

**ADDENDUM TO THE SOFTWARE QUALIFICATION ASSESSMENT
DOCUMENTS FOR THE PARCS-SPECIFIC DATA MAP ROUTINE**
Document Nos. PU/NE-98-11, PU/NE-98-12, PU/NE-98-13, and PU/NE-98-30

R. Matthew Miller, Thomas J. Downar, Douglas A. Barber

School of Nuclear Engineering
Purdue University
W. Lafayette, IN 47907-1290

submitted to:

Division of Systems Technology
Office of Regulatory Research
Nuclear Regulatory Commission
Washington, D.C. 20555-001

Performed Under Subcontract to SCIENTECH, Inc.
Contract SC96-1026-001

November 1998

I. Introduction

This document provides a description of the changes to the PARCS-Specific Data Map Routine⁽¹⁾ (PDMR) necessary to incorporate the functionality described in the Software Requirements Specification (SRS) for the TRAC-M-Specific Data Map Routine⁽²⁾ (TDMR). It should be noted that this document is intended to support the PDMR as described by the Software Qualification Assessment (SQA) documents pertaining to the coupling of RELAP5 and PARCS. However, a complete description of the additional functionality and implementation thereof of the TRAC-M/PARCS coupled code is provided by this document and the SQA documents for the PDMR together.

Minor modifications must be made to the existing PDMR in order to incorporate the functionality required for the TRAC-M/PARCS coupled code. This functionality includes that of the RELAP5/PARCS coupled code as well as additional features made possible through the use of the thermal-hydraulics code TRAC-M. Each additional feature or modification will be described briefly in Section II, which supplements the SRS for the PDMR. The testing of the additional functionality will be outlined in Section III, which specifies the changes in the Qualification Assessment Test Plan (QATP) for the PDMR⁽³⁾. Finally, the changes in the design and implementation of the PDMR will be described in Section IV, which presents modifications to the Software Design and Implementation Document (SDID) for the PDMR⁽⁴⁾ and the modifications already incorporated into the PDMR⁽⁵⁾ since the original RELAP5/PARCS delivery..

II. Addendum to the SRS for the PDMR

The SRS document should provide an analysis of requirements and definitions of specifications for the code being developed or modified. The subsections below describe the functionality to be added/modified in the PDMR currently used for the coupled RELAP5/PARCS code. The mapping input file format should be modified to reflect inherent differences between TRAC-M and RELAP5. Steady-state initialization capability must be available functionality in the PDMR in order to allow this type of calculation in the coupled TRAC-M/PARCS code. Coding for performing a boron criticality search must be added to make use of this PARCS feature in the framework of the coupled code. Finally, complete restart logic needs to be available in the PDMR in order to coordinate with the restart logic of TRAC-M.

II.A. Mapping Input File Format

The mapping input file, MAPTAB, as read by the PDMR requires modification in order to specify the mapping information more clearly for the coupled TRAC-M/PARCS code. The current format for MAPTAB is shown below, taken from the SRS for the PDMR:

Table 1: Thermal-Hydraulic Volume to Neutronic Node Mapping

TRAC-M Volume Composite ID	PARCS Node Number	Weighting Factor
----------------------------	-------------------	------------------

II.B. Steady-State Initialization

The capability of TRAC-M to perform a steady-state initialization calculation must be paralleled by the attendant ability of the coupled TRAC-M/PARCS code to perform the same. This feature has been added to the coupled RELAP5/PARCS code, and provides the basis for this capability in the coupled TRAC-M/PARCS code. A complete description of the coding modifications is provided in the additional documentation for the PDMR⁽⁵⁾.

II.C. Boron Criticality Search

Boron transport calculations are to be handled by TRAC-M, although a boron criticality search may be performed by PARCS during the steady-state eigenvalue calculation. This calculation will define the global boron concentration that is necessary to give a core-wide effective multiplication constant, k_{eff} , of unity. This solution should be reflected in the boron transport calculation being performed in TRAC-M. After PARCS calculates the correct boron concentration, a boron adjustment ratio must be computed and passed to TRAC-M through the existing control buffer interface of the coupled code. The TDMR will then adjust the boron present in and injected into the system by this ratio to reflect the boron concentration calculated by the PARCS boron criticality search.

The TDMR sends the cell-dependent boron concentration to the General Interface⁽⁶⁾, and subsequently, the PDMR, through the time-dependent thermal-hydraulic data vector. When these concentrations are passed to the General Interface, the permutation matrices are applied to the elements of the vector containing the boron concentration. The PDMR calculates a core-averaged boron concentration based on the node-dependent boron concentration sent from the General Interface. This core-averaged boron concentration is then used by PARCS as an initial guess for its boron criticality search calculation. When this calculation has converged, the critical boron concentration will be used to compute the ratio of the critical boron concentration to the initial guess. This ratio will be sent to TRAC-M via the General Interface in the neutronic data type-dependent buffer for double precision floating point variables, `r8bufn`. The TDMR will then adjust the boron concentration in the modeled system to the boron concentration sent from the PARCS criticality search by multiplying all boron concentration variables by this ratio. In addition, any entries in time-dependent boron concentration tables in the TRAC-M data structure must be adjusted by this ratio to prevent step changes in boron injection at the beginning of the transient calculation.

II.D. Restart Logic

The first implementation of the PDMR in the coupled RELAP5/PARCS code did not include restart logic since PARCS did not have the capability to perform restart calculations. Such functionality has since been incorporated into PARCS, and as such, restart logic was added to the PDMR. This feature was made in the coupled RELAP5/PARCS code after the code deliv-

ery date and provides the basis for implementing this feature in the coupled TRAC-M/PARCS code. A complete description is provided in the additional documentation for the PDMR⁽⁵⁾.

II.E. Additional Issues

Several modifications will be made to the PDMR source code, and additional functionality will be incorporated as described in Section IV. These modifications require that the initial and time-dependent control buffers be changed to reflect the additional features of the coupled TRAC-M/PARCS code.

III. Addendum to the QATP for the PDMR

The QATP is to outline the method for verifying and validating the functionality developed from the description in the SRS. The test plan should also demonstrate compliance with the requirements delineated in the SRS. Since a rigorous test plan was developed for the PDMR in the RELAP5/PARCS coupling, only the parts of the PDMR which were changed for the TRAC-M/PARCS implementation need to be thoroughly tested. The subsections below outline the testing procedure for verifying and validating the additional functionality of the PDMR.

The input deck to be used for testing the modified PDMR will be based on the typical PWR input deck used during the verification of the TDMR functionality. This PARCS input deck will be used with a modified Westinghouse 4-loop steady-state initialization input deck for TRAC-M. The coupled TRAC-M/PARCS code will be used for verifying the additional PDMR functionality.

III.A. Mapping Input File Format

The changes to the format of the mapping input file MAPTAB were described in Section II.A. The modifications made to the subroutines `Read_Mat()` and `Build_Mat()` need to be tested for accuracy as well as consistency. This modification simply removes the burden of calculating the composite ID number from the user and places this responsibility on the module used for reading and processing mapping input file data, `Map_Input`. Therefore, verification that the composite ID numbers are being calculated correctly is the only testing necessary for this portion of the QATP. This may be easily accomplished by compiling the PARCS/PDMR code with the preprocessor definition `FEVAL` present. This preprocessor directive activates the portions of the source code relating to this functional evaluation, which includes coding for writing the initial and time-dependent neutronic control buffers to the specified output file. The composite ID numbers written to this file must match the results obtained when calculating these IDs by hand.

III.B. Steady-State Initialization

The testing for steady-state initialization logic in the PDMR is performed in the additional documentation provided for the PDMR⁽⁵⁾.

III.C. Boron Criticality Search

The verification of the boron criticality search functionality will be shown by verifying that the boron adjustment ratio is computed and passed to the General Interface, and subsequently, the TDMR. This value will be passed in the data-type-dependent control buffers during a steady-state initialization calculation, and verification of this requirement will be determined using the same technique as will be used for verification of the composite ID number calculation. The time-dependent data-type-dependent control buffers will be written to the specified output file and checked for the presence of the critical boron concentration which was calculated in the previous steady-state "time step." In addition, a test will be performed to ensure that PARCS disables the critical boron concentration search if the TDMR indicates that the calculation is inconsistent with the TRAC-M model. If this situation occurs, the tenth variable in the logical buffer of the initial thermal-hydraulic control buffer should have a value of `.FALSE.` and PARCS should not be computing a critical boron concentration.

III.D. Restart Logic

The testing for restart logic in the PDMR is performed in the additional documentation provided for the PDMR⁽⁵⁾.

IV. Addendum to the SDID for the PDMR

The purpose of the SDID is to describe the components and sub-components of the software design, including databases and internal interfaces, such that the requirements are satisfied. All modifications made to the PDMR code as it stands in the delivered RELAP5/PARCS implementation are described and discussed below. Furthermore, all new variables needed to accommodate the additional functionality are described below. Appendix A contains a subroutine report indicating which subroutines were added, deleted, or modified, and a brief discussion of each change.

IV.A. Mapping Input File Format

The modifications to the mapping input file format require modifications to the subroutine which reads this file. Module `Map_Input` contains subroutine `Read_Mat()`, which is responsible for reading the mapping input file, `MAPTAB`. The old file format required only three columns of data in both `%TABLE1` and `%TABLE2`, the thermal-hydraulic to neutronic mapping information

and the heat structure to neutronic mapping information, respectively. The partial FORTRAN code for allocating space for and reading these three columns is shown below:

```

c Variable Declaration:
  INTEGER(4), ALLOCATABLE  ::  vol1(:), vol2(:)
  INTEGER(4), ALLOCATABLE  ::  hs1(:), hs2(:)
  REAL(4), ALLOCATABLE    ::  vol3(:), hs3(:)

c Perform 2 loops:  first loop to determine amount of space to
c be allocated, second loop to read the data.
  errinp = .FALSE.
  DO 20 iread=1,2
    OPEN(20,file='MAPTAB',status='old')

    IF (iread .EQ. 2) THEN
      ALLOCATE( vol1(nvolr), vol2(nvolr), vol3(nvolr))
      ALLOCATE( hs1(nhsr), hs2(nhsr), hs3(nhsr))
    ENDIF

c Read volume to neutronic node mapping information.
    IF (table .EQ. 'TABLE1') THEN
      nvolr=nvolr+1
      IF (iread .EQ. 2)
&        READ(line,*,err=900,end=900)
&        vol1(nvolr), vol2(nvolr), vol3(nvolr)

c Read heat structure to neutronic node mapping information.
    IF (table .EQ. 'TABLE2') THEN
      nhsr=nhsr+1
      IF (iread .EQ. 2)
&        READ(line,*,err=900,end=900)
&        hs1(nhsr), hs2(nhsr), hs3(nhsr)

```

The new format requires that both these tables contain five columns each, as specified in Section II.A. Because the coding shown above is expecting only three columns of data, it must be modified to reflect the two additional data columns. The modified coding is shown below:

```

c Variable Declaration:
  INTEGER(4), ALLOCATABLE  ::  vol1(:), vol2(:), vol3(:), vol4(:)
  INTEGER(4), ALLOCATABLE  ::  hs1(:), hs2(:), hs3(:), hs4(:)
  REAL(4), ALLOCATABLE    ::  vol5(:), hs5(:)

c Perform 2 loops:  first loop to determine amount of space to
c be allocated, second loop to read the data.
  errinp = .FALSE.
  DO 20 iread=1,2
    OPEN(20,file='MAPTAB',status='old')

    IF (iread .EQ. 2) THEN
      ALLOCATE( vol1(nvolr), vol2(nvolr), vol3(nvolr))
      ALLOCATE( vol4(nvolr), vol5(nvolr))
      ALLOCATE( hs1(nhsr), hs2(nhsr), hs3(nhsr))
      ALLOCATE( hs4(nhsr), hs5(nhsr))

```

```

ENDIF

c Read volume to neutronic node mapping information.
  IF (table .EQ. 'TABLE1') THEN
    nvolr=nvolr+1
    IF (iread .EQ. 2)
&       READ(line,*,err=900,end=900)
&       vol1(nvolr), vol2(nvolr), vol3(nvolr),
&       vol4(nvolr), vol5(nvolr)

c Read heat structure to neutronic node mapping information.
  IF (table .EQ. 'TABLE2') THEN
    nhshr=nhshr+1
    IF (iread .EQ. 2)
&       READ(line,*,err=900,end=900)
&       hs1(nhshr), hs2(nhshr), hs3(nhshr)
&       hs4(nhshr), hs5(nhshr)

```

The modifications to the mapping input file format also move the burden of calculating a composite ID number from the user to the Map_Input module. Subroutine Build_Mat() will be modified to calculate the composite ID numbers for both thermal-hydraulic cells and heat structure axial levels. This subroutine previously sorted the first two columns of data from both %TABLE1 and %TABLE2 in increasing order. The partial code listing as implemented in the coupled RELAP5/PARCS code for processing %TABLE1 input is shown below:

```

c Allocate temporary arrays used to store the distinct
c volume numbers and the distinct node numbers corresponding
c to the volumes.
  ALLOCATE( itmp1(nvolr))
  ALLOCATE( itmp2(nvolr))

  itmp1(1)=vol1(1)
  itmp2(1)=vol2(1)

c Global permutation matrices.
c
c T/H volume -> Neut. node & Neut. node -> T/H volume portions
c of the permutation matrices.
  DO 190 i=1,nvolr
c form array of matrix values (matvalth, matvaln)
  DO 170 k=1,nsbth2n
    l = (k-1)*nvolr + i
    matvalth(l)=vol3(i)
170  CONTINUE
  DO 180 i=1,nsbn2th
    l = (k-1)*nvolr + i
    matvaln(l)=vol3(:)
180  CONTINUE
190 CONTINUE

```

The partial code listing for reading %TABLE2 input is shown below:

```
c Allocate temporary arrays used to store the distinct
c heat structure numbers and the distinct node numbers
c corresponding to the heat structures.
```

```
  ALLOCATE( itmp1(nhsr))
  ALLOCATE( itmp2(nhsr))
```

```
  itmp1(1)=hs1(1)
  itmp2(1)=hs2(1)
```

```
c Global permutation matrices.
```

```
c
```

```
c HS -> Neut. node & Neut. node -> HS portions of the
c permutation matrices.
```

```
  DO 280 i=1,nhsr
```

```
c form array of matrix values (matvalth, matvaln)
```

```
  DO 260 k=1,nsubhs2n
    l = nsubth2n*nvolr+(k-1)*nhsr + i
    matvalth(l)=hs3(i)
```

```
260  CONTINUE
```

```
  DO 270 k=1,nsubn2hs
    l = nsubn2th*nvolr + (k-1)*nhsr + i
    matvaln(l)=hs3(i)
```

```
270  CONTINUE
```

```
280  CONTINUE
```

It should be noted that the arrays itmp1 and itmp2 are deallocated before the coding to process %TABLE2 input is executed.

To reflect the new structure of the mapping input file, the coding for processing %TABLE1 input has been changed to the following:

```
c Allocate temporary arrays used to store the distinct
c volume numbers and the distinct node numbers corresponding
c to the volumes.
```

```
  ALLOCATE( itmp1(nvolr))
  ALLOCATE( itmp2(nvolr))
```

```
c Combine the data in vol1, vol2, and vol3 into the
c composite ID number and store in vol1.
```

```
  vol1(:) = vol1(:)*1000000 + vol2(:)*1000 + vol3(:)
```

```
c Copy data stored in vol4 to vol2.
```

```
  vol2(:) = vol4(:)
```

```
  itmp1(1)=vol1(1)
  itmp2(1)=vol2(1)
```

```
c Global permutation matrices.
```

```
c
```

```
c T/H volume -> Neut. node & Neut. node -> T/H volume portions
c of the permutation matrices.
```

```
  DO 190 i=1,nvolr
```

```
c form array of matrix values (matvalth, matvaln)
```

```

      DO 170 k=1,nsubth2n
        l = (k-1)*nvolr + i
        matvalth(l)=vol5(i)
170    CONTINUE
      DO 180 i=1,nsubn2th
        l = (k-1)*nvolr + i
        matvaln(l)=vol5(:)
180    CONTINUE
190  CONTINUE

```

By modifying the source code in this manner, the composite ID number is formed from the data in the first three columns of data and stored in the `vol1`, which was previously the RELAP5 volume numbers. The neutronic node numbers corresponding to each volume number are stored in `vol2`. The sorting routine following the lines shown above sorts only two vectors, `vol1` and `vol2`. This modification, while destroying the data originally stored in `vol1` and `vol2`, minimizes the overall modifications to this subroutine. The same strategy is used for modifying the coding for processing the `%TABLE2` input:

```

c Allocate temporary arrays used to store the distinct
c heat structure numbers and the distinct node numbers
c corresponding to the heat structures.
      ALLOCATE( itmp1(nhsr))
      ALLOCATE( itmp2(nhsr))

c Combine the data in hs1, hs2, and hs3 into the
c composite ID number and store in hs1.
      hs1(:) = hs1(:)*1000000 + hs2(:)*1000 + hs3(:)

c Copy data stored in hs4 to hs2.
      hs2(:) = hs4(:)

      itmp1(1)=hs1(1)
      itmp2(1)=hs2(1)
c Global permutation matrices.
c
c HS -> Neut. node & Neut. node -> HS portions of the
c permutation matrices.
      DO 280 i=1,nhsr
c form array of matrix values (matvalth, matvaln)
      DO 260 k=1,nsubhs2n
        l = nsubth2n*nvolr+(k-1)*nhsr + i
        matvalth(l)=hs5(i)
260    CONTINUE
      DO 270 k=1,nsubn2hs
        l = nsubn2th*nvolr + (k-1)*nhsr + i
        matvaln(l)=hs5(i)
270    CONTINUE
280  CONTINUE

```

It should be noted that allowing the data originally in `vol1`, `vol2`, `hs1`, and `hs2` to be destroyed reduces the modifications to the coding building the matrix values to changing `vol3`

references to vol15 for thermal-hydraulic to neutronic mapping and changing hs3 references to hs5 for heat structure to neutronic mapping.

IV.B. Steady-State Initialization

The logic to perform a steady-state initialization calculation is described in the additional documentation for the PDMR⁽⁵⁾.

IV.C. Boron Criticality Search

Coding modifications are required in order to accommodate boron criticality search capability in the coupled TRAC-M/PARCS code. Specifically, a single logical variable must be added to the initial neutronic control buffer in order to indicate to the TDMR and TRAC-M that a boron criticality search is being performed. A logical variable must also be added to the initial thermal-hydraulic control buffer in order to inform the PDMR whether a critical boron concentration calculation is permitted with the TRAC-M input deck. Furthermore, a double precision floating point variable must be added to the time-dependent neutronic control buffer in order to pass the boron adjustment ratio to the TDMR. Modifications to the PDMR are also necessary to compute this ratio. The coding shown below is a partial listing from subroutine PDMR_Buf_Init() in module PDMR_Init_Calc, which packs the initial neutronic control buffer:

```
c Pack the PARCS control buffers.
  lbufn(1) = errcalcn
  lbufn(2) = errdatan
  lbufn(3) = errpvmn
  lbufn(4) = errdatagi
  lbufn(5) = errpvmgi
  lbufn(6) = done
  lbufn(7) = recvn
  i4bufn(1) = itrip
  i4bufn(2) = nvol
  i4bufn(3) = nhs
  i4bufn(4) = ncolth2n
  i4bufn(5) = nrown2th
  i4bufn(6:nvol+5) = ivol(:)
  i4bufn(6+nvol:nvol+nhs+5) = ihs(:)
```

The coding shown above must be changed to incorporate the additional logical variable indicating whether a boron criticality search is to be performed. The proposed coding is shown below:

```
c Pack the PARCS control buffers.
  lbufn(1) = errcalcn
  lbufn(2) = errdatan
  lbufn(3) = errpvmn
  lbufn(4) = errdatagi
  lbufn(5) = errpvmgi
```

```

lbufn(6) = done
lbufn(7) = recvn
lbufn(8) = srchppm
i4bufn(1) = itrip
i4bufn(2) = nvol
i4bufn(3) = nhs
i4bufn(4) = ncolth2n
i4bufn(5) = nrown2th
i4bufn(6:nvol+5) = ivol(:)
i4bufn(6+nvol:nvol+nhs+5) = ihs(:)

```

The additional variable, `srchppm`, is defined in the PARCS header file, `search.h` and defines whether a critical boron concentration calculation is to occur during the steady-state eigenvalue calculation. The TDMR will perform testing to determine whether a critical boron concentration calculation is permitted with the TRAC-M input deck being used. If not, the critical boron concentration calculation will be deactivated via a logical variable in the initial thermal-hydraulic control buffer. If a critical boron concentration will not be calculated by PARCS, this variable will be `.FALSE.`, and no boron concentration adjustments will be made to the TRAC-M model by the TDMR. Also, a logical in the initial thermal-hydraulic control buffer allows the TDMR to reject the request to perform a critical boron concentration search. The unpacking of this logical variable from the initial thermal-hydraulic control buffer is shown in the partial code listing below:

```

c Extract data from the RELAP5 control buffers.
  ALLOCATE( hsdscr(dimbuf(1))
  hsdscr(1:dimbuf(1)) = cbufn(1:dimbuf(1))
  errcalcth = lbufth(1)
  errdatath = lbufth(2)
  errpvmth = lbufth(3)
  errdatagi = lbufth(4)
  errpvmgi = lbufth(5)

c NOTE: General Interface only sends five logicals if errpvmgi
c       is TRUE. All other buffers are empty.
  IF (errpvmth .OR. errpvmgi) GOTO 999

  done = lbufth(6)
  rstrt = lbufth(8)
  ssinit = lbufth(9)
  srchppm = lbufth(10)
  if (rstrt) irstadv = i4bufth(1)
  time = r8bufth(1)
  sstime = time
  calcterm = done

```

The packing of the time-dependent neutronic control buffers must be modified to properly handle the passing of the boron adjustment ratio to the TDMR. The original coding shown below is from subroutine `PDMR_Buf_n2th()` in module `PDMR_Time_Calc`:

```

c Allocate memory for the control buffers.
c (memory deallocated in PDMR_Send_Bufn() )

```

```

dimbuf = (/0, 6, 0, 0, 0, 0/)
ALLOCATE( cbufn(dimbuf(1))
ALLOCATE( lbufn(dimbuf(2))
ALLOCATE( i2bufn(dimbuf(3))
ALLOCATE( i4bufn(dimbuf(4))
ALLOCATE( r4bufn(dimbuf(5))
ALLOCATE( r8bufn(dimbuf(6))

```

c Pack the PARCS control buffers.

c Note: The logicals relating to PDMR errors (errdatan, errpvmn)
c are packed into lbufn in subroutine PDMR_Map_n2th(). In
c addition, the PDMR_Map_n2th() sends the buffers prior to
c sending the vector of PARCS powers.

```

lbufn(1) = errcalcn
lbufn(4) = errdatagi
lbufn(5) = errpvmgi
lbufn(6) = done

```

Due to the additional variable in the time-dependent control buffer, the size of the element in array dimbuf corresponding to double-precision floating point variables must be incremented from 0 to 1. This new location will be used to store the boron ratio from the criticality calculation performed by PARCS. The new coding is shown below:

```

c Allocate memory for the control buffers.
c (memory deallocated in PDMR_Send_Bufn() )
dimbuf = (/0, 6, 0, 0, 0, 1/)
ALLOCATE( cbufn(dimbuf(1))
ALLOCATE( lbufn(dimbuf(2))
ALLOCATE( i2bufn(dimbuf(3))
ALLOCATE( i4bufn(dimbuf(4))
ALLOCATE( r4bufn(dimbuf(5))
ALLOCATE( r8bufn(dimbuf(6))

```

c Pack the PARCS control buffers.

c Note: The logicals relating to PDMR errors (errdatan, errpvmn)
c are packed into lbufn in subroutine PDMR_Map_n2th(). In
c addition, the PDMR_Map_n2th() sends the buffers prior to
c sending the vector of PARCS powers.

```

lbufn(1) = errcalcn
lbufn(4) = errdatagi
lbufn(5) = errpvmgi
lbufn(6) = done
IF (srchppm .AND. ssinit) THEN
  ppmratio = ppm / tppm
  r8bufn(1) = ppmratio
ENDIF

```

The variable ppm is contained in the PARCS header file cnt1.h and contains the current global boron concentration. The variable tppm is calculated in subroutine PDMR_Map_th2n in module PDMR_Time_Calc shown below:

```

m=4*volnode
ppm = 0.0
vcore = 0.0
DO 100 k=1,nz
  DO 90 l=1,nxy
    m=m+1
    ppm1(l.k) = vecthp(m)
    ppm = ppm + vecthp(m)
    vcore = vcore + volnode(l.k)
90  CONTINUE
100 CONTINUE
ppm = ppm/vcore
tppm = ppm

```

IV.D. Restart Logic

Implementation of restart logic in the PDMR is described in the additional documentation for the PDMR⁽⁵⁾.

IV.D.1. Initial Neutronic Control Buffer

The initial neutronic control buffer must be changed in order to pass the necessary information to the thermal-hydraulics code. The modified initial neutronic control buffer is shown below:

1. Indication of an error in PARCS (logical)
2. Indication of a data error in the PARCS-Specific Data Map Routine (logical)
3. Indication of a PVM error in the PARCS-Specific Data Map Routine (logical)
4. Indication of a data error in the General Interface (logical)
5. Indication of a PVM error in the General Interface (logical)
6. Indication of normal calculation termination (logical)
7. Indication of whether the PARCS-Specific Data Map Routine is sending the permutation matrices (logical)
8. Indication of whether PARCS will perform a boron criticality search during the steady-state eigenvalue calculation (logical)
9. TRAC-M trip ID number corresponding to a control rod trip (integer*4)
10. Number of thermal-hydraulic cells being coupled to neutronic nodes (integer*4)
11. Number of heat structure axial levels being coupled to neutronic nodes (integer*4)
12. Number of columns in thermal-hydraulic to neutronic matrix (integer*4)
13. Number of rows in neutronic to thermal-hydraulic matrix (integer*4)
14. Array of TRAC-M composite ID numbers for thermal-hydraulic cells being coupled to neutronic nodes (integer*4)
15. Array of TRAC-M composite ID numbers for heat structure axial levels being coupled to neutronic nodes (integer*4)

IV.D.2. Initial Thermal-Hydraulic Control Buffer

The initial thermal-hydraulic control buffer must be changed in order to pass the necessary information to the neutronics code. The modified initial thermal-hydraulic control buffer is shown below:

1. Array of heat structure geometry descriptors (character*6)
2. Indication of an error in TRAC-M (logical)
3. Indication of a data error in the TRAC-M-Specific Data Map Routine (logical)
4. Indication of a PVM error in the TRAC-M-Specific Data Map Routine (logical)
5. Indication of a data error in the General Interface (logical)
6. Indication of a PVM error in the General Interface (logical)
7. Indication of normal calculation termination (logical)
8. Indication of whether the TRAC-M-Specific Data Map Routine is sending the permutation matrices (logical)
9. Indication of whether this is a “restart” run (logical)
10. Indication of whether a steady-state initialization is to be performed (logical)
11. Indication of whether the boron criticality search is allowed (logical)
12. Time point at which calculation is beginning (real*8)

IV.D.3. Time-Dependent Thermal-Hydraulic Control Buffer

The time-dependent thermal-hydraulic control buffer does not need to be changed in order to incorporate the described functionality, but is listed here for completeness.

1. Indication of an error in TRAC-M (logical)
2. Indication of a data error in the TRAC-M-Specific Data Map Routine (logical)
3. Indication of a PVM error in the TRAC-M-Specific Data Map Routine (logical)
4. Indication of a data error in the General Interface (logical)
5. Indication of a PVM error in the General Interface (logical)
6. Indication of normal calculation termination (logical)
7. Indication that a restart edit should be performed (logical)
8. Indication that a major edit should be performed (logical)
9. Indication of a scram (logical)
10. Current time step size (real*8)

IV.D.4. Time-Dependent Neutronic Control Buffer

The time-dependent neutronic control buffer must be changed in order to pass the necessary information to the thermal-hydraulics code during the transient calculation. The modified time-dependent neutronic control buffer is shown below:

1. Indication of an error in PARCS (logical)
2. Indication of a data error in the PARCS-Specific Data Map Routine (logical)

3. Indication of a PVM error in the PARCS-Specific Data Map Routine (logical)
4. Indication of a data error in the General Interface (logical)
5. Indication of a PVM error in the General Interface (logical)
6. Indication of normal calculation termination (logical)
7. PARCS-calculated boron concentration adjustment ratio (real*8)

V. Summary

This document provided a description of the changes to the PDMR necessary to incorporate the functionality described in the SRS for the TDMR. Since these modifications/additions are minor, a separate set of SQA documents for the PDMR used in coupling TRAC-M and PARCS are not created. This document served as an addendum to the existing SQA documents for the PDMR and described the differences between the PDMR used for coupling PARCS with RELAP5 and the PDMR used for coupling PARCS with TRAC-M.

VI. References

1. D. Barber and T. Downar, "Software Requirements Specification for the PARCS-Specific Data Map Routine in the Coupled RELAP5/PARCS Code," Technical Report, PU/NE-98-11, Purdue University, (1998).
2. R. Miller, T. Downar, and D. Barber, "Software Requirements Specification for the TRAC-M-Specific Data Map Routine in the Coupled TRAC-M/PARCS Code," Technical Report, PU/NE-98-22, Purdue University, (1998).
3. D. Barber and T. Downar, "Qualification Assessment Test Plan for the PARCS-Specific Data Map Routine in the Coupled RELAP5/PARCS Code," Technical Report, PU/NE-98-12, Purdue University, (1998).
4. D. Barber and T. Downar, "Software Design and Implementation Document for the PARCS-Specific Data Map Routine in the Coupled RELAP5/PARCS Code," Technical Report, PU/NE-98-13, Purdue University, (1998).
5. D. Barber and T. Downar, "Modifications to PARCS/PDMR to Accomodate Restart and Steady-State Initialization in the Coupled Code," Technical Report, PU/NE-98-30, Purdue University, (1998).
6. D. Barber and T. Downar, "Software Requirements Specification for the General Interface in the Coupled Code," Technical Report, PU/NE-98-8, Purdue University, (1998).

Appendix A: Subroutine ReportSubroutines Deleted:

none

Subroutines Added:

none

Subroutines Modified:

Module Map_Input: added variables for storing the two additional columns of data in %TABLE1 and %TABLE2 of MAPTAB

Subroutine Read_Map: added coding for reading and storing the additional data columns

Subroutine Build_Mat: added coding for forming the composite ID number for thermal-hydraulic and heat structure components from the first three columns of data in %TABLE1 and %TABLE2 of MAPTAB; modified coding for forming the permutation matrix values to use vol5(:) and hs5(:) instead of vol3(:) and hs3(:)

Module PDMR_Init_Calc:

Subroutine PDMR_Buf_Init: added logical variable to initial neutronic and thermal-hydraulic control buffers to provide critical boron concentration calculation functionality

Module PDMR_Time_Calc:

Subroutine PDMR_Map_th2n: added coding for storing the global boron concentration calculated from the concentrations sent by the TDMR into tppm

Subroutine PDMR_Buf_n2th: added coding for computing and packing the boron adjustment ration, ppmratio

Module PDMR_Var_Decl:

Subroutine PDMR_Var_Decl: added declarations of tppm and ppmratio