The Diffusion of Photovoltaics: Background, Modeling, Calibration and Implications for Government Policy

Gary L. Lilien

May 1978

MIT ENERGY LABORATORY REPORT - MIT-EL-78-019

PREPARED FOR THE UNITED STATES

DEPARTMENT OF ENERGY

Under Contract No. EX-76-A-01-2295
Task Order 37

Note: Charles Allen, Arnold Barnett, Pat Burns, Tom McCormick, Bruce Schweitzer, and Leif Soderberg contributed significantly to the material described in this paper.

APPENDIX 1.1 PV PROGRAM

A.1.1 A Users Guide to the Photovoltaics Simulation System

The photovoltaics simulation system is a self-contained interactive set of computer programs on the Multics system at MIT. It allows the user to exercise the PV model for different sets of parameters by providing general facilities for creating, saving, editing, using, displaying and deleting sets of parameters. The organization of these sets of parameters reflects the model: the full set of input data is kept in a model, which contains parameters which apply across sectors, such as the cost decline factor, as well as the names of its constituent sectors. A sector, in turn, contains the data that are relevant only to itself, such as average PV installation size. Finally, when a model and its associated sectors are used to perform a simulation, the results are saved in a run, which records all the model and sector input data as well. The commands in the system are used to manipulate these three types of sets of data, or records.

The actual model is defined by the following key variables and equations:

X_{it} = number of kwatts of government PV installed in sector i during time t

Y_{it} = number of Kwatts of private PV installed in sector i during time t

C_t = cost of one Kwatt of PV at the end of period t

Z_i = average number of Kwatts per PV installation in sector i

N_{it} = cummulative Kwatts of PV installed in sector i up to time t

N_t = cumulative production of Kwatts of PV up to time t

N_0 = Kwatts of PV installed before model begins

C_0 = cost per Kwatt of PV before model begins

λ = decimal factor by which cost declines when PV production doubles

$F_{c_i}(\cdot)$ = cumulative normal distribution for cost acceptability, parameterized by μ_{c_i} and σ_{c_i}

$F_{s_i}(\cdot)$ = cumulative normal distribution for successful installations acceptability, parameterized by μ_{s_i} and σ_{s_i}

h_i = probability of purchase by randomly selected individual in sector i given acceptability

P_{i0} = initial potential market for PV in sector i

P_{it} = total potential market for PV in sector i at time t

Q_i = potential original equipment PV installation that becomes available per time period

g_i = growth rate of existing market potential in sector i

The equations relating these variables are:

$$N_{it} = \sum_{\tau=1}^{t} (X_{i\tau}+Y_{i\tau})$$

$$N_t = N_0 + \sum_i N_{it}$$

$$P_{it} = (P_{it-1} - Q_i)(1+g_i) + Q_i$$

$1 - F_{c_1}(C_{t-1})$ = probability that cost is acceptable at the beginning of period in sector i.

$F_{s_i}(N_{it-1}/Z_i)$ = probability that installed PV is acceptable in sector i at the beginning of period t.

$$Y_{it} = (P_{it} - N_{it-1}/Z_i)(1-F_{ci}(C_{t-1}))\ F_{si}(N_{it-1}/Z_i) \cdot h_i \cdot Z_i$$

$$C_t = C_0(N_t/N_0)^{\log_2 \lambda}$$

These equations describe the model that is embodied in the simulation system. Tables I, II and II (for models, sectors and runs, respectively) show the correspondence between the variable names used here and the names as they appear in the computer programs and on the output.

In use, the computer and the user exchange information through a computer terminal. There is a wide variety of computer terminals, which have various switches that must be properly set to communicate with Multics. After turning the terminal on, a COM/LCL switch should be on COM, a REMOTE/LOCAL switch on REMOTE, a FULL/HALF switch on HALF, a CONT MSG switch ON, and a speed switch on HIGH. Once the terminal is ready, you must "dial-up" the computer, that is, establish the telephone link between

it and your terminal. The phone number is 258-8215 for IBM 2741-like terminals and 258-8313 for ASCII-type terminals. The general procedure is to dial the number; you should get a high-pitched tone in response. Then, depending on the communication system used, push the DATA button or pull up the white button in the handset cradle or insert the handset into the acoustic coupler. On 2741-like terminals, Multics will respond with a two-line message. For ASCII-type terminals, you must type a speed identification character first, followed by a line-feed. Use "s" for 110-baud terminals (11 characters/second), "y" for 150-baud (15 cps) or "o" for 330-baud (30 cps). Multics will then respond with its two-line message. Then type

login SMcCormick

and hit carriage return for 2741, line-feed for ASCII. You will then be prompted for the password; enter it, again followed by carriage return for 2741, line-feed for ASCII. After that, Multics will print some login messages and the simulation system's opening message will print. From this point on, everything you type on either kind of terminal should be terminated with a carriage return.

The system will ask if you need the system explained (see attached sample terminal session). If you are familiar with the system, type "no", otherwise type "yes". The system expects the full word "yes" or "no" whenever it asks you a question, though it generally assumes a "no" answer if anything other than "yes" is typed. If you answered "yes", a brief description of the commands will be printed. In either case, the system then prints "READY" as a signal that it is awaiting your first command.

You then type in the command that you want performed. If you mistype the command or type a non-existent command, the system will respond:

The command XXX does not exist - try again

with XXX replaced by the command you typed. You should then type in a valid command. After the command has finished processing (which may require that you type in information), the system again prints "READY" and awaits your next command. This cycle is terminated when you type the command:

return

which will stop processing and automatically log you off the computer. You will be informed of this by the message:

You will now be disconnected from the computer. Goodbye.

There may be circumstances when you don't want a command to finish an action it has started. For instance, you may start displaying something that is longer than you thought. In any such situation, press the "quit" button on the terminal (which may be labeled "attn" or "break" on some terminals). Whatever the command is doing will stop, "QUIT" will be printed, and then "READY" will be printed to indicate that the next command may be entered. In certain situations an additional informational message in parentheses will be printed before "QUIT" to inform you of the status of the records that were being processed. Also, should an unforseen error arise, an error message will be printed containing information so the system maintainer can diagnose the error, followed by an artificially created quit signal to return you to command level. Any error messages should be reported promptly to the system maintainer.

There are general conventions that apply to all commands. The format in which they are typed is the command name followed by other information, separated by blanks. For instance,

```
create model example
```

would be used to create a model named example. The commands are flexible though; in this example, "create" could be abbreviated to "cr", and any word beginning with "m" could be substituted for "model." Thus

```
cr m example
create mood example
cr money example
```

would all have the same effect as the previous example. In general, most commands have one or two letter abbreviations, and whenever the next field entered is a type ("model," "sector" or "run"), only the first letter is significant. All names are used as is with no abbreviation allowed. Different types of records are allowed to have the same name (i.e., you can have a model and a sector both named test), but two records of the same type must have different names. The programs check for this and give you a chance to re-enter a different name or overwrite the already existing record if you try (inadvertently or not) to use a duplicate name. You can also choose to "cancel" in such circumstances, which means that you are returned to command level as if you never entered the original command. The programs check to see if you have entered a valid record type, and prompt you if you haven't. For example:

```
cr x wrong
```

is answered by:

First argument must be model or sector. Reenter:

You can then type a word beginning with "m" or "s" and create the proper record. If you realize that you don't want to create anything, you can hit "quit" to return to command level. In addition, if you leave out arguments, the programs will prompt you for them. For example:

cr model

is answered by:

Enter model name:

Also several commands allow you to process several records of the same type at once. For instance:

cr mood test1 test2 test3

will create three models.

The available commands are:

create - creates models or sectors

copy - copies models or sectors

modify - modifies models or sectors

delete - deletes models, sectors or runs

display - displays models, sectors or runs

run - runs a model creating a run

index - lists records on file of a given type

help - prints a brief description of command syntax

return - terminates processing, logs you off

The last two commands are self-explanatory and have no arguments. The other seven commands are described below.

create: Syntax: create TYPE NAME1 NAME2 . . .

Abbreviation: cr

TYPE: can be model or sector

NAMES: are 0 or more names of records to be created.

When this command is invoked it will successively prompt you for all the data items needed to create a model or sector. When the last item has been entered, a message is issued confirming that the record was created, then the next record (if any) is processed.

copy: Syntax: copy TYPE OLDNAME NEWNAME

Abbreviation: cp

TYPE: can be model or sector

OLDNAME: name of existing record which is to be copied

NEWNAME: name that the copy is to saved under

This command copies the record OLDNAME of the indicated TYPE under the name NEWNAME and issues a confirming message. Since the use of this command is frequently to have several versions of the same record with minor changes, you will then be asked if you want to modify the new record. Answering "yes" is equivalent to issuing the command "modify TYPE NEWNAME."

modify: Syntax: modify TYPE NAME1 NAME2 ...

Abbreviation: m

TYPE: can be model or sector

NAMES: are 0 or more names of records to be modified

When invoked, modify asks for your first subcommand. A dictionary of subcommands is printed by typing "help." Most subcommands are the names of variables that can be changed. To do this, enter the variable name. The program reports back the old value and asks for the new value. Then it prompts you for the next subcommand. There are also subcommands to display the modified record and to finish processing and return to command level.

delete: Syntax: delete TYPE NAME1 NAME2 ...

TYPE: can be model, run or sector

NAMES: are 0 or more names of records to be deleted

There is no abbreviation for delete, to avoid accidental deletions. For models and runs, delete checks the index to see if they have been run and displayed, respectively, and asks you to confirm whether you really want to delete these unused records or not. A confirming message is printed after each deletion.

display: Syntax: display TYPE NAME1 NAME2 ...

Abbreviation: dp

TYPE: can be model, sector or run

NAMES: are 0 or more names or records to be displayed

This command prints the requested records on the terminal, one at a time. For a run, display asks you if you want model, sector, both or neither input variables printed (again, only the first letter is significant, so "m", "s", "b" or "n" is sufficient), and give you the same choice for output variables. Successively answering "b", "b" will print everything, "n", "n" will print nothing, "s", "s" will print only sector-specific input and output variables (which depend on model input variables however), etc.

run: Syntax: run MODELNAME RUNNAME

Abbreviation: r

MODELNAME: name of the model to be run

RUNNAME: name that the output will be saved under

After the model is run, the program prints a confirming message and asks if you would like to have the run displayed. Answering yes is equivalent to issuing the command "display run RUNNAME".

index: Syntax: index TYPE

Abbreviation: i

TYPE: can be model, sector, run or all

If TYPE is "all" or left blank, index prints a table of all records created and still on file giving their name, their type, and for models and runs, under the heading "Used?", an indication of whether they have been run and displayed yet. If TYPE is model, sector or run, a table restricted to only that type of record is printed with no type column but retaining the "Used" column.

To clarify these ideas an annotated terminal session is attached, followed by a complete listing of the program.

TABLE I: Model Correspondences

English Concept	Symbolic Notation	PL/I Variable Name	Modify Mnemonic
Name	--	mname	name
Number of Sectors	--	n	n
Number of time periods	--	tmax	tmax
Initial cost	C_0	initcost	initcost
Cost decay rate	λ	costdecay	costdecay
Initial installed Kwatts of PV	N_0	initwatt	initwatt
Name of sector i	--	sn(i)	secname(i)
Government investment in sector i at time t	X_{it}	govinv(i,t)	govinv(i,t)

TABLE II: Sector Correspondences

English Concept	Symbolic Notation	PL/I variable name	Modify mnemonic
Name	--	sname	name
Average PV installation size (Kwatts)	Z_i	avgsize	avgsize
Initial existing potential market (Kwatts)	P_{i0}	initpotinst	initpotin
Growth rate of existing potential market	g_i	buildrate	buildrate
Potential for original equipment installation per time period (Kwatts)	Q_i	newinst	newinst
Purchase probability given acceptance	h_i	ppurch	ppurch
Mean of cost distribution	μ_{c_i}	cmean	costmean
Std. dev. of cost distribution	σ_{c_i}	cdev	costdev
Mean of successes distribution	μ_{s_i}	imean	instmean
Std. dev. of successes distribution	σ_{s_i}	idev	instdev

TABLE III: Run Correspondences

The input parameter parts of a run follow Tables I and II. Model-wide output has these correspondences.

English Concept	Symbolic Notation	PL/I Variable Name	Output Heading
Cost	C_t	cost(t)	cost
Cumulative Kwatts installed by the end of t	N_t	totwatt(t)	totwatt
Governmen investment during this period	$\sum_i X_{it}$	govinvnow(t)	govinv
Private investment during this period	$\sum_i Y_{it}$	privinvnow(t)	privinv
Cumulative government investment including this period	$\sum_{\tau=1}^{t} \sum_i X_{i\tau}$	allgovinv(t)	totginv
Cumulative private investment including this period	$\sum_{\tau=1}^{t} \sum_i Y_{i\tau}$	allprivinv(t)	totpinv

TABLE III: Run Correspondences (Continued)

Sector specific output has these correspondences:

English Concept	Symbolic Notation	PL/I Variable Name	Output Heading
Government investment in i at t (this is repeated from input)	X_{it}	govinv (i,t)	govinv
Private investment in i at t	Y_{it}	privinv(i,t)	privinv
Government PV installations in i at t	$X_{it}/(Z_i C_{t-1})$	govinst(i,t)	govinst
Private PV installations in i at t	$Y_{it}/(Z_i C_{t-1})$	privinst(i,t)	privinst
Cumulative government investment in i including t	$\sum_{\tau=1}^{t} X_{i\tau}$	totgovinv(i,t)	totginv
Cumulative private investment in i including t	$\sum_{\tau=1}^{t} Y_{i\tau}$	totprivinv(i,t)	totpinv
Cumulative total investment in i including t	$\sum_{\tau=1}^{t} (X_{i\tau}+Y_{i\tau})$	totinv(i,t)	totinv
Cumulative government PV installations in i including t	$\sum_{\tau=1}^{t} X_{i\tau}/(Z_i C_{\tau-1})$	totgovinst(i,t)	totginst
Cumulative private PV installations in i including t	$\sum_{\tau=1}^{t} Y_{i\tau}/(Z_i C_{\tau-1})$	totprivinst(i,t)	totpinst

TABLE III: Run Correspondences (Continued)

English Concept	Symbolic Notation	PL/I Variable Name	Output Heading
Cumulative total PV installations in i including t	$\sum_{\tau=1}^{t} (X_{i\tau}+Y_{i\tau})/(Z_i C_{\tau-1})$	totinst(i,t)	totinst
Potential PV market in i at t	P_{it}	potinst(i,t)	potinst
Cost acceptance in i at the beginning of t	$1-F_{c_i}(C_{t-1})$	costaccep(i,t)	costacc
Successes acceptance in i at the beginning of t	$F_{s_i}(N_{it-1}/Z_i$	instaccep(i,t)	instacc

APPENDIX 2

FACTOR STRUCTURE COMPARISON: AGRICULTURAL SECTOR

With Choffray and Lilien's [13] modification of the Chow test, when we compare the evaluation criteria of A^* against C^* we must also look at the evaluation of the combined group, (A + C). Actually, we are comparing A and C with (A + C) to determine equality; our null hypothesis is that the groups are, in fact, equal. Defining:

$$C_p = \left[\frac{N(1-R_p^2)}{n_1(1-R_{p1}^2) + n_2(1-R_{p2}^2)} - 1 \right] \frac{N-2q}{q}$$

where,

R_{p1}^2, R_{p2}^2 = squared multiple correlations associated with the estimation of factor p in sample 1 and 2 respectively

R_p^2 = squared multiple correlation associated with factor p in pooled sample

n_1, n_2 = number of responses in sample 1 and 2 respectively

$N = n_1 + n_2$

q = number of items

When the value of C_p exceeds the corresponding F statistic at the appropriate level of significance, the null hypothesis of equality is rejected.

*Group A = Pre-test and post-test survey
Group B = pre-test survey only
Group C = post-test survey only

APPENDIX 2 (continued)

TABLE 1

For comparison of A and C:

Factor	C_p	$F_{.90}$	$F_{.95}$	$F_{.99}$
1	0.41	1.72	2.01	2.75
2	0.42	1.72	2.01	2.75
3	1.79	1.72	2.01	2.75

null hypothesis is accepted at the .95 level

TABLE 2

For comparison of B and C:

Factor	C_p	$F_{.90}$	$F_{.95}$	$F_{.99}$
1	2.62	1.72	2.01	2.75
2	2.53	1.72	2.01	2.75
3	0.22	1.72	2.01	2.75

null hypothesis is rejected at the .95 level

APPENDIX 3: FACTOR STRUCTURE COMPARISON: RESIDENTIAL SECTOR

TABLE 1

COMPARISON OF GROUP 1 AND GROUP 2

Factor	C_p	$F_{.90}$	$F_{.95}$	$F_{.99}$
1	5.76	1.72	2.01	2.75
2	10.58	1.72	2.01	2.75
3	3.34	1.72	2.01	2.75

Null Hypothesis of equality is rejected

COMPARISON OF GROUP 3 AND GROUP 4

Factor	C_p	$F_{.90}$	$F_{.95}$	$F_{.99}$
1	0.94	1.72	2.01	2.75
2	2.42	1.72	2.01	2.75
3	1.24	1.72	2.01	2.75

Null Hypothesis is rejected at the .95 level

APPENDIX 3 (continued): TABLE 2

MODIFIED CHOW TEST

COMPARISON OF GROUP 1 AND GROUP 4

Factor	C_p	$F_{.90}$	$F_{.95}$	$F_{.99}$
1	2.44	1.72	2.01	2.75
2	3.72	1.72	2.01	2.75
3	-8.88	1.72	2.01	2.75

Null Hypothesis of equality is rejected

COMPARISON OF GROUP 2 AND GROUP 3

Factor	C_p	$F_{.90}$	$F_{.95}$	$F_{.99}$
1	6.57	1.72	2.01	2.75
2	1.92	1.72	2.01	2.75
3	0.88	1.72	2.01	2.75

Null Hypothesis of equality is rejected

A.4 Sample Interactive Program Session

```
Multics 33.0 MIT, Cambridge, Mass.
Load = 36.0 out of 85.0 units: users = 36
! SMcCormick
Password:
xxxxxxxxxx
You are protected from preemption.
SMcCormick SOLAR logged in 05/04/78  1453.8 edt Thu from 2741 terminal "none".
Last login 05/04/78  1435.3 edt Thu from ASCII terminal "none".

Welcome to the photo-voltaics simulation model.

Do you need an explanation of how the simulation system works (yes or no) ? :  yes

Input and output data reside in one of 3 kinds of data blocks - sector, for sector-specific input data,
model, for model-wide input data, and run, for output data (including the inputs which it used).  The
commands for manipulating these blocks are entered as follows (where TYPE is model, sector or run, and
the NAMEs are to be replaced with user-defined names.  Alternatively, any or all parameters may be left
out - you will be prompted for them):

To create models or sectors:
     create
  or       TYPE NAME1 NAME2 ...
     cr

To copy models or sectors:
     copy
  or     TYPE OLDNAME NEWNAME
     cp

To display models, sectors or runs:
     display
  or         TYPE NAME1 NAME2 ...
     dp

To delete models, sectors or runs:
     delete TYPE NAME1 NAME2 ...

To modify models or sectors:
     modify
  or       TYPE NAME1 NAME2 ...
     m

To run a model:
     run
  or     MODEL NAME RUNNAME
     r

To get an index of data on file (all blocks will be listed if TYPE is all or blank):
     index
  or       TYPE
     l

To reprint these instructions:
     help

To stop processing and disconnect from the computer:
     return

READY (Cost: $1.15)
```

Multics sign on messages

login, followed by line-feed for ASCII terminals, by carriage return for 2741 terminals

password, terminated as above

Multics login messages

now all lines are terminated by carriage return, for all terminals

The READY message signifies that the previous action is finished and the system is awaiting your next command. Normally the cost figure would not appear; it has been included here for illustrative purposes.

```
create model showoff
```
← This user-entered command initiates prompting for data entry into a model named "showoff".

```
Enter the following for model showoff:
Number of sectors:  3

Number of time periods:  5

Initial dollar cost per Kwatt:  10000

Decimal factor by which cost declines when production doubles:  70

Initial number of Kwatts installed across all sectors:  1000

The  3 sector names: farmers houses central_power

The  5 government investment values (in M$) for sector

farmers : 40 40 50 50 60

houses : 10 10 10 10 10

central_power : 0 0 0 0 0

The model showoff has been created.
```
← This message confirms that the model has been put on file

```
READY (Cost: $1.25)
```
← and we're ready for our next command

```
cr s farmers houses
```
← This time "create" has been abbreviated to "cr", "sector" to "s". Two sectors will be created

```
Enter the following for sector farmers:
Average Kwatts per installation:  100

Initial potential old installations:  200000

Decimal rate at which potential installations grow:  .03

New installations per period:  0

Decimal purchase probability given acceptance:  .4

Cost per Kwatt acceptance mean and std dev:  800 300

Sucessful installations acceptance mean and std dev:  5 3

The sector farmers has been created.
```
← the first sector is complete

```
Enter the following for sector houses:
Average Kwatts per installation:  500

Initial potential old installations:  50000

Decimal rate at which potential installations grow:  .02

New installations per period:  1000

Decimal purchase probability given acceptance:  .2

Cost per Kwatt acceptance mean and std dev:  500 100

Sucessful installations acceptance mean and std dev:  6 3

The sector houses has been created.
READY (Cost: $0.81)
```

note that this was incorrectly entered as a percent figure rather than the decimal figure that was asked for

```
run showo.                                  ← Now we try to run our model

The sector central_power in model showoff is not in the file.  ← but we have forgotten to create the third sector,
Check the model definition.  Run aborted.                        so nothing happens.
READY (Cost: $0.33)
create s centralpower  ← We have mis-spelled the name.

Enter the following for sector centralpower:
Average Kwatts per installation:  5000

Initial potential old installations:  1000

Decimal rate at which potential installations grow:  .01

New installations per period:  0

Decimal purchase probability given acceptance:  .5

Cost per Kwatt acceptance mean and std dev:  200 50

Sucessful installations acceptance mean and std dev:  3 1

The sector centralpower has been created.
READY (Cost: $0.48)
r showoff

The sector central_power in model showoff is not in the file.  ← Since the name was mis-spelled above, run still can't find
Check the model definition.  Run aborted.                          the third sector
READY (Cost: $0.31)
modify model showoff  ← So we will modify "showoff" so that it points to the sector we created; we could also have modified the name of the
                                                                                                    sector itself
Enter "help" for a dictionary of variables, "quit" to return, or the name of the variable to be modified:  help

Name          Description

name          name of the model
n             number of sectors
tmax          number of time periods
initcost      initial cost per Kwatt of PV
costdecay     decimal fraction by which PV cost declines when production doubles
initwatt      total Kwatts of PV installed at time 0 across all sectors
secname(i)    name of the ith sector in the model
govinv(i,j)   government investment in sector i at time j (in M$)
quit          terminates processing, saves updated model
qdisplay      terminates processing, displays and saves updated model
display       displays updated model so far

Next request:  secname(5)  ← We accidently typed "5" instead of "3"

Subscript out of range.  Request ignored.
Next request:  secname(3)

Old value of secname(3) is central_power.  Enter new value:  centro###alpower

Next request:  secname(3)  ← we request secname(3) again to demonstrate that the character deletion worked properly

Old value of secname(3) is centralpower.  Enter new value:  centralpower

Next request:  nonsense  ← we try a non-existant request
```

When you notice a typing mistake, typing "#" deletes the previous character

```
There is no variable named nonsense.  Try again.
Next request:  initcost

Old value of initcost is 10000.00.  Enter the new value: 14ooo

"14ooo" is an invalid entry.  Request ignored.
Next request:  initcost

Old value of initcost is 10000.00.  Enter the new value: 14000

Next request:  qdisplay

Model showoff:

Cost per Kwatt of PV at time 0                                    14000.00
Decimal fraction by which cost declines when production doubles      70.00
Kwatts of PV installed at time 0 across sectors                    1000.00
The model will run until time =  5, and has  3 sectors.

Sector                        Government investment (in M$) at time
Name                            1        2        3        4        5

farmers                       40.00    40.00    50.00    50.00    60.00
houses                        10.00    10.00    10.00    10.00    10.00
centralpower                   0.00     0.00     0.00     0.00     0.00

READY (Cost: $1.15)
r showoff

Cost decay rate is out of range.  Run aborted.
READY (Cost: $0.32)
m m showoff

Enter "help" for a dictionary of variables, "quit" to return, or the name of the variable to be modified:  costdecay

Old value of costdecay is    70.00.  Enter the new value: .7

Next request:  q

READY (Cost: $0.27)
r showoff

Enter the run name :  output

The model showoff has been run under the name output.
Do you want to display this run? :  no

READY (Cost: $0.54)
display run outputddelete r output

The run output has not been displayed.  Do you still want to delete? :  no

READY (Cost: $0.30)
display r output

Do you want to see model, sector, both or neither input variables? :  s

Same choice for output variables:  both

time      cost      totwatt      govinv      privinv      totginv      totpinv
```

We mis-typed the number "0" as the letter "o"

So initcost wasn't changed

Mis-entering the cost decay rate has finally caught up with us.

Finally everything is ok, so we are prompted for a RUNNAME.

If you want to delete an entire typed line, typing "@" will do it faster than a string of "#"s.

We stupidly try to delete this run before its been displayed.

But the program reminds us of this

We abbreviatedly ask for sector inputs only as we've displayed the model inputs above

But we want to see all output

1	6404.47	4571.43	50.00	0.00	50.00	0.00
2	3835.93	12378.48	50.00	0.00	100.00	0.00
3	2519.41	28020.05	60.00	0.00	160.00	0.00
4	1835.81	51835.13	60.00	0.00	220.00	0.00
5	1382.30	89970.20	70.00	0.01	290.00	0.01

} model-wide output

Sector farmers:

Input and Output for sector "farmers"

Average installation size (Kwatts)	100.00	
Initial potential old installations	200000	
Decimal growth rate of potential installations	0.03	
Number of new installations per period	0.00	
Decimal probability of purchase given acceptability	0.40	
Feasible cost per Kwatt mean and std dev	800.00	300.00
Feasible number of successes mean and std dev	5.00	3.00

time	govinv	privinv	govinst	privinst	totginv	totpinv	totinv	totginst	totpinst	totinst	potinst	costacc	instacc
1	40.00	0.00	28.57	0.00	40.00	0.00	40.00	28.57	0.00	28.57	206000.00	0.0000	0.0092
2	40.00	0.00	62.46	0.00	80.00	0.00	80.00	91.03	0.00	91.03	212179.99	0.0000	1.0000
3	50.00	0.00	130.35	0.00	130.00	0.00	130.00	221.37	0.00	221.37	218545.39	0.0000	1.0000
4	50.00	0.00	198.46	0.00	180.00	0.00	180.00	419.83	0.00	419.83	225101.75	0.0000	1.0000
5	60.00	0.01	326.83	0.05	240.00	0.01	240.01	746.66	0.05	746.71	231854.79	0.0000	1.0000

Sector houses:

Average installation size (Kwatts)	500.00	
Initial potential old installations	50000	
Decimal growth rate of potential installations	0.02	
Number of new installations per period	1000.00	
Decimal probability of purchase given acceptability	0.20	
Feasible cost per Kwatt mean and std dev	500.00	100.00
Feasible number of successes mean and std dev	6.00	3.00

time	govinv	privinv	govinst	privinst	totginv	totpinv	totinv	totginst	totpinst	totinst	potinst	costacc	instacc
1	10.00	0.00	1.43	0.00	10.00	0.00	10.00	1.43	0.00	1.43	52000.00	0.0000	0.0023
2	10.00	0.00	3.12	0.00	20.00	0.00	20.00	4.55	0.00	4.55	53020.00	0.0000	0.0156
3	10.00	0.00	5.21	0.00	30.00	0.00	30.00	9.77	0.00	9.77	54060.40	0.0000	0.2473
4	10.00	0.00	7.94	0.00	40.00	0.00	40.00	17.70	0.00	17.70	55121.61	0.0000	0.9620
5	10.00	0.00	10.89	0.00	50.00	0.00	50.00	28.60	0.00	28.60	56204.04	0.0000	1.0000

Sector centralpower:

Average installation size (Kwatts)	5000.00	
Initial potential old installations	1000	
Decimal growth rate of potential installations	0.01	
Number of new installations per period	0.00	
Decimal probability of purchase given acceptability	0.50	
Feasible cost per Kwatt mean and std dev	200.00	50.00
Feasible number of successes mean and std dev	3.00	1.00

time	govinv	privinv	govinst	privinst	totginv	totpinv	totinv	totginst	totpinst	totinst	potinst	costacc	instacc
1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1010.00	0.0000	0.0000
2	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1020.10	0.0000	0.0000
3	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1030.30	0.0000	0.0000
4	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1040.60	0.0000	0.0000
5	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1051.01	0.0000	0.0000

```
READY (Cost: $0.43)
mod sec farmers
The command mod does not exist - try again
m sec farmers

Enter "help" for a dictionary of variables, "quit" to return, or the name of the variable to be modified: help

Name       Description

name       name of  the sector
avgsize    average number of Kwatts in a PV installation
initpotin  number of potential installations at time 0
buildrate  decimal growth rate of potential installations
newinst    number of new installations per time period
ppurch     decimal probability of PV purchase given acceptability
costmean   mean of cost feasibility distribution
costdev    std dev of cost feasibility distribution
instmean   mean of sucessful installation distribution
instdev    std dev of sucessful installation distribution
quit       terminates processing, saves updated sector
qdisplay   terminated processing, displays and saves updated sector
display    displays updated sector so far

Next request: ppurch

Old value of ppurch is      0.40.  Enter the new value: .8

Next request: q

READY (Cost: $0.54)
r

Enter name of the model to be run : showoff

Enter the run name : highfarmpurch

The model showoff has been run under the name highfarmpurch.
Do you want to display this run? : yes

Do you want to see model, sector, both or neither input variables? : none

Same choice for output variables: melon

time      cost       totwatt      govinv      privinv      totginv      totpinv
  1     6404.47      4571.43       50.00         0.00        50.00         0.00
  2     3835.93     12378.48       50.00         0.00       100.00         0.00
  3     2519.41     28020.05       60.00         0.00       160.00         0.00
  4     1835.81     51835.13       60.00         0.00       220.00         0.00
  5     1382.26     89975.02       70.00         0.02       290.00         0.02

READY (Cost: $0.61)
index all

All records

Name                          Type   Used?

Aagr                          model  yes
```

TYPEs can

Commands cannot be arbitrarily abbreviated the

Here our command is the minimum required - one letter

We are prompted for the missing parameters

We've already seen the inputs

We ask for model output only in a silly way

In this case, changing probability of purchase in sector "farmers" from .4 to .8 has resulted in an increase of only 5 Kwatts in period 5.

Here we ask for an index of all records on file

```
Acr                             model   yes
Arr                             model   yes
agr                             model   yes
pb                              model   yes
rr                              model   yes
showoff                         model   yes
highfarmpurch                   run     yes
output                          run     yes
ag                              sector  n/a
ag1                             sector  n/a
ag2                             sector  n/a
ag3                             sector  n/a
ag4                             sector  n/a
ag5                             sector  n/a
ag6                             sector  n/a
ag7                             sector  n/a
c                               sector  n/a
centralpower                    sector  n/a
farmers                         sector  n/a
houses                          sector  n/a
r                               sector  n/a
r1                              sector  n/a
r2                              sector  n/a
r3                              sector  n/a
r4                              sector  n/a
r5                              sector  n/a
r6                              sector  n/a
r7                              sector  n/a

READY (Cost: $0.21)
copy model showoff demonstration

The model showoff has been copied under the name demonstration.
Do you want to modify model demonstration? : yes

Enter "help" for a dictionary of variables, "quit" to return, or the name of the variable to be modified: costdecay

Old value of costdecay is      0.70. Enter the new value: .6

Next request: q

READY (Cost: $0.51)
index mods

All models

Name                            Used

Aagr                            yes
Acr                             yes
Arr                             yes
agr                             yes
demonstration                   no
pb                              yes
rr                              yes
showoff                         yes

READY (Cost: $0.22)
run demonstration fastdecay
```

The "used" column has "yes" for a model if it has been run and "no" otherwise, has "yes" for a run if it has been displayed at all, "no" otherwise, and always has "n/a" for sectors.

← the new model is added to the index, and has not been used

```
The model demonstration has been run under the name fastdecay.
Do you want to display this run? : yes

Do you want to see model, sector, both or neither input variables? : n

Same choice for output variables: m

time      cost      totwatt      govinv      privinv      totginv      totpinv

  1     4567.65     4571.43       50.00         0.00        50.00         0.00
  2     1855.77    15517.99       50.00         0.00       100.00         0.00
  3      809.24    47855.23       60.00         0.01       160.00         0.01
  4       17.35  8798352.25       60.00      7021.26       220.00      7021.27
  5        7.03 29975536.00       70.00       297.37       290.00      7318.64

READY (Cost: $0.52)
delete r fastdecat
The run fastdecat is not in the file.
Reenter the name (or reply "*" to cancel) : fastdecay

The run fastdecay has been deleted.
READY (Cost: $0.43)
: m

All models

Name                            Used

Aagr                            yes
Acr                             yes
Arr                             yes
agr                             yes
demonstration                   yes
ob                              yes
rr                              yes
showoff                         yes

READY (Cost: $0.20)
m m demonstration

Enter "help" for a dictionary of variables, "quit" to return, or the name of the variable to be modified: n

Old value of n is  3.  Enter the new value: 4

Enter the name of sector  4: farmers

Enter the  5 government investment values (in M$) for this sector: 10 10 10 10 10

Next request: tmax

Old value of tmax is  5.  Enter the new value: 6

Enter the  1 new government investment values (in M$) for times  6 to  6 for sector...
farmers: 60

houses: 10

centralpower: 0
```

We mistyped the name, but are given a chance to redeem ourselves

Changing the cost decay rate from .7 to .6 has caused a 300-fold increase in Kwatts purchased

Used has changed to "yes" for "demonstration".

this is perfectly legal and has the effect of parameterizing the new fourth sector the same as the first sector.

```
farmers: 10

Next request: qdisplay

Model demonstration:

Cost per Kwatt of PV at time 0                                    14000.00
Decimal fraction by which cost declines when production doubles       0.60
Kwatts of PV installed at time 0 across sectors                   1000.00
The model will run until time =  6, and has  4 sectors.

Sector                    Government investment (in M$) at time
Name                        1        2        3        4        5        6

farmers                   40.00    40.00    50.00    50.00    60.00    60.00
houses                    10.00    10.00    10.00    10.00    10.00    10.00
centralpower               0.00     0.00     0.00     0.00     0.00     0.00
farmers                   10.00    10.00    10.00    10.00    10.00    10.00

READY (Cost: $0.60)
m m demonstration

Enter "help" for a dictionary of variables, "quit" to return, or the name of the variable to be modified: ppurch

There is no variable named ppurch.  Try again.
Next request: costdecay

Old value of costdecay is      0.60.  Enter the new value: .7

Next request: q

READY (Cost: $0.41)
r demonstration 4X6

The model demonstration has been run under the name 4X6.
Do you want to display this run? : yes

Do you want to see model, sector, both or neither input variables? : n

Same choice for output variables: m

time      cost       totwatt      govinv      privinv      totginv      totpinv

  1      5943.45      5285.71      60.00         0.00        60.00         0.00
  2      3430.36     15380.87      60.00         0.00       120.00         0.00
  3      2221.38     35786.90      70.00         0.00       190.00         0.00
  4      1605.04     67298.80      70.00         0.00       260.00         0.00
  5      1192.52    119877.39      80.00         4.39       340.00         4.39
  6       335.20   1412037.83      80.00      1460.93       420.00      1465.32

READY (Cost: $0.51)
l

All records

Name                      Type    Used?

Aagr                      model   yes
Acr                       model   yes
Arr                       mod
```

We have effectively turned a 3X5 government investment matrix into a 4X6 one.

The fourth sector with its government investment has caused a significant increase in purchased Kwatts in period 5.

We realize that we have subjected ourselves to a very long printout here, so we pushed the QUIT button to escape it.

```
QUIT

READY (Cost: $0.06)
i r

All runs

Name                            Used

4X6                             yes
highfarmpurch                   yes
output                          yes

READY (Cost: $0.15)
cr s fauxpas

Enter the following for sector fauxpas:
Average Kwatts per Installation:
(sector fauxpas has not been created)
QUIT

READY (Cost: $0.58)
delete r 4x6 highfarmpurch output

The run 4x6 is not in the file.
Reenter the name (or reply "*" to cancel) : 4X6

The run 4X6 has been deleted.
The run highfarmpurch has been deleted.
The run output has been deleted.
READY (Cost: $0.51)
i r

All runs

Name                            Used

There are currently no runs in the file

READY (Cost: $0.19)
return

You will now be disconnected from the computer.  Goodbye.
```

... and we get back to command level

once again, we don't really want to create this sector, so we push QUIT instead of answering the question.

and we are reminded that any partial data entry will be lost.

We accidently type small "x" instead of capital "X".

All our runs are deleted so they aren't taking up space and costing storage charges.

We are done for now, so we "return."

And get this sign-off message. We have been automatically logged off the computer.

A.5 Program Listing

```
algor1 proc(rptr);

%include run;
dcl (a,b,i,t) fixed;
dcl isqrt2 float init(1/sqrt(2));

totgovinv(*,0)=0;
totprivinv(*,0)=0;
totgovinst(*,0)=0;
totprivinst(*,0)=0;
potinst(*,0)=initpotinst(*)+newinst(*);
govinvnow(*)=0;
privinvnow(*)=0;
allgovinv(0)=0;
allprivinv(0)=0;
cost(0)=initcost;
totwatt(0)=initwatt;

do t=1 to tmax;

  totwatt(t)=totwatt(t-1);
  do i=1 to n;

    govinv(i,t)=max(0,govinv(i,t));
    costaccep(i,t)=.5*(1+erf(-isqrt2*(cost(t-1)-cmean(i))/cdev(i)));
    instaccep(i,t)=.5*(1+erf(isqrt2*(totinst(i,t-1)-imean(i))/idev(i)));
    potinst(i,t)=(potinst(i,t-1)-newinst(i))*(1+buildrate(i))+newinst(i);
    privinv(i,t)=max(0,(potinst(i,t)-totinst(i,t-1))*costaccep(i,t)*instaccep(i,t)*ppurch(i)*avgsize(i)*cost(t-1));
    govinst(i,t)=govinv(i,t)/(avgsize(i)*cost(t-1));
    privinst(i,t)=privinv(i,t)/(avgsize(i)*cost(t-1));
    totgovinv(i,t)=totgovinv(i,t-1)+govinv(i,t);
    totprivinv(i,t)=totprivinv(i,t-1)+privinv(i,t);
    totgovinst(i,t)=totgovinst(i,t-1)+govinst(i,t);
    totprivinst(i,t)=totprivinst(i,t-1)+privinst(i,t);
    totinst(i,t)=totgovinst(i,t)+totprivinst(i,t);
    totinv(i,t)=totgovinv(i,t)+totprivinv(i,t);
    govinvnow(t)=govinvnow(t)+govinv(i,t);
    privinvnow(t)=privinvnow(t)+privinv(i,t);
    totwatt(t)=totwatt(t)+(govinv(i,t)+privinv(i,t))/cost(t-1);
    end;

  cost(t)=initcost*(totwatt(t)/initwatt)**log2(costdecay);
  allgovinv(t)=allgovinv(t-1)+govinvnow(t);
  allprivinv(t)=allprivinv(t-1)+privinvnow(t);
  end;

end;
```

```
create:cr:  pc;

dcl 01 model based(mptr),
       02 mname char(30) varying,
       02 n fixed,
       02 tmax fixed,
       02 initcost float,
       02 costdecay float,
       02 initwatt float,
       02 sec(1:ns refer (n)),
          03 id char(30) varying,
          03 govinv(tm refer (tmax)) float;
dcl getarg entry(fixed) returns(char(30) varying aligned);
dcl (key,any_other,quit,conversion) condition;
dcl (oncode,onloc) builtin;
dcl 01 sector controlled,
       02 sid char(30) varying,
       02 avgsize float,
       02 initpotinst float,
       02 buildrate float,
       02 newinst float,
       02 ppurch float,
       02 costp,
          03 cmean float,
          03 cdev float,
       02 instp,
          03 imean float,
          03 idev float;
dcl 01 index,
       02 iname char(31),
       02 type char(6),
       02 used char(3);
dcl (sysin,sysprint,(secfile,modfile,indfile) keyed) file;
dcl (idx,i,ns,tm) fixed;
dcl mptr ptr;
dcl aname char(30) varying aligned;
dcl rtype char(1);
dcl tnam char(30) varying aligned;
dcl nargs fixed external static;
dcl kflag bit(1);
dcl onsource builtin;

  on conversion begin;
    put skip list("""||onsource||""" is an incorrect entry.  Create aborted.");
    delete file(indfile) key(iname);
    go to back;
    end;
  on key(indfile) kflag="1"b;
  aname=getarg(2);
  do while(aname="");
    put skip list("Enter ""model"" or ""sector"" : ");
    get list(aname);
    end;
  rtype=substr(aname,1,1);
  do while(^(rtype="m"|rtype="s"));
    put skip list("First argument must be model or sector.  Reenter : ");
    get list(aname);
    rtype=substr(aname,1,1);
    end;
  if rtype="m" then do;
```

```
   aname="   el";
   used="n  ";
   open file(modfile) keyed direct update;
   end;
 else do;
   aname="sector";
   used="n/a";
   open file(secfile) keyed direct update;
   end;
 type=aname;

 on cond(quit) go to error;
 on cond(any_other) begin;
   put skip edit("Create error - onloc is "||onloc||
     ", oncode is ",oncode)(a,f(12));
   go to error;
   end;
 open file(indfile) keyed direct update;
 do idx=3 to max(3,nargs);
   fnam=getarg(idx);
   do while(fnam="");
     put skip list("Enter "||aname||" name : ");
     get list(fnam);
     end;
   iname=rtype||fnam;
   kflag="0"b;
   write file(indfile) keyfrom(iname) from(index);
   do while(kflag);
     put skip list("There is already a "||aname||" called "||fnam||".  Reenter (or enter ""*"" to cancel): ");
     get list(fnam);
     if fnam="*" then go to back;
     kflag="0"b;
     iname=rtype||fnam;
     write file(indfile) keyfrom(iname) from(index);
     end;
   put skip list("Enter the following for "||aname||" "||fnam||":");
   if rtype="s" then do;
     allocate sector;
     put skip list("Average Kwatts per installation: ");
     get list(avgsize);
     put skip list("Initial potential old installations: ");
     get list(initpotinst);
     put skip list("Decimal rate at which potential installations grow: ");
     get list(buildrate);
     put skip list("New installations per period: ");
     get list(newinst);
     put skip list("Decimal purchase probability given acceptance: ");
     get list(ppurch);
     put skip list("Cost per Kwatt acceptance mean and std dev: ");
     get list(costp);
     put skip list("Sucessful installations acceptance mean and std dev: ");
     get list(instp);
     sid=fnam;
     write file(secfile) keyfrom(sid) from(sector);
     free sector;
     end;
   else do;
     put skip list("Number of sectors: ");
     get list(ns);
     put skip list("Number of time periods: ");
```

```
      get l  t(tm);
      alloc  e model;
      put skip list("Initial dollar cost per Kwatt: ");
      get list(initcost);
      put skip list("Decimal factor by which cost declines when production doubles: ");
      get list(costdecay);
      put skip list("Initial number of Kwatts installed across all sectors: ");
      get list(initwatt);
      put edit("The ",ns," sector names: ")(skip,a,f(2),a);
      get list(sec(*).id);
      put edit("The ",tm," government investment values (in M$) for sector")(skip,a,f(3),a);
      put skip;
      do i=1 to ns;
        put edit(sec(i).id," : ")(skip,a,a);
        get list(sec(i).govinv(*));
        govinv(i,*)=govinv(i,*)*1e6;
        end;
      mname=fnam;
      write file(modfile) keyfrom(mname) from(model);
      free model;
      end;
    put skip list("The "||aname||" "||fnam||" has been created.");
back: end;
close file(indfile),file(modfile),file(secfile);
return;
error: revert cond(quit),cond(any_other);
put skip list("("||aname||" "||fnam||" has not been created) ");
on key(indfile);
delete file(indfile) key(iname);
if rtype="m" then do;
  on key(modfile);
  delete file(modfile) key(mname);
  end;
else do;
  on key(secfile);
  delete file(secfile) key(sid);
  end;
signal cond(quit);
end;
```

```
close1 pro
dcl (indfile,modfile,secfile,runfile,term) file;
close file(indfile),file(modfile),file(secfile),file(runfile),file(term);
end;
```

```
copytcp: proc;

dcl 01 sector based(mptr),
       02 sname char(30) varying,
       02 avgsize float,
       02 initpotinst float,
       02 buildrate float,
       02 newinst float,
       02 ppurch float,
       02 costp,
          03 cmean float,
          03 cdev float,
       02 instp,
          03 imean float,
          03 idev float;
dcl 01 model based(mptr),
       02 mname char(30) varying,
       02 n fixed,
       02 tmax fixed,
       02 initcost float,
       02 costdecay float,
       02 initwatt float,
       02 sections refer (n)),
          03 id char(30) varying,
          03 govinv(tm refer(tmax)) float;
dcl mptr ptr;
dcl 01 index,
       02 iname char(31),
       02 type char(6),
       02 used char(3);
dcl (oldkey,newkey,aname) char(30) varying;
dcl filet file variable;
dcl rtype char(1);
dcl (modfile,secfile,indfile,sysin,sysprint) file;
dcl (key,quit,any_other) condition;
dcl (oncode,onloc) builtin;
dcl getarg entry(fixed) returns(char(30) varying aligned);
dcl mproc variable entry(ptr);
dcl (mmod,msec) entry(ptr);

aname=getarg(2);
if aname="" then do;
  put skip list("Is a model or a sector to be copied? : ");
  get list(aname);
  end;
rtype=substr(aname,1,1);
do while(^(rtype="m"|rtype="s"));
  put skip list("First argument must be model or sector.  Reenter : ");
  get list(aname);
  rtype=substr(aname,1,1);
  end;
if rtype="m" then do;
  aname="model";
  filet=modfile;
  used="no";
  mproc=mmod;
  end;
else do;
  aname="sector";
  filet=secfile;
```

```
  used=";
  mproc=msec;
  end;
oldkey=getarg(3);
if oldkey="" then do;
  put skip list("Enter the name of the "||aname||" to be copied : ");
  get list(oldkey);
  end;

on cond(quit) go to error;
on cond(any_other) begin;
  put skip edit("Copy error - onloc is "||onloc||", oncode is ",oncode)(a,f(12));
  go to error;
  end;
open file(filef) keyed direct update;
on key(filef) begin;
  put skip list("The "||aname||" "||oldkey||" is not in the file.");
  put skip list("Reenter the name (or reply ""*"" to cancel) : ");
  get list(oldkey);
  if oldkey="*" then go to back;
  go to retin;
  end;
retin: read file(filef) key(oldkey) set(mptr);
newkey=getarg(4);
if newkey="" then do;
  put skip list("Enter the name for the copy of "||aname||" "||oldkey||" : ");
  get list(newkey);
  end;

open file(indfile) keyed direct update;
on key(indfile) begin;
  put skip list("There is already a "||aname||" called "||newkey||" in the file.");
  put skip list("Reenter the name (or reply ""*"" to cancel) : ");
  get list(newkey);
  if newkey="*" then go to back;
  go to retry;
  end;
type=aname;
retry: lname=rtype||newkey;
write file(indfile) from(index) keyfrom(lname);
if rtype="m" then do;
  mname=newkey;
  write file(modfile) from(model) keyfrom(mname);
  end;
else do;
  sname=newkey;
  write file(secfile) from(sector) keyfrom(sname);
  end;
put skip list("The "||aname||" "||oldkey||" has been copied under the name "||newkey||".");
revert cond(quit),cond(any_other);
put skip list("Do you want to modify "||aname||" "||newkey||"? : ");
get list(oldkey);
if oldkey="yes" then call mproc(mptr);
back: close file(filef), file(indfile);
return;

error: revert cond(quit),cond(any_other);
put skip list("("||aname||" "||newkey||" has not been saved)");
on key(indfile);
delete file(indfile) key(lname);
```

```
on key(fil    );
delete file(filef) key(newkey);
signal cond(quit);
end;
```

```
delete: proc

dcl (badkey,aname) char(30) varying;
dcl getarg entry(fixed) returns(char(30) varying aligned);
dcl rtype char(1);
dcl (modfile,secfile,runfile,indfile,sysin,sysprint) file;
dcl filet file variable;
dcl verb char(9) varying;
dcl i fixed;
dcl nargs fixed external static;
dcl key condition;
dcl 01 index,
       02 iname char(31),
       02 type char(6),
       02 used char(3);

on key(indfile) begin;
  put skip list("The "||aname||" "||badkey||" is not in the file.");
  put skip list("Reenter the name (or reply ""*"" to cancel) : ");
  get list(badkey);
  if badkey="*" then go to back;
  go to retry;
  end;

aname=getarg(2);
if aname="" then do;
  put skip list("Is a model, sector or run to be deleted? : ");
  get list(aname);
  end;
rtype=substr(aname,1,1);
do while(^(rtype="m"|rtype="s"|rtype="r"));
  put skip list("First argument must be model/sector/run.  Reenter : ");
  get list(aname);
  rtype=substr(aname,1,1);
  end;
if rtype="m" then do;
  verb="run";
  aname="model";
  filet=modfile;
  end;
else if rtype="s" then do;
  aname="sector";
  filet=secfile;
  end;
else do;
  verb="displayed";
  aname="run";
  filet=runfile;
  end;

open file(filet) keyed direct update;
open file(indfile) keyed direct update;
do i=3 to max(3,nargs);
  badkey=getarg(i);
  if badkey="" then do;
    put skip list("Enter the name of the "||aname||" to be deleted : ");
    get list(badkey);
    end;
retry: if rtype^="s" then do;
    read file(indfile) into(index) key(rtype||badkey);
```

```
    if used="r   then do;
      put skip list("The "||aname||" "||badkey||" has not been "||ver..||".  Do you still want to delete? : ");
      get list(type);
      if type^="yes" then do;
        go to back;
        end;
      end;
    end;
  delete file(indfile) key(rtype||badkey);
  delete file(filef) key(badkey);
  put skip list("The "||aname||" "||badkey||" has been deleted.");
back: end;
close file(indfile),file(filef);
end;
```

```
drun1  proc  tr);

%include run;
dcl (i,t,a,b) fixed;
dcl (sysin,sysprint,indfile) file;
dcl 01 index,
      02 iname char(31),
      02 type char(6),
      02 used char(3);
dcl (anso,ansi) char(10);
dcl (dmod,dsec) entry(ptr);

open file(indfile) keyed direct update;
read file(indfile) key("r"||rname) into(index);
used="yes";
rewrite file(indfile) key(iname) from(index);
close file(indfile);
put skip list("Do you want to see model, sector, both or neither input variables? : ");
get list(ansi);
ansi=substr(ansi,1,1);
put skip list("Same choice for output variables: ");
get list(anso);
anso=substr(anso,1,1);
if ansi="m"|ansi="b" then call dmod(addr(model));
if anso="m"|anso="b" then do;
  put skip list("time      cost      totwatt      govinv      privinv      totginv      totpinv    ");
  put skip edit((t,cost(t),totwatt(t),govinvnow(t)*1e-6,privinvnow(t)*1e-6,allgovinv(t)*1e-6,allprivinv(t)*1e-6
    do t=1 to tmax))(skip,f(3),(6)f(12,2));
  put skip;
  end;
do i=1 to n;
  if ansi="s"|ansi="b" then call dsec(addr(sector(i)));
  if anso="s"|anso="b" then do;
    if ansi^="s"&ansi^="b" then do;
      put skip list("Sector "||sname(i)||":");
      put skip;
      end;
    put skip edit("time   govinv   privinv   govinst  privinst   totginv   totpinv    totinv  totginst  totpinst   totinst ")(a);
    put edit(" potinst costacc instacc")(a);
    put skip edit((t,govinv(i,t)*1e-6,privinv(i,t)*1e-6,govinst(i,t),privinst(i,t),totgovinv(i,t)*1e-6,totprivinv(i,t)*1e-6,
      totinv(i,t)*1e-6,totgovinst(i,t),totprivinst(i,t),totinst(i,t),potinst(i,t),costacce(i,t),instacce(i,t)
      do t=1 to tmax))(skip,f(3),(11)f(10,2),(2)f(7,4));
    put skip;
    end;
  end;
end;
```

```
dsect: proc(sptr);

dcl 01 sector based(sptr),
       02 sname char(30) varying,
       02 avgsize float,
       02 initpotinst float,
       02 buildrate float,
       02 newinst float,
       02 ppurch float,
       02 costp,
          03 cmean float,
          03 cdev float,
       02 instp,
          03 imean float,
          03 idev float;
dcl sptr ptr;
dcl sysprint file;

put skip list("Sector "||sname||":");
put skip;
put skip edit("Average installation size (Kwatts)",avgsize)(a,col(67),f(11,2));
put skip edit("Initial potential old installations",initpotinst)(a,col(67),f(10));
put skip edit("Decimal growth rate of potential installations",buildrate)(a,col(67),f(11,2));
put skip edit("Number of new installations per period",newinst)(a,col(67),f(11,2));
put skip edit("Decimal probability of purchase given acceptability",ppurch)(a,col(67),f(11,2));
put skip edit("Feasible cost per Kwatt mean and std dev",costp)(a,col(67),2(f(11,2),x(2)));
put skip edit("Feasible number of successes mean and std dev",instp)(a,col(67),2(f(11,2),x(2)));
put skip;
return;

dmod: entry(mptr);

dcl (i,t) fixed;
dcl 01 model based(mptr),
       02 mname char(30) varying,
       02 n fixed,
       02 tmax fixed,
       02 initcost float,
       02 costdecay float,
       02 initwatt float,
       02 sec(1:n refer (n)),
          03 id char(30) varying,
          03 govinv(1:tmax refer(tmax)) float;
dcl mptr ptr;

put skip list("Model "||mname||":");
put skip;
put skip edit("Cost per Kwatt of PV at time 0",initcost)(a,col(67),f(11,2));
put skip edit("Decimal fraction by which cost declines when production doubles",costdecay)(a,col(67),f(11,2));
put skip edit("Kwatts of PV installed at time 0 across sectors",initwatt)(a,col(67),f(11,2));
put skip edit("The model will run until time =",tmax,", and has",n," sectors.")(a,f(3),a,f(3),a);
put skip(2) edit("Sector","Government investment (in M$) at time","Name",(t do t=1 to tmax),
                 (id(i),(govinv(i,t)*1e-6 do t=1 to tmax) do i=1 to n))
                 (a,col(31),a,skip,a,((floor(tmax/10))(col(35),10(f(3),x(6)),skip),
                 (min(1,mod(tmax,10)))(col(35),(mod(tmax,10))(f(3),x(6)),skip),skip,
                 (n)(a,((floor(tmax/10))(col(31),(10)f(9,2),skip),
                     (min(1,mod(tmax,10)))(col(31),(mod(tmax,10))f(9,2),skip)));
put skip;
return;
end;
```

```
indexi is e.oc;
dcl 01 index,
     02 lname char(31),
     02 type char(6),
     02 used char(3);
dcl (sysin,sysprint,indfile) file;
dcl getarg entry(fixed) returns(char(30) varying aligned);
dcl rtype char(1);
dcl arg char(30) varying aligned;
dcl flag bit(1) init("0"b);
dcl dum char(31);
dcl endfile cond;

arg=getarg(2);
if arg = "" then rtype="a";
  else rtype=substr(arg,1,1);
do while( ^(rtype="a"|rtype="m"|rtype="s"|rtype="r"));
  put skip list("First argument must be model/sector/run/all.  Reenter : ");
  get list (arg);
  rtype=substr(arg,1,1);
  end;
open file(indfile) keyed sequential input;
on endfile(indfile) go to out;
if rtype="a" then do;
  arg="record";
  put skip list("All records");
  put skip(2) edit("Name","Type   Used?")(a,col(31),a);
  put skip;
  do while("1"b);
    read file(indfile) into(index) keyto(dum);
    put skip edit(substr(lname,2,30),type,used)(a,col(31),a,col(38),a);
    flag="1"b;
   end;
  end;
else do;
  if rtype="m" then arg="model";
  if rtype="s" then arg="sector";
  if rtype="r" then arg="run";
  put skip list("All "||arg||"s");
  put skip(2) edit("Name","Used")(a,col(31),a);
  put skip;
  do while("1"b);
    read file(indfile) into(index) ke,to(dum);
    if type=arg then do;
      put skip edit(substr(lname,2,30),used)(a,col(31),a);
      flag="1"b;
      end;
    end;
  end;
out: if ^flag then put skip list("There are currently no "||arg||"s in the file");
put skip;
close file(indfile);
end;
```

```
intbuf: pr

dcl buf(140) static char(1);
dcl tuf char(140) defined(buf);
dcl nargs fixed static external;
dcl code fixed bin(35);
dcl iptr static ptr;
dcl tab(30) fixed static;
dcl iox_$find_iocb entry(char(*),ptr,fixed bin(35));

nargs=0;
call iox_$find_iocb("user_input",iptr,code);
return;

newline: entry;
dcl iox_$get_line entry(ptr,ptr,fixed bin(21),fixed bin(21),fixed bin(35));
dcl j fixed init(0);
dcl sw bit(1) init("0"b);
dcl sysprint file;
dcl i fixed;
dcl ibuf fixed bin(21);
  ibuf=1;
  do while(ibuf=1);
    call iox_$get_line(iptr,addr(buf),140,ibuf,code);
    end;
  buf(ibuf)=" ";
  do i=1 to ibuf;
    if bool((buf(i)^=" "&buf(i)^="	"),sw,"0110"b) then do;
      j=j+1;
      if j>30 then do;
        i=ibuf;
        j=30;
        put list("Only the first 15 fields have been recognized - the rest have been ignored.");
        put skip;
        end;
      else do;
        sw=^sw;
        tab(j)=i;
        end;
      end;
    end;
  nargs=j/2;
  return;

getarg: entry(argind) returns(char(30) varying aligned);
dcl argind fixed;
if argind>0 & argind<=nargs
  then return (substr(tuf,tab(2*argind-1),min(30,tab(2*argind)-tab(2*argind-1))));
  else return ("");

afarg: entry(num);
dcl  num char(*);
dcl cu_$af_return_arg entry(fixed bin(17),ptr,fixed bin(17),fixed bin(35));
dcl sreturn char(ls) varying based(sptr);
dcl  sptr  ptr;
dcl  ls  fixed bin(17);
call cu_$af_return_arg(i,sptr,ls,code);
sreturn=getarg(fixed(num));
end;
```

```
intern proc(resp);

dcl resp char(*);
dcl helpline char(39) init("lo put_chars user_output -sm start.help");
dcl cu_$cp entry(ptr,fixed,fixed bin(35));
dcl close entry();
dcl exline char(6) init("ec ex");
dcl (quit,any_other) condition;
dcl code fixed bin(35);
dcl (oncode,onloc) builtin;

on cond(quit) begin;
  call close();
  put skip list("QUIT");
  put skip;
  go to back;
  end;
on cond(any_other) begin;
  put skip edit("Error - onloc is "||onloc||", oncode is ",oncode)(a,f(12));
  signal cond(quit);
  end;

if resp^="no" then call cu_$cp(addr(helpline),39,code);
back: call cu_$cp(addr(exline),6,code);
end;
```

```
mmod: pr   (mptr);

dcl 01 model based(mptr),
       02 mname char(30) varying,
       02 n fixed,
       02 tmax fixed,
       02 var(3:5) float,
       02 sec(ns refer(n)),
         03 sid char(30) varying,
         03 govinv(tm refer(tmax)) float;
dcl (mptr,newp) ptr;
dcl (modfile,indfile,sysin,sysprint,term) file;
dcl (i,j,k,new,ns,tm) fixed;
dcl rep char(30) varying;
dcl (conversion,key) condition;
dcl onsource builtin;
dcl ans char(10);
dcl 01 indx,
       02 iname char(31),
       02 type char(6),
       02 used  char(3);
dcl mtab(5) char(9) init("n","tmax","initcost","costdecay","initwatt");
dcl dmod entry(ptr);
dcl cu_$cp entry(ptr,fixed,fixed bin(35));
dcl code fixed bin(35);
dcl line char(50) init("lo put_chars user_output -sm mod.help");
dcl flag bit(1) init("1"b);
dcl bad char(30) varying;

on conversion begin;
  if onsource="" then bad=rep;
                 else bad=onsource;
  put skip list(""""||bad||""" is an invalid entry.  Request ignored.");
  go to endloop;
  end;
on key(indfile) begin;
  put skip list("There is already a model in the file called "||rep||".");
  put skip list("Do you want to overwrite/delete it(d), enter a new name(n), or cancel(*)? : ");
  get list(ans);
  if ans="*" then do;
    go to endloop;
    end;
  else if ans="d" then do;
    delete file(indfile) key(iname);
    delete file(modfile) key(rep);
    go to retry;
    end;
  else do;
    put skip list("Enter the new name: ");
    read file(term) into(rep);
    go to retry;
    end;
  end;

open file(term) environment(stringvalue) record input title("record_stream_ user_input");
put skip list("Enter ""help"" for a dictionary of variables, ""quit"" to return, or the name of the variable to be modified: ");
do while(flag);
  read file(term) into(rep);
  if rep="help" then call cu_$cp(addr(line),length(line),code);
  else if substr(rep,1,1)="q" then do;
```

```
    flag= 0;
    if substr(rep,2)="display" then call dmod(mptr);
    end;
  else if rep="display" then call dmod(mptr);
  else if rep="name" then do;
    put skip list("The old name is "||mname||".  Enter the new name: ");
    read file(term) into(rep);
    open file(indfile) keyed direct update;
    read file(indfile) key("m"||mname) into(indx);
retry: iname="m"||rep;
    write file(indfile) keyfrom(iname) from(indx);
    delete file(indfile) key("m"||mname);
    close file(indfile);
    delete file(modfile) key(mname);
    mname=rep;
    write file(modfile) keyfrom(mname) from(model);
    end;
  else do;
    do i=1 to 5 while(mtab(i)^=rep);
      end;
    if i=6 then do;
      if substr(rep,1,min(length(rep),6))="govinv" then do;
        if length(rep)<11|substr(rep,7,1)^="("|substr(rep,length(rep))^=")" then signal conversion;
        k=index(substr(rep,9),",");
        if k=0 then signal conversion;
        i=decimal(substr(rep,8,k));
        j=decimal(substr(rep,9+k,length(rep)-9-k));
        if i<1|i>n|j<1|j>tmax then put skip list("Subscripts out of range.  Request ignored.");
        else do;
          put skip edit("Old value of "||rep||" is",govinv(i,j)*1e-6,".  Enter new value: ")(a,f(9,2),a);
          get list(govinv(i,j));
          govinv(i,j)=govinv(i,j)*1e6;
          end;
        end;
      else if substr(rep,1,min(length(rep),7))="secname" then do;
        if length(rep)<10|substr(rep,8,1)^="("|substr(rep,length(rep))^=")" then signal conversion;
        i=decimal(substr(rep,9,length(rep)-9));
        if i<1|i>n then put skip list("Subscript out of range.  Request ignored.");
        else do;
          put skip list("Old value of "||rep||" is "||sid(i)||".  Enter new value: ");
          get list(sid(i));
          end;
        end;
      else put skip list("There is no variable named "||rep||".  Try again.");
      end;
    else do;
      if i<3 then do;
        if i=1 then new=n;
               else new=tmax;
        put skip edit("Old value of "||rep||" is",new,".  Enter the new value: ")(a,f(3),a);
        get list(new);
        if i=1 then do; ns=new; tm=tmax; end;
               else do; tm=new; ns=n; end;
        allocate model set(newp);
        newp->var(*)=var(*);
        newp->mname=mname;
        do i=1 to min(n,ns);
          newp->sid(i)=sid(i);
          do j=1 to min(tmax,tm);
            newp->govinv(i,j)=govinv(i,j);
```

```
            end;
          end;
        if ns>n then do i=n+1 to ns;
          put skip edit("Enter the name of sector",i,": ")(a,f(3),a);
          get list(newp->sid(i));
          put skip edit("Enter the",tmax," government investment values (in M$) for this sector: ")(a,f(3),a);
          get list((newp->govinv(i,j) do j=1 to tmax));
          newp->govinv(i,*)=newp->govinv(i,*)*1e6;
          end;
        if tm>tmax then do;
          put skip edit("Enter the",tm-tmax," new government investment values (in M$) for times",tmax+1," to",tm," for sector...")
            (a,f(3),a,f(3),a,f(3),a);
          do i=1 to n;
            put skip list(sid(i)||": ");
            get list((newp->govinv(i,j) do j=tmax+1 to tm));
            do j=tmax+1 to tm;
              newp->govinv(i,j)=newp->govinv(i,j)*1e6;
              end;
            end;
          end;
        free model;
        mptr=newp;
        end;
      else do;
        put skip edit("Old value of "||rep||" is",var(i),".  Enter the new value: ")(a,f(9,2),a);
        get list(var(i));
        end;
      end;
    end;
endloop: if flag then put skip list("Next request: ");
  end;
rewrite file(modfile) key(mname) from(model);
close file(term);
end;
```

```
modify: m. display: dp: proc;

dcl filet file variable;
dcl proct(5) entry(ptr) variable init(dmod,mmod,dsec,msec,drun);
dcl (sysin,sysprint,modfile,secfile,runfile) file;
dcl (dmod,dsec,drun,mmod,msec) entry(ptr);
dcl (aname,tnam) char(30) varying;
dcl pname char(9) varying;
dcl qname char(5) varying;
dcl rtype char(1);
dcl rep char(10) init("yes");
dcl nargs fixed external static;
dcl sptr ptr;
dcl getarg entry(fixed) returns(char(30) varying aligned);
dcl (key,quit,any_other) condition;
dcl (onloc,oncode) builtin;
dcl (i,ip) fixed;

if substr(getarg(1),1,1)="d" then do;
  qname="run, ";
  pname="displayed";
  ip=0;
  end;
else do;
  qname="";
  pname="modified";
  ip=1;
  end;
aname=getarg(2);
if aname="" then do;
  put skip list("Is a "||qname||"model or sector to be "||pname||"? :");
  get list(aname);
  end;
rtype=substr(aname,1,1);
do while(^(rtype="m"|rtype="s"|(rtype="r"&ip=0)));
  put skip list("First argument must be "||qname||"model or sector.  Reenter: ");
  get list(aname);
  rtype=substr(aname,1,1);
  end;
if rtype="r" then do;
  aname="run";
  filet=runfile;
  ip=5;
  end;
else if rtype="m" then do;
  aname="model";
  filet=modfile;
  ip=ip+1;
  end;
else do;
  aname="sector";
  filet=secfile;
  ip=ip+3;
  end;
on key(filet) begin;
  put skip list("There is no "||aname||" named "||tnam||" in the file.");
  put skip list("Reenter the name (or reply ""*"" to cancel) : ");
  get list(tnam);
  if tnam="*" then go to back;
  go to retry;
```

```
  end;

on cond(quit) go to error;
on cond(any_other) begin;
  put skip edit("Mod/disp error - onloc is "||onloc||
    ", oncode is ",oncode)(a,f(12));
  go to error;
  end;
open file(filef) keyed direct update;
do i=3 to max(3,nargs);
  fnam=getarg(i);
  if fnam="" then do;
    put skip list("Enter the name of the "||aname||" to be "||pname||": ");
    get list(fnam);
    end;
retry: read file(filef) key(fnam) set(sptr);
  call proc(ip)(sptr);
  end;
back: close file(filef);
return;

error: revert cond(quit),cond(any_other);
if qname="" then put skip list("(modifications may not have been saved)");
signal cond(quit);
end;
```

```
msect proc(s_,f);

dcl 01 sector based(sptr),
       02 sname char(30) varying,
       02 var(9) float;
dcl stab(9) char(9) init("avgsize","initpotin","buildrate","newinst","ppurch","costmean","costdev","instmean","instdev");
dcl sptr ptr;
dcl rep char(30) varying;
dcl ans char(10);
dcl i fixed;
dcl (key,conversion) condition;
dcl (indfile,secfile,sysin,sysprint) file;
dcl 01 index,
       02 iname char(31),
       02 type char(6),
       02 used char(3);
dcl onsource builtin;
dcl dsec entry(ptr);
dcl cu_$cp entry(ptr,fixed,fixed bin(35));
dcl code fixed bin(35);
dcl flag bit(1) init("1"b);
dcl line char(50) init("io put_chars user_output -sm sec.help");

on conversion begin;
  put skip list(""""||onsource||""" is an invalid numeric entry.  Request ignored.");
  go to endloop;
  end;
on key(indfile) begin;
  put skip list("There is already a sector in the file called "||rep||".");
  put skip list("Do you want to overwrite/delete it(d), enter a new name(n), or cancel(*)? : ");
  get list(ans);
  if ans="*" then do;
    go to endloop;
    end;
  else if ans="d" then do;
    delete file(indfile) key(iname);
    delete file(secfile) key(rep);
    go to retry;
    end;
  else do;
    put skip list("Enter the new name: ");
    get list(rep);
    go to retry;
    end;
  end;

put skip list("Enter ""help"" for a dictionary of variables, ""quit"" to return, or the name of the variable to be modified: ");
do while(flag);
  get list(rep);
  if rep="help" then call cu_$cp(addr(line),length(line),code);
  else if substr(rep,1,1)="q" then do;
    flag="0"b;
    if substr(rep,2)="display" then call dsec(sptr);
    end;
  else if rep="display" then call dsec(sptr);
  else if rep="name" then do;
    open file(indfile) keyed direct update;
    read file(indfile) key("s"||sname) into(index);
    put skip list("The old name is "||sname||".  Enter the new name: ");
    get list(rep);
```

```
retry: l   le="s"||rep;
    write file(indfile) keyfrom(lname) from(index);
    delete file(indfile) key("s"||sname);
    close file(indfile);
    delete file(secfile) key(sname)
    sname=rep;
    write file(secfile) keyfrom(sname) from(sector);
    end;
  else do;
    do i=1 to 11 while(stab(i)^=rep);
      end;
    if i<12 then do;
      put skip edit("Old value of "||rep||" is",var(i),".  Enter the new value: ")(a,f(9,2),a);
      get list(var(i));
      end;
    else put skip list("There is no variable named "||rep||".  Try  again.");
    end;
endloop: if flag then put skip list("Next request: ");
  end;
rewrite file(secfile) key(sname) from(sector);
end;
```

```
run: r:    )c;
%include run;
dcl 01 mod based(mptr),
     02 mn char(30) varying,
      02 a fixed,
      02 b fixed,
      02 c float,
      02 d float,
      02 e float,
      02 s(ns refer(a)),
        03 sid char(30) varying,
        03 g(tm refer(b)) float;
dcl mptr ptr;
dcl (runfile,modfile,secfile,indfile,sysin,sysprint) file;
dcl getarg entry(fixed) returns(char(30) varying aligned);
dcl aname char(30) varying;
dcl (key,quit,any_other) condition;
dcl rep char(9);
dcl drun entry(ptr);
dcl algor entry(ptr);
dcl (onkey,onloc,oncode) builtin;
dcl (i,t) fixed;
dcl 01 index,
    02 iname char(31),
    02 type char(6),
    02 used char(3);
dcl 01 rindex,
     02 riname char(31),
     02 rtype char(6),
     02 rused char(3);

aname=getarg(2);
if aname="" then do;
  put skip list("Enter name of the model to be run : ");
  get list(aname);
  end;

open file(indfile) keyed direct update;
on key(indfile) begin;
  if onkey="m"||aname then do;
    put skip list("The model "||aname||" is not in the file.");
    put skip list("Reenter the name (or reply ""*"" to cancel) : ");
    get list(aname);
    if aname="*" then go to back;
    go to modread;
    end;
  else do;
    put skip list("There is already a run named "||rname||" in the file.");
    put skip list("Do you wish to overwrite/delete it(d), enter a new name(n), or cancel(*)? : ");
    get list(rep);
    if rep="*" then go to back;
    if rep="n" then do;
      put skip list("Enter the new name : ");
      get list(rname);
      go to retry;
      end;
    read file(indfile) into(rindex) key("r"||rname);
    if rused="yes" then do;
      put skip list("The run "||rname||" has not been fully displayed.");
      put skip list("Do you still wish to delete it? : ");
```

```
      ge list(rep);
      if rep="yes" then go to back;
      end;
    rused="no";
    rewrite file(indfile) from(rindex) key("r"||rname);
    delete file(runfile) key(rname);
    go to start;
    end;
  end;

open file(modfile) keyed direct input;
modread: read file(indfile) into(index) key("m"||aname);
used="yes";
read file(modfile) key(aname) set(mptr);
allocate run;
model=mod;

open file(secfile) keyed direct input;
on key(secfile) begin;
  put skip list("The sector "||onkey||" in model "||aname||" is not in the file.");
  put skip list("Check the model definition.  Run aborted.");
  go to back;
  end;

do i=1 to n;
  read file(secfile) key(sid(i)) into(sector(i));
  if pos(avgsize(i),"Average size")|pos(initpotinst(i),"Pot. inst.")|
    rates(ppurch(i),"Purchase")|rates(buildrate(i),"Pot. installation")|pos(cdev(i),"Cost st dev")|pos(idev(i),"Inst st dev") then go
 back;
  end;
if pos(initcost,"Initial cost")|pos(initwatt,"Initial watts")|rates(costdecay,"Cost decay") then go to back;


rates: proc(rate,descr) returns(bit(1));
dcl rate float;
dcl descr char(19) varying;
if 0<rate&rate<1 then return("0"b);
put skip list(descr||" rate is out of range.  Run aborted.");
return("1"b);
end;

pos: proc(var,descr) returns(bit(1));
dcl var float;
dcl descr char(13) varying;
if var>=0 then return("0"b);
put skip list(descr||" is not positive.  Run aborted.");
return("1"b);
end;

rname=getarg(3);
if rname="" then do;
  put skip list("Enter the run name : ");
  get list(rname);
  end;
rtype="run";
rused="no";

on cond(quit) go to error;
on cond(any_other) begin;
  put skip edit("Run error - onloc is "||onloc||", oncode is ",oncode)(a,f(12));
```

```
  go to error;
  end;

retry: rname="r"||rname;
write file(indfile) from(rindex) keyfrom(rname);
start: rewrite file(indfile) from(index) key(name);

call algor(rptr);

open file(runfile) keyed direct update;
write file(runfile) from(run) keyfrom(rname);
put skip list("The model "||aname||" has been run under the name "||rname||".");
revert cond(quit),cond(any_other);
put skip list("Do you want to display this run? : ");
get list(aname);
if aname="yes" then call drun(rptr);
back: close file(indfile),file(modfile),file(secfile),file(runfile);
return;

error: revert cond(quit),cond(any_other);
put skip list("(model "||aname||" has not been run)");
on key(indfile);
delete file(indfile) key(rname);
on key(runfile);
delete file(runfile) key(rname);
end;
```

```
dcl 01 ru  )ased(rptr),
       02 rname char(30) varying,
       02 model,
          03 mname char(30) varying,
          03 n fixed,
          03 tmax fixed,
          03 initcost float,
          03 costdecay float,
          03 initwatt float,
          03 sec(a refer(n)),
             04 sn char(30) varying,
             04 govinv(b refer(tmax)) float,
       02 sector(a refer(n)),
          03 sname char(30) varying,
          03 avgsize float,
          03 initpotinst float,
          03 buildrate float,
          03 newinst float,
          03 pourch float,
          03 costp,
             04 cmean float,
             04 cdev float,
          03 instp,
             04 imean float,
             04 idev float,
       02 cost(0:b refer(tmax)) float,
       02 totwatt(0:b refer(tmax)) float,
       02 govinvnow(b refer(tmax)) float,
       02 privinvnow(b refer(tmax)) float,
       02 allgovinv(0:b refer(tmax)) float,
       02 allprivinv(0:b refer(tmax)) float,
       02 secout(a refer(n)),
          03 privinv(b refer(tmax)) float,
          03 govinst(b refer(tmax)) float,
          03 privinst(b refer(tmax)) float,
          03 totgovinv(0:b refer(tmax)) float,
          03 totprivinv(0:b refer(tmax)) float,
          03 totinv(b refer(tmax)) float,
          03 totgovinst(0:b refer(tmax)) float,
          03 totprivinst(0:b refer(tmax)) float,
          03 totinst(b refer(tmax)) float,
          03 potinst(0:b refer(tmax)) float,
          03 costaccep(b refer(tmax)) float,
          03 instaccep(b refer(tmax)) float;
dcl rptr ptr;
```

Input and output data reside in one of 3 kinds of data blocks - sector, for sector-specific input data, model, for model-wide input data, and run, for output data (including the inputs which it used). The commands for manipulating these blocks are entered as follows (where TYPE is model, sector or run, and the NAMEs are to be replaced with user-defined names. Alternatively, any or all parameters may be left out - you will be prompted for them):

To create models or sectors:
```
    create
  or       TYPE NAME1 NAME2 ...
    cr
```

To copy models or sectors:
```
    copy
  or     TYPE OLDNAME NEWNAME
    cp
```

To display models, sectors or runs:
```
    display
  or        TYPE NAME1 NAME2 ...
    dp
```

To delete models, sectors or runs:
```
    delete TYPE NAME1 NAME2 ...
```

To modify models or sectors:
```
    modify
  or       TYPE NAME1 NAME2 ...
    m
```

To run a model:
```
    run
  or    MODELNAME RUNNAME
    r
```

To get an index of data on file (all blocks will be listed if TYPE is all or blank):
```
    index
  or      TYPE
    i
```

To reprint these instructions:
```
    help
```

To stop processing and disconnect from the computer:
```
    return
```

Name	Description
name	name of the sector
avgsize	average number of Kwatts in a PV installation
initpotin	number of potential installations at time 0
buildrate	decimal growth rate of potential installations
newinst	number of new installations per time period
pburch	decimal probability of PV purchase given acceptability
costmean	mean of cost feasibility distribution
costdev	std dev of cost feasibility distribution
instmean	mean of sucessful installation distribution
instdev	std dev of sucessful installation distribution
quit	terminates processing, saves updated sector
qdisplay	terminated processing, displays and saves updated sector
display	displays updated sector so far

Name	Description
name	name of the model
n	number of sectors
tmax	number of time periods
initcost	initial cost per Kwatt of PV
costdecay	decimal fraction by which PV cost declines when production doubles
initwatt	total Kwatts of PV installed at time 0 across all sectors
secname(i)	name of the ith sector in the model
govinv(i,j)	government investment in sector i at time j (in M$)
quit	terminates processing, saves updated model
qdisplay	terminates processing, displays and saves updated model
display	displays updated model so far

```
&command_line off
ready_off
intbuf
line_length 130
&if [user brief_bit] &then general_ready -string READY -set
&else general_ready -string READY ^x "(Cost:" -inc_cost ")" -set
&if [equal [user term_type] "2741"] &then set_tty -modes ^tabs
&else set_tty -modes ifecho,crecho,ll130,^tabs
&if [or [not [exists segment msg.data]] [equal [user name] "Gravens"]]
&then &goto nomsg
&else
&print
io put_chars user_output -sm msg.data
&if [equal [response "Do you want this message deleted? : "] "yes"]
&then dl msg.data
&else
&label nomsg
&print
&goto &ec_name
&label start_up
&print Welcome to the photo-voltaics simulation model.
infer [response "Do you need an explanation of how the simulation system works (yes or no) ? :"]
&goto finale
&label np
infer no
&label finale
&if [equal [afarg 1] "return"]
&then logout -bf -hold
&else general_ready -revert
&quit
```