INL/EXT-15-35005

User Guide for the R5EXEC Coupling Interface in the RELAP5-3D Code

J. Hope Forsmann, Walter L. Weaver III

April 2015

INL/EXT-15-35005

User Guide for the R5EXEC Coupling Interface in the RELAP5-3D Code

J. Hope Forsmann, Walter L. Weaver III

April 2015

Idaho National Laboratory

Idaho Falls, Idaho 83415

http://www.inl.gov

**Prepared for the
U.S. Department of Energy
Office of Nuclear Operations
Under DOE Idaho Operations Office
Contract DE-AC07-05ID14517**

User Guide for the R5EXEC Coupling Interface in the RELAP5-3D Code

INL/EXT-15-35005

J. Hope Forsmann

Walter L. Weaver III

April 2015

User Guide for the R5EXEC Coupling Interface in the RELAP5-3D Code

INL/EXT-15-35005

J. Hope Forsmann
Walter L. Weaver III

April 2015

ABSTRACT

This report describes the R5EXEC coupling interface in the RELAP5-3D computer code from the users perspective. The information in the report is intended for users who want to couple RELAP5-3D to other thermal-hydraulic, neutron kinetics, or control system simulation codes.

CONTENTS

1 Introduction ... 1
- 1.1 Background ... 1
- 1.2 Basis of Coupling Methodology ... 2
- 1.3 Subdivision of a Thermal-Hydraulic System ... 3
- 1.4 Coupling Taxonomy ... 3

2 General Considerations ... 5

3 R5EXEC Program ... 6
- 3.1 Virtual Section ... 6
  - 3.1.1 Keyword ‘machine’ ... 6
  - 3.1.2 Keyword ‘wait’ ... 7
  - 3.1.3 Keyword ‘write’ ... 7
- 3.2 Processes Section ... 8
  - 3.2.1 Keyword ‘machine’ ... 8
  - 3.2.2 Keyword ‘writes’ ... 8
  - 3.2.3 Keyword ‘uses’ ... 8
- 3.3 Simulation Section ... 9
  - 3.3.1 Keyword ‘name’ ... 9
  - 3.3.2 Keywords ‘restart time’ ... 9
  - 3.3.3 Keywords ‘start time’ ... 9
- 3.4 Messages Section ... 10
  - 3.4.1 Keywords ‘sends’ and ‘receives’ ... 10
  - 3.4.2 Keyword ‘awaits’ ... 10
  - 3.4.3 Keyword ‘preceeds’ ... 11
- 3.5 Timesteps Section ... 11

4 Examples of Coupled Simulations ... 12
- 4.1 Explicit Coupling ... 12
  - 4.1.1 Parallel Explicit Coupling ... 13
  - 4.1.2 Sequential Explicit Coupling ... 14
  - 4.1.3 Heat Structure Coupling ... 15
- 4.2 Semi-Implicit Coupling ... 16
- 4.3 Kinetics Coupling ... 18
- 4.4 Control Systems Coupling ... 21

5 Coupling Errors ... 22

6 References ... 24

APPENDIX A R5EXEC Input Data Requirements .... A - 1
A-1 Introduction.. A - 1
A-2 Virtual Section of Input Deck .... A - 2
A-3 Processes Section of Input Deck... A - 4
A-4 Simulation Section of Input Deck.. A - 5
A-5 Message Section of Input Deck .... A - 6
A-6 Time Step Information.. A - 8

List of Figures

Figure 1. Schematic for coupled problem domain 26
Figure 2. Schematic for parallel explicit coupling 27
Figure 3. Schematic for semi-implicit and sequential explicit thermal-hydraulic coupling 28
Figure 4. Schematic of PVMEDA parallel explicit test case 29
Figure 5. Listing of PVMEDAX template file 30
Figure 6. Listing of PVMEDA1X template file 31
Figure 7. Listing of PVMEDARX template file 32
Figure 8. Listing of PVMEDAR1X template file 33
Figure 9. Partial listing of output file from ‘parent’ process in PVMEDAX test case 34
Figure 10. Partial listing of output file from ‘child’ process in PVMEDAX test case 35
Figure 11. Schematic of PVMEDSX sequential explicit test case 36
Figure 12. Listing of PVMEDSX template file 37
Figure 13. Partial listing of output file from ‘leader’ process in PVMEDSX test case 38
Figure 14. Partial listing of output file from ‘follower’ process in PVMEDSX test case 39
Figure 15. Schematic of heat structure coupling 40
Figure 16. Schematic of uncoupled test case for semi-implicit coupling 41
Figure 17. Schematic of coupled test case for semi-implicit coupling 42
Figure 18. Listing of template file for ‘pvmcorex’ test case 43
Figure 19. Schematic of TYPPWR test case 44
Figure 20. Schematic of reactor vessel in TYPPWR test case 45
Figure 21. Schematic of coupled model of reactor vessel in kinetics coupling test cases 46
Figure 22. Schematic of Kinetics Zones 47
Figure 23. Listing of template file for pvmptx point kinetics coupling test case 48
Figure 24. Listing of template file for pvmndx for nodal kinetics coupling test case 49
Figure 25. Schematic of uncoupled control systems test case 50
Figure 26. Schematic of coupled control systems test case 51
Figure 27. Listing of template file for coupled control system test case 52

1 Introduction

This document is the fourth in a series of reports that describe the R5EXEC coupling Application Programming Interface (API) and the code coupling system that was built using the R5EXEC coupling API. The first report[1] describes the R5EXEC coupling Application Programming Interface (API). The second report[2] describes the R5EXEC program that controls and coordinates a coupled simulation. The third report[3] describes the implementation of the R5EXEC coupling API in the RELAP5-3D code. This report describes the R5EXEC coupling interface from the users perspective and provides advice and recommendations as to how to use the coupling interface.

1.1 Background

The R5EXEC API and R5EXEC program were developed to facilitate the simulation of a system (e.g., a nuclear power plant) using several different computer programs to describe the transient behavior of the system. Simulation codes are generally written to provide detailed models of some portion of a system, i.e., COBRA[4], RELAP5-3D[5], FLUENT[6], TRAC-PF1/MOD1[7], and TRACE[8] for the fluid systems, CONTAIN[9] and MELCOR[10] for the containment systems, NESTLE[11] and PARCS[12] for reactor power, etc. The R5EXEC API, the R5EXEC program, and the code coupling system that they implement enables the use of different codes for the simulation of different portions of the system in a unified analysis of the transient behavior of the system.

Each code in the coupled simulation needs data from some of the other codes in the simulation. The data are passed between the coupled codes using the Parallel Virtual Machine[13] (PVM) message passing methodology. All of the coupled codes may be executed on a single computer where the data communication between the codes is internal to the single computer or each code may be executed on its own computer where the data communication occurs over the network connecting the computers. The set of networked computers that are to perform a coupled computation becomes a virtual machine. The PVM message passing library contains all of the functionality needed to set up the virtual machine, execute each of the coupled codes on its designated node in the virtual machine, and provide the communication mechanism for data transfer between the nodes of the virtual machine.

An executive program, R5EXEC, has been developed to initiate the ‘virtual’ machine, startup the codes to be used in the coupled simulation, coordinate the data transfers between the coupled codes, and shutdown the virtual machine at the end of the coupled simulation. In addition, the R5EXEC program monitors the status of each computer and each code executing on the virtual machine so that faults can be handled gracefully and so that the virtual machine might be shutdown once the simulation codes terminate, either normally or abnormally. The R5EXEC program was originally developed to couple the RELAP5-3D code to other thermal-hydraulic codes, however any code that implements the R5EXEC API may be used in a coupled simulation.

This report is intended as a user guide and describes how to design, build, and execute a coupled simulation using codes that implement the coupling interface. The different types of coupling will be illustrated using the installation test cases that are supplied with the distribution of RELAP5-3D that includes the coupling methodology. These test cases were developed to verify that the different types of coupling had been implemented correctly by the R5EXEC and RELAP5-3D codes. All of these test cases use two instances of RELAP5-3D to create the coupled simulation.

The general organization of this report is as follows. First there will be a discussion of the basis of the coupling methodology. Then there will be a discussion of the different types of coupling. This will be followed by a discussion of general principles that apply to all forms of coupling. Next will be a discussion of the input for the R5EXEC program. The input file for the R5EXEC program contains all of the information needed to describe the coupled simulation whereas the input files for the coupled programs only describe that portion of the system that they are to simulate. Finally, there will be several sections describing specific examples of the various types of coupling.

1.2 Basis of Coupling Methodology

The basis of the coupling methodology as implemented by the R5EXEC program and by the R5EXEC Application Programming Interface (API) is domain decomposition. The domains that can be separated into separate pieces are the models of the important physical processes that determine the behavior of the system being simulated. The behavior of the separate pieces of the system being simulated are computed by different computer programs or by different instances of the same computer code. The physical processes that can be modeled by different computer codes are the thermal-hydraulic behavior of the fluids in the system, the neutronic behavior of the reactor core, the thermal-mechanical behavior of the solid structures in the system, and the behavior of the control components in the system being simulated. This means that one computer program may compute the behavior of the fluids in all portions of the system while other computer codes compute the neutronic behavior of the reactor core or the behavior of control components in the system.

Additionally, the model of the thermal-hydraulic system can be subdivided into smaller pieces and these pieces can be simulated by different computer codes using different fluid models for their portion of the system, e.g. the primary coolant loop of a reactor may be divided into the reactor vessel with the reactor core and other internals and the primary piping, pumps, and primary side of the steam generators can be lumped into another piece and these two pieces of the primary system simulated by codes using different fluid models e.g., the COBRA-TF code for the reactor vessel and internals and the RELAP5-3D code for the remainder of the primary coolant system. Or the primary and secondary systems may be modeled by one code, e.g. RELAP5-3D, and the containment systems can be modeled by a different computer code, e.g., the CONTAIN code. The control system may also be subdivided into smaller pieces. The neutronic behavior of the reactor core must be modeled by a single code while the thermal-hydraulic model of the reactor core may be subdivided into several pieces and the several pieces can be simulated by different R5EXEC tasks.

In the R5EXEC coupling methodology described in this report, an instance of a computer program executing on a computational node in the virtual machine is called a task. Each task in the virtual machine simulates one of the coupled domains. The R5EXEC program is also considered a task in the virtual machine. Each task in the virtual machine is assigned a unique identifier called the task id by the PVM library software.

1.3 Subdivision of a Thermal-Hydraulic System

Consider the thermal-hydraulic system that is represented by the schematic in Figure 1 (on page 26). This system has been divided into two domains with only two connections between the domains. The volumes in the left domain, volumes 1 and 2, are adjacent to volumes I and II in the right domain with junctions between the adjacent volumes. Figure 2 (on page 27) shows the schematic of the coupled analogue of this system. The two domains have been separated and the junctions between the adjacent volumes in the uncoupled problem have been represented in the coupled simulation in both domains with coupling junctions. Boundary volumes have been added to the ends of the coupling junctions where the volume labeled 1 in the right computational domain is the analogue of volume 1 in the left computational domain. Note that volume 1 in the right computational domain is a boundary component while volume 1 in the left computational domain is in the interior of the left computational domain; this is also true for the volumes labeled 2. The dotted lines in Figure 2 (on page 27) show the information flow between the coupled tasks.

For this type of coupling in which only volume data, i.e. scalar data, are exchanged between the coupled tasks, the orientation of the coupling junctions can be arbitrary and need not be consistent between the two domains because the velocities in the coupling junctions are computed independently for the two domains. Figure 3 (on page 28) shows the schematic for another type of thermal-hydraulic coupling in which data for the coupling junctions are exchanged between the computational domains. The orientation of the coupling junctions must be consistent between the two domains so that if the domain that computes the velocities in the coupling junctions interprets a positive velocity in the coupling junction as flow into its domain as shown in Figure 3 (on page 28), then the domain that receives the velocities must interpret positive velocities as flow out of its domain.

1.4 Coupling Taxonomy

Several types of coupling have been implemented in the R5EXEC program. These types of coupling can be categorized in three ways: first by the computational models that are being coupled, second by the frequency of data exchanges between the computational tasks, and third by the type of solution algorithm being used by the models being coupled together. The computational models that can be coupled are the thermal-hydraulics models, the neutron kinetics models, and the control systems models. Secondly, coupling can be either synchronous or asynchronous. Synchronous coupling is where each task uses the same time step size and data are exchanged every time step. Asynchronous coupling is where data are exchanged at fixed intervals. In asynchronous coupling, explicitly coupled tasks are free to use their own time step sizes subject to the restriction of exchanging data at the correct time. Finally, the coupling may be categorized by the type of solution algorithm used by the models being coupled together. The coupling for the thermal-hydraulic model may be either explicit or semi-implicit. In explicit coupling the data received from another task remain constant during the time step advancements. In semi-implicit coupling, some of the data received from the coupled task are advanced in time while some of the data are held constant during the time advancement.

Not all types of coupling have been implemented in the R5EXEC program. The types of coupling that have been implemented in the R5EXEC program are synchronous and asynchronous explicit thermal-hydraulic coupling, semi-implicit thermal-hydraulic coupling (synchronous coupling by definition), synchronous kinetics coupling, and synchronous control systems coupling. The kinetics and control systems coupling, as implemented, are a form of explicit coupling because the data exchanged between the coupled codes remain constant while being used by the code that receive them.

Explicit thermal-hydraulic coupling can be further subdivided into parallel explicit coupling and sequential explicit coupling. A schematic of parallel explicit coupling is shown in Figure 2 (on page 27) and a schematic of sequential explicit coupling is shown in Figure 3 (on page 28). In parallel explicit coupling, the coupled tasks are peers and the order of the messages between them is arbitrary. In sequential explicit coupling, the leader task, i.e., the task labeled as the right computational domain in Figure 3, must be advanced first and then the follower task, i.e., the task labeled as the left computational domain in Figure 3, can be advanced using data received from the leader task. These roles define an order for the computations and data exchanges. The difference between parallel explicit coupling and sequential explicit coupling is that sequential explicit coupling is strictly mass and energy conserving while parallel explicit may not conserve mass and energy depending upon which variables are exchanged between the codes and when they are exchanged. The mass and energy errors in parallel explicit coupling can be reduced by reducing the size of the explicit coupling data exchange intervals but the mass and energy errors can not be completely eliminated. It is recommended that sequential explicit coupling be used rather than parallel explicit coupling. The only disadvantage of sequential explicit coupling is that the wall clock time for the simulation will be greater than the wall clock time for the equivalent parallel explicit simulation.

There are roles and a defined order of computation and data exchanges for the other types of coupling. In semi-implicit coupling, the two roles are master and slave. The master task, the task labeled as the left task in Figure 3, computes part of its solution and sends the data from the partial advancement to the slave task, the task labeled as the right task in Figure 3. The slave task has been waiting to receive data from the master task and can begin its time step advancement once the data are received. The slave task uses the data received from the master task to begin and finish its advancement. Once the slave task has completed its advancement, it sends its final values back to the master task. The master task then uses the values received from the slave task to finish its advancement. The advancement of the slave task is embedded in the middle of the master task.

The computational sequence for kinetics coupling is arbitrary where the thermal-hydraulic models can be advanced first using the power computed during the last coupled advancement and then the kinetics models can be advanced using the results of the thermal-hydraulic advancement or vice-versa, the kinetics solution may be advanced first, etc. In the kinetics coupling implemented in the R5EXEC program, the thermal-hydraulic models are advanced first and then the kinetics models are advanced using the thermal-hydraulic conditions in the reactor core computed by the thermal-hydraulic advancement because this is the computational order for uncoupled computations in RELAP5-3D. The task performing the kinetics solution is designated as the server task and the tasks using the power data computed by the server task are called the client tasks. The names client and server are used because the server performs the work requested by the client, i.e., compute the reactor power.

Like kinetics coupling, the computational sequence for control systems coupling is arbitrary and the RELAP5-3D computational order has been adopted by the R5EXEC program; first the thermal-hydraulic models are advanced, then the kinetics models are advanced, and finally the control systems models are advanced. Within RELAP5-3D, the order of the data exchanges for control systems coupling is determined by the user with the numbering of the control components. The control components are advanced in numerical order as specified by the user. Care must be taken in numbering the control components so that the computational order of control components in one task that are to send data is the same as the computational order of the control components in the other tasks that are to receive data and vice-versa.

RELAP5-3D has been modified so that it can function in any role in all types of coupling, either as the master task or as the slave task in semi-implicit thermal-hydraulic coupling, as either the server task or client task in kinetics coupling, and as the leader task or the follower task in explicit sequential thermal-hydraulic coupling. The tasks in the other types of coupling are peers and there is no defined role for these types of coupling.

2 General Considerations

The most important decision in the design of a coupled simulation is the determination of which computer codes to use in the simulation. The codes should be chosen based on the type of transient being simulated as well as on the capabilities of the codes being considered. For example, in the simulation of a Large Break LOCA transient where reflood phenomena are important, a code like COBRA-TF might be chosen to simulate the behavior of the reactor core, the reactor vessel, and its internals and the RELAP5-3D code might be chosen to simulate the behavior of the remainder of the primary system including piping, pumps, and primary side of the steam generators as well as the secondary side of the stream generators and feedwater system. In addition, a code like the CONTAIN code might be used to compute the response of the reactor containment to the addition of mass and energy to the containment due to the blowdown of the reactor primary system through the break.

If more than one code is chosen for the simulation of a thermal-hydraulic system, e.g., the primary coolant system, then the next choice is where to divide the system and which parts of the system are to be simulated by which code. This decision will be guided in part by the choice of the code to be used in the coupled simulation, the capabilities of the codes themselves, and by the type of transient being simulated. For example, the COBRA-TF code uses a three field model of fluid flow whereas the RELAP5-3D code uses a two fluid representation of fluid flow. The parts of the system to be simulated by COBRA-TF should require the three field fluid model for accurate simulation of the important phenomena in the transient being simulated and RELAP5-3D should be used for that portion of the system only requiring a two field fluid model.

When RELAP5-3D is being used in a coupled simulation, the input files being used must be complete models of that portion of the system each RELAP5-3D task is to simulate. This allows the RELAP5-3D tasks to be executed individually without being coupled to the other codes. This is to facilitate the initialization of a coupled computation where the individual RELAP5-3D tasks can be initialized separately before performing a global initialization using the coupling methodology. The initialization of other simulation codes may require them to be initialized in coupled mode depending upon how the coupling has been implemented in these codes.

The next important decision to be made is the type of coupling to use between the several codes. If the parts of the system being simulated by the different codes are tightly coupled, i.e., have similar response times, then synchronous coupling should be used. If the parts of the system being simulated by the different codes are not tightly coupled, then asynchronous coupling can be used. Based on these considerations, semi-implicit coupling (a type of synchronous coupling) would be recommended for coupling COBRA-TF and RELAP5-3D for the simulation of the reactor core, reactor vessel and primary coolant system in order to preserve the numerical stability of the resulting coupled calculation and explicit, asynchronous coupling would be recommended for the coupling of RELAP5-3D to the CONTAIN code for the simulation of the containment response. Because the response of the containment depends upon the mass and energy input to the containment through the break, the sequential subtype of explicit asynchronous coupling would be recommended for coupling RELAP5-3D to the CONTAIN code because it conserves mass and energy between the two codes.

3 R5EXEC Program

The R5EXEC program is used to initiate, control, and terminate the execution of a coupled computation on a virtual machine. The input file for the R5EXEC program contains all of the information about the coupled computation whereas the input files of the coupled codes only contain the information needed to simulate their respective portions of the coupled computation. There are five sections in the input file for the R5EXEC program. These sections are delimited by the keywords 'virtual', 'simulation', 'processes', 'messages', and 'timesteps'. Comments may be included in the input file and are marked by a sharp, i.e., '#', at the beginning of the line. Lines (or cards for older users) may contain up to 256 characters. Figures 5,6,7, and 8 contain examples of input files for the several types of coupling. These files will be discussed in detail in their appropriate sections.

Each section of the input file begins with the reserved keyword by itself on a line in the input file. The lines that follow a keyword are considered part of that section until another keyword is encountered in the input file. The data in the input lines are interpreted as blank delimited words. Some of the data items entered by the user in one section of the input deck become user defined keywords in subsequent sections of the input file. All reserved keywords in the R5EXEC input file must be in lower case. All other character data in the R5EXEC input file can be in either case as appropriate. For example, the command line parameters for RELAP5-3D must be in lower case but the name of the RELAP5-3D executable file can be in either case.

3.1 Virtual Section

The 'virtual' section of the input file is optional and, if present, describes the computers that are to be used in executing the coupled simulation. If this section is absent, the virtual machine consists of the computer executing the R5EXEC program. There can be three types of input lines in the 'virtual' section of the input file. The first type of input line is required if this section is included in the input file. This type of line begins with the name of a computer that is to be used in the coupled simulation. More than one line of this type can be included in the input file. The names of the several computers become user defined keywords in the 'processes' section of the input file. The other two types on input lines begin with the keywords 'wait' and 'write' respectively. The lines with the keywords 'wait' and 'write' are optional. Each type of input line will be described in the following subsections.

3.1.1 Keyword 'machine'

The 'machine' keyword is a user define keyword and multiple 'machine' keywords may be defined in the 'virtual' section of the input deck. The values of the 'machine' keywords are the names of the computers that are to be used in the coupled simulation. Each computer is described on a single line with the network name of the computer appearing first followed by keyword parameters containing options for the virtual machine.

Commonly used options are wd and ep where each option is followed by a user define directory. The wd option specifies the working directory that contains the input files for the simulation codes and is the default location for any output files from the simulation code. The ep option specifies the execution path where the executable file for the simulation code is located. Other parameters may be found in the PVM documentation. If the wd and ep options are omitted, the default working directory and default execution path are the directory from which the R5EXEC program is executed. For options that specify directories, each option may specify a different directory. Figure 5 shows the template file for one of the installation test cases. The character string ‘MACHINE’ in Figure 5 has been used as a placeholder for the actual machine name in this template. The character string ‘WHERE’ is a placeholder for the user defined location of a directory specified by the wd and ep options. The placeholders are modified by the RELAP5-3D installation scripts to insert the name of the computer used for the installation of RELAP5-3D and the directory from which the installation script executes the coupled installation test cases.

3.1.2 Keyword ‘wait’

There are two optional parameters that may be included in the ‘virtual’ section of the R5EXEC input deck. The first option uses the keyword ‘wait’ and is followed by a real number defining the global wait time. The global wait time is the number of seconds that any task must wait to receive a message from another task, including the R5EXEC task.

All data communication between tasks uses a handshake protocol in which a task sends a message to another task and then listens to receive an acknowledgment from the task to which it sent the message verifying that the other task has received the message. Alternately, a task listens to receive a message from another task and then sends an acknowledgment to the task that sent the message verifying to the sender that the message had been received. If a message has not been received within the wait time when waiting to receive a message or an acknowledgment has not been received during the wait time after sending a message, the simulation is terminated after writing a timeout error message. The timeout avoids deadlock situations where either all codes have sent a message and are waiting to receive an acknowledgment or where all codes are listening to receive a message that never arrives. The default global wait time is -1.0 seconds and is used by all tasks if not specifically overwritten later in the input file where a wait time of -1.0 means that all codes will wait indefinitely.

3.1.3 Keyword ‘write’

The other optional parameter uses the keyword ‘write’ and is followed by a filename. When a task is executing in the virtual machine, any output that would normally be written to the computer screen (called stdout in UNIX/LINUX) is collected in a file named pvml.<user number> located on the /tmp directory where <user number> is the id number of the person executing the R5EXEC program. The filename and location are for UNIX/LINUX type operating systems. Consult the PVM documentation for the name and location of this file on WINDOWS operating systems. The user may specify another directory for the pvml file instead of the /tmp directory by specifying a directory using the PVM_TMP environmental variable. Alternately, the user may specify either a relative path name or full path name for this output file using the ‘write’ keyword. This file, however specified, will be used for the stdout from all coupled codes unless different files are specified for each code in the ‘processes’ section of the input file. See Figure 5 (on page

30) for an example of specifying the global stdout file.

3.2 Processes Section

The next section of the input file for the R5EXEC program is the ‘processes’ section. This section of the input file specifies the several codes that are to be used in the coupled simulation. This section is divided into subsections delimited by the network names of the computers specified in the ‘virtual’ section.

3.2.1 Keyword ‘machine’

Each subsection begins with a user defined ‘machine’ keyword that was defined in the ‘virtual’ section of the input file. Each subsection contains lines specifying the codes to be executed on that computer. Within each subsection, each code is described on a single line where the code described on each line becomes a task in the virtual machine. The line begins with a descriptive name for the task, followed by the type of time step control for the task (either synchronous or asynchronous), followed by the name of the executable file for the particular simulation code, finally followed by any command line parameters needed by that particular simulation code. There should be a subsection for each of the computers listed in the ‘virtual’ section of the input file. The descriptive name for each executing code becomes a user defined keyword in the ‘messages’ section of the input file.

3.2.2 Keyword ‘writes’

The output that would normally be written to the global stdout file specified in the virtual section of the input file may be directed to individual files by including lines in each subsection of the tasks section containing the keyword ‘writes’. These lines are optional and begin with the descriptive name of a task followed by the keyword ‘writes’ followed either the relative pathname or the absolute pathname of the file that is to contain the output from the code named by the first word on the line. Files described by a relative pathname will be written to the path beginning at the working directory.

3.2.3 Keyword ‘uses’

Finally, another optional line includes the keyword ‘uses’. This line begins with the descriptive name of a task followed by the keyword ‘uses’ followed by an integer followed by the reserved keyword ‘threads’. This line is used to specify the number of threads that the task is to use when and if the task has been programmed to use multiple threads of execution in its computations. If this line is absent for a given task, the default value of zero is sent to the task. A value of zero indicates that the task should decide for itself how many threads to use for its computations. RELAP5-3D has been programmed to use multiple threads of execution as indicated by the inclusion of OpenMP directives in the code source with the maximum number of threads set to minimum of the number of CPUs in the computer on which the task is executing and a value of 4.

3.3 Simulation Section

The next section of the input file is the ‘simulation’ section of the input file. This section begins with the keyword ‘simulation’ on a single line. This section of the input file is optional as there are default values for the parameters specified in this section. Each parameter is specified on a single line and each line begins with a keyword.

3.3.1 Keyword ‘name’

The first keyword in this section of the input deck is ‘name’ and is followed by a character string of up to 80 characters in length. This string is written to the restart files of the coupled codes for initial simulation runs and is compared to the string written on the restart files for restart runs. If the ‘name’ string in a restart file is not the same as the ‘name’ string in the R5EXEC input deck for a restart run, a warning message is written to the output file of the simulation code and the restart run continues as if the names matched. If this string is not included in the input file for the R5EXEC program, a string of 80 blank characters is used as the default simulation name.

3.3.2 Keywords ‘restart time’

The second keyword is ‘restart time’ and can be used to specify the simulation time at which to begin a restart run. This parameter is optional and if absent, a default value of 0 is sent to coupled codes. This indicates that an initial run is intended. A non-zero value indicates a restart run that is to start from the time specified. A value of -1.0 for the restart time designates that the restart is to occur from the last restart record on the restart file. As implemented in RELAP5-3D, the mode indicated by the value of the restart time, either an initial or a restart run, overrides the mode set in the RELAP5-3D input deck. A warning message is written to the RELAP5-3D output file if the modes do not match and code execution continues. A mismatch in the modes between the mode specified in the R5EXEC input file and the input files of the coupled codes may cause subsequent errors and code failure.

3.3.3 Keywords ‘start time’

The third keyword is ‘start time’. This optional parameter can be used to set the start time of an initial simulation or to reset the simulation time in a restart run. The default start time for an original run is zero seconds, and the default start time for a restart run is the restart time. In RELAP5-3D, the start time is reset to 0 when performing a restart run and switching from steady state mode in the restart file to transient mode for the restart. The start time option can be used to ensure that all of the coupled codes use the same start time. If this parameter is omitted from the input deck, the default start time is used.

3.4 Messages Section

The next section of the input file is the ‘messages’ of the input file. This section of the input file specifies the data that are to be exchanged between the codes executing on the ‘virtual’ machine. This section of the input file is subdivided into subsections beginning with the keywords ‘explicit’ for explicit coupling, ‘semi-implicit’ for semi-implicit coupling, ‘kinetics’ for kinetics coupling, and ‘control system’ for coupling of control system models. The input lines in each subsection specify the data that are to be exchanged between pairs of tasks. The names of the tasks are the user defined keywords that were specified in the ‘processes’ section of the input file. Each line begins with the name of a task as defined in the ‘processes’ section of the input file, followed by the keyword ‘sends’, ‘receives’, ‘awaits’, or ‘preceeds’ followed by the name of another task. Lines containing the keywords ‘sends’ or ‘receives’ specify the data to be exchanged in a message between the named tasks and each message is given a unique identifier by the R5EXEC program called a message tag.

3.4.1 Keywords ‘sends’ and ‘receives’

The lines that use the keyword ‘sends’ or ‘receives’ are terminated with a string (a series of blank delimited words) that defines the data that are to be sent or received. For each line that specifies data to be sent from one task to another there must be a corresponding line that describes how the data that are received by the second task are to be interpreted by the task receiving the data. The order of the names of the two tasks in the receive specification is the reverse of the same two task names as defined in the send data exchange specification. The first send data exchange specification is paired with the first receive data specification with the same pair of task names as the send specification. Each pair of data specifications define a single message that is to be exchanged between the tasks, a data exchange being defined as a pair of messages: the message containing the data followed by the acknowledgment. The string defining the data to be sent or received is sent by the R5EXEC program to the sending or receiving code and the data in the string must be understood by the code receiving the data specification string from the R5EXEC program.

This document describes the format for the data specification string that has been implemented by the RELAP5-3D code. The data specification string for RELAP5-3D consists of pairs of words, each pair consisting of a character string containing the name of a RELAP5-3D variable followed by a fully qualified volume, junction, or component number, or a character string containing a component name as found in the input deck for that particular RELAP5-3D task followed by a volume or junction number within the component named. When a RELAP5-3D component name is used in the data specification, RELAP5-3D replaces the component name with a internally defined list of variable names, where the variable names used depend upon whether a volume component or junction component was named. Each code has its own way of specifying the data items in a particular message. The total number of individual data items specified in a ‘sends’ data specification, either data items named specifically using variable names or data items named implicitly using a component name, must be the same as the total number of items named in the corresponding ‘receives’ data specification.

3.4.2 Keyword ‘awaits’

In addition to the data specification lines in the messages section of the input file, additional lines containing the keyword ‘awaits’ may be included. This line contains a pair of task names separated by the keyword ‘awaits’ and the second task name is followed by a real number that defines the wait time for messages that are to be sent from the second task to the first task for either a data message or an acknowledgment. Two lines containing the same pair of task names but with the names reversed in the second line may be entered but with different wait times. These lines define the wait times for individual messages and allow the wait time for a data message to be different from the wait time for its acknowledgment.

3.4.3 Keyword ‘preceeds’

Finally, one additional line may be included in the message specification section for explicit coupling. This line contains the keyword ‘preceeds’ separating a pair of task names. This line specifies that the pair of tasks named are using sequential explicit coupling and the first task named is the leader in the sequential explicit coupling and that the second task named is the follower in the sequential explicit coupling. If this line is not entered for a pair of tasks, parallel explicit coupling is assumed.

3.5 Timesteps Section

The last section of the input file is the ‘timesteps’ section. This section of the input file specifies the time steps that are to be used for synchronous coupling or the data exchange intervals for asynchronous coupling. There can be any number of time step cards and each input card defined an interval of time during the simulation. The cards use the same data as the RELAP5-3D time step cards for specifying the intervals. Each card begins with the end time of the interval, followed by the minimum and maximum time step sizes allowed during the interval. These items are followed by a packed word specifying whether extra printed output should be produced by synchronously coupled codes, finally followed by four integers that specify the frequency of minor edits and writes to the plot file, the frequency of major edits, the frequency of restart writes, and the frequency of explicit data exchanges. The packed work is equivalent to the ‘dtt’ bits of the control work on the time step cards for RELAP5-3D (See Section 3 of Appendix A of Volume II of the RELAP5-3D manual).

The output frequencies are converted to time intervals by multiplying them by the maximum time step size. The output times for minor edits, major edits, and restart writes determined by these lines supersede the output times determined from the information contained in the input decks of the coupled codes for synchronous coupling. Also, the maximum and minimum time step sizes as well as the packed word defining extra output on these cards supersede the internally defined values for synchronously coupled codes. Only the restart write times supersede the internally defined output times for asynchronously coupled codes (the explicit exchange intervals are not determined locally). The end of interval time on the last line in this section defines the end time of the simulation. The input lines should be entered with increasing end time on each card.

The output intervals are converted into time targets that define points in time that the codes should reach by the adjustment of their time step sizes. The R5EXEC program performs the time step adjustments for synchronously coupled codes and the time step is adjusted locally for asynchronously coupled codes. It is recommended that the end of interval time on each time step card be an even multiple of the maximum time step size in each time interval and that the end of interval times and the maximum and minimum time step sizes in each interval be the same for the R5EXEC program and all of the coupled codes. This is recommended because the time step selection logic in the R5EXEC program (for synchronously coupled codes) and in RELAP5-3D (for asynchronously coupled codes) will reduce the time step size if the time targets are too close together where RELAP5-3D does not tolerate large time step reductions very well.

4 Examples of Coupled Simulations

This section of the report discusses examples of the several types of code coupling using the installation test cases that are included as part of the distribution of the RELAP5-3D code. Each test case demonstrates some aspect of the coupling methodology and consists of two instances of RELAP5-3D coupled together. There is an uncoupled analogue for some of the coupled test cases so that the results of the coupled and uncoupled test cases can be compared to verify that the coupled test case reproduces the results of the uncoupled test case. Each coupled test case utilizes three input decks, one input deck for the R5EXEC program and an input deck for each of the two instances of RELAP5-3D.

The input deck for the R5EXEC program is built using a template file that is modified to include the name of the computer on which the test case is being executed and to include the working directory and execution path from which the R5EXEC program is being executed. The string 'MACHINE' in the template file is replaced by the name of the computer where the test case is being executed and the string 'WHERE' is replaced by the name of the directory containing the input file (See Figure 5 (on page 30) for an example of a template file). The output files will be written to the working directory. It is assumed that the RELAP5-3D executable is located in the same directory. This is not required by the PVM methodology and the working directory and the execution path may be different at the user's discretion. In these examples all input file names begin with the string 'pvm' followed by a three to five character string describing the test case followed by a single letter, 'x' for the input files for the R5EXEC program, and other single letters for the RELAP5-3D input decks that describe their roles in the coupled test case. The input files have the file extension 'i' designating an input file and the template file has the extension 'ii'.

4.1 Explicit Coupling

Explicit coupling is characterized by the fact that the data that are received by a coupled code are held constant during the time step advancements of that code until new data are received. Explicit coupling can be either synchronous where data are exchanged each and every time step or it can be asynchronous where data are exchanged at fixed intervals. Synchronous coupling means the analysis programs use the time step sizes calculated by R5EXEC whereas asynchronously coupled codes may choose their own time step size independently so that each one of the asynchronously coupled code may use a different set of time steps to arrive at the same point in time, i.e., the time target, to perform the next data exchange.

Two forms of explicit coupling have been implemented using the PVM coupling methodology, i.e., parallel explicit coupling and sequential explicit coupling. In parallel explicit coupling, the coupled tasks are peers and can be advanced in time in any order. In sequential explicit coupling, one task, the leader task, must be advanced first over the coupling interval to compute the fluid flow rates in the coupling junctions between the two domains. Once the leader task has computed the flow rates in the coupling junctions, it sends the flow rates to the follower task which then uses the flow rates as its boundary condition and advances through the coupling interval. Sequential explicit coupling is guaranteed to conserve mass and energy between the two coupled codes because the flow rates in one code are the same as the flow rates in the other code during the same coupling interval. Mass and energy are not conserved in the simplest implementation of parallel explicit coupling because the flow rates computed in the coupling junctions in the separate input decks of the two coupled tasks that represent the connection between the coupled domains are computed independently by the two tasks. The results of these independent computations will be different even if the geometry of the junctions are the same in the two tasks and the same code is used to simulate each portion of the coupled system. This is a consequence of the fact that in one task the upstream pressure for the coupling junction will change during the time step advancements because the upstream volume is in the interior of the computational domain for that task while the downstream pressure, being the value received from the other task, will remain constant during the advancements. In the other task it is the upstream pressure that remains constant over the advancements while the downstream pressure changes. This difference in the time levels for the pressures used in the computation of the flow rates in the coupling junction ensures that the flow rates computed by the two coupled codes will be different.

4.1.1 Parallel Explicit Coupling

There are several installation test cases that use parallel explicit coupling. All of these test cases use asynchronous coupling. The pvmeda series of test cases are based on the edhtrk (Edward's pipe blowdown) uncoupled test case where the 20 volume pipe in the uncoupled test case was divided into two 10 volume pipes for the coupled test cases. Figure 4 (on page 29) shows a diagram of both the uncoupled test case and its coupled counterpart. The dashed arrows show the data exchanges between the two codes. The schematic shows that the junction between the two halves of the test section in the uncoupled test case, i.e. the junction between volumes 10 and 11 in the uncoupled test case, is represented by two junctions in the coupled test case, junction 4 in the 'child' input deck and junction 104 in the 'parent' input deck. The junctions must be oriented in the same direction because each task must interpret a positive velocity in the junction in the same way, i.e., as out of the 'child' domain and into the 'parent' domain. If the velocities in the junction are not being exchanged between the two tasks, the junctions can be oriented independently. This test case also illustrates that a component, either a volume or a junction, whose conditions are sent to another task is an 'active' component whose fluid conditions are computed by the solution of the conservation equations. In contrast, the components that receive conditions from another coupled task are 'passive' or boundary components, whose conditions are specified by the user for uncoupled problems.

The test cases include two initial runs and two restart runs. There are three input decks for each test case: the input deck for the R5EXEC program and the two RELAP5-3D input decks. The input decks for the R5EXEC program are built using template files, Figures 5 through 8 (see pages 30 through 33). The two initial runs demonstrate the use of the default start time and the specified start time options in Figure 5 (on page 30) and Figure 6 (on page 31) respectively. The two restart runs demonstrate a restart from a specified restart time during the initial run and a restart from the last restart record on the restart file generated by the initial run, Figure 7 (on page 32) and Figure 8 (on page 33) respectively. Figures 7 and 8 for the restart runs show that the entire R5EXEC input file must be entered for a restart run. This is in contrast to RELAP5-3D where only a minimal input file is needed for a restart run because most of the required data are read from the restart file. The R5EXEC program does not produce a restart file so the user must re-enter all of the data needed for the restart run.

The template files are modified during the installation task to include the name of the computer being used to execute the coupled test case and to include the location of the working directory and execution path for the test cases. The template file assumes that the working directory and the execution path are the same. Figure 5 shows the template file for the pvmedax test case. This is an initial run and is the basis for the two restart runs. The template file for the other initial run, pvmeda1x, is shown in Figure 6. The template files are identical except for the start time option. Test case pvmedax uses a default start time of zero seconds, and the pvmeda1x test case uses a specified start time of 0.0001 seconds.

Figure 5 shows that the descriptive names of the two coupled instances of RELAP5-3D, lines 9 and 11 in the 'processes' section of the input files are user defined keywords in the 'messages' section of the input file that describe the data exchanged between the two codes. The 'messages' section of the input file shows that there can be different numbers of messages sent from one coupled code to another coupled code and that the messages may contain different numbers of data items. The task named 'parent' sends four messages to the task named 'child' whereas the task named 'child' only sends two messages to the task named 'parent'. The send and receive message lines have been listed together so that the number of data items can be compared and so that the user might more easily compare the specifications for the individual data items. Line 15 in Figure 5 shows that the first data item in the first message that the 'parent' task sends is to be the pressure in volume 1 of component 103 and line 16 shows that the 'child' task is to interpret the first data item in the first message received as the pressure in volume 1 of component 5. The two data items use the same format for specifying the data item because the two coupled codes are both instances of RELAP5-3D. If one of the coupled codes had not been RELAP5-3D, the character string specifying pressure might be the string 'pres' instead of 'p' and the location specifier might be the number 10301 or the string 1,103 instead of the number 103010000. The point is that the data specifiers are simulation code specific and have meaning only to that particular code.

The absence of the keyword 'preceeds' in the tasks section of this file designates the coupling is parallel explicit coupling rather than sequential explicit coupling. It is recommended that the specification for a receive message should follow immediately after the specification for the corresponding send message so the number of data items in each specification can be compared and checked that the data items being sent are correct and are being interpreted correctly in the receive message.

The input file shown in Figure 5 shows the use of the 'write' and 'writes' keywords to redirect the standard output from the R5EXEC code and the two instances of RELAP5-3D from the default output file to individual files.

Figure 9 (on page 34) and Figure 10 (on page 35) show a portion of the RELAP5-3D output files for the 'parent' and 'child' tasks. These figures show the description of the data items in each message sent or received by each code along with the message tag that is assigned to each message by the R5EXEC program. This printout provides another opportunity for the user to verify that the messages contain the information intended.

4.1.2 Sequential Explicit Coupling

There are two test cases that illustrate the use of sequential explicit coupling; pvmedsx and pvmeds10x. These two test cases are similar to the pvmeda test cases and are identical except for the explicit coupling exchange frequency. Test case pvmedsx exchanges data with a frequency of one and pvmeds10x exchanges data with a frequency of ten. The data exchange frequency should be chosen so the flow rates do not change too much over the coupling interval where 'too much' depends upon the transient being simulated and the tightness of the coupling between the two domains. The geometry and components are the same as the pvmeda test cases. Figure 11 (on page 36) shows a diagram of the test cases where the arrows show the data exchanges. It is unfortunate that the leader task, i.e., the task that 'preceeds' the other task, is named 'leader' and the follower task is named 'follower' because it conflates the name of the tasks with their roles in the coupling. The leader task sends junction data to the follower task and the follower task sends volume data to the leader task. This is in contrast to the pvmedax test case in which both coupled codes send volume data to the other code.

Figure 11 (on page 36) illustrates an important difference between sequential and parallel explicit coupling. In sequential explicit coupling, the coupling junction in the two coupled tasks, junction 104 in the leader task and junction 4 in the follower task, must observe the same junction orientation. If the coupling junction is oriented so that positive flow in the junction is OUT of the computational domain in one coupled task, then the coupling junction in the other task MUST be oriented so that positive flow in the junction is INTO its computational domain. Alternately, if the coupling junction is oriented so that positive flow in the junction is INTO the computational domain in one task, then the coupling junction in the other task MUST be oriented so that positive flow in the coupling junction is OUT of its computation domain. The R5EXEC coupling system does not impose any convention in the orientation of the coupling junctions in sequential explicit coupling except that the junction orientation be consistent. It is the responsibility of the user to ensure that the orientations are consistent in the input decks of the two coupled tasks. In contrast, the implementation of semi-implicit coupling imposes a sign convention on the orientations of the coupling junctions that will be explained in Section 4.2.

Figure 12 (on page 37) lists the input deck for the pvmedsx test case. The input deck is very similar to the input deck for the pvmedax test case except for the 'messages' section. First, the 'messages' section contains the keyword 'preceeds' to designate that this test case uses sequential explicit coupling instead of the default parallel explicit coupling. Secondly, the data specifiers in the send and receive messages use component names and volume or junction numbers within the component instead of specifying individual data items. Component names and volume or junction numbers within the component MUST be used for sequential coupling but their use is optional in parallel explicit coupling where individual variable names and locations can be used. The component names are shown in parentheses below the component type and number in Figure 11. Component 5 in the follower task is shown using a dashed line to indicate that it must be present in the RELAP5-3D input deck to form a complete system for RELAP5-3D but it is not used in the solution of the conservation equations for the follower task. RELAP5-3D expands a component name into a internally defined list of variables for that component where the list is different for volume components and junction components. Figure 13 (on page 38) and Figure 14 (on page 39) show a portion of the RELAP5-3D output files for this test case. The figures show the result of the expansion of the component name and number into a list of variables.

4.1.3 Heat Structure Coupling

The coupling that has been discussed in the previous sections involves dividing a thermal-hydraulic system, that is, a self-contained network of volumes and junctions, into two or more pieces and simulating the separate pieces in different coupled codes. Heat structure coupling involves coupling separate thermal-hydraulic systems using the heat flux through the solid structures whose surfaces contact fluids in different fluid systems. For example, the primary side of a steam generator is connected to the secondary side of the steam generator by the heat flux through the tubes in the steam generator. The primary coolant loops of a reactor system are coupled to the containment by the heat fluxes through the reactor vessel walls and the walls of the coolant piping. This type of coupling can be accomplished by modeling the heat structure that connects the two systems in the input decks of the coupled codes. These heat structures touch the fluid in their respective computational systems on only one side. The heat flux and the temperature on the surface of the heat structure that touches the fluid are computed by the internally coupled heat conduction and thermal-hydraulic model if a boundary condition can be supplied to the other side of the heat conductor by the other R5EXEC coupled code. One code uses a surface heat flux as its boundary condition and the other code uses the surface temperature as its boundary condition. Either parallel explicit coupling or sequential explicit coupling can be used to exchange the heat flux data and temperature data between the coupled codes. However, if sequential explicit coupling is used, it is recommended that the leader task compute and send the surface heat flux to the follower task and that the follower task send the surface temperature to the leader task so that energy would be conserved between the two tasks.

Figure 15 (on page 40) shows a diagram of heat structure coupling through a tube wall in an uncoupled simulation and in the equivalent coupled simulation. The hashed area represents the heat structure. In the uncoupled simulation, fluid touches the heat structure on both surfaces and convective boundary conditions are used on both sides of the heat structure. In the coupled simulation the fluid in the primary task only touches the fluid on one side of the heat structure, the inside of the tube for a steam generator tube, and the fluid in the secondary task only touches the fluid on the opposite side of the heat structure, the outside of a steam generator tube. A convective boundary condition can be used for the surfaces that touch the fluid and the boundary condition must be specified, i.e., specified by the user or by another task, on the other side of the heat structure. In this case, the boundary condition must be specified on the outside of the tube in the primary task and on the inside of the tube in the secondary task.

The choice of which boundary condition to use for each task, either a surface heat flux or a surface temperature, is up to the user, and both boundary conditions can be the surface temperature, or both boundary conditions can be the surface heat flux, or a combination of surface temperature or surface heat flux can be used. It is recommended that one task use a surface heat flux and the other task use a surface temperature. This ensures that the energy out of one fluid system into the heat conductor is the same as the heat flux into the heat conductor in the other system, thus conserving energy. In RELAP5-3D the surface heat flux and surface temperature can be accessed directly for sending to a coupled task but RELAP5-3D cannot use the values received from another task as boundary conditions for its heat structures. The values received from the other task must be stored in a general table since only general tables can be used to specify a non-convective boundary condition for a heat structure in RELAP5-3D.

4.2 Semi-Implicit Coupling

The other type of thermal-hydraulic coupling is semi-implicit coupling. Semi-implicit coupling is a type of synchronous coupling. All of the tasks participating in semi-implicit coupling must be defined as synchronous tasks in the 'processes' section of the input file of the R5EXEC program. Semi-implicit coupling is characterized by the fact that changes in the conditions in one of the coupled domains during its time step advancement are felt in the other coupled domain during the same time step. This tight coupling of the conditions in the two domains eliminates the sonic Courant stability limit that is present in explicit thermal-hydraulic coupling.

The data to be exchanged between the two domains are specified in the section of the input file for the R5EXEC program that begins with the keyword 'semi-implicit'. The data is defined by specifying the component name and the volume or junction number within that component for the coupling of

RELAP5-3D to another code. The method of specifying the data that the other code is to send to RELAP5-3D or receive from RELAP5-3D is code specific and may be specified in another manner. The input processor in RELAP5-3D expands the component name and number into a predefined list of data items. The list of items for volumes is different from the list of items for junctions. One task sends volume data to the other task and the other task sends junction data back to the first task. The task sending volume data is designated the master task and the task sending junction data is designated the slave task. The slave task is responsible for computing the flow rates in the coupling junction using the conditions in the coupling volume that are received from the master task.

In RELAP5-3D the simulation model as defined in the input deck is divided into self-contained fluid systems such as the primary coolant system and a number of secondary coolant systems. Any one of the fluid systems in a RELAP5-3D simulation may only be connected to a single other task and coupling of a fluid system in RELAP5-3D to multiple tasks is not allowed, i.e., there can only be one 'master' task and one 'slave' task for a fluid system. However, one fluid system in RELAP5-3D may act as the master task in coupling to another task and a different fluid system in the same RELAP5-3D model may function as the 'slave' task when coupling. Figure 17 (on page 42) shows a schematic of a semi-implicitly coupled test case. The lower and upper core components in the master task are placed in the same fluid system by RELAP5-3D because they are connected through the bypass component. If the bypass component was removed, RELAP5-3D would normally put the lower core and upper core components into different fluid systems. However, because both of these components are coupled to the same task, i.e., the slave task, they are placed into the same fluid system by RELAP5-3D.

Figure 16 (on page 41) shows a schematic for the uncoupled version of the 'pvmcore' installation test case and Figure 17 (on page 42) shows the schematic of the coupled version of the same test case. This test case represents a heated reactor core channel and an unheated bypass channel connected between upper and lower plenum volumes. The dashed lines in Figure 17 illustrate the data exchanges between the two tasks. Figure 18 (on page 43) lists the template file for the input deck to the R5EXEC program for this test case.

The coupling boundary components in the master task (the time dependent volume and attached single junction in Figure 17) are passive components because the conditions in these components are computed by the slave task and are sent to the master task. The coupling boundary components can be either time dependent junctions and time dependent volumes whose conditions are normally supplied by the user in tables or they can be single volumes and single junctions. Single volumes and single junctions are normally active components, i.e. their conditions are computed as part of the solution algorithm, but they are converted into passive components in coupled simulations if their conditions are received from another task. Any combination of these coupling boundary components can be used in the master task.

The coupling boundary components in the 'pvmcorep' input deck, the input deck for the master task in the 'pvmcore' test case, are time dependent volumes and single junctions as shown in Figure 17. The coupling boundary components in the pvmnds.i input deck (the input deck for the master task in the 'pvmnd' coupled test case that is a test case for combined semi-implicit thermal-hydraulic coupling and nodal kinetics coupling) are also a combination of time dependent volumes and single junctions (See Figure 21 (on page 46)). The coupling boundary components in the 'pvmnonc' coupled test case (not discussed in this report) are time dependent volumes and time dependent junctions in the master task. These test cases illustrate the different combinations of volume and junction components that may be used as coupling boundary components when RELAP5-3D is used for the master task of semi-implicit coupling. It is recommended that the coupling boundary components in the master task be time dependent volumes and time dependent junctions to emphasize that these components are passive components.

The coupling boundary components in the slave task must be active components, i.e., they must be single volumes and single junctions whose conditions are computed as part of the time step advancement when RELAP5-3D is used for the slave task. The conditions in the coupling boundary components in the slave task are sent to the master task after they are computed during the time step advancement in the slave task. If a task is functioning as the master task in the coupling of a computational system, it can only send volume data to its corresponding slave task and if a task is functioning as the slave task, it must only send junction data. A mixture of volume data and junction data in the messages that the master task sends to or receives from its slave task is not allowed and vice-versa. The three test cases illustrate all of the combinations for the coupling boundary components for the master task in semi-implicit coupling.

The only requirement that is imposed on the input decks of the coupled tasks is that the coupling junctions in the master task be oriented such that a positive flow rate in the junctions indicates flow INTO the domain represented by the master task and conversely, that the coupling junctions in the slave task be oriented such that positive flow in the coupling junction represents flow OUT of the domain represented by the slave task. This is shown in Figure 17 where the arrows representing the coupling junctions show the proper orientation. This is different from the requirement for junction orientation in sequential explicit coupling where it is only required that the orientation of the junctions be consistent between the coupled codes.

Finally, the input decks for RELAP5-3D semi-implicitly coupled codes must be valid input decks, capable of uncoupled execution. The coupling boundary volumes in the master task shown in Figure 17 are shown in dashed lines to indicate that their conditions are never used in coupled problems but that they must be included in the RELAP5-3D input deck to satisfy the RELAP5-3D input processor. If these coupling boundary volumes are not present in the RELAP5-3D input deck, the code will terminate at the end of input processing with an input error.

4.3 Kinetics Coupling

Kinetics coupling is a specialized type of explicit coupling in which one of the coupled codes computes reactor power using either a point kinetics or nodal kinetics model and the other coupled codes use the computed reactor power in their computations. The code performing the kinetics computation is designated as the ‘server’ task and the other codes are designated as the ‘client’ tasks. Kinetics coupling may be either synchronous or asynchronous but only synchronous kinetics coupling has been implemented by the R5EXEC program at this time. The kinetics code needs thermal-hydraulic fluid conditions in the volumes and heat structures that represent the reactor core to perform its computations. The kinetics coupling as implemented in the R5EXEC program expects the client tasks to perform their thermal-hydraulic computations at the beginning of a time step using the reactor power data received from the server task at the end of the previous time step. After the client codes advance their thermal-hydraulic models, they send the required thermal-hydraulic conditions to the server task and listen to receive the computed reactor power data. Before beginning the kinetics computations of a time step advancement the server task listens to receive the thermal-hydraulic conditions from the client tasks, then advances the reactor power using the thermal-hydraulic conditions received from the client tasks. The thermal-hydraulic conditions received from the client tasks are used by the server task to compute the reactivity for point kinetics or the neutron cross sections for nodal kinetics. Once the server task has completed its time step advancement it sends the newly updated reactor power to the client tasks.

The data that are to be sent from the client tasks to the server task consist of predetermined sets of fluid conditions in the specified volumes in the thermal-hydraulic model of the client task or a single value, presently defined as the volume averaged temperature in a heat structure, in the specified heat structures in the client system model. The volumes are designated using the component name and volume number and the volume averaged temperature is designated by the use of the keyword 'heatstr' and the fully qualified heat structure number, where fully qualified means using the heat structure component number and heat structure number within the heat structure component as described in Section 4.8 of the RELAP5-3D input description (Appendix A of Volume II of the RELAP5-3D manuals).

The power data that the server task send to the client tasks consist of a set of five values for point kinetics or sets of five power values for nodal kinetics. The power data consist of the total power, the fission power, the total gamma power, the gamma power from the decay of fission products, and the gamma power from the decay of actinides. The actinides are Neptunium 238 and Plutonium 239 produced by neutron absorption in Uranium 238. The power for point kinetics is specified by the use of the keyword 'power' with the parameter 0 (zero) and the sets of power data are specified by the use of the keyword 'zone' for nodal kinetics power data with the parameter being the zone identification number. The zones are defined in the input data for the kinetics model of the server task.

A 'zone' is a mapping between the volumes and heat structures in a thermal-hydraulic model and the computational nodes in the kinetics model. A zone contains one or more volumes and heat structures from the thermal-hydraulic model and one or more nodes from the kinetics model. The point kinetics model can be considered a kinetics model having only one zone that encompasses the entire reactor core. The conditions in the volumes of a zone are used to compute a weighted average set of fluid properties for the zone and the conditions in the heat structures in a zone are used to compute a weighted average structure temperature for the zone. The fission power in the zone is the summation of the fission powers in the kinetics nodes in the zone. The other powers in the zone are computed by the decay heat model using the total fission power in the zone.

The volume data that are to be received by the server task are defined using the keyword 'volume' and the parameter is a user defined value that must be distinct from the identifiers for the volumes in the thermal-hydraulic model in the server task. When RELAP5-3D is the server, the volume parameters must be numbers less than 1000000. Similarly, the heat structure temperatures to be received by the server task are defined using the keyword 'heatstr' and parameter values less than 10000. If RELAP5-3D is the server task, the reactor power may also be determined using a power table or control variable. These two options are specified using the keywords 'tableout' and 'cntrlvar' respectively. If these options are used, the five power values sent by the server task are all the same value, the output value from the table or control variable.

The implementation of the kinetics coupling has been verified using two sets of test cases, one set of test cases for point kinetics and one set of test cases for nodal kinetics. Each set consists of two test cases, an uncoupled test case and a coupled test case. All test cases are based on the thermal-hydraulic model in the 'typpwr' installation test case. Figure 19 (on page 44) shows a schematic of the uncoupled test case (test case ‘pvmnd’) used to verify the implementation of the kinetics coupling. Figure 20 (on page 45) shows a detailed view of the reactor vessel for the uncoupled test case and Figure 21 (on page 46) shows the schematic of the reactor core for the coupled test case (test case ‘pvmndx’) that uses semi-implicit coupling for thermal-hydraulic coupling. The middle four volumes and heat structures of the reactor core, i.e., pipe 335 in the uncoupled model, have been removed from the input model for the server task and have been moved to the input model for the client task. The coupled test case uses semi-implicit coupling of the thermal-hydraulic model in the server task to the thermal-hydraulic model in the client task. The server task is the master task for semi-implicit coupling and the client task is the slave task for semi-implicit coupling. Figure 22 shows a schematic of the nodal kinetics model in the server task. The figure shows the core model for the uncoupled version of the test case on the left and the coupled version of the test case on the right. Each zone has a single volume and a single heat structure but zones can have multiple volumes and multiple heat structures. The middle four zones in the server task are shown as having phantom volumes and heat structures because these volumes and heat structures do not exist in the thermal-hydraulic model for the server task. Their conditions are computed by the thermal-hydraulic model in the client task. The arrows in Figure 22 show the thermal-hydraulic data flow between the client task and the server task.

The point kinetics test cases are the ‘pvmpt’ test cases and the nodal kinetics test cases are the ‘pvmnd’ test cases. The uncoupled test cases use RELAP5-3D input decks pvmpt.i and pvmnd.i respectively and the coupled test cases use the ‘pvmpts’ (the server task) and ‘pvmptc’ (the client task) input decks for the coupled point kinetics test case and the ‘pvmnds’ (the server task) and ‘pvmndc’ (the client task) input decks for the server and client tasks in the coupled nodal kinetics test case. The template files for the point kinetics and nodal kinetics test cases are ‘pvmptx’ and pvmndx’ respectively. Figure 23 (on page 48) shows the template file for the point kinetics test case and Figure 24 (on page 49) shows the template file for the nodal kinetics test case.

The figures show that the data to be transferred between the server and client tasks are defined by pairs of words as is used by explicit and semi-implicit coupling. Remember that the way the data items are specified to the coupled codes is code specific and that using pairs of words is specific to RELAP5-3D and may be different when coupling different codes to RELAP5-3D. Figure 24 shows how volume numbers less than 1000000 and heat structure numbers less than 10000 are used by the server task to receive data for phantom volumes and phantom heat structures that are not part of its thermal-hydraulic model.

The implementation of the kinetic coupling was verified by comparing the results of the coupled and uncoupled test cases where identical results were expected and where identical results were computed. The test cases used for the verification of the kinetics coupling use both thermal-hydraulic coupling and the kinetics coupling. However, this is not a requirement of kinetic coupling and the server task could be a pure kinetics code having no thermal-hydraulic model for the reactor core. In this case, all of the thermal-hydraulic conditions for the reactor code must be obtained from the client code.

4.4 Control Systems Coupling

Control system coupling is the last type of coupling and like kinetics coupling is a specialized type of synchronous explicit coupling. The assumed order of computations is to advance the thermal-hydraulic model, then advance the kinetics model, and finally advance the control systems model. This is the order of computations in RELAP5-3D. The control systems coupling in RELAP5-3D is implemented by a new type of control block called the ‘cplfnctn’ control block. This control block can send data to its analog (the analogous control component) in another task, and can receive data from its analog in another task. The control block first sends any data specified by the R5EXEC program and then listens to receive any data specified. Because the control blocks in a RELAP5-3D control systems model are updated in numerical order, the sends and receives are not coordinated by the R5EXEC program as is done for parallel, explicit coupling. The user must be careful in ordering the control blocks in the coupled models so that when a control block in one task is sending data, the analog in the other coupled task is listening to receive that data and vice-versa or a deadlock will occur. A deadlock occurs when both tasks are either listening to receive a message or are both listening to receive the acknowledgment to a message that they have just sent.

The ‘cplfnctn’ control block in RELAP5-3D can send multiple data items and can receive multiple data items. The data items that are to be sent must be data items that are defined in the RELAP5-3D input model. A input error results if the user tries to send a data item that is not part of the RELAP5-3D input deck. When receiving data items, a control block needs to store the values so that they may be accessible to the other models in RELAP5-3D. The mechanism chosen for RELAP5-3D was to use the RELAP5-3D interactive variables. The user must define at least as many interactive variables as there are values to be received by the ‘cplfnctn’ control blocks. The variables to be sent or received by RELAP5-3D for control systems coupling are specified in the ‘control’ section of the input deck for the R5EXEC program. The data items are described by the keyword ‘extfnctn’ followed by the control block number of the ‘cplfnctn’ control block in the RELAP5-3D input deck. This pair of descriptors is followed by pairs of descriptors, a name string and a integer parameter, that describe a data item in the RELAP5-3D database. As many pairs of data descriptors are entered as necessary to specify the data items for that ‘cplfnctn’ control block. Multiple ‘extfnctn’ specifications may be entered on the same line in the input file for the R5EXEC program. Note that the identifier for a coupling control block in the input files for the R5EXEC program is the string ‘extfnctn’, which is different from the name of the coupling control block in RELAP5-3D, the string ‘cplfnctn’.

Two test cases were used to verify the implementation of the control system coupling: ‘pvmcs’, and ‘pvmcsx’. Figure 25 (on page 50) shows the schematic of a portion of the control system of the ‘pvmcs’ installation test case, which is the uncoupled analogue of the coupled test case ‘pvmcsx’. This test case is identical to the ‘edhtrk’ test case that is used as the uncoupled analogue of the ‘pvmedax’ coupling test case. The thermal-hydraulic coupling in the ‘pvmcsx’ test case is identical to the thermal-hydraulic coupling portion of the ‘pvmedax’ test case except that the tasks are defined as synchronous tasks instead of asynchronous tasks. Synchronous coupling is used for the control system test cases because only synchronous coupling has been implemented for control systems coupling in the R5EXEC program and in RELAP5-3D.

A schematic of the control systems in the coupled version of this test case is shown in Figure 26 (on page 51). The template for the input file for the R5EXEC program for this test case is shown in Figure 27 (on page 52). Most of the control system in the ‘pvmcs’ test case is included in the input file for the ‘child’ task of the coupled simulation but control blocks 12 and 13 in the uncoupled test case are moved to the ‘parent’ task. These control blocks in the ‘parent’ task receive their input from the interactive variable ‘extvar1’ which in turn receives its value from ‘cplfnctn’ control block 1. Control blocks 12 and 13 in the ‘child’ task are replaced by a ‘cplfnctn’ control block as shown in Figure 26. This control block sends input data to ‘cplfnctn’ control block 1 in the ‘parent’ task and receives output data from ‘cplfnctn’ control block 14 in the ‘parent’ task.

Figure 26 illustrates the use of interactive variables to store the values received by the ‘cplfnctn’ control blocks. Interactive variables are specified using the name of the interactive variable and the parameter 1000000000. It is the parameter 1000000000 that distinguishes a variable as an interactive variable in RELAP5-3D. The user can verify that the variable named ‘extvar1’ specified in the input file for the R5EXEC program is defined in the input deck for the parent task, i.e., file pvmcsp.i, and that the interactive variables specified in the input file for the R5EXEC program for the child task are defined in the RELAP5-3D input deck, i.e., file pvmcsc.i. The implementation of the control systems coupling was verified by comparing the output of the uncoupled test case ‘pvmcs’ to the output of the coupled version of the test case ‘pvmcsx’. The output of the regular control blocks, i.e., the non-coupling control blocks, is expected to be identical to the coupled control blocks.

5 Coupling Errors

There are many types of errors that can occur in a coupled computation. The coupling methodology has been designed to detect a number of these errors and fail gracefully. The first type of error that can occur is that the virtual machine may fail to start up correctly. The PVM software will write a message to the screen and then terminate. This type of error occurs because another virtual machine is already running or a previous coupled run failed and that virtual machine was not shut down properly. In either case the user must clean up the temporary files used by PVM before trying to run again. PVM writes a file named pvmd.<uid> where the string <uid> is the identification number of the user executing the coupled run. This file is written by default to the /tmp directory. The user may redirect this file to the directory defined in the PVM_TMP environmental variable. If this file is present, the virtual machine will fail to start up and the user must either remove this file or change the location for the file. The virtual machine may also fail to start up if the user has failed to define the PVM_ROOT and PVM_ARCH environmental variables that define the location of the PVM software library.

The second type of error is an error in the input processing sections of the coupled codes. The R5EXEC program sends messages to the coupled codes during their input processing phase to define the data that are to be exchanged between the several coupled codes. The input processors of the coupled codes check that the variables named in the messages received from the R5EXEC program exist in the database of the code and return a failure condition to the executive program at the end of initialization if the variables are not defined in the input file. Additionally, a coupled code may fail during initialization because of errors in its input deck. In both of these cases, the code returns a failure condition to the R5EXEC program. The R5EXEC program displays the initialization status of the coupled codes on the terminal screen and in its output file. If there is a failure in the initialization of any of the coupled codes, the coupled run is terminated and the virtual machine is terminated gracefully.

The third kind of error is an error in the transient time step advancements. Several conditions may cause the coupled run to terminate. One of the coupled codes may experience a hard failure such as a divide by zero and its execution terminated by the operating system. The R5EXEC coupling has been coded to detect the failure of one of the coupled tasks and to terminate the other tasks gracefully. The tasks that have not failed will usually write a message saying that they have timed out while waiting for a message from the failed task. They will also inform the R5EXEC program of this condition and the R5EXEC program will direct the remaining tasks to terminate after writing a final restart record. Another kind of timeout may occur if the user has specified a wait time that is too small using the keyword ‘awaits’. This detection and graceful shutdown task will occur UNLESS the user has told the task to wait ‘forever’ using the keyword ‘awaits’ in the input file for the R5EXEC program. An infinite wait time, i.e., a wait time of -1.0, should never be used for production runs because the codes will never timeout when a failure condition is encountered. The user should consult the output file of the code experiencing the hard failure or timeout to determine what has happened.

Another type of error that will cause a coupled run to terminate is if one of the coupled tasks encounters an error in its computations from which it cannot recover. For example, RELAP5-3D cannot recover from a fluid property error. In this case RELAP5-3D tells the R5EXEC program that it must terminate its execution and the R5EXEC program then tells all of the other coupled codes that they should also terminate their execution. It then terminates the virtual machine gracefully after all of the coupled codes have ceased their execution. The user should examine the output file of the code experiencing the problem to determine how to fix it.

The last kind of error is a deadlock where all of the coupled codes are listening to receive data from another code or all codes have just sent a message and are waiting to receive an acknowledgment for the message they just sent. The user will notice that the display on the terminal screen stops being periodically updated which happens when the coupled run is progressing normally. In this case, all of the codes should timeout waiting for the message or the acknowledgment. This kind of deadlock should only occur in control systems coupling where the user has defined the control blocks in the wrong order. Any other type of deadlock indicates an error in the coupling methodology and should be referred to the code support staff.

6 References

1 J. Hope Forsmann, W. L. Weaver, *The Application Programming Interface for the R5EXEC Program and Associated Code Coupling System*, INL/EXT-05-00107, Idaho National Laboratory, April 2015.

2 J. Hope Forsmann, W. L. Weaver, *Programmers Manual for the R5EXEC Program*, INL/EXT-05-00159, Idaho National Laboratory, April, 2015.

3 J. Hope Forsmann, W. L. Weaver, *Programmers Manual for the R5EXEC Interface in the RELAP5-3D Code*, IN/EXT-05-00203, Idaho National Laboratory, April, 2015.

4 C. Y. Paik and L. E. Hochreiter, *Analysis of FLECHT SEASET 163-Rod Blocked Bundle Data Using COBRA-TF*, NUREG/CR-4166, US Nuclear Regulatory Commission, 1986.

5 The RELAP5 Code Development Team, *RELAP5-3D Code Manuals, Vol I, II, IV, and V,* Idaho National Engineering and Environmental Laboratory, INEEL-EXT-98-00834, Revision 2.2, October 2003.

6 Fluent Corporation, *Fluent 6 User's Guide,* 2001.

7 Safety Code Development Group, *TRAC-PF1/MOD1: An Advanced Best-Estimate Computer Program for Pressurizer Water Reactor Thermal-Hydraulic Analysis,* LA-10157-MS, NUREG/CR-3858, Los Alamos National Laboratory, July 1986.

8 J. W. Spore, et. al., *TRAC-M/FORTRAN 90 (Version 3.0) Theory Manual,* NUREG/CR-6724, Los Alamos National Laboratory and Pennsylvania State University, July 2001.

9 K. K. Murata et. al., *Code Manual for CONTAIN 2.0: A Computer Code for Nuclear Reactor Containment Analysis,* SAND97-1735, NUREG/CR-6533, Sandia National Laboratory, December 1997.

10 R. O. Gauntt et. al., *MELCOR Computer Code Manuals: Version 1.8.5,* SAND2000-2417, NUREG/CR-6119, Sandia National Laboratory, October 2000.

11 P. J. Turinsky, R. M. K. Al-Chalabi, P. Engrand, *NESTLE: A Few-Group Neutron Diffusion Equation Solver Utilizing the Nodal Expansion Method for Eigenvalue, Adjoint, Fixed Source Steady State and Transient Problems,* EGG-NRE-11406, Idaho National Engineering Laboratory, 1994.

12 T. Downar et. al., *PARCS: Purdue Advanced Reactor Core Simulator, PU/NE-98-26,* Purdue University, West Lafayette, IN, September, 1998.

13 A. Geist et. al., *PVM (Parallel Virtual Machine) User's Guide and Reference Manual,* Oak Ridge National Laboratory, ORNL/TM-12187, 1993.

14 R. P. Martin, "RELAP5/MOD3 Code Coupling Model," *Nuclear Safety*, *Vol. 36, No. 2*, pp. 290-299, July-December 1995.

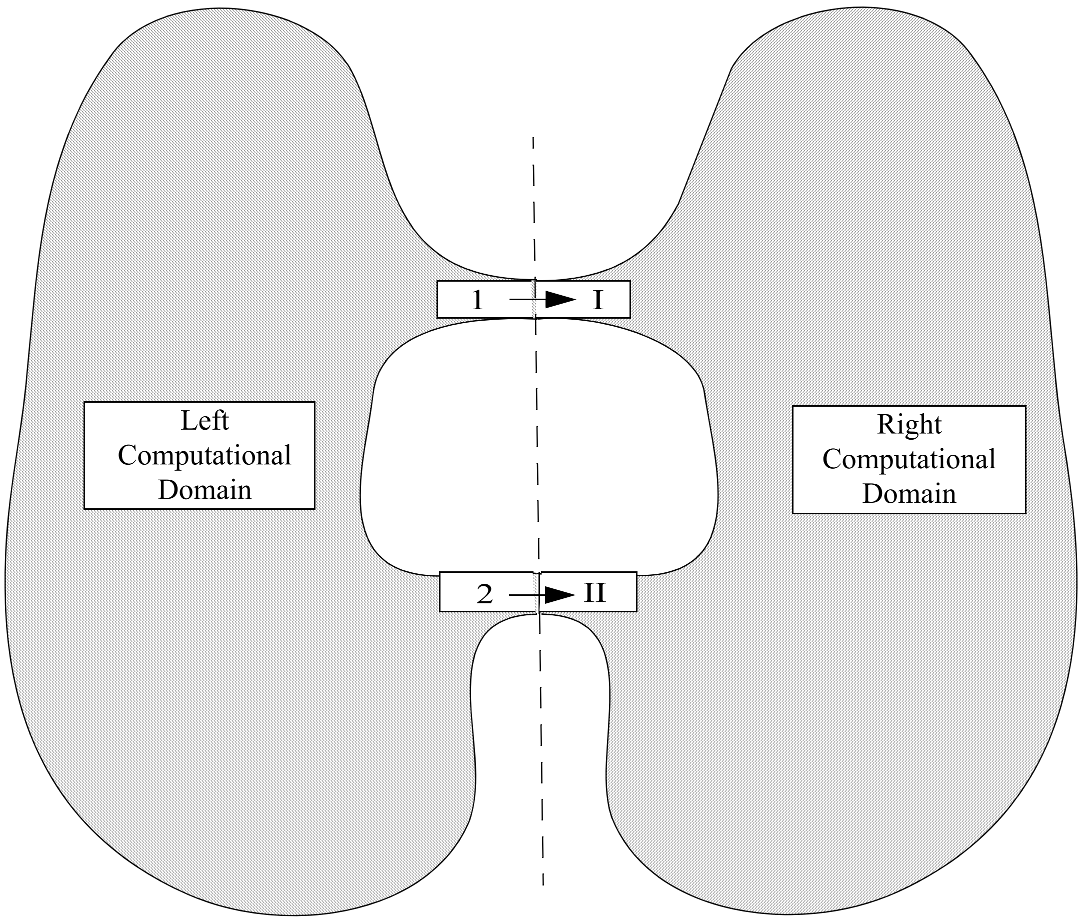

Figure 1 Schematic for coupled problem domain

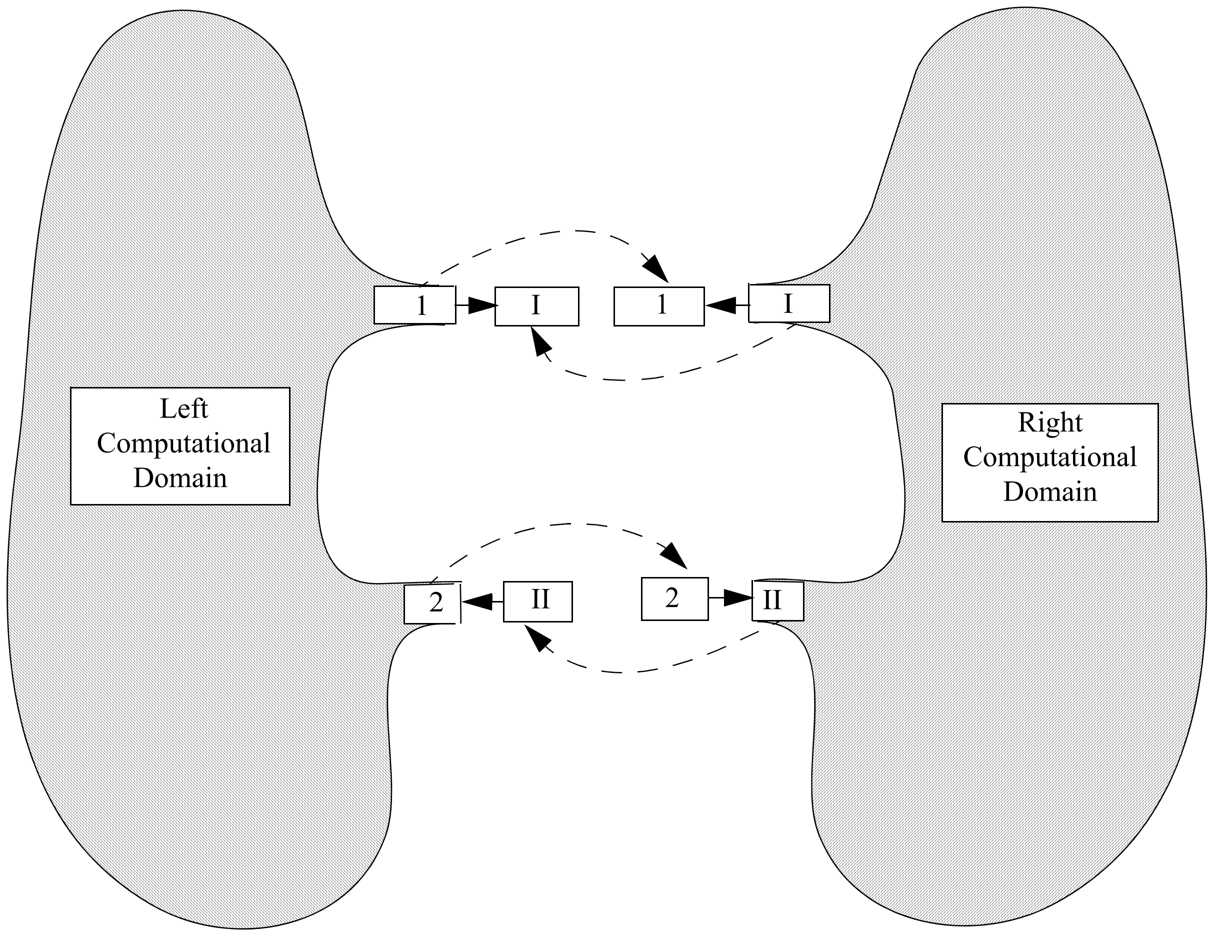

Figure 2 Schematic for parallel explicit coupling

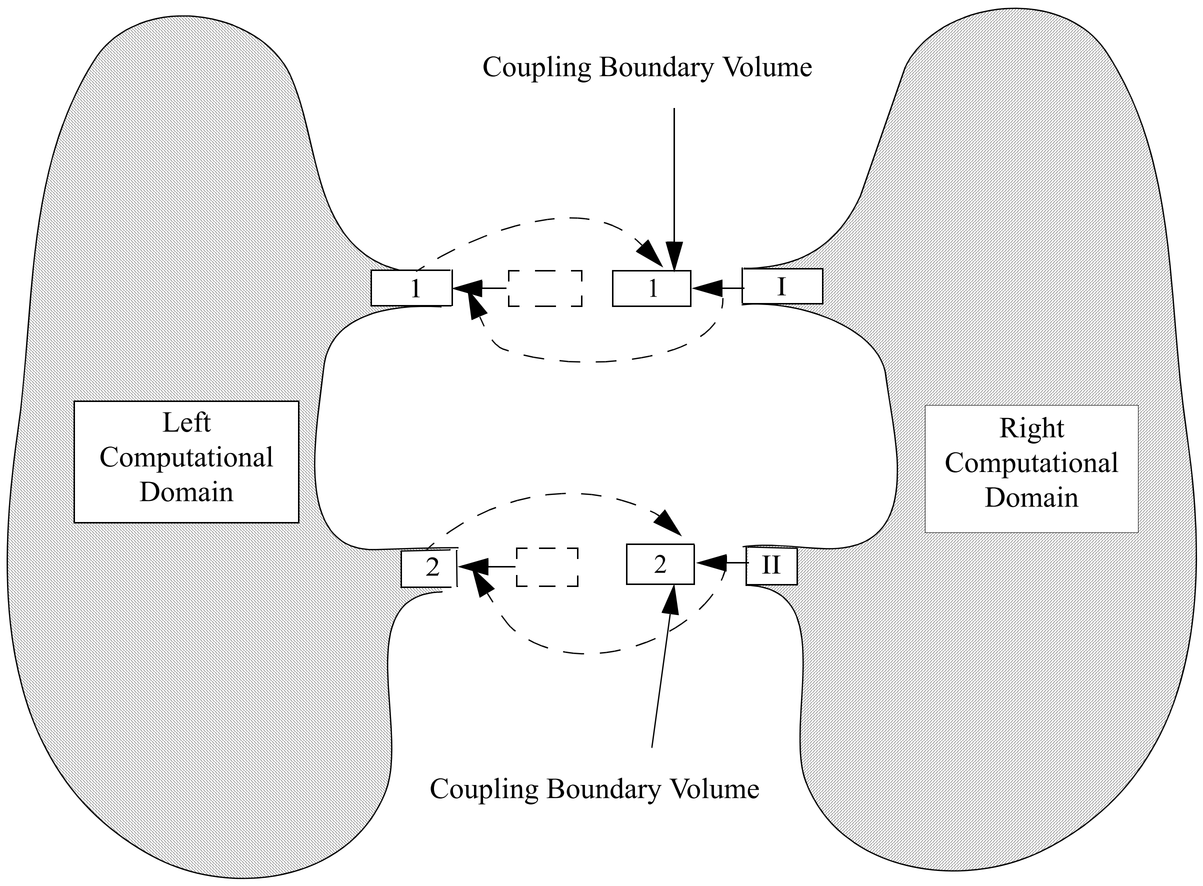

Figure 3 Schematic for semi-implicit and sequential explicit thermal-hydraulic coupling

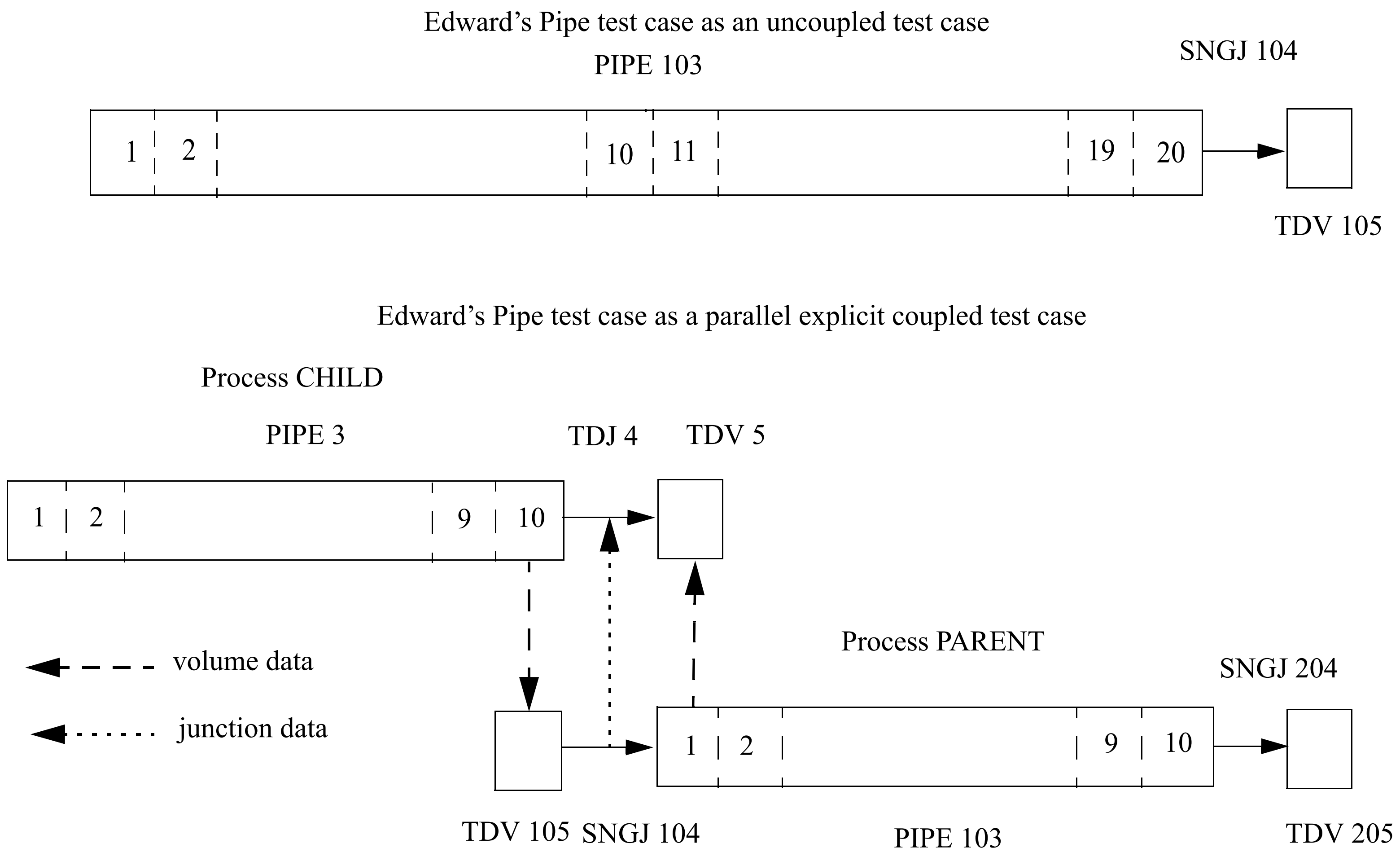

Figure 4 Schematic of PVMEDA parallel explicit test case

```
        # pvm asynchronous explicit test case pvmeda
        virtual
                MACHINE wd=WHERE ep=WHERE
                write pvmedax.out
        simulation
                name edward's pipe asynchronous explicit coupling test case
        processes
                MACHINE
                parent asynchronous relap5.x -i pvmedap.i -o pvmedap.p -r pvmedap.r
                parent writes pvmedap.out
                child asynchronous relap5.x -i pvmedac.i -o pvmedac.p -r pvmedac.r
                child writes pvmedac.out
        messages
                explicit
                parent sends child p 103010000 uf 103010000 ug 103010000
                child receives parent p 5010000 uf 5010000 ug 5010000
                parent sends child voidg 103010000
                child receives parent voidg 5010000
                parent sends child velfj 104000000 velgj 104000000
                child receives parent velfj 4000000 velgj 4000000
                parent sends child mflowgj 104000000 mflowfj 104000000
                child receives parent mflowgj 4000000 mflowfj 4000000
                child sends parent p 3100000 uf 3100000 ug 3100000
                parent receives child p 105010000 uf 105010000 ug 105010000
                child sends parent voidg 3100000
                parent receives child voidg 105010000
#               child sends parent velfj 4000000 velgj 4000000
#               parent receives child velfj 104000000 velgj 104000000
#               child sends parent mflowgj 4000000 mflowgj 4000000
#               parent receives child mflowgj 104000000 mflowfj 104000000
        timesteps
        # 0.0050    0.0000001 0.00005 7   20 50 100 1
                0.050    0.0000001 0.0001 7   10 50 100 1
        # 0.5000    0.0000001 0.0001 7   10 50 100 1
```

Figure 5 Listing of PVMEDAX template file

```
# pvm asynchronous explicit test case pvmeda
    virtual
        MACHINE wd=WHERE ep=WHERE
    processes
        MACHINE
        parent asynchronous relap5.x -i pvmedap.i -o pvmeda1p.p -r pvmeda1p.r
        child asynchronous relap5.x -i pvmedac.i -o pvmeda1c.p -r pvmeda1c.r
    simulation
        start time 0.0001
    messages
        explicit
        parent sends child p 103010000 uf 103010000 ug 103010000
        child receives parent p 5010000 uf 5010000 ug 5010000
        parent sends child voidg 103010000
        child receives parent voidg 5010000
        parent sends child velfj 104000000 velgj 104000000
        child receives parent velfj 4000000 velgj 4000000
        parent sends child mflowgj 104000000 mflowfj 104000000
        child receives parent mflowgj 4000000 mflowfj 4000000
        child sends parent p 3100000 uf 3100000 ug 3100000
        parent receives child p 105010000 uf 105010000 ug 105010000
        child sends parent voidg 3100000
        parent receives child voidg 105010000
    #child sends parent velfj 4000000 velgj 4000000
    #parent receives child velfj 104000000 velgj 104000000
    #child sends parent mflowgj 4000000 mflowgj 4000000
    #parent receives child mflowgj 104000000 mflowfj 104000000
    timesteps
    # 0.0050    0.0000001 0.00005 7   20 50 100 1
        0.050    0.0000001 0.0001 7   10 50 100 1
    # 0.5000    0.0000001 0.0001 7   10 50 100 1
```
Figure 6 Listing of PVMEDA1X template file

```
# pvm asynchronous explicit test case pvmeda
        virtual
                MACHINE wd=WHERE ep=WHERE
        processes
        MACHINE
                parent asynchronous relap5.x -i pvmedarp.i -o pvmedarp.p -r pvmedarp.r
                child asynchronous relap5.x -i pvmedarc.i -o pvmedarc.p -r pvmedarc.r
        simulation
                restart time 0.02
                name edward's pipe asynchronous explicit coupling test case
        messages
                explicit
                parent sends child p 103010000 uf 103010000 ug 103010000
                child receives parent p 5010000 uf 5010000 ug 5010000
                parent sends child voidg 103010000
                child receives parent voidg 5010000
                parent sends child velfj 104000000 velgj 104000000
                child receives parent velfj 4000000 velgj 4000000
                parent sends child mflowgj 104000000 mflowfj 104000000
                child receives parent mflowgj 4000000 mflowfj 4000000
                child sends parent p 3100000 uf 3100000 ug 3100000
                parent receives child p 105010000 uf 105010000 ug 105010000
                child sends parent voidg 3100000
                parent receives child voidg 105010000
        #child sends parent velfj 4000000 velgj 4000000
        #parent receives child velfj 104000000 velgj 104000000
        child sends parent mflowgj 4000000 mflowgj 4000000
        #parent receives child mflowgj 104000000 mflowfj 104000000
        timesteps
        # 0.0050    0.0000001 0.00005 7   20 50 100 1
                 0.050    0.0000001 0.0001 7   10 50 100 1
        # 0.5000    0.0000001 0.0001 7   10 50 100 1
```
Figure 7 Listing of PVMEDARX template file

```
# pvm asynchronous explicit test case pvmeda
        virtual
                MACHINE wd=WHERE ep=WHERE
        processes
                MACHINE
                parent asynchronous relap5.x -i pvmedarp.i -o pvmedar1p.p -r pvmedar1p.r
                child asynchronous relap5.x -i pvmedarc.i -o pvmedar1c.p -r pvmedar1c.r
        simulation
                 restart time -1.0
        messages
                explicit
                parent sends child p 103010000 uf 103010000 ug 103010000
                child receives parent p 5010000 uf 5010000 ug 5010000
                parent sends child voidg 103010000
                child receives parent voidg 5010000
                parent sends child velfj 104000000 velgj 104000000
                child receives parent velfj 4000000 velgj 4000000
                parent sends child mflowgj 104000000 mflowfj 104000000
                child receives parent mflowgj 4000000 mflowfj 4000000
                child sends parent p 3100000 uf 3100000 ug 3100000
                parent receives child p 105010000 uf 105010000 ug 105010000
                child sends parent voidg 3100000
                parent receives child voidg 105010000
        #child sends parent velfj 4000000 velgj 4000000
        #parent receives child velfj 104000000 velgj 104000000
        #child sends parent mflowgj 4000000 mflowgj 4000000
        #parent receives child mflowgj 104000000 mflowfj 104000000
        timesteps
        # 0.0050    0.0000001 0.00005 7   20 50 100 1
                 0.060    0.0000001 0.0001 7   10 50 100 1
        # 0.5000    0.0000001 0.0001 7   10 50 100 1
```

Figure 8 Listing of PVMEDAR1X template file

```
0 Variables for Explicit Send Message Tag   1
   variable code    parameter
 p                103010000
 uf               103010000
 ug               103010000
0 Variables for Explicit Send Message Tag   3
   variable code    parameter
 voidg            103010000
0 Variables for Explicit Send Message Tag   5
   variable code    parameter
 velfj            104000000
 velgj            104000000
0 Variables for Explicit Send Message Tag   7
   variable code    parameter
 mflowgj          104000000
 mflowfj          104000000
0 Variables for Explicit Receive Message Tag   9
   variable code    parameter
 p                105010000
 uf               105010000
 ug               105010000
0 Variables for Explicit Receive Message Tag 11
   variable code    parameter
 voidg            105010000
```
Figure 9 Partial listing of output file from ‘parent’ process in PVMEDAX test case

```
0 Variables for Explicit Send Message Tag   9
   variable code    parameter
 p                  3100000
 uf                 3100000
 ug                 3100000
0 Variables for Explicit Send Message Tag 11
   variable code    parameter
 voidg              3100000
0 Variables for Explicit Receive Message Tag   1
   variable code    parameter
 p                  5010000
 uf                 5010000
 ug                 5010000
0 Variables for Explicit Receive Message Tag   3
   variable code    parameter
 voidg              5010000
0 Variables for Explicit Receive Message Tag   5
   variable code    parameter
 velfj              4000000
 velgj              4000000
0 Variables for Explicit Receive Message Tag   7
   variable code    parameter
 mflowgj            4000000
 mflowfj            4000000
```
Figure 10 Partial listing of output file from ‘child’ process in PVMEDAX test case

Edward's Pipe test case as a sequential explicit coupled test case

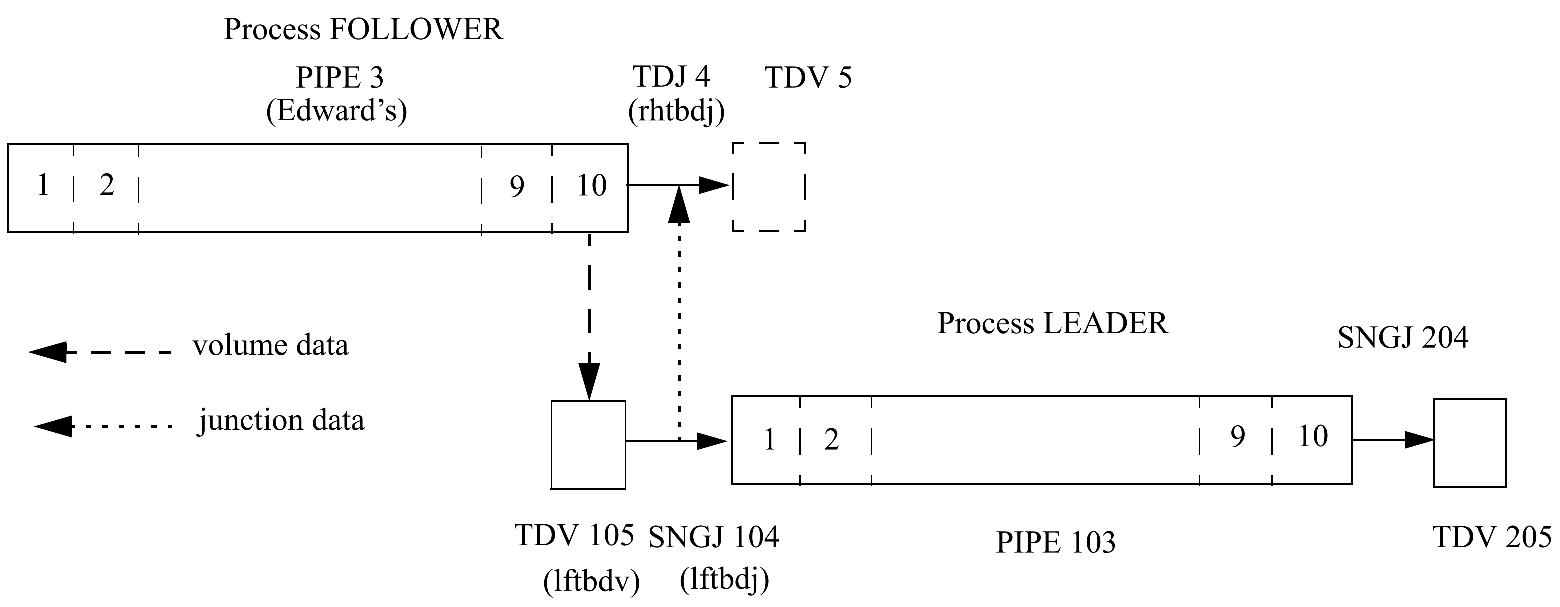

Figure 11 Schematic of PVMEDSX sequential explicit test case

```
# pvm asynchronous explicit conserving test case pvmedsx
        virtual
                MACHINE wd=WHERE ep=WHERE
        #   wait 10000.0 1
        processes
                MACHINE
                leader asynchronous relap5.x -i pvmedsl.i -o pvmedsl.p -r pvmedsl.r
                follower asynchronous relap5.x -i pvmedsf.i -o pvmedsf.p -r pvmedsf.r
        messages
                explicit
                leader preceeds follower
        # follower sends volume conditions to leader
                follower sends leader edward's 10
# leader receives volume conditions from follower
                leader receives follower lftbdv 1
        # leader sends average junction conditions to follower
                leader sends follower lftbdj 0
        # follower receives average junction conditions from leader
                follower receives leader rhtbdj 0
        timesteps
                0.050    0.0000001 0.0001 7   10 50 100 1
```

Figure 12 Listing of PVMEDSX template file

```
0 Variables for Explicit Send Message Tag   3
   variable code    parameter
 voidgj            104000000
 rhogj             104000000
 ugj               104000000
 qualaj            104000000
 velgj             104000000
 voidfj            104000000
 rhofj             104000000
 ufj               104000000
 velfj             104000000
 qualnj1           104000000
 qualnj2           104000000
 qualnj3           104000000
 qualnj4           104000000
 qualnj5           104000000
 aflowgj           104000000
 uflowgj           104000000
 uflowfj           104000000
 mflowgj           104000000
 mflowfj           104000000
 vflowgj           104000000
 vflowfj           104000000
 flenthg           104000000
 flenthf           104000000
 flentha           104000000
 nflowgj1          104000000
 nflowgj2          104000000
 nflowgj3          104000000
 nflowgj4          104000000
 nflowgj5          104000000
0 Variables for Explicit Receive Message Tag   1
   variable code    parameter
 p                 105010000
 voidg             105010000
 voidf             105010000
 ug                105010000
 uf                105010000
 quala             105010000
 rhog              105010000
 rhof              105010000
 tempg             105010000
 tempf             105010000
 qualan1           105010000
 qualan2           105010000
 qualan3           105010000
 qualan4           105010000
 qualan5           105010000
```
Figure 13 Partial listing of output file from ‘leader’ process in PVMEDSX test case

```
 Variables for Explicit Send Message Tag   1
   variable code    parameter
 p                3100000
 voidg            3100000
 voidf            3100000
 ug               3100000
 uf               3100000
 quala            3100000
 rhog             3100000
 rhof             3100000
 tempg            3100000
 tempf            3100000
 qualan1          3100000
 qualan2          3100000
 qualan3          3100000
 qualan4          3100000
 qualan5          3100000
0 Variables for Explicit Receive Message Tag   3
   variable code    parameter
 voidgj           4000000
 rhogj            4000000
 ugj              4000000
 qualaj           4000000
 velgj            4000000
 voidfj           4000000
 rhofj            4000000
 ufj              4000000
 velfj            4000000
 qualnj1          4000000
 qualnj2          4000000
 qualnj3          4000000
 qualnj4          4000000
 qualnj5          4000000
 aflowgj          4000000
 uflowgj          4000000
 uflowfj          4000000
 mflowgj          4000000
 mflowfj          4000000
 vflowgj          4000000
 vflowfj          4000000
 flenthg          4000000
 flenthf          4000000
 flentha          4000000
 nflowgj1         4000000
 nflowgj2         4000000
 nflowgj3         4000000
 nflowgj4         4000000
 nflowgj5         4000000
```

Figure 14 Partial listing of output file from ‘follower’ process in PVMEDSX test case

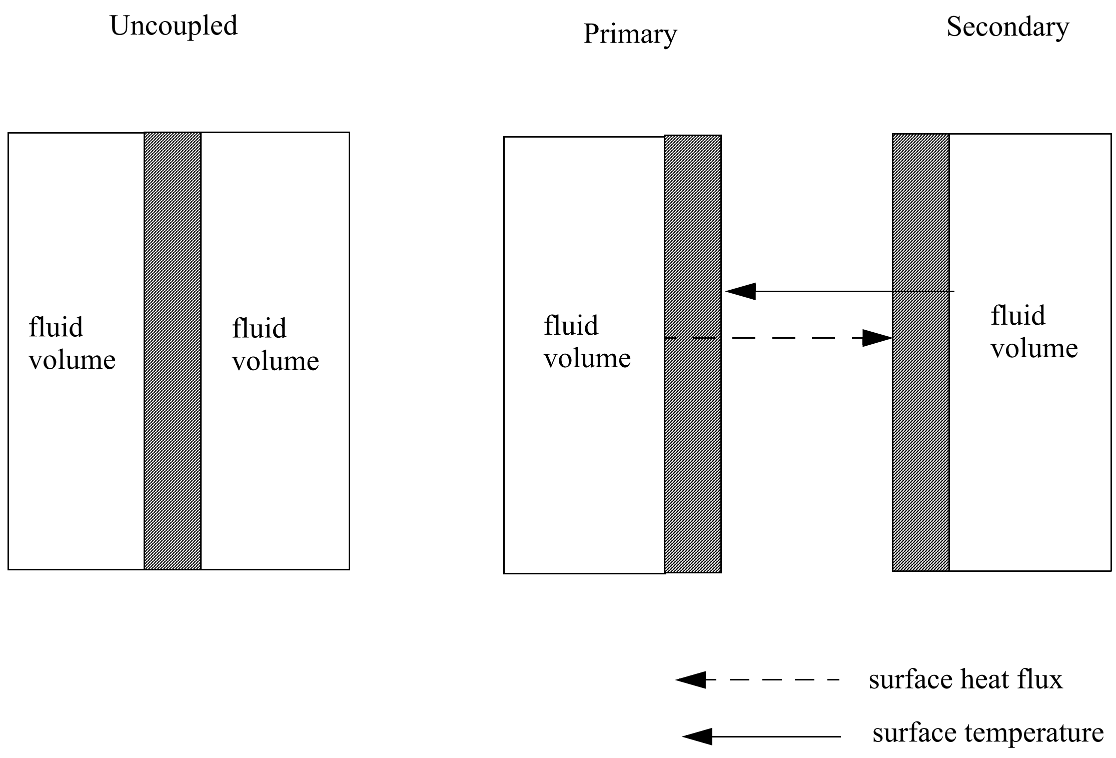

Figure 15 Schematic of heat structure coupling

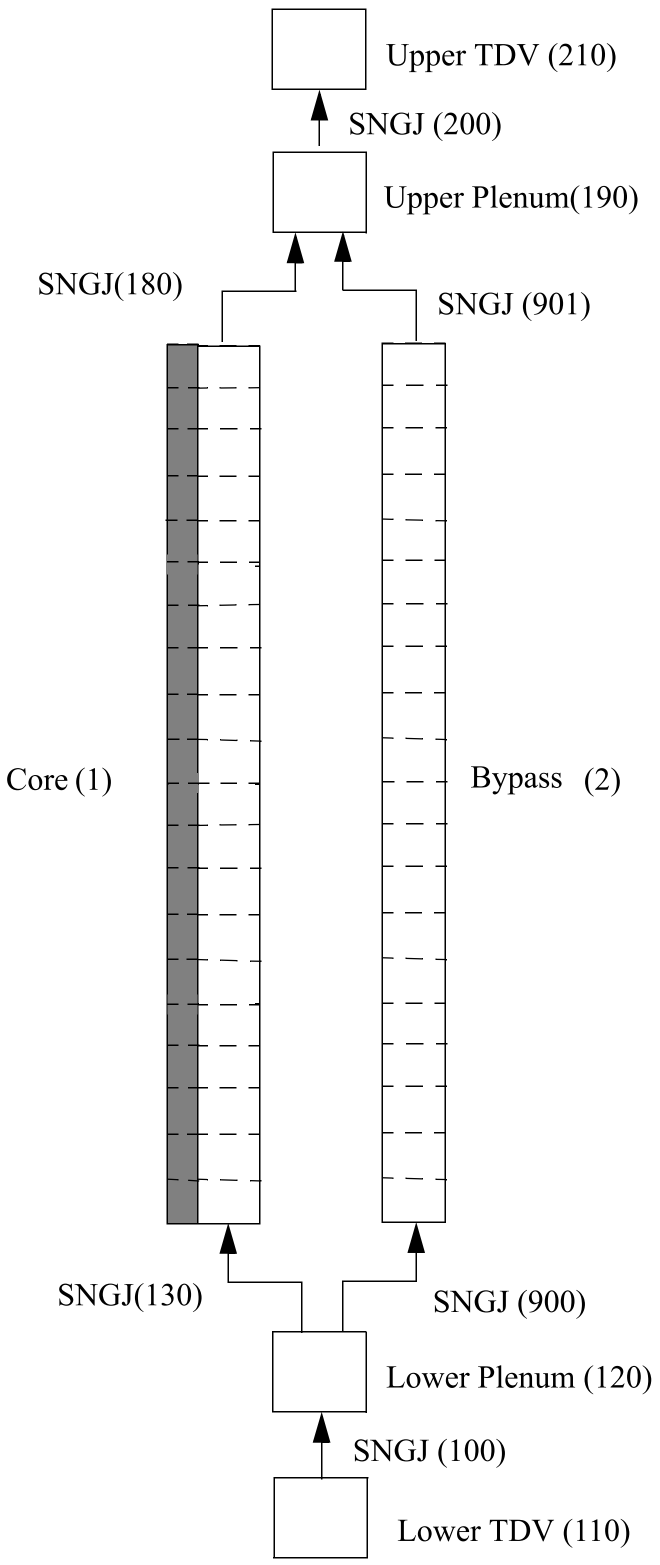

Figure 16 Schematic of uncoupled test case for semi-implicit coupling

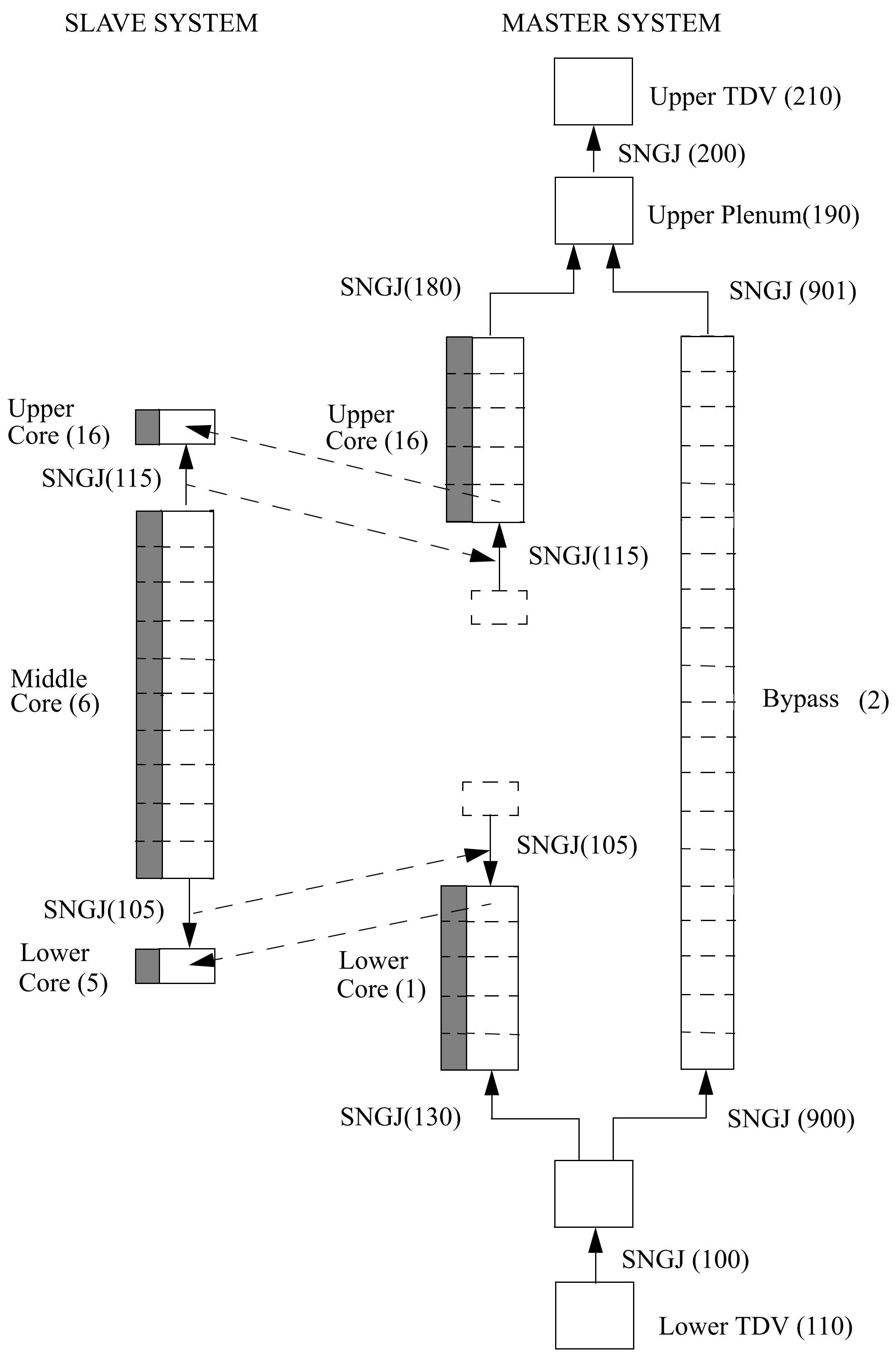

Figure 17 Schematic of coupled test case for semi-implicit coupling

```
# pvm semi-implicit test case pvmcore
virtual
        MACHINE wd=WHERE ep=WHERE
processes
   MACHINE
     primary synchronous relap5.x -i pvmcorep.i -o pvmcorep.p
     core synchronous relap5.x -i pvmcorec.i -o pvmcorec.p
messages
        semi-implicit
                primary sends core lrcore 5 upcore 1
                primary receives core lcrout 0 upcrin 0
                core sends primary lrcout 0 upcrin 0
                core receives primary lrcore 1 upcore 1
timesteps
 0.5000   0.000001 0.00625 403  1 20 500 0
 1.0000   0.000001 0.00625 403 20 20 500 0
 10.000   0.000001 0.0125  403 10 50 500 0
```
Figure 18 Listing of template file for ‘pvmcorex’ test case

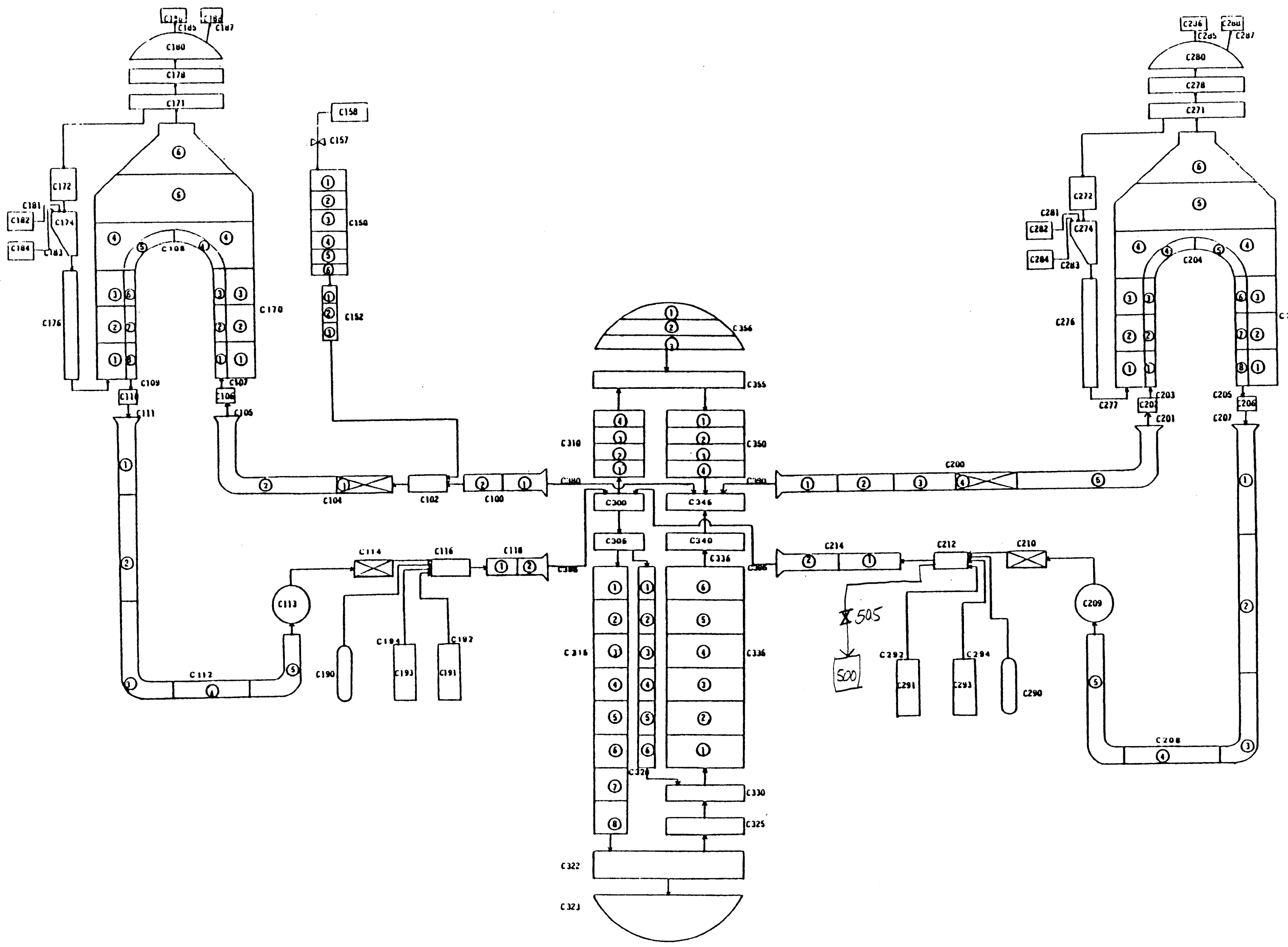

Figure 19 Schematic of TYPPWR test case

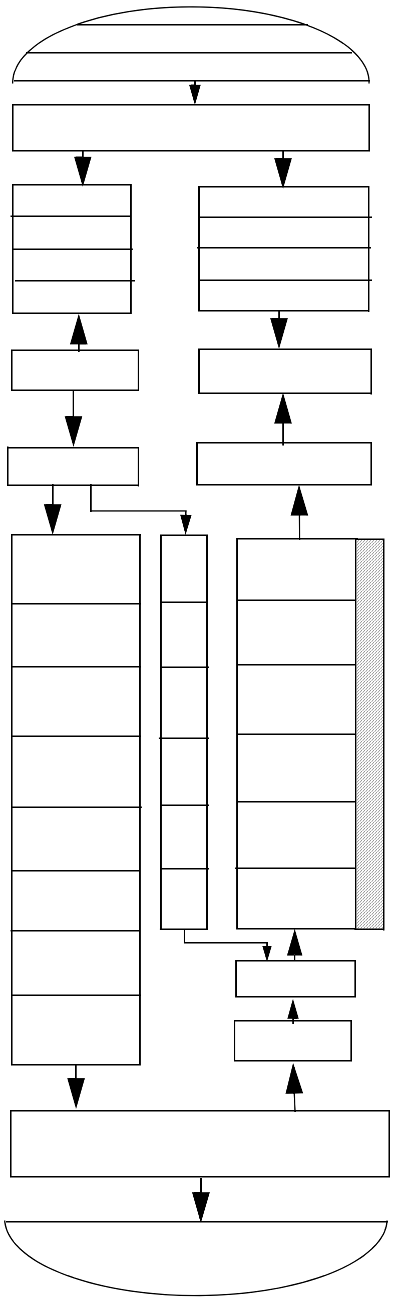
Figure 20 Schematic of reactor vessel in TYPPWR test case

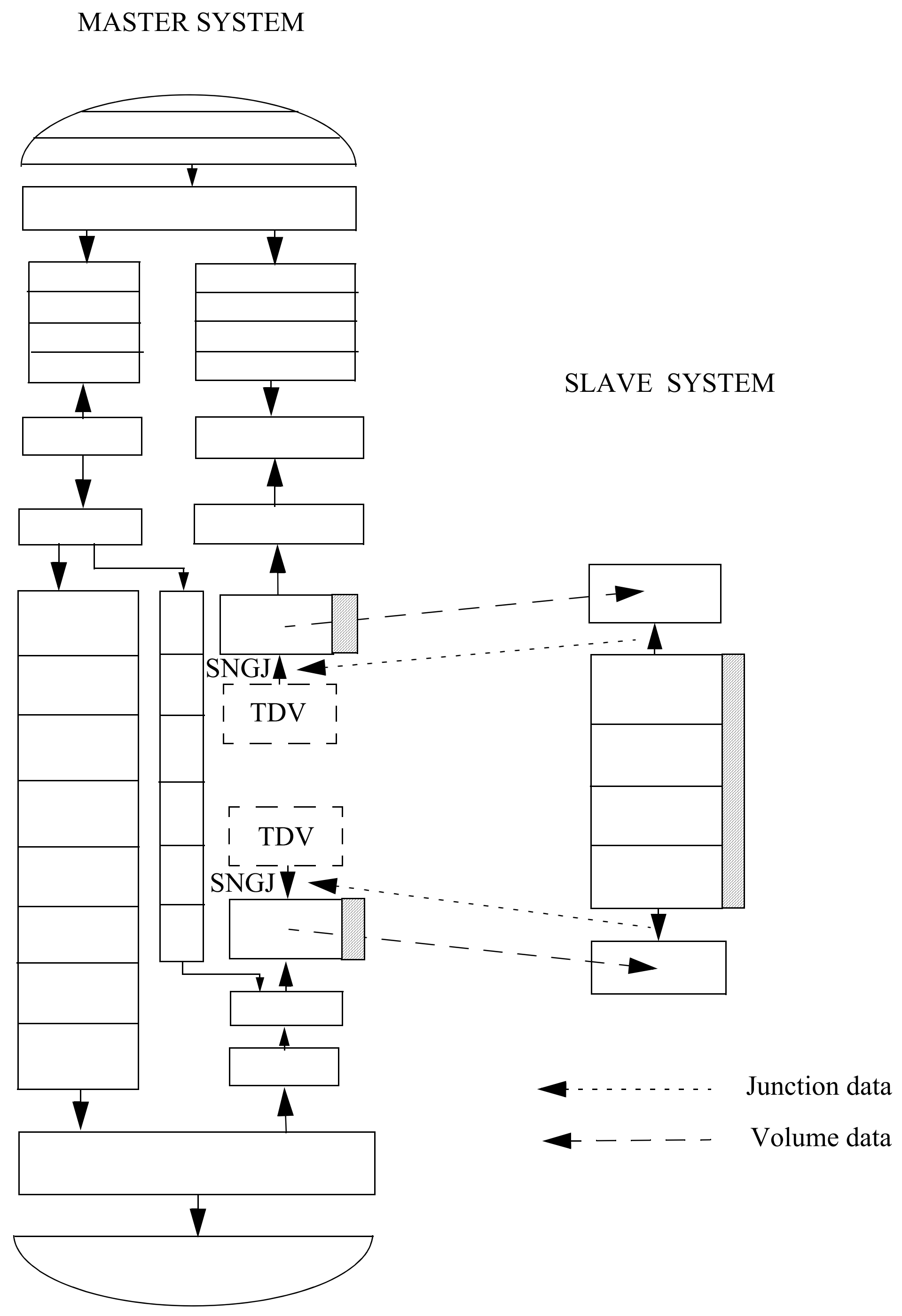

Figure 21 Schematic of coupled model of reactor vessel in kinetics coupling test cases

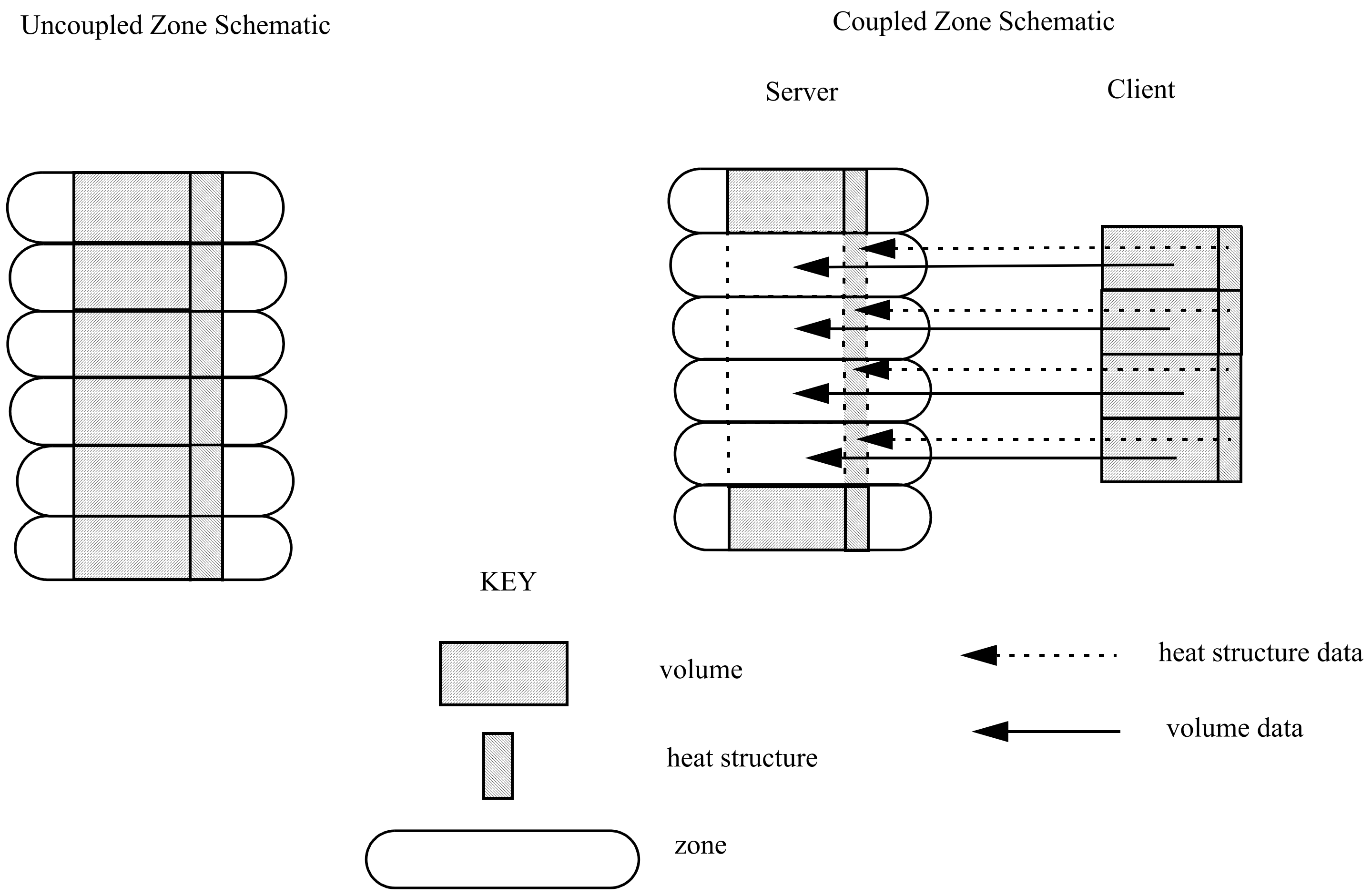

Figure 22 Schematic of Kinetics Zones

```
# pvm executive input for point kinetics coupling
virtual
        MACHINE wd=WHERE ep=WHERE
processes
        MACHINE
           server synchronous relap5.x -i pvmpts.i -o pvmpts.p
           client synchronous relap5.x -i pvmptc.i -o pvmptc.p
messages
        semi-implicit
           server sends client lrcore 1 upcore 1
           client receives server lrcrbdv 1 upcrbdv 1
           client sends server corein 0 coreout 0
           server receives client lrcrout 0 upcrin 0
           kinetics
           server sends client power 0
           client receives server power 0
           client sends server core 1 core 2 core 3 core 4
           server receives client volume 33502 volume 33503 volume 33504 volume 33505
           client sends server heatstr 3350001 heatstr 3350002 heatstr 3350003 heatstr 3350004
           server receives client heatstr 1 heatstr 2 heatstr 3 heatstr 4
timesteps
        1.0 0.0000001 0.50 7 2 2 160 1

```
Figure 23 Listing of template file for pvmptx point kinetics coupling test case

```
5 7vm executive input for nodal kinetics coupling
virtual
        MACHINE wd=WHERE ep=WHERE
processes
        MACHINE
            server synchronous relap5.x -i pvmnds.i -o pvmnds.p
            client synchronous relap5.x -i pvmndc.i -o pvmndc.p
messages
        semi-implicit
           server sends client lrcore 1 upcore 1
           client receives server lrcrbdv 1 upcrbdv 1
           client sends server corein 0 coreout 0
           server receives client lrcrout 0 upcrin 0
        kinetics
           server sends client zone 3 zone 4 zone 5 zone 6
           client receives server zone 3 zone 4 zone 5 zone 6
           client sends server core 1 core 2 core 3 core 4
           server receives client volume 33502 volume 33503 volume 33504 volume 33505
           client sends server heatstr 3350001 heatstr 3350002 heatstr 3350003 heatstr 3350004
           server receives client heatstr 1 heatstr 2 heatstr 3 heatstr 4
timesteps
        1.00 0.0000001 0.03125 7 1 16 160 1

```

Figure 24 Listing of template file for pvmndx for nodal kinetics coupling test case

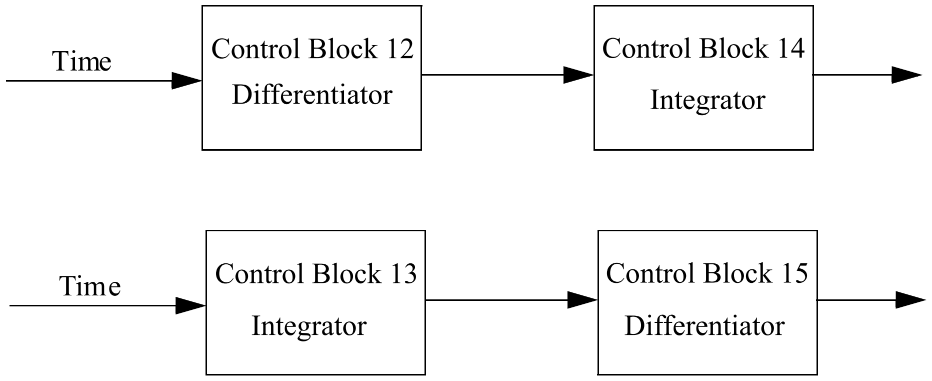

Figure 25 Schematic of uncoupled control systems test case

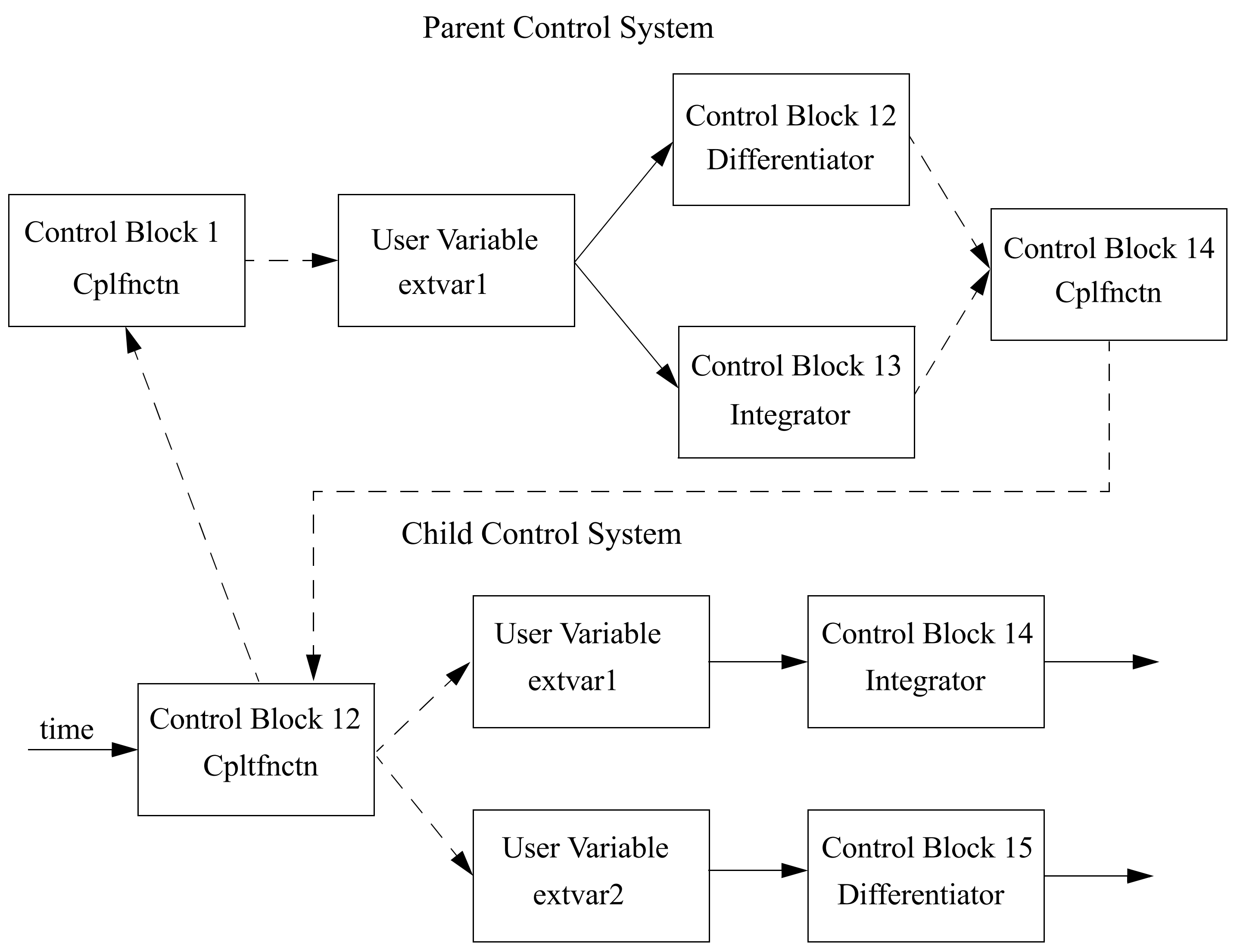

Figure 26 Schematic of coupled control systems test case

```
# pvm synchronous explicit test case pvmcsx for control system coupling
virtual
        MACHINE wd=WHERE ep=WHERE
#           wait 10.0
processes
    MACHINE
      parent synchronous relap5.x -i pvmcsp.i -o pvmcsp.p
      child synchronous relap5.x -i pvmcsc.i -o pvmcsc.p
messages
        explicit
                parent sends child p 103010000 uf 103010000 ug 103010000
                child receives parent p 5010000 uf 5010000 ug 5010000
                parent sends child voidg 103010000
                child receives parent voidg 5010000
                parent sends child velfj 104000000 velgj 104000000
                child receives parent velfj 4000000 velgj 4000000
                parent sends child mflowgj 104000000 mflowfj 104000000
                child receives parent mflowgj 4000000 mflowfj 4000000
                child sends parent p 3100000 uf 3100000 ug 3100000
                parent receives child p 105010000 uf 105010000 ug 105010000
                child sends parent voidg 3100000
                parent receives child voidg 105010000
        control
                child sends parent extfnctn 12 time 0
                parent receives child extfnctn 1 extvar1 1000000000
                parent sends child extfnctn 14 cntrlvar 12 cntrlvar 13
                child receives parent extfnctn 12 extvar1 1000000000 extvar2 1000000000
timesteps
 0.050    0.0000001 0.0001 7   10 50 100 1
```

Figure 27 Listing of template file for coupled control system test case

APPENDIX A
R5EXEC Input Data Requirements

A-1 Introduction

The input to the executive program is divided into five sections. These sections are delimited by reserved keywords. The keyword stands alone on an input card and indicates that the following cards contain data appropriate for that section of the input deck. A section of the input deck begins with a keyword and ends with another keyword or at the end of the input deck. Within each section of the input deck, additional keywords are used to identify the data on individual input cards. The keywords are arranged in a hierarchy where they must be used in the proper order. The reserved keywords are written in lower case and shown below:

- virtual
  - wait
  - write
- simulation
  - name
  - start time
  - restart time
- processes
  - synchronous
  - asynchronous
  - uses
  - writes
- messages
  - explicit
    - preceeds
    - awaits

- semi-implicit
    - awaits
- kinetics
    - awaits
    - power
    - zone
    - heatstr
    - table
    - cntrlvar

• control

- awaits

timesteps

In addition to these reserved keywords, the user can define keywords to be used in subsequent sections of the input decks. The user defined keywords are defined at the beginning of the input deck. Because the sections of the input decks are delimited by keywords, they can appear in any order in the deck but are processed by the input processor in a specific order. The layout of the five sections of the input deck is described in the order in which they are processed. Comment lines contain the sharp character (#) as the first non-blank character on the line and can appear anywhere in the input deck. Blank lines and unusual characters such as a tab are not allowed in the input deck.

A-2 Virtual Section of Input Deck

The first section of the input deck is optional. This section, if present, begins with the keyword 'virtual'. There are two types of input records in this section.The first type of input record specifies the data for the computational nodes in the virtual machine. The data consist of the name of the computational node followed by a variable number of PVM configuration parameters that are sent to the computational node when it is added to the virtual machine. The names of the computational nodes in the virtual machine become user defined keywords. If these input records are omitted, all processes are executed on the processor where the executive program is executed using default values of all PVM configuration parameters. The second type of record in this section specifies the global wait time for communication between the processes. If this record is omitted, the default wait time is infinite.

The format of the first type of records in this section of the input deck is:

W1(A)	Name of computational node
W2(A)	First PVM configuration parameter
W3(A)	Second PVM configuration parameter.

....

See the PVM documentation for the description of the available PVM configuration parameters.

The format of the second type of record for this section of the input decks is:

W1(A)	Keyword ‘wait’
W2(R)	Global wait time (s). This is the amount of time to wait while listening to receive a message from another code or to wait to receive the acknowledgment to a message that has been sent to another coupled code. A value of -1.0 indicates an infinite wait time. This option should be used carefully because a deadlock might result if there is an error in the coupled computation.
W3(I)	Debug flag. A value of 1 indicates that each coupled task will be started up in a debugger window. The debugger to be used depends upon the architecture being used to execute the virtual machine and the compiler used to compile the child processes. This input is intended for code development and should not be used for production runs. In addition, the global wait time specified by the previous word on this card should be set to -1.0 (infinite wait time) so that the coupled tasks do not ‘time out’ while the user is examining the state of the coupled codes in the debugger.

The format of the third and last type of record for the ‘virtual’ section of the input deck is:

W1(A)	Keyword ‘write’
W2(A)	Filename. The information normally written to the ‘standard output’ file for the PVMEXEC program and all child processes will be written to this file instead of the default file named pvml.<uid> located on the /tmp directory or on the directory specified in the PVM_TMP environmental directory. The <uid> is the user identification number of the user executing the coupled simulation.

The data in the first section of the input deck are used to generate a PVM host file for starting the virtual machine.

A-3 Processes Section of Input Deck

The second section of the input deck is required and specifies the simulation codes that are to be executed using the virtual machine along with where they are to be executed. This section of the input deck is delimited by the keyword ‘processes’. This keyword is followed by subsections which are delimited by the names of the computational nodes in the virtual machine that were specified in the ‘virtual’ section of the input deck. Each of these subsections contains a variable number of input records describing the simulation codes to be executed on that computational node.

An input record consists of a user name for the process, the keyword ‘synchronous’ or ‘asynchronous’, and the name of an executable file followed by any command line parameters needed by the executable. The keyword ‘synchronous’ denotes that the time step to be used by the code is controlled by the executive program. The keyword ‘asynchronous’ denotes that the code may select its own time step size subject to the restriction that it must exchange data with other processes at fixed, executive program specified times. Synchronous time step control is required for semi-implicit, control system, and kinetics coupling. Synchronous or asynchronous time step control may be used for explicit coupling. In addition, the executive program controls printing, plotting and restart information generation for synchronous processes while it only controls restart information generation for asynchronous processes. For RELAP5-3D, the parameters would be things like the name of the input file, the name of the output file, etc. The format of this record is:

W1(A)	User name for process
W2(A)	Keyword ‘synchronous’ or ‘asynchronous’
W3(A)	Name of executable file
W4(A)	First command line parameter
W5(A)	Second command line parameter.

A second input record may be included for each named process. This record begins with the user name of the process followed by the keyword ‘uses’, which in turn is followed by an integer that specifies the number of threads of execution to use for the process. Any words after the third word are ignored. This record is used for multi-threaded executables. The format of this second type of record is:

W1(A)	User name of process
W2(A)	Keyword ‘uses’
W3(I)	Number of threads to be used by this process
W4(A)	Keyword ‘threads’. This word is optional.

A third input record may also be included for each named process. This record contains a filename and the information normally written to the 'standard output' file by this process will be written to this file instead. The format of this record is:

W1(A)	User name of process
W2(A)	Keyword 'writes'
W3(A)	Filename. The information normally written to the 'standard output' file by the process names in this input record will be written to this file instead of the default file named pvml.<uid> located on the /tmp directory or on the directory specified in the PVM_TMP environmental variable. The <uid> is the identification number of the user executing the coupled simulation.

A-4 Simulation Section of Input Deck

The third section of the input deck is optional and provides information about the simulation being executed. This section of the input deck begins with the keyword 'simulation'. There can be up to three optional input data records in this section. The first input data record begins with the keyword 'name', followed by a string value that describes the simulation being executed.

W1(A)	Keyword 'name'
W2(A)	String containing description of simulation.

The second input data record begins with the keyword 'restart' followed by the word 'time' followed by a decimal number. The decimal number is the time from which to restart the coupled computation. A value of -1.0 indicates that the restart is to begin from the last restart record from a previous run. If this input record is absent from the input deck, a new problem is assumed.

W1(A)	Keyword 'restart'
W2(A)	Keyword 'time'
W3(R)	Time from which to restart the previous simulation (s).

The third optional input data record begins with the keyword 'start', followed by the word 'time' followed by a real number. If this input record in absent from the input deck, a start time of zero is assumed for a new run and the start time is obtained from the restart record for a restart run. If the problem type is new and this input record is input, the start time is set to the value in this input record. If the problem type is restart, then the only valid value for the start time is zero. This is used to reset the simulation time when switching from a steady state run to a transient run on a restart.

W1(A)	Keyword ‘start’
W2(A)	Keyword ‘time’.
W3(R)	Start time of simulation run (s).

If this entire section is omitted from the input deck, a new run starting from time zero is assumed and the name of the simulation is blank.

A-5 Message Section of Input Deck

The fourth section of the input deck is required and describes the messages that are to be sent between the individual processes. The section begins with the keyword ‘messages’. The data are divided into subsections denoted by the keywords ‘explicit’, ‘semi-implicit’, ‘kinetics’ and ‘control’. The individual input records in each subsection begin with the user name of a process, the keyword ‘sends’ or ‘receives’, and the user name of the process receiving or sending the data as appropriate given the keyword in the second position in the input record. These data are followed by pairs of identifiers which specify what data to send (or receive). The data identifier consists of a component/variable name and a volume/junction number or component number as appropriate. The component name is used with semi-implicit thermal-hydraulic coupling and kinetics coupling, and the variable name is used for explicit thermal- hydraulic coupling and control system coupling. The format of this input record is:

W1(A)	Computational process name
W2(A)	Keyword. Enter ‘sends’ or ‘receives’
W3(A)	Computational process name
W4(A)	Name of first component/variable/keyword
W5(I)	Component or volume/junction number.

....

Input cards for explicit thermal-hydraulic coupling messages may specify either a component name or a variable name while input cards for semi-implicit thermal-hydraulic coupling messages should only specify component names. Any type of volume or junction component may be the source of data being sent to another process. Components receiving data from a coupled process should be boundary component types like time dependent volumes or time dependent junctions whose conditions are determined at the beginning of a time step advancement and would normally remain unchanged during the time step. The conditions in the coupling time dependent volumes and time dependent junctions for semi-implicit thermal-hydraulic coupling change during a time step advancement as a result of the semi-implicit coupling methodology while conditions in boundary coupling components for explicit thermal-hydraulic coupling remain unchanged during the time step advancement.

Input cards for kinetics messages may contain reserved keywords depending on whether the message is sending data to a coupled process or receiving data from a coupled process. Kinetics messages sending data to a coupled process may name volumes using the component name or heat structures using the keyword ‘heatstr’. These components must be present in the input deck for the process sending the message. The keyword ‘power’ is used to specify the reactor power from the point kinetics model and the keyword ‘zone’ is used to specify the power in a zone from the multi-dimensional kinetics model. The parameter for the keyword ‘power’ is 0 (zero). The keywords ‘table’ or ‘cntrlvar’ can be used to send the output from the table or control variable specified. Kinetics messages specifying data to be received by a process use the keywords ‘volume’ and ‘heatstr’ to specify that volume data and heat structure data are being received. The parameter for volume data must be less than 1000000 and the parameter for heat structure data must be less than 10000. The volumes and heat structures specified in kinetics receive messages must only be referenced by the point or multi-dimensional kinetics models.

Power data are received by specifying the keyword ‘power’ or ‘zone’. If ‘table’ or ‘cntrlvar’ values are being sent, the corresponding receive should be specified as ‘power’ (‘zone’ could also be used but is not recommended). The parameter for ‘power’ is 0 (zero). The data specified by the keywords ‘power’ and ‘zone’ should only be referenced by the heat structure source cards, Cards 1CCCG701 through 1CCCG799 for heat structures contained in the input deck of the process receiving the data.

Input cards for control system coupling which specify data to be sent may contain any data item that can be used as an input value to a control variable (See Section 3.1 of Appendix A of Volume II of the RELAP5-3D manual). The corresponding data specifier in the receiving process must be an external function control block or an interactive variable.

In addition to the required input records specifying the contents of the messages, optional asymmetrical wait times may be specified for the communication between each pair of processes for each category of communication (i.e., explicit, semi-implicit, kinetics, and/or control). These wait times are specified by inserting the following input record in the appropriate subsection of the ‘messages’ section of the input deck. The format of the wait time input record is:

W1(A)	Name of process awaiting message
W2(A)	Keyword ‘awaits’
W(A)	Name of process to wait upon
W(R)	Wait time (s).

The wait time is set on a message by message basis and is used as the wait time for the acknowledgment if the message is a send message and is the wait time if the message is a receive message. The global wait time is used if the wait time input record is not input for a given pair of processes. The global wait time is used for communication between the executive process and the other processes in the virtual machine.

The last type of input card in the 'messages' section can only appear in the explicit subsection of the messages section of the input file. This card is optional and controls what type of explicit thermal-hydraulic coupling is used between the processes named on the card. By default, parallel explicit thermal-hydraulic coupling is used between processes not named on this type of input card. If this card is present, the explicit coupling between the two processes named on the card is sequential explicit thermal-hydraulic coupling and the card defines the leader and follower processes for the sequential explicit thermal-hydraulic coupling. The format of the input record is:

W1(A)	Name of leader process in sequential explicit thermal-hydraulic coupling
W2(A)	Keyword 'preceeds'
W3(A)	Name of follower process in sequential explicit thermal-hydraulic coupling.

A-6 Time Step Information

The last section of the input file contains the time step information for the coupled computation. This section is delimited by the keyword 'timesteps'. The data in this section of the input deck describe a series of one or more time intervals with the maximum and minimum time step sizes for the interval along with the number of maximum time steps between plot records, print records, restart records, and explicit asynchronous coupling. A negative entry for any of the edit frequencies (W4, W5, W6, and W7) will cause the corresponding item not to be executed during that time interval. The end times on these input records must be in ascending order.

W1(R) Time interval end time (s)

W2(R) Minimum time step size (s)

W3(R) Maximum time step size (s)

W4(I) Control option. This word has the packed format dtt. It is not necessary to input leading zeros.

The digit d, which represents a number from 0 through 7, can be used to obtain extra output at every hydrodynamic time step. The number is treated as a three-bit binary number. If no bits are set (i.e., the number is 0), the standard output at the requested frequency using the maximum time step is obtained (see words 5 and 6 of this card). If the number is nonzero, output is obtained at each successful time step, and the bits indicate which output is obtained. If the first bit from the right is set (i.e., d = 1 if the other bits are not set), major edits are obtained every successful time step. If the second bit from the right is set (i.e., d = 2 if the other bits are not set), minor edits are obtained every successful time step. If the third bit from the right is set (i.e., d = 4 if the other bits are not set), plot records are written every successful time step. These options should be used carefully, since considerable output can be generated.

The digits ss, which can represent a number from 0 to 99, can be used to modify the solution controls for semi-implicitly coupled processes. The only values permitted at this time are 0 and 1. A value of 1 indicates that the mass error time step control in RELAP5-3D is to be disabled for semi-implicit coupling and a value of 0 means that the individual coupled processes use the time step and/or solution controls indicated on their input cards. The value of this word can be used to override parts of the solution controls for semi-implicitly coupled codes and the response of each code is code specific. RELAP5-3D responds by ignoring the mass error time step control as indicated earlier.

W5(I) Minor edit and plot frequency (-). This is the number of maximum (requested) time advances per minor edit to the printed output and write of plot information to the restart-plot file.

W6(I) Major edit frequency (-). This is the number of maximum (requested) time advances per major edit to the printed output.

W7(I) Restart frequency (-). This is the number of maximum (requested) time-advances per write of restart information to the restart-plot file.

W8(I) Explicit asynchronous coupling frequency (-). This is the number of maximum (requested) time advances per exchange of data between explicitly coupled codes.