

80-13

80-13

A Description of the Display Module
for
Interactive Presentation of Visual Information
on a Plasma Terminal
(ORION)

Albert Boulanger

Internal Report

No. 80-13

Intelligent Systems Group
Department of Computer Science
University of Illinois
Urbana Illinois

This research is supported in part by the US Department of Agriculture grant No. 901-15-44 and in part by the National Science Foundation grant No. MCS 79-06614.

```
000003 215 (*L'DISPLAY MODULE ROUTINES*)
000003 216 (*****
000003 217 X*****
000003 218 X*****
000003 219 X*****
000003 220 X*****
000003 221 X*****
000003 222 X*****
000003 223 X*****
000003 224 X DDDDDDD IIII SSSSS PPPPP LL AAAA YY YY X
000003 225 X DDDDDDD IIII SSSSS PPPPP LL AAAA YY YY X
000003 226 X DD DD II SS SSSSS PPP PP LL AA AA YY YY X
000003 227 X DD DD II SS SSSSS PPP PP LL AA AA YY YY X
000003 228 X DD DD II SS SSSSS PPP PP LL AA AA YY YY X
000003 229 X DD DD II SS SSSSS PPPPP LL AAAAAA YY X
000003 230 X DD DD II SS SSSSS PPPPP LL AAAAAA YY X
000003 231 X DD DD II SS SSSSS PPPPP LL AAAAAA YY X
000003 232 X DD DD II SS SSSSS PPP PP LL AA AA YY X
000003 233 X DD DD II SS SSSSS PPP PP LL AA AA YY X
000003 234 X DD DD II SS SSSSS PPP PP LL AA AA YY X
000003 235 X DD DD II SS SSSSS PPP PP LL AA AA YY X
000003 236 X DDDDDDD IIII SSSSS PP LLLLLL AA AA YY X
000003 237 X DDDDDDD IIII SSSSS PP LLLLLL AA AA YY X
000003 238 X*****
000003 239 X*****
000003 240 X*****
000003 241 X*****
000003 242 X M M CCCCC DDDDD DD UU LL FFFFFFFF X
000003 243 X M M CCCCC DDDDD DD UU LL FFFFFFFF X
000003 244 X M M CCCCC DDD DD UU UU LL FE X
000003 245 X M M CCCCC DDD DD UU UU LL FE X
000003 246 X M M CCCCC DDD DD UU UU LL FE X
000003 247 X M M CCCCC DDD DD UU UU LL FE X
000003 248 X M M CCCCC DDD DD UU UU LL FFFF X
000003 249 X M M CCCCC DDD DD UU UU LL FFFF X
000003 250 X M M CCCCC DDD DD UU UU LL FE X
000003 251 X M M CCCCC DDD DD UU UU LL FE X
000003 252 X M M CCCCC DDD DD UU UU LL FE X
000003 253 X M M CCCCC DDD DD UU UU LL FE X
000003 254 X M M CCCCC DDDDD UUUU LLLLLL FFFFFFFF X
000003 255 X M M CCCCC DDDDD UUUU LLLLLL FFFFFFFF X
000003 256 X*****
000003 257 X*****
000003 258 X*****
000003 259 X*****
000003 260 X*****
000003 261 X*****
000003 262 X*****
000003 263 X*****
000003 264 X*****
000003 265 X*****
000003 266 X*****
000003 267 X*****
000003 268 X*****
000003 269 X*****
000003 270 X*****
000003 271 X*****
```

----- I N T R O D U C T I O N -----

THIS SET OF PASCAL FUNCTIONS AND PROCEDURES ARE THE NECESSARY
COMPONENTS OF THE DISPLAY MODULE. THIS SET OF ROUTINES WAS

```

000003 272 WRITTEN BY ALBERT BOULANGER DURING THE SUMMER AND FALL OF
000003 273 1980. THESE ROUTINES ARE TARGETED FOR THE MAGNAVOX ORION-60
000003 274 PLASMA DISPLAY TERMINAL. EVENTUALLY THESE ROUTINES SHOULD
000003 275 SUPPORT SEVERAL TYPES OF CRT DISPLAY THAT HAVE AT LEAST
000003 276 CURSOR ADDRESSABILITY. A GOOD EXAMPLE IS THE TPKTRONIX 4012.
000003 277 THESE ROUTINES WERE WRITTEN TO BE USED IN THE CO-OPERATIVE
000003 278 KNOWLEDGE BASE EFFORT.
000003 279
000003 280 ----- D E S I G N -----
000003 281
000003 282
000003 283
000003 284
000003 285
000003 286
000003 287
000003 288
000003 289
000003 290
000003 291 THE DISPLAY MODULE WAS DESIGNED TO PROVIDE A HUMAN ENGINEERED
000003 292 INTERFACE TO A KNOWLEDGE BASE SYSTEM. IN THE DISPLAY MODULE PACKAGE,
000003 293 THE MAJOR MEANS FOR A USER TO INTERACT WITH THE COMPUTER IS VIA THE
000003 294 TOUCH PANEL. THE MAJORITY OF GRAPHICS IN A KNOWLEDGE BASE SYSTEM IS
000003 295 TEXTUALLY RELATED. (I.E. DISPLAY AN INFERENCE TREE, OR A TABLE, ETC.)
000003 296 THEREFORE, THIS SYSTEM DOES SUPPORT GRAPHICS, BUT IT IS TEXTUALLY
000003 297 RELATED. THERE ARE NO CLIPPING OR WINDOWING OR TRANSFORMATION
000003 298 FACILITIES IN THE SYSTEM. IT WAS FELT THAT THERE WAS A NEED TO DISPLAY
000003 299 SEVERAL QUESTIONS OR SEVERAL RELATED TOPICS SIMULTANEOUSLY ON THE SCREEN
000003 300 TO MAXIMIZE USER RESPONSE BANDWIDTH. THE DISPLAY SCREEN CONTEXT IN WHICH
000003 301 SEVERAL QUESTIONS OR SEVERAL DIFFERENT TOPICS CAN BE PRESENTED IS A SET
000003 302 OF BLOCKS. A BLOCK IS A VIRTUAL TERMINAL. IDEALLY, A VIRTUAL TERMINAL
000003 303 BEHAVES LIKE A REAL TERMINAL. BLOCKS CAN BE NESTED. THE DISPLAY SCREEN
000003 304 DEVICE WHICH ALLOWS A USER TO RESPOND VIA THE TOUCH PANEL IS THE TOUCH
000003 305 TARGET. (HERE ON ABBREVIATED AS TT.) A TT IS AN AREA OF THE SCREEN
000003 306 THAT HAS ONE OR MORE TOUCH PANEL INTERSECTIONS IN IT. FUNCTIONALLY, A
000003 307 TT TELLS THE USER WHERE THE SENSITIVE AREA FOR A CERTAIN RESPONSE IS.
000003 308 WHEN TOUCHED, THE TT CHANGES ITS VISUAL CHARACTERISTICS TO GIVE THE USER
000003 309 FEEDBACK.
000003 310
000003 311
000003 312 IN ORDER TO COMPLETE THE CONCEPT OF A VIRTUAL TERMINAL, MOST OF THE
000003 313 FUNCTIONS (EXCEPT BLOCK MAINTENANCE AND TT INPUT) ARE ACCESSED THROUGH A
000003 314 BLOCK DISPLAY PROCEDURE. THIS PROCEDURE IS PARAMETERISED IN PART BY A
000003 315 TEXT FILE TO DISPLAY AND A BLOCK IDENTIFIER. TTs ARE PLACED RELATIVE TO
000003 316 DISPLAYED TEXT. WITHIN THE TEXT FILE, THERE ARE CONTROL COMMANDS SUCH
000003 317 AS GRAPHIC ACTION COMMANDS (SUCH AS DRAW A POINT OR A LINE) THAT ARE
000003 318 IDENTIFIED BY AN ESCAPE CHARACTER AND A TWO LETTER COMMAND IDENTIFIER.
000003 319
000003 320
000003 321
000003 322
000003 323 IN ORDER TO MAINTAIN AS MUCH FLEXIBILITY AS POSSIBLE, A HYBRID DESIGN
000003 324 APPROACH WAS ADOPTED. FIRST, WITH A BOTTOM-UP DESIGN, THE BASIC
000003 325 ROUTINES THAT EXPRESS THE CAPABILITY OF THE ORION TERMINAL WERE WRITTEN.
000003 326 WITH THESE PROCEDURES WRITTEN, IT WAS THEN POSSIBLE TO USE THEM AS TOOLS
000003 327 TO TEST HIGH-LEVEL, USER-ORIENTED CONCEPTS. ONCE THE HIGH-LEVEL USER
000003 328 INTERFACE WAS CRYSTALIZED, A SWITCH TO TOP-DOWN DESIGN WAS APPECTED.
  
```

```

000003 329 THIS APPROACH HAD THE ADVANTAGE OF KEEPING THE FLEXIBILITY OF THE LOWER
000003 330 LEVEL PROCEDURES FROM CONVERGING TO QUICKLY TO ONE USER ORIENTED
000003 331 INTERFACE. THUS, ONE COULD STRIP OFF THE TOP LEVEL INTERFACE FROM THE
000003 332 DISPLAY MODULE AND STILL HAVE A USEFUL SET OF TOOLS. ONE COULD, FOR
000003 333 INSTANCE, WRITE A TEKTRONIX EMULATOR USING THE LOWEST LEVEL PROCEDURES.
000003 334 AS A RESULT OF THIS, YOU WILL FIND SOME OF THE OPTIONS TO SOME OF THE
000003 335 LOWER LEVEL PROCEDURES NEVER USED IN THE TOP LEVEL DESIGN.
000003 336
000003 337 HIGH LEVEL OPERATION
000003 338
000003 339 THE USER IS PRESENTED WITH 4 TYPES OF PROCEDURES. ONE SET IS FOR
000003 340 BLOCK MAINTENANCE, ONE SET IS FOR TOUCH PANEL INPUT, ONE PROCEDURE IS
000003 341 FOR TEXT DISPLAY, AND THE LAST PROCEDURE IS USED TO INITIALIZE THE
000003 342 SYSTEM.
000003 343
000003 344
000003 345 THE FRAMEWORK FOR ANYTHING DISPLAYED ON THE SCREEN ARE ALL CURRENTLY
000003 346 CREATED BLOCKS. THEREFORE, IT IS NECESSARY TO CREATE A BLOCK BEFORE
000003 347 WRITING ANYTHING ON THE SCREEN. HERE IS AN EXAMPLE OF A BLOCK CREATION
000003 348 CALL:
000003 349
000003 350 CREATEBLOCK(ID,0,512,0,512,DOTS,DOTS,LL,5,NOSCROLL
000003 351 WRAP,STANDARD,ALTERNATE,ERROR);
000003 352
000003 353 THIS CREATES A BLOCK THAT COVERS THE WHOLE SCREEN, HAS A BORDER
000003 354 THICKNESS OF 5 DOTS, END OF PAGE CONDITION CAUSES THE SCREEN TO CLEAR
000003 355 AND THEN THE NEW TEXT WRITTEN, END OF LINE CONDITION CAUSES WRAPPING OF
000003 356 THE TEXT. THE DEFAULT CHARACTER SET IS STANDARD. THE ALTERNATE
000003 357 CHARACTER SET IS THE PROGRAMMABLE ONE. WHEN CALLED, CREATEBLOCK
000003 358 PROVIDES AN ID FOR YOU WHICH YOU MUST PROVIDE TO THE OTHER ROUTINES TO
000003 359 REFER TO THE BLOCK JUST CREATED. ERROR IS SET TO ANY OF THE POSSIBLE
000003 360 BLOCK CREATION ERRORS.
000003 361
000003 362
000003 363 NOW, LET'S SAY THAT YOU WANT TO DISPLAY A QUESTION WHICH THE USER IS
000003 364 EXPECTED TO ANSWER USING THE TOUCH PANEL. YOU FIRST WRITE THE TEXT OF
000003 365 THE QUESTION TO A TEXT FILE FOLLOWED BY THE SET OF TT ESCAPE SEQUENCES,
000003 366 ONE FOR EACH OF THE POSSIBLE ANSWERS. FOR INSTANCE, LET'S SAY YOU WANT
000003 367 TO ASK THE QUESTION: "HOW ARE YOU TODAY?", AND THE TWO POSSIBLE ANSWERS
000003 368 ARE: "HORRIBLE!" AND "FINE."
000003 369 ONE WOULD CREATE A TEXT FILE WITH THE FOLLOWING TEXT IN IT:
000003 370
000003 371 HOW ARE YOU TODAY?\EL\TT1HORRIBLE!\TEXT2FINE.\TE\EL
000003 372
000003 373 THE \EL SEQUENCE REPRESENTS THE END OF LINE DELIMITER. THE \TT SEQUENCE
000003 374 IS THE TOUCH TARGET COMMAND. THE NUMBER FOLLOWING THE TT IS A TT
000003 375 IDENTIFIER WHICH IS USED TO IDENTIFY WHAT TT WAS TOUCHED. LET'S SAY
000003 376 THAT THE TEXT IS IN THE TEXT FILE OUTTEXT. TO DISPLAY IT IN THE BLOCK
000003 377 CREATED ABOVE YOU WOULD DO THE FOLLOWING:
000003 378
000003 379 DSTEXT(ID,OUTTEXT,ERROR);
000003 380
000003 381 ONCE WE PRESENTED THE QUESTION(S), WE ARE READY TO RETRIEVE THE ANSWER.
000003 382 WE DO THE FOLLOWING:
000003 383
000003 384 GETTARGINP(BLOCKID,TARGID,CHARRAY,CHLEN,FALSE,ERROR);
000003 385

```

000003 386 IF THERE WAS A LEGITIMATE RESPONSE FROM THE USER, THEN THIS PROCEDURE
 000003 387 WILL COME BACK WITH A BLOCK IDENTIFIER IN WHICH A TARGET WAS TOUCHED AND
 000003 388 THE TARGET IDENTIFIER THAT WAS PROVIDED IN THE TTT COMMAND SEQUENCE.
 000003 389 IF ERROR WAS MISTOUCH, THE USER TOUCHED OUTSIDE THE SENSITIVE TARGET
 000003 390 AREAS. IF ERROR WAS BADTOUCH, THE USER PROBABLY ATTEMPTED TO USE THE
 000003 391 KEYBOARD TO ANSWER THE QUESTION. THE BUFFER THAT WAS USED TO INPUT THE
 000003 392 TEXT IS PROVIDED SO THAT YOU CAN RETRIEVE THE KEYBOARD INPUT. THE PARA
 000003 393 WERE ENTERED.

000003 394
 000003 395
 000003 396 THERE IS A SET OF PROCEDURES TO CARRY ON WITH THE BLOCK AFTER IT HAS
 000003 397 BEEN USED. THESE ALL TAKE TWO ARGUMENTS, A BLOCK IDENTIFIER AND AN
 000003 398 ERROR RETURN. DSTBLOCK DESTROYS A BLOCK. CLEARBLOCK AND UNDOBLOCK ARE
 000003 399 WAYS OF READING A BLOCK FOR NEW STUFF. REDORBLOCK IS FOR WHEN YOU HAVE
 000003 400 TRANSMISSION ERRORS. DISARMBLOCK DEACTIVATES ALL TTS IN A BLOCK.
 000003 401 REARMBLOCK REARMS THE TTS IN A BLOCK AS WELL AS RESPTTING THEM VISUALLY.

000003 402
 000003 403
 000003 404 TO INITIALIZE THE SYSTEM, DO THE FOLLOWING:
 000003 405
 000003 406 INITDSPARRAYS(CHFILE);
 000003 407
 000003 408 WHERE CHFILE IS OF TYPE ALPA AND HAS AS A VALUE THE NAME OF THE
 000003 409 PROGRAMMALE CHARACTER SET FILE.

000003 410
 000003 411

000003 412 ----- P R O C E D U R E S -----

000003 413 -----

000003 414
 000003 415 U S E D

000003 416 -----

000003 417
 000003 418 PROG# PROCEDURE LENGTH PROCEDURES CALLED

000003 419 -----

000003 420					
000003 421	(1) DISPLAY	565	OPEN	CLOSE	PUTCH
000003 422			DSPCHARCNT	CNTCHARS	DISABLED
000003 423			CENTERIT	MODE	SETCOORD
000003 424			PUTCHAR	DRAWCHAR	LOADCHRS
000003 425			DRAWBOX	CREATETARG	DSTTARG
000003 426			GETTOUCHIN	HANDLTFCL	INITDSPARR
000003 427			CLEARBLOCK	UNDOBLOCK	REDORBLOCK
000003 428			DSTEXT	GETTARGINP	DISARMBLOC
000003 429					REARMBLOCK

000003 430
 000003 431 INTERNAL EXTERNAL

000003 432 (2) OPEN 8

000003 433 (3) CLOSE 6

000003 434 (4) PUTCH 34

000003 435 (5) CNVT 44

000003 436 (6) CNVTA 44

000003 437 (7) DSPCHARCNT 41

000003 438 (8) CNTCHARS 27

000003 439 (9) DISABLED 30

000003 440 (10) NEST 48

000003 441 (11) CONVMETRIC 43

000003 442 (12) CENTERIT 49

[WRITE,WRITELN]

[READ,ORD]

[READ,ORD]

[DSPCHARCNT]

```

000003 443 ( 13) MODE      117
000003 444 ( 14) SETUP    19
000003 445 ( 15) SETCOORD 42
000003 446 ( 16) DRAWLINE 48
000003 447 ( 17) DRAWPOINT 35
000003 448 ( 18) PUTCHAR   82
000003 449 ( 19) DRAWCHAR  37
000003 450 ( 20) LOADCHRS  73
000003 451 ( 21) GETLN     43
000003 452 ( 22) SETSET    205
000003 453 ( 23) DRAWBOX   68
000003 454 ( 23) DRAWBOX   68
000003 455 ( 24) CREATETARG 108
000003 456
000003 457
000003 458 ( 25) DSTARG     15
000003 459 ( 26) FETCH     28
000003 460 ( 27) DSPLINE   296
000003 461
000003 462
000003 463
000003 464 ( 28) GETTOUCHIN  11
000003 465
000003 466
000003 467 ( 29) HANDLEEOL   51
000003 468 ( 30) INITDSPARR 155
000003 469
000003 470
000003 471
000003 472 ( 31) CREATEBLOC 177
000003 473
000003 474 ( 32) STATUS      58
000003 475 ( 33) DSTBLOCK   70
000003 476
000003 477 ( 34) KILLBLOCK    52
000003 478 ( 35) CLEARBLOCK   11
000003 479 ( 36) UNDOBLOCK   36
000003 480 ( 37) REDOBLOCK    11
000003 481 ( 38) DISARMBLOC  25
000003 482 ( 39) REARMBLOCK   27
000003 483 ( 40) DSTEXT      122
000003 484
000003 485 ( 41) CNVT1        43
000003 486 ( 42) PUTICH      44
000003 487 ( 43) INTERPRET  123
000003 488
000003 489
000003 490 ( 44) GETTARGINP   84
000003 491
000003 492
000003 493
000003 494 ----- R E F E R E N C E S -----
000003 495
000003 496
000003 497 A DESCRIPTION OF AN EDITOR USING THE CRICH-60 WITH
000003 498 A TOUCH PANEL IS FOUND IN MITCHELL LUBARS'S M.S. THESIS:
000003 499 AN EDITOR/GENERATOR SYSTEM FOR CREATION OF
000003 500 VARIABLE PHRASE KEYBOARDS. 1980.
  
```

SETUP

STATUS

KILLBLOCK

CNVT1 PUTICH INTERPRET

```

[ PUTCH ]
[ PUTCH ]
[ PUTCH,CONVMETRIC,MODE ]
[ PUTCH,CONVMETRIC,MODE ]
[ PUTCH,CONVMETRIC,MODE ]
[ PUTCH,MODE ]
[ CONVMETRIC,MODE,PUTCHAR,CENTERIT ]
[ PUTCH ]
[ READ,EOP,EOLN ]
[ GETLN,READLN ]
[ CONVMETRIC,CENTERIT,DRAWPOINT,
DRAWLINE,SETCOORD,ABS ]
[ CONVMETRIC,CENTERIT,CNTCHARS,
DRAWBOX,SETCOORD,MODE,NEW,
WRITE ]
[ ]

[ SETCOORD,FETCH,MODE,
PUTCH,DSPLINE,DRAWLINE,
DRAWPOINT,DRAWCHAR,PUTCHAR,
CREATETARG,DSTARG,WRITE ]
[ SETCOORD,GETLN,MODE,
PUTCH,CNVT1,DRAWCHAR,
WRITELN,GETSET,EOS ]
[ GETTOUCHINP,DSPLINE,DISPOSE ]
[ LINLIMIT,OPEN,CLOSE,
GETSET,SETCOORD,MODE,
LOADCHRS,WRITE,NEW,
PUTCH,RESET ]
[ CONVMETRIC,CENTERIT,DRAWBOX,
SETCOORD,NEW ]
[ NEST ]
[ MODE,PUTCH,SETCOORD,
DRAWLINE,DISABLED ]
[ KILLBLOCK,DISPOSE ]
[ ]
[ DSPLINE,DISPOSE ]
[ ]

[ CNVT,READ,READLN,
EOLN,EOP,DISPOSE ]
[ READ,CNVT ]
[ NEW ]
[ PUTICH,INTERPRET,HANDLEEOL,
DSPLINE,NEW,READ,
REWRITE,WRITELN,RESET,EOLN ]
[ GETTOUCHINP,DISABLED,DRAWBOX,
SETCOORD,MODE,WRITE ]
  
```

000003 500
000003 501 A DESCRIPTION OF A TERMINAL DRIVER SYSTEM THAT HANDELED
000003 502 ORION-60 TERMINALS IS FOUND IN PLINT PELLETT'S M.S. THESIS:
000003 503 TERMINAL MANAGEMENT FOR A USER-ORIENTED SYSTEM.
000003 504 1979.
000003 505
000003 506 THE CENTER FOR ADVANCED COMPUTATION DOCUMENT THAT
000003 507 DESCRIBES THE INTELLIGENT TERMINAL SOFTWARE WHICH WAS
000003 508 A LARGE DISPLAY PACKAGE WRITTEN IN C USING A PLASMA
000003 509 TERMINAL WITH TOUCH PANEL IS:
000003 510
000003 511 INTELLIGENT TERMINAL PROGRAMMER'S MANUAL
000003 512 BY
000003 513 BROWN, KOPETZKY, MULLEN, & WILLCOX
000003 514 IN 2 VOLUMES, CAC DOCUMENT # 236. OCT 31 1977.
000003 515 THIS REFERENCE IS WHERE I GOT THE TERM "TOUCH TARGET".
000003 516
000003 517 THE MANUALS THAT DESCRIBE THE ORION-60 & THE TOUCH PANEL
000003 518 FOR THE ORION-60:
000003 519
000003 520 ORION-60 PLASMA DISPLAY TERMINAL
000003 521 INSTALLATION & OPERATION MANUAL.
000003 522
000003 523
000003 524
000003 525 TOUCH INPUT SYSTEM OPTION USERS MANUAL
000003 526 FOR
000003 527 MODEL 12,000 PLASMA DISPLAY TERMINAL (MODEL 27)
000003 528
000003 529 ORION-60 PLASMA DISPLAY TERMINAL (MODEL 28).
000003 530
000003 531 ALSO, IF THE ORION IS SICK, REFER TO:
000003 532
000003 533 ORION-60 PLASMA DISPLAY TERMINAL
000003 534 MAINTENANCE DOCUMENTATION.
000003 535
000003 536 THE ADDRESS FOR MAGNAVOX IS:
000003 537
000003 538 MAGNAVOX GOVERNMENT & INDUSTRIAL ELECTRONICS CO.
000003 539 1313 PRODUCTION ROAD
000003 540 FORT WANE, INDIANA 46808
000003 541 219 482-4411
000003 542
000003 543 THE ABOVE THESES AND CAC DOCUMENTS ARE IN THE
000003 544 DCL LIBRARY. THE ORION DOCUMENTS SHOULD BE EITHER
000003 545 WITH ME OR WITH THE ORION. A.B. BASKIN ALSO HAS
000003 546 COPIES OF SOME OF THE ORION DOCUMENTS.
000003 547
000003 548
000003 549 THERE IS A SET OF SCHEMATICS ON THE CAIROIL TOUCH
000003 550 PANEL THAT SHOULD BE ASSOCIATED WITH THE ORION-60
000003 551 MAINTENANCE DOCUMENT. THIS PANEL IS A 32X32 THAT BELONGED
000003 552 TO THE INTELLIGENT TERMINAL SYSTEM (THE LSI-11 IN THE QUEBEC ROOM)
000003 553 I KLUDGED IT SO IT WOULD WORK WITH THE ORION. THERE IS
000003 554 A SWITCH-YARD BOARD LOCATED IN ONE OF THE ACCESSORY CARD
000003 555 SLOTS THAT CONDUCTS THE KLUDGE. I DROP THE LSB.
000003 556 -----

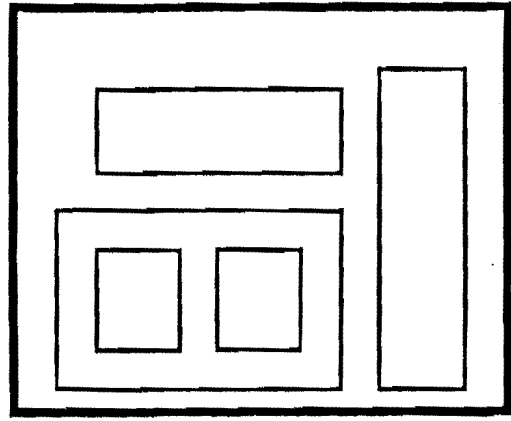
PASCAL COMPTLER - E.T.H. ZURICH / UNIVERSITY OF MINNESOTA.
DISPLAY MODULE
DISPLAY MODULE ROUTINES

000003 557 *)
000003 558
000003 559
000003 560
000003 561

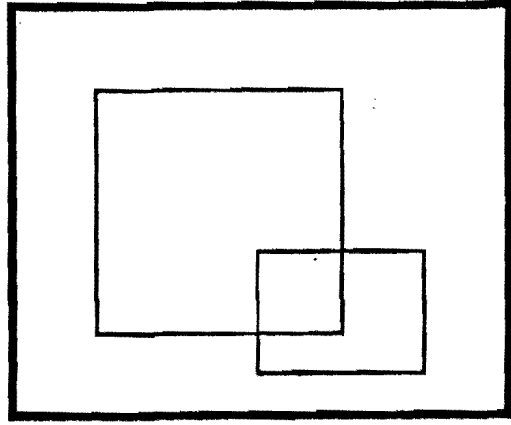
PASCAL 6000 V3.0.0. 80/11/17. 00.56.01
NOS 1.4 (80/04/21) PAGE 16

Proper Nesting of Blocks

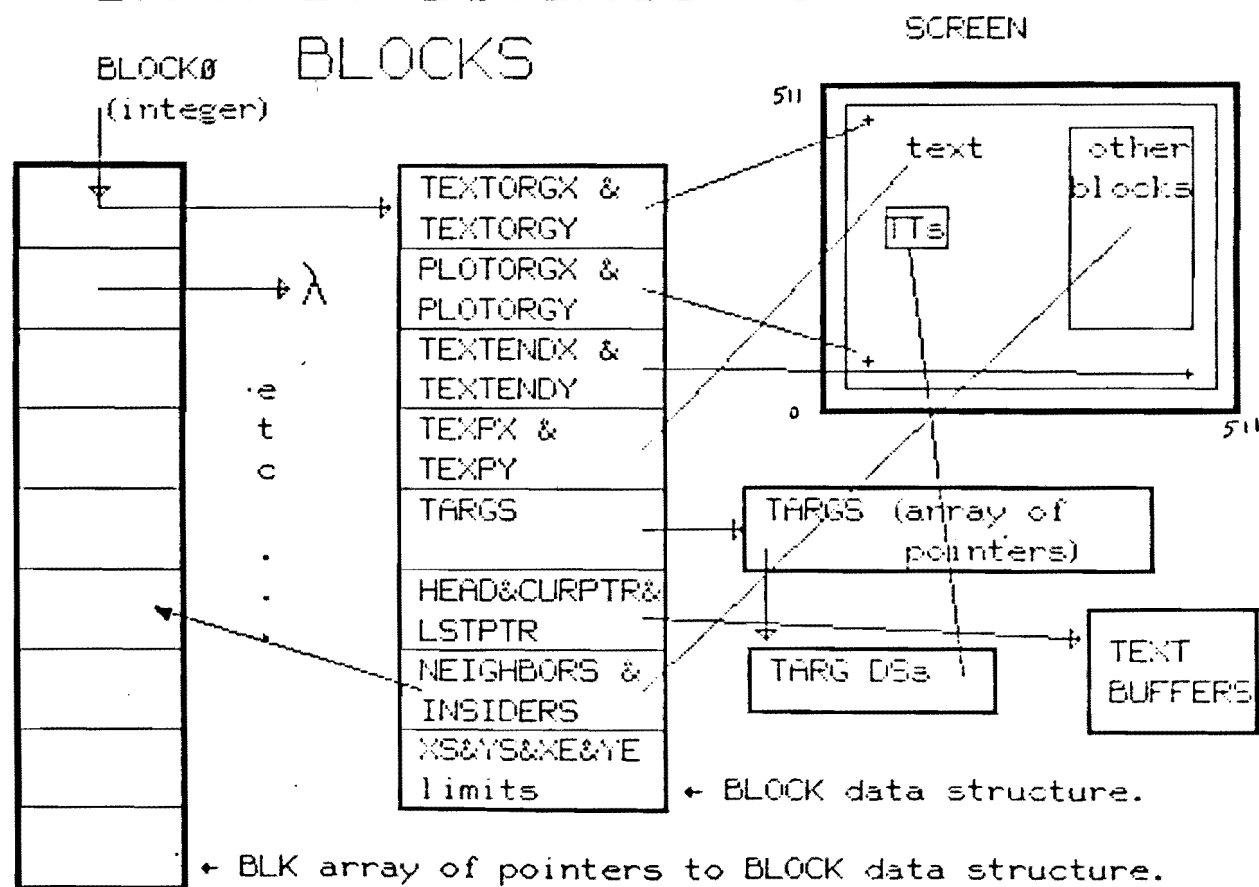
Blocks can be nested but not overlapped



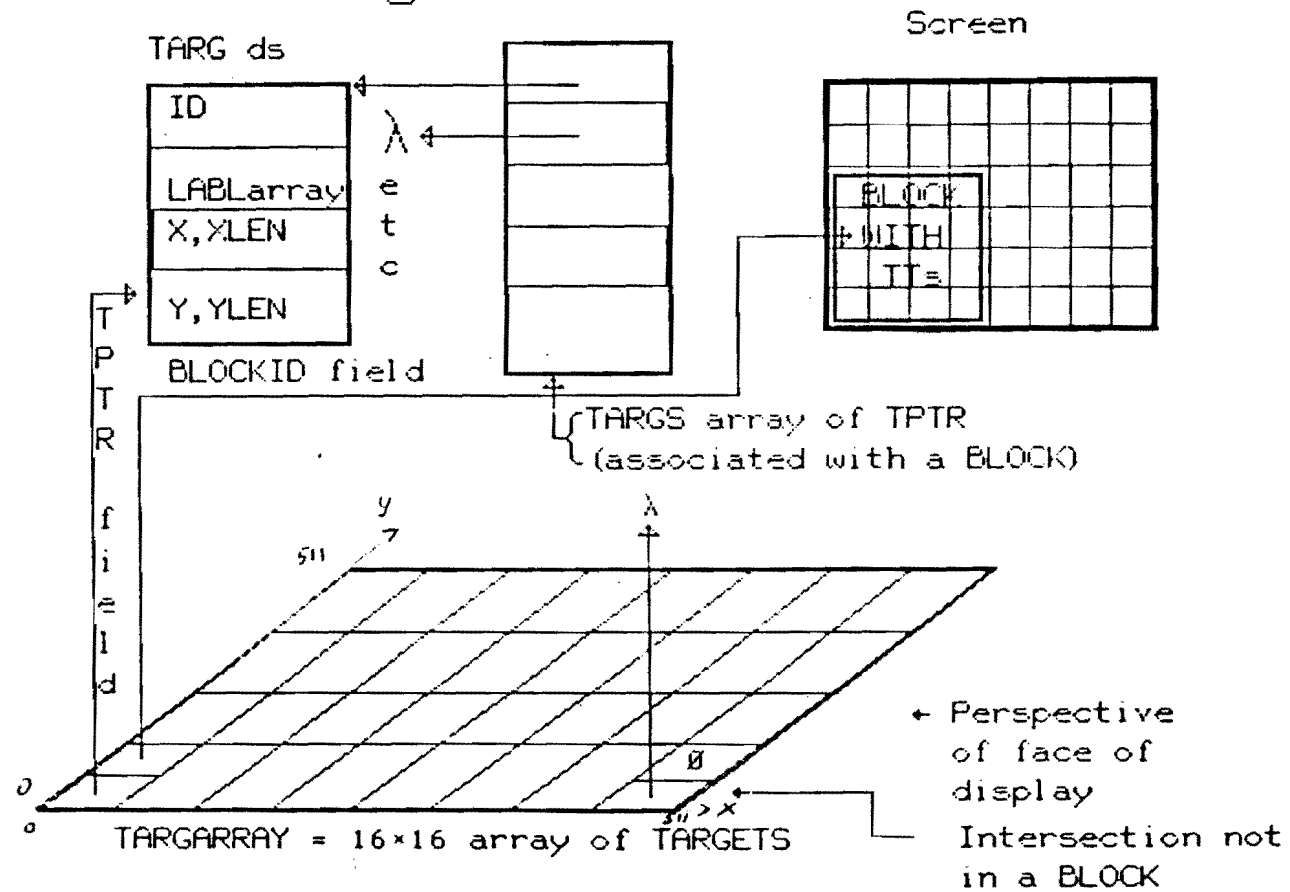
BUT
NOT



DATA STRUCTURES FOR BLOCKS

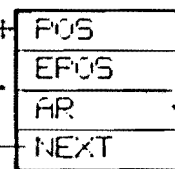
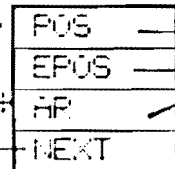
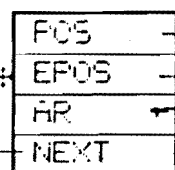
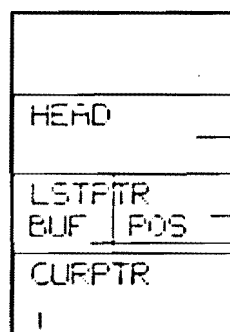


Data Structures for Targets



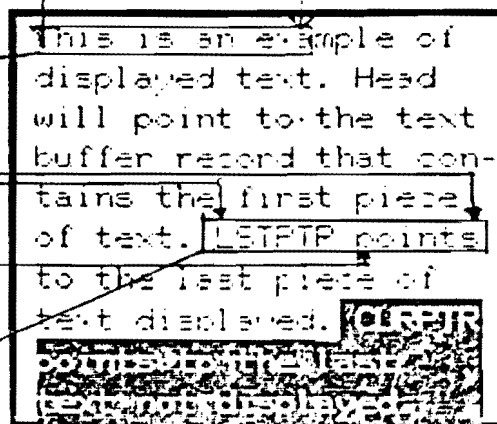
Data Structures for Text

BLOCK ds



DISPLAY BUF ds

SCREEN



Points to last piece of displayed text.

x4

000004 2 (*SL'DATA TYPES FOR DISPLAY MODULE' *)

000004 3 (*SI'GLOBAL'/'KBLTB' *)

----- RTGIN INCLUDED TEXT.

000004 3 PROGRAM KBMAIN(KB,INPUT/+,OUTPUT,MSG);

000074 3 (* GLOBAL CONSTANTS *)

000074 3 CONST

000074 3 CONMAXPRINTNAME = 256; (*MAX LENGTH OF PRINTNAME*)

000074 3 CONMAXTUPLELENGTH = 256; (* MAX TUPLE LENGTH *)

000074 3 (* DISPLAY MODULE CONSTANTS *)
000074 3 BUFLN = 132; (* LENGTH OF INPUT BUFFERS USED IN DISPLAY MODULE *)
000074 3 ESCAPE = '\'; (* DISPLAY MODULE ESCAPE CHARACTER *)

000074 3 (* GLOBAL TYPE DEFINITIONS *)

000074 3 TYPE
000074 3 DUMMY = INTEGER; (* DUMMY STATIC AREA - DO NOT USE *)

000074 3 INTNAMEPTR = ^INTNAMEREC; (*POINTERS TO INTERNAL NAME RECORDS*)

000074 3 INTNAMEFLAG = (NODEFLAG, INTFLAG, REALFLAG);
000074 3 (*FLAG IN INTERNAL NAME INDICATING

000074 3 WHETHER NAME IS AN ACTUAL NODE OR AN INTEGER OR REAL VALUE*)

000074 3
000074 3 (*****
000074 3000074 3 THE FOLLOWING IS THE STRUCTURE OF THE INTERNAL NAME RECORD
000074 3 TYPE. THIS STRUCTURE IS INVISIBLE TO THE USER, HOWEVER. THE
000074 3 USER DEALS ONLY WITH THESE RECORDS AS A WHOLE, AND ONLY THE
000074 3 FOLLOWING SUBROUTINE PRIMITIVES ACCESS INDIVIDUAL FIELDS.
000074 3 USER OPERATIONS UPON INTERNAL NAMES ARE RESTRICTED TO ASSIGNMENTS
000074 3 OF THE TYPE A:=B, THE USER SUPPLIED FUNCTION EQ TO TEST FOR
000074 3 EQUIVALENCE (E.G. IF (EQ(A)=TRUE THEN ...), AND THE PROCEDURES
000074 3 INTERINT AND INTERREAL FOR CONVERTING BACK AND FORTH FROM
000074 3 INTEGER AND REAL NUMBERS TO INTERNAL NAMES.000074 3 *****
000074 3 INTNAMEREC = RECORD (*AN INTERNAL NAME CONSISTS OF A FLAG*)

000074 3 CASE NP: INTNAMEFLAG OF (*AND A MODIFIER DEPENDANT UPON THE FLAG*)

000074 3 NODEFLAG: (NODE : INTEGER); (*ACTUAL NODE*)

000074 3 INTFLAG: (IVALUE : INTEGER); (*INTEGER VALUE*)

000074 3 REALFLAG: (RVALUE : REAL); (*REAL VALUE*)

000074 3 END;

000074 3
000074 3 LEVELTYPE = (PRIVATELEV, (*PRIVATE LEVEL ACCESS TO NODE*)

000074 3 GROUPLV, (*GROUP LEVEL *)

000074 3 GLOBALLEV); (*GLOBAL LEVEL *)

000074 3

```

000074 3 PRINTNAME = PACKED ARRAY [ 1 .. CONMAXPRINTNAME ] OF
000074 3 CHAR; (*CHAR STRING WHERE PRINTNAME IS STORED*)
000074 2 NODEACCESSTYPE = (RDCNLY, RDWRT);
000074 3 (*NODE ACCESS IS READ ONLY OR READ/WRITE*)
000074 3 TUPLE = ARRAY [ 1 .. CONMAXTUPLELENGTH ] OF
000074 3 INTNAMEREC; (*ARRAY TYPE WHERE INTERNAL NAME POINTERS OF TUPLE ARE STORED*)
000074 3 TUPLELENGTH = 0 .. CONMAXTUPLELENGTH;
000074 3 (*A NUMBER TO INDICATED HOW MANY MEMBERS IN A TUPLE*)
000074 3 PRINTNAMELENGTH = 0 .. CONMAXPRINTNAME; (*NUMBER OF CHARS IN PRINTNAME
000074 3 NOT COUNTING THE FOL*)
000074 3
000074 3 ERRORTYPE = (NOERROS, (*NO ERROR OCCURRED*)
000074 3 ERRMAXNODES, (*MAXIMUM NUMBER OF NODES USED*)
000074 3 ERRMAINNTFND, (*MAIN NODE WAS NOT FOUND*)
000074 3 ERRCNTNTOO, (*USAGE OR ATTR. COUNT OF NODE WAS NOT 0*)
000074 3 ERRMAXATTR, (*NODE HAS MAX NUMBER OF ATTRIBUTES*)
000074 3 ERRMAXUSAGE, (*NODE'S USAGE COUNT HAS REACHED MAX*)
000074 3 ERRREADCNLY, (*WRITE ACCESS REQUESTED TO RD ONLY*)
000074 3 ERRPRINTNAME, (*PRINTNAME NOT EVALUATABLE*)
000074 3 ERRTOASCII, (*ERROR IN CONVERTING NUMBER TO ASCII*)
000074 3 ERRPROTASCII, (*ERROR IN CONVERTING FROM ASCII *)
000074 3 ERRINTNODE, (*INTERNAL NAME WAS NOT A NODE*)
000074 3 ERRLOOKUPFAIL, (*PRINTNAME NOT IN DICTIONARY*)
000074 3 ERRDICTFUL, (*DICTIONARY FULL*)
000074 3 ERRTUPNTFND, (*TUPLE WAS NOT FOUND*)
000074 3 ERROPEOF, (*EOF ENCOUNTERED IN OPENNETWORK*)
000074 3 ERROPMSCH, (*MISSING CHARACTER*)
000074 3 ERROPCREA*E, (*OPENNETWORK ERROR FROM CREATENODE*)
000074 3 ERROES*TFI, (* " " " " SFTPRINTNAME*)
000074 3 ERROPNTIER, (* " " " " FNTERDICT*)
000074 3 ERROPLOCKUP, (* " " " " LOOKUP*)
000074 3 ERROPADDATT, (* " " " " ADDATTRIBUTE*)
000074 3 ERROPOPEN, (* " " " " OPENNODE*)
000074 3 ERROPCL*SE, (* " " " " CLOSPNODE*)
000074 3 ERRCLOGETPRI, (*CLOSFNETWORK " " GETPRINTNAME*)
000074 3 ERPCLOGETATT, (* " " " " GETATTRIBUTE*)
000074 3 ERRINTERRINT, (*INTERNAL NAME WAS NOT AN INTEGER*)
000074 3 ERRINTERREAL, (*INTERNAL NAME WAS NOT A REAL*)
000074 3 ERPRSYNTAX, (*PARSER FOUND A SYNTAX ERROR*)
000074 3 ERPROVERFLOW, (*PARSER STACK OVERFLOW*)
000074 3 ERPRNOPARTAB, (*PARSER TABLES FILE MISSING*)
000074 3 ERSPHINTERNAL, (*PARSER INTERNAL ERROR *)
000074 3 (*--- DISPLAY MODULE ERRORS ---*)
000074 3 BLOCKNOTNESTED, (*DISPLAY BLOCK NOT NESTED RIGHT *)
000074 3 TOOMANYBLOCKS, (*TRIED TO MAKE MORE THAN BLOCKMAX BLOCKS *)
000074 3 BLOCKOFFSCREEN, (*TRIED TO CREATE A BLOCK THAT GOES OFF THE SCREEN *)
000074 3 NOSUCHBLOCK, (*THE BLOCK WAS NEVER CREATED *)
000074 3 OUTOFBOUNDS, (*TRIED TO PUT SOMETHING OFF THE SCREEN *)
000074 3 NOMEMORY, (*RAN OUT OF INTERNAL MEMORY IN DSPTXT *)
000074 3 COMMANDERROR, (*COMMAND SYNTAX ERROR IN INPUT FILE TO DSPTXT *)
000074 3 MISTOUCH, (*AN ACTIVE TOUCH TARGET WAS NOT TOUCHED *)
000074 3 BADTOUCH, (*THE USPR PROBABLY TYPED IN A RESPONSE TO TT REQUEST *)
000074 3 LABELTOOLONG, (*A TOUCH TARGET LABEL WAS TOO LONG *)
000074 3
000074 3 );

```

```

000074      3      (* FASTER TUPLE MARKERS *)
000074      3      MARKTYPE = (MKNULL,MKDOMAIN,MKNCMINAL,MKINTERVAL,MKSTRUCTURE,
000074      3      MKUNITS,MKREFINE,MKTRUE,MKFALSE,MKREF0,MKREF1,MKREF2,
000074      3      MKVAR,MKFUNC1,MKFUNC2,MKEQ,MKGT,MKLT,MKNE,MKGE,MKLE,
000074      3      MKALL,MKUNKN,MKUNDEF,MKNA,MKBEHAVIOR,MKINCR,MKDECR,
000074      3      MKRMAX,MKRMIN,MKARGS,MKRULES,MKVARs,MKVARDECL,MKROLE,
000074      3      MKPAREN,MKCS,MKAND1,MKAND2,MKOR1,MKOR2,MKEXCPT,MKEQUIV,
000074      3      MKIMP,MKLM);
000074      3
000074      3      (*----- TYPES FOR DISPLAY MODULE -----*)
000074      3      METRIC = (DOTS,CHS,PARTS); (* METRIC FOR SCREEN COORDS FOR DISPLAY MODULE *)
000074      3      INBUF = PACKED ARRAY[1..BUFLN] OF CHAR; (* INPUT BUFFER USED IN DISPLAY MODULE *)
000074      3      SCROLLTYPE = (SCROLL,NOSCROLL); (*TYPE SPECIFICATION FOR SCROLLING FOR DISPLAY MODULE *)
000074      3      WRAPTYPE = (WRAP,TRUNCATE); (*EOL SPECIFICATION FOR DISPLAY MODULE*)
000074      3      CHSETTYPE = (STANDARD,ALTERNATE); (*CHARACTER SET SPECIFICATION FOR DISPLAY MODULE *)
000074      3      ADJ = (CENTER,IL,UL,LR,UR); (* ORIGIN SPECIFICATION FOR DISPLAY MODULE *)
000074      3
000074      3

```

----- END INCLUDED TEXT.

```

000074      4      CONST
000074      5      STSPEC = 128;
000074      6      ESCCH = 92;      LN = 129;      PT = 130;
000074      7      CH = 131;      CA = 132;      SC = 133;
000074      8      ML = 134;      MR = 135;      MU = 136;
000074      9      MB = 137;      PL = 138;      OV = 139;
000074     10      OA = 140;      ON = 141;      UE = 142;
000074     11      NP = 143;      ND = 144;      RP = 145;
000074     12      RD = 146;      CR = 147;      LI = 148;
000074     13      AL = 149;      NR = 150;      TT = 151;
000074     14      TU = 152;      TE = 153;      GO = 154;
000074     15      SU = 155;      SA = 156;      UU = 157;
000074     16      UA = 158;
000074     17
000074     18      (* ASCII CONSTANTS *)
000074     19      NUL = 0;
000074     20      SOH = 1;
000074     21      STX = 2;
000074     22      ETX = 3;
000074     23      EOT = 4;
000074     24      ENQ = 5;
000074     25      ACK = 6;
000074     26      BEL = 7;
000074     27      BS = 8;
000074     28      HT = 9;
000074     29      LF = 10;
000074     30      VT = 11;
000074     31      FF = 12;
000074     32      DSPCR = 13;
000074     33      SO = 14;
000074     34      SI = 15;
000074     35      DLE = 16;
000074     36      DC1 = 17;
000074     37      DC2 = 18;
000074     38      DC3 = 19;
000074     39      DC4 = 20;

```

```

000074 40      NAK = 21;
000074 41      SYN = 22;
000074 42      ETB = 23;
000074 43      CAN = 24;
000074 44      EM = 25;
000074 45      SUB = 26;
000074 46      ESC = 27;
000074 47      DSPFS = 28;
000074 48      GS = 29;
000074 49      RS = 30;
000074 50      US = 31;
000074 51      DELETE = 127;
000074 52      BLANK = 32;
000074 53      (*PSEUDONAMES*)
000074 54      KEYLOCK = 2; (* STX *)
000074 55      ERASE = 3; (* ETX *)
000074 56      DSPNORM = 4; (* EOT *)
000074 57      OVER = 6; (* ACK *)
000074 58      INVERSE = 16; (* DLE *)
000074 59      BACK = 8; (* BS *)
000074 60      FORWARD = 9; (* HT *)
000074 61      UP = 11; (* VT *)
000074 62      HOME = 12; (* FF *)
000074 63      CLEAR = 19; (* EC3 *)
000074 64      SUBSCRIPT = 18; (* DC2 *)
000074 65      SUPERSCRIPT = 26; (* PS *)
000074 66      ALTCHAR = 14; (* SO *)
000074 67      NORCHAR = 15; (* SI *)
000074 68      TCH = 5; (* ENQ *)
000074 69      BHOME = 30; (* RS *)
000074 70      DSPYES = 1; (*FOR PAGING *)
000074 71      DSPNO = 0;
000074 72      (*BUFLN = 132;*) (* LENGTH OF TEXT BUFFERS *)
000074 73      LABELLEN = 64; (* LENGTH OF TARG LABELS *)
000074 74      TEXTBUFLN = 64; (* LENGTH OF DISPLAY TEXT BUFFERS *)
000074 75      TEXTBUFLN1 = 65; (* TEXTBUFLN + 1 *)
000074 76      TARGMAX = 20; (* MAX TARGS/BLOCK *)
000074 77      BLOCKMAX = 20; (* MAX # OF BLOCKS *)
000074 78      NCOMMANDS = 30; (* NUMBER OF DSPTEXT COMMANDS *)
000074 79      (*ESCAPE = '\';*) (* ESCAPE CHARACTER FOR DISPLAY ROUTINES *)
000074 80      TYPE
000074 81      CRNG = 0..999;
000074 82      NESTYPE = (NESTED, REVNESTED, NOIN*STFL, OVERLAP);
000074 83      (*ERRORTYPE = (
000074 84          BLOCKNOTNESTED,
000074 85          TOOMANYBLOCKS,
000074 86          BLOCKOFFSCREEN,
000074 87          NOSUCHBLOCK,
000074 88          OUTOFBOUNDS,
000074 89          NOMEMORY,
000074 90          COMMANDERROR,
000074 91          MISTOUCH,
000074 92          BADTOUCH,
000074 93          LABELTOOLONG
000074 94      ); *)
000074 95      LABELS = PACKED ARRAY[1..LABELLEN] OF CHAR;
000074 96      LABPTR = ^LABELS; (* A POINTER FOR TARGET LABELS *)

```



```

000074  97      LINENODE = (PRINT,UNDO,CONSUME); (* MODES OF DISPLAYING A LINE *)
000074  98      ARR = PACKED ARRAY[1..TEXTBUFLN] OF 0..511;
000074  99      DSPBUFPTR = ^DISPLAYBUF;
000074 100      DISPLAYBUF = RECORD
000074 101          AR : ARR; (* TEXT BUFFER *)
000074 102          POS : 0..TEXTBUFLN; (* BEGINNING OF GOOD TEXT IN BUFFER *)
000074 103          EPOS : 0..TEXTBUFLN1; (* END OF GOOD TEXT IN BUFFER *)
000074 104          NEXT : DSPBUFPTR; (* NEXT BUFFER *)
000074 105      END;
000074 106      TEXTPTR = RECORD (* 2 TUPLE PCINTER STRUCTURE FOR POINTING TO TEXT *)
000074 107          BUF : DSPBUFPTR;
000074 108          POS : 0..TEXTBUFLN1;
000074 109      END;
000074 110      TPTR = ^TARG;
000074 111      TARGETS = RECORD
000074 112          BLOCKID : 0..BLOCKMAX;
000074 113          TARG : TPTR; (* POINTER TO THE TARGET DS *)
000074 114      END;
000074 115      (*METRIC = (DOTS,CHS,PARTS);*) (* METRIC FOR SCREEN COORDS *)
000074 116      TARGTYPE = (*AT,UNDERLINE*);
000074 117      TARGPTR = ^TARGS;
000074 118      (*INBUF = PACKED APRAY [1..BUFLN] OF CHAR; *)
000074 119      PASBUF = PACKED ARRAY[INTEGER] OF CHAR;
000074 120      TARG = RECORD
000074 121          ID : INTEGER;
000074 122          STYLE : TARGTYPE;
000074 123          LABL : LABELS;
000074 124          LBLPN : INTEGER;
000074 125          X,XLEN : INTEGER;
000074 126          Y,YLEN : INTEGER;
000074 127          LOGX,LOGY : INTEGER;
000074 128          TOUCHED : BOOLEAN;
000074 129      END;
000074 130      TARGS = ARRAY[1..TARGMAX] OF TPTR;
000074 131      (*SCROLLTYPE = (SCROLL,NOSCROLL); *)
000074 132      (*WRAPTYPE = (WRAP,TRUNCATE); *)
000074 133      (*CHSETTYPE = (STANDARD,ALTERNATE); *)
000074 134      BLOCK = RECORD
000074 135          XS : INTEGER;
000074 136          XE : INTEGER;
000074 137          YS : INTEGER;
000074 138          YE : INTEGER;
000074 139          OUTLINE : INTEGER;
000074 140          OVERFLOW : SCROLLTYPE;
000074 141          OVERXPLOW : WRAPTYPE;
000074 142          CHSET : CHSETTYPE;
000074 143          ALTCHSPT : CHSETTYPE;
000074 144          TEXTORGX : INTEGER;
000074 145          TEXTORGY : INTEGER;
000074 146          TFXPX : INTEGER;
000074 147          TFXPY : INTEGER;
000074 148          PLOTORGX : INTEGER;
000074 149          PLOTORGY : INTEGER;
000074 150          WHODE : INTEGER; (* 3,4,6,16 (ERASE,DSPNCRN,OVER,INVERSE) *)
000074 151          TARGPX : INTEGER;
000074 152          TEXTENDX : INTEGER;
000074 153          TEXTENDY : INTEGER;

```

```

000074 154 LINELEN : INTEGER;
000074 155 MAXLINE : INTEGER;
000074 156 CURLINE : INTEGER;
000074 157 TARGS : TARGPTR;
000074 158 H*AD : DSPBUPPTR; (* FIRST LINE BUFFER *)
000074 159 CURPTR : DSPBUPPTR; (* CURRENT LINE BUFFER *)
000074 160 LSTPTR : TEXTPTR; (* WHERE DISPLAY LEFT OFF *)
000074 161 INUSE : BOOLEAN;
000074 162 NEIGHBORS : INTEGER;
000074 163 INSIDERS : INTEGER;
000074 164 END;
000074 165 (*ADJ = (CENTER,LL,UL,LR,UR); *)
000074 166 ROM = PACKED ARRAY[0..127,0..7] OF 0..177777B;
000074 167 MAJORSTATE = (DSPTEXT,GRAPHICS);
000074 168 MINORSTATE = (DSPNONE,TOUCH,PLACE,POINT,LINE,CHPLOT,CHLOAD);
000074 169 MEMORY = (NORMAL,PROGRAMMABLE);
000074 170
000074 171 (* THESE WOULD NORMALLY BE IN THE VARS *)
000074 172 DSSTATIC = RECORD
000074 173   CHARSET : ROM; (* CHARACTER SET ARRAY *)
000074 174   LABARR : ARRAY[1..TARGMAX] OF LABPTR; (* FOR STORAGE OF LABELS BEFORE GIVING THEM TO CREATPTARG *)
000074 175   (* THIS IS SCRAPPED IN DSTBLOCK *)
000074 176   LABCTR : 0..TARGMAX; (* COUNTER OF NUMBER OF TARGETS *)
000074 177   LOOKUP : ARRAY[1..NCOMMANDS,1..3] OF CRNG;
000074 178   DISARM : ARRAY[1..BLOCKMAX] OF INTEGER;
000074 179   BLOCK0 : INTEGER;
000074 180   SAVEMODE : INTEGER;
000074 181   BLK : ARRAY[1..BLOCKMAX] OF PLOCK;
000074 182   TARGARRAY : ARRAY[0..15,0..15] OF TARGETS;
000074 183   BUILD : ARRAY[0..15,0..7] OF CHAR;
000074 184   REV : ARRAY[0..15] OF INTEGER;
000074 185   CHARS : ARRAY[12..127] OF CHAR;
000074 186   CONV : ARRAY['+','-',':'] OF INTEGER;
000074 187   SAVE1 : MAJORSTATE;
000074 188   SAVE2 : MINORSTATE;
000074 189   I,J,K : INTEGER; (* TEMP *)
000074 190   CBUF : INBUF;
000074 191 END;
000074 192 TMSTATIC = DUMMY;
000074 193 PRSTATIC = DUMMY;
000074 194 RESTATIC = DUMMY;
000074 195 SISTATIC = DUMMY;
000074 196 S2STATIC = DUMMY;
000074 197 (*$IGBLVARS/'KBLIR'*)
  
```

----- BEGIN INCLUDED TEXT.

```

000074 197 VAR (* GLOBAL VARIABLE DEFINITIONS *)
000074 197
000074 197 (* POINTERS TO PRIVATE STATIC AREAS *)
000074 197
000074 197 KB : TEXT; (* KNOWLEDGE BASE FILE *)
000130 197 MSG : TEXT; (* MESSAGES WORK FILE *)
000164 197 TMSPTR : ^TMSTATIC; (* FOR THE TUIE MANAGER *)
000165 197 PRSPTR : ^PRSTATIC; (* FOR THE PARSER *)
000166 197 RESPTR : ^RESTATIC; (* FOR THE RULE EVALUATOR *)
000167 197 DSSPTR : ^DSSTATIC; (* FOR TERMINAL DISPLAY FUNCTIONS *)
  
```

```
000170 197 S1SPTR : ^S1STATIC; (* SPARE 1 *)
000171 197 S2SPTR : ^S2STATIC; (* SPARE 2 *)
000172 197 GBLError: ERRORTYPE; (* MAIN ERROR INDICATOR *)
000173 197
000173 197 (*$E+ ASK FOR REAL EXTERNAL NAMES *)
000173 197 (*$R- WITHOUT REDUCE *)
```

----- END INCLUDED TEXT.

```
000173 198
000173 199
000173 200 (*$I'IOAIDS'*)
```

----- BEGIN INCLUDED TEXT.

```
000173 200 (* --- IOAIDS - INPUT/OUTPUT AIDS. --- *)
000173 200 (*$A+ ASCII CHARACTER SET *)
000173 200 PROCEDURE SKIPBLANKS(VAR P : TEXT);
000003 200 BEGIN (* SKIPBLANKS *)
000003 200 WHILE (P^ = ' ') AND NOT EOP(P) DO GET(P)
000014 200 END (* SKIPBLANKS *);
000017 200
000017 200 FUNCTION EOLNS(VAR P : TEXT) : BOOLEAN;
000004 200 BEGIN (* EOLNS *)
000004 200 WHILE (P^ = ' ') AND NOT EOLN(P) DO GET(P);
000015 200 EOLNS := EOLN(P)
000016 200 END (* EOLNS *);
000022 200
000022 200 FUNCTION EOSS(VAR P : SEGTEXT) : BOOLEAN;
000004 200 BEGIN (* EOSS *)
000004 200 WHILE (P^ = ' ') AND NOT EOS(P) DO GET(P);
000015 200 EOSS := EOS(P)
000016 200 END (* EOSS *);
000022 200
000022 200 FUNCTION EOPS(VAR P : TEXT) : BOOLEAN;
000004 200 BEGIN (* EOPS *)
000004 200 WHILE (P^ = ' ') AND NOT EOP(P) DO GET(P);
000015 200 EOPS := EOP(P)
000016 200 END (* EOPS *);
000022 200
000022 200 PROCEDURE READSTRING(VAR P : TEXT; VAR S : DYNAMIC ALFA;
000004 200                      LEN : INTEGER);
000011 200 VAR I, LIM : INTEGER;
000013 200 BEGIN (* READSTRING *)
000013 200 IF EOP(P) THEN HALT(' READ PAST EOS/EOP. ');
000014 200 IF LEN < 0
000014 200 THEN BEGIN SKIPBLANKS(P);
000017 200 I := 0;
000021 200 WHILE P^ <> ' ' DO
000025 200 BEGIN IF I < HIGH(S)
000026 200 THEN BEGIN I := I + 1; S[I] := P^ END;
000045 200 GET(P)
000051 200 END;
000052 200 LEN := I
000052 200 END
000054 200 ELSE BEGIN IF LEN > HIGH(S) THEN LIM := HIGH(S) ELSE LIM := LEN;
000063 200 FOR I := 1 TO LIM DO
```

PASCAL COMPILER - E.T.H. ZURICH / UNIVERSITY OF MINNESOTA.
 DISPLAY MODULE DATA TYPES FOR DISPLAY MODULE

PASCAL 6000 V3.0.0. 80/11/17. 00.56.01.
 NOS 1.4 (80/04/21) PAGE 9

```

000066 200 BEGIN S[I] := P; IF NOT EOLN(P) THEN GET(P) END;
000117 200 END;
000117 200 IF LEN < HIGH(S)
000121 200 THEN FOR I := LEN+1 TO HIGH(S) DO S[I] := ' '
000136 200 ELSE FOR I := HIGH(S)+1 TO LEN DO
000147 200 IF NOT EOLN(P) THEN GET(P)
000156 200 END (* READS *);
000200 200 (*SA=*)

```

----- END INCLUDED TEXT.

```

000200 201 PROCEDURE OPEN (
000202 202   VAR P : TEXT; (* PASCAL TEXT FILE NAME TO OPEN *)
000203 203   N : ALFA; (* LITERAL FILE NAME TO BE OPEN *)
000204 204   OPENWRITE : BOOLEAN (* TRUE FOR WRITE *)
000205 205 );
000206 206 EXTERN;
000207 207
000208 208 PROCEDURE CLOSE (
000209 209   VAR P : TEXT (* PASCAL TEXT NAME TO CLOSE *)
000210 210 );
000211 211 EXTERN;
000212 212
000213 213
000214 214

```

```

000003 562 (*SL'LOW LEVEL ROUTINES'*)
000003 563
000003 564
000003 565
000003 566
000003 567
000003 568 LL 00000 WW WW
000003 569 LL 00 00 WW WW
000003 570 LL 00 0C WW WW
000003 571 LL 00 00 WW WW
000003 572 LL 00 00 WW W WW
000003 573 LL 00 00 WWWWWW
000003 574 LLLLLLL 00000 WW WW
000003 575
000003 576
000003 577
000003 578 LL EEEEEEE V V EEEEEEE LI
000003 579 LL EE V V EE LL
000003 580 LL EE VV VV EE LL
000003 581 LL EEEEE V V EEEEE LL
000003 582 LL EE VVVV E? LI
000003 583 LL EE VV EE LI
000003 584 LLLLLLL EEEEEEE VV EEEEEEE LLLLLLL
000003 585
000003 586
000003 587
000003 588
000003 589 PPPPPP RRRRRR 00000 CCCCC SSSSSS
000003 590 PP PP RR RR 00 00 CC CC SS
000003 591 PP PP RR RR 00 00 CC SS
000003 592 PPPPPP RRRRRR 00 00 CC SSSSS
000003 593 PP RRRR 00 00 CC SS
000003 594 PP RR RR 00 00 CC CC SS
000003 595 PP RR RR 00000 CCCCC SSSSSS
000003 596
000003 597
000003 598
000003 599
000003 600
000003 601
000003 602
000003 603
000003 604
000003 605 (*-----*)
000003 606 PROCEDURE PUTC (
000002 607 VAR OUTFILE : TEXT; (* WHERE TO SEND THE CHARACTER *)
000003 608 IT : INTEGER (* THE INTEGER REPRESENTATION TO BE CONVERTED *)
000004 609 );
000004 610
000004 611 (* USER DESCRIPTION:
000004 612 YOU SHOULD USE THIS PROCEDURE IF YOU WANT TO CONVERT
000004 613 THE INTEGER REPRESENTATION OF AN ASCII CHARACTER TO
000004 614 THE DISPLAY CODE REPRESENTATION OF THAT CHARACTER.
000004 615 IF ONE IS IN ASCII MODE (I.E. BY ISSUING THE ASCII
000004 616 COMMAND TO TELEX), THEN THE ASCII CHARACTER WILL
000004 617 BE DISPLAYED ON THE TERMINAL. THIS INCLUDES ALL CONTROL
000004 618 CODES. IN ADDITION THE NUMBER 128 CAUSES A WRITEIN.

```

```

000004 619  *)
000004 620
000004 621  (* INTERNAL DESCRIPTION:
000004 622  THIS PROCEDURE USES A COMBINATION OF TABLE LOOKUP AND CONDITIONALS.
000004 623  *)
000004 624
000004 625 BEGIN
000004 626   IF (IT >= 0) AND (IT < 32) THEN WRITE(OUTFILE, '-', DSSPTR^.CHARS[IT+96])
000032 627   ELSE IF IT = 58 THEN WRITE(OUTFILE, 'a', DSSPTR^.CHARS[68])
000054 628   ELSE IF IT = 64 THEN WRITE(OUTFILE, 'a', DSSPTR^.CHARS[65])
000076 629   ELSE IF IT = 94 THEN WRITE(OUTFILE, 'a', DSSPTR^.CHARS[66])
000120 630   ELSE IF (IT > 31) AND (IT < 96) THEN WRITE(OUTFILE, DSSPTR^.CHARS[IT])
000142 631   ELSE IF IT = 96 THEN WRITE(OUTFILE, 'a', DSSPTR^.CHARS[71])
000164 632   ELSE IF (IT > 96) AND (IT < 123) THEN
000170 633     WRITE(OUTFILE, '-', DSSPTR^.CHARS[IT-32])
000213 634   ELSE IF (IT > 122) AND (IT < 128) THEN
000217 635     WRITE(OUTFILE, '-', DSSPTR^.CHARS[IT-75])
000243 636   (* ELSE IF IT = 128 THEN WRITELN(OUTFILE) *)
000243 637 END; (* ** PUTCH ** *)
000247 638
000247 639
000247 640  (*=====*)
000247 641  FUNCTION CNVT (
000002 642    VAR IOPFILE : TEXT; (* INPUT FILE BEING CONVERTED *)
000003 643    VAR CH : CHAR (* CHARACTER TO BE CONVERTED *)
000004 644    ) : INTEGER;
000005 645
000005 646  (* USER DESCRIPTION:
000005 647  THIS FUNCTION IS USED TO CONVERT A CHARACTER TO AN INTEGER
000005 648  IS THE ASCII ORD OF THE CHARACTER. SINCE THE DISPLAY
000005 649  CODE REPRESENTATION OF SOME ASCII CHARACTERS ARE 2 CHARACTER
000005 650  SEQUENCES, THIS FUNCTION NEEDS TO KNOW ABOUT THE TEXT
000005 651  FILE FROM WHICH THE CHARACTER WAS OBTAINED. NOTE THAT
000005 652  IF THIS FUNCTION DOES READ IN ANOTHER CHARACTER, THE
000005 653  PARAMETER CH IS CHANGED TO IT.
000005 654  *)
000005 655
000005 656
000005 657  (* INTERNAL DESCRIPTION:
000005 658  THE PROCESS OF CONVERSION IS STRIGHTFORWARD. IF THE CURRENT
000005 659  CHARACTER IS EITHER A ~ OR AN a, THEN THE DISPLAY CODE
000005 660  REPRESENTATION IS A 2 CHARACTER SQUENCE. WE NEED TO
000005 661  READ IN THE SECOND CHARACTER FROM THE FILE IOPFILE.
000005 662  *)
000005 663
000005 664 BEGIN
000005 665   IF CH='~' THEN BEGIN
000006 666     READ(IOPFILE, CH);
000016 667     IF (ORD(CH) >= 32) THEN CNVT := ORD(CH) - 32
000021 668     ELSE CNVT := ORD(CH) + 96
000023 669   END
000024 670   ELSE IF CH='a' THEN BEGIN
000027 671     READ(IOPFILE, CH);
000037 672     CASE CH OF
000041 673       'A' : CNVT := 64;
000043 674       'B' : CNVT := 94;
000045 675       'D' : CNVT := 50;

```

```

000047 676      'G' : CNVT := 96
000047 677      END
000061 678      END
000061 679      ELSE IF CH = ':' THEN CNVT := 59
000064 680      ELSE IF CH IN ['0'..'9'] THEN CNVT := ORD(CH) + 21
000070 681      ELSE IF CH IN ['A'..'Z'] THEN CNVT := ORD(CH) + 64
000075 682      ELSE CNVT := ISSPTR.CONV[CH]
000106 683      END;      (* CNVT *)
000115 684
000115 685
000115 686  (*-----*)
000115 687      FUNCTION CNVTA (
000002 688          BUF : INBUF;      (* ARRAY OF CHARACTERS BEING CONVERTED *)
000003 689          VAR IND : INTEGER (* INDEX INTO THE ARRAY BUF *)
000004 690          ) : INTEGER;
000023 691
000023 692      (* USER DESCRIPTION:
000023 693      THIS FUNCTION IS LIKE THE FUNCTION CNVT, BUT IS USED WHEN
000023 694      THE CHARACTERS TO BE CONVERTED ARE IN AN ARRAY. THIS TIME,
000023 695      IF THERE IS A NEED TO GO TO ANOTHER CHARACTER, THE
000023 696      INDEX INTO THE ARRAY IND IS INCREMENTED.
000023 697      *)
000023 698
000023 699      (* INTERNAL DESCRIPTION:
000023 700      SEE THE DESCRIPTION FOR CNVT.
000023 701      *)
000023 702
000023 703      VAR
000023 704      CH : CHAR;
000024 705
000024 706
000024 707 BEGIN
000024 708     CH := BUF[IND];
000024 709     IF CH = ':' THEN BEGIN
000026 710         IND := IND + 1;
000027 711         CH := BUF[IND];
000043 712         IF (ORD(CH) >= 32) THEN CNVTA := ORD(CH) - 32
000044 713             ELSE CNVTA := ORD(CH) + 96
000046 714         END
000047 715     ELSE IF CH = 'A' THEN BEGIN
000052 716         IND := IND + 1;
000054 717         CH := BUF[IND];
000067 718         CASE CH OF
000070 719             'A' : CNVTA := 64;
000072 720             'R' : CNVTA := 94;
000074 721             'D' : CNVTA := 58;
000076 722             'G' : CNVTA := 96
000076 723         END
000110 724     END
000110 725     ELSE IF CH = ':' THEN CNVTA := 59
000113 726     ELSE IF CH IN ['0'..'9'] THEN CNVTA := ORD(CH) + 21
000117 727     ELSE IF CH IN ['A'..'Z'] THEN CNVTA := ORD(CH) + 64
000121 728     ELSE CNVTA := ISSPTR.CONV[CH]
000133 729     END;      (* CNVTA *)
000143 730
000143 731
000143 732  (*-----*)

```

```

000143 733 FUNCTION DSPCHARCNT (
000002 734     VAR ARR: DYNAMIC PASBUP; (* ARRAY THAT HAS THE CHARACTERS TO BE COUNTED *)
000003 735     LEN : INTEGER      (* LENGTH OF STRING TO BE COUNTED *)
000004 736 ) : INTEGER;
000011 737
000011 738 (* USER DESCRIPTION :
000011 739     THIS FUNCTION COUNTS THE NUMBER OF ASCII CHARACTERS IN
000011 740     THE ARRAY ARR. C/C CYBER DISPLAY CODE IS A 6 BIT CODE
000011 741     SO, IN ORDER TO REPRESENT THE FULL ASCII CHARACTER
000011 742     SET, TWO DISPLAY CODE CHARACTERS ARE USED FOR THE LOWER
000011 743     CASE CHARACTERS & THE CONTROL CHARACTERS. THE CHARACTERS
000011 744     AND AT ARE USED AS ESCAPE CHARACTERS
000011 745 *)
000011 746
000011 747 (* INTERNAL DESCRIPTION:
000011 748     OBVIOUS. LOOP THROUGH INDEXING INTO ARRAY. IF IT IS ONE
000011 749     OF THE MAGIC CHASRACTERS @ OR ^ THEN THIS IS A TWO CHARACTER
000011 750     DISPLAY CODE REPRESENTATION FOR THE ASCII CHARACTER.
000011 751 *)
000011 752
000011 753
000011 754 VAR
000011 755     DCNT : INTEGER;
000012 756     I : INTEGER;
000013 757
000013 758 BEGIN
000013 759     DCNT := 0;
000013 760     I := 1;
000014 761     WHILE I <= LEN DO BEGIN
000017 762         IF (ARR[I] = '^') OR (ARR[I] = '@') THEN BEGIN
000046 763             I := I + 2;
000050 764         END
000050 765         ELSE BEGIN
000051 766             I := I + 1;
000053 767         END;
000053 768         DCNT := DCNT + 1;
000055 769     END;
000056 770     DSPCHARCNT := DCNT;
000060 771 END;
000074 772
000074 773
000074 774 (*=====*)
000074 775 FUNCTION CNTCHARS (
000002 776     LAB1 : LABELS; (* THE ARRAY THAT CONTAINS THE TARGET LABEL *)
000003 777     LABFLLEN : INTEGER (* THE LENGTH OF THE LABEL IN DISPLAY CODE CHARACTERS *)
000004 778 ) : INTEGER;
000014 779
000014 780 (* USER INTERFACE:
000014 781     THE PURPOSE OF THIS FUNCTION IS TO TELL CREATETARG THE
000014 782     DIMENSIONS OF THE LABEL FOR A TOUCH TARGET. THIS NOW
000014 783     ONLY COMPUTES THE X LENGTH, BUT IN THE FUTURE THIS FUNCTION
000014 784     SHOULD BECOME A PROCEDURE AND COMPUTE THE Y LENGTH AS
000014 785     WELL.
000014 786 *)
000014 787
000014 788 (* INTERNAL DESCRIPTION:
000014 789     CALLS DSPCHARCNT FOR NOW, BUT IN THE FUTURE, IF TARGETS

```



```

000014 790 GET THE ABILITY TO HAVE MULTI-LINFS AS WELL AS GRAPHICS,
000014 791 THIS FUNCTION WILL BECOME MUCH MORE INVOLVED.
000014 792 *)
000014 793
000014 794 VAR
000014 795 I : INTEGER;
000015 796
000015 797 BEGIN
000015 798 CNTCHARS := DSPCHARCNT(LABL,LABELLEN);
000014 799 END;
000025 800
000025 801
000025 802 (*=====*)
000025 803 FUNCTION DISABLED (
000002 804 ID : INTEGER; (* BLOCK ID TO CHECK *)
000003 805 VAR I : INTEGER (* THE INDEX WHERE ID WAS FOUND IN THE DISARM ARRAY *)
000004 806 ): BOOLEAN; (* TRUE IF ID WAS DISABLED *)
000005 807
000005 808 (* USER DESCRIPTION:
000005 809 THIS FUNCTION CHECKS TO SEE IF THE BLOCK, ID, IS DISABLED.
000005 810 IF IT IS, IT SETS I TO THE INDEX INTO THE DISARM ARRAY.
000005 811 *)
000005 812
000005 813 (* INTERNAL DESCRIPTION:
000005 814 LOOP THROUGH DISARM ARRAY UNTIL YOU FIND ID. RETURN THE
000005 815 INDEX INTO THE ARRRY FOR ID AS I.
000005 816 *)
000005 817
000005 818
000005 819 VAR
000005 820 HIT : BOOLEAN;
000006 821
000006 822 BEGIN
000006 823 I := 1;
000005 824 HIT := DSSPTR^.DISARM[I] = ID;
000021 825 WHILE (I < BLOCKMAX) AND (NOT HIT) DO BEGIN
000025 826 I := I + 1;
000027 827 HIT := DSSPTR^.DISARM[I] = ID;
000041 828
000042 829 END;
000042 829 DISABLED := HIT;
000045 830
000055 831
000055 832
000055 833 (*=====*)
000055 834 FUNCTION NEST (
000002 835 IS,IS,XE,YE : INTEGER ; (* THE START AND END POINTS OF THE BLOCK
000006 836 TO CHECK FOR NESTING IN THE FOLLOWING BLOCK *)
000006 837 XS1,YS1,XE1,YE1 : INTEGER (* THE NESTING BLOCK *)
000012 838 ): NESTTYPE; (* NESTED, REVNESTED, OR OVERLAP *)
000013 839
000013 840 (* USER DESCRIPTION:
000013 841 THIS FUNCTION CHECKS FOR THE NESTING OF THE BLOCK DESCRIBED
000013 842 BY THE FIRST SET OF ENDPOINTS IN THE BLOCK DESCRIBED
000013 843 IN THE SECOND SET OF ENDPOINTS. THE FUNCTION RETURNS THE
000013 844 SCALAR TYPE NESTTYPE INDICATING WHETHER THE BLOCKS ARE
000013 845 NESTED, REVERSE NESTED, OR OVERLAPPING.
000013 846 *)

```

PASCAL COMPILER - E.T.H. ZURICH / UNIVERSITY OF MINNESOTA.
 DISPLAY MODULE LOW LEVEL ROUTINES

PASCAL 6000 V3.0.0. 80/11/17. 00.56.01. .
 NOS 1.4 (80/04/21) PAGE 22

```

000013 847
000013 848 (* INTERNAL DESCRIPTION:
000013 849 THE LOGIC IS SELF DESCRIPTIVE.
000013 850 *)
000013 851
000013 852
000013 853 BEGIN
000013 854   IF ((XS1 <= XS) AND
000006 855       (XE1 >= XE) AND
000007 856       (YS1 <= YS) AND
000011 857       (YE1 >= YE) ) THEN BEGIN
000014 858     NFST := NESTED;
000015 859   END
000015 860   ELSE BEGIN
000016 861     IF ((XS <= XS1) AND
000020 862         (XE >= XE1) AND
000021 863         (YS <= YS1) AND
000023 864         (YE >= YE1) ) THEN BEGIN
000025 865       NEST := REVNESTED;
000026 866     END
000026 867     ELSE BEGIN
000027 868       IF ((YE1 <= XS) OR
000031 869           (XE <= XS1) OR
000032 870           (YE1 <= YS) OR
000034 871           (YE <= YS1) ) THEN BEGIN
000036 872         NFST := NOTNESTED;
000037 873       END
000037 874       ELSE BEGIN
000040 875         NEST := OVERLAP;
000042 876       END;
000042 877     END;
000042 878   END;
000042 879 END;
000045 880
000045 881
000045 882 (*=====*)
000045 883 PROCEDURE CONVMETRIC (
000002 884   X,Y : INTEGER; (* THE VALUES TO BE CONVERTED *)
000004 885   M : METRIC; (* THE METRIC THESE VALUES ARE IN:
000005 886               DOTS, CHS, OR PARTS *)
000005 887   VAR XS,YS : INTEGER (* THE CONVERTED VALUES IN DOTS *)
000007 888 );
000007 889
000007 890 (* USER DESCRIPTION:
000007 891 THIS UTILITY PROCEDURE CONVERTS ONE TYPE OF SCALE FOR
000007 892 SPECIFYING THE DIMENSIONS OF LINES ETC. ENTERED BY A
000007 893 USER TO THE STANDARD FORM OF IN TERMS OF DOTS. THE
000007 894 USER CAN HAVE THE SCALE IN DOTS, CHARACTERS, OR FRACTIONS
000007 895 OF A SCREEN.
000007 896 *)
000007 897
000007 898
000007 899 (* INTERNAL DESCRIPTION:
000007 900 JUST A CASE STATEMENT. THE WEIRD NUMBERS FOR FRACTIONS
000007 901 OF A SCREEN IS FOR INSURING THAT THE NUMBER 512 IS NEVER
000007 902 GENERATED. THUS 0% IS 0 AND 100% IS 511 NOT 512. ALSO
000007 903 FRACTIONS OF A SCREEN IS ENTERED IN TERMS OF PERCENT.

```

```

000007 904 *)
000007 905
000007 906 BEGIN
000007 907   CASE M OF
000005 908     DOTS : BEGIN
000005 909       XS := X;
000006 910       YS := Y;
000010 911     END;
000011 912
000011 913     CMS : BEGIN
000011 914       XS := X * 8;
000013 915       YS := Y * 16;
000015 916     END;
000016 917
000016 918     PARTS : BEGIN
000016 919       XS := (5120 * X) DIV 1001;
000026 920       YS := (5120 * Y) DIV 1001;
000036 921     END;
000037 922   END; (* OF CASE *)
000044 923 END; (* OF CONVNETIC *)
000060 924
000060 925
000060 926 (*=====*)
000060 927   PROCEDURE CENTERIT (
000002 928     X,XLEN,Y,YLEN : INTEGER; (* X AND Y ARE THE START DISTANCE
000006 929     AWAY FROM THE ORIGIN SPECIFIED IN THE NEXT PARAM. *)
000006 930     CENTERING : ADJ; (* ORIGIN: LL,UL,LR,UR,CENTER *)
000007 931     VAR XS,YS : INTEGER (* THE END POINTS COMPUTED WITH THE LENGTHS XLEN,YLEN *)
000011 932   );
000011 933
000011 934   (* USER DESCRIPTION:
000011 935     THIS PROCEDURE IS USED TO COMPUTE THE END POINTS FROM
000011 936     THE START POINTS AND LENGTHS GIVEN WITH RESPECT TO AN ORIGIN SPECIFIED.
000011 937     THE ORIGIN CAN BE ONE OF THE FOUR CORNERS OF A BLOCK
000011 938     OR ITS CENTER.
000011 939   *)
000011 940
000011 941   (* INTERNAL DESCRIPTION:
000011 942     THESE ARE TRIVIAL CALCULATIONS.
000011 943   *)
000011 944
000011 945
000011 946 BEGIN
000011 947   CASE CENTERING OF
000006 948     CENTER : BEGIN
000006 949       XS := X - (XLEN DIV 2);
000011 950       YS := Y - (YLEN DIV 2);
000013 951     END;
000014 952
000014 953     LL : BEGIN
000014 954       XS := X;
000016 955       YS := Y;
000017 956     END;
000020 957
000020 958     UL : BEGIN
000020 959       XS := X;
000022 960       YS := Y - YLEN;

```

```

000023 961      END;
000024 962
000024 963      LR : BEGIN
000024 964          XS := X - ILEN;
000026 965          YS := Y;
000027 966      END;
000030 967
000030 968      UR : BEGIN
000030 969          XS := X - ILEN;
000032 970          YS := Y - YLEN;
000033 971      END;
000034 972      END; (* OF CASE *)
000042 973      END; (* OF CENTERIT *)
000042 974
000062 975
000062 976      (* ===== *)
000062 977      PROCEDURE MODE (
000002 978          S1 : MAJORSTATE; (* DSPTXT OR GRAPHICS *)
000003 979          S2 : MINORSTATE; (* DSENONE, TOUCH, PLACE, POINT,
000004 980                          LINE, CHPLOT, CHLOAD *)
000004 981          WMODE : INTEGER; (* 3 { = CONST ERASE }
000005 982                          4 { = CONST DSPNORM }
000005 983                          6 { = CONST OVER }
000005 984                          16 { = CONST INVERSE } *)
000005 985          PAGE : INTEGER (* 0 { = CONST DSPNO }
000006 986                          1 { = CONST DSPYES } *)
000006 987      );
000006 988
000006 989      (* USER DESCRIPTION :
000006 990      THIS PROCEDURE GET THE ORION TERMINAL INTO ITS PROPER
000006 991      STATE FOR DOING VARIOUS ACTIONS. THE MAJOR DIVISION
000006 992      IS BETWEEN GRAPHICS AND TEXT. THE MINOR DIVISION IS
000006 993      FOR VARIOUS GRAPHIC ACTIONS AS WELL AS LOADING CHARACTERS
000006 994      INTO THE ORION. WHENEVER THIS PROCEDURE IS CALLED,
000006 995      (AND IT SHOULD BE CALLED WHEN YOU WISH TO DO SOMETHING
000006 996      DIFFERENT FROM WHAT YOU WERE DOING BEFORE), YOU HAVE THE
000006 997      OPTION OF SETTING THE WRITING MODE (SCFF ACTIONS, LIKE
000006 998      LINE DRAWING AND POINT PLOTTING, HAVE ONLY TWO POSSIBLE
000006 999      WRITING MODES (NORMAL OR FRASE), OTHERS HAVE NONE
000006 1000      (CHLOAD IN WHICH CASE THE WRITING MODE CAN BE SET
000006 1001      TO ANYTHING). THERE IS ALSO THE OPTION OF CLEARING THE
000006 1002      SCREEN BEFORE SETTING THE MODE AND THIS IS CONTROLLED
000006 1003      BY PAGE. THIS IS PROVIDED BECAUSE IT WAS PROVIDED BY
000006 1004      THE ORION PROTOCOL AND I WANTED TO PRESERVE AS MUCH
000006 1005      CAPABILITY AS POSSIBLE AT THIS LEVEL. FINALLY, WHEN
000006 1006      SETTING THE MAJOR STATE TO TEXT, SET THE MINOR STATE
000006 1007      TO NONE.
000006 1008      *)
000006 1009
000006 1010      (* INTERNAL DESCRIPTION:
000006 1011      THIS PROCEDURE WORKS BY HAVING THE PRIOR STATE OF THE
000006 1012      TERMINAL REMEMBERED IN TWO GLOBAL VARIABLES. IT THEN
000006 1013      COMPARES THE CURRENT STATE WITH THE PRIOR STATE. IF
000006 1014      THERE IS A DIFFERENCE IN THE MAJOR STATE, THEN THE
000006 1015      SEQUENCE TO GET THE ORION INTO TEXT OR GRAPHICS IS
000006 1016      SENT. IF THERE IS A DIFFERENCE IN THE MINOR STATE,
000006 1017      (OR YOU WANT TO CLEAR THE SCREEN OR CHANGE THE WRITING
  
```

```

000006 1018 MODE WHILE IN GRAPHICS) THEN THE LOAD MODE 3 CHARACTER
000006 1019 SEQUENCE IS SENT. THIS SEQUENCE LOOKS LIKE:
000006 1020
000006 1021 CHARACTER 1 : /1/001/0/XX/ (X IS DON'T CARE )
000006 1022 CHARACTER 2 : /1/000000/
000006 1023 CHARACTER 3 : /1/X/HH/PP/S/ WHERE
000006 1024
000006 1025 PM = 0 - POINT PLOT
000006 1026 1 - VECTOR PLOT
000006 1027 2 - LOAD MEMORY
000006 1028 3 - CHARACTER PLOT
000006 1029
000006 1029 EP = 0 - ERASE
000006 1030 1 - WRITE
000006 1031 2 - INVERSE
000006 1032 3 - OVERSTRIKE
000006 1033
000006 1034 S = 0 - NOERASE
000006 1035 1 - ERASE
000006 1036
000006 1037 FINALLY, IF THE MODE YOU ARE IN IS TEXT, AND YOU SET
000006 1038 THE WRITE MODE DIFFERENTLY OR YOU SET PAGE, THEN
000006 1039 THESE ACTIONS ARE SIMULATED TO INCREASE THE ORTHOGONALITY
000006 1040 OF THE SEMANTICS.
000006 1041 *)
000006 1042
000006 1043
000006 1044 PROCEDURE SETUP( SECMODE : INTEGER ; WMODE,PAGE : INTEGER);
000005 1045 VAR
000005 1046 W : INTEGER;
000006 1047 FUNC : INTEGER;
000007 1048
000007 1049 BEGIN
000007 1050 CASE WMODE OF
000005 1051 3: W := 0;
000007 1052 4: W := 1;
000011 1053 6: W := 3;
000013 1054 16: W := 2;
000015 1055 END;
000030 1056 PUTCH(OUTPUT,LF);
000032 1057 PUTCH(OUTPUT,64); (* AN 0 *)
000034 1058 FUNC := 64 + W*2 + PAGE + SECMODE;
000040 1059 PUTCH(OUTPUT,FUNC);
000042 1060 END;
000056 1061
000056 1062
000056 1063
000056 1064 BEGIN
000056 1065 IF S1 < DSSPTR.SAVE1 THEN BEGIN (*GO INTO TEXT MODE*)
000012 1066
000012 1067 (* THIS SEQUENCE IS FROM THE WORK OF PELLET *)
000012 1068 PUTCH(OUTPUT,07);
000014 1069 PUTCH(OUTPUT,127);
000016 1070 PUTCH(OUTPUT,127);
000020 1071 PUTCH(OUTPUT,DSPCR);
000022 1072 PUTCH(OUTPUT,BEL);
000024 1073 PUTCH(OUTPUT,BEL);
000026 1074 PUTCH(OUTPUT,BEL);

```

```

000030 1075     END;
000030 1076     IF S1 > DSSPTR^.SAVE1 THEN BEGIN      (*GO INTO GRAPHICS *)
000037 1077         PUTCH(OUTPUT,GS);
000041 1078     END;
000041 1079     IF S1 <> DSPTEXT THEN BEGIN
000043 1080         IF (S2 <> DSSPTR^.SAVE2) OR (PAGE = DSPYES) OR (DSSPTR^.SAVEMODE <> WMODE) THEN BEGIN
000061 1081             CASE S2 OF
000062 1082                 POINT: BEGIN
000062 1083                     SETUP(0,WMODE,PAGE);
000064 1084                     DSSPTR^.SAVEMODE := WMODE;
000072 1085                 END;
000073 1086                 LINE : BEGIN
000073 1087                     SETUP(8,WMODE,PAGE);
000075 1088                     DSSPTR^.SAVEMODE := WMODE;
000103 1089                 END;
000104 1090                 CHPIOT : BEGIN
000104 1091                     SETUP(24,WMODE,PAGE);
000106 1092                     DSSPTR^.SAVEMODE := WMODE;
000114 1093                 END;
000115 1094                 CHLOAD : BEGIN
000115 1095                     SETUP(16,WMODE,PAGE);
000117 1096                 END;
000120 1097                 OTHERWISE
000126 1098                     END;      (* OF CASE *)
000126 1099             END;
000126 1100         END
000126 1101     ELSE BEGIN
000127 1102         IF PAGE = DSPYES THEN PUTCH(OUTPUT,CLPAR);
000133 1103         IF (DSSPTR^.SAVEMODE <> WMODE) OR (PAGE = DSPYES) THEN PUTCH(OUTPUT,WMODE);
000145 1104         DSSPTR^.SAVEMODE := WMODE;
000153 1105     END;
000153 1106     DSSPTR^.SAVE1 := S1;
000162 1107     DSSPTR^.SAVE2 := S2;
000171 1108     END;      (* OF MODE*)
000203 1109
000203 1110     (*=====*)
000203 1111     PROCEDURE SITCOORD (
000092 1112         X,Y : INTEGER;      (* X AND Y COORDINATES *)
000094 1113         M : METRIC          (* DOTS,CHS,OR PARTS *)
000095 1114     );
000095 1115
000095 1116     (* USER DESCRIPTION:
000095 1117     NOTHING SPRIAL ABOUT THIS ONE.
000095 1118     *)
000095 1119
000095 1120     (* INTERNAL DESCRIPTION:
000095 1121     SEND OUT THE SET COORIDINATE SEQUENCE AS FOLLOWS:
000095 1122
000095 1123         CHARACTER 1 : /0/010/XXX/
000095 1124         CHARACTER 2 : /1/XX/C/ABA7A6/
000095 1125         CHARACTER 3 : /1/A5A4A3A2A1A0/
000095 1126
000095 1127         WHERE
000095 1128             C = 0 - SET X COORD
000095 1129             1 - SET Y COORD
000095 1130             AN(N=7..0) ARE ADDRESS BITS
000095 1131
000095 1131     THIS SEQUENCE IS SENT OUT TWICE, ONCE FOR X, ONCE FOR Y.

```

```

000005 1132 *)
000005 1133
000005 1134 VAR
000005 1135   X,Y : INTEGER;
000007 1136   X1 , X2 , Y1 , Y2 : INTEGER;
000013 1137 BEGIN
000013 1138   CONV METRIC (X3,Y3,M,X,Y);
000010 1139   MODE (GRAPHICS,PLACE,DSPNORM,DSPNO);
000015 1140   PUTCH (OUTPUT,DLE);
000017 1141   X1 := (X DIV 64) + 64;
000023 1142   PUTCH (OUTPUT,X1);
000025 1143   X2 := (X MOD 64) + 64;
000031 1144   PUTCH (OUTPUT,X2);
000033 1145   PUTCH (OUTPUT,DLE);
000035 1146   Y1 := (Y DIV 64) + 72;
000041 1147   PUTCH (OUTPUT,Y1);
000043 1148   Y2 := (Y MOD 64) + 64;
000047 1149   PUTCH (OUTPUT,Y2);
000051 1150 END;
000075 1151
000075 1152 (*-----*)
000075 1153 PROCEDURE DRAWLINE (
000002 1154   X3,Y3 : INTEGER; (* END POINT COORDS *)
000004 1155   M : METRIC; (* DOTS,CHS, OR PARTS *)
000005 1156   WMODE : INTEGER; (* THE WRITING MODE, SEE MODE *)
000006 1157   PAGE : INTEGER (* CLEAR SCREEN FLAG, SEE MODE *)
000007 1158 );
000007 1159
000007 1160 (* USER DESCRIPTION:
000007 1161   DRAW A LINE FROM THE LAST POINT THE CURSOR WAS SET TO
000007 1162   THE ENDPOINTS X3,Y3. YOU CAN SET THE CURSOR BY ISSUING
000007 1163   A SETCOORD BEFORE CALLING THIS PROCEDURE. BEWARE! IF
000007 1164   YOU WANT TO ERASE A LINE, DRAW THE LINE EXACTLY THE
000007 1165   SAME WAY YOU DID IT ORIGINALLY (BUT NOW IN ERASE MODE)
000007 1166   THIS IS BECAUSE THE LINE DRAWING ROUTINE IN ROM IN THE
000007 1167   ORION WILL DRAW LINES DIFFERENTLY IF THE START AND END
000007 1168   POINTS ARE INTERCHANGED. FINALLY, THE ONLY VALID WRITING
000007 1169   MODES FOR LINES ARE ALLSNORM AND ERASE.
000007 1170 *)
000007 1171
000007 1172 (* INTERNAL DESCRIPTION:
000007 1173   SEND OUT THE 3 CHARACTER ADDRESS SEQUENCE AS FOLLOWS:
000007 1174
000007 1175   CHARACTER 1: /1/X8IX7IX6IX5IX4IX3/
000007 1176   CHARACTER 2: /1/X2IX1IX0IX8IX7IX6/
000007 1177   CHARACTER 3: /1/Y5Y4Y3Y2Y1Y0/
000007 1178
000007 1179   THIS SEQUENCE IS THE SAME FOR DRAWPOINT.
000007 1180 *)
000007 1181
000007 1182
000007 1183 VAR
000007 1184   X,Y : INTEGER;
000011 1185   X1 : INTEGER;
000012 1186   X2ANDY1 : INTEGER;
000013 1187   Y2 : INTEGER;
000014 1188

```

```

000014 1189 BEGIN
000014 1190   CONVNETRIC(X3,Y3,M,X,Y);
000010 1191   MODF(GRAPHICS,LINE,WMODE,PAGE);
000016 1192   X1 := (X DIV 8) + 64;
000021 1193   PUTCH(OUTPUT,X1);
000023 1194   X2ANDY1 := 64 + (X MOD 8)*8 + (Y DIV 64);
000030 1195   PUTCH(OUTPUT,X2ANDY1);
000032 1196   Y2 := (Y MOD 64) + 64;
000036 1197   PUTCH(OUTPUT,Y2);
000040 1198 END;
000066 1199
000066 1200 (*=====*)
000066 1201 PROCEDURE DRAWPOINT (
000002 1202   X3,Y3 : INTEGER; (* WHERE THE POINT GOES *)
000004 1203   M : METRIC; (* DOIS, CHS, OR PARTS *)
000005 1204   WMODE : INTEGER; (* WRITE MODE *)
000006 1205   PAGE : INTEGER (* SCREEN PAGE FLAG *)
000007 1206 );
000007 1207
000007 1208 (* USER DESCRIPTION:
000007 1209   DRAW A POINT AT X3,Y3. ONLY ESPNORM AND FRASE MAKE
000007 1210   SENSE FOR WMODE.
000007 1211 *)
000007 1212
000007 1213 (* INTERNAL DESCRIPTION:
000007 1214   SEE DRAWLINE.
000007 1215 *)
000007 1216
000007 1217 VAR
000007 1218   X,Y : INTEGER;
000011 1219   X1 : INTEGER;
000012 1220   X2ANDY1 : INTEGER;
000013 1221   Y2 : INTEGER;
000014 1222
000014 1223 BEGIN
000014 1224   CONVNETRIC(X3,Y3,M,X,Y);
000010 1225   MODF(GRAPHICS,POINT,WMODE,PAGE);
000016 1226   X1 := (X DIV 8) + 64;
000021 1227   PUTCH(OUTPUT,X1);
000023 1228   X2ANDY1 := 64 + (X MOD 8)*8 + (Y DIV 64);
000030 1229   PUTCH(OUTPUT,X2ANDY1);
000032 1230   Y2 := (Y MOD 64) + 64;
000036 1231   PUTCH(OUTPUT,Y2);
000040 1232 END;
000066 1233
000066 1234
000066 1235
000066 1236 (*=====*)
000066 1237 PROCEDURE PUTCHAR (
000002 1238   CH : INTEGER; (* OFD OF CHARACTER TO DRAW *)
000003 1239   CHMODE : MEMORY; (* NORMAL,PROGRAMMABLE *)
000004 1240   WMODE : INTEGER; (* WRITE MODE, SEE MODE *)
000005 1241   PAGE : INTEGER (* CLEAR SCREEN FLAG, SEE MODE *)
000006 1242 );
000006 1243
000006 1244 (* USER DESCRIPTION:
000006 1245   THIS PROCEDURE "DRAWS" A CHARACTER AT THE CURRENT CURSOR

```



```

000006 1246 POSITION. YOU PASS THE INTEGER ORD OF THE CHARACTER
000006 1247 TO BE DRAWN INSTEAD OF THE CHARACTER ITSELF SINCE THIS
000006 1248 PROCEDURE CAN ALSO DRAW IN THE ALTERNATE CHARACTER SET
000006 1249 AS SPECIFIED BY THE PARAMETER MEMORY. I THOUGHT IT WOULD
000006 1250 BE MUCH MORE USEFUL TO MAP ALTERNATE CHARACTERS TO
000006 1251 INTEGERS AS OPPOSED TO MAPPING THEM TO THE STANDARD
000006 1252 ASCII CHARACTER SET. YOU SHOULD USE THIS ROUTINE TO
000006 1253 DISPLAY THE ALTERNATE CHARACTERS WHOSE ORDS ARE < 32
000006 1254 SINCE THESE ARE NOT ACCESSABLE IN TEXT MODE. (IF YOU
000006 1255 ATTEMPT TO DO THIS, THEY WILL BE INTERPRETED AS BEING
000006 1256 ASCII CONTROL CHARACTERS BY THE ORION.) NOTE: FOR NORMAL
000006 1257 PRINTING OF TEXT, YOU DON'T HAVE TO USE THIS ROUTINE.
000006 1258 IN FACT, THIS ROUTINE IS SLOWER (BY A FACTOR OF 3)
000006 1259 THAN DOING A SIMPLE WRITE.
000006 1260 *)
000006 1261
000006 1262 (* INTERNAL DESCRIPTION:
000006 1263 THIS PROCEDURE SPTS OUT 3 CHARACTER CHARACTER PLOT
000006 1264 SEQUENCES PER CHARACTER PLOTTED. THE FIRST CHARACTER
000006 1265 IS A "CONTROL ACCESS" ESCAPE CHARACTER IMPLING THE
000006 1266 ORION THAT THE FOLLOWING CHARACTER IS FOR CONTROL.
000006 1267 THE SECOND CHARACTER IS A CONTROL CHARACTER THAT
000006 1268 SELECTS THE CHARACTER FONT MEMORY. THERE ARE FOUR
000006 1269 POSSIBILITIES HERE: LOW ROM, HIGH ROM, LOW RAM, AND
000006 1270 HIGH RAM. THE LAST CHARACTER IS THE CHARACTER TO BE
000006 1271 PLOTTED WITH ITS MSB STRIPPED. (THE MSB IS INCORPORATED
000006 1272 IN THE MEMORY SELECT CHARACTER). KLUDGY, HEH? I DID
000006 1273 NOT IMPLIMENT ALL THE FEATURES OF THE "CONTROL ACCESS"
000006 1274 ESCAPE SEQUENCE. THE ORION ALLWS FOR MOST OF THE ASCII
000006 1275 CONTROL CHARACTERS THAT HAVE MEANING TO THE ORION TO
000006 1276 BE ALSO CARRIED OUT BY THIS ESCAPE SEQUENCE. SINCE
000006 1277 ONE IS IN GRAPHICS MODE, ONE DOES NOT HAVE TO GET OUT
000006 1278 OF GRAPHICS TO MOVE DOWN A LINE, SAY. EOT, $X$, WHAT
000006 1279 A PAIN IN THE $' = !!! ALSO, IF ONE REMEMBERS WHAT
000006 1280 MEMORY WAS LAST SELECTED, THEN ONE COULD AVOID SENDING
000006 1281 THE ESCAPE SEQUENCE. THIS WOULD ALLOW FOR 3 CHARACTERS
000006 1282 TO BE DRAWN, BUT COULD YOU IMAGINE THE TROUBLE IF I
000006 1283 TRIED TO MAINTAIN THIS CAPABILITY AT THIS LEVEL OF
000006 1284 SOFTWARE! ENOUGH BITCHING, THE 3 CHARACTER SEQUENCE
000006 1285 IS AS FOLLOWS:
000006 1286
000006 1287 CHARACTER 1 : /1/C5C4C3C2C1C0/ I USE THIS TO SEND OUT A CONTROL ACCESS
000006 1288 CHARACTER, HEX 7F.
000006 1289 CHARACTER 2 : /1/C5C4C3C2C1C0/ I USE THIS TO SEND OUT A SELECT MEMORY
000006 1290 CHARACTER, WHFFF:
000006 1291 C5..0 = 16 - LOW ROM
000006 1292 17 - HIGH ROM
000006 1293 18 - LOW RAM
000006 1294 19 - HIGH RAM
000006 1295 CHARACTER 3 : /1/C5C4C3C2C1C0/ THE CHARACTER WITH MSB STRIPPED.
000006 1296
000006 1297 *)
000006 1298
000006 1299
000006 1300 VAR
000006 1301 ST : INTEGER;
000007 1302

```

```

000007 1303 BEGIN
000007 1304   IF CHMODE = NORMAL THEN ST := 16 ELSE ST := 18;
000011 1305   MODE(GRAPHICS,CHPLOT,WMODE,PAGE);
000017 1306   IF CH >= 64 THEN BEGIN
000022 1307     PUTCH(OUTPUT,DELETE);
000024 1308     PUTCH(OUTPUT,ST + 64 + 1);
000030 1309     PUTCH(OUTPUT,CH);
000032 1310   END
000032 1311   ELSE BEGIN
000033 1312     PUTCH(OUTPUT,DELETE);
000035 1313     PUTCH(OUTPUT,ST + 64);
000040 1314     PUTCH(OUTPUT,CH + 64);
000043 1315   END;
000043 1316 END;
000057 1317
000057 1318
000057 1319 (*-----*)
000057 1320 PROCEDURE DRAWCHAR (
000002 1321   CH : INTEGER;      (* INTEGER ORD OF CHARACTER TO BE DRAWN *)
000003 1322   X1,Y1 : INTEGER;  (* WHERE TO DRAW CHARACTER *)
000005 1323   M : METRIC;      (* DOIS, CHS, OR PARTS *)
000006 1324   CENTERING : ADJ;  (* LL,LR,UL,UR, OR CENTER *)
000007 1325   CHMODE : MEMORY;  (* NORMAL, PROGRAMMABLE *)
000010 1326   WMODE : INTEGER;   (* WRITE MODE, SEE MODE *)
000011 1327   PAGE : INTEGER     (* CLEAR SCREEN FLAG, SEE MODE *)
000012 1328 );
000012 1329
000012 1330 (* USER DESCRIPTION:
000012 1331   DRAW A CHARACTER WHOSE ORD IS CH AT POINT X,Y. THE
000012 1332   REFERENCE POINT IS SPECIFIED BY CENTERING. FOR NORMAL
000012 1333   CHARACTER PLOTTING, SUGGEST USING CENTER FOR CENTERING.
000012 1334   YOU CAN ALSO PLOT PROGRAMMABLE CHARACTERS AS SPECIFIED
000012 1335   BY MEMORY. THIS ROUTINE IS MEANT TO BE A BETTER USER
000012 1336   INTERFACE THAN PUTCHAR. SEE COMMENTS OF PUTCHAR.
000012 1337 *)
000012 1338
000012 1339 (* INTERNAL DESCRIPTION:
000012 1340   THIS ROUTINE SIMPLY CALLS SETCOORD BEFORE CALLING
000012 1341   PUTCHAR.
000012 1342 *)
000012 1343
000012 1344
000012 1345 VAR
000012 1346   X,Y : INTEGER;
000014 1347   XS,YS : INTEGER;
000016 1348
000016 1349 BEGIN
000016 1350   CONVMETRIC(X1,Y1,M,X,Y);
000011 1351   CENTERIT(X,Y,16,CENTERING,XS,YS);
000021 1352   SETCOORD(XS,YS,DOIS);
000026 1353   PUTCHAR(CH, CHMODE, WMODE, PAGE);
000033 1354 END;
000065 1355
000065 1356
000065 1357 (*-----*)
000065 1358 PROCEDURE LOADCHRS (
000002 1359   CHS : ROM;      (* DATA MATRIX OF CHARACTERS *)

```

```

000003 1360          START : INTEGER;  (* START ADDRESS IN ORION TO START PLACING CHARACTERS *)
000004 1361          NUM : INTEGER  (* NUMBER OF CHARACTERS *)
000005 1362          );
000605 1363
000605 1364  (* USER DESCRIPTION:
000605 1365   PROGRAM THE PROGRAMMABLE MEMORY IN THE ORION. THE CHARACTERS
000605 1366   ARE IN THE MATRIX CHS. THIS MATRIX IS [# OF CHARS, 8]
000605 1367   THE VALUES IN THIS MATRIX REPRESENT ONE VERTICAL SLICE
000605 1368   OF A CHARACTER. CHARACTERS ARE IN A 1618 FORMAT. THUS,
000605 1369   THE LARGEST VALUE THAT SHOULD BE IN CHS IS (2**17)-1.
000605 1370   THE PROCEDURE GETSET IS USED TO LOAD MATRICES OF THIS TYPE
000605 1371   FROM AN EXTERNAL FILE. YOU CAN SPECIFY WHERE THE PROGRAMMABLE
000605 1372   CHARACTER SET IS TO START IN RAM OF THE ORION BY START.
000605 1373   YOU MAY WANT TO DO THIS TO BIAS THE START OF THE PROGRAMMABLE
000605 1374   CHARACTER SET TO ABOVE THE CONTROL CHARACTERS.
000605 1375 *)
000605 1376
000605 1377  (* INTERNAL DESCRIPTION:
000605 1378   SPIT OUT THE LOAD ADDRESS 3 CHARACTER SEQUENCE, THEN
000605 1379   PROCEED TO SEND OUT LOAD CHARACTER SEQUENCES UNTIL ALL
000605 1380   CHARACTERS ARE TRANSFERRED. NOTE THAT YOU LOAD 1 COLUMN
000605 1381   AT A TIME. TO THE ORION, EVERYTHING IS IN TERMS OF THESE
000605 1382   VERTICAL SLICES, INCLUDING THE START ADDRESS. I CONVERT
000605 1383   FOR YOU THE START ADDRESS IN TERMS OF CHARACTERS TO ORION'S TERMS.
000605 1384   THE 3 CHARACTER LOAD ADDRESS SEQUENCE IS AS FOLLOWS:
000605 1385
000605 1386       CHARACTER 1 : /0/100/XXX/
000605 1387       CHARACTER 2 : /1/XX/A9A8A7A6/
000605 1388       CHARACTER 3 : /1/A5A4A3A2A1A0/ WHERE A9..0 IS THE 10 BIT COLUMN ADDRESS
000605 1389                   TO START LOADING TO.
000605 1390
000605 1391   THE 3 CHARACTER COLUMN LOAD SEQUENCE IS AS FOLLOWS:
000605 1392
000605 1393       CHARACTER 1 : /1/XX/B16B15B14B13/
000605 1394       CHARACTER 2 : /1/B12B11B10B9B8B7/
000605 1395       CHARACTER 3 : /1/B6B5B4B3B2B1/  WHERE B16..1 ARE THE 16 BITS OF A COLUMN.
000605 1396
000605 1397 *)
000605 1398
000605 1399
000605 1400  VAB
000605 1401      ST, ST1, ST2 : INTEGER;
000610 1402      C1, C2, C3 : INTEGER;
000613 1403      T : INTEGER;
000614 1404      I,J : INTEGER;
000616 1405
000616 1406  BEGIN
000616 1407      MODE (GRAPHICS, CHLOAD, ERASE, DSPNO);
000016 1408      (*LOAD ADDRESS*)
000016 1409      PUTCH (OUTPUT, 32);          (* ASCII SPACE *)
000020 1410      ST := START * 8;
000023 1411      ST1 := (ST DIV 64) * 64;
000025 1412      PUTCH (OUTPUT, ST1);
000027 1413      ST2 := (ST MOD 64) * 64;
000033 1414      PUTCH (OUTPUT, ST2);
000035 1415
000035 1416      (*TRANSFER ROM*)

```

```

000035 1417     FOR I := 0 TO NUM DO BEGIN
000042 1418         FOR J := 0 TO 7 DO BEGIN
000046 1419             T := CHS[I,J];
000061 1420             C1 := (T DIV 4096) + 64;
000063 1421             PUTCH(OUTPUT,C1);
000065 1422             C2 := ((T DIV 64) MOD 64) + 64;
000072 1423             PUTCH(OUTPUT,C2);
000074 1424             C3 := (T MOD 64) + 64;
000100 1425             PUTCH(OUTPUT,C3);
000112 1426         END
000102 1427     END;
000113 1428 END;
000143 1429
000143 1430
000143 1431 (*-----*)
000143 1432     PROCEDURE GETLN (
000002 1433         VAR INFILE : TEXT;      (* INPUT FILE *)
000003 1434         VAR CH : INBUF; (* ARRAY TO PLACE LINE IN *)
000004 1435         VAR LEN : INTEGER;      (* LENGTH OF LINE *)
000005 1436         VAR DONE : BOOLEAN      (* HIT EOF FLAG *)
000006 1437     );
000006 1438
000006 1439     (* USER DESCRIPTION:
000006 1440     THIS UTILITY PROCEDURE GETS A LINE FROM THE TEXT FILE
000006 1441     INFILE. IT PLAC'S THE TEXT IN THE ARRAY INBUF AND SETS
000006 1442     LEN TO THE LENGTH OF THE FILE. NOTE THAT IF THE LINE IS LARGER
000006 1443     THAN THE BUFFER SIZE, THE REST OF THE LINE CAN BE
000006 1444     STILL RETRIEVED BY CALLING GETLN AGAIN. FOR THIS
000006 1445     REASON, YOU ARE RESPONSIBLE FOR ISSUING A READLN
000006 1446     AFTER CALLING GETLN. DONE IS SET TRUE IF THE EOF IS
000006 1447     REACHED.
000006 1448     *)
000006 1449
000006 1450     (* INTERNAL DESCRIPTION:
000006 1451     NOTHING SPECIAL GOING ON HERE. THIS ROUTINE USES THE
000006 1452     GLOBAL CONSTANT, BUFLN, TO TELL WHEN TO QUIT STUFFING
000006 1453     THE BUFFER.
000006 1454     *)
000006 1455
000006 1456
000006 1457     BEGIN
000006 1458         LEN := 1;
000006 1459         DONE := EOF(INFILE);
000006 1460         IF NOT DONE THEN BEGIN
000013 1461             IF NOT EOLN(INFILE) THEN BEGIN
000014 1462                 READ(INFILE,CH[LEN]);
000035 1463                 WHILE(NOT (EOLN(INFILE) OR (LEN = BUFLN))) DO BEGIN
000042 1464                     LEN := LEN + 1;
000043 1465                     READ( INFILE, CH[LEN]);
000064 1466                 END;
000065 1467             END
000065 1468             ELSE BEGIN
000066 1469                 LEN := 0;
000067 1470             END;
000067 1471         END;
000067 1472     END;
000075 1473

```

```
000075 1474 (*-----*)
000075 1475 PROCEDURE GETSET (
000075 1476     VAR CHTEXT : TEXT;    (* WHICH FILE THE CHARACTER SET IS FROM *)
000075 1477     VAR CHARSET : ROM;    (* THE MATRIX TO PLACE THE CHARACTER SET *)
000075 1478     VAR CHNO : INTEGER    (* THE NUMBER OF CHARACTERS READ IN *)
000075 1479 );
000075 1480
000075 1481
000075 1482 (* USER DESCRIPTION:
000075 1483 THIS PROCEDURE FETCHES THE CHARACTER SET LIVING ON THE
000075 1484 TEXT FILE CHTEXT AND PLACES IT IN THE MATRIX CHARSET. CHARSET IS
000075 1485 A 128X8 ARRAY. EACH COLUMN OF THIS ARRAY REPRESENTS ONE OF THE 8
000075 1486 COLUMNS OF A CHARACTER. EACH ARRAY ELEMENT IS A ROW OF
000075 1487 A CHARACTER. THE PARAMETER CHNO TELLS YOU HOW MANY CHARACTERS
000075 1488 WAS LOADED INTO CHARSET. THE STANDARD ALTERNATE CHARACTER
000075 1489 SET IS ILLUSTRATED BELOW. INCREASING INDICES OF THE
000075 1490 CHARSET MATRIX IS ACROSS THE PAGE:
000075 1491 *)
```

```

000005 1492 (*$L'ALTERNATE SET**)
000005 1493 (*)
000005 1494
000005 1495
000005 1496      00      00      00      00      0000      0      00      00      000      000
000005 1497      0      0      0      0      0      0      0      0      0      0      0
000005 1498 000000      0      0      0      0      0      0      0      0      0      0      0
000005 1499      0      00000      00      0      0      0      00      00      0      00000      0      0      0
000005 1500      0      0      0      0      0      0      0      0      0      0      0      00      000      0000      000
000005 1501      0      0      0      0      0      0      0      0      0      0      0      00000      0
000005 1502      0      0      0      0      0      0      0      0      000000      0      0      0      0
000005 1503      0      0      0      0      0      0      0      0      0      00      0      0
000005 1504      0      0      00      00      00      0      0      00      0      0000      0
000005 1505      0      0      0      0      00      0      0      0      0
000005 1506      00      00      00      0      0
000005 1507      0000
000005 1508
000005 1509
000005 1510
000005 1511
000005 1512      0 0      0000      0      0000      0000      00      0      00      00      000000
000005 1513      0 0      0      0      0      0 0      0 0      00      00      0      0      00      0      000000
000005 1514      0000      000      000      0      00      000      0      0      00000      0      0      0      000000      00 00
000005 1515      0      0      0      0      0      0 0      0      0      0      0      0      0      0      0      0
000005 1516      0      000      00      0      0000      0      0      0      0      00000      0      0      0      0      0
000005 1517      0      0      0      0      0      0      0      0      0      0      0      0      0      00      00
000005 1518      0      0      00000      0      0      0      0      00
000005 1519      0      0      0      0      0      0      0      0
000005 1520      000 000      0      00000      0      0      0
000005 1521      00      0      0
000005 1522      000000
000005 1523
000005 1524
000005 1525
000005 1526
000005 1527
000005 1528      0 000 0      000      00      0      0      0      0      0      0      00      000000      000
000005 1529      0 0 0      0      0      00      0      0      0      0      0      0      0      0      0
000005 1530      0 0 0      000      0      0      000      0      0      0      0      0      0      0      0
000005 1531      0 0 0      0 0 0      0      0      0 0      0 0      0      0      0      0      0
000005 1532      000      0 0 0      0      0      0      0      0      0      0      0      0      0      0
000005 1533      0      0 0 0      000000      0      0      0      0      00000      0 0 0      00 0      0      0      0
000005 1534      0      000      0      0      0      0      0      0      000      000      0 0 0      0      0      0
000005 1535      0      0      0      0      0      0      0      0      0      0      0      0      0      0      0
000005 1536      000      000      0      000 000      0      0      00000      0      0      0      0      0      0
000005 1537      0      0      0      0      0      0      0      0      0      0      0      0      0      0      0
000005 1538      0      0      0      0      0      0      0      0      0      0      0      0      0      0      0
000005 1539      0      0      0      0      0      0      0      0      0      0      0      0      0      0      0
000005 1540
000005 1541
000005 1542 *)

```

[illegible]

000005 1594 (0

[illegible]


```

000005 1629 (*$L'LOW LEVEL ROUTINES'*)
000005 1630
000005 1631
000005 1632 (* INTERNAL DESCRIPTION:
000005 1633 DO A RESET ON THE TEXT FILE CHTEXT, THEN READ IN CHARACTERS
000005 1634 IN THIS FORM FROM THE TEXT FILE:
000005 1635
000005 1636
000005 1637
000005 1638
000005 1639
000005 1640
000005 1641 000000
000005 1642 0
000005 1643 0
000005 1644 127 NOT
000005 1645
000005 1646
000005 1647
000005 1648 00000
000005 1649 00 0
000005 1650 0 0
000005 1651 0 0
000005 1652 0 0
000005 1653 0 0
000005 1654 128 PI
000005 1655
000005 1656
000005 1657
000005 1658 00
000005 1659 0 0
000005 1660 0 0
000005 1661 0 0
000005 1662 0 0
000005 1663 00
000005 1664 129 CIRCLE
000005 1665
000005 1666
000005 1667 THE LINES WITH THE NUMBERS AND IDENTIFIERS ON THEM ARE
000005 1668 IGNORED. I GOT THIS CHARACTER SET FROM THE DCS UNIX
000005 1669 SYSTEM AND IT WAS USED FOR THE GOULD PICTER. IN THE
000005 1670 FUTURE THESE NUMBERS COULD BE USED AS INDICES INTO THE
000005 1671 ARRAY CHSET, BUT THIS WAS A QUICK AND DIRTY IMPLEMENTATION.
000005 1672 NOTE THAT THE ROWS OF THE CHARACTER ARE STORED IN CHSET
000005 1673 AS BINARY BITS TO A 16 BIT WORD.
000005 1674 *)
000005 1675 VAR
000005 1676 I, J, K, L, M, MM : INTEGER;
000013 1677 DONE : BOOLEAN;
000014 1678 LFN : INTEGER;
000015 1679 CH : INTRUF;
000033 1680
000033 1681 BEGIN
000033 1682 CHNO := -1;
000006 1683 K := -1;
000007 1684 FOR I := 0 TO 127 DO BEGIN
000013 1685 FOR J := 0 TO 7 DO BEGIN

```

```

000017 1686      CHARSET[I,J] := 0;
000033 1687      END;
000037 1688  END;
000043 1689  RESET(CHTEXT);
000045 1690  GETLN(CHTEXT,CH,LEN,DONE);
000051 1691  READLN(CHTEXT);
000054 1692  WHILE NOT DONE DO BEGIN
000056 1693    IF (CH[1] = ' ') OR (CH[1] = '0') OR (LEN = 0) THEN BEGIN
000065 1694      K := K + 1;
000067 1695      IF LEN > 0 THEN BEGIN
000071 1696        IF LEN > 8 THEN LEN := 8;
000073 1697        FOR J := 1 TO LEN DO BEGIN
000107 1698          DSSPTR^.BUILD[15-K,J-1] := CH[J];
000125 1699        END;
000131 1700      END;
000131 1701  END
000131 1702  ELSE BEGIN
000132 1703    IF K > 0 THEN BEGIN
000134 1704      CHNO := CHNO + 1;
000136 1705      FOR L := 0 TO 7 DO BEGIN
000142 1706        MM := 1;
000144 1707        FOR M := 0 TO 15 DO BEGIN
000147 1708          IF DSSPTR^.BUILD[M,L] = '0' THEN CHARSET[CHNO,L] := CHARSET[CHNO,L] + MM;
000210 1709          MM := MM * 2;
000212 1710          DSSPTR^.BUILD[M,L] := ' ';
000223 1711        END;
000227 1712      END;
000233 1713    END;
000233 1714    K := -1;
000235 1715  END;
000235 1716  GETLN(CHTEXT,CH,LEN,DONE);
000241 1717  IF NOT DONE THEN READLN(CHTEXT);
000246 1718  END
000246 1719  END;
000273 1720
000273 1721
000273 1722  (*-----*)
000273 1723  PROCEDURE DRAWBOX (
000002 1724    X3,XL,Y3,YL : INTEGER;  (* THE ORIGIN LENGTH OF THE BOX *)
000006 1725    M1,M2 : METRIC;  (* SCALE FOR ORIGIN LENGTHS *)
000010 1726    CENTERING : ADJ;  (* LL,UI,LR,UR & CENTER *)
000011 1727    THICKNESS : INTEGER;  (* THICKNESS OF BORDER
000012 1728      - GROWS OUTWARD
000012 1729      + GROWS INWARD
000012 1730      > 512 FILL AREA IN COMPLETELY *)
000012 1731    WMODE : INTEGER;  (* WRITE MODE, SEE MODE *)
000013 1732    PAGE : INTEGER  (* CLEAR SCREEN FLAG, SEE MODE *)
000014 1733  );
000014 1734
000014 1735  (* USER DESCRIPTION :
000014 1736  THIS PROCEDURE DRAWS RECTANGLES THAT HAVE AN ORIGIN AS SPECIFIED
000014 1737  BY X3,Y3 AND THE PARAMETER CENTERING. CENTERING TILLS
000014 1738  YOU WHERE THE POINT X3,Y3 IS RELATIVE TO THE RECTANGLE. THE
000014 1739  LENGTHS OF THE RECTANGLE IS SPECIFIED BY XL,YL. THE
000014 1740  THICKNESS OF THE DRAWN BORDERS OF THE RECTANGLE IS SPECIFIED
000014 1741  BY THICKNESS. IF THICKNESS IS A POSITIVE NUMBER, THEN
000014 1742  THE BORDER IS GROWN INWARDS FROM THE EDGE OF THE RECTANGLE.
  
```

```

000014 1743     IF THICKNESS IS NEGATIVE, THEN THE BOARDER IS GROWN
000014 1744     OUTWARDS FROM THE EDGE OF THE RECTANGLE. IF THICKNESS
000014 1745     IS > 512 THEN THE RECTANGLE IS FILLED COMPLETELY, I.E.
000014 1746     YOU WANT A SOLID FIGURE.
000014 1747     *)
000014 1748
000014 1749     (* INTERNAL DESCRIPTION:
000014 1750     NOTHING SPECIAL ABOUT THIS ROUTINE.
000014 1751     *)
000014 1752
000014 1753
000014 1754     VAR
000014 1755     X,XLEN,Y,YLEN : INTEGER;
000020 1756     IS,YS,X1,X2,Y1,Y2,I,INC,T : INTEGER;
000031 1757
000031 1758     BEGIN
000031 1759     CONVNETRIC(X3,Y3,M1,X,Y);
000031 1760     CONVNETRIC(XL,YL,M2,XLEN,YLEN);
000017 1761     CENTERIT(X,XLEN,Y,YLEN,CENTERING,XS,YS);
000030 1762     IF PAGE = DSPYES THEN DRAWPOINT(XS,YS,DOTS,WMODE,DSPYES);
000037 1763     IF THICKNESS > 512 THEN BEGIN (* FILL IN RECTANGLE COMPLETELY*)
000042 1764     FOR I := 0 TO (XLEN - 1) DO BEGIN
000046 1765     SETCOORD(XS + I, YS,DOTS);
000053 1766     DRAWLINE(XS + I, YS + YLEN - 1,DOTS, WMODE, DSPNO);
000062 1767     END;
000067 1768     END (*OF FULL RECTANGLE*)
000067 1769     ELSE BEGIN
000070 1770     IF THICKNESS > 0 THEN INC := 1 ELSE INC := -1;
000075 1771     T := ABS(THICKNESS);
000100 1772     X1 := XS;
000102 1773     Y1 := YS;
000103 1774     X2 := XS + (XLEN - 1);
000105 1775     Y2 := YS + (YLEN - 1);
000110 1776     FOR I := 1 TO T DO BEGIN
000114 1777     SETCOORD(X1,Y1,DOTS);
000121 1778     DRAWLINE(X2,Y1,DOTS,WMODE,DSPNO);
000127 1779     DRAWLINE(X2,Y2,DOTS,WMODE,DSPNO);
000135 1780     DRAWLINE(X1,Y2,DOTS,WMODE,DSPNO);
000143 1781     DRAWLINE(X1,Y1,DOTS,WMODE,DSPNO);
000151 1782     X1 := X1 + INC;
000153 1783     Y1 := Y1 + INC;
000155 1784     X2 := X2 - INC;
000156 1785     Y2 := Y2 - INC;
000160 1786     END
000160 1787     END;
000160 1787     END (*OF PARTIAL FILL CASE*)
000164 1788
000244 1789
000244 1790
000244 1791     PROCEDURE CREATETARG (
000002 1792     BLOCKID : INTEGER; (* BLOCK FOR TARGET *)
000003 1793     ID : INTEGER; (* USER SPECIFIED TARGET ID *)
000004 1794     X2,XL2 : INTEGER; (* X ORIGIN AND LENGTH OF TARGET *)
000006 1795     Y2,YL2 : INTEGER; (* Y ORIGIN AND LENGTH *)
000010 1796     M1,M2 : METRIC; (* METRICS FOR THE ABOVE DIMENSIONS *)
000012 1797     CENTERING : ADJ; (* LL,UL,LR,UR,CENTER *)
000013 1798     AUTOMAKY : BOOLEAN; (* TRUE IF TARGET SIZE IS TO BE MADE TO FIT THE LABEL *)
000014 1799     AUTOINCREMENT : BOOLEAN; (* TRUE IF THE PLACEMENT OF THE TARGET IS TO BE MADE AUTOMATIC *)

```

```

000015 1800          STYLE : TARGTYPE;      (* PAT, UNDERLINE *)
000016 1801          LABL : LABELS;          (* THE ARRAY WITH THE TARGET LABEL *)
000017 1802          LBLN : INTEGER          (* THE LENGTH OF THE LABEL *)
000020 1803          );
000027 1804
000027 1805      VAR
000027 1806          PTR, PTR1 : TPTR;
000031 1807          I, J : INTEGER;
000033 1808          X, XLN, Y, YLN, XS, YS : INTEGER;
000041 1809          XBOX, YBOX, XCENTER, YCENTER : INTEGER;
000045 1810          XBOX1, YBOX1, X1, Y1, XL, YL : INTEGER;
000053 1811          OUTLINE : INTEGER;
000054 1812
000054 1813      BEGIN
000054 1814          PTR1 := DSSPTR^.BLK[BLOCKID].TARGS^[1];
000025 1815          I := 1;
000027 1816          WHILE (I < TARGMAX) AND (PTR1 <> NIL) DO BEGIN
000034 1817              I := I + 1;
000036 1818              PTR1 := DSSPTR^.BLK[BLOCKID].TARGS^[I];
000055 1819          END;
000056 1820          IF PTR1 = NIL THEN BEGIN
000060 1821              NEW(PTR);
000062 1822              DSSPTR^.BLK[BLOCKID].TARGS^[I] := PTR;
000102 1823
000102 1824              CONVMETRIC(X2, Y2, M1, X, Y);
000110 1825              CONVMETRIC(XL2, YL2, M2, XLN, YLN);
000116 1826              (* CENTERIT(X, XLN, Y, YLN, CENTERING, IS, YS); *)
000116 1827
000116 1828          (* FIND OUT THE DIMENSIONS OF THE RECTANGLE *)
000116 1829          IF AUTOINCREMENT THEN BEGIN
000120 1830              XCENTER := DSSPTR^.BLK[BLOCKID].TARGPX + (16 + 7);
000133 1831              YCENTER := DSSPTR^.BLK[BLOCKID].TARGPY - (16 + 7);
000144 1832          END
000144 1833          ELSE BEGIN
000145 1834              XCENTER := X;
000147 1835              YCENTER := Y;
000151 1836          END;
000151 1837          XBOX := (XCENTER + 16) DIV 32;
000151 1838          YBOX := (YCENTER - 16) DIV 32;
000155 1839
000157 1840          (* COMPUTE SIZE IN X DIRECTION *)
000157 1841          XL := CNTCHARS(LABL, LBLN);
000162 1842          XBOX1 := (XL DIV 4) + 1;      (* # OF BOXES *)
000165 1843
000165 1844          (* COMPUTE SIZE IN Y DIRECTION *)
000165 1845          YL := 1;
000166 1846          YBOX1 := (YL DIV 2) + 1;
000170 1847
000170 1848          (* CHECK IF WE NEED TO GO TO A NEW LINE *)
000170 1849          IF AUTOINCREMENT THEN BEGIN
000171 1850              IF DSSPTR^.BLK[BLOCKID].OUTLINE < 0 THEN OUTLINE := 0 ELSE OUTLINE := - DSSPTR^.PLK[BLOCKID].OUTLINE;
000215 1851              IF ((XBOX + XBOX1) * 32 + (7 - 1)) >= (DSSPTR^.BLK[BLOCKID].X? + OUTLINE) THEN BEGIN
000232 1852                  DSSPTR^.BLK[BLOCKID].TEXFY := YBOX * 32 - (32 - 7);
000244 1853                  YBOX := YBOX - 2;
000246 1854                  DSSPTR^.BLK[BLOCKID].TARGPX := DSSPTR^.BLK[BLOCKID].TEXPX;
000265 1855                  XBOX := (DSSPTR^.PLK[BLOCKID].TARGPX + (32 + 7)) DIV 32;

```

```

000277 1857      END;
000277 1858      END;
000277 1859      PTR^.LORGX := (((32 * XBOX1) - (8 * XL)) DIV 2) + XBOX * 32;
000311 1860      PTR^.LORGY := (((32 * YBOX1) - (16 * YL)) DIV 2) + YBOX * 32;
000321 1861
000321 1862      (* DRAW A BOX AROUND AREA *)
000321 1863      DRAWBOX(XBOX * 32, XBOX1 * 32, YBOX * 32, YBOX1 * 32, DOTS, DOTS, LL, -2, DSPNORM, DSPNO);
000335 1864
000335 1865      (* DISPLAY THE LABEL *)
000335 1866      SETCOORD(PTR^.LORGX, PTR^.LORGY, DC15);
000350 1867      MODE(DSPTEXT, DSPNONE, DSPNORM, DSPNC);
000356 1868      FOR I := 1 TO LBLEN DO BEGIN
000363 1869          WRITE(OUTPUT, LABEL[I]);
000401 1870      END;
000406 1871      SETCOORD(10, 10, DOTS);
000411 1872
000411 1873      (* RESET TPXT POINTERS *)
000411 1874      IF AUTOINCREMENT THEN BEGIN
000413 1875          DSSPTR^.PLK[BLOCKID].TARGPX := (XBOX + XBOX1)*32 + ((32 - 1) - 7);
000427 1876      END;
000427 1877
000427 1878      (* FILL IN DATA STRUCTURES *)
000427 1879      PTR^.TOUCHED := FALSE;
000434 1880      PTR^.ID := ID;
000441 1881      PTR^.STYLE := STYLE;
000450 1882      PTR^.LABEL := LABEL;
000457 1883      PTR^.LBLEN := LBLEN;
000465 1884      PTR^.X := XBOX * 32;
000472 1885      PTR^.XLEN := XBOX1 * 32;
000500 1886      PTR^.Y := YBOX * 32;
000505 1887      PTR^.YLEN := YBOX1 * 32;
000512 1888      FOR I := 1 TO XBOX1 DO BEGIN
000516 1889          FOR J := 1 TO YBOX1 DO BEGIN
000523 1890              DSSPTR^.TARGARRAY[XBOX + I - 1, YBOX - (J - 1)].TARG := PTR;
000537 1891          END;
000544 1892      END;
000551 1893      END
000551 1894      ELSE BEGIN
000552 1895          END;
000552 1896      END;
000660 1897      (* OF CREATETARG *)
000660 1898
000660 1899      (*=====*)
000660 1900      PROCEDURE DSTTARG (
000660 1901          BLOCKID : INTEGER;      (* BLOCK IN WHICH TARGET LIES *)
000660 1902          TARGID : INTEGER      (* ID OF TARGET TO DESTROY *)
000660 1903      );
000660 1904
000660 1905      (* USER DESCRIPTION:
000660 1906      USE THIS PROCEDURE TO DESTROY TARGETS THAT HAVE BEEN
000660 1907      CREATED USING CREATETARG. SCRAPS DATA STRUCTURE FOR
000660 1908      TARGET AND REMOVES TARGET FROM THE SCREEN.
000660 1909      *)
000660 1910
000660 1911      VAR
000660 1912          PTR : TPTR;
  
```

```

000007 1914
000007 1915 BEGIN
000007 1916   I := 1;
000006 1917   WHILE (DSSPTR^.PLK[BLOCKID].TARGS^[I]^ID <> TARGID) DO BEGIN
000032 1918     I := I + 1;
000033 1919   END;
000034 1920   PTR := DSSPTR^.ELK[BLOCKID].TARGS^[I];
000053 1921
000053 1922   (* UNDO LABEL & BOX *)
000053 1923   SETCOORD(PTR^.LORGX,PTR^.LORGY,DOTS);
000066 1924   MOUF (ISPTEXT,DSPNONE,DSPNORM,DSPNO);
000074 1925   FOR I := 1 TO CNTCHARS(PTR^.LABL,PTR^.LPLEN) DO BEGIN
000114 1926     WRITE(OUTPUT,' ');
000121 1927   END;
000126 1928   DRAWBOX ( PTR^.X,PTR^.XLEN,
000137 1929     PTR^.Y,PTR^.YLEN,
000151 1930     DOTS,DOTS,LL,-5,
000161 1931     ERASE,DSPNO
000162 1932   );
000164 1933
000164 1934   (* SCRAP DATA STRUCTURE *)
000164 1935   FOR I := 0 TO 15 DO BEGIN
000170 1936     FOR J := 0 TO 15 DO BEGIN
000174 1937       IF ((DSSPTR^.TARGARRAY[I,J].TARG <> NIL) AND
000206 1938         (DSSPTR^.TARGARRAY[I,J].BLOCKID = BLOCKID)) THEN BEGIN
000220 1939         IF (DSSPTR^.TARGARRAY[I,J].TARG^.ID = TARGID) THEN BEGIN
000234 1940           DISPOSE (DSSPTR^.TARGARRAY[I,J].TARG);
000245 1941           DSSPTR^.TARGARRAY[I,J].TARG := NIL;
000257 1942         END;
000257 1943       END;
000257 1944     END;
000264 1945   END;
000270 1946 END;
000304 1947
000304 1948 (*-----*)
000304 1949 FUNCTION FETCH (
000002 1951   VAR PTR : TEXTPTR;      (* 2TUPLE POINTER TO CURRENT TEXT *)
000033 1952   VAR ENDT : BOOLEAN      (* INDICATES NO MORE TEXT IN BUFFERS *)
000094 1953 ) : CRNG;
000025 1954
000025 1955 (* THIS FUNCTION RETRIEVES A CHARACTER FROM THE TEXT BUFFERS
000035 1956   IT RESETS PTR AS INE EXTRACTS FROM THE BUFFERS *)
000095 1957
000005 1958 BEGIN
000005 1959   ENDT := FALSE;
000005 1960   PTR.POS := PTR.POS + 1;
000011 1961   IF PTR.POS > PTR.BUF^.EPOS THEN BEGIN (* GO TO NEXT BUFFER *)
000016 1962     PTR.BUF := PTR.BUF^.NEXT;
000023 1963     IF PTR.BUF <> NIL THEN BEGIN (* CHECK FOR END OF TEXT *)
000025 1964       PTR.POS := PTR.BUF^.POS + 1;
000034 1965       FETCH := PTR.BUF^.AR[PTR.POS];
000053 1966     END
000053 1967     ELSE BEGIN
000054 1968       ENDT := TRUE;
000055 1969       FETCH := 0;      (* ASCII NULL *)
000057 1970   END;

```

```

000057 1971      END
000057 1972      EISF BEGIN
000060 1973          FETCH := PTF.BUF^AR[PTB.FOS];
000077 1974      END;
000077 1975      END; (* OF FETCH *)
000104 1976
000104 1977      (* ***** *)
000104 1978      PROCEDURE DSPLYNR (
000002 1979          ID : INTEGER; (* BLOCK ID *)
000003 1980          PTR : TEXTPTR; (* POINTER TO A BUFFER TO DISPLAY *)
000004 1981          VAR PTR1 : TEXTPTR; (* PTR AFTER DISPLAYED LINE *)
000005 1982          DMODE : LINFMODE; (* ERASE OR PLACE *)
000006 1983          PX, PY : INTEGER (* POSITION TO START DISPLAY *)
000010 1984      );
000012 1985
000012 1986
000012 1987      VAR
000012 1988          X, Y, I, II : INTEGER;
000015 1989          TTYPE : TARGTYPE;
000017 1990          ENDT : BOOLEAN;
000020 1991          TARGID : INTEGER;
000021 1992          INDEX : INTEGER;
000022 1993          LARCNT : INTEGER;
000023 1994          CHR, CHR1 : INTEGER;
000025 1995          ENDOFLN : BOOLEAN;
000026 1996          WMODE : INTEGER;
000027 1997          PX1 : INTEGER;
000030 1998          SETAG : BOOLEAN;
000031 1999
000031 2000
000031 2001
000031 2002      BEGIN
000031 2003          PTR1 := PTR;
000031 2004          SETAG := FALSE;
000031 2005
000031 2006          IF DMODE <> CONSUME THEN BEGIN (* CASES THAT REQUIRE SEMANTIC ACTIONS *)
000016 2007              SETCOORD(PX, PY, DOTS); (* PLACE CURSOR AT RIGHT PLACE *)
000022 2008              CHR1 := FETCH(PTR1, ENDT);
000026 2009              IF DMODE = UNDO THEN BEGIN
000030 2010                  WMODE := ERASE;
000032 2011              END
000032 2012              ELSE BEGIN
000033 2013                  WMODE := DSSPTR^.BLK[ID].WMODE;
000045 2014              END;
000045 2015              ENDOFLN := FALSE;
000047 2016              PX1 := PX;
000051 2017              WHILE ((CHR1 <> EL) AND
000053 2018                  (CHR1 <> LI) AND
000056 2019                  (NOT ENDOFLN)) DO BEGIN
000061 2020                  IF CHR1 < STSPEC THEN BEGIN (* JUST DISPLAY IT *)
000063 2021                      IF PX1 <= DSSPTR^.BLK[ID].TEXTENDX THEN BEGIN (* OK, ON SAME LINE *)
000075 2022                          IF SETAG THEN SETCOORD(PX1, PY, DOTS); (* GO BACK TO WHERE WE WERE IF NEED BE *)
000102 2023                          SETAG := FALSE;
000104 2024                          IF DMODE <> UNDO THEN BEGIN
000106 2025                              MODE(DSPTEXT, DSPNONE, WMODE, DSPNO);
000114 2026                              PUTCH(OUTPUT, CHR1);
000116 2027                          END

```

```

000116 2028      ELSE BEGIN
000117 2029          MODE(DSPTEXT,DSPNCNE,DSPNORM,DSPNO);
000125 2030          WRITE(OUTPUT,' ');
000132 2031      END;
000132 2032      PX1 := PX1 + 8;
000134 2033  END
000134 2034  ELSE BEGIN (* END OF LINE CASES *)
000135 2035      IF DSSPTR^.BLK[ID].OVERFLOW = WRAP THEN BEGIN
000146 2036          ENDOFLN := TRUE; (* STATE THAT WE ARE AT THE END *)
000147 2037      END
000147 2038      ELSE BEGIN
000150 2039          (* ADVANCE TO END W/O DISPLAY *)
000150 2040          DSPLINE(ID,PTR,PTR1,CONSUME,PX,PY);
000157 2042      END;
000157 2043  END; (* OF ECL CASES *)
000157 2044  END
000157 2045  ELSE BEGIN
000160 2046      CASE CHR1 OF (* SEMANTICS *)
000161 2047
000161 2048          GO : BEGIN (* GRAPHICS ORIGIN *)
000161 2049              X := FETCH(PTR1,ENDT);
000165 2051              Y := FETCH(PTR1,FNCT);
000171 2052              IF X = 999 THEN DSSPTR^.BLK[ID].PLOTORGX := PX1 ELSE
000205 2053                  DSSPTR^.BLK[ID].PLOTORGX := DSSPTR^.BLK[ID].TEXTORGX + X;
000225 2054              IF Y = 999 THEN DSSPTR^.BLK[ID].PLOTORGY := PY ELSE
000241 2055                  DSSPTR^.BLK[ID].PLOTORGY := DSSPTR^.BLK[ID].TEXTENDY + Y;
000261 2056              (* SETCOORD(DSSPTR^.BLK[ID].PLOTORGX,DSSPTR^.BLK[ID].PLOTORGY,DOTS); *)
000261 2057          END;
000262 2058
000262 2059          SC : BEGIN (* SET CURSOR FOR GRAPHICS ETC. *)
000262 2060              X := FETCH(PTR1,ENDT);
000266 2061              Y := FETCH(PTR1,FNCT);
000272 2062              IF X = 999 THEN X := PX1 ELSE X := DSSPTR^.BLK[ID].PLOTORGX + X;
000307 2063              IF Y = 999 THEN Y := PY ELSE Y := DSSPTR^.BLK[ID].PLOTORGY + Y;
000324 2064              SETCOORD(X,Y,DOTS);
000331 2065          END;
000332 2066
000332 2067          LN : BEGIN (* DRAW A LINE *)
000332 2068              SETAG := TRUP;
000334 2069              X := FETCH(PTR1,ENDT);
000337 2070              Y := FETCH(PTR1,FNCT);
000343 2071              DRAWLINE(DSSPTR^.BLK[ID].PLOTORGX + X,
000354 2072                  DSSPTR^.BLK[ID].PLOTORGY + Y,
000364 2073                  DOTS,
000367 2074                  WMCDE,
000370 2075                  DSENO );
000372 2076          END;
000373 2077
000373 2078          PT : BEGIN (* PUT A POINT *)
000373 2079              SETAG := TRUE;
000375 2080              X := FETCH(PTR1,ENDT);
000400 2081              Y := FETCH(PTR1,ENDT);
000404 2082              DRAWPOINT(DSSPTR^.BLK[ID].PLOTORGX + X,
000415 2083                  DSSPTR^.BLK[ID].PLOTORGY + Y,
000425 2084                  DOTS,

```



```

000430 2085          WMODE,
000431 2086          DSPNO );
000433 2087      END;
000434 2088
000434 2089      CH : BEGIN  (* DRAW A CHARACTER *)
000434 2090          SETAG := TRUE;
000436 2091          CHR := FETCH(PTR1,ENDT);
000441 2092          X := FETCH(PTR1,ENCT);
000445 2093          Y := FETCH(PTR1,ENDT);
000451 2094          DRAWCHAR(CHR,
000453 2095              DSSPTR^.BLK[ ID].PLOTORGX + X,
000464 2096              DSSPTR^.BLK[ ID].PLOTORGX + Y,
000474 2097              DOTS,
000476 2098              CENTER,
000481 2099              NORMAL,
000503 2100              WMODE,
000505 2101              DSPNO );
000506 2102      END;
000507 2103
000507 2104      CA : BEGIN  (* DRAW A CHARACTER IN ALTERNATE SET *)
000507 2105          SETAG := TRUE;
000511 2106          CHR := FETCH(PTR1,ENDT);
000514 2107          X := FETCH(PTR1,ENCT);
000522 2108          Y := FETCH(PTR1,ENCT);
000524 2109          DRAWCHAR(CHR,
000526 2110              DSSPTR^.BLK[ ID].PLOTORGX + X,
000537 2111              DSSPTR^.BLK[ ID].PLOTORGX + Y,
000547 2112              DOTS,
000551 2113              CENTER,
000554 2114              PROGRAMMABLE,
000556 2115              WMODE,
000560 2116              DSPNO );
000561 2117      END;
000562 2118
000562 2119      ML : BEGIN  (* SET LEFT MARGIN *)
000562 2120          X := FETCH(PTR1,ENDT);
000566 2121          DSSPTR^.BLK[ ID].TEXTORGX := DSSPTR^.BLK[ ID].TEXTORGX + X*8;
000606 2122      END;
000607 2123
000607 2124      MR : BEGIN  (* SET RIGHT MARGIN *)
000607 2125          X := FETCH(PTR1,ENCT);
000613 2126          DSSPTR^.BLK[ ID].TEXTENDX := DSSPTR^.BLK[ ID].TEXTENDX - X*8;
000634 2127      END;
000634 2128
000634 2129      MU : BEGIN  (* SET UPPER MARGIN *)
000634 2130          Y := FETCH(PTR1,ENCT);
000640 2131          DSSPTR^.BLK[ ID].TEXTORGY := DSSPTR^.BLK[ ID].TEXTORGY - Y*8;
000661 2132      END;
000661 2133
000661 2134      MB : BEGIN  (* SET BOTTOM MARGIN *)
000661 2135          Y := FETCH(PTR1,ENDT);
000665 2136          DSSPTR^.BLK[ ID].TEXTENDY := DSSPTR^.BLK[ ID].TEXTENDY + Y*8;
000706 2137      END;
000706 2138
000706 2139      CR : BEGIN  (* CARRIAGE RETURN W/O LINEFEED *)
000706 2140          DSSPTR^.BLK[ ID].TEXTORGY := DSSPTR^.BLK[ ID].TEXTORGY;
  
```

```

000726 2142      SETCOORD(ISSPTR^.ELK[ID].TEXPY,
000735 2143      DSSPTR^.ELK[ID].TEXPY,
000744 2144      DOTS );
000747 2145  END;
000750 2146
000750 2147  AL : BEGIN (* PRINT STUFF IN ALTERNATE SET *)
000750 2148      IF SETAG THEN SETCOORD(PX1,PY,DOTS); (* GO BACK TO WHERE WE WERE IF NEED BE *)
000755 2149      SETAG := FALSE;
000757 2150      CHR := FETCH(PTR1,ENDT);
000762 2151      WHILE CHR <> NR DC BEGIN
000765 2152          IF DMODE <> UNDO THEN BEGIN
000767 2153              PUTCHAR(CHR,
000770 2154                  PROGRAMMABLE,
000771 2155                  DSSPTR^.BLK[ID].WMODE,
001002 2156                  LSPNO);
001004 2157          END
001004 2158      ELSE BEGIN
001005 2159          MODE(DSPTEXT,DSPNONE,DSPNORM,DSPNO);
001013 2160          WRITE(OUTPUT,' ');
001020 2161      END;
001020 2162      CHR := FETCH(PTR1,ENDT);
001024 2163  END; (*OF WHILE *)
001025 2164  END;
001026 2165
001026 2166  OV : BEGIN (* OVERSTRIKE *)
001026 2167      IF SETAG THEN SETCOORD(PX1,PY,DOTS); (* GO BACK TO WHERE WE WERE IF NEED BE *)
001033 2168      SETAG := FALSE;
001035 2169      CHR := FETCH(PTR1,ENDT);
001040 2170      IF WMODE <> FRASE THEN BEGIN
001043 2171          MODE(DSPTEXT,ISPNONE,OVER,DSPNO);
001050 2172          PUTCH(OUTPUT,EACK); (* BACKSPACE *)
001052 2173          PUTCH(OUTPUT,CHR);
001054 2174      END;
001054 2175  END;
001055 2176
001055 2177  OA : BEGIN (* OVERSTRIKE USING ALTERNATE SET *)
001055 2178      IF SETAG THEN SETCOORD(PX1,PY,DOTS); (* GO BACK TO WHERE WE WERE IF NEED BE *)
001062 2179      SETAG := FALSE;
001064 2180      CHR := FETCH(PTR1,ENDT);
001067 2181      IF WMODE <> FRASE THEN BEGIN
001072 2182          PUTCH(OUTPUT,EACK); (* BACKSPACE *)
001074 2183          PUTCHAR(CHR,
001076 2184              PROGRAMMABLE,
001100 2185              OVER,
001101 2186              DSPNC);
001102 2187      END;
001102 2188  END;
001103 2189
001103 2190  UN : BEGIN (* UNDERLINE TEXT *)
001103 2191      IF SETAG THEN SETCOORD(PX1,PY,DOTS); (* GO BACK TO WHERE WE WERE IF NEED BE *)
001119 2192      SETAG := FALSE;
001112 2193      IF DMODE <> UNDO THEN BEGIN
001114 2194          MODE(DSPTEXT,DSPNONE,DSPNORM,DSPNO);
001122 2195          I := 0;
001124 2196          CHR := FETCH(PTR1,ENDT);
001127 2197          WHILE CHR <> DE DO BEGIN
001132 2198              PUTCHAR(CHR,

```

```

001114 2199          PUTCH(OUTENT,CHR);
001115 2200          CHR := FETCH(PTR1,FNDT);
001116 2201          END;
001117 2202          (* OF WHILE *)
001118 2203          FOR IY := 1 TO I DO BEGIN
001119 2204              PUTCH(OUTENT,BACK);
001120 2205          END;
001121 2206          FOR IY := 1 TO I DO BEGIN
001122 2207              WRITE(OUTENT,'_');
001123 2208          END;
001124 2209      END
001125 2210      ELSE BEGIN
001126 2211          MODE(DSPTEXT,ESPNOFF,DSPNCRM,DSPNO);
001127 2212          CHR := FETCH(PTR1,FNDT);
001128 2213          WHILE CHR <> GF DO BEGIN
001129 2214              WRITE(OUTENT,' ');
001130 2215              CHR := FETCH(PTR1,FNDT);
001131 2216          END;
001132 2217      END;
001133 2218      FND;
001134 2219      ND : BEGIN
001135 2220          (* SET TO NORMAL DESTRUCTIVE *)
001136 2221          DSSPTR^.BLK[ID].WMODE := DSPNCRM;
001137 2222          IF DMODE <> UNDO THEN WMODE := DSPNORM;
001138 2223          MODE(DSPTEXT,DSPNCNF,DSSPTR^.BLK[ID].WMODE,DSPNO);
001139 2224      END;
001140 2225      NP : BEGIN
001141 2226          (* SET TO NORMAL PROTECTIVE *)
001142 2227          DSSPTR^.BLK[ID].WMODE := OVER;
001143 2228          IF DMODE <> UNDO THEN WMODE := OVER;
001144 2229          MODE(DSPTEXT,DSPNCNE,DSSPTR^.BLK[ID].WMODE,DSPNO);
001145 2230      END;
001146 2231      RD : BEGIN
001147 2232          (* SET TO REVERSE DESTRUCTIVE (INVERSE) *)
001148 2233          DSSPTR^.BLK[ID].WMODE := INVERSE;
001149 2234          IF DMODE <> UNDO THEN WMODE := INVERSE;
001150 2235          MODE(DSPTEXT,DSPNCNE,DSSPTR^.BLK[ID].WMODE,DSPNO);
001151 2236      END;
001152 2237      PP : BEGIN
001153 2238          (* SET TO REVERSE PROTECTIVE (ERASE) *)
001154 2239          DSSPTR^.BLK[ID].WMODE := ERASE;
001155 2240          WMODE := ERASE;
001156 2241          MODE(DSPTEXT,DSPNCNF,DSSPTR^.BLK[ID].WMODE,DSPNO);
001157 2242      END;
001158 2243      TT,TU : BEGIN
001159 2244          (* THE TWO TARGET TYPES *)
001160 2245          SETAG := TRUE;
001161 2246          IF WMODE <> ERASE THEN BEGIN
001162 2247              IF CHR1 = TT THEN TTYPE := PAT ELSE TTYPE := UNDERLINE;
001163 2248              TARGID := FETCH(PTR1,FNDT);
001164 2249              INDEX := FETCH(PTR1,FNDT);
001165 2250              LABCNT := FETCH(PTR1,FNDT);
001166 2251              CREATETARG(ID,TARGID,
001167 2252                  0,0,0,0,DOTS,DOTS,
001168 2253                  LI,TRUE,TRUE,
001169 2254                  TTYPE,
001170 2255                  DSSPTR^.LABARR,
001171 2256                  LABCNT);
001172 2257          END;
001173 2258      END;
001174 2259      END;
001175 2260      END;
001176 2261      END;
001177 2262      END;
001178 2263      END;
001179 2264      END;
001180 2265      END;
001181 2266      END;
001182 2267      END;
001183 2268      END;
001184 2269      END;
001185 2270      END;
001186 2271      END;
001187 2272      END;
001188 2273      END;
001189 2274      END;
001190 2275      END;
001191 2276      END;
001192 2277      END;
001193 2278      END;
001194 2279      END;
001195 2280      END;
001196 2281      END;
001197 2282      END;
001198 2283      END;
001199 2284      END;
001200 2285      END;
001201 2286      END;
001202 2287      END;
001203 2288      END;
001204 2289      END;
001205 2290      END;
001206 2291      END;
001207 2292      END;
001208 2293      END;
001209 2294      END;
001210 2295      END;
001211 2296      END;
001212 2297      END;
001213 2298      END;
001214 2299      END;
001215 2300      END;
001216 2301      END;
001217 2302      END;
001218 2303      END;
001219 2304      END;
001220 2305      END;
001221 2306      END;
001222 2307      END;
001223 2308      END;
001224 2309      END;
001225 2310      END;
001226 2311      END;
001227 2312      END;
001228 2313      END;
001229 2314      END;
001230 2315      END;
001231 2316      END;
001232 2317      END;
001233 2318      END;
001234 2319      END;
001235 2320      END;
001236 2321      END;
001237 2322      END;
001238 2323      END;
001239 2324      END;
001240 2325      END;
001241 2326      END;
001242 2327      END;
001243 2328      END;
001244 2329      END;
001245 2330      END;
001246 2331      END;
001247 2332      END;
001248 2333      END;
001249 2334      END;
001250 2335      END;
001251 2336      END;
001252 2337      END;
001253 2338      END;
001254 2339      END;
001255 2340      END;
001256 2341      END;
001257 2342      END;
001258 2343      END;
001259 2344      END;
001260 2345      END;
001261 2346      END;
001262 2347      END;
001263 2348      END;
001264 2349      END;
001265 2350      END;
001266 2351      END;
001267 2352      END;
001268 2353      END;
001269 2354      END;
001270 2355      END;
001271 2356      END;
001272 2357      END;
001273 2358      END;
001274 2359      END;
001275 2360      END;
001276 2361      END;
001277 2362      END;
001278 2363      END;
001279 2364      END;
001280 2365      END;
001281 2366      END;
001282 2367      END;
001283 2368      END;
001284 2369      END;
001285 2370      END;
001286 2371      END;
001287 2372      END;
001288 2373      END;
001289 2374      END;
001290 2375      END;
001291 2376      END;
001292 2377      END;
001293 2378      END;
001294 2379      END;
001295 2380      END;
001296 2381      END;
001297 2382      END;
001298 2383      END;
001299 2384      END;
001300 2385      END;
001301 2386      END;
001302 2387      END;
001303 2388      END;
001304 2389      END;
001305 2390      END;
001306 2391      END;
001307 2392      END;
001308 2393      END;
001309 2394      END;
001310 2395      END;
001311 2396      END;
001312 2397      END;
001313 2398      END;
001314 2399      END;
001315 2400      END;
001316 2401      END;
001317 2402      END;
001318 2403      END;
001319 2404      END;
001320 2405      END;
001321 2406      END;
001322 2407      END;
001323 2408      END;
001324 2409      END;
001325 2410      END;
001326 2411      END;
001327 2412      END;
001328 2413      END;
001329 2414      END;
001330 2415      END;
001331 2416      END;
001332 2417      END;
001333 2418      END;
001334 2419      END;
001335 2420      END;
001336 2421      END;
001337 2422      END;
001338 2423      END;
001339 2424      END;
001340 2425      END;
001341 2426      END;
001342 2427      END;
001343 2428      END;
001344 2429      END;
001345 2430      END;
001346 2431      END;
001347 2432      END;
001348 2433      END;
001349 2434      END;
001350 2435      END;
001351 2436      END;
001352 2437      END;
001353 2438      END;
001354 2439      END;
001355 2440      END;
001356 2441      END;
001357 2442      END;
001358 2443      END;
001359 2444      END;
001360 2445      END;
001361 2446      END;
001362 2447      END;
001363 2448      END;
001364 2449      END;
001365 2450      END;
001366 2451      END;
001367 2452      END;
001368 2453      END;
001369 2454      END;
001370 2455      END;
001371 2456      END;
001372 2457      END;
001373 2458      END;
001374 2459      END;
001375 2460      END;
001376 2461      END;
001377 2462      END;
001378 2463      END;
001379 2464      END;
001380 2465      END;
001381 2466      END;
001382 2467      END;
001383 2468      END;
001384 2469      END;
001385 2470      END;
001386 2471      END;
001387 2472      END;
001388 2473      END;
001389 2474      END;
001390 2475      END;
001391 2476      END;
001392 2477      END;
001393 2478      END;
001394 2479      END;
001395 2480      END;
001396 2481      END;
001397 2482      END;
001398 2483      END;
001399 2484      END;
001400 2485      END;
001401 2486      END;
001402 2487      END;
001403 2488      END;
001404 2489      END;
001405 2490      END;
001406 2491      END;
001407 2492      END;
001408 2493      END;
001409 2494      END;
001410 2495      END;
001411 2496      END;
001412 2497      END;
001413 2498      END;
001414 2499      END;
001415 2500      END;
001416 2501      END;
001417 2502      END;
001418 2503      END;
001419 2504      END;
001420 2505      END;
001421 2506      END;
001422 2507      END;
001423 2508      END;
001424 2509      END;
001425 2510      END;
001426 2511      END;
001427 2512      END;
001428 2513      END;
001429 2514      END;
001430 2515      END;
001431 2516      END;
001432 2517      END;
001433 2518      END;
001434 2519      END;
001435 2520      END;
001436 2521      END;
001437 2522      END;
001438 2523      END;
001439 2524      END;
001440 2525      END;
001441 2526      END;
001442 2527      END;
001443 2528      END;
001444 2529      END;
001445 2530      END;
001446 2531      END;
001447 2532      END;
001448 2533      END;
001449 2534      END;
001450 2535      END;
001451 2536      END;
001452 2537      END;
001453 2538      END;
001454 2539      END;
001455 2540      END;
001456 2541      END;
001457 2542      END;
001458 2543      END;
001459 2544      END;
001460 2545      END;
001461 2546      END;
001462 2547      END;
001463 2548      END;
001464 2549      END;
001465 2550      END;
001466 2551      END;
001467 2552      END;
001468 2553      END;
001469 2554      END;
001470 2555      END;
001471 2556      END;
001472 2557      END;
001473 2558      END;
001474 2559      END;
001475 2560      END;
001476 2561      END;
001477 2562      END;
001478 2563      END;
001479 2564      END;
001480 2565      END;
001481 2566      END;
001482 2567      END;
001483 2568      END;
001484 2569      END;
001485 2570      END;
001486 2571      END;
001487 2572      END;
001488 2573      END;
001489 2574      END;
001490 2575      END;
001491 2576      END;
001492 2577      END;
001493 2578      END;
001494 2579      END;
001495 2580      END;
001496 2581      END;
001497 2582      END;
001498 2583      END;
001499 2584      END;
001500 2585      END;
001501 2586      END;
001502 2587      END;
001503 2588      END;
001504 2589      END;
001505 2590      END;
001506 2591      END;
001507 2592      END;
001508 2593      END;
001509 2594      END;
001510 2595      END;
001511 2596      END;
001512 2597      END;
001513 2598      END;
001514 2599      END;
001515 2600      END;
001516 2601      END;
001517 2602      END;
001518 2603      END;
001519 2604      END;
001520 2605      END;
001521 2606      END;
001522 2607      END;
001523 2608      END;
001524 2609      END;
001525 2610      END;
001526 2611      END;
001527 2612      END;
001528 2613      END;
001529 2614      END;
001530 2615      END;
001531 2616      END;
001532 2617      END;
001533 2618      END;
001534 2619      END;
001535 2620      END;
001536 2621      END;
001537 2622      END;
001538 2623      END;
001539 2624      END;
001540 2625      END;
001541 2626      END;
001542 2627      END;
001543 2628      END;
001544 2629      END;
001545 2630      END;
001546 2631      END;
001547 2632      END;
001548 2633      END;
001549 2634      END;
001550 2635      END;
001551 2636      END;
001552 2637      END;
001553 2638      END;
001554 2639      END;
001555 2640      END;
001556 2641      END;
001557 2642      END;
001558 2643      END;
001559 2644      END;
001560 2645      END;
001561 2646      END;
001562 2647      END;
001563 2648      END;
001564 2649      END;
001565 2650      END;
001566 2651      END;
001567 2652      END;
001568 2653      END;
001569 2654      END;
001570 2655      END;
001571 2656      END;
001572 2657      END;
001573 2658      END;
001574 2659      END;
001575 2660      END;
001576 2661      END;
001577 2662      END;
001578 2663      END;
001579 2664      END;
001580 2665      END;
001581 2666      END;
001582 2667      END;
001583 2668      END;
001584 2669      END;
001585 2670      END;
001586 2671      END;
001587 2672      END;
001588 2673      END;
001589 2674      END;
001590 2675      END;
001591 2676      END;
001592 2677      END;
001593 2678      END;
001594 2679      END;
001595 2680      END;
001596 2681      END;
001597 2682      END;
001598 2683      END;
001599 2684      END;
001600 2685      END;
001601 2686      END;
001602 2687      END;
001603 2688      END;
001604 2689      END;
001605 2690      END;
001606 2691      END;
001607 2692      END;
001608 2693      END;
001609 2694      END;
001610 2695      END;
001611 2696      END;
001612 2697      END;
001613 2698      END;
001614 2699      END;
001615 2700      END;
001616 2701      END;
001617 2702      END;
001618 2703      END;
001619 2704      END;
001620 2705      END;
001621 2706      END;
001622 2707      END;
001623 2708      END;
001624 2709      END;
001625 2710      END;
001626 2711      END;
001627 2712      END;
001628 2713      END;
001629 2714      END;
001630 2715      END;
001631 2716      END;
001632 2717      END;
001633 2718      END;
001634 2719      END;
001635 2720      END;
001636 2721      END;
001637 2722      END;
001638 2723      END;
001639 2724      END;
001640 2725      END;
001641 2726      END;
001642 2727      END;
001643 2728      END;
001644 2729      END;
001645 2730      END;
001646 2731      END;
001647 2732      END;
001648 2733      END;
001649 2734      END;
001650 2735      END;
001651 2736      END;
001652 2737      END;
001653 2738      END;
001654 2739      END;
001655 2740      END;
001656 2741      END;
001657 2742      END;
001658 2743      END;
001659 2744      END;
001660 2745      END;
001661 2746      END;
001662 2747      END;
001663 2748      END;
001664 2749      END;
001665 2750      END;
001666 2751      END;
001667 2752      END;
001668 2753      END;
001669 2754      END;
001670 2755      END;
001671 2756      END;
001672 2757      END;
001673 2758      END;
001674 2759      END;
001675 2760      END;
001676 2761      END;
001677 2762      END;
001678 2763      END;
001679 2764      END;
001680 2765      END;
001681 2766      END;
001682 2767      END;
001683 2768      END;
001684 2769      END;
001685 2770      END;
001686 2771      END;
001687 2772      END;
001688 2773      END;
001689 2774      END;
001690 2775      END;
001691 2776      END;
001692 2777      END;
001693 2778      END;
001694 2779      END;
001695 2780      END;
001696 2781      END;
001697 2782      END;
001698 2783      END;
001699 2784      END;
001700 2785      END;
001701 2786      END;
001702 2787      END;
001703 2788      END;
001704 2789      END;
001705 2790      END;
001706 2791      END;
001707 2792      END;
001708 2793      END;
001709 2794      END;
001710 2795      END;
001711 2796      END;
001712 2797      END;
001713 2798      END;
001714 2799      END;
001715 2800      END;
001716 2801      END;
001717 2802      END;
001718 2803      END;
001719 2804      END;
001720 2805      END;
001721 2806      END;
001722 2807      END;
001723 2808      END;
001724 2809      END;
001725 2810      END;
001726 2811      END;
001727 2812      END;
001728 2813      END;
001729 2814      END;
001730 2815      END;
001731 2816      END;
001732 2817      END;
001733 2818      END;
001734 2819      END;
001735 2820      END;
001736 2821      END;
001737 2822      END;
001738 2823      END;
001739 2824      END;
001740 2825      END;
001741 2826      END;
001742 2827      END;
001743 2828      END;
001744 2829      END;
001745 2830      END;
001746 2831      END;
001747 2832      END;
001748 2833      END;
001749 2834      END;
001750 2835      END;
001751 2836      END;
001752 2837      END;
001753 2838      END;
001754 2839      END;
001755 2840      END;
001756 2841      END;
001757 2842      END;
001758 2843      END;
001759 2844      END;
001760 2845      END;
001761 2846      END;
001762 2847      END;
001763 2848      END;
001764 2849      END;
001765 2850      END;
001766 2851      END;
001767 2852      END;
001768 2853      END;
001769 2854      END;
001770 2855      END;
001771 2856      END;
001772 2857      END;
001773 2858      END;
001774 2859      END;
001775 2860      END;
001776 2861      END;
001777 2862      END;
001778 2863      END;
001779 2864      END;
001780 2865      END;
001781 2866      END;
001782 2867      END;
001783 2868      END;
001784 2869      END;
001785 2870      END;
001786 2871      END;
001787 2872      END;
001788 2873      END;
001789 2874      END;
001790 2875      END;
001791 2876      END;
001792 2877      END;
001793 2878      END;
001794 2879      END;
001795 2880      END;
001796 2881      END;
001797 2882      END;
001798 2883      END;
001799 2884      END;
001800 2885      END;
001801 2886      END;
001802 2887      END;
001803 2888      END;
001804 2889      END;
001805 2890      END;
001806 2891      END;
001807 2892      END;
001808 2893      END;
001809 2894      END;
001810 2895      END;
001811 2896      END;
001812 2897      END;
001813 2898      END;
001814 2899      END;
001815 2900      END;
001816 2901      END;
001817 2902      END;
001818 2903      END;
001819 2904      END;
001820 2905      END;
001821 2906      END;
001822 2907      END;
001823 2908      END;
001824 2909      END;
001825 2910      END;
001826 2911      END;
001827 2912      END;
001828 2913      END;
001829 2914      END;
001830 2915      END;
001831 2916      END;
001832 2917      END;
001833 2918      END;
001834 2919      END;
001835 2920      END;
001836 2921      END;
001837 2922      END;
001838 2923      END;
001839 2924      END;
001840 2925      END;
001841 2926      END;
001842 2927      END;
001843 2928      END;
001844 2929      END;
001845 2930      END;
001846 2931      END;
001847 2932      END;
001848 2933      END;
001849 2934      END;
001850 2935      END;
001851 2936      END;
001852 2937      END;
001853 2938      END;
001854 2939      END;
001855 2940      END;
001856 2941      END;
001857 2942      END;
001858 2943      END;
001859 2944      END;
001860 2945      END;
001861 2946      END;
001862 2947      END;
001863 2948      END;
001864 2949      END;
001865 2950      END;
001866 2951      END;
001867 2952      END;
001868 2953      END;
001869 2954      END;
001870 2955      END;
001871 2956      END;
001872 2957      END;
001873 2958      END;
001874 2959      END;
001875 2960      END;
001876 2961      END;
001877 2962      END;
001878 2963      END;
001879 2964      END;
001880 2965      END;
001881 2966      END;
001882 2967      END;
001883 2968      END;
001884 2969      END;
001885 2970      END;
001886 2971      END;
001887 2972      END;
001888 2973      END;
001889 2974      END;
001890 2975      END;
001891 2976      END;
001892 2977      END;
001893 2978      END;
001894 2979      END;
001895 2980      END;
001896 2981      END;
001897 2982      END;
0
```

```

201460 2256          END
201460 2257          ELSE BEGIN
201461 2258              TARGID := FETCH(PTR1,ENDT);
201465 2259              INDEY := FETCH(PTR1,ENDT);
201471 2260              LABCNT := FETCH(PTR1,ENDT);
201475 2261              DSTTARG(ID,TARGID);
201500 2262          END;
201510 2263      END;
201501 2264
201501 2265      OTHERWISE (* NOTHING *)
201522 2266
201522 2267          END; (* OF CASE *)
201522 2268          FND; (* OF IF *)
201522 2269          CHR1 := FETCH(PTR1,ENDT);
201526 2270          FND; (* OF WHILE *)
201527 2271      END
201527 2272      ELSE BEGIN (* JUST CONSUME UNTIL END OF LINE *)
201530 2273          CHR := FETCH(PTR1,ENDT);
201534 2274          WHILE (CHR <> EL) AND (CHR <> LI) DO BEGIN
201541 2275              CHR := FETCH(PTR1,ENDT);
201545 2276          END;
201546 2277      END;
201546 2278      END; (* OF DSPLINE *)
201616 2279
201616 2280
201616 2281      (*-----*)
201616 2282      PROCEDURE GETTOUCHING (
200002 2283          VAR X,Y : INTEGER; (* X,Y COORDINATES OF TOUCH *)
200004 2284          VAR CHUP : INBUF; (* ARRAY OF INPUT CHARACTERS *)
200005 2285          VAR LEN : INTEGER; (* LENGTH OF STRING IN CHARRAY *)
200006 2286          MKCIRCLE : BOOLEAN; (* MAKE A CIRCLE
200007 2287              WHERE USER TOUCHED *)
200007 2288          VAR ERROR : ERRORTYPE (* BADTOUCH *) );
200010 2289
200010 2290      FORWARD;
200010 2291
200010 2292
200010 2293      (*-----*)
200010 2294      PROCEDURE HANDLEEOL(ID : INTEGER (* BLOCK ID *)); (* HANDLE NEXT LINE CASES *)
200003 2295      VAR
200003 2296          PX,PY : INTEGER;
200005 2297          PTR,PTR1 : TEXTPTR;
200011 2298          TEMP : DSPRUPTR;
200012 2300          CHARR : INBUF;
200030 2301          X,Y : INTEGER;
200032 2302          LEN : INTEGER;
200033 2303          ERROR : ERRORTYPE;
200034 2304
200034 2305      BEGIN
200034 2306          DSSPTR^.BLK[ID].TEXPY := DSSPTR^.BLK[ID].TEXPY - 16; (* GO DOWN A LINE *)
200024 2307          IF DSSPTR^.BLK[ID].TEXPY <= DSSPTR^.BLK[ID].TEXTFNDY THEN BEGIN (* END OF PAGE CONDITION *)
200043 2308
200043 2309              (* PAUSE AND ALLOW USER TO READ THE SCREEN *)
200043 2310              (* USER TOUCHES SCREEN TO CONTINUE *)
200043 2311              GETTOUCHING(X,Y,CHARR,LEN,FALSE,ERROR);
200052 2312
  
```

```

000052 2313      CASE DSSPTR^.BLK[ID].OVERFLOW OF
000063 2314
000063 2315          WOScroll: BEGIN      (* ERASE SCREEN THEN PRINT LINE *)
000063 2316              PY := ISSPTR^.BLK[ID].TEXTORGX;
000073 2317              PY := DSSPTR^.BLK[ID].TEXTORGX;
000102 2318              PTR.BUF := DSSPTR^.BLK[ID].HEAD;
000113 2319              PTR.POS := DSSPTR^.BLK[ID].HEAD^.POS;
000130 2320              WHILE {PTR.BUF <> DSSPTR^.BLK[ID].LSTPTR.BUF} OR
000142 2321                  {PTR.POS <> DSSPTR^.BLK[ID].LSTPTR.POS} DO BEGIN
000153 2322                  DSPLINE(ID,PTR,PTR1,UNDO,PX,PY);      (* UNDO LIN* *)
000161 2323                  WHILE PTR.BUF <> PTR1.BUF DO BEGIN
000164 2324                      TEMP := PTR.BUF^.NEXT;
000171 2325                      DISPOSE (PTR.BUF);
000173 2326                      PTR.BUF := TEMP;
000175 2327                      PTR.POS := PTR.BUF^.POS;
000204 2328                  END;      (* OF WHILE *)
000205 2329                  PTR := PTR1;
000207 2330                  PY := PY - 16;      (* ADVANCE LINE POINTER *)
000211 2331              END;
000212 2332              DSSPTR^.BLK[ID].HEAD := ISSPTR^.BLK[ID].LSTPTR.BUF;
000231 2333              DSSPTR^.BLK[ID].HEAD^.POS := DSSPTR^.BLK[ID].LSTPTR.POS;
000254 2334              DSSPTR^.BLK[ID].TEXTX := ISSPTR^.BLK[ID].TEXTORGX;
000272 2335              DSSPTR^.BLK[ID].TEXTY := ISSPTR^.BLK[ID].TEXTORGX;
000310 2336          END;      (* OF WOScroll *)
000311 2337
000311 2338          SCROLL : BEGIN      (* SCROLL LINES *)
000311 2339          END;
000312 2340      END;      (* OF CASE *)
000312 2341      END;      (* OF IF *)
000316 2342      END;      (* OF HANDLEOL *)
000316 2343
000340 2344
000340 2345
000340 2346      (* ===== *)
000340 2347      PROCEDURE INITDSPARRAYS (
000002 2348          CHFILE : ALFA      (* EXTERNAL FILE NAME OF ALTERNATE CHARACTER SET *)
000003 2349      );
000003 2350
000003 2351      (* USER DESCRIPTION:
000003 2352          THIS PROCEDURE SHOULD BE CALLED ONCE BEFORE
000003 2353          ANY OF THE DISPLAY ROUTINES ARE CALLED. THIS
000003 2354          IS A DATA DEFINITION PROCEDURE.
000003 2355      *)
000003 2356
000003 2357      (* INTERNAL DESCRIPTION:
000003 2358          NOTHING SURPRISING ABOUT THIS PROCEDURE.
000003 2359          CLEANS THE SCREEN WHEN CALLED.
000003 2360      *)
000003 2361
000003 2362
000003 2363
000003 2364      VAR
000003 2365          I,J,K,L      : INTEGER;
000007 2366          ALTCARSET : TEXT;
000043 2367          CHMEM : NOM;
000643 2368          CHNO : INTEGER;
  
```

```

000644 2370 NEW(DSSPTR); (* CREATE THE LOCAL STATIC RECORD STRUCTURE *)
000645 2371 DSSPTR^.SAVEMODE := DSPNORM;
000646 2372 DSSPTR^.SAVE1 := DSPTEXT;
000647 2373 DSSPTR^.SAVE2 := DSPNCRF;
000648 2374 FOR I := 1 TO BLOCKMAX DO DSSPTR^.DISARM[I] := 0;
000649 2375 LINELIMIT(OUTPUT,MAXINT);
000650 2376 DSSPTR^.BLOCK0 := BLOCKMAX + 1;
000651 2377 FOR I := 0 TO 15 DO BEGIN
000652 2378   FOR J := 0 TO 15 TO BEGIN
000653 2379     DSSPTR^.TARGARRAY[I,J].TARG := NIL;
000654 2380   END;
000655 2381 END;
000656 2382 FOR I := 1 TO BLOCKMAX DO DSSPTR^.BLK[I].INUSE := FALSE;
000657 2383 FOR I := 0 TO 1 DO BEGIN
000658 2384   FOR J := 0 TO 1 DO BEGIN
000659 2385     FOR K := 0 TO 1 DO BEGIN
000660 2386       FOR L := 0 TO 1 DO BEGIN
000661 2387         DSSPTR^.REV[I*8 + J*4 + K*2 + L] := L*8 + K*4 + J*2 + I;
000662 2388       END
000663 2389     END
000664 2390   END
000665 2391 END;
000666 2392 FOR I := 0 TO 127 DO BEGIN
000667 2393   FOR J := 0 TO 7 DO BEGIN
000668 2394     DSSPTR^.CHARSET[I,J] := 0;
000669 2395   END
000670 2396 END;
000671 2397 FOR I := 0 TO 15 DO BEGIN
000672 2398   FOR J := 0 TO 7 DO BEGIN
000673 2399     DSSPTR^.BUILD[I,J] := ' ';
000674 2400   END
000675 2401 END;
000676 2402 (* BUILD UP LOOKUP TABLE FOR DISPLAY TEXT *)
000677 2403 DSSPTR^.LOOKUP[1,1] := ORD('L') + 64; DSSPTR^.LOOKUP[1,2] := ORD('N') + 64; DSSPTR^.LOOKUP[1,3] := LN;
000678 2404 DSSPTR^.LOOKUP[2,1] := ORD('P') + 64; DSSPTR^.LOOKUP[2,2] := ORD('T') + 64; DSSPTR^.LOOKUP[2,3] := PT;
000679 2405 DSSPTR^.LOOKUP[3,1] := ORD('C') + 64; DSSPTR^.LOOKUP[3,2] := ORD('H') + 64; DSSPTR^.LOOKUP[3,3] := CH;
000680 2406 DSSPTR^.LOOKUP[4,1] := ORD('C') + 64; DSSPTR^.LOOKUP[4,2] := ORD('A') + 64; DSSPTR^.LOOKUP[4,3] := CA;
000681 2407 DSSPTR^.LOOKUP[5,1] := ORD('S') + 64; DSSPTR^.LOOKUP[5,2] := ORD('C') + 64; DSSPTR^.LOOKUP[5,3] := SC;
000682 2408 DSSPTR^.LOOKUP[6,1] := ORD('M') + 64; DSSPTR^.LOOKUP[6,2] := ORD('L') + 64; DSSPTR^.LOOKUP[6,3] := ML;
000683 2409 DSSPTR^.LOOKUP[7,1] := ORD('M') + 64; DSSPTR^.LOOKUP[7,2] := ORD('R') + 64; DSSPTR^.LOOKUP[7,3] := MR;
000684 2410 DSSPTR^.LOOKUP[8,1] := ORD('M') + 64; DSSPTR^.LOOKUP[8,2] := ORD('U') + 64; DSSPTR^.LOOKUP[8,3] := MU;
000685 2411 DSSPTR^.LOOKUP[9,1] := ORD('M') + 64; DSSPTR^.LOOKUP[9,2] := ORD('B') + 64; DSSPTR^.LOOKUP[9,3] := MB;
000686 2412 DSSPTR^.LOOKUP[10,1] := ORD('E') + 64; DSSPTR^.LOOKUP[10,2] := ORD('I') + 64; DSSPTR^.LOOKUP[10,3] := EL;
000687 2413 DSSPTR^.LOOKUP[11,1] := ORD('O') + 64; DSSPTR^.LOOKUP[11,2] := ORD('V') + 64; DSSPTR^.LOOKUP[11,3] := OV;
000688 2414 DSSPTR^.LOOKUP[12,1] := ORD('O') + 64; DSSPTR^.LOOKUP[12,2] := ORD('A') + 64; DSSPTR^.LOOKUP[12,3] := OA;
000689 2415 DSSPTR^.LOOKUP[13,1] := ORD('U') + 64; DSSPTR^.LOOKUP[13,2] := ORD('N') + 64; DSSPTR^.LOOKUP[13,3] := UN;
000690 2416 DSSPTR^.LOOKUP[14,1] := ORD('U') + 64; DSSPTR^.LOOKUP[14,2] := ORD('P') + 64; DSSPTR^.LOOKUP[14,3] := UP;
000691 2417 DSSPTR^.LOOKUP[15,1] := ORD('N') + 64; DSSPTR^.LOOKUP[15,2] := ORD('P') + 64; DSSPTR^.LOOKUP[15,3] := NP;
000692 2418 DSSPTR^.LOOKUP[16,1] := ORD('N') + 64; DSSPTR^.LOOKUP[16,2] := ORD('D') + 64; DSSPTR^.LOOKUP[16,3] := ND;
000693 2419 DSSPTR^.LOOKUP[17,1] := ORD('R') + 64; DSSPTR^.LOOKUP[17,2] := ORD('P') + 64; DSSPTR^.LOOKUP[17,3] := RP;
000694 2420 DSSPTR^.LOOKUP[18,1] := ORD('R') + 64; DSSPTR^.LOOKUP[18,2] := ORD('D') + 64; DSSPTR^.LOOKUP[18,3] := RD;
000695 2421 DSSPTR^.LOOKUP[19,1] := ORD('C') + 64; DSSPTR^.LOOKUP[19,2] := ORD('R') + 64; DSSPTR^.LOOKUP[19,3] := CR;
000696 2422 DSSPTR^.LOOKUP[20,1] := ORD('L') + 64; DSSPTR^.LOOKUP[20,2] := ORD('I') + 64; DSSPTR^.LOOKUP[20,3] := LI;
000697 2423 DSSPTR^.LOOKUP[21,1] := ORD('A') + 64; DSSPTR^.LOOKUP[21,2] := ORD('L') + 64; DSSPTR^.LOOKUP[21,3] := AL;
000698 2424 DSSPTR^.LOOKUP[22,1] := ORD('N') + 64; DSSPTR^.LOOKUP[22,2] := ORD('R') + 64; DSSPTR^.LOOKUP[22,3] := NR;
000699 2425 DSSPTR^.LOOKUP[23,1] := ORD('T') + 64; DSSPTR^.LOOKUP[23,2] := ORD('T') + 64; DSSPTR^.LOOKUP[23,3] := TT;
000700 2426 DSSPTR^.LOOKUP[24,1] := ORD('T') + 64; DSSPTR^.LOOKUP[24,2] := ORD('U') + 64; DSSPTR^.LOOKUP[24,3] := TU;

```

```

001071 2427 DSSPTR^.LOOKUP[25,1] := ORD('T') + 64; DSSPTR^.LOOKUP[25,2] := ORD('E') + 64; DSSPTR^.LOOKUP[25,3] := TE;
001114 2429 DSSPTR^.LOOKUP[26,1] := ORD('G') + 64; DSSPTR^.LOOKUP[26,2] := ORD('O') + 64; DSSPTR^.LOOKUP[26,3] := GO;
001134 2429 DSSPTR^.LOOKUP[27,1] := ORD('S') + 64; DSSPTR^.LOOKUP[27,2] := ORD('U') + 64; DSSPTR^.LOOKUP[27,3] := SU;
001155 2430 DSSPTR^.LOOKUP[28,1] := ORD('S') + 64; DSSPTR^.LOOKUP[28,2] := ORD('A') + 64; DSSPTR^.LOOKUP[28,3] := SA;
001176 2431 DSSPTR^.LOOKUP[29,1] := ORD('U') + 64; DSSPTR^.LOOKUP[29,2] := ORD('H') + 64; DSSPTR^.LOOKUP[29,3] := UU;
001216 2432 DSSPTR^.LOOKUP[30,1] := ORD('H') + 64; DSSPTR^.LOOKUP[30,2] := ORD('A') + 64; DSSPTR^.LOOKUP[30,3] := UA;
001236 2433 FOR I := 32 TO 127 DO DSSPTR^.CHARS[I] := ' ';
001254 2434 FOR I := 1 TO 26 DO DSSPTR^.CHARS[I+64] := CHR(I+ORD('A')-1);
001275 2435 FOR I := 1 TO 10 DO DSSPTR^.CHARS[I+47] := CHR(I+ORD('0')-1);
001316 2436 DSSPTR^.CHARS[32] := ' ';
001323 2437 DSSPTR^.CHARS[33] := '1';
001330 2438 DSSPTR^.CHARS[34] := '2';
001335 2439 DSSPTR^.CHARS[35] := '3';
001342 2440 DSSPTR^.CHARS[36] := '4';
001347 2441 DSSPTR^.CHARS[37] := '5';
001354 2442 DSSPTR^.CHARS[38] := '6';
001361 2443 DSSPTR^.CHARS[39] := '7';
001366 2444 DSSPTR^.CHARS[40] := '8';
001373 2445 DSSPTR^.CHARS[41] := '9';
001400 2446 DSSPTR^.CHARS[42] := '0';
001405 2447 DSSPTR^.CHARS[43] := 'A';
001412 2448 DSSPTR^.CHARS[44] := 'B';
001417 2449 DSSPTR^.CHARS[45] := 'C';
001424 2450 DSSPTR^.CHARS[46] := 'D';
001431 2451 DSSPTR^.CHARS[47] := 'E';
001436 2452 DSSPTR^.CHARS[48] := 'F';
001443 2453 DSSPTR^.CHARS[49] := 'G';
001450 2454 DSSPTR^.CHARS[50] := 'H';
001455 2455 DSSPTR^.CHARS[51] := 'I';
001462 2456 DSSPTR^.CHARS[52] := 'J';
001467 2457 DSSPTR^.CHARS[53] := 'K';
001474 2458 DSSPTR^.CHARS[54] := 'L';
001501 2459 DSSPTR^.CHARS[55] := 'M';
001506 2460 DSSPTR^.CHARS[56] := 'N';
001513 2461 FOR I := 0 TO 31 DO DSSPTR^.CHARS[I+96] := CHR(I + 32);
001534 2462 DSSPTR^.CONV[' '] := 32;
001541 2463 DSSPTR^.CONV['1'] := 33;
001546 2464 DSSPTR^.CONV['2'] := 34;
001553 2465 DSSPTR^.CONV['3'] := 35;
001560 2466 DSSPTR^.CONV['4'] := 36;
001565 2467 DSSPTR^.CONV['5'] := 37;
001572 2468 DSSPTR^.CONV['6'] := 38;
001577 2469 DSSPTR^.CONV['7'] := 39;
001604 2470 DSSPTR^.CONV['8'] := 40;
001611 2471 DSSPTR^.CONV['9'] := 41;
001616 2472 DSSPTR^.CONV['0'] := 42;
001623 2473 DSSPTR^.CONV['A'] := 43;
001630 2474 DSSPTR^.CONV['B'] := 44;
001635 2475 DSSPTR^.CONV['C'] := 45;
001642 2476 DSSPTR^.CONV['D'] := 46;
001647 2477 DSSPTR^.CONV['E'] := 47;
001654 2478 DSSPTR^.CONV['F'] := 48;
001661 2479 DSSPTR^.CONV['G'] := 49;
001666 2480 DSSPTR^.CONV['H'] := 50;
001673 2481 DSSPTR^.CONV['I'] := 51;
001700 2482 DSSPTR^.CONV['J'] := 52;
001705 2483 DSSPTR^.CONV['K'] := 53;
  
```

(* SINGLE QUOTE *)

(* UNDERLINE *)

PASCAL COMPTLER - E.T.H. ZURICH / UNIVERSITY OF MINNESOTA.
DISPLAY MODULE LOW LEVEL ROUTINES

PASCAL 6000 V3.0.0. 80/11/17. 00.56.01.
NOS 1.4 (80/04/21) PAGE 52

```
001712 2484 DSSPTR^.CONV['\'] := 92;
001717 2485 DSSPTR^.CONV[' ' ] := 93;
001724 2486 DSSPTR^.CONV['_'] := 95;
001731 2487 PUTCH(OUTPUT,CLEAR); (* CLEAR SCREEN *)
001733 2488 OPEN(ALTCHARSET,CHFILE,FALSE); (* OPEN CHARACTER SET FILE *)
001737 2489 RESET(ALTCHARSET);
001741 2490 GETSET(ALTCHARSET,CHFILE,CHNO); (* GET THE CHARACTERS *)
001744 2491 SETCOORD(140,256,DOTS);
001750 2492 MODE(DSPTEXT,DSPNONE,DSPNORM,DSPNO);
001756 2493 WRITE(OUTPUT,'CYBER IS LOADING CHARACTERS');
001761 2494 LOADCHRS(CHFILE,0,CHNO);
001766 2495 MODE(DSPTEXT,DSPNONE,DSPNORM,DSPNO);
001774 2496 WRITE(OUTPUT,' - DONE. ');
002001 2497 CLOSE(ALTCHARSET);
002003 2498 PUTCH(OUTPUT,CLEAR); (* CLEAR SCREEN *)
002005 2499 END; (* ** INITARRAYS ** *)
002031 2500
002031 2501
002031 2502
002031 2503
002031 2504
002031 2505
```



```

002011 2526 (*$L'PROCEDURES FOR BLOCKS'*)
002011 2527 (*-----*)
002011 2528 =
002011 2529 =
002011 2530 =
002011 2531 = BBBB BB LL 00000 CCCCC KK KK SSSSSS =
002011 2532 = BB BB LL 00 00 CC CC KK KK SS =
002011 2533 = BB BB LL 00 00 CC KK KK SS =
002011 2534 = BBBB BB LL 00 00 CC KKKK SSSSS =
002011 2535 = BB BB LL 00 00 CC KKKK SS =
002011 2536 = BB BB LL 00 00 CC CC KK KK SS =
002011 2537 = BBBB BB LLLLLL 00000 CCCCC KK KK SSSSSS =
002011 2538 =
002011 2539 =
002011 2540 =
002011 2541 (*-----*)
002011 2542 (*=====*)
002011 2543 PROCEDURE CREATEDLOCK (
000002 2544   VAR ID : INTEGER; (* BLOCK ID, FROM 1 TO 20 *)
000003 2545   X1,XL : INTEGER; (* ORIGIN AND LENGTH IN HORIZONTAL DIRECTION *)
000005 2546   Y1,YL : INTEGER; (* ORIGIN AND LENGTH IN VERTICAL DIRECTION *)
000007 2547   M1,M2 : METRIC; (* METRIC FOR X1Y1 AND XLENYLEN.
000011 2548   THE METRICS ARE IN TERMS OF DOTS, CHARACTERS,
000011 2549   AND FRACTIONS (0..100) OF A SCREEN *)
000011 2550   CENTERING : ADJ; (* WHERE THE ORIGIN IS. CFNTER,LL,LR,UL,UR *)
000012 2551   OUTLINE : INTEGER; (* THE THICKNESS OF THE OUTLINE. + GROWS INWARD.
000013 2552   - GROWS OUTWARD. *)
000013 2553   OVERFLOW : SCROLLTYPE; (* TELLS HOW TO HANDLE END OF PAGE CONDITION.
000014 2554   PRESENTLY SCROLL, NOSCROLL *)
000014 2555   OVERFLOW : WRAPTYPE; (* TELLS HOW TO HANDLE END OF LINE.
000015 2556   EITHER WRAP OR NOWRAP. *)
000015 2557   CHSET,ALICHSET : CHSETTYPE; (* THE DEFAULT AND ALTERNATE CHARACTER
000017 2558   SETS FOR THIS BLOCK.
000017 2559   PRESENTLY STANDARD, ALTERNATE *)
000017 2560   VAR ERROR : ERRORTYPE (* ERRORS.
000020 2561   BLOCKNOTNESTED,
000020 2562   TOOMANYBLOCKS,
000020 2563   BLOCKCPPSCREEN *) );
000020 2564
000020 2565 (* THIS PROCEDURE CREATES VIRTUAL TERMINALS.
000020 2566   YOU SHOULD BE ABLE TO OPERATE ON A VIRTUAL TERMINAL THE SAME WAY
000020 2567   YOU WOULD OPERATE ON A REAL ONE. THUS, YOU MUST SPECIFY HOW
000020 2568   TEXT BEYOND THE END OF THE VIRTUAL SCREEN IS HANDLED. THIS
000020 2569   IS DONE BY SPECIFYING OVERFLOW. YOU MUST ALSO STATE THE SIZE
000020 2570   AND LOCATION OF THE VIRTUAL TERMINAL. NOTICE THAT THE SCREEN
000020 2571   AREA IS 512X512 DOTS. IF METRIC IS DOTS, THEN THE SIZE UNITS RANGE
000020 2572   FROM (0..511). YOU MUST ALSO STATE HOW THE VIRTUAL TERMINAL WILL BE
000020 2573   OUTLINED. NO OUTLINE AT ALL IS ACHIEVED BY SETTING OUTLINE TO 0.
000020 2574   THE PARAMETER CENTERING SPECIFIES WHERE THE USER ORIGIN IS.
000020 2575   YOU HAVE THE OPTION OF THE FOUR CORNERS OR THE CENTER.
000020 2576   BLOCKS CAN BE NESTED. IN PARTICULAR BLOCK 0 IS THE FULL
000020 2577   SCREEN. SOME OPERATIONS ON BLOCKS AFFECT ANY NESTED BLOCKS
000020 2578   ONE SUCH OPERATION IS THE CLEAR OPERATION. TO CLEAR THE SCREEN,
000020 2579   YOU DO THE CLEAR OPERATION ON BLOCK 0. LIKE A STANDARD TERMINAL,
000020 2580   VIRTUAL TERMINALS HAVE STANDARD AND ALTERNATE CHARACTER
000020 2581   SETS ONE OF THE OPERATIONS WHEN DISPLAYING TEXT IS TO
  
```

```

000020 2563   DISPLAY TEXT IN THE SPECIFIED ALTERNATE SET.
000020 2564   *)
000020 2565
000020 2566   TYPE
000020 2567     PARRAY = ARRAY[1..BLOCKMAX] OF INTEGER;
000020 2568
000020 2569   VAR
000020 2570     X,XLEN,Y,YLEN : INTEGER;
000020 2571     PCNT : INTEGER;
000020 2572     PROGENY : PARRAY;
000020 2573     XS,YS : INTEGER;
000020 2574     XE,YE : INTEGER;
000020 2575     TID,TID1 : INTEGER;
000020 2576     INSIDE : NESTTYPE;
000020 2577     I : INTEGER;
000020 2578     J : INTEGER;
000020 2579     PTX,PTY,LTX,LTY : INTEGER;
000020 2580
000020 2581   FUNCTION STATUS (
000020 2582     VAR TID : INTEGER;
000020 2583     ID : INTEGER;
000020 2584     VAR PROGENY : PARRAY;
000020 2585     VAR PCNT : INTEGER
000020 2586   ) : NESTTYPE;
000020 2587
000020 2588   VAR
000020 2589     TID1 : INTEGER;
000020 2590     XS,YS,XE,YE : INTEGER;
000020 2591     XS1,YS1,XE1,YE1 : INTEGER;
000020 2592
000020 2593   BEGIN
000020 2594     PCNT := 0;
000020 2595     TID1 := TID;
000020 2596     REPEAT
000020 2597       XS := DSSPTR^.BLK[ID].XS;
000020 2598       YS := DSSPTR^.BLK[ID].YS;
000020 2599       XE := DSSPTR^.BLK[ID].XE;
000020 2600       YE := DSSPTR^.BLK[ID].YE;
000020 2601       XS1 := DSSPTR^.BLK[TID].XS;
000020 2602       YS1 := DSSPTR^.BLK[TID].YS;
000020 2603       XE1 := DSSPTR^.BLK[TID].XE;
000020 2604       YE1 := DSSPTR^.BLK[TID].YE;
000020 2605       INSIDE := NEST(XS,XE,YS,YE,
000020 2606         XS1,XE1,YS1,YE1);
000020 2607       TID1 := DSSPTR^.BLK[TID].NEIGHBORS;
000020 2608       IF INSIDE = REVNESTED THEN BEGIN
000020 2609         PCNT := PCNT + 1;
000020 2610         PROGENY[PCNT] := TID;
000020 2611       END;
000020 2612     UNTIL (TID1 = BLOCKMAX + 1) OR
000020 2613       (INSIDE = OVERLAP) OR
000020 2614       (INSIDE = NESTED) );
000020 2615
000020 2616   IF INSIDE = NESTED THEN BEGIN
000020 2617     IF DSSPTR^.BLK[TID].INSIDERS <> 0 THEN BEGIN
000020 2618       TID := DSSPTR^.BLK[TID].INSIDERS;
000020 2619       INSIDE := STATUS(TID,ID,PROGENY,PCNT);
  
```



```

000214 2677      (* MAKE PROGFNY INSIDERS *)
000214 2678      (* ALSO, IN THIS CASE, DON'T UPDATE TARGARRAY! *)
000214 2679      (* FILLED IN LATER *)
000214 2680      END;
000214 2681      END;
000214 2682      END;
000214 2683      IF INSIDE <> OVERLAP THEN BEGIN
000217 2684          IF OUTLINE <> 0 THEN BEGIN
000221 2685              DRAWBOX(XS,XLEN,YS,YIFN,DOTS,DOTS,LL,OUTLINE,DSPNORM,DSPNO);
000235 2686          END;
000235 2687          DSSPTR^.BLK[ID].INUSE := TRUE;
000246 2688          DSSPTR^.BLK[ID].XS := XS;
000257 2689          DSSPTR^.BLK[ID].XE := XS + XLEN;
000267 2690          DSSPTR^.BLK[ID].YS := YS;
000277 2691          DSSPTR^.BLK[ID].YE := YS + YLEN;
000310 2692          DSSPTR^.BLK[ID].OVERFLOW := OVERFLOW;
000322 2693          DSSPTR^.BLK[ID].CHSET := CHSET;
000334 2694          DSSPTR^.BLK[ID].OUTLINE := OUTLINE;
000344 2695          DSSPTR^.BLK[ID].ALCHSET := ALCHSET;
000356 2696          DSSPTR^.BLK[ID].TARGS := NIL;
000366 2697          IF OUTLINE > 0 THEN BEGIN
000370 2698              DSSPTR^.BLK[ID].TEXTORGX := XS + OUTLINE + 4;
000402 2699              DSSPTR^.BLK[ID].TEXTORGY := DSSPTR^.BLK[ID].YE - (OUTLINE + 22);
000423 2700              DSSPTR^.BLK[ID].TPXTENDX := DSSPTR^.BLK[ID].XE - (OUTLINE + 12);
000443 2701              DSSPTR^.BLK[ID].TEXTENDY := DSSPTR^.BLK[ID].YS + OUTLINE + 6;
000463 2702          END
000463 2703          ELSE BEGIN
000464 2704              DSSPTR^.BLK[ID].TPXTORGX := XS + 4;
000475 2705              DSSPTR^.BLK[ID].TEXTORGY := DSSPTR^.BLK[ID].YE - 22;
000515 2706              DSSPTR^.BLK[ID].TEXTENDX := DSSPTR^.BLK[ID].YE - 12;
000534 2707              DSSPTR^.BLK[ID].TPXTENDY := DSSPTR^.BLK[ID].YS + 6;
000553 2708          END;
000553 2709          DSSPTR^.BLK[ID].CURLINE := 1;
000564 2710          DSSPTR^.BLK[ID].LINELEN := ((DSSPTR^.BLK[ID].TEXTENDY - DSSPTR^.BLK[ID].TEXTORGX) DIV 8) + 1;
000614 2711          DSSPTR^.BLK[ID].MAXLINE := ((DSSPTR^.BLK[ID].TEXTORGX - DSSPTR^.BLK[ID].TEXTENDY) DIV 16) + 1;
000644 2712          (* SETCOORD(DSSPTR^.BLK[ID].TPXTORGX,DSSPTR^.BLK[ID].TEXTORGX,DOTS); *)
000644 2713          DSSPTR^.BLK[ID].TEXEX := DSSPTR^.BLK[ID].TEXTORGX;
000663 2714          DSSPTR^.BLK[ID].TARGPX := DSSPTR^.BLK[ID].TEXPX;
000701 2715          DSSPTR^.BLK[ID].TEXPY := DSSPTR^.BLK[ID].TPXTORGX + 16;
000717 2716          DSSPTR^.BLK[ID].PLOTORGX := DSSPTR^.BLK[ID].TEXTORGX;
000736 2717          DSSPTR^.BLK[ID].PLOTORGY := DSSPTR^.BLK[ID].TEXTENDY;
000754 2718          DSSPTR^.BLK[ID].WMODE := DSPNORM;
000764 2719          DSSPTR^.BLK[ID].HEAD := NIL;
000773 2720          DSSPTR^.BLK[ID].CURPTR := NIL;
001003 2721          NEW(DSSPTR^.BLK[ID].TARGS);      (* ALLOCATE THE TARGET ARRAY *)
001014 2722          FOR I := 1 TO TARGMAX DO BEGIN
001020 2723              DSSPTR^.BLK[ID].TARGS[I] := NIL;
001037 2724          END;
001043 2725
001043 2726
001043 2727      (* FIND ALL POSSIBLE TARGETS TO THIS BLOCK *)
001043 2728      PTX := DSSPTR^.BLK[ID].XS DIV 32;
001055 2729      PTY := DSSPTR^.BLK[ID].YS DIV 32;
001066 2730      LTX := (DSSPTR^.BLK[ID].XE DIV 32) - 1;
001100 2731      LTY := (DSSPTR^.BLK[ID].YE DIV 32) - 1;
001111 2732      IF (LTX >= PTX) AND (LTY >= PTY) THEN BEGIN
001113 2733          FOR I := PTX TO LTX DO BEGIN

```

```

001117 2734           FOR J := PTY TO LTY DO BEGIN
001124 2735             DSSPTR^.TARGA[RAY[I,J]].BLOCKID := ID;
001140 2736           END;
001143 2737           END;
001150 2738           END;
001150 2739           ERROR := NOERROR;
001152 2740           FND
001152 2741           ELSE BEGIN
001153 2742             ERROR := BLOCKNOTNESTED;
001155 2743           END;
001155 2744           END
001155 2745           ELSE BEGIN
001156 2746             ERROR := BLOCKOFFSCREEN;
001160 2747           END;
001160 2748           END
001160 2749           ELSE BEGIN
001161 2750             (* CASE OF TOO MANY BLOCKS *)
001161 2751             ERROR := TOOMANYBLOCKS;
001163 2752           END;
001163 2753           END;
001265 2754           (* OF CREATEBLOCK *)
001265 2755           END;
001265 2756
001265 2757
001265 2758
001265 2759 (*-----*)
001265 2760 PROCEDURE DESTBLOCK (
000002 2761   ID : INTEGER; (* WHICH BLOCK TO DESTROY*)
000003 2762   VAR ERROR : ERRORTYPE (* IS SET TO NOSUCHBLOCK IF YOU
000004 2763                       ATTEMPT TO DESTROY A BLOCK NOT CREATED,
000004 2764                       OR THE ID IS WIERD *) );
000004 2765
000004 2766 (* THIS ROUTINE DOES THE OBVIOUS TASK OF GETTING RID OF BLOCKS.
000004 2767 ALL NESTED BLOCKS ARE DESTROYED.
000004 2768 TO DESTROY ALL BLOCKS, DESTROY BLOCK 0 *)
000004 2769
000004 2770 VAR
000004 2771   TID,TID1 : INTEGER;
000006 2772   CONT : BOOLEAN;
000007 2773   ID1 : INTEGER;
000010 2774   I,J : INTEGER;
000012 2775   XS,YS,XE,YE : INTEGER;
000016 2776   PTX,PTY,LTX,LTY : INTEGER;
000022 2777
000022 2778   PROCEDURE KILLBLOCK (
000002 2779     ID : INTEGER
000003 2780   );
000003 2781
000003 2782   VAR
000003 2783     PTR,PTR1 : DSPBPTR;
000005 2784     I,J : INTEGER;
000007 2785     TIL,TID1 : INTEGER;
000011 2786
000011 2787   BEGIN
000011 2788     DSSPTR^.BLK[ID].INUSE := FALSE;
000015 2789
000015 2790   (* DESTROY TARGETS *)

```

```

000015 2791   FOR I := 0 TO 15 DO BEGIN
000020 2792     FOR J := 0 TO 15 DO BEGIN
000024 2793       IF ESSPTR^.TARGARRAY[I,J].TARG <> NIL THEN BEGIN
000036 2794         IF DSSPTR^.TARGARRAY[I,J].BLOCKID = ID THEN DSSPTR^.TARGARRAY[I,J].TARG := NIL;
000057 2795       END;
000057 2796     END;
000064 2797   END;
000070 2798   FOR I := 1 TO TARGMAX DO BEGIN
000073 2799     IF ESSPTR^.BLK[ID].TARG^[I] <> NIL THEN BEGIN
000113 2800       DISPOSE(DSSPTR^.BLK[ID].TARG^[I]);
000131 2801     END;
000131 2802   END;
000136 2803   DISPOSE(DSSPTR^.BLK[ID].TARG);
000150 2804   (* DESTROY TEXT BUFFERS *)
000150 2805   PTR := DSSPTR^.BLK[ID].HEAD;
000162 2806   WHILE PTR <> NIL DO BEGIN
000165 2807     PTR1 := PTR;
000166 2808     DISPOSE(PTR1);
000170 2809     PTR := PTR^.NEXT;
000176 2810   END;
000177 2811   (* REMOVE FROM DISABLED LIST *)
000177 2812   IF DISABLED(ID,ID1) THEN DSSPTR^.DISARM[ID1] := 0;
000213 2813   (* DESTROY ALL NESTED BLOCKS *)
000213 2814   TID := DSSPTR^.BLK[ID].INSIDERS;
000225 2815   DSSPTR^.BLK[ID].INSIDERS := BLOCKMAX + 1;
000234 2816   WHILE TID <> (BLOCKMAX + 1) DO BEGIN
000237 2817     TID1 := DSSPTR^.BLK[TID].NEIGHBORS;
000247 2818     KILLBLOCK(TID);
000251 2819     DSSPTR^.BLK[ID].NEIGHBORS := BLOCKMAX + 1;
000262 2820     TID := TID1;
000264 2821   END;
000265 2822   END;
000305 2823   (* OF KILLBLOCK *)
000305 2824   BEGIN
000305 2825     CONT := ID = 0;
000011 2826     IF NOT CONT THEN CONT := DSSPTR^.BLK[ID].INUSE;
000024 2827     IF CONT THEN BEGIN
000026 2828       (* CLEAR SCREEN THAT BLOCK COVERS *)
000026 2829       IF ID = 0 THEN BEGIN
000030 2830         MODE(DSPTEXT,DSPNONE,DSPNORM,ISPN0);
000035 2831         PUTCH(OUTPUT,CLEAR);
000037 2832       END
000037 2833       ELSE BEGIN
000040 2834         IF DSSPTR^.BLK[ID].OUTLINE < 0 THEN BEGIN
000052 2835           XS := DSSPTR^.BLK[ID].XS + DSSPTR^.BLK[ID].OUTLINE;
000070 2836           YS := DSSPTR^.BLK[ID].YS + DSSPTR^.BLK[ID].OUTLINE;
000106 2837           XF := DSSPTR^.BLK[ID].XF - DSSPTR^.BLK[ID].OUTLINE;
000125 2838           YF := DSSPTR^.BLK[ID].YF - DSSPTR^.BLK[ID].OUTLINE;
000143 2839         END
000143 2840         ELSE BEGIN
000144 2841           XS := DSSPTR^.BLK[ID].XS;
000144 2842           YS := DSSPTR^.BLK[ID].YS;
000154 2843           XF := DSSPTR^.BLK[ID].XF;
000163 2844           YF := DSSPTR^.BLK[ID].YF;

```

```

000173 2849       YF := DSSPTR^.BLK[ID].YE;
000203 2849       END;
000203 2850       FOR I := XS TO XE DO BEGIN
000210 2851         SPTCOORD(I,YE,DOTS);
000214 2852         DRAWLINE(I,YE,DOTS,ERASE,DSPNO);
000222 2853       END;
000227 2854       END;
000227 2855
000227 2856       (* KILL DECENDANTS *)
000227 2857       IF ID <> 0 THEN BEGIN
000231 2858         KILLBLOCK(ID);
000233 2859
000233 2860         (* FIND ALL POSSIBLE TARGETS TO THIS BLOCK *)
000233 2861         PTX := ISSPTR^.BLK[ID].XS DIV 32;
000245 2862         PTY := DSSPTR^.BLK[ID].YS DIV 32;
000256 2863         LTX := (DSSPTR^.BLK[ID].XE DIV 32) - 1;
000270 2864         LTY := (DSSPTR^.BLK[ID].YE DIV 32) - 1;
000301 2865         IF (LTX >= PTX) AND (LTY >= PTY) THEN BEGIN
000303 2866           FOR I := PTX TO LTX DO BEGIN
000307 2867             FOR J := PTY TO LTY DO BEGIN
000314 2868               DSSPTR^.TARGARRAY[I,J].BLOCKID := 0;
000325 2869             END;
000331 2870           END;
000336 2871         END;
000336 2872       END
000336 2873     ELSE BEGIN
000337 2874       TID := DSSPTR^.BLOCK0;
000345 2875       DSSPTR^.BLOCK0 := BLOCKMAX + 1;
000352 2876       WHILE TID <> BLOCKMAX + 1 DO BEGIN
000355 2877         TID1 := DSSPTR^.BLK[TID].NEIGHBORS;
000365 2878         KILLBLOCK(TID);
000367 2879         DSSPTR^.BLK[TID].NEIGHBORS := BLOCKMAX + 1;
000400 2880         TID := TID1;
000402 2881       END;
000403 2882
000403 2883       (* RESET BLOCK ID IN TARGARRAY *)
000403 2884       FOR I := 0 TO 15 DO BEGIN
000407 2885         FOR J := 0 TO 15 DO BEGIN
000413 2886           DSSPTR^.TARGARRAY[I,J].BLOCKID := 0;
000424 2887         END;
000430 2888       END;
000434 2889       END;
000434 2890       EPROR := NOERROR;
000436 2891     END
000436 2892   ELSE BEGIN
000437 2893     ERROR := NOSUCHBLOCK;
000441 2894   END;
000441 2895 END; (* OF DESTROY BLOCK *)
000503 2896
000503 2897
000503 2898
000503 2899
000503 2900 (*=====*)
000503 2901 PROCEDURE CLEARBLOCK (
000503 2902   ID : INTEGER; (* BLOCK ID *)
000503 2903   VAR ERROR : ERRORTYPE (* NOSUCHBLOCK *) );
000503 2904

```

```

000004 2905 (* THIS PROCEDURE ERASES THE SCREEN OF THE VIRTUAL TERMINAL.
000004 2906 THIS OPERATION IS ALLOWED ON BLOCK 0. ALL NESTED BLOCKS
000004 2907 ARE AFFECTED. *)
000004 2908 BEGIN
000004 2909 END;
000012 2910
000012 2911
000012 2912 (*=====*)
000012 2913 PROCEDURE UNDOBLOCK (
000002 2914 ID : INTEGER; (* BLOCK ID *)
000003 2915 VAR ERROR : ERRORTYPE (* NOSUCHBLOCK *) );
000004 2916
000004 2917 (* THIS IS FOR A LESS HARSH CLEANUP OF A BLOCK. THIS DOES
000004 2918 NOT AFFECT THE NESTED BLOCKS. YOU CAN USE THIS TO SCRAP
000004 2919 A BLOCK AND READY IT FOR NEW TEXT W/O ACTUALLY DESTROYING
000004 2920 IT. *)
000004 2921
000004 2922 VAR
000004 2923 PTR1,PTR2 : TEXTPTR;
000010 2924 PX,PY : INTEGER;
000012 2925
000012 2926 BEGIN
000012 2927 IF DSSPTR^.BLK[ID].INUSE THEN BEGIN
000015 2928 ERROR := NOERROR;
000016 2929 PTR1.POS := DSSPTR^.BLK[ID].HEAD^.POS;
000034 2930 PTR1.BUF := DSSPTR^.BLK[ID].HEAD;
000044 2931 PY := DSSPTR^.BLK[ID].TEXTORG;
000055 2932 PX := DSSPTR^.BLK[ID].TEXTORGX;
000065 2933 IF PTR1.BUF <> NIL THEN BEGIN
000067 2934 DSPLINE(ID,PTR1,PTR2,UNDO,PX,PY);
000075 2935 WHILE PTR2.BUF <> NIL DO BEGIN
000100 2936 PTR1 := PTR2;
000102 2937 PY := PY - 16;
000104 2938 PX := DSSPTR^.BLK[ID].TEXTORGX;
000115 2939 DSPLINE(ID,PTR1,PTR2,UNDO,PX,PY);
000123 2940
000123 2941 (*SCRAP TEXT BUFFERS *)
000123 2942 IF PTR1.BUF <> PTR2.BUF THEN DISPOSE(PTR1);
000127 2943 END;
000130 2944 END;
000130 2945 END;
000130 2946 END;
000142 2947
000142 2948
000142 2949 (*=====*)
000142 2950 PROCEDURE REDOBLOCK (
000002 2951 ID : INTEGER; (* BLOCK ID *)
000003 2952 VAR ERROR : ERRORTYPE (* NOSUCHBLOCK *) );
000004 2953
000004 2954 (* THIS PROCEDURE ATTEMPTS TO RECONSTRUCT THE STUFF IN A BLOCK
000004 2955 THIS PROCEDURE IS MEANT TO BE USED IF YOU GET TRANSMISSION
000004 2956 ERRORS. *)
000004 2957 BEGIN
000004 2958 END;
000012 2959
000012 2960
000012 2961 (*=====*)

```



```

000012 2962  PROCEDURE DISARMBLOCK (
000022 2963      ID : INTEGER;    (* BLOCK ID *)
000033 2964      VAR ERROR : ERRORTYPE    (* NOSUCHBLOCK *) );
000044 2965
000054 2966  (* THIS PROCEDURE IS FOR DISARMING ALL THE TOUCH TARGETS
000064 2967    IN A BLOCK. GOOD FOR MULTIPLE QUESTIONS ON ONE SCREEN *)
000074 2968
000084 2969  VAR
000094 2970      I : INTEGER;
000105 2971
000115 2972  BEGIN
000125 2973      IF DSSPTR^.BLK[ID].INUSE THEN BEGIN
000135 2974          I := 1;
000145 2975          WHILE DSSPTR^.DISARM[I] <> 0 DO BEGIN
000155 2976              I := I + 1;
000165 2977          END;
000175 2978          DSSPTR^.DISARM[I] := ID;
000185 2979          ERROR := NOERROR;
000195 2980      END
000205 2981      ELSE BEGIN
000215 2982          ERROR := NOSUCHBLOCK;
000225 2983      END;
000235 2984  END;
000245 2985
000255 2986
000265 2987
000275 2988
000285 2989  (* ===== *)
000295 2990  PROCEDURE REARMBLOCK (
000305 2991      ID : INTEGER;    (* BLOCK ID *)
000315 2992      VAR ERROR : ERRORTYPE    (* NOSUCHBLOCK *) );
000325 2993
000335 2994  (* THIS PROCEDURE REARMS BLOCKS. VISUALLY, THIS MEANS
000345 2995    MAKING TOUCHED TARGETS LOOK LIKE NEW. *)
000355 2996
000365 2997  VAR
000375 2998      I : INTEGER;
000385 2999      MORE : BOOLEAN;
000395 3000
000405 3001  BEGIN
000415 3002      MORE := TRUE;
000425 3003      I := 1;
000435 3004      IF DSSPTR^.BLK[ID].INUSE THEN BEGIN
000445 3005          WHILE (I <= BLOCKMAX) AND MORE DO BEGIN
000455 3006              MORE := DSSPTR^.DISARM[I] <> ID;
000465 3007              I := I + 1;
000475 3008          END;
000485 3009          IF I <> 0 THEN DSSPTR^.DISARM[I] := 0;
000495 3010
000505 3011          (* MAKE TTS LOOK LIKE NEW *)
000515 3012      END
000525 3013      ELSE BEGIN
000535 3014          ERROR := NOSUCHBLOCK;
000545 3015      END;
000555 3016  END;
000565 3017
000575 3018
  
```

67

PASCAL COMPILER - E.T.R. ZURICH / UNIVERSITY OF MINNESOTA.
DISPLAY MODULE

PASCAL 6000 V3.0.0. 80/11/17. 00.56.01.
NOS 1.4 (80/04/21) PAGE 62

000063 3019
000063 3020

```

000063 3021 (*$L'PROCEDURES FOR TEXT'*)
000063 3022 (*-----*)
000063 3023 =
000063 3024 =
000063 3025 =
000063 3026 =
000063 3027 =          TTTTTT  EEEEEEE  XX  XX  TTTTTT
000063 3028 =          TT      EE          XX XX  TT
000063 3029 =          TT      EE          XXX   TT
000063 3030 =          TT      EEEEE          XIX   TT
000063 3031 =          TT      EE          XXX   TT
000063 3032 =          TT      EE          XX XX  TT
000063 3033 =          TT      EEEEEEE  XX  XX  TT
000063 3034 =
000063 3035 =
000063 3036 =
000063 3037 (*-----*)
000063 3038
000063 3039
000063 3040 (*-----*)
000063 3041 PROCEDURE DTEXT (
000063 3042     ID : INTEGER; (* ID OF BLOCK *)
000063 3043     VAR INTXT : TEXT; (* FILE OF INPUT TEXT *)
000063 3044     VAR ERROR : ERRORTYPE (* ERRORS.
000063 3045                          CUTOFFOUNDS,
000063 3046                          NOTEMORY,
000063 3047                          LABELTOOLONG,
000063 3048                          COMMANDERROR,
000063 3049                          NOSUCHBLOCK *) );
000063 3050
000063 3051 (* THIS ROUTINE IS HOW YOU FILL A VIRTUAL TERMINAL WITH STUFF.
000063 3052    YOU FIRST WRITE TEXT TO INTXT, THEN YOU PASS THIS TEXTFILE
000063 3053    TO THIS DISPLAY ROUTINE. JUST AS IN A REAL TERMINAL WITH
000063 3054    GRAPHICS CAPABILITY, THERE ARE ESCAPE SEQUENCES TO DO GRAPHICS.
000063 3055    THERE IS ALSO A SET OF ESCAPE SEQUENCES FOR PLACEMENT OF TEXT AS
000063 3056    WELL AS THE DEFINITION OF TOUCH TARGETS. ALSO SUPPORTED ARE ALTERNATE
000063 3057    CHARACTER SETS. BELOW IS THE INITIAL SET OF COMMANDS:
000063 3058
000063 3059 GRAPHICS
000063 3060 -----
000063 3061 \GOX Y SET GRAPHICS ORIGIN X DOTS ACROSS AND Y DOTS AWAY FROM
000063 3062     BLOCK ORIGIN.
000063 3063 \SCX Y SET CURSOR X DOTS HORIZONTALLY AND Y DOTS
000063 3064     VERTICALLY FROM THE TEXT ORIGIN. IF X OR Y ARE THE CHARACTERS
000063 3065     "999" THEN THE CURRENT POSITION IS USED.
000063 3066 \LNX Y DRAW A LINE FROM THE LAST POSITION TO (X,Y).
000063 3067 \PTX Y DRAW A POINT AT (X,Y).
000063 3068 \CH# X Y DRAW A CHARACTER WHOSE ORD IS # AT (X,Y). (NORMAL SET)
000063 3069 \CA# X Y DRAW A CHARACTER WHOSE ORD IS # AT (X,Y). (ALTERNATE SET)
000063 3070
000063 3071 TEXT
000063 3072 ----
000063 3073 \MLX SET LEFT MARGIN X CHARACTERS AWAY FROM TEXT ORIGIN.
000063 3074 \MRX SET RIGHT MARGIN X CHARACTERS AWAY FROM TEXT ORIGIN.
000063 3075 \MUY SET UPPER MARGIN.
000063 3076 \MDY SET BOTTOM MARGIN.
000063 3077 \OVS OVERSTRIKE THE LAST CHARACTER WITH THE CHARACTER S
  
```

```

000005 3078 \OA# OVERSTRIKE THE LAST CHARACTER WITH THE ALTERNATE CHARACTER SET
000005 3079 CHARACTER WHOSE ORD IS #
000005 3081 \S# SUBSCRIPT THE CHARACTER #.
000005 3081 \SA# SUBSCRIPT THE ALTERNATE CHARACTER WHOSE ORD IS #.
000005 3082 \SS# SUPERSCRIPT THE CHARACTER #.
000005 3083 \UA# SUPERSCRIPT THE ALTERNATE CHARACTER WHOSE ORD IS #.
000005 3084 \UITEXT\DE UNDERLINE TEXT BETWEEN \OL & \UE
000005 3085 \NP,\ND,\RP,\RD SET THE MODE OF DISPLAY.
000005 3086 N IS NORMAL VIDEO
000005 3087 R IS REVERSE VIDEO
000005 3088 P IS PROTECTED (I.E. OVERSTRIKE)
000005 3089 D IS DESTRUCTIVE
000005 3090 \CR CARRIAGE RETURN. THIS IS GOOD FOR DOING FANCY STUFF WITH OVERSTRIKES.
000005 3091 \LF LINE FEED. GO DOWN A LINE W/O A RETURN.
000005 3092 \EL END OF LINE.
000005 3093 \AL####S\NR ESCAPE TO ALTERNATE CHARACTER SET. SINCE THE ALTERNATE
000005 3094 SET IS MAPPED ONTO ALL THE ASCII SET, YOU REPRESENT THESE
000005 3095 CHARACTERS BY THEIR ORDS (SEPPRATED BY SPACES, BASE 10).
000005 3096
000005 3097 TOUCH TARGETS
000005 3098 -----
000005 3099
000005 3100 \TT#LABEL\TE CREATE A TOUCH TARGET WHOSE TARGET ID IS # WITH LABEL
000005 3101 INSIDE IT. THIS TARGET GETS FAT AFTER GETTING TOUCHED.
000005 3102 \TU#LABEL\TE CREATE A TOUCH TARGET WHICH TURNS INTO THE UNDERLINED
000005 3103 LABEL AFTER BEING TOUCHED.
000005 3104
000005 3105 (NOTE: \ IS THE ESCAPE CHARACTER AND IS AVAILABLE TO YOU IN THE
000005 3106 CONSTANT ESCAPE)
000005 3107 *)
000005 3108
000005 3109
000005 3110
000005 3111 VAR
000006 3112 INTRFP : CRNG;
000006 3112 I : INTEGER;
000007 3113 CHR : CHAR;
000010 3114
000010 3115
000010 3116
000010 3117
000010 3118
000003 3119 FUNCTION CNVT1 (VAR INTEXT : TEXT; (*IPYT FILE THAT IS BEING CONVERTED *)
000004 3120 VAR ERROR : ERRORTYPE (* POSSIBLE ERROR IS COMMAND ERROR *)
000005 3121 ) : INTEGER;
000005 3122
000005 3123 (* THIS FUNCTION IS USED TO LOOK UP THE COMMAND FOR SOME
000005 3124 DISPLAY ACTION. IT CONVERTS IT TO AN INTEGER >127 WHICH
000005 3125 IS USED AS REPRESENTING THE COMMAND IN THE INTERNAL
000005 3126 DISPLAY BUFFERS IN THE PROCEDURE DSPTXT *)
000005 3127
000005 3128 (* USES LOOKUP = PACKED ARRAY[1..NCOMMANDS,1..3] OF 0..511 AND
000005 3129 THE CONSTANT NCOMMANDS= NUMBER OF COMMANDS *)
000005 3130
000005 3131 VAR
000006 3132 CH : INTEGER;
000006 3132 CH1 : INTEGER;
000007 3133 CHA : CHAR;
000010 3134 CH1A : CHAR;

```

```

00011 3135      I : INTEGER;
00012 3136      NOTFOUND : BOOLEAN;
00013 3137
00014 3138  BEGIN
00015 3139      NOTFOUND := TRUE;
00016 3140      READ(INTEXT,CHA);
00017 3141      CH := CNVT(INTEXT,CHA);
00018 3142      READ(INTEXT,CH1A);
00019 3143      CH1 := CNVT(INTEXT,CH1A);
00020 3144      (*DO LOOKUP *)
00021 3145      I := 1;
00022 3146      WHILE (I <= NCOMMANDS) AND NOTFOUND DO BEGIN
00023 3147          IF ( (DSSPTR^.LOOKUP[I,1] = CH) AND
00024 3148              (DSSPTR^.LOOKUP[I,2] = CH1) ) THEN NOTFOUND := FALSE;
00025 3149          I := I + 1;
00026 3150      END;
00027 3151      (* OF WHILE *)
00028 3152      IF NOTFOUND THEN BEGIN
00029 3153          ERROR := COMMANDERROR;
00030 3154          CNVT1 := 0;
00031 3155          (* AN ASCII NULL *)
00032 3156      END
00033 3157      ELSE BEGIN
00034 3158          CNVT1 := DSSPTR^.LOOKUP[I-1,3];
00035 3159      END;
00036 3160      (* OF CNVT1 *)
00037 3161
00038 3162  PROCEDURE PUTICH (
00039 3163      CH : CRNG ;      (* INTEGER REPRESENTATION OF CHAR *)
00040 3164      ID : INTEGER      (* BLOCK WHOSE BUFFERS TO STORE IN *)
00041 3165      );
00042 3166
00043 3167  (* THIS PROCEDURE STUFFS THE INTEGER REPRESENTATION OF A
00044 3168  CHARACTER IN THE TEXT BUFFERS OF BLOCK <ID>. THESE BUFFERS
00045 3169  ARE FIXED LENGTH ARRAYS & ARE ALLOCATED INCREMENTALLY AS
00046 3170  THEY ARE FILLED UP.
00047 3171  *)
00048 3172  VAR
00049 3173      NEWBUF : DSEBUFPTR;
00050 3174
00051 3175  BEGIN
00052 3176      IF DSSPTR^.BLK[ID].CURPTR = NIL THEN BEGIN
00053 3177
00054 3178          (*CASE OF NO BUFFERS AT ALL IN CURRENT USE *)
00055 3179          (* CURPTR IS SET TO NIL IN CREATERBLOCK *)
00056 3180          NEW(DSSPTR^.BLK[ID].CURPTR);
00057 3181          DSSPTR^.BLK[ID].CURPTR^.NEXT := NIL;
00058 3182          DSSPTR^.BLK[ID].CURPTR^.POS := 0;
00059 3183          DSSPTR^.BLK[ID].CURPTR^.EPOS := 0;
00060 3184          DSSPTR^.BLK[ID].HEAD := DSSPTR^.BLK[ID].CURPTR;
00061 3185          DSSPTR^.BLK[ID].LSTPTR.BUF := DSSPTR^.BLK[ID].CURPTR;
00062 3186          DSSPTR^.BLK[ID].LSTPTR.POS := DSSPTR^.BLK[ID].CURPTR^.POS;
00063 3187      END;
00064 3188      DSSPTR^.BLK[ID].CURPTR^.EPOS := DSSPTR^.BLK[ID].CURPTR^.EPOS + 1;
00065 3189      IF DSSPTR^.BLK[ID].CURPTR^.EPOS > TEXNOFLEN THEN BEGIN
00066 3190          DSSPTR^.BLK[ID].CURPTR^.EPOS := DSSPTR^.BLK[ID].CURPTR^.EPOS - 1;
00067 3191

```

```

000250 3192      (* TIME FOR A NEW BUFFER *)
000250 3193      NEW(NEWBUF);
000252 3194      DSSPTR^.BLK[ID].CURPTR^.NEXT := NEWBUF;
000267 3195      NEWBUF^.NEXT := NIL;
000274 3196      DSSPTR^.BLK[ID].CURPTR := NEWBUF;  (* POINT TO NEW BUFFER *)
000304 3197      DSSPTR^.BLK[ID].CURPTR^.POS := 0;
000320 3198      DSSPTR^.BLK[ID].CURPTR^.EPOS := 1;
000334 3199      END;
000334 3200
000334 3201      (* PLACE INTEGER REPRESENTATION IN BUFFER *)
000334 3202      DSSPTR^.BLK[ID].CURPTR^.AR[DSSPTR^.BLK[ID].CURPTR^.EPOS] := CH;
000376 3203      END;
000406 3204      , (* OF PUTICH *)
000406 3205
000406 3206
000406 3207
000406 3208
PROCEDURE INTEPRET (
000002 3209      INTREP : CRNG ; (* INTEGER REPRESENTATION OF CHARACTER *)
000003 3210      ID : INTEGER; (* BLOCK ID *)
000004 3211      ERROR : ERRORTYPE
000005 3212      );
000005 3213
(* THIS PROCEDURE IS FOR THE INITIAL CONVERSION OF NUMBERS
IN THE TEXTFILE BEING READ IN TO THEIR INTEGER REPRESENTATION.
IT ALSO DOES END OF LINE HANDLING *)
000005 3214
000005 3215
000005 3216
000005 3217
000005 3218
000005 3219
000011 3220
000014 3221
000015 3222
000016 3223
000052 3224
000052 3225
000052 3226
000015 3227
000017 3228
000017 3229
000017 3230
000027 3231
000034 3232
000041 3233
000042 3234
000042 3235
000042 3236
000055 3237
000062 3238
000067 3239
000074 3240
000075 3241
000075 3242
000075 3243
000102 3244
000107 3245
000110 3246
000110 3247
000110 3248

VAR
PTR, PTR1 : TEXTPTR;
X, Y, NUM : INTEGER;
CH1 : CHAR;
CH0 : INTEGER;
SCRATCH : TEXT;

BEGIN
  PUTICH(INTREP, ID);
  CASE INTREP OF

    SC, IY, PT, GO : BEGIN (* DOUBLE INTEGERS *)
      RFAD(INTEXT, X, Y);
      PUTICH(X, ID);
      PUTICH(Y, ID);
    END;

    CH, CA : BEGIN (* TRIPLE INTEGERS *)
      RFAD(INTEXT, CH0, X, Y);
      PUTICH(CH0, ID);
      PUTICH(X, ID);
      PUTICH(Y, ID);
    END;

    ML, MS, MU, MB, OA : BEGIN (* SINGLE INTEGERS *)
      READ(INTEXT, NUM);
      PUTICH(NUM, ID);
    END;

    AL : BEGIN (* N INTEGERS *)
      REWRITE(SCRATCH);

```

```

000112 3249      READ(INTEXT,CH1);
000122 3250      WHILE CH1 <> ESCAPE DO BEGIN
000125 3251          WRITE(SCRATCH,CH1);
000133 3252          READ(INTEXT,CH1);
000143 3253      END;      (* OF WHILE *)
000144 3254      WRITELN(SCRATCH);
000146 3255      RESET(SCRATCH);
000150 3256      REPEAT
000150 3257          READ(SCRATCH,NUM);
000153 3258          PUTCH(NUM,ID);
000160 3259      UNTIL FCLNS(SCRATCH);

000163 3260
000163 3261      (* GET REST OF END DELIMITER *)
000163 3262      INTREP := CNVT1(INTEXT,PPROR);
000172 3263      IF ERROR = NOERROR THEN BEGIN
000174 3264          INTERPRET(INTREP,IC,ERROR);      (* CALL ME AGAIN ! *)
000201 3265      END
000201 3266      ELSE BEGIN
000202 3267          PUTCH(NR,ID);      (* PUT DELIMITER IN ANYWAY *)
000206 3268      END;
000206 3269  END;      (* OF N NUMBERS CASE *)
000207 3270
000207 3271  EL,LI : BEGIN      (* NEXT LINE CASES *)
000207 3272      HANDLEEOL(ID);
000211 3273      PTR.BUF := DSSPTR^.BLK[ID].LSTPTR.BUF;
000223 3274      PTR.POS := DSSPTR^.BLK[ID].LSTPTR.POS;
000234 3275      DSPLINE(ID,PTR,PTR1,PRINT,DSSPTR^.BLK[ID].TEXPX,DSSPTR^.BLK[ID].TEXPY);
000260 3276      WHILE (PTR1.BUF <> DSSPTR^.BLK[ID].CURPTR) AND
000272 3277          (PTR1.POS <> DSSPTR^.BLK[ID].CURPTR.FPOS) DO BEGIN
000307 3278          PTR := PTR1;
000311 3279          HANDLEEOL(ID);
000313 3280          DSPLINE(ID,PTR,PTR1,PRINT,DSSPTR^.BLK[ID].TEXPX,DSSPTR^.BLK[ID].TEXPY);
000340 3281      END;
000341 3282      DSSPTR^.BLK[ID].LSTPTR := PTR1;
000352 3283  END;      (* OF NEXT LINE CASES *)
000353 3284
000353 3285  TT,TU : BEGIN      (* TOUCH TARGET CASES *)
000353 3286
000353 3287      (* DIVERT LABEL TO LABEL BUFFER *)
000353 3288      DSSPTR^.LABCTR := DSSPTR^.LABCTR + 1;
000367 3289      IF DSSPTR^.LABCTR <= TARGMAX THEN BEGIN      (* ONLY SO MUCH ROOM *)
000374 3290          NEW(DSSPTR^.LABARR[DSSPTR^.LABCTR]);      (* GET A NEW LABEL BUFFER *)
000407 3291          READ(INTEXT,NUM);      (* USER'S TARGET ID *)
000414 3292          PUTCH(NUM,ID);
000421 3293          PUTCH(DSSPTR^.LABCTR,ID);      (* STORE INDEX TO LABEL *)
000432 3294          I:=1;
000434 3295          READ(INTEXT,CH1);
000443 3296          WHILE (CH1 <> ESCAPE) AND ( I <= LABELLEN ) DO BEGIN
000450 3297              DSSPTR^.LABARR[DSSPTR^.LABCTR][I] := CH1;
000503 3298              I := I + 1;
000504 3299              READ(INTEXT,CH1);
000514 3300      END;
000515 3301      PUTCH(I-1,ID);      (* NUMBER OF CHARACTERS IN BUFFER *)
000522 3302      IF CH1 <> ESCAPE THEN BEGIN      (* THROW AWAY REST IF BUFFER IS TOO FULL *)
000525 3303          ERROR := LABELTOOLONG;
000526 3304          READ(INTEXT,CH1);
000536 3305          WHILE CH1 <> ESCAPE DO BEGIN
  
```

```

000541 3306      READ (INTEXT,CHR);
000550 3307      END;      (* OF WHILE *)
000551 3308      END;
000551 3309      INTREP := CNVT1 (INTEXT,ERROR);
000560 3310      IF ERROR = NOERROR THEN BEGIN
000562 3311          INTERPRET (INTREP,ID,ERROR);
000567 3312      END
000567 3313      ELSE BEGIN
000570 3314          PUTICH (TE,ID);
000574 3315      END;
000574 3316      END
000574 3317      ELSE BEGIN
000575 3318          ERROR := NOMEMORY;
000577 3319      END;
000577 3320      END;
000600 3321
000600 3322      OTHERWISE      (* NOTHING *)
000621 3323
000621 3324
000621 3325      END;      (* OF CASE *)
000621 3326
000621 3327      END;      (* OF INTERPRET *)
000645 3328
000645 3329
000645 3330
000645 3331      BEGIN      (*OF MAIN PART *)
000645 3332      IF DSSPTR^.BLK[ID].INUSE THEN BEGIN
000015 3333          DSSPTR^.LABCTR := 0;
000022 3334          READ (INTEXT,CHR);
000031 3335          WHILE (EOLN (INTEXT)) DO BEGIN      (* OMIT EMPTY LINES *)
000033 3336              READLN (INTEXT);
000036 3337          END;
000037 3338
000037 3339      (* GET STUFF UNTIL END OF FILE *)
000037 3340      WHILE (NOT EOP (INTEXT)) DO BEGIN
000041 3341          INTREP := CNVT (INTEXT,CHR);
000047 3342          IF INTREP = ESCCH THEN BEGIN      (* WAS A COMMAND, SO LOOK IT UP *)
000051 3343              INTREP := CNVT1 (INTEXT,ERROR);
000056 3344              IF ERROR = NOERROR THEN BEGIN      (* DON'T CONTINUE IF COMMAND WAS FUNNY *)
000060 3345                  INTERPRET (INTREP,ID,ERROR);      (* DO SOME INITIAL SEMANTICS *)
000065 3346              END;
000065 3347          END
000065 3348          ELSE BEGIN
000066 3349              PUTICH (INTREP,ID);
000072 3350          END;
000072 3351          WHILE (EOLN (INTEXT) AND (NOT EOP (INTEXT))) DO BEGIN
000075 3352              READLN (INTEXT);
000077 3353          END;
000100 3354          IF NOT EOP (INTEXT) THEN READ (INTEXT,CHR);
000110 3355      END;
000111 3356
000111 3357      (* SCRATCH BUFFERS *)
000111 3358      FOR I := 1 TO DSSPTR^.LABCTR DO BEGIN
000122 3359          DISPOSE (DSSPTR^.LABARR[I]);
000133 3360      END;
000140 3361      END
000140 3362      ELSE BEGIN
  
```


PASCAL COMPILER - E.T.H. ZURICH / UNIVERSITY OF MINNESOTA.
DISPLAY MODULE PROCEDURES FOR TEXT

PASCAL 6000 V3.0.0. 80/11/17. 00.56.01.
NOS 1.4 (90/04/21) PAGE 69

000141 3163
000143 3164
000143 3165
000157 3166
ERROR := NOSUCHBLOCK;
END;
END;

```

000157 3367 (*LL*PROCEDURES FOR TOUCH TARGETS**)
000157 3368 (*-----*)
000157 3369 =
000157 3370 =
000157 3371 =          TTTTTT    00000    UU    UU    CCCCC    HH    HH
000157 3372 =          TT      00    00    UU    UU    CC    CC    HH    HH
000157 3373 =          TT      00    00    UU    UU    CC          HH    HH
000157 3374 =          TT      00    00    UU    UU    CC          RHHHHHHH
000157 3375 =          TT      00    00    UU    UU    CC          HH    HH
000157 3376 =          TT      00    00    UU    UU    CC    CC    HH    HH
000157 3377 =          TT      00000    UUUUU    CCCCC    HH    HH
000157 3378 =
000157 3379 =
000157 3380 =
000157 3381 =
000157 3382 = TTTTTT    AAAAA    RRRRRR    GGGGG    EEEEE    TTTTTT
000157 3383 = TT      AA    AA    RR    RR    GG    GG    EE      TT
000157 3384 = TT      AA    AA    RR    RR    GG          EE      TT
000157 3385 = TT      AAAAAA    RRRRRR    GG          EEEEE    TT
000157 3386 = TT      AA    AA    RRRR    GG    GGG    EE      TT
000157 3387 = TT      AA    AA    RR    RR    GG    GG    EE      TT
000157 3388 = TT      AA    AA    RR    RR    GGGGG    EEEEEEE    TT
000157 3389 =
000157 3390 =
000157 3391 (*-----*)
000157 3392
000157 3393
000157 3394
000157 3395 (*=====*)
000157 3396 PROCEDURE GETTOUCHINP (*
000010 3397     VAR X,Y : INTEGER;*) (* X,Y COORDINATES OF TOUCH *)
000010 3398     (*VAR CBUF : INBUF;*) (* ARRAY OF INPUT CHARACTERS *)
000010 3399     (*VAR LEN : INTEGER;*) (* LENGTH OF STRING IN CHARRAY *)
000010 3400     (*MKCIRCLE : BOOLEAN;*) (* MAKE A CIRCLE
000010 3401     WHERE USER TOUCHED *)
000010 3402     (*VAR ERROR : ERRORTYPE*) (* BADTOUCH *) (* ); *) ;
000010 3403
000010 3404
000010 3405 (* USER DESCRIPTION:
000010 3406 THIS PROCEDURE IS PROVIDED FOR ACCEPTING TOUCH INPUT. WHEN THE
000010 3407 SCREEN IS TOUCHED, A SMALL CIRCLE WILL BE PLACED AT THE TOUCH POINT,
000010 3408 IF THE FLAG, MKCIRCLE, IS SET TO TRUE.
000010 3409 IF THE INPUT WAS GARBAGE, ERROR IS SET TO BADTOUCH
000010 3410 *)
000010 3411
000010 3412 (* INTERNAL DESCRIPTION:
000010 3413 FIRST GET TERMINAL IN TOUCH MODE THEN DECODE THE INPUT.
000010 3414 THE INPUT FROM THE TOUCH PANEL IS IN THE FORM OF A 4
000010 3415 CHARACTER SEQUENCE WHERE:
000010 3416     CHARACTER 1 : /0000J10/ STX CHARACTER
000010 3417     CHARACTER 2 : /10/X2X3X4/10/
000010 3418     CHARACTER 3 : /01/Y1Y2Y3Y4X1/
000010 3419     CHARACTER 4 : /0001101/ CR CHARACTER
000010 3420
000010 3421 THE PURPOSE OF THE ORION IN SENDING THE STX CHARACTER
000010 3422 IS THAT IF IT IS ECHOED BACK TO THE ORION, IT
000010 3423 CAUSES THE ORION'S KEYBOARD TO BE DISABLED.

```

```

000010 3424 THE CR ALLOWS THE CYBER TO KNOW THAT THIS IS THE END
000010 3425 OF THE TOUCH INPUT SEQUENCE. NOTICE THAT THE BITS ARE
000010 3426 BACKWARDS IN THE CHARACTERS SENT. (MSB IS LOWER ORDER)
000010 3427 THIS IS WHY THERE IS A BIT REVERSAL LOOKUP TABLE USED
000010 3428 IN THIS PROCEDURE. THIS ROUTINE IGNORES ANY RESPONSE
000010 3429 THAT CONSIST OF JUST A CR. I DON'T BELIEVE THAT I REALLY
000010 3430 NEED TO SEND OUT THE COLON & CONTROL BYTP. (THE FUNCTIONALITY
000010 3431 OF WHICH CAN BE FOUND IN APPENDIX C OF THE NOS TIME
000010 3432 SHARING MANUAL. THIS PROCEDURE DOES NOT WORK PERFECTLY.
000010 3433 IF THE PROGRAM SEEMS TO MISS THE TOUCH INPUT YOU SHOULD
000010 3434 HIT THE CR KEY AND REENTER YOUR TOUCH RESPONSE. THE REASON
000010 3435 FOR FAULTY OPERATION PROBABLY IS DUE TO THE FACT THAT
000010 3436 THERE IS NO TYPE-AHEAD IN THE NOS SYSTEM.
000010 3437 *)
000010 3438
000010 3439 LABEL 1,2:
000010 3440 VAR
000010 3441   IND : INTEGER;
000011 3442   DONE : BOOLEAN;
000012 3443   N1 , N2 , N3 : INTEGER;
000015 3444
000015 3445 BEGIN
000015 3446   SETCOCRD(10,10,DOTS);
000010 3447   MODF(DSPTEXT, TOUCH, DSPNORM, DSPNO);
000015 3448   REPEAT
000015 3449     BUTCH(OUTPUT, TCH); (*GET IT IN TOUCH MODE*)
000017 3450     WRITELN:
000021 3451     WRITELN(' : '); (*SET IN ASCII MODE*)
000027 3452     GETSEG(INPUT);
000031 3453     GET(INPUT);
000035 3454     UNTIL NOT EOS(INPUT);
000037 3455     GETLN(INPUT, CBUF, LEN, DONE);
000042 3456     IND := 1; (* ERROR *)
000044 3457     IF IND > LEN THEN GOTO 1;
000046 3458     N1 := CNVTA(CBUF, IND);
000052 3459     IND := IND + 1;
000054 3460     IF (IND > LEN) OR (N1 <> STX) THEN GOTO 1; (* ERROR *)
000060 3461     N2 := CNVTA(CBUF, IND);
000064 3462     IND := IND + 1;
000066 3463     IF IND > LEN THEN GOTO 1; (* ERROR *)
000071 3464     N3 := CNVTA(CBUF, IND);
000075 3465     IF IND <> LFN THEN GOTO 1; (* ERROR *)
000100 3466     X := ((N2 DIV 4) MOD 8);
000104 3467     X := X + (N3 MOD 2) * 8;
000107 3468     Y := ((N3 DIV 2) MOD 16);
000112 3469     X := DSSPTR^.REV[X]*32 + 15; (*REVERSE THE BITS*)
000122 3470     Y := ESSPTR^.REV[Y]*32 + 15;
000132 3471     IF MCIRCLE THEN DRAWCHAR(111,X,Y,DOTS,CENTER,NORMAL,DSPNORM,DSPNO);
000144 3472     ERROR := NOERROR;
000146 3473     GOTO 2; (* ERROR ISLAND CREATION *)
000147 3474 1:   ERROR := BADTOUCH;
000151 3475 2: END;
000200 3476
000200 3477
000200 3478 (*=====*)
000200 3479 PROCEDURE GETTARGINP (
000002 3480   VAR ID : INTEGER; (* BLOCK ID *)

```

```

000003 3481 VAR TARGID : INTEGER; (* TARGET ID *)
000004 3482 CHARRAY : INPCH; (* ARRAY OF INPUT CHARACTERS *)
000005 3483 CHLEN : INTEGER; (* LENGTH OF STRING IN CHARRAY *)
000006 3484 INCLUDEBLOCKS : BOOLEAN; (* INCLUDE BLOCKS AS TARGETS
000007 3485 HAS A TARGET ID OF 0 *)
000008 3486 ERROR : ERRORTYPE (* MISTOUCH, BADTOUCH *) );
000009 3487
000010 3488 (* USER DESCRIPTION:
000011 3489 READS TOUCH TARGET INPUT. EACH TARGET ON THE SCREEN IS KNOWN BY A
000012 3490 (ID,TARGID) PAIR. IF A TOUCH WAS IN AN INSENSITIVE AREA, THEN ERROR
000013 3491 IS SET TO MISTOUCH. IF THE TOUCH WAS NOT DECODEABLE AS BEING A TOUCH,
000014 3492 THEN ERROR IS SET TO BADTOUCH (I.E. USER PROBABLY USED THE KEYBOARD).
000015 3493 FOR THIS CASE, THE LINE OF INPUT TEXT IS PASSED BACK TO YOU. YOU CAN
000016 3494 THEN USE THIS ARRAY TO FETCH ANY KEYBOARD INPUT.
000017 3495 *)
000018 3496
000019 3497 (* INTERNAL DESCRIPTION:
000020 3498 CALL GETTOUCHINP, CHECK TO SEE IF WE ARE IN AN
000021 3499 ACTIVE TARGET, DO VISUAL RESPONSE TO TOUCH ACTIONS.
000022 3500 IF INCLUDEBLOCKS IS TRUE, THEN ANY TOUCH BEHIND AN INCL
000023 3501 INCLUDED IN A BLOCK TOUCHED WILL COME BACK A
000024 3502 BLOCK TOUCHED AS THE BLOCKID AND A TARGID OF 0.
000025 3503 *)
000026 3504
000027 3505
000028 3506
000029 3507 VAR
000030 3508 ID1 : INTEGER;
000031 3509 X1,Y1 : INTEGER;
000032 3510 I : INTEGER;
000033 3511 PTR : TPTR;
000034 3512 X,Y : INTEGER;
000035 3513
000036 3514 BEGIN
000037 3515 GETTOUCHINP(X,Y,CHARRAY,CHLEN,FALSE,ERROR);
000038 3516 IF ERROR = NOERROR THEN BEGIN
000039 3517
000040 3518 (* CONVERT TO TARG COORDS *)
000041 3519 X1 := X DIV 32;
000042 3520 Y1 := Y DIV 32;
000043 3521
000044 3522 (* DO HOUSEKEEPING *)
000045 3523 PTR := DSSPTR^.TARGARRAY[X1,Y1].TPTR;
000046 3524 IF PTR <> NIL THEN BEGIN (* COINCIDES WITH A REAL TARG *)
000047 3525 IF (NOT PTR^.TOUCHED) AND (NOT DISABLED(ID,ID1)) THEN BEGIN
000048 3526 ID := DSSPTR^.TARGARRAY[X1,Y1].BLOCKID;
000049 3527 TARGID := PTR^.ID;
000050 3528 PTR^.TOUCHED := TRUE;
000051 3529
000052 3530 (* DO VISUAL ACTIONS *)
000053 3531 CASE PTR^.STYLE OF
000054 3532 UNDERLINE : BEGIN
000055 3533 DRAWBOX (PTR^.X,PTR^.XLEN,PTR^.Y,PTR^.YLEN,DOTS,DOTS,LL,-2,ERASE,DSPNO);
000056 3534 SETCOORD (PTR^.LONGX,PTR^.LONGY,DOTS);
000057 3535 MODE (DSPTEXT,DSPNCNF,OVER,DSPNO); (*SET IN MODE TO UNDERLINE *)
000058 3536 FOR I := 1 TO CNTCHARS (PTR^.LABL,PTR^.LBLEN) DO BEGIN
000059 3537 WRITE (OUTPUT,' ');

```

```

000204 3538           END;
000211 3539           END;
000212 3540
000212 3541           PAT : BEGIN
000212 3542             DRAWBOX (PTR^.X,PTR^.YLEN,PTR^.Y,PTR^.YLEN,DOTS,DOTS,LL,-5,DSPNORM,DSPNO);
000244 3543           END;
000245 3544           END; (* OF CASE *)
000251 3545           END
000251 3546           ELSE BEGIN
000252 3547             ERROR := MISTOUCH;
000254 3548           END;
000254 3549           END
000254 3550           ELSE BEGIN
000255 3551             IF INCLUDEBLOCKS AND (NOT DISABLED(ID,ID1)) THEN BEGIN
000263 3552               ID := DSSPTR^.TARGARRAY[X1,Y1].BLOCKID;
000275 3553               TARGID := 0; (* INDICATES WHOLE BLOCK *)
000277 3554             END
000277 3555             ELSE BEGIN
000300 3556               ERROR := MISTOUCH;
000302 3557             END;
000302 3558           END;
000302 3559           END;
000302 3560           END; (* OF GETTARGINP *)
000334 3561
000334 3562
000334 3563
000334 3564 (*----- M A I N -----*)
000334 3565 BEGIN
000334 3566   (*$I'GBLINIT'/'KBLIB'*)
  
```

----- BEGIN INCLUDED TEXT.

```

000334 3566 (* INITIALIZATION OF GLOBAL VARIABLES *)
000334 3566
000334 3566   TMSPTR := NIL;
000040 3566   PRSPTR := NIL;
000041 3566   RESPTR := NIL;
000041 3566   DSSPTR := NIL;
000042 3566   S1SPTR := NIL;
000043 3566   S2SPTR := NIL;
000043 3566   GBLERROR := NOERROR;
  
```

----- END INCLUDED TEXT.

```

000044 3567   INITDSPARRAYS('CHTEXT ');
000046 3568   RESET(MSG);
000050 3569   CREATFBLOCK(DSSPTR^.I,0,512,0,512,DOTS,DOTS,LL,5,NOScroll,WRAP,STANDARD,ALTERNATE,GBLERROR);
000075 3570   DSTEXT(1,MSG,GBLERROR);
000100 3571   WITH DSSPTR DO BEGIN
000105 3572     WHILE TRUE DO BEGIN
000107 3573       GETTARGINP(I,J,CBUF,K,FALSE,GBLERROR);
000121 3574     END;
000122 3575   END;
000122 3576 END.
  
```