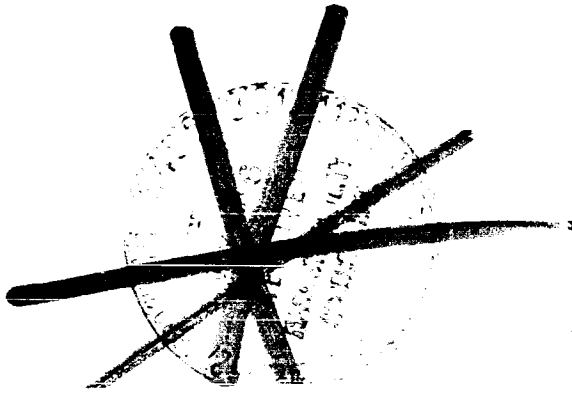
NASA Contractor Report—178052

Evaluation of Automated Decisionmaking Methodologies and Development of an Integrated Robotic System Simulation Appendix B—ROBSIM Programmers Guide



Martin Marietta Aerospace
Denver Aerospace
P.O. Box 179
Denver, Colorado 80201

Contract NAS1-16759

March 1986

Data for general release September 1986

National Aeronautics and Space Administration

Langley Research Center
Hampton, Virginia 23665

EVALUATION OF AUTOMATED DECISIONMAKING METHODOLOGIES AND DEVELOPMENT OF AN INTEGRATED ROBOTIC SYSTEM SIMULATION

Appendix B—ROBSIM Programmer's Guide

Prepared by:

Dennis C. Haley
Bonni J. Almand
Mark M. Thomas
Linda D. Krauze
Keith D. Gremban
James C. Sanborn
Joy H. Kelly
Thomas M. Depkovich
William J. Wolfe
Thuy Nguyen

This work was performed for NASA Langley Research Center under Contract NAS1-16759.

The use of specific equipment or company brand names in this report does not in any way constitute endorsement of those products or companies.

MCR-85-665

MARTIN MARIETTA
DENVER AEROSPACE
P.O. Box 179
Denver, Colorado 80201

APPENDIX B CONTENTS

PRECEDING PAGE BLANK NOT FILMED

Figure

Table

The following is a list of pages modified or added to document the work done under this phase of Contract NAS1-16759 for Langley Research Center.

B-1
B-6
B-7
B-8
B-8a
B-10
B-10a
B-12
B-13
B-14
B-14a
B-15
B-15a
B-16
B-16a
B-17
B-18
B-18a
B-19
B-20
B-20a
B-24
B-24a
B-25
B-25a
B-26
B-26a
B-26b
B-26c
B-27
B-27a
B-29
B-29a
B-31
B-31a
B-40
B-40a
B-43
B-43a
B-48
B-58a
B-51a
B-51b
B-51c
B-51d
B-51e
B-51f
B-51g
B-51h
B-51i
B-51j
B-51k
B-511
B-52
B-52a
B-71
B-71a
B-72
B-72a
B-73
B-73a
B-74
B-74a
B-75
B-75a
B-76
B-76a
B-77
B-77a
B-78
B-78a
B-79
B-79a
B-82
B-82a
B-84
B-84a
B-84b
B-84c
B-84d
B-84e
B-84f
B-84g
B-84h
B-84i
B-93
B-94
B-95
B-96
B-97
B-97a
B-98
B-98a
B-99
B-99a
B-100
B-100a
B-101a
B-101b
B-101c
B-101d
B-101e
B-103
B-103a
B-105
B-105a
B-106
B-106a
B-111
B-111a
B-114
B-114a
B-115
B-115a
B-122
B-122a
B-124a
B-124b
B-124c
B-124d
B-125
B-125a
B-133
B-133a
B-134
B-134a
B-135
B-135a
B-138
B-138a
B-156
B-156a
B-158
B-158a
B-161
B-161a
B-167a
B-167b
B-167c
B-167d
B-167e
B-167f
B-167g
B-167h
B-167i
B-167j
B-167k
B-1671
B-167m
B-167n
B-167o
B-167p
B-167q
B-167r
B-167s
B-167t
B-167u
B-167v
B-172a
B-172b
B-176
B-176a
B-177
B-177a
B-180
B-180a
B-185
B-186
B-187
B-187a
B-188
B-188a
B-189
B-189a
B-190
B-190a
B-191
B-191a
B-192
B-192a
B-193
B-194
B-194a

DOUNTIL: The DOUNTIL is a loop with these characteristics:

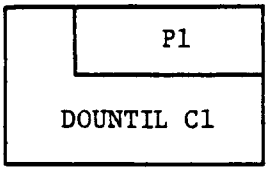
1) The counter or other item to be "incremented" is initialized before entering the loop;

2) The test is performed at the end of the loop. The conditions that must exist to exit from the loop are those that appear in the DOUNTIL test;

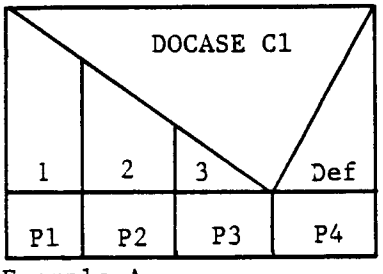
3) The item to be executed must be a standard construct or a single statement;

4) The counter is incremented or other increment-like action is generally taken (e.g., another line is read) at the beginning of the loop.

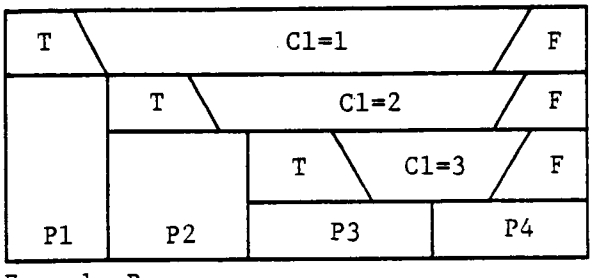
If "C1" is the condition that must exist to exit from the loop and "P1" is a standard construct or single statement, the DOWHILE would be written as



DOCASE: The DOCASE construct is for executing a different set of statements for each of several different values of a variable. If "C1" is the variable being tested and if "C1" may have values 1, 2, or 3, the construct appears



Example A



Example B

Example A is equivalent to the nested IFTHENELSE form shown in B.

Subroutine indexing. - The subroutine descriptions and VCLRs are arranged according to the number assigned in the block diagram. This label consists of three parts (n1.n2.n3). The first part (n1) indicates with which executable--(1) INITDRVR, (2) SIMDRVR, or (3) POSTDRVR--the module is associated. While some routines are used in more than one executable, each is documented only once and labeling number n1 tells which section includes that documentation.

PRECEDING PAGE BLANK NOT FILMED
B1 - B2

The next number (n2) indicates the level in the program hierarchy at which the routine occurs. There are three main levels under each executive driver and a fourth level that is assigned to the utility functions used by a variety of routines. The final number (n3) in the routine label distinguishes the modules within each level of one executable program.

In-code documentation. - Although the information contained in this programmer's guide provides an understanding of the overall program logic and function of the individual subroutines, the bulk of the program documentation is included in the Fortran program modules. This enhances the accessibility of the documentation and allows it to be updated as modifications are made. Each Fortran module contains a preamble that lists the routine's:

1) Purpose;

2) Input (calling arguments and terminal inputs);

3) Output (calling arguments and terminal outputs);

4) Common variables;

5) Internal variables;

6) External references;

7) Functional description;

8) Assumptions and limitations;

9) Special comments;

10) References.

Figure B-1 illustrates an example of this in-code documentation. The file SKLTN.FOR contains a skeleton of the preamble for use in writing new programs.

ORIGINAL PAGE IS
OF POOR QUALITY

```
C*************************************************************
      PROGRAM INITDRVR
C*************************************************************
CD0
CD0   ROBOTICS SIMULATION (ROBSIM) PROGRAM
CD0   SYSTEM DEFINITION FUNCTION ROUTINE
CD0
C*************************************************************
CD1
CD1   PURPOSE
CD1
CD1      The purpose of the ROBSIM Program is to provide a
CD1   broad range of computer capabilities to assist in the
CD1   design, verification, simulation, and study of
CD1   robotic systems.
CD1      The program INITDRVR is the System Definition Function
CD1   driver. It operates in an interactive mode, prompting the
CD1   user for the system definition option desired. Valid options
CD1   are: create or modify an arm data file, create or
CD1   modify a detailed environment file, create or modify load
CD1   objects, create a system data file, or terminate INITDRVR
CD1   program execution.
CD1
C*************************************************************
CD2
CD2   DEFINITION OF INPUT
CD2
CD2      CALLING ARGUMENTS
CD2
CD2         SYMBOL     TYPE  DIM  DEFINITION
CD2
CD2         N/A
CD2
CD2      TERMINAL INPUTS
CD2
CD2         SYMBOL     TYPE  DIM  DEFINITION
CD2
CD2         IMODE      I*4     1  Select flag for mode of operation
CD2                               = 1, Create/modify an arm data file
CD2                               = 2, Create/modify detailed
CD2                                    environment file
CD2                               = 3, Create/modify load objects data file
CD2                               = 4, Create system data file
CD2                               = 5, End ROBSIM INITDRVR executive
CD2         ITERM      I*4     1  Program termination flag,
CD2                               requested following fatal
CD2                               error
CD2                               = 1, Reissue program mode
CD2                                    selection prompt
CD2                               = RETURN, Terminate program
CD2
C*************************************************************
CD3
CD3   DEFINITION OF OUTPUT
CD3
CD3      CALLING ARGUMENTS
CD3
CD3         SYMBOL     TYPE  DIM  DEFINITION
CD3
CD3         N/A
CD3
CD3      TERMINAL OUTPUTS
CD3
CD3         SYMBOL     TYPE  DIM  DEFINITION
CD3
CD3         None
CD3
C*************************************************************
CD4
CD4   COMMON VARIABLES
CD4
CD4      INPUT
CD4
CD4         LU1, LU2
CD4
CD4      OUTPUT
CD4
CD4         NONE
CD4
C*************************************************************
CD5
CD5   INTERNAL VARIABLES
CD5
CD5      SYMBOL    TYPE  DIM  DEFINITION
CD5
CD5      IERROR    I*4     1  Error indicator flag
CD5                            =0, No errors encountered
CD5                            .NE. 0, Error has occurred,
CD5                                    contains appropriate
CD5                                    error number required
CD5                                    by routine ERRMSG
CD5      IHLP      I*4     1  The integer 911 input from terminal to
CD5                           signify user wishes to access help file
CD5      MODE      I*4     1  ROBSIM program mode flag, set to 1
CD5                           to indicate program execution is
CD5                           currently within the System Definition
CD5                           Function
CD5
```

```
C*************************************************************
CD6
CD6   EXTERNAL REFERENCES
CD6
CD6      I/O FILES
CD6
CD6         LU1 - Logical unit assigned for input from the terminal
CD6         LU2 - Logical unit assigned for output to the terminal
CD6
CD6      SCRATCH FILES
CD6
CD6         N/A
CD6
CD6      EXTERNAL ROUTINES
CD6
CD6         BLDARM   - Create/modify arm data file
CD6         BLDENV   - Create/modify detailed environment file
CD6         BLDLOD   - Create/modify load objects data file
CD6         BLDSYS   - Create system data file
CD6         ERRMSG   - Prints error messages for any error occurring
CD6                    during execution
CD6         LBR_HELP - Gains access to the ROBSIM help library
CD6         SETLU    - Default logical units routine
CD6
C*************************************************************
CD7
CD7   FUNCTIONAL DESCRIPTION
CD7
CD7      The ROBSIM command file prompts the user for the
CD7   program function desired. The three ROBSIM program
CD7   functions are System Definition, Analysis Tools, and
CD7   Post Processing. The user may also request program
CD7   termination. Upon receiving a valid function request,
CD7   the ROBSIM command file transfers control to and executes
CD7   the appropriate function driver.
CD7      The System Definition executive calls BLDARM, BLDENV,
CD7   BLDLOD or BLDSYS to create or modify an arm, environment,
CD7   load objects or arm system.
CD7
C*************************************************************
CD8
CD8   ASSUMPTIONS AND LIMITATIONS
CD8
CD8      1. ROBSIM is programmed in FORTRAN 77 for use on
CD8         VAX 11/780 computers under the VMS operating
CD8         system.
CD8      2. ROBSIM uses Evans and Sutherland Multi Picture
CD8         System graphics using MPS FORTRAN callable
CD8         graphics routines. Use of the graphics
CD8         capabilities in ROBSIM is optional, however
CD8         full utilization of the program capabilities
CD8         is greatly limited without the graphics.
CD8
C*************************************************************
CD9
CD9   SPECIAL COMMENTS
CD9
CD9      1. If graphics is desired, the graphics work station
CD9         must be assigned using individual facility
CD9         procedures.
CD9      2. The necessary arm data files must exist prior to
CD9         building a system.
CD9
C*************************************************************
CD10
CD10  REFERENCES
CD10
CD10     None
CD10
C*************************************************************
C     THE FOLLOWING CREATES A HELP LIBRARY MODULE
C*************************************************************
CDX1 INITDRVR
CDX  THIS MODULE IS EXECUTED BY THE SYSTEM DEFINITION PROCESSOR
CDX
CDX2 PARAMETERS
CDX   Qualifiers:
CDX     /IMODE
CDX3 /IMODE
CDX          TYPE   DIM DEFINITION
CDX          I*4      1  Select flag for mode of operation
CDX                        = 1, Create/modify an arm data file
CDX                        = 2, Create/modify detailed
CDX                             environment file
CDX                        = 3, Create/modify load objects data file
CDX                        = 4, Create system data file
CDX                        = 5, End ROBSIM INITDRVR executive
CDX2 FUNCTION
CDX       The ROBSIM command file prompts the user for the
CDX   program function desired. The three ROBSIM program
CDX   functions are System Definition, Analysis Tools, and
CDX   Post Processing. The user may also request program
CDX   termination. Upon receiving a valid function request,
CDX   the ROBSIM command file transfers control to and executes
CDX   the appropriate function driver.
CDX       The System Definition executive calls BLDARM, BLDENV,
CDX   BLDLOD or BLDSYS to create or modify an arm, environment,
CDX   load objects or arm system.
CDX
C*************************************************************
```

Figure B-1.- Example of in-code documentation.

Implementation Notes

This section describes the programming conventions used in implementing ROBSIM on the VAX-11 computer architecture under the VMS operating system. The program consists of a large number of FORTRAN routines and their compiled object modules, along with a limited number of executable images and VMS command files.

Executive-level command file. - The executive level of ROBSIM is handled by an interactive command file ROBSIM.COM. Figure B-2* shows this command file. This file runs one of the ROBSIM executable images (Fig. B-3) selected by the user. PREPDRVR.EXE is designed as a driver for preprocessing data that may be used by the other program executables. INITDRVR.EXE is the executable containing the system definition routines and SIMDRVR.EXE contains the analysis tools image. The postprocessor functions reside in two executable files: (1) POSTDRVT.EXE for video-terminal display of results, and (2) POSTDRHP.EXE for hardcopy plotting. This is because the display software requires linking of different modules for terminal vs hardcopy displays.

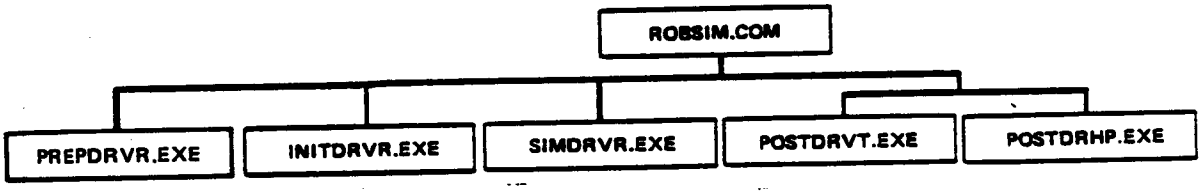


Figure B-3.- ROBSIM executive command file.

Linking the programs. - Each executable image contains an executive routine having the same name as the executable (POSTDRVR for the postprocessor images) and a large number of supporting routines. To facilitate linking, the compiled object modules are included in object libraries. The programs are linked by executing command files that reference these libraries. Figure B-4 shows the linker command files and Table B-I lists the programs in the object libraries.

The bulk of the libraries are contained in the main ROBSIM directory (ROBSIM_DIR:) but some reside in different directories or devices. This is especially convenient for implementations on multiple-disk systems. Table B-II lists the alternate device specifications; these logicals must be assigned to appropriate physical devices, possibly during the log-in procedure.

* Full-page figures can be found at the end of this section.

TABLE B-I. MODULES IN ROBSIM OBJECT LIBRARIES

CNTLLIB.OLB OBJECT LIBRARY MODULES:

CMPCTRL	CNTRSIG	CONTROL
FORREF	FORTOR	ICVTATD
PIDCON	PIDFOR	PIDINIT
POSSENS		

CRLIB.OLB OBJECT LIBRARY MODULES:

BLDARM	BLDDAT	BLDENV
BLDLOD	BLDTRG	BLDSYS
CREATARM	DEFSPJT	ENVIR
LOAD	RDARM	RDARMS
RDENV	RDENVS	RDLOAD
RDLODS	RDTARG	RDTRGS
TARGET	TOTMAS	WRTARM
WRTENV	WRTLOD	WRTSYS
WRTTRG		

EANDSLIB.OLB OBJECT LIBRARY MODULES:

DATOUT	DIALS	ESCNTRL
ESPAUS	FORM	GRAFIX
ISBIT1	LOGO	POSTGRAF
ROTMAT		

GEOMLIB.OLB OBJECT LIBRARY MODULES:

ACTUATOR	BASE	BASELK
BASES	JOINT	LINK
LOCMOD	LOCTMOD	OBJECT
SPAN	TLDMAS	TOOLJT
TOOLLK		

GRAFLIB.OLB OBJECT LIBRARY MODULES:

CADOBJ	CYL	DATATAB
DBAS	DRAW	DRWENV
DRWLOD	DRWTRG	ESMAT
FILLET	GRAPH	GRTERM
MAT	MATVEC	OBSTCL
ORIENT	RCTSTR	RECT
SYSGRAF	TARG	TRISTR

MATHLIB.OLB OBJECT LIBRARY MODULES:

CETM	CRPD	DOT2
GAUSS	MATMPY	REPCOL
SLVLIN	SLVLIN2	SOLVE

TABLE B-I. (cont)

ESCADLIB.OLB OBJECT LIBRARY MODULES:

GRAFCAD		

POSTLIB.OLB OBJECT LIBRARY MODULES:

AXES	HDWMOTIN	LDTHET
MINMAX	ROBPLT	SCAL
SIMOTION	TICMRK	

REQLIB.OLB OBJECT LIBRARY MODULES:

ACTORQ	BASGMNT	BASINIT
BCNTRLR	BORERR	BPCNTRL
BRCNTRL	CABSM	CHKLMT
CNSTFOR	CVTIN	DEBUG
DYNAM	ENDREQ	FORCE
FTIN	JTPOS	LODMOV
ORERR	OUTREQ	PCNTRL
POSGRDJT	POSSPJT	PTACC
PVASPJT	RATEPRO	RCNTRL
REQOPT	REQPLT	REQPRT
REQSOF	REQTRQ	SPRGFOR
SPRGINC	TORQUE	VAGRDJT

SETLIB.OLB OBJECT LIBRARY MODULES:

ADDMAS	ADDMAS2	BASPUT
GRASP	RCICR	RDSYS
RELEAS	SETUP	SETUP2

SIMLIB.OLB OBJECT LIBRARY MODULES:

ACTIVPIH	BDRTORQ	CALCI
DEFCNST	DERIV	DOT
DRTORQ	EFINRT	EFINRT2
ENDSIM	INITCO	INTGRT
LDVOLT	NLINK	OUTSIM
PRTARM	RESPON	SETCNST
SIMLMT	SIMOPT	SIMPLT
SIMPRT	SLVMBAS	SLVTHDD
SLV2ARM	STOPFR	

UTILLIB.OLB OBJECT LIBRARY MODULES:

CVTUNIT	DEFUNIT	ERRMSG
JACOB	LININT	OUTUN
SETLU	ZERCOM	

TABLE B-I. (concl)

PREPLIB.OLB OBJECT LIBRARY MODULES:

BLDCAD		

CADLIB.OLB OBJECT LIBRARY MODULES:

CURVE	LINE	POINT
TRANSF	SPLINE	

TASKLIB.OLB OBJECT LIBRARY MODULES:

ANGLES	CNTRLR	HARALICKR
INITTAR	NEWFRAME	PERSPECT
REQUIR	SEGINIT	SEGTASK
SGMNT	TASK	TRACKING

TABLE B-II. - DEVICE-DIRECTORY SPECIFICATIONS IN ROBSIM

Logical	Modules contained in directory
ROBSIM_DIR:	Basic ROBSIM modules
MPS_DIR:	Evans and Sutherland graphics modules
HELP_DIR:	Help utility modules
DI3000_DIR:	Display modules (video-terminal and hardcopy)
HARDCOPY_DIR:	Modules for creating meta-files from picture files

Fortran files and object files. - Each Fortran routine is included in its own file; the name of the file is that of the routine it contains and the file type is ".FOR". An object file with the same name and type ".OBJ" holds the compiled version of each Fortran file. After a Fortran module is modified, an executable image containing the updated version can be obtained by issuing the following commands:

```
FORTRAN MODULENAME
LIBRARY/REPLACE LIBNAME MODULENAME
@LNKNAME
```

Command files FORROB.COM and REPLIB.COM exist for compiling the entire set of routines in the main directory and updating the object libraries. PRTROB.COM provides for printing all of the Fortran modules. Each of these command files should be updated when routines are added to or deleted from the program.

Fortran COMMON blocks. - The variables used by several routines are arranged into COMMON blocks. The text file ROBCOM.DOC lists and briefly describes the variables included in each COMMON block. A Fortran COMMON statement for each block resides in a file of type ".CMN" that has the same name as the COMMON block. The COMMON blocks are included during compilation of the Fortran modules using the INCLUDE statement. This allows a block to be modified by changing only the ".CMN" file, instead of all the Fortran modules that use this block.

During compilation of the Fortran modules, maximum values must be specified for some of the array dimensions. These maximum dimensions are often defined by PARAMETER statements, and most of these statements are included in the file MXPRMS.TXT. Figure B-5 shows a listing of this file. To change the maximum dimension of some variables (e.g., to increase the number of arms possible in a system), the programmer must only change the appropriate parameter (MXARMS) and recompile the programs.

MXPRMS.TXT

THE PARAMETERS IN THIS INCLUDE FILE ARE CONSTANT VALUES REQUIRED BY THE PROGRAM.

```
INTEGER*2 MAXSPN, IFACTCAD, ISEGCAD

PARAMETER (MXLNKS=10,MXARMS=5)
PARAMETER (MAXORD=3,MAXSEG=20,MXPLTS=31)
PARAMETER (MXENCMPS=30,MXGRCMP=40,MXLDCMPS=10)
PARAMETER (MXPTS=10,MXLODS=10)
PARAMETER (MXTRGCMPS=10,MXTRGS=10)
PARAMETER (MXGRAFPT=5000)
PARAMETER (MAXTRAN=50,MAXLIN=20000,MAXPNT=40)
PARAMETER (MAXARC=300,MAXSPL=200)
PARAMETER (IFLGEND=5557)
PARAMETER (IDIV=10,ISEGCAD=1,IFACTCAD=10)
PARAMETER (ITLNGTH=4,MAXFLD=30000,MAXSPN=500,MAXBRK=4)
```

DEFINITIONS

SYMBOL	TYPE	DIMENSION	DEFINITION
MXARMS	I*4	1	MAXIMUM NUMBER OF ARMS
MXLNKS	I*4	1	MAXIMUM NUMBER LINKS ALLOWABLE
MXPLTS	I*4	1	MAXIMUM NUMBER OF Y ARRAY DATA PARAMETERS WHICH MAY BE WRITTEN TO PLOT FILE
MAXORD	I*4	1	ORDER OF THE POLYNOMIAL DESCRIBING THE MOTION TIME HISTORY
MAXSEG	I*4	1	MAXIMUM NUMBER OF TIME SEGMENTS ALLOWED TO DESCRIBE THE MOTION TIME HISTORY
MXPTS	I*4	1	MAXIMUM NUMBER OF POINT MASSES IN EACH LINK OR LOAD
MXGRCMP	I*4	1	MAXIMUM NUMBER OF GRAPHICS COMPONENTS ALLOWED PER LINK
MXENCMPS	I*4	1	MAXIMUM NUMBER OF GRAPHICS COMPONENTS IN ENVIRONMENT
MXLDCMPS	I*4	1	MAXIMUM NUMBER OF GRAPHICS COMPONENTS IN EACH LOAD OBJECT
MXLODS	I*4	1	MAXIMUM NUMBER OF LOAD OBJECTS ALLOWED
MXTTRCMPS	I*4	1	MAXIMUM NUMBER OF GRAPHICS COMPONENTS IN EACH TARGET OBJECT
MXTRGS	I*4	1	MAXIMUM NUMBER OF TARGET OBJECTS ALLOWED
MXGRAFPT	I*4	1	MAXIMUM NUMBER OF GRAFIX POINTS ALLOWED IN EACH COMPONENT
ISEGCAD	I*2	1	NUMBER OF SEGMENTS FOR THE CAD/CAM GRAPHICS
IFACTCAD	I*2	1	SCALE FACTOR FOR THE CAD/CAM GRAPHICS
MAXSPN	I*2	1	MAXIMUM SPAN OF THE ARM DURING CAD/CAM GRAPHICS

ITLNGTH	I*4	1	LENGTH OF THE CAD/CAM FILE TITLE
IFLGEND	I*4	1	CAD/CAM FILE FLAG SPECIFYING END OF DIRECTORY DATA SECTION IN FILE
MAXFLD	I*4	1	MAXIMUM NUMBER OF LINES IN THE CAD/CAM DATA FILE
MAXARC	I*4	1	MAXIMUM NUMBER ALLOWABLE CAD/CAM ARC ENTITIES
IDIV	I*4	1	NUMBER OF STRAIGHT LINE SEGMENTS INTO WHICH CURVE ENTITY IS DIVIDED FOR CAD/CAM GRAPHICS DISPLAY
MAXLIN	I*4	1	MAXIMUM NUMBER ALLOWABLE CAD/CAM LINE ENTITIES
MAXPNT	I*4	1	MAXIMUM NUMBER ALLOWABLE CAD/CAM POINT ENTITIES
MAXSPL	I*4	1	MAXIMUM NUMBER ALLOWABLE CAD/CAM B-SPLINE ENTITIES
MAXBRK	I*4	1	MAXIMUM ORDER ALLOWED FOR B-SPLINE DEFINING EQUATION DURING CAD/CAM
MAXTRAN	I*4	1	MAXIMUM NUMBER ALLOWABLE CAD/CAM TRANSFORMATION ENTITIES

Figure B-5. - Listing of MXPRMS.TXT.

Interactive help utility. - An interactive help utility is implemented in ROBSIM to provide online assistance to the user for answering some of the program prompts. The utility provides the user with information on the function and form of the routine and its arguments. The utility is implemented using a mixture of custom software and the VMS help utility. Information for the help library is included in the in-code documentation in the FORTRAN modules under the heading "CDX". The command file MAINHLP.COM is executed to set up a help library ROBLIB.HLB from this documentation. The executable image MNEXTRACT.EXE is run to extract the help library information from the FORTRAN modules; it selects all program lines beginning with "CDX" and deletes the "CDX" headings. The formatted file MAINHLP.DOC lists the FORTRAN modules (type ".FOR" implied) to be searched for help library information. All segments that are extracted must follow the conventions for creating help libraries as described in VAX/VMS Volume 4A Program Development Tools Utilities Reference Manual, Section 10.3.2. They are temporarily stored in a file of type ".HLP".

(Warning - all ".HLP" routines are deleted by the command file execution!)

As an example, Figure B-6 shows the help documentation extracted from program POSTDRVR. The help utility is accessed within the ROBSIM program modules by a call to the subroutine LBR_HLP. This module is included in the object library HELP_DIR:QESTLIB.OLB along with the other routines needed for the help utility. Table B-III summarizes the main files employed by the help utility.

```
1 POSTDRVR
  THIS MODULE IS EXECUTED BY THE POST PROCESSOR

2 PARAMETERS
   Qualifiers:
     /IMODE
3 /IMODE
          TYPE    DIM DEFINITION
           I*4      1  Select flag for mode of operation
                       = 1, Replay simulation graphic motion
                            only
                       = 2, Replay simulation versus hardware
                            motion
                       = 3, Parameter versus parameter plots
                       = 4, Return to ROBSIM executive
2 FUNCTION
    The result of executing option 1, is to call subroutine
  SIMOTION which provides a replay of the robotic system motion
  produced from a simulation run.  Option 2 provides a
  comparison of motion resultant from direct hardware theta value
  read and motion resultant from simulation execution, through
  subroutine HDWMOTIN.
  If option 3 is selected, subroutine ROBPLT is called to
  provide parameter versus parameter plots of any of the
  data computed and written to a plot file during the
  Requirements Analysis Tools Function.
  Option 4 returns execution to the primary ROBSIM level.
```

Figure B-6. - Listing of help documentation extracted from POSTDRVR.

TABLE B-III. - FILES USED FOR INTERACTIVE HELP UTILITY

MAINHLP.COM	Executive command file for extracting help library documentation from FORTRAN code
MAINHLP.DOC	Names of FORTRAN modules containing help information
MODULENAME.HLP	Temporary file of help documentation extracted from routine MODULENAME.FOR
ROBLIB.HLB	Data file used for help utility
LIB_HELP.FOR	Module containing program which reads ROBLIB.HLB
HELP_MAC.MAR	Macro routine used to access help facility

Hardcopy utility. - The ROBSIM program provides the capability for interactive display of the manipulator system on an Evans and Sutherland graphics workstation during system creation or analysis. In addition, plots of this display can be generated on a hardcopy plotter for future reference. Generation of the hardcopy plot entails three steps: (1) creation during program execution of a picture file, (2) conversion of this file into a graphics meta-file, and (3) translation of this meta-file into a display or plot.

The first step is initiated by a call to the routine HARD_COP at the points in the program where a hardcopy of the E&S display may be desired. If the user selects to keep a hardcopy of the current display, a picture file named by the user is created. The routines for this procedure are in object libraries HCPIC.OLB and HCMFL.OLB in directory HARDCOPY_DIR: and are linked with the ROBSIM executable images (Fig. B-4). Program execution continues after the meta-file is completed.

After the ROBSIM run terminates, the user can activate programs that convert the picture files into meta-files by executing HCMFL.EXE. The resulting picture file can be translated into a display on the video-terminal or hardcopy plotter using DI3000 software. The images TRANSLATE.EXE and then VTMETTRNS.EXE or HPMETTRNS.EXE in device-directory DI3000_DIR: perform this translation. Activating these images is made easier by assignments in the log-in command file:

```
HCMFL     := RUN HARDCOPY_DIR:HCMFL.EXE

TRANSLATE := RUN DI3000_DIR:TRANSLATE.EXE

HPMETTRNS := RUN DI3000_DIR:HPMETTRNS.EXE

VTMETTRNS := RUN DI3000_DIR:VTMETTRNS.EXE
```

The user need only type the keyword to start execution of these programs.

File "type" conventions. - The different types of files used in creating the ROBSIM program are designated by individual file-type suffixes in their file specifications. It is recommended that the programmer and user maintain these conventions in the files they create. Table B-IV lists the suggested type specifications.

TABLE B-IV. - FILENAME CONVENTIONS USED IN ROBSIM

APPENDAGE FOR FILENAME	DEFINITION
.ARM	ARM GEOMETRY FILE CREATED DURING SYSTEM DEFINITION
.SYS	SYSTEM GEOMETRY FILE CREATED DURING SYSTEM DEFINITION
.LOD	LOAD GEOMETRY FILE CREATED DURING SYSTEM DEFINITION
.ENV	ENVIRONMENT GEOMETRY FILE CREATED DURING SYSTEM DEFINITION
.OBS	OBSTACLE FILE (NONPLANARX,Y,Z COORDINATES) READ FOR DETAILED GEOMETRY INPUT
.ACT	ACTUATOR DEFINITION INPUT FILE READ BY MODULE ACTUATOR DURING SYSTEM DEFINITION
.CON	RESPONSE ANALYSIS CONTROL OPTIONS INPUT FILE READ BY CONTRL MODULE
.THT	HARDWARE THETA ANGLE INPUT FILE CREATED FROM HARDWARE CONVERSION ROUTINE, AND INPUT DURING POST PROCESSING
.VLT	HARDWARE VOLTAGE CONTROL SIGNAL INPUT FILE CREATED FROM HARDWARE CONVERSION ROUTINE, AND INPUT DURING RESPONSE ANALYSIS EXECUTION
.THP	TIME HISTORY PROFILE FILE CREATED BY AN INPUT TO REQUIREMENTS ANALYSIS
.DAT	REQUIRMENTS OR RESPONSE SIMULATION OPTIONS INPUT FILES; ALSO SOME OUTPUT FILES
.CMN	COMMON BLOCKS INCLUDED THROUGHOUT PROGRAM
.OLB	PROGRAM LIBRARY FILES
.LIS	LISTINGS OF SUBROUTINES IN EACH LIBRARY
.FOR	FORTRAN CODE

TABLE B-IV (cont)

.OBJ	FORTRAN OBJECT MODULES
.EXE	ROBSIM PROGRAM AND UTILITIES EXECUTABLES
.COM	COMMAND FILES FOR COMPILING, REPLACING MODULES IN APPROPRIATE LIBRARIES, LINKING THE DRIVERS, PRINTING ALL MODULES, AND RUNNING THE PROGRAMS
.TXT	PARAMETER FILES INCLUDED IN MODULES THROUGHOUT PROGRAM
.DOC	DOCUMENTATION FILES
.HLP	USER HELP FILES GENERATED WITH THE MNEXTRACT UTILITY IN HELPER DIRECTORY ACCESSIBLE WITH THE LBR HELP UTILITY
.PRT	SIMULATION PRINT OUTPUT FILES
.PLT	PLOT OUTPUT FILES FOR HEWLETT PACKARD X-Y PLOTTER OR VT125 GRAPHICS TERMINALS
.AGF	ARM GEOMETRY PRINT OUTPUT FILES CREATED DURING SYSTEM DEFINITION
.PIC	PICTURE FILES OF EVANS AND SUTHERLAND DISPLAYS, GENERATED WITH HARD COP ROUTINE; MAY BE REPRODUCED ON THE HEWLETT PACKARD PLOTTER AFTER CONVERSION TO META-FILE FORMAT
.SOF	SIMULATION OUTPUT FILE FOR POST PROCESSING
.AVT	ACCELERATION-VELOCITY-THETA OUTPUT FILE
.TRQ	TORQUE OUTPUT FILE
.OUT	ACTUAL HARDWARE OUTPUT FILES FOR VOLTAGE CONTROL SIGNALS AND CORRESPONDING THETA ANGLE VALUES
.BMP	BASE MOTION PROFILE CREATED BY AN INPUT TO REQUIREMENTS ANALYSIS
.BTQ	BASE REACTION TORQUES AND FORCES OUTPUT FILE
.NPT	PRINTOUT FILE OF DETAILED GEOMETRY DATA INPUT BY USER

TABLE B-IV (concl)

.CAD	DETAILED GRAPHICS COMPONENT DATA GENERATED FROM CAD/CAM FILE INTERFACE IN PREPROCESSOR
.TRG	TARGET GEOMETRY FILE CREATED DURING SYSTEM DEFINITION
.SGF	SYSTEM GEOMETRY PRINT OUTPUT FILES CREATED WITH UTILITY RDWRTSYS

```
!ROBSIM.COM
$SET NOVERIFY
$SET TERM/NOBROAD
$COUNT=0
$LOOPS:
$COUNT=COUNT+1
$IF COUNT.GT.1 THEN GOTO ASKNEXT
$ASKWICH:
$WRITE SYS$OUTPUT "INPUT (PREPDRVR)-- TO RUN ROBSIM PREPROCESSOR
                                       FUNCTION"
$WRITE SYS$OUTPUT "INPUT (INITDRVR)-- TO RUN ROBSIM SYSTEM
                                       DEFINITION FUNCTION"
$WRITE SYS$OUTPUT "      (SIMDRVR) -- TO RUN ROBSIM
                                       SIMULATION ANALYSIS
                                       TOOLS FUNCTION"
$WRITE SYS$OUTPUT "      (POSTDRVR)-- TO RUN ROBSIM POST
                                       PROCESSOR FUNCTION"
$PROMPT:=WHICH:
$READ/PROMPT="''PROMPT'" SYS$COMMAND WHICH
$IF WHICH.EQS."INITDRVR" THEN GOTO INIT
$IF WHICH.EQS."SIMDRVR" THEN GOTO SIM
$IF WHICH.EQS."POSTDRVR" THEN GOTO POST
$PREP
$ASSIGN TT SYS$INPUT
$ASSIGN TT SYS$OUTPUT
$RUN [ROBSIM.WORK]PREPDRVR
$DEASSIGN SYS$INPUT
$DEASSIGN SYS$OUTPUT
$INQUIRR
$PROMPT:=INPUT(Y) TO RUN PREPROCESSOR FUNCTION AGAIN,-
 (OTHERWISE, RETURN)
$READ/PROMPT="''PROMPT'" SYS$COMMAND WHICH
$IF WHICH.EQS."Y" THEN GOTO PREP
$GOTO LOOPS
$INIT:
$ASSIGN TT SYS$INPUT
$ASSIGN TT SYS$OUTPUT
$RUN [ROBSIM.WORK]INITDRVR
$DEASSIGN SYS$INPUT
$DEASSIGN SYS$OUTPUT
$INQUIRI:
$PROMPT:=INPUT (Y) TO RUN SYSTEM DEFINITION FUNCTION AGAIN, -
 (OTHERWISE, RETURN)
$READ/PROMPT="''PROMPT'" SYS$COMMAND WHICH
$IF WHICH.EQS."Y" THEN GOTO INIT
$GOTO LOOPS
$SIM:
$ASSIGN TT SYS$INPUT
$ASSIGN TT SYS$OUTPUT
```

Figure B-2. - The ROBSIM executive command file.

```
$RUN [ROBSIM.WORK]SIMDRVR
$DEASSIGN SYS$INPUT
$DEASSIGN SYS$OUTPUT
$INQUIRS:
$PROMPT:=INPUT (Y) TO RUN SIMULATION ANALYSIS TOOLS -
 FUNCTION AGAIN, (OTHERWISE, RETURN)
$READ/PROMPT="''PROMPT'" SYS$COMMAND WHICH
$IF WHICH.EQS."Y" THEN GOTO SIM
$GOTO LOOPS
$POST:
$WRITE SYS$OUTPUT "DO YOU WISH (1) TERMINAL OR
                                (2) HARDCOPY PLOTTING?"
$PROMPT:=   ENTER INTEGER:
$READ/PROMPT="''PROMPT'" SYS$COMMAND WHICH
$ASSIGN TT SYS$INPUT
$ASSIGN TT SYS$OUTPUT
$IF WHICH.EQS."1" THEN RUN [ROBSIM.WORK]POSTDRVT
$IF WHICH.EQS."2" THEN RUN [ROBSIM.WORK]POSTDRHP
$DEASSIGN SYS$INPUT
$DEASSIGN SYS$OUTPUT
$INQUIRP:
$PROMPT:=INPUT (Y) TO RUN POST PROCESSOR FUNCTION AGAIN, -
 (OTHERWISE, RETURN)
$READ/PROMPT="''PROMPT'" SYS$COMMAND WHICH
$IF WHICH.EQS."Y" THEN GOTO POST
$GOTO LOOPS
$ASKNEXT:
$PROMPT:=INPUT (Q) IF YOU WISH TO EXIT THE PROGRAM
                                        (OTHERWISE, RETURN)
$READ/PROMPT="''PROMPT'" SYS$COMMAND QUIT
$IF QUIT.EQS."" THEN GOTO ASKWICH
$EXIT
$STOP
```

Figure B-2. (concl)

LNKPREP.COM

```
$LINK/EXECUTABLE=PREPDRVR   PREPDRVR,-
        PREPLIB/LIB,-
        CADLIB/LIB,-
        ESCADLIB/LIB,-
        UTILLIB/LIB,-
        MATHLIB/LIB,-
        HCPIC/LIB,-
        HCMFL/LIB,-
        DISK$USER1:[ROBSIM.HELPER]QESTLIB/LIB,-
        SYS$SYSDEVICE:[MPSGSP]MPLIB/L
```

LNKINIT.COM

```
$LINK/EXECUTABLE=INITDRVR   INITDRVR,-
        CRLIB/LIB,-
        GEOMLIB/LIB,-
        GRAFLIB/LIB,-
        SETLIB/LIB,-
        EANDSLIB/LIB,-
        UTILLIB/LIB,-
        MATHLIB/LIB,-
        HCPIC/LIB,-
        HCMFL/LIB,-
        DISK$USER1:[ROBSIM.HELPER]QESTLIB/LIB,-
        SYS$SYSDEVICE:[MPSGSP]MPLIB/L
```

LNKSIM.COM

```
$LINK/EXECUTABLE=SIMDRVR   SIMDRVR,-
        SIMLIB/LIB,-
        CNTLLIB/LIB,-
        TASKLIB/LIB,-
        REQLIB/LIB,-
        EANDSLIB/LIB,-
        UTILLIB/LIB,-
        SETLIB/LIB,-
        MATHLIB/LIB,-
        HCPIC/LIB,-
        HCMFL/LIB,-
        DISK$USER1:[ROBSIM.HELPER]QESTLIB/LIB,-
        SYS$SYSDEVICE:[MPSGSP]MPLIB/L
```

Figure B-4 - ROBSIM linker command files.

LNKPOSTVT.COM

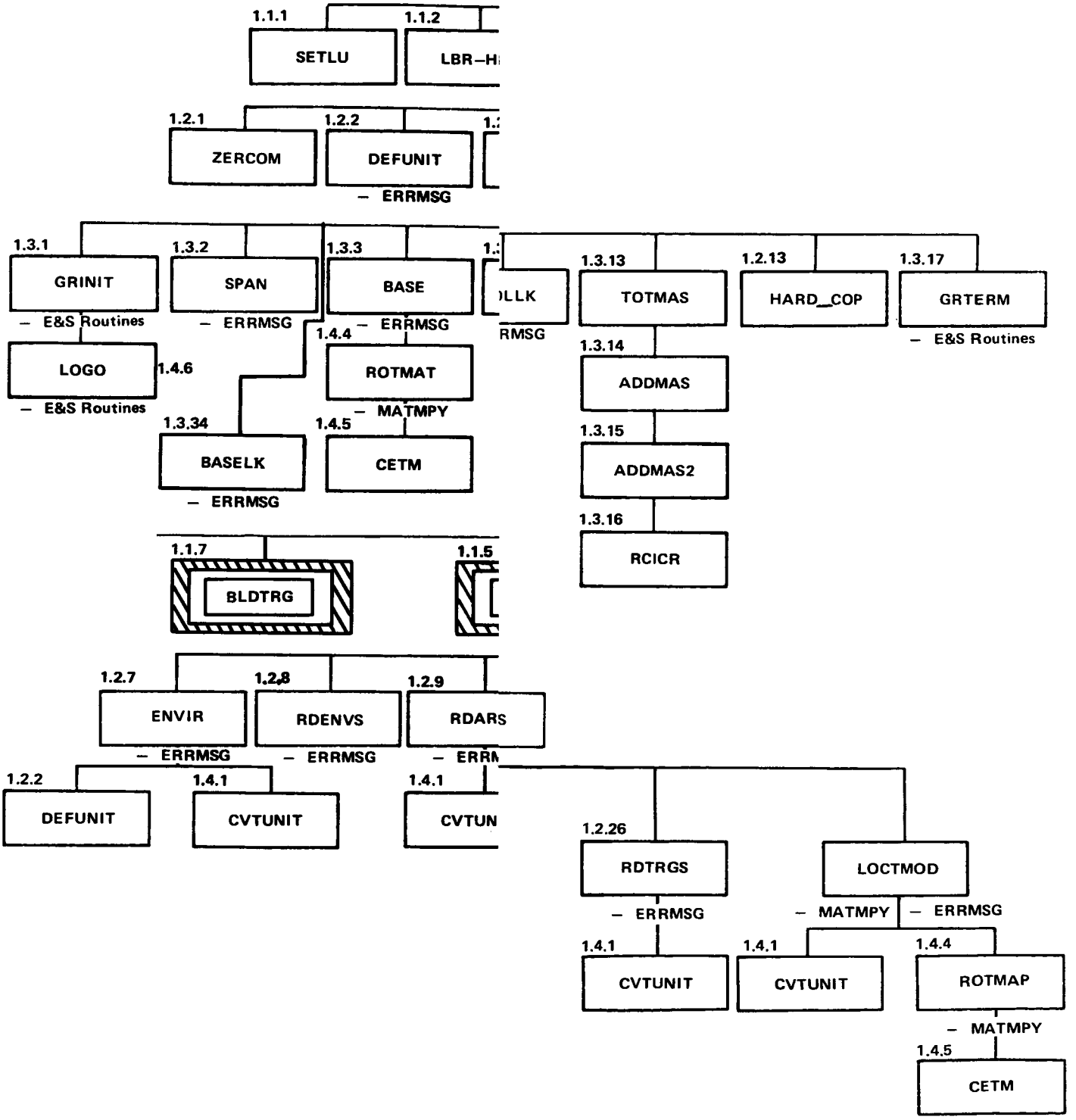
```
$LINK/EXECUTABLE=POSTDRVT    POSTDRVR,-
        POSTLIB/LIB,-
        SETLIB/LIB,-
        UTILLIB/LIB,-
        MATHLIB/LIB,-
        EANDSLIB/LIB,-
        HCPIC/LIB,-
        HCMFL/LIB,-
        DISK$USER1:[ROBSIM.HELPER]QESTLIB/LIB,-
        SYS$SYSDEVICE:[MPSGSP]MPLIB/L
        di3_link:DILIB/LIB,MFNODE/OPT, LVLC/OPT,-
        DI3_DDR:DDR125,-
        DI3_LINK:UTILLIB/LIB
```

LNKPOSTHP.COM

```
$LINK/EXECUTABLE=POSTDRHP    POSTDRVR,-
        POSTLIB/LIB,-
        SETLIB/LIB,-
        UTILLIB/LIB,-
        MATHLIB/LIB,-
        EANDSLIB/LIB,-
        HCPIC/LIB,-
        HCMFL/LIB,-
        DISK$USER1:[ROBSIM.HELPER]QESTLIB/LIB
        SYS$SYSDEVICE:[MPSGSP]MPLIB/L
        DI3_CS:MGRAIN,Q3ATOC,-
        di3_link:DILIB/LIB,MFNODE/OPT,LVLC/OPT,-
        DI3_DDR:DDR721,-
        DI3_LINK:UTILLIB/LIB
```

Figure B-4. (Concl)

The program INITDRVR is the system
shows the program modules employed in I
modules. The set of functional descrip
following pages describe these routines



FOLDOUT FRAME

FOLDOUT FRAME

ORIGINAL PAGE IS
OF POOR QUALITY

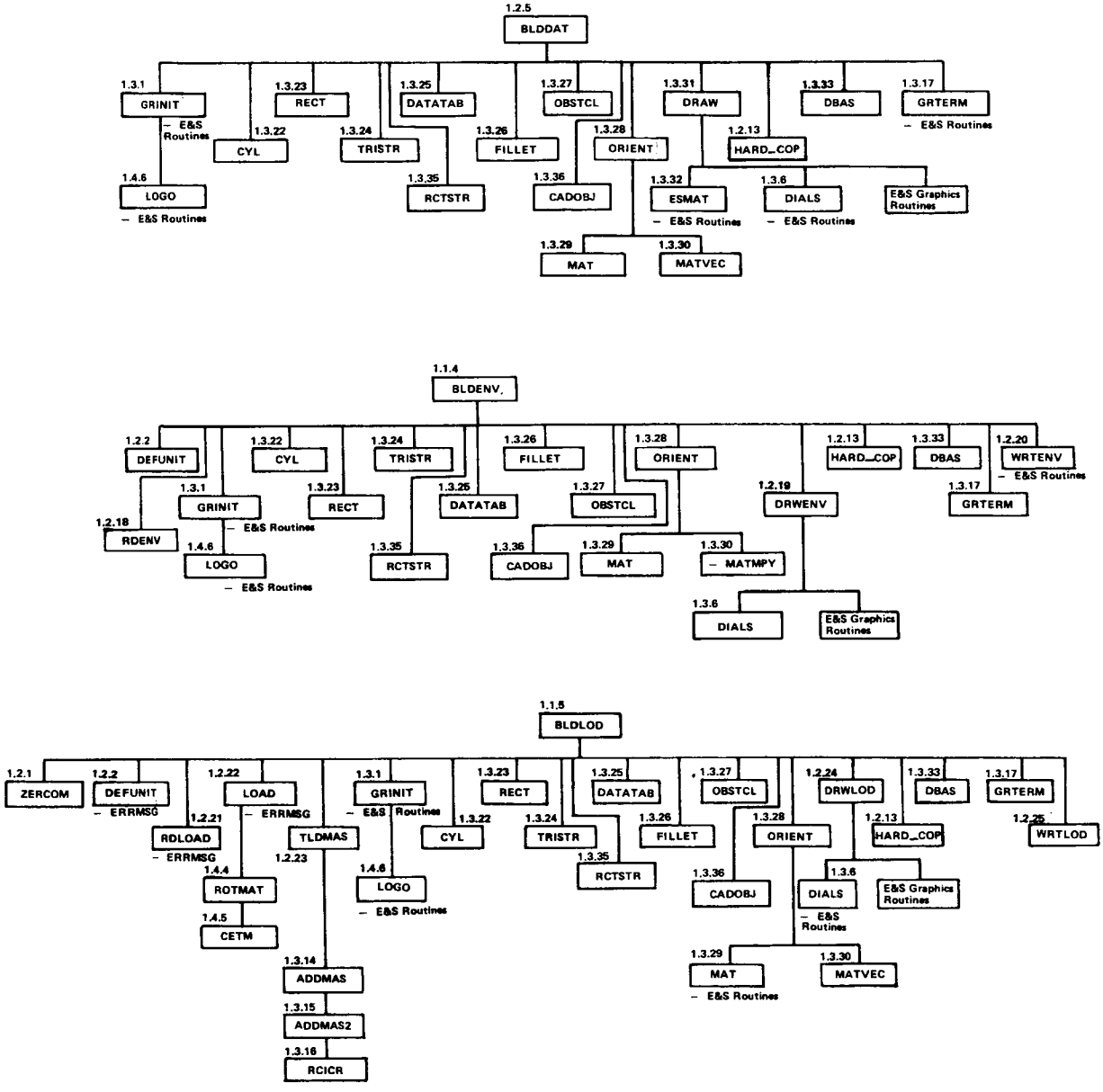


Figure B-7. (cont)

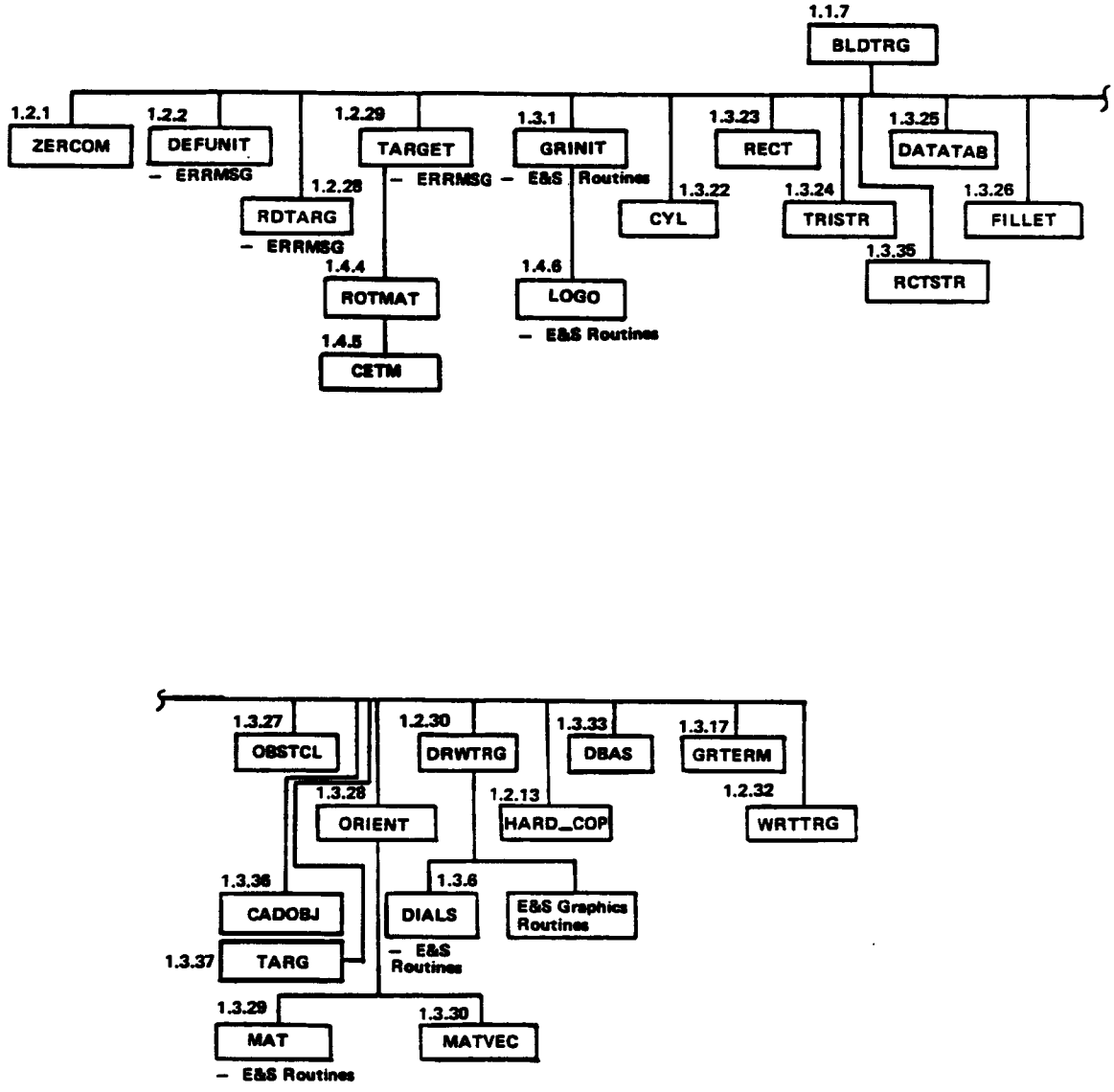


Figure B-7. (concl)

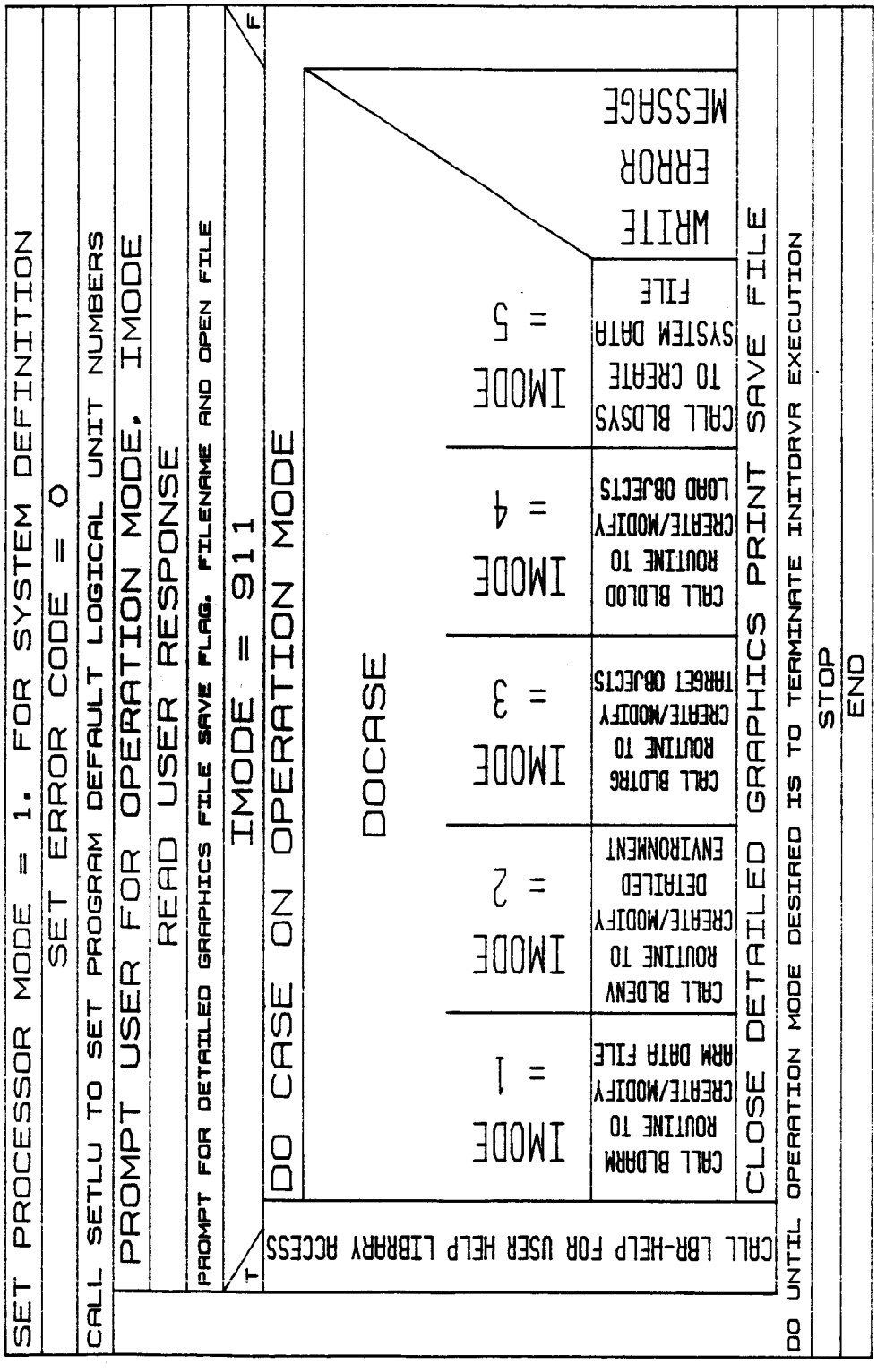
TABLE B-V. - PROGRAMS EMPLOYED IN INITDRVR

1.0	INITDRVR	1.3.1	GRINIT	1.4.1	CVTUNIT
		1.3.2	SPAN	1.4.2	MATMPY
1.1.1	SETLU	1.3.3	BASE	1.4.3	ERRMSG
1.1.2	LBR_HELP	1.3.4	OBJECT	1.4.4	ROTMAT
1.1.3	BLDARM	1.3.5	GRAPH	1.4.5	CETM
1.1.4	BLDENV	1.3.6	DIALS	1.4.6	LOGO
1.1.5	BLDLOD	1.3.7	JOINT	1.4.7	CRPD
1.1.6	BLDSYS	1.3.8	ACTUATOR		
1.1.7	BLDTRG	1.3.9	LINK		
		1.3.10	DEFSPJT		
1.2.1	ZERCOM	1.3.11	TOOLJT		
1.2.2	DEFUNIT	1.3.12	TOOLLK		
1.2.3	CREATARM	1.3.13	TOTMAS		
1.2.4	RDARM	1.3.14	ADDMAS		
1.2.5	BLDDAT	1.3.15	ADDMAS2		
1.2.6	WRTARM	1.3.16	RCICR		
1.2.7	ENVIR	1.3.17	GRTERM		
1.2.8	RDENVS	1.3.18	BASPUT		
1.2.9	RDARMS	1.3.19	JACOB		
1.2.10	BASES	1.3.20	DATOUT		
1.2.11	SETUP	1.3.21	FORM		
1.2.12	SETUP2	1.3.22	CYL		
1.2.13	HARD_COP	1.3.23	RECT		
1.2.14	SYSGRAF	1.3.24	TRISTR		
1.2.15	RDLODS	1.3.25	DATATAB		
1.2.16	LOCMOD	1.3.26	FILLET		
1.2.17	WRTSYS	1.3.27	OBSTCL		
1.2.18	RDENV	1.3.28	ORIENT		
1.2.19	DRWENV	1.3.29	MAT		
1.2.20	WRTENV	1.3.30	MATVEC		
1.2.21	RDLOAD	1.3.31	DRAW		
1.2.22	LOAD	1.3.32	ESMAT		
1.2.23	TLDMAS	1.3.33	DBAS		
1.2.24	DRWLOD	1.3.34	BASELK		
1.2.25	WRTLOD	1.3.35	RCTSTR		
1.2.26	RDTRGS	1.3.36	CADOBJ		
1.2.27	LOCTMOD	1.3.37	TARG		
1.2.28	RDTARG				
1.2.29	TARGET				
1.2.30	DRWTRG				
1.2.31	WRTTRG				

1.0 INITDRVR

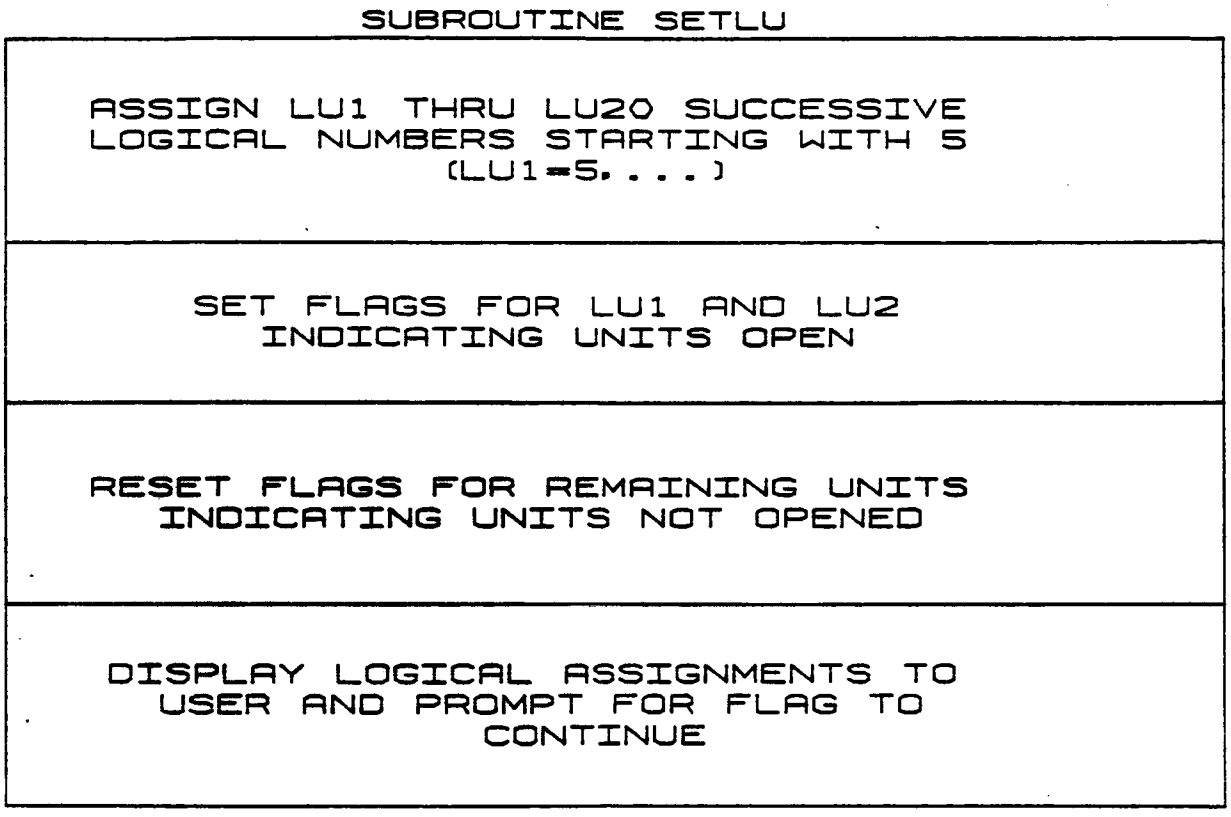
The program INITDRVR is the system definition function driver. It operates in an interactive mode, prompting the user for the system definition option desired--create or modify an arm data file, create or modify a detailed environment file, create or modify a target objects file, create or modify a load objects file, create a system data file, or terminate INITDRVR execution. Subroutine SETLU is called to set the Fortran logical units. The necessary simple cylinder or detailed single arm file must exist prior to building a system. A detailed graphics save file is opened if requested.

ORIGINAL PAGE IS
OF POOR QUALITY



1.1.1 SETLU

SETLU is called from the various executive drivers to set the Fortran logical unit number to be stored in COMMON block LUNITBK for reference by the rest of the ROBSIM program. After assigning the variables to consecutive unit numbers, the "unit open" flags are reset to indicate the units are not open (except the terminal read and write units).



1.1.2 LBR_HELP

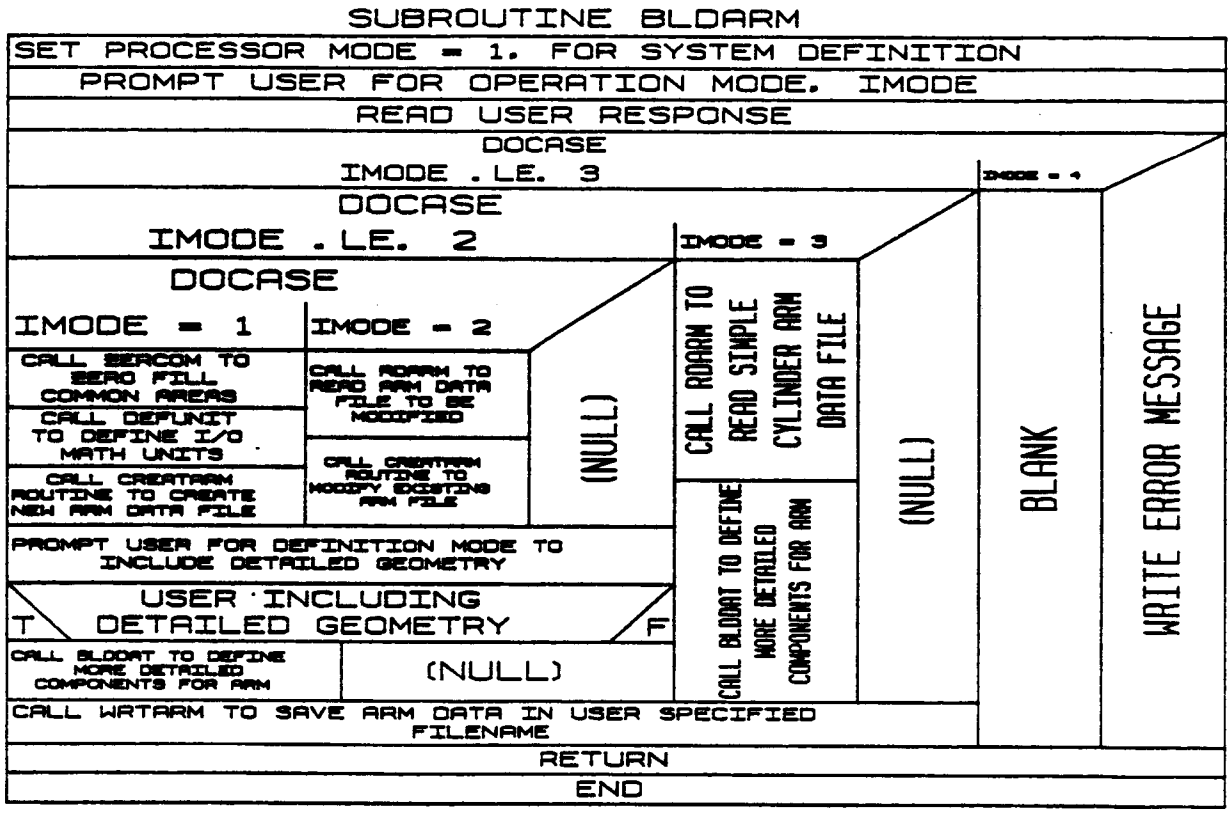
Subroutine LBR_HELP is called to execute the help utility during a ROBSIM run. It uses the object file created from the macro HELPMAC.MAR and runs the system help utilities as required.

(VCLR for LBR_HELP not available.)

ORIGINAL PAGE IS
OF POOR QUALITY

1.1.3 BLDARM

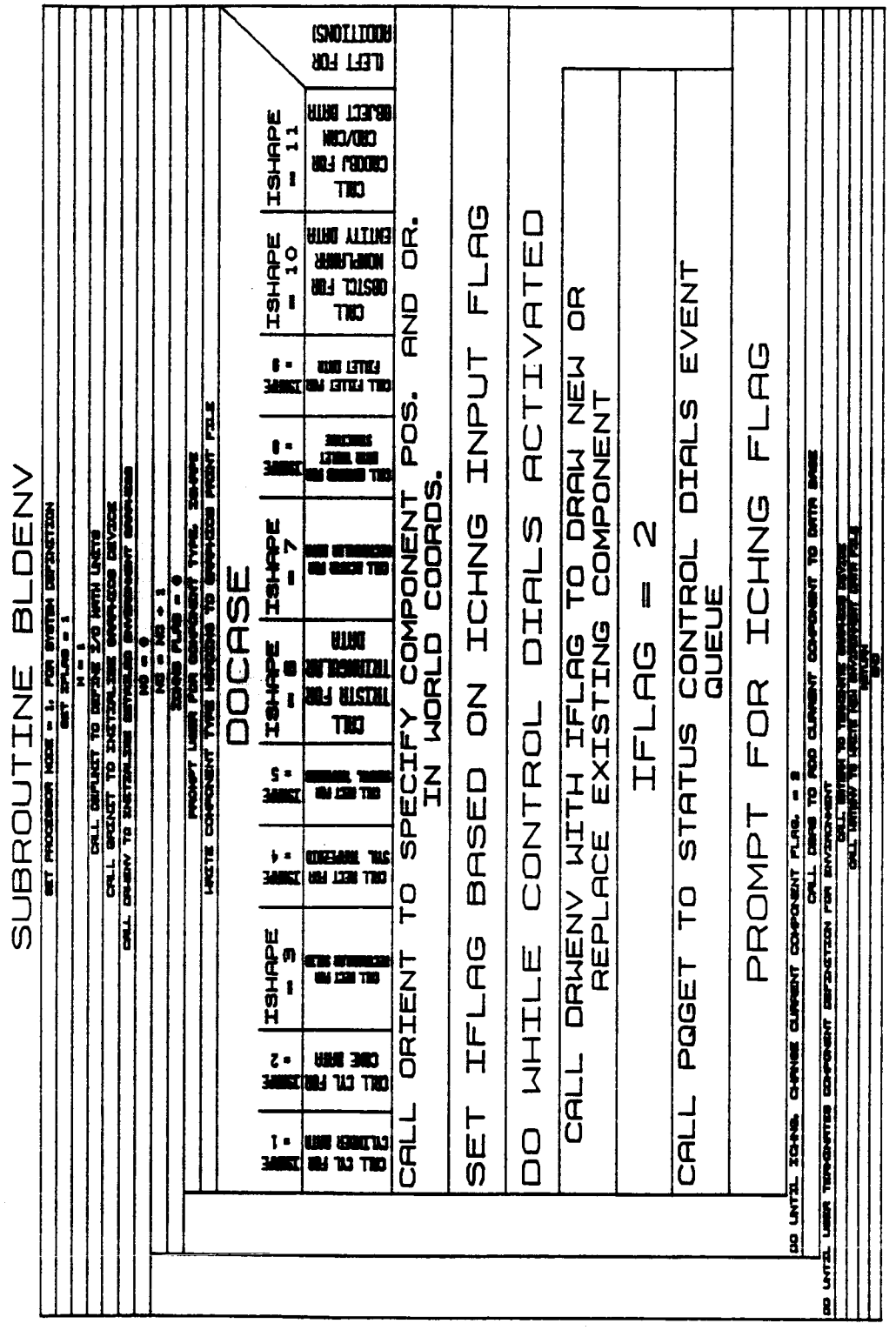
BLDARM is met when a selection of 1, to create or modify an arm data file, is entered from INITDRVR. The user choices for mode of operation are (1) create a simple cylinder arm data file, (2) modify existing arm data file, (3) enter detailed graphics data for arm (a simple cylinder file must already exist), or (4) terminate arm definition and return to the INITDRVR. For initial creation, option (1), subroutine ZERCOM is called to zero the COMMON locations and then CREATARM is called to build the new data file. For modification, RDARM and CREATARM are called when option (2) is requested. BLDDAT is responsible for the invention of detailed arm geometry. In all cases, WRTARM will be called to write the arm data COMMON information.



1.1.4 BLDENV

The user has the capability with routine BLDENV to specify a detailed physical representation for the robotic environment. Components for the environment are defined as basic geometric shapes (cylinders, cones, rectangular solids, symmetric or nonsymmetric trapezoidal figures, triangular cross-sectional beams, rectangular beams, fillet components, data tablet-defined entities, obstacles, and CAD/CAM objects. The component type is written to the detailed graphics save file if requested.

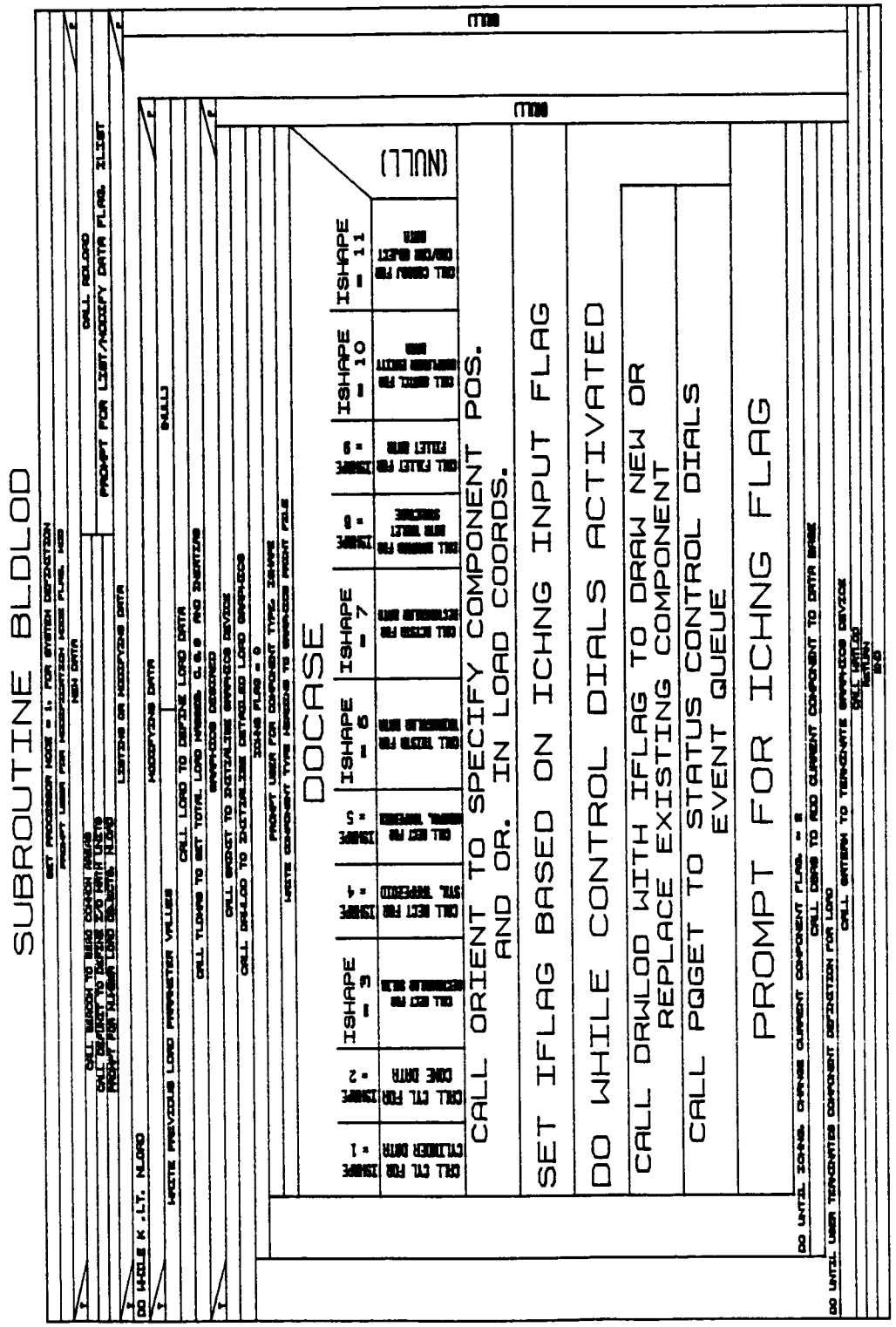
ORIGINAL PAGE IS
OF POOR QUALITY



1.1.5 BLDLOD

Through routine BLDLOD, the user has the capability to specify a detailed physical representation for the robotic load objects to be used. Components for the load objects are defined as basic geometric shapes (cylinders, cones, rectangular solids, symmetric or nonsymmetric trapezoidal figures, triangular cross-sectional beams, rectangular beams, fillet components, data tablet-defined entities, nonplanar entities, and CAD/CAM objects). This subroutine creates a new file, or modifies an existing file of load objects, and includes the capability to specify the detail at the first creation session for the load objects. The component type is written to the detailed graphics save file if requested.

ORIGINAL PAGE IS
OF POOR QUALITY



1.1.6 BLDSYS

BLDSYS prompts the user for the moving base information and then reads individual arm data files into the appropriate system COMMON blocks. It desired, the locations of the fixed bases can be modified. BLDSYS then sets up the environment, load objects, target data, and stores the system file.

SUBROUTINE BLDSYS

PROMPT FOR GRAPHICS DISPLAY DURING SYSTEM DEFINITION

CALL ENVIR FOR USER INPUT OF ENVIRONMENT SET PROPERTIES

CALL RDENVS TO READ ENVIRONMENT DATA IF REQUESTED AND GRAPHICS OPTED

CALL SYSGRAF TO DRAW ENVIRONMENT WHEN GRAPHICS IS DESIRED

PROMPT FOR NUMBER OF MOVING BASES. NBAS. IF TO BE INCLUDED

PROMPT FOR NUMBER OF ROBOTIC ARMS IN SYSTEM. NARM

DO WHILE KARM .LT. NARM

CALL RDARMS TO READ ARM DATA INTO SYSTEM COMMON BLOCKS

INCLUDING MOVING BASES? (T / F)

- T: PROMPT FOR WHICH MOVING BASE ARM IS ATTACHED TO. KBASNUM (KARM)
- F: MODIFY BASE LOCATION OR ORIENTATION IF REQUESTED

CALL BASES TO PROMPT FOR CHANGES

GRAPHICS DESIRED? (T / F)

- T: CALL SETUP FOR ROBL /ROT. AND CALL SYSGRAF TO DRAW ARM WITH DIALS
- F: (NULL)

PROMPT USER FOR DEFINITION MODE TO INCLUDE LOAD OBJECTS

CALL RDLODS TO READ LOAD OBJECTS DATA IF INCLUDING LOADS

DO WHILE K .LT. NLOAD

CHANGE CURRENT LOAD LOC. AND OR. IF REQUESTED

GRAPHICS DESIRED? (T / F)

- T: CALL SYSGRAF TO DRAW CURRENT LOAD WITH DIALS
- F: (NULL)

CALL LOCMOD

PROMPT USER FOR DEFINITION MODE TO INCLUDE TARGETS

CALL RDTRGS TO READ TARGET DATA IF INCLUDING TARGETS

DO WHILE K .LT. NTARG

CHANGE CURRENT TARGET LOC. AND OR. IF REQUESTED

GRAPHICS DESIRED? (T / F)

- T: CALL SYSGRAF TO DRAW CURRENT TARGET WITH DIALS
- F: (NULL)

CALL LOCTMOD

CALL SYSGRAF TO TERMINATE GRAPHICS IF NECESSARY

CALL WRTSYS TO WRITE ROBOTIC SYSTEM FILE IN USER SPECIFIED FILENAME

RETURN

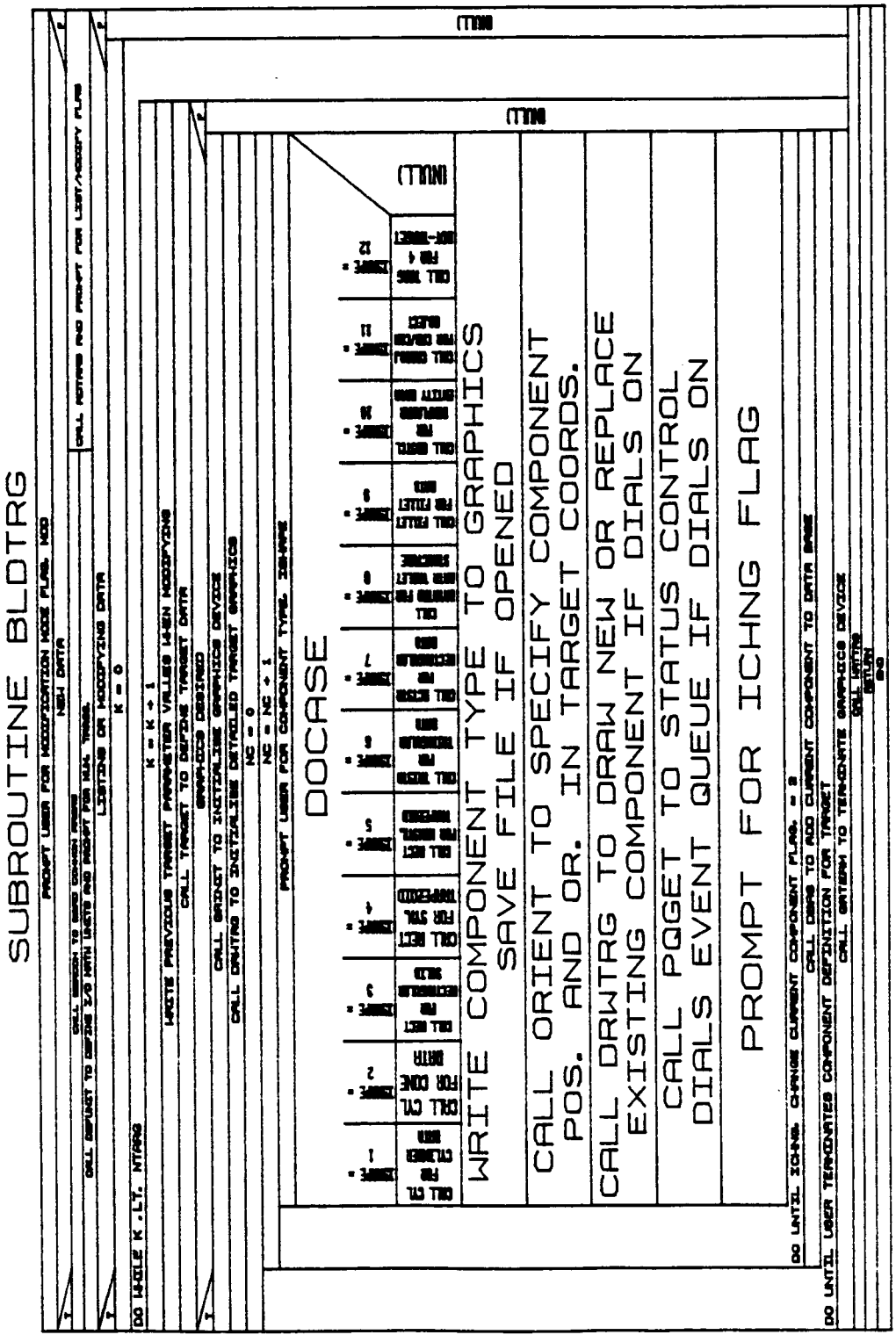
END

ORIGINAL PAGE IS
OF POOR QUALITY

1.1.7 BLDTRG

Through routine BLDTRG, the user can specify a detailed physical representation for the robotic target objects to be used. Components for the target objects are defined as basic geometric shapes (cylinders, cones, rectangular solids, symmetric or nonsymmetric trapezoidal figures, triangular cross-sectional beams, rectangular beams, fillet components, data tablet-defined entities, obstacles, CAD/CAM objects and four-dot target patterns). This subroutine creates a new file or modifies an existing file of target objects, and includes the capability to specify the detail at the first creation session for the target objects.

ORIGINAL PAGE IS
OF POOR QUALITY



1.2.1 ZERCOM

Subroutine ZERCOM is called from BLDARM, BLDLOD, and BLDTRG to initialize the arm, load, and target data COMMON blocks prior to creating new system files. The COMMON blocks initialized to zero include:

1) BLDGBK - arm geometric properties;

2) BLDMBK - arm mass properties;

3) GRAFBK - arm graphics data;

4) BLDABK - arm actuator parameters;

5) BLDSJBK - special joint flags;

6) LOADBK - load geometry and mass properties;

7) LGRAFBK - load graphics data;

8) TARGBK - target geometry and mass properties;

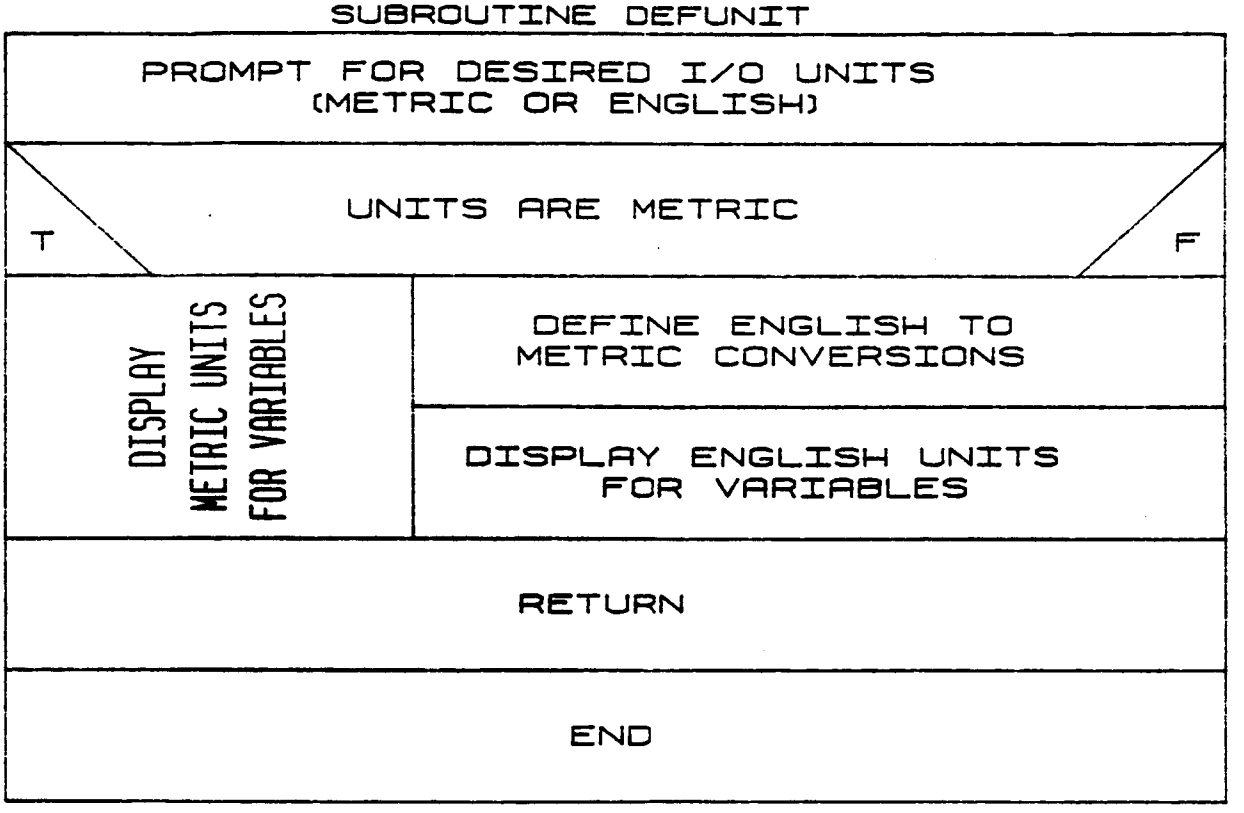
9) TGRAFBK - target graphics data.

SUBROUTINE ZERCOM

NJ = MAXIMUM NUMBER OF LINKS
ZERO VARIABLES IN GEOMETRIC PROPERTIES COMMON BLOCK BLDGBK
DO UNTIL N = NUMBER OF LINKS IN CURRENT ARM
ZERO VARIABLES IN MASS PROPERTIES COMMON BLOCK BLDMBK
DO UNTIL N = NUMBER OF LINKS IN CURRENT ARM
ZERO VARIABLES IN ARM GRAPHICS COMMON BLOCK GRAFBK
ZERO VARIABLES IN ACTUATOR PARAMETERS COMMON BLOCK BLDABK
DO UNTIL N = NUMBER OF JOINTS IN CURRENT ARM
ZERO VARIABLES IN SPECIAL JOINTS COMMON BLOCK BLDSJBK
DO UNTIL N = NUMBER OF SPECIAL JOINTS IN CURRENT ARM
NJ = 0
ZERO VARIABLES IN LOAD GEOMETRY AND MASS COMMON BLOCK LOADBK
DO UNTIL N = NUMBER OF LOADS IN LOAD OBJECT FILE
ZERO VARIABLES IN LOAD GRAPHICS COMMON BLOCK LGRAFBK
ZERO VARIABLES IN TARGET GEOMETRY COMMON BLOCK TARGBK
DO UNTIL N = NUMBER OF TARGETS IN TARGET FILE
ZERO VARIABLES IN TARGET GRAPHICS COMMON BLOCK TGRAFBK
RETURN
END

1.2.2 DEFUNIT

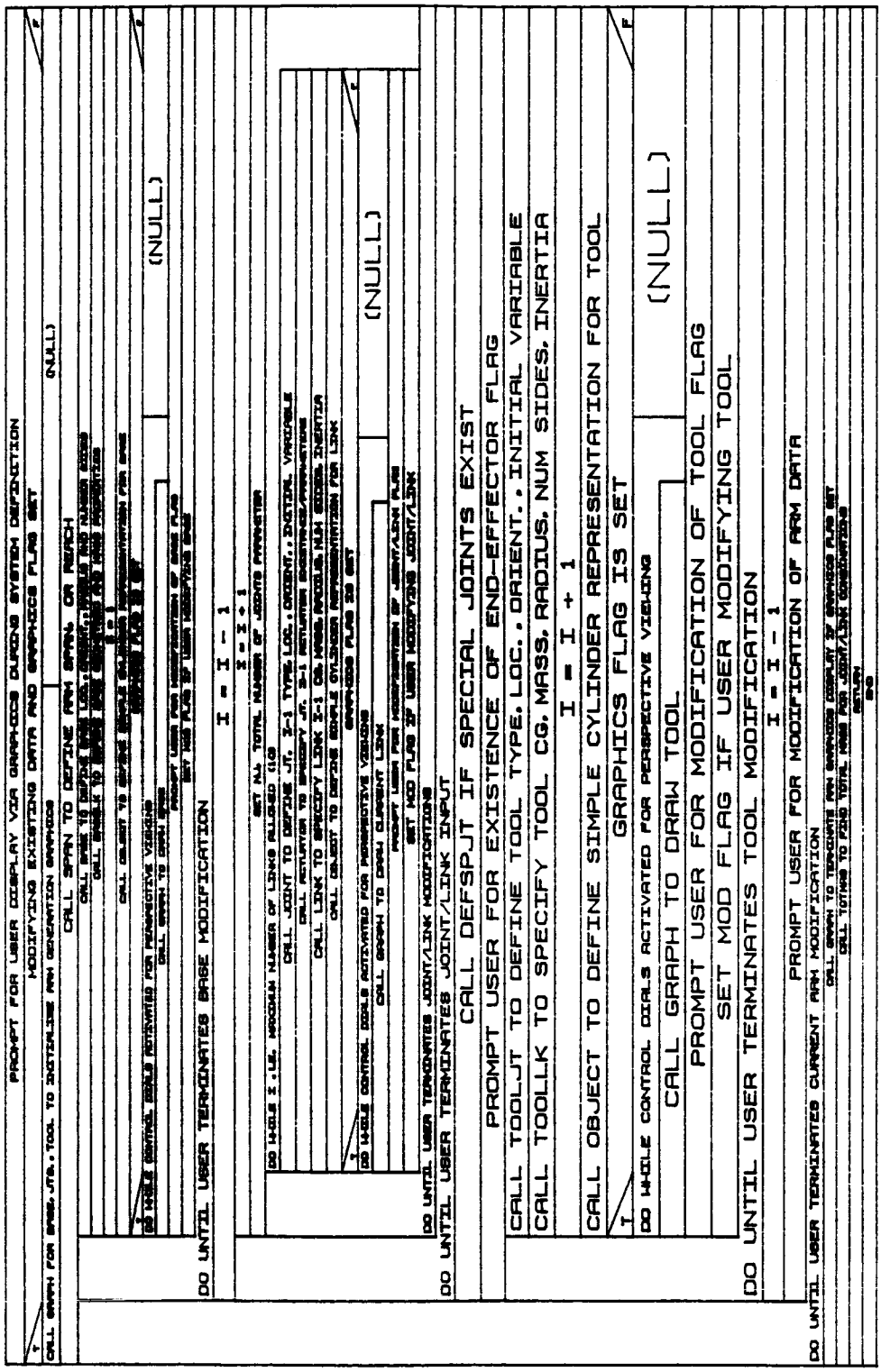
Subroutine DEFUNIT is called during system definition to set up input and output units specified by the user. If these I/O units are not metric, the routine establishes conversion factors between the I/O units and internal (metric) units. The conversion factors are stored in variable CONUNIT, and LISUNIT contains a character string listing the I/O units employed.



1.2.3 CREATARM

Subroutine CREATARM is called within the system definition function to provide control of the creation or modification modes for the simple cylinder arm data file. The basic routines called for either option are SPAN (define arm span), BASE (define base geometric properties), BASELK (define base mass properties), JOINT (define joint), ACTUATOR (optional, to define motor properties), LINK (define link properties), and DEFSPJT (optional, to define special joints). If the user opts for an end-effector, TOOLJT (define tool-joint properties) and TOOLLK (define tool-link properties) are called. Graphics may be requested during CREATARM.

ORIGINAL PAGE IS
OF POOR QUALITY



1.2.4 RDARM

Subroutine RDARM is called from BLDARM to read from an unformatted arm data file the data describing a single arm. Routines called by BLDARM can then modify the data. The user is prompted for the name of the arm data file to be modified. The following data are read from it:

1) Input/output units;

2) Geometric properties;

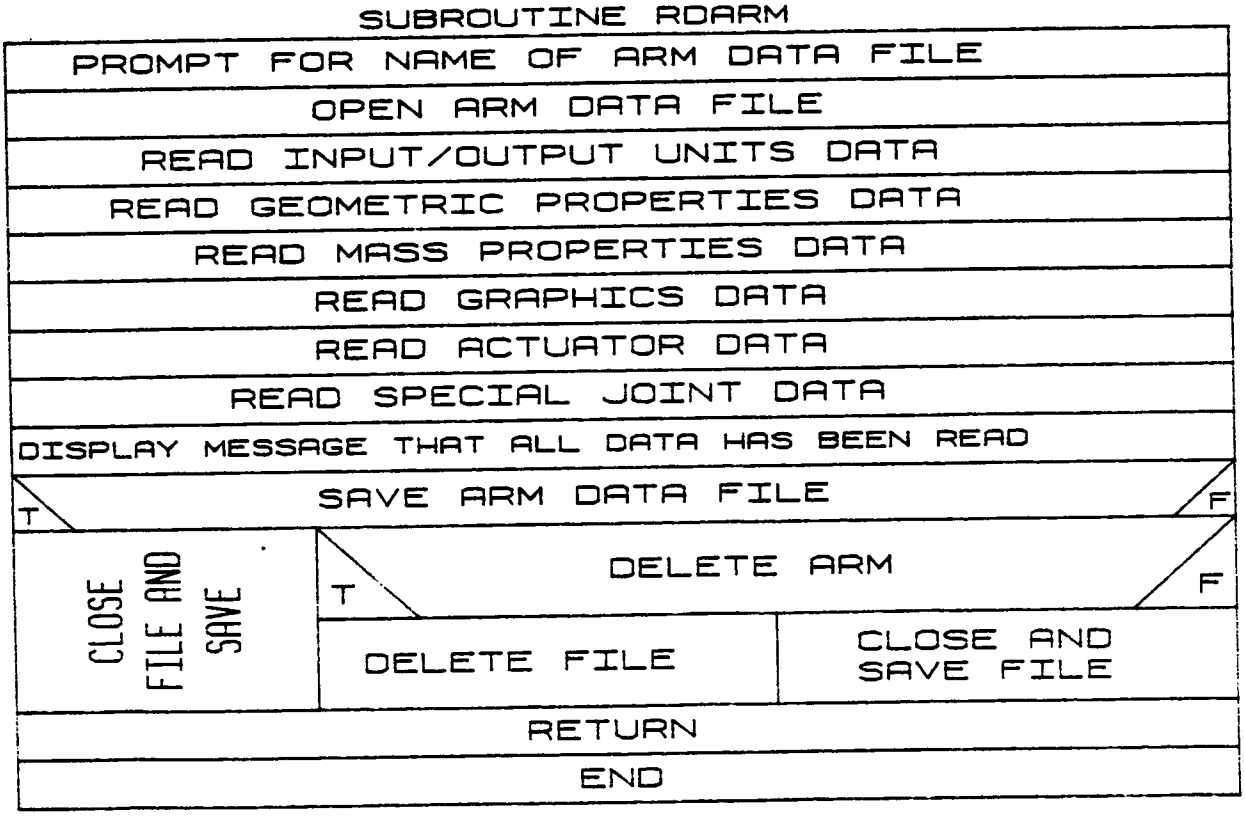
3) Mass properties;

4) Graphics data;

5) Actuator data;

6) Information on special joints.

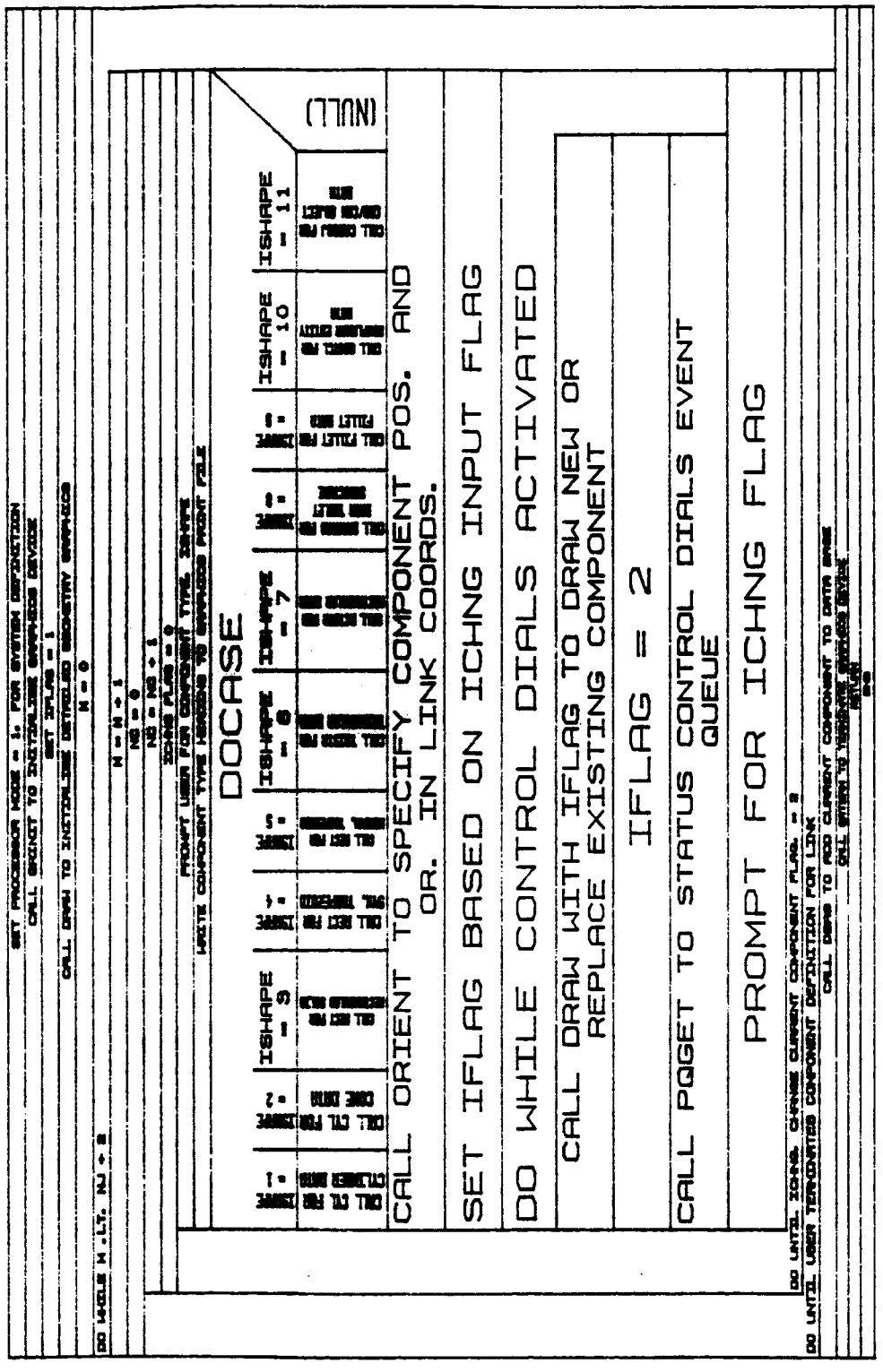
The user has the option of saving or deleting the old data file.



1.2.5 BLDDAT

Subroutine BLDDAT provides the user the capability to specify a more detailed physical representation for the links of the robotic arm. Components of the robotic arm system are defined by combinations of geometric primitives. A number of detailed components can be included for the base, each link extension and the tool definitions. The components are simple three-dimensional shapes: the cylinders, cones, rectangular solids, symmetric trapezoids, nonsymmetric trapezoids, triangular cross-sectional beams, rectangular beams, data tablet structures, fillet components, nonplanar entities, and CAD/CAM objects. Unique subroutines are called to handle loading the graphics object data for the shapes chosen to represent a detailed arm. Additional shapes can be added as required. The component type is written to the detailed graphics save file if requested.

ORIGINAL PAGE IS
OF POOR QUALITY



1.2.6 WRTARM

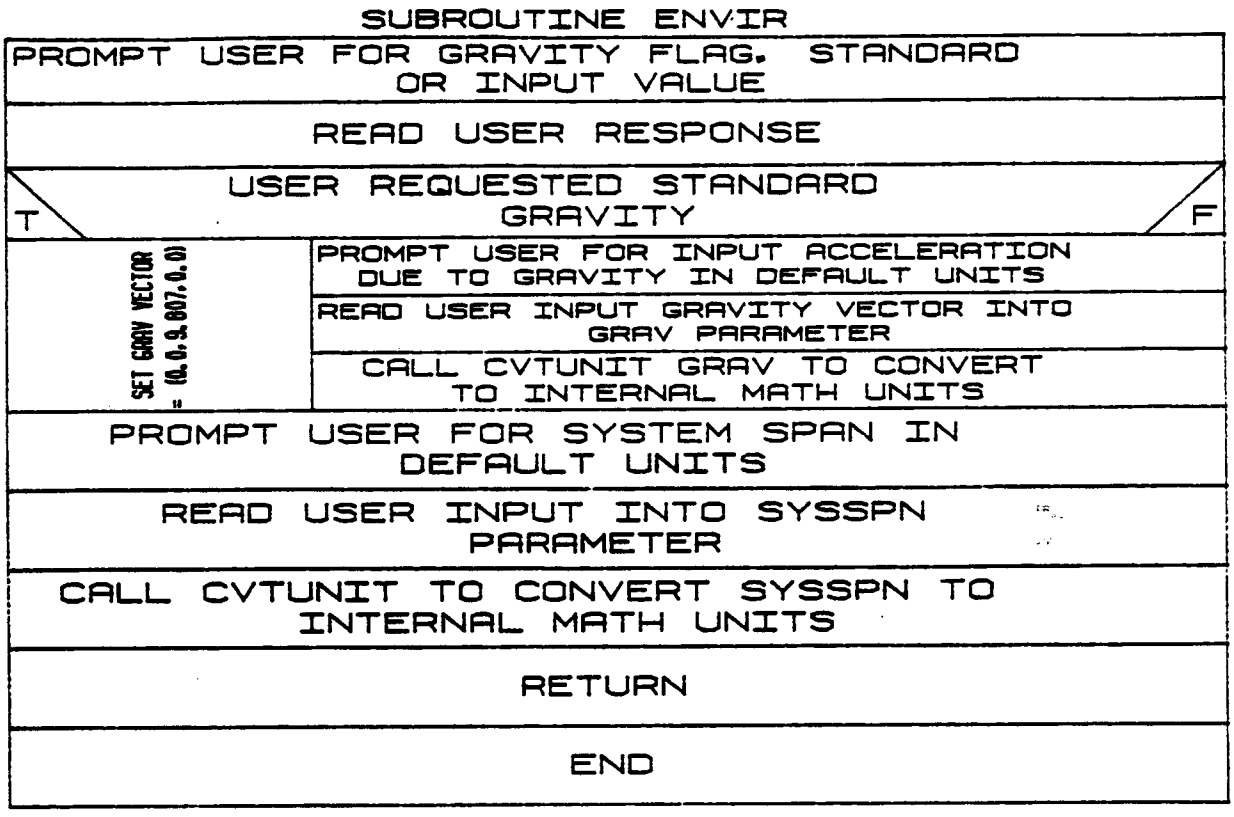
Subroutine WRTARM is called from BLDARM to save, in a user-specified file, the data generated when creating or modifying an arm description. The user is prompted for the name of this file and is also given the option of storing a formatted file containing the arm description for later printing.

SUBROUTINE WRTARM

PROMPT FOR NAME OF FILE TO WRITE ARM DATA TO	
OPEN FILE	
T: WRITE FORMATTED ARM GEOMETRY FILE FOR PRINTING	F: (NULL)
PROMPT FOR NAME OF ARM GEOMETRY PRINTOUT FILE	
OPEN FILE	
WRITE ARM GEOMETRY DESCRIPTION TO FILE	
CLOSE AND SAVE FILE	
WRITE INPUT/OUTPUT UNITS DATA TO FILE	
WRITE ARM GEOMETRY DATA	
WRITE ARM MASS PROPERTIES	
WRITE ARM GRAPHICS DATA	
WRITE ACTUATOR DATA	
WRITE SPECIAL JOINT DATA	
DISPLAY MESSAGE THAT FILE WAS WRITTEN	
CLOSE AND SAVE FILE	
RETURN	
END	

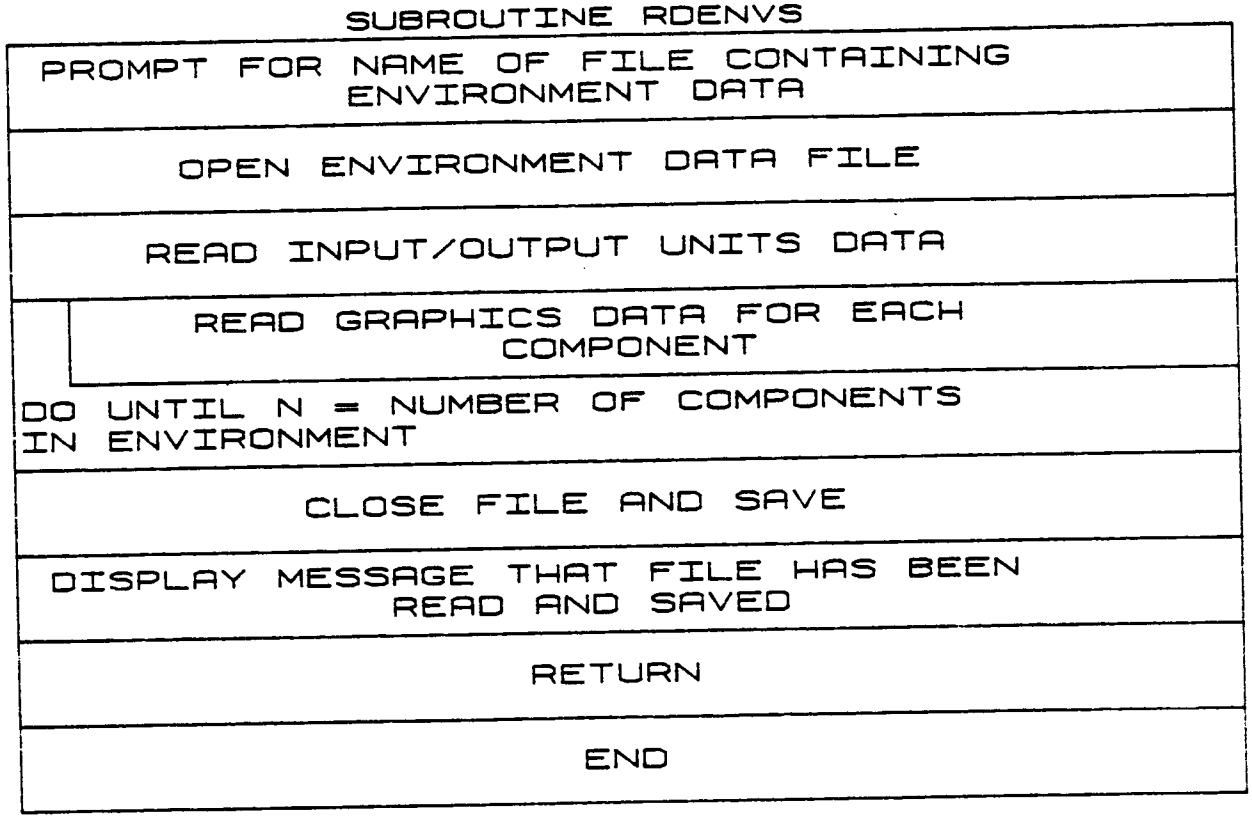
1.2.7 ENVIR

Subroutine ENVIR interactively establishes the basic properties of the system environment during system definition. This includes the input/output units, the gravity vector and the system span.



1.2.8 RDENVS

The subroutine RDENVS is called from BLDSYS if the user wishes to include an environment in the system being created. This routine reads the unformatted environment data file created by the system definition function for the multiarm system. The user is prompted for the name of the data file under which the environment data have been stored. The file is opened and COMMON block ENVTBK loaded from the data file during system creation. The file is closed and saved.



ORIGINAL PAGE IS
OF POOR QUALITY

1.2.9 RDARMS

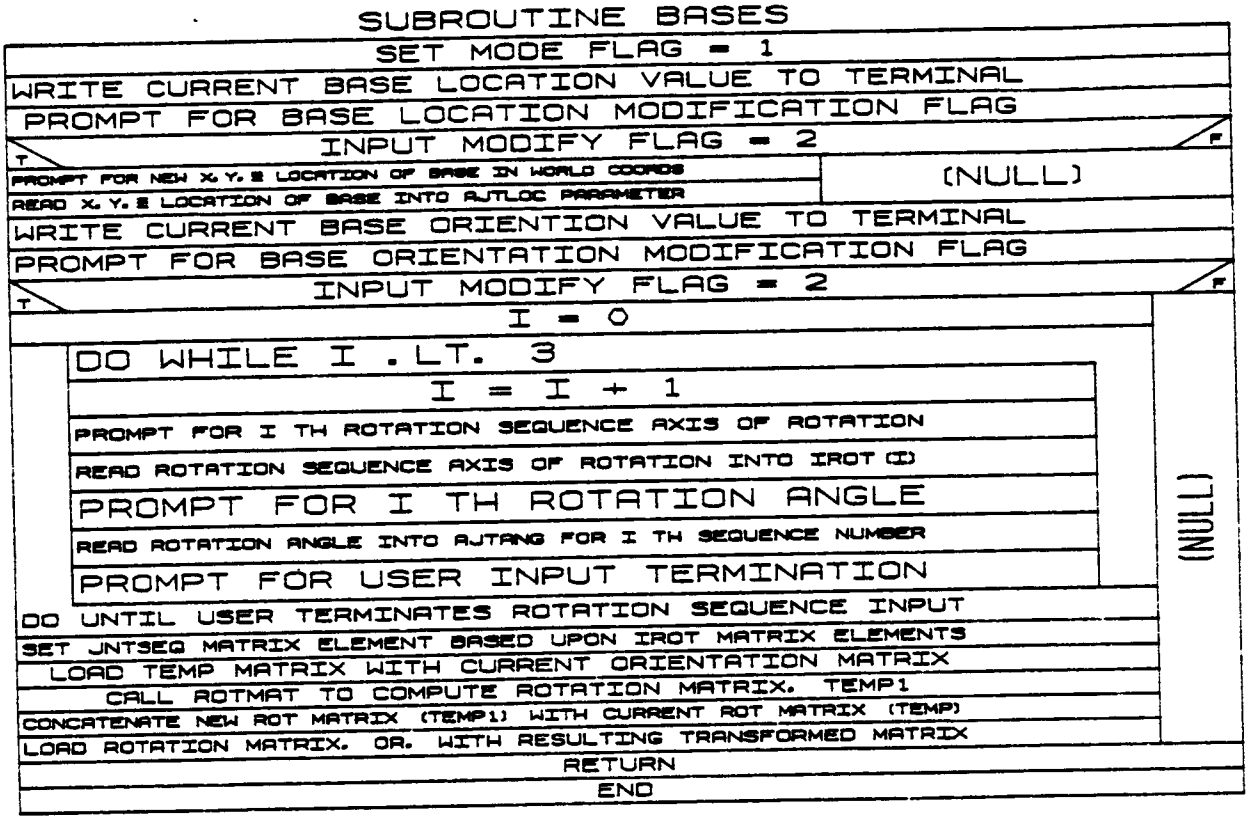
Routine RDARMS is called during the total robotic system creation for each of the arms desired for inclusion in the system setup. The subroutine RDARMS reads the unformatted data file created by the system definition function containing any one arm file. The user is prompted for the name of the data file under which the arm data have been stored. The file is opened and read into the following COMMON blocks: GEOMBK, AMASBK, IOBJBK, TOOLBK, FORCBK, MOTORBK and SPJTBK.

SUBROUTINE RDARMS

SET PROCESSSOR MODE = 1. FOR SYSTEM DEFINITION
PROMPT USER FOR FILENAME OF SINGLE ARM FILE TO READ
OPEN ARM DATA FILE
STORE SYSTEM UNITS IN TEMPORARY ARRAYS
READ UNITS COMMON BLOCK INTO SYSTEM COMMON
READ GEOMETRY COMMON BLOCK FOR BASE, JTS, AND TOOL INTO SYSTEM COMMON
READ MASS PROPERTIES COMMON BLOCK FOR BASE, JTS, AND TOOL INTO SYSTEM COMMON
CONVERT PROPERTIES TO INTERNAL UNITS FOR SLIDING AND ROTATING JTS.
PUT TOOL MASS PROPERTIES INTO SYSTEM TOOL COMMON BLOCK
READ GRAPHICS DATA COMMON BLOCK INTO SYSTEM COMMON
CONVERT ARM SPAN TO INTERNAL UNITS
SCALE GRAPHICS OBJECT DATA BY (ARM SPAN / SYSTEM SPAN)
READ ACTUATOR DATA COMMON BLOCK INTO SYSTEM COMMON
CONVERT ACTUATOR DATA TO INTERNAL UNITS
READ SPECIAL JOINT DATA COMMON BLOCK INTO SYSTEM COMMON
CLOSE SINGLE ARM DATA FILE
REWRITE SYSTEM UNITS INTO SYSTEM UNITS COMMON BLOCK
RETURN
END

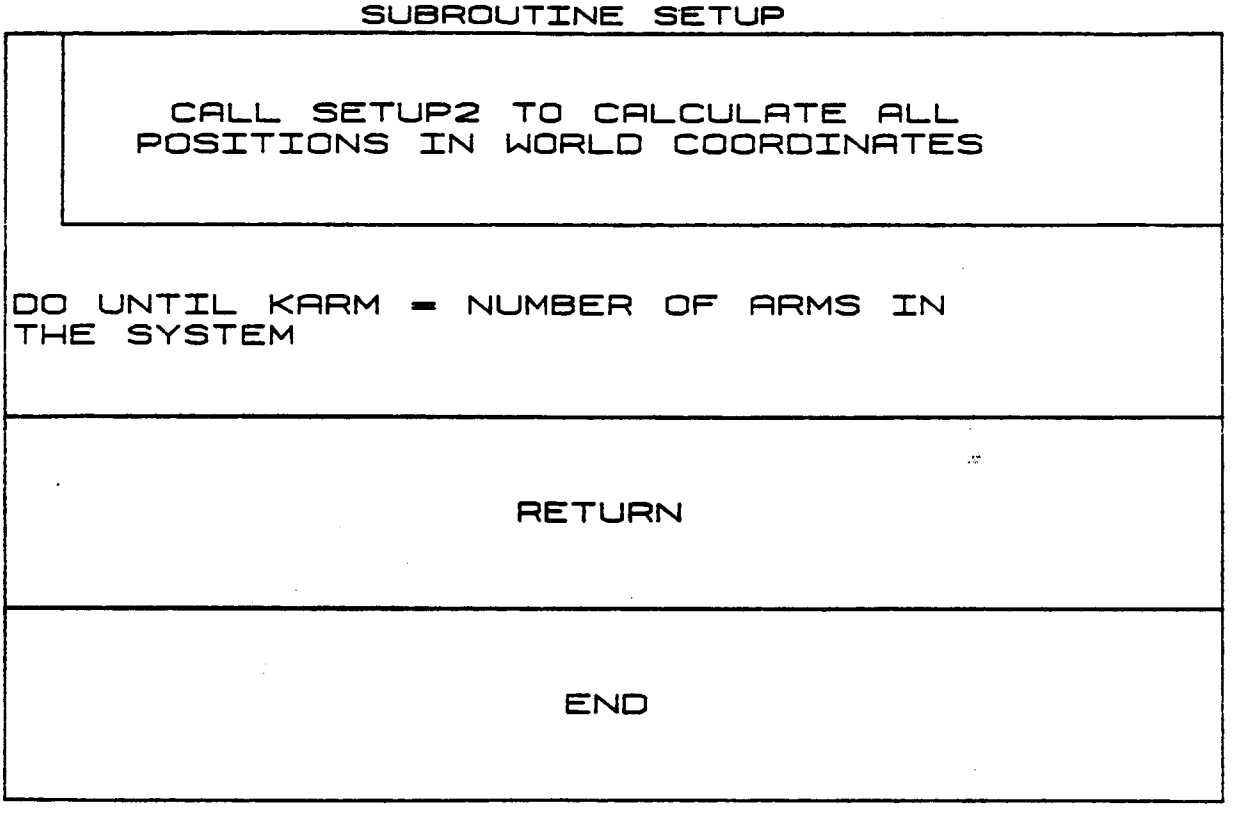
1.2.10 BASES

BASES modifies the base location or orientation when including an arm in a system; it is called from BLDSYS.



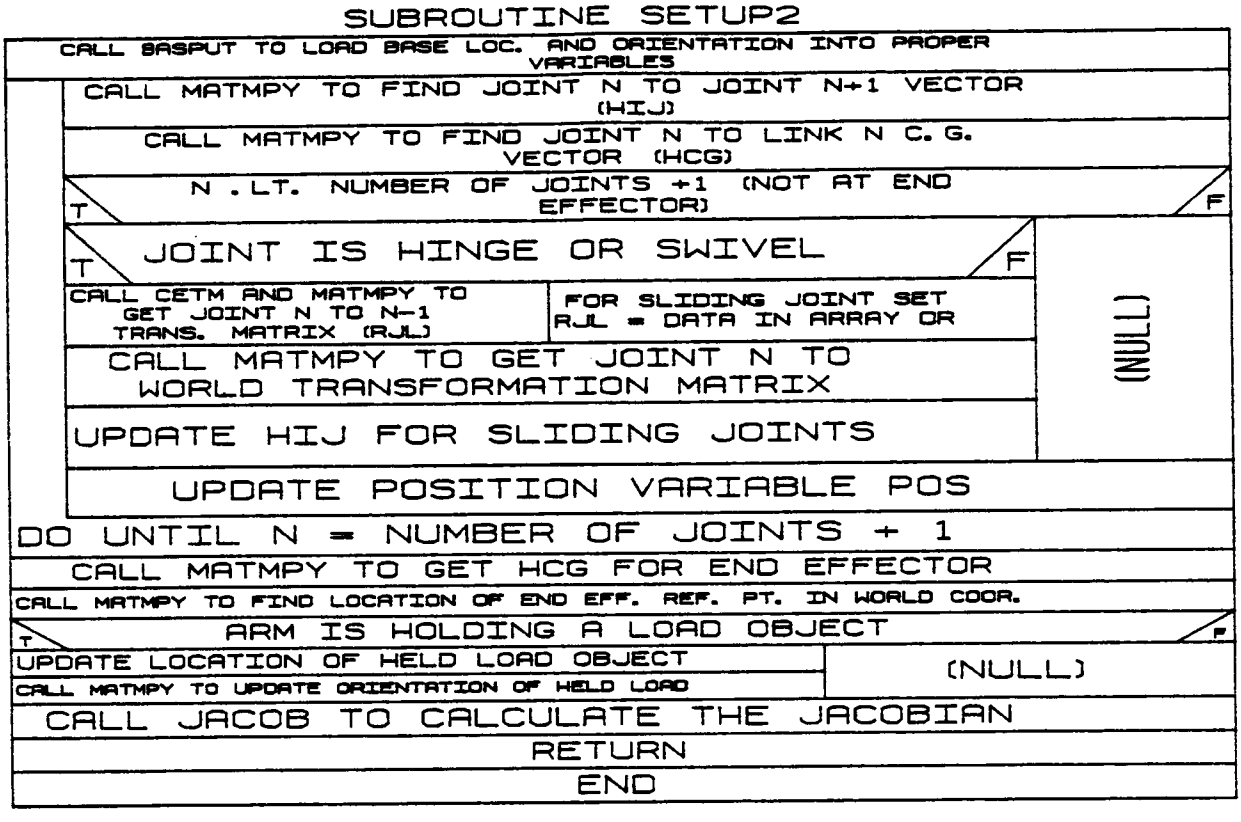
1.2.11 SETUP

Subroutine SETUP calls SETUP2 for each arm in the manipulator system to calculate the positions of all arm components in terms of world coordinates.



1.2.12 SETUP2

SETUP2 works every increment. It calculates the positions of all links (including base, tool, and any held loads) and transforms link and centroid vectors to world coordinates. The recursive positioning method described in the main text is used. Finally, subroutine JACOB is called to compute the Jacobian for the current position.



1.2.13 HARD_COP

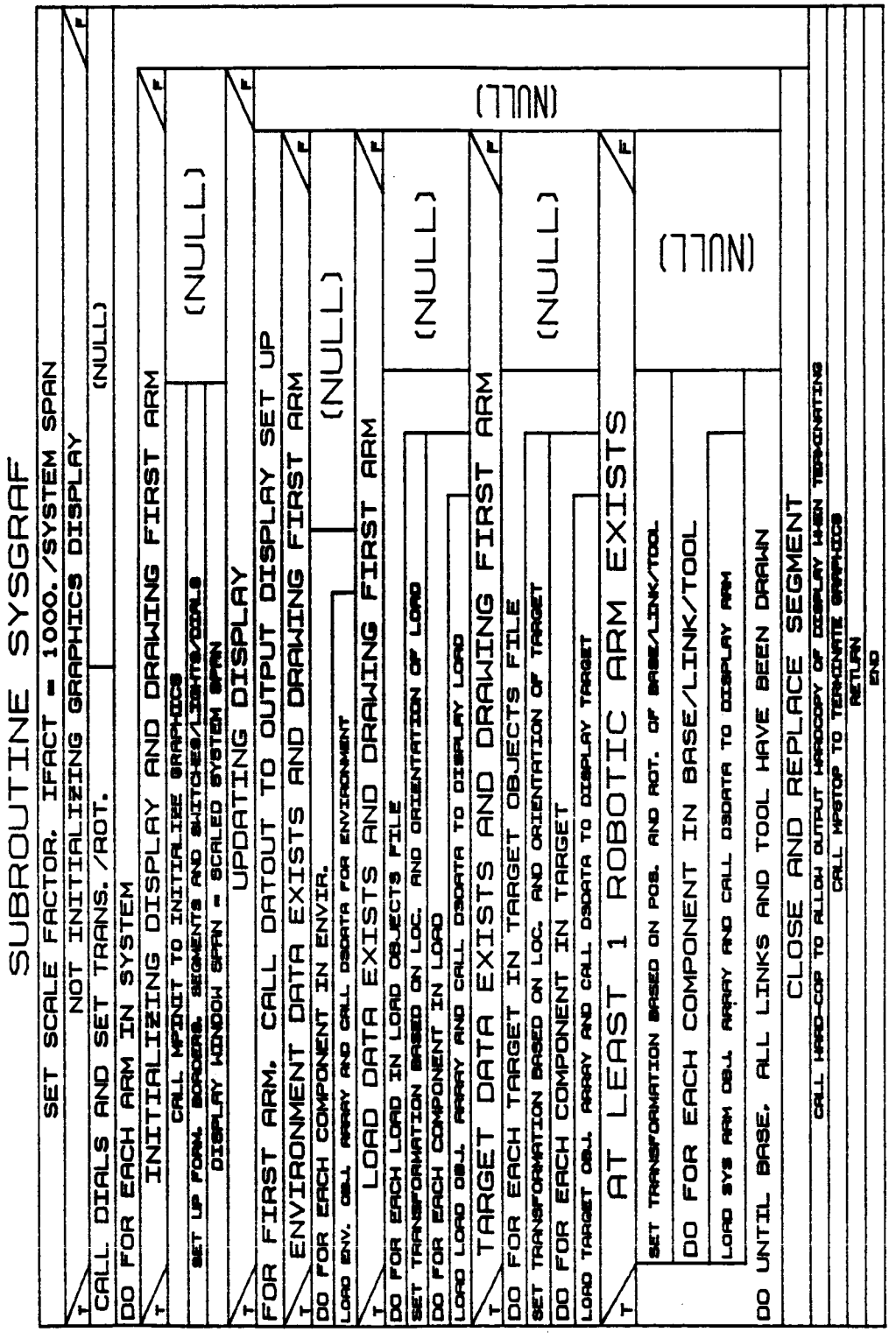
Subroutine HARD_COP is executed when a hardcopy record of the current Evans and Sutherland display may be desired. This routine queries the user to determine if a hardcopy is desired and runs the appropriate routines to create a picture file for later translation into a hardcopy plot.

(VCLR for HARD_COP is not available.)

1.2.14 SYSGRAF

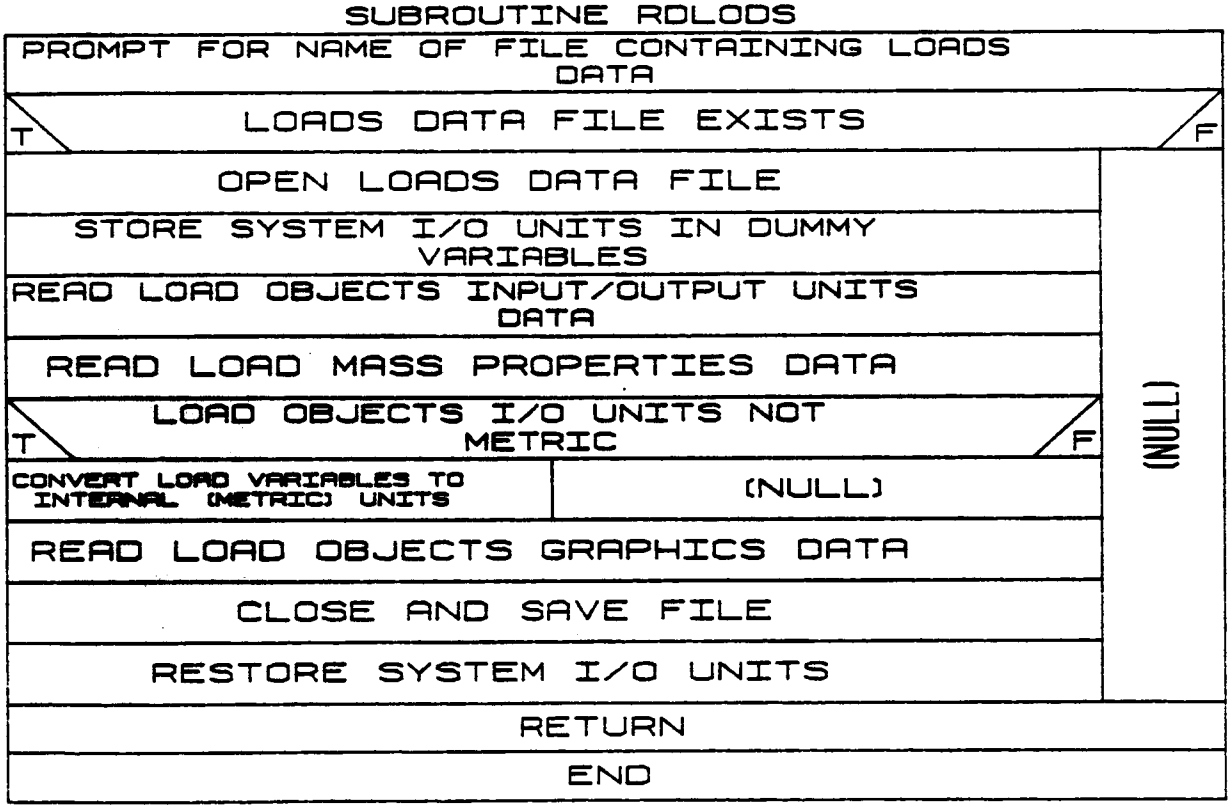
Subroutine SYSGRAF provides the system definition graphics capability in the system definition function. SYSGRAF displays the environment, target, load and robotic arm choices for building a robotic system scenario. It takes as input through the calling sequence, the number of arms in the system, a flag indicating the existence of an environment file for the system, a target file, and a load objects file inclusion indicator. It uses the system span input by the user to scale the graphics picture. IFLAG controls the logical flow in the subroutine. If IFLAG=1, the graphics system is initialized and displayed in the initial condition; if IFLAG =2, the robotic system, targets, loads and environment are displayed; if IFLAG=3, the graphics display is terminated. In the update mode, the environment is constant and therefore not updated. As before, the Evans and Sutherland graphics routines are used to provide all graphic capabilities.

ORIGINAL PAGE IS
OF POOR QUALITY



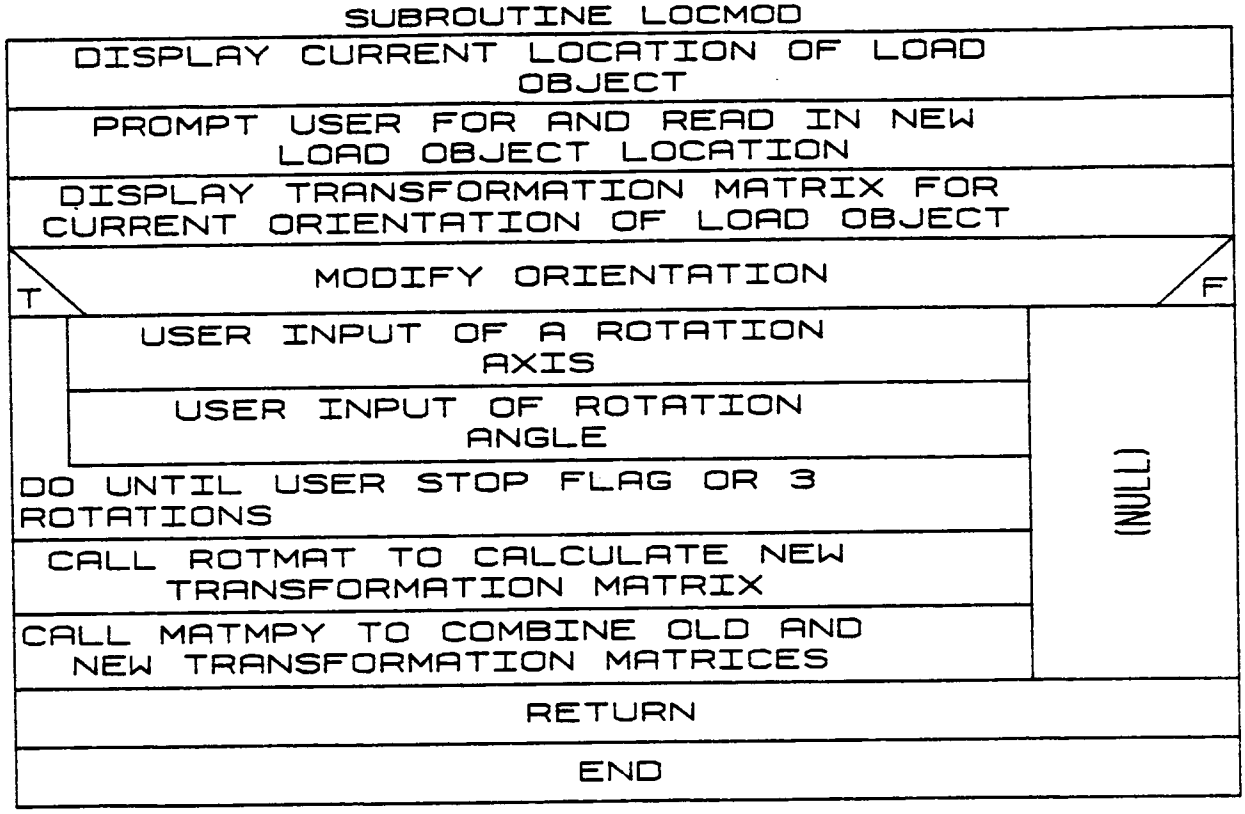
1.2.15 RDLODS

RDLODS reads in load data if the user requests that loads be included in the robotic system under construction. The file read is the unformatted load file created by the system definition function. Subroutine RDLODS prompts the user for the name of the file containing load data, and then reads those data into COMMON blocks LGRAFBK and LOADBK during system creation. The load file is scaled from load units to internal system units, closed, and saved.



1.2.16 LOCMOD

Subroutine LOCMOD is called from BLDSYS to allow the user to modify the locations and orientations of load objects when building a system. The current location is displayed and then the user is prompted for a new location. The subroutine also displays the transformation matrix for the current orientation of the load object and prompts the user for a sequence of rotation axes and angles that define a change in orientation. ROTMAT is called to calculate the transformation matrix from the user input, and MATMPY combines this new transformation matrix with the old one.



1.2.17 WRTSYS

Subroutine WRTSYS is called from BLDSYS to write an unformatted file containing the data needed to describe a system. These date include user units, moving base flags, arm geometry, mass and actuator properties, special joint data, arm graphics, environment, load object and target properties, tool data, and load object and target graphics information.

SUBROUTINE WRTSYS

SET PROCESSSOR MODE = 1. FOR SYSTEM DEFINITION

PROMPT USER FOR FILENAME OF SYSTEM DATA FILE TO WRITE

OPEN SYSTEM DATA FILE

WRITE UNITS INTO SYSTEM COMMON

WRITE MOVING BASE FLAG IMOVBAS INTO SYSTEM COMMON

SYSTEM INCLUDE MOVING BASE

T	F
WRITE NUMBER OF MOVING BASES, NBAS INTO SYSTEM COMMON	(NULL)

KARM, ARM COUNTER, = 0

DO WHILE KARM .LT. NARM (TOTAL NUMBER ARMS)

SYSTEM INCLUDE MOVING BASE

T	F
WRITE BASE INFO, KBASE(1,KARM) INTO SYSTEM COMMON	(NULL)

KARM = KARM + 1

WRITE GEOMETRY PROPERTIES PARAMETERS FOR KARM INTO SYSTEM COMMON

WRITE MASS PROPERTIES PARAMETERS FOR KARM INTO SYSTEM COMMON

WRITE ACTUATOR DATA PARAMETERS FOR KARM INTO SYSTEM COMMON

WRITE SPECIAL JOINT DATA PARAMETERS FOR KARM INTO SYSTEM COMMON

WRITE GRAPHICS DATA FOR EACH ARM INTO SYSTEM COMMON

WRITE ENVIRONMENT DATA PARAMETERS INTO SYSTEM COMMON

WRITE LOAD OBJECTS DATA PARAMETERS FOR EACH LOAD INTO SYSTEM COMMON

WRITE TARGET DATA PARAMETERS FOR EACH TARGET INTO SYSTEM COMMON

WRITE TOOL DATA PARAMETERS FOR EACH ARM INTO SYSTEM COMMON

WRITE LOAD OBJECTS GRAPHICS DATA FOR EACH LOAD INTO SYSTEM COMMON

WRITE TARGET GRAPHICS DATA FOR EACH TARGET INTO SYSTEM COMMON

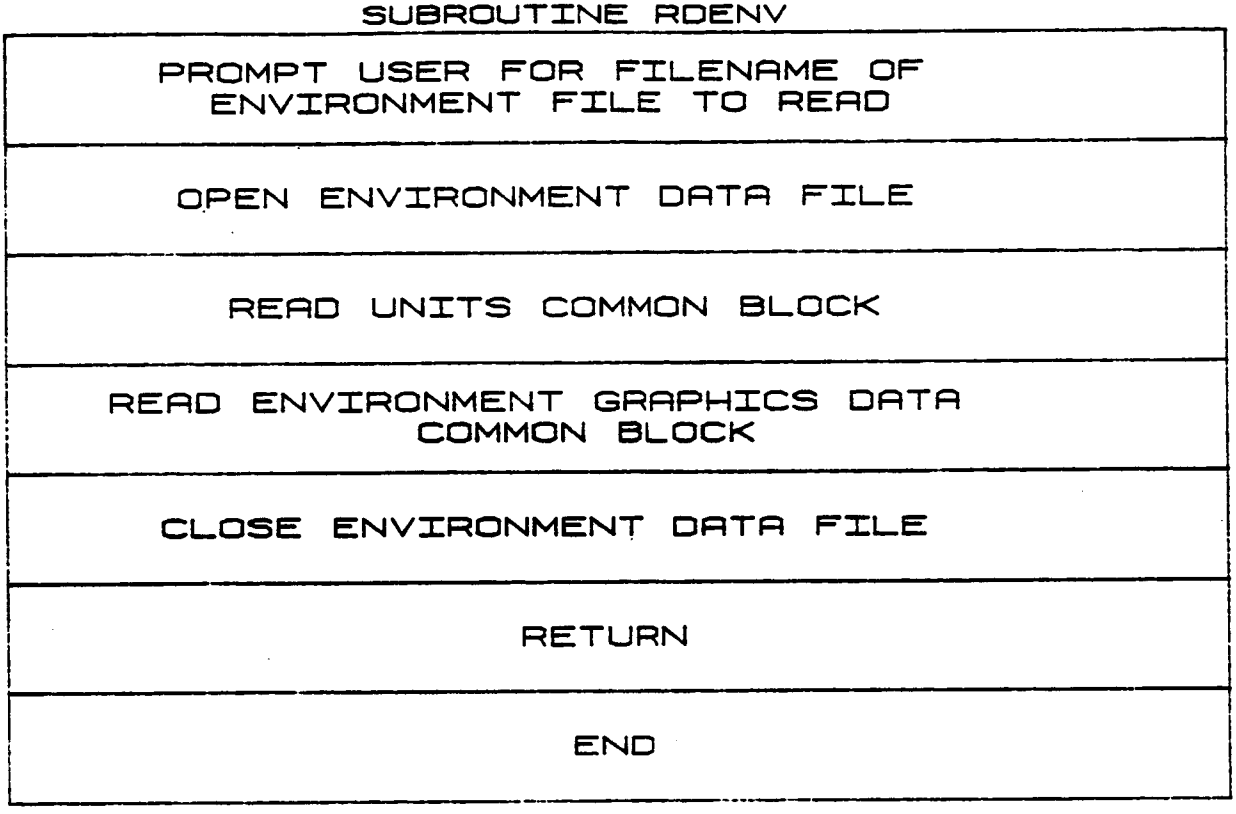
CLOSE SYSTEM DATA FILE

RETURN

END

1.2.18 RDENV

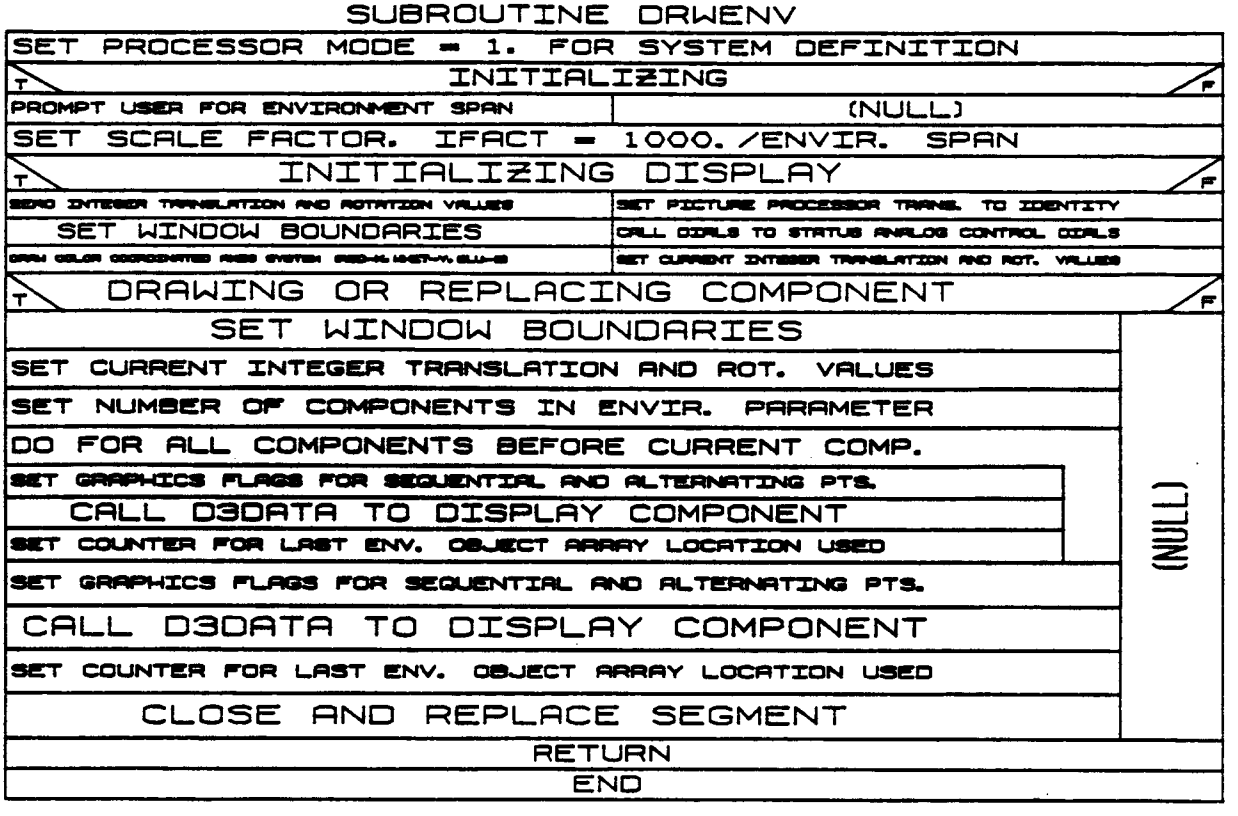
The subroutine RDENV reads an unformatted environment data file during the system definition function. The content of the file is the pertinent COMMON block defining an environment for the robotic system. The user is prompted for the file name from which the file is to be read.



ORIGINAL PAGE IS
OF POOR QUALITY

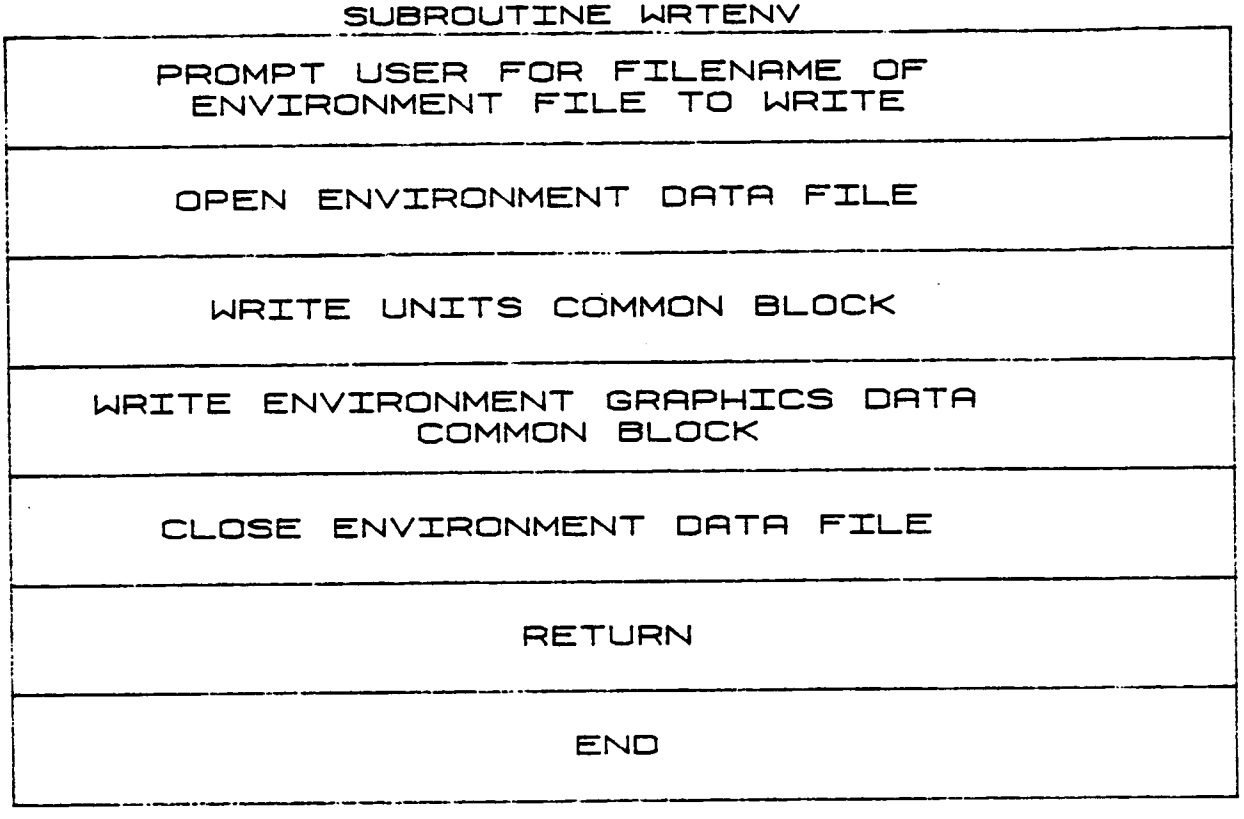
1.2.19 DRWENV

DRWENV is called within the system definition function from BLDENV to provide graphics display during the generation of detailed environment graphics representations. It is called to display each successive environment component as it is defined.



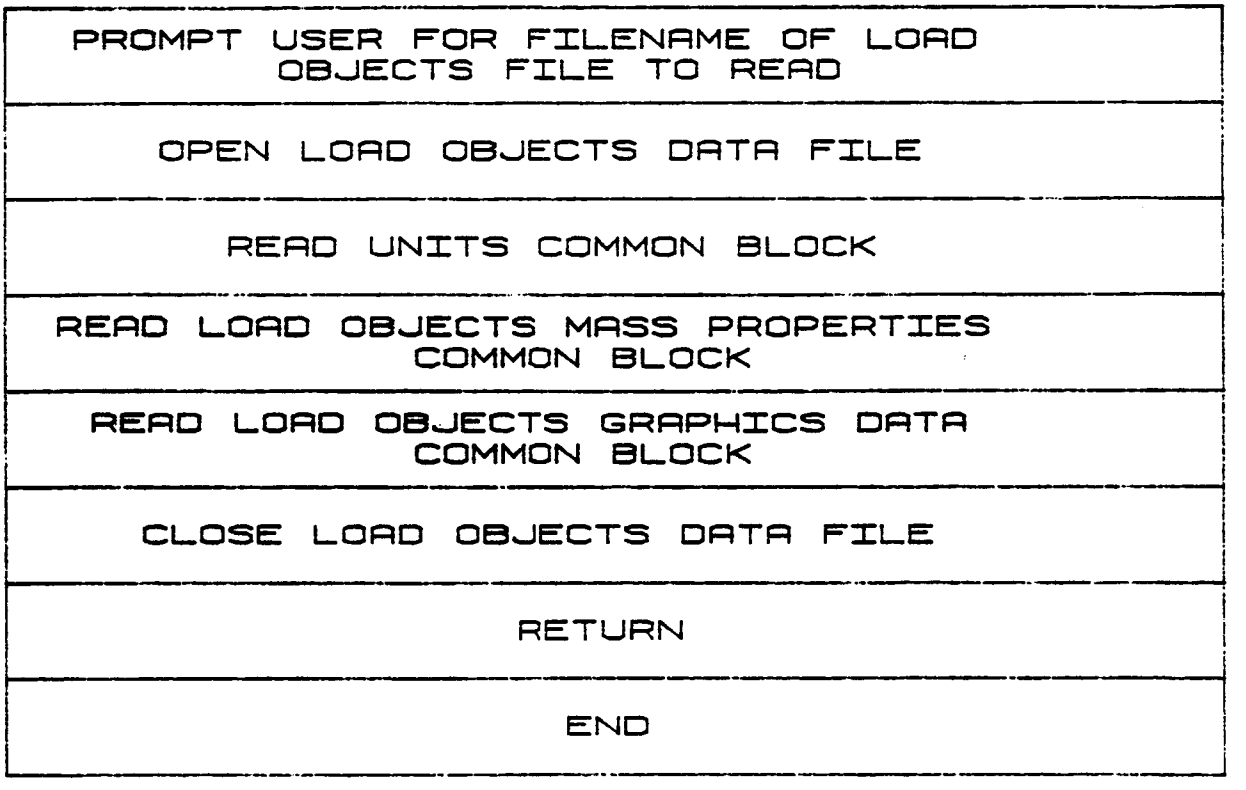
1.2.20 WRTENV

Subroutine WRTENV writes an unformatted environment data file during the system definition function. The content of the file is the pertinent COMMON block defining an environment.



1.2.21 RDLOAD

The subroutine RDLOAD reads an unformatted load objects data file during the system definition function. The contents of the file are the pertinent COMMON blocks defining a load file for the robotic system. The user is prompted for the file name from which the file is to be read.



1.2.22 LOAD

Subroutine LOAD is called during the BLDLOD option of INITDRVR. It allows the user to create and define the mass properties of one or more load objects. If a file of load object data already exists, this subroutine may be used to modify portions of those data. The load parameters for which the user is prompted are listed:

1) Load object name (up to 8 characters);

2) Location and orientation with respect to the world coordinate system;

3) Length and radius;

4) Center of mass;

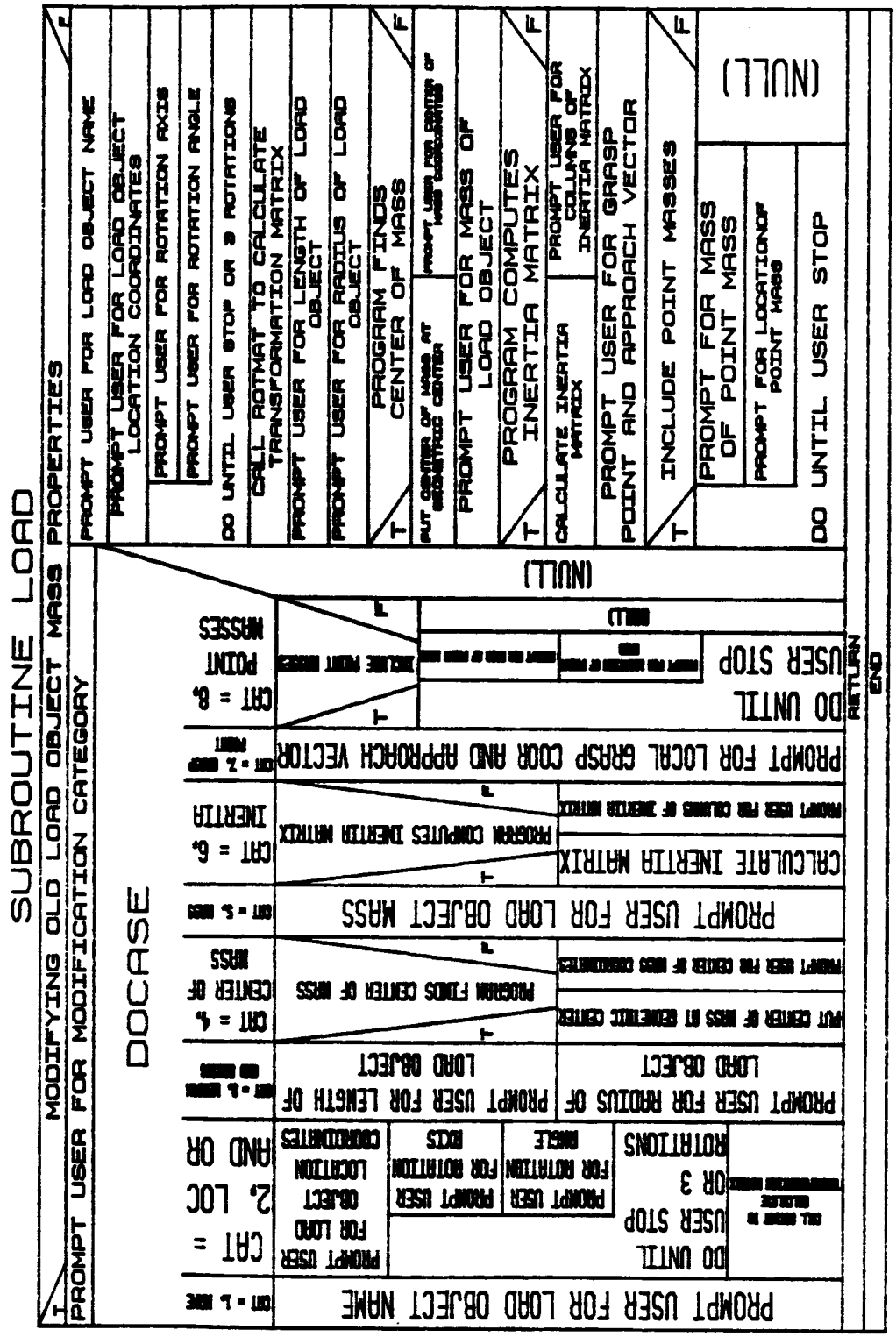
5) Mass;

6) Inertia distribution;

7) Grasp point and approach vector in load local coordinates;

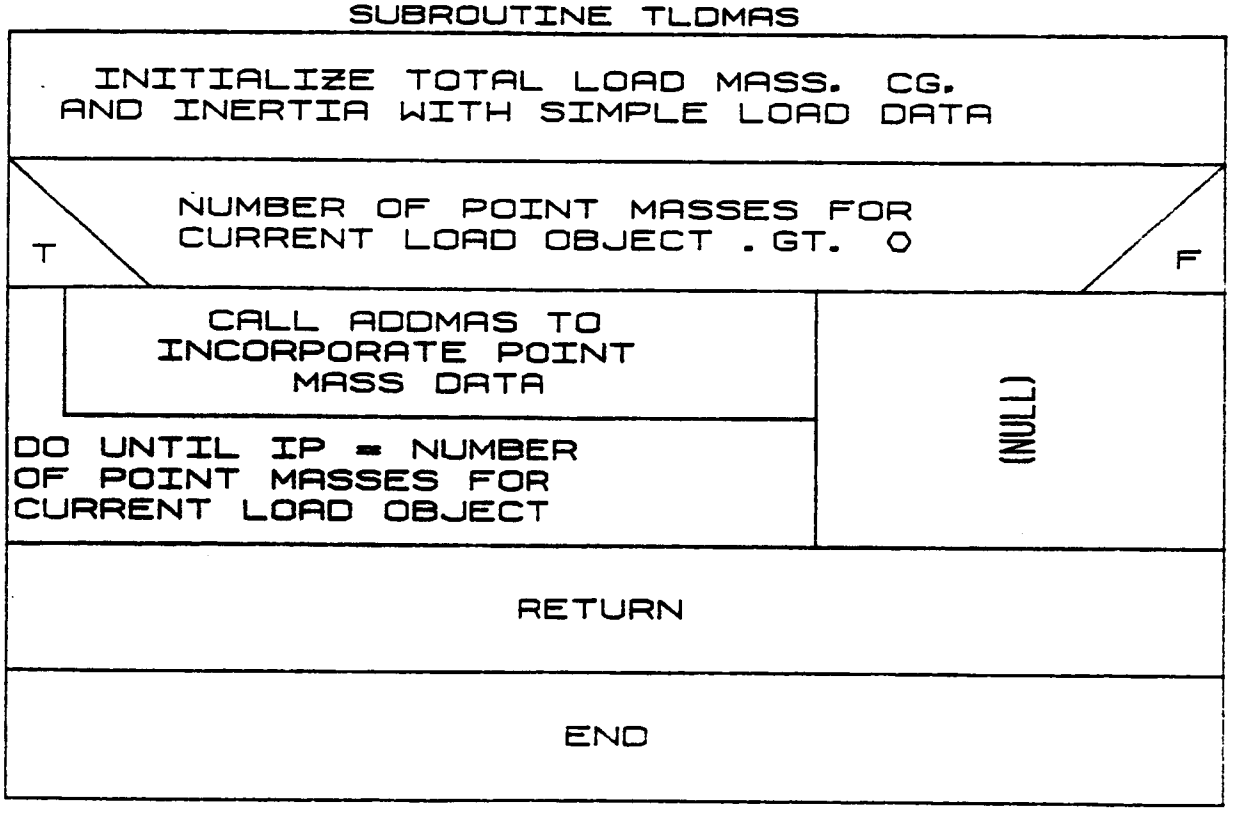
8) Mass and location of any point masses included.

ORIGINAL PAGE IS
OF POOR QUALITY



1.2.23 TLDMAS

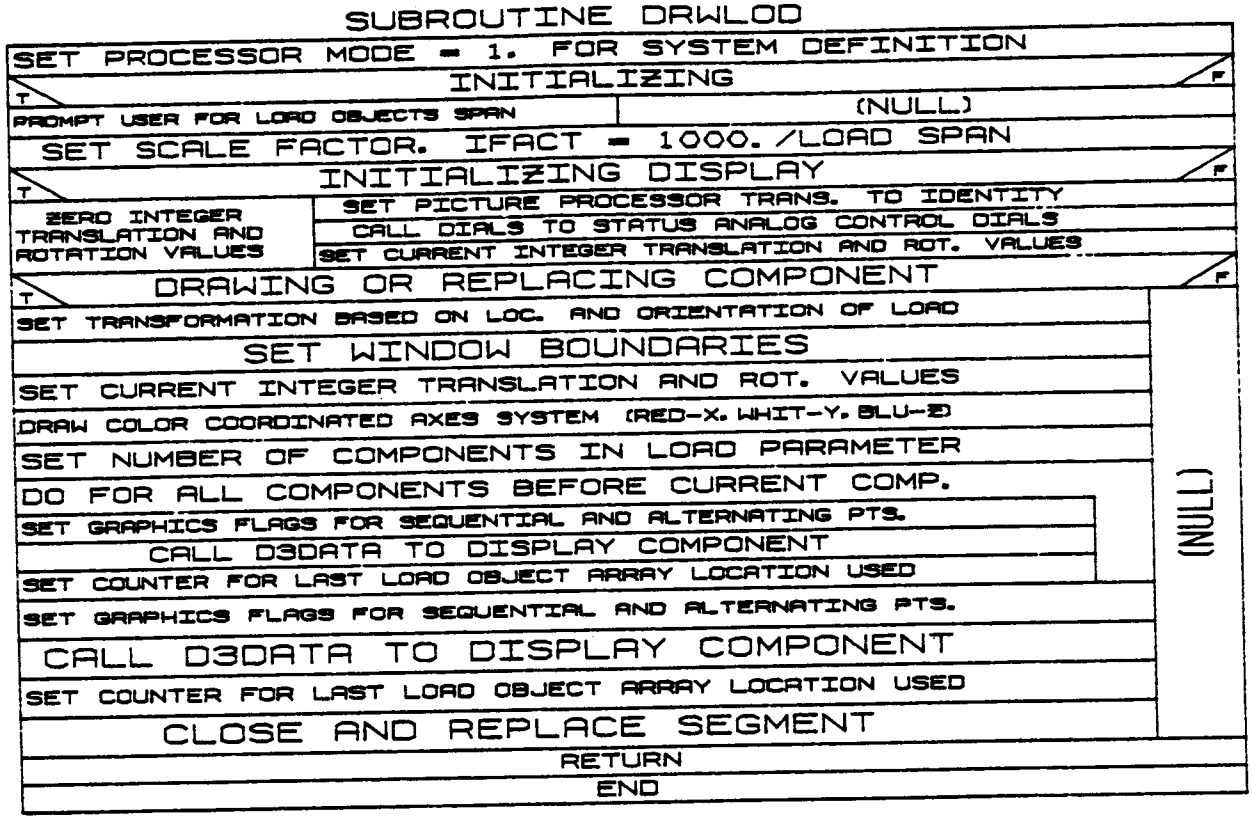
Subroutine TLDMAS is called from BLDLOD to add the effects of point masses included in a load object. Variables for total mass, centroid location, and rotary inertia are initialized with the values for the simple load object. If point masses are included, ADDMAS is called to calculate new values for these variables that include the point mass effects.



ORIGINAL PAGE IS
OF POOR QUALITY

1.2.24 DRWLOD

DRWLOD is called within the system definition function from BLDLOD to provide graphics display during the generation of a detailed load objects file. It is called to display each successive load object component as it is defined.



1.2.25 WRTLOD

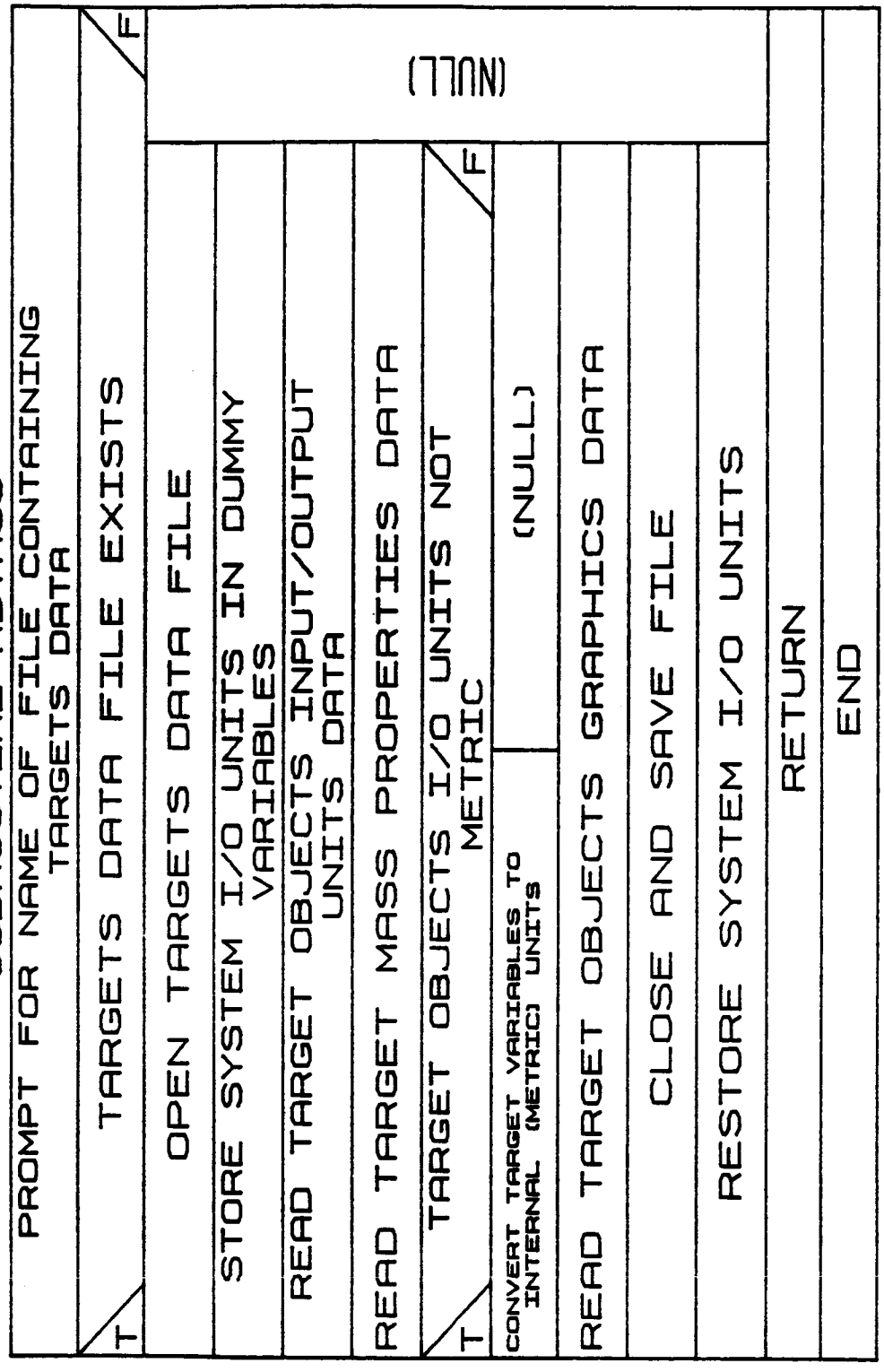
Subroutine WRTLOD writes the unformatted load objects data file during the system definition function. The contents of the file are the pertinent COMMON blocks defining the mass properties and graphics of the loads for a system.

SUBROUTINE WRTLOD

PROMPT USER FOR FILENAME OF LOAD OBJECTS FILE TO WRITE
OPEN LOAD OBJECTS DATA FILE
WRITE UNITS COMMON BLOCK
WRITE LOAD OBJECTS MASS PROPERTIES COMMON BLOCK
WRITE LOAD OBJECTS GRAPHICS DATA COMMON BLOCK
CLOSE LOAD OBJECTS DATA FILE
RETURN
END

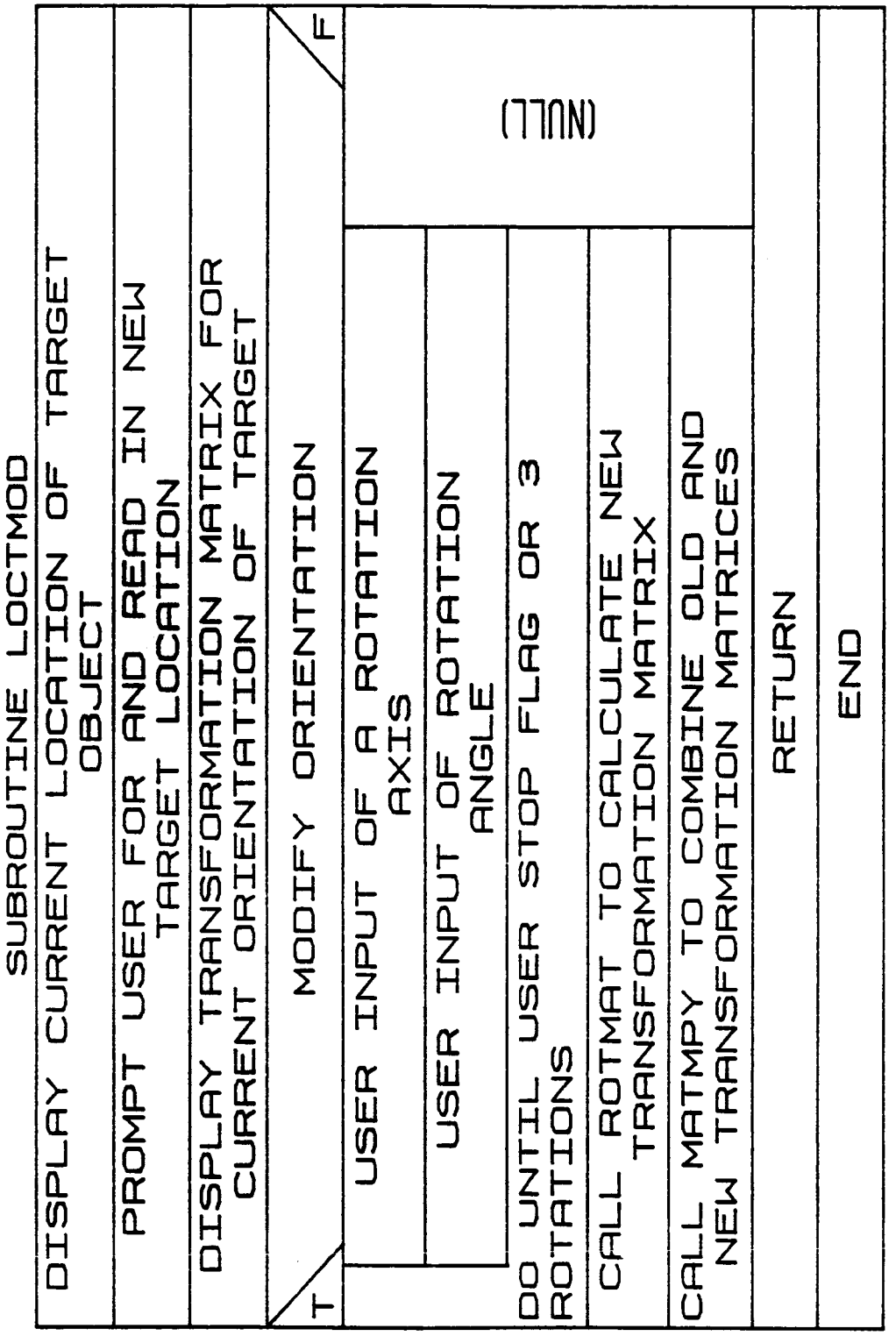
1.2.26 RDTRGS

RDTRGS reads in target data if the user requests that targets be included in the robotic system under construction. The file read is the unformatted target file created by the system definition function. Subroutine RDTRGS prompts the user for the name of the file containing target data, and then reads those data into common blocks TGRAFBK and TARGBK during system creation. The target file is scaled from target units to internal system units, closed, and saved.


SUBROUTINE RDTRGS
PROMPT FOR NAME OF FILE CONTAINING TARGETS DATA
TARGETS DATA FILE EXISTS
T
F
(NULL)
OPEN TARGETS DATA FILE
STORE SYSTEM I/O UNITS IN DUMMY VARIABLES
READ TARGET OBJECTS INPUT/OUTPUT UNITS DATA
READ TARGET MASS PROPERTIES DATA
TARGET OBJECTS I/O UNITS NOT METRIC
T
F
CONVERT TARGET VARIABLES TO INTERNAL (METRIC) UNITS
(NULL)
READ TARGET OBJECTS GRAPHICS DATA
CLOSE AND SAVE FILE
RESTORE SYSTEM I/O UNITS
RETURN
END

1.2.27 LOCTMOD

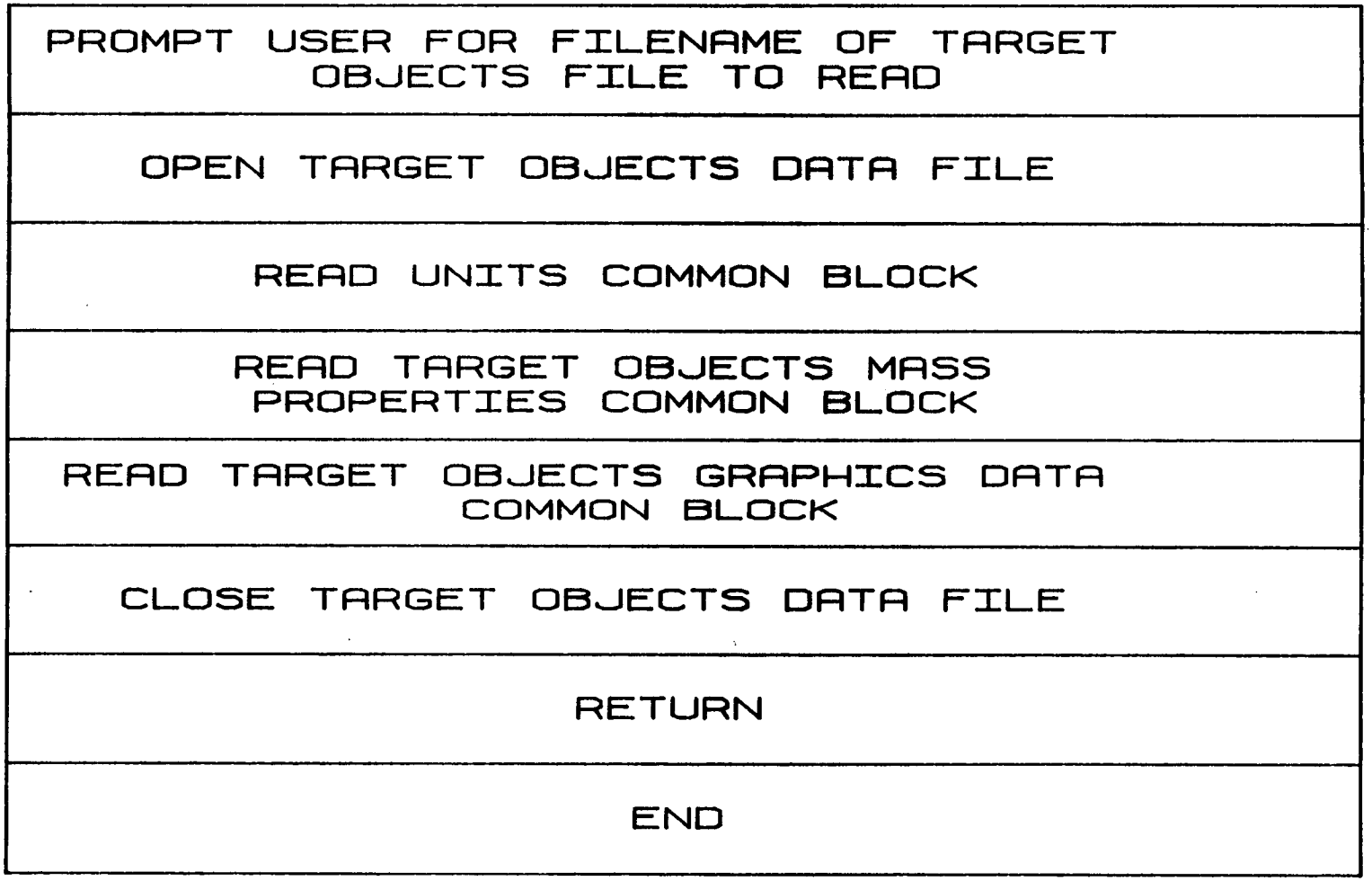
Subroutine LOCTMOD is called from BLDSYS to allow the user to modify the locations and orientations of target objects when building a system. The current location is displayed, and then the user is prompted for a new location. The subroutine also displays the transformation matrix for the current orientation of the target object and prompts the user for a sequence of rotation axes and angles that define a change in orientation. ROTMAT is called to calculate the transformation matrix from the user input, and MATMPY combines this new transformation matrix with the old one.


SUBROUTINE LOCTMOD
DISPLAY CURRENT LOCATION OF TARGET OBJECT
PROMPT USER FOR AND READ IN NEW TARGET LOCATION
DISPLAY TRANSFORMATION MATRIX FOR CURRENT ORIENTATION OF TARGET
MODIFY ORIENTATION
T
F
USER INPUT OF A ROTATION AXIS
USER INPUT OF ROTATION ANGLE
DO UNTIL USER STOP FLAG OR 3 ROTATIONS
(NULL)
CALL ROTMAT TO CALCULATE NEW TRANSFORMATION MATRIX
CALL MATMPY TO COMBINE OLD AND NEW TRANSFORMATION MATRICES
RETURN
END

1.2.28 RDTARG

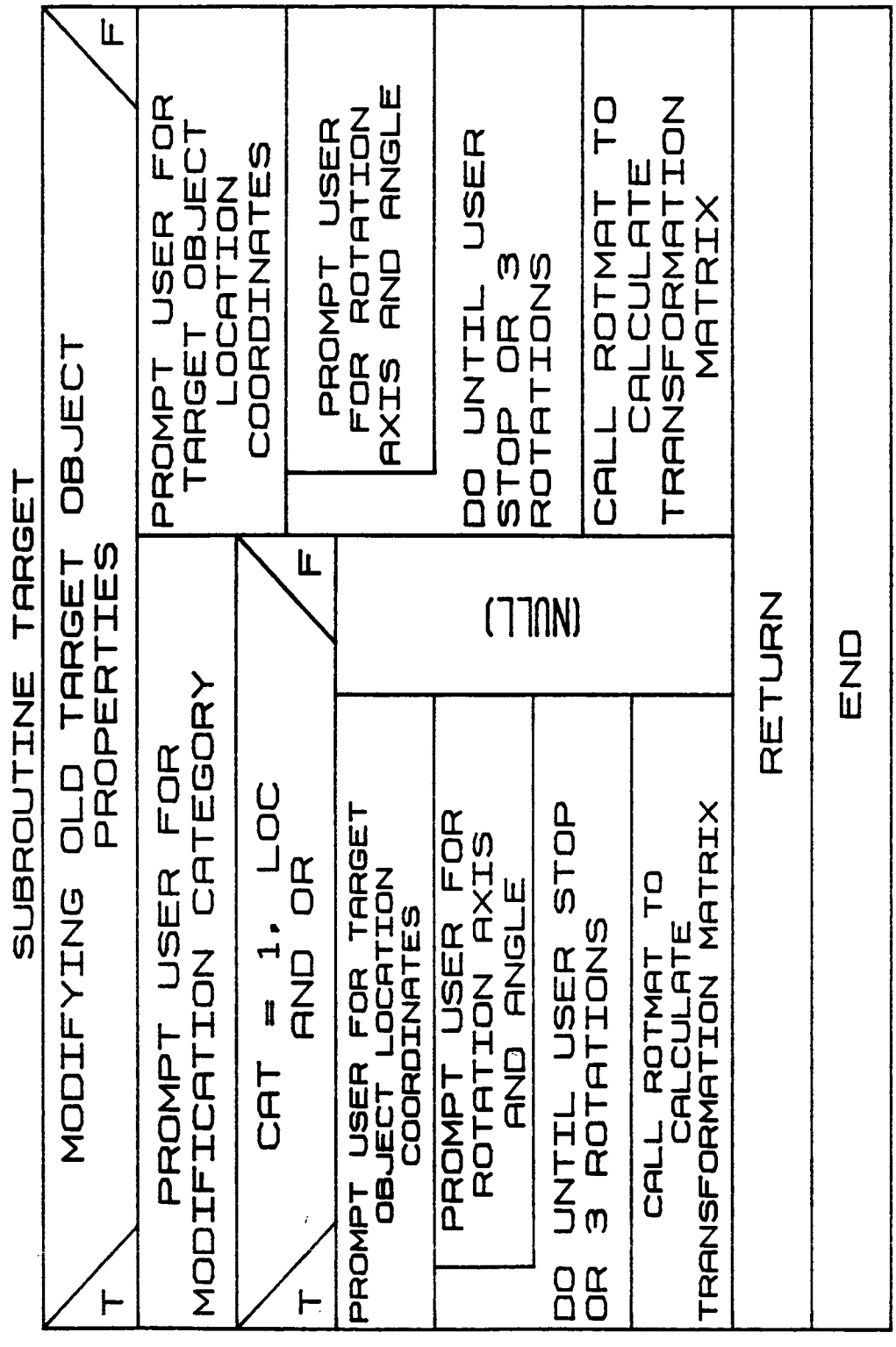
The subroutine RDTARG reads an unformatted target object data file during the system definition function. The contents of the file are the pertinent common blocks defining a target file for the robotic system. The user is prompted for the file name from which the file is to be read.

SUBROUTINE RDTARG



1.2.29 TARGET

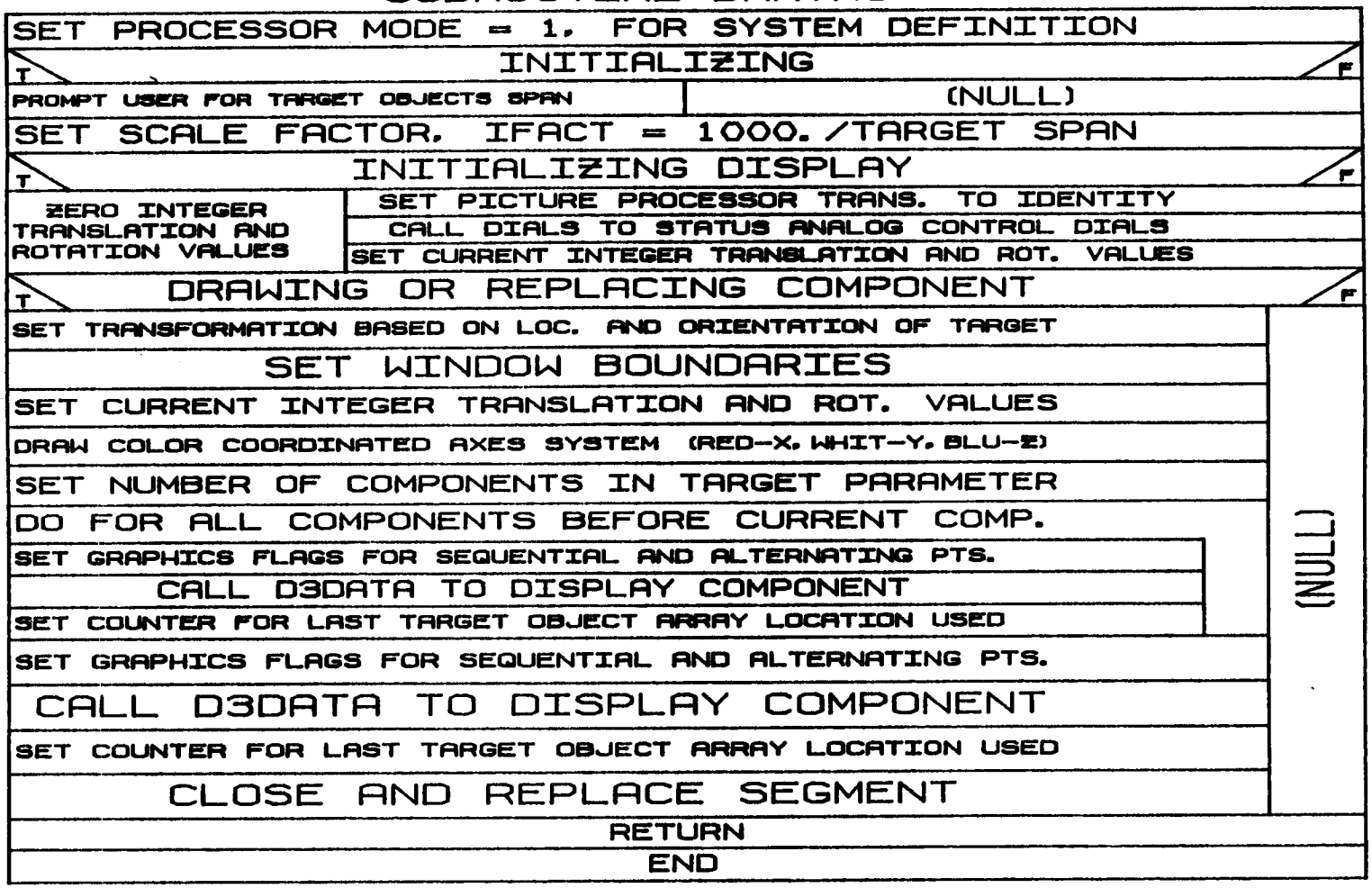
Subroutine TARGET is called during the BLDTRG option of INITDRVR. It allows the user to create and define the properties of one or more target objects. If a file of target object data already exists, this subroutine may be used to modify portions of these data. The target parameters for which the user is prompted are the location and orientation with respect to the world coordinates.


SUBROUTINE TARGET
MODIFYING OLD TARGET OBJECT PROPERTIES
T
F
PROMPT USER FOR MODIFICATION CATEGORY
PROMPT USER FOR TARGET OBJECT LOCATION COORDINATES
PROMPT USER FOR ROTATION AXIS AND ANGLE
DO UNTIL USER STOP OR 3 ROTATIONS
CALL ROTMAT TO CALCULATE TRANSFORMATION MATRIX
CAT = 1, LOC AND OR
T
F
PROMPT USER FOR TARGET OBJECT LOCATION COORDINATES
(NULL)
PROMPT USER FOR ROTATION AXIS AND ANGLE
DO UNTIL USER STOP OR 3 ROTATIONS
CALL ROTMAT TO CALCULATE TRANSFORMATION MATRIX
RETURN
END

1.2.30 DRWTRG

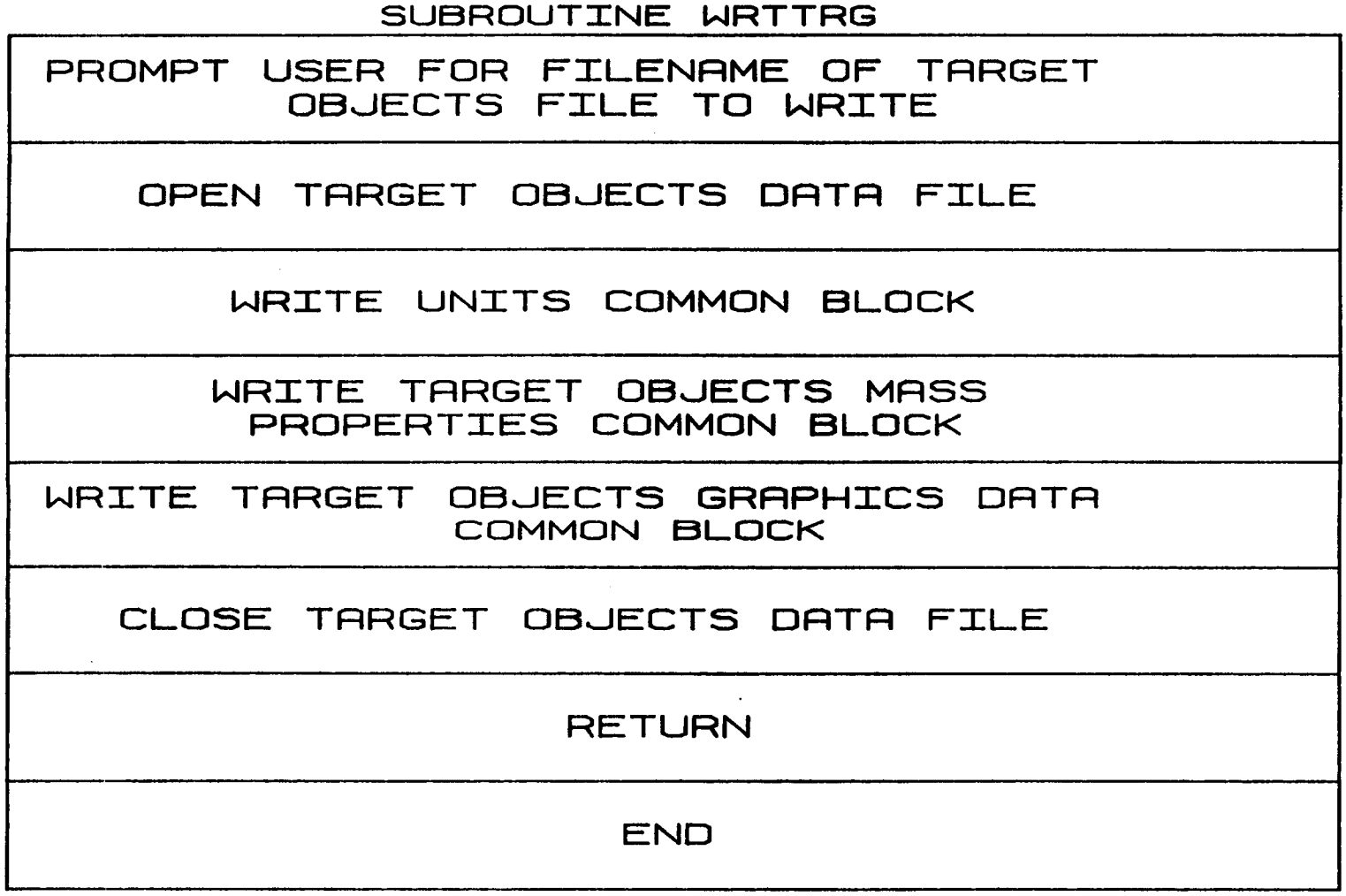
DRWTRG is called within the system definition function from BLDTRG to provide graphics display during the generation of a detailed target object file. It is called to display each successive target object component as it is defined.

SUBROUTINE DRWTRG



1.2.31 WRTTRG

Subroutine WRTTRG writes the unformatted target object data file during the system definition function. The contents of the file are the pertinent common blocks defining the mass properties and graphics of the targets for the system.


SUBROUTINE WRTTRG
PROMPT USER FOR FILENAME OF TARGET OBJECTS FILE TO WRITE
OPEN TARGET OBJECTS DATA FILE
WRITE UNITS COMMON BLOCK
WRITE TARGET OBJECTS MASS PROPERTIES COMMON BLOCK
WRITE TARGET OBJECTS GRAPHICS DATA COMMON BLOCK
CLOSE TARGET OBJECTS DATA FILE
RETURN
END

1.3.1 GRINIT

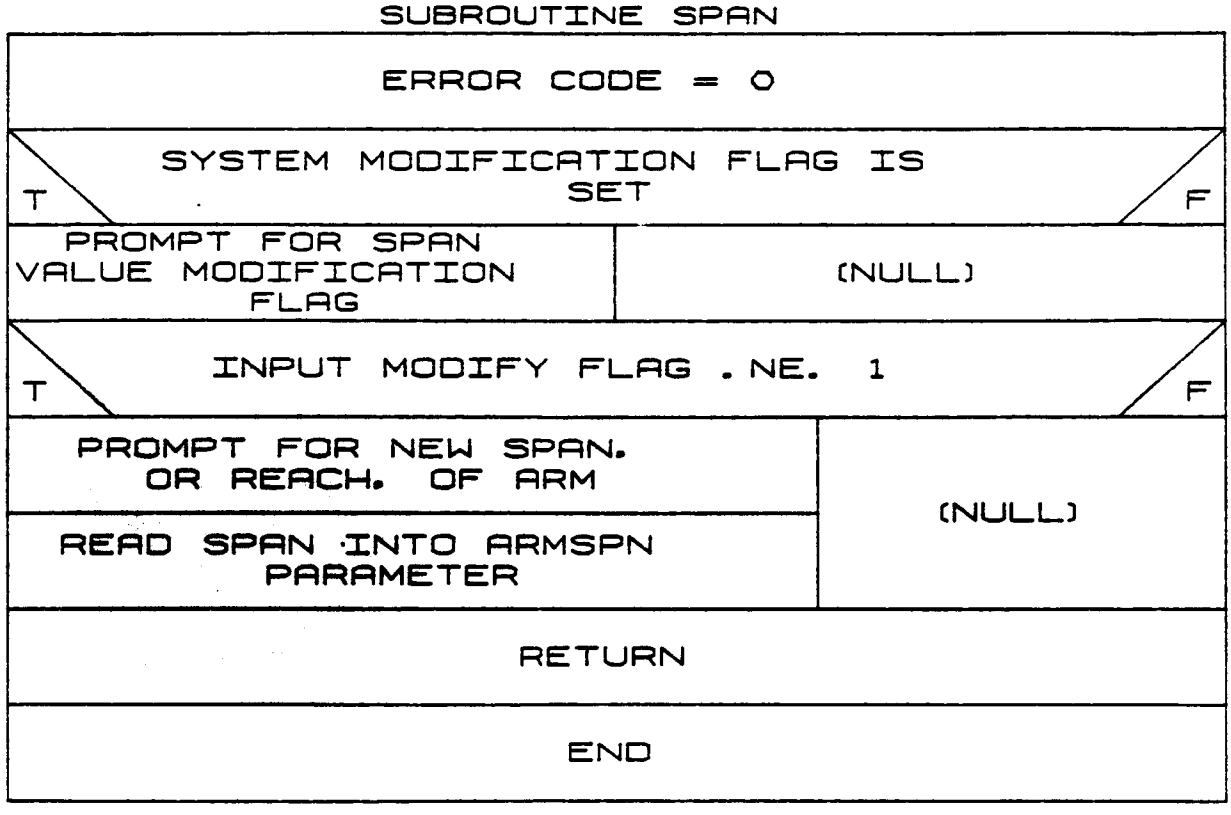
For building an environment, simple arm, detailed arm, targets or loads with graphics, routine GRINIT initializes the E&S display, extended switches/lights, and analog control dials, and draws the graphics display border. Working from the input flag IFLAG, the type heading of the general display is chosen, either simple cylinder, detailed geometry, environment, target or load. The graphics segments are opened and the title for the system definition function driver currently under execution is output.

SUBROUTINE GRINIT

CALL MPINIT TO INITIALIZE GRAPHICS SYSTEM
INITIALIZE EXTENDED FUNCTION KEY SWITCHES
INITIALIZE EXTENDED FUNCTION KEY LIGHTS
INITIALIZE ANALOG CONTROL DIALS
INITIALIZE EVENT QUEUE
DRAW GRAPHICS DISPLAY BORDER
OUTPUT GRAPHICS DISPLAY TITLE
OUTPUT MARTIN MARIETTA COMPANY LOGO
TRANSFER INITIAL READINGS OF CONTROL DIALS FROM DEVICE QUEUE
RETURN
END

1.3.2 SPAN

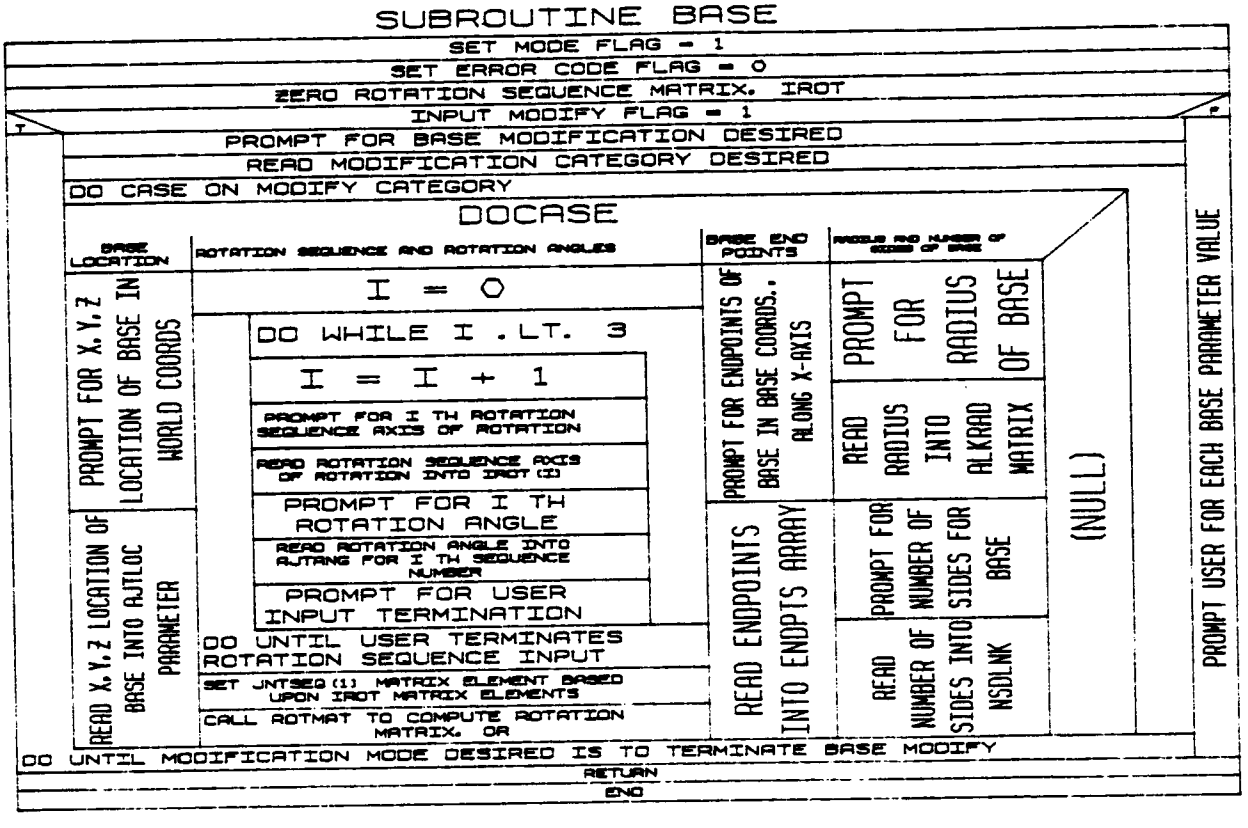
The manipulator arm span is requested as input from the user during initial creation of simple cylinder arm data; modification of the ARMSPN value is also allowed through a call to SPAN during the CREATARM modification mode.



ORIGINAL PAGE IS
OF POOR QUALITY

1.3.3 BASE

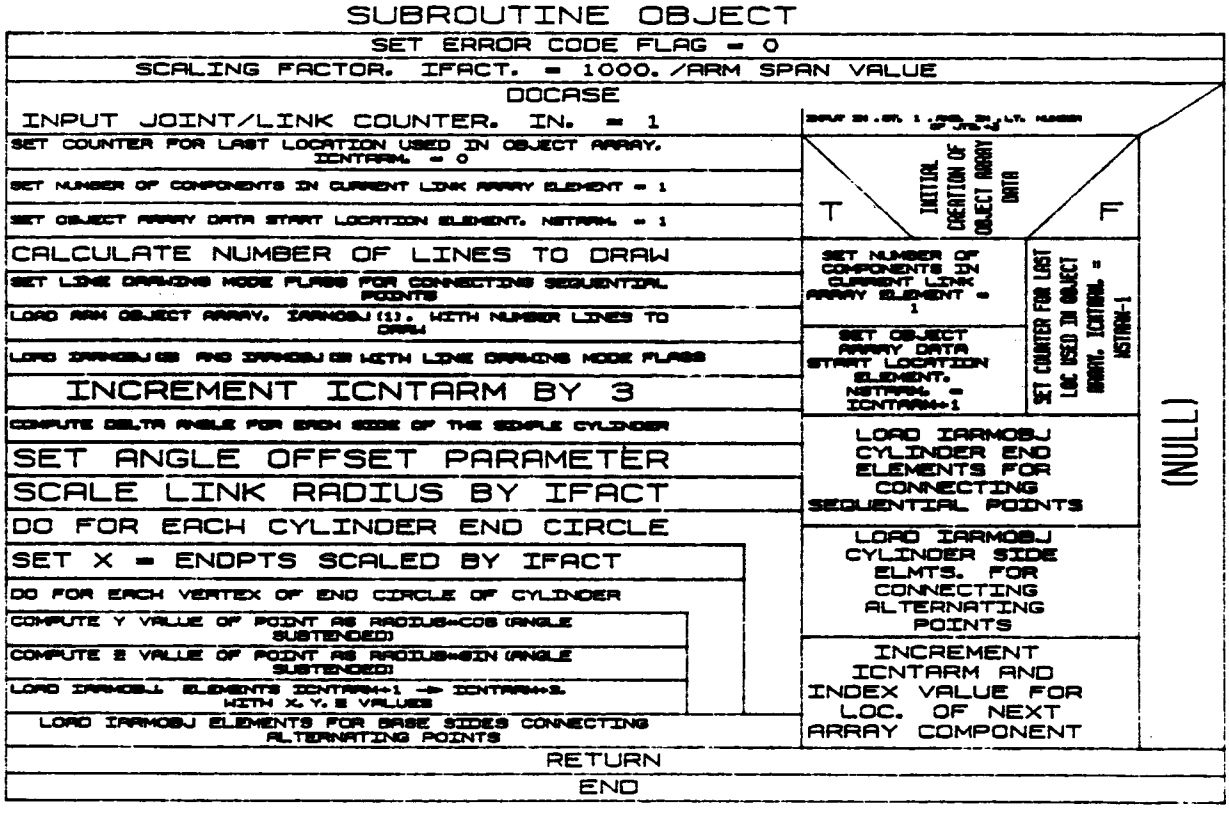
Subroutine BASE is called within the system definition function during definition or modification of the simple cylinder arm or detailed arm geometry file. The purpose of subroutine BASE is to provide the input of the robotic base position, orientation, and physical dimensions (radius of base, endpoints and number of sides), and to load these values into COMMON blocks.



ORIGINAL PAGE IS
OF POOR QUALITY

1.3.4 OBJECT

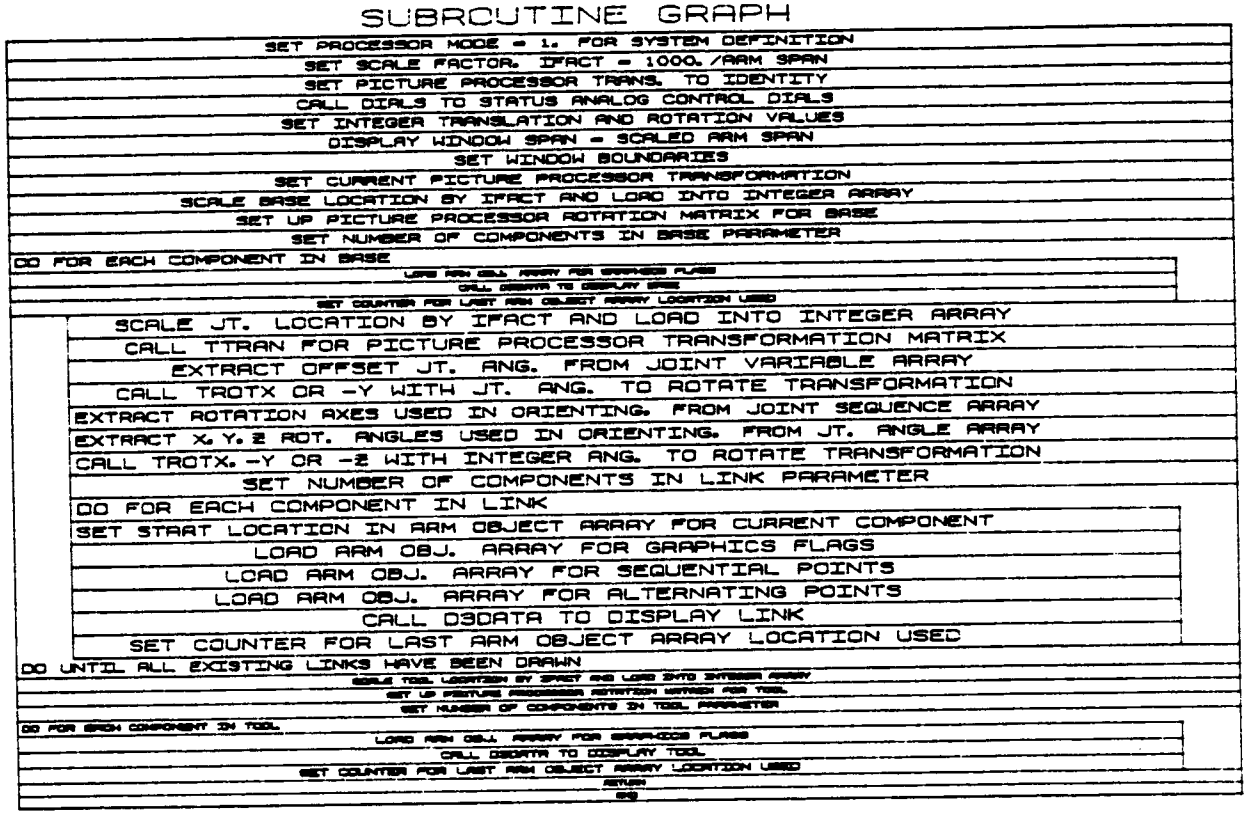
Subroutine OBJECT creates simple cylinder graphics data used by the graphics package to draw the robotic arm during the system definition function. The data created in OBJECT are stored in COMMON block IARMOBJ, and represent a right circular cylinder of the specified size for each system component (the base, each link and the tool). It is called for generation of each of these components in turn.



ORIGINAL PAGE IS
OF POOR QUALITY

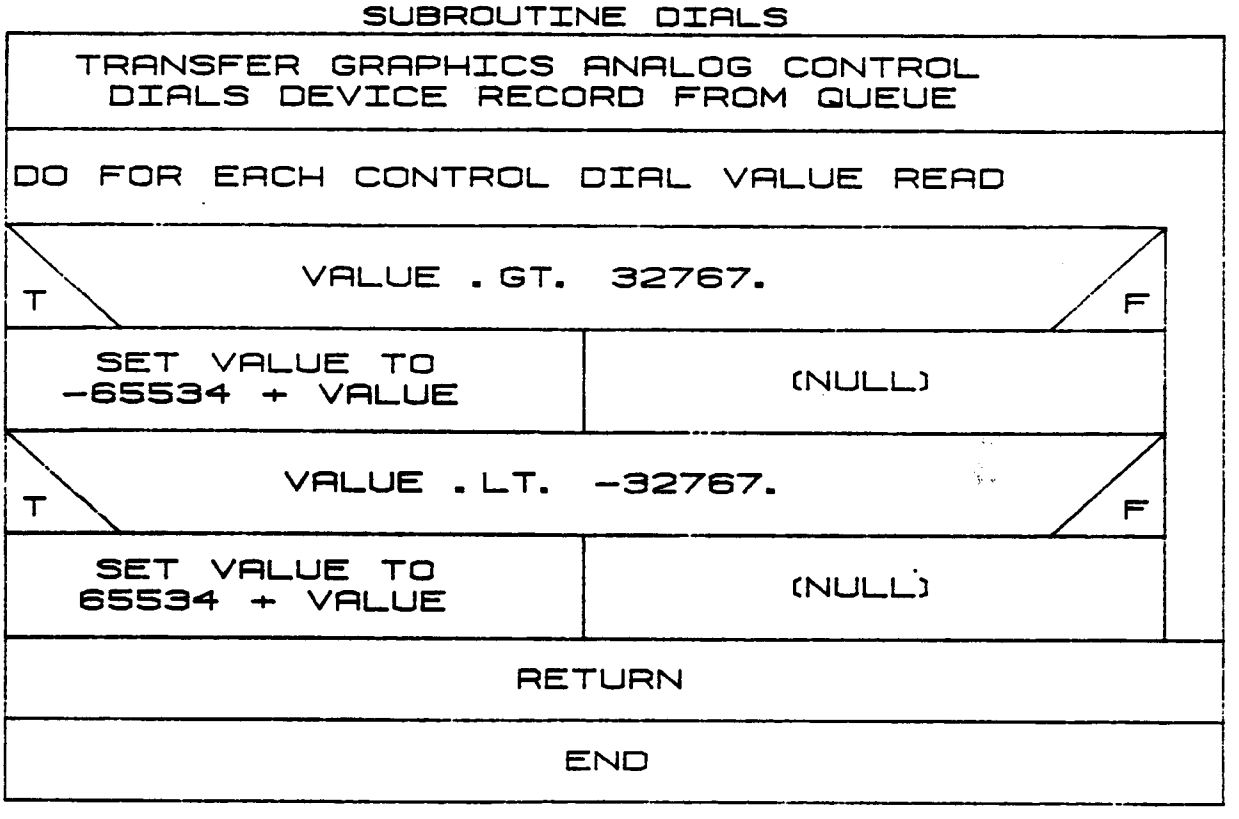
1.3.5 GRAPH

If the graphics are initiated during the simple single arm creation or modification, subroutine GRAPH provides the graphics capability for the simple cylinder representation of the robotic manipulator. GRAPH displays each base, joint/link combination and tool as they are defined. Graphics during the modification mode is handled with calling arguments input to GRAPH; appropriate deletions, additions and changes to the links are visually depicted. GRAPH provides only the simplified robotic arm definition display.



1.3.6 DIALS

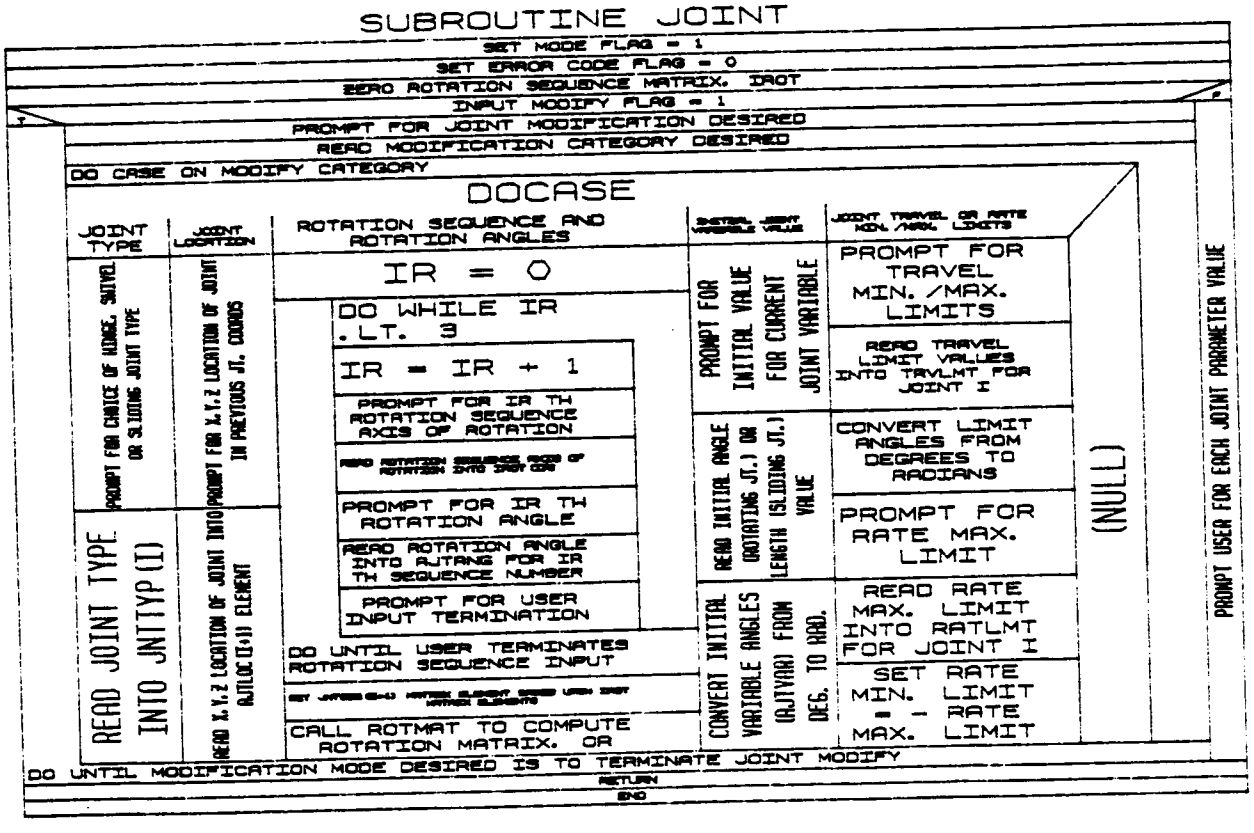
DIALS is called to scale the Evans and Sutherland analog control dials values read during camera perspective changes via the extended E&S dials. The values are scaled to integers between -32767 and +32767.



ORIGINAL PAGE IS
OF POOR QUALITY

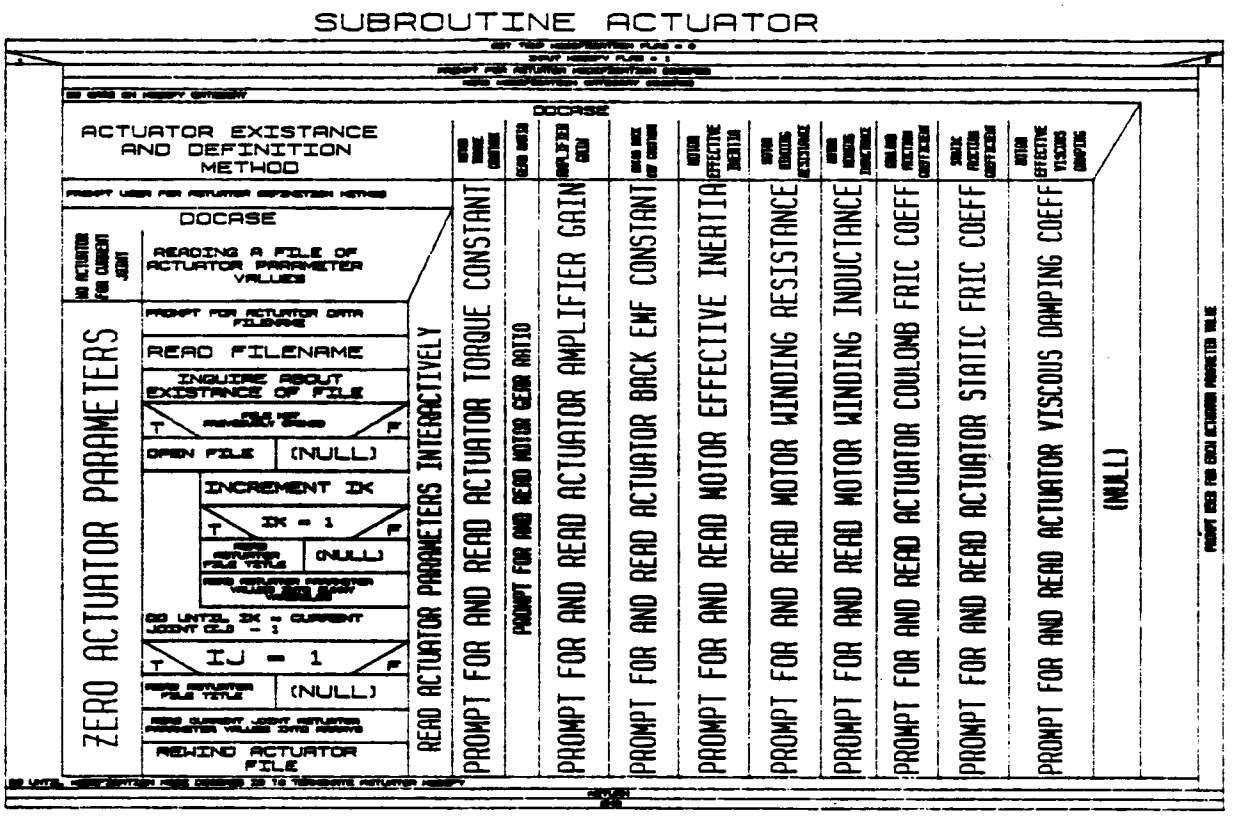
1.3.7 JOINT

For JOINT, the user inputs joint type, joint location as Cartesian coordinates in terms of the coordinate system of the previous joint (or base, if the current joint is joint 1), joint orientation as a rotation sequence of axes and corresponding angles with respect to the previous joint coordinate system (or base, if joint 1), and initial joint state (initial angle for hinge or swivel, or initial length for sliding joints). The x-axis of a joint coordinate system is directed along the centerline of the link between joint i and joint i+1 (or end-effector if the current joint is the final joint in the system). JOINT is called by CREATARM during initial creation or modification of arm data.



1.3.8 ACTUATOR

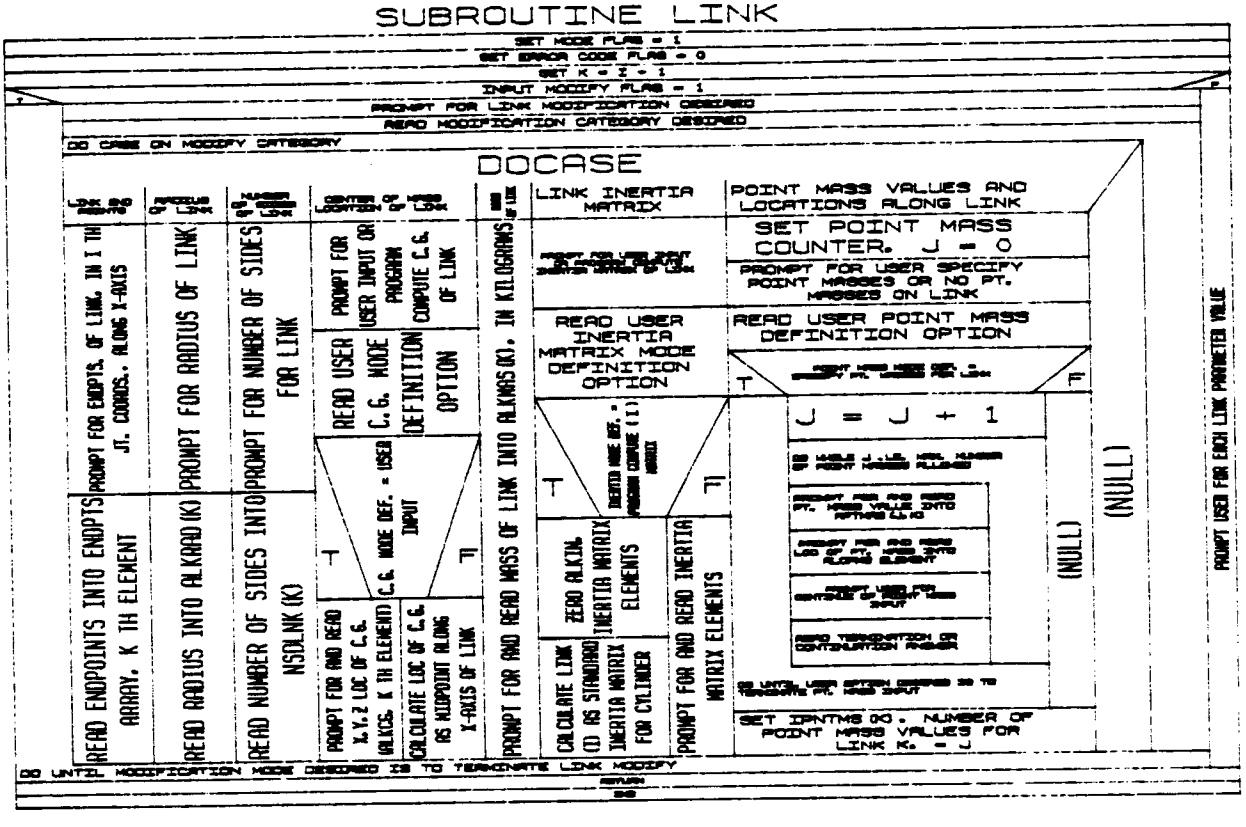
ACTUATOR allows the user to define or modify the COMMON blocks defining actuator properties for the arm by interactively prompting for actuator parameter values or by reading a previously constructed file of actuator parameter values. The user can opt for no actuator definition if desired.



ORIGINAL PAGE IS
OF POOR QUALITY

1.3.9 LINK

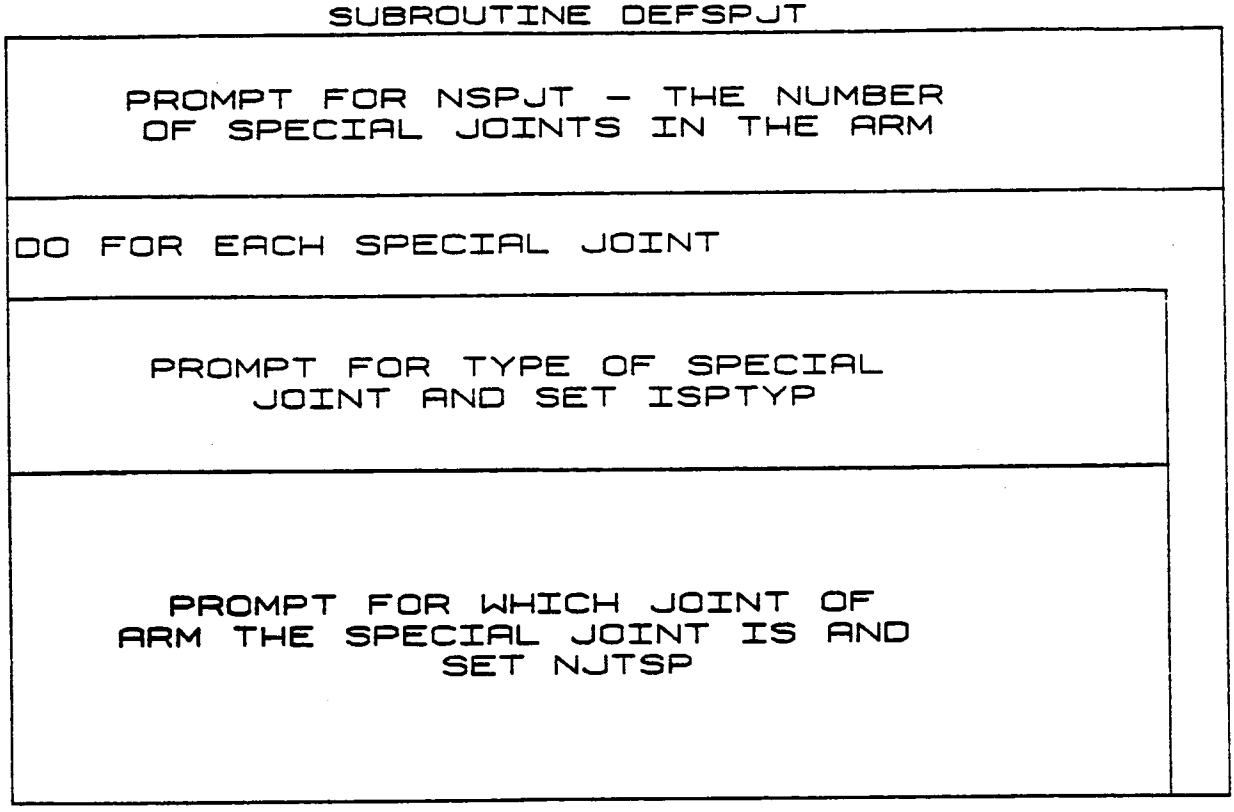
In the create mode of LINK, the user is prompted for link endpoints in coordinates along the x-axis, link radius, the location of center of mass as the Cartesian coordinates of the center of gravity in the coordinate system of the joint at the "base" end of the link, link mass and inertia matrix relative to the centroid and the number of sides for the desired simple cylinder.



C-2

1.3.10 DEFSPJT

This routine interactively establishes the number, type and location of "special joints." These are joints for which a constraint is placed on the relative joint displacements.



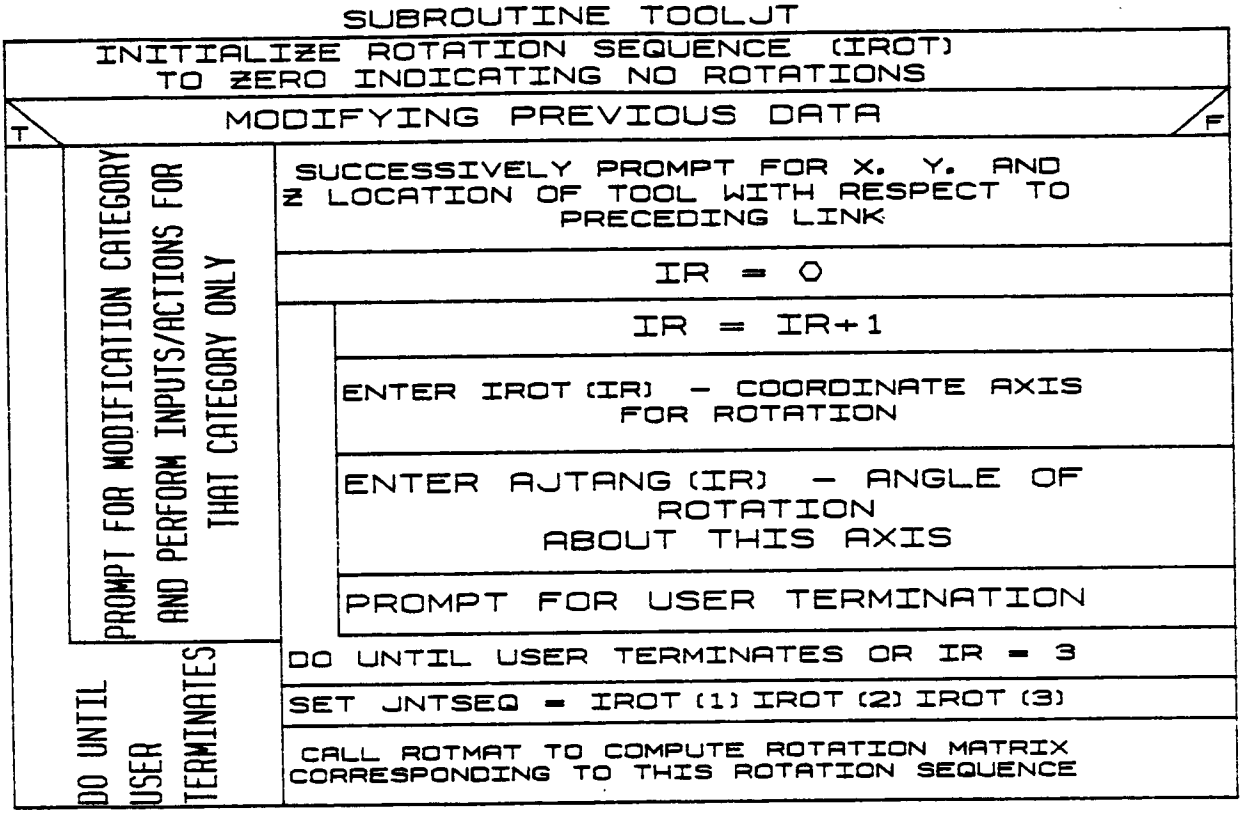
1.3.11 TOOLJT

Subroutine TOOLJT is called by CREATARM to interactively define or modify the geometry properties of the manipulator end-effector. The data for which the user is prompted include:

1) Location of tool with respect to final link;

2) Orientation of tool with respect to final link.

The orientation data are input as a sequence of rotations about coordinate axes and ROTMAT is called to compute the corresponding rotation matrix.



ORIGINAL PAGE IS
OF POOR QUALITY

1.3.12 TOOLLK

Subroutine TOOLLK is called by CREATARM to interactively define or modify the mass and graphics properties of the manipulator end-effector. The data for which the user is prompted include:

1) Endpoints for cylinder representation;

2) Radius of cylinder;

3) Number of sides of cylinder;

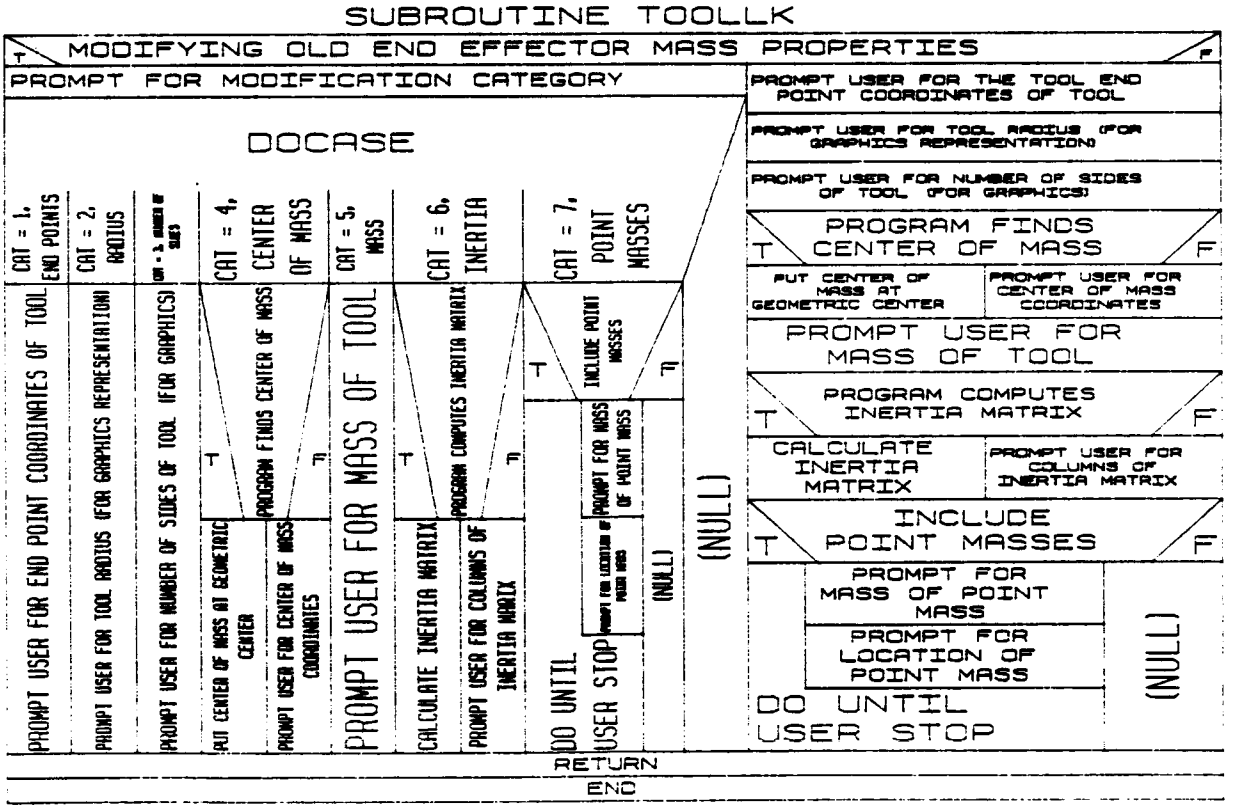
4) Center of mass of end-effector;

5) Mass;

6) Inertia distribution;

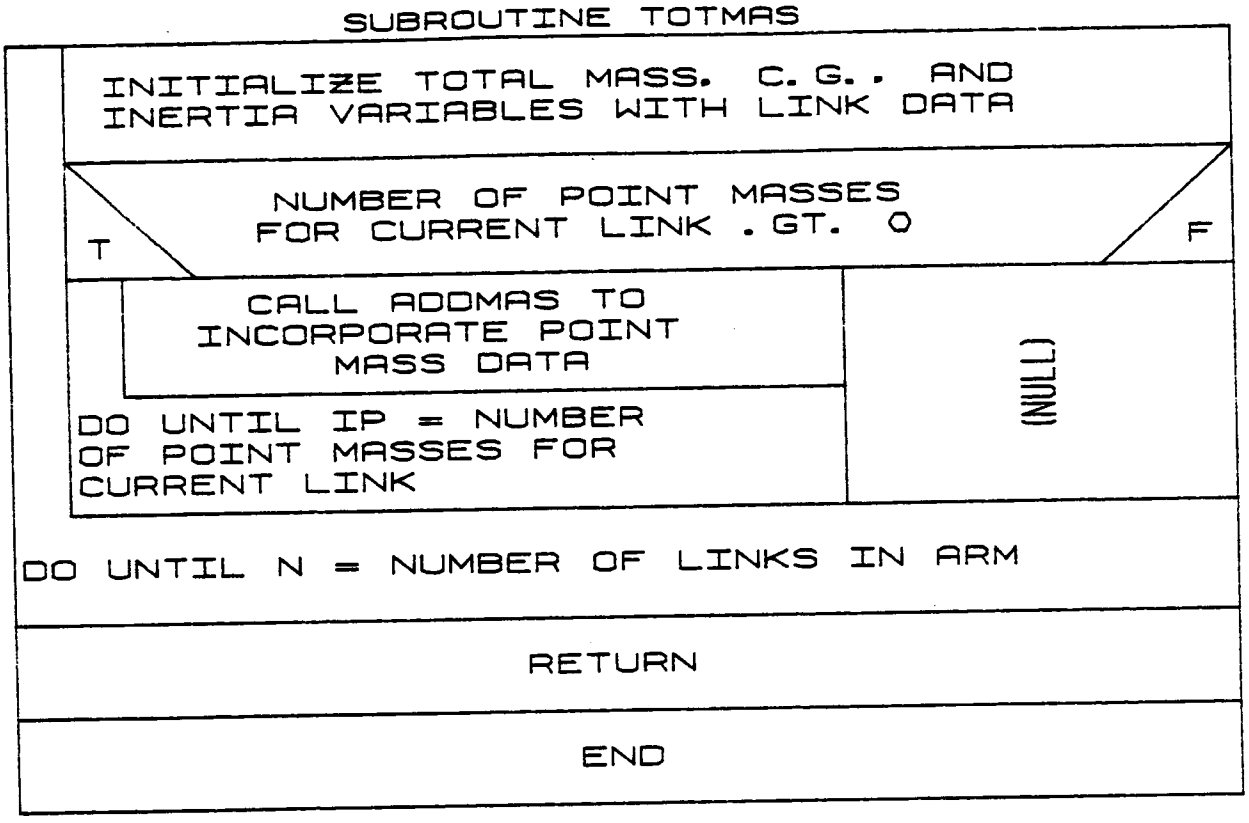
7) Location and mass of point masses.

When modifying existing data, the user has the option of which categories to modify.



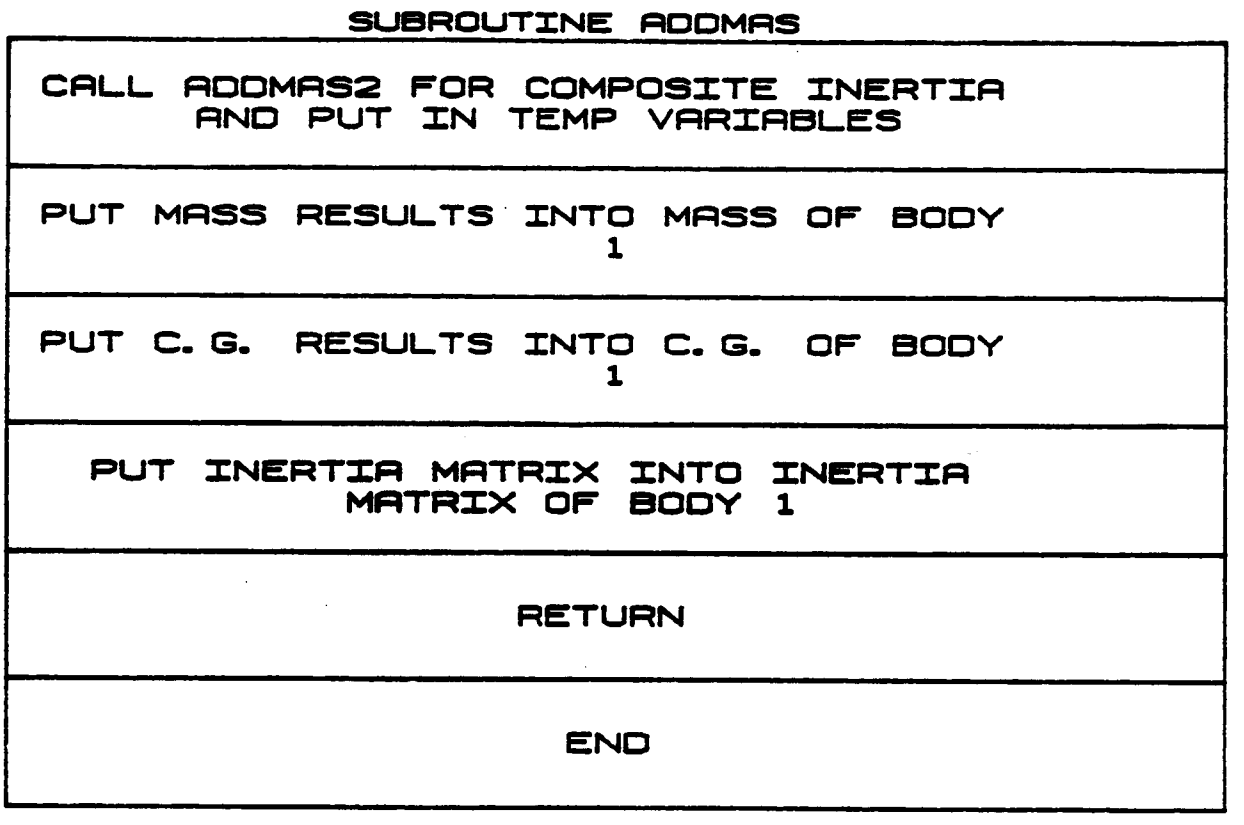
1.3.13 TOTMAS

Routine TOTMAS combines individual components of each link's contributions (e.g., link mass, point masses) to obtain a total mass distribution for the joint/link combinations and tool during robot arm creation. Variables for the total mass, centroid location, and inertia distribution are initialized with the values from the simple link. If point masses are included, ADDMAS is called to add the effects of these additional terms.



1.3.14 ADDMAS

Subroutine ADDMAS combines the mass properties of two objects to obtain composite values for the mass, centroid location, and inertia distribution. ADDMAS calls ADDMAS2 to perform the computations. ADDMAS then loads the results into the first object's mass property variables.



1.3.15 ADDMAS2

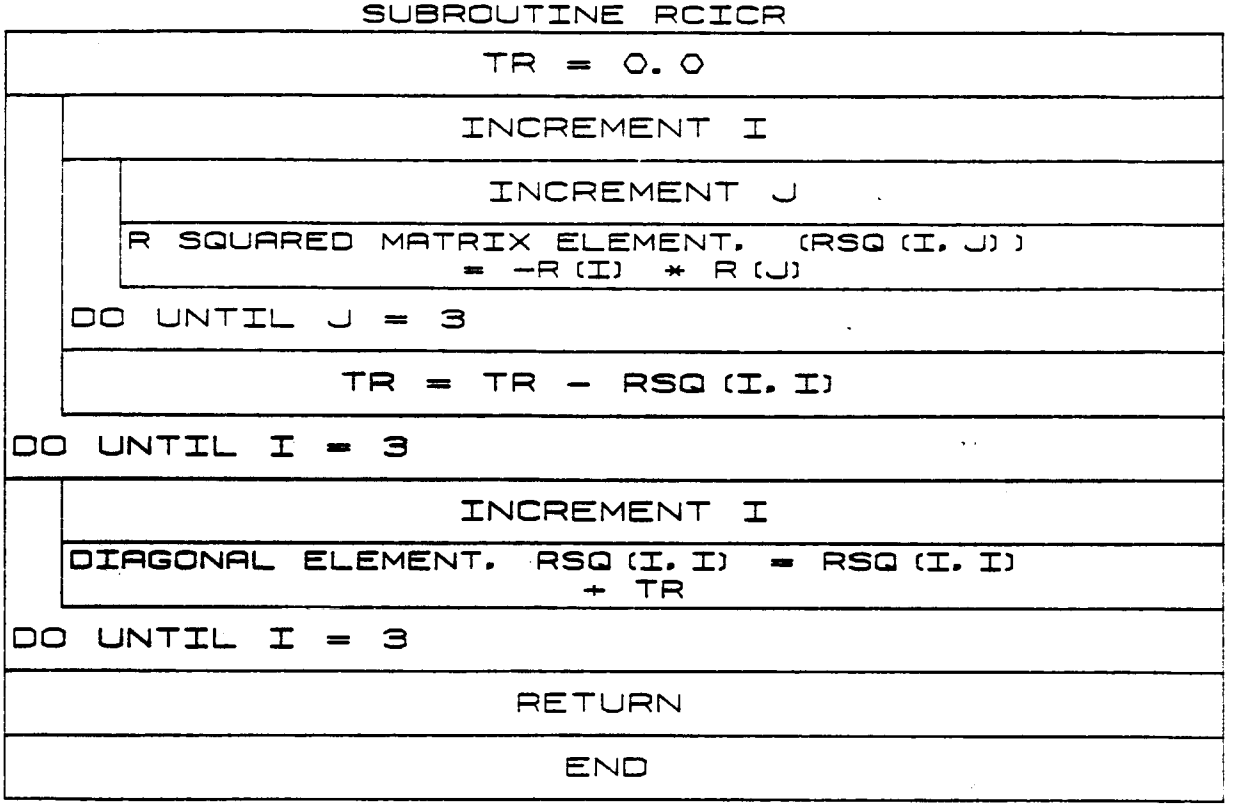
ADDMAS2 calculates the composite mass properties of two rigid bodies joined together. The mass, centroid location, and inertia matrix for the composite body are returned as results.

SUBROUTINE ADDMAS2

TOTAL MASS = MASS OF BODY 1 + MASS OF BODY 2
COMPOSITE CENTROID = (MASS 1 * CG 1 + MASS 2 * CG 2) / TOTAL MASS
R1 = CG 1 - COMPOSITE CENTROID
R2 = CG 2 - COMPOSITE CENTROID
CALL RCICR FOR R1 SQUARED MATRIX (R1SQ) USED TO FIND COMPOSITE INERTIA
CALL RCICR FOR R2 SQUARED MATRIX (R2SQ) USED TO FIND COMPOSITE INERTIA
COMPOSITE INERTIA = AIN1 + AIN2 + (MASS 1 * R1SQ + MASS 2 * R2SQ)
RETURN
END

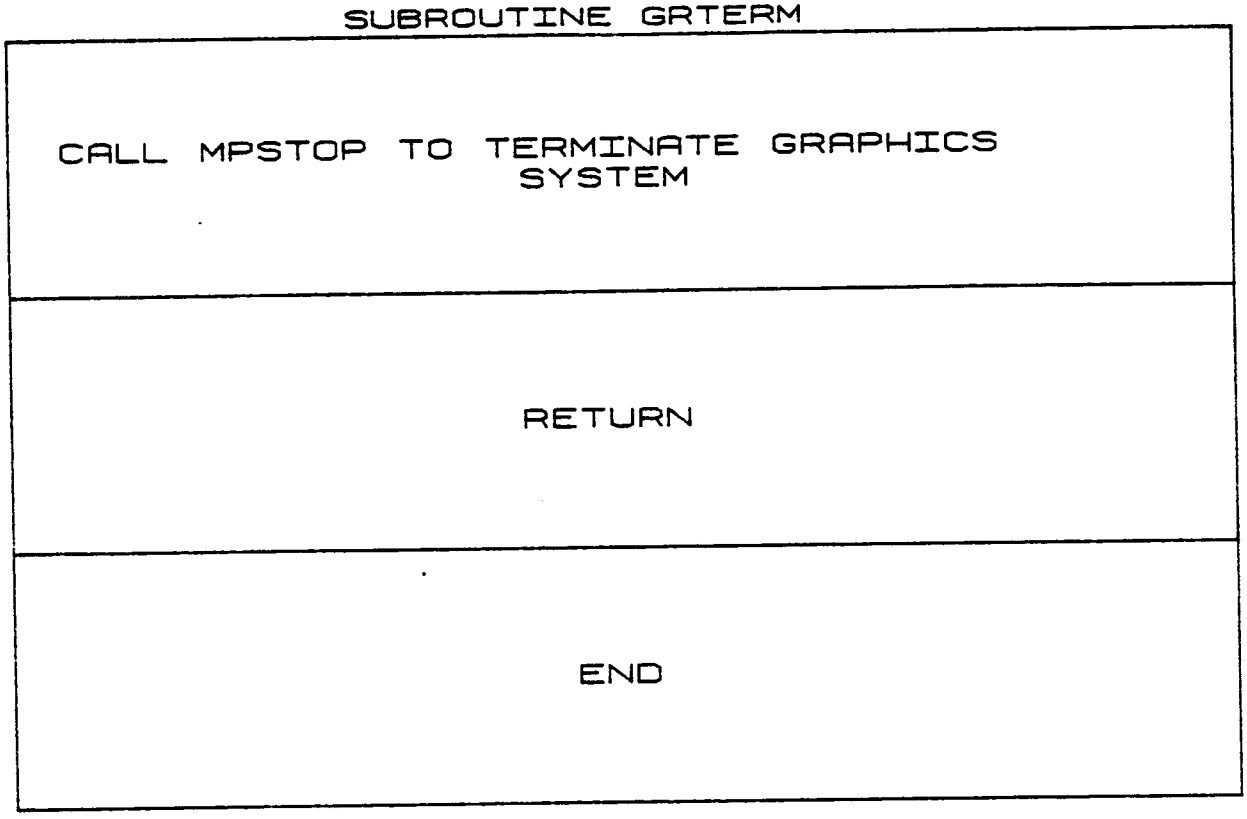
1.3.16 RCICR

Subroutine RCICR is called by ADDMAS2 to set up the inertia matrix corresponding to a point mass displaced from the body centroid. This inertia matrix forms one component of the inertia distribution for the composite body.



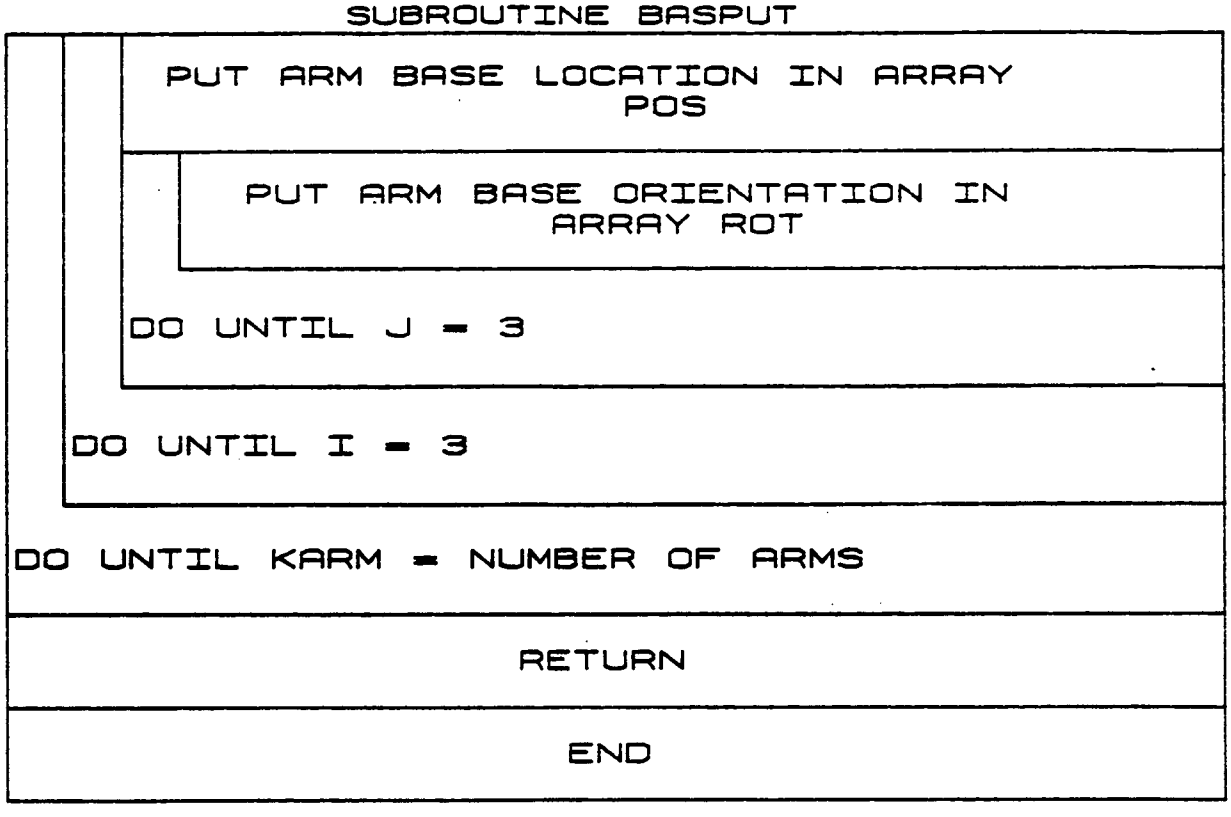
1.3.17 GRTERM

GRTERM is called to terminate the Evans and Sutherland device processor display unit. It calls MPSTOP to terminate the multi-picture processor display unit graphics.



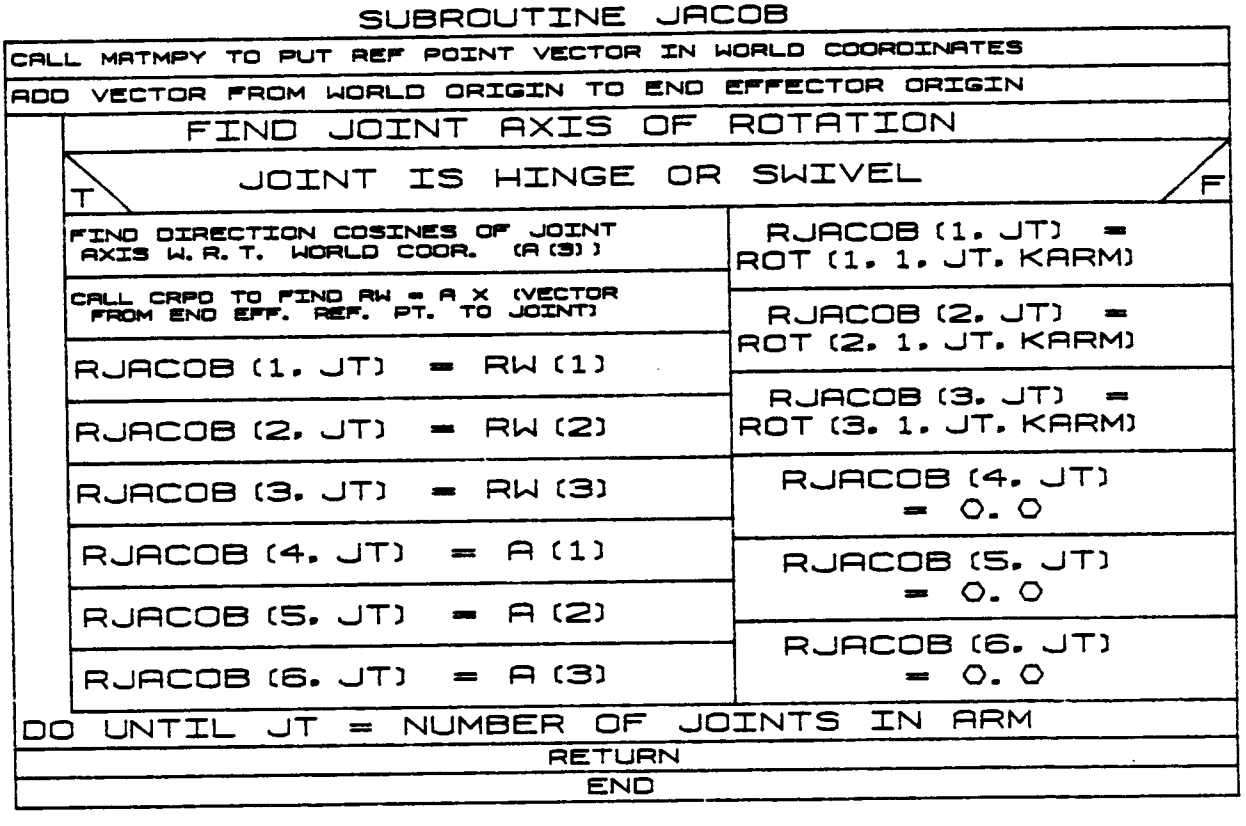
1.3.18 BASPUT

Subroutine BASPUT is called from subroutine SETUP2 during position calculations for the manipulator. This subroutine takes the position and orientation of the base of each arm (with respect to the world coordinate system) and loads these data into the arrays POS and ROT.



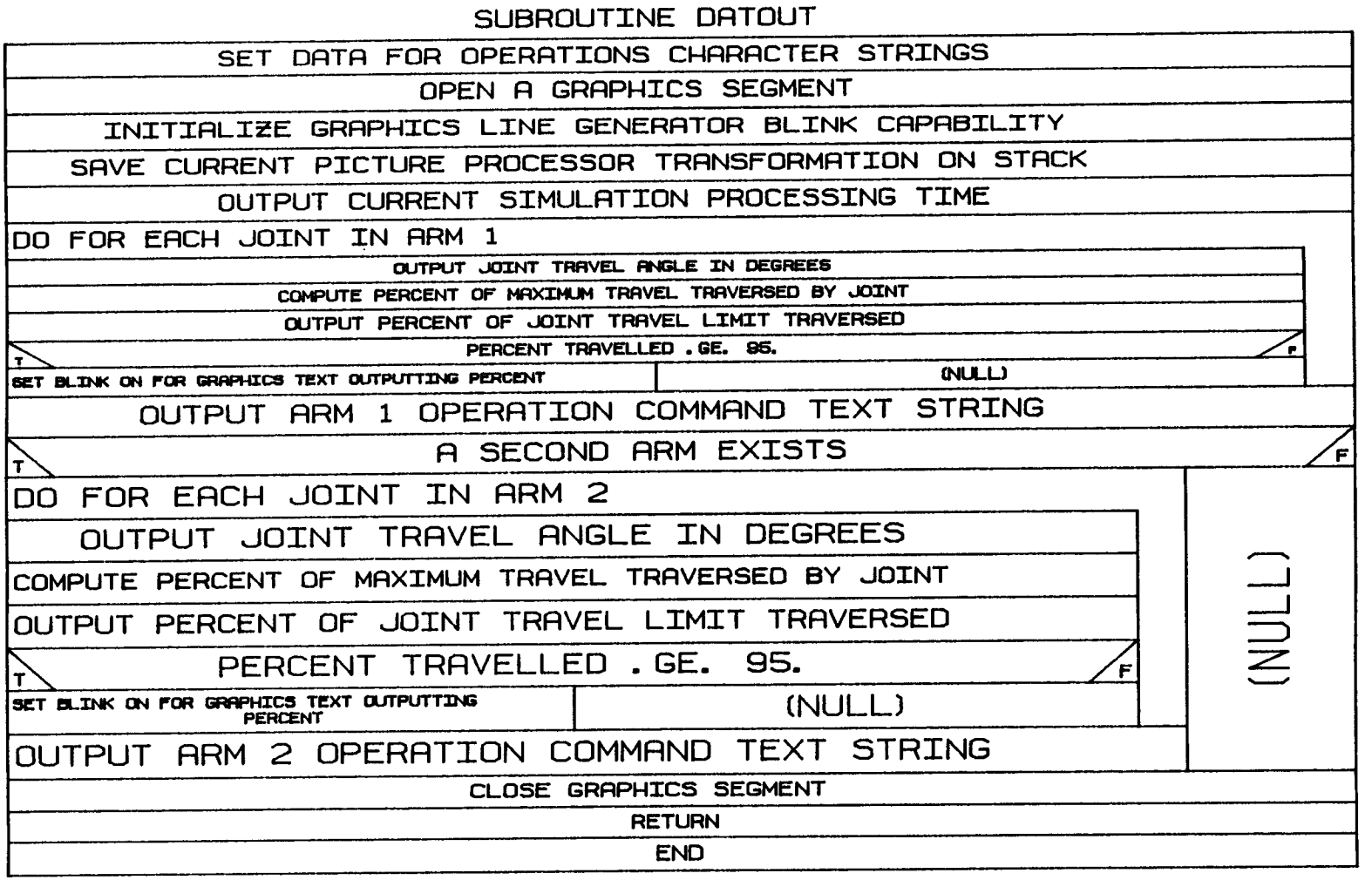
1.3.19 JACOB

Subroutine JACOB sets up the Jacobian matrix that will later be used to solve for individual joint velocities for each arm given the end effector velocity. This subroutine uses end-effector position and joint to world transformation matrices to determine the entries of the Jacobian as described in a previous section. The result is a 6xN matrix for each arm.



1.3.20 DATOUT

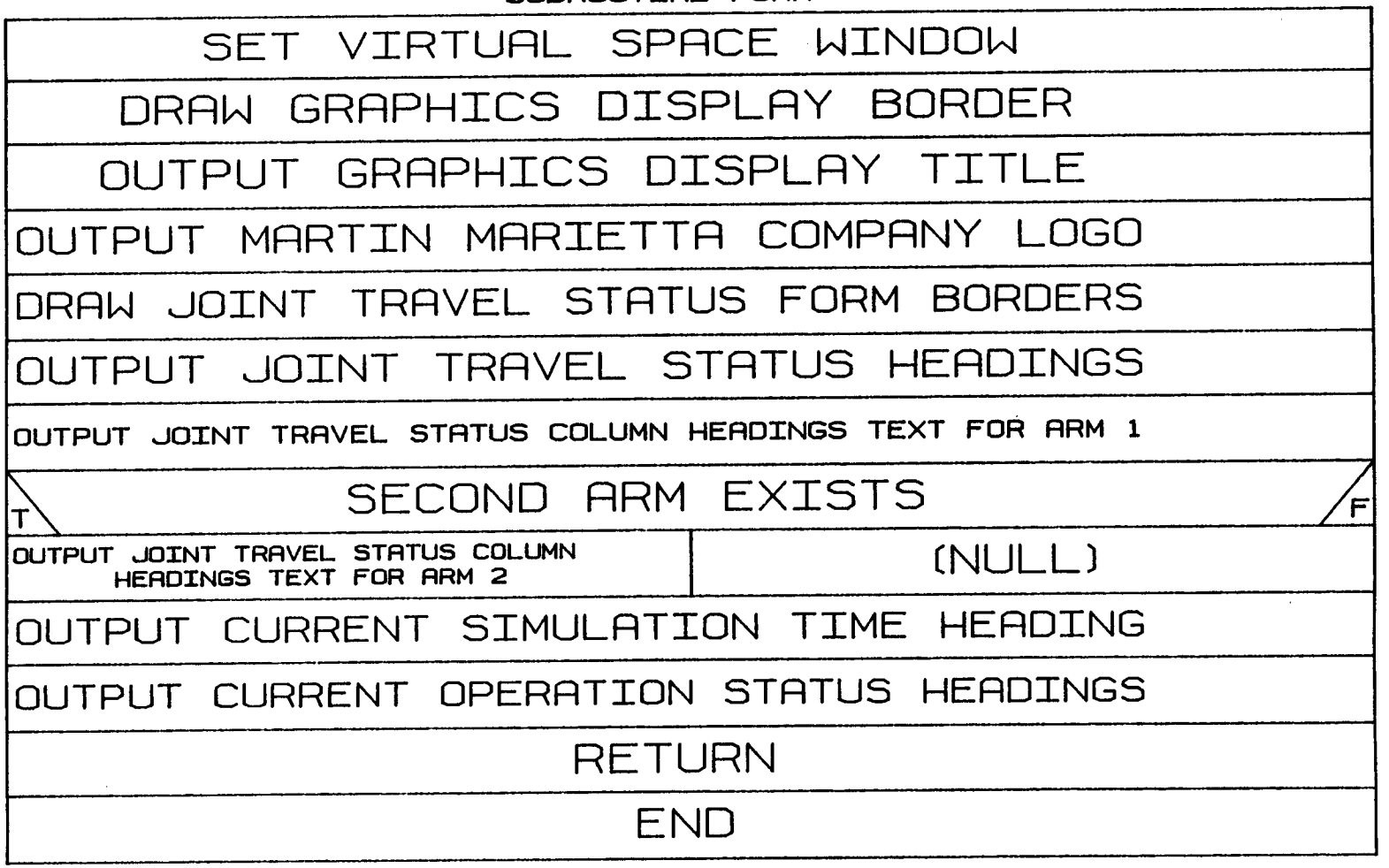
DATOUT is responsible for the data output in the columns set up by subroutine FORM of the E&S robotic simulation display. It includes the current simulation time, joint travel angles, percent of the maximum traveled for each joint and task commands. DATOUT has provisions for only two arms.


SUBROUTINE DATOUT
SET DATA FOR OPERATIONS CHARACTER STRINGS
OPEN A GRAPHICS SEGMENT
INITIALIZE GRAPHICS LINE GENERATOR BLINK CAPABILITY
SAVE CURRENT PICTURE PROCESSOR TRANSFORMATION ON STACK
OUTPUT CURRENT SIMULATION PROCESSING TIME
DO FOR EACH JOINT IN ARM 1
OUTPUT JOINT TRAVEL ANGLE IN DEGREES
COMPUTE PERCENT OF MAXIMUM TRAVEL TRAVERSED BY JOINT
OUTPUT PERCENT OF JOINT TRAVEL LIMIT TRAVERSED
PERCENT TRAVELLED .GE. 95.
T
F
SET BLINK ON FOR GRAPHICS TEXT OUTPUTTING PERCENT
(NULL)
OUTPUT ARM 1 OPERATION COMMAND TEXT STRING
A SECOND ARM EXISTS
T
F
DO FOR EACH JOINT IN ARM 2
OUTPUT JOINT TRAVEL ANGLE IN DEGREES
COMPUTE PERCENT OF MAXIMUM TRAVEL TRAVERSED BY JOINT
OUTPUT PERCENT OF JOINT TRAVEL LIMIT TRAVERSED
PERCENT TRAVELLED . GE. 95.
T
F
SET BLINK ON FOR GRAPHICS TEXT OUTPUTTING PERCENT
(NULL)
OUTPUT ARM 2 OPERATION COMMAND TEXT STRING
(NULL)
CLOSE GRAPHICS SEGMENT
RETURN
END

1.3.21 FORM

The FORM routine sets up the borders and the text output locations for the manipulator display on the E&S graphics unit. It sets up the Evans and Sutherland graphics display borders and outputs the robotic simulation title, current simulation time text title, joint travel status data column and task command headings. FORM has provisions for only two arms.

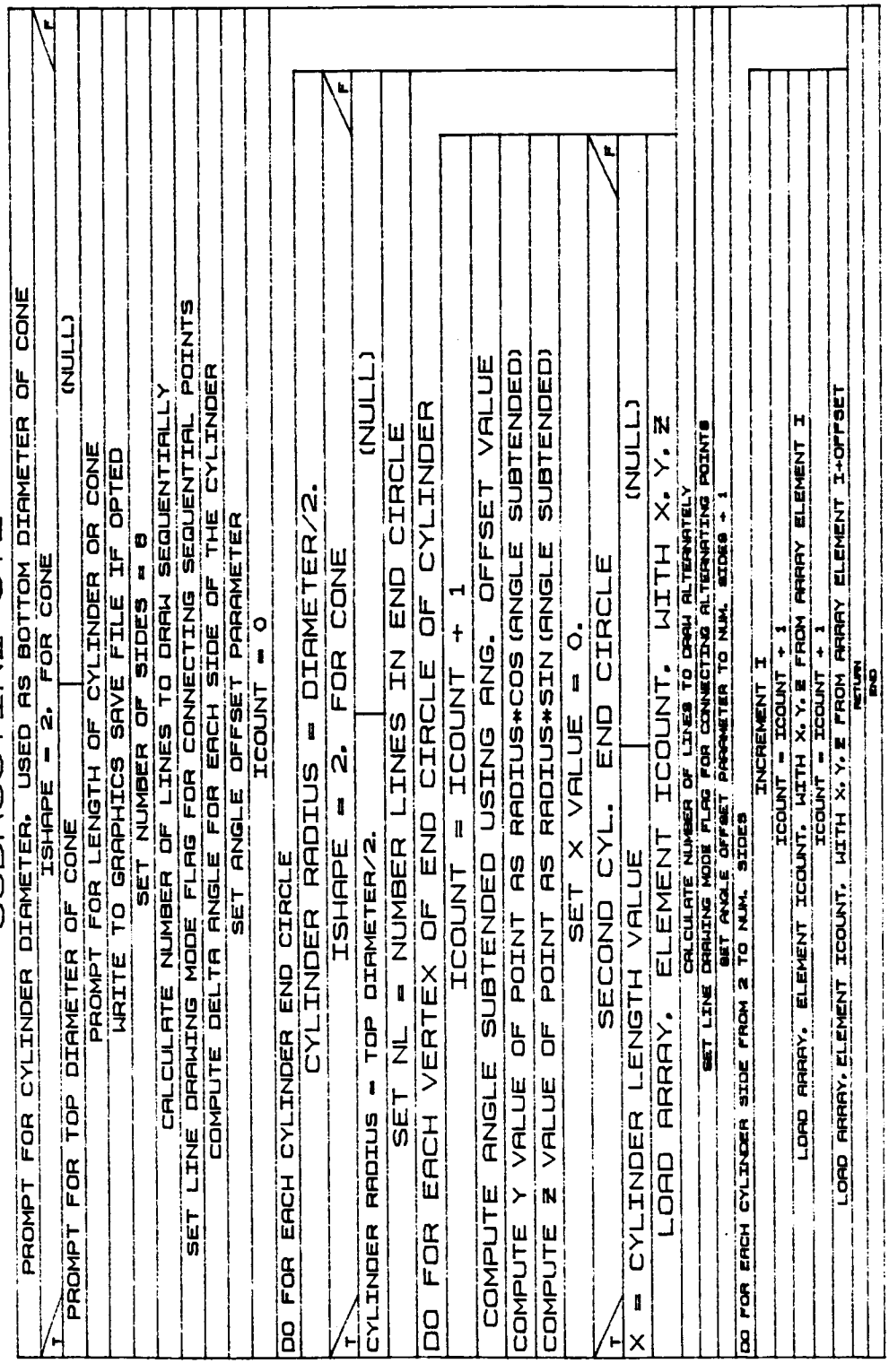
SUBROUTINE FORM



1.3.22 CYL

Subroutine CYL is called within the system definition function during detailed graphic representation generation for the robotic system constituents (environment, arms, targets, loads). If the requested component is a cylinder or cone, it is called to compute data points for the graphic routines. The controlling argument in the call, ISHAPE, determines which geometric shape has been chosen in calling routine BLDENV, BLDDAT, BLDTRG or BLDENV. The detailed graphic component dimensions are written to a print/save file for archiving the program interaction.

ORIGINAL PAGE IS
OF POOR QUALITY



1.3.23 RECT

Subroutine RECT is called within the system definition function during generation of detailed graphic representations for environment, arm, target, or load objects file. It is called if the requested component is a rectangular solid (ISHAPE = 3), a symmetric trapezoidal solid (ISHAPE = 4), or a nonsymmetric trapezoidal solid (ISHAPE = 5) to compute data points for the graphic routines. The detailed graphic component dimensions are written to a print/save file for archiving the program interaction.

SUBROUTINE RECT

PROMPT FOR RECTANGLE/TRAPEZOID LENGTH IN X. +/- WIDTH IN Y

ISHAPE = 3. FOR RECTANGLE SOLID

T	F
PROMPT FOR RECTANGLE +/- BASE DIMENSION	PROMPT FOR TRAPEZOID FIRST AND SECOND BASE DIMENSION

WRITE TO GRAPHICS SAVE FILE IF OPTED

SET NUMBER OF SIDES = 4

CALCULATE NUMBER OF LINES TO DRAW SEQUENTIALLY AND SET MODE

DO FOR EACH RECT./TRAP. END

FIRST END

T	F
SET OFFSET PARAMETER = 0., X = 0.	SET OFFSET PARAMETER = NUM. SIDES + 1
	SET X = RECT./TRAP. LENGTH
SET Z = FIRST RECT./TRAP. BASE VALUE	SET Z = SECOND RECT./TRAP. BASE VALUE

LOAD ARRAY X, Y, Z, ELEMENT 1+OFFSET, WITH X, WIDTH AND BASE VALUE

ISHAPE = 4. FOR SYMMETRIC TRAP.

T	F
LOAD ARRAY Z VALUE, ELEMENT 1+OFFSET, WITH Z	(NULL)

ISHAPE = 5. FOR NONSYMMETRIC TRAP.

T	F
LOAD ARRAY Z VALUE, ELEMENT 1+OFFSET, WITH 0. VALUE	(NULL)

LOAD ARRAY X, Y, Z, ELEMENT 2+OFFSET, WITH X, WIDTH AND -BASE VALUE

ISHAPE = 4. FOR SYMMETRIC TRAP.

T	F
LOAD ARRAY Z VALUE, ELEMENT 2+OFFSET, WITH -Z	(NULL)

ISHAPE = 5. FOR NONSYMMETRIC TRAP.

T	F
LOAD ARRAY Z VALUE, ELEMENT 2+OFFSET, WITH -Z	(NULL)

LOAD ARRAY X, Y, Z, ELEMENT 3+OFFSET, WITH X, -WIDTH AND -BASE VALUE

ISHAPE = 4. FOR SYMMETRIC TRAP.

T	F
LOAD ARRAY Z VALUE, ELEMENT 3+OFFSET, WITH -Z	(NULL)

ISHAPE = 5. FOR NONSYMMETRIC TRAP.

T	F
LOAD ARRAY Z VALUE, ELEMENT 3+OFFSET, WITH -Z	(NULL)

LOAD ARRAY X, Y, Z, ELEMENT 4+OFFSET, WITH X, -WIDTH AND BASE VALUE

ISHAPE = 4. FOR SYMMETRIC TRAP.

T	F
LOAD ARRAY Z VALUE, ELEMENT 4+OFFSET, WITH Z	(NULL)

ISHAPE = 5. FOR NONSYMMETRIC TRAP.

T	F
LOAD ARRAY Z VALUE, ELEMENT 4+OFFSET, WITH 0. VALUE	(NULL)

LOAD ARRAY X, Y, Z, ELEMENT 5+OFFSET, WITH ELEMENT 1+OFFSET VALUES

SET ICOUNT = NUMBER LINES ALREADY DRAWN SEQUENTIALLY

CALCULATE NUMBER OF LINES TO DRAW ALTERNATELY AND SET MODE

SET ANGLE OFFSET PARAMETER TO NUM. SIDES + 1

DO FOR EACH RECT./TRAP. SIDE FROM 2 TO NUM. SIDES

INCREMENT I AND ICOUNT

LOAD ARRAY, ELEMENT ICOUNT, WITH X, Y, Z FROM ARRAY ELEMENT I

ICOUNT = ICOUNT + 1

LOAD ARRAY, ELEMENT ICOUNT, WITH X, Y, Z FROM ARRAY ELEMENT I+OFFSET

RETURN

END

ORIGINAL PAGE IS
OF POOR QUALITY

1.3.24 TRISTR

Subroutine TRISTR is called within the system definition function during detailed graphics representation generation for the environment, manipulator, target, or load objects. If the requested component is a triangular cross-section beam (ISHAPE = 6), it is called to compute data points for the graphics routine. The detailed graphic component dimensions are written to a print/save file for archiving the program interaction.

SUBROUTINE TRISTR

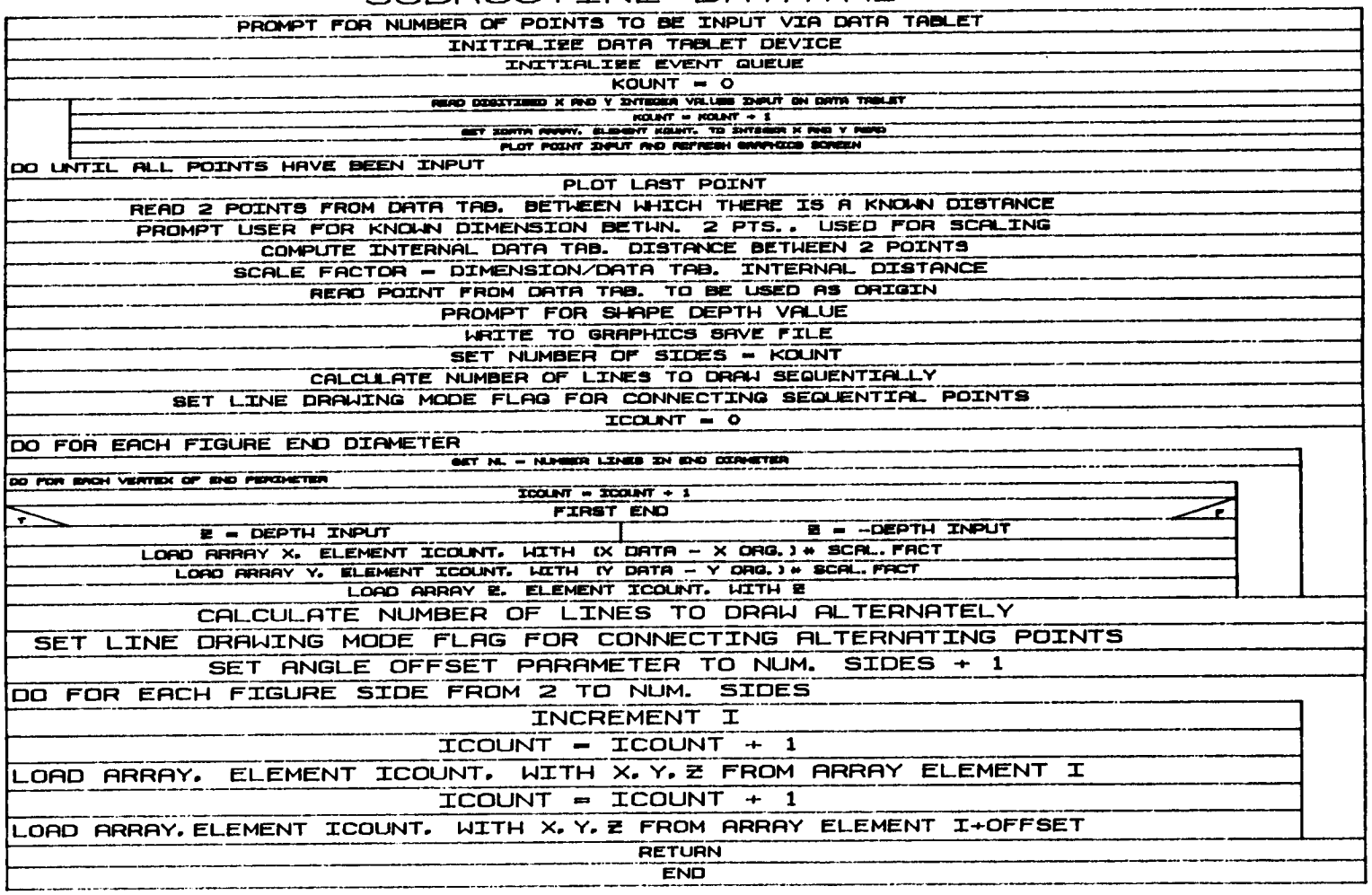
```
PROMPT FOR TRIANGLE SIDE LENGTH
PROMPT FOR SEGMENT LENGTH
PROMPT FOR NUMBER OF SEGMENTS
WRITE TO GRAPHICS SAVE FILE IF OPTED
SET NUMBER OF SIDES = 3
CALCULATE NUMBER OF LINES TO DRAW
SET LINE DRAWING MODE FLAG FOR CONNECTING SEQUENTIAL POINTS
DO FOR EACH SEGMENT OF TRI. STRUCTURE. FROM 1 TO NUM. SIDES + 1
    CALCULATE OFFSET BASED ON CURRENT SEGMENT NUMBER
    X = (CURRENT SEG. NUM. - 1) * (NUM. SIDES + 1)
    LOAD ARRAY X VALUE. ELEMENT 1+OFFSET. WITH X
    LOAD ARRAY Y VALUE. ELEMENT 1+OFFSET. WITH TRI. LENGTH/2.
    LOAD ARRAY Z VALUE. ELEMENT 1+OFFSET. WITH 0.
    LOAD ARRAY X VALUE. ELEMENT 2+OFFSET. WITH X
    LOAD ARRAY Y VALUE. ELEMENT 2+OFFSET. WITH -TRI. LENGTH/2.
    LOAD ARRAY Z VALUE. ELEMENT 2+OFFSET. WITH 0.
    LOAD ARRAY X VALUE. ELEMENT 3+OFFSET. WITH X
    LOAD ARRAY Y VALUE. ELEMENT 3+OFFSET. WITH 0.
    LOAD ARRAY Z VALUE. ELEMENT 3+OFFSET. WITH -TRI. LENGTH
    LOAD ARRAY X. Y. Z VALUES. ELEMENT 4+OFFSET. WITH ELEMT. 1+OFFSET VALUES
SET ICOUNT = NUMBER LINES ALREADY DRAWN SEQUENTIALLY
CALCULATE NUMBER OF LINES TO DRAW ALTERNATELY
SET LINE DRAWING MODE FLAG FOR CONNECTING ALTERNATING POINTS
SET ANGLE OFFSET PARAMETER TO (SEG. NUM.) * (NUM. SIDES + 1)
DO FOR EACH TRIANGLE SIDE FROM 2 TO NUM. SIDES
    INCREMENT I
    ICOUNT = ICOUNT + 1
    LOAD ARRAY. ELEMENT ICOUNT. WITH X. Y. Z FROM ARRAY ELEMENT I
    ICOUNT = ICOUNT + 1
    LOAD ARRAY. ELEMENT ICOUNT. WITH X. Y. Z FROM ARRAY ELEMENT I+OFFSET
RETURN
END
```


ORIGINAL PAGE IS
OF POOR QUALITY

1.3.25 DATATAB

Subroutine DATATAB is called within the system definition function during detailed graphics representation generation for the environment, the robotic arm, target, or load. If the requested component is a data tablet structure, it is called to compute data points for the graphic routines when the input ISHAPE flag = 8. The detailed graphic component dimensions are written to a print/save file for archiving the program interaction.

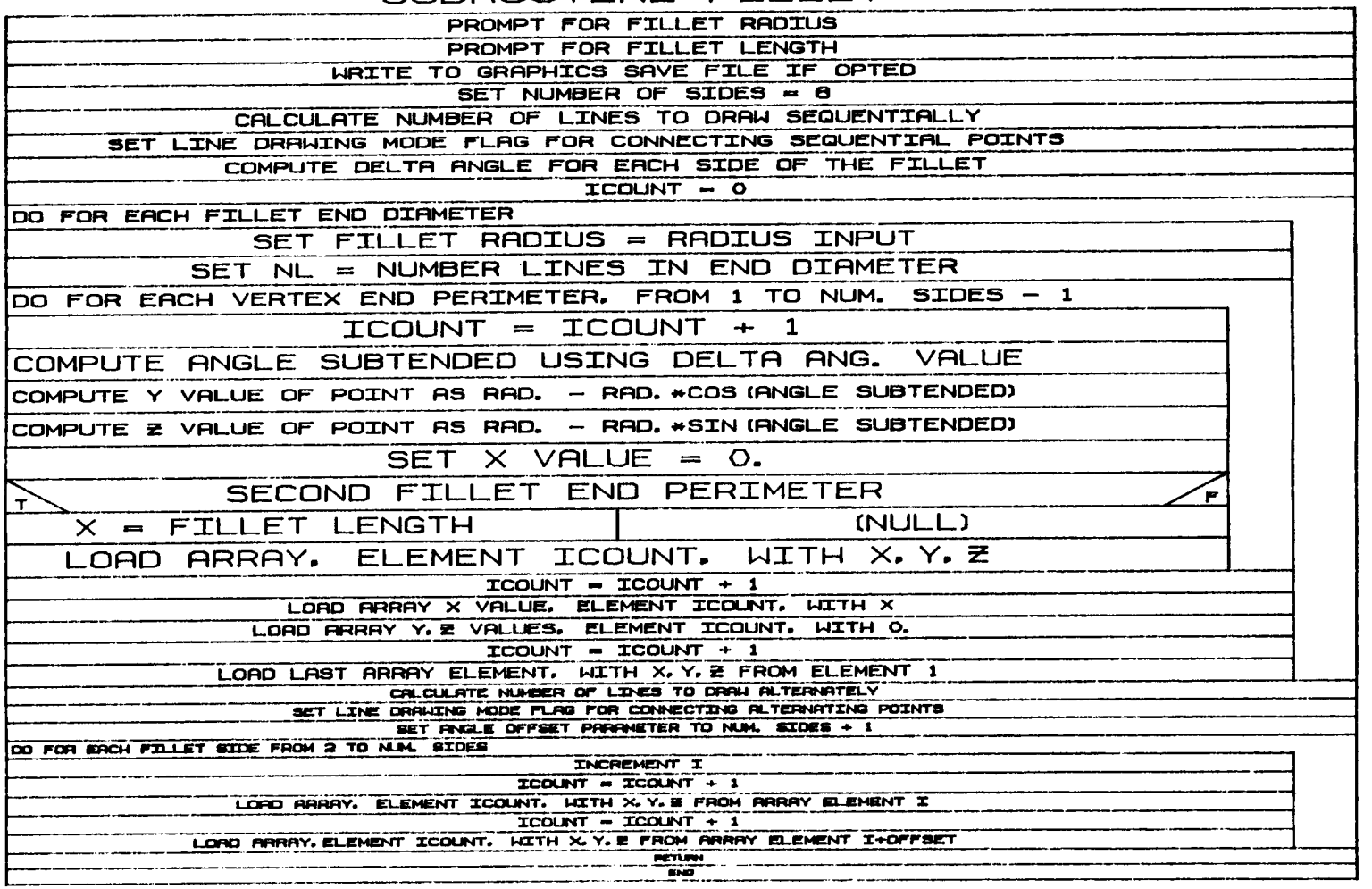
SUBROUTINE DATATAB



1.3.26 FILLET

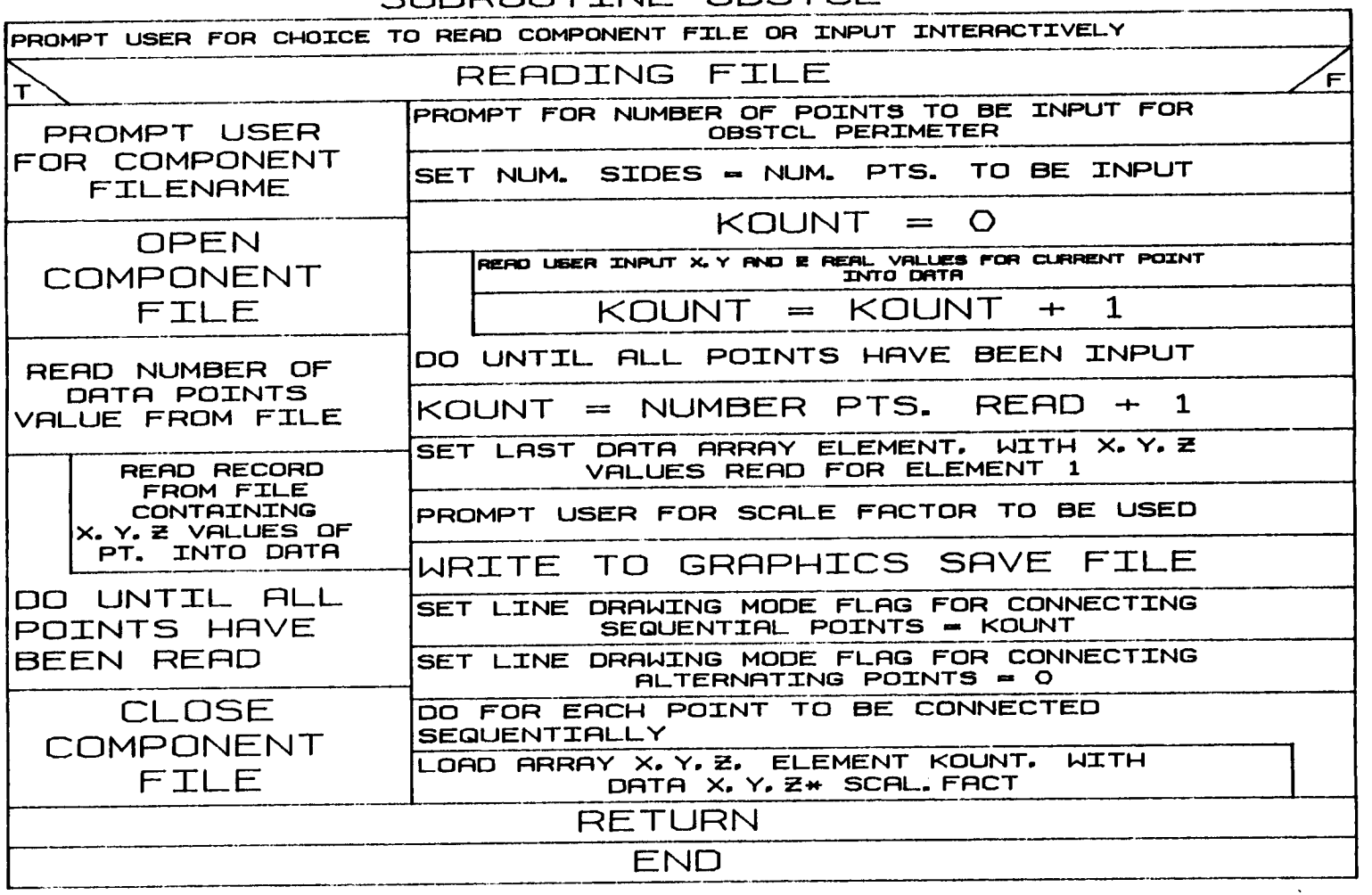
Subroutine FILLET is called within the system definition function during detailed graphics representation generation for the environment, the robotic arm, target, or load. If the requested component is a fillet part, it is called to compute the data points for the graphic routines. For a concave fillet, the input ISHAPE flag is 9. The detailed graphic component dimensions are written to a print/save file for archiving the program interaction.

SUBROUTINE FILLET



1.3.27 OBSTCL

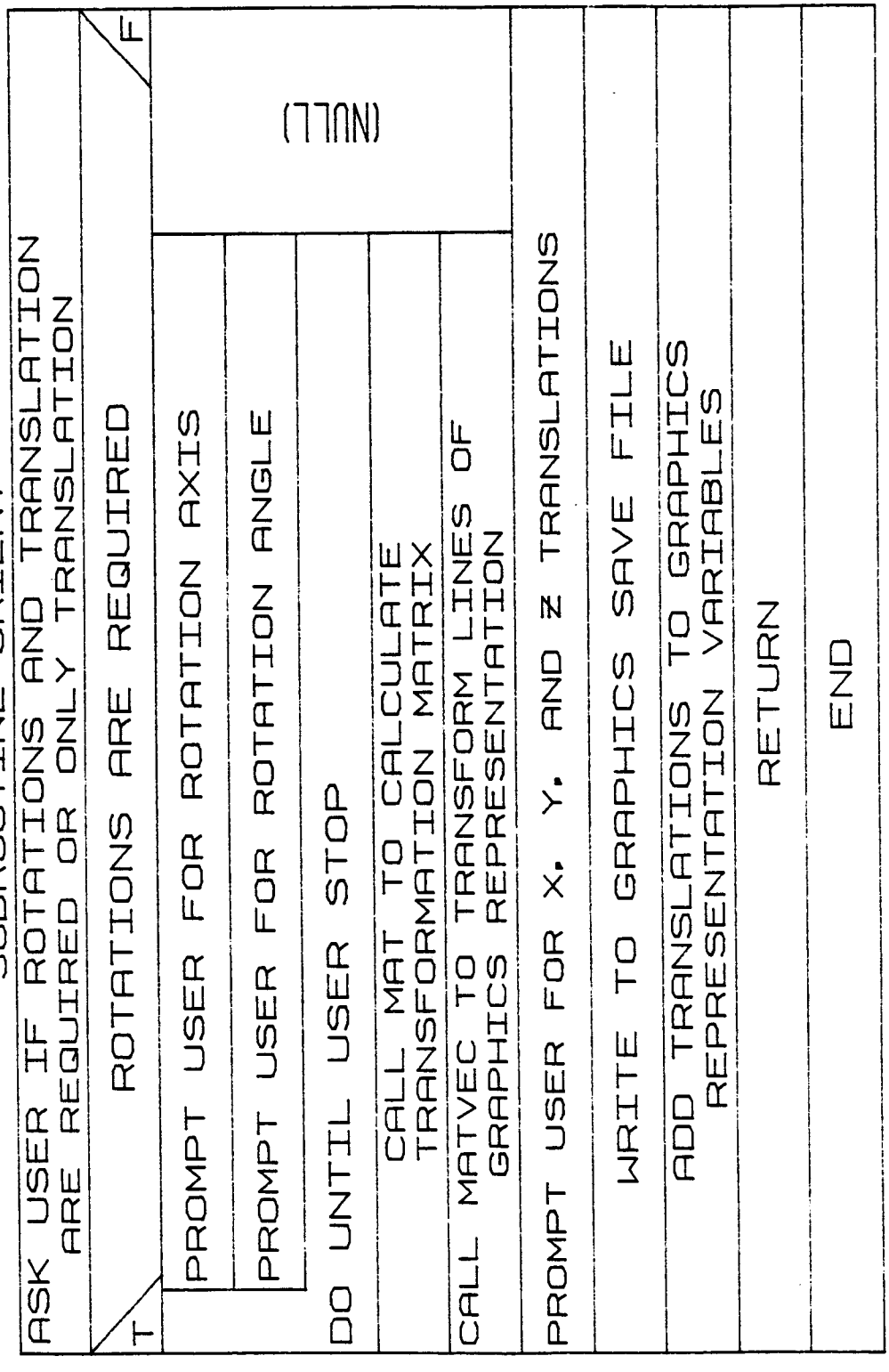
Subroutine OBSTCL is called within the system definition function during graphics representation generation with option ISHAPE equal 10 for the environment, detailed robotic arm, target, or load. It is called if a requested component is an obstacle entity (a choice option from BLDENV) or nonplanar structure (for BLDLOD, BLDTRG or BLDDAT). It computes data points for the graphic routines. The detailed graphic component dimensions are written to a print/save file for archiving the program interaction.



ORIGINAL PAGE IS
OF POOR QUALITY

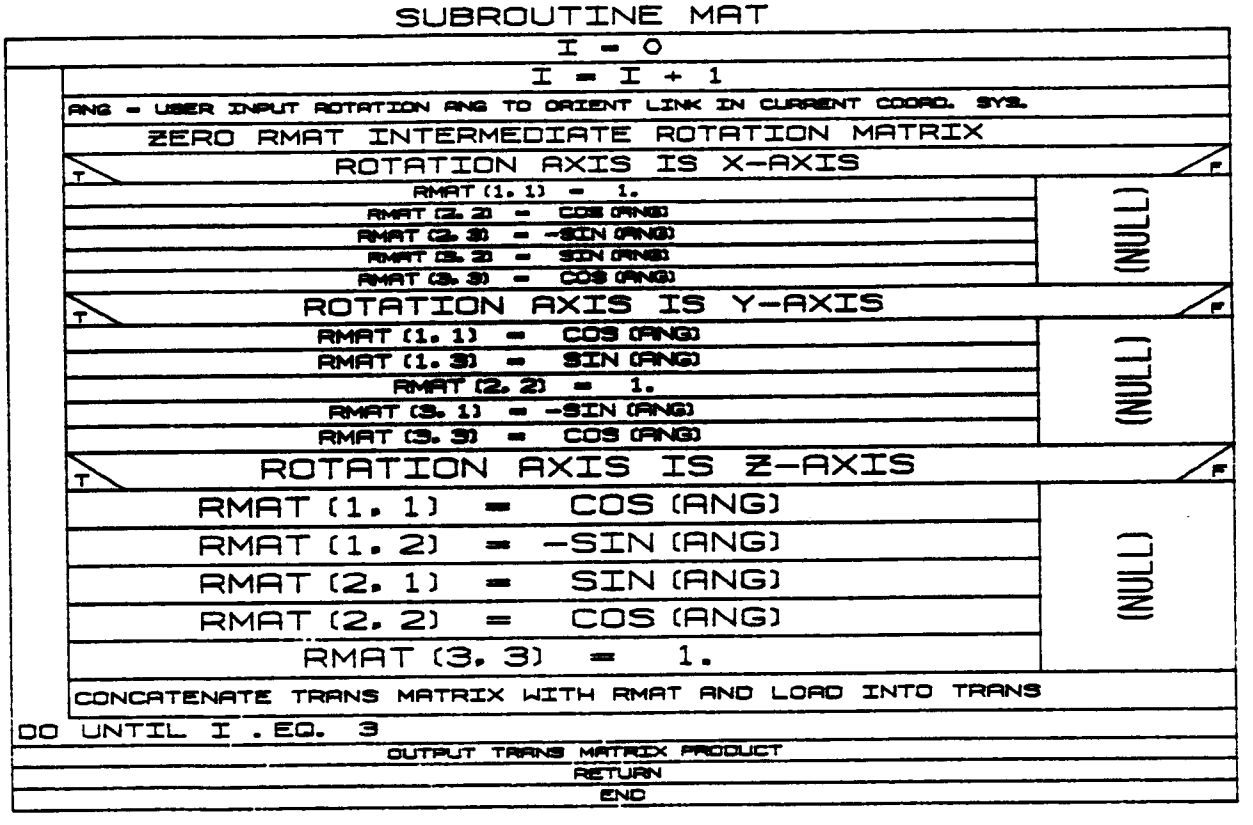
1.3.28 ORIENT

ORIENT is called from most of the build options in INITDRVR, allowing the user to reposition components. The user can input a rotation sequence consisting of rotation axes and angles, and a translation vector to position the origin of the component within the reference coordinate system. MAT is called to compute the total rotation transformation matrix, and MATVEC to transform vectors from the new coordinate system to the reference system. The translation vector is then added to each set of coordinates. The translation and orientation of detailed graphic components are written to a print/save file for archiving the program interaction.


SUBROUTINE ORIENT
ASK USER IF ROTATIONS AND TRANSLATION ARE REQUIRED OR ONLY TRANSLATION
ROTATIONS ARE REQUIRED
T
F
PROMPT USER FOR ROTATION AXIS
PROMPT USER FOR ROTATION ANGLE
DO UNTIL USER STOP
(NULL)
CALL MAT TO CALCULATE TRANSFORMATION MATRIX
CALL MATVEC TO TRANSFORM LINES OF GRAPHICS REPRESENTATION
PROMPT USER FOR X, Y, AND Z TRANSLATIONS
WRITE TO GRAPHICS SAVE FILE
ADD TRANSLATIONS TO GRAPHICS REPRESENTATION VARIABLES
RETURN
END

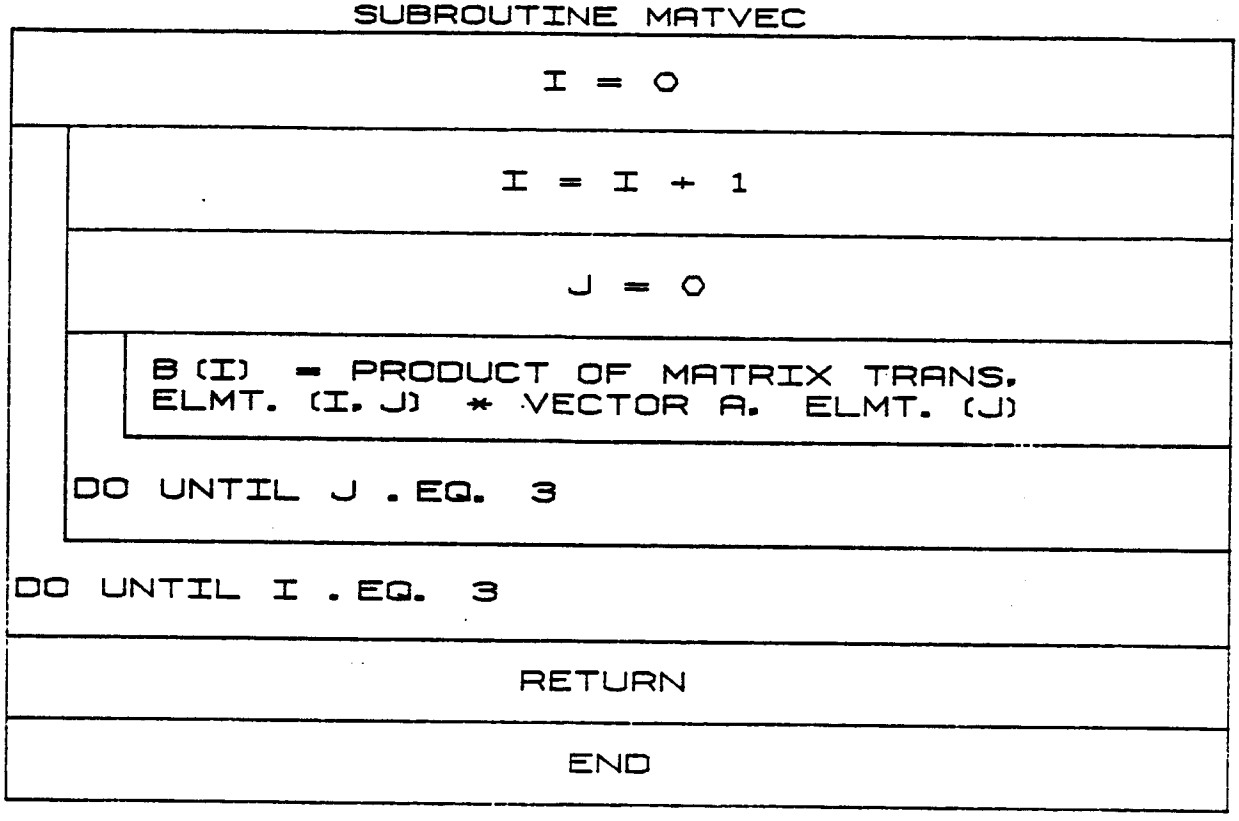
1.3.29 MAT

Subroutine MAT is called during the system definition function to compute the total rotation transformation matrix defined by the input rotation sequence and angles. MAT is called from subroutine ORIENT. The rotation sequence passed to it determines the transformation matrix from the component system to the reference system it calculates. The transpose (inverse) of the normal X,Y and Z-axis rotation matrices are used. For each rotation in the input sequence, the axis rotation matrix is loaded and premultiplied with the current total transformation matrix.



1.3.30 MATVEC

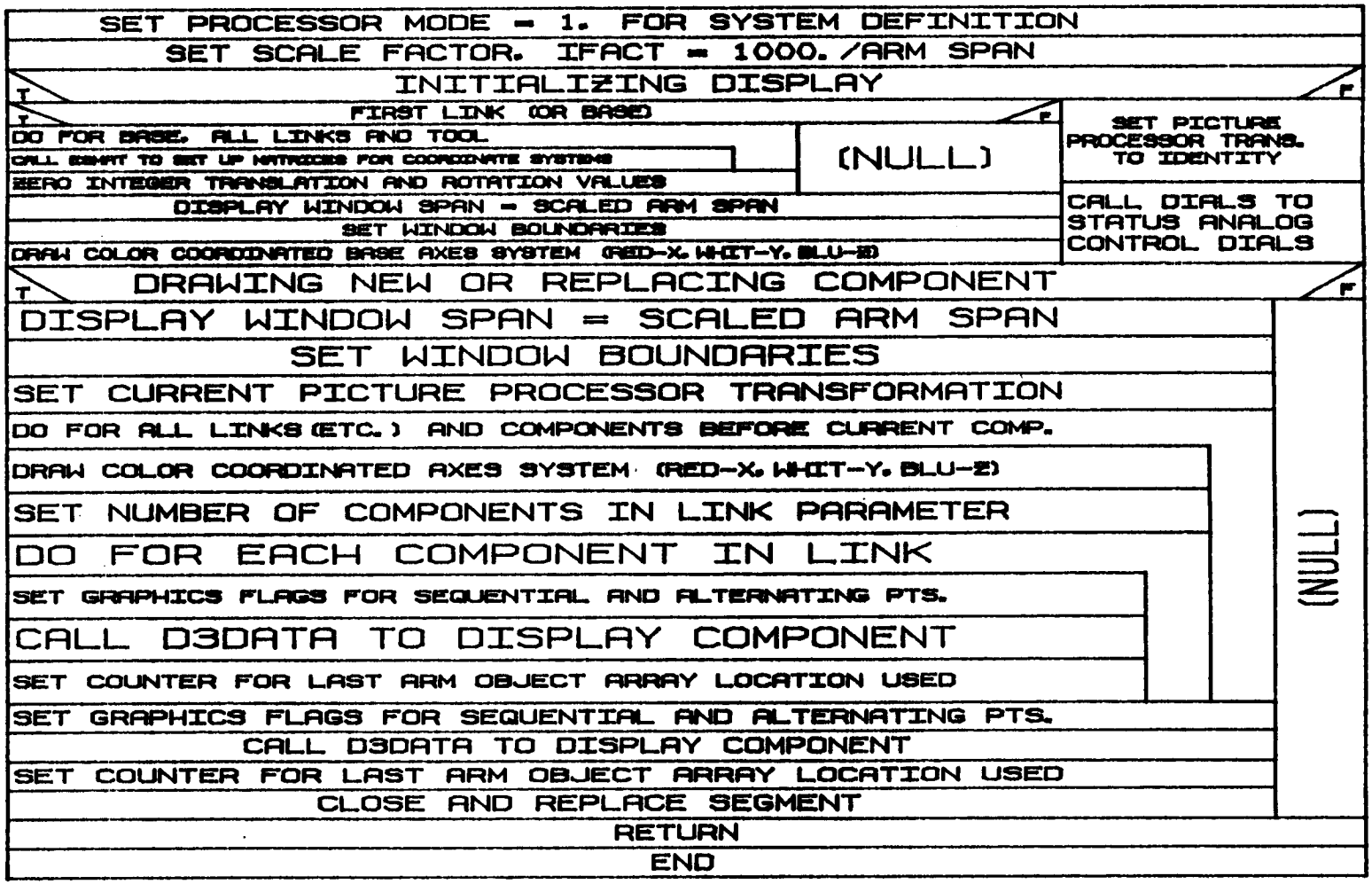
Subroutine MATVEC is called during the system definition function to provide matrix/vector multiplication. The routine is called from ORIENT. The vector A is multiplied by the matrix TRANS to produce output vector B. Note that this matrix/vector multiplication is 3-D only.



1.3.31 DRAW

Subroutine DRAW is called within the system definition function to provide the graphics display during the generation of arm detailed representations. It is called to display each successive component as it is defined. The routine logic is controlled by flag inputs specifying initialization (at which time base/link/tool transformation matrix concatenations to the system are performed), component drawing, or component modification world.

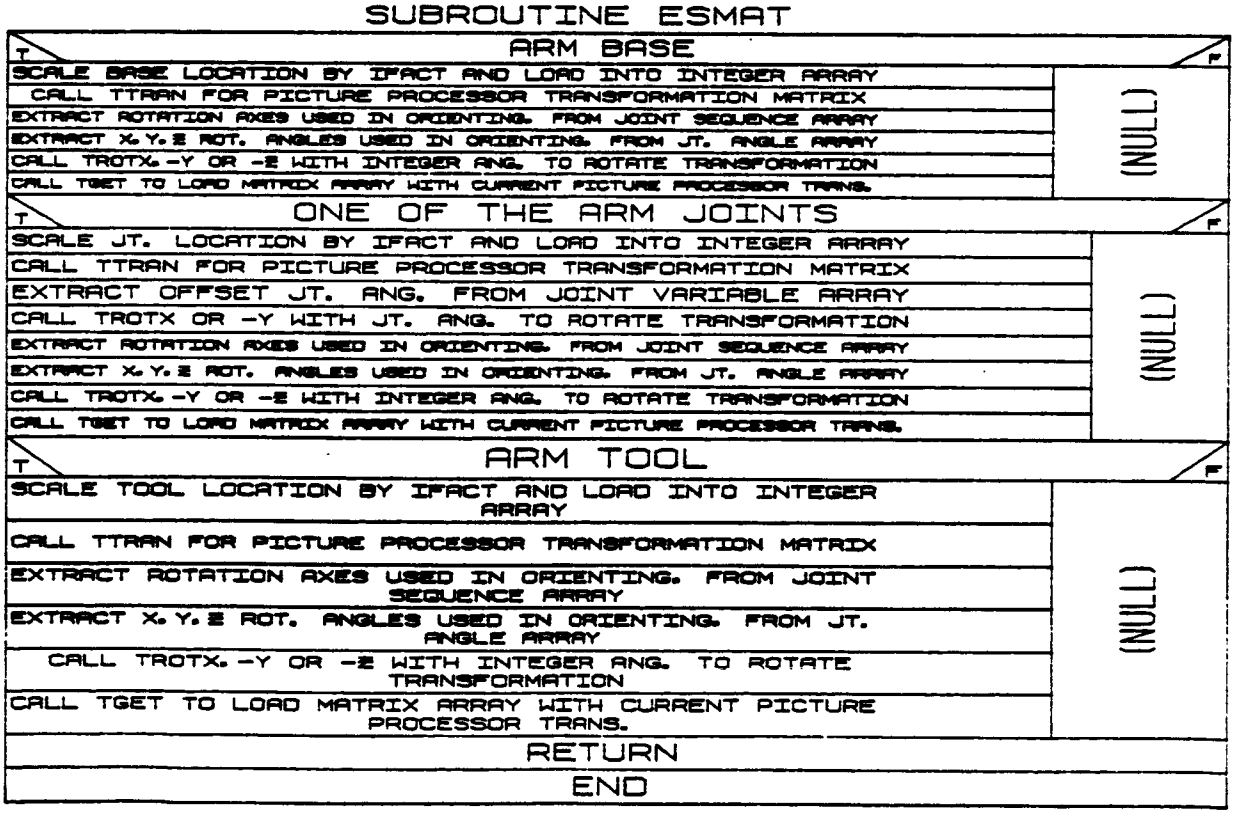
SUBROUTINE DRAW



ORIGINAL PAGE IS
OF POOR QUALITY

1.3.32 ESMAT

Subroutine ESMAT uses Evans and Sutherland graphics routines to construct the required transformation matrices from each system section coordinate system to the graphics coordinate system. Input argument K specifies which system section is under consideration. It is called during execution of the system definition function. It is called from subroutine DRAW to compute the required transformation matrices for each system section. The robotic system has section coordinate systems for the base, each joint/link, and the end-effector. An input value of K=1 indicates the robotic system base. The transformation matrix is composed of a translation matrix based on the base location and rotation matrices constructed using the base orientation parameters. A value of K from 2 to the number of links plus 1 (NJ+1) indicates the (K-1)th joint/link. All transformation matrices from each of the sections to the previous joint (or base, if the current joint is the first joint) are concatenated to form the total transformation matrix to the graphics coordinate system. Each joint transformation matrix is composed of a translation matrix based on the joint position, a rotation matrix based on the initial joint angular displacement, and rotation matrices for joint orientation. A value of K=NJ+2 indicates the end-effector system. The transformation matrix for the end-effector is composed of a translation matrix, and rotation matrices for end-effector orientation. The end-effector location and orientation are specified relative to the coordinate system of the final joint in the system.



1.3.33 DBAS

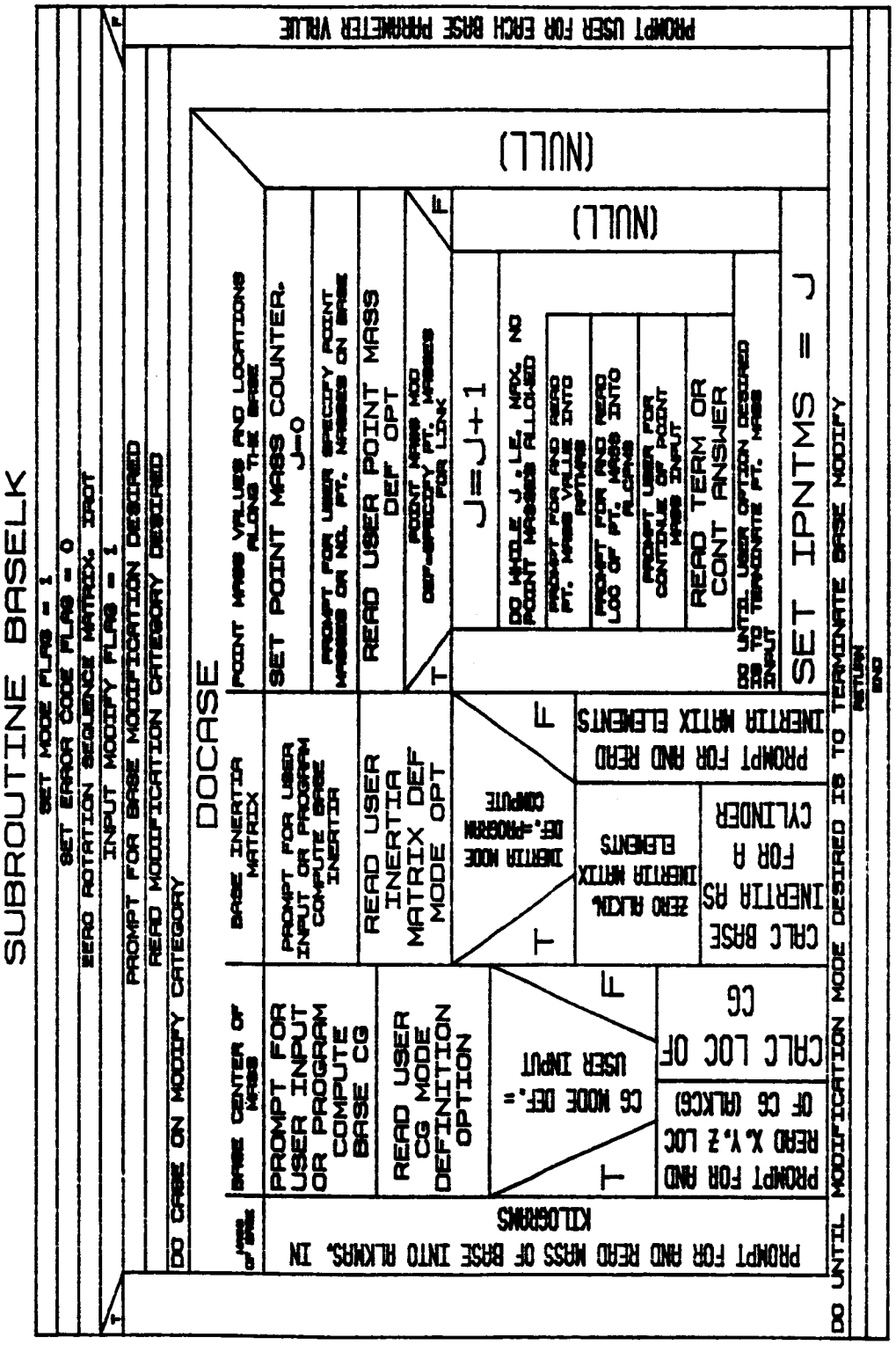
Subroutine DBAS is called within the system definition function during detailed graphics representation generation. For the subroutine, input calling argument IMAN specifies environment, robotic system component, target, or load objects file consideration. Graphics object data IOBJBK are loaded for robotic system components, TGRAFBK is loaded for target components, LGRAFBK is loaded for load components, and ENVTBK is loaded for environment components. The manner in which the data are stored in the COMMON blocks is dictated by the data format used in Evans and Sutherland graphics routine D3DATA.

SUBROUTINE DBAS
FIRST COMPONENT IN DISPLAY SYSTEM
ICOUNT = 0.
(NULL)
DOCASE
DEFINING ROBOT ARM
SET SCALE FACTOR = 1000./ARM SPAN
SET NUMBER OF COMPONENTS IN CURRENT LINK PARAMETER
SET START LOCATION IN ARM OBJECT ARRAY FOR CURRENT COMPONENT
LOAD ARM OBJECT ARRAY, ELEMENT 1+ICOUNT, WITH NUM. SEQUENTIAL PTS.
LOAD ARM OBJ. ARRAY, ELEMENT 2+ THRU 3+ ICOUNT, W/ LINE CONNECT FLAGS
ICOUNT = ICOUNT + 3
DO FOR EACH SEQUENTIAL PT.
LOAD ARM OBJ. ARR., ELMNTS. 1+ THRU 3+ ICOUNT, W/ SCALED ARM OBJ. DATA
ICOUNT = ICOUNT + 3
LOAD ARM OBJ. ARRAY FOR GRAPHICS FLAGS FOR ALTERNATING POINTS
ICOUNT = ICOUNT + 3
DO FOR EACH ALTERNATING PT.
LOAD ARM OBJ. ARR., ELMNTS. 1+ THRU 3+ ICOUNT, W/ SCALED ARM OBJ. DATA
ICOUNT = ICOUNT + 3
SET COUNTER FOR LAST ARM OBJECT ARRAY LOCATION USED
RETURN
END
DEFINING ENVIRONMENT
SET NUMBER OF COMPONENTS IN ENV. PARAMETER
SET START LOCATION IN ENV. OBJECT ARRAY FOR CURRENT COMPONENT
LOAD ENV. OBJ. ARRAY FOR SEQ. AND ALTERN. GRAPHICS PTS.
SET COUNTER FOR LAST OBJECT ARRAY LOCATION USED
DEFINING LOADS
SET SCALE FACTOR = 1000./LOAD SPAN
SET NUMBER OF COMPONENTS IN LOAD PARAMETER
SET START LOCATION IN LOAD OBJECT ARRAY FOR CURRENT COMPONENT
LOAD LOAD OBJ. ARRAY FOR SEQ. AND ALTERN. GRAPHICS PTS.
SET COUNTER FOR LAST LOAD OBJECT ARRAY LOCATION USED
SET SCALE FACTOR = 1000./TARGET SPAN
SET NUMBER OF COMPONENTS IN TARG. PARAMETER
SET START LOCATION IN TARG. OBJECT ARRAY FOR CURRENT COMPONENT
LOAD TARG. OBJ. ARRAY FOR SEQ. AND ALTERN. GRAPHICS PTS.
SET COUNTER FOR LAST TARG. OBJECT ARRAY LOCATION USED

1.3.34 BASELK

In the create mode of BASELK, the user is prompted for the base mass, the location of the base center of mass in the base coordinate system, the base inertia matrix relative to the centroid, the point mass values and locations if desired.

ORIGINAL PAGE IS
OF POOR QUALITY



1.3.35 RCTSTR

Subroutine RCTSTR is called within the system definition function during detailed graphics representation generation for the environment, manipulator, load, or target objects. If the requested component is a rectangular cross-section beam (ISHAPE=7), it is called to compute data points for the graphics routine.

SUBROUTINE RCTSTR

PROMPT FOR RECTANGLE +/−Y SIDE LENGTH, RCTL
PROMPT FOR RECTANGLE +/−Z SIDE LENGTH, RCTL1
PROMPT FOR SEGMENT LENGTH AND NUM. SEGS.
WRITE TO GRAPHICS SAVE FILE IF OPTED
SET NUMBER OF SIDES = 4
SET NUM. LINES AND MODE DRAWING FLAG FOR CONNECTING SEQ. PTS.
DO FOR EACH SEGMENT OF RECT. STRUCTURE. FROM 1 TO NUM. SIDES + 1
SET OFFSET AND X = (CURRENT SEG. NUM. − 1) * (SEGM. LENGTH)
LOAD ARRAY X, Y, Z VALUE. ELEMENT 1+OFFSET. WITH X. RCTL. RCTL1
LOAD ARRAY X, Y, Z VALUE. ELEMENT 2+OFFSET. WITH X. −RCTL. RCTL1
LOAD ARRAY X, Y, Z VALUE. ELEMENT 3+OFFSET. WITH X. −RCTL. −RCTL1
LOAD ARRAY X, Y, Z VALUE. ELEMENT 4+OFFSET. WITH X. RCTL. −RCTL1
LOAD ARRAY X, Y, Z VALUES. ELEMENT 5+OFFSET. WITH ELEMT. 1+OFFSET VALUES
SET ICOUNT = NUMBER LINES ALREADY DRAWN SEQ.
SET NUM. LINES LINE DRAWING MODE FLAG FOR CONNECTING ALT. PTS.
SET ANGLE OFFSET PARAMETER TO (SEG. NUM.) * (NUM. SIDES + 1)
DO FOR EACH RECTANGLE SIDE FROM 2 TO NUM. SIDES
INCREMENT I
ICOUNT = ICOUNT + 1
LOAD ARRAY. ELEMENT ICOUNT. WITH X, Y, Z FROM ARRAY ELEMENT I
ICOUNT = ICOUNT + 1
LOAD ARRAY. ELEMENT ICOUNT. WITH X, Y, Z FROM ARRAY ELEMENT I+OFFSET
RETURN
END

1.3.36 CADOBJ

Subroutine CADOBJ is called within the system definition function during detailed graphics representation generation for the environment, manipulator, load, or target objects. If the requested component is a CAD/CAM object (ISHAPE=11), it is called to compute data points for the graphics routine.

SUBROUTINE CADOBJ

PROMPT USER FOR CHOICE TO READ COMPONENT FILE OR INPUT INTERACTIVELY

READING FILE

T

PROMPT USER FOR COMPONENT FILENAME

OPEN COMPONENT FILE

READ NUMBER OF DATA POINTS VALUE FROM FILE

READ RECORD FROM FILE CONTAINING X, Y, Z VALUES OF PT. INTO DATA

DO UNTIL ALL POINTS HAVE BEEN READ

CLOSE COMPONENT FILE

F

PROMPT FOR NUMBER OF POINTS TO BE INPUT FOR OBSTCL PERIMETER

SET NUM. SIDES = NUM. PTS. TO BE INPUT

KOUNT = 0

READ USER INPUT X, Y AND Z REAL VALUES FOR CURRENT POINT INTO DATA

KOUNT = KOUNT + 1

DO UNTIL ALL POINTS HAVE BEEN INPUT

KOUNT = NUMBER PTS. READ + 1

SET LAST DATA ARRAY ELEMENT, WITH X, Y, Z VALUES READ FOR ELEMENT 1

PROMPT USER FOR SCALE FACTOR TO BE USED

SET LINE DRAWING MODE FLAG FOR CONNECTING SEQUENTIAL POINTS = KOUNT

SET LINE DRAWING MODE FLAG FOR CONNECTING ALTERNATING POINTS = 0

DO FOR EACH POINT TO BE CONNECTED SEQUENTIALLY

LOAD ARRAY X, Y, Z, ELEMENT KOUNT, WITH DATA X, Y, Z* SCAL. FACT

RETURN

END

1.3.37 TARG

Subroutine TARG is called within the system definition function during detailed graphics representation generation for target objects. If the requested component is a 4-dot target object (ISHAPE=12), it is called to compute data points for the graphics routine.

SUBROUTINE TARG

PROMPT FOR TARGET +/- LENGTH IN X

PROMPT FOR TARGET +/- WIDTH IN Y

PROMPT FOR TARGET FOUR SIDED SHAPE

ISHAPE = 9. FOR RECTANGLE TARGET

T	F
PROMPT FOR RECTANGLE +/- BASE DIMENSION	PROMPT FOR TRAPEZOID FIRST BASE DIMENSION PROMPT FOR TRAPEZOID SECOND BASE DIMENSION

SET NUMBER OF SIDES = 8

NUMBER OF LINES TO DRAW SEQUENTIALLY=0

SET LINE DRAWING MODE FLAG FOR CONNECTING ALTERNATING POINTS

DO FOR EACH RECT./TRAP. END

FIRST END

T	F
SET OFFSET PARAMETER = 0.	SET OFFSET PARAMETER = NUM. SIDES + 1
SET X = - RECT./TRAP. LENGTH	SET X = + RECT./TRAP. LENGTH
SET Z = FIRST RECT./TRAP. BASE VALUE	SET Z = SECOND RECT./TRAP. BASE VALUE

LOAD ARRAY X, Y, Z, ELEMENT 1+OFF. AND 2+OFF., WITH X+/-(.1)X, WIDTH AND 8

ISHAPE = 2. FOR SYMMETRIC TRAP.

T	F
LOAD ARRAY Z VALUE, ELEMENT 1+OFF. AND 2+OFF., WITH Z	(NULL)

ISHAPE = 3. FOR NONSYMMETRIC TRAP.

T	F
LOAD ARRAY Z VALUE, ELEMENT 1+OFF. AND 2+OFF., WITH 0.	(NULL)

LOAD ARRAY X, Y, Z, ELEMENT 3+OFF. AND 4+OFF., WITH X, WIDTH+/-(.1)YO AND

ISHAPE = 2. FOR SYMMETRIC TRAP.

T	F
LOAD ARRAY Z VALUE, ELEMENT 3+OFF. AND 4+OFF., WITH Z	(NULL)

ISHAPE = 3. FOR NONSYMMETRIC TRAP.

T	F
LOAD ARRAY Z VALUE, ELEMENT 3+OFF. AND 4+OFF., WITH 0.	(NULL)

LOAD ARRAY X, Y, Z, ELEMENT 5+OFF. AND 6+OFF., WITH X+/-(.1)X, WIDTH AND -

ISHAPE = 2. FOR SYMMETRIC TRAP.

T	F
LOAD ARRAY Z VALUE, ELEMENT 5+OFF. AND 6+OFF., WITH -Z	(NULL)

ISHAPE = 3. FOR NONSYMMETRIC TRAP.

T	F
LOAD ARRAY Z VALUE, ELEMENT 5+OFF. AND 6+OFF., WITH -Z	(NULL)

LOAD ARRAY X, Y, Z, ELEMENT 7+OFF. AND 8+OFF., WITH X, WIDTH+/-(.1)YO AND -

ISHAPE = 2. FOR SYMMETRIC TRAP.

T	F
LOAD ARRAY Z VALUE, ELEMENT 7+OFF. AND 8+OFF., WITH -Z	(NULL)

ISHAPE = 3. FOR NONSYMMETRIC TRAP.

T	F
LOAD ARRAY Z VALUE, ELEMENT 7+OFF. AND 8+OFF., WITH -Z	(NULL)

SET NUMBER OF LINES TO DRAW ALTERNATELY

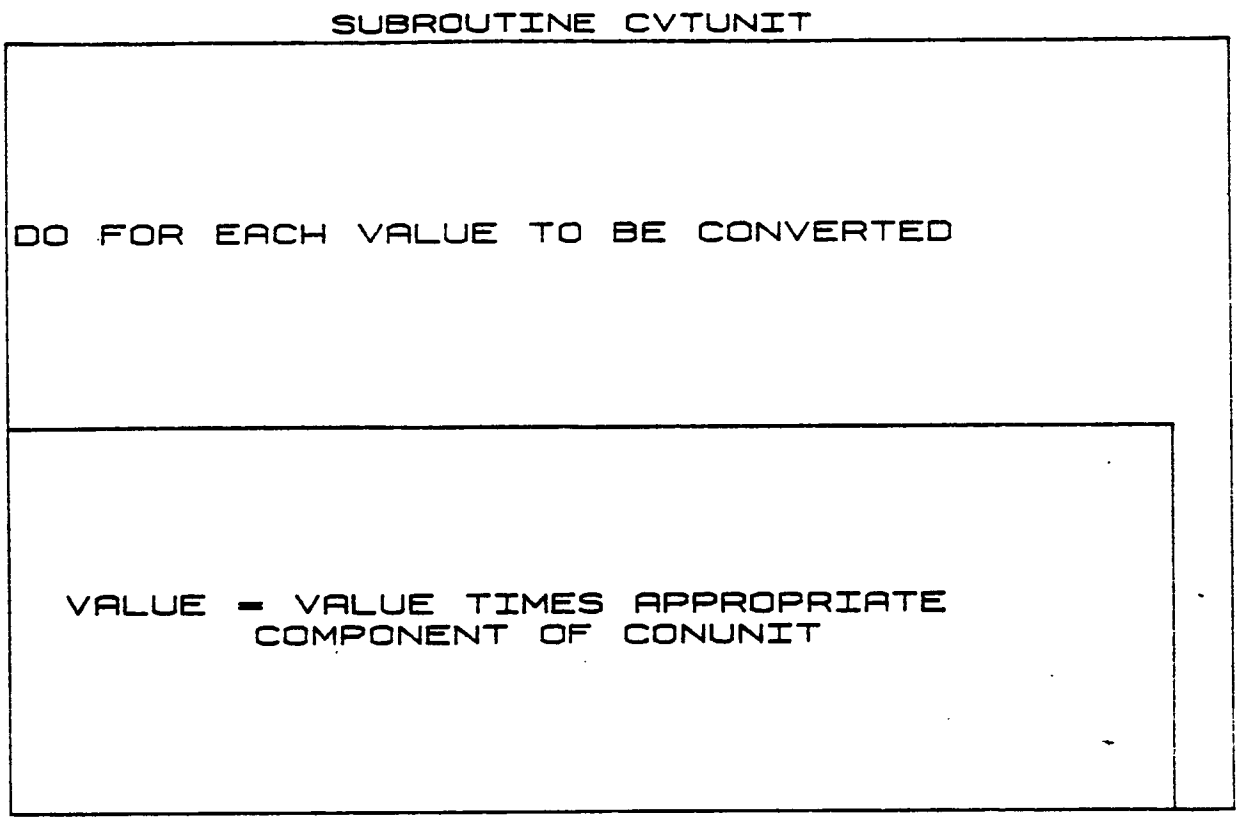
RETURN

END

ORIGINAL PAGE IS
OF POOR QUALITY

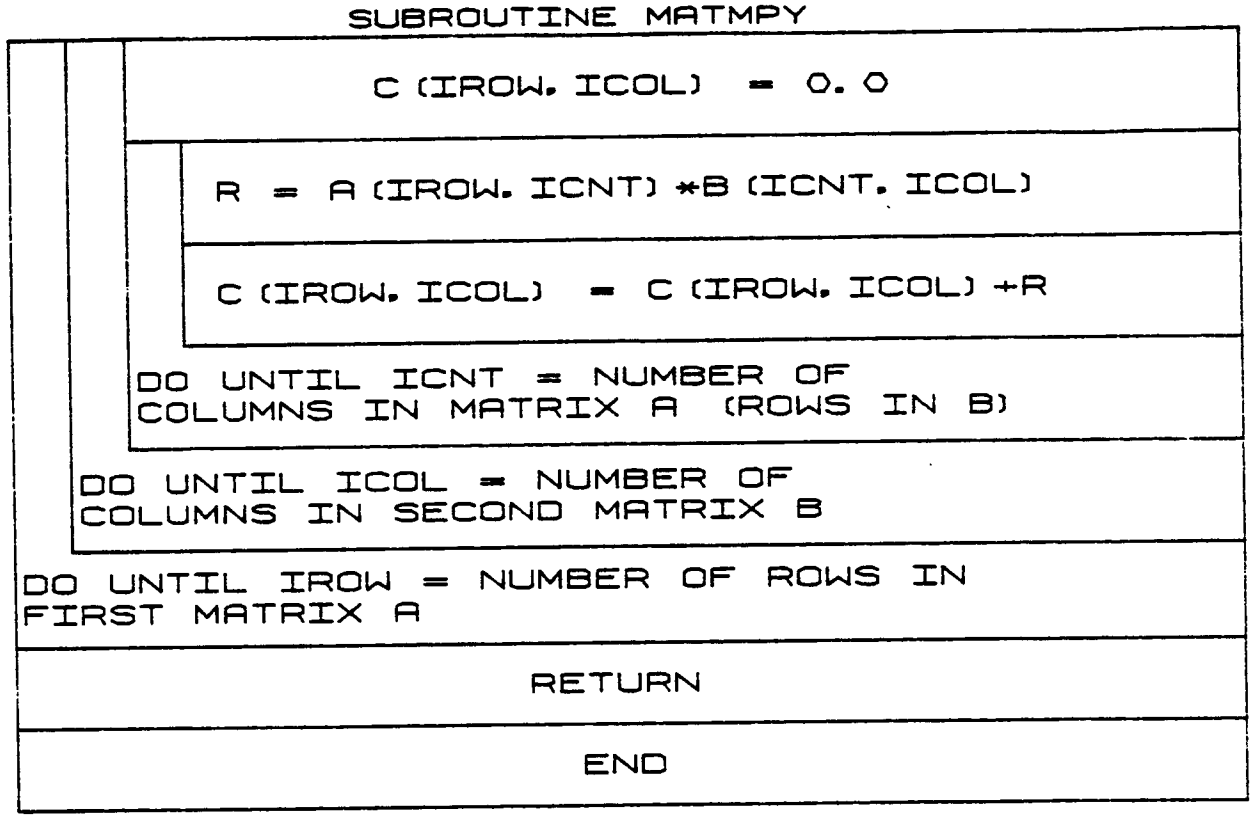
1.4.1 CVTUNIT

Subroutine CVTUNIT is responsible for the conversion of input data from I/O units to internal mathematical units. Each data value VAL is multiplied by CONUNIT(IDIM) and replaced in VAL.



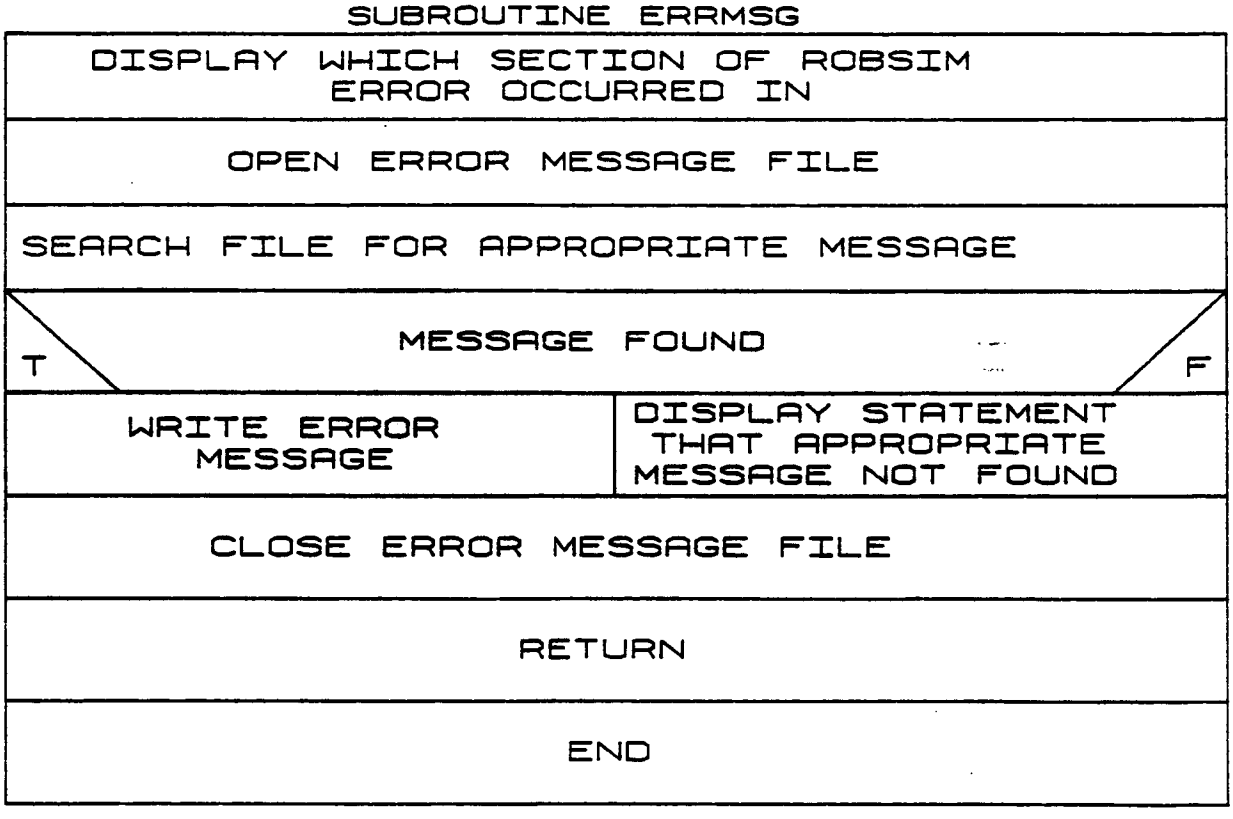
1.4.2 MATMPY

Subroutine MATMPY performs the multiplication of two matrices, AB=C, where A has I rows and J columns, the dimension of B is JxK and C is IxK. The matrices and their sizes are passed to subroutine as calling arguments.



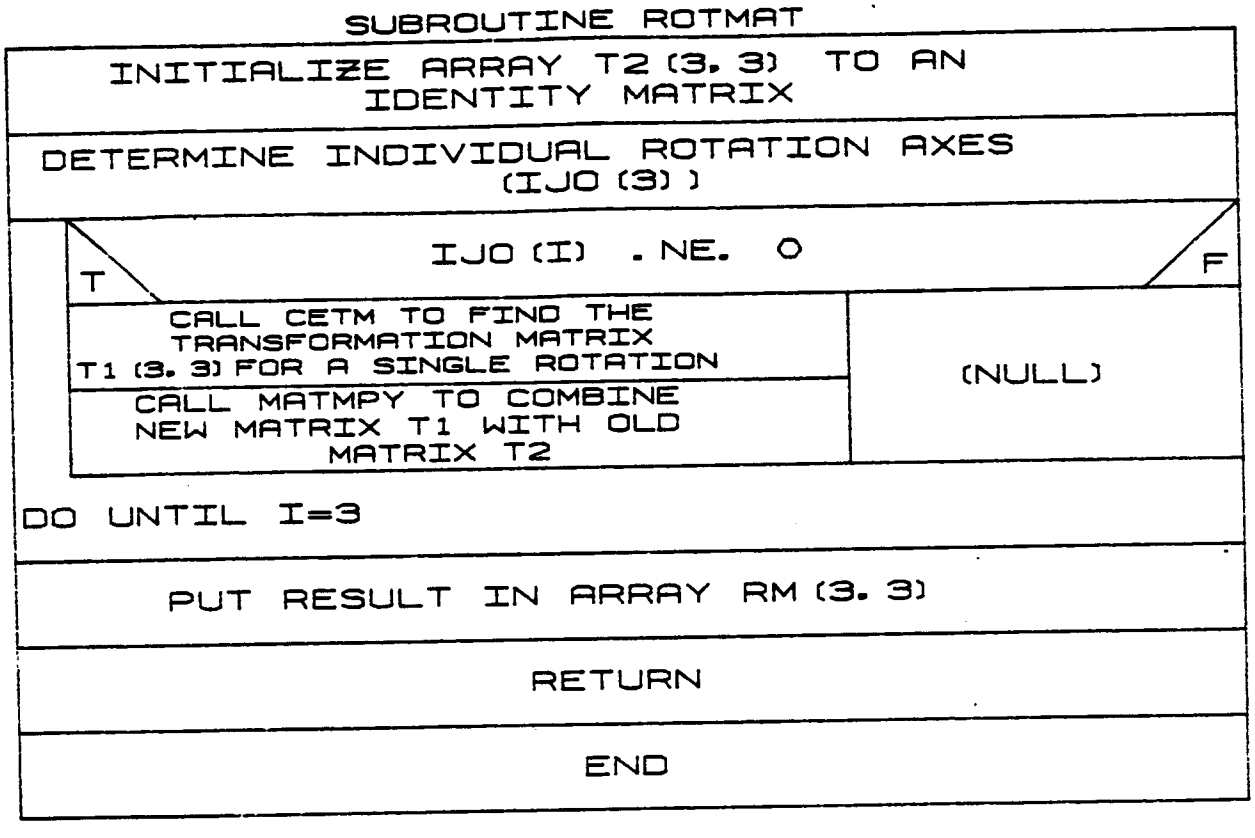
1.4.3 ERRMSG

Subroutine ERRMSG is called when certain errors occur during ROBSIM execution. The routine first displays the current operating mode (i.e., system definition, analysis, or postprocessing). The routine searches the file ERROR.DAT for an error message corresponding to the error number passed to it. The message is typed at the terminal and execution returns to the calling routine, from which it continues or terminates depending on whether the error is fatal.



1.4.4 ROTMAT

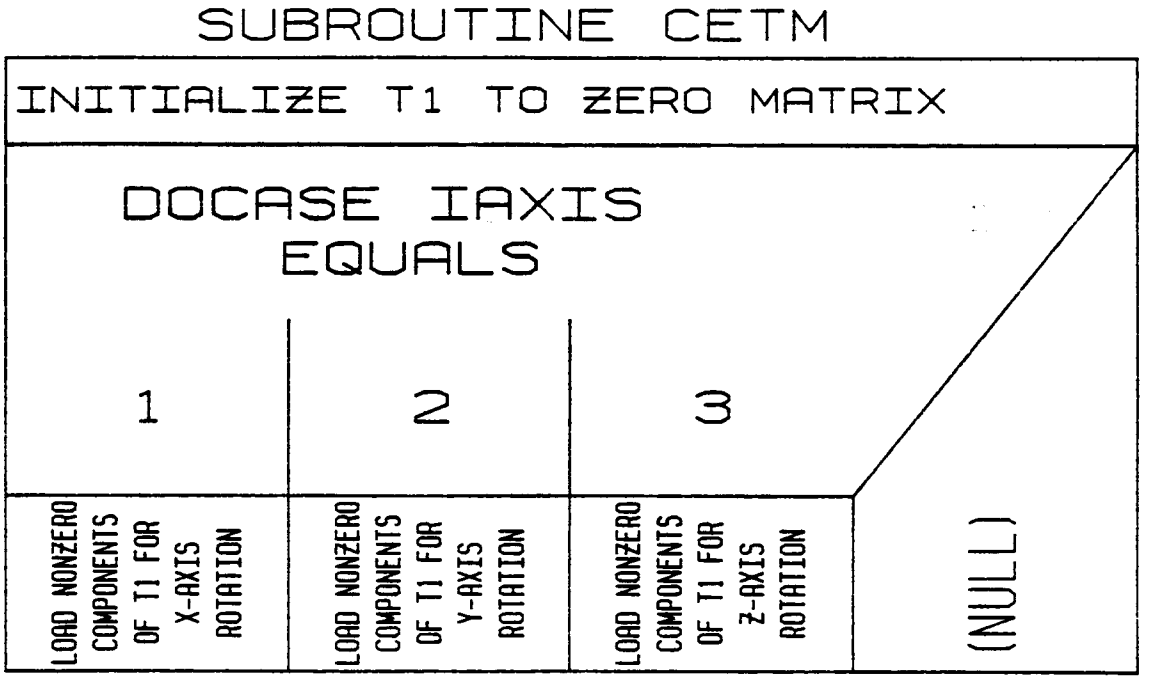
ROTMAT computes a rotation matrix from a sequence of up to three rotations about coordinate axes. It decomposes the input calling argument JSEQ into three successive rotation axes, computes each corresponding rotation matrix from the specified angles of rotation and combines these successively to find the overall rotation matrix.



1.4.5 CETM

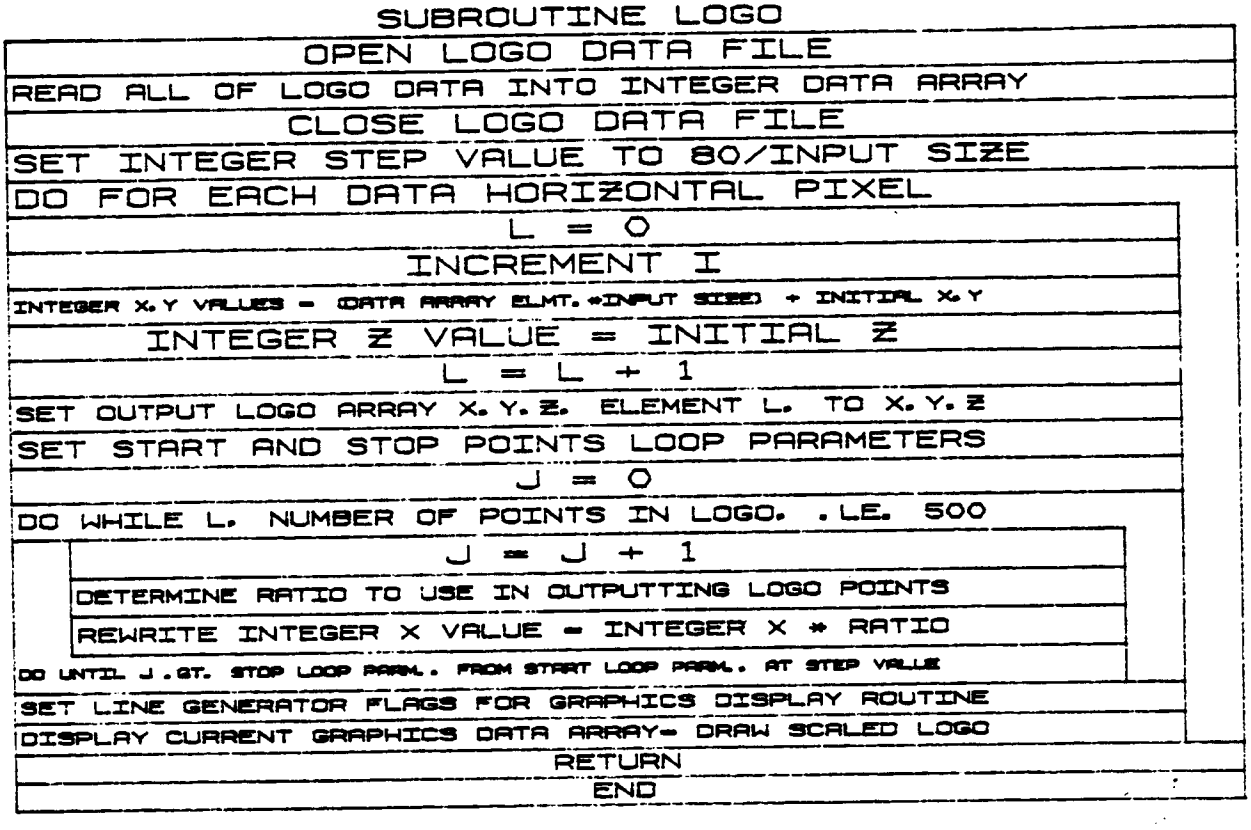
Subroutine CETM calculates a transformation matrix for a specific input axis of rotation and rotation angle by the use of appropriate direction cosines matrix. The calling argument input is:

Symbol	Type	Dim.	Definition
IAXIS	I*4	1	Rotation axis for joint orientation = 1, Rotation about x-axis = 2, Rotation about y-axis = 3, Rotation about z-axis
TH	R*4	1	Rotation angle for joint orientation



1.4.6 LOGO

The LOGO routine calculates data points required to output the Martin Marietta logo, and displays it on the robotic simulation E&S graphics display. It extracts from the data points file LOGO.DAT, the Martin Marietta company logo, scales and displays the logo for the robotic graphics simulation.



1.4.7 CRPD

Subroutine CRPD computes the cross-product of two vectors A and B, each containing three components. The result is put into the vector C.

SUBROUTINE CRPD

```
C(1) = A(2)*B(3) - A(3)*B(2)

C(2) = A(3)*B(1) - A(1)*B(3)

C(3) = A(1)*B(2) - A(2)*B(1)

RETURN

END
```

The Analysis Tools Function

The program SIMDRVR is the analysis tools function driver. The following set of routine functional descriptions and VCLRs (visual control logic representations) are the modules found in the analysis tools function of ROBSIM.

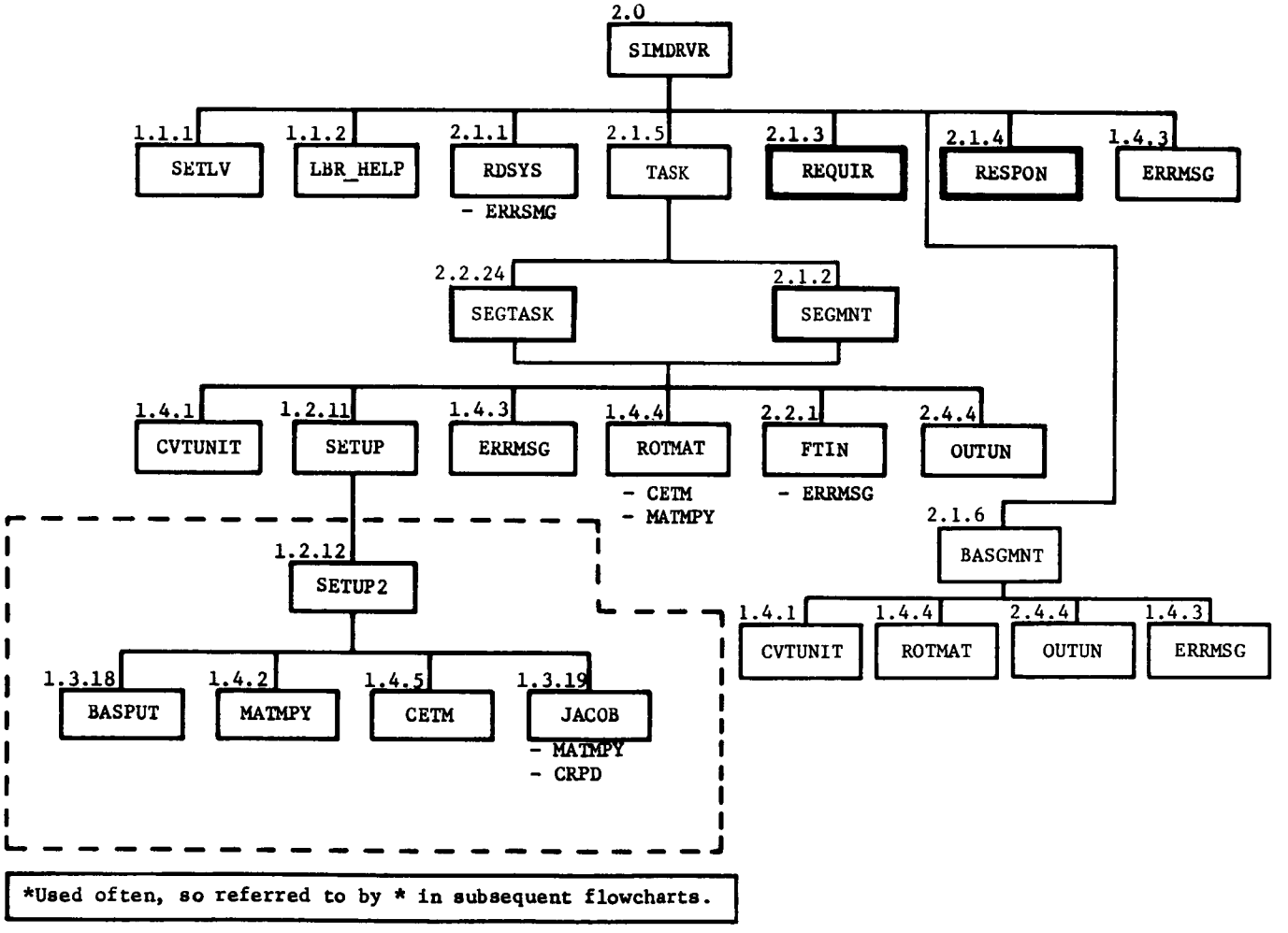
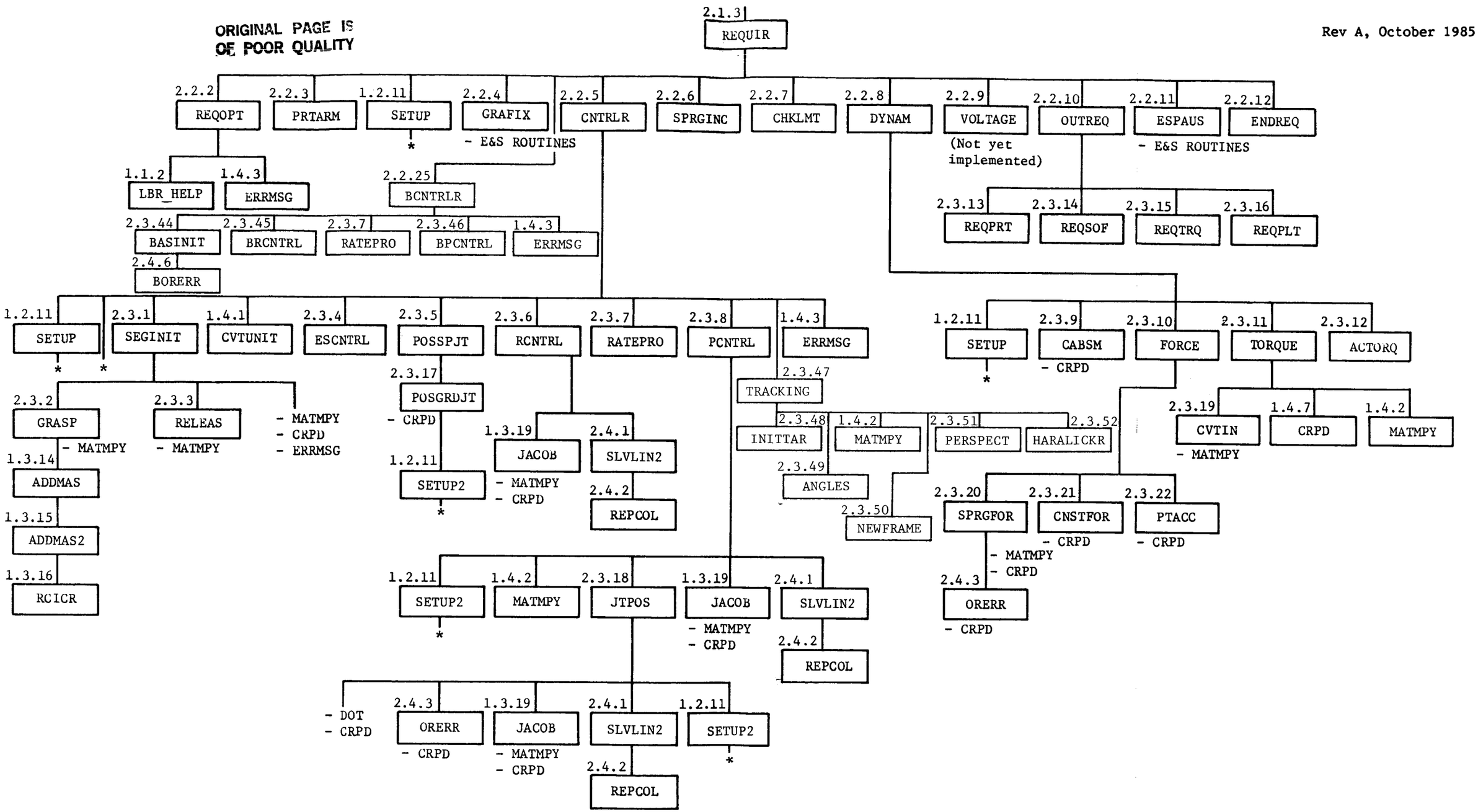


Figure B-8. - Functional block diagram for SIMDRVR.

PRECEDING PAGE BLANK NOT FILMED

ORIGINAL PAGE IS
OF POOR QUALITY

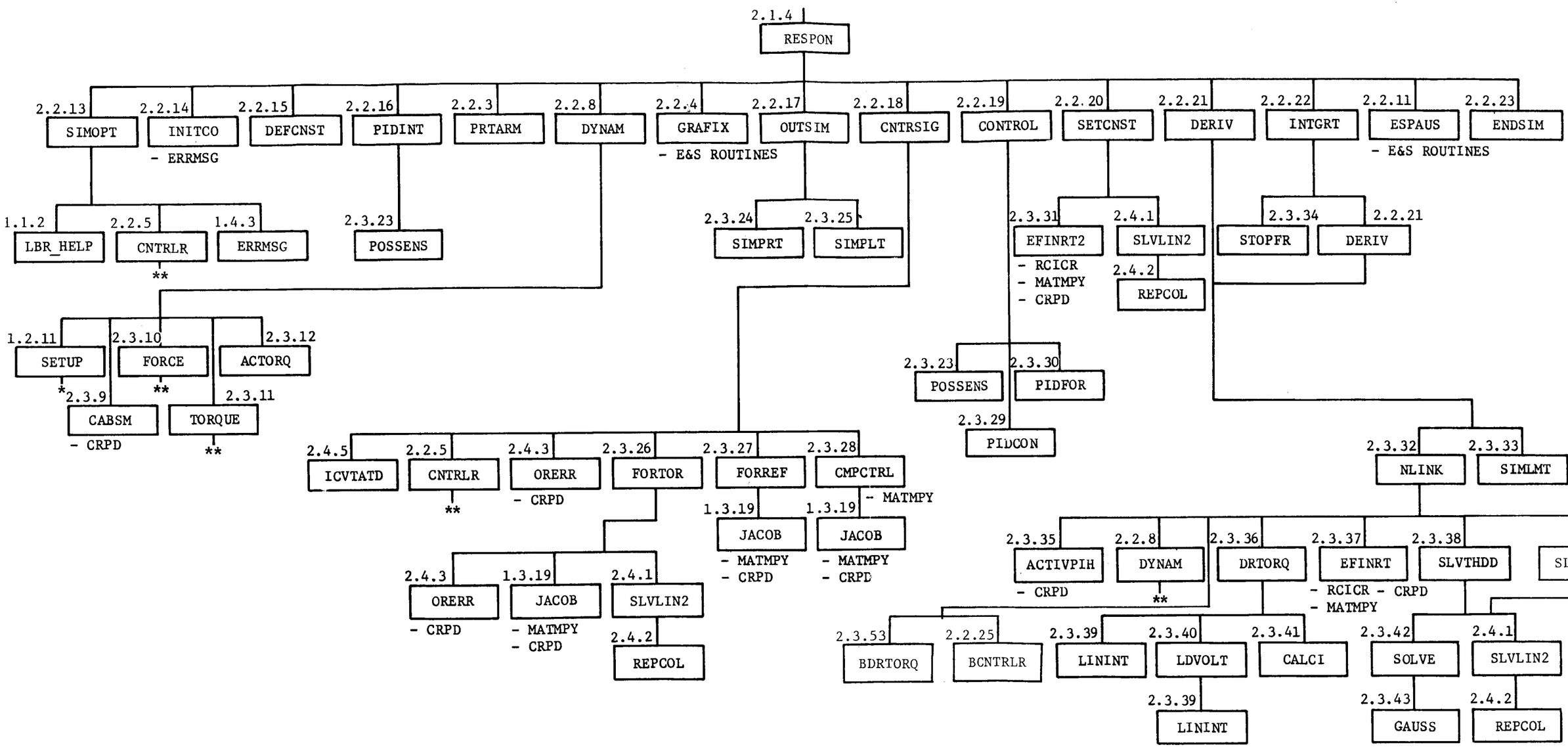


*Defined in previous flowchart SIMDRVR.

Figure B-8. - (cont)

FOLDOUT FRAME
FOLDOUT FRAME

ORIGINAL PAGE IS
OF POOR QUALITY



*Defined in flowchart SIMDRVR.

**These subroutines defined in previous flowchart under REQUIR.

FOLDOUT FRAME

Figure B-8. - (concl)

FOLDOUT FRAME

TABLE B-VI. - PROGRAMS EMPLOYED IN SIMDRVR

2.0	SIMDRVR	2.3.7	RATEPRO	2.3.48	INITTAR
		2.3.8	PCNTRL	2.3.49	ANGLES
2.1.1	RDSYS	2.3.9	CABSIM	2.3.50	NEWFRAME
2.1.2	SGMNT	2.3.10	FORCE	2.3.51	PERSPECT
2.1.3	REQUIR	2.3.11	TORQUE	2.3.52	HARALICKR
2.1.4	RESPON	2.3.12	ACTORQ	2.3.53	BDRTORQ
2.1.5	TASK	2.3.13	REQPRT	2.3.54	SLVMBAS
2.1.6	BASGMNT	2.3.14	REQSOF		
		2.3.15	REQTRQ	2.4.1	SLVLIN2
2.2.1	FTIN	2.3.16	REQPLT	2.4.2	REPCOL
2.2.2	REQOPT	2.3.17	POSGRDJT	2.4.3	ORERR
2.2.3	PRTARM	2.3.18	JTPOS	2.4.4	OUTUN
2.2.4	GRAFIX	2.3.19	CVTIN	2.4.5	ICVTATD
2.2.5	CNTRLR	2.3.20	SPRGFOR	2.4.6	BORERR
2.2.6	SPRGINC	2.3.21	CNSTFOR		
2.2.7	CHKLMT	2.3.22	PTACC		
2.2.8	DYNAM	2.3.23	POSSENS		
2.2.9	VOLTAGE	2.3.24	SIMPRT		
2.2.10	OUTREQ	2.3.25	SIMPLT		
2.2.11	ESPAUS	2.3.26	FORTOR		
2.2.12	ENDREQ	2.3.27	FORREF		
2.2.13	SIMOPT	2.3.28	CMPCTRL		
2.2.14	INITCO	2.3.29	PIDCON		
2.2.15	DEFCNST	2.3.30	PIDFOR		
2.2.16	PIDINIT	2.3.31	EFINRT2		
2.2.17	OUTSIM	2.3.32	NLINK		
2.2.18	CNTRSIG	2.3.33	SIMLMT		
2.2.19	CONTROL	2.3.34	STOPFR		
2.2.20	SETCNST	2.3.35	ACTIVPIH		
2.2.21	DERIV	2.3.36	DRTORQ		
2.2.22	INTGRT	2.3.37	EFINRT		
2.2.23	ENDSIN	2.3.38	SLVTHDD		
2.2.24	SEGTASK	2.3.39	LININT		
2.2.25	BCNTRLR	2.3.40	LDVOLT		
		2.3.41	CALCI		
2.3.1	SEGINIT	2.3.42	SOLVE		
2.3.2	GRASP	2.3.43	GAUSS		
2.3.3	RELEAS	2.3.44	BASINIT		
2.3.4	ESCNTRL	2.3.45	BRCNTRL		
2.3.5	POSSPJT	2.3.46	BPCNTRL		
2.3.6	RCNTRL	2.3.47	TRACKING		

2.0 SIMDRVR

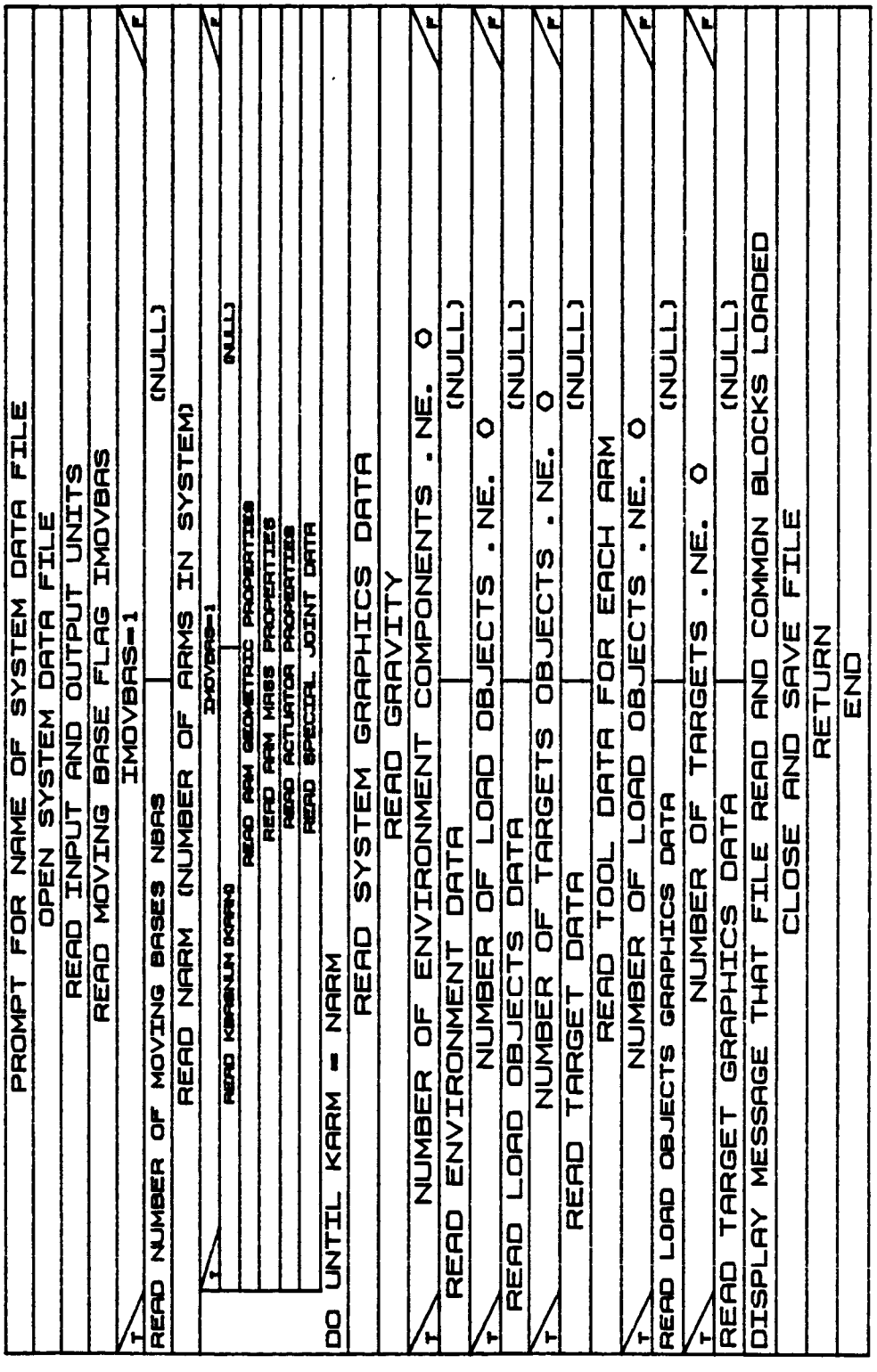
The program SIMDRVR is the analysis tools function driver. It operates in an interactive mode, prompting the user for the analysis option desired: requirements analysis without graphics, requirements analysis with graphics (a display of system motion during program execution), response simulation analysis without graphics, response simulation analysis with graphics, option to set up a base or arm motion program or terminate SIMDRVR execution.

ORIGINAL PAGE IS
OF POOR QUALITY

2.1.1 RDSYS

Subroutine RDSYS is called from SIMDRVR to read in the manipulator system definition data needed to run any of the SIMDRVR analysis options. The routine first prompts the user for the name of the file containing the system's data and then opens that file. If the system includes moving bases it reads the number of bases. Moving base numbers, geometric properties, mass properties, actuator properties and special joint data for each arm are read in, as well as system graphics data and the definition of gravity for the system. If the system contains an environment, the data describing it are read in. If load objects are also to be included, the data describing them are read in. End-effector data and target data are to be read next. After that, the file is closed and saved.

ORIGINAL PAGE IS
OF POOR QUALITY



2.1.2 SGMNT

Subroutine SGMNT allows the user to set up the desired motion profile for a requirements analysis or response simulation run. It is called from SIMDRVR. An existing motion profile file may be read in and modified or the profile may be defined interactively. Motion is specified in one of four ways:

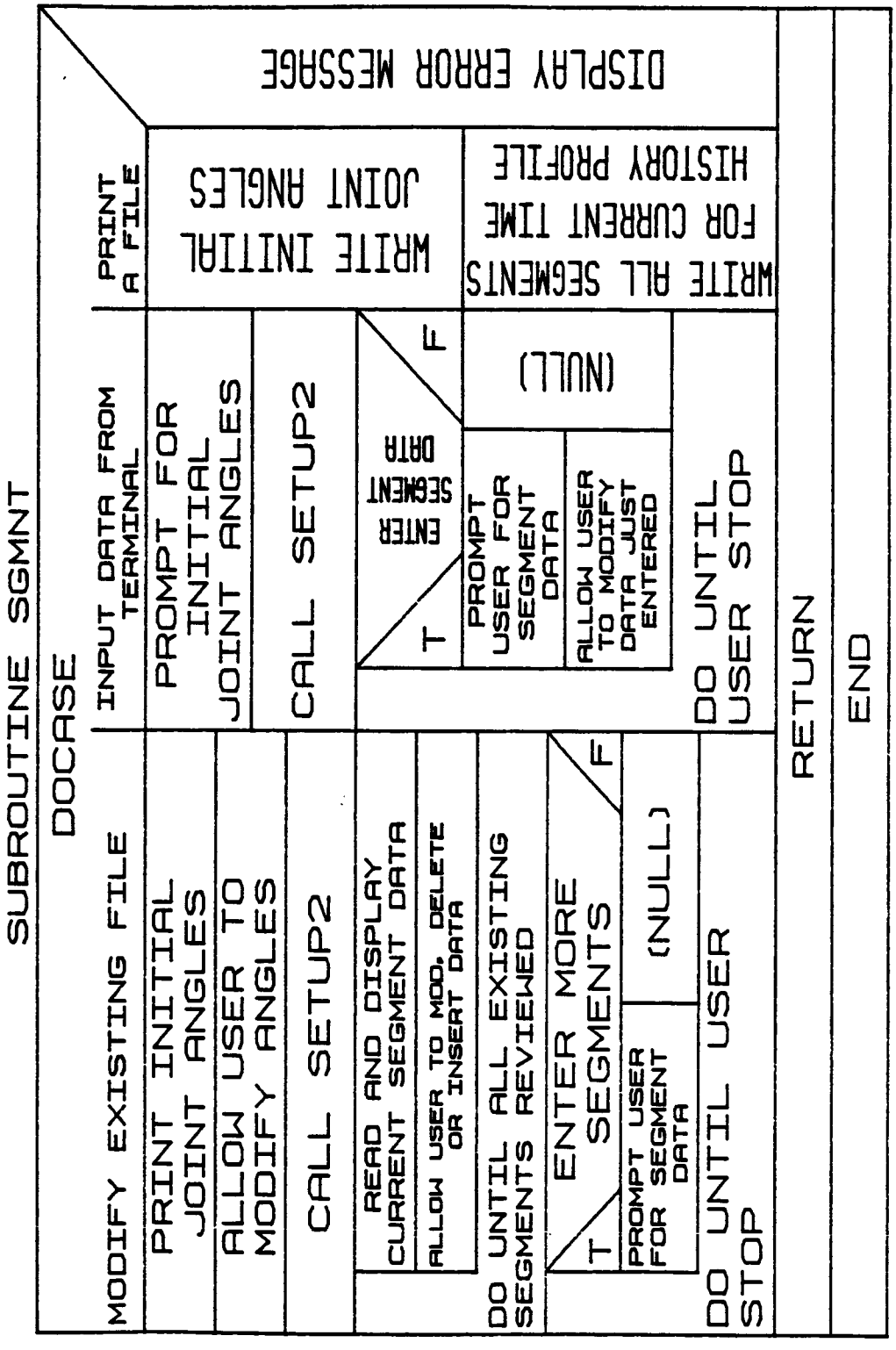
1) Desired position of end-effector;

2) Desired position of each joint;

3) Rate of end-effector movement;

4) Rate of each joint.

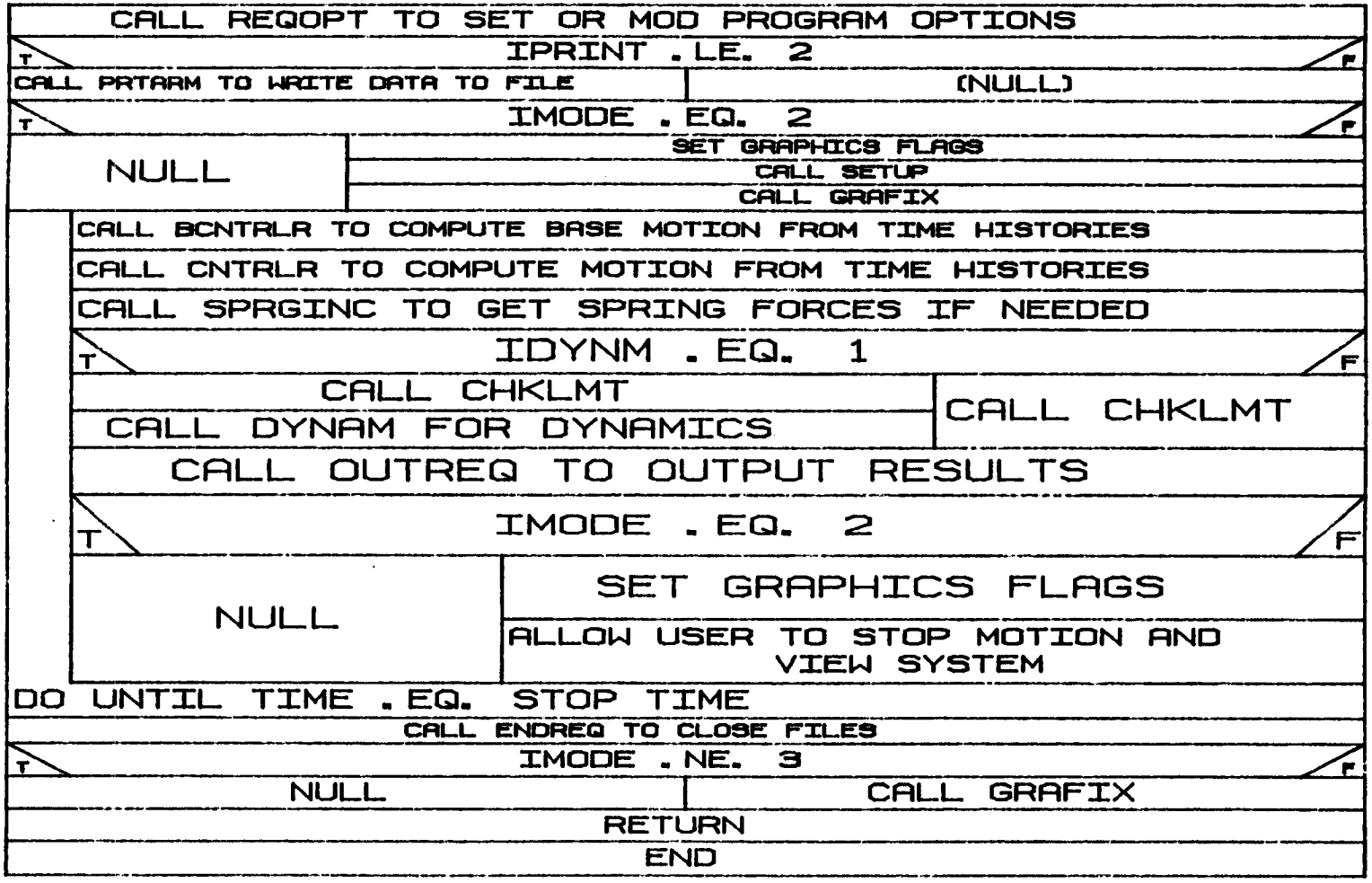
In addition, motion may be specified by having an end-effector-mounted sensor move toward a target. Several nonmotion-type operations such as grasp a load object, release object and wait a given length of time may also be specified.


SUBROUTINE SGMNT
DISPLAY ERROR MESSAGE
DOCASE
MODIFY EXISTING FILE
PRINT INITIAL JOINT ANGLES
ALLOW USER TO MODIFY ANGLES
CALL SETUP2
READ AND DISPLAY CURRENT SEGMENT DATA
ALLOW USER TO MOD. DELETE OR INSERT DATA
DO UNTIL ALL EXISTING SEGMENTS REVIEWED
ENTER MORE SEGMENTS
T
PROMPT USER FOR SEGMENT DATA
F
(NULL)
DO UNTIL USER STOP
INPUT DATA FROM TERMINAL
PROMPT FOR INITIAL JOINT ANGLES
CALL SETUP2
ENTER SEGMENT DATA
T
PROMPT USER FOR SEGMENT DATA
ALLOW USER TO MODIFY DATA JUST ENTERED
F
(NULL)
DO UNTIL USER STOP
PRINT A FILE
WRITE INITIAL JOINT ANGLES
WRITE ALL SEGMENTS FOR CURRENT TIME HISTORY PROFILE
RETURN
END

2.1.3 REQUIR

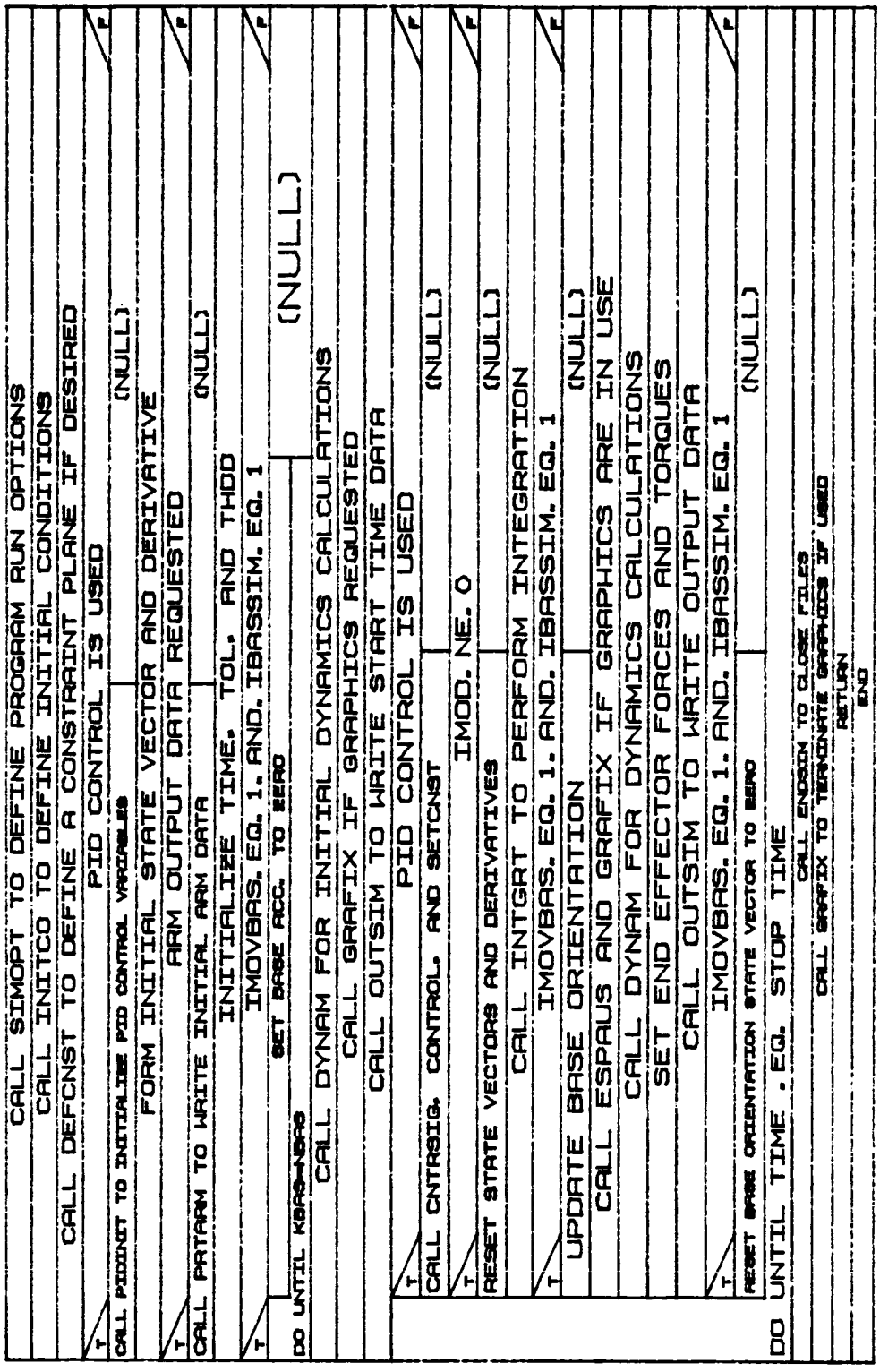
Subroutine REQUIR is called from SIMDRVR and is the routine that controls the execution of any requirements analysis run. It first calls REQOPT to set up program run time options. If requested, PRTARM is called to write a description of the system to an output file. SETUP is called to calculate initial positions. GRAFIX is called if the run is to include graphic displays. The subroutines CNTRLR, SPRGINC, CHKLMT, DYNAM and OUTREQ are called at every increment of a user-defined time loop to calculate the manipulator system's motion, forces and torques, and write these data to an output file. ESPAUS is called when motion is temporarily halted during execution. When the stop time is reached, ENDREQ is called to close any open files.

SUBROUTINE REQUIR



2.1.4 RESPON

Subroutine RESPON is called from SIMDRVR to control the execution of a response simulation run. Run time options and program variables are first initialized. A user-defined time loop is executed to call routines to carry out all the control functions. After execution is completed, ENDSIM is called to close the files.


SUBROUTINE RESPON
CALL SIMOPT TO DEFINE PROGRAM RUN OPTIONS
CALL INITCO TO DEFINE INITIAL CONDITIONS
CALL DEFCNST TO DEFINE A CONSTRAINT PLANE IF DESIRED
PID CONTROL IS USED
T
F
CALL PIDINIT TO INITIALIZE PID CONTROL VARIABLES
(NULL)
FORM INITIAL STATE VECTOR AND DERIVATIVE
ARM OUTPUT DATA REQUESTED
T
F
CALL PRTARM TO WRITE INITIAL ARM DATA
(NULL)
INITIALIZE TIME, TOL, AND THDD
IMOVBAS. EQ. 1. AND. IBASSIM. EQ. 1
T
F
SET BASE ACC. TO ZERO
(NULL)
DO UNTIL KBAS=NBAS
CALL DYNAM FOR INITIAL DYNAMICS CALCULATIONS
CALL GRAFIX IF GRAPHICS REQUESTED
CALL OUTSIM TO WRITE START TIME DATA
PID CONTROL IS USED
T
F
CALL CNTRSIG, CONTROL, AND SETCNST
(NULL)
IMOD. NE. 0
T
F
RESET STATE VECTORS AND DERIVATIVES
(NULL)
CALL INTGRT TO PERFORM INTEGRATION
IMOVBAS. EQ. 1. AND. IBASSIM. EQ. 1
T
F
UPDATE BASE ORIENTATION
(NULL)
CALL ESPAUS AND GRAFIX IF GRAPHICS ARE IN USE
CALL DYNAM FOR DYNAMICS CALCULATIONS
SET END EFFECTOR FORCES AND TORQUES
CALL OUTSIM TO WRITE OUTPUT DATA
IMOVBAS. EQ. 1. AND. IBASSIM. EQ. 1
T
F
RESET BASE ORIENTATION STATE VECTOR TO ZERO
(NULL)
DO UNTIL TIME .EQ. STOP TIME
CALL ENDSIM TO CLOSE FILES
CALL GRAFIX TO TERMINATE GRAPHICS IF USED
RETURN
END

2.1.5 TASK

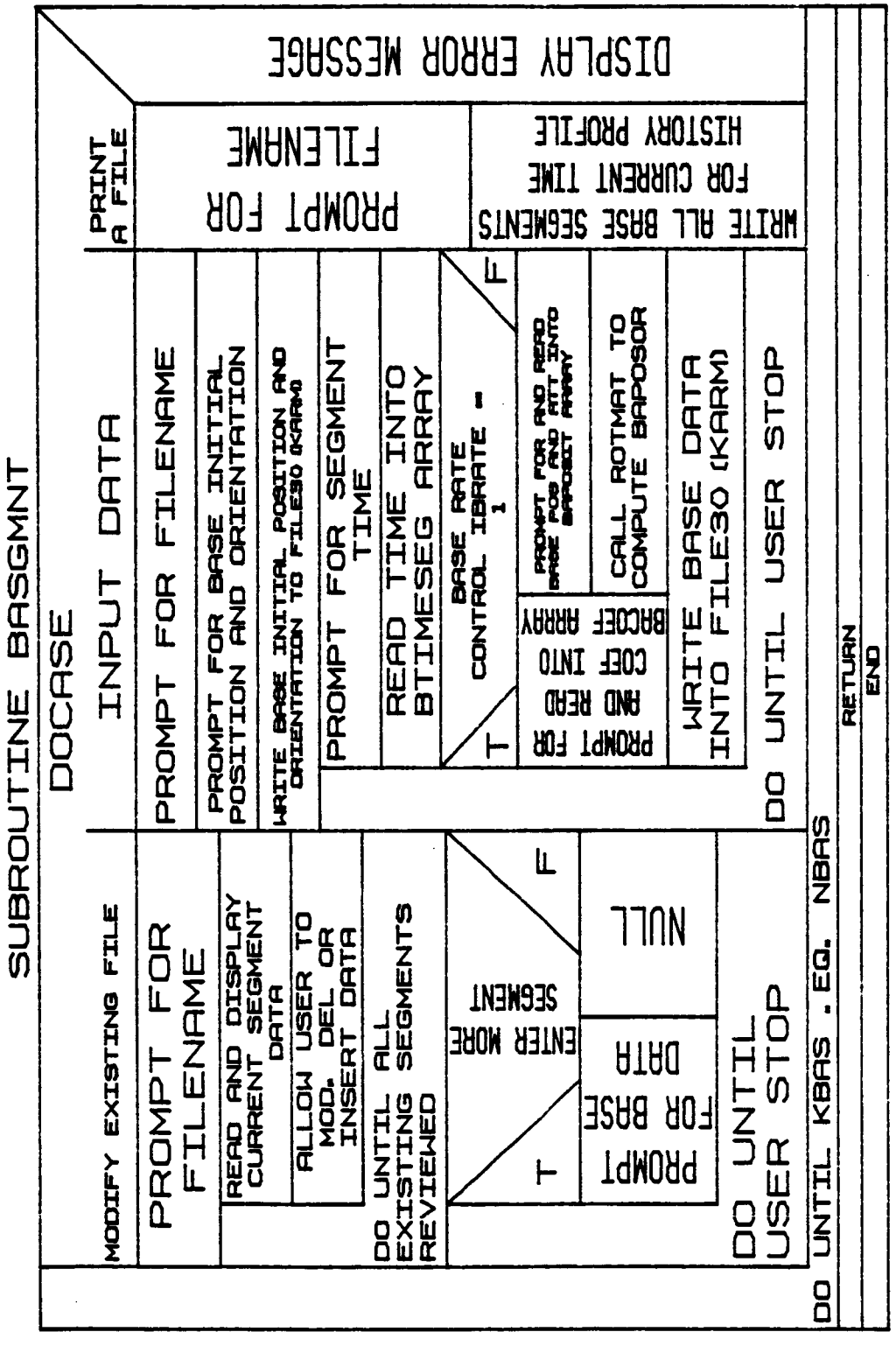
Subroutine TASK is the preliminary routine called when defining manipulator motion. The user has the choice of modifying an existing time history (motion) profile, creating a new profile or writing a user readable, formatted file from an existing time history profile. The subroutine opens the appropriate files and then calls subroutine SGMNT if an existing file is being modified, a formatted file is to be written, or a new file is to be created using just lower level motion commands. If task level commands are used, TASK calls subroutine SEGTASK. For other options task calls SGMNT.

SUBROUTINE TASK
DOCASE
MODIFY EXISTING THP
CREATE NEW THP
PRINT COPY OF EXISTING THP
(NULL)
PROMPT FOR NAME OF FILE TO MODIFY
PROMPT USER FOR NAME OF FILE TO STORE THP IN
PROMPT USER FOR FILENAME OF THP TO PRINT
OPEN FILE
OPEN FILE
OPEN FILE
PROMPT FOR NAME OF FILE TO STORE THP IN
USE TASK LEVEL COMMANDS
T
F
ENTER NAME OF FILE TO WRITE FORMATTED THP DATA TO
OPEN FILE
CALL SEGTASK TO DEFINE TASK DATA
CALL SGMNT TO DEFINE LOWER LEVEL THP DATA
CALL SGMNT TO WRITE FORMATTED FILE
OPEN FILE
CALL SGMNT TO DEFINE NEW THP DATA
DO UNTIL KARM .EQ. NUMBER OF ARMS
RETURN
END

2.1.6 BASGMNT

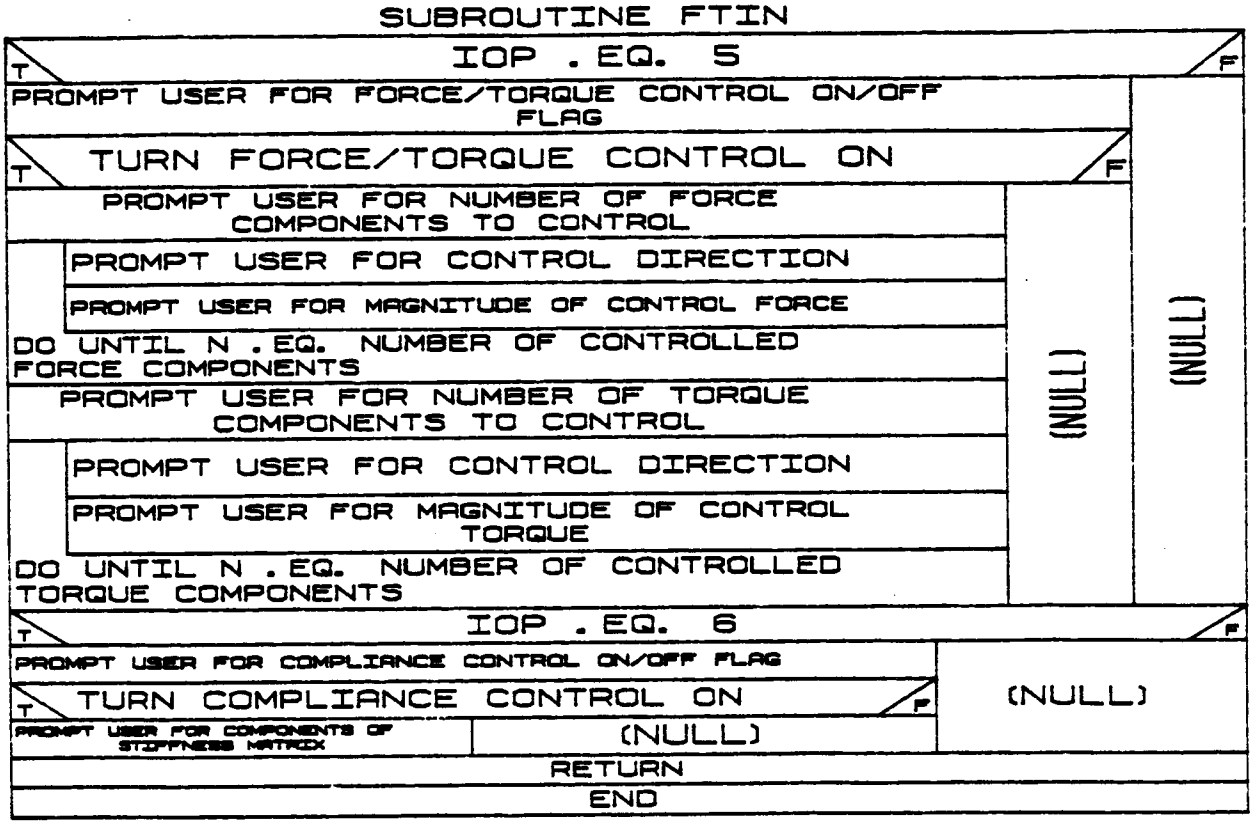
Subroutine BASGMNT allows the user to set up the desired base motion profile for a requirements analysis or response simulation run. It is called from SIMDRVR. An existing base motion profile file may be read in and modified or the profile may be defined interactively. Base motion is specified in one of two ways:

1) Desired position and orientation of the base;
2) Rate of base motion.


SUBROUTINE BASGMNT
DOCASE
MODIFY EXISTING FILE
PROMPT FOR FILENAME
READ AND DISPLAY CURRENT SEGMENT DATA
ALLOW USER TO MOD. DEL OR INSERT DATA
DO UNTIL ALL EXISTING SEGMENTS REVIEWED
ENTER MORE SEGMENT
T
F
PROMPT FOR BASE DATA
NULL
DO UNTIL USER STOP
INPUT DATA
PRINT A FILE
PROMPT FOR FILENAME
PROMPT FOR BASE INITIAL POSITION AND ORIENTATION
WRITE BASE INITIAL POSITION AND ORIENTATION TO FILE30 (KARM)
PROMPT FOR SEGMENT TIME
READ TIME INTO BTIMESEG ARRAY
BASE RATE CONTROL IBRATE = 1
T
F
PROMPT FOR AND READ BASE POS AND ATT INTO BPOSIT ARRAY
CALL ROTMAT TO COMPUTE BAPOSOR
PROMPT FOR AND READ COEF INTO BACOEF ARRAY
WRITE BASE DATA INTO FILE30 (KARM)
DO UNTIL USER STOP
DISPLAY ERROR MESSAGE
WRITE ALL BASE SEGMENTS FOR CURRENT TIME HISTORY PROFILE
DO UNTIL KBAS .EQ. NBAS
RETURN
END

2.2.1 FTIN

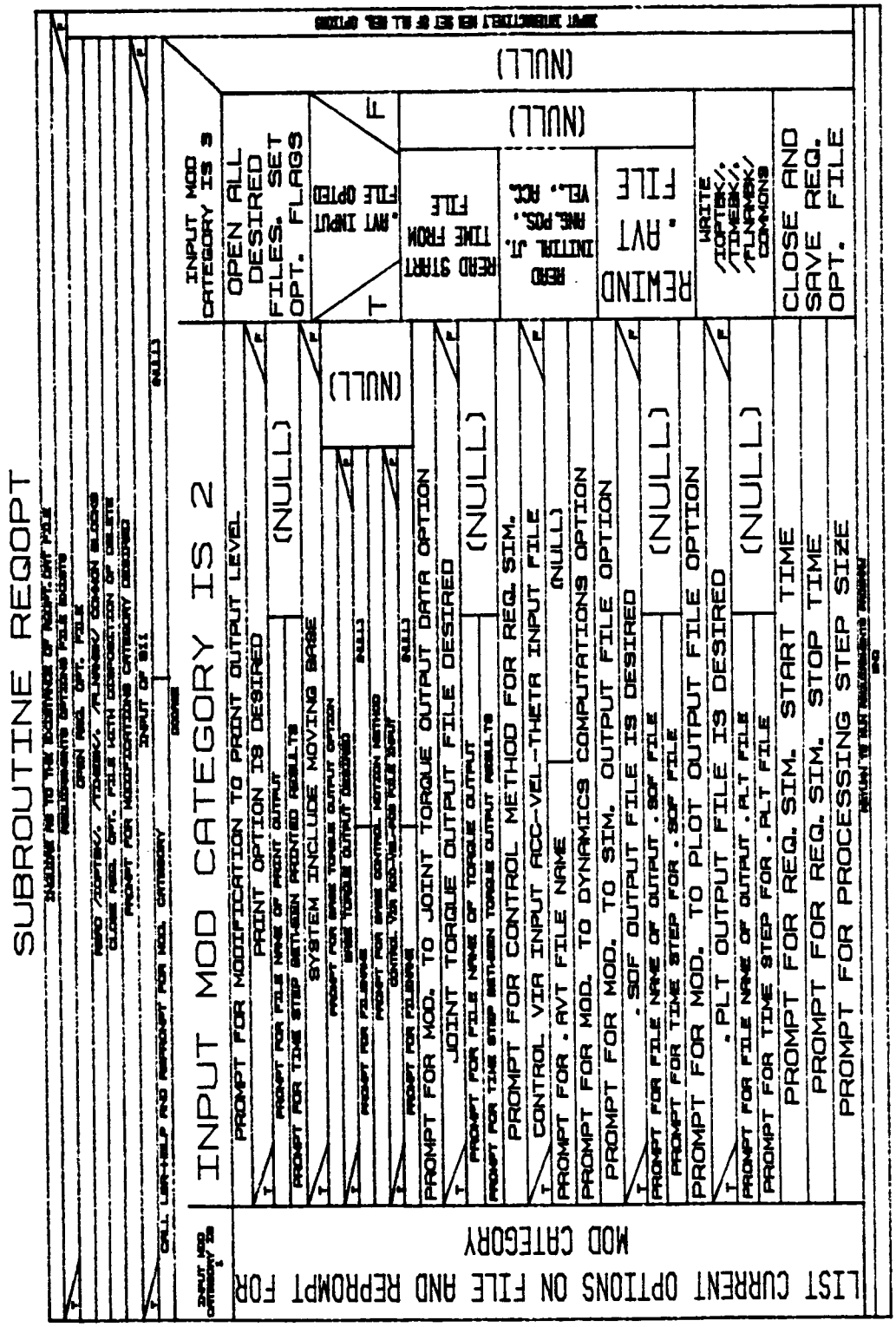
Subroutine FTIN is called from SGMNT if force/torque or active compliance control was specified by the user. If force/torque control was specified, the user is prompted for the number of force and torque components to be controlled, the unit vectors in the directions to be controlled and the magnitude of the control force or torque. If active compliance control was specified, the user is prompted for the stiffness matrix at the end-effector reference point.



2.2.2 REQOPT

Subroutine REQOPT is called from REQUIR to define requirements analysis run time options. The user may list currently defined options and use them or input a new set of options. Options the user may set include run time data file write, torque file write, control method to be used, control of robot base, dynamic calculations, playback file write, plot file write, and simulation start time, stop time, and processing step size.

ORIGINAL PAGE IS
OF POOR QUALITY



2.2.3 PRTARM

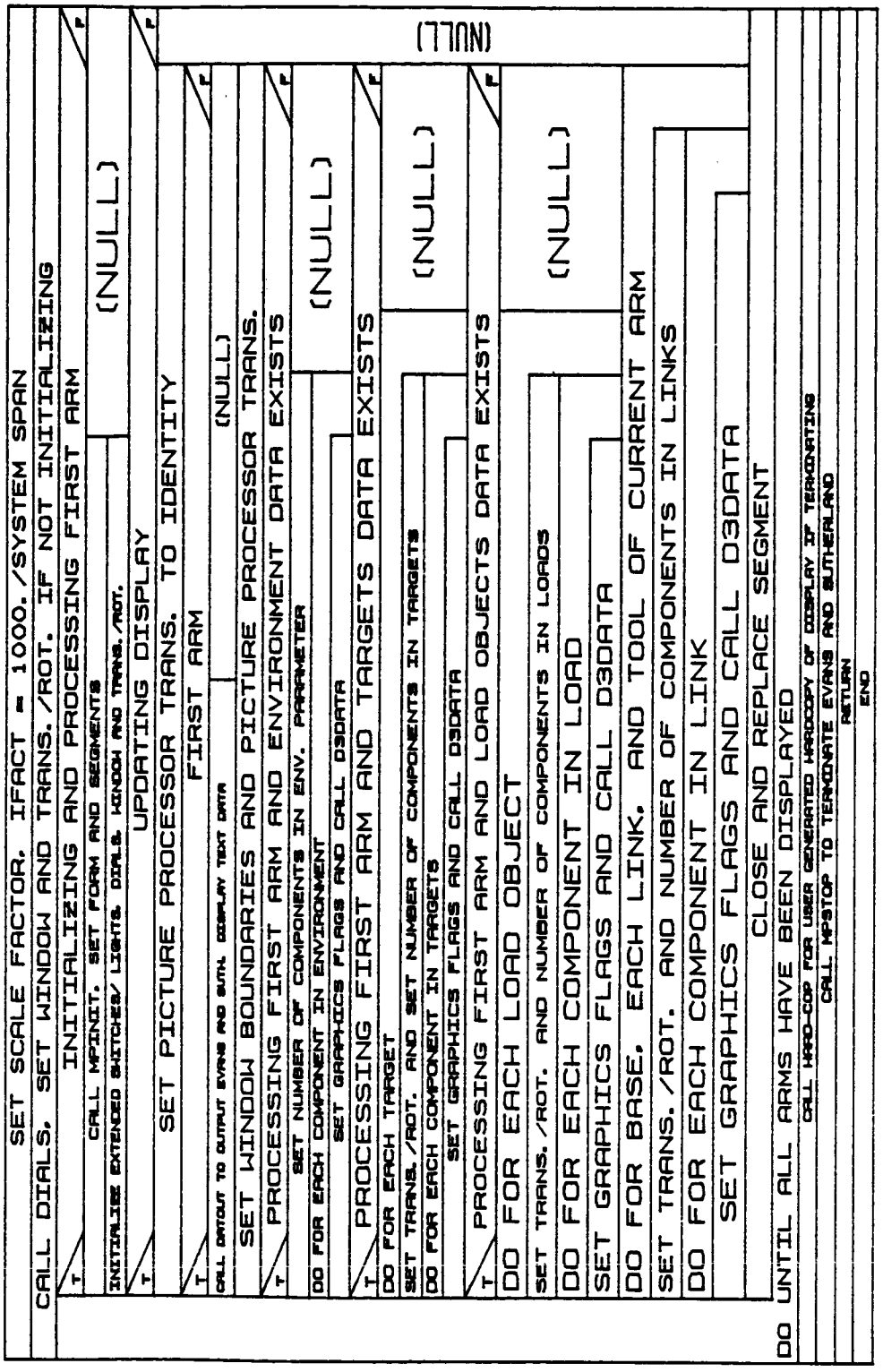
Subroutine PRTARM is called form either REQUIR or RESPON when the flag for printed output of that analysis is set. This routine prints a description of the manipulator system that includes the following variables: current arm number and number of joints per arm, type and mass of each joint, initial angular positions and velocities of each joint, joint travel and rate limits, joint/link centroid locations, joint location relative to previous joint, inertia matrix for each joint, orientation matrix for each joint relative to previous joint, span of the whole system, and the acceleration attributable to gravity.

SUBROUTINE PRTARM

WRITE CURRENT ARM NUMBER AND THE NUMBER OF JOINTS PER ARM
WRITE THE TYPE AND MASS OF EACH JOINT
CONVERT DATA TO BE WRITTEN FROM INTERNAL TO INPUT/OUTPUT UNITS
WRITE JOINT INITIAL ANGULAR POSITIONS AND VELOCITIES
WRITE JOINT TRAVEL AND RATE LIMITS
WRITE JOINT/LINK CENTROID LOCATIONS
WRITE JOINT LOCATIONS RELATIVE TO PREVIOUS JOINT
WRITE INERTIA MATRICES FOR JOINT/LINK COMBINATIONS
WRITE ORIENTATION MATRICES FOR EACH JOINT RELATIVE TO PREVIOUS JOINT
DO UNTIL KARM = NUMBER OF ARMS IN THE SYSTEM
WRITE TOTAL SYSTEM SPAN
WRITE ACCELERATION DUE TO GRAVITY
RETURN
END

2.2.4 GRAFIX

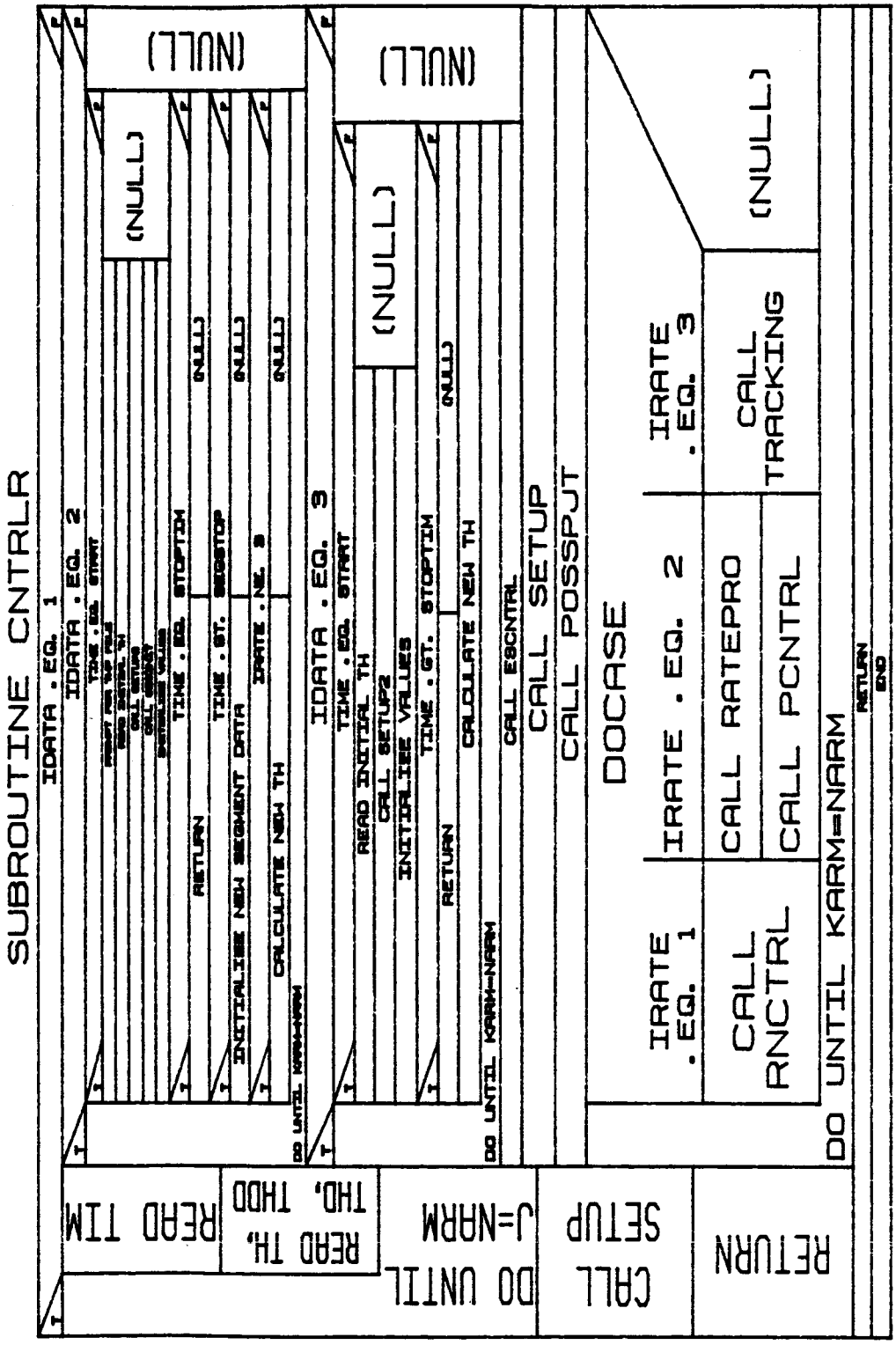
Subroutine GRAFIX provides the motion graphics capability in the response simulation, requirements analysis and postprocessing functions. GRAFIX displays the environment, target, load and robotic system motion within the environment. If IFLAG=1, the graphics system is initialized and displayed in the initial condition; if IFLAG=2, the display is updated to the current time step condition; if IFLAG=3, the motion is complete and the graphics are terminated.


SUBROUTINE GRAFIX
SET SCALE FACTOR, IFACT = 1000./SYSTEM SPAN
CALL DIALS, SET WINDOW AND TRANS./ROT. IF NOT INITIALIZING
INITIALIZING AND PROCESSING FIRST ARM
T
CALL MPINIT, SET FORM AND SEGMENTS
INITIALIZE EXTENDED SWITCHES/ LIGHTS, DIALS, WINDOW AND TRANS./ROT.
(NULL)
F
UPDATING DISPLAY
T
SET PICTURE PROCESSOR TRANS. TO IDENTITY
FIRST ARM
T
CALL DATOUT TO OUTPUT EVANS AND SUTH. DISPLAY TEXT DATA
(NULL)
F
(NULL)
SET WINDOW BOUNDARIES AND PICTURE PROCESSOR TRANS.
T
PROCESSING FIRST ARM AND ENVIRONMENT DATA EXISTS
SET NUMBER OF COMPONENTS IN ENV. PARAMETER
DO FOR EACH COMPONENT IN ENVIRONMENT
SET GRAPHICS FLAGS AND CALL D3DATA
(NULL)
F
T
PROCESSING FIRST ARM AND TARGETS DATA EXISTS
DO FOR EACH TARGET
SET TRANS./ROT. AND SET NUMBER OF COMPONENTS IN TARGETS
DO FOR EACH COMPONENT IN TARGETS
SET GRAPHICS FLAGS AND CALL D3DATA
(NULL)
F
T
PROCESSING FIRST ARM AND LOAD OBJECTS DATA EXISTS
DO FOR EACH LOAD OBJECT
SET TRANS./ROT. AND NUMBER OF COMPONENTS IN LORDS
(NULL)
DO FOR EACH COMPONENT IN LOAD
SET GRAPHICS FLAGS AND CALL D3DATA
DO FOR BASE, EACH LINK, AND TOOL OF CURRENT ARM
SET TRANS./ROT. AND NUMBER OF COMPONENTS IN LINKS
DO FOR EACH COMPONENT IN LINK
SET GRAPHICS FLAGS AND CALL D3DATA
CLOSE AND REPLACE SEGMENT
DO UNTIL ALL ARMS HAVE BEEN DISPLAYED
CALL HARD-COP FOR USER GENERATED HARDCOPY OF DISPLAY IF TERMINATING
CALL MPSTOP TO TERMINATE EVANS AND SUTHERLAND
RETURN
END

2.2.5 CNTRLR

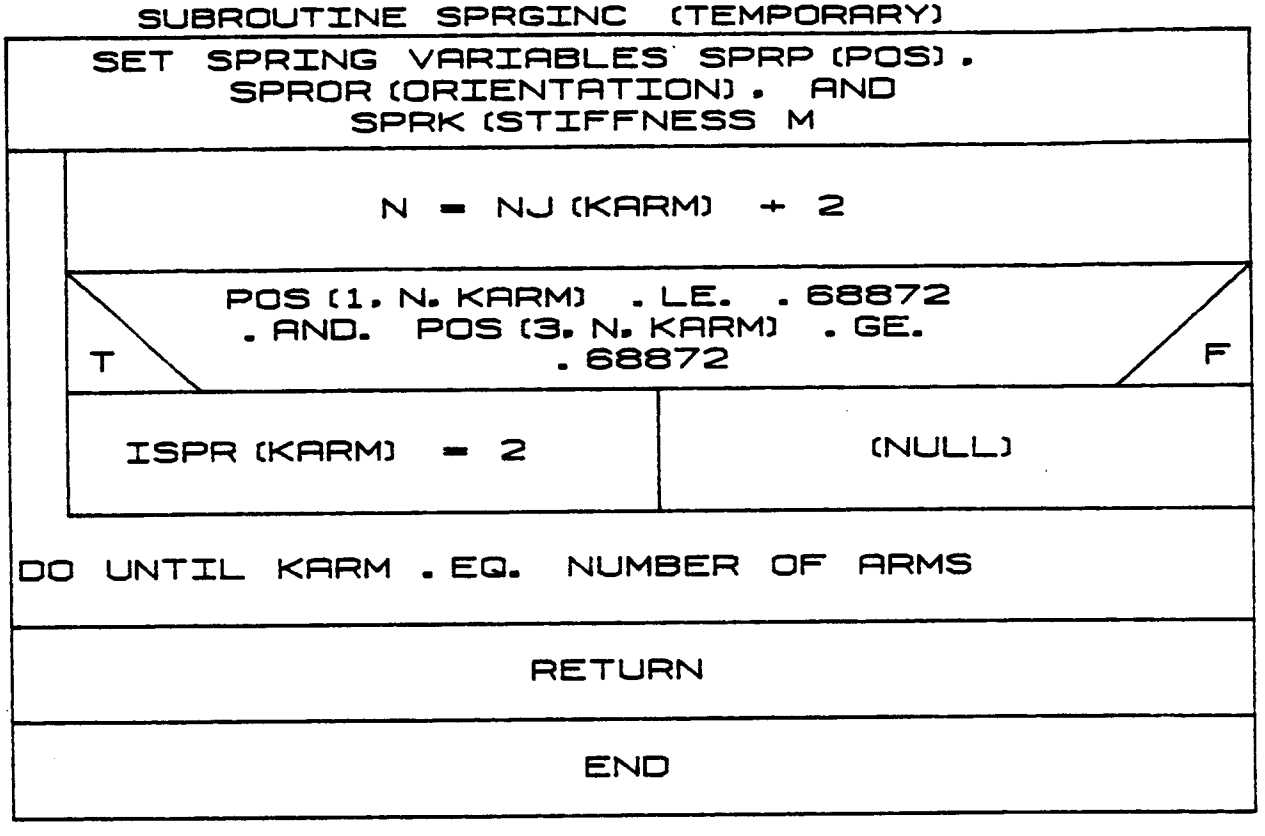
Subroutine CNTRLR is called from REQUIR to obtain the angular position, velocity, and acceleration for each joint of each arm at each processing time step. If the variable IDATA was set to 1 earlier, the data are obtained by reading an existing file that contains just these data. If IDATA equals 2, the values are calculated from the motion profiles. Subroutine PCNTRL is called for the position control calculations and RCNTRL is called for the rate control calculations and TRACKING is called for sensor control. IDATA equal to 3 allows the system motion to be controlled by dials on the Evans and Sutherland.

ORIGINAL PAGE IS
OF POOR QUALITY



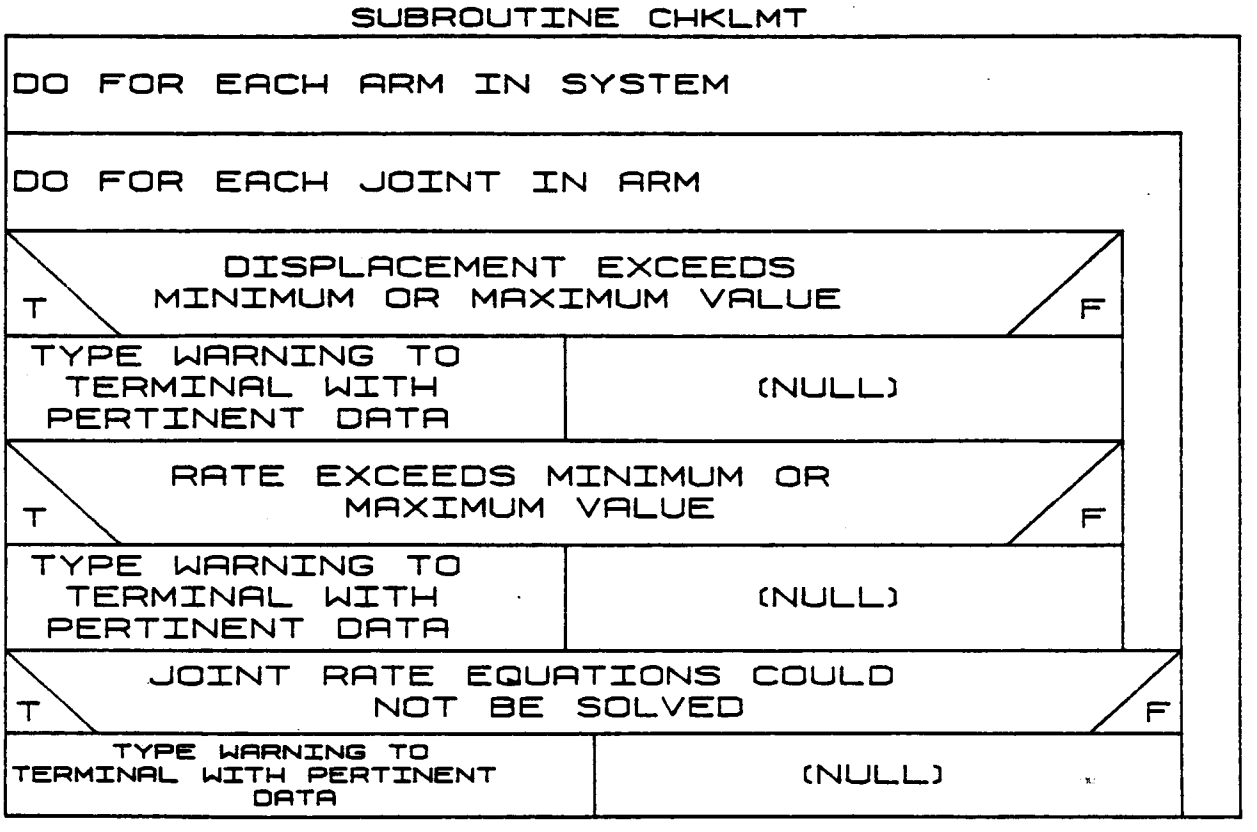
2.2.6 SPRGINC

Subroutine SPRGINC is called from REQUIR to set the variables used when the end-effector is to have compliance associated with it. The variables set includes spring reference position, orientation and the spring constant.



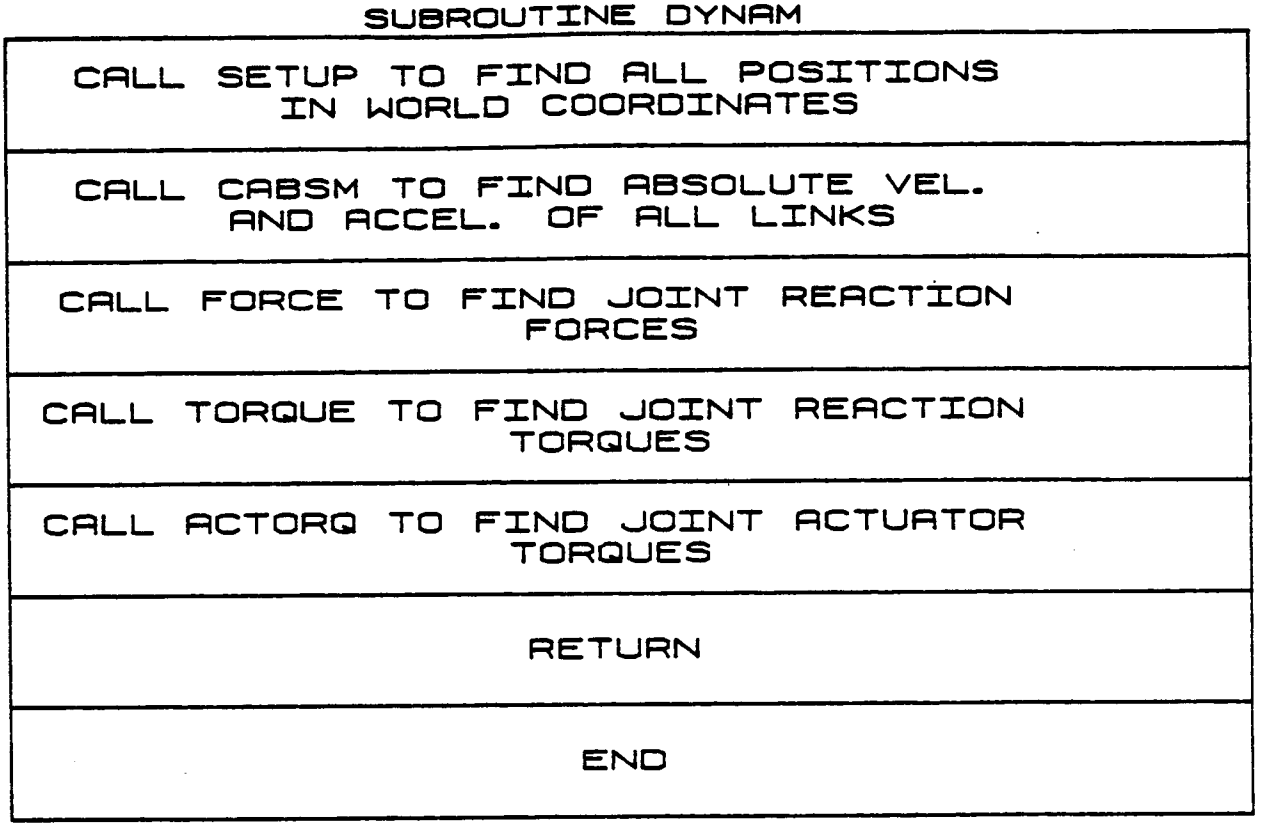
2.2.7 CHKLMT

CHKLMT checks joint displacement and rate limits during requirements analysis. It does not modify any values but prints a warning to the terminal if any limits are exceeded.



2.2.8 DYNAM

Subroutine DYNAM is called from REQUIR to compute the manipulator system dynamics at each processing time step by calling the SETUP, CABSM, FORCE, TORQUE, and ACTORQ subroutines.



2.2.9 VOLTAGE

(Not implemented yet.)

2.2.10 OUTREQ

Subroutine OUTREQ is called from REQUIR to write output data to files requested by the user. The files the user may elect to have data written to are:

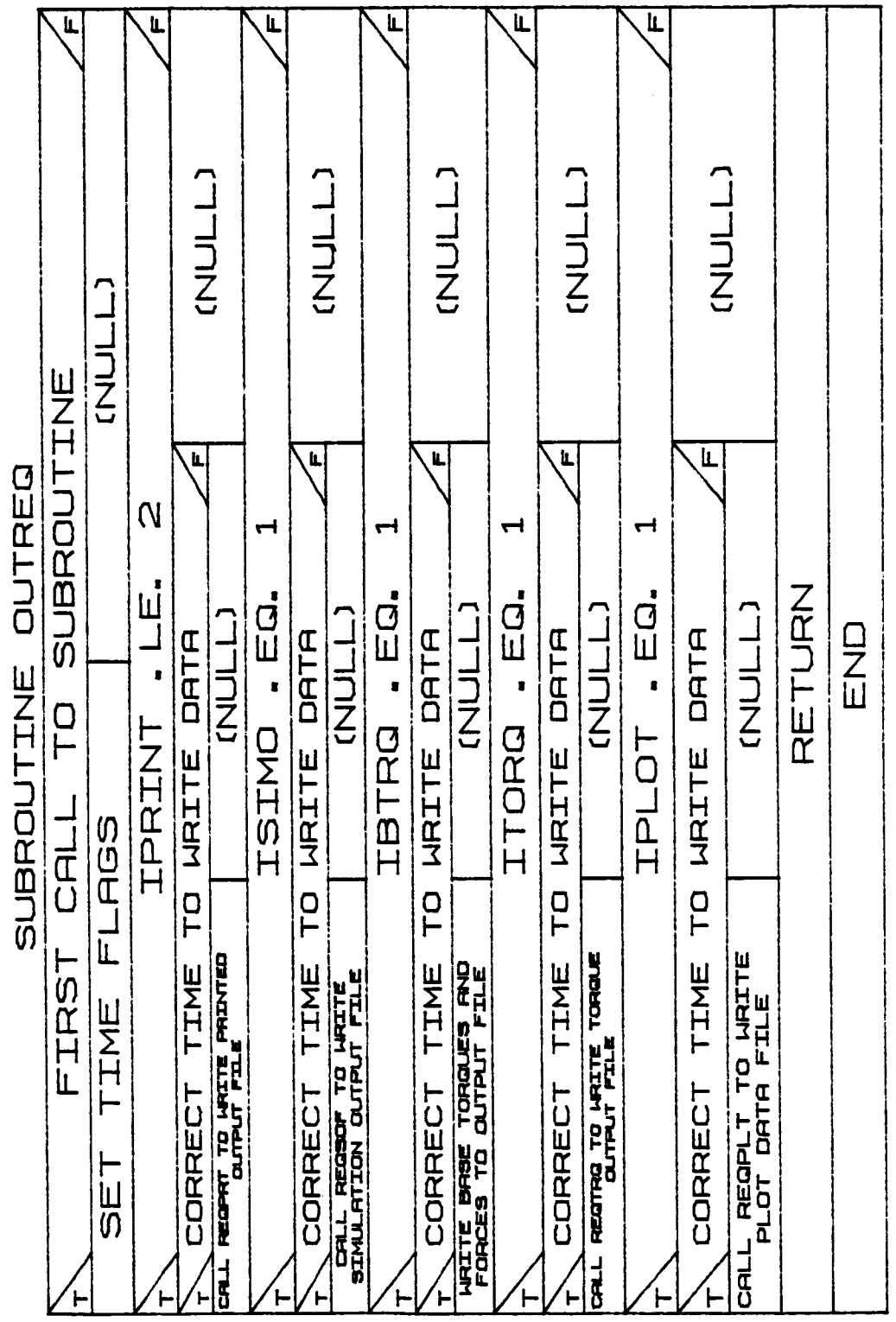
1) Run time output data file;

2) Data file for subsequent replay of motion on a vector graphics system;

3) Actuator torque data file;

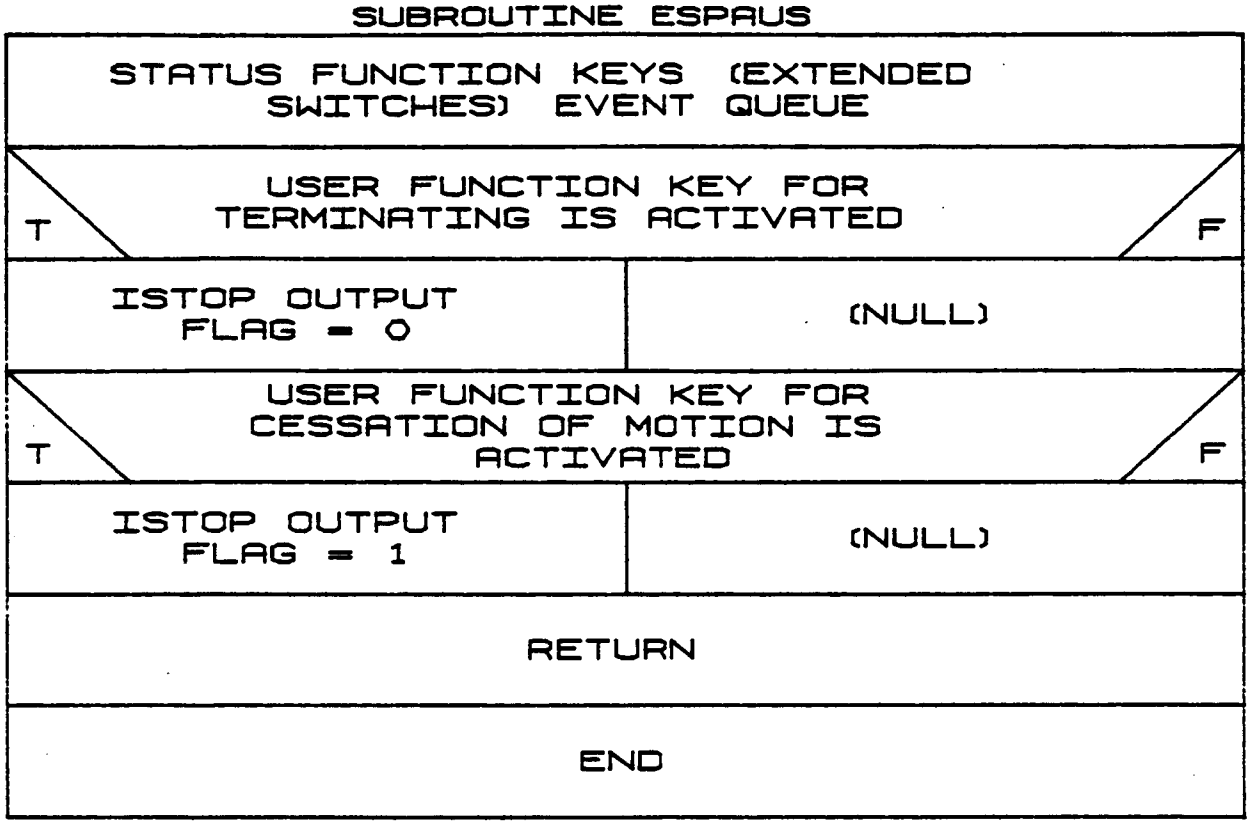
4) Run time data file for subsequent plotting;

5) Base torques and forces data file.


SUBROUTINE OUTREQ
FIRST CALL TO SUBROUTINE
T
F
SET TIME FLAGS
(NULL)
IPRINT .LE. 2
CORRECT TIME TO WRITE DATA
CALL REQPRT TO WRITE PRINTED OUTPUT FILE
(NULL)
ISIMO .EQ. 1
CORRECT TIME TO WRITE DATA
CALL REQSOP TO WRITE SIMULATION OUTPUT FILE
(NULL)
IBTRQ .EQ. 1
CORRECT TIME TO WRITE DATA
WRITE BASE TORQUES AND FORCES TO OUTPUT FILE
(NULL)
ITORQ .EQ. 1
CORRECT TIME TO WRITE DATA
CALL REQTRQ TO WRITE TORQUE OUTPUT FILE
(NULL)
IPLOT .EQ. 1
CORRECT TIME TO WRITE DATA
CALL REQPLT TO WRITE PLOT DATA FILE
(NULL)
RETURN
END

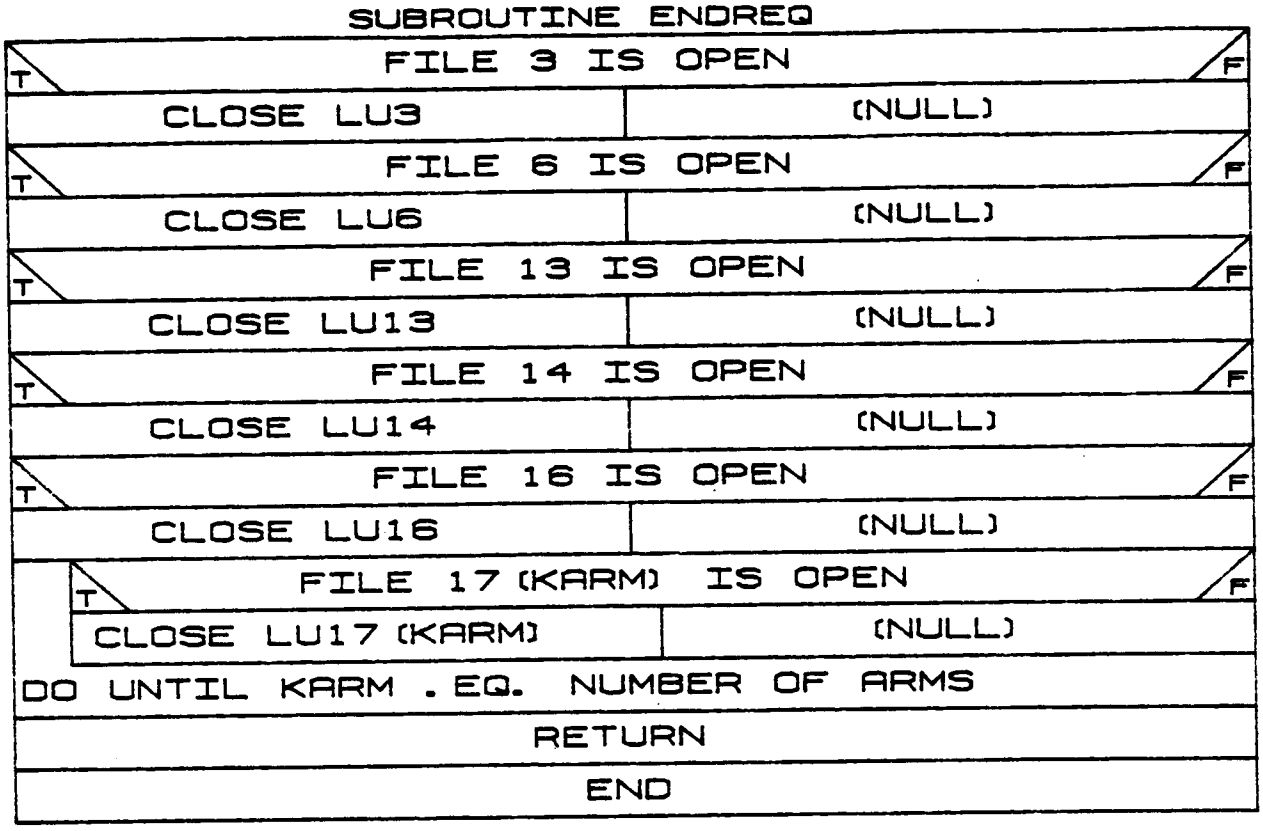
2.2.11 ESPAUS

Routine ESPAUS is responsible for polling the status of the E&S function keys to determine the on/off status of the devices switch for playback motion cessation. A light indicator in the function key is used to inform the user of the key status; when lighted, the perspective viewing is in operation.



2.2.12 ENDREQ

Subroutine ENDREQ closes any files opened during running of the requirements analysis portion of ROBSIM.

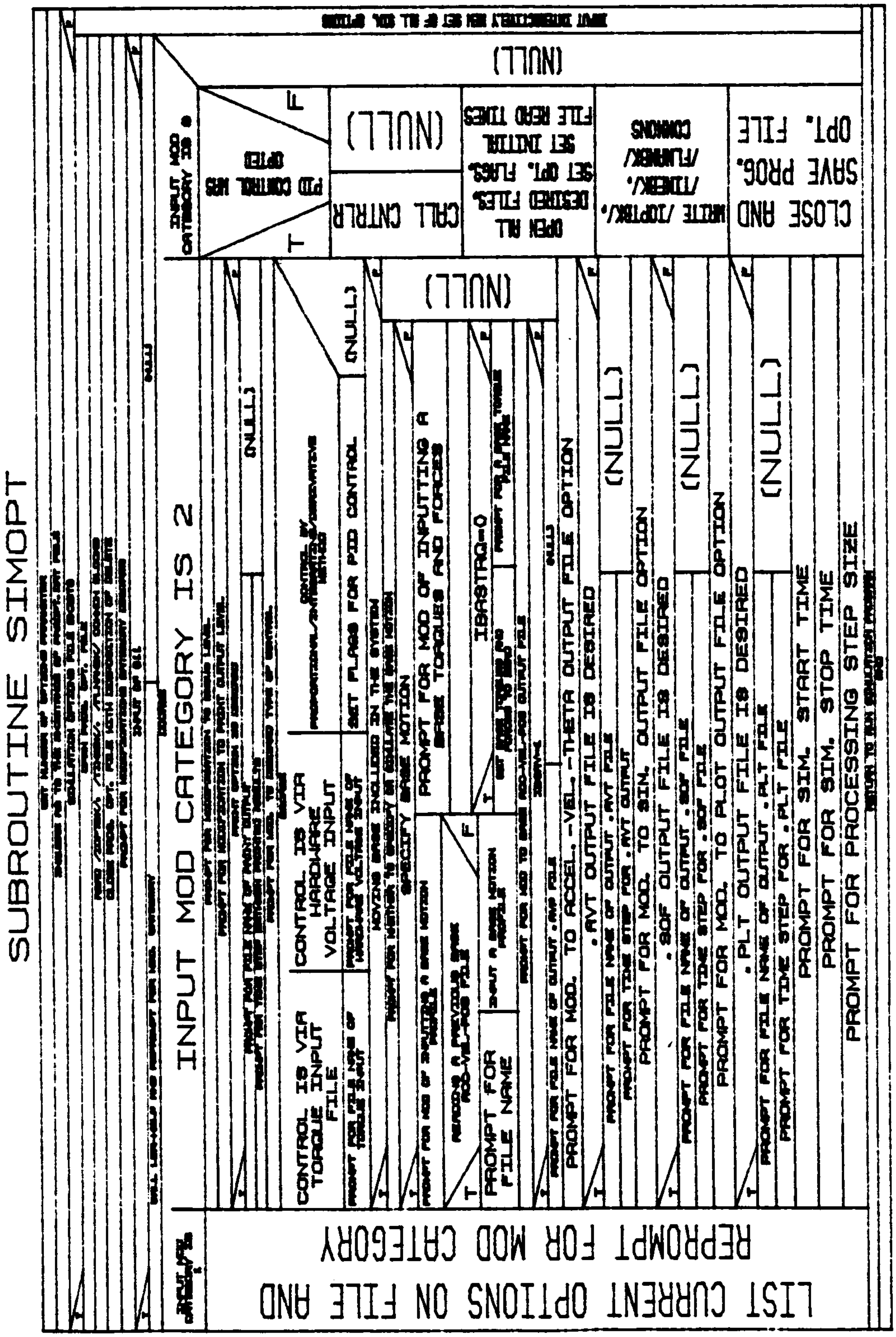


2.2.13 SIMOPT

Subroutine SIMOPT interactively prompts the user for the program start time, stop time, processing time step, and several flags for control of output and the selection of some computational capabilities. Among these output options is a simulation output file that contains the data required by the postprocessing function for further study. The user also specifies the time frequency of the output of data to the file. The user is also allowed to request printed output during the analysis tools function execution.

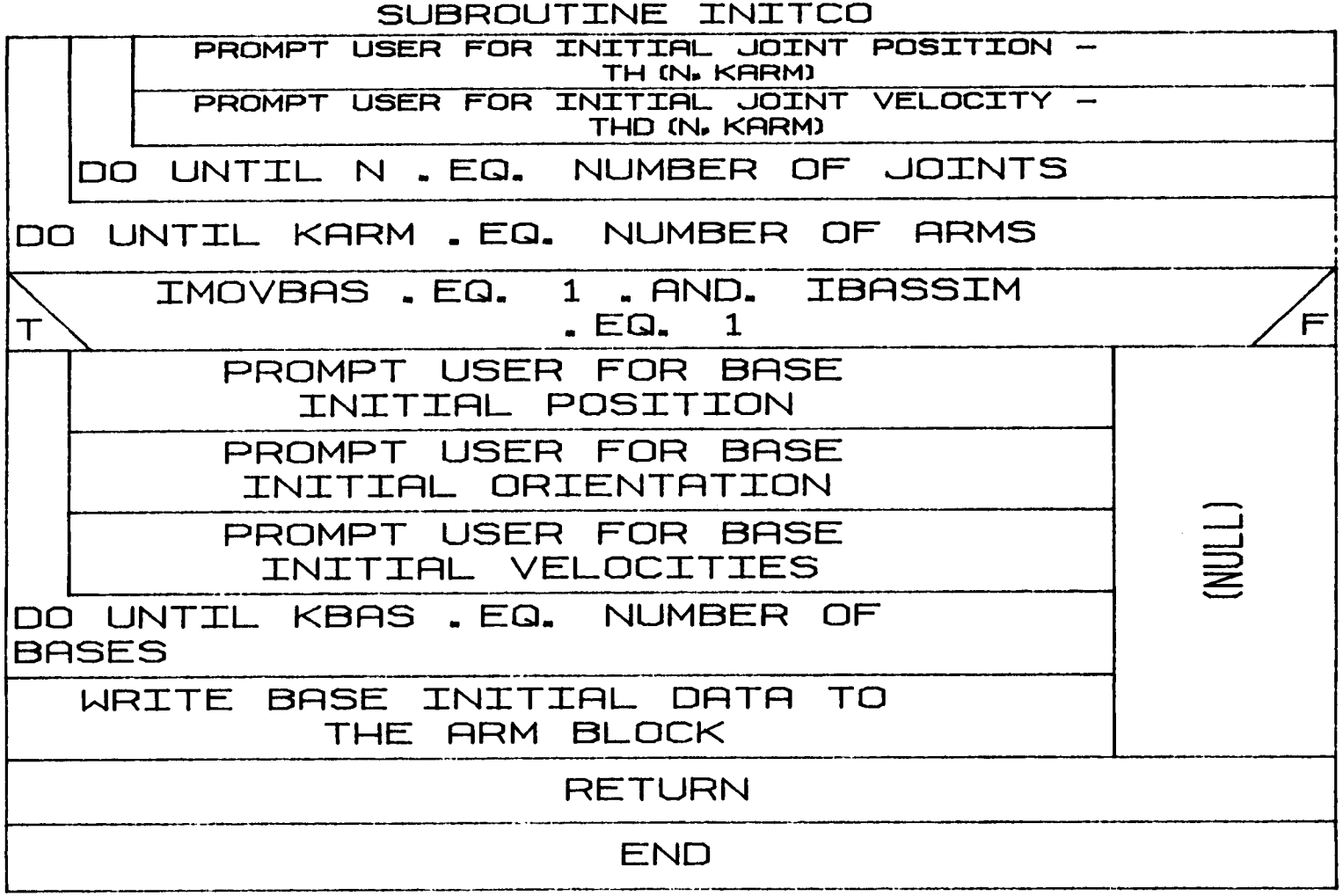
The content and format of the data to be printed are provided for within each of the analysis tools. The flag set within SIMOPT is used only to turn the print routines on. The time frequency of the printed output is also specified. Other options are for generation of an acceleration-velocity-theta file and/or a plot output data file that may be plotted with the ROBSIM postprocessing plot utility with their associated output time steps. The user may also request use of a torque input file or a control option to read a hardware input voltage file for computational capabilities.

ORIGINAL PAGE IS
OF POOR QUALITY



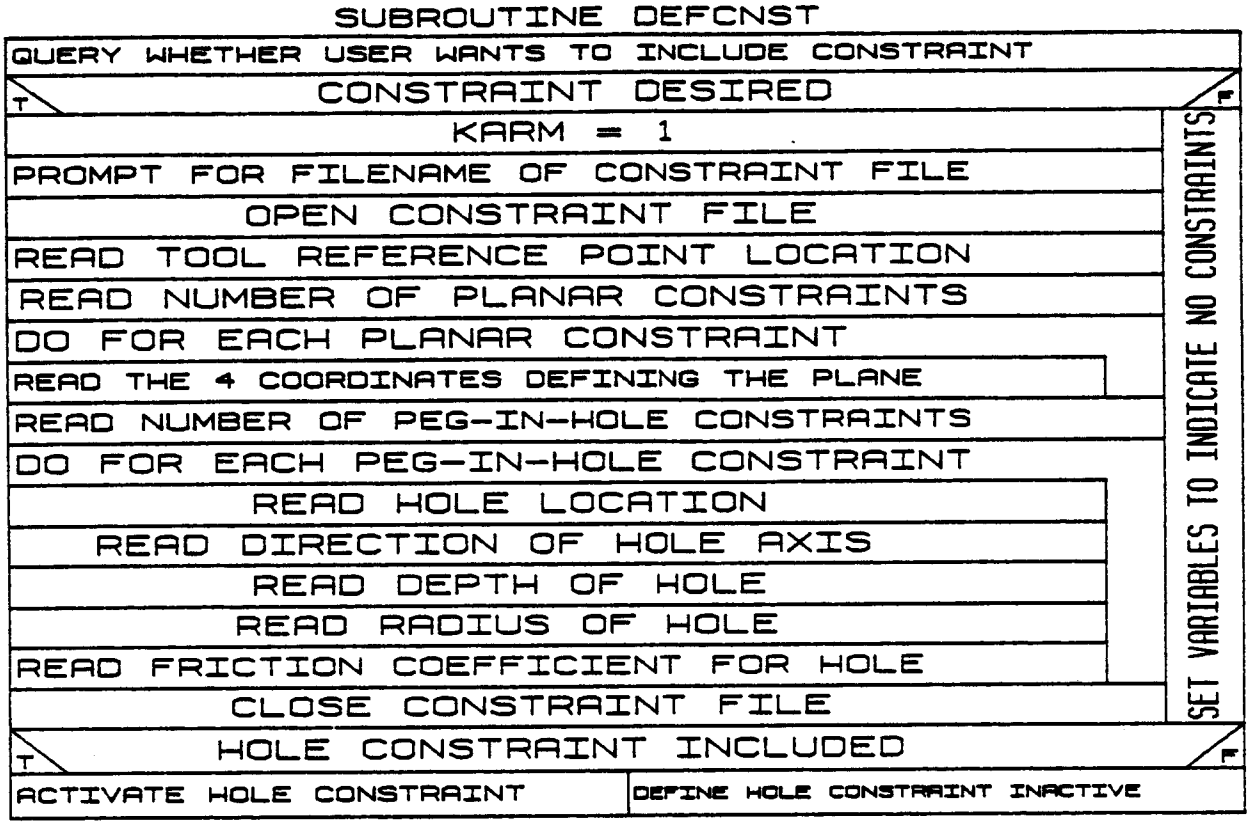
2.2.14 INITCO

Subroutine INITCO prompts the user for the initial joint position (TH) and velocity (THD) of each joint of each arm. If moving base is simulated, the routine prompts the user for the initial base positions, orientations and velocities.


SUBROUTINE INITCO
PROMPT USER FOR INITIAL JOINT POSITION - TH (N, KARM)
PROMPT USER FOR INITIAL JOINT VELOCITY - THD (N, KARM)
DO UNTIL N .EQ. NUMBER OF JOINTS
DO UNTIL KARM .EQ. NUMBER OF ARMS
IMOVBAS .EQ. 1 .AND. IBASSIM .EQ. 1
T
F
PROMPT USER FOR BASE INITIAL POSITION
PROMPT USER FOR BASE INITIAL ORIENTATION
PROMPT USER FOR BASE INITIAL VELOCITIES
DO UNTIL KBAS .EQ. NUMBER OF BASES
WRITE BASE INITIAL DATA TO THE ARM BLOCK
(NULL)
RETURN
END

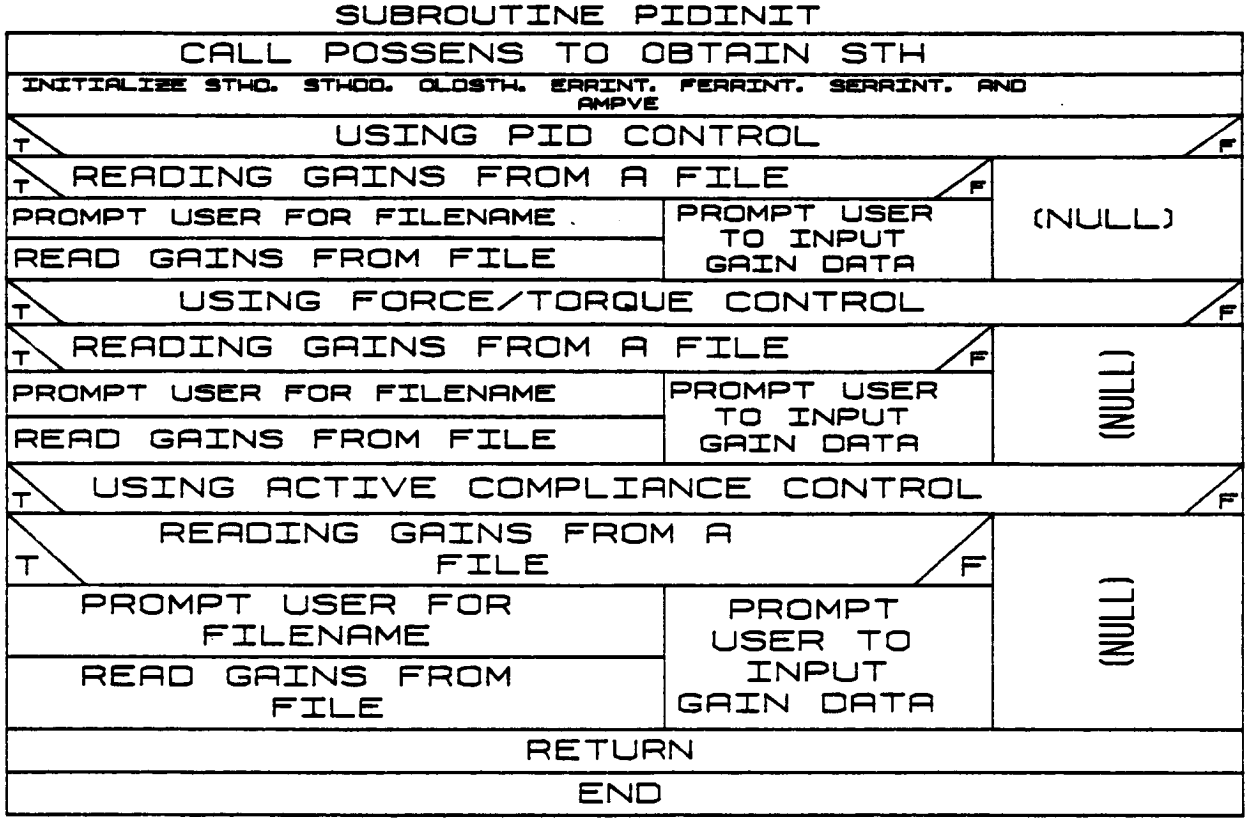
2.2.15 DEFCNST

DEFCNST reads a file containing the information needed to define a constraint (either planar or peg-in-hole type) on the end-effector motion during dynamic simulation of the arm response. The user specifies the name of the constraint file in response to interactive prompts.



2.2.16 PIDINIT

Subroutine PIDINT is called from RESPON to initialize variables used in the program's control algorithms. POSSENS is called first to determine the actual joint positions. Initial values for some control variables are set. The user is then asked to supply system gains for the methods of control that will be used during program execution. These gains may be supplied by either reading in a file of existing gains or by the user interactively inputting the gains.



2.2.17 OUTSIM

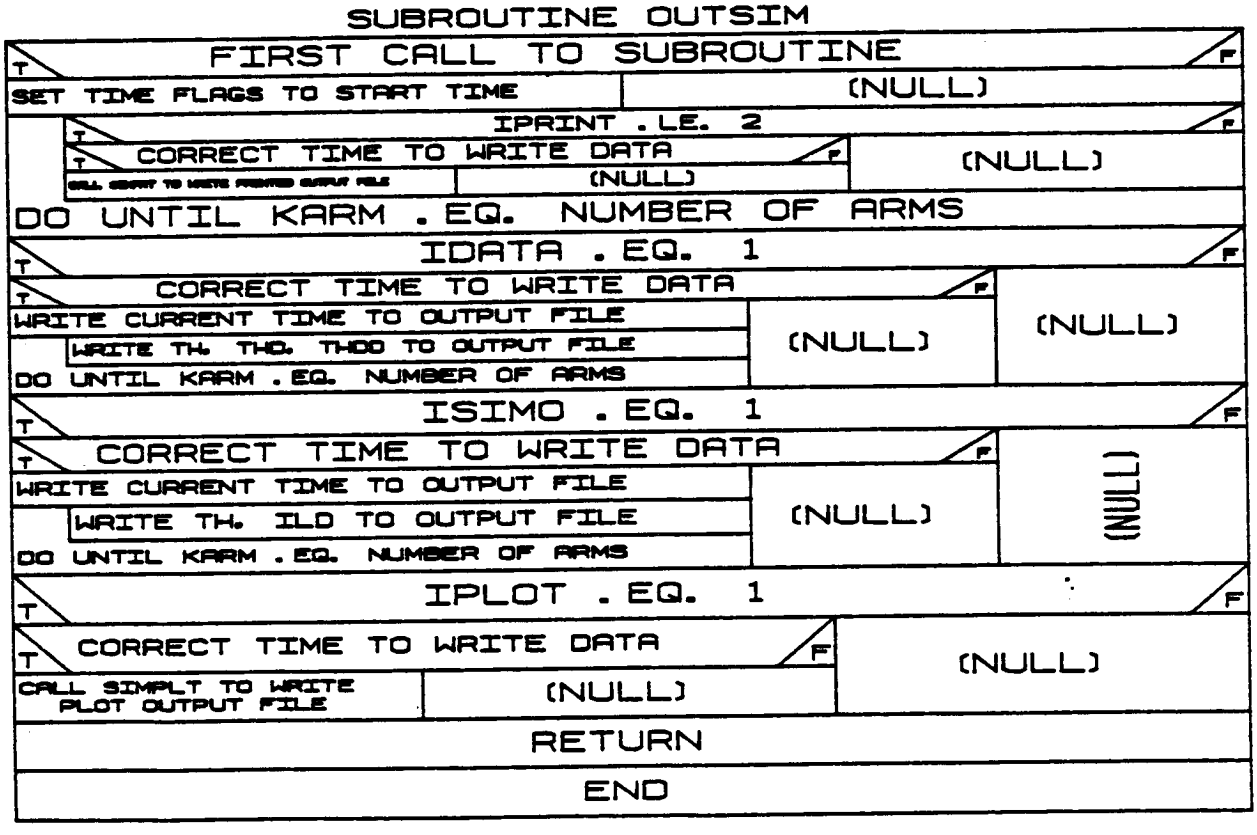
Subroutine OUTSIM is called from RESPON to write the appropriate output data to the different types of files requested by the user. Types of output files available are:

1) File of run time data for subsequent tabular printout;

2) File of joint positions, velocities and accelerations as functions of time;

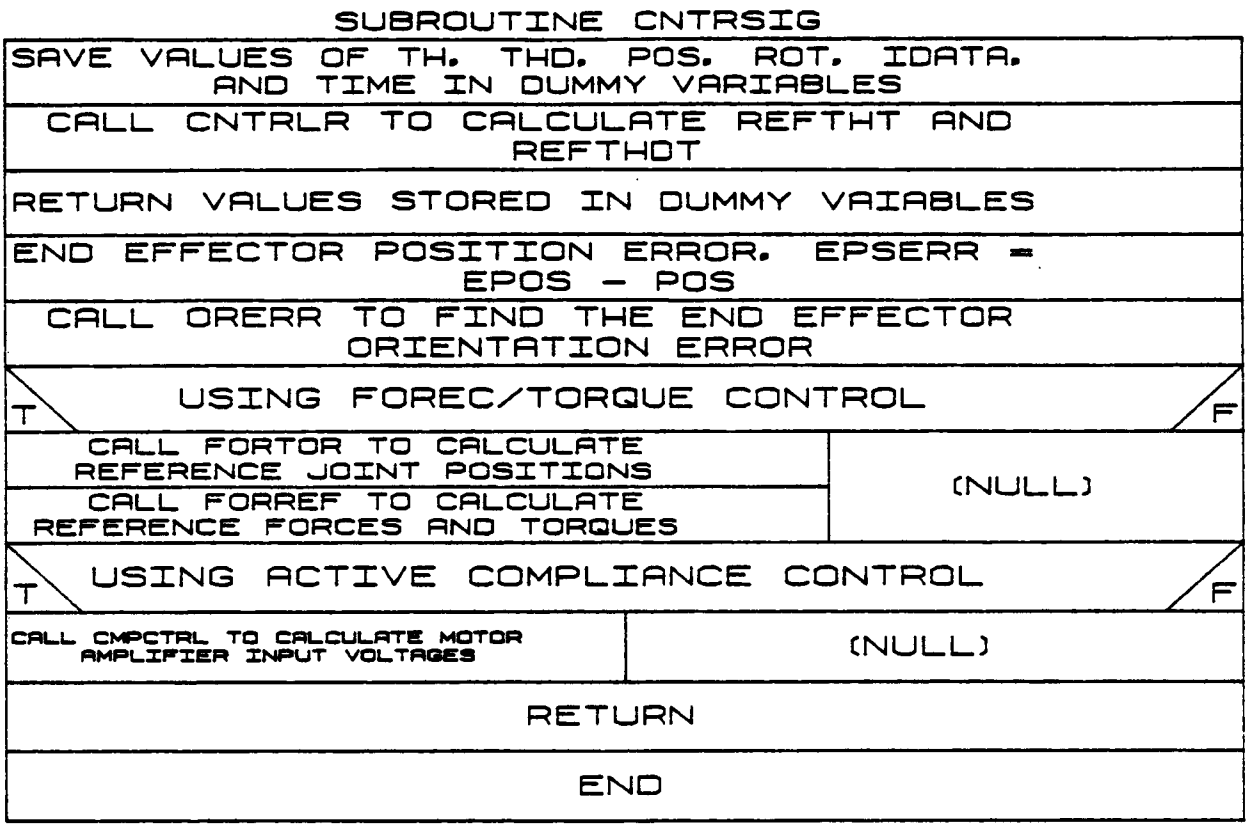
3) Data file for later motion replay on vector graphics machine;

4) File of data for subsequent x-y plotting.



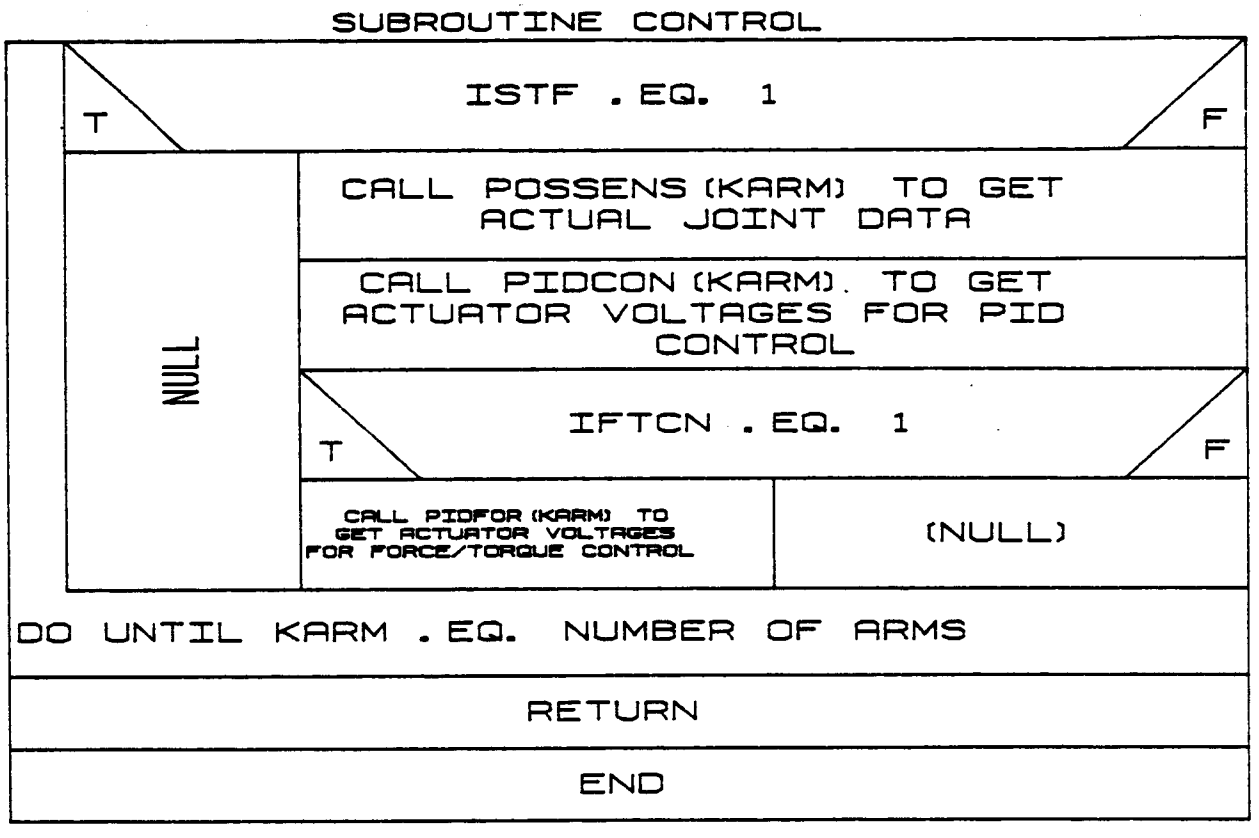
2.2.18 CNTRSIG

Subroutine CNTRSIG is called from the routine REQUIR. Joint variables are stored in dummy variables and CNTRLR is called to calculate joint angular reference positions and velocities. The end-effector position error is calculated and ORERR is called to determine the orientation error. If force/torque control is being used, subroutines FORTOR and FORREF are called to calculate joint reference positions and reference forces and torques. If active compliance control is being used, subroutine CMPCTRL is called to calculate amplifier input voltages.



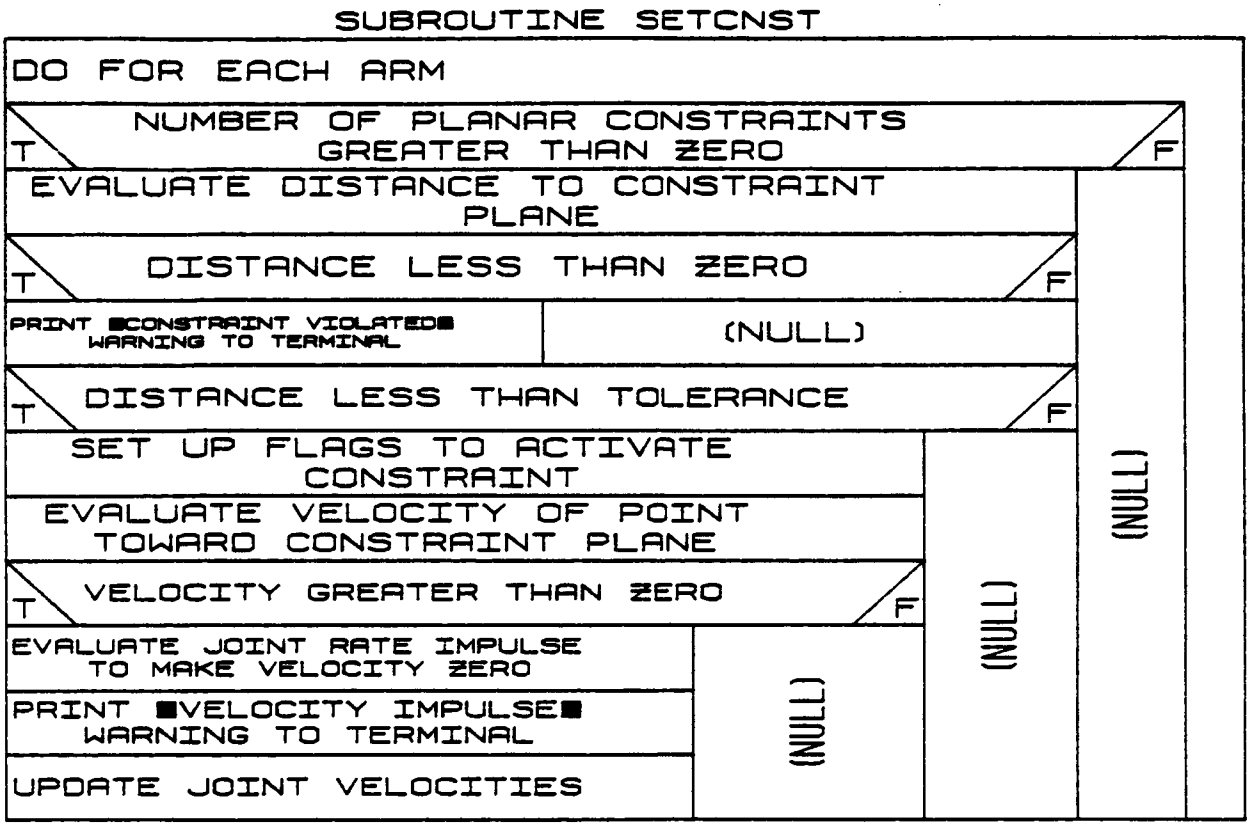
2.2.19 CONTROL

Subroutine CONTROL is called from RESPON at every processing time step. If a feedback control law is to be used, POSSENS is called to get the actual joint data and PIDCON is called to get actuator voltages for PID control. If force/torque control is being used, PIDFOR is also called to get actuator voltages caused by the force-controlled components.



2.2.20 SETCNST

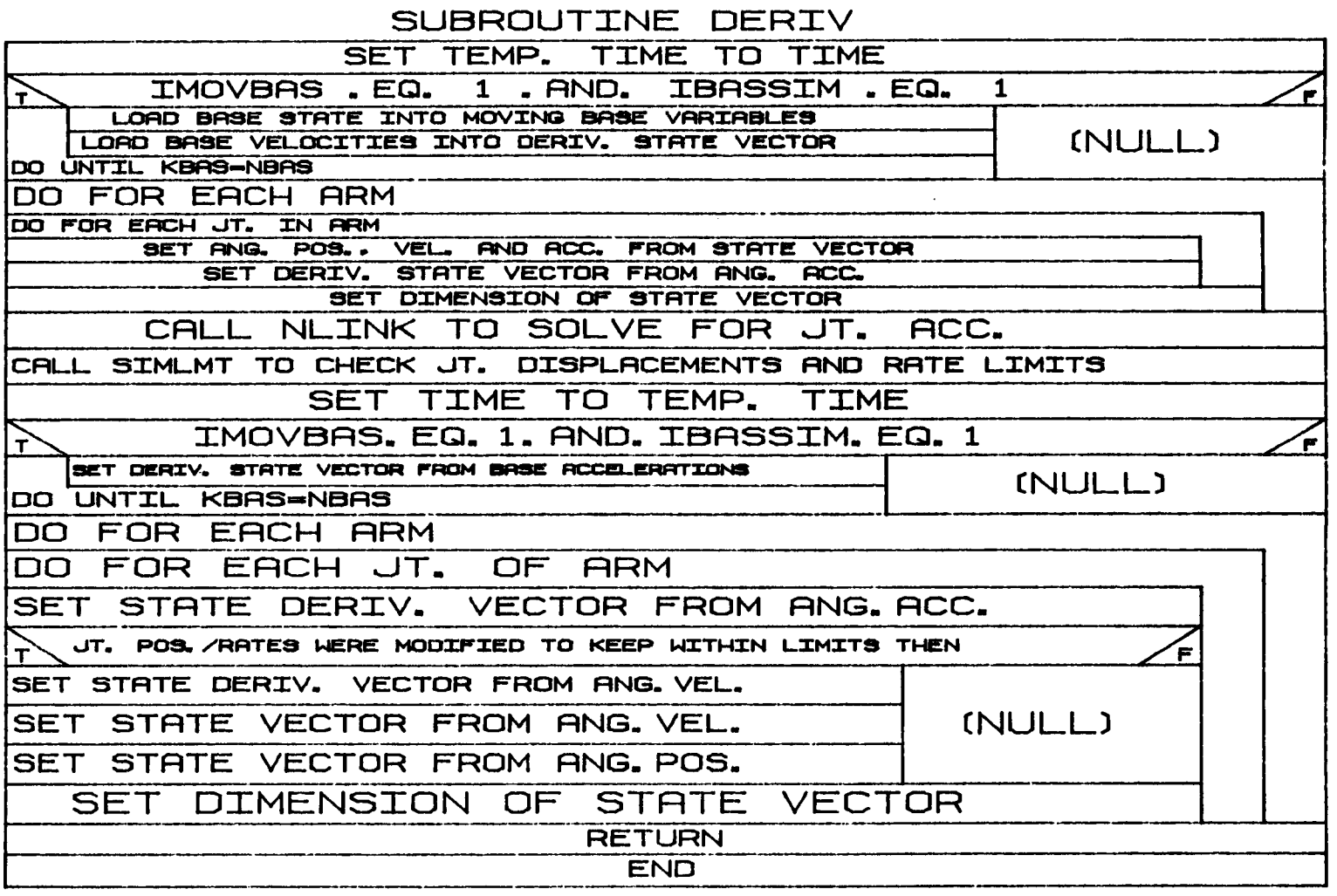
SETCNST checks planar constraints to see if they are violated or need to be activated. If the current velocity violates the constraint, the velocity impulse to satisfy the constraint is evaluated.



2.2.21 DERIV

DERIV is used during response simulation to interface between INTGRT and the dynamics module NLINK. This routine puts the state vector Z into the appropriate common variables, calls NLINK and puts the results from the common variable THDD into ZD.

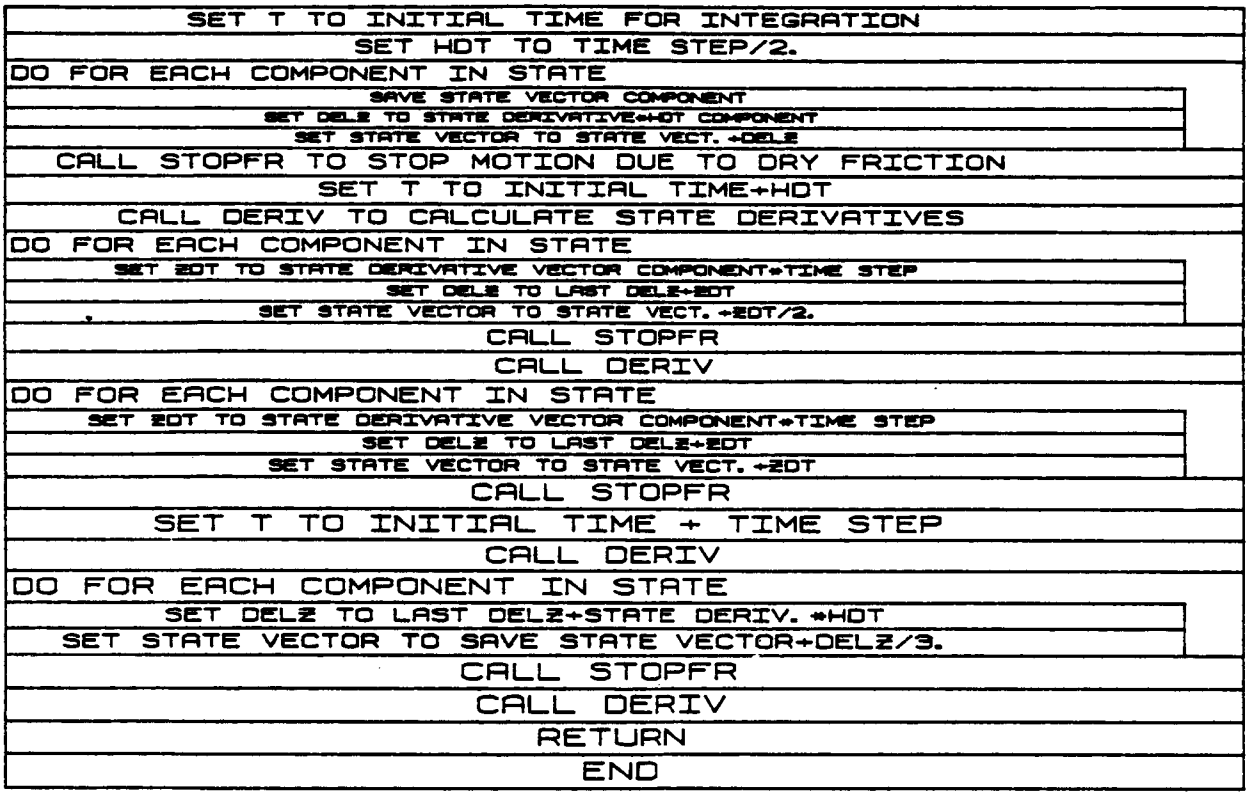
C.3


SUBROUTINE DERIV
SET TEMP. TIME TO TIME
IMOVBAS . EQ. 1 . AND. IBASSIM . EQ. 1
T
F
LOAD BASE STATE INTO MOVING BASE VARIABLES
LOAD BASE VELOCITIES INTO DERIV. STATE VECTOR
DO UNTIL KBAS=NBAS
(NULL)
DO FOR EACH ARM
DO FOR EACH JT. IN ARM
SET ANG. POS., VEL. AND ACC. FROM STATE VECTOR
SET DERIV. STATE VECTOR FROM ANG. ACC.
SET DIMENSION OF STATE VECTOR
CALL NLINK TO SOLVE FOR JT. ACC.
CALL SIMLMT TO CHECK JT. DISPLACEMENTS AND RATE LIMITS
SET TIME TO TEMP. TIME
IMOVBAS. EQ. 1. AND. IBASSIM. EQ. 1
T
F
SET DERIV. STATE VECTOR FROM BASE ACCELERATIONS
DO UNTIL KBAS=NBAS
(NULL)
DO FOR EACH ARM
DO FOR EACH JT. OF ARM
SET STATE DERIV. VECTOR FROM ANG. ACC.
JT. POS./RATES WERE MODIFIED TO KEEP WITHIN LIMITS THEN
T
F
SET STATE DERIV. VECTOR FROM ANG. VEL.
SET STATE VECTOR FROM ANG. VEL.
SET STATE VECTOR FROM ANG. POS.
(NULL)
SET DIMENSION OF STATE VECTOR
RETURN
END

ORIGINAL PAGE IS
OF POOR QUALITY

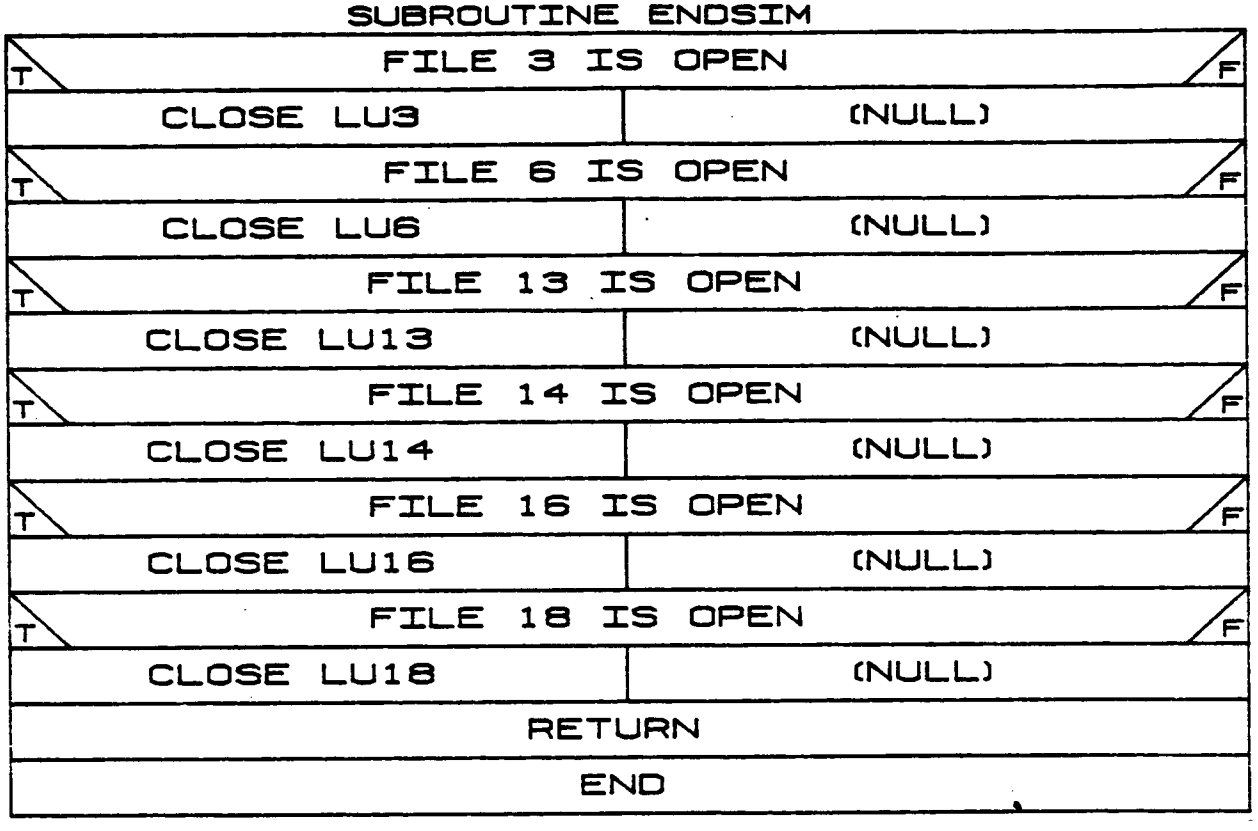
2.2.22 INTGRT

Subroutine INTGRT is called from RESPON and uses a fourth-order Runge-Kutta algorithm to integrate a state vector Z. State derivatives are computed by the subroutine DERIV.



2.2.23 ENDSIM

Subroutine ENDSIM closes any files opened during execution of the response simulation portion of ROBSIM.



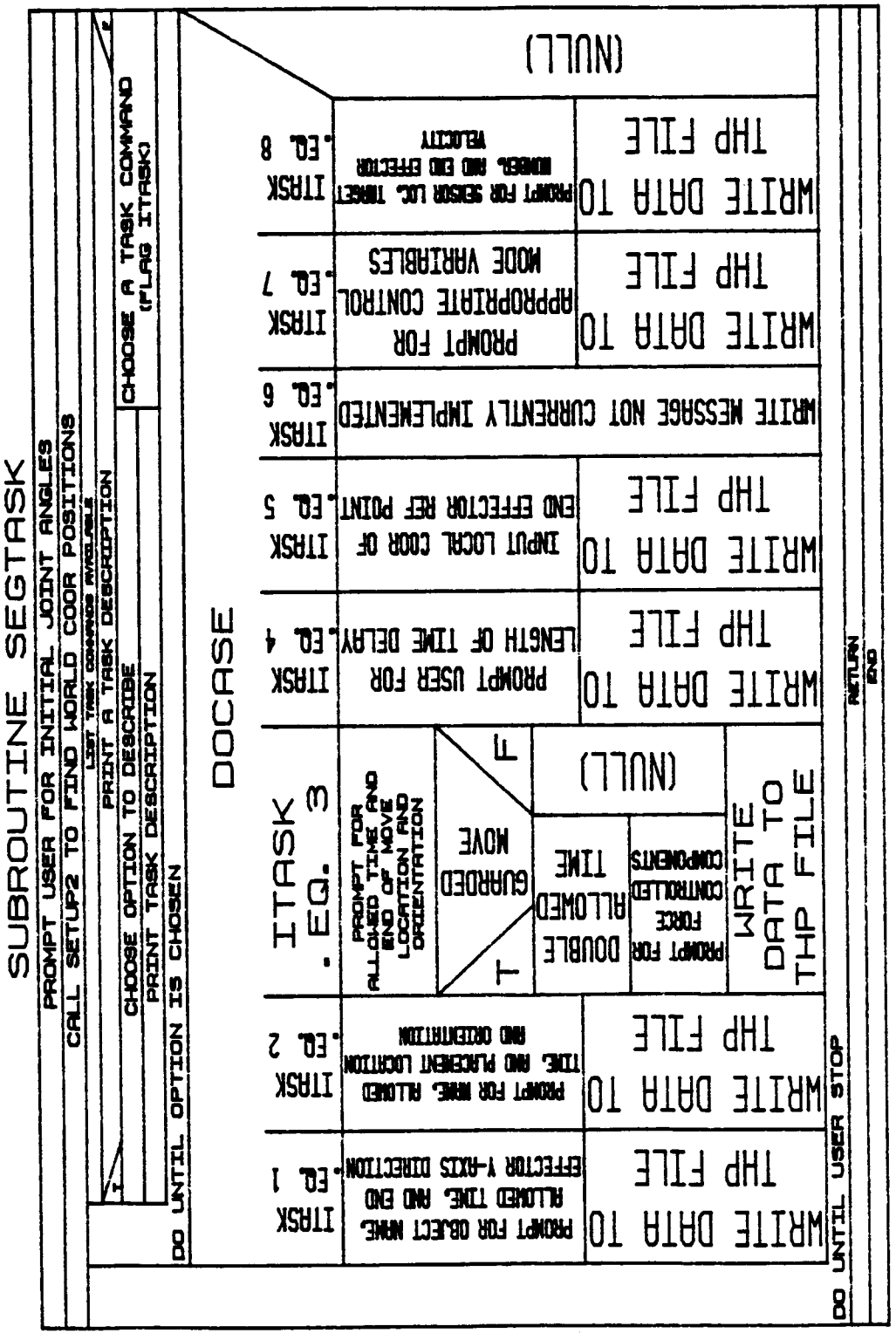
2.2.24 SEGTASK

Subroutine SEGTASK is called from TASK and allows the user to create a file that specifies manipulator motion using the following task commands:

1) Pick up object;
2) Place object at specified location;
3) Move arm;
4) Hold current position;
5) Change end effector reference point;
6) Operator control (not implemented yet);
7) Set control mode for response simulation;
8) Sensor of end effector position.

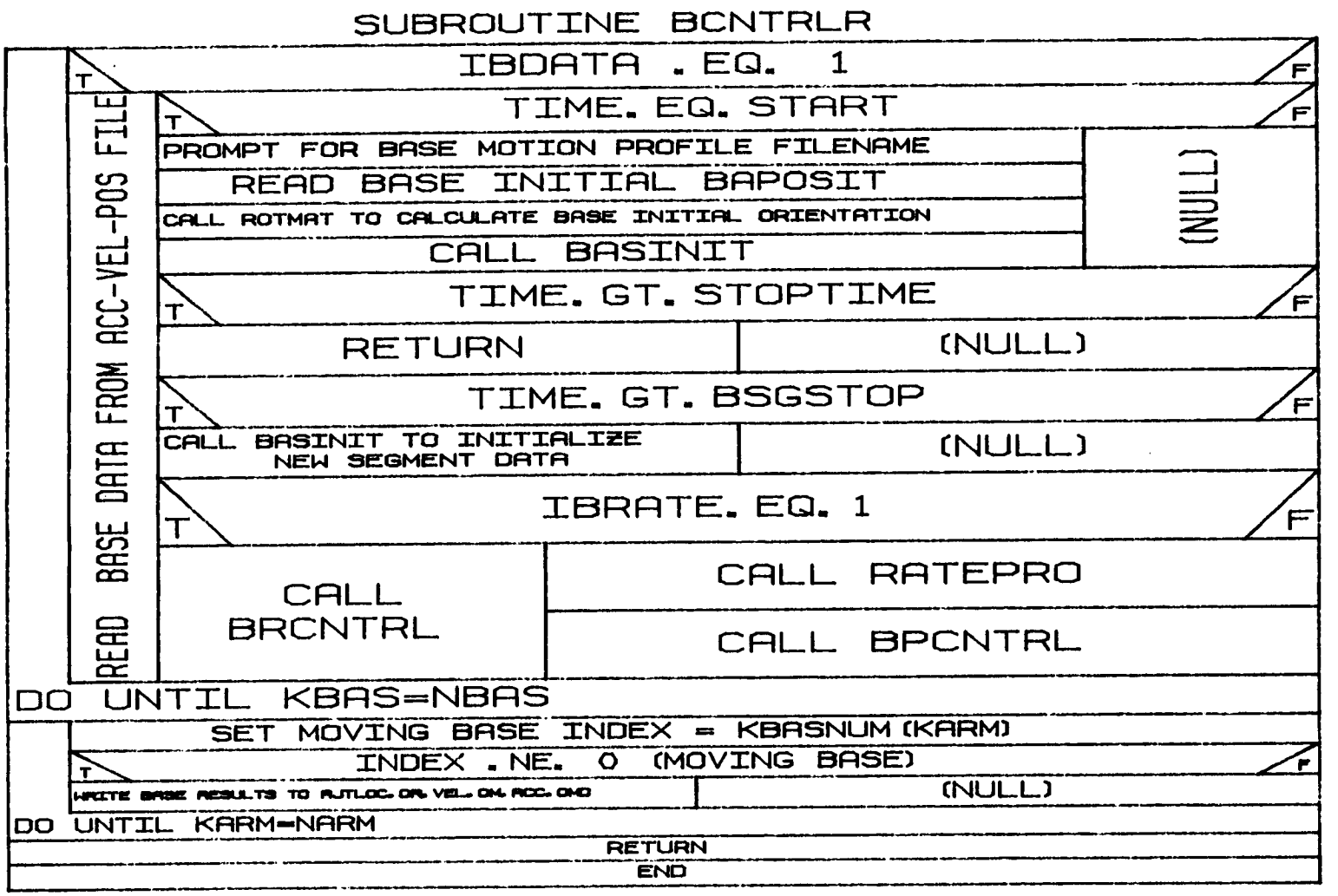
The user is prompted for initial joint angles and other necessary data after which a sequence of task commands may be implemented.

ORIGINAL PAGE IS
OF POOR QUALITY



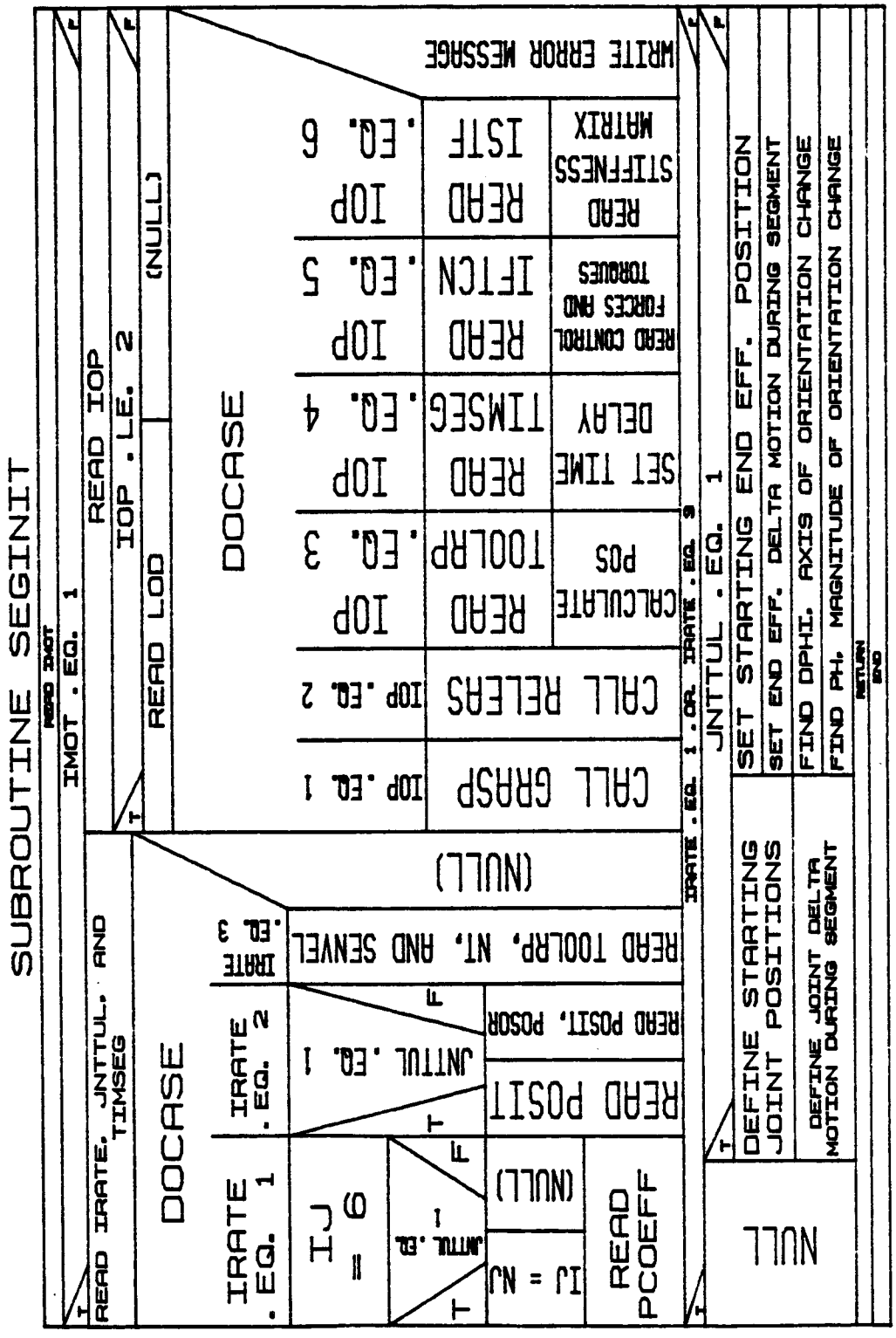
2.2.25 BCNTRLR

Subroutine BCNTRLR is called from REQUIR to obtain the position, velocity and acceleration for each moving base at each processing time step. If the variable IBDATA was set to 1 earlier, the base data are obtained by reading an existing file that contains just these data. If IBDATA equals 2, the base values are calculated from the base motion profiles. Subroutines RATEPRO and BPCNTRL are called for the base position control calculations and BRCNTRL is called for the base rate control calculation.


SUBROUTINE BCNTRLR
IBDATA . EQ. 1
T
F
READ BASE DATA FROM ACC-VEL-POS FILE
TIME. EQ. START
T
F
PROMPT FOR BASE MOTION PROFILE FILENAME
READ BASE INITIAL BAPOSIT
CALL ROTMAT TO CALCULATE BASE INITIAL ORIENTATION
CALL BASINIT
(NULL)
TIME. GT. STOPTIME
T
F
RETURN
(NULL)
TIME. GT. BSGSTOP
T
F
CALL BASINIT TO INITIALIZE NEW SEGMENT DATA
(NULL)
IBRATE. EQ. 1
T
F
CALL BRCNTRL
CALL RATEPRO
CALL BPCNTRL
DO UNTIL KBAS=NBAS
SET MOVING BASE INDEX = KBASNUM (KARM)
INDEX . NE. 0 (MOVING BASE)
T
F
WRITE BASE RESULTS TO AJTLOC, OR VEL, OR ACC, CMD
(NULL)
DO UNTIL KARM=NARM
RETURN
END

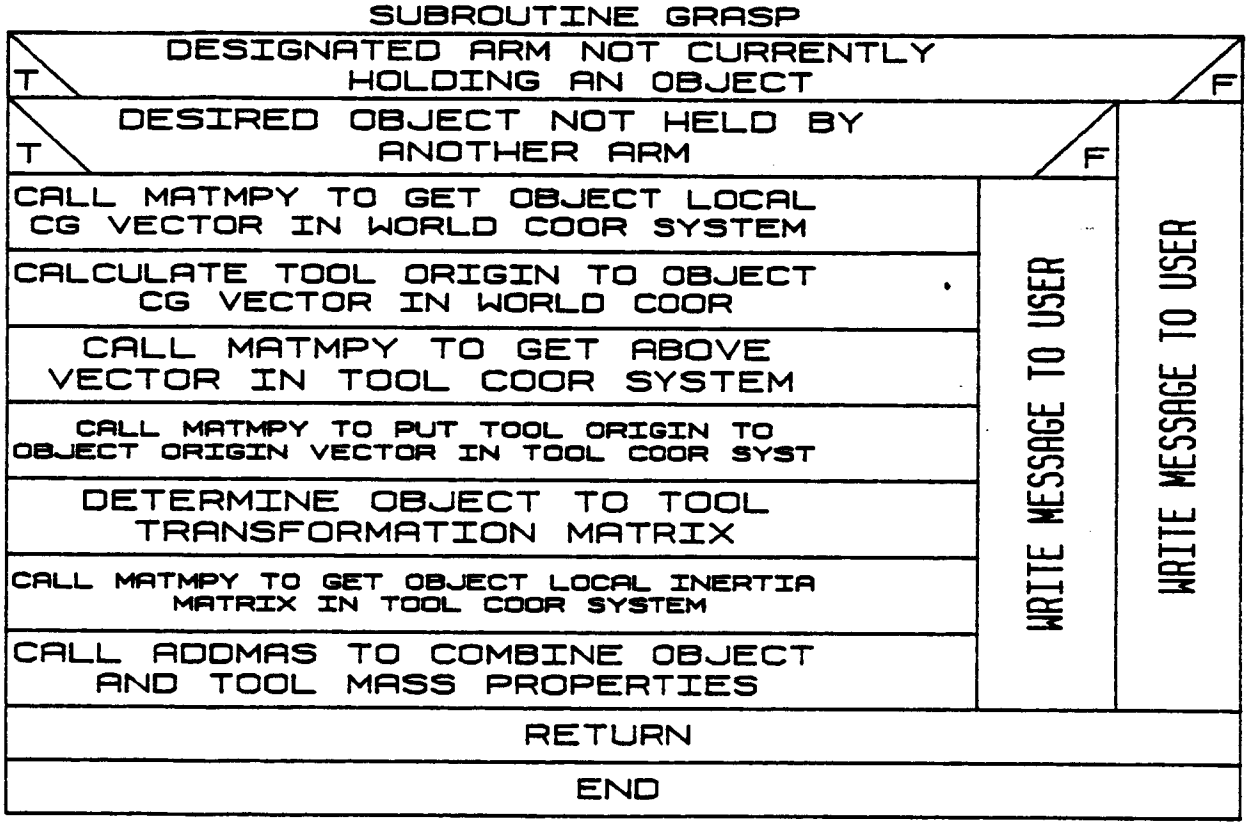
2.3.1 SEGINIT

Subroutine SEGINIT is called from CNTRLR at the beginning of each new motion profile segment. If the segment is to define motion of the manipulator, the sensor and target data or the coefficients of the polynomials defining the motion rates, the desired positions and orientations are read from the motion history file. If it is a nonmotion segment, the appropriate subroutines are called or variables are defined to ensure these actions are carried out. If position control is specified, the motion deltas for the current time segment are calculated.


SUBROUTINE SEGINIT
READ IMOT
IMOT .EQ. 1
READ IRATE, JNTTUL, AND TIMSEG
DOCASE
IRATE .EQ. 1
IJ = NJ
IJ = 0
JNTTUL .EQ. 1
READ PCOEFF
(NULL)
READ POSIT
JNTTUL .EQ. 1
READ POSIT, POSOR
IRATE .EQ. 2
READ TOOLRP, NT, AND SENVEL
IRATE .EQ. 3
(NULL)
READ IOP
IOP .LE. 2
READ LOD
(NULL)
DOCASE
CALL GRASP
IOP .EQ. 1
CALL RELEAS
IOP .EQ. 2
CALCULATE POS
READ TOOLRP
IOP .EQ. 3
SET TIME DELAY
READ TIMSEG
IOP .EQ. 4
READ CONTROL FORCES AND TORQUES
READ IFTCN
IOP .EQ. 5
READ STIFFNESS MATRIX
READ ISTF
IOP .EQ. 6
WRITE ERROR MESSAGE
IRATE .EQ. 1 .OR. IRATE .EQ. 3
JNTTUL .EQ. 1
NULL
DEFINE STARTING JOINT POSITIONS
DEFINE JOINT DELTA MOTION DURING SEGMENT
SET STARTING END EFF. POSITION
SET END EFF. DELTA MOTION DURING SEGMENT
FIND DPHI, AXIS OF ORIENTATION CHANGE
FIND PHI, MAGNITUDE OF ORIENTATION CHANGE
RETURN
END

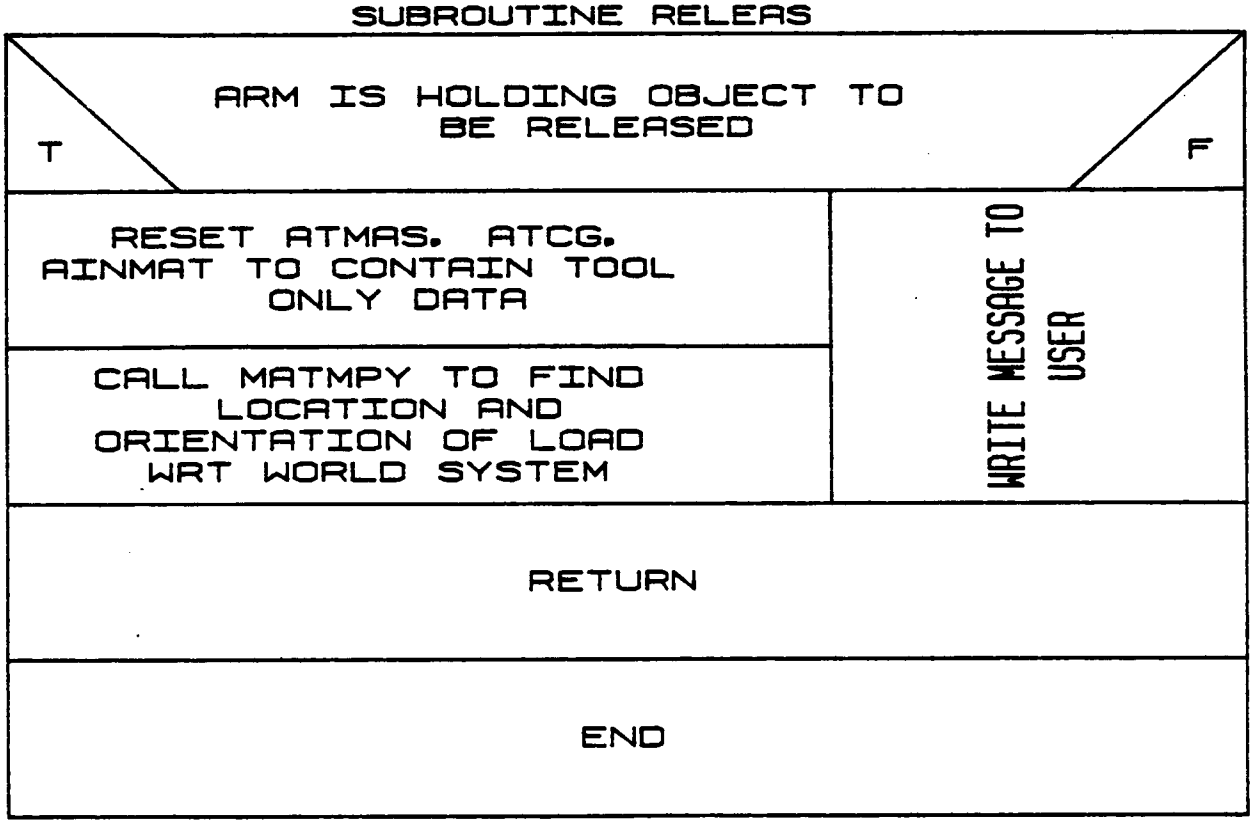
2.3.2 GRASP

Subroutine GRASP is called from SEGINIT when the option flag IOP equals 1 (this denotes that the arm is to grasp a load object). The subroutine first checks to make sure the arm is not already holding an object and that the desired object is not being held by another arm. The location and orientation of the load object are then defined with respect to the end-effector coordinate system. This ensures that the object's location and orientation with respect to the world coordinate system will be updated correctly during a move and that the display shows the object moving with the arm. The end-effector mass properties are modified to include the load object to ensure the correct system response.



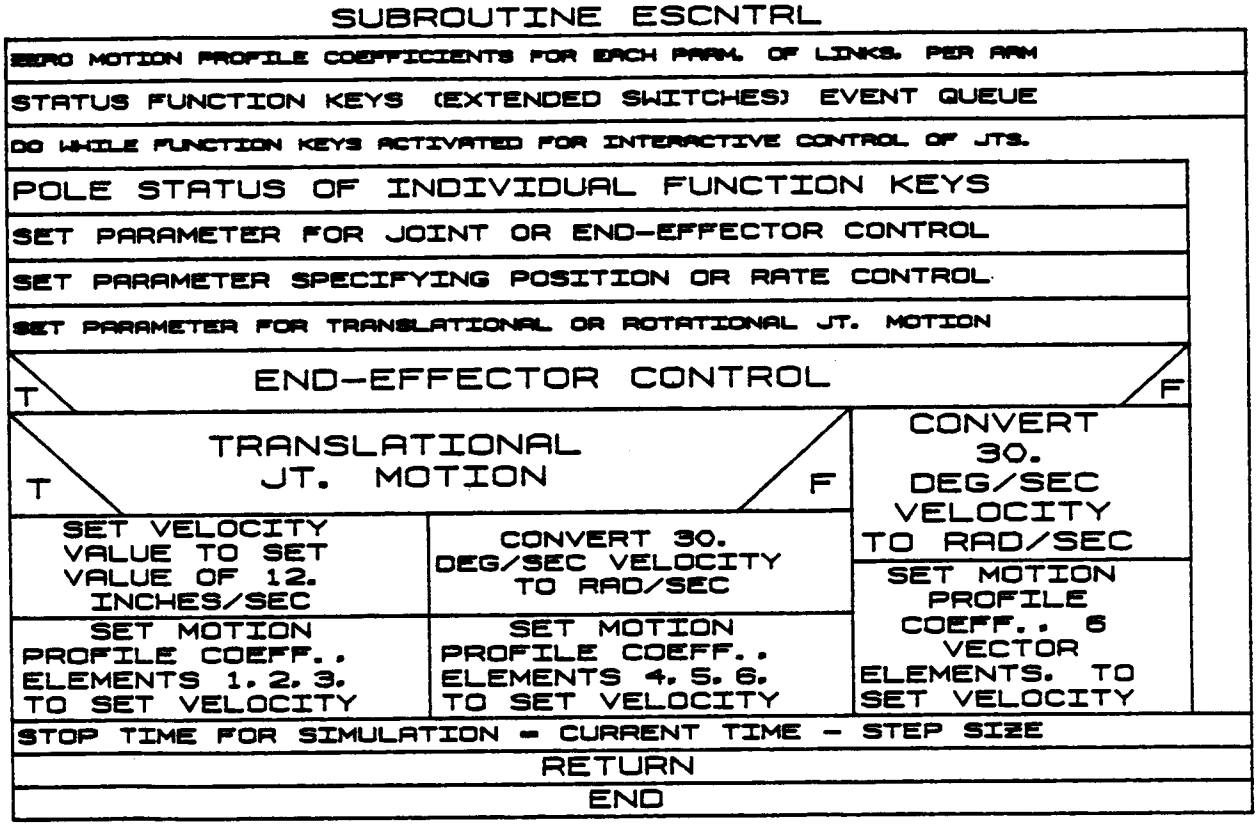
2.3.3 RELEAS

Subroutine RELEAS is called from SEGINIT when the arm is to let go of a load object. The routine first makes sure the load object to be released is being held by the current arm. If it is being held, the end-effector mass properties are reset to the values held before the object was picked up and MATMPY is called to obtain the location and orientation of the load object with respect to the world. If the object is not being held, a message is displayed to the user.



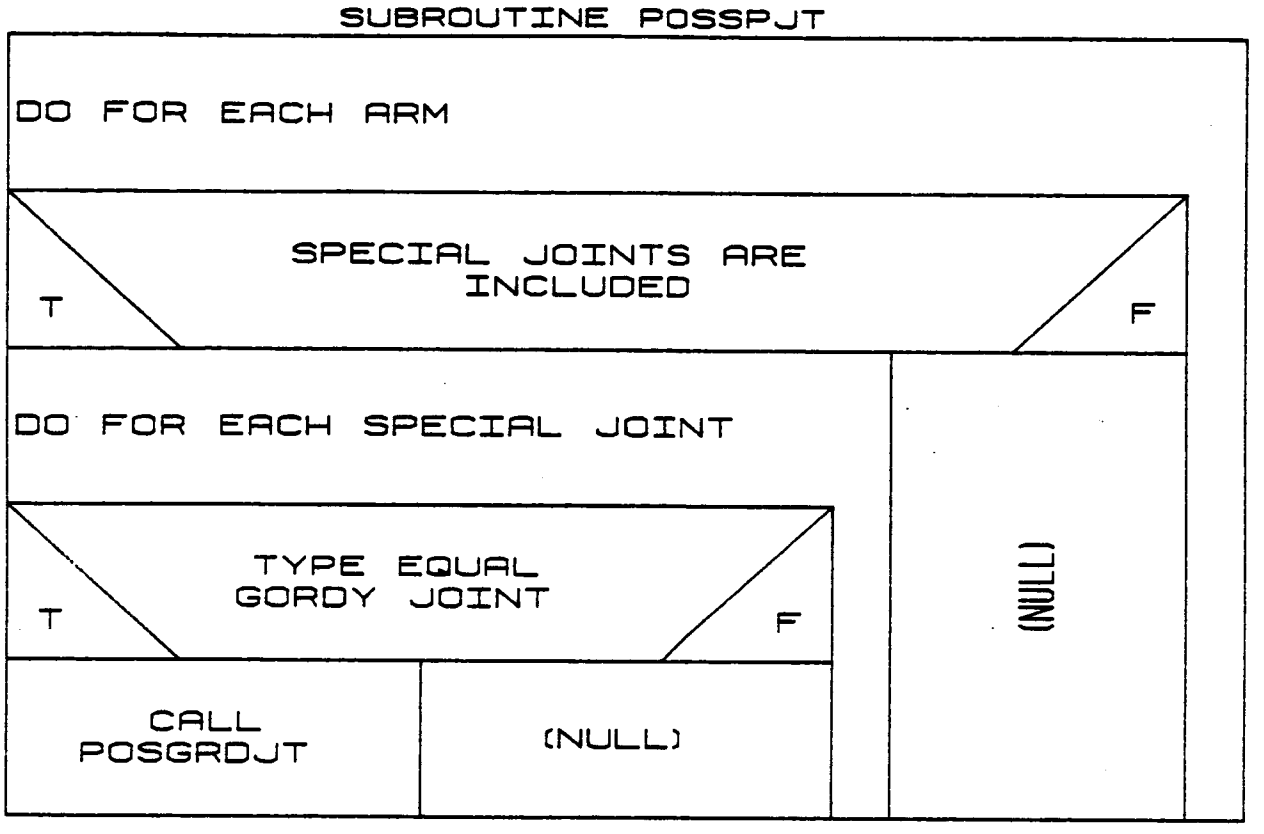
2.3.4 ESCNTRL

ESCNTRL allows control of system motion through use of the Evans and Sutherland extended switches (function keys) and loads the coefficients of the polynomial describing the motion, PCOEF. Options include individual joint control or end-effector control. For end-effector control, either the base coordinates or tool coordinates may be used as the reference frame. Also, the controlled motion may be either translation or rotation. The manipulator motion is always rate-controlled rather than joint-controlled. The user may select which arm and joint is to move.



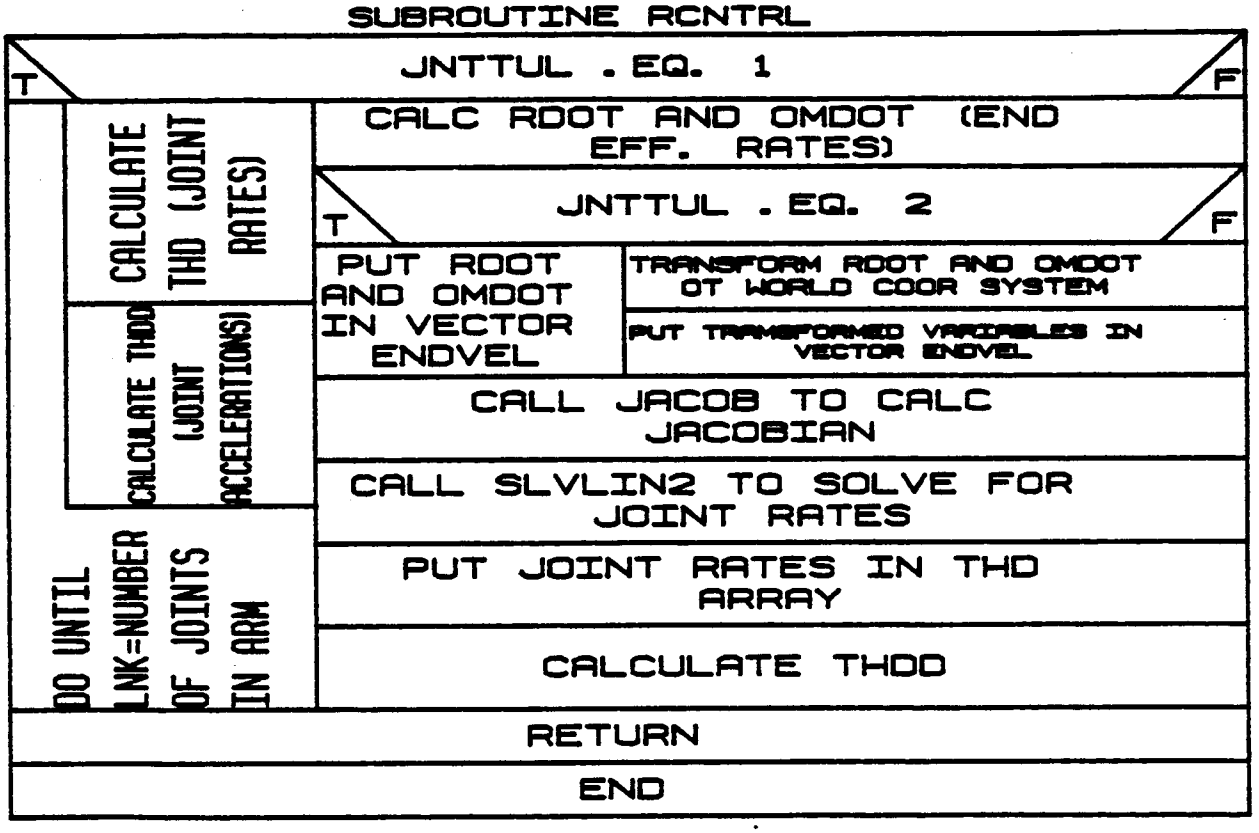
2.3.5 POSSPJT

POSSPJT is the executive routine that calls handling routines for finding the position of special joints within the manipulator. Currently only one type of special joint can be included.



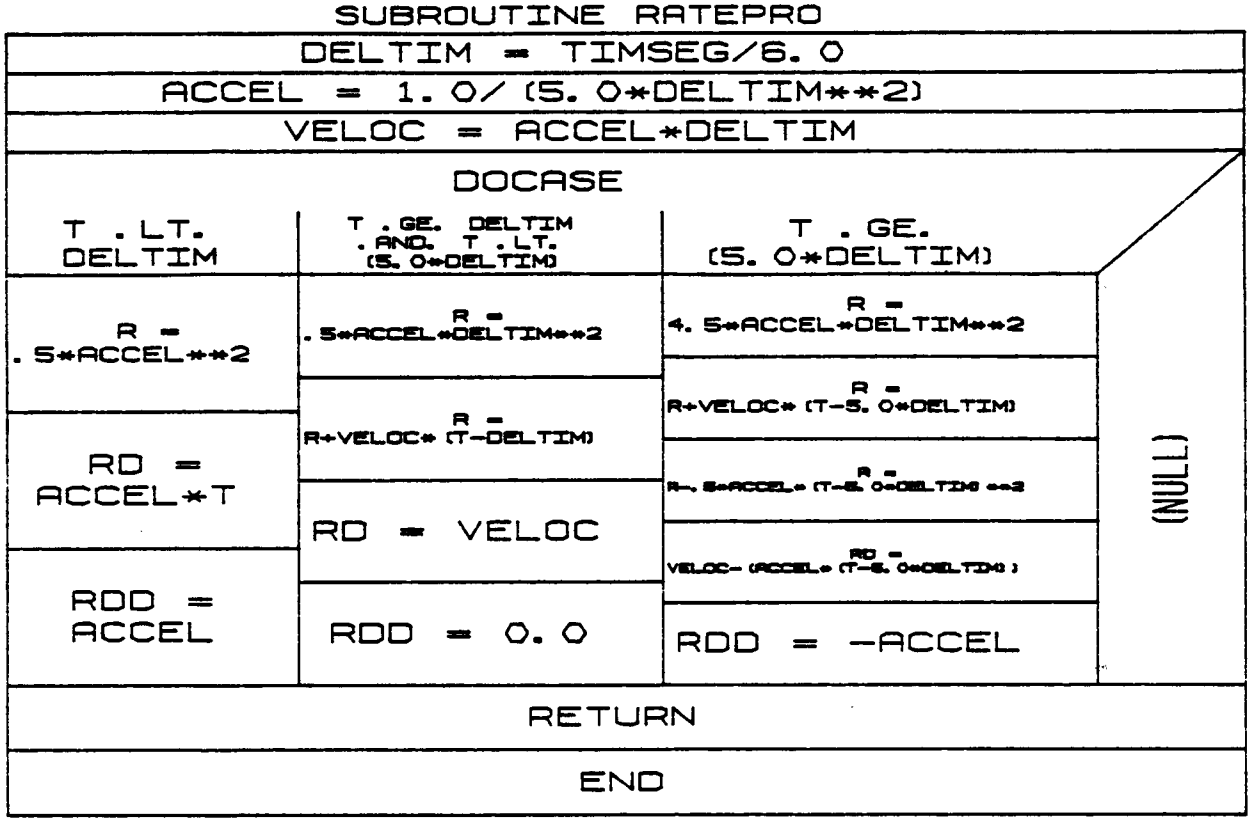
2.3.6 RCNTRL

Subroutine RCNTRL is called from CNTRLR when rate control of the joints or end-effector is specified. If joint rate control was chosen, the joint rates THD are calculated from their polynomial definitions and the accelerations THDD by finite difference methods. For end-effector rate control, the defining polynomials are evaluated for the current time. JACOB and SLVLIN2 are then called to transform these end-effector rates to individual joint rates. Accelerations (of each joint) are again calculated using finite difference methods.



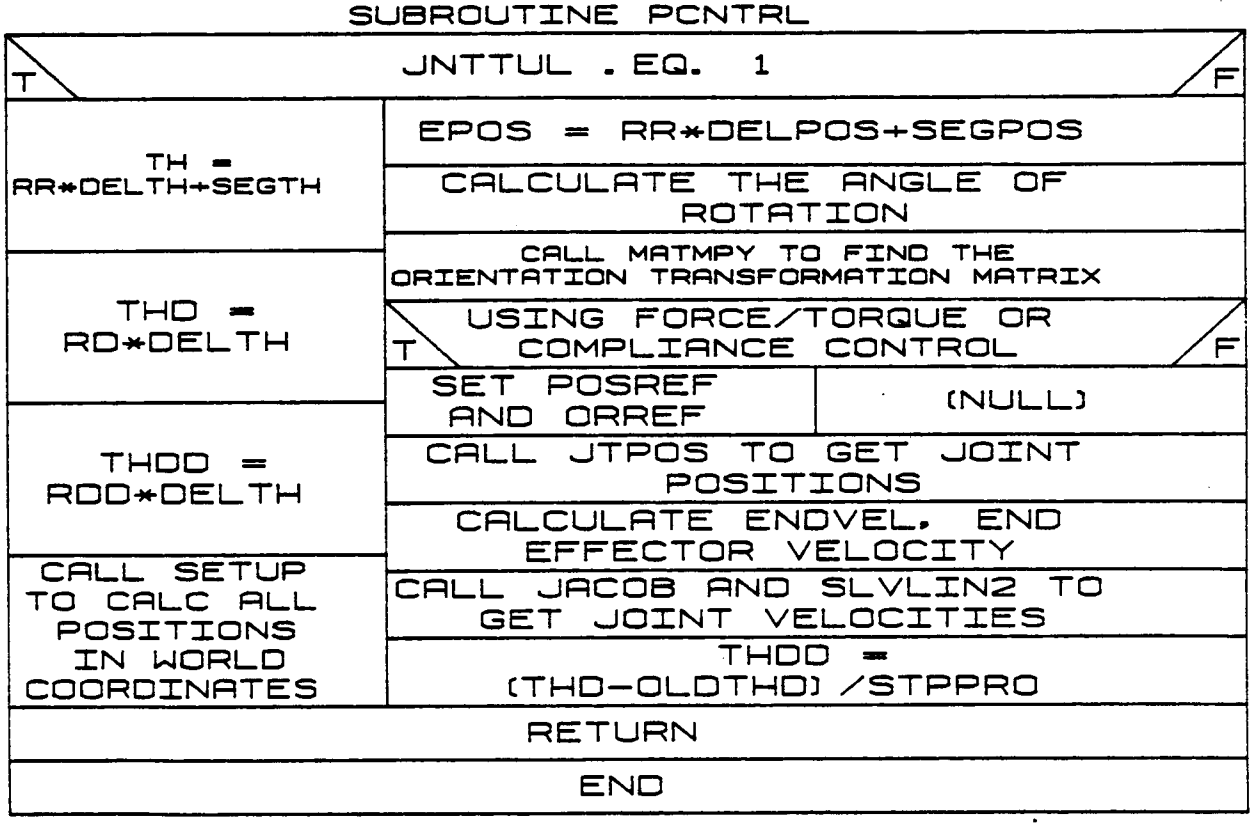
2.3.7 RATEPRO

RATEPRO is called from subroutine CNTRLR when position control of the manipulator is desired. The time allowed for the move is divided into six equal portions. The first portion is defined to be constant acceleration. The next four are constant velocity. The last is constant deceleration equal in magnitude to the first portion. The distance traveled in the whole time is set to 1 and the appropriate distance traveled, velocity and acceleration for each portion are calculated.



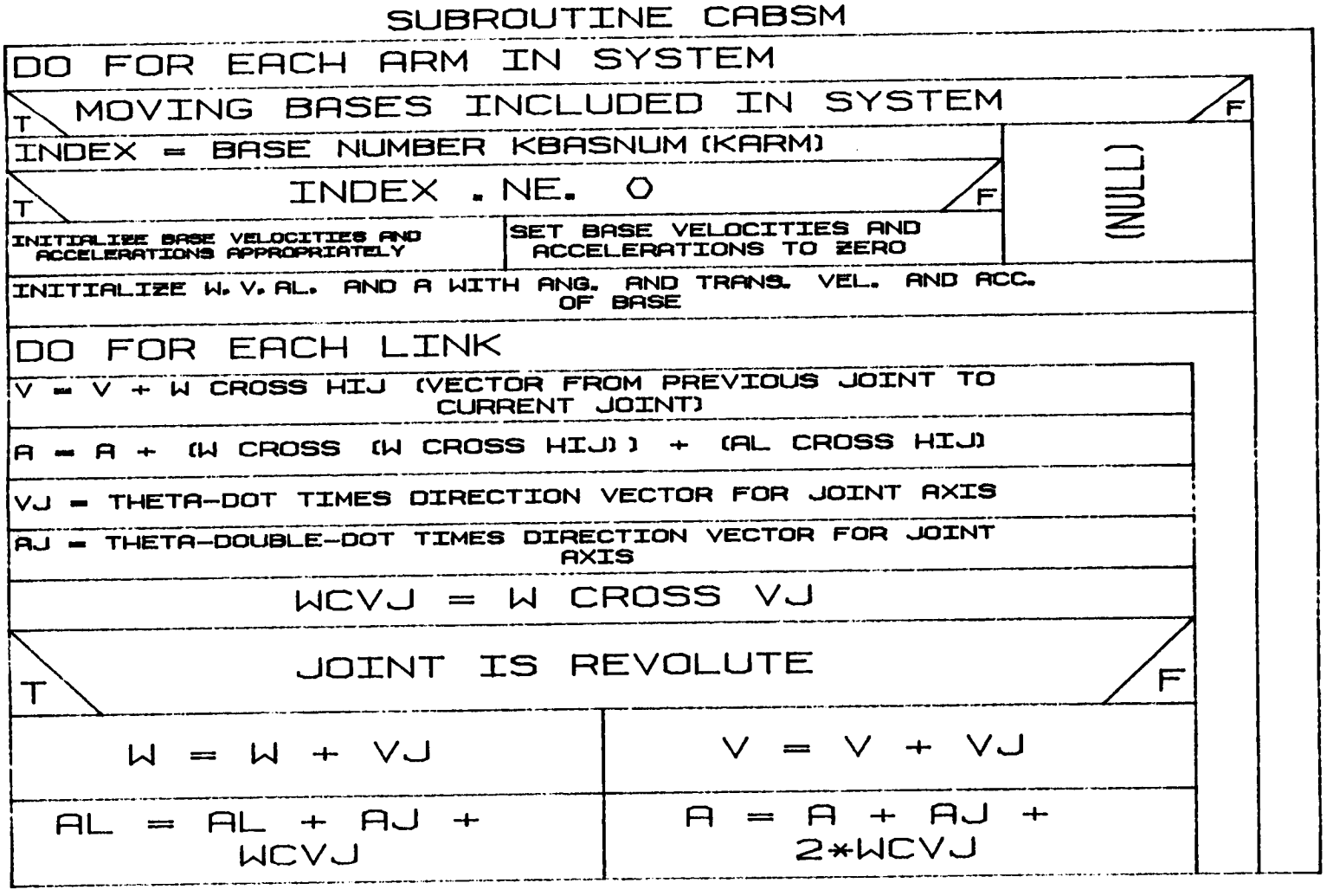
2.3.8 PCNTRL

PCNTRL is called from CNTRLR when position control of the manipulator is to be used. Joint position control uses the segment rate profile defined by subroutine RATEPRO to calculate the joint positions, velocities, and accelerations. End-effector position control uses the same rate profile to get the end-effector rates. JTPOS is then called to get joint positions, and JACOB and SLVLIN2 are called to get the joint velocities. Joint accelerations are calculated by finite difference methods.



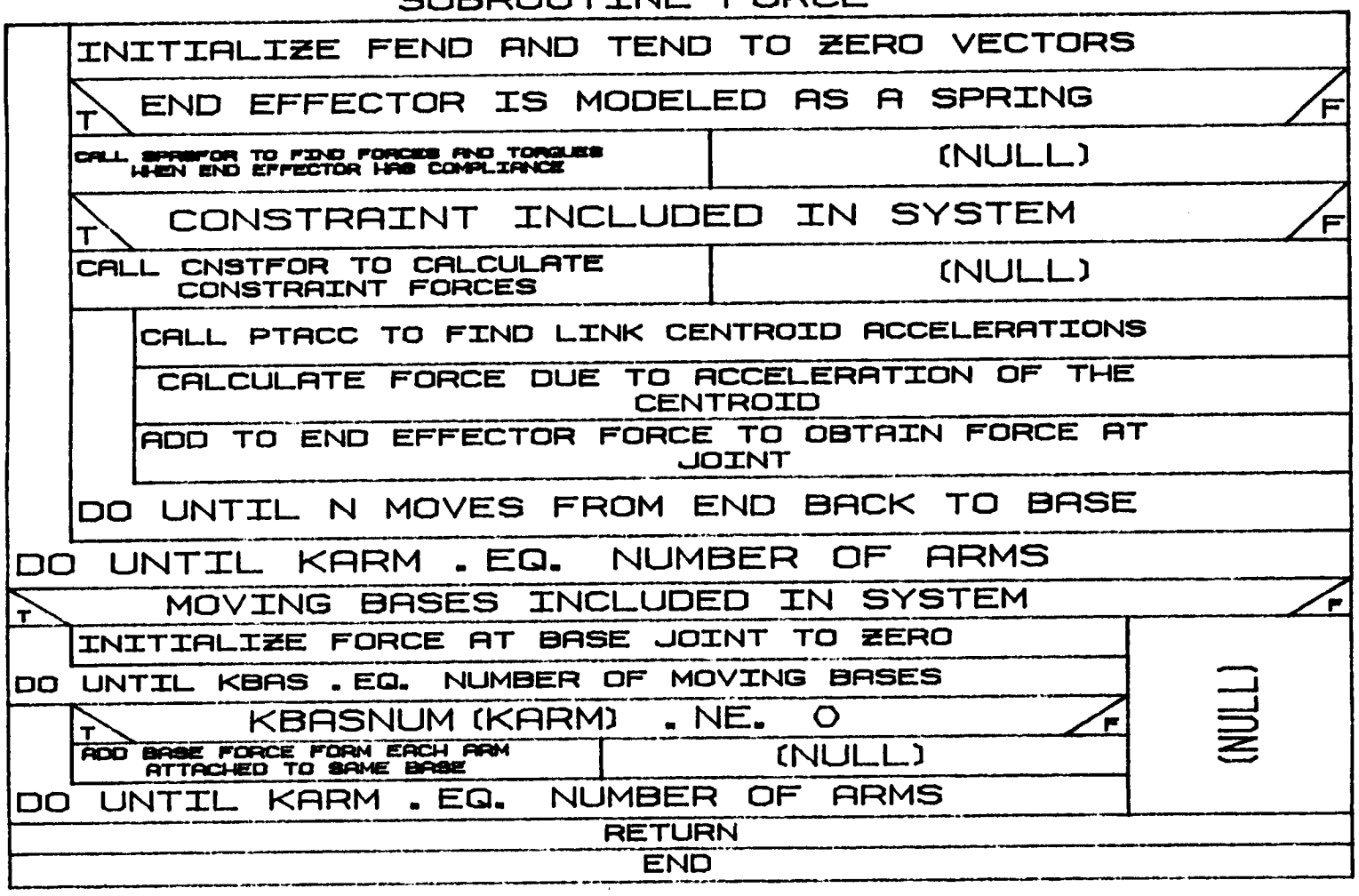
2.3.9 CABSM

CABSM uses a recursive technique to compute the absolute angular and translational velocity and acceleration of each joint/link combination in the system.


SUBROUTINE CABSM
DO FOR EACH ARM IN SYSTEM
MOVING BASES INCLUDED IN SYSTEM
T
F
INDEX = BASE NUMBER KBASNUM (KARM)
INDEX .NE. 0
T
F
(NULL)
INITIALIZE BASE VELOCITIES AND ACCELERATIONS APPROPRIATELY
SET BASE VELOCITIES AND ACCELERATIONS TO ZERO
INITIALIZE W, V, AL, AND A WITH ANG. AND TRANS. VEL. AND ACC. OF BASE
DO FOR EACH LINK
V = V + W CROSS HIJ (VECTOR FROM PREVIOUS JOINT TO CURRENT JOINT)
A = A + (W CROSS (W CROSS HIJ)) + (AL CROSS HIJ)
VJ = THETA-DOT TIMES DIRECTION VECTOR FOR JOINT AXIS
AJ = THETA-DOUBLE-DOT TIMES DIRECTION VECTOR FOR JOINT AXIS
WCVJ = W CROSS VJ
JOINT IS REVOLUTE
T
F
W = W + VJ
V = V + VJ
AL = AL + AJ + WCVJ
A = A + AJ + 2*WCVJ

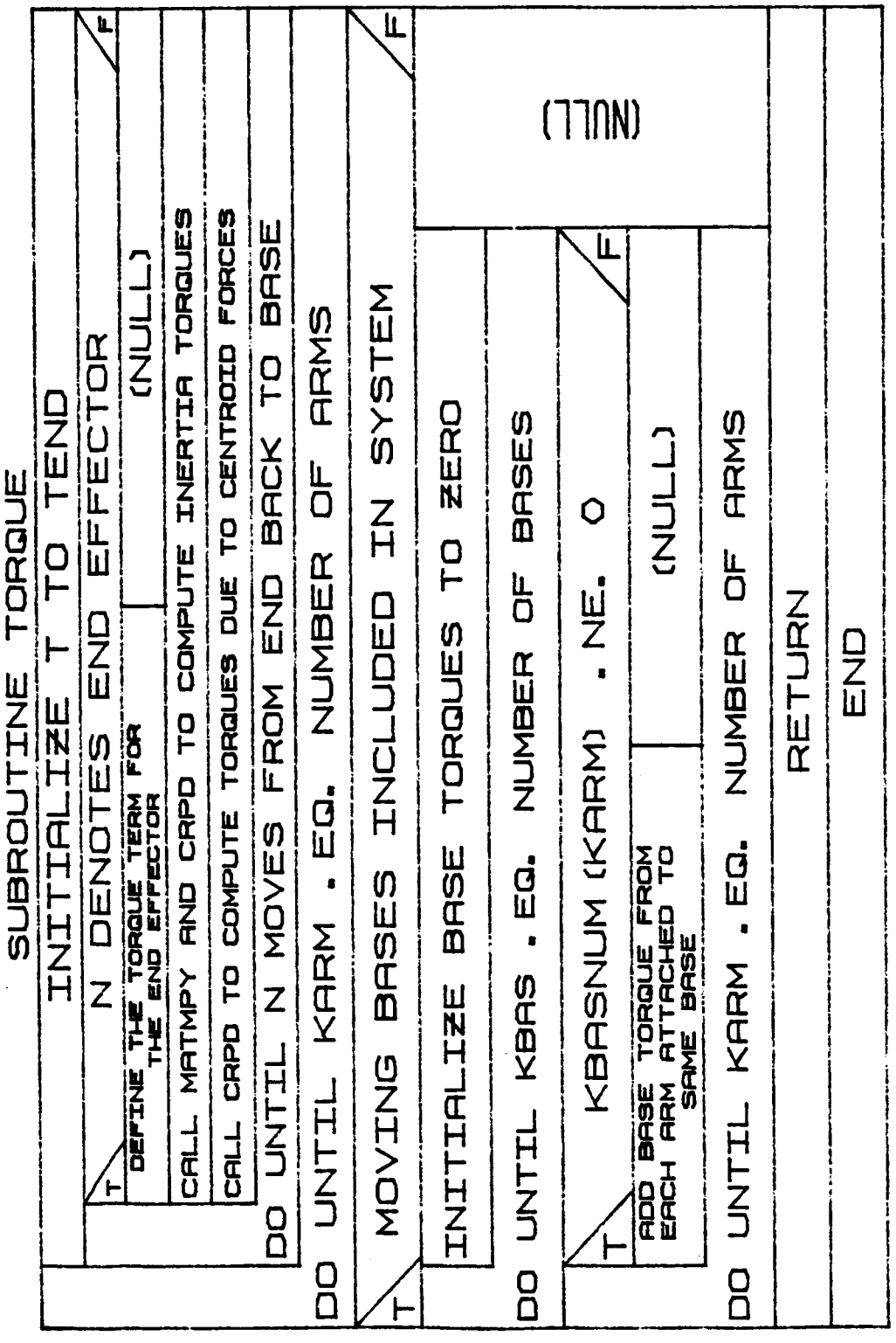
2.3.10 FORCE

Subroutine FORCE is called from DYNAM to calculate the force exerted on each joint. The force at the end-effector is determined first. PTACC is called to find link centroid accelerations, and the forces caused by these accelerations are added to the end-effector forces to find the force at each joint. If the system includes multiple arms on a moving base, the reaction force at the base is a sum of the individual base reaction forces from each arm that is attached to the base.


SUBROUTINE FORCE
INITIALIZE FEND AND TEND TO ZERO VECTORS
END EFFECTOR IS MODELED AS A SPRING
T
F
CALL SPRNFOR TO FIND FORCES AND TORQUES WHEN END EFFECTOR HAS COMPLIANCE
(NULL)
CONSTRAINT INCLUDED IN SYSTEM
T
F
CALL CNSTFOR TO CALCULATE CONSTRAINT FORCES
(NULL)
CALL PTACC TO FIND LINK CENTROID ACCELERATIONS
CALCULATE FORCE DUE TO ACCELERATION OF THE CENTROID
ADD TO END EFFECTOR FORCE TO OBTAIN FORCE AT JOINT
DO UNTIL N MOVES FROM END BACK TO BASE
DO UNTIL KARM .EQ. NUMBER OF ARMS
MOVING BASES INCLUDED IN SYSTEM
T
F
INITIALIZE FORCE AT BASE JOINT TO ZERO
DO UNTIL KBAS .EQ. NUMBER OF MOVING BASES
KBASNUM (KARM) .NE. 0
T
F
ADD BASE FORCE FORM EACH ARM ATTACHED TO SAME BASE
(NULL)
DO UNTIL KARM .EQ. NUMBER OF ARMS
(NULL)
RETURN
END

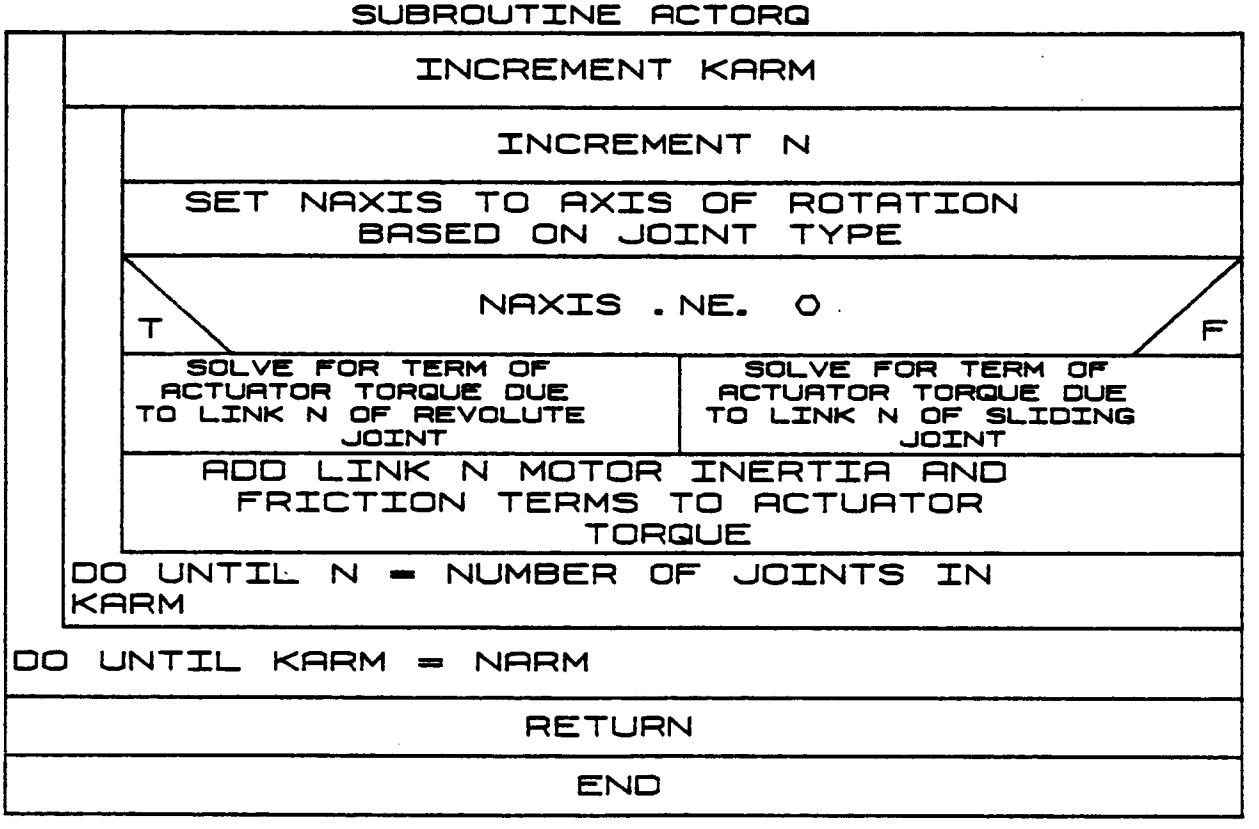
2.3.11 TORQUE

Subroutine TORQUE is called from DYNAM to calculate individual joint torques. The torques at the end-effector are determined first. Torques at the preceding joint are then calculated by adding the torques attributable to link inertias and centroid forces to the end-effector torques. The routine works back toward the base of the manipulator, adding the torques caused by inertias and centroid forces to the cumulative torques thus far to obtain the current joint torques. If the system includes multiple arms on a moving base, the total reaction torque at the base is the sum of all the individual reaction torques at the base from each arm that is attached to the base.


SUBROUTINE TORQUE
INITIALIZE T TO TEND
N DENOTES END EFFECTOR
F
T
DEFINE THE TORQUE TERM FOR THE END EFFECTOR
(NULL)
CALL MATMPY AND CRPD TO COMPUTE INERTIA TORQUES
CALL CRPD TO COMPUTE TORQUES DUE TO CENTROID FORCES
DO UNTIL N MOVES FROM END BACK TO BASE
DO UNTIL KARM . EQ. NUMBER OF ARMS
MOVING BASES INCLUDED IN SYSTEM
F
T
INITIALIZE BASE TORQUES TO ZERO
(NULL)
DO UNTIL KBAS . EQ. NUMBER OF BASES
KBASNUM (KARM) . NE. 0
F
T
ADD BASE TORQUE FROM EACH ARM ATTACHED TO SAME BASE
(NULL)
DO UNTIL KARM . EQ. NUMBER OF ARMS
RETURN
END

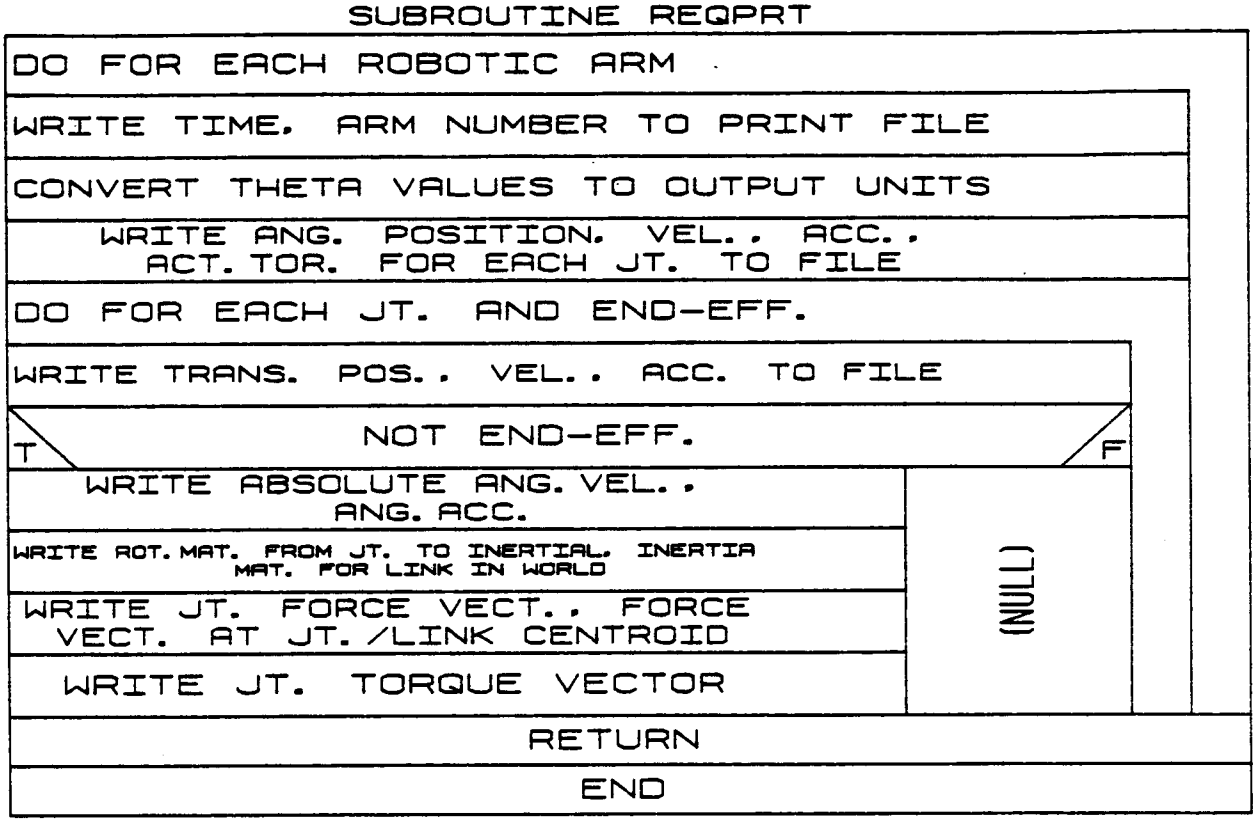
2.3.12 ACTORQ

Subroutine ACTORQ calculates actuator drive torques for each joint. It is called from REQUIR when running requirements analysis and from NLINK when running response simulation. This routine first solves for the free axis of each joint and the component of joint reaction torque about this axis. The torque needed to overcome inertia and viscous and dry friction are added to the joint reaction torques to obtain a total actuator drive torque for each joint.



2.3.13 REQPRT

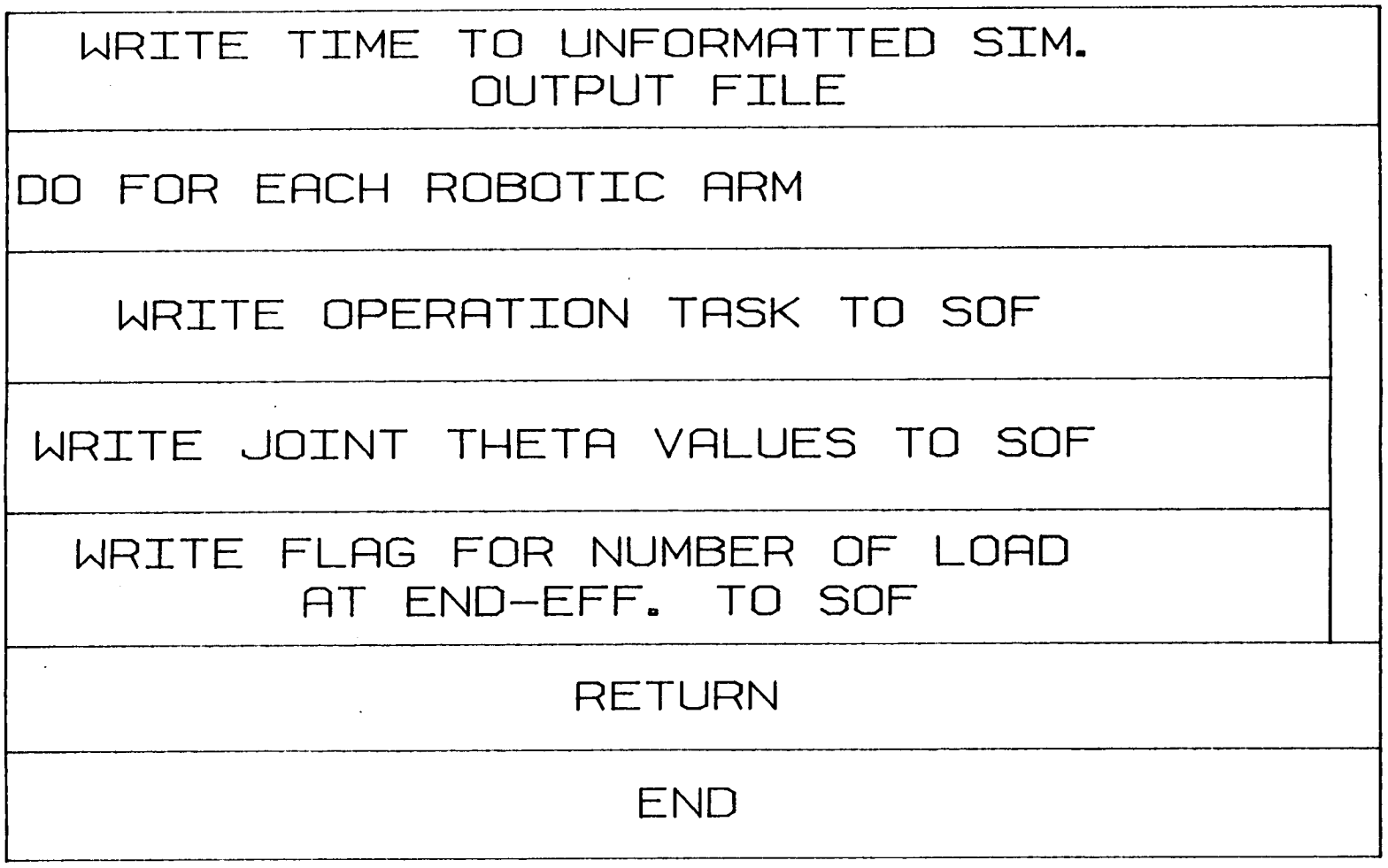
Subroutine REQPRT is called from OUTREQ to write run data to an output file if this option was requested by the user. Data written to this file includes time, angular position, velocity and acceleration, translational position, velocity and acceleration, and joint force and torque vectors.



2.3.14 REQSOF

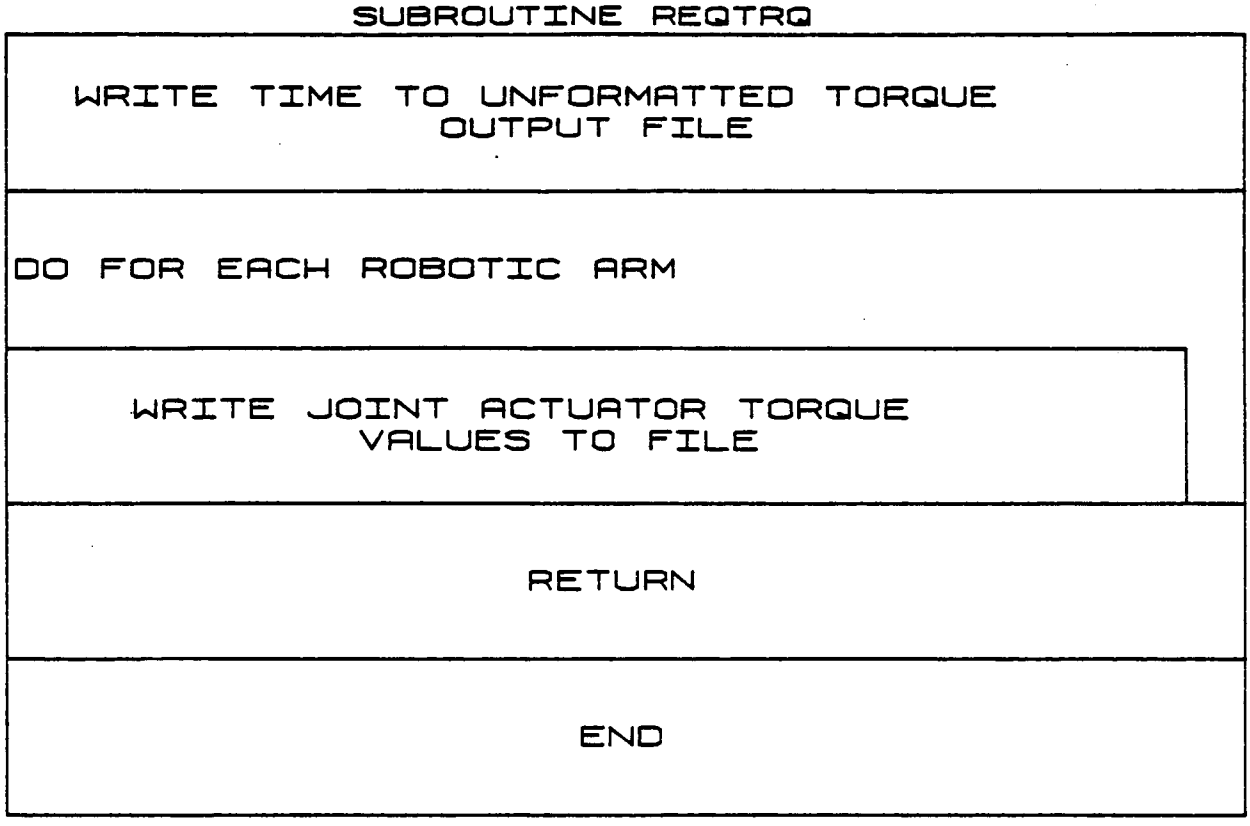
REQSOF is called from OUTREQ to write a simulation playback file if this option was requested by the user. The simulation playback file contains joint angular positions, task commands and load objects flags as a function of time and is used to replay the motion that occurred during a requirements analysis run without doing the calculations normally associated with that run.

SUBROUTINE REQSOF



2.3.15 REQTRQ

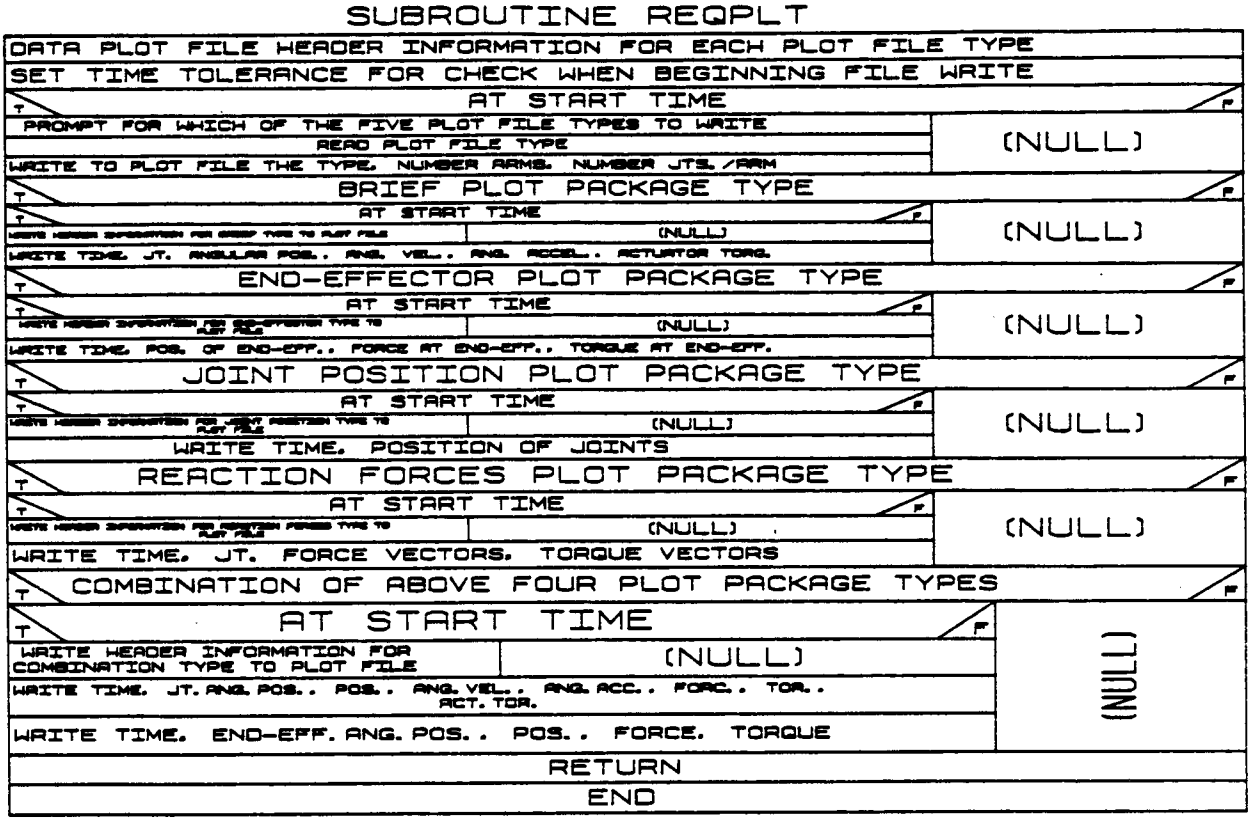
REQTRQ is called from OUTREQ to write a file of actuator torques as a function of time if this option was chosen by the user. These data may then be used to run a response simulation run.



ORIGINAL PAGE IS
OF POOR QUALITY

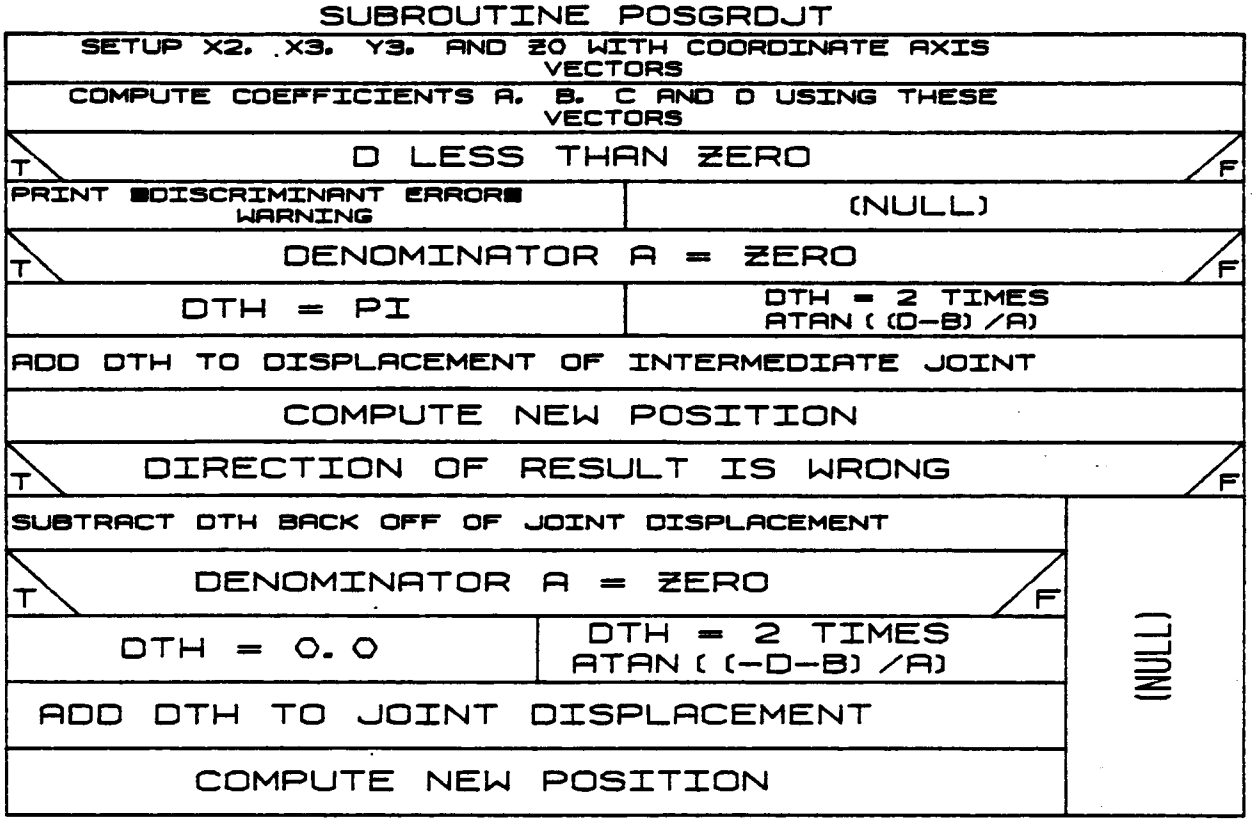
2.3.16 REQPLT

Subroutine REQPLT is called from OUTREQ to write a file of various manipulator parameters as a function of time during a requirements analysis run. This file may then be used to create x-y plots of these parameters as a function of time.



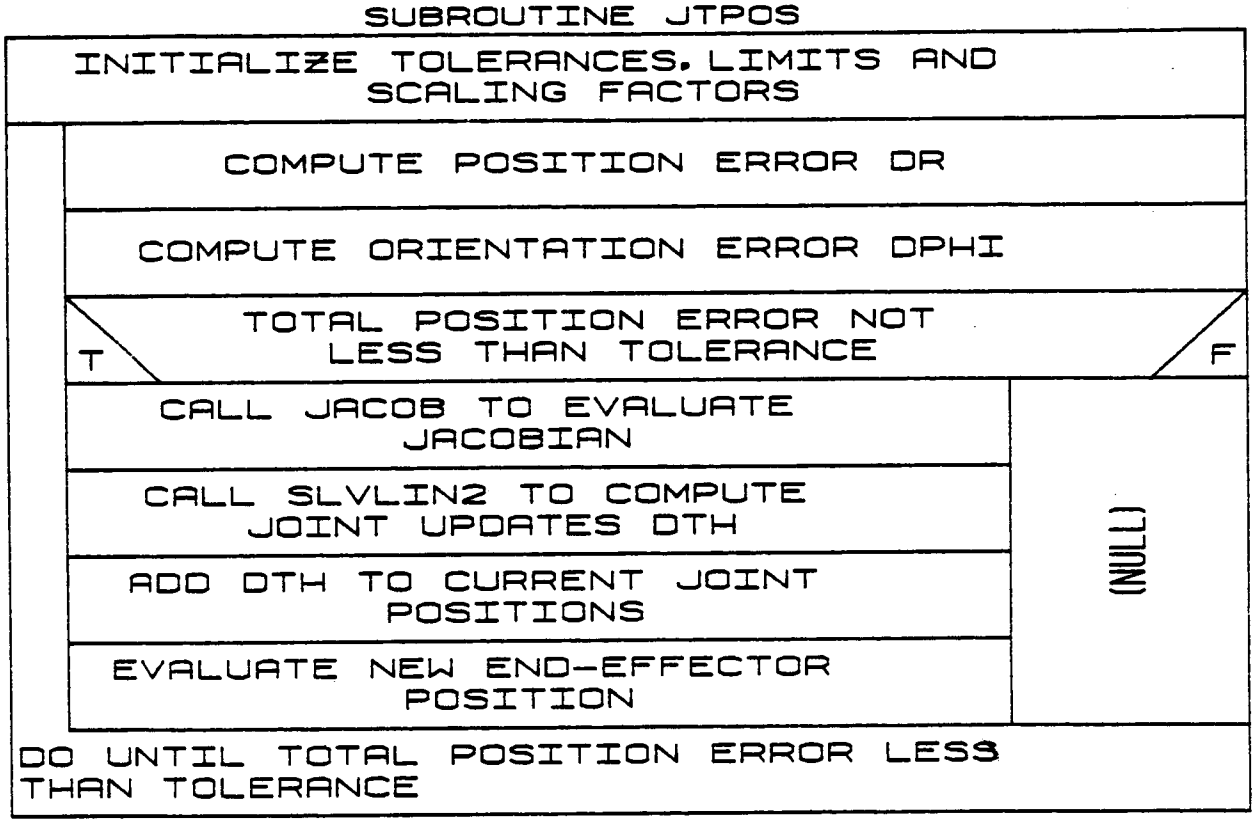
2.3.17 POSGRDJT

POSGRDJT computes the position of the intermediate joint in a special joint combination called a "Gordy Joint." This position is selected to satisfy a constraint on the three joints in this combination.



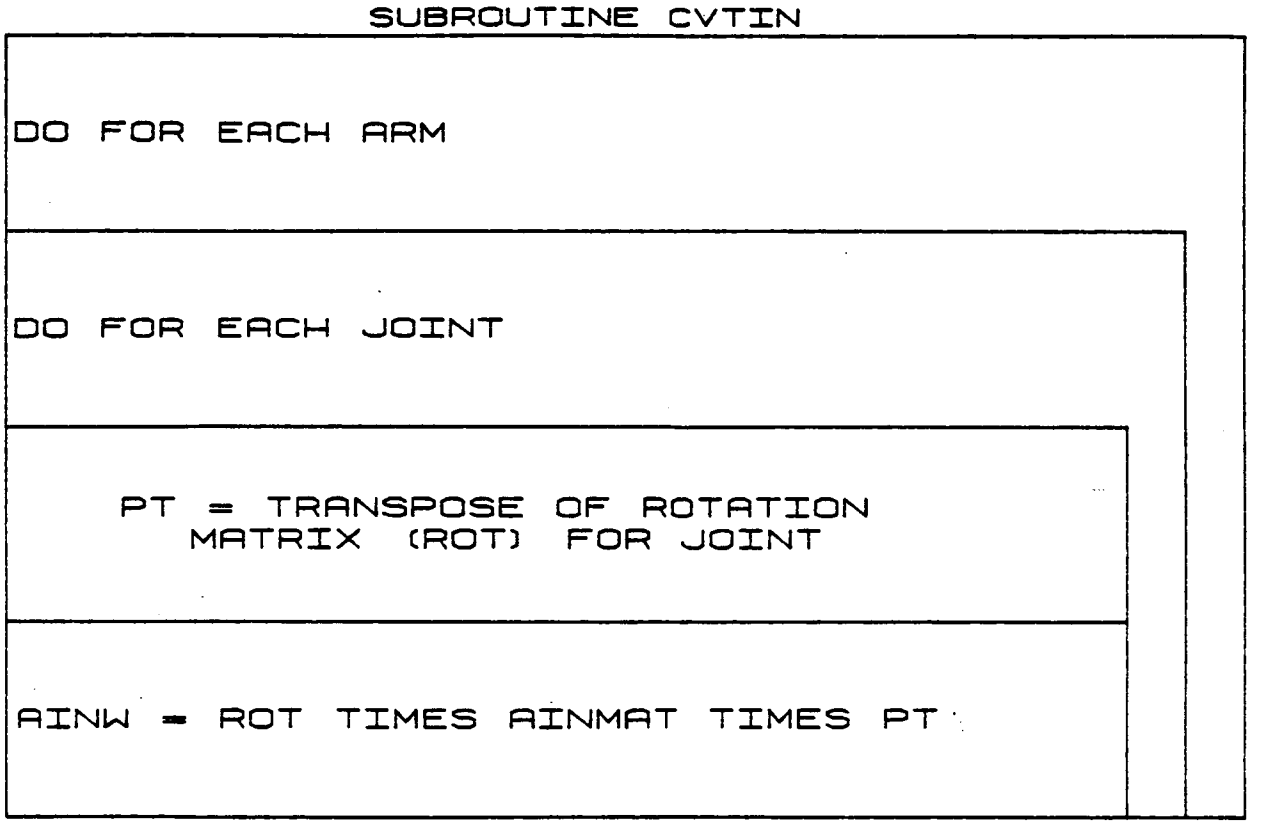
2.3.18 JTPOS

JTPOS is an iterative routine for finding a set of joint angles corresponding to a desired hand position and orientation. The error DPOS in position is calculated and then ORERR is called to find the orientation error and transform it into a rotation vector. This rotation vector is combined with DPOS, giving DP. The Jacobian relating hand motion to joint motion is computed and the set of six linear equations [J](DTG) - (DP) is solved for the joint updates DTH. This procedure is repeated until the desired position is obtained.



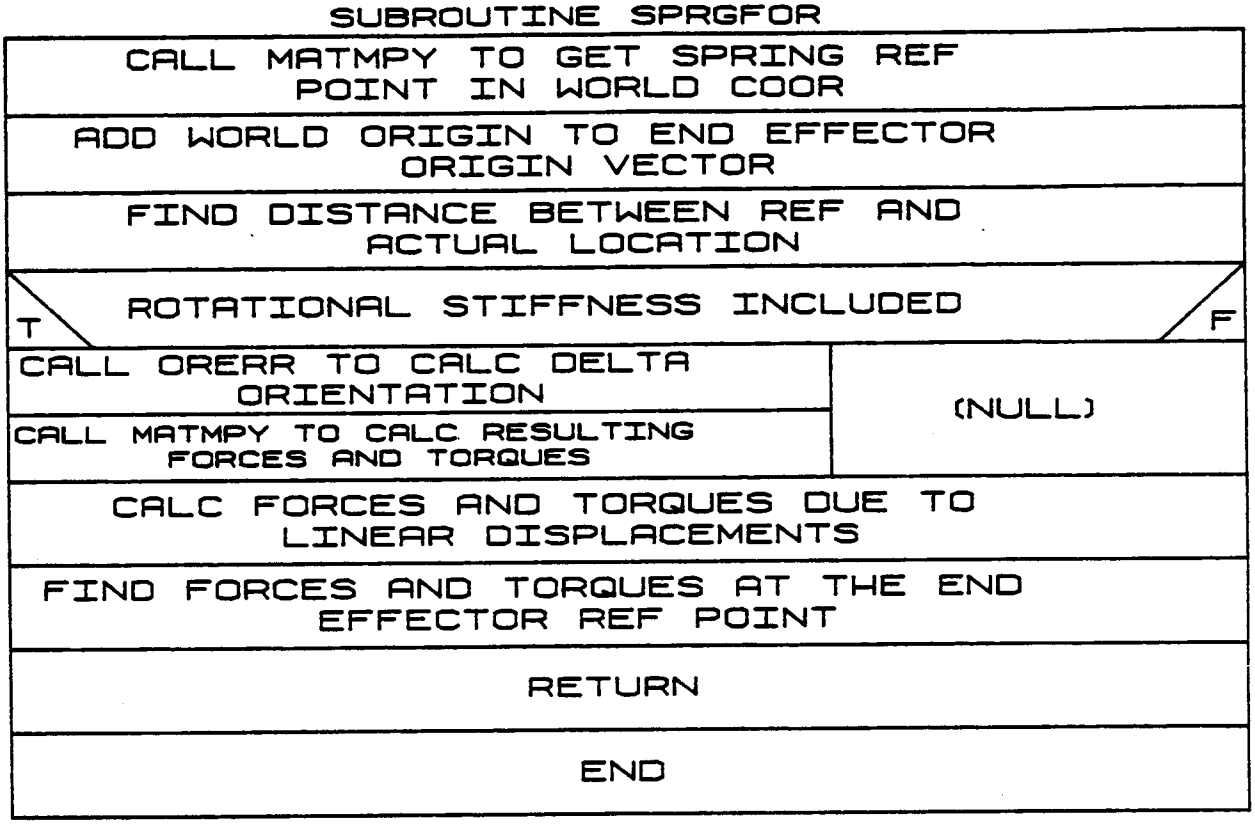
2.3.19 CVTIN

CVTIN transforms link inertia matrices from local coordinates into their equivalent representation in world coordinates for use in dynamic analysis.



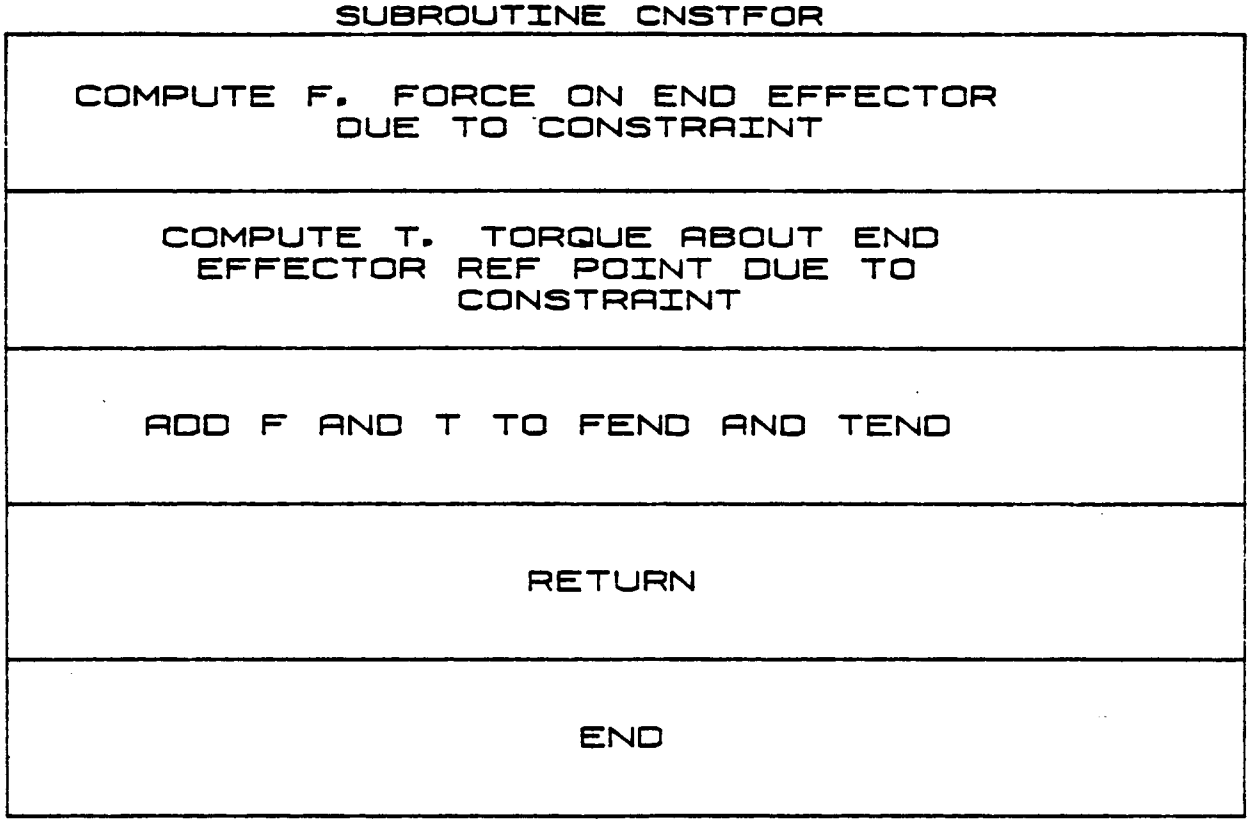
2.3.20 SPRGFOR

Subroutine SPRGFOR is called from FORCE when the manipulator end-effector is modeled as a compliant entity. This routine calculates the forces and torques at the end-effector reference point caused by its having compliance.



2.3.21 CNSTFOR

CNSTFOR is called from subroutine FORCE to compute the force on the end-effector and the torque about the end-effector reference point attributable to external constraints. These values are then added to the variables FEND and TEND.



2.3.22 PTACC

PTACC computes the acceleration of any point in any link of either arm. It uses the angular velocity and acceleration of the link to find the acceleration of the point relative to the acceleration of the link's origin and adds this to the acceleration of this link origin.

SUBROUTINE PTACC

OMEGA = LINK ANGULAR VELOCITY
VEC = VECTOR FROM LINK ORIGIN TO POINT
ALPHA = LINK ANGULAR ACCELERATION
WCV = OMEGA CROSS VEC
WCWCV = OMEGA CROSS WCV
ALCV = ALPHA CROSS VEC
RESULT = ALCV + WCWCV + ACCELERATION OF LINK ORIGIN

2.3.23 POSSENS

POSSENS is called from subroutine CONTROL when one of the feedback control laws is being used to drive a response simulation run. This routine obtains the discrete representation of the actual joint positions and also determines the actual joint velocities and accelerations.

SUBROUTINE POSSENS

DISCRETIZE JOINT POSITION DATA. STH

JOINT VELOCITIES. STHD = (STH-OLDSTH1) / (TIME-SIGLAST)

JOINT ACCEL.. STHDD = (STH-2.0*OLDSTH1+OLDSTH2) / (TIME-SIGLAST)**2

OLDSTH2 = OLDSTH1

OLDSTH1 = STH

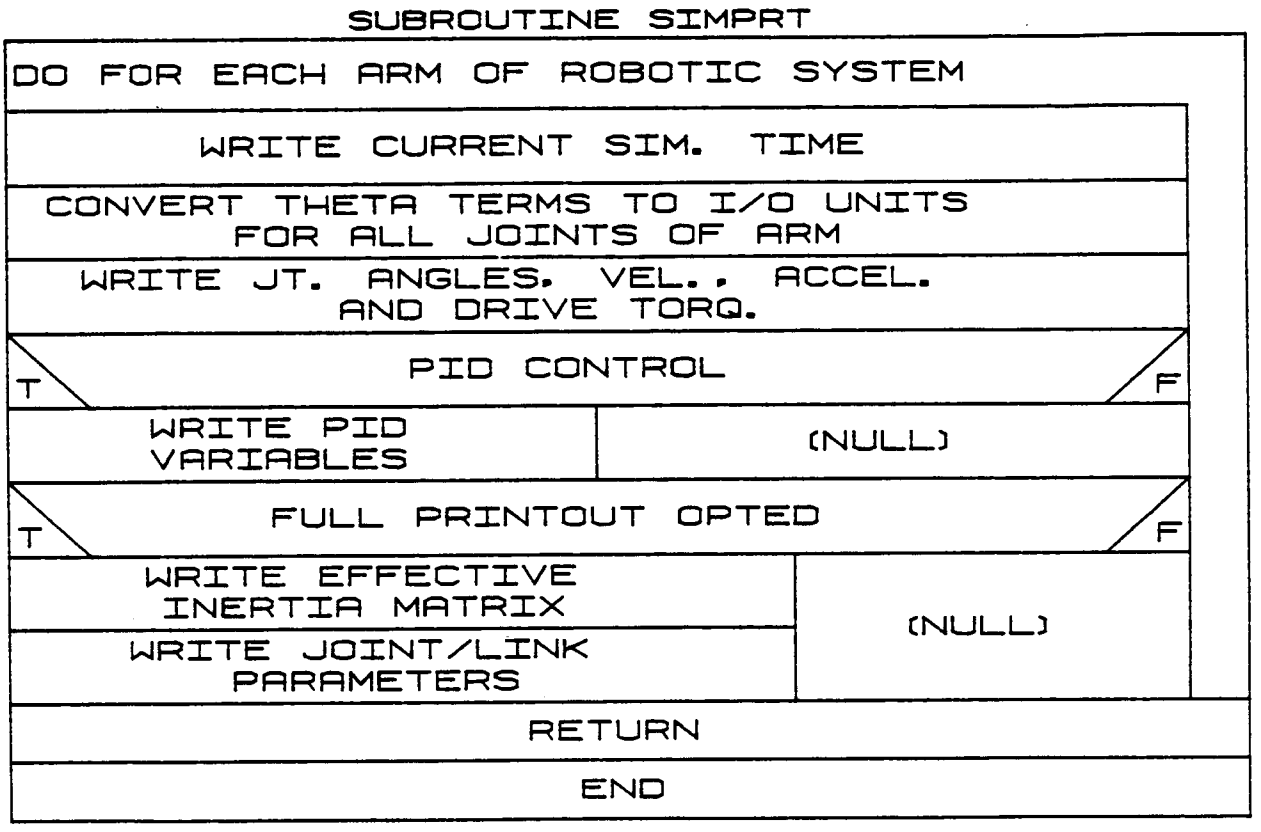
DO UNTIL NJ .EQ. NUMBER OF JOINTS IN CURRENT ARM

RETURN

END

2.3.24 SIMPRT

SIMPRT outputs the condensed or full data printout to file. It prints the position, velocity, and acceleration data for the arm at the time when called. If input flag IPRINT equals 2, a limited amount of information is printed (only TH, THD, THDD, and TDR).



ORIGINAL PAGE IS
OF POOR QUALITY

2.3.25 SIMPLT

SIMPLT allows the user to write a plot file for output. The user is asked to choose from among several different plot package options. The chosen package determines which response simulation parameters are written to the plot file.

Option 1, the BRIEF PLOT PACKAGE, writes joint angular displacements, joint angular velocities, joint angular accelerations and drive torques.

Option 2, the END-EFFECTOR PLOT PACKAGE, writes end-effector translational position, force vector at the end-effector and torque vector at the end-effector.

Option 3, the JOINT POSITIONS PLOT PACKAGE, writes translational joint positions.

Option 4, the REACTION FORCES PLOT PACKAGE, writes force joint vectors and torque joint vectors.

Option 5, the COMBINATION PLOT PACKAGE, writes all of the above--joint angular displacements, translational joint positions, joint angular velocities, joint angular accelerations, force joint vectors, torque joint vectors, drive torques, end-effector translational position, force vector at the end-effector and torque vector at the end-effector.

Option 6, the PID CONTROL PLOT PACKAGE, writes amplifier voltages, joint reference positions, joint position errors, end-effector reference position and end-effector position error.

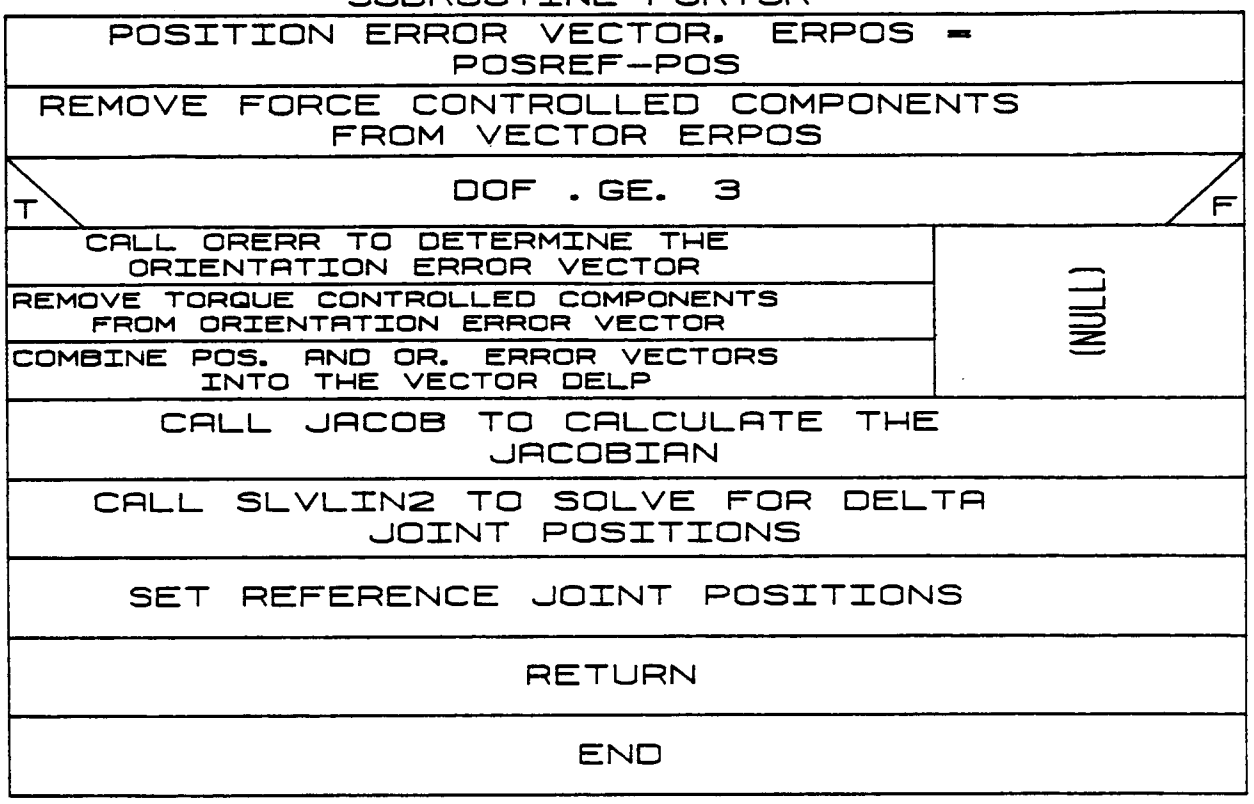
Option 7, the FORCE/TORQUE PLOT PACKAGE, writes amplifier voltages, reference position, reference force, end-effector translational position, force vector at the end-effector, torque vector at the end-effector, error in position and error in force/torque.

SUBROUTINE SIMPLT

SET PLOT HEADER DATA RECORDS FOR EACH PLOT TYPE	
T: SIM. AT START TIME / F: WRITE PLOT FILE TYPE DESCRIPTIONS TO TERMINAL PROMPT FOR PLOT FILE TYPE TO WRITE	(NULL)
T: BRIEF PACKAGE CHOSEN / F: WRITE HEADER RECORD IF START TIME WRITE TIME, ANGLE, VEL., ACC., AND DRIVE TORQ.	(NULL)
T: END-EFFECTOR PACKAGE CHOSEN / F: WRITE HEADER RECORD IF START TIME WRITE TIME, POS., FORC., AND TORQ. AT END-EFFECTOR	(NULL)
T: JOINT POSITION PACKAGE CHOSEN / F: WRITE HEADER RECORD IF START TIME WRITE TIME, AND JOINT POS. VECTORS	(NULL)
T: REACTION FORCES PACKAGE CHOSEN / F: WRITE HEADER RECORD IF START TIME WRITE TIME, FORC. AND TORQUES AT JOINTS	(NULL)
T: COMBINATION PACKAGE CHOSEN / F: WRITE HEADER RECORD IF START TIME WRITE TIME, AND ALL OF THE ABOVE PARAMETERS	(NULL)
T: PID CONTROL PACKAGE CHOSEN / F: WRITE HEADER RECORD IF START TIME WRITE TIME, JOINT VOLTS., REF. ANGLE AND JT. POS. ERRORS WRITE TIME, END-EFFECTOR REF. POS., POS. AND ROT. ERR.	(NULL)
T: FORCE/TORQUE CONTROL PACKAGE CHOSEN / F: WRITE HEADER RECORD IF START TIME WRITE TIME, AND JOINT VOLTS. WRITE TIME, REF. POS., REF. FOR./ROT. VECT., END-EFF. POS./FOR., AND ERRS.	(NULL)
RETURN	
END	

2.3.26 FORTOR

FORTOR is called from subroutine CNTRSIG if manual force/torque control is used to drive a response simulation. This routine calculates the joint position error vectors caused by the error in the position-controlled components of end-effector motion.



2.3.27 FORREF

FORREF is called from subroutine CNTRSIG when manual force/torque control is used to drive a response simulation. Individual joint torque error vectors are calculated from the end-effector force error and torque error vectors.

SUBROUTINE FORREF

CALCULATE END EFFECTOR FORCE ERROR COMPONENTS
CALCULATE END EFFECTOR TORQUE ERROR COMPONENTS
STORE ERROR COMPONENTS IN VECTOR DELFT
CALCULATE REFERENCE FORCE/TORQUE VECTOR
CALL JACOB TO DETERMINE THE JACOBIAN. RJACOB
CALCULATE JOINT TORQUES. TORJNT = TORJNT-DELFT*RJACOB
RETURN
END

2.3.28 CMPCTRL

CMPCTRL is called from CNTRSIG when active compliance control is used in a response simulation run. This subroutine first calculates end position deltas (ref-actual), joint control torques, and joint torque deltas (control-sensed). The thetas are put through a derivative control block to get joint torques. The joint torque deltas are put through a lead-lag filter in parallel with an integrating control block. The joint control torques are summed with the other processed signals to get a total joint torque. This is then converted to motor amplifier input voltages.

SUBROUTINE CMPCTRL

SET TVCVT. JOINT TORQUE TO VOLTS CONVERSION FACTOR
CALL JACOB TO CALCULATE THE JACOBIAN. RJACOB
DETERMINE RJTRANS. THE TRANSPOSE OF THE JACOBIAN
CALL MATMPY TO FIND TOR. THE INPUT TORQUES
CALL MATMPY TO FIND TBIAS. THE BIAS TORQUES
JOINT CONTROL TORQUES. TCTRL = TOR + TBIAS
DETERMINE TSENS. SENSED FORCES AND TORQUES
TORQUE DELTAS. DELTOR = TCTRL + TSENS
CALCULATE RJTORQ. JOINT ACTUATOR DRIVE TORQUES
CONVERT JOINT TORQUES TO INPUT VOLTAGES
RETURN
END

2.3.29 PIDCON

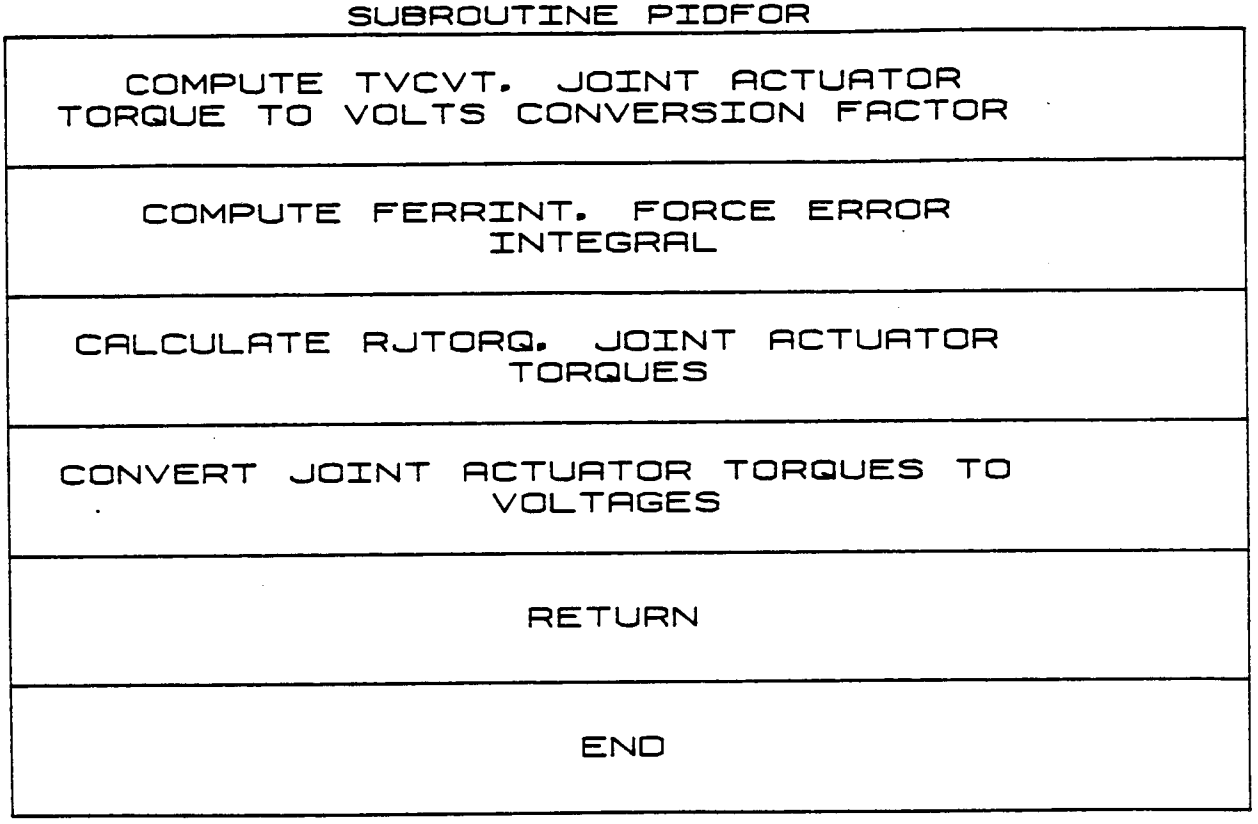
Subroutine PIDCON is called from CONTROL when a control law is used to drive a response simulation run. The routine takes the vector of joint position errors and, simulating a PID control loop, calculates joint actuator voltages.

SUBROUTINE PIDCON

SET JOINT ACTUATOR TORQUE TO VOLTS CONVERSION FACTOR
CALCULATE JOINT POSITION ERROR. DELTH = THREF − STH
CALCULATE ERRINT, THE ERROR INTEGRAL
CALCULATE RJTORQ, JOINT ACTUATOR TORQUES
CONVERT ACTUATOR TORQUES TO VOLTAGES
RETURN
END

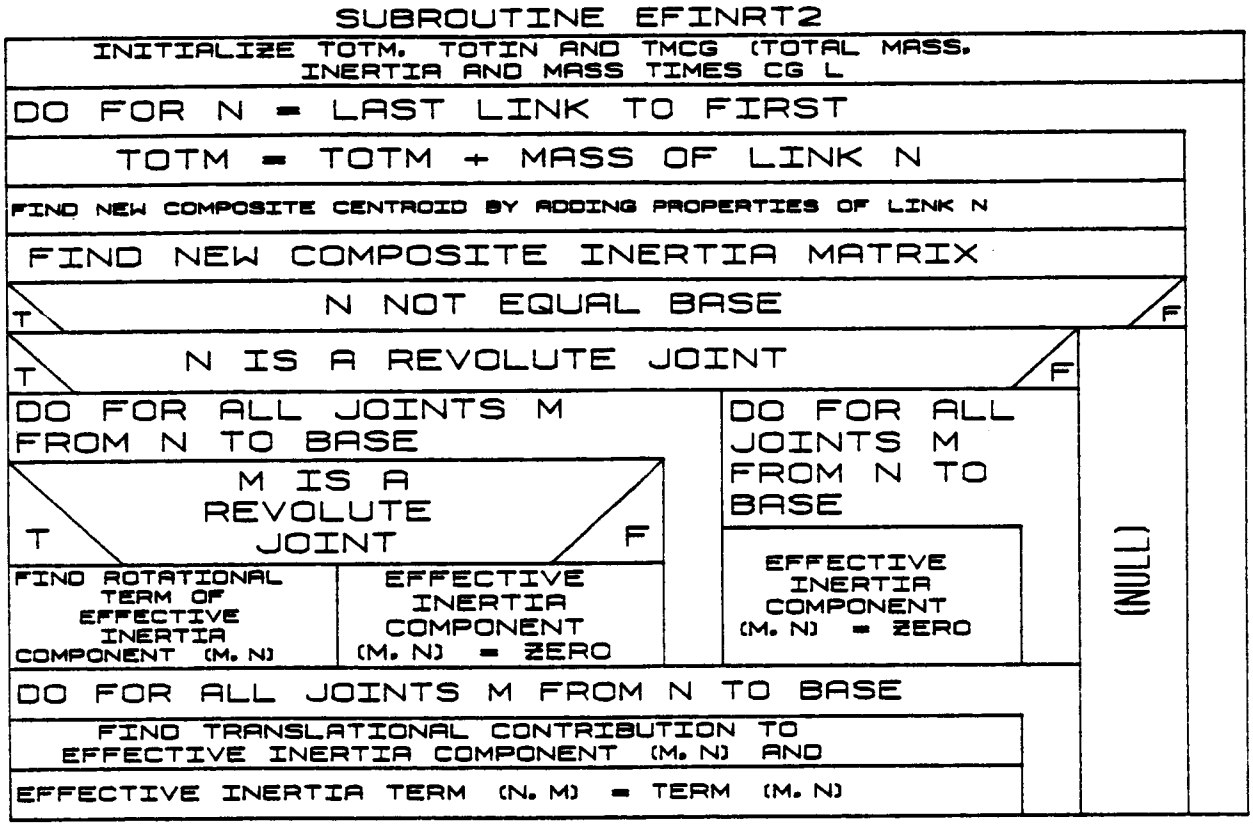
2.3.30 PIDFOR

Subroutine PIDFOR is called from CONTROL when force/torque control is being used to drive a response simulation run. This routine calculates the joint actuator voltages caused by the force-controlled components of manipulator motion.



2.3.31 EFINRT2

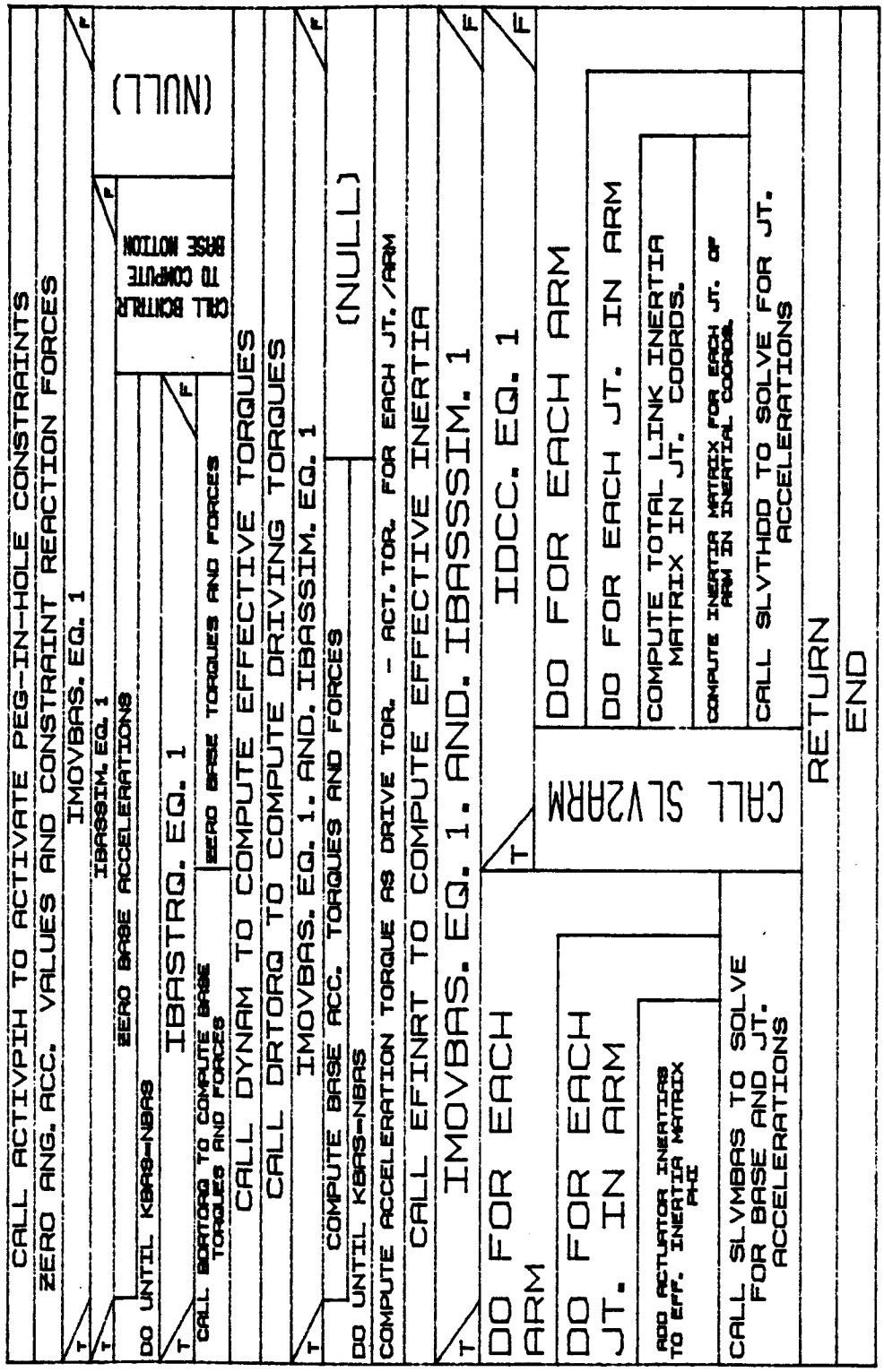
EFINRT2 computes the effective inertia matrix (in joint coordinates) for a manipulator. The effective inertia matrix is an NxN matrix that gives the joint torques attributable to joint accelerations. The (m,n) term corresponds to joints m and n and depends on the mass of the arm from link n to the end-effector so the program evaluates composite masses, centroids and inertia distributions for these "composite masses." Each term of the effective inertia matrix is then evaluated as a combination of dot products and cross-products among the joint axis directions and locations and the mass parameters of the composite links (see Study Results volume).



2.3.32 NLINK

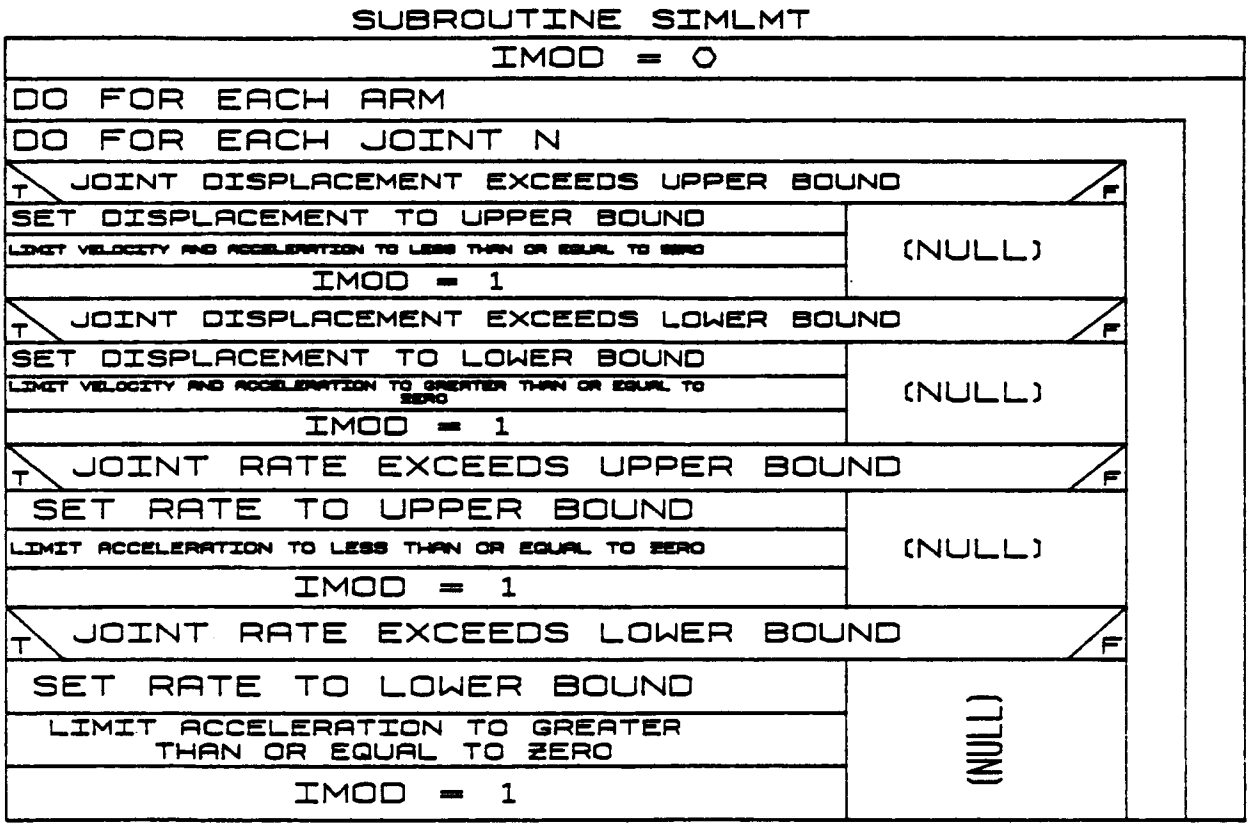
Subroutine NLINK is called from DERIV during response simulation to compute the base accelerations BASACC, BASOMD and joint accelerations THDD. It sets the joint accelerations to zero, the base accelerations to zero if moving base is simulated. The requirement analysis is used to compute effective joint torques. If moving base is simulated, BDRTORQ is called to input base driving torques and the base acceleration torques are calculated and DRTORQ is called to find the joint driving torques. EFINRT is called to compute the effective inertias. If moving base is simulated, SLVMBAS is called to solve for base and joint accelerations. If dual arm control, SLV2ARM is called to compute joint accelerations. Otherwise SLVTHDD is called to solve for joint accelerations.

ORIGINAL PAGE IS
OF POOR QUALITY



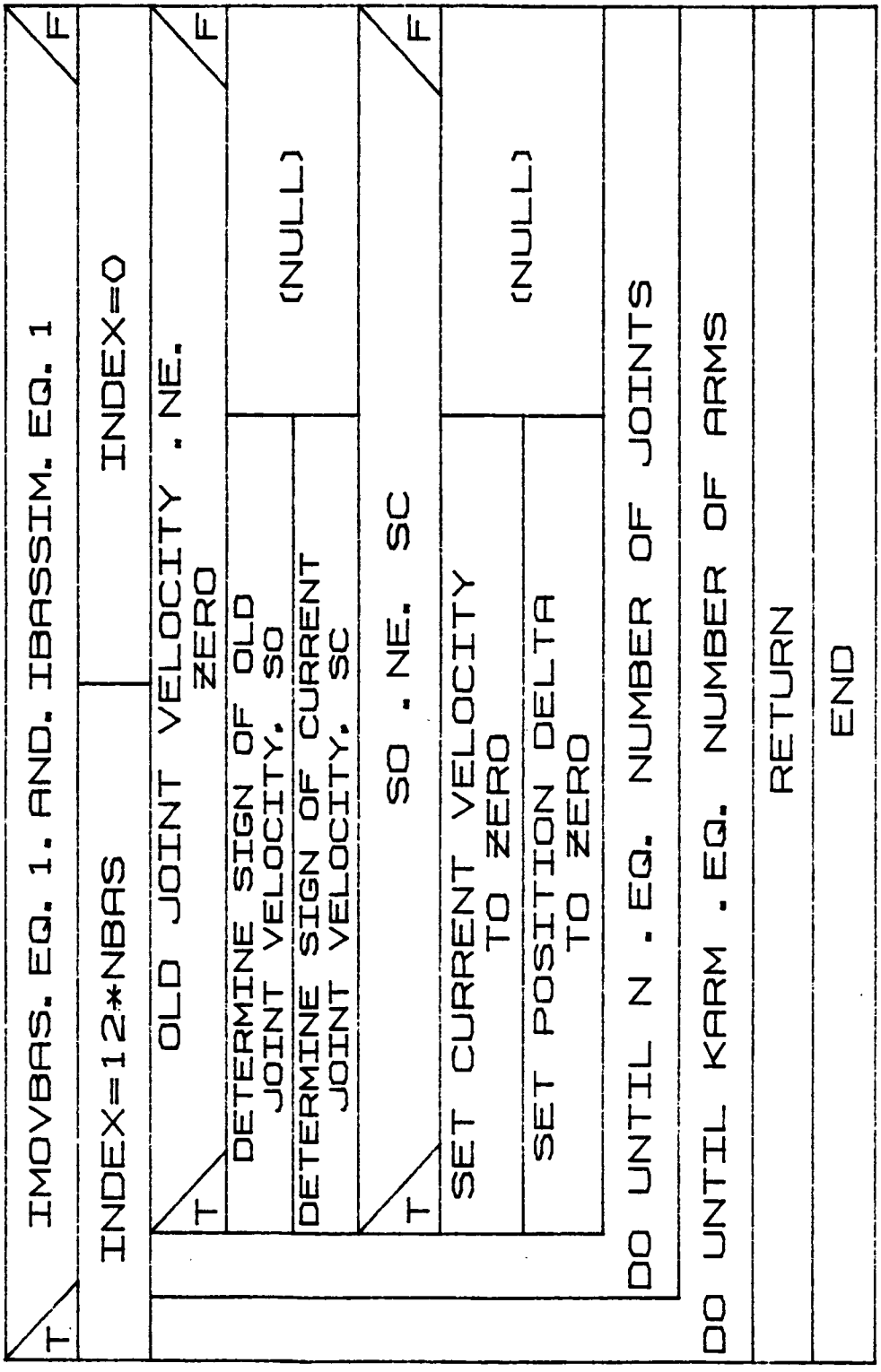
2.3.33 SIMLMT

SIMLMT is called by DERIV and first checks the joint displacements against their limits. If any limits are exceeded, the joint position is set to that limit and the joint rate and acceleration are limited to zero. Similarly, the rate limits are checked and if any are exceeded, the corresponding rate is set to that limit and the acceleration is bounded by zero. IMOD is set if any positions or rates are modified.



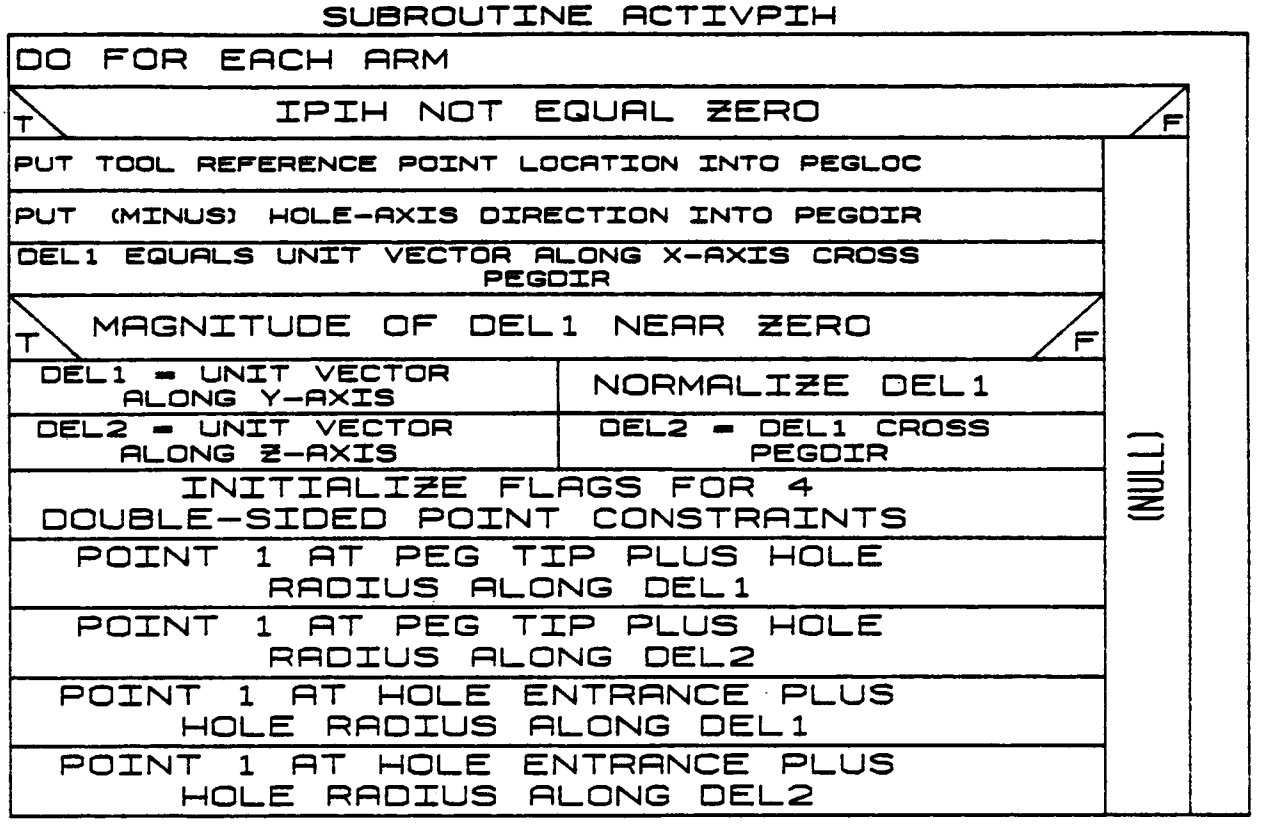
2.3.34 STOPFR

Subroutine STOPFR is called from INTGRT to simulate static friction in the joints during a response simulation run. If the joint velocity at the previous time step is not equal to zero and the sign is the opposite of the sign of the current time step, the current velocity and position delta are set to zero. If moving base is simulated, assume no friction at the base joints.


SUBROUTINE STOPFR
IMOVBAS. EQ. 1. AND. IBASSIM. EQ. 1
T
F
INDEX=12*NBAS
INDEX=0
OLD JOINT VELOCITY . NE. ZERO
T
F
DETERMINE SIGN OF OLD JOINT VELOCITY, SO
(NULL)
DETERMINE SIGN OF CURRENT JOINT VELOCITY, SC
SO . NE. SC
T
F
SET CURRENT VELOCITY TO ZERO
(NULL)
SET POSITION DELTA TO ZERO
DO UNTIL N . EQ. NUMBER OF JOINTS
DO UNTIL KARM . EQ. NUMBER OF ARMS
RETURN
END

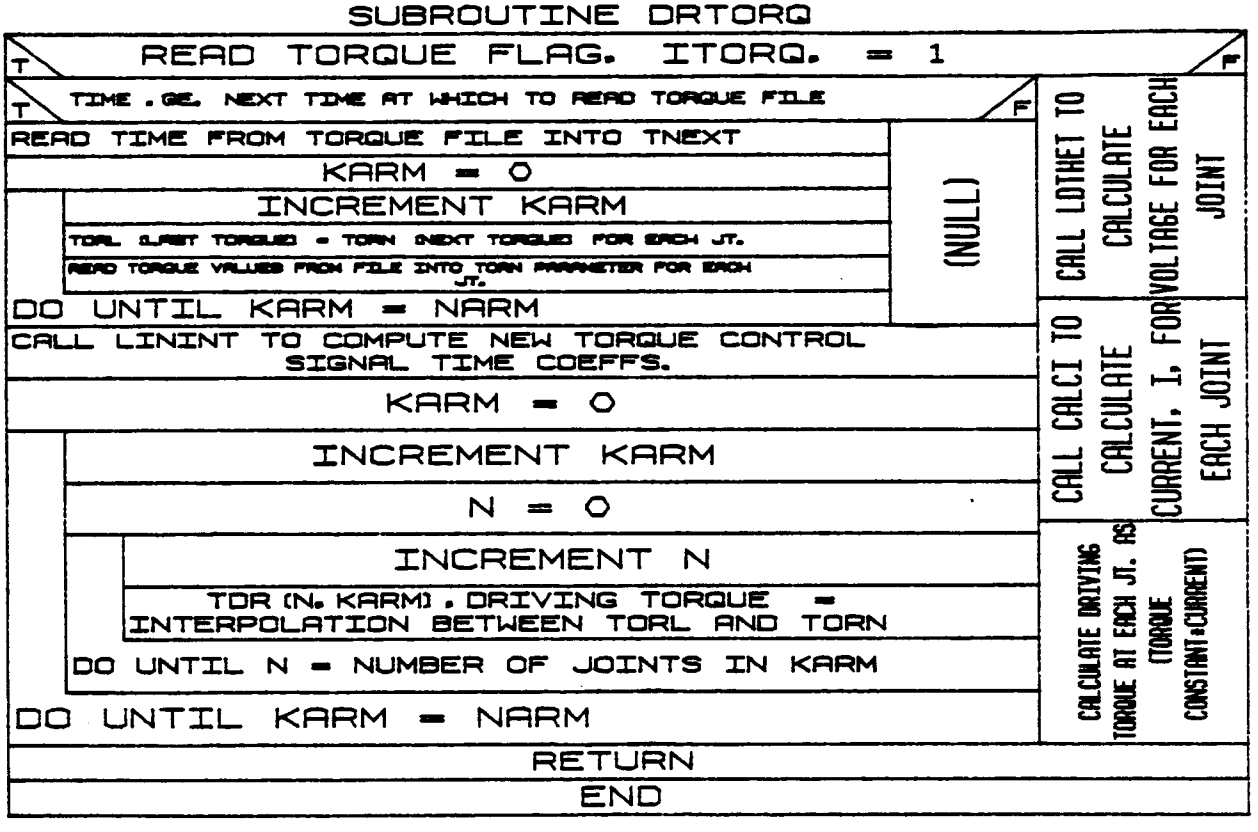
2.3.35 ACTIVPIH

ACTIVPIH sets up the flags and variables activating a peg-in-hole constraint if such a constraint is included. It sets up four point constraints--two each (in orthogonal directions) at the top of the hole and at the tip of the peg.



2.3.36 DRTORQ

The DRTORQ routine calculates the torque output from each joint motor by using a control algorithm strategy or reading them from a file. The calculations are based on the torque constant for each joint and the armature current.

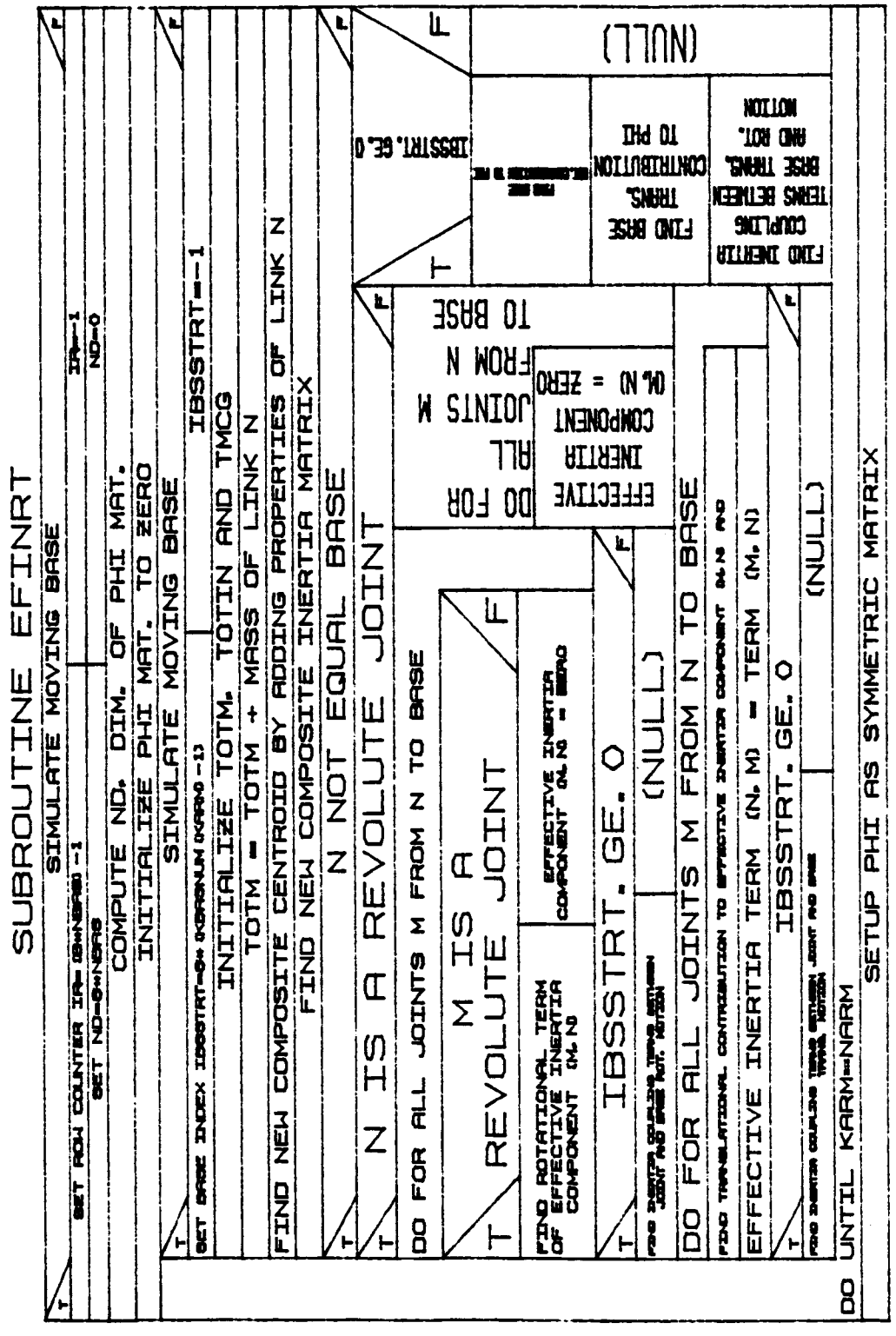


2.3.37 EFINRT

EFINRT computes the effective inertia matrix for a system containing one or more arms on fixed or moving bases. The effective inertia matrix PHI gives the base torques, forces and joint torques attributable to base and joints accelerations. The dimension of PHI is (ND, ND)

where $ND = (6 * NBAS) + \sum_{I=1}^{NARM} NJ(I)$

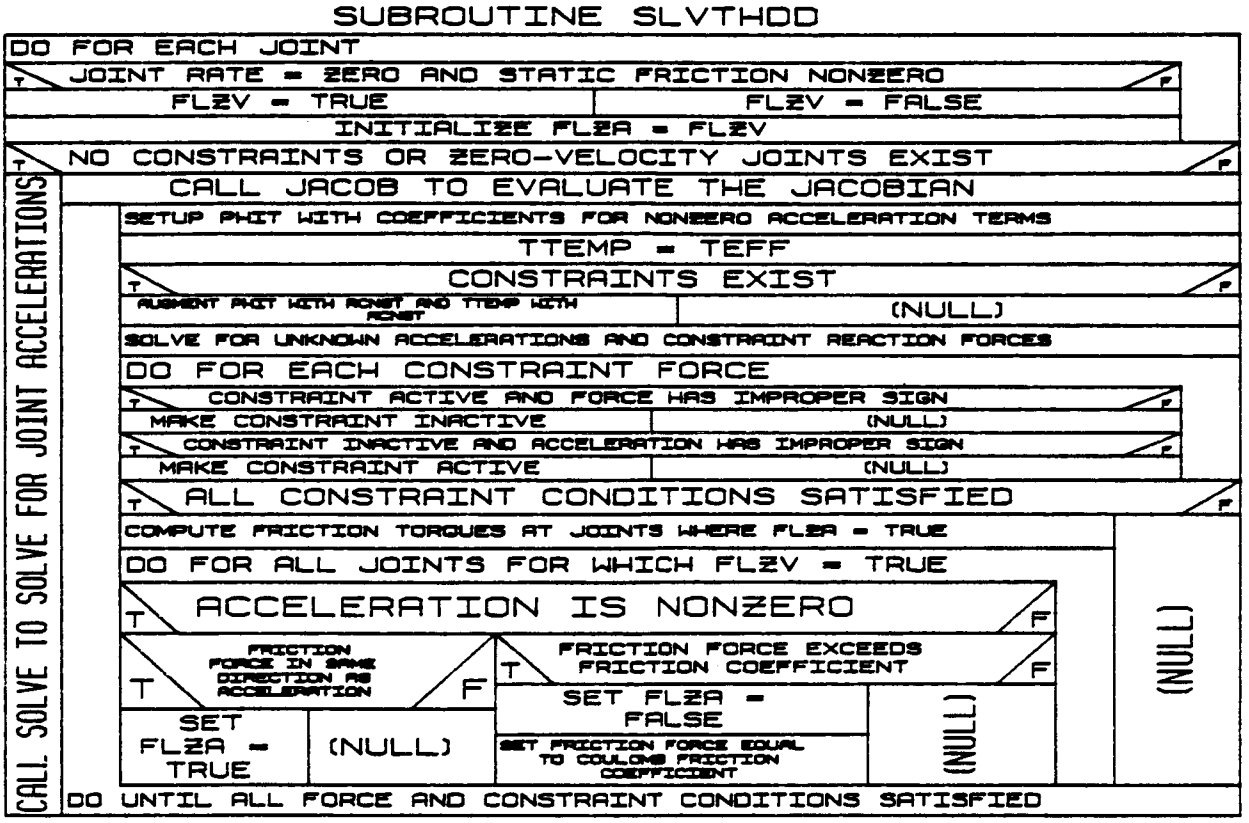
ORIGINAL PAGE
OF POOR QUALITY



2.3.38 SLVTHDD

ORIGINAL PAGE IS
OF POOR QUALITY

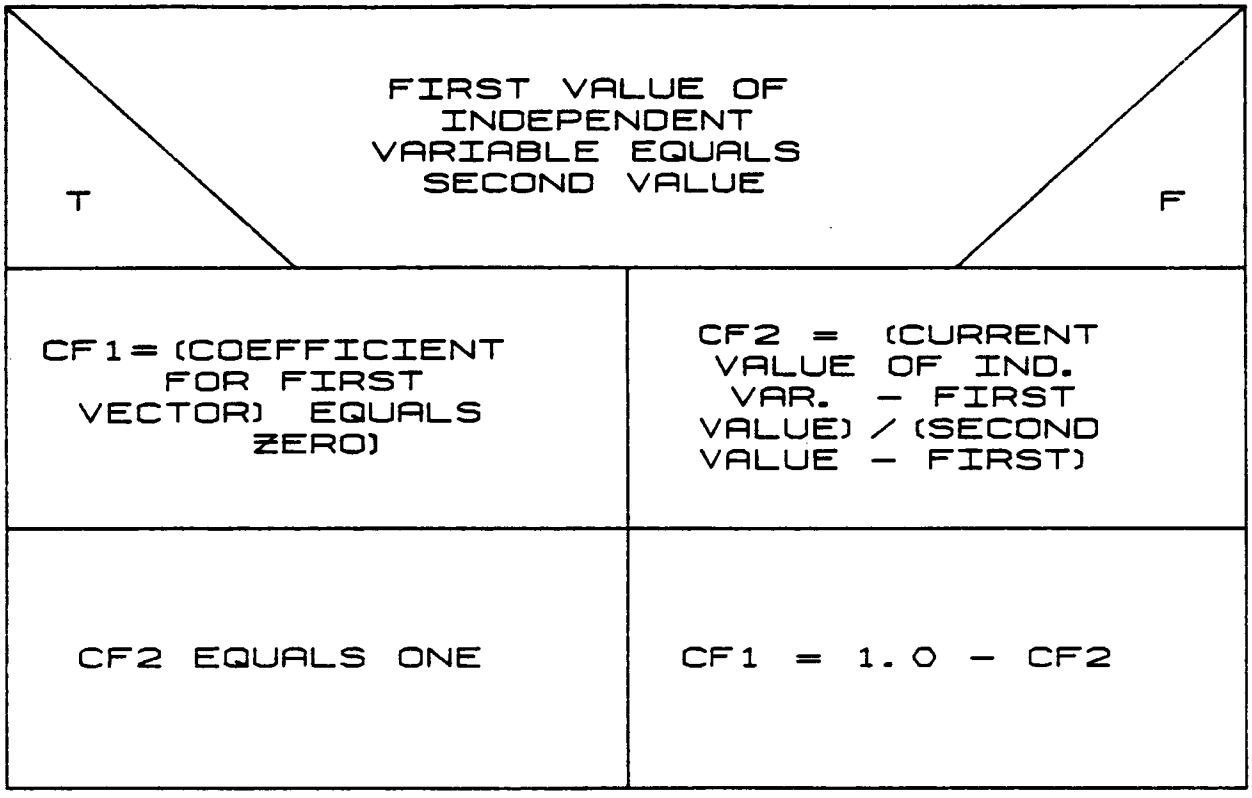
SLVTHDD solves for unknown joint accelerations and constraint reaction forces for a given arm state and joint driving forces. All zero-velocity joints are assumed to have zero acceleration. If the friction forces needed to produce zero acceleration are greater than the static friction force, the acceleration is assumed finite and the equations are re-solved. Similarly, the constraints are assumed active and if the resulting constraint force is in the wrong direction the constraint becomes inactive and the equations are re-solved. This process is repeated until all conditions on the friction forces and constraints are satisfied.



2.3.39 LININT

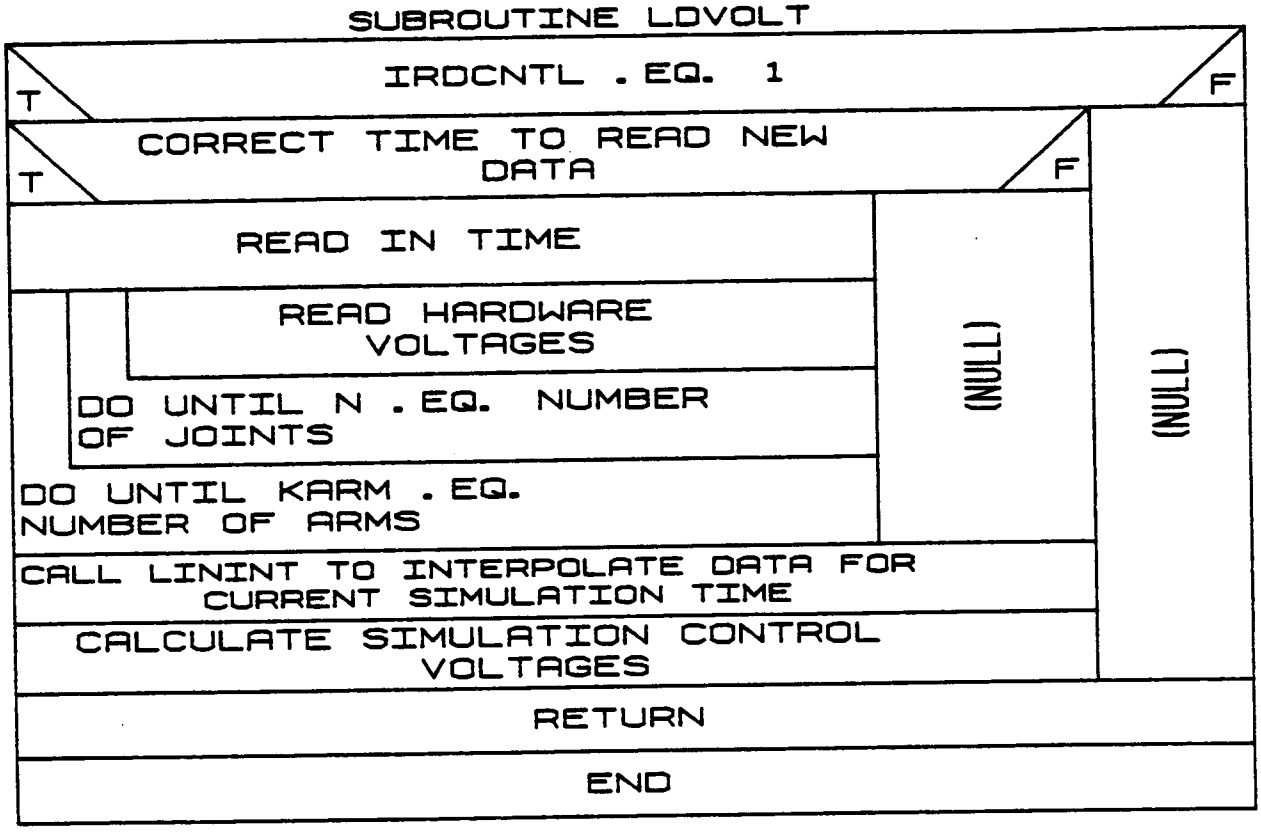
Subroutine LININT is called to set up the coefficient for performing linear interpolation between two vectors.

SUBROUTINE LININT



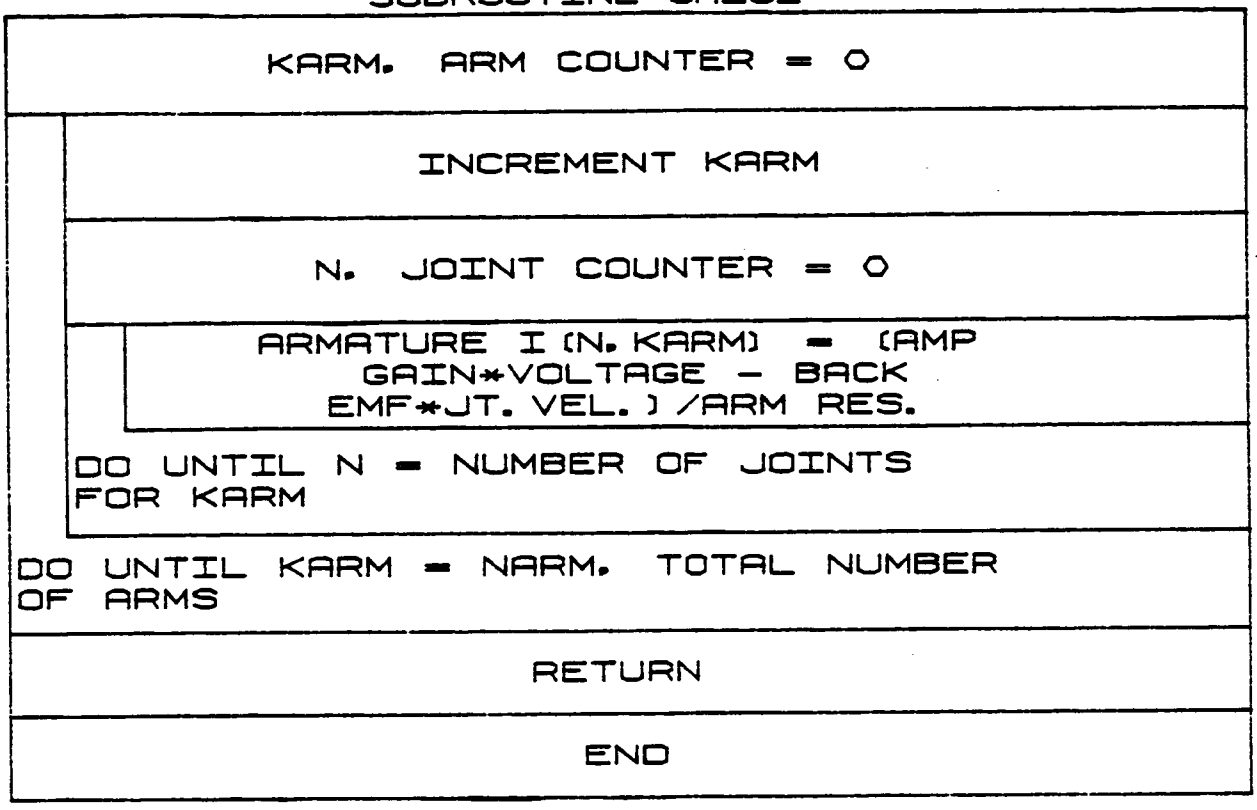
2.3.40 LDVOLT

Subroutine LDVOLT is called from DRTORQ when a file of actuator voltages is to be read in and used to drive a response simulation run. At the correct time the routine reads time and voltage from an existing file. LININT is called to interpolate the best voltage for the current simulation time. The control voltage is then calculated from this.



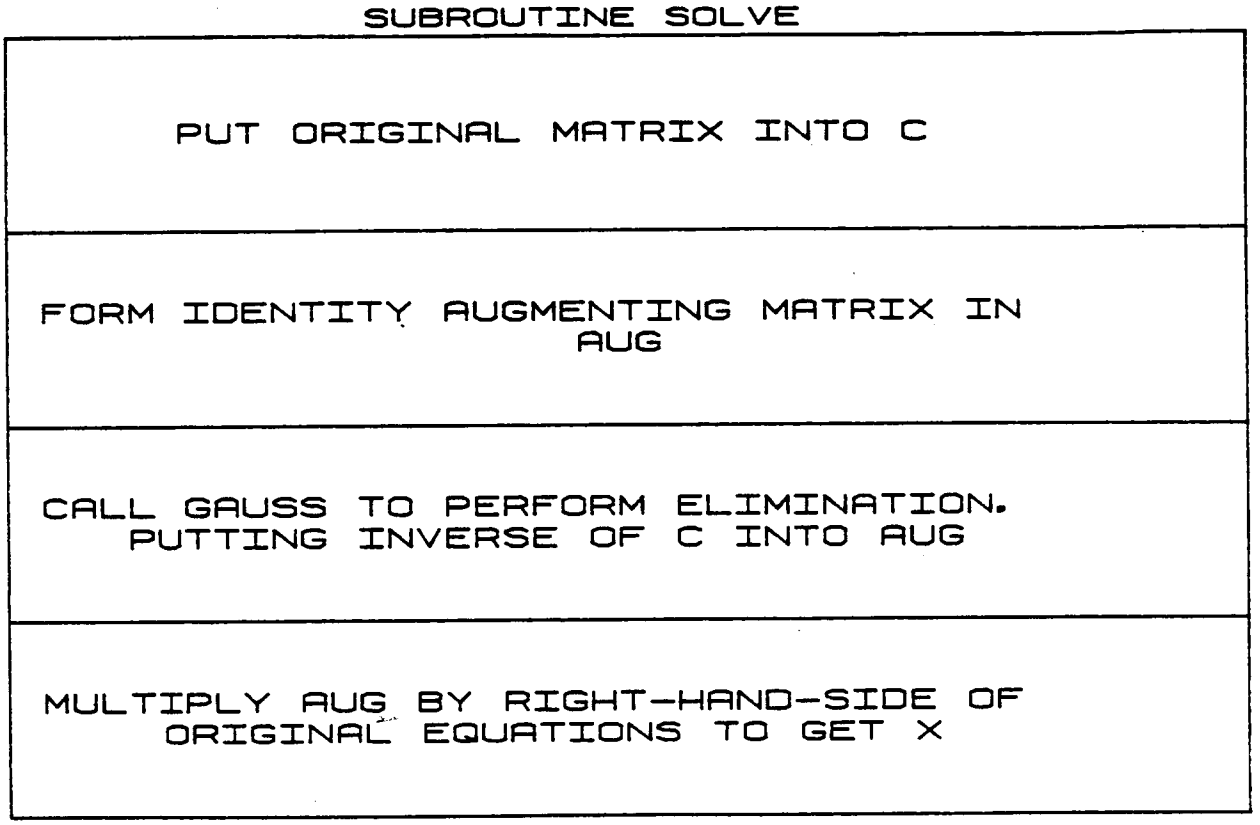
2.3.41 CALCI

The CALCI subroutine calculates the amplifier current values for each of the joints in the system given the motor parameter values and the state velocity.



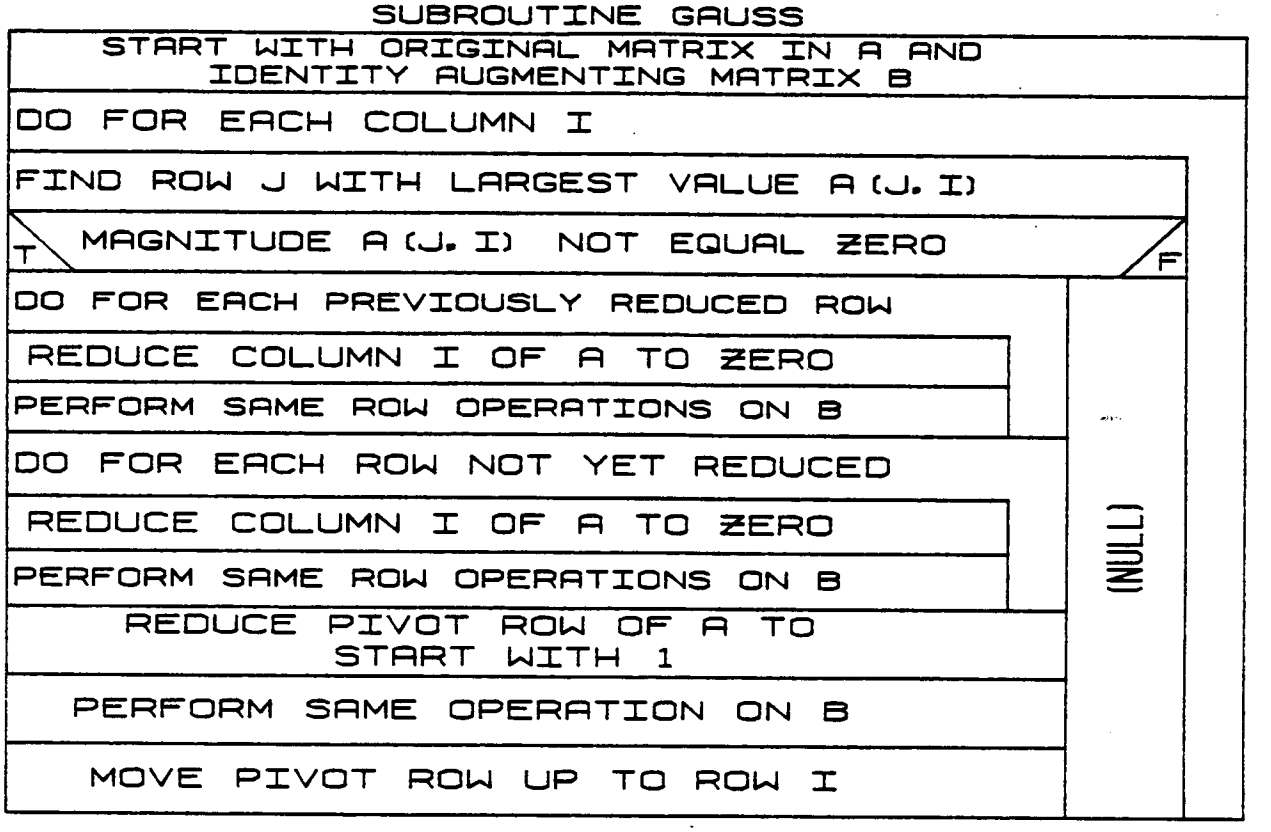
2.3.42 SOLVE

SOLVE is used to solve a set of N linear equations in N unknowns. It sets up an identity-augmenting matrix, calls GAUSS to invert the original matrix and then multiplies this inverse times the right-hand side of the equations to obtain the resulting solution.



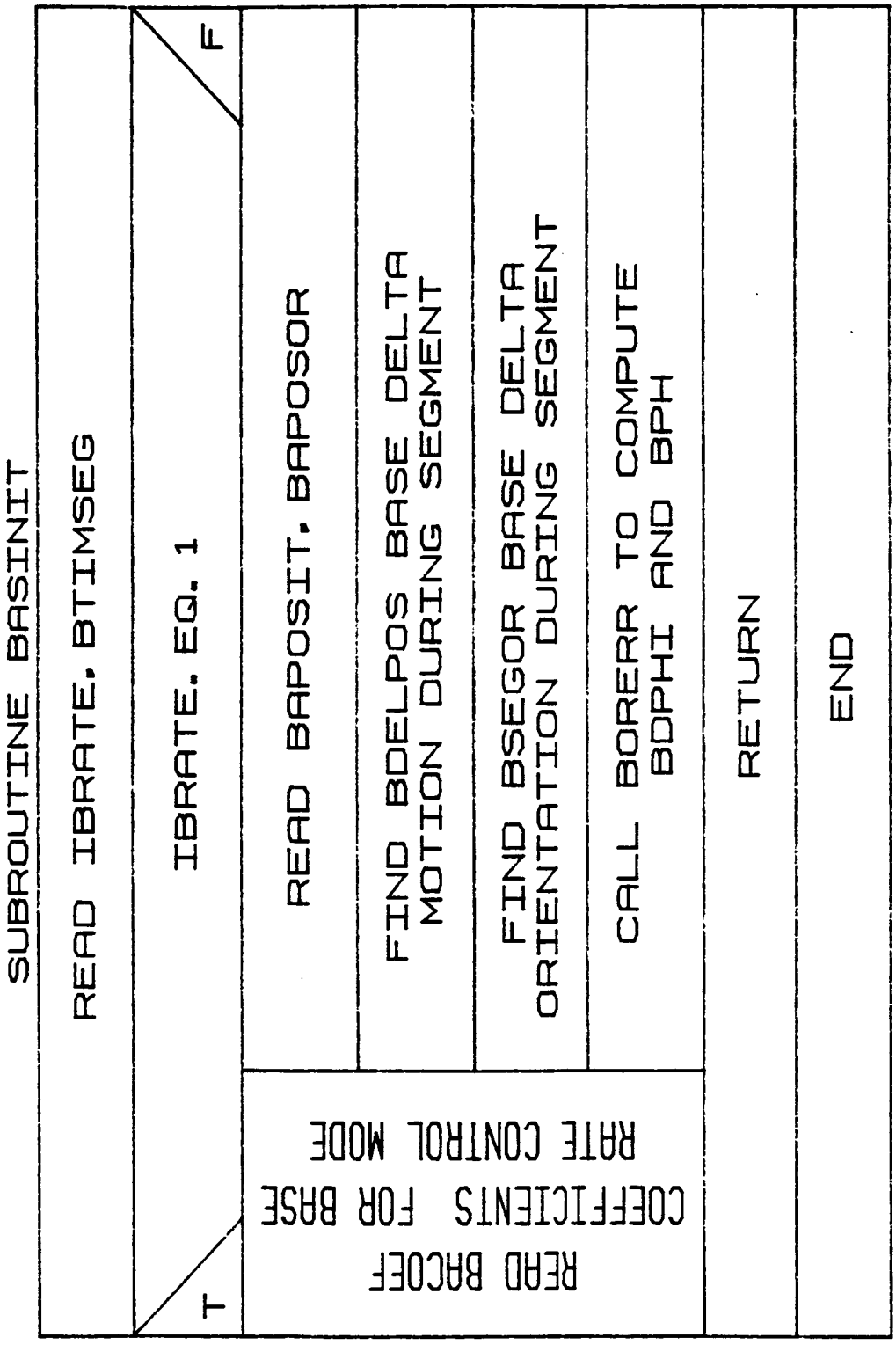
2.3.43 GAUSS

GAUSS performs Gauss-Jordan elimination with partial pivoting on an augmented matrix system to reduce the system to row-echelon form during the matrix inversion process. The largest value remaining in a column is used as the pivot value for that column during reduction.



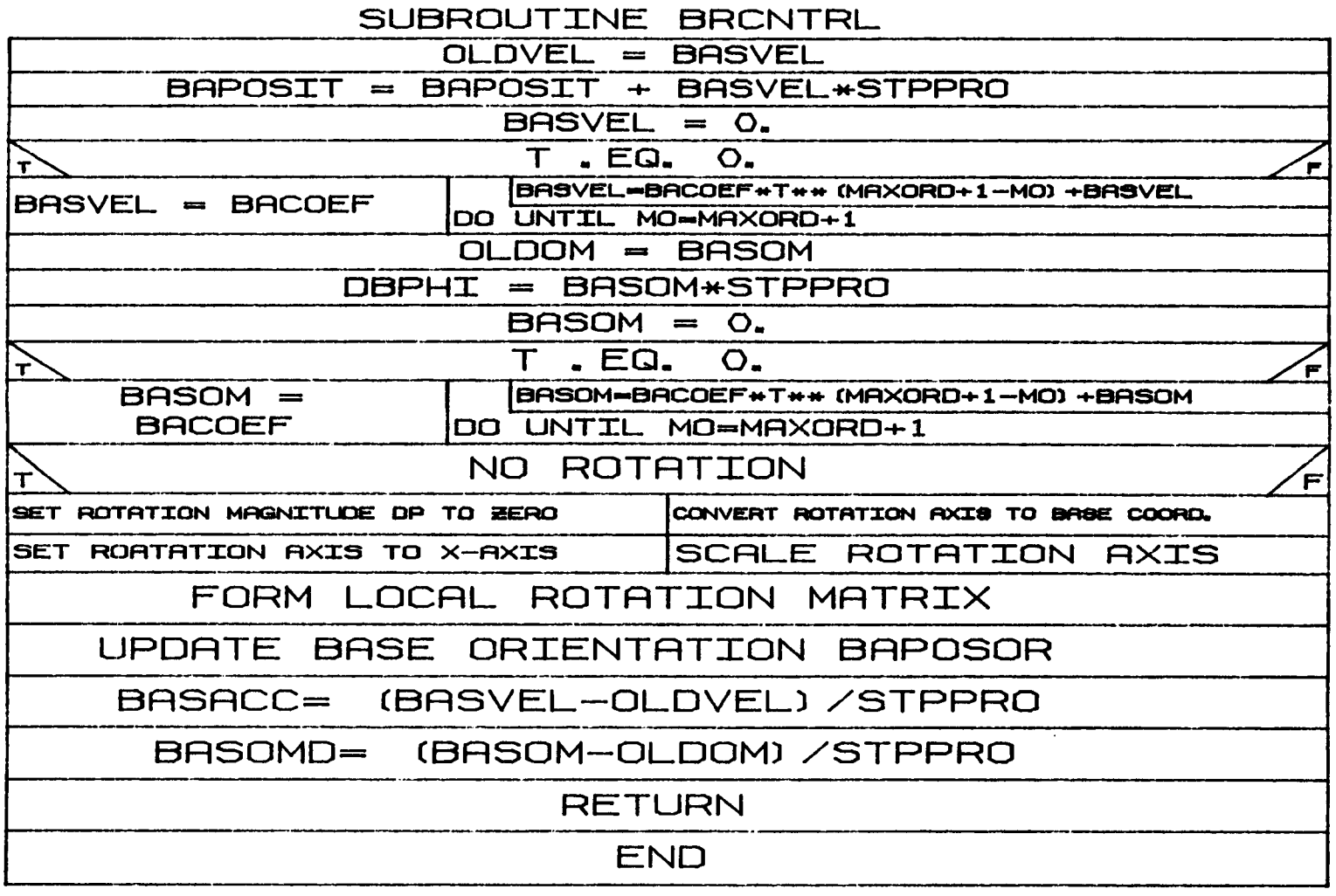
2.3.44 BASINIT

Subroutine BASINIT is called from BCNTRLR at the beginning of each new base motion people segment. The coefficients of the polynomials defining the base rates or the desired positions and orientations are read from the base motion profile.


SUBROUTINE BASINIT
READ IBRATE, BTIMSEG
IBRATE. EQ. 1
F
T
READ BAPOSIT, BAPOSOR
FIND BDELPOS BASE DELTA MOTION DURING SEGMENT
FIND BSEGOR BASE DELTA ORIENTATION DURING SEGMENT
CALL BORERR TO COMPUTE BDPHI AND BPH
READ BRCOEF COEFFICIENTS FOR BASE RATE CONTROL MODE
RETURN
END

2.3.45 BRCNTRL

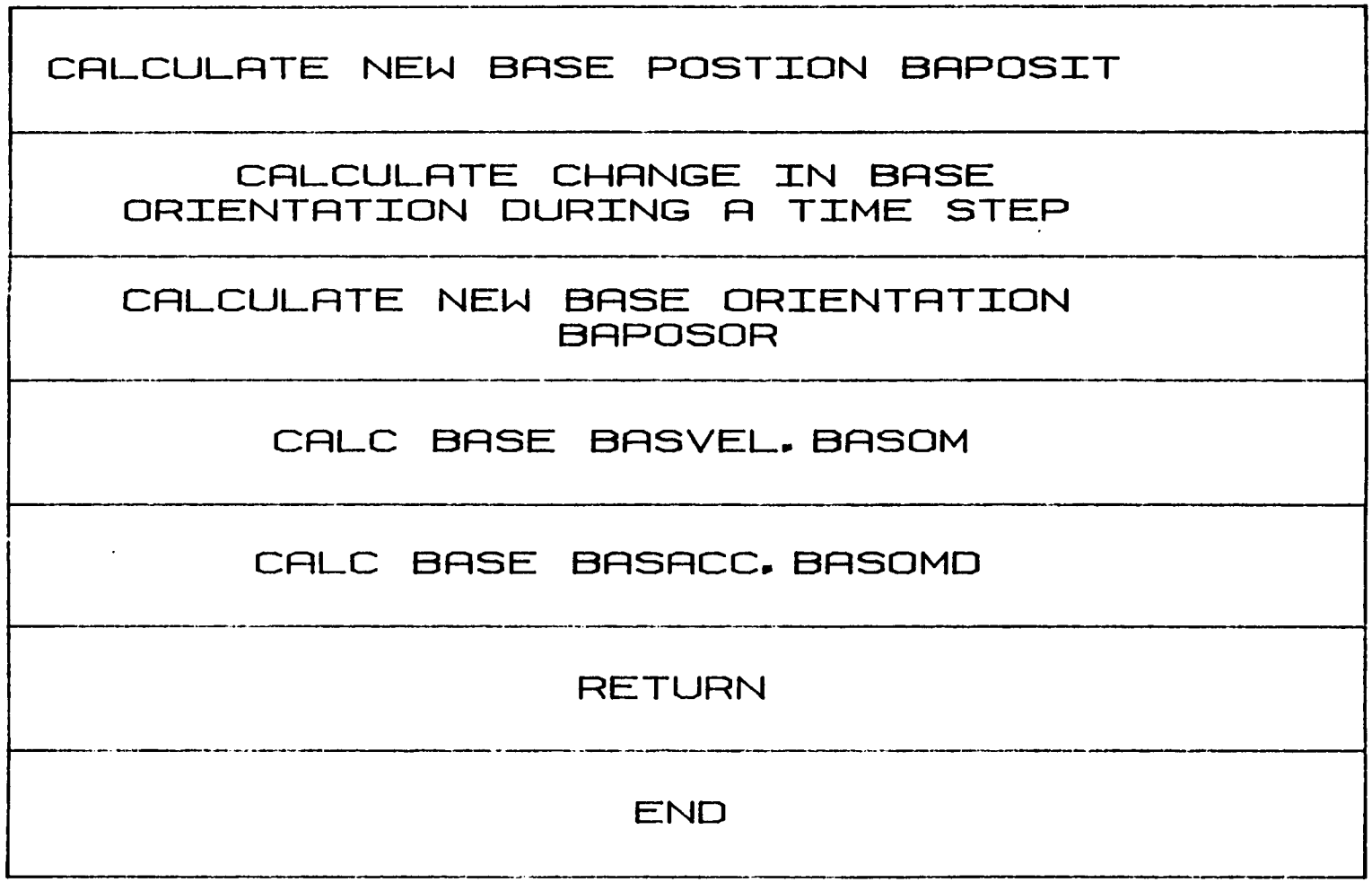
Subroutine BRCNTRL is called from BCNTRLR when rate control of the base is specified. The rate polynomials are used to calculate the base translational and angular velocities. The base positions, orientations and accelerations are calculated from the velocities.


SUBROUTINE BRCNTRL
OLDVEL = BASVEL
BAPOSIT = BAPOSIT + BASVEL*STPPRO
BASVEL = 0.
T . EQ. 0.
T
F
BASVEL = BACOEF
BASVEL=BACOEF*T** (MAXORD+1-MO) +BASVEL
DO UNTIL MO=MAXORD+1
OLDOM = BASOM
DBPHI = BASOM*STPPRO
BASOM = 0.
T . EQ. 0.
T
F
BASOM = BACOEF
BASOM=BACOEF*T** (MAXORD+1-MO) +BASOM
DO UNTIL MO=MAXORD+1
NO ROTATION
T
F
SET ROTATION MAGNITUDE DP TO ZERO
CONVERT ROTATION AXIS TO BASE COORD.
SET ROATATION AXIS TO X-AXIS
SCALE ROTATION AXIS
FORM LOCAL ROTATION MATRIX
UPDATE BASE ORIENTATION BAPOSOR
BASACC= (BASVEL-OLDVEL) /STPPRO
BASOMD= (BASOM-OLDOM) /STPPRO
RETURN
END

2.3.46 BPCNTRL

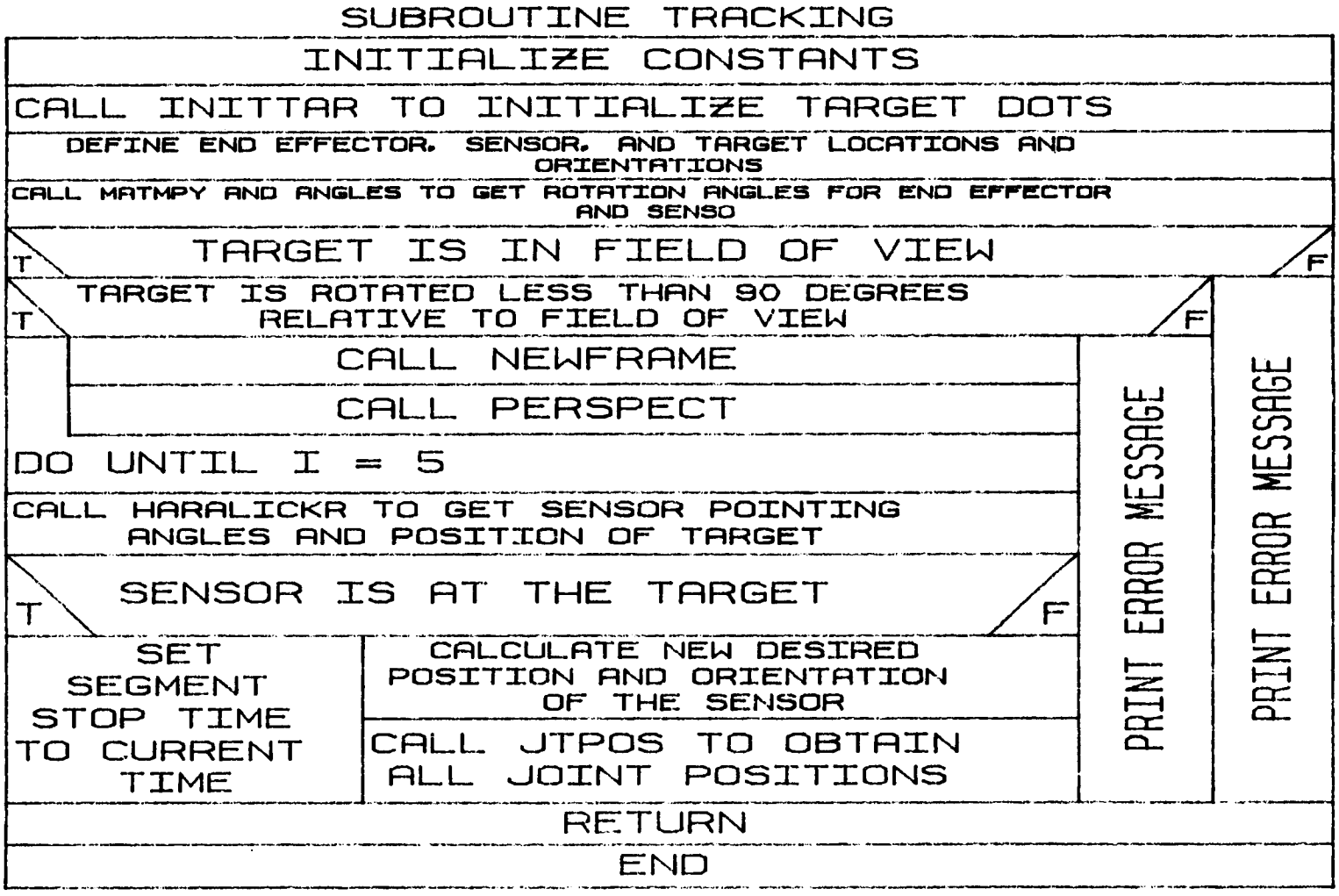
Subroutine BPCNTRL is called from BCNTRLR when position control of the base is to be used. The rate profile is used to calculate the base positions, velocities and accelerations.

SUBROUTINE BPCNTRL



2.3.47 TRACKING

Subroutine TRACKING is called from CNTRLR when sensor control of end-effector motion is chosen by the user. This simulates tracking of a target by a video device mounted on a manipulator end-effector.


SUBROUTINE TRACKING
INITIALIZE CONSTANTS
CALL INITTAR TO INITIALIZE TARGET DOTS
DEFINE END EFFECTOR, SENSOR, AND TARGET LOCATIONS AND ORIENTATIONS
CALL MATMPY AND ANGLES TO GET ROTATION ANGLES FOR END EFFECTOR AND SENSO
T
TARGET IS IN FIELD OF VIEW
F
T
TARGET IS ROTATED LESS THAN 90 DEGREES RELATIVE TO FIELD OF VIEW
F
CALL NEWFRAME
CALL PERSPECT
DO UNTIL I = 5
CALL HARALICKR TO GET SENSOR POINTING ANGLES AND POSITION OF TARGET
T
SENSOR IS AT THE TARGET
F
SET SEGMENT STOP TIME TO CURRENT TIME
CALCULATE NEW DESIRED POSITION AND ORIENTATION OF THE SENSOR
CALL JTPOS TO OBTAIN ALL JOINT POSITIONS
PRINT ERROR MESSAGE
PRINT ERROR MESSAGE
RETURN
END

2.3.48 INITTAR

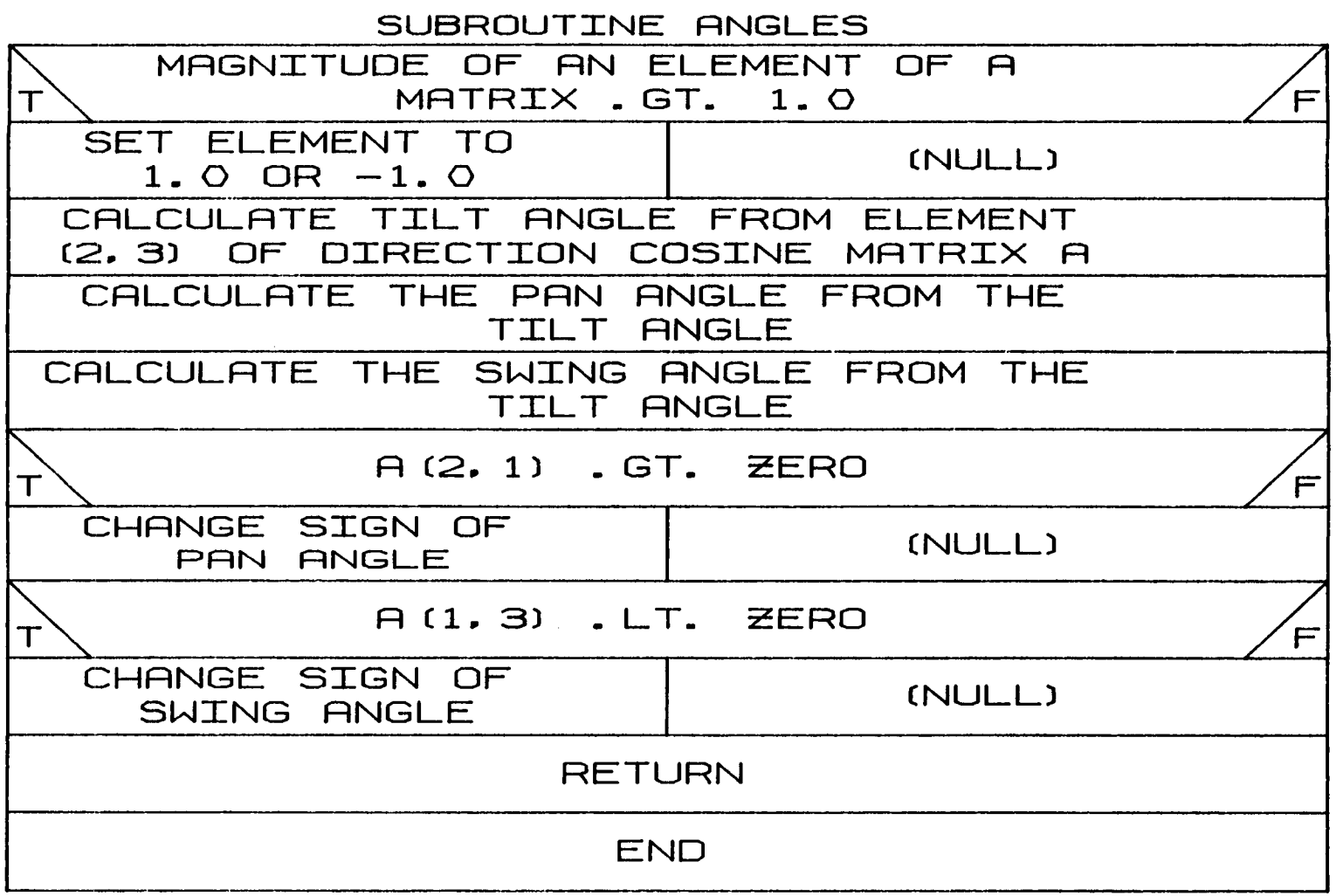
Subroutine INITTAR is called from TRACKING to obtain the coordinates of the target corner points.

SUBROUTINE INITTAR

COORDINATES OF FIRST CORNER OF TARGET ARE PTAR (I, 1) = (-L/2, 0, -W/2)
COORDINATES OF SECOND CORNER OF TARGET ARE PTAR (I, 2) = (L/2, 0, -W/2)
COORDINATES OF THIRD CORNER OF TARGET ARE PTAR (I, 3) = (-L/2, 0, W/2)
COORDINATES OF FOURTH CORNER OF TARGET ARE PTAR (I, 4) = (L/2, 0, W/2)
PTAR (I, 5) = (0, 0, 0)
RETURN
END

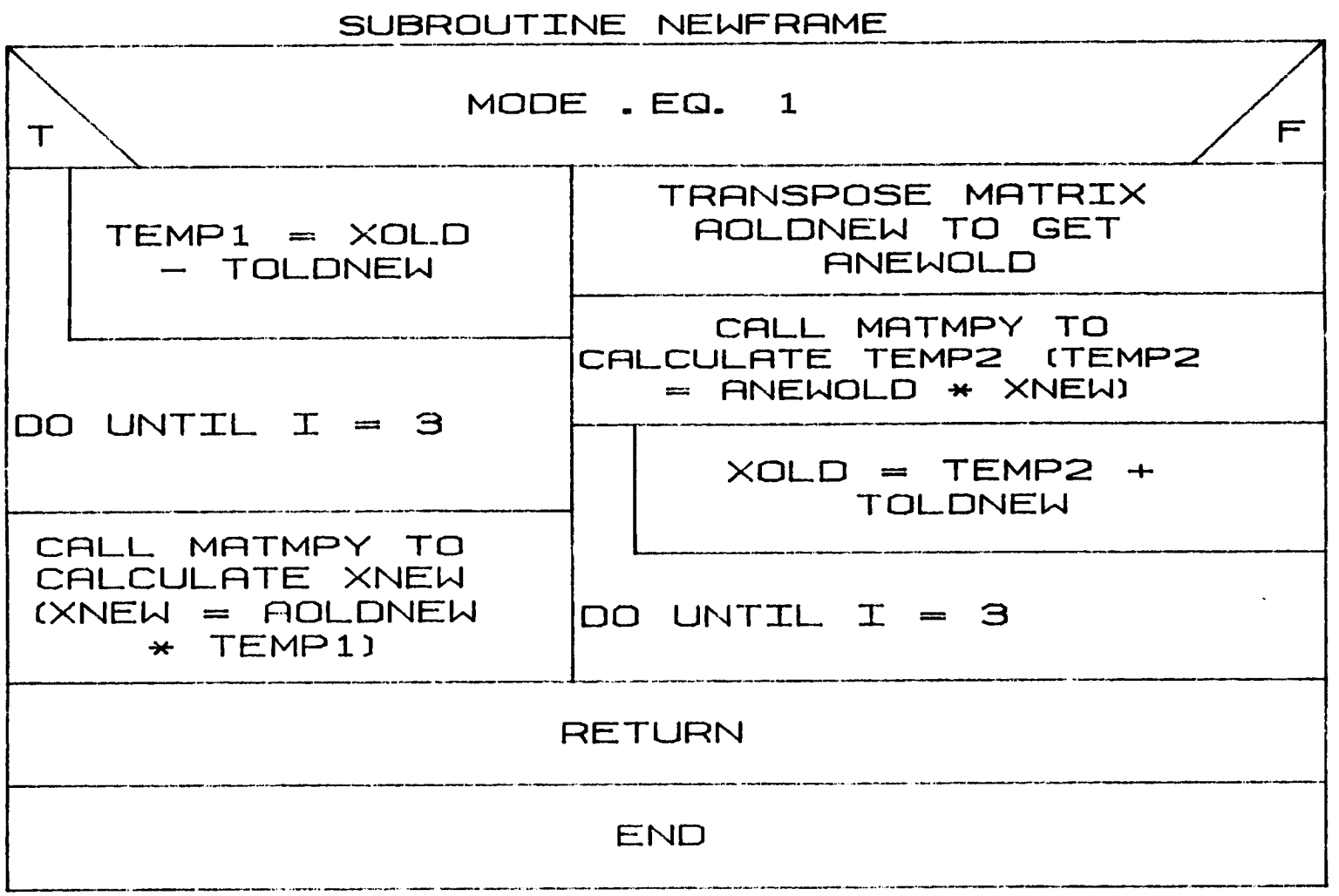
2.3.49 ANGLES

Subroutine ANGLES is called from TRACKING to calculate the Euler angles given a direction cosine matrix.


SUBROUTINE ANGLES
MAGNITUDE OF AN ELEMENT OF A MATRIX .GT. 1.0
T
F
SET ELEMENT TO 1.0 OR -1.0
(NULL)
CALCULATE TILT ANGLE FROM ELEMENT (2,3) OF DIRECTION COSINE MATRIX A
CALCULATE THE PAN ANGLE FROM THE TILT ANGLE
CALCULATE THE SWING ANGLE FROM THE TILT ANGLE
A(2,1) .GT. ZERO
T
F
CHANGE SIGN OF PAN ANGLE
(NULL)
A(1,3) .LT. ZERO
T
F
CHANGE SIGN OF SWING ANGLE
(NULL)
RETURN
END

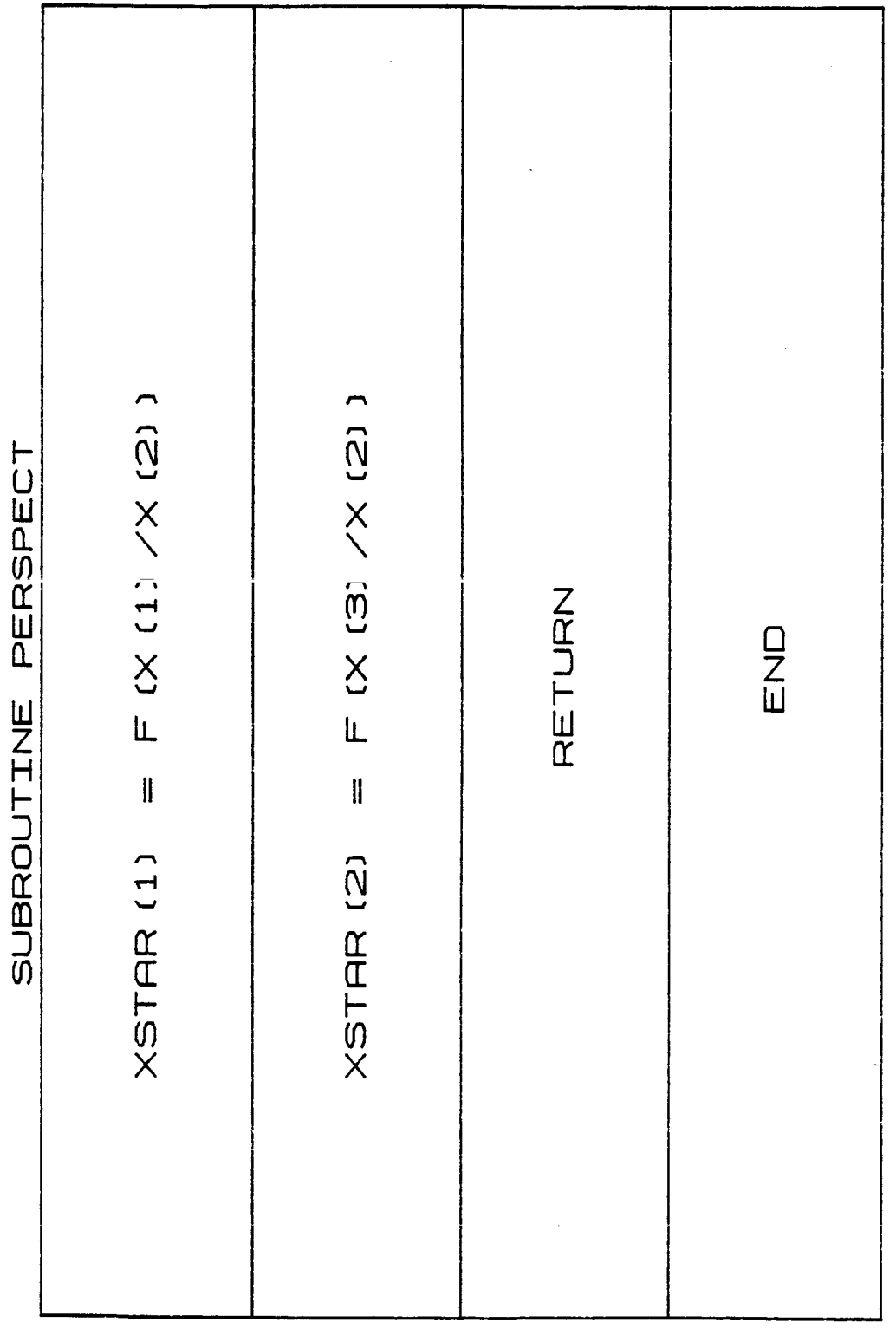
2.3.50 NEWFRAME

Subroutine NEWFRAME is called from TRACKING to obtain the coordinates of a three-dimensional vector in a new coordinate system.


SUBROUTINE NEWFRAME
MODE .EQ. 1
T
F
TEMP1 = XOLD - TOLDNEW
DO UNTIL I = 3
CALL MATMPY TO CALCULATE XNEW (XNEW = AOLDNEW * TEMP1)
TRANSPOSE MATRIX AOLDNEW TO GET ANEWOLD
CALL MATMPY TO CALCULATE TEMP2 (TEMP2 = ANEWOLD * XNEW)
XOLD = TEMP2 + TOLDNEW
DO UNTIL I = 3
RETURN
END

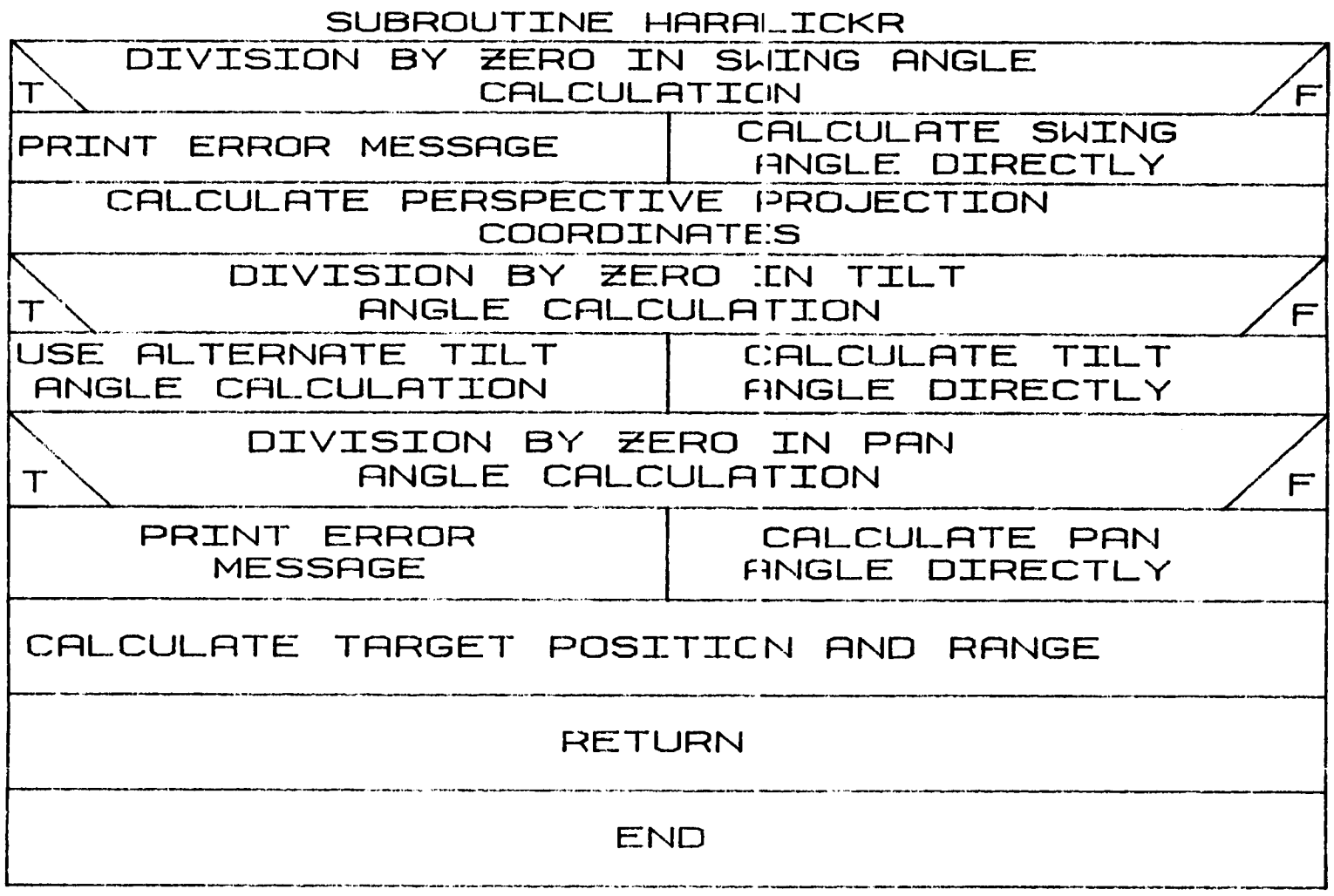
2.3.51 PERSPECT

Subroutine PERSPECT is called from TRACKING to calculate the perspective projection of a three-dimensional vector. The result is a two-dimensional vector.


SUBROUTINE PERSPECT
XSTAR(1) = F(X(1)/X(2))
XSTAR(2) = F(X(3)/X(2))
RETURN
END

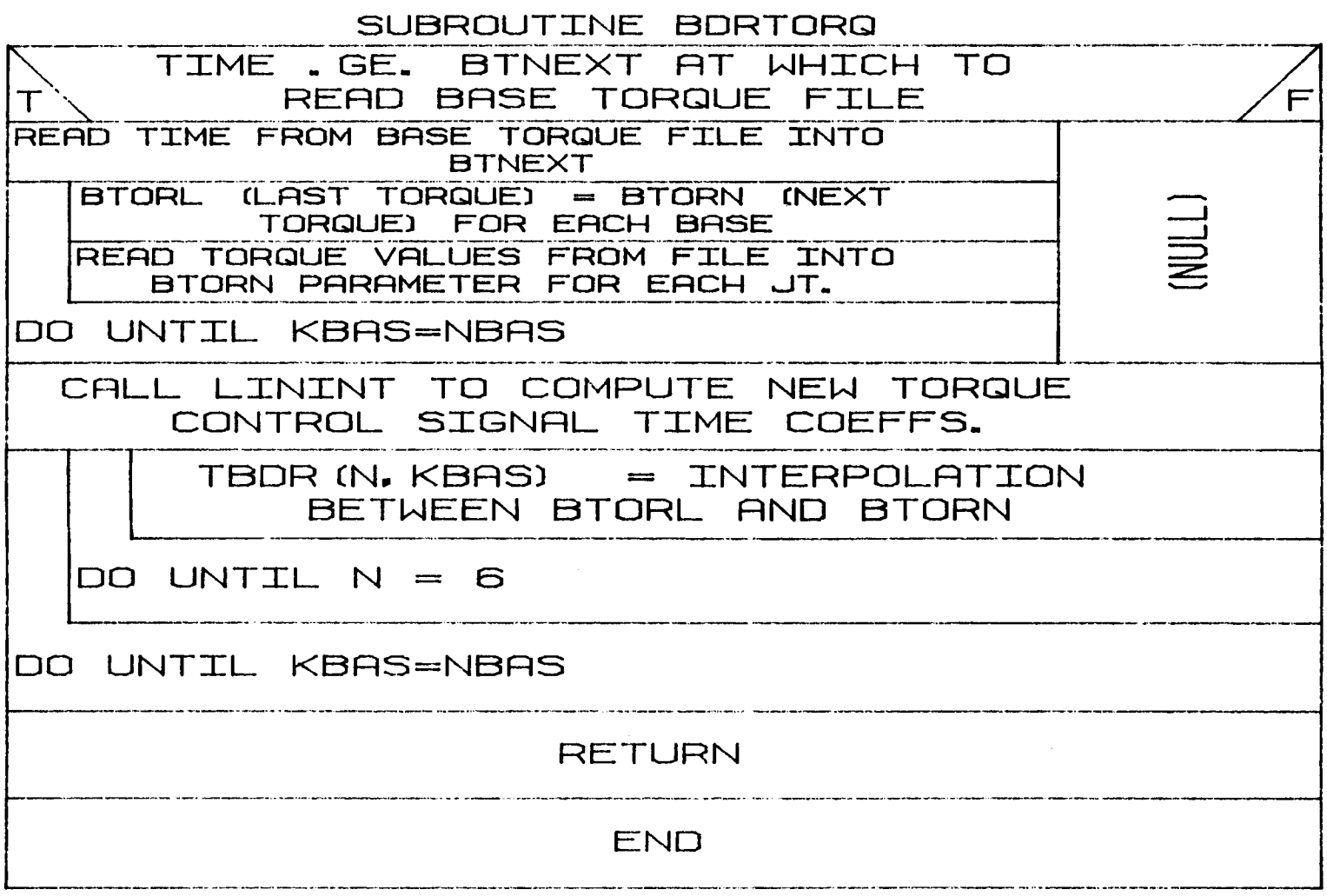
2.3.52 HARALICKR

Subroutine HARALICKER is called from TRACKING and calculates the pointing angles of a camera relative to a rectangular target, based on the perspective projection of the four corners of the rectangle.


SUBROUTINE HARALICKR
DIVISION BY ZERO IN SWING ANGLE CALCULATION
T
F
PRINT ERROR MESSAGE
CALCULATE SWING ANGLE DIRECTLY
CALCULATE PERSPECTIVE PROJECTION COORDINATES
DIVISION BY ZERO IN TILT ANGLE CALCULATION
T
F
USE ALTERNATE TILT ANGLE CALCULATION
CALCULATE TILT ANGLE DIRECTLY
DIVISION BY ZERO IN PAN ANGLE CALCULATION
T
F
PRINT ERROR MESSAGE
CALCULATE PAN ANGLE DIRECTLY
CALCULATE TARGET POSITION AND RANGE
RETURN
END

2.3.53 BDRTORQ

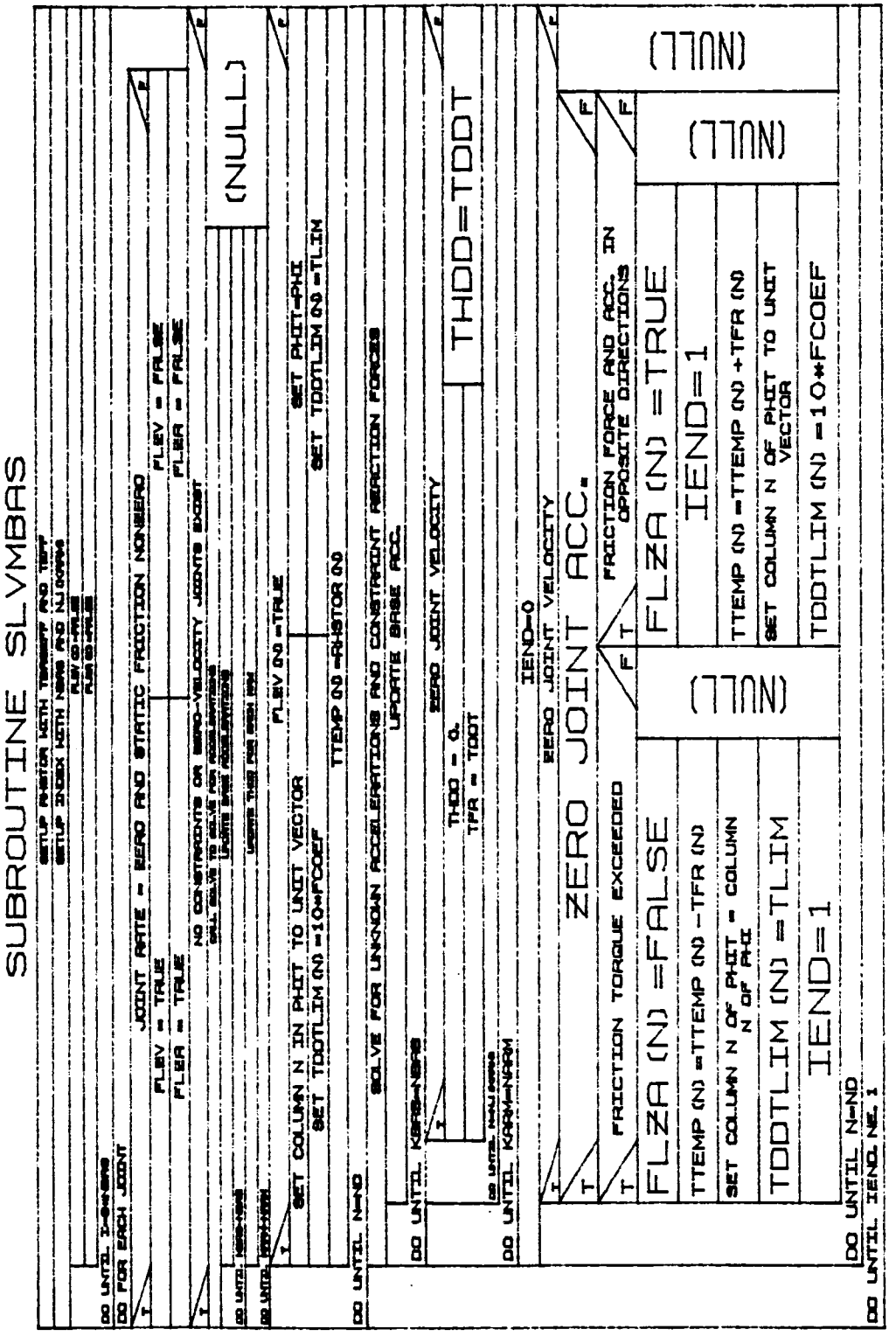
Subroutine BDRTORQ reads base torques and forces from a file and computes new torques and forces by linear interpolation.


SUBROUTINE BDRTORQ
TIME .GE. BTNEXT AT WHICH TO READ BASE TORQUE FILE
T
F
READ TIME FROM BASE TORQUE FILE INTO BTNEXT
BTORL (LAST TORQUE) = BTORN (NEXT TORQUE) FOR EACH BASE
READ TORQUE VALUES FROM FILE INTO BTORN PARAMETER FOR EACH JT.
DO UNTIL KBAS=NBAS
(NULL)
CALL LININT TO COMPUTE NEW TORQUE CONTROL SIGNAL TIME COEFFS.
TBDR (N, KBAS) = INTERPOLATION BETWEEN BTORL AND BTORN
DO UNTIL N = 6
DO UNTIL KBAS=NBAS
RETURN
END

2.3.54 SLVMBAS

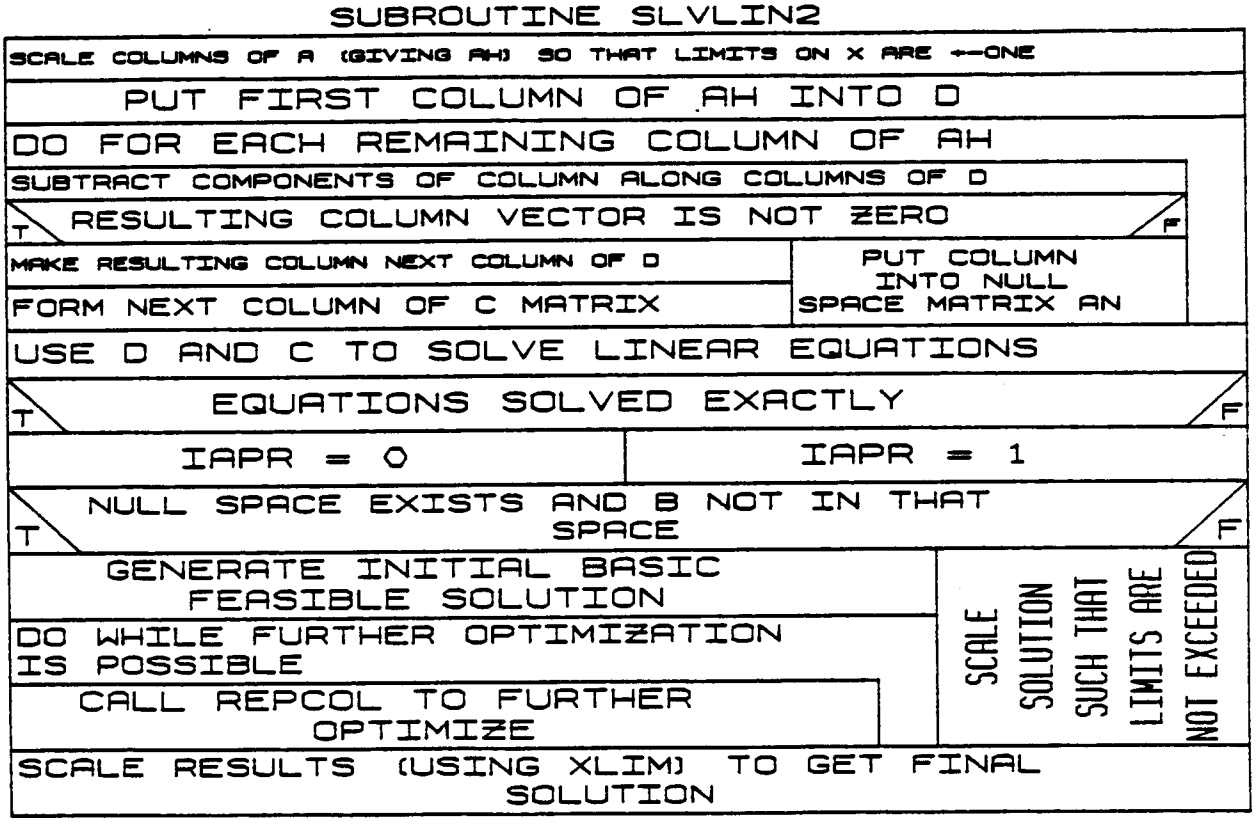
Subroutine SLVMBAS solves for base and joint accelerations for a given arm state and driving forces.

ORIGINAL PAGE IS
OF POOR QUALITY



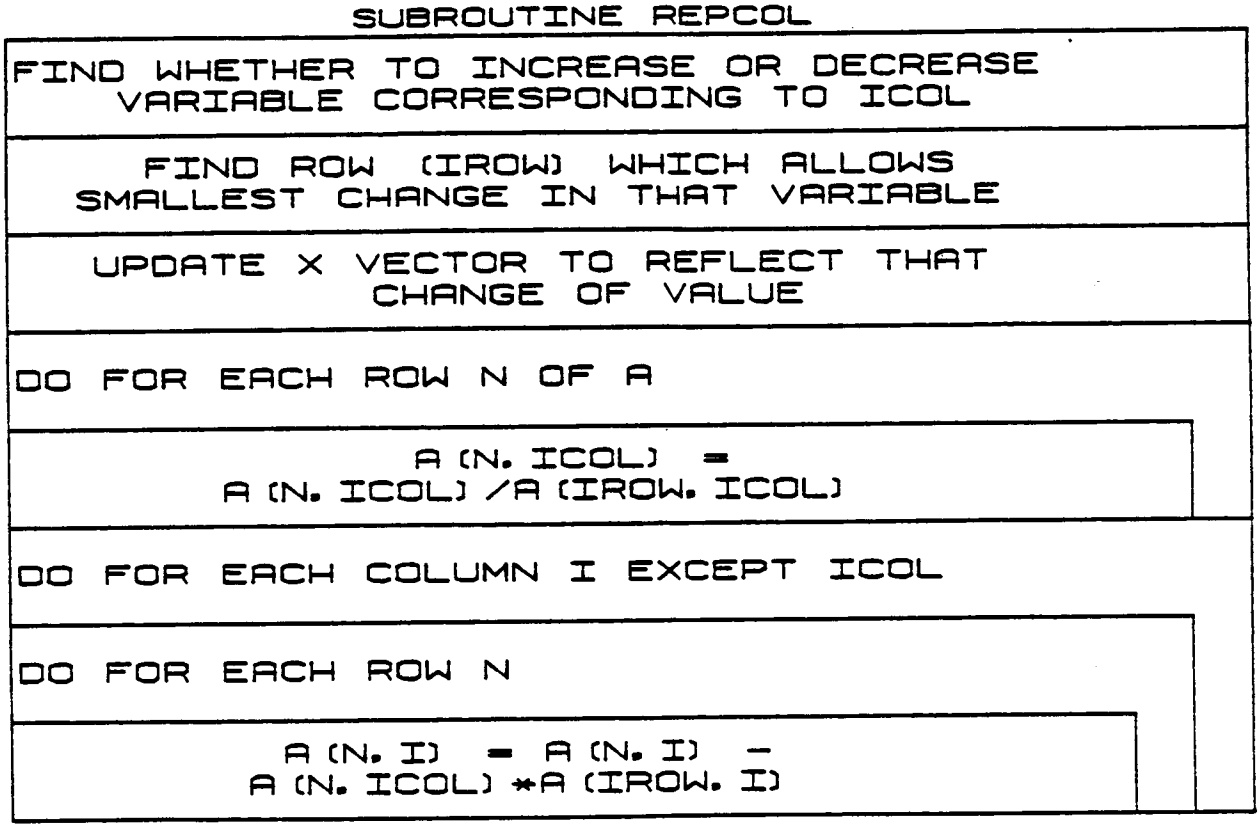
2.4.1 SLVLIN2

SLVLIN2 finds an optimal solution X to a linear set of equations AX = B where the magnitude of each component of X is bounded -XLIM(N)< = X(N) <= XLIM(N). The program first sets up matrices D, which forms an orthogonal basis for the reachable space of A, and C, which provides the conversion from D space to AH space. The matrix AN is also set up; it contains vectors in the null space of H along with the initial solution VH and is used as the tableau for linear programming. Once an initial solution is found, linear programming by a modification of the Simplex method is performed; the magnitude of the result is maximized subject to the constraints on X. This solution is then scaled to give the final solution.



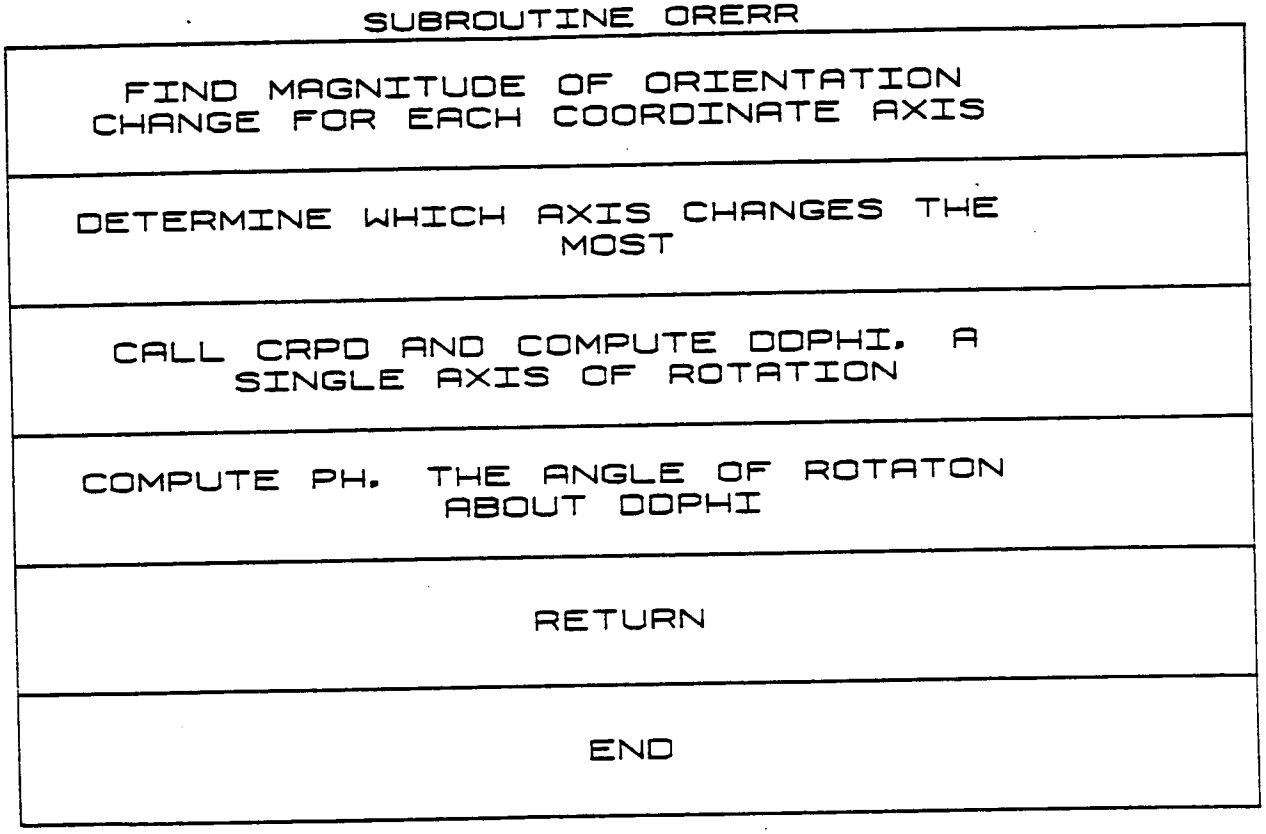
2.4.2 REPCOL

REPCOL replaces column ICOL of matrix A where A represents the tableau for a linear programming problem, X represents the variables, and the limits on each variable are plus or minus one. REPCOL first finds the largest allowable change in the free variable (the variable that corresponds to the column being replaced), and the new constraint variable that limits this change. The solution is updated and tableau A is modified to reflect this change of constraint variables.



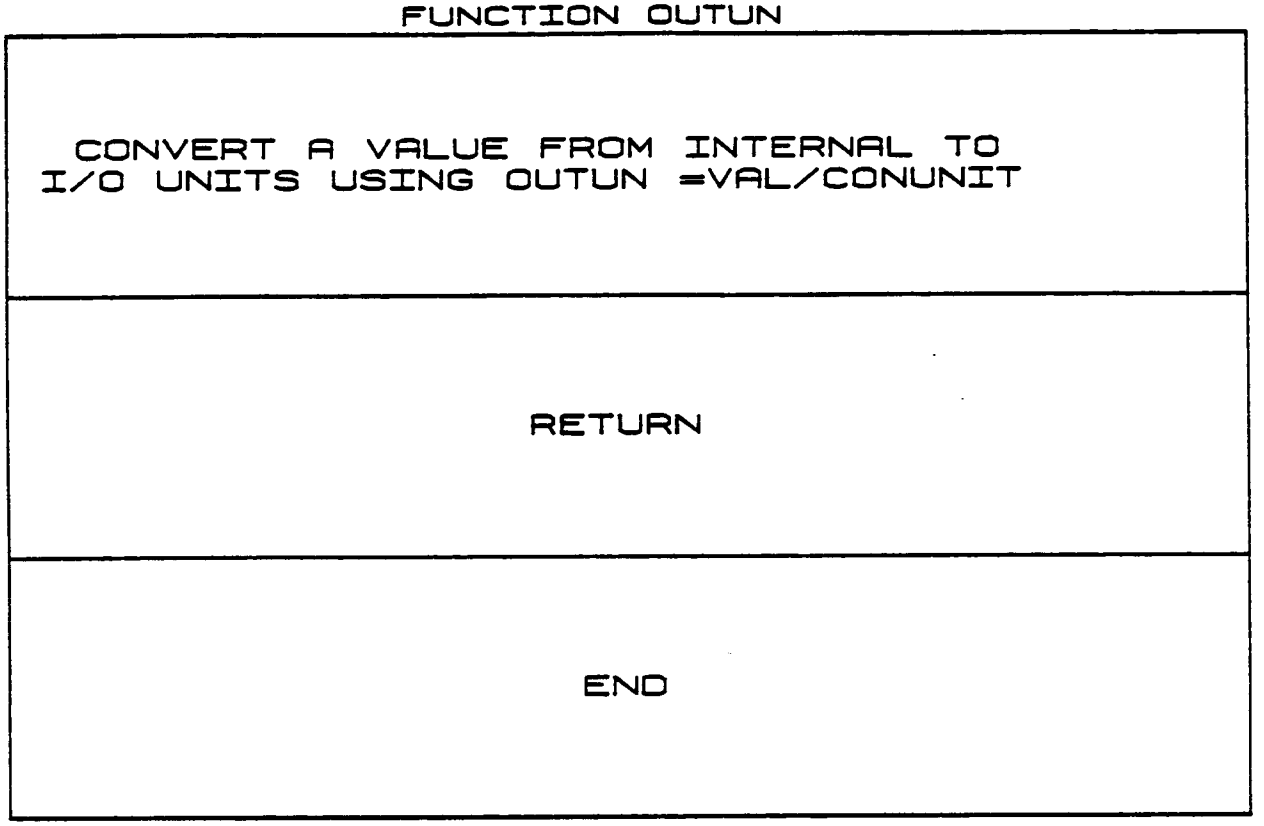
2.4.3 ORERR

Subroutine ORERR is used to find the change in orientation between two coordinate systems. The error in orientation is computed and then transformed into a rotation (magnitude less than pi) about a unique rotation axis. This axis is computed as the cross-product of two of the columns of DOR, and the rotation angle PH is computed by (COS(PH)= 1+.5*(X.DX+Y.DY+Z.DZ), where X.DX is the dot product of the X column of ROR with the X column of DOR, etc.



2.4.4 OUTUN

OUTUN is a function that converts a value from internal (metric) units to the user-specified input/output units by dividing by a conversion factor.



2.4.5 ICVTATD

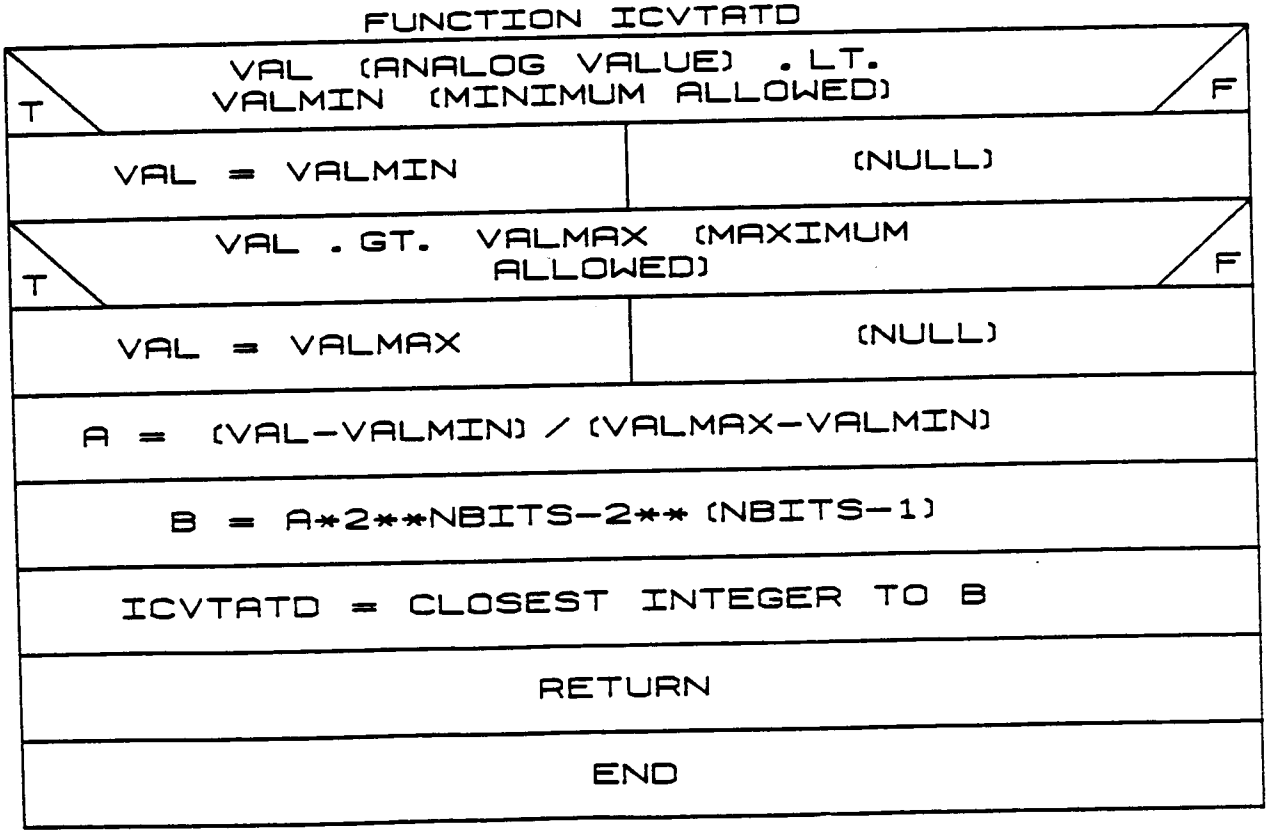
ICVTATD is a function that returns a digitized number when given a real value. If the value is outside an allowable minimum or maximum, it is set equal to the appropriate limit. The digitized number is then computed using the following relation:

$$B = \frac{(VAL - VALMIN)}{(VALMAX - VALMIN)} * \left(2^{MBITS} - 2^{(NBITS-1)}\right)$$

where

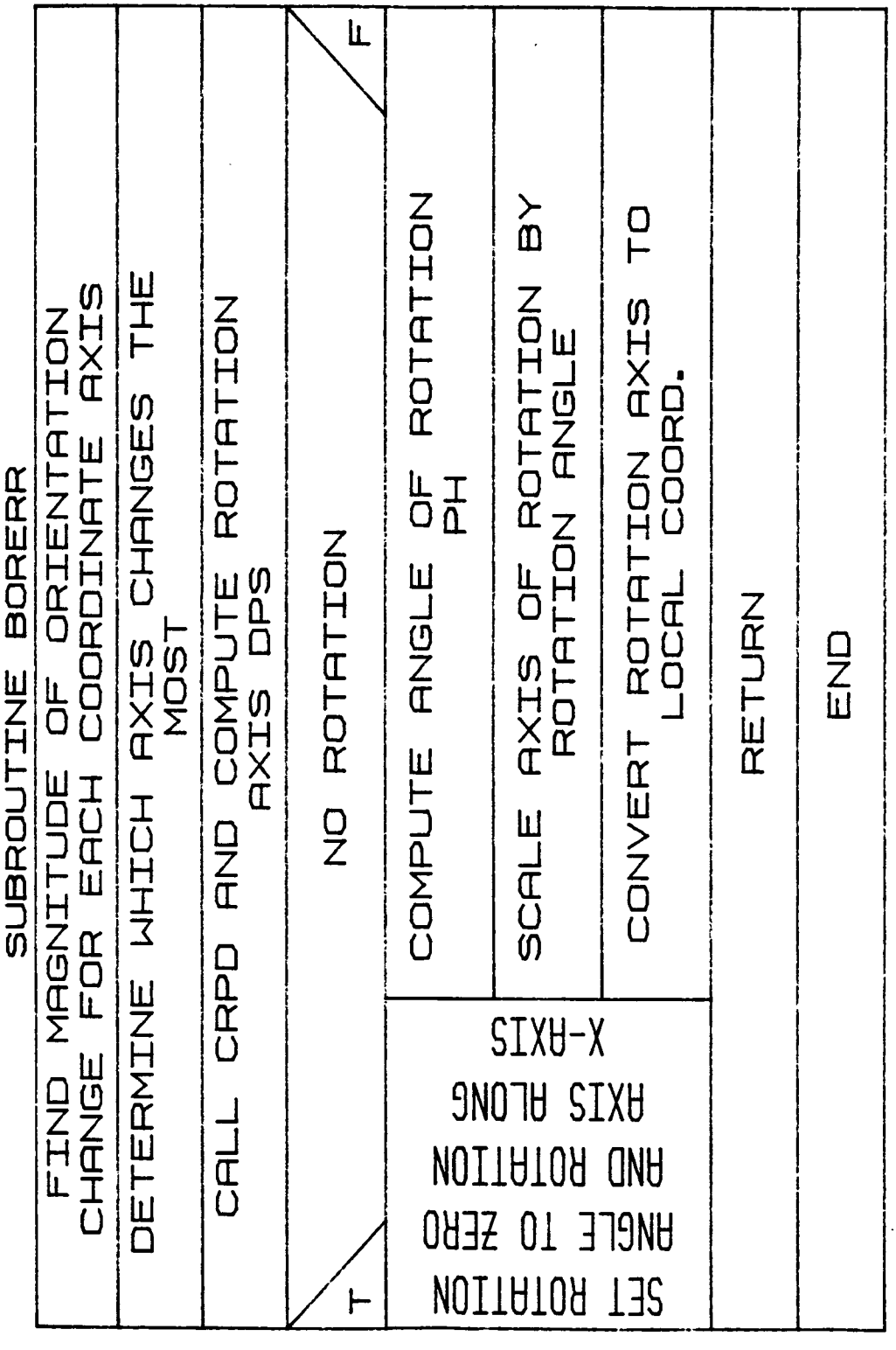
NBITS = number of bits available for digitized value,
VALMIN = minimum allowable value for VAL,
VALMAX = maximum allowable value for VAL,
VAL = output value.

ICVTATD, the digitized number, is set equal to the closest integer to B.



2.4.6 BORERR

Subroutine BORERR is used to find the change in orientation between two coordinate systems. The error in orientation is computed and then transformed into a rotation (magnitude less than pi) about a rotation axis referenced to the base coordinate system.


SUBROUTINE BORERR
FIND MAGNITUDE OF ORIENTATION CHANGE FOR EACH COORDINATE AXIS
DETERMINE WHICH AXIS CHANGES THE MOST
CALL CRPD AND COMPUTE ROTATION AXIS DPS
NO ROTATION
T
F
SET ROTATION ANGLE TO ZERO AND ROTATION AXIS ALONG X-AXIS
COMPUTE ANGLE OF ROTATION PH
SCALE AXIS OF ROTATION BY ROTATION ANGLE
CONVERT ROTATION AXIS TO LOCAL COORD.
RETURN
END

ORIGINAL PAGE IS
OF POOR QUALITY

The program POSTDRVR is the postprocessing function driver. The following set of routine functional descriptions and VCLRs (visual control logic representations) are the modules found in the postprocessor function of ROBSIM.

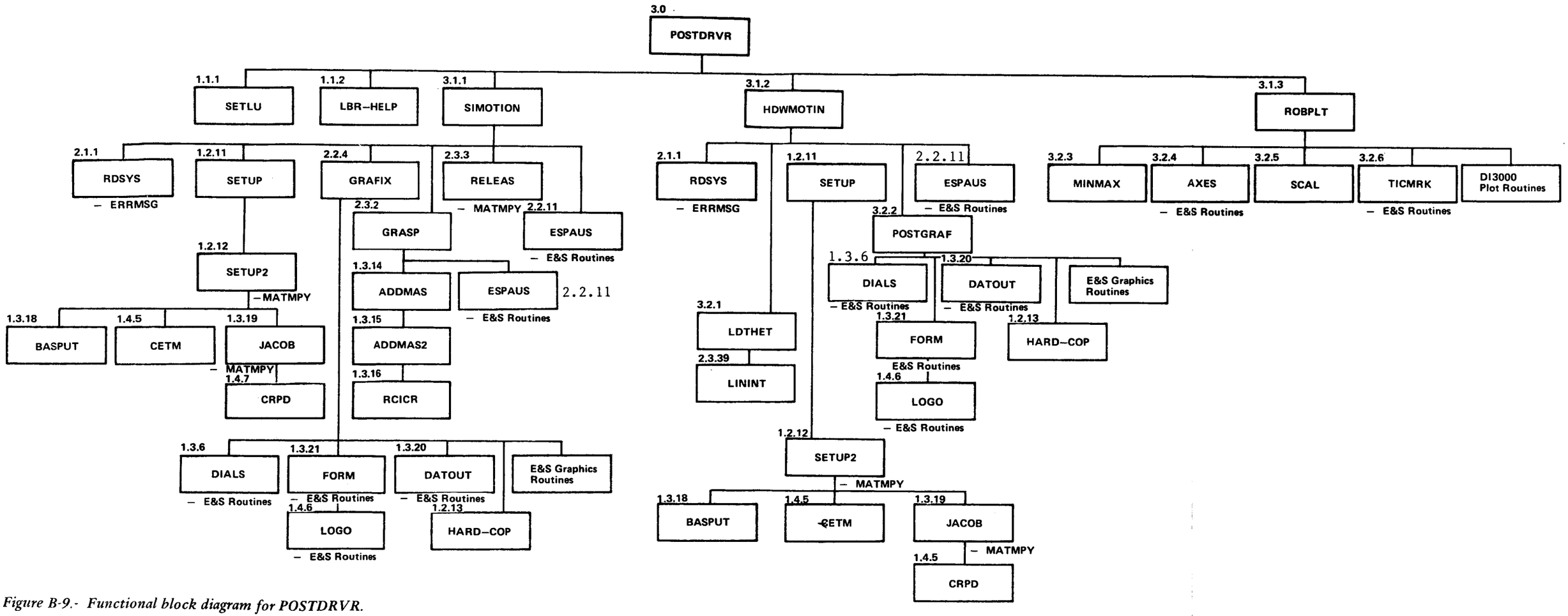


Figure B-9.- Functional block diagram for POSTDRVR.

FOLDOUT FRAME

2 FOLDOUT FRAME

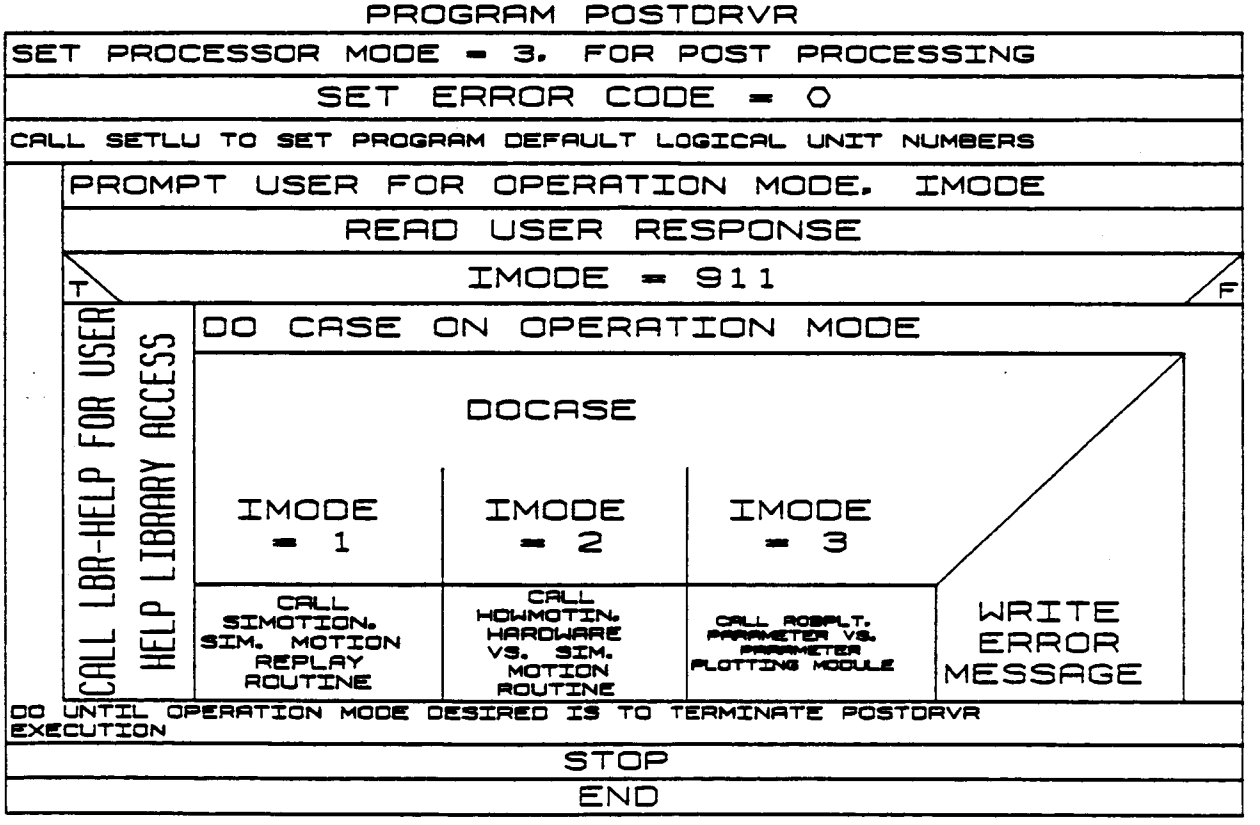
TAble B-VII. - PROGRAMS EMPLOYED IN POSTDRVR

3.0	POSTDRVR
3.1.1	SIMOTION
3.1.2	HDWMOTIN
3.1.3	ROBPLT
3.2.1	LDTHET
3.2.2	POSTGRAF
3.2.3	MINMAX
3.2.4	AXES
3.2.5	SCAL
3.2.6	TICMRK

ORIGINAL PAGE IS
OF POOR QUALITY

3.0 POSTDRVR

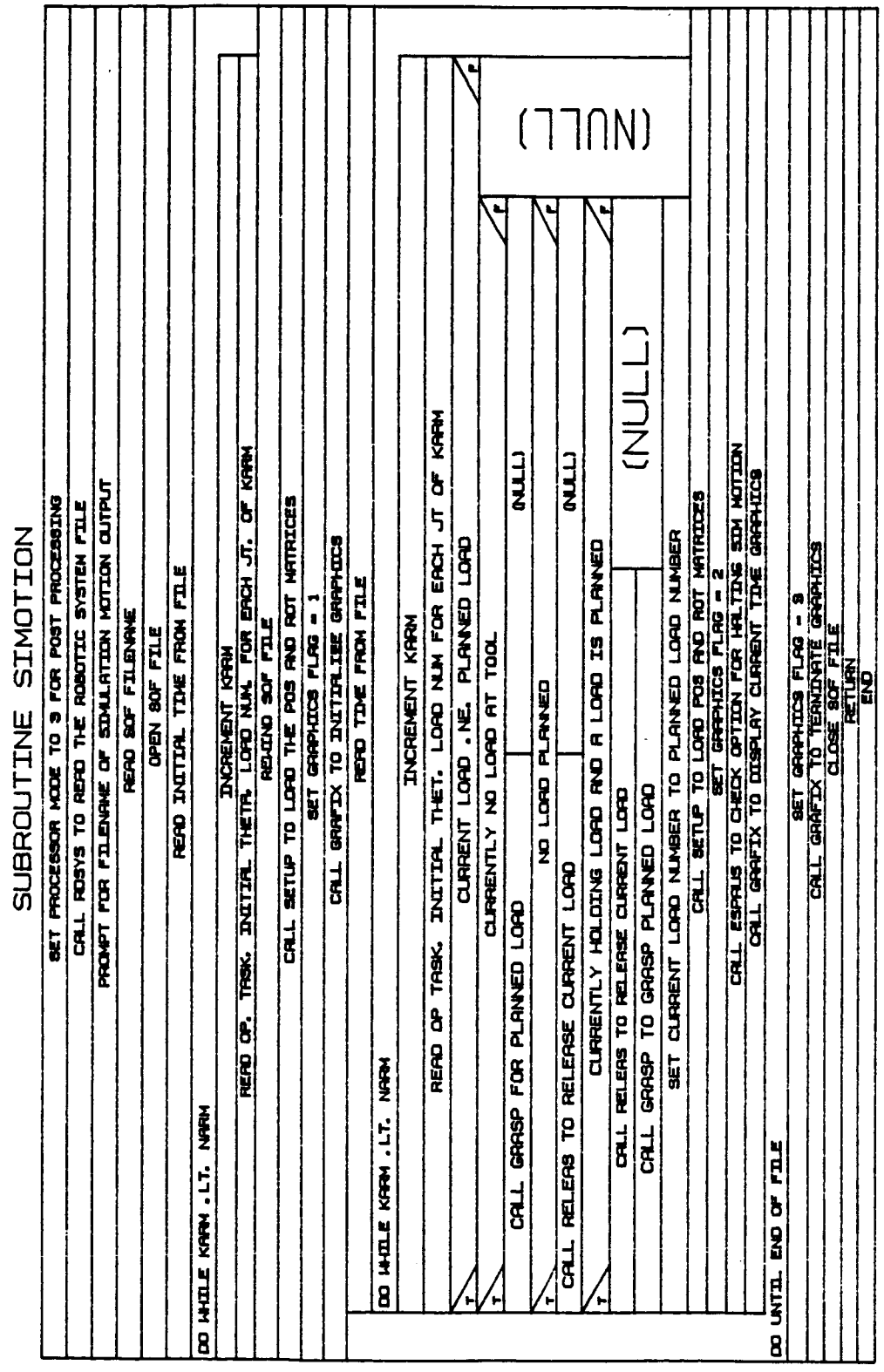
The program POSTDRVR is the postprocessing function driver. It operates in an interactive mode, prompting the user for the postprocessing option desired: replay robotic system simulation motion, replay simulation versus hardware motion, parameter versus parameter plots, or terminate POSTDRVR execution. For simulation replay, option 1, subroutine SIMOTION, is called. Option 2 provides a comparison of hardware and the corresponding simulation motion through subroutine HDWMOTIN. If option 3 is selected, ROBPLT plots any of the data computed and written to one of the seven types of plot file packages during the requirements or simulation analysis tools functions.



3.1.1 SIMOTION

SIMOTION is called during the postprocessing function to provide a replay of the robotic system motion produced during a previous run of the requirements or simulation phase of the analysis tools function. It opens the chosen robotic system geometry file and simulation output file for each graphics replay, and calls GRAFIX with the displacements at each time step to update the system motion display.

ORIGINAL PAGE IS
OF POOR QUALITY



3.1.2 HDWMOTIN

HDWMOTIN is called during the postprocessing function to provide a replay of the robotic system motion produced during the requirements/simulation analysis tools functions versus the actual motion that occurred during the corresponding hardware run. It opens the chosen system geometry file, simulation output file for graphics replay and hardware file containing recorded joint theta values. It calls POSTGRAF with the hardware and simulation displacements at each time step to update the system motion display.

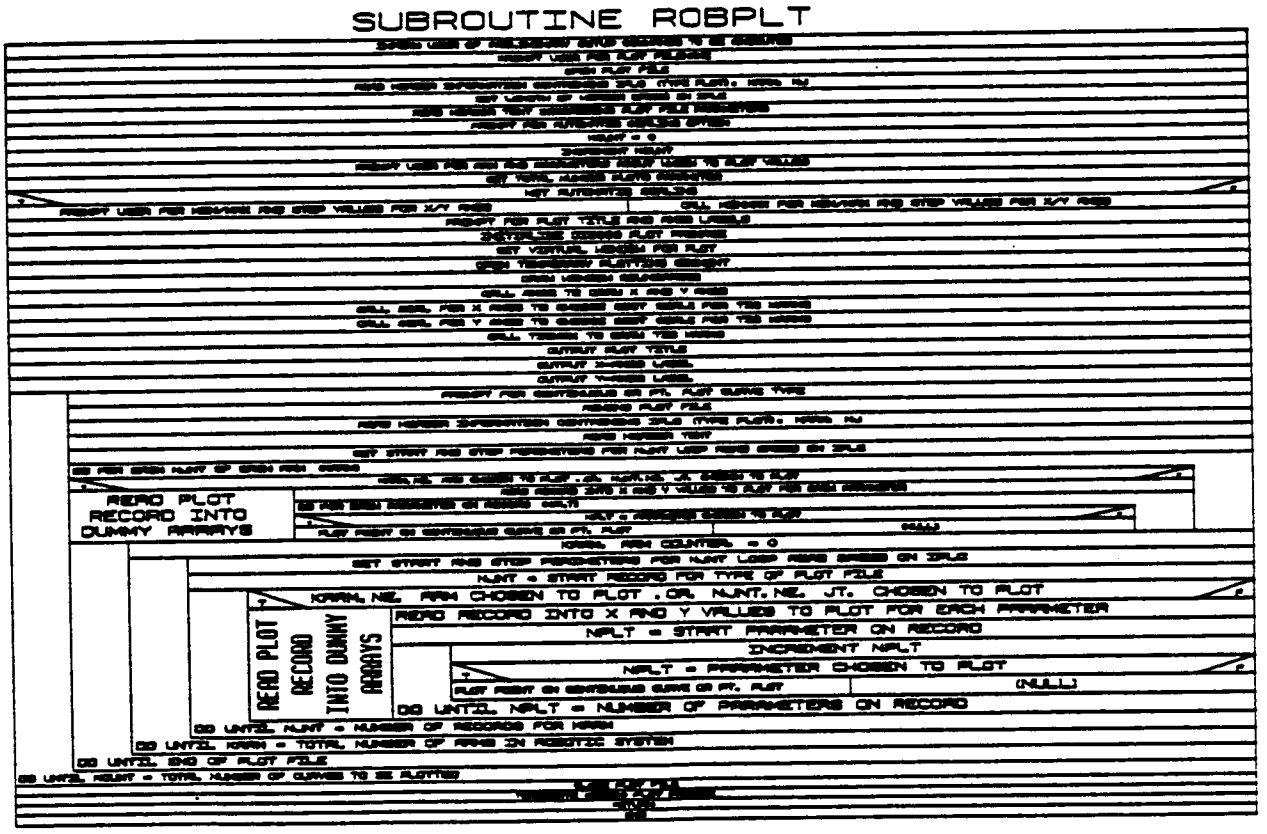
SUBROUTINE HDWMOTIN

```
SET PROCESSOR MODE TO 3 FOR POST PROCESSING
CALL ROSYS TO READ THE ROBOTIC SYSTEM FILE
PROMPT, READ FILENAME OF SIMULATION MOTION OUTPUT FILE, SOF
SET HARDWARE THETA READ FLAG, IRDTHET, = 1
OPEN SOF FILE
PROMPT, READ FILENAME AND OPEN HARDWARE THETA FILE
READ INITIAL TIME FROM SOF
SET THE TIME CHECK INCREMENT PARAMETERS
CALL LDTHET TO LOAD THE HARDWARE THETA VALUES
LOAD HWANG ARRAY WITH HARDWARE THETAS FOR EACH JT. OF EACH ARM
CALL SETUP TO LOAD THE HARDWARE POSITION AND ROTATION MATRICES
LOAD HWPOS AND HWROT WITH VALUES FOR EACH JT. OF EACH ARM
DO WHILE KARM .LT. NARM
    INCREMENT KARM
    READ FROM SOF, OP TASK AND INITIAL THET FOR EACH JT OF KARM
    CALL SETUP TO LOAD THE SIM POS AND ROT MATRICES
    CALL POSTGRAF TO INITIALIZE GRAPHICS W/ FLAG = 1
    READ TIME FROM SOF
    CALL LDTHET TO LOAD THE HARDWARE THETAS
    LOAD HWANG ARRAY WITH THETA VALUES FOR EACH JT. OF EACH ARM
    CALL SETUP TO LOAD THE HARDWARE POS AND ROT MATRICES
    LOAD HWPOS AND HWROT WITH VALUES FOR EACH JT. OF EACH ARM
    DO WHILE KARM .LT. NARM
        INCREMENT KARM
        READ FROM SOF, OP TASK AND INITIAL THET FOR EACH JT OF KARM
        CALL SETUP TO LOAD THE SIM POS AND ROT MATRICES
        CALL ESPAUS TO CHECK OPTION FOR HALTING SIM MOTION
        CALL POSTGRAF TO DISPLAY CURRENT TIME POST GRAPHICS W/ FLAG = 2
    DO UNTIL END OF FILE
    CALL POSTGRAF TO TERMINATE GRAPHICS W/ FLAG = 3
    CLOSE SOF FILE AND HARDWARE THETA FILE
    RETURN
    END
```

ORIGINAL PAGE IS
OF POOR QUALITY

3.1.3 ROBPLT

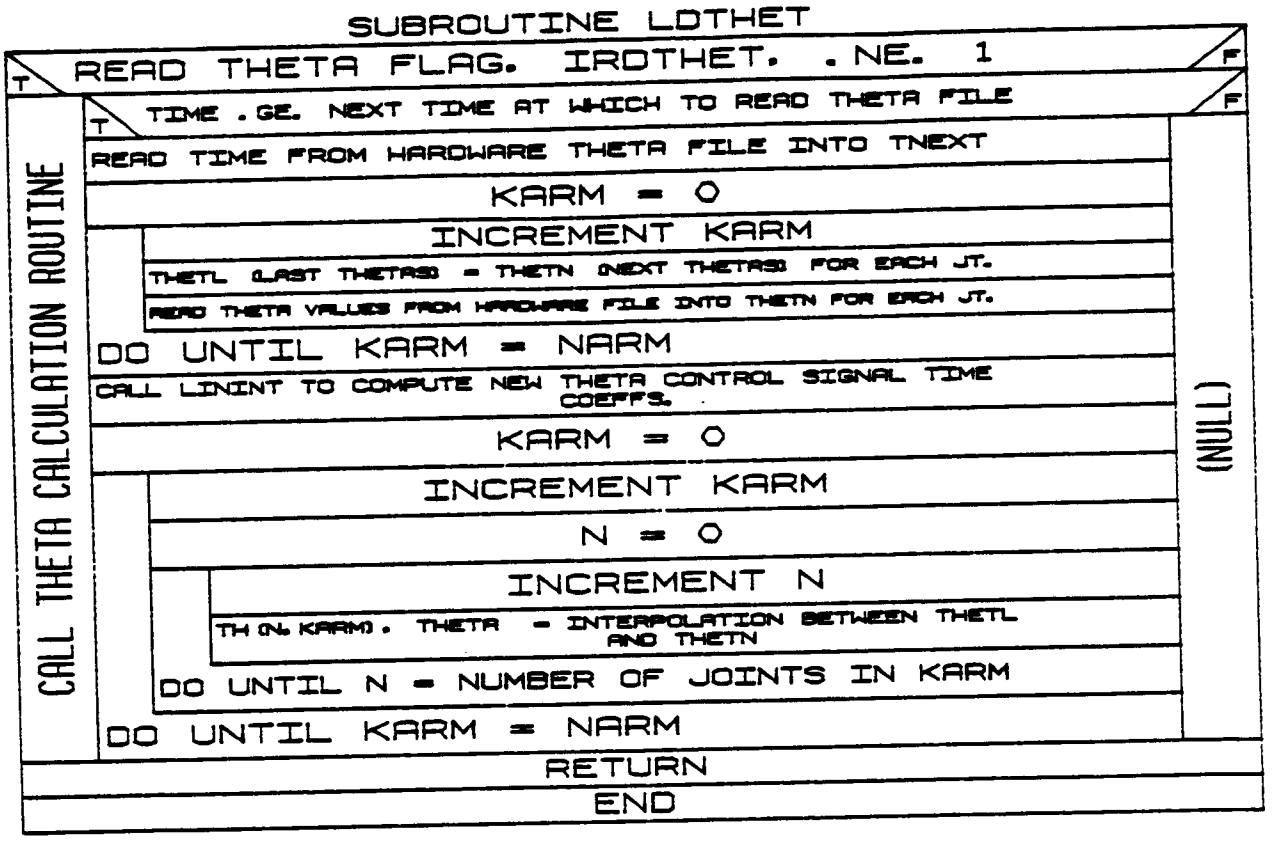
The ROBPLT subroutine plots the contents of one of several choices for plot package formats on a Hewlett-Packard X-Y plotter or a VAX VT125 graphics terminal. It uses exclusively the DI3000 plot package. ROBPLT requests the user to select the appropriate one of seven plot file types that was written at the user's discretion during the requirements or simulation analysis tools functions: the brief package, the end-effector package, the joint positions package, the reaction forces package, a combination of the above four packages, the PID control package or the force/torque control package.



ORIGINAL PAGE IS
OF POOR QUALITY

3.2.1 LDTHET

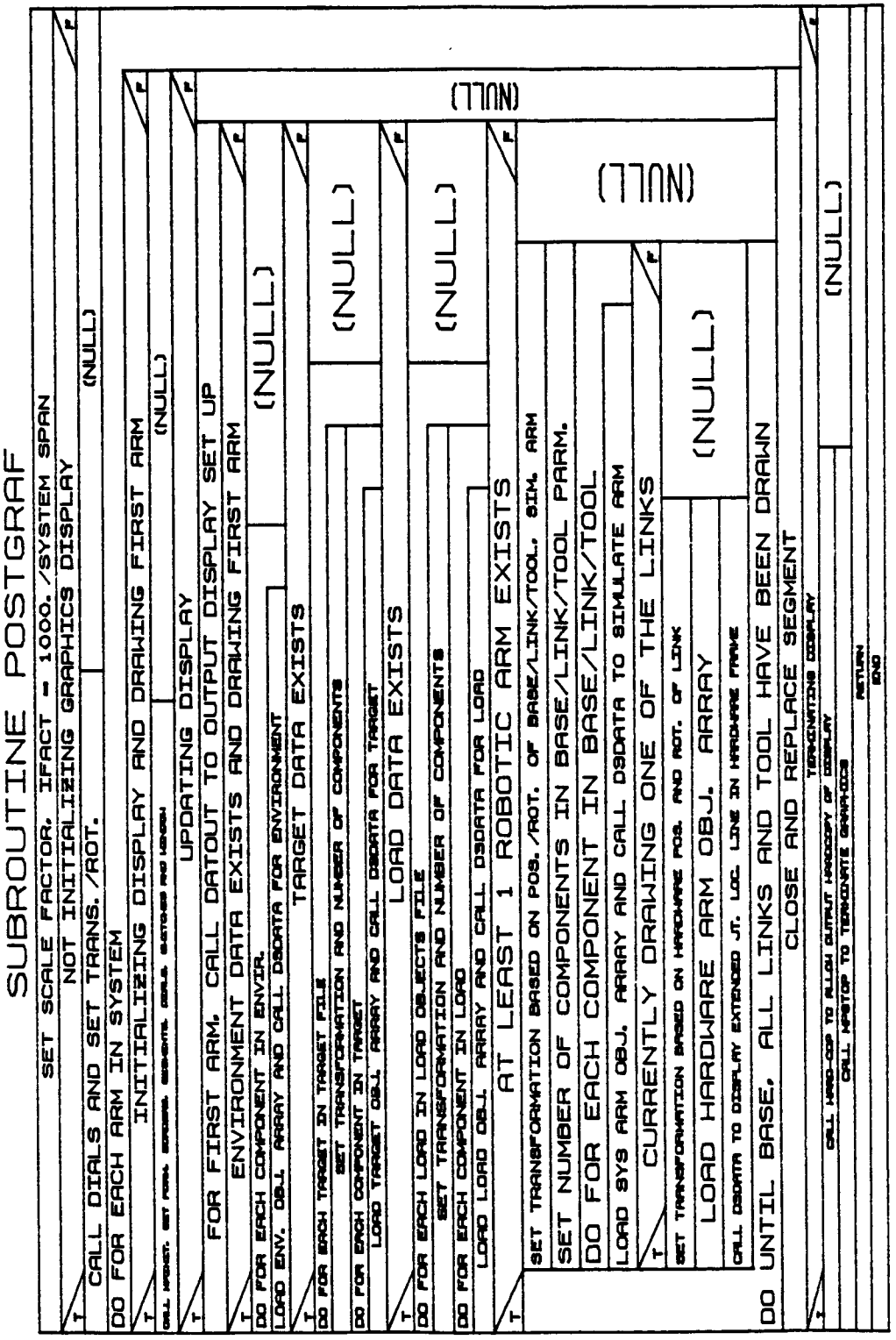
The LDTHET routine loads the theta values for each joint from direct read of the hardware control theta values file. It is called from HDWMOTIN during the postprocessing function for each simulation time step. There is a limit of one theta signal value for each joint that can be read.



C-4

3.2.2 POSTGRAF

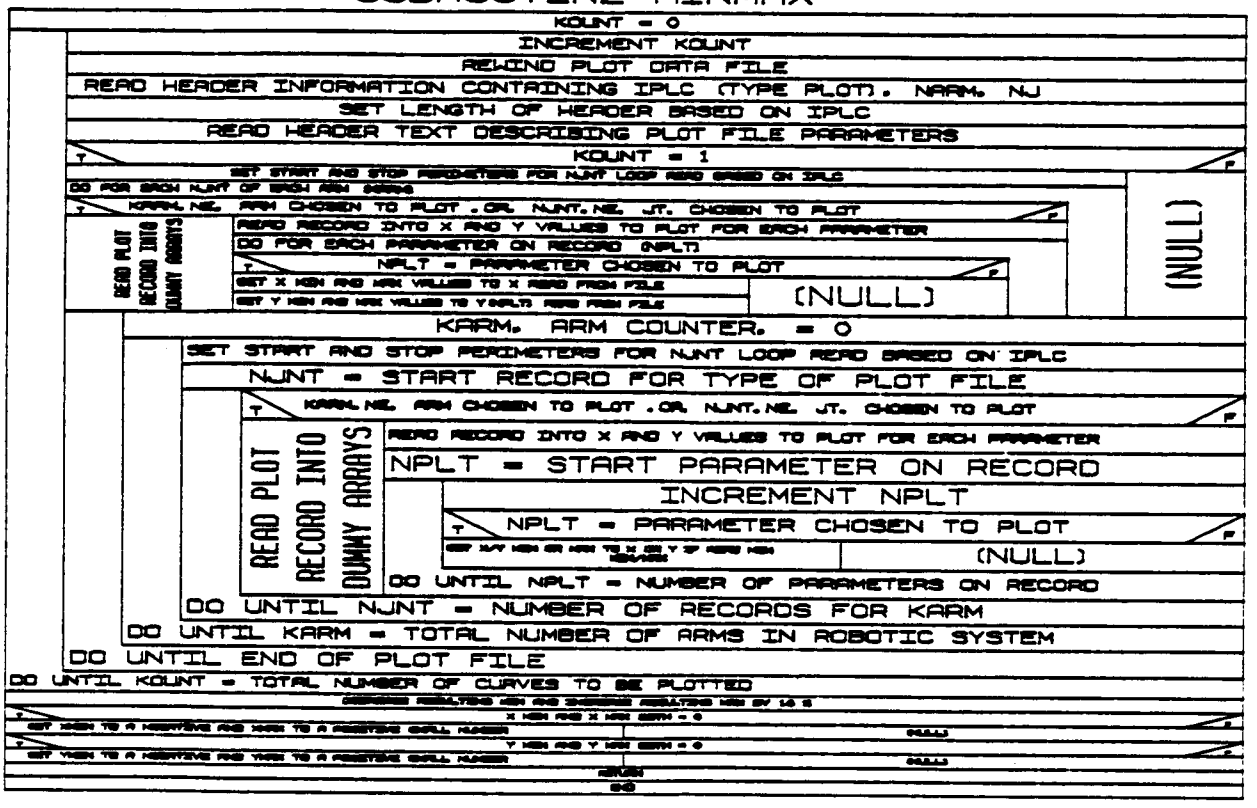
Subroutine POSTGRAF provides the motion graphics capability in the post-processing function for HDWMOTIN, a replay of the simulation motion versus actual hardware motion. The value of the difference in the simulation and hardware thetas is displayed, along with the environment, robotic arms, targets and load objects.


SUBROUTINE POSTGRAF
SET SCALE FACTOR. IFACT = 1000. /SYSTEM SPAN
NOT INITIALIZING GRAPHICS DISPLAY
CALL DIALS AND SET TRANS./ROT.
(NULL)
DO FOR EACH ARM IN SYSTEM
INITIALIZING DISPLAY AND DRAWING FIRST ARM
(NULL)
UPDATING DISPLAY
FOR FIRST ARM, CALL DATOUT TO OUTPUT DISPLAY SET UP
ENVIRONMENT DATA EXISTS AND DRAWING FIRST ARM
(NULL)
DO FOR EACH COMPONENT IN ENVIR.
LOAD ENV. OBJ. ARRAY AND CALL DSDATA FOR ENVIRONMENT
TARGET DATA EXISTS
DO FOR EACH TARGET IN TARGET FILE
SET TRANSFORMATION AND NUMBER OF COMPONENTS
(NULL)
DO FOR EACH COMPONENT IN TARGET
LOAD TARGET OBJ. ARRAY AND CALL DSDATA FOR TARGET
LOAD DATA EXISTS
DO FOR EACH LOAD IN LOAD OBJECTS FILE
SET TRANSFORMATION AND NUMBER OF COMPONENTS
(NULL)
DO FOR EACH COMPONENT IN LOAD
LOAD LOAD OBJ. ARRAY AND CALL DSDATA FOR LOAD
AT LEAST 1 ROBOTIC ARM EXISTS
(NULL)
SET TRANSFORMATION BASED ON POS./ROT. OF BASE/LINK/TOOL, SIM. ARM
SET NUMBER OF COMPONENTS IN BASE/LINK/TOOL PARM.
DO FOR EACH COMPONENT IN BASE/LINK/TOOL
LOAD SYS ARM OBJ. ARRAY AND CALL DSDATA TO SIMULATE ARM
CURRENTLY DRAWING ONE OF THE LINKS
SET TRANSFORMATION BASED ON HARDWARE POS. AND ROT. OF LINK
LOAD HARDWARE ARM OBJ. ARRAY
(NULL)
CALL DSDATA TO DISPLAY EXTENDED JT. LOC. LINE IN HARDWARE FRAME
DO UNTIL BASE, ALL LINKS AND TOOL HAVE BEEN DRAWN
CLOSE AND REPLACE SEGMENT
TERMINATING DISPLAY
CALL HARD-COP TO ALLOW OUTPUT HARDCOPY OF DISPLAY
(NULL)
CALL KPSTOP TO TERMINATE GKS-3D
RETURN
END

ORIGINAL PAGE IS
OF POOR QUALITY

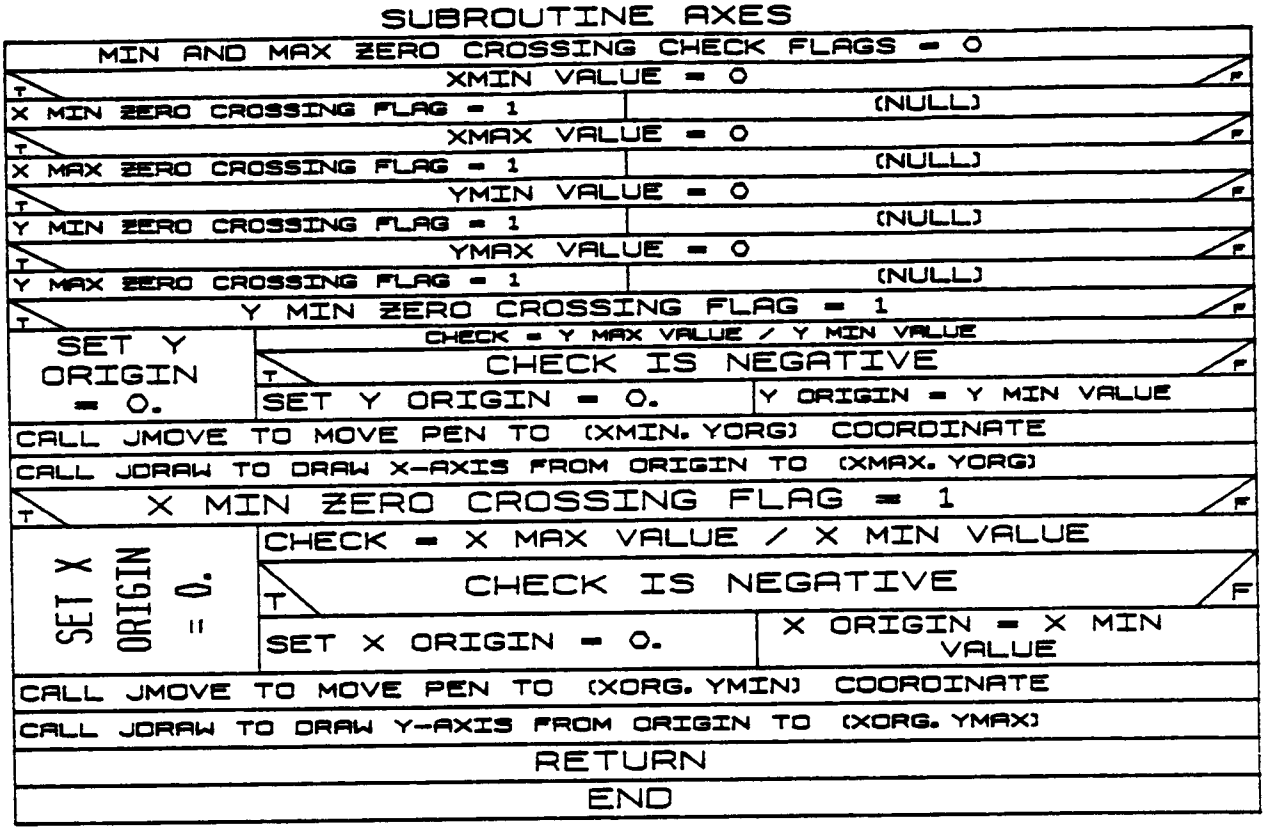
3.2.3 MINMAX

MINMAX searches the postprocessor plot file for the maximum and minimum values to be used in scaling the axes of the plot. The x and y minimums and maximums are found for all parameters the user chooses for plotting.



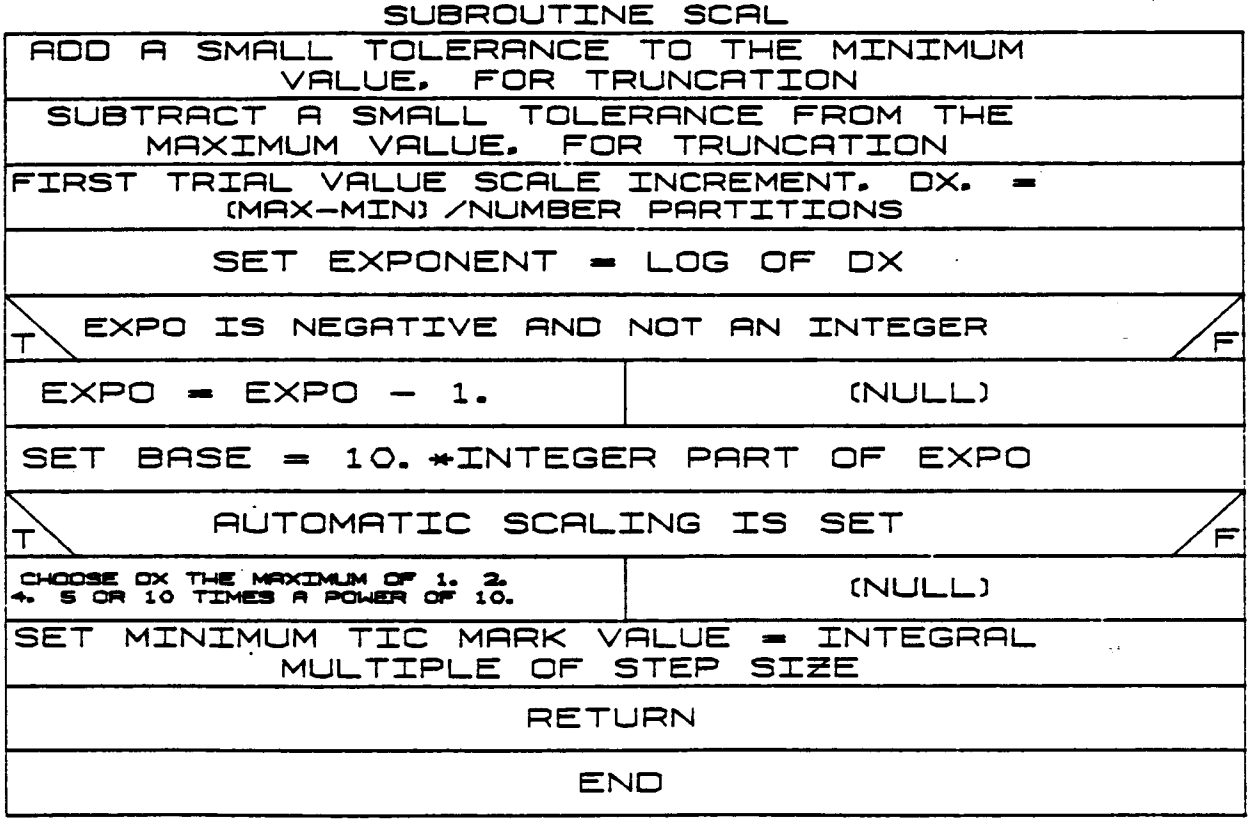
3.2.4 AXES

Subroutine AXES draws the x and y axes for a plot during the x-y plotting option of the postprocessor.



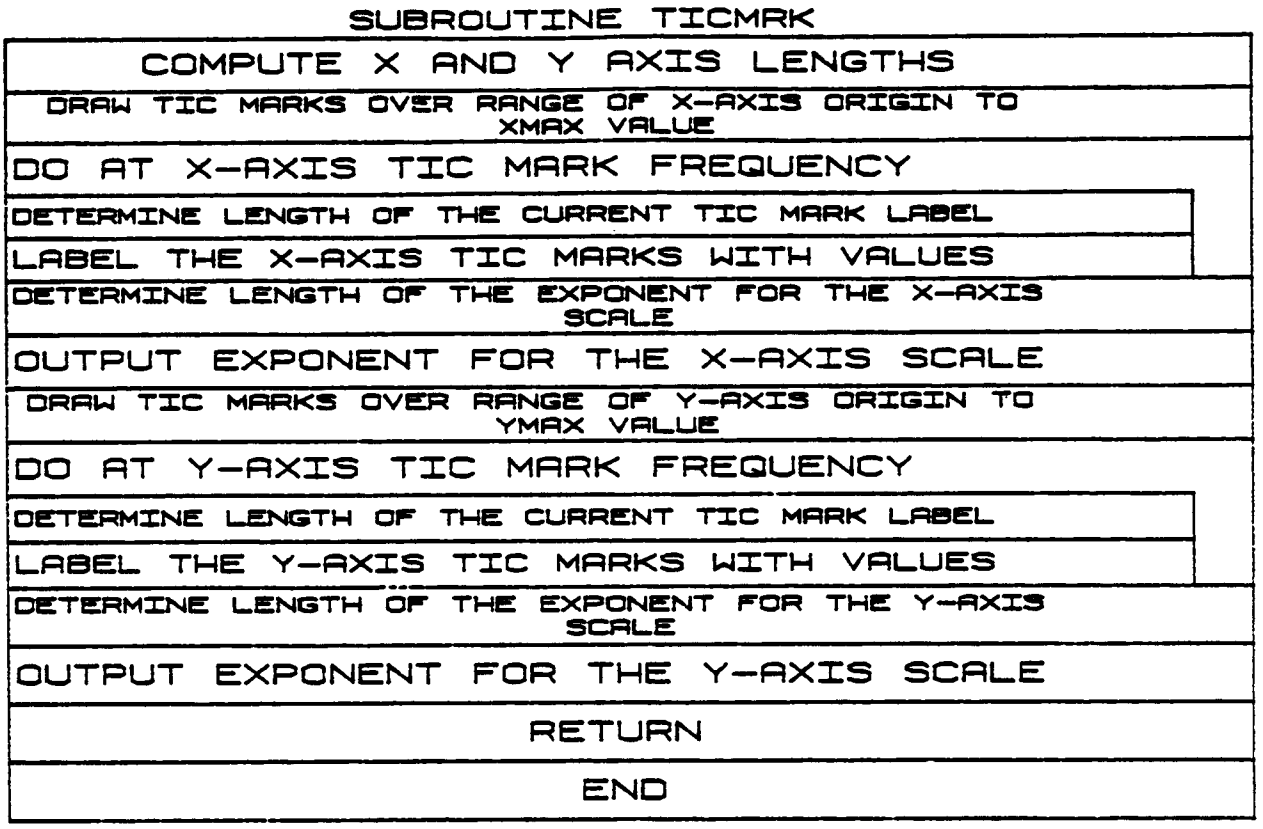
3.2.5 SCAL

For both user-selected automatic or specified scaling of the postprocessor plot file, routine SCAL is called from the ROBPLT option. It chooses the most appropriate scale for the x- and y-axis tic marks. It finds the exponent of the scale base, the tic mark spacing and the minimum tic mark value. The minimum value, XI, to be used for the scale, and DX, the scale increment between tic marks, are chosen to satisfy specific constraints.



3.2.6 TICMRK

Routine TICMRK actually draws and labels the tic marks for a plot during the postprocessor function.



THE PREPROCESSING FUNCTION

The program PREPDRVR is the preprocessor function driver. It operates in an interactive mode, prompting the user to for the preprocessor option desired. Valid options are, create or modify a CAD/CAM object file or terminate PREPDRVR program execution. Figure B-10 shows the functional diagram for PREPDRVR and Table B-VIII lists the subroutine employed.

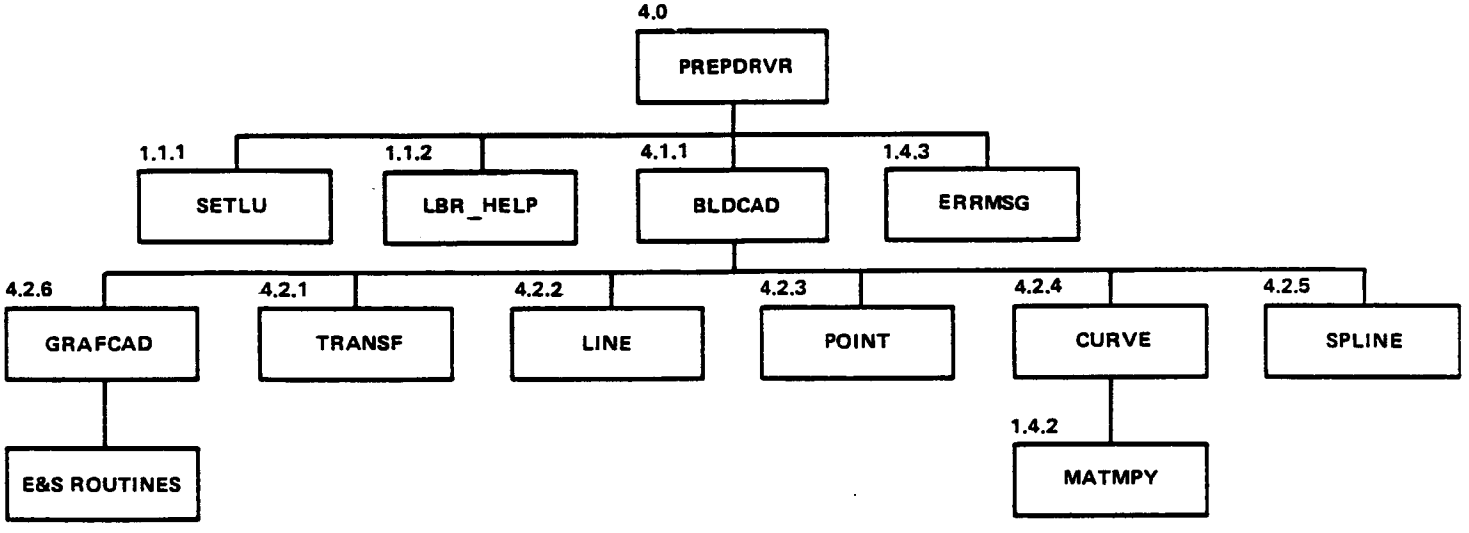


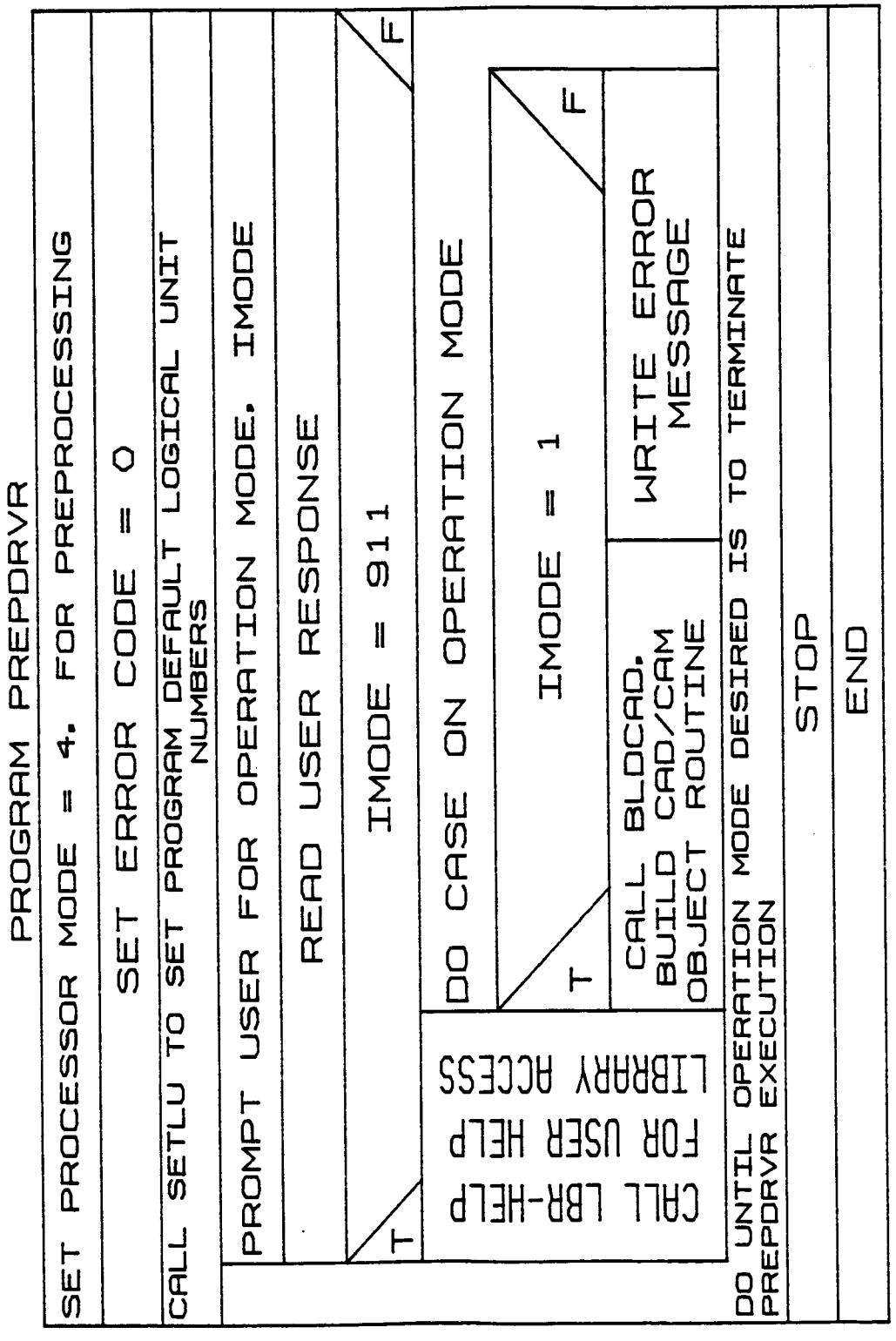
Figure B-10. Functional Block Diagram for PREPDRVR

TABLE B-VIII.-PROGRAMS EMPLOYED IN PREPDRVR

4.0	PREPDRVR
4.1.1	BLDCAD
4.2.1	TRANSF
4.2.2	LINE
4.2.3	POINT
4.2.4	CURVE
4.2.5	SPLINE
4.2.6	GRAFCAD

4.0 PREPDRVR

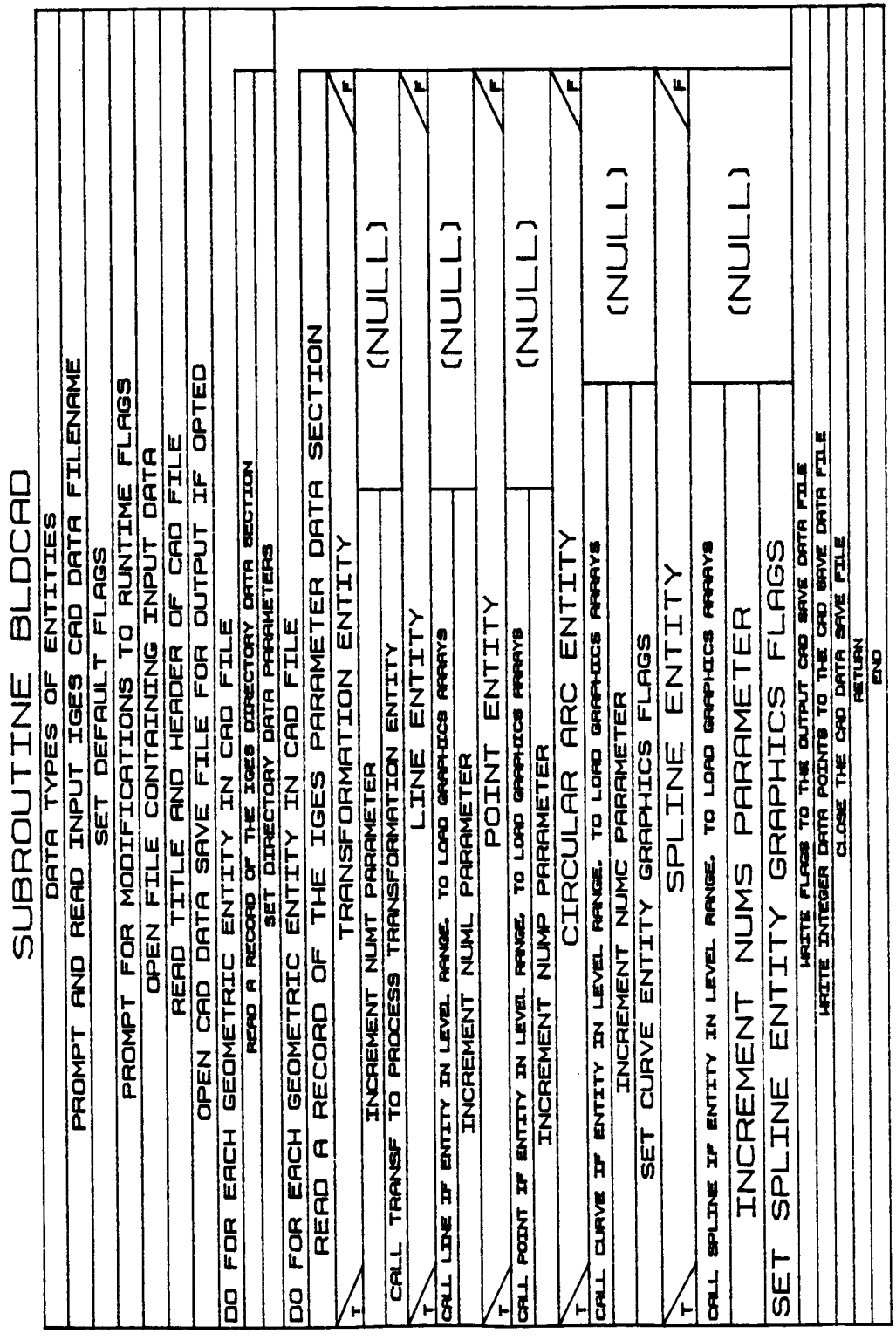
The program PREPDRVR is the Preprocessor function driver. It operates in an interactive mode, prompting the user for the preprocessor option desired. Valid options are, currently: create or modify a CAD/CAM object file or terminate PREPDRVR program execution.


PROGRAM PREPDRVR
SET PROCESSOR MODE = 4. FOR PREPROCESSING
SET ERROR CODE = 0
CALL SETLU TO SET PROGRAM DEFAULT LOGICAL UNIT NUMBERS
PROMPT USER FOR OPERATION MODE. IMODE
READ USER RESPONSE
IMODE = 911
T
F
CALL LBR-HELP FOR USER HELP LIBRARY ACCESS
DO CASE ON OPERATION MODE
IMODE = 1
T
F
CALL BLDCAD. BUILD CAD/CAM OBJECT ROUTINE
WRITE ERROR MESSAGE
DO UNTIL OPERATION MODE DESIRED IS TO TERMINATE PREPDRVR EXECUTION
STOP
END

4.1.1 BLDCAD

BLDCAD is called in the preprocessor driver. It reads a CAD/CAM initial graphics exchange specification (IGES) - formatted file, calls the appropriate entity routine to fill the real number and integer data arrays (for graphics routine interaction), and if graphics display is opted by the user, calls a graphics routine to display the entities or an Evans and Sutherland device. The file of IGES data may be saved for input during the system definition function detailed graphics generation for arms, loads, environment or targets. The format read from the IGES database follows the documentation for version 2.0 published by the National Bureau of Standards.

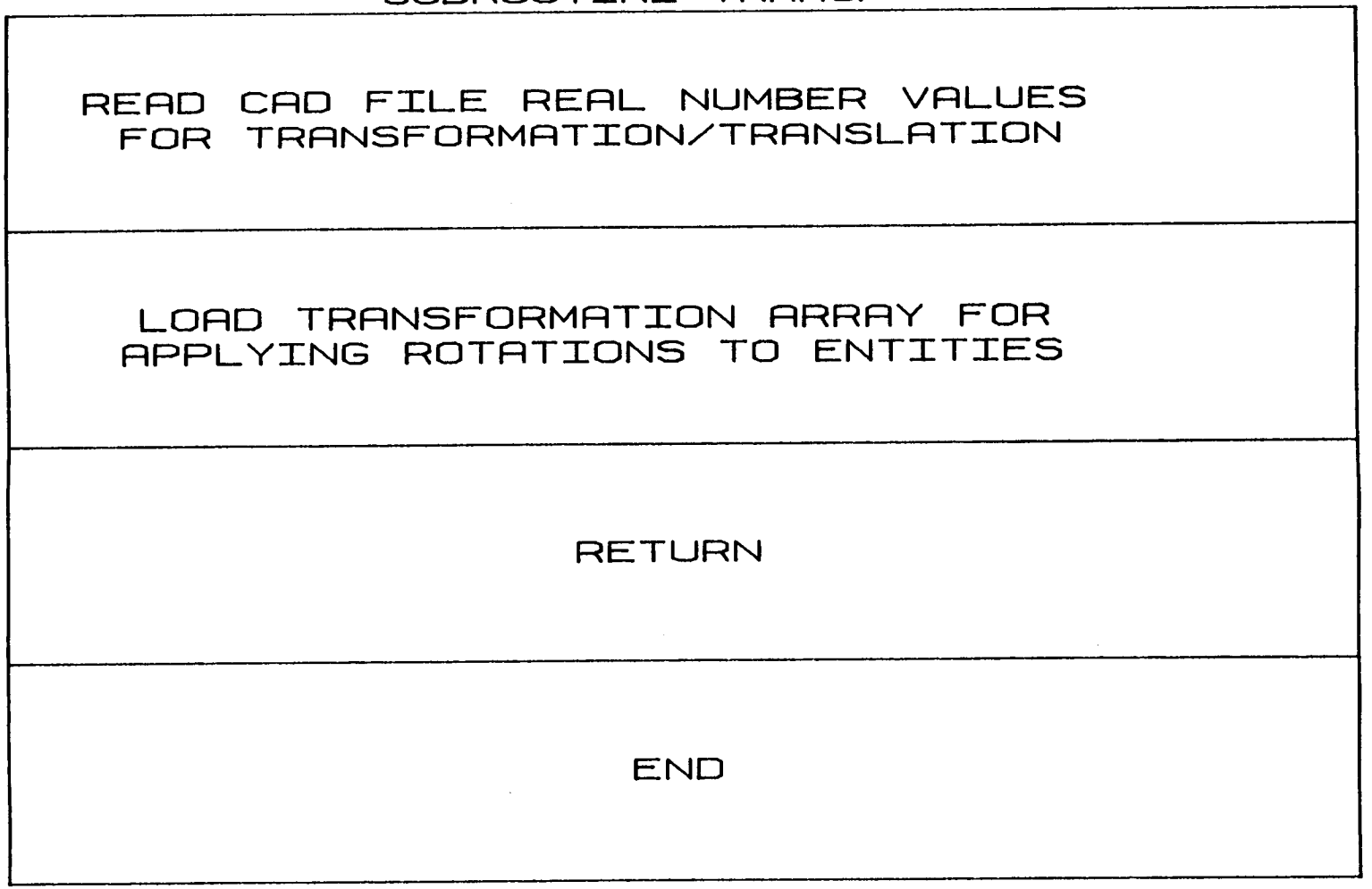
ORIGINAL PAGE IS
OF POOR QUALITY



4.2.1 TRANSF

TRANSF CAD/CAM IGES read entity-associated transformation data and loads the transformation array for the rotations and translations to be applied to the entity before graphics display.

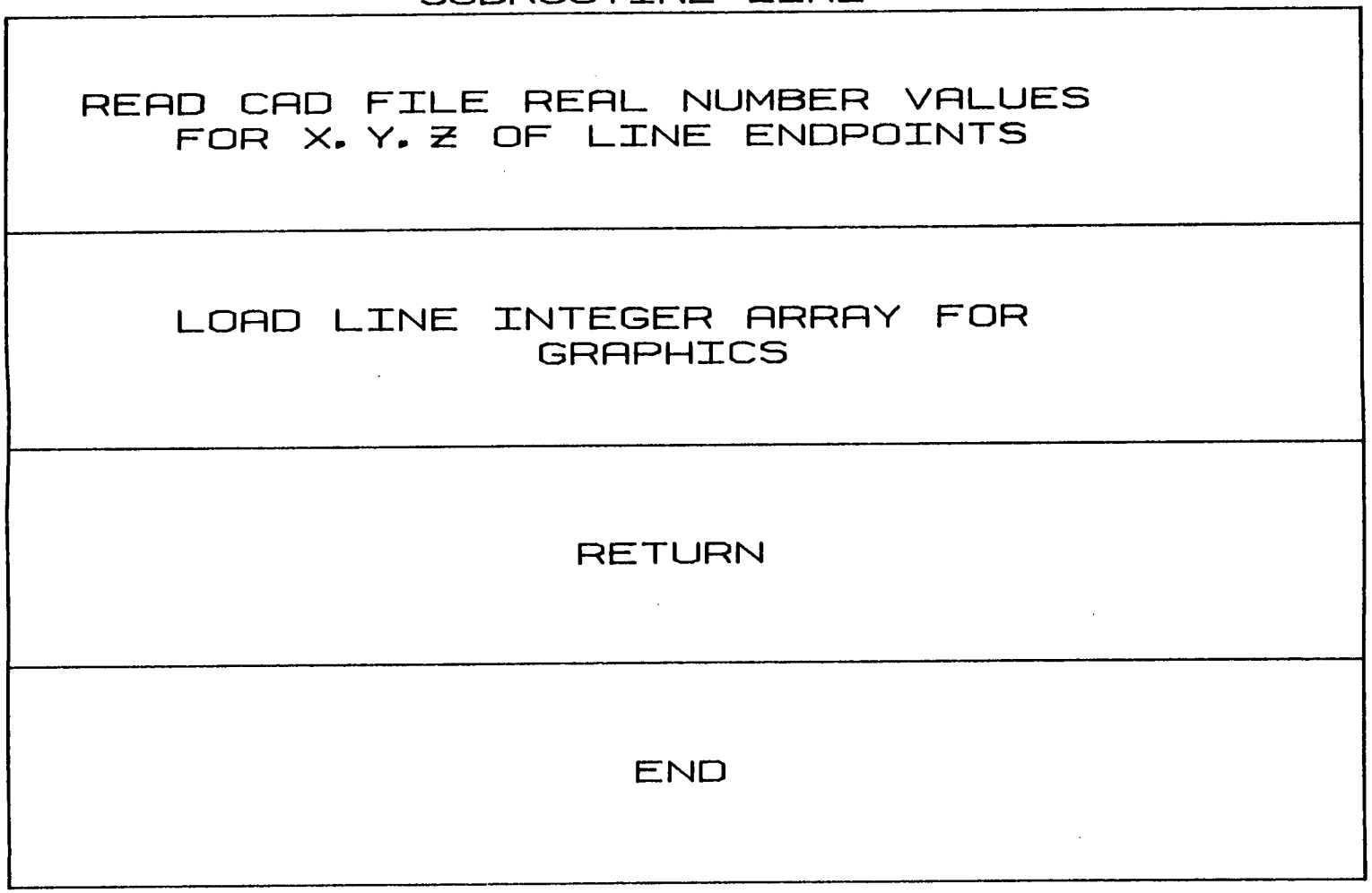
SUBROUTINE TRANSF



4.2.2 LINE

LINE reads CAD/CAM IGES line endpoint entity data and loads the line array for the graphics display.

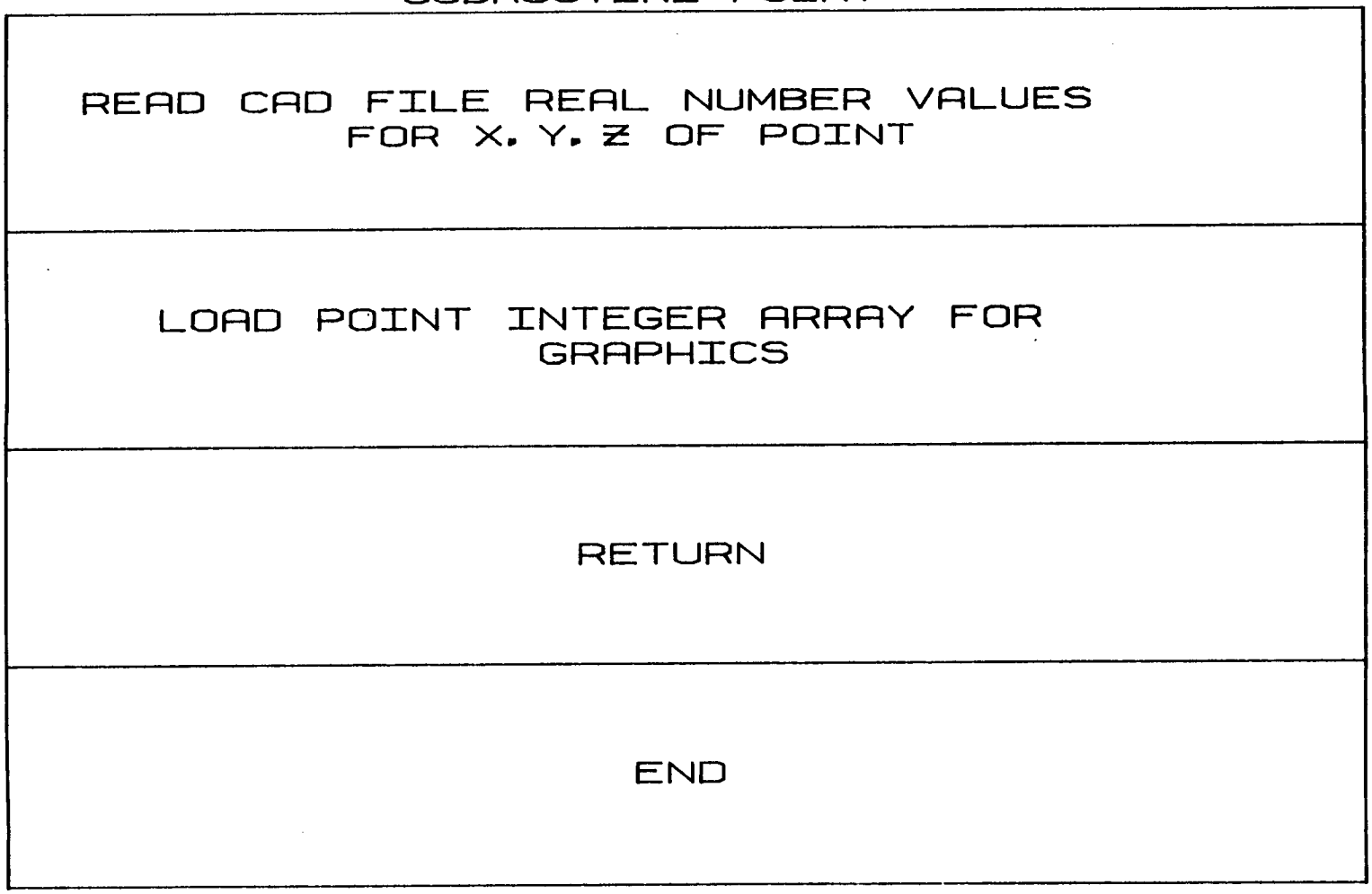
SUBROUTINE LINE



4.2.3 POINT

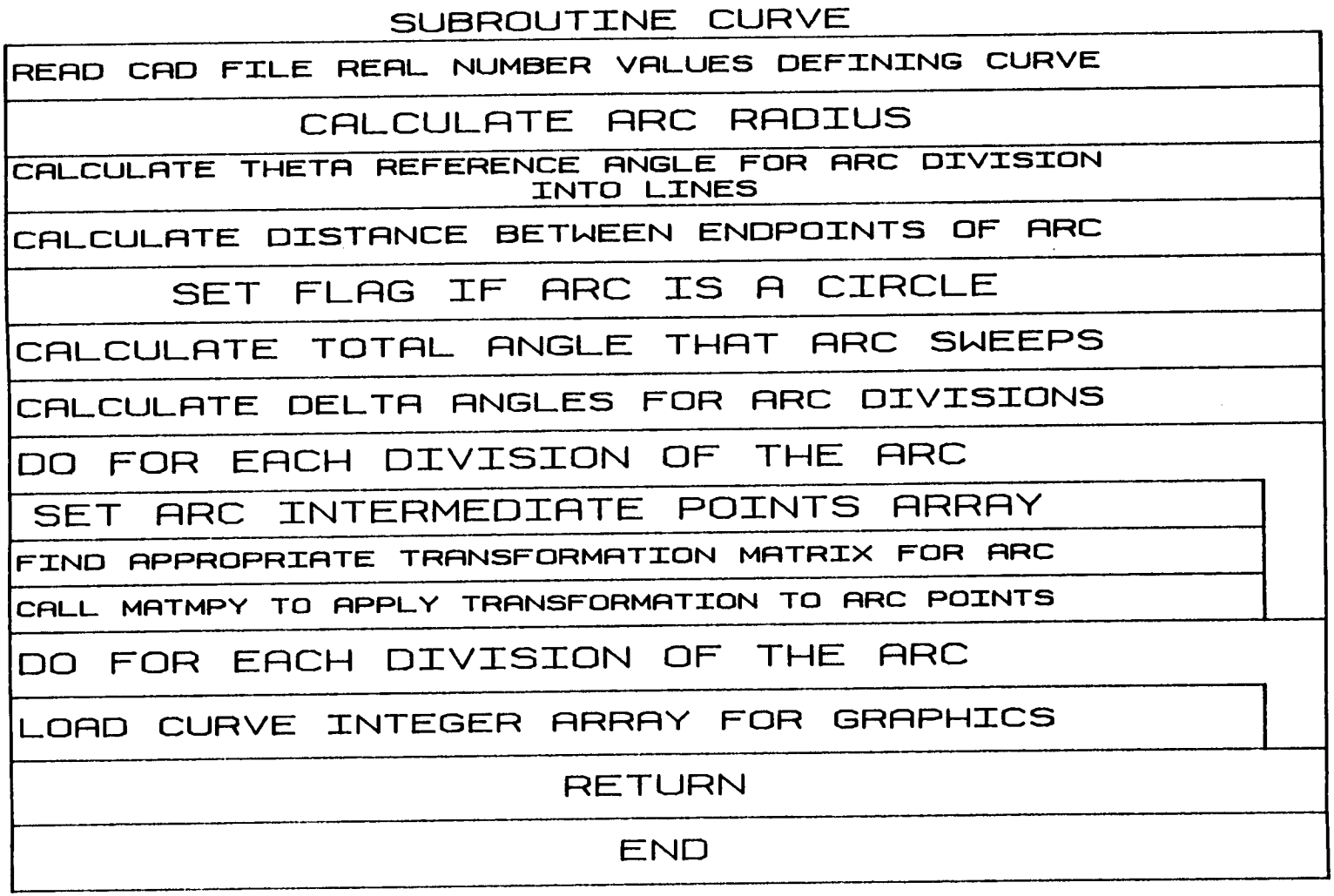
POINT reads CAD/CAM IGES point entity data and loads the points array for the graphics display.

SUBROUTINE POINT



4.2.4 CURVE

CURVE reads CAD/CAM IGES circular arc data and loads the arc points array for the graphics display.


SUBROUTINE CURVE
READ CAD FILE REAL NUMBER VALUES DEFINING CURVE
CALCULATE ARC RADIUS
CALCULATE THETA REFERENCE ANGLE FOR ARC DIVISION INTO LINES
CALCULATE DISTANCE BETWEEN ENDPOINTS OF ARC
SET FLAG IF ARC IS A CIRCLE
CALCULATE TOTAL ANGLE THAT ARC SWEEPS
CALCULATE DELTA ANGLES FOR ARC DIVISIONS
DO FOR EACH DIVISION OF THE ARC
SET ARC INTERMEDIATE POINTS ARRAY
FIND APPROPRIATE TRANSFORMATION MATRIX FOR ARC
CALL MATMPY TO APPLY TRANSFORMATION TO ARC POINTS
DO FOR EACH DIVISION OF THE ARC
LOAD CURVE INTEGER ARRAY FOR GRAPHICS
RETURN
END

4.2.5 SPLINE

(not implemented yet)

4.2.6 GRAFCAD

GRAFCAD displays the CAD/CAM IGES entity data on an Evans and Sutherland graphics device.