SGP Memo No. 25

National Radio Astronomy Observatory
Very Large Array

1st February 1983

To: Addressee

From: Bob Duquet

Subject: The Operational FORTRAN Standard Commands Package

Since approximately mid-September, I have been working on a FORTRAN Standard Commands Package. The first two months were spent on a feasability study and on preparation of a project proposal. Both were described in a memo and attached document dated November 15th. The general reaction to the proposal being at least non-destructive, I proceded to write a full set of standard command routines, first for the VAX then for the PDP-11's. The package is now complete on both machines except for the usual late-stage debugging which waits upon sufficient applications to identify problem areas.

The final package differs in several respects from the one proposed last November. Almost all of the differences are internal. They should be invisible to the average user. The biggest differences are those that were required by the limitations of PDP11 hardware (only 64K bytes of address space) and software (only the subset version of FORTRAN 77 is available). For example, each CHARACTER variable in each subroutine calling sequence had to be replaced by two arguments: the original plus an additional INTEGER variable to carry the length of the CHARACTER string. Some reorganization of the package was necessary to accomodate the elaborate overlay structure forced on the package by the previously mentioned PDP-11 limitations. A few changes were required when it was discovered that one aspect of the SAIL Standard Commands Package (the "options" feature) had been overlooked. Finally, in deference to those who feel that command "grouping" is indispensible, the package was revised to include that rather redundant feature.

The attached document is intended as an overview of the operational FORTRAN Standard Commands Package as it exists now. Each individual subprogram (there are roughly 75) contains "self-documentation" comments which will be extracted by the DOC program, copied, and distributed to the PDP-11 System Library Manuals (the green loose-leaf notebooks).

All material relating to the standard command package (source code, control files, overlay description files, documentation, etc.) is in the NEW area, [220,42], on the SORTER and MAPPER machines. All REL modules can be linked from the library [220,42]SCLIBR.OLB. Presumably, everything will be moved to the PDP-11 system area, [210,42], at some future date. The VAX version is in VAX3::[DUQUET.SC].

The FORTRAN package should be available on the DEC-10 shortly after DEC actually releases the FORTRAN 77 compiler they announced with great fanfare last November. (Delivery is now promised for May or June.) The package has already been tested on the DEC-20 at New Mexico Tech.

The package has been used in Bob Payne's program BTLST on the PDP-11's and in Don Well's program TSTAR on the AIPS machine. It has also been incorporated into the DBFILL pipeline program. Problems with the package should be brought to my attention as soon as possible.

THE OPERATIONAL FORTRAN STANDARD COMMANDS PACKAGE

Robert T. Duquet

February 1983

The FORTRAN Standard Commands Package (FSCP) is a set of approximately seventy five independently compiled subprograms whose purpose is to supply a standard user interface for all VLA application programs. This package should appear to the user nearly identical to the SAIL Standard Commands Package which has been in service on the VLA DEC-10 for the past four or five years.

The FSCP consists of three sets of subprograms. The first set (roughly a dozen) provides the superstructure. The second set consists of 37 routines, each of which impliments one of the standard commands that have been defined so far. (The commands are listed below). The remaining set of 20 "utility" routines provides services commonly required in the other two sets.

A programmer who wishes to use only the defined standard commands needs to be aware of only three of the FSCP routines: one to initialize the package, one to call the package, and the third to obtain parameter values. These notes include both a "cookbook" describing how those three routines are called and a sample program illustrating how it should be done.

The procedure for defining special purpose commands is only slightly more elaborate. A subroutine defining the standard command must be written to have a fixed, pre-defined, calling sequence. One other routine, which must be called SCXTRA, must be written to invoke these special commands. The cookbook and the sample program include the definition of special commands.

User-written special commands must include in their routine the source-code module SCCOMMON.DCL. This defines certain critical parameters and three crucial COMMON areas. The DCL file is in the same location as the rest of the package on each machine.

The structure of the application program that uses the FSCP may be quite arbitrary (except, of course, that the initialization routine should be called before any other FSCP routines are called). The package may be called at any time from any point in the application program. The most apparent restriction on this freedom does not arise from the package itself but from the limitations of the PDP-11 which force an elaborate overlay structure upon any application program that uses the FSCP.

On the PDP-11, the MAIN program of an application must be as small as possible and it should consist of an unending loop that calls the FSCP, acquires and preserves the latest parameter values, calls a subroutine to perform the application task (on the basis of the latest parameters), then recycles to the FSCP. All calls to FGCP routines should be done from this small MAIN program. The application subroutine should NOT call the package directly because, almost invariably, it will overlay the FSCP routines. (See the overlay schematic at the end of this document.)

The operational Standard Commands Package includes the one-line editor discussed in the design document. The editor written for the test version has been modified so that it can be used from ANY type of terminal. The "greater than" and "less than" keys are used to move the cursor right or left i.e. in the direction appropriate to those symbols when they are viewed as arrowheads. The DEL character deletes the symbol under the cursor; editing is terminated by a Carriage return; all other symbols are inserted in the line at the cursor position. All escape sequences, other than those generated by the right and left arrows on the Visual 100 terminal, are ignored.

Each Standard Command is placed in one of the two standard groups (DATASELECT or MISC) defined for commands on the DEC-10. By default, special commands are placed in a group that has the name of the MAIN program in which they are used. Special commands can be placed in any arbitrarily named group by returning that name when the command is initialized (see the sample program).

The following are (or should be) machine-dependant routines:

SCOPEN	Uses system services to obtain user and terminal identification which become part of a file-name.
SCEDX	Uses system services to do direct I/O from the terminal for editing. (Traps control sequences, etc)
L2C	This utility converts a character from LOGICAL*1 to CHARACTER*1 form on the PDP-11. It is a NULL routine on machines with full FORTRAN 77.
SCDBNA	This is the Standard Command DATABASE. It tries to follow our own VLA inconsistancies in naming database files. (For example, with or without leading zeroes for small user-numbers.)
SCCAT	This is the Standard Command CATALOGUE-ID. Same comments as above.
SCINF SCINFS SCOUTF SCOUTS	These 4 commands for INFILE(S) and OUTFILE(S) try to use defaults that match the filename format on the machine in which they are used. For example, the [300,20] UIC vs the [DUQUET.SC] directory.

STANDARD COMMANDS DEFINED IN FSCP

AMP/JY

ANTENNA

ANTENNAS

AVERAGE

BAND

BANDS

CALCODES

CATALOGUE-ID

CHANNEL

CHANNELS

CALIBRATION

DATATYPE

DBNAME

DCSADDRESS

DCSADDRESSES

GTINTERVAL

IF

IFPAIR

IFS

IFPAIRS

INFILE

INFILES

LISTOPTION

LISTOPTIONS

LSQTOLERANCE

MODE

MODES

OUTFILE

OUTFILES

PASSFLAG

POSITION

REFANT

REFANTS

SOURCE

SOURCES

STOKES

TIMERANGE

VISIBILITYTYPE

COOKBOOK FOR USING THE FSCP

(See the next section for subroutine descriptions)

1- Write a MAIN program which contains the following:

A) A call to the FSCP initialization routine SCINIT.

B) An infinite loop consisting of:

a) A CALL to the FSCP routine SCPKG.

b) A Block IF statement with 4 alternatives based on the content of the CHARACTER variable passed to SCPKG.

1) If it is 'HELP' write the HELP text for GO.
2) If it is 'EXPL' write the program's EXPLAIN text.
3) If it is 'EXIT' terminate the program.
4) If it is anything else, it must be the GO argument.
   i) Call the subroutine described in step 2.
   ii) Call the subroutine described in step 3.

c) A statement returning control to the top of the loop.

2- Write a subroutine to be called at the begining of the GO phase. This routine should:

A) Call the parameter-fetching routine, SCPARM, once for each command used by the program, in order to get the latest values.

B) Store the latest parameter values in a location accessible to the routine described in step 3.

3- Write a subroutine to accomplish the application task. Return to the command scanner by executing a RETURN statement. N.B. Steps 2 and 3 can be combined if they will not overlay the the FSCP routines.

4- Write a taskbuilding command file that uses an overlay structure. The overlay structure should be described in terms of the "canned" Standard Commands overlay description contained in [220,42]SCODL.ODL

COOKBOOK (Continued)

The following steps are needed only when special commands must be defined

5- For each command write a subroutine that uses the same calling sequence as SCPARM except for the first argument which is omitted. The calling sequence for SCPARM is described in the next section).

6- Write a version of SCXTRA which calls the routines described in step 5 above. The calling sequence is identical to that of SCPARM. The first argument is the name of the command whose implimenting routine is to be called.

The following steps are needed only when memory limitations require that unused commands be eliminated from the package.

7- Copy the routine SCPARM.FTN from [220,42].

8- Remove the pair of lines that refers to each unused command.

9- Copy the overlay description module SCODL.ODL from [220.42]

10-Remove references to unused command routines.
(See the listing and schematic of the FSCP overlay shown later.)

PROGRAM INTERFACE TO THE FSCP

The FSCP must be initialized by a call to the routine SCINIT. This should precede all other calls to the package except, possibly, those calls that define special default values.

```
CALL SCINIT (HISNAM,L1,HISVER,TEXT,L2,USING,L3.LA,LB,LC,LD)
```

The arguments are:

HISNAM	The name of this application program.
L1	The length of the preceeding string.
HISVER	The version number of this application program.
TEXT	A message (or salutation) to the user.
L2	The length of the preceeding string.
USING	A list of commands (with OPTIONS) to be used.
L3	The length of the preceeding string.
LA,,,,LD	Logical unit numbers for I/O. If any is zero a default will be supplied.

A call to the routine SCPKG is the means by which any application program should turn control over to the Standard Commands Package.

```
CALL SCPKG (ARG, LX)
```

The arguments are:

ARG A CHARACTER variable (supplied by the calling program) in which the command scanner will return requests for services from the GO phase (HELP, EXPLAIN, or EXIT). This string will also be used to relay to the caller whatever text followed the word GO in the user's GO command.

LX The length of the previous arument (supplied by caller).

PROGRAM INTERFACE (Continued)

The SCPARM routine is the "master" command routine for all of the standard commands in the package. It fans out the call for service to the proper command-defining routine.

The program being served by the FSCP may include a modified copy this routine as part of its own definition. This might be desirable in two circumstances:

1- The user wants to intercept communication to or from a standard command in order to impose certain restrictions or alterations upon the normal command. (This routine therefore provides what is known as a "hook".)

2- References to unused commands can be removed thereby paring the memory requirements of the FSCP.

If used in unmodified form, this routine will incorporate in the application program ALL of the defined standard commands (whether they are used or not).

CALL SCPARM (NAM, REQUES, INSTR, L1, OUTSTR, L2, VALUE)

The arguments are:

NAM The name of the command. This must be fully spelled out (no minmatch) in a CHARACTER*14 variable. This is the only argument used by this routine. All of the others are simply passed to the appropriate subroutine representing the command selected.

REQUES An INTEGER that specifies the service desired. The set of valid integers and their meaning are described later.

INSTR A CHARACTER variable (supplied by the caller) used for input to the command. Not used when REQUES = 6.

L1 The length of the preceeding argument.

OUTSTR A CHARACTER variable (supplied by the caller) used by the command to return parameters to the caller. The requisite length of this argument to ensure that all data can be accomodated is command-specific.

L2 The length of the preceeding argument.

VALUE A REAL variable (or array) used by the command to return parameter values to the caller. Whether this is a scalar or an array, and the requisite dimensions of the array in the latter case, are command-specific.

PROGRAM INTERFACE (Continued)

The subroutine for every special command must be defined in terms of a pre-defined set of formal parameters (a fixed calling sequence).

CALL SPECL (REQUES, INSTR, L1, OUTSTR, L2, VALUE)

The arguments are the same as those described for SCPARM above. The meaning of the numeric code used in the first argument is described in the comments shown in the sample program.

The FSCP will call the routine SCXTRA for service expected from all commands that are NOT part of the defined Standard set. The FSCP contains a "Stub" for this routine in order to satisfy linkage requirements whenever no special commands are defined.

CALL SCXTRA (NAM, REQUES, INSTR, L1, OUTSTR, L2, VALUE)

The arguments for this routine are identical to those for SCPARM.

SAMPLE PROGRAM USING THE FSCP

(Main Program)

```
      PROGRAM TSTBED

C     ------------------------------------------------------------
C               This program illustrates use of the
C                FORTRAN Standard Commands Scanner.
C
C     The work to be done by the program is represented by the
C     routine GIANT. The values of parameters needed by GIANT are
C     placed in COMMON by the routine PARAMS. (They cannot be obtained
C     directly because GIANT presumably overlays the command scanner
C     and all of its routines.)
C
C     There is NO significance to the particular combination of standard
C     commands included in this illustration. The "special" commands,
C     MY-EXTRA-ONE and MY-SECOND, are included in order to demonstrate
C     how such commands are written.
C
C     NOTE that the list of commands must:
C     1- Be in one CHARACTER variable, not an array of variables.
C     2- Name "special commands in fully-spelled out form.
C     3- Name standard commands up to the point of un-ambiguity.
C     4- State options by listing the command in fully spelled out form
C           followed by a period followed by the option name.
C     ------------------------------------------------------------


      CHARACTER*80   TEXT
      DATA TEXT /'  This demonstrates the FORTRAN Standard Command Package'/

      REAL*4         VERSN
      DATA  VERSN    /1.1/

      CHARACTER      USE*500
      DATA USE /'SOURCES, ANTENNAS.WITH, POS, IF, VIS,
     1              MY-EXTRA-ONE, MY-SECOND'/

      LOGICAL*1   GOOD
      CHARACTER   TEMP*16


C     This MUST be the initial call to the command scanner

      CALL SCINIT('TSTBED',6,VERSN,TEXT,80,USE,500,0,0,0,0)
```

SAMPLE PROGRAM USING THE FSCP

(Main Program - Continued)

```
C    Now we enter a loop that starts by calling the scanner for service,

11111     CALL SCPKG(USE,200)


C    The scanner may be asking us for the 'HELP' text for GO.
C    If so, supply it then go right back.

     IF (USE(1:4) .EQ. 'HELP') THEN
         WRITE (5,100)
100      FORMAT('0 GO',
     1             T20,'This is the HELP text from the GO subroutine')


C    Alternatively, the scanner may ask us for the EXPLAIN text.

     ELSE IF (USE(1:4) .EQ. 'EXPL') THEN
         WRITE (5,200)
200      FORMAT('  Here is where the ''EXPLAIN'' text should be')


C    The scanner may have received a STOP command.

     ELSE IF (USE(1:4) .EQ. 'EXIT') THEN
         WRITE (5,300)
300      FORMAT('  Normal termination of TSTBED program')
         CALL EXIT


C    All other possibilities mean that we should get to work
C    with whatever the program does. First, however, we get
C    the necessary parameter values and leave them in COMMON.
C    When we call the worker routine GIANT we pass along the
C    text recieved as part of the GO command.

     ELSE
         CALL PARAMS
         CALL FNDLAS(USE,100,LX)
         CALL GIANT(USE,LX)

     END IF

C    After completing our work we go back to the scanner for
C    further instructions.

     GO TO 11111
     END
```

SAMPLE PROGRAM USING THE FSCP

(Acquiring Parameter Values from the FSCP)

```
      SUBROUTINE PARAMS

C     ------------------------------------------------------------------
C     Poll each of the commands that the task listed in order to obtain
C     the latest values of parameters. Leave those values in COMMON
C     where the worker routine GIANT can find them.
C
C     Note that first argument to SCPARM must contain the command name
C     in fully-spelled-out form (no minmatch). Furthermore, the name
C     must be in a CHARACTER variable or constant of length 14.

C     ------------------------------------------------------------------

      CHARACTER         DUM*1, IFID*1, SORCES*250, MY1*80, MY2*80
      REAL              ANTENN(-1:58), QUAL(0:25), CHCODE
      DOUBLE PRECISION  POSITN(2)

      COMMON /PARMS1/ ANTENN, QUAL, CHCODE
      COMMON /PARMS2/ IFID, SORCES, MY1, MY2

      CALL SCPARM('ANTENNAS      ', 6, DUM, 1, DUM,      1, ANTENN)
      CALL SCPARM('SOURCES       ', 6, DUM, 1, SORCES, 250, QUAL)
      CALL SCPARM('POSITION      ', 6, DUM, 1, DUM,      1, POSITN)
      CALL SCPARM('IF            ', 6, DUM, 1, IFID,     1, DUMMY)
      CALL SCPARM('VISIBILITYTYPE', 6, DUM, 1, DUM,      1, CHCODE)
      CALL SCPARM('MY-EXTRA-ONE  ', 6, DUM, 1, MY1,     80, DUMMY)
      CALL SCPARM('MY-SECOND     ', 6, DUM, 1, MY2,     80, DUMMY)

      RETURN
      END
```

SAMPLE PROGRAM USING THE FSCP

(Redefining SCXTRA for Special Commands)

```
      SUBROUTINE SCXTRA (NAME,REQUES,INSTR,L2,OUTSTR,L3,VALUE)

C     ------------------------------------------------------------------
C     This is an example of a "master" routine for user-defined commands.
C     It is called by the standard package whenever any user-defined
C     command is recognized in the input stream.
C     This routine need NOT be supplied in cases where no special
C     commands have been defined. For such cases a dummy routine
C     of the same name (a "stub") has been included in the
C     Standard Commands library.
C
C     Since the first argument is the name of the command that has been
C     encountered, only one master routine is required no matter how
C     many special commands have been defined. (Note that the name being
C     passed will always be fully expanded and in upper case.)
C     Funneling all references to user-defined commands through one
C     conventionally-named "master" routine is the only way that one
C     default routine can satisfy linkage requirements.
C
C     All but the first argument are meant to be passed along to the
C     command-defining routine. They are described elsewhere.
C     Of course the programmer always has the option of combining
C     the command master and the code for the actual commands.
C     This would be especially appropriate if only one special command
C     is being defined.
C     ------------------------------------------------------------------

      INTEGER       REQUES
      CHARACTER*14  NAME
      CHARACTER*(*) INSTR, OUTSTR
      REAL          VALUE


      IF (NAME .EQ. 'MY-EXTRA-ONE') THEN
          CALL MYOWN1 (REQUES, INSTR, L2, OUTSTR, L3, VALUE)

      ELSE IF (NAME .EQ. 'MY-SECOND') THEN
          CALL MYOWN2 (REQUES, INSTR, L2, OUTSTR, L3, VALUE)

      ELSE
          WRITE (5,100) NAME
100       FORMAT('  I don''t recall defining something called ',A)
      END IF

      RETURN
      END
```

SAMPLE PROGRAM USING THE FSCP

(Writing a Special Command)

```
      SUBROUTINE MYOWN1 (REQUES, INSTR, L1, OUTSTR, L2, VALUE)

C     ------------------------------------------------------------
C     This is a sample of a user-defined command.
C
C     Here is the numeric code by which the first argument designates
C     the service requested by the caller:
C
C  Neg- The caller is specifying (in INSTR) the defaults that should
C       be used for this command for this program. (Not very useful in
C       special commands since the "built-in" default can be tailored
C       to the program for which the special command was created.)
C
C  0-   The caller wants to have all parameters defined in this
C       command set to their default value(s). If the second argument
C       is not blank, the caller is simultaneously specifying one or
C       more OPTIONs for this command. The routine may return the
C       name of the group to which it belongs.
C
C  1-   The caller wants to have all parameters defined in this
C       command set to their default value(s).
C
C  2-   The FSCP has read a reference to this command in the input
C       stream. The routine should do whatever it is supposed to do
C       whenever the user types this command (usually giving a new
C       value to one or more parameters). The second argument is the
C       string that followed the command name on the input line.
C
C  3-   The INPUTS command has been encountered in the input stream.
C       The caller therefore wants (in the third argument) the text
C       that should be displayed in response. The caller will take
C       care of doing the actual display.
C
C  4-   Caller wants the string that is to be SAVE'ed. (Redundant;
C       this and the previous function should always be identical.)
C
C  5-   Caller wants to have the HELP text output on LUNTTO.
C
C  6-   Caller wants the value of one or more parameters. They should
C       be returned in the subsequent arguments. The specific values
C       to be returned are command-dependent. For example, the last
C       argument could be either the address of a scalar or the
C       starting address of an array of real values. The next-to-last
C       argument can be used to return character values.  Some complex
C       commands may require that the second argument identify which
C       among several possible responses the called wants.

C     ------------------------------------------------------------
```

SAMPLE PROGRAM USING THE FSCP

(Writing a Special Command - Continued)

```
C     The single parameter involved in this simple example is a character
C     string which contains an arbitrary text.


      INCLUDE 'SCCOMMON.DCL/NOLIST'

      INTEGER        REQUES
      CHARACTER*(*)  INSTR, OUTSTR
      REAL           VALUE
      CHARACTER*60   MYVAL

      SAVE MYVAL

C     No extraordinary default values are expected.
C     The command group is named as "Special".

      IF (REQUES .LE. 1) THEN
          MYVAL='MY first command has been set to its DEFAULT value'
          OUTSTR(1:L2)='SPECIAL'

C     On finding this command in the input stream just take whatever
C     follows the command name as the parameter value.

      ELSE IF (REQUES .EQ. 2) THEN
          MYVAL=INSTR(1:L1)

C     Return the argument as the string to be displayed by the caller
C     in case of INPUT command or SAVE'ed in case of that command.

      ELSE IF ((REQUES .EQ. 3) .OR. (REQUES .EQ. 4)) THEN
          OUTSTR(1:L2)=MYVAL

C     The routine does do its own displaying of the HELP text.

      ELSE IF (REQUES .EQ. 5) THEN
          WRITE (LUNTTO,200)
200       FORMAT('0 MY-OWN-EXTRA',T20,'This is the HELP text')

C     A request for parameter values gets the string in OUTSTR.
C     The last argument (VALUE) is not used in this command.

      ELSE IF (REQUES .EQ. 6) THEN
          OUTSTR(1:L2)=MYVAL

      END IF
      RETURN
      END
```

SAMPLE PROGRAM USING THE FSCP

(Another Special Command)

```
      SUBROUTINE MYOWN2 (REQUES, INSTR, L1, OUTSTR, L2, VALUE)

C     ------------------------------------------------------------
C     This is a second example of a user-defined command.
C
C     It is really just a copy of the first one except that in this
C     case we do not bother giving the command a group name. The FSCP
C     will assign it to a group with the name of the MAIN program.
C     ------------------------------------------------------------

      INCLUDE 'SCCOMMON.DCL/NOLIST'

      INTEGER       REQUES
      CHARACTER*(*) INSTR, OUTSTR
      REAL          VALUE
      CHARACTER*60  MYVAL

      SAVE MYVAL

      IF (REQUES .LE. 1) THEN
          MYVAL='MY second command has been set to its DEFAULT value'

      ELSE IF (REQUES .EQ. 2) THEN
          MYVAL=INSTR(1:L1)

      ELSE IF ((REQUES .EQ. 3) .OR. (REQUES .EQ. 4)) THEN
          OUTSTR(1:L2)=MYVAL

      ELSE IF (REQUES .EQ. 5) THEN
          WRITE (LUNTTO,200)
200       FORMAT('0 MY-SECOND',T20,'This is the second HELP text')

      ELSE IF (REQUES .EQ. 6) THEN
          OUTSTR(1:L2)=MYVAL

      END IF
      RETURN
      END
```

SAMPLE PROGRAM USING THE FSCP

(The Application Task as a Subroutine)

```
      SUBROUTINE GIANT(STRING,LEN)

C     ---------------------------------------------------------------
C     This is a dummy subroutine meant to represent the main body of
C     a large program that uses the FSCP which it overlays.
C     The values of all parameters are available from COMMON where
C     they were placed by the subroutine PARAMS.
C     ---------------------------------------------------------------

      CHARACTER          DUM*1, IFID*1, SORCES*250, MY1*80, MY2*80
      REAL               ANTENN(-1:58), QUAL(0:25), CHCODE
      DOUBLE PRECISION   POSITN(2)

      COMMON /PARMS1/ ANTENN, QUAL, CHCODE
      COMMON /PARMS2/ IFID, SORCES, MY1, MY2

      DIMENSION DUMMY(8000)


      WRITE (5,100)
100   FORMAT('  Here we are in the working part of a large, overlayed',
     1     ' program.')


      RETURN
      END
```

PDP-11 OVERLAY FOR STANDARD COMMANDS

(Refer to schematic diagram which follows)

The sample overlay description shown below is appropriate for the simplest case wherein the standard Commands package can be used "as is". It involves no special commands and no reduction in the set of commands loaded automatically by the system. The main program, BTLST, is a small routine that calls SCINIT, SCPKG and SCPARM. The actual work of the application is done in the routine BTBODY which overlays the standard command routines. References to the factors SC1, SC2, SC3 and SCX are satisfied by the overlay description in [220,42]SCODL.ODL.

```
        .ROOT   BTLST-*(SC1-SC2-*(SC3,SCX),BTB)
BTB:    .FCTR   BTBODY-[210,10]VLAFTN/LB-[210,10]VLAMAC/LB
@[220,42]SCODL.ODL
        .END
```

The next sample overlay description (below) is almost as simple as the previous example. It differs only by having some special commands and, as a consequence, a user-written version of SCXTRA. All of the special comands and the expanded version of SCXTRA are in the user-written module MYOWN which replaces SCX in the example above. The application task is in the routine called GIANT which does not require a factor since it is self-contained (or its library requirements are in the root).

```
        .ROOT   TSTBED-*(SC1-SC2-*(SC3,MINE),GIANT)
MINE:   .FCTR   MYOWN-SCLB
@[220,42]SCODL.ODL
        .END
```

For the third example we have a situation that required elimination of the unused standard commands hence the redefinition of SCPARM. The overlay description that had been refered to indirectly in the previous two examples has been copied into this ODL module. All reference to the superflous command modules were then deleted.

```
        .ROOT   CATLST-*(SC1-SCPARM-*(SC3,XXX),CATB)
SC1:    .FCTR   [220,42]sclibr/LB:SC:SCPKG:SCINIT:SCHELP-SCLB
SC3:    .FCTR   SC3A,SC3B
SC3A:   .FCTR   [220,42]sclibr/LB:SCCAT-SCLB
SC3B:   .FCTR   SCV1-SCV2-SCLB
SCV1:   .FCTR   [220,42]sclibr/LB:SCEDX:SCSRT
SCV2:   .FCTR   [220,42]sclibr/LB:SCOPEN:SCPARS:SCGETC:SCINPT
SCLB:   .FCTR   [220,42]sclibr/LB
XXX:    .FCTR   SCXTRA-SCREC-SCLB
CATB:   .FCTR   CTBODY-MIO/LB-[210,10]VLAFTN/LB-[210,10]VLAMAC/LB
        .END
```

PDP-11 OVERLAY FOR STANDARD COMMANDS (Continued)

(This is the content of the file [220,42]SCODL.ODL)

```
SC1:    .FCTR   [220,42]sclibr/LB:SC:SCPKG:SCINIT:SCHELP-SCLB
SC2:    .FCTR   [220,42]sclibr/LB:SCPARM                                Note 1
SC3:    .FCTR   SC3A,SC3B,SCOV
SC3A:   .FCTR   SCA1,SCA2,SCB,SCC,SCD,SCE,SCF,SCG,SCH,SCI                *
SC3B:   .FCTR   SCJ,SCK,SCL,SCM,SCN,SCO,SCP,SCQ,SCR                     *
SCA1:   .FCTR   [220,42]sclibr/LB:SCANT-SCLB                            Note 2
SCA2:   .FCTR   [220,42]sclibr/LB:SCANTS-SCLB                           Note 2
SCB:    .FCTR   [220,42]sclibr/LB:SCIF:SCIFS:SCIFP:SCIFPS-SCLB          Note 2
SCC:    .FCTR   [220,42]sclibr/LB:SCREF:SCREFS-SCLB                     Note 2
SCD:    .FCTR   [220,42]sclibr/LB:SCCALS:SCPOS-SCLB                     Note 2
SCE:    .FCTR   [220,42]sclibr/LB:SCSRC:SCSRCS-SCLB                     Note 2
SCF:    .FCTR   [220,42]sclibr/LB:SCDBNA-SCLB                           Note 2
SCG:    .FCTR   [220,42]sclibr/LB:SCPASS:SCMODE:SCMODS-SCLB             Note 2
SCH:    .FCTR   [220,42]sclibr/LB:SCCAT-SCLB                            Note 2
SCI:    .FCTR   [220,42]sclibr/LB:SCTIME:SCTIMX-SCLB                    Note 2
SCJ:    .FCTR   [220,42]sclibr/LB:SCCHAN:SCCHNS:SCVIS-SCLB              Note 2
SCK:    .FCTR   [220,42]sclibr/LB:SCBAND:SCBANS-SCLB                    Note 2
SCL:    .FCTR   [220,42]sclibr/LB:SCAMP:SCAVER-SCLB                     Note 2
SCM:    .FCTR   [220,42]sclibr/LB:SCDCS:SCDCSS-SCLB                     Note 2
SCN:    .FCTR   [220,42]sclibr/LB:SCLIST:SCLISS-SCLB                    Note 2
SCO:    .FCTR   [220,42]sclibr/LB:SCINFI:SCINFS-SCLB                    Note 2
SCP:    .FCTR   [220,42]sclibr/LB:SCOUTF:SCOUTS-SCLB                    Note 2
SCQ:    .FCTR   [220,42]sclibr/LB:SCSTOK:SCLSQ-SCLB                     Note 2
SCR:    .FCTR   [220,42]sclibr/LB:SCCLBR:SCDATA:SCGTIN-SCLB             Note 2
SCOV:   .FCTR   SCV1-SCV2-SCLB
SCV1:   .FCTR   [220,42]sclibr/LB:SCEDX:SCSRT
SCV2:   .FCTR   [220,42]sclibr/LB:SCOPEN:SCPARS:SCGETC:SCINPT
SCX:    .FCTR   [220,42]sclibr/LB:SCXTRA-SCLB                           Note 3
SCLB:   .FCTR   [220,42]sclibr/LB
```

Note 1: The reference to SC2 may be replaced by a reference to a (stripped-down) version of SCPARM suplied by the user who needs to minimize memory requirements.

Note 2: This module can be eliminated if the corresponding command is not being used. In that case, a stripped-down version of SCPARM must be supplied (see Note 1 above). Reference to the ODL FCTR must be removed in the lines flagged with an asterick.

Note 3: The reference to this "stub" for SCXTRA must be replaced by a reference to the user's expanded version when special commands have been defined.

OVERLAY SCHEMATIC