BatchMan 1.7 Reference Manual

MATTIAS VERONA

Electronic Warfare

Technical Threat System Assessment

FOI
Defence Research Agency
Command and Control Systems
P.O. Box 1165
SE-581 11 Linköping

Phone: +46 13 37 80 00
Fax: +46 13 37 81 00

www.foi.se

FOI-R--1991--SE
ISSN 1650-1942

Methodology report
May 2006

Command and Control Systems

Mattias Verona

BatchMan 1.7 Reference Manual

Issuing organization	Report number, ISRN	Report type
FOI – Swedish Defence Research Agency Command and Control Systems P.O. Box 1165 SE-581 11 Linköping	FOI-R--1991--SE	Methodology report
	Research area code 6. Electronic Warfare and deceptive measures	
	Month year May 2006	**Project no.** E7014
	Sub area code 61 Electronic Warfare including Electromagnetic Weapons and Protection	
	Sub area code 2	

Author/s (editor/s)	Project manager
Mattias Verona	Mikael Hansson
	Approved by
	Sponsoring agency
	Scientifically and technically responsible

Report title

BatchMan 1.7 Reference Manual

Abstract

The Batch Manager, in the report shortly described as BatchMan, is an application designed to execute multiple simulations with an ACSL® model. BatchMan was created to run batch simulations from the simulation models created by the department Electronic Warfare Assessments at FOI, including simulation models of surface to air missiles, air to air missiles, antiship missiles and air to surface missiles, implemented as software or HWIL simulation models. BatchMan can also be used to create an ACSL command file.
ACSL itself does not have to be installed on the computer. The only files necessary beside the BatchMan files are the model prx and prj files along with any additional files the model might need to run, e.g. flight path files.
Major new features since the last report include:

- Increased integration with AFE and LKZ.
- Ability to sort and modify output data.
- Enhanced user friendliness.
- Enhanced safety and backup possibilities.
- Several batch run simulations can be started and run from one BatchMan configuration.
- Simulations can be run distributed over a network or several processes.
- Storage of data is possible in a MySQL database.

This report describes the functionality of the application BatchMan and how it is installed.

Keywords

Batch run simulations, Simulation, Modelling, Electronic Warfare Assessment, Missiles

Further bibliographic information	Language English
ISSN 1650-1942	**Pages** 45 p.
	Price acc. to pricelist

Utgivare	Rapportnummer, ISRN	Klassificering
FOI - Totalförsvarets forskningsinstitut Ledningssystem Box 1165 581 11 Linköping	FOI-R--1991--SE	Metodrapport
	Forskningsområde 6. Telekrig och vilseledning	
	Månad, år Maj 2006	**Projektnummer** E7014
	Delområde 61 Telekrigföring med EM-vapen och skydd	
	Delområde 2	

Författare/redaktör	Projektledare
Mattias Verona	Mikael Hansson
	Godkänd av
	Uppdragsgivare/kundbeteckning
	Tekniskt och/eller vetenskapligt ansvarig

Rapportens titel

BatchMan 1.7 Referensmanual

Sammanfattning

BatchMan är en applikation som skapats för att utföra mängdsimuleringar med ACSL®-modeller. BatchMan har utvecklats för att förenkla mängdsimuleringar av de simuleringsmodeller som utvecklas vid institutionen för telekrigvärdering. Dessa omfattar luftvärnsrobotar, jaktrobotar, sjörobotar och attackrobotar, implementerade som mjukvarumodeller eller HWIL-modeller. BatchMan kan också användas för att skapa ACSL kommandofiler.

ACSL behöver inte vara installerat på datorn. De enda filer som behövs förutom BatchMan-filerna är modellens prx- och prj-filer samt eventuella filer som behövs för modellen, t.ex. flygdata-filer. BatchMan har uppgraderats på flera punkter sedan den förra officiella rapporten. De största uppgraderingarna omfattar:

- Ökad integration med AFE och LKZ.
- Möjligheter att sortera och modifiera data.
- Ökad användarvänlighet.
- Ökad säkerhet och möjligheter till backuper.
- Flera batch jobb kan startas och köras från en och samma BatchMan-konfiguration.
- Simuleringar kan köras distribuerat över ett nätverk eller flera prossesser.
- Data kan lagras i en MySQL-databas.

Denna rapport beskriver funktionaliteten hos BatchMan och hur applikationen skall installeras.

Nyckelord

Mängdsimuleringar, Simulering, Modellering, Telekrig, Robotar

Övriga bibliografiska uppgifter	Språk Engelska
ISSN 1650-1942	**Antal sidor:** 45 s.
Distribution enligt missiv	**Pris: Enligt prislista**

THIS PAGE IS INTENTIONALLY LEFT BLANK

CONTENTS

Figures

1 INTRODUCTION

1.1 Conclusion

The Batch Manager, in the report shortly described as BatchMan, is an application designed to execute multiple simulations with an ACSL® model. BatchMan was created to run batch simulations from the simulation models created by the department Electronic Warfare Assessments at FOI, including simulation models of surface to air missiles, air to air missiles, antiship missiles and air to surface missiles, implemented as software or HWIL simulation models. BatchMan can also be used to create an ACSL command file.

ACSL itself does not have to be installed on the computer. The only files necessary beside the BatchMan files are the model prx and prj files along with any additional files the model might need to run, e.g. flight path files.

Major new features since the last report include:

- Increased integration with AFE and LKZ.
- Ability to sort and modify output data.
- Enhanced user friendliness.
- Enhanced safety and backup possibilities.
- Several batch run simulations can be started and run from one BatchMan configuration.
- Simulations can be run distributed over a network or several processors.
- Storage of data is possible in a MySQL database.

This report describes the functionality of the application BatchMan and how it is installed.

1.2 Background

The program is designed to make the production of multiple simulations easier for the user. All inputs (model constants) and outputs (model output variables) are controlled through a GUI (Graphical User Interface). BatchMan is designed to keep running even if one run fails, causing the simulation model to crash. One general problem, when producing multiple simulations, is that if one simulation fails, because of a flaw in the model, the batch run of simulations will terminate at that point and has to be restarted manually.

The last official report of BatchMan was “Batch Manager 1.0 - A program for generating multiple simulations with ACSL®” [BATCHMAN].

BatchMan was developed at the department Electronic Warfare Assessments.

1.3 Intended use

The program is intended to be used when multiple simulations are to be produced with a model constructed with either ACSL or ACSL/GM, see [ACSL]. All constants in the model can be viewed and set from BatchMan. They can be stepped or randomly chosen within limits set by the user. Apart from this, any of the output variables can be selected and will be saved

when the simulations are terminated. Enhanced integration with AFE [AFE] also allows drag and drop from the start parameter dialog in AFE.

For the output variables the final value is stored, e.g. the miss distance in a missile model simulation. If any other variable values should be stored during the course of the simulation, this has to be done in the model by storing the value in a variable that remains the value as a final value. BatchMan can also be used to generate a command file for use in ACSL.

1.4 Development environment

The Batch Manager has been developed with Microsofts Studio .NET 2003 [VISUAL] and is written in C++ and C#. It makes use of the class libraries “Microsoft Foundation Classes” and the “.NET Framework Class Library 1.1”. Communication with the ACSL model is handled through an ACSL API (Application Program Interface) delivered by Xcellon, the creators of ACSL, see [Xcellon]. BatchMan is intended to be used with Windows XP.

1.5 Usefulness for the defence

BatchMan is designed to create batch run simulations from the simulation models created by the department Electronic Warfare Assessments at FOI. These simulation models include models of surface to air missiles, air to air missiles, antiship missiles and air to surface missiles. Both software simulation models and HWIL simulation models can be operated from BatchMan. The main benefits for the defence are listed below:

- Makes it possible to perform batch run simulations in order to create launch- and kill zone diagrams.
- Batch runs are performed more effectively since BatchMan is designed to be stable with backup possibilities regardless of the stability of the simulation model in use.
- Easy to use graphical user interface.
- Makes studies of parameter sweeps easy to conduct.
- Integration with the tools LKZ [LKZ] and AFE, also developed at FOI, makes a package solution that is easier to use.

2 INSTALLATION

The Batch Manager does not require ACSL being installed on the computer. It does however require the model .prj and .prx files along with additional files the model might need to run.

All BatchMan files are kept in the BatchMan directory on the CD. This directory should be copied to the hard drive without moving any of the files. To run the program, execute BatchMan.exe from the Windows Explorer or run a .baf file that has been created by and associated with BatchMan.

2.1 System requirements

BatchMan is designed to run on a standard PC with Windows XP Professional edition, with the .NET Framework 1.1 software component installed:

To run BatchMan distributed over a network or over several processors the following extra components also need to be installed:

- Windows Message Queuing
- MySQL database

The .NET Framework is included on the BatchMan CD. MySQL is free software that can be downloaded from http://dev.mysql.com/downloads/. Windows Message Queueing is included in Windows XP Professional edition, but not installed by default. Installation instructions for MySQL and Windows Message Queuing can be found in section 3.11 – “Running BatchMan in distributed mode”.

3 USER'S GUIDE

This chapter describes all of the menus and dialogs used in BatchMan and how they are intended to be used.

3.1 The initial dialog

The easiest way to start BatchMan is through BatchMan.exe in the Windows Explorer. The first dialog viewed to the user is shown in Figure 1. Three options are provided, opening a previously saved BatchMan-file, opening an ACSL model file or opening an empty instance of BatchMan. Previously saved BatchMan-files are saved as .baf-files and will consist of the model path to the model used, and all the saved user parameters. Opening an ACSL model file, a .prj-file, will open a browser dialog from which a model .prj-file can be chosen. BatchMan will then be opened with all of the model constants in the ACSL model that can be changed and all the model variables that can be viewed.

Figure 1 The initial dialog

3.2 The main BatchMan interface

The main interface in BatchMan is used to set all the constant parameters that BatchMan will set in the model before it is run. Figure 2 shows the main interface when BatchMan has been opened with the model file test.prj. The *Model constants* listbox, to the left in the figure, shows all of the model constants in the ACLS model.

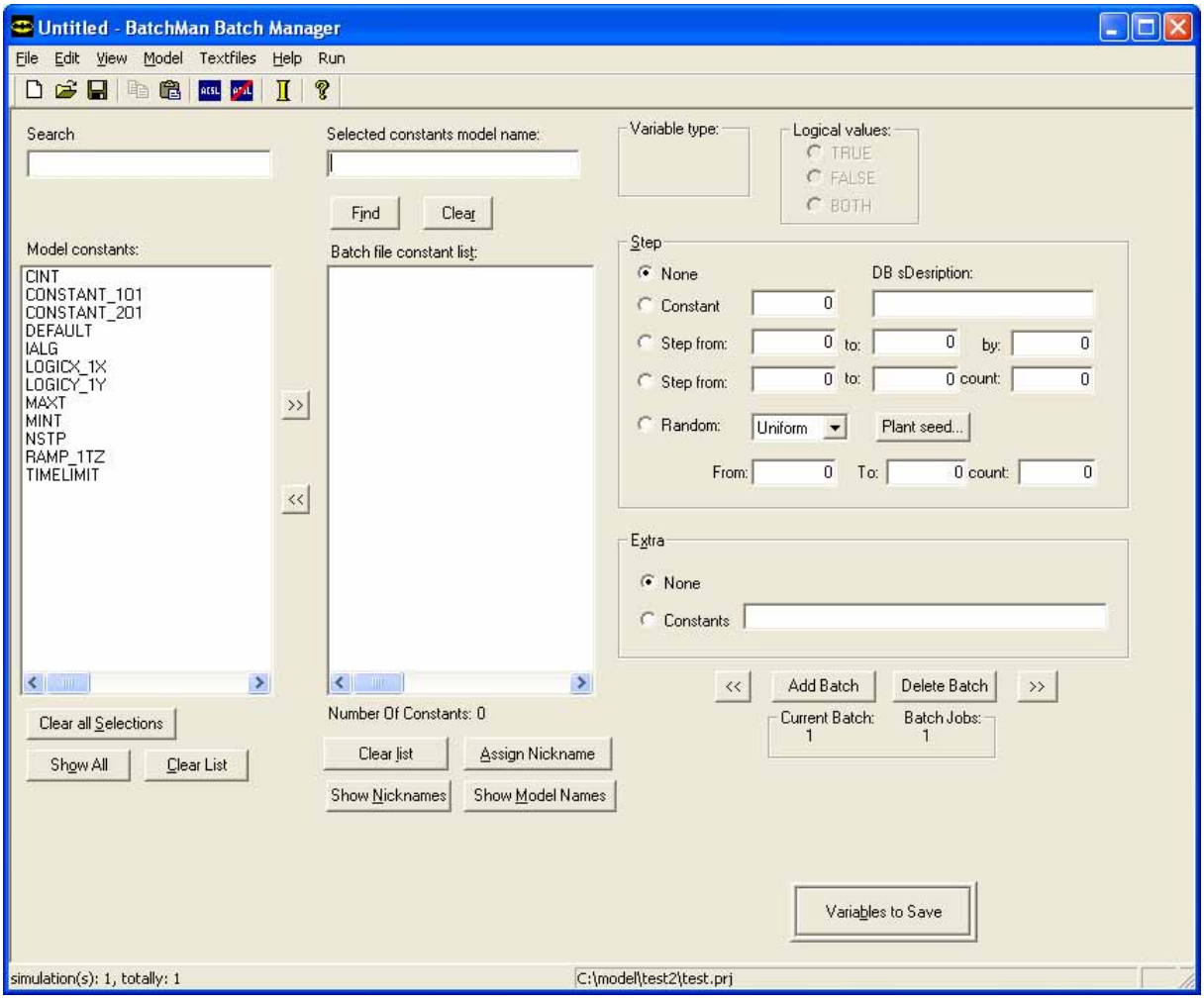

Figure 2 The BatchMan main interface

The default values of the model constants will be those saved in the model dll, i.e. the model .prx-file. If the ACSL model has been developed in ACSL/GM, the values saved to GM, see Figure 3, will be the default values in BatchMan. Any of these values can be changed by the user, either set to a single constant value or looped through a number of values which can be done in different ways described in section 3.5.

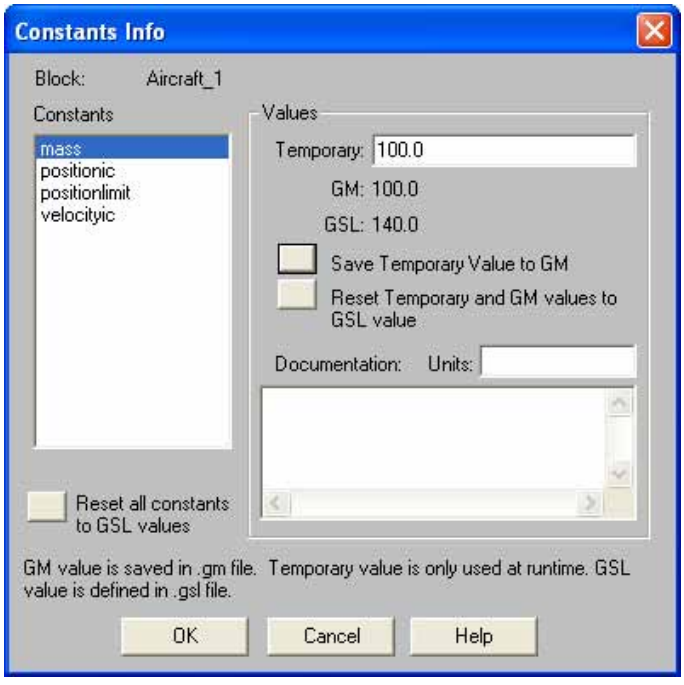

Figure 3 Constant values in ACSL/GM. In this example the value 100.0 will be the default value for the variable mass in BatchMan.

3.3 The main menu

This chapter describes the commands available from the main menu. Some of the commands can also be found in the toolbar or as keyboard shortcuts.

Figure 4 The main menu of BatchMan

3.3.1 File

The submenu File contains commands for opening and saving BatchMan configurations, stored in .baf-files, and for exiting the program.

Open

Opens previous settings stored in a .baf-file. The file contains information about chosen values for constants, chosen variables, conditions for breaking the simulation etc. If the .baf-file is associated with BatchMan.exe in Windows, BatchMan will automatically be executed when the .baf-file is opened. The baf-file also contains the path for the prj-file, relative to the baf-file. The Batch Manager will locate the prj-file and load the model. If the prj-file can't be located by BatchMan, the user will be asked to locate it manually through a standard windows dialog. When this is done the previously stored settings are loaded for the particular model.

Save as Saves the settings in a file with default extension .baf and name chosen by the user. The standard Windows dialog is used for this.

1…5 Recently loaded or saved files.

Exit Exits the program. Saving changes will be possible through the "save as"-dialog as described above before the program is exited.

3.3.2 Edit

The standard Windows commands undo, cut, copy and paste can be used to edit all input fields in the Batch Manager interface.

3.3.3 View

The view menu can be used to decide whether or not to view the toolbar and the status bar. The toolbar consists of shortcuts to the menu and the status bar shows information about the number of simulations to be made and the number of different batch simulations.

3.3.4 Model

Commands for loading and unloading ACSL models through their prj-files as well as an option to create backup files during simulations.

Load Loads a model through its prj-file. The user is asked to save the current configuration through a standard Windows dialog. Then the model that is in use, if any, is unloaded. After a successful model load, the model name with extension appears in the status bar and the model constants appear in the "model constants" list box.

Unload Unloads the currently loaded model, which includes removing all constants, all variables and all stored values. The user is given the opportunity to save the changes before the command is executed.

Backup This will open the backup dialog used to store simulation results during simulations as a backup feature. This is described more detail in section 3.10.2.

3.3.5 Textfiles

Commands for creating and modifying text files used by ACSL or BatchMan.

Generate cmd ACSL can read command files, .cmd-files, in which the user can set start parameters to be used in the model [ACSL]. The *Generate cmd* command in the BatchMan main menu generates such a command file that can be used when the model is run from ACSL. The generated command file will consist of one *prepare /clear /all* and one *start* for each simulation. No variables can be chosen

when generating a cmd-file, only chosen constant values will be stored. Hence, storing the results is up to the user.

When the command “generate cmd” is chosen, a dialog box appears in which the user is asked to name the procedure, see Figure 5:

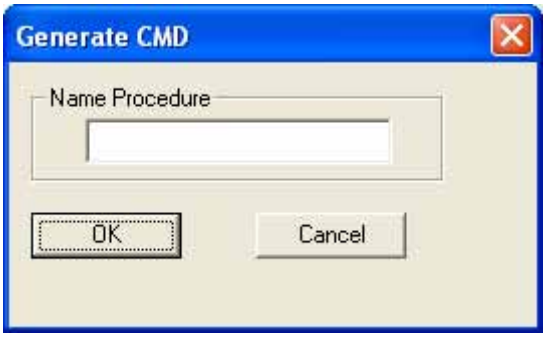

Figure 5 Generate CMD dialog

The procedure is used in ACSL to distinguish between different command settings stored in the same command file. Appendix 4.1 shows an example of a created cmd-file.

BatOrganizer Opens the BatOrganizer dialog used to edit simulation results data files created by BatchMan. This tool is described further in section 3.7.

3.3.6 Help

Information about BatchMan and the currently loaded configuration

About Batch Manager Opens the about dialog with program version number.

Information Opens a dialog containing the full search path to the currently used ACSL model prj-file and the BatchMan configuration baf-file.

3.3.7 Run

Determines how the simulations should be run on the computer.

Distributed Opens the Run Distributed dialog. This is used when BatchMan should be run distributed over a network or locally over parallel processes and is described further in section 3.11.

3.4 The toolbar

The toolbar is normally found just below the menu. The toolbar can be floated or docked with the mouse. Figure 6 shows the toolbar as it appears when floated.

Figure 6 The toolbar

Below follows a description of the tools in the toolbar:

Opens previous settings stored in a .baf-file. The file contains information about chosen constant values, chosen variables, conditions for breaking the simulation etc.

Saves the settings in a file with default extension .baf and name chosen by the user. The standard Windows dialog is used for this.

Copies a value in an edit box.

Pastes a value to an edit box.

Loads a model through its prj-file.

Unloads a model, which includes removing all constants, all variables and all stored values.

Opens the information dialog containing the full search path to the currently used ACSL model prj-file and the BatchMan configuration baf-file.

Opens the about BatchMan dialog.

3.5 Setting the model constants

This chapter describes how the ACSL model constants are chosen whose values should be altered.

3.5.1 Finding the desired constants

When a model has been loaded, all of the ACSL model constants will be listed in the *Model constants* list box on the left side of the Batch Managers interface. Constants are selected by

highlighting their name in the list. Multiple selections can be made simultaneously. The list box with its associated buttons and a *Search* edit box is illustrated in Figure 7.

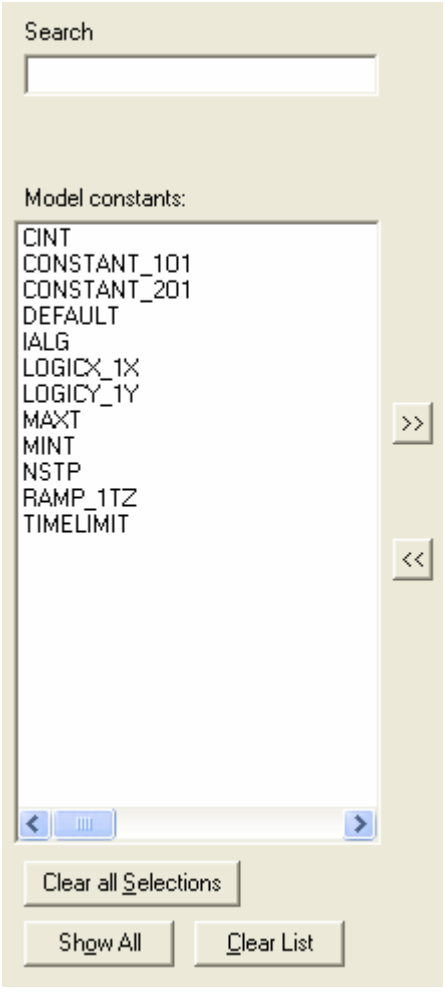

Figure 7 The model constants listbox and associated buttons

The *Search* edit box above the list box can be used to sort the data shown in the list box. When a combination of characters is entered in the Search edit box, only constants containing those characters will be shown in the list box. If the character string is started by a wildcard, '*', any constant containing the written characters will be displayed. If a wildcard is not used only constants beginning with the written characters will be shown. Note that the entered set of characters must begin with the asterisk in order for the wildcard function to work.

Beneath the list box there are three buttons to decide which constants should be shown or chosen. The *Clear all Selections* button can be used to clear the selections if a number of constants have been highlighted. The *Show All* button lists all of the model's constants and the *Clear List* button does the opposite, clears the list of all model constants. The *Show All* button is always executed when a model is loaded, either from "load model" in the menu or initially when BatchMan is first started.

3.5.2 Selecting the constants

When the desired constants have been chosen, simply push the button on the right side of the constant list box, marked >>, to transfer them to the *Batch file constant list* list box, see Figure 2. If the selected constant is a one-dimensional vector, a dialog will appear with the elements of the vector as shown below in Figure 8:

Figure 8 The vector dialog

The desired elements can be chosen either directly in the list box or, if all the elements should be selected, by pressing the *Select All* button. *Clear Selections* deselects all of the highlighted elements in the list. Press *OK* to add the selected elements to the *Batch file constant list* list box or *Cancel* to return to the main interface without adding and constants.

If the vector contains more than 10 elements a warning dialog will appear before the vector dialog appears. This warning dialog will let the user know how many elements the vector contains and will ask the user to continue or cancel. This is necessary since the ACSL API will interpret tables as vectors, and if a table is too large BatchMan can get occupied for a long period.

Note: BatchMan is limited to one-dimensional vectors so matrixes will not appear in the Model constant list box.

When the “>>” button has been pressed the selected constants will appear in the *Batch file constant list* list box. Only one constant at a time can be selected in this box since the value and type of the constant will appear to the right of the list box. To deselect a constant and remove it from this list, simply highlight it and push the button marked <<. To remove all constants from the list, push the clear list button beneath the list box. Figure 9 shows the *Batch file constant list* list box with its associated buttons and edit box.

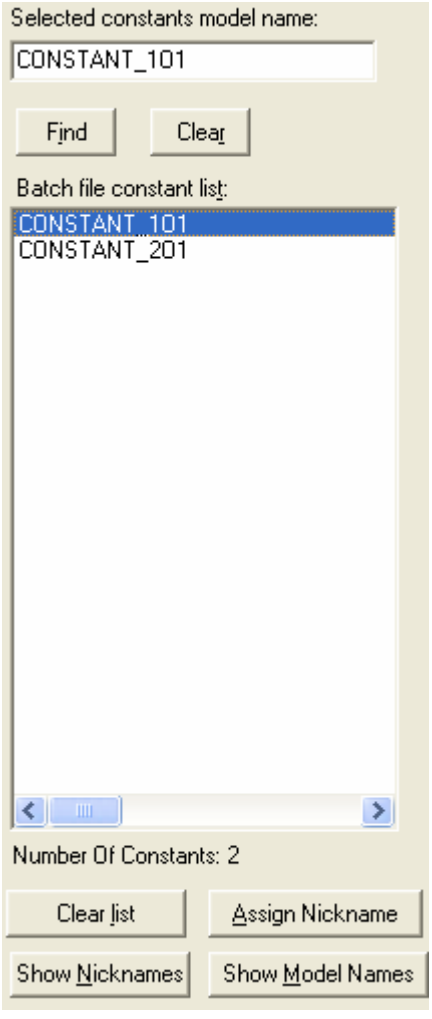

Figure 9 The Batch file constant list box with associated buttons

The search mechanism is somewhat different for the *Batch file constant list* list box. This is due to the fact that only single selections can be made. This list box also shows all the constants all of the time, so that there is no confusion as to what constants are selected. This is probably less of a problem in this list box since it will contain fewer constants than the *Model constants* list box. To find a constant, the entire name must be written in the selected edit box. Then press the *Find* button. The *Clear* button clears the edit box.

The constants in the *Batch file constant list* list box can be given nicknames. When a constant is selected, the *Assign Nickname* button will open the dialog shown in Figure 10.

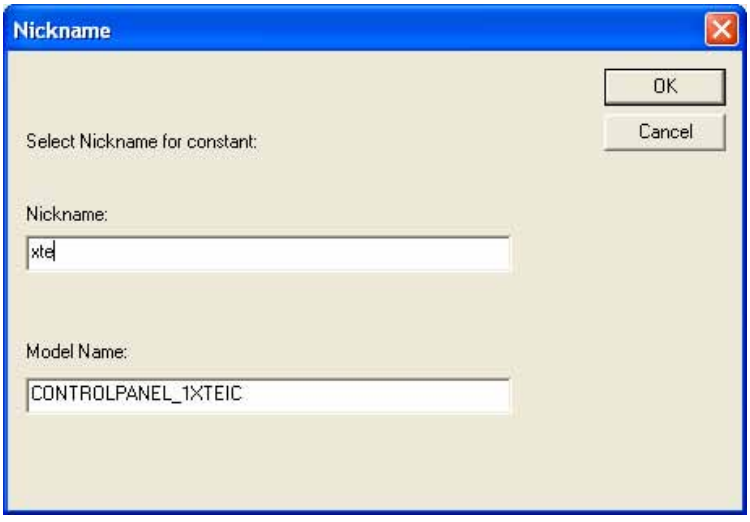

Figure 10 The Nickname dialog

The desired nickname can be entered in the *Nickname* edit box. Switching between showing the constant nicknames and the constant model names can be done with the buttons *Show Nicknames* and *Show Model Names*. If a nickname has not been chosen for a variable, the ACSL model name will be used as a default nickname. Both the model names and the nicknames are stored in the simulation data created by BatchMan.

3.5.3 Setting the constant values

There are different ways of setting the constant values in BatchMan. The area of the interface shown below in Figure 11 will be highlighted, unless the constant is logical.

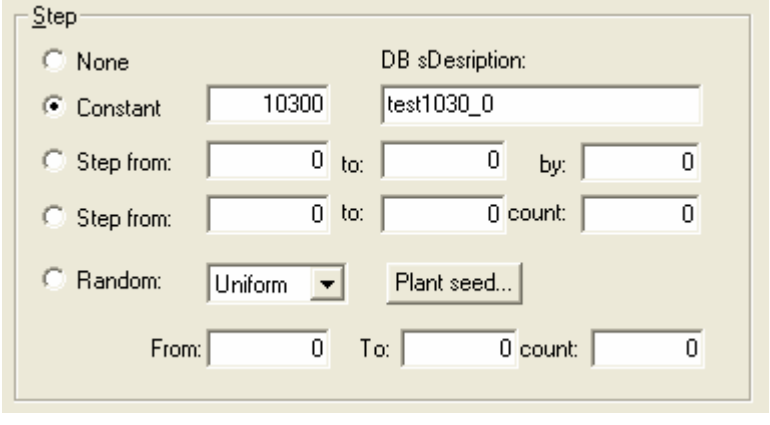

Figure 11 Setting constant values

The example shows a constant with the value of 10300. The value that will appear when a constant is selected is the value stored in the model's prx-file. There are five ways of setting the values for a constant. The first is simply to choose the *Constant* radio button and edit the value in the edit box to the right of the radio button. If the constant value is to be changed during the batch simulation, this can be controlled by the *Step from* radio buttons. The first will step the constant from the value in the *Step from* edit box to the value in the *to* edit box with the step length set in the *by* edit box. E.g. 1 to 7 by 2 will give the values 1, 3, 5, 7 and hence 4 simulations. The second *Step from* radio button works like the first one with the

difference that the number of steps, entered in the *count* edit box, controls the values that the constant will run through. Hence 1 to 7 count 4 will give the same result as above.

Another way of stepping the constants is to do so randomly. Two kinds of distributions can be chosen, Uniform or Gaussian. Like the second *Step from* alternative, the uniform distribution uses a lower and a higher limit together with the number of values in between. With the Gaussian distribution however the edit boxes will set the mean value and the standard deviation together with the number of values, see Figure 12:

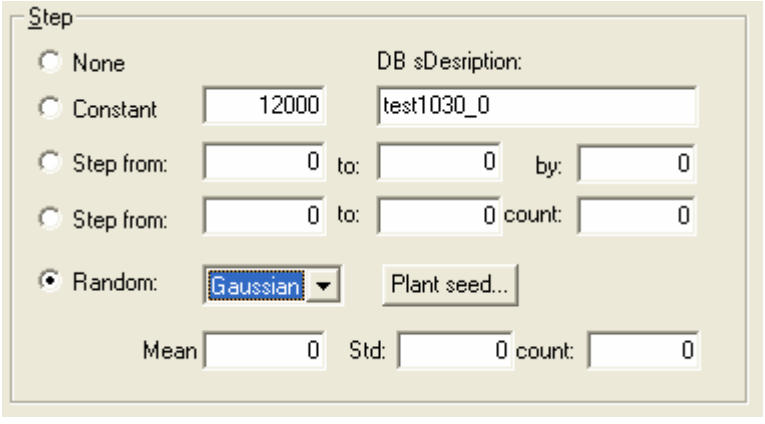

Figure 12 Randomly chosen constants

The distribution demands a seed from which to start. This seed can be set by pressing the Plant seed button which will generate the dialog shown in Figure 13:

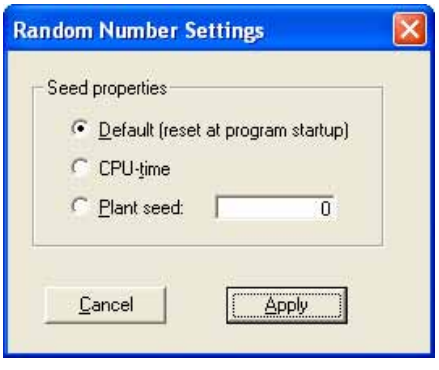

Figure 13 Setting the seed for random distribution

As the figure illustrates three different choices can be made. Default will set the seed to 1 on the first time. CPU-time will use the cpu-time in order to get different results at different times. The third alternative is to set the seed by entering an integer in the edit box to the right of the “Plant seed”-radio button.

If extra values are desired these can be entered in the *Extra* constants edit box beneath the step area described above. Here several numbers can be entered separated by space. The example shown in Figure 14 will generate values 0.1, 6, 16 and 67.

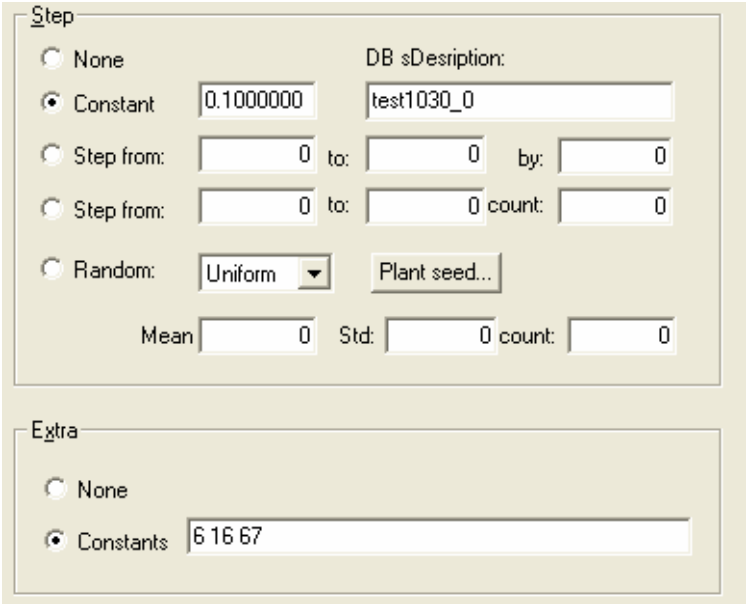

Figure 14 Example using the Extra constants edit box

The selected constant type will be shown in the area marked *Variable type* in the main interface. This can read *Real*, *Integer* or *Boolean.* If the constant selected has a logical value, the type specifier will read *Boolean* and the *Logical values* area will be highlighted instead of the *Step* area as shown in Figure 15.

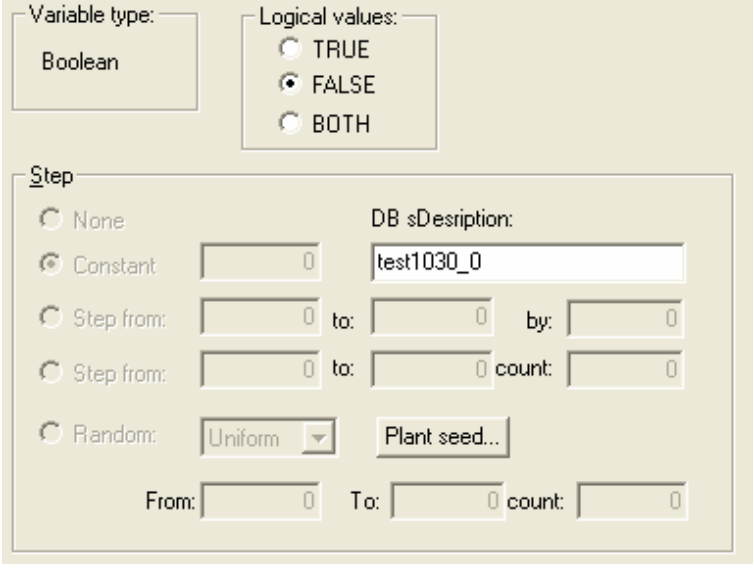

Figure 15 Boolean constant

Here three choices can be made: true, false, or both. Choosing both will generate one simulation with the constant value set to true and one with the constant value set to false.

Note: The total number of simulations that will be generated is the total amount of combinations that can be made out of the chosen values for the selected constants. Hence, if two constants are selected, one with three different values and the other with two, the total number of simulations will be three times two, which equals six.

The buttons *Add Batch* and *Delete Batch*, located beneath the Extra area, can be used to add or remove a batch of simulations from the total amount of simulations that will be performed. One set of simulation data will be created from each batch of simulations. Pressing the *Add Batch* button will copy the currently selected batch of simulations into a new set of simulations. Changes can then be made without interfering with the previous batch simulation. The arrow buttons, << and >>, can be used to switch between the batch simulations. In Figure 16 two batch simulations have been created and the currently chosen batch simulation is batch 1.

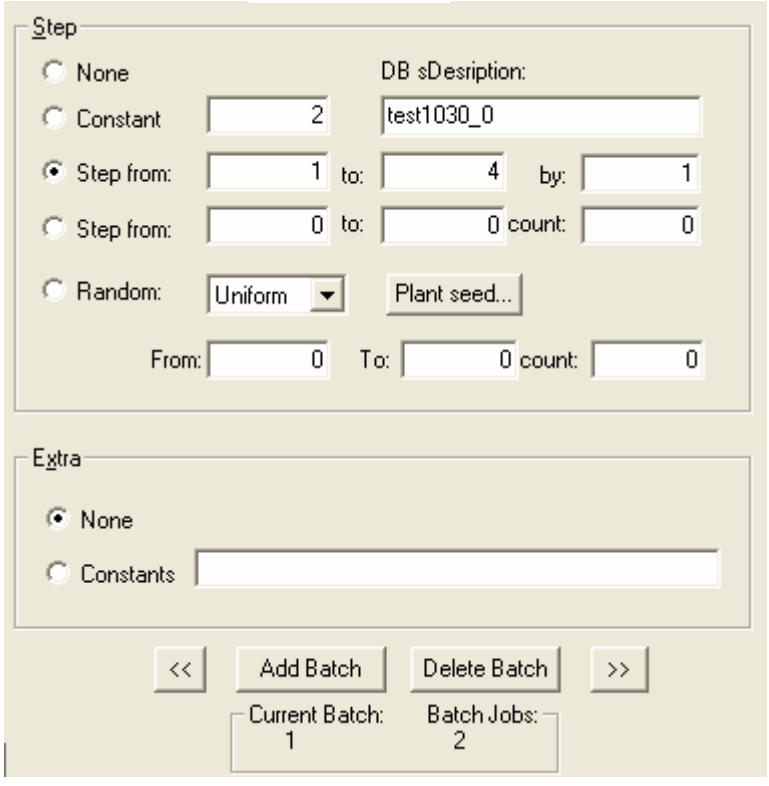

Figure 16 Number of batch simulations

The area DB sDesription is used when simulations are made distributed over a network. In this mode the simulation data results will be stored in a MySQL database instead of in a text file. The description field can be used to more easily identify the correct batch in the database. This is described more thoroughly in section 3.11.

3.5.4 Using drag and drop from AFE

BatchMan supports the ability to set the model constants with the values used in the dialog *Start parameters* in AFE. This integration can be useful if a model constant configuration tested in AFE needs to be batch run. By dragging from the *Start parameters* dialog on to the main interface window in Batch Manager the user can transfer the current start parameter values and configuration to Batch Manager for further processing. Below follows a description of how this is done:

1. Open the model in AFE. AFE needs to be set to use a workspace directory, see [AFE].
2. Open the same model in BatchMan.
3. Open the *Start parameters* dialog in AFE

4. Left click and drag from the white background of the start parameter dialog. The mouse pointer should change appearance if done correctly.
5. Drop on the main interface window of BatchMan
6. The model constants of the *Start parameters* dialog in AFE should appear in the *Batch file constant* list in BatchMan see Figure 2.
7. Edit any constants that should be parameter swept and start BatchMan as usual.

3.6 The status bar

As previously mentioned, the status bar contains information about the number of simulations and the loaded model. At the left side of the status bar the number of simulations in the currently selected batch simulation is shown along with the number of batch simulations. When a constant has been changed the number of simulations will be updated as soon as focus is removed from the edit box that is currently in use.

To the right of the number of simulations the number of successfully completed simulations together with the number of failed simulations will be shown during a batch job so that the user always is updated on the remaining number of simulations. This information will only appear after the first simulation is completed however.

In the second segment of the status bar the search path of the project file loaded is displayed. An example of the status bar is shown below in Figure 17:

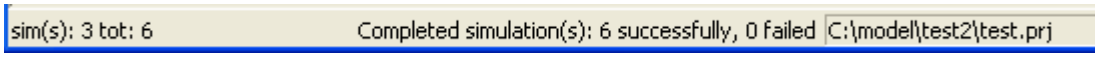

Figure 17 The status bar

3.7 BatOrganizer

The BatOrganizer is a tool developed to organize data files created by BatchMan. It is developed as a separate dll-component that can be incorporated into any dotNET-application, and it can be started from BatchMan as a standard dialog. It is opened from the main menu in BatchMan through *Textfiles->BatOrganizer* and is shown below in Figure 18.

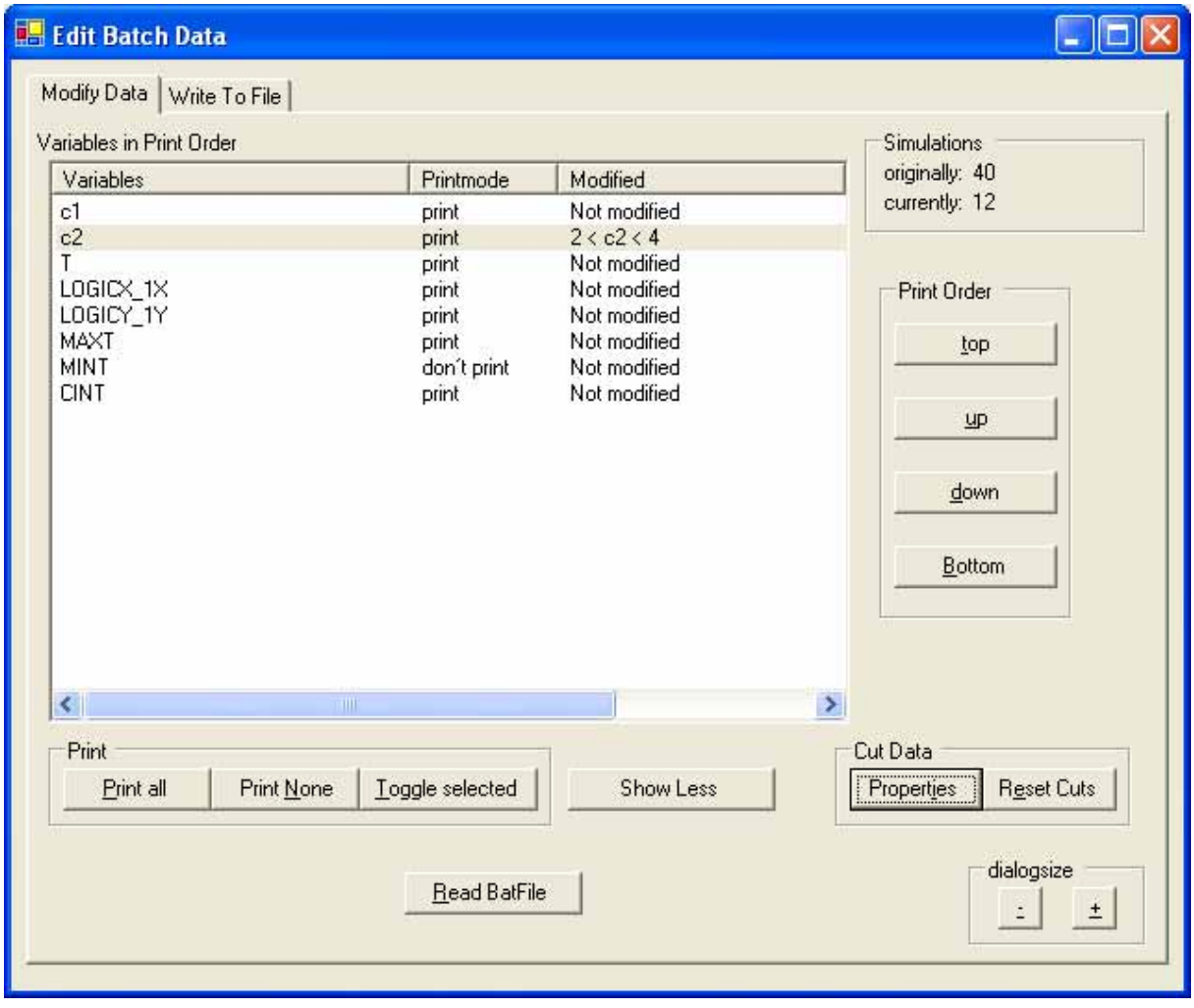

Figure 18 The BatOrganizer dialog

The main dialog consists of two tab pages. The first is Modify Data and is shown in the figure above. A previously created data file, a .baf file, can be opened through the *Read BatFile* button. All the model constant and variables read from the file will be shown in the list box *Variables in Print Order*. The print order of the constants and variables can be changed using the *Print Order* buttons to the right in the dialog. By selecting a variable and pressing one of the buttons the variable will be moved up or down in the print order list. By double clicking a variable it can be set not to be printed in the outdata file. There are also buttons for printing all variables or printing none. *Toggle selected* has the same function as double clicking a variable.

Selecting a constant or variable and pressing the Cut Data button *Properties* will open the dialog shown in Figure 19.

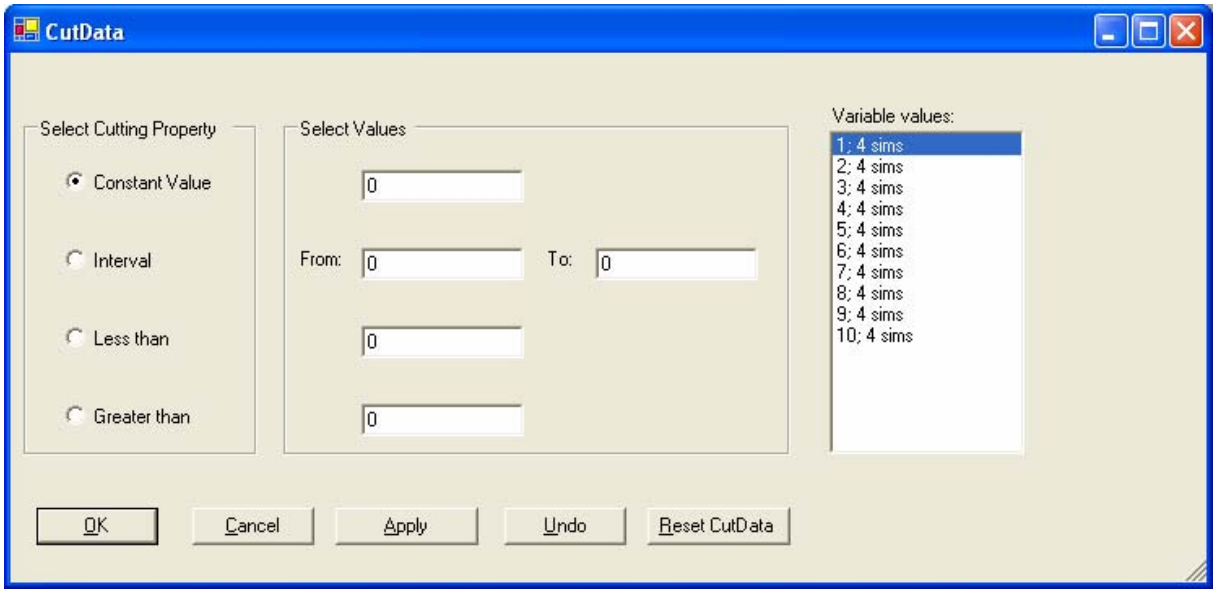

Figure 19 The CutData dialog

This is the *CutData* dialog that can be used to cut data in selected intervals to make the data file smaller. The list box to the right shows the values the variable has in the file and how many simulations of each value there are. The data can be cut in four different ways using the edit boxes in the middle of the dialog. A constant value can be selected. An interval can be selected where all values are left in the region set by the *from* and *to* edit boxes. Or finally a section can be selected less than or greater than a value set in the corresponding edit boxes.

The Reset Cuts button in the main interface will remove any performed cuts of the data and restore the settings to the original ones read from the file.

When the data has been modified through sorting and cutting it can be written to file. The second tab page in the main dialog interface is the *Write To File* page. This is shown in Figure 20.

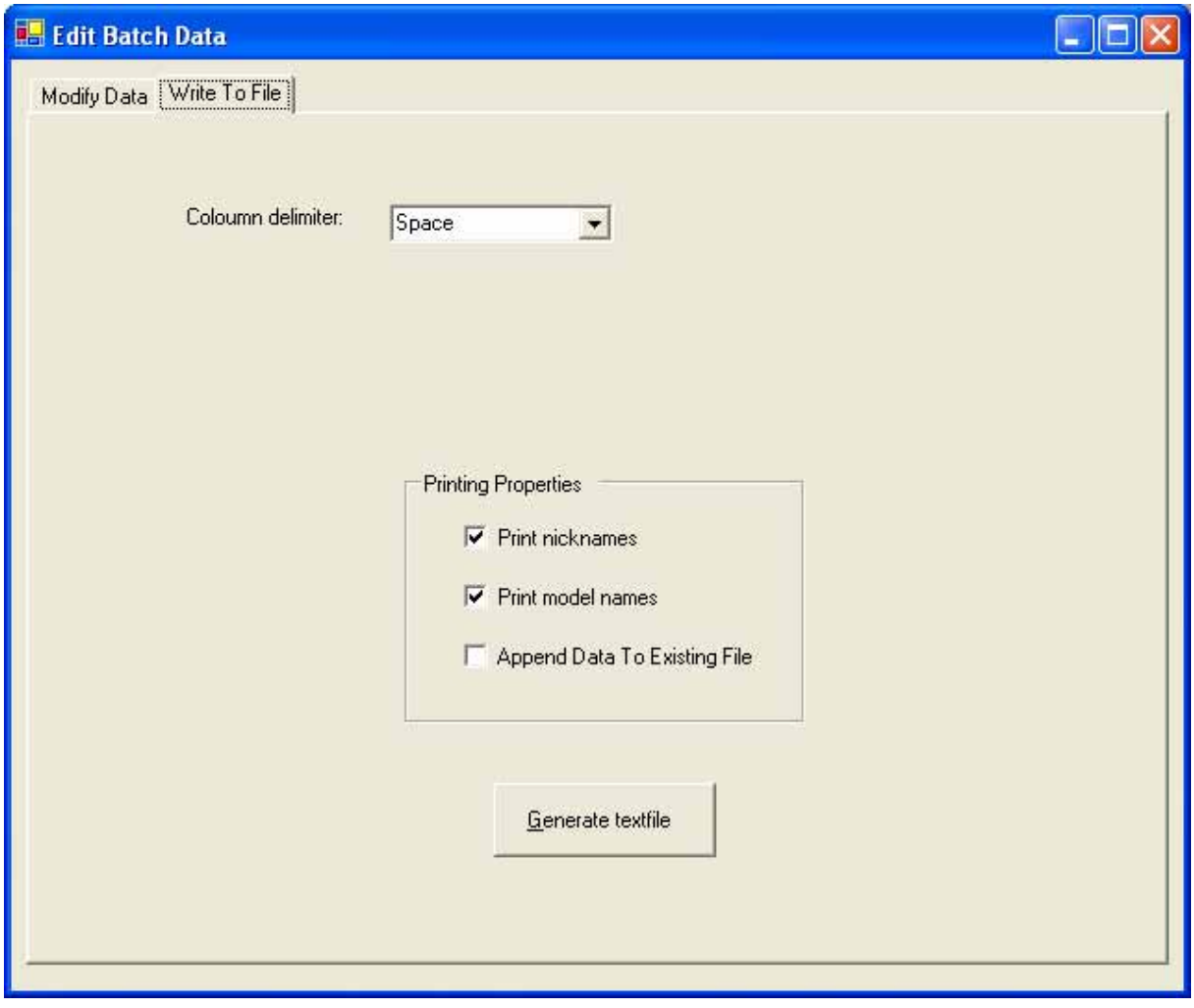

Figure 20 Write BatOrganizer data to file

Here the user can decide what column delimiter to choose. The options are: space, tab, semicolon and slash. The user can also select whether or not to write nicknames and model names at the top of the file. Finally there is an option whether to append the data to an existing file or to overwrite the file content.

3.8 Model variables

The following section describes how to choose the variables whose values are to be saved. A limitation is that only the final value of each variable is stored. However, if the values of every time step would be stored, the number of stored values would be too many in case of a large amount of simulations and a large amount of variables.

To generate a dialog for choosing the variables to save, push the *Variables to Save* button at the bottom of the main interface, see Figure 2.

The dialog generated will appear as shown in Figure 21:

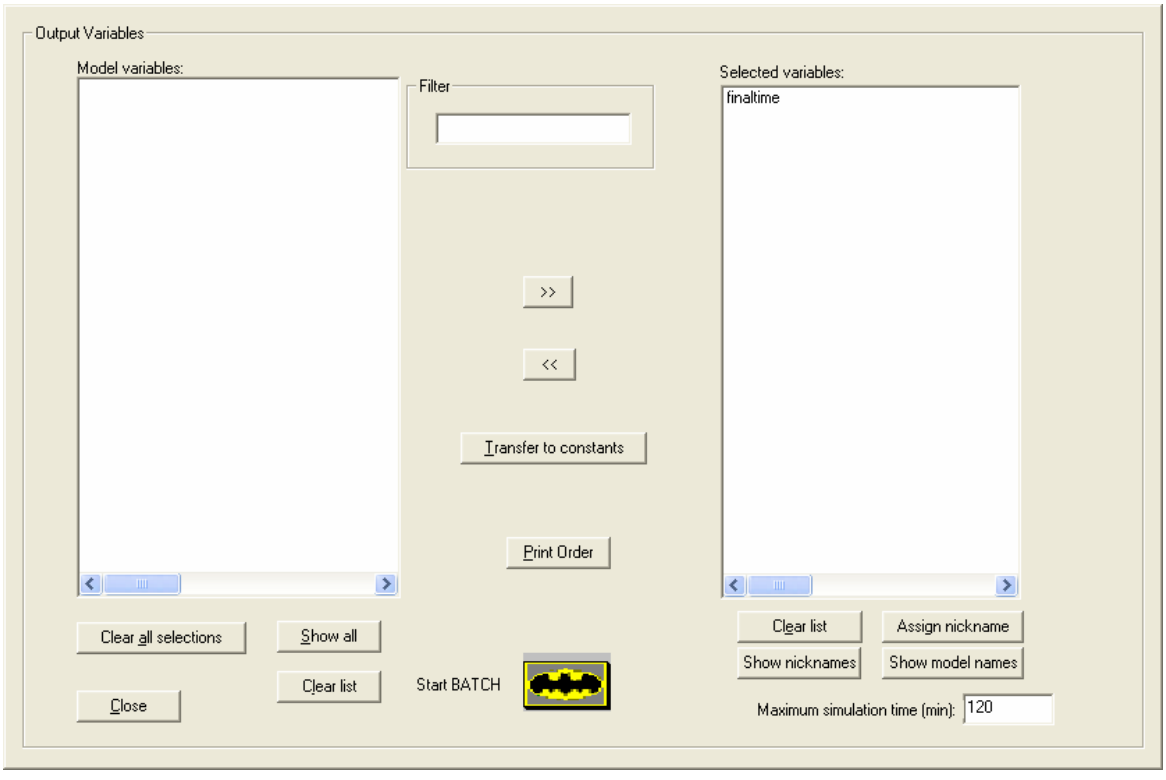

Figure 21 The output variables dialog

This is a modal dialog, which means that the main window shown in figure 1 can not be accessed as long as the output variables dialog is open.

On the left hand side of the dialog there is a list box entitled *Model variables*. This will be empty when the dialog first appears unless the dialog has been previously opened and saved. To show all of the model's variables, press the *Show all* button below the list box. The edit box entitled *Filter*, to the right of the list box, can be used to select which variables to show in the list box. This works in the same way as the filter function in the main window, see section 3.5.1. The button *Clear all selections* will remove the selections of variables that have been made in the list box (the variables will no longer be highlighted) and the *Clear list* button will remove all of the variables shown in the list.

To select the variables whose end values are to be saved, press the button to the right of the *Model variables* list box marked >>. The variables will appear in the list box entitled *Selected variables*. To deselect a variable, highlight the variable in the *Selected variables* list box and press the button marked <<. To deselect all of the variables, press the *Clear list* button below the *Selected variables* list box.

Beneath the *Model variables* list box are buttons to set nicknames for the selected constants. These work the same way as for the *Batch file constant list* list box as described in section 3.5.2.

The button named "Transfer to constants" will move a variable from the *Model variable* list box to the *Model constants* list box in the main interface. This in case the user would like to edit a variable that is not a constant in ACSL. It will have the same effect as using the ACSL runtime command "set" on a variable. This does not convert the variable to a constant. If the model sets the variable in the simulation the user-chosen value will be overwritten.

The button *Print Order* at the bottom of the dialog can be used to set the print order in the created simulation data. This is done through the dialog shown in Figure 22.

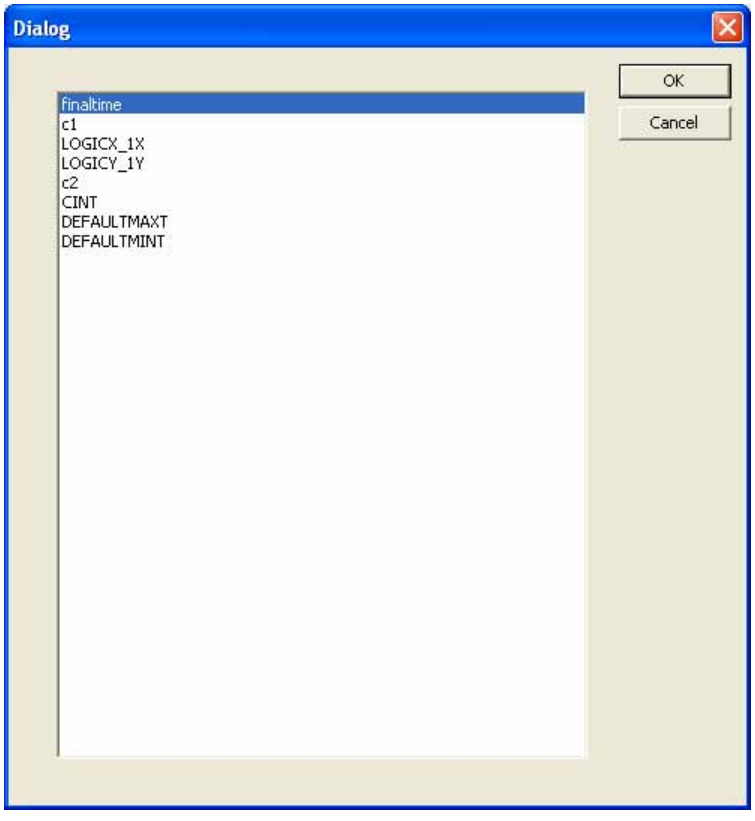

Figure 22 Print order dialog

The print order dialog will show the nicknames of all of the selected constants and variables in the BatchMan configuration. The print order is changed by dragging and dropping selected variables. The variables will be saved in the order of the list box in the simulation data file or the database depending on whether the simulations are performed distributed or not.

3.9 Starting a batch simulation

When all of the constants have been set and the desired variables have been chosen, the model may be run from the Batch Manager. The model will be run in a separate process. This means that if the model crashes the Batch Manager will not be affected. In case of a model crash the Batch Manager will immediately start the next simulation. In case the model gets stuck however, the simulation will be interrupted after the time interval set in the *Maximum simulation time* edit box, see Figure 21. This value is per default set to 120 minutes but can be changed by the user. To start the batch run press the button with the Batman entitled *Start BATCH*. After the batch run has been started, the Start batch button changes its text to *STOP BATCH*. By pressing this button the currently executed simulation will run to the end and after that the batch job is interrupted. The completed simulations up to that point will be stored in the simulation result data, which is stored in a text file or a database depending on whether the simulations are run distributed or not. However, restarting the batch run will start over the simulations from the beginning and not from where the simulation was interrupted.

3.10 Running BatchMan in a none distributed mode

As mentioned earlier, the simulations can be run distributed over a network or multiple processes. The results will be stored in a text file if the simulations are not run distributed and in a database if they are run distributed. This section describes how the simulation data is stored in a text file when BatchMan is run in the mode where the simulations are not run distributed, which is the default mode when the distributed dialog has not been edited, see section 3.11.3.2.

3.10.1 Result files

When the batch simulations are finished the values of the successfully completed simulations will be stored in a text file with the extension .dat, using a standard Windows “Save As”-dialog. The default textfile name will consist of the model name followed by ok.dat and a number that is increased for each batch of simulations performed. So if the model name is test.prj the results of the first batch simulation will be suggested to be stored in a file named test1ok.dat and the second batch simulation under test2ok.dat. If these files already exist, the user will be informed before any data is overwritten in standard Windows manner. If cancel is chosen, the user will be warned that all information will be lost before proceeding.

The ok.dat-file will list the chosen constants with their values and the chosen variables with their values stored from the model in the order specified from the print order dialog, see Figure 22. The first row will contain the nicknames of all the chosen constants and variables and the second row will contain the corresponding model names. Each row following that will contain the results from one simulation. The example below shows a file generated with four successful simulations.

```
xte                   NSTP(1)      tgtlockt0                time
CONTROLPANEL_1XTEIC   NSTP(1)      CTRLPNL_1TGTLOCK_T0      T
12000                 10           1                        45.0006
10000                 10           1                        45.0006
8000                  10           1                        45.0006
6000                  10           1                        45.0006
```

The variable tgtlockt0 is a boolean. This will be 1 for .true. and 0 for .false.. The variable NSTP is a vector. Vectors will be printed as vector name followed by left parenthesis, the index and then a right parenthesis. The last dialog after storing the result file will inform the user where the file has been stored.

Figure 23 Successful simulations search path

After the successfully completed simulations have been stored, the failed simulations, if any, will be stored in a text file named by the model name followed by fail.dat and index number analogous with the successful simulations, i.e. with the example test.prj the failed simulations will be stored under testfail1.dat. Here, the constants will be listed as before but no variables will be there since no values are available from the model. The file is created to let the user know which input values have made the model crash. The constant values will be followed by a column entitled “ACSL_Status”. Here “crash” means the model has crashed and “Timed Out” means that the maximum simulation time set by the user as described earlier, has elapsed. The example below shows a generated failure file:

```
xte                   NSTP(1)      tgtlockt0                 ACSL_Status
CONTROLPANEL_1XTEIC   NSTP(1)      CTRLPNL_1TGTLOCK_T0
12000                 10           0                         crash
10000                 10           0                         crash
8000                  10           0                         Timed Out
6000                  10           0                         Timed Out
```

Finally a dialog will show where the failed simulations have been stored, if this is the case, as with the successful simulations.

The failed simulations file will only be created if there are any failed simulations.

3.10.2 Backup files

There is a possibility to store simulation results incrementally before all simulations are finished. This feature is added in case BatchMan would crash before the simulations have been stored in the result files. To have BatchMan create backup files, open the backup dialog which can be found under menu alternative *Model->Backup*. The dialog is shown in Figure 24.

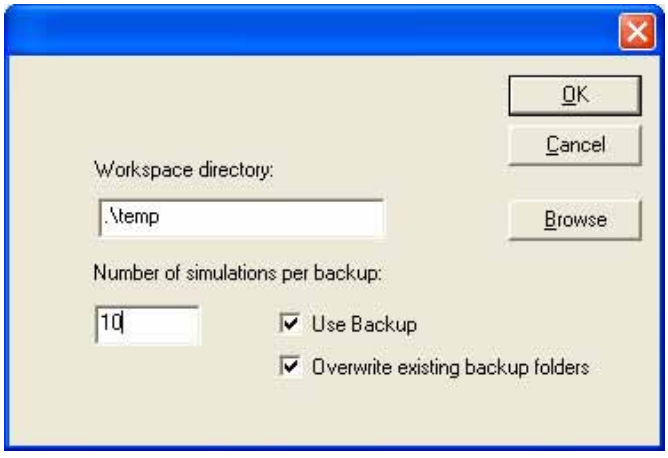

Figure 24 the Backup files dialog

In the *Workspace directory* edit box the search path to a folder, where the backup files will be created, can be set. In the example shown in the figure the relative path '.\temp' is used. This will create the backup files in the folder *temp* which is created in the current directory, which is likely to be the same as the ACSL model directory if the current directory has not been changed since the model was opened. One subdirectory will be created for each batch of simulations, so that if two batch simulations are to be created, the subfolders Batch1 and Batch2 will be created. A separate backup file will be created for each batch of simulations in these subdirectories.

In the edit box *Number of simulations per backup* the user can determine how many simulations will be created before the results are stored to the backup file. In the example 10 simulations are chosen which means that 10 simulations will be completed before any results are printed to file. If BatchMan is terminated due to a program crash this means that a maximum of 10 simulations can be lost. The advantage of not writing to file after every simulation is that is saves time. The results are appended to the same backup file every time BatchMan writes to the file. This means the backup file will contain the same number of results as the OK result file at the end of the batch simulation if BatchMan does not crash. However, the results are not sorted in accordance with the print order dialog in the backup files since this would be too time consuming. This means that the results have to be manually sorted afterwards, which can be done using the *BatOrganizer* tool, see section 3.7.

The default setting when the dialog is opened is that backup files will be created and that any occurring backup files in the selected folder will be overwritten. These settings can be altered with the two check boxes *Use Backup* and *Overwrite existing backup folders.* If the user selects to use backup files but deselects the *Overwrite existing backup folders* option, BatchMan will ask the user before any existing files are overwritten. However, this will halt the simulations until the user has made a selection and is therefore not recommended.

The backup option is only available when simulations are not run distributed, hence when simulation results are stored to file. When the data is stored to a database the backup feature is not necessary since the simulation results will be stored in the database after every successful simulation.

3.11 Running BatchMan in distributed mode

Since version 1.6 of BatchMan it is possible to run batch simulations distributed over a network or locally on a computer over several processes. In version 1.7 the results are stored in a MySQL database through ODBC[1] if the user request the simulations to be run distributed. This chapter describes how BatchMan 1.7 is intended to be used when it is run distributed over a network or multiple processes. The Appendices also contains short descriptions of how the programs work while running distributed, which components are involved and what prior installations need to be done on the network.

3.11.1 Messagequeuing

BatchMan utilises Windows Messagequeues to send data distributed. In order to run BatchMan distributed it is therefore necessary to install this service. This is done through the control panel, *Control Panel->Add Remove Programs->Add Remove Windows Components*. If BatchMan is to be run over a network it is first necessary to set the server to a domain controller, see section 3.11.3.1.

3.11.2 Database Batch in MySQL

3.11.2.1 Installation

In order to run BatchMan distributed the database software MySQL together with a couple of components needs to be installed. In Appendix 4.3 there is a description of how to install the database on the server or a stand alone computer.

3.11.2.2 Storing the results

When BatchMan is run distributed over a network or over multiple processes, the results from the simulations will be stored in two tables in the MySQL database Batch. The tables are named *TestSetTab* and *TestResultTab*. The two tables below, Table 1 and Table 2, show the appearance of the two tables in the database Batch:

BatchID	step	s1	s2	s3	s4	s5	s6	s7	s8	s9	s10	scontainer	systemname	starttime	date	username	sDescription	id
0	0	xte	yte	zte	''	''	''	''	''	''	''	''	SA-XX	153124	20050309	admin	'my descr'	1
0	1	x	y	z	''	''	''	''	''	''	''	''	SA-XX	153124	20050309	admin	'my descr'	2
1	0	xte	yte	zte	''	''	''	''	''	''	''	''	SA-XX	141223	20050310	admin	'my descr'	3
1	1	x	y	z	''	''	''	''	''	''	''	''	SA-XX	141234	20050310	admin	'my descr'	4

Table 1 TestSetTab

[1] ODBC, Open DataBase Connectivity, is a standard database access method which makes it possible to access database data from an external application using a middle layer database driver between the application and the database management system (DBMS). – http://webopedia.com/term/o/odbc.html

BatchID	step	d1	d2	d3	d4	d5	d6	d7	d8	d9	d10	scontainer	systemname	starttime	date	username	id
0	0	0.0	0.0	-1.0	''	''	''	''	''	''	''	''	SA-XX	153124	20050309	admin	1
0	1	1.0	0.0	-1.0	''	''	''	''	''	''	''	''	SA-XX	153124	20050309	admin	2
0	2	2.0	0.0	-1.0	''	''	''	''	''	''	''	''	SA-XX	153124	20050309	admin	3
0	3	3.0	0.0	-1.0	''	''	''	''	''	''	''	''	SA-XX	153124	20050309	admin	4
1	0	0.0	0.0	-2.0	''	''	''	''	''	''	''	''	SA-XX	141223	20050310	admin	5
1	1	1.0	0.0	-2.0	''	''	''	''	''	''	''	''	SA-XX	141223	20050310	admin	6
1	2	2.0	0.0	-2.0	''	''	''	''	''	''	''	''	SA-XX	141223	20050310	admin	7
1	3	3.0	0.0	-2.0	''	''	''	''	''	''	''	''	SA-XX	141234	20050310	admin	8

Table 2 TestResultTab

The first table, *TestSetTab*, contains the model names and nicknames of the chosen model constants and variables. The first row of a batch simulation contains the model names and the second row the nicknames. The first ten names in the BatchMan print order dialog, see Figure 22, are stored in the columns s1-s10. All proceeding names are stored in the string *scontainer* separated by spaces. When fewer than ten elements are stored, the remaining columns will contain empty strings.

The second table, *TestResultTab*, contains the results for the corresponding data, stored as doubles in the columns d1-d10. When more than ten variables are stored, the remaining results are stored in the string *scontainer* separated by spaces.

Each row in the table *TestResultTab* and each two rows in the table *TestSetTab*, correspond to a simulation. The variable *BatchID* is unique for a batch simulation. Step is increased by one for each row in a batch simulation. The column *id* to the far right is the primary key in the database and is automatically increased by one for each new row that is added to a table. The four remaining fields, *system name, start time, date* and *user name*, are set by BatchMan to make easier the identification of the right batch simulation for the user who is searching the database. System name is named by the prj-file name, hence SA-XX.prj will yield the system name SA-XX. *Start time* and *date* is the start time in hours-minutes-seconds and date in year-month-day when the batch simulation was started. Finally *sDescription*, in the table *TestSetTab*, is a string set by the user from BatchMan that can also be used to make the identification of the right batch simulation easier from the user interface LKZ [3]. *TestResultTab* does not contain the column *sDescription* since this does not add any additional information but is simply there to make the identification of the right batch simulation easier from LKZ.

TestSetTab and *TestResultTab* will contain the results of the successful simulations. These tables´ columns are static and will store all of the results for all of the users in a network. However, any failed simulations, due to ACSL model crashes or timed out simulations, will be stored in a failed data text file named *“<modelname><batch instance>fail.dat”*, as described in section 3.10.1.

3.11.2.3 Viewing the results

The recommended way to view the results after being stored by BatchMan is, as mentioned earlier, through LKZ that shows the results in form of a launch- or kill zone diagram. In version 1.7 of BatchMan and 1.5 of LKZ, there is unfortunately no way of generating text files in the same format as when BatchMan is run in the non distributed mode. This is suggested to be a feature in a later version of either LKZ or BatchMan. However, one way of

viewing the tables directly is to use Microsoft Access [4]. The values can be accessed in Access from the menu alternative *File->Open*. The file type is ODBC Databases(). This will yield a dialog as shown in Figure 25.

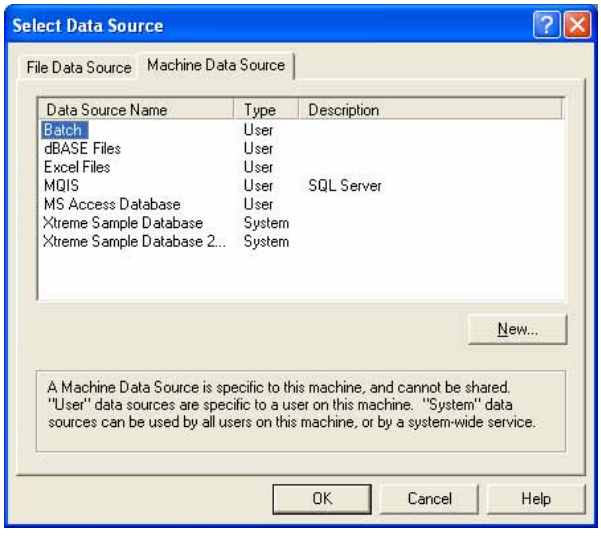

Figure 25 Select Data Source in Microsoft Access

Under the tab *Machine Data Source* the database *Batch* should be chosen. This will open the dialog *Link Tables* in which the tables *TestSetTab* and *TestResultTab* can be chosen.

3.11.3 To run BatchMan distributed over a network

3.11.3.1 To configure the network

Before BatchMan can be run distributed over a network, the network has to be properly configured. The server has to have active directory installed and be made domain controller. Also the service processes *RobinServiceKlient* need to be installed. Appendix 4.4 describes in more detail how the network needs to be configured in order to work together with BatchMan.

3.11.3.2 To run BatchMan

When the network has been correctly configured, BatchMan can be run distributed over the network. The following steps describe how BatchMan can be run over a network:

1. Share the directory folder containing the model that should be run over the network[2]. E.g. if the model is SA-XX.prj and is located in ‘C:\model\SA-XX\SA-XX-prj’, then ‘model’ must be shared over the network.
2. Start BatchMan and choose constant values as described in section 3.5. Give the batch run a description in the field sDescription that can be used to help to identify the batch run later.
3. Open the dialog *Set Distributed Path* from the menu *Run->Distributed*.
4. Choose *Run Distributed* to run the simulations distributed. If *Run locally* is chosen the simulations will be run locally on a computer and no service processes will be used. This is described in section 3.11.4.

[2] The user must have at least domain user privileges to share a folder over a network.

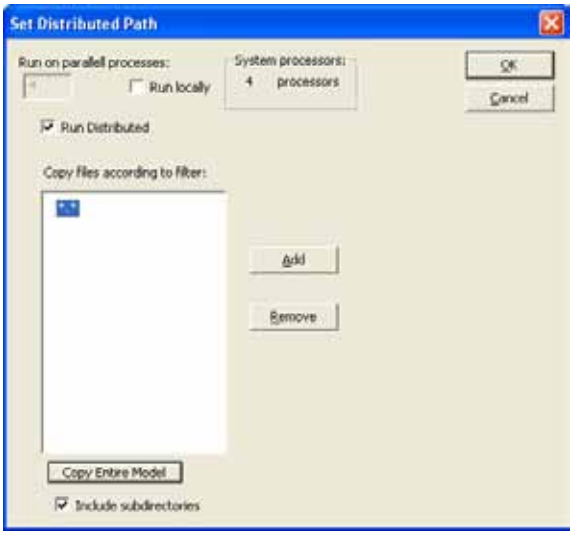

Figure 26 Set Distributed Path dialog

5. Choose which files are necessary to make the model run. In the example shown in Figure 26 the alternative *Copy Entire Model* has been chosen which means the directory containing the model will be copied together with subdirectories to a temporary working directory.
6. Finish the dialog with *OK*.
7. Output variables and print order is chosen as described in section 3.8.
8. Start the batch run by pressing *Start Batch* in the output variables dialog.
9. Start the service processes *RobinServiceKlient* on all computers that should run simulations. A list of the service processes can be found under *Control panel->Administrative Tools->Computer management->Services and applications->Services. RobinServiceKlient* should be there if installed correctly. Start the service process by right clicking it and select start. By starting several versions of *RobinServiceKlient*, multiple batch runs will be run in parallel processes on the computer in use. To start service processes on other computers than the client computer, choose menu alternative *Action->Connect to another computer* in the services window. A separate list of the service processes that can be started on the selected computer will be opened. Find the *RobinServiceKlient* processes as before and start as many instances as desired.
10. When the batch run is finished the service processes must be terminated manually, in version 1.7 of BatchMan, using the task manager. Also the temporary directories created by BatchMan must be manually removed. These are named *batchmandistrtemp* followed by an identification number. However, if a new batch run will be started later with the same model and the same chosen model constants and variables, but with new constant values, the service processes and the temporary model directories can be left since these will be reused by BatchMan.

In appendix 4.5 is shown the working flow of BatchMan 1.7 when it is run over a network. Understanding this can be of interest in order to memorize which steps need to be taken when running BatchMan.

3.11.4 To run BatchMan locally on a computer over multiple processes

When BatchMan is run locally yet distributed, the model is copied to different working directories and the work is divided over several processors run in parallel. Below follows a description of how BatchMan is run utilising multiple processors on a computer:

1. Install MySQL with components in accordance with the instructions in section 3.11.2.1 if this is not already done on the computer.
2. Start BatchMan and choose constants as described in section 3.5.
3. Open the dialog *Set Distributed Path* via the menu *Run->Distributed* and choose *Run locally* using the check box in the dialog.

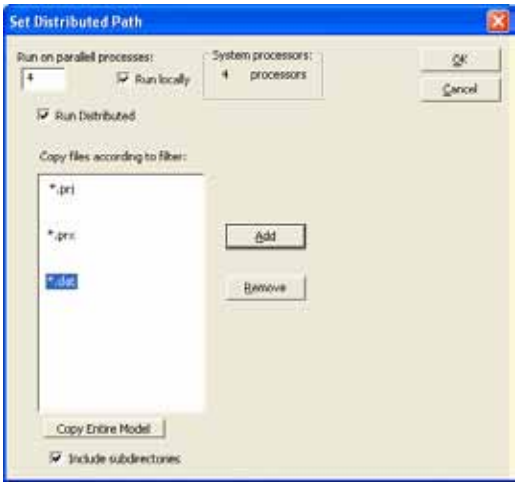

Figure 27 Set distributed path set to run locally distributed over several processes

4. Choose how many processes that should be run in the edit box Run on parallel processes. The default value is the number of processors on the computer. Tests show that the number of processors added by one or two extra processes is optimal if the computer should be optimally used.
5. Choose which files need to be copied, in order to make the model run, using the filter *Copy files according to filter*. In the example shown in the figure, all files with file extensions .prx, .prj and .dat in the model directory and its subdirectories will be copied to temporary working directories.
6. End the dialog with the *OK* button.
7. Select output variables and start a batch run in the same way as described in sections 3.8 and 3.9.
8. The simulations will be stored on the way as they are finished in the tables *TestSetTab* and *TestResultTab* in the MySQL database *Batch* as described in section 3.11.2.2.

The principle utilised when BatchMan is running distributed locally is quite similar to the network distributed case. The difference is that *RobinKlientPrivate* is created on the local computer which is done automatically by BatchMan instead of the service processes *RobinServiceKlient* that are started manually by the user on the computers in use. *RobinKlientPrivate* will handle the communication with BatchMan through private queues on the local computer instead of using public queues on the server. These private queues are created automatically by BatchMan if not already on the computer. The results however are stored in a MySQL database as in the network case.

Temporary working directories named batchmandistrtemp followed by a identification number is created by BatchMan. These are deleted automatically by BatchMan when all simulations are finished.

4 APPENDIX

4.1 CMD files created from BatchMan

BatchMan can be used to create command files, CDM-files, that can be run from ACSL/GM. Below is an example of a CMD-file with procedure name test created from BatchMan:

```
procedure test
 PREPARE /clear /all
 set CONSTANT_1O1  = 1.030000e+004
 set CONSTANT_2O1  = 1.000000e+000
 start
 PREPARE /clear /all
 set CONSTANT_1O1  = 1.030000e+004
 set CONSTANT_2O1  = 2.000000e+000
 start
 PREPARE /clear /all
 set CONSTANT_1O1  = 1.030000e+004
 set CONSTANT_2O1  = 3.000000e+000
 start
 PREPARE /clear /all
 set CONSTANT_1O1  = 1.030000e+004
 set CONSTANT_2O1  = 4.000000e+000
 start
end
```

4.2 Installing service processes

To install the service processes *RobinServiceKlient* on the computers in a network, the tool *Installutil.exe* can be used. *Installutil* is a part of Visual Studio. A description of how the tool can be used to install the service processes needed to run BatchMan distributed over a network follows below:

1. Using the Windows *Command Prompt*, navigate to the directory where Intallutil.exe is located.
2. Type the commandline: installtutil [search path to RobinServiceKlient.exe].
3. Repeat for every instance of RobinServiceKlient that is to be installed.
4. Adding the flag /u after ‘installutil’ will uninstall the service process.

4.3 Installing the MySQL database

Before any results can be stored from a distributed batch run, a MySQL database needs to be installed. This should normally be done on the server of a network but can also be done on a client computer if the results should be run locally and also stored locally on a stand alone computer. Below follows a description of how the database should be installed.

1. Install the latest version of MySQL server that can be downloaded from http://dev.mysql.com/downloads/.
- During installation choose:
    * user: root
    * pwd: batchmanlkz
2. Start MySQL Command Line Client
- Log in using pwd: batchmanlkz
- Create a database, Batch, with the commando: 'create database Batch;'
- Log out with the commando: 'exit'

3. Install MySQL ODBC Driver-MyODBC 3.51 which can be downloaded from: http://dev.mysql.com/downloads/connector/odbc/3.51.html.

4. Set up a MyODBC DSN:
- Choose *Control Panel->Administrative Tools->Data Sources(ODBC)*
- Under *User DSN*, choose *Add*
- Choose *MySQL ODBC 3.51 Driver* and then *finish.*
- Under *Login*, choose:
    * Data Source Name: ’Batch’
    * Description: optional
    * Server: ’local host’ server name if a network is used
    * User: ’root’
    * Password: ’batchmanlkz’
    * Database: ’batch’
- Finish by pressing *OK* twice.

This installation only needs to be done once for a network. Number 2 only needs to be completed in the server but number 3 and 4 need to be completed on all the computers that are to be used for distributed simulations.

4.4 Configure a network for distributed batch runs

Below follows a short description of how the network needs to be configured in order to run BatchMan distributed over the network. This only needs to be done the first time for a specific network.

1. A network where the server has Active directory installed is a required condition in order for BatchMan to be able to run distributed. Message queuing should be installed after the server has been made domain controller.
2. Two public message queues named batchqueue and resultqueue must be created manually on the network which is done through *Control Panel->Administrative Tools->Computer Management->Services and Applications->Message Queuing->Public Queues*. These message queues should be given *permission full control* to the group everyone.
3. Install *RobinServiceKlient* as service processes on the computers that will participate in the batch runs, which can be both client computers and the server. This can be done with Installutil.exe that is a part of Visual Studio, see Appendix 4.2. In order to make use of multiple processors on a computer, at least one instance of *RobinServiceKlient* for each processor should be installed. These must have different names. *SendMessageQueue.dll* must be in the same directory as *RobinServiceKlient*.
4. In the system folder, System 32, there must be a file called BatchMan.ini on every computer that will participate in the batch runs. The file must contain two rows where the first row is the name of the server in the network, and the second row is the full search path to *Robin.exe*. I.e. if the servername is myServer and *Robin.exe* is in the folder *Robin* under C:, then BatchMan.ini should look like:

   ```
   myServer
   C:\Robin\Robin.exe
   ```

   If the simulations are run locally on a computer distributed over the computer processors, see section 3.11.4, the results can optionally be stored in a locally installed database. In this case the server name should be `localhost` in BatchMan.ini. However, this is not a valid alternative when running through service processes since these are assumed to be run over a network via public queues in accordance with number 2 above.

Robin should be copied to the path stated by BatchMan.ini on each computer that will participate in the batch run. *Robin* is recommended to be copied to all computers that could be used in batch runs so that the file can be copied locally. The alternative would be to copy *Robin* over the network from another computer. In this case the search path in BatchMan.ini must be correctly set so that *Robin* can be located over the network.

4.5 Working flow of BatchMan 1.7 when run distributed

The figure below shows the working flow of BatchMan 1.7 when BatchMan is run over a network:

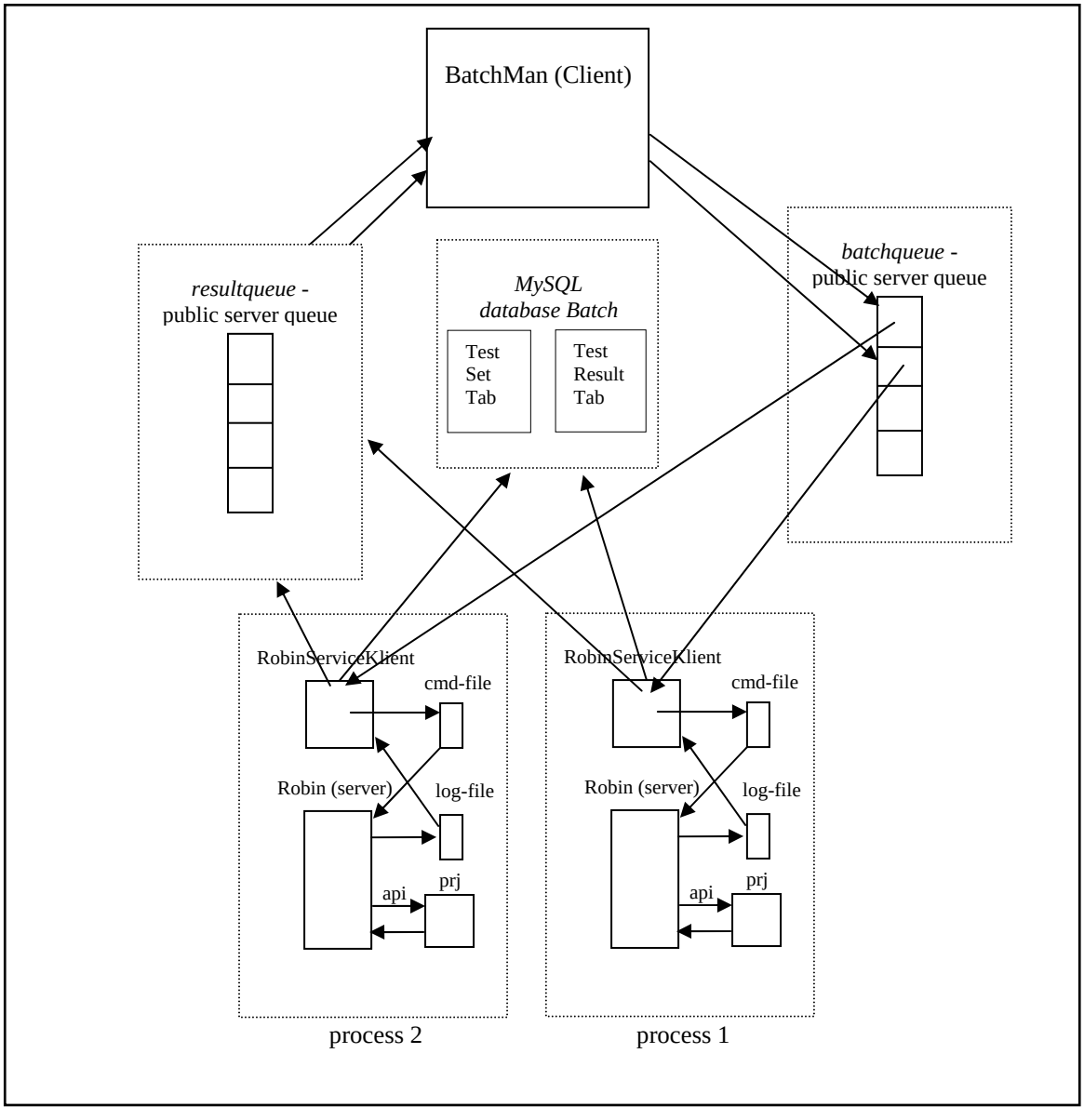

Figure 28 Working flow of BatchMan 1.7

RobinServiceKlient copies the model using a search path sent by a message from BatchMan and creates a local copy of the model on the service process computer. BatchMan sends messages to the public queue *batchqueue* on the server. *RobinServiceKlient* is a service process that should be installed on all computers participating in the batch run simulations. The service processes initially copies the model and then starts an instance of *Robin.exe* for every simulation to be run. The communication between *Robin* and the service processes is done through cmd-files and log-files. The separation of *Robin* in a client process and a server process secures that the system can handle a model crash since a new instance of *Robin* is created for every simulation. After a successful simulation *RobinServiceKlient* reads the result from the log-file created by the model prx-file and stores the result in the MySQL database

Batch. The results are stored in two tables, *TestSetTab* and *TestResultTab* described in section 3.11.2.2. The service processes also sends messages to *resultqueue*, which is the second public queue on the server. This message tells BatchMan whether the simulation was successful or not. BatchMan waits for the messages in *resultqueue* in a separate thread, which ensures that BatchMan will not be locked from waiting. When the result appears in resultqueue, BatchMan reads the message and registers a successful or failed simulation. Processes 1 and 2 in the figure can either be on the same computer or on different computers.

4.6 Files used by BatchMan

Below follows a description of the files used by BatchMan and how they work together. All files have been developed in Visual Studio by FOI.

- BatchMan.exe – the main program run by the user.
- RobinServiceKlient.exe – a service process that copies the model initially, gets and sends messages to the message queues and starts and communicates with Robin.exe. One instance for every process that will be started in parallel must be installed with different names. Five instances are included on the CD which allows up to five processes running in parallel on every computer in the network.
- RobinKlientPrivate.exe – is used by BatchMan when BatchMan is run distributed locally on a computer. BatchMan can create an infinite number of instances of RobinKlientPrivate which is determined by the user. One parallel process will be created for every instance of RobinKlientPrivate.
- SendMessageQueue.dll – an interface towards the messagequeues that BatchMan and RobinServiceKlient use.
- Batorganizer.dll – a dll that BatchMan uses to read and manipulate generated result data files in previous runs by BatchMan. The tool can be used to change the print order of variables and cut data to generate smaller data files.
- Robin.exe – a program started by Robinserviceklient or RobinKlientPrivate that runs the model prx-file through the ACSL API.

5 REFERENCES

[ACSL] ACSL (Advanced Continuous Simulation Language) for Windows, Version 11, 2000, AEgis Technologies Group Inc, USA.

[AFE] Eckerland Johnny, “AFE 2.3 – A general graphical user interface for simulation models”,
FOI-R--0048-SE, ISSN 1650-1942 , 2001.

[BATCHMAN] Verona, Mattias, “Batch Manager 1.0, A program for generating multiple simulations with ACSL®”, FOA-R--00-01623-616--SE, ISSN 1104-9154, 2000

[LKZ] Appleby Niclas, “LKZ – Launch Kill Zone 1.5 - A Data Visualisation Program for Multiple Missile Simulations”, FOI-R—1996—SE, June 2006

[VISUAL] Visual Studio .NET 2003, 2003, Microsoft Corporation, USA

[Xcellon] The Aegis Technologies Group, Inc., USA.