



SPSD/M

Tool User's Guide

This guide describes a number of utilities supplied with but not directly part of SPSPD/M.



Statistics
Canada

Statistique
Canada

Canada

Table of Contents

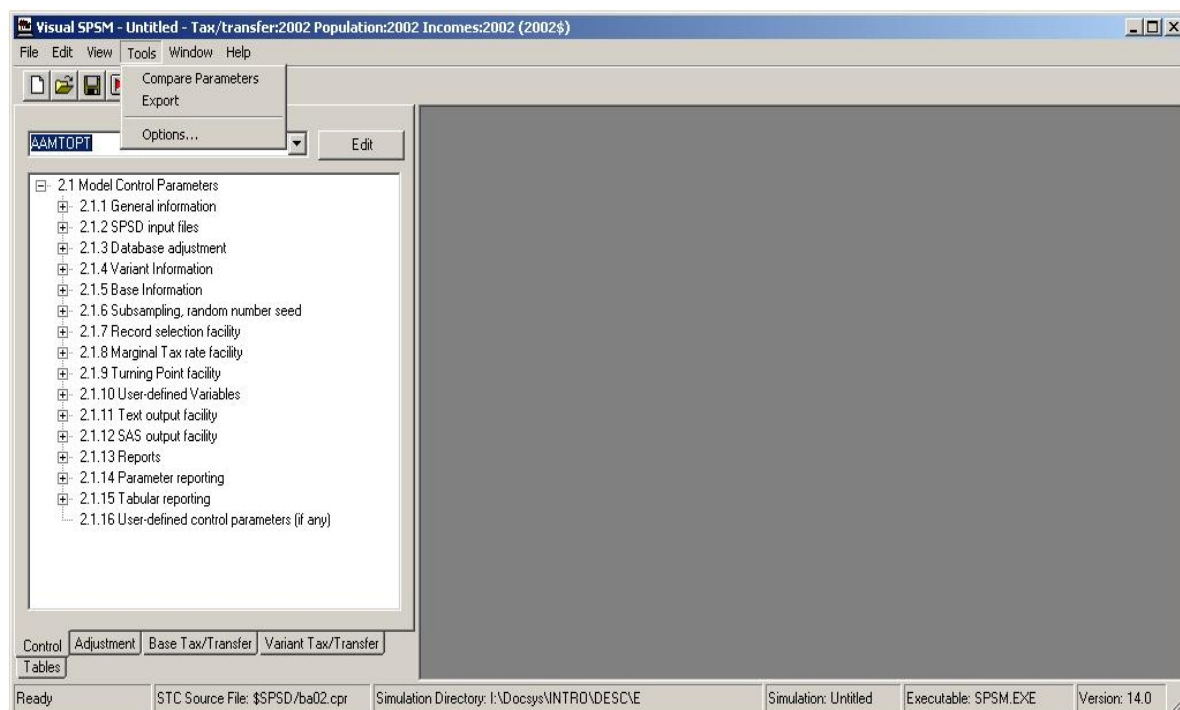
Introduction	1
SPSD/M Utilities	2
Export to Spreadsheet (import.exe)	2
Visual SPSM	2
Classic SPSM	3
spsmiter: SPSM Goal Seeking Facility	4
Iteration Facility Syntax	5
compparm: Parameter File Comparison Utility	9
Visual SPSM	9
Classic SPSM	10
Use of compparm to Update Parameter Files	12
spsdinfo: SPSD/M File Header Utility	14
pupdate: SPSM Parameter Update Facility	15
Update factors currently used in mpr files	17
Update factors currently used in apr files	18
Syntax	18
Examples:	19
bldspd: .spd Database Creation Utility	20
bldfxv: .fxv Database Creation Utility	21
bldwgt: .wgt Database Creation Utility	22
bldmrs: .mrs Database Creation Utility	23
Example: Creating a Subset Database	24
Programming Utilities	26
grep	26
sumskip	27

Introduction

This guide describes a number of stand-alone tools that are not directly part of the SPSM program. They are designed to extend the capabilities of SPSM in various ways. All tools run in Classic SPSM and many of them can also be run in Visual SPSM.

- The SPSPD/M Utilities section provides a description of all utilities available. It, for example, includes a description of the `compparm` utility, which produces a report showing all differences between two parameter files. How to use the tools in both Visual SPSM and Classic SPSM is discussed (where applicable).
- The section on Programming Utilities describes utilities mainly of use in 'glass box' mode, although the `grep` utility is of more general applicability.

Tools can be accessed using the Visual SPSM interface simply by making a selection from **Tools** in the menu bar.



In Classic SPSM, tools can be invoked by typing their name, along with any necessary arguments, on the command line. For example, to display all differences between the two parameter files `ba97.mpr` and `ba98.mpr`, the user would type the following command:

```
compparm ba97.mpr ba98.mpr
```

The tools will be found in the `\spsm\win32` directory. If you access the tools using the Classic SPSM prompt, the correct path will already be included in your

environment so you can simply type in the tool's name at the prompt.

In Classic SPSM, when the syntax of particular tools is given, the following notation is used:

- `compparm` leading terms given in a courier font represent the command used to invoke the tool.
- `[-t]` square brackets indicate an option which may or may not be used.
- `[-|!]` one or more vertical bars separating characters (or character strings) within option brackets indicate that more than one value is available for that option. The vertical bar functions as a separator only and is not entered when invoking the command. Usually the characters function as flags designating that the option is on or off, although some options have multiple values.
- *file1* terms presented in italics usually represent mandatory arguments that must be present on the command line for the tool to be invoked (eg. input and output file names). The exception is when they represent a value associated with an option that the user must enter.

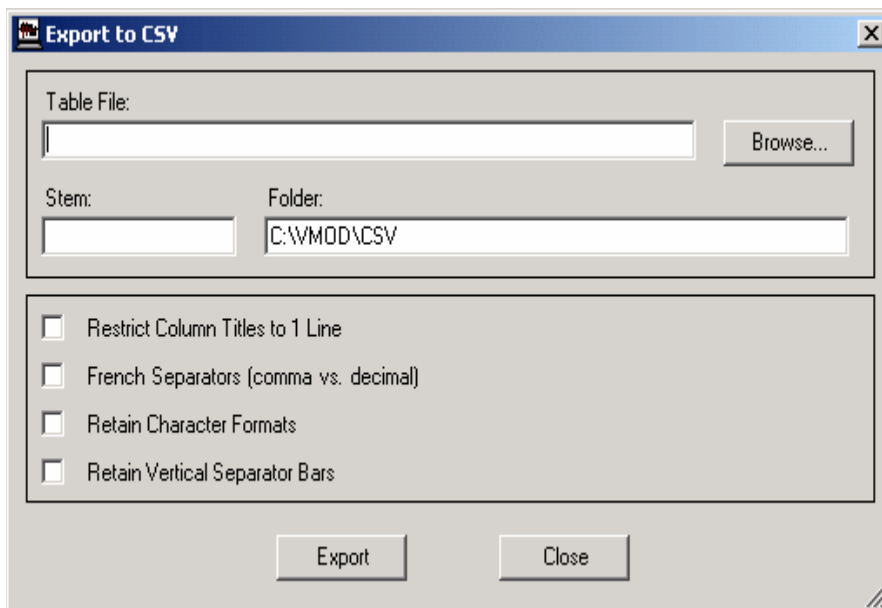
SPSD/M Utilities

EXPORT TO SPREADSHEET (IMPORT.EXE)

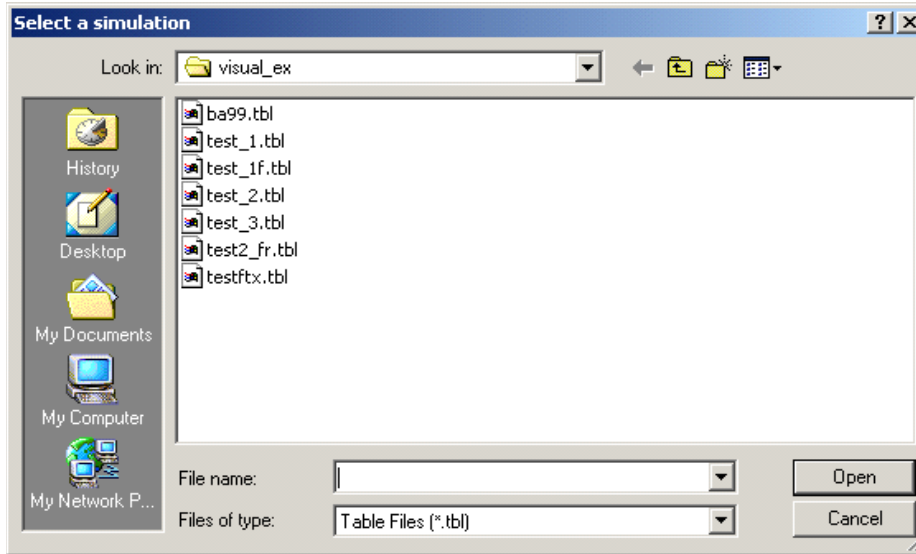
This utility converts each table into the format required by a spreadsheet package like Excel.

Visual SPSM

This tool is found in **Tools, Export to CSV**. The following interface will appear:



First select a table file to export. Table files have the extension `.tbl`. If a simulation is currently open, its `.tbl` filename will appear in the Table File line and its stemname (the prefix that will be given to each individual `.csv` table) will appear in the Stem box. Selecting **Browse** will allow the user to select a `.tbl` file to export. A folder named `\CSV` in the Simulation Directory will be generated automatically when selecting **Export**.



Other options :

- **Restrict Column Titles to 1 Line:** column headers will be truncated so that they only take up 1 row in the table
- **French Separators (comma vs. decimal):** the French comma is used as a decimal separator in table conversion to csv format instead of a dot
- **Retain Character Formats:** numeric fields are to be left in their formatted character form, and not converted into numbers
- **Retain Vertical Separator Bars:** vertical bar symbols (|) will be kept

Classic SPSM

In Classic mode, this is the `import` utility. It is invoked as follows:

```
import [-b] [-t] [-f] [-s] infile stemname
```

where:

- `-t` flag indicates that numeric fields are to be left in their formatted character form, and not converted into numbers.

- `-b` flag indicates that the vertical bar symbols (|) will be retained as well.
- `-f` flag indicates that the French comma is used as a decimal separator in table conversion to csv format instead of a dot.
- `-s` flag indicates that the column headers will be truncated so that they only take up 1 row in the table.
- `infile` is the name of a file containing SPSM tables (.tbl file).
- `stemname` is a string which is used to generate the names of the resulting files. The files produced have the form `stemname_tableid.prn`.
- `tableid` identifies the extracted table. It consists of the alphanumeric string that follows an instance of the string 'Table' in `infile`.

In the case of tables with three or more dimensions, a numeric sub-identifier is appended to `tableid`. For example, assuming that the file `run1.tbl` exists in the current directory, if the command:

```
import run1.tbl tab
```

is given, and if `run1.tbl` contains built-in tables 0, 2, and 2A, as well as user-defined tables 1U and 2U, then `import` would generate the following files:

```
tab_0.csv
tab_2.csv
tab_2A.csv
tab_1U.csv
tab_2U1.csv
tab_2U2.csv
tab_t.csv
```

The file `tab_t.csv` contains the page header lines found at the top the `run1.tbl` file. These page header lines are not included in any of the `.csv` files. The above example assumes that the user-defined Table 2U was three dimensional and had two slices.

`import` normally converts numeric table entries into numeric spreadsheet entries. This allows the tables to be manipulated numerically inside the spreadsheet once they have been imported. The optional `-t` flag indicates that numeric fields are to be left in their formatted character form, and not converted into numbers. If in addition the `-b` flag is specified, the vertical bar symbols (|) will be retained as well.

SPSMITER: SPSM GOAL SEEKING FACILITY

The SPSM goal seeking facility provides a means to modify the dollar value of one or more specified parameters, through an iteration scheme, to produce a desired value of a particular variable in the tax/transfer function. The facility now allows both scalar and array parameters to be selected for goal seeking. The results of the goal seeking facility can then be written to the originating SPSM results file, adjusting the

applicable parameters.

This tool is useful for answering "what if" type questions. For example, what would be the federal surtax rate necessary to achieve revenue neutrality, had the goods and services tax been eliminated in 2001? It would be possible to answer this question by first running a 2001 base case to observe the level of federal tax revenue obtained when the gst is collected and then creating a variant case which sets the gst rates to zero.

The analyst could then create a table using the X-tab Facility (see the *User's Manual*, and *XTab User's Guide* for a description of the X-tab facility) which displays the federal tax revenue difference between the two cases (the loss in federal tax revenue due to the elimination of the tax). The goal seeking facility could then be used to estimate the level of federal surtax needed to reduce the difference between the base and variant runs to zero, thus obtaining revenue neutrality.

The user may specify either Newton's iterative method or a binary search as the iteration scheme. Both methods require the user to provide an initial estimate of the scalar parameter to be adjusted. In addition, the binary search option requires both an upper and lower limit for the parameter to be adjusted. If it is sufficiently close to the solution the procedures will converge, otherwise it will guess again. The following section provides the details necessary to access and use the `spsmiter` facility.

Iteration Facility Syntax

All commands necessary to execute the `spsmiter` facility must be entered in Classic SPSM either on the command line directly or from a batch file (*How To Run the SPSM* contains a description of the SPSM batch facility). To overview the format of the command, type `spsmiter`. The following explanation of the `spsmiter` facility will be output to the screen:

```
usage: spsmiter [-t] [-a] [-b<lower/upper>] sample model cprfile tableid
row col target tolres parm
```

```
-t          - just print action summary (optional)
-a          - modify parameters in .apr instead of .mpr file
-b          - use binary search instead of Newton's method
sample      - sample size(s) (eg. 0.05/1.00)
model       - name of the model (normally spsm)
cprfile     - name of the cpr file to be iterated to convergence
tableid     - table identification number (Eg.2)
row         - table row number (counting headings)
col         - table column number (counting headings)
target      - target value
tolres      - tolerance for convergence
parm        - name of the parameter to be modified to obtain convergence
              (and any extra)
```

Example:

```
spsmiter 1.00 spsm.exe test.cpr 0 7 2 235000 10 CTCPC
```

Description of spsmitter options:

`[-t]:`

When the expression `-t` is inserted between the `spsmitter` command and the sample expression in the batch command, only the action summary will be printed to the screen. This option should be used to check whether the `spsmitter` command statement has referred to the tables and the table values the analyst wanted. Analysts should note that if a variant results file has not been created the `INPVARMPR` control parameter will contain the original `spsm` model tax/transfer parameters and therefore any iteration conducted may damage this file. Checking the action summary before performing an iteration should prevent this mistake from occurring. An example of the action summary is given below – which describes the phrases produced during the iteration for the Worked Example following the Description of `spsmitter` options:

ACTION SUMMARY:

The Program `spsm.exe` will be run with the `test.cpr` control parameter file. The `FSURR1` parameter (along with 0 other) in the file `test.mpr` will be modified until the entry in row 24 and column 2 of table 0 in file `test.tbl` attains the desired value of 511479.8. The current value of this entry is 534702.

`[-a]:`

When the expression `-a` is inserted between the `spsmitter` command and the sample expression in the batch command, the facility will modify an adjustment parameter found in the `.apr` file as opposed to a model parameter found in the `.mpr` file. This gives the user the ability to adjust certain growth parameters to produce a desired model variable. For example, one could adjust the growth factor controlling farming income in Saskatchewan to force the modeled farm income variable to conform to a control total.

`[-b<lower/upper>]:`

When the expression `-b` is inserted between the `spsmitter` command and the sample expression in the batch command, the facility will use the binary search iteration scheme instead of Newton's method. This scheme is more effective when the relationship between the parameter to be adjusted and the target variable is not smooth and continuous. Note that both an upper and lower bound for the parameter must be specified along with the target variable like `-b0.5/1.3` where 0.5 is the lower limit and 1.3 is the upper limit.

sample:

The sample command allows the user to select the sample size for the second iteration, Phase II, from 0% to 100% of the database. Phase I will run

the iteration using 10% of a 100% SPSP database file. However if the iteration refers to SPSP output which was run using the 5% sample of the database, then the Phase I iteration will be run using the 5% sample and not 10% (the message 10% will nonetheless continue to be output to the screen). The user should note that an iteration procedure executed on 100% of the database may be time consuming, and it is therefore useful to perform the iteration using just Phase I by setting the sample command to zero, checking the result and then running the iteration again with the sample command set to one.

If only one sample is given then Phase I is automatically run with the given sample size and Phase II at 100%. It is possible to skip Phase I by setting sample in Phase I to 0% and Phase II at 100%: 0.0/1.0.

model:

The model command should refer only to the spsm.exe file unless the user has created another version of the model through the glass box procedure.

cprfile:

The cprfile command should refer to the .cpr file which produced the table containing the variable selected for goal seeking.

tableid:

The tableid command identifies the table identification number to be used. In the example given below, table 0 from test.cpr was selected. Both built-in tables and tables generated with the X-tab Facility (see User's Guide for a description of the X-tab Facility) can be selected.

row:

The row command identifies the row within the specified table which contains the variable selected for goal seeking.

col:

The col argument identifies the column within the specified table which contains the variable selected for goal seeking. The row and column numbers identify the cell within the table which contains the variable selected for goal seeking. The variable can be either a built-in variable or a user-defined variable (see User's Guide for a description of the User-defined Variable Facility). In the below example, the variable Federal Tax is located in row 24 and column 2 of table 0.

target:

The target value is the desired dollar amount of the variable the analyst has identified through the row and column coordinates. In the example below, the target value the analyst has chosen for total federal taxes is \$146,194.

tolres:

The tolerance for convergence should be set to some value, in our example it is 10 million (10), therefore an iteration result within 10 million dollars of the target value will be selected and the spsmitter procedure will stop. The user should note that the smaller the tolerance for convergence the longer the facility's processing time will be.

parm:

Up to 50 tax and transfer parameters can be selected for modification to obtain convergence. In order to process parameters which contain a vector or matrix of values give the parameter name along with the cell you wish to modify. For example, if you want to modify the marginal tax rate for the lowest income group, the parameter should be written as FTX[0][2] for the first row and third column. The user can identify which parameters are scalars by referring to the [Parameter Guide](#) or by observing the structure of the parameters by browsing through the .mpr file. In the worked example in the next section, the parameter FSURR1, federal surtax rate 1, was selected for modification.

Worked Example

Executing the following command in batch mode:

```
D:\SPSMTEST> spsmitter 1.0 spsm.exe test.cpr 0 25 2 146194 10 FSURR1
```

Produced these results:

ACTION SUMMARY:

The program spsm.exe will be run with the test.cpr control parameter file. The FSURR1 parameter (along with 0 others) in the file test.mpr will be modified until the entry in row 25 and column 2 of table 0 in file test.tbl attains the desired value of 146194. The current value of this entry is 121338, corresponding to parameter value 0.

```
Phase 1: 10% runs
Iteration=0 (10%) FSURR1=0 Result=117526 Error=-28668.9
Iteration=1 (10%) FSURR1=1 Result=200494 Error=54299.2
Iteration=2 (10%) FSURR1=0.345541 Result=146191 Error=-3.1
Exiting - Desired tolerance obtained
Phase 2: 100% runs
Iteration=0 (100%) FSURR1=0.345541 Result=151485 Error=5290.8
Iteration=1 (100%) FSURR1=0.281776 Result=145921 Error=-273
Iteration=2 (100%) FSURR1=0.284905 Result=146194 Error=0
Exiting - Desired tolerance obtained
```

We can answer the question posed above: "What would be the federal surtax rate necessary to achieve revenue neutrality, had the GST been eliminated in 2001?" The goal seeking facility estimated that the federal surtax rate would increase from

0% to 28% for revenue neutrality to be maintained.

The analyst can chose to write the results of the goal seeking facility to the `test.mpr` file thus changing the applicable variable values within the built-in tables, and within the X-tab tables the analyst had created, to reveal the effect of the new surtax.

COMPPARM: PARAMETER FILE COMPARISON UTILITY

The `compparm` utility will compare any two parameter files and produce a report detailing their differences. A number of options are available to control `compparm`'s operation. These options all have default values which are appropriate for on-line use with complete parameter files, or portions of parameter files (i.e. parameter include files).

In addition, `compparm` contains an option which can be used to create a parameter include file. The difference between two parameter files will be contained in an include file which can be read by SPSM. This option is very useful when updating parameter files to operate under a new release or glass box version of SPSM (see section below on updating parameter files).

Visual SPSM

This tool is found in **Tools, Compare Parameters**. The following box will appear:

The screenshot shows the 'Compare Parameters' dialog box. It has a title bar with the text 'Compare Parameters' and a close button. The dialog is divided into several sections. The first section contains 'File 1:' with a text box containing 'c:\spsd\ba02.mpr' and a 'Browse...' button. The second section contains 'File 2:' with an empty text box and a 'Browse...' button. The third section contains 'Maximum Width of Output Lines:' with a spinner box set to '80'. The fourth section contains 'Use Separator Characters:' with a checked checkbox. The fifth section contains three radio buttons: 'Compare Parameters Common to Both Files:' (selected), 'Include Values of Parameters only Found in File 1:', and 'Include Values of Parameters only Found in File 2:'. The sixth section contains 'Create Include File:' with an unchecked checkbox and a 'Filename:' text box. The bottom of the dialog has 'Execute' and 'Close' buttons.

File 1 and File 2 are required arguments, being the names of existing SPSPD/M

parameter files, consisting of parameter names and associated numeric or text values.

The input files do not have to be complete parameter files, and the parameters can be in any order in the input files. The report will be ordered in the same way that File 1 was ordered. It lists the values for File1 in one column and the values for File2 in an adjacent column.

The default setting is to compare parameters common to both files. There is an optional argument available to change this setting (see below).

There are some optional arguments in `compparm`:

- The Maximum Width of Output Lines specifies the total number of print positions that the resulting report will occupy. The default is 80, which is appropriate for screen display. Parameter values longer than width/2 are truncated. Truncation is indicated by a trailing '+' character on the parameter value.
- Use Separator Characters specifies whether the user wishes horizontal separator lines and comments to be included in the report. The default is a fully formatted and commented report.
- The user may also request the `compparm` facility to create a parameter include file (`.apd`, `.cpd`, `.mpd`) containing the differences between two parameter files. If this option is selected, a user-specified Filename must be given.
- As mentioned earlier, the default setting is to compare parameters common to both files. The user may choose whether or not parameters found only in File 1 are to be output. The other option specifies whether or not the user wishes parameters found only in File 2 to be output.

Classic SPSM

The full syntax for invoking `compparm` is as follows:

```
compparm [!|-]i [-w width] [[-|!]sep] [[!|-][12|1|2]]... file1 file2
```

where:

[!|-]i

This optional argument specifies whether the `compparm` utility will create a parameter include file (ie. those with the suffix `.cpi`, `.api`, or `.mpi`) containing the differences between two parameter files. The default is `!`, which results in a `compparm` report instead of an include file. The standard `compparm` report lists the values for file1 in one column and the values for file2 in an adjacent column.

-w width

This optional argument specifies the total number of print positions that the resulting report will occupy. The default is 80, which is appropriate for screen

display. Parameter values longer than width/2 are truncated. Truncation is indicated by a trailing '+' character on the parameter value.

`[-|!]sep`

This optional argument specifies whether the user wishes horizontal separator lines and comments to be included in the report (-) or not (!). The default is -, which results in a fully formatted and commented report. The user may wish to turn off the `sep` option to create a report which is more easily converted into parameter include files.

`[!|-]12`

This optional argument specifies whether or not the user wishes the values for parameters common to the two files to be compared. The default is to have them compared.

`[!|-]1`

This optional argument specifies whether or not the user wishes parameters found only in the first file specified (file1) to be output. The default is not to output these parameters in the report.

`[!|-]2`

This optional argument specifies whether or not the user wishes parameters found only in the second file specified (file2) to be output. The default is not to output these parameters in the report.

`file1, file2`

These are required arguments, being the names of existing SPSPD/M parameter files, consisting of parameter names and associated numeric or text values.

The input files do not have to be complete parameter files, and the parameters can be in any order in the input files. The report will be ordered in the same way that file1 was ordered.

If the user types the command

```
compparm \spsd\squ08.mpr \spsd\bb08.mpr
```

the resulting report would be written to the screen and would appear as follows (only part of the report is shown here):

```

                                PARAMETER FILE DIFFERENCE REPORT
=====
=
      \spsd\squ08.mpr          |          \spsd\bb08.mpr
Mon Oct 17 12:44:14 2005      | Mon Oct 17 12:41:36 2005
=====
=
###
## 2.3.1 Parameter File Description
###
```

```

=====
=
                        Description of tax/transfer parameter file
MPRDESC      Current values for 2008 MPRDESC      Current values for 2008
=====
=
###
## 2.3.2.6 Federal Sales Tax Credit
###
=====
=
                        Federal sales tax credit amount for filer
FSTCF        50.00                                FSTCF        70.00
-----
-
                        Federal sales tax credit amount for spouse
FSTCS        50.00                                FSTCS        70.00
-----
-
                        Federal sales tax credit amount for dependant
FSTCC        25.00                                FSTCC        35.00
-----
-
                        Federal sales tax credit reduction level
FSTCL        15000.00                            FSTCL        16000.00
=====
=

```

The output of `compparm` may be redirected to a file or to a printer. In the latter case, it might be desirable to increase the report width to avoid possible truncation of parameter values. This can be accomplished by typing a line such as the following:

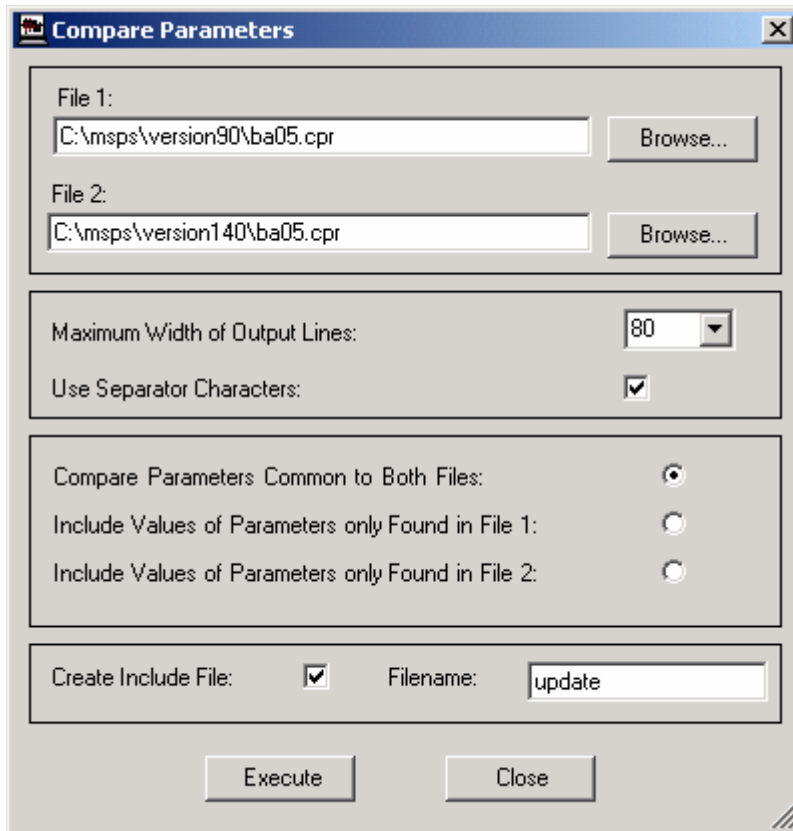
```
compparm -w132 sq08.mpr bb08.mpr >prn
```

Use of `compparm` to Update Parameter Files

The following example updates an old parameter file (i.e. version 9.0) with the changes contained in a new parameter file (i.e. version 14.0) and is only an illustration of the use of `compparm`. **It is strongly suggested to always proceed with the full implementation of SPSPD/M because new parameters are associated with changes in programs or new programs.**

Visual SPSPM

The following interface illustrates how to update the control parameter file `\version90\ba05.cpr` (version 9.0) to reflect the changes contained in the parameter file `\version140\ba05.cpr` (version 14.0).



A user-named Include File, `update`, will be written to the current Simulation Directory and called `update.cpd`. During the next SPSM simulation, the analyst can read in this control parameter include file to update the old control parameter file. The include file created from this command will appear automatically in Visual SPSM and looks like the one created using Classic SPSM (see below). The user can also print the include file from Visual SPSM.

Classic SPSM

When using `compparm`, the analyst is advised to maintain the default settings for options 1, 2, and 12 to ensure that the files become fully updated. For example, to update the control parameter file `\spsd3\ba05.cpr` (version 9.0) to reflect the changes contained in the parameter file `\spsd\ba05.cpr` (version 14.0) the analyst would enter the following command:

```
C:\SPSMCOMP>compparm -i \spsd3\ba05.cpr \spsd\ba05.cpr > update.cpi
```

The resulting include file, `update.cpi` would be written to the SPSMCOMP directory. During an SPSM run the analyst can read in this control parameter include file to update the old `.cpr` file. The include file created from this `compparm` command is as follows (only part of the include file is shown):

```
#####
#
## Include file to convert from '\spsd3\ba05.cpr' to '\spsd\ba05.cpr'  ##
#####
```

```

#
### ## 2.1.1 Descriptive information on this SPSM run ###
ALGDESC                                # Names of standard and alternate
algorithms
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
+\\
|Algorithm|                Standard                |                Alternate                |
|\\
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
+\\
|drv      | $Revision: 1.102 $      Sep 12/05 | None/aucun                Jun 17/05 |
|ui       | $Revision: 1.102 $      Nov 17/05 | None                      Jun 17/05 |
|famod    | $Revision: 1.102 $      Jun 17/05 | Untitled                  Jun 17/05 |
|oas      | $Revision: 1.102 $      Jun 17/05 | None                      Jun 17/05 |
### ## 2.1.2 SPSD input files ###
INSPSD      /spsd/vl40y02.spd      # Name of SPSD file (in)
### ## 2.1.3 Database adjustment ###
INPAPR      /spsd/ba02_05.apr      # Name of database adjustment parameter
file

### ## 2.1.4 Variant Information ###
VARALG      Version 14.0: 88-09    # Name of variant algorithm
#####
#
## Parameters Added: ##
#####
#
CPRVER      14.0                  # SPSD/M Release Version
CPRSFILE    $SPSD/ba05.cpr        # Starting Control parameter file
LICOOPT     1                    # T4 LICO definition 1=BeforeTax
2=AfterTax

```

The analyst should note that any parameters created in the glass box mode and contained in file1 are commented in the Parameters Removed section of the include file. To make these parameters active in the updated parameter file, remove the comment marker which precedes the parameter (#).

SPSDINFO: SPSD/M FILE HEADER UTILITY

SPSD database files and SPSM binary results files have a header which contains information on the file's creation date, version number and a number of other useful items. The `spsdinfo` utility will display this information in printable form. For example, if the user issues the command:

```
C:>spsdinfo $spsd\vl40y02.spd
```

the following report would be printed on the user's terminal:

```

File          : $spsd\vl40y02.spd
Type          : spd
Database ver.: 14.00
Binary ver.  : 14.00
Date         : Mon Oct 17 14:45:13 2005
Label        : Version 140
Licensee     : Internal StatsCan use only
Households   : 80365

```



```
Numbers      : 0
(unused)     : 0
(unused)     : 0
```

The tool produces additional information pertaining to any user variables contained in .mrs files. Specifically, any variable labels or classification level labels created in UVAR will be displayed if the associated variable was specified in OUTMRSVARS.

The following example illustrates this feature:

```
File          : ba08.mrs
Type          : mrs
Database ver. : 4.00
Binary ver.   : 4.00
Date          : Tue Nov 27 10:19:05 1994
Label         : Current values for 08
Licensee      : Statistics Canada
Households    : 2764
Numbers       : 23283
sizeof(MP)    : 11406
#bytes uv     : 190
Variables     : immicons newtax1 newinc1
Variable info:
label(newtax1)="newtax1";
label(newinc1)="Total income Group";
levels(newinc1)=
    "Min-5,000",
    "5,000-10,000",
    "10,000-50,000",
    "50,000-100,000",
    "100,000-MAX";
```

PUPDATE: SPSM PARAMETER UPDATE FACILITY

The **pupdate** utility is used to grow parameters in the “.mpr” (tax/transfer model parameters) or “.apr” (database adjustment parameters) files. Partial include files (“.mpi” “.mpd” “.api” “.apd”) can also be grown.

Pupdate creates a parameter file (out=outfile) by growing a parameter file (cur=file1) using given update factors (name=value). If a second file is given (nxt=file2), then the output file will only grow parameters which were grown or which are missing in the second file. For example, if in the second file a tax rate is given without an update statement whereas a tax credit was grown using CPI, the output file will keep the tax rate from the second file, but grow the tax credit using the appropriate update factor.

Values for update parameters are supplied on the command line upon invocation in the form ‘name=value’ or passed to pupdate in a file.

Syntax:

pupdate cur=file1 [nxt=file2] out=outfile [name=value]...

Or

pupdate -f *filename* where *filename* contains

```
cur=file1
out=outfile
[nxt=file2]
[name1=value1]
.
.
.
[name=value]
[CURNAME=label]
```

Parameter files

cur - original parameter file.
nxt - second parameter file (optional)
out - new parameter file

Source and update statements

Each parameter that can be grown will have the following comment statements:

Source - origin of parameter value.
Update - method of scaling or setting the value.

e.x.

```
GISST      3      GIS take-up rate: single pensioner by benefit level
           0      0.322  (0.0005)
           701    0.680  (0.0001)
           2909   1.000  (0.0001)
                                # Source: Grown from ba99.mpr using CPI=1.014
                                # Update: Factor[1]=CPI
```

Source

The parameter value's source will either be drawn from a number of official publications or from information generated by pupdate.

```
e.x.  # Source: Redbook, 1995 Edition
      or
      # Source: Grown from ba99.mpr using CPI=1.014
      # Source: Copied from ba99.mpr
      # Source: Given as LABEL=...
```

Update

Any parameter having an update statement will be grown, others will be simply copied to the output file. The update statement will contain one of the following entries preceded by the update header "# Update: ":

Value=*value* - Sets parameter or rows of parameter to *value*.

Factor=*value* - Multiplies *value* times all column values in row(s) to compute grown values.

Factor[]=*value* - Multiplies *value* times selected columns values in row(s) to compute grown values.

e.x. # Update: Value=LABEL
 # Update: Factor=CPIM3
 # Update: Factor[2,3]=DEFAULT

Cur File

In the case where only a single file (*cur*) is passed as an argument and where an update statement and factor are found, the parameter will grown. If no update statement is found, the parameter is simply copied from 'cur' to the output file. If an update statement is found and no update factor is provided, an error will be generated.

Nxt File

In the case where a second file is passed (*nxt*), if the parameter is not found in 'nxt' or if the source given in 'nxt' is 'Grown', 'Copied' or, 'Given' then a grown entry from 'cur' is written to the out file. If the source statement contains no update information, the parameter information is simply copied from 'nxt' to the out file.

Arguments

cur - previous years file

nxt - optional second file

out - the resulting output file

CURNAME - base year file; if not given will default to CUR file

all the update factors used in the cur file

Errors and warnings

Warnings will be issued if a parameter source statement is not found in either 'cur' or 'nxt' or if different update methods are used for the same parameter in the two files.

Error messages will be issued and the program halted if files cannot be found or referenced parameters were not passed as command line arguments or contained in the command file.

Update factors currently used in mpr files

LABEL description of applied process

CPI Consumer Price Index
 CPIR Consumer Price Index deflator to estimate previous year income
 CPIRR Consumer Price Index deflator to estimate income for 2 years previous
 CPIM3 Consumer Price Index over 3%
 CPINF Consumer Price Index - Newfoundland
 CPIPE Consumer Price Index - Prince Edward Island
 CPINS Consumer Price Index - Nova Scotia
 CPINB Consumer Price Index - New Brunswick
 CPIQU Consumer Price Index - Quebec
 CPION Consumer Price Index - Ontario
 CPIMA Consumer Price Index - Manitoba
 CPISA Consumer Price Index - Saskatchewan
 CPIAL Consumer Price Index - Alberta
 CPIBC Consumer Price Index - British Columbia
 CPILAG Consumer Price Index - CPI from previous year
 CPIALLAG Consumer Price Index - Alberta CPI from previous year
 CPISALAG Consumer Price Index - Saskatchewan CPI from previous year
 CPIM3LAG Consumer Price Index over 3% from previous year
 AIW the average industrial weekly earnings
 UIR growth rate for UI/EI maximum earnings
 NONE set to 1, used for deflation
 DEFAULT set to 1, used for deflation
 YEAR Target year (4 digits)
 VERSION SPSPD/M Release Version
 SFILE The starting tax/transfer parameter file

Update factors currently used in apr files

LABEL description of applied process
 CPI Consumer Price Index
 AIW the average industrial weekly earnings
 UIR growth rate for UI/EI maximum earnings
 VERSION SPSPD/M Release Version

Syntax

Values for update parameters are supplied on the command line upon invocation in the form ``name=value'` or passed to pupdate in a file.

pupdate *cur=file1* [*nxt=file2*] *out=outfile* [*name=value*]...

Or

pupdate -f filename where *filename* contains

cur=file1
out=outfile
[nxt=file2]
[name1=value1]

.
 .
 .

[name=value]
[CURNAME=label]

Examples:

To update the entire ba99.mpr to the next year with a 3% federal CPI growth rate:

```
C:> pupdate cur=ba99.mpr nxt=ba00.mpr LABEL=Higher_Growth YEAR=2000
CPI=1.03 CPIM3=1.0 AIW=1.026 CPIR=.952 CPIRR=.952 UIR=1.000 CPINF=1.018
CPIPE=1.018 CPINS=1.018 CPINB=1.018 CPIQU=1.018 CPION=1.018 CPIMA=1.018
CPISA=1.018 CPIAL=1.018 CPIBC=1.018 CPILAG=1.018 CPISALAG=1.018
CPIALLAG=1.018 CPIM3LAG=1.0 DEFAULT=1.0 NONE=1.0 VERSION=10.2
SFILE=ba99.mpr out=myout.mpr
```

OR

```
C:> pupdate -f growmpr.lst
```

where growmpr.lst contains the following entries:

```
cur=ba99.mpr
nxt=ba00.mpr
LABEL=Higher_Growth
YEAR=2000
CPI=1.03
CPIM3=1.0
AIW=1.026
CPIR=.952
CPIRR=.952
UIR=1.000
CPINF=1.018
CPIPE=1.018
CPINS=1.018
CPINB=1.018
CPIQU=1.018
CPION=1.018
CPIMA=1.018
CPISA=1.018
CPIAL=1.018
CPIBC=1.018
CPILAG=1.018
CPISALAG=1.018
CPIALLAG=1.018
CPIM3LAG=1.00
DEFAULT=1.0
NONE=1.0
VERSION=10.2
SFILE=ba99.mpr
out=growme.mpr
```

To grow the database adjustment files to 2004:

```
C:> pupdate cur=ba98_03.apr LABEL=Grow_to_2004 CPI=1.018 AIW=1.026
```

```
UIR=1.000 VERSION=10.2 out=new04.apr
```

To grow an mpi file which only contains the parameters for tax on taxable income in BC:

```
C:> pupdate cur=bctest.mpi out=newbc.mpi CPIBC=1.03 NONE=1
```

BLDSPD: .SPD DATABASE CREATION UTILITY

`bldspd` provides a means to convert ascii output files, of a specific format, into a compressed machine readable format which can be read by the `spsm` as an `spsd` database file; therefore, the user is able to create a database containing a desired subset or a user revised version of the existing `SPSD` database (`v140y02.spd`).

For example, the user may wish to create a database containing only one particular household, for use with the turning point facility, or to create a database containing rare household cases which may be used to debug new algorithms. Advanced "shaping" of the data may also be performed by experienced users.

Note that when a subset is created, the results will change slightly since the random numbers will be different. This implies that transfers which depend on take-up (see `gis` and `GISST` for example) may change. If you want to get the same results, you will need to flag off all relevant take-up parameters.

The build programs expect fixed field ASCII files. To easily access this ascii style, read in the control parameter include file `/spsd/bldspd.cpi` as well as the database adjustment parameter include file `/spsd/bldspd.api`.

Note that these files use the ASCII output facility output style (see the [*Parameter Guide*](#) for a description of the parameter `ASCSTYLE`). `ASCSTYLE=5` is a fixed format which contains all variables listed in `ASCVARS`, is blank delimited, and contains all records per case beginning with a household record which is followed by individual records.

The full syntax needed to compress the ascii output file into a machine readable file format using the `bldspd` utility is as follows:

```
bldspd [-a][-x] file1 file2
```

where:

```
[-|!]a
```

This optional argument specifies whether or not the user wants the ascii file to be compressed and appended to an existing compressed file (`.spd`). The default is `!` which results in the creation of a new compressed file. The user may want to use the append option to create a compressed file from more than one ascii file (for example, an ascii file which is so large it must be

separated into sections to conserve disk space).

`[-|!]x`

This optional argument specifies whether or not the ascii file contains additional 'extra' variables. The default is to read extra database variables. These are fields left intentionally blank. Users can employ these variables in either Glass Box applications or for database rebuilding.

file

file1 must be an ascii output file (`.prn`) formatted in `ASCSTYLE=5`. file2 must be a new compressed file (`.spd`) if option a is not used. If option a is used then file2 must be an existing compressed file (`.spd`).

The `bldspd.cpi` can be found in the `/spsd` directory and includes 23 extra variables for the user. These include files `bldspd.cpi` and `bldspd.api` are used to produce the ASCII input files used by `bldspd.exe`.

Example:

```
C:/SPSMBLD> bldspd test1.prn test1.spd
```

The ascii output file, `test1.prn`, will be converted into compressed format and written into file `test1.spd`. To run SPSM with the new database, `test1.spd`, set the control parameter `INSPD` to `test1.spd`.

In the previous example, `test1.prn` read the output layout from the file `/spsd/bldspd.cpi` which provide the final layout for the input file in SPSD/M, and read the file `/spsd/bldspd.api`. Any failure to read those two files will result in serious mistakes.

BLDFXV: .FXV DATABASE CREATION UTILITY

`bldfxv` provides a means to convert ascii output files into a compressed format, readable by the SPSM as a Survey of Household Spending (SHS) Database; thus, the user can modify the existing SPSD SHS database to adjust the SPSD to reflect different assumptions about family expenditures.

ASCII output files must have a specific format as described below. This format may be invoked by reading the file `\spsd\bldfxv.cpi` as well as the corresponding `\spsd\bldfxv.api`.

The full syntax for invoking `bldfxv` is as follows:

```
Bldfxv [-a] file1 file2
```

where:

`[-|!]a`

This optional argument specifies whether or not the user wants the ascii file to be compressed and appended to an existing compressed file (`.fxv`). The

default is ! which results in the creation of a new compressed file. The user may want to use the append option to create a compressed file from more than one ascii file (for example, an ascii file which is so large it must be separated into sections to conserve disk space).

file

file1 must be an ascii output file (.prn) formatted in ASCSTYLE=4. file2 must be a new compressed file (.fxv) if option a is not used. If option a is used then file2 must be an existing compressed file (.fxv).

Example:

```
C:/MYDIR> bldfxv test1.prn test1.fxv
```

The ascii output file test1.prn, which was produced using /spsd/bldfxv.cpi, will be converted into the compressed format, test1.fxv. To run SPSM with the SPSD database adjusted to reflect the new family expenditure assumptions, test1.fxv, set the control parameter INPFXV to test1.fxv.

The bldfxv utility can be used to process an ASCII file produced from SPSM when a subset of households was being selected (such as all those in a particular province). However, users must use the following procedure to create the ASCII file, modifying the parameter values given in /spsd/bldfxv.cpi.

1. A user variable with the value 1 for the household must be created, and this variable should be used instead of fxclohhv in ASCVARS(which is the default in the file /spsd/bldfxv.cpi). So newvar=1/hhnin; or HH:newvar=1; would both be fine.
2. SELSPEC should not include an expression equivalent to or of the form (hdfrstfx==1). or (hdlastfx==1). This will force a SHS record to be output for each household otherwise selected.

Together, these changes will result in a file with correctly duplicated expenditure pattern vectors. This change will allow users to create complete sets of .spd, .fxv and .wgt files for particular provinces, resulting in performance improvements if this is their normal mode of use.

In the previous example, test1.prn read the output layout from the file /spsd/bldfxv.cpi which provide the final layout for the input file in SPSD/M, and read the file /spsd/bldfxv.api. Any failure to read those two files will result in serious mistakes.

BLDWGT: .WGT DATABASE CREATION UTILITY

bldwgt provides a means to convert ascii weight files into a compressed format

which can be read by the `spsm`. Through the `bldwgt` utility the user can modify the existing SPSPD weight files to change the composition of the database.

For example, to increase the number of EI recipients within the database the user could adjust the EI weights to allocate EI transfers to more individuals within the database. The user should be aware that although the process for adjusting weight files is straightforward the implications of any adjustment are complex and the user should fully understand all of the interactions between the existing weight files before attempting this procedure.

The full syntax for invoking `bldwgt` is as follows:

```
bldwgt [-a] totwgt file1 file2
```

where:

`[-|!]a`

This optional argument specifies whether or not the user wants the ascii file to be compressed and appended to an existing compressed file (`.wgt`). The default is `!` which results in the creation of a new compressed file. The user may want to use the append option to create a compressed file from more than one ascii file (for example, an ascii file which is so large it must be separated into sections to conserve disk space).

`totwgt`

This is the sum of weights of the file to be created. It represents the estimated population in numbers of households. In the special case where a full sample is being created (i.e. `SAMPLEREQ` is 1.0000) the user may place the value 0 in this field and the `bldwgt` program will sum the weights on the file.

`file`

`file1` must be an ascii output file (`.wgt`) formatted in `ASCSTYLE=4`. `file2` must be a new compressed file (`.wgt`) if option `a` is not used. If option `a` is used then `file2` must be an existing compressed file (`.wgt`).

Example:

```
C:/SPSMBLD> bldwgt 0 test1.prn test1.wgt
```

The ascii output file `test1.prn` will be compressed into a machine-readable weight file. This weight file can then be used to adjust the database during an SPSM run (set the control parameter `INPWGT` to `test1.wgt`).

BLDMRS: .MRS DATABASE CREATION UTILITY

This tool converts a file of text data into a binary SPSM results (`.mrs`) file. `bldmrs` takes three arguments: the first is the number of households, the second is the input ASCII file, and the third is the name of the `.mrs` file being built. The number of households can be discovered using the `spsdinfo` on the full database.

The input file to `bldmrs` follows a particular format that consists of two sections, a header section followed by a data section. The first line of the header section gives the names of the variables. These names can be optionally surrounded by quotes (") and are separated by one or more spaces. This line is followed by an optional section which can provide labels and level information for user variables. This descriptive information is provided as a set of semi-colon delimited SPSM statements using the `label` and `levels` statements described in the [User's Guide](#). Lines of descriptive information must start with a lower-case alphabetic character in the first column.

The data section consists of one line for each individual in the SPSPD. Each line contains the blank-separated values of the variables for that individual.

The following example illustrates the appearance of the input file used to produce a results file with the SPSM modeled variables for disposable and consumable income. It could also be produced using the text output facility and `ASCSTYLE` equal to 3.

```
immdisp immicons
0 0
1000 987
25000 22500
.
.
.
```

The next example shows how to create a `.mrs` file that can be used to modify the province variable in an SPSM run (using the Reference Value Facility).

```
hdprov
5
5
5
.
.
.
```

The following example shows how to create a `.mrs` file containing user variables. A classification variable for poverty status and a base income variable are defined.

```
baseinc povstat
label(baseinc) = "Base Income";
label(povstat) = "Poverty Status";
levels(povstat) = "Poverty", "Near Poverty", "Non-poverty";
10000 0
10000 0
10000 0
50000 2
.
.
.
```

EXAMPLE: CREATING A SUBSET DATABASE

In the following example, a database for all individuals in Newfoundland, which is a

subset of the complete database, is going to be created. The process has three steps and involves at least three separate model runs.

1. BUILD WEIGHT FILE(S) (.wgt)

A) Run the spsm. Begin the run selecting a default control parameter file and set the output file names to NFLDWGT

B) Read the include file \spsd\bldwgt.cpi to update the control parameters.

C) Alter the selection facility parameters and X-tab facility parameters as follows:

```
SELFLAG      1
SELSPEC      hdprov==NFLD
XTFLAG       1
XTSPEC       HH:{units:S=0}
```

D) Use bldwgt.exe to create the new weight file called NFLDYY.WGT where YY refers to the year of the weight file selected during the model run. The sum of weights may be found in the X-tab table in the .tbl file.

```
C:>bldwgt 162282 nfldwgt.prn nflddy.wgt
```

Because a different weight file is read for each year, you must repeat this entire step for each year for which you want to perform analyses .

2. BUILD SHS FILE (.fxv)

A) Run the spsm. Begin the run, selecting the base year default control parameter file and set the output file names to NFLDFXV.

B) Read the include file \spsd\bldfxv.cpi to update the control parameters.

C) Alter the selection facility as follows:

```
SELSPEC      hdprov==NFLD
```

D) Alter the user variable facility parameter as follows:

```
UVARFLAG     1
UVAR         a=1.0/hhnin;
```

E) The variable a is used to correctly duplicate expenditure pattern vectors and should be used instead of fxclohv in the ASCVARS parameter of the text output facility. Edit ASCVARS as follows:

```
ASCVARS      fxseqhv fxseqhv hdseqhh a
.....
```

F) Read the include file \spsd\bldfxv.api to update the adjustment parameters and then complete the model run.

G) Use bldfxv.exe to create the new FXV file called NFLDYY.FXV.

```
C:\bldfxv nfldfxv.prn nfldyy.fxv
```

3. BUILD DATABASE FILE (.spd)

A) Run the spsm. Begin the run, selecting the base year default control parameter file and set the output file names to `NFLDSPD`.

B) Read the include file `\spsd\bldspd.cpi` to update the control parameters.

C) Alter the selection facility as follows:

```
SELSPEC      hdprov==NFLD
```

D) Read the include file `\spsd\bldspd.api` to update the adjustment parameters and then run the model.

E) Use `bldspd.exe` to create the new SPD file called `NFLDYY.SPD`

```
C:\bldspd nfldspd.prn nfldyy.spd
```

4. BUILD AN INCLUDE FILE (.cpi)

A) Create an include file `NFLDYY.CPI` to replace the SPSD input files `INPWGT`, `INPFXXV`, `INSPD` by the new one just created. This file should have the following statements:

```
INSPD          nfdlyy.spd
FXVFLAG        1
INPFXXV        nfldyy.fxv
WGTFLAG        1
INPWGT         nfldyy.wgt
```

When you will run spsm using the new database, just read the include file `nfldyy.cpi` to update the control parameters to point to the appropriate data files.

WARNING

The results will be slightly different from the full SPSD. This is due to the use of random numbers in calculating GIS take-up rates whatever other take-up rates are used in that model year. Persons will be assigned a different number from the sequence of randoms. If you want to compare the results with the ones obtained from the complete database, set `GISTURFLAG` and any other relevant take-up parameters to zero.

Programming Utilities

GREP

This utility searches all specified files (multiple wildcard file-specs in the current directory are allowed) for a given string and displays lines in which the string is found. Two options are allowed: `-n` precedes each matching line with its line number and `-l` just gives a list of file names satisfying the pattern. `grep` is quite

useful as an on-line cross-reference device for perusing source code files or parameter files. For example, if the command

```
C>grep -l capg *.c
```

were given in the `\spsm\glass` directory, the output, which identifies all modules that refer to capital gains, would look like this:

```
agis.c
amemol.c
atxcalc.c
atxinet.c
atxitax.c
atxqinet.c
gis.c
memol.c
txcalc.c
txinet.c
txitax.c
txqinet.c
```

As another example from a previous version of SPSPD/M, if the command

```
C>grep CTCPC *.mpr
```

were issued in the `\spsd` directory, the output would look like this:

ba82.mpr: CTCPC	343.00	# Child tax credit per child
ba84.mpr: CTCPC	367.00	# Child tax credit per child
ba85.mpr: CTCPC	384.00	# Child tax credit per child
ba86.mpr: CTCPC	454.00	# Child tax credit per child
ba87.mpr: CTCPC	489.00	# Child tax credit per child
ba88.mpr: CTCPC	559.00	# Child tax credit per child
ba88y88.mpr: CTCPC	474.06	# Child tax credit per child
ba89.mpr: CTCPC	565.15	# Child tax credit per child
ba89y88.mpr: CTCPC	460.40	# Child tax credit per child
sq88.mpr: CTCPC	524.00	# Child tax credit per child
sq88y88.mpr: CTCPC	444.38	# Child tax credit per child

As can be seen, the value of `CTCPC` in each tax/transfer parameter file in the `\spsd` directory is displayed.

SUMSKIP

This is an executable that helps Statistics Canada to verify if you have installed your version of SPSPD/M properly or if there is any problem associated with a file.

An example of the command is:

```
C>sumskip $spsd\ba08.mpr
```

This produces an output like the following:

```
skip=194 bytes=254255 checksum=3234115767
```