2HKA
      MCK(3) = 2H
      CALL EXEC (8,MCK)
      CALL CHKA
      CALL CHKB
      CALL CHKC
      CALL COMP
      CALL JSTF
      CALL XFORM
      CALL START
      CALL IRND
      CALL IABS
      CALL IFIX
      CALL ERF
      CALL ERFC
*     CALL RENIO
      CALL SQRT
      CALL ATAN2
      CALL IAND
      CALL IOR
      CALL FLOAT
      CALL ISSW
      CALL ODSC
      END
      END$


FTN4
      SUBROUTINE CHGVL(IMGNO,VAL,IDUM2,LNT)
      DIMENSION NDEC(2),NTERM(2)
C.....ROUTINE TO CHANGE THE NUMERICAL ASCII VALUE IN
C.....IMAGE NUMBER, "IMGNO", TO THE NEW VALUE, "VAL".
C.....FORMAT USED TO DISPLAY OLD VALUE IS MAINTAINED.
      DIMENSION IDUM2(1)
      DATA NDEC/27000B,56B/,NTERM/12000B,24B/
      DATA ISHF8/400B/,ISHF4/20B/
      NLD = 0
      NTD = -1
      INDX=IABS(IDUM2(IMGNO))
      IPTR=INDX+4
      LOT = 0
C.....DETERMINE FORMAT OF OLD VALUE.
      DO 100 I=1,10
      ICHK = IDUM2(IPTR)
      JCHK = IAND(ICHK,177400B)
      KCHK = IAND(ICHK,377B)
      IF(LOT.EQ.1) GO TO 5
      IF(JCHK.EQ.NDEC(1)) GO TO 20
      IF(KCHK.EQ.NDEC(2)) GO TO 30
    5 IF(JCHK.EQ.NTERM(1)) GO TO 260
      IF(KCHK.EQ.NTERM(2)) GO TO 250
      IF(LOT.EQ.1) GO TO 10
      NLD = NLD+2
      IPTR = IPTR+1
      GO TO 100
   10 NTD = NTD + 2
      IPTR = IPTR+1
      GO TO 100
   20 LOT = 1
      GO TO 100
```

```
   30 NTD = 0
      NLD = NLD+1
      LOT = 1
      IPTR = IPTR+1
  100 CONTINUE
      CALL ERR(6,M)
  250 IF(LOT.EQ.1) NTD=NTD+1
      IF(LOT.EQ.M) NLD = NLD+1
  260 IF(LOT.EQ.0) NLD = NLD-1
C.....TRUNCATE "VAL" IF IT IS TOO LARGE.
      VALUE = VAL
      IF(ABS(VAL).LT.(10.0**(NLD-1))) GO TO 270
      VALUE = VAL/(10.0**(NLD-1))
      N = VALUE
      VALUE = (VALUE-FLOAT(N))*(10.0**(NLD-1))
  270 CALL LITE(0,IMGNO)
      CALL STAT
      CALL DISVL(0,0,VALUE,NLD,NTD,0,IMGNO,IDUM2,INDX,LMT)
      CALL LITE(1,IMGNO)
      RETURN
      END
      END$

FTN4
      PROGRAM CHKA,5
C.....FEB.20,1973
C.....LYNN KIRSTEN, MARGARET ROWAN
C.....PROGRAM: CHECK
C.....VERSION 2.0
C.....SEGMENT TO SEARCH FOR PIECES IN DIGITIZER FILES
C.....CONVERT THE DIGITIZER CODES; SMOOTH THE DATA; AND
C.....STORE AS TEMPORARY PIECE ON THE FIXED PLATTER.
      DIMENSION MCK(3),IBUF(128),JBUF(1536),MFIL(3),XREF(2),
     1  YREF(2),IND(500),X(500),Y(500),IRUL(500)
      DIMENSION ISECT(3),IWPRD(3),IRLN(3)
      DIMENSION MNAM(6),IOPTN(3),ITPCD(3),NAMB(6),NAMR(3)
      DIMENSION IDUM(624)
      COMMON JBP(10),JBG(4),IDNO(6),JFLAG,NRLT,IDIG(3),
     1 MSIZ(3),XC,YC,BOX,IBR
      DATA ISECT/2HSE,2HCT,2HR /,IWPRD/2HWP,2HRD,2HD /
C.....POINT TYPE CONSTANTS
      DATA NBP/1/,NIP/2/,NEH/3/,NSL/4/,NND/5/,NGL/6/,NAP/7/,
     1     NNP/8/,NTP/9/,NLP/10/,NREF/11/,NFI/12/
      DATA NGEN/2/,MFIL/2HTP,2HEC,2HS /,MCK/2HCH,2HKB,2H  /,LIN/1/,
     + LOUT/6/
      DATA LMTXY/500/,MSIZ/30/,MD/2HD /,NMIRD/31/,NFINI/34/,
     + IACC/0/
      DATA IOPTN/2HOP,2HTN,2HS /,ITPCD/2HTP,2HEC,2HD /,NAMR/2HRD,
     + 2HEC,2HS /
      DATA SCAL/0.002/,IMIRD/0/,IRDT/0/,IPT/1/,IRL/0/,NRL/0/
      DATA IREF/1/,ISIZ/0/,MDPF/0/,LSEC/0/,LWD/6/
      JFLAG=0
      IBR=0
      MSIZ(1) = 2H
      MSIZ(2) = 2H
      MSIZ(3) = 2H
C.....CREATE FIELD DEFINITIONS FOR PIECE DEFINITION
      CALL FIELD
C.....DETERMINE #DIG.FILES,#SECT. PER FILE AND
C.....START TRACT/SECTOR
    6 CALL EXEC (14,66,IBUF,128,IOPTN,0)
C.....TURN OFF THE "BUSY" LIGHT
      CALL LIGHT(2,3)
      NDIGF = IBUF (6)
      MXSEC = IBUF (5)
```

```
C..........CHECK FOR NEXT NONZERO DIGITIZER FILE
      ISAMP = IBUF(9)
      L1 = IBUF(8)
    5 IF(L1.LE.0.OR.L1.GT.NDIGF) L1 = 1
   10 DO 20 I=L1,NDIGF
      CALL FLNAM (MD,I,IDIG)
      CALL EXEC(14,1028,IDNO,5,IDIG,0)
      CALL STOP
      IF(IDNO(1).NE.0)GO TO 30
C.....   GIVE OPERATOR OPTION TO EXIT ON SEARCH
      CALL DMESS(5,3,IEXIT)
      IF(IEXIT.EQ.-1)GO TO 26
   20 CONTINUE
   21 CALL STOP
C..........END OF DIGITIZER FILES
      CALL DMESS(2,IDNO,ISTAT)
C..........RETURN TO MONITOR
   26 IBUF(8) = 1
      CALL STOP
      CALL EXEC (15,66,IBUF,128,IOPTN,0)
      CALL RTMO
C..........DIGITIZER FILE FOUND,  CHECK?
   30 CALL DMESS(1,IDNO,IND)
      IF(IND.EQ.1)GO TO 33
C.....CHECK FOR "EXIT"
      IF(IND.EQ.3)GO TO 26
C..........NO, KEEP SEARCHING
      IF(I.EQ.NDIGF) GO TO 21
      L1 = I + 1
      GO TO 5
C..........YES, GET RULE SET
   33 N = 3
      IBUF(8) = I+1
      CALL EXEC (15,66,IBUF,128,IOPTN,0)
      CALL DMESS(N,IDNO,NRLT)
C.....N = 3 FOR NORMAL RETURN,N = -1 FOR ACC. W/O DISPLAY
C.....AND N = -2 FOR REJECT W/O DISPLAY
      IF(N.EQ.-2)GO TO 34
      IF(N.EQ.-1) IACC = 1
      GO TO 35
C..........PIECE REJECTED WITHOUT DISPLAY
   34 CONTINUE
      WRITE(LOUT,2)(IDNO(K),K=1,5)
      IBUF(1) = 0
      CALL EXEC(15,66,IBUF,1,IDIG,0)
      GO TO 6
C..........CHECK VALIDITY OF RULE TABLE
35    DNAM(6)=NRLT
      CALL DIREC(30068,MNAM,INTN1,NSCT1,ISTT)
      IF(ISTT.EQ.0)GO TO 40
C..........RULE TABLE NOT FOUND, RETURN
      CALL DMESS(4,IDNO,ISTAT)
      GO TO 6
C..........MAKE DIRECTORY ENTRY
C.....LOOK FOR PIECE I.D. ALREADY IN DIRECTORY
   40 CALL DIREC (10058,IDNO,INTNO,JSEC,ISTAT)
      IF(ISTAT.EQ.6) GO TO 44
C..........ERROR CONDITIONS
      GO TO (41,45,43,44,42),ISTAT
   41 WRITE(LOUT,1008)
      CALL RTMO
C.....INTERNAL NUMBER BAD
   42 CALL ERR(0,P)
C.....CAN NOT FIND TPECO FILE
   43 CALL ERR (0,0)
   44 WRITE(LOUT,1010) (IDNO(K),K=1,5)
```

```
 1010 FORMAT ("***DUPLICATE ID NUMBER OF ",5A2,/,
     1  "ENTER NEW ID NUMBER OR ",6H"EXIT",/,"$@")
C......TURN ON THE "INPUT REQ." LIGHT
       CALL LIGHT(1,1)
       READ (LIN,1001) IDNO
       CALL LIGHT(1,0)
       IF(IDNO(1).EQ.2HEX) GO TO 6
       GO TO 42
C..........CONVERT DIGITIZER CODES
   45 CALL CONVT(IDIG,MXSEC)
C......TURN ON THE "BUSY" LIGHT
       CALL LIGHT(2,1)
       IF(MXSEC.GT.0) GO TO 46
       WRITE(LOUT,8080)
 8080 FORMAT("DIGITIZER FILE OVERFLOW, REDIGITIZE PIECE")
       GO TO 360
   46 CONTINUE
C..........INITIALIZE HEADER INFORMATION
       CALL JSTF(NRLT,JBP(1),JBUF(2))
C..........GET DAY FROM BASE PAGE DATE
       CALL IDAY (ITMP)
       CALL JSTF (ITMP,JBP(6),JBUF)
       ITMP = 0
       CALL JSTF(ITMP,JBP(7),JBUF)
       CALL JSTF (ITMP,JBP(8),JBUF(2))
C..........STEP THROUGH DIGITIZER DATA
       CALL EXEC (14,66,IBUF,128,IDIG,0)
       IF(ISSW(0).LT.0) WRITE(6,1003) IBUF
   50 IF(LWD.GT.128) GO TO 110
       ITMP = IBUF(LWD)
       IF(ITMP.EQ.0) GO TO 120
       IF(ITMP.EQ.NFINI) GO TO 200
       IF(ITMP.EQ.NSIZ) ISIZ = 1
       IF(ITMP.EQ.NMIRO) IMIRO = 1
       IF(ITMP.EQ.NGL) IROT = 1
       IF(ITMP.GE.20) GO TO 90
C..........CURSOR BUTTON
       IF(NRL.NE.0) IRL = NRL
       ISIZ = 0
       IF(IPT.EQ.1) GO TO 60
       IROL(IPT-1) = IRL
   60 X(IPT) = SCAL*FLOAT(IBUF(LWD+1))
       Y(IPT) = SCAL*FLOAT(IBUF(LWD+2))
       NRL = 0
       LWD = LWD+3
       IF(ITMP.EQ.NREF) GO TO 70
C..........DATA POINT
       IND(IPT) = ITMP-1
C..........ITMP IS POINT TYPE
C..........  =1  GRADE POINT
C..........=2  INTERMEDIATE POINT
C..........=3  PUNCH HOLE
C..........=4  STRIPE LINE
C..........=5  NOTCH
C..........=6  GRAIN LINE
C..........=7  ALTERNATE START POINT
C..........=8  INTERNAL POINT
C..........=9  TRANSFER POINT
C..........=10 LETTERING POINT
C..........=11 REFERENCE LINE
C..........=12 FIXED INTERMEDIATE POINT
       IF(ITMP.NE.NGP.AND.ITMP.NE.NIP.AND.ITMP.NE.NTP.AND.ITMP.NE.NFI)
     1  GO TO 65
       IF(ITMP.EQ.NIP.OR.ITMP.EQ.NFI)GO TO 61
```

```
C.....GRADE OR TRANSFER POINT,CHECK FOR PREVIOUS POINT
C.....AT INT. POINT,
      IF(IPT.EQ.1) GO TO 65
      JTMP=IND(IPT-1)+1
      IF(JTMP.EQ.NIP.OR.JTMP.EQ.NFI)GO TO 61
      IF(JTMP.NE.NNO)GO TO 65
      IF(IPT.LE.2) GO TO 65
      JTMP=IND(IPT-2)+1
      IF(JTMP.NE.NIP.AND.JTMP.NE.NFI)GO TO 65
C.....LEAVE ROOM FOR "NGEN" SMOOTHED POINTS
   61 JTMP = IPT+NGEN
      IF(JTMP+1.GT.LMTXY) GO TO 365
      X(JTMP) = X(IPT)
      Y(JTMP) = Y(IPT)
      IND(JTMP) = IND(IPT)
      DO 63 I=1,NGEN
      II = IPT+I-1
      IND(II) = 99
      IRUL(II) = 0
      X(II) = 0.0
   63 Y(II) = 0.0
      IPT = IPT+NGEN+1
      IF(ITMP.NE.NGP.AND.ITMP.NE.NTP)GO TO 50
      MGPL = IPT-1
      GO TO 50
C.....SAVE INDEX FOR FIRST GRADE POINT AND LAST GRADE POINT
   65 IF(ITMP.NE.NGP.AND.ITMP.NE.NTP)GO TO 66
      IF(MGPF.EQ.0)MGPF = IPT
      MGPL = IPT
   66 IPT = IPT + 1
      IF(IPT.GT.LMTXY) GO TO 365
      GO TO 50
C.....REFERENCE POINT
   70 XREF(IREF) = X(IPT)
      YREF(IREF) = Y(IPT)
      IREF = 2
      GO TO 50
C.....FUNCTION BOX BUTTON
   90 IF(ITMP.GT.29) GO TO 97
C.....DIGIT
      IF(ISIZ.EQ.0) GO TO 95
C.....SAMPLE SIZE
      JTMP = (ISIZ+1)/2
      ISHF = 1
      IF(MOD(ISIZ,2).EQ.1) ISHF = 256
      MSIZ(JTMP) = MSIZ(JTMP)-48*ISHF+(ITMP+28)*ISHF
      ISIZ = ISIZ+1
      IF(ISIZ.GT.5) ISIZ = 0
      GO TO 100
C.....RULE NUMBER
   95 NRL = NRL*10+(ITMP-20)
   97 IF(ITMP.NE.NSIZ) ISIZ = 0
  100 LWD = LWD+1
      GO TO 50
C.....BUFFER IN ANOTHER SECTOR OF DIGITIZER DATA
  110 LSEC = LSEC+1
  120 CALL EXEC (14,66,IBUF,128,IDIG,LSEC)
      IF(ISSW(0).LT.0) WRITE(6,1003) IBUF
      LWD = 1
      GO TO 50
C.....FINISHED WITH DIGITIZE DATA BREAKDOWN
  200 IPT = IPT-1
      IF(NRL.NE.0) IRL = NRL
      IRUL(IPT) = IRL
      ITMP=IND(IPT)+1
```

```
      JTMP=IND(IPT-1)+1
      IF(ITMP.EQ.NIP.OR.ITMP.EQ.NFI)GO TO 205
      IF(ITMP.EQ.NND.AND.(JTMP.EQ.NIP.OR.JTMP.EQ.NFI)) GO TO 205
      GO TO 207
  205 IF(IPT+NGEN.GT.LMTXY) GO TO 365
      DO 206 I=1,NGEN
      II = IPT+I
      IND(II) = 99
      IRUL(II) = 0
      X(II) = 0.0
  206 Y(II) = 0.0
      IPT = IPT+NGEN
C.....SET UP ROTATION AND TRANFORMATION INFOR.
  207 ANG = 0.0
      IF(ISSW(1).LT.0) WRITE(6,1004) (IND(K),X(K),Y(K),IRUL(K),K=1,IPT)
      XMAX = -9999.9
      YMAX = -9999.9
      XMIN = 9999.9
      YMIN = 9999.9
      IF(IREF.NE.1) ANG = ATAN2((YREF(2)-YREF(1)),(XREF(2)-XREF(1)))
C.....FORCE REFERENCE LINE TO BE MIRROR
C.....LINE FOR MIRRORED PIECE
      IF(IMIRO.NE.0)ANG = ATAN2((Y(MGPL)-Y(MGPF)),(X(MGPL)-X(MGPF)))
      IF(ANG.EQ.0.0) GO TO 210
      S = SIN(ANG)
      C = COS(ANG)
C.....USE FIRST GRADE POINT AS TEMPORARY ORIGIN
C.....FOR MIRRORED PIECE
  210 IPT1=MGPF
      IF(IMIRO.EQ.0) GO TO 230
      XTR = X(IPT1)
      YTR = Y(IPT1)
  230 DO 250 I=1,IPT
      IF(IMIRO.EQ.0) GO TO 240
      X(I) = X(I)-XTR
      Y(I) = Y(I)-YTR
  240 IF(ANG.EQ.0.0) GO TO 245
      CALL XFORM (X(I),Y(I),X(I),Y(I),S,C)
  245 XMAX = AMAX1(X(I),XMAX)
      XMIN = AMIN1(X(I),XMIN)
      YMAX = AMAX1(Y(I),YMAX)
  250 YMIN = AMIN1(Y(I),YMIN)
C.....SET ORIGIN AS CENTER OF PIECE
      XTR = (XMAX+XMIN)/2.0
      YTR = (YMAX+YMIN)/2.0
      IF(IMIRO.EQ.0) GO TO 255
      YTR = 0.0
      GO TO 292
C.....FORCE CLOSURE OF NON-MIRRORED PIECES
  255 ITMP = 1
      JTMP = IND(IPT1+1)
      IF(JTMP.EQ.NND-1)ITMP=2
      DO 260 I=IPT1,IPT
      JTMP=IND(I)+1
      IF(JTMP.EQ.100)GO TO 260
      GO TO (260,260,270,270,260,270,270,270,260,270,270,260),JTMP
  260 CONTINUE
      IPT2 = IPT+1
      GO TO 285
  270 IPT2 = I
      DO 280 J=IPT,IPT2,-1
      JJ = J+ITMP
      IF(JJ.GT.LMTXY) GO TO 365
      IND(JJ) = IND(J)
      X(JJ) = X(J)
```

```
      Y(JJ) = Y(J)
  280 IRUL(JJ) = IRUL(J)
  285 DO 290 J=1,ITMP
      JJ = IPT2+J-1
      KK = IPT1+J-1
      IND(JJ) = IND(KK)
      X(JJ) = X(KK)
      Y(JJ) = Y(KK)
      IF(J.NE.1) GO TO 286
C.....CHECK TO MAKE SURE ALL RULES ARE IN THE
C.....RULE TABLE.SET IBR = 1 IF NOT
      NAMB(6) = IRLT
      CALL DIREC(3006R,NAMB,LLL,NNN,ISTAT)
      CALL EXEC(14,66,NRULS,1,NAMR,NNN)
      NRULS = JLOF(JFLD(15,9),NRULS)
  286 IF(IRUL(KK).GT.NRULS)IBR = 1
  290 IRUL(JJ) = IRUL(KK)
C.....SMOOTH DATA AND STORE IN JBUF
      IPT = IPT+ITMP
      IF(ISSW(2).LT.0) WRITE(6,1004) (IND(K),X(K),Y(K),IRUL(K),K=1,IPT)
  292 CALL TABLL (IPT,X,Y,IND,NGEN,ISTAT)
      IF(ISTAT.EQ.1) GO TO 360
      IF(ISSW(3).LT.0) WRITE(6,1004) (IND(K),X(K),Y(K),IRUL(K),K=1,IPT)
C.....COMPLETE HEADER
  295 IF(MSIZ.EQ.2H  .AND.ISAMP.NE.0) GO TO 320
      CALL SIZEP(1,NRLT,MSIZ,ISAMP)
C.....CHECK FOR ERROR CONDITION IN SAMPLE SIZE ENTRY
      IF(ISAMP)300,300,320
C.....BAD SIZE ENTRY
  300 WRITE(LOUT,1005) MSIZ
 1005 FORMAT("***SAMPLE SIZE, ",3A2," NOT AVAILABLE",/,
     1 "ENTER CORRECT SAMPLE SIZE OR ",6H"EXIT",/,"$0")
      CALL LIGHT(1,1)
      READ(LIN,1001) MSIZ
      CALL LIGHT(1,0)
      IF(MSIZ(1).EQ.2HEX) GO TO 6
      GO TO 295
C.....CORRECT SIZE, STORE IN BUFFER FOR OUTPUT
C.....MAKE TPECD ENTRY FOR PIECE
  320 CALL DIREC(4058,IDNO,INTNO,JSEC,ISTAT)
      IF(ISTAT.NE.0) GO TO 400
      CALL JSTF(ISAMP,JBP(2),JBUF(2))
      CALL JSTF (IMIRO,JBP(3),JBUF(1))
      CALL JSTF (IROT,JBP(4),JBUF(1))
      NSEC = (IPT/42)+1
      CALL JSTF(NSEC,JBP(5),JBUF(1))
      CALL EXEC (16,MFIL,ITMP)
      IF(NSEC+JSEC-1.LE.ITMP) GO TO 330
      WRITE(LOUT,1008)
 1008 FORMAT ("***TEMPORARY PIECE STORAGE AREA IS FULL.",/,
     1 "   PIECES MUST BE TRANSFERRED TO PERMANENT STORAGE",/,
     2 "   BEFORE CONTINUING.")
C.....CLEAR DIRECTORY ENTRY
  325 CALL DIREC(2408,IDNO,INTNO,JSEC,ISTAT)
      IF(ISTAT.EQ.0) CALL RTMO
      GO TO 400
  330 JPT = 3
C.....STORE PIECE DATA IN BUFFER FOR OUTPUT
      DO 350 I=1,IPT
      CALL JSTF(IND(I),JBP(9),JBUF(JPT))
      CALL JSTF(IBUL(I),JBP(10),JBUF(JPT))
      JBUF(JPT+1) = (X(I)-XTR)*1000.0
      JBUF(JPT+2) = (Y(I)-YTR)*1000.0
      JPT = JPT+3
      IF(MOD(JPT,128).NE.1)GO TO 350
```

```
      JBUF(JPT)=0
      JBUF(JPT+1)=0
      JPT=JPT+2
  350 CONTINUE
C.....END OF DATA CODE
      CALL JSTF(10,JBP(9),JBUF(JPT))
      CALL JSTF(0,JBP(10),JBUF(JPT))
      JBUF(JPT+1) = 0
      JBUF(JPT+2) = 0
      JPT = JPT+2
      NSEC=(JPT-1)/128+1
C.....SUPPRESS DATA BEFORE STORING
      CALL SQUEZ(JBUF,JPT,NSEC)
C.....STORE DATA IN TEMPORARY PIECE FILE, "TPECS"
      CALL JSTF(NSEC,JBP(5),JBUF)
      CALL EXEC (15,66,JBUF,JPT,MFIL,JSEC)
      IF(IOSW(4).LT.0) WRITE(6,1003) JBUF
      IF(IACC.EQ.0)GO TO 355
      IF(IMR.EQ.0) GO TO 351
      WRITE(LOUT,8045)
 8045 FORMAT("BAD RULE FOUND.CORRECT BEFORE SAVING")
      GO TO 325
C.....PIECE ACCEPTED WITHOUT DISPLAY
  351 IBUF = 0
      CALL EXEC(15,66,IBUF,1,IDIG,0)
      IBUF(1) = 2
      IBUF(2) = 6
      CALL EXEC(15,66,IBUF,2,ISECT,0)
      WRITE(LOUT,1)(IDNO(K),K=1,5)
      GO TO 6
C.....DISPLAY PIECE IN NESTED GRADE
  355 CALL EXEC(8,MCK)
      CALL CHK8
      CALL CHECK
C.....RELEASE DIGITIZER FILE
  360 IBUF(1) = 0
      CALL EXEC (15,66,IBUF,1,IDIG,7)
      GO TO 5
C.....ERROR IN DIGITIZER FILE, RELEASE AND STOP
  365 WRITE(LOUT,1030) LMTXY
 1030 FORMAT("NUMBER OF POINTS EXCEEDS LIMIT OF",I4)
      GO TO 360
  400 WRITE(LOUT,8030) ISTAT
 8030 FORMAT("TPECD ENTRY ERROR,ISTAT=",I4)
      CALL RTN0
    1 FORMAT("PIECE ",5A2," ACCEPTED")
    2 FORMAT("PIECE ",5A2," REJECTED")
 1001 FORMAT(6A2)
 1003 FORMAT(8O8)
 1004 FORMAT(I6,2F10.3,I6)
      END
      ENDS


FTN4
      PROGRAM CHK8,5
C.....FEB.26,1973
C.....LYNN KIRSTEN, MARGARET ROWAN
C.....PROGRAMS: CHECK
C.....VERSION 2.0
C.....SEGMENT TO GRADE PIECE TO ALL SIZES REQUESTED;
C.....STORE DATA INTO "CHEK"
      DIMENSION MCK(3),NAMR(3),IBUF(128),ISIZ(42)
      DIMENSIONMNAM(6)
      DIMENSION IOPTN(3)
```

```
      COMMON JOP(10),JAG(4),IDNO(6),JFLAG,IRLT,IDIG(3),
     +NSIZ(3),XC,YC,BOX,IBR
      DATA LIN /1/, LOUT /6/, MCK /2HCH,2HKC,2H  /
      DATA NAMR/2HRU,2HLE,2HS /
      DATA IOPTN/2HOP,2HTN,2HS /,MD/2HD /
C.....GET # OF SIZES FROM RULE TABLE
20    MNAM(6)=IRLT
      CALL DIREC(3HRUL,MNAM,INTNO,NSCTR,ISTAT)
      IF(ISTAT.NE.0)GO TO 41
      CALL EXEC(14,66,IBUF,2,NAMR,NSCTR)
      NS=JFLD(9,10)
      NSIZ=JLOF(NS,IBUF(2))
      JRLT=IRLT
      KRLT = JRLT*256
C.....LIMIT # OF SIZES TO 42
      IF(NSIZ.GT.42)NSIZ=42
C.....SET UP SIZES IN "ISIZ"
      DO 30 I=1,NSIZ
   30 ISIZ(I) = I+KRLT
      IBR = 0
C.....GRADE PIECE
   15 CALL GRADP(0,IDNO,NSIZ,ISIZ,ISTAT)
      IF(ISTAT.EQ.0) GO TO 40
C.....ISTAT = 99 WHEN NO END OF PIECE CODE FOUND
      IF(ISTAT.EQ.99) GO TO 50
C.....ISTAT = 98 WHEN A BAD RULE IS FOUND
      IF(ISTAT.EQ.98)IBR=1
      IF(ISTAT.LT.98)CALL ERR(0,0)
C.....GO TO SEGMENT "CHKC" TO DISPLAY PIECE
   40 CALL EXEC(5,MCK)
      CALL CHKC
      CALL CHECK
41    WRITE(LOUT,1005)ISTAT
1005  FORMAT("RULED FILE ERROR STATUS0= ",I5)
      GOTO 40
C.....ZERO BAD DIGITIZER FILE
   50 IBUF = 0
      CALL EXEC(14,66,IBUF,1,IDIG,0)
C.....PURGE FROM TPECO
      CALL DIREC(2HPR,IDNO,N,N,N)
      CALL RTMO
      END
      END$

FTN4
      PROGRAM CHKC,5
C.....FEB.20,1975
C.....LYNN KIRSTEN, MARGARET ROWAN
C.....PROGRAMS: CHECK
C.....VERSION 2.0
C.....DISPLAY NESTED PIECE FROM FILE CHEK
      DIMENSION LABEL(3),ICHK(3),IBUF(128),ICHKB(3),
     1   ISZ(42),XOFF(42),YOFF(42),ISST(42),ITPEC(3),ITPCS(3),
     2   ISECT(3),IWPRD(3),LBUF(9),LSZ(42),PERIM(42),AREA(42),
     3   IAR(2),IPR(2),INAME(3)
      DIMENSION IPLTM(5000),MCK(3)
      COMMON JAA(2),JA1,JA6,JSECT,JAC(3),JPTYP,JRLNO,JSZ,JAD,JSZN,
     +JRLT,IDNUM(6),JAF,IRLT,IDIG(3),MSIZ(3),XC,YC,BOX,IBR
      INTEGER GRADE(3)
      EQUIVALENCE (IAR,ARF),(IPR,PER)
      DATA ICHKB/2HCH,2HKB,2H  /,ITPEC/2HTP,2HEC,2HD /
      DATA MCK/2HCH,2HKA,2H  /
      DATA LIN/1/,LOUT/6/,ITPCS/2HTP,2HEC,2HS /,ICHK/2HCH,2HEK,2H  /
      DATA GRADE/2HGR,2HAD,2HE /,ISECT/2HSE,2HCT,2HR /,IWPRD/2HWP,2HRD,
     +2HD /
```

```
      DATA  NGTH/2/,LSPAC/5000/,JK/0/
      DATA XMIN/0./,YMIN/0./,XMAX/0./,YMAX/0./
      JPF = JFLD(15,1)
C.....READ HEADER FOR NUMBER OF SIZES AND RULE TABLE
      CALL EXEC(14,55,IBUF,128,ICHK,0)
      NSCTR = JLOF(JAD,IBUF)
      NSZ = (12*NSCTR)/((12*JLOF(JSZ,IBUF))+1)
      NS = (NSZ-1)/12+1
      DO 300 K=1,9
  300 LBUF(K) = 0
C.....LOOP TO READ IN EACH SECTOR OF HEADER INFORMATION
      DO 500 K = 1,NS
      IF(K.EQ.1)GO TO 350
      CALL EXEC(14,66,IBUF,128,ICHK,K-1)
  350 IL = 113
      NSZ = NSZ - 12
      IF(NSZ.LT.0)IL = ((NSZ+12) * 10) - 7
C.....LOOP TO PROCESS ONE SECTOR.  GETS LISTS OF SIZE NUMBERS,
C.....OFFSETS, AREAS, AND PERIMETERS.  FINDS MINIMUM X AND Y.
      DO 400 I=3,IL,10
      JK = JK + 1
      IF(JK.GT.42)GO TO 502
  370 ISZ(JK) = JLOF(JSZN,IBUF(I))
      IAR(1) = IBUF(I+5)
      IAR(2) = IBUF(I+6)
      IPR(1) = IBUF(I+7)
      IPR(2) = IBUF(I+8)
      AREA(JK) = ARE
      PERIM(JK) = PER
      XM = IBUF(I+1)/10000.
      YM = IBUF(I+2)/10000.
      XOFF(JK) = IBUF(I+3)/10000.
      YOFF(JK) = IBUF(I+4)/10000.
      AMX = XM + XOFF(JK)
      AMY = YM + YOFF(JK)
      AMXX = -XM + XOFF(JK)
      AMXY = -YM + YOFF(JK)
      IF(AMX.LT.XMIN)XMIN = AMX
      IF(AMY.LT.YMIN)YMIN = AMY
      IF(AMXX.GT.XMAX)XMAX = AMXX
      IF(AMXY.GT.YMAX)YMAX = AMXY
  400 CONTINUE
  500 CONTINUE
  502 LIM2 = JK
      NRLT = IRLT
      NSZ = JK
  503 LIM1 = 1
  504 AX = ABS(XMIN)
      AY = ABS(YMIN)
C.....INITIALIZE DISPLAY FILE
      CALL FN123(IDNUM,NSZ,ISZ,NRLT,IPLTB,IDF)
      LIDF = IDF
      AMAXX=AMAX1(ABS(XMIN),XMAX)*2.0
      AMAXY=AMAX1(ABS(YMIN),YMAX)*2.0
      AMAX=AMAX1(AMAXX,AMAXY)*1.1
      CALL SCALE(0.0,AMAX,AMAX,1)
      CALL INDSP(IDF,IPLTB,LSPAC)
      IGRL = 1
      DO 550 I=LIM1,LIM2
      ISST(I) = NS
C.....INITIALIZE DISPLAY FOR A PIECE
      CALL SCALE(IX,IY,XOFF(I),YOFF(I),2)
      CALL ENPEC(I,IX,IY,IPLTB,IDF)
      CALL DSPLY(0,0,-3)
      CALL EXEC(14,66,IBUF,128,ICHK,NS)
```

```
      MIF = JLOF(JMI,IBUF(1))
      NSECT = JLOF(JSECT,IBUF(1))
      NSC = NS
      NS = NS + NSECT
      X = 0.
      Y = 0.
      ISTL = 1
      INTP = 1
      INMI = 0
      IFLG = 1
      IGPF = 0
      L1 = 1
      L2 = NSECT
      L3 = 1
      L4 = 3
      L5 = 126
      L6 = 3
  500 DO 800 J = L1,L2,L3
C.....LOOP TO READ IN ALL SECTORS OF EACH PIECE.
      IF(INMI.NE.0.AND.J.NE.L1) L4=126
  510 DO 700 K=L4,L5,L6
C.....LOOP TO DETERMINE AND ACT ON EACH POINT TYPE.
      X1= X
      Y1 = Y
      NP1 = NPTYP
      X = IBUF(K+1)/1000.
      Y = IBUF(K+2)/1000.
      NPTYP = JLOF(JPTYP,IBUF(K))
      IF(INMI.NE.0) Y = -Y
      IF(NPTYP.EQ.0.OR.NPTYP.EQ.8) GO TO 550
C                                  GRADE POINT
      IF(NPTYP.EQ.1.OR.NPTYP.EQ.11) GO TO 550
C                                  INTERMEDIATE POINT
      IF(NPTYP.EQ.4)GO TO 610
C                                  NOTCH POINT
  530 IFLG = 1
      IF(NPTYP.EQ.2) GO TO 570
C                                  PUNCH HOLE POINT
      IF(NPTYP.EQ.3) GO TO 590
C                                  STRIPE LINE POINT
      IF(NPTYP.EQ.5) GO TO 630
C                                  GRAIN LINE POINT
      IF(NPTYP.EQ.6) GO TO 650
C                                  ALTERNATE START POINT
      IF(NPTYP.EQ.7) GO TO 670
C                                  INTERNAL POINT
      IF(NPTYP.EQ.9)GO TO 700
C                     CHARACTER POSITIONING
      IF(NPTYP.EQ.10)GO TO 830
C                                  END POINT
      GO TO 700
C                                  NONE OF THESE, IGNORE
C.....GRADE POINT OR INTERMEDIATE POINT
  550 IF(IFLG.EQ.0)CALL SETDS(X,Y,2)
      IF(IFLG.EQ.1)CALL SETDS(X,Y,3)
      IF(IGPF.EQ.0)IGPF=(J-1)*128 + K
      IGPN = (J-1)*128 + K
      IFLG = 0
      IF(INMI.EQ.0.OR.NP1.NE.4)GO TO 700
C.....AVOID MIRRORING NOTCHES ON THE AXIS
      IF(IGPN.EQ.IGPF.OR.IGPN.EQ.IGPL)GO TO 700
      CALL SETDS(X1,Y1,2)
      CALL SETDS(X,Y,3)
      GO TO 700
```

```
C.....DRAW PUNCH HOLE
  570 CALL SETDS(X-.1,Y,3)
      CALL SETDS(X+.1,Y,2)
      CALL SETDS(X,Y-.1,3)
      CALL SETDS(X,Y+.1,2)
      GO TO 700
C.....DRAW STRIPE LINE
  590 IF(INMI.NE.0)GO TO 700
      ISTF = -1**ISTL
      IF(ISTF.LT.0)CALL SETDS(X,Y,3)
      IF(ISTF.GT.0)CALL SETDS(X,Y,2)
      ISTL = ISTL + 1
      GO TO 700
C.....DRAW NOTCH
  610 IF(INMI.NE.0)GO TO 700
      CALL SETDS(X,Y,2)
      CALL SETDS(X1,Y1,3)
      GO TO 700
C.....DRAW GRAIN LINE
  630 IF(INMI.NE.0.OR.IGRL.GT.2)GO TO 700
      IGRF = -1 **IGRL
      IF(IGRF.GT.0)GO TO 635
      CALL SETDS(X,Y,3)
      GO TO 645
  635 DELX = X - X1
      DELY = Y - Y1
      IF(ABS(DELX).GT..05)GO TO 637
      THETA = 1.57079
      IF(DELY.LT.0)THETA = -THETA
      GO TO 640
  637 THETA = ATAN(DELY/DELX)
      IF(DELX.LT.0.)THETA = THETA + 3.14159
  640 C = .5 * COS(.5236)
      D = .5 * SIN(.5236)
      CCOS = C * COS(THETA)
      CSIN = C * SIN(THETA)
      DCOS = D * COS(THETA)
      DSIN = D * SIN(THETA)
      APP = CCOS - DSIN
      BPP = CSIN + DCOS
      APM = CCOS + DSIN
      BPM = CSIN - DCOS
      AMP = -CCOS - DSIN
      BMP = - CSIN + DCOS
      AMM = -CCOS + DSIN
      BMM = - CSIN - DCOS
      CALL SETDS(X1+APP,Y1+BPP,2)
      CALL SETDS(X1+APM,Y1+BPM,3)
      CALL SETDS(X1,Y1,2)
      CALL SETDS(X,Y,2)
      CALL SETDS(X+AMP,Y+BMP,2)
      CALL SETDS(X+AMM,Y+BMM,3)
      CALL SETDS(X,Y,2)
  645 IGRL = IGRL + 1
      GO TO 700
C.....DRAW ALTERNATE START POINT
  650 IF(INMI.NE.0)GO TO 700
      CALL SETDS(X+.05,Y+.0866,3)
      CALL SETDS(X-.05,Y-.0866,2)
      CALL SETDS(X-.05,Y+.0866,3)
      CALL SETDS(X+.05,Y-.0866,2)
      CALL SETDS(X-.1,Y,3)
      CALL SETDS(X+.1,Y,2)
      GO TO 700
C.....DRAW INTERNAL POINT
  670 INTF = -1**INTP
```

```
      IF(INTF.LT.0)CALL SETDS(X,Y,3)
      IF(INTF.GT.0)CALL SETDS(X,Y,2)
      INTP = INTP+1
  700 CONTINUE
      MSC = NSC + J - INMI
      CALL EXEC (14,66,IBUF,128,ICHK,MSC)
  800 CONTINUE
      GO TO 840
C.....END POINT, IF FIRST SIZE, SET UP DISPLAY BUFFER
C.....LENGTH REQUIRED FOR ONE SIZE.
  830 IF(LNGTH.NE.0)GO TO 835
      LNGTH = IDF - LIDF + 15
      IF(MIF.EQ.1)LNGTH = 2 * LNGTH
      LSPAC = LSPAC - LNGTH - LIDF
  835 IF(MIF.NE.1)GO TO 840
C.....IF MIRRORED, GO THROUGH PRESENT PIECE DATA IN REVERSE,
C.....NEGATING EACH Y VALUE.
      L1 = NSECT
      L2 = 1
      L3 = -1
      L4 = K-3
      L5 = 0
      L6 = -3
      IGPL = IGPN
      INMI = 2
      GO TO 845
  840 LSPAC = LSPAC - LNGTH
      IF(LSPAC.GT.0.OR.I.EQ.NSZ)GO TO 850
C.....IF SUFFICIENT ROOM IN DISPLAY FILE, SEEK NEW SIZE.
C.....OTHERWISE, IGNORE REMAINING SIZES.
      II = NSZ - 1
      DO 845 III = I,II
  845 IPLTB(III+6) = -1
      IPLTB(IDF) = 10000008
      IPLTB(IDF+1) = 0
      IDF = IDF + 2
      II = II - I
      WRITE(6,4)II
    4 FORMAT("0ISPLAY FILE OVERFLOW,  LAST ",I3," SIZES IGNORED.")
      GO TO 900
  850 CONTINUE
  900 IPLTB(NSZ+6) = 10070008
      IPLTB(IDF) = 10000008
      CALL DIREC (1,05B,IDNUM,INT,LSCTR,ISTAT)
      IF(ISTAT.EQ.0) GO TO 905
      WRITE(LOUT,8030) ISTAT
 8030 FORMAT("PIECE NOT FOUND IN TPECD,ISTAT =",I4)
      CALL ERR(0,0)
  905 ISAVE = 0
      JFLAG = 0
      LISTF = 1
      NESTF = 1
      IRF=2
      IMG = 1
      ISX = 1500
      ISY = 1500
      ILE=IDF+1
      CALL START(40,IPLTH,IPLTB)
C.....TURN OFF THE "BUSY" LIGHT
      CALL LIGHT(2,0)
      JF2=JFLD(0,1)
      JF1=JFLD(1,1)
  950 CALL FSHIT(ITMP,0)
      IF(ITMP.EQ.-1)ITMP=17
      IF(ITMP.NE.17.AND.ITMP.NE.0)ITMP=17-ITMP
```

```
      IF(ISSW(10),LT,0)WRITE(1,2222)ITMP
      IF(ITMP,EQ,0)GO TO 952
      IF(ITMP,EQ,17) GO TO 999
C                                          STOP
      IF(ITMP,GE,7)GO TO 955
C                                          NUMERAL
      IF(ITMP,EQ,6)GO TO 960
C                                          LIST RULE
      IF(ITMP,EQ,5)GO TO 965
C                                          CHANGE RULE
      IF(ITMP,EQ,4)GO TO 960
C                                          ACCEPT
      IF(ITMP,EQ,2)GO TO 965
C                                          REJECT
      IF(ITMP,EQ,3)GO TO 960
C                                          PLOT
      IF(ITMP,EQ,1)GO TO 996
C                                  LIST AREAS AND PERIMETERS
C.....NO FB HIT. CHECK STYLUS STATUS
952   CALL DTHIT(ISWD,ISX,ISY)
C.....CHECK FOR STYLUS IN PROXIMITY
      IF(ISSW(10),LT,0)WRITE(1,2222) ISWD,ISX,ISY
2222  FORMAT(08,2I6)
      IF(JLOF(JBW,ISWD),EQ,0)GO TO 950
      CALL CROSS(ISX,ISY,0,IPLTB)
      IX = ISX
      IY = ISY
C.....  LIST RULE ON PRESSURE
      IF(JLOF(JB1,ISWD),EQ,1)GO TO 960
      GO TO 950
  955 ISAVE = ISAVE*10 + 10 = ITMP
      GO TO 950
  960 IFL = 0
      GO TO 970
  965 IFL = 1
      JFLAG = 1
      GO TO 977
C..........SEARCH DATA FOR CLOSEST POINT
  970 CALL SCALE(IX,IY,X,Y,3)
      DELM = 2**,0
      SAVEX = 0.
      SAVEY = 0.
      MSCT = ISST(IMG)
      MS = MSCT
      NRLNO = 0
      CALL EXEC(14,66,IBUF,128,ICHK,MSCT)
      NSECT = JLOF(JSECT,IBUF(1))
      DO 975 I=1,NSECT
      DO 973 J=3,125,3
      NPTYP = JLOF(JPTYP,IBUF(J))
C..........INTERMEDIATE POINT?  IGNOR
      IF(NPTYP,EQ,1,OR,NPTYP,EQ,11) GO TO 973
C..........END OF PIECE?
      IF(NPTYP,EQ,10)GO TO 977
C..........LETTERING POINT?  IGNOR
      IF(NPTYP,EQ,9) GO TO 973
C..........NOTCH OR GRAIN LINE?  IGNOR
      IF(NPTYP,EQ,4,OR,NPTYP,EQ,5)GO TO 973
      XC = IBUF(J+1)/1000, + XOFF(IMG)
      YC = IBUF(J+2)/1000, + YOFF(IMG)
      IF(MIF,EQ,1,AND,Y,LT,0)YC = -YC
      DELTA = SQRT((X-XC)**2 + (Y-YC)**2)
      IF(DELTA,GT,DELM)GO TO 973
      DELM = DELTA
      SAVEX = XC
```

```
      SAVEY = YC
      KSEC = MS
      J = J
      NRLNO = JLOF(JRLNO,IBUF(J))
  975 CONTINUE
      MS = MSCT + 1
      CALL EXEC(14,66,IBUF,128,ICHK,MS)
  975 CONTINUE
  977 IF(IFL.EQ.1)NRLNO = ISAVE
      CALL SCALE(IX,IY,SAVEX,SAVEY,2)
C.....MOVE STAR TO POINT AND LIST RULE NUMBER
      CALL ASTAR(IX,IY,0,IPLTB)
      CALL DSRUL(NRLNO,IPLTB)
      IF(IFL.EQ.0)GO TO 950
C.....CHANGE RULE IN DATA
      CALL EXEC(14,66,IBUF,128,ICHK,KSEC)
      CALL JSTF(ISAVE,JRLNO,IBUF(K))
      CALL EXEC(15,66,IBUF,128,ICHK,KSEC)
      LSEC = KSEC - ISST(IMG) + LSCTR
      CALL EXEC(14,66,IBUF,128,ITPCS,LSEC)
      CALL JSTF(ISAVE,JRLNO,IBUF(K))
      CALL EXEC(15,66,IBUF,128,ITPCS,LSEC)
C.....DELETE STAR
      CALL ASTAR(IX,IY,1,IPLTB)
      ISAVE = 0
      GO TO 950
C.....FLAG TO ACCEPT PIECE
  980 IF(JOP.EQ.0) GO TO 981
      WRITE(LOUT,8060)
 8060 FORMAT("BAD RULE ON PIECE,CORRECT TO ACCEPT")
      GO TO 950
  981 IRF = 0
  123 IPLTB(NSZ+9)=1710018+1024*NESTF+256*LISTF
      GO TO 987
C.....FLAG TO REJECT PIECE
  985 IPLTB(NSZ+9) = 1740018 + 1024*NESTF + 256*LISTF
      IRF = 1
  987 NSEC = INT/18
C.....SET PURGE FLAG IN TEMPORARY PIECE DIRECTORY, "TPECD"
      CALL EXEC(14,66,IBUF,128,ITPEC,NSEC)
      IT = (INT+1-(NSEC*18)) *7
      CALL JSTF(IRF,JPF,IBUF(IT))
      CALL EXEC(15,66,IBUF,128,ITPEC,NSEC)
      GO TO 950
C.....PASS PLOT COMMAND TO MONITOR
  990 LBUF(1) = 2
      LBUF(2) = 4
      DO 995 JL=3,7
  995 LBUF(JL) = IDNUM(JL-2)
      LBUF(8) = IRLT
      CALL EXEC(15,66,LBUF,9,ISECT,0)
      NESTF = 0
      GO TO 985
C.....LIST AREAS FLAG SET
  996 LISTF = 0
C     LEAVE IRF ALONE
      GOTO 123
C.....STOP BUTTON HIT
  999 CALL STOP
      IF(JFLAG.EQ.0)GO TO 1010
C.....RULE CHANGE, REGRADE
      IF(IRF.EQ.1)GO TO 1010
 1005 CALL LIGHT(2,1)
      CALL EXEC(8,ICHKB)
      CALL CHKB
      CALL CHECK
```

```
C..........NOT ACCEPTED OR REJECTED (PURGE FROM "TPECD" ONLY)
1010  IF(IRF.EQ.2)CALL DIREC(2,0058,IDNUM,INT,NSEC,ISTAT)
      IF(LISTF.NE.0)GO TO 1020
      WRITE(LOUT,6)
      DO 1015 LL=1,NSZ
      CALL SIZEP(2,IRLT,INAME,LL)
      WRITE(LOUT,5)(INAME(I),I=1,3),AREA(LL),PERIM(LL)
 1015 CONTINUE
 1020 IF(LBUF(2).EQ.4)GO TO 1030
C..........RETURN COMPLETED TO MONITOR
      LBUF(1) = 2
      LBUF(2) = 6
      CALL EXEC(15,66,LBUF,2,ISECT,0)
      IF(IRF.EQ.0)WRITE(LOUT,1)(IDNUM(K),K=1,5)
      IF(IRF.EQ.1)WRITE(LOUT,2)(IDNUM(K),K=1,5)
      IF(IRF.EQ.2)WRITE(LOUT,7)(IDNUM(K),K=1,5)
    7 FORMAT("PIECE ",5A2," REMAINS IN DIGITIZER FILE")
      IF(IRF.EQ.2)GO TO 1033
C..........NO PLOT, RELEASE THE DIGITIZER FILE
      IBUF(1) = 6
      CALL EXEC(15,66,IBUF,1,IDIG,0)
      GO TO 1033
C..........."PLOT" PUSHED, TRANSFER CHEK TO GRADE.
 1030 CALL DIREC(4,0046,IDNUM,I,N,ISTAT)
      IF(ISTAT.EQ.0) GO TO 1025
      GO TO(1026,1035,1035,1035,1036),ISTAT
C.....CHECK FOR SUFFICIENT ROOM IN "GRADE" FILE
 1025 CALL EXEC(18,GRADE,M)
      IF(NSCTR+N.LT.M) GO TO 1031
 1026 CALL DIREC(2,0048,IDNUM,I,N,ISTAT)
      WRITE(LOUT,8080)
 8080 FORMAT("NO ROOM FOR DATA AT PRESENT,REGRADE TO CLEAR GRADE FILE")
      CALL RTMO
 1031 LSECT = -1
      DO 1032 I=1,NSCTR
      LSECT = LSECT+1
      CALL EXEC(14,1026,IBUF,128,ICHK,LSECT)
      CALL EXEC(15,1026,IBUF,128,GRADE,LSECT+N)
 1032 CONTINUE
      WRITE(6,3)(IDNUM(K),K=1,5)
      CALL EXEC(10,IXPRO)
 1033 CALL EXEC(8,MCK)
      CALL CHKA
      CALL CHECK
C..... PURGE IF DUPLICATE PIECE ALREADY IN GRADD FILE
1035  CALL DIREC(2,0048,IDNUM,I,N,ISTAT)
      IF(ISTAT.NE.0) GO TO 1036
      GO TO 1033
 1036 WRITE(LOUT,8070)ISTAT
 8070 FORMAT("GRADD FILE ERROR,STATUS =",I4)
      CALL ERR(0,0)
      GO TO  1033
    1 FORMAT("PIECE ",5A2," ACCEPTED")
    2 FORMAT("PIECE ",5A2," REJECTED")
    3 FORMAT("PIECE ",5A2," IN PLOT QUEUE")
    5 FORMAT(3A2,5X,F10.3,5X,F10.3)
    6 FORMAT(" SIZE",9X,"AREA",8X,"PERIMETER")
      END
      SUBROUTINE SETDS(X,Y,N)
      CALL SCALE(IX,1Y,X,Y,2)
      CALL DSPLY(IX,1Y,N)
      RETURN
      END
      ENDS
```

```
FTN4
      SUBROUTINE CONVT(INAME,MXSEC)
C.....OCT. 5,1972
C.....MARGARET ROWAN
C.....PROGRAMS:CHECK
C.....VERSION 2.0
      DIMENSION IDATA(256),INAME(3),JB(5),IVAL(15)
      DIMENSION IRTS(10)
      DIMENSION IISW(3)
C.....CONVERTS RAW DIGITIZER FILES
C.....PERFORMS RUBBED OUT DATA AND CANCELED DATA FROM
C.....DIG DATA.
      DATA IVAL/2H0 ,2H1 ,2H2 ,2H3 ,2H4 ,2H5 ,2H6 ,2H7 ,
     1          2H8 ,2H9 ,2H< ,2H= ,2H> ,2H? ,2H; /
C.....RELATION TABLE FOR ROTATING SWITCH
      DATA IRTS /6,15,12,10,9,7,8,3,11,4/
      DATA ICN/1028/,IFINI/34/
      IRMX=0
      IRMY=0
      NSECT = 0
C.....KEEP TWO SECTORS OF FILE IN CORE
      NUM = 256
      CALL EXEC (14,ICN,IDATA,NUM,INAME,NSECT)
      IF(IABS(IDATA(5)).LE.34) RETURN
C.....DEFINE FIELDS OF CONTROL WORD
C.....IMASK       ALWAYS 1111 IF A CURSOR CODE
C.....JB(1)       GRADE POINT BUTTON
C.....JB(2)       INTERM. POINT BUTTON
C.....JB(3)       NOTCH POINT BUTTON
C.....JB(4)       SWITCH BUTTON
C.....JB(5)       REGISTER OVERFLOW
C.....JSW         SCALE
C.....JRSW        ROTARY SWITCH VALUE
      IMASK = JFLD(7,4)
      DO 5   I=1,5
    5 JB(I)=JFLD(16-I,1)
      JSW = JFLD(9,2)
      JRSW = JFLD(3,4)
      N2 = 128
      NSECT = 1
      I = 0
   15 IF(I.GT.256) GO TO 150
      L = IDATA(I)
      IF(L.EQ.0) GO TO 57
      IF(JLDF(IMASK,L).NE.15.AND.L.GT.15) GO TO 20
C.....CURSOR HIT
      IDLL = 4
      IF(L.LT.20.AND.L.GT.0) GO TO 55
C.....DATA NOT PREVIOUSLY CONVERTED
C.....WHAT BUTTON USED?
   17 DO 60 J=1,5
      IF(JLDF(JB(J),L).EQ.1) IDATA(I) = J
   60 CONTINUE
C.....IGNOR ZERO BUTTON HIT
      IF(IDATA(I).EQ.5) IDATA(I) = 15
C.....NOTCH BUTTON HIT
      IF(IDATA(I).EQ.3) IDATA(I) = 5
      IF(IDATA(I).LE.15) GO TO 65
C.....ZERO COORDINATE.  NEW PIECE.
      IDATA(I) = 0
C
C     FOLLOWING ADDED FOR ZERO COOR. DIGITIZER PROBLEM.
C
      NX=(IDATA(I+1)+IRMX)/2
      NY=(IDATA(I+2)+IRMY)/2
```

```
      IX=IX+NX
      IY=IY+NY
      IRMX=(IDATA(I+1)+IRMX)-NX*2
      IRMY=(IDATA(I+2)+IRMY)-NY*2
      IDATA(I+1) = 0
      IDATA(I+2) = 0
      GO TO 55
   65 IF(IDATA(I).NE.4) GO TO 67
C.....ROTATING SWITCH.
      ITMP = JLOF(JRSW,L)+1
      IDATA(I) = IRTS(ITMP)
C          CONVERT DATA TO ABSOLUTE
   67 NX=(IDATA(I+1)+IRMX)/2
      NY=(IDATA(I+2)+IRMY)/2
      IX=IX+NX
      IY=IY+NY
      IRMX=(IDATA(I+1)+IRMX)-NX*2
      IRMY=(IDATA(I+2)+IRMX)-NY*2
C
      IF(IDATA(I).NE.2.AND.IDATA(I).NE.12) GO TO 66
C.....INTERMEDIATE BUTTON HIT, OMIT ANY DOUBLE HITS
      IF(IABS(IDATA(I+1)).LE.4.AND.IABS(IDATA(I+2)).LE.4) GO TO 68
   66 IF(IDATA(I).NE.15) GO TO 70
   68 IDATA(I) = 0
      IDATA(I+1)=0
      IDATA(I+2)=0
      GO TO 55
C.....SAVE ABSOLUTE DATA
   70 IDATA(I+1)=IX
      IDATA(I+2) = IY
   55 I=I+3
      GO TO 15
C.....NO DATA.  UPDATE RUBOUT POINTER
   57 IDEL = IDEL+1
      I = I+1
      GO TO 15
C.....FUNCTION BOX BUTTON HIT
C.....END PIECE, EXIT
   20 IF(L.EQ.IVAL(15)) GO TO 140
C.....RUBOUT
      IF(L.EQ.IVAL(14)) GO TO 50
C.....CANCEL OR *
      IF(L.EQ.IVAL(13)) GO TO 80
C.....CONVERT THROUGH TABLE, IVAL
      DO 21 J=1,12
      IF(L.EQ.IVAL(J)) GO TO 22
   21 CONTINUE
      IF(L.EQ.IFIN1) GO TO 140
      IF(L.LT.32) GO TO 30
C.....BAD DATA IN FILE
      IDATA(I) = 0
      WRITE(6,100) L
  100 FORMAT ("BAD CODE = ",O8)
      GO TO 30
C.....STORE CODE
   22 IDATA(I) = J+19
   30 I = I+1
      IDEL = 2
      GO TO 15
C.....RUB OUT CHARACTER.
   50 DO 25 J=1,IDEL
      M = I-J+1
   25 IDATA(M) = 0
      I = 1
      IF(NSECT.EQ.0) I=6
      GO TO 15
```

```
C..........CANCEL CHARACTER.
   80 IF(IPOST.EQ.NSECT) GO TO 85
      IF(IPOST.NE.NSECT) GO TO 90
      IF(IPC.GT.128) IPC=IPC-128
   85 DO 87 J=I,IPC,-1
   87 IDATA(J) = 0
      GO TO 30
   90 DO 92 J=I,1,-1
   92 IDATA(J) = 0
      CALL EXEC (15,ICN,IDATA(129),128,INAME,MSECT)
      DO 93 J=1,128
   93 IDATA(J+128)=IDATA(J)
      MSECT = MSECT-1
      NSECT = NSECT-1
      CALL EXEC (14,ICN,IDATA,128,INAME,NSECT)
      I = 128
      GO TO 30
C..........END OF DATA
  140 IDATA(I) = 34
      N2 = I
  150 CALL EXEC (15,ICN,IDATA,N2,INAME,NSECT)
      IF(IDATA(I).EQ.IFINI) RETURN
C..........SHIFT SECOND INCORE SECTOR TO FIRST;
C..........READ IN ANOTHER SECTOR.
      DO 160 J=1,128
  160 IDATA(J) = IDATA(J+128)
      NSECT = NSECT+1
      MSECT = MSECT+1
      IF(MSECT+1.GT.MXSEC) GO TO 170
      CALL EXEC (14,ICN,IDATA(129),128,INAME,MSECT)
      I = 129
      GO TO 15
C.....END OF DIGITIZER FILE REACHED W/O FINDING END OF PIECE
  170 MXSEC=-MXSEC
      RETURN
      END
      ENDS


FTN4
      SUBROUTINE CROSS(IX,IY,IDF,IDSPB)
C..........WHEN IDF IS NONZERO,SET UP INSTRUCTIONS FOR
C..........TRACING CROSS.  WHEN IDF IS ZERO, UPDATE
C..........SCREEN LOCATION OF CROSS.
      DIMENSION IBLOC(5),IDSPB(1)
      DATA IBLOC/40B+10B,0,1,101378,25B24B/
      IF(IDF.EQ.0)GO TO 20
      JDF = IDF
      IDSPB(JDF) = 40B22B
      IDSPB(JDF+1) = 177400B
      IDSPB(JDF+2) = 7776B
      DO 10   I=1,5
   10 IDSPB(JDF+I+4) = IBLOC(I)
   20 IDSPB(JDF+3) = IX*16
      IDSPB(JDF+4) = IY*16 +1
      RETURN
      END
      ENDS
```

```
ASMB
      NAM DATE,7
      ENT DATE
      EXT EXEC,.ENTR
*
*
*     SUBROUTINE DATE
*
*     THIS ROUTINE WILL RETURN OR STORE THE DOS-M
*     10 CHARACTER DATE STORED IN THE SYSTEM BUFFER SECTOR
*
*     CALLING SEQUENCE:
*
*     CALL DATE(INAME,IFLAG)
*
*        INAME = 5 WORD ARRAY FOR 10 CHARACTER DATE
*        IFLAG = 0 TO RETURN DATE
*                1 TO STORE DATE
*
*
INAME NOP             ADDRESS OF 5 WORD USER BUFFER
IFLAG NOP             OPERATION FLAG
DATE  NOP
      JSB .ENTR
      DEF INAME
*
      LDA CUDSC       LOAD CURRENT DISK SUBCHANNEL
      STA SAVSC       AND SAVE FOR RESTORE
      LDA SYSSC       LOAD SYSTEM SUBCHANNEL
      STA RQSCH       STORE FOR REQUESTED SUBCHANNEL
      JSB UDIO        CHANGE TO SYSTEM SUBCHANNEL
*
      LDA IFLAG,I     LOAD OPERATION FLAG
      SZA             READ OR STORE DATE?
      JMP STORE       STORE DATE
*
      LDA $DAD        A = SOURCE ADDRESS
      STA SAD         SAVE SOURCE ADDRESS
      LDA .-5         LOAD COUNT
      STA KONT        SAVE
      LDB INAME       B = DESTINATION ADDRESS
LOOP  LDA SAD,I       LOAD SOURCE WORD
      STA B,I         STORE IN DESTINATION BUFFER
      ISZ KONT        DONE?
      INB,RSS         NO, BUMP DESTINATION ADDRESS
      JMP RESET       YES, RESET AND EXIT
      ISZ SAD         BUMP SOURCE ADDRESS
      JMP LOOP        CONTINUE
KONT  OCT 0           WORD COUNT
SAD   OCT 0           SOURCE ADDRESS POINTER
$DAD  OCT 4227        ADDRESS OF DATE IN $BUF
*
RESET LDA SAVSC       LOAD ORIGINAL SUBCHANNEL
      STA RQSCH       STORE FOR REQUESTED SUBCHANNEL
      CPA SYSSC       USER DISK CHANGED?
      JMP DATE,I      NO, RETURN TO USER
      JSB UDIO        RESET TO ORIGINAL SUBCHANNEL
      JMP DATE,I      RETURN TO USER
*
STORE LDA INAME       A = SOURCE ADDRESS
      STA SAD         SAVE SOURCE ADDRESS
      LDA .-5         LOAD WORD COUNT
      STA KONT        SAVE
      LDB $DAD        B = DESTINATION ADDRESS
```

```
LOOP1  LDA SAD,I       LOAD SOURCE WORD
       JSB EXEC        EXEC BASE PAGE STORE
       DEF *+2         RETURN ADDRESS
       DEF RC19        REQUEST CODE ADDRESS
       ISZ SAD         BUMP SOURCE ADDRESS
       ISZ CONT        DONE?
       JMP LOOP1       NO, CONTINUE
       LDA DISCL       LOAD USER LABEL TRK/SEC
       CLA             CLEAR A-REGISTER
       RRR 8           SECTOR IN A, TRACK IN B
       ALF,ALF         ADJUST SECTOR TO LOWER A
       STA SECT#       STORE SECTOR #
       STB TRAK#       STORE TRACK #
       JSB EXEC        EXEC DISK READ/WRITE
       DEF *+7         RETURN ADDRESS
       DEF RC2         REQUEST CODE ADDRESS
       DEF CNTRL       CONTROL WORD ADDRESS
       DEF BBUF,I      BUFFER ADDRESS
       DEF BUFSZ       BUFFER SIZE ADDRESS
       DEF TRAK#       TRACK # ADDRESS
       DEF SECT#       SECTOR # ADDRESS
       JMP RESET       RESET AND EXIT
RC2    DEC -2          PRIV. EXEC WRITE CODE
RC19   DEC -19         BASE PAGE STORE REQUEST CODE
CNTRL  OCT 103         CONTROL WORD
BUFSZ  DEC 128         BUFFER SIZE
TRAK#  OCT 0           TRACK #
SECT#  OCT 0           SECTOR #
*
UDIO   NOP
       JSB EXEC        EXEC CHANGE USER DISK CALL
       DEF *+5         RETURN ADDRESS
       DEF RC23        REQUEST CODE
       DEF *           (NOT USED FOR PRIV. CALL)
       DEF RQSCH       SUBCHANNEL ADDRESS
       DEF RQFLG       PRIV. REQUEST FLAG ADDRESS
       JMP UDIO,I      RETURN
RC23   DEC 23          CHANGE USER DISK CALL
RQSCH  OCT 0           REQUESTED SUBCHANNEL
RQFLG  OCT 1           REQUEST FLAG
SAVSC  OCT 0           ORIGINAL SUBCHANNEL STORAGE
*
*      BASE PAGE COMUNICATION AREA
*
B      EQU 1           B-REGISTER
.      EQU 5B8         BASE PAGE CONSTANTS
SYSSC  EQU 1B5B        SYSTEM SUBCHANNEL
CUDSC  EQU 1B1B        CURRENT USER DISK SUBCHANNEL
DISCL  EQU 2B0B        USER LABEL TRK/SEC
BBUF   OCT 4117        SYSTEM BUFFER ADDRESS
       END

FTN4
       SUBROUTINE IDAY(ID)
C.....SUBROUTINE TO RETURN JULIAN DAY OF THE YEAR TO THE CALLER
       DIMENSION IASCD(5), IND(12), JDATE(10)
       DATA IND/0,31,59,90,120,151,181,212,243,273,304,334/
       IDD = 1
C.....FETCH, UNPACK, AND DECODE ASCII DATE
       IA = JFLD(15,8)
       IB = JFLD(7,8)
       CALL DATE(IASCD,0)
   10  DO 20 I = 1,5
       JDATE((2*I) - 1) = JLDF(IA,IASCD(I)) - 48
       JDATE(2*I) = JLDF(IB,IASCD(I)) - 48
```

```
    20 CONTINUE
       ISAVE = 0
       J = 0
       ICTR = 0
       ID = 0
       DO 30 K = 1,10
       IF(JDATE(K).LT.0)GO TO 25
       IF(JDATE(K).GT.9)GO TO 40
C.....ACCUMULATE NO MORE THAN TWO NUMERALS IN EACH FIELD
       ISAVE = ISAVE*10 + JDATE(K)
       J = J + 1
       IF(J.GT.2)GO TO 40
       GO TO 30
    25 IF(JDATE(K).EQ.-16)GO TO 30
       IF(JDATE(K).NE.-1)GO TO 40
       ICTR = ICTR + 1
       IF(ICTR.GT.1)GO TO 27
C.....FIRST FIELD - MONTH
       DO 26 I=1,12
       IF(ISAVE.EQ.I)ID = IND(I)
    26 CONTINUE
       IF(ID.EQ.0.AND.ISAVE.NE.1)GO TO 40
       GO TO 28
    27 IF(ICTR.GT.2)GO TO 50
C.....DAY
       IF(ISAVE.GT.31)GO TO 40
       ID = ID + ISAVE
    28 ISAVE = 0
       J = 0
    30 CONTINUE
       IF(ICTR.EQ.2)GO TO 50
C.....FORMAT ERROR - INPUT DATE AGAIN
    40 WRITE(100,1)
       READ(100,2)IASCD
       CALL DATE(IASCD,1)
       GO TO 10
    50 DO 55 JK=1,25
C.....CHECK FOR LEAP YEAR
       IF(ISAVE.EQ.4*(JK-1).AND.ID.GT.59)ID = ID + 1
    55 CONTINUE
     1 FORMAT("INPUT NUMERICAL DATE - FORM MO/DA/YR/",/,"$")
     2 FORMAT(5A2)
       RETURN
       END
       ENDS

FTN4
       FUNCTION IDIFF(IRLT,INDP1,INDS1,M,MMM)
C.......... FUNCTION TO CALCULATE THE INCREMENTAL MOVEMENT
C           FOR RULE M BETWEEN SIZES INDP AND INDS, USING
C           RULE TABLE IRLT  THE FLAG IXY INDICATES IF THE
C             MOVEMENT IS FOR X  (IXY=1) OR FOR Y (IXY=2)
       DIMENSION NAMR(3),NAMB(5),IDEL(2)
       INTEGER BUFR(256),R,B
       EQUIVALENCE(NRLT,NAMB(5))
       DATA NRLT/0/,NRUL/0/,NSIZ/0/
       DATA LOUT/6/,NAMR/2HRU,2HLE,2HS /
       IF (ISSW(2).LT.0) WRITE (6,1111) IRLT,INDP1,INDS1,M,MMM
 1111  FORMAT(5I5)
       IXY=IABS(MMM)
       IF(IRLT.EQ.NRLT)GO TO 10
C.......... NEW RULE TABLE
       NRLT=IRLT
       CALL DIREC(3HRUB,NAMB,INT,NSCTR,ISTAT)
       IF(ISTAT.NE.0)GO TO 71
```

```
      CALL EXEC(14,66,BUFR,2,NAMR,NSCTR)
      IRUL=JFLD(15,9)
      ISIZ=JFLD(9,10)
      NRUL=JLOF(IRUL,BUFR(1))
      NSIZ=JLOF(ISIZ,BUFR(2))+1
      BUFR(1)=-1
      BUFR(129)=-1
C.....  CALCULATE SECTOR # & O  OF DATA
C.....     SKIP OVER SIZE RELATION TABLE
      IRDSEC=(NSIZ-1)/42+1+NSCTR
   10 IF(M.LT.1.OR.M.GT.NRUL)GO TO 70
      IF(INDP1.GT.NSIZ.OR.INDS1.GT.NSIZ)GO TO 80
C..... CALCULATE SECTOR CONTAINING SIZE AND RULE
C              FOR INDP AND INDS
      IWDP=(INDP1*NRUL+M)*2-1
      IWDS=(INDS1*NRUL+M)*2-1
      ISECP=IWDP/128
      ISECS=IWDS/128
C CHECK TO SEE IF NEEDED SECTOR(S) IS(ARE) IN CORE.
      IP=0
      IS=0
      K=1
      M=1
   85 F=0
   90 IF(BUFR(K).NE.ISECP) GO TO 100
      F=1
      IP=K
  100 IF(BUFR(K).NE.ISECS) GO TO 110
      F=1
      IS=K
  110 IF(F.NE.0) GO TO 120
      IDEL(M)=K
      M=M+1
  120 IF(K.NE.1) GO TO 130
      K=129
      GO TO 85
  130 IF(IP.NE.0) GO TO 140
      IP=IDEL(1)
      IADPSC=ISECP+IRDSEC
      CALL EXEC(14,66,BUFR(IP),128,NAMR,IADPSC)
      ICURDP=ISECP
      M=2
      IF(ISECS.NE.ISECP) GO TO 140
      IS=IP
      GO TO 150
  140 IF(IS.NE.0) GO TO 150
      IP=IDEL(M)
      IADSSC=ISECS+IRDSEC
      CALL EXEC(14,66,BUFR(IS),128,NAMR,IADSSC)
      BUFR(IS)=ISECS
  150 IWDP=IWDP-ISECP*128+IP
      IWDS=IWDS-ISECS*128+IS
      IDIFF=BUFR(IWDP+IXY)-BUFR(IWDS+IXY)
      IF(ISSW(2).LT.0)WRITE(LOUT,1002)IDIFF,IXY
 1002 FORMAT(2I5)
      RETURN
   71 ISTAT=2
      WRITE(LOUT,1003)ISTAT
 1003 FORMAT("RULEFILE ERROR STATUS = ",I4)
      RETURN
   72 IF(IWDP.LT.0)WRITE(LOUT,1004)M
 1004 FORMAT("RULE",I3,"BAD")
      IDIFF=0
      RETURN
```

```
   80 WRITE(LOUT,1001)IRLT
1001 FORMAT("SIZE NOT IN RULE TABLE",16)
      IDIFF=0
      END
      END$

FTN4
      SUBROUTINE DIREC (ICODE,INAME,INTNO,NSCTR,ISTAT)
C
C     VERSION 2.1
C     EDITED 02 MAR 73
C     RONALD GORON
C
      DIMENSION INAME(6)
C
C     ISTAT = 0    FUNCTION PERFORMED
C     ISTAT = 1    NO MORE SPACE IN DIRECTORY
C     ISTAT = 2    DIRECTORY SEARCH FAILED
C     ISTAT = 3    DIRECTORY NOT FOUND
C     ISTAT = 4    DUPLICATE NAME IN DIRECTORY
C     ISTAT = 5    INTNO OUT OF RANGE
C
C     MFILE,MSECT = USED TO REMEMBER WHAT'S IN THE CORE BUFFER
C     IFIL1,IFIL2 = TABLE OF DIRECTORY NAMES, DATA FILE NAMES
C     ICHNL = TABLE OF CHANNEL ASSIGNMENTS FOR CORRESPONDING FILES
C     ICTRL = CONTROL WORD INDICATS BINARY MODE, DSK CHANNEL 2
C     IFLD1 = MASK FOR RIGHT HALF WORD
C     IFLD2 = MASK FOR NUMBER OF SECTORS (IN DATA AREA HEADER SECTOR)
C     IFLD3 = MASK FOR ADDRESS FIELD IN DIRECTORY ENTRY
C     IFLD4 = BIT MASK TO SET TO BE PURGED FLAG (BIT 15)
C     NENPS = NUMBER OF ENTRIES PER SECTOR IN DIRECTORY
C     NWDPE = NUMBER OF WORDS PER ENTRY IN DIRECTORY
C     IBFR1 = BUFFER SPACE FOR DIRECTORY SECTOR
C     LIMIT = LAST SECTOR IN DIRECTORY (OBTAINED FROM MONITOR)
C     KSIZE = ISIZE - NWDPE - 2
C
      LOGICAL REPLAY
      DIMENSION IBFR1(128)
      DIMENSION IFIL1(21), IFIL2(21), ICHNL(7)
      DIMENSION IFLD2(7)
      INTEGER OWORD, ONTNO, OSECT, OBFR1
      DATA IFIL1 /2HMA,2HRK,2HD ,
     +            2HPI,2HEC,2HD ,
     +            2HST,2HYL,2HD ,
     +            2HGR,2HAD,2HD ,
     +            2HTP,2HEC,2HD ,
     +            2HRU,2HLE,2HD ,
     +            2HMA,2HLF,2HD /
      DATA IFIL2 /2HMA,2HRK,2HR ,
     +            2HPI,2HEC,2HE ,
     +            2HST,2HYL,2HE ,
     +            2HGR,2HAD,2HE ,
     +            2HTP,2HEC,2HS ,
     +            2HRU,2HLE,2HS ,
     +            2HMA,2HLF,2HP /
      DATA ICHNL /1, 1, 0, 0, 0, 0, 0/
      DATA IFLD1 /000177B/,
     +     IFLD2 /077B,017B,017B,777B,017B,177B,003B/,
     +     IFLD3 /017777B/, IFLD4 /100000B/, ICTRL /000102B/
      DATA ISIZE /128/, IREAD /14/, IWRIT /15/, INITL /23/, ISRCH /1
     +     KSIZE /119/, NENPS /15/, NWDPE /7/
      DATA MFILE /0/
      DATA REPLAY /.FALSE./
C
C     EXTRACT IFUNC,INDEX FROM PARAM ICODE.  CHECK FOR VALIDITY.
```

```
C       IF OK, GO TO PROCESS COMAND.  ELSE ABORT.
C
        IFUNC=ICODE/256
        INDEX=IAND(ICODE,IFLD1)
        IF (IFUNC.LT.1 .OR. IFUNC.GT.11) GO TO 9001
        IF (INDEX.LT.1 .OR. INDEX.GT.7) GO TO 9001
        JAB=INDEX*3-2
C                                       SET USER DISK CHANNEL
        CALL UDSC (ICHNL(INDEX))
C                                       CHECK FOR FILE(S) ON DISK
        IF (MFILE.EQ.INDEX .AND. IFUNC.NE.11) GO TO 10
        CALL EXEC (ISRCH,IFIL1(JAB),LIMIT)
        IF (LIMIT.EQ.0) GO TO 9002
        LIMIT=LIMIT-1
   10   ISTAT=0
C       IFUNC = 1   2   3   4   5   6   7   8   9   10  11
        GO TO (100,200,300,200,300,400,300,900,300,200,500), IFUNC
C
C**********CREATE ENTRY IN APPROPPEATE DIRECTORY                (CODE=1)
C
C                                                               (CODE=0)
    90  CONTINUE
        REPLAY=.TRUE.
        UWORD=IWORD
        UNTNO=INTNO
        USECT=ISECT
        UBFR1=IBFR1(IWORD+7)
  100   CONTINUE
        MFILE=INDEX
C                                       LOOK IN EACH SECTOR OF THE DIRECTORY
        KSEC1=0
        DO 120 ISECT=0,LIMIT,1
        CALL EXEC (IREAD,ICTRL,IBFR1,ISIZE,IFIL1(JAB),ISECT)
C                                       GET LAST SECTOR ADRS FROM SECTOR 0
        IF (ISECT.GT.0) GO TO 105
        LASTA=IBFR1(ISIZE)
        IPLAT=IBFR1(ISIZE-1)
C                                       IN EACH SECTOR LOOK AT EACH ENTRY
  105   DO 120 IWORD=0,KSIZE,NWDPE
C                                       CHECK PURGE, FREE, IN USE
        IF (IBFR1(IWORD+7)) 120,115,110
C                                       DUPLICATE NAME?
  110   IF (INAME(1).NE.IBFR1(IWORD+2)) GO TO 120
        IF (INAME(2).NE.IBFR1(IWORD+3)) GO TO 120
        IF (INAME(3).NE.IBFR1(IWORD+4)) GO TO 120
        IF (INAME(4).NE.IBFR1(IWORD+5)) GO TO 120
        IF (INAME(5).NE.IBFR1(IWORD+6)) GO TO 120
        IF (REPLAY) GO TO 120
C                                       BAD SHOW - ITS BEEN DONE BEFORE...
        ISTAT=4
C                                       GIVE HIM NUMBER OF DUPLICATE
        INTNO=ISECT*NENPS+IWORD/NWDPE
        IF (INDEX.EQ.2) INTNO=INTNO+IPLAT*8192
        RETURN
C                                       FREE - SAVE ADRS IF FIRST FREE
  115   IF (KSECT.NE.0) GO TO 120
        KSEC1=ISECT+1
        KWORD=IWORD
        IF (REPLAY) GO TO 122
  120   CONTINUE
C                                       DID WE PICK UP FREE SLOT?
        IF (KSECT.LQ.0) GO TO 9004
        ISECT=KSECT-1
C                                       PULL PROPER SECTOR OF DIR INTO CORE
        IF (ISECT.NE.LIMIT)
```

```
     *     CALL EXEC (IREAD,ICTRL,IBFR1,ISIZE,IFIL1(JAB),ISECT)
C
C      MAKE ENTRY INTO DIR, SET USER PARAMETERS UP.
C
  122 CONTINUE
      INTNO=ISECT*NENPS*KWORD/NWDPE
C                                     ADD IN PLATTER NUMBR FOR PIECE FIL
      IF (INDEX.EQ.2) INTNO=INTNO+IPLAT*8192
C                                     DON'T FORGET TO MOVE NAME TO DIR
  125 DO 130 I=1,5,1
  130 IBFR1(KWORD+1+I)=INAME(I)
C                                     FORM DATA SECTOR ADDRESS
      CALL EXEC (IREAD,ICTRL,IHEAD, 1,IFIL2(JAB),LASTA)
      NSCTR=IAND(IHEAD,IFLD2(INDEX))
C                                     CHECK FOR DATA LOST
      IF (NSCTR.GT.0) GO TO 140
C                                     OK IF FIRST ENTRY
      IF (LASTA.EQ.0) GO TO 140
C                                     *CHEAT FOR USER*
      CALL ERR (31,1)
      CALL EXEC (IWRIT,ICTRL,1,1,IFIL2(JAB),LASTA)
      NSCTR=1
  140 NSCTR=NSCTR+LASTA
      IBFR1(KWORD+7)=NSCTR+1
      IF (REPLAY) IBFR1(KWORD+7)=IOR(OBFR1,IFLD4)
C                                     ENTER WORD ZERO IDENT
      IBFR1(KWORD+1)=INAME(6)
C                                     IF SECTOR 0 IN CORE ENTER LAST ADRS
      IF (ISECT.EQ.0) IBFR1(ISIZE)=NSCTR
      CALL EXEC (IWRIT,ICTRL,IBFR1,ISIZE,IFIL1(JAB),ISECT)
      IF (ISECT.NE.0) GO TO 150
      IF (REPLAY) GO TO 170
      RETURN
C                                     GET SECTOR 0 AND ENTER ADRS
  150 CALL EXEC (IREAD,ICTRL,IBFR1,ISIZE,IFIL1(JAB), 0)
      IBFR1(ISIZE)=NSCTR
      CALL EXEC (IWRIT,ICTRL,IBFR1,ISIZE,IFIL1(JAB), 0)
      ISECT=0
      IF (.NOT.REPLAY) RETURN
  170 CALL EXEC (IREAD,ICTRL,IBFR1,ISIZE,IFIL1(JAB),OSECT)
      IBFR1(OWORD+7)=NSCTR+1
      CALL EXEC (IWRIT,ICTRL,IBFR1,ISIZE,IFIL1(JAB),OSECT)
      REPLAY=.FALSE.
      INTNO=ONINO
      ISECT=OSECT
      RETURN
C
C**********DO DIRECTORY SEARCH GIVEN INAME
C
  200 CONTINUE
      MFILE=INDEX
      DO 210 ISECT=0,LIMIT,1
      CALL EXEC (IREAD,ICTRL,IBFR1,ISIZE,IFIL1(JAB),ISECT)
      IF (ISECT.EQ.0) IPLAT=IBFR1(ISIZE-1)
      DO 210 IWORD=0,KSIZE,NWDPE
C                                     HAS THIS ENTRY BEEN PURGED?
      IF (IBFR1(IWORD+7).LE.0) GO TO 210
C                                     ORDER OF IF'S MAY BE OPTIMIZED
      IF (INAME(1).NE.IBFR1(IWORD+2)) GO TO 210
      IF (INAME(2).NE.IBFR1(IWORD+3)) GO TO 210
      IF (INAME(3).NE.IBFR1(IWORD+4)) GO TO 210
      IF (INAME(4).NE.IBFR1(IWORD+5)) GO TO 210
      IF (INAME(5).EQ.IBFR1(IWORD+6)) GO TO 220
  210 CONTINUE
```

```
C                                       LOOKUP FAILED
      ISTAT=2
      ISECT=LIMIT
      RETURN
C
C     ALWAYS RETURN INTNO TO USER
C
  220 INTNO=ISECT*NENPS+IWORD/NWDPE
      IF (INDEX.EQ.2) INTNO=INTNO+IPLAT*8192
C
C     DIRECTORY ENTRY LOCATED.  CHECK FUNCTION CODE FOR WHAT NEXT?
C
      IF (IFUNC.EQ.4) GO TO 270
C
C     RETURN INTNO & NSCTR TO USER
C                                                              (CODE=2)
      NSCTR=IAND(IBFR1(IWORD+7),IFLD3)+1
      IF (IFUNC.EQ.10) INAME(6)=IBFR1(IWORD+1)
      RETURN
C
C     MARK DIRECTORY = THIS ENTRY TO BE PURGED                 (CODE=4)
C                                                              (CODE=5)
  270 IBFR1(IWORD+7)=IOR(IBFR1(IWORD+7),IFLD4)
  275 CALL EXEC (IWRIT,ICTRL,IBFR1,ISIZE,IFIL1(JAB),ISECT)
      RETURN
C
C**********DO DIRECTORY SEARCH GIVEN INTNO
C           CODES 3, 5, AND 7 COME HERE FIRST
C            CODE 9 TOO!
C
  300 CONTINUE
      I=IAND(INTNO,IFLD3)
      MSECT=I/NENPS
      IWORD=(I-MSECT*NENPS)*NWDPE
C                                       WITHIN RANGE FOR THIS DIR?
      IF (MSECT.LT.0 .OR. MSECT.GT.LIMIT) GO TO 9003
C                                       OPTIMIZED FOR SAME SECTOR REFRENCED
      IF (ISECT.EQ.MSECT .AND. MFILE.EQ.INDEX) GO TO 330
      CALL EXEC (IREAD,ICTRL,IBFR1,ISIZE,IFIL1(JAB),MSECT)
      ISECT=MSECT
      MFILE=INDEX
C                                       QUICK EXIT IF ENTRY ALREADY PURGED
  330 IF (IBFR1(IWORD+7).GT.0) GO TO 340
      ISTAT=2
      RETURN
C                                       BRANCH IF IN PURGE REQUEST
  340 IF (IFUNC.EQ.5) GO TO 270
C                                        BRANCH IF CODE 9
      IF (IFUNC.EQ.9) GO TO 370
C
C     RETURN INAME + NSCTR FROM DIRECTORY ENTRY                  (CODE=3)
C
      DO 350 I=1,5,1
  350 INAME(I)=IBFR1(IWORD+I+1)
      NSCTR=IAND(IBFR1(IWORD+7),IFLD3)+1
C                                       EXTRA FOR CODE 7         (CODE=7)
      IF (IFUNC.EQ.7) INAME(6)=IBFR1(IWORD+1)
      RETURN
C
C     CHANGE DIRECTORY HEADER WORD GIVEN INTNO
C                                                               (CODE=9)
  370 IBFR1(IWORD+1)=INAME(6)
      GO TO 275
C
C**********DO DIRECTORY SEARCH GIVEN WORD ZERO CONTNTS
C                                                              (CODE=6)
```

```
  400 CONTINUE
      ISECT= -1
      IF (MFILE.EQ.INDEX) GO TO 405
      MFILE=INDEX
      DO 410 ISECT=0,LIMIT,1
      CALL EXEC (IREAD,ICTRL,IBFR1,ISIZE,IFIL1(JAB),ISECT)
      IF (ISECT.EQ.0) IPLAT=IBFR1(ISIZE-1)
  405 DO 410 IWORD=0,KSIZE,NWDPE
C                                          HAS ENTRY BEEN PURGED?
      IF (IBFR1(IWORD+7).LE.0) GO TO 410
      IF (IBFR1(IWORD+1).EQ.INAME(6)) GO TO 420
  410 CONTINUE
C                                          LOOKUP FAILED
      ISTAT=2
      ISECT=LIMIT
      RETURN
C                                          RETURN DIRECTORY INFO
  420 DO 430 I=1,5,1
  430 INAME(I)=IBFR1(IWORD+1+I)
      NSCTR=IAND(IBFR1(IWORD+7),IFLD3)-1
      INTNO=ISECT*NENPS+IWORD/NWDPE
      IF (INDEX.EQ.2) INTNO=INTNO+IPLAT*8192
      RETURN
C
C**********RETURN PLATTER NUMBER
C
  500 CALL EXEC (1READ,ICTRL,IBFR1,ISIZE-1,IFIL1(JAB),0)
      INTNO=IBFR1(ISIZE-1)
      MFILE=0
      RETURN
C
C                                          ILLEGAL OPCODE SUPPLIED
 9001 CALL ERR (0,0)
C                                          DIRECTORY NOT FOUND ON DSK
 9002 ISTAT=3
      RETURN
C                                          INTNO OUT OF RANGE
 9003 ISTAT=5
      RETURN
C                                          NO ROOM FOR ENTRY IN DIR
 9004 ISTAT=1
      ISECT=LIMIT
      RETURN
      END
END$


FTN4
      SUBROUTINE DISCH(IX,IY,ISCAL,IIMG,ITEXT,IDUM2,INDX,LMT)
C.........CREATES DISPLAY INSTRUCTIONS TO DISPLAY CHARACTERS
C.........IN ITEXT.  TERMINATES ON CHARACTER, @.
C
C
C
      DIMENSION ITEXT(60),NSCAL(4),NDC4(2),LCHR(2)
      DIMENSION IDUM2(1)
      DATA LD/40000B/,NMR/22B/,NXR/10B/,NCH/10F17B/,
     1      NSCAL/1FFE,12000,14000,16000/,NDC4/12000B,24B/,NTR/1/,
     2      LSHF4/200B/,LSHF6/4000B/,LCHR/4000400B,100B/
      LPFLG = 0
C.........DETERMINE NUMBER OF CHARACTERS
      DO 10   I=1,60
      IF(ITEXT(I).EQ.LCHR(1)) GO TO 20
      N = (ITEXT(I)/LSHF8)*LSHF8
      N = ITEXT(I) - N
```

```
      IF(N.NE.LCHR(2)) GO TO 14
      LRFLG = 1
      GO TO 20
   10 CONTINUE
C..........MORE THAN 12 CHARACTERS
      CALL ERR(5,4)
   20 NNO = INDX+4+I
      IF(NNO.GT.LAST)CALL ERR(4,4)
      IOFPT = INDX
C..........CREATE DISPLAY INSTRUCTIONS
      IOFPT = IOFPT+2
      IDUM2(IOFPT+2) = LD+NXR
      IDUM2(IOFPT+3) = IX*LSHF4
      IDUM2(IOFPT+4) = (IY*LSHF4)+NTB
      IDUM2(IOFPT+5) = NCH+NSCAL(ISCAL)
      IOFPT = IOFPT+6
      M = I-1
      DO 50 I=1,M
      IDUM2(IOFPT) = ITEXT(I)
   50 IOFPT = IOFPT+1
      IF(LRFLG.EQ.1) GO TO 70
      IDUM2(IOFPT) = (ITEXT(I)-LCHR(2))+NOC4(2)
      GO TO 90
   70 IDUM2(IOFPT) = NOC4(1)
   90 INDX = IOFPT + 1
      RETURN
      END
      END$

FTN4
      SUBROUTINE DISVL(IX,IY,VAL,NLD,NTD,ISCAL,IMGNO,IDUM2,KNDX,LMT)
C..........ROUTINE TO CREATE DISPLAY INSTRUCTIONS TO
C..........GENERATE A FORMATTED NUMBER,  "NLD" IS THE
C..........NUMBER OF LEADING DIGITS, AND "NTD" IS THE
C..........NUMBER OF TRAILING DIGITS,  IF "NTD" IS
C..........LESS THAN ZERO, THEN THE NUMBER IS INTEGER.
      DIMENSION IDUM2(1)
      DIMENSION NSCAL(4),NOC4(2),NCHAR(15),NUM(15)
      DATA LD/40000B/,NMR/22B/,NXR/1B/,NCH/10017B/,
     1      NSCAL/100B,120B,140B,160B/,NOC4/120006B,24B/,
     2      NTB/1B/,LSHF4/20B/,LSHF6/4000B/,NUM0/60B/,
     3      NOCPT/56B/,NEG/55B/,NBLK/40B/,NFLG/0/
      IFLG = 0
      NDSP = KNDX
    7 NNX = IABS(NTD)
      IF (NTD.LT.0) NNX = 0
      NTDD = NNX
      VALUE = VAL
      IFTT = 0
      DO 10 I=1,15
   10 NCHAR(I) = NFLG
      IF (NTDD.EQ.0) GO TO 30
      DO 20 I=1,NTDD
   20 NCHAR(I) = NUM0
   30 INDX = NTDD + 1
      M = NOCPT
      IF(NTD.LT.0) M = NBLK
      NCHAR(INDX) = M
      INDX = INDX + 1
      L = 0
C..........STORE ASCII NUMBERS INTO NCHAR IN THE FORMAT
C..........CALLED FOR.
   40 IF (ABS(VALUE).GT.32768.0) L=ABS(VALUE)/10000.0
      A = FLOAT(L)*10000.0
      M = ABS(VALUE)-A
```

```
      MMM = IABS(M)
   50 MM = M/10 * 10
      NCHAR(INDX) = NUM0+M-MM
      IF (INDX.EQ.15) GO TO 100
      INDX = INDX+1
      IF (M.LT.10) GO TO 60
      M = M/10
      GO TO 50
   60 IF(L.EQ.0) GO TO 90
      IF(MMM.GE.10000) GO TO 80
      NEND = 5 - (INDX-1-NTOO)
      DO 70 I=1,NEND
      NCHAR(INDX) = NUM0
   70 INDX = INDX+1
   80 M = L
      L = 0
      GO TO 50
   90 IF (IFTT.EQ.1) GO TO 100
      IF(VALUE.LT.0.0) NCHAR(INDX) = NEG
      IF (NTOO.EQ.0) GO TO 100
      IF (ABS(VALUE).LE.32768.0) GO TO 95
      LL = ABS(VALUE)/10000.0
      VALUE = ABS(VALUE)-(FLOAT(LL)*10000.0)
   95 NN = ABS(VALUE)
      B = NN
      VALUE = (ABS(VALUE)-B)*(10.0**NTOO)
      INDX = 1
      IFTT = 1
      GO TO 40
  100 M = NLO + NTOO + 1
      DO 110 I=1,M
      IF (NCHAR(I).EQ.NFLG) NCHAR(I) = NBLK
  110 CONTINUE
      NX = 0
C.....COUNT NUMBER OF CHARACTERS USED.
      DO 120 I=1,15
      IF (NCHAR(-I+16).EQ.NFLG) GO TO 120
      NX = NX + 1
      NUM(NX) = NCHAR(-I+16)
  120 CONTINUE
  121 NWD = NDSP+4+(NX+1)/2
      IF(NWD.GT.LMT)CALL ERR(4,0)
      IF(ISCAL.LT.0) GO TO 125
      IF(ISCAL.EQ.0) GO TO 124
C.....CREATE HEADER.
      IDUM2(NDSP) = LO+NXR
      IDUM2(NDSP+1) = IX*LSHF4
      IDUM2(NDSP+2) = IY*LSHF4+NTB
      IDUM2(NDSP+3) = NCH+NSCAL(ISCAL)
  124 NDSP = NDSP+4
C.....CREATE INSTRUCTIONS TO DRAW CHARACTERS.
  125 DO 130 I=1,NX,2
      IF(I.EQ.NX) GO TO 140
      IDUM2(NDSP) = (NUM(I)*LSHF8)+NUM(I+1)
      NDSP = NDSP + 1
  130 CONTINUE
      IDUM2(NDSP) = NDC4(1)
      NDSP = NDSP+1
      GO TO 150
  140 IDUM2(NDSP) = (NUM(NX)*LSHF8)+NDC4(2)
      NDSP = NDSP+1
  150 KNOX = NDSP
      RETURN
      END
      ENDS
```

```
FTN4
      SUBROUTINE DMESS(N,IBLOC,ISTAT)
C
C.....FEB.2N,1975
C.....LYNN KIRSTEN, MARGARET ROWAN
C.....VERSION 2.0
C..... USED BY CHECK PROGRAM
C
      DIMENSION ITEX1(9),ITEX2(2),ITEX3(2),ITEX4(10),ITEX5(10),
     1  ITEX6(4),ITEX7(5),ITEX8(2),ITEX9(5),ITE10(3)
      DIMENSION IFLOC(6),ICHR(10),INAME(6),ICHRX(2),NRTBL(128)
      DIMENSION IXMIN(3),IXMAX(3),IYMIN(3),IYMAX(3)
      COMMON JBP(10),JBG(4),IDNO(6),JFLAG,IRLT,IDIG(3),MSIZ(3),
     1 XC,YC,BOX,IHF
      DIMENSION LBUF (250)
      EQUIVALENCE (ICHR(3),INAME)
      EQUIVALENCE (ICHR(9),ICHRX)
      DATA IXMIN/-1100,950,-200/,IXMAX/-600,1200,250/
      DATA IYMIN/-500,-500,-600/,IYMAX/-400,-400,-700/
      DATA ITEX1/2HCH,2HEC,2HK ,2H00,2H00,2H00,2H00,2H00,2H?0/,
     1  ITEX2/2HYE,2HS0/,ITEX3/2HNO,2H0 /,ITEX4/2HAL,2HL ,2HDI,
     2  2HG1,2HTI,2HZE,2HR ,2HFI,2HLE,2HS0/,ITEX5/2HHA,2HVE,2H B,
     3  2HLE,2HN ,2HSE,2HAR,2HCH,2HED,2H0 /,ITEX6/2HSP,2HEC,2HIF,
     4  2HY0/,ITEX7/2HRO,2HLE,2H T,2HAB,2HLE,2H0 /,ITEX8/2HXX,2HX0/,
     5  ITEX9/2H00,2HT ,2HFO,2HUN,2HD0/,ITE10/2HEX,2HIT,2H0 /
      DO 9 I=1,128
9     NRTBL(I)=0
      ISTAT = 0
      NLB = 2500
      ICHR(2)=2H
      ICHRX(1)=2H0
      IF(N.GT.5)RETURN
      GO TO (10,20,30,40,50),N
C.....1) CHECK?
   10 IDF = 6
      LBUF(1) = IDF
      LBUF(5) = 100000B
      CALL HEAD(LBUF,IDF)
      DO 11 K=4,8
   11 ITEX1(K) = IBLOC(K-3)
      CALL DISCH(-800,500,4,0,ITEX1,LBUF,IDF,NLB)
      LBUF(2) = IDF
      CALL DISCH(-1000,-500,4,1,ITEX2,LBUF,IDF,NLB)
      LBUF(3) = IDF
      CALL DISCH(1000,-500,4,2,ITEX3,LBUF,IDF,NLB)
      LBUF(4) = IDF
      CALL DISCH(-200,-800,4,3,ITE10,LBUF,IDF,NLB)
      CALL CROSS(0,0,IDF,LBUF)
      IDF=IDF+10
      LBUF(IDF) = 100000B
      CALL START(40,LBUF,LBUF)
      JB1=JFLD(1,1)
   12 CALL DTHIT(ISW0,LX,LY)
      CALL CROSS(LX,LY,0,LBUF)
C.....CHECK FOR PRESSURE ON THE STYLUS
      IF(ISW0(10).LT.0) WRITE(1,2222)ISW0,LX,LY
 2222 FORMAT(05,2I6)
      IF(JLDF(JB1,ISW0).EQ.0) GO TO 12
      DO 13   I=1,3
      IF(IXMIN(I).GT.LX.OR.IXMAX(I).LT.LX) GO TO 13
      IF(IYMIN(I).GT.LY.OR.IYMAX(I).LT.LY) GO TO 13
      GO TO 14
   13 CONTINUE
      GO TO 12
   14 ISTAT=I
```

```
      IF(ISTAT.EQ.0)GO TO 12
      CALL STOP
      RETURN
C.....2) ALL DIGITIZER FILES HAVE BEEN SEARCHED
   20 IDF = 4
      LBUF(1) = IDF
      LBUF(3) = 1[illegible]
      CALL HEAD(LBUF,IDF)
      CALL DISCH(-1[illegible],5[illegible],4,0,ITEX4,LBUF,IDF,NLB)
      LBUF(2) = IDF
      CALL DISCH(-1[illegible],-5[illegible],4,0,ITEX5,LBUF,IDF,NLB)
   22 LBUF(IDF) = 1[illegible]
      CALL START(4[illegible],LBUF,LBUF)
C.....TIMING LOOP
      DO 24 I=1,2
      DO 24 J=1,25[illegible]
      ND = ND + 1
      ND = ND - 1
   24 CONTINUE
      CALL STOP
      RETURN
C.....3) SPECIFY RULE TABLE
30    IDF=7
      LBUF(1) = IDF
      LBUF(6)=1[illegible]
      CALL HEAD(LBUF,IDF)
      LOF = IDF - 1
      LBUF(LOF-1) = 1774[illegible]
      LBUF(LOF) = 175[illegible]
      LBUF(2) = IDF + 1
      JDF = IDF
      LBUF(IDF)= 0
      IDF = IDF + 1
      CALL DISCH(-400,1750,4,0,ITEX6,LBUF,IDF,NLB)
      LBUF(3) = IDF
      CALL DISCH(-600,150[illegible],4,0,ITEX7,LBUF,IDF,NLB)
      LBUF(4) = -(IDF+1)
      LBUF(IDF) = 1[illegible]
      IDF = IDF+1
      KDF = IDF
      CALL DISCH(-400,-190[illegible],4,0,ITEX8,LBUF,IDF,NLB)
      ISAVE = 0
      LBUF(5)=IDF
      IXR=-190[illegible]
      IYR=15[illegible]
      DO 315 IXK=1,127
      INAME(6)=IXK
      CALL DIREC(3[illegible],INAME,INTNO,NSCTR,ISTAT)
      IF(ISTAT.EQ.0) GOTO 306
      GO TO (305,315,305,305,305),ISTAT
306   NRTBL(IXK)=INAME(6)
      INAME(6)=2H*
      IF(IXK.GT.99)GO TO 3085
      IF(IXK.GT.9)GO TO 309
      ICHR(2)=(IXK+60B)*256+40B
      ICHR(1)=2H
      GO TO 311
C.....  CONVERSION FR 3 DIGIT NUMBWER
3085  IXKT1=IXK/100
      ICHR(1)=(60B+IXKT1)*256
      IXKT2=(IXK-IXKT1*100)/10
      ICHR(1)=ICHR(1)+60B+IXKT2
      ICHR(2)=(IXK-IXKT1*100-IXKT2*10+60B)*256+60B
      GO TO 311
```

```
C.........CONVERSION FOR 2 DIGIT NUMBER
309     IXX1=IXX/10
        IXX=60B+IXX1
        ICHR(1)=400B*256+IXX
        IXX=IXX-IXX1*10
        IXX=IXX+30
        ICHR(2)=IXX+256*40B
311     CALL DISCH(IXB,IYB,2,0,ICHR,LBUF,IDF,NLB)
        IYB=IYB-60
        IF(IYB.LT.-1910)IXB=IXB+800
        IF(IYB.LT.-1900)IYB=1300
315     CONTINUE
316     LBUF(IDF)=100000B
        CALL START(4F,LBUF,LBUF)
        LRL=0
C
C
C..... THE FOLLOWING SECTION OF COE CHECKS STATUS OF FUNCTION BOX
C
C
   31 CALL FSHIT(ITMP,0)
      IF(ITMP.EQ.-1)ITMP=17
      IF(ITMP.NE.17.AND.ITMP.NE.0)ITMP=17-ITMP
      IF(ISS+(10).LT.0) WRITE(1,2222)ITMP
      IF(ITMP.LE.1)GO TO 31
C.........STOP PUSHED GO TO EXIT PROCEDURE
      IF(ITMP.EQ.17)GO TO 39
C.........LIST OR CHANGE RULE
      IF(ITMP.EQ.5.OR.ITMP.EQ.6)GO TO 34
C.........4=ACCEPT
      IF(ITMP.EQ.4)GO TO 32
C.........2=REJECT
      IF(ITMP.EQ.2)GO TO 33
C.........3=PLOT - IGNORED AT THIS POINT
      IF(ITMP.EQ.3)GO TO 36
C.........MUST BE A NUMBER - STACK IT INTO ISAVE
      ISAVE = ISAVE*10 + 16 - ITMP
      GO TO 31
   32 N = -1
      LBUF(LDF) = 1730016
      GO TO 31
   33 N = -2
      LBUF(LDF) = 1760016
      GO TO 31
   34 LBUF(JDF) = 100000B
      LRL=1
      LBUF(2) = -(JDF+1)
      IF(ISAVE.GT.127) GO TO 36
      ISTAT=VRTBL(ISAVE)
      IF(ISAVE.GT.99)GO TO 351
      IF(ISAVE.GE.10)GO TO 35
C.........SINGLE CHARACTER
      LBUF(KDF+6) = 32*256 + 32
      LBUF(KDF+7)  = ((ISAVE+48)*256) + 20
      GO TO 37
C.........DOUBLE CHARACTER
35    IS=ISAVE/10
      LBUF(KDF+6)=400B*256+600B+IS
      ISAVE=ISAVE-IS*10
      LBUF(KDF+7)=((ISAVE+600)*256)+20
      GO TO 37
C.........TRIPLE CHARACTER
351   CONTINUE
      IS=ISAVE/10
      IS2=IS-((IS/10)*10)
```

```
      IS3=ISAVE-IS*10
      LBUF(KDF+6)=51B*256+60B+IS2
      LBUF(KDF+7)=(50B+IS3)*256+20
      GO TO 37
365   WRITE(1,1000)ISTAT
      CALL START(40,LBUF,LBUF)
1000  FORMAT("RULED ERROR",I5)
   36 LBUF(KDF+6)  = 741/4B
      LBUF(KDF+7) = 74024B
   37 LBUF(4) = KDF
      LBUF(KDF-1) = 0
      ISAVE = 0
      GO TO 31
   38 N = 3
      LBUF(LDF) = 175001B
      GO TO 31
39    IF(LRL,EQ,0)GO TO 31
      CALL STOP
      RETURN
C.....4) RULE TABLE NOT FOUND
   40 IDF = 4
      LBUF(1) = IDF
      LBUF(3) = 10000B
      CALL READ(LBUF,IDF)
      CALL DISCH(-600,400,4,0,ITEX7,LBUF,IDF,NLB)
      LBUF(2) = IDF
      CALL DISCH(-500,-400,4,0,ITEX9,LBUF,IDF,NLB)
      GO TO 22
C.....5) EXIT
50    IDF=4
      LBUF(1)=IDF
      LBUF(3)=100000B
      CALL READ(LBUF,IDF)
      CALL DISCH(-240,-800,4,3,ITE10,LBUF,IDF,NLB)
      LBUF(2)=IDF
      CALL DTHIT(ISWD,LX,LY)
      CALL CROSS (LX,LY,IDF,LBUF)
      IDF=IDF+10
      LBUF(IDF)=100000B
      CALL START(40,LBUF,LBUF)
      JB1=JFLD(1,1)
      CALL CROSS (LX,LY,0,LBUF)
      ISTAT=-2
      IF(JLDF(JB1,ISWD),EQ,0)GO TO 51
      IF(IXMIN(3),GT,LX,OR,IXMAX(3),LT,LX)GO TO 51
      IF(IYMIN(3),GT,LY,OR,IYMAX(3),LT,LY)GO TO 51
      ISTAT=-1
51    RETURN
      END
      ENDS


FTN4
      SUBROUTINE DRAWI(IIX,IIY,IORG,IMGNO,ISTAT)
C.....DEC. 7, 1972
C.....MARGARET ROWAN
C.....PROGRAM: MARK
C.....VERSION 2.0
C.....ROUTINE TO CREATE DISPLAY INSTRUCTIONS TO
C.....DRAW A PIECE
      DIMENSION NSLX(10),NSLY(10),IBUF(128),NAMG(3)
      COMMON MOUT,LABF(3),LABR(3),NOSTY,INSTY(31),JBS(15),NAMST(3),
     1 MKUP(5),NSCMD,JBM(20),NAMMK(3),JBG(4),NAMGO(3),JBP(10),
     2 IDMK(5),IODF,NPEC,NSMD,FACTR,NSIZ,ISIZ(42),WIDTH,EFF,SX,SY,KSTL,
```

```
     3 YARDS,NAP,KINT,KFLG,KINC,KSIZ,KXMN,KYMN,KANG,KXMX,KYMX,KOPT,
     4 KAD,KADP,KOF,CSMX,MKWD,CSMN,NREE,IMGP,KREL,ITYPE,
     5 NOIMG,IOFSZ,INGY,IMGE,IMGTC,IMKO1,IMKO2,IMENU,IOXP,IOYP,
     6 IXC,IYC,IXL,IXR,IYU,IYL,CS,MDELX,MDELY,JFLG,MLW,MUP,ISLX,ISLY,
     7 TOTAL,MYL,XPC,YPC,IMGLP,IMESG,IEOG,ISOFF,SOFF,KSLM,
     8 IMGPS,MAXOF,KSLV,KTC,IOPTN,IAREA,ISIZE,KCSR,KYES,KPSIZ,KPLL,
     9 IT1,IT2,IT3,IT4,IT5,IT6,IT7,IT8,IT9,IT10,
     A IOFSET,KPLM,IFLP,ICOST,IMGPK,
     B TLML,T2,T3,T4,T5,T6,T7,T8,T9,ILOOP,INSCP,
     C JSTY,JFG,JMG,JSMO,JFT,
     D ILPO,IMU,IBO,IOV,IBR,IBL,ITCW,ITCCW,
     E IRCW,IRCCW,ISL,IRE,IFL,IPK,ICOM,
     F IOOM(1)
      DATA NAMG/2HCW,2HEX,2H  /
       IF(IMGNO.EQ.0) RETURN
       NN = 0
       IF(JLOF(JBM(15),IOPTN).NE.0)NN=NN+3
       IF(JLOF(JBM(15),IOOM(KFLG+NPEC+IMGNO)).NE.0)NN=NN+4
       IFTT = 0
       ISTAT = 0
       INDSL = 1
       IRT = 64
       IRP = 1
       IIPK = 0
       KAO1 = KADP+IMGNO
       IEND=0
C.....IN DISPLAY FILE?
       IF(IABS(IOOM(KAO1)).EQ.0) GO TO 10
C.....CURRENTLY IN FILE MUST BE REDRAWN DUE TO ROTATION.
C.....SAVE BOUNDS
     5 IEND=IABS(IOOM(KAO1+1))
       CALL INDSP(KOPT,IOOM,IEND-1)
       GO TO 20
    10 IOOM(KAO1)=KOPT
       IFTT = 1
       IRT=45
       IRP = 5
       IIPK = 1
    20 CALL LITE(0,KAO1)
C.....SET DSPLY FLAG FOR MINIMUM MOVES
    30 IF(ISLY.NE.0) IOOM(KSTL+IMGNO) = 9999
       IF(ISLX.NE.0) IOOM(KPLL+IMGNO) = 9999
    35 KOPT = IABS(IOOM(KAO1))
C.....SLAVE PIECE?
       IF(JLOF(JBM(20),IOOM(KFLG+IMGNO)),EQ.1) GO TO 1010
       IOX = KOPT-1
       CALL STEP (IRT-1,2047,1,IIPK)
       IF(IFTT.EQ.0) CALL STAT
C.....SET BLOCK FLAG
       IB = 0
       IF(ISLX*ISLY.NE.0.AND.JLOF(JBM(15),IOOM(KFLG+IMGNO)).EQ.1) IB=1
C.....DATA IS IN CORE ON SMALL PIECE LOOP
       IF(ILOOP.GT.1) GO TO 71
C.....READ GRAIN ANGLE IN FROM "CHEK"
C.....COMPUTE WHAT SECTOR TO READ AND THE RELATIVE WORD
       N = 2*((MAXOF+ICOM)/128)+2+(IMGNO*3)/128
       M = MOD(IMGNO*3,128)
       IF(M.EQ.0) M=128
       CALL EXEC (14,66,IBUF,128,NAMG,N)
       ANGG = IBUF(M)*.00174533
       SG = SIN(ANGG)
       CG = COS(ANGG)
       MSCP = IOOM(KINT+IMGNO)
C.....BUFFER IN DATA
    70 CALL EXEC (14,66,IBUF,128,NAMGO,MSCP)
```

```
   71 ISCP = 1
      NSCP = JLOF(JBP(5),IBUF)
      INSCP = NSCP
      IMIRO = JLOF(JBP(3),IBUF   )
      IFLIP = JLOF(JBM(11),IDUM(KFLG+IMGNO))
      IF(IFTT.EQ.1) GO TO 75
      KOPT = KOPT+NN
      GO TO 80
C.....CREATE HEADER
   75 IROT = JLOF(JBP(4),IBUF   )
      CALL JSTF(IROT,JRT,IDUM(KFLG+IMGNO))
      CALL HEADI(IIX,IIY,IORG,IMGNO)
C.....ROTATION ANGLE
   80 ANG = MOD(IDUM(KANG+IMGNO),3600)*.0017453329
      S = SIN(ANG)
      C = COS(ANG)
      X = 0.0
      Y = 0.0
      JMIRO = 0
      N = 3
      CALL DSPLY(0,0,-3)
      NOPTS = 0
      MAXX = 0
      MINX = 0
      MAXY = 0
      MINY = 0
      L1 = 1
      L2 = NSCP
      L3 = 1
      L4 = 3
      L5 = 126
      L6 = 3
      IN4I = 0
   90 DO 800 J=L1,L2,L3
      IF(JMIRO.NE.0.AND.J.NE.L1) L4=126
      DO 700 K=L4,L5,L6
      X1 = X
      Y1 = Y
      NP1 = NPTYP
      X = IBUF(K+1)*.001*XPC
      Y = IBUF(K+2)*.001*YPC
      NOPTS = NOPTS+1
      NPTYP = JLOF(JBP(9),IBUF(K))+1
      IF(JMIRO.NE.0) Y=-Y
C.....ROTATE THE GRAIN ANGLE
      CALL XFORM(X,Y,X,Y,SG,CG)
C.....FLIP BY STYLE DEFINITION
      IF(IFLIP.EQ.1) Y = -Y
C.....RATATE THE ROTATION ANGLE
      CALL XFORM(X,Y,X,Y,S,C)
      CALL SCALE (NX,NY,X,Y,2)
      GO TO(100,100,300,400,300,700,700,700,100,300,800,100),NPTYP
C.....GRADE POINT AND INTERMEDIATE POINT
  100 NOPTS = 0
      OLDSL = SLOP
      CALL DSPLY(NX,NY,N)
      MAXX = MAX0(MAXX,NX)
      MINX = MIN0(MINX,NX)
      MAXY = MAX0(MAXY,NY)
      MINY = MIN0(MINY,NY)
      N = 2
      GO TO 700
C.....POINTS TO SKIP
  300 X = X1
      Y = Y1
```

```
      NPTYP = NP1
      GO TO 700
C.....STRIPE LINE
  400 IF(NP1.NE.4) GO TO 700
      CALL SCALE (MX,MY,X1,Y1,2)
C.....SAVE STRIPE OR PLAID
      IF(ISLY.EQ.0) GO TO 410
      IF(IABS(NX-MX).LE.IABS(NY-MY)) GO TO 410
      NSLX(INDSL) = MX
      NSLX(INDSL+1) = NX
      NSLY(INDSL) = MY
      NSLY(INDSL+1) = NY
      INDSL = INDSL+2
      IDUM(KSTL+IMGNO)=(NY+MY)/2
      GO TO 420
  410 IF(ISLX.EQ.0) GO TO 420
      IF(IABS(NX-MX).GE.IABS(NY-MY)) GO TO 420
      NSLX(INDSL) = MX
      NSLX(INDSL+1) = NX
      NSLY(INDSL) = MY
      NSLY(INDSL+1) = NY
      INDSL = INDSL+2
      IDUM(KPLL+IMGNO) = (NX+MX)/2
C.....FLAG TO SKIP EVERY OTHER STRIPE POINT
  420 NPTYP = 0
      IF(INDSL.LE.14) GO TO 700
      INDSL = 0
      CALL ERR(19,1)
  700 CONTINUE
      ITMP = MSCP+J-INMI
      CALL EXEC (14,06,IBUF,126,NAMGO,ITMP)
  800 CONTINUE
      GO TO 850
  830 IF(IMIRO.EQ.0) GO TO 850
      L1 = NSCP
      IF(L1.LT.1) L1=1
      L2 = 1
      L3 = -1
      L4 = K-3
      L5 = 3
      L6 = -3
      INMI = 2
      JMIRO = 1
      GO TO 90
C.....DETERMINE MOVE FROM IMAGE ORIGIN TO PIECE ORIGIN.
  850 IX = 0
      IY = 0
      IF(IORG.EQ.0) GO TO 920
      GO TO (860,880,890,910),IORG
  860 IX = MINX
  870 IY = MINY
      GO TO 920
  880 IX = MAXX
      GO TO 870
  890 IX = MAXX
  900 IY = MAXY
      GO TO 920
  910 IX = MINX
      GO TO 900
  920 IND = IABS(IDUM(KAD1))+4
      N=1
      M=168
      IF(NN.EQ.6) GO TO 930
      N=0
      M=1
```

```
  930 IDUM(IND)=(-IX)*16+N
      IDUM(IND+1)=(-IY)*16+M
C.....ADJUST MIN & MAX VALUES.
  940 IXMN = KXMN+IMGND
      IXMX = KXMX+IMGND
      IYMN = KYMN+IMGND
      IYMX = KYMX+IMGND
C.....CHECK FOR BOUNDS OF MARKER
      D = FLOAT(IXC)+FLOAT(IIX-IX+MAXX)
      IF(D.LT.-.5E-M) GO TO 950
      IDUM(IXMX) = MIN1(D,32.5E-M,M)
      IDUM(IXMN) = IDUM(IXMX)-MAXX+MINX
      GO TO 960
  950 D = FLOAT(IXC)+FLOAT(IIX-IX+MINX)
      IDUM(IXMN) = MAX1(D,-32.5E-M,0)
      IDUM(IXMX) = IDUM(IXMN)-MINX+MAXX
  960 CONTINUE
      M = IYC+IIY-IY
      IDUM(IYMX) = M+MAXY
      IDUM(IYMN) = M+MINY
C.....STORE DIMENSIONS OF PIECE
      XDELX = IDUM(IXMX)-IDUM(IXMN)
      XDELY = IDUM(IYMX)-IDUM(IYMN)
C.....ADJUST STRIPE AND PLAID INCREMENTS RELATIVE TO BOX
      IF(ISLY.EQ.0.OR.IDUM(KSTL+IMGND).EQ.9999) GO TO 970
      IDUM(KSTL+IMGND) = IDUM(KSTL+IMGND)-(MAXY+MINY)/2
  970 IF(ISLX.EQ.0.OR.IDUM(KPLL+IMGND).EQ.9999) GO TO 975
      IDUM(KPLL+IMGND) = IDUM(KPLL+IMGND)-(MAXX+MINX)/2
  975 IF(IS.NE.0) GO TO 980
      IF(INDSL.EQ.1) GO TO 985
C.....DRAW STRIPE LINES
C.....THE ZERO WORD FLAGS END OF DATA FOR GENX
      IF(KOPT.LT.IEND.OR.IEND.EQ.0) IDUM(KOPT) = 0
      KOPT = KOPT+1
      INDSL = INDSL-1
      DO 982 I=1,INDSL,2
      CALL DSPLY(NSLX(I),NSLY(I),3)
      CALL DSPLY(NSLX(I+1),NSLY(I+1),2)
  982 CONTINUE
      GO TO 985
C.....DRAW BLOCK
  984 IDUM(IXMX) = IDUM(IXMX)+ISLX
      IF(SY.EQ.99.0) GO TO 981
      IDUM(IYMX) = IDUM(IYMX)+ISLY
  981 IXMX = MAXX
      IXMN = MINX-ISLX
      IYMX = MAXY+ISLY
      IF(SY.EQ.99.0) IYMX = MAXY
      IYMN = MINY
C.....THE ZERO WORD FLAGS END OF DATA FOR GENX
      IDUM(KOPT) = 0
      KOPT = KOPT+1
      CALL DSPLY (IXMN,IYMN,3)
      CALL DSPLY (IXMN,IYMX,2)
      CALL DSPLY (IXMX,IYMX,2)
      CALL DSPLY (IXMX,IYMN,2)
      CALL DSPLY (IXMN,IYMN,2)
  985 IF(IEND.NE.0) GO TO 995
      IF(KOPT.GT.IDFSZ) CALL ERR(4,0)
      CALL LITE(1,KAD1)
  990 RETURN
C.....REDRAW OF ROTATED PIECE. CHECK FOR OVERFLOW.
  995 IF(KOPT.EQ.IEND) GO TO 990
```

```
      IF(KOPT.GT.IEND) GO TO 1005
C.....LESS ROOM WAS REQUIRED, ZERO WHAT WAS LEFT.
      IEND=IEND+1
      DO 1000 I=KOPT,IEND
 1000 IDUM(I)=0
      GO TO 999
C.....NOT ENOUGH ROOM, EXPAND THE DISPLAY FILE AND TRY AGAIN.
 1005 IF(KOPT.GT.IDFS7) CALL ERR(4,0)
      N=KOPT-IEND
      CALL EXPAN(1HSND,N)
      GO TO 5
C..........PUT ONE NOP IN FOR ASLAVE
 1010 IDUM(KOPT)=0
      KOPT=KOPT+1
      CALL LITE(1,KAD1)
      RETURN
      END
      ENDS
FTN4
      PROGRAM DRAWM,3
C
C     VERSION 2.0
C     EDITED 02 MAR 73
C     RONALD GORDON
C
C     BOX ON PUNCH HOLE
C     EXPANDED GRADE FILE
C     SCALE ON XYNETICS 1100 PRINTED
C     GRAIN ANGLE IN TENTHS
C     STRIPE = 99 OPTION
C     USE STRIPE LINE IF NO GRAIN LINE
C     NO PAPER SLEW TO CLEAR TABLE
C     VARIABLE GAP SPACE  (ZERO MEANS NO HEADING)
C     SLAVE / MASTER PIECES RECOGNIZED
C     NOTCH HEAD OPTION
C     NESTED PLOT LOOK BEHIND
C     VERSION 2 FILE CHANGES
C     SIZE PRINT OPTION
C     SPLICE MARKERS OPTION
C     NEW POINT TYPES 8 & 11
C     NEW PIECE MESAGES HANDELING
C     PAPER SLEW FOR XYNETICS 2000
C     SCALE FOR XYNETICS 1100
C     FOR SERIES 2000 XYNETICS PLOTTER AND DRUM
C     PLOTS ON FIXED DISK FOR NOW
C     NEW MESSAGES HANDLER
C     FIX FOR STYLE ARRAY
C     CLEAN UP OPTIONS I/O
C     OPTION TO DRAW TICKS OUT
C     ANNOTATION ON HEAD OF PLOT
C     PLOT OFFSET (0,TICK)
C     STRIPE OFFSET SUPPORTED FOR BERKSHIRE.
C     512 PIECES NOW IN MARKER
C     STYLE LETTERING OPTION BIT 2
C     BUNDLE LETTERING OPTION BIT 0
C
      REAL LBOUND,LLIMIT
      REAL MARGIN
      REAL MINSIZ
      INTEGER OFFON
      INTEGER OPTION
      LOGICAL PLAID
      LOGICAL NTHOPT, STYOPT, LETOPT
      LOGICAL TICOPT
```

```
      LOGICAL XY1OPT, XY2OPT
      LOGICAL LOCKED
      LOGICAL SIZOPT
      LOGICAL NOS12
      LOGICAL BOXOPT
      LOGICAL SLAVE
      LOGICAL STRIPE
C**
C**   SPACE FOR 512 PIECES IN A MARKER
C**
      DIMENSION INCORE(8,512), JNCORE(3)
      DIMENSION KEEPER(512)
      DIMENSION INSTY(5,31)
      DIMENSION MESBIT(512)
C**
C**   SPACE FOR 12 SECTORS OF PIECE DATA
C**
      DIMENSION MASTER(1536)
      DIMENSION ISPLIC(2,15)
      DIMENSION VAL(1)
      DIMENSION IBUFR(128)
      DIMENSION NAME(5), MAME(6)
      INTEGER BITPOS(1)
      INTEGER OPTNS(3)
      INTEGER SECTR(3), GRADE(3), MARKR(3)
      INTEGER STYLE(3)
      EQUIVALENCE (PIECLO,IBUFR(4))
      EQUIVALENCE (VAL(1), IBUFR(1))
      EQUIVALENCE (INCORE(1,1), JNCORE(3))
      EQUIVALENCE (OPTION,IBUFR(2))
C
      COMMON LOUT,MTYPE,ORIGX,ORIGY,LDIRC,XPEND,AXOLD,AYOLD,IPNF(3),
     +      IPNR(3),XMOX,YMOX,SCAL,XLAST,YLAST,LASTP,LASTA,LASTT,
     +      IPLT(3),JTYPE,JOB,JPOS,JTOOL,JHOME,JIZ,SPACE
     +      ,IOPTN,NPLAT,J8W,J81,J82,J83,J84
     +      ,AXCNST
C
C     THIS COMMON AREA IS ONLY TO CONSERVE DISK SPACE
C
      COMMON INCORE, KEEPER, INSTY, MESBIT, MASTER, ISPLIC, IBUFR,
     +      NAME, MAME, BITPOS
C
      DATA SECTR /2HSE,2HCT,2HR /,
     +      MARKR /2HMA,2HRK,2HR /,
     +      GRADE /2HGR,2HAD,2HE /
      DATA OPTNS /2HOP,2HTN,2HS /
      DATA STYLE /2HST,2HYL,2HE /
      DATA XZERO,YZERO /2*0.0/, XMSG /0.0/
      DATA RASTER /0.00363/, FOURK /4096.0/, RINCHS /15.0/
      DATA SEGSIZ /50.0/, SEGMRG /20.0/
      DATA SPLSIZ /3.0/
C
C     SYMSIZ - HEIGHT OF SYMBOLS IN INCHES
C     SIZLIN - PERCENT INCREASE FROM SYMSIZ TO HEIGHT PER LINE
C     TICK - SIZE OF BOARDER TICK MARKS FOR PLAID, STRIPE
C     CROS - SIZE OF PEN STROKE FOR PUNCH HOLE STUFF
C     NTHSIZ - SIZE OF HEAD ON TICK MARKS IF OPTION SPECIFIED
C
      DATA SYMSIZ /0.25/
      DATA SIZLIN /1.35/
      DATA TICK /1.0/
      DATA CROS /0.1/
      DATA NTHSIZ /0.25/
```

```
C                                       RIGHT AND LEFT HALF WORDS
      MASKL=JFLD(15,8)
      MASKR=JFLD(7,8)
C                                       NSECT IN MARKR
      MASK1=JFLD(5,6)
C                                       NSIZES IN MARKR
      MASK2=JFLD(4,5)
C                                       NSECT IN GRADE
      MASK3=JFLD(8,9)
C                                       SECTORS/PIECE
      MASK4=JFLD(12,4)
C                                       RULE IN MARKR
      MASK5=JFLD(9,7)
C                                       STYLE NUMBER IN MARKR
      MASK6=JFLD(15,11)
C                                       POINT TYPE
      MASK7=JFLD(15,4)
C                                       BLOCK BIT IN MARKR
      MASK8=JFLD(0,1)
C                                       MASTER / SLAVE BIT
      MASK9=JFLD(2,1)
C       #                                FLIP FLAG IN MARKR
      MASK11=JFLD(15,1)
C                                       MIRROR FLAG IN GRADE
      MASK12=JFLD(5,1)
C                                       NUMBER OF PIECES, STYLES IN MARKR
      MASK13=JFLD(15,12)
      MASK14=JFLD(15,5)
C                                       NUMBER OF SPLICE MARKS, MAN ID
      MASK15=JFLD(11,4)
      MASK16=JFLD(6,6)
C                                       NPIEC IN STYLE
      MASK17=JFLD(3,4)
C                                       MASKS FOR GRADE SEARCH
      MASK19=JFLD(7,7)
C                                       TO DECODE MESAGE BITS
      DO 50 I=1,10,1
   50 BITPOS(I)=JFLD(I-1,1)
C
C     PICK OUT SYSTEM FLAGS
C
      CALL DDSC (0)
      CALL EXEC (14,1028,IBUFR,11,OPTNS,0)
      LETOPT=JLOF(JFLD(0,1),OPTION) .EQ. 1
      LOCKED=JLOF(JFLD(1,1),OPTION) .EQ. 1
      STYOPT=JLOF(JFLD(2,1),OPTION) .EQ. 1
      XY1OPT=JLOF(JFLD(3,1),OPTION) .EQ. 1
      NTHOPT=JLOF(JFLD(4,1),OPTION) .EQ. 1
      TICOPT=JLOF(JFLD(5,1),OPTION) .EQ. 1
      XY2OPT=JLOF(JFLD(6,1),OPTION) .EQ. 1
      BOXOPT=JLOF(JFLD(7,1),OPTION) .EQ. 1
      SIZOPT=JLOF(JFLD(8,1),OPTION) .EQ. 1
      MDS12 = JLOF(JFLD(9,1),OPTION) .EQ. 1

C
C     PAPER CONSTANTS IN OPTNS TOO.
C
      YMAX=1440.0
      YMAX=FLOAT(IBUFR(12))*0.01
      IF (IBUFR(10).EQ.0) YMAX=72.0
      XYCNST=FLOAT(IBUFR(11))*0.01
      IF (IBUFR(11).EQ.0) XYCNST=71.04
C
C     INITIALIZATION FOR PLOT ROUTINES
```

```
C
      IPLT(1)=2HPL
      IPLT(2)=2HOT
      IPLT(3)=2HP
      LOUT=6
      MTYPE=2HMA
      JTYPE=JFLD(15,3)
      JOB=JFLD(12,1)
      JPOS=JFLD(15,1)
      JTOOL=JFLD(14,2)
      JHOME=JFLD(12,1)
      JIZ=JFLD(11,12)
      CALL UOSC(0)
      CALL EXEC (14,1028,IBUFR,128,IPLT,0)
      L=5
      IF (XY20PT) GO TO 60
      ORIGX=0
      ORIGY=0
      AXOLD=0
      AYOLD=0
      XPEND=0
      GO TO 70
   60 CONTINUE
      ORIGX=VAL(L+1)
      ORIGY=VAL(L+2)
      AXOLD=VAL(L+3)
      AYOLD=VAL(L+4)
      XPEND=VAL(L+5)
   70 LDIRC=1
      LASTT=1
      SCAL=1.0
      XLAST=AXOLD
      YLAST=AYOLD
C
C     PICK UP MARKER NAME
C
      CALL UOSC (0)
      CALL EXEC (14,1028,IQWORD,1,SECTR,0)
      DO 110 I=1,5,1
      IQWORD=IQWORD+1
  110 CALL RWST (IQWORD,NAME(I))
C
C     READ IN MARKER FILE DATA
C     CALL DIREC (10,1B,NAME,INTNO,NSCTR,ISTAT)
      IF (ISTAT.EQ.0) GO TO 113
      WRITE (LOUT,1100) NAME
      CALL FINI (2)
  113 CALL EXEC (14,1028,IBUFR,128,MARKR,NSCTR)
      NSECT=JLOF(MASK1,IBUFR)
      NPIEL=JLOF(MASK13,IBUFR)
      NSTYL=JLOF(MASK14,IBUFR(2))
      NSPLIC=JLOF(MASK15,IBUFR(2))
      MANID=JLOF(MASK16,IBUFR(2))
      RMULT=RFIELD
      RK=FLOAT(IBUFR(6))*0.01
      RL=(FLOAT(IBUFR(7))+32600.0)*RMULT
      DEF=FLOAT(IBUFR(9))*0.01
C
C     LENGTH FOR LOCKED MARKERS IS IN IBUFR(128)
C
      IF (LOCKED .AND. IBUFR(128).NE.0)
     +     RL=(FLOAT(IBUFR(128))+32600.0)*RMULT
C
C     CHECK ON PAPER WIDTH.
C
```

```
      IF (RW.LE.YMOX .OR. XY1OPT) GO TO 114
      WRITE (1,1700) RW,YMOX
      READ (1,1900) ITEMP
      IF (ITEMP.EQ.2HNO) CALL FINI (2)
C
C     WHEN DRAWING A MARKER WE MUST NOT EXCEED A RESOLUTION OF 1 MARKER
C     UNIT.  HERE WE SET UP GRID.
C
  114 TOL=MULT*FOURK/RINCHS * RASTER
      IF (MOS12) TOL = TOL/2.0
      CALL SGRID (TOL)
      IWO=RW/TOL + 0.5
      IWO=IWO-IWO/2
      HALFW=IWO*TOL
      CALL GRID (RW,RL)
      SUF=FLOAT(IBUFR(10))*0.01
      SS=FLOAT(IBUFR(11))*0.01
      PS=FLOAT(IBUFR(12))*0.01
C     Y IS DUMMY
      CALL GRID (SUF,Y)
      CALL GRID (SS,PS)
C                                        FOR BLOCK OPTION LATER
      PLAID=IBUFR(11).GT.0 .AND. IBUFR(12).GT.0
      STRIPE=IBUFR(11).GT.0 .OR. IBUFR(12).GT.0
      IX=JLOF(MASKL,IBUFR(13))
      IF (IX.GE.128) IX=IOR(IX,177400B)
      IY=JLOF(MASKR,IBUFR(13))
      IF (IY.GE.128) IY=IOR(IY,177400B)
      XPN=FLOAT(IX)*0.001+1.0
      YPN=FLOAT(IY)*0.001+1.0
C
C     PICK UP REST OF MARKER DEFINITION
C     THIS IS THE LAST TIME WE ARE ON REMOVABLE PLATTER
C     SAVE IBUFR UNTIL ISW=175
C
      LSCTR=NSCTR+NSECT-1
      NSCTR=NSCTR+1
      J=-119
      DO 115 I=NSCTR,LSCTR,1
      J=J+120
      IX=JNCORE(J)
      IY=JNCORE(J+1)
      CALL EXEC (14,1428,JNCORE(J),128,MARKR,I)
      JNCORE(J)=IX
      JNCORE(J+1)=IY
  115 CONTINUE
C
C     UPDATE TICK OPTION
C
      TICOPT=TICOPT .AND. (IBUFR(12).GT.0 .OR. IBUFR(11).GT.0)
C
C     PICK UP SCALE FOR MARKER AND DO XYNETICS 1100 TRICK
C
      IOWORD=INBUFO+1
      CALL MOST(IOWORD,ITEMP)
      SCAL=FLOAT(ITEMP)*0.01
      IF (XY1OPT) SCAL=AMIN1( SCAL, 55.0/RL, 40.0/RW)
      CALL STORE (-1.0,0,0)
      CALL PLOT (0.0,0.0,-99)
      CALL PLOT (0.0,0.0,-3)
C
C     PICK UP GAP SPACE
```

```
C
      IQWORD=IQWORD+1
      CALL RQST (IQWORD,ITEMP)
      MARGIN=FLOAT(ITEMP)*0.01
C
C     PAPER SLEW IF XYNETICS 2000 AND TABLE
C     IS NOT CLEAR.
C
      IF (.NOT.XY2OPT) GO TO 116
      DELTA=MARGIN+XPAND
      IF (DELTA.LT.TOL) GO TO 116
      ITEMP=DELTA*SCAL*XYCNST
      CALL STORE (1,0A000B,ITEMP,0)
C
C     REORIGIN FOR SPLICE AND TICK MARKS
C
  116 IF(NSPLIC.LE.0) GO TO 117
      CALL PLOT (0.0,SPLSIZ,-3)
      YZERO=YZERO-SPLSIZ
      GO TO 118
  117 IF(.NOT.TICOPT) GO TO 118
      CALL PLOT (0.0,TICK,-3)
      YZERO=YZERO-TICK
C
C     OUTPUT HEADER IF REQUIRED
C
  118 IF (MARGIN.LT.TOL) GO TO 140
      IQWORD=IQWORD+1
      CALL RQST (IQWORD,NMSGS)
      LETTER=JLOF (MASKL,NMSGS)
      IF(LETTER.EQ.4HA) LETTER=1H1B
      NMSGS=JLOF(MASKR,NMSGS)
      X=SYMSIZ
      Y=0.0
      CALL SYMB (X,Y,SYMSIZ,NAME,90.0,10)
      X=X+SYMSIZ*SIZLIN
      IF (NMSGS.EQ.0) GO TO 130
      DO 120 I=1,NMSGS,1
      IQWORD=IQWORD+1
      CALL RQST (IQWORD,ITEMP)
      ITEMP=ITEMP/100
      IF (ITEMP.GT.10) CALL MESAG (ITEMP,X,Y,SYMSIZ,90.0)
      X=X+SYMSIZ*SIZLIN
  120 CONTINUE
  130 CONTINUE
C
C     ANNOTATE PLOT WITH LENGTH, EFF, ETC.
C     FOR XYNETICS 1100 POST SCALE, ALSO IF SCALE .NE. 1.0
C
      LYD=RL/30.0
      LIN=RL*(LYD*36)
      CALL CODE
      WRITE (KEEPER,1800) RW,LYD,LIN,EFF,SCAL
      I=48
      IF (XY10PT .OR. SCAL.NE.1.0) I=62
      CALL SYMB (X,0.0,SYMSIZ,KEEPER,90.0,I)
      X=X+SYMSIZ*SIZLIN
      X=X+TICK
      XMSG=XMSG+X
      CALL PLOT (X,0.0,-3)
      XZERO=XZERO-X
C
C     OPTION TO DRAW TICK MARKS OUT
```

```
C
  140 ATICK=TICK
      HTICK=TICK*2.5
      IF (.NOT.TICOPT) GO TO 150
      SEGMRG=SEGMRG-TICK
      TICK=-TICK
  150 RBOUND=SEGMRG+SEGSIZ-XMSG
      IF (RBOUND+SEGMRG.GT.RL) RBOUND=RL
      IF (.NOT.XY2OPT) RBOUND=RL
      RLIMIT=0.0
      XPS=0.0
C
C     BUILD DATA VECTORS THROUGH KEEPER.  GRAB MESSAGE BITS
C     FROM STYLE FILE
C
      J=14
      NS=0
      NP=0
      DO 170 I=1,NSTYL,1
      NSIZE=JLOF(MASK2,IBUFR(J))
      ISTYL=JLOF(MASK0,IBUFR(J))
      J=J+(NSIZE-1)/2+2
      CALL DIREC (1423B,INSTY(1,I),ISTYL,ISECT,ISTAT)
      IF (ISTAT.EQ.0) GO TO 155
      WRITE (LOUT,1300) ISTYL
      CALL FINI (2)
  155 CALL EXEC (14,1028,MASTER,128,STYLE,ISECT)
      N=JLOF(MASK17,MASTER)
      M=JLOF(MASKL,MASTER)
      IF (N.EQ.1) GO TO 165
      IWORD=3
      DO 160 IS=ISECT+1,ISECT+N-1,1
      IWORD=IWORD+125
  160 CALL EXEC (14,1028,MASTER(IWORD),128,STYLE,IS)
  165 DO 166 K=1,NSIZE,1
      NS=NS+1
  166 KEEPER(NS)=J
      N= -1
      DO 170 L=1,M,1
      N=N+5
    # DO 170 K=1,NSIZE,1
      NP=NP+1
  170 MESBIT(NP)=MASTER(N)
C
C     CAPTURE SPLICE DATA
C
      IF (NSPLIC.LE.0) GO TO 180
      IF (NSPLIC.EQ.0) GO TO 180
      J=76
      DO 175 I=1,NSPLIC,1
      ISPLIC(1,I)=IBUFR(J)
      ISPLIC(2,I)=IBUFR(J+1)
  175 J=J+2
  180 CONTINUE
C
C     DO LEFT BOARDER TOP TO BOTTOM
C
      IF (SS.GT.RW) SS=0.0
      CALL PLOT (0.0,RW,3)
      IF (SS.LT.TOL) GO TO 195
      NSS=(RW+SOF)/SS
      Y=RW-SOF
      DO 190 I=1,NSS,1
      CALL PLOT (0.0,Y,2)
```

```
      CALL PLOT (TICK,Y,2)
      CALL PLOT (0.0,Y,2)
  190 Y=Y-SS
      YSS=Y
  195 CALL PLOT (0.0,0.0,2)
C
C     SORT PIECES FOR POSITIONING
C     ABSOLUTE X POSITION IS USED FOR SEGMENTATION
C
      L=NPIEC-1
      DO 220 I=1,L,1
      M=I+1
      DO 220 J=M,NPIEC,1
      IF (INCORE(6,I).LE.INCORE(6,J)) GO TO 220
      DO 210 K=1,8,1
      ITEMP=INCORE(K,I)
      INCORE(K,I)=INCORE(K,J)
  210 INCORE(K,J)=ITEMP
      ITEMP=MESBIT(I)
      MESBIT(I)=MESBIT(J)
      MESBIT(J)=ITEMP
  220 CONTINUE
      IF (ISSW(7).LT.0) WRITE (6,1500)   ((INCORE(I,J),
     +     I=1,8,1), MESBIT(J), J=1,NPIEC,1)
C
C
C     LOOP TO DRAW PIECE *IPIEC*
C
      DO 800 IPIEC=1,NPIEC,1
      XP=(FLOAT(INCORE(6,IPIEC))+32600.0)*RMULT
      YP=FLOAT(INCORE(7,IPIEC))*RMULT+HALFW
      IF (YP.GE.[illegible]) GO TO 800
      NAME(6)=INCORE(1,IPIEC)
      CALL DIREC (32046,NAME,INTNO,NSCTR,ISTAT)
      IF (ISTAT.EQ.0) GO TO 410
  400 WRITE (LOUT,1200)
      CALL FINI (2)
  410 CALL EXEC (14,1028,IBUFR,128,GRADE,NSCTR)
      JANG=IBUFR(2)
C
C     EXTENDED GRADE FILE FORMULAE USED
C
      NSC=JLOF(MASK3,IBUFR)
      LSCTR=NSCTR+NSC
      NSCP=JLOF(MASK4,IBUFR)
      NSZ=NSC*12/(1+NSCP*12)
      MSCP=(NSZ-1)/12+NSCTR+1
      JDX=0
      MINC=INCORE(4,IPIEC)
      IRUL=JLOF(MASK5,INCORE(3,IPIEC))
      ISIZ=JLOF(MASK8,INCORE(5,IPIEC))
      ITEMP=0
      CALL JSTF (ISIZ,MASKL,ITEMP)
      CALL JSTF (IRUL,MASK19,ITEMP)
  415 DO 420 I=3,119,10
      IF (ITEMP.EQ.IBUFR(I) .AND. MINC.EQ.IBUFR(I+9)) GO TO 430
      JDX=JDX+1
  420 CONTINUE
      NSCTR=NSCTR+1
      IF (NSCTR.GE.LSCTR) GO TO 400
      CALL EXEC (14,1028,IBUFR,128,GRADE,NSCTR)
      GO TO 415
  430 NSCTR=MSCP+JDX*NSCP
C
```

```
C       READ IN ALL PIECE DATA
C
        CALL EXEC (14,1028,MASTER,128*NSCP,GRADE,NSCTR)
C
C       IF PIECE HAS NO GRAIN LINE USE STRIPE LINE
C
        IF (JANG.NE.3600) GO TO 436
        I=0
        OFFON=1
  432   I=I+3
        IF (MOD(I,128).LE.1) I=I+2
        ITYPE=JLOF(MASK7,MASTER(I))
        IF (ITYPE.EQ.10) GO TO 436
        IF (ITYPE.NE.3) GO TO 432
        IBUFR(OFFON)=MASTER(I+1)
        IBUFR(OFFON+1)=MASTER(I+2)
        OFFON=OFFON+2
        IF (OFFON.LE.3) GO TO 432
        RAD=ATAN2((FLOAT(IBUFR(4))-FLOAT(IBUFR(2)))*0.01,
     +            (FLOAT(IBUFR(3))-FLOAT(IBUFR(1)))*0.01)
        JANG = IRND(RAD*572.95779)
  436   RAD=FLOAT(JANG)*0.0017453293
  437   SING=SIN(RAD)
        COSG=COS(RAD)
C
C                                          MASTER OR SLAVE?
        SLAVE=JLOF(MASK9,INCORE(3,IPIEC)) .EQ. 1
        CALL SIZEP (2,IRUL,NAME,ISIZ)
        IF (NAME.GT.0) GO TO 440
        DO 438 M=1,3,1
  438   NAME(M)=2H
C                                          GET FIRST FLIP FROM MARKR
  440   KSIGN= +1
        IF (JLOF(MASK11,INCORE(3,IPIEC)).EQ.1) KSIGN= -1
C                                          GET MIRROR FLAG FROM GRADE
        MIRROR=0
        IF (JLOF(MASK12,MASTER).EQ.1) MIRROR=1
        IANG=INCORE(6,IPIEC)
C                                            2ND FLIP FROM MARKR
        JSIGN= +1
        IF(IABS(IANG).GE.3000) JSIGN = -1
        IANG = MOD(IANG,3600)
        RAD = FLOAT(IANG)*0.0017453293
        SINA=SIN(RAD)
        COSA=COS(RAD)
        MSIGN=-1
        XMIN=0.0
        YMIN=0.0
        XMAX=0.0
        YMAX=0.0
        I=0
        M=-3
        DO 520 K=0,MIRROR,1
        MSIGN=-MSIGN
        M=-M
  500   I=I+M
        IF (MOD(I,128).LE.1) I=I+ISIGN(2,M)
        IF (I.LE.0) GO TO 520
        ITYPE=JLOF(MASK7,MASTER(I))
        IF (ITYPE.EQ.1) GO TO 510
        IF (ITYPE.EQ.0) GO TO 510
        IF (ITYPE.EQ.10) GO TO 520
        IF (ITYPE.NE.11) GO TO 500
  510   X=FLOAT(MASTER(I+1))*0.001 * XPN
        Y=FLOAT(MASTER(I+2)*MSIGN)*0.001 * YPN
```

```
      IF(IANG.NE.0) CALL XFORM(X,Y,X,Y,SING,COSG)
      Y=Y*FLOAT(KSIGN)
      IF (IANG.NE.0) CALL XFORM (X,Y,X,Y,SINA,COSA)
      Y=Y*FLOAT(JSIGN)
      IF (X.LT.XMIN) XMIN=X
      IF (Y.LT.YMIN) YMIN=Y
      IF (X.GT.XMAX) XMAX=X
      IF (Y.GT.YMAX) YMAX=Y
      GO TO 540
  520 CONTINUE
      CALL GRID (XMIN,YMIN)
      XM=XP-XMIN
      YM=YP-YMIN
C
C     PAGING ALGORITHM MOSTLY HERE.  ALSO DOES BOARDER TOP
C     AND BOTTOM AND RIGHT END.
C
C                                        FORCE OUT BOARDER FOR 1ST PIECE
      IF(RLIMIT.LT.TOL) GO TO 640
C                                        MIDPOINT OUT IN RIGHT FIELD?
      IF (XM.LE.RBOUND) GO TO 695
      CALL PLOT (RBOUND,-HTICK,3)
      CALL PLOT (RBOUND,-ATICK,2)
      CALL PLOT (RBOUND-ATICK,-HTICK,2)
      CALL PLOT (RBOUND,-HTICK,2)
      LBOUND=RBOUND
      RBOUND=RBOUND+SEGSIZ
      IF (RBOUND+SEGMRG.GT.RL) RBOUND=RL
C
C     PAPER SLEW AND REORIGIN
C
      CALL PLOT (LBOUND,-HTICK,3)
      OLAST=XLAST
      DELTA=SEGSIZ
      XLAST=(LBOUND+DELTA)
      XZERO=XZERO+XLAST-OLAST
      ITEMP=DELTA*SCAL*XYCONST
      CALL STORE (1,60000,ITEMP,0)
      CALL PLOT (LBOUND+ATICK,-HTICK,2)
      CALL PLOT (LBOUND,0.0,2)
      CALL PLOT (LBOUND,-HTICK,2)
  640 LLIMIT=RLIMIT
      RLIMIT=RBOUND+SEGMRG
      IF (RLIMIT.GT.RL) RLIMIT=RL
C                                        SPLICE MARKS IF NEEDED
  650 IF (ISPCNT.GT.NSPLIC) GO TO 655
      XR=(FLOAT(ISPLIC(2,ISPCNT))+32600.0)*RMULT
      IF (XR.GT.RLIMIT) GO TO 655
      XL=(FLOAT(ISPLIC(1,ISPCNT))+32600.0)*RMULT
      CALL PLOT (XL,0.0,3)
      CALL PLOT (XR,0.0,2)
      CALL PLOT (XR,-SPLSIZ,2)
      CALL PLOT (XL,-SPLSIZ,2)
      CALL PLOT (XL,0.0,2)
      CALL PLOT (XR,-SPLSIZ,2)
      CALL PLOT (XR,0.0,2)
      CALL PLOT (XL,-SPLSIZ,2)
      ISPCNT=ISPCNT+1
      GO TO 650
  655 XPS=0
      CALL PLOT (LLIMIT,0.0,3)
      IF (PS.LT.TOL) GO TO 665
      X=XPS
  660 X=X+PS
      IF (X.GT.RLIMIT) GO TO 685
```

```
      CALL PLOT (X,0.0,2)
      CALL PLOT (X,TICK,2)
      CALL PLOT (X,0.0,2)
      NPS=NPS+1
      GO TO 660
  665 CALL PLOT (RLIMIT,0.0,2)
      XPS=X-PS
C                                        RIGHT LINE
      IF (RL-RLIMIT.GT.TOL) GO TO 680
      IF (SS.LT.TOL) GO TO 675
      Y=YSS
      DO 670 I=1,NSS,1
      Y=Y+SS
      CALL PLOT (RL,Y,2)
      CALL PLOT (RL-TICK,Y,2)
  670 CALL PLOT (RL,Y,2)
  675 CALL PLOT (RL,RW,2)
      GO TO 681
C                                        TOP LINE
  680 CALL PLOT (RLIMIT,RW,3)
  681 IF (NPS.EQ.0) GO TO 690
      DO 685 I=1,NPS,1
      X=X-PS
      CALL PLOT (X,RW,2)
      CALL PLOT (X,RW-TICK,2)
  685 CALL PLOT (X,RW,2)
  690 CALL PLOT (LLIMIT,RW,2)
  695 CONTINUE
C
C     SLAVE LOOP FOR PLOTTING EACH PIECE'S DATA
C
      M=-3
      K=0
      IGPF=0
      NPTYP=0
      X=0.0
      Y=0.0
      IPEN=3
      OFFON=1
      XL=XM
      YL=Y0
      IF (ISSW(7).LT.0) WRITE (6,1000) XMIN,YMIN,XM,Y0,IANG,KSIGN,MIRROR
      MSIGN=-1
      DO 800 I=0,MIRROR,1
      MSIGN=-MSIGN
      M=-M
  710 K=K+4
      IF (IBUF(K,128).LE.1) K=K+ISIGN(2,M)
      IF (K.LE.0) GO TO 800
      X1=X
      Y1=Y
      LPTYP=NPTYP
      X=FLOAT(MASTER(K+1))*0.001 * XPN
      Y=FLOAT(MASTER(K+2)*MSIGN)*0.001 * YPN
      IF(IANG.NE.0) CALL XFORM(X,Y,X,Y,SING,COSG)
      Y=Y*FLOAT(KSIGN)
      NPTYP=JLOF(MASK7,MASTER(K))
      IF (IANG.NE.0) CALL XFORM(X,Y,X,Y,SINA,COSA)
      Y=Y*FLOAT(JSIGN)
      CALL GRID (X,Y)
      X=X+XM
      Y=Y+YM
      JUMP=NPTYP+1
C               0   1   2   3   4   5   6   7   8   9   10  11
      GO TO (720,720,730,735,740,710,710,736,720,760,800,720), JUMP
```

```
C                                          GRADE AND INTERM POINTS
  720 IF (IGPF.EQ.0) IGPF=K
      IF (I.EQ.0) IGPL=K
      CALL PLOT (X,Y,IPEN)
      IPEN=2
      IF (LPTYP.NE.4) GO TO 730
      IF (I.EQ.0) GO TO 730
      IF (K.EQ.IGPF .OR. K.EQ.IGPL) GO TO 730
      CALL PLOT (X1,Y1,2)
      IF (NTHOPT) CALL NOTCH (X,Y,X1,Y1,NTHSIZ)
      CALL PLOT (X,Y,2)
      GO TO 730
C                                          PUNCH HOLE
  734 IPEN=3
      CALL PLOT (X-CROS,Y,3)
      CALL PLOT (X+CROS,Y,2)
      CALL PLOT (X,Y-CROS,3)
      CALL PLOT (X,Y+CROS,2)
      IF (.NOT.BOXOPT) GO TO 730
      CALL PLOT (X+CROS,Y+CROS,2)
      CALL PLOT (X+CROS,Y-CROS,2)
      CALL PLOT (X-CROS,Y-CROS,2)
      CALL PLOT (X-CROS,Y+CROS,2)
      CALL PLOT (X,Y+CROS,2)
      GO TO 730
C                                          STRIPE OR INTERNAL LINE
  735 IF (I.NE.0) GO TO 730
      IF (.NOT.STRIPE) GO TO 730
  736 IF (OFFON.EQ.0) GO TO 737
      CALL PLOT (X,Y,3)
      OFFON=0
      IPEN=3
      GO TO 730
  737 CALL PLOT (X,Y,2)
      OFFON=1
      GO TO 730
C                                          NOTCH POINT
  740 IF (I.NE.0) GO TO 730
      CALL PLOT (X,Y,2)
      IF (NTHOPT) CALL NOTCH (X1,Y1,X,Y,NTHSIZ)
      CALL PLOT (X1,Y1,2)
      GO TO 730
C                                          LETTERING POINT
  750 IF (I.NE.0) GO TO 730
      XL=X
      YL=Y
      GO TO 730
C                                          END PIECE
  800 CONTINUE
C
C     DRAW BLOCK IF BLOCK FLAG SET AND WE HAVE STRIPE AND PLAID
C
      IBLCK=JLOF(MASK8,INCORE(3,IPIEC))
      IF (IBLCK.EQ.0) GO TO 810
      IF (.NOT.PLAID) GO TO 810
      IF (SLAVE) GO TO 813
      APP=XMAX-XMIN+PS
      BPP=YMAX-YMIN+SS
      CALL PLOT (XP,YP,3)
      CALL PLOT (XP,YP+BPP,2)
      CALL PLOT (XP+APP,YP+BPP,2)
      CALL PLOT (XP+APP,YP,2)
      CALL PLOT (XP,YP,2)
C
C     LETTER THE PIECE
```

```
C
  810 APP=XMAX-XMIN-2.0*ABS(XL-X0)
      BPP=YMAX-YMIN-2.0*ABS(YL-Y0)
      RELSIZ=SYMSIZ
      IF (APP.LT.SYMSIZ*9.0) RELSIZ=APP/9.0
      APP=SIZLIN*3.0-1.0
      IF (APP*RELSIZ.GT.BPP) RELSIZ=BPP/APP
      IF (RELSIZ.LT.0.075) RELSIZ=0.075
      X1=XL-RELSIZ*4.0
      Y1=YL+RELSIZ*(SIZLIN-1.0)*0.5
      IF (.NOT.SLAVE) GO TO 815
      X1=X1+RELSIZ*9.0
      GO TO 820
  815 CONTINUE
      I = AREPLR( JLOF(MASKL,INCORE(5,IPIEC)) )
      IF (STYOPT) CALL SYMB (X1,Y1,RELSIZ,INSTY(1,I),0,0,10)
  820 Y1=Y1-RELSIZ*SIZLIN
      IF (SIZOPT) CALL SYMB (X1,Y1,RELSIZ,NAME,0,0,6)
      IF (.NOT.LETOPT) GO TO 835
      I = (JLOF(MASKL,INCORE(5,IPIEC)) -1+LETTER) * 256
      X=X1
      DO 831 K=1,3,1
      IF (NAME(K).NE.2H  ) X=X+RELSIZ*2.0
  831 CONTINUE
      CALL SYMB (X,Y1,RELSIZ,I,0,0,1)
C                                     NEED SPECIAL LABELS?
C
C
C     PUT ON PIECE MESSAGES
C
  835 CONTINUE
      IF (MESBIT(IPIEC).EQ.0) GO TO 880
      IF (SLAVE) GO TO 880
      X=X1+5.0*RELSIZ
      DO 840 I=1,10,1
      IF (JLOF(BITPOS(I),MESBIT(IPIEC)).EQ.0) GO TO 840
      Y1=Y1-RELSIZ*SIZLIN
      CALL MESAG (I,X,Y1,RELSIZ,0,0)
  840 CONTINUE
  880 CONTINUE
C
C     JUST WRAP IT UP NOW
C     DON'T CLEAR TABLE.  HOME HEAD.
C
      IF (XY10PT .OR. XY20PT) GO TO 920
      CALL PLOT (FL,0.0,3)
      GO TO 980
  920 CALL PLOT (XZERO,YZERO,3)
  980 LDIRL=0
      ORIGX=XMOX
      ORIGY=0.0
      XPANU=RLIMIT-XZERO
      AXOLD=0.0
      AYOLD=0.0
      CALL STORE (0,0,0,0)
      CALL FINI (1)
 1000 FORMAT("POS:",4F7.2,I4,2I2)
 1100 FORMAT("MARKER ",5A2," NOT FOUND?")
 1200 FORMAT("MISSING PIECE IN MARKER?")
 1300 FORMAT("MISSING STYLE",I4,"?")
 1400 FORMAT("BAD RULE TABLE",I4,"?")
 1500 FORMAT(8F7.1X,64)
 1600 FORMAT(2I6)
 1700 FORMAT(/,"MARKER WIDTH OF",F5.1," EXCEEDS PAPER WIDTH OF",
     *      F5.1,/,"SHALL I DRAW MARKER?",/,"S@")
```

```
 1000 FORMAT("WIDTH =",F6.2,"IN  LENGTH =",I3,"YD",
     +      I3,"IN  EFF =",F5.1,"%  SCALE =",F5.2)
 1900 FORMAT(A2)
      END
      SUBROUTINE MESAG (MSGNO,X,Y,SYMSIZ,ANGLE)
C
C     ROUTINE TO DRAW CANNED MESSAGES FOR DRAW PROGRAM.
C
      DIMENSION ITEXT(40,3), MESSG(3)
      DATA MESSG /2HME,2HSS,2HG /
      DATA LASTSC /-1/
      IF (MSGNO.LT.1 .OR. MSGNO.GT.60) GO TO 100
      MS=(MSGNO-1)/3
      ML=MOD(MSGNO-1,3)+1
      IF (MS.EQ.LASTSC) GO TO 40
      CALL NDSC (3)
      CALL EXEC (14,1028,ITEXT,120,MESSG,MS)
      LASTSC=MS
   40 IF (ITEXT(1,ML).EQ.0) GO TO 60
      DO 50 I=40,1,-1
      IF (ITEXT(I,ML).NE.2H  ) GO TO 80
   50 CONTINUE
   60 WRITE (6,1000) MSGNO
      RETURN
   80 NCHAR=I*2
      XR=X
      IF (MSGNO.LE.12) XR=X-SYMSIZ*(NCHAR/2)
      IF (ITEXT(1,ML).EQ.2H. ) GO TO 200
      CALL SYMB (XR,Y,SYMSIZ,ITEXT(1,ML),ANGLE,NCHAR)
      RETURN
  100 WRITE (6,2000) MSGNO
      RETURN
C                                        MESAGE WITH UNDERSCORE AS FRST 2H CHA
  200 ATAD=SYMSIZ*0.075
      CALL PLOT (X+ATAD,Y,3)
      CALL PLOT (X+ATAD,Y+SYMSIZ*2.0,0,2)
      CALL SYMB (XR,Y+SYMSIZ*2.0,0,SYMSIZ,ITEXT(2,ML),ANGLE,
     +      NCHAR-2)
      RETURN
 1000 FORMAT("MESSAGE",I3," IS EMPTY!")
 2000 FORMAT("MESSAGE OUT OF RANGE:",I6)
      END
      SUBROUTINE NOTCH (XA,YA, XB,YB, SIZE)
C
C     ROUTINE TO DRAW PERPENDICULAR NOTCH HEAD WHEN REQUESTED.
C     NOTCH IS FROM XA,YA TO XB,YB.  PEN IS ASSUMED TO BE AT
C     THE POINT XB,YB WHEN CALLED.  IT IS RETURNED THERE
C     BEFORE EXITING.
C

      RY=YA-YB
      RX=XB-XA
      RL=SIZE*0.5/SQRT(RX*RX+RY*RY)
      YD=YB+RX*RL
      XD=XB+RY*RL
      YC=YB-RX*RL
      XC=XB-RY*RL
      CALL PLOT (XC,YC,2)
      CALL PLOT (XD,YD,2)
      CALL PLOT (XB,YB,2)
      RETURN
      END
END$
```

```
ASMB
       NAM DSPLY,7
       ENT INDSP,STEP,NSPAC,DSPLY
       EXT .ENTR,ERR
*
*      THE "INDSP" CALL ESTABLISHES A BUFFER AREA AND BUFFER
*      POINTER EXTERNAL TO THE DISPLAY PLOT ROUTINE.
*      IF "INDSP" IS NOT CALLED PRIOR TO USING "DSPLY",
*      AN ERROR ABORT IS TAKEN.
*
*      -CALLING SEQUENCE-
*
*         CALL INDSP(NAW,IBUF,NWORD)
*
*              NAW  :  ADDRESS OF BUFFER POINTER FOR NEXT
*                      AVAILABLE WORD IN DISPLAY BUFFER
*              IBUF :  DISPLAY BUFFER ADDRESS
*              NWORD:  NUMBER OF WORDS IN DISPLAY BUFFER
*
NAW    NOP             BUFFER POINTER ADDRESS
IBUF   NOP             BUFFER ADDRESS
NWORD  NOP             # WORDS IN BUFFER ADDRESS
INDSP  NOP
       JSB .ENTR
       DEF NAW
       LDA NAW         LOAD DISPLAY BUFFER POINTER ADDRESS
       STA POINT       SAVE
       LDA IBUF        LOAD DISPLAY BUFFER ADDRESS
       STA BUFAD       SAVE
       ADA NWORD,I     ADD # WORDS IN BUFFER TO ADDRESS
       STA BUFSZ       SAVE LAST BUFFER ADDRESS+1
       ISZ ICALL       SET "INDSP" CALL FLAG
       JMP INDSP,I     RETURN
POINT  OCT 0           USER BUFFER POINTER ADDRESS
BUFAD  OCT 0           USER DISPLAY FILL BUFFER ADDRESS
BUFSZ  OCT 0           LWA+1 OF USER DISPLAY FILE BUFFER
ICALL  OCT 0           "INDSP" CALL FLAG
*
*      THE "STEP" CALL ALLOWS THE USER TO CHANGE THE
*      MAXIMUM RASTER X AND Y IN "INCREMENTAL" AND "RELATIVE"
*      INSTRUCTIONS, THE MAXIMUM NUMBER OF "INCREMENTAL"
*      INSTRUCTIONS GENERATED PER MOVE, AND WHETHER TO
*      PACK "RELATIVE" MOVES.
*      (MOVES WILL LIMIT TO + INCLUDING "INMAX" AND "IRMAX")
*      (DEFAULTS:  INMAX=63, IRMAX=2047, IMOVE=2, IPACK=0)
*
*      -CALLING SEQUENCE-
*
*         CALL STEP(INMAX,IRMAX,IMOVE,IPACK)
*
*              INMAX:  RASTER X AND Y LIMIT FOR
*                      "INCREMENTAL" INSTRUCTIONS
*              IRMAX:  RASTER X AND Y LIMIT FOR
*                      "RELATIVE" INSTRUCTIONS
*              IMOVE:  MAXIMUM NUMBER OF "INCREMENTAL"
*                      INSTRUCTIONS GENERATED PER MOVE
*              IPACK:  "RELATIVE" DATA PACK FLAG.
*                      0=PACK, 1=NO-PACK
*
INMAX  NOP             "INCREMENTAL" RASTER LIMIT ADDRESS
IRMAX  NOP             "RELATIVE" RASTER LIMIT ADDRESS
IMOVE  NOP             "INCREMENTAL" INSTRUCTION LIMIT ADDRESS
IPACK  NOP             "RELATIVE" DATA PACK FLAG ADDRESS
STEP   NOP
       JSB .ENTR
```

```
        DEF INMAX
        LDA INMAX,I     LOAD "INCREMENTAL" LIMIT
        SSA,RSS         POSITIVE VALUE?
        SZA,RSS         NON-ZERO?
        JMP STEP2       NO, IGNORE
        CMA,INA         COMPLEMENT
        ADA IMAX        ADD MAXIMUM
        SSA             <= MAX?
        JMP STEP6       NO, SET TO DEFAULT
        LDA INMAX,I     YES, LOAD "INCREMENTAL" LIMIT
STEP1   STA ISTEP       RESET "INCREMENTAL" LIMIT
STEP2   LDA IRMAX,I     LOAD "RELATIVE" LIMIT
        SSA,RSS         POSITIVE VALUE?
        SZA,RSS         NON-ZERO?
        JMP STEP4       NO, IGNORE
        CMA,INA         COMPLEMENT
        ADA RMAX        ADD MAXIMUM
        SSA             <= MAX?
        JMP STEP7       NO, SET TO DEFAULT
        LDA IRMAX,I     LOAD "RELATIVE" LIMIT
STEP3   STA RSTEP       RESET "RELATIVE" LIMIT
STEP4   LDA IMOVE,I     LOAD MOVE LIMIT
        SSA,RSS         POSITIVE VALUE?
        SZA,RSS         NON-ZERO?
        JMP STEP5       NO
        STA MAXM        RESET MOVE LIMIT
STEP5   LDA IPACK,I     LOAD "RELATIVE" DATA PACK FLAG
        STA IPAC        SAVE
        JMP STEP,I      RETURN
STEP6   LDA IMAX        LOAD MAXIMUM "INCREMENTAL" LIMIT
        JMP STEP1       SET LIMIT TO MAXIMUM
STEP7   LDA RMAX        LOAD MAXIMUM "RELATIVE" LIMIT
        JMP STEP3       SET LIMIT TO MAXIMUM
IMAX    DEC 63          MAXIMUM "INCREMENTAL" LIMIT
ISTEP   DEC 63          "INCREMENTAL" LIMIT
RMAX    DEC 2047        MAXIMUM "RELATIVE" LIMIT
RSTEP   DEC 2047        "RELATIVE" LIMIT
MAXM    DEC 2           MOVE LIMIT
IPAC    DEC 0           "RELATIVE" DATA PACK FLAG
*
*       THE "NSPACE" CALL ALLOWS THE USER TO CHANGE THE
*       MAXIMUM # SPACE LIMIT FOR ALL X AND Y DATA
*       SENT TO "DSPLY".
*       (LIMIT IS TO, BUT NOT INCLUDING "NRAST")
*       (DEFAULT:  NRAST=2045)
*
*       -CALLING SEQUENCE-
*
*          CALL NSPAC(NRAST)
*
*                NRAST:  NEW RASTER NUMBER SPACE LIMIT
*
NRAST   NOP             RASTER NUMBER SPACE LIMIT ADDRESS
NSPAC   NOP
        JSB .ENTR
        DEF NRAST
        LDA NRAST,I     LOAD NEW # SPACE LIMIT
        SSA,RSS         POSITIVE VALUE?
        SZA,RSS         NON-ZERO?
        JMP NSPAC,I     NO, IGNORE
        CMA,INA         COMPLEMENT
        ADA NMAX        ADD MAXIMUM # SPACE SIZE
        SSA             <= MAX?
        JMP NSP2        NO, SET TO DEFAULT
        LDA NRAST,I     YES, LOAD NEW # SPACE LIMIT
```

```
NSP1  CMA,INA         COMPLEMENT
      STA NUMBS       RESET # SPACE LIMIT
      JMP NSPAC,I     RETURN
NSP2  LDA NMAX        LOAD MAXIMUM # SPACE LIMIT
      JMP NSP1        SET # SPACE LIMIT TO MAX
NMAX  DEC 2048        MAXIMUM # SPACE LIMIT
NUMBS DEC -2048       # SPACE LIMIT (NEG)
*
*     THE "DSPLY" ROUTINE CONVERTS THE DEFINED X,Y RASTER
*     PARAMETERS TO CRT DISPLAY INSTRUCTIONS AND
*     STORES THESE INSTRUCTIONS IN A USER DEFINED
*     DISPLAY FILE BUFFER
*
*     -CALLING SEQUENCE-
*
*        CALL DSPLY(IXR,IYR,IBEAM)
*
*             IXR   :  X-RASTER COORDINATE
*             IYR   :  Y-RASTER COORDINATE
*             IBEAM:  DISPLAY FUNCTION
*                     =2          BEAM ON
*                     =3          BEAM OFF
*                     =-2 OR -3   DEFINE NEW LIST (ORIGIN)
*
IXR   NOP             X-RASTER COORDINATE
IYR   NOP             Y-RASTER COORDINATE
IBEAM NOP             BEAM FUNCTION
DSPLY NOP
      JSB .ENTR
      DEF IXR
      LDA ICALL       LOAD "INDSP" CALL FLAG
      SZA             CALL MADE?
      JMP CPRAM       YES, CHECK CURRENT PARAMETERS
      LDB =D99        NO, LOAD ERROR CODE
      JMP ABORT       ABORT PROGRAM
*
CPRAM LDA IXR,I       LOAD X-RASTER COORDINATE
      SSA             POSITIVE?
      CMA,INA         NO, SET POSITIVE
      ADA NUMBS       SUBTRACT # SPACE LIMIT
      SSA,RSS         IN # SPACE?
      JMP OVER        NO, BAD DATA!
      LDA IYR,I       LOAD Y-RASTER COORDINATE
      SSA             POSITIVE?
      CMA,INA         NO, SET POSITIVE
      ADA NUMBS       SUBTRACT # SPACE LIMIT
      SSA             IN # SPACE?
      JMP CAL         YES, CONTINUE
OVER  CLA,INA         SET FOR NON-ABORT MESSAGE
      LDB =D98        LOAD ERROR CODE
      JMP ABORT       ERROR EXIT
*
CAL   LDA IBEAM,I     LOAD DISPLAY FUNCTION
      SSA,RSS         NEGATIVE REQUEST?
      JMP MOVE        NO, NORMAL MOVE
      CMA,INA         SET FUNCTION POSITIVE
      STA IPN         SAVE BEAM STATUS
      LDA IXR,I       LOAD X-RASTER COORDINATE
      LDB IYR,I       LOAD Y-RASTER COORDINATE
      SZA,RSS         X-COORDINATE = 0?
      SZB             YES, Y-COORDINATE = 0?
      JMP PROC        NO, PROCESS MOVE
      STA IXO         INITIALIZE X POSITION
      STA IYO         INITIALIZE Y POSITION
      STA LMO         INITIALIZE MODE FLAG
```

```
        JMP USPLY,I     RETURN
PROC    STA IXM         SET DELTA X MOVE
        STB IYM         SET DELTA Y MOVE
        LDA IREL        LOAD "RELATIVE" INSTRUCTION HEADER
        LDB .+2         LOAD STORE CODE AND "RELATIVE" MODE FLAG
        STB LMD         SET MODE FLAG
        JSB IOUT        STORE INSTRUCTION HEADER
        CLA             A=-1
        STA N           SET # MOVES = 1
        JMP REL         PROCESS "RELATIVE" INSTRUCTION MOVE
*
MOVE    STA IPN         SAVE BEAM STATUS
        LDB IXO         LOAD OLD X-COORDINATE
        CMB,INB         COMPLEMENT
        ADB IXR,I       COMPUTE DELTA X MOVE
        STB IXM         SAVE
        LDA IYO         LOAD OLD Y-COORDINATE
        CMA,INA         COMPLEMENT
        ADA IYR,I       COMPUTE DELTA Y MOVE
        STA IYM         SAVE
        SZA             DELTA Y MOVE?
        JMP ITYPE       YES, DETERMINE INSTRUCTION MODE
        SZB,RSS         DELTA X MOVE?
        JMP USPLY,I     NO, RETURN
*
ITYPE   SSA,RSS         NEGATIVE DELTA Y MOVE?
        CMA,INA         NO, SET NEGATIVE
        SSB             POSITIVE DELTA X MOVE?
        CMB,INB         NO, SET POSITIVE
        ADA B           DETERMINE LARGEST MOVE
        SSA             X OR Y LARGEST?
        JMP YBIG        Y
        LDA IXM         X, LOAD DELTA X MOVE
        JMP XBIG        CONTINUE
YBIG    LDA IYM         LOAD DELTA Y MOVE
XBIG    SSA             POSITIVE VALUE?
        CMA,INA         NO, SET POSITIVE
        STA TEMP        SAVE LARGEST MOVE
        CLB             B=0
        DIV ISTEP       COMPUTE # STEPS REQUIRED
        SZB             REMAINDER?
        INA             YES, BUMP # STEPS
        CMA,INA         COMPLEMENT
        STA N           SAVE # STEPS
        ADA MAXM        ADD "INCREMENTAL" LIMIT
        SSA             "INCREMENTAL" MODE OK?
        JMP RELM        NO, "RELATIVE" MODE
        CLA,INA         LOAD "INCREMENTAL" MODE FLAG
        JMP SETM        SET MODE
*
RELM    CLA             A=-1
        STA N           SET # STEPS TO 1
        LDA TEMP        LOAD LARGEST MOVE
        CMA,INA         COMPLEMENT
        ADA RSTEP       ADD "RELATIVE" LIMIT
        SSA             MOVE LARGER THAN MAX?
        JMP SETN        YES, DETERMINE # MOVES
        LDA .+2         LOAD "RELATIVE" MODE FLAG
        JMP SETM        SET MODE
*
SETN    CLB             B=0
        LDA TEMP        LOAD LARGEST DELTA MOVE
        DIV RSTEP       COMPUTE # MOVES
        SZB             REMAINDER?
        INA             YES, BUMP # MOVES
```

```
        CMA,INA         COMPLEMENT
        STA N           SET # MOVES
        LDA .+2         LOAD "RELATIVE" MODE FLAG
*
SETM    STA MODE        SET MODE
        CPA LMD         SAME AS LAST MODE?
        JMP CLEAR       YES
        STA LMD         SAVE NEW MODE
        CPA .+1         "INCREMENTAL" MODE?
        JMP .INCR       YES
        LDA IREL        LOAD "RELATIVE" INSTRUCTION HEADER
        JMP .REL        CONTINUE
.INCR   LDA INCR        LOAD "INCREMENTAL" INSTRUCTION HEADER
.REL    LDB .+2         LOAD STORE CODE
        JSB IOUT        STORE IN BUFFER
        JMP DETM        PROCESS MOVE
*
CLEAR   LDB LAST        LOAD STATUS FLAG
        JSB IOUT        CLEAR TERMINATE BIT
*
DETM    LDA MODE        LOAD MODE FLAG
        CPA .+1         "INCREMENTAL" MODE?
        JMP INC         YES, PROCESS
*
REL     LDA IXM         LOAD DELTA X-MOVE
        LDB IPAC        LOAD DATA PACK FLAG
        SZB             "RELATIVE" DATA PACKED?
        JMP REL.1       NO
        SZA,RSS         NON-ZERO MOVE?
        JMP YI          NO, PROCESS Y MOVE
REL.1   LDB N           LOAD # STEPS LEFT
        CMB,INB         COMPLEMENT FOR DIVIDE
        STB #MOVE       SAVE
        CLB             B=0
        SSA             POSITIVE MOVE?
        CCB             NO, B=-1
        DIV #MOVE       COMPUTE STEP MOVE
        STA IXX         SAVE
        CLB             B=0
        RPL 4           SHIFT FOR CONTROL BITS
        INA             X, LOAD REGISTER
        LDB IPAC        LOAD DATA PACK FLAG
        SZB             "RELATIVE" DATA PACKED?
        JMP Y           NO
        LDB IYM         LOAD DELTA Y MOVE
        SZB             NON-ZERO?
        JMP Y           YES
        ADA .+4         LOAD AND DRAW
        CLB             B=0
        STB LAST        SET BEAM STATUS FLAG = ON
        LDB IPN         LOAD BEAM STATUS
        CPB .+2         BEAM ON?
        JMP REL.2       YES, CONTINUE
        ADA .+4         LOAD AND MOVE
        ISZ LAST        SET BEAM STATUS FLAG = OFF
REL.2   ISZ N           BUMP STEP COUNT
        JMP REL.3       MORE STEPS, CONTINUE
        CPB .+3         BEAM OFF?
        JMP MOVEX       YES
        ADA .+6         SET LD+T
        JMP STORX       CONTINUE
MOVEX   LDB .+2         LOAD STORE FLAG
        JSB IOUT        STORE X INSTRUCTION IN BUFFER
        LDA TERM        LOAD TERMINATE INSTRUCTION
```

```
STORX   LDB  .+2         LOAD STORE FLAG
        JSB  IOUT        STORE IN BUFFER
        JMP  EX          EXIT
REL.3   LDB  IXX         LOAD CURRENT X STEP MOVE
        CMB,INB          COMPLEMENT
        ADB  IXM         SUBTRACT FROM TOTAL X MOVE
        STB  IXM         SAVE REMAINDER
        LDB  .+2         LOAD STORE FLAG
        JSB  IOUT        STORE CURRENT X INSTRUCTION IN BUFFER
        LDA  IXM         LOAD REMAINDER
        JMP  REL.1       PROCESS NEXT X
*
Y       LDB  .+2         LOAD STORE FLAG
        JSB  IOUT        STORE X INSTRUCTION IN BUFFER
        LDA  N           LOAD STEP COUNT
        ADA  .+1         BUMP COUNT
        SZA,RSS          LAST STEP?
        JMP  YI          YES, CONTINUE WITH Y MOVE
        LDA  IXX         LOAD CURRENT X STEP MOVE
        CMA,INA          COMPLEMENT
        ADA  IXM         SUBTRACT FROM TOTAL X MOVE
        STA  IXM         SAVE REMAINDER
Y1      LDB  N           LOAD # STEPS LEFT
        CMB,INB          COMPLEMENT FOR DIVIDE
        STB  #MOVE       SAVE
        CLB              B=0
        LDA  IYM         LOAD DELTA Y-MOVE
        SSA              POSITIVE VALUE?
        CCB              NO, B=-1
        DIV  #MOVE       COMPUTE STEP MOVE
        STA  IYY         SAVE
        CLB              B=0
        RRL  4           SHIFT FOR CONTROL BITS
        ADA  .+6         Y, LOAD AND DRAW
        CLB              B=0
        STB  LAST        SET BEAM STATUS FLAG = ON
        LDB  IPN         LOAD BEAM STATUS
        CPB  .+2         BEAM ON?
        JMP  Y.1         YES, CONTINUE
        ADA  .+4         LOAD AND MOVE
        ISZ  LAST        SET BEAM STATUS FLAG = OFF
Y.1     ISZ  N           ALL STEPS PROCESSED?
        JMP  Y.2         NO, CONTINUE
        CPB  .+3         BEAM OFF?
        JMP  MOVEY       YES
        ADA  .+8         SET LD+T
        JMP  STORY       CONTINUE
MOVEY   LDB  .+2         LOAD STORE FLAG
        JSB  IOUT        STORE Y INSTRUCTION IN BUFFER
        LDA  TERM        LOAD TERMINATE INSTRUCTION
STORY   LDB  .+2         LOAD STORE FLAG
        JSB  IOUT        STORE IN BUFFER
        JMP  EX          EXIT
Y.2     LDB  IYY         LOAD CURRENT Y STEP MOVE
        CMB,INB          COMPLEMENT
        ADB  IYM         SUBTRACT FROM TOTAL Y MOVE
        STB  IYM         SAVE REMAINDER
        LDB  .+2         LOAD STORE FLAG
        JSB  IOUT        STORE Y INSTRUCTION IN BUFFER
        LDA  IXM         LOAD X MOVE
        LDB  IPAC        LOAD DATA PACK FLAG
        SZB              "RELATIVE" DATA PACKED?
        JMP  REL.1       NO
        SZA              ZERO?
```

```
        JMP REL.1       NO, PROCESS NEXT X
        JMP YI          PROCESS NEXT Y
*
EX      LDA IXR,I       LOAD X-RASTER COORDINATE
        STA IXO         SAVE FOR OLD POSITION
        LDA IYR,I       LOAD Y-RASTER COORDINATE
        STA IYO         SAVE FOR OLD POSITION
        JMP DSPLY,I     RETURN
*
INC     LDA IYM         LOAD DELTA Y-MOVE
INC.1   LDB N           LOAD # STEPS LEFT
        CMB,INB         COMPLEMENT FOR DIVIDE
        STB #MOVE       SAVE
        CLB             B=0
        SSA             POSITIVE?
        CCB             NO, B=-1
        DIV #MOVE       COMPUTE STEP MOVE
        STA IYY         SAVE Y STEP MOVE
        CLB             B=0
        LDA IXM         LOAD DELTA X-MOVE
        SSA             POSITIVE?
        CCB             NO, B=-1
        DIV #MOVE       COMPUTE STEP MOVE
        STA IXX         SAVE X STEP MOVE
        CLB             B=0
        RRL 1           SHIFT FOR BEAM STATUS
        STA TEMP        SAVE
        LDA IPN         LOAD BEAM STATUS
        CPA .+2         BEAM ON?
        ISZ TEMP        YES, SET ON
        LDB IYY         LOAD Y STEP MOVE
        RRL 9           PUT IN UPPER B
        LDA TEMP        LOAD X PART AND BEAM STATUS
        RRL 7           COMBINE
        CLB             B=0
        RRL 1           SHIFT FOR TERMINATE
        ISZ N           ALL STEPS PROCESSED?
        JMP INC.2       NO, CONTINUE
        INA             YES, SET TERMINATE
        LDB .+2         LOAD STORE FLAG
        JSB IOUT        STORE INSTRUCTION IN BUFFER
        JMP EX          EXIT
INC.2   LDB .+2         LOAD STORE FLAG
        JSB IOUT        STORE INSTRUCTION IN BUFFER
        LDA IXX         LOAD X STEP MOVE
        CMA,INA         COMPLEMENT
        ADA IXM         SUBTRACT FROM TOTAL X MOVE
        STA IXM         SAVE REMAINDER
        LDA IYY         LOAD Y STEP MOVE
        CMA,INA         COMPLEMENT
        ADA IYM         SUBTRACT FORM TOTAL Y MOVE
        STA IYM         SAVE REMAINDER
        JMP INC.1       CONTINUE
*
IOUT    NOP
        CPB .+2         STORE OR CLEAR?
        JMP NORD        STORE
        STB ITY         SAVE TYPE FLAG
        LDB BUFAD       LOAD BUFFER ADDRESS
        ADB POINT,I     ADD BUFFER POINTER
        ADB .-2         SET ADDRESS TO LAST INSTRUCTION
        STB TEMP1       SAVE ADDRESS
        LDB B,I         LOAD INSTRUCTION
        SZB             ZERO WORD?
        JMP IOUT1       NO, OK
```

```
        LDB  TEMP1      LOAD ADDRESS
        ADB  .-1        ADJUST ADDRESS BEFORE ZERO WORD
        STB  TEMP1      RESTORE ADDRESS
IOUT1   CPA  .+1        "INCREMENTAL" MODE?
        JMP  INCM       YES
        LDB  ITY        LOAD TYPE FLAG
        SZB             DRAW OR MOVE?
        JMP  .MOV       MOVE
        LDA  TEMP1,I    LOAD INSTRUCTION
        AND  .-9        TURN OFF BIT 3
EX,1    STA  TEMP1,I    RESTORE INSTRUCTION
        JMP  IOUT,I     RETURN
INCM    LDA  TEMP1,I    LOAD INSTRUCTION
        AND  .-2        TURN OFF BIT 0
        JMP  EX,1       CONTINUE
.MOV    LDA  POINT,I    LOAD BUFFER POINTER
        ADA  .-1        SET TO TERMINATE INSTRUCTION
        STA  POINT,I    RESET BUFFER POINTER
        JMP  IOUT,I     RETURN
NORM    LDB  BUFAD      LOAD BUFFER ADDRESS
        ADB  POINT,I    ADD BUFFER POINTER
        ADB  .-1        SET TO NEXT STORE
        STB  TEMP1      SAVE ADDRESS
        ISZ  POINT,I    BUMP POINTER
        CMB             COMPLEMENT STORE ADDRESS
        ADB  BUFSZ      ADD BUFFER END ADDRESS
        SSB             OVERFLOW?
        JMP  IOUT,I     YES, SKIP STORE
        STA  TEMP1,I    STORE INSTRUCTION IN BUFFER
        JMP  IOUT,I     RETURN
*
ABORT   STB  TEMP       SET ERROR CODE
        STA  TEMP1      SET ABORT/NO ABORT FLAG
        JSB  ERR        OUTPUT MESSAGE
        DEF  *+3        RETURN ADDRESS
        DEF  TEMP       ERROR CODE ADDRESS
        DEF  TEMP1      ABORT/NO ABORT FLAG ADDRESS
        JMP  DSPLY,I    (NON-ABORT RETURN)
*
*       STORAGE
*
IPN     OCT  0          BEAM STATUS
ITY     OCT  0          "IOUT" TYPE FLAG
IXM     OCT  0          DELTA X MOVE
IYM     OCT  0          DELTA Y MOVE
IXO     OCT  0          OLD X POSITION
IYO     OCT  0          OLD Y POSITION
IXX     OCT  0          X STEP MOVE
IYY     OCT  0          Y STEP MOVE
LAST    OCT  0          BEAM STATUS FLAG
LMO     OCT  0          LAST MODE STATUS FLAG
MODE    OCT  0          INSTRUCTION MODE
N       OCT  0          STEP COUNT (NEG)
#MOVE   OCT  0          STEP COUNT (POS)
TEMP    OCT  0          TEMPORARY STORAGE
TEMP1   OCT  0          TEMPORARY STORAGE
INCR    OCT  100000     "INCREMENTAL" INSTRUCTION HEADER
IREL    OCT  102000     "RELATIVE" INSTRUCTION HEADER
TERM    OCT  14         "RELATIVE" TERMINATE INSTRUCTION
*
B       EQU  1          B-REGISTER
.       EQU  B3B        BASE PAGE CONSTANTS, .=0, + OR - 1 TO 10
*
        END
```

```
      SUBROUTINE DSROL(NRLNO,IDSPB)
C..........SUBROUTINE TO DISPLAY A NUMBER AT THE POSITION
C..........SET UP IN EN123.
      DIMENSION IDIGT(5),IDSPB(1)
      IPTR = IABS(IDSPB(2))
      IDSPB(2) = -IPTR
      IDSPB(IPTR-1) = 1000000B
      DO 20 I=1,5
      IDIGT(6-I) = MOD(NRLNO,10)
   20 NRLNO = NRLNO/10
      DO 30 I=1,5
      IF(IDIGT(I).NE.0)GO TO 40
   30 IDIGT(I) = 32
      IDIGT(5) = 48
      GO TO 50
   40 DO 45 K=I,5
   45 IDIGT(K) = 48+IDIGT(K)
   50 IDSPB(IPTR+6) = IDIGT(1)*256 + IDIGT(2)
      IDSPB(IPTR+7) = IDIGT(3)*256 + IDIGT(4)
      IDSPB(IPTR+8) = IDIGT(5)*256 + 20
      IDSPB(2) = IPTR
      IDSPB(IPTR-1) = 0
      RETURN
      END
      ENDS


ASMB,L
      NAM DVR03,7
      ENT DVR03,.CONT
      ENT DEF03
      EXT .ENTR
CHAN  EQU 13B
.     EQU 53B
IB    NOP
IS    NOP
IX    NOP
IY    NOP
IFLAG NOP
DVR03 NOP
      JSB .ENTR
      DEF IB
      LDA BUSY
      SZA
      JMP *-2
      ISZ BUSY
      LDA IB
      STA B
      LDA IS
      STA S
      LDA IX
      STA X
      LDA IY
      STA Y
      LDA IFLAG
      STA FLAG
      CLA
      STA FLAG,I
      LDA =B114003
      STA CHAN
      LDA CONT
      STA 2B
      STC CHAN,C
      JMP DVR03,I
B     NOP
```

```
S       NOP
X       NOP
Y       NOP
FLAG    NOP
CONT    DEF  *+1
.CONT   NOP
        STA SAVEA
        STB SAVEB
        LIA CHAN
        CLB
        RRR 4
        BLF
        STB S,I
        CLB
        RRR 7
        LDB .-0
        CPA .+1
        JMP STOR
        INB
        CPA .
        JMP STOR
        INB
        CPA .+2
        JMP STOR
        INB
        CPA .+4
        JMP STOR
        INB
        CPA .+6
        JMP STOR
        INB
STOR    CMB,INB
        STB B,I
        LIA CHAN
        STA X,I
        LIA CHAN
        STA Y,I
        ISZ FLAG,I
        CLA
        STA BUSY
        LDA SAVEA
        LDB SAVEB
        CLC CHAN
        JMP .CONT,I
BUSY    NOP
SAVEA   NOP
SAVEB   NOP
DFF03   NOP
        JSB .ENTR
        DEF DFF03
        CLA
        STA CHAN
        CLC CHAN,C
        JMP DFF03,I
        END


ASMB,L
        NAM DVR50,7
        ENT DVR50
        ENT DFF50
        EXT .ENTR
CHAN    EQU 12B
NUM     NOP
FLAG    NOP
```

```
DVR5J NOP
      JSB .ENTR
      DEF NUM
      LDA BUSY
      SZA
      JMP *-2
      ISZ BUSY
      LDA FLAG
      STA F
      LDA NUM
      STA N
      LDA =B1140003
      STA CHAN
      LDA INTR
      STA S
      CLA
      STA FLAG,I
      STC CHAN,C
      JMP DVR5J,I
F     NOP
N     NOP
INTR  DEF *+1
CONT  NOP
      STA SAVEA
      LIA CHAN
      AND =B17
      STA N,I
      LDA =B1
      OTA CHAN
      CLA
      OTA CHAN
      ISZ F,I
      CLC CHAN,C
      CLA
      STA CHAN
      STA BUSY
      LDA SAVEA
      JMP CONT,I
SAVEA NOP
BUSY  NOP
OFF5M NOP
      JSB .ENTR
      DEF OFF5M
      CLA
      STA CHAN
      CLC CHAN,C
      JMP OFF5M,I
      END


ASMB
      NAM DVRUP,7
      ENT DVRUP
      EXT .ENTR,MPOFF,MPON,EXEC
*
*     THIS ROUTINE INITIALIZES THE DIGITIZER AND PLOTTER DRIVERS
*
DVRUP NOP
      JSB .ENTR
      DEF DVRUP
*
      JSB EXEC       READ IN OPTIONS FILE
      DEF **7
      DEF D14
```

```
        DEF CW
        DEF BUFER
        DEF .+7
        DEF OPTNS
        DEF .
        LDA BUFER+4
        STA NSEC        SET # SEC/DIG FILE
        LDA BUFER+5
        STA FILES       SET # DIG FILES
        LDA BUFER+6
        CLB
        ERR 8
        SLE,SLE
        STA TRAK        SET DIG START TRACK
        STA SECT        SET DIG START SECTOR
*
        JSB OPOFF       TURN MEMORY PROTECT OFF
        DEF *+1         RETURN ADDRESS
*
*       INITIALIZE DIGITIZER DRIVER
*
        LDA DCHAN       LOAD DIGITIZER I/O SLOT CONTENTS
        CPA JSB2        DIGITIZER DRIVER ALREADY INITIALIZED?
        JMP IGNOR       YES
*
        JSB IN.FD,I     INITIALIZE F.B./DIGITIZER
TRAK    OCT 0           1ST FILE TRK #
SECT    OCT 0           1ST FILE SECTOR #
NSEC    OCT 0           # SEC/FILE
FILES   OCT 0           # FILES
*
*       INITIALIZE PLOTTER DRIVER
*
IGNOR   JSB EXEC        FIND PLOTF
        DEF *+4
        DEF D13
        DEF PLOTF
        DEF SIZE
        STA TRKSC       SET TRACK/SECTOR ADDRESS
        LDA SIZE
        SZA,RSS         PLOTF THERE?
        JMP ERR1        NO, INDICATE
        JMP ERR2        HAVE OPER INITIALIZE XYNETICS
POWER   LIA XYCHN       LOAD XYNETICS STATUS
        AND FAIL        MASK FOR POWER BITS
        SZA,RSS         POWER ON?
        JMP ERR2        NOPE, INDICATE AND WAIT
*
*       CHECK FOR OPERATOR REQUESTED VELOCITY CHANGE
*
        CLA
        SZA,RSS         OPERATOR REQUEST VELOCITY CHANGE?
        JMP INXY        NO
        LDB V30         LOAD 30 IPS DEFAULT
        CPA .+3         10 IPS?
        LDB V10         YES
        CPA =B20        20 IPS?
        LDB V20         YES
        STB PT2,I       SET PT2
        STB SLWPT,I     SET SLWPT
        CLA             A=0
        OTA 1           CLEAR SWITCH REGISTER
*
INXY    JSB XYINT,I     INITIALIZE XYNETICS DRIVER
TRKSC   OCT 0           PLOTF TRACK/SECTOR
```

```
SIZE    OCT 5           PLOTF SIZE
        OCT 0           ON SUBCHAN 0
*
EXIT    JSB MPON        TURN MEMORY PROTECT ON
        DEF **1
        JMP DVRUP,I     RETURN
*
*       MESSAGES
*
ERR1    JSB EXEC        PRINT MESSAGE
        DEF **5
        DEF .+2
        DEF TTY
        DEF MSG1
        DEF EMSG1
        JMP EXIT
MSG1    ASC 7,PLOTF MISSINGA
        OCT 5412
        ASC 12,XYNETICS NOT ACTIVATEDA
EMSG1   ABS *-MSG1
*
ERR2    JSB EXEC        PRINT MESSAGE
        DEF **5
        DEF .+2
        DEF TTY
        DEF MSG2
        DEF EMSG2
        JSB EXEC        WAIT FOR !GO
        DEF **2
        DEF .+7
        LDA 1,I         LOAD PRAMS
        SZA             OPERATOR SAY WHAT?
        JMP EXIT        YEP, IGNORE INIT.
        JMP POWER       TRY AGAIN
MSG2    ASC 10,INITIALIZE XYNETICSA
EMSG2   ABS *-MSG2
*
BUFER   BSS 7
D14     DEC 14
D18     DEC 18
CW      OCT 102
FAIL    OCT 3000
OPTNS   ASC 3,OPTNS
PLOTF   ASC 3,PLOTF
TTY     OCT 101
JSB2    JSB 2,I
V10     OCT 700         10 IPS VELOCITY PARAMETER
V20     OCT 2015        20 IPS VELOCITY PARAMETER
V30     OCT 3143        30 IPS VELOCITY PARAMETER
*
DCHAN   EQU 13B
XYCHN   EQU 24B
.       EQU 53B
*
IN.FD   OCT 5614          F.B./DIG INIT ADDRESS
XYINT   OCT 5703          XYNETICS DRIVER INIT ADDRESS
PT2     OCT 11245       PT2 ADDRESS IN LA001
SLWPT   OCT 11247       SLWPT ADDRESS IN LA001
*
        END
```

```
ASMB
        NAM DVRVG,7
        ENT INIT,COMP
        ENT TSTAT,XTHIT,YTHIT
        ENT BASE,TADD,FSWCH,LSTFB,DSTAT
*
*       VECTOR GENERAL DISPLAY DRIVER
*       FOR 21MXA  DOS-M  REV. D
*
*       DVRVG IS A SINGLE DMA CHANNEL, INTERRUPT CONTROL DISPLAY
*       DRIVER.  THIS DRIVER BY-PASSES DOS-M BY USING I/O
*       SLOT TRAP CELL LOCATIONS TO JUMP DIRECTLY TO THE DRIVER
*       RATHER THAN TO $CIC.  THIS DRIVER USES DMA CHANNEL 1
*       EXCLUSIVELY AND MAKES NO PROVISIONS FOR CHANNEL SWITCHING.
*       IF, ON INITIALIZATION, DMA CHANNEL 1 IS BUSY,
*       THE DRIVER RETURNS TO THE P+2 ERROR RETURN, OTHERWISE
*       IT RETURNS TO THE P+3 NORMAL RETURN.
*
*       DATA AVAILABLE TO USER ROUTINES INCLUDE:
*
*           STATUS WORD OF TABLET              (TSTAT)
*           LAST X-COORDINATE ON TABLET        (XTHIT)
*           LAST Y-COORDINATE ON TABLET        (YTHIT)
*           DISPLAY FILE FWA-1                 (BASE)
*           LIST TABLE FWA                     (TADD)
*           FUNCTION SWITCH REGISTER           (FSWCH)
*           LAST FUNCTION SWITCH HIT           (LSTFB)
*           DISPLAY REFRESH STATUS             (DSTAT)
*
*       A CONTROL WORD IS USED BY THE DRIVER TO DETERMINE
*       IF A REFRESH OF THE DISPLAY IS REQUIRED:
*
*             15                                              1  0
*             ..................................................
*             .                                              . T. C.
*             ..................................................
*
*                     T = END OF TABLE FLAG
*                     C = FRAME CLOCK REFRESH FLAG
*
*       WHEN THE T AND C BITS ARE SET, REFRESH IS ENABLED.
*
*       - INITIALIZATION CALLING SEQUENCE -
*           ...
*           JSB INIT        INITIALIZATION CALL
*           DEF PRAM        ADDRESS OF 3 WORD PARAMETER TABLE
*           ...             (DMA REJECT RETURN)
*           ...             (NORMAL RETURN)
*           ...
*
*       PARAMETER TABLE FORMAT:
*
*
*             ..................................................
*             .  DISPLAY FILE FIRST WORD ADDRESS - 1           .
*             ..................................................
*             .  LIST TABLE FIRST WORD ADDRESS                 .
*             ..................................................
*             .  FRAME RATE COUNT (NEGATIVE)                   .
*             ..................................................
*
*       - COMPLETION CALLING SEQUENCE -
*
*           ...
*           ...
*           JSB COMP        COMPLETION CALL
```

```
*          ...                    (NORMAL RETURN)
*          ...
*
*
*       INITIALIZATION
*
INIT    NOP
        CLF 0               DISABLE INTERRUPTS
        LDA INIT,I          LOAD PARAMETER TABLE ADDRESS
        ISZ INIT            BUMP TO REJECT RETURN
        LDA INTAB,1         LOAD DMA Ch. 1 INTERRUPT WORD
        SZA                 IN USE?
        JMP INIT,I          YES, DMA REJECT
        CCA                 A=-1
        STA INTAB,I         SET DMA CH. 1 BUSY
        LDA DJMPA           LOAD JSB DATA,I INSTRUCTION
        STA DSPL            STORE IN INTERRUPT ADDRESS
        LDA IDDIR           LOAD DRIVER INTERRUPT ENTRY ADDRESS
        STA DATA            STORE IN DATA CH. ADDRESS
        LDA B,I             LOAD DISPLAY FILE FWA-1
        STA BASE            SAVE
        INB                 BUMP TO NEXT ENTRY
        LDA B,I             LOAD LIST TABLE POINTER ADDRESS
        STA TADD            SAVE
        STA POINT           INITIALIZE POINTER
        INB                 BUMP TO NEXT ENTRY
        LDA B,I             LOAD FRAME RATE
        STA FTIME           SAVE
        STA COUNT           INITIALIZE COUNT
        JSB TABPR           SET-UP DISPLAY
        JSB BEGIN           START
        ISZ INIT            BUMP TO NORMAL RETURN
        STF 0               ENABLE INTERRUPTS
        JMP INIT,I          RETURN
DJMPA   JSB DATA,I          INTERRUPT ADDRESS INSTRUCTION
IDDIR   DEF DDIR            DRIVER INTERRUPT ENTRY ADDRESS
*
*       COMPLETION
*
COMP    NOP
        CLF 0               DISABLE INTERRUPTS
        JSB PIO             STOP THE DISPLAY
        OCT 177000          ACKNOWLEDGE ALL INTERRUPTS
        OCT 0               STOP AND CLEAR DISPLAY CONTROLLER
        STF DMAC            ABORT CURRENT DMA TRANSFER, IF ANY
        CLC DSPL,C          CLEAR DISPLAY CHANNEL
        CLC DATA,C          CLEAR DATA CHANNEL
        CLC DTCHN,C         CLEAR TABLET CHANNEL
        STA INTAB,I         CLEAR DMA CHANNEL 1
        STF 0               ENABLE INTERRUPTS
        JMP COMP,I          RETURN
*
*       DISPLAY START ROUTNE
*
BEGIN   NOP
        JSB OUTL            START NEXT IMAGE
ARG     OCT 0               IMAGE LIST ADDRESS
        JSB PIO             START DISPLAY
        OCT 176400          RESET, CLEAR AND START DISPLAY
        OCT 0               OUTPUT ONLY
        JMP BEGIN,I         RETURN
*
*       DISPLAY INTERRUPT PROCESSOR
*
```

```
DDIR   NOP
       CLF 0           DISABLE INTERRUPTS
       STA SAVEA       SAVE A-REGISTER
       STB SAVEB       SAVE B-REGISTER
       SSA,ALS         E-REGISTER TO BIT A15; BIT A0=0
       SOC             TEST OVERFLOW BIT
       INA             BIT A0=1 IF O/F SET, =0 IF NOT
       STA SAVED       SAVE E AND O REGISTERS
       LDA SKIP        LOAD "RSS" INSTRUCTION
       STA RSW         INITIALIZE RESTART SWITCH = RSS
       JSB PIO         READ DISPLAY REGISTERS
       OCT 4           BEGIN WITH REGISTER 4
       OCT 2           READ 2 WORDS
       DEF PIR         STORE STARTING AT PIR
       LDA PIR         LOAD PIR REGISTER
       ALF,ALF         PUT INTERRUPTS IN UPPER-A
       AND MCR         MASK FOR ACTIVE CONDITIONS IN MCR
       AND IMASK       MASK FOR PENDING BITS
       STA AKW         SAVE FOR RESET
       JSB PIO         RESET PENDING REQUESTS
AKW    OCT 0           ACKNOWLEDGE WORD
       OCT 0           OUTPUT ONLY
       LDB AKW         LOAD PENDING INTERRUPT BITS
       RBL             POSITION FOR CLOCK
       SSB             CLOCK INTERRUPT?
       JSB ICLOK       YES, PROCESS
       LDB AKW         LOAD PENDING INTERRUPT BITS
       SSB             P-BIT INTERRUPT?
       JSB IPBIT       YES, PROCESS
RSW    NOP             SKIP RESTART? (YES = RSS)
       JSB BEGIN       START THE DISPLAY
       LDA SAVED       LOAD E AND O REGISTER CONTENTS
       CLO             CLEAR O-REGISTER
       SLA,ELA         PUT A15 INTO E-REGISTER, IF A0=0, SKIP
       STO             SET O-REGISTER
       LDA SAVEA       RESTORE A-REGISTER
       LDB SAVEB       RESTORE B-REGISTER
       STF 0           ENABLE INTERRUPTS
       JMP DDIR,I      RETURN
SAVEA  OCT 0           A-REGISTER STORAGE
SAVEB  OCT 0           B-REGISTER STORAGE
SAVED  OCT 0           E AND O REGISTER STORAGE
PIR    OCT 0           PRIORITY INTERRUPT REGISTER
MCR    OCT 0           MODE CONTROL REGISTER
IMASK  OCT 170000      MASK FOR INTERRUPT BITS
*
*      P-BIT PROCESSOR
*
IPBIT  NOP
       JSB TABPR       PROCESS NEXT TABLE ENTRY
       JSB READY       CHECK DISPLAY STATUS
       JMP IPBIT,I     RETURN
*
*      FRAME CLOCK PROCESSOR
*
ICLOK  NOP
       ISZ COUNT       STEP FRAME TIME COUNT
       JMP NOFRM       NOT END FRAME
       LDA CONTR       LOAD CONTROL WORD
       IOR .+1         SET C FLAG IN CONTROL WORD
       STA CONTR       RESTORE
NOFRM  JSB READY       CHECK DISPLAY STATUS
       JMP ICLOK,I     RETURN
COUNT  OCT 0           FRAME TIME COUNTER
FTIME  OCT 0           FRAME TIME RATE
```

```
*
*       TABLE PROCESSOR
*
TABPR   NOP
TAB1    LDA POINT,I     LOAD LIST TABLE POINTER
        ISZ POINT       STEP TO NEXT ENTRY
        SSA             NEGATIVE ENTRY?
        JMP TAB3        YES, CHECK FOR EOT
        SZA,RSS         NO, ENTRY ZERO?
        JMP TAB1        YES, IGNORE
        ADA BASE        NO, ADD BASE ADDRESS
        STA ARG         SET UP NEXT LIST ADDRESS FOR RESTART
        CLA             A=0
        STA RSW         CLEAR RESTART SWITCH = NOP
TAB2    LDA POINT,I     GET NEXT LIST POINTER
        SSA             POSITIVE ENTRY?
        JMP TABPR,I     NO, LEAVE POINTER AT NEG ENTRY
        ISZ POINT       YES, NEXT ENTRY
        JMP TAB2        CHECK
TAB3    CLE,ELA         SHIFT OUT SIGN BIT
        SZA             EOT?
        JMP TAB1        NO, NEGATIVE ENTRY, IGNORE
        LDA CONTR       YES, LOAD CONTROL WORD
        IOR .+2         SET I FLAG IN CONTROL WORD
        STA CONTR       RESTORE
        JMP TABPR,I     RETURN
POINT   OCT 0           LIST ADDRESS POINTER
BASE    OCT 0           DISPLAY FILE FWA-1
TADD    OCT 0           LIST TABLE FWA
*
*       DISPLAY STATUS CHECK ROUTINE
*
READY   NOP
        LDA CONTR       LOAD CONTROL WORD
        CPA .+3         TIME TO RESTART?
SKIP    CLA,RSS         YES
        JMP READY,I     NO, RETURN
        STA CONTR       CLEAR CONTROL WORD
        LDB TADD        LOAD LIST TABLE ADDRESS
        STB POINT       RESET POINTER
        LDB FTIME       LOAD FRAME TIME
        STB COUNT       RESET COUNTER
        STC DTCHN,C     ENABLE TABLET FOR READ
        SFS DTCHN       READY FOR READ?
        JMP *-1         NO, WAIT
        LIA DTCHN       LOAD STATUS WORD
        STA TSTAT       SAVE
        CLE,ERA         ISOLATE PROX. BIT
        LIA DTCHN       LOAD X-COORDINATE WORD
        LIB DTCHN       LOAD Y-COORDINATE WORD
        SEZ,RSS         PEN IN PROX.?
        JMP NPROX       NO
        STA XTHIT       SAVE X-COORDINATE
        STB YTHIT       SAVE Y-COORDINATE
NPROX   CLC DTCHN,C     RESET AND DISABLE TABLET
        LDA TSTAT       LOAD STATUS WORD
        RAR,RAR         POSITION FS BITS
        AND .+15        MASK FOR FS BITS
        CPA 040         STOP BUTTON?
        CCA             YES, SET A=-1
        STA FSNCH       SET FS REGISTER
        LDA USTAT       LOAD STATUS WORD
        SLA             STATUS BIT ON?
        CCA             YES, SET FOR REVERSE
        INA             REVERSE STATUS BIT
```

```
        STA DSTAT       RESTORE STATUS WORD
        JSB TABPR       RESTART DISPLAY
        JMP READY,I     RETURN
S40     OCT 40          STOP CODE
CONTR   OCT 0           CONTROL WORD
TSTAT   OCT 0           STATUS WORD OF TABLET
XTHIT   OCT 0           LAST X-COORDINATE ON TABLET
YTHIT   OCT 0           LAST Y-COORDINATE ON TABLET
FSWCH   OCT 0           FUNCTION SWITCH REGISTER
LSTFS   OCT 0           LAST FUNCTION SWITCH HIT
DSTAT   OCT 0           DISPLAY REFRESH STATUS
*
*       PROGRAMATIC CHANNEL I/O
*
PIO     NOP
        LDA PIO,I       GET OUTPUT WORD
        OTA DSPL        SET UP PIO REGISTER
        STC DSPL        SEND
        ISZ PIO         STEP TO NEXT ARGUMENT
        LDA PIO,I       LOAD WORD COUNT
        SZA             ZERO?
        JMP INPUT       NO, GO ON FOR INPUT
        CLF DSPL        ENABLE DISPLAY
        JMP PIO,I       RETURN
INPUT   CMA,INA         SET COUNT
        STA ICNT        SAVE
        LDA .+12        LOAD "INP" BIT
        OTA DSPL        SEND TO PIO REGISTER
        ISZ PIO         STEP TO NEXT ARGUMENT
        LDA PIO,I       LOAD STORE ADDRESS
        STA IADR        SAVE
PIO1    LIA DSPL        READ DISPLAY
        STC DSPL        REQUEST NEXT
        STA IADR,I      STORE INPUT
        ISZ IADR        STEP STORE ADDRESS
        ISZ ICNT        ALL WORDS INPUT?
        JMP PIO1        NO
        ISZ PIO         STEP TO RETURN
        JMP PIO,I       RETURN
IADR    OCT 0           STORE ADDRESS OF INPUT DATA
ICNT    OCT 0           INPUT WORD COUNT (NEGATIVE)
*
*       DATA CHANNEL DISPLAY LIST OUTPUT
*
OUTL    NOP
        LDA CLEAR       LOAD CLEAR DISPLAY BITS
        OTA DSPL        PUT IN PIO REGISTER
        STC DSPL        SEND
        CLC DATA,C      CLEAR DMA I/O TO DISPLAY
        LDA OUTL,I      GET LIST ADDRESS
        CLC DMA         SELECT DMA ADDRESS REGISTER
        OTA DMA         SET LIST ADDRESS
        STC DMA         SELECT DMA COUNT REGISTER
        LDA DMACW       GET "DISPLAY OUTPUT" CONTROL WORD
        OTA DMAC,C      SEND CONTROL WORD TO DMA
        LDA MAX         LOAD MAXIMUM WORD COUNT
        OTA DMA         SET WORD COUNT
        STC DMAC        START DMA
        ISZ OUTL        STEP TO RETURN
        JMP OUTL,I      RETURN
CLEAR   OCT 1200        PIO REGISTER CLEAR BITS
MAX     OCT -77777      MAXIMUM WORD COUNT FOR DMA
DMACW   DEF DATA,I      DMA CONTROL WORD
*
A       EQU 0           A-REGISTER
```

```
B       EQU 1           B=REGISTER
DMA     EQU 2           DMA CHAN. 1 CW2 AND CW3
DMAC    EQU 0           DMA CHAN. 1 CW1 AND CHAN. 1 CONTROL
DSPL    EQU 1B          DISPLAY PROGRAM I/O CHANNEL
DATA    EQU 11B         DISPLAY DATA CHANNEL
DTCHN   EQU 25B         DATA TABLET I/O CHANNEL
.       EQU 55B         BASE PAGE CONSTANTS, .=0, + OR - 1 TO 10
INTAB   EQU 2000B       INTERRUPT TABLE ADDRESS
        END


FTN4
      SUBROUTINE FN123(IDUM,NSIZ,ISIZ,IRLT,IDUM,IDF)
C.....ROUTINE TO SET UP HEADER INFORMATION IN DISPLAY FILE
C.....FOR NESTED GRADE.
      DIMENSION IDUM(1),ISIZ(1),IDUM(1),IHED(13),ITEX1(4),
     1  ITEX2(6),ITEX3(8),ITEX4(4),ITEX5(5)
      DATA IHED/4000B,1774000B,1774010B,400210B,733000B,0,77760B,0,1,
     1  400000B,1600110B,400140B,777610B/,LIMIT/2500/
      DATA ITEX1/2HPI,2HEC,2HE ,2H @/
      DATA ITEX2(6)/2H$ /,ITEX4(4)/2H@ /
      DATA ITEX3/2HSI,2HZE,2HS ,2HIN,2H G,2HRA,2HDE,2H@ /
      DATA ITEX5/2H#0,2HLE,2H #,2H  ,2H #/
      DATA IX1/-1950/,IX2/-1710/,IX5/-1250/,ICX/272/
      DATA IY1/1740/,IY3/1070/,IY4/1600/,ICY/70/
C.....ENTRY 1
      IDF = NSIZ+7
      IDUM(1) = IDF
C.....INITIALIZE REGISTERS
      DO 10 I=1,13
      II = I-1
   10 IDUM(IDF+II) = IHED(I)
      IDF = IDF+13
      IF(IDF.GT.LIMIT) CALL ERR(4,0)
C.....DISPLAY "PIECE  XXXXXXXXXX"
      CALL DISCH (IX1,IY1,1,0,ITEX1,IDUM,IDF,LIMIT)
      DO 20 I=1,5
   20 ITEX2(I) = IDNUM(I)
      CALL DISCH (IX2,IY1,1,0,ITEX2,IDUM,IDF,LIMIT)
C.....DISPLAY "SIZES IN GRADE"
      CALL DISCH (IX1,IY3,1,0,ITEX3,IDUM,IDF,LIMIT)
C.....PICK UP SIZE NAMES AND DISPLAY IN ROWS OF 14
      IX = IX1
      IY = IY4
      DO 40 I=1,NSIZ
      CALL SIZEP (2,IRLT,ITEX4,ISIZ(I))
      IF(IX.LT.1860) GO TO 30
      IX = IX1
      IY = IY-ICY
   30 CALL DISCH (IX,IY,1,0,ITEX4,IDUM,IDF,LIMIT)
      IX = IX+ICX
   40 CONTINUE
C.....DISPLAY "RULE #   "
      CALL DISCH (IX5,IY1,1,0,ITEX5,IDUM,IDF,LIMIT)
      IDUM(IDF) = 100000B
      IDF = IDF+1
      IF(IDF.GT.LIMIT) CALL ERR(4,0)
C.....ENTRY 2
      IDUM(2) = -IDF
      CALL DISVL (IX5+ICX,IY1,0.0,4,-1,1,NSIZ+1,IDUM,IDF,LIMIT)
      IDUM(IDF) = 0
      IDF = IDF+1
      IF(IDF.GT.LIMIT) CALL ERR(4,0)
C .....ENTRY 3,TRACKING CROSS
      IDUM(3) = IDF
```

```
      CALL CROSS(1600,1600,IDF,IDUM)
      IDF=IDF+10
      IF(IDF.GT.LIMIT) CALL ERR(4,0)
C.....ENTRY 4,ASTERISK
      CALL ASTAR(0,0,IDF,IDUM)
       IDF = IDF+11
      IF(IDF.GT.LIMIT)CALL ERR(4,0)
C.....ENTRY 5, WILL BE BOX
      IDUM(5) = IDF
      IDUM(IDF)=0
      DO 50 I=1,40
   50 IDUM(IDF+I) = 0
      IDF = IDF + 40
      RETURN
      END
      ENDS


FTN4
      SUBROUTINE ENPEC(IPEC,IXDF,IYDF,IDUM,IDF)
C.....ROUTINE TO SET UP HEADER INFORMATION FOR A PIECE, "IPEC"
      DIMENSION IDUM(1), IHED(6)
      DATA IHED/40002B,0,77760B,0,1,400100B,0,1/
C.....SET UP IMAGE NUMBER
      IHED(2) = IPEC*256
C.....SET OFFSET IN DISPLACEMENT REGISTERS
      IHED(7) = IXDF*16
      IHED(8) = IYDF*16+1
      DO 10 I=1,8
      II = I-1
   10 IDUM(II+IDF) = IHED(I)
      IDUM(IPEC+5) = IDF
      IDF=IDF+8
      RETURN
      END
      ENDS


ASMB
      NAM ERR0,7
      ENT ERR0
      EXT .ENTR,ERR00,EXEC
A     NOP
B     NOP
ERR0  NOP
      JSB .ENTR
      DEF A
      CLB
      LDA A,I
      DIV D10
      STA TENS
      ADA ERTAB
      LDA B,I
      RRR 8
      STA LER
      LDA TENS
      MPY D10
      CMA,INA
      ADA A,I
      ADA ERTAB
      LDA B,I
      LDB LER
      XRL B
      CLA,CLE
      JSB ERR00
```

```
        LDA 0,I
        SZA
        JMP ERR0,I
        JSB EXEC
        DEF **3
        DEF R23
        DEF MODAL
        JSB EXEC
        DEF **2
        DEF **2
        JMP ERR0,I
        DEC 5
R10     DEC 10
R23     DEC 23
ERR     BSS 1
TENS    BSS 1
MODAL   ASC 3,PNORMD
ER      ASC 1,ER
ERTAB   DEF **1
        ASC 1,0
        ASC 1,1
        ASC 1,2
        ASC 1,3
        ASC 1,4
        ASC 1,5
        ASC 1,6
        ASC 1,7
        ASC 1,8
        ASC 1,9
        END
```

```
FTN4
        SUBROUTINE ERR(N,M)
        DIMENSION IPROG(3),IFILE(3),IBUF(2),IPLAT(3)
        DATA IPROG/2HAP,2HR0,2HD /
        DATA IFILE/2HSE,2HCT,2HR /
        DATA IPLAT/2HPN,2HOM,2H00/
        DATA IBUF/2,2/
        CALL ERR(N,1)
        IF(N.NE.0.AND.M.EQ.1) RETURN
        IBUF(2) = 2
        IF(N.EQ.0) IBUF(2) = 7
        CALL EXEC (23,IPLAT)
        CALL EXEC (15,56,IBUF,2,IFILE,0)
        CALL EXEC (10,IPROG)
        END
        ENDS
```

```
FTN4
        SUBROUTINE FALL (LMTX,LMTY,IX,IY,NXY,INX,NIN,JTYP)
C.....FEB. 14,1973
C.....MARGARET ROMAN
C.....PROGRAM MARK
C.....VERSION 2.0
C.....ROUTINE TO FIND A STRING OF POINTS
C.....THAT FALLS IN THE AREA BOUNDED BY LMYX,LMTY.  THE
C.....POLY-STRING OF POINTS THAT IS SEARCHED IS IX,IY.
C.....NXY IS THE NUMBER OF POINTS SENT IN IX,IY AND IS
C.....RETURNED AS THE INDEX OF THE FIRST POINT FOLLOWING
C.....THE STRING FOUND.  "INX" IS THE INDEX OF
C.....THE FIRST POINT IN THE STRING.  "NIN" IS THE
C.....NUMBER OF POINTS IN THE STRING.  "JTYP" IS 1
```

```
C..........FOR AN OPEN STRING OR 2 FOR A CLOSED STRING.
      DIMENSION LMTX(2),LMTY(2),IX(1),IY(1),MFLG(8)
      DATA MFLG/8,4,2,1,9,6,7,5/
      INDX = 0
      IFTT = 0
      NBEG = 2
      TX1 = LMTX
      TX2 = LMTX(2)
      TY1 = LMTY
      TY2 = LMTY(2)
      XX1 = IX
      YY1 = IY
      INX=1
      IF(XX1.LT.TX1.OR.XX1.GT.TX2) GO TO 40
      IF(YY1.LT.TY1.OR.YY1.GT.TY2) GO TO 40
C..........FIRST POINT IN AREA, SEARCH BACKWARDS TO FIND
C..........A POINT OUT OF AREA.
      INDX = 1
      DO 10 I=NXY,2,-1
      XXI = IX(I)
      YYI = IY(I)
      IF(XXI.LT.TX1.OR.XXI.GT.TX2) GO TO 20
      IF(YYI.LT.TY1.OR.YYI.GT.TY2) GO TO 20
   10 CONTINUE
      INX = 1
      NIN = NXY
      RETURN
C..........FOUND FIRST POINT NOW SEARCH FOR THE LAST POINT
   20 INDX = INDX+NXY-I+1
      INX = I
      IFTT = 1
C..........SEARCH FORWARDS TO FIND A POINT OUT OF AREA
   40 NEND = NXY
      IF(JTYP.EQ.2) NEND=NXY+1
      DO 700 II=NBEG,NEND
      I = INDEX(II,NXY)
      XXI = IX(I)
      YYI = IY(I)
      IF(XXI.LT.TX1.OR.XXI.GT.TX2) GO TO 60
      IF(YYI.LT.TY1.OR.YYI.GT.TY2) GO TO 60
C..........POINT IN AREA, SAVE IT AND THE PREVIOUS POINT IF
C..........THIS WAS THE FIRST POINT FOUND IN AREA.
      IF(IFTT.EQ.1) GO TO 50
      INX = II-1
      INDX = 1
   50 IFTT = 1
      INDX = INDX+1
      IF(II.EQ.NEND.AND.INX.NE.1) INDX = INDX+1
      GO TO 690
   60 IF(IFTT.EQ.1) GO TO 720
C..........POINT OUTSIDE AREA, CHECK LINE SEGMENT CONNECTING
C..........PREVIOUS PAIR OF POINTS
      IFLG = 0
      IF(XX1.LT.TX1) IFLG = 1
      IF(XX1.GT.TX2) IFLG = 2
      IF(YY1.LT.TY1) IFLG = IFLG+3
      IF(YY1.GT.TY2) IFLG = IFLG+6
      JFLG = 0
      IF(XXI.LT.TX1) JFLG = 1
      IF(XXI.GT.TX2) JFLG = 2
      IF(YYI.LT.TY1) JFLG = JFLG+3
      IF(YYI.GT.TY2) JFLG = JFLG+6
      IF(IFLG.EQ.JFLG) GO TO 690
      MI = MFLG(IFLG)
```

```
      MJ = MFLG(JFLG)
      M1 = IABS(MI-MJ)
      IF(M1.EQ.1.OR.M1.EQ.7) GO TO 690
      IF(M1.EQ.4) GO TO 102
      IF(MOD(MI,2).EQ.1.AND.MOD(MJ,2).EQ.1) GO TO 690
  102 X1 = XX1
      Y1 = YY1
      X2 = XXI
      Y2 = YYI
      IF(M1.EQ.4.AND.MOD(MI,2).EQ.0) GO TO 120
  103 IF(Y1.NE.Y2) GO TO 104
      IF(Y1.LE.TY2.AND.Y1.GE.TY1) GO TO 120
      GO TO 690
  104 IF(X1.NE.X2) GO TO 105
      IF(X1.LE.TX2.AND.X1.GE.TX1) GO TO 120
      GO TO 690
  105 SL = (Y2-Y1)/(X2-X1)
      B = Y2-SL*X2
      V = SL*TX1+B-TY1
      ISG = SIGN(1.0,V)
      DO 110 K=1,2
      DO 110 J = 1,2
      IF(K.EQ.1.AND.J.EQ.1) GO TO 110
      V = SL*LMTX(K) + B -LMTY(J)
      JSG = SIGN(1.0,V)
      IF(ISG.NE.JSG) GO TO 120
  110 CONTINUE
C.....LINE CROSSES AREA
  120 INX = II-1
      IFIT = 1
      INDX = 1
  150 INDX = INDX+1
  690 XX1 = XXI
      YY1 = YYI
  700 CONTINUE
  710 NIN = 0
      IF(INDX.GT.1) NIN = INDX
      RETURN
  720 INDX = INDX+1
      GO TO 710
      END
      END$

FTN4
      SUBROUTINE FIELD
C.....FEB.20,1973
C.....LYNN KIRSTEN, MARGARET ROWAN
C.....VERSION 2.0
C
      COMMON JBP(15), JBG(4)
      JBG(1)=JFLD(12,4)
      JBG(2)=JFLD(8,9)
      JBG(3)=JFLD(15,8)
      JBG(4)=JFLD(7,7)
      JBP(1)=JFLD(15,7)
      JBP(2)=JFLD(6,8)
      JBP(3)=JFLD(5,1)
      JBP(4)=JFLD(4,1)
      JBP(5)=JFLD(3,4)
      JBP(6)=JFLD(15,9)
      JBP(7)=JFLD(6,1)
      JBP(8)=JFLD(9,1)
      JBP(9)=JFLD(15,4)
      JBP(10)=JFLD(11,12)
      RETURN
      END
      END$
```

```
      FTN4
      SUBROUTINE FINI(N)
      COMMON LOUT,ITYPE,ORIGX,ORIGY,LDIRC,XPEND,AXOLD,AYOLD,
     1    IPNF(3),IPNM(3),XMAX,YMAX,SCAL,XLAST,YLAST,LASTP,LASTA,
     2    LASTT,IPLT(3),JTYPE,JOB,JPOS,JTOOL,JHOME,JIZ,SPACE,IOPTN
     3    ,NPLAT,JH4,JH1,JH2,JH3,JH4
      DIMENSION IFILE(3),IBUF(128),IFIL(3)
      DIMENSION IDUM(10)
      EQUIVALENCE (IDUM(1),VAL1),(IDUM(3),VAL2),(IDUM(5),VAL3)
      EQUIVALENCE (IDUM(7),VAL4),(IDUM(9),VAL5)
      DATA IFIL/2HWP,2HRD,2HD /
      DATA IFILE /2HSE,2HCT,2HR /
      IBUF=2
      IBUF(2)=N
      CALL DDSC (1)
      CALL EXEC(15,1102B,IBUF,2,IFILE,0)
C.....UPDATE THE HEADER FOR FILE "PLOTF"
      IF(N.NE.1) GO TO 200
      CALL DDSC(0)
      CALL EXEC(14,1102B,IBUF,128,IPLT,0)
      M=10
      IF(ITYPE.EQ.2HCU) M=30
      VAL1=ORIGX
      VAL2=ORIGY
      VAL3=AXOLD
      VAL4=AYOLD
      VAL5=XPEND
      L=1
      DO 100   I=1,5
      K=M+L
      IBUF(K)=IDUM(L)
      IBUF(K+1)=IDUM(L+1)
      L=L+2
  100 CONTINUE
      IBUF(M+11)=LDIRC
      CALL EXEC(15,1102B,IBUF,128,IPLT,0)
  200 CONTINUE
      CALL DDSC (0)
      CALL EXEC(10,IFIL)
      RETURN
      END
      ENDS


FTN4
      PROGRAM FIX
      DIMENSION IBUF(128),NAM(3),LAB(3)
      WRITE(1,100)
  100 FORMAT ("ENTER PLATTER")
      READ(1,101) LAB
  101 FORMAT (3A2)
      WRITE(1,102)
  102 FORMAT ("ENTER FILE")
      READ (1,101) NAM
   10 WRITE(1,103)
  103 FORMAT ("ENTER SECTOR NUMBER")
      READ (1,*) NSECT
      IF(NSECT.EQ.999) GO TO 30
      CALL EXEC (23,LAB)
      CALL EXEC (14,66,IBUF,128,NAM,NSECT)
      IF(ISSW(0).LT.0) WRITE(1,104) IBUF
  104 FORMAT (1007)
   20 WRITE(1,105)
  105 FORMAT ("ENTER WORD AND VALUE IN FORMAT I3,1X,O6",/,"$@")
      READ (1,105) WRD,IVAL
```

```
      IF(NWD.EQ.0) GO TO 40
  100 FORMAT (I3,1X,L6)
      IBUF(NWD) = IVAL
      GO TO 20
   30 STOP
   40 CALL LASD (15,66,IBUF,120,NAM,NSECT)
      GO TO 10
      END
      ENDS


ASMB
      NAM FLIP,7
      ENT FLIP
      EXT .ENTR
IMAGE NOP             IMAGE LIST ADDRESS
FLIP  NOP
      JSB .ENTR
      DEF IMAGE
*
MODE  LDA IMAGE,I     LOAD INSTRUCTION
      CPA RHEAD       "RELATIVE" INSTRUCTION HEADER?
      JMP REL         YES, PROCESS
      CPA IHEAD       "INCREMENTAL" INSTRUCTION HEADER?
      JMP INCR        YES, PROCESS
      SZA             TERMINATE SCAN?
      JMP FLIP,I      YES, RETURN
      ISZ IMAGE       BUMP ADDRESS
      JMP MODE        CONTINUE SCAN
*
INCR  ISZ IMAGE       BUMP ADDRESS
      LDA IMAGE,I     LOAD INSTRUCTION
      CLB             B=0
      RRR 1           ROTATE TERMINATE BIT TO B
      STB TBIT        SAVE
      CLB             B=0
      RRR 7           *
      BLF,BLF         *  PUT Y MOVE IN LOWER B
      RRR             *
      CMB,INB         NEGATE Y MOVE
      BLF,BLF         *
      RBL             *  RESTORE Y MOVE
      RRL 7           *
      LDB TBIT        LOAD TERMINATE BIT
      RRL 1           INSERT IN INSTRUCTION
      STA IMAGE,I     RESTORE INSTRUCTION
      SLA,RSS         TERMINATE BIT SET?
      JMP INCR        NO, CONTINUE SCAN
      ISZ IMAGE       BUMP ADDRESS
      JMP MODE        CHECK NEXT INSTRUCTION
*
REL   ISZ IMAGE       BUMP ADDRESS
      LDA IMAGE,I     LOAD INSTRUCTION
      SLA             Y INSTRUCTION?
      JMP CTERM       NO, SEE IF TERMINATE SET
      CLB             B=0
      RRR 4           PUT CONTROL BITS IN B
      CMA,INA         NEGATE Y MOVE
      RRL 4           RESTORE CONTROL BITS
      STA IMAGE,I     RESTORE INSTRUCTION
CTERM AND TERM        ISOLATE TERMINATE BITS
      CPA TERM        SET?
      RSS             YES, PROCESS
      JMP REL         CONTINUE SCAN
      ISZ IMAGE       BUMP ADDRESS
```

```
            JMP MODE        CHECK NEXT INSTRUCTION
      *
      RHEAD OCT 1xxxxx      "RELATIVE" INSTRUCTION HEADER
      IHEAD OCT 1xxxxx      "INCREMENTAL" INSTRUCTION HEADER
      TERM  OCT 14          TERMINATE BIT MASK
      TBIT  OCT 0           TERMINATE BIT STORAGE
            END

FTN4
      SUBROUTINE FLNAM(LET,NUM,NAM)
C.....SUBROUTINE TO SET UP ASCII CHARACTER NAME FOR
C.....DISK FILES AND PLATTER NAMES.  PARAMETERS ARE:
C.....LET = CODE FOR PREFIX LETTER TO BE USED
C                     1 -- STYLE FILE PREFIX (*)
C                     2 -- ROLE TABLE PREFIX (#)
C                     3 -- PLATTER NAME PREFIX (PN)
C                     ASCII CHARACTER -- MARKER FILE PREFIX
C                          MUST BE LEFT-JUSTIFIED IN WORD.
C.....NUM = NUMBER FOR THE FILE NAME, FOR MARKER
C                     FILES SHOULD BE STYLE NUMBER.
C.....NAM(3) = 5 WORD ARRAY CONTAINING THE ASCII
C                       NAME.
      DIMENSION INUM(4),NAM(3)
      DATA LSTY/2H* /,LROL/2H# /,LPLAT/2HPN/,IMSK/177400B/,
     1     NBLK/4Hxx/,NUMX/0xxB/,LSHFB/400B/
      IF(NUM.GT.9999) CALL ERR(24,0)
      ITMP1 = NUM
      DO 5 I=4,1,-1
      ITMP = ITMP1/10
      INUM(I) = ITMP1-(ITMP*10)+NUM0
    5 ITMP1 = ITMP
      IBEG = 2
      IF(LET.EQ.1) NAM(1) = IAND(LSTY,IMSK)
      IF(LET.EQ.2) NAM(1) = IAND(LROL,IMSK)
      IF(LET.EQ.3) GO TO 7
      NAM(1) = LPLAT
      IBEG = 1
      GO TO 15
    7 IF(LET.LE.3) GO TO 10
      NAM(1) = IAND(LET,IMSK)
   10 NAM(1) = IOR(NAM(1),INUM(1))
   15 J = 2
      DO 20 I=IBEG,4,2
      IF(I.EQ.4) GO TO 25
      NAM(4) = (INUM(I)*LSHFB)+INUM(I+1)
   20 J = J+1
      RETURN
   25 NAM(J) = (INUM(4)*LSHFB)+NBLK
      RETURN
      END
      ENDS


FTN4
      SUBROUTINE FLPAL(IMGNO)
C.....OCT. 17,1972
C.....MARGARET ROMAN
C.....PROGRAM MARK
C.....VERSION 2.0
C.....ROUTINE FLIPS PIECES IN THE GROUP WITH "IMGNO"
      COMMON MOUT,LABF(3),LABR(3),NOSTY,INSTY(31),JBS(15),NAMST(3),
     1 MKUP(5),NSCNO,JBM(20),NAMMK(3),JBG(4),NAMGO(3),JBP(10),
     2 IUMK(5),IOOP,NPEC,NSHO,FACTR,NSIZ,ISIZ(42),WIDTH,EFF,SX,SY,KSTL,
     3 YARDS,NAP,KINT,KFLG,KINC,KSIZ,KXMN,KYMN,KANG,KXMX,KYMX,KOPT,
     4 KAO,KAOP,KOF,CSMX,MKWD,CSMN,NKGE,IMGP,KREL,ITYPE,
```

```
     5 NOIMG,IOFSZ,IMGY,IMGE,IMGTC,IMKO1,IMKO2,IMENU,IDXP,IDYP,
     6 IXC,IYC,IXL,IXR,IYU,IYL,CS,MDELX,MDELY,JFLG,MLW,MUP,ISLX,ISLY,
     7 TOTAL,MYL,XPC,YPC,IMGLP,IMESG,IFOG,ISOFF,SOFF,KSLH,
     8 IMGPS,MAXOF,KSLV,KTC,IOPTN,IAREA,ISIZE,KCSR,KYES,KPSIZ,KPLL,
     9 IT1,IT2,IT3,IT4,IT5,IT6,IT7,IT8,IT9,IT10,
     A IOFSET,KPLH,IFLP,ICOST,IMGPK,
     B T1,T2,T3,T4,T5,T6,T7,T8,T9,T10,
     C JSTY,JFG,JSG,JSHO,JRT,
     D ILPS,IHO,IHO,IOV,IGR,IHL,ITCW,ITCCW,
     E IRCW,IPCCW,ISL,IRF,IFL,IPK,ICOM,
     F IDUM(1)
      NN = 5
      IF(JLOF(JFLG(9,1),IOPTN).EQ.1) NN=NN+3
      KSIZE = IDUM(KSIZ+IMGNO)
C.....FIND PIECES IN THE FLIP GROUP
      DO 100 I=1,NOIMG
      IF(IDUM(KSIZ+I).NE.KSIZE) GO TO 100
C.....IGNOR SLAVE PIECES
      IF(JLOF(JHM(24),IDUM(KFLG+I)).EQ.1) GO TO 100
      IF(JLOF(JFG,IDUM(KFLG+I)).EQ.0) GO TO 100
C.....PIECE IN FLIP GROUP
      IAD = IDUM(KADP+I)
      MM=NN
      IF(NN.NE.6.AND.JLOF(JHM(15),IDUM(KFLG+NPEC+I)).NE.0) MM=NN+4
      CALL FLIP (IDUM(IAD+MM))
      IF(ISLY.EQ.0) GO TO 50
C.....ADJUST STRIPE LINES
      ITMP = KSTL+1
      IF(IDUM(ITMP).EQ.9999) GO TO 50
      IDUM(ITMP) = -IDUM(ITMP)
C.....MODIFY ANGLE
   50 NANG = IDUM(KANG+I)
      MMM = NANG+1
      IF(IABS(NANG).GE.3600) MMM = -MMM
      IDUM(KANG+1) = NANG + ISIGN(3600,MMM)
      CALL VISIB(I,JORG,JX,JY)
      CALL ORGIN(JX,JY,JORG+10,I)
      CALL MODI(2,JX,JY,JORG,I)
      CALL LITE (1,KADP+I)
  100 CONTINUE
      RETURN
      END
      END$


FTN4
      SUBROUTINE FUNCH(M)
C.....ROUTINE TO CONVERT CODE FROM FUNCTION BOX TO DESIRED
C.....CODES FOR MARK
      MSK = 4
      CALL FSHIT(M,MSK)
      M = M
      RETURN
      END
      END$


FTN4
      SUBROUTINE GENXY (KK,IX,IY,NXY,IFP,ISTAT)
C.....DEC. 11,1972
C.....MARGARET ROWAN
C.....PROGRAM: MARK
C.....VERSION 2.3
C.....GENERATES X AND Y COORDINATE DATA FOR A PIECE
C.....IN ABSOLUTE MARKER UNITS.  DATA IS FIGURED FROM
```

```
C..........THE DISPLAY FILE INSTRUCTIONS.
      DIMENSION IX(1),IY(1)
      COMMON MOUT,LABF(3),LABR(3),NOSTY,INSTY(31),JBS(15),NAMST(3),
     1 MXUP(5),NSCMD,JBM(25),NAMMK(3),JBG(4),NAMGD(3),JBP(10),
     2 IDUM(5),IDOP,NPEC,NSHD,FACTR,NSIZ,ISIZ(42),WIDTH,EFF,SX,SY,KSTL,
     3 YARDS,MAP,KINT,KFLG,KINC,KSIZ,KXMN,KYMN,KANG,KXMX,KYMX,KDPT,
     4 KAD,KADP,KDP,CSMX,MKWD,CSMN,NRGE,IMGP,KREL,ITYPE,
     5 NOIMG,IOFSZ,IMGY,IMGF,IMGIC,IMKO1,IMKO2,IMEND,IDXP,IDYP,
     6 IXC,IYC,IXL,IXR,IYU,IYL,CS,MDELX,MDELY,JFLG,MLW,MUP,ISLX,ISLY,
     7 TOTAL,MYL,XPC,YPC,IMGLP,IMFSG,IEOG,ISOFF,SOFF,KSLH,
     8 IMGPS,MAXOF,KSLV,KTC,IOPTN,IAREA,ISIZE,KCSR,KYES,KPSIZ,KPLL,
     9 IT1,IT2,IT3,IT4,IT5,IT6,IT7,IT8,IT9,IT10,
     A IOFSET,KPLM,IFLP,ICUST,IMGPK,
     B T1,T2,T3,T4,T5,T6,T7,T8,T9,T10,
     C JSTY,JFG,JMG,JSHD,JRT,
     D ILPB,IRD,IRD,IUV,IBR,IBL,ITCW,ITCCW,
     E IRCW,IRCCW,ISL,IRE,IFL,IPK,ICOM,
     F IZOO(1)
      LMI = ISTAT
      NN=0
      IF(JLOF(JBM(15),IOPTN).NE.0) NN=NN+3
      IF(JLOF(JBM(15),IDUM(KFLG+NPEC+KK)).NE.0) NN=NN+4
      IVALR = JFLD(15,12)
      IXL = JFLD(15,7)
      IYL = JFLD(7,7)
      ISTAT = 0
      NXY = 0
      IF(ISLX+ISLY.NE.0.AND.JLOF(JBM(15),IDUM(KFLG+KK)).EQ.1)
     1  GO TO 200
      MM = IABS(IDUM(KADP+KK))
      ICS = 1
      IFP = 1
      IF(JLOF(JBM(11),IDUM(KFLG+KK)).NE.0) IFP=1
      IF(IABS(IDUM(KANG+KK)).GE.3600) IFP=1-IFP
      SL = 0.0
      NDELX = 0
      NDELY = 0
      IDELX = 0
      IDELY = 0
      NCX=JLOF(IVALR,IDUM(MM+4))
      IF(NCX.GT.37777B)NCX=IOR(NCX,170000B)
      NCX=NCX*ICS
      NCY=JLOF(IVALR,IDUM(MM+5))
      IF(NCY.GT.37777B)NCY=IOR(NCY,170000B)
      NCY=NCY*ICS
      IDX = IDUM(KXMX+KK)
      IF(NCX.GT.0) IDX = IDUM(KXMN+KK)
      IDY = IDUM(KYMX+KK)
      IF(NCY.GT.0) IDY = IDUM(KYMN+KK)
      NCX = NCX+IDX
      NCY = NCY+IDY
      NPTR=MM+NN
    5 IF(IAND(70000B,IDUM(NPTR)).EQ.100000B) GO TO 15
      NPTR = NPTR+1
      GO TO 5
C..........DETERMINE TYPE OF INSTRUCTION
   10 NPTR = NPTR+1
   15 SL=0.0
      NDELX = 0
      NDELY = 0
      IWD = IDUM(NPTR)
      INST = IAND(70000B,IWD)
      IF(INST.NE.100000B) GO TO 100
      MODE = IAND(17B,IWD)
      IF(MODE.EQ.17B) GO TO 100
      IF(MODE.NE.0) GO TO 40
```

```
C..........RELATIVE MOVE
   20 NPTR = NPTR+1
      IND = IOUM(NPTR)
      IF(IND.EQ.0) GO TO 100
      IDELT = JLOF(IVALR,IND)
      IF (IDELT.GT.37778) IDELT=IOR(IDELT,177000B)
      IDELT = IDELT*ICS
      ITYP = IAND(3B,IND)
      IF(ITYP.EQ.1) NDELX = IDELT
      IF(ITYP.EQ.2) NDELY = IDELT
      IOPER = IAND(14B,IND)
      IF(IOPER.EQ.0) GO TO 20
      IF(SL.EQ.0.0) GO TO 25
      SSL = FLOAT(NDELY)/FLOAT(NDELX)
      IF(NDELX.EQ.0) SSL = 99999.0
      IF(SL.EQ.SSL) GO TO 27
   25 NXY = NXY+1
      IF(NXY.GT.LMT) GO TO 120
      SL = FLOAT(NDELY)/FLOAT(NDELX)
      IF(NDELX.EQ.0) SL = 99999.0
   27 IDELX = IDELX+NDELX
      IDELY = IDELY+NDELY
      IF(NDELX.EQ.0.AND.NDELY.EQ.0) GO TO 28
      IX(NXY) = IDELX
      IY(NXY) = IDELY
      GO TO 29
   28 NXY = NXY-1
   29 IF(IOPER.EQ.12) GO TO 10
      NDELX = 0
      NDELY = 0
      GO TO 20
C..........INCREMENTAL MOVE
   40 NPTR = NPTR+1
      IND = IOUM(NPTR)
      IF(IND.EQ.0) GO TO 100
      IOPER = IAND(1,IND)
      NDELX = JLOF(IXI,IND)
      NDELY = JLOF(IYI,IND)
      IF(NDELX.GT.777B)NDELX=IOR(NDELX,177600B)
      IF(NDELY.GT.777B)NDELY=IOR(NDELY,177600B)
      IDELX = IDELX+NDELX*ICS
      IDELY = IDELY+NDELY*ICS
      IF(SL.EQ.0.0) GO TO 45
      SSL = FLOAT(NDELY)/FLOAT(NDELX)
      IF(NDELX.EQ.0) SSL = 99999.0
      IF(SL.EQ.SSL) GO TO 47
   45 NXY = NXY+1
      IF(NXY.GT.LMT) GO TO 120
      SL = FLOAT(NDELY)/FLOAT(NDELX)
      IF(NDELX.EQ.0) SL = 99999.0
   47 IF(NDELX.EQ.0.AND.NDELY.EQ.0) GO TO 48
      IX(NXY) = IDELX
      IY(NXY) = IDELY
      GO TO 49
   48 NXY = NXY-1
   49 IF(IOPER.EQ.1) GO TO 10
      GO TO 40
C..........ADD ON MARKER POSITION
  100 NXY=NXY-1
      DO 110 I=1,NXY
      IX(I) = IX(I)+NCX
  110 IY(I) = IY(I)+NCY
      RETURN
  120 ISTAT = 1
      RETURN
```

```
C..........BLOCKED PIECE
  200 NXY = 4
      IX     = IDUM(KXMN+KK)
      IY     = IDUM(KYMN+KK)
      IX(2)  = IX
      IY(2)  = IDUM(KYMX+KK)
      IX(3)  = IDUM(KXMX+KK)
      IY(3)  = IY(2)
      IX(4)  = IX(3)
      IY(4)  = IY
      IPP = 0
      END
      END$


FTN4
      SUBROUTINE GRADP (ITYPE,IDNOT,NSIZ,ISIZ,ISTAT,INC)
C..........OCT. 9,1972
C..........MARGARET ROWAN
C..........PROGRAMS:CHECK,PGRAB
C..........VERSION 2.0
C..........SUBROUTINE TO GRADE PIECE DATA AND STORE EITHER
C..........IN "GRADE" OR "TPELS".
      DIMENSION NAMP(3),NAMG(3),IDNOT(6),INC(1),
     1 ISIZ(1),IBUF(1536),XBUF(640),IAREA(2),IPREM(2),XGL(2),YGL(2)
      DIMENSION XGRF(9),IXMLT(6)
      DIMENSION NAMR(3)
      COMMON JBP(10),JBG(4)
      EQUIVALENCE (IAREA,AREA),(IPREM,PREM)
      DATA LABG/0/,LMTSC/12/
      DATA LOUT/6/,LIN/1/
      DATA XGRF/32.0,64.0,128.0,256.0,512.0,1024.0,25.4,254.0,8096.0/
      DATA NAMR/2HRO,2HLE,2HS /
      IMGRP=JFLD(15,6)
      NAMG = 3000
       LMLT=0
      IGL = 1
      IF(T1 = 1
      ITYP = IABS(ITYPE)
      IF(ISSW(2).LT.0) WRITE(LOUT,9001) ITYP,IDNOT,NSIZ,
     1 (ISIZ(K),INC(K) ,K=1,NSIZ)
 9001 FORMAT (12,3A2,/,3(1X,I7,/))
      ISTAT = 0
C..........SET UP FILE NAMES AND PLATTER NAMES
      IF(ITYP.EQ.0) GO TO 10
C..........PERMANENT PIECE
      NAMP(1)=2HPI
      NAMP(2)=2HEC
      NAMP(3)=2HE
      IF(ITYPE.LT.0) GO TO 15
      NAMG(1) = 2HGR
      NAMG(2) = 2HAD
      NAMG(3) = 2HE
      GO TO 20
C..........TEMPORARY PIECE
   10 NAMP(1) = 2HTP
      NAMP(2) = 2HEC
      NAMP(3) = 2HS
   15 NAMG(1) = 2HCH
      NAMG(2) = 2HEK
      NAMG(3) = 2H
   20 LABP = ITYP
C..........SET UP BEGINNING SECTOR FOR PIECE DATA
      IF(ITYP.EQ.0) GO TO 40
```

```
C..........GET INTERNAL # AND STARTING SECTOR FOR PERMANENT PIECE
   25 CALL DIREC (10225,IONOT,INTNO,NSECP,JSTAT)
      IF(JSTAT.EQ.0) GO TO 30
C..........ERROR CONDITIONS
      GO TO (27,29,29,27,27),JSTAT
   27 WRITE(LOUT,8030) JSTAT
 8030 FORMAT("DIRECTORY FILE ERROR,STATUS=",I4)
      CALL EXIT(0,0)
   29 ISTAT = 1
      RETURN
C..........MAKE DIRECTORY ENTRY AND GET GRADED STARTING SECTOR
C..........FOR PERMANENT PIECE
   30 IF(ITYPE.LT.0) GO TO 50
      IONOT(6) = INTNO
      CALL DIREC (4448,IONOT,ITMP,NSECG,JSTAT)
      IF(JSTAT.EQ.0) GOTO 60
C..........ERROR CONDITIONS
      GO TO (35,27,37,36,27),JSTAT
   35 WRITE(LOUT,1002)
 1002 FORMAT ("***GRADED PIECE FILE IS FULL")
      ISTAT = 2
      RETURN
C.....AN ENTRY FOR PIECE ALREADY EXISTS IN DIRECTORY
   36 ISTAT = 6
      RETURN
C..........FIXED PLATTER BAD
   37 ISTAT = 4
      WRITE(LOUT,8020)
 8020 FORMAT("FIXED DISK BAD")
      RETURN
C..........GET STARTING SECTOR FOR TEMPORARY PIECE
   44 CALL DIREC (10456,IONOT,INTNO,NSECP,JSTAT)
      IF(JSTAT.EQ.0) GO TO 50
C..........ERROR CONDITIONS
      GO TO (27,45,37,27,27),JSTAT
C.....PIECE NOT FOUND IN DIRECTORY
   45 ISTAT = 3
      RETURN
C..........SET STARTING SECTOR FOR GRADED DATA FOR TEMPORARY PIECE
   50 NSECG = 0
C..........READ PIECE DESCRIPTION INTO IBUF
   60 CALL UDSC(LABP)
   65 CALL EXEC (14,66,IBUF,128,NAMP,NSECP)
      JRLT = JLOF(JBP(1),IBUF(2))
      JSMP = JLOF(JBP(2),IBUF(2))
      JMIRU= JLOF(JBP(3),IBUF(1))
      JBOT = JLOF(JBP(4),IBUF(1))
      JSEC = JLOF(JBP(5),IBUF(1))
      IF(JSEC.LE.LMTSC) GO TO 70
      ISTAT = 7
      WRITE(LOUT,8010)
 8010 FORMAT("NUMBER SECTORS PIECE DATA EXCEEDS LIMIT")
      RETURN
   70 NWDS = (JSEC-1)*128
      CALL EXEC (14,66,IBUF(129),NWDS,NAMP,NSECP+1)
      NWDS = NWDS+128
      NWDS = NWDS-128
      IF(ISSW(1).LT.0) WRITE(LOUT,9002) (IBUF(K),K=1,NWDS+128)
 9002 FORMAT (1X,8A8)
C..........INITIALIZE HEADER INFORMATION
C..........COMPUTE # SECTORS NEEDED FOR GRADED HEADER
      IF(IFTT1.NE.0) GO TO 71
      KBUF(1) = 0
      KBUF(2) = 0
      NDX = 3
```

```
      NSIZ = (NSIZ-1)/12+1
      CALL UDSC(LABG)
C.....ENOUGH ROOM IN "GRADE"?
      ITMP = NGRP*NSECG*JSEC*NSIZ
      CALL EXEC (16,NAMG,JTMP)
      IF(ITMP.LE.JTMP) GO TO 69
      WRITE(LOUT,1003)
 1003 FORMAT("CHEK FILE IS FULL")
      ISTAT = 2
C.....PURGE DIRECTORY ENTRY IN "GRADE" IF CALLED BY PGRAD
      IF(ITYPE.LT.0) RETURN
      CALL DIREC(2HGB,IGNUT,ITMP,JTMP,JSTAT)
      RETURN
C.....GROUP AND GRADE ACCORDING TO COMMON X AND Y PERCENTAGES
   69 IBEG = 1
      IF(NSIZ.EQ.1) GO TO 73
      IF(ITYPE.GT.0) GO TO 71
      NSZ = NSIZ
      IF(IGEG.NE.1) GO TO 510
   73 IBEG = NSIZ+1
      I1 = 2
      I1K = 1
      GO TO 75
   71 JNC = INC(IBEG)
      NSZ = 1
      IF(IBEG.GT.NSIZ) GO TO 510
      I1K = IBEG
      IBEG = IBEG+1
      I1 = IBEG
      IF(IBEG.GT.NSIZ) GO TO75
      DO 72 I1=IBEG,NSIZ
      IF(INC(I1).NE.JNC) GO TO 75
   72 NSZ = NSZ+1
      IBEG = IBEG+NSZ
C.....LOOP TO GRADE
   75 IGOS = JSMP
      IFIT1 = 1
      IBEG = I1
      IXC = 0
      IYC = 0
      INXC = 0
      INYC = 0
      IRLT = IAND(ISIZ(I1K),1774000B)/256
C.....      IF RULE TABLE CHANGES FIND NEW GRF
      IF(IRLT.EQ.LRLT)GO TO 751
C
      LRLT=IRLT
      CALL UDSC(0)
      IXRLT(6)=IRLT
      CALL DIREC(3HGRB,IXRLT,INTNO,NSECR,ISTAT)
      CALL EXEC(14,66,IXRLT,2,NAMR,NSECR)
      GRF=XGRF(JLOF(IMGRF,IXRLT(2)))
      FACT=1000./GRF
751   CONTINUE
C.....SET UP X AND Y PERCENTAGES AND APPLY TO SAMPLE SIZE
      XINC = 1.0
      YINC = 1.0
      KEEP = 0
      IOK = 0
      IF(ITYPE.LE.2) GO TO 70
      KEEP = INC(I1K)
      INCX = IAND(INC(I1K),1774000B)/256
      INCY = IAND(INC(I1K),377B)
      IF(INCY.GE.128) INCY = IOR(INCY,1774000B)
```

```
      XINC = INCX/1000.0+1.0
      YINC = INCY/1000.0+1.0
   76 DO 500 I1=1,NSZ
      MLAST=0
      INCXL=0
      INCYL=0
      PREV = 0.0
      AREA = 0.0
C.....SIZE TO BE GRADED
      INDP = IAND(ISIZ(I1K+I1-1),3778)
      IXMIN=32000
      IXMAX = -32000
      IYMIN=32000
      IYMAX = -32000
      IF(ISSW(5).LT.0) WRITE(LOUT,9003) IRLT,INDS,INDP
      NCRV = 0
      IFIT = 0
C.....LOOP THROUGH DATA
      DO 490 I2=1,JSEC
      L1 = (I2-1)*128+3
      L2 = L1+123
      DO 480 I3=L1,L2,3
C.....CHECK POINT TYPE AND RULE
      JTYP = JLOC(JTP(0),IBUF(I3)) + 1
      JRUL = JLOC(JRP(10),IBUF(I3))
      IF(ISSW(5).LT.3)WRITE(LOUT,9003)JTYP,JRUL,L1,L2,I1,I2
 9003 FORMAT (1017)
      IF(I1.NE.1) GO TO 77
      IBUF(I3+1) = IBUF(I3+1)*XINC
      IBUF(I3+2) = IBUF(I3+2)*YINC
   77 GO TO (80,400,140,140,400,180,140,140,160,135,240,400),JTYP
C.....GRADE POINT
80    INCX2=INCXL
      INCY2=INCYL
C.....SAME RULE  ?
      IF(JRUL.EQ.MLAST) GOTO 81
C.....FIRST GRADE POINT OF OUTLINE
      NNN=1
      IF(I1.EQ.1) NNN=-1
      INCX2=(FLOAT(IDIFF(IRLT,INDP,INDS,JRUL,NNN)))*FACT
      INCY2=(FLOAT(IDIFF(IRLT,INDP,INDS,JRUL,2)))*FACT
      IF(NNN.LT.0) IDR = 1
81    MLAST=JRUL
82    INCXL=INCX2
      INCYL=INCY2
      IF(NCRV.NE.0) GO TO 85
      INCX1=INCX2
      INCY1=INCY2
      IGP1 = I3
      AFLG = 0
      NCRV = 1
      IX0 = IBUF(I3+1)
      IY0 = IBUF(I3+2)
      GO TO 490
C.....GRADE POINT AT END OF CURVE
85    NCRV=2
      IGP2 = I3
C.....PROCESS GRADE ON POINTS FROM IGP1 TO IGP2
   95 IF(NCRV.EQ.1) GO TO 400
      XT = INCX2
      YT = INCY2
      IEASY = 0
      JFIT = 0
      IF((IGP2.EQ.IGP1.OR.(IFIT.EQ.0.AND.IGP2-IGP1.EQ.3)) GO TO 100
      IEASY = 1
```

```
      DELX = FLOAT(IBUF(IGP2+1))-FLOAT(IX0)
      DELY = FLOAT(IBUF(IGP2+2)) - FLOAT(IY0)
      ANG = ATAN2(DELY,DELX)-.78539816
      S = SIN(ANG)
      C = COS(ANG)
      TX = INCX1
      TY = INCY1
      CALL XFORM (XTS,YTS,TX,TY,S,C)
      CALL XFORM (XTE,YTE,XT,YT,S,C)
      CALL XFORM (XE,YE,DELX,DELY,S,C)
  100 IF(KFLG.NE.0.OR.IEASY.NE.0) GO TO 101
      XT = INCX1
      YT = INCY1
  101 KFLG = 1
      DO 125 I4=IGP1,IGP2,3
      IF(MOD(I4,128).LE.2)I4=I4+2
      ITMP = JLOR(JSP(9),IBUF(I4))+1
      IF(IEASY.EQ.0.OR.ITMP.EQ.5) GO TO 110
      X1 = FLOAT(IBUF(I4+1))-FLOAT(IX0)
      Y1 = FLOAT(IBUF(I4+2))-FLOAT(IY0)
  105 CALL XFORM (XMM,YMM,X1,Y1,S,C)
      X1 = (XTS*(XE-XMM)+XTE*XMM)/XE
      Y1 = (YTS*(YE-YMM)+YTE*YMM)/YE
      S1 = -S
      CALL XFORM (XT,YT,X1,Y1,S1,C)
  110 IF(ITMP.NE.5.OR.JFTT.NE.0) GO TO 112
      XT = INCX1
      YT = INCY1
  112 XF = ((FLOAT(IBUF(I4+1))+XT+FLOAT(IXC))/1000.0)
      YF = ((FLOAT(IBUF(I4+2))+YT+FLOAT(IYC))/1000.0)
      IF(JFTT.EQ.0) GO TO 115
      AREA = AREA+(XS-XF)*(YS+YF)/2.0
      X12 = XF-XS
      X13 = YF-YS
      PREM = PREM+SQRT(X12*X12+X13*X13)
  115 XS = XF
      YS = YF
      JFTT = 1
      IF(IEASY.NE.0) GO TO 117
      XT = INCX2
      YT = INCY2
  117 JFTT = 1
      IF(I4.NE.IGP2) GO TO 120
      IX0 = IBUF(IGP2+1)
      IY0 = IBUF(IGP2+2)
  120 IBUF(I4+1) = XF*1000.0
      IBUF(I4+2) = YF*1000.0
      IF(ITMP.EQ.5) GO TO 125
C.....FIND MINS & MAXS
      CALL DIM(IXMIN,IXMAX,IYMIN,IYMAX,IBUF(I4+1),IBUF(I4+2))
      IF(JMIRO.EQ.1) IYMIN  = -IYMAX
  125 CONTINUE
      IF(ISS(9).LT.0) WRITE(LOUT,9003) IXMIN,IXMAX,IYMIN,IYMAX
      INCX1 = INCX2
      INCY1 = INCY2
      NCRV = 1
      IGP1 = IGP2+3
      IF(MOD(IGP1,128).LE.2) IGP1=IGP1+2
      GO TO 400
C.....LETTERING
  135 IF(JROL.NE.0) GO TO 140
      INCX3 = 0
      INCY3 = 0
      GO TO 145
```

```
C..........PUNCH HOLE, STRIPE LINE, ALTERNATE START POINT,
C..........INTERNAL POINT
  140 NNN=1
      IF(I1.EQ.1) NNN=-1
      INCX3=(FLOAT(IDIFF(IRLT,INDP,INDS,JRUL,NNN)))*FACT
      INCY3=(FLOAT(IDIFF(IRLT,INDP,INDS,JRUL,2)))*FACT
      IF(NNN.LT.0) IBR = 1
  145 IGP3 = I3
C..........GRADE THE POINT
  150 IBUF(IGP3+1) = (IBUF(IGP3+1)+INCX3+IXC)
      IBUF(IGP3+2) = (IBUF(IGP3+2)+INCY3+IYC)
C..........FIND MINS & MAXS
      CALL DIM(IXMIN,IXMAX,IYMIN,IYMAX,IBUF(IGP3+1),IBUF(IGP3+2))
      GO TO 444
C..........TRANSFER FUNCTION POINT
  160 IF(IRLT.EQ.JRLT.OR.I1.NE.1) GO TO 80
      IF(-JRUL.EQ.NLAST) GO TO 82
      ITX=IDIFF(IRLT,0,0,JRUL,1)*FACT
      ITY=IDIFF(IRLT,0,0,JRUL,2)*FACT
      NLAST=-JRUL
      GO TO 82
C..........GRAIN LINE
  180 IBUF(I3+1) = (IBUF(I3+1)+IXC)
      IBUF(I3+2) = (IBUF(I3+2)+IYC)
C..........FIND MINS & MAXS
      CALL DIM (IXMIN,IXMAX,IYMIN,IYMAX,IBUF(I3+1),IBUF(I3+2))
      IF(NANG.NE.3600) GO TO 400
      XGL(IGL) = IBUF(I3+1)
      YGL(IGL) = IBUF(I3+2)
  182 IGL = IGL+1
      IF(IGL.EQ.2) GO TO 400
      ANG = ATAN2((YGL(2)-YGL(1)),(XGL(2)-XGL(1)))
      NANG = IRND(ANG*572.95/79)
      IGL = 1
      GO TO 400
  240 ITMP = JLOF(JBP(9),IBUF(IGP1))
      IF(ITMP.NE.4) GO TO 410
      IBUF(IGP1+1) = FLOAT(IBUF(IGP1+1))+FLOAT(INCX1)+FLOAT(IXC)
      IBUF(IGP1+2) = FLOAT(IBUF(IGP1+2))+FLOAT(INCY1)+FLOAT(IYC)
C..........FIND MINS & MAXS
      CALL DIM (IXMIN,IXMAX,IYMIN,IYMAX,IBUF(IGP1+1),IBUF(IGP1+2))
      GO TO 410
  400 CONTINUE
C..........ERROR IF GET HERE WITHOUT REACHING END OF DATA, CONTINUE
      WRITE(LOUT,409)
  409 FORMAT("FINISHED GRADING WITHOUT REACHING END OF DATA")
      LSTAT = 99
      RETURN
C..........RAISE ABOUT CENTER
  410 IF(I1.EQ.1) GO TO 415
      INXC = INXC-IXC+(IXMAX+IXMIN)/2
      INYC = INYC-IYC+(IYMAX+IYMIN)/2
  415 IXC = (IXMAX+IXMIN)/2
      IYC = (IYMAX+IYMIN)/2
      IF(ISSW(4).LT.3) WRITE(LOUT,9003) IXC,IYC,(IBUF(K),K=3,72)
      DO 420 I2=1,JSEC
      L1 = (I2-1)*1200+3
      L2 = L1+123
      DO 420 I3=L1,L2,3
      IBUF(I3+1) = IBUF(I3+1)-IXC
      IBUF(I3+2) = IBUF(I3+2)-IYC
      IF(JLOF(JBP(9),IBUF(I3)).EQ.10) GO TO 425
  420 CONTINUE
  425 IXMIN = IXMIN-IXC
```

```
      IYMIN = IYMIN-IYC
      AREA = ABS(AREA)
C.....STORE HEADER DATA INTO KBUF
      IF(JDIR0.EQ.0) GO TO 426
      AREA = 2.0*AREA
      PREM = 2.0*PREM
  426 KBUF(KDX) = 0
      CALL JSTF(INDP,JBG(3),KBUF(KDX))
      CALL JSTF(IRLT,JBG(4),KBUF(KDX))
      KBUF(KDX+1) = IXMIN
      KBUF(KDX+2) = IYMIN
      KBUF(KDX+3) = INXC
      KBUF(KDX+4) = INYC
      KBUF(KDX+5) = IAREA(1)
      KBUF(KDX+6) = IAREA(2)
      KBUF(KDX+7) = IPREM(1)
      KBUF(KDX+8) = IPREM(2)
      KBUF(KDX+9) = KEEP
      IF(ITYPE.LE.0) KBUF(KDX+9) = 0
      KDX = KDX+10
      IF(MOD(KDX,128).NE.123) GO TO 430
      KDX = KDX+5
      KBUF(KDX) = 0
      KBUF(KDX+1) = 0
      KDX = KDX+2
C.....STORE GRADED PIECE
  430 CALL UDSC(LABG)
      NWDS = JSEC*128
      CALL EXEC (15,05,IBUF,NWDS,NAMG,NSECG+NGH)
      NGH = NGH+JSEC
      IWDS = IWDP
      IF (NGH.LT.513) GOTO 500
      WRITE (LPUT,512) (IDSET(I3),I3=1,5)
512   FORMAT ("***TOO MANY SIZES PIECE  ",5A2,/)
  500 CONTINUE
      IF(ITYPE.NE.0.AND.IBEG.LE.NSIZ) GO TO 60
C.....STORE HEADER INFORMATION
  510 CALL JSTF(NGH,JBG(2),KBUF(1))
      CALL JSTF(JSEC,JBG(1),KBUF(1))
      KBUF(2) = NAMG
      KDX = KDX-1
      IF(MOD(KDX,128).EQ.2) KDX = KDX-8
      CALL UDSC(LABG)
      CALL EXEC (15,06,KBUF,KDX,NAMG,NSECG)
      IF(IBR.NE.0) ISTAT = 98
      RETURN
      END
      SUBROUTINE DIM(IXN,IXX,IYN,IYX,IX,IY)
C.....OCT. 11, 1972
C.....MARGARET ROWAN
C.....PROGRAMS: CHECK, PGRAB
C.....VERSION 2.0
C.....ROUTINE TO COMPUTE THE MINIMUM AND
C.....MAXIMUN VALUES OF IX AND IY
      IXN = MIN0(IXN,IX)
      IXX = MAX0(IXX,IX)
      IYN = MIN0(IYN,IY)
      IYX = MAX0(IYX,IY)
      RETURN
      END
      ENDS
```

```
FTN4
      SUBROUTINE GRAIN (MSEC,LANG)
C.....DEC. 26,1972
C.....MARGARET ROMAN
C.....PROGRAM: MARK, DRAWM
C.....VER 2.0
C.....ROUTINE TO DETERMINE GRAIN ANGLE FROM STRIPE LINE.
C.....ANGLE RETURNED IN TENTHS OF A DEGREE
      DIMENSION XGL(2),YGL(2),IBUF(128)
      DIMENSION NAMGD(3)
      DATA NAMGD/2HGR,2HAD,2HE /
      LANG = 0
      IGL = 1
      JTYP = JFLD(15,4)
      CALL UDSC(3)
      ISEC = MSEC
C.....READ PIECE DATA
   10 CALL EXEC (14,06,IBUF,128,NAMGD,ISEC)
      DO 50 I=3,126,3
      ITYP = JLDF(JTYP,IBUF(I))
C.....END OF DATA?
      IF(ITYP.EQ.16) RETURN
      IF(JTYP.GE.3) GO TO 50
C.....STRIPE LINE
      XGL(IGL) = IBUF(I+1)/100.0
      YGL(IGL) = IBUF(I+2)/100.0
      IGL = IGL+1
      IF(IGL.GT.2) GO TO 100
   50 CONTINUE
      ISEC = ISEC+1
      GO TO 10
C.....SET UP ANGLE
  100 ANG = ATAN2((YGL(2)-YGL(1)),(XGL(2)-XGL(1)))
      LANG = IRND(ANG*572.95779)
      RETURN
      END
      ENDS


ASMB,R
      NAM GRID,7
      ENT GRID
      ENT SGRID
      EXT .ENTR
      EXT FLOAT
      EXT JRND
*
*     SUBROUTINE TO GRID A REAL (X,Y) TO A REAL GRID OF
*     INTERVAL Z.
*
FPONE DEC 1.0
Z     DEF *,I
SGRID JMP *          ENTRY TO SET GRID
      JSB .ENTR
      DEF Z
      DLD FPONE
      FDV Z,I
      DST S1         USE FMP IN GRID - NOT FDV
      DLD Z,I
      DST S2         SAVE FOR GRID ALSO
      JMP SGRID,I
S1    BSS 4,0
S2    BSS 4,0
*****
*****
```

```
X       OCT *,I
Y       OCT *,I
GRID    JMP *           ENTRY TO GRID (X,Y)
        JSB .ENTR
        DEF X
        DLD X,I
        JSB WORK
        DST X,I
        DLD Y,I
        JSB WORK
        DST Y,I
        JMP GRID,I
*****
*****
WORK    JMP *
        FMP S1
        JSB JRND
        JSB FLOAT
        FMP S2
        JMP WORK,I
        END

FTN4
        SUBROUTINE READ(IBUF,IDF)
C.....INITIALIZE DISPLAY BUFFER
        DIMENSION IBUF(1),IBLOC(13)
        DATA IBLOC/4...21B,73...4B,0,7776...B,0,1,40...5B,
       1  15...1B,45...14B,7776...B,4...00B,0,17
        DO 2... I=1,13
    2.. IBUF(IDF+I-1) = IBLOC(1)
        IDF = IDF + 13
        RETURN
        END
        ENDS

FTN4
        SUBROUTINE HEADI(IX,IY,IORG,IMGNO)
C.....DEC. 11, 1972
C.....MARGARET ROMAN
C.....PROGRAM: MARK
C.....VERSION 2.0
C.....ROUTINE TO SET UP HEADER FOR PIECE IMAGE
C
        COMMON MBUF,LABF(3),LABR(3),NDSTY,INSTY(31),JRS(15),NAMST(3),
       1 MKUP(5),NSCNO,JBM(2..),NAMMK(3),JRG(4),NAMGD(3),JBP(1..),
       2 IDPR(5),IDGP,NPEC,NSHO,FACTR,NSIZ,ISIZ(42),WIDTH,EFF,SX,SY,KSTL,
       3 YARDS,MAP,KIMT,KFLG,KINC,KSIZ,KXMN,KYMN,KANG,KXMX,KYMX,KOPT,
       4 NAD,KADP,KDF,CSMX,MKNO,LSMN,NRGE,IMGP,KREL,ITYPE,
       5 NOING,IOFSZ,IMGY,IMGE,IMGTC,IMKO1,IMKO2,IMENU,IDXP,IDYP,
       6 IXC,IYC,IXL,IXP,IYU,IYL,CS,MDELX,MDELY,JFLG,MLW,MUP,ISLX,ISLY,
       7 TOTAL,MYL,XPC,YPC,INGLP,IMESG,IEDG,ISOFF,SOFF,KSLH,
       8 IMGPS,MAXDF,KSLV,KTC,IOPTN,IAREA,ISIZE,KCSR,KYES,KPSIZ,KPLL,
       9 IT1,IT2,IT3,IT4,IT5,IT6,IT7,IT8,IT9,IT10,
       A IOFSET,KPLH,IFLP,ICOST,IMGPK,
       B T1,T2,T3,T4,T5,T6,T7,T8,T9,T10,
       C JSTY,JPN,JNG,JSHO,JRT,
       D ILPM,IBU,IBD,IOV,IBR,IBL,ITCW,ITCCW,
       E IRCW,IRCCW,ISL,IRE,IFL,IPK,ICOM,
       F IDUM(1)
        IN = 5
C.....CHECK FOR ALPHA OPTION
        IF(JLOF(JBM(15),IOPTN).NE.0)NN=NN+3
C.....CHECK FOR SIZE OF PIECE
```

```
      IF(JLOF(JAM(15),IDUM(KFLG+NPEC+IMGNO)).NE.0) NN=NN+4
      KAD1 = KADP+IMGNO
      IDFPT = IABS(IDUM(KAD1))
      IF(IDFPT+NN.GT.IDFSZ) CALL ERR(4,0)
C.....SET UP REGISTERS.
      IDUM(IDFPT) = 402248
      IDUM(IDFPT+1) = 0
      IDUM(IDFPT+2) = 1
C.....SET UP POINT MOVE FROM IMAGE ORIGIN TO PIECE ORIGIN
      IDUM(IDFPT+3) = 100648
      IF(NN.NE.4)IDUM(IDFPT+3)=400108
      CALL MOVI (1,IX,IY,IORG,IMGNO)
      KOPT = IDFPT+4
      IF(NN.GT.0) RETURN
      N = JLOF(JAM(9),IDUM(KSIZ+IMGNO))
C.....ADD ALPHA SIZE INDICATOR
      IDUM(IDFPT+6)=101178
      NNN=(64+K)*256+100
      IF(NN.NE.0) GO TO 100
      IDUM(IDFPT+7)=NNN
      IDUM(IDFPT+8)=412000B
      RETURN
C.....ADD SIZE TO PIECE. FOUR CHAR.MAX.(FIRST FOUR IN SIZE)
  100 IMGS = (K-1)*3+IDFSZ+1
      IDUM(IDFPT+7) = IDUM(IMGS)
      MMM=IDUM(IMGS+1)
      IDUM(IDFPT+8)=MMM
      IF(MMM.EQ.240000) IDUM(IDFPT+8)=NNN
      IDUM(IDFPT+9)=NNN
      IF(NNN.EQ.240000) IDUM(IDFPT+9)=MMM
      IDUM(IDFPT+10)=401105
      IDUM(IDFPT+11)=401400
      IDUM(IDFPT+12)=412000B
      RETURN
      END
      ENDS

FTN4
      SUBROUTINE HALFS(IDXP,IDXL,ITAB,IRLT)
C.....JAN. 5, 1973
C.....MARGARET ROMAN
C.....PROGRAM DARK
C.....VERSION 2.0
C.....ROUTINE TO DETERMINE MASTERS AND SLAVES FOR HALF PIECES
C.....VIA THE TABLES STORED IN "HALFO", "HALFP".
      DIMENSION INAM(6),NAMM(3),IBUF(256),IGP(42)
      COMMON MOUT,LABF(3),LABR(3),NUSTY,INSTY(31),JBS(15),NAMST(3),
     1 NAMW(3),NSQNU,JAM(20),NAMMK(3),JMG(4),NAMGD(3),JBP(10),
     2 IDUM(3),IDOF,NPEC,NGNO,FACTR,NSIZ,ISIZ(42),WIDTH,EFF,SX,SY,KSIL,
     3 KADS,KAP,KIN1,KFLG,KINC,KSIZ,KXMN,KYMN,KANG,KXMX,KYMX,KOPT,
     4 KAD,PAMP,KOF,CSMX,MKWD,CSMN,NMGE,IMGP,KREL,ITYPE,
     5 NOIMG,IDFSZ,IMGF,IMGE,IMGTC,IMK01,IMK02,IMENU,IDXP,IDYP,
     6 IXC,IYC,IXL,IXH,IYU,IYL,CS,MDELX,MDELY,JFLG,MLW,MUP,ISLX,ISLY,
     7 TOTAL,MYL,XPC,YPC,IMGLP,IMESG,IERG,ISOFF,SOFF,KSLH,
     8 IMGFS,MAXOF,KSLV,ATC,IOPTN,IAREA,ISIZE,KCSR,KYES,KPSIZ,KPLL,
     9 IT1,IT2,IT3,IT4,IT5,IT6,IT7,IT8,IT9,IT10,
     A IDFSZ,KPLM,TPLM,ICOST,IMGPK,
     B T1,T2,T3,T4,T5,T6,T7,T8,T9,T10,
     C JST,JPS,JNG,JSNO,JRT,
     D ILP,IPO,IMO,IOV,IRR,IRL,ITCW,ITCCW,
     E IXCW,IYCC,ISL,IRE,IPL,IPK,ICOM,
     F IDUM(1)
      DATA INAM/2HHA,2HLF,2HP /
      JPK = 1
```

```
      IDX = IDXP
      JBIG = JFLD(2,3)
C.....IF ONLY ONE SIZE, RETURN
      IF(IDXP.EQ.IDXL) RETURN
C.....READ IN TABLE
      INAM(5) = ITAB
      CALL DIREC(3HDSK,INAM,JTMP,NSCT,ISTAT)
      IF(ISTAT.EQ.0) GO TO 40
C.....ERROR CONDITIONS
      GO TO (10,20,30,10,10),ISTAT
C.....ERROR OF IMPOSSIBLE RETURN FROM DIREC
   10 CALL EXP(3,4)
C.....ERROR OF TABLE NOT AVAILABLE
   20 WRITE(IOUT,1000) ITAB
 1000 FORMAT("***HALF PIECE TABLE,"I3," NOT AVAILABLE")
      CALL EXIT
C.....ERROR OF "HALFD" NOT SET UP
   30 WRITE(IOUT,1001)
 1001 FORMAT ("*** AHALFDA NOT AVAILABLE")
      CALL EXIT
   40 CALL EXEC(14,55,IBUF,256,NAMH,NSCT)
      LSEC =JLOF(JBIG,IBUF)
      LGRP =JLOF(JBIT(9),IBUF(2))
      IRLT =JLOF(JBIT(10),IBUF(2))
      IF(IRLT.EQ.IRLT) GO TO 50
C.....ERROR RULE TABLES DON'T MATCH
      WRITE(IOUT,1002)ITAB , IRLT
 1002 FORMAT("***HALF PIECE TABLE,",I3,"SET UP FOR RULE TABLE,",
     1    I3)
      CALL EXIT
C.....FIND SIZE AND GROUP FOR ALL PIECES
   50 ISZ = JLOF(JBIT(10),IDUM(KSIZ+IDX))
      JGP = 1
      IPTR = 3
      ILR = 2
   55 NSZ = JLOF(JBIT(9),IBUF(IPTR))
      DO 60 I=1,NSZ
      JSZ = JLOF(JBIT(10+ILR),IBUF(IPTR))
      IF(JSZ.EQ.ISZ) GO TO 100
      ILR = 3-ILR
      IF(ILR.EQ.1) IPTR = IPTR+1
      IF(IPTR.LE.128) GO TO 60
      IF(LSEC.EQ.2) GO TO 56
C.....ERROR IN TABLE
      WRITE(IOUT,1003) ITAB
 1003 FORMAT("***ERROR IN HALF PIECE TABLE")
      CALL EXIT
   60 CONTINUE
      JGP = JGP+1
      IF(JGP.GT.LGRP) GO TO 65
      IF(ILR.EQ.2) IPTR=IPTR+1
      IF(ILR.EQ.1) ILR=2
      GO TO 55
C.....SIZE NOT IN TABLE SO SET IGP TO 0
   65 IGP(JDX) = 0
   70 IDX = IDX+1
      IF(IDX.GT.IDXL) GO TO 200
      JDX = JDX+1
      GO TO 50
C.....SET IGP TO GROUP#REL. SIZE
  100 IGP(JDX) = JGP*256 + I
      GO TO 70
C.....SORT AND SET AS MASTER/SLAVE PAIRS
```

```
C.....SORT BY GROUP, BY SIZE AND STORE INDEX IN IBUF
  200 JJDX = JDX-1
      DO 205 IDX = 1,JDX
  205 IBUF(JDX) = IDX
      DO 220 IDX=1,JJDX
      IIDX = IDX+1
      KDX = IDX
      IGP1 = IGP(IDX)
      DO 210 I=IIDX,JDX
      IF(IGP(I).GE.IGP1) GO TO 210
      KDX = I
      IGP1 = IGP(I)
  210 CONTINUE
      IF(IDX.EQ.KDX) GO TO 220
      ITMP = IBUF(KDX)
      IBUF(KDX) = IBUF(IDX)
      IBUF(IDX) = ITMP
      IGP(KDX) = IGP(IDX)
      IGP(IDX) = IGP1
  220 CONTINUE
C.....PAIR OFF DUPLICATES
      IDX = 1
  222 IF(IDX+1.GE.JDX)GO TO 229
      IF(IGP(IDX).NE.IGP(IDX+1)) GO TO 227
C.....'IDX' IS SLAVE
      ITMP = IBUF(IDX)+IDXF-1+KFLG
C.....'IDX+1' IS MASTER
      JTMP = IBUF(IDX+1)+IDXF-1
C.....SET FLAGS
      CALL JSTF(1,JBM(2),IDUM(ITMP))
      CALL JSTF(JTMP,JBM(7),IDUM(ITMP+NPEC))
      IGP(IDX) = 0
      IGP(IDX+1) = 0
      IDX = IDX+2
      GO TO 222
  227 IDX = IDX+1
      GO TO 222
C.....PAIR OFF WITHIN COMMON GROUPS
  229 IDX = 1
      IDXS = 0
  230 IF(IGP(IDX).EQ.0) GO TO 240
      IF(IDXS.NE.0) GO TO 235
      IDXS = IDX
      GO TO 240
  235 IF(JLDF(JBM(9),IGP(IDXS)).NE.JLDF(JBM(9),IGP(IDX))) GO TO 245
C.....'IDXS' IS SLAVE
      ITMP = IBUF(IDXS)+IDXF-1+KFLG
C.....'IDX' IS MASTER
      JTMP = IBUF(IDX)+IDXF-1
C.....SET FLAGS
      CALL JSTF(1,JBM(2),IDUM(ITMP))
      CALL JSTF(JTMP,JBM(7),IDUM(ITMP+NPEC))
      IGP(IDXS) = 0
      IGP(IDX) = 0
      IDXS = 0
  240 IDX = IDX+1
      IF(IDX.GT.JDX) GO TO 250
      GO TO 230
  245 IDXS = IDX
      GO TO 240
C.....PAIR OFF REMAINDERS BETWEEN GROUPS
  250 IDX = 1
      IDXS = 0
  255 IF(IGP(IDX).EQ.0) GO TO 260
```

```
      IF(IDXS.NE.0) GO TO 265
      IDXS = IDX
  260 IDX = IDX+1
      IF(IDX.GT.JDX) RETURN
      GO TO 255
C..........FIRST FOUND IS SLAVE
  265 ITMP = INDF(IDXS) + IDXF-1+KFLG
C..........SECOND FOUND IS MASTER
      JTMP = INDF(IDX) + IDXF-1
C..........SET FLAGS
      CALL JSTF(1,JM(24),IDUM(ITMP))
      CALL JSTF(JTMP,JM(7),IDUM(ITMP+NPEC))
      IDF(IDX) = 0
      IDF(IDXS) = 0
      IDXS = 0
      GO TO 260
      END
      ENDF


FTN4
      FUNCTION IDIFF(IRLT,INDP1,INDS1,M,MMM)
C.......... FUNCTION TO CALCULATE THE INCREMENTAL MOVEMENT
C           FOR RULE M BETWEEN SIZES INDP AND INDS, USING
C           RULE TABLE IRLT  THE FLAG IXY INDICATES IF THE
C             MOVEMENT IS FOR X  (IXY=1) OR FOR Y (IXY=2)
      DIMENSION IRLT(12,1),INDF2(128),NAMR(3),NAMB(6)
      DATA NRLT/0/,NRUL/0/,NSIZ/0/,LOUT/6/
      DATA NAMR/2HRU,2HLE,2HS /,ICURDS/-1/,ICURDP/-1/
      IF (ISS(2).LT.0) WRITE (6,1111) IRLT,INDP1,INDS1,M,MMM
 1111 FORMAT(517)
      IXY=IABS(MMM)
      IF(IRLT.EQ.NRLT)GO TO 10
C.......... NEW RULE TABLE
      NRLT=IRLT
      NAMB(6)=NRLT
      CALL CDREC(3HFOR,NAMB,INT,NSCTR,ISTAT)
      IF(ISTAT.NE.0)GO TO 71
      ICURDS=-1
      ICURDP=-1
      CALL EXEC(14,60,INDF,2,NAMR,NSCTR)
      NRUL=JFLD(15,0)
      NSIZ=JFLD(0,15)
      NRUL=JLDF(IRUL,INDF(1))
      NSIZ=JLDF(ISIZ,INDF(2))+1
C.........  CALCULATE SECTOR # OF DATA
C.........     SKIP OVER SIZE RELATION TABLE
      INDSEC=(NSIZ-1)/42+1+NSCTR
   10 IF(M.GE.1.AND.M.LE.NRUL)GO TO 20
      IF(MMM.LT.0) WRITE(LOUT,1000) M
 1000 FORMAT("RULE ",I3," BAD")
      IDIFF=0
      RETURN
   20 IF(INDP1.LE.NSIZ.AND.INDS1.LE.NSIZ) GO TO 30
      WRITE(LOUT,1001)IRLT
 1001 FORMAT(" SIZE NOT IN RULE TABLE",I6)
      IDIFF=0
      RETURN
C.......... CALCULATE SECTOR CONTAINING SIZE AND RULE
C                 FOR INDP AND INDS
   30 MMM=IABS(MMM)
      INDP=INDP1*NRUL*2+2*(M-1)+1
      INDS=INDS1*NRUL*2+2*(M-1)+1
      INDPSC=(INDP-1)/128
      INDSSC=(INDS-1)/128
```

```
      INDXDP=INDP-IRDPSC*126+2
      INDXDS=INDS-IRDSSC*126+2
C.....   CHECK TO SEE IF SECTOR CONTAINING INDP I
C          IS IN BUFFER 1
      IF(ICURDP.EQ.IRDPSC)GO TO 40
      IADPSC=INDPSC+INDSEC
      CALL EXEC(14,RD,IRDF,128,NAMR,IADPSC)
      ICURDP=IRDPSC
C.....CHECK TO SEE IF SECTOR CONTAINING INDS IS IN BUFFER2
40    IF(ICURDS.EQ.IRDSSC)GO TO 50
      IADSSC=INDSSC+INDSEC
      CALL EXEC(14,RD,IRDF2,128,NAMR,IADSSC)
      ICURDS=IRDSSC
50    IIXY=IXY-1
      IDIFF=IRDF(INDXDP+IIXY)-IRDF2(INDXDS+IIXY)
      IF(IDIFF(2).LT.0)WRITE(LOUT,1002)IDIFF,IXY
1002  FORMAT(2I8)
      RETURN
71    ISTAT=2
72    WRITE(LOUT,1003)ISTAT
1003  FORMAT("RDLFFILE ERROR STATUS = ",I4)
      END
      ENDS


FTN4
      FUNCTION INDEX(J,I)
C.....FUNCTION TO GENERATE CIRCULAR INDICES
      JJ = MOD(J,I)
      IF(JJ.GT.0) JJ=I
      INDEX = JJ
      RETURN
      END
      ENDS


FTN4
      FUNCTION INOUT(IX0,IY0,IX,IY,N)
      DIMENSION IX(1),IY(1)
      ISTAT = -1
      M=N-1
      DO 300 I=1,M
      JDY = IY0-IY(I)
      JDYI = IY(I+1)-IY0
      IF(FLOAT(JDY)*FLOAT(JDYI).GT.0.0) GO TO 400
      IF(JDYI.EQ.0) GO TO 500
  350 CONTINUE
      GO TO 600
  400 CONTINUE
      JDXI = IX(I+1)-IX(I)
      JDYI = IY(I+1)-IY(I)
      JDX = IX0-IX(I)
      JDY = IY0-IY(I)
      IF(FLOAT(JDY)*FLOAT(JDXI).EQ.FLOAT(JDYI)*FLOAT(JDX))GO TO 700
      IF(FLOAT(JDX).GE.FLOAT(JDY)*FLOAT(JDXI)/FLOAT(JDYI))GO TO 300
  450 ISTAT = -ISTAT
      GO TO 300
  500 CONTINUE
      IF(IX0-IX(I+1))600,700,300
  500 JDY = IY(I+1)-IY(I)
      JDYI = IY(  MOD(I+1,M)+1 )-IY(I+1)
      IF(JDY.GE.0.AND.JDYI.GT.0) GO TO 450
      IF(JDY.LT.0.AND.JDYI.LE.0) GO TO 450
      GO TO 300
  700 ISTAT = 0
```

```
      IF(ISTAT.GE.0) INOUT=1
      RETURN
      END
      ENDS


FTN4
      SUBROUTINE INSPT(N,IA,IB)
      DIMENSION IA(1),IB(1)
      IF(N.LT.0) GO TO 20
      DO 10 I=1,N
   10 IA(I)=IB(I)
      RETURN
   20 M=IABS(N)
      DO 30 I=1,M
   30 IA(I)=2H
      RETURN
      END
      ENDS


FTN4
      SUBROUTINE INTST (IFLAG,JX,JY,XNT,YNT)
      DIMENSION IX(4),IY(4),SL(2),JX(4),JY(4)
      DIMENSION IDUM(256)
      DATA DEL/.001/
C.....THIS ROUTINE CHECKS FOR INTERSECTION BETWEEN TWO LINES
C.....GIVEN THE POINTS IX(1),IY(1) AND IX(2),IY(2) FOR LINE 1
C.....AND IX(3),IY(3) AND IX(4),IY(4) FOR LINE 2.  IFLAG
C.....IS SET NON-ZERO WHEN AN INTERSECTION IS FOUND.  THE
C.....INTERSECTION POINT IS RETURNED IN IX(1),IY(1).
      IFLAG = 0
      N=1
      DO 5 I=1,4
      IX(I) = JX(I)-JX
    5 IY(I) = JY(I)-JY
      DO 10 I=1,3,2
      J=I+1
      SL(N)=FLOAT(IY(J)-IY(I))/FLOAT(IX(J)-IX(I))
      IF(IX(I).EQ.IX(J))SL(N)=9999999.9
   10 N=N+1
      IF(IX    .EQ.IX(2).AND.IX(3).EQ.IX(4)) GO TO 40
   15 IF(SL    .LT.9999999.9) GO TO 20
      XNT = IX
      B2=-SL(2)*IX(4)+IY(4)
      YNT = SL(2)*XNT+B2
      GO TO 30
   20 IF(SL(2).LT.9999999.9) GO TO 25
      XNT = IX(4)
      B1 = -SL    *IX    +IY
      YNT = SL    *XNT+B1
      GO TO 30
   25 B1 = -SL    *IX    +IY
      B2=-SL(2)*IX(4)+IY(4)
      IF(SL    .EQ.SL(2).AND.B1.EQ.B2) GO TO 40
      IF(SL    .EQ.SL(2))RETURN
   27 XNT = (B1-B2)/(SL(2)-SL    )
      YNT = SL    *XNT+B1
      IF(IABS(IX-IX(2)).LE.2) YNT = SL(2)*XNT+B2
   30 D12 = IABS(IX    -IX(2))
      D34 = IABS(IX(3)-IX(4))
      D1 = ABS(IX    -XNT)
      D2 = ABS(IX(2)-XNT)
      D3 = ABS(IX(3)-XNT)
      D4 = ABS(IX(4)-XNT)
```

```
      IF(ABS(D1+D2-D12).GT.DEL.OR.ABS(D3+D4-D34).GT.DEL) RETURN
      D12 = IABS(IY   -IY(2))
      D34 = IABS(IY(3)-IY(4))
      D1 = ABS(IY   -YNT)
      D2 = ABS(IY(2)-YNT)
      D3 = ABS(IY(3)-YNT)
      D4 = ABS(IY(4)-YNT)
      IF(ABS(D1+D2-D12).GT.DEL.OR.ABS(D3+D4-D34).GT.DEL) RETURN
      XNT = XNT+JX
      YNT = YNT+JY
         IFLAG = 1
         RETURN
C.....CHECK FOR TANGENT SEGMENTS
      XL = MIN0(IX,IX(2),IX(3),IX(4))
         IX2=MAX0(IX,IX(2),IX(3),IX(4))
         I012 = IABS(IX   -IX(2))
         I034 = IABS(IX(3)-IX(4))
         I01 = IABS(I02-IX2)
         IF(I012+I034.LT.I01)RETURN
         IY1=MIN0(IY,IY(2),IY(3),IY(4))
         IY2=MAX0(IY,IY(2),IY(3),IY(4))
         I012 = IABS(IY   -IY(2))
         I034 = IABS(IY(3)-IY(4))
         I01 = IABS(IY1-IY2)
         IF(I012+I034.LT.I01)RETURN
         IFLAG = -1
         RETURN
         END
         ENDS

FTN4
      FUNCTION JDST (IX,IY,JXX,JYY,NXYJ,N)
C.....FUNCTION TO FIND THE MINIMUM DISTANCE BETWEEN
C.....A POINT, (IX,IY), AND A POLY-STRING OF POINTS,
C.....(JXX,JYY).  USED FOR OVERLAP.
C.....N MEANS IN X OR Y DIRECTION (1 OR 2, RESPECTIVELY)
      DIMENSION JXX(1),JYY(1)
      D = 9999.0
      IF(NXYJ.LE.1) GO TO 245
      IBEG = 2
      IEND = NXYJ
  100 DO 240 I=IBEG,IEND
  105 J = I-1
      GO TO (110,120),N
  110 ID1 = IABS(IY-JYY(J))
      ID2 = IY-JYY(I)
      ID4 = JYY(J)-JYY(I)
      ID3 = JYX(J)-JXX(I)
      GO TO 130
  120 ID1 = IABS(IX-JXX(J))
      ID2 = IX-JXX(I)
      ID4 = JXX(J)-JXX(I)
      ID3 = JYY(J)-JYY(I)
  130 ID2 = IABS(ID2)+ID1
      ID3 = IABS(ID4)
      IF(ID2.NE.ID3) GO TO 200
      IF(ID3+ID2.EQ.0) GO TO 200
      SL = FLOAT(ID2)/FLOAT(ID4)
      IF(ID2.EQ.0) SL = 0.0
      DST = FLOAT(ID1)*SL
      GO TO (140,150),N
  140 DINT = FLOAT(JXX(I)-IX)+DST
      GO TO 160
  150 DINT = FLOAT(JYY(I)-IY)+DST
```

```
   100 IF(ABS(DINT).GE.0) GO TO 200
       0 = ABS(DINT)
       JDST = DINT+SIGN(1.0,DINT)
   200 CONTINUE
       IF(0.EQ.9999.0) JDST = 9999
       RETURN
   205 JDST = 9999
       RETURN
       END
       ENDS


ASMB
       NAM LOCK0,7
       ENT LOCK0
       EXT .ENTR,EXEC
*
OPER   BSS 1               OPERATION FLAG ADDRESS
LOCK0  BSS 1
       JSB .ENTR
       DEF OPER
       LDA OPFLG           WAIT FOR OPERATOR TO STOP
       SZA                 GAMING WITH THE SYSTEM
       JMP *-2
       LDA OPER,I          LOAD OPERATION FLAG, 0=CLEAR, 1=SET
       LDB $OPER           SET B=>OPER ADDRESS
       JSB EXEC            EXEC BASE PAGE STORE
       DEF *+2             RETURN ADDRESS
       DEF RC19            REQUEST CODE ADDRESS
       JMP LOCK0,I         RETURN
*
$OPER  OCT 2503            ADDRESS OF $OPER ENTRY IN DISCM
RC19   DEC -19             EXEC BASE PAGE STORE REQUEST CODE
OPFLG  EQU 2019            5TH CONSOL BUSY FLAG
*
       END
ENDS


FTN4
       PROGRAM MARK,3
C..........MAIN PROGRAM OF MARK
C.................SETS UP COMMON AND CALLS SEGMENT, MRKA
C
C
       DIMENSION DX(3)
C..... DUMMY ARRY  ARRAY TOLOAD  BOR
       COMMON MOUT,LASF(3),LASR(3),NOSTY,INSTY(31),JBS(15),NAMST(3),
      1 IKUP(5),NSCRO,JBU(20),NAMMK(3),JRG(4),NAMGD(3),JBP(10),
      2 IUMK(5),IOCP,NPEC,NSHO,FACTR,NSIZ,ISIZ(42),WIDTH,EFF,SX,SY,KSIL,
      3 YARDS,NAP,KINT,KFLG,KINC,KSIZ,KXMN,KYMN,KANG,KXMX,KYMX,KDPT,
      4 KAD,KADP,KDF,CSMX,NKWD,CSMN,NRGE,IMGP,KREL,ITYPE,
      5 BDIMG,IOFSZ,IMGY,IMGE,IMGTC,IMKO1,IMKO2,IMENU,IDXP,IDYP,
      6 IXC,IYC,IXL,IXR,IYU,IYL,CS,MDELX,MDELY,JFLG,MLW,MUP,ISLX,ISLY,
      7 TOTAL,MYL,XPC,YPC,IMGLP,IMESG,IEOG,ISOFF,SOFF,KSLH,
      8 IMGPS,MAXOF,KSLV,KTC,IOPTN,IAREA,ISIZE,KCSR,KYES,KPSIZ,KPLL,
      9 IMGIN,ICPEC,IPKC,ISU(7),
      A IOFSET,KPLM,IFLP,ICUST,IMGPX,
      B TLIL,ISNOO,ISHOL,I3,I4,I5,I6,I7,I8,I9,ILOOP,INSCP,
      C JSTY,JPG,JNG,JSMO,JKT,
      D ILPS,IMU,IMU,IOV,IBR,IBL,ITCW,ITCCW,
      E IRCW,IRCCW,ISL,IRE,IFL,IPK,ICOM,
      F IDUM(8390)
       MAXOF = 8390
       ICOM = 136
       DX(1) = 2HMP
```

```
      MK(2) = 2HMA
      MK(3) = 2H
      CALL EXEC (8,MK)
      CALL MRKA
      CALL MRKB
      CALL MRKC
      CALL MRKD
      CALL MRKE
      CALL COMP
C.....CALL MRAG
      CALL MREF
      CALL DDSC
      CALL JFLD
      CALL SWPT
      CALL START
      CALL ISSW
      CALL IOR
      CALL IAND
      CALL FLOAT
      CALL IRND
      CALL IABS
      CALL IFIX
      CALL ERRM
      CALL ERR
C.....CALL MREM
C.....CALL MRKI
      CALL MRKJ
      END
      $EOJ


FTN4
      SUBROUTINE MDTSA (I)
      INTEGER MARKD(3)
      DATA MARKD /2HMA,2HRK,1HD/
      CALL DDSC (1)
      IF (I.LT.1) GO TO 200
   50 CALL DIREC (54/2B,M,IPLAT,M,ISTAT)
      IF (ISTAT.NE.0) GO TO 100
      IF (IPLAT.NE.I-1) GO TO 100
      RETURN
  100 WRITE (1,1000) I
      PAUSE
      CALL EXEC (23,0,1,1)
      GO TO 50
  200 CALL EXEC (1B,MARKD,M)
      IF (4.GT.0) RETURN
      WRITE (1,2000)
      PAUSE
      CALL EXEC (23,0,1,1)
      GO TO 200
 1000 FORMAT(//,"MOUNT PIECE PLATTER",12," THEN TYPE  :GO",//)
 2000 FORMAT(//,"MOUNT MARKER PLATTER, THEN TYPE  :GO",//)
      END
$EOJ


ASMB
      NAM MP,7
      ENT MPOFF,MPON
      EXT EXEC,.ENTR
UPPA  EQU 2528
MPOFF NOP                 MEMORY PROTECT OFF ENTRY POINT
      JSB .ENTR
      DEF MPOFF
      LDA =B1000          SET UP FWA OF SYSTEM
```

```
        STA LOWA        AND SAVE
        LDA RPOFF       GET RETURN ADDRESS
        STA EXIT        AND SAVE FOR RETURN
NO.1    JSB ISZ         PICK UP INSTRUCTION
        CPA STF0        IS IT A STF 0?
        RSS             YES
        JMP NO.1        NO, GET NEXT INSTRUCTION
        JSB ISZ         PICK UP INSTRUCTION
        CPA STC5        IS THE NEXT INSTRUCTION A STC 5?
        RSS             YES, MEMORY PROTECT INSTRUCTION
        JMP NO.1        NO, GET NEXT INSTRUCTION
        LDA STF0        REPLACE STC 5 WITH A STF 0
        JSB EXEC        BASE PAGE STORE ROUTINE
        DEF *+2         RETURN ADDRESS
        DEF RCODE       REQUEST CODE
        JMP NO.1        CONTINUE
ISZ     NOP
        ISZ LOWA        BUMP INSTRUCTION ADDRESS
        LDA LOWA        PUT CURRENT INSTRUCTION ADDRESS IN B
        CPA OPPA        LWA OF SYSTEM?
        JMP EXIT,I      YES, RETURN
        LDA LOWA,I      NO, PUT INSTRUCTION IN A
        JMP ISZ,I       RETURN
MPON    NOP             MEMORY PROTECT ON ENTRY POINT
        JSB .ENTR
        DEF MPON
        LDA =B1000      SET UP FWA OF SYSTEM
        STA LOWA        AND SAVE
        LDA MPON        GET RETURN ADDRESS
        STA EXIT        AND SAVE FOR RETURN
NO.2    JSB ISZ         PICK UP INSTRUCTION
        CPA STF0        IS IT AN STF 0 INSTRUCTION?
        RSS             YES
        JMP NO.2        NO, GET NEXT INSTRUCTION
        JSB ISZ         PICK UP NEXT INSTRUCTION
        CPA STF0        IS IT ANOTHER STF 0 INSTRUCTION?
        RSS             YES
        JMP NO.2        NO, GET NEXT INSTRUCTION
        LDA STC5        PUT BACK THE STC 5 INSTRUCTION
        JSB EXEC        BASE PAGE STORE CALL
        DEF *+2         RETURN ADDRESS
        DEF RCODE       REQUEST CODE
        JMP NO.2        CONTINUE
STF0    STF 0
STC5    STC 5
RCODE   DEC -17
LOWA    OCT 1000
EXIT    OCT 0
        END

FTN4
      SUBROUTINE MODI(N,IX,IY,IORGP,IMGNO)
C.....ROUTINE TO MODIFY POSITION OF IMAGE
C.....N=1    MOVE IMAGE IN WINDOW
C.....N=2    MOVE IMAGE IN WINDOW AND IN MARKER DATA
C
C
C
      COMMON MOUT,LABF(3),LABR(3),NOSTY,INSTY(31),JBS(15),NAMST(3),
     1 MSUP(5),NSCMD,JBM(2),NAMMK(3),JBG(4),NAMGD(3),JBP(12),
     2 IDMK(5),IDOP,NPEC,NSHD,FACTR,NSIZ,ISIZ(42),WIDTH,EFF,SX,SY,KST,
     3 YARDS,NAP,KINT,KFLG,KINC,KSIZ,KXMN,KYMN,KANG,KXMX,KYMX,KOPT,
     4 KAD,KADP,KDP,CSMX,MKWD,CSMN,NAGE,IMGP,KREL,ITYPE,
     5 NOIMG,IOFSZ,IMGY,IMGE,IMGTC,IMK01,IMK02,IMENU,IDXP,IDYP,
```

```
     6 IXC,IYC,IXL,IXR,IYU,IYL,CS,MDELX,MDELY,JFLG,MLW,MUP,ISLX,ISLY,
     7 TOTAL,MYI,XPC,YPC,IMGLP,IMESG,IERG,ISOFF,SOFF,KSLH,
     8 IMGPS,MAXOF,KSLV,PTC,IOPTN,IAREA,ISIZE,KCSR,KYES,KPSIZ,KPLL,
     9 IT1,IT2,IT3,IT4,IT5,IT6,IT7,IT8,IT9,IT10,
     A IOFSET,KPLH,IFLP,ICUST,IMGPK,
     B T1,T2,T3,T4,T5,T6,T7,T8,T9,T10,
     C JSTY,JPG,JNG,JSHD,JRT,
     D ILPe,IRU,IHD,IDv,IBR,IBL,ITCW,ITCCW,
     E IRCW,IRCCW,ISL,IRE,IFL,IPK,ICOM,
     F IDUM(1)
      IF(IMGNO.EQ.0) RETURN
      M = MAOP+IMGNO
      IF(N.EQ.1) GO TO 50
C.....ADJUST MARKER DATA AND ORIGIN OFFSET
      CALL ORGIN(IX,IY,0,IMGNO)
      CALL VISIB (IMGNO,IORG,IX,IY)
      IORGP = IORG
      IF(IORG.EQ.0) RETURN
      CALL STAT
      CALL ORGIN (IX,IY,IORGP,IMGNO)
C.....ADJUST WINDOW OFFSET
   50 IF(IMGNO.LE.0) RETURN
      IND = IABS(IDUM(M))
   55 IDUM(IND+1) = IRND(IX*CS)*16
      IDUM(IND+2) = IRND(IY*CS)*16+1
      RETURN
      END
      ENDS

FTN4
      PROGRAM MONST
      DIMENSION MP(3),IB(3),N0(3),NJG(3),NPLT(3)
      DIMENSION IRAY(1)
C
      DATA MS/2HSF,2HCT,2HR /
      DATA MP/2HMP,2HRG,2HD /
      DATA NJG/2HJG,2HGN,2H  /
      DATA NPLT/2HPL,2HOT,2HF /
C
      IRAY(0)=1
      IRAY(1)=1220
      IRAY(0)=1
      CALL EXEC(-19)
      IRAY(1)=1093
      IRAY(0)=0
      CALL EXEC(-19)
      IRAY(1)=1166
      IRAY(0)=0
      CALL EXEC(-19)
      IRAY(1)=1890
      IRAY(0)=0
      CALL EXEC(-19)
      IRAY(1)=1892
      IRAY(0)=0
      CALL EXEC(-19)
C
      CALL IDAY(NN)
      WRITE(6,100)
  100 FORMAT("  1      MARKAMATIC SYSTEM START",//)
      CALL EXEC(15,66,0,1,NPLT,0)
      IB(1)=1
      IB(2)=7
      IB(3)=257
      CALL OUSC(0)
```

```
      CALL EXEC(15,66,IB,3,NJU,0)
      CALL PLTST(1,0)
      CALL PLTST(2,0)
      CALL PLTST(3,1)
      CALL PLTST(4,1)
      CALL OVRUP
      IB(1)=1
      IB(2)=0
      CALL EXEC(15,66,IB,3,N0,0)
      CALL ACCES(0,NP)
      CALL EXEC (10,NP)
      END
      ENDS


FTN4
      PROGRAM MRKD,5
C.....SEGMENT TO ROLL OR ZOOM
C.....ITYPE = OPERATION TYPE
C.....9 = ROLL LEFT, 10 = ROLL RIGHT
C.....11 = ZOOM OUT, 12 = ZOOM IN
C
      DIMENSION MK(3)
      COMMON MOUT,LABF(3),LABR(3),NOSTY,INSTY(31),JRS(15),NAMST(3),
     1 MKUP(3),NSCMD,JRY(20),NAMMK(3),JHG(4),NAMGD(3),JBP(10),
     2 IOMK(3),IDOP,NPLC,NSHD,FACTR,NSIZ,ISIZ(42),WIDTH,EFF,SX,SY,KSTL,
     3 YANDS,NAP,KINT,KFLG,KINC,KSIZ,KXMN,KYMN,KANG,KXMX,KYMX,KDPT,
     4 KAD,KADP,KDF,CSIX,MKWD,CSMN,NAGE,IMGP,KREL,ITYPE,
     5 NOIMG,IOFSZ,INGY,IMGE,IMGIC,IMKD1,IMKD2,IMENU,IOXP,IOYP,
     6 IXC,IYC,IXL,IXR,IYU,IYL,CS,MODELX,MODELY,JFLG,MLW,MUP,ISLX,ISLY,
     7 TOTAL,MYL,YPC,YPC,IMGLP,IMESG,IEDG,ISOFF,SOFF,KSLH,
     8 IMGPS,MAXOF,KSLV,KTC,IOPTN,IAREA,ISIZE,KCSR,KYES,KPSIZ,KPLL,
     9 IT1,IT2,IT3,IT4,IT5,IT6,IT7,IT8,IT9,IT10,
     A IOFSET,KPLH,IFLP,ICOST,IMGPK,
     B T1,T2,T3,T4,T5,T6,T7,T8,T9,T10,
     C JSTY,JFG,JNG,JSHD,JRT,
     D ILPR,IBO,IRO,IOV,IBR,IBL,ITCW,ITCLW,
     E IRCR,IRLCR,ISL,IRE,IFL,IPK,ICOM,
     F IOOM(1)
      DATA MK /2HMR,2HKE,2H  /, INDRL /1/, INDZM /1/
  310 INC=100
      IF(MOD(ITYPE,2).EQ.0) INC = -100
      IF(ITYPE.GT.10) GO TO 330
C.....ROLL
  320 NXC = IXC+INC*INDRL
      INDZM = 1
      INDRL = INDRL+1
      IF(INDRL.GT.10) INDRL = 10
  322 M = 32700 - IFIX(2048.0/CS)
      IF(M.LT.0) M = 0
      DIF = FLOAT(NXC)-FLOAT(M)
      IF(DIF.GE.0.0) NXC = M
      DIF = FLOAT(NXC)+FLOAT(M)
      IF(DIF.LE.0.0) NXC = -M
      IF(NXC.EQ.IXC.AND.ITYPE.LE.10) GO TO 400
      INCR = NXC-IXC
      IXC = NXC
      MUPM = MKWD/2
C.....ADJUST X BOUNDS FOR ALL PIECES
      DO 325 I=1,NOIMG
C.....IGNOR SLAVE PIECES
      IF(JLOF(JRM(20),IOOM(KFLG+I)),EQ.1) GO TO 325
      MUPP = IOOM(KYMX+I)
C.....IGNORE PIECES ABOVE THE MARKER
      IF(MUPP-1.LE.MUPM)GO TO 325
```

```
      IIAX = KXMX+I
      IIXN = KXMN+I
      MUPP = IDUM(IIXX)-IDUM(IIXN)
      DIF = FLOAT(IDUM(IIXN))+FLOAT(INCR)
      IF(DIF.LT.-32700.0) DIF = -32700.0
      IDUM(IIXN) = DIF
      IDUM(IIXX) = IDUM(IIXN)+MUPP
  325 CONTINUE
      CALL WINDO
      GO TO 440
C.....ZOOM
  330 IF(CS.EQ.CSMX.AND.INC.GT.0) GO TO 400
      IF(CS.EQ.CSMN.AND.INC.LT.0) GO TO 400
      CS = CS+FLOAT(INC*INDZM)/16235.0
      INDRL = 1
      INDZM = INDZM+1
      IF(INDZM.GT.5) INDZM=5
      IF(CS.GT.CSMX) CS=CSMX
      IF(CS.LT.CSMN) CS=CSMN
      IDUM(KCSR) = IFIX(2047.0*CS)*16+1
      MM = 2048.0/CS+.5
      NYL = IYC-MM
      IF(INC.GT.0.AND.NYL.GT.MYL) IYC = MYL+MM
      IF(INC.LT.0.AND.NYL.LT.MYL) IYC = MYL+MM
  332 IF(IMGP.EQ.0) GO TO 335
      MDELX = IDUM(KXMX+IMGP)-IDUM(KXMN+IMGP)
      MDELY = IDUM(KYMX+IMGP)-IDUM(KYMN+IMGP)
  335 NXC = IXL
      GO TO 322
C.....RETURN WHEN STYLUS PRESSURE IS RELEASED
  400 CALL MRKIT(2,NDXP,NDYP,ITYPE)
      IF(ITYPE.GT.0) GO TO 310
      CALL UDSC(0)
      MK(2) = 2HKJ
      CALL EXEC(6,MK)
      CALL MRKJ
      CALL MARK
      END
      END$


FTN4
      PROGRAM MRKF,5
C.....DEC. 7, 1972
C.....MARK,ARET ROMAN
C.....PROGRAM: MARK
C.....VERSION 2.0
C.....SEGMENT TO SAVE PIECES.  IF JFLG=0, MARKER SAVED
C.....TEMPORARILY(IF MORE EFFICIENT OR REQUESTED BY OPERATOR).
C.....IF JFLG=1, MOST EFFICIENT MARKER OR ONE REQUESTED BY
C.....OPERATOR IS SAVED PERMANENTLY IN "MARKD","MARKR".
C**********DUMMY ARRAY
      DIMENSION IDUMM(64)
      DIMENSION ITEX4(5),IXSP1(15),IXSP2(15)
      DIMENSION IBUF(128),NAMT(3),MK(3),ITEX1(11),ITEX2(7),ITEX3(9)
      COMMON NOUT,LABF(3),LABR(3),NOSTY,INSTY(31),JBS(15),NAMST(3),
     1 MKUP(5),NSCMU,JBM(24),NAMMK(3),JBG(4),NAMGD(3),JBP(12),
     2 IDMK(5),IDDP,NPEC,NSHD,FACTR,NSIZ,ISIZ(42),WIDTH,EFF,SX,SY,KSTL,
     3 YARDS,NAP,KINT,KFLG,KINC,KSIZ,KXMN,KYMN,KANG,KXMX,KYMX,KDPT,
     4 PAD,KADP,KOF,CSMX,MKWD,CSMN,NRGE,IMGP,KREL,ITYPE,
     5 NOIMG,IDF3Z,IMGY,IMGE,IMGTC,IMKG1,IMKG2,IMENU,IDXP,IDYP,
     6 IXC,IYC,IXL,IXR,IYU,IYL,CS,MDELX,MDELY,JFLG,MLW,MUP,ISLX,ISLY,
     7 IOTAL,MYL,XPC,YPC,IMGLP,IMESG,IEDG,ISOFF,SOFF,KSLM,
     8 IMGPS,MAXOF,KSLV,KTC,IOPTN,IAREA,ISIZE,KCSR,KYES,KPSIZ,KPLL,
     9 IT1,IT2,IT3,IT4,IT5,IT6,IT7,IT8,IT9,IT10,
```

```
      A IOFSET,KPLR,IFLP,ICOST,IMGPK,
      B TLML,I2,T3,T4,T5,T6,T7,T8,T9,ILOOP,INSCP,
      C JSTY,JFG,JNG,JSHD,JRT,
      D ILPB,IBB,IBD,IOV,IBR,IBL,ITCW,ITCCW,
      E IRCW,IRCCW,ISL,IRE,IFL,IPK,ICOM,
      F IDUM(1)
       EQUIVALENCE (IBUF   ,EFFIC),(IBUF(4),FAC)
       DATA ITEX1/2HOV,2HCK,2HRI,2HDE,2H E,2HFF,2HIC,2HIE,2HNC,2HY?,2-@ /
       DATA ITEX2/2HMA,2HRK,2HER,2H S,2HAV,2HED,2H@ /
       DATA ITEX3/2HMA,2HRK,2HER,2H N,2HOT,2H S,2HAV,2HED,2H@ /
       DATA ITEX4/2HOR,2HAW,2H N,2HOW,2H?@/
       DATA MK/2HMK,2HKE,2H  /
       DATA NAMT/2HCH,2HEK,2H  /
       DATA NOSPL/4/
C.....READ OLD EFFICTENCY
      CALL EXEC (14,66,IBUF,2,NAMT,0)
C.....MESSAGE DELAY
      CALL MESAG(0,2H @,ITMP)
      IF(JFLG.EQ.1) GO TO 500
C.....TEMPORARY STORE
      IF(IBUF   .EQ.0) GO TO 100
      IF(EFFIC.LT.EFF) GO TO 100
C.....DISPLAY "OVERRIDE EFFICIENCY?", "YES OR NO"
      CALL MESAG (1,ITEX1,IFLG)
      IF(IFLG.EQ.0) GO TO 200
C.....STORE CURRENT MARKER IN "CHEK"
  100 CALL EXEC(15,66,EFF,MAXOF+ICOM,NAMT,0)
C.....DISPLAY "MARKER SAVED"
      CALL MESAG (0,ITEX2,IFLG)
      GO TO 210
C.....DISPLAY "MARKER NOT SAVED"
  200 CALL MESAG (2,ITEX3,IFLG)
C.....RETURN TO INTERACTIVE LOOP
  210 JFLG = 1
      CALL EXEC (8,MK)
      CALL MARK
      CALL MRKE
C.....STORE PERMANENTLY
C.....DISPLAY "PLOT NOW?", "YES OR NO"
  500 CALL MESAG(1,ITEX4,IPLOT)
      IF(IBUF   .EQ.0) GO TO 600
      IF(EFFIC.LT.EFF) GO TO 600
C.....DISPLAY "OVERRIDE EFFICIENCY?", "YES OR NO"
      CALL MESAG (1,ITEX1,IFLG)
      IF(IFLG.EQ.1) GO TO 600
C.....STORE OLD MARKER, READ IN FROM "CHEK"
  550 CALL STOP
      CALL EXEC(14,66,EFF,MAXOF+ICOM,NAMT,0)
C.....CREAT MARKER FILE DIRECTORY NAME
  600 JTMP = 4016
C.....SAVE IDOP SO DIREC CAN USE IT AS 6TH WORD OF IDMK
      LTMP = IDOP
      IDOP = 0
      IF(IDMK.EQ.MKUP.AND.IDMK(2).EQ.MKUP(2).AND.IDMK(3).EQ.
     1  MKUP(3).AND.IDMK(4).EQ.MKUP(4).AND.IDMK(5).EQ.MKUP(5))
     2  JTMP = 10016
      CALL DIREC (JTMP,IDMK,ITMP,NSEC,ISTAT)
      IDOP = LTMP
C.....PLACE SPLICE MARKS?
C     IF(JLOF(JFLD(6,1),IOPTN).EQ.1) CALL SPLIC(NOSPL,IXSP1,IXSP2)
      CALL STOP
      IF(ISTAT.EQ.0) GO TO 620
C.....ERROR CONDITIONS
      GO TO (605,610,615,617,610),ISTAT
```

```
  605 WRITE(MOUT,1000)
 1000 FORMAT ("***MARKER PLATTER FULL")
      CALL RTN
  610 CALL ERR(0,0)
  615 WRITE(MOUT,1001)
 1001 FORMAT ("***ERROR ON REMOVABLE PLATTER")
      CALL RTN
  617 WRITE(MOUT,1002)
 1002 FORMAT ("***DUPLICATE MARKER ID")
      CALL RTN
C.....STORE DATA INTO "MARKR", HEADER INFO.
  620 NSC = 2*(NOIMG-1)/15
C.....CHECK FOR ROOM IN "MARKR"
      CALL EXEC (16,NAMMK,NSCT)
      IF(NSEC+NSC.LE.NSC1) GO TO 625
C.....NO ROOM, PURGE DIRECTORY ENTRY
      CALL EXEC (15,66,1,1,NAMMK,NSEC)
      CALL DIREC(2HM1M,IDMK,ITMP,NSEC,ISTAT)
      GO TO 605
  625 IBUF = 0
C.....WORD 1
      CALL JSTF(NOIMG,JBM   ,IBUF   )
      CALL JSTF(NSC  ,JBM(3),IBUF   )
C.....WORD 2
      IBUF(2) = 0
      CALL JSTF(NOSTY,JBM(4),IBUF(2))
      CALL JSTF(NOSPL,JBM(16),IBUF(2))
      CALL JSTF(IDOP ,JBM(2) ,IBUF(2))
C.....WORD 3
      IBUF(3) = 0
      CALL JSTF(NSHD,JBM(5),IBUF(3))
C.....WORDS 4,5,6,7
      FAC = FACTR
      IBUF(6) = IRND(WIDTH*100.0)
      IBUF(7) = (YARDS*36.0)/FACTR-32600.0
C.....WORD 8
      IBUF(8) = NRGE
C.....WORD 9
      IBUF(9) = IRND(EFF*100.0)
C.....WORD 10
      IBUF(10) = IRND(SOFF*100.0)
C.....WORD 11
      IBUF(11) = IRND(SY*100.0)
C.....WORD 12
      IBUF(12) = IRND(SX*100.0)
      IDX = 14
      ITMP = (XPC-1.0)*1000.0
      JTMP = (YPC-1.0)*1000.0
      CALL JSTF(ITMP,JBM(9),IBUF(13))
      CALL JSTF(JTMP,JBM(10),IBUF(13))
C.....STYLE & SIZES
      DO 640 I=1,NOSTY
      NSZ = 0
      ILR = 1
      JDX = 1
      DO 630 J=1,NSIZ
      IF(JLOF(JBM(9),ISIZ(J)).NE.I) GO TO 630
C.....STORE TWO SIZES PER WORD
      NSZ = NSZ+1
      ISZ = JLOF(JBM(10),ISIZ(J))
      CALL JSTF(ISZ,JBM(5+ILR),IBUF(IDX+JDX))
      ILR = 3-ILR
      IF(ILR.EQ.1) JDX = JDX+1
  630 CONTINUE
      IF(ILR.EQ.1) GO TO 635
```

```
      CALL JSTF(0,JBM(10),IBUF(IDX+JDX))
      JDX = JDX+1
  035 CALL JSTF(JLOF(JSTY,INSTY(I)),JBM(7),IBUF(IDX))
      CALL JSTF(NSZ,JBM(8),IBUF(IDX))
      IDX = IDX+JDX
  040 CONTINUE
C.....STORE SPLICE MARKS
      IDX = 76
      IF(NOSPL.EQ.0) GO TO 047
      DO 045 I=1,NOSPL
      IBUF(IDX) = IXSP1(I)
      IBUF(IDX+1) = IXSP2(I)
  045 IDX = IDX+2
  047 CALL UDSC(1)
C.....STORE ACTUAL LENGTH IN WORD 128
      IBUF(128)=(TLNL*36.0)/FACTR-32600.0
      CALL EXEC (15,06,IBUF,128,NAMMK,NSEC)
C.....STORE DATA INTO "MARKR", PIECE INFO,
      IND1 = 1
C.....READ INTERNAL # AND % FROM "CHEK"
      CALL UDSC(0)
      NWDS = NPEC*3
      N = 2*((MAXDF+ICOM)/128)+2
      CALL EXEC(14,06,IBUF(KAD+1),NWDS,NAMT,N)
      DO 049 I=1,NPEC
      IBUF(KINT+I) = IBUF(KAD+1)
      IBUF(KINC+I) = IBUF(KAD+2)
  049 KAD=KAD+3
  051 NSEC = NSEC+1
      IBUF    = 0
      IBUF(2) = 0
      IDX = 3
  055 IF(IDX.GE.128) GO TO 700
C.....STORE DATA FOR A PIECE
      IBUF(IDX) = IBUF(KINT+IND1)
      IBUF(IDX+1) = IBUF(KFLG+NPEC+IND1)
      CALL JSTF(JLOF(JFG,IBUF(KFLG+IND1)),JBM(17),IBUF(IDX+1))
      CALL JSTF(JLOF(JNG,IBUF(KFLG+IND1)),JBM(18),IBUF(IDX+1))
      IBUF(IDX+2) = IBUF(KFLG+IND1)
      ISZ = JLOF(JBM(9),IBUF(KSIZ+IND1))
      IST = JLOF(JBM(9),ISIZ(ISZ))
      CALL JSTF(JLOF(JBS(2),INSTY(IST)),JBM(12),IBUF(IDX+2))
      IBUF(IDX+3) = IBUF(KINC+IND1)
      IBUF(IDX+4) = IBUF(KSIZ+IND1)
      IF(JLOF(JBM(20),IBUF(KFLG+IND1)).EQ.0) GO TO 657
C.....SLAVE PIECE, POSITION WITH MASTER
      INUM = JLOF(JBM(7),IBUF(KFLG+IND1+NPEC))
      IBUF(IDX+5) = IBUF(KXMN+INUM)
      IBUF(IDX+6) = IBUF(KYMN+INUM)
      IBUF(IDX+7) = IBUF(KANG+INUM)
      GO TO 658
  657 IBUF(IDX+5) = IBUF(KXMN+IND1)
      IBUF(IDX+6) = IBUF(KYMN+IND1)
      IBUF(IDX+7) = IBUF(KANG+IND1)
  658 IDX = IDX+8
      IND1 = IND1+1
      IF(IND1.LE.NOIMG) GO TO 655
      IF(IDX.GE.128) GO TO 700
      DO 660 I=IDX,128
  660 IBUF(I) = 0
  700 CALL UDSC(1)
      CALL EXEC (15,06,IBUF,128,NAMMK,NSEC)
      IF(IND1.LE.NOIMG) GO TO 650
C.....ZERO OUT "CHEK"
      CALL UDSC(0)
```

```
      CALL EXEC (15,66,M,1,NAMT,0)
C.....PRINT IF "FLIP" OR "PACK" ON OPTION
      IF(JLOF(JFLD(5,1),IOPTN).EQ.0) GO TO 7#1
      IF(JLOF(JFLD(4,1),ICUST).EQ.1) WRITE(MOUT,1003)
 1003 FORMAT ("PACK WAS USED")
      IF(JLOF(JFLD(1,1),ICUST).EQ.1) WRITE(MOUT,1004)
 1004 FORMAT ("FLIP WAS USED")
C.....RETURN TO MONITOR
  7#1 IBUF    = 2
      IBUF(2) = 1
      NAMT    = 2HSE
      NAMT(2) = 2HCT
      NAMT(3) = 2HR
      MK    = 2HWP
      MK(2) = 2HF0
      MK(3) = 2H0
      IF(IPLOT.EQ.#) GO TO 71#
      IBUF(2) = 5
      DO 7#5 I=1,5
  7#5 IBUF(2+I) = IDMK(I)
      IBUF(8) = 1
      IBUF(9) = 4
      CALL EXEC (15,66,IBUF,9,NAMT,0)
      CALL EXEC (1#,MK)
  71# CALL EXEC (15,66,IBUF,2,NAMT,0)
      CALL EXEC (1#,MK)
      END
      ENDS

FTN4
      PROGRAM MRKG,5
C.....BUMP PIECES SUCH THAT THEY ARE TANGENT,
C.....AND CHECK OVERLAP
      DIMENSION IDUMM(222)
      DIMENSION MK(3),MNX(4),LMX(2,3#),LMY(2,3#),INTPC(3#),
     1          IX1(2#1),IY1(2#1),IX2(2#1),IY2(2#1),
     2         IX(4),IY(4)
      DIMENSION ITEX1(6),ITEX2(12),ITEX3(8)
      COMMON MOUT,LABF(3),LABR(3),NOSTY,INSTY(31),JBS(15),NAMST(3),
     1 MAUP(5),NSCNU,JMM(27),NAMMK(3),JBG(4),NAMGD(3),JBP(1#),
     2 IDMX(5),IDOP,MPEC,NSMD,FACTR,NSIZ,ISIZ(42),WIDTH,EFF,SX,SY,KSTL,
     3 YARDS,NAP,PINT,KFLG,KINC,KSIZ,KXMN,KYMN,KANG,KXMX,KYMX,KOPT,
     4 KAD,KADP,KDF,CSMX,MAND,CSMN,NRGE,IMGP,KREL,ITYPE,
     5 NOIMG,IDFSZ,IMGY,IMGF,IMGTC,IMKO1,IMKO2,IMENU,IDXP,IDYP,
     6 IXC,IYC,IXL,IXR,IYU,IYL,CS,MDELX,MDELY,JFLG,MLW,MUP,ISLX,ISLY,
     7 TOTAL,MYL,XPC,YPC,IMGLP,IMESG,IEDG,ISOFF,SOFF,KSLH,
     8 IMGPS,MAXOF,KSLV,KTC,IOPTN,IAREA,ISIZE,KCSR,KYES,KPSIZ,KPLL,
     9 IMGIN,ILPEC,IT3,IT4,IT5,IT6,IT7,IT8,IT9,IT1#,
     A IOFSET,KPLH,IFLP,ICUST,IMGPK,
     B T1,T2,T3,T4,T5,T6,T7,T8,T9,T1#,
     C JSTY,JFG,JNG,JSMD,JRT,
     D ILPN,IBU,IBD,IOV,IBR,IBL,ITCW,ITCCW,
     E IRCW,IPCCW,ISL,IRE,IFL,IPK,ICOM,
     F IDUM(1)
      DATA LMT/241/
      ITEX1    = 2H N
      ITEX1(2) = 2H0
      ITEX1(3) = 2HOV
      ITEX1(4) = 2HER
      ITEX1(5) = 2HLA
      ITEX1(6) = 2HP@
      ITEX2    = 2HTO
      ITEX2(2) = 2H0
      ITEX2(3) = 2HMA
```

```
      ITEX2(4) = 2HNY
      ITEX2(5) = 2H P
      ITEX2(6) = 2HIE
      ITEX2(7) = 2HCE
      ITEX2(8) = 2HS
      ITEX2(9) = 2HIN
      ITEX2(10)= 2H A
      ITEX2(11)= 2HRE
      ITEX2(12)= 2HA@
      ITEX3    = 2HPI
      ITEX3(2) = 2HEC
      ITEX3(3) = 2HE
      ITEX3(4) = 2HTO
      ITEX3(5) = 2HO
      ITEX3(6) = 2HLA
      ITEX3(7) = 2HRG
      ITEX3(8) = 2HE@
      ICM = 0
      ICI = 0
      ICO = 0
      MK    = 2HMR
      MK(2) = 2HKE
      IF(LABR.EQ.2) MK(2) = 2HKR
      MK(3) = 2H
      JCTR = 0
      IF(IMGP.EQ.0) GO TO 900
C.....SEND MASK TO READ "PICK" FOR ABORTING OPERATION
      CALL PSHIT(MMMM,100000B)
      MMMM = LABR
      LABR = IABS(LABR)
      JXMN = KXMN+IMGP
      JXMX = KXMX+IMGP
      JYMN = KYMN+IMGP
      JYMX = KYMX+IMGP
      MXP = IEDG
      MXN = -32000
      IF(IDUM(JXMN).GT.IEDG) MXN = IEDG
      DMXN = FLOAT(MXN)
      DMXP = FLOAT(MXP)
      MYP = MKND/2
      MYN = -(MKND-MYP)
      IF(ITYPE.NE.IOV) GO TO 1
      IXP = 0
      IXN = 0
      IYP = 0
      IYN = 0
    1 MNX1 = IDUM(JXMN)
      MXX1 = IDUM(JXMX)
      MNY1 = IDUM(JYMN)
      MXY1 = IDUM(JYMX)
      JCTR = JCTR+1
C.....TRY FIVE TIMES TO CORRECT
      IF(JCTR.GT.5) GO TO 920
      MDLX = MXX1-MNX1
      MDLY = MXY1-MNY1
      MNX    = MNX1-1
      MNX(2) = MXX1+1
      MNX(3) = MNY1-1
      MNX(4) = MXY1+1
      IF(ITYPE.NE.IOV) GO TO 2
      IF(MNX1.LE.MXN) IXN = 1
      IF(MXY1.GE.MYP) IYP = 1
      IF(MNY1.LE.MYN) IYN = 1
C.....CHECK FOR BLOCKED PIECE
      IF(ISLX*ISLY.NE.0.AND.JLOF(JBM(15),IDUM(KFLG+IMGP)).EQ.1) GO TO 6
```

```
      IF(ISLY.EQ.0) GO TO 3
      IF(IDUM(KSTL+IMGP).EQ.9999) GO TO 3
C.....SPECIAL STRIPE GROUP?
      IF(SY.EQ.99.0.AND.JLOF(JBM(13),IDUM(KFLG+NPEC*IMGP)).EQ.0)GO TO 3
      IYP = 1
      IYN = 1
      GO TO 6
    3 IF(ISLX.EQ.0) GO TO 6
      IF(IDUM(KPLL+IMGP).EQ.9999) GO TO 6
      IXP=1
      IXN=1
      GO TO 6
    2 IF(ITYPE.EQ.IBL) IDIR = 1
      IF(ITYPE.EQ.IBR) IDIR = 2
      IF(ITYPE.EQ.IBD) IDIR = 3
      IF(ITYPE.EQ.IBU) IDIR = 4
      TMP=FLOAT(MNX(IDIR))+(-1**IDIR)*400.0
      TMP = AMIN1(TMP,32000.0)
      TMP = AMAX1(TMP,-32000.0)
      MNX(IDIR) = TMP
    6 J = 1
      DO 10 I=1,NOIMG
      IF(I.EQ.IMGP) GO TO 10
C.....SLAVE PIECE?
      IF(JLOF(JBM(20),IDUM(KFLG+I)).EQ.1) GO TO 10
      MNX2 = IDUM(KXMN+I)-1
      MXX2 = IDUM(KXMX+I)+1
      MNY2 = IDUM(KYMN+I)-1
      MXY2 = IDUM(KYMX+I)+1
      IF(MNY2.GE.MYP) GO TO 10
      IF(MNX2.GE.MNX(2).OR.MXX2.LE.MNX   ) GO TO 10
      IF(MNY2.GE.MNX(4).OR.MXY2.LE.MNX(3)) GO TO 10
      IF(ITYPE.NE.IOV) GO TO 7
      LMX(1,J) = MAX0(MNX   ,MNX2)
      LMX(2,J) = MIN0(MNX(2),MXX2)
      LMY(1,J) = MAX0(MNX(3),MNY2)
      LMY(2,J) = MIN0(MNX(4),MXY2)
      ICM = ICM+4
    7 INTPC(J) = I
      J = J+1
      IF(J.LE.30) GO TO 10
C.....DISPLAY "TOO MANY PIECES IN AREA"
      CALL MESAG (0,ITEX2,ITMP)
      GO TO 930
   10 CONTINUE
      IF(ITYPE.EQ.IOV) GO TO 500
C.....BUMP UP
      IF(J.NE.1) GO TO 435
  405 IF(LABR.EQ.2) GO TO 900
      GO TO (410,415,420,425),IDIR
  410 IDUM(JXMN) = MXN
      IDUM(JXMX) = MXN+MDLX
      GO TO 430
C.....BUMP RIGHT TO LIMIT ONLY IF LEFT OF MXP
  415 IF(IDUM(JXMX).GT.MXP) GO TO 430
      IDUM(JXMX) = MXP
      IDUM(JXMN) = MXP-MDLX
      GO TO 430
  420 IDUM(JYMN) = MYN
      IDUM(JYMX) = MYN+MDLY
      GO TO 430
  425 IDUM(JYMX) = MYP
      IDUM(JYMN) = MYP-MDLY
  430 CALL VISIB (IMGP,IDRG,IDXP,IDYP)
      IF(IDRG.EQ.0) GO TO 432
```

```
      CALL ORGIN(IOXP,IOYP,IORG,IMGP)
      CALL MODI(1,IOXP,IOYP,IORG,IMGP)
  431 CALL MODI(1,IOXP,IOYP,0,0)
      IF(ITYPE.EQ.IOV) GO TO 1
      GO TO 900
  432 III = IDUM(KADP+IMGP)
      CALL LITE(0,KADP+IMGP)
      IOXP = 0
      IOYP = 0
      IMGP = 0
      GO TO 431
  435 NINT = J-1
      IDIST = 9999
      JSTAT = LMT
      CALL GENXY (IMGP,IX1,IY1,IXY1,IFP,JSTAT)
      IF(JSTAT.EQ.1) GO TO 910
      IXY1 = IXY1+1
      IX1(IXY1) = IX1
      IY1(IXY1) = IY1
      IF(ISSW(14).LT.0) WRITE(6,1001)IFP,(IX1(KK),IY1(KK),KK=1,IXY1)
      DO 439 I=1,NINT
      JSTAT = LMT
      CALL GENXY (INTPC(I),IX2,IY2,IXY2,JFP,JSTAT)
      IF(JSTAT.EQ.1) GO TO 910
      IXY2 = IXY2+1
      IX2(IXY2) = IX2
      IY2(IXY2) = IY2
      IF(ISSW(14).LT.0)WRITE(6,1001)JFP,(IX2(KK),IY2(KK),KK=1,IXY2)
      JTC = 2
      NBEG2 = 1
      NVAL2 = IXY2
      IFTT=0
  440 IF(NBEG2.LT.IXY2) GO TO 442
      IF(IFTT.EQ.1) GO TO 450
      IFTT=1
      IF(NBEG2.EQ.IXY2) GO TO 450
      NBEG2=NBEG2-IXY2
      NVAL2 = IXY2-NXY2+3
  442 CALL FALL(MNX,MNX(3),IX2(NBEG2),IY2(NBEG2),NVAL2,JX2,NXY2,JTC)
      IF(ISSW(14).LT.0) WRITE(6,1001) NXY2,JX2
      IF(NXY2.LE.0) GO TO 446
      JX2 = JX2+NBEG2-1
      IF(NBEG2.EQ.1) KX2 = JX2
      JTC = 1
      NBEG2 = JX2+NXY2-1
      NVAL2 = IXY2-NBEG2+1
C.....MINIMUM DISTANCE FROM INTPC TO IMGP
      DO 445 J = 1,NXY2
      JJ = INDEX(JX2+J-1,IXY2)
      ID = IDST(IX2(JJ),IY2(JJ),IX1,IY1,IXY1,IDIR,IFP)
      IF(ISSW(14).LT.0) WRITE(6,1001) ID
      ICO = ICO+IXY1-1
      IF(ID.LT.IDIST) IDIST = ID
  445 CONTINUE
      IF(ISSW(14).LT.0) WRITE(6,1001) IDIST
C.....MINIMUM DISTANCE FROM IMGP TO INTPC
      DO 447 J=1,IXY1
      NXY = NXY2
      IF(JX2.LE.IXY2) GO TO 1446
      JX2 = MOD(JX2,IXY2)
      GO TO 446
 1446 IF(NBEG2.LE.IXY2) GO TO 446
      NXY = MOD(NBEG2,IXY2)
      ID = IDST(IX1(J),IY1(J),IX2,IY2,NXY,-IDIR,JFP)
      IF(ISSW(14).LT.0) WRITE(6,1001) ID
```

```
      IF(ID.LT.IDIST) IDIST=ID
      NXY = NXY2-NXY
  440 ID = IDST(IX1(J),IY1(J),IX2(JX2),IY2(JX2),NXY,-IDIR,JFP)
      IF(ISSW(14).LT.0) WRITE(6,1001) ID
      ICO = ICO+NXY2-1
      IF(ID.LT.IDIST) IDIST=ID
  447 CONTINUE
      IF(ISSW(14).LT.0) WRITE(6,1001)IDIST
      GO TO 440
  448 IF(NBEG2.EQ.1) GO TO 450
      NXY2 = IXY2-KX2+3
      NBEG2 = IXY2+1
      GO TO 440
  450 CONTINUE
      IF(IDIST.GE.9998) GO TO 405
      IDIST = IDIST*(-1**IDIR)
      IF(LABR.EQ.2) GO TO 940
      GO TO (460,465,470,470),IDIR
  460 DXN = FLOAT(IDUM(JXMN))+FLOAT(IDIST)
      IDUM(JXMN) = MAX1(DXN,DMXN)
      ICM = ICM+1
      IDUM(JXMX) = IDUM(JXMN)+MDLX
      GO TO 430
C.....BUMP RIGHT TO LENGTH
  465 IF(MXN.EQ.MXP) GO TO 450
      DXN = FLOAT(IDUM(JXMX))+FLOAT(IDIST)
      IDUM(JXMX) = MIN1(DXN,DMXP)
      IDUM(JXMN)=IDUM(JXMX)-MDLX
      GO TO 430
  470 IDUM(JYMX) = IDUM(JYMX)+IDIST
      IDUM(JYMN) = IDUM(JYMN)+IDIST
      IF(IDUM(JYMX).GT.MYP) GO TO 425
      IF(IDUM(JYMN).LT.MYN) GO TO 420
      GO TO 430
C.....OVERLAP
  500 IF(J.EQ.1) GO TO 930
C.....CONTINUE ?
      IF(JCTR.EQ.1) GO TO 501
      CALL FSHIT(ITMP,10000006)
      IF(ITMP.EQ.IPK) GO TO 920
  501 NINT=J-1
      IF(ISSW(14).LT.0) WRITE(6,1001)(INTPC(KK),KK=1,NINT),LMX,LMY
      JSTAT = LMT
      CALL GENXY (IMGP,IX1,IY1,IXY1,IFP,JSTAT)
      IF(JSTAT.EQ.1) GO TO 910
      IF(ISSW(14).LT.0) WRITE(6,1001)IFP,(IX1(KK),IY1(KK),KK=1,IXY1)
C.....CHECK WHERE POSSIBLE CONFLICT MAY OCCUR FOR A
C.....POSSIBLE INTERSECTION.
      DO 400 I=1,NINT
      JSTAT = LMT
      CALL GENXY (INTPC(I),IX2,IY2,IXY2,JFP,JSTAT)
      IF(JSTAT.EQ.1) GO TO 910
      IF(ISSW(14).LT.0) WRITE(6,1001)JFP,(IX2(KK),IY2(KK),KK=1,IXY2)
      NBEG1 = 1
      NVAL1 = IXY1
   20 NBEG2 = 1
      NVAL2 = IXY2
      IF(NBEG1.GE.IXY1) GO TO 400
C.....FIND A STRING OF POINTS IN CONFLICT AREA FOR
C.....PIECE, IMGP.
      JTC = 1
      IF(NVAL1.EQ.IXY1) JTC = 2
      CALL FALL(LMX(1,I),LMY(1,I),IX1(NBEG1),IY1(NBEG1),
     1  NVAL1,JX1,NXY1,JTC)
      IF(ISSW(14).LT.0) WRITE(6,1001)NXY1,JX1
```

```
      IF(NXY1.LE.0) GO TO 400
      JX1 = JX1+NBEG1-1
      ITMP = JX1+NXY1-1
      IF(NBEG1.EQ.1.AND.MOD(ITMP,IXY1).GT.1) ITMP=MOD(ITMP,IXY1)
      NBEG1 = ITMP
      NVAL1 = IXY1-NBEG1+1
   20 IF(NBEG2.GE.IXY2) GO TO 20
C.....FIND A STRING OF POINTS IN CONFLICT AREA FOR
C.....PIECE, INTPC(I)
      JTC = 1
      IF(NVAL2.EQ.IXY2) JTC = 2
      CALL FALL(LMX(1,I),LMY(1,I),IX2(NBEG2),IY2(NBEG2),
     1 NVAL2,JX2,NXY2,JTC)
      IF(ISSW(14).LT.0) WRITE(6,1001) NXY2,JX2
      IF(NXY2.LE.0) GO TO 20
      JX2 = JX2+NBEG2-1
      ITMP = JX2+NXY2-1
      IF(NBEG2.EQ.1.AND.MOD(ITMP,IXY2).GT.1) ITMP=MOD(ITMP,IXY2)
      NBEG2 = ITMP
      NVAL2 = IXY2-NBEG2+1
C.....CHECK FOR INTERSECTION OF LINESEGMENTS
  115 IX    = IX1(JX1)
      IY    = IY1(JX1)
      IF(ISSW(14).LT.0) WRITE(6,1001) LMX(1,I),LMX(2,I),LMY(1,I),
     1  LMY(2,I),NXY1,NXY2
      IFLG = 0
C.....FLAG FOR HORIZONTAL/VERTICAL OVERLAP
      IHV = 0
      IEND = NXY1-1
      IF(NXY1.EQ.IXY1) IEND = NXY1
      DO 150 II=1,IEND
      III = INDEX(JX1+II,IXY1)
      IX(2) = IX1(III)
      IY(2) = IY1(III)
      IF(IX    .EQ.IX(2).AND.IY    .EQ.IY(2)) GO TO 150
      IX(3) = IX2(JX2)
      IY(3) = IY2(JX2)
      JEND = NXY2-1
      IF(NXY2.EQ.IXY2) JEND = NXY2
      DO 140 JJ=1,JEND
      JJJ = INDEX(JX2+JJ,IXY2)
      IX(4) = IX2(JJJ)
      IY(4) = IY2(JJJ)
      IF(ISSW(14).LT.0) WRITE(6,6789) (IX(KKK),IY(KKK),KKK=1,4)
 6789 FORMAT("INT.",8I7)
      IF(IX(3).EQ.IX(4).AND.IY(3).EQ.IY(4)) GO TO 130
C.....ALWAYS CHECK AFTER TWO INTERSECTIONS FOUND
      IF(IFLG.EQ.2) GO TO 1120
C.....ELIMINATE IMPOSSIBLE CASES DUE TO SEGMENTS BOUNDS
      IMXV = MAX0(IX(3),IX(4))
      IMNV = MIN0(IX(3),IX(4))
      JMXV = MAX0(IX,IX(2))
      JMNV = MIN0(IX,IX(2))
      IF(ISSW(14).LT.0) WRITE(6,6789) IMXV,IMNV,JMXV,JMNV
      IF(FLOAT(JMXV).LT.FLOAT(IMNV).OR.FLOAT(JMNV).GT.FLOAT(IMXV))
     + GO TO 130
      IMXV = MAX0(IY(3),IY(4))
      IMNV = MIN0(IY(3),IY(4))
      JMXV = MAX0(IY,IY(2))
      JMNV = MIN0(IY,IY(2))
      IF(ISSW(14).LT.0) WRITE(6,6789) IMXV,IMNV,JMXV,JMNV
      IF(JMXV.LT.IMNV.OR.JMNV.GT.IMXV) GO TO 130
 1120 CONTINUE
      IF(ISSW(14).LT.0) WRITE(6,6789) (IX(KKK),IY(KKK),KKK=1,4)
      CALL INTST (ITFLG,IX,IY,XINT,YINT)
```

```
      IF(ISSW(14).LT.0) WRITE(5,6790) ITFLG,XINT,YINT,IFLG
 6790 FORMAT(I8,1X,2F10.5,I8)
      IF(IFLG.EQ.2.AND.ITFLG.NE.-1) GO TO 160
      IF(ITFLG)132,130,120
C.....CHECK FOR TANGENT POINT
  120 INTRP = 0
      IF(XINT.EQ.FLOAT(IX).AND.YINT.EQ.FLOAT(IY))INTRP=1
      IF(XINT.EQ.FLOAT(IX(2)).AND.YINT.EQ.FLOAT(IY(2)))INTRP=2
      IF(XINT.EQ.FLOAT(IX(3)).AND.YINT.EQ.FLOAT(IY(3)))INTRP=3
      IF(XINT.EQ.FLOAT(IX(4)).AND.YINT.EQ.FLOAT(IY(4)))INTRP=4
      IF(INTRP.EQ.0) GO TO 116
C.....CHECK POINT BEFORE AND AFTER FOR POSITION OF IN OR OUT
      IF(INTRP.LE.2) GO TO 117
C.....POINT ON INTPC(I)
      INTRP = JJJ+INTRP-4
      JXP = IDUM(KXMN+IMGP)-1
      JYP = IDUM(KYMN+IMGP)-1
      KKK = INDEX(INTRP-1,IXY2)
      IN1 = INOUT(IX2(KKK),IY2(KKK),IX1,IY1,IXY1,JXP,JYP)
      IF(ISSW(14).LT.0) WRITE(6,1001) IX2(KKK),IY2(KKK),IN1
      KKK = INDEX(INTRP+1,IXY2)
      IN2 = INOUT(IX2(KKK),IY2(KKK),IX1,IY1,IXY1,JXP,JYP)
      IF(ISSW(14).LT.0) WRITE(6,1001) IX2(KKK),IY2(KKK),IN2
      IF(IN1.EQ.IN2) GO TO 130
      IF(ISSW(14).LT.0) WRITE(6,6788)
 6788 FORMAT("INTER.")
      GO TO 116
C.....POINT ON IMGP
  117 INTRP = III+INTRP-2
      ITMP = INTPC(I)
      JXP = IDUM(KXMN+ITMP)-1
      JYP = IDUM(KYMN+ITMP)-1
      KKK = INDEX(INTRP-1,IXY1)
      IN1 = INOUT(IX1(KKK),IY1(KKK),IX2,IY2,IXY2,JXP,JYP)
      IF(ISSW(14).LT.0) WRITE(6,1001) IX1(KKK),IY1(KKK),IN1
      KKK = INDEX(INTRP+1,IXY1)
      IN2 = INOUT(IX1(KKK),IY1(KKK),IX2,IY2,IXY2,JXP,JYP)
      IF(ISSW(14).LT.0) WRITE(6,1001) IX1(KKK),IY1(KKK),IN2
      IF(IN1.EQ.IN2) GO TO 132
      IF(ISSW(14).LT.0) WRITE(6,6788)
      GO TO 116
  116 IF(IFLG.EQ.1) GO TO 127
      IPT1 = II-1+JX1
      IPT2 = JJ+JX2
      IIX = IRND(XINT)
      IIY = IRND(YINT)
      IFLG = 1
      GO TO 130
  127 IF(IRND(XINT).EQ.IIX.AND.IRND(YINT).EQ.IIY) GO TO 130
      JPT1=II+JX1
      JPT2 = JJ-1+JX2
      JJX = IRND(XINT)
      JJY = IRND(YINT)
      IF(JPT1-IPT1.NE.1.AND.IFP.EQ.JFP) GO TO 129
      IPT2 = IPT2-1
      JPT2 = JPT2+1
  129 ICTR = 0
      IF(JPT2.GT.IPT2) GO TO 131
      ITMP = JPT2
      JPT2 = IPT2
      IPT2 = ITMP
  131 IJPT1 = 0
      IJPT2 = 0
      IF(JPT1-IPT1.LE.1) IJPT1 = 1
      IF(JPT2-IPT2.LE.1) IJPT2 = 1
```

```
      IF(ISSW(14).LT.0) WRITE(6,1001) IPT1,JPT1,IPT2,JPT2
      IFLG = 2
      GO TO 134
C.....FIRST INTERSECTION IS ON TANGENT LINE SO OMIT
  132 IFLG = 0
  134 IX(3) = IX(4)
      IY(3) = IY(4)
  140 CONTINUE
      IX = IX(2)
      IY = IY(2)
  150 CONTINUE
      IF(IFLG.NE.2) GO TO 25
  160 ICTR = ICTR+1
      IF((IXP*IXP+IYN*IYP).EQ.0.OR.ICTR.LT.10) GO TO 165
C.....PIECE NO LONGER MOVABLE
      GO TO 920
  165 IDIST = 0
      IDIR = 1
      IF(ISSW(14).LT.0) WRITE(6,1001) IIX,IIY,JJX,JJY
      IF(JJX.EQ.IIX.AND.JJY.EQ.IIY) GO TO 25
      IF(IABS(JJX-IIX).GT.IABS(JJY-IIY)) IDIR = 2
      IF(IDIR.EQ.1.AND.(IXP*IXN).EQ.1) GO TO 167
      IF(IDIR.EQ.2.AND.(IYP*IYN).EQ.1) GO TO 169
      GO TO 170
  167 IDIR = 2
      GO TO 170
  169 IDIR = 1
  170 IF(IJPT1.EQ.1.AND.IJPT2.EQ.1) GO TO 25
      IF(IJPT1.EQ.1) GO TO 185
      KBEG = IPT1+1
      KEND = JPT1-1
      KFTT = 0
      DO 180 K=KBEG,KEND
      KK = INDEX(K,IXY1)
      NXY = JPT2-IPT2+1
      IDT = 9999
      IF(JPT2.LE.IXY2) GO TO 175
      NXY = MOD(JPT2,IXY2)
      IDT = JDST(IX1(KK),IY1(KK),IX2,IY2,NXY,IDIR)
      NXY = JPT2-IPT2+2-NXY
      IX2(IXY2+1) = IX2
      IY2(IXY2+1) = IY2
  175 ID = JDST(IX1(KK),IY1(KK),IX2(IPT2),IY2(IPT2),NXY,IDIR)
      IF(IABS(ID).GT.IABS(IDT))ID=IDT
      IF(IABS(ID).EQ.9999) GO TO 180
      IF(IABS(ID).GT.IABS(IDIST)) IDIST=ID
      KFTT = 1
  180 CONTINUE
      IF(KFTT.EQ.0) IDIST=9999
      IF(ISSW(14).LT.0) WRITE(6,1001) IDIST
  185 IF(IJPT2.EQ.1) GO TO 195
      KBEG = IPT2+1
      KEND = JPT2-1
      KFTT = 0
      DO 190 K=KBEG,KEND
      KK = INDEX(K,IXY2)
      NXY = JPT1-IPT1+1
      IDT = 9999
      IF(JPT1.LE.IXY1) GO TO 187
      NXY = MOD(JPT1,IXY1)
      IDT = JDST(IX2(KK),IY2(KK),IX1,IY1,NXY,IDIR)*(-1)
      NXY = JPT1-IPT1+2-NXY
      IX1(IXY1+1) = IX1
      IY1(IXY1+1) = IY1
```

```
  187 ID=JDIST(IY2(KK),IY2(KK),IX1(IPT1),IY1(IPT1),NXY,IDIR)*(-1)
      IF(IABS(ID).GT.IABS(IDT))ID=IDT
      IF(IABS(ID).EQ.9999) GO TO 190
      IF(IABS(ID).GT.IABS(JDIST)) IDIST=ID
      KFTT = 1
  190 CONTINUE
      IF(KFTT.EQ.0) IDIST = 9999
      IF(ISSW(14).LT.0) WRITE(6,1001) IDIST
  195 IF(IDIST.EQ.0) GO TO 260
      IF(IABS(IDIST).EQ.9999) GO TO 250
      IF(LABR.EQ.2) GO TO 940
      GO TO (205,305),IDIR
  205 IF(IDIST.GT.0) IXN = 1
      IF(IDIST.LT.0) IXP = 1
      IF(IDIST.GT.0.AND.IXP.EQ.0) GO TO 220
      IF(IDIST.LT.0.AND.IXN.EQ.0) GO TO 220
      GO TO 250
  220 IXP = 0
      IXN = 0
      GO TO 460
  250 IF(IHV.EQ.2) GO TO 920
C.....THY OTHER DIRECTION
      IDIR = 3-IDIR
      IHV = 2
      IDIST = 0
      GO TO 170
C.....CHECK REST OF POINTS IN AREA FOR INTERSECTIONS
  260 IF(JPT2.GE.JX2+NXY2-1) GO TO 25
      NXY2 = NXY2-(JPT2-JX2)
      JX2 = INDEX(JPT2,IXY2)
      GO TO 115
  305 IF(IDIST.GT.0) IYN = 1
      IF(IDIST.LT.0) IYP = 1
      IF(IDIST.GT.0.AND.IYP.EQ.0) GO TO 320
      IF(IDIST.LT.0.AND.IYN.EQ.0) GO TO 320
      GO TO 250
  320 IXP = 0
      IXN = 0
      GO TO 470
  400 CONTINUE
      GO TO 930
  900 JFLG = 0
      IF(ISSW(13).LT.0) WRITE(6,1003) ICM,ICI,ICD
 1003 FORMAT ("MIN,MAX ",I8," INTER. ",I8," DIST. ",I8)
      CALL DOSC(0)
      CALL EXEC (6,MR)
      CALL MARK
      CALL MRKH
      CALL MRKE
C.....DISPLAY "PIECE TOO LARGE"
  910 CALL MESAG(0,ITEX3,ITMP)
      LABR(3) = -1
      GO TO 990
C.....OVERLAP EXISTS
  920 IF(MAMM.LT.0) GO TO 921
C.....TURN ON "OVERLAP" AND DELAY
      CALL LIGHT(3,1)
      DO 9210 I=1,20000
      ITMP = ITMP+1-1
 9210 CONTINUE
  921 LABR(3) = -1
      GO TO 990
C.....NO OVERLAP, RETURN
  930 LABR(2) = 0
  935 LABR(3) = 0
      GO TO 990
```

```
C..........RETURN PARAMETERS FOR PACK
  940 LABR(2) = IDIST
      LABR(3) = IDIR
      GO TO 960
  950 GO TO (951,952,953,954),IDIR
  951 LABR(2) = MXN-IDUM(JXMN)
      GO TO 955
  952 LABR(2) = MXP-IDUM(JXMX)
      GO TO 955
  953 LABR(2) = MYN-IDUM(JYMN)
      GO TO 955
  954 LABR(2) = MYP-IDUM(JYMX)
  955 LABR(3) = IDIR
      GO TO 991
 1001 FORMAT (6I8)
 1002 FORMAT (6F10.4)
      END
      END$
FTN4
      PROGRAM MRKR,5
C..........PROGRAM TO PACK MARKER TIGHT TO THE LEFT AND DOWN,
C..........INDICATING TROUBLE SPOTS FOR THE OPERATOR TO CORRECT.
C
C
C***************DUMMY DIMENSION FOR LOADER ***************
      DIMENSION IDUMM(1)
      COMMON NOUT,LABR(3),LABR(3),NOSTY,INSTY(31),JBS(15),NAMST(3),
     1 MAUP(5),NSCMD,JBM(20),NAMMK(3),JBG(4),NAMGD(3),JBP(10),
     2 IDMK(5),IDOP,NPEC,NSMD,FACTR,NSIZ,ISIZ(42),WIDTH,EFF,SX,SY,KSTL,
     3 YAPOS,NAP,KINT,KFLG,KINC,KSIZ,KXMN,KYMN,KANG,KXMX,KYMX,KOPT,
     4 KAL,KADP,KDF,CSMX,MKRD,CSMN,NRGE,IMGP,KREL,ITYPE,
     5 NOIMG,IOFSZ,IMGY,IMGE,IMGTC,IMKD1,IMKD2,IMENU,IOXP,IOYP,
     6 IXC,IYC,IXL,IXR,IYU,IYL,CS,MDELX,MDELY,JFLG,MLW,MUP,ISLX,ISLY,
     7 TOTAL,MYL,XPC,YPC,IMGLP,IMESG,IEDG,ISOFF,SOFF,KSLH,
     8 IMGPS,MAXOF,KSLV,KTL,IOPTN,IAREA,ISIZE,KCSR,KYES,KPSIZ,KPLL,
     9 IMGIN,ICPEC,IT3,IT4,IT5,IT6,IT7,IT8,IT9,IT10,
     A IOFSET,KPLH,IPLP,ICUST,IMGPK,
     B T1,T2,T3,T4,T5,T6,T7,T8,T9,T10,
     C JSTY,JFG,JNG,JSHD,JRT,
     D ILPO,IROF,IRD,IOV,INR,IBLFT,ITCW,ITCCW,
     E INCW,IPCCW,ISL,IRE,IFL,IPK,ICOM,
     F IDUM(1)
C************NOTE COMMON (CONT. CARD "D" IS NON-STANDARD********
      DIMENSION MK(3),IORDR(200),DLT(200),ITEX(11),ITEX1(11)
      DATA ITEX/2HOP,2HER,2HAT,2HOR,2H M,2HUS,2HT ,2HCO,2HRR,2HEC,
     12H10/
      DATA ITEX1/3*2H  ,2HCO,2HMP,2HLE,2HTE,2HD0/
      DATA MK/2HMR/
      DATA XLEFT/-32600.0/
      DATA LOUT/6/
      CALL LIGHT(2,1)
      MK(3) = 2H
C..........SET UP ORDER OF PIECES IN MARKER STARTING
C..........WITH THE LOWER-LEFT CORNER
      MYP=MKRD/2
      MYN=-(MKRD-MYP)
      MYP2=FLOAT(MYP)-.1*FLOAT(MYP)
C.....SORT PIECES STARTING AT THE LOWER LEFT
      IF(IMGP.NE.0) CALL ORGIN(IOXP,IOYP,0,IMGP)
      DO 10 II=1,NOIMG
   10 IORDR(II)=II
      YLOWR = MYN
      DO 20 II=1,NOIMG
      DLT(II)=9999999.0
      XDLT = FLOAT(IDUM(KXMN+II))-XLEFT
```

```
      YDLT = FLOAT(IDUM(KYMN+II))-YLOWR
      DLT(II)=SQRT(XDLT*XDLT+YDLT*YDLT)
   20 CONTINUE
      DO 30 I=1,NOIMG
      DLTMN=9999999.0
      DO 25 II=1,NOIMG
      IF(DLT(II).GE.DLTMN) GO TO 25
      JJ=II
      DLTMN=DLT(II)
   25 CONTINUE
      IORDR(I)=JJ
      DLT(JJ)=9999999.0
   30 CONTINUE
C.....STEP THROUGH PIECES IN ORDER
   35 DO 900   I=1,NOIMG
C.....SET SUBSCRIPTS
      IMGPX=IORDR(I)
      IF(IMGPX.LE.0) GO TO 900
C.....IGNOR SLAVE PIECES
      IF(JLDF(JBM(20),IDUM(KFLG+IMGPX)).EQ.1) GO TO 900
      JXMN=KXMN+IMGPX
      JXMX=KXMX+IMGPX
      JYMN=KYMN+IMGPX
      JYMX=KYMX+IMGPX
      JSTL=KSTL+IMGPX
C.....DUMP PIECES IN THE BOTTOM HALF DOWN AND VICE VERSA
      IUPFLG=0
      IF(IDUM(JYMX).GT.MYP2)IUPFLG=1
      IXN = 0
      IYP = 0
      IYN = 0
      IBL=0
      IBU=0
      IYNB=0
       ILNT=0
C.....IGNORE PIECES OFF THE MARKER
      IF(IDUM(JYMN).GT.MYP) GO TO 900
      IMGP=IMGPX
C.....FLAG INVALID DIRECTIONS
      IF(IDUM(JXMN).EQ.-32000) IXN = 1
      IF(IDUM(JYMX).EQ.MYP)IYP=1
      IF(IDUM(JYMN).EQ.MYN) IYN=1
      MDLX = IDUM(JXMX)-IDUM(JXMN)
      MDLY = IDUM(JYMX)-IDUM(JYMN)
C.....PLAID PIECES CAN NOT MOVE LEFT
      IF(ISLX.EQ.0.OR.IDUM(KPLL+IMGPX).EQ.9999) GO TO 35
      IXN = 1
   35 IF(ISLY.EQ.0) GO TO 40
C.....STRIPE PIECES CAN NOT MOVE UP OR DOWN UNLESS NOT IN A
C.....SPECIAL STRIPE GROUP
      IF(IDUM(JSTL).EQ.9999) GO TO 40
      IF(SY.EQ.99..AND.JLDF(JBM(13),IDUM(KFLG+NPEC+IMGPX)).EQ.0)GO TO 47
      IYNB = 1
      IYP =1
      IYN = 1
C.....CHECK OVERLAP
   40 ITYPE = IOV
      IFLAG = 1
100   CALL VISIB(IMGP,IORG,IDXP,IDYP)
      IF(IORG.EQ.0)GO TO 101
      CALL ORGIN(IDXP,IDYP,IORG,IMGP)
C.....MOVE CURSOR TO PIECE
      ITX1 = IRND(IDXP*CS)
      ITY1=IRND(IDYP*CS)
      CALL CROSS(ITX1,ITY1,KTC,IDUM)
```

```
101   CONTINUE
      CALL BUMP(2,IDIST,IDIR)
109   CONTINUE
C.....CHECK OR PACK?
      IF(JFLG.EQ.5) GO TO 1090
C.....CHECK
      IF(IDIR.EQ.0) GO TO 900
      GO TO 200
C.....PACK
1090  IF(IDIR.EQ.0) GO TO 500
      IF(IDIR.EQ.-1) GO TO 200
C.....OVERLAP PRESENT
      IF(IDIR.EQ.1.AND.IDIST.GT.0) GO TO 200
C.....CAN CORRECT OVERLAP BY MOVE LEFT, UP OR DOWN
      GO TO (110,120),IDIR
C.....MOVE LEFT, CHECK FOR MARKER EDGE
  110 IF(IXN.EQ.1) GO TO 200
      IF(IXL.EQ.1)GO TO 200
      IXL=1
      IF(IDIST.GT.0)IDIST=0
      DXN = FLOAT(IDUM(JXMN))+FLOAT(IDIST)
      IF(DXN.LT.XLEFT) GO TO 200
      IDUM(JXMN) = DXN
      IDUM(JXMX) = IDUM(JXMN)+MDLX
      IF(DXN.EQ.XLEFT) IXN = 1
      GO TO 130
C.....MOVE UP OR DOWN, CHECK FOR MARKER EDGE
  120 IF(IYP.EQ.1.AND.IDIST.GT.0) GO TO 200
      IF(IYN.EQ.1.AND.IDIST.LT.0) GO TO 200
      IF(IYU.EQ.1)GO TO 200
      IYU=1
      NYP = IDUM(JYMX)+IDIST
      NYN = IDUM(JYMN)+IDIST
      IF(NYP.GT.MYP) GO TO 200
      IF(NYN.LT.MYN) GO TO 200
      IDUM(JYMN) = NYN
      IDUM(JYMX) = NYP
      IYP = 0
      IYN = 0
      IF(NYP.EQ.MYP) IYP=1
      IF(NYN.EQ.MYN) IYN=1
C.....MOVE PIECE IN DISPLAY
  130 CONTINUE
      CALL VISIB(IMGP,IORG,IDXP,IDYP)
      IF(IORG.EQ.0) GO TO 140
      CALL ORGIN(IDXP,IDYP,IORG,IMGP)
      CALL MODI (1,IDXP,IDYP,IORG,IMGP)
      IF(ISSW(11).LT.0) GO TO 901
      IF(ICNT.EQ.2)GO TO 700
      IF(LOST+MOST.EQ.0) GO TO 700
      GO TO (101,510,520),IFLAG
C.....PIECE IS OFF THE SCREEN, DELETE FROM THE DISPLAY.
  140 III = IABS(IDUM(KADP+IMGP))
      CALL LITE(0,III)
      GO TO (101,510,520),IFLAG
C.....OPERATOR MUST CORRECT OVERLAP AREA
  200 CALL MESAG(3,ITEX,ITMP)
  210 CALL VISIB(IMGP,IORG,IDXP,IDYP)
      IF(IORG.EQ.0) GO TO 220
C.....CONFLICTING PIECE IS PICKED TO SHOW AREA
      CALL ORGIN(IDXP,IDYP,IORG,IMGP)
      MK(2) = 2HKE
      JFLG = 0
      MOELX=MDLX
      MOELY=MDLY
```

```
      CALL ORGIN(IDXP,IDYP,0,IMGP)
      CALL UDSC(3)
      CALL LIGHT(2,0)
      CALL EXEC (8,MK)
      CALL MRKE
      CALL MARK
C.....PIECE NOT VISIBLE.  ROLL SO THAT IT IS VISIBLE
  220 IXC = IDUM(JXMX)
      IK(2)=2MKI
      CALL UDSC(0)
      CALL EXEC(3,MK)
      CALL MRKI
      GO TO 210
C.....NO OVERLAP SO BUMP LEFT AND DOWN
  500 MDST = -9999
  510 ITYPE = IBLFT
      IF(IXN.EQ.1)GO TO 545
      CALL BUMP(2,LDST,IDIR)
      IF(LDST.GT.0)LDST=0
      IF(LDST.LT.-400)LDST=-400
      IF(IDIR.EQ.-1) GO TO 200
      IF(LDST+MDST.EQ.0) GO TO 700
      IF(LDST.EQ.0) GO TO 550
      IF(IABS(LDST).EQ.1)GO TO 545
      DXN=FLOAT(IDUM(JXMN))+FLOAT(LDST)
      IF(DXN.LE.XLEFT) LDST = 0
      IDUM(JXMN) = MAX1(DXN,XLEFT)
      IDUM(JXMX) = IDUM(JXMN)+MDLX
      IF(IDUM(JXMN).EQ.-32800)IXN=1
      GO TO 550
  520 IF(IYNB.EQ.1) GO TO 700
  525 ITYPE = IBD
      IF(IYNB.EQ.1)GO TO 550
      IF(IUPFLG.EQ.1) ITYPE=IBUP
      CALL BUMP(2,MDST,IDIR)
      ICNT=2
      IF(MDST.GT.0.AND.ITYPE.EQ.IBD) MDST = 0
      IF(MDST.LT.0.AND.ITYPE.EQ.IBUP) MDST = 0
      IF(ITYPE.EQ.IBD.AND.MDST.LT.-400) MDST=-400
      IF(ITYPE.EQ.IBUP.AND.MDST.GT.400)MDST=400
      IF(IDIR.EQ.-1) GO TO 200
      IF(MDST+LDST.EQ.0) GO TO 700
      IF(MDST.EQ.0) GO TO 550
      IF(IABS(MDST).EQ.1)GO TO 546
      NYP = IDUM(JYMX)+MDST
      NYN = IDUM(JYMN)+MDST
      IF(NYP.LT.MYP) GO TO 530
      IYNB=1
      MDST = 0
      IDUM(JYMX) = MYP
      IDUM(JYMN)=MYP-MDLY
      GO TO 550
  530 IF(NYN.GT.MYN)GO TO 540
      IYNB=1
      MDST = 0
      IDUM(JYMN)=MYN
      IDUM(JYMX) =MYN+MDLY
      GO TO 550
  540 IDUM(JYMN) = NYN
      IDUM(JYMX) = NYP
      GO TO 550
  545 LDST=0
      GO TO 550
  546 MDST=0
  550 CONTINUE
```

```
      IFLAG=2
      IF(ITYPE.EQ.IHLFT) IFLAG = 3
      GO TO 130
C.....PIECE IS FIXED INTO PLACE
  700 IORDR(I) = -IORDR(I)
  900 CONTINUE
901   CONTINUE
      CALL VISIB(IMGP,IORG,IDXP,IDYP)
      IF(IORG.NE.0) CALL ORGIN(IDXP,IDYP,0,IMGP)
C.....PACKING COMPLETED, RETURN TO OPERATOR
      ITEX1   = 2HCH
      ITEX1(2) = 2HEC
      ITEX1(3) = 2HK
      IF(JFLG.EQ.6) GO TO 9010
      ITEX1   = 2H P
      ITEX1(2) = 2HAC
      ITEX1(3) = 2HK
 9010 CALL MESAG(0,ITEX1,ITMP)
      MK(2) = 2HKE
      JFLG = 0
      MDELX=MDLX
      MDELY=MDLY
C.....SEND FLAG TO MRKE TO RELEASE PIECE
      IMGP = 9999
      CALL DOSC(0)
      CALL LIGHT(2,0)
      CALL EXEC (8,MK)
      END
      ENDS
FTN4
      PROGRAM MRKI,5
C
C
C
C
C
C
      COMMON MOUT,LABF(3),LABR(3),NOSTY,INSTY(31),JBS(15),NAMST(3),
     1 MKUP(5),NSCHU,JBM(20),NAMMK(3),JBG(4),NAMGO(3),JBP(10),
     2 IDMK(5),IDOP,NPEC,NSHD,FACTR,NSIZ,ISIZ(42),WIDTH,EFF,SX,SY,KSTL,
     3 YARDS,MAP,KINT,KFLG,KINC,KSIZ,KXMN,KYMN,KANG,KXMX,KYMX,KOPT,
     4 KAD,KADP,KOF,CSMX,MKWD,CSMN,NRGE,IMGP,KREL,ITYPE,
     5 NOIMG,IOFSZ,IMGY,IMGE,IMGTC,IMKD1,IMKD2,IMEND,IDXP,IDYP,
     6 IXC,IYC,IXL,IXR,IYU,IYL,CS,MDELX,MDELY,JFLG,MLW,MUP,ISLX,ISLY,
     7 TOTAL,MYL,XPC,YPC,IMGLP,IMESG,IEOG,ISOFF,SOFF,KSLH,
     8 IMGPS,MAXOF,KSLV,KTC,IOPTN,IAREA,ISIZE,KCSR,KYES,KPSIZ,KPLL,
     9 IT1,IT2,IT3,IT4,IT5,IT6,IT7,IT8,IT9,IT10,
     A IOFSET,KPLH,IPLP,ICUST,IMGPK,
     B T1,T2,T3,T4,T5,T6,T7,T8,T9,T10,
     C JSTY,JFG,JNG,JSHD,JRT,
     D ILPn,IBU,IBD,IDV,IBR,IBL,ITCW,ITCCW,
     E IRCW,IRCCW,ISL,IRE,IFL,IPK,ICOM,
     F IDUM(1950)
      DIMENSION MK(3)
      MK(1)=2HMR
      MK(3)=2H
      CALL MINUO
      MK(2)=2HKM
      LABR(1)=999
      CALL EXEC(8,MK)
      CALL MRKM
      CALL MARK
      END
      ENDS
```

```
FTN4
      SUBROUTINE NAREA(MSCP,ARA,PRM)
C..........
C..........MARGARET ROWAN
C..........PROGRAM: MARK
C..........VERSION 2.0
C..........ROUTINE TO CALCULATE AREA OF GRADED PIECE
C..........STORED AT SECTOR "MSCP" IN FILE "GRADE".
      DIMENSION IBUF(128)
      COMMON MOUT,LABF(3),LABR(3),NOSTY,INSTY(31),JBS(15),NAMST(3),
     1 MKUP(5),NSCMU,JUM(20),NAMMK(3),JBG(4),NAMGD(3),JBP(10),
     2 IDMK(5),IDDP,NPEC,NSHD,FACTR,NSIZ,ISIZ(42),WIDTH,EFF,SX,SY,KSTL,
     3 YARDS,MAP,KINT,KFLG,KINC,KSIZ,KXMN,KYMN,KANG,KXMX,KYMX,KDPT,
     4 KAD,KADP,KDF,CSMX,MKND,LSMN,NRGE,IMGP,KREL,ITYPE,
     5 NOIMG,IUFSZ,INGY,IMGE,INGIC,IMK01,IMK02,IMEND,IDXP,IDYP,
     6 IXC,IYC,IXL,IXR,IYU,IYL,CS,MDELX,MDELY,JFLG,MLW,MUP,ISLX,ISLY,
     7 TOTAL,NYL,XPC,YPC,IMGLP,INESG,IEDG,ISOFF,SOFF,KSLH,
     8 IMGPS,MAXDF,KSLV,KTC,IDFTN,IAREA,ISIZL,KCSR,KYES,KPSIZ,KPLL,
     9 IT1,IT2,IT3,IT4,IT5,IT6,IT7,IT8,IT9,IT10,
     A IUFSET,KPLH,IFLP,ICUST,IMGPK,
     B T1,T2,T3,T4,T5,T6,T7,T8,T9,T10,
     C JSTY,JFG,JNG,JSHD,JRT,
     D ILPB,IBO,IBD,IOV,IBR,IBL,ITCW,ITCCW,
     E IRCW,IPCCW,ISL,IRE,IFL,IPK,ICOM,
     F IUUM(1)
       NSEC = MSCP
       ARA = 0.0
       PRM = 0.0
       IFTT = 0
C..........BUFFER IN PIECE DATA
   10  CALL EXEC (14,05,IBUF,128,NAMGD,NSEC)
       IF(NSEC.NE.NSCP) GO TO 20
C..........SET UP AMOUNT OF DATA AND MIRROR FLAG
       LSEC = JLDF(JBP(5),IBUF)+NSEC
       IMIR = JLDF(JBP(3),IBUF)
   20  DO 100 I1=3,126,3
       ITMP = JLDF(JBP(9),IBUF(I1))
       IF(ITMP.EQ.10) GO TO 200
       IF(ITMP.GT.1.AND.ITMP.NE.8.AND.ITMP.NE.11) GO TO 100
       XF = FLOAT(IBUF(I1+1))*XPC/1000.0
       YF = FLOAT(IBUF(I1+2))*YPC/1000.0
       IF(IFTT.EQ.0) GO TO 30
C..........COMPUTE AREA AND PERIMETER
       ARA = ARA+(XS-XF)*(YS+YF)/2.0
       PRM = PRM+SQRT((XF-XS)**2+(YF-YS)**2)
   30  XS = XF
       YS = YF
       IFTT = 1
  100  CONTINUE
       NSEC = NSEC+1
       IF(NSEC.LT.LSEC) GO TO 10
       CALL ERR(12,1)
       RETURN
  200  IF(IMIR.NE.1) RETURN
C..........DOUBLE IF MIRRORED
       ARA = ARA*2.0
       PRM = PRM*2.0
       RETURN
       END
       END$
FTN4
      SUBROUTINE NXYMN(MSEC,XMN,YMN,NANG,LANG,JFP)
C..........DEC. 10, 1972
C..........MARGARET ROWAN
C..........PROGRAM: MARK
```

```
C..........VERSION 2.X
C..........ROUTINE TO COMPUTE NEW MINIMUM VALUES FOR PIECE AT
C..........AN ANGLE.
      DIMENSION IBUF(128)
      COMMON MOUT,LABF(3),LABR(3),NOSTY,INSTY(31),JBS(15),NAMST(3),
     1 NKUP(5),NSCHU,JBM(21),NAMMK(3),JBG(4),NAMGD(3),JBP(10),
     2 IDMK(5),IDDP,NPEC,NSHD,FACTR,NSIZ,ISIZ(42),WIDTH,EFF,SX,SY,KSTL,
     3 YARDS,NAP,KINT,KFLG,KINC,KSIZ,KXMN,KYMN,KANG,KXMX,KYMX,KOPT,
     4 KAD,KADP,KDP,CSMX,MKWD,CSMN,NAGE,IMGP,KREL,ITYPE,
     5 NOIMG,IOFSZ,IMGY,IMGE,IMGIC,IMKD1,IMKD2,IMENU,IDXP,IDYP,
     6 IXC,IYC,IXL,IXR,IYU,IYL,CS,MDELX,MDELY,JFLG,MLW,MUP,ISLX,ISLY,
     7 TOTAL,MYL,XPC,YPC,IMGLP,IMESG,IBOG,ISOFF,SOFF,KSLH,
     8 IMGP3,MAXOF,KSLV,KTC,IOPTN,IAREA,ISIZE,KCSR,KYES,KPSIZ,KPLL,
     9 IT1,IT2,IT3,IT4,IT5,IT6,IT7,IT8,IT9,IT10,
     A IOFSET,KPLH,IFLP,ICUST,IMGPK,
     B T1,T2,T3,T4,T5,T6,T7,T8,T9,T10,
     C JSTY,JFG,JNG,JSHD,JRT,
     D ILPR,IRU,IRD,IOV,IBR,IBL,ITCW,ITCCW,
     E IRCW,IRCCW,ISL,IRE,IFL,IPK,ICOM,
     F IOUM(1)
       XMIN = 0.0
       YMIN = 0.0
       XMAX = 0.0
       YMAX = 0.0
       MINX = 0
       MAXX = 0
       MINY = 0
       MAXY = 0
       IFP=0
C..........SET FLIP FLAG FOR FLIP BY OPERATOR
       IF(IABS(NANG).GE.3600) IFP=1
C..........ANGLE OF ROTATION FROM OPERATOR
       MANG = MOD(NANG,3600)
       ANG = MANG*.00174533
       S = SIN(ANG)
       C = COS(ANG)
       CALL UDSC(0)
       ISEC = MSEC
    10 CALL EXEC (14,66,IBUF,128,NAMGD,ISEC)
C..........SET MIRROR FLAG
       IMIRO = JLOF(JBP(3),IBUF   )
C..........MIRRORED & FLIPPED = MIRRORED & NOT FLIPPED
       IF(IMIRO.EQ.1.AND.JFP.EQ.1) JFP = 0
       LSEC = MSEC+JLOF(JBP(5),IBUF(1))
C..........GRAIN ANGLE
       ANGG = LANG*.00174533
       SG = SIN(ANGG)
       CG = COS(ANGG)
       IF(NKUP.EQ.2HFR) GO TO 20
C..........SET UP SCALE TO MARKER UNITS FOR OLD MARKER
       DIM = FACTR*4.96.0
       CALL SCALE (0.0,DIM,DIM,1)
    20 DO 40 I=3,120,3
       ITYP = JLOF(JBP(9),IBUF(I))
       IF(ITYP.EQ.10) GO TO 120
       IF(ITYP.GT.1.AND.ITYP.NE.8.AND.ITYP.NE.11) GO TO 40
C..........ONLY USE OUTLINE POINTS
       X = IBUF(I+1)*.001
       Y = IBUF(I+2)*.001
       IF(IMIRO.EQ.0) GO TO 30
       X1 = X
       Y1 = -Y
       CALL XFORM(X1,Y1,X1,Y1,SG,CG)
       CALL XFORM(X1,Y1,X1,Y1,S,C)
       IF(NKUP.EQ.2HFR) GO TO 25
```

```
C..........OLD MARKER
      CALL SCALE (IX1,IY1,X1,Y1,2)
      MINX = MINO(MINX,IX1)
      MAXX = MAXO(MAXX,IX1)
      MINY = MINO(MINY,IY1)
      MAXY = MAXO(MAXY,IY1)
      GO TO 30
C..........FRESH START
   25 XMIN = AMIN1(XMIN,X1)
      XMAX = AMAX1(XMAX,X1)
      YMIN = AMIN1(YMIN,Y1)
      YMAX = AMAX1(YMAX,Y1)
   30 CALL XFORM(X,Y,X,Y,SG,CG)
      IF(JFP.EQ.1) Y = -Y
      CALL XFORM(X,Y,X,Y,S,C)
      IF(IFP.EQ.1) Y = -Y
      IF(MKUP.EQ.2HER) GO TO 35
C..........OLD MARKER
      CALL SCALE (IX,IY,X,Y,2)
      MINX = MINO(MINX,IX)
      MAXX = MAXO(MAXX,IX)
      MINY = MINO(MINY,IY)
      MAXY = MAXO(MAXY,IY)
      GO TO 40
C..........FRESH START
   35 XMIN = AMIN1(XMIN,X)
      XMAX = AMAX1(XMAX,X)
      YMIN = AMIN1(YMIN,Y)
      YMAX = AMAX1(YMAX,Y)
   40 CONTINUE
      IF(ISEC.EQ.LSEC) GO TO 100
      ISEC = ISEC+1
      CALL EXEC (14,66,IBUF,128,NAMGD,ISEC)
      GO TO 24
  100 IF(MKUP.EQ.2HER) GO TO 105
C..........OLD MARKER
      XMN = FLOAT(MAXX-MINX)
      YMN = FLOAT(MAXY-MINY)
      RETURN
C..........FRESH START
  105 XMN = (XMAX-XMIN)*1000.0
      YMN = (YMAX-YMIN)*1000.0
      RETURN
      END
      ENDS
FTN4
      SUBROUTINE ORGIN (IX,IY,JORG,IMGNO)
C..........ROUTINE TO COMPUTE MOVE FROM IMAGE TO PIECE
C..........ORIGIN.  MOVE IS STORED IN DISPLAY FILE (JORG>0)
C..........OR RETURNED IN IX,IY (JORG<0).  MIN&MAX VALUES
C..........ARE UPDATED (JORG=0).
C
C
      COMMON MOUT,LABF(3),LABR(3),NOSTY,INSTY(31),JBS(15),NAMST(3),
     1 MKUP(5),NSCMD,JBN(20),NAMMK(3),JBG(4),NAMGD(3),JBP(12),
     2 IOMK(5),IDDP,NPEC,NSHD,FACTR,NSIZ,ISIZ(42),WIDTH,EFF,SX,SY,KSTL,
     3 YAPOS,NAP,KINT,KFLG,K1NC,KSIZ,KXMN,KYMN,KANG,KXMX,KYMX,KOPT,
     4 KAD,KADP,KOP,CSMX,MKWD,CSMN,NAGE,IMGP,KREL,ITYPE,
     5 NOIMG,IOFSZ,IMGY,IMGE,IMGTC,IMKC1,IMKC2,IMENU,IDXP,IDYP,
     6 IXC,IYC,IXL,IXR,IYU,IYL,CS,MOELX,MOELY,JFLG,MLW,MUP,ISLX,ISLY,
     7 TOTAL,MYL,XPC,YPC,IMGLP,IMESG,IEOG,ISOFF,SOFF,KSLH,
     8 IMGPS,MAXOF,KSLV,KTC,IOPTN,IAREA,ISIZE,KCSR,KYES,KPSIZ,KPLL,
     9 IT1,IT2,IT3,IT4,IT5,IT6,IT7,IT8,IT9,IT10,
     A IOFSET,KPLN,IFLP,ICUST,IMGPK,
     B T1,T2,T3,T4,T5,T6,T7,T8,T9,T10,
```

```
      C  JSTY,JFG,JBG,JSMD,JRT,
      D  ILPR,IRD,IRU,IDV,IRR,IRL,ITCW,ITCCW,
      E  INCW,IRLCR,ISL,IRE,IFL,IPK,ICOM,
      F  IDUM(1)
         DATA IONE/1/
         IF(IMGNO.EQ.0) RETURN
         NN = 4
         IALPH=JLOF(JBM(15),IOPTN)
         IORG = IABS(JORG)
         KK = IMGNO
         KAD1 = KADR+KK
C.........SET UP POSSIBLE X&Y MOVES FOR ANY ORIGIN.
         IND=IABS(IDUM(KAD1))+NN
         NX=IDUM(IND)
         NY=IDUM(IND+1)
         IF(IALPH.EQ.0) GO TO 5
         MINX=NX/16
         MINY=(NY-1)/16
         GO TO 10
       5 MINY=(NX-1)/16
         MINY=(NY-16R)/16
      10 NX=MINX
         NY = MINY
         KKXMX = KXMX+KK
         KKXMN = KXMN+KK
         KKYMX = KYMX+KK
         KKYMN = KYMN+KK
         MXMX = IDUM(KKXMX)
         MXMN = IDUM(KKXMN)
         MYMX = IDUM(KKYMX)
         MYMN = IDUM(KKYMN)
         MX = MXMX-MXMN
         MY = MYMX-MYMN
         M1 = 0
         IF(ISLX*ISLY.EQ.0.OR.JLOF(JBM(15),IDUM(KFLG+IMGNO)).EQ.0)
        1   GO TO 20
         M1   = 1
         MX = MX-ISLX
         IF(SY.EQ.99.0) GO TO 20
         MY = MY-ISLY
      20 IF(MINX.LT.0) MINX = MX+MINX
         IF(MINY.LT.0) MINY = MY+MINY
         MAXX = MX-MINX
         MAXY = MY-MINY
         IF(IORG.EQ.0) GO TO 210
C.........SELECT MOVE FOR REQUESTED ORIGIN.
         IF(IORG.LT.10) GO TO 100
         IORG = IORG-10
         ITMP = MINY
         MINY = MAXY
         MAXY = ITMP
     100 GO TO (110,120,130,140),IORG
     110 NX = MINX
     115 NY = MINY
         GO TO 200
     120 NX = -MAXX
         GO TO 115
     130 NX = -MAXX
     135 NY = -MAXY
         GO TO 200
     140 NX = MINX
         GO TO 135
     200 CONTINUE
     201 IF(JORG.GT.0) GO TO 202
C.........RETURN MOVE TO CALLING PROGRAM.
```

```
      IX = NX
      IY = NY
      RETURN
C.....PLACE MOVE IN DISPLAY FILE.
  202 IF(IALPH.EQ.0) GO TO 203
      IDUM(IND)=NX*16
      IDUM(IND+1)=NY*16+1
      GO TO 204
  203 IDUM(IND)=NX*16+1
      IDUM(IND+1) = NY*16+1600
  204 GO TO (310,320,330,340),IORG
C.....ADJUST IX, IY IF NECESSARY
  310 IX = MXMN-IXC
  315 IY = MYMN-IYC
      RETURN
  320 IX = MXMX-IXC
      GO TO 315
  330 IX = MXMX-IXC
  335 IY = MYMX-IYC
      RETURN
  340 IX = MXMN-IXC
      GO TO 335
C.....UPDATE MIN & MAX VALUES.
  210 IF(N1.EQ.0)GO TO 360
      NX = NX+ISLX
      IF(SY.EQ.99.0) GO TO 360
      NY = MY+ISLY
  360 D = FLOAT(IXC)+FLOAT(IX)
      IF(NX.GT.0) D = D+FLOAT(MX)
      BOUND=32700.0
      IF(IYC+IY+MY.LE.MKWD/2) BOUND=32600.0
      IDUM(KKXMX)= MIN1(D,BOUND)
      D = FLOAT(IDUM(KKXMX))-FLOAT(MX)
      IDUM(KKXMN) = MAX1(D,-BOUND)
      IDUM(KKXMX)=IDUM(KKXMN)+MX
      N = IYC + IY
      IF(NY.LT.0) N = N-MY
      MKWD2 = -(MKWD)-MKWD/2)
      IF(N.LT.MKWD2) N=MKWD2
      IDUM(KKYMN) = N
      IDUM(KKYMX) = IDUM(KKYMN) + MY
      RETURN
      END
      ENDS**********************************************************

FTN4
      SUBROUTINE OUTLN
C.....ROUTINE TO GENERATE DISPLAY INSTRUCTIONS TO DRAW
C.....THE MARKER OUTLINE (KTR=1) OR THE STRIPE AND
C.....PLAID LINES (KTR=2).
C
C
      DIMENSION IHED (11)
      COMMON MOUT,LABF(3),LABR(3),NOSTY,INSTY(31),JBS(15),NAMST(3),
     1 MKUP(5),NSCNU,JBM(20),NAMMK(3),JBG(4),NAMGD(3),JBP(12),
     2 IOMK(5),IDDP,NPEC,NSHU,FACTR,NSIZ,ISIZ(42),WIDTH,EFF,SX,SY,KSTL,
     3 YARDS,NAP,KINT,KFLG,KINC,KSIZ,KXMN,KYMN,KANG,KXMX,KYMX,KDPT,
     4 KAD,KADP,KDF,CSMX,MKWD,CSMN,NNGE,IMGP,KREL,ITYPE,
     5 NUIMG,IDFSZ,IMGY,IMGE,IMGTC,IMK01,IMK02,IMENU,IDXP,IDYP,
     6 IXC,IYC,IXL,IXR,IYU,IYL,CS,MDELX,MDELY,JFLG,MLN,MUP,ISLX,ISLY,
     7 TOTAL,MYL,XPC,YPC,IMGLP,IMESG,IEDG,ISOFF,SOFF,KSLN,
     8 IMGPS,MAXOF,KSLV,KTC,IOPTN,IAREA,ISIZE,KCSR,KYES,KPSIZ,KPLL,
     9 IT1,IT2,IT3,IT4,IT5,IT6,IT7,IT8,IT9,IT10,
     A IOFSET,KPLN,IFLP,ICUST,IMGPK,
     B T1,T2,T3,T4,T5,T6,T7,T8,T9,T10,
     C JSTY,JFG,JNG,JSHD,JRT,
```

```
     D ILPB,IBD,IBD,IDV,IBR,IBL,ITCW,ITCCW,
     E IRCW,IRCCW,ISL,IRE,IFL,IPK,ICOM,
     F IDUM(1)
      DATA IFTT/1/,JFTT/1/
      DATA IHED /4H0022B,0,/7760B,0,1,400010B,0,1,400140B,1,1B0045/
C.....COMPUTE EDGES OF MARKER ON DISPLAY SCREEN
      M = MKWD/2
      MUP = IRND((M-IYC)*CS)
      MUP = MIN0(MUP,2047)
      MLW=IRND((-(MKWD-M)-IYC)*CS)
      MLW = MAX0(MLW,-2047)
      MLF = -2047
      IF(FLOAT(IXL).LE.-32600.0) MLF = IRND((-32600-IXC)*CS)
      MRG = 2047
      IF(FLOAT(IXR).GE.32600.0) MRG = IRND((32600-IXC)*CS)
      MGOAL = 9999
      IF(NRGE.LT.IXR.AND.NRGE.GT.IXL) MGOAL = IRND((NRGE-IXC)*CS)
      MEDG = 9999
      IF(IEDG.LT.IXR.AND.IEDG.GT.IXL) MEDG = IRND((IEDG-IXC)*CS)
C.....DRAW MARKER OUTLINE
      KAD1 = KADP+IMK01
      IF(IDUM(KAD1).NE.0) GO TO 4
      IFTT=0
      IDUM(KAD1)=KDP1
    4 IND=IABS(IDUM(KAD1))
      CALL LITE(0,KAD1)
      CALL STAT
      IF(IND+20.GT.IDFSZ) CALL ERR(4,0)
      IF(IFTT.EQ.1) GO TO 5
C.....SET UP HEADER
      DO 110 I=1,5
      L = IND+I-1
  110 IDUM(L) = IHED(I)
      IFTT = 1
    5 IND = IND+5
C.....DRAW USING ABSOLUTE DRAW INSTRUCTIONS.
      IDUM(IND) = 100048
      IDUM(IND+1) = MLF*16+9
      IDUM(IND+2) = MUP*16+10
      IDRAW = 0
      IF(MUP.LT.2047) IDRAW = -4
      IDUM(IND+3) = MRG*16+9+IDRAW
      IDRAW = 0
      IF(MRG.LT.2047) IDRAW = -4
      IDUM(IND+4) = MLW*16+10+IDRAW
      IDRAW = 0
      IF(MLW.GT.-2047) IDRAW = -4
      IDUM(IND+5) = MLF*16+9+IDRAW
      IDRAW = 0
      IF(MLF.GT.-2047) IDRAW = -4
      IDUM(IND+6) = MUP*16+10+IDRAW
      IDUM(IND+7) = MUP*16+14
      JND = IND+8
      IF(MGOAL.EQ.9999) GO TO 90
      IDUM(JND) = 100248
      IDUM(JND+1) = MGOAL*16+9
      IDUM(JND+2) = MLW*16+14
      GO TO 95
   90 IDUM(JND) = 0
      IDUM(JND+1) = 0
      IDUM(JND+2) = 0
   95 IF(MEDG.EQ.9999) GO TO 97
      IDUM(JND+3) = 100048
      IDUM(JND+4) = MEDG*16+9
      IDUM(JND+5) = MUP*16+10
```

```
      IDUM(JND+6) = MLW*16+14
      GO TO 99
   97 IDUM(JND+3) = 0
      IDUM(JND+4) = 0
      IDUM(JND+5) = 0
      IDUM(JND+6) = 0
   99 KOPT = MAX0(JND+6,KOPT)
C.....DRAW SHADE ZONE BOUNDARIES
      JND = JND+8
      IF(NSHD.LE.1) GO TO 100
      ISHD = IFND((MKWD/NSHD)*CS)
      MSHD = MUP-ISHD
      IDUM(JND) = 100248
  199 IF(JND+3.GT.IDFSZ) CALL ERR(4,0)
      IDUM(JND+1) = MLF*16+9
      IDUM(JND+2) = MSHD*16+10
      IDUM(JND+3) = MRG*16+5
      JND = JND+3
      MSHD = MSHD-ISHD
      IF(MSHD-5.GT.MLW) GO TO 199
      IF(JND+3.GT.IDFSZ) CALL ERR(4,0)
      IDUM(JND+1) = MRG*16+15B
      KOPT = MAX0(JND+3,KOPT)
C.....DRAW STRIPE AND PLAID LINES
  100 IF(ISLX.NE.0.OR.(ISLY.NE.0.AND.SY.NE.99.0)) GO TO 101
      CALL LITE(1,KAD1)
      IDUM(KADP+IMK02) = 100000B
      IDUM(KOPT) = 1000005
      RETURN
  101 KAD1 = KADP+IMK02
      JFTT = 0
      IF(IDUM(KAD1).LT.0) JFTT = 1
      IOFPT = IABS(IDUM(KAD1))
      IF(IOFPT.NE.0) GO TO 105
      JFTT = -1
      IDUM(KAD1) = KOPT
      IOFPT = KOPT
  105 JFTT = JFTT+1
      CALL LITE(0,KAD1)
      CALL STAT
      IF(JFTT.GE.1) GO TO 9
      IF(IOFPT+11.GT.IOFSZ) CALL ERR(4,0)
C.....SET UP HEADER
      DO 120 I=1,11
      L = IOFPT+I-1
  120 IDUM(L) = IHED(I)
C.....TURN OFF FIRST TIME DRAWN
      IDUM(IOFPT) = 1400226
    9 KOPT = IOFPT+11
      MMLW = MLW*16+10
      MMUP = MUP*16+5
      MMRG = MRG*16+1
      MMLF = MLF*16+5
      IF(ISLX.EQ.0) GO TO 50
C.....PLAID LINES
      DEL = FLOAT(IXL) +32000.0
      IF(DEL.LT.0.0) DEL = 0.0
      ASLX = FLOAT(ISLX)
      IX1 = ((FLOAT(IFIX(DEL/ASLX)+1)*ASLX-32000.0)-FLOAT(IXC))
      GO TO 20
   10 IF(KOPT+3.GT.IOFSZ) CALL ERR(4,0)
      IDUM(KOPT) = IRND(IX1*CS)*16+1
      IDUM(KOPT+1) = MMLW
      IDUM(KOPT+2) = MMUP
      KOPT = KOPT+3
```

```
      IX1 = IX1+ISLX
   20 IF(IRND(IX1*CS).GT.MRG) GO TO 50
      GO TO 10
C.....STRIPE LINES
   50 IF(ISLY.EQ.P.OR.SY.EQ.99.0) GO TO 80
      IY1 = MKWD/2-IYC-ISOFF
   60 IF(IY1.GT.(MKWD/2-IYC)) GO TO 70
      IF(KOPT+3.GT.IOFSZ) CALL ERR(4,0)
      IOUM(KOPT) = MMRG
      IOUM(KOPT+1) = IRND(IY1*CS)*16+10
      IOUM(KOPT+2) = MMLF
      KOPT = KOPT+3
   70 IY1 = IY1-ISLY
      IF(IY1.GE.-(MKWD-MKWD/2)-IYC) GO TO 60
   80 IF(KOPT.GT.IOFPT+11) GO TO 85
      IOUM(KOPT) = 10
      KOPT = KOPT+1
   85 IOUM(KOPT-1) = IOUM(KOPT-1)+8
      IOUM(KOPT) = 10000008
      KOPT = KOPT+1
      IF(JPTT.NE.1) GO TO 88
      CALL LITE(1,KAD1)
   88 KAD1 = KADP+IMK01
      CALL LITE(1,KAD1)
      RETURN
      END
      ENDS
FTN4
      PROGRAM PGRAM
C
C     VERSION 2.1
C     EDITED 20 FEB 73
C     RONALD GORDON
C
      DIMENSION IWORK(1)
      DIMENSION IBUFR(126)
      DIMENSION NAMF(5), NAME3(3), NAME6(6)
      INTEGER USECT
      INTEGER OPTNS(3)
      INTEGER PLATNO
      INTEGER ROSTER(5)
      INTEGER PASS, OPNTR, FLAG3
      INTEGER DONE, STYFLG, BUFSIZ
      INTEGER MARKR(3), PIECE(3), STYLE(3), TPECS(3), TPECO(3),
     +        RULES(3), GRADO(3), GRADE(3), CHEK(3)
      INTEGER PSORT(3), PGRAD(3), WPROD(5), KSTYL(3)
      LOGICAL DEBUG
      LOGICAL MASHD
      LOGICAL OLDON, GOPIEC, GOMARK
      LOGICAL FLAG1, FLAG2, FLAG4, FLAG5
C
C     COMMON AREA COMMUNICATES WITH SPOOL
C
      COMMON DEBUG
      EQUIVALENCE (ROSTER(1),PASS),
     +            (ROSTER(2),OPNTR),
     +            (ROSTER(3),FLAG1),
     +            (ROSTER(4),FLAG2),
     +            (ROSTER(5),FLAG3)
      DATA MARKR /2HMA,2HRK,1HR/,
     +     PIECE /2HPI,2HEC,1HE/,
     +     STYLE /2HST,2HYL,1HE/,
     +     TPECS /2HTP,2HEC,1HS/,
     +     RULES /2HRU,2HLE,1HS/,
     +     TPECO /2HTP,2HEC,1HO/,
```

```
     +      GRADU /2HGR,2HAD,1HU/,
     +      GRADE /2HGR,2HAD,1HE/,
     +      CHEK  /2HCH,2HEK,1H /
      DATA OPTNS /2HOP,2HTN,1HS/
      DATA PSORT /2HPS,2HOR,1HT/,
     +      PGRAD /2HPG,2HRA,1HD/,
     +      WPROD /2HWP,2HRO,1HD/,
     +      WSTYL /2HWS,2HTY,1HL/
      DATA NULL /037777B/, DONE /037777B/, STYFLG /177777B/
      DATA ISRCH1 /10001B/
      DATA IGRAB1 /34001B/
      DATA ISRCH2 /14002B/, ISTOR2 /04002B/, ILOOK2 /1402B/,
     +      IREPL2 /44002B/, IPURG2 /24002B/
      DATA IGRAB2 /34402B/
      DATA ISRCH3 /10003B/
      DATA IPURG5 /10005B/, ILOOK5 /1405B/
      DATA IGETS6 /30006B/
      DATA OLDON,GDPIEC,GDMARK /3*.FALSE./
      DATA PLATNO /1/
C
C     GET INPUT PARAMETERS
C
      CALL RMPAR (ROSTER)
C
C     GET OPTIONS FROM FILE
C
      CALL DDSC (0)
      CALL EXEC (14,1002B,IBUFR,4,OPTNS,0)
      DEBUG=JLOF(JFLD(1,1),IBUFR(4)) .EQ. 1
      DEBUG = DEBUG .OR. ISSW(13).LT.0
      CALL ALLOC (IWORK(0),BUFSIZ)
      IF (DEBUG) WRITE (1,1001) PASS,OPNTR,FLAG1,FLAG2,FLAG3,
     +      BUFSIZ,IBUFR(-2)
      IPNTR=0
      IF (PASS.GT.1 .AND. PASS.LE.PLATNO+1) GO TO 640
C                                     FLAG1 TRUE WHEN TPECS DONE
      FLAG1=.FALSE.
C                                     FLAG2 TRUE WHEN GRADE DONE
      FLAG2=.FALSE.
C                                     FLAG3 0 NO STYLES
C                                           1 LOAD WSTYL
C                                          -1 WAIT FOR NEXT PLATTER
      FLAG3=0
      OPNTR=0
      IF (PASS.EQ.99) FLAG2=.TRUE.
      PASS=1
C
C                                     QUEUE TYPE
      MASK1=JFLD(14,4)
C                                     NSIZES IN RULES
      MASK2=JFLD(9,10)
C                                     RULE FIELD IN STYLE
      MASK3 = JFLD(6,7)
C                                     NSECTS IN DATA FILE HEADER
      MASK4=JFLD(3,4)
C                                     RULE FIELD IN NESTED PLOT QUEUE
      MASK5=JFLD(13,6)
C                                     RULE IN MARKER
      MASK6 = JFLD(9,7)
C                                     PLATER NUMBER FOR PIECES
      MASK7=JFLD(14,2)
C                                      OLD MARKER QUEUE BIT
      MASK8=JFLD(13,1)
C                                      ;SECTORS, ;PIECES IN MARKR
```

```
      MASK9=JFLD(5,6)
      MASK10=JFLD(15,10)
C                                     RULE TABLE NUMBR IN PIECE
      MASK11=JFLD(15,7)
C                                     RIGHT AND LEFT HALF WORD
      MASKR=JFLD(7,8)
      MASKL=JFLD(15,8)
C
C
C     TEST FOR NO GRADE OPTION
C
      IF (FLAG2) GO TO 640
      IF (DEBUG) WRITE (1,1011)
C
C
C     SCAN JOBQ ENTRIES
C
C     STEP THROUGH QUEUE.  LOOK FOR TYPES 4 THROUGH 8.
C     FLAG4 IS TRUE IF WE HAVEN'T CHECKED FOR PROPER DISK YET
C
      FLAG4=.TRUE.
C                                     FETCH SIZE OF QUEUE
      CALL RQST (2,LQ)
      LQ=LQ-2
      IF (LQ.LT.7) GO TO 640
      DO 555 IQ=7,LQ,2
C                                     GET TYPE FOR THIS ENTRY
      CALL RQST (IQ,IQWORD)
      ITYPE=JLOF(MASK1,IQWORD)
C                                     GET DATA VECTOR, CHECK IF PURGED
      CALL RQST (IQ+1,IQWORD)
      IF (IQWORD.LT.257) GO TO 555
C                                     IQWRD POINTS TO START OF DATA NOW
      ITYPE=ITYPE-2
      GO TO (555,150,250,350,450,550,555), ITYPE
C
C     PROCESS STYLE DEFINITION REQUEST                          QUEUE=4
C
  150 CONTINUE
      FLAG3=1
C                                     PICK UP NUMBER OF PIECES IN STYLE
      CALL RQST (IQWORD+6,ITEMP)
      NPIEC=JLOF(MASKR,ITEMP)
      IQWORD=IQWORD+7
C                                     BUILD STYLE RECORDS
      DO 170 I=1,NPIEC,1
      IF (IPNTR.EQ.BUFSIZ) CALL SPOOL (IWORK(1),IPNTR,DPNTR)
      DO 165 J=1,5,1
      CALL RQST (IQWORD,IWORK(IPNTR+J))
  165 IQWORD=IQWORD+1
      IWORK(IPNTR+6)=NULL
      IWORK(IPNTR+7)=STYFLG
      IWORK(IPNTR+8)=0
      IPNTR=IPNTR+8
  170 IQWORD=IQWORD+5
      GO TO 555
C
C     MARKER REQUEST ENTRY                                      QUEUE=5
C
  250 CONTINUE
C                                     GET NUMBER OF STYLES FOR THIS ONE
      CALL RQST (IQWORD+8,ITEMP)
      NSTYL=JLOF(MASKR,ITEMP)
      OLDQN=JLOF(MASK8,ITEMP).EQ.1
      IQWORD=IQWORD+13
```

```
  255 CONTINUE
      IF (.NOT.OLDUN.AND.NSTYL.EQ.0) GO TO 555
C                                     DO FOR ALL PIECES IN EACH STYLE
      DO 310 IS=1,NSTYL,1
C                                     GET STYLE NAME
      DO 260 I=1,5,1
  260 CALL WGET (IGWORD+1,NAME(I))
      IF (OLDUN) GO TO 305
C                                     GET STUFF FROM DIREC
      CALL DIREC (ISRCH3,NAME,INTNO,NSCTR,ISTAT)
      IF (ISTAT.EQ.0) GO TO 270
      WRITE (6,1500) NAME
      GO TO 555
  270 ISCTR=NSCTR
      CALL EXEC (14,1428,IBUFR,128,STYLE,NSCTR)
      IRULE = JLOF(MASK3,IBUFR(2))
      LP=JLOF(MASKL,IBUFR)
      LSCTR=ISCTR+JLOF(MASK4,IBUFR)
C                                     GET NUMBER OF SIZES
      CALL WGET (IGWORD+5,ITEMP)
      NSIZE=JLOF(MASK8,ITEMP)
      MASHD=.FALSE.
      IGWORD=IGWORD+6
      DO 315 IZ=1,NSIZE,1
      IP=0
C                                     GET SIZE FOR EACH PIECE IN STYLE
      DO 280 I=1,3,1
  280 CALL WGET (IGWORD+I,NAME3(I))
      NAME3=IAND(NAME3,0777778)
      CALL SIZEP (1,IRULE,NAME3,KSIZE)
      IF (KSIZE) 283,286,290
  283 WRITE (6,8090) NAME3
      GO TO 555
  286 WRITE (6,8080) IRULE
      GO TO 555
C                                     CHECK IF NEED NEXT BUFFER IN CORE
  290 IF (MASHD) CALL EXEC (14,1028,IBUFR,128,STYLE,ISCTR)
      ISCTR=ISCTR+1
      IP = IP+25
      IF (IP.GT.LP) IP=LP
      L = MOD(IP-1,25)*5+3
      DO 300 J=3,L,5
      IF (IPNTR.EQ.BUFSIZ) CALL SPOOL (IWORK(1),IPNTR,DPNTR)
      IWORK(IPNTR+1)=NULL
      IWORK(IPNTR+6)=IBUFR(J)
      IWORK(IPNTR+7)=KSIZE
      CALL JSTF (IRULE,MASKL,IWORK(IPNTR+7))
      IWORK(IPNTR+8)=IBUFR(J+4)
  300 IPNTR=IPNTR+8
      IF (ISCTR.GE.LSCTR) GO TO 305
C                                     SET FLAG = BUFFER TO COME IN
      MASHD=.TRUE.
      GO TO 290
  305 IGWORD=IGWORD+3
  310 ISCTR=NSCTR
      GO TO 555
C
C     DRAW A MARKER COMMAND                                    QUEUE=6
C
  350 CONTINUE
C                                     GET MARKER ID
      DO 360 I=1,5,1
  360 CALL WGET (IGWORD+1,NAME(I))
C                                     GET MARKER INFO
  365 IF (FLAG4) CALL MDISK (-1)
```

```
      FLAG4=.FALSE.
      CALL DIREC (ISRCH1,NAME,INTNO,NSCTR,ISTAT)
      IF (ISTAT.EQ.0) GO TO 370
      IF (OLDUN.AND.NSTYL.GT.1) GO TO 420
      WRITE (6,1200) NAME
      GO TO 555
  370 ISCTR=NSCTR
  380 CALL UDSC (1)
      CALL EXEC (14,1428,IBUFR,128,MARKR,ISCTR)
      IF (ISCTR.NE.NSCTR) GO TO 440
      LSCTR=NSCTR+JLDF(MASK9,IBUFR)
      LP = JLDF(MASK10,IBUFR)
      IP=0
      ISCTR=ISCTR+1
      GO TO 380
  400 ISCTR=ISCTR+1
      IP = IP+15
      IF (IP.GT.LP) IP=LP
      L = MOD(IP-1,15)*8+3
      DO 410 I=3,L,8
      IF (IPNTR.EQ.BUFSIZ) CALL SPOOL (IWORK(1),IPNTR,OPNTR)
      IWORK(IPNTR+1)=NULL
      IWORK(IPNTR+6)=IBUFR(I)
      IWORK(IPNTR+7)=IBUFR(I+4)
      IRULE=JLDF(MASK6,IBUFR(I+2))
      CALL JSTF (IRULE,MASKL,IWORK(IPNTR+7))
      IWORK(IPNTR+8)=IBUFR(I+3)
  410 IPNTR=IPNTR+8
      IF (ISCTR.LT.LSCTR) GO TO 380
      IF (.NOT.OLDUN) GO TO 555
  420 CONTINUE
      OLDUN=.FALSE.
      IQWORD=IQWORD+5
      GO TO 255
C
C     PLOT NESTED PIECE                                          QUEUE=7
C
  450 CONTINUE
C                                     GET PIECE ID
      DO 460 I=1,5,1
  460 CALL MOST (IQWORD+I,NAME(I))
C                                     PICK UP RULE AND NUMBER OF SIZES
      IQWORD=IQWORD+1
      CALL MOST(IQWORD+6,ITEMP)
      IRULE=JLDF(MASK5,ITEMP)
      NSIZE=JLDF(MASK8,ITEMP)
      IF (NSIZE.GT.0) GO TO 470
C                                     ALL SIZES
      NAME6(6)=IRULE
      CALL DIREC (IGETS6,NAME6,INTNO,NSCTR,ISTAT)
      IF (ISTAT.NE.0) GO TO 286
      CALL EXEC (14,1026,IBUFR,2,RULES,NSCTR)
      NSIZE=JLDF(MASK2,IBUFR(2))
      DO 462 I=1,NSIZE,1
      IF (IPNTR.EQ.BUFSIZ) CALL SPOOL (IWORK(1),IPNTR,OPNTR)
      DO 461 J=1,5,1
  461 IWORK(IPNTR+J)=NAME(J)
      IWORK(IPNTR+6)=NULL
      IWORK(IPNTR+7)=I
      CALL JSTF (IRULE,MASKL,IWORK(IPNTR+7))
      IWORK(IPNTR+8)=0
  462 IPNTR=IPNTR+8
      GO TO 555
                                      SPECIFIC SIZES PLOTED
  470 IQWORD=IQWORD+6
```

```
      DO 490 I=1,NSIZE,1
C                                        DO YOUR THING WITH PIECE SIZES
      DO 475 J=1,3,1
  475 CALL MOST (IQWORD+J,NAME3(J))
      CALL SIZEP (1,IRULE,NAME3,KSIZE)
      IF (KSIZE) 283,280,478
  478 IF (IPNTR.EQ.BUFSIZ) CALL SPOOL (IWORK(1),IPNTR,DPNTR)
      DO 480 J=1,5,1
  480 IWORK(IPNTR+J)=NAME(J)
      IWORK(IPNTR+6)=NULL
C                                        TRANSFER RULE AND SIZE
      IWORK(IPNTR+7)=KSIZE
      CALL JSTF (IRULE,MASKL,IWORK(IPNTR+7))
      IWORK(IPNTR+8)=0
      IQWORD=IQWORD+3
      IPNTR=IPNTR+8
  490 CONTINUE
      GO TO 555
C
C       SET FLAG TO GRADE ZERO FLAGGED PIECES AND/OR MARKERS
C
  550 CONTINUE
      CALL MOST (IQWORD+1,ITEMP)
      IF (ITEMP.EQ.1) GDMARK=.TRUE.
      IF (ITEMP.EQ.2) GDPIEC=.TRUE.
  555 CONTINUE
C
C      GRADE PIECES AND MARKERS CALLED FOR IN LASERMATIC SYSTEM
C
      IF (.NOT.GDMARK) GO TO 575
      IF (FLAG4) CALL MDISK (-1)
      FLAG4=.TRUE.
      WRITE (6,9050)
      INTNO=-1
  560 INTNO=INTNO+1
      CALL DIREC (IGRAB1,NAME6,INTNO,NSCTR,ISTAT)
      IF (ISTAT.EQ.5) GO TO 575
      IF (ISTAT.NE.0) GO TO 560
      IF (NAME6(6).NE.0) GO TO 560
      WRITE (6,9030) (NAME6(I), I=1,5,1)
      ISCTR=NSCTR
  571 CALL QDSC (1)
      CALL EXEC (14,1020,IBUFR,128,MARKR,ISCTR)
      IF (ISCTR.NE.NSCTR) GO TO 572
      LSCTR=NSCTR+JLOF(MASK4,IBUFR)
      LP=JLOF(MASK10,IBUFR)
      IP=0
      ISCTR=ISCTR+1
      GO TO 571
  572 ISCTR=ISCTR+1
      IP=IP+15
      IF (IP.GT.LP) IP=LP
      L=MOD(IP-1,15)*8+3
      DO 574 I=3,L,8
      IF (IPNTR.EQ.BUFSIZ) CALL SPOOL (IWORK(1),IPNTR,DPNTR)
      IWORK(IPNTR+1)=NULL
      IWORK(IPNTR+6)=IBUFR(I)
      IWORK(IPNTR+7)=IBUFR(I+4)
      IRULE=JLOF(MASK6,IBUFR(I+2))
      CALL JSTF (IRULE,MASKL,IWORK(IPNTR+7))
      IWORK(IPNTR+8)=IBUFR(I+3)
  574 IPNTR=IPNTR+8
      IF (ISCTR.LT.LSCTR) GO TO 571
      GO TO 560
C
```

```
C       GRADE PIECES CALLED FOR
C
  575 CONTINUE
      IF (.NOT.GOPIEC) GO TO 595
      CALL MOISK (1)
      WRITE (6,9070)
      INTNO = -1
  580 INTNO = INTNO+1
      CALL DIREC (IGRAB2,NAME6,INTNO,NSCTR,ISTAT)
      IF(ISTAT.EQ.5) GO TO 595
      IF(ISTAT.NE.0) GO TO 580
      IF (NAME6(6).NE.0) GO TO 580
      WRITE (6,9030) (NAME6(I), I=1,5,1)
      CALL EXEC (14,1028,NAME3,2,PIECE,NSCTR)
      IRULE=JLOF(MASK2,NAME3(2))
      NAME6(6)=IRULE
      CALL DIREC (IGETS6,NAME6,JNTNO,NSCTR,ISTAT)
      IF (ISTAT.EQ.0) GO TO 582
      WRITE (6,8080) IRULE
      GO TO 580
  582 CONTINUE
      CALL EXEC (14,1028,IBUFR,2,RULES,NSCTR)
      NSIZE=JLOF(MASK11,IBUFR(2))
      DO 585 I=1,NSIZE,1
      IF (IPNTR.EQ.BUFSIZ) CALL SPOOL (IWORK(1),IPNTR,OPNTR)
      DO 584 J=1,5,1
  584 IWORK(IPNTR+J) = NAME(J)
      IWORK(IPNTR+6) = INTNO
      IWORK(IPNTR+7) = I
      CALL JSTF(IRULE,MASKL,IWORK(IPNTR+7))
      IWORK(IPNTR+8) = 0
  585 IPNTR = IPNTR+8
      GO TO 580
  595 CONTINUE
C
C       ZERO GRADE FILE IF NEW PIECES ARE TO BE ADDED
C
      IF (IPNTR.GT.0) GO TO 605
      FLAG2=.TRUE.
      GO TO 640
  605 CONTINUE
      I=IPNTR/6
      IF (DEBUG) WRITE (1,1002) I
      CALL UDSC (4)
      CALL EXEC (18,GRADD,LIMIT)
      DO 610 ISECT=0,LIMIT-1,1
  610 CALL EXEC (15,201028,0,1,GRADD,ISECT)
      CALL EXEC (15,201028,0,1,GRADE,0)
C
C       TOP OF MULTI PLATTER LOOP
C
  640 CONTINUE
      IF (PASS.LE.PLATNO) GO TO 650
      WRITE (1,9020)
      IF (FLAG3.NE.0) GO TO 935
      GO TO 940
  650 FLAG4=.TRUE.
C
C       TRANSFER PIECES FROM TPECD TO PIECE.  FLAG1 IS TRUE IF WE'VE
C       GONE THROUGH HERE LAST TIME AND HAVE FOUND NO PIECES LEFT TO MOVE.
C
      IF (FLAG1) GO TO 800
      FLAG1=.TRUE.
      IF (DEBUG) WRITE (1,1003)
C
```

```
C     QUICK CHECK FOR VIRGIN FILE
C
      CALL UDSC (0)
      CALL EXEC (14,1028,ISTAT,1,TPECS,0)
      IF (ISTAT.EQ.0) GO TO 800
      INTNO=-1
      NAME6(6)=0
C                                     FLAG5 TRUE IF HEADING NOT OUT
      FLAG5=.TRUE.
  730 INTNO=INTNO+1
      CALL DIREC (ILOOK5,NAME6,INTNO,NSCTR,ISTAT)
      IF (ISTAT.EQ.5) GO TO 775
      IF (ISTAT.NE.0) GO TO 730
C
C     USE BUFFER TO MOVE PIECES
C
      IF (BUFSIZ-IPNTR.LT.1536) CALL SPOOL (IWORK(1),IPNTR,DPNTR)
C                                     GET FIRST SECT OF PIECE
      CALL EXEC (14,1028,IWORK(IPNTR+1),128,TPECS,NSCTR)
      NSECT=JLOF(MASK4,IWORK(IPNTR+1))
      IF (NSECT.GT.0) GO TO 735
      CALL ERR (31,1)
      GO TO 760
C                                     GET REST OF DATA
  735 IF (NSECT.GT.1) CALL EXEC (14,1028,IWORK(IPNTR+129),
     +      (NSECT-1)*128,TPECS,NSCTR+1)
      IF (.NOT.FLAG4) GO TO 736
      FLAG4 = .FALSE.
      CALL MOISK (PASS)
      CALL EXEC (18,PIECE,LIMIT)
C                                     TRY TO GET DIRECTORY SPACE
  736 CALL DIREC (ISTOR2,NAME6,JNTNO,JSCTR,ISTAT)
  737 IF (ISTAT.EQ.0) GO TO 740
      IF (ISTAT.EQ.1) GO TO 748
C                                     DUPLICATE NAME - TELL OPR
      WRITE (1,9010) (NAME6(I), I=1,5,1)
      READ (1,9030) ISTAT
      IF (ISTAT.NE.2HYE) GO TO 760
      CALL DIREC (IREPL2,NAME,JNTNO,JSCTR,ISTAT)
      FLAG5=.TRUE.
      GO TO 737
C                                     CHECK IF WE HAVE SPACE FOR DATA
  740 IF (JSCTR+NSECT.LE.LIMIT) GO TO 750
      FLAG1=.FALSE.
      CALL DIREC (IPURG2,NAME6,JNTNO,JSCTR,ISTAT)
      IF (JSCTR.LT.LIMIT) CALL EXEC (15,1028,1,1,PIECE,JSCTR)
  748 FLAG1=.FALSE.
      WRITE (6,9000)
      GO TO 800
  750 IF (FLAG5) WRITE (6,9040)
      FLAG5=.FALSE.
      WRITE (6,9030) (NAME6(I), I=1,5,1)
      CALL EXEC (15,1028,IWORK(IPNTR+1),NSECT*128,PIECE,JSCTR)
  760 CALL DIREC (IPURG5,NAME,INTNO,NSCTR,ISTAT)
      GO TO 730
C
C     CLEAR OUT TPECS FILE
C
  775 CONTINUE
      IF (DEBUG) WRITE (1,1004)
      CALL UDSC (0)
      CALL EXEC (18,TPECD,LIMIT)
      DO 780 I=0,LIMIT-1,1
  780 CALL EXEC (15,201028,0,1,TPECD,I)
      CALL EXEC (15,201028,0,1,TPECS,0)
```

```
C
C     LOOKUP PIECE NAMES
C
  800 IF (FLAG2.AND.FLAG3.GE.0) GO TO 920
      IF (DEBUG) WRITE (1,1005)
      IF (FLAG4) CALL MDISK(PASS)
      FLAG4=.FALSE.
      ISECT=0
      NPLATR=PASS+1
      OSECT=OPNTR/128
      LSECT=( (OPNTR-1)/BUFSIZ ) * ( BUFSIZ/128 )
      IF (LSECT.LT.0) LSECT=0
      IF (IPNTR.GT.0) GO TO 815
      IPNTR=OPNTR
      IF (IPNTR.GT.BUFSIZ) IPNTR=BUFSIZ
      GO TO 870
C
C     THIS IS THE CRUNCHER LOOKUP LOOP
C
  815 DO 860 I=1,IPNTR,8
      IF (IWORK(I).NE.NULL) GO TO 820
C                                     NAME LOOKUP GOVEN INTNO
      IF (JLOF(MASK7,IWORK(I+5)).NE.NPLATR) GO TO 860
      CALL DIREC (ILOOK2,IWORK(I),IWORK(I+5),NSCTR,ISTAT)
      GO TO 860
C                                     INTNO LOOKUP GIVEN NAME
  820 IF (IWORK(I+5).NE.NULL) GO TO 860
      IF (IWORK(I).NE.IWORK(I-8)) GO TO 830
      IF (IWORK(I+1).NE.IWORK(I-7)) GO TO 830
      IF (IWORK(I+2).NE.IWORK(I-6)) GO TO 830
      IF (IWORK(I+3).NE.IWORK(I-5)) GO TO 830
      IF (IWORK(I+4).NE.IWORK(I-4)) GO TO 830
      IF (I.EQ.1) GO TO 830
C                                     SAME NAME
      IWORK(I+5)=IWORK(I-3)
      IF (IWORK(I+6).EQ.STYFLG) GO TO 860
      ITEMP=JLOF(MASKL,IWORK(I+6))
      IF (ITEMP.EQ.0) CALL JSTF (IRULE,MASKL,IWORK(I+6))
      GO TO 860
  830 CALL DIREC (ISRCH2,IWORK(I),IWORK(I+5),NSCTR,ISTAT)
C                                     STYLE RECORD CHECK
      IF (IWORK(I+6).EQ.STYFLG) GO TO 840
      IRULE=JLOF(MASKL,IWORK(I+6))
      IF (IRULE.NE.0) GO TO 860
C                                     GET RULE ; FROM PIECE
      CALL EXEC (14,1028,NAME3,2,PIECE,NSCTR)
      IRULE=JLOF(MASK11,NAME3(2))
      CALL JSTF (IRULE,MASKL,IWORK(I+6))
      GO TO 860
C                                     MAINTAIN STYLE RECORD
  840 IF (ISTAT.EQ.0) GO TO 850
      FLAG3=-1
  850 IWORK(I+5)= -(IWORK(I+5)+1)
  860 CONTINUE
C
C     END LOOP FOR THIS SPOOL
C
      IF (OPNTR.EQ.0) GO TO 910
      IF (DEBUG) WRITE (1,1006) IPNTR,OSECT
      CALL EXEC (15,102H,IWORK(1),IPNTR,CHEK,OSECT)
      IF (OSECT.EQ.LSECT) GO TO 900
      IPNTR=OPNTR-ISECT*128
      IF (IPNTR.GT.BUFSIZ) IPNTR=BUFSIZ
  870 IF (DEBUG) WRITE (1,1007) IPNTR,ISECT
      CALL EXEC (14,1028,IWORK(1),IPNTR,CHEK,ISECT)
```

```
      USECT=ISECT
      ISECT=ISECT+BUFSIZ/128
      GO TO 815
C
C     END PIECE LOOKUP - SORT INTERNAL OR EXTERNAL
C
  900 OPNTR=OPNTR+IPNTR
      IF (DEBUG) WRITE (1,1008) PSORT,PASS,OPNTR,FLAG1,FLAG2,FLAG3
      CALL UDSC (0)
      CALL EXEC (10,PSORT,PASS,OPNTR,FLAG1,FLAG2,FLAG3)
C
C                                        INTERNAL SORT
  910 CONTINUE
      IF (DEBUG) WRITE (1,1009)
      IF (ISSW(0).LT.0) WRITE (6,1952) (IWORK(I), I=1,IPNTR,1)
      CALL SORTP (IWORK(1),IPNTR)
      IF (ISSW(0).LT.0) WRITE (6,1952) (IWORK(I), I=1,IPNTR,1)
      CALL SPOOL (IWORK(1),IPNTR,OPNTR)
C
C     LOAD PGRAD CHECK
C
      IF (FLAG2) GO TO 920
      IF (DEBUG) WRITE (1,1008) PGRAD,PASS,OPNTR,FLAG1,FLAG2,FLAG3
      CALL UDSC (0)
      CALL EXEC (10,PGRAD,PASS,OPNTR,FLAG1,FLAG2,FLAG3)
C
C     TRANSFER MORE PIECES CHECK
C
  920 CONTINUE
      IF (FLAG1) GO TO 930
      PASS=PASS+1
      GO TO 640
C
C     READY TO LOAD STYLE
C
  930 CONTINUE
      IF (FLAG3.EQ.0) GO TO 940
      PASS=PASS+1
      IF (FLAG3.NE.1) GO TO 640
  935 CONTINUE
      IF (DEBUG) WRITE (1,1008) WSTYL,PASS,OPNTR,FLAG1,FLAG2,FLAG3
      CALL MDISK (-1)
      CALL UDSC (0)
      CALL EXEC (10,WSTYL,OPNTR)
C
C     NOTHING TO DO BUT CALL DAVE BACK IN
C
  940 CONTINUE
      IF (DEBUG) WRITE (1,1010)
      CALL MDISK (-1)
      CALL UDSC (0)
      CALL EXEC (10,WPROD)
 1001 FORMAT("PGRAD LOADED.  CONSTANTS ARE:",I2,I6,2(1X,L1),I3,/,
     +       "BUFFER SPACE=",O6," STARTS AT ",O6)
 1002 FORMAT(15," PIECES IN LIST.",/,"INITIALIZE GRADE FILE.")
 1003 FORMAT("SCAN TREES.")
 1004 FORMAT("ZERO TREES FILE.")
 1005 FORMAT("FIND PIECES ON PLATTER.")
 1006 FORMAT("SPOOL OUT ",O6," WORDS STARTING AT SECTOR",I4)
 1007 FORMAT("SPOOL IN ",O6," WORDS STARTING AT SECTOR",I4)
 1008 FORMAT("LOAD ",2A2,A1,".  CONSTANTS ARE:",I2,I6,2(1X,L1),I3)
 1009 FORMAT("SORT IS INTERNAL.")
 1010 FORMAT("LOAD MONITOR.")
 1011 FORMAT("SCAN JOBQ.")
 1204 FORMAT("CANNOT FIND MARKER ",5A2)
```

```
 1004 FORMAT("STYLE NOT FOUND",I5)
 1952 FORMAT("PIECE LIST IN CORE:",/,(8O7))
 8080 FORMAT("RULE TABLE ",I3," NOT FOUND")
 8090 FORMAT("SIZE ",3A2," NOT AVALIABLE")
 9000 FORMAT("THIS PIECE FILE IS FULL")
 9010 FORMAT(/,5A2," DUPLICATE NAME - REPLACE (YES/NO)",/,"$")
 9020 FORMAT("ALL PIECE FILES SCANNED.")
 9030 FORMAT(5A2)
 9040 FORMAT(/,"PIECES TRANSFERED")
 9060 FORMAT("MARKERS FOR LASERMATIC:")
 9070 FORMAT("PIECES FOR LASERMATIC")
      END
      SUBROUTINE SPOOL (IWORK,IPNTR,OPNTR)
C
C     VERSION 2.1
C     EDITED 23 FEB 73
C     RONALD GORDON
C
      DIMENSION IWORK(1)
      INTEGER OPNTR
      INTEGER CHEK(3)
      LOGICAL DEBUG
      COMMON DEBUG
      DATA CHEK /2HCH,2HEK,1H /
      ISECT=OPNTR/128
      IF (DEBUG) WRITE (1,1001) IPNTR,ISECT
      CALL DOSC (2)
      CALL EXEC (15,102H,IWORK,IPNTR,CHEK,ISECT)
      OPNTR=OPNTR+IPNTR
      IPNTR=0
      RETURN
 1001 FORMAT("SPOOL OUT ",O6," WORDS STARTING AT SECTOR",I4)
      END
$END
FTN4
      PROGRAM PGRAD
C
C     VERSION 2.1
C     EDITED 20 FEB 73
C     RONALD GORDON
C
      DIMENSION IWORK(1)
      DIMENSION NAME6(6)
      INTEGER OSECT
      INTEGER CHEK(3)
      INTEGER BUFSIZ
      INTEGER ROSTER(5)
      INTEGER DONE,NULL,STYFLG
      INTEGER PASS,OPNTR,FLAG3
      INTEGER XY(256),WS(256)
      INTEGER OPTNS(3),PGRAB(3),WSTYL(3),WPROD(3)
      LOGICAL DEBUG
      LOGICAL FLAG1, FLAG2
      EQUIVALENCE (ROSTER(1),PASS),
     +            (ROSTER(2),OPNTR),
     +            (ROSTER(3),FLAG1),
     +            (ROSTER(4),FLAG2),
     +            (ROSTER(5),FLAG3)
      EQUIVALENCE (INTNO,NAME6(6))
C
C     COMMON AREA FOR PGRAD SET UP BY FIELD
C
      COMMON JFIELD(14)
      DATA OPTNS /2HOP,2HTN,1HS/,
     +     PGRAB /2HPG,2HRA,1HB/,
```

```
     +      ,STYL /2HWS,2HTY,1HL/,
     +      ,PROD /2HWP,2HRO,1HD/
      DATA CHEK /2HCH,2HEK,1H /
      DATA NULL /03777700B/, DONE /037777B/, STYFLG /177777B/
C
C     GET INPUT PARAMETERS
C
      CALL RMPAR (ROSTER)
      CALL ALLOC (IWORK(0),BUFSIZ)
      CALL UDSC (0)
      CALL EXEC (14,1102B,XY,4,OPTNS,0)
      DEBUG=JLOF(JFLD(1,1),XY(4)) .EQ.1
      DEBUG=DEBUG .OR. ISSW(13).LT.0
      IF (DEBUG) WRITE (1,1001) PASS,OPNTR,FLAG1,FLAG2,FLAG3,
     +      BUFSIZ,CHEK(-2)
      CALL FIELD
      KNT=0
      NPNTR=0
      ISECT=0
      INTNO=DONE
      FLAG2=.TRUE.
C
C     TOP OF LOOP TO SPOOL IN DATA
C
  210 NPNTR=NPNTR+BUFSIZ
      IF (NPNTR.GT.OPNTR) NPNTR=OPNTR
      IF (DEBUG) WRITE (1,1004) NPNTR,ISECT
      CALL UDSC (0)
      CALL EXEC (14,1102B,IWORK(1),NPNTR,CHEK,ISECT)
      OSECT=ISECT
      ISECT=ISECT+NPNTR/128
      IPNTR=0
C
C     TOP OF LOOP TO GRADE PIECES IN THIS BATCH
C
  310 CONTINUE
      IF (IWORK(IPNTR+1).EQ.NULL) GO TO 370
      IF (IWORK(IPNTR+6).EQ.INTNO) GO TO 340
      IF (IWORK(IPNTR+7).EQ.STYFLG) GO TO 350
      IF (IWORK(IPNTR+6).EQ.NULL) GO TO 370
      IF (KNT.EQ.0) GO TO 325
C
C     GRADE PIECE IN OUR BUFFER
C
      CALL GRADP (1,NAME6,KNT,RS,ISTAT,XY)
      IF(ISTAT.NE.0.AND.ISTAT.NE.98) GO TO 410
      KNT=0
  325 DO 330 I=1,6,1
  330 NAME6(I)=IWORK(IPNTR+I)
C                                   ADDITION TO LIST
  340 KNT=KNT+1
      RS(KNT)=IWORK(IPNTR+7)
      XY(KNT)=IWORK(IPNTR+8)
C                                   END LOOP
  350 IPNTR=IPNTR+8
      IF (IPNTR.LT.NPNTR) GO TO 310
      IF (NPNTR.EQ.OPNTR.AND.FLAG2) GO TO 400
      IF (DEBUG) WRITE (1,1002) IPNTR,OSECT
      CALL UDSC (0)
      CALL EXEC (15,1102B,IWORK(1),IPNTR,CHEK,OSECT)
      IF (NPNTR.LT.OPNTR) GO TO 210
      GO TO 400
C                                   NULL FOUND
  370 FLAG2=.FALSE.
      GO TO 350
```

```
C
C       END OF DATA
C
  400 IF (KNT.EQ.0) GO TO 420
      CALL GRADP (1,NAMEb,KNT,RS,ISTAT,XY)
      IF(ISTAT.EQ.4.OR.ISTAT.EQ.98) GO TO 420
  410 WRITE (6,2000) ISTAT
C
C       LOAD SOMEONE ELSE
C
  420 CONTINUE
      PASS=PASS+1
      IF (FLAG2) GO TO 430
  425 IF (DEBUG) WRITE (1,1008) PGRAB,PASS,DPNTR,FLAG1,FLAG2,FLAG3
      CALL DOSC (4)
      CALL EXEC (10,PGRAB,PASS,DPNTR,FLAG1,FLAG2,FLAG3)
  430 CONTINUE
      IF (.NOT.FLAG1) GO TO 425
      IF (FLAG3.EQ.4) GO TO 440
      IF (FLAG3.NE.1) GO TO 425
      IF (DEBUG) WRITE (1,1008) WSTYL,PASS,DPNTR,FLAG1,FLAG2,FLAG3
      CALL MOISK (-1)
      CALL DOSC (0)
      CALL EXEC (10,WSTYL,DPNTR)
  440 CONTINUE
      IF (DEBUG) WRITE (1,1009)
      CALL MOISK (-1)
      CALL DOSC (4)
      CALL EXEC (10,WPROD)
 1001 FORMAT("PGRAD LOADED.  CONSTANTS ARE:",I2,I5,2(1X,L1),I3,/,
     +      "BUFFER SPACE=",Ob," STARTS AT ",O6)
 1002 FORMAT("SPOOL OUT ",O6," WORDS STARTING AT SECTOR",I4)
 1004 FORMAT("SPOOL IN ",O6," WORDS STARTING AT SECTOR",I4)
 1006 FORMAT("LOAD ",2A2,A1,",  CONSTANTS ARE:",I2,I6,2(1X,L1),I3)
 1009 FORMAT("LOAD MONITOR.")
 2000 FORMAT("ERROR STATUS IN GRADP RETURNED AS",I2)
      END
ENDS
FTN4
      PROGRAM PLOTP
C.........VERSION 2.0  (MR)
C.....PROGRAM TO OUTPUT COMMANDS TO DRAW NESTED PLOTS OF PIECES
C.....ON THE GERBER DRUM PLOTTER OR TO CUT PIECES ON THE GERBER 622
C.....PATTERN CUTTING SYSTEM
      DIMENSION IFILE(3),IDNU(5),MSIZ(3),ISZ(42),XMIN(42),YMIN(42)
      DIMENSION XOFF(42),YOFF(42),ISIZE(42)
      DIMENSION IBUF(128),IENTN(42)
      DIMENSION VAL(20)
      DIMENSION IDUM(1536)
      DIMENSION IOPT(3)
      COMMON LOUT,ITYPE,ORIGX,ORIGY,LDIRC,XPoND,AXOLD,AYOLD,
     1   IPNF(3),IPNR(3),XMAX,YMAX,SCAL,XLAST,YLAST,LASTP,LASTA,
     2   LASTT,IPLT(3),JTYPE,JOB,JPOS,JTOOL,JHOME,JIZ,SPACE,IOPTN
     3   ,NPLAT,JB0,JB1,JB2,JB3,JB4
     +     ,XYCNST,MIRROR
      DATA IOPT/2HOP,2HTN,2HS /
      DATA IFILE/2HSE,2HCT,2HR /
C.....SET PLOTTER NUMBER FOR "PLOTF" FILE
      NPLAT=0
      IPLT(1)=2HPL
      IPLT(2)=2HOT
      IPLT(3)=2HF
C.....READ OPTION WORD
C.....BIT 0= GERBER DRUM, BIT 1= XYNETICS 2000, BIT 2= XYNETICS 1140
C.....BIT 3= GERBER 622, BIT 4= XYNETICS 2000 W/CUTTER
```

```
      CALL EXEC (14,1025,IDUM,11,IGP1,0)
      IOPTN=IDUM(3)
      YMAX=FLOAT(IDUM(10))*0.01
      IF (IDUM(10).EQ.0) YMAX=72.0
      XYCNST=FLOAT(IDUM(11))*0.01
      IF (IDUM(11).EQ.0) XYCNST=71.04
      MAXSZ=42
      SPACE=.500
      LOUT=6
      LASTT=1
C.....DEFINE DATA FIELDS
      J00=JFLD(0,1)
      JB1=JFLD(1,1)
      JB2=JFLD(2,1)
      JB3=JFLD(3,1)
      JB4=JFLD(4,1)
      JSZN=JFLD(15,8)
      JTYP=JFLD(15,1)
      JRUL = JFLD(7,7)
      JRLT=JFLD(13,6)
      JSIZ=JFLD(7,8)
      JSZ=JFLD(12,4)
      JTSE=JFLD(6,9)
      JMI=JFLD(5,1)
      JSECT=JFLD(3,4)
      JPTYP=JFLD(15,4)
      JTYPE=JFLD(15,3)
      JOB=JFLD(12,1)
      JPOS=JFLD(15,1)
      JTOOL=JFLD(14,2)
      JHOME=JFLD(12,1)
      JIZ=JFLD(11,12)
      JPAGE=JFLD(11,1)
      JSAMP = JFLD(8,8)
      JRULE=JFLD(11,12)
      CALL EXEC(14,1020,IADDS,1,IFILE,0)
C.....READ THE PIECE I.D. NUMBER
      DO 100    I=1,5
      CALL RDST(IADDS+I,IDNO(I))
  100 CONTINUE
 6010 FORMAT("PIECE NUMBER ",5A2)
      CALL DIREC(10048,IDNO,INTNO,NSEC,ISTAT)
      IF(ISTAT.EQ.0) GO TO 120
      WRITE(LOUT,6000) IDNO
 6000 FORMAT("PIECE ",5A2," NOT AVAILABLE")
      CALL FINI(2)
  120 IADDS=IADDS+6
C.....GET THE SCALE FACTOR
      CALL RDST(IADDS,IVAL)
      SCAL=FLOAT(IVAL)/100.
      IADDS=IADDS+1
C.....GET OPERATION TYPE, RULE SET AND NUMBER OF SIZES
      CALL RDST(IADDS,IVAL)
      IF(ISSW(15).LT.0)WRITE(LOUT,6030) IVAL
 6030 FORMAT(06)
      ITYPE=2HCH
      XMAX=1440.
C.....REDUCE LIMITS IF XYNETICS 1100 PLOTTER
      IF(JLDF(JB2,IOPTN).NE.1) GO TO 122
      XMAX=55.0
C.....REDUCE LIMITS IF XYNETICS 2000
  122 IF(JLDF(JB1,IOPTN).NE.1) GO TO 123
      XMAX=114.0
123   CALL DOSC(0)
C.....READ TABLE POSITION VALUES
```

```
      CALL EXEC(14,1028,VAL,40,IPLT,0)
      M=30
      IF(JLDF(JTYP,IVAL).EQ.1) M=10
      L=5
      IF(M.EQ.30) L=15
C.....M=10 FOR "CHECK" AND M=30 "CUT"
C.....READ TABLE POSITION AND DIRECTION
      ORIGX=VAL(L+1)
      ORIGY=VAL(L+2)
      AXOLD=VAL(L+3)
      AYOLD=VAL(L+4)
      XPEND=VAL(L+5)
      IF(ISSW(15).LT.0) WRITE(LOUT,3333)ORIGX,ORIGY,AXOLD,AYOLD,XPEND
 3333 FORMAT(5F9.3)
      XLAST=AXOLD
      YLAST=AYOLD
C
      IF(M.EQ.10) GO TO 125
C......"CUT" OPERATION. CHECK FOR CUTTER PRESENT.
      IF(JLDF(JB3,IOPTN).NE.0.OR.JLDF(JB4,IOPTN).NE.0) GO TO 124
      WRITE(LOUT,1040)
 1040 FORMAT("CUTTER NOT AVAILABLE")
      CALL FINI(2)
  124 CONTINUE
      ITYPE=2HCU
      XMAX=60.
      YMAX=48.
  125 IFILE(1)=2HGR
      IFILE(2)=2HAD
      IFILE(3)=2HE
      CALL EXEC(14,1028,IBUF,128,IPLT,0)
      LDIRC=IBUF(M+11)
      IF (LDIRC.EQ.0) LDIRC=1
      CALL UDSC (0)
      CALL EXEC(14,1028,IBUF,128,IFILE,NSEC)
      IRLT=JLDF(JRLT,IVAL)
      NSIZE=JLDF(JSIZ,IVAL)
      IF(NSIZE.EQ.0) GO TO 170
C.....READ SIZES REQUIRED
      IF (NSIZE.LE.MAXSZ) GO TO 135
  130 WRITE (LOUT,6200) MAXSZ
 6200 FORMAT("NUMBER OF SIZES EXCEEDS ",I4)
      CALL FINI (2)
  135 DO 150   J=1,NSIZE
      DO 140   K=1,3
      IADD=(IADDS+(J-1)*3)+K
  140 CALL RWST(IADD,MSIZ(K))
      IF(ISSW(15).LT.0)WRITE(LOUT,6020) MSIZ
 6020 FORMAT("SIZE ",3A2)
      CALL SIZEP (1,IRLT,MSIZ,ISIZE(J))
  150 CONTINUE
  170 NSZ=(12*JLDF(JTSE,IBUF))/((12*JLDF(JSZ,IBUF))+1)
      NSZ1=NSZ
      NS=(NSZ-1)/12+1
 1000 FORMAT (6I8)
      IF(NS.LE.12) GO TO 175
      WRITE(LOUT,1015)
 1015 FORMAT("MORE THAN 12 SECTORS OF DATA")
      CALL FINI(2)
  175 JK=0
      ISIZN=0
      DXP=0.
      DXN=0.
      DYP=0.
      DYN=0.
```

```
      FACTR=SCAL/1000.
C.....COMPUTE OFFSET VALUES FOR PIECE
      DO 410   K=1,NS
      IF(NS.EQ.1) GO TO 350
      IF(K.EQ.1) CALL EXEC(14,1028,IDUM,126*NS,IFILE,NSEC)
      IND=(K-1)*128+1
C.....NOTE...S/R TRANS IS USED FOR CORE TRANSFERS FROM THE 12 SECTOR
C.....CORE BUFFER (IDUM) TO THE 1 SECTOR CORE BUFFER (IBUF).
C.....THIS WAS DONE TO OPTIMIZE DISK USAGE WITH MINIMUM RE-WRITE.
      CALL TRANS(IBUF,IDUM(IND))
  350 IL=113
      NSZ=NSZ-12
      IF(NSZ.LT.0) IL=((NSZ+12)*10)-7
      IF(ISSW(6).LT.0) WRITE(LOUT,1000)NS,NSZ,K,IL
C.....SCAN SIZES FOR OFFSET VALUES
      DO 400   I=3,IL,10
      ISIZN=ISIZN+1
C.....CHECK FOR RULE TABLE MATCH
      IF (JLOF(JRUL,IBUF(I)).NE.IRLT) GO TO 400
      IF(NSIZE.EQ.0) GO TO 360
C.....CHECK FOR SIZE TO PLOT OR CUT
      DO 360   M=1,NSIZE
      IF(JLOF(JSZN,IBUF(I)).EQ.ISIZE(M)) GO TO 380
  360 CONTINUE
      GO TO 400
  380 JK=JK+1
      IF (JK.GT.MAXSZ) GO TO 130
      ISZ(JK) = JLOF(JSZN,IBUF(I))
      IENTN(JK)=ISIZN
      XMIN(JK)=IBUF(I+1)*FACTR
      YMIN(JK)=IBUF(I+2)*FACTR
      XOFF(JK)=FACTR*FLOAT(IBUF(I+3)*LOIRC)
      YOFF(JK)=FACTR*FLOAT(IBUF(I+4)*LOIRC)
      IF(ISSW(15).LT.0) WRITE(LOUT,5)XMIN(JK),YMIN(JK),XOFF(JK),YOFF(JK)
    5 FORMAT(4F8.3)
      AX=XMIN(JK)+XOFF(JK)
      AY=YMIN(JK)+YOFF(JK)
      BX=-XMIN(JK)+XOFF(JK)
      BY=-YMIN(JK)+YOFF(JK)
      DXP=AMAX1(BX,DXP)
      DXN=AMIN1(AX,DXN)
      DYP=AMAX1(BY,DYP)
      DYN=AMIN1(AY,DYN)
  400 CONTINUE
  410 CONTINUE
      NSZ=NSZ1
      IGRL = 1
      CALL EXEC(14,1026,IBUF,1,IFILE,NS+NSEC)
      NSECT=JLOF(JSECT,IBUF)
      IF(NSECT.LE.12) GO TO 415
      WRITE(LOUT,1015)
      CALL FINI(2)
C.....INITIALIZE S/R STORE
  415 CALL STORE(-1,0,0,0)
      MM=0
C.....PROCESS SIZES REQUIRED.
      DO 855   I=1,NSZ
      DO 430   M=1,JK
      IF(I.EQ.IENTN(M)) GO TO 440
  430 CONTINUE
C.....SKIP THIS SIZE
      NS=NS+NSECT
      GO TO 850
  440 CONTINUE
      MM=MM+1
```

```
      LSTAT=4H
      IF(ITYPE.EQ.2HCH.AND.MM.EQ.1) GO TO 450
      IF(ITYPE.NE.2HCU) GO TO 460
      DXN=XMIN(MM)
      DXP=-DXN
      DYN=YMIN(MM)
      DYP=-DYN
      IF(ISSW(13).LT.0)WRITE(LOUT,6666)AX,AY,I,(XMIN(L),L=1,5)
 6666 FORMAT(2F9.3,I6,5F9.3)
  450 CALL LOCAT(DXN,DXP,DYN,DYP,LSTAT)
      IF(LSTAT.EQ.0)GO TO 455
      WRITE(LOUT,2222)
 2222 FORMAT("PIECE EXCEEDS PLOTTER LIMITS")
      GO TO 999
  455 TX=0.0
      TY = 0.0
C.....COORDINATES FOR SIZES AND LETTERING POINT
      XS=DXN
      YS=DYN-0.3*SCAL
      XL=0.0
      YL=0.0
      IF(ITYPE.EQ.2HCU) GO TO 490
  460 TX=XOFF(MM)
      TY=YOFF(MM)
  490 CALL UDSC(0)
      CALL EXEC(14,102B,IDUM,128*NSECT,IFILE,NSEC+NS)
      CALL TRANS(IBUF,IDUM)
      IF (ITYPE.NE.2HCH) GO TO 500
C.....PLOT GRADE RULE NUMBERS WHEN ONLY THE
C.....SAMPLE SIZE IS PLOTTED
      IPRUL=0
      N=2
      IF(JLOF(JSAMP,IBUF(N)).EQ.ISIZE(MM).AND.NSIZE.EQ.1) IPRUL=1
      MMM=ISIZE(MM)
      IF(NSIZE.EQ.0)MMM=MM
      CALL SIZEP(2,IRLT,MSIZ,MMM)
      IF(ISSW(15).LT.0)WRITE(LOUT,9999)MSIZ,MM,MMM,NSIZE,I,ISIZE(MM)
 9999 FORMAT(3A2,5I6)
      CALL SYMB(XS,YS,0.25*SCAL,MSIZ,0.0,6)
      NCH=1
C.....SUPPRESS BLANKS
      DO 485   JJJ=1,3
      IF(MSIZ(JJJ).EQ.2H  ) GO TO 485
      NCH=NCH+2
  485 CONTINUE
      XS=XS+0.25*SCAL*NCH
  500 CONTINUE
      IBASE=NSEC+NS
      MIF = JLOF(JMI,IBUF(1))
      NSC = NS
      NS = NS + NSECT
      X = TX
      Y = TY
      ISTL = 1
      INTP = 1
      INMI = 0
      IFLG = 1
      L1 = 1
      L2 = NSECT
      L3 = 1
      L4 = 3
      L5 = 126
      L6 = 3
      IGPF=0
      L7=-1
```

```
  505 DO 800 J = L1,L2,L3
      IF(INM1.GE.0.AND.J.NE.L1) L4=126
  510 DO 700 K=L4,L5,L6
      X1= X
      Y1 = Y
      NP1 = NPTYP
      X=IBUF(K+1)*FACTR+TX
      Y=IBUF(K+2)*FACTR+TY
      NPTYP = JLOF(JPTYP,IBUF(K))
      IF(IPROL.NE.0)IROLE=JLOF(JROLE,IBUF(K))
      IF(ISW1.GE.0) Y= -Y
      IF(NPTYP.LE.1.OR.NPTYP.EQ.8.OR.NPTYP.EQ.11) GO TO 550
C                                   GRADE OR INTERMEDIATE
      IF(NPTYP.EQ.4)GO TO 610
C                                   NOTCH POINT
  530 IFLG = 1
      IF(NPTYP.EQ.2) GO TO 570
C                                   PUNCH HOLE POINT
      IF(NPTYP.EQ.3) GO TO 590
C                                   STRIPE LINE POINT
      IF(NPTYP.EQ.5) GO TO 630
C                                   GRAIN LINE POINT
      IF(NPTYP.EQ.6) GO TO 650
C                                   ALTERNATE START POINT
      IF(NPTYP.EQ.7) GO TO 670
C                                   INTERNAL POINT
      IF(NPTYP.EQ.10) GO TO 830
C                                   END POINT
      IF (NPTYP.NE.9) GO TO 700
C.....LETTERING POINT
      XL=X
      YL=Y
      GO TO 700
C                                   NONE OF THESE, IGNORE
C.....GRADE OR INTERMEDIATE POINT
  550 IF (ITYPE.EQ.2HCM.AND.LASTT.NE.1) CALL TOOL (1)
      IF (ITYPE.EQ.2HCU.AND.LASTT.NE.3) CALL TOOL (3)
      IF(IGPF.EQ.0)IGPF=(J+L7)*128+K
      IGPN=(J+L7)*128+K
      IF(ISSW(10).LT.0)WRITE(6,5555)X,Y
 5555 FORMAT(2F9.3)
      CALL PLOT(X,Y,IFLG+2)
      IF(IPROL.NE.0 .AND. NPTYP.EQ.0) CALL ROLEN (IROLE,X,Y)
      IFLG = 0
      IF(INM1.EQ.0.OR.NP1.NE.4)GO TO 700
      IF(IGPN.EQ.IGPF.OR.IGPN.EQ.IGPL)GO TO 700
      IF(ITYPE.EQ.2HCM) GO TO 560
      CALL NOTCH(X1,Y1,X,Y)
      GO TO 700
  560 CALL PLOT(X1,Y1,2)
      CALL PLOT(X,Y,3)
      GO TO 700
C.....PUNCH HOLE. SELECT PEN.
  570 CALL TOOL(1)
      DEL=.1*SCAL
      CALL PLOT(X-DEL,Y,3)
      CALL PLOT(X+DEL,Y,2)
      CALL PLOT(X,Y-DEL,3)
      CALL PLOT(X,Y+DEL,2)
      IF (IPROL.NE.0) CALL ROLEN (IROLE,X,Y)
      GO TO 700
C.....STRIPE LINE
  590 IF (INM1.NE.0) GOTO 700
      ISTF = -1**IS1L
      CALL TOOL(1)
```

```
      IF(ISTF.LT.0)CALL PLOT(X,Y,3)
      IF(ISTF.GT.0)CALL PLOT(X,Y,2)
      IF(IPROL.NE.0) CALL RULER (IRULE,X,Y)
      ISTL = ISTL + 1
      GO TO 700
C.....NOTCH
  610 IF(INMI.NE.0)GO TO 700
      IF(ITYPE.EQ.2HCH) GO TO 620
      CALL NOTCH(X,Y,X1,Y1)
      GO TO 700
  620 CALL PLOT(X,Y,2)
      CALL PLOT(X1,Y1,3)
      GO TO 700
C.....GRAIN LINE
  630 IF(INMI.NE.0)GO TO 700
      IF(IGRL.GT.2.AND.ITYPE.EQ.2HCH) GO TO 700
      CALL TOOL(1)
      IGRF = -1 **IGRL
      IF(IGRF.GT.0)GO TO 635
      CALL PLOT(X,Y,3)
      GO TO 645
  635 DELX = X - X1
      DELY = Y - Y1
      IF(ABS(DELX).GT..05)GO TO 637
      THETA = 1.57079
      IF(DELY.LT.0)THETA = -THETA
      GO TO 640
  637 THETA = ATAN(DELY/DELX)
      IF(DELX.LT.0.)THETA = THETA + 3.14159
  640 C=.5*COS(.5236)*SCAL
      D=.5*SIN(.5236)*SCAL
      CCOS = C * COS(THETA)
      CSIN = C * SIN(THETA)
      DCOS = D * COS(THETA)
      DSIN = D * SIN(THETA)
      APP = CCOS - DSIN
      BPP = CSIN + DCOS
      APM = CCOS + DSIN
      BPM = CSIN - DCOS
      AMP = -CCOS - DSIN
      BMP = - CSIN + DCOS
      AMM = -CCOS + DSIN
      BMM = - CSIN - DCOS
      CALL PLOT(X1+APP,Y1+BPP,2)
      CALL PLOT(X1+APM,Y1+BPM,3)
      CALL PLOT(X1,Y1,2)
      CALL PLOT(X,Y,2)
      CALL PLOT(X+AMP,Y+BMP,2)
      CALL PLOT(X+AMM,Y+BMM,3)
      CALL PLOT(X,Y,2)
  645 IGRL = IGRL + 1
      GO TO 700
C.....ALTERNATE START POINT
  650 IF(INMI.NE.0)GO TO 700
      CALL TOOL(1)
      DELX=.05*SCAL
      DELY=.0866*SCAL
      CALL PLOT(X+DELX,Y+DELY,3)
      CALL PLOT(X-DELX,Y-DELY,2)
      CALL PLOT(X-DELX,Y+DELY,3)
      CALL PLOT(X+DELX,Y-DELY,2)
      CALL PLOT(X-DELX*2.,Y,3)
      CALL PLOT(X+DELX*2.,Y,2)
      GO TO 700
  670 INTF = -1**INTP
```

```
      IF(INTP.LT.0)CALL PLOT(X,Y,3)
      IF(INTP.GT.0)CALL PLOT(X,Y,2)
      INTP = INTP+1
  700 CONTINUE
      NSC=NSC+J-INMI+NSEC
      IND=(NSC-IBASE)*128+1
      CALL TRANS(IDUM,IDUM(IND))
  800 CONTINUE
      GO TO 850
  830 IF(MIF.NE.1) GO TO 850
C.....MIRRORED PIECE
      L1 = NSECT
      L2 = 1
      L3 = -1
      L4 = K-3
      L5 = 3
      L6 = -3
      IGPL=IGPM
      INMI = 2
      GO TO 505
  850 IF(ITYPE.EQ.2HCH) GO TO 855
C.....RAISE THE CUTTER HEAD.
C.....DETAIL TO ALLOW HEAD TO CLEAR.
      CALL PLOT(XLAST,YLAST,3)
      CALL PLOT(XLAST,YLAST,3)
      CALL PLOT(0.,0.,3)
      CALL TOOL(1)
  855 CONTINUE
      IF(ITYPE.EQ.2HCH)CALL SYMB(XL-(0.3*SCAL*5.0),YL,0.3*SCAL,IDUM,0.0,
     * 10)
C.....LEAVE TOOL AT CENTER OF PIECE.
      CALL PLOT (0.0,0.0,3)
      CALL STORE(0,0,0,0)
  900 CALL FINI(1)
      STOP
      END
      ENDS


ASMB
      NAM PLTST,7
      ENT PLTST
      EXT .ENTR,EXEC
IND   NOP           OPERATION INDICATOR ADDRESS
IVAL  NOP           LOAD/STORE VALUE ADDRESS
PLTST NOP
      JSB .ENTR
      DEF IND
*
      LDB IND,I     LOAD OPERATION INDICATOR
      SSB           STORE OR LOAD?
      JMP LOAD      LOAD
*
      LDA PUT       LOAD "PUT" ADDRESS
      SLB           OPERATION FOR "GET" OR "PUT"?
      LDA GET       "GET", LOAD "GET" ADDRESS
      STA ADD       SAVE
      ADA .-3       ADJUST OPERATION INDICATOR
      LDA SPMSK     LOAD "STOP/FINI" MASK
      SSB           SAVE "STOP/FINI"?
      LDA GPMSK     NO, LOAD "GET/PUT" MASK
      STA MASK      SAVE MASK
      LDA IVAL,I    LOAD NEW VALUE
      SSB           CHANGE "STOP/FINI"?
      JMP ADJ       YES
```

```
        AND GPMSK       MASK FOR NEW "GET/PUT"
        JMP STORE       STORE NEW VALUE
ADJ     AND .+1         MASK FOR NEW "STOP/FINI"
        RAR             ADJUST TO B15
STORE   STA B           SAVE NEW VALUE
        LDA ADD,I       LOAD DATA WORD
        AND MASK        MASK OFF OLD VALUE
        IOR B           INSERT NEW VALUE
        LDB ADD         LOAD DATA WORD ADDRESS
        JSB EXEC        EXEC BASE PAGE STORE
        DEF *+2         RETURN ADDRESS
        DEF RC19        REQUEST CODE ADDRESS
        JMP PLIST,I     RETURN
*
LOAD    LDA PUT,I       LOAD "PUT" CONTENTS
        SLB             OPERATION FOR "GET" OR "PUT"?
        LDA GET,I       "GET", LOAD "GET" CONTENTS
        STA MASK        SAVE
        ADB .+2         ADJUST OPERATION INDICATOR
        LDA SPMSK       LOAD "STOP/FINI" MASK
        SSB             RETURN "STOP/FINI"?
        LDA GPMSK       NO, LOAD "GET/PUT" MASK
        AND MASK        MASK FOR VALUE
        SSB,RSS         "GET/PUT"?
        RAL             NO, ADJUST "STOP/FINI" TO B0
        STA IVAL,I      STORE VALUE
        JMP PLIST,I     RETURN
*
SPMSK   OCT 100000      "STOP/FINI" MASK
GPMSK   OCT 077777      "GET/PUT" MASK
MASK    OCT 0           TEMPORARY STORAGE
ADJ     OCT 0           TEMPORARY STORAGE
RC19    DEC -19         BASE PAGE STORE REQUEST CODE
*
GET     OCT 36          ADDRESS OF "GET" DATA WORD
PUT     OCT 37          ADDRESS OF "PUT" DATA WORD
*
A       EQU 0           A-REGISTER
B       EQU 1           B-REGISTER
.       EQU 536         BASE PAGE CONSTANTS
        END


FTN4
        PROGRAM POPTS
        DIMENSION IBUF(128)
        DIMENSION MARKA(3)
        DATA MARKA/2HOP,2HTN,2HS /
        IPNT = 1
        YES = 2HYE
        NO = 2HNO
C.....IANS IS THE STORAGE LOCATION FOR  ALPHA ANSWERS
C.....INUM IS STORAGE LOCATION FOR NUMERIC ANSWERS
C.....ERROR MESSAGE 01...YOU MUST ENTER A YES OR NO REPLY TO CONTINUE
C.....ERROR MESSAGE 02...YOU MUST ENTER A NUMERIC VALUE TO CONTINUE
        DO 2 I=1,128
        ITEMP = 0
    2   IBUF(I) =0
        WRITE(1,12)
    1   CONTINUE
C.....OPTION QUESTIONS FOR WORD 1
  100   WRITE(1,13)
        READ(1,15)IANS
        IF(IANS.NE. YES.AND.IANS.NE. NO ) GO TO 500
        IF(IANS.EQ.YES) IBUF(1) = IOR  (IBUF(1),000001B)
```

```
1501  WRITE(1,25)
      READ(1,15) IANS
      IF(IANS.NE. YES.AND.IANS.NE. NO ) GO TO 501
      IF(IANS.EQ.YES) IBUF(1) = IOR  (IBUF(1),000004B)
1502  WRITE(1,30)
      READ(1,15)IANS
      IF(IANS.NE. YES.AND.IANS.NE. NO ) GO TO 502
      IF(IANS.EQ.YES) IBUF(1) = IOR  (IBUF(1),000010B)
1503  WRITE(1,33)
      READ(1,15)IANS
      IF(IANS.NE. YES.AND.IANS.NE. NO ) GO TO 503
      IF(IANS.EQ.YES) IBUF(1) = IOR  (IBUF(1),000020B)
1504  WRITE(1,35)
      READ(1,15)IANS
      IF(IANS.NE. YES.AND.IANS.NE. NO ) GO TO 504
      IF(IANS.EQ.YES) IBUF(1) = IOR  (IBUF(1),000040B)
1505  WRITE(1,38)
      READ(1,15)IANS
      IF(IANS.NE. YES.AND.IANS.NE. NO ) GO TO 505
      IF(IANS.EQ.YES) IBUF(1) = IOR  (IBUF(1),000100B)
1506  WRITE(1,40)
      READ(1,15)IANS
      IF(IANS.NE. YES.AND.IANS.NE. NO ) GO TO 506
      IF(IANS.EQ.YES) IBUF(1) = IOR  (IBUF(1),000200B)
1405  WRITE(1,404)
      READ(1,15) IANS
      IF(IANS.NE.YES .AND. IANS .NE. NO) GO TO 405
      IF(IANS.EQ.YES) IBUF(1) = IOR (IBUF(1),000400B)
1710  WRITE(1,740)
      READ(1,15) IANS
      IF(IANS.NE. YES .AND. IANS.NE. NO) GO TO 710
      IF(IANS.EQ. YES) IBUF(1) = IOR (IBUF(1),100000B)
      WRITE(1,41)
   3  CONTINUE
C.....OPTION QUESTIONS FOR WORD 2
 1507 WRITE(1,50)
      READ(1,15)IANS
      IF(IANS.NE. YES.AND.IANS.NE. NO ) GO TO 507
      IF(IANS.EQ.YES) IBUF(2) = IOR (IBUF(2),000001B)
 1508 WRITE(1,55)
      READ(1,15)IANS
      IF(IANS.NE. YES.AND.IANS.NE. NO ) GO TO 508
      IF(IANS.EQ.YES) IBUF(2) = IOR (IBUF(2),000004B)
 1509 WRITE(1,60)
      READ(1,15)IANS
      IF(IANS.NE. YES.AND.IANS.NE. NO ) GO TO 509
      IF(IANS.EQ.YES) IBUF(2) = IOR (IBUF(2),000020B)
 1510 WRITE(1,65)
      READ(1,15)IANS
      IF(IANS.NE. YES.AND.IANS.NE. NO ) GO TO 510
      IF(IANS.EQ.YES) IBUF(2) = IOR (IBUF(2),000040B)
 1511 WRITE(1,70)
      READ(1,15)IANS
      IF(IANS.NE. YES.AND.IANS.NE. NO ) GO TO 511
      IF(IANS.EQ.YES) IBUF(2) = IOR (IBUF(2),000200B)
 1512 WRITE(1,75)
      READ(1,15)IANS
      IF(IANS.NE. YES.AND.IANS.NE. NO ) GO TO 512
      IF(IANS.EQ.YES) IBUF(2) = IOR (IBUF(2),000400B)
 1702 WRITE(1,280)
      READ(1,15) IANS
      IF(IANS .NE. YES .AND. IANS .NE. NO) GO TO 702
      IF(IANS .EQ. YES) IBUF(2) = IOR (IBUF(2),001000B)
 1531 WRITE(1,76)
      READ(1,15) IANS
```

```
      IF(IANS .NE. YES .AND. IANS .NE.  NO) GO TO 531
      IF(IANS .EQ. YES) IBUF(2) = IOR (IBUF(2),#00200#B)
C.....OPTION QUESTIONS FOR WORD 3
      WRITE(1,80 )
   4  CONTINUE
 1514 WRITE(1,83)
      READ(1,15) IANS
      IF(IANS.NE. YES.AND.IANS.NE. NO ) GO TO 514
      IF(IANS.EQ.YES) IBUF(3) = IOR (IBUF(3),#00010#B)
C.....OPTION QUESTIONS FOR WORD 4
      WRITE(1,141)
 1515 WRITE(1,142)
      READ(1,15) IANS
      IF(IANS.NE. YES.AND.IANS.NE. NO ) GO TO 515
      IF(IANS.EQ.YES) IBUF(4) = IOR (IBUF(4),#v0#018)
 1526 WRITE(1,193)
      READ(1,15) IANS
      IF(IANS .NE. YES .AND. IANS .NE. NO) GO TO 526
      IF(IANS .EQ. YES) IBUF(4) = IOR (IBUF(4),#0#042B)
C.....GENERAL OPTION QUESTIONS
      WRITE(1,195)
   6  CONTINUE
 1516 WRITE(1,196)
      READ(1,15)IANS
      IF(IANS.NE. YES.AND.IANS.NE. NO ) GO TO 516
      IF(IANS.EQ.YES) IBUF(1) = IOR (IBUF(1),#v#v#2B)
      IF(IANS.NE. NO) IBUF(2) = IOR (IBUF(2),#v#v#2B)
 1517 WRITE(1,81)
      READ(1,15)IANS
      IF(IANS.NE. YES.AND.IANS.NE. NO ) GO TO 517
      IF(IANS.EQ.YES) IBUF(2) = IOR (IBUF(2),#v#01vB)
      IF(IANS.NE.NO ) IBUF(3) = IOR (IBUF(3),#v#v#45)
      IF(IAND(IBUF(2),1#B).NE.# .OR.IAND(IBUF(3),46).NE.#) GO TO 1535
 1518 WRITE(1,82)
      READ(1,15)IANS
      IF(IANS.NE. YES.AND.IANS.NE. NO ) GO TO 518
      IF(IANS.EQ.YES) IBUF(2) = IOR (IBUF(2),#v#1#0B)
      IF(IANS.NE.NO) IBUF(3) = IOR (IBUF(3),#v#v#28)
      IF(IAND(IBUF(2),1#0B).NE.# .OR.IAND(IBUF(3),28).NE.#) GO TO 1535
 1513 WRITE(1,#5)
      READ(1,15)IANS
      IF(IANS.NE. YES.AND.IANS.NE. NO ) GO TO 513
      IF(IANS.EQ.YES) IBUF(3) = IOR (IBUF(3),#0#C315)
 1535 WRITE(1,221)
 1519 WRITE(1,2#0)
      IF(INUM .EQ. YES .AND. INUM .EQ. NO) GO TO 519
      READ(1,*) INUM
      IBUF(5) = INUM
 1520 WRITE(1,205)
      READ(1,*) INUM
      IF(INUM .EQ. YES .AND. INUM .EQ. NO) GO TO 520
      IBUF(6) = INUM
 1521 WRITE(1,218)
      READ(1,*) INUM
      IF(INUM .EQ. YES .AND. INUM .EQ. NO) GO TO 521
      ITRK = JFLD(15,8)
      CALL JSTF (INUM,ITRK,IBUF(7))
 1522 WRITE(1,220)
      READ(1,*) INUM
      IF(INUM .EQ. YES .AND. INUM .EQ. NO) GO TO 522
      ISCTR = JFLD(7,8)
      CALL JSTF (INUM,ISCTR,IBUF(7))
      IBUF(8) = IPNT
C.....IF NO DEFAULT SAMPLE SIZE ENTERED VIA THE DIGITIZER
C.....3 MEANS USE THE THIRD ENTRY IN RULE TABLE
```

```
1406  WRITE(1,400)
      READ(1,*) INUM
      IF(INUM .EQ. YES .AND. INUM .EQ. NO) GO TO 406
      IBUF(9) = INUM
1524  WRITE(1,215)
      READ(1,*) INUM
      IF(INUM .EQ. YES .AND. INUM .EQ. NO) GO TO 524
      IBUF(10) = INUM
1525  WRITE(1,216)
      READ(1,*)ANUM
      IF(INUM .EQ. YES .AND. INUM .EQ. NO) GO TO 525
      IBUF(11) = ANUM*100
1701  WRITE(1,227)
      READ(1,15) IANS
      IF(IANS.NE.YES.AND.IANS.NE. NO)GO TO 701
      IF(IANS.EQ.YES) 700,10
 700  CONTINUE
      IF(IAND(IBUF(1),1B) .EQ. 0) GO TO 110
 112  WRITE(1,123)
 110  CONTINUE
      IF(IAND(IBUF(1),2B).EQ. 0) GO TO 111
 113  WRITE(1,130)
 111  CONTINUE
      IF(IAND(IBUF(1),4B).EQ. 0) GO TO 100
 114  WRITE(1,133)
 100  CONTINUE
      IF(IAND(IBUF(1),10B).EQ. 0) GO TO 86
 115  WRITE(1,135)
  86  CONTINUE
      IF(IAND(IBUF(1),20B).EQ. 0) GO TO 87
 116  WRITE(1,140)
  87  CONTINUE
      IF(IAND(IBUF(1),40B).EQ. 0 ) GO TO 88
 117  WRITE(1,141)
  88  CONTINUE
      IF(IAND(IBUF(1),100B).EQ. 0) GO TO 90
 118  WRITE(1,150)
  90  CONTINUE
      IF(IAND(IBUF(1),200B).EQ. 0) GO TO 92
 119  WRITE(1,155)
  92  CONTINUE
      IF(IAND(IBUF(1),400B) .EQ. 0) GO TO 412
      WRITE(1,404)
 412  CONTINUE
      IF(IAND(IBUF(1),100000B) .EQ. 0) GO TO 94
1000  WRITE(1,740)
  94  CONTINUE
      IF(IAND(IBUF(2),1B).EQ. 0) GO TO 93
 212  WRITE(1,160)
  93  CONTINUE
      IF(IAND(IBUF(2),4B).EQ. 0) GO TO 95
 222  WRITE(1,165)
  95  CONTINUE
      IF(IAND(IBUF(2),10B).EQ. 0) GO TO 96
 219  WRITE(1,183)
  96  CONTINUE
      IF(IAND(IBUF(2),20B).EQ. 0) GO TO 99
 223  WRITE(1,170)
  99  CONTINUE
      IF(IAND(IBUF(2),40B).EQ. 0) GO TO 97
 224  WRITE(1,175)
  97  CONTINUE
      IF(IAND(IBUF(2),100B).EQ. 0) GO TO 7
 230  WRITE(1,182)
```

```
  7 CONTINUE
    IF(IAND(IBUF(2),2000B).EQ. 0) GO TO 8
231 WRITE(1,180)
  8 CONTINUE
    IF(IAND(IBUF(2),4000B).EQ. 0) GO TO 5
229 WRITE(1,181)
  5 CONTINUE
    IF(IAND(IBUF(2),10000B) .EQ. 0) GO TO 28
289 WRITE(1,288)
 28 CONTINUE
    IF(IAND(IBUF(2),20000B).EQ. 0) GO TO 6
    WRITE(1,176)
  6 CONTINUE
    IF(IAND(IBUF(3),1B).EQ. 0) GO TO 9
232 WRITE(1,290)
  9 CONTINUE
    IF(IAND(IBUF(3),10B).EQ. 0) GO TO 29
233 WRITE(1,184)
 29 CONTINUE
    IF(IAND(IBUF(4),1B).EQ. 0) GO TO 39
235 WRITE(1,351)
 39 CONTINUE
    IF(IAND(IBUF(4),2B).EQ. 0) GO TO 44
244 WRITE(1,354)
 44 CONTINUE
237 WRITE(1,250) IBUF(5)
136 CONTINUE
239 WRITE(1,241) IBUF(6)
137 CONTINUE

    ITRK = JLOF (JFLD(15,8),IBUF(7))
704 WRITE(1,320)  ITRK
751 CONTINUE
    ISCTR = JLOF (JFLD( 7,8),IBUF(7))
706 WRITE(1,323) ISCTR
777 WRITE(1,402) IBUF(9)
747 CONTINUE
236 WRITE(1,248) IBUF(10)
139 CONTINUE
    ANUM=FLOAT(IBUF(11))/100.
238 WRITE(1,247) ANUM
142 CONTINUE
    IF(IANS.NE.NO) IBUF(1) = IOR (IBUF(1),000001B)
    GO TO 10
500 WRITE(1,178)
    GO TO 1500
501 WRITE(1,178)
    GO TO 1501
502 WRITE(1,178)
    GO TO 1502
503 WRITE(1,178)
    GO TO 1503
504 WRITE(1,178)
    GO TO 1504
505 WRITE(1,178)
    GO TO 1505
506 WRITE(1,178)
    GO TO 1506
405 WRITE(1,178)
    GO TO 1405
710 WRITE(1,178)
    GO TO 1710
507 WRITE(1,178)
    GO TO 1507
508 WRITE(1,178)
```

```
      GO TO 1508
509   WRITE(1,176)
      GO TO 1509
510   WRITE(1,178)
      GO TO 1510
511   WRITE(1,178)
      GO TO 1511
512   WRITE(1,176)
      GO TO 1512
702   WRITE(1,178)
      GO TO 1702
513   WRITE(1,176)
      GO TO 1513
531   WRITE(1,178)
      GO TO 1531
514   WRITE(1,176)
      GO TO 1514
515   WRITE(1,178)
      GO TO 1515
526   WRITE(1,176)
      GO TO 1526
516   WRITE(1,178)
      GO TO 1516
517   WRITE(1,176)
      GO TO 1517
518   WRITE(1,178)
      GO TO 1518
519   WRITE(1,179)
      GO TO 1519
520   WRITE(1,179)
      GO TO 1520
521   WRITE(1,179)
      GO TO 1521
522   WRITE(1,179)
      GO TO 1522
406   WRITE(1,179)
      GO TO 1406
524   WRITE(1,179)
      GO TO 1524
525   WRITE(1,179)
      GO TO 1525
701   WRITE(1,176)
      GO TO 1701
227   FORMAT("DO YOU WANT OPTIONS REPORT")
 15   FORMAT(A2)
 16   FORMAT(F5.3)
 12   FORMAT("**OPTIONS FOR MARK**")
 13   FORMAT(/////,"ALPHA CHARACTERS?")
 25   FORMAT("TILT WANTED?")
 30   FORMAT("FLIP WANTED?")
 33   FORMAT("ROTATION ?")
 35   FORMAT("FLIP OR PACK PRINTING ?")
 38   FORMAT("PLACE SPLICE MARKS ?")
 40   FORMAT("NO OVERLAP IN PACK AND CHECK ?")
404   FORMAT("SIZES TO BE DISPLAYED ON LARGE PIECES")
 41   FORMAT("*OPTIONS FOR DRAW*")
 50   FORMAT(/////,"PLOT ALPHA CHARACTERS?")
 55   FORMAT("DRAW STYLE NUMBERS?")
 60   FORMAT("PLACE HEAD ON NOTCHES?")
 65   FORMAT("TICK MARKS TO OUTSIDE?")
 70   FORMAT("DRAW BOX AND CROSS FOR PUNCH HOLE")
 75   FORMAT("PRINT SIZE PER PIECE?",/)
 80   FORMAT("*OPTIONS FOR PLOTP*")
 81   FORMAT(////,"MODEL 2000 PLOTTER ?")
 82   FORMAT("MODEL 1100 PLOTTER ?")
```

```
  83 FORMAT("MODEL 622 CUTTER ?")
  85 FORMAT(/////,"GERBER DRUM PLOTTER?")
 101 FORMAT("*OPTIONS FOR GRADING*")
 102 FORMAT(////,"METERIC GRADE INCREMENTS ?",///)
 195 FORMAT("*ALL OPTIONS FOR GENERAL OPTIONS PROGRAM*")
 196 FORMAT(///,"LOCKED MARKERS ?")
 200 FORMAT("NO. OF SECTORS PER DIGITIZER FILE")
 205 FORMAT("NO. OF DIGITIZER FILES?")
 218 FORMAT("STARTING TRACK OF DIGITIZER ")
 224 FORMAT("STARTING SECTOR OF DIGITIZER")
 210 FORMAT("PAPER ADVANCE FACTOR ?")
 215 FORMAT("PLOT PAPER WIDTH ?")
 221 FORMAT(////,"*OPTIONS FOR MONITOR WPROD*")
 241 FORMAT("NUMBER OF DIGITIZER FILES IS",1X,I5)
 250 FORMAT("NUMBER OF SECTORS PER DIGITIZER FILE IS",1X,I5)
 123 FORMAT("ALPHA CHARACTERS SELECTED")
 130 FORMAT("LOCKED MARKERS SELECTED")
 133 FORMAT("TILT MARKERS SELECTED")
 135 FORMAT("FLIP MARKERS SELECTED")
 140 FORMAT("ROTATE MARKER SELECTED")
 141 FORMAT("FLIP OR PACK PRINTING SELECTED")
 150 FORMAT("PLACE SPLICE SELECTED")
 155 FORMAT("NO OVERLAP IN PACK/CHECK SELECTED")
 160 FORMAT("ALPHA CHARACTERS ON PLOT SELECTED")
 165 FORMAT("DRAW STYLE NUMBERS SELECTED")
 170 FORMAT("HEADS ON NOTCHES SELECTED")
 175 FORMAT("TICK MARKS TO OUTSIDE SELECTED")
 180 FORMAT("BOX AND CROSS FOR PUNCH HOLE SELECTED")
 181 FORMAT("PRINT SIZE PER PIECE SELECTED")
 182 FORMAT("MODEL 1100 PLOTTER SELECTED")
 183 FORMAT("MODEL 2000 PLOTTER SELECTED")
 184 FORMAT("MODEL 622 CUTTER SELECTED")
 296 FORMAT("GERBER DRUM PLOTTER SELECTED")
 351 FORMAT("METERIC GRADE INCREMENTS SELECTED")
 247 FORMAT("PAPER ADVANCE FACTOR IS",1X,F7.2)
 248 FORMAT("PLOTTER PAPER WIDTH IS",1X,I5)
 320 FORMAT("STARTING TRACK= DIGITIZER IS",1X,I5)
 325 FORMAT("STARTING SECTOR=DIGITIZER IS",1X,I5)
 103 FORMAT("TRACE PROGRAM FLOW ?")
 354 FORMAT("PROGRAM FLOW TRACER SELECTED")
 288 FORMAT("DRAW MARKER AT 1/2 UNIT SCALE")
 740 FORMAT("MULTI SCOPE OPTION PRESENT  ")
 178 FORMAT("***ERR#1. ANSWER IS NOT A YES OR NO***")
 179 FORMAT("***ERR#2. ANSWER IS NOT A NUMERIC VALUE***")
 402 FORMAT("DEFAULT SAMPLE SIZE IS",1X,I4)
 400 FORMAT("DEFAULT SAMPLE SIZE ?")
  10 CONTINUE
     CALL UDSC(0)
     CALL EXEC(15,66,IBUF,128,MARKA,0)
     END
     ENDS


FTN4
      PROGRAM PSORT
C
C     VERSION 2.1
C     EDITED 24 FEB 73
C     RONALD GORDON
C
      DIMENSION IWORK(1)
      INTEGER BUFSIZ
      INTEGER ROSTER(5),PASS,FLAG3,OPNIR
      INTEGER PGRAB(3), PGRAD(3), WSTYL(3), CHEK(3)
      LOGICAL FLAG1, FLAG2
```

```
      EQUIVALENCE (ROSTER(1),PASS),
     +            (ROSTER(2),DPNTR),
     +            (ROSTER(3),FLAG1),
     +            (ROSTER(4),FLAG2),
     +            (ROSTER(5),FLAG3)
      DATA WSTYL /2HWS,2HTY,1HL/,
     +     PGRAD /2HPG,2HRA,1HD/,
     +     PGRAB /2HPG,2HRA,1HB/,
     +     CHEK  /2HCH,2HEK,1H /
C
C     GET INPUT PARAMETERS
C
      CALL RMPAR(ROSTER)
      CALL ALLOC (IWORK(0),BUFSIZ)
      CALL QDSC (0)
      CALL EXEC (14,1028,IWORK(1),DPNTR,CHEK,0)
      CALL SORTP (IWORK,DPNTR)
      CALL EXEC (15,1028,IWORK(1),DPNTR,CHEK,0)
      IF (.NOT.FLAG2)
     +     CALL EXEC (10,PGRAD,PASS,DPNTR,FLAG1,FLAG2,FLAG3)
      IF (FLAG1) GO TO 180
  150 PASS=PASS+1
      CALL EXEC (10,PGRAB,PASS,DPNTR,FLAG1,FLAG2,FLAG3)
  180 IF (FLAG3.NE.1) GO TO 150
      CALL EXEC (10,WSTYL,DPNTR)
      END
END$


FTN4
      SUBROUTINE QLIST (JFLAG)
      DIMENSION NAME(5), LAZYB(5)
      DIMENSION BAME(6)
      DATA IO6 /6/
      M14C4=JFLD(14,4)
      M15C6=JFLD(15,6)
      M15C1=JFLD(15,1)
      M14C1=JFLD(14,1)
      M14C4=JFLD(14,4)
      M10C3=JFLD(10,3)
      M12C5=JFLD(12,5)
      M13C1=JFLD(13,1)
      MASKR=JFLD(7,8)
      MASKL=JFLD(15,8)
      DO 10 I=0,4,1
   10 LAZYB(I+1)=JFLD(I,1)
C
C**********REPORT FOR FILE QUEUE ENTRIES
C
      CALL WOST(2,LQ)
      LQ=LQ-2
      IF (JLOF(LAZYB(1),JFLAG),EQ,0) GO TO 200
      WRITE (IO6,1110)
 1110 FORMAT(//,"F I L E   Q U E U E")
      DO 190 IQ=7,LQ,2
      CALL WOST (IQ,IQWORD)
      ITYPE=JLOF(M14C4,IQWORD)
      IF (ITYPE.NE.6) GO TO 190
      IPRIO=JLOF(M10C3,IQWORD)
      CALL WOST (IQ+1,IQWORD)
      IF (IQWORD.LT.256) GO TO 190
      WRITE (IO6,1100) IPRIO
 1100 FORMAT(/,"FILE:  P=",I1)
      IQWORD=IQWORD+1
      CALL WOST (IQWORD,NPIEC)
```

```
      DO 130 I=1,NFILE,1
      DO 120 J=1,5,1
      IQWORD=IQ*ORD+1
      CALL RQST (IQWORD,NAME(J))
  120 CONTINUE
  130 WRITE (IO6,1200) NAME
  190 CONTINUE
C
C**********REPORT FOR NESTED PLOT ENTRY
C
  200 CONTINUE
      IF (JLOF(LAZYB(2),JFLAG),EQ,0) GO TO 300
      WRITE (IO6,1310)
 1310 FORMAT(//,"P L O T   Q U E U E")
      DO 290 IG=7,LQ,2
      CALL RQST (IG,IQWORD)
      ITYPE=JLOF(M14C4,IQWORD)
      IF (ITYPE.NE.7) GO TO 290
      IPRIO=JLOF(M14C3,IQWORD)
      CALL RQST (IG+1,IQWORD)
      IF (IQ*ORD.LT.256) GO TO 290
      WRITE (IO6,1320) IPRIO
 1320 FORMAT(/,"PLOT:  P=",I1)
      DO 210 I=1,5,1
      IQ*ORD=IQWORD+1
  210 CALL RQST (IQWORD,NAME(I))
      IQ*ORD=IQWORD+1
      CALL RQST (IQWORD,ITEMP)
      RS=FLOAT(ITEMP)*0.01
      WRITE (IO5,1450) NAME,RS
 1450 FORMAT(5X,5A2," SCALE=",F4,2)
      IQ*ORD=IQ*ORD+1
      CALL RQST (IQ*ORD,ITEMP)
      NSIZE=JLOF(MASKB,ITEMP)
      IRULE=JLOF(M13C8,ITEMP)
      INEST=JLOF(M15C1,ITEMP)
      ICUTR=JLOF(M14C1,ITEMP)
      IF (INEST.EQ.1) WRITE (IO6,1500) IRULE
      IF (ICUTR.EQ.1) WRITE (IO6,1550) IRULE
 1500 FORMAT(5X,"RULE=",I3," NEST!")
 1550 FORMAT(5X,"RULE=",I3," CUT!")
      DO 230 I=1,NSIZE,1
      DO 240 J=1,3,1
      IQWORD=IQ*ORD+1
  240 CALL RQST (IQWORD,NAME(J))
  230 WRITE (IO5,1200) (NAME(J), J=1,3,1)
  290 CONTINUE
C
C**********REPORT FOR DRAW QUEUE ENTRIES
C
  300 CONTNIUE
      IF (JLOF(LAZYB(3),JFLAG),EQ,0) GO TO 400
      WRITE (IO5,1710)
 1710 FORMAT(//"D R A W   Q U E U E")
      DO 390 IG=7,LQ,2
      CALL RQST (IQ,IQ*ORD)
      ITYPE=JLOF(M14C4,IQWORD)
      IF (ITYPE.NE.6) GO TO 390
      IPRIO=JLOF(M14C3,IQ*ORD)
      CALL RQST (IQ+1,IQ*ORD)
      IF (IQ*ORD.LT.256) GO TO 390
      WRITE (IO6,1700) IPRIO
 1700 FORMAT(/,"DRAW:  P=",I1)
      DO 310 I=1,5,1
      IQ*ORD=IQWORD+1
```

```
  310 CALL WGST(IGWORD,NAME(I))
      IGWORD=IGWORD+1
      CALL WGST (IGWORD,ITEMP)
      RS=FLOAT(ITEMP)*0.01
      IGWORD=IGWORD+1
      CALL WGST (IGWORD,ITEMP)
      NPLOT=JLOF(MASKL,ITEMP)
      WRITE (IO6,1800) NAME,RS,NPLOT
 1800 FORMAT(5X,5A2," SCALE=",F4.2," RPT=",I1)
      NMSGS=JLOF(MASKR,ITEMP)
      DO 320 I=1,NMSGS,1
      CALL WGST (IGWORD+I,ITEMP)
      ITEMP=ITEMP/100
  320 WRITE (IO6,1900) ITEMP
  390 CONTINUE
 1900 FORMAT(5X,"MSG:",I4)
C
C**********REPORT FOR MARK ENTRIES
C
  400 CONTINUE
      IF (JLOF(LAZYB(4),JFLAG).EQ.0) GO TO 500
      WRITE (IO6,2010)
 2010 FORMAT(//,"M A R K   Q U E U E")
      DO 490 IG=7,LG,2
      CALL WGST (IG,IGWORD)
      ITYPE=JLOF(M14C4,IGWORD)
      IF (ITYPE.NE.5) GO TO 490
      IPRIO=JLOF(M14C3,IGWORD)
      CALL WGST (IG+1,IGWORD)
      IF (IGWORD.LT.256) GO TO 490
      WRITE (IO6,2000) IPRIO
 2000 FORMAT(/,"MARK:  P=",I1)
      DO 410 I=1,5,1
      IGWORD=IGWORD+1
  410 CALL WGST (IGWORD,NAME(I))
      WRITE (IO6,1200) NAME
      CALL WGST (IGWORD+1,ITEMP)
      WIDTH=ITEMP*0.01
      CALL WGST (IGWORD+2,ITEMP)
      EFFIC=ITEMP*0.01
      CALL WGST (IGWORD+3,ITEMP)
      ISHAD=JLOF(M12C5,ITEMP)
      NSTYL=JLOF(MASKR,ITEMP)
      OLDUN=JLOF(M13C1,ITEMP)
      CALL WGST (IGWORD+4,ITEMP)
      PCNTX=ITEMP*0.01
      CALL WGST (IGWORD+5,ITEMP)
      PCNTY=ITEMP*0.01
      CALL WGST (IGWORD+6,ITEMP)
      STRIPE=ITEMP*0.01
      CALL WGST (IGWORD+7,ITEMP)
      PLAID=ITEMP*0.01
      IGWORD=IGWORD+7
      WRITE (IO6,2100) WIDTH,EFFIC,ISHAD,PCNTX,PCNTY,STRIPE,PLAID
 2100 FORMAT(5X,"WIDTH=",F6.2," EFFIC=",F6.2,/,5X,"SHADE=",I1," %X=",
     +  F4.2," %Y=",F4.2,/,5X,"STRIPE=",F6.2," PLAID=",F6.2)
      IF (OLDUN.NE.0) GO TO 450
      DO 440 I=1,NSTYL,1
      DO 420 J=1,5,1
      IGWORD=IGWORD+1
  420 CALL WGST (IGWORD,NAME(J))
      WRITE (IO6,2200) NAME
 2200 FORMAT(4X,"*",5A2)
      IGWORD=IGWORD+1
      CALL WGST (IGWORD,ITEMP)
```

```
      NSIZE=JLOF(MASKR,ITEMP)
      DO 440 J=1,NSIZE,1
      DO 430 K=1,3,1
      IQWORD=IQWORD+1
  430 CALL WQST (IQWORD,NAME(K))
  440 WRITE (IO6,2250) (NAME(K), K=1,3,1)
 2250 FORMAT(8X,3A2)
      GO TO 490
  450 DO 460 I=1,5,1
      IQWORD=IQWORD+1
  460 CALL WQST (IQWORD,NAME(I))
      WRITE (IO6,3100) NAME
 3100 FORMAT(5X,"OLD ",5A2)
      IF (NSTYL.EQ.0) GO TO 490
      DO 475 I=1,NSTYL,1
      DO 470 J=1,5,1
      IQWORD=IQWORD+1
  470 CALL WQST (IQWORD,NAME(J))
      WRITE (IO6,3200) NAME
 3200 FORMAT(8X,"REPLACE ",5A2)
      IQWORD=IQWORD+1
      CALL WQST (IQWORD,ITEMP)
      LSIZE=JLOF(MASKL,ITEMP)/2
      DO 475 J=1,LSIZE,1
      DO 473 K=1,6,1
      IQWORD=IQWORD+1
  473 CALL WQST (IQWORD,NAME(K))
  475 WRITE (IO6,3300) NAME
 3300 FORMAT(8X,3A2," <= ",3A2)
  490 CONTINUE
C
C**********DO STYLE QUEUE REPORT
C
  500 CONTINUE
      IF (JLOF(LAZYR(5),JFLAG),EQ,0) GO TO 600
      WRITE (IO6,2310)
 2310 FORMAT(//,"S T Y L E   Q U E U E")
      DO 590 IQ=7,LQ,2
      CALL WQST (IQ,IQWORD)
      ITYPE=JLOF(M14C4,IQWORD)
      IF (ITYPE.NE.4) GO TO 590
      IPRIO=JLOF(M10C3,IQWORD)
      CALL WQST (IQ+1,IQWORD)
      IF (IQWORD.LT.256) GO TO 590
      WRITE (IO6,2300) IPRIO
 2300 FORMAT(/,"STYLE:  P=",I1)
      DO 510 I=1,5,1
      IQWORD=IQWORD+1
  510 CALL WQST (IQWORD,NAME(I))
      IQWORD=IQWORD+1
      CALL WQST (IQWORD,ITEMP)
      IRULE=JLOF(MASKL,ITEMP)
      NPIEC=JLOF(MASKR,ITEMP)
      WRITE (IO6,2400) NAME,IRULE
 2400 FORMAT(5X,5A2," RULE=",I3)
      WRITE (IO6,3000)
      DO 550 I=1,NPIEC,1
      DO 520 J=1,5,1
      IQWORD=IQWORD+1
  520 CALL WQST (IQWORD,NAME(J))
      IQWORD=IQWORD+1
      CALL WQST (IQWORD,ITEMP)
      IFLIP=JLOF(M15C1,ITEMP)
      ISTRP=JLOF(M14C1,ITEMP)
      IBLCK=JLOF(M12C5,ITEMP)
```

```
      NREPT=JLOF(MASKR,ITEMP)
      IGWORD=IGWORD+1
      CALL WOST (IGWORD,ITEMP)
      PCNTX=ITEMP*M.M1
      IGWORD=IGWORD+1
      PCNTY=ITEMP*M.M1
  530 WRITE (IO6,1200) NAME,IFLIP,ISTRP,IBLCK,PCNTX,PCNTY,NREPT
 1200 FORMAT(5X,5A2,3I7,2F6.2,I3)
 3000 FORMAT(5X,"NAME",1ØX,"FLIP  BLOCK STRIPE  %X",4X,"%Y  RPT")
  590 CONTINUE
  600 CONTINUE
      RETURN
      END
END$


FTN4
      SUBROUTINE ROTAL (IMGNO)
C..........DEC. 7, 1972
C..........MARGARET ROMAN
C..........PROGRAMS: MARK
C..........VERSION 2.0
C..........ROUTINE TO ROTATE ALL PIECES OF ONE SIZE, THE
C..........SIZE OF THE PIECE DENOTED BY "IMGNO".
      COMMON MOUT,LABF(3),LABR(3),NOSTY,INSTY(31),JBS(15),NAMST(3),
     1 MKUP(5),NSCMD,JBM(20),NAMMX(3),JBG(4),NAMGD(3),JBP(10),
     2 IOMK(5),IDOP,NPEC,NSHD,FACTR,NSIZ,ISIZ(42),KIDTH,EFF,SX,SY,KSTL,
     3 YARDS,MAP,KINT,KFLG,KINC,KSIZ,KXMN,KYMN,KANG,KXMX,KYMX,KOPT,
     4 KAD,KADP,KDF,CSMX,MAKD,CSMN,NRGE,IMGP,KREL,ITYPE,
     5 NOIMG,IOPSZ,IMGY,IMGE,IMGTC,IMKD1,IMKD2,IMENU,IOXP,IOYP,
     6 IXC,IYC,IXL,IXR,IYU,IYL,CS,MDELX,MDELY,JFLG,MLW,MUP,ISLX,ISLY,
     7 TOTAL,MYL,XPC,YPC,IMGLP,IMESG,IEDG,ISOFF,SOFF,KSLH,
     8 IMGPS,MAXOF,KSLV,KTC,IOPTN,IAREA,ISIZE,KCSR,KYES,KPSIZ,KPLL,
     9 IT1,IT2,IT3,IT4,IT5,IT6,IT7,IT8,IT9,IT10,
     A IOFSET,KPLM,IFLP,ICUS1,IMGPK,
     B T1,T2,T3,T4,T5,T6,T7,T8,T9,T10,
     C JSTY,JFS,JNG,JSHD,JKT,
     D ILPS,IBG,IHD,IOV,IGR,IBL,ITCW,ITCCW,
     E IRCW,IRCCW,ISL,IRE,IFL,IPK,ICUM,
     F IDUM(1)
C..........COMPUTE UNIQUE SIZE NUMBER OF PIECE PICKED
      KSIZE = IDUM(KSIZ+IMGNO)
C..........ROTATE ALL PIECES IN THAT SIZE BY 180 DEGREES
      DO 10 I=1,NOIMG
      IF(IDUM(KSIZ+I).NE.KSIZE) GO TO 10
C..........PIECE IN SIZE
C..........IGNOR SLAVE PIECES
      IF(JLOF(JBM(20),IDUM(KFLG+I)),EQ.1) GO TO 10
      IF(JLOF(JNG,IDUM(KFLG+I)),EQ.0) GO TO 10
C..........PIECE ALSO IN NAP GROUP
      JANG = KANG+I
      NANG = ISIGN(1800,IDUM(JANG))
      ITMP = MOD(IDUM(JANG),3600)
      IF(IABS(ITMP).GE.1800) NANG = -NANG
      IDUM(JANG)=IDUM(JANG)+NANG
C..........ADJUST STRIPE LINE
      IF(ISLY.EQ.0) GO TO 5
      IF(IDUM(KSTL+I).EQ.9999) GO TO 5
      IDUM(KSTL+I) = -IDUM(KSTL+I)
C..........ADJUST PLAID LINE
    5 IF(ISLX.EQ.0) GO TO 7
      IF(IDUM(KPLL+I).EQ.9999) GO TO 7
      IDUM(KPLL+I) = -IDUM(KPLL+I)
C..........REDRAW PIECE AT NEW ANGLE
    7 CALL VISIB(I,IORG,JX,JY)
```

```
C.........PIECE VISIBLE?
      JORG = 2
      IF(IORG.NE.0) JORG = IORG
      CALL DRAWI(JX,JY,JORG,I,ISTAT)
      IF(IORG.NE.0) CALL ORGIN(JX,JY,JORG,1)
   30 IF(IABS(IDUM(KANG+I)).LT.3600) GO TO 25
C.........FLIPPED FROM FUNCTION BOX
      NN=0
      IF(JLOF(JBM(15),IOPTN).NE.0)NN=NN+3
      IF(JLOF(JBM(15),IDUM(KFLG+NPEC+I)).NE.0) NN=NN+4
      IAD = IABS(IDUM(KADP+I))
      CALL FLIP(IDUM(IAD+NN))
      CALL ORGIN(JX,JY,JORG+10,I)
   25 IF(IORG.EQ.0) GO TO 26
      CALL MOVI (2,JX,JY,JORG,I)
   26 CALL LITE(1,KADP+I)
   10 CONTINUE
      MDELX = IDUM(KXMX+IMGNO)-IDUM(KXMN+IMGNO)
      MDELY = IDUM(KYMX+IMGNO)-IDUM(KYMN+IMGNO)
      RETURN
      END
      ENDS


FTN4
      SUBROUTINE RTM
C.........ROUTINE TO RETURN TO MONITER WITH NOT READY STATIS
      DIMENSION IPROG(3),IFILE(3),IBUF(2)
      DATA IPROG/2HWP,2HRO,2HD /
      DATA IFILE/2HSE,2HLT,2HR /
      DATA IBUF/2,2/
C.........TEMPORARY
      IF(ISSW(0).LT.0) STOP
      CALL UDSC(0)
      CALL EXEC (15,66,IBUF,2,IFILE,0)
      CALL EXEC (10,IPROG)
      END
      ENDS


FTN4
      SUBROUTINE RTMO
C.........ROUTINE TO RETURN TO MONITER WITH NOT READY STATUS
      DIMENSION IPROG(3),IFILE(3),IBUF(2),IPLAT(3)
      DATA IPROG/2HWP,2HRO,2HD /
      DATA IFILE/2HSE,2HLT,2HR /
      DATA IBUF/2,5/
      DATA IPLAT/2HPN,2H00,2H00/
      CALL EXEC (23,IPLAT)
      CALL EXEC (15,66,IBUF,2,IFILE,0)
      CALL EXEC (10,IPROG)
      END
      ENDS


FTN4
      SUBROUTINE SCALE(IX,IY,X,Y,N)
C.......N=1  SET UP SCALE FACTORS FOR DISPLAY
C.......N=2  SCALE FROM REAL UNITS TO RASTER UNITS
C.......N=3  SCALE FROM RASTER UNITS TO REAL UNITS
C
C
C
      DATA RSTR/.003663/
      GO TO (100,200,300),N
C.......SET UP SCALE FACTORS
```

```
  100 SCALX = X/15.0
      SCALY = Y/15.0
      IOFX = IX
      IOFY = IY
      RETURN
C.....REAL TO RASTER CONVERSION
  200 IX = IRND(X/(SCALX*RSTR))+IOFX
      IY = IRND(Y/(SCALY*RSTR))+IOFY
      RETURN
C.....RASTER TO READ CONVERSION
  300 X = FLOAT(IX-IOFX)*SCALX*RSTR
      Y = FLOAT(IY-IOFY)*SCALY*RSTR
      RETURN
      END
      ENDS


FTN4
      SUBROUTINE SHADE(IMGL)
C.....FEB. 9,1973
C.....MARGARET ROWAN
C.....PROGRAM MARK
C.....VERSION 2.0
C.....ROUTINE TO SET SHADE ZONE FOR ALL PIECES OF THAT SIZE
C.....FLAGGED FOR SHADE CONSTRAINTS
      COMMON MOUT,LABF(3),LABR(3),NOSTY,INSTY(31),JRS(15),NAMST(3),
     1 MKOP(5),NSCMD,JRM(20),NAMMK(3),JRG(4),NAMGO(3),JRP(10),
     2 IDMK(5),IDOP,NPEC,NSHD,FACTR,NSIZ,ISIZ(42),MIOTM,EFF,SX,SY,KSIL,
     3 YARDS,NAP,KINT,KFLG,KINC,KSIZ,KXMN,KYMN,KANG,KXMX,KYMX,KOPT,
     4 KAD,KADP,KOF,CSMX,MKWD,CSMN,NAGE,IMGP,KREL,ITYPE,
     5 NOIMG,IOFSZ,IMGY,IMGE,IMGTC,IMKO1,IMKO2,IMEND,IOXP,IOYP,
     6 IXC,IYC,IXL,IXR,IYU,IYL,CS,MDELX,MDELY,JFLG,MLW,MUP,ISLX,ISLY,
     7 TOTAL,NYL,XPC,YPC,IMGLP,IMESG,IBUG,ISOFF,SOFF,KSLH,
     8 IMGFS,MAXOF,KSLV,KTC,IOPTN,IAREA,ISIZE,KOSP,KYES,KPSIZ,KPLL,
     9 IT1,IT2,IT3,IT4,IT5,IT6,IT7,IT8,IT9,IT10,
     A IOFSET,KPLH,IFLP,ICUST,IMGPK,
     B T1,T2,T3,T4,T5,T6,T7,T8,T9,T10,
     C JSTY,JFG,JNG,JSHD,JRT,
     D ILPD,IBU,IBD,IOV,IBR,IBL,ITCW,ITCCW,
     E IRCW,IRCCW,ISL,IRE,IFL,IPK,ICOM,
     F IDUM(1)
      IMG = IMGP
      IF(IMGP.EQ.0) IMG = IMGL
C.....PIECE IN SHADE GROUP?
      IF(JLOF(JBM(14),IDUM(KFLG+IMG)),EQ.0) RETURN
C.....PIECE HAVE ZONE SET?
      IF(JLOF(JSHD,IDUM(KFLG+IMG)).NE.0) RETURN
C.....FIND CENTER OF PIECE IN Y
      IYYC=(IDUM(KYMN+IMG)+IDUM(KYMX+IMG))/2
C.....WHAT ZONE IS IT IN?
      INCSH = MKWD/NSHD
      ISHZ = 1
      ITOP = MKWD/2
   10 IBOT = ITOP-INCSH
      IF(ITOP.GT.IYYC.AND.IBOT.LE.IYYC) GO TO 20
      ISHZ = ISHZ+1
      ITOP = IBOT
      IF(ITOP.GT.-(MKWD-MKWD/2)) GO TO 10
C.....PIECE NOT IN MARKER
      RETURN
C.....FIND PIECES IN SIZE GROUP WITH SHADE CONSTRAINT AND SET ZONE
   20 ISZE = IDUM(KSIZ+IMG)
      DO 50 I=1,NOIMG
      IF(IDUM(KSIZ+I).NE.ISZE) GO TO 50
      IF(JLOF(JBM(14),IDUM(KFLG+I)).EQ.0) GO TO 50
```

```
      CALL JSTF(ISHZ,JSHD,IDUM(KFLG+I))
   50 CONTINUE
      RETURN
      END
      ENDS


FTN4
      SUBROUTINE SIZEP(N,IFILE,INAME,IPTR)
C.....ROUTINE TO CONVERT THE 6 ASCKII CHARACTER S
C.....SIZE NAME OF INAME TO THE SIZE/RELATION POI
C..... POINTER FOR THAT SIZE IN RULE TABLE IFILE
C.....  OR THE REVERSE  FROM POINTER TO NAME
      DIMENSION IFNA(3),INAME(3),IBUF(128)
      DIMENSION INAM(6)
      DATA NOLD/0/,ISCTR/0/,NSCTR/0/
      DATA IFNA/2HRD,2HLE,2HS /
C.....IF TABLE IN CORE BYPASS INITIALIZATION
      IF(IFILE.EQ.NOLD) GO TO 30
      NOLD=IFILE
      INAM(6)=IFILE
      CALL DIREC(3HOOGB,INAM,NRT,NSCTR,ISTAT)
      IF(ISTAT.EQ.0) GO TO 20
      IPTR=0
      NOLD=0
      IF (N.NE.1) INAME=0
      RETURN
20    CALL EXEC(14,66,IBUF,128,IFNA,NSCTR)
      NS=JFLD(9,10)
      NOS=JLOF(NS,IBUF(2))+1
      NTSEC=(NOS-1)/42+1
      ICURS=0
C.....IF N=1 GO FROM NAME TO POINTER
30    IF(N.EQ.1)GO TO 40
C.....IF N .NE1 GO FROM POINTER TO NAME
      IF(IPTR.LE.0.OR.IPTR.GT.NOS) GO TO 35
      NPTR=IPTR+1
C.....ADJUST INDEX TO BUFFER IF NEEDED, MAX OF 20
C.....          SECTORS OF SIZE RELATION TABLE
C.....             OR 840 SIZES
      DO 31 KJ=1,20
      IF(NPTR.LE.42) GO TO 32
      NPTR=NPTR-42
31    CONTINUE
      GO TO 35
C.....  CALCULATE REQUIRED SECTOR AND CHECK TO SE
C.....    IF IN CORE
32    IRES=KJ-1
      IF(IRES.EQ.ICURS) GO TO 33
      ICURS = IRES
      ISCTR=NSCTR+ICURS
      CALL DOSC (0)
      CALL EXEC(14,66,IBUF,128,IFNA,ISCTR)
33    J=NPTR*3
      INAME(1)=IBUF(J)
      INAME(2)=IBUF(J+1)
      INAME(3)=IBUF(J+2)
      RETURN
   35 INAME=0
      RETURN
40    ISCTR=NSCTR
      CALL DOSC(0)
      DO 50 KI=1,NTSEC
      IF(ICURS.EQ.ISCTR-NSCTR) GO TO 41
      CALL EXEC(14,66,IBUF,128,IFNA,ISCTR)
```

```
41      ISCTR=ISCTR+1
        ICURS = KI-1
        DO 50 I=1,42
        J=I*3
        IF(INAME(1).EQ.IBUF(J).AND.INAME(2).EQ.IBUF(J+
       11).AND.INAME(3).EQ.IBUF(J+2)) GO TO 60
50      CONTINUE
        IPTR=-NOS
        RETURN
60      IPTR=I+(KI-1)*42
        IPTR=I+(KI-1)*42-1
        END
        ENDS


FTN4
        SUBROUTINE SPLIC (NSP,IX1,IX2)
C..........JAN. 22, 1973
C..........MARGARET ROMAN
C..........VERSION 2.0
C..........ROUTINE TO REQUEST USER TO PLACE UP TO 15 SPLICE MARKS
        DIMENSION IX1(15),IX2(15),ITEX(9)
        COMMON NOUT,LABR(3),LABR(3),NUSTY,INSTY(31),JBS(15),NAMST(3),
       1 MKUP(5),NSCMD,JBM(20),NAMMK(3),JBG(4),NAMGD(3),JBP(12),
       2 IDMK(5),IJOP,NPEC,NSMD,FACTR,NSIZ,ISIZ(42),MIDTM,EFF,SX,SY,KSIZ,
       3 YARDS,NAP,KJMT,KFLG,KINC,KSIZ,KXMN,KYMN,KANG,KXMX,KYMX,KOPT,
       4 KAD,KADP,KDF,CSMX,MKND,CSMN,DRGE,IMGP,KREL,ITYPE,
       5 NOIMG,IOFS7,IMGY,IMGE,IMGTC,IMK01,IMK02,IMEND,IDXP,IDYP,
       6 IXC,IYC,IXL,IXR,IYU,IYL,CS,MDELX,MDELY,JFLG,MLW,MUP,ISLX,ISLY,
       7 TOTAL,MYL,XPC,YPC,IMGLP,IMESG,IEOG,ISOFF,SOFF,KSLM,
       8 IMGPS,MAXOF,KSLV,KTC,IOPTN,IAREA,ISIZE,KCSR,KYES,KPSIZ,KPLL,
       9 IT1,IT2,IT3,IT4,IT5,IT6,IT7,IT8,IT9,IT10,
       A IOFSET,KPLM,IFLP,ICUST,IMGPK,
       B T1,T2,T3,T4,T5,T6,T7,T8,T9,T10,
       C JSTY,JFG,JBG,JSMD,JRT,
       D ILPB,IRD,IRU,IDV,IBR,IRL,ITCW,ITCCW,
       E IRCW,IRCCW,ISL,IRE,IFL,IPK,ICOM,
       F IDUM(1)
        DATA ITEX/2HPL,2HAC,2HE ,2HSP,2HLI,2HCE,2H M,2HAR,2HKS /
        IREB = 6.0/FACTR
        NSP = 0
        IDX = 1
        JB1 = JFLD(1,1)
C..........DISPLAY "PLACE SPLICE MARKS" IN MESAGE AREA
        M = KAD+IMESG
        IAD = IABS(IDUM(M))+3
        DO 10 I=1,9
   10   IDUM(IAD+I) = ITEX(I)
        IDUM(IAD+10) = 120006
        CALL LITE(1,M)
C..........ZERO LIGHT PEN BUTTONS AREA AND SIZE OF PICKED PIECE
C..........AREA FOR SPLICE "X"
        IA1 = IABS(IDUM(KAD+IMGPS))
        IA2 = IABS(IDUM(M))-1
        DO 15 I=IA1,IA2
   15   IDUM(I) = 0
        IDUM(KAD+IMGPS) = IA1
        IDUM(KAD+IMGLP) = IA1
C..........SCALE WINDOW SO AS TO VIEW ENTIRE MARKER
        LNGH = (11*36.0)/FACTR-32600.0+100.0
        CSN = 4096.0/(FLOAT(LNGH)+3270.0)
        IF(CSN.GT.CSMX) CSN=CSMX
        IF(CSN.GT.CS) GO TO 20
        CS = CSN
        IXC = -32700.0+2048.0/CS
```

```
      IYC = MYL+2048.0/CS
      CALL WINDO
C.....TRACK THE CURSOR
   20 CALL DIHIT(ISWD,LX,LY)
      LX = MAX0(LX,-1964)
      LX = MIN0(LX,1964)
      LY = MAX0(LY,-1964)
      LY = MIN0(LY,1964)
      CALL CROSS (LX,LY,KTC,IDUM)
C.....CHECK FOR PRESSURE
      IF(JLOF(JR1,ISWD).EQ.1) GO TO 25
C.....READ FUNCTION BOX
      CALL FUNCB(ITYP)
C.....QUIT IF "RELEASE" IS HIT
      IF(ITYP.EQ.IRE) GO TO 500
      GO TO 20
C.....PEN PRESSURE SETS SPLICE MARK
   25 IXX = IFIX(FLOAT(LX)/CS)+IXC
C.....DISPLAY AN "X" FOR THE MARK
      IDUM(KAD+IMGPS) = -IA1
      IDUM(KAD+IMGLP) = -IA1
      IDUM(IA1) = 100000B
      IF(NSP.GT.0) GO TO 30
C.....FIRST "X" SET Y REGISTER
      IDUM(IA1+IDX) = 40V24B
      IDUM(IA1+IDX+1) = LX*16
      NY = FLOAT(MKWD/2-IYC)*CS
      IDUM(IA1+IDX+2) = NY*16+1
      IDUM(IA1+IDX+3) = 10117B
      IDUM(IA1+IDX+4) = 54V24B
      IDX = IDX+5
      GO TO 40
C.....NO NEED TO SET Y REGISTER AGAIN
   30 IDUM(IA1+IDX) = 40V24B
      IDUM(IA1+IDX+1) = LX*16+1
      IDUM(IA1+IDX+2) = 10117B
      IDUM(IA1+IDX+3) = 54024B
      IDX = IDX+4
   40 IDUM(IA1-1) = 0
      IDUM(KAD+IMGPS) = IA1
      IDUM(KAD+IMGPL) = IA1
      NSP = NSP+1
C.....COMPUTE SPLICE MARK AREA
      IXLF = IXX+IBF6
      IXRT = IXX-IBF6
C.....SET UP 12 INCH BUFFER AREA
      SXLF = FLOAT(IXX-IBF6)
      SXRT = FLOAT(IXX+IBF6)
      DO 200 I=1,NOIMG
      IF(IDUM(KYMX+I).GT.MKWD/2) GO TO 200
C.....PIECE IS IN MARKER
      PXLF = FLOAT(IDUM(KXMN+I))
      PXRT = FLOAT(IDUM(KXMX+I))
      IF(PXLF.GT.SXRT.OR.PXLF.LT.SXLF) GO TO 100
C.....LEFT EDGE IN BUFFER AREA
      IF(PXRT.LE.SXRT.AND.PXRT.GE.SXLF) GO TO 200
C.....ENTIRE PIECE IS NOT IN AREA, USE FOR LEFT EDGE
      IXLF = MIN0(IXLF,IDUM(KXMN+I))
      GO TO 200
  100 IF(PXRT.LT.SXRT.OR.PXRT.GT.SXLF) GO TO 200
C.....RIGHT EDGE IN BUFFER AREA, USE FOR RIGHT EDGE
      IXRT = MAX0(IXRT,IDUM(KXMX+I))
  200 CONTINUE
C.....STORE SPLICE LIMITS
      IA1(NSP) = IXLF
```

```
      IX2(NSP) = IXRT
C.....QUIT IF FIFTEENTH SPLICE MARK
      IF(NSP.LT.15) GO TO 20
  500 RETURN
      END
      ENDS


FTN4
      SUBROUTINE SPEED
C.....ROUTINE TO POSITION PIECES ABOVE THE MATERIAL
C
C
C
C
C
      COMMON MOUT,LABF(3),LABR(3),NOSTY,INSTY(31),JBS(15),NAMST(3),
     1 MKUP(5),NSCMD,JBM(20),NAMMK(3),JBG(4),NAMGD(3),JBP(12),
     2 IDMK(5),IDUP,NPEC,NSHD,FACTR,NSIZ,ISIZ(42),WIDTH,EFF,SX,SY,KSTL,
     3 YARDS,MAP,KIN1,KFLG,KINC,KSIZ,KXMN,KYMN,KANG,KXMX,KYMX,KOPT,
     4 KAD,KADP,KOF,CSMX,MKWD,CSMN,NMGE,IMGP,KREL,ITYPE,
     5 NOING,IOFSZ,IMGY,IMGE,IMGTC,IMK01,IMK02,IMEND,IDXP,IDYP,
     6 IXC,IYC,IXL,IXR,IYU,IYL,CS,MDELX,MDELY,JFLG,MLW,MUP,ISLX,ISLY,
     7 TOTAL,MYL,XPC,YPC,IMGLP,IMESG,IEDG,ISOFF,SOFF,KSLH,
     8 IMGPS,MAXDS,KSLV,KTC,IOPTN,IAREA,ISIZE,KCSR,KYES,KPSIZ,KPLL,
     9 IT1,IT2,IT3,IT4,IT5,IT6,IT7,IT8,IT9,IT10,
     A IOFSET,KPLM,IPLP,ICOST,IMGPK,
     B T1,T2,T3,T4,T5,T6,T7,T8,T9,T10,
     C JSTY,JFG,JNG,JSHD,JRT,
     D ILPS,ISD,IRD,IOV,IBR,IBL,ITCW,ITCCW,
     E IRCW,IRCCW,ISL,IPE,IFL,IPK,ICOM,
     F IDUM(1)
      FACT = 1.2
      IF(MKUP.NE.2HFR) GO TO105
   60 LNGHX = FACT*FLOAT(2*MKWD)
      LNGHX = MAXM(LNGHX,4096)
      CS = IFIX(4095.0/FLOAT(LNGHX)*2047.0)
      CS = CS/2047.0
      LNGHY = FLOAT(LNGHX-MKWD)-250.0/CS+.5
      LSTPC = 0
      INCX = 0
      ISTX = -32600
      LMTX = ISTX+LNGHX-10
      ISTY = MKWD/2+5
      LMTY = ISTY+LNGHY-450.0/CS
C.....MOVE WINDOW LEFT SLIGHTLY SO LEFT EDGE OF
C.....MARKER IS ALWAYS DISPLAYED.
      IXC = -32510+LNGHX/2
      IYC = ISTY+LNGHY-LNGHX/2
      IF(ISSW(1).LT.0) WRITE(6,1000) IXC,IYC,LNGHX,LNGHY,CS
C.....COUNT # MASTER PIECES
      NPECM=0
      DO 65 I=1,NPEC
      IF(JLOF(JBM(20),IDUM(KFLG+I)),EQ.1) GO TO 65
      NPECM=NPECM+1
   65 CONTINUE
      DO 100 I=1,NPECM
      MAXX = 0
      DO 70 J=1,NPEC
      IF(IDUM(KXMX+J).LE.0) GO TO 70
      IF(FLOAT(IDUM(KXMX+J)).LT.FLOAT(MAXX)) GO TO 70
      MAXX = IDUM(KXMX+J)
      MAXY = IDUM(KYMX+J)
      INDX = J
   70 CONTINUE
```

```
      IDUM(KXMX+INDX) = -MAXX
      ISTY = ISTY+MAXY
      IF(ISTY.LT.LMTY.AND.ISTX+MAXX.LT.LMTX) GO TO 90
      ISTY = MKWD/2+MAXY+5
      ISTX = ISTX+INCX
      INCX = 0
      IF(ISTX.LT.LMTX) GO TO 90
      FACT = FACT+.5
      DO 80 J=1,NPEC
      IF(IDUM(KXMX+J).GT.0) GO TO 80
      IDUM(KXMX+J) = -IDUM(KXMX+J)
   80 CONTINUE
      GO TO 60
   90 IDUM(KXMN+INDX) = ISTX
      IDUM(KYMN+INDX) = ISTY-MAXY
      INCX = MAXO(INCX,MAXX)
      IF(ISSW(1).LT.0)WRITE(6,1000) IDUM(KXMN+INDX),IDUM(KYMN+INDX),
     1 INDX,NPEC,FACT
 1000 FORMAT (4I6,F8.3)
  100 CONTINUE
  105 DO 110 I=1,NPEC
C.....,IGNOR SLAVE PIECES
      IF(JLOF(JBP(20),IDUM(KFLG+I)),EQ,1) GO TO 110
      IF(MKOP.NE.2HFR) IDUM(KXMX+I) = -IDUM(KXMX+I)
      IDUM(KXMN+I) = IDUM(KXMN+I)-IDUM(KXMX+I)
      IDUM(KYMX+I)=IDUM(KYMN+I)+IDUM(KYMX+I)
  110 CONTINUE
      RETURN
      END
      ENDS

FTN4
      SUBROUTINE SQUEZ(IBUF,NWDS,NSEC)
C.....OCT. 9, 1972
C.....MARGARET ROMAN
C.....PROGRAMS:CHECK,PGRAB
C.....VERSION 2.0
C.....TEMPORARY ROUTINE FOR DATA SUPPRESSION
      DIMENSION IBUF(1),JBUF(128)
      DIMENSION IDDX(462)
      COMMON JBP(10),JBG(4)
C.....SET ANGLE OF DEVIATION AND TOLERANCE
C.....SET STANDARD DEVIATION ANGLE TO 2 DEGREES FOR NOW
      ANG=.0349
      TOL = .02
      IF(ISSW(11).LT.0) ANG = .0174
C.....USE SWITCH 12 FOR NO DATA SUPPRESSION
      IF(ISSW(12).LT.0) ANG = 0.0
      IF(ISSW(13).LT.0) TOL = .03
      IF(ISSW(14).LT.0) TOL = .04
      IF(ISSW(15).LT.0) TOL = .05
      IF(ANG.EQ.0.0) RETURN
      IFTT = 0
C.....FLAG POINTS TO SUPPRESS BY SETTING POINT TYPE FIELD (JBP(9))
C.....TO 15.
      DO 105   I1=1,NSEC
      L1 = (I1-1)*128+3
      L2 = L1+123
      DO 100 I2=L1,L2,3
C.....END OF DATA?
      IF(IBUF(I2).EQ.12000000B) GO TO 110
      XX = IBUF(I2+1)*.001
      YY = IBUF(I2+2)*.001
      ITYPP=JLOF(JBP(9),IBUF(I2))
      IF(ISSW(4).LT.0)WRITE(6,3333)ITYPP,XX,YY,IFTT
```

```
 3333 FORMAT(I4,2F9.3,I6)
C..........NOTCH? YES, SAVE PREVIOUS POINT
      IF(ITYPP.EQ.4) GO TO 30
C..........NOT OUTLINE POINT, SAVE IT
      IF(ITYPP.GT.1.AND.ITYPP.NE.8.AND.ITYPP.NE.11) GO TO 50
C..........DON'T SUPPRESS A FIXED INTERMEDIATE POINT
      IF(ITYPP.EQ.11) GO TO 30
C..........SUPPRESS POINT?
      IF(IFTT.NE.0) GO TO 20
      X1 = XX
      Y1 = YY
   10 A1 = ATAN2(Y1-YP,X1-XP)
      S = SIN(A1)
      C = COS(A1)
      IFTT = I2
      IF(ISSW(4).LT.0)WRITE(6,4444)A1,S,C,XP,YP,X1,Y1,IFTT
 4444 FORMAT(5X,7F10.3,I6)
      GO TO 100
   20 X2 = XX
      Y2 = YY
C..........DON'T SUPPRESS A GRADE POINT OR A NOTCH
C..........OR A FIXED INTERMEDIATE POINT
      ITMP = JLOF(JBP(9),IBUF(IFTT))
      IF(ITMP.EQ.0.OR.ITMP.EQ.8.OR.ITMP.EQ.4.OR.ITMP.EQ.11) GO TO 30
      CALL XFORM(XT,YT,X2-XP,Y2-YP,S,C)
      A1=ABS(ATAN2(YT,XT))
      IF(ISSW(4).LT.0)WRITE(6,1111)A1,XT,YT,X2,Y2,XP,YP,X1,Y1
 1111 FORMAT(9F8.3)
C..........CHECK ANGLE OF DEVIATION AND TOLERANCE
      IF(A1.GT.ANG.OR.ABS(YT).GT.TOL) GO TO 30
C..........FLAG PREVIOUS POINT (X1,X1) FOR SUPPRESSION
      CALL JSTF(15,JBP(9),IBUF(IFTT))
      X1 = X2
      Y1 = Y2
      IFTT = I2
      GO TO 100
C..........SAVE PREVIOUS POINT
   30 XP=X1
      YP = Y1
      X1 = X2
      Y1 = Y2
      GO TO 10
C..........SAVE POINT
   50 IF(IFTT.NE.0) GO TO 100
      XP = XX
      YP = YY
      X1 = XX
      Y1 = YY
  100 CONTINUE
      M2=L2+2
      IF(ISSW(4).LT.0)WRITE(6,2222)(IBUF(JJ),JJ=L1,M2)
 2222 FORMAT(8O8)
  105 CONTINUE
C..........PACK DATA
  110 JPTR = 3
      JBUF    = IBUF
      JBUF(2) = IBUF(2)
      NSEC = 1
      DO 500 I1=1,NSEC
      L1 = (I1-1)*128+3
      L2 = L1+125
      DO 500 I2=L1,L2,3
C..........END OF DATA?
      IF(IBUF(I2).EQ.1200000B) GO TO 510
      IF(JLOF(JBP(9),IBUF(I2)).EQ.15) GO TO 500
```

```
C..........POINT NOT FLAGGED FOR SUPRESSION
      JBUF(JPTR) = IBUF(12)
      JBUF(JPTR+1) = IBUF(12+1)
      JBUF(JPTR+2) = IBUF(12+2)
      JPTR = JPTR+3
      IF(JPTR.LT.128) GO TO 500
C..........DUMP SUPPRESSED DATA BACK INTO "IBUF", ONE SECTOR AT A TI
      IPTR = (NSEC-1)*128
      DO 200 I3=1,128
  200 IBUF(IPTR+I3) = JBUF(I3)
      MSEC = MSEC+1
      JBUF    = 0
      JBUF(2) = 0
      JPTR = 3
  500 CONTINUE
C..........DUMP LAST SECTOR INTO "IBUF"
  510 JBUF(JPTR) = IBUF(12)
      IPTR = (NSEC-1)*128
      DO 520 I3=1,JPTR
  520 IBUF(IPTR+I3) = JBUF(I3)
      NWDS = IPTR+JPTR
      IF(ISSW(10).LT.0) WRITE(6,1000) NSEC,MSEC,I2,NWDS
 1000 FORMAT (2I8)
      NSEC = MSEC
      IF(NWDS.EQ.MWDS) GO TO 550
C..........CLEAR OUT REST OF "IBUF"
       I2 = NWDS+1
      DO 530 I3=I2,NWDS
  530 IBUF(I3) = 0
  550 NWDS = MWDS
      CALL JSTF (MSEC,JBP(5),IBUF)
      RETURN
      END
      ENDS


FTN4
      SUBROUTINE SORTP (IWORK,IPNTR)
C
C     VERSION 2.1
C     EDITED 20 FEB 73
C     RONALD GORDON
C
      DIMENSION IWORK(1)
      INTEGER DONE
      DATA DONE /037777B/, NULL /037770B/
      DO 260 I=0,IPNTR-9,8
      K=I
      DO 240 J=I+8,IPNTR-1,8
C                                     INTNO
      IF (IWORK(K+6)-IWORK(J+6)) 240,200,230
C                                     RULE+SIZE
  200 IF (IWORK(K+7)-IWORK(J+7)) 240,210,230
C                                     XY PERCENT
  210 IF (IWORK(K+8)-IWORK(J+8)) 240,220,230
C                                     MARK DUPLICATE
  220 IF (IWORK(J+6).NE.NULL) IWORK(J+6)=DONE
      GO TO 240
C                                     REMEMBER SWAP
  230 K=J
  240 CONTINUE
C                                     DELETE DONE'S AT END
      IF (IWORK(K+6).EQ.DONE) GO TO 270
      IF (K.EQ.I) GO TO 260
C                                     SWAP 'EM
```

```
      DO 250 J=1,8,1
      ITEMP=IWORK(I+J)
      IWORK(I+J)=IWORK(K+J)
  250 IWORK(K+J)=ITEMP
  260 CONTINUE
      IF (IWORK(IPNTR-2).EQ.DONE) IPNTR=IPNTR-8
      RETURN
  270 IPNTR=I
      RETURN
      END
END$
      FTN4
      SUBROUTINE STORE(IND,ISTAT,IX,IY)
      DIMENSION IBUF(128),ITYP(3)
      COMMON LOUT,ITYPE,ORIGX,ORIGY,LDIRC,XPEND,AXOLD,AYOLD,
     1   IPNF(3),IPNR(3),XMAX,YMAX,SCAL,XLAST,YLAST,LASTP,LASTA,
     2   LASTT,IPLT(3),JTYPE,JOB,JPOS,JTOOL,JHOME,JIZ,SPACE,IOPTN
     3   ,NPLAT,JRV,JR1,JR2,JR3,JR4
      DATA ITYP/2HCH,2HMA,2HLO/
      M=IND+2
      GO TO (100,200,300,400,200),M
C.....START OF A NEW PLOT
  100 CALL UDSC(0)
      CALL EXEC(15,IPLT,MAXSL)
      MAXSL=MAXSL-1
      CALL PLTST (-4,NSECT)
      NSECS=NSECT
      CALL EXEC(14,1028,IBUF,128,IPLT,0)
      LTYPE=IBUF(3)
      IBUF(3)=ITYPE
      CALL EXEC(15,1028,IBUF,128,IPLT,0)
C.....SET THE PLOT TYPE FLAG
      IBUF(1)=0
      DO 105   I=1,3
      IF(ITYPE.EQ.ITYP(I)) GO TO 110
  105 CONTINUE
  110 CALL JSTF(I,JTYPE,IBUF)
C.....SET NEW JOB FLAG IF NOT SAME TYPE OPERATION
      IF(ITYPE.NE.LTYPE) CALL JSTF(1,JOB,IBUF)
      IBUF(2)=NSECS
      ICNT=3
      RETURN
C.....TERMINATE PLOT, WRITE END OF FILE STATUS WORD.
  200 IBUF(ICNT)=-1
      IBUF(ICNT+1)=NSECS
      IBUF(ICNT+2)=NSECT
      CALL UDSC(0)
      CALL EXEC(15,1028,IBUF,128,IPLT,NSECT)
C.....UPDATE "PUT" POINTER
  205 NSECT=NSECT+1
      IF(NSECT.GT.MAXSL) NSECT=1
C.....THE FILE IS FULL WHEN THE "PUT" POINTER EQUALS THE "GET" POINTER
      CALL PLTST (-3,NSG)
      IF (NSECT.NE.NSG) GO TO 220
C.....NO ROOM, RESET "PUT" POINTER AND ABORT.
      N=2
      CALL PLTST (4,NSECS)
  210 CALL FINI (N)
  220 IF(IND.NE.0.AND.IND.NE.3)GO TO 305
      CALL PLTST (4,NSECT)
      IF(IND.EQ.3) GO TO 305
      N=1
      GO TO 210
C.....NORMAL STORE
  300 IBUF(ICNT)=ISTAT
```

```
      IBUF(ICNT+1)=IX
      IBUF(ICNT+2)=IY
      ICNT=ICNT+3
      IF(ICNT.LT.129) RETURN
C.....SECTOR FULL. DUMP AND RESET THE COUNTER.
      ICNT=3
      CALL UDSC(0)
      CALL EXEC(15,1028,IBUF,128,IPLT,NSECT)
      GO TO 205
  305 DO 310   I=1,128
  310 IBUF(I)=0
  320 RETURN
C.....SET THE NEW JOB FLAG
  400 CALL JSTF(1,JOB,IBUF)
      RETURN
      END
      END$


FTN4
      SUBROUTINE TABCL (N,PX,PY,IND,NGEN,ISTAT)
C..........OCT. 9, 1972
C..........MARGARET ROMAN
C..........PROGRAMS: CHECK
      DIMENSION PX(1),PY(1),IND(1)
C.....THIS ROUTINE GENERATES POINTS BETWEEN INPUT DATA POINTS
C.....TO ASSURE SMOOTH CURVES.  THE INPUT DATA IS ORGANIZED WITH
C.....NGEN WORD GAPS BETWEEN INPUT POINTS.  TABCL FILLS THOSE
C.....GAPS.  THE IND ARRAY SPECIFIES THE POINT TYPE WHERE
C.....0=CORNER AND 1=INTERMEDIATE.  BETWEEN CORNER POINTS IX AND IY
C.....SHOULD BE LEFT BLANK.
      DIMENSION Z(3,35),S(35),X(35),Y(35),C(35),B(35),
     1          U(35),V(35)
      ISTAT = 0
      NOUT=1
      POINT = NGEN+1
      NBEG = 1
      NINC = NGEN+1
   10 NPT = 1
C..........LOOP TO FIND ONE CURVE AT A TIME
      DO 20 I=NBEG,N
      ITMP = IND(I)
C..........IGNOR POINTS NOT ON OUTLINE
      IF(ITMP.GT.1.AND.ITMP.NE.8.AND.ITMP.NE.11) GO TO 20
      U(NPT) = PX(I)
      V(NPT) = PY(I)
      IF(NPT.EQ.1) INDEX = I+1
      IF((ITMP.EQ.0.OR.ITMP.EQ.8).AND.NPT.NE.1) GO TO 30
      NPT = NPT+1
      IF(NPT.EQ.32) GO TO 30
   20 CONTINUE
      RETURN
C..........CURVE FOUND
   30 NBEG = I
      IF(NPT.GT.2) GO TO 50
C..........TWO ADJACENT CORNER POINTS
      GO TO 10
C..........GENERATE "NGEN" ADDITIONAL POINTS THRU TABCL
   50 U(NPT+1) = U(NPT-2)
      V(NPT+1) = V(NPT-2)
      MEND = NPT-1
C..........COMPUTE SLOPES FOR POINTS U,V ON THE CURVE
      DO 80 M=1,MEND
      J = M+2
      I = M+1
```

```
      K = M-1
      G=U(I)-U(M)
      H=V(I)-V(M)
      W = SQRT(G**2+H**2)
      X(I) = W
C.....DUPLICATE POINTS?
      IF(ABS(W).LT..0005) GO TO 165
      DU = U(J)-U(M)
      DV = V(J)-V(M)
      X(J) = (G*DU+H*DV)/W
      Y(J) = (-H*DU+G*DV)/W
      C(M) = -ATAN2(H,G)
      IF(M.LE.1) GO TO 70
      B(M) = S(M)
      S(M) = TAN(ATAN(S(M))+C(M)-C(K))
   70 DEN = (X(J)**2+Y(J)**2-X(I)*X(J))
      IF(ABS(DEN).LT..000001) GO TO 165
      S(1) = X(I)*Y(J)/DEN
      IF(M.GT.1) GO TO 80
      S(M) = -S(I)
   80 CONTINUE
      B(NPT)=S(NPT)
C.....ADJUST SLOPES TO ASSURE CONSTANT RADIUS OF CURVATURE
      DO 110 J=1,15
      L = 1
      DO 100 M=2,MEND
      I = M+1
      K = M-1
      A = S(M)
      C1 = TAN(C(K)-C(M))
   90 S3 = -(4.0*S(M)+2.0*B(I))/(X(I)*(1.0+S(M)**2)**1.5)
      S4 = (2.0*S(K)+4.0*B(M))/(X(M)*(1.0+B(M)**2)**1.5)
      D3 = S3-S4
      D4 = 4.0/X(M)+4.0/X(I)
      A = A+D3/D4
      S(M) = A
      B(M) = TAN(ATAN(A)-C(M)+C(K))
      L = L+1
      IF(L.GT.500) GO TO 170
      IF(ABS(D3).GE.0.05) GO TO 90
  100 CONTINUE
      IF(L.EQ.MEND) GO TO 120
  110 CONTINUE
C.....COMPUTE COEFFICIENTS
  120 B(1) = -B(2)
      S(NPT)=-B(MEND)
      DO 130 M=2,NPT
      K = M-1
      Z(1,K) = (B(M)+S(K))/X(M)**2
      Z(2,K) = -(2.0*S(K)+B(M))/X(M)
  130 Z(3,K) = S(K)
C.....COMPUTE ADDITIONAL POINTS USING EQUATION (Z)
      I = 0
      IPTYP=1
      DO 160 M=1,2000
      IF(M.NE.1) GO TO 150
  140 I = I+1
      IF(I.GE.NPT) GO TO 10
      X3 = 0.0
C.....DIVIDE CURVE INTO EQUAL SIZED STEPS
      STEP = X(I+1)/POINT
      ITMP=IND(I+1)
      IF(ITMP.EQ.1.OR.ITMP.EQ.11) IPTYP=IND(I+1)
  150 X3 = X3+STEP
      IF((X3-X(I+1)).GT..01) GO TO 140
```

```
      Y3 = ((Z(1,I)*X3+Z(2,I))*X3+Z(3,I))*X3
      DU = C(I)
      DV = COS(DU)
      DU = SIN(DU)
      XX = X3*DV+Y3*DU+U(I)
      YY = -X3*DU+Y3*DV+V(I)
  152 ITMP = IND(INDEX)
      IF(ITMP.NE.4) GO TO 155
      INDEX = INDEX+1
      GO TO 152
  155 PX(INDEX) = XX
      PY(INDEX) = YY
      IF(ITMP.EQ.99) IND(INDEX) = IPTYP
      INDEX = INDEX+1
  160 CONTINUE
      IF(NSEG.EQ.N) RETURN
      GO TO 10
  165 CALL ERR(3,1)
      ISTAT = 1
      RETURN
  170 WRITE(5,1000)
 1000 FORMAT("***BAD DIGITIZE POINT")
      CALL RTN
      END
      ENDS


ASMB
      NAM TOP
      EXT EXEC
START NOP
      JSB EXEC
      DEF *+5
      OCT 55
      OCT 01
      DEF FORM
      OCT 54
      JSB EXEC
      DEF *+2
      DEF *+1
      DEC 0
FORM  OCT 0040
      END START


ASMB,R
      NAM UDLBL,7
      ENT UDLBL
*
*     RETURNS USER DISK NUMBER (LAST DIGIT OF LABEL)
*
      EXT .ENTR,EXEC
LABEL BSS 1
UDLBL NOP
      JSB .ENTR
      DEF LABEL
      JSB EXEC        READ LABEL, ETC.
      DEF *+5
      DEF RCODE
RCODE DEC 23          UNUSED BY EXEC
      DEF CODSC
      DEF .+1
      LDA BB5,I       GET LAST WORD OF LABEL
      ADA =B147720    SUBTRACT ASC 00
      STA LABEL,I     PUT IN USER LIST
```

```
      JMP UDLNL,I
*
BBB   OCT 4124        BASE PAGE SYSTEM BUFR
CUUSC EQU 1016        CURRENT USER DISK SUBCHANNEL
.     EQU 536         BASE PAGE CONSTANTS
      END


ASMB,R
      NAM UDSC,7
      ENT UDSC
*
*     MOVES USER TO SPECIFIED SUBCHANNEL
*
*     CALLING SEQ:  CALL UDSC (IDISK)
*     WHERE IDISK=0 MEANS FIXED, SUBCHANNEL 0
*           IDISK=1 MEANS REMOVABLE, SUBCHANNEL 1
*
      EXT .ENTR,EXEC
CHANL BSS 1
UDSC  NOP
      JSB .ENTR
      DEF CHANL
      LDA CHANL,I     GET REQUESTED SUBCHANNEL
      CPA CUUSC       ARE WE THERE ALREADY?
      JMP UDSC,I      YES - RETURN FAST
      JSB EXEC        NO - MOVE HIM THERE
      DEF *+3
      DEF RCODE
RCODE DEC 23          UNUSED BY EXEC
      DEF CHANL,I
      DEF .+1
      JMP UDSC,I      RETURN
CUUSC EQU 1016        CURRENT USER DISK SUBCHANNEL
.     EQU 536         BASE PAGE CONSTANTS
      END


FTN4
      SUBROUTINE VALID(ICODE,IO,IRSLT)
      DIMENSION IFN(5,15),IOON(3,6),IPN(3,31)
      DIMENSION IFIL(3),ISEG(3),IFN(6),INAME(5),IBUF(128)
      DIMENSION ICS(15),INS(15),NGE(6),NTE(6),IBL(6),IBR(6)
      DATA IPN/2HMO,2HNS,2HT ,2HWP,2HRD,2HD ,2HPG,2HRA,2HB ,2HDS,2HCA,
     12HM ,2HPG,2HRA,2HD ,2HRS,2HTY,2HL ,2HCH,2HEC,2HK ,2HCH,2HKA,2H  ,
     12HCH,2HKB,2H  ,2HCH,2HKC,2H  ,2HMA,2HRK,2H  ,2HMR,2HKA,2H  ,2HMR,
     12HKH,2H  ,2HMR,2HKC,2H  ,2HMR,2HKD,2H  ,2HMR,2HKE,2H  ,2HMR,2HKF,
     12H  ,2HMR,2HKG,2H  ,2HMR,2HKH,2H  ,2HMR,2HKI,2H  ,2HMR,2HKJ,2H  ,
     12HOR,2HAW,2HM ,2HPL,2HOT,2HP ,2HMA,2HIN,2HT ,2HMA,2HNT,2HA ,2HMA,
     12HNT,2HR ,2HRR,2HDL,2HE ,2HBE,2HNT,2HR ,2HBB,2HUL,2HD ,2HBL,2HIS,
     13HT ,2HMA,2HLF,2H  /
      DATA ION/2HMA,2HRK,2HR ,2HPI,2HEC,2HE ,2HMA,2HLF,2HP ,2HST,2HYL,
     12HL ,2HGR,2HAD,2HE ,2HRU,2HLE,2HS ,2HPL,2HOT,2HF ,2HJO,2HBQ,2H  ,
     12HSE,2HCT,2HR ,2HCH,2HEK,2H  ,2HCH,2HDI,2HG ,2HOP,2HTN,2HS ,2HME,
     12HSS,2HG ,2HTP,2HEC,2HD ,2HTP,2HEC,2HS /
      DATA IOON/2HMA,2HRK,2HD ,2HPI,2HEC,2HD ,2HMA,2HLF,2HD ,2HST,2HYL,
     12HD ,2HGR,2HAD,2HD ,2HRU,2HLE,2HD /
      DATA ICS/1,1,0,0,0,0,0,0,0,0,0,0,0,0,0/
      DATA IFN/1,2,7,3,4,5/
      DATA IBL/5,3,2,3,5,6/
      DATA IBR/6,4,3,4,7,7/
      IRSLT=0
      CALL UDSC(0)
      DO 15 I=1,31
      DO 5 J=1,3
```

```
 5 IFIL(J)=IPN(J,I)
   CALL EXEC(18,IFIL,ISECT)
   IF (ISECT.NE.0) GO TO 15
   IRSLT=IRSLT+1
   WRITE (IO,10) IFIL
10 FORMAT ("PROGRAM ",3A2,"NOT ON FIXED PLATTER")
15 CONTINUE
   DO 35 I=1,15
   DO 20 J=1,3
20 IFIL(J)=IDN(J,I)
   CALL UDSC(ICS(I))
   CALL EXEC(18,IFIL,ISECT)
   IF (ISECT.EQ.0) GO TO 25
   INS(I)=ISECT
   GO TO 35
25 IRSLT=IRSLT+1
   INS(I)=0
   WRITE (IO,30) IFIL,ICS(I)
30 FORMAT ("DATA FILE ",3A2,"NOT ON SUBCHANNEL ",I1)
35 CONTINUE
   DO 65 I=1,6
   DO 40 J=1,3
40 IFIL(J)=IDDN(J,I)
   INTNO=0
   NGE(I)=0
   NTE(I)=0
   ICD=IPN(I)+1400B
   CALL UDSC(ICS(I))
45 CALL DIREC(ICD,INAME,INTNO,NSCTR,ISTAT)
   IF (ISTAT.EQ.2) GO TO 50
   IF (ISTAT.EQ.3) GO TO 55
   IF (ISTAT.EQ.5) GO TO 65
   NGE(I)=NGE(I)+1
50 NTE(I)=NTE(I)+1
   INTNO=INTNO+1
   GO TO 45
55 IRSLT=IRSLT+1
   WRITE (IO,60) IFIL,ICS(I)
60 FORMAT ("DIRECTORY ",3A2,"NOT ON SUBCHANNEL ",I1)
65 CONTINUE
   IF (ICODE.EQ.0) RETURN
   WRITE (IO,70)
70 FORMAT (//"MARKAMATIC SYSTEM"//"FILE UTILIZATION"/33X,"AVERAGE",
  1 5X,"CURRENT      DIRECTORY"/"  DIRECTORY      DATA FILE      FILE
  1SIZE    FILE SIZE      FILE SIZE")
   DO 85 I=1,6
   IF (INS(I).EQ.0) GO TO 85
   IF (NTE(I).EQ.0) GO TO 85
   DO 75 J=1,3
   IFIL(J)=IDDN(J,I)
75 ISEG(J)=IDN(J,1)
   CALL UDSC(ICS(I))
   CALL EXEC(14,66,IBUF,128,IFIL,0)
   ISL=JLOF(JFLD(14,15),IBUF(128))
   CALL EXEC(14,66,IBUF,128,ISEG,ISL)
   SLC=ISL+JLOF(JFLD(15L(I),15R(I)),IBUF(1))
   A=NGE(I)*100.0/NTE(I)
   B=SLC*100.0/INS(I)
   C=SLC/NGE(I)
   IF (SLC.EQ.0.OR.NGE(I).EQ.0) C=0
   CALL EXEC(18,IFIL,ISECT)
   WRITE (IO,80) IFIL,A,ISEG,B,C,INS(I),ISECT
80 FORMAT (3A2,F5.1,"%    ",3A2,F5.1,"%     ",F6.1," SEC      ",I4,
  1" SEC",6X,I4," SEC")
```

```
   85 CONTINUE
      RETURN
      END
      END$


ASMB
      NAM VGISV,3
      ENT VGISV
      EXT MPOFF
VGISV JSB MPOFF
      DEF DSPLY
DSPLY LDA ZERO
      OTA 6
      LDA CW5
      OTA 103
      STC 100
      CLC 118,C
      LDA CW2
      CLC 2
      OTA 2
      STC 2
      LDA CW1
      OTA 6,C
      LDA CW3
      OTA 2
      STC 6
      LDA CW4
      OTA 10B
      STC 100
      LDA TIM        SET UP DELAY
      STA CNT        STORE DELAY VALUE
      LIA 1          READ "S" REGISTER
      STA PS         STORE IN PS=DATA LIST
DEL1  ISZ CNT        DELAY OVER?
      JMP DEL1       NO, ADD AND TEST
      JMP DSPLY      YES, START DISPLAY OVER
      NOP
CW1   OCT 100011     OUTPUT, DMA ADD 11
CW2   DEF FWA        START OF LIST ADDRESS
CW3   OCT 037777     DMA WORD COUNT
CW4   OCT 176400     DISPLAY START
CW5   OCT 171000     DELAY FACTOR
TIM   OCT 174000     TEMP STOR
CNT   NOP
ZERO  NOP
FWA   OCT 140005     START OF DISPLAY LOAD MCR
      OCT 361        TURN ON 4 SCOPES, TERM
      OCT 40014      LOAD IOR
      OCT 07776l     MAX ON, TERM
      OCT 40021      LOAD THE FOLLOWING
PS    BSS 1          PSR = S REG CONTENTS
      OCT 0          NMR=0
      OCT 07776B     CSR=+1
      OCT 0          DXR=0
      OCT 1          DYR=0 AND TERM
      OCT 10004      VECTOR ABSOLUTE
      OCT 100001     LOAD X = -1
      OCT 100012     LOAD, MOVE Y=+1
      OCT 07776S     DRAW X=+1
      OCT 07776B     DRAW Y=+1
      OCT 100005     DRAW X=-1
      OCT 100016     DRAW Y=-1 AND TERM
      OCT 34000l     LOAD PSR
      OCT 03776l     + ,5 , TERM
```

```
        OCT 014004      VECTOR ABSOLUTE
        OCT 1[[[31      ...
        OCT 100016
        OCT 044021      LOAD PSR
        OCT 077761      *1, TERM
        OCT 010157      CHARACTERS, SIZE 1
        OCT 000000      FORM FEED
        OCT 040002      A, B
        OCT 040304      C, D
        OCT 040506      E, F
        OCT 040710      G, H
        OCT 041012      I, J
        OCT 000024      NOOP, TERM
        OCT 044010      LOAD
        OCT 000000      XR=0
        OCT 000001      YR=0, TERM
        OCT 010177      CHARACTERS, SIZE 3
        OCT 101         A
        OCT 102         B
        OCT 103         C
        OCT 104         D
        OCT 105         E
        OCT 24          TERM
        OCT 030000      HALT DMA
        END VGISV
        END$


ASMB
        NAM VGUTL,7
        ENT START,DTHIT,LIGHT,FSHIT,STAT,STOP
        EXT .ENTR,MPOFF,MPON
        EXT INIT,COMP,ISTAT,XTHIT,YTHIT,FSWCH,LSTFB,DSTAT
*
*       VECTOR GENERAL DISPLAY UTILITY ROUTINES
*
*            ROUTINE                          FUNCTION
*
*            START(IRATE,ITAB,IFILE)          DISPLAY START PROCESSOR
*            DTHIT(ISTAT,IX,IY)               DATA TABLET PROCESSOR
*            LIGHT(INUM,IOPER)                DISPLAY FB LIGHT PROCESSOR
*            FSHIT(N,IMASK)                   FB PROCESSOR
*            FSACT(IACT)                      FB ACTIVE BUTTON PROCESSOR
*            STAT                             DISPLAY STATUS PROCESSOR
*            STOP                             DISPLAY STOP PROCESSOR
*
*       START ROUTINE
*
RATE    NOP             REFRESH RATE ADDRESS
TABLE   NOP             LIST TABLE ADDRESS
DFILE   NOP             DISPLAY FILE ADDRESS
START   NOP
        JSB .ENTR
        DEF RATE
*
        CLA             A=0
        STA ISTAT       *
        STA XTHIT       *
        STA YTHIT       *  CLEAR COMMUNICATION WORDS
        STA FSWCH       *
        STA LSTFB       *
        LDA DFILE       LOAD DISPLAY FILE ADDRESS
        ADA .-1         ADJUST FOR POINTER ADDITION
        STA PTAB        STORE IN PARAMETER LIST
        LDA TABLE       LOAD LIST TABLE ADDRESS
```

```
        STA PTAB+1      STORE IN PARAMETER LIST
        LDA .-3         LOAD 40 CPS REFRESH COUNT
        LDB RATE,I      LOAD REQUESTED REFRESH RATE
        CPB R30         30 CPS?
        LDA .-4         YES, LOAD 30 CPS REFRESH COUNT
        STA PTAB+2      STORE IN PARAMETER LIST
        JSB MPOFF       TURN MEMORY PROTECT OFF
        DEF *+1         RETURN ADDRESS
CALL    JSB INIT        INITIALIZE DISPLAY
        DEF PTAB        PARAMETER TABLE ADDRESS
        JMP CALL        DMA REJECT RETURN
        ISZ RUN         SET RUN FLAG
        JMP START,I     RETURN
*
RUN     OCT 0           RUN FLAG
*
*       PARAMETER TABLE
*
PTAB    OCT 0           DISPLAY FILE FWA-1
        OCT 0           LIST TABLE FWA
        OCT 0           REFRESH RATE
*
R30     DEC 30          30 CPS REFRESH CODE
*
*       DATA TABLET ROUTINE
*
ISTAT   NOP             TABLET STATUS ADDRESS
X       NOP             X COORDINATE ADDRESS
Y       NOP             Y COORDINATE ADDRESS
DTHIT   NOP
        JSB .ENTR
        DEF ISTAT
*
        LDA TSTAT       LOAD TABLET STATUS WORD
        AND .+3         MASK FOR PROX AND PRESS
        STA ISTAT,I     STORE STATUS
        LDA XTHIT       LOAD X TABLET COORDINATE
        ALS,ALS         MULTIPLY BY 4
        ADA N2047       SUBTRACT 2K
        LDB YTHIT       LOAD Y TABLET COORDINATE
        BLS,BLS         MULTIPLY BY 4
        ADB N2047       SUBTRACT 2K
        STA X,I         STORE X POSITION
        STB Y,I         STORE Y POSITION
        JMP DTHIT,I     RETURN
*
N2047   DEC -2047       -2K CONSTANT
*
*       DISPLAY FUNCTION BOX LIGHT PROCESSOR
*
LNUM    NOP             LIGHT NUMBER
IOPER   NOP             OPERATION 0=OFF, 1=ON
LIGHT   NOP
        JSB .ENTR
        DEF LNUM
*
        LDA RUN         LOAD RUN FLAG
        SZA             DISPLAY ON?
        JMP MPOK        YEP
        JSB MPOFF       KILL MEMORY PROTECT SYSTEM
        DEF *+1         RETURN ADDRESS
MPOK    LDA LNUM,I      GET LIGHT NUMBER
        SSA             SINGLE LIGHT REQUEST?
        JMP ALL         NO, DO ALL
        CMA             SET ROTATE COUNT
```

```
        STA LNUM        SAVE
        ADA .+8         ADD MAX LIGHT #
        SSA,RSS         IN RANGE?
        CPA .+8         = 0?
        JMP LIGHT,I     NOT LEGAL LIGHT #, RETURN
        LDA IOPER,I     GET ON/OFF BIT
        AND .+1         MASK OFF ANY BAD BITS
        STA B           SAVE
        CLA,INA         SET KILL BIT TO 1ST LIGHT
        JMP LIT2        ROTATE BITS INTO POSITION
LIT1    RAL             KILL BIT LEFT ONE
        RBL             ON/OFF BIT LEFT ONE
LIT2    ISZ LNUM        DONE?
        JMP LIT1        NO
        STA LNUM        SAVE POSITIONED KILL BIT
        AND LREG        GET CURRENT BIT STATUS
        SZA             BIT = 0?
        JMP LIT3        NOPE, GO ON
        SZB,RSS         WANT BIT OFF?
        JMP LIGHT,I     YEP, EASY KILL
LIT3    LDA LNUM        GET POSITIONED KILL BIT
        XOR LREG        KILL OLD BIT
        IOR B           INSERT NEW OPERATION
LIT4    STA LREG        SAVE NEW REGISTER CONTENTS
        OTA OTCHN       SEND
        JMP LIGHT,I     RETURN
*
ALL     LDA IOPER,I     GET OPERATION
        SZA             LIGHTS OFF?
        LDA .+16        NOPE
        JMP LIT4        GO DO IT
*
*       FUNCTION SWITCH HIT ROUTINE
*
N       NOP             FS BUTTON # ADDRESS
IMASK   NOP             REPEAT MASK ADDRESS
FSHIT   NOP
        JSB .ENTR
        DEF N
*
        LDB FSWCH       LOAD FS REGISTER
        SZB             NO HIT?
        CPB .-1         STOP CODE?
        JMP NOHIT       YES, PROCESS
        CPB LSTFB       SAME AS LAST HIT?
        JMP RPEAT       YES, CHECK FOR REPEAT
*
        LDA ACTIV       LOAD ACTIVE BUTTON MASK
        CMB,INB         SET ROTATE COUNT
LOOP1   ERA             ROTATE MASK
        ISZ B           THIS BIT?
        JMP LOOP1       NO, CONTINUE
        SEZ             ACTIVE BUTTON?
        LDB FSWCH       YES
        STB LSTFB       SET LAST HIT
EX,1    STB N,I         SET BUTTON #
        JMP FSHIT,I     RETURN
*
RPEAT   LDA IMASK,I     LOAD REPEAT MASK
        CMB,INB         SET ROTATE COUNT
LOOP2   ERA             ROTATE MASK
        ISZ B           THIS BIT?
        JMP LOOP2       NO, CONTINUE
        SEZ             OK TO REPEAT?
        LDB FSWCH       YES
```

```
       JMP EX,1        CONTINUE
*
NOHIT  CLA             A=0
       STA LSTFB       CLEAR LAST HIT
       JMP EX,1        EXIT
*
*      FUNCTION SWITCH ACTIVE BUTTON ROUTINE
*
IACT   NOP             ACTIVE BUTTON MASK ADDRESS
FSACT  NOP
       JSB .ENTR
       DEF IACT
*
       LDA IACT,I      LOAD ACTIVE BUTTON MASK
       STA ACTIV       SAVE
       JMP FSACT,I     RETURN
*
ACTIV  OCT 177777      ACTIVE BUTTON MASK
*
*      DISPLAY STATUS ROUTINE
*
STAT   NOP
       JSB .ENTR
       DEF STAT
*
       LDA INTAB,I     LOAD INT. TAB. ENTRY FOR DMA CH. 1
       CPA .-1         ASSIGNED BY DISPLAY DRIVER?
       RSS             YES
       JMP STAT,I      RETURN
       LDB DSTAT       LOAD INITIAL STATUS
LOOP   LDA DSTAT       LOAD STATUS WORD
       CPA B           SAME AS INITIAL STATUS?
       JMP LOOP        YES, WAIT FOR CHANGE
       JMP STAT,I      RETURN
*
*      DISPLAY STOP ROUTINE
*
STOP   NOP
       JSB .ENTR
       DEF STOP
*
       JSB MPOFF       TURN MEMORY PROTECT OFF
       DEF *+1         RETURN ADDRESS
       JSB COMP        TERMINATE DISPLAY OPERATIONS
       JSB MPON        TURN MEMORY PROTECT ON
       DEF *+1         RETURN ADDRESS
       CLA             A=0
       STA RUN         CLEAR RUN FLAG
       JMP STOP,I      RETURN
*
*      BASE PAGE COMMUNICATION AREA
*
B      EQU 1           B-REGISTER
DTCHN  EQU 25B         DATA TABLET/FB I/O CHANNEL
LREG   EQU 35B         LIGHT REGISTER STORAGE
.      EQU 36B         BASE PAGE CONSTANTS, .=0, + OR - 1 TO 10
INTAB  EQU 2M1B        INTERRUPT TABLE ADDRESS
*
       END


FTN4
      SUBROUTINE VISIB (IMGNO,IORG,IX,IY)
C.....ROUTINE TO DETERMINE VISIBILITY OF PIECE ON THE
C.....DISPLAY SCREEN.  ASSIGN THE IMAGE ORIGIN.
```

```
C
C
C
C
      COMMON MOUT,LABF(3),LABR(3),NOSTY,INSTY(31),JBS(15),NAMST(3),
     1 MKUP(3),NSCMD,JCM(20),NAMMK(3),JBG(4),NAMGD(3),JBP(12),
     2 IDMK(3),IJOP,NPEC,NSHD,FACTR,NSIZ,ISIZ(42),WIDTH,EFF,SX,SY,KSTL,
     3 YARDS,NAP,KINT,KFLG,KINC,KSIZ,KXMN,KYMN,KANG,KXMX,KYMX,KOPT,
     4 KAD,KADP,KDF,LSMX,MKWD,CSMN,NRGE,IMGP,KREL,ITYPE,
     5 NOIMG,IOFSZ,IMGY,IMGE,IMGTC,IMKO1,IMKO2,IMEND,IOXP,IOYP,
     6 IXC,IYC,IXL,IXR,IYU,IYL,CS,MDELX,MDELY,JFLG,MLW,MUP,ISLX,ISLY,
     7 TOTAL,MYL,XPC,YPC,IMGLP,IMESG,IEDG,ISOFF,SOFF,KSLH,
     8 IMGPS,MAXOF,KSLV,KTC,IOPTN,IAREA,ISIZE,KCSR,KYES,KPSIZ,KPLL,
     9 IT1,IT2,IT3,IT4,IT5,IT6,IT7,IT8,IT9,IT10,
     A IOFSET,KPLH,IPLP,ICUST,IMGPK,
     B T1,T2,T3,T4,T5,T6,T7,T8,T9,T10,
     C JSTY,JPG,JNG,JSHD,JRT,
     D ILPB,IBU,IRD,IOV,IBR,IBL,ITCW,ITCCW,
     E IRCW,IRCCW,ISL,IRE,IFL,IPK,ICOM,
     F IDUM(1)
      IORG = 0
      KK = IMGNO
      LMT=IRND(2000./CS)
      X1 = IDUM(KXMN+KK)
      X2 = IDUM(KXMX+KK)
      Y1 = IDUM(KYMN+KK)
      Y2 = IDUM(KYMX+KK)
      X1C = X1-FLOAT(IXC)
      X2C = X2-FLOAT(IXC)
      Y1C = Y1-FLOAT(IYC)
      Y2C = Y2-FLOAT(IYC)
C.........CHECK LIMITS OF PIECE AGAINST LIMITS OF SCREEN.
      IF((X2-FLOAT(IXL)).GT.5.0.AND.(X2-FLOAT(IXR)).LT.-5.0)GO TO 10
      IF((X1-FLOAT(IXL)).GT.5.0.AND.(X1-FLOAT(IXR)).LT.-5.0)GO TO 10
      RETURN
   10 IF((Y2-FLOAT(IYL)).GT.5.0.AND.(Y2-FLOAT(IYU)).LT.-5.0)GO TO 20
      IF((Y1-FLOAT(IYL)).GT.5.0.AND.(Y1-FLOAT(IYU)).LT.-5.0)GO TO 20
      RETURN
C.......VISIBLE, DETERMINE LOCATION OF THE ORIGIN
   20 IORG = 2
      IX = X2C
      IY = Y1C
      IF(IX.LT.LMT.AND.IY.GT.-LMT) RETURN
C.........VISIBLE BUT NORMAL ORIGIN IS OFF 4000X4000 SCREEN
      IF(IX.LT.LMT) GO TO 40
      IORG = 1
      IX = X1C
   40 IF(IY.GT.-LMT) RETURN
      IORG = 5-IORG
      IY = Y2C
      RETURN
      END
      ENDG


ASMB,L,R
      NAM WCRS,7
      ENT WCRS
*
*     SNEAKY ROUTINE TO RETURN THE REAL TIME STATUS
*     OF THE CARD READER PLUGED INTO I/O SLOT 17
*
      EXT .ENTR,MPON,MPOFF
STATS BSS 1
WCRS  NOP
```

```
      JSB .ENTR
      DEF STATS
      JSB MPOFF       TURN IT LOOSE
      DEF *+1
      LIA 17B         MUST BE PLUGGED INTO SLOT 17
      STA STATS,I
*     JSB MPON        TURN MEM PRTCT ON
*     DEF *+1
      JMP WORD,I      RETURN
      END
ENDS


FTN4
      SUBROUTINE WINDO
C.....DEC. 7, 1972
C.....MARGARET ROMAN
C.....PROGRAM: MARK
C.....VERSION 2.0
C
      DIMENSION IXN(200),IYM(200),IXN(200),IYN(200),IMRK(200)
      COMMON MOUT,LABF(3),LABR(3),NOSTY,INSTY(31),JBS(15),NAMST(3),
     1 MADP(5),NSCMD,JBM(24),NAMMK(3),JBG(4),NAMGD(3),JBP(10),
     2 IDMK(5),IDDP,NPEC,NSHD,FACTR,NSIZ,ISIZ(42),WIDTH,EFF,SX,SY,KSTL,
     3 YARDS,MAP,KINT,KFLG,KINC,KSIZ,KXMN,KYMN,KANG,KXMX,KYMX,KOPT,
     4 KAD,KADP,KDF,CSMX,MKWD,CSMN,NRGE,IMGP,KREL,ITYPE,
     5 NOIMG,IDFSZ,IMGY,IMGE,IMGTC,IMKD1,IMKD2,IMENU,IDXP,IDYP,
     6 IXC,IYC,IXL,IXR,IYU,IYL,CS,MDELX,MDELY,JFLG,MLW,MUP,ISLX,ISLY,
     7 TOTAL,MYL,XPC,YPC,IMGLP,IMESG,IEDG,ISOFF,SOFF,KSLH,
     8 IMGPS,MAXDF,KSLV,KTC,IOPTN,IAREA,ISIZE,KCSR,KYES,KPSIZ,KPLL,
     9 IT1,IT2,IT3,IT4,IT5,IT6,IT7,IT8,IT9,IT10,
     A IOFSET,KPLM,IFLP,ICOST,IMGPK,
     B T1,T2,T3,T4,T5,T6,T7,T8,T9,T10,
     C JSTY,JFG,JNG,JSMD,JRT,
     D ILPB,IMD,IPD,IDV,IBR,IBL,ITCW,ITCCW,
     E IRCW,IRCCW,ISL,IRE,IFL,IPK,ICOM,
     F IDUM(1)
C.....CHECK FOR ALPHA OPTION
      NN=0
      IALPH=4
      IF(JLOF(JBM(15),IOPTN).NE.0)IALPH=3
      NN=NN+IALPH
      IF(NOIMG.GT.200) CALL ERR(1,0)
C.....SET UP WINDOW BOUNDSIN MARKER UNITS
      LMT=IRND(2000./CS)
      DEL = 2048.0/CS
      IXL = IXC - DEL
      IF((FLOAT(IXC)-DEL).LT.-32767.0) IXL = -32767
      IXR = IXC + DEL
      IF((FLOAT(IXC)+DEL).GT.32767.0) IXR = 32767
      IYU = IYC + DEL
      IF((FLOAT(IYC)+DEL).GT.32767.0) IYU = 32767
      IYL = IYC - DEL
      IF((FLOAT(IYC)-DEL).LT.-32767.0) IYL = -32767
C.....ADJUST THE DISPLAY FOR A NEW WINDOW POSITION
      DO 200 IMGNO = 1,NOIMG
      KAD1 = KADP+IMGNO
      IAD1 = IDUM(KAD1)
C.....SLAVE PIECE?
      IF(JLOF(JBM(20),IDUM(KFLG+IMGNO)).EQ.1) GO TO 20
C.....CHECK FOR VISIBILITY
      CALL VISIB (IMGNO,IORG,IX,IY)
      IF(IORG.EQ.0) GO TO 100
C.....VISIBLE, CHECK FOR CURRENTLY IN DISPLAY FILE
   20 IF(IAD1) 30,40,30
```

```
C.....NOT CURRENTLY DISPLAYED, ADD TO DISPLAY FILE
C.....OR VISIBLE AND CURRENTLY IN DISPLAY FILE
C.....SLAVE PIECE?
   30 IF(JLOF(JBM(20),IDUM(KFLG+IMGNO)).EQ.1) GO TO 45
      IXM(IMGNO)=IX
      IYM(IMGNO) = IY
      JORG = -IORG
      CALL ORGIN (IXN(IMGNO),IYN(IMGNO),JORG,IMGNO)
      IMRK(IMGNO) = IORG
      GO TO 200
   40 CALL DRAWI (IX,IY,IORG,IMGNO,ISTAT)
   45 IXM(IMGNO) = 0
      IYM(IMGNO) = 0
      IXN(IMGNO) = 0
      IYN(IMGNO) = 0
      IMRK(IMGNO) = 0
C.....SLAVE PIECE?
      IF(JLOF(JBM(20),IDUM(KFLG+IMGNO)).EQ.1) GO TO 200
      MMM=NN
C.....CHECK FOR SIZE ON PIECE
      IF(IALPH.EQ.0) GO TO 50
      IF(JLOF(JBM(15),IDUM(KFLG+NPEC+IMGNO)).NE.0) MMM=NN+4
   50 IAD = IABS(IDUM(KAD1))+MMM
C.....CHECK FOR FLIP FROM FUNCTION BOX
      IF(IABS(IDUM(KANG+IMGNO)).LT.30000) GO TO 200
      CALL FLIP(IDUM(IAD))
      CALL ORGIN (IX,IY,IORG+10,IMGNO)
C.....ADJUST STRIPE LINE
      IF(ISLY.EQ.0) GO TO 200
      IF(IDUM(KSTL+IMGNO).EQ.9999) GO TO 200
      IDUM(KSTL+IMGNO) = -IDUM(KSTL+IMGNO)
      GO TO 200
C.....NOT VISIBLE
  100 IXM(IMGNO) = 0
      IYM(IMGNO) = 0
      IXN(IMGNO) = 0
      IYN(IMGNO) = 0
      IMRK(IMGNO) = -1
  200 CONTINUE
      ICS = IFIX(2047.0*CS)*16
C.....LOOP TO UPDATE DISPLAY FILE
      DO 500 IMGNO = 1,NUING
      IORG = IMRK(IMGNO)
      IF(IORG.EQ.0) GO TO 500
      KAD1 = KADP+IMGNO
      IAD1 = IDUM(KAD1)
      IF(IORG.GT.0) GO TO 410
C.....NOT VISIBLE
      IF(IAD1.LE.0) GO TO 500
      IF(IMGNO.NE.IMGP) GO TO 300
C.....PICKED PIECE IS OFF SCREEN, RELEASE.
      IDXP = 0
      IDYP = 0
      IMGP = 0
      IAD = KPSIZ
      IDUM(IAD) = 20040B
      IDUM(IAD+1) = 20040B
      IDUM(IAD+2) = 20040B
  300 CALL LITE(-1,KAD1)
      GO TO 500
C.....VISIBLE
  410 IX = IXM(IMGNO)
      IY = IYM(IMGNO)
      NX = IXN(IMGNO)
      NY = IYN(IMGNO)
```

```
      IND = IABS(IAD1)
      IDUM(IND+1)=IRND(IX*CS)*16
      IDUM(IND+2)=IRND(IY*CS)*16+1
C.....CHECK FOR ALPHA OPTION  (DISPLAY FILE DIFFERS IN FIRST MOVE)
      IF(IALPH.EQ.0) GO TO 420
      IDUM(IND+4)=NX*16
      IDUM(IND+5)=NY*16+1
      GO TO 425
  420 IDUM(IND+4)=NX*16+1
      IDUM(IND+5) = NY*16+16B
  425 IF(IAD1.GT.0) GO TO 411
      CALL LITE(1,KAD1)
  411 IF(IMGNO.NE.IMGP) GO TO 500
      IDXP = IX
      IDYP = IY
      IF(IX.GT.-LMT.AND.IX.LT.LMT) GO TO 500
C.....PICKED PIECE IS OFF SCREEN, RELEASE.
      IMGP = 0
      IX = 0
      IY = 0
      IAD = KPSIZ
      IDUM(IAD) = 20040B
      IDUM(IAD+1) = 20040B
      IDUM(IAD+2) = 20040B
      IDXP = 0
      IDYP = 0
  500 CONTINUE
C.....DRAW MARKER OUTLINE AND MATERIAL
      CALL OUTLN
      RETURN
      END
      ENDS


FTN4
      PROGRAM WPROD,3
C.....
C
      DIMENSION NCS0 (3)
      DIMENSION LOT (256)
      DIMENSION NSW(128)
      DIMENSION NBK(5)
      DIMENSION MSG0(10)
      LOGICAL NSTRT
      LOGICAL NCRT
      LOGICAL NHALT
      DIMENSION NMARK(3)
      DIMENSION MMARK (3)
      DIMENSION NCHEH(3)
      DIMENSION NASTY (3)
      DIMENSION NPLOT(3)
      DIMENSION NDRAW(3)
      DIMENSION NGRAB(3)
      DIMENSION NMANI (3)
      DIMENSION NRRUL (3)
      DIMENSION NMES(3)
      DIMENSION NFU(15)
      DIMENSION KEY(36)
      DIMENSION NPLTN(3)
      DIMENSION NXFR(3)
      DIMENSION NTARP(3)
      DIMENSION NOPTN (3)
      DIMENSION NPT(6)
      DIMENSION IRAY(1)
C
```

```
      LOGICAL MCD,M23,MDSC,MDS,M04
C.....
      DATA NCSW/2HSE,2HCT,2HR /
      DATA NPLTN/2HPL,2HOT,2HF /
      DATA NMES/2HME,2HSS,2HG /
      DATA IDOL /2H$ /
      DATA ISEM /2H; /
      DATA ICOL /2H: /
      DATA ICOM /2H, /
      DATA IEQ /2H= /
      DATA ISLSH/2H/ /
      DATA IR   /2HR /
      DATA KEY/2HHA,2HEN,2HME,2HST,2HMA,2HOR,2HPL,2HXF,2HLI,2HOP,
     +     2HPR,2HNO,2HUT,2HCL,2HRO,2HTA,2HMM,2HPA/
      DATA IFL /2HFL/
      DATA IBL /2HBL/
      DATA IST /2HST/
      DATA NOON /32767/
      DATA INE /2HNE/
      DATA IDI/2HDI/
      DATA IWI /2HWI/
      DATA IEF/2HGO/
      DATA ISH /2HSH/
      DATA IX /2HX /
      DATA IY /2HY /
      DATA ICU /2HCU/
      DATA IMA /2HMA/
      DATA IOR/2HOR/
      DATA IFI /2HFI/
      DATA IPL /2HPL/
      DATA ICH/2HCH/
      DATA IGR/2HGR/
      DATA INA/2HNA/
      DATA IOL/2HOL/
      DATA IOP /2HOP/
      DATA IRO /2HRO/
      DATA ITA /2HTA/
      DATA ITI /2HTI/
      DATA IAL /2HAL/
      DATA IGA /2HGA/
      DATA ISC /2HSC/
      DATA ISL /2HSL/
      DATA IBLNK /2H  /
      DATA I37/31/
      DATA NORT /.TRUE./
      DATA LOUT /6/
      DATA NKEY/18/
      DATA NOUSY /2/
      DATA NCHEM /2HCH,2HEC,2HK /
      DATA NMARK /2HMA,2HRK,2H  /
      DATA MMARK /2HMM,2HAR,2HK /
      DATA NASTY/2HWS,2HTY,2HL /
      DATA NPLOT/2HPL,2HOT,2HP /
      DATA NDRAW/2HNO,2HRA,2HW /
      DATA NGRAB /2HPG,2HRA,2HB /
      DATA NMANT/2HMA,2HIN,2HT /
      DATA NBRUL/2HBR,2HUL,2HE /
      DATA NXFR/2HLA,2HSE,2HM /
      DATA NTABP/2HHA,2HLF,2H  /
      DATA NUPTN/2HOP,2HTN,2HS /
      DATA NBK/5*2H  /
C.....
C.....
900   FORMAT (5X,"JOB NOT READY")
901   FORMAT (5X,"COMPLETED")
```

```
0002   FORMAT (1X,"*****FIELD LENGTH ERROR*****")
0003   FORMAT (1X,"*****ILLEGAL KEY WORD OR VALUE*****")
0004   FORMAT (1X,"*****DELIMITER ERROR*****")
0005   FORMAT (1X,"*****FIELD MUST BE AN INTEGER*****")
0006   FORMAT (1X,"*****QUEUE NAME IS NOT A KEYWORD*****")
0007   FORMAT (1X,"*****IDENTIFIER CANNOT BE FOUND IN JOB QUEUE*****")
0008   FORMAT (1X,"*****ILLEGAL FIELD DESCRIPTOR*****")
0009   FORMAT (1X,"*****MISSING PIECE DESCRIPTIONS*****")
0010   FORMAT (1X,"*****MISSING WIDTH VALUE*****")
0011   FORMAT (1X,/,1X,"****DIAGNOSTICS-MAKE CARD READER READY****",//)
0012   FORMAT (1X,"*****ROLL TABLE NOT AVAILABLE*****")
0013   FORMAT (1X,/,1X,"*****CARD READER IS NOT READY*****",//)
0014   FORMAT (1X,/,1X,"*****MOUNT THE MARKER PLATTER*****",//)
0015   FORMAT (1X,/,1X,"****NO. SIZES DOES NOT MATCH****",//)
0016   FORMAT (1X,/,1X,"****NO. STYLES MUST BE EVEN****",//)
0017   FORMAT (1X,/,1X,"****JOB QUEUE IS FULL****",//)
0098   FORMAT (1X,"*****STATEMENT",I6,"*****")
0099   FORMAT (1X,"*****STATEMENT",I6,"    COLUMN",I6,
     *          "    *****")
C.....
C.....
C.....CALL ACCES(1,2HWP)
      CALL LIGHT(-1,0)
      CALL UDSC(0)
      CALL EXEC (14,06,NS0,128,NCS0,0)
      NRTN=NS0(1)
      NSTAT=NS0(2)
100   CONTINUE
      CALL LIGHT (NBUSY,1)
      NQAV=JFLD(15,1)
      NQPU=JFLD(15,1)
      NQAD=JFLD(14,15)
      NQJB=JFLD(14,4)
      NQPR=JFLD(14,3)
      NPIC=JFLD(7,8)
      NFLP=JFLD(15,1)
      NSMF=JFLD(14,1)
      NRLO=JFLD(13,1)
      NREP=JFLD(7,8)
      NDIM=JFLD(15,1)
      NSMA=JFLD(11,4)
      NSTY=JFLD(7,8)
      NEST=JFLD(15,1)
      NCUT=JFLD(14,1)
      NBST=JFLD(4,1)
      NBMA=JFLD(3,1)
      NBPL=JFLD(2,1)
      NBNC=JFLD(1,1)
      NBFI=JFLD(0,1)
      NPLT=JFLD(15,8)
      NROL=JFLD(13,8)
      NMAP=JFLD(14,1)
      HOLD=JFLD(13,1)
      NSTNA=JFLD(10,1)
      NSTOP=JFLD(12,1)
      NSTRO=JFLD(11,1)
      NMFLP=JFLD(12,1)
      NRFLG=JFLD(15,1)
      NCHFG=JFLD(15,8)
      NTAB=JFLD(12,13)
      NSTG=JFLD(15,3)
      NSUF=JFLD(2,3)
      NNPG=JFLD(15,1)
      NNIN=JFLD(14,1)
      NNER=JFLD(13,1)
```

```
      NSLD=JFLD(15,1)
      NWDTH=JFLD(14,15)
      DO 109 I=1,10
109   MSGD(I)=JFLD(I+1,1)
C.....
C.....
C.....SET THE START UP FLAG FALSE
      NSTRT=.FALSE.
      CALL MGST (5,NOPND)
      IF (NRTN.NE.1) GOTO 110
C
C      * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C
C.....
C.....MARKAMATIC SYSTEM INITIAL START UP PROCEDURE
C.....
      NSTRT=.TRUE.
      CALL MGST(-6,1)
      NOPND=1
      CALL DSPLA (11,999)
      CALL MGST (-5,NOPND)
C.....
C.....LOADED MONITOR IS A RESULT OF AN EXECUTED PROGRAM MODULE
C.....
C.....CHECK RETURN FOR ACTIVE
110   IF (NRTN.EQ.3) GOTO 115
C.....CHECK RETURN FOR RETURN
      IF (NRTN.NE.2) GOTO 115
C.....CHECK FOR COMPLETE
      IF (NSTAT.EQ.1) GOTO 112
C.....CHECK FOR NOT READY
      IF (NSTAT.EQ.2) GOTO 111
C.....CHECK FOR NO QUEUE ENTRY
      IF (NSTAT.EQ.3) GOTO 115
C     CHECK FOR SYSTEM ABORT
      IF (NSTAT.EQ.7) GOTO 400
C.....BUILD PLOT (NEST) QUEUE ENTRY
      IF (NSTAT.EQ.5) GOTO 1152
      IF (NSTAT.EQ.6) GOTO 115
C     RETURN IS PLOT OR DRAW QUEUE ENTRY FOR JOB QUEUE
      DO 116 I=1,2
      IF (NS0(I+7).EQ.0) GOTO 116
      CALL MGST (2,NQ)
      CALL MGST (3,ND)
      CALL MGST (-ND,NQ)
      ND=ND+1
      DO 117 J=0,4
      CALL MGST (-(ND+J),NS0(J+3))
117   CONTINUE
      ND=ND+5
      CALL MGST (-ND,100)
      ND=ND+1
      IF (NSTAT.NE.6) GOTO 118
C     SET DRAW ENTRY
      NM=205
      CALL MGST(-ND,600)
      ND=ND+1
      GOTO 119
118   NM=0
      CALL JSTF(1,NEST,NM)
      CALL JSTF(NS0(I+7),NRUL,NM)
119   CALL MGST (-ND,NM)
      ND=ND+1
      CALL MGST (-ND,NDQN)
      ND=ND+1
```

```
      N=0
      CALL JSTF(7,NQJB,N)
      IF (NSTAT.EQ.8) CALL JSTF(6,NQJB,N)
      CALL JSTF(1,NUPR,N)
      CALL WQST (-NQ,N)
      CALL WQST(3,K)
      CALL WQST (-(NQ+1),K)
      NQ=NQ+2
      CALL WQST (-2,ND)
      CALL WQST (-3,ND)
110   CONTINUE
      IF (NSTAT.EQ.8) GOTO 112
      GO TO 115
C
C.....SET PURGE QUEUE ENTRY BIT IN COMPLETED QUEUE ENTRY
C
112   WRITE (LOUT,6001)
      CALL WQST (4,N)
      IF (N.EQ.0) GOTO 1151
113   CALL WQST (N+1,M)
      CALL JSTF(1,NQPU,M)
      CALL WQST (-(N+1),M)
      GOTO 1151
C
C.....SET THE PRIORITY OF THE QUEUE ENTRY TO ZERO
C
111   CALL WQST(4,N)
      IF (N.EQ.0) GOTO 114
      CALL WQST(N,L)
      CALL JSTF(4,NUPR,L)
      CALL JSTF(1,NQAV,L)
      CALL WQST (-N,L)
114   WRITE (LOUT,6000)
      GOTO 113
C
C.....SET COMPLETE FOR PROGRAM
C
115   WRITE (LOUT,6001)
1151  CALL WSCTR(2,6,0)
      GOTO 200
C
C.....SET JOB NOT READY
C
1152  WRITE (LOUT,6000)
      GOTO 240
C
C.....*************************************************************************
C.....
C.....BEGINS MAIN LOOP OF THE PRODUCTION MONITOR
C.....
200   HALT=.FALSE.
      CALL LIGHT (NBUSY,1)
      IDLE=0
      CALL WQST(1,N)
      NPGB=JLOF(NNPG,N)
      NPGIN=JLOF(NNIN,N)
      NER=JLOF(NNER,N)
      NMG=0
      NRULE=0
212   CALL WCRS(ISTAT)
C.....  CHECK CARD READER READY
      M=JFLD(15,1)
      L=JLOF(M,ISTAT)
      IF(L.EQ.0) GO TO 201
      GO TO 211
```

```
C.....
C.....READ THE CARD READER FOR JOB REQUEST INPUT
C.....AND ADD THE ENTERIES TO THE WORK QUEUE
C.....
211   CALL DSPLA(0)
C
C.....PURGE AND PACK THE QUEUE REQUESTS AND PACK THE QUEUE
C
      CALL WQST (2,N)
      NQ=7
      NQQ=257
      N=N-2
      IF (N.LT.7.OR.N.GT.255) GOTO 125
      DO 120 I=7,N,2
      CALL WQST (I,L)
      IF (JLOF(NQPR,L).EQ.0) GOTO 120
      CALL WQST ((I+1),L)
      IF (L.LT.257) GOTO 120
C.....MOVE THE QUEUE ENTRY
      DO 121 K=0,1
      CALL WQST (I+K,J)
      CALL WQST (-(NQ+K),J)
121   CONTINUE
C.....MOVE THE DATA FOR THE QUEUE ENTRY
      CALL WQST (-NQQ,NQ)
      DO 122 M=1,256
      CALL WQST (L+M,LOT(M))
      IF (LOT(M).EQ.NOQN) GOTO 126
122   CONTINUE
126   DO 129 MM=1,M
      CALL WQST (-(NQQ+MM),LOT(MM))
129   CONTINUE
123   CALL WQST (-(NQ+1),NQQ)
      NQQ=NQQ+M+1
      NQ=NQ+2
120   CONTINUE
125   CALL WQST (-2,NQ)
      CALL WQST (-3,NQQ)
      DO 128 I=NQ,256
      CALL WQST (-I,0)
128   CONTINUE
C
C.....CLEAR THE NOT READY BITS IN QUEUE
C
      DO 127 I=7,NQ,2
      CALL WQST (I,L)
      J=JLOF(NQJB,L)
      IF (J.GT.7.AND.J.LT.12) GOTO 127
      CALL JSTF(0,NQAV,L)
      CALL JSTF(1,NQPR,L)
      CALL WQST (-I,L)
127   CONTINUE
      NSTMT=4
C.....GET THE KEYWORD FIELD, STATEMENT POSITION SHOULD
C.....CURRENTLY BE AT THE  $  COLUMN
1000  NSTMT=NSTMT+1
1001  CALL NXF (NFD)
      IF (NFD(2).EQ.3) GOTO 1001
      N=NFD(3)
      IF (N.EQ.ICOL.OR.N.EQ.IDOL) GOTO 1002
      GOTO 1071
1002  IF (NFD(2).EQ.2) GOTO 1070
C.....PACK THE FIELD AND CHECK FOR KEY WORDS
1010  CALL NPK(NFD(5),2,NFD(5),1)
      K=NFD(5)
```

```
      DO 1011 I=1,NKEY
      IF (K.EQ.KEY(I)) GOTO 1020
1011  CONTINUE
C.....KEYWORD ERROR
      GOTO 1072
C.....GO TO APPROPRIAYE KEYWORD FUNCTION
1020  CALL RQST(2,NQ)
      CALL RQST (3,ND)
C.....SET QUEUE BACK POINTER
      CALL RQST (-ND,NQ)
      ND=ND+1
      GOTO (1100,1200,2600,1400,1500,1600,1700,1800,1900,2000,2100,
     +       2200,2300,2400,2500,2700,1500,2800          ),I
C.....
C.....SET FIELD ERROR DIAGNOSTIC
C.....
1070  WRITE (LOUT,6002)
      GOTO 1090
C.....
C.....SET DELIMITER ERROR DIAGNOSTIC
C.....
1071  WRITE (LOUT,6004)
      GOTO 1090
C.....
C.....KEYWORD ERROR DIAGNOSTIC
C.....
1072  WRITE (LOUT,6003)
      GOTO 1090
C.....
C.....FIELD IS NOT AND INTEGER DIAGNOSTIC
C.....
1073  WRITE (LOUT,6005)
      GOTO 1090
1078  WRITE (6,5008)
      GOTO 1090
C.....STORE THE QUEUE ENTRY AND THE END OF DATA
C.....
C.....STORE THE END OF DATA
C.....
1080  CALL RQST(-ND,NDON)
      ND=ND+1
C.....STORE QUEUE ENTRY
      N=0
      CALL JSTF (NT,NOJB,N)
      CALL JSTF(1,NOPR,N)
      CALL RQST (2,J)
      CALL RQST (-J,N)
      CALL RQST (3,K)
      CALL RQST(-(J+1),K)
C.....UPDATE POINTERS
C.....CHECK FOR JOB QUEUE OVERFLOW
      IF (J.LT.255) GOTO 1082
      WRITE (LOUT,6017)
      GOTO 1000
1082  J=J+2
      CALL RQST (-2,J)
      CALL RQST (-3,ND)
      NPGB=1
      GOTO 1000
C.....
C.....OUTPUT DIAGNOSTIC MESSAGE AND GO TO NEXT $
C.....
1090  NERFO(15)=1
      NER=1
      IF (N.EQ.0) GOTO 10901
```

```
      WRITE (6,6099) NSTMT,N
      GOTO 1091
10901 WRITE (6,6098) NSTMT
1091  IF (NFD(3).EQ.IDOL) GOTO 1000
      CALL AGF(NFD)
      GOTO 1091
C.....
C.....
C.....PROCESS THE $HALT STATEMENT
C.....
1100  NHALT=.TRUE.
      GOTO 1000
C.....
C.....PROCESS THE $END STATEMENT
C.....
1200  NPGIN=1
      IF (NER.EQ.0) GOTO 201
      WRITE (6,6011)
      NER=0
      CALL SFLG (NPGB,NPGIN,NER)
1202  CALL WCRS(ISTAT)
      M=JFLD(15,1)
      L=JLOF(M,ISTAT)
      IF (L.EQ.0) GOTO 1202
      GOTO 1000
C
C.....
C.....PROCESS THE $STYLE STATEMENT AND BUILD THE JOB QUEUE ENTRY
C.....TEMPORARY LOCATIONS NP=NUMBER OF PIECES ENCOUNTERED
C.....                    NQ=POSITION FOR THEIS QUEUE ENTRY
C.....                    ND=DATA POSITION FOR THIS QUEUE ENTRY
C.....                    NR=LOCATION FOR PIECES AND RULE
C.....                    NT=TYPE OF JOB
C.....
C.....SET THE QUEUE POINTER IN THE DATA AREA
1400  NT=4
C.....GET THE STYLE IDENTIFICATION AND STORE IT IN THE DATA AREA
      CALL NGF(NFD)
C.....CHECK FIELD FOR ERROR
      IF (M23(NFD(2))) GOTO 1070
      IF (NFD(3).NE.ICOM) GOTO 1071
C.....STORE STYLE IDENTIFICATION IN QUEUE
      CALL NSPD(NFD,ND)
C.....SET UP FOR NUMBER OF PIECES AND RULE NUMBER
      NR=ND
      ND=ND+1
C.....GET THE STYLE RULE NUMBER
      CALL NGF(NFD)
C.....RULE NUMBER MUST BA NAME
      IF(NFD(2).NE.1) GO TO 1070
      IF (NFD(3).NE.ISEM) GOTO 1071
      CALL NPK(NFD(5),10,NRT(1),1)
      CALL DIREC(10006,NRT,INTNO,NSXTR,ISTAX)
      IF(ISTAX.EQ.0)GO TO 1409
      WRITE (LOUT,6012)
      GOTO 1091
1409  CALL DIREC(34006,NRT,INTNO,NSXTR,ISTAX)
      NP=NRT(6)*256
C.....GET EACH PIECE DATA AND STORE IN THE DATA QUEUE AREA
1410  CALL NGF(NFD)
      IF (M23(NFD(2))) GOTO 1070
      IF (MDSC(NFD(3))) GOTO 1411
      GOTO 1071
1411  N=0
      KX=0
```

```
      KY=0
      KTA=0
      KTI=3100
C.....STORE PIECE IDENTIFICATION IN DATA AREA
      CALL NSPD(NFD,ND)
      NP=NP+1
      IF (M03(NFD(3))) GOTO 1400
C.....PICK OFF THE PARAMETERS AND STORE IN QUEUE
1420  CALL NGF(NFD)
      IF (M23(NFD(2))) GOTO 1070
C.....CHECK THE PARAMETER FIELD
      CALL KPK(NFD(5),2,K,1)
      IF (K.NE.IFL) GOTO 1421
      CALL JSTF (1,NFLP,N)
      GOTO 1430
1421  IF (K.NE.INA) GOTO 1422
      CALL JSTF(1,NSTNA,N)
      GOTO 1430
1422  IF (K.NE.IBL) GOTO 1423
      CALL JSTF(1,NBLO,N)
      GOTO 1430
1423  IF(K.NE.ISH) GO TO 14231
      CALL JSTF (1,NSHF,N)
      GO TO 1430
14231 IF (K.NE.IOP) GOTO 14232
      CALL JSTF(1,NSTOP,N)
      GOTO 1430
14232 IF (K.NE.IRO) GOTO 1424
      CALL JSTF(1,NSTRO,N)
      GOTO 1430
C.....PARAMETER WORD IS NOT FLIP, SHADE, BLOCK, OPPOSITE, ROTATE, NAP
1424  IF(K.EQ.IX.OR.K.EQ.IY.OR.K.EQ.ITI
     1.OR.K.EQ.ITA.OR.K.EQ.IST) GO TO 14241
C.....CHECK FIELD FOR MESSAGES FOR PIECE
      IF (M04(NFD(2))) GOTO 14243
      GOTO 1478
14243 L=NFD(4)/100
      IF (L.LT.1 .OR. L.GT.10) GOTO 1073
      CALL JSTF(1,MSGD(L),N)
      GOTO 1430
14241 CALL NGF(NFD)
      IF (M04(NFD(2))) GOTO 14242
      GOTO 1470
14242 IF (K.NE.IX) GOTO 1425
      KX=NFD(4)
      GOTO 1430
1425  IF (K.NE.IY) GOTO 14251
      KY=NFD(4)
      GOTO 1430
14251 IF (K.NE.ITA) GOTO 14252
      CALL JSTF((NFD(4)/100),NTAB,KTA)
      GOTO 1430
14252 IF (K.NE.ITI) GO TO 14253
      KTI=NFD(4)
      GOTO 1430
14253 IF(K.NE.IST) GO TO 1078
      CALL JSTF((NFD(4)/100),NSTG,KTA)
      GO TO 1430
C.....CHECK FOR END OF PIECE, STATEMENT OR PARAMETER
1430  M=NFD(3)
      IF (M.EQ.ICOM) GOTO 1420
      IF (M.EQ.ISEM) GOTO 1480
      IF (M.NE.IDOL) GOTO 1071
C.....STORE PARAMETER WORD
1480  CALL NGST(-ND,N)
```

```
      ND=ND+1
      CALL WQST(-ND,KX)
      ND=ND+1
      CALL WQST(-ND,KY)
      ND=ND+1
      CALL WQST(-ND,KTA)
      ND=ND+1
      CALL WQST(-ND,KTI)
      ND=ND+1
C.....CHECK FOR END OF STATEMENT
      IF (NFD(5).NE.IDOL) GOTO 1410
      CALL RQST(-NR,NP)
      IF (JLOF(NPIC,NP).GT.0) GOTO 1080
      WRITE (LOUT,6409)
      GOTO 1091
C.....
C.....PROCESS THE $MARK STATEMENT AND BUILD THE JOB
C.....WORK QUEUE ENTRY
C.....TEMPORARY LOCATIONS NQ=POSITION FOR THIS QUEUE ENTRY
C.....                    ND=DATA POSITION FOR THIS QUEUE ENTRY
C.....                    NT=TYPE OF JOB (5)
C.....                    NR=ADDRESS FOR WIDTH WORD
C.....                    NS=NUMBER OF STYLES
C.....                    NP=ADDRESS OF SIZE WORD
C.....                    NPN=NUMBER OF SIZES
C.....
C.....SET THE QUEUE POINTER IN THE DATA AREA
C.....
1500  NT=5
      IF(I.EQ.17) NT=9
C.....GET THE MARKER IDENTIFICATION
      CALL RGF(NFD)
      IF (M23(NFD(2))) GOTO 1070
      IF (NFD(3).NE.ICOM) GOTO 1071
C.....STORE THE MARKER IDENTIFICATION IN THE QUEUE
      CALL WSPD(NFD,ND)
C.....SET UP FOR PARAMETERS
      NR=ND
      ND=ND+6
      M=4
      DO 1510   I=0,7
      CALL WQST(-(NR+I),0)
1510  CONTINUE
      CALL WQST (-(NR+1),9,00)
C.....GET THE MARKER PARAMETERS
1520  CALL RGF(NFD)
      IF (M23(NFD(2))) GOTO 1070
      N=NFD(3)
      IF (N.EQ.ICOM.OR.N.EQ.IEQ.OR.N.EQ.ISEM) GOTO 1521
      GOTO 1071
C.....PACK THE KEYWORD
1521  CALL WPK (NFD(5),2,KEYW,1)
C.....IS THE KEYWORD A DOUBLE - KEY=VALUE?
      IF (N.EQ.IEQ) GOTO 1530
C.....CHECK FOR DIMENSION REQUEST
      IF (KEYW.NE.INA) GOTO 1524
      CALL JSTF(1,NNAP,M)
      GOTO 1526
1524  IF (KEYW.NE.IOL) GOTO 1525
      CALL JSTF(1,NOLD,M)
      GOTO 1526
1525  IF (KEYW.NE.IDI) GOTO 15251
C.....SET THE DIMENSION REQUEST
      CALL JSTF(1,NDIM,M)
      GOTO 1526
```

```
C.....CHECK FOR FLIP REQUEST
15251 IF (KEYW .NE. IFL) GOTO 15252
      CALL JSTF(1,NMFLP,M)
      GOTO 1525
15252 IF (KEYW .NE. ISL) GOTO 1072
      CALL JSTF(1,NSLD,IWDTH)
      GOTO 1525
C.....REQUEST IS A TWO WORD ENTRY - KEY = VALUE
C.....GET THE VALUE FIELD
1530  CALL WGF(LOT)
      NFD(15)=LOT(15)
      NFD(1)=LOT(1)
      IF (M23(LOT(2))) GOTO 1070
      N=LOT(3)
      IF (N.EQ.ICOM.OR.N.EQ.ISEM.OR.N.EQ.ISLSH) GOTO 1531
      GOTO 1071
C.....IS THE KEYWORD = WIDTH
1531  IF (KEYW.NE.IWI) GOTO 1532
      CALL JSTF(LOT(4),NWDTH,IWDTH)
      GOTO 1525
C.....IS THE KEYWORD = GUAL
1532  IF (KEYW.NE.IEF) GOTO 1533
      NYD=LOT(4)
      LOT(4)=0
      IF (LOT(3).NE.ISLSH) GOTO 15321
      CALL WGF(LOT)
      NFD(15)=LOT(15)
      NFD(1)=LOT(1)
      IF (M23(LOT(2))) GOTO 1070
      N=LOT(3)
      IF (N.EQ.ICOM.OR.N.EQ.ISEM) GOTO 15321
      GOTO 1070
15321 NYD=(NYD/100*30+LOT(4)/100)*10
      CALL WOST(-(NR+1),NYD)
      GOTO 1525
C.....IS THE KEYWORD = SHADE
1533  IF (KEYW.NE.ISH) GOTO 1534
      J=LOT(4)/100
      CALL JSTF(J,NSHA,M)
      GOTO 1525
C.....IS THE KEYWORD = X
1534  IF (KEYW.NE.IX) GOTO 1535
      CALL WOST(-(NR+3),LOT(4))
      GOTO 1525
C.....IS THE KEYWORD = Y
1535  IF (KEYW.NE.IY) GOTO 1536
      CALL WOST(-(NR+4),LOT(4))
      GOTO 1525
C.....IS THE KEYWORD=STRIPE
1536  IF (KEYW.NE.IST) GOTO 1537
      CALL WOST(-(NR+6),LOT(4))
      LOT(4)=0
      IF (LOT(3).NE.ISLSH) GOTO 15361
      CALL WGF(LOT)
      NFD(15)=LOT(15)
      NFD(1)=LOT(1)
      IF (M23(LOT(2))) GOTO 1070
      N=LOT(3)
      IF (N.EQ.ICOM.OR.N.EQ.ISEM) GOTO 15361
      GOTO 1070
15361 CALL WOST(-(NR+5),LOT(4))
      GOTO 1525
C.....IS THE KEYWORD=PLAID
1537  IF (KEYW.NE.IPL) GOTO 1072
      CALL WOST (-(NR+7),LOT(4))
```

```
      GOTO 1520
C.....ARE WE FINISHED WITH THE KEYWORD PARAMETERS
1520  IF (N.NE.ISEM) GOTO 1520
C.....CHECK FOR WIDTH=0
      CALL WGST(-NR,IWDTH)
      N=JLOF(NWDTH,IWDTH)
      IF (N.NE.0) GOTO 1539
      WRITE (LOUT,6010)
      GOTO 1091
1539  CONTINUE
C.....SET UP THE STYLE AND SIZE DEFINITION REQUESTS
C.....FOR THE MARKER QUEUE ENTRY
      MM=M
1540  NS=0
1540  CALL WGF (NFD)
      IF (M23(NFD(2))) GOTO 1070
      IF (JLOF(NOLD,MM).EQ.1) GOTO 1555
      IF (NFD(3).NE.ICOM) GOTO 1071
C.....STORE STYLE IDENTIFICATION
      CALL WSFD(NFD,ND)
C.....SET UP VARIABLES
      NS=NS+1
      NP=ND
      ND=ND+1
      NPN=0
C.....GET THE SIZES FOR THE STYLE
1550  CALL WGF(NFD)
      IF (M23(NFD(2))) GOTO 1070
      N=NFD(3)
      IF (MOSC(N)) GOTO 1551
      GOTO 1071
C.....A GOOD SIZE WAS LOCATED
1551  NPN=NPN+1
      K=0
      IF (NFD(5).EQ.IR) K=1
      CALL WPK (NFD(K+5),6,NFD(5),1)
      CALL JSTF (K,NRFLG,NFD(5))
      DO 1552 I=0,2
      CALL WGST(-(ND+I),NFD(I+5))
1552  CONTINUE
      ND=ND+3
C.....IS THIS SIZE THE LAST OF THE SET
      IF (N.EQ.ICOD) GOTO 1550
C.....CHECK FOR MATCHING SIZES FOR EACH PAIR OF SUBSTITUTIONS
      IF (JLOF(NOLD,MM) .EQ. 0) GOTO 1556
      IF (NS-(NS/2*2) .NE. 0) GOTO 1556
      IF (NSZL .EQ. NPN) GOTO 1556
      WRITE (LOUT,6015)
      GOTO 1090
1556  NSZL=NPN
      CALL WGST (-NP,NPN)
      IF (N.EQ.ISEM) GOTO 1540
C.....END OF STATEMENT
C.....STORE NUMBER OF STYLES
1554  CALL JSTF(NS,NSTY,M)
      CALL WGST(-(NR+2),M)
      IF (JLOF(NOLD,MM) .EQ. 0) GOTO 1080
      IF (NS-(NS/2*2) .EQ. 0) GOTO 1080
      WRITE (LOUT,6016)
      GOTO 1090
C
C.....SET UP OLD MARKER IDENTIFICATION
C.....AND SIZE SUBSTITUTION
C
1555  N=NFD(3)
```

```
      IF (MDS(N)) GOTO 1560
      GOTO 1071
1500  CALL WSPD (NFD,ND)
      MM=0
      IF (N.EQ.IDOL) GOTO 1554
      GOTO 1540
C.....PROCESS THE 3DRAW STATEMENT AND BUILD THE JOB QUEUE ENTRY
C.....TEMPORARY LOCATIONS NQ=NEXT QUEUE ENTRY POSITION
C.....                    ND=NEXT DATA ENTRY POSITION
C.....                    NT=#PLOT (6)
C.....                    NM=NUMBER OF MESSAGES WORD
C.....                    NR=ADDRESS FOR NM
C.....
C.....STORE THE QUEUE POINTER IN THE DATA QUEUE AREA
C.....
1600  NM=0
      NSC=100
      NGAP=000
      CALL JSTF(IBLNK,NCHFG,NM)
      NT=0
C.....GET MARKER IDENTIFICATION TO PLOT
      CALL NGF(NFD)
      IF (M23(NFD(2))) GOTO 1070
      IF (NOSC(NFD(3))) GOTO 1610
      GOTO 1071
C.....STORE THE MARKER IDENTIFICATION TO BE PLOTTED
1610  CALL WSPD(NFD,ND)
      NR=ND
C.....CHECK FOR PARAMETERS ALPHA, GAP, SCALE
      ND=ND+2
      N=NFD(3)
      IF(N.EQ.IDOL) GO TO 1619
      IF(N.EQ.ISEM) GO TO 1613
1611  CALL NGF(NFD)
      IF(NFD(2).NE.1) GO TO 1072
      CALL WPK(NFD(5),2,LOT(1),1)
      IF(NFD(3).NE.IEQ) GO TO 1071
      CALL NGF(NFD)
      IF(M23(NFD(2))) GO TO 1070
      IF(LOT(1).NE.IAL) GO TO 1621
C.....STORE ALPHA CHARACTER
      M=JLDF(NCHFG,NFD(5))
      CALL JSTF(M,NCHFG,NM)
      GO TO 1629
1621  IF(LOT(1).NE.IGA) GO TO 1622
C.....STORE GAP VALUE
1623  NGAP=NFD(4)
      GO TO 1629
1622  IF(LOT(1).NE.ISC) GO TO 1072
      NSC=NFD(4)
1629  IF(NFD(3).EQ.IDOL) GO TO 1619
      IF (NFD(3).NE.ISEM) GOTO 1611
C.....GET THE MESSAGE NUMBERS
1613  ND=ND+1
      CALL NGF(NFD)
      IF (M23(NFD(2))) GOTO 1070
      N=NFD(3)
      IF (MCO(N)) GOTO 1615
      GOTO 1071
1615  IF (NFD(2).NE.0) GOTO 1070
      NM=NM+1
      CALL WQST(-ND,NFD(4))
      IF (N.NE.IDOL) GOTO 1613
1619  CALL WQST(-NR,NSC)
      CALL WQST (-(NR+1),NGAP)
```

```
      CALL WGST (-(NR+2),NM)
      ND=ND+1
      GOTO 1000
C.....PROCESS THE $PLOT STATEMENT AND BUILD THE
C.....WORK QUEUE ENTRY
C.....TEMPORARY LOCATIONS NQ=CURRENT JOB QUEUE POSITION
C.....                    ND=CURRENT DATA QUEUE AREA
C.....                    NM=NUMBER OF SIZES WORD
C.....                    NR=ADDRESS FOR NUMBER OF SIZES PER WORD
C.....
1700  NT=7
      NM=0
      CALL JSTF (1,NEST,NM)
      NSCL=100
C     GET PIECE IDENTIFICATION
      CALL WGF (NFD)
      IF (M23(NFD(2))) GOTO 1070
      CALL WSPD (NFD,ND)
      NR=ND+1
      ND=ND+2
      N=NFD(3)
1710  IF(N.EQ.IDOL) GOTO 1770
      IF(N.EQ.ISEM) GO TO 1760
C     PROCESS THE PARAMETERS (CUT,NEST,RULE,SC=VALUE)
      CALL WGF (NFD)
      IF (M23(NFD(2))) GOTO 1070
      N=NFD(3)
      CALL WPK (NFD(5),10,NFD(5),1)
      IF (N.NE.IEQ) GOTO 1720
C     PARAMETER SHOULD BE SC=VALUE
      IF (NFD(5).NE.ISC) GOTO 1072
      CALL WGF (NFD)
      IF (.NOT.M04(NFD(2))) GOTO 1070
      NSCL=NFD(4)
      N=NFD(3)
      GO TO 1710
C     CHECK FOR RULE TABLE MATCH
1720  CALL DIREC (19000B,NFD(5),INTO,NSXTR,ISTAX)
      IF (ISTAX.NE.0) GOTO 1730
      CALL DIREC (34000B,NFD(5),INTO, NSXTR,ISTAX)
      CALL JSTF (NFD(10),NRUL,NM)
      GOTO 1710
C     CHECK FOR CUT OR NEST
1730  IF (NFD(5).EQ.INE) GOTO 1710
      IF (NFD(5).NE.ICU) GOTO 1072
      CALL JSTF (0,NEST,NM)
      CALL JSTF (1,NCUT,NM)
      GOTO 1710
C     GET READY FOR SIZE SPECIFICATIONS
1760  CALL WGF (NFD)
      IF (M23(NFD(2))) GOTO 1070
      N=NFD(3)
      IF (.NOT.MCD(N)) GOTO 1071
C     STORE SIZE
      NM=NM+1
      CALL WPK (NFD(5),6,NFD(5),1)
      DO 1762 I=0,2
      CALL WQST (-(ND+I),NFD(I+5))
1762  CONTINUE
      ND=ND+3
      IF (N.EQ.ICOM) GOTO 1760
C     FINISH QUEUE ENTRY
1770  CALL WQST (-NR,NM)
      CALL WQST (-(NR-1),NSCL)
      GOTO 1000
```

```
C
C
C.....PROCESS THE $XFER STATEMENT
C
C       $XFER:1         TRANSFER MARKER DATA
C       $XFER:2         TRANSFER PIECE DATA
C       $XFER:3         MARKER STATUS
C       $XFER:4         PIECE STATUS
C       $XFER:5         CHANGE MARKER STATUS
C       $XFER:6         CHANGE PIECE STATUS
C
1800    NT=8
        CALL WGF(NFD)
        IF (M23(NFD(2))) GOTO 1070
        NP=NFD(4)/100
        IF (NP.EQ.5 .OR. NP.EQ.6) GOTO 1850
C.....MAKE QUEUE ENTRY
        CALL WQST(-ND,NP)
        ND=ND+1
        GOTO 1000
C.....CALL THE XFER PROGRAM FOR ITEM 5 OR 6
1850    CALL WQST(-4,0)
        CALL SFLG(NPGB,NPGIN,NER)
        CALL DSPLA(0)
        CALL WQST (0,N)
        CALL WSCTR(NP,0,0)
        CALL WRITE (N8K)
1859    FORMAT (1X,/,T3,2X,"XFER",2X,5A2)
        CALL ACCES(0,NXFR)
        CALL EXEC (10,NXFR)
C
C
C.....PROCESS THE $LIST STATEMENT AND OUTPUT THE QUEUE
C.....
1900    CALL WDMP(LOT)
        GOTO 1000
C
C
C.....PROCESS THE OPERATOR NUMBER
C
2000    CALL WGF(NFD)
        IF (M04(NFD(2))) GOTO 2001
        GOTO 1070
2001    NOPNO=NFD(4)/100
        CALL WQST (-5,NOPNO)
        GOTO 1000
C
C.....PROCESS THE $PRIORITY WORK QUEUE ENTRY
C
2100    CALL WGF (NFD)
        IF (M23(NFD(2))) GOTO 1070
        CALL WRK (NFD(5),2,K,1)
C.....CHECK TYPE OF JOB QUEUE TO SEARCH
        DO 2101 N=4,7
        IF (KEY(N).EQ.K) GOTO 2102
2101    CONTINUE
        WRITE (LOUT,6006)
        GOTO 1090
C.....JOB TYPE IS VALUE N
2102    CALL WGF (NFD)
        IF (NFD(2).NE.0) GOTO 1070
        NP=NFD(4)/100
C.....NP=NEW PRIORITY VALUE
2110    IF (NFD(5).EQ.IDOL) GOTO 1000
        CALL WGF(NFD)
```

```
      IF (N25(NFD(2))) GOTO 1079
      CALL WPK(NFD(5),10,NFD(5),1)
C.....CHECK QUEUE FOR ID AND TYPE ENTRY
      K=7
2112  IF (K.GT.255) GO TO 2120
      CALL WQST (K,L)
      IF (JLOF(NWJB,L).NE.M) GOTO 2115
      CALL WQST (K+1,L)
      IF (L.LT.257) GOTO 2115
      DO 2114 I=1,5
      CALL WQST (L+I,M)
      IF (M.NE.NFD(I+4)) GOTO 2115
2114  CONTINUE
C.....SET THE NEW PRIORITY
      CALL WQST (K,L)
      CALL JSTF (NP,NQPR,L)
      CALL WQST (-K,L)
      IF (NP.NE.0) GOTO 2110
      CALL WQST(K+1,L)
      CALL JSTF(1,NQPU,L)
      CALL WQST(-(K+1),L)
      GOTO 2110
C.....INCREMENT TO NEXT QUEUE ENTRY
2115  K=K+2
      GOTO 2112
C.....IDENTIFIER NOT FOUND IN QUEUE FOR JOB TYPE
2120  WRITE (LOUT,6007)
      GOTO 1490
C
C.....PROCESS THE NO GRADE OPTION REQUEST
C
2200  NPGB=0
      GOTO 1900
C
C.....PROCESS THE $UTILITY STATEMENT
C.....AND CALL THE MAINTENANCE PROGRAM
C
2300  CALL WQST (-4,0)
      CALL SFLG(NPGB,NPGIN,NER)
      CALL DSPLA (0)
      CALL WQST (0,N)
      CALL UDSC(0)
      CALL WRITE (NBK)
2309  FORMAT (1X,/,I3,2X,"UTILITY",2X,5A2)
      CALL ACCES(0,NMANT)
      CALL EXEC(10,NMANT,99)
C.....
C.....PROCESS THE $CLEAR JOB REQUEST
C.....CLEAR THE WORK QUEUE
2400  DO 2410 I=7,256
      CALL WQST(-I,0)
2410  CONTINUE
      CALL WQST (-2,7)
      CALL WQST(-3,257)
      CALL WQST(-4,0)
      CALL WQST(-5,0)
      GOTO 1900
C
C
C.....PROCESS THE $RULE PROGRAM STATEMENT
C
2500  CALL WQST (-4,0)
      CALL SFLG(NPGB,NPGIN,NER)
      CALL DSPLA(0)
      CALL WQST (0,N)
```

```
      CALL WSCTR(2,3,0)
      CALL WRITE (NBK)
2549  FORMAT (1X,/,I3,2X,"RULE",2X,5A2)
      CALL ACCES(0,NWRUL)
      CALL EXEC (10,NWRUL,99)
C
C.....READ THE MESSAGES INTO THE DISC MESSAGE
C.....AREA.  THE FORMAT OF EACH MESSAGE IS
C.....  AN THE MESSAGE LENGTH IS 78 CHARACTERS
C.....THE FIRST 10 MESSAGES GO ON SPECIFIC PIECES
C.....THE MESSAGES 11-60 GOT ON THE MARKER HEADING
C
C.....AN=MESSAGE NUMBER
C.....MESSAGE SECTOR MS=(M+2)/3-1
C.....MESSAGE LOCATION IN SECTOR  ML=((M-(MS*3))-1)*40
C
C
2600  DO 2601 I=1,40
2601  LOT (I)=NBK
      READ (5,2599) M,(LOT(I),I=1,39)
2599  FORMAT (I2,39A2)
      IF (M.LT.1) GOTO 1000
      IF (M.GT.60) GOTO 2620
      MS=(M+2)/3-1
      CALL EXEC (14,66,NS0,128,NMES,MS)
      ML=((M-MS*3)-1)*40
      DO 2605 I=1,40
2605  NS0(ML+I)=LOT(I)
      CALL EXEC (15,66,NS0,128,NMES,MS)
      GOTO 2600
2620  WRITE (6,2621) M
2621  FORMAT (1X,"MESSAGE NUMBER ",I3,"  OUT OF RANGE")
      GOTO 2600
C
C
C.....PROCESS THE $TABLE: STATEMENT
C
2700  CALL WWST (-4,0)
      CALL SELG(NPGS,NPGIN,NER)
      CALL DSPLA (M)
      CALL WWST (M,N)
      CALL WSCTR(2,3,M)
      CALL WRITE (NBK)
2709  FORMAT (1X,/,I3,2X,"TABLE",2X,5A2)
      CALL ACCES(0,NTABP)
      CALL EXEC (10,NTABP,99)
C
C.....PROCESS THE $PAPER: WIDTH,SLEW  STATEMENT
C
C
2800  CALL WGF(NFD)
      N=NFD(2)
      IF (M23(N)) GOTO 1072
      NWDTH=NFD(4)
      NSLW=0
      IF (NFD(3) .EQ. IDOL) GOTO 2810
C.....GET THE ADVANCE FACTOR
      CALL WGF(NFD)
      N=NFD(2)
      IF (M23(N)) GOTO 1070
      IF (N.NE.1) GOTO 1072
      NSLW=NFD(4)
      IF (NFD(3).NE.IDOL) GOTO 1071
C.....STORE WORDS IN OPTION FILE
2810  CALL EXEC (14,66,LOT,128,NOPTN,0)
```

```
      LOT(10) = NWDTH
      LOT(11) = NSLW
      CALL EXEC (15,66,LOT,128,NOPTN,0)
      GOTO 1000
C.....
201   CONTINUE
      IF (NHALT) GOTO 400
      IF (NPGIN.EQ.1) GOTO 273
      IF (.NOT.NSTRT) GOTO 300
      CALL WSCTR(2,3,0)
C
C.....GET THE PIECES FROM THE PIECE FILES
C
273   CALL WRITE (NRK)
9059  FORMAT (1X,/,I3,2X,"TRANSFER AND GRADE PIECES",5A2)
      CALL DSPLA(0)
      CALL DDSC(0)
      CALL SFLG(0,0,0)
      CALL WSCTR(2,3,0)
      CALL LIGHT (NBUSY,1)
      CALL WNST (0,N)
      CALL ACCES(0,NGRAB)
      IF (NPGR.EQ.0) CALL EXEC(10,NGRAB,99)
      CALL EXEC(10,NGRAB,1)
C
C
C.....INITIALIZE PLOT POINTERS
C
300   CALL WGST(1,N)
      N=JLOF(NSUF,N)
      CALL LIGHT (-1,0)
      IF (N.EQ.1) GOTO 308
      GOTO 305
  308 DO 303 I=1,128
      LOT(I)=0
303   CONTINUE
      CALL DDSC(0)
      CALL EXEC(15,66,LOT,128,NPLTN,0)
      CALL WGST(1,N)
      CALL JSTF(0,NSUF,N)
      CALL WQST(-1,N)
C
C.....CHECK FOR A HALT - PROCESS QUEUE ENTERIES
C
305   IF (NHALT) GOTO 400
C
C
C.....GET THE NEXT JOB
C
      GOTO 329
C.....CHECK FOR SPACE IN PLOT BUFFER
C
  306 CALL EXEC(18,NPLTN,N)
      CALL PLTST(-3,L)
      CALL PLTST(-4,M)
307   K=L-M
      IF (0.GE.L) K=(N-M)+L-1
      IF (K.GT.350) GOTO 310
      GOTO 350
C
C
C
C.....PROCESS THE STYLE QUEUE
C
329   K=2
```

```
      CALL NXTJB(K,NA)
      IF (NA.EQ.0) GOTO 330
      CALL DSPLA(0)
C.....CALL THE STYLE PROGRAM
      CALL WRITE (NSK)
9001  FORMAT (1X,/,I3,2X,"STYLE",2X,5A2)
      CALL WJST (0,N)
      CALL WSCTR(2,3,0)
      CALL LIGHT (NBUSY,1)
      CALL ACCES(0,NASTY)
      CALL EXEC(10,NASTY)
C
C
C.....CALL A PLOT OR DRAW PROGRAM
C
310   K=1
      CALL NXTJB(K,NA)
      IF (NA.EQ.0) GOTO 350
      CALL LIGHT (NBUSY,1)
      IF (K.EQ.7) GOTO 312
      CALL DSPLA(0)
      DO 311 I=1,5
      CALL WJST(NA+I,LOT(I))
311   CONTINUE
      CALL DIREC(10001B,LOT(1),I,I,N)
      IF (N.NE.0) GOTO 3100
      CALL WRITE (LOT)
9002  FORMAT (1X,/,I3,2X,"DRAW",2X,5A2)
      CALL WJST (0,N)
      CALL WSCTR(NA,0,0)
      CALL ACCES(0,NDRAW)
      CALL EXEC(10,NDRAW)
C
312   CALL DSPLA(0)
      DO 313 I=1,5
      CALL WJST(NA+I,LOT(I))
313   CONTINUE
      CALL DIREC(10004B,LOT(1),I,I,N)
      IF (N.NE.0) GOTO 3100
      CALL WRITE (LOT)
9003  FORMAT (1X,/,I3,2X,"PLOT",2X,5A2)
      CALL WJST (0,N)
      CALL WSCTR(NA,0,0)
      CALL ACCES(0,NPLOT)
      CALL EXEC(10,NPLOT)
C
C.....SET PRIORITY OF JOB TO 0
C
3100   CALL WJST (NA,K)
      CALL LIGHT (-1,0)
      CALL WJST (K,L)
      CALL JSTF(0,NOPR,L)
      CALL WJST (-K,L)
      GOTO 350
C
C.....GOTO THE DISPLAY AND MARK OR CHECK
C
330   CALL DSPLA(0)
      CALL LIGHT (-1,0)
      CALL DSPLA(6,1,N)
      IF (N.EQ.0) GOTO 306
      IF (N.EQ.2) GOTO 332
C.....CALL MARK PROGRAM
      NJT=5
      CALL NXTJB(NJT,NA)
```

```
      IF (NA.EQ.0) GOTO 338
      DO 331 I=1,5
      CALL MOST(NA+I,LOT(I))
331   CONTINUE
      CALL WRITE (LOT)
9004  FORMAT (1X,/,I3,2X,"MARK",2X,5A2)
      CALL MOST (0,N)
      CALL MSCTR(NA,MOPNO,0)
      CALL ACCES(0,MMARK)
      CALL LIGHT (NBUSY,1)
      IF(NJT.EQ.9) CALL EXEC(10,MMARK)
      CALL EXEC (10,MMARK)
C
C.....CALL CHECK PROGRAM
C
332   CALL MOST(-4,0)
      CALL WRITE (NBK)
9005  FORMAT (1X,/,I3,2X,"CHECK",2X,5A2)
      CALL MOST (0,N)
      CALL MSCTR(2,3,0)
      CALL ACCES(0,NCHEK)
      CALL LIGHT (NBUSY,1)
      CALL EXEC (10,NCHEK)
C
C.....THE MARK OR CHECK QUEUES ARE EMPTY
C
338   CALL DSPLA(4,5)
      GOTO 200
C
C
C.....CALL THE XFER PROGRAM
C
350   K=3
      CALL NXTJB(K,NA)
      IF (NA.EQ.0) GOTO 200
      CALL MOST (NA+1,NA)
      CALL WRITE (NBK)
9009  FORMAT (1X,/,I3,2X,"XFER",2X,5A2)
      CALL DSPLA(4)
      CALL MOST(4,N)
      CALL MSCTR(NA,0,0)
      CALL ACCES(0,NXFR)
      CALL LIGHT (NBUSY,1)
      CALL EXEC (10,NXFR)
C
C
C.....DISPLAY SHUTTING DOWN SYSTEM
C
400   CALL DSPLA(0)
      CALL DSPLA(2)
      CALL MOST (0,N)
C.....CHECK FOR PLOTTER FINISHED
      CALL PLTST(1,1)
  444 CALL PLTST(-2,IVAL)
      IF(IVAL.NE.1)GO TO 444
      CALL DSPLA(1,5)
      CALL ACCES(0,0)
      CALL LIGHT (-1,0)
      IF (.NOT. NHALT) GOTO 200
C
      IRAY(0)=1
      IRAY(1)=1220
      IRAY(0)=1
      CALL EXEC(-19)
      IRAY(1)=1093
```

```
      IRAY(0)=1027
      CALL EXEC(-19)
      IRAY(1)=1106
      IRAY(0)=11429
      CALL EXEC(-19)
      IRAY(1)=1890
      IRAY(0)=12216
      CALL EXEC(-19)
      IRAY(1)=1892
      IRAY(0)=12216
      CALL EXEC(-19)
      IRAY(1)=1220
      IRAY(0)=0
      CALL EXEC(-19)
C
      STOP
      END
      SUBROUTINE WDMP(LOT)
      DIMENSION LOT(1)
 8001 FORMAT (1X,/,          7X,"1",7X,"2",7X,"3",7X,"4",7X,"5",7X,"6")
 8002 FORMAT (1X,6(3X,I5),/)
 8003 FORMAT (14X,  "READY",6X,"TYPE",2X,"PROIORITY",5X,"PURGE",6X,"DATA
     +       ")
 8004 FORMAT (1X,D6,2X,5(4X,I6))
 8005 FORMAT (10X)
 8006 FORMAT (I3,10(1X,03,1X,03))
      NQAV=JFLD(15,1)
      NQJB=JFLD(14,4)
      NQPR=JFLD(10,3)
      NQPU=JFLD(15,1)
      NQAD=JFLD(14,15)
      CALL WQST (2,NQ)
      CALL WQST(3,N0)
      WRITE (6, 8001)
      DO  8010 I=1,6
      CALL WQST (I,LOT(I))
 8010 CONTINUE
      WRITE (6, 8002) (LOT(I),I=1,6)
      WRITE (6, 8003)
      DO  8011 I=7,N0,2
      CALL WQST (I,N1)
      CALL WQST (I+1,N2)
      M1=JLDF(NQAV,N1)
      M2=JLDF(NQJB,N1)
      M3=JLDF(NQPR,N1)
      M4=JLDF(NQPU,N2)
      M5=JLDF(NQAD,N2)
      WRITE (6, 8004) I,M1,M2,M3,M4,M5
 8011 CONTINUE
      WRITE (6,8005)
      DO  8012 I=257,ND,10
      DO  8013 J=0,9
      CALL WQST (I+J,LOT(J+1))
 8013 CONTINUE
      I1=JFLD(15,6)
      I2=JFLD(7,8)
      DO 8014 J=0,9
      LOT(J*2+20)=JLDF(I1,LOT(J+1))
      LOT(J*2+21)=JLDF(I2,LOT(J+1))
8014  CONTINUE
      WRITE (6,8006) I,(LOT(K),K=20,39)
 8012 CONTINUE
      RETURN
      END
      SUBROUTINE WQS(NJOB,NPR,NWA,NQ)
```

```
C.....
C.....STORE Q ENTRY IN WORK QUEUE
C.....
      NJ=JFLD(14,4)
      NP=JFLD(14,3)
      CALL JSTF(NJOB,NJ,N)
      CALL JSTF(NPR,NP,N)
      CALL WQST(-NQ,N)
      CALL WQST(-(NQ+1),NWA)
      NQ=NQ+2
      RETURN
      END
      SUBROUTINE WQST(LOC,NWD)
C.....WQST STORES NWD INTO THE JOB QUEUE AT
C.....LOCATION LOC.  THE QUEUE IS ON THE DISC FILE
C.....AND IS AUTOMATICALLY READ AND WRITTEN WHEN
C.....REQUIRED..
C.....   LOC=0,WRITE SECTORS TO DISC
C.....      =1-32766, STORE NWD AT LOC
C.....   NWD=WORD TO BE STORED
C.....
      DIMENSION NA(384),IO(3),NDISK(3)
C.....
      DATA NBASE,NR/0,0/,IO/2HJO,2HBQ,2H  /,NDISK/2HPN,2H00,2H00/
C.....
      CALL DOSC(0)
      IF (LOC.EQ.0) GOTO 200
      L=IABS(LOC)
      IF (L.LT.257) GOTO 300
      IF (L.GT.32766) GOTO 500
      NRR=(L+127)/128
      NWR=L-(NRR-1)*128
      IF (NRR.NE.NR) GOTO 400
101   IF (LOC.LT.0) GOTO 102
      NWD=NA(NWR+256)
      GOTO 500
102   NA(NWR+256)=NWD
500   RETURN
200   IF (NBASE.EQ.0) GOTO 201
      CALL EXEC (15,56,NA,256,IO,0)
201   IF (NR.EQ.0) GOTO 500
      N=NR-1
      CALL EXEC (15,56,NA(257),128,IO,N)
      GOTO 500
300   IF (NBASE.NE.0) GOTO 301
      CALL EXEC (14,64,NA,256,IO,0)
      NBASE=1
301   IF (LOC.GT.0) GOTO 302
      NA(L)=NWD
      GOTO 500
302   NWD=NA(L)
      GOTO 500
400   IF (NR.EQ.0) GOTO 401
      N=NR-1
      CALL EXEC (15,56,NA(257),128,IO,N)
401   N=NRR-1
      CALL EXEC (14,56,NA(257),128,IO,N)
      NR=NRR
      GOTO 101
      END
      SUBROUTINE WSPO(NA,ND)
C.....
C.....STORE THE IDENTIFICATION INTO THE WRK QUEUE DATA AREA
C.....
      DIMENSION NA(1)
```

```
C.....
      DATA IBK/2H  /
C.....
      DO 100 I=0,4
      CALL WRST(-(ND+I),IBK)
100   CONTINUE
      CALL WPK(NA(5),NA(1),NA(5),1)
      N=(NA(1)+1)/2 -1
      DO 101 I=0,N
      CALL WRST (-(ND+I),NA(I+5))
101   CONTINUE
      ND=ND+5
      RETURN
      END
      SUBROUTINE WGC (NCHAR,NCOL,NPT)
C     WGC SETS NCHAR TO THE NEXT CHARACTER (0-0XXXXXXXXX) FROM THE LINE
C     ARRAY.  THE ROUTINE AUTOMATICALLY READS THE NEXT LINE WHEN THE
C     CURRENT LINE IS DEPLETED.
C     NCHAR = ASCII CHARACTER LEFT JUSTIFIED
C     NCOL = CURRENT COLUMN NUMBER
C     NPT = 0, RETURN A CHARACTER
C         = 1, PRINT LINE AND RETURN CHARACTER
C
      DIMENSION LINE (40)
      DIMENSION K(2)
C
      DATA NC/81/, IBK/2H  /, IDOL/2H$ /
      DATA ISQ/0/
C
101   IF (NC.LT.81) GOTO 100
      READ (5,10) LINE
      LNB=1
      DO 120 I=1,40
      IF (LINE(I).NE.IBK) LNB=I
120   CONTINUE
      IF (NPT.NE.1) GOTO 102
      DO 103 I=1,40
      K(1)=IBK
      K(2)=IBK
      CALL WUP (LINE(I),1,K,1)
      IF (K(1).EQ.IDOL.OR.K(2).EQ.IDOL) GOTO 104
103   CONTINUE
      WRITE (6,12) IBK,IBK,(LINE(I),I=1,LNB)
      GOTO 102
104   ISQ=ISQ+1
      WRITE (6,11) ISQ,(LINE(I),I=1,LNB)
102   NC=1
      NOE=1
100   N=(NC+1)/2
      K(1)=IBK
      K(2)=IBK
      CALL WUP (LINE(N),1,K,1)
      NCHAR=K(NOE)
      NOE=3-NOE
      NCOL=NC
      NC=NC+1
      IF (NCHAR.EQ.IBK) GOTO 101
      RETURN
10    FORMAT (40A2)
11    FORMAT (1X,I4,2X,40A2)
12    FORMAT (1X,2A2,2X,40A2)
      END

      SUBROUTINE WGF (NA)
```

```
C       THIS ROUTINE SCANS A CHARACTER INPUT AND PICKS OFF ONE FIELD OF
C       DATA.  THE DATA FIELD MAY BE ALPHANUMERIC, NUMERIC, ALPHA,OR
C       EMPTY.  THE RESULTS ARE RETURNED IN A 14 WORD ARRAY.
C
C       WORD 1 = NUMBER OF CHARACTERS IN THE FIELD OR THE ERROR COLUMN
C       WORD 2 = STATUS WORD = 0, INTEGER VALUE
C                              =1, ALPHANUMERIC
C                              =2, LENGTH ERROR
C                              =3, EMPTY
C                              =4, DECIMAL VALUE
C       WORD 3 = DELIMETER RETURNED $ ; , = :
C       WORD 4 = VALUE OF INTEGER AS XXX.XX
C       WORD 5 - 14 = 10 OR LESS CHARACTER LIST
C       WORD 15 = CURRENT COLUMN NUMBER
C
        DIMENSION NA(15)
C
        DATA IDOL,ISEM,ICOL,IEQ/2H$ ,2H; ,2H: ,2H= /
        DATA IPER,ISLSH/2H. ,2H/ /
        DATA I0,I9/2H0 ,2H9 /
        DATA IBK, ICOM /2H  ,2H, /
C
        DO 101 I=1,4
101     NA(I)=0
        DO 104 I=5,14
104     NA(I)=IBK
        NDEC=0
        NA(1)=1
100     CALL NGC (NCHAR,NCOL,1)
        NA(15)=NCOL
        IF (NCHAR.EQ.IDOL .OR. NCHAR.EQ.ISEM .OR. NCHAR.EQ.ICOL
     +   .OR.NCHAR.EQ.IEQ.OR.NCHAR.EQ.ICOM.OR.NCHAR.EQ.ISLSH) GOTO 110
        IF (NCHAR.NE.IPER) GOTO 102
        NDEC=1
        GOTO 130
102     IF (NCHAR.GT.I9 .OR. NCHAR.LT.I0) GOTO 120
        NA(4)=NA(4)*10+((NCHAR-I0)/256)
        GOTO 130
120     NA(2)=1
130     I=NA(1)
        IF (I.GT.10) GOTO 140
        NA(I+4)=NCHAR
131     NA(1)=I+1
        GOTO 100
140     NA(2)=2
        GOTO 131
110     NA(3)=NCHAR
        IF (NA(1).EQ.1) NA(2)=3
        IF (NA(2).EQ.2 .OR. NA(2).EQ.3) NA(1)=NCOL
        IF (NA(2).EQ.0 .AND. NDEC.EQ.1) GOTO 150
        IF (NA(2).EQ.0) NA(4)=NA(4)*100
160     NA(1)=NA(1)-1
        GOTO 190
C       BIAS THE DECIMAL NUMBER TO XXXX.XX
150     NA(2)=4
        DO 151 I=5,14
        IF (NA(I).EQ.IPER) GOTO 153
151     CONTINUE
153     N=I+1
        DO 154 I=N,14
        IF (NA(I).EQ.IBK) GOTO 155
154     CONTINUE
155     M=2-(I-N)
        IF (M.LT.0) GOTO 170
        NA(4)=NA(4)*10**M
```

```
      GOTO 190
170   NA(4)=NA(4)/(10**M)
190   RETURN
      END
      SUBROUTINE WPK (LCHAR,NCHAR,INTO,LR)
C     PACK NCHAR FROM THE ARRAY 'LCHAR' INTO THE ARRAY 'INTO', LCHAR
C     IS LEFT JUSTIFIED IF LR=1,, RIGHT JUSTIFIED IF LR=2.
C
      DIMENSION LCHAR(1), INTO(1), LF(2)
C
      DATA IBK /2H  /
C
      LF(1)=JFLD(15,8)
      LF(2)=JFLD(7,8)
      N=(NCHAR+1)/2
      N=1
      DO 190 I=1,NCHAR
      K=(I+1)/2
      M=JLOF(LF(LR),LCHAR(I))
      CALL JSTF (M,LF(N),INTO(K))
      N=3-N
190   CONTINUE
      RETURN
      END
      SUBROUTINE WUP (LWD,NWD,INTO,LR)
C     UNPACK NWD WORDS FROM ARRAY LWD AND STORE EACH CHARACTER IN
C     ARRAY INTO EITHER LEFT (1) OR RIGHT (2) JUSTIFIED.
C
      DIMENSION LWD(1), INTO(1), LF(2)
C
      LF(1)=JFLD(15,8)
      LF(2)=JFLD(7,8)
      DO 190 K=1,NWD
      K2=K-1
      DO 120 K1=1,2
      M=JLOF(LF(K1),LWD(K))
      CALL JSTF (M,LF(LR),INTO(K2+K1))
120   CONTINUE
190   CONTINUE
      RETURN
      END
      SUBROUTINE WQLST(LOT)
      DIMENSION LOT(1)
      CALL WDMP(LOT)
      RETURN
      END
      SUBROUTINE NXTJB(NT,IA)
C
C.....GET THE HIGHEST PRIORITY JOB IN THE QUEUE
C
C.....NT(ENTRY)=5 OR 9, GET NEXT MARK JOB QUEUE ENTRY
C.....          =1, GET THE NEXT (DRAW,PLOT) QUEUE ENTRY
C.....          =2, GET THE NEXT (STYLE) QUEUE ENTRY
C.....          =3, GET THE NEXT (XFER) QUEUE ENTRY
C.....NT(EXIT)  =4, STYLE
C.....          =5 OR 9, MARK
C.....          =6, DRAW
C.....          =7, PLOT
C.....          =8, XFER
C.....IA(EXIT)  =0, NO ENTRY FOUND
C.....          = ADDRESS OF JOB QUEUE ENTRY
C
      NOJB=JFLD(14,4)
      NXFR=JFLD(10,3)
      JNAD=0
```

```
      JOPR=M
      NJ=0
      CALL MGST (2,MJOE)
      IJJE=MJOE-2
C
C.....START THE SEARCH FOR AN ENTRY
C
100   MJJ=MJJ+2
      IF (MJJ.GT.MJOE) GOTO 150
C.....GET THE JOB QUEUE ENTRY
      CALL MGST (MJJ,MQW1)
      CALL MGST(MJJ+1,MQW2)
      IF (MQW2.LT.0) GOTO 100
      NJ=JLOF(NQJB,MQW1)
C.....CHECK FOR MARK REQUEST
      IF (NT.EQ.NJ) GOTO 110
      IF (NT.EQ.1 .AND. (NJ.EQ.6.OR.NJ.EQ.7)) GOTO 110
      IF (NT.EQ.2 .AND. NJ.EQ.4) GOTO 110
      IF (NT.EQ.3 .AND. NJ.EQ.8) GOTO 110
      GOTO 100
C
C.....SELECT THE HIGHEST PRIORITY ENTRY
C
110   N=JLOF(NJPR,MQW1)
      IF (N.EQ.0) GOTO 100
      IF (N.EQ.7) GOTO 120
      N=N+1
      CALL JSTF(N,NJPR,MQW1)
      CALL MGST(-MJJ,MQW1)
120   IF (N.LE.JJPR) GOTO 100
      JJAD=MJJ
      JJPR=N
      GOTO 100
C
C.....GET THE EXIT WORDS
C
150   IA=JJAD
      IF (JJAD.EQ.0) GOTO 99
      CALL MGST(JJAD,N)
      NT=JLOF(NQJB,N)
      CALL MGST(JJAD+1,IA)
      CALL MGST (-4,JJAD)
99    RETURN
      END
      LOGICAL FUNCTION M04(N)
      M04=.FALSE.
      IF (N.EQ.0.OR.N.EQ.4) M04=.TRUE.
      RETURN
      END
      LOGICAL FUNCTION MDS(N)
      DATA IDDL,ISEM/2HS ,2HM /
      MDS=.FALSE.
      IF (N.EQ.IDDL.OR.N.EQ.ISEM) MDS=.TRUE.
      RETURN
      END
      LOGICAL FUNCTION MDSC(N)
      DATA IDDL,ISEM,ICOM/2HS ,2HM ,2HC, /
      MDSC=.FALSE.
      IF (N.EQ.IDDL.OR.N.EQ.ISEM.OR.N.EQ.ICOM) MDSC=.TRUE.
      RETURN
      END
      LOGICAL FUNCTION M23(N)
      M23=.FALSE.
      IF (N.EQ.2.OR.N.EQ.3) M23=.TRUE.
      RETURN
```

```
      END
      LOGICAL FUNCTION MCD(N)
      DATA ICOM,IOOL/2H, ,2H$ /
      MCD=.FALSE.
      IF (N.EQ.ICOM.OR.N.EQ.IOOL) MCD=.TRUE.
      RETURN
      END
      SUBROUTINE WSCTR(N1,N2,N3)
      DIMENSION LOT(3)
      DIMENSION NCS0(3)
      DATA NCS0/2HSF,2HCT,2HR /
      CALL UOSC(0)
      LOT(1)=N1
      LOT(2)=N2
      LOT(3)=N3
      CALL EXEC (15,06,LOT,3,NCS0,0)
      RETURN
      END
      SUBROUTINE SFLG (NPG0,NPGIN,NER)
      NNPG=JFLD(15,1)
      NNIN=JFLD(14,1)
      NNER=JFLD(13,1)
      CALL WOST(1,N)
      CALL JSTF(NPG0,NNPG,N)
      CALL JSTF(NPGIN,NNIN,N)
      CALL JSTF(NER,NNER,N)
      CALL WOST(-1,N)
      RETURN
      END
      ENDS
```

```
FTN4
      SUBROUTINE WOST(LOC,NWD)
C.....WOST STORES NWD INTO THE JOB QUEUE AT
C.....LOCATION LOC.  THE QUEUE IS ON THE DISC FILE
C.....AND IS AUTOMATICALLY READ AND WRITTEN WHEN
C.....REQUIRED..
C.....    LOC=0,WRITE SECTORS TO DISC
C.....       =1-32765, STORE NWD AT LOC
C.....    NWD=WORD TO BE STORED
C.....
      DIMENSION NA(384),IO(3),NDISK(3)
C.....
      DATA NBASE,NR/0,0/,IO/2HJO,2HBQ,2H  /,NDISK/2HPN,2H00,2H00/
C.....
      IF (LOC.EQ.0) GOTO 200
      L=IABS(LOC)
      IF (L.LT.257) GOTO 300
      IF (L.GT.32700) GOTO 500
      NRN=(L+127)/128
      NRR=L-(NRR-1)*128
      IF (NRR.NE.NR) GOTO 400
101   IF (LOC.LT.0) GOTO 102
      NWD=NA(NRR+256)
      GOTO 500
102   NA(NRR+256)=NWD
500   RETURN
200   CALL EXEC(23,NDISK)
      IF (NBASE.EQ.0) GOTO 201
      CALL EXEC (15,06,NA,256,IO,0)
201   IF (NR.EQ.0) GOTO 500
      N=NR-1
      CALL EXEC (15,06,NA(257),128,IO,N)
      GOTO 500
```

```
300     IF (NBASE.NE.0) GOTO 301
        CALL EXEC(23,NDTSK)
        CALL EXEC (14,66,NA,256,IQ,0)
        NBASE=1
301     IF (LOC.GT.0) GOTO 302
        NA(L)=NWD
        GOTO 500
302     NWD=NA(L)
        GOTO 500
400     CALL EXEC(23,NDTSK)
        IF (NR.EQ.0) GOTO 401
        N=NR-1
        CALL EXEC (15,66,NA(257),128,IQ,N)
401     N=NRR-1
        CALL EXEC (14,66,NA(257),128,IQ,N)
        NR=NRR
        GOTO 101
        END
        ENDS


ASMB,L,F
        NAM WRITE,7
        ENT WRITE
*
*       THIS ROUTINE WILL OUTPUT AN EVENT COUNTER AND
*       FORMAT STATEMENT WHICH IS LOCATED JUST AFTER
*       THE CALL STMT IN FORTRAN.  ROUTINE COUNTS ON
*       THE HP FTN4 COMPILER TO GENERATE CODE LIKE
*            JSB WRITE
*            DEF *+2
*            DEF ITEXT
*            JMP XXX            JUMP TO END OF FRMT
*            ASC 1,(I           ASCII FORMAT STMT
*
        EXT .DIO.,.IOI.,.IAY.,.DTA.,.ENTR,WQST
ARRAY   BSS 1           ADDRESS OF 5 WORD ARRAY
WRITE   NOP
        JSB .ENTR       GO TO GET
        DEF ARRAY       ADDRESS OF ARRAY
        LDA WRITE       GET ADDRESS OF JSB+2
        ADA .+1         FORM ADDRESS OF FORMAT STMT
        STA FRMT        FIX FOR CALL TO FRMTR
        LDA .+0         LOAD WITH UNIT 6 NUMBER
        CLB
        JSB .DIO.       FOR DETAILS SEE 3-26 OF REL SUB
FRMT    DEF *-*         ADDRESS OF FORMAT
        DEF BOMB        WHERE TO GO IF ERROR
        JSB WQST        PICK UP COUNTER CURRENT VALUE
        DEF *+3
        DEF .+0         SIXTH WORD READ
        DEF CNTR
        LDA CNTR        PICK UP EVENT COUNTER
        ADA .+1         INCREMENT
        STA CNTR        SAVE FOR LATER
        JSB .IOI.       PASS TO FRMTR
        JSB WQST        WRITE IT BACK OUT
        DEF *+3
        DEF .-0         SIXTH WORD WRITE
        DEF CNTR
        JSB .IAY.       ADD TO DATA ROSTER
        DEF ARRAY,I
        DEF .+5,I       LENGTH IS 5 WORDS
        JSB .DTA.       CRUNCH HIM
BOMB    JMP WRITE,I     RETURN FROM WHENCE THY CAME
```

```
*
CNTR    BSS 1           TEMPORARY MEM CELL
*       EXT JSB         POINTER TO BASE PAGE CONSTANTS
        END
ENDS


FTN4
      PROGRAM WSTYL,3
C
C     VERSION 2,1
C     EDITED 29 FEB 73
C     RONALD GORDON
C
      INTEGER OUT
      INTEGER OPNTR
      LOGICAL DEBUG
      LOGICAL ERRORS
      INTEGER OPTNS(3)
      INTEGER WPROD(3)
      INTEGER SECTR(3), STYLE(3)
      INTEGER PNAME(5), PNTNO
      DIMENSION IBUFR(128), NAME(5)
C
C     COMMUNICATION WITH GETIN
C
      COMMON OPNTR,DEBUG
C
C     JUST TO SAVE DISK SPACE
C
      COMMON IBUFR,PNAME,NAME
      DATA SECTR /2HSE,2HCT,2HR /
      DATA STYLE /2HST,2HYL,2HE /
      DATA WPROD /2HWP,2HRO,2HD /
      DATA OPTNS /2HOP,2HTN,1HS/
      DATA NULL /43777B/
      DATA OUT /6/
C
C     PICK UP PARAMETERS
C
      CALL RMPAR (OPNTR)
      CALL UDSC (3)
      CALL EXEC (14,1428,IBUFR,4,OPTNS,0)
      DEBUG=JLUF(JFLD(1,1),IBUFR(4)) .EQ. 1
      DEBUG=DEBUG .OR. ISSW(13).LT.0
      IF (DEBUG) WRITE (1,1001) OPNTR
      MASAL=JFLD(15,0)
      MASKR=JFLD(7,8)
      D4C1=JFLD(0,1)
      M1C1=JFLD(1,1)
      D2C1=JFLD(2,1)
      M3C1=JFLD(3,1)
      D6C3=JFLD(6,3)
      D9C7=JFLD(9,7)
      M9C14=JFLD(9,14)
      A1MC1=JFLD(10,1)
      M11C1=JFLD(11,1)
      A12C1=JFLD(12,1)
      M12C13=JFLD(12,13)
      M13C1=JFLD(13,1)
      M14C1=JFLD(14,1)
      M14C4=JFLD(14,4)
      M14C5=JFLD(14,5)
      M15C1=JFLD(15,1)
      M15C3=JFLD(15,3)
```

```
C
C       LOOP THROUGH ALL STYLE ENTRIES
C
        CALL UDSC (0)
        CALL MGST (2,LN)
        L0=LN-2
        IF (LN.LT.7) GO TO 900
        DO 500 I0=7,L0,2
        CALL MGST (I0,ITEMP)
        ITYPE=JLOF(M14C4,ITEMP)
        IF (ITYPE.NE.4) GO TO 500
        CALL MGST (-I0,0)
        CALL MGST (I0+1,IGWORD)
        CALL MGST (-(I0+1),IOR(100000B,IGWORD))
        ERRORS = .FALSE.
C
C       PROCESS THIS STYLE REQUEST
C
        DO 130 I=1,5,1
        IGWORD=IGWORD+1
  130   CALL MGST (IGWORD,NAME(I))
        WRITE (OUT,1700) NAME
        IGWORD=IGWORD+1
        CALL MGST (IGWORD,ITEMP)
        NPIEC=JLOF(MASKR,ITEMP)
        IRULE=JLOF(MASKL,ITEMP)
C
C       ATTEMPT TO ALLOCATE SPACE FOR THIS STYLE
C
        CALL DIREC (14B3B,NAME,INTNO,ISCTR,ISTAT)
        ISECT=ISCTR
  135   IF (ISTAT.EQ.0) GO TO 150
        ERRORS=.TRUE.
        IF (ISTAT.EQ.1) GO TO 145
        IF (ISTAT.NE.4) GO TO 900
        WRITE (OUT,1600)
        GO TO 155
  145   WRITE (OUT,1200) NAME
C
C       SET UP TO PROCESS PIECE DEFINITIONS
C
  155   CONTINUE
        IF (ERRORS) GO TO 160
        CALL EXEC (10,STYLE,LIMIT)
        NSECT=(NPIEC-1)/25+1
        IF (NSECT+ISCTR.LT.LIMIT) GO TO 160
        WRITE (OUT,1201) NAME
        CALL EXEC (15,1028,1,1,STYLE,ISECT)
        ERRORS=.TRUE.
  160   IBUFL=2
        IBUFR(1)=NSECT
        CALL JSIF (NPIEC,MASKL,IBUFR)
        IBUFR(2)=IRULE
        DO 300 I=1,NPIEC,1
C                                       GET PIECE NAME AND NUMBER
        DO 170 J=1,5,1
        IGWORD=IGWORD+1
  170   CALL MGST (IGWORD,PNAME(J))
        CALL GETIN (PNAME,PNTNO)
        IF (PNTNO.NE.NULL) GO TO 180
        WRITE (OUT,1300) PNAME
        IF (ERRORS) GO TO 300
        CALL DIREC (24B3B,NAME,INTNO,ISCTR,ISTAT)
        CALL EXEC (15,1028,1,1,STYLE,ISCTR)
        ERRORS=.TRUE.
```

```
 180  IF (ERRORS) GO TO 340
C                                       INTNO
      IBUFR(IWORD+1)=PRTNO
      IWORD=IWORD+1
      CALL HOST (IWORD,ITEMP)
C                                       PIECE MESAGES
      IBUFR(IWORD+2)=JLOF(M9C1,ITEMP)
      IBUFR(IWORD+3)=0
C                                       MAP GROUP
      CALL JSTF (JLOF(M1OC1,ITEMP),M9C1,IBUFR(IWORD+3))
C                                       FLIP GROUP
      CALL JSTF (JLOF(M13C1,ITEMP),M1C1,IBUFR(IWORD+3))
C                                       ROTATE FLAG
      CALL JSTF (JLOF(M11C1,ITEMP),M2C1,IBUFR(IWORD+3))
C                                       SHADE CONTROL
      CALL JSTF (JLOF(M14C1,ITEMP),M3C1,IBUFR(IWORD+3))
      IBUFR(IWORD+4)=0
C                                       BLOCK FLAG
      CALL JSTF (JLOF(M15C1,ITEMP),M0C1,IBUFR(IWORD+4))
C                                       OPPOSIT FLAG
      CALL JSTF (JLOF(M12C1,ITEMP),M15C1,IBUFR(IWORD+4))
      IWORD=IWORD+1
      CALL HOST (IWORD,ITEMP)
C                                       X %
      CALL JSTF (ITEMP/10,MASKL,IBUFR(IWORD+5))
      IWORD=IWORD+1
      CALL HOST (IWORD,ITEMP)
C                                       Y %
      CALL JSTF (ITEMP/10,MASKR,IBUFR(IWORD+5))
      IWORD=IWORD+1
      CALL HOST (IWORD,ITEMP)
C                                       STRIPE FIELD
      CALL JSTF (JLOF(M15C3,ITEMP),M6C3,IBUFR(IWORD+3))
C                                       HALF PIECE TABLE
      CALL JSTF (JLOF(M12C13,ITEMP),M9C7,IBUFR(IWORD+4))
      IWORD=IWORD+1
      CALL HOST (IWORD,ITEMP)
C                                       MAX TILT VALUE
      CALL JSTF (ITEMP/100,M14C5,IBUFR(IWORD+4))
      IWORD=IWORD+5
      IF (IWORD.LT.124) GO TO 300
      IWORD=2
      CALL EXEC (15,1028,IBUFR,128,STYLE,ISCTR)
      ISCTR=ISCTR+1
      IBUFR(1)=0
      IBUFR(2)=0
  300 IF (ERRORS) IWORD=IWORD+5
      IF (IWORD.GT.2) CALL EXEC (15,1028,IBUFR,IWORD,STYLE,ISCTR)
      IF (.NOT.ERRORS) WRITE (OUT,1500)
  310 CONTINUE
  900 CONTINUE
      CALL HOST (0,0)
      CALL DOSC (0)
      CALL EXEC (10,WPROG)
 1000 FORMAT("STYL LOADED.  CONSTANT IS: ",06)
 1200 FORMAT("NO ROOM FOR STYLE ",5A2)
 1300 FORMAT("CAN'T FIND PIECE ",5A2)
 1500 FORMAT("COMPLETED",/)
 1700 FORMAT(/,"STYLE ",5A2)
 1800 FORMAT("STYLE BY THIS NAME ALREADY EXISTS")
      END
      SUBROUTINE GETIN (INAME,INTNO)
      DIMENSION INAME(5)
      LOGICAL INCORE
      LOGICAL DEBUG
```

```
      INTEGER CHEK(3)
      INTEGER OPNTR
      DIMENSION MONSTR(1)
      COMMON OPNTR,DEBUG
      DATA NULL /O3777777/
      DATA CHEK /2HCM,2HEK,2H  /
      DATA INCORE /.FALSE./
      IF (INCORE) GO TO 100
      CALL ALLOC (MONSTR(0),LIMIT)
      IF (DEBUG) WRITE (1,1002) LIMIT
      LIMIT=OPNTR
      CALL DUSL (0)
      CALL LAPG (14,1H2B,MONSTR(1),LIMIT,CHEK,0)
      IF(CHEK.EQ.IBUF.
  100 CONTINUE
      DO 200 I=1,LIMIT,8
      IF (INAME(1).NE.MONSTR(I)) GO TO 200
      IF (INAME(2).NE.MONSTR(I+1)) GO TO 200
      IF (INAME(3).NE.MONSTR(I+2)) GO TO 200
      IF (INAME(4).NE.MONSTR(I+3)) GO TO 200
      IF (INAME(5).NE.MONSTR(I+4)) GO TO 200
      INTNO=-(MONSTR(I+5)+1)
      RETURN
  200 CONTINUE
      INTNO=NULL
      RETURN
 1002 FORMAT("BUFFER SPACE=",O6)
      END
      END$


FTN4,P
      SUBROUTINE XFORM(X2,Y2,X0,Y0,SINE,COSIN)
      X1=X2
      X2=COSIN*X1+SINE*Y0
      Y2=COSIN*Y0-SINE*X1
      RET
      END$


FTN4
      SUBROUTINE XREF1 (IQUEU,LIM4,IPNTR)
      LOGICAL NPOST
C**
C**   WARNING:  IQUEU MUST BE DIMENSIONED IQUEU(LIM4,3) BUT
C**   ERROR IN VERSION 24177-5 KILLS US.  KLUGE FOLLOWS
C**
      DIMENSION IQUEU(120,3)
      DIMENSION IPNTR(6)
      DIMENSION INAME(5), IFILE(3)
      DIMENSION ISRCH(3)
      COMMON IBUFR(128)
      DATA ISRCH /0410H2B, 0010H2B, 0010038/
      DATA IFILE /2HST,2HYL,2HB /
      DATA NEMPS /25/, NAPE /5/, ICTRL /001028/,
     +     ILOOK /0H1403B/, IREAD /14/, IPURG /002H03B/, ISIZE /128/
      DATA IUN /6/
      WRITE (1U6,1000)
C
C     OBTAIN INTNO FOR NAMES IN XREF LISTS
C
      DO 50 I=1,3,1
      L=IPNTR(I)
      IF (L.EQ.0) GO TO 50
      DO 25 J=1,L,6
```

```
      J=J+4
      CALL DIREC (ISRCH(I),IQUEU(J,I),IQUEU(J+5,I),NSCTR,ISTAT)
      IF (ISTAT.EQ.0) GO TO 10
      IQUEU(J+5,I)=-1
      WRITE (IO6,1150) (IQUEU(K,I), K=J,M,1)
      GO TO 20
   10 WRITE (IO6,1200) (IQUEU(K,I), K=J,M,1)
   20 CONTINUE
      WRITE (IO6,1300)
   30 CONTINUE
C
C     GO PIECES IN A STYLE REFRENCE
C
  200 L=IPNTR(1)
      IF (L.EQ.0) RETURN
      INTNO=-1
      WRITE (IO6,1500)
      MASK1=JFLD(3,4)
      MASK2=JFLD(15,5)
  220 INTNO=INTNO+1
      NPOST=.TRUE.
      CALL DIREC (ILOOK,INAME,INTNO,NSCTR,ISTAT)
      IF (ISTAT.EQ.5) RETURN
      IF (ISTAT.NE.0) GO TO 220
      ISCTR=NSCTR
  225 CALL EXEC (IREAD,ICTRL,IBUFR,ISIZE,IFILE,ISCTR)
      IF (ISCTR.NE.NSCTR) GO TO 230
      LSCTR=ISCTR+JLOF(MASK1,IBUFR)
      LP=JLOF(MASK2,IBUFR)
  230 ISCTR=ISCTR+1
      IP=IP+MENPS
      IF (IP.GT.LP) IP=LP
      N=MOD(IP-1,MENPS)*NWORE+1
      DO 250 J=1,N,NWORE
      DO 240 I=1,L,5
      IF (IQUEU(I+5,1).NE.IBUFR(J+2)) GO TO 240
      IF (NPOST) WRITE (IO6,1600) INAME
      NPOST=.FALSE.
      M=I+4
      WRITE (IO6,1200) (IQUEU(K), K=I,M,1)
  240 CONTINUE
  250 CONTINUE
      IF (ISCTR.LT.LSCTR) GO TO 225
      GO TO 220
 1000 FORMAT(/,9HXREF UTIL,/,14HNAMES IN XREF:)
 1100 FORMAT(5A2,1H2)
 1200 FORMAT(5A2)
 1300 FORMAT(/)
 1500 FORMAT(/,23HPIECES IN STYLE XREF)
 1600 FORMAT(/,1H*,5A2)
      END
      SUBROUTINE XREF2 (IQUEU,LIM4,IPNTR)
C
C     IQUEU = LIST OF NAMES TO XREF AND INTNO'S FOR PIECES AND STYLES
C     LIM4 = SIZE OF XREF TABLES
C     IPNTR = POINTERS FOR XREF ROUTINE
C     IO6 = FORTRAN LOGICAL UNIT OF LIST DEVICE
C
      LOGICAL NPOST
C**
C**   WARNING:  IQUEU MUST BE DIMENSIONED IQUEU(LIM4,3) BUT ERROR IN
C**   VERSION 241770 TRIPS THIS UP.  KLUGE FOLLOWS
C**
      DIMENSION IQUEU(120,3)
      DIMENSION IPNTR(6)
```

```
      DIMENSION INAME(5)
      DIMENSION IFILE(3)
      COMMON IBUFR(128)
      DATA IFILE /2HMA,2HRK,2HR /
      DATA ISIZE /128/, ICTRL /00010208/, ILOOK /00140108/, IREAD /14/,
     +     NEOPS /10/, NWOPE /8/
      DATA IO6 /6/
      WRITE (IO6,1000)
C
C     DO PIECES IN A MARKER REFRENCE
C
      N=IPNTR(2)
      IF (N.EQ.0) GO TO 200
      WRITE (IO6,1100)
      INTNO=-1
       MASK1=JFLD(5,0)
       MASK2=JFLD(15,16)
  120 INTNO=INTNO+1
      CALL DIREC (ILOOK,INAME,INTNO,NSCTR,ISTAT)
      IF (ISTAT.EQ.5) GO TO 200
      IF (ISTAT.NE.0) GO TO 120
      ISCTR=NSCTR
      NPOST=.TRUE.
  130 CALL EXEC (IREAD,ICTRL,IBUFR,ISIZE,IFILE,ISCTR)
      ISCTR=ISCTR+1
      IF (ISCTR.NE.NSCTR+1) GO TO 140
      LSCTR=ISCTR+JLOF(MASK1,IBUFR)
      LP=JLOF(MASK2,IBUFR)
      IP=0
      GO TO 130
  140 IP=IP+NEOPS
      IF (IP.GT.LP) IP=LP
      L=MOD(IP-1,NEOPS)*NWOPE+1
      DO 150 I=1,L,NWOPE
      DO 150 J=1,N,5
      IF (IQUEU(J+5,2).NE.IBUFR(I+2)) GO TO 150
      IF (NPOST) WRITE (IO6,1200) INAME
      NPOST=.FALSE.
      M=J+4
      WRITE (IO6,1300) (IQUEU(K,2), K=J,M,1)
  150 CONTINUE
      IF (ISCTR.LT.LSCTR) GO TO 130
      GO TO 120
C
C     DO STYLES IN A MARKER XREF
C
  200 L=IPNTR(3)
      IF (L.EQ.0) RETURN
      WRITE (IO6,1400)
      INTNO=-1
      MASK1=JFLD(15,11)
      MASK2=JFLD(4,5)
      MASK3=JFLD(15,5)
  210 INTNO=INTNO+1
      CALL DIREC (ILOOK,INAME,INTNO,NSCTR,ISTAT)
      IF (ISTAT.EQ.5) RETURN
      IF (ISTAT.NE.0) GO TO 210
      NPOST=.TRUE.
      CALL EXEC (IREAD,ICTRL,IBUFR,ISIZE,IFILE,NSCTR)
      LSTYL=JLOF(MASK3,IBUFR(2))
      IWORD=14
      DO 220 J=1,LSTYL,1
      INTNO=JLOF(MASK1,IBUFR(IWORD))
      IWORD=IWORD+JLOF(MASK2,IBUFR(IWORD)-1)/2+2
      DO 230 I=1,L,5
```

```
      IF (MNTNU.NE.IQUEU(I+5,3)) GO TO 230
      IF (NPOST) WRITE (IO6,1200) INAME
      NPOST=.FALSE.
      M=I+4
      WRITE (IO6,1300) (IQUEU(K,3), K=I,M,1)
  230 CONTINUE
  220 CONTINUE
      GO TO 210
 1000 FORMAT(/,9HXREF DUMP)
 1100 FORMAT(21HPIECES IN MARKER XREF)
 1200 FORMAT(/,1H*,5A2)
 1300 FORMAT(5A2)
 1400 FORMAT(/,21HSTYLES IN MARKER XREF)
      END
$EOD


FTN4
      PROGRAM ZERO
C.....PROGRAM TO ZERO OUT ANY DISK FILE.
      DIMENSION NAM(3),LAB(3),IBUF(128),LABM(3)
      DATA IBUF/128*0/,MIN/1/,MOUT/1/,LABM/2HPN,2H00,2H01/
      WRITE(MOUT,1000)
 1000 FORMAT ("ENTER FILE NAME",/,"$@")
      READ(MIN,1001) NAM
 1001 FORMAT (3A2)
      WRITE(MOUT,1002)
 1002 FORMAT ("ENTER PLATTER LABEL",/,"$@")
      READ(MIN,1001) LAB
      CALL EXEC (23,LAB)
      CALL EXEC (18,NAM,NSECT)
      IF(NSECT.NE.0) GO TO 10
      WRITE(MOUT,1003) NAM,LAB
 1003 FORMAT ("*** FILE, ",3A2,"NOT ON PLATTER, ",3A2)
      GO TO 200
   10 WRITE (1,1005)
      READ (1,*) ISTRT
      ISTRT=ISTRT+1
 1005 FORMAT("ENTER TRUE START SECTOR",/,"$@")
      DO 100 I=ISTRT,NSECT,1
      ISECT = I-1
      CALL EXEC (15,65,IBUF,128,NAM,ISECT)
  100 CONTINUE
  200 CONTINUE
      STOP
      END
      END$


FTN4
      PROGRAM BBUILD,5
C
C..... DEC 15 1972
C..... BOB PATRICK
C..... RULE PROGRAM
C
       COMMON IN,IO,ISEG(3),IRX(840),IRY(840),IBUF(128),IPFLG
      COMMON ISC(3),IPROG(3)
      DIMENSION NSAV(3),IPLAT(3),NRT(6),IRULE(3),NRT2(6)
      DIMENSION IBUF2(128)
      DATA NRRS/128/
      DATA NRT/0,0,0,0,0,0/
      ICW = 65
      DO 773 K=1,5
773   NRT(K)=2H
```

```
      DO 774 K=1,128
774    IBUF(K)=0
      NSEC = 0
      INCRF=JFLD(15,6)
      NRUL=JFLD(15,9)
      NSC=JFLD(6,7)
      NSZ=JFLD(9,10)
      IRULE(1)=2HRU
      IRULE(2)=2HLE
      IRULE(3)=2HS
      IBUF(1)=0
      IBUF(2)=0
      CALL LOCKO(1)
      CALL UDSC(0)
      IF(IN.NE.5)GO TO 7751
7752  CALL WCRS(ISTAT)
      IF(ISTAT.LT.0)GO TO 7751
      WRITE(1,7753)
7753  FORMAT("***CARD READER NOT READY***")
      GO TO 7752
7751  CONTINUE
      IF(IN.NE.IO)GO TO 2
1     WRITE(IO,100)
100   FORMAT("ENTER RULE SET ",/,"$")
2     READ(IN,102)NRT,INCR,LSIZ,LRUL
102   FORMAT(6A2,I5,2I4)
C.....          SET UP GRADING INCREMENT FOR GRADP
      IF(INCR.EQ.0)INCR=1
      IF(INCR.EQ.32)INCR=1
      IF(INCR.EQ.64)INCR=2
      IF(INCR.EQ.128)INCR=3
      IF(INCR.EQ.256)INCR=4
      IF(INCR.EQ.512)INCR=5
      IF(INCR.EQ.1024)INCR=6
      IF(INCR.EQ.10)INCR=7
      IF(INCR.EQ.104)INCR=8
      IF(INCR.EQ.8192)INCR=9
      IF(INCR.GT.9.OR.INCR.LT.1)GO TO 7777
C.........          SEARCH FOR RULE TABLE BY NAME
      CALL DIREC(1000B,NRT,INTNO,NSCTR,ISTAT)
      IF(ISTAT.EQ.0) GO TO 130
      GO TO (140,150,140,130,140)ISTAT
C.........................RETRIEVE UNIQUE #, PURGE BY INT #
130   CALL DIREC(3400B,NRT,INTNO,NSCTR,ISTAT)
      CALL DIREC(2400B,NRT,INTNO,NSCTR,ISTAT)
150    DO 143 KX=1,NRRS
      NRT2(6)=KX
C.........................SEARCH W  FIRST WORD
      CALL DIREC(3000B,NRT2,INT2,NSCTR2,ISTAT)
      IF(ISTAT.EQ.0) GO TO 143
C.................................IF NOT USED PU IN WRD 6 OF NRT
      NRT(6)=KX
      GO TO 151
143   CONTINUE
      GO TO 169
C.............................MAKE ENTRY BY NAME
151   CALL DIREC(4000B,NRT,INTNO,NSCTR,ISTAT)
      IF(ISTAT.NE.0) GO TO 140
169   CONTINUE
C.......
C.......                              CHECKFOR SPACE IN FILE RULE
      CALL EXEC(18,IRULE,ISECT)
      IF(NSCTR.EQ.ISECT) GO TO 167
C.........          READ SIZE DES CRIPTORS AND WRI
      NSEC=NSEC+NSCTR
```

```
      IFFLG =1
      IF(IN.NE.IO) GO TO 5
      WRITE(IO,103)
103   FORMAT("ENTER SIZE RELATION TABLE ",/,
     1  " 1 ENTRY PER LINE, /E TO TERMINATE")
5      I=2
      IBUF(1)=1
      IBUF(3)=2HTR
      IBUF(4)=2HNS
      IBUF(5)=2HFR
      NOS = 0
6     J=I+3
      IF(IN.NE.IO) GO TO 7
      WRITE(IO,300)
300   FORMAT("$")
7     IF(IN.NE.5)GO TO 507
506   CALL WCRS(ISTAT)
      IF(ISTAT.LT.0)GO TO 507
      WRITE(IO,7753)
      GO TO 506
507   CONTINUE
      READ(IN,200)ISEG
200   FORMAT(3A2)
      IF(ISEG(1).EQ.2H/E) GO TO 9
      IBUF(J)=ISEG(1)
      IBUF(J+1) = ISEG(2)
      IBUF(J+2) = ISEG(3)
      NOS = NOS +1
      I = I+1
      IF(I.LE.42) GO TO 6
      I=1
      CALL EXEC(15,ICW,IBUF,128,IRULE,NSEC)
      NSEC = NSEC+1
      IF(NSEC-NSCTR.GE.ISECT)GO TO 167
      DO 775 K=1,128
775   IBUF(K)=0
      GO TO 6
C.........      NOS2 = NUMBER OS SEZES +1 FOR TRANSFER FUNCTION
9     NOS2=NOS+1
      IF(I.EQ.1) GO TO 95
      CALL EXEC(15,ICW,IBUF,128,IRULE,NSEC)
      NSEC=NSEC+1
      IF(NSEC-NSCTR.GE.ISECT)GO TO 167
       DO 775 K =1,128
775   IBUF(K)=0
95    CONTINUE
      INSEC=NSEC-NSCTR
      IF(IN.NE.IO) GO TO 10
      WRITE(IO,106)NOS2
106   FORMAT("ENTERX,Y DATA ",I3,"INCREMENTS",/,
     1                  "ENTER 999 TO TERMINATE")
      WRITE(IO,900)
900   FORMAT("THE LAST INCREMENT IS THE TRANSFER")
10    INDX = 3
      I=1
C..............@   NOW BUILD TABLE OF X Y INCREMENTS BY RULES
11    ITY=1HX
      DO 12 J=1,NOS2+1
12    IRX(J)=0
      IF(IN.NE.IO) GO TO 14
13    WRITE(IO,107)I,ITY
107   FORMAT(I3,A1,"=")
14    IF(IN.NE.5)GO TO 509
510   CALL WCRS(ISTAT)
      IF(ISTAT.LT.0)GO TO 509
```

```
      WRITE(IO,7753)
      GO TO 510
509   READ(IN,*)(IRX(J+1),J=1,NOS2)
      IF(IRX(2).EQ.999)GO TO 25
C.....  N  MOVE TRANSFER FUNCTION TO FIRST POSITI
      IRX(1)=IRX(NOS2+1)
      ITY = 1HY
      DO 51 J=1,NOS2+1
51    IRY (J)=0
      IF(IN.NE.IO) GO TO 19
18    WRITE(IO,107)I,ITY
19    IF(IN.NE.5)GO TO 511
512   CALL WCRS(ISTAT)
      IF(ISTAT.LT.0)GO TO 511
      WRITE(IO,7753)
      GO TO 512
511   READ(IN,*)(IRY(J+1),J=1,NOS2)
      IF(IRY(2).EQ.999)GO TO 508
C.....        MOVE TRANSFER FUNCTION TO FIRST POS
      IRY(1)=IRY(NOS2+1)
C
C.....MOVE X Y INCREMENTS INTO BUFFER AND WRITE TO D
      DO 24 J = 1,NOS2
      IBUF(INDX) = IRX(J)
      IBUF(INDX+1) = IRY(J)
      INDX=INDX+2
      IF(INDX.LT.128)GO TO 24
      INDX = 3
      CALL EXEC(15,ICW,IBUF,128,IRULE,NSEC)
      NSEC=NSEC+1
      IF(NSEC.GE.ISECT+NSCTR) GO TO 167
24    CONTINUE
      I = I+1
      GO TO 11
25    IF(INDX.EQ.3)GO TO 26
      CALL EXEC(15,ICW,IBUF,INDX,IRULE,NSEC)
      NSEC=NSEC+1
26    NOR=I-1
C.....  CHECK FOR CORRECT NUMBER OF RULES AND SIZES
      IF(LSIZ.EQ.0)LSIZ=NOS
      IF(LRUL.EQ.0)LRUL=NOR
      IF(LSIZ.NE.NOS)GO TO 874
      IF(LRUL.NE.NOR)GO TO 875
C.....    PUT 2 WORD HEADER IN SEC 0
      CALL EXEC(14,ICW,IBUF,128,IRULE,NSCTR)
       CALL JSTF(NOR,NRUL,IBUF(1))
      CALL JSTF(NOS,NSZ,IBUF(2))
C.....  STORE THE GRADING INCREMENT
      CALL JSTF(INCR,INCRF,IBUF(2))
      NSC2 = NSEC-NSCTR
      CALL JSTF(NSC2,NSC,IBUF(1))
      CALL EXEC(15,ICW,IBUF,128,IRULE,NSCTR)
30    ISEG(1) = 2HBE
      ISEG(2)=2HNT
      ISEG(3)=2HR
C.....  PURGE BY INTNO  THEN CREATE NEW ENTRY
C.....THE FOLLOWING CODE TRANSPOSES THE MATRIX AND STORES IT BY SIZE
      CALL DIREC(2406B,NRT,INTNO,NSCTR,ISTAT)
      IF(ISTAT.NE.0) GO TO 140
      CALL DIREC(4006,NRT,ININD,NSCTR2,ISTAT)
      IF(ISTAT.NE.0) GO TO 140
      NS1=NSCTR
      NS2=NSCTR2
      DO 27 K = 1,IHSEC
      CALL EXEC(14,ICW,IBUF,128,IRULE,NS1)
```

```
      NS1=NS1+1
      CALL EXEC(15,ICW,IBUF,128,IRULE,NS2)
      NS2=NS2+1
      IF(NS2-NSCTR2.GE.ISECT)GO TO 167
27    CONTINUE
      KIX=3
      NS11=NS1
      DO 26 KJ=1,NOS2
      KIX2=2*KJ-1
      INC=(KIX2-1)/126
      CALL EXEC(14,ICW,IBUF,126,IRULE,NS11+INC)
      KIX2=KIX2-126*INC+2
      DO 29 KI=1,NOR
      IBUF2(KIX)=IBUF(KIX2)
      IBUF2(KIX+1)=IBUF(KIX2+1)
      KIX=KIX+2
      KIX2=KIX2+(2*NOS2)
      IF(KIX.EQ.129) GO TO 37
36    IF(KIX2.GT.128)GO TO 38
      GO TO 29
37    CALL EXEC(15,ICW,IBUF2,128,IRULE,NS2)
      NS2=NS2+1
      IF(NS2-NSCTR2.GE.ISECT)GO TO 167
      KIX=3
      DO 375 KT=1,128
375   IBUF2(KT)=0
      GO TO 36
38    KIX2=KIX2+INC*126-2
      INC=(KIX2-1)/126
      CALL EXEC(14,ICW,IBUF,126,IRULE,NS11+INC)
      KIX2=KIX2-126*INC+2
29    CONTINUE
28    CONTINUE
      IF(KIX.EQ.3)GO TO 39
      CALL EXEC(15,ICW,IBUF2,128,IRULE,NS2)
39    CONTINUE
      CALL FLPAK(6)
C..... RETURN TO MONITOR
7776  CALL OOSC(0)
      CALL EXEC(10,IPROG)
      CALL BRULE
7777  WRITE(1,8751)
8751  FORMAT("ILLEGAL GRADE INCREMENT")
      GO TO 801
167   WRITE (10,800)
800   FORMAT(" RULES FILE OUT OF SPACE")
801   CALL DIREC(2H06R,NRT,INTNO,NSCTR,ISTAT)
      CALL FLPAK(6)
      GOTO 7776
168   CONTINUE
      WRITE(10,873)
873   FORMAT(" ERROR IN RULE BUILD ")
      GO TO 801
169   CONTINUE
      WRITE(10,878)
878   FORMAT("TO MANY RULE FILES")
      GO TO 801
140    WRITE(10,700)ISTAT
700    FORMAT("DIRECTORY FAILURE IN BUILD, OF RULE, STATUS =",I3)
8     CALL LOCKO(0)
      WRITE(10,704)
704   FORMAT("BUILD INCOMPLETE")
      GO TO 801
500   WRITE(10,701)
701   FORMAT("POSSIBLE MISSING CARD")
```

```
      GO TO 801
874   WRITE(IO,751)
761   FORMAT("MISSING SIZE CARD(S)")
      GO TO 801
875   WRITE(IO,703)
703   FORMAT("MISSING RULE ")
      GO TO 801
      END
      ENDS
FTN4
       PROGRAM RENTR,5
C
C.....DEC 13 1972
C.....  BOB PATRICK
C..... RRULE PROGRAM
C
      COMMON IN,IO,ISEG(3),IRX(840),IRY(840),IBUF(128),IPFLG
      COMMON ISC(3),IPROG(3)
      DIMENSION ICOM(8)
      DIMENSION NAMR(6)
      DIMENSION IFIL(3)
      DATA MCOD/1234/
      DATA NRWS/128/
      DATA IFIL/2HRU,2HLE,2HS /
      CALL DDSC(0)
      ICOM(1)=2HBU
      ICOM(2)=2HED
      ICOM(3)=2HLI
      ICOM(4)=2HLU
      ICOM(5)=2HOU
      ICOM(6)=2HPU
      ICOM(7)=2H/E
      ICOM(8)=2H*A
      IF(IN.NE.1)GO TO3
      IN=IO
    1 WRITE(IO,100)
  100 FORMAT("ENTER INPUT DEVICE #",/,"$@")
2     READ(IN,*)IN
3     IF(IN.NE.1)GO TO 5
    4 WRITE(IO,101)
  101 FORMAT("ENTER OPERATION",/,"$@")
5     IF(IN.NE.5)GO TO 202
201   CALL RCRS(ISTAT)
      IF(ISTAT.LT.0)GO TO 202
      WRITE(IO,209)
209   FORMAT("**** CARD READER NOT READY ****")
      GO TO 201
202   CONTINUE
      READ(IN,200)IOP
  200 FORMAT(A2)
      DO 6 I=1,8
      IF(IOP.EQ.ICOM(I))GO TO 8
    6 CONTINUE
      WRITE(IO,102) IOP
  102 FORMAT("ILLEGAL OPERATION:  ",A2)
      IF(IN.EQ.1)GO TO 4
7     IBUF(1)=2
      IBUF(2)=3
      CALL EXEC(15,66,IBUF,2,ISC,0)
      CALL EXEC(10,IPROG)
       CALL RRULE
8     GO TO (10,11,14,13,15,16,7,19),I
10    CONTINUE
20     ISEG(1)=2HBn
       ISEG(2)=2HUL
```

```
      ISEG(3)=2H0
      CALL EXEC(8,ISEG)
       CALL GBULD
11    CONTINUE
12      ISEG(1)=2HBL
        ISEG(2)=2HIS
       ISEG(3)=2HT
      CALL EXEC(8,ISEG)
      CALL BLIST
C.....LU CHANGE
13    IF(IN.EQ.1)GO TO 1
      GO TO 2
C..... LIST
   14 IFLG=0
      GO TO 12
C..... DUMP
   15 IFLG=1
      GO TO 12
16    CONTINUE
      WRITE(10,103)
103   FORMAT("ENTER SECURITY CODE")
      READ(IN,*)ISCOD
      IF(ISCOD.NE.MCOD)GO TO 17
      WRITE(10,105)
105   FORMAT("ENTER RULE SET TO BE PURGED")
      READ(IN,106)NAMR
106   FORMAT(5A2)
      CALL DIREC(10068,NAMR,INTNO,NSCTR,ISTAT)
      IF(ISTAT.NE.0)GO TO 18
      CALL DIREC(24468,NAMR,INTNO,NSCTR,ISTAT)
      WRITE(10,107)(NAMR(J),J=1,5)
107   FORMAT("RULE"5A2,"PURGED")
      CALL FLPAK(6)
      GO TO 1
18    WRITE(10,108)(NAMR(I),I=1,5)
108   FORMAT(5A2," DOES NOT EXIST")
      GO TO 1
17    WRITE(10,109)
109   FORMAT("ILLEGAL SECURITY CODE FOR RULE PURGE")
      GO TO 1
C.....LIST NAMES OF ALL RULE TABLES IN SYSTEM
19    CONTINUE
      WRITE(10,111)
111   FORMAT (2X,"NAME",2X,"#SIZES",2X,"#RULES")
      DO 21 KX=1,NRRS
      NAMR(5)=KX
      CALL DIREC(30068,NAMR,INT2,NSCTR,ISTAT)
      IF(ISTAT.NE.0)GO TO 21
      CALL EXEC(14,60,IBUF,128,IFIL,NSCTR)
      NUMR=JLDF(JFLD(15,9),IBUF(1))
      NUMS=JLDF(JFLD(9,10),IBUF(2))
      WRITE (10,110) (NAMR(I),I=1,5),NUMS,NUMR
110   FORMAT (5A2,I4,5X,I3)
21    CONTINUE
      WRITE(10,112)
112   FORMAT("*** LIST END")
      GO TO 7
      END
      END$
FTN4
      PROGRAM BLIST,5
C
C..... DEC 15 1972
C..... BOB PATRICK
C..... RULE PROGRAM
```

```
C
      COMMON IN,IO,ISEG(3),IRX(840),IRY(840),IBUF(128),IPFLG
      COMMON ISC(3),IPROG(3)
      DIMENSION ILEAD(40)
      DIMENSION NRT(5)
      DIMENSION IRN(30),JBUF(30)
      DIMENSION KBUF(35)
      DATA ILEAD/40*0/
C.............      IFLG=0  LIST RULE
C.............      IFLG=1 DUMP RULE TO TTY
      IO=6
      IL=1
      IF(IN.EQ.1)WRITE(1,1234)
1234  FORMAT(" ENTER OUTPUT DEVICE")
      IF(IN.EQ.1)READ(1,1235)IO
      IF(IN.NE.5)GO TO 1237
1238  CALL WCRS(ISTAT)
      IF(ISTAT.LT.0)GO TO 1237
      WRITE(1,7752)
7752  FORMAT("***CARD READER NOT READY***")
      GO TO 1238
1237  CONTINUE
1235  FORMAT(I1)
      ICW=66
       DO 987 I=1,5
987     NRT(I)=2H
      ISEG(1)=2HRU
      ISEG(2)=2HLE
      ISEG(3)=2HS
      INCRF=JFLD(15,6)
      NR=JFLD(15,9)
      NS=JFLD(9,10)
      IF(IN.NE.1)GO TO 2
1     WRITE(1,100)
  100 FORMAT ("ENTER RULE SET ",/,"$")
    2 DO 3 I=1,30
    3 IRN(I)=0
      MON=0
      IJK=0
      READ (IN,500) NRT,KBUF
      IK=2H
      DO 23 I=1,35
      IF (KBUF(I).NE.IK) GO TO 24
   23 CONTINUE
      GO TO 13
   24 CALL CODE
      READ (KBUF,*) IJK,(IRN(I),I=1,IJK)
   13 CONTINUE
       CALL DIREC(1000,NRT,INTNO,NSCTR,ISTAT)
       IF(ISTAT.NE.0) GO TO 1000
    5 IF(IFLG.EQ.0)GO TO 6
  500 FORMAT (5A2,35A2)
      CALL EXEC(2,1000+ID,ILEAD,40)
      WRITE(IO,300)NRT
300   FORMAT("RU",/,5A2)
      GO TO 7
6     WRITE(IO,200)NRT
  200 FORMAT ("RULE SET ",5A2,/)
      IF (IJK.NE.0) GO TO 7
      WRITE (IO,22)
   22 FORMAT (/,"SIZE RELATION TABLE",/)
7      CALL EXEC(14,ICW,IBUF,128,ISEG,NSCTR)
       NSEC =1+NSCTR
      NOR=JLOF(NR,IBUF(1))
      NOS=JLOF(NS,IBUF(2))
```

```
      IF(IFLG.NE.0)GO TO 75
      IF (IJK.NE.0) GO TO 75
      INCR=JLOF(INCRF,IBUF(2))
      IF(INCR.EQ.1)INCR=32
      IF(INCR.EQ.2)INCR=64
      IF(INCR.EQ.3)INCR=128
      IF(INCR.EQ.4)INCR=256
      IF(INCR.EQ.5)INCR=512
      IF(INCR.EQ.6)INCR=1024
      IF(INCR.EQ.7)INCR=1M
      IF(INCR.EQ.8)INCR=100
      WRITE(IO,401)INCR
401   FORMAT("GRADE INCREMENT = ",I3)
      WRITE(IO,402)NOR,NOS
402   FORMAT("NUMBER OF RULES",I4,//,"NUMBER OF SIZES ",I4,//)
75    CONTINUE
        M=M
      NSZ=NOS+1
      DO 9 I=1,NSZ
8     M=M+1
       J=M*3
      K=J+2
       IF(M.GT.42)GO TO 95
      IF (IJK.NE.M) GO TO 9
      GO TO 88
  301 FORMAT(3A2)
88    WRITE(IO,201)I,(IBUF(L),L=J,K)
  201 FORMAT("ENTRY",I4,5X,3A2)
       GO TO 9
95    M=0
       CALL EXEC(14,ICW,IBUF,128,ISEG,NSEC)
        NSEC = NSEC+1
      GO TO 8
    9 CONTINUE
   21 IF (IFLG.EQ.1) WRITE (IO,302)
  302 FORMAT("/E")
      IF(IFLG.EQ.M)WRITE(IO,202)
  202 FORMAT(/,"RULE DATA",/)
      IFRSC=NSEC
       DO 17 J=1,NOR
      IF (IRN(1).EQ.0) GO TO 12
      IF (MON.LT.M) GO TO 17
      MON=MON+1
      IF (IRN(MON).GT.0) GO TO 4
      MON=-1
      GO TO 17
    4 IF (IRN(MON)-J) 10,12,11
   10 J=M
   11 MON=MON-1
      GO TO 17
   12 INDEX=2*(J-1)+1
      INC=(INDEX-1)/126
      NSEC=IFRSC+INC
      CALL EXEC(14,ICW,IBUF,128,ISEG,NSEC)
      INDEX=INDEX-126*INC+2
      NSEC=NSEC+1
      DO 15 JK=1,NSZ
      IRX(JK)=IBUF(INDEX)
      IRY(JK)=IBUF(INDEX+1)
      INDEX=INDEX+2*NOR
      IF(INDEX.GT.128)GO TO 14
       GO TO 15
14    INDEX=INDEX+INC*126-2
      INC=(INDEX-1)/126
      NSEC=IFRSC+INC
```

```
      CALL EXEC(14,ICW,IBUF,128,ISEG,NSEC)
      INDEX=INDEX+128*INL+2
   15 CONTINUE
      IF(IPFLG.EQ.0)GO TO 16
      WRITE(IO,303)(IRX(I),I=1,NSZ)
      WRITE(IO,303)(IRY(I),I=1,NSZ)
  303 FORMAT(42I5)
      GO TO 17
   16 ITY=1HX
      WRITE(IO,203) J,ITY,(IRX(I),I=1,NSZ)
      ITY=1HY
      WRITE(IO,203) J,ITY,(IRY(I),I=1,NSZ)
  203 FORMAT (I3,A2,12I5/3(5X,12I5/),10(/4(5X,12I5/)))
17      CONTINUE
   18 IF(IPFLG.EQ.0)GO TO 19
      WRITE(IO,304)
304   FORMAT("999",/,"/E")
      CALL EXEC(2,100B+ID,ILEAD,40)
      IF(IN.NE.IO)GO TO 20
      WRITE(IO,305)
  305 FORMAT(/,"DUMP END",/)
      GO TO 20
   19 WRITE(IL,204)
  204 FORMAT(/,"LIST END",/)
      IF(IL.EQ.IO)GO TO 20
      WRITE(IO,204)
20    IBUF(1)=2
      IBUF(2)=3
      CALL EXEC(10,IPROG)
      CALL PRULE
1000   WRITE(IO,600) NRT,ISTAT
600   FORMAT("RULE FILE",5A2," DOES NOT EXIST ", I5)
      GO TO 20
      END
      END$
FTN4
      PROGRAM BRULE,3
      COMMON IN,IO,ISEG(3),IRX(840),IRY(840),IBUF(128),IPFLG
      COMMON ISC(3),IPROG(3)
      DIMENSION IPAR(5)
      CALL RMPAR(IPAR)
      IF(IPAR(1).EQ.99)IN=5
      IF(IPAR(1).NE.99)IN=1
      IO=1
      ISC(1)=2HSE
      ISC(2)=2HCT
      ISC(3)=2HR
      IPROG(1)=2HWP
      IPROG(2)=2HRU
      IPROG(3)=2HU
      IPFLG=0
       ISEG(1)=2HBE
       ISEG(2)=2HNT
       ISEG(3)=2HR
      CALL EXEC(6,ISEG)
       CALL BENTR
       CALL BBULD
      CALL BLIST
      CALL DIREC
      CALL FLPAK
      CALL LOCKU
      CALL WCRS
      I=JFLD(15,3)
      I=JLDF(I,1)
      CALL JSTF
```

```
      WRITE(1,10)
10    FORMAT("RULE")
      END
      ENDS
FTN4
      PROGRAM PGRAB
C
C     VERSION 2.0
C     EDITED 06 FEB 73
C     RONALD GORDON
C
C     EXPANDED GRADE FILE
C     LASERMATIC DISK HANDLER FIXED
C     RULE TABLE FIX
C     GRADE OPTION FOR LASERMATIC
C     F737, REPLACE PIECE ON PLATTER
C     F955, PIECE NOT FOUND DIAGNOSTIC
C     F000, FIX FOR NO GRADE OPT
C     F170, STYLE QUEUE CHANGE
C     F200, MARKR QUEUE CHANGE
C     F260, ROTATE BIT FOR SIZE IN MARKR
C     F450, NEW RULE FILE
C     F600, NO GRADE IF SECTR=0
C     F736, REPLACE DUPLICATE PIECES
C     F955, LIST OF PIECES NOT FOUND
      DIMENSION MSEG1(3)
      LOGICAL PASS2
      LOGICAL FLAG1,FLAG2,FLAG3,FLAG4,FLAG5,FLAG6
      COMMON JOP(10), JRG(4), PASS2, MSIZE, NSIZE, IWORK(8200),
     +     JWORK(256), FLAG1,FLAG2,FLAG3,FLAG4,FLAG5,FLAG6,
     +     IPNTR,JPNTR,MPNTR, IPLATR
      DATA MSEG1 /2HQS,2HCA,2HN /
C                                  PARAMETER 99 MEANS NO GRADE
      CALL RMPAR (IWORK)
      FLAG1=.TRUE.
      IF (IWORK.EQ.99) FLAG1=.FALSE.
      MSIZE=8000
      NSIZE=256
      PASS2=.FALSE.
      CALL QDSC (0)
      CALL EXEC (8,MSEG1)
      CALL QSCAN
      CALL PGRAD
C     ROUTINES FOR ALL SEGMENTS
      CALL DIREC
      CALL NEWIO
      CALL JFLD
      CALL MOD
      CALL IABS
      CALL ISSW
      CALL FLNAM
      END
      PROGRAM QSCAN,5
C
C     THIS ROUTINE WILL SCAN THE QUEUES AND EXTRACT PIECES NEEDED
C
      LOGICAL GDPIEC, GDMARK
      LOGICAL MASHD
      LOGICAL OLDUN
      LOGICAL DOGRD
      DIMENSION NAME(5),MAME(3),IBUFR(128)
      DIMENSION IFREE(5)
      DIMENSION KRULE(6)
      DIMENSION MSEG2(3)
      DIMENSION IFILE1(3), IFILE2(3), IFILE3(3), IFILE5(3)
```

```
      DIMENSION IWPROD(3)
      DIMENSION IFILE4(3), JFILE4(3)
      DIMENSION JFILE5(3)
      DIMENSION IRSX(3)
      DIMENSION NTEMP(2)
      INTEGER CONCEL(3)
      INTEGER DONE
      LOGICAL FLAG1,FLAG2,FLAG3,FLAG4
      LOGICAL FLAG5,FLAG6
      LOGICAL PASS2
C                                     COMMON FOR GRADP ROUTINE
      COMMON JBP(10), JBG(4), PASS2, MSIZE, NSIZE, IWORK(8000),
     +     JWORK(256), FLAG1,FLAG2,FLAG3,FLAG4,FLAG5,FLAG6,
     +     IPNTR,JPNTR,MPNTR, IPLATR
C
      EQUIVALENCE (IBUFR(1), JWORK(1))
C          FOR RULE LOOK UP
      EQUIVALENCE (KRULE,NAME)
C          FOR NAME(6) REFRENCES
      EQUIVALENCE (IFREE,NAME)
      DATA MSEG2 /2HPG,2HRA,2HD /
      DATA IFILE4/2HGR,2HAD,2HC /, JFILE4 /2HGR,2HAD,2HE /
      DATA IWPROD /2HWP,2HRO,2HD /
      DATA ISTOR2 /4002B/, ILOOK5 /14056/
      DATA ISRCH1, ISRCH2, ISRCH3 /001001B, 001002B, 001003B/,
     +      ILOOK2 /00140ZB/
      DATA IREPL2 /40028/, IPURG2 /2402B/
      DATA IPURG5 /0024056/
      DATA DONE /-2/
      DATA NPLATR /1/
      DATA ISIZE /128/, IREAD /14/, IWRIT /15/, ICTRL /000102B/
      DATA IFILE1 /2HMA,2HRK,2HR /,
     +      IFILE2 /2HPI,2HEC,2HE /,
     +      IFILE3 /2HST,2HYL,2HE /,
     +      IFILE5 /2HTP,2HEC,2HS /
      DATA JFILE5 /2HTP,2HEC,2HD /
      DATA CONCEL /2HCH,2HEK,2H  /
      DATA NULL /-1/
      DATA OLDUN /.FALSE./
      DATA IO5 /5/
      DATA IRSX/2HRU,2HLE,2HS /
      DATA GOPIEC, GOMARK /2* .FALSE./
C***                                  SET UP COMMON AREA FOR MARGRET
      IF (PASS2) GO TO 900
      PASS2=.TRUE.
      JBP(1) = JFLD(15,7)
      JBP(2) = JFLD(8,8)
      JBP(3) = JFLD(5,1)
      JBP(4) = JFLD(4,1)
      JBP(5) = JFLD(3,4)
      JBP(6) = JFLD(15,9)
      JBP(7) = JFLD(6,6)
      JBP(8) = JFLD(0,1)
      JBP(9) = JFLD(15,4)
      JBP(10) = JFLD(11,12)
      JBG(1) = JFLD(12,4)
      JBG(2) = JFLD(8,9)
      JBG(3) = JFLD(15,8)
      JBG(4) = JFLD(7,7)
      MPNTR=NSIZE/2
C                                     QUEUE TYPE
      MASK1=JFLD(14,4)
C                                     RULE FIELD IN STYLE
      MASK3 = JFLD(6,7)
C                                     NSECTS IN DATA FILE HEADER
```

```
      MASK4=JFLD(3,4)
C                                     RULE FIELD IN NESTED PLOT QUEUE
      MASK5=JFLD(13,6)
C                                     RULE IN MARKER
      MASK6 = JFLD(9,7)
C                                     PLATER NUMBER FOR PIECES
      MASK7=JFLD(15,3)
C                                      OLD MARKER QUEUE BIT
      MASK8=JFLD(13,1)
C                                      #SECTORS, #PIECES IN MARKR
      MASK9=JFLD(5,6)
      MASK10=JFLD(15,10)
C                                     RIGHT AND LEFT HALF WORD
      MASKR=JFLD(7,8)
      MASKL=JFLD(15,8)
C
C     STEP THROUGH QUEUE.  LOOK FOR TYPES 4 THROUGH 7.
C
      FLAG4=.TRUE.
      IPNTR=0
      JPNTR=MSIZE+1
      IF (.NOT.FLAG1) GO TO 640
C                                     FETCH SIZE OF QUEUE
      CALL WQST (2,LQ)
      LQ=LQ-2
      IF (LQ.LT.7) GO TO 640
      DO 600 IQ=7,LQ,2
C                                     GET TYPE FOR THIS ENTRY
      CALL WQST (IQ,IQWORD)
      ITYPE=JLDF(MASK1,IQWORD)
      IF (ITYPE.LT.4) GO TO 600
      IF(ITYPE.GT.6) GO TO 600
C                                     GET DATA VECTOR, CHECK IF PURGED
      CALL WQST (IQ+1,IQWORD)
      IF (IQWORD.LT.257) GO TO 600
C                                     IQWRD POINTS TO START OF DATA NOW
      ITYPE=ITYPE-3
      GO TO (150,250,350,450,550),ITYPE
C
C     PROCESS STYLE DEFINITION REQUEST                          QUEUE=4
C
  150 CONTINUE
C                                     PICK UP NUMBER OF PIECES IN STYLE
      CALL WQST (IQWORD+6,ITEMP)
      NPIEC=JLDF(MASKR,ITEMP)
C                                     GOT ROOM FOR THAT MUCH?
      IF (JPNTR-IPNTR-NPIEC*6 .GT. 0) GO TO 160
C                                     MOVE NAME OF STYLE OUT
      DO 155 I=1,5,1
  155 CALL WQST (IQWORD+1,NAME(I))
      WRITE (106,1100) NAME
      GO TO 600
  160 IQWORD=IQWORD+7
      DO 170 I=1,NPIEC,1
C                                     TRANSFER PIECE NAME
      JPNTR=JPNTR-6
      DO 165 J=0,4,1
  165 CALL WQST (IQWORD+J,IWORK(JPNTR+J))
      IWORK(JPNTR+5)=NULL
  170 IQWORD=IQWORD+10
      GO TO 600
C
C     MARKER REQUEST ENTRY                                      QUEUE=5
C
  250 CONTINUE
```

```
C                                       GET NUMBER OF STYLES FOR THIS ONE
      CALL WGST (IGWORD+8,ITEMP)
      NSTYL=JLOF(MASKR,ITEMP)
      OLDUN=JLOF(MASK8,ITEMP).EQ.1
      IGWORD=IGWORD+13
  255 CONTINUE
      IF (.NOT.OLDUN.AND.NSTYL.EQ.0) GO TO 600
C                                       DO FOR ALL PIECES IN EACH STYLE
      DO 310 IS=1,NSTYL,1
C                                       GET STYLE NAME
      DO 200 I=1,5,1
  200 CALL WGST (IGWORD+I,NAME(I))
      IF (OLDUN) GO TO 305
C                                       GET STUFF FROM DIREC
      CALL DIREC (ISRCH3,NAME,INTNO,NSCTR,ISTAT)
      IF (ISTAT.EQ.0) GO TO 270
      WRITE (IO6,1500) NAME
      GO TO 600
  270 ISCTR=NSCTR
      CALL EXEC (IREAD,ICTRL,IBUFR,ISIZE,IFILE3,NSCTR)
      IRULE = JLOF(MASK3,IBUFR(2))
      LP=JLOF(MASKL,IBUFR)
      LSCTR=ISCTR+JLOF(MASK4,IBUFR)
C                                       GET NUMBER OF SIZES
      CALL WGST (IGWORD+6,ITEMP)
      NSIZE=JLOF(MASKR,ITEMP)
      IF (JPNTR=IPNTR-NSIZE*LP*8 .GT. 0) GO TO 275
      WRITE (IO6,1100) NAME
      GO TO 600
  275 MASHD=.FALSE.
      IGWORD=IGWORD+6
      DO 310 IZ=1,NSIZE,1
      IP=0
C                                       GET SIZE FOR EACH PIECE IN STYLE
      DO 280 I=1,3,1
  280 CALL WGST (IGWORD+I,NAME(I))
      NAME(1)=IAND(NAME(1),0777778)
      IF (ISSW(0).LT.0) WRITE (1,1221) IRULE,NAME
 1221 FORMAT("SIZEP: ",O6,2X,3A2)
      CALL SIZEP (1,IRULE,NAME,KSIZE)
      IF (KSIZE) 283,286,290
  283 WRITE (IO6,8090) NAME
      GO TO 600
  286 WRITE (IO6,8080) IRULE
      GO TO 600
C                                       CHECK IF NEED NEXT BUFFER IN CORE
  290 IF (MASHD) CALL EXEC (IREAD,ICTRL,IBUFR,ISIZE,IFILE3,ISCTR)
      ISCTR=ISCTR+1
      IP = IP+25
      IF (IP.GT.LP) IP=LP
      L = MOD(IP-1,25)*5+3
      DO 300 J=3,L,5
      IWORK(IPNTR+1)=NULL
      IWORK(IPNTR+6)=IBUFR(J)
      IWORK(IPNTR+7)=KSIZE
      CALL JSTF (IRULE,MASKL,IWORK(IPNTR+7))
      IWORK(IPNTR+8)=IBUFR(J+4)
  300 IPNTR=IPNTR+8
      IF (ISCTR.GE.LSCTR) GO TO 305
C                                       SET FLAG - BUFFER TO COME IN
      MASHD=.TRUE.
      GO TO 290
  305 IGWORD=IGWORD+3
  310 ISCTR=NSCTR
      GO TO 600
```

```
C
C       DRAW A MARKER COMMAND                                   QUEUE=6
C
  350 CONTINUE
C                                       GET MARKER ID
      DO 360 I=1,5,1
  360 CALL WGST (IGWORD+I,NAME(I))
C                                       GET MARKER INFO
  365 IF (FLAG4) CALL MDISK (-1)
      FLAG4=.FALSE.
      CALL DIREC (ISRCH1,NAME,INTNO,NSCTR,ISTAT)
      IF (ISTAT.EQ.0) GO TO 370
      IF (OLDUN.AND.NSTYL.GT.0) GO TO 420
      WRITE (IO6,1200) NAME
      GO TO 600
  370 ISCTR=NSCTR
  380 CALL EXEC (IREAD,ICTRL,IBUFR,ISIZE,IFILE1,ISCTR)
      IF (ISCTR.NE.NSCTR) GO TO 400
      LSCTR=NSCTR+JLOF(MASK9,IBUFR)
      LP = JLOF(MASK10,IBUFR)
      IP=0
      ISCTR=ISCTR+1
C                                       GOT ROOM FOR MORE OF THIS STUFF?
      IF (JPNTR-IPNTR-LP*8 .GT.0) GO TO 380
      WRITE (IO6,1400) NAME
      GO TO 600
  400 ISCTR=ISCTR+1
      IP = IP+15
      IF (IP.GT.LP) IP=LP
      L = MOD(IP-1,15)*8+3
      DO 410 I=3,L,8
      IWORK(IPNTR+1)=NULL
      IWORK(IPNTR+6)=IBUFR(I)
      IWORK(IPNTR+7)=IBUFR(I+4)
      IRULE=JLOF(MASK6,IBUFR(I+2))
      CALL JSTF (IRULE,MASKL,IWORK(IPNTR+7))
      IWORK(IPNTR+6)=IBUFR(I+3)
  410 IPNTR=IPNTR+8
      IF (ISCTR.LT.LSCTR) GO TO 380
      IF (.NOT.OLDUN) GO TO 600
  420 CONTINUE
      OLDUN=.FALSE.
      IGWORD=IGWORD+5
      GO TO 250
C
C       PLOT NESTED PIECE                                       QUEUE=7
C
  450 CONTINUE
C                                       GET PIECE ID
      DO 460 I=1,5,1
  460 CALL WGST (IGWORD+I,NAME(I))
C                                       PICK UP RULE AND NUMBER OF SIZES
      IGWORD=IGWORD+1
      CALL WGST(IGWORD+6,ITEMP)
      IRULE=JLOF(MASK5,ITEMP)
      NSIZE=JLOF(MASK8,ITEMP)
      IF (NSIZE.GT.0) GO TO 465
      NAME(6)=IRULE
      CALL DIREC(30068,NAME,INTNO,NSCTR,ISTAT)
      IF (ISTAT.NE.0) GO TO 286
      NSX=JFLD(9,10)
      CALL DDSC(6)
      CALL EXEC(14,1028,IBUFR,2,IRSX,NSCTR)
      NSIZE=JLOF(NSX,IBUFR(2))
      IF (JPNTR-IPNTR-NSIZE*8.LE.0) GO TO 466
```

```
      DO 462 I=1,NSIZE,1
      DO 461 J=1,5,1
  461 IWORK(IPNTR+J)=NAME(J)
      IWORK(IPNTR+6)=NULL
      IWORK(IPNTR+7)=I
      CALL JSTF (IRULE,MASKL,IWORK(IPNTR+7))
      IWORK(IPNTR+8)=0
  462 IPNTR=IPNTR+8
      GO TO 600
C                                     CHECK FOR SIZE AGAIN... HO HUMM...
  465 IF (JPNTR-IPNTR-NSIZE*8 .GT. 0) GO TO 470
  466 WRITE (IO6,1600) NAME
      GO TO 500
  470 IQWORD=IQWORD+6
      DO 490 I=1,NSIZE,1
C                                     STICK IN PIECE NAME
      DO 480 J=1,5,1
  480 IWORK(IPNTR+J)=NAME(J)
      IWORK(IPNTR+6)=NULL
C                                     DO YOUR THING WITH PIECE SIZES
      DO 485 J=1,3,1
  485 CALL MOST (IQWORD+J,NAME(J))
      CALL SIZEP(1,IRULE,NAME,KSIZE)
      IF (KSIZE) 283,286,486
C                                     TRANSFER RULE AND SIZE
  486 IWORK(IPNTR+7)=KSIZE
      CALL JSTF (IRULE,MASKL,IWORK(IPNTR+7))
      IWORK(IPNTR+6)=0
      IQWORD=IQWORD+3
      IPNTR=IPNTR+8
  490 CONTINUE
      GO TO 600
C
C     SET FLAG TO GRADE ZERO FLAGGED PIECES
C
  550 CONTINUE
      CALL MOST (IQWORD+1,ITEMP)
      IF (ITEMP.EQ.1) GDMARK=.TRUE.
      IF (ITEMP.EQ.2) GDPIEC=.TRUE.
  600 CONTINUE
C
C     GRADE PIECES AND MARKERS CALLED FOR IN LASERMATIC SYSTEM
C
      IF (.NOT.GDMARK) GO TO 575
      IF (FLAG4) CALL MDISK (-1)
      FLAG4=.TRUE.
      WRITE (IO6,9060)
      INTNO=-1
  560 INTNO=INTNO+1
      CALL DIREC (34W1B,NAME,INTNO,NSCTR,ISTAT)
      IF (ISTAT.EQ.5) GO TO 575
      IF (ISTAT.NE.0) GO TO 560
      IF (NAME(6).NE.0) GO TO 560
      WRITE (IO6,9030) NAME
  570 ISCTR=NSCTR
  571 CALL EXEC (IREAD,ICTRL,IBUFR,ISIZE,IFILE1,ISCTR)
      IF (ISCTR.NE.NSCTR) GO TO 572
      LSCTR=NSCTR+JLOF(MASK4,IBUFR)
      LP=JLOF(MASK10,IBUFR)
      IP=0
      ISCTR=ISCTR+1
      IF (JPNTR-IPNTR-LP*8.GT.0) GO TO 571
      WRITE (IO6,1400) NAME
      GO TO 575
  572 ISCTR=ISCTR+1
```

```
      IP=IP+15
      IF (IP.GT.LP) IP=LP
      L=MOD(IP-1,15)*8+3
      DO 574 I=3,L,8
      IWORK(IPNTR+1)=NULL
      IWORK(IPNTR+6)=IBUFR(I)
      IWORK(IPNTR+7)=IBUFR(I+4)
      IRULE=JLOF(MASK6,IBUFR(I+2))
      CALL JSTF (IRULE,MASKL,IWORK(IPNTR+7))
      IWORK(IPNTR+5)=IBUFR(I+3)
  574 IPNTR=IPNTR+8
      IF (ISCTR.LT.LSCTR) GO TO 571
      GO TO 560
C
C     GRADE PIECES CALLED FOR
C
  575 CONTINUE
      IF (.NOT.GDPIEC) GO TO 595
      CALL MOISK (1)
      WRITE (IO6,9270)
      NSX = JFLD(9,10)
      IRL=JFLD(15,7)
      INTNO = -1
  580 INTNO = INTNO+1
      CALL DIREC(3402B,NAME,INTNO,NSCTR,ISTAT)
      IF(ISTAT.EQ.5) GO TO 595
      IF(ISTAT.NE.0) GO TO 580
      IF (NAME(6).NE.0) GO TO 580
      WRITE (IO6,9030) NAME
      CALL EXEC (14,1002B,NAME,2,IFILE2,NSCTR)
      IRULE=JLOF(IRL,NAME(2))
      NAME(6) = IRULE
      CALL DIREC(3006B,NAME,JNTNO,NSCTR,ISTAT)
      IF (ISTAT.EQ.0) GO TO 582
      WRITE (IO6,8050) IRULE
      GO TO 580
  582 CONTINUE
      CALL EXEC(14,1002B,IBUFR,2,IRSX,NSCTR)
      NSIZE = JLOF(NSX,IBUFR(2))
      IF (JPNTR-IPNTR-NSIZE*8.GT.0) GO TO 583
      WRITE (IO6,1600) NAME
      GO TO 595
  583 DO 585 I = 1,NSIZE,1
      DO 584 J = 1,5,1
  584 IWORK(IPNTR+J) = NAME(J)
      IWORK(IPNTR+6) = INTNO
      IWORK(IPNTR+7) = I
      CALL JSTF(IRULE,MASKL,IWORK(IPNTR+7))
      IWORK(IPNTR+8) = 0
  585 IPNTR = IPNTR+8
      GO TO 580
  595 CONTINUE
C
C     ZERO GRADE FILE IF NEW PIECES ARE TO BE ADDED
C
      IF (IPNTR.EQ.0) GO TO 640
      CALL UDSC (0)
      CALL EXEC (18,IFILE4,LIMIT)
      DO 610 ISECT=0,LIMIT-1,1
  610 CALL EXEC (IWRIT,ICTRL,0,1,IFILE4,ISECT)
      CALL EXEC (IWRIT,ICTRL,0,1,JFILE4,0)
      I=IPNTR/8
      WRITE (IO6,9000) I
C
C     TOP OF MULTI PLATTER LOOP
```

```
C
  640 CONTINUE
      FLAG1=.FALSE.
      FLAG2=.FALSE.
      FLAG6=.TRUE.
      IPLATR=0
  700 CONTINUE
      IPLATR=IPLATR+1
      FLAG4=.FALSE.
C
C     TRANSFER PIECES FROM TPECD TO PIECE.  FLAG1 IS TRUE IF WE'VE
C     GONE THROUGH HERE LAST TIME AND HAVE FOUND NO PIECES LEFT TO MOVE.
C
      MASHD=.FALSE.
      IF (FLAG1) GO TO 775
      FLAG1=.TRUE.
      INTNO=-1
      NAME(6)=0
      FLAG5=.TRUE.
      MASHD=.FALSE.
  730 INTNO=INTNO+1
      CALL DIREC (ILOOK5,NAME,INTNO,NSCTR,ISTAT)
      IF (ISTAT.EQ.5) GO TO 775
      IF (ISTAT.NE.0) GO TO 736
      IF (MASHD) GO TO 733
      IF (JPNTR-IPNTR+1 .GE. 1536) GO TO 733
      CALL UDSC (2)
      IF (IPNTR.GT.0) CALL EXEC (IWRIT,ICTRL,IWORK,IPNTR,COMCEL,0)
      IF (MSIZE-JPNTR+1.GT.0) CALL EXEC (IWRIT,ICTRL,IWORK(JPNTR),
     +     MSIZE-JPNTR+1,COMCEL,(IPNTR-1)/128+1)
      ITEMP=IPNTR
      IPNTR=0
      MASHD=.TRUE.
  733 CALL EXEC (IREAD,ICTRL,IWORK(IPNTR+1),128,IFILE5,NSCTR)
      NSECT = JLOF(JBP(5),IWORK(IPNTR+1))
      IF (NSECT.GT.0) GO TO 735
      CALL ERR (31,1)
      GO TO 760
  735 IF (NSECT.GT.1) CALL EXEC (IREAD,ICTRL,IWORK(IPNTR+129),
     +     (NSECT-1)*128,IFILE5,NSCTR+1)
      IF (FLAG4) GO TO 736
      FLAG4=.TRUE.
      CALL MDISK (IPLATR)
      CALL EXEC (16,IFILE2,LIMIT)
  736 CALL DIREC (ISTOR2,NAME,JNTNO,JSCTR,ISTAT)
  737 IF (ISTAT.EQ.0) GO TO 740
      IF (ISTAT.EQ.1) GO TO 748
      WRITE (IO6,9010) NAME
      READ (1,9030) ISTAT
      IF (ISTAT.NE.2HYE) GO TO 760
      CALL DIREC (IREPL2,NAME,JNTNO,JSCTR,ISTAT)
      FLAG5=.TRUE.
      GO TO 737
  740 IF (JSCTR+NSECT.LE.LIMIT) GO TO 750
  745 FLAG1=.TRUE.
      CALL DIREC (IPURG2,NAME,JNTNO,JSCTR,ISTAT)
      IF (JSCTR.LT.LIMIT) CALL EXEC (IWRIT,ICTRL,1,1,IFILE2,JSCTR)
  748 FLAG6=.TRUE.
      FLAG1=.FALSE.
      WRITE (1,9000)
      GO TO 775
  750 IF (FLAG5) WRITE (IO6,9040)
      FLAG5=.FALSE.
      FLAG6=.FALSE.
      WRITE (IO6,9030) NAME
```

```
      CALL EXEC (IWRIT,ICTRL,IWORK(IPNTR+1),NSECT*128,IFILE2,JSCTR)
  760 CALL DIREC (IPURGS,NAME,INTNO,NSCTR,ISTAT)
      GO TO 730
C
C     PRESORT TO CUT DOWN REDUNDANT DIREC CALLS
C
  775 CONTINUE
      IF (.NOT.MASHD) GO TO 776
      IPNTR=ITEMP
      CALL UDSC (0)
      IF (IPNTR.GT.0) CALL EXEC (IREAD,ICTRL,IWORK,IPNTR,COMCEL,0)
      IF (MSIZE-JPNTR+1.GT.0) CALL EXEC (IREAD,ICTRL,IWORK(JPNTR),
     +     MSIZE-JPNTR+1,COMCEL,(IPNTR-1)/128+1)
  776 CONTINUE
      IF (ISSW(0).LT.0) WRITE (1,1952) (IWORK(I), I=1,IPNTR,1)
C
C     FILL IN EMPTY AREAS IN GRADE LIST FROM INFO ON CURRENT PLATTER
C
      IF (FLAG2) GO TO 900
      IF (IPNTR.EQ.0) GO TO 900
      FLAG2=.TRUE.
      DO 790 I=1,IPNTR,8
      IF (IWORK(I+6).EQ.DONE) GO TO 790
      IF (.NOT.FLAG4) CALL MOISK (IPLATR)
      FLAG4=.TRUE.
      IF (IWORK(I).NE.NULL) GO TO 780
C     IF (JLOF(MASK7,IWORK(I+5)).NE.IPLATR-1) GO TO 786
      CALL DIREC (ILOOK2,IWORK(I),IWORK(I+5),NSCTR,ISTAT)
      IF (ISTAT) 785,790,786
  780 IF (IWORK(I+5).NE.NULL) GO TO 790
      IF (IWORK(I).NE.IWORK(I-8)) GO TO 805
      IF (IWORK(I+1).NE.IWORK(I-7)) GO TO 805
      IF (IWORK(I+2).NE.IWORK(I-6)) GO TO 805
      IF (IWORK(I+3).NE.IWORK(I-5)) GO TO 805
      IF (IWORK(I+4).NE.IWORK(I-4)) GO TO 805
      IF (I.EQ.1) GO TO 805
      IWORK(I+5)=IWORK(I-3)
      IRULE=JLOF(MASKL,IWORK(I+6))
      IF (IRULE.EQ.0)
     +     CALL JSTF (JLOF(MASKL,IWORK(I-2)),MASKL,IWORK(I+6))
      GO TO 790
  805 CALL DIREC (ISRCH2,IWORK(I),IWORK(I+5),NSCTR,ISTAT)
      IF (ISTAT.NE.0) GO TO 786
  808 IRULE=JLOF(MASKL,IWORK(I+6))
      IF (IRULE.NE.0) GO TO 790
      CALL EXEC (IREAD,ICTRL,NTEMP,2,IFILE2,NSCTR)
      IRULE=JLOF(JOP(1),NTEMP(2))
      CALL JSTF (IRULE,MASKL,IWORK(I+6))
      GO TO 790
  786 FLAG2=.FALSE.
  790 CONTINUE
C
C     SORT GRADE LIST FOR REDUNDANCY CHECK
C
      L=IPNTR-8
      IF (L.EQ.0) GO TO 865
      DO 860 I=1,L,8
      M=I+8
      DO 860 J=M,IPNTR,8
C                                     SORT ON INTNO
      IF (IWORK(I+5)-IWORK(J+5)) 840,810,860
810   IF (IWORK(I+6)-IWORK(J+6)) 860,820,840
C                                       % INCREASE FIELD
820   IF (IWORK(I+7)-IWORK(J+7)) 860,860,840
  840 DO 850 K=0,7,1
```

```
      ITEMP=IWORK(I+K)
      IWORK(I+K)=IWORK(J+K)
  850 IWORK(J+K)=ITEMP
  860 CONTINUE
C
C     GRADE THOSE PIECES FOUND ON THIS PLATTER
C     DO PROGRAM LOAD FOR GRADE
C
  865 CONTINUE
      CALL DOSC (0)
      CALL EXEC (8,MSEG2)
      CALL PGRAD
      CALL PGRAB
C
C     GET NUMBERS FOR STYLE DEFINITION
C
  900 CONTINUE
      IF (JPNTR.GT.MSIZE) GO TO 955
      DO 950 I=JPNTR,MSIZE,8
      IF (IWORK(I+5).NE.NULL) GO TO 950
      IF (.NOT.FLAG4) CALL MDISK (IPLATR)
      FLAG4=.TRUE.
      CALL DIREC (ISRCH2,IWORK(I),IWORK(I+5),NSCTR,ISTAT)
  950 CONTINUE
  955 IF (IPNTR.EQ.0) GO TO 982
      DO 980 I=1,JPNTR,8
      IF (IWORK(I+5).NE.NULL) GO TO 980
      IF(I.EQ.1) GO TO 960
      IF(IWORK(I).NE.IWORK(I-8)) GO TO 960
      IF(IWORK(I+1).NE.IWORK(I-7)) GO TO 960
      IF(IWORK(I+2).NE.IWORK(I-6)) GO TO 960
      IF(IWORK(I+3).NE.IWORK(I-5)) GO TO 960
      IF(IWORK(I+4).EQ.IWORK(I-4)) GO TO 980
  960 L = I+4
      WRITE (106,9050) (IWORK(J), J=I,L,1)
  980 CONTINUE
  982 IF (IPLATR.LT.NPLATR) GO TO 700
C
C     DONE WITH ALL PLATERS
C     DUMP PIECE NUMBERS FOR STYLE DEFINITIONS, DO STUF FOR EXEC.
C
      CALL MDISK (-1)
      JPNTR=JPNTR-1
      IWORK(JPNTR)=MSIZE-JPNTR+1
      CALL DOSC (0)
      CALL EXEC (IWRIT,ICTRL,IWORK(JPNTR),IWORK(JPNTR),COMCEL,0)
      IF (FLAG6) GO TO 1116
      CALL EXEC (18,JFILE5,LIMIT)
      LIMIT=LIMIT-1
      DO 1101 I=0,LIMIT,1
 1101 CALL EXEC (IWRIT,ICTRL,0,1,JFILE5,I)
      CALL EXEC (IWRIT,ICTRL,0,1,IFILE5,0)
 1116 CONTINUE
C                                        RELOAD DAVE'S PSUEDO EXEC
      CALL EXEC (10,IWPROD)
 1100 FORMAT("NO ROOM FOR PIECES OF STYLE ",5A2)
 1200 FORMAT("CAN'T FIND MARKER ",5A2)
 1300 FORMAT("MARKER ",5A2," MISSING STYLE",I5,"?")
 1400 FORMAT("NO ROOM FOR PIECES OF MARKER ",5A2)
 1500 FORMAT("STYLE ",5A2," NOT FOUND?")
 1700 FORMAT("MISSING PIECE",I5,"?")
 1800 FORMAT("NO ROOM FOR PIECE ",5A2)
 1952 FORMAT(807)
 8010 FORMAT("PIECE OMITED: ",5A2)
 8080 FORMAT("RULE TABLE ",I3," NOT FOUND?")
```

```
8090 FORMAT("SIZE ",3A2," NOT AVALIABLE?")
9000 FORMAT("THIS PIECE PLATTER IS FULL!")
9010 FORMAT(/,5A2," DUPLICATE NAME - REPLACE?",/,'$')
9020 FORMAT("NO ROOM FOR PIECE ",5A2)
9030 FORMAT(5A2)
9040 FORMAT(/,"PIECES TRANSFERED")
9050 FORMAT("MARKERS FOR LASERMATIC:")
9070 FORMAT("PIECES FOR LASERMATIC:")
9080 FORMAT(/,I4," PIECES IN GRADE LIST")
9090 FORMAT("DIDN'T FIND ",5A2)
     END
     PROGRAM PGRAD,5
     DIMENSION MSEG1(3)
     LOGICAL PASS2
     LOGICAL FLAG1,FLAG2,FLAG3,FLAG4,FLAG5,FLAG6
     INTEGER DONE
     COMMON JOP(10), JBG(4), PASS2, MSIZE, NSIZE, IWORK(8020),
    +      JWORK(250), FLAG1,FLAG2,FLAG3,FLAG4,FLAG5,FLAG6,
    +      IPNTR,JPNTR,MPNTR, IPLATR
     DATA MSEG1 /2HUS,2HCA,2HN /
     DATA DONE /-2/, NULL /-1/
     DATA IO6 /6/
 865 I=1
     J=1
     LSIZE=NULL
     LPRCT=NULL
     KPNTR=0
     L=IPNTR+1
     IF (ISS4(0).LT.0) WRITE (IO6,1952) (IWORK(K), K=1,IPNTR,1)
1952 FORMAT(8O7)
     DO 890 J=1,L,8
     IF (J.EQ.L) GO TO 870
     IF (IWORK(J+6).EQ.DONE) GO TO 890
     IF (IWORK(J+5).EQ.NULL .OR. IWORK(J).EQ.NULL) GO TO 890
     IF (IWORK(I+5).EQ.IWORK(J+5)) GO TO 880
 870 CONTINUE
     IF (KPNTR.EQ.0) GO TO 1875
     CALL GRADP (1,IWORK(I),KPNTR,JWORK,ISTAT,JWORK(MPNTR))
1875 IF (J.EQ.L) GO TO 900
     I=J
     IF (KPNTR.EQ.0) GO TO 885
     KPNTR=0
     IF (ISTAT.EQ.0) GO TO 885
     GO TO (871,872,873,874,875,876,877), ISTAT
 871 WRITE (IO6,8030) (IWORK(I+K), K=-8,-4,1)
     GO TO 885
 872 WRITE (IO6,8020)
     GO TO 900
 873 WRITE (IO6,8030) (IWORK(I+K), K=-8,-4,1)
     GO TO 885
 874 WRITE (IO6,8040)
     CALL ERR (50,0)
 875 WRITE (IO6,8050)
     CALL ERR (50,0)
 876 WRITE (IO6,8060) (IWORK(I+K), K=-8,-4,1)
     GO TO 885
 877 WRITE (IO6,8070) (IWORK(I+K), K=-8,-4,1)
     GO TO 885
 880 IF (LSIZE.EQ.IWORK(J+6).AND.LPRCT.EQ.IWORK(J+7))
    +      GO TO 889
 885 LSIZE=IWORK(J+6)
     LPRCT=IWORK(J+7)
     KPNTR=KPNTR+1
     IF (KPNTR.LT.MPNTR) GO TO 888
     KPNTR=KPNTR-1
```

```
      WRITE (1,8090) (IWORK(I+K), K=0,4,1)
  886 J=J+1
      IF (J.EQ.L) GO TO 870
      IF (IWORK(I+5).EQ.IWORK(J+5)) GO TO 886
      GO TO 870
  888 CONTINUE
      JWORK(MPNTR+KPNTR-1)=IWORK(J+7)
      JWORK(KPNTR)=LSIZE
  889 IWORK(J+5)=DONE
  890 CONTINUE
  900 CONTINUE
      CALL UOSC (0)
      CALL EXEC (8,MSEG1)
      CALL OSCAN
      CALL PGRAB
 8020 FORMAT("GRADE FILE FULL!")
 8030 FORMAT("PIECE NOT FOUND ",5A2,"?")
 8040 FORMAT("FIXED DISK ERROR IN GRADE")
 8050 FORMAT("REMOVABLE DISK ERROR IN GRADE")
 8060 FORMAT("PIECE PREVIOUSLY GRADED ",5A2,"?")
 8070 FORMAT("PIECE TOO LARGE ",5A2,"?")
 8080 FORMAT("TOO MANY SIZES FOR ",5A2)
      END
      SUBROUTINE MDISK(I)
      INTEGER PIECD(3),MARKD(3)
      DATA PIECD /2HPI,2HEC,2HD /, MARKD /2HMA,2HRK,2HD /
      CALL UOSC (1)
      CALL UOLBL (N)
      IF (I.EQ.-1) GO TO 200
      CALL EXEC (16,PIECD,M)
      IF (M.GT.0.AND.N.EQ.I) RETURN
C                                        DISK 1 IS OPTIONAL
      IF (N.EQ.1 .AND. I.EQ.1 .AND. M.EQ.0) I=1+1
  100 WRITE (1,1000)
      PAUSE
      CALL EXEC (23,0,1,1)
      CALL EXEC (16,PIECD,M)
      IF (M.GT.0) RETURN
      GO TO 100
  200 CALL EXEC (16,MARKD,M)
      IF (N.EQ.1.AND.M.GT.0) RETURN
  300 WRITE (1,2000)
      PAUSE
      CALL EXEC (23,0,1,1)
      CALL EXEC (16,MARKD,M)
      IF (M.GT.0) RETURN
      GO TO 300
 1000 FORMAT("MOUNT PIECE PLATTER, THEN TYPE  :GO")
 2000 FORMAT("MOUNT MARKER PLATTER, THEN TYPE  :GO")
      END
END$
```

What is claimed is:

1. A system for laying out an apparel pattern marker wherein a plurality of apparel pattern pieces are arranged within a predetermined marker area comprising:

screen means for displaying to an operator scaled miniature representations of said pattern pieces adjacent a scaled miniature representation of said marker area;

positioning means movable by the operator to selectively move said representations of said pattern pieces one at a time on said screen means to within said representation of said marker area, wherein said representations of said pattern pieces may be efficiently arranged within said representation of said marker area to form a simulated marker; and

means responsive to the orientation of said representations of said pattern pieces forming said simulated marker on said screen for plotting the full size marker layout of said pattern pieces in the predetermined full size marker area.

2. The system of claim 1 and further comprising:

a data tablet interconnected with said screen means and said positioning means for generating electrical signals representative of the position of said positioning means relative to said data tablet, and

means responsive to said electrical signals for causing selective movement of said representations of said pattern pieces on said screen means.

3. The system of claim 2 and further comprising:

circuitry responsive to movement of said positioning means relative to said data tablet for causing corresponding movement of a light cursor over said screen means.

4. The system of claim 3 and further comprising pick means for being operated such that pattern pieces displayed on said screen jump to a predetermined location relative to said cursor when said cursor is moved adjacent said pattern pieces.

5. The system of claim 1 and further comprising:

a function box interconnected with said screen means and including function buttons for varying the position of said pattern pieces within said displayed marker area.

6. The system of claim 5 wherein said function box includes a button for causing selected pattern pieces to be abutted within said displayed marker area.

7. The system of claim 5 wherein said function box includes a button for causing a pattern piece to be selectively rotated within said displayed marker area.

8. The system of claim 5 wherein said function box includes a button for causing a pattern piece to be flipped over within said displayed marker area.

9. A system for preparing an apparel pattern marker comprising:

a digitizer for generating digital coordinate signals representative of the coordinates of points defining a plurality of model size pattern pieces;

means for storing said coordinate signals and for grading said stored model pattern pieces according to predetermined grading rules;

a display screen operable in response to said stored coordinate signals for displaying miniature representations of a predetermined size of pattern pieces and a miniature representation of a marker area;

a data tablet interconnected with said display screen and said storing and grading means and including a stylus movable adjacent said data tablet,

circuitry responsive to movement of said stylus relative to said data tablet for selectively causing movement of said pattern pieces on said display screen into said marker area;

a function box operable to arrange said pattern pieces in an efficient abutting relationship within said marker area; and

a plotter for plotting a full sized marker corresponding to a final arrangement of pattern pieces within said marker area on said display screen.

10. The system of claim 9 wherein said digitizer includes a light for being illuminated if the operator does not operate said digitizer in accordance with predetermined criteria.

11. The system of claim 9 wherein said digitizer includes a cursor member movable over a table and including switches operable to designate types of coordinate signals being generated.

12. The system of claim 9 and further comprising means for calculating the efficiency of the utilizatiton of the marker area by a particular arrangement of pattern pieces on said display screen within the marker area, and means for displaying the efficiency of a finished marker layout on said display screen.

13. The system of claim 9 and further comprising means for terminating display of pattern pieces and marker area and for displaying a nested plot of various graded sizes of a pattern piece on said display screen, said stylus operable to check the grading applied to said nested plot.

14. A system for laying out a plurality of pattern pieces in a pattern marker comprising:

a digitizer for generating pattern pieces coordinate data from said pattern pieces and grading instructions from said coordinate data;

means for storing said coordinate data and grading instructions;

means for selectively retrieving portions of said coordinate data and grading instructions corresponding to pattern pieces of a preselected style;

means for applying said grading instructions to said retrieved coordinate data to grade said pattern pieces to preselected sizes;

a display screen for displaying miniature representations of said graded pattern pieces and said marker area;

a data tablet;

stylus means movable relative to said data tablet;

circuitry responsive to movement of said stylus means relative to said data tablet for causing movement of said pattern pieces on said display screen into said displayed marker area; and

means for plotting a full sized marker corresponding to the final arrangement of pattern pieces within said marker area on said display screen.

15. The system of claim 14 and further comprising means for displaying said miniature pattern pieces on said display screen in an order of decreasing size.

16. The system of claim 14 and further comprising a function box having a plurality of keys, one of said keys operable to cause one of said displayed pattern pieces to be locked on to movement of said light cursor in order to move said pattern pieces into said displayed marker area.

17. The system of claim 16 wherein one of said keys is operable to cause selected pattern pieces to be abutted within said displayed marker area.

18. The system of claim 14 wherein said forming means comprises a plotter for plotting a full sized marker on a sheet of pattern paper.

19. The system of claim 14 and further comprising means for displaying a goal line on said marker area to designate a desired area to be filled with said pattern pieces.

20. The system of claim 14 and further comprising means for generating an alarm indication if the boundaries of any of the pattern pieces arranged in said displayed marker area by operation of said stylus means are overlapped with one another.

21. A system for grading and checking model pattern pieces comprising:

a digitizer for generating pattern piece coordinate data and grading instructions for said coordinate data;

means for storing said coordinate data and grading instructions;

means for applying said grading instructions to said

coordinate data according to predetermined grade rules;

display means for selectively displaying nested plots of a a plurality of graded sizes of each of said pattern pieces;

a data tablet interconnected with said displaying means and including a stylus;

means responsive to movement of said stylus adjacent said data tablet for displaying grading instructions on said display means corresponding to selected points on said displayed nested plots;

a function box having switches operable to revise said grading instructions displayed on said display means corresponding to said selected points on said displayed nested plots; and

means for regrading said selected points on said displayed nested plots according to said revised grading instructions.

22. The system of claim **21** and further comprising means for displaying on said display means a rule table selected from said function box.

23. The system of claim **21** and further including means for displaying a selected plurality of pattern pieces adjacent a marker area and for moving said pattern pieces into said marker area.

24. A method for arranging a plurality of apparel pattern pieces within a marker area to form a pattern marker comprising:

displaying on a screen miniature representations of said pattern pieces and of said marker area;

operating a positioning system to move said pattern pieces into said marker area on said screen and to arrange said pattern pieces within said marker area; and

automatically tracing out a full sized apparel pattern marker corresponding to the final arrangement on said screen of said pattern pieces in said marker area.

25. The method of claim **24** and further comprising:

operating buttons on a function box to cause selected ones of said displayed pattern pieces to be abutted together within said marker area.

26. A method of preparing an apparel pattern marker comprising:

generating digital coordinate data representative of the coordinates of points defining a plurality of model size pattern pieces;

storing said coordinate data;

grading said stored coordinate data according to predetermined grading rules;

displaying on a screen miniature representations of a predetermined size and style of pattern pieces and a miniature representation of a marker area;

moving a cursor on said screen for picking selected pattern pieces displayed on said screen and for moving said pattern pieces into said displayed marker area;

operating function buttons for arranging and orienting said displayed pattern pieces within said displayed marker area; and

plotting a full size marker corresponding to the final arrangement of pattern pieces within said marker area on said screen.

27. The method of claim **26** and further comprising operating ones of said function buttons for causing pattern pieces to be moved into abutment with one another within said displayed marker area.

28. The method of claim **26** and further comprising:

operating ones of said function buttons for causing rotation of selected ones of said displayed pattern pieces within said displayed marker area.

29. The method of claim **26** and further comprising generating an alarm indication when pattern pieces overlap one another within said marker area.

30. A system for preparing an apparel pattern marker wherein previous markers have been prepared and stored comprising:

a display screen;

means for displaying on said screen a miniature representation of a marker area with a previous arrangement of pattern pieces of a first style formed within said marker area;

means for replacing on said screen said previous arrangement of pattern pieces with pattern pieces of a second different style;

positioning means for selectively moving said displayed pattern pieces of said second style on said display screen within said marker area to form a miniature representation of a marker including pattern pieces of said second style; and

means for tracing the outline of a full sized marker corresponding to the final arrangement of pattern pieces within said marker area on said display screen.

* * * * *