Automating Pro/Engineer Using Trail Files and External Programs

Ken Chow
Lawrence Berkeley National Laboratory
May 21, 1996

Keyboard macros provide shortcuts to many repetitive command sequences in Pro/Engineer[1]. They map any number of frequently used command sequences to user-selected keyboard keys. They may be nested within each other and may also include user keyboard entry within the macro.

Another powerful feature of Pro/Engineer is adding menu options. Menu options may be added to any Pro/Engineer menu and is an effective way to graphically display keyboard macros to make them more accessible. Command sequences are mapped to a single user-defined menu option added to the bottom of any Pro/Engineer window. The "@setbutton" command added to the "menu_def.pro" file specifies the commands to associate with added menu options.

Menu options may also be used to execute non-Pro/Engineer commands. The associated command is linked to a menu option within Pro/Engineer's Utilities menu (in the Misc menu) and is issued whenever the menu option is selected. Such a feature is useful for adding menu options to start the Pro/Engineer User Guide utility, start a text editor from within Pro/Engineer, or perform system level actions. The "@setbutton_exec" command is used in the "menu_def.pro" file for non-Pro/Engineer commands. A more detailed description of adding menu options can be found in the *Pro/Engineer Fundamentals Guide*.

Another useful component of Pro/Engineer is the trail file functionality. Trail files are automatically created every time a new working Pro/Engineer session begins. Although they are typically used to reconstruct a previous working session, they can also be used to automate a series of commands. By specifying all the commands in a trail file, a user can issue the commands quickly and repetitively using the command sequence "Misc-Trail" and specifying the trail file name. All actions, including keyboard entries and mouse click locations, may be included in a trail file.

Although adding menu options and using trial files provide powerful functionality in Pro/Engineer, neither the menu options functionality nor trail files allow interaction between the model and the commands to perform. The commands performed by the menu options are static and cannot depend on model features and parameters. Commands cannot query the model and perform different actions or calculations based on the query results; the menu options commands cannot interact with Pro/Engineer. As an example, suppose we wish to generate a feature listing for all parts and subassemblies in a model. We can add a menu option or a keyboard macro to make the feature listing command easier to execute, but we would still need to select each part and subassembly individually (either through screen selection, selection by menu, or entering the component name). The task becomes quite tedious if we have an assembly with a large number of subassembly and part components.

[1] "Pro/ENGINEER User's Guides Release 16.0", Parametric Technology Corporation, 128 Technology Drive, Waltham, MA, 1995.

A versatile and powerful method for automating many Pro/Engineer tasks is to combine the menu options functionality with Trail files and the information files which Pro/Engineer creates during information listings. The combination provides communication between Pro/Engineer and other programs and enables automation of a large variety of commands.

The automation scheme is comprised of three components:

- the information files (usually with extensions "inf" or "lst")
- a menu option to issue a program or command external to Pro/Engineer
- a menu option to run both the external program and its associated trail file

Figure 1 shows how each of the components are linked together. Each are discussed in the following sections.

The information files

Pro/Engineer writes many information files to the launching directory (the directory from which Pro/Engineer was launched) when information listings are requested. These files are in ASCII text format. By first issuing commands to create an information file, external programs can then be used to read the information files and perform calculations and other actions based on values in the files. For example, we can define a parameter called HEIGHT and issue "Setup-Parameter-Info" to create the "param.inf" file. Any external program or script can then be issued to read "param.inf", extract the HEIGHT value, and perform calculations or actions based on this value.

Six common information files are listed in Table 1. Although many other information files are available, the six listed in Table 1 are generally the most useful for automation purposes. In particular, the "param.inf" file is very versatile since it can be used in most of the Pro/Engineer modules and many different items may defined as a parameter.

External program menu option

A menu option should be added to the Utilities menu to run the external program or command. The external program should be written to read the information file, perform any desired calculations or actions, and then write a viable Pro/Engineer trail file. The external program may be a FORTRAN compiled program, a shell or awk script, or any program which can read the information file and write a trail file. Several different programs and commands can even be linked together under a single menu option.

Pro/Engineer is generally very sensitive to trail files and exits prematurely if an unexpected command is encountered during reading of a trail file. To ensure the external program writes a viable trail file, it is best to manually issue the desired commands in a Pro/Engineer session and use the automatically created "trail.txt" file as a template for the external program. The first two lines of a trail file must include the Pro/Engineer header information. Most system messages may be deleted from the trail file and keyboard entries may be included. However, it is generally not advised to include mouse activity in a manually created trail file since all mouse activity assumes a specific size active window to locate the mouse clicks. When the trail file is run with a different size active window, the program will exit prematurely. Instead of using the mouse to select items, the trail file should use "Select By Menu" and select items based on name, ID, or item number.

Automation menu option

Although the user can run the external program and subsequently issue "Misc-Trail" to run the trail file, a more efficient system is to add a second menu option which selects the external program menu option from the Utilities menu and then selects "Misc-Trail" to automatically run the associated trail file once the external program is finished. Unlike the external program menu option, this automation menu option may be placed on any menu since it consists of only Pro/Engineer commands.

We may also define a keyboard macro instead of creating an automation menu option.

Other options

The automation scheme is very versatile. The external program may be written in any format and can be used to perform any type of calculation or function. The scheme may also be modified for many other duties:

- exclude reading of information--to automate a combination of commands which may be too long or complicated for keyboard macros (such as when the commands involve numerical calculations)
- exclude writing a trail file--to extract and reformat model parameter information for use as input to other programs
- exclude both information and trail file--to automate a complicated function external to Pro/Engineer (such as running a script to generate a tabulated and sorted list of files in the working directory)

The automation scheme has been successfully implemented in a utility called ProBEND to automatically generate parts and automatically assemble the parts into an assembly representing a solid model of a magnet coil. The utility uses a 2D cross section, runs an external FORTRAN program to perform calculations of minimum strain energy, and automatically generates a magnet coil model based on the FORTRAN program calculations.

Example 1: Automatic Creation of Shade Plots Based on Model Height

Figure 2 displays the model. In the model dimension d3 is set equal to a user defined parameter called HEIGHT. We wish to create a menu option in the Part menu which automatically creates a shaded Postscript plot if HEIGHT is less than 20. The added menu option is useful if we had to perform the same operation on many different models.

Issuing "Setup-Parameter-Info" displays the parameter listing on the screen and also writes the "param.inf" file to the working directory. The FORTRAN program "heightplot" is written to read the "param.inf" file, determine if a Postscript plot should be generated, and then write a trail file if a plot file is needed.

Program "heightplot" is shown in Figure 3. The trail file which is written by "heightplot" (if HEIGHT is less than 20) is shown in Figure 4. Heightplot names the trail file "plottrail.txt" and places it in the working directory.

Two menu options are created:

- a menu option called "HTPLOT Prog" is added to the Utilities menu for executing program "heightplot"
- a menu option called "HeightPlot" is added to the Part menu which first selects the "HTPLOT Prog" menu option and then issues "Misc-Trail" to run the "plottrail.txt" trail file.

The two "menu_def.pro" entries to add the two menu options are shown in Figure 5. Note that a dummy menu entry is used to add a separator between the regular menu options and the added menu option in the Part menu.

To issue the command, the user simply issues "Setup-Parameter-Info" and then clicks on the "HeightPlot" menu option in the Part menu. If the HEIGHT parameter is less than 20, a Postscript file called "height.ps" is created, otherwise nothing happens.

Example 2: Creating Feature Lists of All Model Parts and Subassemblies.

A useful utility for documentation of Pro/Engineer models is the feature list. The feature list provides information on all features in a model, including the feature numbers, ID values, and status. It is often useful to refer to the feature list to obtain feature numbers or ID values so that you can select features using "Select by Menu". Generating a list of all the features in a large assembly is also useful for documentation of the model.

In Pro/Engineer, generating a complete feature list for all the parts and subassemblies becomes quite tedious when the assembly is large and complicated. The user uses the "Info" menu and has to individually select each part and subassembly in the assembly. This example shows how to use the automation scheme to automatically generate a complete feature list for all parts and subassemblies in a model.

Ordinarily, we select the menu commands "Info-Feature List" to get a listing of the features for a part or subassembly. When the feature list command is issued, Pro/Engineer displays the features on the screen but also creates a feature list file named "feature.lst" in the launching directory. To automatically list all the features in all parts and subassemblies, we simply need to repetitively issue the "Info-Feature List" command for each part and subassembly in the model. There are two problems which need to be addressed:

- how do we get a list of all the parts and assemblies in the model?
- how do we keep the feature list file "feature.lst" from being overwritten each time we issue a feature list command?

To solve the first problem, we note that an assembly tree containing information on all parts and subassemblies is listed when we issue the command sequence "Info-Model Info-Top Level", and that the information is also contained in a file named "*modelname*.inf.*n*", where *modelname* is the name of the model and *n* is a sequentially increasing integer. We therefore simply need to issue "Info-Model Info-Top Level" and extract the part and subassembly names from the "*modelname*.inf.*n*" file.

To solve the second problem, we issue a system level command which appends file "feature.lst" to a global file before each new feature listing. The global file will thus contain all the feature lists when the commands are completed.

Two shell scripts and one awk program are used to set up automatic feature listing:

1. list_top_lvl	UNIX shell script which creates a Pro/Engineer trail file called "list1.txt". "list1.txt" runs command sequences which creates a top level feature list file ("feature.lst") and a file with assembly tree information ("*modelname*.inf.1"). The script insures that file "*modelname*.inf.1" ends with a 1 by deleting all existing files with a "inf" extension.
2. list_all_feat	UNIX shell script which gets the model name from the top level feature list file ("feature.lst") and then runs an awk program on the assembly tree information file ("*modelname*.inf.1"). The awk program commands are contained in file "listfeat.awkin". The top level feature listing file is copied to the global feature list file ("allfeat.lst") after the awk program is completed.
3. listfeat.awkin	An awk program which extracts names of all the parts and subassemblies and writes a Pro/Engineer trail file called "list2.txt". "list2.txt" issues the command sequence "Info-Feature List-Assembly-Name" for each subassembly and "Info-Feature List-Part-Name" for each part. "list2.txt" also issues the command sequence

"Misc-Utilities-Append Feat" (see menu button #3 below) between each feature list command.

The shell scripts are linked together through added menu options defined in the "menu_def.pro" file. This file adds the following menu buttons:

1. List Top Lvl	Utilities menu button. Button runs shell script "list_top_lvl".
2. List AllFeat	Utilities menu button. Button runs shell script "list_all_feat".
3. Append Feat	Utilities menu button. Button runs system command to append file "feature.lst" to file "allfeat.lst".
4. Clean Up	Utilities menu button. Button runs system command to delete files created by menu buttons 1-3.
5. All Features	Assembly menu button. Button performs following functions: a. runs command sequence "Misc-Utilities-List Top Lvl" b. runs trail file "list1.txt" c. runs command sequence "Misc-Utilities-List AllFeat" d. runs trail file "list2.txt" e. runs command sequence "Misc-Utilities-Clean Up"

To generate a file containing feature lists for all parts and assemblies in a model, the user simply selects the "All Features" menu button from the Assembly menu. A file named "allfeat.lst" containing all the feature lists will then be written to the current directory.

The shell scripts and "menu_def.pro" definitions are shown in Figures 6-9.

File Name	Command to create file	Information in file
param.inf	"Setup-Parameter-Info"	user defined parameter names and values
rels.inf	"Relations-Show Relations"	relations and values
feature.lst	"Info-Feature List-..."	feature names, numbers, ID's in part or assembly
modelname.inf.#	"Info-Model Info-Top Level"	component assembly tree for model
feat_1.ibl	"Modify-Value" (in Part Mode)	cross section coordinates for parts created using "blend from file"
partname.ptd	"Save" (after creating Family Table)	family table instances

Table 1. Several common information files created by Pro/Engineer.

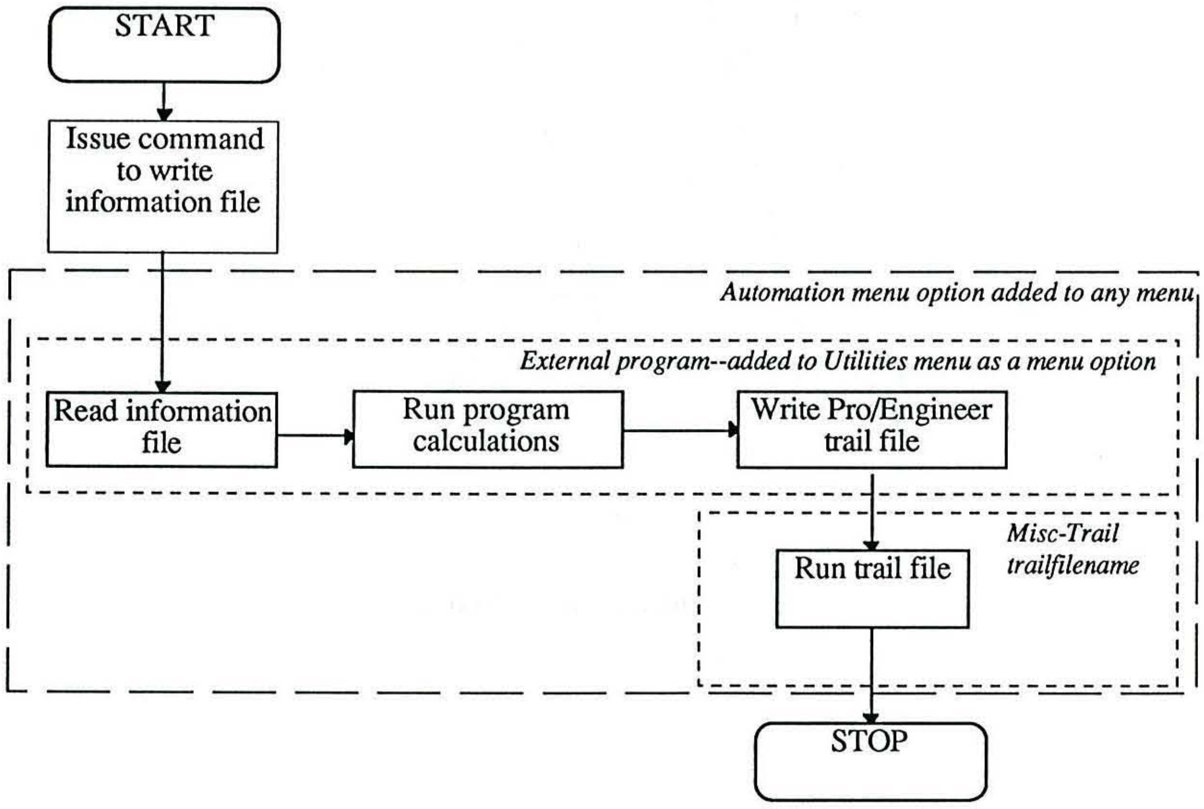
START
Issue command to write information file
Automation menu option added to any menu
External program--added to Utilities menu as a menu option
Read information file
Run program calculations
Write Pro/Engineer trail file
Misc-Trail trailfilename
Run trail file
STOP

Fig. 1. Automation scheme using added menu options, trail files, and information file.

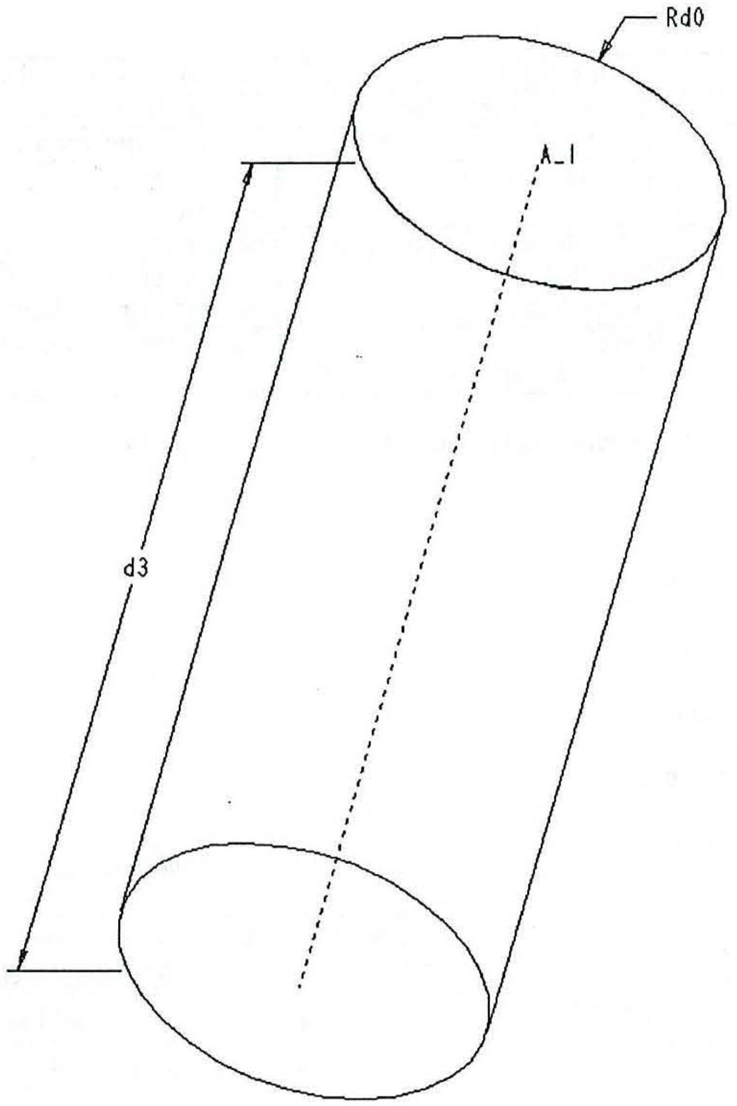

Fig. 2 Cylinder model.

```
PROGRAM HEIGHTPLOT
*
      INTEGER               endf,plot
      REAL                  height,numval
      CHARACTER             coname*18
*
*  Read HEIGHT parameter value from param.inf
*
      endf=0
      OPEN( UNIT=20, FILE='param.inf', STATUS='OLD')
      read(20,*)
      read(20,*)
      read(20,*)
      read(20,*)
      read(20,*)
      DO while  (endf.eq.0)
        READ(20,1000,END=900)coname,numval
        IF(coname.eq.'HEIGHT            ')then
            height=numval
        ENDIF
        goto 999
  900   CONTINUE
        endf=1
  999   CONTINUE
      ENDDO
      close(20)
 1000 FORMAT(1X,a18,22X,1pe12.6)
*
*  Determine if Postscript plot should be generated
*
      if  (height.lt.20.)then
         plot=1
      else
         plot=0
      endif
*
*  Write Trail file
*
      open(UNIT=22,FILE='plottrail.txt',STATUS='UNKNOWN')
      write(22,5000)
      if  (plot.eq.1)then
         write(22,5010)
      endif
      close(22)
 5000 FORMAT('!trail file version No. 876'
     | /'!Pro/ENGINEER  TM  Release 16.0  (c) '
     |  '1988-95 by Parametric Technology '
     |  'Corporation  All Rights Reserved.')
 5010 FORMAT('#VIEW'/'#COSMETIC'/'#SHADE'/'#DISPLAY'/'#POSTSCRIPT'
     | /'#DESIGNJET650C'/'#RESOLUTION'/'#100 DPI'/'#IMAGE SIZE'
     | /'#A'/'#OUTPUT TO PS'/'YES!'/'height'/'#DONE-RETURN')
      STOP
      END
*
```

Fig. 3. HEIGHTPLOT Program for generating plot file if HEIGHT parameter is less than 20.

```
!trail file version No. 876
!Pro/ENGINEER  TM  Release 16.0  (c) 1988-95 by Parametric Technology Corporation  All
Rights Reserved.
#VIEW
#COSMETIC
#SHADE
#DISPLAY
#POSTSCRIPT
#DESIGNJET650C
#RESOLUTION
#100 DPI
#IMAGE SIZE
#A
#OUTPUT TO PS
YES!
height
#DONE-RETURN
```

Fig. 4. Trail file "plottrail.txt" written by program HEIGHTPLOT.

```
@setbutton_exec UTILITIES HTPLOT#Prog "heightplot" "Runs HEIGHTPLOT program"
@setbutton PART =================== "" ""
@setbutton PART HeightPlot "#MISC;#UTILITIES;#HTPLOT Prog;#TRAIL;plottrail;\
#DONE-RETURN" "Runs HeightPlot and its trail file."
```

Fig. 5. "menu_def.pro" entries to add menu options for HEIGHTPLOT and HTPLOT Prog.

```
# Shell script list_top_lvl: creates a ProE trail file which performs
# two actions:
#  1. writes out file 'feature.lst'
#  2. writes out file 'modelname.inf.1'
# feature.lst      --> listing of top level features also used
#                      for getting the name of the model
# modelname.inf.1 --> file for getting subassem and part names
# Trail file name is 'list1.txt'
#
# Ken Chow
# May 21, 1996
# ----------------------------------------
rm -f *.inf.*
echo '!trail file version No. 876' > list1.txt
echo '!Pro/ENGINEER  TM  Release 16.0  (c) 1988-95 by Parametric Technology
Corporation  All Rights Reserved.' >> list1.txt
echo '#INFO' >> list1.txt
echo '#FEATURE LIST' >> list1.txt
echo '!Select assembly.' >> list1.txt
echo '#TOP LEVEL' >> list1.txt
echo '!Select assembly.' >> list1.txt
echo '#DONE/RETURN@INFO' >> list1.txt
echo '#INFO' >> list1.txt
echo '#MODEL INFO' >> list1.txt
echo '!Select assembly.' >> list1.txt
echo '#TOP LEVEL' >> list1.txt
echo '!Information written to file filename.inf.1.' >> list1.txt
echo '!Select assembly.' >> list1.txt
echo '#DONE/RETURN@INFO' >> list1.txt
```

Fig. 6. Shell script "list_top_lvl".

```
# Script file list_all_feat:
# 1. Gets the model name from top level 'feature.lst' file
# 4. Reads 'modelname.inf.1' to get subassem and part names and
#    then writes a ProE trail file named 'list2.txt' to generate
#    feature listing files ('feature.lst') of all subassemblies
#    and parts. Trail file also appends feature listing files to
#    model feature listing file called 'allfeat.lst'
# 3. Renames top level 'feature.lst' file to 'allfeat.lst'.
#
#   Ken Chow
#   May 21, 1996
#----------------------------------------
fnam=`awk '/MODEL NAME/ {print tolower($4)".inf.1"}' feature.lst`
cat $fnam | awk -f listfeat.awkin - > list2.txt
mv feature.lst allfeat.lst
```

Fig. 7. Shell script "list_all_feat".

```
# Awk input file 'listfeat.awkin' used in script 'list_all_feat'
# Writes a ProE trail file which selects each model subassembly and part
# and lists its features. Also runs menu option 'Append Feat' after each
# feature listing to catenate files into one large file named 'allfeat.lst'.
#
#  Ken Chow
#  May 21, 1996
# --------------------------------------------------
BEGIN {print "!trail file version No. 876";
       print "!Pro/ENGINEER  TM  Release 16.0  (c) 1988-95 by Parametric Technology
Corporation  All Rights Reserved."}  # print trail file header
/LEVEL/ && /ASSEM/ {if ($2!=1) {print "#INFO";
                                print "#FEATURE LIST";
                                print "!Select assembly.";
                                print "#ASSEMBLY";
                                print "!Select assembly.";
                                print "#NAME";
                                print "!Enter assembly name:";
                                print $3;
                                print "#MISC";
                                print "#UTILITIES";
                                print "#APPEND FEAT";
                                print "#DONE/RETURN@INFO"}}
/LEVEL/ && /PART/ {if ($2!=1) {print "#INFO";
                                print "#FEATURE LIST";
                                print "!Select assembly.";
                                print "#PART";
                                print "!Select PART.";
                                print "#NAME";
                                print "!Enter part name:";
                                print $3;
                                print "#MISC";
                                print "#UTILITIES";
                                print "#APPEND FEAT";
                                print "#DONE/RETURN@INFO"}}
```

Fig. 8. Awk input file "listfeat.awkin".

```
@setbutton_exec UTILITIES List#Top#Lvl "sh list_top_lvl" "Creates trail file for top
level feature listing."
@setbutton_exec UTILITIES List#AllFeat "sh list_all_feat" "Creates trail file for
listing features in all parts."
@setbutton_exec UTILITIES Append#Feat "cat feature.lst >> allfeat.lst" "Appends
feature list file to allfeat.lst file."
@setbutton_exec UTILITIES Clean#Up "rm -f feature.lst *.inf.1 list1.txt list2.txt"
"Cleans up directory."
@setbutton ASSEMBLY ==================== "" ""
@setbutton ASSEMBLY AllFeatures "#MISC;#UTILITIES;#List Top Lvl;#TRAIL;list1;#MISC;\
#UTILITIES;#List AllFeat;#TRAIL;list2;#MISC;#UTILITIES;#Clean Up;#DONE-RETURN"\
"Creates file allfeat.lst--listing of all subassem and part features."
```

Fig. 9. Menu button definitions used in file "menu_def.pro".