NUREG/IA-0514

International Agreement Report

Customization of XTV Graphics Output in TRACE v5.0 Patches 5, 4 & 3

Prepared by:
O. Zerkak and I. Clifford

Paul Scherrer Institut
5232 Villigen PSI
Switzerland

K. Tien, Project Manager

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

Manuscript Completed: February 2019
Date Published: August 2019

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

AVAILABILITY OF REFERENCE MATERIALS IN NRC PUBLICATIONS

NRC Reference Material

As of November 1999, you may electronically access NUREG-series publications and other NRC records at NRC's Library at www.nrc.gov/reading-rm.html. Publicly released records include, to name a few, NUREG-series publications; *Federal Register* notices; applicant, licensee, and vendor documents and correspondence; NRC correspondence and internal memoranda; bulletins and information notices; inspection and investigative reports; licensee event reports; and Commission papers and their attachments.

NRC publications in the NUREG series, NRC regulations, and Title 10, "Energy," in the *Code of Federal Regulations* may also be purchased from one of these two sources.

1. **The Superintendent of Documents**
   U.S. Government Publishing Office
   Mail Stop IDCC
   Washington, DC 20402-0001
   Internet: bookstore.gpo.gov
   Telephone: (202) 512-1800
   Fax: (202) 512-2104

2. **The National Technical Information Service**
   5301 Shawnee Road
   Alexandria, VA 22312-0002
   www.ntis.gov
   1-800-553-6847 or, locally, (703) 605-6000

A single copy of each NRC draft report for comment is available free, to the extent of supply, upon written request as follows:

Address: **U.S. Nuclear Regulatory Commission**
Office of Administration
Multimedia, Graphics, and Storage & Distribution Branch
Washington, DC 20555-0001
E-mail: distribution.resource@nrc.gov
Facsimile: (301) 415-2289

Some publications in the NUREG series that are posted at NRC's Web site address www.nrc.gov/reading-rm/doc-collections/nuregs are updated periodically and may differ from the last printed version. Although references to material found on a Web site bear the date the material was accessed, the material available on the date cited may subsequently be removed from the site.

Non-NRC Reference Material

Documents available from public and special technical libraries include all open literature items, such as books, journal articles, transactions, *Federal Register* notices, Federal and State legislation, and congressional reports. Such documents as theses, dissertations, foreign reports and translations, and non-NRC conference proceedings may be purchased from their sponsoring organization.

Copies of industry codes and standards used in a substantive manner in the NRC regulatory process are maintained at—

The NRC Technical Library
Two White Flint North
11545 Rockville Pike
Rockville, MD 20852-2738

These standards are available in the library for reference use by the public. Codes and standards are usually copyrighted and may be purchased from the originating organization or, if they are American National Standards, from—

American National Standards Institute
11 West 42nd Street
New York, NY 10036-8002
www.ansi.org
(212) 642-4900

Legally binding regulatory requirements are stated only in laws; NRC regulations; licenses, including technical specifications; or orders, not in NUREG-series publications. The views expressed in contractor prepared publications in this series are not necessarily those of the NRC.

The NUREG series comprises (1) technical and administrative reports and books prepared by the staff (NUREG–XXXX) or agency contractors (NUREG/CR–XXXX), (2) proceedings of conferences (NUREG/CP–XXXX), (3) reports resulting from international agreements (NUREG/IA–XXXX), (4) brochures (NUREG/BR–XXXX), and (5) compilations of legal decisions and orders of the Commission and Atomic and Safety Licensing Boards and of Directors' decisions under Section 2.206 of NRC's regulations (NUREG–0750).

NUREG/IA- 0514

International Agreement Report

Customization of XTV Graphics Output in TRACE v5.0 Patches 5, 4 & 3

Prepared by:
O. Zerkak and I. Clifford

Paul Scherrer Institut
5232 Villigen PSI
Switzerland

K. Tien, Project Manager

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

Manuscript Completed: February 2019
Date Published: August 2019

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

This report describes a developmental patch applicable to TRACE v5.0 that provides the user with additional flexibility in using the existing option graphlevel to specify the list of variables to be included in the graphics output file (XTV). This option is activated from the TRACE input model using newly added Namelist option lists graphCustId and graphCustVar.

This capability, referred to as the XTV-Customize option, is particularly indicated to optimize storage space and/or post-processing memory space for production studies that require large input models, and for research studies that include sensitivity analysis or uncertainty quantification involving very large number of TRACE simulations of a same input file.

The XTV-Customize option is provided in the form of three separate code installation patches applicable to versions v5.0p5, v5.0p4 and v5.0p3 of the TRACE code, respectively.

The developed patch has been verified on the LCLRS Linux 64-bit servers of PSI for a representative sample of test cases, and for the two different TRACE output graphics file formats, namely XTV and DMX. This patch should nevertheless be still considered as developmental. Some recommendations for improvement of the implementation are provided at the end of the report.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

EXECUTIVE SUMMARY

This report describes a developmental patch applicable to TRACE v5.0 that provides the user with additional flexibility in using the existing option graphlevel to specify the list of variables to be included in the graphics output file (XTV). This option is activated from the TRACE input model using newly added Namelist option lists graphCustId and graphCustVar.

This capability, referred to as the XTV-Customize option, is particularly indicated to optimize storage space and/or post-processing memory space for production studies that require large input models, and for research studies that include sensitivity analysis or uncertainty quantification involving very large number of TRACE simulations of a same input file.

The XTV-Customize option is provided in the form of three separate code installation patches applicable to versions v5.0p5, v5.0p4 and v5.0p3 of the TRACE code, respectively.

The developed patch has been verified on the LCLRS Linux 64-bit servers of PSI for a representative sample of test cases, and for the two different TRACE output graphics file formats, namely XTV and DMX. This patch should nevertheless be still considered as developmental. Some recommendations for improvement of the implementation are provided at the end of the report.

ACKNOWLEDGMENTS

This work was partly funded by the Swiss Federal Nuclear Safety Inspectorate ENSI (Eidgenössisches Nuklearsicherheitsinspektorat) and the Swiss Federal Office of Energy (Bundesamt für Energie).

ABBREVIATIONS AND ACRONYMS

ANS	American National Standard
DMX	Demultiplexed Graphics Output File
ENSI	Swiss Federal Nuclear Safety Inspectorate (Eidgenössisches Nuklearsicherheitsinspektorat)
ID	Identifier
KIT	Karlsruhe Institute of Technology
LOCA	Loss Of Coolant Accident
PDF	Probability Density Function
PSI	Paul Scherrer Institut
TPR	Standard Output Dump File
USNRC	U.S. Nuclear Regulatory Commission
XTV	Graphics Output File

1 INTRODUCTION

The current trend of using TRACE for detailed “full-core” transient analyses to determine accurate fuel rod failure statistics, such as for example the study of the loss-of-coolant accident (LOCA), is posing increasingly challenging space and memory requirements. Thus, a better control of the content (and therefore size) of the output files produced by TRACE is desirable.

For large reactor models where each fuel assembly is modelled explicitly and can include several different types of fuel rods, the output produced with the Namelist options graphlevel="full" and graphlevel="limited" may be too large for effective post-processing since it can contain many variables that are not necessarily relevant to the study at hand.

The issue is further exacerbated when sensitivity analysis or uncertainty propagation is to be included in the study and would therefore require the production and storage of a large number of graphics output files (XTV).

This limitation in TRACE can be overcome by providing the user with additional flexibility in using the existing option graphlevel to specify the list of variables to be included in the XTV file. This report describes a developmental patch that allows for this in TRACE v5.0 patches 5, 4 and 3 ([1], [2], [3]).

2 DESCRIPTION

Two new Namelist option lists have been added:

- The option graphCustVar receives a 1-d array of character strings, containing a list of variable names (a selection list) for inclusion in the XTV graphics output. Note that these names apply to dynamic variables only, which include the so-called general problem variables (e.g. “delt”, “dprmax”) and all the variables related to components, except for the static variables, namely “vol” which is not affected by the patch. All other variables, namely the control block, signal and trip variables (e.g. trip1000) are unaffected by the patch.

- The option graphCustId receives a 1-d array of integers, containing a list of component identifiers to apply the XTV variable selection list to. No identifier (ID) is required for general problem variables.

Consider the sample Namelist below:

```
&INOPTS
graphlevel   = "minimal",
graphCustId  = 10,20,
graphCustVar = "dprmax","pn","alpn","tln","gamn"
&END
```

The behaviour of the patched version of TRACE based on the input above is as follows

- The graphics output option "minimal" is selected. This behaviour is unchanged from the standard TRACE version.

- The XTV graphics file will contain the general problem variable "dprmax", which is usually possible in standard TRACE only by using graphlevel option "full".

- For components 10 and 20 of the input file, all output variables that belong to graphlevel categories higher than "minimal" are matched against the content of the option list graphCustVar before being added to the XTV graphics file. The result is that the XTV graphics output file will contain variables pn, alpn, tln, and gamn for components 10 and 20, although the graphlevel option "minimal" should not allow for this in the standard version of TRACE.

Some additional comments

- The resulting XTV graphics file can be post-processed using AptPlot as before, or can be converted to a DMX file using the demultiplexing tool (xtv2dmx) as before.

- If graphlevel is set to "full", graphCustVar and graphCustId will have no effect since full graphics output is requested.

- If graphCustVar is omitted or contains only empty strings, no filtering of the output is done and the content of the graphics output will be determined uniquely by graphlevel, as in standard TRACE.

- If a variable listed in graphCustVar is not consistent with the set of model options of the input file, the variable will be simply ignored and no warning message will be prompted.

- Any general problem variable specified in graphCustVar requires no specific ID in graphCustId to be written in the XTV graphics file.

- If graphCustId is omitted or empty, the filtering of the output will be done only for the eventual general problem variable specified in graphCustVar, while the content of the graphics output for components will be determined uniquely by graphlevel, as in standard TRACE.

- If graphCustId includes some IDs that do not match any component of the input file, filtering will be done only for matching IDs and no warning message will be prompted.

- For components including several sub-components (e.g. heat structures in a CHAN component), the IDs of the sub-components of interest shall be provided individually in graphCustId. For instance, to include in the XTV file a variable related to heat structure 11002 that is spawned by CHAN component 11, the ID 11002 should be listed in graphCustId.

- Variable names in the list graphCustVar are case-sensitive and wildcards cannot be used. If there is an error in a variable name, it will simply be ignored and not written.

- In case of a restart model, if graphCustVar and graphCustId are not defined in the new input file, the previous contents for these two variables from the restart file are not considered, and the new run assumes default (i.e. empty) content for the two variables.

3 IMPLEMENTATION

The XTV-Customize option described in section 2 is implemented as a new Fortran 90 module <XtvCustom> in source file [src/XtvCustomM.f90] and required limited changes to 6 to 7 TRACE source code files, depending on the code version.

The TRACE versions v5.0p4 and v5.0p3 required modifications to the source files [src/NamlistDatM.f90], [src/NamlistInputM.f90], [src/NamlistM.f90], [src/XtvCompsM.f90], [src/XtvDumpM.f90] and [src/XtvSetupM.f90].

As for version v5.0p5, a minor modification to the file [src/InfoOutM.f90] was also necessary in addition to the aforementioned changes.

3.1 Modifications to the Source Code

An illustration of the implementation of the XTV-Customize option in the program flowchart of TRACE v5.0p5 is shown in Figure 3-1. The structure of the implementation is very similar for the other code versions.

The followed approach is to assume a full graphics output in the relevant component-specific graphics file setup subroutines of module <XtvComps> in file [src/XtvCompsM.f90] but to filter out any unrequested dynamic variables of the XTV graphics output file.

First, the activation of the filtering mode is handled using a flag (graphFltFlag) that is set “on” or “off”, once and for all at the code execution through a call to subroutine {InitCustomGraphLevel} (module <XtvCustom>) from subroutine {namlst} (module <Namlist>). The flag is set to “on” if graphlevel option is different from “full” and both graphCustId and graphCustVar are non-empty.

The filtering is done by a call to subroutine {XtvVarFilterOut} of module <XtvCustomM> (in source file [src/XtvCustomM.f90]) from the subroutines {AddVectorR1Var}, {AddVectorI1Var}, {AddVectorR2Var}, {AddVectorR3Var}, {AddVectorI3Var} of module <XtvSetup> (in file [src/XtvSetupM.f90]).

Note that the maximum number of component IDs and of variable names that can be supplied in Namelist option lists graphCustId and graphCustVar are specified in source file [src/XtvCustomM.f90]. The limits are set to 1024 items per list but this can be changed easily in the source file.

The filtering control logic in subroutine {XtvVarFilterOut} requires the lists of XTV variables available for the 3 possible graphlevel options, namely “minimal”, “limited”, and “full”, in addition to the content of Namelist options graphCustId and graphCustVar. These 3 lists are prepared for each TRACE component of the input file by subroutine {XtvVarListsPrep} of module <XtvCustom>.

In order to be consistent with the set of Namelist and component model options specified in the input file, the subroutine {XtvVarListsPrep} replicates the control logic used in the relevant component-specific graphics file setup subroutines of module <XtvComps>.

Moreover, as can be noted in legend items /J/ and /K/ of Figure 3-1, few minor modifications were made to subroutines {LoadTraceGName} and {LoadTraceLName} of module <TraceSpeciesData> (source file [src/TraceSpeciesDataM.f90]), and to subroutine {InitLabels} of module <EngUnits> (source file [src/EngUnitsM.f90]). This was necessary to avoid a few variable extraction errors when using the AptPlot tool to post-process the XTV graphics output file. The affected XTV variables were: **a)** the conditional variables with label prefixes "Gas " or "Liq " that are associated with Namelist option ITRACE (enables the model for gaseous and liquid trace species transport in TRACE), and **b)** the conditional variable "ecr50_46" associated with option NMWRX for the metal-water reaction model in TRACE components 'Chan' and 'HtStr'.

The variable extraction problem is independent of the XTV-Customize option, and was traced to an extraction error in AptPlot when utilized in batch mode. The presence of a blank " " or underscore symbol "_" in the variable name is apparently not compatible with the command interpreter language of AptPlot. The issue has been observed for several releases of TRACE, namely v5.0p5 [1], v5.0p4 [2] and v5.0p3 [3], and bug reports to the code developers have been prepared for the latest v5.0p5 [4].

In the context of this work, these issues have been simply eliminated by removing the outstanding characters in the XTV variable names (namely, variable "ecr50_46" changed to "ecr50u46" and blank character removed in variable prefixes "Gas" and "Liq ").

Furthermore, another modification proved necessary to avoid an XTV post-processing error with AptPlot when the XTV-Customize option is used concurrent with graphlevel option "minimal". For cases where a thermal-hydraulic component was defined in the XTV file although without any graphics output variable included, an AptPlot execution error was experienced when attempting to post-process the file. Note that the possibility of a corruption of the XTV file for such case could be ruled out by the successful extraction of variable using the in-house xtvReader tool [4]. No bug report has been submitted however, since AptPlot is not intended for such configuration of the XTV file.

In the context of this work, the aforementioned issue has been addressed by simply including at least one variable per component in the XTV file when graphlevel option "minimal" is selected, and this independently of the content of option lists graphCustId and graphCustVar. These variables were selected arbitrarily, namely, "vol" for any fluid component where applicable, and scalar variables "eTransi", "qrad", "rpower" and "tramax" for components 'HtStr', 'Radenc', 'HtStrC' and 'Power', respectively. For 'Contan' components, the added variable depends on the compartment type and associated model options. This fix is implemented in subroutine {XtvVarFilterOut} of the additional source file [src/XtvCompsM.f90].

Finally, it should be mentioned that the XTV-Customize option has not been designed at this stage for consistency with non-default values of the Namelist option XtvAppend. This is an option that allows for appending graphics output to an old XTV file (see the user's manual for more details, e.g. [1] for TRACE version v5.0p5). The XTV-Customize option has in fact been developed and tested only for the cases where the simulation creates a new XTV file, namely for the default value XtvAppend=0.

Figure 3-1 Implementation of XTV-Customize Option in Program Flowchart v5.0p5

3.2 Distribution

The code changes are applicable to the following versions of TRACE: v5.0p5 [1], v5.0p4 [2] and v5.0p3 [3].

3.3 Installation

A source code patch is provided for each version. A patch may be applied using the Linux tar command before the installation command, for example

```
tar -xzf trace_5.0p5_customizeXTV.dev.tar.gz
```

A complete description of the installation procedure on a Linux system is provided in the PSI report [4] for each version of TRACE. The report also describes the patch that fixes the few XTV variable extraction issues previously described in subsection 3.1.

Text copies of the patches for the additional and modification source code are also provided in 7 APPENDIX A and 7 APPENDIX B, respectively.

4 VERIFICATION

Series of verification tests were executed for all TRACE versions (v5.0p5, v5.0p4 and v5.0p3) using input models selected from the test cases distributed with each official TRACE release. The verification of the XTV-Customize option included two components, namely a non-regression testing and a functional testing.

4.1 Procedure

For each code version and each test case, the verification procedure consisted in comparing the XTV (and DMX) graphics output files obtained from the following code and model executions

- Running with a standard TRACE executable the test case for the 3 possible options of Namelist option graphlevel, namely “minimal”, “limited” and “full”.
- Running the aforementioned 3 models with the modified TRACE executable, thus providing the non-regression part of the testing.
- For each of the 3 different options of graphlevel, running with the modified TRACE executable several variants of the test case input file including Namelist options graphCustId and graphCustVar, considering different contents for the component IDs and XTV variables.

The set of XTV graphics outputs file obtained in the first bullet point will be used as reference for both non-regression and functional testing and are hereinafter referred to as “reference”. The non-regression test is positive when the size and content of the two XTV files obtained with a same input file but with the two different executables (first and second bullet points) are identical. Since testing is done for all three possible options, this results in the three different test variants 00, 01 and 02 that are listed in Table 4-1.

Table 4-1 Non-regression Tests Variants

Test case variant # | folder	Test objective	Graphlevel option	XTV file size comparison =/≠ | reference	Variable extraction from XTV/DMX & comparison variable | reference
00 | m_exe	non-regression with modified executable	minimal	= | minimal	random pick | minimal
01 | l_exe	non-regression with modified executable	limited	= | limited	random pick | limited
02 | f_exe	non-regression with modified executable	full	= | full	random pick | full

As can be seen in the table, a first verification of the variants is to compare the obtained and reference XTV files for size. The two sizes should be identical for all variants.

The second verification is made by extracting the same component variable from the two graphics files, i.e. variant and reference, and comparing the two ASCII time history files obtained. The variable name, component ID and node number (when non-scalar) are sampled randomly. Moreover, each variable extraction can be made either from the XTV file or from a DMX file, after random selection.

As for the functional tests (third bullet point), the different variants of model are defined in order to test the XTV-customized option for several aspects. Thus, for each test case model, twelve different variants are prepared and executed. The variants are listed in Table 4-2, which also details for each variant the test objective and the actual subfolder names where each simulation is executed.

Variants 10, 11 and 12 in Table 4-2 are designed to verify that the code adequately addresses an erroneous or irrelevant pair of entries for Namelist options graphCustVar and/or graphCustId, and does so for any graphlevel option. Erroneous entries can consist of omission of one of the two Namelist options, or empty list content, or non-existent variable or component ID, or even inconsistent pairing of valid variable and component ID. The type of error and associated pair variable/ID pair are randomly sampled using information in the reference XTV files. The sampling is implemented using the Python XTV File Reader tool developed and used at PSI [4], hereinafter referred to as xtvReader.

Variants 20, 21 and 22 shall result in the addition in the XTV file of one variable that has been specified using Namelist options graphCustVar and graphCustId. By design, the selected variable does not belong and is actually additional to the graphlevel option set in the model, namely “minimal” for variant 20 or 21, and “limited” for variant 22. Here again, the consistent pairing of component and additional variable that is specified in the input file is sampled randomly using information in the reference XTV files previously obtained.

Table 4-2 Functional Tests Variants

Test case variant # | folder	Test objective	Graphlevel option	XTV file size comparison =/≠ | reference	Variable extraction from XTV/DMX & comparison variable | reference
10 | m_m2m	handling of erroneous graphCustVar and/or graphCustId	minimal	= | minimal	random pick | minimal
11 | l_l2l	handling of erroneous graphCustVar and/or graphCustId	limited	= | limited	random pick | limited
12 | f_f2f	handling of erroneous graphCustVar and/or graphCustId	full	= | full	random pick | full
20 | m_m2l	correctness for one pair variable / component ID	minimal	≠ | minimal	in Namelist | limited
21 | m_m2f	correctness for one pair variable / component ID	minimal	≠ | minimal	in Namelist | full

Table 4-2 Functional Tests Variants (Continued)

Test case variant # | folder	Test objective	Graphlevel option	XTV file size comparison =/≠ | reference	Variable extraction from XTV/DMX & comparison variable | reference
22 | l_l2f	correctness for one pair variable / component ID	limited	≠ | limited	in Namelist | full
30 | m_m2L	exhaustiveness for all model variables / IDs from reference XTV file	minimal	= | limited	random pick | limited
31 | m_m2F	exhaustiveness for all model variables / IDs from reference XTV file	minimal	= | full	random pick | full
32 | l_l2F	exhaustiveness for all model variables / IDs from reference XTV file	limited	= | full	random pick | full
40 | m_m2x	consistency of XTV variables lists in [src/XtvCustomM.f90]	minimal	= | limited	random pick | limited
41 | m_m2X	consistency of XTV variables lists in [src/XtvCustomM.f90]	minimal	= | full	random pick | full
42 | l_l2X	consistency of XTV variables lists in [src/XtvCustomM.f90]	limited	= | full	random pick | full

Variants 30, 31 and 32 are designed to verify the exhaustiveness of the XTV-customize option for each test case model. For these variants, the input file is provided with a Namelist option list graphCustVar that includes all additional variables necessary to match a graphics output level immediately above or two-level beyond than that specified by Namelist option graphlevel in the input model. In addition, the option list graphCustID includes all existing component IDs of the model. The exhaustiveness of the Namelist options lists is ensured by proper inventory of the content of the reference XTV files previously obtained, here again using xtvReader resources.

Variants 40, 41 and 42 are designed to verify the consistency of the lists of variables as defined in the new Fortran 90 module <`XtvCustom`>. Unlike the previous variants 30 to 32, the source of information to set up the content of option list graphCustVar is not the reference XTV files but the content of the parameters lists in the Fortran 90 source [`src/XtvCustomM.f90`]. As a result, these variants not only check for the exhaustiveness of the source code variables lists, but also test the XTV-Customize option for some of the variables that are not activated by the test case models.

As for the size comparison, the XTV file size should be identical to the reference value for most variants, except for variants 20 to 22 that, by design, shall result in slightly larger XTV files, compared to their reference counterparts. This difference in success criteria of the functional test is indicated using self-explanatory symbols “=” and “≠” in Table 4-2 (and also in Table 4-1 for sake of consistency).

The functional testing is executed the same way as the non-regression tests, except for variants 20 to 22 where the XTV variable is not randomly sampled but is uniquely determined by the pair graphCustID/graphCustVar in the Namelist option list of the model variant, since this is the outstanding variable for these variants. This difference in the variable selection method is marked as “in Namelist” in the right-most column of the table.

One should finally mention that, for all variants of the non-regression and functional tests, each test case requiring a restart model employs a restart file from the same variant. Similarly, for the reference simulations using a restart file, the graphlevel option is kept unchanged across restarts.

More details on the scripts developed and employed for the verification procedure are provided in [4].

4.2 Non-regression Tests Results

The results of the non-regression tests are summarized in Table 4-3. In line with the procedure described in previous subsection 4.1, each selected model involved 6 simulations, 3 using the reference executable and 3 using the executable modified for the XTV-Customize option. All of the test variant simulations were properly executed and no issue was observed with respect to the converted DMX files.

As can be seen in the table, not all of the test cases from the official TRACE releases were suitable for testing the XTV-Customize option. Indeed some test cases had to be excluded from the verification procedure for one of the following reasons:

- Test cases with input file written in the legacy format of TRACE, so called “fixed-format”, instead of the more recent free format, and therefore not allowing the use of Namelist options.

- Test cases originally designed to test input errors and resulting in no or faulty XTV file after execution using a standard TRACE executable (is considered faulty, an XTV file resulting in failure for extraction using AptPlot).

- Test cases for which the restart model could not be identified or found.

The complete listing of the test cases selected for the testing, together with information on the unselected ones, is provided in 7 APPENDIX C for TRACE version v5.0p5. Similar lists have been established for previous official releases of TRACE, namely versions v5.0p4 and v5.0p3.

As shown in Table 4-3, the non-regression tests for XTV file size were always positive for all code versions, i.e. the number of positives is equal to the number of simulated variants (half of the total number of simulations, since each model required 3 reference simulations and 3 test simulations, namely variants 00, 01 and 02). As for the random variable extraction and comparison tests, the results were always positive, without any erroneous conversion to DMX format when requested.

In conclusion, the test variants 00, 01 and 02 showed that the executables implementing the XTV-Customize option worked properly for all code versions while all the generated XTV files were confirmed to be fully functional, even after conversion to DMX format.

Table 4-3 Non-regression Tests Results

TRACE version	Number of test models total | selected	Number of simulations (variants only)	Positives for XTV file size	Failed cases for variable extraction XTV | DMX	Ratio of positives
v5.0p5	1249 | 1121	3363	3363	0 | 0	100%
v5.0p4	1011 | 916	2748	2748	0 | 0	100%
V5.0p3	925 | 840	2520	2520	0 | 0	100%

4.3 Functional Tests Results

The results of the functional tests are summarized in Table 4-4. In line with the number of test variants previously listed in Table 4-2, each selected model involved 12 additional simulations for the functional verification step. To start, all the simulations were properly executed using the modified executables, including the cases requiring a restart and/or a coupling with a PARCS 3-D kinetics model. Also, the conversion to DMX files did not pose any problem (note that two variable extractions per case functional test were required, one for the variant and one for the reference case, reason why the numbers indicated in the 5th column of Table 4-4 add up to twice the actual number of failed variable extraction tests per version).

Table 4-4 Functional Tests Results

TRACE version	Number of test models total | selected	Number of simulations (variants only)	Positives for XTV file size	Failed cases x2 for variable extraction XTV | DMX	Ratio of positives
v5.0p5	1249 | 1121	13452	12368 (13452*)	3 | 1 (0 | 0*)	95.96% (100.00%*)
v5.0p4	1011 | 916	10992	10363 (10992*)	0 | 0	97.14% (100.00%*)
V5.0p3	925 | 840	10080	9527 (10078*)	31 | 51	96.85% (99.58%*)

Also, for all the tested versions, most of the selected test cases have resulted in the expected outcome, namely correct addition of variables in the XTV file when expected, and no enlargement, shrinking or corruption of the XTV file for the variants designed for inconsistent or incomplete input to graphCustVar and graphCustId.

As a side note, one can mention that the size of the standard dump output file (TPR) obtained with the XTV-Customize option is not identical to that obtained with the reference case, which

was not the case in the non-regression test. This is normal since the TPR file includes the content of Namelist variables graphCustVar and graphCustId only when the XTV-Customize option is enabled.

More importantly, note in Table 4-4 the small proportion of apparently failed tests, which have been investigated in details and were most of the time found to relate to a few root causes that are understood and should not necessarily be considered as errors. When including these clarifications, the ratio of positives is actually larger than initially estimated. Better informed estimates of the positive tests and overall ratio are indicated in Table 4-4 with symbol '*' for each version of the code.

These detailed investigations that helped refining the estimates of positives are presented in the next two subsections, the first on the XTV file size test and the second on the XTV variable extraction and comparison test.

4.3.1 Failed Tests for XTV File Size

- For code versions v5.0p5 and v5.0p4, some test cases of variants 30 and 40 resulted in an XTV file slightly larger than expected. This undesirable outcome is due to the synonymy of some (few) XTV variables for different TRACE model components and at distinct graphics output levels. This is the case for eight XTV variables (namely alven, alvn, chtan, chtin, cl, cv, visl and visv), which are graphics output at level "limited" of component type 'Contan' but also at level "full" of any 1-D (e.g. 'Pipe', 'Tee') or 3-D ('Vessel') component type. These failed tests are due to the use of two separated variables lists in the Namelist option lists, which can lead to equivocal declaration of XTV variables and hence larger than expected XTV files for models including components of type 'Contan'.

- The aforementioned equivocal variable declaration risk is augmented in version v5.0p5 because of its new variable peakClad, which is the output of both component types 'Htstr' and 'Chan' but at different levels "limited" and "full", respectively. Here again, this could make the XTV file larger than expected for models including both types of components.

- Moreover, note that the vast majority of the failed tests for variant 10 are related to the arbitrary assignment of one variable per component in the XTV file, independent of the content of option lists graphCustId and graphCustVar. As explained at the end of subsection 3.1, this was necessary to circumvent an AptPlot post-processing issue that is unrelated to the XTV-Customize option. In these cases, which can affect any code version, the file size difference compared to the reference XTV file obtained with level "minimal" is simply due to the arbitrarily added XTV variables.

- For code version v5.0p3, few cases of variant 10 failed the XTV file size test for a less obvious reason, since the number of XTV variables was identical to the reference file but the number of time records was slightly different (hence the file size difference). In version v5.0p3 for instance, this corresponded to two models, namely w4loopri.inp of suites 'ConSys' and SameRodsTR.inp of suites 'Power'. Note that both models were actually restarts from simulations with models w4loopi.inp and SameRodsSS.inp, respectively, which had in common the use of a steady-state initialization option (input stdyst > 0) and included a component 'Prizer'. This hinted at the possible root cause of the discrepancy, as this will be further discussed in next subsection. Note finally that most recent versions v5.0p4 and v5.0p5 were not affected by this problem, as well as most of the restart cases from versions v5.0p3.

4.3.2 Failed Tests for XTV File Content

- Code version v5.0p4 did not result in any failed test, while code version v5.0p5 resulted in only two failed tests in total. These two apparent failures were found to be in fact related to the aforementioned duplicity of variable peakClad. In the functional testing campaign for variant 30, that variable was randomly picked for a component ‘Chan’, although the variable is not an output for such component type in the reference XTV file (i.e. level "limited", as shown in Table 4-2). Here again, the two apparent failures were only artificial outliers resulting from loopholes in the verification procedure, and versions v5.0p4 and v5.0p5 did in fact result in no error for XTV file content.

- As for code version v5.0p3, several cases of variant 10 failed the variable comparison test. These cases were all associated with variables sampled from types ‘Global’, ‘Signal-variable’, ‘Trip-variable’, or ‘Control-block’, by definition of the variant 10, which should be consistent with a reference case using level "minimal" (designed to output only signal, trip and control system variables in the XTV file).

- A visual inspection of the aforementioned discrepancies has been carried out for version v5.0p3 using the comparison plots shown in 7 APPENDIX D. As can be seen, the differences are very small. They essentially reflect the differences in time-stepping, indicated by the evolution of the time-step variable "delt", also shown in the appendix for each affected test. As explained in more details in the appendix, this difference stems from a minor numerical diffusion issue associated with the TRACE component ‘Prizer’ when using the steady-state initialization option (stdyst > 0). Note that this issue has been verified to affect only version v5.0p3, whereas the most recent versions (v5.0p5 and v5.0p4) were verified to include a fix in the original source code that prevents such discrepancy in any case.

4.3.3 Summary of the Tests Results

The non-regression and functional verifications of the XTV-Customize option have been conducted over the all set of test models provided by the code developer, thus covering a very wide range of possible combination of modelling options available in TRACE. The results showed a complete success for the non-regression test suite and a very high success ratio in the functional test suite, and this for all codes versions, precisely 0% error for v5.0p5 and v5.0p4, and less than 0.5% error for earlier version v5.0p3.

As for the two versions based on v5.0p3, a detailed investigation of the failed tests showed that the issue was minor and actually unrelated to the XTV-Customize option itself. Essentially, the user willing to use the XTV-Customize option should expect some round-off error to marginally affect scalar results (global, signal, trip and control block variables) if the model employed includes a component ‘Prizer’ and uses a steady-state initialization option (stdyst > 0). Nevertheless, as shown in 7 APPENDIX D, the difference will be very small, if noticeable at all, compared to a model executed with a reference TRACE executable.

As for newer versions v5.0p5 and v5.0p4, the user will see no difference in the extracted XTV variables whether obtained with the XTV-Customize option or not. In some cases where graphics output option "minimal" is selected, the XTV file may however contain more variables than expected (namely peakClad, alven, alvn, chtan, chtin, cl, cv, visl and visv). A fix to avoid this is proposed in the next section, together with other possible improvements.

5 APPLICATION TO AN UNCERTAINTY QUANTIFICATION STUDY

As mentioned in introduction, the XTV-Customize option can help optimizing the computation resources for analyses using large simulation models where an exhaustive description of the model output is not necessary. The option is also particularly indicated for analyses that include detailed sensitivity studies or large forward uncertainty propagation campaigns. This section illustrates the latter with a simplified uncertainty propagation study applied to a TRACE model of the Feba-216 reflood experiment test [5]. This model has been developed as part of PSI participation to the OECD PREMIUM project ([6], [7]) and has been intensively employed in [8] for the development of a Bayesian data assimilation method to derive the uncertainty of physical models in TRACE. A brief description of the TRACE model is provided in 7 APPENDIX E.

The Feba-216 test data provided in PREMIUM essentially comprise time histories of heater rod outer wall temperatures at 8 different elevations, axial pressure differences across 4 segments of the test section, and several more measurements (liquid carry-over off the test section, quench front progression, housing wall temperature at one elevation). Thus, the model output needed for the validation of the model with associated uncertainties could be limited to the set of XTV variables necessary to reproduce the aforementioned measurements. Using the XTV-Customize option, this can be achieved with the following additional Namelist entries to the Feba-216 model:

```
&INOPTS
  graphlevel   = "minimal",
  graphCustId  = 1,10,20,30,40,99
  graphCustVar = "pn","tln","vln","bottomQF","rftn","trhmax","tpower"
&END
```

The gains in terms of XTV file storage and variable extraction efforts are evaluated for a simplified variant of the uncertainty propagation study where only the uncertainties in the test boundary conditions are considered, namely heating power, system pressure and inlet coolant conditions (temperature and velocity). The probability density functions (PDF) of these boundary conditions, shown in Table 5, are consistent with the PREMIUM specifications.

Table 5-1 Study Feba-216: Input Uncertainties

Input parameter	PDF Type	PDF Units	PDF Mean	PDF Range
Outlet pressure	Uniform Multiplicative	[-]	1.00	[0.90 , 1.10]
Inlet coolant temperature	Uniform Additive	[K]	0.00	[–5.0 , +5.0]
Inlet coolant velocity	Uniform Multiplicative	[-]	1.00	[0.90 , 1.10]
Heater rods power	Uniform Multiplicative	[-]	1.00	[0.95 , 1.05]

For this comparison, the uncertainty propagation study is performed two times, one with XTV file customization and one without but using Namelist option graphlevel="full", since the variables rftn are required to reproduce the rod outer temperatures at the correct axial elevations where the thermocouples are located. The comparison is made for two versions of the TRACE executable implementing the XTV-Customize option, based on v5.0p5 and v5.0p3, respectively. Figure 5-1 shows a comparison of some of the model predictions with test data.

All the variables extracted from the customized XTV file were first verified to be identical to that from the standard XTV file. Two figures of merit were then used to compare the studies, namely the cumulated size of all the XTV files generated by one uncertainty propagation set, and the wall-time required by the AptPlot tool to extract (in batch mode) all the model variables necessary to replicate the test data.

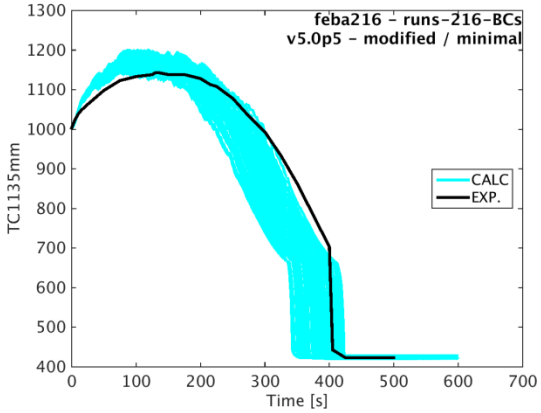
As can be seen in Table 6, the customization of the XTV file to the objectives of the study not only reduced the storage requirements by nearly a factor 3, but also helped reducing the post-processing time to a similar fraction since the AptPlot tool extracts variables more efficiently when the XTV file is smaller. One can also see in the table that the gains are slightly more substantial for v5.0p5 than for v5.0p3 both in terms of storage space and processing time, in particular because v5.0p5 includes more XTV variables than v5.0p3 for option graphlevel="full".

Table 6 Study Feba-216: Comparison of XTV Files Size and Post-Processing Time

TRACE version	Runs per study	XTV files total size XTV-custom | level "full"	Post-processing time XTV-custom | level "full"	Relative difference space | time
v5.0p5	50	4'843 MB | 14'027 MB	1'107 sec | 2'892 sec	–65.5% | –61.7%
V5.0p3	50	4'997 MB | 13'018 MB	1'090 sec | 2'602 sec	–61.1% | –58.1%

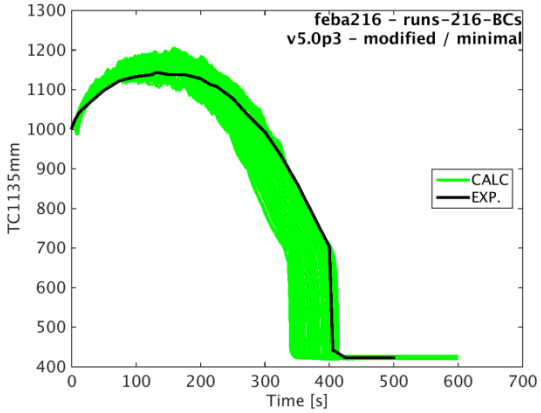
Note, finally, that the Feba-216 model is not a particularly large TRACE model (composed of 7 thermal-hydraulics components), and that the variability bands have been obtained using just 50 simulations per study (just for the sake of the demonstration, these bands should not be considered as statistically meaningful). Thus, with this simple example, one can consider the potential savings in hardware and computation requirements for more complex simulation models, in particular at the stage(s) of the analysis where an exhaustive knowledge of the model output is not (or is no longer) necessary.

study: runs-216-BCs
code: V5.0P5



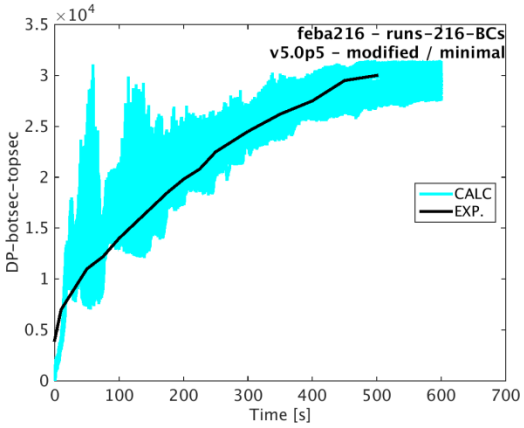
Rod outer temperature [K] at z = –1135 mm

study: runs-216-BCs
code: V5.0P3



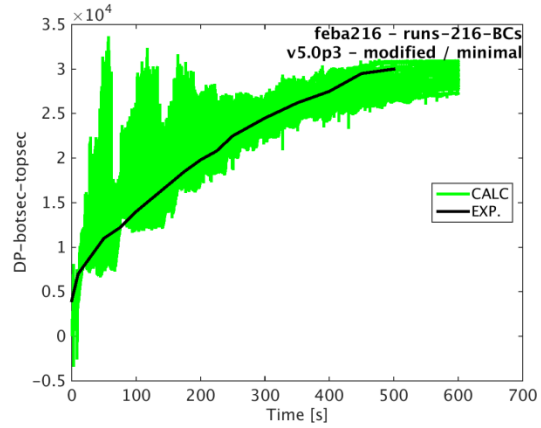
Rod outer temperature [K] at z = –1135 mm

study: runs-216-BCs
code: V5.0P5



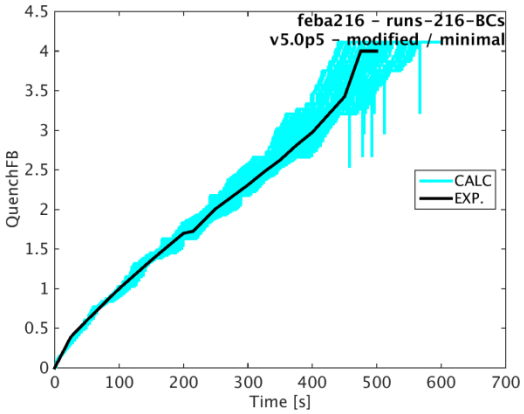
Pressure difference [Pa] throughout test section

study: runs-216-BCs
code: V5.0P3



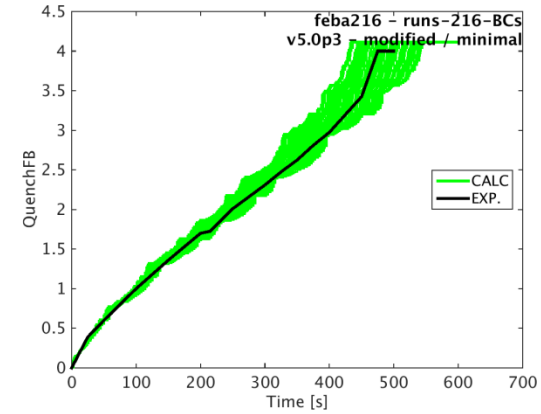
Pressure difference [Pa] throughout test section

study: runs-216-BCs
code: V5.0P5



Quench front progression [m]

study: runs-216-BCs
code: V5.0P3



Quench front progression [m]

Figure 5-1 Simulation of Feba-216 – Effects of Uncertainty on Boundary Conditions

6 POSSIBLE IMPROVEMENTS

In this section, some possible improvements to the current patch are discussed.

6.1 Merging of the Two Namelist Option Lists

The two lists graphCustVar and graphCustId could be merged into one single list that would combine both types of information and so allow for unequivocal declaration of the additional variables requested for each component.

Here is an example of how a single declaration list (e.g. graphCustom) could be employed:

```
&INOPTS
  graphlevel   = "minimal",
  graphCustom  = "dprmax",
                 10,20,"pn","alpn","tln","gamn"
                 30,"pn","alpn"
&END
```

The behaviour of TRACE based on such input list would be as follows

- The general problem variable dprmax, which normally pertains to graphics output level "full", will be added to the XTV file.
- For components 10 and 20 of the input file, output variables pn, alpn, tln, and gamn will be added to the XTV file.
- For component 30, only output variables pn and, alpn will be added to the XTV file.

As can be inferred from the example for components 10 and 20, the single list could also readily emulate the two lists approach currently adopted where one additional variable can be declared for several components at once.

Since the single list would combine integers and strings, such improvement may require a modification in module <`NamlistDat`> of interface (`InitNmlstVal`) and relevant associated subroutines.

6.2 Improvement of Input Error Detection and Warning

In this developmental version, a rather permissive input processing approach has been adopted, where incorrect or improper variable declaration does not trigger any error or warning message and is simply overlooked.

However, in the context of the preparation of large simulation campaigns, a more rigorous input error processing would be desirable to assist the user.

6.3 Centralization of XTV Variables Declaration in Source Code

Beyond the XTV-Customize option itself, this work provided the opportunity to group within the new module <`XtvCustom`> the declaration of all the XTV variables in a set of parameters lists.

This allowed for a compact segregation of all the XTV variables across component types, memory allocation (static vs dynamic), and variable category (default vs optional). The differentiation for variable dimension (scalar, 1-D, 2-D, or 3-D) was not necessary for the current version of the XTV-Customize option, but could also be implemented.

A centralization of the XTV variable lists could allow for a more generic formulation in the original module <XtvComps> of the various component-specific subroutines (e.g. {Xtv1D}, {XtvHtStr}) where calls of the subroutine {AddXtvVar} have to be made individually with a specific Fortran line for each XTV variable.

Some advantages of such factorization of the XTV variables declaration could be:

- Reduction of the size (and possibly number) of subroutines in component <XtvComps>.
- Simplification of the process of addition or modification of XTV variables for future versions of TRACE.
- Removing the necessity to replicate in subroutine {XtvVarListsPrep} the program flow logic for optional XTV variables that is written in all the different component specific subroutines {AddXtvVar}, and which is a drawback of the current implementation of the XTV-Customize option.

If such centralization is pursued, one could eventually move the XTV variables lists and associated resources of module <XtvCustom> into the original module <XtvVar> in source file [src/XtvVarM.f90], or into the original module <XtvComps>.

6.4 Compatibility with Namelist Option XtvAppend

The XTV-Customize option has been designed for a situation where the simulation creates a new XTV file, namely for the default value XtvAppend=0. Non-default values of option XtvAppend allow for appending graphics output to an old XTV file. The possibility to extend the XTV-Customize option to such cases has not been investigated and therefore has not been tested in this study, but such development could be considered in future versions.

7 REFERENCES

1. USNRC. Transmittal of TRACE V5.0, Patch 05, with Source. August 24 2017

2. USNRC. Transmittal of TRACE V5.0, Patch04 with Source - Replacement DVD. June 12 2014

3. USNRC. TRACE V5.0 User's Manual. Transmittal of TRACE V5.0, Patch03 with Source Code. June 29 2012

4. O. Zerkak. Customization of XTV Graphics Output in TRACE v5.0p5 – Applicable to Versions v5.0p5, v5.0p4, v5.0p3 and v5.0p3UQ. PSI Memorandum, October 2018 (SB-TRCE-ACT-001-09.002)

5. P. Ihle and K. Rust. FEBA—Flooding Experiments with Blocked Arrays Evaluation Report. Kernforschungzentrum-Karlsruhe, Report KfK 3657, March1984

6. Reventós, E. de Alfonso and R. Mendizábal Sanz. PREMIUM, a Benchmark on the Quantification of the Uncertainty of the Physical Models in the System Thermal-hydraulic Codes: Methodologies and Data Review. CSNI report NEA/CSNI/R(2016)9, April 2016

7. Mendizábal, E. de Alfonso, J. Freixa and F. Reventós, T. Skorek, J. Baccou, J. Zhang, E. Nouy and P. Emonot. Post-BEMUSE Reflood Model Input Uncertainty Methods (PREMIUM) Benchmark – Final Report. CSNI report NEA/CSNI/R(2016)18, August 2017

8. D. Wicaksono. Bayesian Uncertainty Quantification of Physical Models in Thermal-hydraulics System Codes. EPFL Thesis N° 8426 (2018)

9. D. Wicaksono, O. Zerkak and A. Pautz. Global Sensitivity Analysis of Transient Code Output Applied to a Reflood Experiment Model Using the TRACE Code. Nuclear Science and Engineering, Vol. 184, 400-429, (2016)

10. USNRC. TRACE Pressurized Water Reactor Modeling Guidance. Preliminary Draft Report. 2012. Washington, DC

APPENDIX A
ADDITIONAL MODULE XTVCUSTOM FOR V5.0P5

This appendix presents the content of the additional source file [src/XtvCustomM.f90] for the additional module <XtvCustom> that implements the XTV-Customize option. This is the source code applicable to TRACE v5.0p5 without implementation of the fixes for the few XTV variable extraction issues previously described in subsection 3.1. Note that the structure of the module is very similar for the other code versions v5.0p4 and v5.0p3.

```
MODULE XtvCustom
!
!     (c) 2017 Paul Scherrer Institut (PSI)
!     This module is provided as in-kind contribution within
!     the CAMP Implementing Agreement between NRC, PSI and
!     the Swiss Federal Nuclear Safety Inspectorate
!     (SB-MNG-CCN-002-15, 2015-2019)
!
!     This module contains the resources for custom XTV output capability
!
!     BEGIN MODULE USE
      USE GlobalDat, ONLY: sik
      USE IntrType,  ONLY: sdk
      USE Io, ONLY: outUnt
!
      IMPLICIT NONE
!
      INTERFACE XtvVarListMerge
         MODULE PROCEDURE XtvVarListMerge2A
         MODULE PROCEDURE XtvVarListMerge3A
         MODULE PROCEDURE XtvVarListMerge4A
      END INTERFACE

      INTEGER(sik), PARAMETER             :: graphCustIdDef = 0
      ! Optional component ID numbers for graphics filter
      INTEGER(sik), DIMENSION(1024)       :: graphCustId = graphCustIdDef

      CHARACTER(LEN=13), PARAMETER        :: graphCustVarDef = ''
      ! Variable names list for applying dynamic variable filter
      CHARACTER(LEN=13), DIMENSION(1024)  :: graphCustVar = graphCustVarDef

      ! Xtv filtering option flag
      INTEGER(sik)                        :: graphFltFlag = 0

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

      !CHARACTER(LEN=13), ALLOCATABLE, DIMENSION(:)   :: XtvSVarsMin
      !CHARACTER(LEN=13), ALLOCATABLE, DIMENSION(:)   :: XtvSVarsLim
      !CHARACTER(LEN=13), ALLOCATABLE, DIMENSION(:)   :: XtvSVarsFull

      CHARACTER(LEN=13), ALLOCATABLE, DIMENSION(:)   :: XtvDVarsMin
      CHARACTER(LEN=13), ALLOCATABLE, DIMENSION(:)   :: XtvDVarsLim
      CHARACTER(LEN=13), ALLOCATABLE, DIMENSION(:)   :: XtvDVarsFull

      !CHARACTER(LEN=13), ALLOCATABLE, DIMENSION(:,:) :: XtvDVars

!!!!!!!!!!!!!!!!!!!!!!!!!  General problem variables  !!!!!!!!!!!!!!!!!!!!!!!!!

      ! All graphics lists for static variables
```

```
      CHARACTER(LEN=13), PARAMETER        :: XtvSVarsGnPrMin(0) = ''
      CHARACTER(LEN=13), PARAMETER        :: XtvSVarsGnPrLim(0) = ''
      CHARACTER(LEN=13), PARAMETER        :: XtvSVarsGnPrFull(0) = ''

      ! Dynamic variables

      ! Graphics list for graphLevel = "minimal"
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsGnPrMin(26) = (/           &
        'alp', 'cmass', 'cMNCG', 'cpuTime', 'delt', 'eMass', 'eMNCG', 'massB',  &
        'massE', 'massF', 'mNCGB', 'mNCGE', 'mNCGF', 'tnstep',                  &
        'cSolidsM', 'cSolM', 'eSolM', 'mSolB', 'mSolE', 'mSolF',                &
        'cB10M', 'cB10Sold', 'eB10M', 'mB10B', 'mB10E', 'mB10F'                 &
        /)
      ! Unconditional variables
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsGnPrMinDef(14) = (/        &
        'alp', 'cmass', 'cMNCG', 'cpuTime', 'delt', 'eMass', 'eMNCG', 'massB',  &
        'massE', 'massF', 'mNCGB', 'mNCGE', 'mNCGF', 'tnstep'                   &
        /)
      ! Conditional variables for Namelist option iSolut > 0
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsGnPrMinIsolut(6) = (/      &
        'cSolidsM', 'cSolM', 'eSolM', 'mSolB', 'mSolE', 'mSolF'                 &
        /)
      ! Conditional variables for Namelist option iSolut > 0 and fracB10 < 1
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsGnPrMinIsolutFracB(6) = (/ &
        'cB10M', 'cB10Sold', 'eB10M', 'mB10B', 'mB10E', 'mB10F'                 &
        /)

      ! Graphics list for graphLevel = "limited"
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsGnPrLim(0) = ''
      ! Unconditional variables
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsGnPrLimDef(1) = (/         &
        'dummy'                                                                 &
        /)

      ! Additional graphics list for graphLevel = "full"
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsGnPrFull(20) = (/          &
        'deltTC', 'dprmax', 'dtlmax', 'dtrmax', 'dtvmax', 'eError', 'fracC',    &
        'fracMA', 'fracMD', 'fracMF', 'fracMP', 'fracMS', 'mError', 'nStepC',   &
        'rmaC', 'rmbC', 'rmdC', 'rmfC', 'rmlC', 'rmsC'                          &
        /)
      ! Unconditional variables
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsGnPrFullDef(4) = (/        &
        'dprmax', 'dtlmax', 'dtrmax', 'dtvmax'                                  &
        /)
      ! Conditional variables for Namelist option ncontant > 0_sik
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsGnPrFullCont(16) = (/      &
        'deltTC', 'eError', 'fracC', 'fracMA', 'fracMD', 'fracMF', 'fracMP',    &
        'fracMS', 'mError', 'nStepC', 'rmaC', 'rmbC', 'rmdC', 'rmfC', 'rmlC',   &
        'rmsC'                                                                  &
        /)

!!!!!!!!!!!!!!!!!!  Fluid components mass balance variables  !!!!!!!!!!!!!!!!!!

      ! All graphics lists for static variables

      CHARACTER(LEN=13), PARAMETER        :: XtvSVarsMassBMin(0) = ''
      CHARACTER(LEN=13), PARAMETER        :: XtvSVarsMassBLim(0) = ''
      CHARACTER(LEN=13), PARAMETER        :: XtvSVarsMassBFull(0) = ''

      ! Dynamic variables

      ! Graphics list for graphLevel = "minimal"
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsMassBMin(0) = ''
```

```
      ! Unconditional variables
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsMassBMinDef(1) = (/        &
        'dummy'                                                                 &
        /)

      ! Graphics list for graphLevel = "limited"
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsMassBLim(18) = (/          &
        'cB10M', 'cB10Sold', 'cmass', 'cMNCG', 'cSolidsM', 'cSolM', 'eB10M',    &
        'eMass', 'eMNCG', 'eSolM', 'massC', 'massE', 'mB10C', 'mB10E', 'mNCGC', &
        'mNCGE', 'mSolC', 'mSolE'                                               &
        /)
      ! Unconditional variables
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsMassBLimDef(8) = (/        &
        'cmass', 'cMNCG', 'eMass', 'eMNCG', 'massC', 'massE', 'mNCGC', 'mNCGE'  &
        /)
      ! Conditional variables for Namelist option iSolut > 0
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsMassBLimIsolut(5) = (/     &
        'cSolidsM', 'cSolM', 'eSolM', 'mSolC', 'mSolE'                          &
        /)
      ! Conditional variables for Namelist option iSolut > 0 and fracB10 < 1
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsMassBLimIsolutFracB(5) = (/&
        'cB10M', 'cB10Sold', 'eB10M', 'mB10C', 'mB10E'                          &
        /)

      ! Additional graphics list for graphLevel = "full"
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsMassBFull(1) = (/          &
        'dummy'                                                                 &
        /)
      ! Unconditional variables
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsMassBFullDef(1) = (/       &
        'dummy'                                                                 &
        /)

!!!!!!!!!!!!!!!!!!!!!!!!!  Tee component variables  !!!!!!!!!!!!!!!!!!!!!!!!!

      ! All graphics lists for static variables

      CHARACTER(LEN=13), PARAMETER        :: XtvSVarsTeeMin(0) = ''
      CHARACTER(LEN=13), PARAMETER        :: XtvSVarsTeeLim(0) = ''
      CHARACTER(LEN=13), PARAMETER        :: XtvSVarsTeeFull(0) = ''

      ! Dynamic variables

      ! Graphics list for graphLevel = "minimal"
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsTeeMin(0) = ''
      ! Unconditional variables
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsTeeMinDef(1) = (/          &
        'dummy'                                                                 &
        /)

      ! Graphics list for graphLevel = "limited"
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsTeeLim(1) = (/             &
        'dummy'                                                                 &
        /)
      ! Unconditional variables
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsTeeLimDef(1) = (/          &
        'dummy'                                                                 &
        /)

      ! Additional graphics list for graphLevel = "full"
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsTeeFull(2) = (/            &
        'powr1', 'powr2'                                                        &
        /)
```

```
      ! Unconditional variables
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsTeeFullDef(2) = (/         &
        'powr1', 'powr2'                                                        &
        /)

!!!!!!!!!!!!!!!!!!!!!  Generic 1D component variables  !!!!!!!!!!!!!!!!!!!!!

      ! All graphics lists for static variables

      CHARACTER(LEN=13), PARAMETER        :: XtvSVars1DMin(0) = ''
      CHARACTER(LEN=13), PARAMETER        :: XtvSVars1DLim(1) = (/              &
        'vol'                                                                   &
        /)
      CHARACTER(LEN=13), PARAMETER        :: XtvSVars1DFull(0) = ''

      ! Dynamic variables

      ! Graphics list for graphLevel = "minimal"
      CHARACTER(LEN=13), PARAMETER        :: XtvDVars1DMin(0) = ''
      ! Unconditional variables
      CHARACTER(LEN=13), PARAMETER        :: XtvDVars1DMinDef(1) = (/           &
        'dummy'                                                                 &
        /)

      ! Graphics list for graphLevel = "limited"
      CHARACTER(LEN=13), PARAMETER        :: XtvDVars1DLim(71) = (/             &
        'alpn', 'am', 'bubAi-', 'bubaitot', 'bubFrc-', 'bubVel-', 'choked',     &
        'concS', 'concW', 'dropAi-', 'dropFrc-', 'dropVel-', 'el', 'ev', 'fa',  &
        'mFrB10', 'pan', 'pn', 'regnm', 'rlmf', 'rmvm', 'roan', 'roln', 'rom',  &
        'rovn', 'rvmf', 'sn', 'solMaxS', 'solMaxW', 'sSolids', 'tln', 'tsat',   &
        'tssn', 'tvn', 'vln', 'vvn', 'x',                                       &
        'g1BE', 'g1RC', 'g1TI', 'g1WE',                                         &
        'g1WN111', 'g2BE1', 'g2BE2', 'g2BE121', 'g2BE122', 'g2RC111',           &
        'g2RC1121', 'g2RC1122', 'g2RC1221', 'g2RC1222', 'g2RC222', 'g2SI222',   &
        'g2SO211', 'g2SO212', 'g2TI111', 'g2TI211', 'g2TI222', 'g2WE111',       &
        'g2WE1121', 'g2WE1122', 'g2WE1221', 'g2WE1222', 'g2WE222', 'intEx122',  &
        'intRC112', 'intRC122', 'intSO211', 'intTI212', 'intWE112', 'intWE122'  &
        /)
      ! Unconditional variables
      CHARACTER(LEN=13), PARAMETER        :: XtvDVars1DLimDef(22) = (/          &
        'alpn', 'am', 'choked', 'el', 'ev', 'fa', 'pan', 'pn', 'regnm', 'rlmf', &
        'rmvm', 'roan', 'roln', 'rom', 'rovn', 'rvmf', 'tln', 'tsat', 'tssn',   &
        'tvn', 'vln', 'vvn'                                                     &
        /)
      ! Conditional variables for Namelist option iSolut > 0
      CHARACTER(LEN=13), PARAMETER        :: XtvDVars1DLimIsolut(6) = (/        &
        'concS', 'concW', 'sn', 'solMaxS', 'solMaxW', 'sSolids'                 &
        /)
      ! Conditional variables for Namelist option iSolut > 0 & solveB10 = .TRUE.
      CHARACTER(LEN=13), PARAMETER        :: XtvDVars1DLimIsolutsolveB10(1) = (/&
        'mFrB10'                                                                &
        /)
      ! Conditional variables for Namelist options nTraceG != 0 or nTraceL != 0
      ! xor Namelist option igas > 11 (therefore numberOfNCGases > 1)
      CHARACTER(LEN=13), PARAMETER        :: XtvDVars1DLimItrace(1) = (/        &
        'x'                                                                     &
        /)
      ! Conditional variables for bubbles when Namelist option disfields > 0
      CHARACTER(LEN=13), PARAMETER        :: XtvDVars1DLimDisfieldsB(3) =  (/   &
        'bubAi-', 'bubFrc-', 'bubVel-'                                          &
        /)
      ! Conditional variables for droplets when Namelist option disfields > 0
      CHARACTER(LEN=13), PARAMETER        :: XtvDVars1DLimDisfieldsD(3) =  (/   &
```

```
        'dropAi-', 'dropFrc-', 'dropVel-'                                    &
        /)
      ! Conditional variables for Namelist option disfields > 0 and nBubbles = 1
      CHARACTER(LEN=13), PARAMETER        :: XtvDVars1DLimDisfieldsnB1(4) =  (/ &
        'g1BE', 'g1RC', 'g1TI', 'g1WE'                                       &
        /)
      ! Conditional variables for Namelist option disfields > 0 and nBubbles = 2
      CHARACTER(LEN=13), PARAMETER        :: XtvDVars1DLimDisfieldsnB2(30) = (/ &
        'g1WN111', 'g2BE1', 'g2BE2', 'g2BE121', 'g2BE122', 'g2RC111',        &
        'g2RC1121', 'g2RC1122', 'g2RC1221', 'g2RC1222', 'g2RC222', 'g2SI222', &
        'g2SO211', 'g2SO212', 'g2TI111', 'g2TI211', 'g2TI222', 'g2WE111',    &
        'g2WE1121', 'g2WE1122', 'g2WE1221', 'g2WE1222', 'g2WE222', 'intEx122', &
        'intRC112', 'intRC122', 'intSO211', 'intTI212', 'intWE112', 'intWE122' &
        /)
      ! Conditional variables for Namelist option bubDev = .TRUE.
      ! Note: bubDev is always .TRUE. in v5.0Patch5
      CHARACTER(LEN=13), PARAMETER        :: XtvDVars1DLimBubdev(1) = (/       &
        'bubaitot'                                                           &
        /)

      ! Additional graphics list for graphLevel = "full"
      CHARACTER(LEN=13), PARAMETER        :: XtvDVars1DFull(40) = (/           &
        'alpE', 'alpmn', 'alpmx', 'alven', 'alvn', 'b10Sppm', 'b10Wppm',     &
        'chtan', 'chtin', 'cifn', 'cSppm', 'cWppm', 'cl', 'cv', 'elm', 'evm', &
        'flowQual', 'flv', 'gamn', 'hgam', 'horLev', 'hOverD', 'phIL', 'pStag', &
        'pStagE', 'rhofj', 'rhogj', 'slpratio', 'visl', 'visv', 'vlvol',     &
        'voidfj', 'voidgj', 'volM', 'volP', 'vvvol', 'wfhf', 'wfhfE', 'wfl', &
        'wfv'                                                                &
        /)
      ! Unconditional variables
      CHARACTER(LEN=13), PARAMETER        :: XtvDVars1DFullDef(36) = (/        &
        'alpE', 'alpmn', 'alpmx', 'alven', 'alvn', 'chtan', 'chtin', 'cifn', &
        'cl', 'cv', 'elm', 'evm', 'flowQual', 'flv', 'gamn', 'hgam', 'horLev', &
        'hOverD', 'phIL', 'pStag', 'pStagE', 'rhofj', 'rhogj', 'slpratio',   &
        'visl', 'visv', 'vlvol', 'voidfj', 'voidgj', 'volM', 'volP', 'vvvol', &
        'wfhf', 'wfhfE', 'wfl', 'wfv'                                        &
        /)
      ! Conditional variables for Namelist option iSolut > 0
      CHARACTER(LEN=13), PARAMETER        :: XtvDVars1DFullIsolut(4) = (/      &
        'b10Sppm', 'b10Wppm', 'cSppm', 'cWppm'                               &
        /)

!!!!!!!!!!!!!!!!!!!!!!!!  Break component variables  !!!!!!!!!!!!!!!!!!!!!!!!

      ! All graphics lists for static variables

      CHARACTER(LEN=13), PARAMETER        :: XtvSVarsBreakMin(0) = ''
      CHARACTER(LEN=13), PARAMETER        :: XtvSVarsBreakLim(1) = (/          &
        'vol'                                                                &
        /)
      CHARACTER(LEN=13), PARAMETER        :: XtvSVarsBreakFull(0) = ''

      ! Dynamic variables

      ! Graphics list for graphLevel = "minimal"
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsBreakMin(0) = ''
      ! Unconditional variables
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsBreakMinDef(1) = (/       &
        'dummy'                                                              &
        /)

      ! Graphics list for graphLevel = "limited"
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsBreakLim(19) = (/         &
```

```
        'alpn', 'b10MFR', 'bsa', 'bsmass', 'bxa', 'bxmass', 'concS', 'concW',   &
        'mB10C', 'mFrB10', 'mSolC', 'enth', 'm', 'mfr', 'pan', 'pn', 'solutMFR',&
        'tln', 'tvn'                                                            &
        /)
      ! Unconditional variables
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsBreakLimDef(10) = (/       &
        'alpn', 'bsa', 'bsmass', 'bxa', 'bxmass', 'enth', 'pan', 'pn', 'tln',   &
        'tvn'                                                                   &
        /)
      ! Conditional variables for Namelist option iSolut != 0
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsBreakLimIsolut(4) = (/     &
        'concS', 'concW', 'mSolC', 'solutMFR'                                   &
        /)
      ! Conditional variables for Namelist option iSolut != 0 & solveB10 = .TRUE.
      CHARACTER(LEN=13), PARAMETER     :: XtvDVarsBreakLimIsolutsolveB10(3) = (/&
        'b10MFR', 'mB10C', 'mFrB10'                                             &
        /)
      ! Conditional variables for Namelist options nTraceG != 0 or nTraceL != 0
      ! xor Namelist option igas > 11 (therefore numberOfNCGases > 1)
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsBreakLimItrace(2) = (/     &
        'm', 'mfr'                                                              &
        /)

      ! Additional graphics list for graphLevel = "full"
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsBreakFull(4) = (/          &
        'b10Sppm', 'b10Wppm', 'cSppm', 'cWppm'                                  &
        /)
      ! Unconditional variables
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsBreakFullDef(1) = (/       &
        'dummy'                                                                 &
        /)
      ! Conditional variables for Namelist option iSolut != 0
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsBreakFullIsolut(4) = (/    &
        'b10Sppm', 'b10Wppm', 'cSppm', 'cWppm'                                  &
        /)

!!!!!!!!!!!!!!!!!!!!!!!!!  Chan component variables  !!!!!!!!!!!!!!!!!!!!!!!!!

      ! All graphics lists for static variables

      CHARACTER(LEN=13), PARAMETER        :: XtvSVarsChanMin(0) = ''
      CHARACTER(LEN=13), PARAMETER        :: XtvSVarsChanLim(0) = ''
      CHARACTER(LEN=13), PARAMETER        :: XtvSVarsChanFull(0) = ''

      ! Dynamic variables

      ! Graphics list for graphLevel = "minimal"
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsChanMin(0) = ''
      ! Unconditional variables
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsChanMinDef(1) = (/         &
        'dummy'                                                                 &
        /)

      ! Graphics list for graphLevel = "limited"
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsChanLim(0) = ''
      ! Unconditional variables
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsChanLimDef(1) = (/         &
        'dummy'                                                                 &
        /)

      ! Additional graphics list for graphLevel = "full"
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsChanFull(4) = (/           &
        'mcpr', 'peakClad', 'xcrit', 'xequil'                                   &
```

```
        /)
      ! Unconditional variables
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsChanFullDef(4) = (/        &
        'mcpr', 'peakClad', 'xcrit', 'xequil'                                   &
        /)

!!!!!!!!!!!!!!!!!!!!!!!!!  Fill component variables  !!!!!!!!!!!!!!!!!!!!!!!!!

      ! All graphics lists for static variables

      CHARACTER(LEN=13), PARAMETER        :: XtvSVarsFillMin(0) = ''
      CHARACTER(LEN=13), PARAMETER        :: XtvSVarsFillLim(1) = (/            &
        'vol'                                                                   &
        /)
      CHARACTER(LEN=13), PARAMETER        :: XtvSVarsFillFull(0) = ''

      ! Dynamic variables

      ! Graphics list for graphLevel = "minimal"
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsFillMin(0) = ''
      ! Unconditional variables
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsFillMinDef(1) = (/         &
        'dummy'                                                                 &
        /)

      ! Graphics list for graphLevel = "limited"
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsFillLim(21) = (/           &
        'alpn', 'b10MFR', 'concS', 'concW', 'enth', 'famass', 'fxmass', 'm',    &
        'massC', 'mB10C', 'mfr', 'mFrB10', 'mNCGC', 'mSolC', 'pan', 'pn',       &
        'solutMFR', 'tln', 'tvn', 'vln', 'vvn'                                  &
        /)
      ! Unconditional variables
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsFillLimDef(12) = (/        &
        'alpn', 'enth', 'famass', 'fxmass', 'massC', 'mNCGC', 'pan', 'pn',      &
        'tln', 'tvn', 'vln', 'vvn'                                              &
        /)
      ! Conditional variables for Namelist option iSolut != 0
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsFillLimIsolut(4) = (/      &
        'concS', 'concW', 'mSolC', 'solutMFR'                                   &
        /)
      ! Conditional variables for Namelist option iSolut != 0 & solveB10 = .TRUE.
      CHARACTER(LEN=13), PARAMETER     :: XtvDVarsFillLimIsolutsolveB10(3) = (/ &
        'b10MFR', 'mB10C', 'mFrB10'                                             &
        /)
      ! Conditional variables for Namelist options nTraceG != 0 or nTraceL != 0
      ! xor Namelist option igas > 11 (therefore numberOfNCGases > 1)
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsFillLimItrace(2) = (/      &
        'm', 'mfr'                                                              &
        /)

      ! Additional graphics list for graphLevel = "full"
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsFillFull(4) = (/           &
        'b10Sppm', 'b10Wppm', 'cSppm', 'cWppm'                                  &
        /)
      ! Unconditional variables
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsFillFullDef(1) = (/        &
        'dummy'                                                                 &
        /)
      ! Conditional variables for Namelist option iSolut != 0
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsFillFullIsolut(4) = (/     &
        'b10Sppm', 'b10Wppm', 'cSppm', 'cWppm'                                  &
        /)
```

```
!!!!!!!!!!!!!!!!!!!!!!!!!  Power component variables  !!!!!!!!!!!!!!!!!!!!!!!!!

      ! All graphics lists for static variables

      CHARACTER(LEN=13), PARAMETER        :: XtvSVarsPowerMin(0) = ''
      CHARACTER(LEN=13), PARAMETER        :: XtvSVarsPowerLim(0) = ''
      CHARACTER(LEN=13), PARAMETER        :: XtvSVarsPowerFull(0) = ''

      ! Dynamic variables

      ! Graphics list for graphLevel = "minimal"
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsPowerMin(0) = ''
      ! Unconditional variables
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsPowerMinDef(1) = (/        &
        'dummy'                                                                 &
        /)

      ! Graphics lists for graphLevel = "limited"
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsPowerLim(38) = (/          &
        'aldelk', 'alpCAvg', 'alreac', 'borAvgC', 'borAvgD', 'borMFS', 'borMFW',&
        'dbdelk', 'dbreac', 'denCAvg', 'ecrpCore', 'ecrpMax', 'fuelAvgT',       &
        'mcprc', 'pctAvg', 'pctHot', 'pgdelk', 'pgreac', 'powDecay', 'powFiss', &
        'powModerChan', 'powModerTot', 'powModerVess', 'powWaterRod', 'rmckn',  &
        'rpower', 'sqTFCAvg', 'tcdelk', 'tcreac', 'tCoolAvg', 'tfdelk',         &
        'tfreac', 'tFuelAvg', 'tLiqCAvg', 'tpower', 'tramax', 'trhmax',         &
        'totReact'                                                              &
        /)
      ! Unconditional variables
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsPowerLimDef(17) = (/       &
        'ecrpCore', 'ecrpMax', 'fuelAvgT', 'mcprc', 'pctAvg', 'pctHot',         &
        'powDecay', 'powFiss', 'powModerChan', 'powModerTot', 'powModerVess',   &
        'powWaterRod', 'rmckn', 'rpower', 'tpower', 'tramax', 'trhmax'          &
        /)
      ! Conditional variables for option irpwty > 10, ifbtyp(1,2,3)=0 & ibu(4)>=0
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsPowerLimIrpwty(15) = (/    &
        'aldelk', 'alpCAvg', 'alreac', 'borAvgD', 'dbdelk', 'dbreac', 'pgdelk', &
        'pgreac', 'tcdelk', 'tcreac', 'tCoolAvg', 'tfdelk', 'tfreac',           &
        'tFuelAvg', 'totReact'                                                  &
        /)
      ! Alternative variables for option irpwty > 10, ifbtyp(1,2,3)=1 & ibu(4)<0
      ! (1)'sqTFCAvg' is alternative to 'tFuelAvg' (14 in XtvDVarsPowerLimIrpwty)
      ! (2)'tLiqCAvg' is alternative to 'tCoolAvg' (11 in XtvDVarsPowerLimIrpwty)
      ! (3)'denCAvg'  is alternative to 'alpCAvg'  (2  in XtvDVarsPowerLimIrpwty)
      ! (4)'borAvgC'  is alternative to 'borAvgD'  (4  in XtvDVarsPowerLimIrpwty)
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsPowerLimIrpwtyAlt(4) = (/  &
        'sqTFCAvg', 'tLiqCAvg', 'denCAvg', 'borAvgC'                            &
        /)
      ! Alternative variables for option irpwty > 10, ibu(4)=2 & iSolut = 0,1,2
      ! (1)'borMFS' is alternative to 'borAvgC' (or 'borAvgD') when iSolut != 1
      ! (2)'borMFW' is alternative to 'borAvgC' (or 'borAvgD') when iSolut  = 1
      CHARACTER(LEN=13), PARAMETER      :: XtvDVarsPowerLimIrpwtyAltIbu(2) = (/ &
        'borMFS', 'borMFW'                                                      &
        /)

      ! Additional graphics list for graphLevel = "full"
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsPowerFull(0) = ''
      ! Unconditional variables
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsPowerFullDef(1) = (/       &
        'dummy'                                                                 &
        /)

!!!!!!!!!!!!!!!!!!  Radiation enclosure component variables  !!!!!!!!!!!!!!!!!!
```

```
      ! All graphics lists for static variables

      CHARACTER(LEN=13), PARAMETER        :: XtvSVarsRadEncMin(0) = ''
      CHARACTER(LEN=13), PARAMETER        :: XtvSVarsRadEncLim(0) = ''
      CHARACTER(LEN=13), PARAMETER        :: XtvSVarsRadEncFull(0) = ''

      ! Dynamic variables

      ! Graphics list for graphLevel = "minimal"
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsRadEncMin(0) = ''
      ! Unconditional variables
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsRadEncMinDef(1) = (/       &
        'dummy'                                                                 &
        /)

      ! Graphics list for graphLevel = "limited"
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsRadEncLim(1) = (/          &
        'qrad'                                                                  &
        /)
      ! Unconditional variables
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsRadEncLimDef(1) = (/       &
        'qrad'                                                                  &
        /)

      ! Additional graphics list for graphLevel = "full"
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsRadEncFull(0) = ''
      ! Unconditional variables
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsRadEncFullDef(1) = (/      &
        'dummy'                                                                 &
        /)

!!!!!!!!!!!!!!!!!!!!!!  Fluid power component variables  !!!!!!!!!!!!!!!!!!!!!!

      ! All graphics lists for static variables

      CHARACTER(LEN=13), PARAMETER        :: XtvSVarsFlPowerMin(0) = ''
      CHARACTER(LEN=13), PARAMETER        :: XtvSVarsFlPowerLim(0) = ''
      CHARACTER(LEN=13), PARAMETER        :: XtvSVarsFlPowerFull(0) = ''

      ! Dynamic variables

      ! Graphics list for graphLevel = "minimal"
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsFlPowerMin(0) = ''
      ! Unconditional variables
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsFlPowerMinDef(1) = (/      &
        'dummy'                                                                 &
        /)

      ! Graphics list for graphLevel = "limited"
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsFlPowerLim(2) = (/         &
        'powb', 'powd'                                                          &
        /)
      ! Unconditional variables
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsFlPowerLimDef(2) = (/      &
        'powb', 'powd'                                                          &
        /)

      ! Additional graphics list for graphLevel = "full"
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsFlPowerFull(0) = ''
      ! Unconditional variables
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsFlPowerFullDef(1) = (/     &
        'dummy'                                                                 &
        /)
```

```
!!!!!!!!!!!!!!!!!!!!  Heat structure component variables  !!!!!!!!!!!!!!!!!!!!

!!!!!!!!!!!!!!!!!!!!  HtStr

      ! All graphics lists for static variables

      CHARACTER(LEN=13), PARAMETER        :: XtvSVarsHtStrMin(0) = ''
      CHARACTER(LEN=13), PARAMETER        :: XtvSVarsHtStrLim(0) = ''
      CHARACTER(LEN=13), PARAMETER        :: XtvSVarsHtStrFull(0) = ''

      ! Dynamic variables

      ! Graphics list for graphLevel = "minimal"
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsHtStrMin(0) = ''
      ! Unconditional variables
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsHtStrMinDef(1) = (/        &
        'dummy'                                                                 &
        /)

      ! Graphics lists for graphLevel = "limited"
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsHtStrLim(41) = (/          &
        'amh2', 'avgFuelT', 'axialStress', 'bottomQF', 'cladThick',             &
        'dGapRelocate', 'ecrpLimit', 'ecrpMax', 'eFuelDens', 'eFuelSwell',      &
        'gapEffDim', 'gapWidth', 'hContact', 'heatingR', 'hGapGas', 'hGapRad',  &
        'hoopStress', 'kGasGap', 'peakClad', 'pInt', 'qppi', 'qppo', 'qwrx',    &
        'rdzNperm', 'tClad', 'tempJump', 'tFuelAvg', 'tGap', 'topQF',           &
        'tPlastic', 'tpowi', 'tpowo', 'tramax', 'trhmax', 'tRupture', 'tsurfi', &
        'tsurfo', 'uCladElastic', 'uCladPlastic', 'uFuelThermal', 'uRClad'      &
        /)
      ! Unconditional variables
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsHtStrLimDef(14) = (/       &
        'avgFuelT', 'bottomQF', 'peakClad', 'qppi', 'qppo', 'rdzNperm',         &
        'tFuelAvg', 'topQF', 'tpowi', 'tpowo', 'tramax', 'trhmax', 'tsurfi',    &
        'tsurfo'                                                                &
        /)
      ! Conditional variables for option nmwrx /= 0
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsHtStrLimNmwrx(5) = (/      &
        'amh2', 'cladThick', 'ecrpLimit', 'ecrpMax', 'qwrx'                     &
        /)
      ! Conditional variables for option nfci /= 0
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsHtStrLimNfci(19) = (/      &
        'axialStress', 'dGapRelocate', 'eFuelDens', 'eFuelSwell', 'gapEffDim',  &
        'gapWidth', 'hContact', 'heatingR', 'hGapGas', 'hGapRad', 'hoopStress', &
        'kGasGap', 'pInt', 'tClad', 'tempJump', 'tGap', 'uCladElastic',         &
        'uFuelThermal', 'uRClad'                                                &
        /)
      ! Conditional variables for option nfci > 1
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsHtStrLimNfci2(3) = (/      &
        'uCladPlastic', 'tPlastic', 'tRupture'                                  &
        /)

      ! Additional graphics lists for graphLevel = "full"
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsHtStrFull(11) = (/         &
        'alpri', 'alpro', 'enhnfac', 'gridtemp', 'powli', 'powlo', 'powvi',     &
        'powvo', 'qradi', 'qrado', 'rdzN'                                       &
        /)
      ! Unconditional variables
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsHtStrFullDef(10) = (/      &
        'alpri', 'alpro', 'enhnfac', 'powli', 'powlo', 'powvi', 'powvo',        &
        'qradi', 'qrado', 'rdzN'                                                &
        /)
      ! Conditional variables for option ngrids > 0
```

```
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsHtStrFullNgrids(1) = (/    &
        'gridtemp'                                                              &
        /)

!!!!!!!!!!!!!!!!!!!!  HtStrC

      ! All graphics lists for static variables

      CHARACTER(LEN=13), PARAMETER        :: XtvSVarsHtStrCMin(0) = ''
      CHARACTER(LEN=13), PARAMETER        :: XtvSVarsHtStrCLim(0) = ''
      CHARACTER(LEN=13), PARAMETER        :: XtvSVarsHtStrCFull(0) = ''

      ! Graphics list for graphLevel = "minimal"
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsHtStrCMin(0) = ''
      ! Unconditional variables
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsHtStrCMinDef(1) = (/       &
        'dummy'                                                                 &
        /)

      ! Graphics lists for graphLevel = "limited"
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsHtStrCLim(9) = (/          &
        'depthZRI', 'depthZRO', 'ecr50_46', 'eTransi', 'eTranso', 'hgap',       &
        'pgapt', 'tempGas', 'volGas'                                            &
        /)
      ! Unconditional variables
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsHtStrCLimDef(2) = (/       &
        'eTransi', 'eTranso'                                                    &
        /)
      ! Conditional variables for option nmwrx /= 0
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsHtStrCLimNmwrx(3) = (/     &
        'depthZRI', 'depthZRO', 'ecr50_46'                                      &
        /)
      ! Conditional variables for option |nfci| > 0
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsHtStrCLimNfci(2) = (/      &
        'hgap', 'pgapt'                                                         &
        /)
      ! Conditional variables for option |nfci| > 0 and fRGasP is .TRUE.
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsHtStrCLimNfcifRGasP(2) = (/&
        'tempGas', 'volGas'                                                     &
        /)

      ! Additional graphics list for graphLevel = "full"
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsHtStrCFull(30) = (/        &
        'eLossAi', 'eLossAo', 'eLossEi', 'eLossEo', 'hrfgi', 'hrfgo', 'hrfli',  &
        'hrflo', 'hrfvi', 'hrfvo', 'ihtfi', 'ihtfo', 'pLossi', 'pLosso',        &
        'qchfi', 'qchfo', 'rftn', 'tchfi', 'tchfo', 'tmini', 'tmino', 'vFCH',   &
        'vFCrack', 'vFDish', 'vFFRPLen', 'vFGap', 'vFInt', 'vFPore', 'vFRough', &
        'zht'                                                                   &
        /)
      ! Unconditional variables
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsHtStrCFullDef(22) = (/     &
        'eLossAi', 'eLossAo', 'eLossEi', 'eLossEo', 'hrfgi', 'hrfgo', 'hrfli',  &
        'hrflo', 'hrfvi', 'hrfvo', 'ihtfi', 'ihtfo', 'pLossi', 'pLosso',        &
        'qchfi', 'qchfo', 'rftn', 'tchfi', 'tchfo', 'tmini', 'tmino', 'zht'     &
        /)
      ! Conditional variables for option |nfci| > 0
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsHtStrCFullNfci(1) = (/     &
        'dummy'                                                                 &
        /)
      ! Conditional variables for option |nfci| > 0 and fRGasP is .TRUE.
      CHARACTER(LEN=13), PARAMETER      :: XtvDVarsHtStrCFullNfcifRGasP(8) = (/ &
        'vFCH', 'vFCrack', 'vFDish', 'vFFRPLen', 'vFGap', 'vFInt', 'vFPore',    &
        'vFRough'                                                               &
```

```
        /)

!!!!!!!!!!!!!!!!!!!  Feedwater heater component variables  !!!!!!!!!!!!!!!!!!!

      ! All graphics lists for static variables

      CHARACTER(LEN=13), PARAMETER        :: XtvSVarsHeatrMin(0) = ''
      CHARACTER(LEN=13), PARAMETER        :: XtvSVarsHeatrLim(0) = ''
      CHARACTER(LEN=13), PARAMETER        :: XtvSVarsHeatrFull(0) = ''

      ! Dynamic variables

      ! Graphics list for graphLevel = "minimal"
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsHeatrMin(0) = ''
      ! Unconditional variables
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsHeatrMinDef(1) = (/        &
        'dummy'                                                                 &
        /)

      ! Graphics list for graphLevel = "limited"
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsHeatrLim(1) = (/           &
        'liqlev'                                                                &
        /)
      ! Unconditional variables
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsHeatrLimDef(1) = (/        &
        'liqlev'                                                                &
        /)

      ! Additional graphics list for graphLevel = "full"
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsHeatrFull(0) = ''
      ! Unconditional variables
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsHeatrFullDef(1) = (/       &
        'dummy'                                                                 &
        /)

!!!!!!!!!!!!!!!!!!!!!!!  Jet pump component variables  !!!!!!!!!!!!!!!!!!!!!!!

      ! All graphics lists for static variables

      CHARACTER(LEN=13), PARAMETER        :: XtvSVarsJetpMin(0) = ''
      CHARACTER(LEN=13), PARAMETER        :: XtvSVarsJetpLim(0) = ''
      CHARACTER(LEN=13), PARAMETER        :: XtvSVarsJetpFull(0) = ''

      ! Dynamic variables

      ! Graphics list for graphLevel = "minimal"
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsJetpMin(0) = ''
      ! Unconditional variables
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsJetpMinDef(1) = (/         &
        'dummy'                                                                 &
        /)

      ! Graphics list for graphLevel = "limited"
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsJetpLim(2) = (/            &
        'mr', 'nrapp'                                                           &
        /)
      ! Unconditional variables
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsJetpLimDef(2) = (/         &
        'mr', 'nrapp'                                                           &
        /)

      ! Additional graphics list for graphLevel = "full"
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsJetpFull(3) = (/           &
```

```
        'etapp', 'eteff', 'nreff'                                              &
        /)
      ! Unconditional variables
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsJetpFullDef(3) = (/       &
        'etapp', 'eteff', 'nreff'                                              &
        /)

!!!!!!!!!!!!!!!!!!!!!!!!!  Pipe component variables  !!!!!!!!!!!!!!!!!!!!!!!!!

      ! All graphics lists for static variables

      CHARACTER(LEN=13), PARAMETER        :: XtvSVarsPipeMin(0) = ''
      CHARACTER(LEN=13), PARAMETER        :: XtvSVarsPipeLim(0) = ''
      CHARACTER(LEN=13), PARAMETER        :: XtvSVarsPipeFull(0) = ''

      ! Dynamic variables

      ! Graphics list for graphLevel = "minimal"
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsPipeMin(0) = ''
      ! Unconditional variables
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsPipeMinDef(1) = (/        &
        'dummy'                                                                &
        /)

      ! Graphics list for graphLevel = "limited"
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsPipeLim(3) = (/           &
        'cpow', 'qout', 'z'                                                    &
        /)
      ! Unconditional variables
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsPipeLimDef(1) = (/        &
        'cpow'                                                                 &
        /)
      ! Conditional variables for option pipetype = 1, 2, 3 or 9
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsPipeLimPipetype(2) = (/   &
        'qout', 'z'                                                            &
        /)

      ! Additional graphics list for graphLevel = "full"
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsPipeFull(1) = (/          &
        'vflow'                                                                &
        /)
      ! Unconditional variables
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsPipeFullDef(1) = (/       &
        'dummy'                                                                &
        /)
      ! Conditional variables for option pipetype = 1, 2, 3 or 9
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsPipeFullPipetype(1) = (/  &
        'vflow'                                                                &
        /)

!!!!!!!!!!!!!!!!!!!!!!!!  Plenum component variables  !!!!!!!!!!!!!!!!!!!!!!!!

      ! All graphics lists for static variables

      CHARACTER(LEN=13), PARAMETER        :: XtvSVarsPlenMin(0) = ''
      CHARACTER(LEN=13), PARAMETER        :: XtvSVarsPlenLim(1) = (/           &
        'vol'                                                                  &
        /)
      CHARACTER(LEN=13), PARAMETER        :: XtvSVarsPlenFull(0) = ''

      ! Dynamic variables

      ! Graphics list for graphLevel = "minimal"
```

```
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsPlenMin(0) = ''
      ! Unconditional variables
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsPlenMinDef(1) = (/         &
        'dummy'                                                                 &
        /)

      ! Graphics list for graphLevel = "limited"
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsPlenLim(18) = (/           &
        'alpn', 'am', 'concS', 'concW', 'mFrB10', 'pan', 'pn', 'roan', 'roln',  &
        'rom', 'rovn', 'sn', 'solMaxS', 'solMaxW', 'sSolids', 'tln', 'tsat',    &
        'tvn'                                                                   &
        /)
      ! Unconditional variables
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsPlenLimDef(11) = (/        &
        'alpn', 'am', 'pan', 'pn', 'roan', 'roln', 'rom', 'rovn', 'tln', 'tsat',&
        'tvn'                                                                   &
        /)
      ! Conditional variables for Namelist option iSolut != 0
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsPlenLimIsolut(6) = (/      &
        'concS', 'concW', 'sn', 'solMaxS', 'solMaxW', 'sSolids'                 &
        /)
      ! Conditional variables for Namelist option iSolut > 0 & solveB10 = .TRUE.
      CHARACTER(LEN=13), PARAMETER      :: XtvDVarsPlenLimIsolutsolveB10(1) = (/&
        'mFrB10'                                                                &
        /)

      ! Additional graphics list for graphLevel = "full"
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsPlenFull(8) = (/           &
        'b10Sppm', 'b10Wppm', 'cSppm', 'cWppm', 'gamn', 'pStag', 'vlvol',       &
        'vvvol'                                                                 &
        /)
      ! Unconditional variables
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsPlenFullDef(4) = (/        &
        'gamn', 'pStag', 'vlvol', 'vvvol'                                       &
        /)
      ! Conditional variables for Namelist option iSolut > 0
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsPlenFullIsolut(4) = (/     &
        'b10Sppm', 'b10Wppm', 'cSppm', 'cWppm'                                  &
        /)

!!!!!!!!!!!!!!!!!!!!!!!!  Prizer component variables  !!!!!!!!!!!!!!!!!!!!!!!!

      ! All graphics lists for static variables

      CHARACTER(LEN=13), PARAMETER        :: XtvSVarsPrzrMin(0) = ''
      CHARACTER(LEN=13), PARAMETER        :: XtvSVarsPrzrLim(0) = ''
      CHARACTER(LEN=13), PARAMETER        :: XtvSVarsPrzrFull(0) = ''

      ! Dynamic variables

      ! Graphics list for graphLevel = "minimal"
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsPrzrMin(0) = ''
      ! Unconditional variables
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsPrzrMinDef(1) = (/         &
        'dummy'                                                                 &
        /)

      ! Graphics list for graphLevel = "limited"
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsPrzrLim(2) = (/            &
        'qout', 'z'                                                             &
        /)
      ! Unconditional variables
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsPrzrLimDef(2) = (/         &
```

```
        'qout', 'z'                                                          &
        /)

      ! Additional graphics list for graphLevel = "full"
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsPrzrFull(2) = (/          &
        'qin', 'flow'                                                        &
        /)
      ! Unconditional variables
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsPrzrFullDef(2) = (/       &
        'qin', 'flow'                                                        &
        /)

!!!!!!!!!!!!!!!!!!!!!!!!  Pump component variables  !!!!!!!!!!!!!!!!!!!!!!!!!

      ! All graphics lists for static variables

      CHARACTER(LEN=13), PARAMETER        :: XtvSVarsPumpMin(0) = ''
      CHARACTER(LEN=13), PARAMETER        :: XtvSVarsPumpLim(0) = ''
      CHARACTER(LEN=13), PARAMETER        :: XtvSVarsPumpFull(0) = ''

      ! Dynamic variables

      ! Graphics list for graphLevel = "minimal"
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsPumpMin(0) = ''
      ! Unconditional variables
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsPumpMinDef(1) = (/        &
        'dummy'                                                              &
        /)

      ! Graphics list for graphLevel = "limited"
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsPumpLim(3) = (/           &
        'head', 'mflow', 'omegan'                                            &
        /)
      ! Unconditional variables
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsPumpLimDef(3) = (/        &
        'head', 'mflow', 'omegan'                                            &
        /)

      ! Additional graphics list for graphLevel = "full"
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsPumpFull(9) = (/          &
        'alpha', 'delp', 'flow', 'mtorque', 'pumpQFL', 'pumpQFV', 'rho', 'smom',&
        'torque'                                                             &
        /)
      ! Unconditional variables
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsPumpFullDef(9) = (/       &
        'alpha', 'delp', 'flow', 'mtorque', 'pumpQFL', 'pumpQFV', 'rho', 'smom',&
        'torque'                                                             &
        /)

!!!!!!!!!!!!!!!!!!!!!!  Separator component variables  !!!!!!!!!!!!!!!!!!!!!!

      ! All graphics lists for static variables

      CHARACTER(LEN=13), PARAMETER        :: XtvSVarsSepdMin(0) = ''
      CHARACTER(LEN=13), PARAMETER        :: XtvSVarsSepdLim(0) = ''
      CHARACTER(LEN=13), PARAMETER        :: XtvSVarsSepdFull(0) = ''

      ! Dynamic variables

      ! Graphics list for graphLevel = "minimal"
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsSepdMin(0) = ''
      ! Unconditional variables
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsSepdMinDef(1) = (/        &
```

```
        'dummy'                                                                 &
        /)

      ! Graphics list for graphLevel = "limited"
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsSepdLim(3) = (/            &
        'xci', 'xco', 'xcu'                                                     &
        /)
      ! Unconditional variables
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsSepdLimDef(3) = (/         &
        'xci', 'xco', 'xcu'                                                     &
        /)

      ! Additional graphics list for graphLevel = "full"
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsSepdFull(9) = (/           &
        'dAlp', 'deff', 'dpss', 'isSepSep', 'veldis', 'vlev', 'wlev', 'xdin',   &
        'xcrit'                                                                 &
        /)
      ! Unconditional variables
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsSepdFullDef(9) = (/        &
        'dAlp', 'deff', 'dpss', 'isSepSep', 'veldis', 'vlev', 'wlev', 'xdin',   &
        'xcrit'                                                                 &
        /)

!!!!!!!!!!!!!!!!!!!!!!!!  Turbine component variables  !!!!!!!!!!!!!!!!!!!!!!!!

      ! All graphics lists for static variables

      CHARACTER(LEN=13), PARAMETER        :: XtvSVarsTurbMin(0) = ''
      CHARACTER(LEN=13), PARAMETER        :: XtvSVarsTurbLim(0) = ''
      CHARACTER(LEN=13), PARAMETER        :: XtvSVarsTurbFull(0) = ''

      ! Dynamic variables

      ! Graphics list for graphLevel = "minimal"
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsTurbMin(0) = ''
      ! Unconditional variables
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsTurbMinDef(1) = (/         &
        'dummy'                                                                 &
        /)

      ! Graphics list for graphLevel = "limited"
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsTurbLim(2) = (/            &
        'omegt', 'torqt'                                                        &
        /)
      ! Unconditional variables
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsTurbLimDef(2) = (/         &
        'omegt', 'torqt'                                                        &
        /)

      ! Additional graphics list for graphLevel = "full"
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsTurbFull(0) = ''
      ! Unconditional variables
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsTurbFullDef(1) = (/        &
        'dummy'                                                                 &
        /)

!!!!!!!!!!!!!!!!!!!!!!!!  Valve component variables  !!!!!!!!!!!!!!!!!!!!!!!!

      ! All graphics lists for static variables

      CHARACTER(LEN=13), PARAMETER        :: XtvSVarsValveMin(0) = ''
      CHARACTER(LEN=13), PARAMETER        :: XtvSVarsValveLim(0) = ''
      CHARACTER(LEN=13), PARAMETER        :: XtvSVarsValveFull(0) = ''
```

```
      ! Dynamic variables

      ! Graphics list for graphLevel = "minimal"
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsValveMin(0) = ''
      ! Unconditional variables
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsValveMinDef(1) = (/        &
        'dummy'                                                                 &
        /)

      ! Graphics list for graphLevel = "limited"
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsValveLim(10) = (/          &
        'area', 'dOmegadt', 'dtRK', 'fTorq', 'hd', 'mTorq', 'omega', 'pTorq',   &
        'theta', 'thetarad'                                                     &
        /)
      ! Unconditional variables
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsValveLimDef(2) = (/        &
        'area', 'hd'                                                            &
        /)
      ! Conditional variables for option ivty = vSwing
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsValveLimIvty(8) = (/       &
        'dOmegadt', 'dtRK', 'fTorq', 'mTorq', 'omega', 'pTorq', 'theta',        &
        'thetarad'                                                              &
        /)

      ! Additional graphics list for graphLevel = "full"
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsValveFull(0) = ''
      ! Unconditional variables
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsValveFullDef(1) = (/       &
        'dummy'                                                                 &
        /)

!!!!!!!!!!!!!!!!!!!!!!!!  Vessel component variables  !!!!!!!!!!!!!!!!!!!!!!!!

      ! All graphics lists for static variables

      CHARACTER(LEN=13), PARAMETER        :: XtvSVarsVsslMin(0) = ''
      CHARACTER(LEN=13), PARAMETER        :: XtvSVarsVsslLim(1) = (/            &
        'vol'                                                                   &
        /)
      CHARACTER(LEN=13), PARAMETER        :: XtvSVarsVsslFull(0) = ''

      ! Dynamic variables

      ! Graphics list for graphLevel = "minimal"
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsVsslMin(0) = ''
      ! Unconditional variables
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsVsslMinDef(1) = (/         &
        'dummy'                                                                 &
        /)

      ! Graphics list for graphLevel = "limited"
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsVsslLim(84) = (/           &
        'alpn', 'am', 'bubAi-', 'bubVelXR-', 'bubVelYT-', 'bubVelZ-',          &
        'bubVolFrc-', 'cimfr', 'cimfrl', 'cimfrv', 'comfr', 'comfrl', 'comfrv', &
        'concS', 'concW', 'corelq', 'dcflow', 'dclqvl', 'dropAi-', 'dropVelXR-',&
        'dropVelYT-', 'dropVelZ-', 'dropVolFrc-', 'el', 'ev', 'mFrB10',        &
        'mmflxr', 'mmflyt', 'mmflz', 'pan', 'pcore', 'pdc', 'plp', 'pn', 'pup', &
        'qhstot', 'roan', 'roln', 'rom', 'rovn', 'sn', 'solMaxS', 'solMaxW',    &
        'sSolids', 'tcilmf', 'tcivmf', 'tcolmf', 'tcore', 'tcovmf', 'tdc',     &
        'tln', 'tlp', 'tscore', 'tsdc', 'tslp', 'tsn', 'tsup', 'tup', 'tvn',   &
        'vcore', 'vdclq', 'vlnxr', 'vlnyt', 'vlnz', 'vlpliq', 'vlplm', 'vlplq', &
        'vlqmss', 'vmfrl', 'vmfrlxr', 'vmfrlyt', 'vmfrlz', 'vmfrv', 'vmfrvxr',  &
```

```
  'vmfrvyt', 'vmfrvz', 'vsflow', 'vupliq', 'vuplm', 'vvnxr', 'vvnyt',     &
  'vvnz', 'xg', 'xl'                                                      &
  /)
! Unconditional variables
CHARACTER(LEN=13), PARAMETER        :: XtvDVarsVsslLimDef(25) = (/        &
  'alpn', 'am', 'el', 'ev', 'mmflxr', 'mmflyt', 'mmflz', 'pan', 'pn',     &
  'qhstot', 'roan', 'roln', 'rom', 'rovn', 'tln', 'tsn', 'tvn', 'vlnxr',  &
  'vlnyt', 'vlnz', 'vlqmss', 'vsflow', 'vvnxr', 'vvnyt', 'vvnz'           &
  /)
! Conditional variables for Namelist option imfr = 1
CHARACTER(LEN=13), PARAMETER        :: XtvDVarsVsslLimImfr1(2) = (/       &
  'vmfrl', 'vmfrv'                                                        &
  /)
! Conditional variables for Namelist option imfr = 3
CHARACTER(LEN=13), PARAMETER        :: XtvDVarsVsslLimImfr3(6) = (/       &
  'vmfrlxr', 'vmfrlyt', 'vmfrlz', 'vmfrvxr', 'vmfrvyt', 'vmfrvz'          &
  /)
! Conditional variables for Namelist option iSolut != 0
CHARACTER(LEN=13), PARAMETER        :: XtvDVarsVsslLimIsolut(6) = (/      &
  'concS', 'concW', 'sn', 'solMaxS', 'solMaxW', 'sSolids'                 &
  /)
! Conditional variables for Namelist option iSolut != 0 & solveB10 = .TRUE.
CHARACTER(LEN=13), PARAMETER      :: XtvDVarsVsslLimIsolutsolveB10(1) = (/&
  'mFrB10'                                                                &
  /)
! Conditional variables for Namelist options nTraceG != 0
! xor Namelist option igas > 11 (therefore numberOfNCGases > 1)
CHARACTER(LEN=13), PARAMETER        :: XtvDVarsVsslLimNtraceG(1) = (/     &
  'xg'                                                                    &
  /)
! Conditional variables for Namelist options nTraceL != 0
CHARACTER(LEN=13), PARAMETER        :: XtvDVarsVsslLimNtraceL(1) = (/     &
  'xl'                                                                    &
  /)
! Conditional variables for bubbles when Namelist option disfields > 0
CHARACTER(LEN=13), PARAMETER     :: XtvDVarsVsslLimDisfieldsB(5) =  (/    &
   'bubAi-',  'bubVelXR-',  'bubVelYT-',  'bubVelZ-',  'bubVolFrc-'       &
  /)
! Conditional variables for droplets when Namelist option disfields > 0
CHARACTER(LEN=13), PARAMETER     :: XtvDVarsVsslLimDisfieldsD(5) =  (/    &
  'dropAi-', 'dropVelXR-', 'dropVelYT-', 'dropVelZ-', 'dropVolFrc-'       &
  /)
! Conditional variables for Namelist option disfields > 0 and nBubbles = 1
CHARACTER(LEN=13), PARAMETER      :: XtvDVarsVsslLimDisfieldsnB1(1) =  (/ &
  'dummy'                                                                 &
  /)
! Conditional variables for Namelist option disfields > 0 and nBubbles = 2
CHARACTER(LEN=13), PARAMETER      :: XtvDVarsVsslLimDisfieldsnB2(1) =  (/ &
  'dummy'                                                                 &
  /)
! Conditional variables for option icrr != 0
CHARACTER(LEN=13), PARAMETER        :: XtvDVarsVsslLimIcrr(1) = (/        &
  'vcore'                                                                 &
  /)
! Conditional variables for option idcr != 0
CHARACTER(LEN=13), PARAMETER        :: XtvDVarsVsslLimIdcr(7) = (/        &
  'dcflow', 'dclqvl', 'pdc', 'tdc', 'tsdc', 'vdclq', 'vlplq'              &
  /)
! Conditional variables for option ilcsp != 0
CHARACTER(LEN=13), PARAMETER        :: XtvDVarsVsslLimIlcsp(5) = (/       &
  'plp', 'vlpliq', 'vlplm', 'tlp', 'tslp'                                 &
  /)
! Conditional variables for option iucsp != 0
```

```
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsVsslLimIucsp(19) = (/      &
        'cimfr', 'cimfrl', 'cimfrv', 'comfr', 'comfrl', 'comfrv', 'corelq',     &
        'pcore', 'pup', 'tcilmf', 'tcivmf', 'tcolmf', 'tcore', 'tcovmf',        &
        'tscore', 'tsup', 'tup', 'vupliq', 'vuplm'                              &
        /)

      ! Additional graphics list for graphLevel = "full"
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsVsslFull(38) = (/          &
        'alpA', 'alpB', 'alpzE', 'alven', 'alvn', 'b10Sppm', 'b10Wppm', 'chtan',&
        'chtin', 'cixr', 'ciyt', 'ciz', 'cl', 'cSppm', 'cv', 'cWppm', 'dpxri',  &
        'dpyti', 'dpzi', 'gamn', 'hgam', 'phIL', 'pStag', 'qsl', 'visl', 'visv',&
        'vLev', 'vlvol', 'volM', 'volP', 'vvvol', 'wfhf', 'wflxr', 'wflyt',     &
        'wflz', 'wfvxr', 'wfvyt', 'wfvz'                                        &
        /)
      ! Unconditional variables
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsVsslFullDef(24) = (/       &
        'alven', 'alvn', 'chtan', 'chtin', 'cixr', 'ciyt', 'ciz', 'cl', 'cv',   &
        'gamn', 'hgam', 'pStag', 'qsl', 'visl', 'visv', 'vlvol', 'vvvol',       &
        'wfhf', 'wflxr', 'wflyt', 'wflz', 'wfvxr', 'wfvyt', 'wfvz'              &
        /)
      ! Conditional variables for option nolt < 1
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsVsslFullNolt0(10) = (/     &
        'alpA', 'alpB', 'alpzE', 'dpxri', 'dpyti', 'dpzi', 'phIL', 'vLev',      &
        'volM', 'volP'                                                          &
        /)
      ! Conditional variables for Namelist option iSolut <> 0
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsVsslFullIsolut(4) = (/     &
        'b10Sppm', 'b10Wppm', 'cSppm', 'cWppm'                                  &
        /)

!!!!!!!!!!!!!!!!!!!!!!!!  Contan component variables  !!!!!!!!!!!!!!!!!!!!!!!!

      ! All graphics lists for static variables

      CHARACTER(LEN=13), PARAMETER        :: XtvSVarsContMin(0) = ''
      CHARACTER(LEN=13), PARAMETER        :: XtvSVarsContLim(1) = (/            &
        'vol'                                                                   &
        /)
      CHARACTER(LEN=13), PARAMETER        :: XtvSVarsContFull(0) = ''

      ! Dynamic variables

      ! Graphics list for graphLevel = "minimal"
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsContMin(0) = ''
      ! Unconditional variables
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsContMinDef(1) = (/         &
        'dummy'                                                                 &
        /)

      ! Graphics list for graphLevel = "limited"
      ! REMARK: Twice 'rmd' as per original compartment variable description
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsContLim(101) = (/          &
        'alpn', 'alven', 'alvn', 'apool', 'chtan', 'chtin', 'cl', 'concps',     &
        'concpw', 'cpl', 'cpv', 'cv', 'd', 'dpdt', 'droldt', 'drovdt', 'ea',    &
        'ed', 'efc', 'el', 'enflo', 'enss', 'ev', 'filmTh', 'hfgp', 'hli',      &
        'hlmfr', 'hlo', 'htc', 'htmfr', 'hvi', 'hvo', 'intQSurf', 'mfcnds',     &
        'nDrop', 'p', 'pa', 'qcl', 'qcld', 'qcnds', 'qerr', 'qgas', 'qLiqHT',   &
        'qSurf', 'qVapHT', 'radDrop', 'rma', 'rmain', 'rmb', 'rmb2', 'rmd',     &
        'rmd', 'rmdaf', 'rmdap', 'rmdas', 'rmdlf', 'rmdlp', 'rmdls', 'rmdsf',   &
        'rmdsin', 'rmdsp', 'rmdss', 'rmdot', 'rmdota', 'rmdotd', 'rmdotf',      &
        'rmdotl', 'rmdots', 'rmfc', 'rml', 'rmlin', 'rms', 'roa', 'rod', 'rol', &
        'rov', 'sig', 'solMaxs', 'solMaxW', 'sSolids', 'td', 'tfc', 'tl', 'tli',&
        'tlo', 'tsatp', 'tsats', 'tv', 'tvi', 'tvo', 'ud', 'udotd', 'udotf',    &
```

```
  'udotl', 'udotv', 'ufc', 'ul', 'uv', 'velDrop', 'visl', 'visv'          &
  /)
! Unconditional variables
CHARACTER(LEN=13), PARAMETER        :: XtvDVarsContLimDef(1) = (/         &
  'dummy'                                                                &
  /)
! Conditional variables for compartment, i.e (contanTab%ncomt .NE. 0)
CHARACTER(LEN=13), PARAMETER        :: XtvDVarsContLimCompart(54) = (/    &
  'alpn', 'alven', 'alvn', 'apool', 'chtan', 'chtin', 'cl', 'concps',    &
  'concpw', 'cpl', 'cpv', 'cv', 'd', 'dpdt', 'droldt', 'drovdt', 'ea',   &
  'efc', 'el', 'ev', 'hfgp', 'hlmfr', 'htmfr', 'p', 'pa', 'rma', 'rmb',  &
  'rmdota', 'rmdotf', 'rmdotl', 'rmdots', 'rmfc', 'rml', 'rms', 'roa',   &
  'rol', 'rov', 'sig', 'solMaxs', 'solMaxW', 'sSolids', 'tfc', 'tl',     &
  'tsatp', 'tsats', 'tv', 'udotf', 'udotl', 'udotv', 'ufc', 'ul', 'uv',  &
  'visl', 'visv'                                                         &
  /)
! Conditional variables for 2d droplet field arrays
! i.e. (contanDropOn .AND. contanTab%nField > 0_sik)
! REMARK: Twice 'rmd' as per original compartment variable description
CHARACTER(LEN=13), PARAMETER        :: XtvDVarsContLimDropOn(12) = (/     &
  'ed', 'nDrop', 'radDrop', 'rmb2', 'rmd', 'rmd', 'rmdotd', 'rod', 'td', &
  'ud', 'udotd', 'velDrop'                                               &
  /)
! Conditional variables for cooler, i.e. (contanTab%ncool .NE. 0)
CHARACTER(LEN=13), PARAMETER        :: XtvDVarsContLimCooler(3) = (/      &
  'htc', 'qcl', 'qcld'                                                   &
  /)
! Conditional variables for fan cooler, i.e. (contanTab%nfanc .NE. 0)
CHARACTER(LEN=13), PARAMETER        :: XtvDVarsContLimFCool(4) = (/       &
  'mfcnds', 'qcnds', 'qerr', 'qgas'                                      &
  /)
! Conditional variables for passive junction, i.e. (contanTab%njct .NE. 0)
CHARACTER(LEN=13), PARAMETER        :: XtvDVarsContLimJct(4) = (/         &
  'rmdap', 'rmdlp', 'rmdot', 'rmdsp'                                     &
  /)
! Conditional variables for forced junction, i.e. (contanTab%njctf .NE. 0)
CHARACTER(LEN=13), PARAMETER        :: XtvDVarsContLimJctf(3) = (/        &
  'rmdaf', 'rmdlf', 'rmdsf'                                              &
  /)
! Conditional variables for source/sink, i.e. (contanTab%njcts .NE. 0)
CHARACTER(LEN=13), PARAMETER        :: XtvDVarsContLimJcts(8) = (/        &
  'enflo', 'enss', 'rmain', 'rmdas', 'rmdls', 'rmdsin', 'rmdss', 'rmlin' &
  /)
! Conditional variables for HS, i.e. (contanTab%nhs > 0)
CHARACTER(LEN=13), PARAMETER        :: XtvDVarsContLimHS(10) = (/         &
  'hli', 'hlo', 'hvi', 'hvo', 'intQSurf', 'qSurf', 'tli', 'tlo', 'tvi',  &
  'tvo'                                                                  &
  /)
! Conditional variables for HS connected to CONTAN fluid compartments
! i.e. (contanTab%nhsp > 0_sik)
CHARACTER(LEN=13), PARAMETER        :: XtvDVarsContLimTHS(3) = (/         &
  'filmTh', 'qLiqHT', 'qVapHT'                                           &
  /)

! Additional graphics list for graphLevel = "full"
CHARACTER(LEN=13), PARAMETER        :: XtvDVarsContFull(19) = (/          &
  'condFrac', 'filmDR', 'filmMT', 'fr', 'hLiq', 'hVap', 'mfloc', 'mflov', &
  'nfin', 'pclin', 'rmf', 'tcl', 'tclin', 'tf', 'ti', 'tifc', 'tsurf',   &
  'twali', 'twalo'                                                       &
  /)
! Unconditional variables
CHARACTER(LEN=13), PARAMETER        :: XtvDVarsContFullDef(1) = (/        &
  'dummy'                                                                &
```

```
        /)
      ! Conditional variables for compartment, i.e (contanTab%ncomt .NE. 0)
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsContFullCompart(1) = (/     &
        'dummy'                                                                  &
        /)
      ! Conditional variables for 2d droplet field arrays
      ! i.e. (contanDropOn .AND. contanTab%nField > 0_sik)
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsContFullDropOn(1) = (/      &
        'dummy'                                                                  &
        /)
      ! Conditional variables for cooler, i.e. (contanTab%ncool .NE. 0)
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsContFullCooler(1) = (/      &
        'dummy'                                                                  &
        /)
      ! Conditional variables for fan cooler, i.e. (contanTab%nfanc .NE. 0)
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsContFullFCool(9) = (/       &
        'mfloc', 'mflov', 'nfin', 'pclin', 'tcl', 'tclin', 'tifc', 'twali',      &
        'twalo'                                                                  &
        /)
      ! Conditional variables for passive junction, i.e. (contanTab%njct .NE. 0)
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsContFullJct(1) = (/         &
        'fr'                                                                     &
        /)
      ! Conditional variables for forced junction, i.e. (contanTab%njctf .NE. 0)
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsContFullJctf(1) = (/        &
        'dummy'                                                                  &
        /)
      ! Conditional variables for source/sink, i.e. (contanTab%njcts .NE. 0)
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsContFullJcts(1) = (/        &
        'dummy'                                                                  &
        /)
      ! Conditional variables for HS, i.e. (contanTab%nhs > 0)
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsContFullHS(1) = (/          &
        'dummy'                                                                  &
        /)
      ! Conditional variables for HS connected to CONTAN fluid compartments
      ! i.e. (contanTab%nhsp > 0_sik)
      CHARACTER(LEN=13), PARAMETER        :: XtvDVarsContFullTHS(9) = (/         &
        'condFrac', 'filmDR', 'filmMT', 'hLiq', 'hVap', 'rmf', 'tf', 'ti',       &
        'tsurf'                                                                  &
        /)

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

CONTAINS


      SUBROUTINE InitCustomGraphLevel(graphLevel)
!
!     This subroutine sets the filtering option flag "graphFltFlag" ON or OFF
!     depending on content of namelist variables "graphCustVar" and "graphCustId".
!
         IMPLICIT NONE
!
!     Dummy Variable Declarations
         CHARACTER(len=8), INTENT(INOUT) :: graphLevel
!
!     Local Variable Declarations
         INTEGER(sik) :: i

         graphFltFlag = 0

         ! Add a summary message in the output files
```

```
      IF (graphCustVar(1) /= '') THEN

          NAMELIST / graphOpts / graphLevel, graphCustVar, graphCustId

          DO i = 1, 3
              WRITE (outUnt(i), *) "Summary of graphics file NAMELIST options:"
              WRITE (outUnt(i), graphOpts)
          END DO

          ! Set filtering flag ON only when relevant
          IF ( (graphCustId(1) /= '') .AND. (graphLevel /= 'full') ) THEN
              graphFltFlag  = 1
              DO i = 1, 3
                 WRITE (outUnt(i), *) "User-defined graphics file output selected."
                 WRITE (outUnt(i), *) "graphFltFlag is set ON"
                 !WRITE (outUnt(i), *) graphFltFlag
              END DO
          ELSE
              DO i = 1, 3
                 WRITE (outUnt(i), *) "graphFltFlag is set OFF"
                 !WRITE (outUnt(i), *) graphFltFlag
              END DO
          END IF

      END IF

   END SUBROUTINE InitCustomGraphLevel


   SUBROUTINE XtvVarFilterOut(compId, varName, graphLevel, FltOutFlag)
!
!     This subroutine determines if the supplied graph variable varName
!     is to be filtered out of the Xtv file.
!
!     BEGIN MODULE USE
      USE GlobalDat, ONLY: sik
      USE XtvData,   ONLY: curCmpIdx, xtvCompList
!
      IMPLICIT NONE
!
!     Dummy Variable Declarations
      INTEGER(sik),     INTENT(IN)  :: compId
      CHARACTER(*),     INTENT(IN)  :: varName
      CHARACTER(LEN=8), INTENT(IN)  :: graphLevel
      INTEGER(sik),     INTENT(OUT) :: FltOutFlag
!
!     Local Variable Declarations

      FltOutFlag = 0_sik

      IF ( ALL(compId /= graphCustId) ) THEN

         IF (graphLevel == 'minimal') THEN
           IF ( ALL(varName /= XtvDVarsMin) .AND. ANY(varName == XtvDVarsLim) ) THEN
               FltOutFlag = 1_sik
            END IF
           IF ( ALL(varName /= XtvDVarsMin) .AND. ANY(varName == XtvDVarsFull) ) THEN
               FltOutFlag = 1_sik
            END IF
         ELSE IF (graphLevel == 'limited') THEN
           IF ( ALL(varName /= XtvDVarsLim) .AND. ANY(varName == XtvDVarsFull) ) THEN
               FltOutFlag = 1_sik
            END IF
```

```
            ELSE ! graphLevel == 'full'
               IF ( ANY(varName == XtvDVarsFull) ) THEN
                  FltOutFlag = 0_sik
               END IF
            END IF

         END IF

         IF ( ANY(compId == graphCustId) ) THEN

            IF (graphLevel == 'minimal') THEN
              IF ( ANY(varName == XtvDVarsLim) .AND. ALL(varName /= graphCustVar) ) THEN
                  FltOutFlag = 1_sik
               END IF
              IF ( ANY(varName == XtvDVarsFull) .AND. ALL(varName /= graphCustVar) ) THEN
                  FltOutFlag = 1_sik
               END IF
            ELSE IF (graphLevel == 'limited') THEN
              IF ( ANY(varName == XtvDVarsFull) .AND. ALL(varName /= graphCustVar) ) THEN
                  FltOutFlag = 1_sik
               END IF
            ELSE ! graphLevel == 'full'
               IF ( ANY(varName == XtvDVarsFull) ) THEN
                  FltOutFlag = 0_sik
               END IF
            END IF

         END IF

         RETURN

      END SUBROUTINE XtvVarFilterOut


      SUBROUTINE XtvVarListMerge2A(xvlist,xvlistplus)
!
!     This subroutine merges the two variables lists provided as argument
!     (version with 2 arguments)
!
!     BEGIN MODULE USE
         USE GlobalDat, ONLY: sik
!
         IMPLICIT NONE
!
!     Dummy Variable Declarations
         CHARACTER(LEN=13), ALLOCATABLE, DIMENSION(:), INTENT(INOUT) :: xvlist
         CHARACTER(LEN=13),              DIMENSION(:), INTENT(IN)    :: xvlistplus
!
!     Local Variable Declarations
         INTEGER(sik)                                    :: i, ierr, j, k, siz
         CHARACTER(LEN=13), ALLOCATABLE, DIMENSION(:) :: XtvDVarsBuffer

         ierr = 0
         k    = 0

         DO i = 1, SIZE(xvlist)
            DO j = 1, SIZE(xvlistplus)
               IF (xvlist(i) == xvlistplus(j)) THEN
                  k = k + 1
               END IF
            END DO
         END DO
```

```
      siz = SIZE(xvlist) + SIZE(xvlistplus) - k

      ALLOCATE(XtvDVarsBuffer(siz), STAT=ierr)

      j = 0

      DO i = 1, SIZE(xvlist)
         j = j + 1
         XtvDVarsBuffer(j) = xvlist(i)
      END DO

      DO i = 1, SIZE(xvlistplus)
         IF ( ALL(xvlistplus(i) /= xvlist) ) THEN
            j = j + 1
            XtvDVarsBuffer(j) = xvlistplus(i)
         END IF
      END DO

      DEALLOCATE(xvlist, STAT=ierr)
      ALLOCATE(xvlist(siz), STAT=ierr)
      xvlist = XtvDVarsBuffer

      DEALLOCATE(XtvDVarsBuffer, STAT=ierr)

   END SUBROUTINE XtvVarListMerge2A


   SUBROUTINE XtvVarListMerge3A(xvlist,xvlistplus,ncgas)
!
!     This subroutine merges the two variables lists provided as argument
!     (version with 3 arguments, to accommodate for non-condensale gases variable names)
!
!     BEGIN MODULE USE
      USE GlobalDat, ONLY: sik
!
      IMPLICIT NONE
!
!     Dummy Variable Declarations
      CHARACTER(LEN=13), ALLOCATABLE, DIMENSION(:), INTENT(INOUT) :: xvlist
      CHARACTER(LEN=13),              DIMENSION(:), INTENT(IN)    :: xvlistplus
      CHARACTER(LEN=3),                             INTENT(IN)    :: ncgas
!
!     Local Variable Declarations
      INTEGER(sik)                                    :: ierr, j
      CHARACTER(LEN=13), ALLOCATABLE, DIMENSION(:) :: XtvDVarsBuffer
      CHARACTER(LEN=13), ALLOCATABLE, DIMENSION(:) :: XtvDVarsOptions

      ierr = 0

      ALLOCATE(XtvDVarsOptions(SIZE(xvlistplus)), STAT=ierr)

      DO j = 1, SIZE(xvlistplus)
         XtvDVarsOptions(j) = TRIM(xvlistplus(j)) // ncgas
      END DO

      CALL XtvVarListMerge2A(xvlist,XtvDVarsOptions)

      DEALLOCATE(XtvDVarsOptions, STAT=ierr)

   END SUBROUTINE XtvVarListMerge3A


   SUBROUTINE XtvVarListMerge4A(xvlist,xvlistplus,fluid,iarg)
```

```
!
!     This subroutine merges the two variables lists provided as argument
!     (version with 4 arguments, to accommodate for trace species variable names)
!
!     BEGIN MODULE USE
         USE GlobalDat, ONLY: sik
!
         IMPLICIT NONE
!
!     Dummy Variable Declarations
         CHARACTER(LEN=13), ALLOCATABLE, DIMENSION(:), INTENT(INOUT) :: xvlist
         CHARACTER(LEN=13),              DIMENSION(:), INTENT(IN)    :: xvlistplus
         CHARACTER(LEN=3),                             INTENT(IN)    :: fluid
         INTEGER(sik),                                 INTENT(IN)    :: iarg
!
!     Local Variable Declarations
         INTEGER(sik)                                  :: ierr, j
         CHARACTER(LEN=13), ALLOCATABLE, DIMENSION(:) :: XtvDVarsBuffer
         CHARACTER(LEN=13), ALLOCATABLE, DIMENSION(:) :: XtvDVarsOptions

         INTEGER(sik)                                  :: itens, iones
         CHARACTER(LEN=1), DIMENSION(0:9), PARAMETER  :: ndigs = (/ '0', '1', '2', '3',
'4',&
        &                                                           '5', '6', '7', '8',
'9' /)
         CHARACTER(LEN=2)                              :: num

         ierr = 0

         ALLOCATE(XtvDVarsOptions(SIZE(xvlistplus)), STAT=ierr)

         itens = iarg / 10
         iones = iarg - 10 * itens
         num = ndigs(itens) // ndigs(iones)
         DO j = 1, SIZE(xvlistplus)
            XtvDVarsOptions(j) = TRIM(xvlistplus(j)) // fluid // num
         END DO

         CALL XtvVarListMerge2A(xvlist,XtvDVarsOptions)

         DEALLOCATE(XtvDVarsOptions, STAT=ierr)

      END SUBROUTINE XtvVarListMerge4A


      SUBROUTINE XtvVarUpdMassB(graphlist)
!
!     Blabla
!
!     BEGIN MODULE USE
         USE GlobalDat,  ONLY: sik
         USE SoluteData, ONLY: iSolut, fracB10
!
         IMPLICIT NONE
!
!     Dummy Variable Declarations
         CHARACTER(LEN=7), INTENT(IN) :: graphlist
!
!     Local Variable Declarations

         ! Update for option 'minimal'
         IF (TRIM(graphlist) .EQ. 'minimal') THEN
```

```
            CALL XtvVarListMerge(XtvDVarsMin,XtvDVarsMassBMinDef)

         ! Update for option 'limited'
         ELSE IF (TRIM(graphlist) .EQ. 'limited') THEN

            CALL XtvVarListMerge(XtvDVarsLim,XtvDVarsMassBLimDef)

            IF (iSolut > 0_sik) THEN

               CALL XtvVarListMerge(XtvDVarsLim,XtvDVarsMassBLimIsolut)

               IF (fracB10 < 1.0_sdk) THEN

                  CALL XtvVarListMerge(XtvDVarsLim,XtvDVarsMassBLimIsolutFracB)

               END IF

            END IF

         ! Update for option 'full'
         ELSE

            CALL XtvVarListMerge(XtvDVarsFull,XtvDVarsMassBFullDef)

         END IF

      END SUBROUTINE XtvVarUpdMassB


      SUBROUTINE XtvVarUpd1D(idx,graphlist)
!
!     Creating dynamic variables lists for Generic 1D variables
!
!     BEGIN MODULE USE
         USE EosNCGData,       ONLY: ncGasSpecies
         USE GlobalDat,            ONLY: sik, nTraceG, nTraceL, disFields, nBubbles,
nDroplets, bubDev
         USE GlobalDat,        ONLY: numberOfNCGases
         USE SoluteData,       ONLY: iSolut, fracB10, solveB10
         !USE TraceSpeciesData, ONLY: traceGName, traceLName
!
         IMPLICIT NONE
!
!     Dummy Variable Declarations
         INTEGER(sik),     INTENT(IN) :: idx
         CHARACTER(LEN=7), INTENT(IN) :: graphlist
!
!     Local Variable Declarations
         INTEGER(sik)     :: i
         CHARACTER(LEN=4) :: number

         ! Update for option 'minimal'
         IF (TRIM(graphlist) .EQ. 'minimal') THEN

            CALL XtvVarListIni(XtvDVarsMin,XtvDVars1DMinDef)
            CALL XtvVarUpdMassB('minimal')

         ! Update for option 'limited'
         ELSE IF (TRIM(graphlist) .EQ. 'limited') THEN

            CALL XtvVarListIni(XtvDVarsLim,XtvDVars1DLimDef)
            CALL XtvVarUpdMassB('limited')
```

```
IF (iSolut > 0_sik) THEN

   CALL XtvVarListMerge(XtvDVarsLim,XtvDVars1DLimIsolut)

   IF (solveB10) THEN

      CALL XtvVarListMerge(XtvDVarsLim,XtvDVars1DLimIsolutsolveB10)

   END IF

END IF

IF (numberOfNCGases > 1_sik) THEN

   DO i = 1, numberOfNCGases

     CALL XtvVarListMerge(XtvDVarsLim,XtvDVars1DLimItrace,ncGasSpecies(i))

   END DO

ELSE

   IF (nTraceG /= 0) THEN

      DO i = 1, nTraceG

         CALL XtvVarListMerge(XtvDVarsLim,XtvDVars1DLimItrace,'Gas',i)

      END DO

   END IF

   IF (nTraceL /= 0) THEN

      DO i = 1, nTraceL

      CALL XtvVarListMerge(XtvDVarsLim,XtvDVars1DLimItrace,'Liq',i)

      END DO

   END IF

END IF

IF (bubDev) THEN

!  Note: bubDev is always .TRUE. in v5.0Patch5
   CALL XtvVarListMerge(XtvDVarsLim,XtvDVars1DLimBubdev)

END IF

IF (disFields .GT. 0) THEN

   DO i = 1, disFields

      IF (i .LE. nDroplets) THEN
         WRITE(number,'(i4)') i
         CALL XtvVarListMerge(XtvDVarsLim,XtvDVars1DLimDisfieldsD,TRIM(ADJUSTL(number)))
      ELSE
         WRITE(number,'(i4)') i-nDroplets
         CALL XtvVarListMerge(XtvDVarsLim,XtvDVars1DLimDisfieldsB,TRIM(ADJUSTL(number)))
      END IF

   END DO
```

```
            IF (nBubbles .EQ. 1_sik) THEN

               CALL XtvVarListMerge(XtvDVarsLim,XtvDVars1DLimDisfieldsnB1)

            ELSE IF (nBubbles .EQ. 2_sik) THEN

               CALL XtvVarListMerge(XtvDVarsLim,XtvDVars1DLimDisfieldsnB2)

            END IF

         END IF

      ! Update for option 'full'
      ELSE

         CALL XtvVarListIni(XtvDVarsFull,XtvDVars1DFullDef)
         CALL XtvVarUpdMassB('full   ')

         IF (iSolut > 0_sik) THEN

            CALL XtvVarListMerge(XtvDVarsFull,XtvDVars1DFullIsolut)

         END IF

      END IF

      RETURN

   END SUBROUTINE XtvVarUpd1D


   SUBROUTINE XtvVarUpd3D(idx,graphlist)
!
!     Creating dynamic variables lists for Vessel variables
!
!     BEGIN MODULE USE
         USE EosNCGData,       ONLY: ncGasSpecies
         USE GlobalDat,        ONLY: sik, imfr, nTraceG, nTraceL, disFields, nDroplets
         USE GlobalDat,        ONLY: numberOfNCGases
         USE SoluteData,       ONLY: iSolut, solveB10
         USE VessVlt
         !USE TraceSpeciesData, ONLY: traceGName, traceLName
!
         IMPLICIT NONE
!
!     Dummy Variable Declarations
         INTEGER(sik),     INTENT(IN) :: idx
         CHARACTER(LEN=7), INTENT(IN) :: graphlist
!
!     Local Variable Declarations
         INTEGER(sik)     :: i
         CHARACTER(LEN=4) :: number

         ! Update for option 'minimal'
         IF (TRIM(graphlist) .EQ. 'minimal') THEN

            CALL XtvVarListIni(XtvDVarsMin,XtvDVarsVsslMinDef)
            CALL XtvVarUpdMassB('minimal')

         ! Update for option 'limited'
         ELSE IF (TRIM(graphlist) .EQ. 'limited') THEN
```

```
CALL XtvVarListIni(XtvDVarsLim,XtvDVarsVsslLimDef)
CALL XtvVarUpdMassB('limited')

IF (imfr .EQ. 1) THEN

   CALL XtvVarListMerge(XtvDVarsLim,XtvDVarsVsslLimImfr1)

ELSE IF (imfr .EQ. 3) THEN

   CALL XtvVarListMerge(XtvDVarsLim,XtvDVarsVsslLimImfr3)

END IF

IF (iSolut .NE. 0) THEN

   CALL XtvVarListMerge(XtvDVarsLim,XtvDVarsVsslLimIsolut)

   IF (solveB10) THEN

      CALL XtvVarListMerge(XtvDVarsLim,XtvDVarsVsslLimIsolutsolveB10)

   END IF

END IF

IF (numberOfNCGases > 1_sik) THEN

   DO i = 1, numberOfNCGases

     CALL XtvVarListMerge(XtvDVarsLim,XtvDVarsVsslLimNtraceG,ncGasSpecies(i))

   END DO

ELSE

   IF (nTraceG > 0) THEN

      DO i = 1, nTraceG

         CALL XtvVarListMerge(XtvDVarsLim,XtvDVarsVsslLimNtraceG,'Gas',i)

      END DO

   END IF

   IF (nTraceL > 0) THEN

      DO i = 1, nTraceL

         CALL XtvVarListMerge(XtvDVarsLim,XtvDVarsVsslLimNtraceL,'Liq',i)

      END DO

   END IF

END IF

IF (disFields .GT. 0) THEN

   DO i = 1, disFields

      IF (i .LE. nDroplets) THEN
         WRITE(number,'(i4)') i
```

```
                CALL XtvVarListMerge(XtvDVarsLim,XtvDVarsVsslLimDisfieldsD,TRIM(ADJUSTL(number)))
                  ELSE
                     WRITE(number,'(i4)') i-nDroplets
                CALL XtvVarListMerge(XtvDVarsLim,XtvDVarsVsslLimDisfieldsB,TRIM(ADJUSTL(number)))
                  END IF

               END DO

            END IF

            IF (vessTab(idx)%idcr .NE. 0) THEN

               CALL XtvVarListMerge(XtvDVarsLim,XtvDVarsVsslLimIdcr)

            END IF

            IF (vessTab(idx)%ilcsp .NE. 0) THEN

               CALL XtvVarListMerge(XtvDVarsLim,XtvDVarsVsslLimIlcsp)

            END IF

            IF (vessTab(idx)%icrr .NE. 0) THEN

               CALL XtvVarListMerge(XtvDVarsLim,XtvDVarsVsslLimIcrr)

            END IF

            IF (vessTab(idx)%iucsp .NE. 0) THEN

               CALL XtvVarListMerge(XtvDVarsLim,XtvDVarsVsslLimIucsp)

            END IF

         ! Update for option 'full'
         ELSE

            CALL XtvVarListIni(XtvDVarsFull,XtvDVarsVsslFullDef)
            CALL XtvVarUpdMassB('full   ')

            IF (vessTab(idx)%nolt < 1) THEN

               CALL XtvVarListMerge(XtvDVarsFull,XtvDVarsVsslFullNolt0)

            END IF

            IF (iSolut .NE. 0) THEN

               CALL XtvVarListMerge(XtvDVarsFull,XtvDVarsVsslFullIsolut)

            END IF

         END IF

         RETURN

      END SUBROUTINE XtvVarUpd3D


!     Initiates update of list 'xvlist' using content of list 'xvlistDef'
      SUBROUTINE XtvVarListIni(xvlist,xvlistDef)
!
!     BEGIN MODULE USE
         USE GlobalDat, ONLY: sik
```

```
!
         IMPLICIT NONE
!
!     Dummy Variable Declarations
         CHARACTER(LEN=13), ALLOCATABLE, DIMENSION(:), INTENT(INOUT) :: xvlist
         CHARACTER(LEN=13),              DIMENSION(:), INTENT(IN)    :: xvlistDef
!
!     Local Variable Declarations
         INTEGER(sik) :: ierr

         ALLOCATE(xvlist(SIZE(xvlistDef)), STAT=ierr)
         xvlist = xvlistDef

         RETURN

      END SUBROUTINE XtvVarListIni


      SUBROUTINE XtvVarListsPrep(idx,doit)
!
!      Preparation or deallocation of all dynamic variables lists (minimal, limited and full)
!     of one component where argument 'idx' is the component index,
!     and argument 'doit' is a flag to select between preparing or deallocation
!
!     BEGIN MODULE USE
         USE CompTyp,           ONLY: chanh, teeh, jetph, turbh, heatrh, sepdh, pumph,
valveh, plenh, prizrh, pipeh
         USE CompTyp,            ONLY: breakh, fillh, vsslh, htstrh, powerh, flpowerh,
radench, contanh, canchanh
         USE ContanData,       ONLY: contanTab, contanDropOn
         USE EosNCGData,       ONLY: ncGasSpecies
         USE Flt,              ONLY: genTab
         USE FuelRodModelData, ONLY: frGasP
         USE GlobalDat,        ONLY: sik
         USE GlobalDat,        ONLY: ncontant, nTraceG, nTraceL
         USE GlobalDat,        ONLY: numberOfNCGases
         USE HSVlt,            ONLY: hsTab
         USE  PipeVlt,              ONLY:  pipeTab,  prizerOpt,  accumSharp,  accumNoGas,
accumSphere
         USE PowVlt,           ONLY: powTab
         USE SoluteData,       ONLY: iSolut, fracB10, solveB10
         USE ValveVlt,         ONLY: valveTab, vSwing
         !USE TraceSpeciesData, ONLY: traceGName, traceLName
!
         IMPLICIT NONE
!
!     Dummy Variable Declarations
         INTEGER(sik), INTENT(IN) :: idx, doit
!
!     Local Variable Declarations
         INTEGER(sik)                                  :: compId, ierr, i
         CHARACTER(LEN=13), ALLOCATABLE, DIMENSION(:) :: XtvDVarsOptions

         ierr = 0

!
!        Deallocate all dynamic variables lists and exits
         IF (doit == 0) THEN

            DEALLOCATE(XtvDVarsMin, STAT=ierr)
            DEALLOCATE(XtvDVarsLim, STAT=ierr)
            DEALLOCATE(XtvDVarsFull, STAT=ierr)
```

```
            RETURN

         END IF

!
!        Prepare variables lists for general problem variables
         IF (idx .LT. 1) THEN

!!!!!!!!!!!!!!!!!!!       Update for option 'minimal'       !!!!!!!!!!!!!!!!!!!

         !  Begin list update for General problem variables
            CALL XtvVarListIni(XtvDVarsMin,XtvDVarsGnPrMinDef)

            IF (iSolut > 0_sik) THEN

               CALL XtvVarListMerge(XtvDVarsMin,XtvDVarsGnPrMinIsolut)

               IF (fracB10 < 1.0_sdk) THEN

                  CALL XtvVarListMerge(XtvDVarsMin,XtvDVarsGnPrMinIsolutFracB)

               END IF

            END IF

!!!!!!!!!!!!!!!!!!!       Update for option 'limited'       !!!!!!!!!!!!!!!!!!!

         !  Begin list update for General problem variables
            CALL XtvVarListIni(XtvDVarsLim,XtvDVarsGnPrLimDef)

!!!!!!!!!!!!!!!!!!!        Update for option 'full'         !!!!!!!!!!!!!!!!!!!

         !  Begin list update for General problem variables
            CALL XtvVarListIni(XtvDVarsFull,XtvDVarsGnPrFullDef)

            IF (ncontant > 0_sik) THEN

               CALL XtvVarListMerge(XtvDVarsFull,XtvDVarsGnPrFullCont)

            END IF

!
!        Prepare variables lists for components
         ELSE

            compId = genTab(idx)%num

           IF ( (genTab(idx)%type .EQ. pipeh) .OR. (genTab(idx)%type .EQ. canchanh) ) THEN

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!     Begin update of component pipe      !!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

!!!!!!!!!!!!!!!!!!!       Update for option 'minimal'       !!!!!!!!!!!!!!!!!!!

            !  Begin list update for Generic 1D variables
               CALL XtvVarUpd1D(idx,'minimal')

            !  Begin list update for Pipe variables
               CALL XtvVarListMerge(XtvDVarsMin,XtvDVarsPipeMinDef)

!!!!!!!!!!!!!!!!!!!       Update for option 'limited'       !!!!!!!!!!!!!!!!!!!
```

```
            !  Begin list update for Generic 1D variables
               CALL XtvVarUpd1D(idx,'limited')

            !  Begin list update for Pipe variables
               CALL XtvVarListMerge(XtvDVarsLim,XtvDVarsPipeLimDef)

               SELECT CASE (pipeTab(idx)%pipeType)

                  CASE (accumSharp, accumNoGas, accumSphere, prizerOpt)
                     CALL XtvVarListMerge(XtvDVarsLim,XtvDVarsPipeLimPipetype)

               END SELECT

!!!!!!!!!!!!!!!!!!!        Update for option 'full'         !!!!!!!!!!!!!!!!!!!

            !  Begin list update for Generic 1D variables
               CALL XtvVarUpd1D(idx,'full   ')

            !  Begin list update for Pipe variables
               CALL XtvVarListMerge(XtvDVarsFull,XtvDVarsPipeFullDef)

               SELECT CASE (pipeTab(idx)%pipeType)

                  CASE (accumSharp, accumNoGas, accumSphere, prizerOpt)
                     CALL XtvVarListMerge(XtvDVarsFull,XtvDVarsPipeFullPipetype)

               END SELECT

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!      End update of component pipe       !!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

            ELSE IF (genTab(idx)%type .EQ. chanh) THEN

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!     Begin update of component chan      !!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

!!!!!!!!!!!!!!!!!!!       Update for option 'minimal'       !!!!!!!!!!!!!!!!!!!

            !  Begin list update for Generic 1D variables
               CALL XtvVarUpd1D(idx,'minimal')

            !  Begin list update for Pipe variables
               CALL XtvVarListMerge(XtvDVarsMin,XtvDVarsPipeMinDef)

            !  Begin list update for Chan variables
               CALL XtvVarListMerge(XtvDVarsMin,XtvDVarsChanMinDef)

!!!!!!!!!!!!!!!!!!!       Update for option 'limited'       !!!!!!!!!!!!!!!!!!!

            !  Begin list update for Generic 1D variables
               CALL XtvVarUpd1D(idx,'limited')

            !  Begin list update for Pipe variables
               CALL XtvVarListMerge(XtvDVarsLim,XtvDVarsPipeLimDef)

               SELECT CASE (pipeTab(idx)%pipeType)

                  CASE (accumSharp, accumNoGas, accumSphere, prizerOpt)
                     CALL XtvVarListMerge(XtvDVarsLim,XtvDVarsPipeLimPipetype)

               END SELECT
```

```
            !  Begin list update for Chan variables
               CALL XtvVarListMerge(XtvDVarsLim,XtvDVarsChanLimDef)

!!!!!!!!!!!!!!!!!!!        Update for option 'full'         !!!!!!!!!!!!!!!!!!!

            !  Begin list update for Generic 1D variables
               CALL XtvVarUpd1D(idx,'full   ')

            !  Begin list update for Pipe variables
               CALL XtvVarListMerge(XtvDVarsFull,XtvDVarsPipeFullDef)

               SELECT CASE (pipeTab(idx)%pipeType)

                  CASE (accumSharp, accumNoGas, accumSphere, prizerOpt)
                     CALL XtvVarListMerge(XtvDVarsFull,XtvDVarsPipeFullPipetype)

               END SELECT

            !  Begin list update for Chan variables
               CALL XtvVarListMerge(XtvDVarsFull,XtvDVarsChanFullDef)

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!      End update of component chan       !!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

         ELSE IF (genTab(idx)%type .EQ. teeh) THEN

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!      Begin update of component tee      !!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

!!!!!!!!!!!!!!!!!!!       Update for option 'minimal'       !!!!!!!!!!!!!!!!!!!

            !  Begin list update for Generic 1D variables
               CALL XtvVarUpd1D(idx,'minimal')

            !  Begin list update for Tee variables
               CALL XtvVarListMerge(XtvDVarsMin,XtvDVarsTeeMinDef)

!!!!!!!!!!!!!!!!!!!       Update for option 'limited'       !!!!!!!!!!!!!!!!!!!

            !  Begin list update for Generic 1D variables
               CALL XtvVarUpd1D(idx,'limited')

            !  Begin list update for Tee variables
               CALL XtvVarListMerge(XtvDVarsLim,XtvDVarsTeeLimDef)

!!!!!!!!!!!!!!!!!!!        Update for option 'full'         !!!!!!!!!!!!!!!!!!!

            !  Begin list update for Generic 1D variables
               CALL XtvVarUpd1D(idx,'full   ')

            !  Begin list update for Tee variables
               CALL XtvVarListMerge(XtvDVarsFull,XtvDVarsTeeFullDef)

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!       End update of component tee       !!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

         ELSE IF (genTab(idx)%type .EQ. jetph) THEN

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
```

```
!!!!!!!!!!!!!!!!!!!   Begin update of component jet pump    !!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

!!!!!!!!!!!!!!!!!!!       Update for option 'minimal'       !!!!!!!!!!!!!!!!!!!

            !  Begin list update for Generic 1D variables
               CALL XtvVarUpd1D(idx,'minimal')

            !  Begin list update for Tee variables
               CALL XtvVarListMerge(XtvDVarsMin,XtvDVarsTeeMinDef)

            !  Begin list update for Jet pump variables
               CALL XtvVarListMerge(XtvDVarsMin,XtvDVarsJetpMinDef)

!!!!!!!!!!!!!!!!!!!       Update for option 'limited'       !!!!!!!!!!!!!!!!!!!

            !  Begin list update for Generic 1D variables
               CALL XtvVarUpd1D(idx,'limited')

            !  Begin list update for Tee variables
               CALL XtvVarListMerge(XtvDVarsLim,XtvDVarsTeeLimDef)

            !  Begin list update for Jet pump variables
               CALL XtvVarListMerge(XtvDVarsLim,XtvDVarsJetpLimDef)

!!!!!!!!!!!!!!!!!!!        Update for option 'full'         !!!!!!!!!!!!!!!!!!!

            !  Begin list update for Generic 1D variables
               CALL XtvVarUpd1D(idx,'full   ')

            !  Begin list update for Tee variables
               CALL XtvVarListMerge(XtvDVarsFull,XtvDVarsTeeFullDef)

            !  Begin list update for Jet pump variables
               CALL XtvVarListMerge(XtvDVarsFull,XtvDVarsJetpFullDef)

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!    End update of component jet pump     !!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

            ELSE IF (genTab(idx)%type .EQ. turbh) THEN

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!    Begin update of component turbine    !!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

!!!!!!!!!!!!!!!!!!!       Update for option 'minimal'       !!!!!!!!!!!!!!!!!!!

            !  Begin list update for Generic 1D variables
               CALL XtvVarUpd1D(idx,'minimal')

            !  Begin list update for Tee variables
               CALL XtvVarListMerge(XtvDVarsMin,XtvDVarsTeeMinDef)

            !  Begin list update for Turbine variables
               CALL XtvVarListMerge(XtvDVarsMin,XtvDVarsTurbMinDef)

!!!!!!!!!!!!!!!!!!!       Update for option 'limited'       !!!!!!!!!!!!!!!!!!!

            !  Begin list update for Generic 1D variables
               CALL XtvVarUpd1D(idx,'limited')

            !  Begin list update for Tee variables
```

```
               CALL XtvVarListMerge(XtvDVarsLim,XtvDVarsTeeLimDef)

            !  Begin list update for Turbine variables
               CALL XtvVarListMerge(XtvDVarsLim,XtvDVarsTurbLimDef)

!!!!!!!!!!!!!!!!!!!        Update for option 'full'         !!!!!!!!!!!!!!!!!!!

            !  Begin list update for Generic 1D variables
               CALL XtvVarUpd1D(idx,'full   ')

            !  Begin list update for Tee variables
               CALL XtvVarListMerge(XtvDVarsFull,XtvDVarsTeeFullDef)

            !  Begin list update for Turbine variables
               CALL XtvVarListMerge(XtvDVarsFull,XtvDVarsTurbFullDef)

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!     End update of component turbine     !!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

            ELSE IF (genTab(idx)%type .EQ. heatrh) THEN

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!! Begin update of component feedwater heater  !!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

!!!!!!!!!!!!!!!!!!!       Update for option 'minimal'       !!!!!!!!!!!!!!!!!!!

            !  Begin list update for Generic 1D variables
               CALL XtvVarUpd1D(idx,'minimal')

            !  Begin list update for Tee variables
               CALL XtvVarListMerge(XtvDVarsMin,XtvDVarsTeeMinDef)

            !  Begin list update for Feedwater heater variables
               CALL XtvVarListMerge(XtvDVarsMin,XtvDVarsHeatrMinDef)

!!!!!!!!!!!!!!!!!!!       Update for option 'limited'       !!!!!!!!!!!!!!!!!!!

            !  Begin list update for Generic 1D variables
               CALL XtvVarUpd1D(idx,'limited')

            !  Begin list update for Tee variables
               CALL XtvVarListMerge(XtvDVarsLim,XtvDVarsTeeLimDef)

            !  Begin list update for Feedwater heater variables
               CALL XtvVarListMerge(XtvDVarsLim,XtvDVarsHeatrLimDef)

!!!!!!!!!!!!!!!!!!!        Update for option 'full'         !!!!!!!!!!!!!!!!!!!

            !  Begin list update for Generic 1D variables
               CALL XtvVarUpd1D(idx,'full   ')

            !  Begin list update for Tee variables
               CALL XtvVarListMerge(XtvDVarsFull,XtvDVarsTeeFullDef)

            !  Begin list update for Feedwater heater variables
               CALL XtvVarListMerge(XtvDVarsFull,XtvDVarsHeatrFullDef)

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!! End update of component feedwater heater  !!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
```

```
            ELSE IF (genTab(idx)%type .EQ. sepdh) THEN

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!   Begin update of component separator   !!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

!!!!!!!!!!!!!!!!!!!!       Update for option 'minimal'       !!!!!!!!!!!!!!!!!!!!

            !  Begin list update for Generic 1D variables
               CALL XtvVarUpd1D(idx,'minimal')

            !  Begin list update for Tee variables
               CALL XtvVarListMerge(XtvDVarsMin,XtvDVarsTeeMinDef)

            !  Begin list update for Separator variables
               CALL XtvVarListMerge(XtvDVarsMin,XtvDVarsSepdMinDef)

!!!!!!!!!!!!!!!!!!!!       Update for option 'limited'       !!!!!!!!!!!!!!!!!!!!

            !  Begin list update for Generic 1D variables
               CALL XtvVarUpd1D(idx,'limited')

            !  Begin list update for Tee variables
               CALL XtvVarListMerge(XtvDVarsLim,XtvDVarsTeeLimDef)

            !  Begin list update for Separator variables
               CALL XtvVarListMerge(XtvDVarsLim,XtvDVarsSepdLimDef)

!!!!!!!!!!!!!!!!!!!!        Update for option 'full'         !!!!!!!!!!!!!!!!!!!!

            !  Begin list update for Generic 1D variables
               CALL XtvVarUpd1D(idx,'full   ')

            !  Begin list update for Tee variables
               CALL XtvVarListMerge(XtvDVarsFull,XtvDVarsTeeFullDef)

            !  Begin list update for Separator variables
               CALL XtvVarListMerge(XtvDVarsFull,XtvDVarsSepdFullDef)

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!    End update of component separator    !!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

            ELSE IF (genTab(idx)%type .EQ. pumph) THEN

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!     Begin update of component pump      !!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

!!!!!!!!!!!!!!!!!!!!       Update for option 'minimal'       !!!!!!!!!!!!!!!!!!!!

            !  Begin list update for Generic 1D variables
               CALL XtvVarUpd1D(idx,'minimal')

            !  Begin list update for Pump variables
               CALL XtvVarListMerge(XtvDVarsMin,XtvDVarsPumpMinDef)

!!!!!!!!!!!!!!!!!!!!       Update for option 'limited'       !!!!!!!!!!!!!!!!!!!!

            !  Begin list update for Generic 1D variables
               CALL XtvVarUpd1D(idx,'limited')

            !  Begin list update for Pump variables
```

```
               CALL XtvVarListMerge(XtvDVarsLim,XtvDVarsPumpLimDef)

!!!!!!!!!!!!!!!!!!!        Update for option 'full'         !!!!!!!!!!!!!!!!!!!

            !  Begin list update for Generic 1D variables
               CALL XtvVarUpd1D(idx,'full   ')

            !  Begin list update for Pump variables
               CALL XtvVarListMerge(XtvDVarsFull,XtvDVarsPumpFullDef)

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!      End update of component pump       !!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

            ELSE IF (genTab(idx)%type .EQ. valveh) THEN

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!     Begin update of component valve     !!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

!!!!!!!!!!!!!!!!!!!        Update for option 'minimal'       !!!!!!!!!!!!!!!!!!!

            !  Begin list update for Generic 1D variables
               CALL XtvVarUpd1D(idx,'minimal')

            !  Begin list update for Valve variables
               CALL XtvVarListMerge(XtvDVarsMin,XtvDVarsValveMinDef)

!!!!!!!!!!!!!!!!!!!        Update for option 'limited'       !!!!!!!!!!!!!!!!!!!

            !  Begin list update for Generic 1D variables
               CALL XtvVarUpd1D(idx,'limited')

            !  Begin list update for Valve variables
               CALL XtvVarListMerge(XtvDVarsLim,XtvDVarsValveLimDef)

               IF (valveTab(idx)%ivty == vSwing) THEN

                  CALL XtvVarListMerge(XtvDVarsLim,XtvDVarsValveLimIvty)

               END IF

!!!!!!!!!!!!!!!!!!!        Update for option 'full'         !!!!!!!!!!!!!!!!!!!

            !  Begin list update for Generic 1D variables
               CALL XtvVarUpd1D(idx,'full   ')

            !  Begin list update for Valve variables
               CALL XtvVarListMerge(XtvDVarsFull,XtvDVarsValveFullDef)

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!      End update of component valve      !!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

            ELSE IF (genTab(idx)%type .EQ. plenh) THEN

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!     Begin update of component plenum    !!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

!!!!!!!!!!!!!!!!!!!        Update for option 'minimal'       !!!!!!!!!!!!!!!!!!!

            !  Begin list update for Plenum variables
```

```
               CALL XtvVarListIni(XtvDVarsMin,XtvDVarsPlenMinDef)
               CALL XtvVarUpdMassB('minimal')

!!!!!!!!!!!!!!!!!!!       Update for option 'limited'       !!!!!!!!!!!!!!!!!!!

            !  Begin list update for Plenum variables
               CALL XtvVarListIni(XtvDVarsLim,XtvDVarsPlenLimDef)
               CALL XtvVarUpdMassB('limited')

               IF (isolut > 0_sik) THEN

                  CALL XtvVarListMerge(XtvDVarsLim,XtvDVarsPlenLimIsolut)

                  IF (solveB10) THEN
                     CALL XtvVarListMerge(XtvDVarsLim,XtvDVarsPlenLimIsolutsolveB10)
                  END IF

               END IF

!!!!!!!!!!!!!!!!!!!        Update for option 'full'         !!!!!!!!!!!!!!!!!!!

            !  Begin list update for Plenum variables
               CALL XtvVarListIni(XtvDVarsFull,XtvDVarsPlenFullDef)
               CALL XtvVarUpdMassB('full   ')

               IF (isolut > 0_sik) THEN

                  CALL XtvVarListMerge(XtvDVarsFull,XtvDVarsPlenFullIsolut)

               END IF

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!      End update of component plenum     !!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

            ELSE IF (genTab(idx)%type .EQ. breakh) THEN

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!      Begin update of component break     !!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

!!!!!!!!!!!!!!!!!!!       Update for option 'minimal'       !!!!!!!!!!!!!!!!!!!

            !  Begin list update for Break variables
               CALL XtvVarListIni(XtvDVarsMin,XtvDVarsBreakMinDef)

!!!!!!!!!!!!!!!!!!!       Update for option 'limited'       !!!!!!!!!!!!!!!!!!!

            !  Begin list update for Break variables
               CALL XtvVarListIni(XtvDVarsLim,XtvDVarsBreakLimDef)

               IF (iSolut .NE. 0) THEN

                  CALL XtvVarListMerge(XtvDVarsLim,XtvDVarsBreakLimIsolut)

                  IF (solveB10) THEN

                     CALL XtvVarListMerge(XtvDVarsLim,XtvDVarsBreakLimIsolutsolveB10)

                  END IF

               END IF
```

```
                IF (numberOfNCGases > 1_sik) THEN

                   DO i = 1, numberOfNCGases

                    CALL XtvVarListMerge(XtvDVarsLim,XtvDVarsBreakLimItrace,ncGasSpecies(i))

                   END DO

                ELSE

                   IF (nTraceG > 0) THEN

                      DO i = 1, nTraceG

                        CALL XtvVarListMerge(XtvDVarsLim,XtvDVarsBreakLimItrace,'Gas',i)

                      END DO

                   END IF

                   IF (nTraceL > 0) THEN

                      DO i = 1, nTraceL

                        CALL XtvVarListMerge(XtvDVarsLim,XtvDVarsBreakLimItrace,'Liq',i)

                      END DO

                   END IF

                END IF

!!!!!!!!!!!!!!!!!!!        Update for option 'full'         !!!!!!!!!!!!!!!!!!!

             !  Begin list update for Break variables
                CALL XtvVarListIni(XtvDVarsFull,XtvDVarsBreakFullDef)

                IF (iSolut .NE. 0) THEN

                   CALL XtvVarListMerge(XtvDVarsFull,XtvDVarsBreakFullIsolut)

                END IF

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!      End update of component break      !!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

             ELSE IF (genTab(idx)%type .EQ. fillh) THEN

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!     Begin update of component fill      !!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

!!!!!!!!!!!!!!!!!!!       Update for option 'minimal'       !!!!!!!!!!!!!!!!!!!

             !  Begin list update for Fill variables
                CALL XtvVarListIni(XtvDVarsMin,XtvDVarsFillMinDef)

!!!!!!!!!!!!!!!!!!!       Update for option 'limited'       !!!!!!!!!!!!!!!!!!!

             !  Begin list update for Fill variables
                CALL XtvVarListIni(XtvDVarsLim,XtvDVarsFillLimDef)
```

```
                IF (iSolut .NE. 0) THEN

                   CALL XtvVarListMerge(XtvDVarsLim,XtvDVarsFillLimIsolut)

                   IF (solveB10) THEN

                      CALL XtvVarListMerge(XtvDVarsLim,XtvDVarsFillLimIsolutsolveB10)

                   END IF

                END IF

                IF (numberOfNCGases > 1_sik) THEN

                   DO i = 1, numberOfNCGases

                    CALL XtvVarListMerge(XtvDVarsLim,XtvDVarsFillLimItrace,ncGasSpecies(i))

                   END DO

                ELSE

                   IF (nTraceG > 0) THEN

                      DO i = 1, nTraceG

                       CALL XtvVarListMerge(XtvDVarsLim,XtvDVarsFillLimItrace,'Gas',i)

                      END DO

                   END IF

                   IF (nTraceL > 0) THEN

                      DO i = 1, nTraceL

                       CALL XtvVarListMerge(XtvDVarsLim,XtvDVarsFillLimItrace,'Liq',i)

                      END DO

                   END IF

                END IF

!!!!!!!!!!!!!!!!!!!        Update for option 'full'         !!!!!!!!!!!!!!!!!!!

             !  Begin list update for Fill variables
                CALL XtvVarListIni(XtvDVarsFull,XtvDVarsFillFullDef)

                IF (iSolut .NE. 0) THEN

                   CALL XtvVarListMerge(XtvDVarsFull,XtvDVarsFillFullIsolut)

                END IF

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!      End update of component fill       !!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

             ELSE IF (genTab(idx)%type .EQ. prizrh) THEN

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!    Begin update of component prizer     !!!!!!!!!!!!!!!!!!!
```

```
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

!!!!!!!!!!!!!!!!!!!!       Update for option 'minimal'      !!!!!!!!!!!!!!!!!!!!

            !  Begin list update for Generic 1D variables
               CALL XtvVarUpd1D(idx,'minimal')

            !  Begin list update for Prizer variables
               CALL XtvVarListMerge(XtvDVarsMin,XtvDVarsPrzrMinDef)

!!!!!!!!!!!!!!!!!!!!       Update for option 'limited'      !!!!!!!!!!!!!!!!!!!!

            !  Begin list update for Generic 1D variables
               CALL XtvVarUpd1D(idx,'limited')

            !  Begin list update for Prizer variables
               CALL XtvVarListMerge(XtvDVarsLim,XtvDVarsPrzrLimDef)

!!!!!!!!!!!!!!!!!!!!        Update for option 'full'        !!!!!!!!!!!!!!!!!!!!

            !  Begin list update for Generic 1D variables
               CALL XtvVarUpd1D(idx,'full   ')

            !  Begin list update for Prizer variables
               CALL XtvVarListMerge(XtvDVarsFull,XtvDVarsPrzrFullDef)

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!     End update of component prizer     !!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

            ELSE IF (genTab(idx)%type .EQ. vsslh) THEN

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!    Begin update of component vessel    !!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

!!!!!!!!!!!!!!!!!!!!    Update for all graphlevel options   !!!!!!!!!!!!!!!!!!!!

            !  Begin list update for Vessel variables
               CALL XtvVarUpd3D(idx,'minimal')
               CALL XtvVarUpd3D(idx,'limited')
               CALL XtvVarUpd3D(idx,'full   ')

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!     End update of component vessel     !!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

            ELSE IF (genTab(idx)%type .EQ. htstrh) THEN

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!     Begin update of component htstr    !!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

!!!!!!!!!!!!!!!!!!!!       Update for option 'minimal'      !!!!!!!!!!!!!!!!!!!!

            !  Begin list update for variables in 'HtStr'
               CALL XtvVarListIni(XtvDVarsMin,XtvDVarsHtStrMinDef)

            !  Begin list update for variables in 'HtStrC'
               CALL XtvVarListMerge(XtvDVarsMin,XtvDVarsHtStrCMinDef)

!!!!!!!!!!!!!!!!!!!!       Update for option 'limited'      !!!!!!!!!!!!!!!!!!!!
```

```
            !  Begin list update for variables in 'HtStr'
               CALL XtvVarListIni(XtvDVarsLim,XtvDVarsHtStrLimDef)

               IF (hsTab(idx)%nfci /= 0_sik) THEN

                  CALL XtvVarListMerge(XtvDVarsLim,XtvDVarsHtStrLimNfci)

                  IF (hsTab(idx)%nfci > 1_sik) THEN
                     CALL XtvVarListMerge(XtvDVarsLim,XtvDVarsHtStrLimNfci2)
                  END IF

               END IF

               IF (hsTab(idx)%nmwrx /= 0) THEN

                  CALL XtvVarListMerge(XtvDVarsLim,XtvDVarsHtStrLimNmwrx)

               END IF

            !  Begin list update for variables in 'HtStrC'
               CALL XtvVarListMerge(XtvDVarsLim,XtvDVarsHtStrCLimDef)

               IF (ABS(hsTab(idx)%nfci) > 0_sik) THEN

                  CALL XtvVarListMerge(XtvDVarsLim,XtvDVarsHtStrCLimNfci)

                  IF (fRGasP) THEN
                     CALL XtvVarListMerge(XtvDVarsLim,XtvDVarsHtStrCLimNfcifRGasP)
                  END IF

               END IF

               IF (hsTab(idx)%nmwrx /= 0) THEN

                  CALL XtvVarListMerge(XtvDVarsLim,XtvDVarsHtStrCLimNmwrx)

               END IF

!!!!!!!!!!!!!!!!!!!        Update for option 'full'         !!!!!!!!!!!!!!!!!!!

            !  Begin list update for variables in 'HtStr'
               CALL XtvVarListIni(XtvDVarsFull,XtvDVarsHtStrFullDef)

               IF (hsTab(idx)%nfci /= 0_sik) THEN

               !  Place-holder (no additional XTV variable in this version of TRACE)
                  ierr = 0

               END IF

               IF (hsTab(idx)%nmwrx /= 0) THEN

               !  Place-holder (no additional XTV variable in this version of TRACE)
                  ierr = 0

               END IF

               IF (hsTab(idx)%ngrids > 0) THEN

                  CALL XtvVarListMerge(XtvDVarsFull,XtvDVarsHtStrFullNgrids)

               END IF
```

```
            !  Begin list update for variables in 'HtStrC'
               CALL XtvVarListMerge(XtvDVarsFull,XtvDVarsHtStrCFullDef)

               IF (ABS(hsTab(idx)%nfci) > 0_sik) THEN

                  CALL XtvVarListMerge(XtvDVarsFull,XtvDVarsHtStrCFullNfci)

                  IF (fRGasP) THEN
                     CALL XtvVarListMerge(XtvDVarsFull,XtvDVarsHtStrCFullNfcifRGasP)
                  END IF

               END IF

               IF (hsTab(idx)%nmwrx /= 0) THEN

               !  Place-holder (no additional XTV variable in this version of TRACE)
                  ierr = 0

               END IF

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!      End update of component htstr      !!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

            ELSE IF (genTab(idx)%type .EQ. powerh) THEN

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!     Begin update of component power     !!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

!!!!!!!!!!!!!!!!!!!!       Update for option 'minimal'       !!!!!!!!!!!!!!!!!!!!

            !  Begin list update for variables in 'Power'
               CALL XtvVarListIni(XtvDVarsMin,XtvDVarsPowerMinDef)

!!!!!!!!!!!!!!!!!!!!       Update for option 'limited'       !!!!!!!!!!!!!!!!!!!!

            !  Begin list update for variables in 'Power'
               CALL XtvVarListIni(XtvDVarsLim,XtvDVarsPowerLimDef)

               IF (powTab(idx)%irpwty >= 11_sik) THEN

                  ALLOCATE(XtvDVarsOptions(SIZE(XtvDVarsPowerLimIrpwty)), STAT=ierr)
                  XtvDVarsOptions = XtvDVarsPowerLimIrpwty

                  ! Alternative variables for ifbtyp(1,2,3)=1 & ibu(4)<0

                  IF (powTab(idx)%ifbtyp(1) > 0_sik) THEN
                     XtvDVarsOptions(14) = XtvDVarsPowerLimIrpwtyAlt(1)
                  END IF

                  IF (powTab(idx)%ifbtyp(2) > 0_sik) THEN
                     XtvDVarsOptions(11) = XtvDVarsPowerLimIrpwtyAlt(2)
                  END IF

                  IF (powTab(idx)%ifbtyp(3) > 0_sik) THEN
                     XtvDVarsOptions(2) = XtvDVarsPowerLimIrpwtyAlt(3)
                  END IF

                  IF (powTab(idx)%ibu(4) < 0) THEN
                     XtvDVarsOptions(4) = XtvDVarsPowerLimIrpwtyAlt(4)
                  ELSE
                     IF (powTab(idx)%ibu(4) > 1) THEN
```

```
                        IF (iSolut == 1_sik) THEN
                           XtvDVarsOptions(4) = XtvDVarsPowerLimIrpwtyAltIbu(2)
                        ELSE
                           XtvDVarsOptions(4) = XtvDVarsPowerLimIrpwtyAltIbu(1)
                        END IF
                     END IF
                  END IF

                  CALL XtvVarListMerge(XtvDVarsLim,XtvDVarsOptions)
                  DEALLOCATE(XtvDVarsOptions, STAT=ierr)

               END IF

!!!!!!!!!!!!!!!!!!!        Update for option 'full'         !!!!!!!!!!!!!!!!!!!

            !  Begin list update for variables in 'Power'
               CALL XtvVarListIni(XtvDVarsFull,XtvDVarsPowerFullDef)

               IF (powTab(idx)%irpwty >= 11_sik) THEN

               !  Place-holder (no additional XTV variable in this version of TRACE)
                  ierr = 0

               END IF

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!      End update of component power      !!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

            ELSE IF (genTab(idx)%type .EQ. flpowerh) THEN

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!   Begin update of component flpowerh    !!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

!!!!!!!!!!!!!!!!!!!    Update for all graphlevel options    !!!!!!!!!!!!!!!!!!!!

            !  Begin list update for variables in 'Flpowerh'
               CALL XtvVarListIni(XtvDVarsMin,XtvDVarsFlPowerMinDef)
               CALL XtvVarListIni(XtvDVarsLim,XtvDVarsFlPowerLimDef)
               CALL XtvVarListIni(XtvDVarsFull,XtvDVarsFlPowerFullDef)

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!    End update of component flpowerh     !!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

            ELSE IF (genTab(idx)%type .EQ. radench) THEN

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!    Begin update of component radenc     !!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

!!!!!!!!!!!!!!!!!!!    Update for all graphlevel options    !!!!!!!!!!!!!!!!!!!!

            !  Begin list update for variables in 'Radenc'
               CALL XtvVarListIni(XtvDVarsMin,XtvDVarsRadEncMinDef)
               CALL XtvVarListIni(XtvDVarsLim,XtvDVarsRadEncLimDef)
               CALL XtvVarListIni(XtvDVarsFull,XtvDVarsRadEncFullDef)

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!     End update of component radenc      !!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
```

```
         ELSE IF (genTab(idx)%type .EQ. contanh) THEN

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!    Begin update of component contan     !!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

!!!!!!!!!!!!!!!!!!!!    Update for all graphlevel options    !!!!!!!!!!!!!!!!!!!!

         !  Begin list update for variables in 'Contan'
            CALL XtvVarListIni(XtvDVarsMin,XtvDVarsContMinDef)
            CALL XtvVarListIni(XtvDVarsLim,XtvDVarsContLimDef)
            CALL XtvVarListIni(XtvDVarsFull,XtvDVarsContFullDef)

            IF (contanTab%ncomt .NE. 0) THEN

               !CALL XtvVarListMerge(XtvDVarsMin,XtvDVarsContMinCompart)
               CALL XtvVarListMerge(XtvDVarsLim,XtvDVarsContLimCompart)
               CALL XtvVarListMerge(XtvDVarsFull,XtvDVarsContFullCompart)

            END IF

            IF (contanDropOn .AND. contanTab%nField > 0_sik) THEN

               !CALL XtvVarListMerge(XtvDVarsMin,XtvDVarsContMinDropOn)
               CALL XtvVarListMerge(XtvDVarsLim,XtvDVarsContLimDropOn)
               CALL XtvVarListMerge(XtvDVarsFull,XtvDVarsContFullDropOn)

            END IF

            IF (contanTab%ncool .NE. 0) THEN

               !CALL XtvVarListMerge(XtvDVarsMin,XtvDVarsContMinCooler)
               CALL XtvVarListMerge(XtvDVarsLim,XtvDVarsContLimCooler)
               CALL XtvVarListMerge(XtvDVarsFull,XtvDVarsContFullCooler)

            END IF

            IF (contanTab%nfanc .NE. 0) THEN

               !CALL XtvVarListMerge(XtvDVarsMin,XtvDVarsContMinFCool)
               CALL XtvVarListMerge(XtvDVarsLim,XtvDVarsContLimFCool)
               CALL XtvVarListMerge(XtvDVarsFull,XtvDVarsContFullFCool)

            END IF

            IF (contanTab%njct .NE. 0) THEN

               !CALL XtvVarListMerge(XtvDVarsMin,XtvDVarsContMinJct)
               CALL XtvVarListMerge(XtvDVarsLim,XtvDVarsContLimJct)
               CALL XtvVarListMerge(XtvDVarsFull,XtvDVarsContFullJct)

            END IF

            IF (contanTab%njctf .NE. 0) THEN

               !CALL XtvVarListMerge(XtvDVarsMin,XtvDVarsContMinJctf)
               CALL XtvVarListMerge(XtvDVarsLim,XtvDVarsContLimJctf)
               CALL XtvVarListMerge(XtvDVarsFull,XtvDVarsContFullJctf)

            END IF

            IF (contanTab%njcts .NE. 0) THEN
```

```
                  !CALL XtvVarListMerge(XtvDVarsMin,XtvDVarsContMinJcts)
                  CALL XtvVarListMerge(XtvDVarsLim,XtvDVarsContLimJcts)
                  CALL XtvVarListMerge(XtvDVarsFull,XtvDVarsContFullJcts)

               END IF

               IF (contanTab%nhs > 0) THEN

                  !CALL XtvVarListMerge(XtvDVarsMin,XtvDVarsContMinHS)
                  CALL XtvVarListMerge(XtvDVarsLim,XtvDVarsContLimHS)
                  CALL XtvVarListMerge(XtvDVarsFull,XtvDVarsContFullHS)

               END IF

               IF (contanTab%nhsp > 0_sik) THEN

                  !CALL XtvVarListMerge(XtvDVarsMin,XtvDVarsContMinTHS)
                  CALL XtvVarListMerge(XtvDVarsLim,XtvDVarsContLimTHS)
                  CALL XtvVarListMerge(XtvDVarsFull,XtvDVarsContFullTHS)

               END IF

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!     End update of component contan      !!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

            ELSE

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!! Begin genTab(idx)%type does not match any component type  !!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

            !  Could be improved with a cleaner error message and run abort
               ierr = 0

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!  End genTab(idx)%type does not match any component type   !!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

            END IF

         END IF

         RETURN

      END SUBROUTINE XtvVarListsPrep

END MODULE XtvCustom
```

APPENDIX B
SOURCE CODE MODIFICATIONS FOR V5.0P5

This appendix presents in subsections B.1 to B.7 the necessary modifications to the original TRACE v5.0p5 sources files [src/InfoOutM.f90], [src/NamlistDatM.f90], [src/NamlistInputM.f90], [src/NamlistM.f90], [src/XtvCompsM.f90], [src/XtvDumpM.f90] and [src/XtvSetupM.f90] for the implementations of the XTV-Customize option.

For one pair "$file.f90" and "$file.f90.mod" of the original and modified versions of the aforementioned source files, the patch file presented in each subsection has been obtained with the following Linux command:

```
diff -u $file.f90 $file.f90.mod > $file.patch
```

To create the modified source "$file.f90.mod" from the original source file "$file.f90", the patch file is to be applied using the following Linux command:

```
patch < $file.patch
```

These patches do not include the fixes for the few XTV variable extraction issues mentioned in subsection 3.1 . Note also that the structure of these modifications is very similar to that of other code versions v5.0p4 and v5.0p3.

B.1 Patch file for src/InfoOutM.f90

```
--- InfoOutM.f90    2017-09-29 11:22:48.000000002 +0200
+++ InfoOutM.f90.mod       2018-07-02 18:30:04.000000002 +0200
@@ -4,6 +4,8 @@
 !     This Software was developed for Purdue Subcontract Agreement 640-01812-2
 !     under NRC Prime Contract no. NRC-04-97-046 (Task #2)
 !
+!     Modified by Paul Scherrer Institut (07/18) - to include custom XTV output
capability.
+!
 !
       USE IntrType
       USE GlobalDat, ONLY: one, zero
@@ -1165,6 +1167,7 @@
           USE GlobalDat, ONLY: graphLevel
           USE MatrixDat, ONLY: blocks, iDmCL, iDMG
           USE SysConfig, ONLY: nVolTot, nEdges
+          USE XtvCustom, ONLY: graphFltFlag
 !
           IMPLICIT NONE
 !
@@ -1181,7 +1184,8 @@
           REAL(sdk) :: dxP, dxM, pStagM, pStagP
 !
 !         Check to see if these parameters are needed or not.
-          IF (graphLevel == 'full') THEN
+          !IF (graphLevel == 'full') THEN
+          IF ( (graphLevel == 'full') .OR. ( graphFltFlag == 1 ) ) THEN
 !
 !            Loop over volumes
             DO iVol = 1, nVolTot
```

B.2 Patch file for src/NamlistDatM.f90

```
--- NamlistDatM.f90 2017-09-29 11:22:48.000000002 +0200
+++ NamlistDatM.f90.mod    2018-07-02 18:29:59.000000002 +0200
@@ -4,6 +4,7 @@
 !     This Software was developed for Purdue Subcontract Agreement 640-01812-2
 !     under NCR Prime Contract no. NRC-04-97-046 (Task #2)
 !
+!     Modified by Paul Scherrer Institut (07/18) - to include custom XTV output
capability.
 !
       USE IntrType
       USE ErrorInterface, ONLY : error
@@ -73,7 +74,7 @@
          MODULE PROCEDURE InitNamelistString1D
       END INTERFACE
 !
-      INTEGER(sik), PARAMETER :: lenNamelist = 149_sik
+      INTEGER(sik), PARAMETER :: lenNamelist = 151_sik
       INTEGER(sik) :: cNmIdx = 0
 !     Note that this namelist list does not include any namelist options
 !     that are input only.  All options here are transferred to TPR
@@ -161,6 +162,7 @@
          USE SoluteData, ONLY: soluCW, soluCWDef, soluB, soluBDef, soluV, soluVDef,
fracB10, fracB10Def
          USE SpacerGridData, ONLY: gridTypes, gridTypesDef
          USE TimeStepDat, ONLY: dtstrt, dtStrtDef, dtstrtSet, icdelt, icDelTDef
+         USE XtvCustom, ONLY: graphCustId, graphCustIdDef, graphCustVar,
graphCustVarDef
 !
          IMPLICIT NONE
 !
@@ -460,6 +462,12 @@
          CALL InitNmlstVal('graphLevel  ', graphLevelDef, graphLevel, .TRUE., &
         & description='Flag for graphic level output')

+         CALL InitNmlstVal('graphCustId ', graphCustIdDef, graphCustId, .TRUE., &
+        & description='Component IDs for additional graphics output')
+
+         CALL InitNmlstVal('graphCustVar', graphCustVarDef, graphCustVar, .TRUE., &
+        & description='Additional variables for graphics output')
+
          CALL InitNmlstVal('chfMult     ', chfMultDef, chfMult, .TRUE., &
         & description='Multiplier for critical heat flux')

@@ -1015,6 +1023,7 @@
          USE EosNCGData, ONLY: ncGasSpecies
          USE GlobalDat, ONLY: graphLevel, repeatLoc, repeatLocSet
          USE Util, ONLY: SetBackupLoc
+         USE XtvCustom, ONLY: graphCustVar

 !
 !        This routine loads up namelist strings and 1D string arrays in the nmlstInfo
array.
@@ -1068,6 +1077,10 @@
                      IF (ncGasSpecies(1) /= '   ') THEN
                         nmlstInfo(idx)%s1dVal = ncGasSpecies
                      END IF
+                  CASE ('graphCustVar')
+                     IF (graphCustVar(1) /= '     ') THEN
+                        nmlstInfo(idx)%s1dVal = graphCustVar
+                     END IF
                   CASE DEFAULT
```

```
                      WRITE (message, '(2a)') ' *UpdateNamelistStrings* No coding
currently for string = ', &
                     & nmlstInfo(idx)%name
```

B.3 Patch file for src/NamlistInputM.f90

```
--- NamlistInputM.f90      2017-09-29 11:22:48.000000002 +0200
+++ NamlistInputM.f90.mod  2018-07-02 18:29:54.000000002 +0200
@@ -8,6 +8,8 @@
 !     Module NamlistInput Contains the sections of subroutine Input
 !     Specifically associated with input
 !
+!     Modified by Paul Scherrer Institut (07/18) - to include custom XTV output
capability.
+!
       USE IntrType
       USE ErrorInterface, ONLY : error

@@ -89,6 +91,7 @@
          USE SpacerGridData, ONLY: gridTypes
          USE TDMRVarDecl, ONLY: tdmrRamp
          USE TimeStepDat, ONLY: dtstrt, icdelt
+         USE XtvCustom, ONLY: graphCustId, graphCustVar
 !
          IMPLICIT NONE
          INTEGER(sik) :: ifs0
@@ -123,6 +126,7 @@
         & xtvRes, iamBWR, nofat, useSJC, nsolver, ncontant, useVessHS, naxn,
tracbOut, xtvAppend, nPower, creepAxial, &
        & trcdif, use_IAPWS_st, tdmrramp, nolt1d, nolt3d, iflcond, nflpower, fluids,
iTrace, radBurn, crudIn, &
        & numGenTbl, nEnclosure, R5dh, R5fdbk, PumpFricQ, ncGasSpecies, maxDxgn,
graphLevel, nrot, chfMult, &
+       & graphCustId, graphCustVar, &
        & disFields, lbdrag, viewFactReverse, iWriteMsgLim, fPowerIn, fPowerDe,
fReactIn, fReactDe, fPMax, detailedFRM, &
        & use_modNFI_k, matTypFuel, homMultAWD, blockageOn, multBFL, ecrpLimit,
fOxLayer, dOxLayer, matTypCladOx, &
        & iSolveDef, gridTypes, use_D2O_st, flowDepK, cSSMaxNumOpt, closest,
nAdjLosses, use_ncg_stens, iateModel, &
```

B.4 Patch file for src/NamlistM.f90

```
--- NamlistM.f90    2017-09-29 11:22:48.000000002 +0200
+++ NamlistM.f90.mod       2018-07-02 18:29:48.000000002 +0200
@@ -9,6 +9,8 @@
 !      defines the namelist list, and contains the routines that
 !      deal with the namelist.
 !
+!     Modified by Paul Scherrer Institut (07/18) - to include custom XTV output
capability.
+!

       CONTAINS

@@ -18,7 +20,8 @@
 !     namelist variables. This is targeted around the SNAP / TRAC
 !     interface so Defval for example is not supported.
 !
-      USE TprNamelist, ONLY: NameListT, WriteNamelistT, FreeNamelistT
+      USE TprNamelist
+      !USE TprNamelist, ONLY: NameListT, WriteNamelistT, FreeNamelistT
       USE NamlistDat, ONLY: nmlstInfo, lenNameList, nmInt, nmReal, nmLog, nmStr,
nmInt1D, nmReal1D, nmLog1D, nmStr1D
       USE TprBase, ONLY: TprUnknownInt, TprUnknownReal, tprFileIndex
 !
@@ -142,7 +145,8 @@
               CASE (nmInt1D)
                  nmLst%arTriplets(arCnt)%name = TRIM(nmlstInfo(i)%name)//CHAR(0)
                  nmLst%arTriplets(arCnt)%vType = i1dType
-                 nmLst%arTriplets(arCnt)%i1dValue => nmlstInfo(i)%i1dVal
+                 CALL
DEEPCOPY_INTARR(nmLst%arTriplets(arCnt)%i1dValue,nmlstInfo(i)%i1dVal)
+!                 nmLst%arTriplets(arCnt)%i1dValue => nmlstInfo(i)%i1dVal
                  NULLIFY(nmLst%arTriplets(arCnt)%d1dValue)
                  NULLIFY(nmLst%arTriplets(arCnt)%l1dValue)
                  NULLIFY(nmLst%arTriplets(arCnt)%s1dValue)
@@ -151,7 +155,8 @@
               CASE (nmReal1D)
                  nmLst%arTriplets(arCnt)%name = TRIM(nmlstInfo(i)%name)//CHAR(0)
                  nmLst%arTriplets(arCnt)%vType = d1dType
-                 nmLst%arTriplets(arCnt)%d1dValue => nmlstInfo(i)%r1dVal
+                 CALL
DEEPCOPY_DOUBLEARR(nmLst%arTriplets(arCnt)%d1dValue,nmlstInfo(i)%r1dVal)
+!                 nmLst%arTriplets(arCnt)%d1dValue => nmlstInfo(i)%r1dVal
                  NULLIFY(nmLst%arTriplets(arCnt)%i1dValue)
                  NULLIFY(nmLst%arTriplets(arCnt)%l1dValue)
                  NULLIFY(nmLst%arTriplets(arCnt)%s1dValue)
@@ -160,7 +165,8 @@
               CASE (nmLog1D)
                  nmLst%arTriplets(arCnt)%name = TRIM(nmlstInfo(i)%name)//CHAR(0)
                  nmLst%arTriplets(arCnt)%vType = l1dType
-                 nmLst%arTriplets(arCnt)%l1dValue => nmlstInfo(i)%l1dVal
+                 CALL
DEEPCOPY_LOGICALARR(nmLst%arTriplets(arCnt)%l1dValue,nmlstInfo(i)%l1dVal)
+!                 nmLst%arTriplets(arCnt)%l1dValue => nmlstInfo(i)%l1dVal
                  NULLIFY(nmLst%arTriplets(arCnt)%i1dValue)
                  NULLIFY(nmLst%arTriplets(arCnt)%d1dValue)
                  NULLIFY(nmLst%arTriplets(arCnt)%s1dValue)
@@ -291,6 +297,8 @@
 !
          DO  i=1,nmLst%nArOptionsOutput
             arOption = nmLst%arTriplets(i)
```

```
+!        Prevent reuse of namelist variables 'graphCustId' or 'graphCustVar' from the
restart file
+            IF
((arOption%name(1:11)=='graphCustId').OR.(arOption%name(1:12)=='graphCustVar')) CYCLE
 !        Select by name then by value
             nmInfoIdx = GetNmlstInfoIdx(arOption%name)
             IF ((nmlstInfo(nmInfoIdx)%setByUser .OR.
.NOT.nmlstInfo(nmInfoIdx)%changeOnRestart) .AND. &
@@ -684,6 +692,7 @@
          USE SoluteData, ONLY: soluCW, soluB, soluV, iSolut, solveB10, fracB10,
isolut, solveB10
          USE SpacerGridData, ONLY: gridTypes
          USE TimeStepDat, ONLY: dtstrt, icdelt
+         USE XtvCustom, ONLY: InitCustomGraphLevel
 !
          IMPLICIT NONE
 !
@@ -1566,6 +1575,7 @@
 !
 !
 !  check the flag that determines the graphic output level
+         CALL InitCustomGraphLevel(graphLevel)
          IF (graphLevel /= 'limited' .AND. graphLevel /= 'full' .AND. graphLevel /=
'minimal') THEN
             WRITE (detail(1), 554) graphLevel
 554         FORMAT ('namelist variable graphLevel =', a, ' should be "full", &
```

B.5 Patch file for src/XtvCompsM.f90

```
--- XtvCompsM.f90   2017-09-29 11:22:49.000000002 +0200
+++ XtvCompsM.f90.mod      2018-07-02 18:29:44.000000002 +0200
@@ -4,6 +4,8 @@
 !     This Software was developed for Purdue Subcontract Agreement 640-01812-2
 !     under NRC Prime Contract no. NRC-04-97-046 (Task #2)
 !
+!     Modified by Paul Scherrer Institut (07/18) - to include custom XTV output
capability.
+!
 !
       USE IntrType
       USE ErrorInterface, ONLY : error
@@ -37,6 +39,7 @@
          USE PrizeVlt, ONLY: prizerAdj, prizerHdAdj
          USE PowVlt, ONLY: powTab
          USE VessVlt, ONLY: nVessels
+         USE XtvCustom, ONLY: graphFltFlag, XtvVarListsPrep
          USE XtvData, ONLY: nXTVdumps, uTypeTime, nPipePrizer, prizerIndices,
currentPrizer, vessIndices
          USE XtvData, ONLY: xtvBufAr, lenXtvBufAr, AllocXtvCompList, AllocPlenAux
          USE XtvDump, ONLY: CreateXtvHeader, SetupXtvAppend
@@ -72,13 +75,15 @@
 !         1 for each containment substructure type (compartment, cooler, passive
junction, etc) (if Present)
 !

-         IF (graphLevel /= 'minimal') THEN
+         !IF (graphLevel /= 'minimal') THEN
+         IF ( (graphLevel /= 'minimal') .OR. ( graphFltFlag == 1 ) ) THEN
            locXtvCmps = ncompt-ncontant + 1
          ELSE
            locXtvCmps = 1
          END IF

-         IF (graphLevel /= 'minimal') THEN
+         !IF (graphLevel /= 'minimal') THEN
+         IF ( (graphLevel /= 'minimal') .OR. ( graphFltFlag == 1 ) ) THEN
            DO cco = bIndHS, eIndPW
              IF (genTab(cco)%type == htstrh) THEN
                locXtvCmps = locXtvCmps + hsTab(cco)%nrods
@@ -90,7 +95,8 @@
          IF (csGl%ntcb .GT. 0) locXtvCmps = locXtvCmps + 1
          IF (csGl%ntrp .GT. 0) locXtvCmps = locXtvCmps + 1

-         IF (ncontant .GT. 0 .AND. graphLevel /= 'minimal') THEN
+         !IF (ncontant .GT. 0 .AND. graphLevel /= 'minimal') THEN
+         IF ( ncontant .GT. 0 .AND. ( (graphLevel /= 'minimal') .OR. ( graphFltFlag
== 1 ) ) ) THEN
           IF (contanTab%ncomt .NE. 0) locXtvCmps = locXtvCmps + 1
           IF (contanTab%ncool .NE. 0) locXtvCmps = locXtvCmps + 1
           IF (contanTab%nfanc .NE. 0) locXtvCmps = locXtvCmps + 1
@@ -136,10 +142,14 @@
           NULLIFY(vessIndices)
         END IF
 !
-         IF (graphLevel /= 'minimal') THEN
+         !IF (graphLevel /= 'minimal') THEN
+         IF ( (graphLevel /= 'minimal') .OR. ( graphFltFlag == 1 ) ) THEN

           DO cco = 1, ncompt
 !
```

```
+!              Prepare Xtv variables lists for one component
+               CALL XtvVarListsPrep(cco,1)
+!
 !     branch on component type
 !
                IF (genTab(cco)%type .EQ. pipeh) THEN
@@ -186,12 +196,19 @@
                   CALL XtvContan(cco)
                END IF
 !
+!              Clean up Xtv variables lists
+               CALL XtvVarListsPrep(cco,0)
+!
             END DO
 !
          END IF
 !
+!        Prepare Xtv variables lists for general problem
+         CALL XtvVarListsPrep(0,1)
          CALL XtvGnPr
          CALL XtvCntl
+!        Clean up Xtv variables lists
+         CALL XtvVarListsPrep(0,0)
 !
 !     Alloc Xtv Buffer
          CALL TracAllo(xtvBufAr, lenXtvBufAr, 'xtvBufAr', 0.d0)
@@ -229,6 +246,7 @@
          USE MatrixDat, ONLY : iLiqVF
          USE SoluteData, ONLY: iSolut, fracB10, solveB10
          USE TraceSpeciesData, ONLY: traceGName, traceLName
+         USE XtvCustom, ONLY: graphFltFlag
          USE XtvData, ONLY: vDynamic, v1dCc, vWtr, v1dFa, cylRZ, vStatic
          USE XtvSetup, ONLY: CreateSummedR1, AddXtvTemplate, AddXtvJun, AddXtvLeg,
AddXtvVar, AddXtvComp
         USE XtvSetup, ONLY: CreateSArFrVal
@@ -311,10 +329,12 @@
          END IF
 !
 !       calc actual Num variables to be output
-         IF (graphLevel == 'limited') THEN
+         !IF (graphLevel == 'limited') THEN
+         IF ( (graphLevel == 'limited') .AND. ( graphFltFlag == 0 ) ) THEN
            nSVar = n1dSVarLim
            nDVar = n1dDVarLim
-         ELSE IF (graphLevel == 'full') THEN
+         !ELSE IF (graphLevel == 'full') THEN
+         ELSE IF ( (graphLevel == 'full') .OR. ( graphFltFlag == 1 ) ) THEN
            nSVar = n1dSVarFull
            nDVar = n1dDVarFull
          ELSE
@@ -335,7 +355,8 @@
            nDVar = nDVar - nIsolutVar
         ELSE
           nDVar = nDVar + nIsoluteVarFull
-          IF (graphLevel == 'full') THEN
+          !IF (graphLevel == 'full') THEN
+          IF ( (graphLevel == 'full') .OR. ( graphFltFlag == 1 ) ) THEN
             nDVar = nDVar + nSolutePPM
           END IF
           IF (fracB10 < 1.0_sdk) THEN
@@ -467,7 +488,8 @@
         CALL AddXtvVar(g1dAr(idx)%eln, vDynamic, v1dCc, 1, vWtr, 'el', 'liquid
internal energy')
```

```
          CALL AddXtvVar(g1dAr(idx)%evn, vDynamic, v1dCc, 1, vWtr, 'ev', 'gas internal
energy')
          CALL AddXtvVar(g1dAr(idx)%choked, vDynamic, v1dFa, 1, vWtr, 'choked',
'choking flag')
-         IF (graphLevel == 'full') THEN
+         !IF (graphLevel == 'full') THEN
+         IF ( (graphLevel == 'full') .OR. ( graphFltFlag == 1 ) ) THEN
             CALL AddXtvVar(g1dAr(idx)%gamn, vDynamic, v1dCc, 1, vWtr, 'gamn', 'mass
phase change rate')
             CALL AddXtvVar(g1dAr(idx)%hgam, vDynamic, v1dCc, 1, vWtr, 'hgam',
'subcooled boiling heat flux')
             CALL AddXtvVar(g1dAr(idx)%alvn, vDynamic, v1dCc, 1, vWtr, 'alvn',
'flashing interfacial HTC')
@@ -518,7 +540,8 @@
             CALL AddXtvVar(g1dAr(idx)%concSn, vDynamic, v1dCc, 1, vWtr, 'concS',
'solute mass fraction')
             CALL AddXtvVar(g1dAr(idx)%sn, vDynamic, v1dCc, 1, vWtr, 'sn', 'plateout
density')
             CALL AddXtvVar(g1dAr(idx)%sSolids, vDynamic, v1dCc, 1, vWtr, 'sSolids',
'plateout mass')
-            IF (graphLevel == 'full') THEN
+            !IF (graphLevel == 'full') THEN
+            IF ( (graphLevel == 'full') .OR. ( graphFltFlag == 1 ) ) THEN
               CALL AddXtvVar(g1dAr(idx)%cSppm, vDynamic, v1dCc, 1, vWtr, 'cSppm',
'solute ppm per unit solu.')
               CALL AddXtvVar(g1dAr(idx)%cWppm, vDynamic, v1dCc, 1, vWtr, 'cWppm',
'solute ppm per unit water')
               CALL AddXtvVar(g1dAr(idx)%b10Sppm, vDynamic, v1dCc, 1, vWtr,
'b10Sppm', 'B10 ppm per unit solu.')
@@ -659,6 +682,7 @@
          USE GlobalDat, ONLY: nTraceG, nTraceL, graphLevel, numberOfNCGases
          USE SoluteData, ONLY: iSolut, solveB10
          USE TraceSpeciesData, ONLY: traceGName, traceLName
+         USE XtvCustom, ONLY: graphFltFlag
          USE XtvData, ONLY: vStatic, v1dCc, vWtr, vDynamic, vScalar
          USE XtvSetup, ONLY: AddXtvTemplate, AddXtvJun, AddXtvVar, AddXtvComp
 !
@@ -686,10 +710,12 @@
          c8type = gettype(genTab(idx)%type)

 !     calc number of variables actually output
-         IF (graphLevel == 'limited') THEN
+         !IF (graphLevel == 'limited') THEN
+         IF ( (graphLevel == 'limited') .AND. ( graphFltFlag == 0 ) ) THEN
            nSVar = nBrSVarLim
            nDVar = nBrDVarLim
-         ELSE IF (graphLevel == 'full') THEN
+         !ELSE IF (graphLevel == 'full') THEN
+         ELSE IF ( (graphLevel == 'full') .OR. ( graphFltFlag == 1 ) ) THEN
            nSVar = nBrSVarFull
            nDVar = nBrDVarFull
          ELSE
@@ -704,10 +730,12 @@
            nDVar = nDVar - nIsolutVar
          ELSE IF (solveB10) THEN
            nDVar = nDVar + nMFrB10Var
-           IF (graphLevel == 'full') THEN
+           !IF (graphLevel == 'full') THEN
+           IF ( (graphLevel == 'full') .OR. ( graphFltFlag == 1 ) ) THEN
              nDVar = nDVar + nSolutePPM
            END IF
-         ELSE IF (graphLevel == 'full') THEN
+         !ELSE IF (graphLevel == 'full') THEN
+         ELSE IF ( (graphLevel == 'full') .OR. ( graphFltFlag == 1 ) ) THEN
```

```
            nDVar = nDVar + nSolutePPM
         END IF

@@ -745,7 +773,8 @@
            CALL AddXtvVar(g1dAr(idx)%concSn, vDynamic, v1dCc, 1, vWtr, 'concS',
'solute mass fraction')
            CALL AddXtvVar(genTab(idx)%massSoluteChange, vDynamic, vScalar, 0, vWtr,
'mSolC', 'Time int. solute mfr')
            CALL AddXtvVar(breakTab(idx)%massSoluteFlowRate, vDynamic, vScalar, 0,
vWtr, 'solutMFR', 'Solute mfr')
-           IF (graphLevel == 'full') THEN
+           !IF (graphLevel == 'full') THEN
+           IF ( (graphLevel == 'full') .OR. ( graphFltFlag == 1 ) ) THEN
               CALL AddXtvVar(g1dAr(idx)%cSppm, vDynamic, v1dCc, 1, vWtr, 'cSppm',
'solute ppm per unit solu.')
               CALL AddXtvVar(g1dAr(idx)%cWppm, vDynamic, v1dCc, 1, vWtr, 'cWppm',
'solute ppm per unit water')
               CALL AddXtvVar(g1dAr(idx)%b10Sppm, vDynamic, v1dCc, 1, vWtr,
'b10Sppm', 'B10 ppm per unit solu.')
@@ -833,6 +862,7 @@
         USE Flt, ONLY: genTab
         USE Gen1DArray, ONLY: g1dAr
         USE GlobalDat, ONLY: graphLevel
+        USE XtvCustom, ONLY: graphFltFlag
         USE XtvData, ONLY: vDynamic, vScalar, v1dCc, vHot, vWtr
         USE XtvSetup, ONLY: AddXtvVar, AddXtvTemplate, CreateSummedR1
 !
@@ -858,10 +888,12 @@
         c8type = gettype(genTab(idx)%type)
 !
 !       Calculate number variables for CHAN.
-        IF (graphLevel == 'limited') THEN
+        !IF (graphLevel == 'limited') THEN
+        IF ( (graphLevel == 'limited') .AND. ( graphFltFlag == 0 ) ) THEN
            nSVar = nSVarLim
            nDVar = nDVarLim
-        ELSE IF (graphLevel == 'full') THEN
+        !ELSE IF (graphLevel == 'full') THEN
+        ELSE IF ( (graphLevel == 'full') .OR. ( graphFltFlag == 1 ) ) THEN
            nSVar = nSVarFull
            nDVar = nDVarFull
         ELSE
@@ -884,7 +916,8 @@
         CALL AddXtvTemplate(genTab(idx)%num, 0_sik, 1, chanTab(idx)%ncrz, iFaces,
grav, faI)
 !
 !       Output CHAN specific variables.
-        IF (graphLevel == 'full') THEN
+        !IF (graphLevel == 'full') THEN
+        IF ( (graphLevel == 'full') .OR. ( graphFltFlag == 1 ) ) THEN
            CALL AddXtvVar(chanTab(idx)%MCPR, vDynamic, vScalar, 0, vHot, 'mcpr',
'minimum crit. power ratio')
            CALL AddXtvVar(chanTab(idx)%peakCladTemp, vDynamic, vScalar, 0, vHot,
'peakClad', 'peak cladding temp')
            CALL AddXtvVar(chanAr(idx)%XEquil, vDynamic, v1dCc, nTempl, vWtr,
'xequil', 'Equilibrium flow quality')
@@ -940,6 +973,7 @@
 !
         USE Flt, ONLY: genTab
         USE ContanData, ONLY: contanTab, compart, contanDropOn
+        USE XtvCustom, ONLY: graphFltFlag
         USE XtvData, ONLY: vDynamic, v1dCc, vWtr, cylRZ, v2dCc
         USE XtvSetup, ONLY: AddXtvComp, CreateArFrVal, CreateSArFrVal, AddXtvVar,
AddXtvTemplate
```

```
          USE GlobalDat, ONLY : graphLevel
@@ -959,10 +993,12 @@
          c8type = "Compart "
 !
 !     calc number of variables actually output
-         IF (graphLevel == 'limited') THEN
+         !IF (graphLevel == 'limited') THEN
+         IF ( (graphLevel == 'limited') .AND. ( graphFltFlag == 0 ) ) THEN
            nSVar = nCtCpSVarLim
            nDVar = nCtCpDVarLim
-         ELSE IF (graphLevel == 'full') THEN
+         !ELSE IF (graphLevel == 'full') THEN
+         ELSE IF ( (graphLevel == 'full') .OR. ( graphFltFlag == 1 ) ) THEN
            nSVar = nCtCpSVarFull
            nDVar = nCtCpDVarFull
          ELSE
@@ -1085,6 +1121,7 @@
          USE ContanData, ONLY: contanTab, cooler
          USE Flt, ONLY: genTab
          USE GlobalDat, ONLY : graphLevel
+         USE XtvCustom, ONLY: graphFltFlag
          USE XtvData, ONLY: vDynamic, v1dCc, vWtr
          USE XtvSetup, ONLY: AddXtvComp, CreateSArFrVal, CreateArFrVal,
AddXtvTemplate, AddXtvVar
 !
@@ -1100,10 +1137,12 @@
          c8type = "Cooler  "
 !
 !     calc number of variables actually output
-         IF (graphLevel == 'limited') THEN
+         !IF (graphLevel == 'limited') THEN
+         IF ( (graphLevel == 'limited') .AND. ( graphFltFlag == 0 ) ) THEN
            nSVar = nCtCoSVarLim
            nDVar = nCtCoDVarLim
-         ELSE IF (graphLevel == 'full') THEN
+         !ELSE IF (graphLevel == 'full') THEN
+         ELSE IF ( (graphLevel == 'full') .OR. ( graphFltFlag == 1 ) ) THEN
            nSVar = nCtCoSVarFull
            nDVar = nCtCoDVarFull
          ELSE
@@ -1142,6 +1181,7 @@
 !
          USE Flt, ONLY: genTab
          USE ContanData, ONLY: contanTab, fancooler
+         USE XtvCustom, ONLY: graphFltFlag
          USE XtvData, ONLY: vDynamic, v1dCc, vWtr
          USE XtvSetup, ONLY: AddXtvComp, CreateSArFrVal, CreateArFrVal,
AddXtvTemplate, AddXtvVar
          USE GlobalDat, ONLY : graphLevel
@@ -1158,10 +1198,12 @@
          c8type = "FanCoolr"
 !
 !     calc number of variables actually output
-         IF (graphLevel == 'limited') THEN
+         !IF (graphLevel == 'limited') THEN
+         IF ( (graphLevel == 'limited') .AND. ( graphFltFlag == 0 ) ) THEN
            nSVar = nCtFcSVarLim
            nDVar = nCtFcDVarLim
-         ELSE IF (graphLevel == 'full') THEN
+         !ELSE IF (graphLevel == 'full') THEN
+         ELSE IF ( (graphLevel == 'full') .OR. ( graphFltFlag == 1 ) ) THEN
            nSVar = nCtFcSVarFull
            nDVar = nCtFcDVarFull
          ELSE
```

```
@@ -1185,7 +1227,8 @@
          CALL AddXtvVar(fancooler%mfCnds, vDynamic, v1dCc, 1, vWtr, 'mfcnds',
'Condensate mass transfer')
          CALL AddXtvVar(fancooler%qerr,   vDynamic, v1dCc, 1, vWtr, 'qerr',  'heat
balance error fraction')

-         IF (graphLevel == 'full') THEN
+         !IF (graphLevel == 'full') THEN
+         IF ( (graphLevel == 'full') .OR. ( graphFltFlag == 1 ) ) THEN
            CALL AddXtvVar(fancooler%nfin,  vDynamic, v1dCc, 1, vWtr, 'nfin',  'Fin
efficiency')
            CALL AddXtvVar(fancooler%tifc,  vDynamic, v1dCc, 1, vWtr, 'tifc',
'Air/water interface Temp.')
            CALL AddXtvVar(fancooler%twalo, vDynamic, v1dCc, 1, vWtr, 'twalo', 'temp.
outer tube wall')
@@ -1210,6 +1253,7 @@
 !
          USE Flt, ONLY: genTab
          USE ContanData, ONLY: contanTab, passJunc
+         USE XtvCustom, ONLY: graphFltFlag
          USE XtvData, ONLY: vDynamic, v1dCc, vWtr
          USE XtvSetup, ONLY: AddXtvComp, CreateSArFrVal, CreateArFrVal,
AddXtvTemplate, AddXtvVar
          USE GlobalDat, ONLY : graphLevel
@@ -1228,10 +1272,12 @@
          c8type = "PassJunc"
 !
 !     calc number of variables actually output
-         IF (graphLevel == 'limited') THEN
+         !IF (graphLevel == 'limited') THEN
+         IF ( (graphLevel == 'limited') .AND. ( graphFltFlag == 0 ) ) THEN
            nSVar = nCtPJSVarLim
            nDVar = nCtPJDVarLim
-         ELSE IF (graphLevel == 'full') THEN
+         !ELSE IF (graphLevel == 'full') THEN
+         ELSE IF ( (graphLevel == 'full') .OR. ( graphFltFlag == 1 ) ) THEN
            nSVar = nCtPJSVarFull
            nDVar = nCtPJDVarFull
          ELSE
@@ -1254,7 +1300,8 @@
          CALL AddXtvVar(passJunc%rmdap, vDynamic, v1dCc, 1, vWtr, 'rmdap', 'Air Flow
Rate')
          CALL AddXtvVar(passJunc%rmdlp, vDynamic, v1dCc, 1, vWtr, 'rmdlp', 'Liquid
Flow Rate')
          CALL AddXtvVar(passJunc%rmdot, vDynamic, v1dCc, 1, vWtr, 'rmdot', 'Total
Mass Flow Rate')
-         IF (graphLevel == 'full') THEN
+         !IF (graphLevel == 'full') THEN
+         IF ( (graphLevel == 'full') .OR. ( graphFltFlag == 1 ) ) THEN
            CALL AddXtvVar(passJunc%fr, vDynamic, v1dCc, 1, vWtr, 'fr', 'Friction
Factor')
          END IF
 !
@@ -1269,6 +1316,7 @@
 !
          USE Flt, ONLY: genTab
          USE ContanData, ONLY: contanTab, forceJunc
+         USE XtvCustom, ONLY: graphFltFlag
          USE XtvData, ONLY: vDynamic, v1dCc, vWtr
          USE XtvSetup, ONLY: AddXtvComp, CreateSArFrVal, CreateArFrVal,
AddXtvTemplate, AddXtvVar
          USE GlobalDat, ONLY : graphLevel
@@ -1287,10 +1335,12 @@
          c8type = "ForsJunc"
```

```
 !
 !     calc number of variables actually output
-         IF (graphLevel == 'limited') THEN
+         !IF (graphLevel == 'limited') THEN
+         IF ( (graphLevel == 'limited') .AND. ( graphFltFlag == 0 ) ) THEN
             nSVar = nCtFJSVarLim
             nDVar = nCtFJDVarLim
-         ELSE IF (graphLevel == 'full') THEN
+         !ELSE IF (graphLevel == 'full') THEN
+         ELSE IF ( (graphLevel == 'full') .OR. ( graphFltFlag == 1 ) ) THEN
             nSVar = nCtFJSVarFull
             nDVar = nCtFJDVarFull
          ELSE
@@ -1326,6 +1376,7 @@
 !
          USE Flt, ONLY: genTab
          USE ContanData, ONLY: contanTab, sourceSink
+         USE XtvCustom, ONLY: graphFltFlag
          USE XtvData, ONLY: vDynamic, v1dCc, vWtr
          USE XtvSetup, ONLY: AddXtvComp, CreateSArFrVal, CreateArFrVal,
AddXtvTemplate, AddXtvVar
          USE GlobalDat, ONLY : graphLevel
@@ -1344,10 +1395,12 @@
          c8type = "Source  "
 !
 !     calc number of variables actually output
-         IF (graphLevel == 'limited') THEN
+         !IF (graphLevel == 'limited') THEN
+         IF ( (graphLevel == 'limited') .AND. ( graphFltFlag == 0 ) ) THEN
             nSVar = nCtSJSVarLim
             nDVar = nCtSJDVarLim
-         ELSE IF (graphLevel == 'full') THEN
+         !ELSE IF (graphLevel == 'full') THEN
+         ELSE IF ( (graphLevel == 'full') .OR. ( graphFltFlag == 1 ) ) THEN
             nSVar = nCtSJSVarFull
             nDVar = nCtSJDVarFull
          ELSE
@@ -1391,6 +1444,7 @@
          USE GlobalDat, ONLY: nTraceG, nTraceL, graphLevel, numberOfNCGases
          USE SoluteData, ONLY: iSolut, solveB10
          USE TraceSpeciesData, ONLY: traceGName, traceLName
+         USE XtvCustom, ONLY: graphFltFlag
          USE XtvData, ONLY: vDynamic, v1dCc, vWtr, vScalar, v1dFa, vStatic
          USE XtvSetup, ONLY: AddXtvComp, AddXtvTemplate, AddXtvJun, AddXtvVar
 !
@@ -1416,10 +1470,12 @@
          c8type = gettype(genTab(idx)%type)
 !
 !     calc number of variables actually output
-         IF (graphLevel == 'limited') THEN
+         !IF (graphLevel == 'limited') THEN
+         IF ( (graphLevel == 'limited') .AND. ( graphFltFlag == 0 ) ) THEN
             nSVar = nFiSVarLim
             nDVar = nFiDVarLim
-         ELSE IF (graphLevel == 'full') THEN
+         !ELSE IF (graphLevel == 'full') THEN
+         ELSE IF ( (graphLevel == 'full') .OR. ( graphFltFlag == 1 ) ) THEN
             nSVar = nFiSVarFull
             nDVar = nFiDVarFull
          ELSE
@@ -1434,10 +1490,12 @@
             nDVar = nDVar - nIsolutVar
          ELSE IF (solveB10) THEN
             nDVar = nDVar + nMFrB10Var
```

```
-            IF (graphLevel == 'full') THEN
+            !IF (graphLevel == 'full') THEN
+            IF ( (graphLevel == 'full') .OR. ( graphFltFlag == 1 ) ) THEN
               nDVar = nDVar + nSolutePPM
             END IF
-         ELSE IF (graphLevel == 'full') THEN
+         !ELSE IF (graphLevel == 'full') THEN
+         ELSE IF ( (graphLevel == 'full') .OR. ( graphFltFlag == 1 ) ) THEN
            nDVar = nDVar + nSolutePPM
          END IF
 !
@@ -1484,7 +1542,8 @@
            CALL AddXtvVar(g1dAr(idx)%concSn, vDynamic, v1dCc, 1, vWtr, 'concS',
'solute mass fraction')
            CALL AddXtvVar(genTab(idx)%massSoluteChange, vDynamic, vScalar, 0, vWtr,
'mSolC', 'Time int. solute flow rate')
            CALL AddXtvVar(fillTab(idx)%massSoluteFlowRate, vDynamic, vScalar, 0,
vWtr, 'solutMFR', 'Solute flow rate')
-            IF (graphLevel == 'full') THEN
+            !IF (graphLevel == 'full') THEN
+            IF ( (graphLevel == 'full') .OR. ( graphFltFlag == 1 ) ) THEN
               CALL AddXtvVar(g1dAr(idx)%cSppm, vDynamic, v1dCc, 1, vWtr, 'cSppm',
'solute ppm per unit solu.')
               CALL AddXtvVar(g1dAr(idx)%cWppm, vDynamic, v1dCc, 1, vWtr, 'cWppm',
'solute ppm per unit water')
               CALL AddXtvVar(g1dAr(idx)%b10Sppm, vDynamic, v1dCc, 1, vWtr,
'b10Sppm', 'B10 ppm per unit solu.')
@@ -1559,6 +1618,7 @@
         USE MassBalData, ONLY: integratedSoluteFill, finalB10Mass, adjustedB10Mass,
massB10Error, integratedB10Break
         USE MassBalData, ONLY: integratedB10Fill, finalSoluteSolids, finalB10Solids
         USE SoluteData, ONLY: iSolut, fracB10
+         USE XtvCustom, ONLY: graphFltFlag
         USE XtvData, ONLY: vDynamic, vScalar, vWtr, xtvCpuTime, xtvCpuDummy, tnstep
         USE XtvSetup, ONLY: AddXtvComp, AddXtvVar
 !
@@ -1595,13 +1655,16 @@
         END IF
 !
 !     calc number of variables actually output
-         IF (graphLevel == 'limited') THEN
+         !IF (graphLevel == 'limited') THEN
+         IF ( (graphLevel == 'limited') .AND. ( graphFltFlag == 0 ) ) THEN
            nSVar = nGPSVarLim
            nDVar = nGPDVarLim
-         ELSE IF (graphLevel == 'minimal') THEN
+         !ELSE IF (graphLevel == 'minimal') THEN
+         ELSE IF ( (graphLevel == 'minimal') .AND. ( graphFltFlag == 0 ) ) THEN
            nSVar = nGPSVarMin
            nDVar = nGPDVarMin
-         ELSE IF (graphLevel == 'full') THEN
+         !ELSE IF (graphLevel == 'full') THEN
+         ELSE IF ( (graphLevel == 'full') .OR. ( graphFltFlag == 1 ) ) THEN
            nSVar = nGPSVarFull
            nDVar = nGPDVarFull
         ELSE
@@ -1614,7 +1677,8 @@
           END IF
         END IF
         IF (ncontant > 0_sik) THEN
-          IF (graphLevel == 'full') THEN
+          !IF (graphLevel == 'full') THEN
+          IF ( (graphLevel == 'full') .OR. ( graphFltFlag == 1 ) ) THEN
             nDVar = nDVar + 16
```

```
            END IF
         END IF
@@ -1667,7 +1731,8 @@
                & 'Estimated system initial B10 mass')
            END IF
         END IF
-         IF (graphLevel == 'full') THEN
+         !IF (graphLevel == 'full') THEN
+         IF ( (graphLevel == 'full') .OR. ( graphFltFlag == 1 ) ) THEN
            CALL AddXtvVar(dPrMx, vDynamic, vScalar, 0, vWtr, 'dprmax', 'Max. Frac.
pressure change')
            CALL AddXtvVar(dTlMx, vDynamic, vScalar, 0, vWtr, 'dtlmax', 'Max. Frac.
liq temp change')
            CALL AddXtvVar(dTvMx, vDynamic, vScalar, 0, vWtr, 'dtvmax', 'Max. Frac.
gas temp change')
@@ -1718,6 +1783,7 @@
         USE PowerArray, ONLY: powAr
         USE SoluteData, ONLY: iSolut
         USE VessArray, ONLY: vsAr
+         USE XtvCustom, ONLY: graphFltFlag
         USE XtvData, ONLY: vDynamic, vScalar, vHot
         USE XtvSetup, ONLY: AddXtvComp, CreateArZerFrAr, CreateArFrVal,
AddXtvTemplate, AddXtvVar
 !
@@ -1781,10 +1847,12 @@
         END IF
 !
 !     calc number of variables actually output
-         IF (graphLevel == 'limited') THEN
+         !IF (graphLevel == 'limited') THEN
+         IF ( (graphLevel == 'limited') .AND. ( graphFltFlag == 0 ) ) THEN
            nSVar = nPowSVarLim
            nDVar = nPowDVarLim
-         ELSE IF (graphLevel == 'full') THEN
+         !ELSE IF (graphLevel == 'full') THEN
+         ELSE IF ( (graphLevel == 'full') .OR. ( graphFltFlag == 1 ) ) THEN
            nSVar = nPowSVarFull
            nDVar = nPowDVarFull
         ELSE
@@ -1929,6 +1997,7 @@
         USE Flt, ONLY: genTab
         USE GlobalDat, ONLY : graphLevel
         USE RadEncData, ONLY: radTab, radAr
+         USE XtvCustom, ONLY: graphFltFlag
         USE XtvData, ONLY: vDynamic, v2dCc, vHot, cylRZ
         USE XtvSetup, ONLY: AddXtvComp, CreateArFrVal, CreateSArFrVal,
AddXtvTemplate, AddXtvVar
 !
@@ -1953,10 +2022,12 @@
         idxP = 0
 !
 !     calc number of variables actually output
-         IF (graphLevel == 'limited') THEN
+         !IF (graphLevel == 'limited') THEN
+         IF ( (graphLevel == 'limited') .AND. ( graphFltFlag == 0 ) ) THEN
           nSVar = nRadEncSVarLim
           nDVar = nRadEncDVarLim
-         ELSE IF (graphLevel == 'full') THEN
+         !ELSE IF (graphLevel == 'full') THEN
+         ELSE IF ( (graphLevel == 'full') .OR. ( graphFltFlag == 1 ) ) THEN
           nSVar = nRadEncSVarFull
           nDVar = nRadEncDVarFull
         ELSE
@@ -1994,6 +2065,7 @@
```

```
          USE FlPowVlt, ONLY: flpowTab
          USE Flt, ONLY: genTab
          USE GlobalDat, ONLY : graphLevel
+         USE XtvCustom, ONLY: graphFltFlag
          USE XtvData, ONLY: vDynamic, vScalar, vHot
          USE XtvSetup, ONLY: AddXtvComp, AddXtvVar
 !
@@ -2015,10 +2087,12 @@
          numChildren = 0
 !
 !     calc number of variables actually output
-         IF (graphLevel == 'limited') THEN
+         !IF (graphLevel == 'limited') THEN
+         IF ( (graphLevel == 'limited') .AND. ( graphFltFlag == 0 ) ) THEN
            nSVar = nFlPowSVarLim
            nDVar = nFlPowDVarLim
-         ELSE IF (graphLevel == 'full') THEN
+         !ELSE IF (graphLevel == 'full') THEN
+         ELSE IF ( (graphLevel == 'full') .OR. ( graphFltFlag == 1 ) ) THEN
            nSVar = nFlPowSVarFull
            nDVar = nFlPowDVarFull
          ELSE
@@ -2046,6 +2120,7 @@
          USE HSArray, ONLY: hsAr
          USE HSEnum, ONLY: geomSlab
          USE HSVlt, ONLY: hsTab
+         USE XtvCustom, ONLY: graphFltFlag, XtvVarFilterOut
          USE XtvData, ONLY: vDynamic, v1dCc, vHot, vScalar, cylRZ, cart2D, v1dFa,
v2dFaJ, dynNeg, dgnJ
          USE XtvSetup, ONLY: AddXtvComp, CreateArFrVal, CreateArZerFrAr,
AddXtvTemplate, AddXtvVar
          USE XtvSetup, ONLY: AddXtvDynAx
@@ -2064,6 +2139,7 @@
 !
          INTEGER(sik) :: nDVar, nSVar, cSys2D, idxP, ncrzp1, ncrz
          INTEGER(sik) :: compId, compSsId, numChildren, nTemplates
+         INTEGER(sik) :: FltFlag
          REAL(sdk), POINTER, DIMENSION(:) :: iFaces=>NULL(), grav=>NULL(), fa=>NULL()
          REAL(sdk), POINTER, DIMENSION(:) :: iFaces2=>NULL(), grav2=>NULL(),
fa2=>NULL()
          REAL(sdk), POINTER, DIMENSION(:) :: iFaces3=>NULL(), jFaces3=>NULL()
@@ -2078,11 +2154,13 @@
          idxP = 0
 !
 !     calc number of variables actually output
-         IF (graphLevel == 'limited') THEN
+         !IF (graphLevel == 'limited') THEN
+         IF ( (graphLevel == 'limited') .AND. ( graphFltFlag == 0 ) ) THEN
            nSVar = nGhSVarLim
            nDVar = nGhDVarLim
            nTemplates = 2
-         ELSE IF (graphLevel == 'full') THEN
+         !ELSE IF (graphLevel == 'full') THEN
+         ELSE IF ( (graphLevel == 'full') .OR. ( graphFltFlag == 1 ) ) THEN
            nSVar = nGhSVarFull
            nDVar = nGhDVarFull
 !
@@ -2091,6 +2169,11 @@
              nTemplates = 2
            ELSE
              nTemplates = 3
+             CALL XtvVarFilterOut(compId, 'gridtemp', graphLevel, FltFlag)
+             IF (FltFlag == 1_sik) THEN
+               nDvar = nDVar - 1
```

```
+                 nTemplates = 2
+              END IF
            END IF
         ELSE
            CALL error(5, 'Attempting to process an unknown graphLevel string', idx,
calledBy = 'XtvHtStr')
@@ -2186,7 +2269,8 @@
                      'cladding rupture temperature')
            END IF
         END IF
-         IF (graphLevel == 'full') THEN
+         !IF (graphLevel == 'full') THEN
+         IF ( (graphLevel == 'full') .OR. ( graphFltFlag == 1 ) ) THEN
            CALL AddXtvVar(hsAr(idx)%rdzN, vDynamic, v1dCc, 1, vHot, 'rdzN', 'generic
node row height')
            CALL AddXtvVar(hsAr(idx)%qWallLiq(:,1), vDynamic, v1dCc, 1, vHot,
'powli', 'inner surf HT to liquid')
            CALL AddXtvVar(hsAr(idx)%qWallVap(:,1), vDynamic, v1dCc, 1, vHot,
'powvi', 'inner surf HT to gas')
@@ -2201,12 +2285,15 @@
            CALL AddXtvVar(hsAr(idx)%enhnfac, vDynamic, v1dCc, 1, vHot, 'enhnfac',
'HT enhancement factor')
 ! Spacer grid surface temperatures
            IF (hsTab(idx)%ngrids > 0) THEN
-               NULLIFY(iFaces4, grav4, fa4)
-               CALL CreateArZerFrAr(iFaces4, hsAr(idx)%rdzNperm,
hsTab(idx)%ngrids+1)
-               CALL CreateArFrVal(grav4, 1.0d0, hsTab(idx)%ngrids+1)
-               CALL CreateArFrVal(fa4, 1.0d0, hsTab(idx)%ngrids+1)
-               CALL AddXtvTemplate(compId, compSsId, 1, hsTab(idx)%ngrids, iFaces4,
grav4, fa4)
-               CALL AddXtvVar(hsAr(idx)%gridtemp(:), vDynamic, v1dCc,3, vHot,
'gridtemp', 'Spacer Grid Surf Temp')
+              CALL XtvVarFilterOut(compId, 'gridtemp', graphLevel, FltFlag)
+              IF (FltFlag == 0_sik) THEN
+                 NULLIFY(iFaces4, grav4, fa4)
+                 CALL CreateArZerFrAr(iFaces4, hsAr(idx)%rdzNperm,
hsTab(idx)%ngrids+1)
+                 CALL CreateArFrVal(grav4, 1.0d0, hsTab(idx)%ngrids+1)
+                 CALL CreateArFrVal(fa4, 1.0d0, hsTab(idx)%ngrids+1)
+                 CALL AddXtvTemplate(compId, compSsId, 1, hsTab(idx)%ngrids,
iFaces4, grav4, fa4)
+                 CALL AddXtvVar(hsAr(idx)%gridtemp(:), vDynamic, v1dCc,3, vHot,
'gridtemp', 'Spacer Grid Surf Temp')
+              END IF
            END IF
         END IF
 !
@@ -2222,10 +2309,12 @@
         END IF
 !
 !     calc number of variables actually output
-         IF (graphLevel == 'limited') THEN
+         !IF (graphLevel == 'limited') THEN
+         IF ( (graphLevel == 'limited') .AND. ( graphFltFlag == 0 ) ) THEN
            nSVar = nHsSVarLim
            nDVar = nHsDVarLim
-         ELSE IF (graphLevel == 'full') THEN
+         !ELSE IF (graphLevel == 'full') THEN
+         ELSE IF ( (graphLevel == 'full') .OR. ( graphFltFlag == 1 ) ) THEN
            nSVar = nHsSVarFull
            nDVar = nHsDVarFull
         ELSE
@@ -2236,7 +2325,8 @@
```

```
             nDVar = nDVar + 2_sik
             IF (fRGasP) THEN
                nDVar = nDVar + 2_sik
-               IF (graphLevel =='full') THEN
+               !IF (graphLevel =='full') THEN
+               IF ( (graphLevel == 'full') .OR. ( graphFltFlag == 1 ) ) THEN
                   nDVar = nDVar + 8_sik
                END IF
             END IF
@@ -2274,7 +2364,8 @@
             IF (fRGasP) THEN
                CALL AddXtvVar(hsTab(idx)%volGas, vDynamic, vScalar, 0, vHot,
'volGas', 'fuel rod gas volume')
                CALL AddXtvVar(hsTab(idx)%tempGas, vDynamic, vScalar, 0, vHot,
'tempGas', 'fuel rod gas temperature')
-               IF (graphLevel =='full') THEN
+               !IF (graphLevel =='full') THEN
+               IF ( (graphLevel == 'full') .OR. ( graphFltFlag == 1 ) ) THEN
                   CALL AddXtvVar(hsTab(idx)%vFGap   , vDynamic, vScalar, 0, vHot,
'vFGap'  , &
                    & 'Gap gas volume fraction')
                   CALL AddXtvVar(hsTab(idx)%vFFRPlen, vDynamic, vScalar, 0, vHot,
'vFFRPLen', &
@@ -2301,7 +2392,8 @@
             CALL AddXtvVar(hsAr(idx)%depthZrRxIn(:), vDynamic, v1dFa, 1, vHot,
'depthZRI', &
              & 'inner surf depth of Zr reacted')
          END IF
-         IF (graphLevel =='full') THEN
+         !IF (graphLevel =='full') THEN
+         IF ( (graphLevel == 'full') .OR. ( graphFltFlag == 1 ) ) THEN
             CALL AddXtvVar(hsAr(idx)%zht(1:hsTab(idx)%nzmax), vDynamic, v1dFa, 1,
vHot, 'zht', 'node row elevation')
             CALL AddXtvVar(hsAr(idx)%hrfl(:, 1), vDynamic, v1dFa, 1, vHot, 'hrfli',
'inner surf liquid convective HTC')
             CALL AddXtvVar(hsAr(idx)%hrfv(:, 1), vDynamic, v1dFa, 1, vHot, 'hrfvi',
'inner surf gas HTC')
@@ -2423,6 +2515,7 @@
          USE GlobalDat, ONLY : graphLevel
          USE HeatrVlt,ONLY: heatrTab
          USE TeeVlt, ONLY: teeTab
+         USE XtvCustom, ONLY: graphFltFlag
          USE XtvData, ONLY: vDynamic, vScalar, vWtr
          USE XtvSetup, ONLY: AddXtvVar
 !
@@ -2440,10 +2533,12 @@

 !
 !     calc number of variables actually output
-         IF (graphLevel == 'limited') THEN
+         !IF (graphLevel == 'limited') THEN
+         IF ( (graphLevel == 'limited') .AND. ( graphFltFlag == 0 ) ) THEN
             nSVar = nHtSVarLim + nTeSVarLim
             nDVar = nHtDVarLim + nTeDVarLim
-         ELSE IF (graphLevel == 'full') THEN
+         !ELSE IF (graphLevel == 'full') THEN
+         ELSE IF ( (graphLevel == 'full') .OR. ( graphFltFlag == 1 ) ) THEN
             nSVar = nHtSVarFull + nTeSVarFull
             nDVar = nHtDVarFull + nTeDVarFull
          ELSE
@@ -2470,6 +2565,7 @@
          USE GlobalDat, ONLY : graphLevel
          USE JetpVlt, ONLY: jetpTab
          USE TeeVlt, ONLY: teeTab
```

```
+         USE XtvCustom, ONLY: graphFltFlag
          USE XtvData, ONLY: vDynamic, vScalar, vWtr
          USE XtvSetup, ONLY: AddXtvVar
 !
@@ -2487,10 +2583,12 @@

 !
 !     calc number of variables actually output
-         IF (graphLevel == 'limited') THEN
+         !IF (graphLevel == 'limited') THEN
+         IF ( (graphLevel == 'limited') .AND. ( graphFltFlag == 0 ) ) THEN
            nSVar = nJpSVarLim + nTeSVarLim
            nDVar = nJpDVarLim + nTeDVarLim
-         ELSE IF (graphLevel == 'full') THEN
+         !ELSE IF (graphLevel == 'full') THEN
+         ELSE IF ( (graphLevel == 'full') .OR. ( graphFltFlag == 1 ) ) THEN
            nSVar = nJpSVarFull + nTeSVarFull
            nDVar = nJpDVarFull + nTeDVarFull
          ELSE
@@ -2506,7 +2604,8 @@
          CALL AddXtvVar(jetpTab(idx)%mr, vDynamic, vScalar, 0, vWtr, 'mr', 'jet pump
flow ratio')
          CALL AddXtvVar(jetpTab(idx)%nrapp, vDynamic, vScalar, 0, vWtr, 'nrapp', 'jet
pump application head r&
         &atio')
-         IF (graphLevel == 'full') THEN
+         !IF (graphLevel == 'full') THEN
+         IF ( (graphLevel == 'full') .OR. ( graphFltFlag == 1 ) ) THEN
            CALL AddXtvVar(jetpTab(idx)%nreff, vDynamic, vScalar, 0, vWtr, 'nreff',
'jet pump effective head rat&
             &io')
            CALL AddXtvVar(jetpTab(idx)%etapp, vDynamic, vScalar, 0, vWtr, 'etapp',
'jet pump application effici&
@@ -2528,6 +2627,7 @@
          USE GlobalDat, ONLY : graphLevel, stdyst
          USE PipeVlt, ONLY: pipeTab, prizerOpt, accumSharp, accumNoGas, accumSphere
          USE PrizeVlt, ONLY: prizerAdj, prizerHdAdj
+         USE XtvCustom, ONLY: graphFltFlag
          USE XtvData, ONLY: currentPrizer, prizerIndices, vDynamic, vScalar, vHot
          USE XtvSetup, ONLY: AddXtvVar
 !
@@ -2551,11 +2651,13 @@
          REAL(sdk) :: rdummy = 0.0_sdk
 !
 !     calc number of variables actually output
-         IF (graphLevel == 'limited') THEN
+         !IF (graphLevel == 'limited') THEN
+         IF ( (graphLevel == 'limited') .AND. ( graphFltFlag == 0 ) ) THEN
            nSVar = nPiSVarLim
            nDVar = nPiDVarLim
            nAccVar = nAccVarLim
-         ELSE IF (graphLevel == 'full') THEN
+         !ELSE IF (graphLevel == 'full') THEN
+         ELSE IF ( (graphLevel == 'full') .OR. ( graphFltFlag == 1 ) ) THEN
            nSVar = nPiSVarFull
            nDVar = nPiDVarFull
            nAccVar = nAccVarFull
@@ -2625,7 +2727,8 @@
          CASE (accumSharp, accumNoGas, accumSphere, prizerOpt)
            CALL AddXtvVar(pipeTab(idx)%z, vDynamic, vScalar, 0, vHot, 'z', 'water
level')
            CALL AddXtvVar(pipeTab(idx)%qout, vDynamic, vScalar, 0, vHot, 'qout',
'liquid volume discharged')
-           IF (graphLevel == 'full') THEN
```

```
+            !IF (graphLevel == 'full') THEN
+            IF ( (graphLevel == 'full') .OR. ( graphFltFlag == 1 ) ) THEN
               CALL AddXtvVar(pipeTab(idx)%vflow, vDynamic, vScalar, 0, vHot,
'vflow', 'volumetric flow at exit &
                &face')
             END IF
@@ -2651,6 +2754,7 @@
          USE PlenArray, ONLY: plenAr
          USE PlenVlt, ONLY: plenTab, currentPlenumInd
          USE SoluteData, ONLY: iSolut, fracB10, solveB10
+         USE XtvCustom, ONLY: graphFltFlag
          USE XtvData, ONLY: vStatic, v1dCc, vWtr, vDynamic, plenAux
          USE XtvSetup, ONLY: AddXtvComp, AddXtvTemplate, AddXtvJun, AddXtvVar
 !
@@ -2682,10 +2786,12 @@
          c8type = gettype(genTab(idx)%type)
 !
 !     calc number of variables actually output
-         IF (graphLevel == 'limited') THEN
+         !IF (graphLevel == 'limited') THEN
+         IF ( (graphLevel == 'limited') .AND. ( graphFltFlag == 0 ) ) THEN
            nSVar = nPlSVarLim
            nDVar = nPlDVarLim
-         ELSE IF (graphLevel == 'full') THEN
+         !ELSE IF (graphLevel == 'full') THEN
+         ELSE IF ( (graphLevel == 'full') .OR. ( graphFltFlag == 1 ) ) THEN
            nSVar = nPlSVarFull
            nDVar = nPlDVarFull
          ELSE
@@ -2696,7 +2802,8 @@
            nDVar = nDVar - nIsolutVar
          ELSE
            nDVar = nDVar + nIsolutVarFull
-           IF (graphLevel == 'full') THEN
+           !IF (graphLevel == 'full') THEN
+           IF ( (graphLevel == 'full') .OR. ( graphFltFlag == 1 ) ) THEN
              nDVar = nDVar + nSolutePPM
            END IF
            IF (fracB10 < 1.0_sdk) THEN
@@ -2756,7 +2863,8 @@
          CALL AddXtvVar(g1dAr(idx)%roan, vDynamic, v1dCc, 1, vWtr, 'roan',
'noncondensable gas density')
          CALL AddXtvVar(g1dAr(idx)%rom, vDynamic, v1dCc, 1, vWtr, 'rom', 'mixture
density')
          CALL AddXtvVar(g1dAr(idx)%am, vDynamic, v1dCc, 1, vWtr, 'am',
'noncondensable gas mass')
-         IF (graphLevel == 'full') THEN
+         !IF (graphLevel == 'full') THEN
+         IF ( (graphLevel == 'full') .OR. ( graphFltFlag == 1 ) ) THEN
            CALL AddXtvVar(g1dAr(idx)%gamn, vDynamic, v1dCc, 1, vWtr, 'gamn', 'mass
phase change rate')
            CALL AddXtvVar(plenAr(idx)%vvvul, vDynamic, v1dCc, 1, vWtr, 'vvvol', 'Gas
vol avg vel')
            CALL AddXtvVar(plenAr(idx)%vlvul, vDynamic, v1dCc, 1, vWtr, 'vlvol', 'Liq
vol avg vel')
@@ -2772,7 +2880,8 @@
            CALL AddXtvVar(g1dAr(idx)%concSn, vDynamic, v1dCc, 1, vWtr, 'concS',
'solute mass fraction')
            CALL AddXtvVar(g1dAr(idx)%sn, vDynamic, v1dCc, 1, vWtr, 'sn', 'plateout
density')
            CALL AddXtvVar(g1dAr(idx)%sSolids, vDynamic, v1dCc, 1, vWtr, 'sSolids',
'plateout mass')
-           IF (graphLevel == 'full') THEN
+           !IF (graphLevel == 'full') THEN
```

```
+            IF ( (graphLevel == 'full') .OR. ( graphFltFlag == 1 ) ) THEN
               CALL AddXtvVar(g1dAr(idx)%cSppm, vDynamic, v1dCc, 1, vWtr, 'cSppm',
'solute ppm per unit solu.')
               CALL AddXtvVar(g1dAr(idx)%cWppm, vDynamic, v1dCc, 1, vWtr, 'cWppm',
'solute ppm per unit water')
               CALL AddXtvVar(g1dAr(idx)%b10Sppm, vDynamic, v1dCc, 1, vWtr,
'b10Sppm', 'B10 ppm per unit solu.')
@@ -2794,6 +2903,7 @@
          USE Gen1DArray, ONLY: g1dAr
          USE GlobalDat, ONLY : graphLevel, stdyst
          USE PrizeVlt, ONLY: prizeTab, prizerAdj, prizerHdAdj
+         USE XtvCustom, ONLY: graphFltFlag
          USE XtvData, ONLY: vDynamic, vScalar, vHot, currentPrizer, prizerIndices,
vWtr
          USE XtvSetup, ONLY: AddXtvVar
 !
@@ -2808,10 +2918,12 @@
          LOGICAL :: prIs2D
 !
 !     calc number of variables actually output
-         IF (graphLevel == 'limited') THEN
+         !IF (graphLevel == 'limited') THEN
+         IF ( (graphLevel == 'limited') .AND. ( graphFltFlag == 0 ) ) THEN
            nSVar = nPrSVarLim
            nDVar = nPrDVarLim
-         ELSE IF (graphLevel == 'full') THEN
+         !ELSE IF (graphLevel == 'full') THEN
+         ELSE IF ( (graphLevel == 'full') .OR. ( graphFltFlag == 1 ) ) THEN
            nSVar = nPrSVarFull
            nDVar = nPrDVarFull
          ELSE
@@ -2841,7 +2953,8 @@
 !
          CALL AddXtvVar(prizeTab(idx)%z, vDynamic, vScalar, 0, vWtr, 'z', 'water
level')
          CALL AddXtvVar(prizeTab(idx)%qout, vDynamic, vScalar, 0, vWtr, 'qout',
'liquid volume discharged')
-         IF (graphLevel == 'full') THEN
+         !IF (graphLevel == 'full') THEN
+         IF ( (graphLevel == 'full') .OR. ( graphFltFlag == 1 ) ) THEN
            CALL AddXtvVar(prizeTab(idx)%qin, vDynamic, vScalar, 0, vHot, 'qin',
'heater sprayer power')
            CALL AddXtvVar(prizeTab(idx)%flow, vDynamic, vScalar, 0, vWtr, 'flow',
'volumetric flow at exit')
          END IF
@@ -2861,6 +2974,7 @@
 !
          USE GlobalDat, ONLY: graphLevel
          USE PumpVlt, ONLY: pumpTab
+         USE XtvCustom, ONLY: graphFltFlag
          USE XtvData, ONLY: vDynamic, vScalar, vWtr, vHot
          USE XtvSetup, ONLY: AddXtvVar
 !
@@ -2878,10 +2992,12 @@
          REAL(sdk) :: th = 0.0_sdk
 !
 !     calc number of variables actually output
-         IF (graphLevel == 'limited') THEN
+         !IF (graphLevel == 'limited') THEN
+         IF ( (graphLevel == 'limited') .AND. ( graphFltFlag == 0 ) ) THEN
            nSVar = nPuSVarLim
            nDVar = nPuDVarLim
-         ELSE IF (graphLevel == 'full') THEN
+         !ELSE IF (graphLevel == 'full') THEN
+         !ELSE IF (graphLevel == 'full') THEN
```

```
+         ELSE IF ( (graphLevel == 'full') .OR. ( graphFltFlag == 1 ) ) THEN
             nSVar = nPuSVarFull
             nDVar = nPuDVarFull
          ELSE
@@ -2897,7 +3013,8 @@
          CALL AddXtvVar(pumpTab(idx)%omegan, vDynamic, vScalar, 0, vHot, 'omegan',
'impeller rotational speed&
         &')
          CALL AddXtvVar(pumpTab(idx)%mflow, vDynamic, vScalar, 0, vWtr, 'mflow',
'pump mass flow')
-         IF (graphLevel == 'full') THEN
+         !IF (graphLevel == 'full') THEN
+         IF ( (graphLevel == 'full') .OR. ( graphFltFlag == 1 ) ) THEN
             CALL AddXtvVar(pumpTab(idx)%rho, vDynamic, vScalar, 0, vWtr, 'rho', 'pump
mixture density')
             CALL AddXtvVar(pumpTab(idx)%smom, vDynamic, vScalar, 0, vWtr, 'smom',
'pump momentum source')
             CALL AddXtvVar(pumpTab(idx)%alpha, vDynamic, vScalar, 0, vWtr, 'alpha',
'pump volume fraction')
@@ -2920,6 +3037,7 @@
          USE GlobalDat, ONLY : graphLevel
          USE SepdVlt, ONLY: sepdTab
          USE TeeVlt, ONLY: teeTab
+         USE XtvCustom, ONLY: graphFltFlag
          USE XtvData, ONLY: vDynamic, vScalar, vWtr
          USE XtvSetup, ONLY: AddXtvVar
 !
@@ -2936,10 +3054,12 @@
          REAL(sdk) :: th1 = 0.0_sdk
 !
 !     calc number of variables actually output
-         IF (graphLevel == 'limited') THEN
+         !IF (graphLevel == 'limited') THEN
+         IF ( (graphLevel == 'limited') .AND. ( graphFltFlag == 0 ) ) THEN
             nSVar = nSpSVarLim + nTeSVarLim
             nDVar = nSpDVarLim + nTeDVarLim
-         ELSE IF (graphLevel == 'full') THEN
+         !ELSE IF (graphLevel == 'full') THEN
+         ELSE IF ( (graphLevel == 'full') .OR. ( graphFltFlag == 1 ) ) THEN
             nSVar = nSpSVarFull + nTeSVarFull
             nDVar = nSpDVarFull + nTeDVarFull
          ELSE
@@ -2956,7 +3076,8 @@
          CALL AddXtvVar(sepdTab(idx)%xci, vDynamic, vScalar, 0, vWtr, 'xci',
'separator inlet quality')
          CALL AddXtvVar(sepdTab(idx)%xcu, vDynamic, vScalar, 0, vWtr, 'xcu', 'vapor
carryunder quality')
          CALL AddXtvVar(sepdTab(idx)%xco, vDynamic, vScalar, 0, vWtr, 'xco', 'liquid
carryover quality')
-         IF (graphLevel == 'full') THEN
+         !IF (graphLevel == 'full') THEN
+         IF ( (graphLevel == 'full') .OR. ( graphFltFlag == 1 ) ) THEN
             CALL AddXtvVar(sepdTab(idx)%veldis, vDynamic, vScalar, 0, vWtr, 'veldis',
'discharge HEM velocity')
             CALL AddXtvVar(sepdTab(idx)%vlev, vDynamic, vScalar, 0, vWtr, 'vlev',
'sepd interface velocity')
             CALL AddXtvVar(sepdTab(idx)%dpss, vDynamic, vScalar, 0, vWtr, 'dpss',
'separator pressure drop')
@@ -2978,6 +3099,7 @@
 !
          USE GlobalDat, ONLY : graphLevel
          USE TeeVlt, ONLY: teeTab
+         USE XtvCustom, ONLY: graphFltFlag
 !
```

```
          IMPLICIT NONE
 !
@@ -2993,10 +3115,12 @@
          INTEGER(sik) :: nSVar, nDVar
 !
 !     calc number of variables actually output
-         IF (graphLevel == 'limited') THEN
+         !IF (graphLevel == 'limited') THEN
+         IF ( (graphLevel == 'limited') .AND. ( graphFltFlag == 0 ) ) THEN
            nSVar = nTeSVarLim
            nDVar = nTeDVarLim
-         ELSE IF (graphLevel == 'full') THEN
+         !ELSE IF (graphLevel == 'full') THEN
+         ELSE IF ( (graphLevel == 'full') .OR. ( graphFltFlag == 1 ) ) THEN
            nSVar = nTeSVarFull
            nDVar = nTeDVarFull
          ELSE
@@ -3021,6 +3145,7 @@
 !
          USE GlobalDat, ONLY : graphLevel
          USE TeeVlt, ONLY: teeTab
+         USE XtvCustom, ONLY: graphFltFlag
          USE XtvData, ONLY: vDynamic, vScalar, vHot
          USE XtvSetup, ONLY: AddXtvVar
 !
@@ -3031,7 +3156,8 @@
 !     Output Tee specific variables
 !
-         IF (graphLevel == 'full') THEN
+         !IF (graphLevel == 'full') THEN
+         IF ( (graphLevel == 'full') .OR. ( graphFltFlag == 1 ) ) THEN
            CALL AddXtvVar(teeTab(idx)%powr1, vDynamic, vScalar, 0, vHot, 'powr1',
'heater power to main tube')
            CALL AddXtvVar(teeTab(idx)%powr2, vDynamic, vScalar, 0, vHot, 'powr2',
'heater power to side tube')
          END IF
@@ -3048,6 +3174,7 @@
          USE GlobalDat, ONLY : graphLevel
          USE TeeVlt, ONLY: teeTab
          USE TurbVlt, ONLY: turbTab
+         USE XtvCustom, ONLY: graphFltFlag
          USE XtvData, ONLY: vDynamic, vScalar, vHot
          USE XtvSetup, ONLY: AddXtvVar
 !
@@ -3064,10 +3191,12 @@
          REAL(sdk) :: th1 = 0.0_sdk
 !
 !     calc number of variables actually output
-         IF (graphLevel == 'limited') THEN
+         !IF (graphLevel == 'limited') THEN
+         IF ( (graphLevel == 'limited') .AND. ( graphFltFlag == 0 ) ) THEN
            nSVar = nTbSVarLim + nTeSVarLim
            nDVar = nTbDVarLim + nTeDVarLim
-         ELSE IF (graphLevel == 'full') THEN
+         !ELSE IF (graphLevel == 'full') THEN
+         ELSE IF ( (graphLevel == 'full') .OR. ( graphFltFlag == 1 ) ) THEN
            nSVar = nTbSVarFull + nTeSVarFull
            nDVar = nTbDVarFull + nTeDVarFull
          ELSE
@@ -3096,6 +3225,7 @@
          USE Gen1DArray, ONLY: g1dAr
          USE GlobalDat, ONLY : graphLevel
          USE ValveVlt, ONLY: valveTab, vSwing
```

```
+         USE XtvCustom, ONLY: graphFltFlag
          USE XtvData, ONLY: vDynamic, vScalar, vWtr
          USE XtvSetup, ONLY: AddXtvVar
 !
@@ -3113,10 +3243,12 @@

 !
 !     calc number of variables actually output
-         IF (graphLevel == 'limited') THEN
+         !IF (graphLevel == 'limited') THEN
+         IF ( (graphLevel == 'limited') .AND. ( graphFltFlag == 0 ) ) THEN
            nSVar = nVlSVarLim
            nDVar = nVlDVarLim
-         ELSE IF (graphLevel == 'full') THEN
+         !ELSE IF (graphLevel == 'full') THEN
+         ELSE IF ( (graphLevel == 'full') .OR. ( graphFltFlag == 1 ) ) THEN
            nSVar = nVlSVarFull
            nDVar = nVlDVarFull
          ELSE
@@ -3169,6 +3301,7 @@
          USE VessArray3, ONLY: vsAr3
          USE VessArray4, ONLY: vsAr4
          USE VessVlt, ONLY: vessTab
+         USE XtvCustom, ONLY: graphFltFlag
          USE XtvData, ONLY: cyl3d, cart3d, vStatic, v3dCc, vWtr, vDynamic, v3dFaI,
v3dFaJ, v3dFaK
          USE XtvData, ONLY: vScalar, vectI, vectK, vectJ
          USE XtvSetup, ONLY: AddXtvComp, CreateArZerFrAr, AddXtvTemplate, AddXtvJun,
AddXtvVar
@@ -3239,10 +3372,12 @@
          gravAr = vessTab(idx)%gravz
 !
 !     calc number of variables actually output
-         IF (graphLevel == 'limited') THEN
+         !IF (graphLevel == 'limited') THEN
+         IF ( (graphLevel == 'limited') .AND. ( graphFltFlag == 0 ) ) THEN
            nSVar = nVsSVarLim
            nDVar = nVsDVarLim
-         ELSE IF (graphLevel == 'full') THEN
+         !ELSE IF (graphLevel == 'full') THEN
+         ELSE IF ( (graphLevel == 'full') .OR. ( graphFltFlag == 1 ) ) THEN
            nSVar = nVsSVarFull
            nDVar = nVsDVarFull
          ELSE
@@ -3251,6 +3386,8 @@

          IF (disFields .GT. 0) THEN
            nDVar = nDVar + 5*disFields
+           !ZO41 - These additions of 4 or 14 XTV variables conditioned by option
nBubbles seem incorrect,
+           !       since there is no call of AddXtvVar() conditioned by option
nBubbles in this subroutine
            IF (nBubbles .EQ. 1_sik) THEN
              nDVar = nDvar + 4_sik
            ELSE IF (nBubbles .EQ. 2_sik) THEN
@@ -3262,7 +3399,8 @@
            nDVar = nDVar - nIsolutVar
          ELSE
            nDVar = nDVar + nIsoluteVarFull
-           IF (graphLevel == 'full') THEN
+           !IF (graphLevel == 'full') THEN
+           IF ( (graphLevel == 'full') .OR. ( graphFltFlag == 1 ) ) THEN
              nDVar = nDVar + nSolutePPM
            END IF
```

```
             IF (fracB10 < 1.0_sdk) THEN
@@ -3300,7 +3438,8 @@
                nDVar = nDVar + nTraceL
             END IF
          END IF
-         IF (vessTab(idx)%nolt > 0 .AND. graphLevel == 'full') THEN
+         !IF (vessTab(idx)%nolt > 0 .AND. graphLevel == 'full') THEN
+         IF ( vessTab(idx)%nolt > 0 .AND. ( (graphLevel == 'full') .OR. (graphFltFlag
== 1) ) ) THEN
            nDVar = nDVar - nLev3DVar
          END IF
          nVect = 2
@@ -3432,7 +3571,8 @@


 ! Additional vars (graphLevel = full)
-         IF (graphLevel == 'full') THEN
+         !IF (graphLevel == 'full') THEN
+         IF ( (graphLevel == 'full') .OR. ( graphFltFlag == 1 ) ) THEN
            CALL AddXtvVar(vsAr3(idx)%qsl(ic0:ix, jc0:jx, k0:kx), vDynamic, v3dCc, 1,
vWtr, 'qsl', 'heat structu&
            &re power')

@@ -3521,7 +3661,8 @@
            &nsity')
            CALL AddXtvVar(vsAr3(idx)%sSolids(ic0:ix, jc0:jx, k0:kx), vDynamic,
v3dCc, 1, vWtr, 'sSolids', &
            & 'plateout mass')
-           IF (graphLevel == 'full') THEN
+           !IF (graphLevel == 'full') THEN
+           IF ( (graphLevel == 'full') .OR. ( graphFltFlag == 1 ) ) THEN
              CALL AddXtvVar(vsAr3(idx)%cSppm(ic0:ix, jc0:jx, k0:kx), vDynamic,
v3dCc, 1, vWtr, 'cSppm', &
              & 'solute ppm per unit solu.')
              CALL AddXtvVar(vsAr3(idx)%cWppm(ic0:ix, jc0:jx, k0:kx), vDynamic,
v3dCc, 1, vWtr, 'cWppm', &
@@ -3693,6 +3834,7 @@
          USE ContanData, ONLY: HtStruct
          USE Flt, ONLY: genTab
          USE GlobalDat, ONLY : graphLevel
+         USE XtvCustom, ONLY: graphFltFlag
          USE XtvData, ONLY: vDynamic, v1dCc, vWtr, vScalar
          USE XtvSetup, ONLY: AddXtvComp, CreateArFrVal, CreateSArFrVal,
AddXtvTemplate, AddXtvVar
 !
@@ -3711,10 +3853,12 @@
          c8type = "ContHS  "
 !
 !     calc number of variables actually output
-         IF (graphLevel == 'limited') THEN
+         !IF (graphLevel == 'limited') THEN
+         IF ( (graphLevel == 'limited') .AND. ( graphFltFlag == 0 ) ) THEN
            nSVar = nCtHSSVarLim
            nDVar = nCtHSDVarLim
-         ELSE IF (graphLevel == 'full') THEN
+         !ELSE IF (graphLevel == 'full') THEN
+         ELSE IF ( (graphLevel == 'full') .OR. ( graphFltFlag == 1 ) ) THEN
            nSVar = nCtHSSVarFull
            nDVar = nCtHSDVarFull
          ELSE
@@ -3755,6 +3899,7 @@
          USE ContanData, ONLY: contanTab, tHSCOntan
          USE Flt, ONLY: genTab
          USE GlobalDat, ONLY : graphLevel
```

```
+         USE XtvCustom, ONLY: graphFltFlag
          USE XtvData, ONLY: vDynamic, v1dCc, vWtr
          USE XtvSetup, ONLY: AddXtvComp, CreateArFrVal, CreateSArFrVal,
AddXtvTemplate, AddXtvVar
 !
@@ -3772,10 +3917,12 @@
          c8type = "THSCont "
 !
 !     calc number of variables actually output
-         IF (graphLevel == 'limited') THEN
+         !IF (graphLevel == 'limited') THEN
+         IF ( (graphLevel == 'limited') .AND. ( graphFltFlag == 0 ) ) THEN
            nSVar = nCtTHSSVarLim
            nDVar = nCtTHSDVarLim
-         ELSE IF (graphLevel == 'full') THEN
+         !ELSE IF (graphLevel == 'full') THEN
+         ELSE IF ( (graphLevel == 'full') .OR. ( graphFltFlag == 1 ) ) THEN
            nSVar = nCtTHSSVarFull
            nDVar = nCtTHSDVarFull
          ELSE
@@ -3798,7 +3945,8 @@
          CALL AddXtvVar(tHSContan%qLiq, vDynamic, v1dCc, 1, vWtr, 'qLiqHT', 'Liquid
Heat Transfer Rate')
          CALL AddXtvVar(tHSContan%qVap, vDynamic, v1dCc, 1, vWtr, 'qVapHT', 'Vapor
Heat Transfer Rate')
          CALL AddXtvVar(tHSContan%filmTh, vDynamic, v1dCc, 1, vWtr, 'filmTh', 'Film
Thickness')
-         IF (graphLevel == 'full') THEN
+         !IF (graphLevel == 'full') THEN
+         IF ( (graphLevel == 'full') .OR. ( graphFltFlag == 1 ) ) THEN
            CALL AddXtvVar(tHSContan%rmf, vDynamic, v1dCc, 1, vWtr, 'rmf', 'Film
Mass')
            CALL AddXtvVar(tHSContan%filmMT, vDynamic, v1dCc, 1, vWtr, 'filmMT',
'Film Mass Transfer')
            CALL AddXtvVar(tHSContan%filmDR, vDynamic, v1dCc, 1, vWtr, 'filmDR',
'Film Drain Rate')
```

B.6 Patch file for src/XtvDumpM.f90

```
--- XtvDumpM.f90    2017-09-29 11:22:49.000000002 +0200
+++ XtvDumpM.f90.mod       2018-07-02 18:29:30.000000002 +0200
@@ -9,6 +9,8 @@
 !     contained in or refernced by the data structures in XtvData.
 !     It has (almost) no understanding of TRAC or TRAC components.
 !
+!     Modified by Paul Scherrer Institut (07/18) - to include custom XTV output
capability.
+!
 !     BEGIN MODULE USE
       USE IntrType
       USE Io
@@ -133,6 +135,7 @@
          USE CXtvXFaces
          USE GlobalDat, ONLY: cput, eTime, istdy, nStep, nStepT, graphLevel, cpuTime,
cpuflg
          USE SysTime
+         USE XtvCustom, ONLY: graphFltFlag
          USE XtvData
          USE SnapIntf
          USE TDMRVarDecl
@@ -149,7 +152,8 @@
          IF (nStep .NE. 0 .AND. eTime .EQ. 0.0d0) RETURN
          plted = .TRUE.
 !
-         IF (graphLevel .NE. 'minimal') THEN
+         !IF (graphLevel .NE. 'minimal') THEN
+         IF ( (graphLevel .NE. 'minimal') .OR. ( graphFltFlag == 1 ) ) THEN
            ALLOCATE(prizerVars(nPrizers))
 !          Adjust Steady State Prizer data
            IF (istdy .NE. 0) THEN
@@ -210,7 +214,8 @@
            CALL error(1, '*XtvDump* error writing xtv data block header.')
          END IF
 !
-         IF (graphLevel /= 'minimal') THEN
+         !IF (graphLevel /= 'minimal') THEN
+         IF ( (graphLevel /= 'minimal') .OR. ( graphFltFlag == 1 ) ) THEN
            IF (istdy .NE. 0) THEN
              CALL UnAdjPrizerDVars(prizerVars)
            END IF
```

B.7 Patch file for src/XtvSetupM.f90

```
--- XtvSetupM.f90   2017-09-29 11:22:49.000000002 +0200
+++ XtvSetupM.f90.mod      2018-07-02 18:29:20.000000002 +0200
@@ -9,6 +9,9 @@
 !     the data structures in XtvData. It performs no output to
 !     the graphics file.
 !
+!     Modified by Paul Scherrer Institut (07/18) - to include custom XTV output
capability.
+!
+!
 !     BEGIN MODULE USE
       USE IntrType
       USE Io
@@ -721,6 +724,8 @@
 !     Generic interface for Scalar reals.  This routine should not
 !     be accessed directly.
 !
+         USE GlobalDat, ONLY: graphLevel
+         USE XtvCustom, ONLY: graphFltFlag, XtvVarFilterOut
          USE XtvData
 !
          IMPLICIT NONE
@@ -734,6 +739,8 @@
 !
          TYPE(xtvVarT), POINTER :: curVar
 !
+         INTEGER(sik) FltFlag
+!
 !     set curVar Ptr
          IF (varList .EQ. vStatic) THEN
            curSVarIdx = curSVarIdx + 1
@@ -744,6 +751,14 @@
            curVar => xtvCompList(curCmpIdx)%sVar(curSVarIdx)
            totalStaticChannels = totalStaticChannels + 1
          ELSE IF (varList .EQ. vDynamic) THEN
+!          Filter for custom XTV output
+           IF (graphFltFlag == 1_sik) THEN
+              CALL XtvVarFilterOut(xtvCompList(curCmpIdx)%compId, name, graphLevel,
FltFlag)
+              IF (FltFlag == 1_sik) THEN
+                  xtvCompList(curCmpIdx)%nDVar = xtvCompList(curCmpIdx)%nDVar - 1
+                  RETURN
+              END IF
+           END IF
            curDVarIdx = curDVarIdx + 1
            IF (curDVarIdx .GT. xtvCompList(curCmpIdx)%nDVar) THEN
              CALL error(5, '*AddScalarRVar* more Dynamic'//' vars than allocated
space, &
@@ -801,6 +816,8 @@
 !     Generic interface for Scalar Integers.  This routine should not
 !     be accessed directly.
 !
+         USE GlobalDat, ONLY: graphLevel
+         USE XtvCustom, ONLY: graphFltFlag, XtvVarFilterOut
          USE XtvData
 !
          IMPLICIT NONE
@@ -814,6 +831,8 @@
 !
          TYPE(xtvVarT), POINTER :: curVar
 !
```

```
+         INTEGER(sik) FltFlag
+!
 !     set curVar Ptr
          IF (varList .EQ. vStatic) THEN
             curSVarIdx = curSVarIdx + 1
@@ -824,6 +843,14 @@
             curVar => xtvCompList(curCmpIdx)%sVar(curSVarIdx)
             totalStaticChannels = totalStaticChannels + 1
          ELSE IF (varList .EQ. vDynamic) THEN
+!           Filter for custom XTV output
+            IF (graphFltFlag == 1_sik) THEN
+               CALL XtvVarFilterOut(xtvCompList(curCmpIdx)%compId, name, graphLevel,
FltFlag)
+               IF (FltFlag == 1_sik) THEN
+                   xtvCompList(curCmpIdx)%nDVar = xtvCompList(curCmpIdx)%nDVar - 1
+                   RETURN
+               END IF
+            END IF
             curDVarIdx = curDVarIdx + 1
             IF (curDVarIdx .GT. xtvCompList(curCmpIdx)%nDVar) THEN
                CALL error(5, '*AddScalarIVar* more Dynamic'//' vars than allocated
space, &
@@ -882,6 +909,8 @@
 !     Generic interface for 1D reals.  This routine should not
 !     be accessed directly.
 !
+         USE GlobalDat, ONLY: graphLevel
+         USE XtvCustom, ONLY: graphFltFlag, XtvVarFilterOut
          USE XtvData
 !
          IMPLICIT NONE
@@ -895,6 +924,8 @@
 !
          TYPE(xtvVarT), POINTER :: curVar
 !
+         INTEGER(sik) FltFlag
+!
 !     set curVar Ptr
          IF (varList .EQ. vStatic) THEN
             curSVarIdx = curSVarIdx + 1
@@ -904,6 +935,14 @@
             END IF
             curVar => xtvCompList(curCmpIdx)%sVar(curSVarIdx)
          ELSE IF (varList .EQ. vDynamic) THEN
+!           Filter for custom XTV output
+            IF (graphFltFlag == 1_sik) THEN
+               CALL XtvVarFilterOut(xtvCompList(curCmpIdx)%compId, name, graphLevel,
FltFlag)
+               IF (FltFlag == 1_sik) THEN
+                   xtvCompList(curCmpIdx)%nDVar = xtvCompList(curCmpIdx)%nDVar - 1
+                   RETURN
+               END IF
+            END IF
             curDVarIdx = curDVarIdx + 1
             IF (curDVarIdx .GT. xtvCompList(curCmpIdx)%nDVar) THEN
                CALL error(5, '*AddVectorR1Var* more '//'dynamic vars than allocated
space,&
@@ -983,6 +1022,8 @@
 !     Generic interface for 1D Integers.  This routine should not
 !     be accessed directly.
 !
+         USE GlobalDat, ONLY: graphLevel
+         USE XtvCustom, ONLY: graphFltFlag, XtvVarFilterOut
          USE XtvData
```

```
 !
          IMPLICIT NONE
@@ -996,6 +1037,8 @@
 !
          TYPE(xtvVarT), POINTER :: curVar
 !
+         INTEGER(sik) FltFlag
+!
 !     set curVar Ptr
          IF (varList .EQ. vStatic) THEN
             curSVarIdx = curSVarIdx + 1
@@ -1005,6 +1048,14 @@
             END IF
             curVar => xtvCompList(curCmpIdx)%sVar(curSVarIdx)
          ELSE IF (varList .EQ. vDynamic) THEN
+!           Filter for custom XTV output
+            IF (graphFltFlag == 1_sik) THEN
+               CALL XtvVarFilterOut(xtvCompList(curCmpIdx)%compId, name, graphLevel,
FltFlag)
+               IF (FltFlag == 1_sik) THEN
+                   xtvCompList(curCmpIdx)%nDVar = xtvCompList(curCmpIdx)%nDVar - 1
+                   RETURN
+               END IF
+            END IF
             curDVarIdx = curDVarIdx + 1
             IF (curDVarIdx .GT. xtvCompList(curCmpIdx)%nDVar) THEN
                CALL error(5, '*AddVectorI1Var* more '//'dynamic vars than allocated
space,&
@@ -1080,6 +1131,8 @@
 !     Generic interface for 2D reals.  This routine should not
 !     be accessed directly.
 !
+         USE GlobalDat, ONLY: graphLevel
+         USE XtvCustom, ONLY: graphFltFlag, XtvVarFilterOut
          USE XtvData
 !
          IMPLICIT NONE
@@ -1094,6 +1147,8 @@
          TYPE(xtvVarT), POINTER :: curVar
          TYPE(xtvTemplateT), POINTER :: curTempl
 !
+         INTEGER(sik) FltFlag
+!
 !     set curVar Ptr
          IF (varList .EQ. vStatic) THEN
             curSVarIdx = curSVarIdx + 1
@@ -1103,6 +1158,14 @@
             END IF
             curVar => xtvCompList(curCmpIdx)%sVar(curSVarIdx)
          ELSE IF (varList .EQ. vDynamic) THEN
+!           Filter for custom XTV output
+            IF (graphFltFlag == 1_sik) THEN
+               CALL XtvVarFilterOut(xtvCompList(curCmpIdx)%compId, name, graphLevel,
FltFlag)
+               IF (FltFlag == 1_sik) THEN
+                   xtvCompList(curCmpIdx)%nDVar = xtvCompList(curCmpIdx)%nDVar - 1
+                   RETURN
+               END IF
+            END IF
             curDVarIdx = curDVarIdx + 1
             IF (curDVarIdx .GT. xtvCompList(curCmpIdx)%nDVar) THEN
                CALL error(5, '*AddVectorR2Var* more '//'dynamic vars than allocated
space,&
@@ -1186,6 +1249,8 @@
```

```
 !     Generic interface for 3D reals.  This routine should not
 !     be accessed directly.
 !
+         USE GlobalDat, ONLY: graphLevel
+         USE XtvCustom, ONLY: graphFltFlag, XtvVarFilterOut
          USE XtvData
 !
          IMPLICIT NONE
@@ -1200,6 +1265,8 @@
          TYPE(xtvVarT), POINTER :: curVar
          TYPE(xtvTemplateT), POINTER :: curTempl
 !
+         INTEGER(sik) FltFlag
+!
 !     set curVar Ptr
          IF (varList .EQ. vStatic) THEN
            curSVarIdx = curSVarIdx + 1
@@ -1209,6 +1276,14 @@
             END IF
             curVar => xtvCompList(curCmpIdx)%sVar(curSVarIdx)
          ELSE IF (varList .EQ. vDynamic) THEN
+!           Filter for custom XTV output
+            IF (graphFltFlag == 1_sik) THEN
+               CALL XtvVarFilterOut(xtvCompList(curCmpIdx)%compId, name, graphLevel,
FltFlag)
+               IF (FltFlag == 1_sik) THEN
+                   xtvCompList(curCmpIdx)%nDVar = xtvCompList(curCmpIdx)%nDVar - 1
+                   RETURN
+               END IF
+            END IF
             curDVarIdx = curDVarIdx + 1
             IF (curDVarIdx .GT. xtvCompList(curCmpIdx)%nDVar) THEN
               CALL error(5, '*AddVectorR3Var* more '//'dynamic vars than allocated
space,&
@@ -1305,6 +1380,8 @@
 !     Generic interface for 3D Integers.  This routine should not
 !     be accessed directly.
 !
+         USE GlobalDat, ONLY: graphLevel
+         USE XtvCustom, ONLY: graphFltFlag, XtvVarFilterOut
          USE XtvData
 !
          IMPLICIT NONE
@@ -1319,6 +1396,8 @@
          TYPE(xtvVarT), POINTER :: curVar
          TYPE(xtvTemplateT), POINTER :: curTempl
 !
+         INTEGER(sik) FltFlag
+!
 !     set curVar Ptr
          IF (varList .EQ. vStatic) THEN
            curSVarIdx = curSVarIdx + 1
@@ -1328,6 +1407,14 @@
             END IF
             curVar => xtvCompList(curCmpIdx)%sVar(curSVarIdx)
          ELSE IF (varList .EQ. vDynamic) THEN
+!           Filter for custom XTV output
+            IF (graphFltFlag == 1_sik) THEN
+               CALL XtvVarFilterOut(xtvCompList(curCmpIdx)%compId, name, graphLevel,
FltFlag)
+               IF (FltFlag == 1_sik) THEN
+                   xtvCompList(curCmpIdx)%nDVar = xtvCompList(curCmpIdx)%nDVar - 1
+                   RETURN
+               END IF
```

```
+            END IF
             curDVarIdx = curDVarIdx + 1
             IF (curDVarIdx .GT. xtvCompList(curCmpIdx)%nDVar) THEN
                CALL error(5, '*AddVectorI3Var* more dynamic vars than allocated
space, &
```

APPENDIX C
TEST CASES SELECTION FOR THE VERIFICATION OF V5.0P5

This appendix lists all the test cases included in the official release of TRACE v5.0p5 [1] that have been considered for the verification procedure.

Each subsection C.1 to C.24 corresponds to one of the verification suites defined by the code developers. Each of the suites focuses on the testing of a particular aspect of the code, such as the control system, the different thermal-hydraulic components, or other relevant modelling options. It is materialized by a subfolder with a self-explanatory name as provided by the code developers (e.g. subfolders “./Accum”, “./ConSys”, “./HtStr”, “./Power”, “./TraceSpecies”, etc.).

The tables list all the original models, including the ones that have been rejected with a mention of the reason for non-selection. The most frequent reason for non-selection was the absence of an XTV file after execution of the reference model and reference code executable (a total of 105 models, marked as ‘no XTV file’ in the tables). Also, in few instances, namely 4 models in suites “./ConSys”, an XTV file was generated but could not be post-processed using AptPlot (marked as ‘faulty XTV file’). These two categories of models were designed to test input errors and therefore were not relevant to this verification study.

Moreover, models rejected because of the use of fixed-format input have been marked as ‘fixed-format input’ in the tables (19 models in total). And, for test models that require a restart, necessary information to identify the upstream model is provided.

It should be noted that, contrary to previous code versions, all the test models of the official release v5.0p5 [1] have been provided as a single folder merging all the input files together. Nevertheless, using information in the comment sections of each input file of release v5.0p5 it was possible to reconstruct the different suites listed in C.1 to C.24. The scripts used to reconstruct the suites are described appendix 2, subsection A.2.3 of [4].

Compared to previous versions, one can also note that release v5.0p5 involved a supplementary suites, labelled “./PARCS”, that includes new test cases for the verification of the spatial kinetics option of TRACE (enabled by Namelist option itdmr=1). For these models, information on the additional PARCS input model is also provided, including restart information when applicable (see subsection C.13).

C.1 Suites ./Accum

Number of selected input files: 9
Input files modified for selection: 0

Number of non-selected input files: 1
Input files written in fixed-format: 0
Input files yielding no or faulty XTV: 1

Total number of input files: 10

Model in ./Accum	Restart file	Selected	Reason for non-selection
accum1.inp		yes	
accum1LT.inp		yes	
accum1LTa.inp		yes	
accum2.inp		yes	
accum2errors.inp			no XTV file
accum2LTa.inp		yes	
accum2LT.inp		yes	
accum2-2.inp		yes	
AccumDischarge.inp		yes	
pipeLT.inp		yes	

C.2 Suites ./ATW

Number of selected input files: 8
Input files modified for selection: 0

Number of non-selected input files: 0
Input files written in fixed-format: 0
Input files yielding no or faulty XTV: 0

Total number of input files: 8

Model in ./ATW	Restart file	Selected	Reason for non-selection
drain_ATW.inp		yes	
multiloop.inp		yes	
O2Injection.inp		yes	
PbBi_flcond.inp		yes	
PbBi_pump01.inp		yes	
PbBi_pump02.inp		yes	
tarloop.inp		yes	
two_fluid_HX.inp		yes	

C.3 Suites ./Chan

Number of selected input files: 60
Input files modified for selection: 0

Number of non-selected input files: 2
Input files written in fixed-format: 0
Input files yielding no or faulty XTV: 2

Total number of input files: 62

Model in ./Chan	Restart file	Selected	Reason for non-selection
3CHAN.inp		yes	
3CHANr.inp	**3CHAN.tpr**	yes	
3CHAN.NotRad.inp		yes	
3CHAN_FE_Rev1.inp		yes	
3CHAN_FE_Rev2.inp		yes	
3CHAN_FE_Rev3.inp		yes	
3ChanReflood_Eng.inp		yes	
3ChanReflood.inp		yes	
3ChanRefloodRst.inp	**3ChanReflood.tpr**	yes	
3ChanRefloodR1.inp	**3ChanReflood.Mod3.tpr**	yes	
3ChanRefloodR2.inp	**3ChanReflood.Mod3.tpr**	yes	
3ChanRefloodR3.inp	**3ChanReflood.Mod3.tpr**	yes	
3ChanReflood.Mod1.inp		yes	
3ChanReflood.Mod2.inp		yes	
3ChanReflood.Mod3.inp		yes	
3ChanRefloodRst.Mod1.inp	**3ChanReflood.Mod1.tpr**	yes	

Model in ./Chan	Restart file	Selected	Reason for non-selection
3ChanRefloodRst.Mod2.inp	3ChanReflood.Mod2.tpr	yes	
3ChanRefloodRst.Mod3.inp	3ChanReflood.Mod3.tpr	yes	
3ChanRefloodRst.Mod4.inp	3ChanReflood.Mod3.tpr	yes	
BFerryChanSSM.inp		yes	
BFerryChanTRM.inp	BFerryChanSSM.tpr	yes	
CHANFineMesh.inp		yes	
CHANIAXCND0.inp		yes	
CHANIAXCND10.inp		yes	
CHANIAXCND1.inp		yes	
CHANIAXCND9.inp		yes	
chanoldmatch.inp		yes	
chanoldmatchname.inp		yes	
ChanReflood.Mod1.inp		yes	
Chansuccessgadmax.inp		yes	
Chansuccessgadmin.inp		yes	
Chansuccess.inp		yes	
delzLK.inp		yes	
delzLKNoWaterRod.inp		yes	
delzLKWRLast.inp		yes	
GEFBPOP.inp			no XTV file
grid_chan.inp		yes	

Model in ./Chan	Restart file	Selected	Reason for non-selection
grid_chan.Rev1.inp		yes	
grid_chan.Rev1R.inp	grid_chan.Rev1.tpr	yes	
NoDelzLKNoWaterRod.inp		yes	
OrnlChanM.inp		yes	
Req1chan.inp		yes	
Req1chan-rest.inp	Req1chan.tpr	yes	
Req1pipe.inp		yes	
Req1pipe-rest.inp	Req1pipe.tpr	yes	
Req3-1Dchan.inp		yes	
Req3-1Dslab.inp		yes	
Req3-3Dchan.inp		yes	
Req3-3Dslab.inp		yes	
Req4-1rod.inp		yes	
Req4-4rod.inp		yes	
Req5-10chans.inp		yes	
Req5-1chan.inp		yes	
Req6-ichf1.inp		yes	
Req6-ichf2.inp		yes	
Req6-ichf3.inp		yes	
Req7.1chan.inp		yes	
Req7.6.inp		yes	

Model in ./Chan	Restart file	Selected	Reason for non-selection
Req7.7.inp		yes	
Req7-Simple.inp		yes	
two_chan.inp			no XTV file
ZeroDelzLKNoWaterRod.inp		yes	

C.4 Suites ./ConSys

Number of selected input files: 98
Input files modified for selection: 10 (marked in table as 'yes*')

Number of non-selected input files: 29
Input files written in fixed-format: 0
Input files yielding no or faulty XTV: 29 (including 4 faulty XTV files)

Total number of input files: 127

Model in ./ConSys	Restart file	Selected	Reason for non-selection
AxialPowerCheck3.inp		yes	
CnImpLoopM.inp		yes	
CnSys1M.inp		yes	
CnSysDelt1.inp		yes	
CnSysDelt2.inp		yes	
CnSysM.inp		yes	
conblock1.inp			no XTV file
conblock2.inp			no XTV file
conblock3.inp			no XTV file
conblock3_rst1.inp	**conblock3.tpr**		no XTV file
conblock3_rst2.inp	**conblock3.tpr**		no XTV file
conblock4.inp			no XTV file
conblock5.inp			no XTV file
conblock6.inp		yes	
conblock7.inp		yes	
ConDrpg.inp			faulty XTV file

Model in ./ConSys	Restart file	Selected	Reason for non-selection
ConFlow.inp		yes	
ConLevel.inp		yes	
ConPress1.inp		yes	
ConPress2.inp		yes	
ConPress3.inp		yes	
ConRodA.inp		yes	
ConRodP.inp		yes	
CoreAvBoronConcen.inp		yes	
CoreAvBoronConcen.Mod1.inp		yes	
CoreAvBoronConcen.Mod1R.inp	CoreAvBoronConcen.Mod1.tpr	yes	
CoreAvBoronConcen.Mod2.inp		yes	
CoreAvBoronConcen.Mod3A.inp		yes*	
CoreAvBoronConcen.Mod3B.inp		yes	
CoreAvBoronConcen.Mod3C.inp		yes	
CoreAvBoronConcen.Mod3D.inp		yes*	
CoreAvBoronConcen.Mod3E.inp		yes*	
CoreAvBoronConcen.Mod3F.inp		yes*	
CoreAvBoronConcen.Mod3G.inp		yes*	
CoreAvBoronConcen.Mod3H.inp		yes*	
CoreAvBoronConcen.Mod3I.inp		yes*	
CoreAvBoronConcen.Mod3.inp		yes	

Model in ./ConSys	Restart file	Selected	Reason for non-selection
CoreAvBoronConcen.Mod4.inp		yes	
CoreAvBoronConcen.Mod5.inp		yes	
CoreAvBoronConcen.Mod6.inp		yes	
CoreAvBoronConcen.Mod7.inp		yes	
CoreAvBoronConcen.Mod7R.inp	CoreAvBoronConcen.Mod7.tpr	yes	
CoreAvBoronConcen.Mod8.inp		yes	
CoreAvBoronConcen.Mod8R.inp	CoreAvBoronConcen.Mod8.tpr	yes	
CoreAvBoronConcen.Mod9.inp		yes	
CoreAvBoronConcen.Mod10.inp		yes	
CoreAvBoronConcen.Mod11.inp		yes	
CoreAvBoronConcen.Mod12.inp		yes	
DownComerLvl_err.inp			no XTV file
DownComerLvl.inp		yes	
DrainValve.inp		yes	
genSVTest1.inp		yes	
genSVTest2.inp		yes	
genSVTest3.inp		yes	
genSVTest4.inp		yes	
genSVTest5.inp		yes	
genSVTest6.inp			no XTV file
genSVTest7.inp			no XTV file

Model in ./ConSys	Restart file	Selected	Reason for non-selection
genSVTest8.inp			no XTV file
genSVTest9.inp			no XTV file
genSVTest10.inp		yes	
genSVTest11.inp		yes	
lcbn78.inp		yes	
lcbn78re.inp	**lcbn78.tpr**	yes	
lmpLoopNew.inp		yes	
lmpLoopNewR.inp	**lmpLoopNew.tpr**	yes	
lint.inp		yes	
napcss.1.inp		yes	
napcss.2.inp		yes	
napcss.inp		yes	
power_test1.inp			faulty XTV file
power_test2.inp			faulty XTV file
power_test3.inp			faulty XTV file
power_test4.inp		yes	
power_test.inp		yes	
PromHeat0.1.inp		yes	
PromHeat0.1R.inp	**PromHeat0.1.tpr**	yes	
PumpTorq.inp		yes	
PumpTorq2.inp		yes	

Model in ./ConSys	Restart file	Selected	Reason for non-selection
PumpTorq3.inp		yes	
PumpTorq3R.inp	**PumpTorq3.tpr**	yes	
PumpTorq4.inp			no XTV file
PumpTorq5.inp			no XTV file
rst_tr175.inp	**ss_tr175.tpr**	yes	
ShellHTM.inp		yes	
SigHmLvl.inp		yes	
sigvar1.inp		yes	
sigvar2.inp		yes	
sigvar3.inp		yes	
sigvar4.inp		yes	
sigvar5.inp		yes	
sigvar6.inp		yes	
sigvar7.inp			no XTV file
sigvar8.inp			no XTV file
sigvar9.inp			no XTV file
sigvar10.inp			no XTV file
sigvar11.inp		yes	
sigvar12.inp		yes	
sigvar13.inp		yes	
sigvar14.inp		yes	

Model in ./ConSys	Restart file	Selected	Reason for non-selection
sigvarErr1.inp		yes	
ss_tr175.inp		yes	
TestDupCB.inp			no XTV file
testltf1.inp		yes*	
testltf1n.inp		yes	
testltf2.inp		yes*	
testltf2n.inp		yes	
testltf3.inp		yes*	
testltf3n.inp		yes	
trip1.inp			no XTV file
trip2.inp		yes	
trip3.inp			no XTV file
trip4.inp			no XTV file
trip5.inp		yes	
trip6.inp			no XTV file
trip7.inp			no XTV file
trip8.inp			no XTV file
trip9.inp		yes	
trip_drain.inp		yes	
trip_marvik.inp		yes	
trip_zionpwr.inp		yes	

Model in ./ConSys	Restart file	Selected	Reason for non-selection
TurbTorqueSpeed.inp		yes	
VsslMixtNoLevM.inp	**VsslMixtNoLevSSM.tpr**	yes	
VsslMixtNoLevSSM.inp		yes	
w4loopErr.inp		yes	
w4loopi.inp		yes	
w4loopri.inp	**w4loopi.tpr**	yes	

C.5 Suites ./DeadEnd

Number of selected input files: 25
Input files modified for selection: 1 (marked in table as 'yes*')

Number of non-selected input files: 0
Input files written in fixed-format: 0
Input files yielding no or faulty XTV: 0

Total number of input files: 25

Model in ./DeadEnd	Restart file	Selected	Reason for non-selection
deadEndG.inp		yes	
deadEndGr.inp	**deadEndG.tpr**	yes	
deadEndL.inp		yes	
deadEndLr.inp	**deadEndL.tpr**	yes	
deadEndside.inp		yes*	
deadEndw4.inp		yes	
deadEndw4r.inp	**deadEndw4.tpr**	yes	
fillEndG.inp		yes	
fillEndGr.inp	**fillEndG.tpr**	yes	
fillEndL.inp		yes	
fillEndLr.inp	**fillEndL.tpr**	yes	
ParallelSide1.inp		yes	
ParallelSide2.inp		yes	
Side1a.inp		yes	
Side1b.inp		yes	
Side2a.inp		yes	

Model in ./DeadEnd	Restart file	Selected	Reason for non-selection
Side2b.inp		yes	
Side3a.inp		yes	
Side3b.inp		yes	
sides.inp		yes	
TenCellsTenAxial.inp		yes	
TwoDead.inp		yes	
TwoDead2.inp		yes	
TwoFill.inp		yes	
TwoFill2.inp		yes	

C.6 Suites ./FluidPower

Number of selected input files: 5
Input files modified for selection: 0

Number of non-selected input files: 0
Input files written in fixed-format: 0
Input files yielding no or faulty XTV: 0

Total number of input files: 5

Model in ./FluidPower	Restart file	Selected	Reason for non-selection
msvl_flpower_1.inp		yes	
msvl_flpower_2.inp		yes	
PbBi_flpower.inp		yes	
PbBi_pipe_flpower.inp		yes	
PbBi_pipe_flpower_rst.inp	**PbBi_pipe_flpower.tpr**	yes	

C.7 Suites ./GapHT

Number of selected input files: 85
Input files modified for selection: 1 (marked in table as 'yes*')

Number of non-selected input files: 13
Input files written in fixed-format: 0
Input files yielding no or faulty XTV: 13

Total number of input files: 98

Model in ./GapHT	Restart file	Selected	Reason for non-selection
3chan_nfciTest.inp		yes	
chan_nfciTest-13.Rev6.inp		yes	
chan_nfciTest-13.Rev6R.inp	**chan_nfciTest-13.Rev6.tpr**	yes	
chan_nfciTest-13.Rev8.FE.inp			no XTV file
chan_nfciTest-13.Rev9.FE.inp		yes	
chan_nfciTest-13.Rev9R.FE.inp	**chan_nfciTest-13.Rev9.FE.tpr**	yes	
chan_nfciTest-13.Rev9R0.FE.inp	**chan_nfciTest-13.Rev9.FE.tpr**	yes	
chan_nfciTest-13.Rev10.FE.inp		yes	
chan_nfciTest-13.Rev10.FE.Check.inp			no XTV file
chan_nfciTest-13.Rev10.FE.Check1.inp			no XTV file
chan_nfciTest-13.Rev10.FE.Check2.inp			no XTV file
chan_nfciTest-13.Rev10.FE.Check3.inp			no XTV file
chan_nfciTest-13.Rev10.FE.Check4.inp		yes	
chan_nfciTest-13.Rev10.FE.Check5.inp		yes	
chan_nfciTest-13.Rev10.FE.Check6.inp		yes	
chan_nfciTest-13.Rev10R.FE.inp	**chan_nfciTest-13.Rev10.FE.tpr**	yes	

Model in ./GapHT	Restart file	Selected	Reason for non-selection
chan_nfciTest-13.Rev10R0.FE.inp	chan_nfciTest-13.Rev10.FE.tpr	yes	
chan_nfciTest-13.Rev11.FE.inp		yes	
chan_nfciTest-13.Rev12.FE.inp		yes	
chan_nfciTest-13_Rev13_FE.inp		yes	
chan_nfciTest-13_Rev14_FE.inp		yes	
chantest.inp		yes	
HGAP.2.98-12.inp		yes	
HGAP.2.98.-12.inp		yes	
HGAP.2.98-12.FE.Rev1.inp		yes	
HGAP.2.98.1.inp		yes	
HGAP.2.98.2.inp		yes	
HGAP.2.98.82.inp		yes	
HGAP.4.9-12.inp		yes	
HGAP.4.9-12.FE.Rev1.inp		yes	
HGAP.4.9.1.inp		yes	
HGAP.4.9.2.inp		yes	
HGAP.4.9.82.inp		yes	
HGAP.9.9-12.inp		yes	
HGAP.9.9-12.FE.Rev1.inp		yes	
HGAP.9.9.1.inp		yes	
HGAP.9.9.2.inp		yes	

Model in ./GapHT	Restart file	Selected	Reason for non-selection
HGAP.9.9.82.inp		yes	
HGAP.12.4-12.inp		yes	
HGAP.12.4-12.FE.Rev1.inp		yes	
HGAP.12.4.1.inp		yes	
HGAP.12.4.2.inp		yes	
HGAP.12.4.82.inp		yes	
HGAP.14.9-12.inp		yes	
HGAP.14.9-12.FE.Rev1.inp		yes	
HGAP.14.9.1.inp		yes	
HGAP.14.9.2.inp		yes	
HGAP.14.9.82.inp		yes	
HGAP.16-12.inp		yes	
HGAP.16-12.FE.Rev1.inp		yes	
HGAP.16.1.inp		yes	
HGAP.16.2.inp		yes	
HGAP.16.82.inp		yes	
HGAP.18-12.inp		yes	
HGAP.18-12.FE.Rev1.inp		yes	
HGAP.18-12.FE.Rev2.inp		yes	
HGAP.18-12.FE.Rev3.inp		yes	
HGAP.18-12.FE.Rev4.inp		yes	

Model in ./GapHT	Restart file	Selected	Reason for non-selection
HGAP.18-12.FE.Rev4R0.inp	**HGAP.18-12.FE.Rev4.tpr**	yes	
HGAP.18-12.FE.Rev4R10.inp	**HGAP.18-12.FE.Rev4.tpr**	yes	
HGAP.18-12.FER.Rev1.inp	**HGAP.18-12.FE.Rev1.tpr**	yes	
HGAP.18-12.FER.Rev2.inp	**HGAP.18-12.FE.Rev1.tpr**	yes	
HGAP.18.2.inp		yes	
HGAP.18.82.inp		yes	
HGAP.check.inp		yes	
HGAP.check.Rev1.inp		yes	
HGAP.check.Rev2.inp		yes	
HGAP.check.Rev3.inp		yes	
HGAP.check.Rev4.inp		yes	
HGAP.check.Rev5.inp		yes	
HGAP.check.Rev6.inp			no XTV file
HGAP.check.Rev7.inp			no XTV file
HGAP.check.Rev8.inp			no XTV file
HGAP.check.Rev9.inp			no XTV file
HGAP.check.Rev10.inp			no XTV file
HGAP.check.Rev11.inp			no XTV file
HGAP.check.Rev12.inp			no XTV file
HGAP.check.Rev13.inp		yes	
HGAP.check.Rev14.inp		yes	

Model in ./GapHT	Restart file	Selected	Reason for non-selection
hot_assembly_rev9_n13.inp		yes	
hot_assembly_rev9_n13.OneHS.inp		yes	
hot_assembly_rev10_n1.inp		yes	
mwrxOn.inp		yes	
nfci2.inp			no XTV file
nfci3.inp		yes	
nfci73.inp		yes	
nfci73_5ax.inp		yes	
nfci73_5axRst.inp	**nfci73_5ax.tpr**	yes	
norupture.inp		yes	
rupture.inp		yes	
rupture2a.inp		yes	
vsltest.inp		yes	
vsltest2.inp		yes	
vsltest3.inp		yes	
vsltest3Rst.inp	**vsltest3.tpr**	yes	
vsltest4.inp		yes	
vsltest4Rst.inp	**vsltest4.tpr**	yes*	
w3loopLOCA.inp		yes	

C.8 Suites ./GenTbl

Number of selected input files: 14
Input files modified for selection: 0

Number of non-selected input files: 1
Input files written in fixed-format: 0
Input files yielding no or faulty XTV: 1

Total number of input files: 15

Model in ./GenTbl	Restart file	Selected	Reason for non-selection
EngMatFunc.inp		yes	
idbc3.inp		yes	
idbc3r.inp	**idbc3.tpr**	yes	
idbc3r2.inp	**idbc3.tpr**	yes	
idbc3t3.inp		yes	
idbc3t3r.inp	**idbc3t3.tpr**	yes	
idbc12.inp		yes	
idbc12r.inp	**idbc12.tpr**	yes	
idbcBackup.inp		yes	
idbcErr.inp		yes	
idbcErr2.inp			no XTV file
idbcUnits.inp		yes	
idbcUnits2.inp		yes	
mat51.inp		yes	
mat53.inp		yes	

C.9 Suites ./HtStr

Number of selected input files: 89
Input files modified for selection: 2 (marked in table as 'yes*')

Number of non-selected input files: 11
Input files written in fixed-format: 0
Input files yielding no or faulty XTV: 11

Total number of input files: 100

Model in ./HtStr	Restart file	Selected	Reason for non-selection
2sHS1.inp		yes	
30burn5gad.inp		yes	
BB_60_U2.inp		yes	
BB_60_U2.Rev1.inp		yes	
BB_60_U2.Rev2.inp		yes	
BB_60_U2.Rev3.inp		yes	
BB_60_U2.Rev4.inp		yes	
BB_60_U2.Rev5.inp		yes	
BB_60_U2.Rev6.inp		yes	
BB_60_U2.Rev7.inp		yes	
BB_60_U2.Rev8.inp		yes	
BB_60_U2.Rev9.inp		yes	
COTINCO_0deg.inp		yes	
COTINCO_0deg_Rev1.inp		yes	
COTINCO_0deg_Rev2.inp			no XTV file
COTINCO_0deg_Rev3.inp			no XTV file

Model in ./HtStr	Restart file	Selected	Reason for non-selection
COTINCO_0deg_Rev4.inp			no XTV file
COTINCO_0degR.inp	**COTINCO_0deg.tpr**	yes	
COTINCO_5deg.inp		yes	
COTINCO_15deg.inp		yes	
COTINCO_30deg.inp		yes	
COTINCO_45deg.inp		yes	
HSBCTest.inp		yes	
HSgas1.inp		yes	
HSiafb.inp		yes	
HSliq1.inp		yes	
HSnb.inp		yes	
HSpCHF.inp		yes	
HS_test_export2.inp		yes	
HS_test_export2r.inp	**HS_test_export2.tpr**	yes	
htstrold.inp		yes	
htstroldname.inp		yes	
htstrsuccessgadmax.inp		yes	
htstrsuccess.inp		yes	
lumpedCyl.inp		yes	
m1d.inp		yes	
multiHydro.inp		yes	

Model in ./HtStr	Restart file	Selected	Reason for non-selection
PipeType_Error_Check.inp			no XTV file
PoolBoiling.inp		yes	
SlabHS1Pipe.inp		yes	
TableBC.inp		yes	
testin.1.inp		yes	
testin.1n.inp		yes	
testin.2.inp		yes	
testin.2n.inp		yes	
testin.5.inp		yes	
testin.5n.inp		yes	
testin.6.inp		yes	
testin.6n.inp		yes	
testin.7.inp		yes	
testin.7n.inp		yes	
testin.7r.inp	**testin.7.tpr**	yes*	
testin.7rn.inp	**testin.7n.tpr**	yes*	
THI_Looppt1.inp		yes	
THI_Looprst.inp	**THI_Looppt1.tpr**	yes	
THI_Looptot.inp			
TwoSidedHS.inp		yes	
vess_and_chan.inp			no XTV file

Model in ./HtStr	Restart file	Selected	Reason for non-selection
vess_and_chan_SS.inp		yes	
vess_and_chan_TR.inp	**vess_and_chan_SS.tpr**		no XTV file
vess_and_chan_TR_wk.inp	**vess_and_chan_SS.tpr**	yes	
vess_and_chan_wk.inp		yes	
vess_and_rod_SS.inp		yes	
vess_and_rod_TR.inp	**vess_and_rod_SS.tpr**		no XTV file
W4loopHotRod.inp		yes	
W4loopHotRodR.inp	**W4loopHotRod.tpr**	yes	
W4loopHotRodNoRep.inp		yes	
W4loopHotRod.NoRep.inp		yes	
W4loopHotRod.NoRep.Mod1.inp		yes	
W4loopHotRod.NoRep.Mod2.inp		yes	
W4loopHotRod.NoRep.Mod3.inp		yes	
W4loopHotRod.NoRep.Mod4.inp		yes	
W4loopHotRod.NoRep.Mod5.inp		yes	
W4loopHotRod.NoRep.Mod6.inp		yes	
W4loopHotRodR.NoRep.inp	**W4loopHotRod.NoRep.tpr**	yes	
W4loopHotRodR.NoRep.Mod1.inp	**W4loopHotRod.NoRep.Mod1.tpr**	yes	
W4loopHotRodR.NoRep.Mod2.inp	**W4loopHotRod.NoRep.Mod2.tpr**	yes	
W4loopHotRodR.NoRep.Mod3.inp	**W4loopHotRod.NoRep.Mod3.tpr**	yes	
W4loopHotRodR.NoRep.Mod4.inp	**W4loopHotRod.NoRep.Mod4.tpr**	yes	

Model in ./HtStr	Restart file	Selected	Reason for non-selection
W4loopHotRodR.NoRep.Mod5.inp	W4loopHotRod.NoRep.Mod5.tpr	yes	
W4loopHotRodR.NoRep.Mod6.inp	W4loopHotRod.NoRep.Mod6.tpr	yes	
W4loopHotRod.RadialBurn0.inp		yes	
W4loopHotRod.RadialBurn1.inp		yes	
W4loopHotRod.RadialBurn1.InpErr1.inp			no XTV file
W4loopHotRod.RadialBurn2.inp		yes	
W4loopHotRod.RadialBurn2.InpErr2.inp			no XTV file
W4loopHotRod.RadialBurn3.inp		yes	
W4loopHotRod.RadialBurn3.InpErr3.inp			no XTV file
W4loopHotRod.RadialBurn3.Rst0.inp	W4loopHotRod.RadialBurn3.tpr	yes	
W4loopHotRod.RadialBurn3.Rst1.inp	W4loopHotRod.RadialBurn3.Rst0.tpr	yes	
W4loopHotRod.RadialBurn3.Rst2.inp	W4loopHotRod.RadialBurn3.Rst1.tpr	yes	
W4loopHotRod.RadBurn4.inp		yes	
W4loopHotRod.RadBurn4.Rst0.inp	W4loopHotRod.RadBurn4.tpr	yes	
W4loopHotRod.RadBurn4.Rst1.inp	W4loopHotRod.RadBurn4.Rst0.tpr	yes	
W4loopHotRod.RadBurn4.Rst2.inp	W4loopHotRod.RadBurn4.Rst1.tpr	yes	
W4loop.NoHotRod.inp		yes	
W4loopNoHotRod.inp		yes	
W4loopR.NoHotRod.inp	W4loop.NoHotRod.tpr	yes	
ZeroHSVol.inp			no XTV file
ZeroHSVol1.inp		yes	

C.10 Suites ./Kinetics

Number of selected input files: 11
Input files modified for selection: 0

Number of non-selected input files: 0
Input files written in fixed-format: 0
Input files yielding no or faulty XTV: 0

Total number of input files: 11

Model in ./Kinetics	Restart file	Selected	Reason for non-selection
DecHeatMult10.inp		yes	
DecHeatMult15.inp		yes	
DHMultSS1.inp		yes	
DHMultSS12.inp		yes	
DHMultTR1.inp	**DHMultSS1.tpr**	yes	
DHMultTR12.inp	**DHMultSS12.tpr**	yes	
pk_ans94_dh.inp		yes	
pk_base.inp		yes	
pk_sep_fdbk.inp		yes	
pk_tbl3_fdbk.inp		yes	
pk_tbl3a_fdbk.inp		yes	

C.11 Suites ./NCGases

Number of selected input files: 47
Input files modified for selection: 0

Number of non-selected input files: 8
Input files written in fixed-format: 0
Input files yielding no or faulty XTV: 8

Total number of input files: 55

Model in ./NCGases	Restart file	Selected	Reason for non-selection
in2phair.inp		yes	
in2phmix.inp		yes	
in2phncg1.inp		yes	
in2phncg2.inp		yes	
in2phncg3.inp		yes	
in2phncg4.inp		yes	
in2phsteam.inp		yes	
inair.inp		yes	
inerr1.inp			no XTV file
inerr2.inp			no XTV file
inerr3.inp			no XTV file
inerr4.inp		yes	
inerr5.inp			no XTV file
inerr6.inp			no XTV file
inerr7.inp			no XTV file
inerr8.inp			no XTV file

Model in ./NCGases	Restart file	Selected	Reason for non-selection
inigas1allair.inp		yes	
inigas1allsteam.inp		yes	
inigas1.inp		yes	
inigas2.inp		yes	
inigas3.inp		yes	
inigas4.inp		yes	
inigas5.inp		yes	
inigas6.inp		yes	
inigas7.inp		yes	
inigas8.inp			no XTV file
inloopair.inp		yes	
inloopallair.inp		yes	
inloopallncg.inp		yes	
inloopallsteam.inp		yes	
inloopar.inp		yes	
inlooph2.inp		yes	
inloophe.inp		yes	
inloopioflow.inp		yes	
inloopkr.inp		yes	
inloopn2.inp		yes	
inloopxe.inp		yes	

Model in ./NCGases	Restart file	Selected	Reason for non-selection
inmix.inp		yes	
inncg1.inp		yes	
inncg2.inp		yes	
inncg3.inp		yes	
inncg4.inp		yes	
inNCGTestA.inp		yes	
inNCGTestB.inp		yes	
inNCGTestC.inp		yes	
inNCGTestD.inp		yes	
inNCGTestE.inp		yes	
inNCGTestF.inp		yes	
inNCGTestG.inp		yes	
inss7.inp		yes	
insteam.inp		yes	
intr1.inp	**inss7.tpr**	yes	
invssl16.inp		yes	
NatCirc.inp		yes	
NCGflow.inp		yes	

C.12 Suites ./OffTake

Number of selected input files: 35
Input files modified for selection: 0

Number of non-selected input files: 2
Input files written in fixed-format: 0
Input files yielding no or faulty XTV: 2

Total number of input files: 37

Model in ./Offtake	Restart file	Selected	Reason for non-selection
ColdLegBreak.inp		yes	
ColdLegBreakPIPE.inp		yes	
ColdLegBreakPIPEDown.inp		yes	
ColdLegBreakPIPEDown.0.999.inp		yes	
ColdLegBreakPIPEDown.0.9999.inp		yes	
ColdLegBreakPIPEUp.inp		yes	
ColdLegBreakRst.inp	**ColdLegBreak.tpr**	yes	
ColdLegBreakSJC.inp		yes	
ColdLegBreakSJC.Rev1.inp		yes	
ColdLegBreakSJC.Rev2.inp		yes	
ColdLegBreakSJC.Rev3.inp		yes	
ColdLegBreakSJC.Rev4.inp		yes	
ColdLegBreakSJC.0.01.inp		yes	
ColdLegBreakSJC.0.1.inp		yes	
ColdLegBreakSJC.0.9.inp		yes	
ColdLegBreakSJC.0.99.inp		yes	

Model in ./Offtake	Restart file	Selected	Reason for non-selection
ColdLegBreakSJC.0.999.inp		yes	
ColdLegBreakSJCDown.inp		yes	
ColdLegBreakSJCUp.inp		yes	
ColdLegBreakTee.inp		yes	
ColdLegBreakTeeDown.inp		yes	
ColdLegBreakTeeUp.inp		yes	
offset_up_rev03.inp		yes	
offset_up_rev04.inp			no XTV file
offset_up_rev05.inp			no XTV file
offset_up_rev06.inp		yes	
offset_up_rev06R.inp	**offset_up_rev06.tpr**	yes	
offtakeA.inp		yes	
offtakeA10.inp		yes	
offtakeB.inp		yes	
offtakeB10.inp		yes	
tindwn.inp		yes	
tindwnSide.inp		yes	
tindwnSideReversed.inp		yes	
tinup.inp		yes	
tinupSide.inp		yes	
tinupSideReversed.inp		yes	

C.13 Suites ./PARCS

Number of selected input files: 13
Input files modified for selection: 0

Number of non-selected input files: 0
Input files written in fixed-format: 0
Input files yielding no or faulty XTV: 0

Total number of input files: 13

Model in ./PARCS	Restart file	Selected	Reason for non-selection
CartVess.inp		yes	
DirectCoolantHeating.inp		yes	
mslb3_cartvess_ss.inp		yes	
mslb3_cartvess_tr.inp	**(TRACE) mslb3_cartvess_ss.tpr** **(PARCS) mslb3_cartvess_ss.parcs_rst**	yes	
mslb_cart_suprod_ss.inp		yes	
mslb_trace_htrd_ss.inp		yes	
mslb_trace_ss.inp		yes	
mslb_trace_tr.inp	**(TRACE) mslb_trace_ss.tpr** **(PARCS) mslb_trace_ss.parcs_rst**	yes	
pbtt_trace_sa.inp		yes	
pbtt_trace_ss.inp		yes	
pbtt_trace_tr.inp	**(TRACE) pbtt_trace_ss.tpr** **(PARCS) pbtt_trace_ss.parcs_rst**	yes	
pbtt_trace_ws.inp	**(TRACE) pbtt_trace_ss.tpr** **(PARCS) pbtt_trace_ss.parcs_rst**	yes	
VesselChan.inp		yes	

C.14 Suites ./PlantsShort

Number of selected input files: 4
Input files modified for selection: 0

Number of non-selected input files: 1
Input files written in fixed-format: 1 (no Namelist option)
Input files yielding no or faulty XTV: 0

Total number of input files: 5

Model in ./PlantsShort	Restart file	Selected	Reason for non-selection
mslb_mod8a.inp		yes	
mslb_mod8ar.inp	**mslb_mod8a.tpr**	yes	
PeachBottom.inp			fixed-format input
w3loopss.inp		yes	
w3looptr.inp	**w3loopss.tpr**	yes	

C.15 Suites ./Power

Number of selected input files: 119
Input files modified for selection: 0

Number of non-selected input files: 2
Input files written in fixed-format: 2 (no Namelist option)
Input files yielding no or faulty XTV: 0

Total number of input files: 121

Model in ./Power	Restart file	Selected	Reason for non-selection
1group-irpwty3-M.inp		yes	
1group-irpwty13-M.inp		yes	
3Chan.Mod2.inp		yes	
3Chan.Mod3.inp		yes	
3Chan.Mod4.inp		yes	
3Chan.Mod5.inp		yes	
3Chan.Mod6.inp		yes	
3Chans.FE-1.inp		yes	
3Chans.FE-2.inp		yes	
3Chans.FE-3.inp		yes	
3Chans.FE-4.inp		yes	
3Chans.FE-5.inp		yes	
3Chans.FE-6.inp		yes	
3Chans.FE-7.inp		yes	
3Chans.FE-8.inp		yes	
3Chans.FE-9.inp		yes	

Model in ./Power	Restart file	Selected	Reason for non-selection
3Chans.inp		yes	
3Chans.Rev1.inp		yes	
3Chans.Rev2.inp		yes	
AxialPowerCheck1.inp		yes	
AxialPowerCheck2.inp		yes	
case1.inp		yes	
case2.inp		yes	
case3.inp		yes	
case3-1.inp		yes	
case3-2.inp		yes	
case3-3.inp		yes	
case3-4.inp		yes	
case3-4R.inp	case3-4.tpr	yes	
case3-4R1.inp	case3-4.tpr	yes	
case3-5.inp		yes	
case3-6.inp		yes	
case3-6R.inp	case3-6.tpr	yes	
case3-7.inp		yes	
case3-8.inp		yes	
case3-9.inp		yes	
case3-10.inp		yes	

Model in ./Power	Restart file	Selected	Reason for non-selection
case3-10R_16.inp	**case3-10.tpr**	yes	
case3-10R_22.inp	**case3-10.tpr**	yes	
case3-10R_30.inp	**case3-10.tpr**	yes	
case4.inp		yes	
case4R.inp	**case4.tpr**	yes	
case5.inp		yes	
case5R.inp	**case5.tpr**	yes	
Chan-B-irpop7-irod.inp			fixed-format input
chan-pow11jn.inp		yes	
chan-pow11jn.Rev1.inp		yes	
chan-pow11jnr.inp	**chan-pow11jn.tpr**	yes	
Chan-Pow11XC.inp		yes	
Chan-Pow11XCR.inp	**Chan-Pow11XC.tpr**	yes	
chanpow.inp		yes	
chanpow_reordered.inp		yes	
DiffRodsAxialSS.inp		yes	
DiffRodsAxialTR.inp	**DiffRodsAxialSS.tpr**	yes	
DiffRodsMatSS.inp		yes	
DiffRodsMatTR.inp	**DiffRodsMatSS.tpr**	yes	
DiffRodsRadialSS.inp		yes	
DiffRodsRadialTR.inp	**DiffRodsRadialSS.tpr**	yes	

Model in ./Power	Restart file	Selected	Reason for non-selection
DiffRodsSS.inp		yes	
ipowrTest1.inp		yes	
ipowrTest2.inp		yes	
ipowrTest3.inp		yes	
ipowrTest4.inp		yes	
ipowrTest5.inp		yes	
ipowrTest6.inp		yes	
ipowrTest7.inp		yes	
ipowrTest8.inp		yes	
ipowrTest9.inp		yes	
ipowrTest10.inp		yes	
ipowrTest11.inp		yes	
ipowrTest12.inp		yes	
ipowrTest13.inp		yes	
ipowrTest13r_a.inp	ipowrTest13.tpr	yes	
ipowrTest13r_b.inp	ipowrTest13.tpr	yes	
ipowrTest14.inp		yes	
ipowrTest15.inp		yes	
ipowrTest15r_a.inp	ipowrTest15.tpr	yes	
ipowrTest15r_b.inp	ipowrTest15.tpr	yes	
ipowrTest16.inp		yes	

Model in ./Power	Restart file	Selected	Reason for non-selection
ipowrTest17.inp		yes	
ipowrTest17r_a.inp	ipowrTest17.tpr	yes	
ipowrTest17r_b.inp	ipowrTest17.tpr	yes	
ipowrTest18.inp		yes	
ipowrTest19.inp		yes	
ipowrTest19r_a.inp	ipowrTest19.tpr	yes	
ipowrTest20.inp		yes	
ipowrTest21.inp		yes	
ipowrTest22.inp		yes	
ipowrTest23.inp		yes	
ipowrTest24.inp		yes	
lrpwty7-3ChanXC.inp		yes	
multi_pow.inp		yes	
multi_powr.inp	multi_pow.tpr	yes	
OnePower.inp		yes	
pipe.PowTest.inp		yes	
pipePowTest.inp		yes	
PKPowSource1.inp		yes	
PKPowSource2.inp		yes	
powerss.inp		yes	
powertr.inp	powerss.tpr	yes	

Model in ./Power	Restart file	Selected	Reason for non-selection
PrizerPower.inp		yes	
PromHeat0.025.inp		yes	
PromHeat0.05.inp		yes	
PromHeat0.05R.inp	PromHeat0.05.tpr	yes	
reactivitytest_pk_ans94.inp		yes	
rod2-M-irpwty13.inp		yes	
RPOWRI-Zero.inp		yes	
RSTtest1.inp	SStest1.tpr	yes	
SameRodsSS.inp		yes	
SameRodsTR.inp	SameRodsSS.tpr	yes	
SCTF604.inp		yes	
SCTF604R.inp	SCTF604.tpr	yes	
SpherePower.inp		yes	
SpherePowerRst.inp	SpherePower.tpr	yes	
SStest1.inp		yes	
test_spower.inp		yes	
TwoPower.inp		yes	
Vess-Chan-B-irpop7.inp			fixed-format input
Vess-Chan-M-irpwty3.inp		yes	
Vess-Chan-M-irpwty13.inp		yes	
Vess-Chan-M-irpwty14.inp		yes	

C.16 Suites ./Radiation

Number of selected input files: 26
Input files modified for selection: 0

Number of non-selected input files: 2
Input files written in fixed-format: 2 (no Namelist option)
Input files yielding no or faulty XTV: 0

Total number of input files: 28

Model in ./Radiation	Restart file	Selected	Reason for non-selection
30910i-B-60rod1-ibeam.inp			fixed-format input
30910i-B-60rod1-irod.inp			fixed-format input
30910i-M-ibeam.inp		yes	
30910i-M-mrod.inp		yes	
chan-pow11.inp		yes	
chan-pow11-rst.inp	**chan-pow11.tpr**	yes	
GotaM.inp		yes	
hcond1_ibeam.inp		yes	
hcond1_ibeam-rst.inp	**hcond1_ibeam.tpr**	yes	
hcond1pow.inp		yes	
hcond1pow-rst.inp	**hcond1pow.tpr**	yes	
irpwty1-1chan.inp		yes	
irpwty11-1chan.inp		yes	
irpwty11-1chan-rst.inp	**irpwty11-1chan.tpr**	yes	
irpwty7.inp		yes	
irpwty7-direct.inp		yes	

Model in ./Radiation	Restart file	Selected	Reason for non-selection
irpwty7-3chan.inp		yes	
RadEncCylinder.inp		yes	
RadEncSlab.inp		yes	
RadEncSphere.inp		yes	
radiationFn.inp		yes	
radiationIn.inp		yes	
tracinG.inp		yes	
tracinH.inp		yes	
W4loop.inp		yes	
W4loop-rst.inp	**W4loop.tpr**	yes	
W4loop-pow.inp		yes	
W4loop-pow-rst.inp	**W4loop-pow.tpr**	yes	

C.17 Suites ./Sepd

Number of selected input files: 12
Input files modified for selection: 0

Number of non-selected input files: 9
Input files written in fixed-format: 9 (no Namelist option)
Input files yielding no or faulty XTV: 0

Total number of input files: 21

Model in ./Sepd	Restart file	Selected	Reason for non-selection
HFlow.inp			fixed-format input
MixSeparators.inp		yes	
Sepd.inp			fixed-format input
SepdInputErrorCheck1.inp			fixed-format input
SepdInputErrorCheck2.inp			fixed-format input
Sepd_istage1.inp		yes	
Sepd_istage-3.inp		yes	
Sepd_istage-3.Rst.inp	**Sepd_istage-3.tpr**	yes	
SepdSemi.inp			fixed-format input
SepdSemiRst.inp	**SepdSemi.tpr**		fixed-format input
SepdSETS.inp			fixed-format input
SepdSETSM.inp		yes	
SepdSETSM.Rev1.inp		yes	
SepdSETSM.Rev2.inp		yes	
SepdSETSRst.inp	**SepdSETS.tpr**		fixed-format input
SepdTee.inp			fixed-format input

Model in ./Sepd	Restart file	Selected	Reason for non-selection
Sepd_Test0.inp		yes	
Sepd_Test1.inp		yes	
Sepd_Test2.inp		yes	
Sepd_Test3.inp		yes	
SVolSepdSETSM.inp		yes	

C.18 Suites ./Short

Number of selected input files: 113
Input files modified for selection: 2 (marked in table as 'yes*')

Number of non-selected input files: 0
Input files written in fixed-format: 0
Input files yielding no or faulty XTV: 0

Total number of input files: 113

Model in ./Short	Restart file	Selected	Reason for non-selection
AdjP.inp		yes	
bingham_pump.inp		yes	
boron.inp		yes	
boron1.inp		yes	
boron2.inp		yes	
boronOSPRE.inp		yes	
boronOSPRE.isolut2.inp		yes	
boronre.inp	**boron.tpr**	yes*	
boronSemi.inp		yes	
boronSemi.isolut2.inp		yes	
boronSETS.inp		yes	
boronSETS.isolut2.inp		yes	
BreakEnthalpy.inp		yes	
CCTF_ColdLeg.inp		yes	
CulvertTest.inp		yes	
Dalp.inp		yes	

Model in ./Short	Restart file	Selected	Reason for non-selection
Dpg.inp		yes	
drain.inp		yes	
ElbowLong1.inp		yes	
ElbowLong2.inp		yes	
ElbowLong3.inp		yes	
FillBCv.inp		yes	
fxtsd.inp		yes	
hcond1.inp		yes	
hcond2.inp		yes	
hcond3.inp		yes	
hcond4.inp		yes	
hcond5.inp		yes	
HomMultAWDdefault.inp		yes	
HomMultAWD.inp		yes	
HomMultAWDre.inp	HomMultAWD.tpr	yes	
LargeNum.inp		yes	
marvik.inp		yes	
mscl.inp		yes	
mscv.inp		yes	
msc2.inp		yes	
msp2.inp		yes	

Model in ./Short	Restart file	Selected	Reason for non-selection
mspl.inp		yes	
mspv.inp		yes	
msrl.inp		yes	
msrv.inp		yes	
msr2.inp		yes	
mss2.inp		yes	
mssl.inp		yes	
mssv.inp		yes	
mssv_neg_timet.inp		yes	
mstl.inp		yes	
mstv.inp		yes	
mst2.inp		yes	
msv2.inp		yes	
msvl.inp		yes	
msvlr.inp	msvl.tpr	yes	
msvv.inp		yes	
OneLevelFourCell.inp		yes	
OneLevelTwoCell.inp		yes	
pipeHF.inp		yes	
Pipelelv0.inp		yes	
Pipelelv1.inp		yes	

Model in ./Short	Restart file	Selected	Reason for non-selection
Pipelelv2.inp		yes	
pressurizer_pdrop1.inp		yes	
pressurizer_pdrop2.inp		yes	
pressurizer_pinc1.inp		yes	
pressurizer_pinc2.inp		yes	
pressurizer_pinc3.inp		yes	
pressurizer_pinc4.inp		yes	
PumpBCmf.inp		yes	
PumpBCp.inp		yes	
PumpBCpR.inp	**PumpBCp.tpr**	yes	
PumpBCv.inp		yes	
PumpLim.inp		yes	
PumpTyp8.inp		yes	
PumpTyp8.Rev1.inp		yes	
PumpTyp8.Rev1R.inp	**PumpTyp8.Rev1.tpr**	yes*	
PumpTyp9.inp		yes	
PumpTyp9.Rev1.inp		yes	
PumpTyp9.Rev1R.inp	**PumpTyp9.Rev1.tpr**	yes	
PumpTyp10.inp		yes	
PumpTyp10.Rev1.inp		yes	
PumpTyp10.Rev1R.inp	**PumpTyp10.Rev1.tpr**	yes	

Model in ./Short	Restart file	Selected	Reason for non-selection
PumpTyp11.inp		yes	
PumpTyp11.Rev1.inp		yes	
PumpTyp11.Rev1R.inp	PumpTyp11.Rev1.tpr	yes	
radial_vessel_3rings1.inp		yes	
rod2.inp		yes	
semimatrix.inp		yes	
SimpleVentTest2.inp		yes	
SingleCellPump.inp		yes	
SingleCellPump1.inp		yes	
SingleCellPump2.inp		yes	
Styrikovich1.inp		yes	
Styrikovich2.inp		yes	
Styrikovich3.inp		yes	
Styrikovich4.inp		yes	
tfpipe2.inp		yes	
ThreeLevelFourCell.inp		yes	
ThreeLevelOneCell.inp		yes	
TwoLevelFourCell.inp		yes	
TwoPhase_Down.inp		yes	
TwoPhaseHeat.inp		yes	
TwoPhaseNoHeat.inp		yes	

Model in ./Short	Restart file	Selected	Reason for non-selection
TwoPhase_Up.inp		yes	
utube.inp		yes	
Vess3Dtester.inp		yes	
w4loopn.inp		yes	
w4looprn.300.inp	w4loopn.tpr	yes	
w4looprn.inp	w4loopn.tpr	yes	
w4loop.PIPE.inp		yes	
w4loop.PIPE1.inp		yes	
w4lpr1.PIPE.inp	w4loop.PIPE.tpr	yes	
w4lpr1.PIPE1.inp	w4loop.PIPE1.tpr	yes	
w4VesselFl.inp		yes	
w4Vessel.inp		yes	
westinghouse_pump.inp		yes	

C.19 Suites ./Short2

Number of selected input files: 113
Input files modified for selection: 1 (marked in table as 'yes*')

Number of non-selected input files: 9
Input files written in fixed-format: 2 (no Namelist option)
Input files yielding no or faulty XTV: 7

Total number of input files: 122

Model in ./Short2	Restart file	Selected	Reason for non-selection
2hscase.1.old.inp		yes	
accumTest.inp		yes*	
brksat.inp		yes	
brksatn.inp		yes	
dtdiag.1.inp		yes	
dtdiag.2.inp			no XTV file
edwards.inp		yes	
fill1.inp		yes	
fill2.inp		yes	
Fill2Steam.inp		yes	
fill3.inp		yes	
fill4.inp		yes	
fill5.inp		yes	
fill6.inp		yes	
fill7.inp		yes	
fill8.inp		yes	

Model in ./Short2	Restart file	Selected	Reason for non-selection
fill9.inp		yes	
fill9_r.inp	fill9.tpr	yes	
FILLP.inp		yes	
FILLPSideJ.inp		yes	
genbrk.inp		yes	
genbrkn.inp		yes	
LargeInput.inp		yes	
loft_ie.inp		yes	
loftk.inp		yes	
loftkr.inp	loftk.tpr	yes	
marv4.inp		yes	
marv13.inp		yes	
marv20.inp		yes	
marv22.inp		yes	
marv24.inp		yes	
MIT-ST4-40Nodes.inp		yes	
MIT-ST4-40Nodes.Ves.inp		yes	
MIT-ST4-Part4.inp		yes	
MIT-ST4-Part4.Ves.inp		yes	
ois1a.inp		yes	
ois1an.inp		yes	

Model in ./Short2	Restart file	Selected	Reason for non-selection
oish1-new.inp		yes	
oish1-newn.inp		yes	
ois-hs-v_2t.inp		yes	
ois-hs-v_2tn.inp		yes	
ois-hs-v.inp		yes	
ois-hs-vn.inp		yes	
pipeRupture1.inp		yes	
pipeRupture2.inp		yes	
pipeRupture3.inp		yes	
pipeRupture4.inp		yes	
pipeRupture5.inp		yes	
pipeRupture6.inp		yes	
pipeRupture7.inp		yes	
pipeRupture8.inp		yes	
pipeRupture9.inp		yes	
pipeRupture10.inp		yes	
PumpFlowLoss.inp		yes	
PumpFlowLoss.Mod1.inp		yes	
PumpFlowLoss.Mod2.inp		yes	
PumpFlowLoss.Mod3.inp		yes	
PumpFlowLoss.Mod4.inp		yes	

Model in ./Short2	Restart file	Selected	Reason for non-selection
PumpFlowLoss.Mod5.inp		yes	
PumpFlowLoss.Mod6.inp		yes	
PumpFlowLoss.Mod7.inp		yes	
PumpFlowLoss.Mod8.inp		yes	
PumpFlowLoss.Mod9.inp		yes	
PumpFlowLoss.Mod10.inp		yes	
PumpFlowLossR.Mod4.inp	**PumpFlowLoss.Mod4.tpr**	yes	
rvssl.inp		yes	
rvssl.2.inp		yes	
rvssl.3.inp		yes	
slabHtStr.inp		yes	
slabHtStrn.inp		yes	
TransiStdystTest1.inp		yes	
TransiStdystTest2.inp			no XTV file
TransiStdystTest3.inp			fixed-format input
TransiStdystTest4.inp			fixed-format input
updmfc1.inp		yes	
updmfc1n.inp		yes	
updmfc2.inp		yes	
updmfc2n.inp		yes	
updmfc4.inp		yes	

Model in ./Short2	Restart file	Selected	Reason for non-selection
updmfc4n.inp		yes	
updmfc5.inp		yes	
updmfc5n.inp		yes	
updmfc6.inp		yes	
updmfc6n.inp		yes	
Vess-Chan-AdjAddLoss.inp		yes	
Vess-Chan-AdjAddLoss.Mod1.inp		yes	
Vess-Chan-AdjAddLoss.Mod2.inp		yes	
Vess-Chan-AdjAddLossR.inp	Vess-Chan-AdjAddLoss.tpr	yes	
Vess-Chan-AdjAddLossR2err.inp	Vess-Chan-AdjAddLossR.tpr	yes	
Vess-Chan-AdjAddLossR2.inp	Vess-Chan-AdjAddLossR.tpr	yes	
Vess-Chan-AdjAddLossR3.inp	Vess-Chan-AdjAddLoss.tpr	yes	
Vess-Chan-AdjAddLossR4.inp	Vess-Chan-AdjAddLoss.tpr	yes	
vessel.1.inp			no XTV file
vessel.2.inp			no XTV file
vessel.3.inp			no XTV file
vessel.4.inp			no XTV file
vessel.5.inp			no XTV file
W4loop.inp		yes	
w4loopi.Mod0.inp		yes	
w4loopi.Mod1.inp		yes	

Model in ./Short2	Restart file	Selected	Reason for non-selection
w4loopi.Mod2.inp		yes	
w4loopi.Mod3.inp		yes	
w4loopi.Mod4.inp		yes	
w4loopi.Mod5.inp		yes	
w4loopi.Mod6.inp		yes	
w4loopi.Mod7.inp		yes	
w4looprCSS.Mod6.inp	w4loopi.Mod6.tpr	yes	
w4looprCSS.Mod7.inp	w4loopi.Mod6.tpr	yes	
w4loopri.Mod0.inp	w4loopi.Mod0.tpr	yes	
w4loopri.Mod1.inp	w4loopi.Mod1.tpr	yes	
w4loopri.Mod2.inp	w4loopi.Mod2.tpr	yes	
w4loopri.Mod3.inp	w4loopi.Mod3.tpr	yes	
w4loopri.Mod4.inp	w4loopi.Mod4.tpr	yes	
w4loopri.Mod5.inp	w4loopi.Mod5.tpr	yes	
w4loopri.Mod6.inp	w4loopi.Mod6.tpr	yes	
w4lpr1.inp	W4loop.tpr	yes	
w4lpr2.inp	W4loop.tpr	yes	
w4lpr3.inp	W4loop.tpr	yes	
w4lpr4.inp	W4loop.tpr	yes	
w4lpr5.inp	W4loop.tpr	yes	
w4lpr6.inp	W4loop.tpr	yes	

Model in ./Short2	Restart file	Selected	Reason for non-selection
w4lpr7.inp	**W4loop.tpr**	yes	

C.20 Suites ./SideJun

Number of selected input files: 49
Input files modified for selection: 0

Number of non-selected input files: 0
Input files written in fixed-format: 0
Input files yielding no or faulty XTV: 0

Total number of input files: 49

Model in ./SideJun	Restart file	Selected	Reason for non-selection
2pipes.inp		yes	
boron-FILL.inp		yes	
boron-FILL.isolut2.inp		yes	
ex0grav.inp		yes	
ex2Agrav.inp		yes	
ex2Bgrav.inp		yes	
ex2Cgrav.inp		yes	
ex2Dgrav.inp		yes	
ex2Egrav.inp		yes	
ex2Fgrav.inp		yes	
ex2Ggrav.inp		yes	
ex2Hgrav.inp		yes	
ex2Igrav.inp		yes	
ex2Jgrav.inp		yes	
ex2Kgrav.inp		yes	
ex2Lgrav.inp		yes	

Model in ./SideJun	Restart file	Selected	Reason for non-selection
ex2Mgrav.inp		yes	
ex2Ngrav.inp		yes	
ex2Ograv.inp		yes	
ex2Pgrav.inp		yes	
ex2Qgrav.inp		yes	
l-2pip-a.inp		yes	
l-2pip-a1r.inp		yes	
l-2pip-a2r.inp		yes	
l-tee-a.inp		yes	
PP-0.inp		yes	
PP-45.inp		yes	
PP-90-Eng.inp		yes	
PP-90.inp		yes	
PP-135.inp		yes	
PP-180.inp		yes	
T-0.inp		yes	
T-45.inp		yes	
T-90.inp		yes	
T-135.inp		yes	
T-180.inp		yes	
t-pLiq00a.inp		yes	

Model in ./SideJun	Restart file	Selected	Reason for non-selection
t-pLiq00b.inp		yes	
t-pLiq00c.inp		yes	
t-pLiq90a.inp		yes	
t-pLiq90b.inp		yes	
t-pLiq90c.inp		yes	
t-pLiq180a.inp		yes	
t-pLiq180b.inp		yes	
t-pLiq180c.inp		yes	
v-2pip-a.inp		yes	
v-2pip-a1r.inp		yes	
v-2pip-a2r.inp		yes	
v-tee-a.inp		yes	

C.21 Suites ./SJC

Number of selected input files: 62
Input files modified for selection: 0

Number of non-selected input files: 1
Input files written in fixed-format: 0
Input files yielding no or faulty XTV: 1

Total number of input files: 63

Model in ./SJC	Restart file	Selected	Reason for non-selection
1LegPT.inp		yes	
1LegTee.inp		yes	
2LegTee.inp		yes	
ACRSingleCanChanNoPwrRupt.inp		yes	
BentPipe1.inp		yes	
p2pSide.inp		yes	
p2pSideR.inp		yes	
p2vSide.inp		yes	
PbBi_H2O_withSJC.inp		yes	
sideBrkg.inp		yes	
sideBrkl.inp		yes	
SingleValveConst.inp		yes	
SJCconst.inp		yes	
t2pSide.inp		yes	
t2pSideR.inp		yes	
t2vSide.inp		yes	

Model in ./SJC	Restart file	Selected	Reason for non-selection
teeBrkg.inp		yes	
teeBrkl.inp		yes	
Test-SJC-Elev.inp			no XTV file
Test-SJC-Elev.Rev1.inp		yes	
Test-SJC-GRAV.inp		yes	
Test-SJC-GRAV.Rev1.inp		yes	
tinm1p1-b.inp		yes	
tinm1p1-s.inp		yes	
tinm1p2-b.inp		yes	
tinm1p2-s.inp		yes	
tinm1p3-b.inp		yes	
tinm1p3-s.inp		yes	
tinm2p1-b.inp		yes	
tinm2p1-b.Mod1.inp		yes	
tinm2p1-s.inp		yes	
tinm2p2-b.inp		yes	
tinm2p2-s.inp		yes	
tinm2p3-b.inp		yes	
tinm2p3-s.inp		yes	
tinm2p4-b.inp		yes	
tinm2p4-s.inp		yes	

Model in ./SJC	Restart file	Selected	Reason for non-selection
tinm2p5-b.inp		yes	
tinm2p5-s.inp		yes	
tinm2p6-b.inp	**tinm2p1-b.tpr**	yes	
tinm2p6-s.inp	**tinm2p1-s.tpr**	yes	
tinm2p7-b.inp		yes	
tinm2p7-s.inp		yes	
tinm3p1-b.inp		yes	
tinm3p1-s.inp		yes	
tinm4p1-b.inp		yes	
tinm4p1-s.inp		yes	
tinm4p2-b.inp		yes	
tinm4p2-s.inp		yes	
tinm5p1-b.inp		yes	
tinm5p1-s.inp		yes	
tinm6p1-b.inp		yes	
tinm6p1-s.inp		yes	
tinm6p2-b.inp		yes	
tinm6p2-s.inp		yes	
tinm7p1-b.inp		yes	
tinm7p1-s.inp		yes	
tinm7p2-b.inp		yes	

Model in ./SJC	Restart file	Selected	Reason for non-selection
tinm7p2-s.inp		yes	
tinm8p1-b.inp		yes	
tinm8p1-s.inp		yes	
valveTest1.inp		yes	
valveTest2.inp		yes	

C.22 Suites ./TraceSpecies

Number of selected input files: 13
Input files modified for selection: 0

Number of non-selected input files: 0
Input files written in fixed-format: 0
Input files yielding no or faulty XTV: 0

Total number of input files: 13

Model in ./TraceSpecies	Restart file	Selected	Reason for non-selection
mscl_itrace1.inp		yes	
mscv_itrace1.inp		yes	
mspl_itrace1.inp		yes	
mspv_itrace1.inp		yes	
mssl_itrace1.inp		yes	
mssv_itrace1.inp		yes	
mstl_itrace1.inp		yes	
mstv_itrace1.inp		yes	
msvl_itrace1.inp		yes	
msvlr_itrace1.inp	**msvl_itrace1.tpr**	yes	
msvv_itrace1.inp		yes	
msvl-trace.inp		yes	
msvv-trace.inp		yes	

C.23 Suites ./Valve

Number of selected input files: 88
Input files modified for selection: 1 (marked in table as '*')

Number of non-selected input files: 35
Input files written in fixed-format: 3 (no Namelist option)
Input files yielding no or faulty XTV: 32

Total number of input files: 123

Model in ./Valve	Restart file	Selected	Reason for non-selection
1valv-SD-BC-Pcte.inp		yes	
CBValve.inp		yes	
CBValveCv.inp		yes	
CBValveCv_Err1.inp		*	no XTV file (even after modification)
CBValveCv_Err2.inp			no XTV file
CBValveCv_Err3.inp			no XTV file
CBValveCv_Err4.inp		yes	
CBValveCv_rst.inp	**CBValveCv.tpr**	yes	
CBValveCv_rst0.inp	**CBValveCv.tpr**	yes	
CBValveCvSJC.inp		yes	
CBValveSJC.inp		yes	
CBValveStemCv.inp		yes	
CBValveStemCv_rst.inp	**CBValveStemCv.tpr**	yes	
CBValveStemCv_rst0.inp	**CBValveStemCv.tpr**	yes	
CBValveStemCvSJC.inp		yes	
CBValveStem.inp		yes	

Model in ./Valve	Restart file	Selected	Reason for non-selection
MotorValve.inp		yes	
MotorValve1.inp		yes	
MotorValve2.inp		yes	
MotorValve3.inp		yes	
MotorValve4.inp			no XTV file
MotorValve5.inp			no XTV file
MotorValve6.inp			no XTV file
MotorValve7.inp			no XTV file
MotorValveCv.inp		yes	
MotorValveCv.Mod1.inp		yes	
MotorValveCv_Err1.inp			no XTV file
MotorValveCv_Err2.inp			no XTV file
MotorValveCv_Err3.inp			no XTV file
MotorValveCv_Err4.inp		yes	
MotorValveCvFor.inp		yes	
MotorValveCv_rst.inp	**MotorValveCv.tpr**	yes	
MotorValveCv_rst0.inp	**MotorValveCv.tpr**	yes	
MotorValveStem.inp		yes	
MotorValveStemCv.inp		yes	
MotorValveStemCvFor.inp		yes	
MotorValveStemCv_rst.inp	**MotorValveStemCv.tpr**	yes	

Model in ./Valve	Restart file	Selected	Reason for non-selection
MotorValveStemCv_rst0.inp	MotorValveStemCv.tpr	yes	
TestFILLVALVE.inp		yes	
Type0Valve.inp		yes	
Type0Valve.Mod1.inp		yes	
Type0Valve.Mod2.inp			no XTV file
Type1Valve.inp		yes	
Type1Valve.Mod1.inp		yes	
Type1Valve.Mod2.inp		yes	
Type1Valve.Mod3.inp		yes	
Type1Valve.Mod3R.inp	Type1Valve.Mod3.tpr	yes	
Type2Valve.inp		yes	
Type2Valve.Mod1.inp		yes	
Type2Valve.Mod2.inp			no XTV file
Type3Valve.inp		yes	
Type3Valve.Mod1.inp		yes	
Type3Valve.Mod2.inp			no XTV file
Type3ValveR1.inp	Type3Valve.tpr	yes	
Type3ValveR2.inp	Type3Valve.tpr	yes	
Type3ValveR3.inp	Type3Valve.tpr	yes	
Type3ValveR4.inp	Type3Valve.tpr	yes	
Type4Valve.inp		yes	

Model in ./Valve	Restart file	Selected	Reason for non-selection
Type4Valve.Mod1.inp		yes	
Type4Valve.Mod2.inp			no XTV file
Type5Valve.inp		yes	
Type5Valve.Mod1.inp		yes	
Type5Valve.SJC.inp		yes	
Type6Valve.inp			no XTV file
Type6Valve.Mod1.inp			no XTV file
Type7Valve.inp		yes	
Type7Valve.Mod1.inp		yes	
Type7Valve.Mod2.inp		yes	
Type7Valve.Mod3.inp		yes	
Type7Valve.Mod4.inp		yes	
Type7Valve.Mod5.inp		yes	
Type7Valve.Mod6.inp		yes	
Type7Valve.Mod7.inp		yes	
Type7Valve.Mod8.inp		yes	
Type7Valve.Mod9.inp		yes	
Type7Valve.Mod10.inp		yes	
Type7Valve.Mod11.inp		yes	
Type7Valve.Mod12.inp		yes	
Type7Valve.Mod13.inp		yes	

Model in ./Valve	Restart file	Selected	Reason for non-selection
Type8Valve.inp		yes	
Type8Valve.Err1.inp			no XTV file
Type8Valve.Err2.inp			no XTV file
Type8Valve.Err3.inp			no XTV file
Type8Valve.Err4.inp			no XTV file
Type8Valve.Mod1.inp		yes	
Type8Valve.Mod1R.inp	**Type8Valve.Mod1.tpr**	yes	
Type8Valve.Mod2.inp			no XTV file
Type8Valve.Mod3.inp		yes	
Type8Valve.Mod3R.inp	**Type8Valve.Mod3.tpr**	yes	
Type8Valve.SJC.inp			no XTV file
Type8Valve.Warn1.inp		yes	
Type8Valve.Warn2.inp		yes	
Type9Ck.inp		yes	
Type9Ck-Mod2.inp		yes	
Type9Ck-Mod3.inp		yes	
Type9Valve.inp		yes	
Type9Valve.Mod1.inp			no XTV file
Type9Valve.Mod2.inp			no XTV file
Type9Valve.Mod3.inp			no XTV file
Type10Valve.inp		yes	

Model in ./Valve	Restart file	Selected	Reason for non-selection
Type10Valve.Mod1.inp			no XTV file
Type10Valve_Mod2.inp		yes	
Type10Valve_Mod2R.inp	**Type10Valve_Mod2.tpr**	yes	
Type10Valve_Mod3.inp		yes	
Type10Valve_Mod4.inp		yes	
Type10Valve_Mod5.inp		yes	
Type10Valve_Mod6.inp		yes	
Type11Valve.inp		yes	
Type11Valve.Mod1.inp			no XTV file
Type-1ValveCv.inp		yes	
Type-1Valve.inp		yes	
Type-1Valve.Mod1.inp		yes	
Type-1Valve.Mod2.inp		yes	
Type-1Valve.Mod3.inp			no XTV file
Type-1Valve.Mod4.inp			no XTV file
Type-1Valve.Mod5.inp		yes	
Type-1Valve.Mod6.inp			no XTV file
Type-1Valve.Mod7.inp			no XTV file
Type-1Valve.Mod8.inp			no XTV file
ValveFxModel.inp		yes	
ValveTyp6.inp			fixed-format input

Model in ./Valve	Restart file	Selected	Reason for non-selection
ValveTyp7.inp			fixed-format input
ValveTyp7b.inp			fixed-format input

C.24 Suites ./Vess2Vess

Number of selected input files: 23
Input files modified for selection: 0

Number of non-selected input files: 2
Input files written in fixed-format: 0
Input files yielding no or faulty XTV: 2

Total number of input files: 25

Model in ./Vess2Vess	Restart file	Selected	Reason for non-selection
Bankoff2HD.inp		yes	
Bankoffsatstm.inp		yes	
BankoffSJC.inp		yes	
BankoffSJC2.inp		yes	
BankoffVess.inp		yes	
I-2pip-SJC.inp		yes	
mid3Dch.inp		yes	
midPipe1Ch.inp		yes	
midPipe3Ch.inp		yes	
One3d.inp		yes	
One3dl.inp		yes	
One3dSJCbugs.inp			no XTV file
One3dSJC.inp		yes	
One3DSJCl.inp		yes	
pipe1SJC.inp		yes	
pipe2SJC.inp		yes	

Model in ./Vess2Vess	Restart file	Selected	Reason for non-selection
pipeOnly.inp		yes	
Three3D.inp		yes	
Two3D.inp		yes	
Two3D2.inp		yes	
v2vBase.inp		yes	
v2vInBugs.inp			no XTV file
v2vPipes.inp		yes	
v2vSJC.inp		yes	
W412SSOldVessel.inp		yes	

APPENDIX D
ANALYSIS OF FAILED VERIFICATION TESTS OF V5.0P3

This appendix lists and discusses comparison plots of the evolutions of time-step and sampled XTV variables for each verification case of TRACE version v5.0p3 that resulted in failed functional tests for XTV file content. We recall that failure means the XTV variable evolution obtained from one of the test model variants was not identical to the same evolution obtained with the reference executable.

In the verification of version v5.0p3, 41 cases failed the functional test, specifically 6 models from suites “ConSys” (see plots in subsection D.1), 12 from “HtStr” (D.2), 11 from “Power” (D.3) and 12 from “Short2” (D.4). All these tests belong to variant 10, which compares to a reference case that uses graphics output level "minimal".

For each of the failed test cases, the corresponding pair of figures in the following subsections also indicates whether the model was executed standalone or using a restart file (.tpr). In fact, 25 of the affected models were standalone and the other 16 were using a restart file. Note that most of the restart files were obtained from one or the other of the standalone models that failed the comparison test. Therefore, one can safely assume that the root cause of the discrepancy between reference and modified results resides in the standalone simulations.

A closer look at the figures shows that for all of the (standalone) cases, differences between the extracted variables are minimal (if not inexistent) until the instant when the time-steps start to differ. This, together with the fact that the difference between the variables remain small even after onset of time-step dissimilarities, hints at some round-off errors or numerical diffusion resulting from the implementation of the XTV-Customize option.

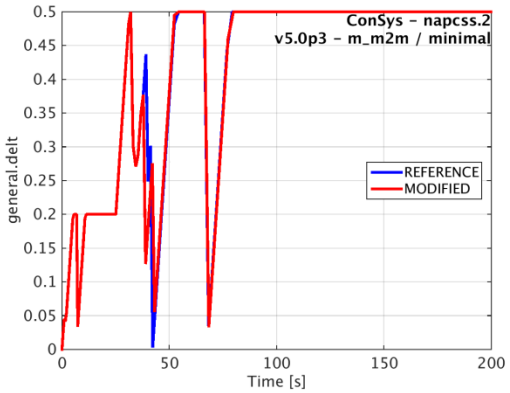
It was noted that all the affected standalone models include a TRACE component ‘Prizer’ and make use of the steady-state initialization option (TRACE input variable ‘istdyst’ > 0). Further analysis of the source code of TRACE v5.0p3 confirmed that this combination of input could trigger an incomplete treatment in subroutine {AddXtvDump} (source file [src/XtvDumpM.f90]) of some of the ‘Prizer’ variables that are iteratively adjusted to achieve a converged steady-state. This incomplete treatment is actually more revealed than caused by the XTV-Customize option.

Note also that the aforementioned steady-state treatment of the ‘Prizer’ variables has been made insensitive to changes of option graphlevel in the latest versions of TRACE (v5.0p4 and v5.0p5), which indeed did not exhibit such undesirable test results after implementation of the XTV-Customize option (see subsection 4.3.2).

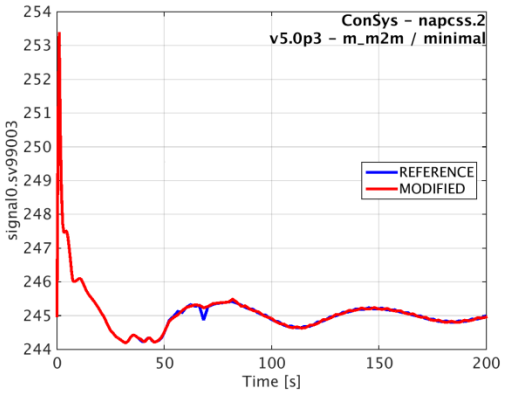
Given the small amplitude of the discrepancy, as confirmed by visual inspection of the figures presented thereafter, and since the affected code version(s) is superseded by versions that correct for the issue, it was not deemed necessary to retro-actively modify the subroutine {AddXtvDump} in the XTV-Customize option patch for v5.0p3. Note also that this modification would not be judicious because it would detrimentally affect the outcome of the non-regression test for some of the test cases of this code version (see Table 4-3).

D.1 Comparison Figures for Suites ./ConSys

input: napcss.2.inp
restart: none

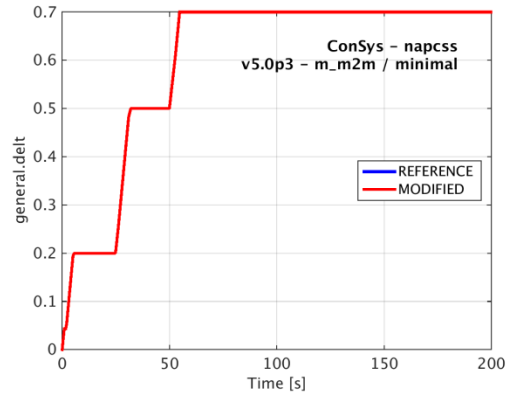


time-step

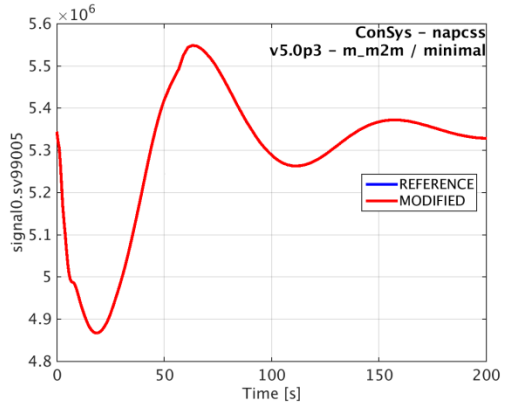


sampled variable

input: napcss.inp
restart: none

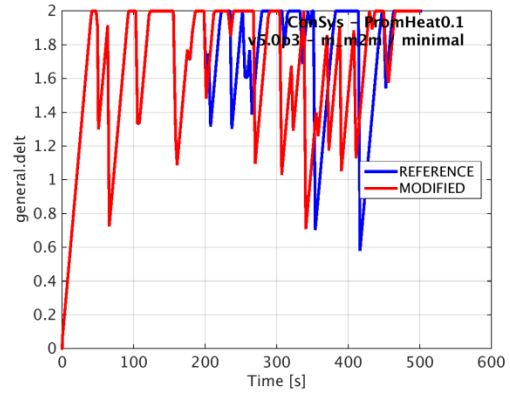


time-step

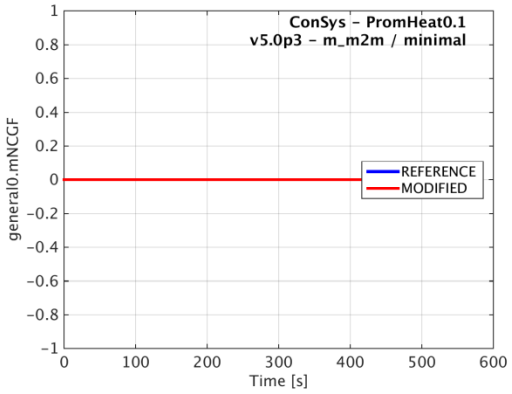


sampled variable

input: PromHeat0.1.inp
restart: none

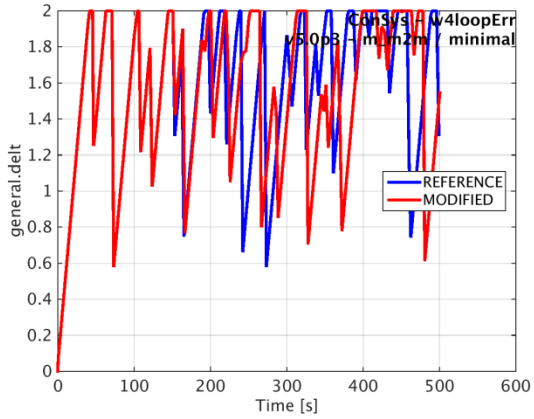


time-step

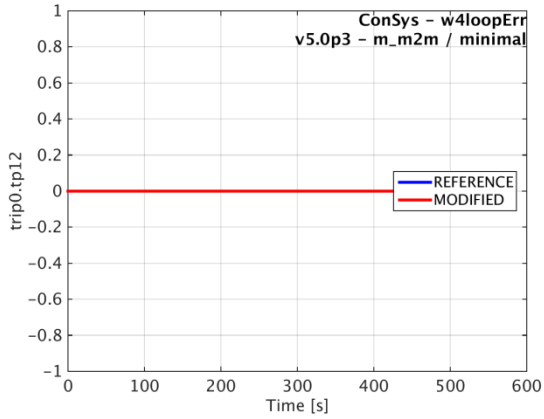


sampled variable

input: w4loopErr.inp
restart: none

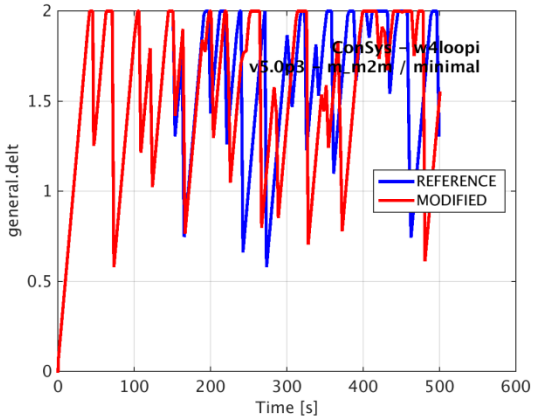


time-step

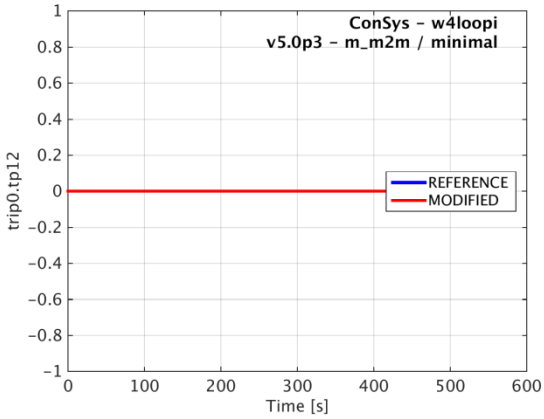


sampled variable

input: w4loopi.inp
restart: none

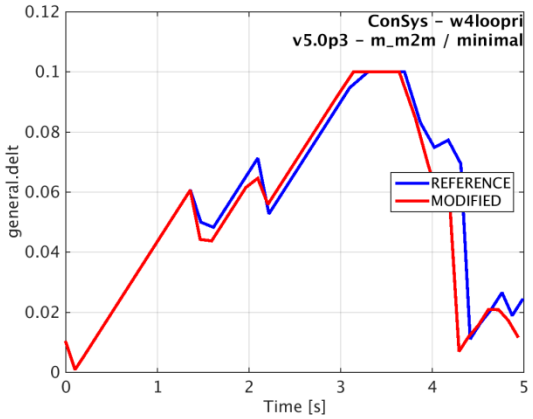


time-step

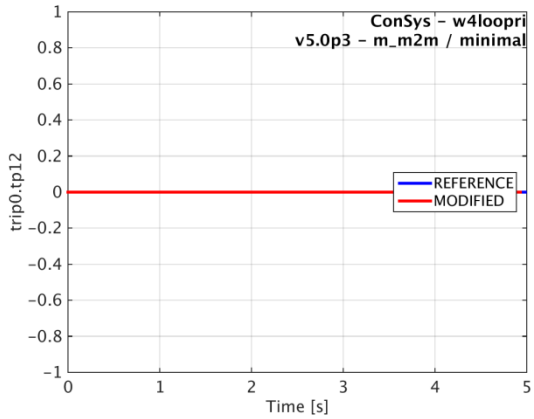


sampled variable

input: w4loopri.inp
restart: w4loopi.tpr



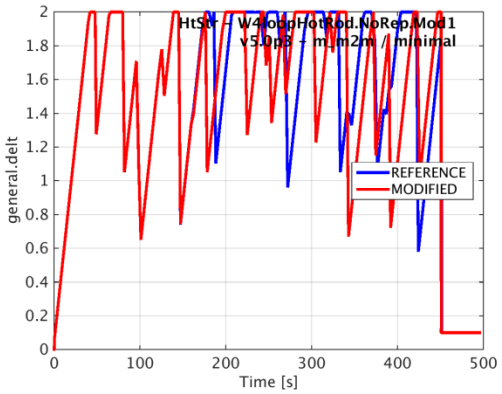
time-step



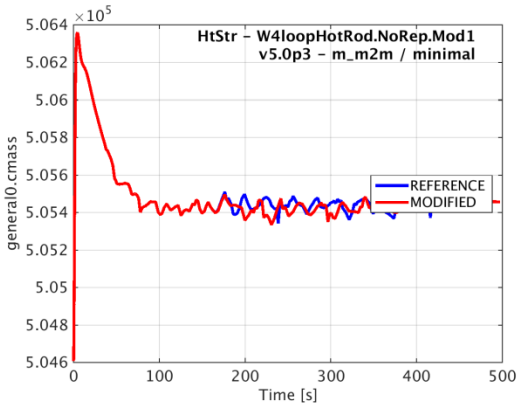
sampled variable

D.2 Comparison Figures for Suites ./HtStr

input: W4loopHotRod.NoRep.Mod1.inp
restart: none

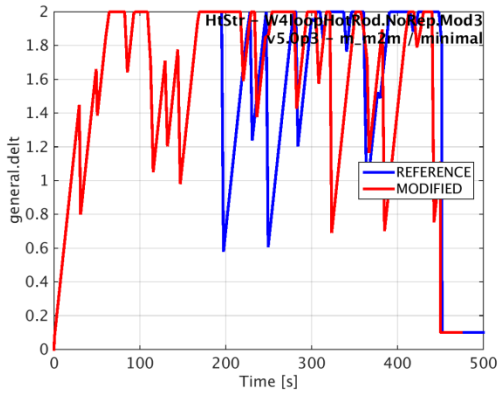


time-step

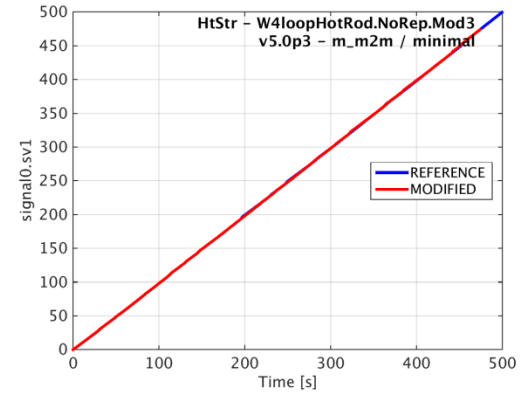


sampled variable

input: W4loopHotRod.NoRep.Mod3.inp
restart: none

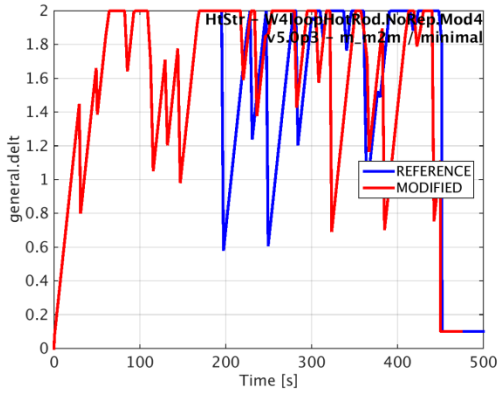


time-step

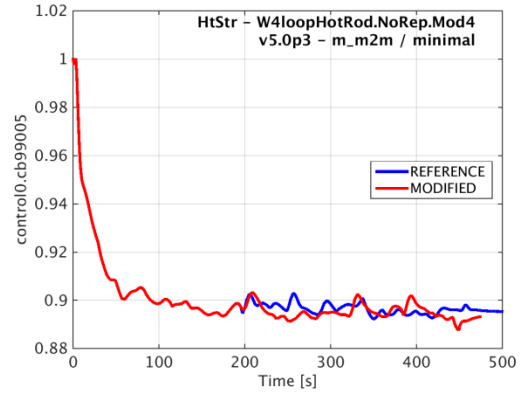


sampled variable

input: W4loopHotRod.NoRep.Mod4.inp
restart: none

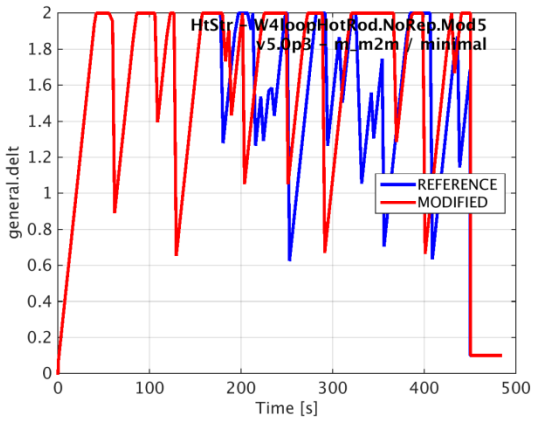


time-step

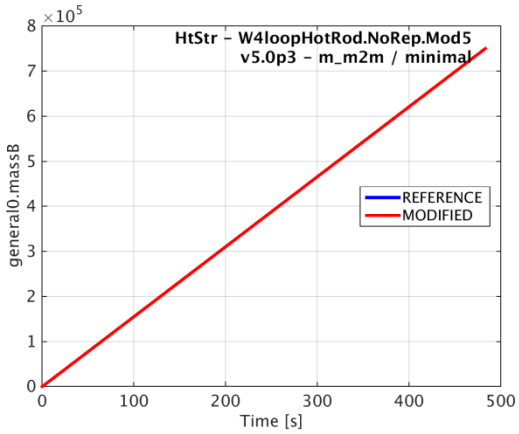


sampled variable

input: W4loopHotRod.NoRep.Mod5.inp
restart: none

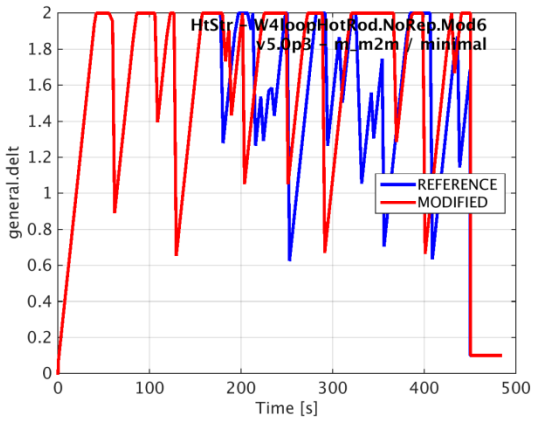


time-step

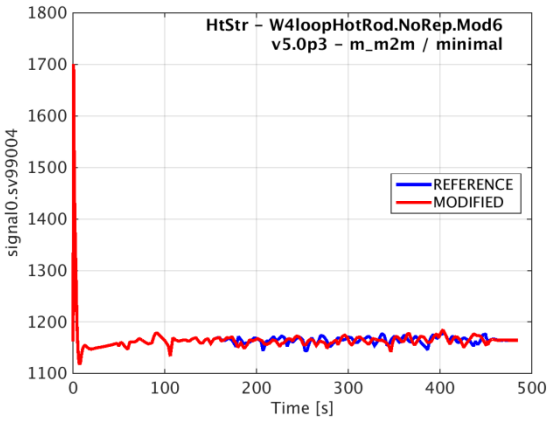


sampled variable

input: W4loopHotRod.NoRep.Mod6.inp
restart: none

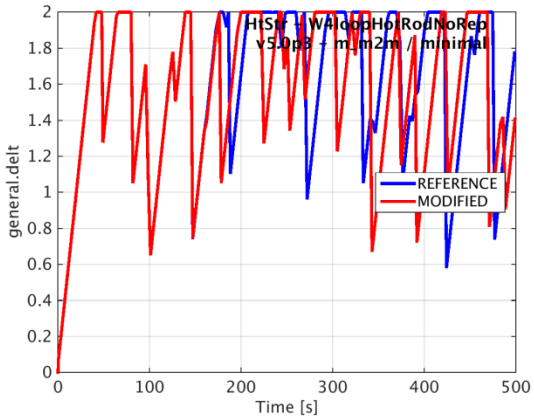


time-step

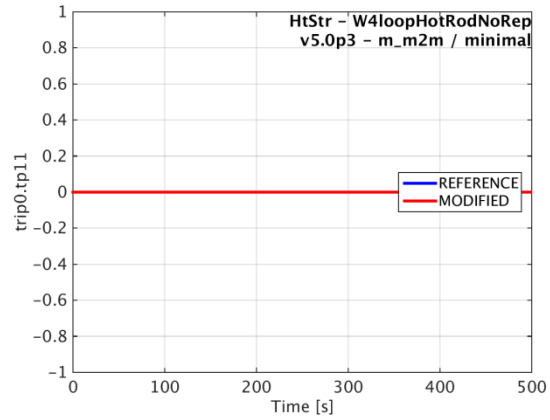


sampled variable

input: W4loopHotRodNoRep.inp
restart: none

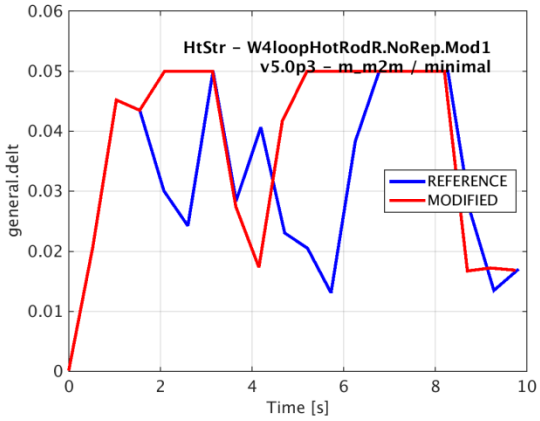


time-step

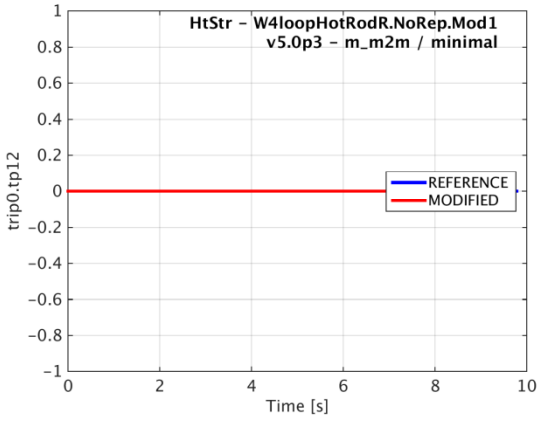


sampled variable

input: W4loopHotRodR.NoRep.Mod1.inp
restart: W4loopHotRod.NoRep.Mod1.tpr

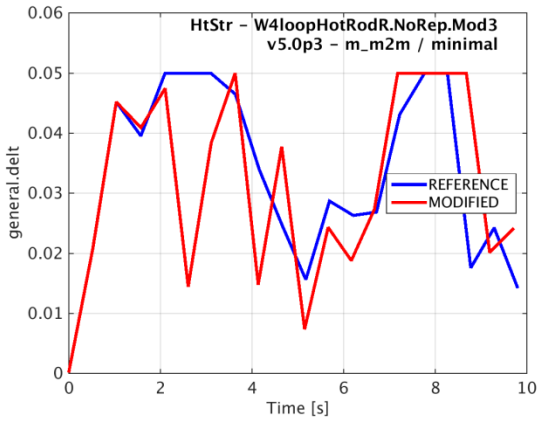


time-step

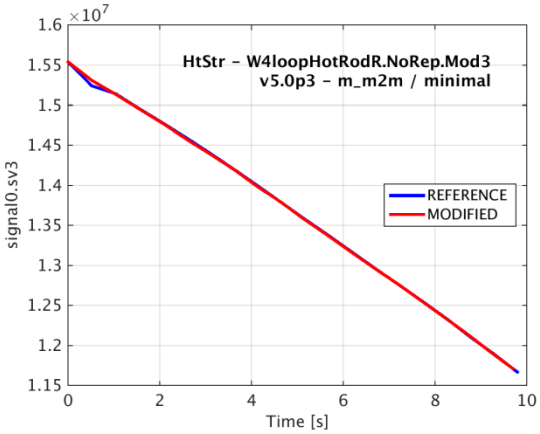


sampled variable

input: W4loopHotRodR.NoRep.Mod3.inp
restart: W4loopHotRod.NoRep.Mod3.tpr

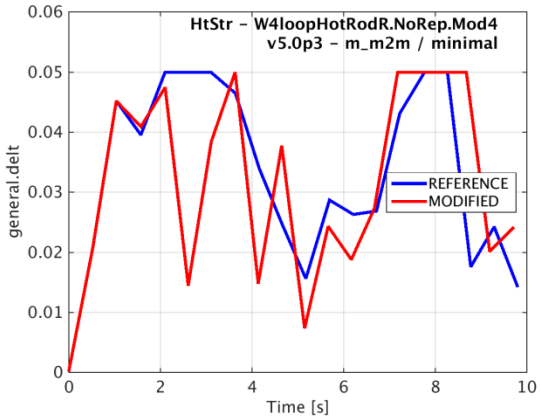


time-step

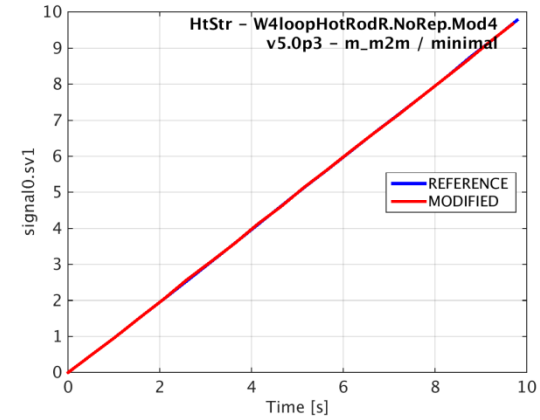


sampled variable

input: W4loopHotRodR.NoRep.Mod4.inp
restart: W4loopHotRod.NoRep.Mod4.tpr

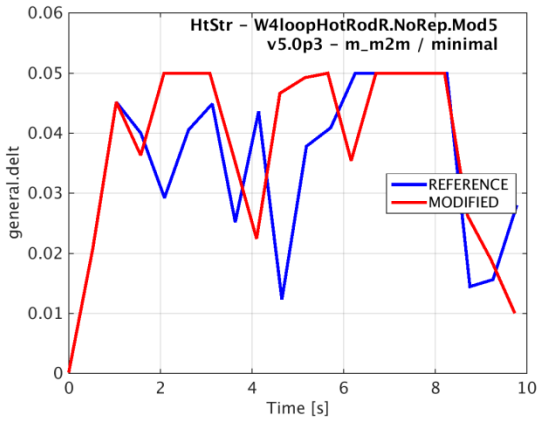


time-step

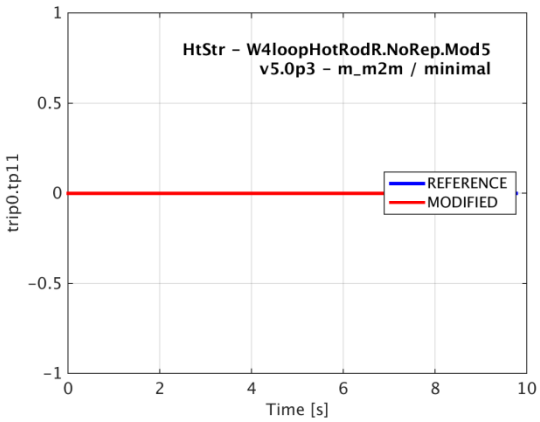


sampled variable

input: W4loopHotRodR.NoRep.Mod5.inp
restart: W4loopHotRod.NoRep.Mod5.tpr

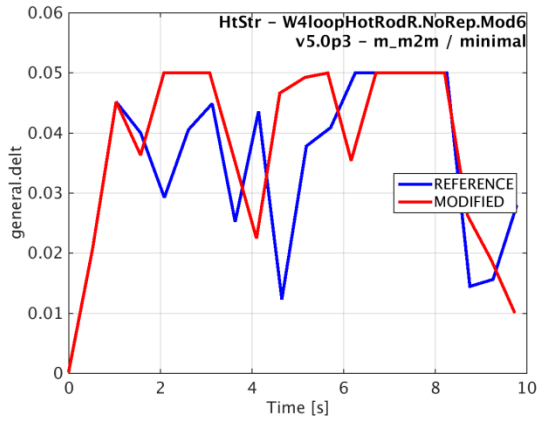


time-step

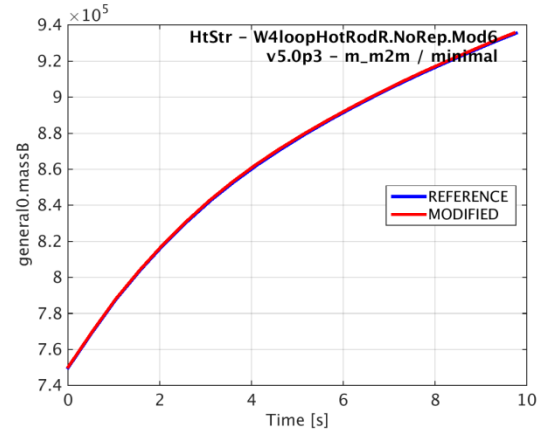


sampled variable

input: W4loopHotRodR.NoRep.Mod6.inp
restart: W4loopHotRod.NoRep.Mod6.tpr

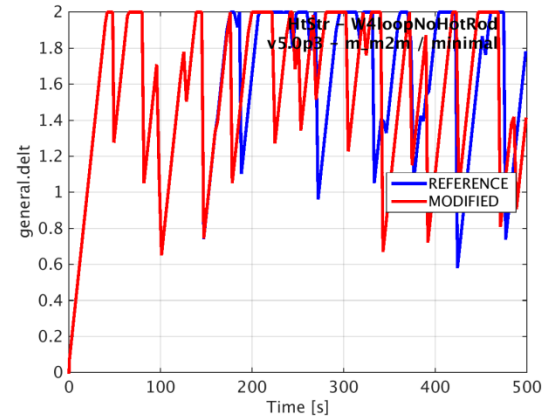


time-step

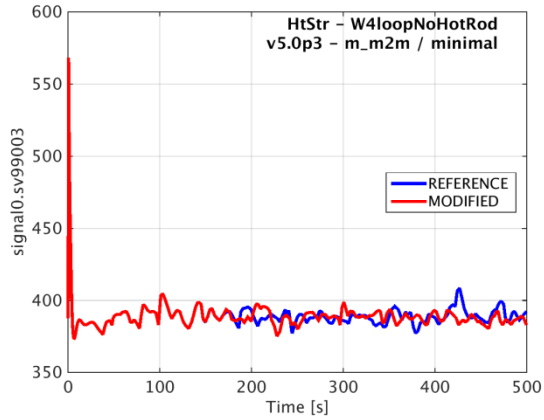


sampled variable

input: W4loopNoHotRod.inp
restart: none



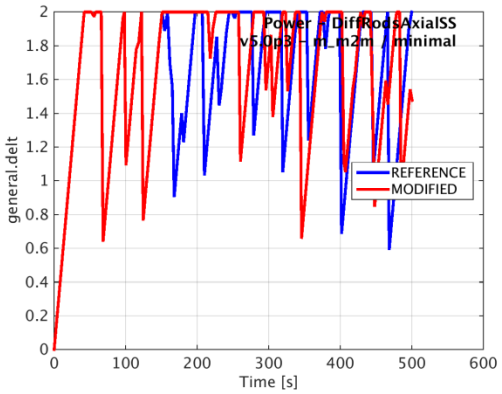
time-step



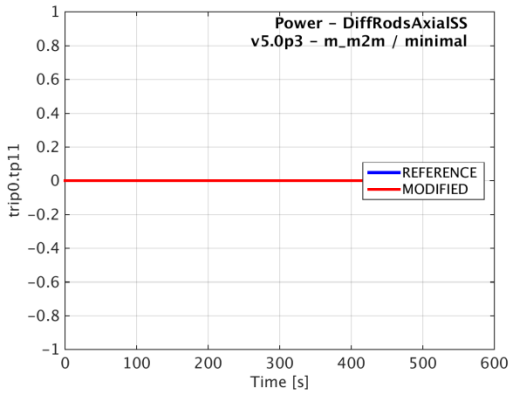
sampled variable

D.3 Comparison Figures for Suites ./Power

input: DiffRodsAxialSS.inp
restart: none

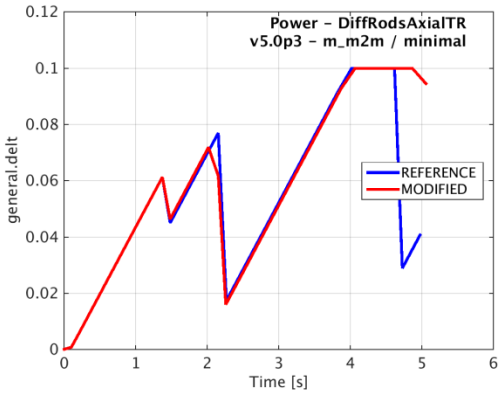


time-step

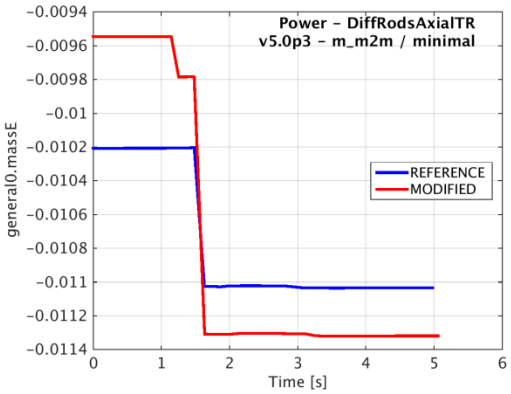


sampled variable

input: DiffRodsAxialTR.inp
restart: DiffRodsAxialSS.tpr

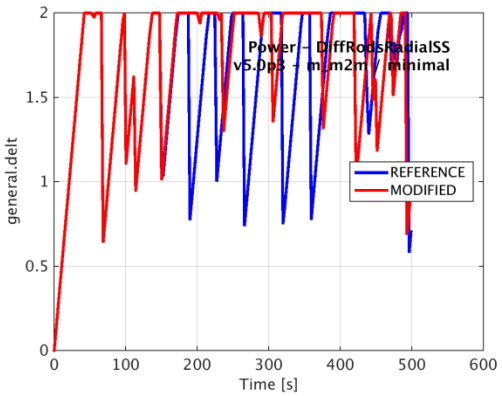


time-step

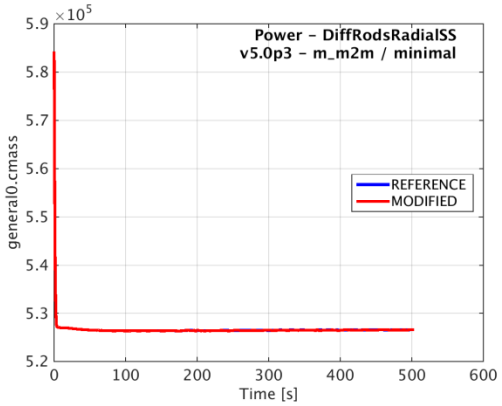


sampled variable

input: DiffRodsRadialSS.inp
restart: none

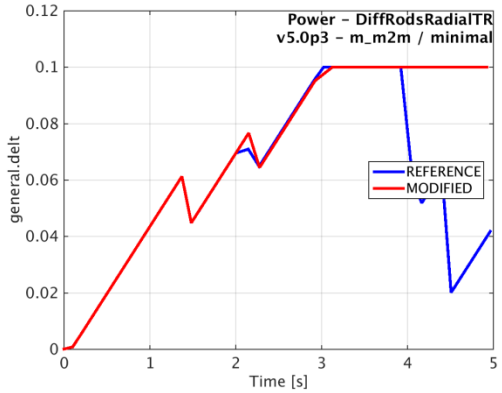


time-step

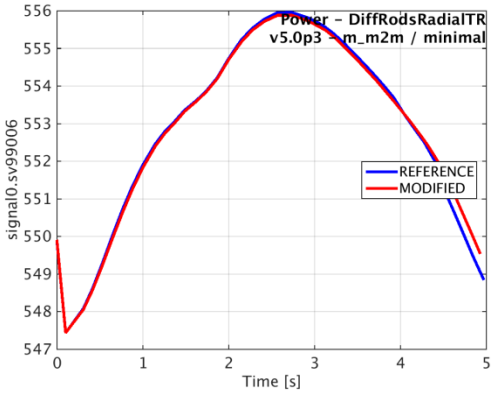


sampled variable

input: DiffRodsRadialTR.inp
restart: DiffRodsRadialSS.tpr

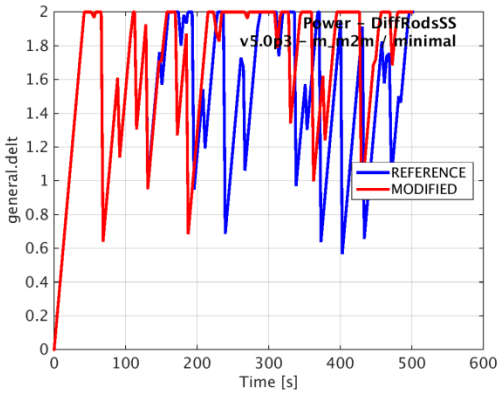


time-step

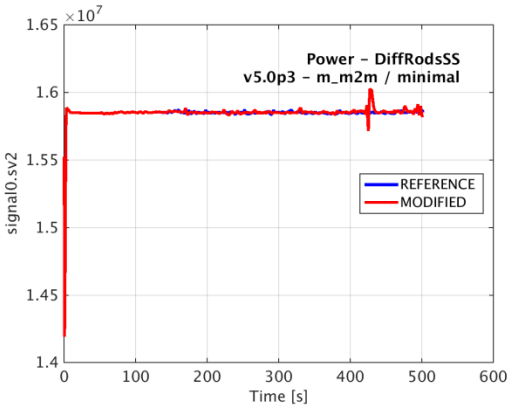


sampled variable

input: DiffRodsSS.inp
restart: none

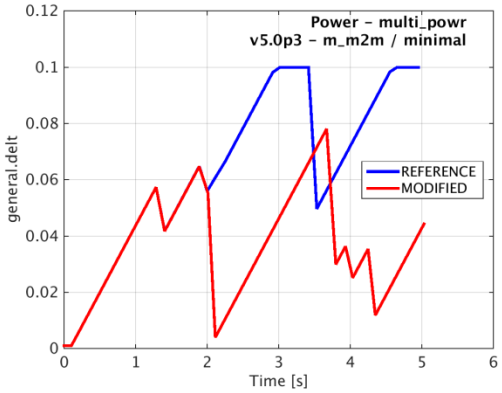


time-step

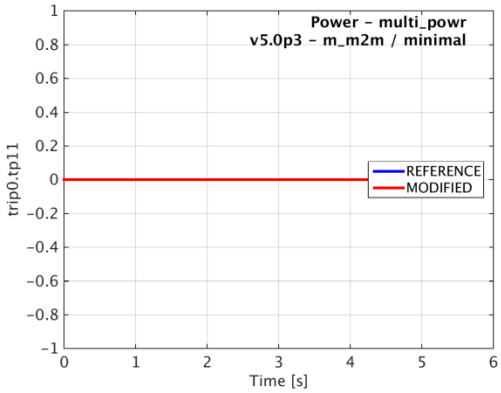


sampled variable

input: multi_powr.inp
restart: multi_pow.tpr

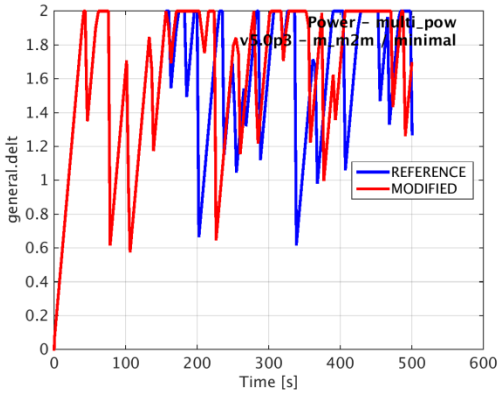


time-step

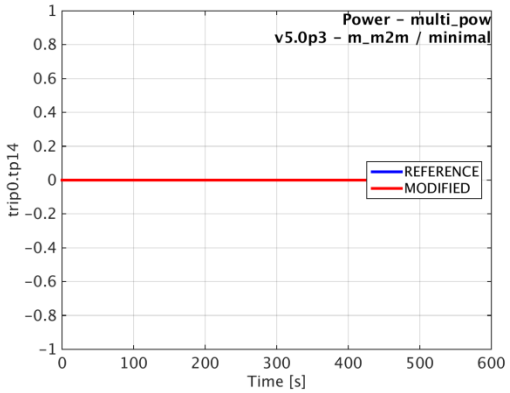


sampled variable

input: multi_pow.inp
restart: none

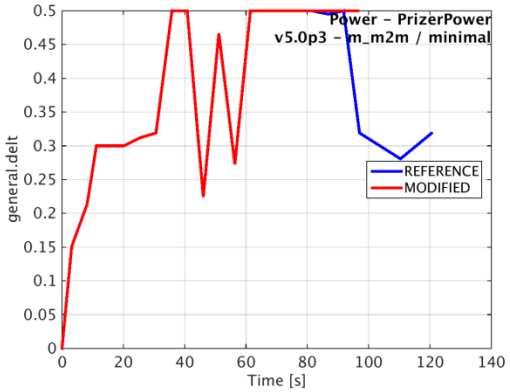


time-step

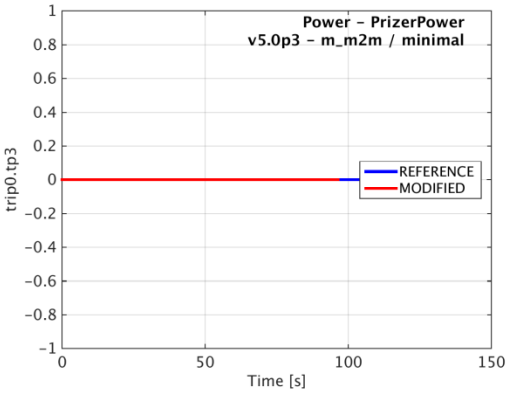


sampled variable

input: PrizerPower.inp
restart: none

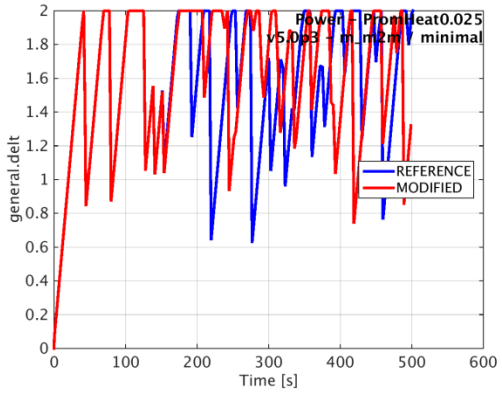


time-step

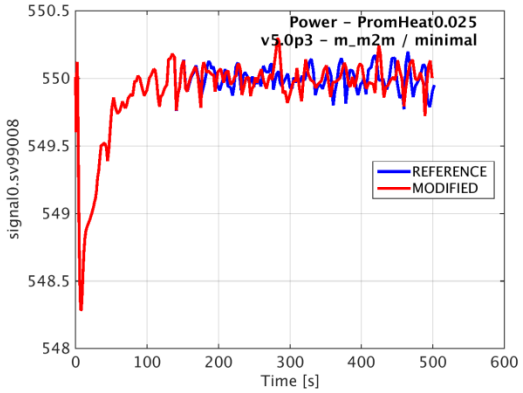


sampled variable

input: PromHeat0.025.inp
restart: none

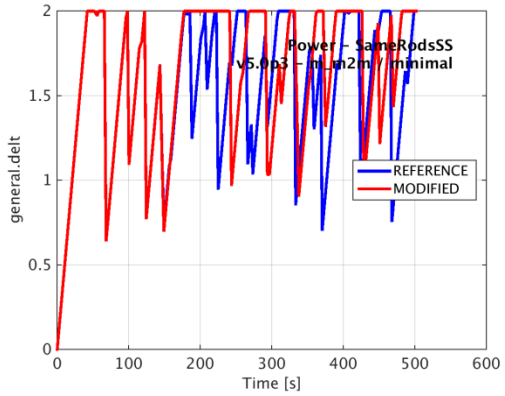


time-step

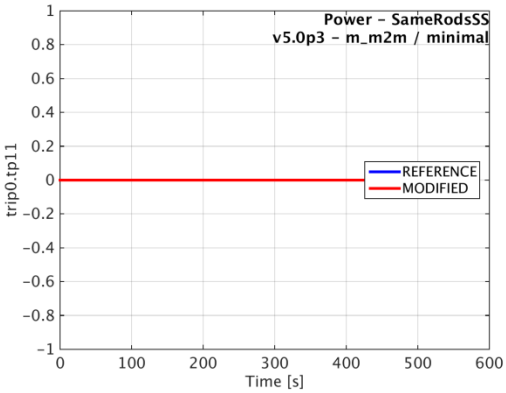


sampled variable

input: SameRodsSS.inp
restart: none

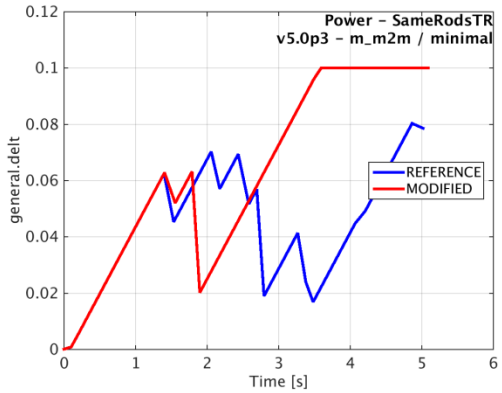


time-step

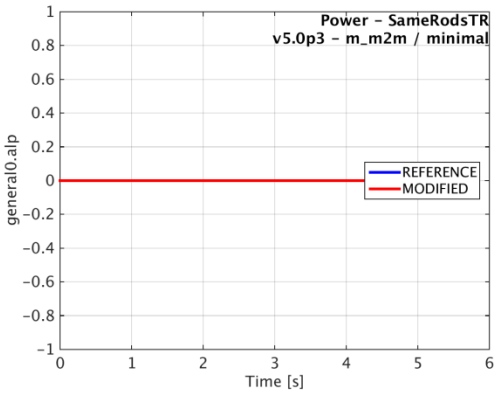


sampled variable

input: SameRodsTR.inp
restart: SameRodsSS.tpr



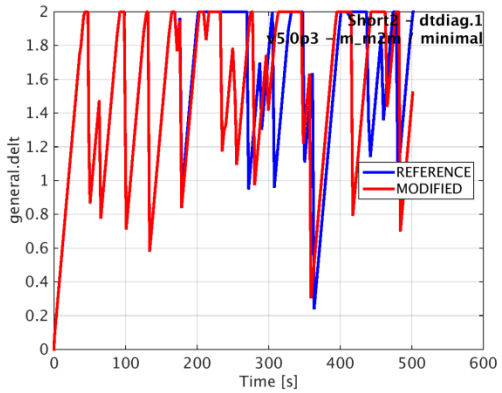
time-step



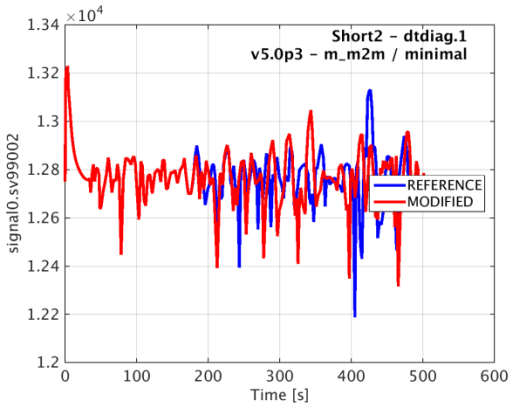
sampled variable

D.4 Comparison Figures for Suites ./Short2

input: dtdiag.1.inp
restart: none

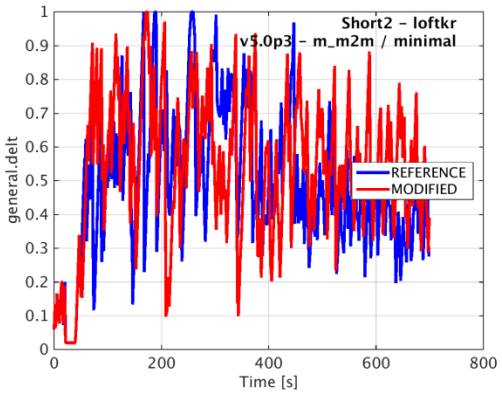


time-step

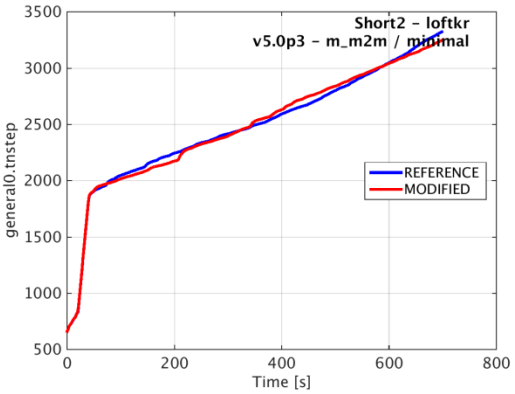


sampled variable

input: loftkr.inp
restart: loftk.tpr

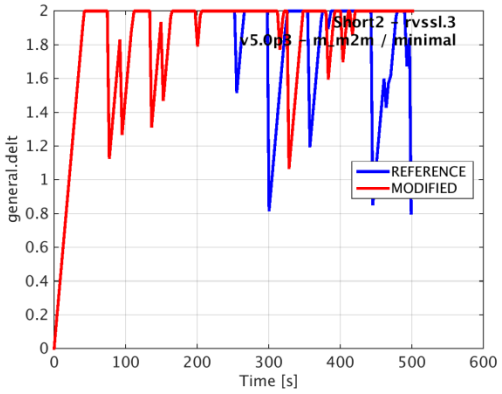


time-step

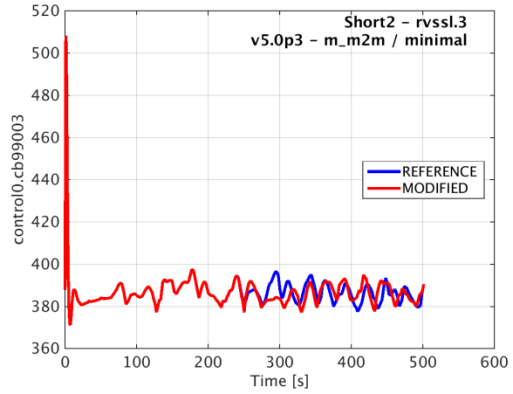


sampled variable

input: rvssl.3.inp
restart: none

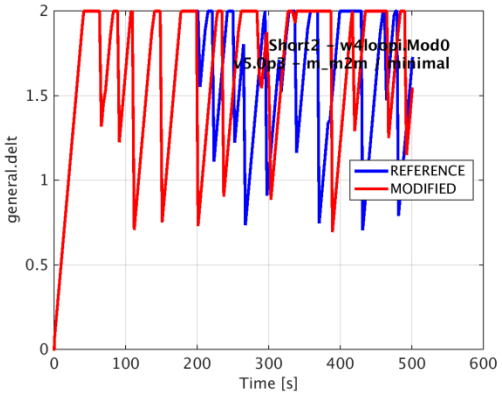


time-step

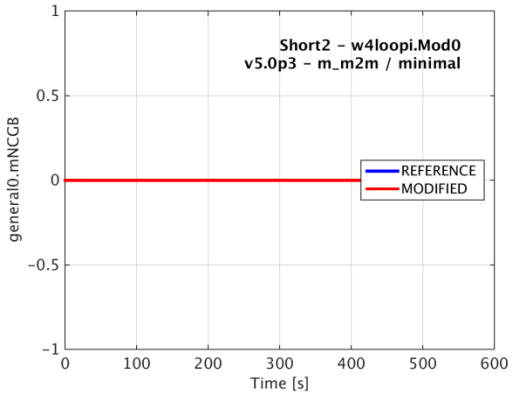


sampled variable

input: w4loopi.Mod0.inp
restart: none

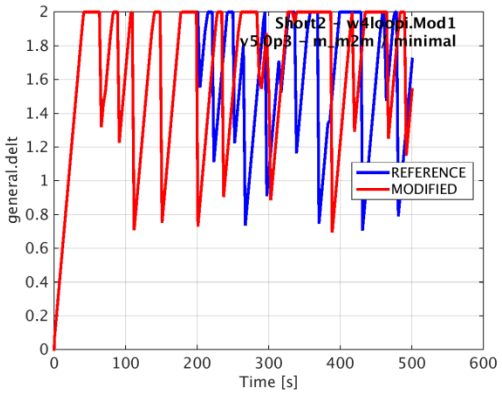


time-step

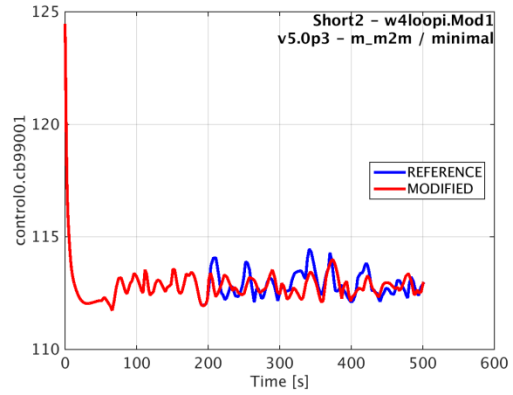


sampled variable

input: w4loopi.Mod1.inp
restart: none

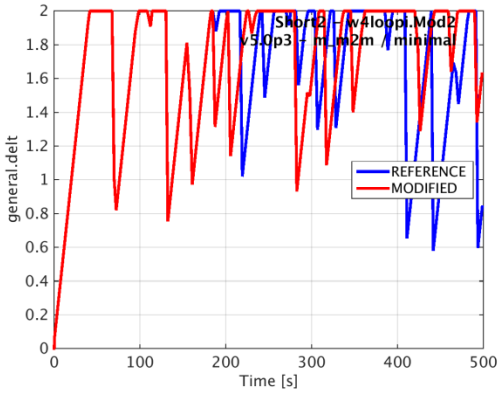


time-step

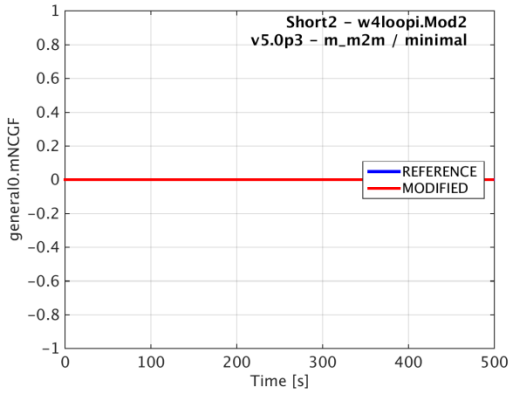


sampled variable

input: w4loopi.Mod2.inp
restart: none

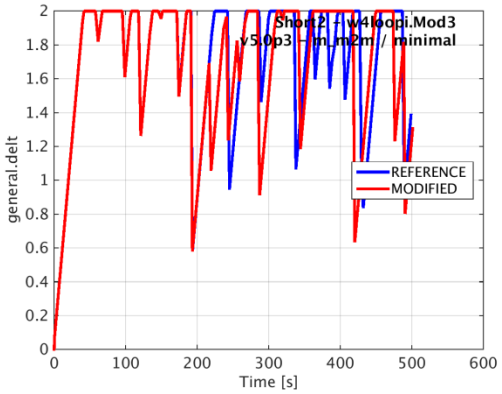


time-step

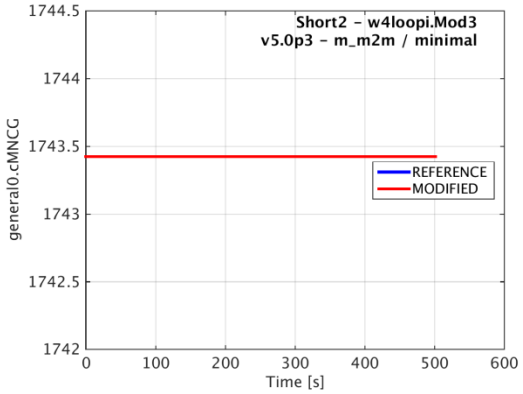


sampled variable

input: w4loopi.Mod3.inp
restart: none

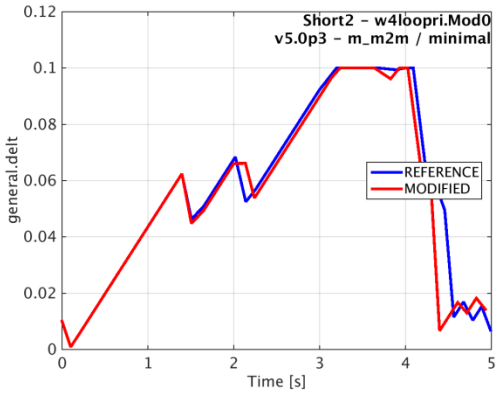


time-step

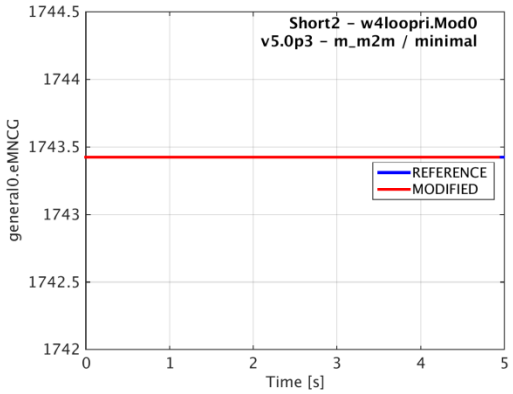


sampled variable

input: w4loopri.Mod0.inp
restart: w4loopi.Mod0.tpr

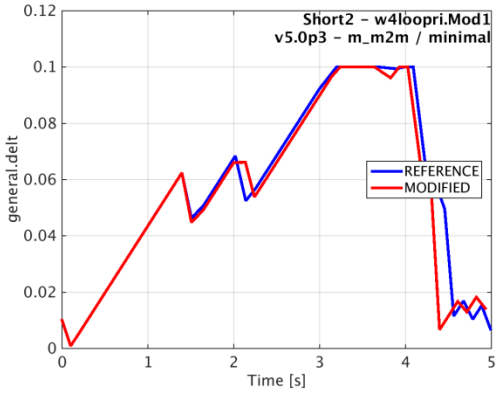


time-step

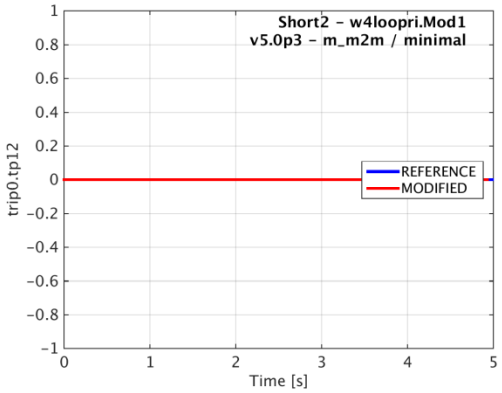


sampled variable

input: w4loopri.Mod1.inp
restart: w4loopi.Mod1.tpr

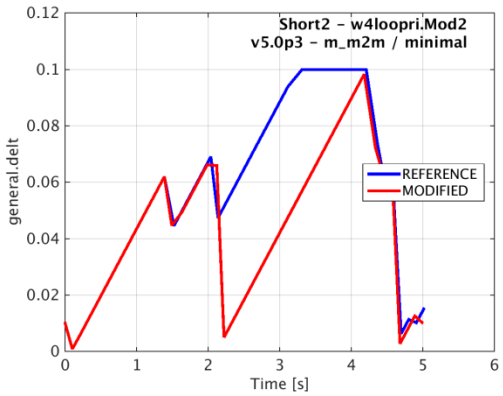


time-step

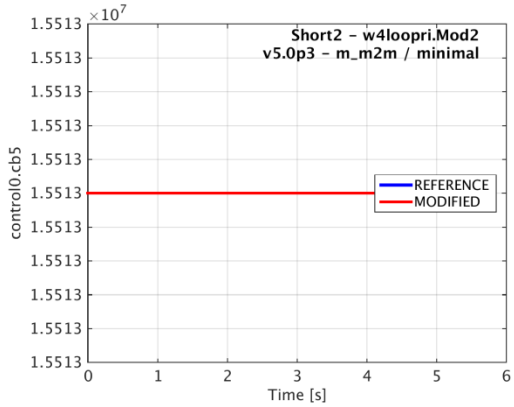


sampled variable

input: w4loopri.Mod2.inp
restart: w4loopi.Mod2.tpr

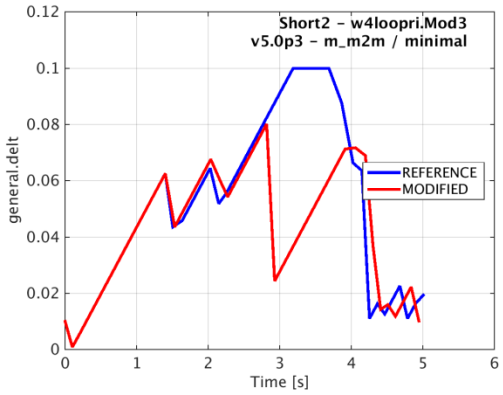


time-step

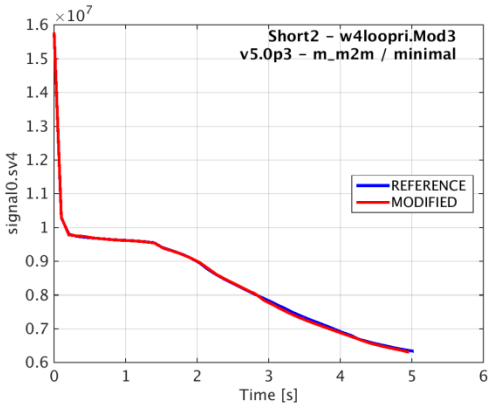


sampled variable

input: w4loopri.Mod3.inp
restart: w4loopi.Mod3.tpr

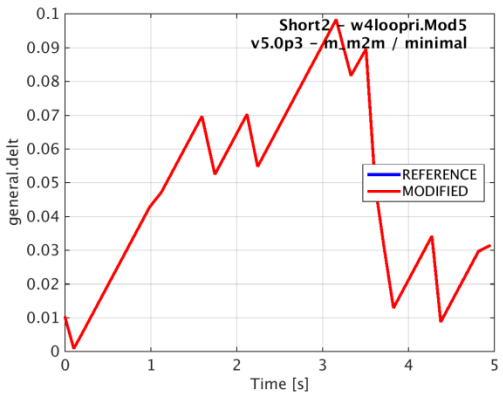


time-step

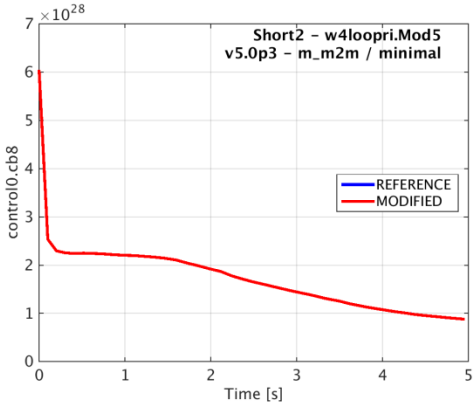


sampled variable

input: w4loopri.Mod5.inp
restart: w4loopi.Mod5.tpr



time-step



sampled variable

APPENDIX E
TRACE MODEL OF THE REFLOOD TEST FEBA-216

This appendix describes the Feba test facility and the corresponding TRACE model to simulate the reflood test 216. This appendix is based on information of sections 2.3 and 2.4 of [8]. More detailed information on the Feba facility can be found in [5].

E.1 FEBA Reflood Facility

A series of Feba experiments was conducted in the 1980s at the Karlsruhe Institute of Technology (KIT) to improve the knowledge on heat transfer mechanism during the reflood phase of a LB-LOCA transient, with a special focus on the effects of spacer grids and flow blockage due to fuel rod ballooning. The data from the facility was also intended to provide data for code and models validation.

The facility consisted of a test section with a full height 5x5 bundle of PWR fuel rod simulator (Figure E-1a) enclosed in a rectangular stainless steel housing (Figure E-1b). An approximate cosine power profile was mapped over the height of the fuel rod simulators (Figure E-1c). Seven spacer grids were used to provide mechanical support of the fuel rod simulators (Figure E-1d).

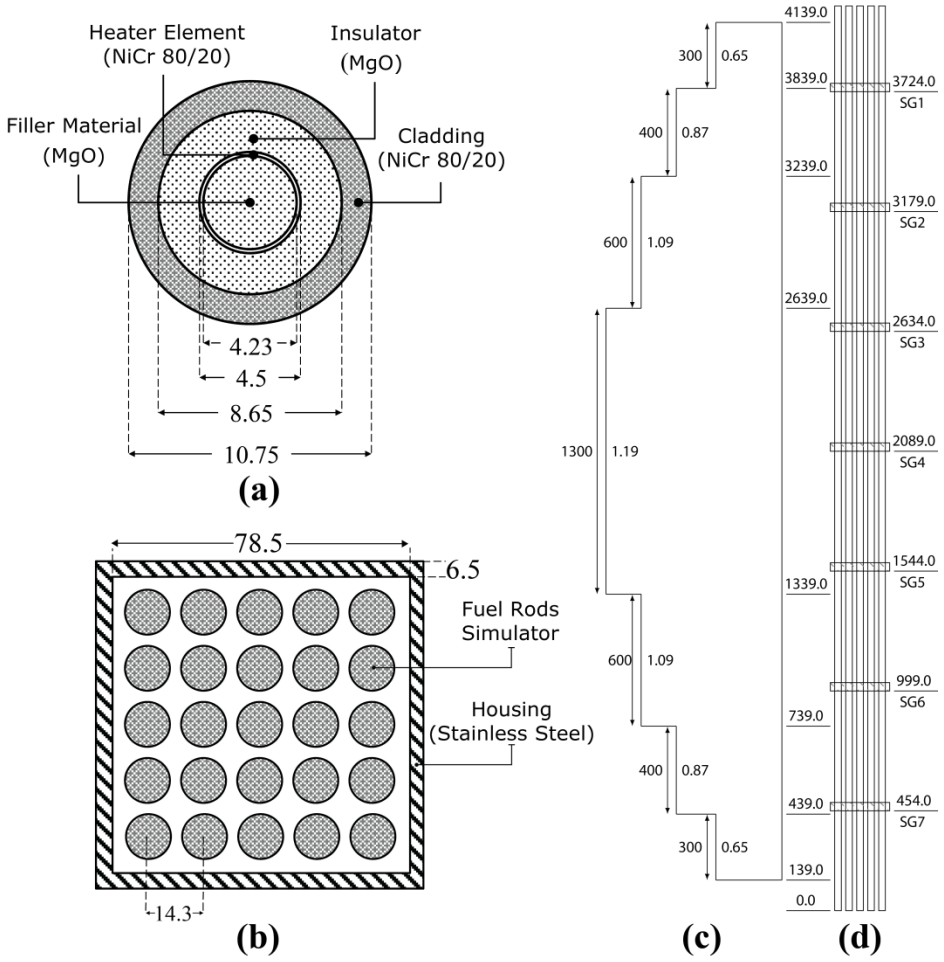


Figure E-1 Test Section of the Feba Test Facility (from [9])

During the initialization phase of the experiment, the test section was heated up at low nominal power (200 [kW]) to achieve a specified initial heater rod temperature, with essentially no liquid present in the test section. The transient phase of the experiment was initiated by ramping up the power according to 120% (ANS) decay heat power curve while simultaneously injecting subcooled liquid at the bottom inlet of the test section.

Eight different test series were performed in the Feba facility. The first two test series (I and II) used two different numbers of spacer grids, seven and six, respectively. The middle spacer grid was removed in test series II to investigate the effect of spacer grids in a reflood transient. The other test series used different flow area blockage sizes at mid-height of the test section in order to investigate the effect of rod ballooning.

In each test series, combinations of two different inlet liquid velocities and three different system backpressure were imposed. The Feba test 216 belongs to test series I. This particular test was conducted at a system pressure of 4.2 [Bar] and a liquid inlet velocity of 3.81 [$cm \cdot s^{-1}$], with all seven spacer grids mounted and no flow area blockage.

Three types of time histories were measured and recorded in the test. These included thermocouples to measure the outer rod clad temperature at eight different axial locations,

pressure probes to measure the pressure difference over four different axial segments of the test assembly, and a collecting tank measuring the mass of water carried over at the outlet of the test section. Note that the collecting tank for measuring the liquid carryover was saturated at 10 [kg] and thus no measurement above that value is available.

E.2 TRACE Model

The model was developed based on specifications provided within the context of the PREMIUM benchmark [7], following whenever possible the modeling best-practices guidelines for TRACE in order to minimize user effect [10].

The model includes the following TRACE components:

- A 1-dimensional 'Vessel' component to model the bundle test section,
- A 'Pipe' component to model the upper plenum of the test section,
- A 'Fill' component to set the inlet flow and inlet temperature boundary conditions,
- A 'Break' component to model the outlet pressure boundary condition,
- Two 'HtStr' components to model the heater rods simulator and the test section housing,
- A 'Power' component to impose the electrical power boundary condition.

The 'Vessel' component was divided into 28 hydraulic control volumes of sizes ranging from 60 to 315 [mm].

The two 'HtStr' components were axially divided into the same number of coarse axial conduction nodes. However, since a large axial temperature gradient is expected in a reflood transient, the fine-mesh reflood flag in TRACE was enabled. As a result, each of the coarse conduction nodes was divided uniformly in five, yielding a total of 142 axial conduction nodes.

The geometry and experiment specifications of the model are provided in Figure E-2.

Parameter	Units	Value	Nodalization schematic (from [9])
Test section total length	[m]	4.114	
Total heated length	[m]	3.9	
Flow area	[m^2]	$3.901 \cdot 10^{-3}$	
Hydraulic diameter	[mm]	13.45	
Rectangular housing width	[mm]	78.55	
Rectangular housing thickness	[mm]	6.5	
Number of rods	[-]	25	
Rod outer diameter	[mm]	10.75	
Pitch-to-Diameter ratio	[-]	1.33	
Number of spacer grids	[-]	7	
Spacer grid flow obstruction	[%]	20	
Spacer grid axial locations	[m]	0.454, 0.999, 1.544, 2.089, 2.634, 3.179, 3.724	
Number of hydraulic nodes	[-]	28 (varying length)	
Number of axial nodes	[-]	28 (coarse) 142 (fine)	
Inlet liquid temperature	[K]	312	
Inlet flow velocity	[cm s^{-1}]	3.81	
System backpressure	[Bar]	4.2	

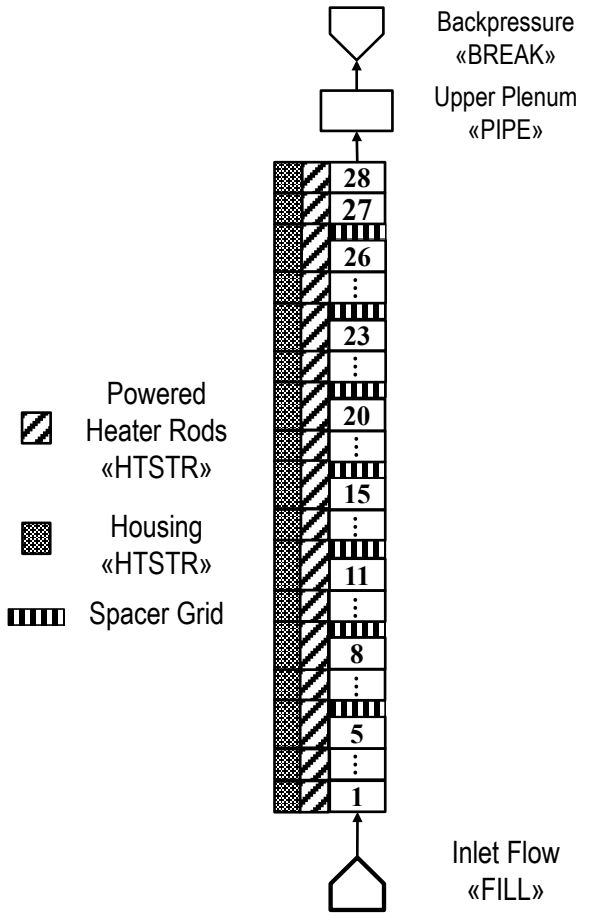


Figure E-2 Parameters and Nodalization of the TRACE Model

NRC FORM 335
(12-2010)
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-0514

2. TITLE AND SUBTITLE

Customization of XTV Graphics Output in TRACE v5.0 Patches 5, 4 & 3

3. DATE REPORT PUBLISHED

MONTH	YEAR
August	**2019**

4. FIN OR GRANT NUMBER

5. AUTHOR(S)

O. Zerkak, I. Clifford

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.)

Paul Scherrer Institut
5232 Villigen PSI
Switzerland

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

10. SUPPLEMENTARY NOTES

K. Tien, Project Manager

11. ABSTRACT (200 words or less)

This report describes a developmental patch applicable to TRACE v5.0 that provides the user with additional flexibility in using the existing option graphlevel to specify the list of variables to be included in the graphics output file (XTV). This option is activated from the TRACE input model using newly added Namelist option lists graphCustId and graphCustVar.

This capability, referred to as the XTV-Customize option, is particularly indicated to optimize storage space and/or post-processing memory space for production studies that require large input models, and for research studies that include sensitivity analysis or uncertainty quantification involving very large number of TRACE simulations of a same input file.

The XTV-Customize option is provided in the form of three separate code installation patches applicable to versions v5.0p5, v5.0p4 and v5.0p3 of the TRACE code, respectively.

The developed patch has been verified on the LCLRS Linux 64-bit servers of PSI for a representative sample of test cases, and for the two different TRACE output graphics file formats, namely XTV and DMX. This patch should nevertheless be still considered as developmental. Some recommendations for improvement of the implementation are provided at the end of the report.

12. KEY WORDS/DESCRIPTORS (List words or phrases that will assist researchers in locating the report.)

TRACE, XTV graphics output file, Namelist, Graphlevel, Customization

13. AVAILABILITY STATEMENT

unlimited

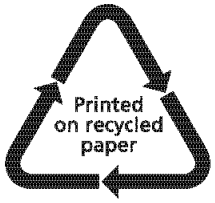
14. SECURITY CLASSIFICATION

(This Page)
unclassified

(This Report)
unclassified

15. NUMBER OF PAGES

16. PRICE



Federal Recycling Program

UNITED STATES
NUCLEAR REGULATORY COMMSSION
WASHINGTON, DC 20555-0001

OFFICIAL BUSINESS