SHARE PROGRAM LIBRARY AGENCY

PROGRAM NUMBER

234005

University of Miami

1365 MEMORIAL DRIVE - CORAL GABLES, FLORIDA
(305) - 284-6257

Triangle Universities Computation Center
Post Office Box 12076
Research Triangle Park, North Carolina
27709 USA

SPLA CONTROL NUMBER:

This form should be completed and submitted with the program package to the SHARE Program Library Agency at the address shown above. Standards and instructions for submitting programs are in the "SHARE Program Library Standards Manual".

(1) Program Number (to be filled in by SPLA) 370D-23.4.005

(2) System Type (machine) 370-168

(3) Search Key .....

(4) Programming Language ..... Fortran IV and Assembly Language

(5) Author's Name and Address ..... Robert J. Haugen
Rohr Industries, Inc.
Chula Vista, California 92012

(6) Direct Inquiries to Name and Address ..... J. A. Cooper
(if different than Author)
Rohr Industries, Inc.
P. O. Box 878
Chula Vista, California 92012

(7) Title of Program ..... 370 APT-AC (PTF3)
APTLFT Implementation

(8) Submitter's Installation Membership Code..... RHR

(9) Submitter's Own Program Identification and Suffix(Optional)... APT/NC

(10) Primary Subject Code..... 23 4

(11) Operating or Monitor System Required OS

(12) New or Revision Code (if revision, show prior Program Number in Item 1).. NEW

(13) Year Completed..... 1974

(14) Date of Submittal..... Oct. 28, 1974

(15) Documentation (number of original pages submitted)..... 75

(16) Abstract (should contain sufficient information for a reader to determine the value of the program). Listed on the reverse side of this form are subjects which may serve as a guide for a descriptive abstract.

SHARE PROGRAM LIBRARY SUBMITTAL FORM

Subject Guide:

a. Purpose
b. Programming Language used
c. Version and modification level or release number
d. Field of application
e. Type of routine (main program, subroutine, etc.)
f. Specific description of machine requirements

ABSTRACT

This package provides the necessary updates, new programs, overlays, test programs, and JCL to implement and test system/370 APT-AC (PTF3) with APTLFT (Section 0, 1, 2). All preparation was done on 370-168/3330 (OS-MVT REL 21.7). Enclosed tape is standard label 9 track 1600 BPI VOLSER is APLSHR. Part 3 of documentation reflects JCL to build new load modules for section 0, 1, 2 and to build new diagnostic library. Part 3 also shows JCL to build tape and to copy tape to a 3330

NOTE: PLEASE RETURN TAPES AS SOON AS POSSIBLE. THANK YOU

(Please attach additional pages if necessary).....Total pages attached

Permission to Publish

"I hereby give the SHARE Program Library Agency permission to reprint, reproduce, and distribute this program."

(17) Signature of Submitter and Date Robert J Haugen

(18) Signature of Installation Addressee Jerry A. Cooper

APTLFT DOCUMENTATION

PART 1 - Changes to be made to the source for each IBM sub-routine affected by APTLFT, plus new routines to be added and updates to the Library pool error data set.

PART 2 - User documentation, listing each APTLFT capability and explaining it's use.

PART 3 - Listing of JCL needed to build load modules for sections 0, 1, and 2 using APT-AC source with updates provided on tape.

MAGNETIC TAPE CONTENTS

FILE 1 - Update source - contains updates to be run against APT-AC source with IEBUPDTE, plus complete source for new routines.

DSN=APTLFT.SOURCE.SHARE.PTF3,(RECFM=FB,LRECL=80,BLKSIZE=3520)

UNIT=3330,SPACE=(TRK,(13,1,1))

FILE 2 - Test programs - part programs to test APTLFT capabilities in Sections 1 and 2.

DSN = APTLFT.TESTPRG,DCB=(RECFM=FB,LRECL=80,BLKSIZE=800)

UNIT=3330,SPACE=(TRK,(2,1,1))

MEMBERS:

TEST1 - Program to test section 1 capabilities.
(INPUT DSN=MDIS.TEST370 on FT08)

TEST2 - Program to test section 2 with type 6 MDI's.
(INPUT DSN=F14.LONG163.STP4 on FT08)

TEST3 - Program to test section 2 with type 2, 5, and 6 MDI's.
(INPUT DSN=MDIS.TEST370 on FT08)

FILE 3 - Overlays - provided to build a 348K system for O/S operations system. (Not required for V/S)

DSN=APTACOVY.SHAR,DCB=(RECFM=FB,LRECL=80,BLKSIZE=800)

UNIT=3330,SPACE=(TRK,(3,1,1))

MEMBERS:

SEC1SOVY (For Section 1)
SEC2SOVY (For Section 2)

NOTE: POOLSIZE=(5,5)

FILE 4 - MDI file for test programs.

DSN=MDIS.TEST370,DCB=(RECFM=VB,LRECL=248,BLKSIZE=3972)

UNIT=3330,SPACE=(TRK,(1,1))

FILE 5 - MDI file for test programs

DSN=F14.LONG163.STP4,DCB=(RECFM=VB,LRECL=248,BLKSIZE=3972)

UNIT=3330,SPACE=(TRK,(1,1))

370APTAC(PTF3) APTLFT IMPLEMENTATION

TABLE OF CONTENTS

370APTAC(PTF3) APTLFT IMPLEMENTATION

TABLE OF CONTENTS

PART 1

APTLFT COMPUTER

PROGRAMMER DOCUMENTATION

MEMBER NAME DKUAM000

SUBROUTINE NAME:	ABUFTP
PROGRAM NUMBER:	UAM00000
SECTION REFERENCE:	Section 0
REVISION:	Add FILETAB parameters for FT08F001

MEMBER NAME DKUEA000

SUBROUTINE NAME: DEFPRE

PROGRAM NUMBER: UEA00000

SECTION REFERENCE: Section 1

REVISION:

a) Add definition vocabulary for the following geometric definitions:

```
RPT1 = POINT/XYMOD,INTOF,KN1,LOFT1
RPT2 = POINT/XYMOD,INTOF,CR1,LOFT1
RPT3 = POINT/XYMOD,INTOF,LOFT1,LOFT2
RPT4 = POINT/N,ON,LOFT1
RPT5 = POINT/NOZ,XYMOD,INTOF,LN1,LOFT1
RPT6 = POINT/NOZ,XYMOD,INTOF,CR1,LOFT1
RPT7 = POINT/NOZ,XYMOD,INTOF,LOFT1,LOFT2
CR14A = CIRCLE/TANTO,IOMOD,C1,IOMOD,C2,XYMOD,RADIOS,R
LF5 = LOFT/MDI,PV1,PV2,DIR,TAPE,POS
RPV1 = PNTVCT/X,Y,Z,I,J,K
RPV2 = PNTVCT/P1,V1
RPV3 = PNTVCT/P1,I,J,K
RPV4 = PNTVCT/X,Y,Z,V1
RPV5 = PNTVCT/N,INTOF,PL1,LOFT1
RPV6 = PNTVCT/N,ON,LOFT
RPV7 = PNTVCT/CANON,X,Y,Z,I,J,K
RPV8 = PNTVCT/PV1,CANON - - - -
```

b) Define canoncial lengths of LOFT and PNTVCT appropriate displacement flags and address constants were also added.

MEMBER NAME DKUEC000

SUBROUTINE NAME: REDUEC

PROGRAM NAME: VEC00000

SECTION REFERENCE: Section 1

REVISIONS:

a) Change save area logic from general pool to local

b) Reset flag to 0 in Rohr common (RORSCI) if possible MDI data starting in DEFTAB(1109) is invalid.

MEMBER NAME DKUGD000

SUBROUTINE NAME:	CANPRT
PROGRAM NUMBER:	UGD00000
SECTION REFERENCE:	Section 1
REVISION:	Define geometric indicator 26 as being for point Vectors (PNTVCT)

MEMBER NAME DKUHC000

SUBROUTINE NAME: DRPTS

PROGRAM NUMBER: UHC00000

SECTION REFERENCE: Section 1

REVISION: Recognize the following APT motion commands.

FROM/PNTVCT

FROM/PNTVCT,F

GOTO/PNTVCT

GOTO/PNTVCT,F

MEMBER NAME DKUH7000

SUBROUTINE NAME:	CFMGET
PROGRAM NUMBER:	UH700000
SECTION REF.:	Section 1
REVISION:	Change save area logic from general pool to local.

MEMBER NAME DKUJ3000

SUBROUTINE NAME:	NCDICT
PROGRAM NUMBER:	UJ300000
SECTION REFERENCE:	Section 1
REVISION:	Add logic for PNTVCT definitions

MEMBER NAME DKUIF000

SUBROUTINE NAME:	TRFRM
PROGRAM NUMBER:	UIF00000
SECTION REFERENCE:	Section 1
REVISION:	Add logic to transform PNTVCT in REFSYS

MEMBER NAME DKV1P000

SUBROUTINE NAME:	EXMAC
PROGRAM NUMBER:	V1P00000
SECTION REFERENCE:	Section 1
REVISIONS:	Incorporate CALL to Rohr program APLDEF for APTLFT processing.

MEMBER NAME DKW10000

SUBROUTINE NAME:	SECT1
PROGRAM NUMBER:	W1000000
SECTION REFERENCE:	Section 1
REVISION:	Increase the length of DEFTAB (CDSTAB) array to 3520 double words.

MEMBER NAME XLOFTX

SUBROUTINE NAME:	XLOFTX
PROGRAM NUMBER:	15000000
PURPOSE:	Section 1 program to process LOFT definitions
CALLING SEQUENCE:	CALL XLOFTX (NR) NR contains code number of LOFT definition pattern
ROUTINES CALLED:	None
ERROR PROCEDURES:	None
RESTRICTIONS:	None
DATA USED:	DEFTAB Array DEFANS Array
METHOD:	The pattern definitions code number from DEFPRE is passed in the argument list. A computed GOTO statement branches to the correct area of the program to store the canconical form in the DEFANS array.
REMARKS:	Used in conjunction with APT definitions: LA=LOFT/MDI,XB,YB,XE,YE,DIR,TAPE,POS LB=LOFT/MDI,PNT1,XE,YE,DIR,TAPE,POS LC=LOFT/MDI,XB,YB,PNT2,DIR,TAPE,POS LD=LOFT/MDI,PNT1,PNT2,DIR,TAPE,POS LD2=LOFT/MDI,PTVCT1,PTVCT2,DIR,TAPE,POS LE=LOFT/MDI,NMDI,SKIP,FC,FM,DIR,TAPE,Z

MEMBER NAME REDUK

SUBROUTINE NAME: REDUK

SUBROUTINE NUMBER: 15100000

PURPOSE: Reduces and rewrites MDI data to include only those points between a given start and end point.

CALLING SEQUENCE: CALL REDUK(PT1,PT2,XY,NWORD,TOL,*)

INPUT: PT1 - 2 dimensional array containing X and Y of first point in specified range.

PT2 - 2 dimensional array containing X and Y of last point in specified range.

XY - Array containing MDI surface data

NWORD - number of words in XY array

TOL - allowable tolerance between MDI point and specified point to be the acceptable first or last point of the new MDI array.

* - Fortran error return (RETURN1)

OUTPUT: XY - Reduced MDI data array

XY(1) = number of pieces of data in the new array.

RESTRICTIONS: MDI surface data must be of type 2.

DATA USED: MDI surface data file

METHOD: The input MDI data file (XY) array is searched until a match with the PT1 array, within the specified tolerance (TOL), is found. This is then used as the first point in the MDI to be rewritten. The same procedure is then followed until a match with the PT2 array within tolerance is found. The MDI data files then rewritten to include only the points between these first and last points. The number of pieces of data in the new data file is stored in XY(1).

MEMBER NAME PTVCT2

SUBROUTINE NAME:	PTVCT2
PROGRAM NUMBER:	15200000
PURPOSE:	Computes the intersection of a plane and a LOFT surface.
CALLING SEQUENCE:	CALL PTVCT2
ROUTINE CALLED:	GITTA, VNORM3, VDOT3F, DIST3F, FRMTI4
ERROR PROCEDURES:	1541, 1543, 1545, 1551, 1553, 1562
RESTRICTIONS:	Loft surface must be of type 5 or 6 MDI data.
DATA USED:	DEFTAB array DEFANS array MDI surface data file on external data set RORSC1 common

MEMBER NAME NOZPNT

SUBROUTINE NAME:	NOZPNT
PROGRAM NUMBER:	15300000
PURPOSE:	Determines the point on a LOFT surface that is closest to the intersection point of a LINE, CIRCLE or another LOFT surface
CALLING SEQUENCE:	CALL NOZPNT
ROUTINES CALLED:	LOFTXY, VLNG2F
ERROR PROCEDURES:	None
RESTRICTIONS:	Loft surface must be of type 2 MDI data.
DATA USED:	DEFTAB array DEFANS array
METHOD:	Calls LOFTXY to determine the intersection point of the LOFT surface and the second specified surface (line, circle or second LOFT surface) upon returning routine determines which of the two adjacent points on LOFT surface is closest to the computer point of intersection. The X-Y coordinates and point number are stored into the DEFANS array.
REMARKS:	Used in conjunction with APT definitions: PT1 = POINT/XYMOD,INTOF,LN1,LOFT PT2 = POINT/XYMOD,INTOF,CR1,LOFT PT3 = POINT/XYMOD,INTOF,LOFT2,LOFT

MEMBER NAME LOFTXY

SUBROUTINE NAME:	LOFTXY
PROGRAM NUMBER:	15400000
PURPOSE:	Computes the intersection point of a loft surface with a line circle or another loft surface.
CALLING SEQUENCE:	CALL LOFTXY
ROUTINES CALLED:	GITTA, REDUK, G, LN2PTS, LINCIR, VDOT2F LINLIN
ERROR PROCEDURES:	1541, 1543, 1544, 1545, 1551, 1553, 1554
RESTRICTIONS:	Loft surface must be of type 2 MDI data
DATA USED:	DEFTAB array DEFANS array MDI data external data set RORSC1 common
METHOD:	The previously defined LOFT surface is an array of N points. Each two adjacent point define a straight line segment. An attempt is made to interest each of these segments with the given line or circle or in the case of a second LOFT surface each straight line segment of that LOFT surface. When a point of intersection is found that portion of the LOFT surface is indexed and the point is temporarily saved. Attempted intersection of the remaining line segments continues. If more than two intersecting segments are found an error condition exists. When two solutions exists the modifier is used to select the desired point.
REMARKS:	Used for APT definitions: P1 = POINT/XYMOD,INTOF,sym line,LOFT P2 = POINT/XYMOD,INTOF,sym circle,LOFT P3 = POINT/XYMOD,INTOF,LOFT2,LOFT
GLOSSARY:	TABCAN array equivalenced to DEFTAB(1109) used to store MDI data in core. NROOTS - Number of intersections found. TMPLN - Current line segment on defined LOFT TMPLN2 - Current line segment on 2nd loft

MEMBER NAME ITHPNT

SUBROUTINE NAME: ITHPNT

PROGRAM NUMBER: 15500000

PURPOSE: To define a point or PNTVCT on a surface defined as a LOFT by its indicated positions in array.

CALLING SEQUENCE: CALL ITHPNT

ROUTINES CALLED: GITTA, ZSRXY

ERROR PROCEDURES: 1541, 1543, 1552, 1553

DATA USED:
DEFTAB array
DEFANS array
MDI data on external data set
RORSC1 common

METHOD: A check is made to see if required MDI data is in core if not a call to GITTA is made. Subscript is computed for ITH point. If I=0 last point is obtained. If type 2 MDI data is processed at call to ZSRXY is made to obtain the needed Z value. For a POINT definition the X,Y,Z coordinate values are returned regardless if the MDI data is type 2, 5, or 6.

For PNTVCT definition a surface normal of 0, 0, 1 is returned for types 2 and 5. The surface normal from the MDI data is returned for a type 6.

REMARKS: Used for APT definitions:

P1 = POINT/I,ON,LOFT

P2 = PNTVCT/I,ON,LOFT

GLOSSARY:

ITHPT = number of point desired for definition

IX = computed subscript for point in MDI data.

IPTFLG = type of surface in definition

0 = POINT
1 = PNTVCT

MEMBER NAME GITTA

SUBROUTINE NAME: GITTA

PROGRAM NUMBER: 15600000

PURPOSE: Section 1 program to read a specified MDI surface into core from an external data set.

CALLING SEQUENCE: Call GITTA (INPUT,OUTPUT,NTYPE,*)

INPUT = Specified MDI number or record number

OUTPUT = Array of maximum dimension 2412 8 byte words

OUTPUT(1) will contain number of words (not including this one) returned in output array in left most four bytes and the MDI number in the right most 4 bytes.

OUTPUT(2) X - coordinates of first MDI point.

OUTPUT(3) Y - coordinates of first MDI point.

OUTPUT(NWDS+1) last coordinate of array.

NTYPE = (INPUT) even number specifies record number provided
Odd number specifies MDI number provided.

(OUTPUT) type of MDI read (2, 5 or 6)

(ERROR) error number indicated 1 = difficulty in reading MDI data set. 3 = requested MDI or record number not found on external data set

* = Error Return

ROUTINE CALLED: ATAPRD, ATAPOP, ASEARCH

ERROR PROCEDURES: See NTYPE in Calling Sequence

RESTRICTIONS: MDI external data set's equivalenced to PLOTAP (FT08F001)
ATAPTB is 92 bytes in length.

DATA USED: DEFTAB array equivalenced to Output array in calling program
EQUIVALENCE (DEFTAB(1109),OUTPUT(1))
MDI surface data may not exceed 2411 coordinate values.

METHOD: If record number is provided in calling sequence ASEARCH is employed to locate data.
If MDI number is provided a record by record check is made by calling ATAPRD and checking each MDI number found against requested MDI number.

REMARKS:

Format of MDI on external data set

1st	4 bytes	blank
2nd	4 bytes	RECORD number (sequential order)
3rd	4 bytes	MDI number (may be in any random order)
4th	4 bytes	MDI type. 2 = MDI consists of X, Y, coordinat 5 = MDI consists of X,Y,Z coordinat 6 = MDI consists of X,Y,Z coordinat I,J,K associated surface normal

Next 8 bytes X coordinates of first point.

Next 8 bytes Y coordinates of first point.

MEMBER NAME G

FUNCTION NAME:	G
PROGRAM NUMBER:	15700000
PURPOSE:	Compute a distance from a point to a line or to a circle
CALLING SEQUENCE:	G(X,IS) X is array of dimension (z) X(1)=X-coordinate of point X(z)=Y-coordinate of point. IS is the subscript of the DEFTAB array which points to the definitions code of the second geometry requirements.
ERROR PROCEDURES:	None
RESTRICTIONS:	None
DATA USED:	DEFTAB array and input variable.

MEMBER NAME BLKDT

PROGRAM NUMBER: 15800000

PURPOSE: Blockdata program for RORSC1

GLOSSARY: MDICHK(10)

MDICHK(1) flag to indicate if a valid MDI is in stored in TABCAN array.

0 = invalid data

1 = valid data

MDICHK(2) thru MDICHK(10) not used

MEMBER NAME APLDEF

SUBROUTINE NAME:	APLDEF
PROGRAM NUMBER:	15900000
PURPOSE:	Decodes and format checks a CALL/APTLFT statement and writes the 1000 class and 12000 class PROFIL records.
CALLING SEQUENCE:	CALL APLDEF
ROUTINES CALLED:	PRO1K, DATPUT, ERRTN, REDUC
ERROR PROCEDURES:	Generates error numbers 1187 and 1525
RESTRICTIONS:	In CALL/APTLFT statement Missing parameters must be indicated by commas No trailing commas are permitted Up to 13 parameters may be included
DATA USED:	CSMTAB PROBUF DEFTAB
METHOD:	If CALL/APTLFT statement has no parameters one 1000 class record and one 1200 class subclass record is written on PROFIL. If CALL/APTLFT statement has parameters the CSMTAB (symbol table) and DEFTAB (definitation table) arrays are checked for accuracy of data in statement. If data is found to be correct one 1000 class record and two 1200 class subclass records are written on PROFIL. The first 1200 class record is a dummy record only to second 1200 class record contains all of the part programs input parameters.
GLOSSARY:	I = Next CSNTAB index J = Next DEFTAB index K = Next PROBUF index

MEMBER NAME DKWRC000

SUBROUTINE NAME: A2CTRL

PROGRAM NUMBER: WRC00000

SECTION REFERENCE: Section 2

REVISION: a) Add APTLFT common ATYPE2

ITLPOS = positional parametes for cutter when driving LOFT surface of TYPE2 data.

b) Add CALL to branch to APTLFT processing when a CALL/APTLFT profil record is encountered (RECORD CLASS 12000, SUBCLASS 4 or 5)

MEMBER NAME DKWTH000

SUBROUTINE NAME:	USURF
PROGRAM NUMBER:	WTH00000
SECTION REFERENCE:	Section 2
REVISION:	Delete APTLFT processing call.(Moved to A2CTRL)

MEMBER NAME APTLFT

SUBROUTINE NAME: APTLFT

PROGRAM NUMBER: 27000000

FUNCTION: This routines creates CL. data by driving a cutter on a Loft surface.

CALLING SEQUENCE: CALL APTLFT

ROUTINES CALLED: AERR, CCPR, ZVAL, ASTOP, REDUX, TYPE6, TAPEWT, GITTA2, INSECT, PFREAD

ERROR PROCEDURES: -2192, -2172, -2167, -2155, 2150, 2151, 2153, 2154, 2161, 2163, 2164, 2165, 2166, 2168, 2169

DATA USED: Profil (main source of input to Section 2)

FT08F001 MDI sequential external data set

RESTRICTIONS: NONE

METHOD: A2CTRL calls APTLFT after reading a CALL/APTLFT (12000TYPE, SUBCLASS 4 or 5) record. APTLFT reads the next record from the profil, stores it in the AMPRGM array; and branches to the proper area to process 1000, 2000, 7000, 12000, 3000 (Loft surfaces only), and 5000 (GOTO/POINT or GOTO/PNTVCT-SUBCLASS 5) classes of data. APTLFT continues to process profil records until an illegal class is encountered, then "KEYPRO" is reset to TTRD (profil address) of last 1000 class record and the program returns to A2CTRL.

If record read in is a 3000 type and the 8th 4 byte word in AMPRGM is 16 a GO/LOFT instruction has been encountered GITTA is then called to read MDI from FT08F001 and store it in the "A" array. Cutter CL data is then computed and written on CLTAPE.

REMARKS: If multax is on and MDI is type 6 CLDATA will be identical to MDI data IE: each point will be X, Y, Z, I, J, K. If multax is on and MDI is type 5 TLAXIS will be 0, 0, 1.

GLOSSARY: ATYPE6 Common

MP(I*4)	AMPRGM(R*8)	
(2)		RT, 4 Bytes class of profil record
(4)		RT, 4 Bytes subclass of profil reco
	(3)	Double word parameters in profil record
	THRU	
	(2+NWDS)	

GLOSSARY: MP(I*4) AMPRGM(R*8)

(continued)

	(61)	ACUT (1) inter of corner R, and bottom (x)
	(62)	ACUT (2) corner R
	(63)	ACUT (3) X coordinate of corner R, center (+or-)
	(64)	ACUT (4) Y coordinate of corner R, center (+ or -)
	(65)	ACUT (5) int. of corner R, and side (Y)
(131)		ICUTTER cutter parameter switch = 1 after ACUT array is established
(132)		NWDSS (used in TYPE6) points to code word for next APTLFT parameter
	(67)	PTGO(1) X value clearance point (CALL/APTLFT parameter 9)
	(68)	PTGO(2) Y value clearance point (CALL/APTLFT parameter 8)
	(69)	PTGO(3) Z value of clearance point (CALL/APTLFT parameter 8)
	(70)	NAME(1) data name 'CLEARPT'
	(71)	NAME(2) = 0.0 (subscript of 'CLEARP'
(143)		NWORD is initialized as No.of doubl words in MDI IE A(2) thru A(NWDS+1) NWORD is modified as MDI data is revised
(144)		NWDS is initialized as No.of double words in profil record IE: AMPRGM(3) thru AMPRGM(NWDS+2), also used as no of words being written on CLTAPE
(145)		IPTGO - Initialized = 0 if = 1 clearance point (CALL/APTLFT paramet 8) was encountered
(146)		MDI - MDI number of current MDI in "A" array
(147)		MATSW - initialized = 0 if = non zer matrix enountered in CALL/APTLFT instruction.
(148)		IST - MDI start point No. from CALL/APTLFT parameter #6 if no parameter 6 IST=1
(149)		IND - MDI end point No. from CALL/APTLFT parameter 7
		If no parameter 7 IND=1

GLOSSARY: MP(I*4) AMPRGM(R*8)

(continued)

(150) ISW - flag denoting check point or check plane logic in CALL/APTLFT parameter 6 and 7.

= 0 no check point or check plane

= 1 start point to end point

= 2 start point to check plane.

= 3 check plane to end point

= 4 check plane to end plane

(151) IWRT - initialized = 0. If cutter corner radius (R) with useable heigh (H) will not come tangent to check plane it is set to non zero and a warning is written out.

(152) ISED - = 0 no surface extensions

= 1 surface extension at either end or both ends

= 2 no surface extension but check planes are used at either end or both ends

(76) HIN - not used on this processor.

(155) I5AX - = 0 multax turned off

= 1 multax turned on and if TYPE 5 or 6 MDI data CLPRNT will contain (X,Y,Z,I,J,K) data

(156) IFIL - not used on this processor.

ATYPE2 consist of miscellaneous vaiables for processing TYPE 2 MDIS used in programs APTLFT, CCPR, INSECT, REDUX.

MEMBER NAME CKGOUG

SUBROUTINE NAME: CKGOUG

PROGRAM NUMBER: 27100000

PURPOSE: Check for cutter gouge conditions when check surfaces are being processed in APTLFT

CALLING SEQUENCE: CALL CKGOUG(KSPT,ISED,NWORD,EINSF,MDI)

KSPT - flag to compute point on which gouge exist.

ISED - check plane flag
0 = no CHECK plane
1 = check point
2 = check plane

NWORD - subscript of last if coordinate in A array

EINSF - surface normal flag

- 1 = normal surface
1 = flag to flip surface normal

MDI - MDI number being acted up

ERROR PROCEDURES: APTLFT warnings are printed from program "APTLFT warning ISN ____MDI ____POINT ____cannot use described cutter"

"MAX useable height on cutter is _____ required height is _____".

DATA USED: Cutter parameters from ATYPE6 common MDI file (A array) stored from CANSTO 1589 to end. Max length of MDI file is 2412 pieces of data long.

METHOD: Each point on the MDI is checked for a gouge condition. The e, f, h and r parameters from the cutter description are used to calculate the gouge (gouge = f + r + z component of surface normal). Gouge must be negative or greater than h for the gouge condition to exist.

ROUTINES CALLED: FRMTI4, FRMATD, PRINT

MEMBER NAME SRFEXT

SUBROUTINE NAME: SRFEXT

PROGRAM NUMBER: 27200000

PURPOSE: To compute extension points on one or both ends of a loft surface being processed in APTLFT.

CALLING SEQUENCE: CALL SRFEXT (ISOE,DELTA,ALPHA,NWORD,EINSF)

ISOE - surface extension flag.

- 0 = extension at both ends points with same deltas and alphas
- 1 = extensions at first point only
- 2 = extensions at last point only
- 3 = extensions at both ends with different deltes and alphas

DELTA - array of max length (2) containing the length (ISOE=0,1, or 2) or lengths (ISOE=3) extensions.

ALPHA - array of max length (2) containing the angle (ISOE= 0,1, or 2) or angles (ISOE = 3) of deviation from the tangent. Positive angle is the same direction as the surface normal, negative angle opposite direction from the surface normal

NWORD - subscript of last Y coordinate in loft surface

EINSF - surface normal flag
-1 = normal setting
1 = flip surface normals

DATA USED: Cutter data from ATYPE6 common loft surface stored in A array beginning at CANSTO (1589) max of 2411 pieces of data

METHOD: Using the cutter center line path surface normal and cutter definition, a point on the LOFT surface is computed. Using 3 of those computed points on the LOFT surface the plane of the LOFT surface is computed. The surface extension are then computed by intersecting this plane and the plane tangent to the LOFT surface and moving along and away from this line the specified delta and alphas respectively.

The LOFT surface is restored into the A array with the extension points added.

MEMBER NAME CCPR

SUBROUTINE NAME: CCPR

PROGRAM NUMBER: 27300000

PURPOSE: Calculates the cutter center line relative to the tool position to or on the drive surface.

CALLING SEQUENCE: CALL CCPR

DATA USED: APTLFT common ATYPE2

INPUT:

TA2(1)		cutter position 1 = left 2 = right 3 = on right 4 bytes (equivalenced to ITPOS.)
TA2(2)		cutter diameter (equivalenced to CUTDIA)
TA2(3)	XA	coordinate of first point on drive surface
TA2(4)	YA	
TA2(5)	XB	coordinate of 2nd point on drive surface
TA2(6)	YB	

OUTPUT:

TOUT(1)	X1	coordinates of first point on cutter center line path
TOUT(2)	Y1	
TOUT(3)	X2	coordinates of 2nd point on cutter center line path.
TOUT(4)	Y2	

METHOD: Using the two adjacent input points a surface normal is computed. The cutter center line path is then computed using the cutter radius, input point and surface normal.

REMARKS: Used for Loft surface composed of type 2 MDI data.

MEMBER NAME INSECT

SUBROUTINE NAME: INSECT

PROGRAM NUMBER: 27400000

PURPOSE: To compute the intersection of two lines each being defined by 2 points.

CALLING SEQUENCE: CALL INSECT

DATA USED: APTLFT common ATYPE2

INPUT -

TIN(1) = X1 TIN(2) = Y1	coordinate of first point for line 1
TIN(3) = X2 TIN(4) = Y2	coordinate of second point for line 1
TOUT(1) = X3 TOUT(2) = Y3	coordinate of first point for line 2
TOUT(3) = X4 TOUT(4) = Y4	coordinate of second point for line 2

OUTPUT -

TA11(3)	X coordinate of point of intersection
TA11(4)	Y coordinate of point of intersection

METHOD: Straight lines are computed for each pair of input points and the two lines are then intersected to give desired results.

MEMBER NAME REDUX

SUBROUTINE NAME: REDUX

PROGRAM NUMBER: 27500000

PURPOSE: Reduces and rewrites a LOFT surface to include only those points between a given start and end point.

CALLING SEQUENCE: CALL REDUX (NWORD,TOLCHK,IER)

NWORD - number of words in the A array(LOFT surface)

TOLCHK - allowable tolerance between point on surface and the specified point to be the acceptable first or last point of the new LOFT surface.

IER - return

1 = given point not within tolerance of any point on LOFT surface.

RESTRICTIONS: Loft surface must be of type 2 MDI data

DATA USED: Loft surface stored in A array (CANSTO (1589)to end) Contains not more than 2411 pieces of data

APTLFT common ATYPE2

TA(1) x coordinate
TA(2) y coordinate — of first ppint of specified range.

TA(3) x coordinate
TA(4) y coordinate — of end point of specified range

METHOD: The input LOFT surface is searched until a match with the first point (TA(1),TA(2)), within the specified tolerance (TOLCHK) is found. This point is then used as the first point of the new LOFT surface. The same procedure is followed until a match with the end point (TA(3),TA(4)), within tolerance is found. The LOFT surface is then rewritten starting in A(2) to include only the points between these first and last points. The number of pieces of data in the new surface is stored in A(1).

MEMBER NAME ZVAL

SUBROUTINE NAME: ZVAL

PROGRAM NUMBER: 27600000

PURPOSE: Calculate the Z value using the part surface currently in affect.

CALLING SEQUENCE: CALL ZVAL(X,Y,ZVALUE,IER)

$\begin{matrix} X \\ Y \end{matrix}$ coordinate values of point

ZVALUE computed Z coordinate

IER - return value
0 = Z value calculated
1 = no PSIS

RESTRICTIONS: Part surface must be a plane.

DATA USED: CANSTO array, IIPS in STATI4 common, IPONTR in SURF14 common.

METHOD: Checks to see if part surface is a plane. If so Z coordinate is computed using X and Y coordinates.

MEMBER NAME TOLSEG

SUBROUTINE NAME: TOLSEG

PROGRAM NUMBER: 27700000

PURPOSE: Process cutter definition section of CALL/APTLFT statement

CALLING SEQUENCE: CALL TOLSEG(IR)

IR - return variable.

No zero = error in cutter description.

DATA USED: APTLFT common ATYPE6

METHOD: Makes checks on cutter description to determine validity and acceptability to APTLFT.

REMARKS:

a. Checks intersection of bottom of cutter and side radius must be positive.

b. Difference of radius and Y coordinate of cutter must be positive within epsilon (.0001)

c. Check point of intersection for E and F parameters of cutter description.

d. Usable cutter height must be a positive value.

MEMBER NAME TYPE6

SUBROUTINE NAME: TYPE6

PROGRAM NUMBER: 27800000

PURPOSE: Program is divided into two sections. First section processes the CALL/APTLFT parameters.

Second section generates the cutter centerline path includin processing use of check surfaces and extending the surface.

CALLING SEQUENCE: CALL TYPE6 (JSW)

JSW = 1 process CALL/APTLFT parameters

JSW = 2 or 3 generate cutter center line path

ROUTINES CALLED: TOLSEG, AERR, FRMTI4, FRMATD, PRINT, CKGOUG, SRFCXT

ERROR PROCEDURES: 2173, 2174, 2175, 2176, 2197, 2198, 2199

DATA USED: Section 2 commons STATI4, STATIO, STATR8, SRFSTR

APTLFT common ATYPE6

ISOE - surface extension flag

0 = both ends have extensions points of same length

1 = extensions at first point

2 = extensions at last point

3 = extension at both ends different lengths

XMAT - matix parameters from CALL/APTLFT statement

METHOD: In the first section the CALL/APTLFT parameters are processed. Code words are checked and corresponding data are stored in local arrays.

The second section computes the cutter center line path applies check surfaces and computes any necessary surface extensions.

MEMBER NAME GITTA2

SUBROUTINE NAME: GITTA2

PROGRAM NUMBER: 27900000

PURPOSE: Section 2 program to read a specified MDI surface into core from an external data set

CALLING SEQUENCE: Call GITTA2 (INPUT,OUTPUT,NTYPE,*)

INPUT = specified MDI number or record number

OUTPUT = Array of maximum dimension 2412 8 byte words
- OUTPUT(1) will contain number of words (not including this one) returned in output array in left most four bytes and the MDI number in the right most 4 bytes.
- OUTPUT(2) X - coordinates of first MDI point.
- OUTPUT(3) Y - coordinates of first MDI point.
- OUTPUT(NWDS+1) last coordinates of array.

NTYPE = (INPUT) even number specifies record number provided
Odd number specifies MDI number provided.

(OUTPUT) type of MDI read (2, 5 or 6)

(ERROR) error number indicated 1 = difficulty in reading MDI data set. 3 = requested MDI or record number not found on external data set.

* = Error Return

ROUTINE CALLED: ATAPRD, ATAPOP, ASEARCH

ERROR PROCEDURES: See NTYPE in Calling Sequence.

RESTRICTIONS: MDI external data set's equivalenced to PLOTAP (FT08F001) ATAPTB is 92 bytes in length.

DATA USED: MDI surface data may not exceed 2411 coordinate values.

METHOD: If record number is provided in calling sequence ASEARCH is employed to locate data.
If MDI number is provided a record by record check is made by calling ATAPRD and checking each MDI number found against requested MDI number.

REMARKS: Format of MDI on external data set.

1st 4 bytes blank
2nd 4 bytes RECORD number (sequential order)
3rd 4 bytes MDI number (may be in any random order)
4th 4 bytes MDI type. 2 = MDI consists of X,Y, coordinate
5 = MDI consists of X,Y,Z coordinat
6 = MDI consists of X,Y,Z coordinat I,J,K associated surface normal

Next 8 bytes X coordinates of first point.
Next 8 bytes Y coordinates of first point.

MEMBER NAME DKWZED

PROGRAM NUMBER: WZE00000

PURPOSE: Diagnostic source file updates for APTLFT Section 1 and Section 2.

See user Documentation for detailed explanation.

PART 2

APTLFT USER DOCUMENTATION

R1.01.01 A POINT DEFINED AS THE INTERSECTION OF A LINE AND A LOFT

FORMAT

RPT=POINT/ XLARGE / XSMALL / YLARGE / YSMALL , INTOF,LINE,LOFT

METHOD

The desired point will be determined from the first two intersections found (if two exists). This search is taken in the direction the MDI is defined.

LIMITATIONS

Only applicable on Type 2 MDI's.

R1.01.02 A POINT DEFINED AS THE INTERSECTION OF A CIRCLE AND A LOFT

FORMAT

RPT=POINT/ XLARGE / XSMALL / YLARGE / YSMALL , INTOF,CIRCLE,LOFT

METHOD

The desired point will be determined from the first two intersections found (if two exist). This search is taken in the direction the MDI is defined.

LIMITATIONS

Only applicable on Type 2 MDI's.

R1.01.03 A POINT DEFINED AS THE INTERSECTION OF TWO LOFTS

FORMAT

RPT=POINT/ XLARGE / XSMALL / YLARGE / YSMALL ,INTOF,LOFT1,LOFT2

METHOD

The desired point will be determined from the first two intersections found (if two exist). This search is taken starting with LOFT1 and taken in the direction the MDI's are defined.

LIMITATIONS

Only applicable on Type 2 MDI's.

R1.01.04 A POINT DEFINED AS A DEFINITE POINT ON A LOFT

FORMAT

RPT=POINT/N,ON,LOFT

METHOD

"N" is the number of the desired point on the MDI, if "N" equals 0 (zero) the desired point is the last one.

LIMITATIONS

Applicable with Type 2, 5, or 6 MDI's.

R1.01.05 A POINT BY FINDING THE NEAREST POINT TO THE INTERSECTION OF A LINE AND A LOFT

FORMAT

RPT=POINT/NOZ, XLARGE / XSMALL / YLARGE / YSMALL ,INTOF,LINE,LOFT

METHOD

The desired point will be determined from the first two intersections found (if two exist). The search is taken in the direction the MDI is defined. The actual output is the X and Y values of the MDI point, and the Z value is equal to the number of the MDI point.

LIMITATIONS

Applicable with Type 2, MDI's.

R1.01.06 A POINT BY FINDING THE NEAREST POINT TO THE INTERSECTION OF A CIRCLE AND A LOFT

FORMAT

RPT=POINT/NOZ, XLARGE / XSMALL / YLARGE / YSMALL ,INTOF,CIRCLE,LOFT

METHOD

The desired point will be determined from the first two intersections found (if two exist). The search is taken in the direction the MDI is defined. The actual output is the X and Y values of the MDI point, and the Z value is equal to the number of the MDI point.

LIMITATIONS

Applicable with Type 2.

R1.01.07 POINT BY FINDING THE NEAREST POINT TO THE INTERSECTION OF (2) LOFTS

FORMAT

PT=POINT/NOZ, XLARGE / XSMALL / YLARGE / YSMALL ,INTOF,LFT1,LFT2

METHOD

The desired point will be determined from the first two intersections found (if two exist). The search is taken in the direction the MDI's are defined. The output is the X and Y values of the nearest point on the second loft, and the Z value is equal to the number of the MDI point.

LIMITATIONS

Applicable with Type 2 MDI's.

R1.16 LOFT SURFACE (Section 1)

A LOFT is the APT designation of a MDI which is an input array of points or point vectors which define a plane or space curve. Note that a LOFT is the system's method of surface type definition for an MDI, and consequently, MDI and LOFT are not analogous.

MDI's can be generated by FMILL.

External sequential data sets are used to store Master Dimension records. These MDI files can be created by the user for selected tasks.

TYPES OF MDI'S

Type 2	Two-dimensional, ordered pair (U,V).
Type 5	Three-dimensional, ordered triplet (U,V,W).
Type 6	Three-dimensional, pair of ordered triplets denoting a point (U,V,W) and corresponding normal (I,J,K).

NOTE: Two point minimum requirements for any of the above MDI types.
FMILL generates type 6 MDIS

FORMATS

The following formats can be used as nested named or unnamed loft definitions.

a) RLFT=LOFT/NUMBER,XB,YB,XE,YE,DIR,TAPE,POS
b) RLFT=LOFT/NUMBER,POINT1,POINT2,DIR,TAPE,POS
c) RLFT=LOFT/NUMBER,XB,YB,POINT2,DIR,TAPE,POS
d) RLFT=LOFT/NUMBER,POINT1,XE,YE,DIR,TAPE,POS
e) RLFT=LOFT/NUMBER,NMDI,SKIP,FC,FM,DIR,TAPE,Z
f) RLFT=LOFT/NUMBER,PNTVC1,PNTVC2,DIR,TAPE,POS

NOTE: Format (e) is a special purpose format, which enables sequential MDI's to be called out by one instruction in the part program. This is used in conjunction with an APTLFT Part Programming Documentation)

LOFT PARAMETERS

NUMBER is any legal symbolic name of number representing the MDI number or record number (see TAPE parameter). In the case of format (e), NUMBER is set in the first MDI number or record number (see TAPE parameter), to be processed of the required block. Additionally, if desired, using format (e) and type 6 data, normal vector direction can be reversed by negating the required NUMBER specification.

XB,YB are point coordinate values within the range of the MDI, denoting the loft starting point. Either symbolic or numeric representation is acceptable. (2)

R1.16 (continued)

XE,YE are point coordinate values within the range of the MDI, denoting the loft ending point. Either symbolic or numeric acceptable. (2)

POINT1 is any legal symbolic name of a point whose XB,YB coordinates are the MDI coordinate values of the loft starting point. (2)

POINT2 is any legal symbolic name of a point whose XE,YE coordinates are the MDI coordinate values of the loft ending point.(2)

(2)NOTE: The starting and ending points are points on the MDI in the same direction that the DIR is defined. If the complete MDI is required, the starting point and ending point must both be equal to 0,0 (zero, zero). It should also be noted that with types 5 or 6, that the beginning and ending points will be ignored and the LOFT will be the complete MDI.

DIR may be any legal symbolic name or number 0,1,2,3 and denotes direction between starting point and ending point. If DIR is set to 0 (zero), cutting will be in one direction only, in the order that the curves were defined. If set to 1 (one), the cutting will be in one direction, in the opposite direction that the curves were defined. If set to 2 (two), the cutting will be in alternate directions on MDI's designated, starting in the direction curves were defined. If set to 3 (three), the cutting will be in alternate directions on MDI's designated, starting in the opposite direction curves were defined.

TAPE may be any legal symbolic name or numeric, and denotes the following information. (*)

If TAPE is set to 0 (zero), part of one MDI is to be processed where NUMBER represents the MDI RECORD NUMBER. (3)

Set to 1 (one) if part of one MDI is to be processed where NUMBER represents the MDI NUMBER. (3)

(3)NOTE: Only applicable if MDI is type 2, not using format (e).

Set to 2 (two) if one complete MDI is to be processed, where NUMBER specifies the MDI RECORD NUMBER. (4)

Set to 3 (three) if one complete MDI is to be processed, where NUMBER is the MDI NUMBER.(4)

(4)NOTE: Not applicable if using format (e).

R1.16 (con't)

Set to 4 (four) if using format (e), and where NUMBER represents MDI RECORD NUMBER.

Set to 5 (five) if using format (e), and where NUMBER represents MDI NUMBER.

(*)NOTE: If the TAPE parameter is an EVEN specification, selection of MDI's will be by MDI RECORD NUMBER. If the TAPE parameteris an ODD specification, selection of MDI's will be by MDI NUMBER.

POS is any legal symbolic name or numeric and is set to 0 (zero) normally. POS value may be set to 1 (one), if cutter motion is desired to be tangent (inclusive) to the MDI and tangent to lines normal to curve through starting point and ending point. Only applicable if using type 2 MDI's.

NMDI is any legal symbolic name or numeric and is set to the number of MDI's that are to be processed when using format (e).

SKIP is any legal symbolic name or number representing the MDI SKIP value when using format (e). NOTE: The SKIP value may be positive or negative.

EXAMPLES

=1	Every MDI from MDINO sequentially up through (NUMBER+NMDI-1)
=N	Every Nth MDI from MDINO sequentially up through (N(NMDI-1)+NUMBER)
=-1	Every MDI down through (NUMBER/-NMDI+1)
=-N	Every Nth down through (N(NMDI-1)-NUMBER)

FC is any legal symbolic name or number representing the cutter feedrate for clear of the cutting surface when using format (e).

FM is any legal symbolci name or number representing the cutter feedrate for milling when using format (e).

Z is any legal symbolic name or number representing the Delta motion in Z direction above beginning and ending points when using format (e). NOTE: Z must be defined even = 0 (zero).

ERRORS

If an error is encountered, an appropriate error comment is printed.

R1.26.01 PNTVCT DEFINED BY A POINT AND VECTOR

FORMAT

A) PVC = PNTVCT/X,Y,Z,I,J,K

B) PVC = PNTVCT/CANON,X,Y,Z,I,J,K

C) PVC = PNTVCT/POINT,VECTOR

D) PVC = PNTVCT/POINT,I,J,K

E) PVC = PNTVCT/X,Y,Z,VECTOR

METHOD

X, Y, Z are coordinates of a point, and I, J, K are direction numbers of a vector.

NOTE: Direction numbers will not be normalized by the system.

R1.26.02 PNTVCT BY INTERSECTION OF A SYMBOLIC PLANE AND A LOFT

FORMAT

RPVCT=PNTVCT/N,INTOF,PLANE,LOFT

METHOD

N equals the number of intersection, i.e., if N=1 the first intersection found will be used; if N=2 the second intersection found will be used, etc. The search is taken in the direction the MDI is defined.

PLANE

Must be symbolic (subscript OK) name of a previously defined plane.

LOFT

Must be symbolic (subscript OK) name of a previously defined loft.

LIMITATIONS

Only applicable to Type 5 and Type 6 MDI's.
If Type 5 (X,Y,Z) vector will be 0,0,1.
If Type 6 (X,Y,Z,I,J,K) vector will be normal to MDI

R1.26.03 PNTVCT BY A DEFINITE POINT ON A LOFT

FORMAT

RPTVCT=PNTVCT/I,ON,LOFT

METHOD

I equals the number of the point on the LOFT
If I equals 0 (zero) the last point defined on the LOFT will be used.

LOFT

Must be symbolic (subscript OK) name of a previously defined LOFT.

LIMITATIONS

If Type 2 (X,Y) Z will be taken from last ZSURF and vector will be 0,0,1.
If Type 5 (X,Y,Z) vector will be 0,0,1.
If Type 6 (X,Y,Z,I,J,K) vector will be vector associated with MDI.

R 2.1 APTLFT (Section 2 processing cutter motion on MDI's*)

* NOTE: An MDI is a series of continuous points or points and surface normal vectors defining a space curve (see R 1.16).

The APTLFT program is entered with a CALL/APTLFT instruction. Once APTLFT is entered, it will process motion instructions for LOFT (MDI surfaces, GOTO/POINT, and POSTPROCESSOR commands. Any other motion instruction will cause an exit from APTLFT.**)

Reentry to APTLFT can be made by repeating a CALL/APTLFT instruction. ***

** NOTE: Placement of symbolic definitions in a CALL/APTLFT sequence will not terminate APTLFT processing.

*** NOTE: There is no apparent limit to number of CALL/APTLFT sequences.

CALL/APTLFT Format

	1	2	3	4	5	6	7	8	9	10	11	12	13
CALL/APTLFT,	M,	R,	E,	F,	H,	I,	J,	P,	K,	D1,	A1,	D2,	A2
CALL/APTLFT,	M,	R,	E,	F,	H,	PL1,	PL2,	P,	K,	D1,	A1,	D2,	A2
CALL/APTLFT,	M,	R,	E,	F,	H,	I,	PL2,	P,	K,	D1,	A1,	D2,	A2
CALL/APTLFT,	M,	R,	E,	F,	H,	PL1,	J,	P,	K,	D1,	A1,	D2,	A2

Missing parameters in the CALL/APTLFT statement must be indicated by commas when there are following parameters. Trailing commas must be omitted.

EXAMPLES:

```
CALL/APTLFT
CALL/APTLFT,,,,,,4,1,P1
CALL/APTLFT,,,,,,PL1,1,,0.,5,0
```

NOTE:

a) Nesting is allowed in a CALL/APTLFT instruction.
b) Scalars may be numbers or symbolic (subscript O.K.).
c) Normal CL output is in 3-axis mode.
d) If multax is turned on tool end point and tool axis will be same as MDI coordinates if MDI is Type 6. Tool axis will be 0,0,1 if MDI is Type 5.

R 2.1
(con't)

APTLFT MATRIX Option

M is parameter #1 and it is a symbolic name of a previously defined MATRIX. If this parameter is used, the MDI data will be transformed through this MATRIX before ℄ CUTTER is calculated.

EXAMPLE:

If input MATRIX is defined a1,b1,c1,d1,a2,b2,c2,d2, a3,b3,c3,d3

	X	Y	Z	
XS	a1	b1	c1	d1
YS	a2	b2	c2	d2
ZS	a3	b3	c3	d3

NOTE: If the MDINO in a loft definition is negative, the surface normals are flipped (see R1.16).

EXAMPLE:
If A,Y,Z,I,J,K are coordinates of an MDI point, then the new point XS,YS,ZS,IS,JS,KS is:

$XS=a_1*X+b1*Y+c1*Z+d1$

$YS=a_2*X+b2*Y+c2*Z=d2$

$ZS=a_3*X+b3*Y+c3*Z+d3$

$IS=a_1*I+b1*J+c1*K$

$JS=a_2*I+b2*J+c2*K$

$KS=a_3*I+b3*J+c3*K$

APTLFT CUTTER Option

R,E,F,H are for APTLFT cutter definition (parameters 2,3,4,5)

R is cutter corner (or side) radius
E is the X of the center of R
F is the Z of the center of R
H is the usable height of the cutter

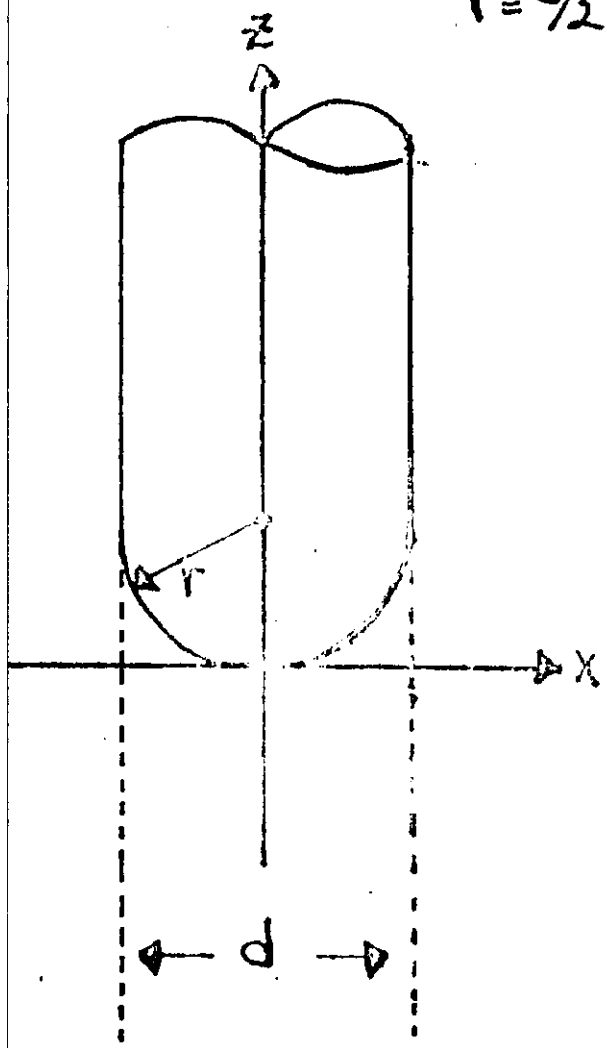
NOTE: a) The APT cutter currently in effect will be used if R,E,F,H are not specified.

b) If APTLFT cutter parameters are used, all four numbers must be defined.

c) The APTLFT cutter does not alter the APT cutter definition currently in effect.

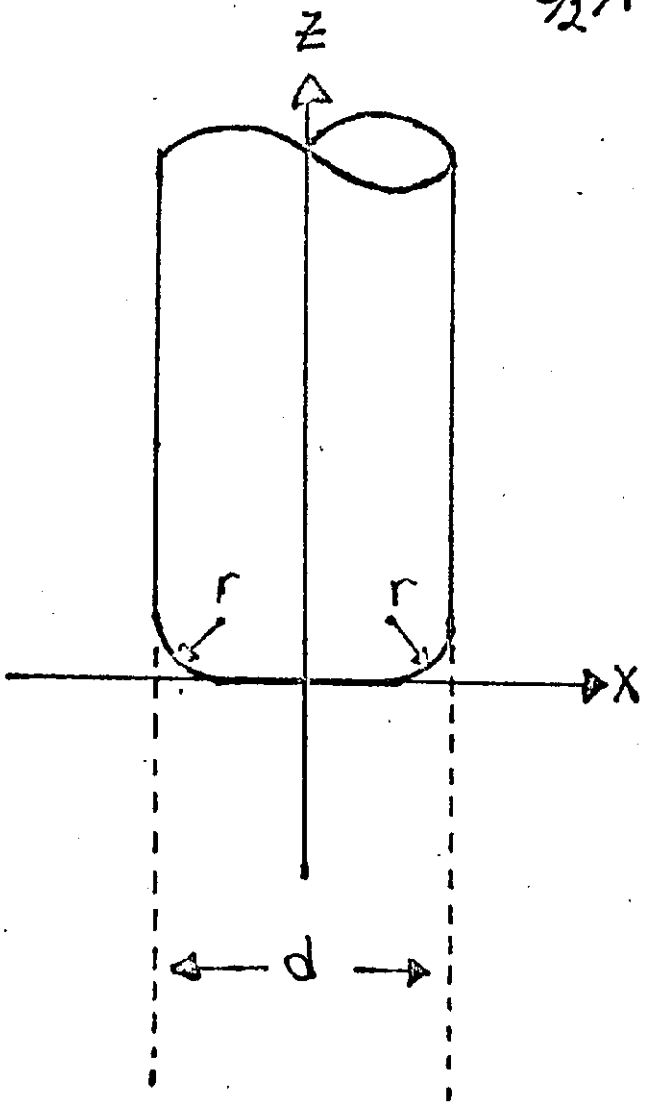
CUTTER/d,r

r = d/2

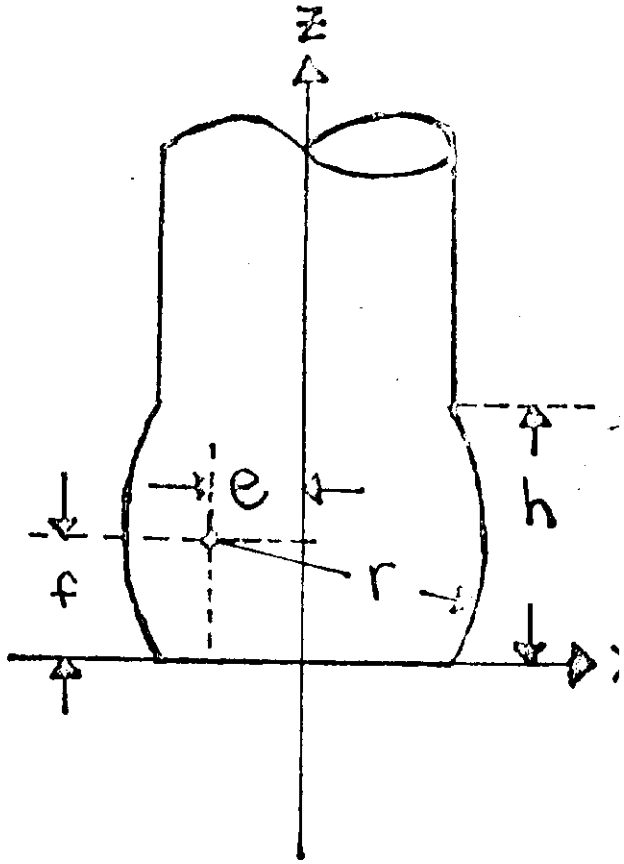


CUTTER/d,r

d/2 > r



CALL/APTLFT,,r,e,f,



R2.1
(con't)

APTLFT BOUNDARY CONTROL

Parameters 6 and 7 are for boundary control and define the portion of the MDI's that is to be used for cutter motion. If this option is used, both parameters must be specified.

THESE PARMETERS HAVE NOTHING TO DE WITH CUTTER DIRECTION.

a) PARAMETER 6 (either I or PL1):

I - Specifies the sequence number of an MDI point (relative to the beginning of MDI) and is used for boundary control.

PL1 - Is a symbolic name of previously defined plane. It is the beginning check surface and is tangent to ball of cutter. This plane is not affected by the APTLFT MATRIX.

b) PARAMETER 7 (either J or PL2):

J - Specifies the sequence number of an MDI point (relative to the end of MDI) and is used for boundary control.

PL2 - Is a symbolic name of a previously defined plane. It is the ending check surface and is tangent to ball of cutter. This plane is not affected by the APTLFT MATRIX.

CLEARANCE POINT

a) PARAMETER 8

P - Is a symbolic name of a previously defined point. This point is not affected by the APTLFT MATRIX. The cutter will travel to this point before processing each MDI if using multiple MDI format.

APTLFT SURFACE EXTENSION CAPABILITY:

Parameters 9 through 13 are used to extend the MDI's at either end or at both ends. THESE PARMETERS HAVE NOTHING TO DO WITH CUTTER DIRECTION.

If this capability is used, parameters (9,10,11); or parameters (9,10,11,12,13) must all be used.

a) PARAMETER 9, 10, 11:

a) K = 0 Extension at both ends using D1 and A1 at both ends.

= 1 Extension at the beginning only, using D1 and A1

= 2 Extension at the end only using D1 and A1.

R 2.1
(con't)

APTLFT SURFACE EXTENSION CAPABILITY:

a) PARAMETER 9, 10, 11:

2) D1 Is the length of the extension in the plane of the MDI

3) A1 Is the angle of the surface extension from the tangent plane of the MDI (can be zero); plus toward the surface normal, minus away from surface normal.

b) PARAMETER 9, 10, 11, 12, 13

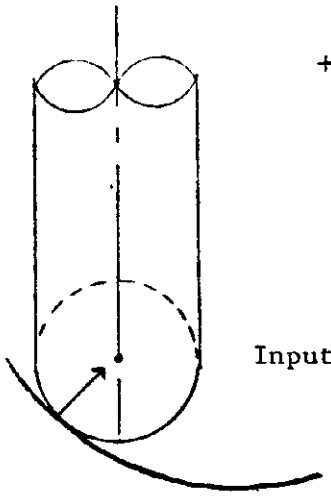
If K=3 (parameter 9) then D1 and A1 (parameter 10 and 11) are used at the beginning and D2 and A2 (parameter 12 and 13) are used at the end.

If check surface planes and/or beginning and ending points are used, they are processed before the surface extensions.

R 2.1.2 Type 6 MDI (or TDI) and APT Cutter Relationship

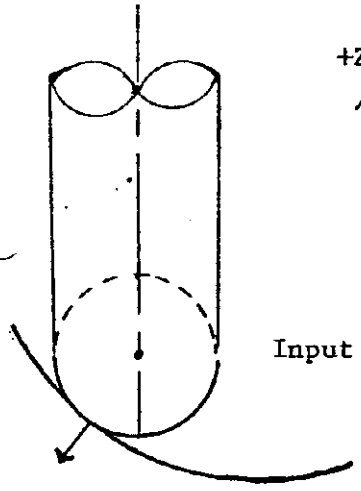
Following are 8 examples of making a shown or opposite part (either sex) from MDIS (type 6). These examples should help clarify the use of a negative "MDINO"

Type 6 MDI and APT Cutter Relationship



Normal application
SHN. Part (same sex)

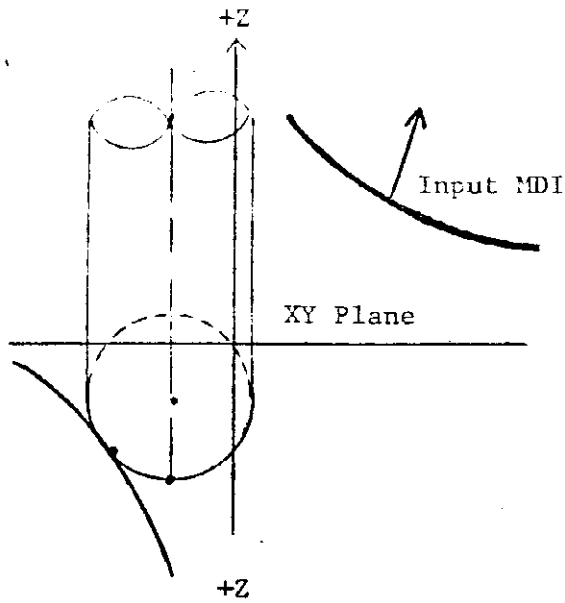
```
MDI vector is +Z
MDINO = N
RLFT1 = LOFT/MDINO, etc.
CALL/APTLFT
GO/RLFT1
```



Abnormal application
SHN. Part (same sex)

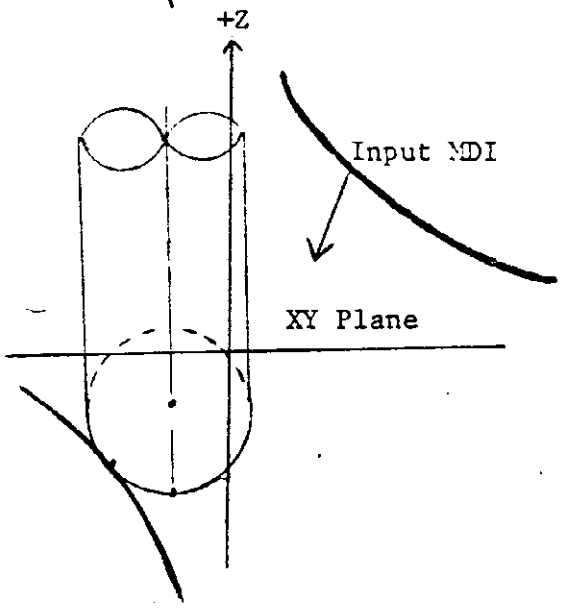
```
MDI vector is -Z
EITHER ACCEPTABLE { M1 = Matrix/YZROT, 180
                  { M1 = Matrix/ZXROT, 180
                  { M1 = Matrix/MIRROR, XY Plan

MDINO = -N
RLFT2 = LOFT/MDINO, etc.
TRACUT/M1
CALL/APTLFT, M1
GO/RLFT2
TRACUT/NOMORE
```

Type 6 MDI and APT Cutter Relationship



Noraml application
SRF. Part (other sex)

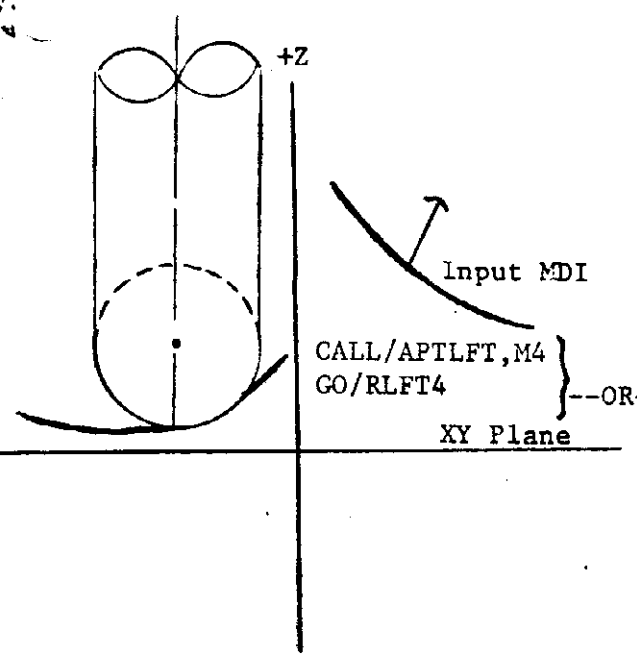
```
   MDI vector is +Z
M2 = Matrix/YZROT, 180 or
M2 = Matrix/ZXROT, 180

   MDINO = -N
   RLFT3 = LOFT/MDINO, etc.
   TRACUT/M2
   CALL/APTLFT
   GO/RLFT3
   TRACUT/NOMORE
```



Abnormal application
SRF. Part (other sex)

```
   MDI vector is -Z
M3 = Matrix/YZROT, 180 or
M3 = Matrix/ZXROT, 180

   MDINO = N
   RLFT4 = LOFT/MDINO, etc.
   CALL/APTLFT, M3
   GO/RLFT4
```

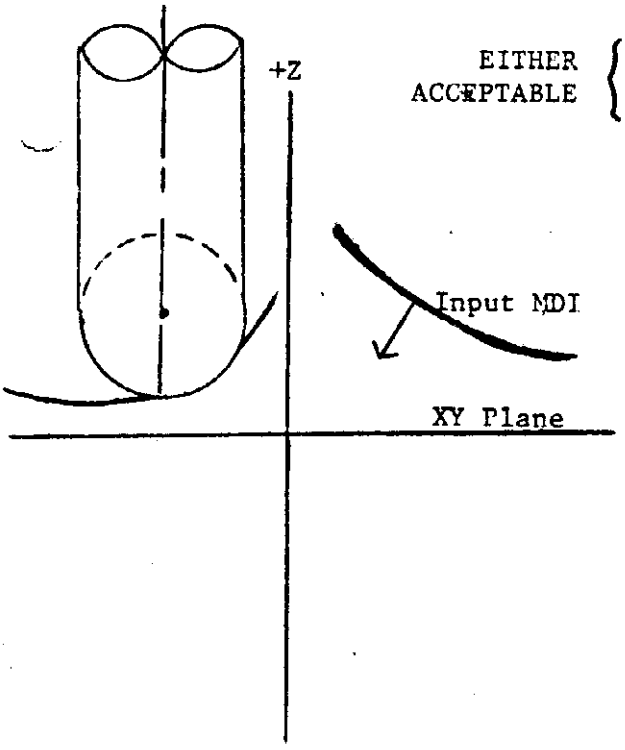
Type 6 MDI and APT Cutter Relationship



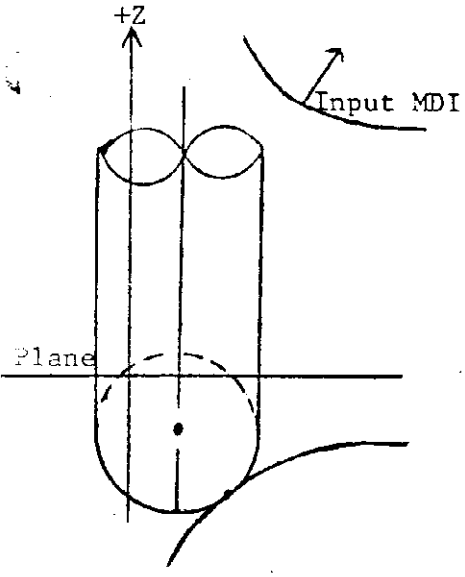
Normal application
OPP. Part (same sex)

```
   MDI vector is +Z
M4 = Matrix/MIRROR, ZX Plan or
M4 = Matrix/MIRROR, YZ Plan

   MDINO = N
   RLFTS = LOFT/MDINO, etc.
   TRACUT/M4
   CALL/APTLFT
   GO/RLFT5
   TRACUT/NOMORE
```

Abnormal application
OPP. Part (same sex)



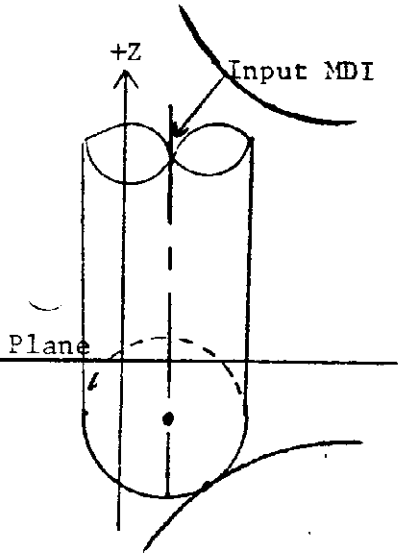

```
   MDI vector is -Z
EITHER
ACCEPTABLE
M5 = Matrix/MIRROR, XY Plan
M5 = Matrix/YZROT, 180
M5 = Matrix/ZXROT, 180
M6 = Matrix/MIRROR, ZX Plan or
M6 = Matrix/MIRROR, YZ Plan
M7 = Matrix/M5, M6

   MDINO = -N
   RLFT6 = LOFT/MDINO, etc.
   TRACUT/M7
   CALL/APTLFT, M5
   GO/RLFT6
   TRACUT/NOMORE
```



Type 6 MDI and APT Cutter Relationship

Normal application
OPP. Part (other sex)

```
   MDI vector is +Z
M8 = Matrix/MIRROR, XY Plan

   MDINO = -N
   RLFT7 = LOFT/MDINO, etc.
   TRACUT/M8
   CALL/APTLFT
   GO/RLFT7
   TRACUT/NOMORE
```



Abnormal application
OPP. Part (other sex)

```
   MDI vector is -Z
M9 = Matrix/MIRROR, XY Plan

   MDINO = N
   RLFT8 = LOFT/MDINO, etc.
   GO/RLFT7
```

APT LOFT ERRORS SECTION 1

1500 Illegal format for PNTVCT

1525 CALL/APTLFT parameters in valid

1541 Difficulty in reading MDI data set in section

1543 MDI not on input data set

1544 MDI has less than two points

1545 Loft definition or MDI is illegal data type

1547 Start and or end points not on MDI

1551 No intersection with MDI was found

1552 MDI called for not on MDI

1553 Each MDI can not be greater than 1205 words

1562 Warning could not find NTH Intersection used --- intersection

APTLFT ERRORS SECTION 2

2150 Difficulty in writing CLTAPE in APTLFT

2151 Difficulty in reading MDI in APTLFT

-2155 MDI is Type 1 and is used as Type 2

2161 Incorrect number of parameters in CALL/APTLFT

2163 Error in beginning or ending points in Loft definition

2164 More than 2411 words on MDI

2165 MDI Type 2 must have less than 804 points

2166 MDI Type 5 multax (ON or OFF) must have less than 402 points.

-2167 APTLFT parameters are ignored except parameter 8

2168 Error inorder of APTLFT statements

2169 Error in reading PROFIL

-2172 Type 2 MDI can not be processed in multax mode processing will continue

2173 Incorrect format for surface extension

2174 Cutter definition not acceptable to APTLFT

2175 Start and end point numbers overlap in APTLFT call

APTLFT ERRORS SECTION 2

2176 25 attempts to position cutter to MDI with useable cutter

-2192 Sucessive CALL/APTLFT not permitted unless GO/LOFT processed

-2197 Start CHK PLN does not intersect MDI

-2198 End CHK PLN does not intersect MDI

-2199 Cutter will not come tangent to start or end plane

2153 MDI is not on external data set

PART 3

APTLFT JCL

TO BUILD SECTION 0, SECTION 1,

SECTION 2 LOAD MODULES;

AND NEW DIAGNOSTIC DATA SET

JCL example to build new member DKWO (Section 0) from SYS1.DKWLM using updates on APTLFT.SOURCE.SHARE.PTF3

Length of DKWO will be approximately AEA8 (44712) bytes.

```
//S1 EXEC PGM=IEBUPDTE
//SYSPRINT DD SYSOUT=A
//SYSUT1 DD UNIT=3330,VOL=SER=ANCA92,DSN=SYS1.DKWSORCE,DISP=SHR
//SYSUT2 DD UNIT=3330,VOL=REF=USR01,DSN=&OH,DISP=(NEW,PASS),
//    SPACE=(CYL,(10,5,4)),DCB=(RECFM=FB,LRECL=80,BLKSIZE=3520)
//SYSIN DD UNIT=3330,VOL=SER=ANCA92,DISP=SHR,
//     DSN=APTLFT.SOURCE.SHARE.PTF3(DKUAM000)
//SYSIN DD *
./      ENDUP
//SSO EXEC ASSEMBLE
//A.SYSGO DD DCB=(RECFM=FB,LRECL=80,BLKSIZE=1680)
//A.SYSIN DD UNIT=3330,VOL=REF=USR01,DISP=(OLD,PASS),DSN=&OH(DKUAM000)
//S4 EXEC PGM=IEWL,PARM='LET,MAP,LIST,OVLY,XREF'
//SYSUT1 DD UNIT=D23,SPACE=(CYL,(5,2))
//SYSLIB DD DSN=SYS1.FORTLIB,DISP=SHR
//SYSPRINT DD SYSOUT=A
//OLDLIB DD DSN=SYS1.DKWLM,UNIT=3330,VOL=SER=ANCA92,DISP=SHR
//SYSLMOD DD DSN=SYS1.DKWLM.SHARE.PTF3,UNIT=3330,VOL=SER=ANCA92,
//    DISP=(NEW,KEEP),SPACE=(CYL,(3,1,5))
//SYSLIN DD DSN=&LOADSET,UNIT=D23,DISP=(OLD,DELETE)
//       DD *
     INCLUDE OLDLIB(DKWO)
     ENTRY ASECT0
     NAME DKWO(R)
/*
```

JCL EXAMPLE TO BUILD NEW MEMBER DKW1 (SECTION 1) from SYS1.DKWLM using updates on APTLFT.SOURCE.SHARE.PTF3. Length of DKW1 will be approximately 4C308(312072) bytes.

If overlay for APTLFT Section 1 is used length of DKW1 will be approximately 39DC8(237000) bytes.

NOTE: Programs SECT1(DKW10000) and ERRTN(DKUE0000) must be first items in load module do to method of establishing common blocks.

```
//S1 EXEC PGM=IEBUPDTE
//SYSPRINT DD SYSOUT=A
//SYSUT1 DD UNIT=3330,VOL=SER=ANCA92,DSN=SYS1.DKWSORCE,DISP=SHR
//SYSUT2 DD UNIT=3330,VOL=REF=USR01,DSN=&OH,DISP=(NEW,PASS),
//       SPACE=(CYL,(10,5,4)),DCB=(RECFM=FB,LRECL=80,BLKSIZE=3520)
//SYSIN DD UNIT=3330,VOL=SER=ANCA92,DISP=SHR,
//       DSN=APTLFT.SOURCE.SHARE.PTF3(DKUEA000)
//         DD UNIT=3330,VOL=SER=ANCA92,DISP=SHR,
//       DSN=APTLFT.SOURCE.SHARE.PTF3(DKUEC000)
//         DD UNIT=3330,VOL=SER=ANCA92,DISP=SHR,
//       DSN=APTLFT.SOURCE.SHARE.PTF3(DKUGD000)
//         DD UNIT=3330,VOL=SER=ANCA92,DISP=SHR,
//       DSN=APTLFT.SOURCE.SHARE.PTF3(DKUHC000)
//         DD UNIT=3330,VOL=SER=ANCA92,DISP=SHR,
//       DSN=APTLFT.SOURCE.SHARE.PTF3(DKUH7000)
//         DD UNIT=3330,VOL=SER=ANCA92,DISP=SHR,
//       DSN=APTLFT.SOURCE.SHARE.PTF3(DKUIF000)
//         DD UNIT=3330,VOL=SER=ANCA92,DISP=SHR,
//       DSN=APTLFT.SOURCE.SHARE.PTF3(DKUJ3000)
//         DD UNIT=3330,VOL=SER=ANCA92,DISP=SHR,
//       DSN=APTLFT.SOURCE.SHARE.PTF3(DKV1P000)
//         DD UNIT=3330,VOL=SER=ANCA92,DISP=SHR,
//       DSN=APTLFT.SOURCE.SHARE.PTF3(DKW10000)
//SYSIN DD *
./       ENDUP
//S2  EXEC ASSEMBLE
//A.SYSGO DD DCB=(RECFM=FB,LRECL=80,BLKSIZE=1680)
//A.SYSIN DD UNIT=3330,VOL=REF=USR01,DISP=(OLD,PASS),DSN=&OH(DKW10000)
//S3  EXEC ASSEMBLE
//A.SYSGO DD DCB=(RECFM=FB,LRECL=80,BLKSIZE=1680)
//A.SYSIN DD UNIT=3330,VOL=SER=ANCA92,DISP=SHR,
//         DSN=SYS1.DKWSORCE(DKUE0000)
//S4  EXEC ASSEMBLE
//A.SYSGO DD DCB=(RECFM=FB,LRECL=80,BLKSIZE=1680)
//A.SYSIN DD UNIT=3330,VOL=REF=USR01,DISP=(OLD,PASS),DSN=&OH(DKUEA000)
//S5  EXEC ASSEMBLE
//A.SYSGO DD DCB=(RECFM=FB,LRECL=80,BLKSIZE=1680)
//A.SYSIN DD UNIT=3330,VOL=REF=USR01,DISP=(OLD,PASS),DSN=&OH(DKUEC000)
//S6  EXEC ASSEMBLE
//A.SYSGO DD DCB=(RECFM=FB,LRECL=80,BLKSIZE=1680)
//A.SYSIN DD UNIT=3330,VOL=REF=USR01,DISP=(OLD,PASS),DSN=&OH(DKUGD000)
//S7  EXEC ASSEMBLE
//A.SYSGO DD DCB=(RECFM=FB,LRECL=80,BLKSIZE=1680)
//A.SYSIN DD UNIT=3330,VOL=REF=USR01,DISP=(OLD,PASS),DSN=&OH(DKUHC000)
//S8  EXEC ASSEMBLE
//A.SYSGO DD DCB=(RECFM=FB,LRECL=80,BLKSIZE=1680)
//A.SYSIN DD UNIT=3330,VOL=REF=USR01,DISP=(OLD,PASS),DSN=&OH(DKUIF000)
//S9  EXEC ASSEMBLE
//A.SYSGO DD DCB=(RECFM=FB,LRECL=80,BLKSIZE=1680)
//A.SYSIN DD UNIT=3330,VOL=REF=USR01,DISP=(OLD,PASS),DSN=&OH(DKUJ3000)
```

```
//S10 EXEC ASSEMBLE
//A.SYSGO DD DCB=(RECFM=FB,LRECL=80,BLKSIZE=1680)
//A.SYSIN DD UNIT=3330,VOL=REF=USR01,DISP=(OLD,PASS),DSN=&OH(DKV1P000)
//S11 EXEC ASSEMBLE
//A.SYSGO DD DCB=(RECFM=FB,LRECL=80,BLKSIZE=1680)
//A.SYSIN DD UNIT=3330,VOL=REF=USR01,DISP=(OLD,PASS),DSN=&OH(DKUH7000)
//S12 EXEC FORTH,PARM.A='OPT=2,XREF,ID,MAP'
//A.SYSIN DD UNIT=3330,VOL=SER=ANCA92,DISP=SHR,
// DSN=APTLFT.SOURCE.SHARE.PTF3(APLDEF)
//       DD UNIT=3330,VOL=SER=ANCA92,DISP=SHR,
// DSN=APTLFT.SOURCE.SHARE.PTF3(BLKDT)
//       DD UNIT=3330,VOL=SER=ANCA92,DISP=SHR,
// DSN=APTLFT.SOURCE.SHARE.PTF3(G)
//       DD UNIT=3330,VOL=SER=ANCA92,DISP=SHR,
// DSN=APTLFT.SOURCE.SHARE.PTF3(GITTA)
//       DD UNIT=3330,VOL=SER=ANCA92,DISP=SHR,
// DSN=APTLFT.SOURCE.SHARE.PTF3(ITHPNT)
//       DD UNIT=3330,VOL=SER=ANCA92,DISP=SHR,
// DSN=APTLFT.SOURCE.SHARE.PTF3(LOFTXY)
//       DD UNIT=3330,VOL=SER=ANCA92,DISP=SHR,
// DSN=APTLFT.SOURCE.SHARE.PTF3(NOZPNT)
//       DD UNIT=3330,VOL=SER=ANCA92,DISP=SHR,
// DSN=APTLFT.SOURCE.SHARE.PTF3(PTVCT2)
//       DD UNIT=3330,VOL=SER=ANCA92,DISP=SHR,
// DSN=APTLFT.SOURCE.SHARE.PTF3(REDUK)
//       DD UNIT=3330,VOL=SER=ANCA92,DISP=SHR,
// DSN=APTLFT.SOURCE.SHARE.PTF3(XLOFTX)
//S13 EXEC PGM=IEWL,PARM='LET,MAP,LIST,OVLY,XREF'
//SYSUT1 DD UNIT=D23,SPACE=(CYL,(5,2))
//SYSLIB DD DSN=FORTLIB,DISP=SHR
//SYSPRINT DD SYSOUT=A
//OLDLIB DD DSN=SYS1.DKWLM,UNIT=3330,VOL=SER=ANCA92,DISP=SHR
//SYSLMOD DD DSN=SYS1.DKWLM.SHARE.PTF3,UNIT=3330,DISP=SHR,
//     VOL=SER=ANCA92
//SYSLIN DD DSN=&LOADSET,UNIT=D23,DISP=(OLD,DELETE)
//       DD *
     INCLUDE OLDLIB(DKW1)
     ENTRY ASECT1
     NAME DKW1(R)
/*
```

JCL EXAMPLE TO BUILD NEW MEMBER DKW2 (SECTION 2) from SYS1.DKWLM using updates on APTLFT.SOURCE. SHARE.PTF3. Length of DKW2 will be approximately 56E90(355984) bytes.

If overlay for APTLFT Section 2 is used, length of DKWR will be approximately 3B920 (244000) bytes.

```
//S1 EXEC PGM=IEBUPDTE
//SYSPRINT DD SYSOUT=A
//SYSUT1 DD UNIT=3330,VOL=SER=ANCA92,DSN=SYS1.DKWSORCE,DISP=SHR
//SYSUT2 DD UNIT=3330,VOL=REF=USR01,DSN=&OH,DISP=(NEW,PASS),
//       SPACE=(CYL,(10,5,4)),DCB=(RECFM=FB,LRECL=80,BLKSIZE=3520)
//SYSIN DD UNIT=3330,VOL=SER=ANCA92,DISP=SHR,
//       DSN=APTLFT.SOURCE.SHARE.PTF3(DKWRC000)
//         DD UNIT=3330,VOL=SER=ANCA92,DISP=SHR,
//       DSN=APTLFT.SOURCE.SHARE.PTF3(DKWTH000)
//SYSIN DD *
./       ENDUP
//S12 EXEC FORTH,PARM.A='OPT=2,XREF,ID,MAP'
//A.SYSIN DD UNIT=3330,VOL=SER=ANCA92,DISP=SHR,
//       DSN=APTLFT.SOURCE.SHARE.PTF3(GITTA2)
//         DD UNIT=3330,VOL=SER=ANCA92,DISP=SHR,
//       DSN=APTLFT.SOURCE.SHARE.PTF3(APTLFT)
//         DD UNIT=3330,VOL=SER=ANCA92,DISP=SHR,
//       DSN=APTLFT.SOURCE.SHARE.PTF3(CKGOUG)
//         DD UNIT=3330,VOL=SER=ANCA92,DISP=SHR,
//       DSN=APTLFT.SOURCE.SHARE.PTF3(SRFEXT)
//         DD UNIT=3330,VOL=SER=ANCA92,DISP=SHR,
//       DSN=APTLFT.SOURCE.SHARE.PTF3(CCPR)
//         DD UNIT=3330,VOL=SER=ANCA92,DISP=SHR,
//       DSN=APTLFT.SOURCE.SHARE.PTF3(INSECT)
//         DD UNIT=3330,VOL=SER=ANCA92,DISP=SHR,
//       DSN=APTLFT.SOURCE.SHARE.PTF3(REDUX)
//         DD UNIT=3330,VOL=SER=ANCA92,DISP=SHR,
//       DSN=APTLFT.SOURCE.SHARE.PTF3(ZVAL)
//         DD UNIT=3330,VOL=SER=ANCA92,DISP=SHR,
//       DSN=APTLFT.SOURCE.SHARE.PTF3(TOLSEG)
//         DD UNIT=3330,VOL=SER=ANCA92,DISP=SHR,
//       DSN=APTLFT.SOURCE.SHARE.PTF3(TYPE6)
//S1BA EXEC FORTH,PARM.A='OPT=2,XREF,ID,MAP'
//A.SYSIN DD UNIT=3330,VOL=REF=USR01,DSN=&OH(DKWRC000),DISP=(OLD,PASS)
//         DD UNIT=3330,VOL=REF=USR01,DSN=&OH(DKWTH000),DISP=(OLD,PASS)
//S13 EXEC PGM=IEWL,PARM='LET,MAP,LIST,OVLY,XREF'
//SYSUT1 DD UNIT=D23,SPACE=(CYL,(5,2))
//SYSLIB DD DSN=FORTLIB,DISP=SHR
//SYSPRINT DD SYSOUT=A
//OLDLIB DD DSN=SYS1.DKWLM,UNIT=3330,VOL=SER=ANCA92,DISP=SHR
//SYSLMOD DD DSN=SYS1.DKWLM.SHARE.PTF3,UNIT=3330,DISP=SHR,
//       VOL=SER=ANCA92
//SYSLIN DD DSN=&LOADSET,UNIT=D23,DISP=(OLD,DELETE)
//         DD *
      INCLUDE OLDLIB(DKW2)
        ENTRY ASECT2
      NAME DKW2(R)
```

JCL example to build a new diagnostic table using SYS1.DKWDAT(DKWZED) and updates on APTLFT.SOURCE.SHARE.PTF3 (DKWZED).

NOTE: DCB=BLKSIZE=4096 in the DD statement will provide page size of 4K.

```
//STP1 EXEC PGM=IEHPRCGM
//SYSPRINT DD SYSOUT=A
//DD3 DD UNIT=3330,VOL=SER=ANCA92,DISP=SHR
//SYSIN DD *
 SCRATCH DSNAME=SYS1.DKWDATA.SHAR,VOL=3330=ANCA92
 SCRATCH DSNAME=APTAC.LIBDPCOL.SHAR,VOL=3330=ANCA92
//S1 EXEC PGM=IEBUPDTE
//SYSPRINT DD SYSOUT=A
//SYSUT1 DD UNIT=3330,VOL=SER=ANCA92,DSN=SYS1.DKWDATA,DISP=(OLD,KEEP)
//SYSUT2 DD UNIT=3330,VOL=SER=ANCA92,SPACE=(CYL,(5,1,2)),
//     DCB=(RECFM=FB,LRECL=80,BLKSIZE=800),DSN=SYS1.DKWDATA.SHAR,
//      DISP=(NEW,PASS)
//SYSIN DD UNIT=3330,VOL=SER=ANCA92,DISP=SHR,
//  DSN=APTLFT.SOURCE.SHARE.PTF3(DKWZED)
//SYSIN DD *
./      ENDUP
//S2 EXEC PGM=DKWBLD,PARM=BUILD,REGION=100K
//STEPLIB DD DSN=SYS1.DKWLM,UNIT=3330,VOL=SER=ANCA92,DISP=SHR
//FT06F001 DD SYSOUT=A
//DPCOL DD DSN=APTAC.LIBDPCOL.SHAR,UNIT=3330,VOL=SER=ANCA92,
//    SPACE=(CYL,(4,1),RLSE),DCB=BLKSIZE=4096,DISP=(NEW,KEEP)
//SYSIN DD UNIT=3330,VOL=SER=ANCA92,DISP=SHR,
//   DSN=SYS1.DKWDATA.SHAR(DKWZED)
```

```
//SHARTAPE JOB '44499R35504E166190                    ',44435069,CLASS=I,      JOB 16
//   MSGLEVEL=(1,1)
//S1 EXEC LABEL
XXLABEL   EXEC PGM=IEHINITT                                                  00000040
XXSYSPRINT DD SYSOUT=A                                                       00000060
   TAPE DD UNIT=(TAPE,1,DEFER),DCB=(DEN=3),VOL=(PRIVATE,RETAIN)              00000080
//LABEL.SYSIN DD *
IEF236I ALLOC. FOR SHARTAPE LABEL    S1
IEF237I 4C2   ALLOCATED TO SYSPRINT
IEF237I 381   ALLOCATED TO TAPE
IEF237I CC1   ALLOCATED TO SYSIN
IEF142I - STEP WAS EXECUTED - COND CODE 0000
IEF373I STEP /LABEL    / START 74295.1236
IEF374I STEP /LABEL    / STOP  74295.1258 CPU    0MIN 00.14SEC MAIN   12K LCS    0K
//S2 EXEC PGM=IEHMOVE
//SYSPRINT DD SYSOUT=A
//SYSUT1 DD UNIT=SYSDA,SPACE=(CYL,(4,1))
//CC2 DD UNIT=3330,VOL=SER=ANCAS2,DISP=SHR
//CC3 DD UNIT=(TAPE,,DEFER),VOL=SER=APLSHR,LABEL=(,SL),DISP=OLD,
//      DCB=(RECFM=FB,LRECL=80,BLKSIZE=3520,DEN=3)
//CC4 DD UNIT=(TAPE,,DEFER),VOL=SER=APLSHR,LABEL=(,SL),DISP=OLD,
//      DCB=(RECFM=FB,LRECL=80,BLKSIZE=800,DEN=3)
//CC5 DD UNIT=(TAPE,,DEFER),VOL=SER=APLSHR,LABEL=(,SL),DISP=OLD,
//      DCB=(RECFM=VB,LRECL=246,BLKSIZE=3972,DEN=3)
//SYSIN DD *
//
IEF236I ALLOC. FOR SHARTAPE S2
IEF237I 4C2   ALLOCATED TO SYSPRINT
IEF237I 1C2   ALLOCATED TO SYSUT1
IEF237I 251   ALLOCATED TO CC2
   F237I 381   ALLOCATED TO CC3
IEF237I 381   ALLOCATED TO CC4
IEF237I 381   ALLOCATED TO CC5
IEF237I CC3   ALLOCATED TO SYSIN
IEF142I - STEP WAS EXECUTED - COND CODE 0004
IEF285I   SYS74294.T090712.RVCCC.SHARTAPE.R0011849      DELETED
IEF285I   VOL SER NOS= MCCUT3.
IEF285I   F14.LONG163.STP4                              KEPT
IEF285I   VOL SER NOS= ANCAS2.
IEF285I   F14.LONG163.STP4                              KEPT
IEF285I   VOL SER NOS= APLSHR.
IEF285I   SYS74294.T090712.RVCCC.SHARTAPE.R0011852      KEPT
IEF285I   VOL SER NOS= APLSHR.
IEF285I   SYS74294.T090712.RVCCC.SHARTAPE.R0011853      KEPT
IEF285I   VOL SER NOS= APLSHR.
IEF373I STEP /S2      / START 74295.1258
IEF374I STEP /S2      / STOP  74295.1309 CPU    0MIN 01.96SEC MAIN   70K LCS    0K
IEF375I  JOB /SHARTAPE/ START 74295.1236
IEF376I  JOB /SHARTAPE/ STOP  74295.1309 CPU    0MIN 02.10SEC
*********************************************************************************
*


SYSTEM SUPPORT UTILITIES      IEHINITT


TAPE     INITT SER=APLSHR,OWNER=ANCA
VOL1APLSHR                              ANCA
```

```
 COPY DSNAME=APTLET.SOURCE.SHARE.PTF3,TO=TAPE=(APLSHR,1),                *
               FROM=3330=ANCA92,TODD=DD3
  H411I DATA SET APTLET.SOURCE.SHARE.PTF3 UNLOADED BECAUSE ACCESS METHOD NOT CO
IEH418I DATA SET APTLET.SOURCE.SHARE.PTF3 NOT MOVED/COPIED BECAUSE DEVICE TYPES
MEMBR APLDEF     HAS BEEN UNLOADED
MEMBR APTLET     HAS BEEN UNLOADED
MEMBR BLKDT      HAS BEEN UNLOADED
MEMBR CCPR       HAS BEEN UNLOADED
MEMBR CKCOUG     HAS BEEN UNLOADED
MEMBR DKUAM000   HAS BEEN UNLOADED
MEMBR DKUEA000   HAS BEEN UNLOADED
MEMBR DKUEC000   HAS BEEN UNLOADED
MEMBR DKUGD000   HAS BEEN UNLOADED
MEMBR DKUHC000   HAS BEEN UNLOADED
MEMBR DKUH7000   HAS BEEN UNLOADED
MEMBR DKUIE000   HAS BEEN UNLOADED
MEMBR DKUJ3000   HAS BEEN UNLOADED
MEMBR DKV1P000   HAS BEEN UNLOADED
MEMBR DKWRC000   HAS BEEN UNLOADED
MEMBR DKWTH000   HAS BEEN UNLOADED
MEMBR DKWZE0     HAS BEEN UNLOADED
MEMBR DKW10000   HAS BEEN UNLOADED
MEMBR G          HAS BEEN UNLOADED
MEMBR GITTA      HAS BEEN UNLOADED
MEMBR GITTA2     HAS BEEN UNLOADED
MEMBR INSECT     HAS BEEN UNLOADED
MEMBR ITHPNT     HAS BEEN UNLOADED
MEMBR LOFTXY     HAS BEEN UNLOADED
MEMBR NOZPNT     HAS BEEN UNLOADED
MEMBR PTVCT2     HAS BEEN UNLOADED
MEMBR REDUK      HAS BEEN UNLOADED
MEMBR REDLX      HAS BEEN UNLOADED
MEMBR SREEXT     HAS BEEN UNLOADED
MEMBR TOL SEG    HAS BEEN UNLOADED
MEMBR TYPE6      HAS BEEN UNLOADED
MEMBR XLOFTX     HAS BEEN UNLOADED
MEMBR ZVAL       HAS BEEN UNLOADED
        DATA SET APTLET.SOURCE.SHARE.PTF3 HAS BEEN COPIED TO VOLUME(S)
                    APLSHR,0001


  COPY DSNAME=APTLET.TESTPRG,TO=TAPE=(APLSHR,2),FROM=3330=ANCA92,      *
               TODD=DD4
IEH411I DATA SET APTLET.TESTPRG UNLOADED BECAUSE ACCESS METHOD NOT COMPATIBLE
IEH418I DATA SET APTLET.TESTPRG NOT MOVED/COPIED BECAUSE DEVICE TYPES NOT COMP
MEMBR TEST1      HAS BEEN UNLOADED
MEMBR TEST2      HAS BEEN UNLOADED
MEMBR TEST3      HAS BEEN UNLOADED
        DATA SET APTLET.TESTPRG HAS BEEN COPIED TO VOLUME(S)
                    APLSHR,0002


 COPY DSNAME=APTACOVY.SHAR,TO=TAPE=(APLSHR,3),FROM=3330=ANCA92,        *
               TODD=DD4
IEH411I DATA SET APTACOVY.SHAR UNLOADED BECAUSE ACCESS METHOD NOT COMPATIBLE
IEH418I DATA SET APTACOVY.SHAR NOT MOVED/COPIED BECAUSE DEVICE TYPES NOT COMPA
MEMBR SECISOVY HAS BEEN UNLOADED
```

SYSTEM SUPPORT UTILITIES ---- IEHMOVE

```
MEMBR SEC2SOVY HAS BEEN UNLOADED
        DATA SET APTACOVY.SHAR HAS BEEN COPIED TO VOLUME(S)
                APLSHR,OCC3

  COPY DSNAME=MDIS.TEST370,TO=TAPE=(APLSHR,4),FROM=3330=ANCA92,TODD=OD5
        DATA SET MDIS.TEST370 HAS BEEN COPIED TO VOLUME(S)
                APLSHR,OCC4

  COPY DSNAME=F14.LONG163.STP4,TO=TAPE=(APLSHR,5),FROM=3330=ANCA92,      *
                TODD=OD5
        DATA SET F14.LONG163.STP4 HAS BEEN COPIED TO VOLUME(S)
                APLSHR,OCC5
```

```
//SHARETST JOB '44499P350348200010                    ',44435069,CLASS=C,
//  MSGLEVEL=(1,1)
//STA EXEC PGM=IEBPTPCH
//DD1 DD UNIT=3330,VOL=SER=ANCA91,DISP=(NEW,KEEP),SPACE=(TRK,(15,2,2)),
//      DCB=(RECFM=FB,LRECL=80,BLKSIZE=3520),DSN=APTLFT.SOURCE.SHARE.PTF3
//DD2 DD UNIT=3330,VOL=SER=ANCA91,DISP=(NEW,KEEP),SPACE=(TRK,(2,1,1)),
//      DCB=(RECFM=FB,LRECL=80,BLKSIZE=800),DSN=APTLFT.TESTPRG
//DD3 DD UNIT=3330,VOL=SER=ANCA91,DISP=(NEW,KEEP),SPACE=(TRK,(2,1,1)),
//      DCB=(RECFM=FB,LRECL=80,BLKSIZE=800),DSN=APTACOVY.SHAR
//DD4 DD UNIT=3330,VOL=SER=ANCA91,DISP=(NEW,KEEP),SPACE=(TRK,(2,1)),
//      DCB=(RECFM=VB,LRECL=248,BLKSIZE=3972),DSN=MDIS.TEST370
//DD5 DD UNIT=3330,VOL=SER=ANCA91,DISP=(NEW,KEEP),SPACE=(TRK,(2,1)),
//      DCB=(RECFM=VB,LRECL=248,BLKSIZE=3972),DSN=F14.LONG163.STP4
//*
//STA2 EXEC PGM=IEHMOVE
//SYSPRINT DD SYSOUT=A
//SYSUT1 DD UNIT=SYSDA,SPACE=(CYL,(4,1))
//DD2 DD UNIT=3330,VOL=SER=ANCA91,DISP=SHR
//DD3 DD UNIT=(TAPE,,DEFER),VOL=SER=APLSHR,LABEL=(,SL),DISP=OLD,
//      DCB=(RECFM=FB,LRECL=80,BLKSIZE=800,DEN=3)
//DD4 DD UNIT=(TAPE,,DEFER),VOL=SER=APLSHR,LABEL=(,SL),DISP=OLD,
//      DCB=(RECFM=FB,LRECL=80,BLKSIZE=800,DEN=3)
//DD5 DD UNIT=(TAPE,,DEFER),VOL=SER=APLSHR,LABEL=(,SL),DISP=OLD,
//      DCB=(RECFM=VB,LRECL=248,BLKSIZE=3972,DEN=3)
//SYSIN DD *
```

		FORTRAN STATEMENT
COPY		DSNAME=APTLFT,SOURCE,SHARE,PTF3,TO=3330=ANCA91,
		FROM=TAPE=(APLSHR,1),FROMDD=DD3
COPY		DSNAME=APTLFT,TESTPRG,TO=3330=ANCA91,
		FROM=TAPE=(APLSHR,2),FROMDD=DD4
COPY		DSNAME=APTACOVY,SHAR,TO=3330=ANCA91,
		FROM=TAPE=(APLSHR,3),FROMDD=DD4
COPY		DSNAME=MDIS,TEST370,TO=3330=ANCA91,
		FROM=TAPE=(APLSHR,4),FROMDD=DD5
COPY		DSNAME=F14,LONG163,STP4,TO=3330=ANCA91,
		FROM=TAPE=(APLSHR,5),FROMDD=DD5