NUREG/IA-0433

International Agreement Report

RELAP5/MOD3.3 RELEASE Pre & Postprocessor

Prepared by:
Dr. W. Tietsch

Westinghouse Electric Germany GmbH
Dudenstrasse 44
68167 Mannheim
Germany

Kirk Tien, NRC Project Manager

Division of Systems Analysis
Office of Nuclear Regulatory Research
U.S. Nuclear Regulatory Commission
Washington, DC 20555-0001

Manuscript Completed: April 2013
Date Published: December 2013

Prepared as part of
The Agreement on Research Participation and Technical Exchange
Under the Thermal-Hydraulic Code Applications and Maintenance Program (CAMP)

Published by
U.S. Nuclear Regulatory Commission

NUREG/IA-0433

International Agreement Report

RELAP5/MOD3.3 RELEASE Pre & Postprocessor

Prepared by:
Dr. W. Tietsch

Westinghouse Electric Germany GmbH
Dudenstrasse 44
68167 Mannheim
Germany

Kirk Tien, NRC Project Manager

Division of Systems Analysis
Office of Nuclear Regulatory Research
U.S. Nuclear Regulatory Commission
Washington, DC 20555-0001

Manuscript Completed: April 2013
Date Published: December 2013

Prepared as part of
The Agreement on Research Participation and Technical Exchange
Under the Thermal-Hydraulic Code Applications and Maintenance Program (CAMP)

Published by
U.S. Nuclear Regulatory Commission

ABSTRACT

Since the introduction of the 32-Bit Windows Operating System technology for Personal Computers with INTEL CPU architecture Relap has been migrated from mainframe computers and UNIX workstations to the desktop personal computers. Due to the general structure of Relap the user interface however did not change in the course of this migration. Relap running on PCs still is command line driven and does not take advantage of the benefits of the windows environments. Inputs to the program and outputs have the same structure as the UNIX or mainframe versions. Therefore the classic way of performing analyses with Relap which consist of several, mostly iterative and consecutive steps did not change with PC installations of the program.

Westinghouse Reaktor mid of the 90th developed several tools running on the PC to support the analysts retrieving and documenting results of Relap analyses. Based on these tools and using the power of the object oriented technology and third party functionality for the graphics generation a Graphical Users Interface has been developed by Westinghouse Reaktor which integrates all steps of a typical Relap analysis process. This Pre & Postprocessor has been designed to run in all known Windows environments (Windows95/98, ME, NT4, 2000 and XP). It has the feel and look of typical Windows software and an intuitive users interface which only requires a minimum of training. The actual version 3.3.0 of the Pre & Postprocessor cooperates with the Intel Relap5Mod3.3Release version.

The present report describes the general features of the Pre & Postprocessor, the installation, the typical analysis process including restarts and strip cases and the data retrieval and visualization process. Additionally the Restart Postprocessor, which is a FORTRAN program with command line control, which is used by the Pre & Postprocessor running in the background is described and a source code listing of this program is added.

The Pre & Postprocessor currently is used at Westinghouse Reaktor GmbH as a standard tool and has proven to support the Relap analysis process and to enhance the efficiency of the analysts.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS

ATHLET	Analysis of Thermal-Hydraulics of Leaks and Transients
BFBT	BWR Full-Size Fine-Mesh Bundle Test
BWR	Boiling Water Reactor
CATHARE	Code advancé de thermohydraulique pour accidents de reacteur à eau (advanced Thermohydraulic Code for the Simulation of Accidents in Light Water Reactors)
CHF	Critical Heat Flux
CT	Computer Tomography
GSS	Generalized steady state
LOCA	Lost of Coolant Accident
LWR	Light Water Reactor
NPP	Nuclear Power Plant
NUPEC	Nuclear Power Engineering Cooperation
PWR	Pressure Water Reactor
RELAP	Reactor loss of coolant Analysis Program
RSME	Root Square Mean Error
TRAC	Transient Reactor Analysis Code
TRACE	TRAC&RELAP advanced computational Engine
US-NRC	U. S. Nuclear Regulatory Commission

1 INTRODUCTION

Relap like all of the other codes and programs used for safety evaluations and design calculations for nuclear installations have been developed in the past on big mainframe computers. All of these programs have several common characteristics. The general structure of the programs is similar. They are programmed in FORTRAN supplemented by machine dependent environments. Input to the programs in general is organized by cards or data files with no direct dialog between the programs and the user. The programs are command line driven and designed to be run as batch cases. The output in general is organized in the ASCII format and stored in files or being printed on line printers. Running these programs, especially Relap, thus always required special user dependent setups on the computer platforms and to the best, task dependent shell scripts.

The classic way of performing analyses with Relap ever since consisted of several, mostly iterative and consecutive steps. The two major steps of the analysis methodology with respect to work effort include input preparation and the extraction and documentation of the calculated results from the often voluminous output files or printouts. Especially for the presentation and documentation of important data in the form of time history plots quite early general purpose plotting tools have been developed. In the early days these programs were running on the mainframes also and were designed to read the ASCII formatted output files and to further process these data for the visualization on screen (e.g. ReGIS Graphics on VT-Terminals) or for plotting graphs on line-printers or on Calcomp plotters.

With UNIX Workstations becoming more and more powerful in the late 80’s the FORTRAN programming language very much supported the migration of Relap and other nuclear safety codes from mainframe computers to UNIX workstations. The general handling of Relap on UNIX machines was somehow simplified by the better access of the users to the computers and with the development of graphical user interfaces (X-window) built into the UNIX operating systems. Based on these enhancements some users of Relap for example developed user interfaces to Relap utilizing the Tcl/Tk package. One of the first programs for the post-processing of Relap results which uses the X-windows power was integrated into the Nuclear Plant Analyzer (NPA) package (/1/), developed by EG&G in Idaho Falls under the sponsorship of the U.S. Nuclear Regulatory Commission. The NPA provided the analysts with tools which allowed him to visually display static or dynamic data on a graphic workstation. However NPA as a simulation tool primarily has been designed for the display of dynamic data with the capability of controlling data sources like Relap. Static data (graphs of parameters) could be displayed on screen or printed by postscript printers, however with a limited performance. Also the plotting of graphs was not very flexible. Later (1994) the NPA internal plot package was upgraded to allow for external plotting programs also. The latest version of NPA (1.4a) was running on all of the major-brand UNIX workstations at that time. Since 1995 NPA however has not been developed further. Especially the program has not been migrated to INTEL operating systems. Now it is the NRC plan to replace NPA totally by SNAP (/3/).

Rather in parallel to the NPA development the ACE/GR data extraction and plotting software has been introduced by Paul J. Turner (/2/), which has been especially upgraded by Ken Jones to provide an interface to the Relap restart and strip data (Xmgr5). This software originally also run only on UNIX systems and requires a X-windows server. Because of the flexibility and versatility of this program and the capability to read binary restart files, Xmgr5 more and more evolved as the standard post-processing tool for Relap.

With the onset of the “PC-Age” and the replacement of the line or graphics terminals by PC’s, which were connected to the mainframes by networks, part of the analysis process (input preparation, visualization and plotting of Relap data) moved to the PC’s. The usage of the standard PC Software for these tasks increased the effectiveness for this parts of analysis work. However these standard tools often required separate interface tools, e.g. for UNIX file-format conversion or for the extraction of plot data formatted to meet the requirements of the plotting software in use. With the fast development of PC-based Network Client Services X-windows server were running on PCs also. This allowed Xmgr5 (running on the UNIX platform) to be used for data post-processing with this tool on a PC. However there still was a problem in the generation of paper plots. Xmgr5 required plotting hardware to be attached to the UNIX computer.

As the INTEL processors became more powerful, as a logical step Relap was modified go run on INTEL-PCs. Already back in 1993 Westinghouse Reaktor developed a fully tested and verified special version of Relap5MOD1 for Intel –386/486 and Pentium processors which was based on a special IBM RISC-6000 version of Relap5MOD1 Cy19 (/4/). Since that time Westinghouse Reactor also developed tools running on the PC to support the Relap analysts. Today, Relap Intel installations (Windows OS or LINUX) are as common and widely used as UNIX versions on workstations and the future trend points to Relap running on INTEL systems. Among others one main reason for this evolution is the substantially decrease of cost of PC components and the steady increase of performance of PC-systems which currently already exceeds the performance of much more costly UNIX-systems.

The typical procedure for Relap calculations in a PC-environment for the time being, which is quite similar to mainframe and UNIX installations, looks like this:

- Relap, still a pure non-Windows program runs in a “DOS Box”
- Input editing is performed by NotePad or special Programmers Editor
- Relap is run by command line or with help of batch-scripts (base or restart cases)
- Results are stored in the Restart-Plot files or the Major Output file

Minor Edit data cannot be extracted from the Main Output without a special tool. All of the plot data is stored in binary form in the Restart-Plot file. The normal procedure to get the interesting data is the stripping process or alternatively applying Xmgr5 (Intel version with X-Windows server running on the PC)

- Generation of strip run inputs
- Run Relap strip case / cases

In the ASCII Strip output file the data is arranged as blocks of parameter data for each time step. Special software is needed to read these data blocks and converse the data into columns which can be directly processed by most of the plotting programs.

- Run the Strip data extraction program
- Run the plotting software and visualize or print the data

The alternative to the post-processing of the Relap data as described above is using the Intel version of Xmgr5. This version is able to read the binary (Intel) Restart-Plot files. The problem with this program however is that it needs a X-Windows server running on that same PC and preferably needs a UNIX simulator like the CYGWIN package. Commercial X-Windows servers however are expensive, freeware servers are not supporting all of the features of Xmgr5.

It can be stated that running Relap on a PC still is a multi-step procedure like running Relap on UNIX platforms or mainframe computers. The advantage of a PC installation is that PC hardware today is much cheaper and more efficient than UNIX hardware. Additionally the usage of standard PC-Windows-Software for the data printing and for documentation like the Office Package (EXCEL) only requires a minimum of additional training.

Our (Westinghouse Reaktor's) goal since the first installations of Relap on PC-systems was to equip Relap with a Graphical Users Interface (GUI) which integrates the total analysis process as described above and makes the handling of Relap as convenient as the usage of other typical PC-programs. The main driver for developing this GUI of course was the expected increase of effectiveness of the analysis process by providing the analysts a tool which is intuitively and easy to use and which supports him the best in performing all of the analysis tasks.

Westinghouse Reaktor GmbH has presented the Relap5 Pre- and Postprocessor (RelapPP) to the CAMP community at the Fall 2000 and Fall 2001 CAMP meetings. Currently this tool works together with the latest Relap5Mod3.3 (xx)-Release version. The latest version of RelapPP33 has been presented at the Fall 2002 CAMP meeting in Alexandria, VA. The program is widely used within Westinghouse and other organizations and has proven to act as a useful add-on to Relap. The program is free to all CAMP members.

2 DESIGN CONSIDERATIONS AND FEATURES

The main goal for the development of the RelapPP Pre- and Postprocessor was to provide Relap on PC platforms an integrated Graphical Users Interface (GUI) which contains all of the necessary tools needed to perform all parts of standard Relap analyses. This GUI should act as a shell to run the command line Relap and additional programs needed and should cover the total analysis process from input preparation to the generation of paper plots for the documentation of Relap results. The Pre- and Postprocessing tool thus should support the analysts in performing Relap analyses. It should increase effectiveness of the analysis process, enhance the quality of the results and last but not least should save money.

To meet these goals the following design considerations were taken:

- The program should run on all standard PCs with all 32 Bit (and up) Windows Operating Systems.

- The installation of the software on the PC should be easy and user specific like most of the Windows applications and should not require administrator rights. The program should make use of the Windows InstallShield process.

- The GUI of the Pre- and Postprocessor must provide the same Look and Feel and the functionality of standard Windows applications in order to minimize the training for the usage of the program.

- The graphics and plotting interface of the program should allow easy access to all of the Relap data including Minor Edit data from the main output, strip data and plot data from restart files.

- The graphics and plotting interface should provide plotting data in common ASCII format for the exchange of this data with external plotting tools like Xmgr5 and should provide plotted graphs that can easy be integrated into other Windows applications like WORD or EXCEL for documentation purposes and printing of these graphs using the Windows printing capabilities.

- The basic graphics and plotting software should be integrated or imbedded into the Pre- and Postprocessor. The user must be able to setup a personal plotting environment.

- The Pre & Postprocessor should support the analysts with tools and means to easily manage all of his Relap projects and all of the tasks coming along with the performance of Relap analyses.

- All of the Relap documents should be provided to the analyst in electronic form online during the analysis.

- The Pre- and Postprocessor should cooperate with the latest version of Relap5 (Mod3.3(xx) Release)

Based on these principal design considerations the Pre- and Postprocessor has been developed using the MicroSoft Visual Studio programming environment to ensure the best fitting into the MicroSoft Windows world and the best outlook for further smooth development and easy enhancements without compatibility problems.

The RelapPP Pre- and Postprocessor for the Intel Versions of Relap5Mod3.3 Release (RelapPP33R) has the following general features:

- RelapPP33R runs on all standard PCs with nowadays hardware equipment

- RelapPP33R requires the Operating Systems Windows 95 / 98 / ME / NT4.0 / 2000 / XP

- RelapPP33R requires EXCEL95/97 or 2000 / 2002 as Graphics Engine for the embedded plotting of graphs or any other Graphing Software with ASCII plotting data import

- RelapPP33R provides a shell to the Relap Intel Command-Line Version (3.3 Release) with the following sub functions:

- Organize Relap projects, containing inputs, main outputs, strip runs, restarts and all of the different plot data of the different projects
- Make, edit and view input files
- Automatically generate Restart input files
- Automatically generate Strip input files
- Organize restarts
- Organize strip runs
- Set and manage Command-Line Parameters for Relap Runs
- View Input Manuals during Input Preparation
- View Output and Log Files
- View Restart and Plotdata information files
- Compare Output and Input Files using integrated DIFF-function
- Convert Capitals in Relap inputs to lower case
- View the total Relap documentation including the guidelines

- The post-processing of the Relap data is performed by:

- Reading the Minor Edits data from the Relap Main Output and reorganizing the data format to an interchangeable ASCII format (PLO-Format).
- Merging of Plot data from Minor Edits of successive restart runs.
- Reading the Strip Data format and reorganizing the data to the PLO-format.
- Reading the binary plot data from Restarts (Version 3.3 Release) either from Intel plat form or from any other Unix platform and reorganizing the data to an ASCII format (PLOUT-Format) which is used for further postprocessing of this data.
- Selection of parameters to be printed or plotted.
- Generating graphs and Plots from all data sources using the embedded EXCEL Spreadsheet and Graphing Engine.
- Full Excel Plot functionality for the graphs.
- Individual users environment for graphing by EXCEL functions.
- Export of graphing data in three different ASCII data formats (including PLOT Software and Xmgr5).

- Configuration files (INI-Format) for the RelapPP33R environment, for general plot de sign and for the Units of Parameters needed for graphing and data printouts.
- RelapPP33R uses the Standard Windows-Editor Notepad for small files and Pad97 for large Files. Any other Editor can be used by configuration. For very small files the application contains a simple internal ASCII editor.

- RelapPP33R makes use of the Acrobat Reader, Version 5 or up, which has to be installed on the system or will be installed on request during the installation process of the application.

- On demand an external very flexible large file editor (PAD97) and a file management tool will additionally be available. Any preferred other editor can be used.

3 PROGRAM DESCRIPTION AND FUNCTIONALITY

RelapPP33R (RelapPP) is an integrated pre- and postprocessor for the Intel Version of Relap5Mod3.3Release and runs in a typical Windows PC environment. The program communicates with the user by a graphical users interface (GUI) which has the look of a standard Windows application. The task of performing analyses with Relap can be subdivided in subtasks, e.g. input preparation, run Relap, post-processing of results and plotting graphs. The menus, submenus and task windows (Modules) are designed to group all of the necessary subtasks in such way that all tasks for the different steps of a Relap analysis can be accessed within one window. This proceeding helps to support the Relap analyst in performing his task without much training. A detailed description of the program and of all the modules built into the application is given in the following chapters.

3.1 Program Architecture

RelapPP is a Windows application. The development of the GUI has been performed by using the advanced object oriented programming techniques of the MicroSoft Visual Studio Package. Most of the file-handling, the project management, the variable control and management and the not time-sensitive data handling have been programmed using the BASIC language. Some tools like Relap itself and the Restart Postprocessor are programmed using the FORTRAN language. These programs are being called as external programs by the Pre- and Postprocessor and run in the fore or background.

Am important programming principle followed with the development of the application was that all of the functions and subroutines uniquely used by one Form (Task Window) are coded together with this form. All Subroutines and functions used by several forms are put together in common modules. The same is true for the variables and arrays. The program in total contains 18 forms and 4 common modules.

Object oriented design generally allows simultaneous operation of different tasks. This principle in general is valid for RelapPP also. It is designed like many Windows applications as a MDI (Multiple Document Interface) application. In the Pre & Postprocessor however some tasks depend on data being set or changed by other tasks (e.g. configuration changes). For these cases a built in concurrency control manages possible conflicts between different program tasks. The protection against conflicting access and data corruption e.g. does not allow the user to perform tasks from the menus or open additional windows as long as possible conflicts exist. The user is not especially alerted when certain menus or data is inaccessible.

In a multitasking MDI application the main window, the top level window is called the Shell-Window. This main window which pops up after starting the program, contains all of the menus in a menu bar at the top and a status bar at the bottom. This ShellWindow is a container of all of the ChildApplicationWindows which are subordinate and secondary windows for the separate tasks of the application. The principal structure of the program is shown in Figure A1 (Annex A).

The ChildWindows of the RelapPP are structured in this way that they contain all of the methods and controls to perform all of the main and subtasks described in the preceding paragraph. The main tasks of the application are:

- Perform a new Relap analysis (new Relap project)

- Continue the current Relap project

- Post-process old Relap projects and extract data results from any other Relap run.

From the main menu there is also access to all of the Relap documentation by launching the Adobe Acrobat Reader and access to configuration tools, utilities and options.

The most important form of the RelapPP is the Relap Command Center which contains all of the tools and provides access to all external programs and functions for the three main tasks of Relap analyses:

1. Setting up and running Relap Base cases

2. Making Restart Input files and performing Restart cases

3. Making Strip Input files and performing Strip cases

The Relap Command Center runs the Relap Code in a Command Window (DOS-Box). The Relap Screen Output is saved to a Log File. The progress of the calculation can be watched online. The Command Center also runs the external Restart Postprocessor, however in the background (silent mode).

From the Command Center there is access to all postprocessors, the Output Postprocessor for the Minor Edit Plotdata, the Strip Postprocessor for the Plotdata in the Relap Strip Data format and the Restart-Plotdata Postprocessor for the post-processing of the binary Plotdata contained in the Restart files. Additionally the Command Center integrates some help functions to support the analyst. The Postprocessors convert the Plotdata into a common Format (PLO-format). Additionally to the PLO data format the parameter information for the PLO file are documented in a PAR file. The PLO format and a typical PAR file are shown in Table B1. Parameter files from other data sources (PLOUT/Restart or Strip data) have additionally a units column as the units of the parameters are usually not provided with these data (Table B2).

The conversion of the binary data from the restarts is done by a 2-step process:

The binary data is extracted first from the restarts with the help of the external Restart Postprocessor (Command line controlled Fortran program). The Plotdata is saved in the ASCII PLOUT format (Table B3). This data format can be used directly for the visualization of interesting parameters and the plotting of graphs (Data Visualization Module).

Mostly however the restarts contain a huge number of variables. The direct Visualization may therefore take some time to process each parameter. In this cases it is better to extract first only the parameters of interest and then save them in the PLO format. This is performed by the “Selection of ...” - Module. The actual limit of parameters for the direct post-processing of PLOUT data with the RelapPP is 32768. If a Relap case has more parameters a warning is given and only the first 32768 parameters are processed. In this cases it is recommended to use the stripping function of Relap to extract the plot data of interest from the restarts. The usage of the Relap stripping also is recommended for big restart cases (restart files > 2 GB).

The PLO format originally was designed to represent all of the max. 99 Minor Edit variables, however rearranged compared to the Main Relap Output in order to have only one line of data for all of the parameters for each time step. This format is easy to handle and specific data is easy to extract. Additionally the data can be viewed by any editor with no line length limits (like the PAD97 editor, which comes with the installation package).

Once the data is converted to the PLO format or the plot data exists in the Strip data format (max 999 parameters), parameters of interest can be selected. These groups of parameters can be visualized, exported in three different formats or plotted with the help of the integrated EXCEL engine.

3.2 Managing Relap Projects

Relap projects usually contain a lot of different files where the user sometimes can get lost, especially when projects are named quite similar. In RelapPP a Relap project is defined by a “base”-name. All of the other files affiliated with the project are named at front of the filename with the base name appended by some additional characters, figures and a filename appendix. This guarantees a clear identification of the type of file. All of these files may reside in one directory (“user” directory). RelapPP is designed that only those files correlated to the current project are accessible if the “Run a Relap Project mode” is chosen. The Relap user however can create additional directories, one for each project in order to separate more clearly the files of the different projects. This proceeding can be better compared to the often found workspace organization of projects. Both of the methods are supported by the Pre- and Postprocessor. In RelapPP subprojects, e.g. for parametric studies are not supported either. Each Base Input file defines a new project. Subprojects however can be organized by using different subdirectories as described above.

RelapPP is a 32-Bit Windows application and in general the naming convention of Windows applies. RelapPP will warn the user if the file names are too long (>40 characters). In the Windows environment the file name length is not really limited (256). The Relap program itself currently defines this limit. The Restart-Postprocessor (Fortran program) limits the length of the file names to 60 characters. If there is a problem with too long file names there are the following possibilities to solve this problem: a) the file appendices should be as short as possible (i = input, o = output, etc.), b) RelapPP should be installed in the main root of a partition (e.g. G:\R5\....) and c) the project name should be shortened. It is recommended to use a meaningful name which describes the project precisely and does not exceed 8 characters (as the DOS convention).

A Relap project usually consists of one Base Case only.

The different files are organized and named as follows:

- Base Case	*	
- Input File	*.i	(Suffix changeable)
- Output File	*.o	(Suffix changeable)
- Restart File	*.r	(Suffix changeable)
- Log File	*.scn	(Suffix changeable)
- Restart Input Files	*_Rxx.i	(Suffix changeable)
- Restart Output Files	*_Rxx.o	(Suffix changeable)
- Restart Log File	*_Rxx.scn	(Suffix changeable)
- Strip Input Files	*_Sxx.i	(Suffix changeable)
- Strip Output Files	*_Sxx.o	(Suffix changeable)
- Strip Files	*_Sxx.str	(Suffix changeable)
- Restart Information File	*.rsinf	
- Restart Plot Data Information File	*.plinf	

A Relap project also contains the output files of the different postprocessors

- Minor edit Data from Main Output	*.plo	
- Parameter file for PLO files	*.par	
- Ascii Plotdata from Binary Restart	*.plout	
- Plot Data from PLOUT in PLO Form	*_xx.plo	
- Parameter file for this PLO files	*_xx.par	
- Strip Files	*_Sxx.str	(Suffix changeable)
- Parameter file for Strip data file	*_Sxx.par	

Five of the file suffixes are user changeable by the configuration option (s. Chapter 3.5). These suffixes and the base case name of the current project are stored in the configuration file (INI-File) on request and at quitting the program. The restart runs (max 99) are characterized by a "_R" followed by a counter. It is recommended to save the original restart file (e.g. of a steady state run) if a lot of test calculations shall be performed with using that restart as a basis. Relap replaces the original restart and appends the new data for each restart run. In some cases it is not desired to use these modified restart files as basis.

The Strip runs (max 99) are characterized by a "_S" followed by a counter. For each restart run RelapPP offers to generate a Restart Information file and a Plot Data Information file. These files are identified by the appendices "*.rsinf" and "*.plinf". Both of these information files are produced by the external Restart Postprocessor. The data contained in these files give a quick overview over the restart run and the data available. The information is needed for the automatic generation of the restart input files and of the Strip input files. The contents of these files are shown in Table B4.

The Output Postprocessor (Minor Edits) generates the plot data files (*.PLO) and for each PLO file a parameter file. The Strip Postprocessor generates a Parameter file with naming relation to the Strip File. Each time the Restart Processor is launched for the extraction of Restart Plot data both files RSINF and PLINF are made also. The naming of the PLO files which are the result of a parameter selection process of Restart Plot Data (PLOUT) files are named with a "_" and a counter.

A Relap project in total can consist of:

- 1	Base case input file
- 1	Main output file
- 1-99	Restart Input files
- 1-99	Restart Output files
- 1	Restart file (binary)
- 1-99	Plot Data file (PLO) for Minor Edits
- 1-99	Par files for Plot Data from Minor Edits
- 1	ASCII Plot Data File from Restart (PLOUT)
- 1	Restart Information File
- 1	Restart Plot Data Information File
- 1-99	Plot Data files from PLOUT in PLO Form
- 1-99	Par files for Plot Data from PLOUT
- 1-99	Strip Input files, Strip Output files and Strip files
- 1-99	Par files for Strip Data files

Thus RelapPP can handle complex projects of up to 1000 different files.

3.3 Main Functions of RelapPP

The principal structure of the RelapPP Pre- and Postprocessor is shown in Figure A-1 of Annex A. The functionality of each of the ChildApplication Windows (each of them can contain additional sub or child windows), are explained in the following chapters.

3.3.1 Overview

These are the main tasks of the RelapPP:

Top Level Window

- Welcome Window
- Basic directory setting
- Access to all functions via menus
- Status bar informations

Integrated Relap Shell (Command Center)

- Input preparation (manual, automatic)
- Run Relap, Command line controlled
- Access to Relap Documentation
- Access to all Postprocessors
- Run statistics and information for the user

Output Postprocessor

- Extract Minor Edit Data from main Output and write all Plot Data in PLO format
- Write parameter Data in PAR format
- Link to the Plot Postprocessor for the selection of parameters and groups of parameters to be plotted, printed or visualized

Strip Postprocessor:

- Extract the Parameters data from the Strip files and write the data in PAR format
- Link to the Plot Postprocessor for the selection of parameters and groups of parameters to be plotted, printed or visualized

Restart Postprocessor:

- Extract all Plot Data from the binary Restart file and write the data in ASCII format by the external program RestR533R
- Extraction of up to 99 Parameters and write in PLO format
- Link to the Plot Postprocessor for the direct selection of parameters and groups of pa rameters to be plotted, printed or visualized (similar Xmgr5)

Plot Postprocessor:

- Selection of Variables and groups of variables
- Print selected Parameters in ASCII TXT Format
- Print selected Parameters in PLT Format
- Print selected Parameters in Xmgr5 Format
- Plot Data/Graphs with integrated EXCEL
- Automatic transfer of Data to full-scope EXCEL

Options, Utilities and Add-On Programs:

- Configuration management (INI - Files for Environment, Plot-Defaults, Units of Parame ters)
- DIFF-Utility
- Conversion of Capital characters to lower case in input files
- Online Documentation
- User and Debug Options
- Additional programs

3.3.2 Top Level Window

The Relap Pre- and Postprocessor starts with the Main (Top Level) Window shown in Figure A-2 (Annex A) with a centered welcome screen which displays the actual date, the Version of RelapPP, the version of the Relap program and copyright notes. The status bar at the bottom contains the actual project informations (Project name, Relap Rood directory, the users directory and the path where all of the executables are installed). If the application is run the first time the informations in the status bar may be not correct. This is indicated by different color (red) or “invalid”. In this case the configuration has to be performed first. The procedure is explained in chapter 4.

The menus for all of the functions of RelapPP are arranged in a Menu Bar at the top. The menus are grouped with respect to the tasks and are designed as anchored pop-up menus. There are five Top Level Menus:

1. Project
2. Relap Command Center
3. Data Visualization
4. Documentation
5. Utilities and Options

Additionally there is Information accessible by clicking on “?”, which shows the copyright, the actual sub-version of the application and a serial number.

The Project Menu is subdivided in a “Paths & Directories” option and two submenus: “New Relap project” and “Current Relap Project” (see Figure 3.3-1)

Current Project

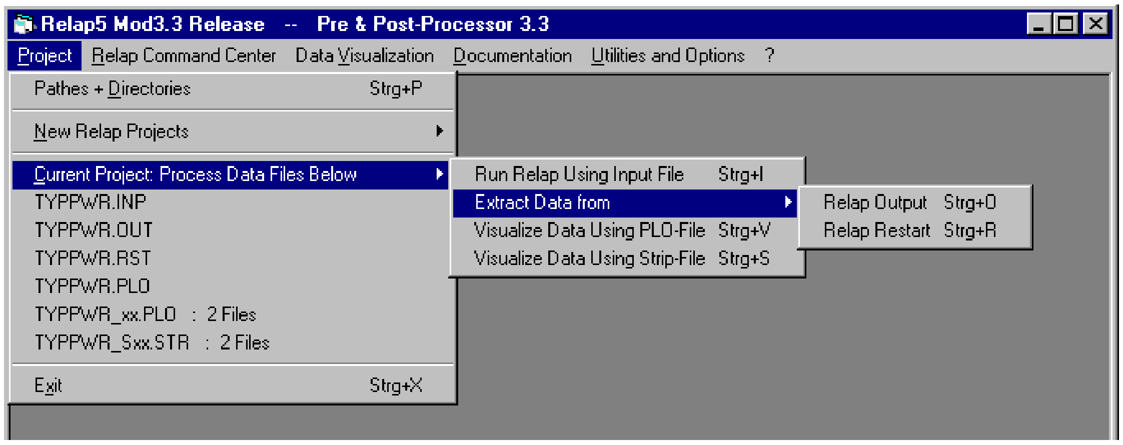


Figure 3.3-1: Project Menu: “Current Project”

The current project is the project which is shown in the bottom information bar. This menu option has conventional pop-up sub menus as shown in the Figure above and a short path to all of the actual available project files which can be post-processed. The screen shot above for example shows a project which contains an Input File, an Output File, a Restart File and a Plot Data File (*.PLO) of a previous application of the Output Postprocessor to the Output File. Additionally there are also two Plot Data files which are the result of a previous selection of plot data from the restart file and two strip data files. Clicking on the filenames will start the appropriate Pre- or Postprocessor. Clicking on “TYPPWR.INP” will launch the “Relap Command Center”, ready to work with this Input file. The same can be achieved by selecting the pop-up sub menu “Run Relap Using Input File” or by selecting the Main Menu Option “Relap Command Center”.

Clicking on “TYPPWR.OUT” or “TYPPWR.RST” will immediately launch the Output or the Restart Postprocessor. Again the same is attainable by selecting the pop-up sub menus at the right as shown in the Figure 3.3-1.

Clicking on “TYPPWR.PLO” will guide directly to the Plot Postprocessor where paper plots, or data tables can be made based on the plot data contained in the file. If there is only one file “TYPPWR_01.PLO” then clicking on this filename leads directly to the plot processor. If there are more PLO-files which have been generated previously by the Restart Postprocessor, the number of files appears, as the screenshot shows. When clicking on “TYPPWR_xx.PLO” a file selection window pops up, where the user selects the PLO-file he wants to work with. The same selection can be made for the strip files, if there is more than one. The file selection window in general appears, when using the sub menus “Visualize Data Using PLO- File” and “Visualize Data Using Strip-File”

New Relap Projects

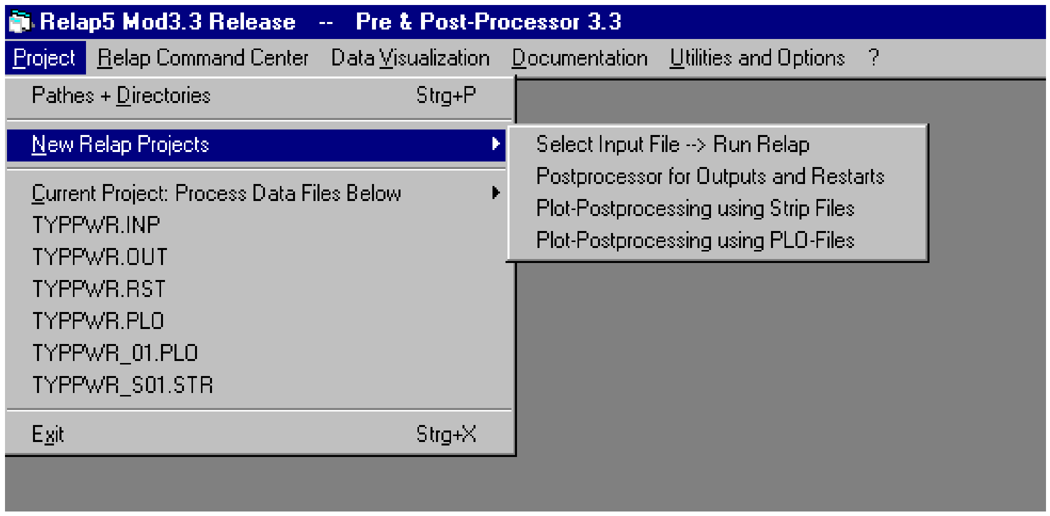

Relap5 Mod3.3 Release -- Pre & Post-Processor 3.3
Project
Relap Command Center
Data Visualization
Documentation
Utilities and Options
?
Pathes + Directories Strg+P
New Relap Projects
Current Project: Process Data Files Below
TYPPWR.INP
TYPPWR.OUT
TYPPWR.RST
TYPPWR.PLO
TYPPWR_01.PLO
TYPPWR_S01.STR
Exit Strg+X
Select Input File --> Run Relap
Postprocessor for Outputs and Restarts
Plot-Postprocessing using Strip Files
Plot-Postprocessing using PLO-Files

Figure 3.3-2: Project Menu: “Current Project”

The New Relap Projects option lets the user start a completely new Relap project either by selection of an existing Relap input file, or by compiling a new input file (Relap Command Center). Once a new project is selected and Relap is run, after quitting the Command Center the user will be asked if the new project shall be the new default (current) project. If the user affirms, the dynamic short links of the current project and the bottom status bar will be updated to reflect the new status. During shutdown process of the RelapPP application the user is asked again if the new project shall really be the actual one. All of the new project information is saved permanently if this is requested by the user.

The “New Relap Projects” menu item also includes all of the post-processing of Relap Results of other previous projects as shown in Figure 3.3-2. The Post-processing of existent Relap data can additionally be performed from the menu “Data Visualization” from the Menu bar at the top of the window.

Data Visualization

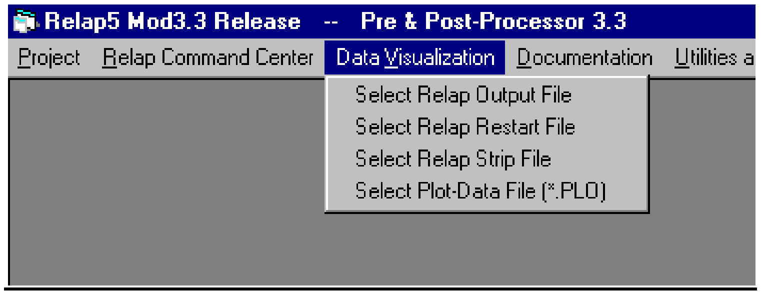


Figure 3.3-3: Menu: "Data Visualization"

The visualization of Relap data is described in Chapter 3.3.4 in detail. This menu allows the post processing of the current or actual Relap project or of past projects. It even allows the post-processing of outputs and strips of previous Relap versions. Only the Restart processor requires the actual Relap version 3.3 Release (aa-cp). The Restart processor however allows the reading of any binary UNIX restart as long as the Relap version is the one required (s. Chapter 3.3.7 for more detail).

Documentation

The documentation menu is described in Chapter 3.3.5.

Utilities & Options

All the Utilities, options and add-on programs intended to support the user are described in Chapter 3.3.6.

3.3.3 Relap Command Center

A screenshot of the Relap Command Center is shown in Figure A-3 (Annex A). The Command Center can be launched, as described above, from three different menus. It is a Child-Application Window which contains all the means and controls to successfully perform a complete Relap analysis. The window size is fixed and cannot be minimized or changed in size. The Command Center in general starts with the default or current project data, found in the Relap-INI configuration file. It is structured in such way that controls, command buttons, and text boxes for information and inputs are grouped with respect to certain sub tasks of a Relap analysis.

The structure of the window includes the following areas:

1. Menu Bar at the top

2. Header with the project name

3. File suffix selection area

4. Input/output, Strip and Restart file selection and information area

5. Results and Restart directory selection, Relap executable and steam table selection

6. Relap Problem Type (New, Restart, Strip) selection, input preparation and run area

7. Message window

8. Viewing of results and log files

9. Post-processing area

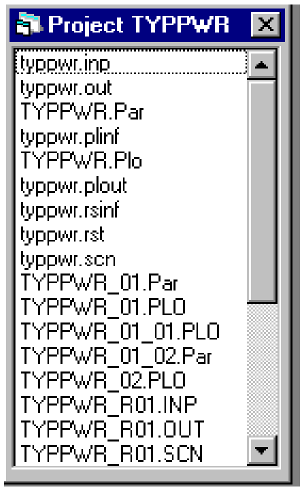
All of these 9 groups are described in detail.

<u>Menu bar</u>

The menu bar at the top of the window contains three menu items: The "Base-File List Box", the "Print Project Parameters" option and the "Exit" command. The exit form the Command Center can be achieved by three different commands: By the menu option "Exit", by the Exit Command Button at the lower right corner or the Windows X-button at the top right corner.

The "Print Project Parameters" menu option prints all the file names, directories and settings of the current Relap project using the default Windows Printer.

The "Base-File List Box" option pops up a Listbox which contains all the files belonging to the current Relap project. An example is shown in the next figure.



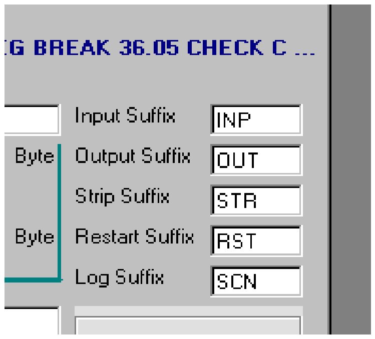
typpwr.inp
typpwr.out
TYPPWR.Par
typpwr.plinf
TYPPWR.Plo
typpwr.plout
typpwr.rsinf
typpwr.rst
typpwr.scn
TYPPWR_01.Par
TYPPWR_01.PLO
TYPPWR_01_01.PLO
TYPPWR_01_02.Par
TYPPWR_02.PLO
TYPPWR_R01.INP
TYPPWR_R01.OUT
TYPPWR_R01.SCN

Figure 3.3-4: Files of the current project

Project header

The project header is a line of information just below the menu bar which describes the Project. The Command Center reads the “Title Card” of the Relap input file during initialization and whenever a new input file (new project) has been selected and presents the title information in the project header line. If a selected input file cannot be found the line will show the information “..... Input not found!”

File suffix selection area

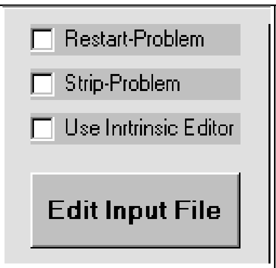
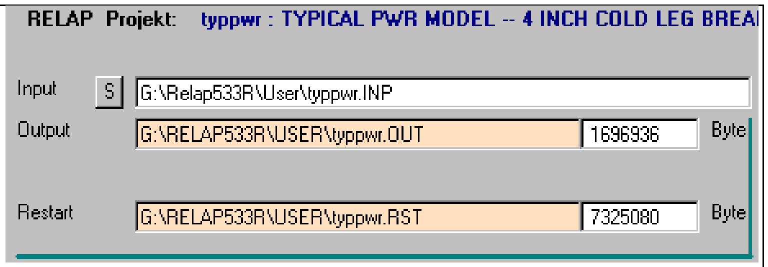

:G BREAK 36.05 CHECK C ...

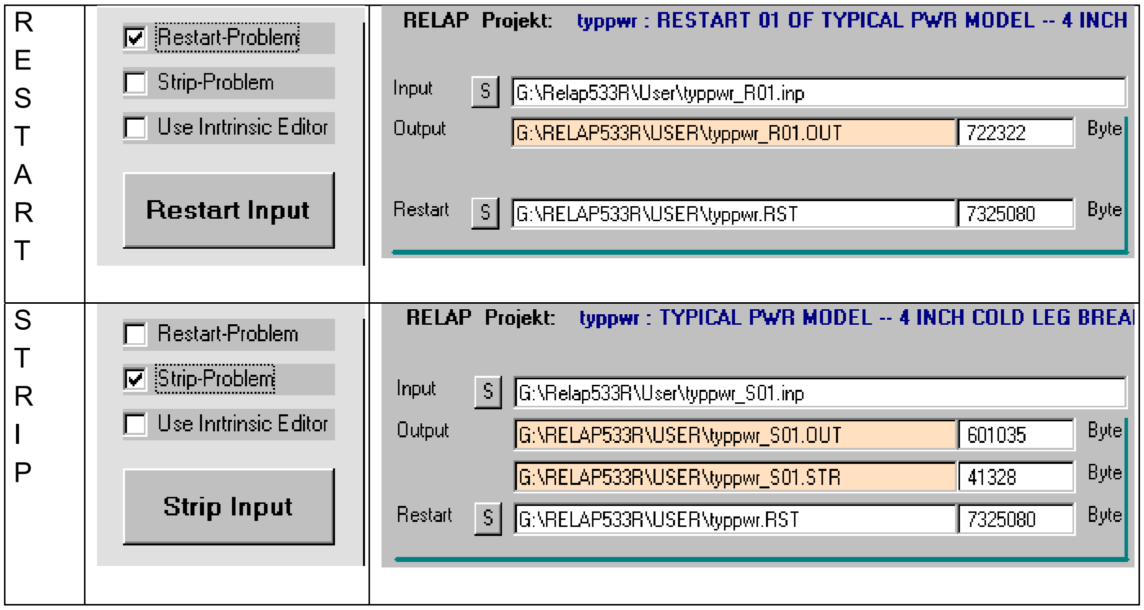
	Input Suffix	INP
Byte	Output Suffix	OUT
	Strip Suffix	STR
Byte	Restart Suffix	RST
	Log Suffix	SCN

Figure 3.3-5: File Suffix Selection Area

The figure above shows the default suffixes of the input files, the output files, the strip files, the restart and of the Log files. These defaults can be changed by placing the mouse cursor into the text boxes and typing the changes. The new suffixes will be saved on the Relap configuration file. The suffix of all the other files of a Relap project can not be changed. The result of changing a suffix, e.g. of the input file, can be immediately observed at the filename field to the left. If the current input file “...\TYPPWR.INP” exists, the input file name color is black. If the input suffix is changed by the user to “i”, the file name immediately changes also to “...\TYPPWR.i”. The color of the file name becomes red in case this file does not exist. The project header then changes to “TYPPWR-Input not found!”. The easiest way to change the input suffix is to make a copy of the input file with a new suffix and then select this input file. The file type of the file selection window has to be set to “All Files” in this case, because the default file type is the current input suffix. The file copy can be made either by the “Edit Input File” – command or by the “File Management” command.

Input/output, Strip and Restart file selection and information area

N E W	 ☐ Restart-Problem ☐ Strip-Problem ☐ Use Inrtinsic Editor Edit Input File	 RELAP Projekt: typpwr : TYPICAL PWR MODEL -- 4 INCH COLD LEG BREAI Input S G:\Relap533R\User\typpwr.INP Output G:\RELAP533R\USER\typpwr.OUT 1696936 Byte Restart G:\RELAP533R\USER\typpwr.RST 7325080 Byte



R E S T A R T

[✓] Restart-Problem
[ ] Strip-Problem
[ ] Use Inrtrinsic Editor
Restart Input

RELAP Projekt: typpwr : RESTART 01 OF TYPICAL PWR MODEL -- 4 INCH

Input	S	G:\Relap533R\User\typpwr_R01.inp		
Output		G:\RELAP533R\USER\typpwr_R01.OUT	722322	Byte
Restart	S	G:\RELAP533R\USER\typpwr.RST	7325080	Byte

S T R I P

[ ] Restart-Problem
[✓] Strip-Problem
[ ] Use Inrtrinsic Editor
Strip Input

RELAP Projekt: typpwr : TYPICAL PWR MODEL -- 4 INCH COLD LEG BREA

Input	S	G:\Relap533R\User\typpwr_S01.inp		
Output		G:\RELAP533R\USER\typpwr_S01.OUT	601035	Byte
		G:\RELAP533R\USER\typpwr_S01.STR	41328	Byte
Restart	S	G:\RELAP533R\USER\typpwr.RST	7325080	Byte

Figure 3.3-6: Problem and File Selection Area

The file selection area presents different information for each of the three Relap problem Types which can be handled by the Command Center. The Relap problem can be selected by a check box in the problem selection area as shown in the figure above. For each of the three problem types all the necessary “input” files are shown. The files which can be selected for the different problem types are selectable by the small “S” button. Pressing this button pops up the standard file selection option. These files can also be selected by typing into the text box fields directly. This however is not recommended. The “output” files are not selectable. The background color of these text boxes is orange. For all the output files and for the restart file a file length information is presented at the right side of the file name box. This file length information is dynamic and is updated during a Relap run in a multi tasking operating system environment (e.g. NT4).

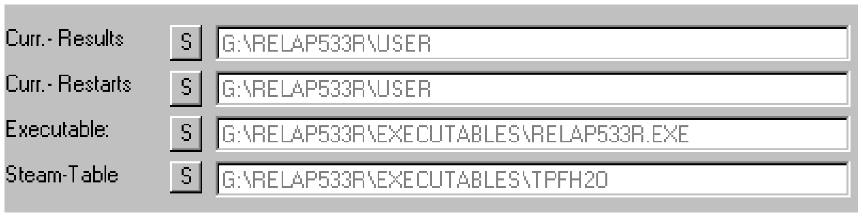
The examples above show the file information after Relap has been successfully run. For a new run (not strip or restart) Relap always requests that any existing output file and any existing restart file should be deleted first. If the user wants to save the current Output file or the current Restart file this can be done by the File Management command. Otherwise the “Start Relap” Command deletes these two files upon user request. This deletion is required only for NEW Relap problems. If there is no Output or no Restart for a new problem then the color of the file names is red and the file length is zero.

For big problems Restart files can become rather large. The RelapPP in general can handle file sizes as big as the size the operation system can handle. The file length function however has a limit at 2.147.483.647 Byte (type: long). If restart files exceed this file size this is indicated in the file size text box by “> 2 GByte”. If the length of a file name including the path exceeds 40 characters the color of the filename changes to blue. This is to indicate that Relap probably will not accept file names this long.

If the user positions the mouse cursor over one of the filename text box fields, a tool-tip-text box pops up encouraging the user to view or edit the file by simply clicking on the filename. Clicking on the restart filename will start a file selection box offering either the restart information file or the restart plot data information file to be viewed.

<u>Results and Restart directory selection, Relap executable and steam table selection</u>

The directories where the results of Relap runs (Output files, Plotdata files) or restarts are stored can be selected in this area of the RelapPP:



Curr.- Results [S] G:\RELAP533R\USER
Curr.- Restarts [S] G:\RELAP533R\USER
Executable: [S] G:\RELAP533R\EXECUTABLES\RELAP533R.EXE
Steam-Table [S] G:\RELAP533R\EXECUTABLES\TPFH2O

Figure 3.3-7: Directory selection

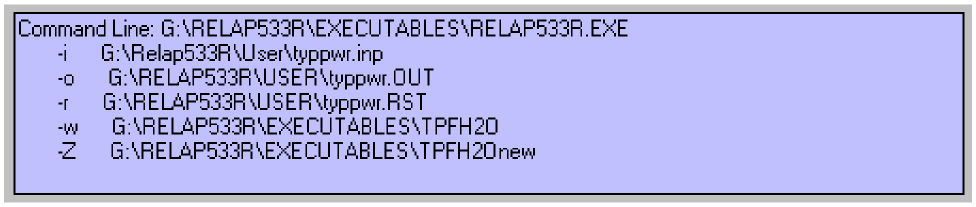
Additionally the Relap executable and the steam table can be selected. The possibility for the selection of different Relap executables is provided here because there are often new patches or minor versions releases of the Relap code which can be managed without renaming of the executable. This is favorable also if new minor Relap releases should be assessed against older versions. With the selection of the steam table TPFH2O also the "NEW" version of the steam table will be selected. In principle also older Relap version can be selected, however the postprocessors may not work with older versions of the code. A version management for all of the past Relap versions was under consideration for RelapPP, however is not completed now. The path name and file name textboxes as shown in Figure 3.3-7 can not be edited. This is indicated by the gray color of the filenames.

<u>Relap Problem Type (New, Restart, Strip) selection, input preparation and run area, Message Area</u>

As shown in Figure 3.3-6 the Relap problem will be selected by the problem check boxes. If nothing is checked the problem type is “NEW”. The command button for editing of the input file and the information on the file selection area vary, as described above for the problem type. There is a third checkbox “Use intrinsic Editor”. If this option is checked a built-in simple ASCII editor which has all the necessary functions will be used whenever the file size of a input file does not exceed 16 KB. Otherwise the “Big-Editor” (e.g. PAD97) will be used.

1. Run a new Relap case:

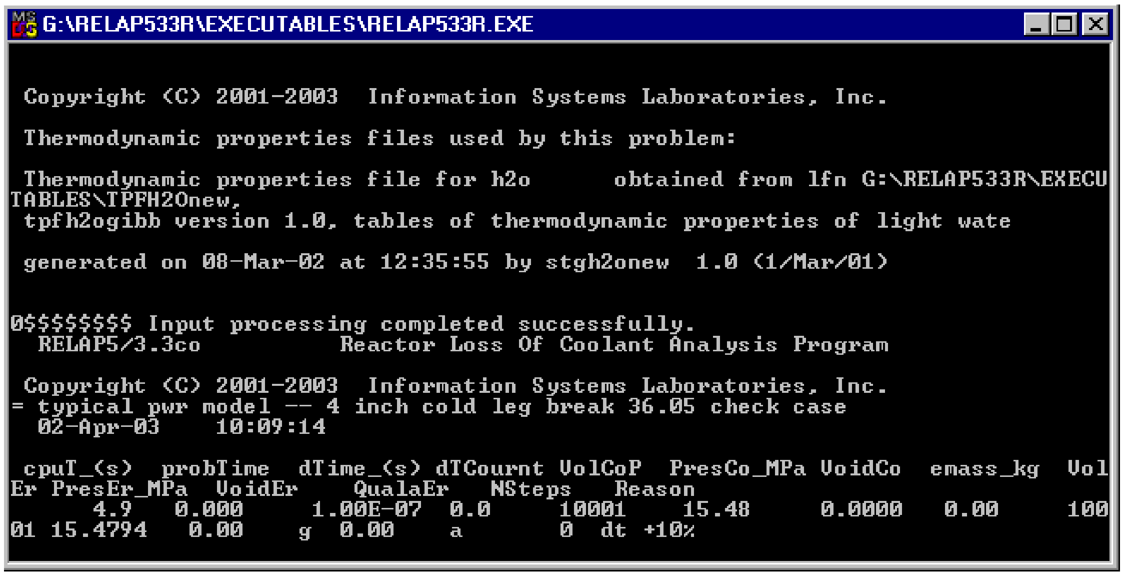
For running a new Relap case none of the problem checkboxes is checked. The Command “Edit Input File” allows modification of the input file of the current project. A new Relap project (Input file) can be selected by pressing the [S] button left of the Input file selection text box (s. Figure 3.3-6). If the user selects “NewFile.INP” he can start right away with the creation of a new Relap project input. A file with a red colored file name does not exist and cannot be edited. In parallel to the editing of the input file the user has access to the Relap Input Manual or all of the guidelines which can help him modifying or making the Relap input. All of this documentation is in PDF format and will be displayed by means of the installed Acrobat Reader. Multi instances; i.e. Input Manual and several guidelines in parallel are allowed. Once the Input is ready the user starts the Relap run by pressing the “Start Relap” button. The command line parameters of the Relap run will be displayed in the Message window and the user is asked to proceed with these parameters (Fig. 3.3-8):



```
Command Line: G:\RELAP533R\EXECUTABLES\RELAP533R.EXE
     -i    G:\Relap533R\User\typpwr.inp
     -o    G:\RELAP533R\USER\typpwr.OUT
     -r    G:\RELAP533R\USER\typpwr.RST
     -w    G:\RELAP533R\EXECUTABLES\TPFH2O
     -Z    G:\RELAP533R\EXECUTABLES\TPFH2Onew
```

Figure 3.3-8: Command Line Example

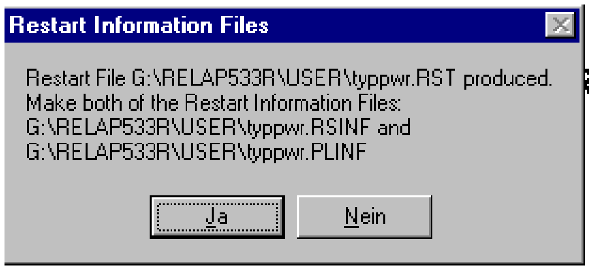
The example shown in the figure above is taken from a previous run where already exists the log file, the output file and a PLO file with data extracted from this output file and the restart file together with a PLOUT file converted Plot Data. The user will be asked to allow the program to erase these files. If this is not desired because the files have to be saved first the Relap run will not start. If the user allows the deletion of these files the Relap run starts. The Relap Screen output will be displayed in a Windows command window (DOS-Box).



```
G:\RELAP533R\EXECUTABLES\RELAP533R.EXE

 Copyright (C) 2001-2003  Information Systems Laboratories, Inc.

 Thermodynamic properties files used by this problem:

 Thermodynamic properties file for h2o      obtained from lfn G:\RELAP533R\EXECU
TABLES\TPFH2Onew,
 tpfh2ogibb version 1.0, tables of thermodynamic properties of light wate

 generated on 08-Mar-02 at 12:35:55 by stgh2onew  1.0 (1/Mar/01)


0$$$$$$$$ Input processing completed successfully.
  RELAP5/3.3co           Reactor Loss Of Coolant Analysis Program

 Copyright (C) 2001-2003  Information Systems Laboratories, Inc.
= typical pwr model -- 4 inch cold leg break 36.05 check case
  02-Apr-03    10:09:14

 cpuT_(s)  probTime  dTime_(s) dTCournt VolCoP  PresCo_MPa VoidCo   emass_kg  Vol
Er PresEr_MPa  VoidEr     QualaEr   NSteps    Reason
      4.9    0.000     1.00E-07  0.0     10001     15.48     0.0000   0.00      100
01 15.4794    0.00     g  0.00     a       0  dt +10%
```

Figure 3.3-9: Relap Run

All the information will be logged to the log file. After the successful completion of the Relap run, including the creation of a complete Restart File, the user will be asked to allow the program to generate the two Restart information files *.PLINF and *.RSINF (s. Table B-4).

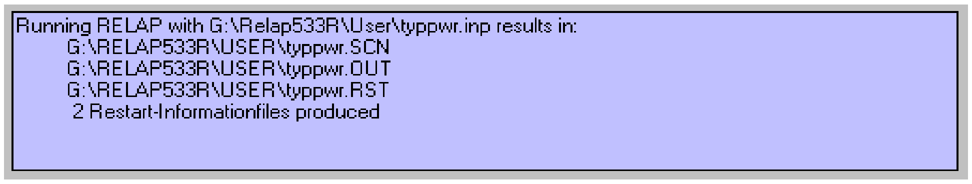


```
Restart Information Files

Restart File G:\RELAP533R\USER\typpwr.RST produced.
Make both of the Restart Information Files:
G:\RELAP533R\USER\typpwr.RSINF and
G:\RELAP533R\USER\typpwr.PLINF
```

Figure 3.3-10: Message about Restart Files

(Ja = Yes, Nein = No, in the English version of the program all standard messages are in English language)

These two files are generated by the external Restart-Postprocessor in the background. It is convenient to have these two files right after the end of a Relap run available in order to have a look on the data provided there. The data of these two files is needed by the restart input processor, the strip input processor, and by some plot data processors. Whenever the data is needed the existence of these files will be checked and if necessary the files will be generated once again. Once the two files are made the success message shown in the next figure will be displayed in the message box shown in figure 3.3-11:



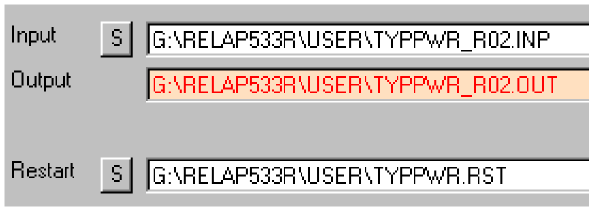
```
Running RELAP with G:\Relap533R\User\typpwr.inp results in:
       G:\RELAP533R\USER\typpwr.SCN
       G:\RELAP533R\USER\typpwr.OUT
       G:\RELAP533R\USER\typpwr.RST
        2 Restart-Informationfiles produced
```

Figure 3.3-11: Success Message at the end of a Relap run

The user should check the log file for the confirmation of a successful run.

2. Run a Relap Restart case:

A Relap Restart case requires the checking of the “Restart-Problem” checkbox. In the example shown in figure 3.3-6 there already exists a Restart Input file which is identified by “_R01”, which means that there is only one restart input available. The Command Center always checks for the latest restart number. For example: if there exist already two restart input files, the file selection information would look like this:

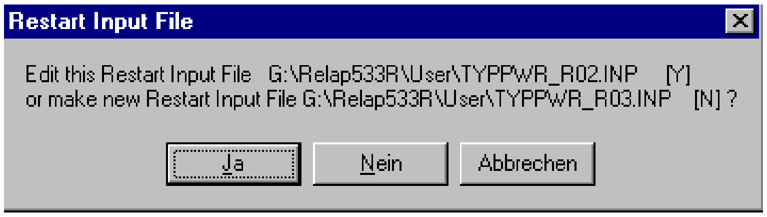


```
Input    S  G:\RELAP533R\USER\TYPPWR_R02.INP
Output      G:\RELAP533R\USER\TYPPWR_R02.OUT

Restart  S  G:\RELAP533R\USER\TYPPWR.RST
```

Figure 3.3-12: File Information

Always the latest version of existing restart input files (_R02) will be selected and is available by default. In the example of figure 3.3-12 the input file exist, but has not been used for a Relap run so far: there is no output file; the color of the filename of the output file is red and the file length is set to “0”. Other Restart input files can be selected as described previously by the file selection command [S].

If the latest version of the restart input file is selected, as shown in the example above (figure 3.3-12), the user is asked to edit this file or make a new one by the restart input processor (Figure 3.3-13). If the user selects a previous version of restart input file, he can edit this file directly without additional dialogue by pressing the “Restart Input” command button.


Restart Input File

Edit this Restart Input File   G:\Relap533R\User\TYPPWR_R02.INP   [Y]
or make new Restart Input File G:\Relap533R\User\TYPPWR_R03.INP   [N] ?

Figure 3.3-13: Restart option selection

(Abbrechen = exit or quit without actions, in the English version of the program all standard messages are in English language)

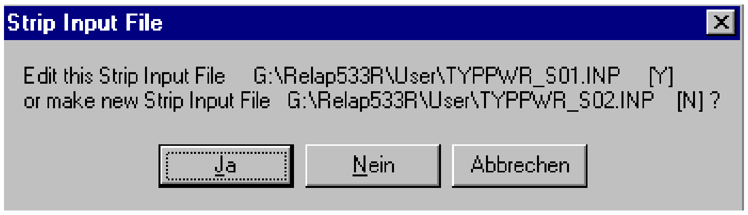
If the user chooses to make a new restart input file the Restart Input Processor generates a new input file with the data found in the *.RSINF file and the Relap Base (main-) input. If the *.RSINF file does not exist at that moment it will be made automatically (Table B-4). A typical automatically made restart input file is shown in table B-5. All the user has to modify is the restart number and the new end time on the time card. In order to have a consistent set of continuous plot data for the base case and the subsequent restarts the user should not modify any model data or add additional minor edit data. It is however not a restriction to change the model data. In these cases the user should be careful in interpreting the plot data.

The restart file which shall be used for the restart run should be selected before the new input file is generated in order to be consistent. The restart is started by pressing the “Start Relap” button. The start procedure will be the same as described above for the “NEW”-problem run.

3. Run a Relap Strip case:

A Relap Strip case requires the checking of the “Strip-Problem” checkbox. If this option is checked the Restart option is automatically unchecked (Radio Button behavior). In the example shown in figure 3.3-6 there already exists a Strip Input file which is identified by “_S01”, which means that there is only one Strip input and one output and strip file available for that case. Like for the restarts the Command Center always checks for the latest strip case number. If there are more strip cases which have been performed always the latest version of the strip input will be selected and is available by default. Other strip cases, available for the actual project file can be selected by the strip file selection command [S] (s. fig. 3.3-6).

After pressing the “Strip Input” command button the user is asked to edit this file or to ask the strip input generator to make a new strip input file:


Strip Input File

Edit this Strip Input File   G:\Relap533R\User\TYPPWR_S01.INP   [Y]
or make new Strip Input File   G:\Relap533R\User\TYPPWR_S02.INP   [N] ?

Figure 3.3-14: Strip option selection

If the user chooses to make a new strip input file based on the current restart file and the base case input, the Strip Input Processor generates a new input file (*_S02.INP) with the plot parameter data found in the *.PLINF file. If the *.PLINF file does not exist at that moment it will be generated (Table B-4). A typical automatically made strip input file is shown in table B-6. After generation of the new strip input an informative message is displayed in a message window. The strip input consists of a skeleton of a working strip input. It contains all the accessible parameters (in the TYPPWR example: 6481), arranged as comment cards. The maximum number of parameters to be stripped from a restart file is the Relap limit of 999. The user has to select the parameters he wants to be stripped and has to renumber them in the allowed range from 1001 to 1999. An effective procedure for doing this is the copying of all the selected commented parameters and paste them just after the 103 card and then renumber them. All of the other lines can be left in the file as commented lines.

The strip run is started by pressing the “Start Relap” button. The start procedure will be same as described above for the “NEW”-problem run.

At the end of the strip run the success message window will appear:

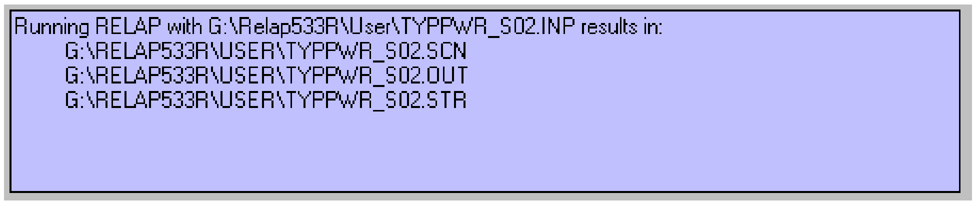

```
Running RELAP with G:\Relap533R\User\TYPPWR_S02.INP results in:
      G:\RELAP533R\USER\TYPPWR_S02.SCN
      G:\RELAP533R\USER\TYPPWR_S02.OUT
      G:\RELAP533R\USER\TYPPWR_S02.STR
```

Figure 3.3-15: Success Message at the end of a Relap Strip run

<u>Viewing of results and log files</u>

Results of Relap runs can be viewed selectively by several commands in the Viewing & Editing of Results area:

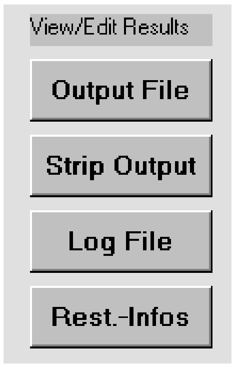


Figure 3.3-16: Results selection

Whenever one result file exists or is available the commands shown in the figure above are enabled. Otherwise they are disabled. If the problem type is “Restart” then the Strip Output command is disabled. For the Output, Log and Restart Informations a file selection window will pop up first with the default selection of the current active project file. Pressing the “Strip Output” Command will first cause the code to check if the current strip file is in the binary format or in ASCII format. A warning is displayed, if the strip file appears to be binary or is produced by a not supported Relap version. A possible problem can be revealed by clicking on the file name directly in the file selection area. In any case the Big-File-Editor is used for the viewing of the file contents.

<u>Post-processing area</u>

Relap results can be post processed by four different processors. They are accessible by the commands in the post-processing area:

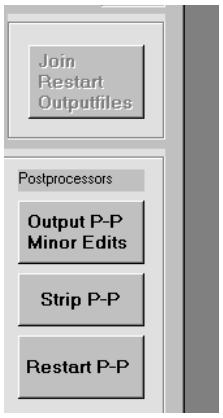


Figure 3.3-17: Relap Post Processors

1. The “Output P-P” extracts the Minor Edit Plot Data from the Main Relap outputs.

2. The “Join Restart Output Files” appends Outputs from Restarts to Outputs from the base case in order to have continuous Minor Edit data.

3. The “Restart P-P” post processes plot data from binary Restart files.

4. The “Strip P-P” post processes plot data from ASCII Strip files.

All of the plot data post processing is described in the next chapter 3.3.4. The output append function is described in the actual chapter. The post processors are available only if prerequisite files exist. This is shown in the next table.

Table 3.3-1: Prerequisites for the Post Processors

Problem	New	Restart	Strip
Output P-P	Base Case Output File	Restart Output Appended Output	-
Join Restart Output Files	Base Case Output File	Restart Output File	-
Restart P-P	Restart File	Restart File	-
Strip P-P	-	-	Strip File

If the files do not exist, the Postprocessor is not available within the Relap Command Center. In these cases the Command button is disabled as shown in Figure 3.3-17 for the “Join Restart Output Files” Command.

After pressing the **<u>“Output P-P”</u>** button the user is first asked if he wants to proceed, because the Extraction of the Minor Edit data can take some time for big outputs. If a Plot Data File (PLO-File) and a parameter file (PAR-File) already exist for that run, the Parameter file is checked for consistency and a message is displayed. If both files are consistent the user will be asked to keep these two files and continue or to replace the two files by new versions. If the user wants to keep the existing files the program control is transferred to the Data Visualization program after some data loading and organizations. The result of this operation is displayed in the message window.

If the user wants new versions of the plot data files the post processing is started. The progress of the data extraction process is displayed in the message window. When the data is extracted the following message appears:

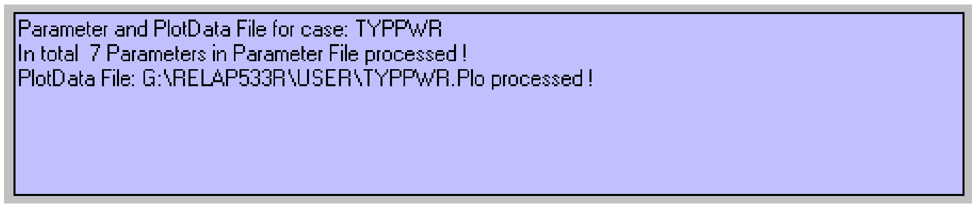
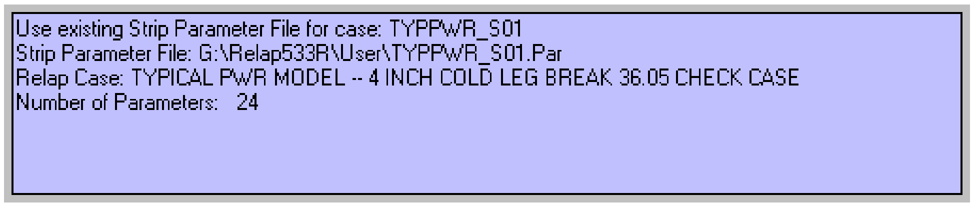

```
Parameter and PlotData File for case: TYPPWR
In total  7 Parameters in Parameter File processed !
PlotData File: G:\RELAP533R\USER\TYPPWR.Plo processed !
```

Figure 3.3-18: Success Message at the end of the Plot Data File generation

The program control is then transferred to the Data Visualization program, which is described in chapter 3.3.4.4.

The **"Strip P-P"** Command first will cause the code to check if the current strip file is in the binary format or in ASCII format. If the actual project has more strip files than one, first a file selection window is displayed with the current strip file as the default selection. A warning is displayed, if the selected strip file appears to be binary or is produced by a not supported Relap version. The Strip P-P is not available in these cases. If the strip file has been checked "OK" the user is asked to overwrite the existing strip parameter file (s. Table B-2), if there is one from a previous strip post-processing run, or to make a new one. If the existing file shall be used, this file will be checked first. The parameter file for strip files has a similar organization as the parameter file for Plot Data files (PLO) and contains the informations, which are not available in the strip files.

The following message is displayed when using an existing Strip parameter file:



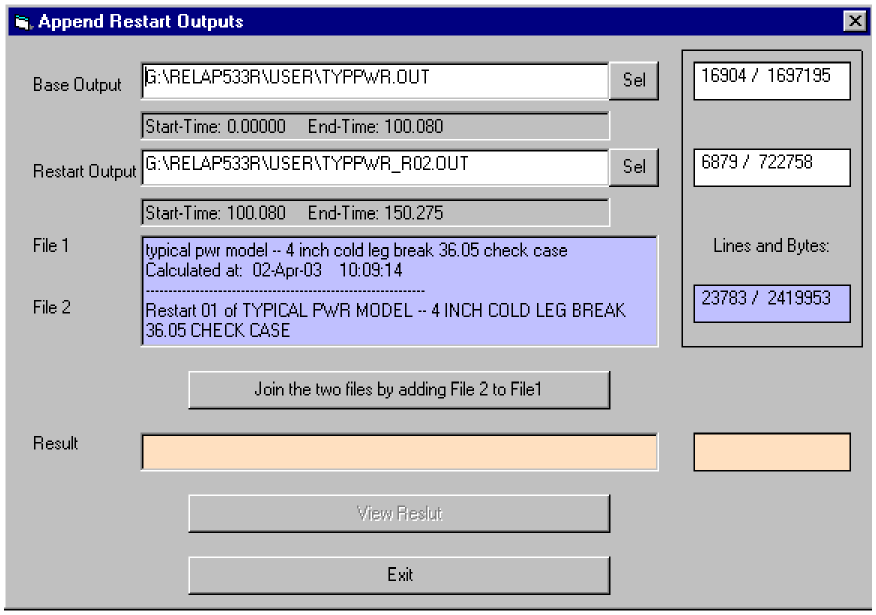
```
Use existing Strip Parameter File for case: TYPPWR_S01
Strip Parameter File: G:\Relap533R\User\TYPPWR_S01.Par
Relap Case: TYPICAL PWR MODEL -- 4 INCH COLD LEG BREAK 36.05 CHECK CASE
Number of Parameters:   24
```

Figure 3.3-19: Success Message using an existing Strip Parameter File

If the user wants the post processor to make a new Strip Parameter file or if no file exists, the parameter file will be generated automatically and the program control will be transferred to the "Data Visualization"

The **"Restart P-P"** Command uses the actual Restart file of the current project. There is only one restart file, therefore no file selection window will be displayed. The user is asked to proceed or quit the command. If the user wants to proceed, the "Extract and Process Relap5 Mod3.3 Release Restart Data" processor will be launched as a Child Application Window. This processor is described in chapter 3.3.4.2.

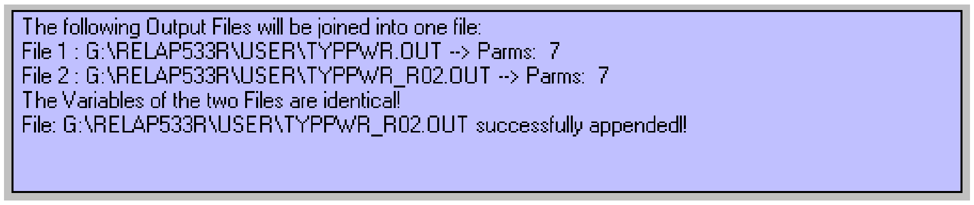
The **"Join Restart Output Files"** Command appends Outputs from Restarts at the end of Outputs from the base case in order to have continuous Minor Edit data. The alternative could be the gluing of successive PLO files. The command (Fig.3.3-17) is available only if a base case output and the main output of a restart run exist. Pressing the command pops up the following window (Fig. 3.3-20):



```
G:\RELAP533R\USER\TYPPWR.OUT
Start-Time: 0.00000    End-Time: 100.080
G:\RELAP533R\USER\TYPPWR_R02.OUT
Start-Time: 100.080    End-Time: 150.275
typical pwr model -- 4 inch cold leg break 36.05 check case
Calculated at:  02-Apr-03    10:09:14
-----------------------------------------------------------
Restart 01 of TYPICAL PWR MODEL -- 4 INCH COLD LEG BREAK
36.05 CHECK CASE
16904 /  1697195
6879 /  722758
23783 /  2419953
```

Figure 3.3-20: Append Restart Outputs Control Dialogue

The Base Case output and the main output of the current restart run is automatically selected. The processor reads the start time and end time of the two runs and the number of lines and bytes of each file. Additionally the processor reads the problem description lines of the two output files and presents this information in the blue message field, together with the number of lines and the total number of bytes in case both files are added together. At this time the user can select other than the preselected output files by pressing the Sel-(ect)-command buttons. If the user is confident that the two outputs should be merged together he initiates this process by pressing the “Join..” command. The processor then starts a more thorough checking, if the two files are compatible with respect to the number and the sequence of the Minor edit parameters. In case the Minor edits are identical, the user is requested to provide a filename for the new output file. The result of the parameter check is displayed in the information field of the Relap Command Center.



```
The following Output Files will be joined into one file:
File 1 : G:\RELAP533R\USER\TYPPWR.OUT --> Parms:  7
File 2 : G:\RELAP533R\USER\TYPPWR_R02.OUT --> Parms:  7
The Variables of the two Files are identical!
File: G:\RELAP533R\USER\TYPPWR_R02.OUT successfully appended!!
```

Figure 3.3-21: Success Message of Output File Check

If there are differences in the number of Minor edits and their sequence, the joining of the two files makes no sense and is not performed. The user is informed to locate the problem. If the two files are added together successfully, the filename of the new file, the number of lines and the file length appears in the “Results” fields. At this time the user can view the new output by pressing the “View Results” Command. The processor adds the following two bold lines of information to the top of the file:

```
1    RBIC/3.3co          RELAP5 Based Integrated Code
```

0Restart: Outputfile from Restart run added to this output

Final time=   150.275   sec   advancements attempted=   1019

```
1 Copyright (C) 2001-2003 Information Systems Laboratories, Inc.

0Execute file: G:\RELAP533R\EXECUTABLES\RELAP533R.EXE
```

3.3.4 Post Processing and Data Visualizing

Relap results are available from the Minor Edit data in the Relap Main Output, from the Plot Data in the Restart-Plot Data files and the Strip-Data as a result of strip runs. For the post processing of all of the data from these sources there are special programs (processors) integrated into the RelapPP, which extract these data from the files and arrange the data in a form that can be used by the plotting and visualization software. The four post processors are:

- Output Postprocessor
- Restart Postprocessor
- Strip Postprocessor
- Data Visualization and Plot Postprocessor

As described in preceding chapters the post processors are accessible from different menus of the RelapPP:

Table 3.3-2: Menu - Post Processor Matrix

Main Menu	Sub Menu	Output Proc.	Restart Proc.	Strip Proc	Vis. & Plot Proc
“Project”	“New Relap Projects” (Selection of new files)	Output / Restart automatic selection of appropriate Proc.		Strip	PLO-File
	“Current Project” (default file or selection)	Output def.	Restart def.	Strip sel.	PLO-File sel.
	“Process Data Files Below”	If Output exists	If Restart exists	If Strip exist	If PLOexist

Main Menu	Sub Menu	Output Proc.	Restart Proc.	Strip Proc	Vis. & Plot Proc
				+ sel.	+ sel.
“Relap Command Center”	Command Buttons	If Output exists	If Restart exists	If Strip exist + sel.	If PLOexist + sel.
“Data Visualization”	“Select Relap Output File”	Selected Output			
	“Select Relap Output File”		Selected Restart		
	“Select Relap Strip File”			Selected Strip	
	“Select Plot-Data File (*.PLO)				Selected PLO file

The Main Menu: “Project” – “Current Project” and the short paths are presented in Figure 3.3-1, the sub menu “New Relap Project” is presented in Figure 3.3-2 and the sub menus of the “Data Visualization” menu is seen in the Figure 3.3-3. The access of the post processors from the Relap Command Center has been described in chapter 3.3.3. The structure of the user interface (three child windows) and the data flow of the postprocessors are shown in Figure A-4. The Output- and the Restart-Post Processor have a common user interface, however with different meaning of the data fields and different captions. The parameter conversion processor which converts selected Plot Data in the Restart/PLOUT format to the PLO-format only can be accessed from the Restart Postprocessor function. The Data Visualization Processor has several data input channels (Data in PLO-format, PLOUT-format and STRIP-format). The looks of the user interface and the contents of several data fields vary with the input data format.

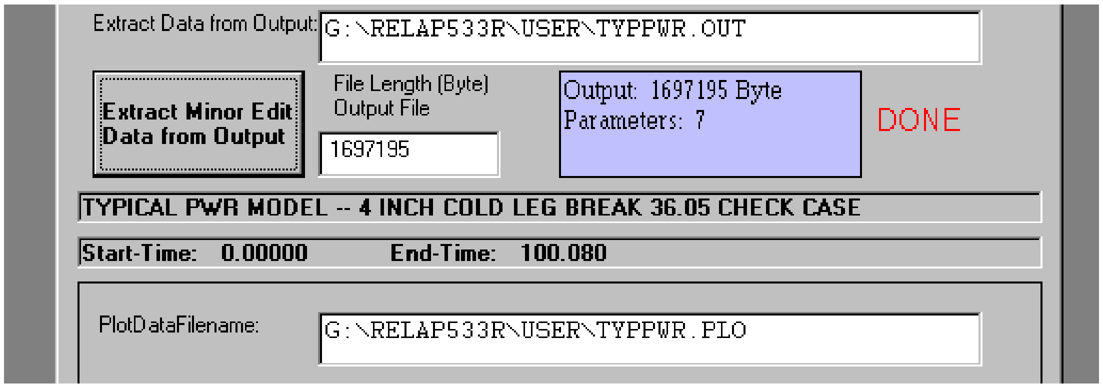
3.3.4.1 Output Post Processor

The function of this processor is:

- Extract the Minor Edit Data from the Main Output
- Write the Minor Edits Parameter Data in the PAR file
- Write all Plot Data in PLO format
- Guide the user to the Plot Postprocessor for the selection of parameters and of groups of parameters to be plotted, printed or visualized

The user interface (Child Application Window) of the Output Post Processor is shown in Figure A-5 (Annex A). This window will pop up when the Output Processor option for the current project or any output file is selected. If the Output Post Processor is selected from the “Relap Command Center” (Command Button: “Output P-P Minor Edits”, Fig. 3.3-17) the user interface does not pop up. However all internal functions of the postprocessor work, all files will be generated und the user will be led directly to the Plot Post Processor. If the Output Post Processor is selected from the main menu “Projects”, sub menu “New Relap Projects” – “Postprocessor for Outputs and Restarts” (s. Figure 3.3-2), the user has the choice to select any Relap Output file or any Restart file. In case of the option “New Relap Project” the file suffixes may be different from the current suffix for output or restart files. Therefore the selected file, is checked if it is a valid output file or a valid restart file. Based on the file type the appropriate post processor will be launched. Otherwise an error message will appear.

The screenshot (Fig. A-5) shows the situation where only the main output file of a new or of the current project TYPPWR exists. If a PLO file containing the minor edit data already exists for the selected output file, the Post Processor presents the informations shown in Figure A-6 (Annex A). The user has to start the extraction process of the Minor Edit data by pressing the command button “”Extract Minor Edit Data from Output”. In case a parameter file (PAR) already exists, a message will ask the user to replace this file. This is recommended. If the user wants to extract the minor edit data once more and if the data extraction process has been ended successful the following information will be shown:


Extract Data from Output: G:\RELAP533R\USER\TYPPWR.OUT

File Length (Byte) Output File: 1697195

Output: 1697195 Byte
Parameters: 7

DONE

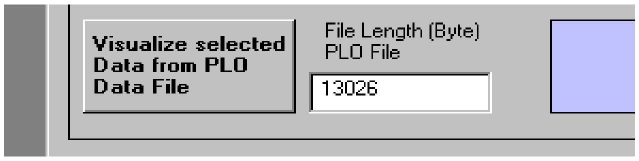
TYPICAL PWR MODEL -- 4 INCH COLD LEG BREAK 36.05 CHECK CASE

Start-Time: 0.00000 End-Time: 100.080

PlotDataFilename: G:\RELAP533R\USER\TYPPWR.PLO

Figure 3.3-22: Successful extraction of Minor Edit Data from a Relap Output file

The start time and the end time of the analyzed transient is read by the post processor from the PLO file. Double clicking on the Filename field launches the Big-Editor which automatically loads the Output file or the PLO file. The length of the PLO file is displayed at the bottom of the Window. The Command button for the Plot Post Processor is enabled if the PLO file has been created successfully (s. Figure 3.3-23). The user can directly proceed to plot data by pressing this command.



File Length (Byte) PLO File: 13026

Figure 3.3-23: Plot Post Processor Command button

3.3.4.2 Restart Post Processor

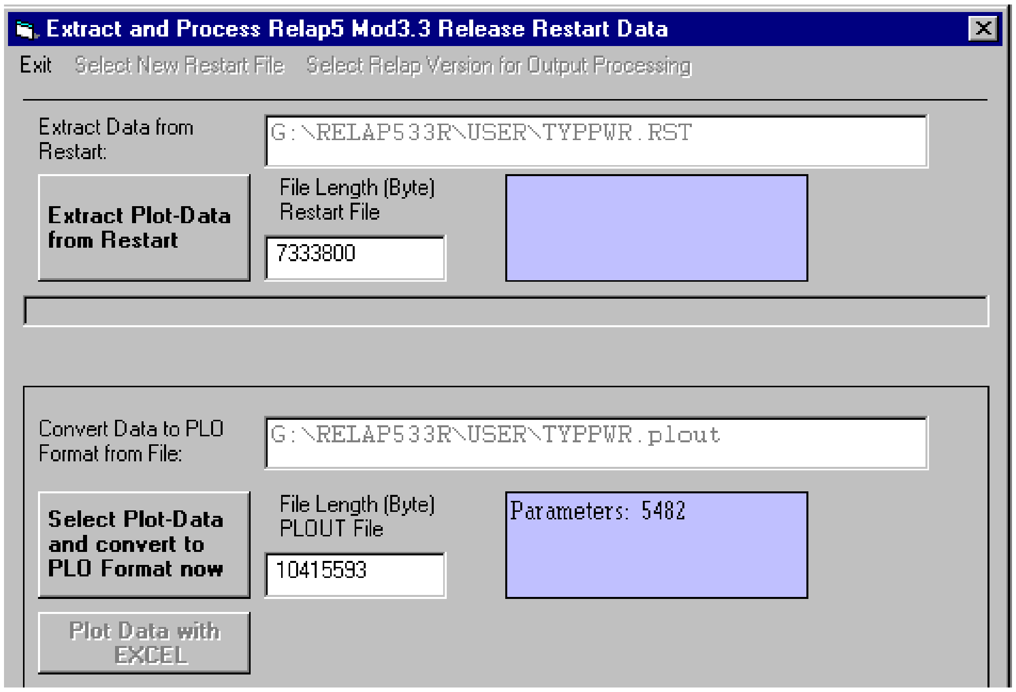
The function of this processor is:

- Extract all Plot Data from the binary Restart file and write the data in ASCII format (Pro gram RestR533R).
- Select and extract parameters from the PLOUT file and write the data in PLO format of up to 99 parameters per PLO file.
- Guide the user to the Plot Post Processor for the selection of parameters and of groups of parameters to be plotted, printed or visualized

The user interface (Child Application Window) of the Restart Post Processor is shown in Figure A-7 (Annex A). This window will pop up when the Restart Processor option for the current project or any other restart file is selected. If the Restart Post Processor is selected from the "Relap Command Center" (Command Button: "Restart P-P", Fig. 3.3-17) before starting the post processing application the user is first asked if the analysis is to proceed or not. If the Restart Post Processor is selected from the main menu "Projects", sub menu "New Relap Projects" – "Postprocessor for Outputs and Restarts" (s. Figure 3.3-2), the user has the choice to select any Relap Restart file. In case of choosing the option "New Relap Project" the file suffix of the restart file may be different to the restart suffix of the current project. Therefore the selected file is checked if it is a valid restart file, otherwise an error message will appear.

The screenshot in figure A-7 shows the situation where only the restart file of a new or the current project TYPPWR exists. The problem description and the start- and end time of the Relap calculation, is displayed only if a valid RSINF file exists for this case. The Restart processor reads this file and extracts this type of informations. Usually after each Relap run with restart option a Restart – Information file (RSINF-file) and a Plot Data – Information file (PLINF-file) will be generated. However the user will be asked to do so (s. Fig. 3.3-10).

If there already exists a PLOUT file containing the plot data from the restart file in the PLOUT ASCII format then some additional information is shown on the Application window. Figure 3.3-24 shows a screen shot of the case that a PLOUT-file exists, but no Restart-Information file. If the user wants to extract the plot data from the restart file, he has to press the command button "Extract Plot Data from Restart". If the user wants to extract the data again as shown in figure 3.3-24, after pressing the command, he will be informed that the PLOUT file already exists and will be asked if he wants to replace this file. Since the extraction process may last long for very big restart files, it is recommended that the user replaces the PLOUT file only if he is not sure if the PLOUT file is compatible with the restart file. Otherwise, if the data in the existent PLOUT file is corrupt (e.g. parts missing) the extraction process has to be started again. If there is no RSINF file or no PLINF file, both files will be created new. During the extraction process which will be performed in the background by launching the external program RESTR533R.EXE, a progress bar will appear at the bottom of the application window showing to the user that the process is still active.


Extract and Process Relap5 Mod3.3 Release Restart Data
Exit
Select New Restart File
Select Relap Version for Output Processing
Extract Plot-Data from Restart
Select Plot-Data and convert to PLO Format now
Plot Data with EXCEL

Extract Data from Restart: G:\RELAP533R\USER\TYPPWR.RST

File Length (Byte) Restart File: 7333800

Convert Data to PLO Format from File: G:\RELAP533R\USER\TYPPWR.plout

File Length (Byte) PLOUT File: 10415593

Parameters: 5482

Figure 3.3-24: Restart Post Processor with PLOUT file, w/o RSINF file

At the end of a successful extraction process the messages and information seen on Figure A-8 is displayed. The start time and the end time of the analyzed transient is read by the post processor from the RSINF file. The Relap case description is read from the PLINF file. Double clicking on the file name field of the restart file launches the Small-Editor which automatically loads the RSINF file. Double clicking on the file name field of the PLOUT file launches the Big-Editor which automatically loads the PLINF file. If both of the RSINF-file and the PLINF-file do not exist, as shown in the figure above (Fig. 3.3-24), the file names are printed in gray color and double clicking on the file names is not possible.

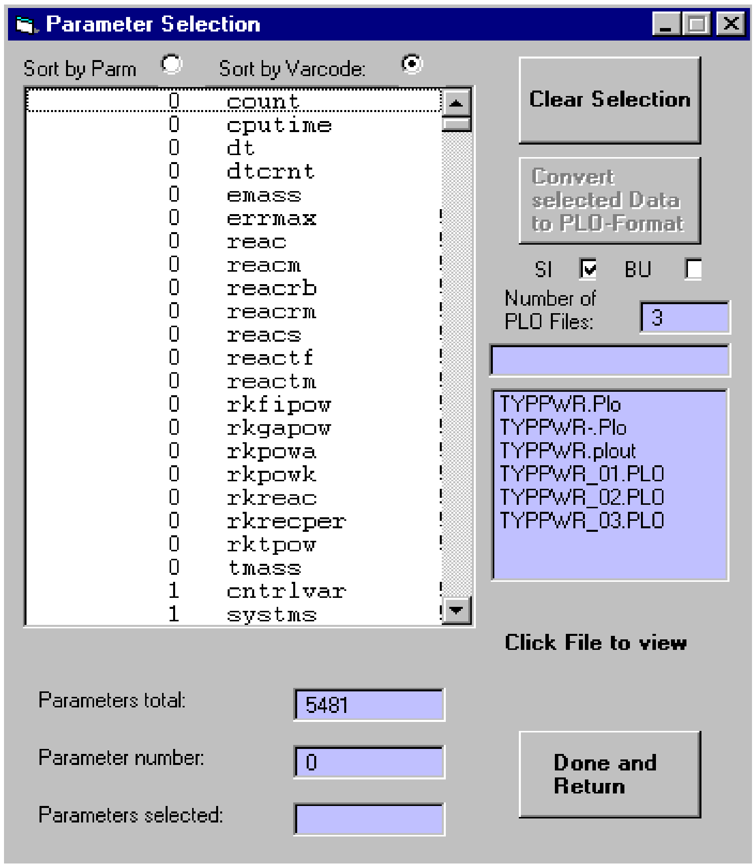
After a successful extraction run some additional statistical messages appear in the blue message field (s. Fig. A-8). If the Relap problem contains more that 99 plot data parameters (which is almost the case even for small problems) an additional Message pops up informing the user that only 99 parameters are allowed for each Plot data file in the PLO format and that more PLO files are necessary if more parameter have to be converted to PLO format. The structure of a PLO file is shown in Table B-1 (Annex B). The limit to 99 parameters comes from the limit of minor edits in the main output of a Relap run. Because of practical reasons it is not recommended to put such a large number of parameters into one PLO file. After a successful run the two command buttons for the further plotting process are enabled. The two alternative possibilities to proceed are shown in the Figure A-4 (Annex A):

- “Select Plot Data and convert to PLO Format now”
- “Visualize selected Data from PLOUT directly w/o PLO conversion”

The two choices are in principle equivalent with respect to the visualization of parameters, the printing of graphs and the printing of data tables. However there is some difference in the processing speed and the handling of data of very big restart files (many parameters and many times steps). There is some difference with respect to data documentation also. Because of the structure of the data in PLO files these files can be documented better than PLOUT data files. For very big restart files it is recommended not to proceed without the PLO conversion at first. The user has an alternative to speed up the plot process for big restart files by stripping the interesting data first and as a second step by using the Strip Data Post Processor.

The command "Visualize selected Data from PLOUT directly w/o PLO conversion" directly guides the user to the Plot Post Processor providing the PLOUT data and the plot parameter information found in the PLINF file. If there is no PLINF file (s. Fig. 3.3-24) an error message is displayed. The plot post processor is described below.

The command "Select Plot Data and convert to PLO Format now" starts the child application "Parameter Selection" (s. Figure 3.3-25). All the parameters found in the PLINF file are listed in a list box sorted by the "Variable code" (Varcode). Sorting by parameters can be selected by the radio button at the top. This sorting sometimes is more convenient for the extraction of groups of parameters, e.g. all of the Mass Flows.


Parameter Selection

Sort by Parm ○ [✓ Sort by Varcode: ●]

```
0  count
0  cputime
0  dt
0  dtcrnt
0  emass
0  errmax
0  reac
0  reacm
0  reacrb
0  reacrm
0  reacs
0  reactf
0  reactm
0  rkfipow
0  rkgapow
0  rkpowa
0  rkpowk
0  rkreac
0  rkrecper
0  rktpow
0  tmass
1  cntrlvar
1  systms
```

Clear Selection

Convert selected Data to PLO-Format

SI [✓] BU [ ]

Number of PLO Files: 3

TYPPWR.Plo
TYPPWR-.Plo
TYPPWR.plout
TYPPWR_01.PLO
TYPPWR_02.PLO
TYPPWR_03.PLO

Click File to view

Parameters total: 5481

Parameter number: 0

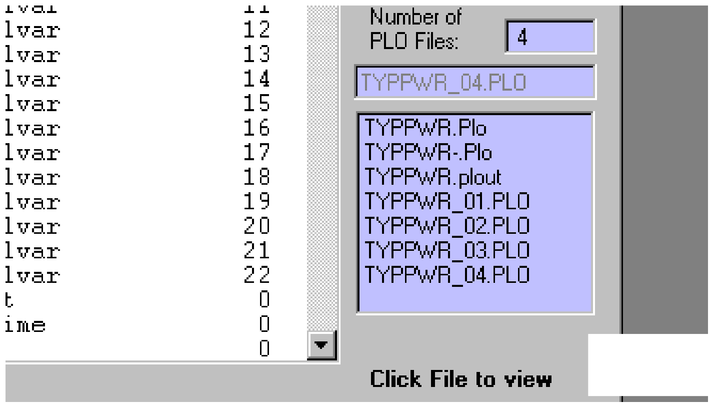
Parameters selected:

Done and Return

Figure 3.3-25: Parameter Selection for the conversion of PLOUT Plot Data

In the blue message fields at the bottom of the window the total number of parameters, the actual selected parameter and the number of selected parameters are presented. If more then 99 parameter are selected a warning is popped up. In the message fields at the right the total number of PLO files corresponding to the actual PLOUT file is given and the file names of all PLO files belonging to the current project.

Selection of parameters is possible by clicking with the left mouse button. If the first parameter is selected the “Convert Selected Data to PLO-Format” command is enabled. Deselection can be performed by left mouse click with CTRL-Key of the keyboard pressed or by the command “Clear Selection”. Multiple Selection of parameters is possible by either pressing the CTRL-Key and multiple left mouse clicks on single selected parameters or for groups of parameters (in the sort by parameter mode) by pressing the Shift-key and using the mouse or the navigation keys (Page-up, page-down, up, down) to maneuver through the list. Once the selection is made the command “Convert Selected Data to PLO-Format” starts the extraction process. The result will be written into a new PLO file with automatic selection of the next available PLO-file number. The new file name is shown in the message field. Clicking on file names of existing PLO files will launch an appropriate editor for the viewing of the file. Clicking on the PLOUT file should be avoided since PLOUT files tend to be very big. Since conversion can take some time, a progress bar is displayed on the bottom of the main window of the Restart Post processor. The result of a successful extraction is seen on figure 3.3-26.



```
lvar            12
lvar            13
lvar            14
lvar            15
lvar            16
lvar            17
lvar            18
lvar            19
lvar            20
lvar            21
lvar            22
t                0
ime              0
                 0
```

Number of PLO Files: 4

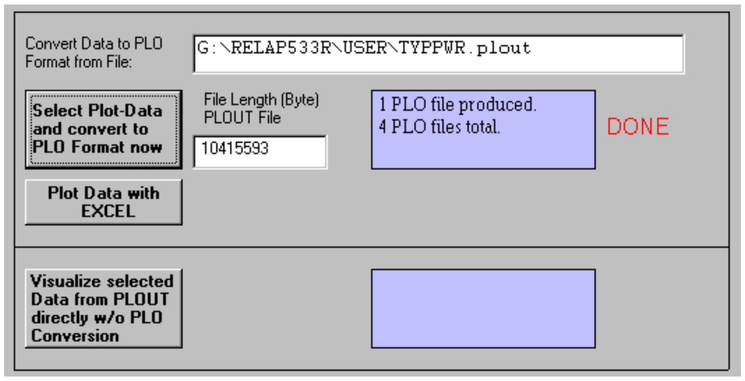
TYPPWR_04.PLO

TYPPWR.Plo
TYPPWR-.Plo
TYPPWR.plout
TYPPWR_01.PLO
TYPPWR_02.PLO
TYPPWR_03.PLO
TYPPWR_04.PLO

Click File to view

Figure 3.3-26: Plot Files Information

Up to 99 PLO files can be produced by this tool. The “Done and Return” command gets back to the Main Application. The result of the data extraction process is summarized as shown in the next figure.


Convert Data to PLO Format from File: G:\RELAP533R\USER\TYPPWR.plout

File Length (Byte) PLOUT File: 10415593

1 PLO file produced.
4 PLO files total.

DONE

Select Plot-Data and convert to PLO Format now
Plot Data with EXCEL
Visualize selected Data from PLOUT directly w/o PLO Conversion

Figure 3.3-27: Successful conversions of data from PLOUT format to PLO format

For the visualization of data in PLO files the command “Plot Data with EXCEL” is enabled now. If more than one PLO files are available for the current restart case a file selection dialog is displayed before starting the Plot Postprocessor.

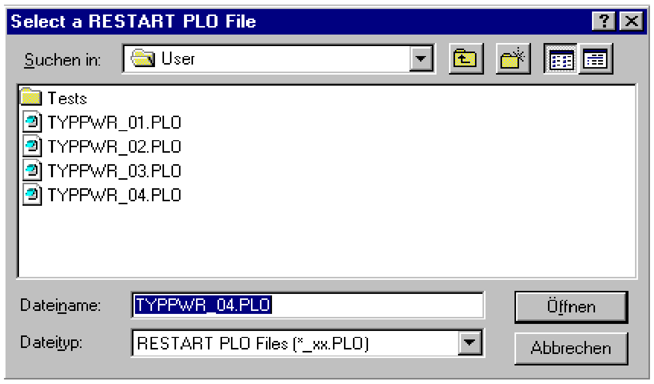


Figure 3.3-28: File Selection for multiple Restart PLO files

The number of Restart PLO files is updated also for the short path of the Main Menu “Current project”, as shown below. Clicking on the menu line “TYPPWR_xx.PLO: 4 Files” will pop up the same file selection window as shown in figure 3.3-28.

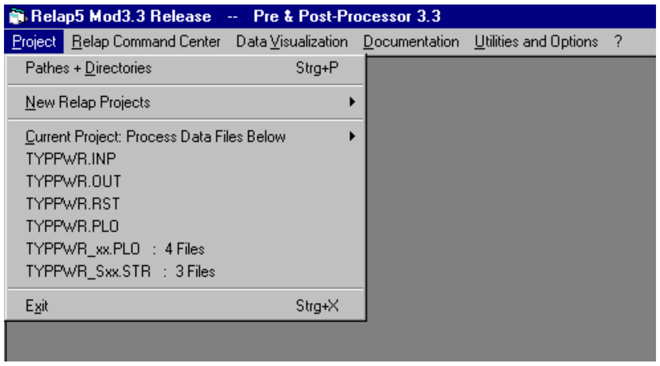


Figure 3.3-29: Current Project direct Pre/Post Processing access

3.3.4.3 Strip Post Processor

The Strip Post Processor can be started from five different menu options, as described in previous chapters (from the main menu bar: 3 sub menu options of “Project”, the “Relap Command Center” and from the “Data Visualization” menu).

The function of this processor is:

- Check the validity of the strip file selected.
- Extract the parameter data from the selected strip file and write the data in a PAR file.
- Provide a link to the Plot Postprocessor and provide the Plot Post Processor with the strip and the PAR file

If the current project has more strip files than one, a selection window (like Fig. 3.3-28) lets the user make a choice of the strip file he wants to work with. After the selection of a strip file the processor first checks if the file has all of the attributes of a valid strip file (Relap has the option of making also binary strip files). If the file is not a valid strip file the user is warned and the processor ends. If the strip file passes the checks the user is asked to overwrite any existing strip parameter file (s. Table B-2). If there is an existing one the user can chose to work with this file. It is recommended however to let the program make new PAR files. The PAR files contain plot data informations which are not found in the strip files (e.g. units of variables). If the PAR file has been made, the control is transferred to the Plot Post Processor which pops up with the “Data Visualization” window.

3.3.4.4 Plot Post Processor

As shown in figure A-4 (Annex A) the Plot Post Processor has four input data channels. The input data to the processor has the following data formats:

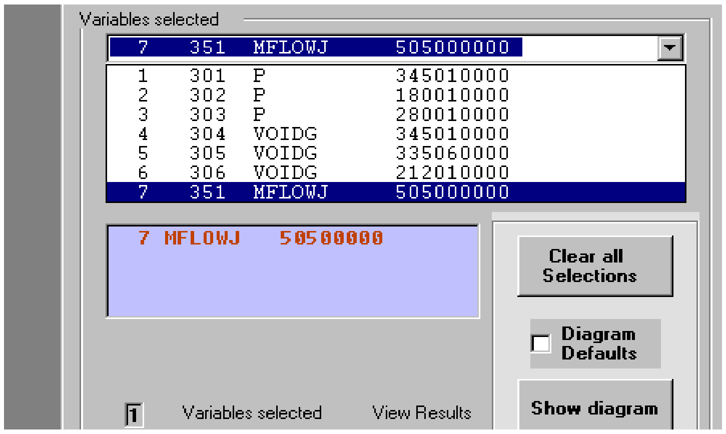
- PLO format (s. table B1), from Minor Edits or selections from PLOUT
- PLOUT format (s. table B-3), ASCII from restart plot data
- Strip format, Relap ASCII option
- PAR format (from PLO data and from strip data)

The function of the processor is:

- Display plot data information and selection of plot data files
- Selection of variables and groups of variables to be printed or viewed
- Print selected parameters in ASCII TXT format
- Print selected parameters in PLT format
- Print selected parameters in Xmgr5 format
- Make graphs of selected plot data with the integrated EXCEL engine
- Plot the graphs
- Launch the full-scope EXCEL for additional functions

The user interface of the application (“Data Visualization Window”), which is shown in figure A-9 (Annex A), contains all of the commands and tools for these tasks. The application window is divided into two functional parts. In the upper part there is a menu bar and data fields for the display of the Relap case description and the time interval of the analysis, data fields of the filenames containing the plot data and variable selection Combo Box. The screen shot in figure A-9 shows the processor for the post processing of plot data in the PLO format. New PLO files, other than the default one can be selected also either by the command in menu bar or by the small command button right of the file name field. If a new PLO file has been selected the processor automatically looks for the accompanying PAR file. If there is no PAR file, a new one will be generated automatically and loaded. The user even can select only a PAR file. The program then looks for the appropriate PLO file. The correlation will be checked in this case and the PAR data will be loaded. By clicking on the file name text fields the files can be viewed via an editor as described previously.

The selection of single variables or groups of variables for a new plot or graph is the same as described in the previous chapter. The selection box displays the informations also found in the PAR files. Once the first variable has been selected, the “Show Diagram” menu option in the menu bar at the top of the window and the command button at the right is enabled (s. figure 3.3-30).

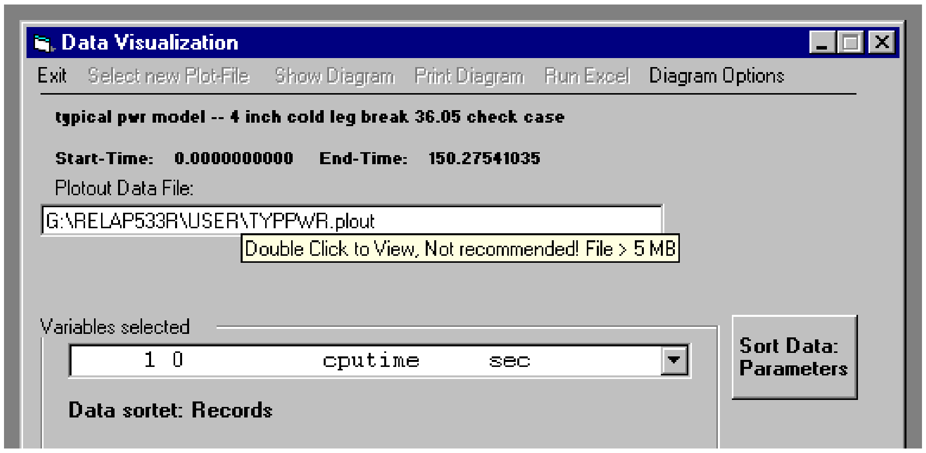


```
7    351   MFLOWJ       505000000

1    301   P            345010000
2    302   P            180010000
3    303   P            280010000
4    304   VOIDG        345010000
5    305   VOIDG        335060000
6    306   VOIDG        212010000
7    351   MFLOWJ       505000000

7 MFLOWJ    50500000
```

Figure 3.3-30: Selection of variables for the “PLO” Plot Post Processor

All the selected variables are listed in the blue message window shown in the figure above. The message window displays up to 4 parameters. For five and more a scroll bar appears at the right of the message window. The total number of selected variables is displayed in the small text field as shown in figure 3.3-30. The deselection of single variables or groups of variables has been described previously. The “Clear all Selections” command resets all selected variables.

In case the Plot Post Processor is called from the Restart Post Processor for the processing of PLOUT plot data the plot post processing is a child process of the Restart Post Processor. This is shown in Figure A-10 (Annex A). The upper part of the “Data Visualization” window is modified for the PLOUT processing as shown in Figure 3.3-31.



typical pwr model -- 4 inch cold leg break 36.05 check case

Start-Time: 0.0000000000 End-Time: 150.27541035

Plotout Data File: G:\RELAP533R\USER\TYPPWR.plout

Double Click to View, Not recommended! File > 5 MB

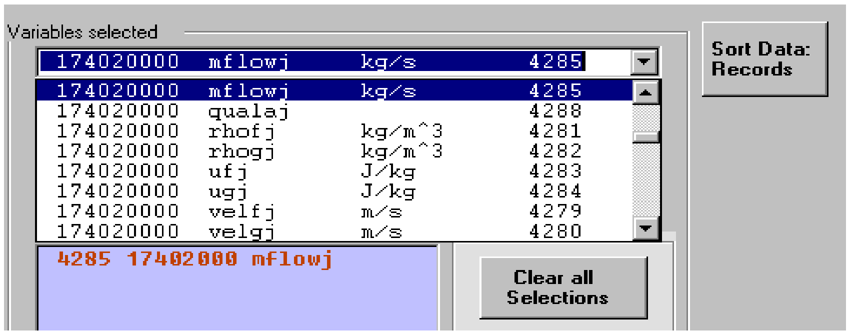
Variables selected: 1 0 cputime sec

Data sortet: Records

Figure 3.3-31: Processing of PLOUT plot data

The information about the Relap case description and the time intervals of the analysis are read from the associated PLINF and RSINF files. In the file name data field the active PLOUT file is displayed. However it cannot be reselected, like in the PLO case. The file selection command in the menu bar and the file selection button at the right of the file name field is disabled. If the mouse cursor is put over the file name field, the “ToolTipText” shown in the figure above is displayed, warning the user not to double click on this file for viewing. The “Big-file” editor launched by double clicking works on even very big files, however it may take time. If the user wants information about the PLOUT file he better double clicks on the PLOUT file name in the Restart Post processing window (s. Fig. A-10). For the PLOUT data processing there is no parameter file in PAR format. All data needed by the plot post processor will be read from the PLINF file and stored in internal arrays supplemented by additional data like the units of the variables. Consequently the PAR file name field is disabled here.

Restart/Plot files usually contain very much variables. All of the parameters of the current restart file are listed in the variables selection field (COMBO Box). The order is by default the order found in the Restart file (PLOUT, PLINF files). This default order is “Sorted by Records” as shown in figure 3.3-31. To make it more convenient to select associated data from the considerable amount of data, the user can change the sorting order of the data from “Record” to “Parameters” to “Variables” and back to Records by consecutively pressing the Sorting Command button. The sorting order “Variables” e.g. is shown in the next figure.



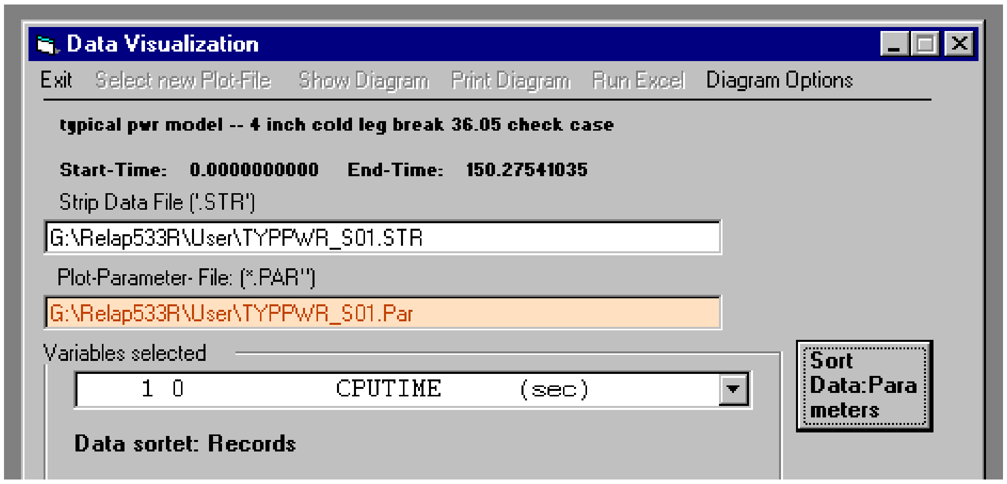
```
174020000  mflowj    kg/s      4285
174020000  mflowj    kg/s      4285
174020000  qualaj              4288
174020000  rhofj     kg/m^3    4281
174020000  rhogj     kg/m^3    4282
174020000  ufj       J/kg      4283
174020000  ugj       J/kg      4284
174020000  velfj     m/s       4279
174020000  velgj     m/s       4280
```

```
4285 17402000 mflowj
```

Figure 3.3-32: Sorting of PLOUT Plot Data

The caption of the sorting command button changes accordingly with the sorting order. In the figure above it is shown that the units of the parameters are listed also. This information is kept in a “Units” file, managed by the “Utilities and Options” Menu of the Main Menu. Selection of single Parameters and groups of parameters, and deselection is the same as for the PLO data case.

In case the Plot Post Processor is called in order to process Strip plot data the upper part of the “Data Visualization” window looks as shown in Figure 3.3-33. The information about the Relap case description and the time intervals of the analysis are read from the Strip file and the associated RSINF file. In the file name data field the Strip file selected is displayed. Like in the case for PLOUT files it cannot be reselected. The file selection command in the menu bar and the file selection button right of the file name field is disabled. Double clicking on the file name field launches the big-file editor for the viewing of this file. For each Strip file there is also a PAR file containing additional data. The file name is displayed in the orange file name field below the Strip file name field. Double clicking on the file name starts the editor loading the PAR file.



typical pwr model -- 4 inch cold leg break 36.05 check case

Start-Time: 0.0000000000 End-Time: 150.27541035

Strip Data File ('.STR')

G:\Relap533R\User\TYPPWR_S01.STR

Plot-Parameter- File: (*.PAR")

G:\Relap533R\User\TYPPWR_S01.Par

Variables selected

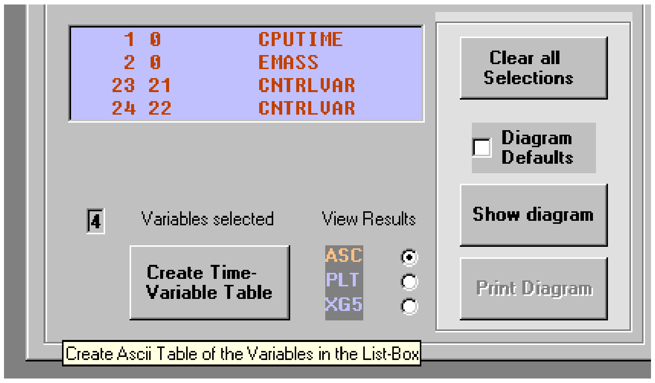
```
1 0          CPUTIME       (sec)
```

Data sortet: Records

Figure 3.3-33: Processing of Strip plot data

Strip files can contain up to 999 variables. All the variables of the current strip file are listed in the variables selection field (COMBO Box). The order is by default the order found in the strip file (PAR file). This default order is “Sorted by Records” as shown in the figure above. The sorting order can be changed as described for the PLOUT case. The selection of single parameters and groups of parameters, and the deselection has been described previously.

The selection of parameters for visualization or for plotting graphs should be made with respect to consistent units for the Y-axis of the graph. If the user does not want to plot the data in graphs and rather wants to store the data in separate files or print the data on paper he can select groups of any parameters, e.g. all of the parameters related to one hydraulic component.



```
 1  0         CPUTIME
 2  0         EMASS
23 21         CNTRLVAR
24 22         CNTRLVAR
```

4 Variables selected

View Results: [✓ ASC] PLT XG5

Clear all Selections
Diagram Defaults
Show diagram
Print Diagram
Create Time-Variable Table
Create Ascii Table of the Variables in the List-Box

Figure 3.3-34: Processing of Strip plot data (Ascii tables)

As shown in the figure above there are three different data formats available for storing or printing the data in ASCII format:

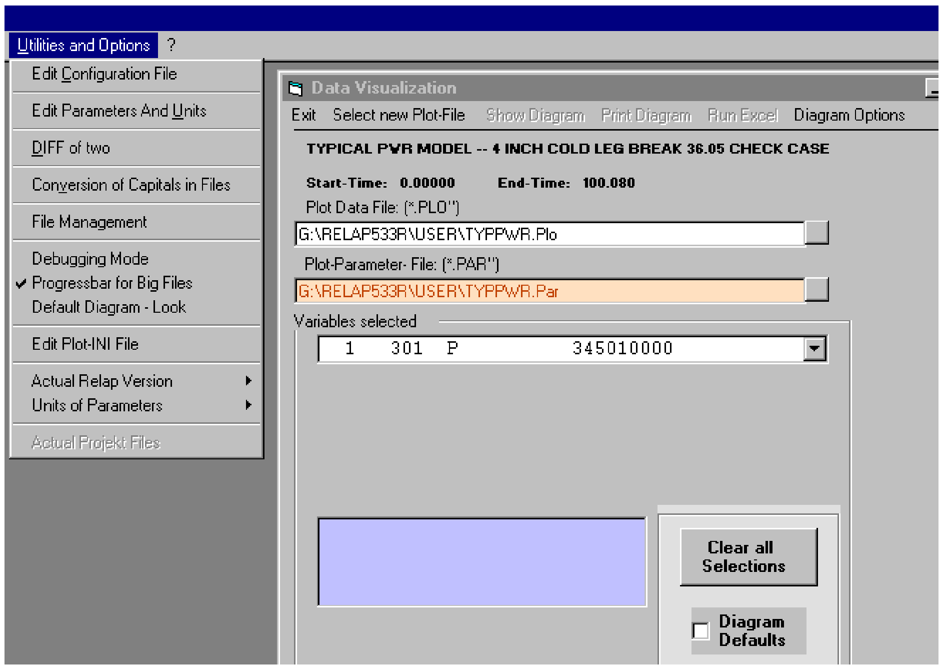
- Pure ASCII format
- PLT Plot Format
- Xmgr5 Format

All of these different data formats are shown in the Table B-7. In the ASCII Format the plot data is arranged in columns. This data format is understood by most of the external plotting software. In the PLT format the data for each parameter is arranged in blocks of 6 columns. This PLT format has been used in the past by special plotting programs in connection with main frame applications. The XG5 format can be imported directly by the Xmgr5 software.

The different formats can be selected by three radio buttons shown in the figure 3.3-34. The format selected changes the color to orange. Once the user clicked on the command button “Create Time Variable Table” he is asked to provide a file name. The file suffix is automatically preset, as shown in Table B-7 (Annex B). Once the file has been created the user can view the result by clicking on the orange Format name in the field left of the radio buttons. Only the actual created file in the orange colored Format is available.

If at least one variable has been selected, the command button and the top menu option “Show Diagram” is enabled. The style of a graph on the screen or a paperplot can be determined by the user within the limits build into the RelapPP and within the limits of the embedded EXCEL engine. The general configuration of RelapPP is described in chapter 3.3.6. During the startup of RelapPP the program reads all the INI files. In the PLOT33R.INI file there is some information about the style of the graph. The “Default Look” of the graph is characterized by all the logical variables in the PLOT33R.INI file set to TRUE. In this case the checkbox “Diagram Defaults”, seen in figure 3.3-34 is checked.

Figure A-11 shows an example of the PLOT33R.INI file where one of the logical variables, the variable L_Colored_Lines, has been set to FALSE. If this option is true the lines on the graph are colored, which enhances the differences of individual multiple lines. If the logical variable L_Colored_Lines is set to FALSE, B&W printers print the lines in different styles (e.g. dashed, solid). Figure A-11 also shows the meaning of the different variables in the INI file. For the visualization on the screen or by printing with a color printer all the logical variables may be set to TRUE. Alternatively either the option “Default Diagram Look” on the Main Menu option “Utilities & Options” or the checkbox on the Data Visualization Application Window can be checked. The correlation of these two possibilities is shown in the following screenshot (Fig. 3.3-35). If the “Diagrams Default” Option or checkbox is checked, the “Edit Plot INI File” Command on the Main Menu and the “Diagram Options” Command on the “Data Visualization” Window is not available. The figure below shows the situation where these options are available to the user.



TYPICAL PWR MODEL -- 4 INCH COLD LEG BREAK 36.05 CHECK CASE

Start-Time: 0.00000 End-Time: 100.080

Plot Data File: (*.PLO")

G:\RELAP533R\USER\TYPPWR.Plo

Plot-Parameter- File: (*.PAR")

G:\RELAP533R\USER\TYPPWR.Par

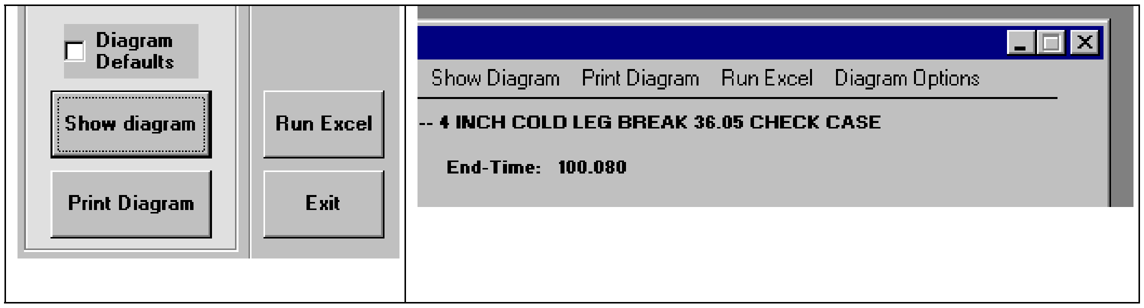
Variables selected

```
 1    301   P          345010000
```

Figure 3.3-35: Plotting w/o Default Style

Once variables have been selected the “Show Diagram” commands become available. These commands prepare the plot data for the built in link to the EXCEL engine, transfer all of the required data to the EXCEL spread sheet and initiate the graphing parameters and display the graph. A typical graph is shown in Figure A-11 (Annex A). EXCEL spread sheets have a limit of 32767 lines. In the plotting postprocessor any line (time step number) beyond this limit is ignored (the ASCII tables are not limited). The number of Columns (Plot Parameters + Time Column) is limited to less than 100. With respect to the dynamically allocated data arrays the number of columns allowed minus 1 times the number of rows should be less than 524288 (2^{19}). The program checks these limits and automatically corrects faulty figures in the PLOT33R.INI file. For max.-lines-graphs the maximum value of parameters allowed is 16.

Once a graph has been generated the commands "Print Diagram" are enabled (Fig. 3.3-36). The graph will be plotted by the default printer. The printer can be changed by the Windows functions to change or add printers.



-- 4 INCH COLD LEG BREAK 36.05 CHECK CASE

End-Time: 100.080

Figure 3.3-36: Print Diagram & EXCEL Full Scope Commands

By the commands "Run Excel", which are available now, the user can start the full EXCEL. All the data including the graph are transferred to the full EXCEL application organized in Maps. Equivalent to these commands the user can right-click on the graph (s. figure 3.3-37) and chose the "open" command.

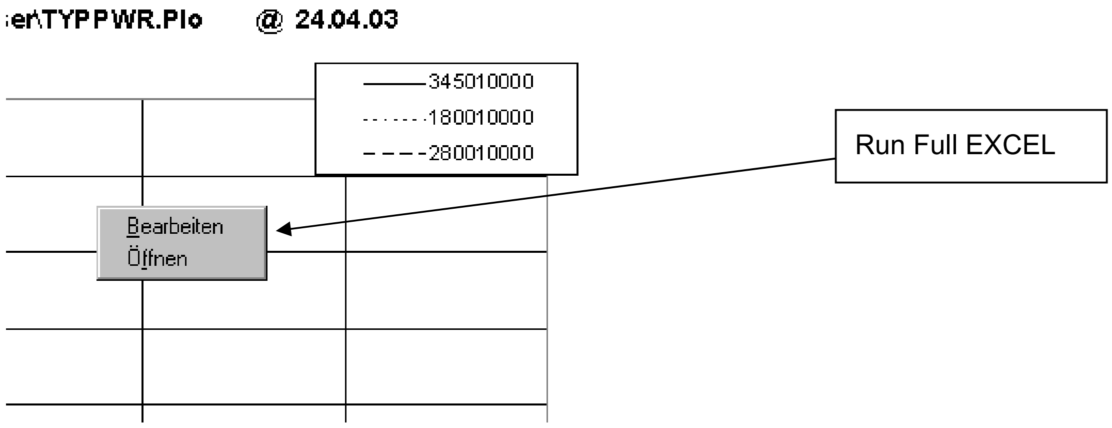


Figure 3.3-37: "Öffnen" = Open EXCEL

(In the US Version of RelapPP all the messages and captions are in English language)

Within the Full Scope EXCEL all defaults for plots, like color of the graphing area, style of the headers, axes and legends and others can be personalized. By choosing the upper command “Bearbeiten” (= Edit, s. above figure) all the desired changes can be made even easier. The Excel Plot Window changes into the “Edit” mode which is characterized by the tabs for the selection of the spread sheets of a map (s. figure 3.3-38). The diagram spreadsheet always contains the graph, the first table spreadsheet contains the plot data. All the other spread sheets (Default = 2) are empty and can be used for special purposes. The number of additional spread sheets can be setup within EXCEL. Two times “ESC” on the keyboard leaves the Edit Mode.

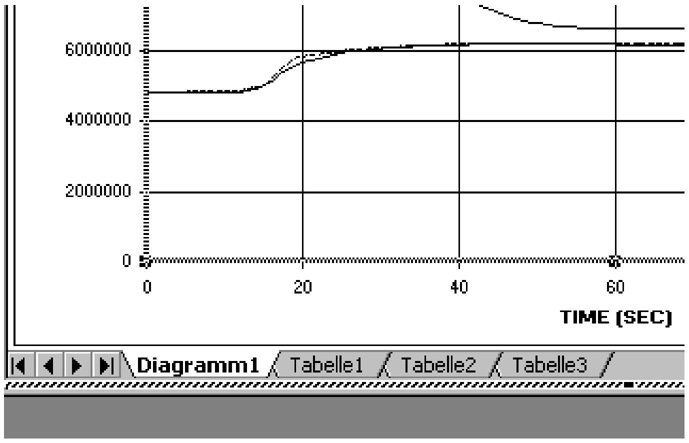


Figure 3.3-38: Excel Graph in Edit Mode

Just left-clicking on any graph element lets the user change the style of this element within the limits EXCEL provides. Clicking the right mouse button on any space between graph elements pops up the menu shown in the next figure.

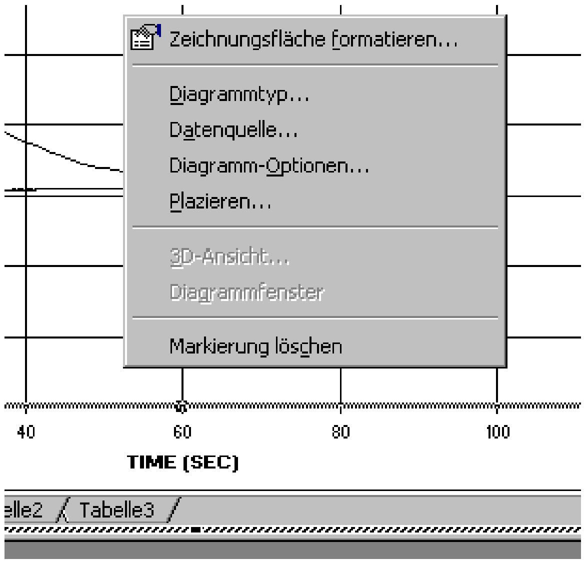


Figure 3.3-39: Graph Edit Menu

This menu allows the formatting of the canvas of the plot and provides several sub menus to edit the diagram type, to edit the data source (Data in table 1) and several other diagram options. The user specific default or standard diagram type can be set up by the “Diagram Type...” sub menu. More details for the formatting of graphs with EXCEL are available from the MicroSoft Press or any other literature for EXCEL (/5/ or later versions).

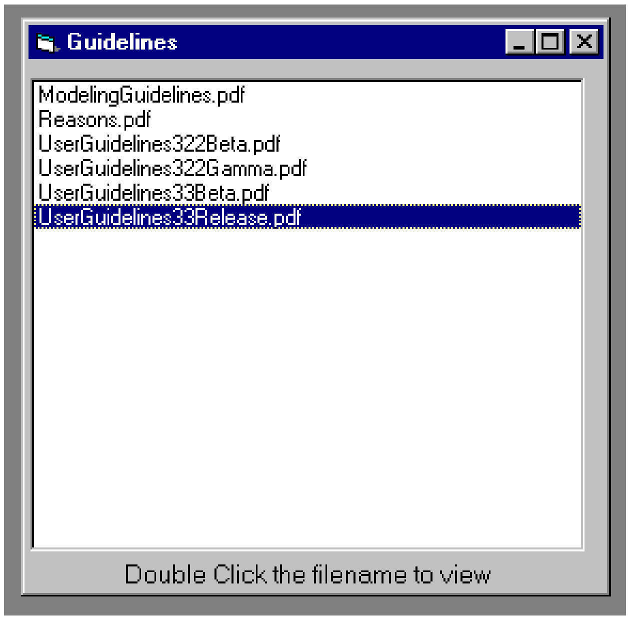
3.3.5 Documentation

The Relap documentation can be fully accessed by the main menu “Documentation” or partly by the Relap Command Center.

<u>Input Descriprion and User Guidelines</u>

The Relap Command Center (Figure A-3) provides a “Documentation” section where the user has access to the Relap Input Manual or to the “User Guidelines” released for the current Relap version. The Input Manual is installed by default into the directory “RelapRootDirectory\Manuals” (s. installation chapter). The filename of the input manual is set by the initialization of the application (s. next section). Pressing the command button launches the Acrobat Reader which is installed in the system which automatically loads the Input Manual. The manual can be viewed in parallel to the input development.

The “Guidelines” command provides the choice of viewing all the guidelines applicable to the Relap version installed. Pressing the “Guidelines” command button starts a selection window, shown in the next figure.



ModelingGuidelines.pdf
Reasons.pdf
UserGuidelines322Beta.pdf
UserGuidelines322Gamma.pdf
UserGuidelines33Beta.pdf
UserGuidelines33Release.pdf

Double Click the filename to view

Figure 3.3-40: Guidelines

Any of the guidelines in the list can be viewed by the Acrobat Reader just by double clicking on the file name. The list dynamically displays all of the *.PDF files in the Guidelines directory

(RelapRootDirectory\Guidelines, s. installation chapter). The major guidelines distributed by the USNRC are installed by default. Any new guideline in PDF format can be added by just copying the file into in this directory.

<u>Relap Documentation and Quicklook RelapPP</u>

All the Relap documentation is accessible from the Top Level Menu “Documentation” (s. Figure A-2, Annex A). A screenshot of this menu is shown in the next figure.

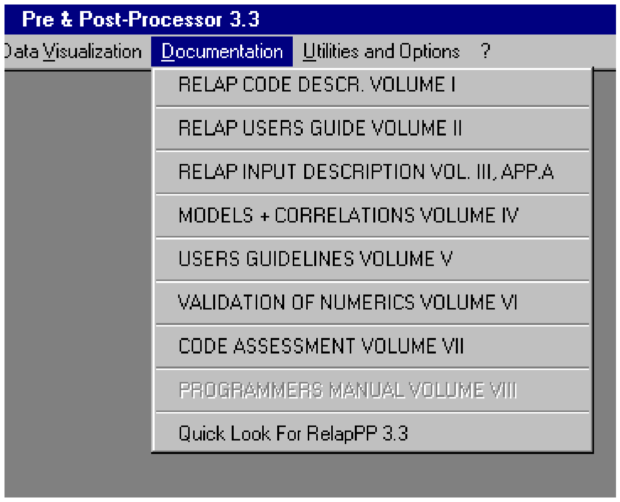

Pre & Post-Processor 3.3
Data Visualization
Documentation
Utilities and Options
?
RELAP CODE DESCR. VOLUME I
RELAP USERS GUIDE VOLUME II
RELAP INPUT DESCRIPTION VOL. III, APP.A
MODELS + CORRELATIONS VOLUME IV
USERS GUIDELINES VOLUME V
VALIDATION OF NUMERICS VOLUME VI
CODE ASSESSMENT VOLUME VII
PROGRAMMERS MANUAL VOLUME VIII
Quick Look For RelapPP 3.3

Figure 3.3-41: Relap Documentation

Clicking on any of the menu items starts the ACROBAT Reader, which loads the documentation selected. Like the Input Manual all the documentation are in PDF format and installed by default in the directory “RelapRootDirectory\Manuals” (s. installation chapter). The caption of the menu items is set by the initialization of the RelapPP application. The user can change these in the “Edit Configuration File” sub menu, which is described in the next chapter. If any of the files is missing in the manuals directory or the file name is misspelled an error message warns the user.

3.3.6 Options, Utilities and Add-on Programs

Most of the utilities, some of the options and add-on programs are available from the Top Level Menu “Utilities and Options”. The next figure shows the submenus of this menu. Most of the utilities and options deal with configuration subjects. There however is one configuration aspect which has been put to the Top Level Menu “Projects”: The “Paths and Directories” submenu (s. Figure 3.3-2). This submenu provides access to the basic directory structure of the RelapPP application. This command opens the same directory name dialog as is automatically popped up if the application is run the first time. The procedure of changing and adapting the directory structure is explained in chapter 4. The user is advised that it is not recommended to change the directory structure of a current project. It may be convenient only to change the user’s directory for different projects. Another configuration aspect has already been explained in the description of the “Relap Command Center”: The change of the directory for the current results, the current restarts directory, the intermediate change of the Relap Executable, and the steam table. These options have been put to the Relap Command Center because during Relap analyses it may be convenient to better structure the current project. The access to other Relap versions via the executables may be helpful locating Relap errors by comparison to older Relap versions.

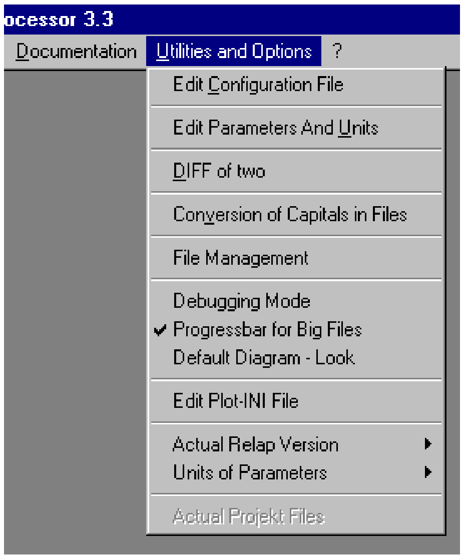


Figure 3.3-42: Utilities and Options Menu

All the 12 submenus are described subsequently.

Edit Configuration File

This command starts the built in simple text editor which allows editing of the Relap – INI file. The contents of this file are shown in table B-8 (Annex B). The INI file consists of several elements and sections. The sections are headed by section headers and keywords, which must not be changed. All comments are preceded by ";" (semicolon). The end of the file is marked by "; EOF". The sequence of the different sections must not be changed either. The sections are:

1. Start-up

In this section the current major Relap version, the Relap executable, the main steam table and the directory structure of the current project is referenced. These data will be read by the application during startup and saved internally. If the directory structure read from the INI file cannot be verified by the program the user will be notified and provided with the directory name dialog box as described earlier (s. chapter 4). Normally all output will be saved in the user directory. The user however can change the output and restart file path for the current project by the "Path and Directories" submenu (explained in previous chapters) or in the Relap Command Center. The Relap Version information should not be changed also. This information is used by the program to distinguish between different Relap versions, because during the development of the application Relap Output design and Restart design has been changed.

If the user selects a new Relap project and he is quitting the RelapPP he will be asked if the new project should be the current or default one for the next start up. If the user affirms the new data is saved in the INI file.

2. Manuals

All the manuals are installed by default in the "ManualsDir". If the user wants the manuals in another directory, he has to change the data for this keyword. The next 8 keywords contain the text of the captions of the command options of the "Documentation" top level menu. The program reads this information during startup and dynamically adapts the submenu items, even the width of the menu bars. The user can change the text and adapt to his needs. The next 8 keywords "Pdf(0..7)" contain the associated filenames of the documentation files. The manuals data only can be changed by the INI-editor option.

3. Run

The Run section contains the base-name of the current Relap project and the default file suffixes of the input, the output, the restart, the strip file and the Log-file. At startup the RelapPP application initializes the current Relap project, using this data. All of this data is subject to change with changing to another Relap project or changing the suffixes in the Relap Command Center. As explained earlier the user always will be asked to confirm the change to another project at the end of the RelapPP application.

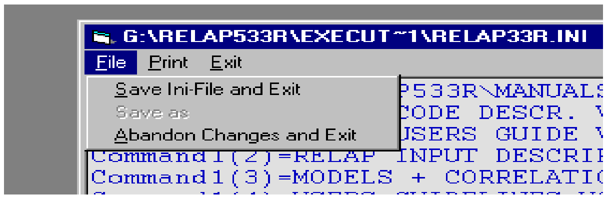
4. Editor

The INI file contains link data to two different editors which are used by RelapPP. The DOS-editor is currently not working, has to remain however for internal reasons. The Win_Small_ed – Editor will be used for small files like the Log-File, RSNIF files and other small files (< 16KB) The Win-Big_ed – Editor will be used for rather large files. The user can define which editor shall be used for small files and which one for big files. The RelapPP installation package provides the option to install a rather powerfull freeware editor (PAD97) for Big files. Users who like to install the PAD97 editor have to obey to the License terms and conditions for the use of this program (see Chapter 4 for more information). For small files the standard Notepad is a good choice. Any other editor can be used. The RelapPP application additionally provides a built-in simple text editor for special purposes, especially for the INI files. This editor can also be used for small files optionally (explained in the Relap Command Center chapter).

5. Browser

The browser section contains the link to the PDF Reader. The RelapPP installation provides the option to install the latest ACROBAT Reader. However any Reader installed in the system (Acrobat Version 4 and up) can be used. The link to the ACROBAT Reader executable has to be changed in this case.

The INI file can be printed on the default printer or saved, if changes have been made. (s. Figure 3.3-43). In this case all the data will be read again and checked and all the internal data of the program will be updated with respect to the changes made.



```
P533R\MANUALS
CODE DESCR.
USERS GUIDE
Command1(2)=RELAP INPUT DESCRI
Command1(3)=MODELS + CORRELATI
```

Figure 3.3-43: INI Editor Menu

Edit Parameters and Units

This command again starts the built in simple text editor loading the UNITS - INI file. The contents of this file are shown in Table B-9 (Annex B). This INI file contains all of the Relap parameters which are available from any of the Relap data sources (Minor edits, Strip data, Restart Plot data). As has been explained earlier, the Relap Plot Data mostly comes without any units or nomenclature of the parameters (despite the Minor Edits). The data from this file will be read by the program in case of data visualizations, plotting and plot data table generation (s. previous chapters) and will be used for the automatic description of the graph axes.

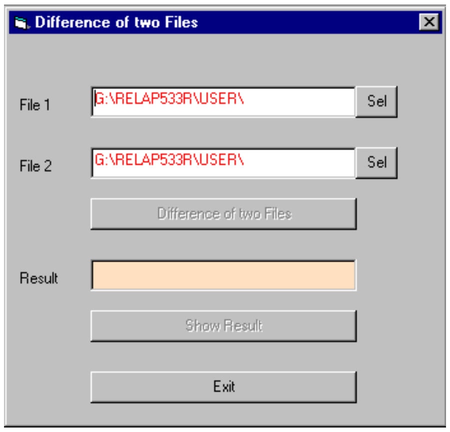
The data in this INI file can be edited, including data removal or additions of new parameters. All of the cautions to handle the file are shown in table B-9:

- Data begins at line number 5 (the first 4 lines are ignored)
- Any data line has a total width of 77 characters including the "." (period) at the end-of-line
- The column width for the parameter data is marked by a ruler in the fourth line
- Empty lines are allowed for better structuring (no end of line period)
- Any number of new parameters and their description can be added
- The last line is marked with a "@" at the first place.
- The sequence of data is not important

The INI file can be printed on the default printer or saved. As shown for the Relap – INI file.

Diff of two

The purpose of the "Diff of two" Command is the comparison of two ASCII files and the documentation of the changes. The command provides a file selection dialog as shown in the next figure.

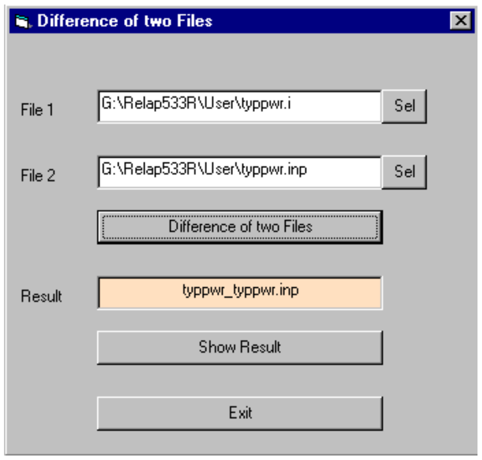


File 1: G:\RELAP533R\USER\

File 2: G:\RELAP533R\USER\

Figure 3.3-44: File Difference Dialog

By default the files are assumed to be in the Users-directory. If the user selects one file and wants to compare it to the same file he will get the warning that the files are identical and no action will be taken. If the files selected have different file names and they are identical, the user will be notified also.



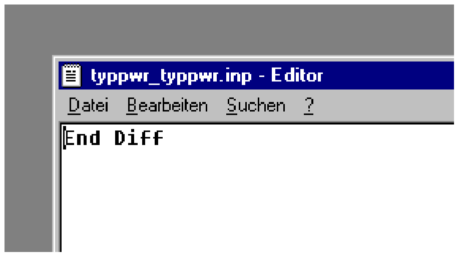
File 1: G:\Relap533R\User\typpwr.i

File 2: G:\Relap533R\User\typpwr.inp

Result: typpwr_typpwr.inp

Figure 3.3-45: File Difference Results

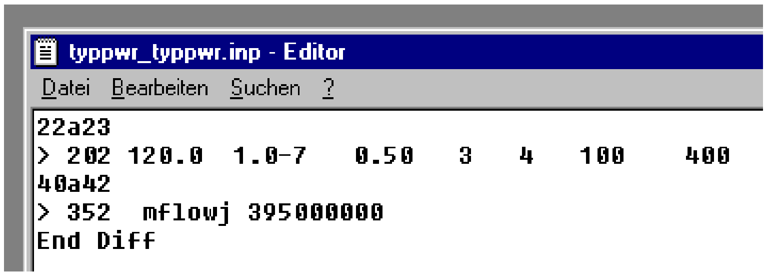
The result of the comparison will be saved in a file with a meaningful file name as shown in the example above. The result can be viewed by the “Show Results” command. In case the files are identical the contents of the results file is shown in the figure 3.3-46.



```
End Diff
```

Figure 3.3-46: DIFF Result of identical files

The result of the comparison looks like the screenshot in Figure 3.3-47, if the second file, for example contains two additional lines.



```
22a23
> 202 120.0  1.0-7   0.50   3   4   100    400
40a42
> 352  mflowj 395000000
End Diff
```

Figure 3.3-47: DIFF Result of two different files (File 2 with 2 additional lines)

The representation of the DIFF results is almost the same as known from the Unix DIFF command. The DIFF function internally is performed by a small external program, which is controlled by command line options.

The DIFF feature not only can be used for Input files, but also for big Output files or plot data files to search differences between different calculations.

<u>Conversion of Capitals in Files</u>

Relap unfortunately does not allow capital letters in input files (Unix as well as Intel versions). Only comments in comment lines marked with the "*" are allowed to be spelled in capital letters. Relap reacts unpredictable if capitals are found in the input data stream and the search for the error is painful for big models. A common error is the spelling of floating point numbers like 1.345E3. To help the user with eliminating this error source and to convert e.g. older Relap models which are still with capital letters (older Relap version prior to 5 MOD 3 did accept capitals) the conversion utility has been integrated into the RelapPP.

After starting the utility the user gets a file selection dialog window where he can determine the input file to be processed. After selection of this file he has to define the filename of the converted file. Per default the Utility suggests to add "LowerCase" to the original file name. The Utility ignores any comments in capitals. After the conversion the user is asked if he likes to edit/view the new file.

<u>File Management</u>

The file management option, which also is available from the Relap Command Center, provides a link to an external file management application (TN Turbo Navigator) which is supplied with the installation packet and which is rather powerful. Users who like to install the Turbo Navigator have to obey to the License Terms and Conditions which come with the TN installation (see Chapter 4 for more information). If TN is not installed, RelapPP launches the standard Windows EXPLORER instead.

Debugging Mode

The “Debugging Mode” (s. Figure 3.3-42) is a menu option which can be checked like a check box. The startup default is “disabled”. If the debugging mode is enabled additional information is displayed in message boxes to help the user understand data transfer or to know which files are at work in case of troubles. The following list shows where additional information is given. There may be additional needs for more information. These are subject for implementation from case to case.

- Relap Command Center: Additional information at starting a Relap run and generation strip and restart input files
- Minor Edit and Restart Plot Data processor: Extracting the Data and transfer of data for plotting graphs
- Plotprocessor: Additional list used internally for the selection of paramenters and selection of Plot Data files
- Conversion of capital letters into lower cases
- Difference of two files

Progressbar for Big Files

There are operations in connection with the plot data retrieving which take some time for big files. In these cases there are “Progressbars” built in some child application windows which notify the user that the retrieval process is still going on. The “Progressbar for Big Files” flag is set by default at startup of the RelapPP. The operation however takes some processor time itself. The option can be disabled if not needed. The progressbar is implemented in the Plot Data Extraction processor for Relap Output and Restart files and for the extraction of selected Plot data from PLOUT files. However if enabled the progressbar is active for the Minor Edit Data processing only for output files larger than 50 Kbytes.

Default Diagram – Look

The “Default Diagram – Look” option has been described already in chapter 3.3.4.4 “Plot Post Processor”. The menu option and its linkage to the options on the plot post processor application window are shown in figure 3.3-35.

Edit Plot-INI File

The “Edit Plot-INI File command is disabled if the “Default Diagram – Look” option is checked as shown in the next figure.

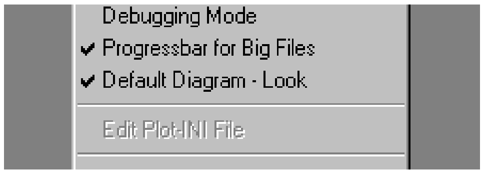


Figure 3.3-48: Edit Plot-INI File

If the command is enabled the user can edit the PLOT33R.INI file with the built in simple INI editor. The figure A-11 shows an example of the PLOT33R.INI file and the meaning of the parameters. More details are explained in the chapter 3.2.4.4.

Actual Relap Version

As shown in the next figure this command only displays the current actual Relap version which RelapPP is working with. Originally a selection of different Relap Versions was intended. This, however, has been removed due to excessive management and overhead load in the application. As has been explained in the chapter “Relap Command Center” the user still has the choice to implement different Relap Versions by the selection of different executables. In this case the user is advised to accept error messages from RelapPP because of design changes of Outputs and Restarts from Relap version to Relap version.

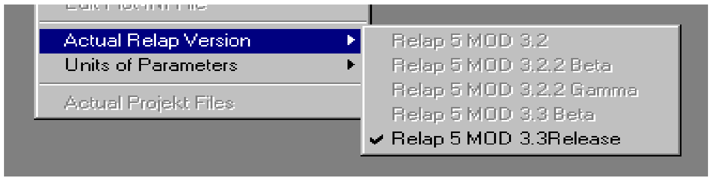


Figure 3.3-49: Relap Version

Units of Parameters

As shown in the next figure the RelapPP application by default uses the SI units for plotting and graphing. British Units cannot be selected at the moment, however this option will be implemented in the future. The restriction to SI units, however, is not a real limitation, because SI is the international standard now and the user can use British units by modifying the UNITS.INI file as explained previously. (see Table B-9).

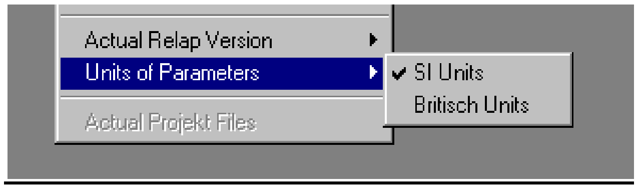


Figure 3.3-50: Units

Actual Project Files

As shown in the figure above (3.3-49) the “Actual Project Files” command is disabled. As a replacement this command has been implemented into the Relap Command center.

3.3.7 Restart Postprocessor

The Restart Postprocessor RSTR533R which is called by RelapPP is an external program controlled by command line options. The programming language used is FORTRAN95 which guarantees compatibility with the INTEL Relap version. The postprocessor has been designed to run on 32 Bit Windows platforms like Windos95 / 98 / NT4.0, Windows2000 and XP. The main features of RSTR533R are:

- Extract PlotData from Relap Restart/Plot-File
- Create Information files for Plots and Restarts
- Create Informations of Plotparameters for Strip runs
- Create automatic Strip Input for RELAP 5/MOD3.3

RSTR533R has been designed to read and process restarts, including appending and replacing restarts of RELAP5MOD3.3 Release Version running on:

- Intel (PC) Version
- IBM AIX
- DEC Alpha
- HPC180
- SGI84
- SUN OS5.6

The program can be used also as standalone program within the windows environment (DOS-BOX or CMD-Shell) with the command line options shown in the next table. As part of RelapPP the program is used in silent mode (option –no). The command line options and the organization of options are known from typical UNIX programs or from Relap itself.

Table 3.3-3: Command Line Options of the Relap Restart Postprocessor

Options		**Default**
-ri	Restart/strip input file name	RESTINP
-ro	Restart output w/o plot data	RESTOUT
-or	Restart information file	RESTINF
-po	Plot parameter & data file	PLOTOUT
-op	Plot parameter info file	PLOTINF
-so	Strip input file for RELAP	STRIPINP
-r	Write restart output file	
-p	Write plot data file	
-s	Write strip input file	
-d	Debug data file	DEBUG
-no	No screen output	
-h / ?	Help	

The Restart Postprocessor creates all the necessary files used by RelapPP (Restart Information, Plot Data Information and the PLOUT file). The program also generates the Strip Input file described in the Relap Command Center chapter. Without the options “-p” and/or “-r” by default only the RESTINF and PLOTINF data will be written. When used as stand alone program the program allows the extraction of any plot data from a restart file and write the restart file again without the plot data (this saves hard disk space in some cases). If Unix restart files are processed (e.g. from IBM AIX or SGI installations or any other non-INTEL restart files) the RESTOUT file will not be written by default.

The “-h /?” option provides the user with a UNIX like simple help screen of how to use the command line parameters. The debug option produces some very helpful details regarding the single steps of working through a restart file including binary information. This is helpful for reading non – INTEL restarts or for finding the sources of troubles. The debug information will be visible on screen and will be saved in the file named “DEBUG”.

The source code of the Relap Restart Postprocessor is listed in Annex C. The data record definitions of Relap restarts (RSTPLT, PLOTINF, PLOTALF, RESTART and PLOTREC records), the technique of reading non-INTEL restarts and the maneuvering through a restart is described in the source code comments. The actual version of the program allows the processing of restarts of Relap 5 MOD 3.3 Release (aa..zz) versions. The version will be checked by the program. Any other version will be rejected.

3.4 Limits

The size of a certain Relap project literately is only limited by the PC hardware and the Windows environment. However there are some practical limits with respect to calculation speed for the Relap calculations itself and the post processing of big data files. One of the future improvements of the program will be to enhance the data extraction speed which currently is influenced considerably by the not optimal internal data management. This drawback however can be compensated by more powerful INTEL CPUs and faster Hard Disks.

Most of the limits are imposed by Relap itself, like the limit of maximal variables, Minor edits, Strip parameters and others. We (Westinghouse) have successfully calculated Relap problems on PC with going to the Relap limits and Restart file sizes up to 20 Gbyte. Even data files this big could be post processed with RelapPP.

The most prominent limits with RelapPP are the following (Some are described in previous chapters):

- The file name length including the path for Relap runs is limited to 41 characters
- The file name length including the path for stand alone Restart Post processor is limited to 60 characters
- File length of restarts or Output files > 2147 483 647 Byte (2Gbyte) cannot be shown in the file length data fields in the Relap Command Center or the Postprocessing application

window. In these cases a message is presented that the file length exceeds this limit.

- The number of parameters in PLO file is limited to 99 (+ time) with respect to the total maximal number of minor edits in Relap runs.

- The number of parameters in strip files is limited to 999 with respect to the total maximal number strip variables in Relap runs.

- The number of parameters for the conversion of selected data from PLOUT format is limited also to 99 (+ time) with respect to PLO format

- The maximum number of parameters for restart files from which variables can be selected is limited to 32167 (internal pointer is limited to 2^{15} -1). This limit can be overcome as explained previously by the stripping of selected data.

- The maximum number of parameters for restart files for the direct access to the plot da ta from Restart files is limited to 32167 (internal pointer of the selection table is limited to 2^{15} -1)

- The maximum number of time steps which can be processed by EXCEL is limited by EX CEL to 32168 (The number of Columns (all Plot Parameters + Time Column) is limited to less than 100. With respect to the dynamically allocated data arrays the number of columns allowed minus 1 times number of rows should be less than 524288 (2^{19}). This has been explained in a previous chapter

3.5 Tips, Hints and Troubleshooting

Most of the applications built in the RelapPP Pre- and Post processor are equipped with meaningful information to guide the user through all the steps of a full Relap analysis. Every of the bugs usually found in any software product should be eliminated now based on the extensive use of the program within Westinghouse and useful hints from other users form the CAMP community. However it is still a software product and all of the multi thousands of combinations of different successive steps for the performing of analyses and of maneuvering through an object oriented program cannot be tested sufficiently. If the user gets unexpected messages or experiences unexpected behavior he has the choice first to activate the debug mode and check again and second to analyze the problem step by step.

Most errors however come from data files which for some reason do not contain the correct information expected by the program. One of the first actions to analyze problems should be to view the participating data files, especially the PLO data file, the PLOUT file, the RSINF and the PLINF file. All the data formats used are listed in Annex B.

Another problem can come from the extraction of data from the binary restart files. The debugging is described in the previous chapter. Often, if there are for some reason no resulting files, another try to extract the data from restarts helps to solve that problem. The reason for that problem sometimes could be traced down to Windows internal processes. After starting the external “Restart Post Postprocessor” program RelapPP permanently checks if the process is still active and if the files, which have to be generated have been written and closed. This very seldom is boykottet by Windows.

If strip output files are processed, the user should be sure, the data format of the strip files is in the ASCII format. The format will be checked by RelapPP. However, the user may not understand the error message if the file is in the ASCII format, and only some important information is missing at the head of the file.

Very big files sometimes take long to process. The user should not become unpatient, especially when the progress-bar mode is not activated. He much more should check the CPU activity with help of the Windows task manager or the hard disk activity LED on the Desktop or Midi Tower case to decide if there is a hang up or not.

4 SYSTEM REQUIREMENTS, INSTALLATION AND CUSTOMIZED CONFIGURATION

The RelapPP Pre- and Post Processor is distributed by CD-ROM. All of the programs and Libraries needed for a complete RelapPP installation are provided. The CD contains:

- The RelapPP Pre- and Post Processor for Relap5MOD3.3Rrelase (eventually a Patch of the latest sub version on diskette)

- Relap5MOD3.3Release, latest sub version with all accompanying files and documenta tions as distributed by the NRC

- The Restart Post Processor RSTR533R.EXE

- The UNIX like Diff-utility DIFF.EXE

- Acrobat Reader 5.x, as distributed by Acrobat (including the Acrobat license agreements)

- PAD97 big file editor by GTSoft, Version 1.05, Copyright © 1997-98 by Gustavo Tondello, Pad 97 is a replacement for Notepad wich has many features not present in Notepad. All accompanying files including the license agreement and terms and conditions of use

- Turbo Navigator v1.44, Copyright © 1999-2000 by Marko Vodopija, www.TurboNavigator.com, with all accompanying files including the license agreement and terms and conditions of use

- XMGR5 for INTEL as distributed by the NRC

- Testversion of Xmanager(R) version 1.3.9, Copyright (c) 1997-2002 by NetSarang Com puter, Inc., Xmanager is a high performance X11R6 PC X server for Windows 95/98/ME and Windows NT/2000/XP. If you are not a registered user, you can use this program 30 days only for evaluation and that any other use requires purchase of a license. All informations are included.

The hardware requirements for RelapPP are the following:

- Standard PC with Intel Processor (CPU frequency: 200 Mhz as a minimum, 2.0 Ghz and up is recommended for good performance for big problems)

- A minimum of 64 MB of Ram. 256/512 MB for PIV processors

- A minimum of 100 MB of Hard Disk - Space. Up to 30 GB needed for big problems

- Fast Hard disk drive (7200 rpm and up) for good performance

- Standard graphic devices with medium resolution minimum 800*600 (recommended: 1280*1024)

- Windows Operating System   95 / 98 / ME / NT4.0 / 2000 / XP latest Service pack
- EXCEL 95/97, 2000, 2002 for Plotting Graphs (EXCEL comes with the MicroSoft Office packages)
- Acrobat Reader Version 4 or up (provided with the installation packet)

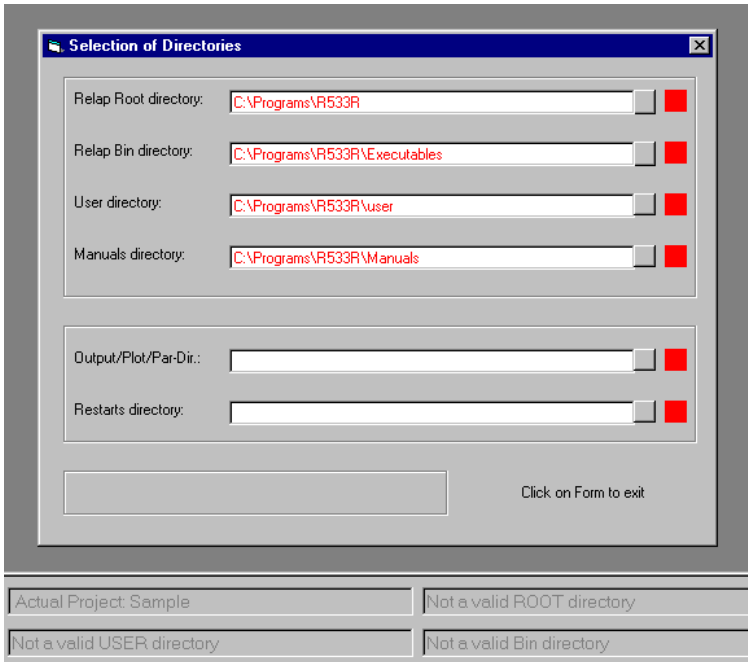
The installation of the program is performed automatically by the autostart function of the CD-ROM or manually by starting the setup.exe program in the root directory of the CD. The setup program guides the user through the complete installation process. A typical, a full and a user defined installation is provided for the users convenience.

The Installshield process creates the following directory structure in the default directory (can be changed) c:\program:

- Relap Root Directory:	C:\Program\R533R (~)
- Acrobat Directory:	~\Acrobat
- Relap Bin Directory:	~\Executables
- Guidelines Directory:	~\Guidelines
- Manuals Directory:	~\Manuals
- Relap User Directdory:	~\User
- Xmgr5 & X-manager	~\Xmgr

Relap5 MOD 3.3R, the steam tables, the relapPP pre- and Postprocessor and all of the INI-files, the Restart Postprocessor RSTR533R, the DIFF-utility, TN and PAD97 and all files belonging to these are installed in the Excecutables directory. All of the Manuals are installed in directory ~\Manuals, the guidelines provided by NRC are copied into the guidelines directory. The test input, the test output and the test restart files are copied into the users directory. In order to obey to the limit of file name length for Relap runs it is strongly advised to change the “Relap Root Directory” to C:\R533R for example. The installation routine registers all the programs, creates several Icons on the desktop, and selections on the “Windows Start Menu”, depending on the installed programs which come with the distribution CD.

As shown in Table B-8 (Annex B) the Relap533R-INI file contains the actual directory structure. In the distribution CD the INI file contains a default directory structure. At the first startup of RelapPP the directory structure probably has to be changed. In case the provided defaults cannot be verified by the program at the first startup the following screen with directory selection dialogs pops up. The user has to change all the paths marked red.



Selection of Directories

Relap Root directory: C:\Programs\R533R

Relap Bin directory: C:\Programs\R533R\Executables

User directory: C:\Programs\R533R\user

Manuals directory: C:\Programs\R533R\Manuals

Output/Plot/Par-Dir.:

Restarts directory:

Click on Form to exit

Actual Project: Sample

Not a valid ROOT directory

Not a valid USER directory

Not a valid Bin directory

Figure 4-1: Directory structure adaptation dialogue

He cannot continue without a correct directory structure. He additionally has the choice to set a path for Outputs, Plot data and for restart data at this time. This information, however, will not be saved in the INI file as explained in previous sections. Whenever the directory structure found in the INI file cannot be verified by RelaPP this dialog starts at the beginning. This can happen, if the user changes directory names outside the control of the RelapPP application. Red colored path names indicate that the path cannot be found. This information is also given in the message boxes at the bottom of the main screen of the RelapPP application.

Once the directory structure has been adapted, the user should check and modify if necessary the following details prior to start any Relap analysis:

- Description of the commands to view the Relap manuals and the filenames (explained in chapter 3.3.6, section "Edit Configuration File")
- Path to the Windows Editor NOTEPAD or WORDPAD
- Path to the big-file editor PAD97 if installed or to any other capable editor
- Path to the Acrobat Reader if the Relap Root Directory na me has been changed or to Acrobat as already installed on the system
- Check of the filename of the "Quick-Look" documentation for RelapPP in the Manuals directory. If necessary, the file name has to be changed to "Quick-Look.PDF".
- Change or adaptation of the units of the parameters for the data plotting. This is described in chapter 3.3.6 in section "Edit Parameters and Units"
- Change or adaptation of the plot settings for the data plotting. This is described in chapter 3.3.6 in section "Edit Plot-INI File"

5 CONCLUSIONS AND OUTLOOK

The RelapPP Pre- and Post Processor have been developed to support the Relap analyst. The tool is not intended to replace any of the NRC tools like Xmgr5 or SNAP. It is intended to be an add on for Relap PC users who like to work with an integrated application where all of the programs needed for complete Relap analyses are available by key stroke. The tool has proven its effectiveness during long time usage within Westinghouse.

The development of the Pre- and Post processor has reached now a status of projected completeness. However, there still are ideas for improvements and additions. Some of these ideas for future development are:

- Enhancement of performance for very large problems
- Extension of the 32167 Parameter Limit for the direct selection for the Restart Processing
- Extension or workaround of the time step limit of 32168 which comes with EXCEL
- Integration of Xmgr5 or other Plot-Package (Simple Plots with HPGL and Quick Viewer)
- Extension of plotting Capabilities when using EXCEL by user-specific layouts and easier plotting of clipped time sections
- Addition of Relap Project specific Sets of Parameters and the Management of these Sets (comparable to Xmgr5)
- Input Preprocessor with User Guidance and Intelligence
- Implementation of a Help System

6 REFERENCES

/1/ Dale M. Snider et al. “Nuclear Plant Analyzer (NPA), EGG-EAST-9096
EG&G Idaho Inc. April 1990

/2/ Paul J. Turner, “ACE/gr, Graphics for Exploratory Data Analysis” Oregon, Graduate In stitute of Technology, 1991 – 1995, XMGR5 Extensions by K. Jones, Idaho Nation al Engineering and Environmental Laboratory, further developed by ACE/gr Development Team (1995 – 1998), currently maintained by Evgeny Stambulchik, Rehovot, Israel. Actual version: 4.1.2I

/3/ Symbolic Nuclear Analysis Package SNAP, U.S. Nuclear Regulatory Commission, under development

/4/ B. Worth et al. “Generation and Verification of a RELAP5/MOD1-EUR Code Version Running on IBM RISC-6000 Workstations”, Joint Reserch Center ISPRA Site, Safety Technology Institute, Tech.Note I93.140, October 1993

/5/ “Users Manual Microsoft Excel 97”, Microsoft Corporation, 1997

Annex A-1

Figures

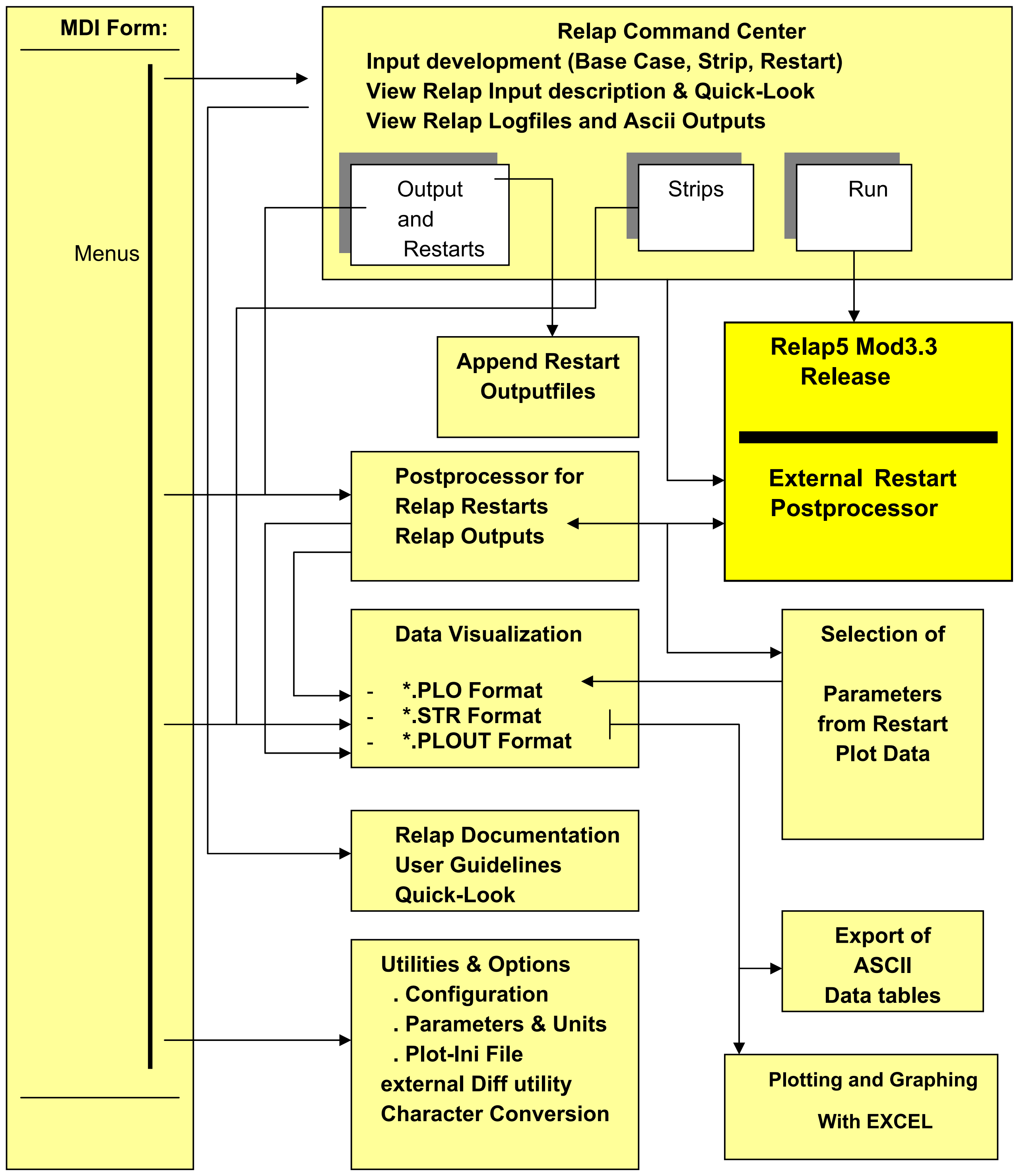
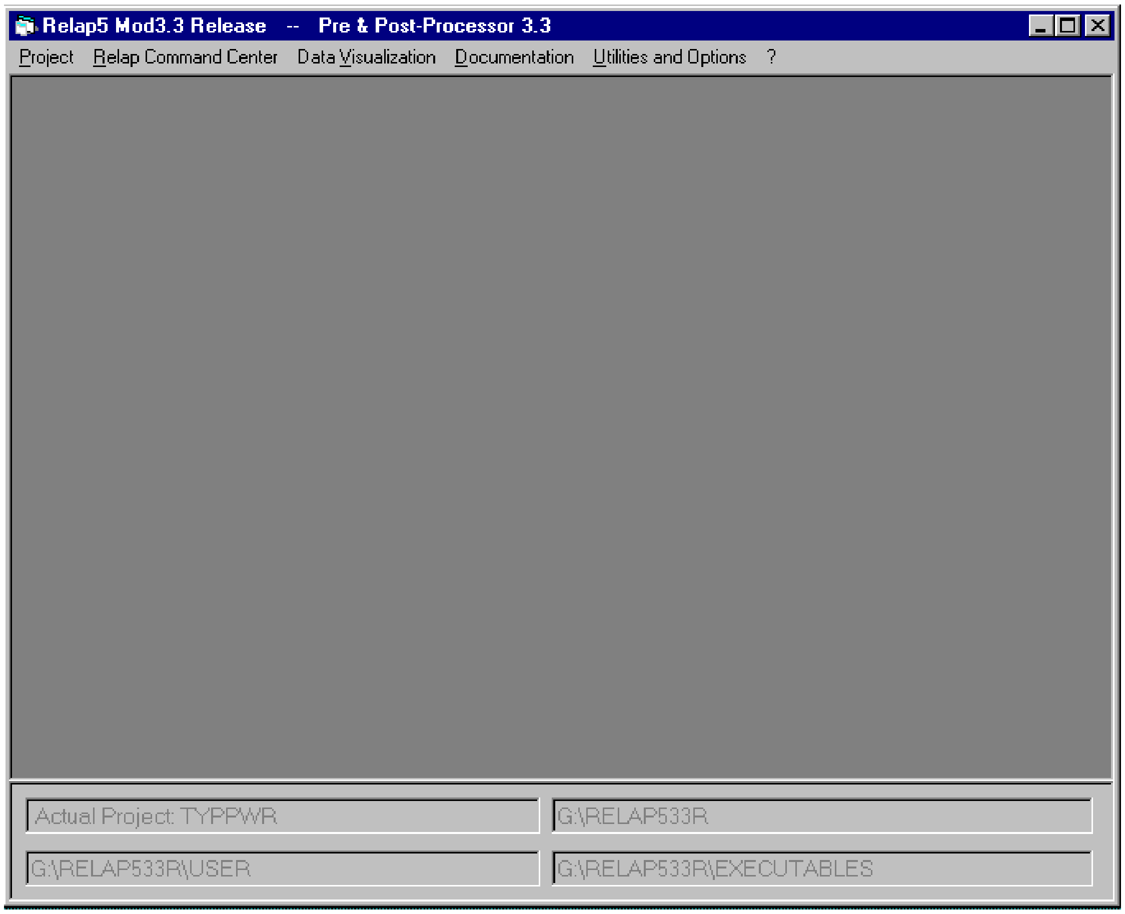

MDI Form:
Menus
Relap Command Center
Input development (Base Case, Strip, Restart)
View Relap Input description & Quick-Look
View Relap Logfiles and Ascii Outputs
Output and Restarts
Strips
Run
Append Restart Outputfiles
Relap5 Mod3.3 Release
External Restart Postprocessor
Postprocessor for Relap Restarts Relap Outputs
Data Visualization
- *.PLO Format
- *.STR Format
- *.PLOUT Format
Selection of Parameters from Restart Plot Data
Relap Documentation User Guidelines Quick-Look
Export of ASCII Data tables
Utilities & Options
. Configuration
. Parameters & Units
. Plot-Ini File
external Diff utility
Character Conversion
Plotting and Graphing With EXCEL

Figure A-1: Principal Structure of RelapPP



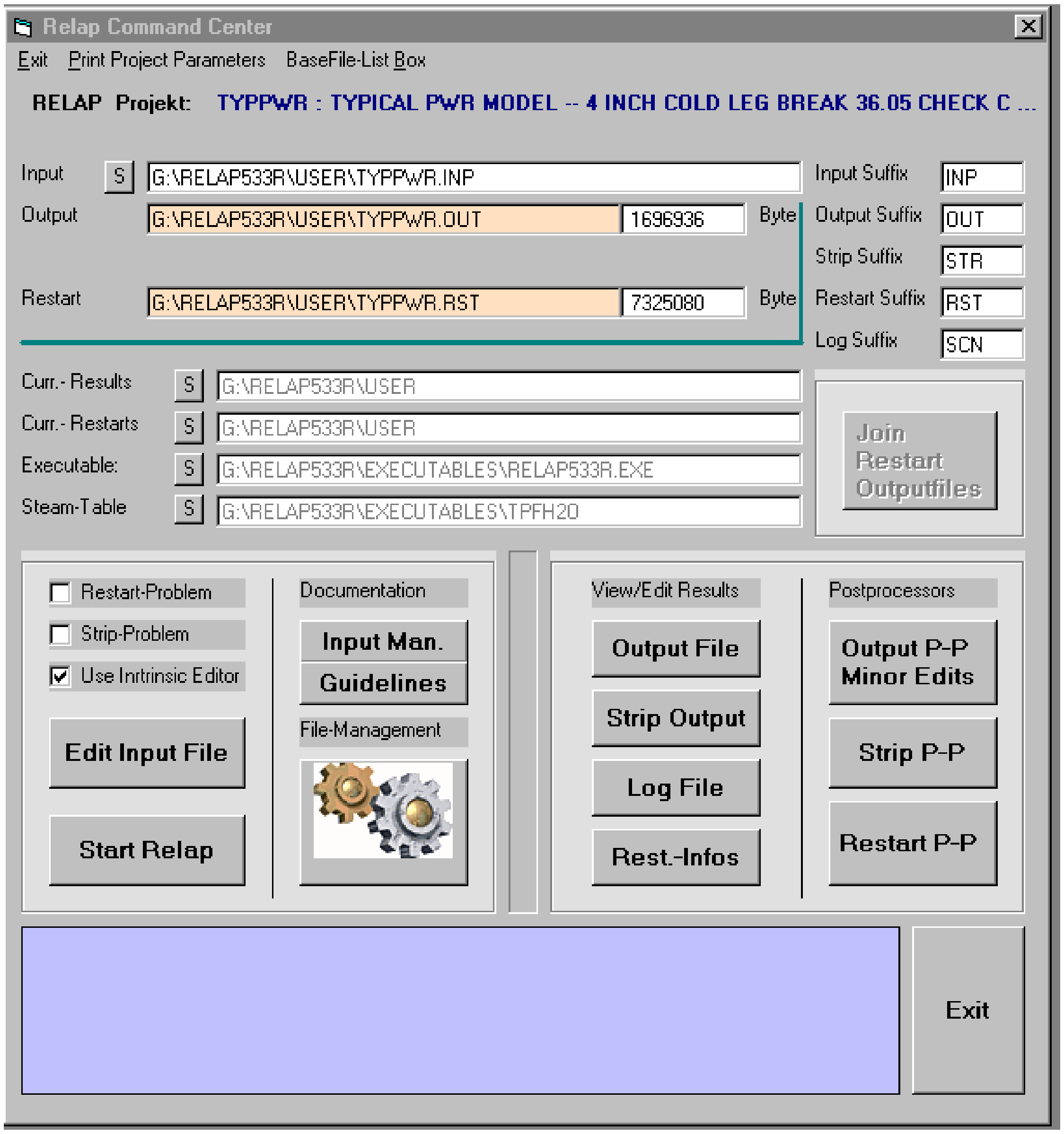
Actual Project: TYPPWR

G:\RELAP533R

G:\RELAP533R\USER

G:\RELAP533R\EXECUTABLES

Figure A-2: Top Level Window


Relap Command Center

Exit   Print Project Parameters   BaseFile-List Box

RELAP Projekt: TYPPWR : TYPICAL PWR MODEL -- 4 INCH COLD LEG BREAK 36.05 CHECK C ...

Input S G:\RELAP533R\USER\TYPPWR.INP

Output G:\RELAP533R\USER\TYPPWR.OUT 1696936 Byte

Restart G:\RELAP533R\USER\TYPPWR.RST 7325080 Byte

Input Suffix INP

Output Suffix OUT

Strip Suffix STR

Restart Suffix RST

Log Suffix SCN

Curr.- Results S G:\RELAP533R\USER

Curr.- Restarts S G:\RELAP533R\USER

Executable: S G:\RELAP533R\EXECUTABLES\RELAP533R.EXE

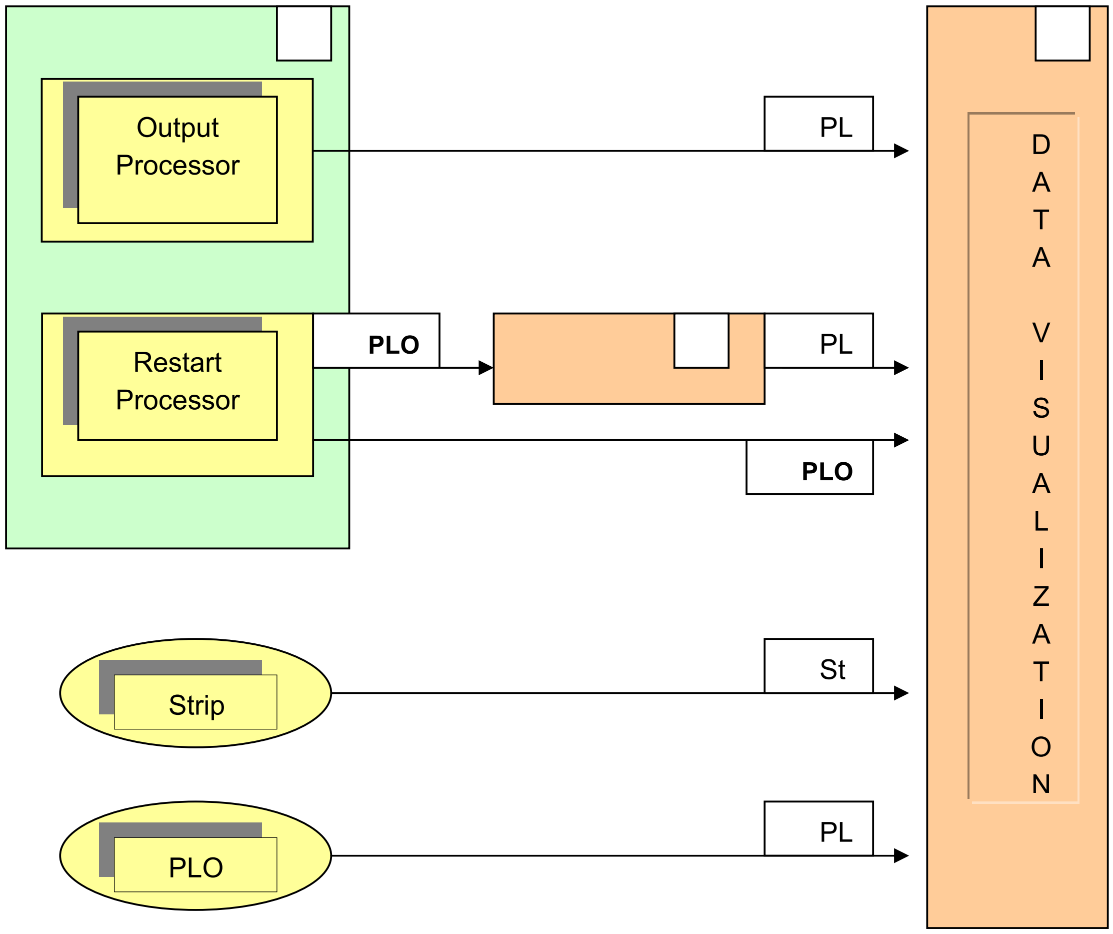
Steam-Table S G:\RELAP533R\EXECUTABLES\TPFH2O

☐ Restart-Problem

☐ Strip-Problem

☑ Use Inrtrinsic Editor

Figure A-3: Relap Command Center

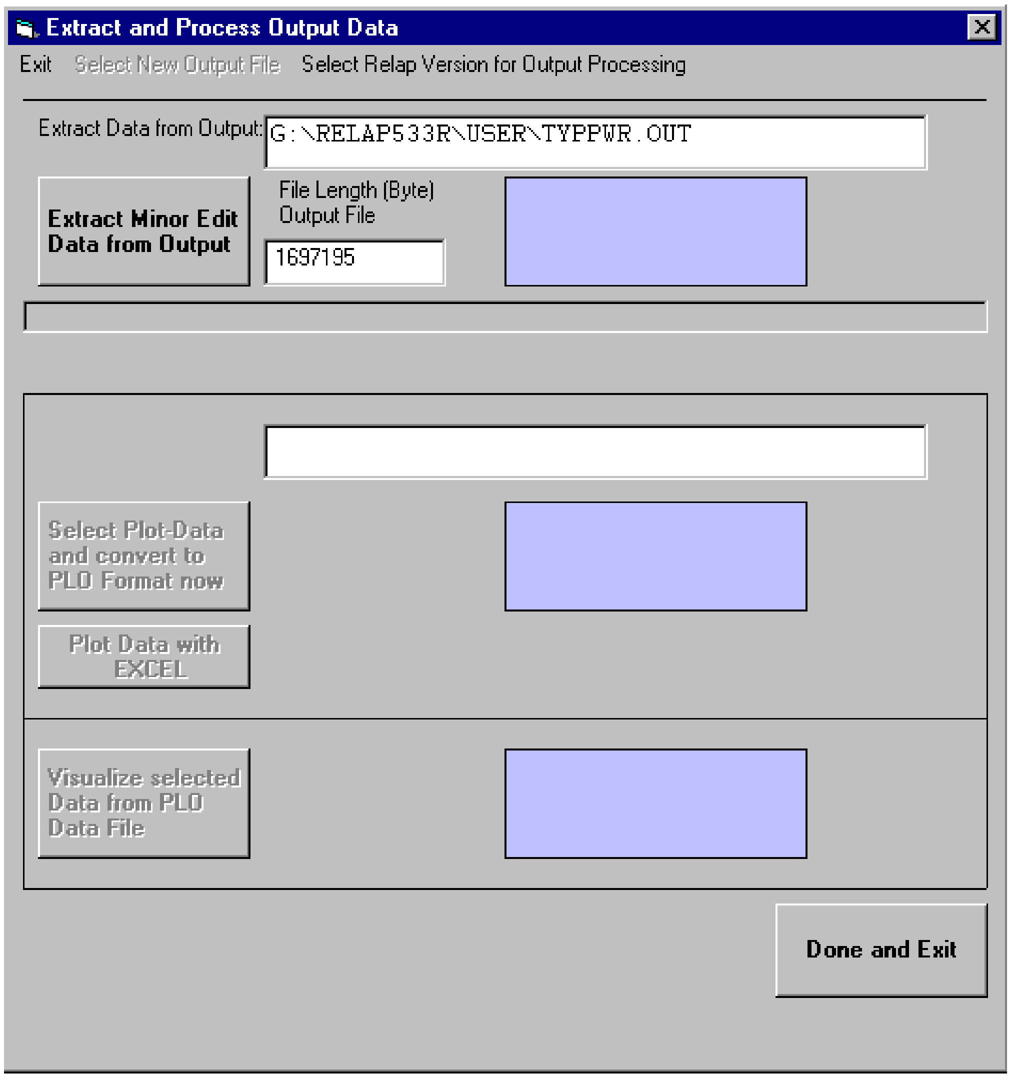


1: “Extract and Process Output / Restart Data”

2: “Parameter Selection” (PLOUT to PLO Conversion)

3: “Data Visualization” (+ Ascii Tables + Plotting with EXCEL-engine)

Figure A-4: Structure of the Post Processors


Extract and Process Output Data

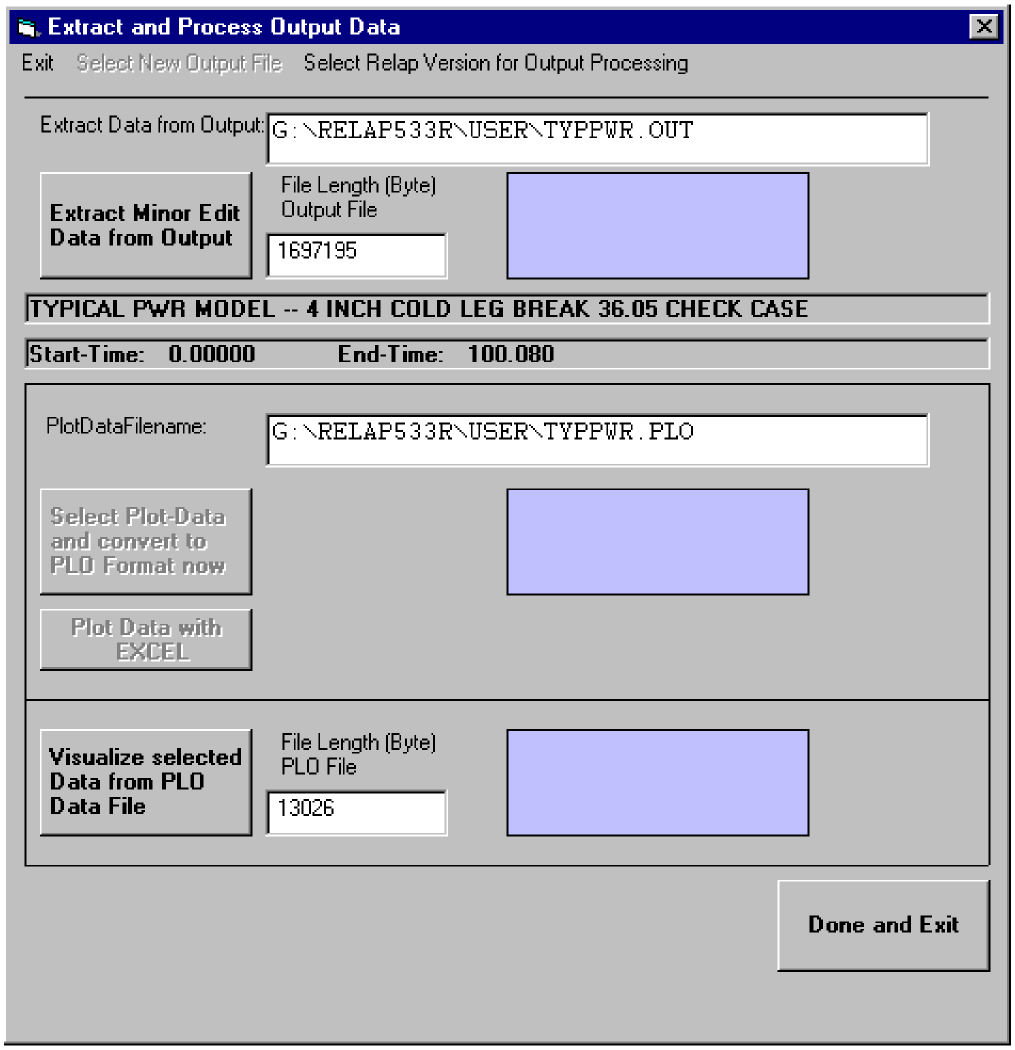
Exit   Select New Output File   Select Relap Version for Output Processing

Extract Data from Output: G:\RELAP533R\USER\TYPPWR.OUT

File Length (Byte) Output File: 1697195

Extract Minor Edit Data from Output
Select Plot-Data and convert to PLO Format now
Plot Data with EXCEL
Visualize selected Data from PLO Data File
Done and Exit

Figure A-5: Extract and process Output Data


Extract and Process Output Data
Exit
Select New Output File
Select Relap Version for Output Processing
Extract Minor Edit Data from Output
Select Plot-Data and convert to PLO Format now
Plot Data with EXCEL
Visualize selected Data from PLO Data File
Done and Exit

Extract Data from Output: G:\RELAP533R\USER\TYPPWR.OUT

File Length (Byte) Output File: 1697195

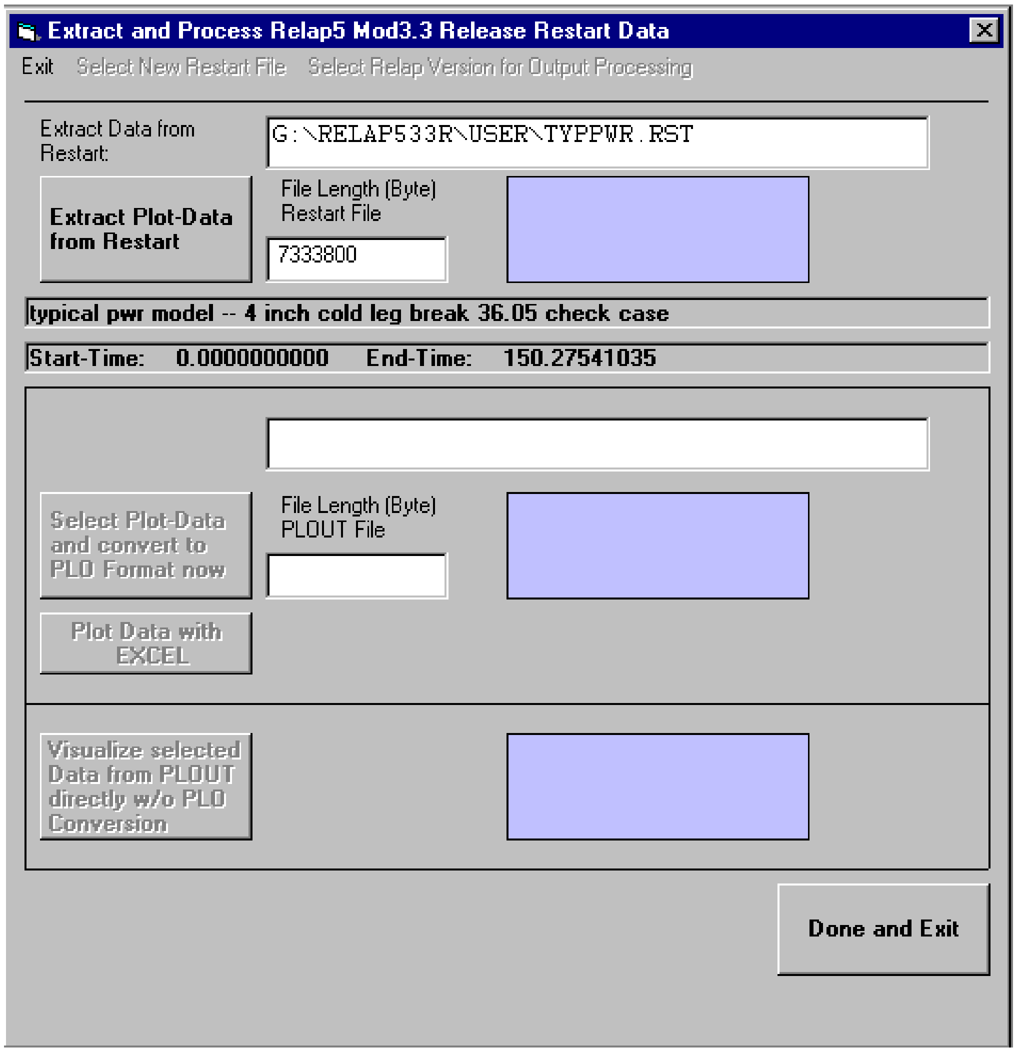
TYPICAL PWR MODEL -- 4 INCH COLD LEG BREAK 36.05 CHECK CASE

Start-Time: 0.00000 End-Time: 100.080

PlotDataFilename: G:\RELAP533R\USER\TYPPWR.PLO

File Length (Byte) PLO File: 13026

Figure A-6: Extract and process Output Data (PLO file exist)


Extract and Process Relap5 Mod3.3 Release Restart Data
Exit
Select New Restart File
Select Relap Version for Output Processing
Extract Plot-Data from Restart
Select Plot-Data and convert to PLO Format now
Plot Data with EXCEL
Visualize selected Data from PLOUT directly w/o PLO Conversion
Done and Exit

Extract Data from Restart: G:\RELAP533R\USER\TYPPWR.RST

File Length (Byte) Restart File: 7333800

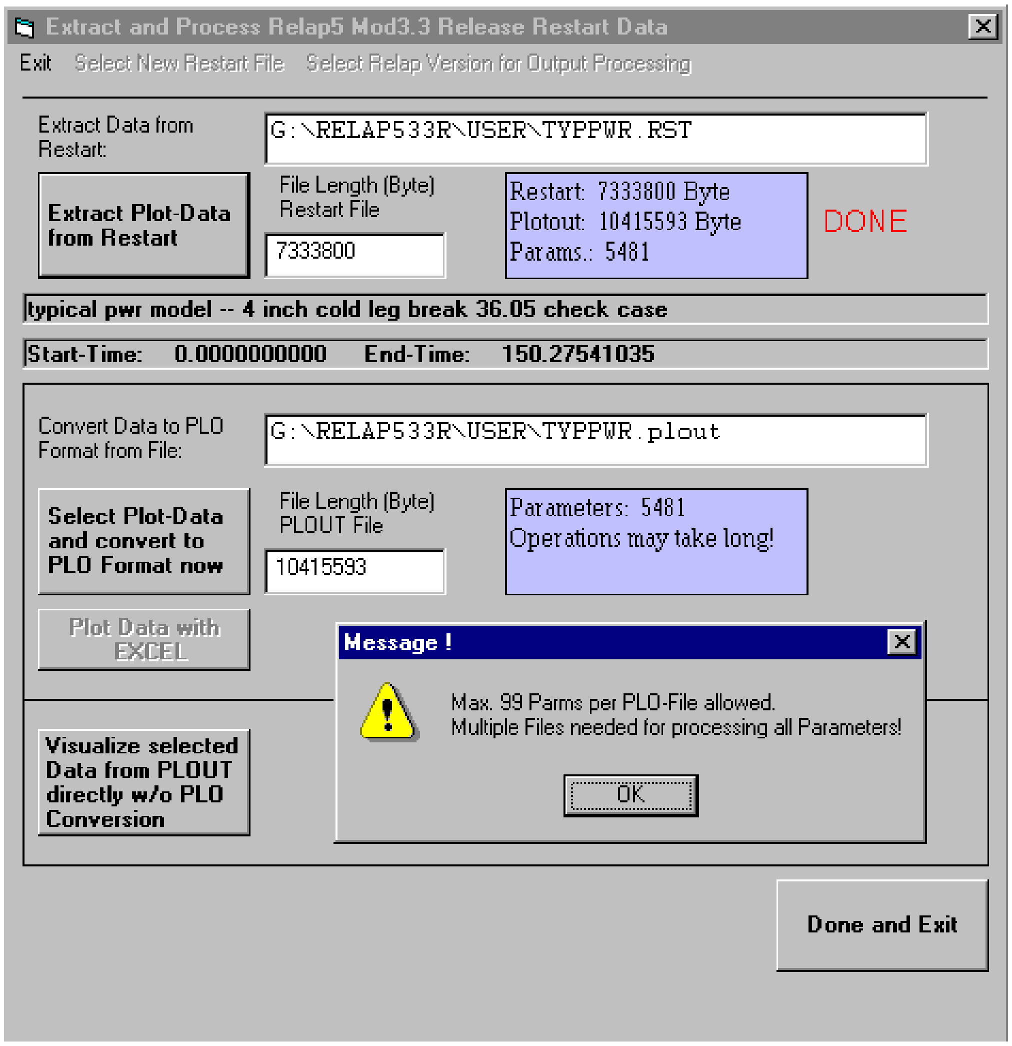
typical pwr model -- 4 inch cold leg break 36.05 check case

Start-Time: 0.0000000000 End-Time: 150.27541035

File Length (Byte) PLOUT File

Figure A-7: Extract and process Restart Plot Data

(RSINF file exist)


Extract and Process Relap5 Mod3.3 Release Restart Data
Exit
Select New Restart File
Select Relap Version for Output Processing
Extract Data from Restart:
Extract Plot-Data from Restart
File Length (Byte) Restart File
Convert Data to PLO Format from File:
Select Plot-Data and convert to PLO Format now
File Length (Byte) PLOUT File
Plot Data with EXCEL
Visualize selected Data from PLOUT directly w/o PLO Conversion
Message !
OK
Done and Exit

G:\RELAP533R\USER\TYPPWR.RST

7333800

Restart: 7333800 Byte
Plotout: 10415593 Byte
Params.: 5481

DONE

typical pwr model -- 4 inch cold leg break 36.05 check case

Start-Time: 0.0000000000 End-Time: 150.27541035

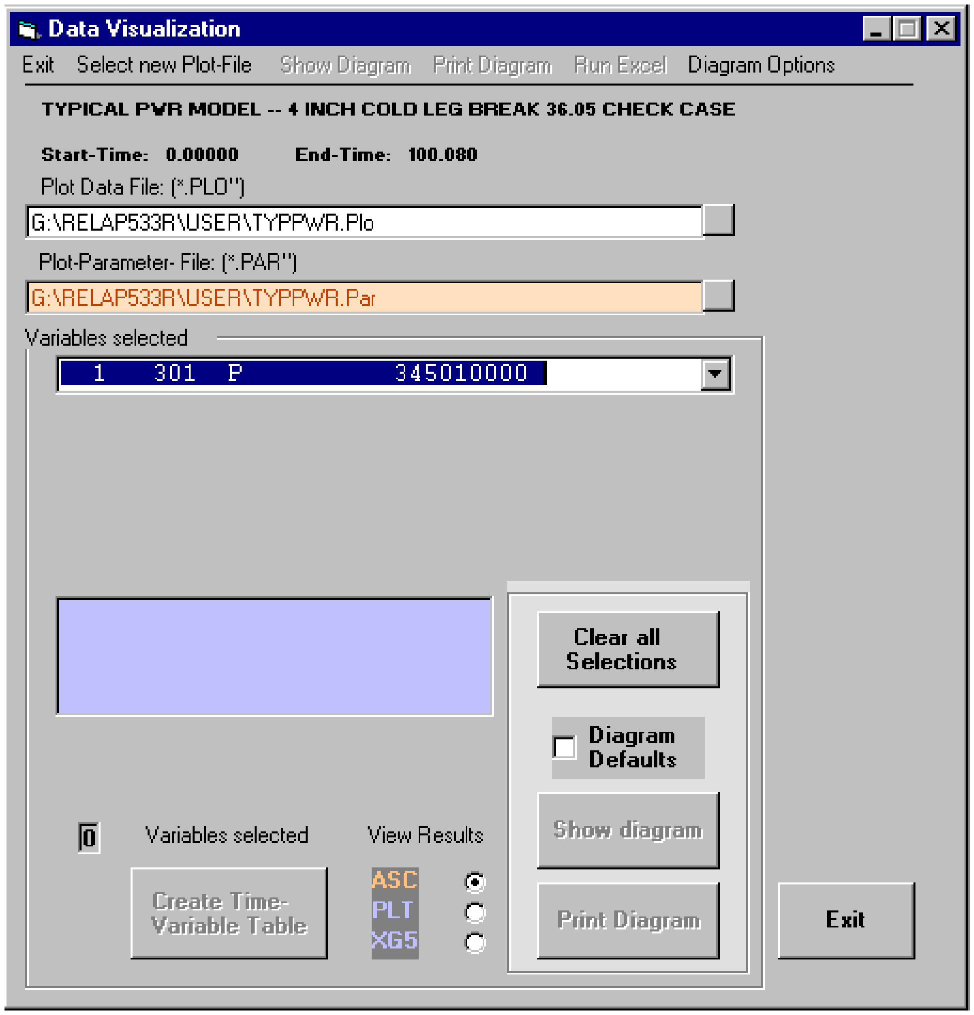
G:\RELAP533R\USER\TYPPWR.plout

10415593

Parameters: 5481
Operations may take long!

Max. 99 Parms per PLO-File allowed.
Multiple Files needed for processing all Parameters!

Figure A-8: Successful Extraction of Restart Plot Data



TYPICAL PWR MODEL -- 4 INCH COLD LEG BREAK 36.05 CHECK CASE

Start-Time: 0.00000 End-Time: 100.080

Plot Data File: (*.PLO")

G:\RELAP533R\USER\TYPPWR.Plo

Plot-Parameter- File: (*.PAR")

G:\RELAP533R\USER\TYPPWR.Par

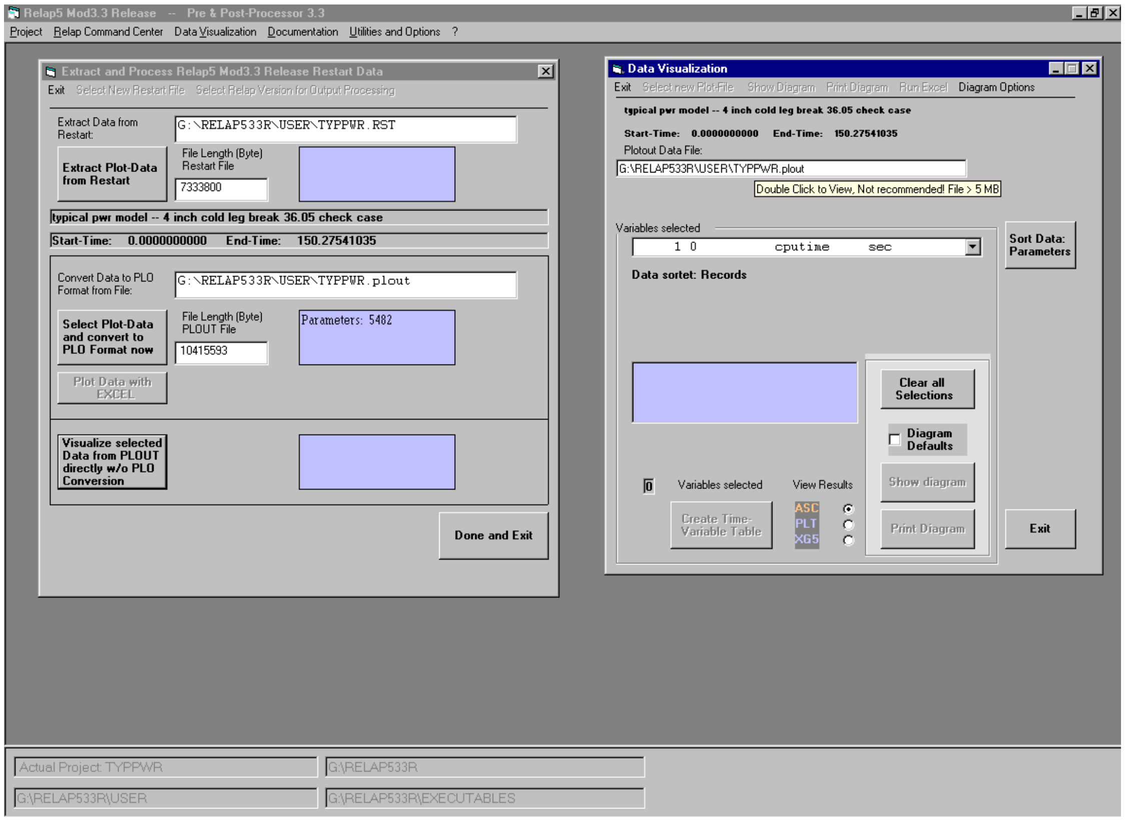
Variables selected

```
  1    301   P          345010000
```

0 Variables selected

View Results: [✓ ASC] PLT XG5

Figure A-9: Plot Post processor: Data Visualization (PLO Channel)


Relap5 Mod3.3 Release -- Pre & Post-Processor 3.3
Project Relap Command Center Data Visualization Documentation Utilities and Options ?
Extract and Process Relap5 Mod3.3 Release Restart Data
Exit Select New Restart File Select Relap Version for Output Processing
Extract Data from Restart:
Extract Plot-Data from Restart
File Length (Byte) Restart File
Convert Data to PLO Format from File:
Select Plot-Data and convert to PLO Format now
File Length (Byte) PLOUT File
Plot Data with EXCEL
Visualize selected Data from PLOUT directly w/o PLO Conversion
Done and Exit
Data Visualization
Exit Select new Plot-File Show Diagram Print Diagram Run Excel Diagram Options
Plotout Data File:
Double Click to View, Not recommended! File > 5 MB
Variables selected
Sort Data: Parameters
Data sortet: Records
Clear all Selections
Diagram Defaults
Show diagram
Variables selected
View Results
Create Time-Variable Table
ASC
PLT
XG5
Print Diagram
Exit

G:\RELAP533R\USER\TYPPWR.RST

7333800

typical pwr model -- 4 inch cold leg break 36.05 check case

Start-Time: 0.0000000000 End-Time: 150.27541035

G:\RELAP533R\USER\TYPPWR.plout

10415593

Parameters: 5482

typical pwr model -- 4 inch cold leg break 36.05 check case

Start-Time: 0.0000000000 End-Time: 150.27541035

G:\RELAP533R\USER\TYPPWR.plout

1 0 cputime sec

0

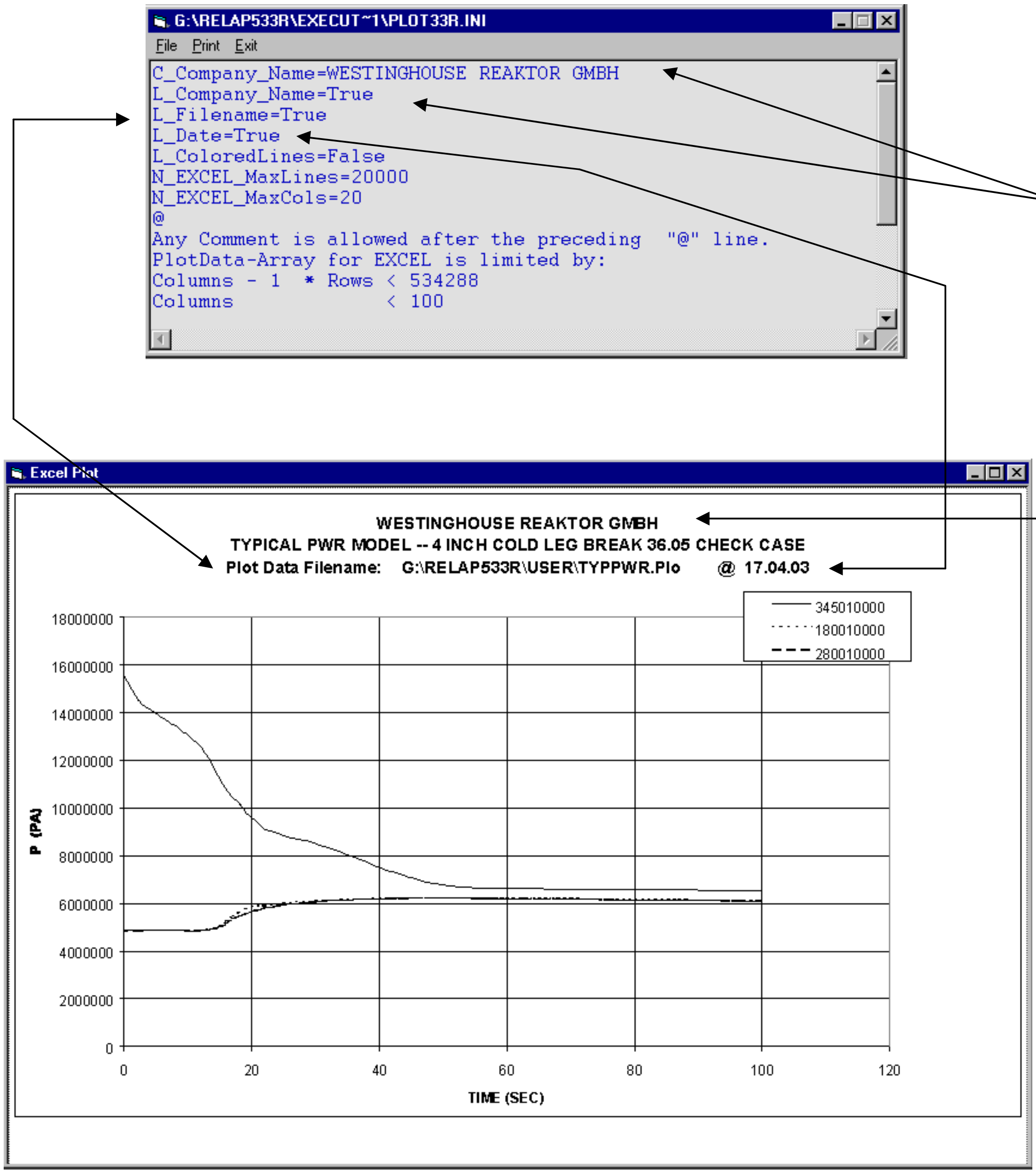
Actual Project: TYPPWR

G:\RELAP533R

G:\RELAP533R\USER

G:\RELAP533R\EXECUTABLES

Figure A-10: Plot Post processor: PLOUT Plot Data processing



```
C_Company_Name=WESTINGHOUSE REAKTOR GMBH
L_Company_Name=True
L_Filename=True
L_Date=True
L_ColoredLines=False
N_EXCEL_MaxLines=20000
N_EXCEL_MaxCols=20
@
Any Comment is allowed after the preceding  "@" line.
PlotData-Array for EXCEL is limited by:
Columns - 1  * Rows < 534288
Columns             < 100
```

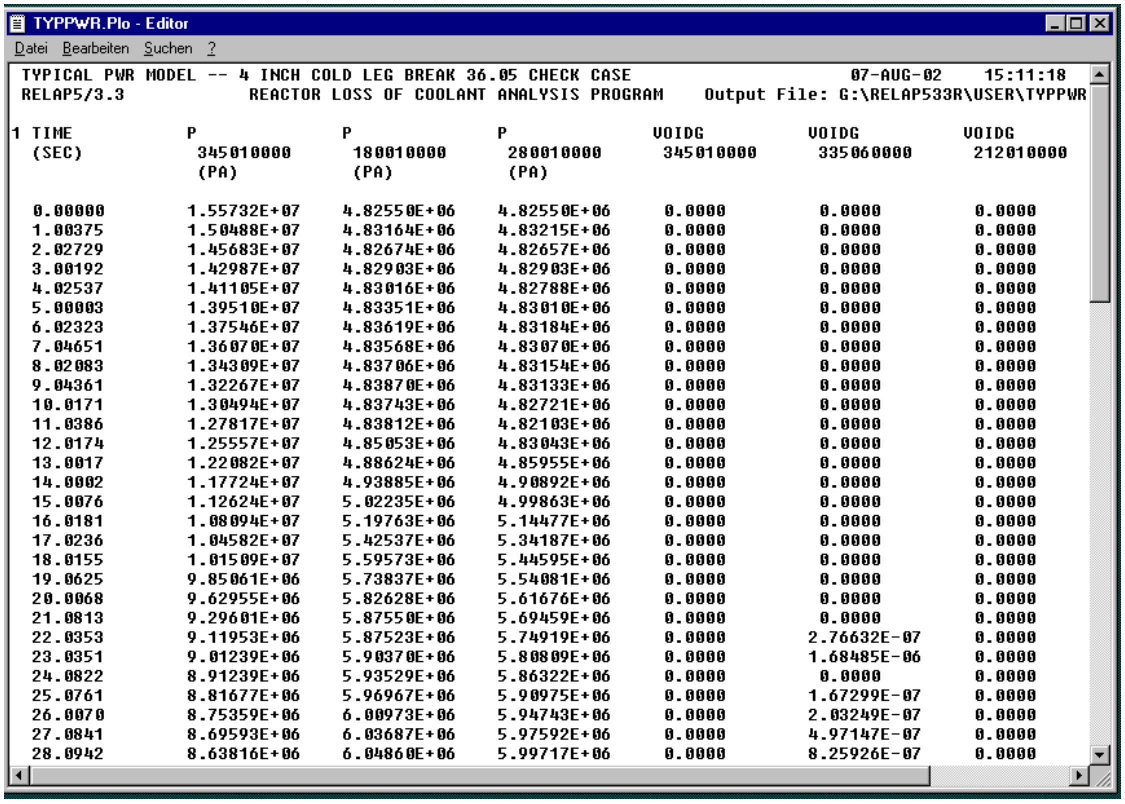
WESTINGHOUSE REAKTOR GMBH
TYPICAL PWR MODEL -- 4 INCH COLD LEG BREAK 36.05 CHECK CASE
Plot Data Filename: G:\RELAP533R\USER\TYPPWR.Plo @ 17.04.03

Figure A-11: Plotting Graphs: Styles affected by PLOT33R.INI

Annex B-1

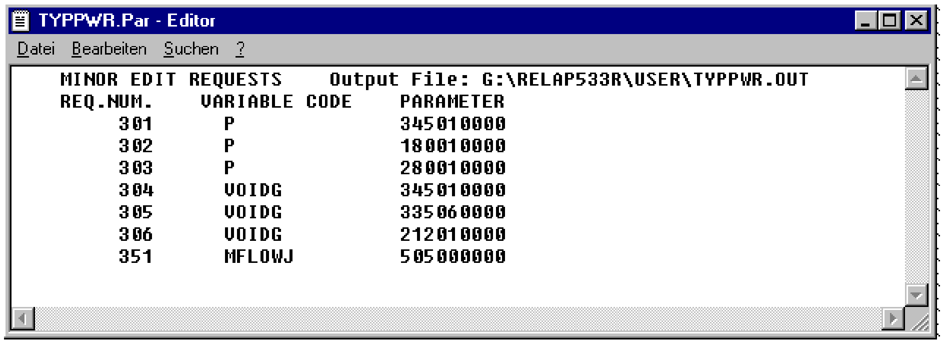
Tables

Table B-1: PLO Plotdata Format and typical PAR file


TYPPWR.Plo - Editor

Datei Bearbeiten Suchen ?

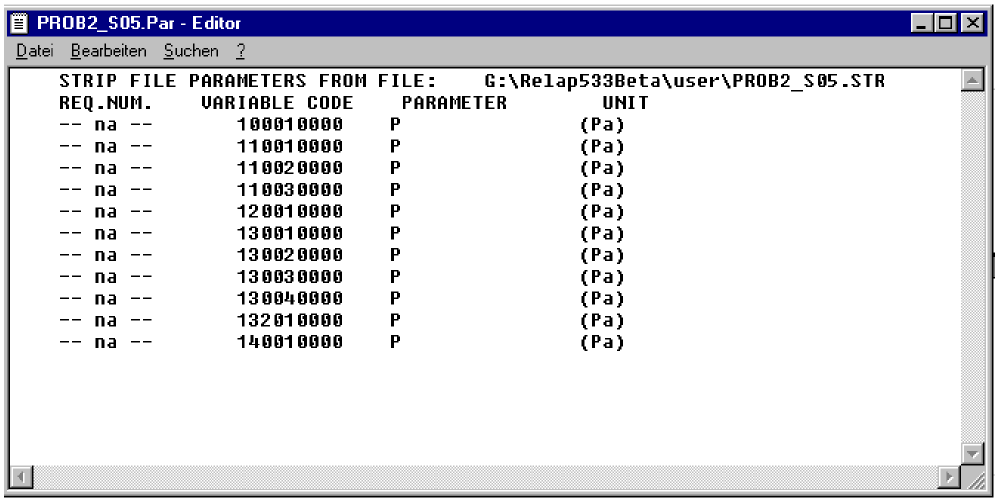
```
 TYPICAL PWR MODEL -- 4 INCH COLD LEG BREAK 36.05 CHECK CASE                 07-AUG-02    15:11:18
 RELAP5/3.3           REACTOR LOSS OF COOLANT ANALYSIS PROGRAM    Output File: G:\RELAP533R\USER\TYPPWR

1 TIME              P              P              P              VOIDG          VOIDG          VOIDG
  (SEC)              345010000      180010000      280010000      345010000      335060000      212010000
                     (PA)           (PA)           (PA)

  0.00000           1.55732E+07    4.82550E+06    4.82550E+06    0.0000         0.0000         0.0000
  1.00375           1.50488E+07    4.83164E+06    4.83215E+06    0.0000         0.0000         0.0000
  2.02729           1.45683E+07    4.82674E+06    4.82657E+06    0.0000         0.0000         0.0000
  3.00192           1.42987E+07    4.82903E+06    4.82903E+06    0.0000         0.0000         0.0000
  4.02537           1.41105E+07    4.83016E+06    4.82788E+06    0.0000         0.0000         0.0000
  5.00003           1.39510E+07    4.83351E+06    4.83010E+06    0.0000         0.0000         0.0000
  6.02323           1.37546E+07    4.83619E+06    4.83184E+06    0.0000         0.0000         0.0000
  7.04651           1.36070E+07    4.83568E+06    4.83070E+06    0.0000         0.0000         0.0000
  8.02083           1.34309E+07    4.83706E+06    4.83154E+06    0.0000         0.0000         0.0000
  9.04361           1.32267E+07    4.83870E+06    4.83133E+06    0.0000         0.0000         0.0000
  10.0171           1.30494E+07    4.83743E+06    4.82721E+06    0.0000         0.0000         0.0000
  11.0386           1.27817E+07    4.83812E+06    4.82103E+06    0.0000         0.0000         0.0000
  12.0174           1.25557E+07    4.85053E+06    4.83043E+06    0.0000         0.0000         0.0000
  13.0017           1.22082E+07    4.88624E+06    4.85955E+06    0.0000         0.0000         0.0000
  14.0002           1.17724E+07    4.93885E+06    4.90892E+06    0.0000         0.0000         0.0000
  15.0076           1.12624E+07    5.02235E+06    4.99863E+06    0.0000         0.0000         0.0000
  16.0181           1.08094E+07    5.19763E+06    5.14477E+06    0.0000         0.0000         0.0000
  17.0236           1.04582E+07    5.42537E+06    5.34187E+06    0.0000         0.0000         0.0000
  18.0155           1.01509E+07    5.59573E+06    5.44595E+06    0.0000         0.0000         0.0000
  19.0625           9.85061E+06    5.73837E+06    5.54081E+06    0.0000         0.0000         0.0000
  20.0068           9.62955E+06    5.82628E+06    5.61676E+06    0.0000         0.0000         0.0000
  21.0813           9.29601E+06    5.87550E+06    5.69459E+06    0.0000         0.0000         0.0000
  22.0353           9.11953E+06    5.87523E+06    5.74919E+06    0.0000         2.76632E-07    0.0000
  23.0351           9.01239E+06    5.90370E+06    5.80809E+06    0.0000         1.68485E-06    0.0000
  24.0822           8.91239E+06    5.93529E+06    5.86322E+06    0.0000          0.0000        0.0000
  25.0761           8.81677E+06    5.96967E+06    5.90975E+06    0.0000         1.67299E-07    0.0000
  26.0070           8.75359E+06    6.00973E+06    5.94743E+06    0.0000         2.03249E-07    0.0000
  27.0841           8.69593E+06    6.03687E+06    5.97592E+06    0.0000         4.97147E-07    0.0000
  28.0942           8.63816E+06    6.04860E+06    5.99717E+06    0.0000         8.25926E-07    0.0000
```


TYPPWR.Par - Editor

Datei Bearbeiten Suchen ?

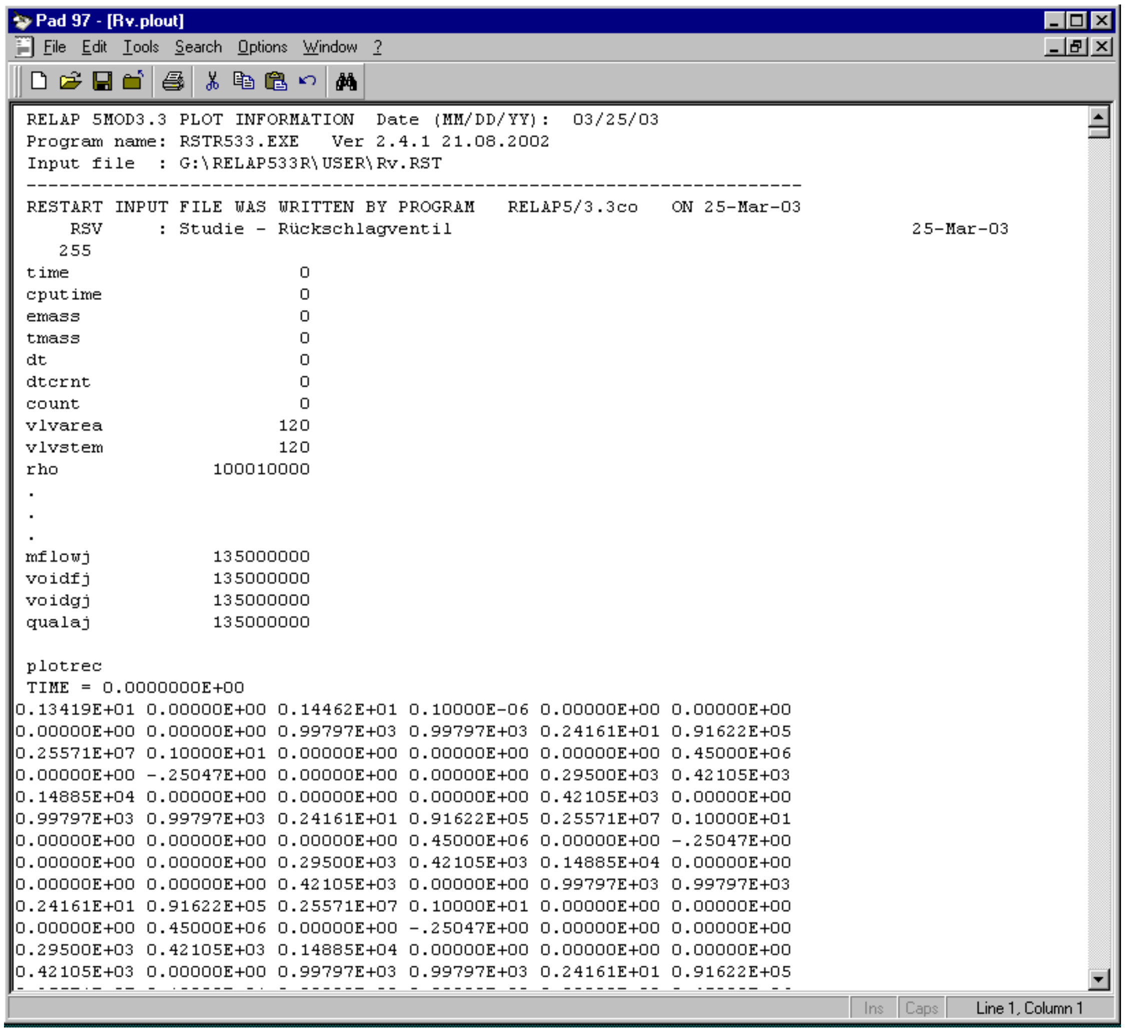
```
    MINOR EDIT REQUESTS    Output File: G:\RELAP533R\USER\TYPPWR.OUT
    REQ.NUM.     VARIABLE CODE      PARAMETER
         301       P                345010000
         302       P                180010000
         303       P                280010000
         304       VOIDG            345010000
         305       VOIDG            335060000
         306       VOIDG            212010000
         351       MFLOWJ           505000000
```

Table B-2: Parameter File for Strip Data



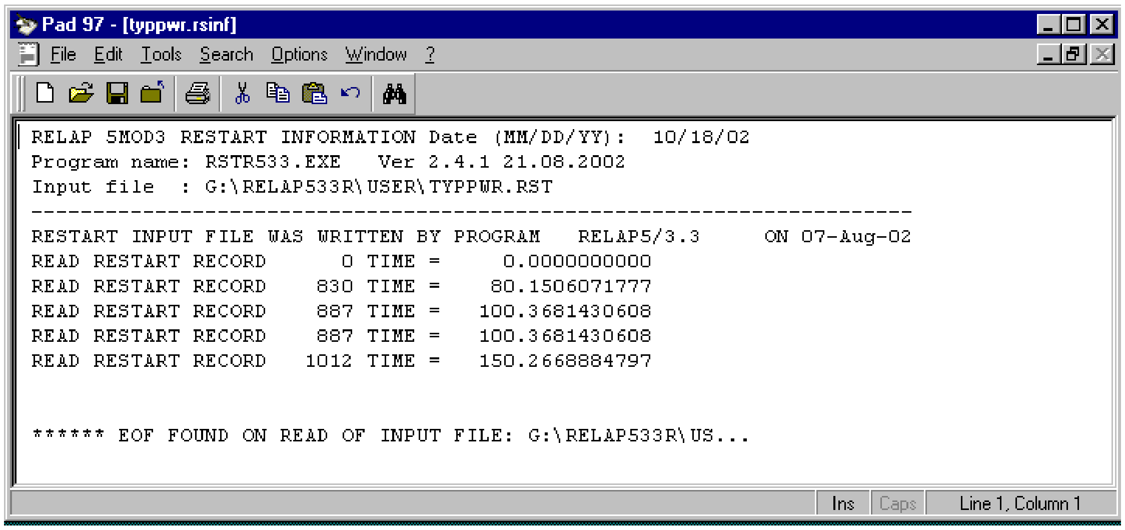
```
    STRIP FILE PARAMETERS FROM FILE:    G:\Relap533Beta\user\PROB2_S05.STR
    REQ.NUM.    VARIABLE CODE    PARAMETER         UNIT
    -- na --       100010000    P                (Pa)
    -- na --       110010000    P                (Pa)
    -- na --       110020000    P                (Pa)
    -- na --       110030000    P                (Pa)
    -- na --       120010000    P                (Pa)
    -- na --       130010000    P                (Pa)
    -- na --       130020000    P                (Pa)
    -- na --       130030000    P                (Pa)
    -- na --       130040000    P                (Pa)
    -- na --       132010000    P                (Pa)
    -- na --       140010000    P                (Pa)
```

Table B-3: PLOUT Data Format

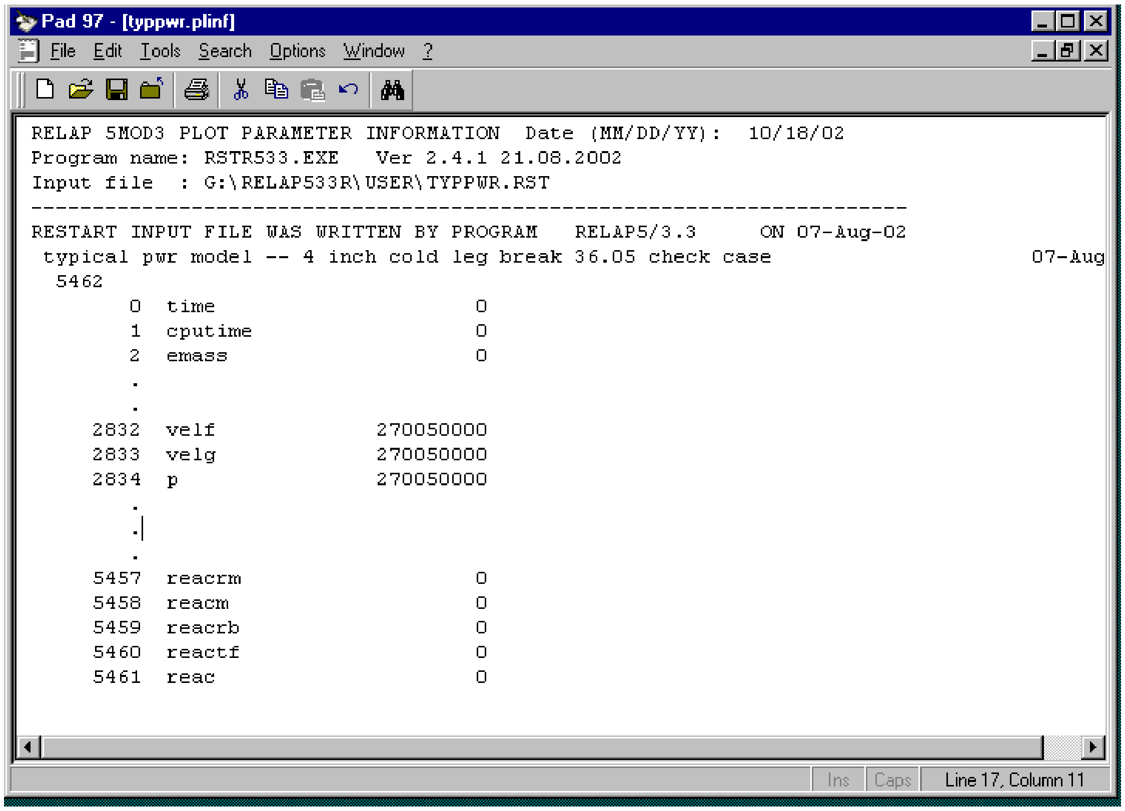

Pad 97 - [Rv.plout]
File Edit Tools Search Options Window ?
Ins Caps Line 1, Column 1

```
RELAP 5MOD3.3 PLOT INFORMATION  Date (MM/DD/YY):  03/25/03
Program name: RSTR533.EXE   Ver 2.4.1 21.08.2002
Input file  : G:\RELAP533R\USER\Rv.RST
----------------------------------------------------------------------
RESTART INPUT FILE WAS WRITTEN BY PROGRAM   RELAP5/3.3co   ON 25-Mar-03
    RSV     : Studie - Rückschlagventil                                   25-Mar-03
   255
time                     0
cputime                  0
emass                    0
tmass                    0
dt                       0
dtcrnt                   0
count                    0
vlvarea                120
vlvstem                120
rho              100010000
.
.
.
mflowj           135000000
voidfj           135000000
voidgj           135000000
qualaj           135000000

plotrec
TIME = 0.0000000E+00
0.13419E+01 0.00000E+00 0.14462E+01 0.10000E-06 0.00000E+00 0.00000E+00
0.00000E+00 0.00000E+00 0.99797E+03 0.99797E+03 0.24161E+01 0.91622E+05
0.25571E+07 0.10000E+01 0.00000E+00 0.00000E+00 0.00000E+00 0.45000E+06
0.00000E+00 -.25047E+00 0.00000E+00 0.00000E+00 0.29500E+03 0.42105E+03
0.14885E+04 0.00000E+00 0.00000E+00 0.00000E+00 0.42105E+03 0.00000E+00
0.99797E+03 0.99797E+03 0.24161E+01 0.91622E+05 0.25571E+07 0.10000E+01
0.00000E+00 0.00000E+00 0.00000E+00 0.45000E+06 0.00000E+00 -.25047E+00
0.00000E+00 0.00000E+00 0.29500E+03 0.42105E+03 0.14885E+04 0.00000E+00
0.00000E+00 0.00000E+00 0.42105E+03 0.00000E+00 0.99797E+03 0.99797E+03
0.24161E+01 0.91622E+05 0.25571E+07 0.10000E+01 0.00000E+00 0.00000E+00
0.00000E+00 0.45000E+06 0.00000E+00 -.25047E+00 0.00000E+00 0.00000E+00
0.29500E+03 0.42105E+03 0.14885E+04 0.00000E+00 0.00000E+00 0.00000E+00
0.42105E+03 0.00000E+00 0.99797E+03 0.99797E+03 0.24161E+01 0.91622E+05
```

Table B-4: Restart & Plot Information Files

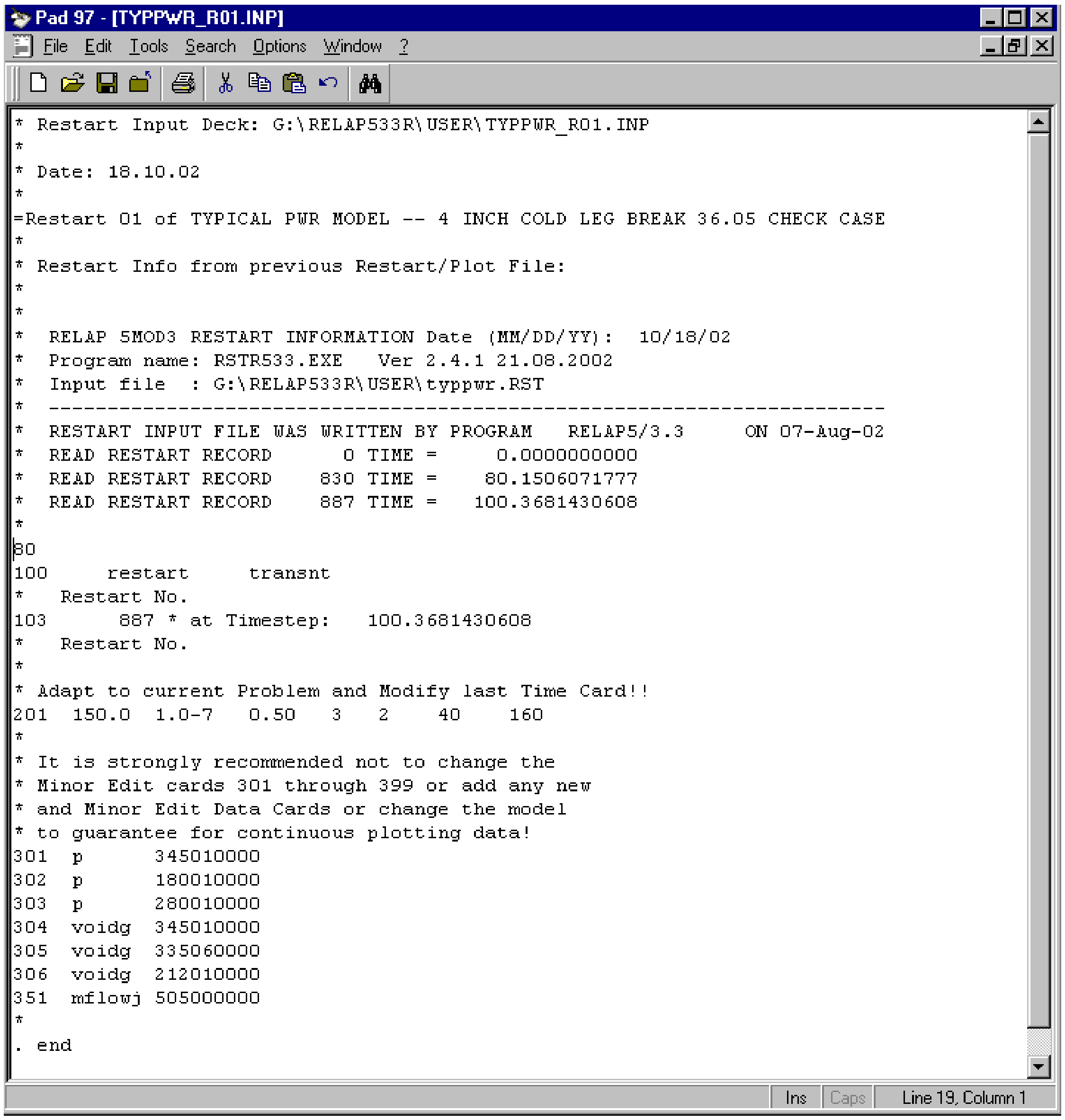

Pad 97 - [typpwr.rsinf]
File Edit Tools Search Options Window ?
Ins Caps Line 1, Column 1

```
RELAP 5MOD3 RESTART INFORMATION Date (MM/DD/YY):  10/18/02
Program name: RSTR533.EXE   Ver 2.4.1 21.08.2002
Input file  : G:\RELAP533R\USER\TYPPWR.RST
------------------------------------------------------------------------
RESTART INPUT FILE WAS WRITTEN BY PROGRAM   RELAP5/3.3     ON 07-Aug-02
READ RESTART RECORD      0 TIME =     0.0000000000
READ RESTART RECORD    830 TIME =    80.1506071777
READ RESTART RECORD    887 TIME =   100.3681430608
READ RESTART RECORD    887 TIME =   100.3681430608
READ RESTART RECORD   1012 TIME =   150.2668884797


****** EOF FOUND ON READ OF INPUT FILE: G:\RELAP533R\US...
```


Pad 97 - [typpwr.plinf]
File Edit Tools Search Options Window ?
Ins Caps Line 17, Column 11

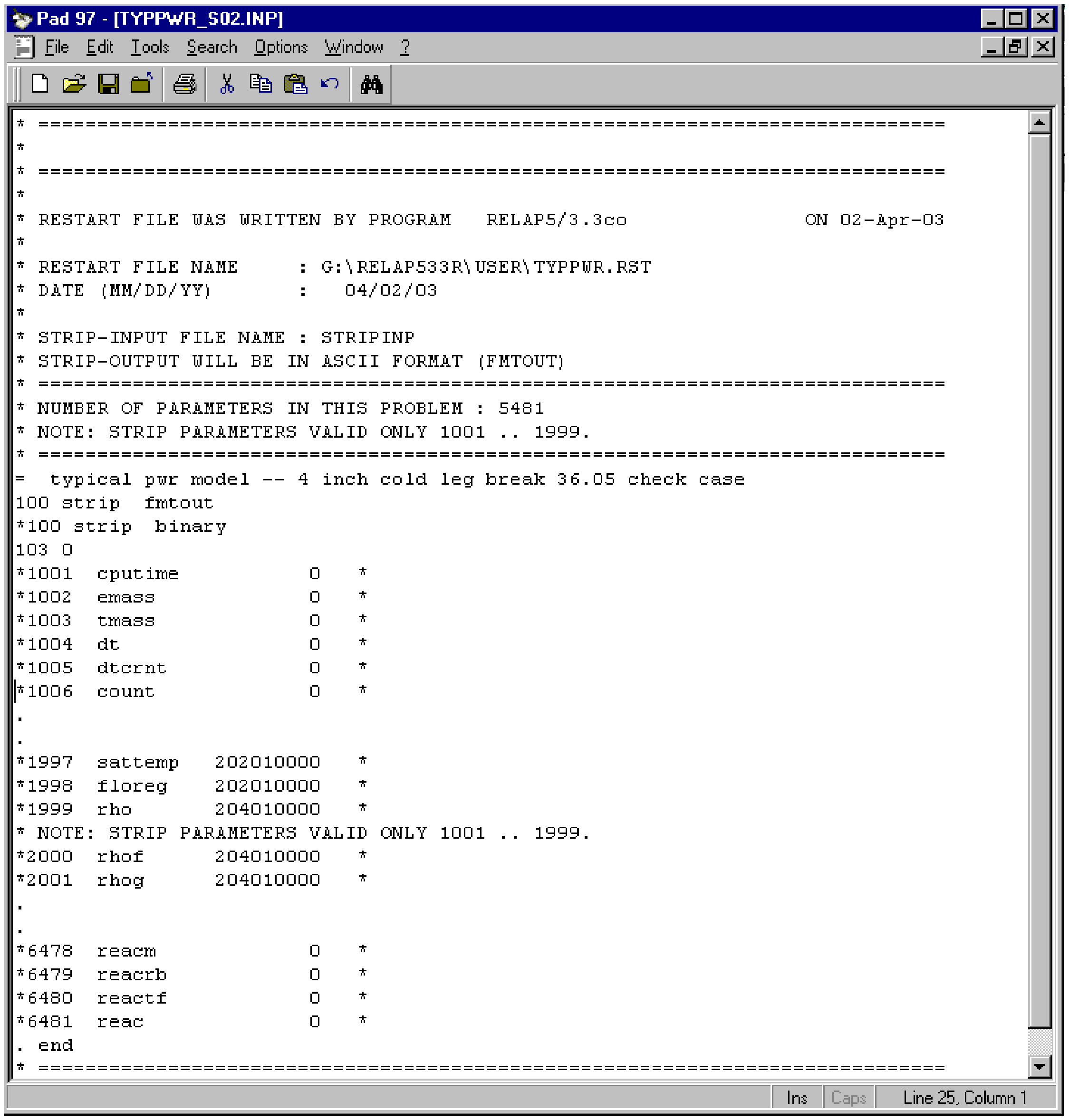
```
RELAP 5MOD3 PLOT PARAMETER INFORMATION  Date (MM/DD/YY):  10/18/02
Program name: RSTR533.EXE   Ver 2.4.1 21.08.2002
Input file  : G:\RELAP533R\USER\TYPPWR.RST
------------------------------------------------------------------------
RESTART INPUT FILE WAS WRITTEN BY PROGRAM   RELAP5/3.3     ON 07-Aug-02
 typical pwr model -- 4 inch cold leg break 36.05 check case                     07-Aug
  5462
        0  time                     0
        1  cputime                  0
        2  emass                    0
        .
        .
     2832  velf             270050000
     2833  velg             270050000
     2834  p                270050000
        .
        .
        .
     5457  reacrm                   0
     5458  reacm                    0
     5459  reacrb                   0
     5460  reactf                   0
     5461  reac                     0
```

Table B-5: Restart Input File


Pad 97 - [TYPPWR_R01.INP]
File Edit Tools Search Options Window ?
Ins Caps Line 19, Column 1

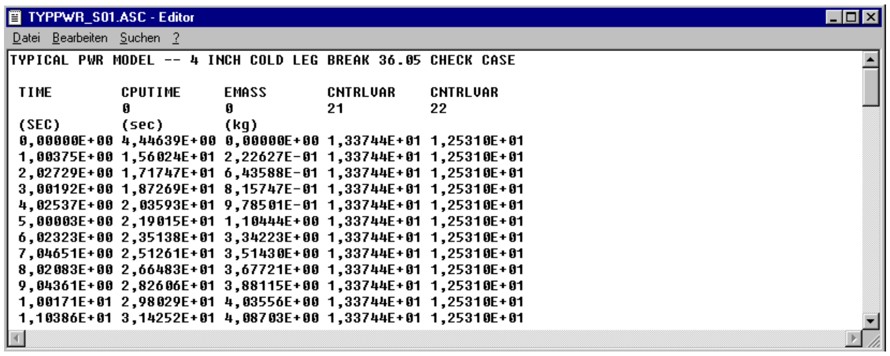
```
* Restart Input Deck: G:\RELAP533R\USER\TYPPWR_R01.INP
*
* Date: 18.10.02
*
=Restart 01 of TYPICAL PWR MODEL -- 4 INCH COLD LEG BREAK 36.05 CHECK CASE
*
* Restart Info from previous Restart/Plot File:
*
*
*   RELAP 5MOD3 RESTART INFORMATION Date (MM/DD/YY):  10/18/02
*   Program name: RSTR533.EXE   Ver 2.4.1 21.08.2002
*   Input file  : G:\RELAP533R\USER\typpwr.RST
*   ----------------------------------------------------------------------
*   RESTART INPUT FILE WAS WRITTEN BY PROGRAM   RELAP5/3.3     ON 07-Aug-02
*   READ RESTART RECORD      0 TIME =     0.0000000000
*   READ RESTART RECORD    830 TIME =    80.1506071777
*   READ RESTART RECORD    887 TIME =   100.3681430608
*
80
100      restart      transnt
*   Restart No.
103       887 * at Timestep:   100.3681430608
*   Restart No.
*
* Adapt to current Problem and Modify last Time Card!!
201   150.0   1.0-7    0.50    3    2    40     160
*
* It is strongly recommended not to change the
* Minor Edit cards 301 through 399 or add any new
* and Minor Edit Data Cards or change the model
* to guarantee for continuous plotting data!
301   p      345010000
302   p      180010000
303   p      280010000
304   voidg  345010000
305   voidg  335060000
306   voidg  212010000
351   mflowj 505000000
*
. end
```

Table B-6: Strip Input File

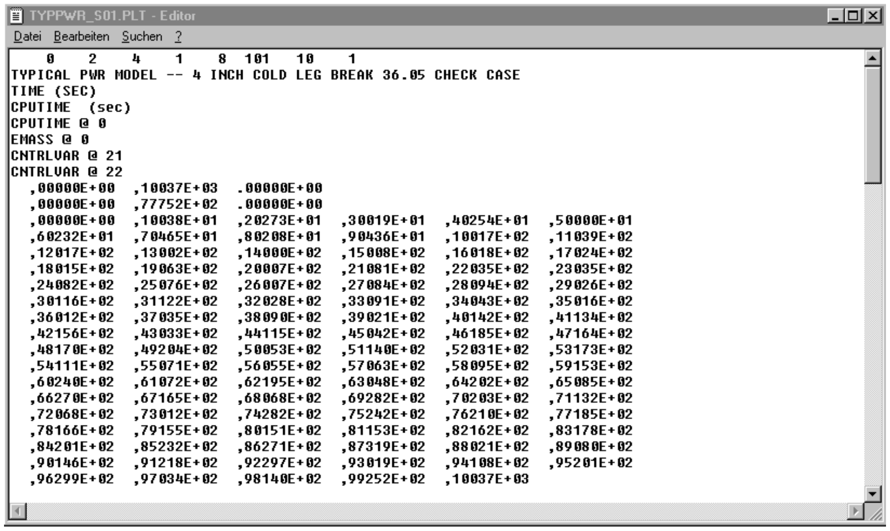

Pad 97 - [TYPPWR_S02.INP]

```
* ============================================================================
*
* ============================================================================
*
* RESTART FILE WAS WRITTEN BY PROGRAM   RELAP5/3.3co              ON 02-Apr-03
*
* RESTART FILE NAME     : G:\RELAP533R\USER\TYPPWR.RST
* DATE (MM/DD/YY)       :   04/02/03
*
* STRIP-INPUT FILE NAME : STRIPINP
* STRIP-OUTPUT WILL BE IN ASCII FORMAT (FMTOUT)
* ============================================================================
* NUMBER OF PARAMETERS IN THIS PROBLEM : 5481
* NOTE: STRIP PARAMETERS VALID ONLY 1001 .. 1999.
* ============================================================================
=  typical pwr model -- 4 inch cold leg break 36.05 check case
100 strip  fmtout
*100 strip  binary
103 0
*1001  cputime           0    *
*1002  emass             0    *
*1003  tmass             0    *
*1004  dt                0    *
*1005  dtcrnt            0    *
*1006  count             0    *
.
.
*1997  sattemp   202010000    *
*1998  floreg    202010000    *
*1999  rho       204010000    *
* NOTE: STRIP PARAMETERS VALID ONLY 1001 .. 1999.
*2000  rhof      204010000    *
*2001  rhog      204010000    *
.
.
*6478  reacm             0    *
*6479  reacrb            0    *
*6480  reactf            0    *
*6481  reac              0    *
. end
* ============================================================================
```

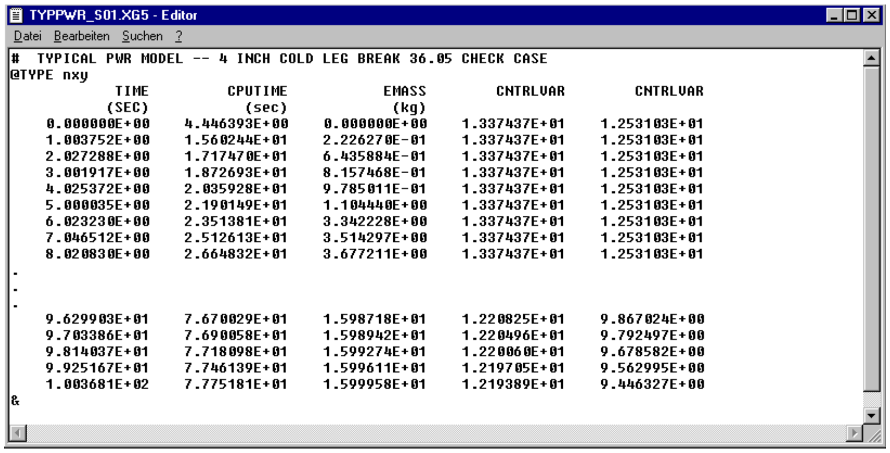
Table B-7: Storing Plot Data in three different Formats



```
TYPICAL PWR MODEL -- 4 INCH COLD LEG BREAK 36.05 CHECK CASE

 TIME        CPUTIME     EMASS       CNTRLVAR    CNTRLVAR
             0           0           21          22
 (SEC)       (sec)       (kg)
 0,00000E+00 4,44639E+00 0,00000E+00 1,33744E+01 1,25310E+01
 1,00375E+00 1,56024E+01 2,22627E-01 1,33744E+01 1,25310E+01
 2,02729E+00 1,71747E+01 6,43588E-01 1,33744E+01 1,25310E+01
 3,00192E+00 1,87269E+01 8,15747E-01 1,33744E+01 1,25310E+01
 4,02537E+00 2,03593E+01 9,78501E-01 1,33744E+01 1,25310E+01
 5,00003E+00 2,19015E+01 1,10444E+00 1,33744E+01 1,25310E+01
 6,02323E+00 2,35138E+01 3,34223E+00 1,33744E+01 1,25310E+01
 7,04651E+00 2,51261E+01 3,51430E+00 1,33744E+01 1,25310E+01
 8,02083E+00 2,66483E+01 3,67721E+00 1,33744E+01 1,25310E+01
 9,04361E+00 2,82606E+01 3,88115E+00 1,33744E+01 1,25310E+01
 1,00171E+01 2,98029E+01 4,03556E+00 1,33744E+01 1,25310E+01
 1,10386E+01 3,14252E+01 4,08703E+00 1,33744E+01 1,25310E+01
```

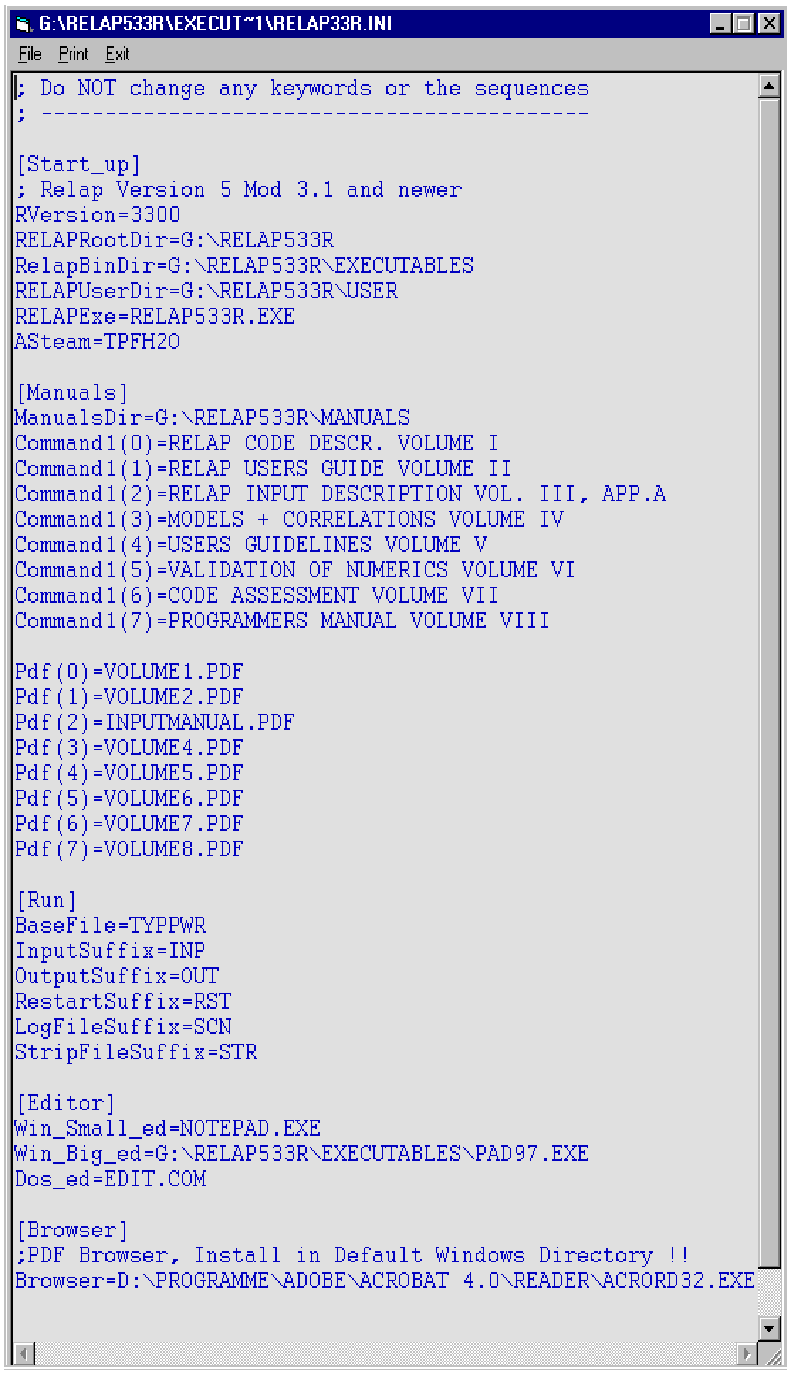


```
    0    2    4    1    8  101   10    1
TYPICAL PWR MODEL -- 4 INCH COLD LEG BREAK 36.05 CHECK CASE
TIME (SEC)
CPUTIME  (sec)
CPUTIME @ 0
EMASS @ 0
CNTRLVAR @ 21
CNTRLVAR @ 22
  ,00000E+00  ,10037E+03  .00000E+00
  ,00000E+00  ,77752E+02  .00000E+00
  ,00000E+00  ,10038E+01  ,20273E+01  ,30019E+01  ,40254E+01  ,50000E+01
  ,60232E+01  ,70465E+01  ,80208E+01  ,90436E+01  ,10017E+02  ,11039E+02
  ,12017E+02  ,13002E+02  ,14000E+02  ,15008E+02  ,16018E+02  ,17024E+02
  ,18015E+02  ,19063E+02  ,20007E+02  ,21081E+02  ,22035E+02  ,23035E+02
  ,24082E+02  ,25076E+02  ,26007E+02  ,27084E+02  ,28094E+02  ,29026E+02
  ,30116E+02  ,31122E+02  ,32028E+02  ,33091E+02  ,34043E+02  ,35016E+02
  ,36012E+02  ,37035E+02  ,38090E+02  ,39021E+02  ,40142E+02  ,41134E+02
  ,42156E+02  ,43033E+02  ,44115E+02  ,45042E+02  ,46185E+02  ,47164E+02
  ,48170E+02  ,49204E+02  ,50053E+02  ,51140E+02  ,52031E+02  ,53173E+02
  ,54111E+02  ,55071E+02  ,56055E+02  ,57063E+02  ,58095E+02  ,59153E+02
  ,60240E+02  ,61072E+02  ,62195E+02  ,63048E+02  ,64202E+02  ,65085E+02
  ,66270E+02  ,67165E+02  ,68068E+02  ,69282E+02  ,70203E+02  ,71132E+02
  ,72068E+02  ,73012E+02  ,74282E+02  ,75242E+02  ,76210E+02  ,77185E+02
  ,78166E+02  ,79155E+02  ,80151E+02  ,81153E+02  ,82162E+02  ,83178E+02
  ,84201E+02  ,85232E+02  ,86271E+02  ,87319E+02  ,88021E+02  ,89080E+02
  ,90146E+02  ,91218E+02  ,92297E+02  ,93019E+02  ,94108E+02  ,95201E+02
  ,96299E+02  ,97034E+02  ,98140E+02  ,99252E+02  ,10037E+03
```



```
#  TYPICAL PWR MODEL -- 4 INCH COLD LEG BREAK 36.05 CHECK CASE
@TYPE nxy
            TIME         CPUTIME           EMASS        CNTRLVAR        CNTRLVAR
           (SEC)           (sec)            (kg)
    0.000000E+00    4.446393E+00    0.000000E+00    1.337437E+01    1.253103E+01
    1.003752E+00    1.560244E+01    2.226270E-01    1.337437E+01    1.253103E+01
    2.027288E+00    1.717470E+01    6.435884E-01    1.337437E+01    1.253103E+01
    3.001917E+00    1.872693E+01    8.157468E-01    1.337437E+01    1.253103E+01
    4.025372E+00    2.035928E+01    9.785011E-01    1.337437E+01    1.253103E+01
    5.000035E+00    2.190149E+01    1.104440E+00    1.337437E+01    1.253103E+01
    6.023230E+00    2.351381E+01    3.342228E+00    1.337437E+01    1.253103E+01
    7.046512E+00    2.512613E+01    3.514297E+00    1.337437E+01    1.253103E+01
    8.020830E+00    2.664832E+01    3.677211E+00    1.337437E+01    1.253103E+01
.
.
.
    9.629903E+01    7.670029E+01    1.598718E+01    1.220825E+01    9.867024E+00
    9.703386E+01    7.690058E+01    1.598942E+01    1.220496E+01    9.792497E+00
    9.814037E+01    7.718098E+01    1.599274E+01    1.220060E+01    9.678582E+00
    9.925167E+01    7.746139E+01    1.599611E+01    1.219705E+01    9.562995E+00
    1.003681E+02    7.775181E+01    1.599958E+01    1.219389E+01    9.446327E+00
&
```

Table B-8: Relap-INI File



```
; Do NOT change any keywords or the sequences
; -------------------------------------------

[Start_up]
; Relap Version 5 Mod 3.1 and newer
RVersion=3300
RELAPRootDir=G:\RELAP533R
RelapBinDir=G:\RELAP533R\EXECUTABLES
RELAPUserDir=G:\RELAP533R\USER
RELAPExe=RELAP533R.EXE
ASteam=TPFH2O

[Manuals]
ManualsDir=G:\RELAP533R\MANUALS
Command1(0)=RELAP CODE DESCR. VOLUME I
Command1(1)=RELAP USERS GUIDE VOLUME II
Command1(2)=RELAP INPUT DESCRIPTION VOL. III, APP.A
Command1(3)=MODELS + CORRELATIONS VOLUME IV
Command1(4)=USERS GUIDELINES VOLUME V
Command1(5)=VALIDATION OF NUMERICS VOLUME VI
Command1(6)=CODE ASSESSMENT VOLUME VII
Command1(7)=PROGRAMMERS MANUAL VOLUME VIII

Pdf(0)=VOLUME1.PDF
Pdf(1)=VOLUME2.PDF
Pdf(2)=INPUTMANUAL.PDF
Pdf(3)=VOLUME4.PDF
Pdf(4)=VOLUME5.PDF
Pdf(5)=VOLUME6.PDF
Pdf(6)=VOLUME7.PDF
Pdf(7)=VOLUME8.PDF

[Run]
BaseFile=TYPPWR
InputSuffix=INP
OutputSuffix=OUT
RestartSuffix=RST
LogFileSuffix=SCN
StripFileSuffix=STR

[Editor]
Win_Small_ed=NOTEPAD.EXE
Win_Big_ed=G:\RELAP533R\EXECUTABLES\PAD97.EXE
Dos_ed=EDIT.COM

[Browser]
;PDF Browser, Install in Default Windows Directory !!
Browser=D:\PROGRAMME\ADOBE\ACROBAT 4.0\READER\ACRORD32.EXE
```

Table B-9: UNITS-INI File

```
; Do NOT change the Width of any Column. Data @ line 5. Last line: @
; Paramter - Units - 29.04.2001     Line Length: 76 + . (77)
; --------------------------------------------------------------------------.
ppppppppppppsisisisisisibubububububusssssssssssssssssssssbbbbbbbbbbbbbbbbbbbb.
time        sec         sec         Zeit                Time                .
count                               Rechenschritt       Count Advancement   .
cputime     sec         sec         CPU-Zeit            CPU Time            .
emass       kg          lb          Fehler-Masse        Error Mass          .
tmass       kg          lb          Totale Masse        Total Mass          .
dt          sec         sec         Zeitschritt         Time Step           .
dtcrnt      sec         sec         Courant Zeitschritt Courant Time Step   .

acqtank     W           btu/s       Accu.Tot.Energ.TransAccuTotEnergieTransp.
acrhon      kg/m^3      lb/ft^3     Accu.NonCondensDichtAccuNonCond.Density .
acttank     K           °F          Accu.mit.Wand.Temp. Accu.av.Wall Temp.  .
acvdm       m^3         ft^3        Accu.Gas Volume     Accu.Gas Volume     .
acvliq      m^3         ft^3        Accu.Liqu.Volume    Accu.Liqu. Volume   .

pmpvel      U/min       rad/s       Pumpen Drehzahl     Pump Rot.Velocity   .
pmphead     Pa          lbf/in^2    Pumpenhöhe          Pumphead            .
pmptrq      N*m         lbf*ft      Pumpen Torque       Pump Torque         .

turef                               Effektivität TurbineTurbine Efficiency  .
turpow      W           Btu/s       Turbinen Leistung   Turbine Power       .
turtrq      N*m         lbf*ft      Turbinen Torque     Turbine Torque      .
turvel      U/min       rad/s       Turbinen Drehzahl   Turbine Rot.Velocity.

vlvarea                             Ventil Fläche       Valve Area Ratio    .
vlvstem                             Ventil Öffnung      Valve stem position .

boron       kg/m^3      lb/ft^3     BOR Dichte          Boron Density       .
floreg                              FLOREG              Flow regime number  .
p           Pa          lbf/in      Druck               Volume Pressure     .
q           W           Btu/s       Leistung im Volumen Tot.vol.heat source .
quala                               NonCond.Mass.Verh.  Vol.noncond.massfrac.
quale                               Equ.Qualität        Vol.equil.quality   .
quals                               Statische Qualität  Vol.static.quality  .
qwg         W           Btu/s       Volumen Leistung    Vol.wal heat source .
rho         kg/m^3      lb/ft^3     Dichte              Total density       .
rhof        kg/m^3      lb/ft^3     Flui Dichte         Liquid density      .
rhog        kg/m^3      lb/ft^3     Dampf Dichte        Vapor density       .
sattemp     K           °F          SättigungstemperaturVolume sat.temp.    .
sounde      m/s         ft/s        Schallgeschwind.    Volume son.velocity .
tempf       K           °F          Fluid Temperatur    Volume liqu.temp.   .
tempg       K           °F          Dampf Temperatur    Volume vap.temp.    .
uf          J/kg        Btu/lb      Vol. Liqu.Spec.Ener Volume liq.energy   .
ug          J/kg        Btu/lb      Vol. Gas.Spec.Ener  Volume vap.energy   .
vapgen      kg/m^3*s    lb/ft^3*s   Verdampf/Conden.    Vap.gen./Cond. rate .
velf        m/s         ft/s        Vol.Fluid Geschw.   Volume Liq.velocity .
velg        m/s         ft/s        Vol.Dampf Geschw.   Volume Vap.velocity .
voidf                               Volume Void (fluid) Vol. liquid fraction.
voidg                               Volume Void (gas)   Vol. gas fraction   .
```

Table B-10: UNITS-INI File, Con't

```
mflowj      kg/s        lb/s        Fluid&Gas Durchsatz Liq. & Vap.flow rate.
qualaj                              Junc. NonCond.Qual  Jun.noncond.mass.fra.
rhofj       kg/m^3      lb/ft^3     Junc. Fluid.Dichte  Jun.Liqu.density    .
rhogj       kg/m^3      lb/ft^3     Junc. Dampf.Dichte  Jun.Gas.density     .
sonicj      m/s         ft/s        Junc. Schallgeschw  Jun. sound speed    .
ufj         J/kg        Btu/lb      Junc. Liqu.Spec.EnerJun. liqu.spec.ener..
ugj         J/kg        Btu/lb      Junc. Gas.Spec.Ener Jun. vap.spec.energy.
velfj       m/s         ft/s        Junc. Liqu.Geschw.  Jun. liqu.velocity  .
velgj       m/s         ft/s        Junc. Gas Geschw.   Jun. Vap.velocity   .
voidfj                              Junc. Liqu. FraktionJun. liquid fraction.
voidgj                              Junc. Gas Fraktion  Jun. vapor fraction .

htchf       W/m^2       Btu/s·ft^2  CHF                 Critical heat flux  .
hthtc       W/m^2·K     Btu/s-ft^2°FHeat transfer coeff.Heat transfer coeff .
htrnr       W/m^2       Btu/s·ft^2  Wärmefluss          Heat flux           .
httemp      K           °F          Mesh Point Temp.    Mesh point temp.    .

reac        $           $           Totale Reaktivität  Tot.Reactiv.feedback.
reacm       $           $           Tot.Moderator Rückk.Tot.React.feedb.mod..
reacrb      $           $           Bor Rückkopplung    React.feedb. boron  .
reacrm      $           $           Moderator Rückk.    Reac.feedb.moderator.
reacs       $           $           Abschaltraktivität  ScramReactivity     .
reactf      $           $           Brennstoff.Rückkop. TempReactivity feedb.
reactm      $           $           Mod.Temp Rückkop.   Reac.feedb.mod.temp..
rkfipow     W           W           Reaktor Leistung    Reactor pow. fission.
rkgapow     W           W           Nachzerfallsleist.  Reactor pow. decay  .
rkpowa      W           W           Nazef Actiniden     React.pow.decay.acti.
rkpowk      W           W           Nazef Fission Prod. React.pow.decay.fiss.
rkreac      $           $           Reaktivität         Reactivity          .
rkrecper    s^-1        s^-1        ReaaktorPeriode     Reciprocal period   .
rktpow      W           W           Totale ReaktorLeist.Total reactor power .
@
```

Annex C-1

Source Code Listing of the Relap5MOD3.3R Restart Postprocessor

```
C     Last change:  DWT  20 Sep 2002    1:03 pm
C
C
C RSTR533B.FOR ======================================================
C     Purpose:
C             Extract PlotData from Restart/Plot-File
C             Create Inormation files about Plots & Restarts
C             Create Information about Plotparameters for Strip files
C             Create automatic Strip Input for RELAP 5/MOD3.2.2Gamma
C                                      and RELAP 5/MOD3.3Beta
C                                      and RELAP 5/MOD3.3 Release   RSTR33.R
C                    Options: -s and -so for stripfile
C ==================================================================
C
C  Development:  Dr. Tietsch
C  Based on RESTI120
C  Created:      02.07.1994
C  Version:      1.10 fuer IBM-RISK/486-PC
C  Version:      1.20 fuer IBM-RISK/486-PC (06.10.94)
C  Version:      1.21 fuer IBM-RISK/486-PC Lahey Fortran 90 Vers. 40a   restlh9
C  Version:      1.22 fuer IBM-RISK/486-PC Lahey Fortran 95 Vers. 5.5   resti95
C  Version:      1.30 fuer IBM-RISK                                     restaix
C                               read IBM-Risc6000 Restart Data of      restaix
C                               relap5 Mod 1 ABB version               restaix
C  Version:      2.0    fuer Relap5 3.2.2.Gamma INTEL Version          RSTR532
C  Version:      2.1    fuer Relap5 3.2.2.Gamma AIX   Version          RSTR32a
C  Version:      2.2    fuer Relap5 3.2.2.Gamma INTEL/AIX   Version    RSTR32u
C  Version:      2.2.1  fuer Relap5 3.2.2.Gamma INTEL/AIX   Version    RSTR32u1
C  Version:      2.2.2  fuer Relap5 3.2.2.Gamma ma..z Version for      RSTR32u2
C                INTEL/AIX/DECALPHA/hpc180/sgi64/sunOS56 platforms     RSTR32u2
C  Version:      2.2.3  fuer Relap5 3.2.2.Gamma ma..z Version          RSTR32u3
C  Version:      2.2.4  fuer Relap5 3.2.2.Gamma ma..z Version          RSTR32u4
C                       Process Restartfiles from Restarts also        RSTR32u4
C  Version:      2.3    fuer Relap5 3.3 Beta pt Version                RSTR33
C                       Contrl + 4 real variables                      RSTR33
C  Version:      2.3.0  fuer Relap5 3.3 Beta pt Version                RSTR33.0
C  Version:      2.3.1  fuer Relap5 3.3 Beta pt Version mit Änderungen  RSTR33.1
C                                          fuer grosse Files           RSTR33.1
C  Version:      2.4.0  fuer Relap5 3.3 Release Version                RSTR33.R
C  Version:      2.4.1  fuer Relap5 3.3 Release Version mit Änderungen  RSTR33R1
C                                          fuer grosse Files           RSTR33R1
C
C  Last Update:  27.09.1994 , 30.11.2000 (1.21)                        restlh9
C  Last Update:               07.12.2000 (1.30)                        restaix
C  Last Update:               17.01.2001 (2.00)                        RSTR532
C  Last Update:               17.01.2001 (2.10)                        RSTR32a
C  Last Update:               18.01.2001 (2.20)                        RSTR32u
C  Last Update:               19.01.2001 (2.21)                        RSTR32u1
C  Last Update:               04.03.2001 (2.22)                        RSTR32u2
C  Last Update:               27.04.2001 (2.23)                        RSTR32u3
C  Last Update:               12.06.2001 (2.24)                        RSTR32u4
C  Last Update:               23.09.2001 (2.3)                         RSTR33
C  Last Update:               07.10.2001 (2.3.0)                       RSTR33.0
C  Last Update:               21.08.2002 (2.3.1)                       RSTR33.1
C  Last Update:               08.08.2002 (2.4.0)                       RSTR33.R
C  Last Update:               20.09.2002 (2.4.1)                       RSTR33R1
C
C ===================================================================
C  Options:                                  Defaults:
C          -ri   restart/strip_input_file             RESTINP
C          -ro   restart_output w/o plotdata          RESTOUT
C          -or   restart_info_file                    RESTINF
C          -po   plot_parm & data_file                PLOTOUT
C          -op   plot_parameter_info_file             PLOTINF
C          -so   strip inputfile for RELAP            STRIPINP
C          -d    debug data in file                   DEBUG
C          -r    write restart output file
C          -s    write strip input file
C          -p    write plotdata file
C          -no   no screen output
C          -h/?  HELP
C
C     w/o -p and/or -r the default is:
C                only RESTINF and/or PLOTINF files will be written
C     For AIX/DEC/SUN and SGI Restarts RESTOUT will not be processed!'  RSTR32u2
C ===================================================================
C  Technology reading restart files with PC written with AIX :          RESTaix
C         (HPC180,SGI64 and SunOS56 are equivalent to AIX)              RSTR32u2
C         (DECalpha in equivalent to Intel)                             RSTR32u2
C     AIX writes Bits of Integer*2,4, Real4, Real8 in opposite order    RESTaix
C     than Intel based compilers, so they must be swept. Example:       RESTaix
C                                                                       RESTaix
C                          I1int(1)=I1aix(4)                            RESTaix
C                          I1int(2)=I1aix(3)                            RESTaix
```

```
C                        I1int(3)=I1aix(2)                            RESTaix
C                        I1int(4)=I1aix(1)                            RESTaix
C                                                                     RESTaix
C     new function:       intel-integer*4=iflip(aix-integer*4)        RESTaix
C                                                                     RESTaix
C     Data written with AIX (and s.o.)operation system must be read by  RSTR32u2
C     INTEL programs in 4 Byte pieces. Bits must be flipped.          RESTaix
C     e.g. R8 aix writes   8 Bytes as follows:                        RESTaix
C                                                                     RESTaix
C                  Bytes:     8 7 6 5    4 3 2 1                      RESTaix
C                             -------    -------                      RESTaix
C     equivalence:            I4aix(1)   I4aix(2)                     RESTaix
C                                                                     RESTaix
C     flip Bits with Iflip    5 6 7 8    1 2 3 4                      RESTaix
C                             I4int(2)   I4int(1)                     RESTaix
C     change order            1 2 3 4    5 6 7 8                      RESTaix
C                             -------    -------                      RESTaix
C     equivalence:            I4aix(1)   I4aix(2)                     RESTaix
C                                                                     RESTaix
C     Result:                      R8int                              RESTaix
C                                                                     RESTaix
C      REAL*8    R8aix,R8int                                          RESTaix
C      INTEGER*4 I4aix(2), I4int(2)                                   RESTaix
C      EQUIVALENCE (I4aix(1), R8aix)                                  RESTaix
C      EQUIVALENCE (I4int(1), R8int)                                  RESTaix
C                                                                     RESTaix
C                                                                     RESTaix
C      AIX data must be read in by INTEL in 4BYTE chunks in transparent RESTaix
C      access mode.                                                   RESTaix
C                                                                     RESTaix
C      First Byte of INTEL restart plot file is an 8, AIX is .ne. 8   RESTaix
C                                                                     RESTaix
C      Method to read in the binary data stream from aix restart file   RESTaix
C      is to first read the pointer as 2 Real*4 numbers, which are    RESTaix
C      equivalenced to I*4, than reading the data defined b the pointer RESTaix
C      with one read statement, then reading the next pointer and so on.RESTaix
C                                                                     RESTaix
C      Meaning of the I*4 Pointers:                                   RESTaix
C                                                                     RESTaix
C      Pointer(1)= Number Bytes back to preceding Pointer             RESTaix
C      Pointer(2)= Number Bytes to next Pointer                       RESTaix
C                                                                     RESTaix
C      Example: 112 80 :                                              RESTaix
C                     112 Byte = 14 Real*8 back                       RESTaix
C                      80 Byte = 10 Real*8 to next Pointer            RESTaix
C      Note: Relap writes additional Pointers as well (lx, iwrd)      RESTaix
C            which are needed for the determination of the length of  RESTaix
C            the data blocks                                          RESTaix
C ======================================================================
C  Technology:
C       RELAP5 MOd 3.2 and up Restart consits of the following parts:
C                                              Type Wordlengt Length
C       I  RSTPLT Record: Program Identifications
C
C          1)    Program Identification            a      8       2
C                Example: '  RELAP5/3.2mz  '
C          2)    Empty A8 '        '               a      8       1
C
C          3)    Characterstring                   a      8       3
C                Example: 'restart-plot file       '
C          4)    Date/Time file was written        a      8       3
C                Example  '13-DEC-00    18:55:29   '
C          5)    Two Intgers, 2nd Variable irout   i      4       2
C
C       II  PLOTINF Record: 'plotinf'
C
C          1)    Length of PLOTALF record (lenb)   r      8       1
C                Example: 0, 379
C          2)    Length of PLOTREC record (lenc)   r      8       1
C                Example: 0, 190
C                if lenc=(lenb+1)/2 then plot data is compressed
C                using the sqoz subroutine (two r4 values are
C                packed int one r8)
C
C       III PLOTALF Record: 'plotalf'
C
C          1)    alphanumeric part of the variable a      8       lenb
C                Example: 'time    '
C
C       IV  PLOTNUM Record: 'plotnum'
C
C          1)    numeric part of the variable      r      8       lenb
C                Example: 0.000000
C
C       V   RESTART Record: 'restart'
```

```
C
C          1)    Print variable iprint             r      8        1
C                two integers, second: iprint
C                Example: 0, 23
C          2)    no of restart record ncount       r      8        1
C                two integers, second: ncount
C                Example: 0, 0
C     --->  these are written only for restart No. 0
C           and for every restartet problem at time of restart
C          3)    Title record, ptitle string       r      8        8
C          4)    Title record, ctitle string       r      8       12
C          5)    Named common block comctl         r      8       lx
C     --->  these are written for every restart data block
C          6)    Named common block contrl         r      8       lx
C                Time of restart:  timehy R8 on pos 5 in contrl block
C                Number restartblock :  rstblcknumber1531 I4 on pos 47 / 1
C                Restart number      : ncount             I4 on pos 48 / 2
C          7)    Other named common blocks and     r      4/8      lx
C                dynamic commonblocks
C
C       VI  PLOTREC Record: 'plotrec'
C
C          1)    numeric value of variable         r      8        lenc
C                Example: 0.000000
C                two R4 values packed into one R8, when compressed
C                use unsqoz to uncompress or even better in case
C                of INTEL read in continuously as R4. Data is in
C                right order
C
C
C       The principal access to the data is achieved by consideration
C       of the data organzation in RSTPLT files as follows:
C           - Two Integers (4) must be read before any data
C             LEN or lz     : defines the length of the data record
C             iwrd8 or iwrd : defines the length of word to be read
C                             example 8  ---> read in as A8 or R8
C                                     4  ---> read in as Integer or
C                                             two of them as R8 or A8
C          - Depending of these informations the following data block
C             has to be read in in fa array as R8 or in any other array
C
C             EXAMPLE:  read(rstplt-file) LEN,iwrd  --> 50, 8
C                       j = LEN*iwrd/8
C                       read(rstplt-file) (fa(i-1),i=1,j)
C
C          - If one of the pltrst records follows, this is indicated
C             by one of the keywords 'plotinf'
C                                    'plotalf'
C                                    'plotnum'
C                                    'plotrec'
C                                    'restart'
C
C                                              on position i (=1) in fa
C          - The restart record consists in principal of 5 blocks of
C             data, three of them are written at each restart block,
C             two (the title record and the comctl common data block)
C             are written only at time 0
C
C ====================================================================
      PROGRAM RSTR533                                                   RESTR33.R
      IMPLICIT REAL*8 (A-H,O-Z)
C
*comdeck fast
C
C      PARAMETER (LFSIZ=2**17-1)                                         RESTI120
C      PARAMETER (LFSIZ=100000)
       parameter (lfsiz=2200000)                                        RSTR5322
       common /fast/ fa(lfsiz)
       real*8 fa
       integer ia(2,lfsiz)
       equivalence (fa(1),ia(1,1))
       REAL*4 FFA (LFSIZ*2)                                             RSTR53u
       INTEGER iia (LFSIZ*2)                                            RSTR53u
       EQUIVALENCE (FFA(1),fa(1))                                       RSTR53u
       EQUIVALENCE (FFA(1),iia(1))                                      RSTR53u
C
*comdeck fastc
C
c  arrays define dynamic pool area.  fa and ia are equivalent floating
c  and integer areas.
c
*comdeck comctl
c
       integer ncoms,nfiles
       parameter (ncoms=104, nfiles=50)
```

```
c
       common /comctl/ comdat(ncoms),comdln(ncoms),filid(nfiles) ,
     & filsiz(nfiles),filndx(0:nfiles)                          ,
     & writedynamicfile0(ncoms+1)                               ,
     & writecommonblck2(ncoms+1)                                ,
     & is4(ncoms+1)                                             ,
     & safe1
c
       real*8 filid,safe1
       integer comdat,comdln,filsiz,filndx
       logical newrst
c
       equivalence (safe1,newrst)
c
       logical writedynamicfile0,
     &         writecommonblck2 ,
     &         is4
c*******************************************************************
c Data dictionary for local variables
c i=integer r=real l=logical c=character
c*******************************************************************
c Type    Name                              Definition
c-------------------------------------------------------------------
c  i      comdat(ncoms)     = index relative to fa of first word of
c                              common block to be saved.
c  i      comdln(ncoms)     = length of common block to be saved.
c  r      filid(nfiles)     = ftb filid for dynamic storage files.
c         filid(1)  input data and work scratch space during advancement
c         filid(2)  time step control block.
c         filid(3)  component description block.
c         filid(4)  hydrodynamic volumes block.
c         filid(5)  hydrodynamic junctions block.
c         filid(6)  Thermodynamic property file.
c         filid(7)  interactive input and output variable storage.
c         filid(8)  heat structure geometry and temperature block.
c         filid(9)  heat structure material property storage.
c         filid(10) table of inlet and outlet junctions for each volume.
c         filid(11) general table storage.
c         filid(12) minor edit file.
c         filid(13) time dependent volumes and junctions pointers.
c         filid(14) table of heat structures and data for each volume.
c         filid(15) plot heading and control information.
c         filid(16) minor edit control, save area, and labels.
c         filid(17) plot record buffer.
c         filid(18) trip block.
c         filid(19) internal plot file control information.
c         filid(20) file for statistics during advancement.
c         filid(21) reactor kinetics data.
c         filid(22) 2d plot requests and specifications.
c         filid(23) plot comparison data tables.
c         filid(24) steady state block for statistics counters and uo.
c         filid(25) Volume data needed for a moving system.
c         filid(26) plot data for the internal plot routines.
c         filid(27) control system block.
c         filid(28) component indices in normal (input) order.
c         filid(29) Tabular data for rotations and translations for
c                    moving problem.
c         filid(30) hydrodynamic system control information.
c         filid(31) code coupling data
c         filid(32) reflood rezoning model storage space.
c         filid(33) user supplied plot record variable requests.
c         filid(34) fission product data.
c         filid(35) fixed list vectors.
c         filid(36) scdap data.
c         filid(37) steady state initialization check file.
c         filid(38) file for radiation heat transfer.
c         filid(40) file for sparse matrix strategy.
c         filid(43) file for level model geometry description
c                    ( i.e. stacks )
c  i      filndx(0:nfiles)  = index of dynamic file in fast or ftblcm
c                              block.
c  i      filsiz(nfiles)    = length of dynamic file.
c  l      is4               = tbd
c  i      ncoms             = number of common control slots.
c                              parameter = 104
c  l      newrst            = true if a restart problem, used during
c                              input processing.
c  i      nfiles            = number of dynamic file slots.
c                              parameter = 50
c  r      safe1             = not written on restart file, provided to
c                              allow length checking when reading from
c                              restart file.
c                             Used for timer argument when timing
c                              transient subroutine execution times.
c  l      writecommonblck2  = flag to write common block at complete
```

```
c                              restart.
c  l      writedynamicfile0 = flag for dynamic file on disk to be
c                              written at co restart.
c*********************************************************************
*comdeck comctlc
c
c  ncoms   number of common control slots.
c  nfiles  number of dynamic file slots.
c  comdat  index relative to fa of first word of common block to be
c          saved.
c  comdln  length of common block to be saved.
c  filid   ftb filid for dynamic storage files.
c  filsiz  length of dynamic file.
c  safe1   not  written on restart file, provided to allow length
c          checking when reading from restart file.  Used for
c          timer argument when timing transient subroutine
c          execution times.
c  newrst  true if a restart problem, used during input processing.
c  filndx  index of dynamic file in fast or ftblcm block.
c  filflg  flag for dynamic file on disk to be written at complete
c          restart (bit 1), flag to write common block file at complete
c          restart (bit 4).
c
c  filflg has been replaced with the following variables:
c l  (replaces bit  1)     writedynamicfile0    flag for dynamic file on
c                                               disk to be written at co
c                                               restart.
c l  (replaces bit  3)     writecommonblck2     flag to write common blo
c                                               at complete restart.
c
c l  (replaces bit  5)     is4                  tbd
c
c  filid(1)  input data and work scratch space during advancement.
c  filid(2)  time step control block.
c  filid(3)  component description block.
c  filid(4)  hydrodynamic volumes block.
c  filid(5)  hydrodynamic junctions block.
c  filid(6)  Thermodynamic property file.
c  filid(7)  interactive input and output variable storage.
c  filid(8)  heat structure geometry and temperature block.
c  filid(9)  heat structure material property storage.
c  filid(10) table of inlet and outlet junctions for each volume.
c  filid(11) general table storage.
c  filid(12) minor edit file.
c  filid(13) time dependent volumes and junctions pointers.
c  filid(14) table of heat structures and data for each volume.
c  filid(15) plot heading and control information.
c  filid(16) minor edit control, save area, and labels.
c  filid(17) plot record buffer.
c  filid(18) trip block.
c  filid(19) internal plot file control information.
c  filid(20) file for statistics during advancement.
c  filid(21) reactor kinetics data.
c  filid(22) 2d plot requests and specifications.
c  filid(23) plot comparison data tables.
c  filid(24) steady state block for statistics counters and uo.
c  filid(25) Volume data needed for a moving system.
c  filid(26) plot data for the internal plot routines.
c  filid(27) control system block.
c  filid(28) component indices in normal (input) order.
c  filid(29) Tabular data for rotations and translations for moving
c            problem.
c  filid(30) hydrodynamic system control information.
c  filid(31) code coupling data
c  filid(32) reflood rezoning model storage space.
c  filid(33) user supplied plot record variable requests.
c  filid(34) fission product data.
c  filid(35) fixed list vectors.
c  filid(36) scdap data.
c  filid(37) steady state initialization check file.
c  filid(38) file for radiation heat transfer.
c  filid(40) file for sparse matrix strategy.
c  filid(43) file for level model geometry description ( i.e. stacks )
c*********************************************************************
c
*comdeck contrl
C    Changes from versin 3.2.2Gamma to 3.3Beta
c
       common /contrl/ dthy,dtht,dtn,dt,timehy,timeht,errmax,tmass ,
C    & tmasso,emass,emasso,count,cpurem(5),stdtrn,gravcn,testda(20),    RESTR33
C      New in Relap5 3.3Beta     ----------------------------           RESTR33
     & tmasso,emass,emasso,count,curtmi,curtmj,curtrs,prevnd,           RESTR33
C      New in Relap5 3.3Beta     ----------------------------           RESTR33
     & cpurem(5),stdtrn,gravcn,testda(20),                              RESTR33
     & aflag,isrestartenabled0,ismajoreditenabled1,
     & isminoreditenabled2,isplotenabled3,iscompleterestart4,
     & interactiveflag5,isimplicithttrnsfr6,istwostepflag7,
     & stdystatetermflg8,integrationflag9,isathenaopt10,
     & isscdapopt11,isplotsqzflg12,rwtrstpltcmprssflag13,
     & transrotatflag14,rstblcknumber1531,succes,done,ncount,
     & nstsp,nrepet,help,nany,skipt,problemtype05,
     & problemopt611,ncase,fail,uniti,unito,chngno(90),
     & ishydrochng0,isheatcondchng1,isrknchng2,ispltrcrdchng3,
     & isradiationchng4,isfissionprdctchng5,iscntrlsyschng6,
     & pageno,isprntaccum0,isprntbrntrn1,
     & isprntccfl2,isprntchfcal3,isprntconden4,isprntdittus5,
     & isprnteqfinl6,isprntfwdrag7,isprntht2tdp9,isprnthtfinl12,
     & isprnthtrc113,isprnthydro15,isprntistate17,isprntjchoke18,
     & isprntjprop19,isprntnoncnd20,isprntphantj21,isprntphantv22,
     & isprntpimplt23,isprntpintfc24,isprntprednb25,isprntpreseq26,
     & isprntpstdnb27,isprntqfmove28,isprntsimplt29,isprntstacc0,
     & isprntstate1,isprntstatep2,isprntsuboil3,isprntsysitr4,
     & isprnttstate6,isprntvalve7,isprntvexplt8,isprntvfinl9,
     & isprntvimplt10,isprntvlvela11,isprntvolvel12,isprnttrip13,
     & isprntpower14,isprntvolume15,isprntjunction16,isprntheatstr17,
     & isprntradht18,isprntreflood19,isprntfsnprdtr20,isprntcontrol21,
     & isprntinput22,isprntmiedit23,islevelmodel0,
     & islevelcrossing1,issnapon,
     & rktpow3d,safe2
c
       real*8 dthy,dtht,dtn,dt,timehy,timeht,errmax,tmass,tmasso,emass,
C    & emasso,count,cpurem,stdtrn,gravcn,testda,rktpow3d,safe2          RESTR33
     & emasso,count,curtmi,curtmj,curtrs,prevnd,cpurem,stdtrn,gravcn,   RESTR33
     & testda,rktpow3d,safe2                                            RESTR33

c
       integer succes,done,ncount,nstsp,nrepet,help,nany,
     & ncase,pageno,cpurei(2,5)
c
       logical aflag,skipt,fail,uniti,unito,chngno,nmechk,issnapon
c
cblh replacement variables for iextra
       logical ishydrochng0       ,
     &         isheatcondchng1    ,
     &         isrknchng2         ,
     &         ispltrcrdchng3     ,
     &         isradiationchng4   ,
     &         isfissionprdctchng5,
     &         iscntrlsyschng6
c
cblh replacement variables for imdctl
       logical islevelmodel0       ,
     &         islevelcrossing1
c
cblh replacement variables for iroute
       integer problemtype05       ,
     &         problemopt611
c
cblh replacement variables for print
       logical
     &         isrestartenabled0    ,
     &         ismajoreditenabled1  ,
     &         isminoreditenabled2  ,
     &         isplotenabled3       ,
     &         iscompleterestart4   ,
     &         interactiveflag5     ,
     &         isimplicithttrnsfr6  ,
     &         istwostepflag7       ,
     &         stdystatetermflg8    ,
     &         integrationflag9     ,
     &         isathenaopt10        ,
     &         isscdapopt11         ,
     &         isplotsqzflg12       ,
     &         rwtrstpltcmprssflag13,
     &         transrotatflag14
c
cblh replacement variables for ihlppr
       logical
     &         isprntaccum0,
     &         isprntbrntrn1,
     &         isprntccfl2,
     &         isprntchfcal3,
     &         isprntconden4,
     &         isprntdittus5,
     &         isprnteqfinl6,
     &         isprntfwdrag7,
     &         isprntht2tdp9,
     &         isprnthtfinl12,
     &         isprnthtrc113,
     &         isprnthydro15,
     &         isprntistate17,
     &         isprntjchoke18,
     &         isprntjprop19
       logical
     &         isprntnoncnd20,
     &         isprntphantj21,
     &         isprntphantv22,
     &         isprntpimplt23,
     &         isprntpintfc24,
     &         isprntprednb25,
     &         isprntpreseq26,
     &         isprntpstdnb27,
     &         isprntqfmove28,
     &         isprntsimplt29
       logical
     &         isprntstacc0,
     &         isprntstate1,
     &         isprntstatep2,
     &         isprntsuboil3,
     &         isprntsysitr4,
     &         isprnttstate6,
     &         isprntvalve7,
     &         isprntvexplt8,
     &         isprntvfinl9,
     &         isprntvimplt10,
     &         isprntvlvela11,
     &         isprntvolvel12,
     &         isprnttrip13,
     &         isprntpower14,
     &         isprntvolume15
       logical
     &         isprntjunction16,
     &         isprntheatstr17,
     &         isprntradht18,
     &         isprntreflood19,
     &         isprntfsnprdtr20,
     &         isprntcontrol21,
     &         isprntinput22,
     &         isprntmiedit23
c
       integer rstblcknumber1531
c
       equivalence (safe2,nmechk), (cpurem(1),cpurei(1,1))
c*********************************************************************
c Data dictionary for local variables
c i=integer r=real l=logical c=character
c*********************************************************************
c Type    Name                              Definition
c---------------------------------------------------------------------
c  l aflag                 = Set true in dtstep when heat
c                           structures are to be advanced.
c                           checked in htadv
c  l chngno(90)            = true if option # is set
c  r count                 = Real value of the integer ncount
c  i cpurei(5)             = Integer values (5) of the real cpurem
c                           (1,2,3) not used
c                           (4) word 4 from 105 card
c                            = ncount1, ncount for starting
c                               diagnostic edits
c                            = -1, write to dumpfil1 for
c                               ncount2 = cpurei(5), stop
c                            = -2, write to dumpfil1 for
c                               ncount2 = cpurei(5),
c                               redo timestep, write to
c                               dumpfil2, stop
c                            = -3, set in dtstep for -2 case.
c                           (5) word 5 from 105 card
c                            = ncount2, ncount for
c                               terminating diagnostic edits
c  r cpurem(5)             = Contains cpu remaining times values
c                            and advancement counts to start/stop
c                            diagnostic edit from input.
C  -----------new in Relap5 mod3.3beta---------------------------------RESTR33
c  r curtmi                = Time for next minor edit.                  RESTR33
c  r curtmj                = Time for next major edit.                  RESTR33
c  r curtrs                = Time for next restart edit.                RESTR33
c  r prevnd                = Previous time step card end time.          RESTR33
C  -----------new in Relap5 mod3.3beta---------------------------------RESTR33
c  i done                  = Integer flag indicating state of
c                            transient.
c                            0   = advancements are to continue,
c                            !=0 = advancements are to be
c                             terminated.
c  r dt                    = Current time step.
c  r dtht                  = Heat structure time step, currently
```

```
c                              the same as dthy.
c  r dthy                    = Hydrodynamic time step requested by
c                              user.
c  r dtn                     = Time step limit due to material
c                              transport stability limit
c                             (Courant limit).
c  r emass                   = Current mass error.
c  r emasso                  = Old mass error.
c  r errmax                  = Error estimate used in time step
c                              control.
c  l fail                    = Set to true if error encountered.
c  r gravcn                  = Gravitational constant (which may be
c                              set by input).
c  i help                    = Used to control debug editing and
c                              termination.
c  l integrationflag9        = on-line selection of time integration
c                              flag
c  l interactiveflag5        = interactive flag.
c  l isathenaopt10           = athena option.
c  l iscntrlsyschng6         = control system change.
c  l iscompleterestart4      = complete restart
c  l isfissionprdctchng5     = fission product change.
c  l isheatcondchng1         = heat conduction change.
c  l ishydrochng0            = hydrodynamic change.
c  l isimplicithttrnsfr6     = implicit heat transfer
c  l islevelcrossing1        = level crossing activated.
c  l islevelmodel0           = level model activated.
c  l ismajoreditenabled1     = major edit enabled
c  l isminoreditenabled2     = minor edit enabled
c  l isplotenabled3          = plot enabled
c  l isplotsqzflg12          = plot record squoz flag.
c  l ispltrcrdchng3          = plot record change.
c  l isprntaccum0            = activate accum print block
c  l isprntbrntrn1           = activate brntrn print block
c  l isprntccfl2             = activate ccfl print block
c  l isprntchfcal3           = activate chfcal print block
c  l isprntconden4           = activate conden print block
c  l isprntcontrol21         = activate control print block
c  l isprntdittus5           = activate dittus print block
c  l isprnteqfinl6           = activate eqfinl print block
c  l isprntfsnprdtr20        = activate fsnprdtr print block
c  l isprntfwdrag7           = activate fwdrag print block
c  l isprntheatstr17         = activate heatstr print block
c  l isprntht2tdp9           = activate ht2tdp print block
c  l isprnthtfinl12          = activate htfinl print block
c  l isprnthtrc113           = activate htrc print block
c  l isprnthydro15           = activate hydro print block
c  l isprntinput22           = activate input print block
c  l isprntistate17          = activate istate print block
c  l isprntjchoke18          = activate jchoke print block
c  l isprntjprop19           = activate jprop print block
c  l isprntjunction16        = activate junction print block
c  l isprntmiedit23          = activate miedit print block
c  l isprntnoncnd20          = activate noncnd print block
c  l isprntphantj21          = activate phantj print block
c  l isprntphantv22          = activate phantv print block
c  l isprntpimplt23          = activate phantv pimplt block
c  l isprntpintfc24          = activate phantv pintfc block
c  l isprntpower14           = activate power print block
c  l isprntprednb25          = activate prednb print block
c  l isprntpreseq26          = activate preseq print block
c  l isprntpstdnb27          = activate pstdnb print block
c  l isprntqfmove28          = activate qfmove print block
c  l isprntradht18           = activate radht print block
c  l isprntreflood19         = activate reflood print block
c  l isprntsimplt29          = activate simplt print block
c  l isprntstacc0            = activate stacc print block
c  l isprntstate1            = activate state print block
c  l isprntstatep2           = activate statep print block
c  l isprntsuboil3           = activate suboil print block
c  l isprntsysitr4           = activate sysitr print block
c  l isprnttrip13            = activate trip print block
c  l isprnttstate6           = activate tstate print block
c  l isprntvalve7            = activate valve print block
c  l isprntvexplt8           = activate vexplt print block
c  l isprntvfinl9            = activate vfinl print block
c  l isprntvimplt10          = activate vimplt print block
c  l isprntvlvela11          = activate vlvela print block
c  l isprntvolume15          = activate volume print block
c  l isprntvolvel12          = activate volvel print block
c  l isradiationchng4        = radiation change.
c  l isrestartenabled0       = restart enable bit.
c  l isrknchng2              = reactor kinetics change.
c  l isscdapopt11            = scdap is active
c  l istwostepflag7          = two step method
```

```
c  i nany                  = Number of mass error messages
c                            remaining to be issued.
c  i ncase                 = Case number
c                           Initially = -1 in blkdta
c                           0 if last case of series
c  i ncount                = Count of number of advancements,
c                            successful or otherwise.
c  l nmechk                = true if no mass error check requested
c                            on time step card
c  i nrepet                = Number of hydrodynamic advancements
c                            at current dt to finish a requested
c                            time step of dthy.
c  i nstsp                 = Number of standard advancements.
c  i pageno                = Page number.
c  i problemopt611         = Problem option (see ityp2 in inputd).
c                           1 = stdy-st (1 and 2 are used for new
c                                 and restart)
c                           2 = transnt
c                           3 = binary (3 and 4 are used for strip
c                           4 = fmtout
c  i problemtype05         = Problem type (see ityp1 in inputd).
c                            1 = new
c                            2 = restart
c                            3 = relap5 internal plots (not used)
c                            5 = strip
c                            6 = cmpcom (compare dump records)
c  i rstblcknumber1531     = restart block number.
c  l rwtrstpltcmprssflag13 =
c  r safe2                 = Last word in common block. Its address is
c                            obtained using locf and is used with the
c                            address of the first word, dthy, to
c                            compute the length of the common block.
c                            The length is used when the common block
c                            is written out to a file.
c                            Make sure it is on an 8-byte boundary.
c  l skipt                 = Logical flag used to control first
c                            entry processing in
c                            subroutine dtstep.
c  r stdtrn                = Steady state - transient flag
c  l stdystatetermflg8     = steady state termination
c  i succes                = Integer flag indicating success of
c                            the advancement.
c                            0 = no need to repeat advancement
c                                 with reduced time step
c                            1 = excessive truncation error
c                            2 = water property error
c                            3 = non-diagonal matrix
c                            4 = metal appears
c  r rktpow3d              = Total reactor power calculated by RELAP5/PA
c  r testda(20)            = Data array for minor edits and
c                            plotting during debug.
c                           Temporary coding is used to load data
c                            in this array and scnreq accepts
c                            testda as a legal request.
c  r timeht                = Problem time for heat structure
c                            advancements.
c  r timehy                = Problem time for hydrodynamic
c                            advancements.
c  r tmass                 = Total mass of water in system
c                            currently.
c  r tmasso                = Total mass of water in system at
c                            time = 0.0.
c  l transrotatflag14      = transient rotation flag
c                            0.0 = steady state
c                            1.0 = transient
c  l uniti                 = If true, SI units on input.
c  l unito                 = If true, SI units on output.
c***********************************************************************
*comdeck contrx
c
c  variables in comctl.h common block
c
c T Name    Definition
c r dthy    Hydrodynamic time step requested by user.
c r dtht    Heat structure time step, currently the same as dthy.
c r dtn     Time step limit due to material transport stability limit
c              (Courant limit).
c r dt      Current time step.
c r timehy  Problem time for hydrodynamic advancements.
c r timeht  Problem time for heat structure advancements.
c r errmax  Error estimate used in time step control.
c r tmass   Total mass of water in system currently.
c r tmasso  Total mass of water in system at time = 0.0.
c r emass   Current mass error.
c r emasso  Old mass error.
```

```
c r cpurem(5)  Contains cpu remaining times values and advancement
c                 counts to start/stop diagnostic edit from input.
c i cpurei  Integer values (5) of the real cpurem
c           (1=>3) not used
c           (4) word 4 from 105 card
c               = ncount1, ncount for starting diagnostic edits
c               = -1, write to dumpfil1 for ncount2 = cpurei(5), stop
c               = -2, write to dumpfil1 for ncount2 = cpurei(5),
c                    redo timestep, write to dumpfil2, stop
c               = -3, set in dtstep for -2 case.
c           (5) word 5 from 105 card
c               = ncount2, ncount for terminating diagnostic edits
c r stdtrn  Steady state - transient flag
c           0.0 = steady state
c           1.0 = transient
c r gravcn  Gravitational constant (which may be set by input).
c r testda  Data array for minor edits and plotting during debug.
c           Temporary coding is used to load data in this array and
c           scnreq accepts testda as a legal request.
c l aflag   Is set true in dtstep when heat structures are to be advance
c           checked in htadv
c i print   Misc. packed word.
c           Bits numbered 1-32 from right end.
c           1 (=1) restart enable bit,
c             (bit 1 is set in inputd if rstrip is to write formatted output
c           2 (=2) major edit enable bit,
c           3 (=4) minor edit enable bit,
c           4 (=8) plot enable bit,
c           5 (=16) complete restart switch,
c           6 (=32) interactive flag,
c           7 (=64) implicit heat transfer flag,
c           8 (=128) two step flag,
c           9 (=256) steady state termination flag,
c           10 (=512) on-line selection of time integration flag,
c           11 (=1024) athena option,
c           12 (=2048) scdap option,
c           13 (=4096) plot record squoz flag,
c           14 (=8192) restart-plot file compress flag,
c           15 (=16384) transient rotation flag,
c           16-32 restart block number.
c
c print replaced with the following variables:
c l  (replaces bit      1)     isrestartenabled0     restart enable bit.
c l  (         bit      2)     ismajoreditenabled1   major edit enable bit
c l  (         bit      3)     isminoreditenabled2   minor edit enable bit
c l  (         bit      4)     isplotenabled3        plot enable bit.
c l  (         bit      5)     iscompleterestart4    complete restart switc
c l  (         bit      6)     interactiveflag5      interactive flag.
c l  (         bit      7)     isimplicithttrnsfr6   implicit heat transfe
c l  (         bit      8)     istwostepflag7        two step flag.
c l  (         bit      9)     stdystatetermflg8     steady state terminatio
c l  (         bit     10)     integrationflag9      on-line selection of ti
c                                                    integration flag
c l  (         bit     11)     isathenaopt10         athena option.
c l  (         bit     12)     isscdapopt11          scdap option.
c l  (         bit     13)     isplotsqzflg12        plot record squoz flag.
c l  (         bit     15)     transrotatflag14      transient rotation flag
c i  (replaces bits 16-32)     rstblcknumber1531     restart block number.
c
c i succes  Integer flag indicating success of the advancement.
c           0 = no need to repeat advancement with reduced time step
c           1 = excessive truncation error
c           2 = water property error
c           3 = non-diagonal matrix
c           4 = metal appears
c i done    Integer flag indicating state of transient.
c           0   = advancements are to continue,
c           !=0 = advancements are to be terminated.
c i ncount  Count of number of advancements, successful or otherwise.
c r count   Real value of the integer ncount
c i nstsp   Number of standard advancements.
c i nrepet  Number of hydrodynamic advancements at current dt to finish
c           a requested time step of dthy.
c i help    Used to control debug editing and termination.
c i nany    Number of mass error messages remaining to be issued.
c l skipt   Logical flag used to control first entry processing in
c           subroutine dtstep.
c i iroute  Problem type and problem option from 100 card
c           Initially = -1 in blkdta
c           Set to 0 at top of rcards, reset to 1, if insufficient room
c           In inputd, it contains two integers packed into one word
c           Problem type, bits 1-6 (see ityp1 in inputd)
c            1 = new
c            2 = restart
c            3 = relap5 internal plots (not used)
```

```
c            4 = re-edit problem (not used)
c            5 = strip
c            6 = cmpcom (compare dump records)
c           Problem option, bits 7-12 (see ityp2 in inputd)
c            1 = stdy-st (1 and 2 are used for new and restart)
c            2 = transnt
c            3 = binary (3 and 4 are used for strip)
c            4 = fmtout
c           if rnewp is called in inputd, iroute = problem option
c           if rstrip is called in inputd, iroute = 5
c
c iroute replaced with the following variables:
c i  (replaces bits 1- 6)     problemtype05 (see ityp1 in inputd).
c i  (         bits 7-12)     problemopt611 (see ityp2 in inputd).
c
c i ncase   Case number
c           Initially = -1 in blkdta
c           = zero if last case of series
c l fail    Set to true if error encountered.
c l uniti   If true, SI units on input.
c l unito   If true, SI units on output.
c i iextra  Packed word containing flags indicating what models
c           were changed at restart.
c           Bits numbered 1-32 from right end.
c           1 (=1) hydrodynamic change
c           2 (=2) heat conduction change
c           3 (=4) reactor kinetics change
c           4 (=8) plot record change
c           5 (=16) radiation change
c           6 (=32) fission product change
c           7 (=64) control system change
c           This quantity could be equivalenced with nmechk
c           if this quantity is not needed to pad to an even number
c           of integer words.
c
c iextra replaced with the following variables:
c l  (replaces bit 1)     ishydrochng0           hydrodynamic change.
c l  (         bit 2)     isheatcondchng1heat    conduction change.
c l  (         bit 3)     isrknchng2             reactor kinetics change.
c l  (         bit 4)     ispltrcrdchng3         plot record change.
c l  (         bit 5)     isradiationchng4       radiation change.
c l  (         bit 6)     isfissionprdctchng5    fission product change.
c l  (         bit 7)     iscntrlsyschng6control system change.
c
c i pageno                  Page number.
c r rktpow3d              Total reactor power calculated by RELAP5/PARCS
c r safe2                   Same purpose as safe1
c l nmechk                true if no mass error check requested on time step card
c                         (no 1 bit on time step card)
c equivalenced to safe2
c imdctl(1) control word for individual code models
c           Bits numbered 1-32 from right end
c           1 (=1) level model activated
c           2 (=2) level crossing activated
c           3-31 nod used
c
c imdctl(1) replaced with the following variables:
c l  (replaces bit 1)     islevelmodel0       level model activated.
c l  (         bit 2)     islevelcrossing1    level crossing activated.
c
c imdctl(2)                                   not used
c
c ihlppr(1) and ihlppr(2) are replaced by the following logical variable
c l isprntaccum0          activate accum print block
c l isprntbrntrn1         activate brntrn print block
c l isprntccfl2           activate ccfl print block
c l isprntchfcal3         activate chfcal print block
c l isprntconden4         activate conden print block
c l isprntdittus5         activate dittus print block
c l isprnteqfinl6         activate eqfinl print block
c l isprntfwdrag7         activate fwdrag print block
c l isprntht2tdp9         activate ht2tdp print block
c l isprnthtfinl12    activate htfinl print block
c l isprnthtrc113     activate htrc print block
c l isprnthydro15     activate hydro print block
c l isprntistate17    activate istate print block
c l isprntjchoke18    activate jchoke print block
c l isprntjprop19     activate jprop print block
c l isprntnoncnd20    activate noncnd print block
c l isprntphantj21    activate phantj print block
c l isprntphantv22    activate phantv print block
c l isprntpimplt23    activate pimplt print block
c l isprntpintfc24    activate pintfc print block
c l isprntprednb25    activate prednb print block
c l isprntpreseq26    activate preseq print block
```

```
c l isprntpstdnb27    activate pstdnb print block
c l isprntqfmove28    activate qfmove print block
c l isprntsimplt29    activate simplt print block
c l isprntstacc0      activate stacc print block
c l isprntstate1      activate state print block
c l isprntstatep2     activate statep print block
c l isprntsuboil3     activate suboil print block
c l isprntsysitr4     activate sysitr print block
c l isprnttstate6     activate tstate print block
c l isprntvalve7      activate valve print block
c l isprntvexplt8     activate vexplt print block
c l isprntvfinl9      activate vfinl print block
c l isprntvimplt10    activate vimplt print block
c l isprntvlvela11    activate vlvela print block
c l isprntvolvel12    activate volvel print block
c l isprnttrip13      activate trip print block
c l isprntpower14     activate power print block
c l isprntvolume15    activate volume print block
c l isprntjunction16  activate junction print block
c l isprntheatstr17   activate heatstr print block
c l isprntradht18     activate radht print block
c l isprntreflood19   activate reflood print block
c l isprntfsnprdtr20  activate fsnprdtr print block
c l isprntcontrol21   activate control print block
c l isprntinput22     activate input print block
c l isprntmiedit23    activate miedit print block
c
c*********************************************************************
c
*comdeck genrl
c
       common /genrl/ ctitle,ptitle
       character ctitle*108,ptitle*64
c*********************************************************************
*comdeck genrlc
c
c  ctitle  contains title card of case, time, and date.
c  ptitle  contains program version identification and title.
c  uniti   units for input, si if true, british if false.
c  unito   units for output, si if true, british if false.
c  safe3   same purpose as safe1.
c*********************************************************************
C
C------------------------------------------------------------------------------
      LOGICAL FL,TERMINAL,RSTPLOT,INTEL,DEBUG,SCREEN,LHELP
      LOGICAL RESTI,PLOTI,RESTO,PLOTO,STRIPO, BINARY
C
      CHARACTER TITOLO(8)*8                                           RSTR532
      EQUIVALENCE (PTITLE,TITOLO)                                     RSTR532
      CHARACTER TITELC(14)*8                                          RSTR532
      EQUIVALENCE (CTITLE,TITELC)                                     RSTR532
C
      INTEGER*4 LEN, iwrd8, iwrd, lx, lxa, lxn, lenb, lenc, ix, iza     RSTR532
      LOGICAL lcompr                                                  RSTR532
      LOGICAL lpt                                                     RESTR33
      INTEGER nmbrst,i,j, icard, irstbl                               RSTR532u4
C     INTEGER problemtype05, problemopt611                            RSTR532u4
      character*8 frstrc(6),aW(14),cfa(1),strec(6)
      CHARACTER*1 frstrca(8)                                          RSTR532u2
      CHARACTER*1 frstrcb(8)                                          RSTR533.R
      equivalence (frstrc(2),frstrca(1))                              RSTR532u2
      equivalence (aw(2),frstrcb(1))                                  RSTR533.R
      equivalence (cfa,fa)                                            RSTR532
      REAL*8 IW(10)
      EQUIVALENCE(aW(1),IW(1))
      INTEGER*4 iaW(2,14)                                             RSTR532
      EQUIVALENCE(aW(1),iaW(1,1))                                     RSTR532
C
      REAL*8    R8aix                                                 RESTaix
      REAL*8    R8int                                                 RESTaix
      REAL*4    R4aix(2)                                              RESTaix
      REAL*4    R4int(2)                                              RESTaix
      INTEGER*1 Pointer(8)                                            RESTaix
      REAL*4    RPOINTR(2)                                            RESTaix
      INTEGER*1 I1aix1(4),I1aix2(4)                                   RESTaix
      INTEGER*1 I1int1(4),I1int2(4)                                   RESTaix
      INTEGER*4 I4aix(2), I4int(2)                                    RESTaix
      LOGICAL   L4int(2)                                              RESTaix
      EQUIVALENCE (I1int1(1), I4int(1))                               RESTaix
      EQUIVALENCE (I1int2(1), I4int(2))                               RESTaix
      EQUIVALENCE (I1int1(1), L4int(1))                               RESTaix
      EQUIVALENCE (I1int2(1), L4int(2))                               RESTaix
      EQUIVALENCE (RPOINTR(1), R8aix)                                 RESTaix
      EQUIVALENCE (I4int(1), R8int)                                   RESTaix
      EQUIVALENCE (I4int(1), R4int(1))                                RESTaix
      EQUIVALENCE (RPOINTR(1), Pointer(1))                              RESTaix
      EQUIVALENCE (RPOINTR(1), I4aix(1))                                RESTaix
      EQUIVALENCE (RPOINTR(1), I1aix1(1))                               RESTaix
      EQUIVALENCE (RPOINTR(2), I1aix2(1))                               RESTaix
C
      character*8 label
      PARAMETER (NFIL=13)
      character*50 arg,progr,arg0
      CHARACTER cword*8                                                 RESTLH9
      INTEGER ISTATUS
      INTEGER RSIN,RSOUT,RSINF,PLOUT,PLINF,DEBU,IOUT                    RSTR532u1
      INTEGER RSTPLT,RSTIN,STRIPF                                       RSTR532
C     CHARACTER args(NFIL)*4,FNAMES(NFIL)*40,FORMS(3)*12
      CHARACTER args(NFIL)*4,FNAMES(NFIL)*60,FORMS(3)*12                RSTR532u3
      CHARACTER STAT(3)*7                                               RESTLH9
C
      DIMENSION IUN(NFIL),IFORMS(NFIL),ISTAT(NFIL)
      data forms/'FORMATTED   ','UNFORMATTED ','            '/
      data STAT /'OLD    ','UNKNOWN','       '/
      DATA (IUN(I),IFORMS(I),ISTAT(I),ARGS(I),FNAMES(I),I=1,NFIL)
     &/12,2,2,'-ri ','RESTINP                                           RSTR532u3
     &          '                                                       RSTR532u3
     &,11,2,2,'-ro ','RESTOUT                                           RSTR532u3
     &          '                                                       RSTR532u3
     &, 7,1,2,'-or ','RESTINF                                           RSTR532u3
     &          '                                                       RSTR532u3
     &, 9,1,2,'-po ','PLOTOUT                                           RSTR532u3
     &          '                                                       RSTR532u3
     &, 8,1,2,'-op ','PLOTINF                                           RSTR532u3
     &          '                                                       RSTR532u3
     &,10,1,2,'-so ','STRIPINP                                          RSTR532u3
     &          '                                                       RSTR532u3
     &,13,1,2,'-d  ','                                                  RSTR532u3
     &          '                                                       RSTR532u3
     &, 0,3,3,'-r  ','                                                  RSTR532u3
     &          '                                                       RSTR532u3
     &, 0,3,3,'-s  ','                                                  RSTR532u3
     &          '                                                       RSTR532u3
     &, 0,3,3,'-p  ','                                                  RSTR532u3
     &          '                                                       RSTR532u3
     &, 0,3,3,'-no ','LOGGING TO SCREEN                                 RSTR532u3
     &          '                                                       RSTR532u3
     &, 0,3,3,'-h  ','                                                  RSTR532u3
     &          '                                                       RSTR532u3
     &, 0,3,3,'?   ','                                                  RSTR532u3
     &          '/                                                      RSTR532u3
C
C
C     DATA FRSTRC/'        ','        ','RESTART ','PLOT FIL',
C    1'E       ','        '/
      DATA FRSTRC/'        ','        ','restart-','plot fil',          RSTR532
     1'e       ','        '/                                            RSTR532
C
       DATA STREC/'        ','        ','STRIP FI','LE      ',
     *'        ','        '/
C
C      DATA PTITLE/'RELAP5/M','OD1/019I','REACTOR ','LOSS OF ',         RESTLH9
C    *'COOLANT ','ANALYSIS',' PROGRAM'/                                 RESTLH9
C      DATA TITOLO/'RELAP5/M','OD1/019I','REACTOR ','LOSS OF ',         RESTLH9
C    *'COOLANT ','ANALYSIS',' PROGRAM'/                                 RESTLH9
C      DATA TITOLO/' Relap 5','/3.2mz  ','REACTOR ','LOSS OF ',         RSTR532
C    *'COOLANT ','ANALYSIS',' PROGRAM','        '/                      RSTR532
C      DATA TITOLO/' Relap 5','/3.2pt  ','REACTOR ','LOSS OF ',         RESTR33
C    *'COOLANT ','ANALYSIS',' PROGRAM','        '/                      RESTR33
C      DATA TITOLO/' Relap 5','/3.3    ','REACTOR ','LOSS OF ',         RESTR33.R
C    *'COOLANT ','ANALYSIS',' PROGRAM','        '/                      RESTR33.R
C
      DATA TERMINAL/.TRUE./
      DATA RSTPLOT/.TRUE./
      DATA STRIPO/.FALSE./
      DATA BINARY/.FALSE./
      DATA RESTO/.FALSE./
      DATA PLOTO/.FALSE./
      DATA RESTI/.FALSE./
      DATA PLOTI/.FALSE./
      DATA INTEL/.TRUE./                                                RSTR5322
      DATA DEBUG/.FALSE./
      DATA LHELP/.FALSE./
      DATA SCREEN/.TRUE./
C
C     PTITLE=' Relap 5/3.2m   REACTOR LOSS OF COOLANT ANALYSIS PROGRAM  RSTR532
C     PTITLE=' Relap 5/3.2pt  REACTOR LOSS OF COOLANT ANALYSIS PROGRAM  REST33
      PTITLE=' Relap 5/3.3    REACTOR LOSS OF COOLANT ANALYSIS PROGRAM  REST33.R
     *     '                                                            RSTR532
C     arg0  ='RSTR5322G.EXE Ver 2.2.1 19.01.2001'                       RSTR53u1
```

```
C     arg0  ='RSTR5322G.EXE Ver 2.2.2 04.03.2001'                       RSTR53u2
C     arg0  ='RSTR5322G.EXE Ver 2.2.3 27.04.2001'                       RSTR53u3
C     arg0  ='RSTR5322G.EXE Ver 2.2.4 11.06.2001'                       RSTR53u4
C     arg0  ='RSTR533B.EXE  Ver 2.3   23.09.2001'                       RESTR33
C     arg0  ='RSTR533B.EXE  Ver 2.3.0 07.10.2001'                       RESTR33.0
C     arg0  ='RSTR533B.EXE  Ver 2.3.1 21.08.2002'                       RESTR33.1
C     arg0  ='RSTR533.EXE   Ver 2.4   08.08.2002'                       RESTR33.R
      arg0  ='RSTR533.EXE   Ver 2.4.1 21.08.2002'                       RESTR33R1
C     FRSTRC(2) = PTITLE(2)                                             RESTLH9
      FRSTRC(1) = TITOLO(1)                                             RESTLH9
      FRSTRC(2) = TITOLO(2)                                             RESTLH9
C
C ----------------------------------------------------------------------
C
*     do 9 I= 1,NFIL
*        print *,' I/O = ',iun(I),' ',forms(iforms(I)),' ',
*    &           stat(istat(I)),' ',args(I),' ',fnames(I)(1:20)
*  9  continue
C
      narg=nargs()
C
      i0=0
      i1=1
      i2=narg-1
      call getarg(i0,progr,istatus)
      progr = arg0                                                      RSTR532
C     print *,'narg=',narg,' i0=',i0,' progr=',progr,' Length= ',istatus
      i=i1
      do 10 while (i.le.i2)
         call getarg(i,arg,istatus)
C        print *,'i=',i,' arg=',arg
         do j=1,nfil
            if (arg(1:4).eq.'-r  ') then
               TERMINAL =.FALSE.
               RESTO    =.TRUE.
               write (fnames(8)(1:20),'(a20)') 'WRITE RESTART FILE  '
               i = i + 1
               go to 10
            endif
            if (arg(1:4).eq.'-s  ') then
               TERMINAL =.FALSE.
               PLOTO    =.FALSE.
               RESTO    =.FALSE.
               STRIPO   =.TRUE.
               write (fnames(9)(1:20),'(a20)') 'WRITE STRIP INFILE  '
               i = i + 1
               go to 10
            endif
            if (arg(1:4).eq.'-p  ') then
               TERMINAL =.FALSE.
               PLOTO    =.TRUE.
               write (fnames(10)(1:20),'(a20)') 'WRITE PLOT-DATA FILE'
               i = i + 1
               go to 10
            endif
            if (arg(1:4).eq.'-no ') then
               SCREEN   =.FALSE.
               write (fnames(11)(1:20),'(a20)') 'NO SCREEN OUTPUT    '
               i = i + 1
               go to 10
            endif
            if ((arg(1:4).eq.'-h  ').or.(arg(1:4).eq.'?   ')) then
               LHELP     =.TRUE.
               write (fnames(12)(1:20),'(a20)') 'Help invoked        '
               write (fnames(13)(1:20),'(a20)') 'Help invoked        '
               i = i + 1
               go to 10
            endif
            if (arg(1:4).eq.'-d  ') then
               DEBUG     =.TRUE.
               write (fnames(7)(1:20),'(a20)') 'DEBUG               '
               i = i + 1
               go to 10
            endif
            if (arg(1:4).eq.args(j))then
C              call getarg(i+1,fnames(j),istatus)                       RESTLH9
               call getarg(i+1,arg,istatus)                             RESTLH9
               fnames(j) = arg                                          RESTLH9
C              print *,'i+1=',i+1,' fname=',fnames(j)
               i = i + 2
               go to 10
            endif
         enddo
         print *,'argument ',arg(1:4),' not existent'
         i = i + 1
```

```
 10   continue
C
      IF (LHELP) GOTO 9000
C
      RSTIN  = iun(1)
      RSOUT  = iun(2)
      RSINF  = iun(3)
      PLOUT  = iun(4)
      PLINF  = iun(5)
      STRIPF = iun(6)
      DEBU   = iun(7)
C                                                                       RSTR532u1
      IF(SCREEN) THEN                                                   RSTR532u1
        IOUT   = 0                                                      RSTR532u1
      ELSE                                                              RSTR532u1
        IF (DEBUG) THEN                                                 RSTR532u1
           IOUT = DEBU                                                  RSTR532u1
        ELSE                                                            RSTR532u1
           IOUT = RSINF                                                 RSTR532u1
        END IF                                                          RSTR532u1
      ENDIF                                                             RSTR532u1
C                                                                       RSTR532u1
      if (DEBUG) then
         open (DEBU,FILE=fnames(7),FORM=forms(iforms(7))
     &      ,STATUS=stat(istat(7)))
         write (DEBU,'(A)') ' DEBUGGING INFORMATIONS'
         write (DEBU,'(A)') ' Parameters in commandline processed:'
         do 11 I= 1,NFIL
            write (DEBU,'(A,I3,A,A,A,A,A,A,A)')' I/O = ',iun(I),' ',
     &                                          forms(iforms(I)),' ',
     &                  stat(istat(I)),' ',args(I),' ',fnames(I)(1:60)  RSTR532u3
 11      continue
      end if
C
C ---------------------------------------------------------------------
C     CHECK Machine Version
C ---------------------------------------------------------------------
C
      open (RSTIN,FILE=fnames(1),FORM='UNFORMATTED'                     RSTR32a
     &     ,STATUS=stat(istat(1)),ACCESS='TRANSPARENT')                 RSTR32a
      READ (RSTIN,       err=20,IOSTAT=IOS) POINTER                     RSTR32a
      if (POINTER(1).ne.8) INTEL = .FALSE.                              RSTR32a
C     If not INTEL machine then write restart w/o plot data is          RSTR32a
C     not indented                                                      RSTR32a
      IF(.NOT.INTEL) RESTO = .FALSE.                                    RSTR32a
      if (DEBUG) then                                                   RSTR32a
        write (DEBU,'(1x,A,8(1x,I2))') 'FIRST 8 BYTES : ',POINTER       RSTR32a
      endif                                                             RSTR32a
      if (screen) then                                                  RSTR32a
        if (INTEL) then                                                 RSTR32a
C          write (*,*) ' Machine Version: INTEL / PC'                   RSTR32a
           write (*,*) ' Machine Version: Intel or DEC-Alpha'           RSTR32u2
C       if (DEBUG) write (DEBU,'(A)') ' Machine Version: INTEL / PC'    RSTR32a
        if (DEBUG) write (DEBU,'(A)') ' Machine Version: INTEL or DEC-AlRSTR32u2
     &pha'                                                              RSTR32u2
        else                                                            RSTR32a
C          write (*,*) ' Machine Version: WS / AIX'                     RSTR32a
           write (*,*) ' Machine Version: IBM-AIX, SunOS56, SGI64, HPC18RSTR32u2
     &0'                                                                RSTR32u2
C       if (DEBUG) write (DEBU,'(A)') ' Machine Version: WS / AIX'      RSTR32a
        if (DEBUG) write (DEBU,'(A)') ' Machine Version: IBM-AIX, SunOS5RSTR32u2
     &6, SGI64, HPC180'                                                 RSTR32u2
        end if                                                          RSTR32a
      endif                                                             RSTR32a
      close (RSTIN)                                                     RSTR32a
C
C ---------------------------------------------------------------------
C     CHECK FILE HEADER and RELAP-VERSION
C ---------------------------------------------------------------------
C
      IF(INTEL) THEN
         open (RSTIN,FILE=fnames(1),FORM=forms(iforms(1))               RESTLH9
     &      ,STATUS=stat(istat(1)))                                     RESTLH9
         read (RSTIN,end=20,err=20) LEN,iwrd8
         read (RSTIN,end=20,err=20) (aW(i),i=1,LEN)
      ELSE
         open (RSTIN,FILE=fnames(1),FORM='UNFORMATTED'                  RSTR32a
     &      ,STATUS=stat(istat(1)),ACCESS='TRANSPARENT')                RSTR32a
         READ (RSTIN,err=20,IOSTAT=IOS) RPOINTR(2),LEN,iwrd8            RSTR32a
         I4int(2)=Iflip(I4aix(2))                                       RSTR32a
*        WRITE(*,*) I4int(2)                                            RSTR32a
         LEN = Iflip(LEN)                                               RSTR32a
         iwrd8 = Iflip(iwrd8)                                           RSTR32a
*        WRITE(*,*) LEN, iwrd8                                          RSTR32a
         READ (RSTIN,err=20,IOSTAT=IOS) RPOINTR                         RSTR32a
```

```
            I4int(1)=Iflip(I4aix(1))                                    RSTR32a
            I4int(2)=Iflip(I4aix(2))                                    RSTR32a
*           WRITE(*,*) I4int                                            RSTR32a
            READ (RSTIN,err=20,IOSTAT=IOS) (aW(i),i=1,LEN)              RSTR32a
*           WRITE(*,*) aW(1),' ',aw(2),' ',aw(3),' ',aw(4),' '
*           WRITE(*,*) aw(5),' ',aw(6),' ',aw(7),' ',aw(8),' '
         ENDIF
C
C        iaW(2,10) = iroute: replaced by
         IF (INTEL) THEN
            problemopt611 = iaW(1,10)
            problemtype05 = iaW(2,10)
         ELSE
            problemopt611 = IFlip(iaW(1,10))                            RSTR5322
            problemtype05 = IFlip(iaW(2,10))                            RSTR5322
         END IF
C
         if (DEBUG) then
           write (DEBU,'(1x,A,I6,A,I6)') 'LEN = ',LEN,'  iwrd8 =', iwrd8
           write (DEBU,'(1x,A)')'-------- -------- -------- -------- ------
        &--'
           write (DEBU,'(1x,9A)')aW(1),' ',aW(2),' ',aW(3),' ',aW(4),' ',
        &                      aW(5)
           write (DEBU,'(1x,9A)')aW(6),' ',aW(7),' ',aW(8),' ',aW(9),' ',
        &                      aW(10)
           write (DEBU,'(1x,A,I6)') 'problemopt611 = ',problemopt611
           write (DEBU,'(1x,A,I6)') 'problemtype05 = ',problemtype05
         endif
C
         GO TO 30
C
      20 WRITE (*,2020) IOS
    2020 FORMAT (1X,'****** ERROR ON FIRST READ OF INPUT FILE. '
        &  ,' IOS = ',I4)
         GO TO 25
C
C -----------------------------------------------------------------
C        CHECK RELAP VERSION (RELAP5/MOD3.2xx)                          RSTR532
C        CHECK RELAP VERSION (RELAP5/MOD3.3)                            RSTR533.R
C -----------------------------------------------------------------
C
      30 CONTINUE
         if (DEBUG) then
            write (DEBU,'(1x,A)')'Check Relap Version'
            write (DEBU,'(1x,A)') aW(2)
            write (DEBU,'(1x,A)') FRSTRC(2)
         endif
         IF (aW(2) .EQ. FRSTRC(2)) GO TO 35
C        Check additionally for Development Versions aa .. zz           RSTR33.R
C              FRSTRCA ( 1 ) =  /                                       RSTR33.R
C              FRSTRCA ( 2 ) =  3                                       RSTR33.R
C              FRSTRCA ( 3 ) =  .                                       RSTR33.R
C              FRSTRCA ( 4 ) =  3                                       RSTR33.R
C              FRSTRCA ( 5 ) =  a .. z    CHAR(97)  = a                 RSTR33.R
C              FRSTRCA ( 6 ) =  a .. z    CHAR(122) = z                 RSTR33.R
C              FRSTRCA ( 7..8 ) =  " "                                  RSTR33.R
         DO j=5,6                                                       RSTR33.R
            DO i=97,122                                                 RSTR532u2
C              frstrca(6) = CHAR(i)                                     RSTR532u2
               frstrca(j) = CHAR(i)                                     RSTR33.R
C              write (DEBU,'(1x,A,5x,A)') aW(2),FRSTRC(2)               RSTR532u2
               if (frstrcb(j) .EQ. frstrca(j)) go to 34                 RSTR33.R
            END DO                                                      RSTR33.R
      34 CONTINUE                                                       RSTR33.R
            IF (aW(2) .EQ. FRSTRC(2)) GO TO 35                          RSTR532u2
         END DO                                                         RSTR532u2
         WRITE (*,2030)
         lpt = .FALSE.                                                  RESTR33
    2030 FORMAT (1X,'****** RESTART-PLOT FILE IS NOT COMPATIBLE WITH'
        &,' THIS VERSION OF THE PROGRAM.')
         GO TO 25
C
C        CHECK IF RESTART-PLOT or STRIP OUTPUT
C
      35 CONTINUE
         lpt = .TRUE.                                                   RESTR33
         if (DEBUG) then
            write (DEBU,'(1x,A)') 'Check if RESTART-PLOT or STRIP OUTPUT'
            write (DEBU,'(1x,A,5x,a)')aW(4),aW(5)
            write (DEBU,'(1x,A,5x,a)')FRSTRC(3),FRSTRC(4)
            write (DEBU,'(1x,A,5x,a)')STREC(3),STREC(4)
         endif
         IF (aW(4).EQ.FRSTRC(3).AND.aW(5).EQ.FRSTRC(4)) RSTPLOT = .TRUE.
         IF (aW(4).EQ. STREC(3).AND.aW(5).EQ. STREC(4)) RSTPLOT = .FALSE.
         IF (STRIPO) RSTPLOT = .FALSE.
```

```
C
      IF(.NOT.RSTPLOT)RESTO=.FALSE.
      GO TO 40
   25 CLOSE (RSTIN)
      GO TO 1000
C ----------------------------------------------------------------------
C     EVERYTHING IS OK, NOW OPEN ALL REQUIRED FILES
C ----------------------------------------------------------------------
   40 CONTINUE
      REWIND (RSTIN)
C     -----------------
      IF(RSTPLOT) THEN
         DO I=3,5,2
            open (IUN(I),FILE=fnames(I),FORM=forms(iforms(I))
     &                 ,STATUS=stat(istat(I)))
         ENDDO
      ELSE
         write (fnames(3)(1:20),'(a20)') '                    '
         open (IUN(5),FILE=fnames(5),FORM=forms(iforms(5))
     &              ,STATUS=stat(istat(5)))
         open (IUN(6),FILE=fnames(6),FORM=forms(iforms(6))
     &              ,STATUS=stat(istat(6)))
      ENDIF
C     -----------------
      IF (RESTO) THEN
         open (IUN(2),FILE=fnames(2),FORM=forms(iforms(2))
     &              ,STATUS=stat(istat(2)))
      ELSE
         write (fnames(2)(1:20),'(a20)') '                    '
      ENDIF
C     -----------------
      IF (PLOTO) THEN
         open (IUN(4),FILE=fnames(4),FORM=forms(iforms(4))
     &              ,STATUS=stat(istat(4)))
      ELSE
         write (fnames(4)(1:20),'(a20)') '                    '
      ENDIF
C     -----------------
      IF (STRIPO) THEN
         open (IUN(6),FILE=fnames(6),FORM=forms(iforms(6))
     &              ,STATUS=stat(istat(6)))
      ELSE
         write (fnames(6)(1:20),'(a20)') '                    '
      ENDIF
C
C ----------------------------------------------------------------------
      call cdate(cword)
C ----------------------------------------------------------------------
C
      IF (SCREEN) THEN
      IF (RSTPLOT) THEN
C        write(*,'(/1X,a\)')'RELAP5 MOD3.2 RESTART/PLOT INFORMATION    'RSTR532
         write(*,'(/1X,a\)')'RELAP5 MOD3.3 RESTART/PLOT INFORMATION    'RSTR533.R
      ELSE
C        write(*,'(/1X,a\)')'RELAP5 MOD3.2 STRIP INFORMATION           'RSTR532
         write(*,'(/1X,a\)')'RELAP5 MOD3.3 STRIP INFORMATION           'RSTR533.R
      ENDIF
      WRITE(*,'(1x,a,a10)')'Date (MM/DD/YY):',cword
      write(*,'(1x,a,a50)')'Program name   :',progr
      write(*,*)' ',(args(i),'  ',fnames(i)(1:20),i=1,3)
      write(*,*)' ',(args(i),'  ',fnames(i)(1:20),i=4,6)
      write(*    ,*)' -----------------------------------------------
     &---------------------'
      END IF
C
      write(PLINF,'(1X,a\)',ERR=900)'RELAP 5MOD3 PLOT PARAMETER INFORMAT
     &ION '
      write(PLINF,'(1x,a,a10)')'Date (MM/DD/YY):',cword
      write(PLINF,'(1x,a,a50)')'Program name: ',progr
C     write(PLINF,'(1X,a,a20)')'Input file  : ',fnames(1)(1:20)
      write(PLINF,'(1X,a,a60)')'Input file  : ',fnames(1)(1:60)         RSTR32u3
      write(PLINF,*)'-------------------------------------------------
     &--------------------'
C
      IF (PLOTO) THEN
C        write(PLOUT,'(1X,a\)',ERR=901)'RELAP 5MOD3.2 PLOT INFORMATION '
         write(PLOUT,'(1X,a\)',ERR=901)'RELAP 5MOD3.3 PLOT INFORMATION 'RSTR33.R
         write(PLOUT,'(1x,a,a10)')'Date (MM/DD/YY):',cword
         write(PLOUT,'(1x,a,a50)')'Program name: ',progr
C        write(PLOUT,'(1X,a,a20)')'Input file  : ',fnames(1)(1:20)
         write(PLOUT,'(1X,a,a60)')'Input file  : ',fnames(1)(1:60)      RSTR32u3
      write(PLOUT,*)'-------------------------------------------------
     &--------------------'
      ENDIF
C
```

```
      IF (RSTPLOT) THEN
         write(RSINF,'(1X,a\)',ERR=997)'RELAP 5MOD3 RESTART INFORMATION'
         write(RSINF,'(1x,a,a10)')'Date (MM/DD/YY):',cword
         write(RSINF,'(1x,a,a50)')'Program name: ',progr
C        write(RSINF,'(1X,a,a20)')'Input file  : ',fnames(1)(1:20)
         write(RSINF,'(1X,a,a60)')'Input file  : ',fnames(1)(1:60)      RSTR32u3
      write(RSINF,*)'--------------------------------------------------
     &--------------------'
      ENDIF
C
C ----------------------------------------------------------------------
C     NOW READ FILE HEADER AND THE OTHER DATA OF RST-PLT FILE
C ----------------------------------------------------------------------
C
      IF(INTEL) THEN
         read (RSTIN,end=20,err=20) LEN,iwrd8
         read (RSTIN,end=20,err=20) (aW(i),i=1,LEN)
      ELSE
         READ (RSTIN,err=20,IOSTAT=IOS) RPOINTR(2),LEN,iwrd8            RSTR32a
         LEN = Iflip(LEN)                                               RSTR32a
         iwrd8 = Iflip(iwrd8)                                           RSTR32a
*        WRITE(*,*) LEN, iwrd8                                          RSTR32a
         READ (RSTIN,err=20,IOSTAT=IOS) RPOINTR                         RSTR32a
         READ (RSTIN,err=20,IOSTAT=IOS) (aW(i),i=1,LEN)                 RSTR32a
      ENDIF
      IF (SCREEN) THEN
         WRITE (*,2004) IW(1),IW(2),IW(7),IW(8)
      ENDIF
C
      IF(.NOT.TERMINAL) THEN
         IF (RSTPLOT) THEN
            IF (PLOTO) WRITE (PLOUT,2004) IW(1),IW(2),IW(7),IW(8)       RSTR532
            IF (RESTO) WRITE (RSOUT,ERR=998) LEN,iwrd8                  RSTR532
            IF (RESTO) WRITE (RSOUT,ERR=998) (aW(KL),KL=1,LEN)          RSTR532
         ELSE
            IF (PLOTO) WRITE (PLOUT,2005) IW(1),IW(2),IW(7),IW(8)       RSTR532
         ENDIF
      ENDIF
      IF (RSTPLOT) THEN
         WRITE (RSINF,2004) IW(1),IW(2),IW(7),IW(8)                     RSTR532
         WRITE (PLINF,2004) IW(1),IW(2),IW(7),IW(8)                     RSTR532
      ELSE
         WRITE (PLINF,2005) IW(1),IW(2),IW(7),IW(8)                     RSTR532
      ENDIF
C
      IF(STRIPO) THEN
         WRITE (STRIPF,2001)
         WRITE (STRIPF,2002)
         WRITE (STRIPF,2001)
         WRITE (STRIPF,2002)
         WRITE (STRIPF,2003) IW(1),IW(2),IW(7),IW(8)
         WRITE (STRIPF,2002)
         WRITE (STRIPF,2006) fnames(1)(1:40)                            RSTR32u3
         WRITE (STRIPF,2007) cword
         WRITE (STRIPF,2002)
         WRITE (STRIPF,2008) fnames(6)(1:40)                            RSTR32u3
         WRITE (STRIPF,2009)
         WRITE (STRIPF,2001)
      ENDIF
C
 2001 FORMAT ('* ', 77('='))
 2002 FORMAT ('* ')
 2003 FORMAT ('* RESTART FILE WAS WRITTEN BY PROGRAM ',     2A8
     &       ,'             ON ',A8,A1)
 2004 FORMAT (' RESTART INPUT FILE WAS WRITTEN BY PROGRAM ',2A8
     &       ,' ON ',A8,A1)
 2005 FORMAT (' STRIP INPUT SOURCE WAS WRITTEN BY PROGRAM ',2A8
     &       ,' ON ',A8,A1)
 2006 FORMAT ('* RESTART FILE NAME     : ',A40)                         RSTR32u3
 2007 FORMAT ('* DATE (MM/DD/YY)       : ',A10)
 2008 FORMAT ('* STRIP-INPUT FILE NAME : ',A30)
 2009 FORMAT ('* STRIP-OUTPUT WILL BE IN ASCII FORMAT (FMTOUT)')
 2010 FORMAT ('* NUMBER OF PARAMETERS IN THIS PROBLEM : ',I4)
 2011 FORMAT ('* ONLY THE FIRST 999 PARAMETERS ARE CONSIDERED.')
 2012 FORMAT ('* THE REMAINDER OF ',I4,' ARE TREATED AS COMMENT (*).')
 2013 FORMAT ('* NOTE: STRIP PARAMETERS VALID ONLY 1001 .. 1999.')      RSTR33.0
C
C ----------------- NOW READ RESTART/PLOT/STRIP DATA -------------------
C
      nmbrst = 0
      LENB = 0
      LENC = 0                                                          RSTR532
      ix = 1
      IZA = 1
C
```

```
C--------------------------------------LOOP----------------------------
C
      if (DEBUG) then
         write (DEBU,'(1x,A)') 'Enter RESTART-PLOTDATA LOOP'
      endif
      GOTO 50
   45 continue
      IF(STRIPO) THEN
         WRITE (STRIPF,2010) lenb-2
         if (lenb.gt.999) then
C           WRITE (STRIPF,2011)                                         RSTR33.0
C           WRITE (STRIPF,2012) (lenb-2-999)                            RSTR33.0
            WRITE (STRIPF,2013)                                         RSTR33.0
         end if
         WRITE (STRIPF,2001)
      ENDIF
      GOTO 50
C
   50 continue
C
C ---------------------------------------------------------------------
C     READ RECORD LENGTH AND WORD-WIDTH
C ---------------------------------------------------------------------
C
      if(INTEL) then
         read (rstin,err=903,end=902,iostat=ios) lx, iwrd              RSTR532
      else
         READ (RSTIN,err=903,iostat=ios) RPOINTR                        RSTR32a
         I4int(1)=Iflip(I4aix(1))                                       RSTR32a
         I4int(2)=Iflip(I4aix(2))                                       RSTR32a
*        WRITE(*,*) 'A:I4int = ',I4int(1), I4int(2)                     RSTR32a
         IF(I4int(2).eq.0)goto 902
         READ (RSTIN,err=903,iostat=ios) lx, iwrd                       RSTR32a
         lx = Iflip(lx)                                                 RSTR32a
         iwrd = Iflip(iwrd)                                             RSTR32a
      endif
      j = lx*iwrd/8
      lx = j
*     write (*,*) lx, iwrd
C
      if(INTEL) then
         read (rstin,err=903,end=902,iostat=ios) (fa(i+ix-1),i=1,lx)    RSTR532
      else
         READ (RSTIN,err=903,IOSTAT=IOS) RPOINTR                        RSTR32a
         I4int(1)=Iflip(I4aix(1))                                       RSTR32a
         I4int(2)=Iflip(I4aix(2))                                       RSTR32a
*        WRITE(*,*) 'B:I4int = ',I4int(1), I4int(2)                     RSTR32a
         READ (RSTIN,err=903,IOSTAT=IOS) (fa(i+ix-1),i=1,lx)            RSTR32a
      endif
      label = cfa(ix)                                                   RSTR532
*     WRITE(*,*) label
      if (DEBUG) then
        write (DEBU,'(1x,A,I6,A,I6,A,A)') 'lx  = ',lx,'  iwrd  =', iwrd,
     &                                    ' label = ',label
      endif
   51 IF (label.eq.'plotrec ') THEN
         IF (.NOT.TERMINAL) THEN
            if (ploto) write(PLOUT,'(/1x,a8)')label
         ENDIF
         if (SCREEN) then
            IF (IZA.eq.1) write(*,'(/1x,a8\)')label
            write(*,'(A1\)')'.'
         endif
         IZA = IZA + 1
         GO TO 90
      ENDIF
      IF(.NOT.TERMINAL) THEN
         IF (RSTPLOT) THEN
            IF (RESTO) WRITE (RSOUT,ERR=998) lx, iwrd                   RSTR532
            IF (RESTO) WRITE (RSOUT,ERR=998) (fa(i+ix-1),i=1,lx)        RSTR532
         ENDIF
      ENDIF
      IF (label.eq.'plotinf ') GO TO 60                                 RSTR532
      IF (label.eq.'plotalf ') GO TO 65                                 RSTR532
      IF (label.eq.'plotnum ') GO TO 70                                 RSTR532
      IF (label.eq.'restart ') GO TO 80                                 RSTR532
C        no label ---> common block of restart will be read
      RESTI = .TRUE.
      GO TO 85
C
C--------------------------------------PLOTINF-------------------------
C
C     READ PLOTINF RECORDS
C
C ---------------------------------------------------------------------
```

```
C
C     ---------------First read Length of plotdata record
C
   60 continue
      if(INTEL) then
         lenb = ia(2,ix+1)                                              RSTR5322
         lenc = ia(2,ix+2)                                              RSTR5322
      else
         lenb = Iflip(ia(2,ix+1))                                       RSTR5322
         lenc = Iflip(ia(2,ix+2))                                       RSTR5322
      endif
      if (DEBUG) then
         WRITE(DEBU,'(1x,A,I6)') 'LENGTH OF PLOTINF-RECORDS: ',lenb     RSTR532
         WRITE(DEBU,'(1x,A,I6)') 'LENGTH OF PLOTREC-RECORDS: ',lenc     RSTR532
      end if
      lcompr = .false.
      if (lenc.NE.((lenb+2)/2)) then                                    RSTR532
         GOTO 906                                                       RSTR532
      ELSE                                                              RSTR532
         if (DEBUG) then
            WRITE(DEBU,'(1x,A)') 'PLOTRECORDS ARE COMPRESSED!!'         RSTR532
         endif
         lcompr = .true.
      endif
      GOTO 61
C ----------------------------------------------------------------------
C     Write this info later, when Problem has been read in restart block
      IF (.NOT.TERMINAL) THEN                                           RSTR532
C        if (PLOTO) write(PLOUT,'(I4)') ' ',lenb-1                      RSTR532
         if (PLOTO) write(PLOUT,'(1X,I6)') ' ',lenb-1                   RSTR33.11
C        no write into RSTOUT FILE                                      RSTR532
      ENDIF                                                             RSTR532
C     write(PLINF,'(I4)') ' ',lenb-1                                    RSTR532u3
      write(PLINF,'(1X,I6)') ' ',lenb-1                                 RSTR33.11
C ----------------------------------------------------------------------
   61 ix = ix + lx                                                      RSTR532
      GOTO 45                                                           RSTR532
C
C ----------------------------------------------------------------------
C
C     ---------------Second read PLOTALF record (parameters)
C
   65 lxa = lx                                                          RSTR532
      if (DEBUG) then
         WRITE(DEBU,'(1x,2(A,I6),A)') 'PLOTALF-RECORD : ',lxa,
     &                                ' x ',iwrd,'  BYTES'
      endif
      ix = ix + lx                                                      RSTR532
      GOTO 50                                                           RSTR532
C
C ----------------------------------------------------------------------
C
C     ---------------Third Read PLOTNUM record (volumes/junctions,...)
C
   70 lxn = lx                                                          RSTR532
      if (lxa.ne.lxn) then                                              RSTR532
         GOTO 904                                                       RSTR532
      end if                                                            RSTR532
      if (DEBUG) then
         WRITE(DEBU,'(1x,2(A,I6),A)') 'PLOTNUM-RECORD : ',lxn,
     &                                ' x ',iwrd,'  BYTES'
      end if
      ix = ix + lx                                                      RSTR532
      GOTO 50                                                           RSTR532
C
C --------------------------------------RESTART INFO--------------------
C     READ RESTART INFORMATION
C--------------------------------------RST-BLOCK-----------------------
C
   80 continue
      if(INTEL) then
C        Read print variable (2 * I4)                                   RSTR532
         iprint = ia(2,ix+1)                                            RSTR532
C        Read ncount variable (2 * I4)                                  RSTR532
         ncount = ia(2,ix+2)                                            RSTR532
      else
         iprint = Iflip(ia(2,ix+1))                                     RSTR5322
         ncount = Iflip(ia(2,ix+2))                                     RSTR5322
      endif
C        i print and n count variable are not longer suportet in Relap  RSTR533.R
C        version 3.3                                                    RSTR533.R
      if (DEBUG) then
C        WRITE(DEBU,'(1x,A,I6)') 'PRINT VARIABLE           : ',iprint   RSTR532
C        WRITE(DEBU,'(1x,A,I6)') 'NCOUNT VARIABLE (NO.RST) : ',ncount   RSTR532
         WRITE(DEBU,'(1x,A,I6,A)') 'PRINT VARIABLE           : ',iprint,RSTR533.R
     *                             '  Not supportet in Version 3.3!'    RSTR533.R
        WRITE(DEBU,'(1x,A,I6,A)') 'NCOUNT VARIABLE (NO.RST) : ',ncount,RSTR533.R
     *                             '  Not supportet in Version 3.3!'    RSTR533.R
      end if
      ix = ix + lx
C     if (ncount>0) GOTO 84                                             RSTR32u4
C        The next line is no longer valid in version 3.3 and replaced   RSTR533.R
C        by the line following the lx checking for next word            RSTR533.R
C     if (iprint>32767) GOTO 84                                         RSTR32u4
C                flag 16 - 32 = Restart Blocknumber (=0 for restart)    RSTR32u4
C
C     The GENRL Data and the comctl Data are read in only the first time
C     for each new or restartet problem
C
      if(INTEL) then
         read (rstin,err=903,end=902,iostat=ios) lx, iwrd               RSTR532
      else
         READ (RSTIN,err=903,IOSTAT=IOS) RPOINTR                        RSTR32a
         I4int(1)=Iflip(I4aix(1))                                       RSTR32a
         I4int(2)=Iflip(I4aix(2))                                       RSTR32a
*        WRITE(*,*) I4int                                               RSTR32a
         READ (RSTIN,err=903,IOSTAT=IOS) lx, iwrd                       RSTR32a
         lx = Iflip(lx)                                                 RSTR32a
         iwrd = Iflip(iwrd)                                             RSTR32a
      endif
C
C     The next three lines replace the iprint check                     RSTR533.R
C
      if (lx .GT. 30 ) then                                             RSTR533.R
         GOTO 88                                                        RSTR533.R
      end if                                                            RSTR533.R
C
      if (DEBUG) then
         WRITE(DEBU,'(1x,2(A,I6),A)') 'TITEL---RECORD : ',lx,
     &                                ' x ',iwrd,'  BYTES'
      end if
      if(INTEL) then
         read (rstin,err=903,end=902,iostat=ios) (fa(i+ix-1),i=1,lx)    RSTR532
      else
         READ (RSTIN,err=903,IOSTAT=IOS) RPOINTR                        RSTR32a
         I4int(1)=Iflip(I4aix(1))                                       RSTR32a
         I4int(2)=Iflip(I4aix(2))                                       RSTR32a
*        WRITE(*,*) I4int                                               RSTR32a
         READ (RSTIN,err=903,IOSTAT=IOS) (fa(i+ix-1),i=1,lx)            RSTR32a
      endif
      IF(.NOT.TERMINAL) THEN
         IF (RSTPLOT) THEN
            IF (RESTO) WRITE (RSOUT,ERR=998) lx, iwrd                   RSTR532
            IF (RESTO) WRITE (RSOUT,ERR=998) (fa(i+ix-1),i=1,lx)        RSTR532
         ENDIF
      ENDIF
      write (ptitle,'(8a8)') (fa(i+ix-1),i=1,8)
      write (ctitle,'(12a8)') (fa(i+ix-1),i=9,20)
      if (SCREEN) then
         write (*,*) ptitle
         write (*,'(A80)') ctitle
      endif
      if (DEBUG) then
         WRITE(DEBU,'(1x,A)') ptitle                                    RSTR532
         WRITE(DEBU,'(1x,A)') ctitle                                    RSTR532
      end if
C
      if (ncount>0) GOTO 86                                             RSTR32u4
C
C ----------------------------------------------------------------------
      IF (.NOT.TERMINAL) THEN                                           RSTR532
         if (PLOTO)  write (PLOUT,'(1x,A94)') ctitle                    RSTR532
C------------------------------------new in 2.3.1                       RSTR33.1
C        if (PLOTO)  write (PLOUT,'(1X,I4)')lenb-1                      RSTR532
         if (PLOTO)  write (PLOUT,'(1X,I6)')lenb-1                      RSTR532
C------------------------------------new in 2.3.1                       RSTR33.1
C        no write into RSTOUT FILE                                      RSTR532
      ENDIF                                                             RSTR532
      write(PLINF,'(1X,A94)') ctitle                                    RSTR532
C------------------------------------new in 2.3.1                       RSTR33.1
C     write(PLINF,'(1X,I4)') lenb-1                                     RSTR532u3
      write(PLINF,'(1X,I6)') lenb-1                                     RSTR33.1
C------------------------------------new in 2.3.1                       RSTR33.1
      IF(STRIPO) THEN
         WRITE (STRIPF,'(2H= ,A77)') ctitle
         WRITE (STRIPF,'(17H100 strip  fmtout)')
         WRITE (STRIPF,'(18H*100 strip  binary)')
         WRITE (STRIPF,'(5H103 0)')
      ENDIF
C
```

```
C ----------------------------------------------------------------------
C
C     ---------------Print Plotdata-Info in PLOTOUT, PLOTINF
C
C ----------------------------------------------------------------------
      IF (.NOT.TERMINAL) THEN                                           RSTR532
         if (PLOTO) then                                                RSTR532
            do i=5,lenb+3                                               RSTR532
               j = lenb+i                                               RSTR532
               If(INTEL) then
                  write(PLOUT,81) cfa(i),ia(2,j)                        RSTR5322
               else
                  write(PLOUT,81) cfa(i),Iflip(ia(2,j))                 RSTR5322
               endif
            end do                                                      RSTR532
         end if                                                         RSTR532
C        no write into RSTOUT FILE                                      RSTR532
      ENDIF                                                             RSTR532
      do i=5,lenb+3                                                     RSTR532
         j = lenb+i                                                     RSTR532
         If(INTEL) then
C           write(PLINF,81) cfa(i),ia(2,j)                              RSTR5322
            write(PLINF,181) i-5,cfa(i),ia(2,j)                         RSTR5322u3
         else
C           write(PLINF,81) cfa(i),Iflip(ia(2,j))                       RSTR5322
            write(PLINF,181) i-5,cfa(i),Iflip(ia(2,j))                  RSTR5322u3
         endif
      end do                                                            RSTR532
   81 format(1X,A8,'        ',I9,'  ')                                  RSTR532
  181 format(1X,I9,'  ',A8,'        ',I9,'  ')                          RSTR532u3
c        write to stripinput file
      icard = 1000
      IF(STRIPO) THEN
         do i=6,lenb+3
            j = lenb+i
            icard = icard + 1
            If(INTEL) then                                              RSTR5322
               if (icard.gt.1999) then                                  RSTR5322
                  if (icard.eq.2000) then                               RSTR33.0
                      WRITE(STRIPF,2013)                                RSTR33.0
                  end if                                                RSTR33.0
                  write(STRIPF,83) icard, cfa(i),ia(2,j)                RSTR5322
               else
C                 write(STRIPF,82) icard, cfa(i),ia(2,j)                RSTR5322
                  write(STRIPF,83) icard, cfa(i),ia(2,j)                RSTR33.0
               end if
            else
               IF((cfa(i)(1:3)).eq.'ext') then                          RSTR5322
                  icard = icard - 1                                     RSTR5322
                  write(STRIPF,83) icard, cfa(i),Iflip(ia(2,j))         RSTR5322
               else
                  if (icard.gt.1999) then                               RSTR5322
                     write(STRIPF,83) icard, cfa(i),Iflip(ia(2,j))      RSTR5322
                  else
                     write(STRIPF,82) icard, cfa(i),Iflip(ia(2,j))      RSTR5322
                  end if
               endif
            endif
         end do
         WRITE (STRIPF,'(5H. end)')
         WRITE (STRIPF,2001)
         GOTO 999
   82 format(I4,'  ',A8,'  ',I9,'   *')
   83 format('*',I4,'  'A8,'  ',I9,'   *')
      ENDIF

C
C ----------------------------------------------------------------------
C
   86 continue                                                          RSTR5322u4
C
      ix = ix + lx
      icomctl = ix
      if (DEBUG) then
         WRITE(DEBU,'(1x,A,I6)') 'POSITION OF COMCTL =       ',lx       RSTR532
      end if
C
C     CommonBlock /comctl/ starts at position icomstart
C
C     This Datablock is read only once
C
      if(INTEL) then
         read (rstin,err=903,end=902,iostat=ios) lx, iwrd               RSTR532
      else
         READ (RSTIN,err=903,IOSTAT=IOS) RPOINTR                        RSTR32a
         I4int(1)=Iflip(I4aix(1))                                       RSTR32a
         I4int(2)=Iflip(I4aix(2))                                       RSTR32a
*        WRITE(*,*) I4int                                               RSTR32a
         READ (RSTIN,err=903,IOSTAT=IOS) lx, iwrd                       RSTR32a
         lx = Iflip(lx)                                                 RSTR32a
         iwrd = Iflip(iwrd)                                             RSTR32a
      endif
C
      if (DEBUG) then
         WRITE(DEBU,'(1x,2(A,I6),A)') 'COMCTL--RECORD : ',lx,
     &                                ' x ',iwrd,'  BYTES'
      end if
      j = lx*iwrd/8
      if(INTEL) then
         read (rstin,err=903,end=902,iostat=ios) (fa(i+ix-1),i=1,j)     RSTR532
      else
         READ (RSTIN,err=903,IOSTAT=IOS) RPOINTR                        RSTR32a
         I4int(1)=Iflip(I4aix(1))                                       RSTR32a
         I4int(2)=Iflip(I4aix(2))                                       RSTR32a
*        WRITE(*,*) I4int                                               RSTR32a
         READ (RSTIN,err=903,IOSTAT=IOS) (fa(i+ix-1),i=1,j)             RSTR32a
      endif
C     Now write the data
      IF(.NOT.TERMINAL) THEN
         IF (RSTPLOT) THEN
            IF (RESTO) WRITE (RSOUT,ERR=998) lx, iwrd                   RSTR532
            IF (RESTO) WRITE (RSOUT,ERR=998) (fa(i+ix-1),i=1,j)         RSTR532
         ENDIF
      ENDIF
      ix = ix + J
C
C ----------------------------------------------------------------------
C
   84 continue
      icontrl = ix
      if (DEBUG) then
         WRITE(DEBU,'(1x,A,I6)') 'POSITION OF CONTRL =       ',lx       RSTR532
      end if
C
C     CommonBlock /contrl/ starts at position icontrlstart
C
C     This Datablock is read in every restart block
C
      if(INTEL) then
         read (rstin,err=903,end=902,iostat=ios) lx, iwrd               RSTR532
      else
         READ (RSTIN,err=903,IOSTAT=IOS) RPOINTR                        RSTR32a
         I4int(1)=Iflip(I4aix(1))                                       RSTR32a
         I4int(2)=Iflip(I4aix(2))                                       RSTR32a
*        WRITE(*,*) I4int                                               RSTR32a
         READ (RSTIN,err=903,IOSTAT=IOS) lx, iwrd                       RSTR32a
         lx = Iflip(lx)                                                 RSTR32a
         iwrd = Iflip(iwrd)                                             RSTR32a
      endif
C
C     -----> new in Relap 3.3                                           RSTR533.R
C     GOTO 87                                                           RSTR533.R
   88 CONTINUE                                                          RSTR533.R
C                                                                       RSTR533.R
      icontrl = ix                                                      RSTR533.R
      if (DEBUG) then                                                   RSTR533.R
         WRITE(DEBU,'(1x,A,I6)') 'POSITION OF CONTRL =       ',ix       RSTR533.R
      end if                                                            RSTR533.R
C                                                                       RSTR533.R
   87 CONTINUE                                                          RSTR533.R
C     -----> new in Relap 3.3                                           RSTR533.R
C
      if (DEBUG) then
         WRITE(DEBU,'(1x,2(A,I6),A)') 'CONTRL--RECORD : ',lx,
     &                                ' x ',iwrd,'  BYTES'
      end if
      j = lx*iwrd/8
      if(INTEL) then
         read (rstin,err=903,end=902,iostat=ios) (fa(i+ix-1),i=1,j)     RSTR532
      else
         READ (RSTIN,err=903,IOSTAT=IOS) RPOINTR                        RSTR32a
         I4int(1)=Iflip(I4aix(1))                                       RSTR32a
         I4int(2)=Iflip(I4aix(2))                                       RSTR32a
*        WRITE(*,*) I4int                                               RSTR32a
         READ (RSTIN,err=903,IOSTAT=IOS) (fa(i+ix-1),i=1,j)             RSTR32a
      endif
C     Now write the data
      IF(.NOT.TERMINAL) THEN
         IF (RSTPLOT) THEN
            IF (RESTO) WRITE (RSOUT,ERR=998) lx, iwrd                   RSTR532
            IF (RESTO) WRITE (RSOUT,ERR=998) (fa(i+ix-1),i=1,j)         RSTR532
         ENDIF
      ENDIF
C
C     check Number of restart and time of restart block
C
C     Time of restart:  timehy R8 on position 5 in contrl block
C     Number restart :  rstblcknumber1531 I4 on position 48 / 2
C
      IF(INTEL) then
C        timehy = fa(icontrl+5)                                         RSTR5322
         timehy = fa(icontrl+4)                                         RSTR32u4
C        irstbl = ia(1,icontrl+47)                                      RSTR32u4
C        done = ia(1,icontrl+48)                                        RSTR32u4
C        nmbrst = ia(2,icontrl+48)                                      RSTR5322
C        problemtype05 = ia(2,icontrl+51)                               RSTR32u4
C        problemopt611 = ia(1,icontrl+52)                               RSTR32u4
         irstbl = ia(1,icontrl+51)                                      RSTR33
         done = ia(1,icontrl+52)                                        RSTR33
         nmbrst = ia(2,icontrl+52)                                      RSTR33
         problemtype05 = ia(2,icontrl+55)                               RSTR32
         problemopt611 = ia(1,icontrl+56)                               RSTR32
      else
C        I4int(1)=Iflip(ia(2,icontrl+5))                                RSTR5322
C        I4int(2)=Iflip(ia(1,icontrl+5))                                RSTR5322
         I4int(1)=Iflip(ia(2,icontrl+5))                                RSTR32u4
         I4int(2)=Iflip(ia(1,icontrl+5))                                RSTR32u4
         timehy = r8int                                                 RSTR5322
C        nmbrst = Iflip(ia(2,icontrl+48))                               RSTR5322
C        problemtype05 = Iflip(ia(1,icontrl+53))                        RSTR32u4
C        problemopt611 = Iflip(ia(2,icontrl+53))                        RSTR32u4
         nmbrst = Iflip(ia(2,icontrl+52))                               RSTR33
         problemtype05 = Iflip(ia(1,icontrl+57))                        RSTR33
         problemopt611 = Iflip(ia(2,icontrl+57))                        RSTR33
      endif
C
      if (SCREEN) then
         write(*,'(/)')
*        write(*    ,*)'READ RESTART RECORD ',nmbrst,' TIME =',timehy
         write (*,'(1X,A,I6,A,f17.10)')'READ RESTART RECORD ',nmbrst,
     &                                ' TIME =',            timehy
      endif
      if (DEBUG) then
         write(DEBU,'(/)')
*        write(*    ,*)'READ RESTART RECORD ',nmbrst,' TIME =',timehy
         write (DEBU,'(1X,A,I6,A,f17.10)')'READ RESTART RECORD ',nmbrst,
     &                                   ' TIME =',            timehy
         WRITE(DEBU,'(1X,A,I6)          ')'Done:         ',done         REST533
         WRITE(DEBU,'(1X,A,I6)          ')'RestartBlock: ',irstbl       REST533
         WRITE(DEBU,'(1X,A,I6)          ')'Problemtype:  ',problemtype05REST533
         WRITE(DEBU,'(1X,A,I6)          ')'ProblemOpt:   ',problemopt611REST533
      endif
      IF(RSTPLOT) THEN
*         write(RSINF,*)'READ RESTART RECORD ',nmbrst,' TIME =',timehy
          write (RSINF,'(1X,A,I6,A,f17.10)')'READ RESTART RECORD ',
     &                           nmbrst,' TIME =',            timehy
      ENDIF
C
C     New in Relap 5 Mod 3.3 : ncount is not assigned anymore           RSTR33.R
      ncount = nmbrst                                                   RSTR33.R
C
      if (nmbrst.ne.ncount) then
         goto 910
      end if
C
C     now start over in fa position 1 to read in the dynamic rst-data
C
      ix = 1
      GOTO 50                                                           RSTR532
C
   85 continue
C
C     read in all other common blocks
C
      ix = ix + lx
C     j = lx*iwrd/8
      if (iwrd.eq.4) then
         j = lx * 2
      end if
      if (DEBUG) then
         WRITE(DEBU,'(1x,2(A,I6),A)') 'COMMON/DYNAMIC : ',lx,
     &                               ' x ',iwrd,'  BYTES'
      end if
      GO TO 50
C
```

```
C----------------------------------------PLOTREC-------------------------
C
C     READ PLOTREC RECORD
C
   90 continue
C               Length of PlotData Record                              RESTLH9
      if (lx.ne.lenc) then
         GOTO 907
      end if
      if (lcompr) then
         if (iwrd.eq.4) then
            GOTO 908
         end if
      else
         if (iwrd.eq.4) then
            GOTO 909
         end if
      end if
C     Data is in fa array from ix - ix+lx (lx=lenc)
      if (lcompr) then
         IF(.NOT.TERMINAL) THEN
         if(INTEL) then
         if (PLOTO) write (PLOUT,'(1X,A,e13.7)')'TIME = ',ffa(2*ix+1)   RSTR5322
         if (PLOTO) write (PLOUT,'(6(e11.5,1X))',err=900) (ffa(2*ix+1+  RSTR5322
     &                    kk),kk=1,lenb-2)                              RSTR5322
         else
            if(PLOTO) then
C              iia(2*ix+1)=ffa(2*ix+1)                                  RSTR5322
               I4int(2)=Iflip(iia(2*ix+1))                              RSTR5322
               write (PLOUT,'(1X,A,e13.7)')'TIME = ',R4int(2)           RSTR5322
               do kk=1,lenb-2                                           RSTR5322
                  iia(2*ix+1+kk)=Iflip(iia(2*ix+1+kk))                  RSTR5322
               end do                                                   RSTR5322
               write (PLOUT,'(6(e11.5,1X))',err=900)                    RSTR5322
     &               (ffa(2*ix+1+kk),kk=1,lenb-2)                       RSTR5322
            endif
         endif
C        no write into RSTOUT FILE !!
         ENDIF
C        alternative: Use of unsqoz function:
C        fa array of length lenc will be expanded by factor of two
C        call unsqoz (fa(ix+1),lenb)
C        write (*,*) fa(ix+1),fa(ix+2),fa(ix+3)
C        if (PLOTO) write (PLOUT,'(1X,A,e13.7)')'TIME = ',fa(ix+1)
C        if (PLOTO) write (PLOUT,'(6(e11.5,1X))',err=900) (fa(ix+1+
C    &                    kk),kk=1,lenb-2)
      else
C        not compressed data, so read in PLOTOUT directly
         IF(.NOT.TERMINAL) THEN
         if(INTEL) then
         if (PLOTO) write (PLOUT,'(1X,A,e13.7)')'TIME = ',fa(ix+1)      RSTR5322
         if (PLOTO) write (PLOUT,'(6(e11.5,1X))',err=900) (fa(ix+1+i),  RSTR5322
     &                     i = 1,lenc-1)                                RSTR5322
         else
            if(PLOTO) then                                              RSTR5322
               R8aix = fa(ix+1)                                         RSTR5322
               I4int(1)=Iflip(I4aix(2))                                 RSTR5322
               I4int(2)=Iflip(I4aix(1))                                 RSTR5322
               write (PLOUT,'(1X,A,e13.7)')'TIME = ',R8int              RSTR5322
               do i=1,lenc-1                                            RSTR5322
                  R8aix = fa(ix+1+i)                                    RSTR5322
                  I4int(1)=Iflip(I4aix(2))                              RSTR5322
                  I4int(2)=Iflip(I4aix(1))                              RSTR5322
                  fa(ix+1+i)=R8int                                      RSTR5322
               end do                                                   RSTR5322
               write (PLOUT,'(6(e11.5,1X))',err=900)                    RSTR5322
     &               (fa(ix+1+i),i=1,lenc-1)                            RSTR5322
            endif                                                       RSTR5322
         endif
C        no write into RSTOUT FILE !!
         ENDIF
      end if
      ix = 1
      IZA = IZA + 1
      PLOTI = .TRUE.
      GO TO 50
C
C---------------------------------------------------------FEHLER
C
  100 WRITE (IOUT,2036)                                                 RSTR532u1
 2036 FORMAT (//' ******** WRONG INFORMATION IN HEADER OF DATA '
     &   ,'RECORD.')
      GO TO 999
C
C---------------------------------------------------------WEITER
```

```
C
  900 write (IOUT,2200)                                                 RSTR532u1
 2200 FORMAT(//' ****** Write ERROR on PLOTDATA INFORMATION FILE !!')
      GO TO 1000
C
  901 write (IOUT,2201)                                                 RSTR532u1
 2201 FORMAT(//' ****** Write ERROR on PLOTDATA FILE !!')
      GO TO 1000
C
  902 WRITE (IOUT,2202) fnames(1)(1:15)                                 RSTR532u1
 2202 FORMAT (//' ****** EOF FOUND ON READ OF '                         RSTR532u3
     &          ,'INPUT FILE: ',A15,'...')
      GO TO 999
C
  903 WRITE (IOUT,2203) IOS                                             RSTR532u1
 2203 FORMAT (/,' ****** ERROR ON READ OF INPUT FILE. '                 RSTR532u1
     &  ,' IOS = ',I4)
      GO TO 1000
C
  904 WRITE (IOUT,2204) lxa,lxn                                         RSTR532u1
 2204 FORMAT (1X,'****** PLOTINF RECORD LENGTH ERROR: '
     &  ,I4, ' # ',I4)
      GO TO 1000
C
  906 WRITE (IOUT,2206) lenb,lenc                                       RSTR532u1
 2206 FORMAT (1X,'****** PLOTDATA RECORD LENGTH ERROR: '
     &  ,I4, ' # ',I4)
      GO TO 1000
  907 WRITE (IOUT,2206) lenc,J                                          RSTR532u1
      GO TO 1000
  908 WRITE (IOUT,2208) iwrd                                            RSTR532u1
 2208 FORMAT (1X,'****** COMPRESSED DATA WORD LENGTH: 'I4)
      GO TO 1000
  909 WRITE (IOUT,2209) iwrd                                            RSTR532u1
 2209 FORMAT (1X,'****** UNCOMPRESSED DATA WORD LENGTH: 'I4)
      GO TO 1000
  910 WRITE (IOUT,2210) nmbrst, ncount                                  RSTR532u1
 2210 FORMAT (1X,'****** ERROR READING RESTART BLOCK NUMBER:  No:'
     &  ,I4, ' #   ',I4)
      GO TO 1000
  997 write (IOUT,2299)                                                 RSTR532u1
 2299 FORMAT(//' ****** Write ERROR on RESTART INFORMATION FILE !!')
      GO TO 1000
C
  998 WRITE (PLOUT,2300)
 2300 FORMAT (//' ****** Write ERROR on RESTART FILE !!')
C
  999 CONTINUE
      IF (RSTPLOT) THEN
         IF (.NOT.RESTI) WRITE (RSINF,2290)
         IF (.NOT.PLOTI) WRITE (PLOUT,2291)
         IF (SCREEN) THEN
            IF (.NOT.RESTI) WRITE (*,2290)
            IF (.NOT.PLOTI) WRITE (*,2291)
         END IF
      ELSE
         IF (.NOT.PLOTI) WRITE (PLINF,2295)
      ENDIF
      do i = 1,nfil-2
         close (iun(i))
      enddo
      IF (SCREEN) THEN
         IF (RESTI) WRITE (*,2293)
         IF (PLOTI.or.STRIPO) WRITE (*,2294)
      ENDIF
      GO TO 1000
 2290 FORMAT(//' ****** NO RESTARTS ON RESTART/PLOT INPUT FILE !!')
 2291 FORMAT(//' ****** NO PLOT DATA ON RESTART/PLOT INPUT FILE !!')
 2292 FORMAT(//' ****** NO PLOT DATA ON STRIP FILE !!')
 2293 FORMAT('  ****** RESTARTS PROCESSED WELL !!')
 2294 FORMAT('  ****** PLOTDATA PROCESSED WELL !!')
 2295 FORMAT(' ****** NO PLOT DATA PROCESSED, STRIP CASE !!')           RSTR532u1
C ========================================================================
C9000 write (*,*) 'PROGRAM RESTR532G Version 2.24'                      RSTR32u4
C9000 write (*,*) 'PROGRAM RSTR33B Version 2.3.1'                       RSTR33.1      write
(*,*)
 9000 write (*,*) 'PROGRAM RSTR533 Version 2.4.1 '                      RSTR33R1
      write (*,*)
      write (*,*)' Options:                              Defaults:'
      write (*,*)'   -ri   restart/strip_input_file         RESTINP'
      write (*,*)'   -ro   restart_output w/o plotdata      RESTOUT'
      write (*,*)'   -or   restart_info_file                RESTINF'
      write (*,*)'   -po   plot_parm & data_file            PLOTOUT'
      write (*,*)'   -op   plot_parameter_info_file         PLOTINF'
      write (*,*)'   -so   strip inputfile for RELAP        STRIPINP'
      write (*,*)'   -d    debug data in file               DEBUG'
      write (*,*)'   -r    write restart output file'
      write (*,*)'   -s    write strip input file'
      write (*,*)'   -p    write plotdata file'
      write (*,*)'   -no   no screen output'
      write (*,*)'   -h/?  HELP'
      write (*,*)'    w/o -p and/or -r the default is:'
      write (*,*)'    only RESTINF and/or PLOTINF files will be written'
      write (*,*)'                -----> try again !'
      write (*,*)'    For AIX Restarts RESTOUT will not be processed!'
C
C =================================================================
 1000 Continue                                                          RESTLH9
C1000 RETURN                                                            RESTLH9
C     Changed for Lahey F90                                             RESTLH9
      END
C-----------------------------------------------------------------C
C                                                                 C
C                            CDATE                                C
C                                                                 C
C-----------------------------------------------------------------C
       SUBROUTINE CDATE (CWORD)
C  RETURNS DATE IN WORD IN CHARACTER TYPE IN FORM MM/DD/YY
C
      character*8 cword
C
C     The following additions are due to LF90                           RESTLH9
      CHARACTER*10 date, time, zone                                     RESTLH9
      INTEGER dt(8), IM, ID, IY                                         RESTLH9
C     --------------------------------------
C     CALL GETDAT(IY,IM,ID)                                             RESTLH9
C     Changes due to Lahey LF90                                         RESTLH9
      call date_and_time (date, time, zone, dt)                         RESTLH9
C     The following additions are due to LF90                           RESTLH9
      IM=dt(2)                                                          RESTLH9
      ID=dt(3)                                                          RESTLH9
      IY=dt(1)                                                          RESTLH9
*     write (*,*) IY,IM,ID                                              RESTLH9
      if (IY>1999) then                                                 RESTLH9
         IY = IY - 2000                                                 RESTLH9
      else                                                              RESTLH9
         IY = IY - 1900                                                 RESTLH9
      end if                                                            RESTLH9
C     --------------------------------------
      WRITE(CWORD,100)IM,ID,IY                                          RESTLH9
100   FORMAT(I2,'/',I2,'/',I2)
C     --------------------------------------
      if (IM.lt.10) write(CWORD(1:1),'(a1)') '0'
      if (ID.lt.10) write(CWORD(4:4),'(a1)') '0'
      if (IY.lt.10) write(CWORD(7:7),'(a1)') '0'
C     --------------------------------------
      RETURN
      END
C-----------------------------------------------------------------C
C                                                                 C
C                            NARGS                                C  RESTLH9
C                for Lahey F90 Version 4.0a Compiler              C
C-----------------------------------------------------------------C
      INTEGER FUNCTION NARGS()
C     RETURNS Number of Arguments including Programm-Name
C
      INTEGER I, J, PARCNT
      LOGICAL FOUND
      CHARACTER(len = 512) acl, acm
      PARCNT=0
      J = 1
      FOUND = .FALSE.
C     Get Command Line String and Determine Length
      CALL getcl(acl)
      acm = TRIM(acl)
C     acm = 'ee ee ee ee'
      I = NEWLEN(acm)
      do while (J <= I)
         if ((acm(J:J).ne.' ').and..NOT.(FOUND)) then
            FOUND = .TRUE.
C           WRITE(*,*)'J1+ = ', J
            PARCNT=PARCNT+1
            J = J + 1
            if (acm(J:J).eq.' ') then
               FOUND = .FALSE.
C              WRITE(*,*)'J2a- = ', J
               J = J + 1
            else
               do WHILE (FOUND)
                  if ((acm(J:J).ne.' ') .AND. J < I) then
```

```
                     FOUND = .TRUE.
C                    WRITE(*,*)'J2a+ = ', J
                  else
                     FOUND = .FALSE.
C                    WRITE(*,*)'J2a- = ', J
                  END if
                  J = J + 1
               end do
            end if
         else
            FOUND = .FALSE.
C           WRITE(*,*)'J1- = ', J
            J = J + 1
         end if
      end do
C     write(*,*)'CommandLine = ',acm,' Length = ',I, ' NARGS = ',PARCNT
      NARGS=PARCNT + 1
C                    | to be compatible with Powerstation Fortran
C
      RETURN
      END
C-----------------------------------------------------------------C
C                                                                 C
C                            GETARG                               C  RESTLH9
C                                                                 C
C-----------------------------------------------------------------C
      SUBROUTINE GETARG(NO,ARGUM,ST)

C     RETURNS Argument Number NO in String ARGUM and Length of Argument
C     Usage:
C     Call getarg(NO,ARGUM,ST)
C                 n     : Integer*2; input: position of desired argument
C                                    0 : The command itself
C                 ARGUM : Character*(*); output: Argument
C                 ST    : Integer*2; output: Completion status
C                         = -1 if n < 0 or n > number of args
C
      INTEGER ST
      INTEGER*2 iarg, Ilen, ipos, jpos, iparlen, NO, ipar
      CHARACTER*50 ARGUM
      CHARACTER(len = 512) acl, acm
      LOGICAL FOUND, START
      FOUND = .FALSE.
      START = .FALSE.
      ST = 1
      ipar = 0
      jpos = 0
      iparlen = 0
      ARGUM = ' '
      iarg = nargs()
      CALL getcl(acl)
      acm = TRIM(acl)
      Ilen = NEWLEN(acm)
      if (NO==0) then
         ARGUM = 'RESTaix.EXE'
         ST = 7
         GOTO 20
      end if
      if ((NO<0).or.(NO>(iarg-1))) then
         ST = -1
         GOTO 20
      end if
C
      do 10 ipos = 1, Ilen+1
         if (acm(ipos:ipos).ne.' ') then
            if (.NOT.(FOUND)) ipar = ipar + 1
            FOUND = .TRUE.
            if (START) then
            else
               if (ipar==NO) START = .TRUE.
               jpos = ipos
            end if
            if (ipar==NO) START = .TRUE.
            iparlen = iparlen+1
         else
            FOUND = .FALSE.
            if (ipar==NO) GOTO 15
            iparlen = 0
         endif
C        write (*,*) 'ip=',ipar,' j=',jpos,' i=',ipos,' len=',iparlen
   10 continue
   15 continue
      st = iparlen
      ARGUM = acm(jpos:jpos+iparlen-1)
C     write (*,*) 'Parameter No:',NO,' Length=',st
```

```
C     write (*,*) 'Parameter   :',ARGUM
   20 CONTINUE
      RETURN
      END
C-----------------------------------------------------------------C
C                                                                 C
C                              NEWLEN                             C  RESTLH9
C                                                                 C
C-----------------------------------------------------------------C
      Integer Function NEWLEN(Str)

C     RETURNS Length of String indep. of defined Character Length
C     Max Stringlenth = ?
C
      CHARACTER(LEN = *) :: str
      Integer ILEN, J
      ILEN = LEN(STR)
C
      do 10  J = ILEN, 0 , -1
             if (STR(J:J).eq.' ') then
             else
                goto 20
             END if
   10 Continue
   20 Continue
C
      NEWLEN = J
      RETURN
      END
C-----------------------------------------------------------------C
C                                                                 C
C                              IREC                               C  RESTLH9
C                                                                 C
C-----------------------------------------------------------------C
      Integer Function IREC(I,iMESS)

C     RETURNS Length of Record for reading formatted Data
C     written with MS-FORTRAN PowerStation 1.0
C
      INTEGER*1 I
      INTEGER BYTES,iMESS
C
      IF (I==-127) THEN
         BYTES = 128
      else
C             < -127  (-128) or < 0
         IF (I<0) THEN
            BYTES = 128
            IF (iMESS>0) write (iMESS,*) 'WARNING! RECORD Pointer: ',I,
     &                                   ' set to 128 !'
            IF (iMESS<0) write (*,*)     'WARNING! RECORD Pointer: ',I,
     &                                   ' set to 128 !'
          ELSE
            BYTES = I
          END IF
      END IF
      IREC=BYTES
      RETURN
      END
C-----------------------------------------------------------------C
C                                                                 C
C                              IFLIP                              C  RESTaix
C                                                                 C
C-----------------------------------------------------------------C
      Integer Function IFLIP(I4)
C
C     RETURNS Integer*4 for Intel
C     AIX writes Bytes of Integer*2 or 4 in opposite order
C     so they must be swept. Example
C     new function intel-integer*4=iflip(aix-integer*4)
C
C     The same holds true for conversion of R4:
C         first equivalence r4 to I4
C         then swap bytes with iflip function
C
C     Conversion of Real*8 additionally needs swapping
C     of the two I4 integers:
C     Example:
C                  REAL*8    R8aix
C                  REAL*8    R8int
C                  INTEGER*4 I4aix(2), I4int(2)
C                  EQUIVALENCE (I4aix(1), R8aix)
C                  EQUIVALENCE (I4int(1), R8int)
C
C                  R8aix=Real8Number
```

```
C                  I4int(2)=iflip(I4aix(1))
C                  I4int(1)=iflip(I4aix(2))
C                  Real8Number=R8int
C
      INTEGER*1 I1int(4), I1aix(4)
      INTEGER*4 I4aix, I4int,I4
      EQUIVALENCE (I1aix(1), I4aix)
      EQUIVALENCE (I1int(1), I4int)
       I4aix=I4
       I1int(1)=I1aix(4)
       I1int(2)=I1aix(3)
       I1int(3)=I1aix(2)
       I1int(4)=I1aix(1)
      IFLIP=I4int
      RETURN
      END
C-------------------------------------------------------------------C
C                                                                   C
C                            UNSQOZ                                 C  RSTR532
C                                                                   C
C-------------------------------------------------------------------C

       subroutine unsqoz (a,num)
       implicit none
c       Unsqueeze one packed word (obtained from sqoz) into two floating
c       point words.
c       On 32 bit machines, simply convert real*8 words to real*4 words.
c       Locals
       integer i, i1, n, num
       real*8 a(num)
       real*8 da
       real*4 fa(2)
       equivalence (da,fa(1))
       n = iand(num+1,not(1))
       i1 = ishft(n,-1)
       write (*,*)n,i1
       do 10 i = i1,1,-1
         da = a(i)
         a(n) = fa(2)
         a(n-1) = fa(1)
         n = n - 2
   10  continue
       Return
end
```

NRC FORM 335 (9-2004) NRCMD 3.7 **U.S. NUCLEAR REGULATORY COMMISSION** **BIBLIOGRAPHIC DATA SHEET** *(See instructions on the reverse)*	1. REPORT NUMBER **(Assigned by NRC, Add Vol., Supp., Rev., and Addendum Numbers, if any.)** NUREG/IA-0433
2. TITLE AND SUBTITLE RELAP5/MOD3.3 RELEASE Pre & Postprocessor	3. DATE REPORT PUBLISHED MONTH: December YEAR: 2013
	4. FIN OR GRANT NUMBER
5. AUTHOR(S) Dr. W. Tietsch	6. TYPE OF REPORT Technical
	7. PERIOD COVERED *(Inclusive Dates)*

8. PERFORMING ORGANIZATION - NAME AND ADDRESS *(If NRC, provide Division, Office or Region, U.S. Nuclear Regulatory Commission, and mailing address; if contractor, provide name and mailing address.)*
Westinghouse Electric Germany GmbH
Dudenstrasse 44
68167 Mannheim
Germany

9. SPONSORING ORGANIZATION - NAME AND ADDRESS *(If NRC, type "Same as above"; if contractor, provide NRC Division, Office or Region, U.S. Nuclear Regulatory Commission, and mailing address.)*
Division of Systems Analysis
Office of Nuclear Regulatory Research
U.S. Nuclear Regulatory Commission
Washington, DC 20555-0001

10. SUPPLEMENTARY NOTES
A. Calvo, NRC Project Manager

11. ABSTRACT *(200 words or less)*
Since the introduction of the 32-Bit Windows Operating System technology for Personal Computers with INTEL CPU architecture Relap has been migrated from mainframe comput-ers and UNIX workstations to the desktop personal computers. Due to the general structure of Relap the user interface however did not change in the course of this migration. Relap running on PCs still is command line driven and does not take advantage of the benefits of the windows environments. Inputs to the program and outputs have the same structure as the UNIX or mainframe versions. Therefore the classic way of performing analyses with Re-lap which consist of several, mostly iterative and consecutive steps did not change with PC installations of the program.

Westinghouse Reaktor mid of the 90th developed several tools running on the PC to support the analysts retrieving and documenting results of Relap analyses. Based on these tools and using the power of the object oriented technology and third party functionality for the graphics generation a Graphical Users Interface has been developed by Westinghouse Reaktor which integrates all steps of a typical Relap analysis process. This Pre & Postprocessor has been designed to run in all known Windows environments (Win-dows95/98, ME, NT4, 2000 and XP). It has the feel and look of typical Windows software and an intuitive users interface which only requires a minimum of training. The actual version 3.3.0 of the Pre & Postproces-sor cooperates with the Intel Relap5Mod3.3Release version.

12. KEY WORDS/DESCRIPTORS *(List words or phrases that will assist researchers in locating the report.)*	13. AVAILABILITY STATEMENT unlimited
ATHLET Westinghouse Electric Germany GmbH Pre-processor Relap5Mod3.3 Postprocessor U.S. Nuclear Regulatory Commission SNAP	14. SECURITY CLASSIFICATION *(This Page)* unclassified *(This Report)* unclassified
	15. NUMBER OF PAGES
	16. PRICE

NUREG/IA-0433 RELAP5/MOD3.3 RELEASE Pre & Postprocessor December 2013