

**RELAP5/MOD3.3 CODE MANUAL**  
**VOLUME VIII:**  
**PROGRAMMERS MANUAL**

**Nuclear Safety Analysis Division**

**March 2006**

**Information Systems Laboratories, Inc.**  
**Rockville, Maryland**  
**Idaho Falls, Idaho**

**Prepared for the**  
**Division of Systems Research**  
**Office of Nuclear Regulatory Research**  
**U. S. Nuclear Regulatory Commission**  
**Washington, DC 20555**

**Manual updated for RELAP5/MOD3.3Patch03**  
**Copyright © 2001-6 Information Systems Laboratories, Inc. All rights reserved.**

## **ACKNOWLEDGMENTS**

Development of a complex computer code such as RELAP5 is the result of team effort and requires the diverse talents of a large number of people. Special acknowledgment is given to those who pioneered and continue to contribute to the RELAP5 code, in particular, V. H. Ransom, J. A. Trapp, and R. J. Wagner. A number of other people have made and continue to make significant contributions to the continuing development of the RELAP5 code. Recognition and gratitude is given to the members of the INEL RELAP5 team:

V. T. Berta	C. E. Lenglade	R. A. Riemke
K. E. Carlson	M. A. Lintner	R. R. Schultz
C. D. Fletcher	C. C. McKenzie	A. S-L. Shieh
E. E. Jenkins	G. L. Mesina	R. W. Shumway
E. C. Johnsen	C. S. Miller	C. E. Slater
G. W. Johnsen	G. A. Mortensen	S. M. Sloan
J. M. Kelly	P. E. Murray	M. Warnick
H-H. Kuo	R. B. Nielson	W. L. Weaver
N. S. Larson	S. Paik	G. E. Wilson

The list of contributors is incomplete, as many others have made significant contributions in the past. Rather than attempt to list them all and risk unknowingly omitting some who have contributed, we acknowledge them as a group and express our appreciation for their contributions to the success of the RELAP5 effort.

The current list of contributors to the RELAP5 Program at SCIENTECH, Inc., and ISL, Inc. include:

Bill Arcieri	Don Fletcher	Doug Barber
Robert Beaton	Mark Bolander	Robert Copp
Byron Hansen	Scott Lucas	Glen Mortensen
Dan Prelewicz	Rex Shumway	Randy Tompot
Weidong Wang	David Caraher	

The RELAP5 Program is indebted to the technical monitors from the U. S. Nuclear Regulatory Commission and the Department of Energy-Idaho Operations Office for giving direction and management to the overall program. Those from the NRC include W. Lyon, Y. Chen, R. Lee, R. Landry, H. Scott, M. Rubin, D. E. Solberg, D. Ebert, S. Smith, T. Lee, V. Mousseau, and M. Rubin. Monitors from DOE-ID when the RELAP5 program was at the INEL include N. Bonicelli, C. Noble, W. Rettig, and D. Majumdar.

The technical editing of the RELAP5 manuals when the RELAP5 program was at the INEL was done by D. Pack and E. May.

Finally, acknowledgment is made of all the code users who have been very helpful in stimulating timely correction of code deficiencies and suggesting improvements.

## **ABSTRACT**

The RELAP5 code has been developed for the best-estimate transient simulation of light water reactor coolant systems during postulated accidents. The code models the coupled behavior of the reactor coolant system and the core for loss-of-coolant accidents and operational transients such as anticipated transient without scram, loss of offsite power, loss of feedwater, and loss of flow. A generic modeling approach is used that permits simulating a variety of thermal hydraulic systems. Control system and secondary system components are included to permit modeling of plant controls, turbines, condensers, and secondary feedwater systems.

RELAP5/MOD3.3 code documentation is divided into eight volumes: Volume 1 presents modeling theory and associated numerical schemes; Volume 2 details instructions for code application and input data preparation; Volume 3 presents the results of developmental assessment cases that demonstrate and verify the models used in the code; Volume 4 discusses in detail RELAP5 models and correlations; Volume 5 presents guidelines that have evolved over the past several years through the use of the RELAP5 code; Volume 6 discusses the numerical scheme used in RELAP5; Volume 7 presents a collection of independent assessment calculations; and Volume 8 presents the programmer's background information.



## CONTENTS

	Page
1      INTRODUCTION .....	1
2      CODING CONVENTIONS .....	3
2.1     Programming Language .....	3
2.2     FORTRAN 77 and Extensions .....	3
2.3     File Naming Conventions .....	3
2.4     Preprocessors .....	4
2.4.1    select Precompiler .....	4
2.4.2    cnv32 Preprocessor .....	5
2.5     Job Control Language (cshell).....	7
2.6     Static and Dynamic Storage .....	7
2.7     Input Deck Structure.....	7
2.8     Restart-Plot File Structure .....	7
3      PROGRAM STRUCTURE .....	9
3.1     Input Processing Subroutines .....	9
3.1.1    CVI Package Subroutines .....	11
3.1.2    R-level Subroutines .....	12
3.1.2.1   rrstd .....	13
3.1.2.2   rrestf .....	13
3.1.2.3   rchng .....	14
3.1.2.4   rtsc .....	14
3.1.2.5   rmiedt .....	15
3.1.2.6   rintrv .....	15
3.1.2.7   rtrip .....	15
3.1.2.8   rmoncn .....	15
3.1.2.9   rmflds .....	16
3.1.2.10   rcompn .....	16
3.1.2.11   rhtemp .....	18
3.1.2.12   rradht .....	18
3.1.2.13   rmdat .....	18
3.1.2.14   rrkin .....	19
3.1.2.15   rgntbl .....	19
3.1.2.16   rconvr .....	19
3.1.2.17   rusrvr .....	19
3.1.3    I-level Subroutines .....	19
3.1.3.1   itstck .....	19
3.1.3.2   icmpn1 .....	20
3.1.3.3   itrip .....	20
3.1.3.4   igntbl .....	20
3.1.3.5   icompn .....	20

3.1.3.6	ihtcmp.....	21
3.1.3.7	iradht .....	21
3.1.3.8	irflht.....	22
3.1.3.9	invhts.....	22
3.1.3.10	irkin.....	22
3.1.3.11	iconvr .....	22
3.1.3.12	imiedt .....	22
3.1.3.13	issi .....	23
3.1.3.14	iusrvr .....	23
3.1.3.15	wrplid .....	23
3.1.4	Card Number, Subroutine Name, and Common Block Connections .....	23
3.2	Transient Processing Subroutines.....	27
3.2.1	trnset .....	27
3.2.2	tran .....	28
3.2.2.1	dtstep .....	29
3.2.2.2	chklev .....	30
3.2.2.3	tstate .....	30
3.2.2.4	htadv.....	31
3.2.2.5	pzrlev.....	31
3.2.2.6	hydro .....	31
3.2.2.7	rkin .....	32
3.2.2.8	convar.....	32
3.2.3	trnfin .....	32
3.3	Hydrodynamics Subroutines .....	33
3.3.1	stcset .....	33
3.3.2	valve.....	34
3.3.3	acclev .....	34
3.3.4	volvel .....	34
3.3.5	tfront .....	35
3.3.6	phantv .....	35
3.3.6.1	eccmxv .....	36
3.3.6.2	pumphifreg .....	36
3.3.6.3	horizhifreg .....	36
3.3.6.4	verthifreg .....	37
3.3.6.5	wethif .....	37
3.3.6.6	dryhif .....	37
3.3.6.7	vstrathif .....	37
3.3.6.8	levelhif .....	37
3.3.6.9	jethif .....	38
3.3.6.10	filterhif .....	38
3.3.6.11	bugouthif .....	38
3.3.7	phantj .....	38
3.3.7.1	eccmxj .....	39
3.3.7.2	pumpdragreg .....	39
3.3.7.3	horizdragreg .....	39
3.3.7.4	vertdragreg .....	39
3.3.7.5	wetdrag .....	39
3.3.7.6	drydrag .....	40

3.3.7.7	adjustthif.....	40
3.3.7.8	chng18drag.....	40
3.3.7.9	filterdrag.....	40
3.3.7.10	finaldrag .....	40
3.3.7.11	bugoutdrag .....	40
3.3.8	fwdrag.....	40
3.3.8.1	viscos1.....	41
3.3.9	hloss .....	41
3.3.10	vexplt .....	41
3.3.10.1	pump and pump2.....	41
3.3.10.2	accum .....	42
3.3.10.3	turst.....	42
3.3.11	jchoke.....	42
3.3.11.1	gesub .....	42
3.3.11.2	gctpm.....	42
3.3.11.3	gctpmoody.....	43
3.3.12	ccfl .....	43
3.3.13	vfinl.....	43
3.3.13.1	preseq .....	43
3.3.13.2	syssol.....	44
3.3.13.3	packer.....	44
3.3.14	eqfinl.....	44
3.3.15	vimplt.....	45
3.3.16	pimplt.....	45
3.3.17	simplt .....	45
3.3.18	jprop .....	46
3.3.18.1	stdsp .....	46
3.3.18.2	dryer .....	46
3.3.18.3	gesep.....	46
3.3.18.4	stdry .....	46
3.3.18.5	gedry.....	46
3.3.18.6	hseflw .....	46
3.3.18.7	hzflow.....	47
3.3.19	brntrn.....	47
3.3.20	state .....	47
3.3.21	stacc .....	47
3.3.22	statep .....	47
3.3.22.1	classifyvols.....	47
3.3.22.2	supertosub .....	48
3.3.22.3	withair and nwwithair .....	48
3.3.22.4	withoutair .....	48
3.3.23	masserror.....	49
3.3.24	level.....	49
3.3.25	vlvela.....	49
3.3.26	htfinl.....	49
3.4	Heat Structure Subroutines.....	49
3.4.1	htadv .....	49
3.4.2	radht .....	50
3.4.3	qfmove .....	50

3.4.4	ht1tdp .....	50
3.4.5	htcond .....	51
3.5	Reactor Kinetics Subroutines .....	51
3.5.1	rkin .....	51
3.5.2	rrkin .....	51
3.5.3	irkin .....	51
3.6	Control System Subroutines .....	52
3.6.1	convar .....	52
3.6.2	rconvar .....	52
3.6.3	iconvr .....	52
3.7	Tricks of the Trade.....	52
3.7.1	voldat .....	52
3.7.2	jundat .....	52
3.7.3	lpdat .....	53
3.7.4	Loop over Systems .....	53
3.7.5	Loop over Volumes .....	53
3.7.6	Loop over Junctions.....	54
3.7.7	Loop over Volume Scratch Variables.....	54
3.7.8	Adding New Integers or Logicals to Dynamic Comdecks .....	54
3.7.9	Adding a New Default Plot Variable to the Default List .....	55
3.8	Output Files .....	55
3.8.1	Major Edits .....	55
3.8.2	Minor Edits .....	55
3.8.3	Strip Edits .....	56
3.8.4	Screen Edits .....	56
3.8.5	Snapit Edits.....	56
3.8.6	Debug Edits .....	56
3.8.7	Binary Restart-Plot Files.....	56
4	CODE DATABASE.....	57
4.1	Labeled Common Blocks .....	57
4.1.1	Main control database - contrl.hh - /contrl/ and /rstsum/.....	57
4.1.2	Dynamic storage block assignments - comctl.hh - /comctl/ .....	59
4.1.3	Dynamic storage pool - fast.hh - /fast/.....	61
4.1.4	Equivalenced fa and ia arrays in /fast/.....	62
4.1.5	Bit numbering convention in a packed word.....	63
4.1.6	Volume database - voldat.hh .....	64
4.1.7	Junction database - jundat.hh .....	65
4.1.8	Scratch database - scrtch.hh.....	67
5	FILES.....	69
5.1	Input.....	69
5.2	Output.....	69
5.2.1	Print file .....	69
5.2.2	Restart-plot file .....	69

6	ENVIRONMENTAL SUBROUTINES .....	71
6.1	CVI Package.....	71
6.2	INP Package .....	71
6.3	FTB Package .....	71
6.4	Thermodynamic Property Package.....	71
6.4.1	Steam Table Generator.....	71
6.4.2	Interpolation Subroutines.....	71
6.5	Buffered I/O Subroutines .....	71
6.6	Locf Subroutines .....	72
6.7	Date and Time .....	72
6.8	CPU Time .....	72
7	INSTALLATION AND MAINTENANCE .....	73
7.1	Computers.....	73
7.2	Directory Structure .....	73
7.2.1	relap .....	73
7.2.2	envrl .....	73
7.2.3	run .....	73
7.3	Installation Scripts.....	74
7.4	dinstls.....	74
7.4.1	denvrl .....	74
7.4.2	drelap .....	74
7.4.3	dstgxx.....	74
7.4.4	runx .....	74
7.5	Precompliers .....	74
7.5.1	select .....	74
7.5.2	cnv32 - mlist - in32 - in32end.....	74
7.5.3	Makefiles .....	74
7.5.4	bldmak script .....	74
7.5.5	Makeee - Liste .....	74
7.5.6	Maker - Listr .....	74
7.6	Libraries.....	74
7.6.1	envrl.a .....	74
7.6.2	relap.a.....	74
7.7	Revision Control System.....	74
8	DEBUGGING TOOLS.....	75
8.1	MakeNew Script.....	75
8.2	Diagnostic Debug Printouts.....	76
8.3	Dynamic Block Snapit Printouts .....	77

8.4	Additional Variables .....	77
8.4.1	2080xxxx Card Variables.....	77
8.4.2	Testda Variables .....	78
8.4.3	Extra Volume Variables (extvnn).....	78
8.4.4	Extra Junction Variables (extjnn).....	78
8.4.5	Extra System Variables (extsns).....	78
8.5	Flowcharter.....	78
9	CVI PACKAGE.....	81
9.1	Data Field Description.....	81
9.1.1	Decimal Fields.....	82
9.1.2	Hexadecimal Fields .....	84
9.1.3	Octal Fields.....	84
9.1.4	Hollerith or Non-Numeric Fields.....	84
9.2	Subroutine Usage.....	85
9.2.1	cvic - Decode One Card.....	86
9.2.2	cvirc - Read One Card and Call cvic .....	87
9.2.3	tcvi - Test cvic.....	87
10	COMDECK DESCRIPTIONS .....	89
10.1	cmpalf.hh .....	89
10.2	cmpdac.hh & cmpdacc.hh .....	90
10.3	cmpdat.hh & cmpdatec.hh.....	92
10.4	cmpdtv.hh & cmpdtvc.hh .....	95
10.5	cnvtpa.hh & cnvtpad.hh.....	97
10.6	comctl.hh & comctlc.hh .....	98
10.7	comlst.hh .....	100
10.8	cons.hh.....	100
10.9	contrl.hh & contrx.hh .....	101
10.10	convarc.hh & convarx.hh .....	106
10.11	eccmxcc.hh & eccmxcc.hh.....	109
10.12	fast.hh & fastc.hh.....	109
10.13	flood.hh.....	110
10.14	ftbcom.hh.....	110
10.15	gapvar.hh .....	112
10.16	genrl.hh & genrlc.hh.....	112
10.17	gentblc.hh & gentblx.hh .....	112
10.18	htrcom.hh & htrcomc.hh.....	113
10.19	htrflb.hh & htrflbc.hh .....	116

10.20	htscr1.hh .....	117
10.21	htscr2.hh & htscr2c.hh.....	118
10.22	htscr.hh .....	119
10.23	htsrcm.hh & htsrcmc.hh .....	120
10.24	intrac.hh & intracc.hh.....	126
10.25	invhtb.hh & invhtbc.hh.....	127
10.26	invtbl.hh & invtblc.hh.....	127
10.27	jundat.hh & jundatec.hh .....	128
10.28	lcntrl.hh & lcntrlc.hh .....	135
10.29	levtbl.hh & levtblc.hh.....	136
10.30	lpdat.hh & lpdatec.hh .....	136
10.31	lvectr.hh & lvectrc.hh.....	138
10.32	lvel.hh .....	139
10.33	machaf.hh & machas.hh .....	140
10.34	macheff.hh & maches.hh .....	140
10.35	machlf.hh & machls.hh.....	140
10.36	machof.hh & machos.hh .....	142
10.37	machss.hh .....	142
10.38	makmap.hh .....	142
10.39	maxmem.hh .....	143
10.40	miedtc.hh & miedtcl.hh .....	143
10.41	mtbls.hh & mtblc.hh .....	144
10.42	mxnfcd.hh & mtblc.hh.....	144
10.43	npacom.hh .....	145
10.44	plotpc.hh .....	146
10.45	przdat.hh & przdatac.hh .....	146
10.46	pumpblk.hh & pumpblkx.hh.....	146
10.47	radhtc.hh & radhtcc.hh .....	149
10.48	rcompc.hh and cmpalf.hh .....	150
10.49	rflhtc.hh .....	151
10.50	rkinc.hh and rkincc.hh .....	152
10.51	rknathb.hh and rknatbc.hh .....	161
10.52	rmadac.hh .....	161
10.53	rrkinc.hh .....	162

10.54	scrq.hh .....	163
10.55	scrtch.hh & scrtchc.hh .....	165
10.56	separ.hh & separc.hh .....	183
10.57	ssctrn.hh & ssctrnc.hh.....	184
10.58	ssiblk.hh & ssiblkc.hh.....	185
10.59	statc.hh & statcc.hh.....	185
10.60	statec.hh & statecc.hh .....	187
10.61	stcblk.hh & stcblkc.hh .....	189
10.62	stcom.hh & stcomc.hh .....	189
10.63	sysdate.hh & sysdatm.hh .....	190
10.64	tdpptr.hh.....	191
10.65	tmsrcm.hh .....	191
10.66	trnhlp.hh & trnhlpc.hh .....	192
10.67	trpbhk.h & trpbhk.c.hh .....	192
10.68	tsctlc.hh.....	194
10.69	tstpct.hh & tstpcte.hh.....	194
10.70	turbin.hh & turbinc.hh .....	196
10.71	ufilef.hh, ufiles.hh, & ufilesc.hh.....	197
10.72	usrvar.hh .....	198
10.73	voldat.hh & voldatc.hh .....	198
10.74	vreqs.hh & vreqd.hh .....	212
10.75	zalfag.hh .....	221
11	INP PACKAGE .....	223
11.1	User Aspects of INP .....	223
11.1.1	Title Cards.....	224
11.1.2	Comment Cards .....	224
11.1.3	End Cards.....	224
11.1.4	Data Cards .....	224
11.1.5	Example Cards.....	225
11.2	Programmer Aspects of INP .....	225
11.2.1	Sequential Expansion.....	226
11.2.2	Overlay Expansion.....	227
11.2.3	Dense Storage .....	227
11.2.4	Scattered Storage .....	227
11.2.5	efiless - File Unit Numbers.....	228
11.2.6	inp - Input Echo and Package Initialization.....	228
11.2.7	inp2 - Read One Dataset.....	231
11.2.8	inp4 - Read Multiple Datasets .....	232

11.2.9	inp5 - Read Vector Dataset .....	233
11.2.10	inp6 - Error Processing for inp2 .....	235
11.2.11	inp7 - Error Message Printing.....	235
11.2.12	inp8 - Count Unreferenced Cards .....	235
11.2.13	inp9 - Delete All Processed Cards .....	236
11.2.14	inp10 - Delete Specified Range of Processed Cards .....	236
11.2.15	inplnk - Card Number Search.....	236
11.2.16	inpmod - Data Type Identification.....	237
11.2.17	inppck - Pack Mode Words in List Array .....	238
11.2.18	inpukp - Unpack Mode Words from List Array .....	238
11.3	INP Summary .....	238
11.3.1	Error Message Summary .....	238
11.3.2	Word Structure Used in x11.....	239
11.3.2.1	Control Word Structure.....	239
11.3.2.2	Table Word Structure .....	239
11.3.2.3	Mode Indicator Word Structure .....	241
11.4	References .....	241
12	THERMODYNAMIC PROPERTY PACKAGE .....	243
12.1	stgh2o - Generate Water Properties .....	244
12.1.1	Input Data .....	244
12.1.2	Sample Input File.....	245
12.1.3	Binary Table Format .....	246
12.1.3.1	Table 1, Temperatures .....	248
12.1.3.2	Table 2, Pressures.....	248
12.1.3.3	Table 3, Saturation Properties vs. Temperature .....	248
12.1.3.4	Table 4, Saturation Properties vs. Pressure .....	249
12.1.3.5	Table 5, Single Phase Properties .....	250
12.2	Interpolation Subroutines .....	251
12.2.1	stread - Read Water Properties.....	252
12.2.2	sth2x0 - Saturated Properties, F(T).....	252
12.2.3	sth2x1 - Saturated Properties, F(T,x) .....	253
12.2.4	sth2x2 - Saturated Properties, F(P,x) .....	254
12.2.5	sth2x3 - Single Phase Properties, F(T,P) .....	256
12.2.6	sth2x4 - Water Properties, F(T,v).....	257
12.2.7	sth2x5 - Water Properties, F(P,h) .....	257
12.2.8	sth2x6 - Water Properties, F(P,u) .....	257
12.2.9	psatpd - Saturation Pressure or Temperature and dP/dT.....	258
12.3	Interpolation Procedures.....	258
12.4	References .....	260
13	RSTPLT FILE STRUCTURE .....	261
13.1	Rstplt Record Structure .....	262
13.2	Plotinf Record Structure .....	262
13.3	Plotalf Record Structure .....	263

13.4	Plotnum Record Structure .....	263
13.5	Plotrec Record Structure.....	264
13.6	Restart Record Structure.....	264
13.6.1	Restart Header Record.....	265
13.6.2	Restart Title Record .....	265
13.6.3	Restart Common Block Records .....	265
13.7	Summary.....	266
14	FTB PACKAGE .....	267
14.1	Overview .....	267
14.2	FTB Subroutines And Common Blocks.....	271
14.2.1	/fast/ - FTB SCM Storage Pool.....	272
14.2.2	/ftb/ - FTB Variables .....	272
14.2.3	dmpfil - Print a FTB File .....	273
14.2.4	dmplst - Print Bookkeeping Information for FTB Files .....	273
14.2.5	ftbchk - Buffered Input Output Synchronization Check.....	275
14.2.6	ftbcls - File Close.....	275
14.2.7	ftbcpy - File Copy .....	275
14.2.8	ftbdel - File Delete .....	275
14.2.9	fldmp - Print Bookkeeping Information for RELAP5 FTB Files.....	276
14.2.10	ftbdsb - Create a New Process File .....	277
14.2.11	ftberr - Write FTB File Errors.....	277
14.2.12	ftbexp - Expand FTB Memory .....	279
14.2.13	ftfbtb - Initialize FTB Arrays.....	280
14.2.14	ftbget - Read a Set from a Process File.....	280
14.2.15	ftbin - Buffered Input from Disk.....	280
14.2.16	ftbint - Initialize FTB Package .....	280
14.2.17	ftblct - Find Contiguous Space on a Unit .....	281
14.2.18	ftbmem - Measure and Modify FTB Memory Size .....	281
14.2.19	ftbmov - Move a Block of Words in Memory .....	282
14.2.20	ftbnid - Get Next Available FTB File Identifier .....	282
14.2.21	ftbopn - Open a Process File.....	282
14.2.22	ftbout - Buffered Output to a Disk .....	283
14.2.23	ftbpr1 - Process One Process File .....	283
14.2.24	ftbpr2 - Process Two Process Files .....	283
14.2.25	ftbpr3 - Process Three Process Files .....	283
14.2.26	ftbpr4 - Process Four Process Files .....	284
14.2.27	ftbput - Write a Set to a Process File .....	284
14.2.28	ftbrdc - Release Excess Space .....	284
14.2.29	ftbrsv - Create a New Reserve File .....	284
14.2.30	ftbsft - Shift and Resize a Reserve File .....	285
14.2.31	ftbslk - Shift Links Toward Preferred End.....	285
14.2.32	ftbtnc - Truncate and Close a Process File .....	285
14.2.33	idfind - Find Index for a File .....	286
14.2.34	inxget - Get Index for a File .....	286
14.2.35	isfdes - Is Process File Described .....	286
14.2.36	isfopn - Is Process File Open .....	286

14.2.37	issfrg - Is Space on Unit Fragmented.....	286
14.2.38	lavail - Space Available in This Unit .....	286
14.2.39	lcntgs - Largest Contiguous Space Left on Unit after a File is Loaded.....	287
14.2.40	lcontg - Largest Contiguous Space on Unit.....	287
14.2.41	lifopn - Space Required to Open This Process File.....	287
14.2.42	mxsets - Number of Sets of Given Size on This Unit.....	287
14.2.43	nfsets - Number of Sets in a Process File .....	287
14.2.44	nfsiz - Process File Setsize or Reserve File Size .....	288
14.2.45	nfunit - Unit Number for This File .....	288
14.3	References .....	288
15	UNIT TEST PACKAGE.....	289
15.1	Overall View Of The Unit Test Packages.....	289
15.2	Interphase Drag Unit Test Package .....	292
15.3	Interphase Heat Transfer Unit Test Package .....	292
15.4	Wall Heat Transfer Unit Test Package.....	292
15.5	Surface Plotting Package.....	292
16	TO DO LIST.....	293
17	INDEX OF VARIABLES IN COMDECKS .....	295
18	INDEX OF CARD NUMBERS .....	347
19	INDEX OF COMMON BLOCKS .....	349
20	INDEX OF SUBROUTINES .....	351
21	INDEX OF VARIABLES.....	355



## **FIGURES**

	Page
41-1 Arrangement of Words for a Big Endian Computer.....	62
41-2 Arrangement of Words for a Little Endian Computer .....	63
85-1 Flowchart for the timstop Subroutine .....	79
112-1 Storage Layout for the xl1 Array (64-bit Words) in the INP Package .....	229
113-1 Storage Layout for the xl1 Array (64-bit Words) in the INP Package .....	240
121-1 Pressure-Temperature Diagram Showing Regions and Critical and Triple Points.....	247
151-1 Flow Diagram of the phantvd Unit Test Package for RELAP5 .....	290
151-2 Surface Plot from the Unit Test Package for phantv .....	291



## TABLES

Page	
3.1-1	Connection between Card Numbers, Subroutines, and Common Blocks ..... 23
4.1-1	Association of a Dynamic Common Block and its Number in the filndx Array ..... 59
4.1-2	Voldat Comdeck Variable Arrangement ..... 65
4.1-3	Jundat Comdeck Variable Arrangement ..... 66
9.2-1	Format Codes Returned from Cvic Subroutine ..... 86
9.2-2	Example of Hollerith Data in the bin and ic Arrays ..... 86
10.1-1	Component Names in Comdeck cmpalf.hh ..... 89
10.2-1	Accumulator Component Variable Names in Comdeck cmpdac.hh ..... 90
10.3-1	Component Variable Names in Comdeck cmpdat.hh ..... 92
10.4-1	Valve Component Variable Names in Comdeck cmpdtv.hh ..... 95
10.5-1	Control System Operator Names in Comdeck cnvtpa.hh ..... 97
10.6-1	Control Block Variable Names in Comdeck comctl.hh ..... 98
10.8-1	Constants Variable Names in Comdeck cons.hh ..... 100
10.9-1	Main Control Block Names in Comdeck contrl.hh ..... 101
10.10-1	Control Variable Names in Comdeck convarc.hh ..... 106
10.11-1	ECC Mixer Component Variable Name in Comdeck eccmxc.hh ..... 109
10.12-1	/fast/ Storage Pool Variable Names in Comdeck fast.hh ..... 109
10.13-1	Reflood Variable Names in Comdeck flood.hh ..... 110
10.14-1	FTB Package Variable Names in Comdeck ftbcom.hh ..... 110
10.15-1	Fuel Pin Gap Variable Names in Comdeck gapvar.hh ..... 112
10.16-1	Problem Title Names in Comdeck genrl.hh ..... 112
10.17-1	General Table Variable Names in Comdeck gentblc.hh ..... 112
10.18-1	Heat Transfer Correlation Variable Names in Comdeck htrcom.hh ..... 113
10.19-1	Reflood Variable Names in Comdeck htrflb.hh ..... 116
10.20-1	Reflood Scratch Variable Names in Comdeck htscr1.hh ..... 118
10.21-1	Reflood Scratch Variable Names in Comdeck htscr2.hh ..... 118
10.22-1	Heat Transfer Scratch Variable Names in Comdeck htscr.hh ..... 119
10.23-1	Heat Structure Variable Names in Comdeck htsrcm.hh ..... 120
10.24-1	Interactive Variable Names in Comdeck intrac.hh ..... 126
10.25-1	Inverted Heat Structure Table Variable Names in Comdeck invhtb.hh ..... 127
10.26-1	Inverted Junction Table Variable Names in Comdeck invtbl.hh ..... 128
10.27-1	Junction Variable Names in Comdeck jundat.hh ..... 128
10.28-1	Loop Control Array Variable Names in Comdeck lcntrl.hh ..... 136
10.29-1	Level Model Variable Names in Comdeck levtbl.hh ..... 136
10.30-1	Hydrodynamic Systems Control Variable Names in Comdeck lpdat.hh ..... 137
10.31-1	List Vector Pointer Variable Names in Comdeck lvectr.hh ..... 139
10.32-1	Level Model Variable Names in Comdeck lvl1.hh ..... 140

10.38-1	Unit Test Variable Names in Comdeck makmap.hh.....	142
10.39-1	Primary and Secondary Memory Allocation Variable Names in Comdeck maxmem.hh ..	143
10.40-1	Minor Edit Variable Names in Comdeck miedtc.hh.....	143
10.41-1	Thermal Property Variable Names in Comdeck mtbls.hh .....	144
10.42-1	Thermal Property Variable Names in Comdeck mxnfcd.hh.....	145
10.43-1	NPA Link Names in Comdeck npacom.hh .....	145
10.44-1	PC Plot Variable Names in Comdeck plotpc.hh .....	146
10.45-1	Pressurizer Component Variable Names in Comdeck przdat.hh.....	146
10.46-1	Pump Component Variable Names in Comdeck pumpblk.hh .....	147
10.47-1	Radiation Heat Transfer Variable Names in Comdeck radhtc.hh.....	149
10.48-1	Component Variable Names in Comdeck rcompc.hh.....	151
10.49-1	Reflood Heat Transfer Variable Names in Comdeck rflhtc.hh.....	151
10.50-1	Reactor Kinetics Variable Names in Comdeck rkinc.hh .....	152
10.51-1	ANS79 Neutron Absorption Variable Names in Comdeck rknatb.hh .....	161
10.52-1	Material Data for Heat Structures Variable Names in Comdeck rmadac.hh .....	161
10.53-1	Reactor Kinetics Data Variable Names in Comdeck rrkinc.hh .....	162
10.54-1	Stateq Variable Names in Comdeck scrq.hh.....	163
10.55-1	Scratch Variable Names in Comdeck scrtch.hh.....	165
10.56-1	Accumulator Component Variable Names in Comdeck separ.hh .....	183
10.57-1	Steady-State Component Variable Names in Comdeck sscntr.hh .....	184
10.58-1	Steady-State Self Initialization Variable Names in Comdeck ssiblk.hh .....	185
10.59-1	Advancement Statistics Variable Names in Comdeck statc.hh .....	186
10.60-1	State Property Variable Names in Comdeck statec.hh .....	188
10.61-1	Fluid Type Variable Names in Comdeck stcblk.hh .....	189
10.62-1	Steam Table Parameters in Comdeck stcom.hh .....	189
10.63-1	System Variable Names in Comdeck sysdate.hh.....	190
10.64-1	Time Dependent Volume-Junction Pointers in Comdeck tdpptr.hh .....	191
10.65-1	Temporary Common Block for the Steam Table Parameters in Comdeck tmsrcm.hh.....	191
10.66-1	Temporary Common Block for the Steam Table Parameters in Comdeck trnhlp.hh .....	192
10.67-1	Trip Components in Comdeck trpbblk.hh.....	193
10.68-1	Mass Error Time Step Control Limits in Comdeck tsctlc.hh.....	194
10.69-1	Trip Components in Comdeck tstpct.hh .....	195
10.70-1	Turbine Component Variable Names in Comdeck turbin.hh .....	196
10.71-1	File Name and Unit Numbers in Comdecks ufilef.hh and ufiles.hh .....	197
10.72-1	User Variable Names in Comdeck usrvar.hh.....	198
10.73-1	User Variable Names in Comdeck voldat.hh.....	198
10.74-1	Plot Variable Names in Comdecks vreqs.hh and vreqd.hh.....	212
11.1-1	Hierarchy of the File, Set, and Case Levels in INP Input Data File .....	223
11.2-1	File Names and Unit Numbers.....	228
11.3-1	INP Package Error Messages.....	238

11.3-2	Table Word Structure.....	240
12.0-1	Thermodynamic Quantities.....	243
12.1-1	Data Limits for Water .....	245
12.1-2	Record 1 of the Steam Table Binary File, tpfh2o .....	246
12.1-3	Record 2 of the Steam Table Binary File, tpfh2o .....	248
12.1-4	Saturation Quantities as a Function of Temperature in Table 3.....	249
12.1-5	Saturation Quantities as a Function of Pressure in Table 4 .....	249
12.1-6	Single Phase Quantities as a Function of Temperature and Pressure in Table 5 .....	250
12.2-1	Thermodynamic Properties Returned in the s Array .....	251
12.2-2	Parameters for Saturation Pressure vs. Temperature .....	253
12.2-3	Pressure Limits for Saturation Temperature vs. Pressure Formulas .....	255
12.2-4	Parameters for Saturation Temperature vs. Pressure Formula for Low Pressure Range .....	255
12.2-5	Parameters for Saturation Temperature vs. Pressure Formula for High Pressure Range.....	256
12.2-6	Parameters for Saturation Temperature vs. Pressure Formula for Below Triple Point Pressure .....	256
13.1-1	Rstplt Record Structure.....	262
13.2-1	Plotinf Record Structure .....	263
13.3-1	Plotalf Record Structure.....	263
13.4-1	Plotnum Record Structure.....	264
13.5-1	Plotrec Record Structure .....	264
13.6-1	Restart Header Record Structure .....	265
13.6-2	Restart Title Record Structure.....	265
13.6-3	Restart Common Block Record Structure.....	266
14.1-1	Four-Word File Description Slot for a Link.....	271
14.1-2	Four-Word File Description Slot for a Reserve File .....	271
14.2-1	Output from dmplst for the edhtrk.i Input Deck .....	274
14.2-2	Output From fildmp for the edhtrk.i Input Deck .....	276
14.2-3	Ftb Errors .....	278



## **1 INTRODUCTION**

Volume 8 contains information for programmers of RELAP5/MOD3.3. This document contains detailed information of the program flow, internal databases, and debugging facilities in RELAP5. RELAP5 was developed for the U.S. Nuclear Regulatory Commission (NRC) and is in use throughout the world for nuclear reactor safety evaluations.

The code is capable of modeling a wide range of systems that include single pipes to small-scale experimental facilities to full nuclear reactor plants. RELAP5 has models for thermal hydraulics including noncondensable gas transport, control systems, heat transfer to and from solid surfaces, and nuclear reactor kinetics. The models are built up from volumes connected together with junctions. The volumes can have associated heat structures attached to them. Junctions can include valves of various types.

This manual discusses the programming techniques utilized in the RELAP5/MOD3.3 computer code. In particular, the dynamic common blocks and their associated manipulation subroutines, the input processing subroutines, the property tables and their interpolation subroutines, the restart and plot file structures are discussed.



## **2 CODING CONVENTIONS**

### **2.1 Programming Language**

The programming language used in RELAP5 is primarily FORTRAN 77 with some extensions. In particular, the MIL-STD 1753 set of extensions are used for bit manipulation within packed words. Variable names can be up to 14 characters long. C-language subroutines are used on some of the CPU platforms for trapping floating point underflows and overflows. On the HP platform, a C-language subroutine is used to trap the interrupt signal from a control-C command so the code can write out a restart file at the time the interrupt signal is detected. One C-language subroutine is used to link RELAP5 to SNAP, the graphical user input program.

### **2.2 FORTRAN 77 and Extensions**

The DOD military standard set of extensions, MIL-STD 1753, are used to extend the capability of FORTRAN 77 to manipulate, test, and set individual bits in the computer words. These extensions were used extensively for the packed words, but now that most of the packed words have been removed from RELAP5 they are rarely used inside the thermohydraulics part of code. The MIL-STD-1753 subroutines are still used, however, in the dynamic storage, input card processing, and sparse matrix manipulation subroutines.

### **2.3 File Naming Conventions**

The FORTRAN 77 source files have either a “.ff” or a “.hh” suffix. The “.ff” files are source code for programs, subroutines, and functions. All the “.ff” files have a “\*deck name” comment card as the first line in the file where name is the name of the file without the “.ff” suffix. The “.hh” files are source code files for the include files like common blocks, specification statements, data statements, statement functions, etc. All the “.hh” files have a “\*comdeck name” comment card as the first line where name is again the name of the file without the “.hh” suffix. This process of preceding each file with its name is useful later when the usplit program is used to split apart the individual files from a concatenated file containing all the individual files. These concatenated files are distributed when a new version of the code is released.

The “.ff” files are sequentially run through the two preprocessors, select and cnv32, to yield a file with a “.f” suffix. The “.hh” files are also run through these same two preprocessors to yield a file with a “.h” suffix. In earlier versions of the RELAP5 code prior to Version 3.3Beta, the “.ff” files used a “.F” suffix and the “.hh” files used a “.H” suffix. The “.F” suffix was changed to the “.ff” suffix and the “.H” suffix was changed to the “.hh” suffix when the code was moved to operating systems in which the upper and lower case file names are identical. Windows is one such example.

All the executable files have a “.x” suffix and the scripts that are used in building the code start with the letter “d”, e.g., dinstls, denvrl, drelap, dstgxxx, etc. Recently, the scripts used to build the code have been converted to a new system in which uses a group of makefiles. The initial build is started by typing “configure” and after configure script is finished, by typing “make”.

## 2.4 Preprocessors

Two preprocessors are used sequentially to preprocess the source code before it is sent to the FORTRAN 77 compiler. The first preprocessor, select, is used to mimic the processing that was originally done by the CDC Update program. It allows the code to have only one source file that can be filtered so that it can be configured to run on different CPU platforms. The second preprocessor, cnv32, is used to convert constants in the code to double precision, e.g., 1.0e1 is converted to 1.0d0, and to doubly subscript integer variables that are equivalenced to the double precision fa array, which is in the /fast/ common block. For example, integer volno(1) in the voldat.hh comdeck is changed to integer volno(2,1) and the equivalence statement, equivalence (volno(1),ia(4)) is changed to equivalence (volno(1,1), ia(1.4)) on a little endian computer platform or to equivalence (volno(2,1), ia(2,4)) on a big endian computer platform.

### 2.4.1 select Precompiler

The select preprocessor works by commenting out various parts of the code. The parts that are commented out are under the control of if-define statements: "if def,parameter" or "if -def,parameter". These if-define statements are embedded in the source file itself. The select program recognizes five specification and control statements in the source code: "\$define parameter", "\$if def,parameter", "\$if -def,parameter", "\$endif", and "\*call name". The "\$define parameter" statement sets the parameter names that are to be defined for this run through the select preprocessor. A parameter name is limited to 8 characters. The "\$if def,parameter" and "\$endif" pair of statements delimit a block of coding that is not commented out if parameter is defined in a define statement. Otherwise, the block of coding is commented out. The inverse of this control block, i.e., "\$if -def,parameter" and "\$endif" delimits a block of coding that is not commented out if parameter is not defined and is commented out if parameter is defined. The select preprocessor replaces the "\$" in these preprocessor control statements by an "\*", which changes the card to a comment card that is recognized by the FORTRAN 77 compiler. The commenting out is done by replacing the character in column 1 by a "\*", not by inserting a "\*" and moving the rest of the line over one character. Thus, comment cards that have a "c" in column 1 now have an "\*" in column 1, and numeric labels that start in column 1 now have the first digit replaced by a "\*".

Another variation on the "\$if def,parameter" statement is also recognized. This is the "\$if def,parameter,number" where number indicates the number of following statements that are under the control of the if-define statement. This form does not have the "\$endif" at the end of the block. In fact, if a "\$endif" statement is included, select flags it as an error. Originally, RELAP5 had both types of these if-define statements. All the alternate statements, i.e., the ones that do not require the "\$endif" statement, have been converted to the "\$if def-\$endif" type because this type results in fewer programmer errors when a change is made to the coding. It was too easy to add or delete a statement when using the alternate form and forget to change the number on the "\$if def,parameter,number" statement. Sometimes, feedback on the error was given to the programmer by select or later by the Fortran compiler. But, more often than not, no feedback was given and the error remained in the code until it was discovered by some other method. The basic problem with the alternate form was that making a change to the code in one location, like adding or deleting a statement, required the programmer to make a change in a secondary, non-local location. The block "\$if def-\$endif" form does not require the programmer to make a secondary, non-local change to the

source code. This is just one of the many changes that have been made to the RELAP5 source code to make its maintenance more robust.

The "\*call name" preprocessor statement is converted to an include statement for the corresponding comdeck. For example, "\*call fast" is converted to "include fast.h" where there are six spaces before the include so that the statement begins in column 7. The fast.h file and all the other ".h" files are constructed by running the fast.hh file through the same two preprocessors, select and cnv32.

## **2.4.2 cnv32 Preprocessor**

The cnv32 preprocessor adds a "d0" to all the numeric constants in the coding. It also converts any constants written with the "e±#" structure to use "d±#", where "#" is any positive or negative integer. This conversion converts all the constants in the code to double precision. The programmer should not code a constant with the "d##" notation. Use the "e±#" notation. If the "d##" notation is used, the result, after running it through cnv32, will be "d±#d0" and the FORTRAN compiler will flag the statement as an error.

The other task for the cnv32 preprocessor is to add the "2," or "1," in front of the subscript on integer variables that are equivalenced to fa in the /fast/ common block. The "2," is used on big Endian computers, and "1," is used on little Endian computers. This real\*8 variable, fa, is used for dynamic storage in RELAP5. Integers, logicals, and double precision variables are stored in this real\*8 array by using equivalence statements. The fa array is dimensioned for 6,600,000 if the line "define smallfa" is in the define file and 10,000,000 if the line "define bigfa" is in the define file. Since integers and logicals are only 4 bytes as opposed to 8 bytes for the double precision fa variable, there is room for two of them in each 8-byte fa word. Thus, all integers and logicals are dimensioned to be (2,1) and are then equivalenced to ia(1,n) and to fa(1) with the statement "equivalence (fa(1),ia(1,1))". There was a choice as to whether to store the integer in the first part, ia(1,1), of the second part, ia(2,1), of the fa word. On big Endian computers like the IBM-360, which is where RELAP5 was developed, the choice was made to use the second part because it would not disturb the sign and exponent part of the real\*8 word, which is in the ia(1,1) part of the word. The ia(1,1) part of the word was set to zero. The sign is the first bit and is a single bit. The exponent is 11 bits long in a double precision (real\*8) word. This decision then required the programmer to either insert a "2," in front of every subscript for every integer or logical variable that is equivalenced to the fa array or for a program to be written that would do it automatically. The cnv32 program is that program. The list of global integer and logical variables that have to be converted is stored in a file called mlist. In addition to mlist, there are cases where local integer and logical variables have to be converted and in order for cnv32 to know about them, two additional control statements were used, "\*in32 variablename" and "\*in32end". For example, if iprop is a local integer variable that is equivalenced to the real\*8 word prop, which in turn is equivalenced to a location in the fa array, then the statement containing "in32 iprop" is added to the subroutine near the front. There could be multiple "in32 variable" statements, so the last one is followed by the "in32end" statement to signify to cnv32.x that there are no more. These local variables are added to set from mlist for this subroutine only and cnv32.x proceeds to preprocess the deck.

On little Endian computers, the sign and exponent bits of a real\*8 word are stored in the ia(2,1) part of the word, so the cnv32.f coding was modified to insert a "1," in front of every subscript for every integer or logical variable that is equivalenced to the fa array. The specification statements were not changed since the integers and logicals still have to be doubly dimensioned. Also the equivalence statements were not changed since the first part of the doubly dimensioned integer or logical is equivalenced to the fa word. The define file contains the line 'define loadhi' on little Endian computers and this parameter is used in "\$if def,parameter" to determine whether "2," or "1," is used in the cnv32 program itself. This means that cnv32.f has to be run through select.x before it can be compiled. Before this change to correctly handle little Endian computers, cnv32.f could be compiled directly.

This difference in how floating point real\*8 words are arranged in memory was discovered when porting the code to an Intel Pentium processor using the Compact Visual Fortran compiler (CVF 6.0), it was discovered that some of the integers that were equivalenced to real\*8 word were being changed by the RELAP5 program. In particular, the time variable used in control systems and trips was being modified after a restart. The time variable lives in the /contrl/ common block. In the control system or trip coding, the time variable is looked up when it is needed by using the offset to where the variable is located relative to the start of the fa array, fa(1). This procedure is a unique feature of RELAP5 and results in a faster running code. It turns out that the /contrl/ common block was being stored at a much lower memory location than was the /fast/ common block. Thus, the offset to the time variable was a large negative integer number. A negative integer has the feature of having almost all of its bits set to ones, starting with the sign bit, because it is stored in either a two's complement or one's complement form. When this negative number is overlaid on the real\*8 fa word on a little Endian computer, it looks like a NaN (Not a Number) because of all the one bits in the top 12 bits of the word. A real\*8 NaN has the sign bit, all eleven of the exponent bits, and one or more of the mantissa bits set to one. There are two types of NaNs, signaling and quiet NaNs. A signaling NaN has a zero bit in position 19 of the 32-bit part of the word that contains the sign and exponent bits. Bit positions are numbered from the right-most part of the 32-bit word at bit 0 and go up to bit 31 at the left-most part of the 32-bit word. A quiet NaN has a one bit in position 19. This is the first bit beyond the last exponent bit, which is bit 20. Any floating point operation, including a load when using CVF 6.0, that involves a signaling NaN generates an exception. If the exception is masked, bit 19 is converted to a one to convert the number to a quiet NaN so that the exception is not raised again when the number is accessed. This is where the problem arose with overlaying the top part of a real\*8 floating point word with a negative integer. The equivalenced real\*8 number shows up as a NaN, and if the bit pattern is such that it looks like a signaling NaN, it will get converted to a quiet NaN. Consequently, the top part of the word, which is where the integer lives, has been changed. This is what was happening to the time variable in the control system and trips in the restart runs on the Intel processor. The problem was with the Intel PU and not the compiler. It also occurs in the AMD CPU, which emulates the Intel CPU. The memory location that contained the offset to the time variable was stored in the /fast/ common block. Parts of the /fast/ common block get moved in memory during the restart process. It was not happening for all problems because it depended on having a timely offset large enough to have bit 19 set to zero. Subsequently, the order of loading the common blocks was changed in the blkdat subroutine so that the /fast/ common block is loaded first and then the /contrl/ common block is loaded. This worked on some loaders, but created problems on others. Hence, this was when loadhi parameter was inverted and used in cnv32 and in other parts of the RELAP5 coding, e.g. sparse matrix solvers.

## 2.5 Job Control Language (cshell)

All the scripts in the installation use csh for its job control language. All the scripts have the following line as their first line to make sure the c-shell is used for processing the script.

```
#!/bin/csh
```

On Windows computers, which do not have csh available, the distribution includes tcsh, so the c-shell can be used for the installation on all computers. The tcsh stands for total csh and is freely available on almost all computers. By using tcsh, the maintenance of the RELAP5 installation scripts will be easier in that only one set of scripts will need to be maintained.

The later versions of the code require the user to install the CYGWIN package in order to install the code. This package allows the use of GNU routines to emulate UNIX routines on Windows. It simplifies the maintenance of the configure installation script.

## 2.6 Static and Dynamic Storage

RELAP5 uses both static and dynamic data storage. The static storage uses normal Fortran named common blocks and the dynamic storage is done using a set of FTB subroutines to manipulate the storage in the large fa array in the normal common block named /fast/. Detailed descriptions of the static and dynamic common blocks are given in Section 10. The the FTB dynamic storage manipulation package is described in Section 14.

## 2.7 Input Deck Structure

The input decks have an optional title card, individually numbered cards, and an end card. The default name for this file is indta. The title card has an equal sign in column 1, and the end card has a period in column 1. The individually numbered cards start with a card number that signifies the type of the card. Comment cards have an asterisk in column 1. Comments can be added to the individually numbered cards by adding an asterisk after the last number on the card, followed by the comment text. Duplicate cards, i.e., cards with identical card numbers, can also be a part of the input deck. In that case, the last duplicate card is the one that is used. The duplicate cards do not have to be physically adjacent in the deck, but it is strongly recommended that they are in order to eliminate problems with the user making a change in the first card and expecting to see a difference in the output, but in fact no changes are seen because the wrong card was changed. The input cards are read by subroutines in the CVI package, which is described in detail in Section 9.

## 2.8 Restart-Plot File Structure

The restart-plot file is really two binary files merged into one. The default name for this file is rstplt. This approach was taken to reduce the number of magnetic tapes involved when the code was run on a large mainframe computer like the IBM-360. This restart-plot binary file contains interspersed blocks of restart file records and plot file records. Each block of records is delimited by a special record that tells the

## Restart-Plot File Structure

kind and length of the record or records that follow. The restart-plot file is initialized in the rrstd subroutine. The wrplid subroutine writes the labels for the plots at the start of the run or at the start of a restart run if 2080xxxx cards are in the restart input deck. The pltwrt subroutine writes the numeric value of the plot variables each time a plot edit is requested. The rrestf subroutine reads the restart cards in the input deck and processes them. The rstrec subroutine writes the restart records to the file each time a restart edit is requested. Details on the restart-plot file structure are given in Section 13.

### 3 PROGRAM STRUCTURE

The RELAP5 program, relap5.x, operates in three stages. During the first stage, the input deck is read into memory, and the r-level subroutines are called to process the input deck. Any errors in the input deck are noted in the output file. In order to minimize the number times the input deck has to be processed to remove all the errors, any input errors that are found are temporarily patched, i.e., benign numbers are used to replace the bad input numbers, and the input processing continues. This procedure eliminates the tedious process of finding the first error, fixing it, re-running the problem, finding the second error, and repeating the process until all the errors are found and fixed.

During the second stage, the i-level subroutines are called and the links between the various parts of dynamic storage area are calculated and set, e.g., the location in the fa array of the pressure for a volume that is specified in a trip is determined, and the offset is stored in the trip dynamic storage area. This preprocessing speeds up the code because this offset does not have to be recomputed each time the trip logic is processed during the transient.

During the third stage, the transient is run. During the transient, major and minor edits are written to the output file, and plot records and restart records are written to the restart-plot file. The frequency of these writes is set by values on the time step input cards.

#### 3.1 Input Processing Subroutines

The input processing is started with the subroutine `inptd` which is called from `relap5.x`. `Inptd` calls three subroutines.

1. `gninit` does the general initialization
2. `rcards` reads in the input cards
3. `rnewp` processes the input data for a new problem

The `rnewp` subroutine calls the r-level subroutines (all these subroutines start with the letter r) to process the input cards.

1. `rrstd` creates the restart file
2. `rrestf` reads in an existing restart file
3. `rchng` processes the card one options
4. `rtsc` processes the time step control cards
5. `rssi` processes the optional self-initialization control and identification cards

## Input Processing Subroutines

6. rmiedt processes the minor edit cards
7. rintry processes the interactive input and output variable cards
8. rtrip processes the trip cards
9. rnoncn processes input data for constants needed in noncondensable calculations
10. rmflds processes optional hydrodynamic system cards
11. rcompn processes the hydrodynamic components
12. rhtcmp processes the heat structure components
13. rradht processes the radiation heat transfer components
14. rmadat processes thermal property composition cards
15. rrkin processes the reactor kinetics input
16. rgntbl processes the general tables
17. rconvr processes the control system components
18. rusrvr processes the user-supplied variables to be added to the restart-plot file for plotting and/or interactive mode

After the r-level subroutines are called by rnewp to process the input cards, rnewp calls the i-level subroutines (almost all these subroutines start with the letter i) to check the input and link up the various components.

1. itstck checks the control system variable reference for time step control
2. icmpn1 cross checks the hydrodynamic component input and performs the first pass of component initialization
3. itrip checks and processes the trip data and sets the initial values of the trips
4. igtntbl cross checks the general tables and establishes links for the general tables and trips
5. icompn cross checks the component input and completes the component initialization
6. ihtcmp checks the heat structures and sets up the referrals
7. iradht checks the radiation heat transfer input

8. irflht checks the reflood input and sets up the reflood indices
9. invhts prepares the inverted heat structure table that links heat structures to volumes
10. irkin initializes the reactor kinetics
11. iconvr initializes the control variables
12. imiedt sets up the internal storage and links for the minor edit variables
13. issi checks the optional self initialization input
14. iusrvr checks the user-supplied restart-plot file variable requests for validity
15. wrplid writes the plot variable labels to the restart-plot file

After the input processing in rnewp has completed successfully, the transient control subroutine, trnctl, is called. The trnctl subroutine calls three subroutines.

1. trnset sets up the transient calculation
2. tran controls the subroutines that perform the transient
3. trnfin cleans up after the transient is run

The details of the input processing will be discussed in the following sections. The set of subroutines that process the input cards is called the CVI Package and it is discussed in detail in Section 9.

### **3.1.1 CVI Package Subroutines**

The CVI package contains two subroutines, cvic and cvirc. The cvirc subroutine reads one card or record into an internal array and then calls the cvic subroutine to process the data in the internal array. While cards are not used anymore for input, it is convenient to refer to each record as a card. The card contains fields of data in integer, floating point, or character formats. The principal advantage of the cvic subroutine over standard FORTRAN 77 formatted input is that the input data need not be entered in predetermined columns. Thus, the input data forms can be simpler and the data entry easier. The cvic subroutine can easily handle input cards with a variable number of fields on the card.

The CVI package also allows replacement cards in the input deck. Replacement cards are cards that occur later in the deck that have the same ID number as an earlier card. When such a card is encountered, it replaces the earlier card entirely. While some analysts follow the practice of putting all change cards to a base deck at the end of the base deck, this practice can sometimes come back to haunt them later in that if they forget about the change cards they might change the first card and expect to see a difference in the output, but none appears because the later change card rules.

### 3.1.2 R-level Subroutines

The r-level and i-level subroutines in RELAP5 will be briefly discussed in the order in which they are encountered in the rnewp, which is the main driver for calling these subroutines. The rnewp subroutine processes the input data for a new problem. It calls the r-level subroutines: rrstd or rrestf, rchng, rtsc, rmiedt, rintrv, rtrip, rmoncn, rmflds, rcompn, rhtcmp, rradht, rmadat, rrkin, rgntbl, rconvr, and rusrvr. It then calls the i-level subroutines: itstck, icmpn1, itrip, igntbl, icompn, ihtcmp, iradht, irflht, invhts, irkin, iconvr, imiedt, issi, iusrvr, and finally wrplid. The r-level subroutines are concerned with reading in the information from the input cards and are described in this section. The i-level subroutines are described in the next section.

An abbreviated flowchart of the rnewp subroutine is shown in below. The variable, newrst, is true if this is a new problem and false if it is a restart problem. The if statements that check if filid(n) is not equal to zero are to check to make sure the dynamic common file does exist, e.g., filid(8) is not zero if there are heat transfer slabs in the problem. If there are heat slabs in the problem, then the rnewp has to call the ihtcmp and irflht subroutines. If there are radiation surfaces in the problem, then filid(38) is not zero and there is a need to call the iradht subroutine.

```

subroutine rnewp
  newrst = problemtype05. eq. 1
a<<<<<<< if (newrst) then
  a          call rrstd
a<<<<<<< else
  a          call rrestf (nogof)
  a          newrst = .false.
a>>>>>>> endif
  call rchng
  call rtsc
  call rssi
  call rmiedt
  call rintrv
  call rtrip
  if (newrst) call rmoncn
  call rmflds
  call rcompn
  call rhtcmp
  call rradht
  call rmadat
  call rrkin
  call rgntbl
  call rconvr
  call rusrvr
  if (filid(2) .ne. 0) call itstck
  if (filid(3) .ne. 0) call icmpn1
  if (filid(18) .ne. 0) call itrip
  if (filid(11) .ne. 0) call igntbl
  if (filid(3) .ne. 0) call icompn
b<<<<<<< if (filid(8) .ne. 0) then
  b          call ihtcmp
  b          if (filid(38) .ne. 0) call iradht
  b          call irflht
b>>>>>>> endif
  call invhts
  if (filid(21) .ne. 0) call irkin
  if (filid(27) .ne. 0) call iconvr
  if (filid(12) .ne. 0) call imiedt

```

```

if (filid(37) .ne. 0) call issi
if (filid(33) .ne. 0) call iusrvr
if ((isrestartenabled0      .and.
&      (ishydrochng0          .or.
&      isheatcondchng1        .or.
&      isrkchng2              .or.
&      ispltrcrdchng3         .or.
&      isradiationchng4       .or.
&      isfissionprdctchng5   .or.
&      iscntrlsyschng6))     .or.
&      interactiveflag5       .or.
&      lreset) call wrplid
      return
    end

```

### **3.1.2.1 rrstd**

The rrstd subroutine processes the cards for a new restart-plot file. It opens the restart file as a new file. The name of the restart file is either the default value of “rstplt” or the value entered on the command line after the “-r” or “-R” option. The rrstd subroutine writes the first two records on the restart file. The first record consists of two integer words, 10 and 8. The first word, 10, is the number of words in the second record that will be written to the file and the second word, 8, is the length of these words in bytes. The second record consists of three 24-byte chunks followed by one real\*8 word. Each chunk consists of three 8-byte words or 24 characters. If the chunk is not 24 characters long, it is padded on the right with blanks. The first chunk is the program version, e.g., “RELAP5/3.3al”. The second chunk is the type of the file and contains the character string “restart-plot file”. The last chunk is the date and time of the run, e.g., “29-JUN-01 09:10:35”. The last word consists of the variable problemopt611 This variable contains information of the type of the run, e.g., steady-state (=1) or transient (=2). The integer variable problemopt611 is equivalenced to a real\*8 word.

### **3.1.2.2 rrestf**

The rrestf subroutine processes the 103 and optional 104 restart cards for an existing restart-plot file. The desired restart record is entered in the input deck on card 103, and the restart-plot file name is on the optional 104 card. The rrestf subroutine opens the restart-plot file as an old file if the command line option “-r” is used and as an unknown file if the “-R” is used ahead of the restart-plot file name. The rrestf subroutine starts reading the file until it finds the desired restart record number. If it finds the restart record number, the restart records for this restart are read into the internal database and the transient is started. Any new restart-plot records are then written after the desired restart record. Thus, any restart-plot records from the earlier run that were written after the desired restart record are overwritten.

Some convenience features associated with the restart-plot file have been added to the code. A restart record of -1 on the 103 card signifies that the last restart-plot file record is to be used for the restart. All the restart records that are on the restart-plot file are written at the end of the output file in a format that is suitable for cutting and pasting into a restart input deck. The 103 cards are in the proper format, have the time of the restart as a comment on the card, and the 103 on each card is preceded by an asterisk so it looks like a comment card to the input deck processor. That way, the user only needs to remove the asterisk on

the card that has the desired restart number on it, and the problem is off and running again. This eliminates transcription errors. In the past, users have had to perform an intermediate steady-state run in order to eliminate the front end of the restart-plot file and set the time to back to zero. This procedure is customarily done in order to eliminate the “null” transient run that is commonly used to initialize the problem prior to running a transient. This process is simplified by use of the word “reset” instead of the “restart” on the 100 card. When reset is used, the front end of the restart-plot file is eliminated. The time is not reset to zero, but by using the 200 card, the time can easily be reset to zero. As a convenience to the program debugger in which multiple runs are made using the same restart file name, the “-R restartfilename” and “-O outputfilename” options have been added. Use of these options will allow overwriting of the restart file and output files, so that they do not have to be removed before each debug run.

### **3.1.2.3 rchng**

The rchng subroutine processes the card 1 developmental model options. It sets the appropriate logical chngno variables to true. For example, if option 6 is entered on card 1, then chngno(6) is set to true. Initially, all the 90 elements in the chngno array, with the one exception of option 60, are set to false. The user can get a short description of all the currently available options in any version of the code by entering 0 for the option number. To turn off an option on a restart, the user enters the option number again with a negative sign in front of it.

### **3.1.2.4 rtsc**

The rtsc subroutine processes the time step control cards, 200-299. The optional 200 card contains the initial time and an optional control variable number for controlling the time step size. The 201-299 cards contain the end time, minimum and maximum time step sizes, control options in a packed word, and output frequencies as a multiple of the maximum time step size for the minor edits (and plots), major edits, and restarts. In the original RELAP5 code, the time steps sizes were either halved or doubled as appropriate. This resulted in the code stopping exactly at the end time, since it was usually a multiple of the maximum time step size. In versions of the code prior to MOD 3.3Beta, the doubling of the time step size was replaced by a 10% increase. The halving was still kept as a way of getting out of trouble quickly when things started happening too fast in the transient. The end time was still sacred in that the two time steps before the end time was reached were changed so that the end time was hit exactly. This resulted in differences in what should have been identical transients in which there was an additional time step card containing an intermediate end time and one in which there was no intermediate card. In order to eliminate this behavior, MOD 3.3Beta was changed so that the time steps before the end time are not modified. Consequently, the actual time the transient ends is the first time step beyond the end time. This eliminated the differences between the restarted run and un-restarted run. In order to accommodate the user that still wants to end the transient exactly on the end time, the code now allows the user to put in a negative sign in front of the end time. This signifies to the code that the user wants to hit the end time exactly, and the two smaller time steps will be taken before the end time in order to hit the end time exactly.

The time step cards are stored in the dynamic common block 2. It has an id of filid(2), starts at filndx(2), and has a length of filsiz(2). As one might expect, it is a relatively small dynamic common block. This dynamic common block is written to the restart file on a restart edit.

### **3.1.2.5 *rmiedt***

The rmiedt subroutine processes the 301-399 minor edit request cards. Eight different categories of quantities can be requested: general, component, volume, junction, heat structure, reflood-related, reactor kinetics, and control system. The rmiedt subroutine creates a control file from the input cards that is used later by imiedt. This control file contains three word per minor edit request: (1) card number, e.g., 301, (2) variable requested, e.g., "p", and (3) parameter value, e.g., 345010000. This control file is dynamic common block 12. It has an id of filid(12), starts at filndx(12), and a length of filsiz(12). It is not written to the restart file on a restart because if the restart input file has any 301-399 cards in it, all the cards from the previous problem are deleted.

### **3.1.2.6 *rintrv***

The rintrv subroutine processes the 801-899 interactive input cards. These are used when the output of the code is interfaced with the Nuclear Plant Analyzer. This input can also be used by the user to enter commands via the command line during the transient. The control file for storage of the interactive input information is the intrac dynamic common block, number 7. It has an id of filid(7), starts at filndx(7), and a length of filsiz(7). This dynamic common block is written to the restart file on a restart.

### **3.1.2.7 *rtrip***

The rtrip subroutine processes the 400, 401-599, 600, 601-799, 20600000, 20600010-2060100000, and 20610010-29620000 trip input cards. If the 20600000 card is present, then the 2060 expanded series of trips are used. Otherwise, the 400-799 series of trips are used. The control file for storage of the trip information is the trblk dynamic common block, number 18. It has an id of filid(18), starts at filndx(18), and a length of filsiz(18). This dynamic common block is written to the restart file on a restart.

### **3.1.2.8 *rnoncn***

The rnoncn subroutine processes the 110 and 115 cards. It is called for new problems to calculate the mixture constants for the noncondensable gases. Up to five gases can be specified on these cards. Available gases include: helium, hydrogen, nitrogen, krypton, xenon, air, argon, and SF<sub>6</sub>. The mixture constants are: the gas constant, the heat capacity at constant volume, and the internal energy at 0 K. The mixture values are stored in the normal common block, /statec/, which is written to the restart file on a restart.

### **3.1.2.9 *rmflds***

The rmflds subroutine reads in the data from the optional 120-129 hydrodynamic system cards. The information on these cards define for each system, the reference volume, the elevation of the reference volume, the fluid type in this system, the name of the system, the system information flag, and the thermodynamic property file name for the system. The fluid types include both the original (old) light and heavy water fluids as well as the new light water fluid. The system information flag specifies whether noncondensables are present in the system at the initialization time. The control file for storage of the data processed by the rmflds subroutine is the sysdate dynamic common block, number 28. It has an id of filid(28), starts at filndx(28), and a length of filsiz(28). This dynamic common block is written to the restart file on a restart.

### **3.1.2.10 *rcompn***

The rcompn subroutine process the data on the CCC0000 card to determine the component name and type. It then calls the appropriate r-level subroutine, e.g., rpipe, raccum, etc., to process the rest of the cards for the component. The control file for storage of the component information is the cmpdat dynamic common block, number 3. It has an id of filid(3), starts at filndx(3), and a length of filsiz(3). This dynamic common block is written to the restart file on a restart. The various r-level subroutines that rompn calls are listed below.

#### **3.1.2.10.1 *rcdelt***

This subroutine processes the component deletion (“delete”) cards.

#### **3.1.2.10.2 *rpipe***

This subroutine processes the pipe (“pipe”) components as well as the annulus (“annulus”), pressurizer (“prizer”), and CANDU channel (“canchan”) components.

#### **3.1.2.10.3 *rtmdv***

This subroutine processes the time-dependent volume (“tmdpvol”) components.

#### **3.1.2.10.4 *rmtplj***

This subroutine processes the multiple junction (“mtpljun”) components.

#### **3.1.2.10.5 *rpump***

This subroutine processes the pump (“pump”) components.

#### **3.1.2.10.6 *rbrnch***

This subroutine processes the branch (“branch”) components as well as the jetmixer (“jetmixer”), separator (“separatr”), and ECC mixer (“eccmix”) components.

### **3.1.2.10.7 *rtmdj***

This subroutine processes the time-dependent junction (“tmdpjun”) components.

### **3.1.2.10.8 *rsngv***

This subroutine processes the single volume (“snglvol”) components.

### **3.1.2.10.9 *rsngj***

This subroutine processes the single junction (“sngljun”) components. The junction control word (“jefvcahs”) is parsed in this subroutine in the following way.

j-flag => ijt, if (ijt .ne. 0) then isjetjunflg25 = .true.

e-flag => ief, if (ief .eq. 1) then isdonprespvwrk15 = .true.

f-flag => icc, if (icc .eq. 1) then isnpccflflg2 = .true.

v-flag => ist, if (ist .eq. 1) then isstratinpdat17 = .true.

if (ist .eq. 2) then isstratinpdat18 = .true.

if (ist .eq. 3) then both of the above are set .true.

c-flag => ick, if (ick .ne. 0) isnochokflg4 = .true.

a-flag => irf, if (irf .eq. 2) then irf = 1 and ink = 1

if (irf .ne. 0) then isabrareachgflg8 = .true.

if (irk .eq. 1) then isnolosscoefabrjun29 = .true.

h-flag => ihf, if (ihf .ne. 0) then is2v3l1velflg9 = .true.

s-flag => isf, if (isf .eq. 1) then istomomfluxoff12 = .true.

if (isf .eq. 2) then isfrmommomfluxopt13 = .true.

if (isf .eq. 3) then both of the above are set .true.

### **3.1.2.10.10 *rvalve***

This subroutine processes the valve (“valve”) components.

### **3.1.2.10.11 *raccum***

This subroutine processes the accumulator (“accum”) components.

### **3.1.2.10.12 *rturb***

This subroutine processes the turbine (“turbine”) components.

### **3.1.2.10.13 *r3dcmp***

This subroutine processes the three dimensional components. This capability is not available in the current code.

### **3.1.2.11 *rhtcmp***

The rhtcmp subroutine processes the input data and sets up storage for the heat structures, cards 1CCCCxxx. It calls the ht1inp subroutine to process the geometry dependent input data. It also processes the gap data cards. The control file for storage of the heat structure information is the htsrcm dynamic common block, number 8. It has an id of filid(8), starts at filndx(8), and a length of filsiz(8). This dynamic common block is written to the restart file on a restart.

### **3.1.2.12 *rradht***

The rradht subroutine processes the radiation heat transfer input cards, 60000000 and 6SSNNxxx. This subroutine calls the simul function to invert the view factor matrix. The iradht subroutine will complete the processing of the radiation heat transfer database when it is called later. The control file for storage of the radiation heat transfer information is the radhtc dynamic common block, number 38. It has an id of filid(38), starts at filndx(38), and a length of filsiz(38). This dynamic common block is written to the restart file on a restart.

### **3.1.2.13 *rmadat***

The rmadat subroutine processes the thermal property composition data on cards 201MMMxx. The control file for storage of the thermal property composition information is the rmadac normal common block and mtbls dynamic common block, number 9. The mtbls dynamic common block has an id of filid(9), starts at filndx(9), and a length of filsiz(9). It is written to the restart file on a restart.

### **3.1.2.14 rrkin**

The rrkin subroutine processes the 30000000, 30000001, 30000002, 30000101-199, 30000011-20, 30000201-299, 30000301, and 30000401-499 reactor kinetics cards. It calls the rrknh subroutine to compute the initial conditions of the fission product decay groups. The rrkin subroutine calls the rrknp subroutine to process the 30000501-599, 30000601-699, 30000701-799, 30000801-899, 30001701-1799, 30001801-1899, 300019C1-19C8, and 30002001-2999 reactor kinetics feedback cards. The normal common block /rknatb/ contains the table for interpolating for the ANS79 neutron absorption effect in the decay heating calculations. The dynamic common block rrknc holds some pointers. The control file for storage of the reactor kinetics information is the rkinc dynamic common block, number 21. It has an id of filid(21), starts at filndx(21), and a length of filsiz(21). This dynamic common block is written to the restart file on a restart.

### **3.1.2.15 rgntbl**

The rgntbl subroutine processes the 202TTT00-99 general table cards. The control file for storage of the general table information is the gentble dynamic common block, number 11. It has an id of filid(11), starts at filndx(11), and a length of filsiz(11). This dynamic common block is written to the restart file on a restart.

### **3.1.2.16 rconvr**

The rconvr subroutine processes the 20500000, 205CCC00, and 205CCCC0 control variable cards. The control file for storage of the control variable information is the convarc dynamic common block, number 27. It has an id of filid(27), starts at filndx(27), and a length of filsiz(27). This dynamic common block is written to the restart file on a restart.

### **3.1.2.17 rusrvr**

The rusrvr subroutine processes the 20800001-9999 expanded plot variable cards. The control file for storage of the expanded plot variable information is the usrvvar dynamic common block, number 33. It has an id of filid(33), starts at filndx(33), and a length of filsiz(33). This dynamic common block is written to the restart file on a restart.

## **3.1.3 I-level Subroutines**

After all the r-level subroutines have read in the data from the input cards, the i-level subroutines perform checks on the data and computes and sets the offsets for any data that needs to be linked.

### **3.1.3.1 itstck**

The itstck subroutine checks to make sure the control variable specified on the 200 card, which was processed by the rtsc subroutine, does indeed exist.

### **3.1.3.2 *icmpn1***

The icmpn1 subroutine cross checks the component input and does the first pass at component initialization. It sets pointers in the component block to the first volume, nvco, and the first junction, njco, of the component. The time-dependent volume flag, istdpvol0, and time-dependent junction flag, istdpjunflg1, are set true for those components. It calls ipipe to set the to-volume number and the from-volume number for junctions internal to the pipe, annulus, multid, prizer, and canchan components. It calls isngj to set the to-volume number and the from-volume number for mtpljun, pump, branch, jetmixer, separatr, tmdpjun, sngljun, valve, accum, turbine, and eccmix components. It changes the volume coordinate direction in use flags, iscoorddirc14(i), iscoorddirc14(i+1), and iscoorddirc14(i+2), from false to true based on whether there is a connection to faces 1 or 2, 3 or 4, and 5 or 6. It sets the each of the flow regime map information words, ismapinfo05(i), ismapinfo05(i+1), and ismapinfo05(i+2), whenever the corresponding iscoorddirc14 flag is true. The map information word is set to 3 for an annulus, 4 for a pump, and 5 for an eccmix component. The gravitational constant, -gravcn, is stored in the gravv(i+2) or z-direction gravity vector. The constant, gravcn, is set in rmflds to the constant, 9.80665.

After the volume and junction values are initialized to zeros or some other appropriate value, the imlp subroutine is called to prepare the multiple loop table, load the number of volumes, junctions, and components per loop, and establish the order for processing. The pump flags: singlphasreferralflag6, twophasereferralflag10, headtorqmlptrflg8, and motortorqreferalfag12 are reset. The thermodynamic properties file or files are read into the fa array at filndx(6) by calling stread for the old H<sub>2</sub>O and D<sub>2</sub>O tables and newstread for the new H<sub>2</sub>O tables.

### **3.1.3.3 *itrip***

The itrip subroutine completes the checking and processing of the trip data and sets the initial values of the trips. It checks for illegal trip variables and gets the conversion factors for the constant factors. It checks the logical trip references and gets the location of the references.

### **3.1.3.4 *igntbl***

The igntbl subroutine cross checks and establishes linkages for the general tables and trips. It checks the trip references from the general tables.

### **3.1.3.5 *icompn***

The icompn subroutine controls the cross checking of component input and completes the component initialization. It sets table and trip information for time-dependent volume and time-dependent junction components. It calls istate to get the initial thermodynamic conditions for the volumes and ivelst to get the initial velocities for the junctions. It also sets the table and trip information for other types for components. For example, for valves, it sets the appropriate values for check, trip, inertial, motor, servo, and relief valves. It sets the pump table and trip values. It sets the accumulator junction isolation trip variable, acctrp. It sets the turbine terms. It calls flostj to set the junction flags like isupdwnjunflag27 and

ishorzvertjunflg26. These variables are confusing as to their meaning and will be renamed in a future version of RELAP5 to isdwnjunflg27 and isvertjunflg26. It calls flostv to set the vertical orientation and the flow regime map information variable, ismapinfo05.

The icompn1 subroutine calls invjt to prepare the inverted junction table and checks for the proper number of junctions per volume. The inverted junction table contains for each volume, the number of inlet and outlet junctions, invcnt, followed by four flags:

- isreversecoordflg0 - true if this junction has its coordinates reversed, i.e., it points from a larger x-coordinate towards a smaller x-coordinate
- isoutletflag1 - true if this is an outlet junction
- istoflag2 - true if this volume is hooked to the "to" end of the junction
- ismomfluxoff3 - true if momentum flux is off in this end of the junction.

In addition, the position number of the junction in the jundat junction block, invvno, and the index of the junction, invvnx, are stored in this table. The icompn1 subroutine calls levskt to prepare the level stack table. This table consists of five items:

- the index of the volume in the stack containing the mixture level
- the index of the volume at the top or head of the stack
- the index of the volume at the top or head of the level stack that is above this level stack
- the index of the volume that is at the top or head of the level stack that is below this level stack
- a flag that indicates that the stack has been processed

### **3.1.3.6 *ihtcmp***

The ihtcmp subroutine resolves geometry temperature referrals between heat structures. It also checks the geometry references. It checks the initial temperature references. It checks and sets pointers for the interconnections between the heat structure block, the material property block, and the general table block. It checks the source type and sets the appropriate pointers. It checks the compositions. It does the boundary related checks and checks the references to the general tables for temperature. It checks the references to the boundary volumes. It checks the PG CHFR specific data for validity. It checks the boundary condition type for validity. It sets the addresses for the trips in the general tables. It calls the steady-state initialization subroutine ht1sst to get the initialization for the transient solution and for the steady-state temperatures in those structures that specified it. It initializes the gap deformation model.

### **3.1.3.7 *iradht***

The iradht subroutine checks to make sure that no more than 100 conductors radiate. It checks the radiation conductor surface numbers for validity. It checks the validity of the head conductor number. It saves the offsets to the surface area, surface temperature, and fluid temperature. It checks the view factors

and the reciprocal relationship for validity. It calculates the inverted form-factor matrix by calling the simul matrix inversion function. It calls radht to calculate the radiosities and net radiative heat fluxes at the radiating surfaces.

### **3.1.3.8 *irflht***

The irflht subroutine checks the reflood input and sets up the reflood indices.

### **3.1.3.9 *invhts***

The invhts subroutine prepares the inverted heat structure table. The inverted table is a table that contains for each volume:

- the number of heat structures
- the inlet junction
- the outlet junction
- an index to the surface temperature that is in contact with the volume

### **3.1.3.10 *irkin***

The irkin subroutine completes the checking of the reactor kinetics input, sets pointers, and computes the bias reactivity. It checks the scram table reference and sets pointers. It checks the volume references and sets pointers. It checks the heat structure references and sets pointers. It computes the bias reactivity and builds the scram curves. It checks for separable or table feedback. If separable feedback, it does the volume feedback and heat structure feedback. If table feedback, it does the volume averaging, the heat structure averaging, and searches the coordinate tables. It computes the interpolation elements, evaluates the reactivity, and sets up for the first advancement. It changes the RELAP5 heat structure indices in the Doppler feedback data to offsets.

### **3.1.3.11 *iconvr***

The iconvr subroutine checks the control system legal variable requests and computes the initial values for all the components.

### **3.1.3.12 *imiedt***

The imiedt subroutine uses the control file, dynamic common block 12, created by rmiedt to create an internal buffer large enough for holding 50 values of each minor edit variable requested on the 301-399 cards. This buffer is in dynamic common block 16. This block has an id of filid(16), starts at filndx(16), and has a length of filsiz(16). Once the 50 values are obtained after taking 50 time steps at the maximum time step, the values are written to the output file, and the buffer area is cleared so that it can accumulate the next 50 values. This dynamic common block is written to the restart file on a restart edit.

### **3.1.3.13 *issi***

The issi subroutine does the checking and setup for the steady-state initialization option on the 140-147 cards.

### **3.1.3.14 *iusrvr***

The iusrvr subroutine sets up the pointers to the expanded plot variables that are input on the 20800001-9999 cards.

### **3.1.3.15 *wrplid***

The wrplid subroutine writes the headers for the plot variables to the restart-plot file. Record 1 consists of two integers, 3 and 8. The first integer is the number of 8-byte words in the next record and the second integer is the size of the words in bytes. Record 2 consists of three words. Word 1 is the character string “plotinf”. Word 2 is the number of variables written each time a plot record is written. Word 3 is this number divided by two plus one. It is essentially half the number of variables rounded up to make sure that there is enough room if the number of plot variables is an odd number. Record 2 is written as three floating point 8-byte values. Record 3 consists of two integers: the number of variables and the integer 8 to indicate that the words are 8-bytes long. Record 4 consists of the character string “plotalf” followed by all the names of the plot variables, e.g., “time”, “count”, etc. Record 5 is like the Record 3 and consists of the two integers: the number of variables and the number 8. Record 6 consists of the character string “plotnum” followed by the numeric parameter for each the variables, e.g., volume number for volume variables, junction number for junction variables, zero for the time variable, etc. These six records are only written at the beginning of a new restart-plot file and are essentially the labels for the data values that are written later. These six records will be written again after a restart if there are changes in the number of variables, e.g., additional variables added to the 2080001-9999 cards, deleted components, etc. The actual data values for these variables are written to the restart-plot file in the pltwrt subroutine.

## **3.1.4 Card Number, Subroutine Name, and Common Block Connections**

**Table 3.1-1** relates the card numbers in the input deck with the subroutine names that are used to process those cards, and the normal or dynamic common block in which the information is stored.

**Table 3.1-1** Connection between Card Numbers, Subroutines, and Common Blocks

Card Numbers	Type of Input	Subroutines	Common Block	id in filid(id)
1	Developmental model control	rchng	/contrl/	
2-5	Debug printout of selected volumes and junctions	rhelp	/contrl/	

**Table 3.1-1** Connection between Card Numbers, Subroutines, and Common Blocks

<b>Card Numbers</b>	<b>Type of Input</b>	<b>Subroutines</b>	<b>Common Block</b>	<b>id in filid(id)</b>
100-101	Problem type and option, Input check or run option	inputd	/comctl/	
102	Units selection	rnewp	/contrl/	
103	Restart input file control	rrestf, srestf		
104	Restart-Plot file control	rrestf, rrstd	/ufilef/	
105	CPU time remaining and diagnostics	rnewp	/contrl/	
110, 115	Noncondensable gas species, noncondensable mass fractions	rnoncn	/statec/	
120-129	Hydrodynamic system	rmflds icmpn1 stcset	sysdatc steam tables /stcom/ /stcblk/	28 6
140-147	Self-initialization option control	rssi	ssiblk	37
200-299	Time step control	rtsc	tstpct	2
301-399	Minor edits	rmiedt imiedt	miedtc	12 16
20800001-20809999	Expanded plot variables	rusrvr	usrvar	33
400-799 20600000-20620000	Trips, variable and logical, trip stop advancement	rtrip itrip	trpb lk	18
801-899	Interactive input	rintrv	intrac	7
1001-1999	Strip requests	rstrip	none	3
1001-1999	Compare dump files	cmpcom		
CCC0000	Hydrodynamic component name and type	rcompn	cmpdat /rcomp/	3
SNGVOL CCC0101-CCC0200	Single-volume component	rsngv	cmpdat voldat	3 4
TMDPVOL CCC0101-CCC0299	Time-dependent volume component	rtmdv	cmpdat voldat	3 4

**Table 3.1-1** Connection between Card Numbers, Subroutines, and Common Blocks

<b>Card Numbers</b>	<b>Type of Input</b>	<b>Subroutines</b>	<b>Common Block</b>	<b>id in filid(id)</b>
SNGLJUN CCC0101-CCC0201	Single-junction component	rsngj	cmpdat jundat	3 5
TMDPJUN CCC0101-CCC0299	Time-dependent junction component	rtmdj	cmpdat jundat	3 5
PIPE CCC0101-CCC3199	Pipe component	rpipe	cmpdat voldat jundat	3 4 5
ANNULUS CCC0101-CCC3199	Annulus component	rpipe	cmpdat voldat jundat	3 4 5
PRIZER CCC0001-CCC3199	Pressurizer component	rpipe	cmpdat voldat jundat	3 4 5
CANDU CCC0001-CCC3199	CANDU channel component	rpipe	cmpdat voldat jundat	3 4 5
BRANCH CCC0001-CCCN201	Branch component	rbrnch	cmpdat voldat jundat	3 4 5
SEPARATR CCC0001-CCC0600	Separator component	rbrnch	cmpdat voldat jundat	3 4 5
JETMIXER CCC0001-CCCN201	Jetmixer component	rbrnch	cmpdat voldat jundat	3 4 5
TURBINE CCC0001-CCC0400	Turbine component	rturb	cmpdat voldat jundat	3 4 5
ECCMIX CCC0001-CCCN201	ECC mixer component	rbrnch	cmpdat voldat jundat	3 4 5
VALVE CCC0101-CCC0499	Valve component	rvalve	cmpdat jundat	3 5
PUMP CCC0101-CCC6199	Pump component	rpump	cmpdat voldat jundat	3 4 5

**Table 3.1-1** Connection between Card Numbers, Subroutines, and Common Blocks

<b>Card Numbers</b>	<b>Type of Input</b>	<b>Subroutines</b>	<b>Common Block</b>	<b>id in filid(id)</b>
MTPLJUN CCC0001- CCC3NNM	Multiple junction component	rmtplj	cmpdat voldat jundat	3 4 5
ACCUM CCC0101-CCC2200	Accumulator component	raccum	cmpdat voldat jundat	3 4 5
1CCCC000- 1CCCC999	Heat structures	rhtcmp	htsrcm	8
60000000- 6SSNN199	Radiation heat slabs	rradht	radhtc	38
201MMM00- 201MMM99	Heat structure thermal properties	rmadat	mtbls /matdat/	9
202TTT00- 201TTT99	General table data	rgntbl	gentblc	11
20500000- 205CCC98 or 20500000- 205CCCC8	Control system	rconvr	convarc	27
30000000-30000499	Reactor kinetics	rrkin	rkinc /rknatb/ rrkinc/	21
30000011-30000020 and 30000501-30002999	Reactor kinetics feedback	rrkinp	rkinc /rrkinc/	21
	Inverted junction table	invjt	invtbl	10
	Time dependent volumes and junction pointers	trnset	tdpptr	13
	Inverted heat structure table	invhts	invhtb	14
	Statistics for transient run	icompn	statc	20
	Statistics for steady-state run	trnset	sscntr	24
	Hydrodynamic systems	imlp	lpdat	30
20900000-20903000	PVM coupling to other codes, primarily CONTAIN	rr5pvmc	r5pvmc	31

**Table 3.1-1** Connection between Card Numbers, Subroutines, and Common Blocks

<b>Card Numbers</b>	<b>Type of Input</b>	<b>Subroutines</b>	<b>Common Block</b>	<b>id in filid(id)</b>
	Reflood input check and index setup	irflht	htrflb /rflhtc/	32
	SCDAP fission product data input	fpinit	scddat	34
	List vectors for various components	trnset	lvectr	35
	SCDAP data		scddat	36
	Sparse matrix controls	tsets1		40
	Level stack table	levskt	levtbl	43
all	All the input cards	inp		1

## 3.2 Transient Processing Subroutines

The transient is controlled by the trnctl subroutine, which is called by relap5. Trnctl calls three subroutines: trnset, tran, and trnfin.

### 3.2.1 trnset

The trnset subroutine initializes the problem database for the transient. It moves the thermodynamic property file, filid(6), to low memory. It sets numerous offsets. One example is the offset to the control variable that is used to control the time step size, which was input on the 200 card. The trnset subroutine sets offsets in the lpdat dynamic common block for pointers to the start of the component, volume, and junction dynamic common blocks. It sets the level stack indices. It sets component, volume, and junction loop parameters and sets indices in the component block. It sets indices in the inverted heat structures block. It sets pointers for time-dependent volumes and junctions that have more than one set of data, i.e., the ones that have to be interpolated. It sets indices for accumulators, heat structures, reflood calculations, user supplied restart-plot file variables, addresses for trips in general tables, addresses for trip statements, addresses for minor edits, and steady-state statistics counters. It also sets up separate list vectors for accumulators, pumps, separators, valves, turbines, jetmixers and eccmixers, 3-D components (none in RELAP5 now), real volumes, time-dependent volumes, volumes with no friction, volumes with friction, abrupt area change junctions, horizontal flow junctions, and junction-source loops. It builds a list of all the junctions and a list of all the volumes in the model. It sets up space for the scratch list vector. It sets the volume and junction flags that are used in the Godunov method. At the very end of the trnset subroutine, a call to the fildmp subroutine is made. This subroutine prints a list of all the dynamic common blocks, their number, their filid value, their name (comdeck name), where they start in the fa array, and their length to the output file.

The list vectors were put together to reduce the running time on a vector computer, e.g., Cray. However, most of the computers that the RELAP5 code runs on today are not vector computers, so this extra work of creating the list vectors which allows the transient part of the code to use separate do loops for a set of volumes or junctions that are in the list vector may actually increase the running time. These list vectors probably do more harm than good in that it makes the maintenance of the code much more difficult. There seem to be separate loops for every little part of the code. The maintenance would be much easier if there was just one loop over all the volumes with appropriate if statements to exclude or include the appropriate volumes for the type of processing that was being performed in that section of coding. The coding is gradually being migrated in that direction.

### 3.2.2 tran

The `tran` subroutine controls the transient. It calls the following subroutines: `dtstep`, `chklev`, `tstate`, `htadv`, `pzrlev`, `hydro`, `rkin`, and `convar`. Below is a flowchart for `tran`. Some of the calls to the subroutines are conditional. For example, the level model subroutine, `chklev`, is not called if the level model is not active. The level model is active if the variable `islevelmodel0` is true. Other subroutines are only called if a corresponding dynamic common block has been initialized. For example, `tstate` is called only if the time-dependent volume and junction block, `tdpptr`, which is dynamic common block 13, exists. The if tests involving `cpurei` are related to stopping the transient at the appropriate time step number based on the input on optional card 105. The variable `done` is set to a positive non-zero number when the transient is to be terminated. The variable `succes` is used to determine if the time step is successful. It is zero for a successful time step and non zero for an unsuccessful time step. The `succes` variable is checked in the subroutine `mover` to determine if the new-time variables are to be moved to their old-time locations or if the old-time values are to be moved to the new-time locations and the time step repeated.

```

return
end

```

The subroutines called by tran are briefly described in the following sections.

### **3.2.2.1 dtstep**

Time step control is done in the subroutine, dtstep. This is the first subroutine called by tran. RELAP5 starts a new transient using the minimum time step size on the appropriate 201-199 card. If after the time step taken, the mass error is below the preset limit errlo, RELAP5 increases the time step by 10% for the next time step. If the mass error is greater than the preset lower limit errlo and less than the preset upper limit errhi, the time step is not changed for the next time step. If the mass error is greater than the preset upper limit errhi, the time step is repeated with the time step size cut in half. These preset limits are in the /tsctlc/ common block. They are initialized in data statements in the blkdata subroutine to 0.0008 for errlo and 0.008 for errhi. The mass error used for time step control is defined as the difference between the mixture density from the equation of state, which is a function of the new-time pressure and internal energies, and the mixture density determined by the initial mixture density in a volume plus any additions minus any subtractions as a result of convection and mass transfer during the time step. This difference is divided by the mixture density to normalize it and the result is the mass error. So, the mass error is really the fractional difference in the volume mass computed by two different methods: equation of state and continuity equation. The mass error is computed for each volume in the masserror subroutine, which is called from the state subroutine. Only the maximum mass error in any volume is used in the comparison with errlo and errhi to determine if the time step should be changed for the next time step.

The sysmer variable that is printed out in the major edits for each system is defined differently than the mass error used for time step control. The sysmer variable is the total system mass from the continuity equation minus the total system mass from the state equation. Thus, sysmer could be zero for a time step in which the time step size is cut in half or in which the time step is increased by 10%. The variable emass is the sum of the sysmer values for each of the systems, e.g., primary, secondary. If emass is negative, the entire system is gaining mass, and if emass is positive, the system is losing mass. Since emass is the sum of the mass errors for multiple systems, it too can be deceiving in that one system could be gaining mass while the other system could be loosing mass, which would result in the emass value not changing.

The time step size is also be limited by the Courant limit. The Courant limit is determined at the end of a time step and is used to limit the time step size for the next time step. Two Courant limits are computed and the smallest one is used for the limit. One of the limits is volume based and other is junction based. Both limits are computed in the courn1 subroutine. The idea behind the volume Courant limit is to limit the time step size so as to not remove more material from a volume than is present in the volume at the beginning of the time step. The idea behind the junction Courant limit is to limit the time step size so as to not remove more momentum from a junction than is present in the junction at the beginning of the time step. The minimum Courant time step size limit is written to the screen file. If the volume Courant limit had the smallest limit, the number of the volume that had the limiting time step size is written to the screen file. If the junction Courant limit had the smallest limit, the number of the upstream volume for that junction is written to the screen file. So in both cases, the number of a volume is written to the screen file.

In order to distinguish between the volume and the junction limits and between the liquid and vapor phase limits, upper or lower case letters "f" or "g" are also written to the screen file to signify that liquid or vapor had the limit. Upper case letters are used if the limit was volume based and lower case letters are used if the limit was junction based.

The original method of computing the Courant limit in RELAP5 is in the cournt subroutine. It is not used anymore.

After a restart, the time step size is the same as it would have been if the restart had not occurred. Also the end time on the time step cards is not hit exactly. The code overruns the end time because the last time step may not be exactly equal to the value necessary to end at the end time on the appropriate 201-299 card. This is in contrast to the earlier versions of the code, i.e. versions before 3.3Beta, in which the last two time step sizes were adjusted so as to hit the end time exactly. This adjustment of the last time steps caused the code to compute different results whenever a restart time was inserted in the middle of a long run. This has been fixed in 3.3Beta, and now there is no adjustment of the last two time step sizes. If the user wants to hit the end time exactly, then the end time on the appropriate time step card has to have a minus sign in front of it.

### **3.2.2.2 *chklev***

The chklev subroutine controls the movement of two-phase levels between volumes. This subroutine limits the maximum time step size to be less than 0.5 ms when the level is going to cross a junction. It keeps cutting the time step by factors of two until it is less than 0.5 ms. To determine if the level is going to cross a junction in the next time step, the level subroutine is called with an argument of 1. This call causes the level subroutine to extrapolate the level position using its past position and the past level velocity.

The output variables from chklev are dt, dtht, timehy, and nrepel.

### **3.2.2.3 *tstate***

The tstate subroutine updates the time-dependent volume and junction conditions for the next time step. It interpolates between any tabular values that were entered on the time-dependent volume or time-dependent junction cards.

For time-dependent volumes, the input variables are the search variable, e.g. time, and the appropriate tabulated set of thermodynamic properties. The thermodynamic properties that are in the table depend on the input flag. The output variables are the thermodynamic properties interpolated between the tabular values entered on the input cards.

For time-dependent junctions, the input variables are the search variable, e.g., time, and the tabular velocities or mass flow rates that are also input. The output variables are the two velocities, velfj and velg, interpolated between the tabular values entered or computed from the data on the input cards.

### **3.2.2.4 htadv**

The htadv subroutine is the main driver for the heat transfer advancement. If the radiation input has not been processed yet, and there are radiation heat structures in the problem, the htadv subroutine calls radht to process the radiation input cards. It also calls qfmove to set up for the reflood calculations if necessary. It loops over all the heat structures and computes the power input to the hydrodynamic volumes from heat transfer from the heat structures and any direct heating by calling the ht1tdp subroutine. The ht1tdp subroutine calls the madata, kloss, htcond, and qmwr subroutines. The madata subroutine calls the gasthc and gapcon subroutines. The htcond subroutine calls the htrc1 subroutine. The htrc1 subroutine calls stcset, dittus, sth2x2, chfcal, prednb, prebun, suboil, pstdnb, condn2, and condens subroutines. The chfcal subroutine calls chfsrl, chfd2o, chforn, chftab, chfpgp, chfpgg, chfgpf, and chfkup subroutines. The condn2 subroutine calls the htfilm, dittus, and ncwall subroutines. The condens subroutine calls the dittus and nonend subroutines. The details of all these subroutines will be discussed later.

### **3.2.2.5 pzrlev**

The pzrlev subroutine calculates the water level in the prizer pressurizer component.

### **3.2.2.6 hydro**

The hydro subroutine is the main driver for the hydrodynamics solution. The details of the subroutines that hydro calls will be covered in a later section. The following subroutines are called by hydro.

- stcset loads the proper thermodynamic properties into the stcom common block
- valve computes the behavior of the valves
- acclev predicts if the accumulator will empty in this time step
- volvel computes the junction averaged velocity for use in the wall friction calculations and the donor volume averaged velocities for use in the momentum flux calculation
- tfront detects if a thermal front exists in a volume
- phantv computes the interphase heat transfer and some information for vexplt
- phantj computes the interphase drag and some information for vexplt
- fwdrag computes the wall drag terms including flow regimes and the HTFS two-phase friction factor correlation
- hloss calculates the void fractions at the throat and downstream of an abrupt area change
- vexplt computes the explicit liquid and vapor velocities, the pressure gradient coefficients needed for the implicit pressure solution, and the old-time source terms for the mass and energy equations
- jchoke computes the choking velocity at junctions using the Henry-Fauske model or the original RELAP5 choking model.

- ccfl determines if countercurrent flow limiting occurs and if it does, a flooding correlation of the Wallis-Kudateladze type is used instead of the difference momentum equation
- jprop computes the junction donor properties from the upstream volume
- vfinl calls the subroutines preseq and syssol to load and solve the pressure matrix, computes the new-time velocities, mass flow rates, and checks for bad donor and water packing
- eqfinl computes the new-time pressure and carries out the back substitution to obtain the new-time values for the noncondensable quality, vapor internal energy, liquid internal energy, vapor void fraction, pressure, vapor generation rate, and mixture density
- vimlt computes the new-time liquid and vapor velocities using nearly-implicit coupled momentum equations and the old-time source terms for the mass and energy
- pimplt carries out the back substitution to obtain the new-time pressure and boron density as well as the intermediate-time liquid internal energy, vapor internal energy, void fraction, noncondensable quality, vapor generation rate, and remainder of the source terms for the mass and energy equations used in the nearly-implicit solution scheme
- simplt computes the new-time liquid internal energy, vapor internal energy, void fraction, noncondensable quality, and boron density using nearly-implicit convective terms in the mass and energy equations
- brntrn computes the new-time boron concentration using a second-order Godunov method
- state evaluates the equation of state for all components
- level detects a two-phase level in a volume and computes the level propagation velocity
- vlevela computes the volume-averaged velocities for use in the momentum flux calculations
- htfinl advances the heat structures by adding the effects of the changed fluid temperatures

### **3.2.2.7 rkin**

The rkin subroutine advances the point kinetics solution for one time step.

### **3.2.2.8 convar**

The convar subroutine advances the control variables for one time step.

## **3.2.3 trnfin**

The trnfin subroutine cleans up after the transient successfully completes or fails. It rewinds and closes the restart-plot file. It releases all the dynamic common blocks that were set up prior to the transient.

### 3.3 Hydrodynamics Subroutines

The hydro subroutine is the main driver for the hydrodynamics solution. A abbreviated flowchart of hydro is shown below.

```

subroutine hydro
a<<<<<<< do 10 m = 1, nloops(2,issys)
a      call stcset(volmat(2,liv(2,issys)))
a      call valve(.false.)
a      call acclev
a      call volvel
a      call tfront
a      call phantv
a      call phantj(0)
a      call fwdrag
a      call hloss
a b<<<<<<< if (.not.istwostepflag7) then
a b   c  Semi-explicit advancement scheme
a b       call vexplt
a b       call jprop(1)
a b       call jchoke
a b       call ccfl
a b       call vfinl
a b       call eqfinl
a b<<<<<<< else
a b   c  Nearly implicit advancement scheme
a b       call vimplt
a b       call pimplt
a b       call jprop(0)
a b       call simplt
a b>>>>>>> endif
a       call brntrn
a       call state(1)
a       call level(0)
a       call jprop(0)
a       call vlvela
a>>>>> 10  continue
c<<<<<<< if (succes .ne. 2) then
c       if (isimplicithttrnsfr6) call htfinl
c>>>>>>> endif
d<<<<<<< if (succes .ne. 2) then
d       if (errmax .ge. errhi) succes = max (succes, 1)
d>>>>>>> endif
      return
end

```

The hydro subroutine is essentially one loop over each of the separate systems in the model. Some of the subroutine calls are conditional in that they are only called when a certain condition is true. This detail is not shown in the flowchart in order to simplify the diagram. Each of the subroutines well be explained in more detail below.

#### 3.3.1 stcset

The stcset subroutine loads the proper thermodynamic properties into the /stcom/ common block. Thus, the heavy water properties get loaded if the primary system contains heavy water and light water properties get loaded if the secondary system contains light water. This subroutine also sets a pointer to the proper location in the fa array where the property tables start.

### 3.3.2 valve

The valve subroutine computes the behavior of the valves. Internally, there is a loop over all the valves in the system and appropriate subroutines are called to process that type of valve. The types of valves include:

1. vname = chkvlv (check valve)
2. vname = trpvlv (trip valve)
3. vname = inrvlv (inertial valve)
4. vname = mtrvlv (motor valve)
5. vname = srvvlv (servo valve)
6. vname = rlfvlv (relief valve)

In the coding, the variable vlvnm is set equal to the number corresponding to the type of valve in the above list.

### 3.3.3 acclev

The acclev subroutine has a loop over all the accumulators in the system. It predicts if the accumulator will empty in this time step. If the accumulator will empty, the effective liquid and vapor void fractions are computed and used for the next time step. This prediction procedure prevents the large negative mass error that occurs during the time step when an accumulator empties. Flow from an accumulator is assumed to be all liquid, even during the time step in which it empties. However, during the time step in which the accumulator empties, only the first part of the time step will have liquid flowing. During the second part of the time step, vapor will be flowing. To account for this properly, the acclev subroutine modifies the junction void fractions for this time step so that only the remaining liquid in the accumulators is added to the primary system.

### 3.3.4 volvel

The volvel subroutine computes the junction phasic mean absolute values of the velocities normalized to the volume flow area for use in the wall friction calculations. It loops over all the junctions in the system in two loops. The first loop is a vectorized loop over junctions and that accumulates sums in the two volumes that the junction connects. The ordering of the junctions comes from a list vector that was constructed in tsets1. The junctions are ordered so that no two adjacent junctions have the same volumes attached to them. In a pipe, this would be odd numbered junctions followed by the even numbered junctions, providing the pipe had three or more internal junctions. This way, when the sums were accumulated in the volume slots, they would not overwrite each other on a vector computer. The second

loop is a scalar loop and covers all the junctions that could not be put into the first list vector. The Cray directives "cdir\$ ivdep" and "cdir\$ nextscalar" give you a hint as to what each loop is for.

The volvel subroutine also computes the donored volume-averaged velocity that is used in the momentum flux calculation. The momentum flux at a momentum cell face in the difference equation notation is the product of the density times the velocity at the face, which gives the momentum, times another velocity to get the momentum flux. In RELAP5, the volume-averaged velocity is used for the momentum since it is located at the same place as the density, which is at the center of the volume. In RELAP5, the velocity that is used to multiply the momentum by to get the momentum flux is not the same volume-averaged velocity because that would result in an unstable finite difference scheme. Instead, a donored velocity is used. This donored velocity is an averaged upstream junction velocity. In a pipe, this donored velocity is just the upstream (based on the direction of the volume-averaged velocity) junction velocity. This donored velocity is computed in the vlevela subroutine, which is covered later.

### **3.3.5 tfront**

The tfront subroutine detects if a thermal front exists in a volume. If it does, the tfront subroutine computes the thermal front propagation velocity, vfront, and the front position relative to the bottom of the cell, dfront. Inside tfront there is a loop over all the volumes that have the thermal front flag set. Inside this volume loop, there is a loop over all the junctions attached to this volume. Only vertical junctions are considered in this junction loop. A thermal front exists in a volume if the absolute difference in liquid density between the volume above and the volume with the interface divided by the liquid density of the volume with the interface is greater than 1%. A similar condition relating the volume below and the volume with the interface is also checked. Once a thermal front is detected, the code assumes that the liquid internal energy in the volume with the interface is discontinuous. Inside this volume that has the thermal front, the internal energy of the liquid below the interface is assumed to be the same as the internal energy of the liquid in the volume below the interface, and the internal energy of the liquid above the interface is assumed to be the same as the internal energy of the liquid in the volume above the volume with the interface. This is essentially donor-acceptor differencing of the liquid internal energy for volumes in which a thermal front is detected.

### **3.3.6 phantv**

The phantv subroutine calculates the interphase heat transfer coefficient times an area per unit volume. The variables hif, hig, and hgf are used for these quantities. The letters after the h in these variable names stand for interface-to-liquid, the interface-to-liquid, and vapor-to-liquid. The phantv subroutine also calculates the wall friction and K-loss variables: fidxup, pfinrg, fwfxaf, fwfxag, fwalf, fwalg, and celvec. It calculates the flow regime, floreg. In earlier versions of RELAP5, this subroutine was very large (>3000 lines). The phantv subroutine was re-engineered and is now a driver for a number of smaller, specific subroutines. It calls the eccmxv, pumphifreg, horizhifreg, verthifreg, wethif, dryhif, vstrathif, levelhif, jetjhif, filterhif, and bugouthif subroutines. Some of these subroutines call other subroutines like bubhif, slughif, amisthif, dispwethif, horizhif, invannhif, invslughif, dispdryhif, htlev, ncfilm and helphd. Some of

these subroutines call lower level subroutines like hifbub, fidisv, and htfilm. Below is a calling tree for phantv.

```

eccmxv - flow regime and hif and hig for ECC mixers
pumphifreg - flow regime for pumps
horizhifreg - flow regime for horizontal components
verthifreg - flow regime for vertical components
wethif - output variables for volumes with wet walls
    bubhif - bubbly flow hif and hig
        hifbub - hif for bubbly flow
    slughif - slug flow hif and hig
        hifbub
    amisthif - annular mist hif and hig
        fidisv - mist part of hif and hig
        htfilm - film part of hif and hig
    dispwethif - dispersed wet wall hif and hig
        fidisv
    horizhif - horizontally stratified hif and hig
dryhif - volumes with dry walls (post CHF)
    invannhif - inverted annular hif and hig
        fidisv
        hifbub
    invslughif - inverted slug hif and hig
        fidisv
    dispdryhif - dispersed hif and hig
        fidisv
    vstrathif - vertically stratified hif and hig
levelhif - stratified per level model hif and hig
jetjhif - stratified volumes connected to jet junction hif and hig
    htlev
filterhif - time step smoothing for all volumes
    ncfilm
bugouthif - debug output for all volumes
    helphd

```

### **3.3.6.1 eccmxv**

The eccmxv subroutine calculates the mixing of ECC liquid with cold leg flow and steam condensation in different flow regimes in the ECC mixing component, eccmxv, and returns the flow regime number as well as the liquid and vapor interphase heat transfer coefficients. Possible flow regime numbers range between 16 and 26. They are essentially the non-eccmixer flow regime number plus 15.

### **3.3.6.2 pumphifreg**

The pumphifreg subroutine calculates the flow regime in a pump. Three flow regimes are possible, bubbly (flomap = 1), transition (flomap = 2) and dispersed (flomap = 3). If voidg is less than or equal to 0.5, the regime is bubbly. If voidg is greater than 0.5 and less than 0.95, the regime is transition. If voidg is greater than or equal to 0.95, the regime is dispersed. This subroutine also returns fbub and fdis.

### **3.3.6.3 horizhifreg**

The horizhifreg subroutine calculates the horizontal map flow regime. Four flow regimes are possible: bubbly (flomap = 4), slug (flomap = 5), annular mist (flomap = 6), and dispersed (flomap = 7). If voidg is less than or equal to alphab, which is about 0.25, the regime is bubbly, else if voidg is less than or

equal to alphac, which is about 0.8, the regime is slug. If the voidf is greater than alphad, which is about 0.0001 (voidg less than 0.9999), the regime is annular mist. Above voidg greater than 0.9999, the regime is dispersed.

### **3.3.6.4 *verthifreg***

The verthifreg subroutine calculates the interphase heat transfer coefficients for vertical volumes and some information for use in vexplt. Eight flow regimes are possible: bubbly (flomap = 4), inverted annular (flomap = 8), slug (flomap = 5), inverted slug (flomap = 9), annular mist (flomap = 6), dispersed or mist (flomap = 10), mist pre-CHF (flomap = 7), and mist post-CHF (flomap = 11). If voidg is less than or equal to alphab, which is about 0.5, the regime is bubbly as long as poschf is false. Otherwise, the regime is inverted annular. If voidg is greater than alphab and less than alphac, which is about 0.8, the regime is slug as long as poschf is false. Otherwise, the regime is inverted slug. If voidg is greater than alphac and less than (1.0 - alphad), which is about 0.9999, the regime is annular mist as long as poschf is false. Otherwise, the regime is mist. If voidg is greater than or equal to (1.0 - alphad), the regime is mist pre-CHF as long as poschf is false. Otherwise, the regime is mist post-CHF.

### **3.3.6.5 *wethif***

The wethif subroutine calculates the wetted wall interphase heat transfer coefficients. It uses the flow regime determined from earlier subroutines and then calls various subroutines to compute the interphase heat transfer coefficients. It calls bubhif if the flow regime is bubbly, slughif if the flow regime is slug, amisthif if the flow regime is annular mist, dispwethif if the flow regime is dispersed, or horizhif if the flow regime is horizontally stratified.

### **3.3.6.6 *dryhif***

The dryhif subroutine calculates the interphase heat transfer coefficients for dry walls. It uses the flow regime determined from earlier subroutines and then calls various subroutines to compute the interphase heat transfer coefficients. It calls invannhif if the flow regime is inverted annular, invslughif if the flow regime is inverted slug, or dispdryhif if the flow regime is dispersed.

### **3.3.6.7 *vstrathif***

The vstrathif subroutine calculates the interphase heat transfer coefficients for volumes that have vertically stratified liquid. In this case, flomap = 13.

### **3.3.6.8 *levelhif***

The levelhif subroutine calculates the interphase heat transfer coefficients for volumes in which there is a level. In this case, flomap = 14.

**3.3.6.9 *jethif***

The jetjhif subroutine calculates the interphase heat transfer coefficients for volumes which are connected to a jet junction. In this case, flomap = 15.

**3.3.6.10 *filterhif***

The filterhif subroutine does the final adjustments on the interphase heat transfer coefficients. There is also a time smoothing or under relaxation of the interphase heat transfer coefficients.

**3.3.6.11 *bugouthif***

The bugouthif subroutine gives debug printout of the phantv variables when the help variable is not equal to zero.

**3.3.7 *phantj***

The phantj subroutine calculates the interfacial shear (drag) coefficients, fij and fxj, and void distribution coefficient, c0j, for each normal junction, i.e., non time-dependent junctions. It calculates the pressure drop variables, dpstf, dpstfk, and dpstfl, needed by the vexplt subroutine. It calculates the junction flow regime, florgj. In earlier versions of RELAP5, this subroutine was very large (> 3000 lines). The phantj subroutine was re-engineered and is now a driver for a number of smaller, specific subroutines. It calls the eccmxj, pumpdragreg, horizdragreg, vertdragreg, wetdrag, drydrag, adjusthif, chng18drag, filterdrag, finaldrag, and bugoutdrag subroutines. Some of these second level subroutines call third level subroutines like fidis2, fidisj, slugdrag, amistdrag, dispdrag, horizstartdrag, invanndrag, and invslugdrag. Some of third level subroutines call fourth level subroutines like fidis2, and fidisj. Below is a calling tree for phantj.

```

eccjmj - flow regime and drag for ECC mixers
    fidis2
pumpdragreg - flow regime for pumps
horizdragreg - flow regime for horizontal components
vertdragreg - flow regime for vertical components
wetdrag - drag for volumes with wet walls
    bubdrag - drag for bubbly flow regime
        fidis2 - drag for bubbles and droplets
        fidisj - drag for dispersed vapor flow
    slugdrag - drag for slug flow regime
        fidis2
        fidisj
    amistdrag - drag for annular mist flow regime
        fidis2
    dispdrag - drag for dispersed flow regime
        fidis2
    hstratdrag - drag for horizontally stratified flow regime
drydrag - drag for volumes with dry walls
    invanndrag - drag for inverted annular regime
        fidis2
    invslugdrag - drag for inverted slug regime
        fidis2
        fidisj
adjusthif - adjusts hif when flow regime is vertically stratified

```

```

chng18drag - adjusts drag when option 18 is activated
filterdrag - old time weights the drag values
finaldrag - loads junction flow regime and calculates void gradient forces
buboutdrag - debug output

```

### **3.3.7.1 *eccmxj***

The eccmxj subroutine calculates the interphase friction factor for the volume in which the mixing of ECC liquid with cold leg flow and steam condensation occurs.

### **3.3.7.2 *pumpdragreg***

The pumpdragreg subroutine computes the interphase drag flow regime for pumps. Three high mixing junction flow regimes are possible: bubbly (flompj = 1), transition (flompj = 2), and dispersed (flompj = 3). If voidj is less than or equal to 0.5, the regime is bubbly. If voidj is greater than 0.5 and less than 0.95, the regime is transition. If voidj is greater than 0.95, the regime is dispersed. The junction weighted void fraction, voidj, is defined in phantj.

### **3.3.7.3 *horizdragreg***

The horizdragreg subroutine computes the horizontal interphase drag flow regime parameters. Four junction flow regimes are possible: bubbly (flompj = 4), slug (flompj = 5), annular mist (flompj = 6), and dispersed (flompj = 7). If voidj is less than or equal to alphab, which is about 0.25, the regime is bubbly. If voidj is greater than alphab and less than or equal to alphac, which is about 0.8, the regime is slug. If voidj is greater than alphac and less than or equal to (1.0 - alphad), which is 0.9999, the regime is annular mist. If voidj is greater than (1.0 - alphad), the regime is dispersed. The junction weighted void fraction, voidj, is defined in phantj.

### **3.3.7.4 *vertdragreg***

The vertdragreg subroutine computes the interphase drag flow regime for vertical junctions. Eight junction flow regimes are possible: bubbly (flompj = 4), inverted annular (flompj = 8), slug (flompj = 5), inverted slug (flompj = 9), annular mist (flompj = 6), mist (flompj = 10), dispersed preCHF (flompj = 7), and dispersed postCHF (flompj = 11). If voidj is less than or equal to alphab, which is about 0.5, the regime is bubbly as long as poschf is false. Otherwise, the regime is inverted annular. If voidj is greater than alphab and less than alphac, which is about 0.8, the regime is slug as long as poschf is false. Otherwise, the regime is inverted slug. If voidj is greater than alphac and less than (1.0 - alphad), which is 0.9999, the regime is annular mist as long as poschf is false. Otherwise, the regime is mist. If voidj is greater than or equal to (1.0 - alphad), the regime is mist pre-CHF as long as poschf is false. Otherwise, the regime is mist post-CHF.

### **3.3.7.5 *wetdrag***

The wetdrag subroutine computes the interphase drag for wetted walls. It uses the junction flow regime determined from earlier subroutines and then calls various subroutines to compute the interphase

drag coefficients. It calls bubdrag if the flow regime is bubbly, slugdrag if the flow regime is slug, amistdrag if the flow regime is annular mist, or dispdrag if the flow regime is dispersed or mist. It calls hstratdrag if the junction flow regime is horizontally stratified.

### **3.3.7.6 *drydrag***

The drydrag subroutine computes the interphase drag for dry walls. It uses the junction flow regime determined from earlier subroutines and then calls various subroutines to compute the interphase drag coefficients. It calls the invanndrag if the regime is inverted annular, invslugdrag if the regime is inverted slug, and fidis2 if the regime is dispersed.

### **3.3.7.7 *adjusthif***

The adjusthif subroutine modifies the interphase heat transfer coefficients if the volume is vertically stratified.

### **3.3.7.8 *chng18drag***

The chng18drag subroutine computes the interphase drag for bubbly flow for the special cases where there is an inverted void profile, i.e., voidg decreases with increasing elevation, and for the case where there is a sharp voidg interface between volumes.

### **3.3.7.9 *filterdrag***

The filterdrag subroutines time smooths the interphase drag coefficients.

### **3.3.7.10 *finaldrag***

The finaldrag subroutine makes the final interphase drag changes and computes dpstf that is used in vexplt.

### **3.3.7.11 *bugoutdrag***

The bugoutdrag subroutine gives debug printout of the phantj variables when the help variable is not equal to zero.

## **3.3.8 *fwdrag***

The fwdrag subroutine computes the wall drag terms including flow regimes and the two-phase flow friction HTFS correlation. This subroutine calls viscos1 twice: first to get the liquid viscosity and second to get the vapor viscosity. Both of these calls use the surface temperature for the temperature.

### **3.3.8.1 *viscos1***

The *viscos1* subroutine computes the viscosity at the given input temperature.

### **3.3.9 *hloss***

The *hloss* subroutine calculates void fractions at the throat and downstream of an abrupt area change. Internally, the subroutine only loops over abrupt area junctions that are not time-dependent junctions. The two junction based variables are returned from the call to *hloss* are *formfj* and *formgj*. These are the abrupt area form losses for liquid and vapor flow in the current flow direction.

The next set of five subroutines, *vexplt*, *jchoke*, *ccfl*, *vfinl*, and *eqfinl*, are called if the semi-implicit solution scheme is used. Another set is used for the nearly-implicit solution scheme. These will be covered later.

### **3.3.10 *vexplt***

The *vexplt* subroutine computes the explicit liquid and vapor velocities, the pressure gradient coefficients needed for the implicit pressure solution, and the old-time source terms for the mass and energy equations. The explicit velocities are the junction velocities resulting from using the old-time pressure gradient. These explicit velocities are later refined to give the new-time velocities in the *vfinl* subroutine after the new-time pressure gradients are known. The pressure gradient coefficients are used for this adjustment.

The junction based variables returned from *vexplt* are *fwallfj*, *fwallgj*, *faaj*, *fjunft*, *fjunrt*, *fij*, *velfj*, and *velgj*. The *fwallfj* and *fwallgj* variables are the wall friction terms in the liquid and vapor momentum equations, *faaj* is the added mass term, *fjunft* and *fjunrt* are the forward and reverse loss coefficients, and of course, *velfj* and *velgj* are the liquid and vapor explicit velocities. Scratch variables, which are stored in *scrch* dynamic common block, are also computed in *vexplt*. These variables are needed in later subroutines. The scratch variables include *fal*, *ff*, *gal*, *gg*, *ihld3*, *pk*, *pl*, *pmhig*, *pmpph*, *pshig*, *sathfx*, *sathgx*, *scvjck*, *scvtur*, *sourca*, *sourcf*, *sourcg*, *sourcm*, *sourcn*, *vfdpk*, *vgdpk*, *vfdpl*, and *vfdpl*.

The *vexplt* subroutine calls the *pump*, *accum*, and *turbst* subroutines. The *pump* subroutine computes the pressure change and torque for the pump, the torque for the inductive motor if present, and advances the pump speed if is not connected to a shaft control component. It calls the *pump2* subroutine once to get the single-phase homologous data and again to get the two-phase homologous data. The *accum* subroutine computes the accumulator mass, wall heat transfer, and momentum model parameters. The *turbst* subroutine evaluates the turbine stage performance factors, power, and torque.

#### **3.3.10.1 *pump and pump2***

The *pump* subroutine computes the pump head. It calls the *pump2* subroutine twice. The *pump2* subroutine interpolates the pump homologous curves. the first call is to get the single-phase homologous data and second call to get the two-phase homologous data.

### **3.3.10.2 *accum***

The accum subroutine does the mass, momentum, and energy balances in the accumulator.

### **3.3.10.3 *turst***

The turbst subroutine evaluates the turbine stage performance factors, power, and torque.

## **3.3.11 *jchoke***

The jchoke subroutine computes the choking velocity at junctions using one of three choking models.

1. Original RELAP5 model
2. Henry-Fauske model
3. Moody model

For the Henry-Fauske choking calculations, jchoke calls gesub and gctpm. For the Moody choking model calculations, jchoke calls the gtpmoody subroutine. The gcsup subroutine computes the critical mass flux for subcooled liquid for the Henry-Fauske choked flow model. The gctpm does the same thing for the two-phase flow. The gtpmoody subroutine computes the critical mass flux for the Moody choking model. This later model was added as part of the CANDU updates supplied by the Korean CAMP group.

The jchoke subroutine calls the stcset subroutine to get the proper fluid properties of the donor volume. The stcset subroutine is explained in Section 3.3.1. If the Henry-Fauske choked flow model is used, which is the default choking model, then jchoke calls gcsup or gctpm to get the critical mass flux. If the Moody choked flow model is used, then the gtpmoody subroutine is called to get the critical mass flux. The Moody model does not consider noncondensable gases, so whenever there are noncondensable gases present in the upstream volume, the gctpm subroutine is called even if the Moody model was selected by use of option 54 on Card 1.

### **3.3.11.1 *gcsup***

The gesub subroutine returns the critical mass flux for subcooled liquid using the Henry-Fauske choking model.

### **3.3.11.2 *gctpm***

The gctpm subroutine returns the critical mass flux for two-phase fluids using the Henry-Fauske choking model.

### **3.3.11.3 *gctpmoody***

The gctpmoody subroutine returns the critical mass flux for subcooled liquid using the Moody choking model.

### **3.3.12 *ccfl***

The ccfl subroutine determines if countercurrent flow limiting occurs, and if it does, a flooding correlation of the Wallis-Kudateladze type is used instead of the difference momentum equation

### **3.3.13 *vfinl***

The vfinl subroutine calls the subroutines preseq and syssol to load and solve the pressure matrix, computes the new-time velocities and mass flow rates, and checks for bad donorizing and/or water packing. If bad donorizing occurs, recalculation of the new-time velocities is repeated up to four times or until there is no more bad donorizing. If water packing occurs, the new-time velocities are recomputed once.

Bad donorizing is not determined as one might expect by a velocity reversal. It is based on the total donorized internal energy change during the time step across any junction. If the change is greater than 20% of its value the last time step, or last iteration if there are iterations, then the bad donorizing flag is set true. If bad donorizing occurs, the jprop subroutine (see Section 3.3.18) is called to get new donorized properties based on the latest estimate for the final velocities. The pressure matrix is solved again for the new-time pressure differences, and these pressure differences are used to compute the new-time velocities. Bad donorizing is checked up to four times, and if it occurs, the iteration through the redonorizing, pressure difference solution, and new-time velocity calculation is repeated. If bad donorizing does not occur, the vfinl subroutine continues on to check for water packing by calling the packer subroutine. If water packing occurs, the pressure difference solution is redone, and the new-time velocities are recomputed. There is no further check to see if water packing occurred again. The results of the redone solution are accepted as final.

After all the bad donorizing and water packing logic is finished, the jprop subroutine is called to get new donor properties based on the new-time velocities. Bad donorizing is checked again and if it occurs after water packing, the junctions are redonorized and the pressures are solved for to insure the packed and stretched junctions have the correct donorizing.

#### **3.3.13.1 *preseq***

The preseq subroutine fills the coefficient matrix and row reduces this matrix prior to the call to syssol. The row reduction reduces the block 5x5 matrix that has one block for each volume in the loop that is being solved to a block 1x1 matrix. The block 1x1 matrix is the pressure matrix and is sent to syssol to solve for the increment in pressure in each volume for this time step.

### 3.3.13.2 *syssol*

The syssol subroutine is the sparse matrix solver that is used in RELAP5 to solve the pressure matrix for the increment in pressure in each volume. This subroutine is also used to solve the block 2x2 matrix for the velocities when the nearly implicit scheme is used. It is also used for solving other sparse matrix systems in the nearly implicit scheme.

### 3.3.13.3 *packer*

The packer subroutine determines if water packing occurs, and if it does, modifies terms used in the velocity equations to ameliorate the water packing.

## 3.3.14 *eqfinl*

The eqfinl subroutine computes the new-time pressure (p) using the old-time pressure and the new-time pressure difference that was computed in vfinl. It then does the back substitution to obtain the new-time difference values and adds them to the old-time values for the noncondensable quality (quala), vapor internal energy (ug), liquid internal energy (uf), and vapor void fraction (voidg). It also computes new-time values for the mixture density (rhom), mass generation rate (dotm), vapor generation rate at the wall (gammaw), vapor condensation rate (gammac), and net vapor generation rate (vapgen). The internal energies are then recomputed using a conservative form of the vapor and liquid energy equations as opposed to the nonconservative form of the vapor and liquid energy equations used in the original solution. The noncondensable quality is also recomputed using a conservative form of the noncondensable continuity equation. The vapor void fraction is also recomputed using the equation.

$$\alpha_g = 1 - \frac{(\alpha\rho)_f}{\rho_f + \frac{\partial\rho_f}{\partial P}\Delta P + \frac{\partial\rho_f}{\partial u_f}\Delta u_f} \quad (3.3-1)$$

where  $(\alpha\rho)_f$  comes from the conservative form of the liquid continuity equation.

When a noncondensable first appears in a volume, i.e., qualao = 0 and quala > 0, the code originally would do a partial backup inside hydro and come back through the phantv, phantj, vexplt, etc. subroutines with the volume seeded with a qualao equal to 1.0E-8. This clever, but complicated, technique usually worked, but in some cases large mass errors would occur. It was traced to the fact that the velocity at the junction that fluxed the noncondensable into the volume reversed sign. Now the code was trying to flux a noncondensable out of a volume in which there was no noncondensable.

The fix was to change the code. Now, whenever the code first detects the appearance of a noncondensable in a volume, it accepts the quala and ug solution from vfinl and eqfinl and then adjusts the noncondensable mass fraction by assuming the minimum partial pressure is the triple point pressure and the temperature is the triple point temperature. What happens when a noncondensable first appears is that

the set of five equations that are row reduced in preseq and solved in syssol are not correct because the noncondensable continuity equation was replaced by an equation that essentially sets the change in the noncondensable quality to zero. Consequently, the solution that is obtained is a bad solution. Quite often the noncondensable quality is greater than 1.0, and the vapor internal energy is less than the saturated steam value. By making this one-time adjustment in the noncondensable quality and a corresponding thermodynamically consistent adjustment in the steam internal energy, the code is able to recover and continue on without having to do the complicated partial backup procedure. The new technique is working much better.

Old-time values of hif, hig, and hgf are recomputed in eqfinl whenever there is a time step reduction from over condensation. This is done so that the old-time weighting that is done in phantv will reduce the values of hif, hig, and hgf sufficiently so that the overcondensation can be reduced as the time step is reduced. Otherwise, without any changes in the old-time values, whenever the time step was reduced, the old-time weighting was actually increasing the values of hif, hig, and hgf.

The following set of three subroutines, vimplt, pimplt, and simplt, are used if the nearly implicit solution scheme is used. The jchoke subroutine is also called by both solution schemes. It was described earlier. The jprop subroutine is called by both the semi-implicit and nearly-implicit solution schemes to get the donored junction properties. It is described later.

### **3.3.15 vimplt**

The vimplt subroutine computes the new-time liquid and vapor velocities using the implicitly coupled momentum equations and the old-time source terms for the mass and energy equations. It calls pump (see Section 3.3.10.1), turbst (see Section 3.3.10.3), accum (see Section 3.3.10.2), jprop (see Section 3.3.18), jchoke (see Section 3.3.11), ccfl (see Section 3.3.12), and syssol (see Section 3.3.13.2) with matrix set to 2.

### **3.3.16 pimplt**

The pimplt subroutine carries out the back substitution to obtain the new-time pressure and boron density as well as the intermediate-time liquid internal energy, vapor internal energy, void fraction, and noncondensable quality. It also computes the vapor generation rate and the remainder of the source terms for the mass and energy equations used in the implicit solution scheme.

### **3.3.17 simplt**

The simplt subroutine computes the new time liquid internal energy, vapor internal energy, void fraction, noncondensable quality, and boron density using implicit convective terms in the mass and energy equations. It calls syssol (see Section 3.3.13.2) with matrix set to 1 five times. The first call is to get the mass  $\alpha_f \rho_f$  and  $\alpha_g \rho_g$ , the second time is to get the noncondensable quality  $\alpha_g \rho_g x_n$ , the third call is to get the vapor energy  $\alpha_g \rho_g u_g$ , the fourth call is to get the liquid internal energy  $\alpha_f \rho_f u_f$ , the fifth call is to get the boron density.

### **3.3.18 jprop**

The jprop subroutine computes the junction donor properties from the upstream volume. Quantities that are donated include the densities, void fractions, internal energies, and noncondensable quality. If the level tracking model is activated in the volume, then the junction void fractions are adjusted according to where the level is located in the volume. If the thermal-front tracking model is activated in the volume, the donated internal energies and densities are adjusted according to where the thermal front is located in the volume. The jprop subroutine also calls the appropriate separator subroutine, stdsp, dryer, gesep, stdry, or gedry according to the type of separator in the input model. If there is an accumulator in the input deck then a special donating scheme is used. If the junction is horizontal, then either the hzflow or hseflw subroutine is called to adjust the donating when horizontal stratification occurs. The hseflw subroutine is called when Option 77 on Card 1 is present in the input deck, otherwise hzflow is called. Both subroutines compute the vapor pull-through and liquid entrainment for stratified horizontal flow. They also adjust the donor void fractions for inverted u-tube and normal u-tube geometries. The hseflw, which is called when option 77 is on card 1, introduces some new geometric restrictions on the hse model.

#### **3.3.18.1 stdsp**

The stdsp subroutine donors the separator void fraction using simple vover and vunder input variables.

#### **3.3.18.2 dryer**

The dryer subroutine dryer computes the exit vapor volume fraction of a dryer using the GE dryer model.

#### **3.3.18.3 gesep**

The gesep subroutine computes the needed input to the GE mechanistic separator model and calls the GE subroutines.

#### **3.3.18.4 stdry**

The stdry subroutine is a combined separator/dryer model.

#### **3.3.18.5 gedry**

The gedry subroutine computes the needed input to the GE mechanistic separator model and calls the GE subroutines plus the dryer model.

#### **3.3.18.6 hseflw**

The hseflw subroutine is the vapor pull-through and liquid entrainment model for stratified horizontal flow. It also adjusts the donor void fractions for inverted u-tube geometry and normal u-tube

geometry. This subroutine introduces some new geometric restrictions on the hse model. It is called when Card 1 Option 77 is present in the input deck.

### **3.3.18.7 *hzflow***

The hzflow subroutine is the vapor pull-through and liquid entrainment model for stratified horizontal flow. It also adjusts the donor void fractions for inverted u-tube geometry and normal u-tube geometry.

### **3.3.19 *brntrn***

The brntrn subroutine computes the new-time boron concentration (boron) using the second-order Godunov method that does not smear the boron front as much as the donorizing method does.

### **3.3.20 *state***

The state subroutine evaluates the equation of state for all components. First, it calls stacc for accumulators. Second, it calls statep for all the real volumes, i.e., non time-dependent volumes. Last, it calls masserror to compute the mass error.

### **3.3.21 *stacc***

The stacc subroutine calls a number of the sth2x subroutines or their equivalents for D2O, std2x, or their equivalents for the new water properties, nth2x, to obtain the thermodynamic properties of the fluid in the accumulators.

### **3.3.22 *statep***

The statep subroutine is the driver for computing the thermodynamic properties for all the real volumes. It first calls classifyvols to sort the volumes into normal volumes, volumes with noncondensables present, and volumes with no noncondensable present. The number of each of these types of volumes are called numnormvol, numairvol, and numnoairvol, respectively. Next, the supertosub subroutine is called to handle the normal volumes that are transitioning from a supercritical pressure condition to a subcritical pressure condition. Then the subroutines, withair or its counterpart for the new water properties, nwwithair, is called to get the thermodynamic properties for the volumes that have noncondensables in them. Next, the withoutair subroutine is called to get the thermodynamic properties for the volumes that do not have noncondensables in them. The statefinal subroutine is then called to get the surface tension for all the normal volumes and to check for thermodynamic properties being out of range.

#### **3.3.22.1 *classifyvols***

The classifyvols subroutine classifies the volumes in a system according to normal volumes, air volumes, and no-air volumes. Normal volumes are volumes that are not accumulators or time-dependent volumes. Air volumes are volumes in which quala  $\geq 1.0E-9$  or air is going to appear in the volume in the

next time step. No-air volumes are volumes in which  $\text{quala} < 1.0\text{E-}9$  and air is not going to appear in the volume in the next time step. Air in the previous context means any noncondensable gas. The flag, `isgasappearance6`, is set to true if air is going appear in the volume in the next time step.

### **3.3.22.2 *supertosub***

The supertosub subroutine determines if any normal volume transitioned from supercritical pressure to subcritical pressure in the last time step. This subroutine sets the void fractions to appropriate values in these volumes.

### **3.3.22.3 *withair and nwithair***

The withair and nwithair subroutines compute the thermodynamic properties for volumes with air (noncondensables) present. First, the airprops subroutine is called to get the mixture properties of the noncondensable mixture. Then a check is made to see if the volume contains only noncondensables, and if it does, it calls the idealgas subroutine to compute the thermodynamic properties. If the volume has some water liquid or vapor in it, then a different approach is taken. Two cases can occur, no liquid is present and some liquid is present. If the liquid void fraction is greater than zero, then the liquid thermodynamic properties are looked up in the steam tables. After the liquid thermodynamic properties are determined, an iteration is done to get the vapor thermodynamic properties. An iteration is necessary because we need to solve two simultaneous equations, one equation says the vapor internal energy is a weighted sum of the steam internal energy and the air internal energy. The weighting factor is the noncondensable quality (quala). The other equation says that the volume occupied by the steam is the same volume occupied by the air. These two equations are solved by Newton's method to get final values for the air quality (quala), the steam internal energy (ustm), and the partial pressure of the steam (pps). The nwithair subroutine does the same job as withair, but was rewritten to speed up the solution when the new steam tables are used.

### **3.3.22.4 *withoutair***

The withoutair subroutine gets the thermodynamic properties for volumes that contain no air. It first checks for supercritical pressure volumes and gets the thermodynamic properties for that fluid, which is neither liquid nor vapor. For subcritical pressure volumes, the subroutine first gets the saturation properties at the pressure in the volume. If there is no liquid in the volume, it sets the liquid thermodynamic properties to the saturated liquid properties. If there is liquid in the volume, it gets the thermodynamic properties for the liquid from the steam tables. It then checks to see if there is vapor in the volume. If there is no vapor, it sets the vapor thermodynamic properties to the saturated vapor values. If there is vapor in the volume, it gets the thermodynamic properties for the vapor from the steam tables. After all these computations are finished, it computes the mixture thermodynamic properties and the homogeneous equilibrium sound speeds. It then calls the viscos1 and thcond1 subroutines to compute the liquid and vapor viscosities and thermal conductivities.

### **3.3.23 masserror**

The masserror subroutine computes the mass error for all the volumes.

### **3.3.24 level**

The level subroutine detects a two-phase level in a volume. Once it detects a level, it computes the level propagation velocity, the above-level void fraction, the below-level void fraction, and the level position relative to the bottom of the volume. The level position is determined by either an examination of the level criteria (icheck = 0 case) or by extrapolating the level position from the earlier time step (icheck = 1 case) using the old-time level position and the old-time level velocity. After the level position has been determined, the pressure gradient correction term is computed for the adjacent junctions. The velocities in these junctions are then recomputed to be consistent with the level position. The level subroutine calls jprop, phantj, and vlevela.

### **3.3.25 vlevela**

The vlevela subroutine computes the two volume-averaged velocities (velf and velg). These volume-averaged velocities are used as one of the velocities in the momentum flux term in the momentum equation. The other velocity is the donored junction velocity that was computed earlier in the volvel subroutine. There are three possible volume-averaged velocities for each volume, one for each coordinate direction. If the coordinate direction is not being used, i.e., there are no cross flow junctions attached to the volume in that coordinate direction, then the volume-averaged velocities are set to zero. If there is only a junction on one side of a volume such as might occur when the volume is dead-ended, the volume-averaged velocity is set equal to the junction velocity. It might be argued that it should be one-half of the junction velocity, but experience has shown that this is not correct.

### **3.3.26 htfinl**

The htfinl subroutine advances the heat structures by adding the effects of the changed fluid temperatures. It uses the projected increments in the liquid, vapor, saturation temperatures to correct the surface temperatures at convecting boundaries.

## **3.4 Heat Structure Subroutines**

### **3.4.1 htadv**

The htadv subroutine is the driver for the heat conduction solution. It calls the radht subroutine if there are radiation structures and the qfmove subroutine if reflood is turned on. It then calls the ht1tdp subroutine to do the one-dimensional heat conduction solution.

A flow chart for htadv showing the calls is shown below. Filid(38) is the filid for the radiation heat transfer database. This filid is 0.0 if there are no radiation heat structures. Filid(32) is the filid for the reflood heat transfer database. This filid is 0.0 if reflood is not activated.

```

        subroutine htadv
a<<<<<<< if (aflag) then
a b<<<<<< if (filid(38) .ne. 0.0d0) then
a b           if (succes .eq. 0) call radht - radiation heat transfer
a b>>>>>>> endif
a           if (filid(32) .ne. 0.0d0) call qfmove - reflood calculations
a c<<<<<<< do 10 m = 1,nhtstr(2,i)
a c d<<<<<<< if (.not.isstdystateflg0(2,j)) then
a c d           call ht1tdp (i,timeht,dht) - one-dimensional transient heat conduction
a c d>>>>>>> endif
a c>>>> 10      continue
a>>>>>>> endif
               return

```

### 3.4.2 radht

The radht subroutine does the radiation heat transfer calculations.

### 3.4.3 qfmove

The qfmove subroutine does the reflood calculations.

### 3.4.4 ht1tdp

The ht1tdp subroutine does the one-dimensional heat conduction solution. It calls the madata subroutine to get the material properties. It also calls the kloss subroutine to get new loss coefficients for the junctions above and below a volume in which a fuel rod has ruptured. It then calls the htcond subroutine to get the left and right boundary conditions for the heat structure and the qmwr subroutine if there has been any metal-water reaction on the outside of the cladding, and if the fuel rod has ruptured, the inside of the cladding.

```

        subroutine ht1tdp (ih,time,dtime) - one-dimensional transient heat conduction
a<<<<<<< if (succes .eq. 0) then
a           call ftbmov (htttmp(temidx),htttmp(initem),cols)
a>>>>>>> endif
b<<<<<<< if (chngno(88) .and. i .eq. 20000) then
b           continue
b<<<<<<< else
b c<<<<<<< if (i .eq. 0) then
b c           continue
b c<<<<<<< else
b c d<<<<<<<< if (gttrp(2,i) .ne. 0) then
b c d e<<<<<<<< if (tempn .eq. gtarg(i)) then
b c d e           continue
b c d e<<<<<<< else
b c d e           call polat (gtbl(i-3),gtbl(i),tempn,htpown(hindex),err)
b c d e>>>>>>> endif
b c d>>>>>>> endif
b c>>>>>>> endif
b>>>>>>> endif
               call madata (initem,index,idxo,cols,probid,
&                   .false.,errsw,hindex,dtime,dtdt,
&                   cltave,block,irup,iplas) - material properties
f<<<<<<< if (iplas .ne. 0) then
f g<<<<<<< if (.not.isrupture9(2,idxtop)) then
f g h<<<<<<< if (block .gt. 0.0d0) then

```

```

f g h           if (chnglosscoeff12(2,idxtop) .and.
f g h   &      .not.islosscoeffchng10(2,idxtop))
f g h   &      call kloss (block, hindex) - loss coefficients for blocked channels
f g h>>>>>>>> endif
f g>>>>>>> endif
f>>>>>>> endif
          call htcond (hindex,index,initem,initem+nn,
          &                  time,dtime,probid) - left & right BC
i<<<<<<<< if (nn .ge. 2) then
i j<<<<<<<< if (ismwrsetinput14(2,hindex)) then
i j          call qmwr (hindex,qdi,qdo,dtime) - metal-water reaction on cladding
i j>>>>>>> endif
i>>>>>>> endif
          return
end

```

### 3.4.5 htcond

The htcond subroutine gets the left and right boundary conditions for a heat structure.

## 3.5 Reactor Kinetics Subroutines

### 3.5.1 rkin

The rkin subroutine, which is called from the tran subroutine, advances the reactor kinetics calculation for one time step. The reading of the reactor kinetics input cards is done by the rrkin subroutine and the linking of the reactor kinetics with the rest of the code is done by the irkin subroutine.

### 3.5.2 rrkin

The rrkin subroutine reads the 30000000-2, 30000101-199, 30000301-399, and 30000401-499 reactor kinetics input cards. It calls the rrkinp subroutine to read the 3000011-20, 30000501-599, 30000601-699, 30000701-799, 30000801-899, 30001701-1799, 30001801-1899, 300019C1-19C9, 30002001-2999 reactor kinetics feedback input cards. The reactor kinetics database is in the rkinc dynamic common block. The reactor kinetics table for interpolation for the ANS79 fission product decay is in the normal common block /rkntbl/.

### 3.5.3 irkin

The irkin subroutine links the reactor kinetics variables with variables in the rest of the code. It checks the scram table reference and sets some pointers. It checks the volume references and sets some pointers. It checks the heat structure references and sets some pointers. It computes the bias reactivity. It does the scram curves. It checks to see if there is separable or table feedback. It does the heat structure feedback or table feedback depending on which one is active. If table feedback is active, it does the volume averaging and heat structure averaging. It evaluates the reactivity and sets up for the first time step. For the Doppler feedback, it sets the offsets to the heat structures so that the code will run faster during the transient.

## 3.6 Control System Subroutines

### 3.6.1 convar

The convar subroutine, which is called from the tran subroutine, does the control system calculation for one time step. The reading of the control system input cards is done in the rconvr subroutine and the linking of the control system with the rest of the code is done in the iconvr subroutine.

### 3.6.2 rconvar

The rconvar subroutine reads the 20500000, 205CCC00 or 205CCCC0, 205CCC01-98 or 205CCCC1-8 control system input cards.

### 3.6.3 iconvr

The iconvr subroutine checks for legal variable requests. It sets the initial conditions for all the control components and edits out the initial conditions for all the components.

## 3.7 Tricks of the Trade

This section contains some the "tricks of the trade" as used in RELAP5. It shows snippets of coding that is used throughout the code that are useful to the RELAP5 programmer. But before we get to the coding snippets, we need to provide some background on the way the dynamic common blocks are organized in RELAP5. We will cover the voldat, jundat, and lpdat common blocks. The other dynamic common blocks are organized in a similar manner.

### 3.7.1 voldat

The volume variables, e.g., pressure and saturation temperature, live in the voldat dynamic common block. There are two possible ways to arrange this two-dimensional group of variables. The first way is to have all the pressures for all the volumes in contiguous memory followed by all the saturation temperatures for all the volumes in contiguous memory, etc. The second way is to have all the volume variables for the first volume in contiguous memory followed by all the volume variables for the second volume in contiguous memory, etc. RELAP5 organizes its dynamic common blocks using the second method. This method requires the use of skip factor when you are looping over all the volumes in a do loop. The volume skip factor variable is called ivskp. The first variable in the voldat dynamic common block is the number of volumes, nvols.

### 3.7.2 jundat

The junction variables, e.g., liquid velocity and vapor velocity, live in the jundat dynamic common block. It too is arranged so that all the junction variables for each junction are in contiguous memory. The junction skip factor variable is called ijskp. The first variable in the jundat dynamic common block is the number of junctions, njuns.

### 3.7.3 lpdat

The loop or system variables, e.g., number of volumes in a system and pointers to the first volume in a system, live in the lpdat dynamic common block. It too is arranged so that all the system variables for each system are in contiguous memory. The loop skip factor variable is called lpskp. The first variable in the lpdat common block is the number of systems, nloops.

A system or loop is a collection of volumes that are interconnected with junctions. Some of the junctions can be valves which are closed. So, while it may appear that there are two separate systems separated by a closed valve, RELAP5 regards this model a one system because the valve could be opened at some future time and the fluids could mix. An example of a system or loop is the primary system in a plant model. The secondary system could be a second system in this plant model as long as it is unconnected hydrodynamically to another system.

### 3.7.4 Loop over Systems

The following code snippet loops over systems in the model. It first sets the system pointer variable issys to the start of the lpdat dynamic common block, filndx(30). It then sets up a do 10 loop to loop over all the systems in the model. The number of systems is given by the variable nloops(issys). At the end of the do 10 loop, it increments the system pointer variable issys by the skip factor, lpskp, so that issys is now pointing to the next set of system variables in the lpdat dynamic common block.

```
issys = filndx(30)
do 10 is = 1, nloops(issys)
c do something for this loop using issys for the system index
    issys = issys + lpskp
10 continue
```

### 3.7.5 Loop over Volumes

The following code snippet loops over all the volumes in a model that has multiple systems. It uses the snippet in Section 3.7.4 for its outer-most loop over the systems in the model. After the system pointer variable issys is determined, it sets the volume pointer variable iv to the first volume in the system, liv(issys). It then sets up a do 5 loop to loop over all the volumes in this system. The number of volumes in this system is given by the variable livn(issys). At the end of the do 5 loop, it increments the volume pointer variable iv by the skip factor, ivskp, so that iv is now pointing to the next set of volume variables in the voldat dynamic common block.

```
issys = filndx(30)
do 10 is = 1, nloops(issys)
    iv = liv(issys)
    do 5 iv = 1, livn(issys)
c do something for this volume using iv for the volume index
    iv = iv + ivskp
5 continue
    issys = issys + lpskp
10 continue
```

### 3.7.6 Loop over Junctions

The following code snippet loops over all the junctions in a model that has multiple systems. It uses the snippet in Section 3.7.4 for its outer-most loop over the systems in the model. After the system pointer variable issys is determined, it sets the junctions pointer variable jj to the first junction in the system, lijn(issys). It then sets up a do 5 loop to loop over all the junctions in this system. The number of junctions in the system is given by the variable lijn(issys). At the end of the do 5 loop, it increments the junction pointer variable jj by the skip factor, ijskp, so that jj is now pointing to the next set of junction variables in the jundat dynamic common block.

```

issys = filndx(30)
do 10 is = 1, nloops(issys)
  iv = lij(issys)
  do 5 jj = 1, lijn(issys)
    c  do something for this junction using jj for the junction index
      jj = jj + ijskp
    5   continue
    issys = issys + lpskp
  10  continue

```

### 3.7.7 Loop over Volume Scratch Variables

The following code snippet loops over all the volumes in a model that has multiple systems. It uses the snippet in Section 3.7.4 for its outer-most loop over the systems in the model. After the system pointer variable issys is determined, it sets the volume pointer variable iv to the first volume in the system, liv(issys). It then sets up a do 5 loop to loop over all the volumes in this system. The number of volumes in this system is given by the variable livn(issys). At the end of the do 5 loop, it increments the volume pointer variable iv by the skip factor, ivskp, so that iv is now pointing to the next set of volume variables in the voldat dynamic common block.

```

issys = filndx(30)
do 10 is = 1, nloops(issys)
  iv = liv(issys)
  do 5 iv = 1, livn(issys)
    c  do something for this volume using iv for the volume index
      iv = iv + ivskp
    5   continue
    issys = issys + lpskp
  10  continue

```

To get the index for the first volume, ixvff, and the index for the first junction, ixjff, in the scrch common block use the following

```

issys = filndx(30)
ixvff = vctrls(liv(issys))
ixjff = jcnnxs(ljj(issys))

```

### 3.7.8 Adding New Integers or Logicals to Dynamic Comdecks

Whenever an integer or logical variable is either added or removed from the comdecks that are not really common blocks but are sets of specification and equivalence statements between the variable names

and the fa array in the /fast/ common block, remember to change the mlist file. The mlist file contains all the integer and logical (4-byte words) words that are equivalenced to the fa array (actually to the ia array). The cnv32 preprocessor program uses this list to determine which variables should have a “(1,” or “(2,” added to the variable’s index. If the variable is not included in mlist, then the preprocessor will not append the “(1,” or “(2,” and the code could access the wrong half of the eight byte word. This may be OK when the “(1,” is used, but it is not OK when the “(2,” is used. The cnv32 program assumes that mlist is sorted alphabetically, so remember to insert the new integer or logical variable in its proper location.

### **3.7.9 Adding a New Default Plot Variable to the Default List**

When you need to add a new plot variable to the default list, i.e., the list of variables that are underlined in Section A4 Minor Edits, 301-399, you have to make changes to three subroutines: wrplid.ff, pltwrt.ff, and rusrvr.ff. The change to wrplid.ff involves adding the new variable to the set of labels that are written at the beginning of the plot file. For example, when the junction variable, chokef, was made one of the default junction variables, coding had to be added to read the label, chokef<sup>f</sup>, into buf(j) from the t3 array that is stored in the vreqd.hh comdeck. Since chokef is in slot 22, the read statement was

```
read (t3(22),'(a8)') buf(j)
```

This statement was followed by the storage of the junction number, junno(i) in ibuf(k). This step takes care of getting the plot label written to the plot file. Next, the values themselves have to be written out by making changes in the pltrec.ff subroutine. This was done by adding the statement

```
fa(j+10) = chokef(i)
```

to the list of other junction variables that are written out to the plot file whenever a plot record is written. The third step was to add chokef to the list of default variables that are given in the rusrvr.ff subroutine.

## **3.8 Output Files**

### **3.8.1 Major Edits**

The major edit output frequency is determined by the time-step control cards, 201-299. The major edit contains information about all the volumes, junctions, heat structures, reactor kinetics, control systems, etc. at a specific time in the transient. One subroutine, majout, is responsible for generating the major edit. It is called by dtstep.

### **3.8.2 Minor Edits**

The minor edit output frequency is determined by the time-step control cards, 201-199. The minor edit contains information about all the variables requested on the 301-399 cards. Only one variable may be requested per card, so there is a limit of 99 minor edit variables. One subroutine, mirec, is responsible for generating the minor edits. It is called by dtstep. The minor edits are accumulated in an internal buffer in the /fast/ common block and are dumped to the output file whenever 50 sets have been accumulated.

### 3.8.3 Strip Edits

The strip edit output file variable list is determined by the strip request cards, 1001-1999. Only one variable can be entered per card, so there is a limit of 999 strip edit variables. One subroutine, strip, is responsible for generating the strip edits. It is called by relap5. The strip edits are taken from the plot file that is written to the restart-plot file during the transient. The frequency of the strip file edits is therefore determined by the plot file frequency edits. This frequency is determined by what the user enters on the time-step control cards, 201-299.

### 3.8.4 Screen Edits

The screen file shows the user the current status of RELAP5 as it is running. Displayed on the screen are the CPU time, the problem time, some variables associated with the Courant limit, some variables associated with the mass error, and the reason the time step is either decreasing, staying the same, or increasing. The variables associated with the Courant limit are the Courant time step, the volume in which this limit occurs, the pressure in that volume and the void fraction in that volume. The variables associated with the mass error are the mass error, the volume in which the largest mass error occurs, the pressure in this volume, and the void coefficient in this volume. The ncount is also put out. The reason column contains the reason that RELAP5 uses to adjust the time step size. The ideal reason is the one called Courant. This means the code is running at the Courant limit, which is full out when using the semi-implicit method. Other reason that a user might see is dtmax, which means the code is running at the maximum time step that the user entered on the time step card. During the early part of the transient, the user will see the dt +10% reason, which means that the time step was increased by 10%. Since RELAP5 starts out the transient at the minimum time step size, you will see this early in the transient.

### 3.8.5 Snapit Edits

These are generated by calling the snapit subroutine. An example of this call is in the hydro.ff file. The output is a formatted printout of some of the more important the dynamic common blocks. The voldat, jundat, and lpdat are some of the ones that can be printed out. The file name of the snapit output file is specified in the snapit call statement.

### 3.8.6 Debug Edits

Debug edits are obtained by use of the 105 card in the input deck.

### 3.8.7 Binary Restart-Plot Files

The rstplt file is generated whenever a restart file is requested. This file also contains the plot variables. that can be plotted using xmgr5.

## 4 CODE DATABASE

The RELAP5 code uses two types of common blocks for its database; normal FORTRAN labeled common blocks and dynamic common blocks. The dynamic common blocks are actually blocks of storage within the normal FORTRAN labeled common block called /fast/, which is defined in fast.hh comdeck file.

All the common block files have the comment card, \*comdeck name, as their first card. These comment cards are used by the usplit program as a delimiter when it is dividing up the relap.s, envrl.s, etc. files to create the individual FORTRAN source files. If the identifier is \*comdeck name, then a name.hh file is created.

### 4.1 Labeled Common Blocks

These are the normal FORTRAN labeled common blocks and have define or more labeled common blocks.

#### 4.1.1 Main control database - contrl.hh - /contrl/ and /rstsum/

The contrl.hh file contains the /contrl/ and the /rstsum/ common blocks. The /contrl/ common block contains the main control information that is used by RELAP5. For example, the values for the current problem time for the thermal hydraulics (timehy), the current time step size (dt), the current cycle count (count), the total mass in all the loops (tmass), the total mass error in all the loops (emass), etc., are stored in this common block. The /rstsum/ common block contains all the restart information that is used for printing the 103 cards at the end of a run. The values of the restart number (irstno) and corresponding problem time (rtime), which are conveniently written to the output file for the user at the end of a transient, are also stored in the contrl.hh file in the rstsum common block. A listing of the contrl.hh file follows.

```
*comdeck contrl
    common /contrl/ dthy,dtht,dttn,dt,timehy,timeht,errmax,tmass ,
    & tmasso,emass,emasso,count,curtmi,curtmj,curtrs,prevnd,
    & cpurem(5),stdtrn,gravcn,testda(20),
    & aflag,isrestartenabled0,ismajoreditenabled1,
    & isminoreditenabled2,isplotenabled3,iscompleterestart4,
    & interactiveflag5,isimplicithttrnsfr6,istwostepflag7,
    & stdystatetermflg8,integrationflag9,isathenaopt10,
    & isscdapopt11,isplotsqzflg12,rwtrstpltcmprssflag13,
    & transrotatflag14,rstblknumber1531,succes,done,ncount,
    & nstsp,nrepet,help,nany,skipt,problemtype05,
    & problemopt611,ncase,fail,uniti,unito,chngno(90),
    & ishydrochng0,isheatcondchng1,isrkncnchng2,ispltrcrdchng3,
    & isradiationchng4,isfissionprdctchng5,iscntrlsyschng6,
    & pageno,isprntaccum0,isprntbrntrn1,
    & isprntccfl2,isprntchfc13,isprntconden4,isprntdittus5,
    & isprnteqfin16,isprntfwdrag7,isprntht2tdp9,isprnthtf112,
    & isprnhtrc113,isprnthydro15,isprntistate17,isprntjchoke18,
    & isprntjprop19,isprntnoncnd20,isprntphantj21,isprntphantv22,
    & isprntpimplt23,isprntpintfc24,isprntprednb25,isprntpreseq26,
    & isprntpstdnb27,isprntqfmove28,isprntsimplt29,isprntstacc0,
```

## Labeled Common Blocks

```

& isprntstate1,isprntstatep2,isprntsuboil3,isprntsitsit4,
& isprnttstate6,isprntvalve7,isprntvexpl8,isprntvfinl9,
& isprntvimplt10,isprntvvela11,isprntvvel12,isprnttrip13,
& isprntrpower14,isprntvolume15,isprntjunction16,isprntheatstr17,
& isprntradht18,isprntreflood19,isprntfsnprdtr20,isprntcontrol21,
& isprntinput22,isprntmiedit23,islevelmodel0,
& islevelcrossing1,issnapon,
& rktpow3d,safe2

c
      real dthy,dtht,dt,dt,tmass,tmasso,emass,
& emasso,count,curtmi,curtmj,curtrs,prevnd,cpurem,stdtrn,gravcn,
& testda,rktpow3d,safe2
      integer succes,done,ncount,nstsp,nrepet,help,nany,
& ncase,pageno,cpurei(5)
      logical aflag,skipt,fail,uniti,unito,chngno,nmechk,issnapon
      logical ishydrochng0      , isheatcondchng1      ,
&      isrkncnchng2      , ispltrcrdcnchng3      ,
&      isradiationchng4      , isfissionprdctchng5      ,
&      iscntrlsyschng6      ,
      logical islevelmodel0      , islevelcrossing1
      integer problemtyp05      , problemopt611
      logical isrestartenabled0      , ismajoreditenabled1      ,
&      isminoreditenabled2      , isplotenabled3      ,
&      iscompleterestart4      , interactiveflag5      ,
&      isimplicithttrnsfr6      , istwostepflag7      ,
&      stdystatetermflg8      , integrationflag9      ,
&      isathenaopt10      , isscdapopt11      ,
&      isplotsqzflg12      , rwtrstpltcmprssflag13,
&      transrotatflag14
      logical isprntaccum0      , isprntbrntrn1      ,
&      isprntccfl2      , isprntchfc13      ,
&      isprntconden4      , isprntdittus5      ,
&      isprnteqfin16      , isprntfwdrag7      ,
&      isprnht2tdp9      , isprnhtfin12,
&      isprnhtrc113      , isprnthydro15,
&      isprntistate17      , isprnjchoke18,
&      isprnjprop19
      logical isprntnoncnd20, isprntphantj21,
&      isprntphantv22, isprntimplt23,
&      isprntpintfc24, isprntprednb25,
&      isprntpreseq26, isprntpstdnb27,
&      isprntqfmove28, isprntsimplt29
      logical isprntstacc0      , isprntstate1      ,
&      isprntstatep2      , isprntsuboil3      ,
&      isprntsitsit4      , isprnttstate6      ,
&      isprntvalve7      , isprntvexpl8      ,
&      isprntvfinl9      , isprntvimplt10,
&      isprntvvela11      , isprntvvel12,
&      isprnttrip13      , isprntrpower14      ,
&      isprntvolume15
      logical isprntjunction16, isprntheatstr17,
&      isprntradht18      , isprntreflood19,
&      isprntfsnprdtr20, isprntcontrol21,
&      isprntinput22      , isprntmiedit23
      integer rstblknumber1531
      equivalence (safe2,nmechk), (cpurem(1),cpurei(1))
      integer mxrst
      parameter (mxrst=500)
      integer nrstblk, irstno(mxrst), iblkno(mxrst)

```

```

real rtime(mxrst)
common /rstsum/ rtime, nrstblk, irstno, iblkno

```

#### 4.1.2 Dynamic storage block assignments - comctl.hh - /comctl/

The comctl.hh file contains the /comctl/ common block. It contains common control information for the dynamic common blocks that are stored in the /fast/ common block. A set of subroutines, the FTB package, are used in RELAP5 to manipulate the storage within the /fast/ common block in Small Core Memory (SCM), in Large Core Memory (LCM), on one or more disks up to a maximum of 4 disks, or on a drum. RELAP5 only uses SCM, which is normal memory on all computers RELAP5 is currently supported on.

```

*comdeck comctl
integer ncoms,nfiles
parameter (ncoms=104, nfiles=50)
common /comctl/ comdat(ncoms),comdln(ncoms),filid(nfiles),
& filsiz(nfiles),filndx(0:nfiles),
& witedynamicfile0(ncoms+1),
& writecommonblk2(ncoms+1),
& is4(ncoms+1),
& safel
real filid,safel
integer comdat,comdln,filsiz,filndx
logical newrst
equivalence (safel,newrst)
logical witedynamicfile0,
& writecommonblk2 ,
& is4

```

The maximum number of dynamic storage common blocks is given in the parameter statement for nfiles as 50. For each of the dynamic common blocks there are three variables: filid, which is like a laundry ticket that can be used to identify the block later, filsiz, which is the length of the dynamic common block, and filndx, which is the starting location in the fa array for the block. **Table 4.1-1** shows the association between each dynamic common block and the number used in the filndx, filsiz, filid, etc. arrays.

**Table 4.1-1** Association of a Dynamic Common Block and its Number in the filndx Array

Number	Description
1	Input data and work scratch space during advancement
2	Time step control block
3	Component description block
4	Hydrodynamic volumes block
5	Hydrodynamic junctions block
6	Thermodynamic property file
7	Interactive input and output variable storage

**Table 4.1-1** Association of a Dynamic Common Block and its Number in the filndx Array

<b>Number</b>	<b>Description</b>
8	Heat structure geometry and temperature block
9	Heat structure material property storage
10	Table of inlet and outlet junctions for each volume
11	General table storage
12	Minor edit file
13	Time dependent volumes and junctions pointers
14	Table of heat structures and data for each volume
15	Plot heading and control information (obsolete)
16	Minor edit control, save area, and labels
17	Plot record buffer (obsolete)
18	Trip block
19	Internal plot file control information
20	File for statistics during advancement
21	Reactor kinetics data
22	2-D plot requests and specifications (obsolete)
23	Plot comparison data tables (obsolete)
24	Steady-state block for statistics counter
25	Volume data needed for a moving system (obsolete)
26	Plot data for the internal plot subroutines (obsolete)
27	Control system block
28	Component indices in normal (input) order
29	Tabular data for rotations and translations for moving problem (obsolete)
30	Hydrodynamic system control information
31	Code coupling data
32	Reflood rezoning model storage space
33	User supplied plot record variable requests (obsolete)
34	Fission product data
35	Fixed list vectors

**Table 4.1-1** Association of a Dynamic Common Block and its Number in the filndx Array

Number	Description
36	SCDAP data
37	Steady-state initialization check file
38	File for radiation heat transfer
39	Not used
40	File for sparse matrix strategy
41	Storage for print control cards on cards 2-5 (obsolete)
42	Not used
43	File for level model geometry description, i.e. stacks
44-50	Not used

The other parameter, ncoms, in the /comctl/ common block is the maximum number of normal FORTRAN labeled common blocks that may or may not be written to the restart file. For each of these blocks there are five variables: comdat, which is the name of the common block, comdln, which is the length of the common block, writedynamicfile0, which is a logical flag that is true if the dynamic common block on disk is to be written at a complete restart, writecommonblck2, which is another logical flag that is true if the normal common block is to be written at a complete restart file, is4, which is ??. The variable safe1 is the last variable in this common block and is used to determine the length of the /comctl/ common block so that it can be written to the restart file using a "block" write using the first variable, comdat(1) and a length instead of writing out each individual variable. The locf function is used to get the memory locations for comdat(1) and safe1. The difference of these locations plus 1 is the length.

#### 4.1.3 Dynamic storage pool - fast.hh - /fast/

The dynamic common blocks reside in the fa array which is in the /fast/ common block. Originally, on the large mainframe computers, /fast/ was in blank common, which was loaded at the end of the program. This allowed the program to give back to the operating system any excess memory, i.e., the upper part of the fa array that was not needed for the problem currently being run. This allowed the user to lower the cost of the run because on the large mainframe computers, the cost of the run included a factor for the amount of memory used. Two developments in the computer field have made this procedure of giving back memory un-necessary. One development is the inclusion of virtual memory in modern operating systems that automatically pages out parts of memory that are not being used to disk. Thus, many of the features of the FTB package that was developed by Bettis Atomic Power Laboratories to handle these dynamic common blocks are now handled by the operating system (some more efficiently than others of course). Another development has been the dramatic drop in the price of memory. The cost of a chuck of memory to hold the fa array, about 18 Mbytes (2.2 million 8-byte words), is now only few dollars.

\*comdeck fast

```

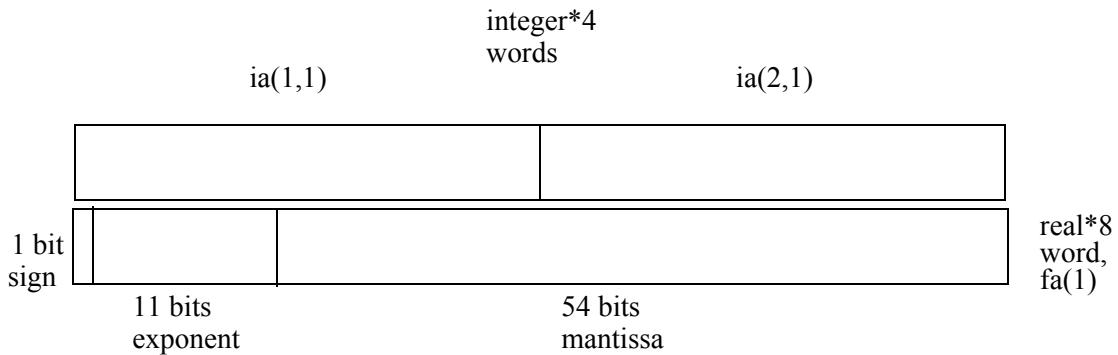
integer lfsiz
parameter (lfsiz=2200000)
common /fast/ fa(lfsiz)
real*8 fa
integer ia(2,lfsiz)
equivalence (fa(1),ia(1,1))

```

#### 4.1.4 Equivalenced fa and ia arrays in /fast/

The 8-byte fa array is equivalenced to a 4-byte integer ia array so that integers can also be stored in the /fast/ common block. In order to be able to use the same subscript for the real\*8 and integer\*4 words, the integer array is doubly dimensioned. Thus, two integer words overlay one real\*8 word. Since FORTRAN stores doubly dimensioned array column wise, the two 4-byte integer words with the same second subscript, ia(1,1) and ia(2,1), occupy the same memory as does the 8-byte real word, fa(1).

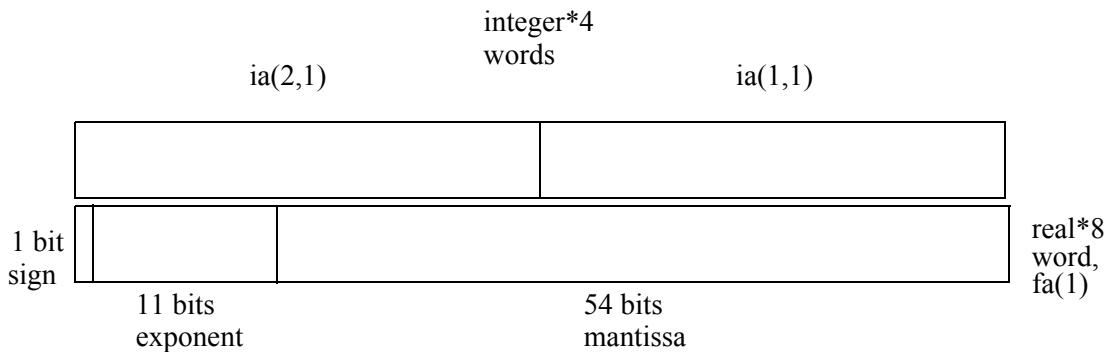
The original computers on which RELAP5 was developed were big Endian computers (IBM, CDC, CRAY, HP, SUN) in that the sign bit and exponent bits occupied the same memory location as did the 4-byte ia(1,1) word. **Figure 41-1** shows the arrangement. Because the RELAP5 developers did not want to disturb the sign and exponent bits of the equivalenced 8-byte real word, they decided to only use the second 4-byte integer, ia(2,1), for storing any logical or integer words that needed to be stored in the /fast/ common block. The concern was that a negative integer has its sign bit and many of the bits adjacent to the sign bit set to ones, which is a result of using the one's or two's complement for storing negative numbers. Thus, any negative number would be storing bits in the exponent part of the equivalenced 8-byte real word and this may cause problems when these words were processed through the floating point processing unit.



**Figure 41-1** Arrangement of Words for a Big Endian Computer

On little Endian computers, like DEC and INTEL, the storage for the two integers is reversed in that the sign and exponent bits of the 8-byte real fa word now reside in the ia(2,1) word. **Figure 41-2** shows the arrangement. The RELAP5 code was not changed to run on these computers primarily because the code ran successfully without any changes. However, it was noticed recently that some RELAP5 restart problems were not giving the correct answers in that trips failed to trip and control variables were incorrect after a restart. In the control variable case, a control variable was set to the problem time and it was correct for the initial run, but when the problem was restarted, the control variable was not equal to the problem

time. The error was traced to the fact that the offset for the control variable was a large negative value. It was the offset from the start of the fa array, fa(1) in the /fast/ common block, to the time variable, timehy, which lives in a /ctrl/ common block. This offset had been changed from its value prior to the restart to a different value after the restart file was read in and the transient started. The error was eventually traced to fact that on an INTEL computer whenever a signaling NaN is loaded into the floating point processing unit, the number is changed to a quiet NaN. A NaN has the sign and all 11 of the exponent bits set to ones and one or more bits of the remaining 56 bits in the mantissa set to a one. The difference between a signaling NaN and a quiet NaN is in bit 19 in the 32-bit word that holds the sign, exponent, and more significant bits of the mantissa. This is the bit that is the first bit (bit 19 counting right-to-left from bit 0) in the mantissa; it is adjacent to the right-most exponent bit. A signaling NaN has bit 19 set to zero and a quiet NaN has bit 19 set to one. Now you can see what can happen by storing a negative integer in the same location as the sign and exponent parts of the 8-byte real word. We could accidentally have a negative integer number that has bit 19 set to zero, and when that number gets loaded into the floating point unit, bit 19 would be set to one. This would change the magnitude of the integer. The floating point load could be as simple as moving the dynamic common block to a different location in memory.



**Figure 41-2** Arrangement of Words for a Little Endian Computer

As of RELAP5 version 3.3aj, when the code is built on INTEL computers, the word that holds the integer is ia(1,1) instead of ia(2,1). This change solved the restart problem with the control variable. It also solved the trip problem for the same reason because the trip that was missed was a time trip, hence, it too was a large negative offset. Since small negative integers will have bit 19 set, and hence have no problem, it appears that the error will only affect large problems in which the offset to time is large.

#### 4.1.5 Bit numbering convention in a packed word

While this is of lesser interest in the code since the bit-packed words have been removed from the main RELAP5 code, there are still places in the environmental subroutines where packed words still exist. The bit numbers start at the right-most bit at 0 and continue to the left up to 31 for an integer\*4 word and up to 63 for a real\*8 word. As shown in **Figure 41-1** for a real\*8 word, the sign is in bit position 63, the exponent is in bits 52-62, and the mantissa is in bits 0-51. For an integer\*4 word, the sign bit is in position 31 and the number is in bits 0-30. Negative numbers are stored as one or two's complement, which

essentially flips all the bits of an equivalent positive number. This means that small negative numbers are essentially all one bits, including the sign bit.

Special bit patterns are used in the IEEE standard to represent the special numbers like  $\pm\infty$  and an indeterminate number like  $0/0$ , which is not a number or NaN. All of these bit patterns have all the exponent bits set to 1s. The sign bit determines if the quantity is + or -. Infinity has all 0 bits in the mantissa and NaN has at least one of the mantissa bits set to 1.

#### 4.1.6 Volume database - voldat.hh

The database for each volume in the problem is stored in a set of variables that are defined in the voldat comdeck. The first few lines of this file are shown below.

```
*comdeck voldat
    integer
& nvols(2,1) , savebtflag(2,1), volno(2,1) ,
& blhimap(1) , volmat(2,1)
c
    real*8
& v(1)           , recipv(1)           , avol(1)   ,
& dl(1)          , diamv(1)
c
    equivalence
& (nvols(1,1), ia(1,1)), (savebtflag(1,1),ia(1,2)),
& (volmat(1,1),ia(1,3)), (volno(1,1),      ia(1,4)),
& (blhimap(1), ia(1,5)), (v(1),            fa(8)),
& (recipv(1),   fa(9)),  (avol(1),         fa(10)),
& (dl(1),       fa(13)), (diamv(1),        fa(16))
```

The first integer variable is the number of volumes in the problem, nvols. The second and fourth integer variables are placeholders that are a result of pulling out the bit-packed variables, btflag and imap. There were actually three slots for imap, one for each direction. The bits that were originally packed into these two words are now defined later in individual words later in the voldat block. The third integer variable, volno, is the number of the volume and the fifth integer variable, volmat, is the material number for the fluid that is in the volume. The first real variable is the volume of the volume. The second real variable is one over the volume. The third real variable is the flow area of the volume. Actually, there are three flow areas, one for each of the three directions. These three values are referenced in the code as avol(i), avol(i+1), and avol(i+2). They do not have separate variable names. The fourth real variable is the length of the volume, dl. This variable also takes up three slots, one for each of the three directions. The fifth real variable is the diameter of the volume, diamv, and this also takes three slots. **Table 4.1-2** shows the arrangement of storage. The actual location in the fa array where the voldat storage block is located is

determined at runtime and is based on the problem. All the indices, offsets, etc., are determined in the code using subroutines from the FTB package. The starting index for the voldat block is kept in filndx(4).

**Table 4.1-2** Voldat Comdeck Variable Arrangement

Type	fa Slot	Variable	Description
integer	1	nvols	Number of volumes
integer	2	savebtflag	Placeholder for btflag bit-packed word, which was removed
integer	3	volmat	Material number of fluid in this volume
integer	4	volno	Volume number
integer	5	blhimap(1)	Placeholder for imap(1) bit-packed word, which was removed (direction 1)
integer	6	blhimap(2)	Placeholder for imap(2) bit-packed word, which was removed (direction 2)
integer	7	blhimap(3)	Placeholder for imap(3) bit-packed word, which was removed (direction 3)
real	8	v	Volume of the volume
real	9	recipv	Reciprocal of the volume, $1/v$
real	10	avol(1)	Cross-sectional flow area, direction 1
real	11	avol(2)	Cross-sectional flow area, direction 2
real	12	avol(3)	Cross-sectional flow area, direction 3
real	13	dl(1)	Length of volume, direction 1
real	14	dl(2)	Length of volume, direction 2
real	15	dl(3)	Length of volume, direction 3
real	16	diamv(1)	Diameter of volume, direction 1
real	17	diamv(2)	Diameter of volume, direction 2
real	18	diamv(3)	Diameter of volume, direction 3

There is only one slot in the fa array for nvols, the number of volumes, but there are nvols slots for each one of the variables in the voldat comdeck that follow nvols, e.g., volno, v, recipv, etc. The storage is such that all the variables associated with one volume are contiguous and the same variable, e.g., volno, in adjacent volumes is a constant distance apart in memory. The skip factor is a parameter called ivskp.

#### 4.1.7 Junction database - jundat.hh

The database for each junction in the problem is stored in a set of variables that are defined in the jundat comdeck. The few lines of this file are shown below.

```

*comdeck jundat
    integer njuns(2,1), ij1(2,1), ij2(2,1), blhjc(1),
&           junftl(2,1), ij1vn(2,1), ij2vn(2,1)
    real*8 ajun(1), athrot(1), arat(1), diamj(1),
&           velfj(1), velfjo(1)

C
    equivalence (njuns(1,1),ia(1,1)), (ij1(1,1),ia(1,2)),
&           (ij2(1,1),ia(1,3)), (blhjc(1),ia(1,4)),
&           (ij1vn(1,1),ia(1,5)), (ij2vn(1,1),ia(1,6)),
&           (junftl(1,1),ia(1,7)), (ajun(1),fa(9)),
&           (athrot(1),fa(10)), (arat(1),fa(11)),
&           (diamj(1),fa(13)), (velfj(1),fa(14)),
&           (velfjo(1),fa(15)), (velgj(1),fa(16))

```

The first integer variable, njun, is the number of junctions. The second and third integer variables, ij1 and ij2, are the from and to volume input code numbers. The fourth integer variable, blhjc, is a placeholder for the jc bit-packed word, which was removed. The fifth and sixth integer words, ij1vn and ij2vn, are to and from volume ordinal numbers. The seventh real word, junftl, is the mixture volumetric flow rate for the junction divided by the total mixture volumetric flow rate on that end of the volume. There are two slots for this variable, slot 1 is for the from volume and slot 2 is for the to volume. The ninth real word, ajun, is the area of the junction. The tenth real word, athrot, is the ratio of the orifice area to the junction area. The eleventh real word, arat, is the mixture volumetric flow rate for the junction divided by the total mixture volumetric flow rate on that end of the volume. There are two slots for this variable. Slot 1 is for from volume and slot 2 is for the to volume. The thirteenth real word, diamj, is the diameter of the junction. The fourteenth real word, velfj, is the liquid velocity. The fifteenth real word, velfjo, is the liquid velocity at the previous time step. The sixteenth real word, velgj, is the vapor velocity. **Table 4.1-3** shows the arrangement of storage. The actual location in the fa array where the jundat storage block is located is determined at runtime and is based on the problem. All the indices, offsets, etc., are determined in the code using subroutines from the FTB package. The starting index for the jundat block is kept in filndx(5).

**Table 4.1-3** Jundat Comdeck Variable Arrangement

Type	fa slot	Variable	Description
integer	1	njuns	Number of junctions
integer	2	ij1	From volume input code
integer	3	ij2	To volume input code
integer	4	blhjc	Placeholder for jc(1) bit-packed word, which was removed
integer	5	ij1vn	From volume ordinal number
integer	6	ij2vn	To volume ordinal number
integer	7	junftl(1)	From pointer in output form without sign
integer	8	junftl(2)	To pointer in output form without sign

**Table 4.1-3** Jundat Comdeck Variable Arrangement

Type	fa slot	Variable	Description
real	9	ajun	Area of junction
real	10	athrot	Ratio of orifice area to junction area
real	11	arat(1)	Mixture volumetric flow rate for the junction divided by the total mixture volumetric flow rate on that end of the volume. 1 is for from volume.
real	12	arat(2)	Mixture volumetric flow rate for the junction divided by the total mixture volumetric flow rate on that end of the volume. 2 is for to volume.
real	13	diamj	Diameter of junction
real	14	velfj	Liquid velocity
real	15	velfjo	Liquid velocity previous time step
real	16	velgj	Vapor velocity

As in the case of the voldat database, there is only one slot in the fa array for the number of junctions, njun. As in the volume case, there are njun slots for each of the variables that follow njun, e.g., ajun, diamj, etc. The storage is such that all the variables associated with one junction are contiguous and the same variable, e.g., ajun, in adjacent junctions is a constant distance apart in memory. The skip factor is a parameter called ijskp.

#### 4.1.8 Scratch database - scrtch.hh

The scratch database is used to hold variables that do not need to survive from time step to time step. Some of the variables only need to survive from a loop at the front of a subroutine to be used in another loop later in the subroutine. Some variables need to survive from one subroutine, where the variable is first computed, to a subroutine that is called later where the variable is used or possibly recomputed. The scratch database can hold either volume data or junction data, and since the number of volumes is not equal to the number of junctions, the length of the scratch area is determined by the maximum of the number of volumes and the number of junctions. The number of variables in the scratch database for each volume or junction is set by the parameter, scskp. Each of the variables in the scrtch file are equivalenced to an fa slot and the start of the scrtch database is set by filndx(1).

Information on the layout of the scratch database is detailed in four lists in the scrtchc.hh file. These lists are ordered as follows:

## Labeled Common Blocks

1. The variables are listed by increasing slot number. This list indicates the type of variable: vol or jun. It also indicates in which subroutine the variable is finished being used and can be set to a NaN if the code was built with the NaN option.
2. The variables are listed in alphabetic order.
3. The variables used in each subroutine are listed by increasing slot number.
4. The variables are listed by increasing slot number. This list has ">"s between the subroutine names to indicate that the variable has to survive from the previous subroutine to the following subroutine. This list is useful for determining if a slot or variable is free to be used elsewhere.

## **5 FILES**

RELAP5 needs to read in an input file and write out an output file when it is run. If a restart is requested, then a restart-plot file is also generated.

### **5.1 Input**

The input file is entered after the -i on the command line. Normally, these files end with a “.i”. If no input file name is given on the command line, then the default file name, indta, is used as the input file.

The input file is opened in gninit and read into the fa array in rcards. It is read into dynamic common block 1. After the file is read into memory, individual cards are read in the r-level subroutines out of the fa array. The file is closed in xxx.

### **5.2 Output**

Normally, there are two output files: the “printed” output file and the “tape” output file. These designations were given to the files when the code was developed on the large mainframe computers. On these computers, the input file was really a set of punched cards, the output print file was 132 column computer printer paper, and the output tape file was magnetic tape. On the present day computers, all of these files are disk files.

#### **5.2.1 Print file**

The print file is where the major and minor edits are written. This file name is entered on the command line after the -o option. If the file name is not entered on the command line, then the default file name of outdta is used. The print file is opened in the xxx subroutine as unit xxx. It is closed in the xxx subroutine.

#### **5.2.2 Restart-plot file**

The restart-plot file is really two files merged into one file. It was initially set up this way to minimize the number of magnetic tapes required for a RELAP5 run. The name of the restart-plot file is entered on the command line after the -r option. If a restart-plot file name is not entered, the default name of rstplt is used. The restart-plot file is opened in the xxx subroutine as unit xxx. It is closed in the xxx subroutine.

The restart-plot is a binary file with multiple records. Each record containing data is preceded by a header record that identifies the type of the record that follows. This file is opened in the gninit subroutine as unit xxx. The restart portion of the file is written in the xxx subroutine. If this is a restart run, the restart portion of the file is read into memory in the xxx subroutine. The plot portion of the file is written by two subroutines, wrplid and pltwrt. The wrplid subroutine writes the header information. The header information for the plots and consists of two records. The first record contains the alphanumeric portion of the plot variable names, which are identical to the minor edit variable names. The second record contains

## Output

the numeric portion of the plot variable names. An example of the alphanumeric portion is tempf and an example of the numeric portion is 3010000. The actual values of the plot variables are written in the pltwrt subroutine. These plot variables, even though they are double precision (real\*8) in memory, are written out as single precision (real\*4) variables. The conversion is done by the sqoz subroutine. This keeps the plot file smaller and is more than sufficient for plotting purposes. However, be advised that if you do arithmetic on these plot variables using subtraction in xmgr5, that you can lose significant figures very quickly if the values are close to each other. The restart-plot file is closed in the xxx subroutine.

## **6 ENVIRONMENTAL SUBROUTINES**

### **6.1 CVI Package**

The details of the CVI package are contained in Section 9 on page 81. Basically, the CVI package is responsible for reading in the free-format RELAP5 input cards and parsing the information on the cards into data fields. It determines the type of the data in the fields, i.e., integer, real, non-numeric or Hollerith (character). If the CVI package finds an error in one or more of the data fields, the error is marked on the output so the user can easily fix it.

### **6.2 INP Package**

The details of the INP package are contained in Section 11 on page 223. Basically, the INP package is responsible for interpreting the information on the cards that is returned by the CVI package. It checks for the correct number of words on the card, the type of data in each of the words on the card. It is also responsible for expanding the input by creating additional cards if requested by the input on the card supplied in the input deck. This simplifies the input deck for the user. The INP package can also handle multiple input cases that are assembled together into one input deck. The individual cases are separated by a slash (/) cards.

### **6.3 FTB Package**

The details of the FTB package are contained in Section 14 on page 267. Basically, the FTB package is responsible for the dynamic storage management that is used in RELAP5 to minimize the memory required by the code. The FTB package is capable of managing three levels of storage, but only one (small core memory) is used by RELAP5.

## **6.4 Thermodynamic Property Package**

### **6.4.1 Steam Table Generator**

The details of the original steam table generator are contained in Section 12.1 on page 244. The information on the new steam table generator is yet to be added to this manual.

### **6.4.2 Interpolation Subroutines**

The details of the original steam table interpolation subroutines are contained in Section 12.2 on page 251. The information on the new steam table interpolation subroutines is yet to be added to this manual.

## **6.5 Buffered I/O Subroutines**

These subroutines are only used on the CDC computers.

## **6.6 Locf Subroutines**

These subroutines determine the memory location of a variable. They are used by a number of subroutines, e.g., they are used to resolve the memory location for the temperature in a volume so that the heat conduction subroutine does not have to look it up at each time step. They are also used by the restart subroutines to get the start and ending locations for the data contained in a common block, so that the data can be written to the disk file as one binary write.

## **6.7 Date and Time**

## **6.8 CPU Time**

## **7 INSTALLATION AND MAINTENANCE**

### **7.1 Computers**

RELAP5 runs on UNIX, CRAY, CDC, Windows, and Linux computers. It was originally vectorized for the CRAY, but since the CRAY is not a popular computer to run RELAP5 on now days, most of the vectorization has been taken out of the code in order to simplify it for easier maintenance. The types of UNIX computers include DEC, IBM, HP, SGI, SUN. The code has been installed on systems using the Mandrake and SuSE Linux distributions on Intel chips using both the g77 and Intel ifc compilers. The code has also been installed on a SuSE Linux system running on a Macintosh computer that uses the Motorola PowerPC chip.

### **7.2 Directory Structure**

The distribution of RELAP5 consists of 11 main files: conv32.f, configure, dutilty, envrl.s, goodies, indecks, rdmr.s, ReadMe, relap.s, select.f, and usplit.f, plus a large number of Windows specific files. The conv32.f file is the Fortran source for a program that adds "(2," to integers equivalenced to real\*8 words in the fa array, "d0" to real\*8 constants, etc. The configure file is a c-shell script for determining the platform on which the installation is occurring and building the .config file. The dutilty file is a c-shell script to build the executables for cnv32, select, and usplit. The envrl.s file is a concatenated file of all the routines in the environmental library. The goodies file contains a large number of miscellaneous files that are needed to built the RELAP5 code. The indecks file is a concatenated file of all the input decks that are run as part of the installation. The rdmr.s file is a concatenated file of all the coupling routines for building the RELAP5-PARCS code. The ReadMe file has instructions for building the code on various computers. The relap.s file is a concatenated file of all the relap5 routines. The select.f file is the Fortran source for a program that is used to preprocess the Fortran source code so that multiple versions can be maintained in one file. The usplit.f file is the Fortran source for a program that splits out the individual parts from any of the concatenated files that are part of the distribution.

#### **7.2.1 relap**

This directory contains all the main RELAP5 .ff and .hh files.

#### **7.2.2 envrl**

The envrl directory contains all the environmental .ff and .hh files. These are files that have more widespread use than just RELAP5. The ASTEM steam table set of subroutines is in the envrl directory.

#### **7.2.3 run**

The run directory contains all the input decks and the output and rstplt files after the input decks have been run.

## 7.3 Installation Scripts

The installation of the RELAP5 program progresses in three steps. During the first step, the environmental library, envrl.a, is built. This library contains common environmental subroutines that are useful to many programs including RELAP5. After the environmental library is build, the steam tables are built. There are currently three steam tables in RELAP5: tpfh2o, tpf2o, and tpfh2onew. Tpfh2o is the original water properties tables and tpfh2onew is the new water properties tables. Tpf2o is the heavy water properties tables. The last step in the installation of RELAP5 is the building of the relap.a library. These relap and environmental libraries are linked with the relap5.ff code to give the relap5.x executable program.

## 7.4 dinstls

### 7.4.1 denvrl

### 7.4.2 drelap

### 7.4.3 dstgxx

### 7.4.4 runx

## 7.5 Precompliers

### 7.5.1 select

### 7.5.2 cnv32 - mlist - in32 - in32end

### 7.5.3 Makefiles

### 7.5.4 bldmak script

### 7.5.5 Makee - Liste

### 7.5.6 Maker - Listr

## 7.6 Libraries

### 7.6.1 envrl.a

### 7.6.2 relap.a

## 7.7 Revision Control System

## 8 DEBUGGING TOOLS

This chapter contains a description of some of the debugging tools that are included as part of the RELAP5 distribution. The MakeNew script can be used to make minor updates to specific subroutines and then recompile the code and run it with the changes added. The code also allows diagnostic debug printouts via the 105 card and via the call to the snapit subroutine. Additional plot variables can be obtained by including 2080xxxx cards in the input deck. The code also contains some extra unassigned variables that can be used for various purposes like testing a modification to the code via the MakeNew script, or for plotting quantities that are not normally available to the plotting output via the 2080xxxx cards. There are 20 testda variables available. These are like global variables in that there are only 20 of them available in the code no matter what problem is being run. In addition, there are either 20 or 100 extra volume variables. These are like volume variables, e.g., p, in that there are 20 or 100 of these variables available for each volume. There are also 20 extra junction variables available. These are like junction variables, e.g., velfj, in that there are 20 of these variables available for each junction. There are also 20 extra system variables available. These are like system variables, e.g., sysmer, in that there are 20 of these variables available for each system. Systems are a collection of volumes and junctions that are hydraulically independent of all the other system. For example, the primary and two secondary systems in the typical PWR input deck are not connected hydraulically, so they are separate systems.

### 8.1 MakeNew Script

The MakeNew script is a script that creates a set of parallel directories to the relap, envrl, and run directories that are created when RELAP5 is installed. Into these parallel directories the programmer can copy a file from the original installation directory and make changes to the coding without affecting the original source file. When you run the MakeNew script, you give it one argument, which will be the prefix for the new parallel directories. For example, if we are in the parent directory of the relap, envrl, and run directories and type the command:

```
MakeNew new
```

The MakeNew script will make four new parallel directories: newRelap, newEnvrl, newRun, and newMods. The first three new directories are like the corresponding directories in the original installation. These three directories initially contain only copies of the \*.h files and no \*.ff or \*.hh files. If we wanted to make a modification to the vexplt.ff file, we need to get a copy of the original vexplt.ff file from the relap directory. This is done by using the following command from the inside the newRelap directory:

```
Getit vexplt.ff
```

The Getit script copies the vexplt.ff file from the relap directory into the newRelap directory and makes it writable so that we can change it.

We can now modify vexplt.ff in the newRelap directory. When we are done modifying the file, we can recompile it by using the command

```
doitf vexplt
```

The doitf script runs the modified vexplt.ff file through the select and cnv32 precompilers to generate the vexplt.f file. It then sends vexplt.f to the FORTRAN compiler to generate the vexplt.o file. The doitf

script does one other useful thing and that is to copy the modified version of vexplt.ff to the newMods directory. Thus, after the programmer is done modifying all the files necessary to make the change, copies of the modified files are in one directory, newMods.

The next step after the compilation is to load the new object files in with the other unmodified RELAP5 files to make a new executable file. This is done by modifying the dloadnew script. The new object files are added after the \$LOAD command in the dloadnew script. The line above and below the added vexplt.o line is shown below.

```
$LOAD -o relap5new.x relap5.o \
vexplt.o \
relaplr.a relaplq.a ../newEnvrl/envrl.a $rdmrlib
```

When the dloadnew script is run with an parameter of the machine type, a new version of RELAP5 will be built, relap5new.x, that contains the revised version of vexplt. To make the execution of this new version easier, in the newRun directory there are two other scripts, Runitorig and Runitnew, that can be used to run either the original or the newly created RELAP5 code. The argument to these scripts is the input deck name without the trailing .i. For example, if you change to the newRun directory and copy over the Edwards pipe problem, edhtrk.i by using the command

```
Getit edhtrk.i
```

Now, you can run this problem with the new code by using the command

```
Runitnew edhtrk
```

The output file can be compared with the original output file in the run directory by use of another script, Cf. This script takes as its one argument the name of the file to be compared,

```
Cf edhtrk.o
```

The Cf script is also available in the newRelap and newEnvrl directories for comparing \*.ff and \*.hh files you have modified with the originals in the relap and envrl directories. There are also Cf types scripts in the newMods directory, only they need to know where the original file is located, so they are two scripts, Cfs and Cfe. The Cfs script makes its comparison the relap directory file and the Cfe script makes its comparison with in the envrl directory file.

## 8.2 Diagnostic Debug Printouts

The 105 card can be used to provide diagnostic debug printouts at a specific value of ncount. This card was originally added for stopping the code gracefully when the amount of CPU time was about to expire. On the mainframe computers, the maximum amount of CPU time for the job was entered as part of the job control cards. This prevented the user from using up more CPU time than he or she could afford. The first three words on the 105 card are CPU times and are of no use on present day computers, but the last two words are useful if you want the code to generate a debug printout at a specific value of ncount. The user just enters the same ncount number for both words 4 and 5 on this card, and the code will generate the debug printout and stop when that value of ncount is encountered. This has limited usefulness with the current debuggers that are available on modern computers, but it may come in handy in some cases.

## 8.3 Dynamic Block Snapit Printouts

The snapit subroutine calls a set of p-level subroutines to printout the contents of some of the more common dynamic common blocks, like voldat, jundat, etc. This subroutine is in the relap directory and its subroutine statement is:

```
subroutine snapit (subnam, filnam)
subnam is the normally the name of the calling subroutine
filnam is the name of the file that contains the output
```

An example of the use of snapit is in the hydro.ff file as comment cards. It is setup to call snapit on the first time step, i.e., when ncount = 1.

```
if (ncount .eq. 1) call snapit('hydro','snap.hydro')
```

The dynamic comdecks that can be printed include lpdat, contrl, voldat, and jundat. The user can add or remove the comments from column 1 in the snapit.ff file and control which of these dynamic comdecks get printed. This is most easily done by using the MakeNew script described above.

## 8.4 Additional Variables

This section covers the various types of additional variables that are available to the programmer. All of these variables are available for plotting using xmgr5 by using the 2080xxxx cards. The 2080xxxx cards will be discussed first, and then the various types of additional variables that are available to the programmer will be discussed.

### 8.4.1 2080xxxx Card Variables

The 2080xxxx cards can be used to request additional plot variables. A standard set of variables is normally written to the rstplt file, and there are some that are available in the basic code that are available but not normally written to the rstplt file. The list of available variables is given in Section A4 Minor Edits, 301-399, in the RELAP5 Input Requirements manual. The variables that are listed there that have an underscore under the name are written to the rstplt file. The exception is the extxxx variables. These are only written when the code has been built with the extra flag. This flag is the fourth parameter in the dinstls command. For example, to build the code on an HP computer with the extra variables available, the programmer would build the code using the command

```
dinstls hp relap nonpa extra
```

This code would then write the extsnn, extvnn, and extjnn variables to the rstplt file. The testda variables, however, have to be requested by using 2080xxxx cards.

If the code is not built with extra on, the extra variables are not available to the programmer. Another word of caution is that a rstplt file from a version of the code with extra not on is not compatible with a rstplt file from a version of the code with extra on. Both the restart and plot portions of the rstplt file are larger when extra is on because on the extra variables.

### 8.4.2 Testda Variables

The testda variables are global variables in that there are only 20 of them total. They are always available, no matter what options you built the code with. There is not one testda variable for each volume or one for each junction, like the extvnn and extjnn variables. The testda variables are useful for computing a quantity that is global, like the total energy lost from the system from all the breaks and writing it to the rstplt file using a 2080xxxx card so that you can plot it.

### 8.4.3 Extra Volume Variables (extvnn)

The extra volume variables, extvnn, are available whenever the code was built using the extra parameter on the dinstls command line. There are 20 extra volume variables available for each volume, so these variables are like pressure, temperature, etc. They have names like extv01, extv02, ... extv20. With some coding changes, 100 extra volume variables can be compiled into the code also, but this is normally more extra volume variables than the programmer needs to have. Keeping track of more than 20 extra volume variables requires a good bookkeeping system.

These variables are useful for debugging the code and for making temporary changes to the code to check out a potential new model or to fix an ailing model. They can not be used in control systems or in table lookups, because they were implemented only in the plot file part of the code. In order to be used in control systems, they would have to be integrated into the scnreq subroutine, which is a tall order given the complexity of this subroutine.

### 8.4.4 Extra Junction Variables (extjnn)

The extra junction variables, extjnn are also available whenever the code is built using the extra parameter on the dinstls command line. They have names like extj01, extj02, ... extj20. There are 20 of these variables for each junction.

### 8.4.5 Extra System Variables (extsnsn)

The extra system variables, extsnsn are also available whenever the code is built using the extra parameter on the dinstls command line. They have names like exts01, exts02, ... exts20. There are 20 of these variables for each system. A system is like the primary or secondary system in the typical PWR model.

## 8.5 Flowcharter

The flowcharter is useful to examine the logical flow through a subroutine or function. Its output is 130 columns wide and is not much longer than a listing the subroutine. There is only one extra line added for each target of a go to statement. An example flowchart is shown below in **Figure 85-1**. Note that the do loops and if-endif block limits are shown on the left, and the go to branches are shown on the right. The letters that are used to show the limits of the do loops and if-endif blocks start with a and go up through the alphabet. A similar set is used for the go to traces. This makes following these traces across multiple pages much easier.

**Figure 85-1** Flowchart for the timstop Subroutine



## 9 CVI PACKAGE

The CVI package contains two subroutines, cvic and cvirc, one test program, tcvi, and one test input deck, tcvid. The cvirc subroutine reads one card into a internal array and then calls cvic to process the card. While cards are not used anymore for input, it is still convenient in this description to refer to each individual line of input as a card. The cvic subroutine processes one card of data that are in fields which are self delimiting and which can contain data in integer and floating point decimal, hexadecimal, octal, and three character (Hollerith) forms. The principal advantage of this input subroutine over the standard FORTRAN 77 formatted input subroutine is that the input data need not be entered in predetermined columns. This allows the design of input data forms to be simpler and the data entry easier. The subroutine also facilitates the handling of data input lines with variable numbers of words per input line.

Two typical input lines are:

```
105,318,105+2,-14,-25-4 *THIS IS A COMMENT  
10 25 2.1E-3 -1.4+5 245.65, TEST
```

The cvic subroutine converts the first two integer input fields, 105 and 318, to their binary integer equivalents. The floating point entry, 105+2, is converted to the binary floating equivalent of 10.5. In a similar fashion, the next integer entry, -14, is converted to a negative binary integer, and the next floating point entry, -25-4, is converted to the binary floating point equivalent of -0.000025. The asterisk indicates the end of the input fields to be converted, and the remainder of the input line contains the comment, THIS IS A COMMENT. The second input line contains two integers, 10 and 25, three floating point numbers, 0.0021, -140,000, and 245.65, and one Hollerith string, TEST.

### 9.1 Data Field Description

Input information on a card is grouped into fields. A field on a card is defined to be those characters of information which are between two successive separators, with one exception that will be mentioned in the Hollerith description. The first field on a data card is assumed to start at the first non-blank character. The last field on a card extends up to the end of the scanned characters. If the last field is completely blank, it is ignored.

There are three type of fields on a card: (1) integer, (2) real, and (3) non-numeric or Hollerith. This last type of field will be referred to as Hollerith in the following discussion. Field type is determined from context, except that the Hollerith type can be forced regardless of context by the use of the two exceptional characters: begin and end Hollerith indicators.

The conversion process uses a left-to-right scan of the input line columns. The scan begins with a search for the beginning of a field, where the beginning of a field is the first non-blank character. Partial conversion occurs during the scan of the field, and the conversion is completed after detecting the end of the field. If no errors are found, zero is returned for the error flag. If errors are found, a positive number is returned for the error flag. If the error was detected during the scan of the field, the number returned for the error flag is the column within the field. If the error was detected during the completion of the conversion, the number returned for the error flag is the terminating column number. When the conversion of a field is completed, the search for a new field resumes. Thus, fields may be separated by any number of blanks. If an asterisk (\*) or dollar sign (\$) is encountered during the search for a new field, the scan is terminated and

the remainder of the input line may contain comments. If a field is terminated by an asterisk or dollar sign, the conversion is completed and the remainder of the field may be comments. Decimal, hexadecimal, octal, and Hollerith fields can be mixed in any order on a data line.

The rules governing the conversion are listed separately for each type. Some of the paragraphs are numbered in the following discussion to permit cross references between the paragraphs.

### 9.1.1 Decimal Fields

Decimal fields include both integers and floating point numbers. Floating point numbers are real\*8 (64 bits long) and integers are integer\*4 (32 bits long). The integers are doubly subscripted arrays, e.g., ibin(2,40). In the code, the integers and real\*8 variables are equivalenced in such a way that the integers occupy the lower 32 bits of a 64-bit word, e.g. equivalence (bin(1),ibin(1,1)). The integers are referenced with the first subscript equal to two, e.g., ibin(2,i), so as to get the lower 32 bits of the 64 bit word. Integers do not have decimal points or exponents in their field, whereas floating point numbers have one or both of these. Fields are separated by blanks or commas. The rules for decimal fields are discussed in the following paragraphs. Note, while many of these comments are directed at CPUs with 32-bit 2-s complement arithmetic, the INP subroutines also work on the Cray computer.

1. Each decimal field generates one binary word. Fields are initiated by the first non-blank character that is encountered in the left-to-right scan of the card. A field is terminated by either a blank (), a comma (,), an asterisk (\*), or a dollar sign (\$).
2. The sign of the decimal number, if present, occupies the first column of the field. A plus sign (+) is assumed if no sign is present.
3. An integer is indicated if none of the following items are present in a field:
  - a decimal point (.), or
  - a sign, called an exponent sign, following a digit inside a field and followed by an integer number, or
  - an E (e) or D (d) with or without a following exponent sign (+, -, or &) following a digit inside a field and followed by an integer number.
4. A floating point quantity is indicated if any of the conditions listed in (3) above are present. The digits preceding the exponent sign, E or D, make up the fractional part of the floating point number. If no decimal point is present, a decimal point is assumed to be in front of the first digit of the fraction. The exponent sign and following number are the power of ten by which the fraction is multiplied. The exponent field does not need to be present if a decimal point is used. One blank immediately following an E or D is treated as if it were a plus exponent sign. This is the only exception to the rule that a blank terminates a decimal field.

5. Note that the decimal field format is such that data input lines generated by FORTRAN 77 subroutines using I, E, D, F, or G output conversion can be read in by cvic as long as at least one blank or a comma separates each field. Also this format is compatible with the input conventions of FORTRAN 77 except:
  - the restrictions of no blanks between characters in the number,
  - the convention of an assumed decimal point at the beginning of the fractional part of a number if no decimal point is present, and
  - the placement of data on the input line with arbitrary spacing between fields.
6. The range of integer data must be such that it can be stored in a four-byte word (32 bits). Thus, the integer quantity must be equal to or greater than  $-2^{31}$  (-2,147,483,648) and less than  $+2^{31} - 1$  (+2,147,483,647). The double precision word (64 bits) appears as a correct double precision integer in that the high order 32 bits are set to zero if the integer is positive and are set to ones if the integer is negative.
7. The magnitude of the fractional part of the floating point quantity with the decimal point removed must be less than  $2^{64}$ . The magnitude of the floating point quantity must be within the floating point range of the CPU. On the UNIX workstations that use IEEE-754 arithmetic, this range is  $2.22507\dots \times 10^{-308}$  to  $1.79769\dots \times 10^{+308}$  for positive numbers and  $-1.79769\dots \times 10^{+308}$  to  $-2.22507\dots \times 10^{-308}$  for negative numbers. Floating point values beyond these values are handled by the floating point overflow and underflow subroutines supplied by the operating system. A floating point quantity is converted as a double precision floating point number. Both integer zero and floating point zero, regardless of sign, are stored as plus zero, which is all zero bits.
8. Both integer and floating point zeros will thus test zero in either an integer or floating point test. Hence, +0, -0, +0.0+0 all generate the same binary quantity. (Note that the machine is incapable of representing integer minus zero, but can represent floating point minus zero.)
9. Leading zeros are ignored. However, zeros preceding non-zero digits are significant in a floating point number without a decimal point because of the assumed decimal point in front of the first zero.
10. Errors detected in decimal fields include illegal characters, multiple decimal points, a decimal point in the exponent field, sign placement errors, multiple Es or Ds, no digits in the fraction or exponent fields, or magnitude errors described in (6) and (7) above. Floating point overflow and underflow are not detected by cvic. A comma may be used only to terminate a field. A comma preceded by a blank is an error. Commas entered in adjacent columns may not be used to enter a zero.

### 9.1.2 Hexadecimal Fields

1. Each hexadecimal field generates one binary word. The hexadecimal (base 16) field is initiated by the first two non-blank characters being slash-x (/x). The field is terminated by either a blank ( ), a comma (,), an asterisk (\*), or a dollar sign (\$).
2. The digits allowed in a hexadecimal field are the ten decimal digits, 0 through 9, and the six alphabetic characters, A (a) through F (f). The alphabetic characters have values 10 through 15. No signs are allowed. The converted quantity is assumed to be a hexadecimal integer. Leading zeros are ignored.
3. Hexadecimal numbers can be considered a short, convenient form for binary information, and are usually used to enter masks for packing bits into a word and unpacking bits from a word. Each hexadecimal number represents four binary bits. If the hexadecimal number represents a negative numeric quantity, then it needs to be entered in two's complement form.
4. A field can contain up to sixteen hexadecimal digits (64 bits).
5. Examples of hexadecimal fields are /xff00, /xffffffff0, and /x123456789abcdef.
6. Illegal characters and magnitude violations are detected as errors.

### 9.1.3 Octal Fields

Octal (base 8) fields are similar to the hexadecimal fields with the following differences.

1. The octal field is initiated by the first two non-blank characters being slash-oh (/o).
2. The digits allowed in an octal field are the eight digits, 0 through 7.
3. The size of the binary quantity must not exceed 64 bits. Thus, a maximum of 21 octal digits are allowed and the 21st octal digit from the right must be less than 2.
4. Examples of octal fields are /o774, /o3777777770, and /o1234567.

### 9.1.4 Hollerith or Non-Numeric Fields

Three types of Hollerith fields are allowed, two of which are identical to the Hollerith fields used in FORTRAN 77 format and data statements.

1. Each Hollerith field generates one or more binary words. One of the FORTRAN-like fields is initiated by the first non-blank character being an apostrophe ('') or quotation mark (""). In the following discussion, we will refer to both the apostrophe and the quotation

mark as the apostrophe. Fields started by an apostrophe are terminated by an apostrophe. The end of an input line cannot terminate the Hollerith field started by an apostrophe.

2. The second FORTRAN-like field is initiated by the letter H (h) immediately following an integer decimal number. Fields started with the letter H extend for the number of columns indicated by the decimal number that precedes the letter H. The number of columns indicated must not extend beyond column 80. In these two types of Hollerith fields, it is an error if a blank or a comma does not follow the field unless the field is terminated in column 80.
3. The third type of Hollerith field is indicated by the first non-blank character being other than a digit (0-9), a plus sign (+), a minus sign (-), a comma (,), an asterisk (\*), a dollar sign (\$), or a slash (/). Thus, the first character being either an alphabetic character or any other character except those just listed, indicates the third type of Hollerith field. The third type of Hollerith field is terminated by a comma or a blank.
4. All 256 ASCII characters are allowed in the first two types of Hollerith fields. The third type is restricted only in the first character of the field, and of course, the terminating characters cannot be entered. The restrictions on the first character exist because these characters specify other formats or terminate the scan. For example, a sign or a digit indicates a decimal field. The field 100K is an erroneous decimal field, but '100K' is a valid Hollerith field. An asterisk or dollar sign in a Hollerith field does not terminate the scan of the input line. In fields initiated by an apostrophe, and apostrophe can be entered by entering two apostrophes for everyone wanted in the final string. This is done to distinguish it from the terminating apostrophe. As examples of Hollerith fields and rules for apostrophes, the character field don't and 'don't' are written using the first two Hollerith types as:
5. 5hdon't, 'don"t' 7h'don't' "dont"t"
6. The Hollerith field don't can be written simply as don't when using the third type of Hollerith field. However, the field'don't' can not be entered in that Hollerith field type. A single apostrophe in column 80 or an odd numbered apostrophe of consecutive apostrophes ending on column 80 is treated as a terminating apostrophe.
7. The first character is stored in the left-most byte of the storage location. Succeeding characters are stored in consecutive bytes. Eight characters are stored in a word. If necessary, blanks are inserted to pad out the word.

## 9.2 Subroutine Usage

The main subroutine that does the parsing of the input 80-column card is cvic. The cvirc subroutine reads one card at a time from the input unit (defined in eflesd.hh file as unit 11) and then calls cvic to parse the card.

### 9.2.1 cvic - Decode One Card

The cvic subroutine is used to parse one 80-column card. The converted binary information from the card is stored in the bin array. For each word stored in the bin array, a format code word is stored in the ic array. **Table 9.2-1** contains the various values of the format codes.

**Table 9.2-1** Format Codes Returned from Cvic Subroutine

Code	Meaning
0	Zero in a decimal field (integer or floating point)
1	Integer number entered in a decimal field (non-zero)
2	Floating point number entered in a decimal field (non-zero)
3	Hexadecimal number entered in a hexadecimal field
4	Octal number entered in an octal field
< 0	Hollerith information entered in Hollerith field

Hollerith fields can generate more than one binary word. The format code word for the first binary word of a Hollerith field contains the number of characters in the field as a negative quantity. Succeeding format code words generated from the Hollerith field have a value of 8 greater than the preceding code. The code for the last binary word of a Hollerith field contains a number less than zero and equal to or greater than -8. The format code numbers do not include the blanks that may have been added to fill out the remainder of the 8-character word. **Table 9.2-2** shows the values of the bin and ic arrays for the following Hollerith input line:

```
'abcdefgij'      'klm'
```

**Table 9.2-2** Example of Hollerith Data in the bin and ic Arrays

i	bin(i)	ic(2,i)
1	abcdefg	-10
2	ij	-2
3	klm	-3

The subroutine returns in num the number of binary quantities converted until a normal scan termination or a detection of an error. The variable num will be zero if the card is all blank. It could also be zero if the first non-blank character on the card is an asterisk or dollar sign.

The variable nerr is used as both an input quantity and as an output quantity. If nerr is zero on input, and an error is detected, a one line error statement is printed on unit 6 just above an echo of the card in error. This error print statement is:

```
***** error in card below at col.
```

If nerr is non-zero on input, then no error statement is printed. On return from cvic, if no error is detected, nerr is zero. If an error occurs, nerr contains the column number of the error on the card.

Note that the bin array can contain a mixture of real\*8, integer\*4, and Hollerith information. Thus, the FORTRAN variables used for this information must be chosen such that no undesirable integer-to-real or real-to-integer conversions occurs. Equivalence statements can be used to avoid this problem, e.g., here are some suggested specification statements for the calling subroutine:

```
character*80 bcd
real*8 bin(40)
integer ibin(2,40), ic(2,40), num, nerr
equivalence (bin(1),ibin(1,1))
```

If the ith variable is real\*8, it can be obtained from bin(i), and if it is integer, it can be obtained ibin(2,i). Also a common user error when inputting data with this subroutine is to enter a floating point number in integer form or visa versa. This can be detected by testing the appropriate entry in the ic array. The cvic subroutine statement is:

```
subroutine cvic (bcd, bin, ic, num, nerr)
bcd      character*80 field containing the card to be converted input
bin      40-word real*8 array containing num converted quantities; output
ic       40-word integer array, ic(2,40), containing num code words for the
          converted quantities; output
num     number of converted quantities; output
nerr    error flag, input/output
input:
          nerr = 0, any error comments are written to the output file
          nerr > 0, no error comments are written to the output file
output:
          nerr = 0, no error
          nerr > 0, column number containing the error
```

### **9.2.2 cvirc - Read One Card and Call cvic**

The cvirc subroutine reads one 80-column card image from the input (unit 11) file, calls the cvic subroutine to parse the card image, and returns. The cvirc subroutine statement is identical to the cvic subroutine statement.

```
subroutine cvirc (bcd, bin, ic, num, nerr)
```

### **9.2.3 tcvi - Test cvic**

The tcvi program can be used to test the cvic subroutine. The data input file for this program, tcvid, is useful because it contains many examples of legal and illegal input data cards. The tcvi program also contains examples of ways to check the format code array for format errors.

## Subroutine Usage

## 10 COMDECK DESCRIPTIONS

This chapter contains a description of the comdecks in the RELAP5 distribution. These comdecks consist of:

- labeled common blocks along with their specification statements,
- dynamic blocks along with their specification and equivalence statements,
- specification statements alone,
- data statements alone, or
- statement functions alone.

Comdecks containing labeled common blocks are your normal FORTRAN 77 common block. Comdecks containing dynamic blocks are how RELAP5 is able to use dynamic memory allocation in FORTRAN 77 and still have meaningful mnemonics for the variable names. These dynamic blocks are allocated using the FTB package subroutines from the large fa array. Comdecks containing specification statements alone are quite often used in combination with another comdeck containing a set of corresponding data statements alone or another comdeck containing a set of corresponding statement functions alone. The dynamic block comdecks as well as the labeled common block comdecks will also contain their corresponding specification statements, e.g., integer, real, logical, and character statements.

### 10.1 cmpalf.hh

The cmpalf.hh comdeck is a data statement comdeck, and contains one character\*8 array, cmpalf. This array initializes the names of the components names used in RELAP5. The names and their position in the cmpalf array are given below. They are sorted alphabetically by name.

**Table 10.1-1** Component Names in Comdeck cmpalf.hh

Name	Index
accum	13
annulus	7
branch	5
eccmix	15
jetmixer	6
mpljun	3
multid	16

**Table 10.1-1** Component Names in Comdeck cmpalf.hh

Name	Index
pipe	1
prizer	17
pump	4
separatr	8
sngljun	11
snglvol	10
tmdpjun	9
tmdpvol	2
turbine	14
valve	12

## 10.2 cmpdac.hh & cmpdacc.hh

The cmpdac.hh comdeck is dynamic block for the accumulator component. It contains the specification statements for the accumulator variables and their equivalences to the fa array. The cmpdacc.hh comdeck contains a description of each of the variable names used in the cmpdac.hh comdeck. The descriptions of the variables are given below in alphabetical order where the slot number in the fa array that the variable is equivalenced to is given in the fa(i) column. The dimensions and type of the variable, real\*8, integer, logical, or character, are given in the type column. These variables start at slot 14 in the fa array. The first 13 slots are occupied by variables that are common to all the components and are described in the cmpdat.hh comdeck section.

**Table 10.2-1** Accumulator Component Variable Names in Comdeck cmpdac.hh

Variable	fa(i)	Type	Description
acctrp	52	I(2)	Word 1: accumulator trip number. Word 2: trip offset during input, index during transient
acpgtg	41	R	Vapor specific heat, $C_p$ , at the vapor temperature
acpnit	44	R	Noncondensable specific heat, $C_p$ , at the vapor temperature
acvgtg	42	R	Vapor specific heat, $C_v$ , at the vapor temperature
ahfgtf	36	R	Heat of vaporization at the liquid temperature
ahfgtg	37	R	Heat of vaporization at the vapor temperature
ahftg	40	R	Liquid enthalpy at the vapor temperature

**Table 10.2-1** Accumulator Component Variable Names in Comdeck cmpdac.hh

<b>Variable</b>	<b>fa(i)</b>	<b>Type</b>	<b>Description</b>
ahgtf	47	R	Vapor enthalpy at the liquid temperature
atank	45	R	Tank cross sectional area
avfgtf	39	R	$v_g - v_f$ at the liquid temperature
avgtg	38	R	Vapor specific volume at the vapor temperature
aviscn	43	R	Noncondensable viscosity
betav	33	R	Steam saturation coefficient of expansion
claptf	51	R	Clapyron coefficient at the liquid temperature
cvnit	31	R	Noncondensable specific heat capacity, $C_v$
dialn	13	R	Surge line diameter
diamtk	46	R	Tank diameter
dmgdt	50	R	Time rate of change in dome vapor mass (equilibrium model)
dpd	34	R	Variable used in solution matrix (right hand side)
dpddp	35	R	Variable used in solution matrix (left hand side)
dvliq	57	R	Change in liquid volume over last time step
dztank	48	R	Tank elevation change
gasln	54	R	Length of gas above the surge line
gaslno	55	R	Length of gas above the surge line, old-time
htcap	20	R	Tank wall specific heat capacity
htxr	22	R	Heat transfer flag
lnelv	14	R	Surge line elevation drop
lnlen	16	R	Surge line length
qtank	23	R	Heat transport to noncondensable from all sources
qtanko	49	R	Heat transport to noncondensable from all sources (old-time)
rhon	28	R	Noncondensable density
rhono	29	R	Noncondensable density, old-time
rhot	21	R	Tank wall density
thend	19	R	Noncondensable thermal conductivity
thick	15	R	Tank wall thickness

**Table 10.2-1** Accumulator Component Variable Names in Comdeck cmpdac.hh

<b>Variable</b>	<b>fa(i)</b>	<b>Type</b>	<b>Description</b>
tklen	56	R	Length of the tank above the surge line
ttank	17	R	Tank wall temperature
ttanko	18	R	Tank wall temperature, old-time
tvapo	30	R	Noncondensable temperature, old-time (tempg is new-time)
vdm	24	R	Dome volume (noncondensable)
vdmo	25	R	Dome volume (noncondensable), old-time
vliq	26	R	Liquid volume (tank and surge line)
vliqo	27	R	Liquid volume (tank and surge line), old-time
vtank	32	R	Tank volume

### 10.3 cmpdat.hh & cmpdatac.hh

The cmpdat.hh comdeck is a dynamic block for the components. It contains the specification statements for variables that are common to all the components and variables that are used for time dependent volumes and junctions. The cmpdatac.hh comdeck contains a description of each of the variable names used in the cmpdat.hh comdeck. The descriptions of the variables are given below in alphabetical order where the slot number in the fa array that the variable is equivalenced to is given in the fa(i) column. The dimensions and type of the variable, real\*8, integer, logical, or character, are given in the type column.

**Table 10.3-1** Component Variable Names in Comdeck cmpdat.hh

<b>Variable</b>	<b>fa(i)</b>	<b>Type</b>	<b>Description</b>
cmlplen	5	I	Length of component data
cmplnam	3	R	Component name (alphanumeric)
cmplnum	2	I	Component number
cmplopt	12	I	Old packed word, not used anymore
cmptbl	39	R	Table entries
cmptyp	4	I	Component type (defined in cmpalf.hh)
cyl360degflg7	19	L	
cylmltisctr5	17	L	
geometryflag4	16	L	
haslatchingoccurred5	17	L	

**Table 10.3-1** Component Variable Names in Comdeck cmpdat.hh

<b>Variable</b>	<b>fa(i)</b>	<b>Type</b>	<b>Description</b>
headntorqmltplrlflg8	20	L	
headntorqmltplrsatflg9	21	L	
is3dflg2	14	L	
ischkvlvclosed4	16	L	
isdrainflag2	14	L	
isdum3	15	L	
isdum6	18	L	
ishysteresis2	14	L	
isinitllyopen4	16	L	
isloopcomponent1	13	L	
ismulti3	15	L	
isnegvelflg5	17	L	
isopt1latch3	15	L	
isopt2latch6	18	L	
isprescntrld3	15	L	
ispumpstopflg2	14	L	
ispumpstoppedflg3	15	L	
isreliefvlv4	16	L	
isshfctconnectedflg4	16	L	
issphericalaccum8	20	L	
istripvlv3	15	L	
istripvlv4	16	L	
isturbtypeone3	15	L	
isturbtypetwo4	16	L	
isvlvcycled2	14	L	
motortorqreferalflg12	24	L	
ncmps	1	I	Number of components
nctalf	31	R	Alphanumeric part of variable request code

**Table 10.3-1 Component Variable Names in Comdeck cmpdat.hh**

<b>Variable</b>	<b>fa(i)</b>	<b>Type</b>	<b>Description</b>
nctble	36	I	Number of items/entry
nctblt	35	I	Table type
nctbtn	37	I	Total number of items
nctbx	38	I	Last index
nctdpv	32	I	Numeric part of variable request code
nctpc	33	I(2)	Word 1: 0 if common block, block number if dynamic block Word 2: offset, then index in transient subroutines
nctrp	29	I	Trip number
nctrx	30	I	Offset in trip block, then index to trip during transient subroutines
newtimenooutflowflg2	14	L	
newtimevolactveflg4	16	L	
njc	9	I	Number of junctions in component
njcn	10	I	Position number of first junction of this component
njco	11	I	Offset to first junction of this component in junction block
nonzeroinnerradiusflag6	18	L	
nvc	6	I	Number of volumes in component
nvcn	7	I	Position number of first volume of this component
nvco	8	I	Offset to first volume of this component in volume block
octantnumber2431	28	I	
oldtimenooutflowflg5	17	L	
oldtimevolactveflg7	19	L	
pumpspeedtblreferralsflg14	26	L	
pumpspeedtblreferralsatflg15	27	L	
snglphasreferalsatflg7	19	L	
snglphasreferralflg6	18	L	
twophasereferralflg10	22	L	

**Table 10.3-1** Component Variable Names in Comdeck cmpdat.hh

Variable	fa(i)	Type	Description
twophasereferralsatflg11	23	L	
velmassflwswtch0	12	L	

## 10.4 cmpdtv.hh & cmpdtvc.hh

The cmpdtv.hh comdeck is a dynamic block for the valves. It contains the specification statements for variables that are common to all the valves and variables that are used for specific valves like a trip valves, motor and servo valves, and relief valves. The cmpdtvc.hh comdeck contains a description of each of the variable names used in the cmpdtv.hh comdeck. The descriptions of the variables are given below in alphabetical order where the slot number in the fa array that the variable is equivalenced to is given in the fa(i) column. The dimensions and type of the variable, real\*8, integer, logical, or character, are given in the type column. The valve type column indicates the type of the valve for which this variable is used.

**Table 10.4-1** Valve Component Variable Names in Comdeck cmpdtv.hh

Variable	fa(i)	Type	Valve Type	Description
abelan	31	R	relief	
abelin	32	R	relief	
adann	34	R	relief	
adisk	30	R	relief	
aflapr	28	R	inertial	Valve disk area
aleak	18	R	check	Normalized area of valve leakage
areaxx	18	R		
arings	33	R	relief	
atlast	14	R	all	Normalized full open valve area
bdamp	23	R	relief	
clstrp	19	I	motor	Close trip number
clstrp	19	I	servo	Used for control variable information
cvtbl	28	R	motor	Table entries (sets of three starting with normalized stem position, fjunf, fjunr)
drings	22	R	relief	
dseat	21	R	relief	
hring1	35	R	relief	

**Table 10.4-1** Valve Component Variable Names in Comdeck cmpdtv.hh

<b>Variable</b>	<b>fa(i)</b>	<b>Type</b>	<b>Valve Type</b>	<b>Description</b>
hring2	36	R	relief	
lnglv	25	R	inertial	Length of valve disk moment arm
maxtht	22	R	inertial	Maximum valve disk angular position
mintht	21	R	inertial	Minimum valve disk angular position
moment	16	R	inertial	Mass moment of inertia of valve disk
ncvtbl	25	R	motor	Variable used for table interpolation as follows: unused (15 bits), number of items/entry (15 bits), total number of items (15 bits), current subscript (15 bits)
omega	23	R	inertial	Valve disk angular velocity
omegao	24	R	inertial	Valve disk angular velocity, old-time
opntrp	17	I	motor	open trip number
opntrp	17	I	servo	Control variable number
pbellw	17	R		
pcv	17	R	check	Back pressure to close valve
pcv	17	R	inertial	Valve hinge cracking delta pressure used for coulomb damping torque
rdsvlv	27	R	inertial	Valve disk radius
sprcon	24	R	relief	
tblnum	21	R		
theta	19	R	inertial	Valve disk angular position
thetao	20	R	inertial	Valve disk angular position, old-time
velvl	27	R	relief	
velvlo	28	R	relief	
vlfmas	26	R		
vlsetp	25	R	relief	
vlstm	15	R	motor	Normalized stem position
vlstm	15	R	servo	Normalized stem position, controlled by control variable
vlstmo	16	R	motor	Normalized stem position, old-time

**Table 10.4-1** Valve Component Variable Names in Comdeck cmpdtv.hh

<b>Variable</b>	<b>fa(i)</b>	<b>Type</b>	<b>Valve Type</b>	<b>Description</b>
vlstmx	29	R	relief	
vlvcon	24	R	motor	Multiplier for converting csubv to form loss
vlvnm	13	I	all	Valve type
vlvslp	23	R	motor	Rate of change of normalized stem position
vlvtrp	19	I	trip	Trip number

## 10.5 cnvtpa.hh & cnvtpad.hh

The cnvtpa.hh comdeck is a specification block for the control system operators, e.g. sum, mult, etc. It contains the specification statements for character string array, cnvtpa, that contains the names of these variables. The cnvtpad.hh comdeck is the data deck that initializes this cnvtpa array. The names assigned to each position in the array are given below. They are sorted alphabetically by name.

**Table 10.5-1** Control System Operator Names in Comdeck cnvtpa.hh

<b>Name</b>	<b>Index</b>
constant	18
delay	7
diffrend	5
diffreni	4
div	3
feedctl	22
function	8
integral	6
lag	16
lead-lag	17
mult	2
poweri	12
powerr	13
powerx	14
prop-int	15
pumpctl	20

**Table 10.5-1** Control System Operator Names in Comdeck cnvtpa.hh

Name	Index
shaft	19
stdfnctn	9
steamctl	21
sum	1
tripdlay	11
tripunit	10

## 10.6 comctl.hh & comctlc.hh

The comctl.hh comdeck is a specification block and /comctl/ common block. It is the main control block for storage in RELAP5. It contains the file identification numbers for all the dynamic storage blocks that are used in RELAP5. The comctlc.hh comdeck contains a description of each of the variable names used in the comctl.hh comdeck. The descriptions of the variables are given below in alphabetical order. The dimensions and type of the variable, real\*8, integer, logical, or character, are given in the type column.

**Table 10.6-1** Control Block Variable Names in Comdeck comctl.hh

Variable	Type	Description
comdat	I(ncoms)	Index relative to fa of first word of common block to be saved
comdln	I(ncoms)	Length of common block to be saved
filflg	I(ncoms+1)	Flag for dynamic file on disk to be written at complete restart (bit 1) Flag to write common block file at complete restart (bit 4)

**Table 10.6-1** Control Block Variable Names in Comdeck comctl.hh

Variable	Type	Description
filid	R(nfiles)	FTB filid for dynamic storage files: (1) Input data and work scratch space during advancement (2) Time step control block (3) Component description block (4) Hydrodynamic volumes block: (5) Hydrodynamic junctions block (6) Thermodynamic property file (7) Interactive input and output variable storage (8) Heat structure geometry and temperature block (9) Heat structure material property storage (10) Table of inlet and outlet junctions for each volume (11) General table storage (12) Minor edit file (13) Time dependent volumes and junctions pointer (14) Table of heat structures and data for each volume (15) Plot heading and control information (16) Minor edit control, save area, and labels (17) Plot record buffer (18) Trip block (19) Internal plot file control information (20) File for statistics during advancement (21) Reactor kinetics data (22) 2-D plot requests and specifications (23) Plot comparison data tables (24) Steady-state block for statistics counters and uo (25) Volume data needed for a moving system (26) Plot data for the internal plot subroutines (27) Control system block (28) Component indices in normal (input) order (29) Tabular data for rotations and translations for moving problem (30) Hydrodynamic system control information (31) Code coupling data (32) Reflood rezoning model storage space (33) User supplied plot record variable requests (34) Fission product data (35) Fixed list vectors (36) SCDAP data (37) Steady-state initialization check file (38) File for radiation heat transfer (40) File for sparse matrix strategy (43) File for level model geometry description ( i.e. stacks)
filndx	I(0:nfiles)	Index of dynamic file in /fast/ or ftblcm block
filsiz	I(nfiles)	Length of dynamic file

**Table 10.6-1** Control Block Variable Names in Comdeck comctl.hh

<b>Variable</b>	<b>Type</b>	<b>Description</b>
ncoms	I	Parameter: number of common control slots (initialized in a parameter statement to 104)
newrst	L	True if a restart problem, used during input processing (equivalenced to newrst)
nfiles	I	Parameter: number of dynamic file slots (initialized in a parameter statement to 50)
safe1	R	Not written on restart file, provided to allow length checking when reading from restart file Used for timer argument when timing transient subroutine execution times (equivalenced to newrst)

## 10.7 comlst.hh

The comlst.hh comdeck is a specification block and /comlst/ common block. It contains the names of the common blocks that are written to the restart and dump files in RELAP5. These names are initialized in the gninit subroutine. There are 6 RELAP5 common blocks that are written to the restart and dump files: /comctl/, /contrl/, /statec/, /k3all/, /k3point/, and /lvel/. There are many SCDAP common blocks also written to the restart and dump files. The list of common blocks that were written to the restart and dump files is built dynamically at run time and is done in such a way that common blocks can easily be added to or removed from the list.

## 10.8 cons.hh

The cons.hh comdeck is a specification block and /cons/ common block. It contains the names of some constants that are used in RELAP5. These constants are initialized in the gninit subroutine to the values shown in the table below. The dimensions and type of the variable, real\*8, integer, logical, or character, are given in the type column.

**Table 10.8-1** Constants Variable Names in Comdeck cons.hh

<b>Variable</b>	<b>Type</b>	<b>Value</b>
pi	R	3.14159265359
e	R	5.0e-7
twopi	R	2.0 * pi
piovr2	R	pi * 0.5
piovr4	R	pi * 0.25
sqrtpi	R	sqrt(pi)
sqrt2	R	sqrt(2.0)

**Table 10.8-1** Constants Variable Names in Comdeck cons.hh

Variable	Type	Value
sqrt2p	R	sqrt2 * sqrtpi
mcme17	I	None

## 10.9 contrl.hh & contrx.hh

The contrl.hh comdeck is a specification block and /contrl/ common block. It is the main control block for many of the logical decisions made in RELAP5. The contrx.hh comdeck contains a description of each of the variable names used in the contrl.hh comdeck. The descriptions of the variables are given below in alphabetical order. The dimensions and type of the variable, real\*8, integer, logical, or character, are given in the type column.

**Table 10.9-1** Main Control Block Names in Comdeck contrl.hh

Variable	Type	Description
aflag	L	Set true in dtstep when heat structures are to be advance Checked in htadv
chngno	L(90)	Flags for card 1 options
count	R	Real value of the integer ncount
cpurei	I(5)	Integer values (5) of the real cpurem (1=>3) not used (4) word 4 from 105 card = ncount1, ncount for starting diagnostic edits = -1, write to dumpfil1 for ncount2 = cpurei(5), stop = -2, write to dumpfil1 for ncount2 = cpurei(5), redo time step, write to dumpfil2, stop = -3, set in dtstep for -2 case. (5) word 5 from 105 card = ncount2, ncount for terminating diagnostic edits (equivalenced to cpurem)
cpurem	R(5)	Contains CPU remaining times values and advancement counts to start/stop diagnostic edit from input (equivalenced to cpurei)
done	I	Integer flag indicating state of transient. 0 = advancements are to continue, !=0 = advancements are to be terminated
dt	R	Current time step
dtht	R	Heat structure time step, currently the same as dthy
dthy	R	Hydrodynamic time step requested by user

**Table 10.9-1** Main Control Block Names in Comdeck contrl.hh

<b>Variable</b>	<b>Type</b>	<b>Description</b>
dtn	R	Time step limit due to material transport stability limit (Courant limit)
emass	R	Current mass error
emasso	R	Old mass error
errmax	R	Error estimate used in time step control
fail	L	Set to true if error encountered
gravcn	R	Gravitational constant (which may be set by input)
help	I	Used to control debug editing and termination
iextr2	I	
iextra	I	Old packed word, not used anymore
ihlppr	I(2)	Print control used by cards 2-5
imdctl	I(2)	Old packed word, not used anymore
integrationflag9	L	On-line selection of time integration flag
interactiveflag5	L	Interactive flag
iroute	I	Old packed word, not used anymore
isathenaopt10	L	Athena option
iscntrlsyschng6	L	Control system change
iscompleterestart4	L	Complete restart
isfissionprdctchng5	L	Fission product change
isheatcondchng1	L	Heat conduction change
ishydrochng0	L	Hydrodynamic change
isimplicithttrnsfr6	L	Implicit heat transfer
islevelcrossing	L	Level crossing activated
islevelmodel0	L	Level model activated
ismajoreditenabled1	L	Major edit enabled
isminoreditenabled2	L	Minor edit enabled
isplotenabled3	L	Plot enabled
isplotsqzflg12	L	Plot record squoz flag
ispltrcrdchng3	L	Plot record change

**Table 10.9-1** Main Control Block Names in Comdeck contrl.hh

<b>Variable</b>	<b>Type</b>	<b>Description</b>
isprntaccum0	L	Activate accum print block
isprntbrntrn1	L	Activate brntrn print block
isprntccfl2	L	Activate ccfl print block
isprntchfcal3	L	Activate chfcal print block
isprntconden4	L	Activate conden print block
isprntcontrol21	L	Activate control print block
isprntdittus5	L	Activate dittus print block
isprnteqfinl6	L	Activate eqfinl print block
isprntfsnprdtr20	L	Activate fsnprdtr print block
isprntfwdrag7	L	Activate fwdrag print block
isprntheatstr17	L	Activate heatstr print block
isprnht2tdp9	L	Activate ht2tdp print block
isprnhtffinl12	L	Activate htffinl print block
isprnhtrc113	L	Activate htrc print block
isprnhydro15	L	Activate hydro print block
isprntinput22	L	Activate input print block
isprntistate17	L	Activate istate print block
isprntjchoke18	L	Activate jchoke print block
isprntjprop19	L	Activate jprop print block
isprntjunction16	L	Activate junction print block
isprntmiedit23	L	Activate miedit print block
isprntnoncnd20	L	Activate noncnd print block
isprntphantj21	L	Activate phantomj print block
isprntphantv22	L	Activate phantomv print block
isprntpimplt23	L	Activate phantomv pimplt block
isprntpintfc24	L	Activate phantomv pintfc block
isprntpower14	L	Activate power print block
isprntprednb25	L	Activate prednb print block

**Table 10.9-1** Main Control Block Names in Comdeck contrl.hh

<b>Variable</b>	<b>Type</b>	<b>Description</b>
isprnppreseq26	L	Activate preseq print block
isprnppstdnb27	L	Activate pstdnb print block
isprnqfmove28	L	Activate qfmove print block
isprntradht18	L	Activate radht print block
isprntreflood19	L	Activate reflood print block
isprntsimplt29	L	Activate simplt print block
isprntstacc0	L	Activate stacc print block
isprntstate1	L	Activate state print block
isprntstatep2	L	Activate statep print block
isprntsuboil3	L	Activate suboil print block
isprntsysisitr4	L	Activate sysitr print block
isprntrip13	L	Activate trip print block
isprnttstate6	L	Activate tstate print block
isprntvalve7	L	Activate valve print block
isprntvexplt8	L	Activate vexplt print block
isprntvfinl9	L	Activate vfinl print block
isprntvimplt10	L	Activate vimplt print block
isprntvlvela11	L	Activate vlvela print block
isprntvolume15	L	Activate volume print block
isprntvolvel12	L	Activate volvel print block
isradiationchng4	L	Radiation change
isrestartenabled0	L	Restart enable bit
isrkncchnng2	L	Reactor kinetics change
isscdapopt11	L	Scdap is active
istwostepflag7	L	Two step method
nany	I	Number of mass error messages remaining to be issued
ncase	I	Case number Initially = -1 in blkdta Set to zero if last case of series

**Table 10.9-1** Main Control Block Names in Comdeck contrl.hh

Variable	Type	Description
ncount	I	Count of number of advancements, successful or otherwise
nmechk	L	True if no mass error check requested on time step card (no 1 bit on time step card) (equivalenced to safe2)
nrepet	I	Number of hydrodynamic advancements at current dt to finish a requested time step of dthy
nstsp	I	Number of standard advancements
pageno	I	Page number
print	I	Old packed word, not used anymore
problemopt61	I	Problem option (see ityp2 in inputd) 1 = stdy-st (1 and 2 are used for new and restart) 2 = transnt 3 = binary (3 and 4 are used for strip)
problemtype05	I	Problem type (see ityp1 in inputd) 1 = new 2 = restart 3 = RELAP5 internal plots (not used) 5 = strip 6 = cmpcom (compare dump records)
rstblknumber1531	I	Restart block number
rwtrstpltcmprssflag13	L	
safe2	R	Same purpose as safe1 (equivalenced to nmechk)
skipt	L	Logical flag used to control first entry processing in subroutine dtstep
stdtrn	R	Steady-state - transient flag 0.0 = steady-state 1.0 = transient
stdystatetermflg8	L	Steady-state termination
succes	I	Integer flag indicating success of the advancement. 0 = No need to repeat advancement with reduced time step 1 = Excessive truncation error 2 = Water property error 3 = Non-diagonal matrix 4 = Metal appears
testda	R(20)	Data array for minor edits and plotting during debug. Temporary coding is used to load data in this array and scnreq accepts testda as a legal request

**Table 10.9-1** Main Control Block Names in Comdeck contrl.hh

Variable	Type	Description
timeht	R	Problem time for heat structure advancements
timehy	R	Problem time for hydrodynamic advancements
tmass	R	Total mass of water in system currently
transrotatflag14	L	Transient rotation flag, 0.0 = steady-state 1.0 = transient
tmasso	R	Total mass of water in system at time = 0.0
uniti	L	Units for input True = SI units False = British units
unito	L	Units for output True = SI units False = British units

## 10.10 convarc.hh & convarx.hh

The convarc.hh comdeck is a dynamic block for control variables. It contains the specification statements for variables that are common to all the control variables. The convarx.hh comdeck contains a description of each of the variable names used in the convarc.hh comdeck. The descriptions of the variables are given below in alphabetical order where the slot number in the fa array that the variable is equivalenced to is given in the fa(i) column. The dimensions and type of the variable, real\*8, integer, logical, or character, are given in the type column.

**Table 10.10-1** Control Variable Names in Comdeck convarc.hh

Variable	fa(i)	Type	Description
cnavlf	1	R	Alphanumeric part of variable request code
cnavrn	11	R	New value of control variable
cnavro	12	R	Old value of control variable
cndct	21	I(3)	1: Last entered table position 2: Number of table positions 3: Delay table position
cndft	20	R	Slope for delay interpolation
cndla		R	Delay time
cndli	19	R	Delay time divided by number of hold values

**Table 10.10-1** Control Variable Names in Comdeck convarc.hh

<b>Variable</b>	<b>fa(i)</b>	<b>Type</b>	<b>Description</b>
cvgengen	18	I(2)	1: Table number input 2: Offset during input processing, index during transient
cnvint	15	I	Integer part of variable request code
cnylen	7	I	Number of words in component.
cnvmax	10	R	Maximum value permitted component
cnvmin	9	R	Minimum value permitted component
cnvnam	5	R	Control component alphanumeric name
cnnop	8	I	Initialization flag (1 bit) Initialization type,(2 bit) Minimum flag (4 bit) Maximum flag (8 bit) Generator flag ( 16 bit) Units (32 bit) Trip complement (64 bit) Standard function type (shifted left 24 bits)
cnnpa	6	I	Number of terms or factors
cnnnum	1	I	Number of components
cnnxt	24	R	Time to store next value
cnvold	18	R	Old values needed in difference or pid component blocks
cnopt	2	I	Units (1 bit) Flag for maximum number of components possible (2 bit) Generator present flag (4 bit)
cnpck	16	I(2)	1: Block number 2: Offset within block during input processing, index during transient
cnpnm	3	I	Component number
cnsan	18	R	“A” factor in sum or pid control components
cnscl	13	R	Scale factor

**Table 10.10-1** Control Variable Names in Comdeck convarc.hh

<b>Variable</b>	<b>fa(i)</b>	<b>Type</b>	<b>Description</b>
cnvscp	17	I(4)	1: Connected component type 2: Connected component number 3: Block number 4: Offset during input processing, index during transient
cnvscf	15	I(2)	1: Control variable number for torque 2: Offset during input processing, index during transient
cnvsfr	14	R	Friction factor for shaft component
cnvsin	13	R	Inertia for shaft component
cvtbl	25	R	Delay table values
cnvtrp	14	I(2)	1: Trip number 2: Offset within block to trip during input processing, index during transient
cnytyp		I	Control component type
gendtr	19	I(2)	1: Generator disconnect trip number 2: Offset during input processing, index during transient
generatorflag4	15	L	
genfr	24	R	Generator friction factor
genint	23	R	Generator inertia
genpow	26	R	Generator power input
gensvl	22	R	Generator synchronous rotational velocity
gentrp	17	I(2)	1: Generator trip number 2: Offset during input processing, index during transient
gentrq	25	R	Generator torque
genvel	21	R	Generator rotational velocity
initflag0	11	L	
inittype1	12	L	
isgeneratorflag2	4	L	
ismaxcomponents1	3	L	
maximumflag3	14	L	

**Table 10.10-1** Control Variable Names in Comdeck convarc.hh

Variable	fa(i)	Type	Description
minimumflag2	13	L	
stdfncttype2431	10	I	
tripcomplement6	17	L	
units0	2	L	
units5	16	L	

## 10.11 eccmxc.hh & eccmxcc.hh

The eccmxc.hh comdeck is a dynamic block for ECC mixer variable. It contains the specification statements for variable that is common to the ECC mixer. The eccmxcc.hh comdeck contains a description of the variable name used in the eccmxc.hh comdeck. The descriptions of the variables are given below in alphabetical order where the slot number in the fa array that the variable is equivalenced to is given in the fa(i) column. The dimensions and type of the variable, real\*8, integer, logical, or character, are given in the type column.

**Table 10.11-1** ECC Mixer Component Variable Name in Comdeck eccmxc.hh

Variable	fa(i)	Type	Description
cmpphi	13	R	injection angle of ECC

## 10.12 fast.hh & fastc.hh

The fast.hh comdeck is a specification block and /fast/ common block. It is used for all the dynamic storage that is allocated by the FRB package in RELAP5. This comdeck also contains the parameter that sets the length of the fa array, lfsiz. The fastc.hh comdeck contains a description of the fast.hh comdeck. The descriptions of the variables are given below in alphabetical order. The dimensions and type of the variable, real\*8, integer, logical, or character, are given in the type column.

**Table 10.12-1** /fast/ Storage Pool Variable Names in Comdeck fast.hh

Variable	Type	Description
fa	R(lfsiz)	Dynamic pool area (equivalenced to ia)
ia	I(lfsiz)	Dynamic pool area (equivalenced to ia)
lfsiz	I	Parameter, size of fa array Initialized to 2200000

## 10.13 flood.hh

The flood.hh comdeck is a specification block and /flood / common block. It is used for the variables that are common to the reflood heat transfer subroutines. The descriptions of the variables are given below in alphabetical order. The dimensions and type of the variable, real\*8, integer, logical, or character, are given in the type column.

**Table 10.13-1** Reflood Variable Names in Comdeck flood.hh

Variable	Type	Description
hmac	R	
hmic	R	
itrchf	I	Iteration counter for finding twchf
onrefl	I	1: If reflood logic is currently active, 0: If reflood logic is currently inactive
refbun	I	1: If heat structure is a bundle and reflood may exist 0: otherwise
twchf	R	
twnvg	R	
twqf	R	
wetbot	R	Height of bottom quench region
wettop	R	Length of top quench region
zbun	R	Bundle length
zqf	R	Height of the node above the bottom quench front
zqftop	R	Distance of the node from the top quench front

## 10.14 ftbcom.hh

The ftbcom.hh comdeck is a specification block and /ftb / common block. It is used for FTB package variables. The descriptions of the variables are given below in alphabetical order. The dimensions and type of the variable, real\*8, integer, logical, or character, are given in the type column.

**Table 10.14-1** FTB Package Variable Names in Comdeck ftbcom.hh

Variable	Type	Description
dlt	L	Logical flag to control link shifting, false implies that links are shifted as far as possible

**Table 10.14-1** FTB Package Variable Names in Comdeck ftbcom.hh

<b>Variable</b>	<b>Type</b>	<b>Description</b>
dly	L	Logical flag that is set when a write is issued to force a check before deleting the buffer
dpn	L(4)	Logical flag to check for open files on units 3-6
first	I	Integer flag, if non-zero, indicates that the FTB subroutines have been initialized. Initialized to 0 in ftbftb and set to iftb after the call to ftbftb. iftb = 12345 in a data statement in ftbint
ftbun	I(5)	Only on Cray. disk unit numbers 21-25, set in ftbftb
hilo	L	Logical flag indicating preferred end of SCM True if preferred end is the high end (RELAP5 used this end) False if preferred end is the low end
ireclt	I(5)	Buffer size on units 3-7 (disk units)
lasdes	I	Address of last described file
link1	I	Address of first link table (RELAP5 uses only one link)
maxz	I(7)	Address of last available word in pool on this unit fa(maxz(i) - 1) = last location we can store information into
minz	I(7)	Address of first available word in pool on this unit fa[minz(i)] = first location we can store information into
ndsk2	I	Number of disk units defined (set to 7 in ftbftb)
nexdes	I	Address of location to place next file description
nofils	I	Number of file descriptions in the current link table
nolink	I	Number of link tables
reclim	I	Length of buffer for disk files, initialized to 1024 in ftbftb
size	I(7)	Number of words available on each unit
szz	I(8)	Initial space on each unit On units 1, 2, and 8, szz is initialized to 0 in ftbftb On units 3-7, szz is initialized to 100,000,000 in ftbftb

## 10.15 gapvar.hh

The gapvar.hh comdeck is a specification block and /gapvar/ common block. It is used for a couple of fuel pin gap variables. The descriptions of the variables are given below in alphabetical order. The dimensions and type of the variable, real\*8, integer, logical, or character, are given in the type column.

**Table 10.15-1** Fuel Pin Gap Variable Names in Comdeck gapvar.hh

Variable	Type	Description
pfluid	R	Fluid number of the gas in the fuel pin gap
pgas	R	Pressure in the gas in the fuel pin gap

## 10.16 genrl.hh & genrlc.hh

The genrl.hh comdeck is a specification block and /genrl/ common block. The /genrlc/ common block contains a description of the variables in the /genrl/ common block. It holds the character strings for the title of the problem case that is being run and the code version. A description is given below of these variables in alphabetical order. The dimensions and type of the variable, real\*8, integer, logical, or character, is given in the type column..

**Table 10.16-1** Problem Title Names in Comdeck genrl.hh

Variable	Type	Description
ctitle	C*108	Contains title card of case, time, and date
ptitle	C*64	Contains program version identification and title

## 10.17 gentblc.hh & gentblx.hh

The gentblc.hh comdeck is a dynamic block for general table variables. It contains the specification statements for variables that are common to the general tables. The gentblx.hh comdeck contains a description of the variable names used in the gentblc.hh comdeck. This description is given below where the slot number in the fa array that the variable is equivalenced to is given in the fa(i) column. The dimensions and type of the variable, real\*8, integer, logical, or character, is given in the type column.

**Table 10.17-1** General Table Variable Names in Comdeck gentblc.hh

Variable	fa(i)	Type	Description
gtarg	5	R	Argument for last table lookup
gtbl	11	R	Table values

**Table 10.17-1** General Table Variable Names in Comdeck gentblc.hh

<b>Variable</b>	<b>fa(i)</b>	<b>Type</b>	<b>Description</b>
gtbptr	1	I	Table error flag (1 bit) Not used (17 bits) Table length, (12 bits) Table number (12 bit, Table offset (18 bits)
gtinfo	8	I	Table type (15 bits) Number of items/entry (15 bits) Total number of items (15 bits) Current subscript (15 bits)
gtlen	2	I	
gtnum	1	I	
gttrp	3	I	Trip number (12 bits) Unused (12 bits) Trip location, relative to trip block (18 bits) Trip location relative to dynamic block (18 bits)
gttyp	3	I	
gtval	6	R	Result from last table lookup
ngtbls	1	I	Number of general tables

## 10.18 htrcom.hh & htrcomc.hh

The htrcom.hh comdeck is a specification block and /htrcom/ common block. This common block holds that variables that are used in the heat transfer correlations. The /htrcomc/ common block contains a description of the variables in the /htrcom/ common block. A description is given below of these variables in alphabetical order. The dimensions and type of the variable, real\*8, integer, logical, or character, is given in the type column. In the description column, input means that the value is set before calling hrc1, computed means that it is set inside hrc1 and is for communication to lower level subroutines, and output means that the quantity is returned by hrc1.

**Table 10.18-1** Heat Transfer Correlation Variable Names in Comdeck htrcom.hh

<b>Variable</b>	<b>Type</b>	<b>Description</b>
alpf10	R	
alpg10	R	
axpf	R	Axial power peaking factor (used for chf) (input)
beta	R	Coefficient of thermal expansion of appropriate phase (computed)

**Table 10.18-1 Heat Transfer Correlation Variable Names in Comdeck htcom.hh**

<b>Variable</b>	<b>Type</b>	<b>Description</b>
chf	R	Critical heat flux (output)
chfmul	R	Critical heat flux table lookup multiplier (output) or critical heat flux ratio (used for pg chfr)
cps	R	Specific heat of appropriate phase (computed)
delgrv	R	
drod	R	Rod outer diameter
dtsat	R	Wall temperature minus saturation temperature (computed)
enliq	R	Liquid enthalpy (computed)
fstrat	R	Percent of liquid stratification (computed)
g	R	Total cell centered mass flux (input)
gabs	R	Absolute value of total centered mass flux (input)
gamw	R	Direct wall flashing or condensation (output)
gcrossf	R	
gcross	R	
ggasa	R	Cell centered vapor mass flux (input)
gliqa	R	Cell centered liquid mass flux (input)
gridk	R	Critical heat flux ratio (used for pg chfr) spacer pressure loss coefficient (used for chf) (input) or grid spacer factor (used for pg chfr)
gridz	R	Heated channel inlet equilibrium quality based on total pressure (used for pg chfr)
hd	R	
hfg	R	Heat of vaporization (enthalpy difference between saturated vapor and saturated liquid) (input)
hfgp	R	Hfg based on total pressure (computed)
htcf	R	Heat transfer coefficient to liquid (output)
htcg	R	Heat transfer coefficient to vapor (output)
htcnon	R	
htcoef	R	Total heat transfer coefficient (output)
htcond	R	
htdiam	R	Heated equivalent diameter (input)

**Table 10.18-1 Heat Transfer Correlation Variable Names in Comdeck htrcom.hh**

<b>Variable</b>	<b>Type</b>	<b>Description</b>
htgamf	R	Factor to convert heat flux to interphase mass transfer due to flashing at wall
htgamg	R	Factor to convert heat flux to interphase mass transfer due to condensing at wall
htlen	R	Heat transfer length from some position (used for chf) (input) or Reduced heat transfer length from some position (used for pg chfr) (input)
htlenc	R	Heat transfer length for natural convection (input)
htopta	I	Heat transfer package option number
htqof	R	Old-time heat flux to liquid
htqog	R	Old-time heat flux to vapor
htqot	R	Old-time total heat flux to fluid
htsa	R	heat transfer surface area (input)
htzhff	R	Heat transfer coefficient from wall to liquid using liquid temperature
htzhft	R	Heat transfer coefficient from wall to liquid using saturation temperature corresponding to total pressure
htzhgg	R	Heat transfer coefficient from wall to vapor using vapor temperature
htzhgp	R	Heat transfer coefficient from wall to vapor using saturation temperature corresponding to partial pressure
htzhgt	R	Heat transfer coefficient from wall to vapor using saturation temperature corresponding to total pressure
htzht	R	Total heat transfer coefficient from wall to fluid
ibundl	I	Rod bundle flag (computed)
incnd	I	
irwt	I	Vertical/horizontal flag (computed)
iv	I	Volume index (input)
iv1	I	
iv2	I	
ivindx	I	
mode	I	Mode of heat transfer (output)
pgflag	I	
pgopta	I	

**Table 10.18-1** Heat Transfer Correlation Variable Names in Comdeck htrcom.hh

<b>Variable</b>	<b>Type</b>	<b>Description</b>
pithier	R	Rod pitch/diameter. (input)
pr	R	
pvblk	R	
qffo	R	Heat flux to liquid (output)
qfgo	R	Heat flux to vapor (output)
qfgox	R	Left (1) Right (2) heat flux to vapor (output)
qfluxo	R	Total heat flux (output)
qualep	R	Equilibrium quality based on total pressure (computed)
rey	R	
rhos	R	
sathfp	R	Saturation liquid enthalpy based on total pressure (computed)
tf	R	Volume liquid temperature (computed)
thcons	R	Thermal conductivity of appropriate phase (computed)
tw	R	Wall surface temperature (input)
viscs	R	Viscosity of appropriate phase (computed)

## 10.19 htrflb.hh & htrflbc.hh

The htrflb.hh comdeck is a dynamic block for reflood variables. It contains the specification statements for variables that are common to reflood. The htrflbc.hh comdeck contains a description of the variable names used in the htrflb.hh comdeck. This description is given below where the slot number in the fa array that the variable is equivalenced to is given in the fa(i) column. The dimensions and type of the variable, real\*8, integer, logical, or character, is given in the type column.

**Table 10.19-1** Reflood Variable Names in Comdeck htrflb.hh

<b>Variable</b>	<b>fa(i)</b>	<b>Type</b>	<b>Description</b>
fines	12	R	Current number of axial nodes on a reflood heat structure
htdzlm	1	R	
htlenz	1	R	Axial mesh points coordinate
iglrfl	7	I(2)	1: Maximum number of axial levels 2: Current number of axial levels

**Table 10.19-1** Reflood Variable Names in Comdeck htrflb.hh

Variable	fa(i)	Type	Description
inxrfl	3	I(4)	1: Offset/index to heat structure dependent data 2: Offset/index to axial mesh point dependent data 3: Offset/index to temperature data 4: Offset/index to mesh point positioning data
lhtrfl	2	I	Offset to ihtptr for first heat structure in a heat structure designated for reflood
nhtga	-	I	initialized to 14 in parameter statement
nhtma	-	I	Initialized to 13 in parameter statement
nmzht	1	I	
nrfht	1	I	Number of heat structure-geometries designated for reflood-rezoning calculations
nscre1	-	I	Initialized to 13 in parameter statement
nscre2	-	I	Initialized to 5 in parameter statement
strgeo	15	I	Heat structure-geometry which reflood set belongs to on restart this word is set to 0 or negative if the heat structure is deleted or overlaid in rhtcmp
tchfqp	13	R	Temperature corresponding to qfchfn
tmprfn	1	R	New-time temperature array
tmprfo	1	R	Old-time temperature array
tmpsfn	2	R	
tmpsfo	1	R	
trewet	14	R	Quench, Leidenfrost or rewet temperature
zbunht	9	R	Height of reflood bundle
zqbot	10	R	Height of reflood bottom quenched region
zqtop	11	R	Height of reflood top quenched region

## 10.20 htscr1.hh

The htscr1.hh comdeck is a dynamic block for scratch storage of reflood scratch variables. It contains the specification statements for variables that are used for scratch storage during a reflood calculation. A description is given below where the slot number in the fa array that the variable is equivalenced to is

given in the fa(i) column. The dimensions and type of the variable, real\*8, integer, logical, or character, is given in the type column.

**Table 10.20-1** Reflood Scratch Variable Names in Comdeck htscr1.hh

Variable	fa(i)	Type	Description
indzhs	11	I	
htb2	3	R	
htc2	4	R	
htf2	5	R	
htts2	6	R	
httr2	2	R	
htdz	1	R	
htbc2	7	R	
htbd2	8	R	
htv1	9	R	
htv2	10	R	

## 10.21 htscr2.hh & htscr2c.hh

The htscr2.hh comdeck is another dynamic block for scratch storage of reflood scratch variables. It contains the specification statements for variables that are used for scratch storage during a reflood calculation. The htscr2c.hh comdeck contains a description of these variables. This description is given below where the slot number in the fa array that the variable is equivalenced to is given in the fa(i) column. The dimensions and type of the variable, real\*8, integer, logical, or character, is given in the type column.

**Table 10.21-1** Reflood Scratch Variable Names in Comdeck htscr2.hh

Variable	fa(i)	Type	Description
htvhc	1	R	1: Volumetric heat capacities for mesh intervals 2: Volume weighted volumetric heat capacities for mesh points
httc2	2	R	
aijr	2	R	1: Thermal conductivities for mesh intervals 2: Thermal conductivities divided by two for mesh intervals 3: Mesh point temperatures during mesh renodalization 4: Radial off-diagonal element
htpws	3	R	Time dependent and volume weighted power term
aijt	4	R	Axial off-diagonal element

**Table 10.21-1** Reflood Scratch Variable Names in Comdeck htscr2.hh

<b>Variable</b>	<b>fa(i)</b>	<b>Type</b>	<b>Description</b>
tmpscr	5	R	Mesh point temperatures during actual advancement

## **10.22 htscr.hh**

The htscr.hh comdeck is another dynamic block for scratch storage of heat transfer scratch variables. It contains the specification statements for variables that are used for scratch storage during a the heat transfer advancement calculation. A description is given below where the slot number in the fa array that the variable is equivalenced to is given in the fa(i) column. The dimensions and type of the variable, real\*8, integer, logical, or character, is given in the type column.

**Table 10.22-1** Heat Transfer Scratch Variable Names in Comdeck htscr.hh

<b>Variable</b>	<b>fa(i)</b>	<b>Type</b>	<b>Description</b>
htb	28	R	
hte	27	R	
htee	29	R	
htf	30	R	
htflag	25	I	
htgsmf	1	R	
htgsmg	3	R	
hthhff	7	R	
hthhft	9	R	
hthhg	11	R	
hthhgp	15	R	
hthhgt	13	R	
hthht	5	R	
htqosf	17	R	
htqosg	19	R	
htqost	21	R	
htscrp	-	I	Initialized to 14 in a parameter statement
htsxrp	-	I	Initialized to 27 in a parameter statement
httc	39	R	

**Table 10.22-1 Heat Transfer Scratch Variable Names in Comdeck htscr.hh**

<b>Variable</b>	<b>fa(i)</b>	<b>Type</b>	<b>Description</b>
htvhc	40	R	
qradlr	23	R	

## **10.23 htsrcm.hh & htscrmc.hh**

The htsrcm.hh comdeck is another dynamic block for scratch storage of heat structure variables. It contains the specification statements for variables that are used for heat structure storage during a the heat transfer advancement calculation. The htscrmc.hh comdeck contains a description of these variables. This description is given below where the slot number in the fa array that the variable is equivalenced to is given in the fa(i) column. The dimensions and type of the variable, real\*8, integer, logical, or character, is given in the type column.

**Table 10.23-1 Heat Structure Variable Names in Comdeck htsrcm.hh**

<b>Variable</b>	<b>fa(i)</b>	<b>Type</b>	<b>Description</b>
apfacn	99	R	Axial power peaking factor, right-hand side of heat structure
apfaco	98	R	Axial power peaking factor, left-hand side of heat structure (local flux/average flux from start of boiling to local point)
arean	2	R	Surface weight at right boundary
areao	1	R	Surface weight at left boundary
chfmun	71	R	Critical heat flux multiplier, right-hand side
chfmuo	70	R	Critical heat flux multiplier, left-hand side
chnglosscoeff12	111	L	
cladex	116	R	Clad expanded outer radius (negative if it has burst)
coordcodeoroffset1215	34	I	
correlationnum15	114	L	
gapwd	115	R	Gap width for the gap conductance and ballooning case
gpintp	3	R	Gap initial pressure at input, initial pressure / temperature of reference volume (top of the core) after initialization

**Table 10.23-1 Heat Structure Variable Names in Comdeck htsrcm.hh**

<b>Variable</b>	<b>fa(i)</b>	<b>Type</b>	<b>Description</b>
gprinc	4	R	Radial interval width up to the gap plus inner and outer radii of cladding
gprouf	82	R	Fuel surface roughness + cladding surface roughness
gpudis	83	R	Cladding creep down radial displacement - radial displacement due to fission gas induced fuel swelling and densification
grdkfn	95	R	Grid spacer forward form loss coefficient, right-hand side
grdkfo	94	R	Grid spacer forward form loss coefficient, left-hand side
grdkrn	97	R	Grid spacer reverse form loss coefficient, right-hand side
grdkro	96	R	Grid spacer reverse form loss coefficient, left-hand side
grdzfn	91	R	Forward direction length from grid spacer, right-hand side or for pg chfr: forward direction grid spacer factor, right-hand side
grdzfo	90	R	Forward direction length from grid spacer, left-hand side or for pg chfr: forward direction grid spacer factor, left-hand side
grdzrn	93	R	For pg chfr: reverse direction grid spacer factor, right-hand side
grdzro	92	R	Reverse direction length from grid spacer, left-hand side or for pg chfr: reverse direction grid spacer factor, left-hand side
h2gen	95	R	New hydrogen generated (kg)
h2geno	96	R	Old value of hydrogen generated (kg)
hetrat	91	R	Is the cladding heatup rate
htavwt	6	R	Volume weights for average volume temperature calculation at each mesh point
htbcan	69	R	Right boundary condition value, usually heat transfer coefficient
htbcao	68	R	Left boundary condition value, usually heat transfer coefficient

**Table 10.23-1 Heat Structure Variable Names in Comdeck htsrcm.hh**

<b>Variable</b>	<b>fa(i)</b>	<b>Type</b>	<b>Description</b>
htbccn	73	R	Right boundary condition value, usually sink temperature related quantity or heat flux
htbcco	72	R	Left boundary condition value, usually sink temperature related quantity or heat flux
htbnt	26	I	Boundary type input
htbntr	28	I	Reduced boundary type
htbnts	-	I	Old packed word, not used anymore
htbvc	22	I	Boundary volume input code
htbvo	24	I	Boundary volume offset during input, index during transient or general table offset
htcffn	53	R	Heat transfer coefficient to liquid at right boundary (equivalence to htrnsn)
htcffo	52	R	Heat transfer coefficient to liquid at left boundary (equivalence to htrnso)
htcfgn	81	R	(Equivalenced to htrgon)
htcfgo	80	R	(Equivalenced to htrgoo)
htchfn	64	R	Right side critical heat flux
htchfo	63	R	Left side critical heat flux
htcmp	7	I(2)	1: Composition number 2: Composition offset
htcmpf	7	R	
htdt	60	R	Time increment
htdtmn	67	R	Right side old (twall - twater)
htdtmo	66	R	Left side old (twall - twater)
htfcctr	47	R	Source multiplier
htfftr	129	R	
htfixa	-	I	Number of mesh size independent words for each heat structure Initialized to 143 in a parameter statement
htftrn	55	R	Right direct source factor
htftro	54	R	Left direct source factor

**Table 10.23-1 Heat Structure Variable Names in Comdeck htsrcm.hh**

<b>Variable</b>	<b>fa(i)</b>	<b>Type</b>	<b>Description</b>
htgap	40	I(2)	1: Gap deformation model reference volume number 2: Gap deformation model reference volume offset
htgmr	18	I	Heat structure-geometry referral number
htgom	19	I	Offset to heat structure geometry data
htgskp	-	I	Skip Factor for geometry data Initialized to 8 in parameter statement
hthdmn	62	R	Right side heated equivalent diameter
hthdmo	61	R	Left side heated equivalent diameter
htimeo	59	R	Time at beginning of advancement
htiscr	100	I	Index for scratch space during transient advancement
htivfc	131	I(2)	1: Left and 2: Right forward direction boundary volume input code to define equilibrium quality based on total pressure at the inlet to heated channel
htivfo	133	I(2)	1: Left and 2: Right forward direction boundary volume offset index during transient to define equilibrium quality based on total pressure at the inlet to heated channel
htivrc	135	I(2)	1: Left and 2: Right reverse direction boundary volume input code to define equilibrium quality based on total pressure at the inlet to heated channel
htivro	137	I(2)	1: Left and 2: Right reverse direction boundary volume offset index during transient to define equilibrium quality based on total pressure at the inlet to heated channel
htlncf	125	R(2)	1: Left and 2: Right forward natural convection length from entrance
htlnfn	87	R	Forward direction heated length from inlet, right-hand side or for pg chfr: forward direction reduced heated length from inlet

**Table 10.23-1 Heat Structure Variable Names in Comdeck htsrcm.hh**

<b>Variable</b>	<b>fa(i)</b>	<b>Type</b>	<b>Description</b>
htlnfo	86	R	Forward direction heated length from inlet, left-hand side or for pg chfr
htlnrn	89	R	Reverse direction heated length from inlet, right-hand side or for pg chfr: reverse direction reduced heated length from inlet
htlnro	88	R	Reverse direction heated length from inlet, left-hand side or for pg chfr: reverse direction reduced heated length from inlet
htlvwt	11	R	Left volume weights at each mesh interval
htmod	38	R	Heat transfer mode number during transient. This quantity and the preceding quantity occupy the same location
htnaxl	45	I	Axial rezoning limit number
htnmpt	42	I	Number of mesh points
htnusr	46	I	Number of heat structures with this geometry
htopt	-	I	Old packed word, not used anymore
htpovd	127	R(2)	1: Left and 2: Right rod bundle pitch to diameter ratio
htpown	57	R	New power value
htpowo	56	R	Old power value
htradn	85	R	Radius at right boundary
htrado	84	R	Radius at left boundary
htrflg	43	I(2)	1: Reflood flag 2: Offset for trip when trip number in htrflg(1)
htrfnn	75	R	
htrfno	74	R	
htrfon	77	R	
htrfoo	76	R	
htrfpt	143	R	
htrgnn	79	R	
htrgno	78	R	
htrgon	81	R	(Equivalenced to htcfgn)

**Table 10.23-1 Heat Structure Variable Names in Comdeck htsrcm.hh**

<b>Variable</b>	<b>fa(i)</b>	<b>Type</b>	<b>Description</b>
htrgoo	80	R	(Equivalenced to htcfgo)
htrnrrn	51	R	Heat transfer rate at right boundary (new)
htrnro	50	R	Heat transfer rate at left boundary (new)
htrnsn	53	R	Heat transfer rate at right boundary (old) (equivalence to htcffn)
htrns0	52	R	Heat transfer rate at left boundary (old) (equivalence to htcffo)
htrvwt	10	R	Right volume weights at each mesh interval
htsrc	5	R	Space dependent source factor times volume weights at each mesh point
htsrfn	49	R	Area at right boundary
htsrfo	48	R	Area at left boundary
htsrt	20	I(2)	1: Heat source type 2: Heat source offset for general table or control system power source; heat structure-geometry referral number for initial temperatures
htsrwt	9	R	Surface weights at each mesh interval
htstno	1	I	Heat structure number
htstyp	38	I	Heat transfer surface type during input
htttmp	1	R	Temperature in heat structure
httots	58	R	Integral of the source distribution over space
htvatp	65	R	Volume averaged temperature
htxft	16	I	Offset to end of time step temperatures
htxit	17	I	Offset to beginning of time step temperatures
ihtptr	2	I	Offset to heat structure
imw	-	R	Old packed word, not used anymore
iscoordcode11	36	L	
islosscoeffchng10	109	L	
ismwrinnersurface13	112	L	
ismwrsetinput14	113	L	
isplasticstrain11	110	L	

**Table 10.23-1 Heat Structure Variable Names in Comdeck htsrcm.hh**

<b>Variable</b>	<b>fa(i)</b>	<b>Type</b>	<b>Description</b>
isruption9	108	L	
meshptnumber08	107	I	
nhtstr	1	I	Number of heat structures
optnumortblnum06	30	I	
oxti	101	R	New oxide thickness on inside of cladding (m)
oxtio	103	R	Old value of inside oxide thickness (m)
oxto	102	R	New oxide thickness on outside of cladding (m)
oxtoo	104	R	Old value of outside oxide thickness (m) (can be input as w2 on the 1cccg003 card in rhtcmp)
pecln	124	R	Left, right side Peclet numbers
peclo	123	R	Left, right side Peclet numbers
pgcall	141	I(2)	1: Left and2: right chf correlation call flag
pgopti	139	I(2)	1: Left and2: right chf correlation option
stantn	122	R	Left, right side Stanton numbers
stanto	121	R	Left, right side Stanton numbers
strnpl	118	R	Is the maximum plastic strain
subrnum0710	32	I	

## 10.24 intrac.hh & intracc.hh

The intrac.hh comdeck is a dynamic block for the interactive variables. It contains the specification statements for the interactive variables that can be displayed during a RELAP5 calculation. The intracc.hh comdeck contains a description of these variables. This description is given below where the slot number in the fa array that the variable is equivalenced to is given in the fa(i) column. The dimensions and type of the variable, real\*8, integer, logical, or character, is given in the type column.

**Table 10.24-1 Interactive Variable Names in Comdeck intrac.hh**

<b>Variable</b>	<b>fa(i)</b>	<b>Type</b>	<b>Description</b>
intrcv	3	R	Conversion factor to SI units
intrla	2	R	Variable name of input quantity or label of output quantity
intrni	5	I	Input variable card number

**Table 10.24-1** Interactive Variable Names in Comdeck intrac.hh

Variable	fa(i)	Type	Description
intrno	1	I	Number of interactive variables
intrva	4	R	Current value of input quantity

## 10.25 invhtb.hh & invhtbc.hh

The invhtb.hh comdeck is a dynamic block for the inverted heat structure table. It contains the specification statements for the inverted heat structure table that can be used to find all the heat structures attached to a given volume. The invhtbc.hh comdeck contains a description of these variables. This description is given below where the slot number in the fa array that the variable is equivalenced to is given in the fa(i) column. The dimensions and type of the variable, real\*8, integer, logical, or character, is given in the type column.

**Table 10.25-1** Inverted Heat Structure Table Variable Names in Comdeck invhtb.hh

Variable	fa(i)	Type	Description
frahaw	6	R	Ratio of heat structure surface area to total surface area of all heat structures attached to volume
frphpw	5	R	Ratio of volume hydraulic diameter to heat structure heated hydraulic diameter
inhtent	2	I	Number of heat structures surfaces attached to this volume
inhtno	3	I	Heat structure number
insrft	4	I	Index to surface temperature
invhkfp	-	I	Skip factor for each heat structures block Initialized to 4 in a parameter statement
invhos	1	I	Offset used to compute indexes stored in insrft

## 10.26 invtbl.hh & invtblc.hh

The invtbl.hh comdeck is a dynamic block for the inverted junction table. It contains the specification statements for the inverted junction table which can be used to find all the junctions attached to a given volume. The invtblc.hh comdeck contains a description of these variables. This description is given below where the slot number in the fa array that the variable is equivalenced to is given in the fa(i)

column. The dimensions and type of the variable, real\*8, integer, logical, or character, is given in the type column.

**Table 10.26-1** Inverted Junction Table Variable Names in Comdeck invtbl.hh

Variable	fa(i)	Type	Description
invct	2	I	Number of junctions connected to each volume
invjun	3	I	Old packed word, not used anymore
invofs	1	I	Offset used to compute indexes stored in invvnx
invskp	-	I	Initialized to 3 in a parameter statement
invvno	4	I	Position number of junction in junction block
invvnx	5	I	Index of junction
ismomfluxoff3	5	L	true if momentum flux is off
isoutletflag1	4	L	true if outlet
isreversecoordflg0	3	L	true if reverse coordinate direction
istoflag2	5	L	true if to connection

## 10.27 jundat.hh & jundatc.hh

The jundat.hh comdeck is a dynamic block for the junctions. It contains the specification statements for the junction variables. The jundatc.hh comdeck contains a description of these variables. This description is given below where the slot number in the fa array that the variable is equivalenced to is given in the fa(i) column. The dimensions and type of the variable, real\*8, integer, logical, or character, is given in the type column.

**Table 10.27-1** Junction Variable Names in Comdeck jundat.hh

Variable	fa(i)	Type	Description
ajun	9	R	Area of junction
ajuno	-	R	Old ajun. (SCDAP only) positioned at ijsk3 + 2
arat	11	R(2)	1: Mixture volumetric flow rate for the junction divided by the total mixture volumetric flow rate on that end of the volume. Mixture is obtained by using sum of absolute value of phasic volumetric flow rates 2: same as 1, except for "to" volume
athrot	10	R	Ratio of orifice area to junction area

**Table 10.27-1** Junction Variable Names in Comdeck jundat.hh

<b>Variable</b>	<b>fa(i)</b>	<b>Type</b>	<b>Description</b>
betacc	57	R	Form of CCFL correlation 0. = Wallis 1. = Kutateladze
c0j	60	R	Junction distribution coefficient
c0jo	61	R	Junction distribution coefficient previous time step
c0jos	85	R	Saved beginning of advancement phasic distribution coefficient for level model backup
chokef	82	R	Junction choking flag 0 = flow not choked 1 = flow choked
constc	58	R	Gas intercept for CCFL correlation
constm	59	R	Slope for CCFL correlation
diamj	13	R	Diameter of junction
diamjo	-	R	Old diamj. (SCDAP only) Positioned at ijsk3 + 3
extjnn	8888 9997 7:-	R(20)	Extra junction data for programmer use, one per junction nn goes from 01 to 20 positioned at ijsk5 + 2 => ijsk5 + 21
faaj	32	R	Virtual mass
fij	33	R	Interphase friction
fijo	34	R	Interphase friction previous time step
fijos	83	R	Saved beginning of advancement interfacial friction coefficient for level model backup
fjunf	27	R	Constant term for form loss coefficient for irreversible losses, forward
fjunfb	76	R	Multiplier term for form loss coefficient for irreversible losses, forward
fjunfc	77	R	Exponent term for form loss coefficient for irreversible losses, forward
fjunft	92	R	Total form loss coefficient for irreversible losses, forward
fjunr	28	R	Constant term for form loss coefficient for irreversible losses, reverse

**Table 10.27-1** Junction Variable Names in Comdeck jundat.hh

<b>Variable</b>	<b>fa(i)</b>	<b>Type</b>	<b>Description</b>
fjunrb	78	R	Multiplier term for form loss coefficient for irreversible losses, reverse
fjunrc	79	R	Exponent term for form loss coefficient for irreversible losses, reverse
fjunrt	93	R	Total form loss coefficient for irreversible losses, reverse
flenlh	80	R	Total enthalpy flow in junction. Includes both phases and noncondensables
florgj	71	R	Junction flow regime number in real format 1. CTB high mixing bubbly 2. CTT high mixing bubbly/mist transition 3. CTM high mixing mist 4. BBY bubbly 5. SLG slug 6. ANM annular-mist 7. MPR mist-pre-CHF 8. IAN inverted annular 9. ISL inverted slug 10. MST mist 11. MPO mist-post-CHF 12. HST horizontal stratified 13. VST vertical stratified 14. MWY ECC mixer wavy 15. MWA ECC mixer wavy/annular-mist 16. MAM ECC mixer annular-mist 17. MMS ECC mixer mist 18. MWS ECC mixer wavy/slug transition 19. MWP ECC mixer wavy-plug-slug transition 20. MPL ECC mixer plug 21. MPS ECC mixer plug-slug transition 22. MSL ECC mixer slug 23. MPB ECC mixer plug-bubbly transition 24. MBB ECC mixer bubbly
formfj	29	R	Liquid form loss term
formgj	30	R	Vapor form loss term
fromface1214	130	I	from face 1 bits
fwalfj	90	R	Non-dimensional liquid wall friction coefficient
fwalgj	91	R	Non-dimensional vapor wall friction coefficient
fxj	68	R	Wall friction interpolating factor

**Table 10.27-1** Junction Variable Names in Comdeck jundat.hh

<b>Variable</b>	<b>fa(i)</b>	<b>Type</b>	<b>Description</b>
fxjo	69	R	Wall friction interpolation factor for previous time step
fxjos	84	R	Saved beginning of advancement interfacial friction interpolation coefficient for level model backup
ij1	2	I	From volume input code
ij1nx	42	I	From volume index
ij1vn	5	I	From volume ordinal number
ij2	3	I	To volume input code
ij2nx	43	I	To volume index
ij2vn	6	I	To volume ordinal number
ijflg	75	I	Junction direction flag 0 = 1D/1D or 1D/3D or 3D/1D 1 = 3D/3D direction 1 2 = 3D/3D direction 2 3 = 3D/3D direction 3
ijsk2	-	I	0
ijsk3	-	I	ijsk1 + ijsk2
ijsk4	-	I	0 if selap is not defined 2 if selap is defined (selap is the defined if SCDAP is used)
ijsk5	-	I	ijsk3 + ijsk4
ijsk6	-	I	0 if extjun and extj20 are not defined 20 if extjun and extj20 are defined These are the extra junction plot variables that are useful for debugging purposes
ijskp	-	I	Junction skip factor Initialized to 92 in a parameter statement
iregj	72	R	Vertical bubbly/slug flow junction flow regime number in real format
is2turbjunflg29	123	L	
is2vel1velflg9	103	L	
isab rareachgflg8	102	L	
isaccactvflg15	109	L	true if accumulator is active

**Table 10.27-1** Junction Variable Names in Comdeck jundat.hh

<b>Variable</b>	<b>fa(i)</b>	<b>Type</b>	<b>Description</b>
isairjunflg30	124	L	
isccflflg1	126	L	
ischokflg0	94	L	true if choked
ischoktstflgaccjun6	100	L	
isdbgprntflg20	136	L	
isdonprespvwrk15	131	L	
iseccmixflg18	134	L	
iseccmixflg19	135	L	
iselevchk0	125	L	
isfrmmomflxopt13	107	L	true if from volume has momentum flux off
isgodunovflg22	138	L	
isgodunovflg23	139	L	
ishorzvertjunflg26	120	L	
isinithsemmod24	140	L	
isinithsemmod25	141	L	
isinpflg7	101	L	
isjetjunflg25	119	L	
isjetmixflg19	113	L	
isjetmixflg20	114	L	
isjetmixflg21	115	L	
isjunfloregnum38	128	L	
islevelmod27	142	L	
isliqentrain30	145	L	
ismomflxoffflg14	108	L	
isnochokflg4	98	L	
isnolosscoefbrjun29	144	L	
isnpccflflg2	127	L	
isoldtimchokflg5	99	L	

**Table 10.27-1** Junction Variable Names in Comdeck jundat.hh

<b>Variable</b>	<b>fa(i)</b>	<b>Type</b>	<b>Description</b>
isrevfrmvolconflg2	96	L	true if reversed from volume connection
isrevtovolconflg3	97	L	true if reversed to volume connection
issepflg10	104	L	
issepflg22	116	L	
issepflg23	117	L	
issepflg24	118	L	
isstratflg16	110	L	
isstratflwflg11	105	L	
isstratinpdat17	111	L	
isstratinpdat18	112	L	
isstrtchjunflg17	133	L	
istdpjunflg1	95	L	true if time dependent junction
istomomfluxoffopt12	106	L	true if to volume has momentum flux off
isupdwnjunflg27	121	L	
isvapcontphasejun28	143	L	
isvlvflg28	122	L	
iswatpackflg21	137	L	
iswatpackjunflg16	132	L	
jc (now called blhjc)	4	I	Old packed word, not used anymore
jcatn	35	R	Density correction factor ( $\sqrt{\frac{\rho_i}{\rho_j}}$ ) applied to the junction convective term in choking
jcato	36	R	Density correction factor applied to the junction convective term in choking previous time step
jcex	56	I	Old packed word, not used anymore
jcnfnd	81	I	Index to component block for junction
jcnx1	44	I(2)	1: Index to scratch space for from volume 2: Same for to volume
jcnx2	46	I(2)	1: Index to diagonal matrix element for from volume 2: Same for to volume

**Table 10.27-1** Junction Variable Names in Comdeck jundat.hh

<b>Variable</b>	<b>fa(i)</b>	<b>Type</b>	<b>Description</b>
jcnx3	48	I(2)	1: Index to off-diagonal matrix element for from volume 2: Same for to volume
jcnxd	50	I(2)	1: Diagonal index for sum momentum equation 2: Diagonal index for difference momentum equation
jcnxs	52	I	Index to scratch space for junction
jdissc	54	R	Subcooled discharge coefficient
jdissh	74	R	Superheated discharge coefficient
jdistp	55	R	Two-phase discharge coefficient
junftl	7	I(2)	1: From pointer in output form without sign 2: To pointer in output form without sign
junno	53	I	Junction number for output editing
mflowj	31	R	Mass flow rate
mflwjo	88	R	Mass flow rate previous time step
njuns	1	I	Number of junctions
qualaj	22	R	Junction noncondensable quality
qualnj	37	R(5)	1: First noncondensable junction mass fraction 2: Second noncondensable junction mass fraction 3: Third noncondensable junction mass fraction 4: Fourth noncondensable junction mass fraction 5: Fifth noncondensable junction mass fraction
rhofj	23	R	Junction liquid density
rhogj	24	R	Junction vapor density
soncjo	89	R	Junction sound speed previous time step
sonicj	63	R	Junction sound speed divided by the junction density ratio (jcatn)
toface911	129	I	to face 1 bits
ufj	18	R	Junction liquid specific internal energy
ugj	19	R	Junction vapor specific internal energy
vdfjoo	66	R	Junction liquid void fraction time step two levels back

**Table 10.27-1** Junction Variable Names in Comdeck jundat.hh

<b>Variable</b>	<b>fa(i)</b>	<b>Type</b>	<b>Description</b>
vdgjoo	67	R	Junction vapor void fraction time step two levels back
velfj	14	R	Liquid velocity
velfjo	15	R	Liquid velocity previous time step
velgj	16	R	Vapor velocity
velgjo	17	R	Vapor velocity previous time step
vgjj	70	R	Vapor drift velocity
vlfjos	86	R	Saved beginning of advancement liquid velocity for level model backup
vlgjos	87	R	Saved beginning of advancement vapor velocity for level model backup
vodfjo	64	R	Junction liquid void fraction previous time step
vodfjr	25	R	Ratio of junction liquid void to upstream volume liquid void
vodgjo	65	R	Junction vapor void fraction previous time step
vodgjr	26	R	Ratio of junction vapor void to upstream volume vapor void
voidfj	20	R	Junction liquid void fraction
voidgj	21	R	Junction vapor void fraction
voidj	73	R	Junction vapor void fraction used in the interphase drag
xej	62	R	Junction equilibrium quality based on extrapolated pressure & internal energy from jchoke

## 10.28 lcntrl.hh & lcntrlc.hh

The lcntrl.hh comdeck is a dynamic block for the loop control array. It is used to hold the volume number and position data during sorting. The lcntrlc.hh comdeck contains a description of these variables. This description is given below where the slot number in the fa array that the variable is equivalenced to is

given in the fa(i) column. The dimensions and type of the variable, real\*8, integer, logical, or character, is given in the type column.

**Table 10.28-1** Loop Control Array Variable Names in Comdeck lcntrl.hh

Variable	fa(i)	Type	Description
vlpndx	1	I	Used to hold volume number and position data during hydrodynamic system sorting. Holds component indexes to allow printing of component, volume, and junction information in numerical order.

## 10.29 levtbl.hh & levtblc.hh

The levtbl.hh comdeck is a dynamic block for the level model. It is used to hold the level model data. The levtblc.hh comdeck contains a description of these variables. This description is given below where the slot number in the fa array that the variable is equivalenced to is given in the fa(i) column. The dimensions and type of the variable, real\*8, integer, logical, or character, is given in the type column.

**Table 10.29-1** Level Model Variable Names in Comdeck levtbl.hh

Variable	fa(i)	Type	Description
levska	5	I	Index of volume above head of this stack, used when moving a level from the head volume of this stack into the tail (bottom) volume of an adjacent stack where the adjacent stack is above this stack
levskb	6	I	Index of volume below the tail of this stack, used when moving a level from the tail volume of this stack into the head volume of an adjacent stack where the adjacent stack is below this stack
levskd	7	I	Flag indicating that this stack has been processed during this time advancement, used when moving a level from one stack into an adjacent stack
levskh	4	I	Index of volume at head or top of this level stack
levskl	2	I	Index of volume containing level in this level stack
levskn	1	I	Number of level stacks in this system
levsko	3	I	Index of volume containing level in this stack during previous time advancement

## 10.30 lpdat.hh & lpdatc.hh

The lpdat.hh comdeck is a dynamic block for the hydrodynamic systems control block. It is used to hold the variables that are unique to the hydrodynamic system loops. The lpdatc.hh comdeck contains a description of these variables. This description is given below where the slot number in the fa array that the

variable is equivalenced to is given in the fa(i) column. The dimensions and type of the variable, real\*8, integer, logical, or character, is given in the type column.

**Table 10.30-1** Hydrodynamic Systems Control Variable Names in Comdeck lpdat.hh

Variable	fa(i)	Type	Description
extsnn	1	R(20)	Extra system variables for plotting purposes (nn = 0->20)
gerrs	1	R	Error at which scaling and solution order will be reevaluated
ixcofp	1	R	Index to array holding non-zero values of matrix
ixip	1	R(2)	Pointer to control array for matrix inversion subroutine
ixipr	1	R(2)	Pointer indicating first non-zero in row of solution matrix
ixipx	1	R	Pointer to ip array during pminvd call
ixirn	1	R	Similar function to ixirnr for previously inverted array
ixirnr	1	R	Index to array holding position of non-zeros for matrix
ixirnx	1	R(2)	Pointer to irn array during pminvd call
ixivrn	1	R	Pointer to array holding indexes of matrix elements for nearly implicit advancement
ixsopr	1	R	Index to right hand side and solution results
ixw	1	R	Index to work array for matrix inversion subroutine
levstk	1	R	Offset (input); index (transient) to the level stack data for this system
lic	2	I	Offset (input); index (transient) to first hydrodynamic component in a system
licn	3	I	Number of hydrodynamic components in a system
lij	8	I	Offset (input); index (transient) to first junction in a system
lijn	1	R	Number of junctions in a system
liv	4	I	Offset (input); index(transient) to first volume in a system
livn	5	I	Number of volumes in a system
livnn	6	I	The number of the first volume in a system
llvect	1	R	Index to list vector for system
lnoncn	1	R	Number of noncondensable gasses in system
lnonmf	1	R	Number of molten metal species in system
lpackr	1	R	Packing flag from packer subroutine
lpskp	-	I	System skip factor, lpskp = lpskp1 + lpskp2

**Table 10.30-1** Hydrodynamic Systems Control Variable Names in Comdeck lpdat.hh

Variable	fa(i)	Type	Description
lpskp1	-	I	Number of variables slots in lpdat dynamic array Initialized to 47 in a parameter statement
lpskp2	-	I	Number of extra system plot variables Initialized to 20 in a parameter statement if extsys is defined, otherwise it is initialized to 0 in a parameter statement
lsuces	1	R	For each system: 0 If no need to repeat advancement with reduced time step, 1 If excessive truncation error, 2 If water property error, 3 If non-diagonal matrix, -1 If repeat advancement after success on previous advancement attempt
lsysnm	1	R	System name
nloops	1	I	Number of hydrodynamic systems
nnz	1	R(2)	Number of non zero elements in solution matrix
nnz2	1	R(2)	Number of matrix elements allowed during solution
nvr	1	R(2)	Order of matrix
nvrp	1	R(2)	Order of matrix plus one
sflag	1	R(2)	Indicates whether new scaling and solution order needed
sysdtc	1	R	System Courant limit
sysebt	1	R	System error measure
sysmer	1	R	System mass error
sysmtc	1	R	System mass from conservation law applied to system, new-time value
sysmto	1	R	Same as sysmtc but old-time value
sysmts	1	R	System mass from sum of densities times volumes

## 10.31 Ivectr.hh & Ivectrc.hh

The lpdat.hh comdeck is a dynamic block for the list vector pointers block. It is used to hold the pointers to the list vectors that are used in RELAP5 for vectorization. The Ivectrc.hh comdeck contains a description of these variables. This description is given below where the slot number in the fa array that the

variable is equivalenced to is given in the fa(i) column. The dimensions and type of the variable, real\*8, integer, logical, or character, is given in the type column.

**Table 10.31-1** List Vector Pointer Variable Names in Comdeck lvectr.hh

Variable	fa(i)	Type	Description
lv3d	14	I	Pointer to multidimensional components
lvabrp	10	I	
lvaccm	1	I	Pointer to accumulator list
lvajun	16	I	Pointer to list of all junctions
lvavol	17	I	Pointer to list of all volumes
lvhzfl	11	I	
lvjtmx	6	I	Pointer to jetmixer list
lvjusr	9	I	Pointer to junction-source list
lvnofr	13	I	Pointer to volumes without wall friction list
lvprz	15	I	Pointer to pressurizer component junction list
lvptr	1	I	List values
lpump	2	I	Pointer to pump list
lrvvol	7	I	Pointer to real volume list
lvscr	18	I	Pointer to scratch list
lvsepr	3	I	Pointer to separator list
lvtturb	5	I	Pointer to turbine list
lvtvol	8	I	Pointer to time dependent volume list
lvvalv	4	I	Pointer to valve list
lvwifr	12	I	Pointer to volumes with wall friction list

## 10.32 lvel.hh

The lpdat.hh comdeck is a dynamic block for the level model block. It is used to hold the level model variables. A description of these variables is given below where the slot number in the fa array that the

variable is equivalenced to is given in the fa(i) column. The dimensions and type of the variable, real\*8, integer, logical, or character, is given in the type column

**Table 10.32-1** Level Model Variable Names in Comdeck lvl.hh

Variable	Type	Description
dtlev	R	
dvoidc	R	
dvoidi	R	
epsal1	R	
safe7	R	
voidlt	R	

### 10.33 machaf.hh & machas.hh

The machas.hh comdeck contains specification statement for the integer variables used in “iand” statement function for Apollo and Sun computers.

```
integer ilwxyz, i2wxyz, iand
```

The machaf.hh comdeck contains the statement function definition of “iand”. The statement function converts the “iand” function to the “and” function.

```
iand(ilwxyz, i2wxyz) = and(ilwxyz, i2wxyz)
```

### 10.34 machef.hh & maches.hh

The maches.hh comdeck contains the specification statement for the integer variables that are used in the “ieor” statement function for the Apollo and Sun computers.

```
integer i3wxyz, i4wxyz, ieor
```

The machef.hh comdeck contains the statement function definition of “ieor”. The statement function converts the “ieor” function to the “xor” function.

```
ieor(i3wxyz, i4wxyz) = xor(i3wxyz, i4wxyz)
```

### 10.35 machlf.hh & machls.hh

The machls.hh comdeck contains the specification statements for the integer and real variables that are used in the “locf”, “locf4”, “locfi”, and “locfi4” statement functions for the Apollo, Cray, DEC, HP, IBM, PC, and Sun computers.

```
$if def,apollo,2
    integer locf, locf4, locfi, locfi4
```

```

        external locf, locf4, locfi, locfi4
$if def,cray,2
        real xawxyz
        integer iawxyz, locf, locf4, locfi, locfi4
$if def,decrisc,2
        integer locf, locf4, locfi, locfi4
        external locf, locf4, locfi, locfi4
$if def,decrisc2,2
        integer locf, locf4, locfi, locfi4
        external locf, locf4, locfi, locfi4
$if def,decalpha,2
        integer locf, locf4, locfi, locfi4
        external locf, locf4, locfi, locfi4
$if def,hp,2
        integer locf, locf4, locfi, locfi4
        external locf, locf4, locfi, locfi4
$if def,ibm,2
        integer locf, locf4, locfi, locfi4
        external locf, locf4, locfi, locfi4
$if def,ibmrisc,2
        integer locf, locf4, locfi, locfi4
        external locf, locf4, locfi, locfi4
$if def,laheyf77,2
        integer locf, locf4, locfi, locfi4
        external locf, locf4, locfi, locfi4
$if def,sun,2
        integer locf, locf4, locfi, locfi4
        external locf, locf4, locfi, locfi4

```

The machlf.hh comdeck contains the statement function definition for "locf", "locf4", "locfi", and "locfi4". These statement function convert the four address location functions to the appropriate function for the specific computer.

```

$if def,cray,4
    locf(xawxyz) = loc(xawxyz)
    locf4(xawxyz) = loc(xawxyz)
    locfi(iawxyz) = loc(iawxyz)
    locfi4(iawxyz) = loc(iawxyz)

$if def,vax,8

```

```

locf(xawxyz) = ishft(%loc(xawxyz), -3)
locfi(iawxyz) = ishft(%loc(iawxyz), -3)

$if -def,fourbyt,2
    locf4(xawxyz) = ishft(%loc(xawxyz), -3)
    locfi4(iawxyz) = ishft(%loc(iawxyz), -3)

$if def,fourbyt,2
    locf4(xawxyz) = ishft(%loc(xawxyz), -2)
    locfi4(iawxyz) = ishft(%loc(iawxyz), -2)

```

## 10.36 machof.hh & machos.hh

The machos.hh comdeck contains the specification statement for the integer variables that are used in the “ior” statement function for the Apollo and Sun computers.

```
integer i5wxyz, i6wxyz, ior
```

The machof.hh comdeck contains the statement function definition of “ior”. The statement function converts the “ior” function to the “or” function.

```
ior(i5wxyz, i6wxyz) = or(i5wxyz, i6wxyz)
```

## 10.37 machss.hh

The machss.hh comdeck contains the specification statements for the “ishft” integer function variables.

```
integer ishft
external ishft
```

## 10.38 makmap.hh

The makmap.hh comdeck is a specification block and /makmap/ common block. It contains the names of some variables that are used in the unit test programs. A description is given below of these variables in alphabetical order. The dimensions and type of the variable, real\*8, integer, logical, or character, is given in the type column.

**Table 10.38-1** Unit Test Variable Names in Comdeck makmap.hh

Variable	Type	Description
targcn	I	Ncount for making the map
target	I	Junction number for phantj map Volume number for phantv map Heat structure number for htcond map

**Table 10.38-1** Unit Test Variable Names in Comdeck makmap.hh

Variable	Type	Description
targmp	I	0 As the default 1 For phantj map 2 For phantv map 3 For htadv map

## 10.39 maxmem.hh

The maxmem.hh comdeck is a specification block and /maxmem/ common block. It contains the names of two variables that are used in the to set the primary and secondary memory allocation. A description is given below of these variables in alphabetical order. The dimensions and type of the variable, real\*8, integer, logical, or character, is given in the type column.

**Table 10.39-1** Primary and Secondary Memory Allocation Variable Names in Comdeck maxmem.hh

Variable	Type	Description
maxlcm.hh	I	Maximum amount of secondary memory (none now used)
maxscm.hh	I	Maximum amount of primary memory

## 10.40 miedtc.hh & miedtcl.hh

The miedtc.hh comdeck is a dynamic block for the minor edits. It contains the specification statements for the minor edit variables and their equivalences with the fa array. The miedtcl.hh comdeck contains a description of each of the variable names used in the miedtc.hh comdeck. A description is given below of these variables in alphabetical order and the equivalences to their slot number in the fa array. The dimensions and type of the variable, real\*8, integer, logical, or character, is given in the type column.

**Table 10.40-1** Minor Edit Variable Names in Comdeck miedtc.hh

Variable	fa(i)	Type	Description
micode.hh	6	I(2)	1: Number of control block or if 0 then common(12 bits max) 2: Location offset in block (18 bits max)
miconv.hh	8	R	
mietab	2	I	1: Integer part of minor edit request (card number)
mietab	2	I	3: Integer part of minor edit request
mietaf	2	R	2: Alphanumeric part of minor edit request
mihold	6	R	
milabl	6	R	

**Table 10.40-1** Minor Edit Variable Names in Comdeck miedtc.hh

Variable	fa(i)	Type	Description
mipck	1	I(5)	1: Number of items to be saved per time step (12 bits max) 2: Increment, 1 if no conversion factors, 2 if conversion factors (2 bits max) 3: Label pointer (18 bits max) 4: Save area pointer (18 bits max) 5: Number of time steps saved (6 bits max)
nmiet	1	I	Number of minor edit requests.

## 10.41 mtbls.hh & mtblc.hh

The mtbls.hh comdeck is a dynamic block for the thermal properties. It contains the specification statements for the thermal properties variables and their equivalences with the fa array. The mtblc.hh comdeck contains a description of each of the variable names used in the mtbls.hh comdeck. A description is given below of these variables in alphabetical order and the equivalences to their slot number in the fa array. The dimensions and type of the variable, real\*8, integer, logical, or character, is given in the type column.

**Table 10.41-1** Thermal Property Variable Names in Comdeck mtbls.hh

Variable	fa(i)	Type	Description
mtbl	3	I	Used for integer information in table
mtblen	2	I	Length of data for material
mtblr	3	R	Used for real (floating point) information in table
mtbnum	1	I	Composition or material number
mtbptr	1	I	Offset to material data
nmtbls	1	I	Number of materials

## 10.42 mxnfcd.hh & mtblc.hh

The mxnfcd.hh comdeck is a dynamic block for the thermodynamic property file names and molecular weights. It contains the specification statements for the thermal property file name and molecular weights variables and their equivalences with the fa array. The mtblc.hh comdeck contains a description of each of the variable names used in the mxnfcd.hh comdeck. A description is given below of

these variables in alphabetical order and the equivalences to their slot number in the fa array. The dimensions and type of the variable, real\*8, integer, logical, or character, is given in the type column.

**Table 10.42-1** Thermal Property Variable Names in Comdeck mxnfed.hh

Variable	fa(i)	Type	Description
fsymb1	1	C*8(mxnfld)	Symbols for available fluids set in blkdfa.F
mxnfld	1	I	tnorm Initialized to 14 in parameter statement
tpfnam	1	C*40(mxnfld)	Names of thermodynamic properties files from which steam tables are to be obtained for needed fluids File names initialized in blkdfa, and overridden by file names given in tpfnc1 and tpfnn1
tpfnc1	1	C*40(mxnfld)	Names of thermodynamic properties files as obtained from command line
tpfnn1	1	C*40(mxnfld)	Names of thermodynamic properties files as obtained from 120-129 input cards Overridden by file names given in tpfnc1
wmoles	1	R(mxnfld)	Molecular weight of fluid

## 10.43 npacom.hh

The npacom.hh comdeck is a specification block for the /npacom/ and /npaccm/ common blocks. It contains the names of the variables that are used in the link to the NPA. A description is given below of these variables in alphabetical order. The dimensions and type of the variable, real\*8, integer, logical, or character, is given in the type column.

**Table 10.43-1** NPA Link Names in Comdeck npacom.hh

Variable	Type	Description
npamsg	C*96(5)	
nwq	I	

## 10.44 plotpc.hh

The plotpc.hh comdeck is a specification block for the /plotpc/ common block. It contains the names of the variable that is used for plots on the PC. A description is given below of this variable. The dimensions and type of the variable, real\*8, integer, logical, or character, is given in the type column.

**Table 10.44-1** PC Plot Variable Names in Comdeck plotpc.hh

Variable	Type	Description
image	C*1(101,51)	

## 10.45 przdat.hh & przdatc.hh

The przdat.hh comdeck is a dynamic block for the pressurizer component. It contains the specification statements for the pressurizer variables and their equivalences with the fa array. The przdatc.hh comdeck contains a description of each of the variable names used in the przdat.hh comdeck. A description is given below of these variables in alphabetical order and the equivalences to their slot number in the fa array. The dimensions and type of the variable, real\*8, integer, logical, or character, is given in the type column.

**Table 10.45-1** Pressurizer Component Variable Names in Comdeck przdat.hh

Variable	fa(i)	Type	Description
prjnx	20	I	Pressurizer junction indices starting from the top
przln	16	I	Pressurizer liquid level node index
prlvl	13	R	Liquid level in pressurizer
prznds	19	I	Number of pressurizer volumes (and junctions including the surge line junction)
przvnx	21	I	Pressurizer volume indices starting from the top
pzhctf	14	R	User supplied interphase heat transfer coefficient for liquid
pzhctg	15	R	User supplied interphase heat transfer coefficient for vapor
srgvol	18	I	
srgjun	17	I	Surge line junction index

## 10.46 pumpblk.hh & pumpblx.hh

The pumpblk.hh comdeck is a dynamic block for the pump component. It contains the specification statements for the pump component variables and their equivalences with the fa array. The pumpblx.hh comdeck contains a description of each of the variable names used in the pumpblk.hh comdeck. A description is given below of these variables in alphabetical order and the equivalences to their slot number

in the fa array. The dimensions and type of the variable, real\*8, integer, logical, or character, is given in the type column.

**Table 10.46-1** Pump Component Variable Names in Comdeck pumpblk.hh

Variable	fa(i)	Type	Description
ipmtbl	2	I	Equivalenced to impvtr
ipmval	4	R	Alphanumeric part of variable request code
ipmvnm	5	I	Numeric part of variable request code
ipmvpc	6	I(2)	1: Dynamic block number 2: Offset during input, index during transient
ipmvtl	8	I(3)	1: Number of items/entry 2: Total number of items 3: current position in table
ipmvtr	2	I(2)	1: Pump velocity table trip number 2: Offset to trip during input, index during transient
ipu2di	39	I(3)	1: Offset from beginning of this component to two phase homologous table octant offsets if no referral, referred component number if referral 2: Offset from origin of component block to two phase octant offsets 3: Offset to be added to octant offsets
ipuctr	46	I(2)	1: Shaft disconnect trip number 2: Offset to trip during input, index during transient
ipuhmi	35	I(2)	1: Offset within this component to two phase head multiplier if no referral, referred component number if referral 2: Offset from origin of component block to two phase head multiplier table
ipumtk	42	I(2)	1: Offset within this component to pump motor torque if no referral, referred component number if referral 2: Offset from origin of component block to pump motor torque table
ipurvi	31	I	Shaft component number
ipuspi	44	I(2)	1: Offset within this component to pump speed if no referral, referred component number if referral 2: Offset from origin of component block to pump speed table

**Table 10.46-1** Pump Component Variable Names in Comdeck pumpblk.hh

<b>Variable</b>	<b>fa(i)</b>	<b>Type</b>	<b>Description</b>
iputdi	32	I(3)	1: Offset from beginning of this component to single phase homologous table octant offsets if no referral, referred component number if referral 2: Offset from origin of component block to single phase octant offsets 3: Offset to be added to octant offsets
iputmi	37	I(2)	1: Offset within this component to two phase torque multiplier if no referral, referred component number if referral 2: Offset from origin of component block to two phase torque multiplier table
iputrp	29	I(2)	1: Trip for pump motor power 2: Offset to trip during input, index during transient
pmpbsp	28	R	Maximum reverse speed for trip
pmpfsp	27	R	Maximum forward speed for trip
pmpint	19	R	Pump moment of inertia
pmpmt	48	R	Motor torque
pmpnrt	49	R	Calculated pump inertia
pmpold	13	R	Actual pump angular velocity
pmprf1	16	R	Rated pump flow
pmprhsd	17	R	Rated pump head
pmprho	20	R	Rated or initial density
pmprmr	21	R	Rated pump motor torque
pmprsp	14	R	Rated pump angular velocity
pmprtk	18	R	Rated pump torque
pmpspr	15	R	Ratio of initial to rated pump velocity (input), old angular velocity (during advancement)
pmpstm	26	R	Elapsed time of pump trip
pmptbl	2	R	
pmptf1	22	R	Coefficients for frictional torque as a cubic function of speed ratio
pmptf2	23	R	Coefficients for frictional torque as a cubic function of speed ratio
pmptf3	24	R	Coefficients for frictional torque as a cubic function of speed ratio
pmptfy			

**Table 10.46-1** Pump Component Variable Names in Comdeck pumpblk.hh

<b>Variable</b>	<b>fa(i)</b>	<b>Type</b>	<b>Description</b>
pmpthd	50	R	Pump head
pmptrp	52	R	
pmpttk	51	R	Pump torque
pmpvtl	11	R	Entries for pump velocity table

## 10.47 radhtc.hh & radhtcc.hh

The radhtc.hh comdeck is a dynamic block for the radiation heat transfer. It contains the specification statements for the radiation heat transfer variables and their equivalences with the fa array. The radhtcc.hh comdeck contains a description of each of the variable names used in the radhtc.hh comdeck. A description is given below of these variables in alphabetical order and the equivalences to their slot number in the fa array. The dimensions and type of the variable, real\*8, integer, logical, or character, is given in the type column.

**Table 10.47-1** Radiation Heat Transfer Variable Names in Comdeck radhtc.hh

<b>Variable</b>	<b>fa(i)</b>	<b>Type</b>	<b>Description</b>
emis	3	R	Emissivity of a surface
ialpof	6	I	
iareof	5	I	
ias	0	R	Offset to surface area (pointer into heat structure file)
irhflx	2	I	Offset to radiation heat fluxes which are for all heat structure.
irhoff	10	I	Offset to radiation heat transfer surface data for a set
itemof	4	I	Offset to surface temperature (pointer into heat structure file)
itg	0	R	Offset to fluid temperature (pointer into volume file)
ivewof	11	I	Offset to vfij matrix for a set
jlr	2	I	0 = Left surface (input) 1 = Right surface (input) In iradht subroutine jlr is changed to the sequence number of radiation surface so that it points to item of location in the heat structure dynamic file. This index is: - for left surface + for right surface
jrh	1	I	Heat conductor surface number (input)
nrh	5	I	Number of radiation surfaces in the set

**Table 10.47-1** Radiation Heat Transfer Variable Names in Comdeck radhtc.hh

<b>Variable</b>	<b>fa(i)</b>	<b>Type</b>	<b>Description</b>
nrhskp	-	I	Skip factor for radiation heat transfer surface variables, parameter set to 7
nrset	1	I	Number of sets of radiation surfaces
nrsskp	-	I	Skip factor for radiation heat transfer set variables, parameter set to 9
qlrad	1	R	Left radiation heat flux to heat conductors (not just radiative heat conductors)
qrad	7	R	Radiation heat flux for a surface
qrrad	1	R	Right radiation heat flux to heat conductors (not just radiative heat conductors)
refset	4	I	The set number from which this set gets its view factors. Refset must always be less than setno.
setno	3	I	The set number. May be negative during input processing
timrof	9	R	Last time radiation heat transfer ended for a set
timron	8	R	Last time radiation heat transfer calculation began for a set Timron positive means radiation calculation is off Timron negative means radiation calculation is on
trmin	6	R	Minimum surface temperature of surfaces. If all surfaces have temp less than trmin no radiation calculated for the set
vfij	1	R	View factors for first, next,... set (nrh(1)**2 words)
voidmn	7	R	Minimum vapor void fraction. if boundary volume void fraction is less than voidmn no radiation calculated for the set

## 10.48 rcompc.hh and cmpalf.hh

The rcompc.hh comdeck is a specification block and /rcompa/ and /rcompc/ common blocks. It contains the specification statements for the component variable names. The cmpalf.hh comdeck contains the data statements for the component names in the /rcompa/ common block. A description is given below

of these variables in alphabetical order. The dimensions and type of the variable, real\*8, integer, logical, or character, is given in the type column.

**Table 10.48-1** Component Variable Names in Comdeck rcompc.hh

Variable	Type	Description
cmpalf	C(17)*8	Component name. Set in data statement in cmpalf.hh. 1 pipe 2 tmdpvol 3 mtpljun 4 pump 5 branch 6 jetmixer 7 annulus 8 separatr 9 tmdpjun 10 snglvol 11 sngljun 12 valve 13 accum 14 turbine 15 eccmix 16 multid 17 prizer
cmpflg	L	
cmpsra	L	
ncomp	I	

## 10.49 rflhtc.hh

The rflhtc.hh comdeck is a specification block and /rflhtc/ common block. It contains the specification statements for the reflood heat transfer variable names. A description is given below of these variables in alphabetical order. The dimensions and type of the variable, real\*8, integer, logical, or character, is given in the type column.

**Table 10.49-1** Reflood Heat Transfer Variable Names in Comdeck rflhtc.hh

Variable	Type	Description
idrfl	I	
indtol	I	
inscr1	I	
inscr2	I	

**Table 10.49-1** Reflood Heat Transfer Variable Names in Comdeck rflhtc.hh

<b>Variable</b>	<b>Type</b>	<b>Description</b>
inxlsr	I(2)	
inxtmn	I	
mesh4	I	
meshy	I	
mmeshz	I	
nqfmov	I	

## 10.50 rkinc.hh and rkincc.hh

The rkinc.hh comdeck is a dynamic block for the reactor kinetics variables. It contains the specification statements for the reactor kinetics variables and their equivalences with the fa array. The rkincc.hh comdeck contains a description of each of the variable names used in the rkinc.hh comdeck. A description is given below of these variables in alphabetical order and the equivalences to their slot number in the fa array. The dimensions and type of the variable, real\*8, integer, logical, or character, is given in the type column.

**Table 10.50-1** Reactor Kinetics Variable Names in Comdeck rkinc.hh

<b>Variable</b>	<b>fa(i)</b>	<b>Type</b>	<b>Description</b>
alttlbf9	9	L	
gmaxfctrstarflg14	14	L	
initfailflg8	8	L	
initflag6	6	L	
iscnfnctcntrl01	3	I	No. table coordinates bits, 4, 8, or 16
nmbrtblcoord0204	4	I	
reftempunits10	10	L	
reftempunits11	11	L	
rkbol	25	R	Delayed neutron fraction over generation time, point kinetics power option
rkcapt	35	R	Reactor operating period initially, then current time value (starting from zero) during decay heat initialization and transient advancement, point kinetics power option

**Table 10.50-1** Reactor Kinetics Variable Names in Comdeck rkinc.hh

<b>Variable</b>	<b>fa(i)</b>	<b>Type</b>	<b>Description</b>
rkcnf1	47	R	Integration weights at full interval, point kinetics power option
rkcnf2	48	R	Integration weights at full interval, point kinetics power option
rkcnf3	49	R	Integration weights at full interval, point kinetics power option
rkcnh1	44	R	Integration weights at half interval, point kinetics power option
rkcnh2	45	R	Integration weights at half interval, point kinetics power option
rkcnh3	46	R	Integration weights at half interval, point kinetics power option
rkcoef	1	R	Interpolating elements, point kinetics power option
rkcof1	6	R	
rkcof2	7	R	
rkcof3	8	R	
rkcoh1	3	R	
rkcoh2	4	R	
rkcoh3	5	R	
rkdeni	1	I(3)	1: Number of quantities per set in density table, point kinetics power option 2: Number of quantities in table 3: Current position in table
rkdenr	4	R	Density reactivity table if separable Coordinates if table, point kinetics power option
rkdepv	41	R	Dependent variables being advanced in time, point kinetics power option
rkdnppt	19	I	Pointer to density reactivity data (point separable feedback) Pointer to coordinate data (tabular feedback) Offset/index to composition data (1-D), point kinetics power option

**Table 10.50-1** Reactor Kinetics Variable Names in Comdeck rkinc.hh

<b>Variable</b>	<b>fa(i)</b>	<b>Type</b>	<b>Description</b>
rkdopi	1	I(3)	1: Number of quantities per set in Doppler table, point kinetics power option 2: Number of quantities in table 3: Current position in table
rkdopr	4	R	Doppler reactivity table, point kinetics power option
rkdptt	20	I	Pointer to Doppler reactivity data (point separable feedback) Pointer to tabular data (point tabular feedback) Offset/index to zone data (one-d), point kinetics power option
rkdt	30	R	Reactor kinetic time step, point and nodal kinetics power option
rkfcd	1	I	Code to determine if new interpolating elements are needed, point kinetics power option
rkfi	42	R	Delayed neutron and gamma yield fractions, point kinetics power option
rkfta	4	R	Heat structure temperature reactivity coefficient, point kinetics power option
rkf38	33	R	U239 production factor, point kinetics power option Number of U238 atoms produced per fission, nodal kinetics power option
rkwf	3	R	Doppler weight for heat structure feedback, point kinetics power option
rkhtno	1	I(2)	1: Heat structure number, point kinetics power option 2: Offset to heat structure data
rkinfeedbackopt5	5	L	
rkinpoint1dflg7	7	L	
rklmda	43	R	Decay constants for delayed neutrons and gamma decay, point kinetics power option
rknden	1	I	Coordinate subscripts, point kinetics power option
rknsc	17	I	Number of scram data entries, point kinetics power option
rknser	1	I	1: General table number or control variable number, point kinetics power option 2: Offset or index to general table or control variable

**Table 10.50-1** Reactor Kinetics Variable Names in Comdeck rkinc.hh

<b>Variable</b>	<b>fa(i)</b>	<b>Type</b>	<b>Description</b>
rknfb	23	I(2)	1: Number of heat structures in reactivity feedback, point kinetics power option 2: Pointer to heat structure feedback data
rknum	1	I	Number of reactor kinetics equations, point and nodal kinetics power option
rknumd	2	I	Number of delay groups, point and nodal kinetics power option
rknvfb	21	I(2)	1: Number of volumes in reactivity feedback, point kinetics power option 2: Pointer to volume feedback data
rkoaa	2	R	Coefficients for cross sections, nodal kinetics power option
rkoacf	4	I(2)	1: Composition figure for axial mesh plane, nodal kinetics power option 2: Offset/index to composition figure for axial mesh plane
rkoalp	2	R	Averaged void fraction in volume region, nodal kinetics power option
rkoazf	2	I(2)	1: Zone figure identifier for axial mesh plane, nodal kinetics power option 2: Offset/index to zone figure for axial mesh plane
rkob	4	R	Averaged poison density in volume region, nodal kinetics power option
rkocby	20	I	Maximum order of Chebychev fission source extrapolation polynomial, nodal kinetics power option
rkocfg	2	I	1: Composition identifier in composition figure, nodal kinetics power option 2: Offset/index to composition data
rkocfi	1	I	Composition figure identifier, nodal kinetics power option
rkocid	1	I	Material composition identifier, nodal kinetics power option
rkoden	2	R	Average fluid density in volume region, nodal kinetics power option
rkoegv	38	R	Eigenvalue, nodal kinetics power option

**Table 10.50-1** Reactor Kinetics Variable Names in Comdeck rkinc.hh

<b>Variable</b>	<b>fa(i)</b>	<b>Type</b>	<b>Description</b>
rkoepk	34	R	Eigenvalue convergence criterion, nodal kinetics power option
rkofbs	3	I	Number of heat structures in heat structure region, nodal kinetics power option
rkofbv	5	I	Number of volumes in volume region, nodal kinetics power option
rkoffa	16	I	Offset for loop over all kinetics equations (point) Last filndx(21) value to compute indexes, point kinetics power option
rkoffd	15	I	Offset for loop over delay groups (point) Offset/index to kinetics mesh information (1-D), point kinetics power option
rkofi	1	R	Decay heat group yields, nodal kinetics power option
rkoih0	13	I	Axial plane exterior boundary condition flag, nodal kinetics power option
rkoil2	37	R	Inner iteration L 2 convergence criterion, nodal kinetics power option
rkoiot	32	I	Maximum number of outer iterations per time step in transient, nodal kinetics power option
rkoips	14	I	Axial plane interior boundary condition flag, nodal kinetics power option
rkoizd	16	I	Bottom boundary condition flag, nodal kinetics power option
rkoizu	15	I	Top boundary condition flag, nodal kinetics power option
rkokit	31	I	Maximum number of outer iterations per invocation of the kinetics modules, nodal kinetics power option
rkolmd	2	R	Decay heat decay constants, nodal kinetics power option
rkomcr	1	I	Number of control rods associated with a kinetics node, nodal kinetics power option
rkomeg	28	R	Current reciprocal period, point kinetics power option
rkomnd	1	R	Height of top of each of kinetic mesh interval from bottom of reactor core, nodal kinetics power option

**Table 10.50-1** Reactor Kinetics Variable Names in Comdeck rkinc.hh

<b>Variable</b>	<b>fa(i)</b>	<b>Type</b>	<b>Description</b>
rkonc	43	I	Number of material compositions, nodal kinetics power option
rkoncf	44	I	Number of composition figures, nodal kinetics power option
rkongt	9	I	Number of thermal neutron energy groups, nodal kinetics power option
rkonhr	7	I	Number of rings in hex geometry, nodal kinetics power option
rkonmg	11	I	Number of decay heat equations, nodal kinetics power option
rkonn	47	R(7)	1: Offset/index to zone average property data, nodal kinetics power option 2: Offset/index to composition data 3: Offset/index to control rod insertion data 4: Offset/index to control fraction data 5: Offset/index to zone power data 6: Offset/index to kinetics mesh data 7: Offset/index to temporary decay heat powers and precursor densities 8: Offset/index to zone figure data 9: Offset/index to composition figure data 10: Offset/index to control rod weight factor data 11: Offset/index to decay heat yields and decay constants
rkonng	8	I	Number of neutron energy groups, nodal kinetics power option
rkonnm	18	I	Number of outer iteration between nodal expansion method coupling coefficient computations, nodal kinetics power option
rkonnx	4	I	Number of meshes in x direction on axial mesh plane, nodal kinetics power option
rkonny	5	I	Number of meshes in y direction on axial mesh plane, nodal kinetics power option
rkonnz	6	I	Number of axial mesh planes, nodal kinetics power option
rkonr	45	I	Number of control rod groups, nodal kinetics power option

**Table 10.50-1** Reactor Kinetics Variable Names in Comdeck rkinc.hh

<b>Variable</b>	<b>fa(i)</b>	<b>Type</b>	<b>Description</b>
rkonsr	42	I	Number of heat structure regions in a zone, nodal kinetics power option
rkonth	10	I	Number of thermal scattering iterations, nodal kinetics power option
rkonvr	41	I	Number of volume regions in a zone, nodal kinetics power option
rkonxy	33	I	Number of solution nodes in an axial plane, nodal kinetics power option
rkonz	39	I	Number of zones, nodal kinetics power option
rkonzf	40	I	Number of zone figures, nodal kinetics power option
rkool2	36	R	Outer iteration L 2 convergence criterion, nodal kinetics power option
rkooli	35	R	Outer iteration L infinity convergence criterion, nodal kinetics power option
rkopt (now called blhrkopt)	3	I	Old packed word, not used anymore
rkoral	6	I	Active length of control rod, nodal kinetics power option
rkorcf	1	R(2)	1: Control rod control fraction on each axial plane for active portion of rod, nodal kinetics power option 2: Control rod control fraction on each axial plane for driver portion of rod
rkorcl	4	I(2)	1: Identifier of control system variable which moves this rod, nodal kinetics power option 2: Offset/index in control system data block where data controlling this rod can be found
rkorf	2	I(2)	1: Identifier of control rod associated with kinetics mesh, nodal kinetics power option 2: Offset/index to control fraction data for this rod
rkorid	1	I	Control rod identifier, nodal kinetics power option, nodal kinetics power option
rkorif	46	I	Control rod insertion location flag (0 = top, 1 = bottom), nodal kinetics power option
rkorin	2	R(2)	1: Insertion depth of control rod, nodal kinetics power option 2: Initial insertion depth of control rod

**Table 10.50-1** Reactor Kinetics Variable Names in Comdeck rkinc.hh

Variable	fa(i)	Type	Description
rkosn	4	I(2)	1: Heat structure identifier, nodal kinetics power option 2: Offset/index in heat structure database for data for this heat structure
rkoswf	6	R	Heat structure weight factor, nodal kinetics power option
rkosym	12	I	Axial mesh plane symmetry flag, nodal kinetics power option
rkotf	2	R	Structure region average fuel temperature, nodal kinetics power option
rkotm	3	R	Volume region average moderator temperature, nodal kinetics power option
rkovn	6	I(2)	1: Volume identifier, nodal kinetics power option 2: Offset/index in volume database for data for this volume
rkovwf	8	R	Volume weight factors, no, nodal kinetics power optional kinetics power option, (1) factor for average void fraction or density, (2) factor for average fluid temperature, and (3) factor for average poison density
rkozap	5	R	Zone actinide decay power
rkozdv	6	R	Dependent variables for zone decay heat model, nodal kinetics power option
rkozfg	2	I	Zone figure, nodal kinetics power option
rkozfī	1	I	Zone figure identifier, nodal kinetics power option
rkozfp	3	R	Zone fission power, nodal kinetics power option
rkozgp	2	R	Zone decay power, nodal kinetics power option
rkozid	1	I	Zone identifier, nodal kinetics power option
rkozkp	4	R	Zone fission product decay power, nodal kinetics power option
rkoztp	1	R	Zone total power, nodal kinetics power option
rkpow	36	R	Total reactor power, i.e., the sum of fission power and gamma decay power, point and nodal kinetics power option

**Table 10.50-1** Reactor Kinetics Variable Names in Comdeck rkinc.hh

<b>Variable</b>	<b>fa(i)</b>	<b>Type</b>	<b>Description</b>
rkpowa	40	R	Actinide decay power, point and nodal kinetics power option
rkpowf	38	R	Power from fission, point and nodal kinetics power option
rkpowg	37	R	Power from gamma decay, i.e. the sum of fission product decay power and actinide decay power, point and nodal kinetics power option
rkpowk	39	R	Fission product decay power, point and nodal kinetics power option
rkpsi	34	R	Fissions per initial fissile atom, point kinetics power option
rkqval	32	R	Fraction of fission energy released at fission, point and nodal kinetics power option
rkrn	27	R	Current reactivity, point kinetics power option
rkro	26	R	Reactivity bias after initialization, point kinetics power option
rkslob	29	R	Source divided by delayed neutron fraction over generation, point kinetics power option
rksptr	18	I	Pointer to scram data, point kinetics power option, point kinetics power option
rksum	31	R	Holds delayed neutron summation, point kinetics power option
rktabl	1	R	Table data, point kinetics power option
rkvoln	1	I(2)	1: Volume number, point kinetics power option 2: Offset or index to volume data
rkvta	4	R	Volume temperature reactivity coefficient, point kinetics power option
rkvwf	3	R	Density weight for volume feedback, point kinetics power option
spckindensvar12	12	L	
spckincpwrlgl3	13	L	

## 10.51 rknatb.hh and rknatbc.hh

The rknatb.hh comdeck is a specification block and /rknatb/ common block. It contains the specification statements for the ANS79 neutron absorption effect variable names. The rknatbc.hh comdeck contains the descriptions of the variable names in the /rknatb/ common block. A description is given below of these variables in alphabetical order. The dimensions and type of the variable, real\*8, integer, logical, or character, is given in the type column.

**Table 10.51-1** ANS79 Neutron Absorption Variable Names in Comdeck rknatb.hh

Variable	Type	Description
rkilk	I	Current position in table
rkntbl	R(2,48)	Table of neutron absorption 'g factor' as a function of shutdown time
rkntx	R	Current interpolation factor

## 10.52 rmadac.hh

The rmadac.hh comdeck is a specification block and /rmadac/ common block. It contains the specification statements for the material data for heat structures variable names. A description is given below of these variables in alphabetical order. The dimensions and type of the variable, real\*8, integer, logical, or character, is given in the type column.

**Table 10.52-1** Material Data for Heat Structures Variable Names in Comdeck rmadac.hh

Variable	Type	Description
m1a	I(4)	True 32 bit integer, equivalenced to m1af
m1af	R(4)	Equivalenced to m1a
m1b	R	
m1c	R	
m2a	I(4)	True 32 bit integer, equivalenced to m2af
m2af	R(4)	Equivalenced to m2a
m2b	R	
m2c	R	
m3a	I(4)	True 32 bit integer, equivalenced to m3af
m3af	R(4)	Equivalenced to m3a
m3b	R(4)	
m3c	R(18)	

**Table 10.52-1** Material Data for Heat Structures Variable Names in Comdeck rmadac.hh

<b>Variable</b>	<b>Type</b>	<b>Description</b>
m4a	I(4)	True 32 bit integer, equivalenced to m4af
m4af	R(4)	Equivalenced to m4a
m4b	R(40)	
m4c	R(32)	
m5a	I(4)	True 32 bit integer, equivalenced to m5af
m5af	R(4)	Equivalenced to m5a
m5b	R(22)	
m5c	R(10)	

## **10.53 rrkinc.hh**

The rrkinc.hh comdeck is a specification block and /rrkinc/ common block. It contains the specification statements for the reactor kinetics data variable names. A description is given below of these variables in alphabetical order. The dimensions and type of the variable, real\*8, integer, logical, or character, is given in the type column.

**Table 10.53-1** Reactor Kinetics Data Variable Names in Comdeck rrkinc.hh

<b>Variable</b>	<b>Type</b>	<b>Description</b>
fberr	L	
idin	I(8)	Equivalenced to rdin
ix	I	
ld	I	
lde	I	
lim	I	
lx	I	
lxa	I	
lxd	I	
lxl	I	
lxp	I	
rdin	R(8)	Equivalenced to idin
rknumi	I	

## 10.54 scrq.hh

The scrq.hh comdeck is a dynamic block for the stateq variables. It contains the specification statements for the stateq variables and their equivalences with the scratch part of the fa or tt array. The tt(1) array is equivalenced to the fa(1) in scrtch.hh A description is given below of these variables in alphabetical order and the equivalences to their slot number in the fa array. The dimensions and type of the variable, real\*8, integer, logical, or character, is given in the type column.

**Table 10.54-1** Stateq Variable Names in Comdeck scrq.hh

Variable	fa(ix)	Type	Description
beta	6	R	
betaf	17	R	
betafs	67	R	
betag	18	R	
betags	68	R	
cp	8	R	
cpf	21	R	
cpfs	71	R	
cpg	22	R	
cpgs	72	R	
cps	58	R	
entfs	75	R	
entgs	76	R	
entpy	24	R	
entpyf	25	R	
entpyg	26	R	
hbar	5	R	
hsubf	15	R	
hsubfs	65	R	
hsubg	16	R	
hsubgs	66	R	
ihld1	42	I	
ihld2	43	I	

**Table 10.54-1** Stateq Variable Names in Comdeck scrq.hh

<b>Variable</b>	<b>fa(ix)</b>	<b>Type</b>	<b>Description</b>
ihld2a	108	I	
ihld4a	109	I	
ihld9	48	I	
ihld10	49	I	
is23	23	I	
itt	1	I(109)	
kapa	7	R	
kapaf	19	R	
kapaff	106	R	
kapafs	69	R	
kapag	20	R	
kapagg	107	R	
kapags	70	R	
lflag	41	L	
lflag2	40	L	
pres	2	R	
ps	52	R	
psat	10	R	
qter	36	I	
qual	9	R	
rhofs	102	R	
rhogs	103	R	
tf	104	R	
tsat	51	R	
tsater	82	L	
tt	1	R(109)	
tter	32	L	
ttg	105	R	

**Table 10.54-1** Stateq Variable Names in Comdeck scrq.hh

Variable	fa(ix)	Type	Description
ubar	4	R	
usubf	13	R	
usubfs	63	R	
usubg	14	R	
usubgs	64	R	
vbar	3	R	
vs	53	R	
vsubf	11	R	
vsubfs	61	R	
vsubg	12	R	
vsubgs	62	R	

## 10.55 scrtch.hh & scrtchc.hh

The scrtch.hh comdeck is dynamic block for the scratch variables. It contains the specification statements for the scratch variables and their equivalences with the fa array. The scrtchc.hh comdeck contains a description of where each of the variable names are used in the RELAP5 code. The descriptions of the variables are given below in alphabetical order where the slot number in the fa array that the variable is equivalence to is given in the fa(i) column. The dimensions and type of the variable, real\*8, integer, logical, or character, are given in the type column. In the Description column, the subname1>subname2 syntax means that the variable has to survive from the first subroutine to the second subroutine.

**Table 10.55-1** Scratch Variable Names in Comdeck scrtch.hh

Variable	fa(i)	Type	Description
a1	70	R	preseq>eqfinl vimplt>pimplt
a2	71	R	preseq>eqfinl vimplt>pimplt
a3	72	R	preseq>eqfinl vimplt>pimplt
a4	73	R	preseq>eqfinl vimplt>pimplt
a5	74	R	preseq>eqfinl vimplt>pimplt
a11	45	R	preseq>eqfinl vimplt>pimplt
a12	46	R	preseq>eqfinl vimplt>pimplt
a14	48	R	preseq>eqfinl vimplt>pimplt

**Table 10.55-1** Scratch Variable Names in Comdeck scrtch.hh

<b>Variable</b>	<b>fa(i)</b>	<b>Type</b>	<b>Description</b>
a15	49	R	preseq>eqfinl vimplt>pimplt
a21	50	R	preseq>eqfinl vimplt>pimplt
a22	51	R	preseq>eqfinl vimplt>pimplt
a23	52	R	preseq>eqfinl vimplt>pimplt
a24	53	R	preseq>eqfinl vimplt>pimplt
a25	54	R	preseq>eqfinl vimplt>pimplt
a31	55	R	preseq>eqfinl vimplt>pimplt
a32	56	R	preseq>eqfinl vimplt>pimplt
a33	57	R	preseq>eqfinl vimplt>pimplt
a34	58	R	preseq>eqfinl vimplt>pimplt
a35	59	R	preseq>eqfinl vimplt>pimplt
a41	60	R	preseq>eqfinl vimplt>pimplt
a42	61	R	preseq>eqfinl vimplt>pimplt
a43	62	R	preseq>eqfinl vimplt>pimplt
a44	63	R	preseq>eqfinl vimplt>pimplt
a45	64	R	preseq>eqfinl vimplt>pimplt
a51	65	R	preseq vimplt>pimplt
a51s	1	R	vimplt
a52	66	R	preseq vimplt>pimplt
a52s	2	R	vimplt
a53	67	R	preseq vimplt>pimplt
a53s	3	R	vimplt
a54	68	R	preseq vimplt>pimplt
a54s	4	R	vimplt
a55	69	R	preseq>eqfinl vimplt>pimplt
a55s	25	R	vimplt
afrf	88	R	eqfinl
afuf	87	R	preseq eqfinl vimplt

**Table 10.55-1** Scratch Variable Names in Comdeck scrtch.hh

<b>Variable</b>	<b>fa(i)</b>	<b>Type</b>	<b>Description</b>
agrg	80	R	eqfinl
agug	86	R	preseq eqfinl vimplt
agxa	79	R	preseq vimplt
angmm	13	R	pump>vexplt>vimplt
areav	71	R(6)	vovel vlevela
arhof	89	R	vovel vlevela
arhog	95	R	vovel vlevela
aterm	45	R	fwdrag
avelf	45	R	phantv hifbub
avelfg	61	R	phantv
aviscf	141	R	vimplt
aviscg	142	R	vimplt
avisfs	149	R	vimplt
avisgs	150	R	vimplt
avkx	25	R	phantj>jchoke>vexplt>vimplt
avlx	26	R	phantj>jchoke>vexplt>vimplt
avrf	153	R	vexplt vimplt
avrgf	22	R	vexplt
avrg	154	R	vexplt vimplt
axvelf	47	R	fwdrag
b1	1	R	eqfinl pimplt
b2	2	R	eqfinl pimplt
b3	3	R	eqfinl pimplt
b4	4	R	eqfinl pimplt
b5	25	R	eqfinl
badfw	44	I	fwdrag
bb5	5	R	pimplt
beta	6	R	statep

**Table 10.55-1** Scratch Variable Names in Comdeck scrtch.hh

<b>Variable</b>	<b>fa(i)</b>	<b>Type</b>	<b>Description</b>
betaf	17	R	statep
betafs	67	R	statep
betag	18	R	statep
betags	68	R	statep
borona	5	R	brntrn
boronb	6	R	brntrn
bterm	46	R	fwdrag
bxvelg	48	R	fwdrag
celvec	64	R	phantv
cflno	3	R	brntrn
coefp	1	R	preseq>syssol >syssol simplt>syssol
coefv	1	R	vimplt>coev3d>jchoke>ccfl
coefx	1	R	preseq simplt tsetsl >syssol
cond	5	R	eqfinl pimplt
conm	6	R	eqfinl pimplt
conmf	3	R	eqfinl pimplt
conmg	4	R	eqfinl pimplt
convf	61	R	coev3d>vimplt flux3d>vexplt
convfs	63	R	vexplt vimplt
convg	62	R	coev3d>vimplt flux3d>vexplt
convgs	64	R	vexplt vimplt
costhe	181	R	phantv>phantj
cp	8	R	statep
cpf	21	R	statep
cpfs	71	R	statep
cpg	22	R	statep
cpgs	72	R	statep
cps	57	R	statep

**Table 10.55-1** Scratch Variable Names in Comdeck scrtch.hh

<b>Variable</b>	<b>fa(i)</b>	<b>Type</b>	<b>Description</b>
ctermx	15	R	fwdrag
cvelf	1	R	vovel>vimplt
cvelg	2	R	vovel>vimplt
cvmrv	97	R	vexplt
dbodx1	7	R	brntrn
dbodx2	8	R	brntrn
dbodxj	4	R	brntrn
dbodxv	1	R	brntrn
delv	96	R	vexplt vimplt
delxa	5	R	eqfinl simplt
difdpk	28	R	jchoke>ccfl>vimplt>simplt
difdpl	30	R	jchoke>ccfl>vimplt>simplt
diff	76	R	vexplt
difg	77	R	vexplt
difold	78	R	vexplt
difr	95	R	vexplt
difvfx	196	R(3)	vovel>vexplt
difvgx	199	R(3)	vovel>vexplt
dmvvec	65	R	phantv
dpdxfx	6	R	fwdrag
dpdxgx	7	R	fwdrag
dpstf	1	R	phantj>vexplt phantj>vimplt
drho	1	R	state
drivew	172	I	vimplt
dstar	178	R	phantv>phantj
dstvec	68	R	phantv
dternf	3	R	courn1
dterng	4	R	courn1

**Table 10.55-1** Scratch Variable Names in Comdeck scrtch.hh

<b>Variable</b>	<b>fa(i)</b>	<b>Type</b>	<b>Description</b>
dtsf	41	R	phantv hifbub
dtsfm	56	R	phantv
dtsfsb	60	R	phantv
dtsfsp	59	R	phantv
dtsg	42	R	phantv
dtsgm	58	R	phantv
dtsgms	57	R	phantv
dvelfj	1	R	vfinl
dvelgj	2	R	vfinl
dx	80	R	vexplt vimplt
dxk	40	R	vexplt vimplt
dxkx	2	R	phantj>ccfl>jchoke>vexplt>vimplt
dxl	41	R	vexplt vimplt
dxlx	3	R	phantj>ccfl>jchoke>vexplt>vimplt
dxx	47	R	vexplt vimplt
entfs	75	R	statep
entgs	76	R	statep
entpy	24	R	statep
entpyf	25	R	statep
entpyg	26	R	statep
errm	3	R	state
errmf	4	R	state
f2	62	R	phantv
fal	89	R	vexplt preseq>eqfinl vimplt>pimplt
fanms	78	R	phantv>fwdrag
ff	81	R	vexplt>preseq>eqfinl pimplt vimplt
fgrw	91	R	preseq>eqfinl
fidxup	160	R	hydro>phantv>hloss>jchoke>phantj>vexplt>vimplt

**Table 10.55-1** Scratch Variable Names in Comdeck scrtch.hh

<b>Variable</b>	<b>fa(i)</b>	<b>Type</b>	<b>Description</b>
fifj	87	R	vexplt vimplt
figj	79	R	vexplt vimplt
fjet	14	R	vexplt vimplt
flomap	172	I	eccmxv>phantv
flompj	172	I	eccmxj>phantj
fluxm	62	R	phantv
flxtrn	65	R	losubboil phantv
fp	95	R	preseq>eqfinl vimplt>pimplt
fpress	64	R	losubboil phantv
fracag	98	R	preseq>eqfinl
fractal	47	R	preseq>eqfinl
fricfj	40	R	vexplt vimplt
fricfk	57	R	vexplt vimplt
fricfl	58	R	vexplt vimplt
fricgj	41	R	vexplt vimplt
fricgk	59	R	vexplt vimplt
fricgl	60	R	vexplt vimplt
frlmf1	55	R	fwdrag
frlmf2	61	R	fwdrag
frlmg1	56	R	fwdrag
frlmg2	62	R	fwdrag
frtbfl	59	R	fwdrag
frtbf2	65	R	fwdrag
frtbg1	60	R	fwdrag
frtbg2	66	R	fwdrag
frtrf1	57	R	fwdrag
frtrf2	63	R	fwdrag
frtrg1	58	R	fwdrag

**Table 10.55-1** Scratch Variable Names in Comdeck scrtch.hh

<b>Variable</b>	<b>fa(i)</b>	<b>Type</b>	<b>Description</b>
frtrg2	64	R	fwdrag
fuf	100	R	preseq>eqfinl vimplt>pimplt
fug	99	R	preseq>eqfinl vimplt>pimplt
fwf1	6	R	fwdrag
fwf2	10	R	fwdrag
fwfaf1	4	R	fwdrag
fwfag1	5	R	fwdrag
fwfxaf	163	R(3)	phantv>eccmxv>vexplt>fwdrag>ccfl
fwfxag	163	R(3)	phantv>eccmxv>vexplt>fwdrag>ccfl
fwg1	7	R	fwdrag
fwg2	13	R	fwdrag
fxa	76	R	preseq>eqfinl
fxaa	145	R	vimplt>pimplt
gal	90	R	vexplt preseq>eqfinl vimplt>pimplt
gammsc	223	R	tran>htadv>eqfinl simplt
gammsw	222	R	tran>htadv>eqfinl simplt
gfwabs	4	R	fwdrag
gg	82	R	vexplt>preseq>eqfinl>pimplt vimplt
gp	96	R	preseq>eqfinl vimplt>pimplt
gsum	65	R(6)	volvel
guf	101	R	preseq>eqfinl
gug	97	R	preseq>eqfinl vimplt>pimplt
gxa	75	R	preseq>eqfinl
gxaa	144	R	vimplt>pimplt
hbar	5	R	statep
hfg	43	R	phantv hifbub
hgfc	38	R	phantv
hgfc1	38	R	phantv

**Table 10.55-1** Scratch Variable Names in Comdeck scrtch.hh

<b>Variable</b>	<b>fa(i)</b>	<b>Type</b>	<b>Description</b>
hggff	28	R	phantv
hifc	53	R	phantv hifbub
hifc1	54	R	phantv hifbub
hiff	184	R	phantv
hifh	76	R	preseq vimplt>pimplt
hifhdt	93	R	eqfinl
hige	51	R	phantv
higc1	52	R	phantv
higg	187	R	phantv
high	75	R	preseq vimplt>pimplt
highdt	92	R	eqfinl
higsub	55	R	phantv
hlossf	98	R	vexplt vimplt
hlossg	99	R	vexplt vimplt
hsgf	85	R	vexplt>preseq vimplt
hsubfs	65	R	statep
hsubgs	66	R	statep
htcff	219	R	tran>htadv>eqfinl>phantv>eqfinl>pimplt>preseq>vimplt
htcfg	202	R	tran>htadv>eqfinl>vimplt>pimplt>preseq
htcfp	220	R	tran>htadv>eqfinl>phantv>eqfinl>pimplt>preseq>vimplt
htcft	203	R	tran>htadv>eqfinl>vimplt>pimplt>preseq
htcfgf	204	R	tran>htadv>eqfinl>vimplt>pimplt>preseq
htcgg	218	R	tran>htadv>eqfinl>phantv>eqfinl>pimplt>preseq>vimplt
htcgp	217	R	tran>htadv>eqfinl>phantv>eqfinl>pimplt>preseq>vimplt
htcggt	205	R	tran>htadv>eqfinl>vimplt>pimplt>preseq
htgcgf	213	R	tran>htadv>eqfinl>phantv>eqfinl>pimplt>preseq>vimplt
htgcgg	214	R	tran>htadv>eqfinl>phantv>eqfinl>pimplt>preseq>vimplt
htgcgp	215	R	tran>htadv>eqfinl>phantv>eqfinl>pimplt>preseq>vimplt

**Table 10.55-1** Scratch Variable Names in Comdeck scrtch.hh

<b>Variable</b>	<b>fa(i)</b>	<b>Type</b>	<b>Description</b>
htgct	216	R	tran>htadv>eqfinl>phantv>eqfinl>pimplt>preseq>vimplt
htgwff	210	R	tran>htadv>eqfinl>phantv>eqfinl>pimplt>preseq>vimplt
htgwfg	211	R	tran>htadv>eqfinl>phantv>eqfinl>pimplt>preseq>vimplt
htgwfp	221	R	tran>htadv>eqfinl>phantv>eqfinl>pimplt>preseq>vimplt
htgwft	212	R	tran>htadv>eqfinl>phantv>eqfinl>pimplt>preseq>vimplt
hydltf	206	R	eqfinl>pimplt>htfinl
hydltg	207	R	eqfinl>pimplt>htfinl
hydltp	209	R	eqfinl>pimplt>htfinl
hydltt	208	R	eqfinl>pimplt>htfinl
ihh1	4	I	jprop stdry
ihh2	6	I	jprop stdry
ihh3	5	I	jprop
ihld1	42	I	preseq>eqfinl
ihld1	42	I	pimplt simplt statep vexplt vimplt
ihld2	43	I	preseq>eqfinl pimplt>simplt
ihld2	43	I	statep vexplt vimplt
ihld2a	108	I	statep
ihld3	44	I	vexplt statep preseq>eqfinl
ihld4	113	I	statep
ihld4a	109	I	statep
ihld5	46	I	statep
ihld6	47	I	statep
ihld7a	101	I	svh2x2
ihld8	49	I	svh2x2
ihld9	48	I	svh2x2 statep thcond
ihld10	49	I	statep thcond
ip	1	I	tsets1 syssol
ipr	1	I	tsets1>vimplt>simplt>syssol

**Table 10.55-1** Scratch Variable Names in Comdeck scrtch.hh

<b>Variable</b>	<b>fa(i)</b>	<b>Type</b>	<b>Description</b>
ireg	7	R	fidisj>phantj
irn	1	I	tsetsl syssol
irnr	1	I	tsetsl>syssol
irnr	1	I	tsetsl>vimplt>coev3d>syssol>simplt
is23	23	I	stvrpx svh2x2
isptra	116	I	vimplt
ivert	73	I	phantv
ivrn	1	I	tsetsl>vimplt
ja	33	I	stvrpx svh2x2
jb	34	I	stvrpx svh2x2
jetdf	184	I	vimplt
jetdg	183	I	vimplt
jetfdf	180	I	vimplt
jetjdg	179	I	vimplt
jetjsf	178	I	vimplt
jetjsg	177	I	vimplt
jetsf	182	I	vimplt
jetsg	181	I	vimplt
jhld1	31	I	phantv fwdrag
jhld2	32	I	phantv fwdrag
jhld3	33	I	phantv fwdrag
jhld4	34	I	phantv
jhld5	35	I	phantv
jhld6	36	I	phantv
jhl	37	I	phantv
jtcond	186	R	vimplt
jtcons	185	R	vimplt
jtdfdf	176	I	vimplt

**Table 10.55-1** Scratch Variable Names in Comdeck scrtch.hh

<b>Variable</b>	<b>fa(i)</b>	<b>Type</b>	<b>Description</b>
jtdjdg	175	I	vimplt
jtdjsf	174	I	vimplt
jtdjsg	173	I	vimplt
kapa	7	R	statep
kapaf	19	R	statep
kapaff	106	R	statep state
kapafs	69	R	statep
kapag	20	R	statep
kapagg	107	R	statep state
kapags	70	R	statep
lflag	41	I	statep
lflag2	40	I	statep
lgg1	8	I	jprop
lrhof	150	R	simplt
lrhog	149	R	simplt
ltest	8	I	eqfinl
ltest2	9	I	eqfinl
ltest3	10	I	eqfinl
ltestt	56	I	vexplt
ncrosk	45	R	vexplt vimplt
ncrosl	46	R	vexplt vimplt
nix	7	I	jprop
nmapp	72	I	phantv
nvalhi	74	I	phantv
nvalhx	75	I	phantv
paa	77	R	stvrpx svh2x2
pack	44	I	packer
pbb	78	R	stvrpx svh2x2

**Table 10.55-1** Scratch Variable Names in Comdeck scrtch.hh

<b>Variable</b>	<b>fa(i)</b>	<b>Type</b>	<b>Description</b>
pfinrg	169	R	phantv>eccmxv>vexplt>fwdrag>ccfl
pk	18	R	vexplt
pl	20	R	vexplt
pmhig	151	R	vexplt>preseq vimplt
pmpph	17	R	vexplt jchoke vimplt
pres	2	R	statep
ps	52	R	statep
psat	10	R	statep
pshig	152	R	vexplt>preseq vimplt
psld	72	R	vexplt vimplt
pslope	12	R	pump>vexplt>vimplt
psmf	70	R	vexplt vimplt
psmg	71	R	vexplt vimplt
psumf	91	R	vexplt
psumg	92	R	vexplt
pumpv	15	R	vimplt
qsater	86	I	statep
qter	36	I	statep
qual	9	R	statep
qualem	94	R	preseq
ratdpf	67	R	fwdrag
ratio	2	R	brntrn
ratiof	53	R(6)	volvel
ratiog	59	R(6)	volvel
ravrf	52	R	vexplt vimplt
ravrg	53	R	vexplt vimplt
resorm	93	R	preseq>eqfinl
resoru	92	R	preseq>eqfinl

**Table 10.55-1** Scratch Variable Names in Comdeck scrtch.hh

<b>Variable</b>	<b>fa(i)</b>	<b>Type</b>	<b>Description</b>
reynfl	91	R	fwdrag
reynf2	14	R	fwdrag
reyng1	92	R	fwdrag
reyng2	15	R	fwdrag
rfvfj	53	R	vexplt vimplt
rfvfrc	10	R	vimplt
rgvgj	54	R	vexplt vimplt
rgvgrc	11	R	vimplt
rhfg	44	R	phantv hifbub
rhocpf	46	R	phantv hifbub
rhofa	48	R	vexplt vimplt
rhofs	102	R	statep
rhoga	49	R	vexplt vimplt
rhogs	103	R	statep
rhov	2	R	state
rvcrit	175	R	phantv
sathfx	83	R	vexplt>preseq vimplt
sathgx	84	R	vexplt>preseq vimplt
sbmlhf	66	R	losubboil phantv
sbmllf	67	R	losubboil phantv
scrchh	63	R	phantv
scrchx	94	R	vexplt
scv1	7	R	eqfinl pimplt
scv2	8	R	eqfinl pimplt
scv2n	17	R(5)	eqfinl pimplt
scv3	9	R	eqfinl pimplt
scv4	10	R	eqfinl pimplt
scv5	11	R	eqfinl pimplt

**Table 10.55-1** Scratch Variable Names in Comdeck scrtch.hh

<b>Variable</b>	<b>fa(i)</b>	<b>Type</b>	<b>Description</b>
scv6	12	R	eqfinl pimplt
scv7	13	R	pimplt
scv8	14	R	pimplt
scv9	15	R	pimplt
scv10	16	R	pimplt
scvj1	14	R(11)	vfinl packer
scvj2	15	R	vfinl
scvj3	16	R	packer>vfinl
scvj4	17	R	vfinl
scvj5	191	R	vimplt
scvj6	192	R	vimplt
scvj7	193	R	vimplt
scvj8	194	R	vimplt
scvj11	187	R	vimplt
scvj12	25	R	vfinl
scvj22	188	R	vimplt
scvj33	189	R	vimplt
scvj44	190	R	vimplt
scvjck	1	R	vexplt>jchoke vimplt>jchoke
scvjn	163	R(5)	vimplt
scvtur	16	R	vexplt>jchoke vimplt
sinbt	10	R	phantj>fidisj
snk	65	R(2)	vexplt vimplt
snl	67	R(2)	vexplt vimplt
sorp	26	R	eqfinl vimplt
soura	112	R	vimplt>simplt
sourca	31	R	vexplt>preseq>eqfinl vimplt>pimplt
sourcf	33	R	vexplt>accum>preseq>eqfinl vimplt>pimplt

**Table 10.55-1** Scratch Variable Names in Comdeck scrtch.hh

<b>Variable</b>	<b>fa(i)</b>	<b>Type</b>	<b>Description</b>
sourcg	32	R	vexplt>accum>preseq>eqfinl vimplt>pimplt
sourci	94	R	eqfinl
sourcm	34	R	vexplt>preseq>eqfinl>vimplt>pimplt
sourcm	34	R	statep
sourcn	35	R	vexplt>eqfinl vimplt>pimplt
sourcp	1	R	preseq>packer>vfinl>eqfinl
sourcp	1	R	packer>vfinl pimplt>simplt
sourcq	111	R	vimplt>pimplt
sourcv	1	R	jchoke>ccfl>vimplt
sourff	22	R	pimplt>simplt
sourgg	23	R	pimplt>simplt
sourn	113	R(5)	vimplt>simplt
stret	155	I	packer
sumdpk	27	R	jchoke>vimplt>simplt
sumdpl	29	R	jchoke>vimplt>simplt
sumf	73	R	vexplt
sumg	74	R	vexplt
sumold	75	R	vexplt
sumvfx	190	R(3)	vovel>fwdrag
sumvgx	193	R(3)	vovel>fwdrag
taa	79	R	stvrpx svh2x2
tbb	80	R	stvrpx svh2x2
tcouri	71	R	phantv
tf	104	R	statep
timinv	70	R	phantv
tloc	13	R	vexplt vimplt
tloc2	14	R	vexplt vimplt
tnmins	45	R	simplt

**Table 10.55-1** Scratch Variable Names in Comdeck scrtch.hh

<b>Variable</b>	<b>fa(i)</b>	<b>Type</b>	<b>Description</b>
tnplus	46	R	simplt
tpdpdx	14	R	fwdrag
tquala	147	R	pimplt>simplt
trmm	49	R	phantv
trmm1	50	R	phantv
tsat	51	R	statep
tsater	82	I	statep
tt	1	R	statep
tter	32	I	statep
ttg	105	R	statep
tuf	24	R	pimplt>simplt
tug	25	R	pimplt>simplt
ubar	4	R	statep
ufnc	7	R	eqfinl
ugnc	6	R	eqfinl
us	54	R	statep
usubf	13	R	statep
usubfs	63	R	statep
usubg	14	R	statep
usubgs	64	R	statep
vagrg	15	R	vexplt vimplt
vbar	3	R	statep
velfjs	102	R	vfinl
velgjs	103	R	vfinl
velvcf	66	R	phantv
velvcg	67	R	phantv
vfa	77	R(6)	volvel
vfdpk	27	R	accum>ccfl>jchoke>packer>vexplt>preseq>vfinl

**Table 10.55-1** Scratch Variable Names in Comdeck scrtch.hh

<b>Variable</b>	<b>fa(i)</b>	<b>Type</b>	<b>Description</b>
vfdpl	29	R	accum>ccfl>jchoke>packer>vexplt>preseq>vfinl
vga	83	R(6)	volvel
vgdpk	28	R	accum>ccfl>jchoke>packer>vexplt>preseq>vfinl
vgdpl	30	R	accum>ccfl>jchoke>packer>vexplt>preseq>vfinl
vmgnx	93	R	vexplt
vngen	69	R	vexplt vimplt
voidaa	4	R	eqfinl
voidfa	88	R	vexplt vimplt
voidga	86	R	vexplt vimplt
voidgd	12	R	phantv>phantj
voidgk	19	R	vexplt
voidgl	21	R	vexplt
voidgu	11	R	phantv>phantj
vpgen	50	R	vexplt vimplt
vpgnx	47	R	vexplt
vrhof	13	R(6)	volvel vlvela
vrhog	19	R(6)	volvel vlvela
vruf	142	R	simplt
vrug	141	R	simplt
vrxg	51	R	simplt
vrxgn	11	R(5)	simplt
vs	53	R	statep
vsubf	11	R	statep
vsubfs	61	R	statep
vsubg	12	R	statep
vsubgs	62	R	statep
vvfx	1	R(6)	volvel vlvela
vvgx	7	R(6)	volvel vlvela

**Table 10.55-1** Scratch Variable Names in Comdeck scrtch.hh

Variable	fa(i)	Type	Description
xliqh	47	R	phantv
xncn	11	R	eqfinl
xvaph	48	R	phantv

## 10.56 separ.hh & separec.hh

The separ.hh comdeck is dynamic block for the separator component. It contains the specification statements for the separator variables and their equivalences with the fa array. The separec.hh comdeck contains a description of each of the variable names used in the separ.hh comdeck. The descriptions of the variables are given below in alphabetical order where the slot number in the fa array that the variable is equivalence to is given in the fa(i) column. The dimensions and type of the variable, real\*8, integer, logical, or character, are given in the type column.

**Table 10.56-1** Accumulator Component Variable Names in Comdeck separ.hh

Variable	fa(i)	Type	Description
cwfco	58	R	GE separator geometric quantities
cwgcu	59	R	GE separator geometric quantities
deldim	52	R	Dryer quantities
geaas	44	R	GE separator geometric quantities
geads	26	R	GE separator geometric quantities
geai	16	R	GE separator geometric quantities
gean	17	R	GE separator geometric quantities
geang	19	R	GE separator geometric quantities
gebbs	47	R	GE separator geometric quantities
gecks	38	R	GE separator geometric quantities
gedds	29	R	GE separator geometric quantities
geefflds	41	R	GE separator geometric quantities
gehds	32	R	GE separator geometric quantities
gehsks	35	R	GE separator geometric quantities
gerh	18	R	GE separator geometric quantities
gerr1	15	R	GE separator geometric quantities

**Table 10.56-1** Accumulator Component Variable Names in Comdeck separ.hh

<b>Variable</b>	<b>fa(i)</b>	<b>Type</b>	<b>Description</b>
gerrss	23	R	GE separator geometric quantities
gerws	20	R	GE separator geometric quantities
isepst	53	I	Separator designator
nsep	54	I	Separator designator
vdryl	50	R	Dryer quantities
vdryu	51	R	Dryer quantities
vover	13	R	Void limit for ideal vapor outflow. Standard separator component block
vunder	14	R	Void limit for ideal liquid fall back. Standard separator component block
xco	56	R	GE separator performance quantities
xcu	57	R	GE separator performance quantities
xim	55	R	GE separator performance quantities

## 10.57 sscntr.hh & sscntrc.hh

The sscntr.hh comdeck is dynamic block for the steady-state variables. It contains the specification statements for the steady-state variables and their equivalences with the fa array. The sscntrc.hh comdeck contains a description of each of the variable names used in the sscntr.hh comdeck. The descriptions of the variables are given below in alphabetical order where the slot number in the fa array that the variable is equivalence to is given in the fa(i) column. The dimensions and type of the variable, real\*8, integer, logical, or character, are given in the type column.

**Table 10.57-1** Steady-State Component Variable Names in Comdeck sscntr.hh

<b>Variable</b>	<b>fa(i)</b>	<b>Type</b>	<b>Description</b>
l24skp	-	I	Skip factor for steady-state block. Set in parameter statement in sscntr to 9. Note...packed as...(30 bit) = total count, (30 bit) = edit count
smndrh	5	I	Count of min d(rho*h)/dt for a volume
smnrho	7	I	Count of min (rho-rhom) for a volume
smxdrh	1	I	Count of max d(rho*h)/dt for a volume
smxrho	3	I	Count of max (rho-rhom) for a volume
uo	9	R	Old-time mixture u for a volume

## 10.58 ssiblk.hh & ssiblkc.hh

The ssiblk.hh comdeck is dynamic block for the steady-state self initialization variables. It contains the specification statements for the steady-state self initialization variables and their equivalences with the fa array. The ssiblkc.hh comdeck contains a description of each of the variable names used in the ssiblk.hh comdeck. The descriptions of the variables are given below in alphabetical order where the slot number in the fa array that the variable is equivalence to is given in the fa(i) column. The dimensions and type of the variable, real\*8, integer, logical, or character, are given in the type column.

**Table 10.58-1** Steady-State Self Initialization Variable Names in Comdeck ssiblk.hh

Variable	fa(i)	Type	Description
ssiskp	-	I	Skip factor, the number of words from compid to cmpiii Set in parameter statement to 9 ins ssiblk
npumps	1	I	No. of pump component, controller pairs
nstctl	2	I	No. of steam component, controller pairs
nfeeds	3	I	No. of feed component, controller pairs
compid	4	I	Component no. of regulated component
contid	5	I	Controller no. regulating compid
cvrtno	6	I	Controller type no.
cmptno	7	I	Component type no.
cmpvnm	8	R	Component search or control variable name
cmpvno	9	I	Component search or control variable no.
cmptrp	10	I	Component controlling trip no.
cmpiii	12	I	Time dependent data table type
pzpres	11	R	Pressurizer pressure

## 10.59 statc.hh & statcc.hh

The statc.hh comdeck is dynamic block for the advancement statistics variables. It contains the specification statements for the advancement statistics variables and their equivalences with the fa array. The statcc.hh comdeck contains a description of each of the variable names used in the statc.hh comdeck. The descriptions of the variables are given below in alphabetical order where the slot number in the fa

array that the variable is equivalenced to is given in the fa(i) column. The dimensions and type of the variable, real\*8, integer, logical, or character, are given in the type column.

**Table 10.59-1** Advancement Statistics Variable Names in Comdeck statec.hh

Variable	fa(i)	Type	Description
stccf1	3	I	Number of times junction used CCFL correlation in entire problem
stccf2	4	I	Number of times junction used CCFL correlation in this major print interval
stjck1	1	I	Number of times junction choked in entire problem
stjck2	2	I	Number of times junction choked in this major print interval
stjpk1	5	I	
stjpk2	6	I	
stjskp	-	I	Parameter set in statec.h, currently = 6
stlte1	10	I	Number of times volume had largest mass error in entire problem
stlte2	11	I	Number of times volume had largest mass error in this major print interval
strap1	24	I	Number of time step repeats due to air appearance in this volume in entire problem
strap2	25	I	Number of time step repeats due to air appearance in this volume in this major print interval
strcl1	22	I	Number of times Courant limit for this volume caused reduced time step in entire problem
strcl2	23	I	Number of times Courant limit for this volume caused reduced time step in this major print interval
strdc1	1	I	Number of repeated time steps in entire problem
strdc2	2	I	Number of repeated time steps in this major print interval
strdp1	26	I	Number of time step repeats due to pressure change in this volume in entire problem
strdp2	27	I	Number of time step repeats due to pressure change in this volume in this major print interval
strex1	14	I	Number of times state extrapolation in volume caused reduced time steps in entire problem
strex2	15	I	Number of times state extrapolation in volume caused reduced time steps in this major print interval
strpe1	18	I	Number of times water property error in volume caused reduced time steps in entire problem

**Table 10.59-1** Advancement Statistics Variable Names in Comdeck statec.hh

<b>Variable</b>	<b>fa(i)</b>	<b>Type</b>	<b>Description</b>
strpe2	19	I	Number of times water property error in volume caused reduced time steps during this major print interval
strte1	16	I	Number of times mass error in volume caused reduced time steps in entire problem
strte2	17	I	Number of times mass error in volume caused reduced time steps during this major print interval
strxl1	12	I	Number of times quality adjustment in volume caused reduced time steps in entire problem
strxl2	13	I	Number of times quality adjustment in volume caused reduced time steps in this major print interval
stsatp	4	I	Total number of advancements
stscl1	20	I	Number of times volume had smallest Courant limit in entire problem
stscl2	21	I	Number of times volume had smallest Courant limit in this major print interval
stscpu	6	R	CPU time required
stsdata	9	R	Sum of time steps for average over edit
stsdtm	7	R	Minimum time step during edit
stsdtx	8	R	Maximum time step during edit
stsjpt	3	I	Pointer to junction array
stslen	-	I	Parameter set in statec.h, currently = 9
stsreq	5	I	Total number of requested advancements
stsskp	-	I	Parameter set in statec.h, currently = 20
stvpk1	28	I	
stvpk2	29	I	

## 10.60 statec.hh & statecc.hh

The statec.hh comdeck is a specification block and /statec/ common block. It contains the state properties that are used in RELAP5. The statecc.hh comdeck contains a description of each of the variable names used in the statec.hh comdeck. The descriptions of the variables are given below in alphabetical

order. The dimensions and type of the variable, real\*8, integer, logical, or character, are given in the type column.

**Table 10.60-1** State Property Variable Names in Comdeck statec.hh

Variable	Type	Description
advn	R(5)	
cvaox	R(5)	Same as rax
dconst	R(5)	Diffusion coefficient at reference conditions for non-condensable gasses and steam
dcvax	R(5)	Same as rax
nctype	R(5)	
nonair	I	
nonar	I	
noncn	I	Number of non-condensable gasses
nondum	I	
nonhe	I	
nonhy	I	
nonkr	I	
nonni	I	
nonxe	I	
prop	R	Array for sth2x calls, also used for scratch
qn	R(5)	
rax	R(5)	Gas constant of non-condensable gas
s	R	Same as prop
tao	R	Term in $cv = cvao + dcva * (t - tao)$ where cv is heat capacity and t is temperature
thca	R(5)	
thcb	R(5)	
tref	R(5)	
uaox	R(5)	Term in $u = uao + \int (cv * dt)$ where u is internal energy
visao	R(5)	
wmolea	R(5)	Molecular mass of non-condensable gas

## 10.61 stcblk.hh & stcblkc.hh

The stcblk.hh comdeck is a specification block and /stcblk/ common block. It contains the fluid type that are used in RELAP5. The stcblkc.hh comdeck contains a description of each of the variable names used in the stcblk.hh comdeck. The descriptions of the variables are given below in alphabetical order. The dimensions and type of the variable, real\*8, integer, logical, or character, are given in the type column.

**Table 10.61-1** Fluid Type Variable Names in Comdeck stcblk.hh

Variable	Type	Description
ndxstd	I	Pointer in fa array to first word of steam tables for fluid with fluid number nfluid
nfluid	I	Fluid number: 1 = light water 2 = heavy water 3 = hydrogen 4 = lithium 5 = potassium 6 = helium 7 = nitrogen 8 = sodium 9 = NaK 10 = lithium-lead 11 = ammonia 12 = new light water (based on internal energy and pressure) 13 = glycerol 14 = blood

## 10.62 stcom.hh & stcomc.hh

The stcom.hh comdeck is a specification block and /stcblk/ common block. It contains the steam table parameters that are used in RELAP5. The stcomc.hh comdeck contains a description of each of the variable names used in the stcom.hh comdeck. The descriptions of the variables are given below in alphabetical order. The dimensions and type of the variable, real\*8, integer, logical, or character, are given in the type column.

**Table 10.62-1** Steam Table Parameters in Comdeck stcom.hh

Variable	Type	Description
it3bp	I	Base pointer to third table in steam tables for current fluid
it3p0	I	Pointer to zeroth word of third table in steam tables for current fluid
it4bp	I	Base pointer to fourth table in steam tables for current fluid
it5bp	I	Base pointer to fifth table in steam tables for current fluid

**Table 10.62-1** Steam Table Parameters in Comdeck stcom.hh

<b>Variable</b>	<b>Type</b>	<b>Description</b>
lstcom	I	Number of words (single precision) in the /stcom/ common block Set in parameter statement to 15 in stcom.hh
np	I	Number of pressures in steam tables for current fluid
nprpnt	I	Number of thermodynamic properties in steam tables times nt for the current fluid
nsp	I	Number of saturation pressures in steam tables for current fluid
nst	I	Number of saturation temperatures in steam tables for current fluid
nt	I	Number of temperatures in steam tables for current fluid
pcrit	R	Critical point pressure of current fluid
pmax	I	Maximum allowed pressure for current fluid
pmin	R	Minimum allowed pressure for current fluid
ptrip	R	Triple point pressure of current fluid
tcrit	R	Critical point temperature of current fluid
tmax	R	Maximum allowed temperature for current fluid
tmin	R	Minimum allowed temperature for current fluid
ttrip	R	Triple point temperature of current fluid
verit	R	Critical point volume of current fluid
vtrip	R	Triple point volume of current fluid

## 10.63 sysdatc.hh & sysdatm.hh

The sysdatc.hh comdeck is dynamic block for the system variables. It contains the specification statements for the system variables and their equivalences with the fa array. The sysdatm.hh comdeck contains a description of each of the variable names used in the sysdatc.hh comdeck. The descriptions of the variables are given below in alphabetical order where the slot number in the fa array that the variable is equivalence to is given in the fa(i) column. The dimensions and type of the variable, real\*8, integer, logical, or character, are given in the type column.

**Table 10.63-1** System Variable Names in Comdeck sysdatc.hh

<b>Variable</b>	<b>fa(i)</b>	<b>Type</b>	<b>Description</b>
isnononcond0	8	L	True if there are no noncondensables in the system. It is input on the 120 card.

**Table 10.63-1** System Variable Names in Comdeck sysdate.hh

<b>Variable</b>	<b>fa(i)</b>	<b>Type</b>	<b>Description</b>
isyskp	-	I	Skip factor for sysdate variable processing Set in a parameter statement to 12 in sysdate.hh
nusys	1	I	Number of hydrodynamic systems
sysel	3	R	Elevation of system reference volume
sysmaf	6	I	Material in system
sysmat	6	R	Material number in system. This and the previous word occupy the same location
sysnam	7	R	Name for system
sysopt	-	I	Old packed word, not used anymore
sysvol	2	I	Reference volume of system
tpfnha	9	R	Thermodynamic properties file name (5 Hollerith words) for sysmat, as read from 120-129 input cards

## 10.64 tdpptr.hh

The tdpptr.hh comdeck is dynamic block for the time dependent volume-junction pointers. It contains the specification statements for the system variables and their equivalences with the fa array. The descriptions of the variables are given below in alphabetical order where the slot number in the fa array that the variable is equivalence to is given in the fa(i) column. The dimensions and type of the variable, real\*8, integer, logical, or character, are given in the type column.

**Table 10.64-1** Time Dependent Volume-Junction Pointers in Comdeck tdpptr.hh

<b>Variable</b>	<b>fa(i)</b>	<b>Type</b>	<b>Description</b>
ntdpvs	1	I	Time-dependent volume-junction pointers

## 10.65 tmsrcm.hh

The tmsrcm.hh comdeck is a specification block and /tmsrcm/ common block. It contains a temporary common block for the steam table parameters that are used in RELAP5. The descriptions of the variables are given below in alphabetical order. The dimensions and type of the variable, real\*8, integer, logical, or character, are given in the type column.

**Table 10.65-1** Temporary Common Block for the Steam Table Parameters in Comdeck tmsrcm.hh

<b>Variable</b>	<b>Type</b>	<b>Description</b>
prop	R(26)	

**Table 10.65-1** Temporary Common Block for the Steam Table Parameters in Comdeck tmsrcm.hh

<b>Variable</b>	<b>Type</b>	<b>Description</b>
s	R(26)	

## 10.66 trnhlp.hh & trnhlpc.hh

The trnhlp.hh comdeck is a specification block and /trnhlp/ common block. It contains a common block for the some parameters that are used in RELAP5 transient calculation. The trnhlpc.hh comdeck contains a description of each of the variable names used in the trnhlp.hh comdeck. The descriptions of the variables are given below in alphabetical order. The dimensions and type of the variable, real\*8, integer, logical, or character, are given in the type column.

**Table 10.66-1** Temporary Common Block for the Steam Table Parameters in Comdeck trnhlp.hh

<b>Variable</b>	<b>Type</b>	<b>Description</b>
gerr	I	Estimated error in inversion process,
gerrs	I	Error at which scaling and solution order will be reevaluated
ixcofp	I	Pointer to array holding non zero values of matrix
ixip	I	Pointer to control array for matrix inversion subroutine
ixipr	I	Pointer indicating first non zero in row of solution matrix
ixirn	I	Similar function to ixirnr for previously inverted array
ixirnr	I	Pointer to array holding position of non zeros for matrix inversion subroutine
ixpv	I	Pointer to array for right hand side
ixw	I	Pointer to work array for matrix inversion subroutine
lhtsol	I	Used to determine scratch storage lengths in trnset
mtype	I	Type of matrix multiply, needed in call to matrix product subroutine
nnz	I	Number of non zero elements in solution matrix
nvr	I	Order of matrix, also number of non-time dependent volumes
sflag	I	Indicates whether new scaling and solution order needed

## 10.67 trpblk.h & trpbblkc.hh

The trpblk.hh comdeck is dynamic block for the trip components. It contains the specification statements for the trip components and their equivalences with the fa array. The trpbblkc.hh comdeck contains a description of each of the variable names used in the trpblk.hh comdeck. The descriptions of the variables are given below in alphabetical order where the slot number in the fa array that the variable is

equivalenced to is given in the fa(i) column. The dimensions and type of the variable, real\*8, integer, logical, or character, are given in the type column.

**Table 10.67-1** Trip Components in Comdeck trpblk.hh

Variable	fa(i)	Type	Description
inittrp0	11	L	
latchcode1	12	L	
ntlskp	-	I	Logical trip skip factor (parameter to 13 in trpblk.hh)
ntrcv1	18	I	Variable code, left side
ntrcv2	22	I	Variable code, right side
ntrnv1	19	I	Number part of variable code, left side
ntrnv2	23	I	Number part of variable code, right side
ntrpc1	20	I(2)	1: Block number of left variable 2: Offset to left variable during input, index to left variable during transient
ntrpc2	24	I(2)	1: Block number of right variable 2: Offset to right variable during input, index to right variable during transient
ntrpff	8	I	
ntrpnl	2	I	Number of logical trips
ntrpno	10	I	Trip number
ntrpnv	1	I	Number of variable trips
ntrpof	3	I	Offset to logical trips
ntrpop	11	I	Old packed word, not used anymore
ntrps1	4	I(2)	1: first trip number of terminate advancement 2: offset to first trip during input processing, index to first trip during transient
ntrps2	6	I(2)	1: Second trip number of terminate advancement 2: Offset to second trip during input processing, index to second trip during transient
ntrtr1	18	I(2)	1: Left operand trip number 2: Offset to left trip time during input processing, index to left trip time during transient
ntrtr2	20	I(2)	1: Right operand trip number 2: Offset to right trip time during input processing, index to right trip time during transient

**Table 10.67-1** Trip Components in Comdeck trpbblk.hh

Variable	fa(i)	Type	Description
ntvskp	-	I	Variable trip skip factor Parameter set to 18 in trpbblk.hh
opcode2431	17	I	
signbitlefttripnum4	15	L	
signbitrighttripnum5	16	L	
timeoflgleftvar2	13	L	
timeoflgrightvar3	14	L	
trpcon	26	R	Constant on right side
trptim	9	R	Time trip was set, also used as trip switch Negative if trip not set or false Positive if set or true and the value is the time the trip was set
trptm	1	R	Same variable as trptim but used with different subscripts Trptim used in loops over all trips. Trptm used when subscripting obtained from itrsen subroutine is used

## 10.68 tsctlc.hh

The tsctlc.hh comdeck is a specification block and /tsctlc/ common block. It contains a common block for the mass error limits that are used in RELAP5. The trnhlpc.hh comdeck contains a description of each of the variable names used in the trnhlp.hh comdeck. The descriptions of the variables are given below in alphabetical order. The dimensions and type of the variable, real\*8, integer, logical, or character, are given in the type column.

**Table 10.68-1** Mass Error Time Step Control Limits in Comdeck tsctlc.hh

Variable	Type	Description
errhi	R	Upper limit on mass error control See dtstep for common statement.
errllo	R	Lower limit on mass error control.

## 10.69 tstpct.hh & tstpctc.hh

The tstpct.hh comdeck is dynamic block for the time step control. It contains the specification statements for the time step control and their equivalences with the fa array. The tstpctc.hh comdeck

contains a description of each of the variable names used in the tstdpct.hh comdeck. The descriptions of the variables are given below in alphabetical order where the slot number in the fa array that the variable is equivalenced to is given in the fa(i) column. The dimensions and type of the variable, real\*8, integer, logical, or character, are given in the type column.

**Table 10.69-1** Trip Components in Comdeck tstdpct.hh

Variable	fa(i)	Type	Description
curclm	1	I	Limit value for card pointer
curcmi	3	I	Current counter for plot and minor edit
curcmj	4	I	Current counter for major edit
curcrs	5	I	Current counter for restart
curctl	2	I	Pointer to current card
curtmi	6	R	Current time for plot and minor edit (chngno(15))
curtmj	7	R	Current time for major edit (chngno(15))
curtrs	8	R	Current time for restart (chngno(15))
dtmax	10	R	Maximum time step
dtmin	9	R	Minimum time step
-----	10	I	Offset to control variable filndx(27) in itstck => index in tran referenced as tspctr(filndx(2)+1)
heathydrocoupledimplct2	13	L	True if hydro and heat transfer solutions coupled implicitly
heathydrotimestepequal1	12	L	True if heat transfer time step is set to hydro time step
heatstrcttmpblkcomittd6	17	L	True if heat structure temperature block is omitted in major edits
hydronearlyimplicit3	14	L	True if nearly implicit solution method for hydro
hydrotmeintgrtionsel5	16	L	True if on-line selection of time integration for hydro
istsppac10	21	L	not used
istsppac11	22	L	not used
majeditevrytmestp12	23	L	True if major edits every successful time step
masserrtimestepcntr0	11	L	True if mass error estimate is used to control time step
mnreditevrytmestp13	24	L	True if minor edits every successful time step
pltrcdevrytmestp14	25	L	True if plot records every successful time step

**Table 10.69-1** Trip Components in Comdeck tstdpct.hh

<b>Variable</b>	<b>fa(i)</b>	<b>Type</b>	<b>Description</b>
scndjunblkommittd7	18	L	True if second part of junction block is omitted in major edits
sscnvgnottested4	15	L	True if test for convergence of steady-state is not made
statsblkommittd9	20	L	True if statistics block is omitted in major edits
thrdfrthvolblkommittd8	19	L	True if third and fourth parts of volume block are omitted in major edits
tspctr	6	I	Control variable number for user controlled time step
tspend	8	R	End time for associated time step values and counters
tsppac (new called blhtsppac)	11	I	Old packed word, not used anymore
tsppmi	26	I	Minor edit and plot frequency (max dt steps/edit)
tsppmj	27	I	Major edit frequency
tspprs	28	I	Restart frequency
tspskp	-	I	Skip parameter Set in parameter statement to 21

## 10.70 turbin.hh & turbinc.hh

The turbin.hh comdeck is dynamic block for the turbine component. It contains the specification statements for the turbine component and their equivalences with the fa array. The turbinc.hh comdeck contains a description of each of the variable names used in the turbin.hh comdeck. The descriptions of the variables are given below in alphabetical order where the slot number in the fa array that the variable is equivalence to is given in the fa(i) column. The dimensions and type of the variable, real\*8, integer, logical, or character, are given in the type column.

**Table 10.70-1** Turbine Component Variable Names in Comdeck turbin.hh

<b>Variable</b>	<b>fa(i)</b>	<b>Type</b>	<b>Description</b>
turctr	14	I(2)	1: Turbine disconnect trip number 2: Offset for above during input, index during transient
turdef	24	R	Max. stage efficiency
tureff	22	R	Turbine efficiency
turfr	19	R	Turbine friction factor
turint	18	R	Turbine moment of inertia

**Table 10.70-1** Turbine Component Variable Names in Comdeck turbin.hh

<b>Variable</b>	<b>fa(i)</b>	<b>Type</b>	<b>Description</b>
turpow	20	R	Power developed by turbine
turrd <sub>s</sub>	23	R	Mean stage blade radius
turscm	13	I	Shaft component number
turtrq	21	R	Torque developed by turbine
turupj	16	I	Junction number upstream of stage
turvel	17	R	Turbine rotational velocity
turx	25	R	Stage reaction ratio

## 10.71 ufilef.hh, ufiles.hh, & ufilesc.hh

The ufilef.hh and ufiles.hh comdecks are specification blocks and /ufilef/ and /ufiles/ common blocks. They contain file names and unit numbers that are used in RELAP5. The ufilesc.hh comdeck contains a description of each of the variable names used in these two comdecks. The descriptions of the variables are given below in alphabetical order. The dimensions and type of the variable, real\*8, integer, logical, or character, are given in the type column.

**Table 10.71-1** File Name and Unit Numbers in Comdecks ufilef.hh and ufiles.hh

<b>Variable</b>	<b>Type</b>	<b>Description</b>
coupfl	I	
eoin	I	
filsch	C*40(nflsch)	Holds file names used in open statements (except for thermodynamic properties files) Default values set by data statements in blkdfa User supplied values can be optionally entered in some machine versions
inpout	I	
input	I	Input file
jbinfo	I	User file to be copied to job output file
nflsch	I	Number of file names, set to 16 in a parameter statement in ufilef.hh
output	I	Printed output file
plotfl	I	Scratch file for internal plot capability
rstplt	I	Restart-plot file
sth2xt	I	Used for all thermodynamic property files

**Table 10.71-1** File Name and Unit Numbers in Comdecks ufilef.hh and ufiles.hh

<b>Variable</b>	<b>Type</b>	<b>Description</b>
stripf	I	Strip file
tty	I	On-line screen file

## 10.72 usrvar.hh

The usrvar.hh comdeck is dynamic block for the user variables. It contains the specification statements for the user variables and their equivalences with the fa array. The descriptions of the variables are given below in alphabetical order where the slot number in the fa array that the variable is equivalence to is given in the fa(i) column. The dimensions and type of the variable, real\*8, integer, logical, or character, are given in the type column.

**Table 10.72-1** User Variable Names in Comdeck usrvar.hh

<b>Variable</b>	<b>fa(i)</b>	<b>Type</b>	<b>Description</b>
nindx	5	I	
ntabl	2	I	
ntabla	2	R	
nusvar	1	I	

## 10.73 voldat.hh & voldatc.hh

The voldat.hh comdeck is dynamic block for the volume variables. It contains the specification statements for the volume variables and their equivalences with the fa array. The voldatc.hh comdeck contains a description of each of the variable names used in the voldat.hh comdeck. The descriptions of the variables are given below in alphabetical order where the slot number in the fa array that the variable is equivalence to is given in the fa(i) column. The dimensions and type of the variable, real\*8, integer, logical, or character, are given in the type column.

**Table 10.73-1** User Variable Names in Comdeck voldat.hh

<b>Variable</b>	<b>fa(i)</b>	<b>Type</b>	<b>Description</b>
avol	10	R	Area of volume, three quantities, one per coordinate
betaff	58	R	Liquid isobaric coefficient of thermal expansion at bulk conditions
betagg	59	R	Vapor isobaric coefficient of thermal expansion at bulk conditions
boron	36	R	Boron density (mass of boron per cell volume)

**Table 10.73-1** User Variable Names in Comdeck voldat.hh

<b>Variable</b>	<b>fa(i)</b>	<b>Type</b>	<b>Description</b>
borono	37	R	Boron density previous time step
csubpf	60	R	Liquid specific heat capacity at constant pressure at bulk conditions
csubpg	61	R	Vapor specific heat capacity at constant pressure at bulk conditions
dfrnto	206	R	Location of thermal front (old-time)
dfront	205	R	Location of thermal front
diamv	16	R	Equivalent flow diameter Three quantities, one per coordinate
dl	13	R	Volume length, three quantities, one per coordinate
dlev	201	R	Location of two-phase mixture level
dlevo	202	R	Location of two-phase mixture level (old-time)
dotm	68	R	Bulk vapor generation rate per unit volume
dplev	231	R	Difference in pressure between level position and volume center
drfdp	77	R	Partial derivative of rhof with respect to pressure
drfduf	78	R	Partial derivative of rhof with respect to liquid specific internal energy
drgdp	79	R	Partial derivative of rhog with respect to pressure
drgdug	80	R	Partial derivative of rhog with respect to vapor specific internal energy
drgdxa	81	R	Partial derivative of rhog with respect to noncondensable quality
dtdp	87	R	Partial derivative of satt with respect to pressure
dtdug	88	R	Partial derivative of satt with respect to vapor specific internal energy
dtdxa	89	R	Partial derivative of satt with respect to noncondensable quality
dtdfp	82	R	Partial derivative of tempf with respect to pressure
dtdfuf	83	R	Partial derivative of tempf with respect to liquid specific internal energy
dtgdpg	84	R	Partial derivative of tempg with respect to pressure

**Table 10.73-1** User Variable Names in Comdeck voldat.hh

<b>Variable</b>	<b>fa(i)</b>	<b>Type</b>	<b>Description</b>
dtgdug	85	R	Partial derivative of tempg with respect to vapor specific internal energy
dtgdxa	86	R	Partial derivative of tempg with respect to noncondensable quality
dtidp	-	R	Interfacial temperature partial derivative with respect to p, fa(ivskp9+5)
dtiduf	-	R	Interfacial temperature partial derivative with respect to uf, fa(ivskp9+4)
dtidug	-	R	Interfacial temperature partial derivative with respect to ug, fa(ivskp9+3)
dtidxn	-	R	Interfacial temperature partial derivative with respect to quala, fa(ivskp9+6)
dttdp	90	R	
enthn	104	R(5)	Enthalpy of noncondensable source Five quantities, one per species
enths	110	R	Enthalpy of the solute source
extvnn	-	R	Extra volume data for programmer use, one for each volume To activate, add extvol and either extv20 (nn = 1->20) or extv100 (nn = 1->100) to the define file, fa(ivskp7+2)

**Table 10.73-1** User Variable Names in Comdeck voldat.hh

Variable	fa(i)	Type	Description
floreg	91	R	<p>Flow regime number in real format</p> <ul style="list-style-type: none"> <li>1. CTB high mixing bubbly</li> <li>2. CTT high mixing bubbly/mist transition</li> <li>3. CTM high mixing mist</li> <li>4. BBY bubbly</li> <li>5. SLG slug</li> <li>6. ANM annular-mist</li> <li>7. MPR mist-pre-CHF</li> <li>8. IAN inverted annular</li> <li>9. ISL inverted slug</li> <li>10. MST mist</li> <li>11. MPO mist-post-CHF</li> <li>12. HST horizontal stratified</li> <li>13. VST vertical stratified</li> <li>14. MWY ECC mixer wavy</li> <li>15. MWA ECC mixer wavy/annular-mist</li> <li>16. MAM ECC mixer annular-mist</li> <li>17. MMS ECC mixer mist</li> <li>18. MWS ECC mixer wavy/slug transition</li> <li>19. MWP ECC mixer wavy-plug-slug transition</li> <li>20. MPL ECC mixer plug</li> <li>21. MPS ECC mixer plug-slug transition</li> <li>22. MSL ECC mixer slug</li> <li>23. MPB ECC mixer plug-bubbly transition</li> <li>24. MBB ECC mixer bubbly</li> </ul>
fmurex	153	R(3)	Viscosity ratio for wall friction (one per coordinate)
frica	163	R(3)	Constant term in experimental friction correlation Three quantities, one per coordinate
fricb	166	R(3)	Multiplier term in experimental friction correlation Three quantities, one per coordinate
fricc	169	R(3)	Power term in experimental friction correlation Three quantities, one per coordinate
fshape	150	R	Wall friction shape factor (one per coordinate)
fstrt	131	R	Horizontal stratification interpolating factor
fwalf	132	R(3)	Liquid wall friction coefficient Three quantities, one per coordinate
fwalg	135	R(3)	Vapor wall friction coefficient Three quantities, one per coordinate
gaman	99	R(5)	noncondensable generation rate per unit volume Five quantities, one per species

**Table 10.73-1** User Variable Names in Comdeck voldat.hh

<b>Variable</b>	<b>fa(i)</b>	<b>Type</b>	<b>Description</b>
gamas	109	R	Solute generation rate per unit volume
gammac	73	R	Condensation rate per unit volume associated with wall heat transfer
gammaw	72	R	Vapor generation rate per unit volume associated with wall heat transfer
ggas	121	R(3)	Cell centered gas mass flux Three quantities, one per coordinate
qliq	124	R(3)	Cell centered liquid mass flux Three quantities, one per coordinate
gravv	194	R(3)	1: Coordinate of gravity along inertial x coordinate 2: Coordinate of gravity along inertial y coordinate 3: Coordinate of gravity along inertial z coordinate
hgf	161	R	Direct heating heat transfer coefficient per unit volume
hgfo	229	R	Direct heating heat transfer coefficient per unit volume previous time step
hgfos	235	R	Saved beginning of advancement direct noncondensable gas - liquid heat transfer coefficient for level model backup
hgrad	162	R	Used in radiation heat transfer from SCDAP components to volumes
hif	70	R	Liquid side interfacial heat transfer coefficient per unit volume
hifo	92	R	Liquid side interfacial heat transfer coefficient per unit volume previous time step
hifos	234	R	Saved beginning of advancement liquid side interfacial heat transfer coefficient for level model backup
hig	71	R	Vapor side interfacial heat transfer coefficient per unit volume
higo	93	R	Vapor side interfacial heat transfer coefficient per unit volume previous time step
higos	233	R	Saved beginning of advancement vapor side interfacial heat transfer coefficient for level model backup

**Table 10.73-1** User Variable Names in Comdeck voldat.hh

<b>Variable</b>	<b>fa(i)</b>	<b>Type</b>	<b>Description</b>
hsteam	55	R	Steam specific enthalpy at bulk conditions using partial pressure of steam
htsens	214	R	Heat transfer coefficient for sensible heat transfer between vapor/gas mixture and liquid
hvmix	146	R	Volume mixture enthalpy
hyanpr	219	R	Rotation matrix used to convert from local coordinates to fixed coordinates
hyarf	157	R	Liquid void fraction times density
hyarg	158	R	Vapor void fraction times density
hyaruf	159	R	Product of liquid void fraction, density, and internal energy
hyarug	160	R	Product of vapor void fraction, density, and internal energy
hydxc	173	R(6)	<p>1: Change along inertial x axis due to moving from face 1 to center of volume along local x coordinate r<sub>hydxc</sub></p> <p>2: Change along inertial x axis due to moving from center of volume to face 2 along local x coordinate r<sub>hydxc</sub></p> <p>3: Change along inertial x axis due to moving from face 3 to center of volume along local y coordinate r<sub>hydxc</sub></p> <p>4: Change along inertial x axis due to moving from center of volume to face 4 along local y coordinate r<sub>hydxc</sub></p> <p>5: Change along inertial x axis due to moving from face 5 to center of volume along local z coordinate r<sub>hydxc</sub></p> <p>6: Change along inertial x axis due to moving from center of volume to face 6 along local z coordinate r<sub>hydxc</sub></p>

**Table 10.73-1** User Variable Names in Comdeck voldat.hh

Variable	fa(i)	Type	Description
hydyc	179	R(6)	<p>1: Change along inertial y axis due to moving from face 1 to center of volume along local x coordinate r hydyc</p> <p>2: Change along inertial y axis due to moving from center of volume to face 2 along local x coordinate r hydyc</p> <p>3: Change along inertial y axis due to moving from face 3 to center of volume along local y coordinate r hydyc</p> <p>4: Change along inertial y axis due to moving from center of volume to face 4 along local y coordinate r hydyc</p> <p>5: Change along inertial y axis due to moving from face 5 to center of volume along local z coordinate r hydyc</p> <p>6: Change along inertial y axis due to moving from center of volume to face 6 along local z coordinate</p>
hydzc	185	R(6)	<p>1: Change along inertial z axis due to moving from face 1 to center of volume along local x coordinate r hydzc</p> <p>2: Change along inertial z axis due to moving from center of volume to face 2 along local x coordinate r hydzc</p> <p>3: Change along inertial z axis due to moving from face 3 to center of volume along local y coordinate r hydzc</p> <p>4: Change along inertial z axis due to moving from center of volume to face 4 along local y coordinate r hydzc</p> <p>5: Change along inertial z axis due to moving from face 5 to center of volume along local z coordinate r hydzc</p> <p>6: Change along inertial z axis due to moving from center of volume to face 6 along local z coordinate</p>
hyposv	191	R(3)	<p>1: Coordinate along x inertial axis of vector from center of rotation to center of volume r hyposv</p> <p>2: Coordinate along y inertial axis of vector from center of rotation to center of volume r hyposv</p> <p>3: Coordinate along z inertial axis of vector from center of rotation to center of volume</p>
imap (replaced with blhimap)	5	I(3)	Old packed word, not used anymore
invfnd	142	I	Index to inverted junction table

**Table 10.73-1** User Variable Names in Comdeck voldat.hh

<b>Variable</b>	<b>fa(i)</b>	<b>Type</b>	<b>Description</b>
invhtf	172	I	Index to inverted heat structure table
is3dvol17	318	L	true if volume if is multidimensional
isaccum5	242	L	true if this is an accumulator volume
isairrpt1	270	L	true if air has appeared and a partial backup is needed
isansinpflg27	330	L	true if this has the ANS model active in this volume
isbundle16	315	L	true if bundle in this volume
isbundle30	267	L	true if there is a bundle in this volume
iscoorddir14	312	L	true if this coordinate direction is being used in this volume
iscurrsaveflg9	278	L	saved current level model flag (iscurrvollvlflg7)
iscurrvollvlflg7	276	L	true if level is in this volume this time step
isdbgprntflg0	269	L	true if debug prints are desired
isequilflg1	238	L	true if equilibrium model is on in this volume
isexpfric12	306	L	true if experimental friction is being used in this volume
isflowregmnum1823	285	I	flow regime number
isgasappearance6	288	L	noncondensable gas appearance flag
isgodunovflg4	273	L	flags used for boron transport Godunov method
isgodunovflg5	274	L	flags used for boron transport Godunov method
ish2opackervol30	339	L	true if water packing model is active in this volume
ish2opck7	244	L	true if input specifies water packing is on in volume
ish2opckstat3	272	L	
ishorzstrat7	291	L	horizontal stratification flag
isininput3	240	L	
isinpvertstrat9	297	L	true if vertical stratification input flag is set
islamfricfactor26	327	L	true if laminar friction factor numerator is 64, false if laminar friction factor numerator is 96
isleveltrckinp28	333	L	true if level tracking model is active in this volume
islvlflg6	275	L	flag used in level model

**Table 10.73-1** User Variable Names in Comdeck voldat.hh

<b>Variable</b>	<b>fa(i)</b>	<b>Type</b>	<b>Description</b>
ismapinfo05	282	L	
ismetalapprnce24	321	L	not used
ismetalapprnce25	324	L	not used
isnegvoid12	281	L	
isnew2phaserr15	252	L	true if error in two-phase region in this volume for this time step
isnewcnvgerr16	253	L	true if non convergence of noncondensable solution iterations in this volume for this time step
isnewdrvqtyerr18	255	L	true if negative derived quantities in this volume for this time step
isnewliqerr14	251	L	true if error in liquid phase in this volume for this time step
isnewlrgstmaserr12	249	L	true if largest mass error in this volume for this time step
isnewmasserr9	246	L	true if mass error in this volume for this time step
isnewnegpres8	245	L	true if negative pressure in this volume for this time step
isnewqualovrn11	248	L	true if quality exceeds 0 or 1 in this volume for this time step
isnewsonvelerr17	254	L	true if negative sonic velocity in this volume for this time step
isnewvaperr13	250	L	true if error in vapor phase in this volume for this time step
isnewxtraperr10	247	L	true if state properties extrapolation error in this volume for this time step
isnoncond11	280	L	
isold2phaserr26	263	L	true if error in two-phase region in this volume for this time step
isoldcnvgerr27	264	L	true if non convergence of noncondensable solution iterations in this volume for this time step
isolddrvqtyerr29	266	L	true if negative derived quantities in this volume for this time step
isoldliqerr25	262	L	true if error in liquid phase in this volume for this time step

**Table 10.73-1** User Variable Names in Comdeck voldat.hh

<b>Variable</b>	<b>fa(i)</b>	<b>Type</b>	<b>Description</b>
isoldrgstmaserr23	260	L	true if largest mass error in this volume for this time step
isoldmasserr20	257	L	true if mass error in this volume for this time step
isoldnegpres19	256	L	true if negative pressure in this volume for previous time step
isoldqualovrn22	259	L	true if quality exceeds 0 or 1 in this volume for this time step
isoldsonvelerr28	265	L	true if negative sonic velocity in this volume for this time step
isoldvaperr24	261	L	true if error in vapor phase in this volume for this time step
isoldxtraperr21	258	L	true if state properties extrapolation error in this volume for this time step
ispresschngrpt2	271	L	true if a pressure change repeat when air appeared
isprevsaveflg10	279	L	saved previous level model flag (isprevvollvlflg8)
isprevvollvlflg8	277	L	true if level was in this volume in previous time step
ispump6	243	L	true if this is a pump volume
isreflood29	336	L	true if reflood model is active in this volume
issstretch8	294	L	stretch flag
istdpvol0	237	L	true if this is a time-dependent volume
isthermfrnt2	239	L	true if thermal front is in this volume
isvapdis4	241	L	
isvertstrat10	300	L	vertical stratification flags
isvertstrat11	303	L	vertical stratification flags
iswallfricinp13	309	L	true wall friction is on in this volume
ivsk1	-	I	Set in a parameter statement to 340 in voldat.hh
ivsk2	-	I	Set to 0
ivsk3	-	I	Set to ivsk1 + ivsk2 in voldat.hh
ivsk4	-	I	Used for selap in voldat.hh Set to 0 for RELAP5
ivsk5	-	I	Set to ivsk3 + ivsk4 in voldat.hh

**Table 10.73-1** User Variable Names in Comdeck voldat.hh

<b>Variable</b>	<b>fa(i)</b>	<b>Type</b>	<b>Description</b>
ivsk6	-	I	Set to 0 in voldat.hh
ivsk7	-	I	Set to ivsk5 + ivsk6 in voldat.hh
ivsk8	-	I	Set to 20 in voldat.hh if extvol and extv20 are defined Set to 100 in voldat.hh if extvol and extv100 are defined
ivsk9	-	I	Set to ivsk7 + ivsk8 in voldat.hh
ivsk10	-	I	Set to 5 in voldat.hh
ivskp	-	I	Volume skip factor Parameter set at end of voldat.hh to ivsk9 + ivsk10)
jjet	236	I	Address in the junction block of the jet junction for this v
nvols	1	I	Number of volumes
p	25	R	Average pressure
pecltv	149	R	Volume Peclet number
po	26	R	Average pressure previous time step
pps	67	R	Vapor partial pressure
ppso	118	R	Vapor partial pressure (old-time)
ptans	147	R	Pitch between fuel plates (ANS)
q	74	R	Total heat transfer rate from wall to fluid
quala	34	R	noncondensable quality.
qualan	94	R(5)	noncondensable mass fraction Five quantities, one per species
qualao	35	R	noncondensable quality previous time step
quale	39	R	Equilibrium quality
qualno	112	R(5)	noncondensable mass fraction previous time step Five quantities, one per species
quals	38	R	Static quality
qwf	75	R	Heat transfer rate from wall to liquid
qwg	76	R	Heat transfer rate from wall to vapor
recipv	9	R	Reciprocal of volume (v), zero if v is zero

**Table 10.73-1** User Variable Names in Comdeck voldat.hh

<b>Variable</b>	<b>fa(i)</b>	<b>Type</b>	<b>Description</b>
recrit	22	R	Critical Reynolds number Three quantities, one per coordinate friction factor = constant; see roughv
rho	40	R	Total density
rhof	43	R	Liquid density Warning: the ordering of rhof and rhog must be maintained since fidis assumes this order
rhog	44	R	Vapor density Warning: the ordering of rhof and rhog must be maintained since fidis assumes this order
rhogo	117	R	Vapor density previous time step
rhom	41	R	Total density for mass error check
rhoo	42	R	Total density previous time step
roughv	19	R	Wall roughness factor for direction 1, as input Reset in icmpn1 to Colebrook full turbulence friction factor
sathf	56	R	Liquid specific enthalpy at saturation conditions
sathg	57	R	Vapor specific enthalpy at saturation conditions
satt	45	R	Saturation temperature based on the steam partial pressure
savebtflag	2	I	
sigma	64	R	Surface tension
sinb	143	R	Sine function of volume vertical angle Three quantities, one per coordinate
sounde	54	R	Homogeneous equilibrium sound speed Also, used for scratch in hydro
span	148	R	Length of fuel plates (ANS)
srcamn	215	R	
sth2xv	141	I	Index data for sth2x water property subroutines
temp	45	R	Used in r-level subroutines and is equivalenced to satt
tempf	46	R	Liquid temperature
tempg	47	R	Vapor temperature

**Table 10.73-1** User Variable Names in Comdeck voldat.hh

<b>Variable</b>	<b>fa(i)</b>	<b>Type</b>	<b>Description</b>
thconf	65	R	Liquid thermal conductivity
thcong	66	R	Vapor thermal conductivity
tiengv	218	R	Total internal energy in volume Includes both phases and noncondensables
tintf	213	R	Interphase temperature when noncondensable is present, saturation temperature at total pressure otherwise
tmassv	217	R	Total mass in volume Includes both phases and noncondensables
tsatt	156	R	Saturation temperature based on the total pressure
ttempi	-	R	Interfacial temperature, at fa(ivsk9+2)
uf	27	R	Liquid specific internal energy
ufao	209	R	Liquid internal energy above the thermal front (old-time)
ufbo	211	R	Liquid internal energy below the thermal front (old-time)
ufla	208	R	Liquid internal energy above the thermal front
uflb	210	R	Liquid internal energy below the thermal front
ufo	28	R	Liquid specific internal energy previous time step
ug	29	R	Vapor specific internal energy
ugo	30	R	Vapor specific internal energy previous time step
ustm	119	R	Vapor specific internal energy at pps and tempg with non-condensable present
ustmo	120	R	Vapor specific internal energy at pps and tempg with non-condensable present (old-time)
v	8	R	Volume
vapgen	228	R	Total vapor generation, sum of bulk vapor generation and vapor generation from flashing and condensation $vapgen = dotm + gammaw + gammac$
vapgno	69	R	Vapor generation rate per unit volume previous time step
vcnfnd	230	I	Index to component for volume

**Table 10.73-1** User Variable Names in Comdeck voldat.hh

<b>Variable</b>	<b>fa(i)</b>	<b>Type</b>	<b>Description</b>
vctrl	-	I	Old packed word, not used anymore
vctrld	139	I	Index to diagonal matrix element
vctrln	138	R	Position of volume in volume block
vctrls	140	I	Index to volume scratch space
vctrlx (replaced with blhvctrlx)	216	I	Old packed word, not used anymore
velf	48	R(3)	Average liquid velocity in a volume Three quantities, one per coordinate
velfo	127	R	Volume average liquid velocity previous time step x direction only
velfoo	129	R	Volume average liquid velocity previous time step but one x direction only
velg	51	R(3)	Average vapor velocity in a volume Three quantities, one per coordinate
velgo	128	R	Volume average vapor velocity previous time step x direction only
velgoo	130	R	Volume average vapor velocity previous time step but one x direction only
vfront	207	R	Velocity of thermal front
viscf	62	R	Liquid viscosity
viscg	63	R	Vapor viscosity
vlev	203	R	Velocity of two-phase level movement
vlevo	232	R	
vo	111	R	Volume previous time step
vodgoo	212	R	Vapor void fraction at old-time step (n - 1)
voidao	198	R	Void fraction above the mixture level (old-time)
voidbo	200	R	Void fraction below the mixture level (old-time)
voidf	31	R	Liquid void fraction
voidg	32	R	Vapor void fraction
voidgo	33	R	Vapor void fraction previous time step (n)
voidla	197	R	Void fraction above the mixture level

**Table 10.73-1** User Variable Names in Comdeck voldat.hh

Variable	fa(i)	Type	Description
voidlb	199	R	Void fraction below the mixture level
vollev	204	R	Position of level within volume
volmat	3	I	Fluid type in volume
volno	4	I	Volume number for editing

## 10.74 vreqs.hh & vreqd.hh

The vreqs.hh comdeck is the specification block and vreqs.hh comdecks is the data deck containing the names of the variables that can be plotted using XMGR5. These variables are processed by the wrplid subroutine. The descriptions of the variables are given below in alphabetical order. The dimensions and type of the variable, real\*8, integer, logical, or character, are given in the type column.

**Table 10.74-1** Plot Variable Names in Comdecks vreqs.hh and vreqd.hh

Variable	Type	Description
t1	C*8(19)	1: time 2: cputime 3: emass 4: tmass 5: dt 6: stdtrn 7: dternt 8: count 9: errmax 10: testda 11: rrangl 12: rrangw 13: rranga 14: rrdisp 15: rrdisv 16: rrdisa 17: rromeg 18: rrdmeg 19: rktpow3d

**Table 10.74-1** Plot Variable Names in Comdecks vreqs.hh and vreqd.hh

<b>Variable</b>	<b>Type</b>	<b>Description</b>
t2	C*8(70)	1: rho 2: rhof 3: rhog 4: uf 5: ug 6: voidf 7: voidg 8: velf 9: velg 10: p 11: quals 12: quale 13: q 14: qwg 15: tempf 16: tempg 17: sounde 18: vapgen 19: quala 20: boron 21: sattemp 22: floreg 23: rhom 24: hsteam 25: sathf 26: sathg 27: betaf f28: betagg 29: csubpf 30: csubpg 31: viscf 32: viscg 33: sigm a34: theconf 35: thcong 36: pps 37: hif 38: hig 39: gammaw 40: gammac 41: drfdp 42: drfduf 43: drgdip 44: drgdug 45: drgdx 46: dtfdp

**Table 10.74-1** Plot Variable Names in Comdecks vreqs.hh and vreqd.hh

<b>Variable</b>	<b>Type</b>	<b>Description</b>
t2 (continued)	C*8(70)	47: dtfduf 48: dtgdpr 49: dtgdug 50: dtgdxar 51: dtdp 52: dtdu 53: dtdxar 54: fwalf 55: fwalg 56: avol 57: hvmix 58: pecltv 59: vvol 60: tsatt 61: hyposv 62: gravv 63: vollev 64: voidla 65: voidlb 66: tmassv 67: tiengv 68: gammal 69: hgf 70: qualhy

**Table 10.74-1** Plot Variable Names in Comdecks vreqs.hh and vreqd.hh

<b>Variable</b>	<b>Type</b>	<b>Description</b>
t3	C*8(26)	1: velfj 2: velg j3: rhofj 4: rhogj 5: uffj 6: ugj 7: mflowj 8: voidfj 9: voidgj 10: qualaj 11: fij 12: formfj 13: formgj 14: xej 15: sonicj 16: c0j 17: vgjj 18: florgj 19: iregj 20: voidj 21: flenth 22: chokef 23: fwalfj 24: fwalgj 25: fjunft 26: fjunrt
t4	C*8(10)	1: htvat 2: htrnr 3: htchf 4: htffc 5: httemp 6: htmode 7: htreg 8: htgamw 9: stant 10: pecl

**Table 10.74-1** Plot Variable Names in Comdecks vreqs.hh and vreqd.hh

<b>Variable</b>	<b>Type</b>	<b>Description</b>
t5	C*8(36)	1: pmpvel 2: pmphead 3: pmptrq 4: vlvarea 5: vlvstem 6: acttank 7: acvliq 8: acvdm 9: acqtank 10: acrhon 11: turpow 12: turtrq 13: turvel 14: tureff 15: przlvl 16: pmpmt 17: pmpnrt 18: theta 19: omega 20: betav 21: ahfgtf 22: ahfgtg 23: avgtg 24: ahftg 25: acpgtg 26: acvgtg 27: aviscn 28: acpnit 29: ahgtf 30: dmgd 31: xi 32: xco 33: xcu 34: dim 35: gdry 36: cdim

**Table 10.74-1** Plot Variable Names in Comdecks vreqs.hh and vreqd.hh

<b>Variable</b>	<b>Type</b>	<b>Description</b>
t6	C*8(38)	1: rktpow 2: rkfpow 3: rkgapow 4: rkreac 5: rkrecper 6: rkpowk 7: rkpowa 8: rkotpow 9: rkofipow 10: rkogapow 11: rkorecpr 12: rkopowa 13: rkopowk 14: rkozntpw 15: rkoznfip 16: rkozngap 17: rkoznpwk 18: rkoznpwa 19: rkondfip 20: rkophi 21: rkobk 22: rkobtb 23: rkod 24: rkosiga 25: rkosigf 26: rkosigs1 27: rkosigs2 28: rkosigs3 29: rkozntm 30: rkoznalp 31: rkoznden 32: rkoznbor 33: rkozntf 34: rkocrpsn 35: rkocracf 36: rkocrdcf 37: rkondfpd 38: rkondrfp
t7	C*8	cntrlvar
t8	C*8(5)	1: zqbot 2: zqtop 3: fines 4: tchfqf 5: trewet

**Table 10.74-1** Plot Variable Names in Comdecks vreqs.hh and vreqd.hh

<b>Variable</b>	<b>Type</b>	<b>Description</b>
t9a	C*8(12)	1: bgth 2: bgnhg 3: bgmct 4: bgtprs 5: bgtprn 6: bgthq 7: bgthu 8: bgthqu 9: crucb 10: repool 11: shqin 12: shqout
t9b	C*8(7)	1: achdpn 2: pgas 3: wdtqlp 4: zbtrub 5: ztprub 6: zbtcoh 7: ztpcoh

**Table 10.74-1** Plot Variable Names in Comdecks vreqs.hh and vreqd.hh

<b>Variable</b>	<b>Type</b>	<b>Description</b>
t9c	C*8(27)	1: brchv 2: cggivy 3: damlev 4: dzfrcq 5: effoxd 6: h2oxd2 7: hoop 8: oxdeo 9: rci 10: rco 11: rnalf 12: rnoxd 13: rocrst 14: rpel 15: ruliq 16: wfrosr 17: wfrouo 18: wfrozr 19: wremsr 20: wremuo 21: wremzr 22: qscd 23: qwgscd 24: qflux0 25: qfg0 26: hfixf 27: hfixg
t9d	C*8	cadet
t9e	C*8(2)	1: dcreph 2: dcrepc

**Table 10.74-1** Plot Variable Names in Comdecks vreqs.hh and vreqd.hh

<b>Variable</b>	<b>Type</b>	<b>Description</b>
t10a	C*8(28)	1: tmpdmx 2: hgtdeb 3: pdbtot 4: twalmx 5: tmpdav 6: masliq 7: liqavg 8: massz r9: massu 10: massfe 11: massag 12: masb4c 13: masuo2 14: maszo2 15: massal 16: massli 17: masscd 18: liqfe 19: liqzr 20: liqag 21: liquo2 22: liqzo2 23: denrgy 24: csenrg 25: intpow 26: intq 27: debqup 28: mppden
t10b	C*8(11)	1: tmptcou 2: fpdeb 3: oxthk 4: afbulk 5: fracml 6: tmltel 7: pore 8: powdb 9: tothtc 10: gaphtc 11: mphtc
t11	C*8(2)	1: sysmer 2: systms

## 10.75 zalfag.hh

The zalfag.hh comdeck is a statement function block. It contains the statement function to calculate void fraction from quality, vapor specific volume, and liquid specific volume. The formula is given below.

$$\alpha_g(x, v_g, v_f) = \max \left\{ 0, \min \left[ 1, \frac{xv_g}{v_g + x(v_g - v_f)} \right] \right\} \quad (10.75-1)$$

where

$\alpha_g$  = void fraction

$x$  = quality

$v_g$  = vapor specific volume

$v_f$  = liquid specific volume

zalfag.hh

## 11 INP PACKAGE

The INP package is a convenient set of data input subroutines for use with FORTRAN 77 programs. The INP package that is used in RELAP5 came from the INPF package that was developed at Bettis Atomic Power Laboratory (BAPL) in the 1960-1970 period.<sup>11.4-1,11.4-3</sup> It was converted to the IBM-360-370 by Dick Wagner when the PDQ program was imported from BAPL<sup>11.4-2</sup> and renamed INP. It has later been converted to work on the Cray computer as well as numerous UNIX workstations. For the user, INP offers: free-form input, card numbers to identify each card, arbitrary ordering of input cards except for replacement cards, arbitrary use of comments both on the data card and as separate cards, easy preparation of multiple input cases in which only a few cards are changed, and a listing of the card input data. For the programmer, INP offers a convenient method of implementing a user-oriented card input scheme and includes: extensive checking of the number of entries on a card, the mode (integer, floating point, or alphanumeric) of the entries, automatic expansion of sequential and overlay type of input data, deletion of un-needed cards, checking for extraneous input, and the ability to detect several types of input errors. In today's terminology, we should be using record instead of card, but we will stick to the ancient terminology because of its historical significance to this package.

The following two sections will cover INP as seen from a user's point of view and from a programmer's point of view. A third section summarizes the error messages and the table-list structure.

### 11.1 User Aspects of INP

The data file contains input for one or more problem sets. No relationship is assumed between problem sets. Each problem set consists of one or more cases in which the input data for the second and succeeding cases consist of the data from the previous case plus any modification cards entered for the present case. Thus, we have the hierarchy as shown in **Table 11.1-1**. The individual cases are separated by a slash (/) card, and the final case is terminated by a period (.) card. The period card always serves as a separator between sets.

**Table 11.1-1** Hierarchy of the File, Set, and Case Levels in INP Input Data File

File Level	Set Level	Case Level
File	Set 1	Case 1.1
		Case 1.2
		Case 1.3
		Case 1.4
	Set 2	Case 2.1
		Case 2.2
		Case 2.3

A list containing the card sequence number and an image of each card is printed to the output file at the beginning of the printed output for each case. The card sequence number starts at one for each case. The first line of the listing contains the heading, LISTING OF INPUT FOR CASE n, where n is the case number. A RELAP5 input deck normally contains only one case. Even though the capability to create multiple cases and multiple sets in one input deck, it is rarely done by the RELAP5 users. For this reason, most users only have a single title card at the beginning of the deck and a single end-of-set card (period card) at the end of the deck.

### **11.1.1 Title Cards**

Each case can have zero, one, or more title cards. The title card has an equal (=) character as the first non-blank character in the card. Any alphanumeric characters can be in the remainder of the card. The title card is used as a page header. It is recommended that one title card be entered for each case. If more than one title card is entered in a case, the contents of the last title card is used for the page heading.

### **11.1.2 Comment Cards**

Comment cards have either an asterisk (\*) or a dollar sign (\$) as the first non-blank character on the card. Any information may be entered in the remainder of the card. Blank cards are considered comment cards. There is no processing of comment cards other than listing them.

### **11.1.3 End Cards**

The end-of-case card has a slash (/) as the first non-blank character in the card. Comments may follow the slash. The end-of-set card has a period (.) as the first non-blank character in the card. Comments may follow the period. Another terminator is the end-of-file mark on the file.

### **11.1.4 Data Cards**

All cards other than title, comment, end-of-case, or end-of-set cards are data cards. Data cards can contain a variable number of fields. The types of fields on a data card are related as follows:

```
alphanumeric
numeric
    decimal, 0-9,+,-,. and does not start with 0
    integer, no decimal point or exponent
    floating point, decimal point or exponent or both
    octal, starts with 0, contains [0-7]
    hexadecimal, starts with x, contains [0-9, a-f]
```

Alphanumeric fields are separated by commas and numeric fields are separated by one or more blanks or commas.

An alphanumeric field starts with any character other than a digit (0-9), sign (+-), decimal point (.), asterisk (\*), dollar sign (\$), slash (/), or apostrophe('). The field is terminated by a comma (,) or the end of

the card. All characters except commas are allowed. Imbedded blanks are considered part of the field and do not terminate the field.

A decimal field starts with a digit (0-9), sign (+ or -), or decimal point (.). A decimal integer field has no decimal point or exponent part; it only has an optional sign followed by digits, e.g. 1234 or -1234. A decimal floating point field has a decimal point, exponent part, or both. The exponent part has a sign, an e(E) or d(D), or an e(E) or d(D) followed by a sign that is followed by a number. The number gives the power of ten to multiply the preceding number by, e.g., 0.123 or -0.123 or 1.23-3 or 1.45+3 or 1.25e2 or 4.67D-9.

An octal field starts with the digit 0 and can contain digits (0-7) only.

A hexadecimal field starts with an x(X) and can contain the digits (0-9) and the letters (a-f, A-F).

Zero is unique in that it can be entered as either a decimal floating point number, 0.0, or as a decimal integer, 0.

### **11.1.5 Example Cards**

Here are some examples of cards from a RELAP5 input deck.

```
= Edward's pipe problem base case with extras
* Configuration Control Problem
0000100 new transnt
301 p 3010000
0000201 0.020,1.0-7,.001,7, 2,10,100
505 velfj 3010000 ge velgj 3010000 0.0 n
0030000 edward's pipe
0030101 4.56037-3,20
0050201 0.0,1.0+5,1.0 100.0,1.0+5,1.0
20100300 tbl/fctn 2 3
. end of case
```

## **11.2 Programmer Aspects of INP**

The INP package is a set of thirteen subroutines that are useful in any FORTRAN program to handle input data. Four subroutines, inp, inp2, inplnk, and inpmid, form a basic set and are used together. Seven of the subroutines provide specialized input processing and storage capability, and the remaining two subroutines are used internally for packing and unpacking format indicators into a single word.

One call to the inp subroutine gets each set or case from the input file. The inp subroutine reads and converts the data, removes duplicate cards, and forms an input buffer. This input buffer consists of a sorted table array of card numbers that are cross referenced to a list array containing data words taken from the cards together with mode words generated during input conversion. The type of conversion is dictated by the card fields themselves and is stored in the mode word. There are three types of conversions, alphanumeric, real, and integer. The ordering of the cards in a deck, other than the initial title card and the final end-of-set card, is not important, except for duplicate card numbers. If there are two cards having the

same card number, the last card is used. The last title card is used in the output title line, and several card listing options are provided.

Subroutine inplnk locates one card at a time and provides the linkage between a FORTRAN program and the input buffer containing the table array and list array prepared by inp. Subroutine inpmod is used to check the appropriate mode of the data against a specified list, also one card at a time. Subroutine inp2 is used to move and check data from the input buffer array through calls to inplnk and inpmod. The inp2 subroutine can be used to process data from a single card or from a set of cards within a specified range. It also checks for the specified minimum and maximum number of items and the mode of the data. It then transfers the data to the array specified in the call statement.

Subroutine inp4 executes a do loop which includes calls on inp2, modifying the parameters for each subsequent call by specified increments. Subroutine inp5 is similar to inp4 in that it will process more than one set of cards. However, it has the additional capability to expand either of two input types and store the data in either of two modes. Subroutines inp6 and inp7 are used for printing error comments. Function inp8 is used to determine if there are cards in the table which were not referenced by inplnk. Finally, functions inp9 and inp10 can be used to delete cards from the table array and list array that have been either selected or processed by inplnk.

The INP package includes facilities for expanding data that has been specified in a compact notation on the card. The expansion may be of sequential or overlay type, and the expanded data may be stored in either dense or scattered mode. In all cases, it is assumed that the data is to be associated with some index value, e.g., a junction number or a volume number. The compact notation simplifies input preparation when the data for several indices is the same, and the expansion then reproduces the data and associates it with each of these index values.

### 11.2.1 Sequential Expansion

The basic unit of input for sequential expansion is the set of data to be expanded followed by the terminal index for the expansion. For example, if the input consists of the values:

1.1, 1.2, 1.3, 3, 1.4, 1.5, 1.6, 5

the first set is (1.1, 1.2, 1.2) with terminal index 3 and the second set is (1.4, 1.5, 1.6) with terminal index 5. The calling program specifies the base index (usually zero) which is one less than the starting index value. Assuming a zero base index for the above input, the first set is reproduced three times and associated with indices 1, 2, and 3. The second set is reproduced twice and associated with indices 4 and 5. The terminal indices must form an increasing sequence, and the calling program may specify an upper bound for this sequence.

This method of expansion is used quite often in RELAP5 input decks for components. For example, if the user wanted to input the same volume length, 0.45 m, for all 15 volumes in pipe component number 102, then the input card is:

10200301 0.45 15

### 11.2.2 Overlay Expansion

The basic unit of overlay expansion is the set of data to be expanded preceded by the initial index for the expansion and followed by the terminal index for the expansion. For example, if the input consists of the values:

3, 2.1, 2,2, 6, 5, 2.3, 2,4, 8

the first set is (2.1, 2.2) with indices 3 and 6. The second set is (2.3, 2.4) with indices 5 and 8,. This example show that the index ranges for the various sets can overlap. The sets are expanded sequentially and the expansion of each set overlays or replaces any data previously associated with its indices. For the above example, the complete expansion associates the first set with indices 3 and 4 while the second set is associated with indices 5, 6, 7, and 8. The calling program may specify both a lower bound for the initial index and an upper bound for the terminal index. The base index value is always zero.

This method of expansion is not used in RELAP5. It is only included here to complete the description of the INP package.

### 11.2.3 Dense Storage

When dense storage is used, the storage array is assumed to contain a separate store vector for each value of the index. The store vector must be a least as large as the input set, and the calling program must specify the starting location in the store vector, i.e., the position in the store vector at which the input set is to begin. For example, consider the sequential expansion input 3.1, 3,2, 1, 3.3, 3.4, 3 together with a store vector length of 4 and a starting location of 2. The expansion results in the storage array

---, 3.1, 3.2, ---  
---, 3.1, 3.2, ---  
---, 3,3, 3,4, ---

where three dashes indicates a location that is not altered by the expansion. It should be noted that the storage array is actually one-dimensional, and the above two-dimensional representation has been used only to demonstrate its format.

### 11.2.4 Scattered Storage

When scattered storage is used, the storage array is assumed to contain a separate store vector for each element of the input set. The store vectors are of a specified length and contain a single data value for each value of the index. For example, consider the overlay expansion input

2, 4.1, 4.2, 4.3, 3, 5, 4.4, 4.5, 4.6, 5

together with a store vector length of 6. The expansion results in the storage array

---, 4.1, 4.1, ---, 4.4, ---  
---, 4.2, 4.2, ---, 4.5, ---  
---, 4.3, 4.3, ---, 4.6, ---

Here again, the storage array is actually one-dimensional and the three dashes indicate and unaltered location. Note that the first value in each input set is stored in the first line (first store vector), the second value is stored in the second line, and so forth. By specifying a starting location (other than 1) in the storage array, the calling program can force the expansion to begin in the store vector other than the first. For example, with a starting location of 7, the above expansion becomes:

```
---, ---, ---, ---, ---, ---
---, 4.1, 4.1, ---, 4.4, ---
---, 4.2, 4.2, ---, 4.5, ---
---, 4.3, 4.3, ---, 4.6, ---
```

In the following descriptions, variable arrays are named for their real or integer formats. Arrays l3 and a3 are equivalent and arrays l1 and l2 will probably have equivalent real names in the calling program.

### **11.2.5 efiless - File Unit Numbers**

The efiless.hh comdeck contains the specification statements for the input, output, inout, and tty files. The efilesd.hh comdeck contains the data statements for initializing the unit numbers, which are given in the following list:

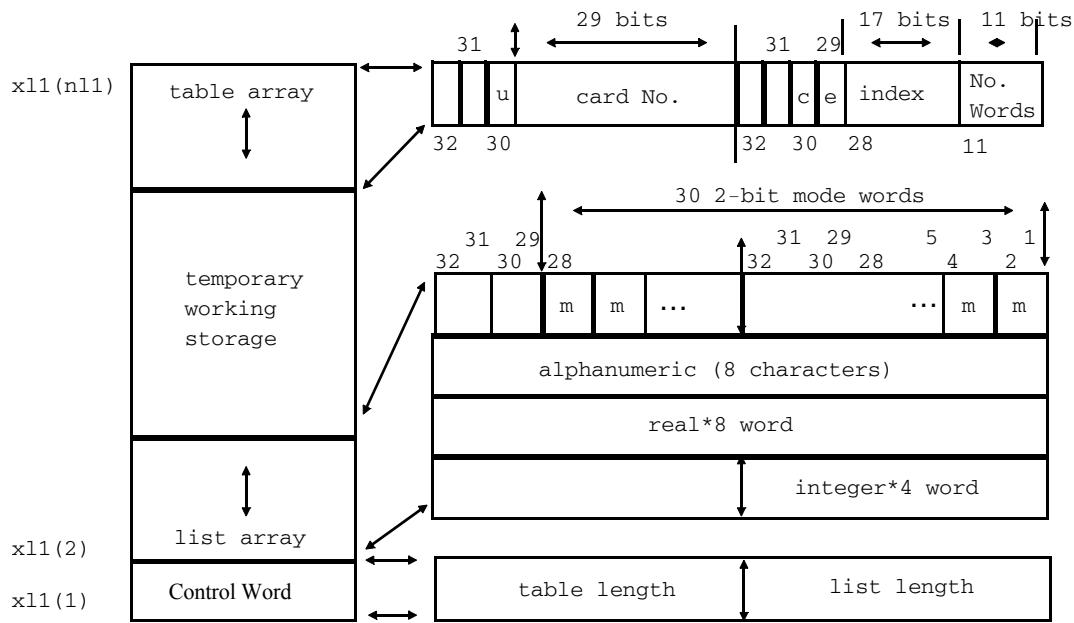
**Table 11.2-1** File Names and Unit Numbers

File Name	Unit	Description
input	11	input file unit number
output	12	output file unit number
inout	20	inp output file unit number
tty	6	message file unit number

### **11.2.6 inp - Input Echo and Package Initialization**

The inp subroutine is used to read in the data cards for the next case. It reads cards from the input file until an end-of-case card (period card), an end-of-set card (slash card) card is read, or an end-of-file mark is encountered. The data is stored in the x11 array. The title card string from the input deck is inserted into the hed character string along with the current time and date character strings. The ncase parameter should be set to 0 for the first call to inp. It is incremented by 1 inside inp, and the sign bit is set if this is the last case of a problem set, i.e., the data is terminated by a period card.

The x11 array consists of nl1 real\*8 words. One control word is stored at location 1, the list array starts at location 2, and the table array starts at last location, x11(nl1). The table extends downward towards the origin of the x11 array. The space between the end of the list array and the end of the table array is used for temporary working storage. **Figure 112-1** shows that storage layout in the x11 array.



**Figure 112-1** Storage Layout for the `xl1` Array (64-bit Words) in the INP Package

Data card information is converted to binary form through calls to the `cvic` subroutine in the CVI package. The binary information from a data card that is not a continuation card is stored starting at the beginning of the temporary working storage, and the mode indicators are stored beginning at the middle of the temporary working storage. The binary information and mode flags for continuation cards are stored following the information for the preceding card. As each individual data card is processed, there must be 40 words between the end of the list array or the last converted binary quantity and the beginning of the mode indicators. This space is necessary to prevent the binary quantities from overrunning the mode indicators and the mode indicators from overrunning the table array. Forty words are necessary because this is the maximum number of quantities that can be entered on an 80-column card. After the data card and any continuation cards have been converted, the binary data and mode indicators are stored as if the data were entered on one card. Subsequent processing can proceed as if only one card was entered. A table word is then constructed consisting of the card number (or the card sequence number if the card number is illegal, `c` flag = 1 in this case), an error flag specifying whether a card format error was detected (`e` flag = 1 in this case), an index into the list array where the binary data starts (`index`), the number of data words on the card, and a usage indicator (`u` flag = 0 initially). If there are data other than the card number on the card, the binary information is moved downward one word eliminating the card number from the list array. The corresponding mode indicators returned from `cvic` are converted to two-bit mode flags (`m` flags = -1, 0, or 1) and stored in a packed form, 30 flags per word, following the binary information. The table array words, one for each card, constitute the table array, and the list array is made up of the converted binary information and associated mode flag words.

In the replacement card checking phase, the new card number is compared against the current card numbers stored in the table array. If a duplication is found, the table word containing the card number is replaced by the new table word. Space occupied by the replaced list array data is recovered by:

1. shifting the list array data down over the top of the replaced data when the number of words on the replaced card and the new card are different, or
2. overwriting the replaced list array data when the number of words are the same.

Table array index pointers are updated when the list array data are moved. When a replacement card contains only a card number, the card acts as a deletion card. In this case, the table array word is deleted, and the list array data space is retrieved by shifting the list array data over the deleted data. A message, "card above is replacement card," is printed below any card that replaces or deletes a data card. A card containing only a card number that is not a duplicated card number has no effect on the table or list arrays, and no message is printed.

As the data cards are being converted to binary, each quantity is checked to see if it is identical to the hexadecimal quantity, 7fffffffffffff, which is the largest positive quantity that could be expressed in floating point notation on the IBM360-370, which where this coding originated. This quantity is not allowed as an input item for programming efficiency as explained in the inp9 and inp10 descriptions, and the quantity is changed to 7fffffff000000fe. This is not considered an error. However, when this occurs, the message,

\$\$\$\$\$\$ Word nnn has unallowed bit pattern, it has been changed.

is printed to the output file where nnn indicates the word on the card that was modified.

The IEEE-754 standard has changed the meaning of 7fffffffffffff. It is now a NaN, so that no input number from a card can possibly be equal to that number, and all the extra checking that is done to check for that number and modify it by one bit in least significant place is probably not necessary on a computer that uses IEEE-754 arithmetic, which is all the workstations and PCs that are currently being manufactured.

A normal return from the inp subroutine is made if a period or slash card is read or an end-of-file mark is encountered after at least one input card was read. Before a normal return, the table array is sorted by card number and is moved adjacent to the list array and the number of words in the list array and the number of words in the table array are packed into the control word at xl1(1). Upon a normal exit, the absolute value of ndata is set equal to the number of words needed in xl1 to hold the control word, the list array, and the table array. The number of words needed is equal to one plus the number of words in the list array plus the number of words in the table array. The sign of ndata is set minus if no data cards (cards other than title, comment, slash, or period cards) are entered for a case. In normal usage this indicates that no input data were entered, and that the succeeding case may have input identical to the preceding case.

On entry to inp, ndata indicates whether the array x11 contains data from a previous case. If ndata = 0, then x11 contains no data from a previous case, and the table array and list array are assumed empty. If ndata > 0, then x11 is assumed to contain data from a previous problem in the same compacted format as that obtained upon a return from inp, i.e., x11 contains a control word containing the number of words in the list and table arrays followed by the list and table arrays. In that case, the table array is moved to the end of x11, the use flags in the table array are cleared, and the input cards for the current case are then processed as described previously.

The parameter isw indicates the return status. If isw = 0, a normal return was made and no errors were detected during the processing of the input cards. If isw = 1, the end-of-file mark was encountered when trying to read the first data card of a case, and inp returned immediately. If isw = 2, a normal return was made, but card format errors were detected and the list of input cards contains one or more errors (messages 3 through 6 in **Table 11.3-1**). The usual practice in this case is to continue checking the input data for additional errors, but execution is terminated after input checking is completed. If isw = 3, the array x11 is not large enough to process the input data (messages 1 or 2 in **Table 11.3-1**), and inp returns immediately. The inp subroutine statement is:

```

subroutine inp (x11, n11, hed, ncase, ndata, isw)
  x11      array for table array and list array storage, real*8; input
  n11      length of storage locations available in x11 array, integer*4;
           input
  hed       heading, contains title card string, date, and time,
           character*108; output
  ncase     case number of previous case, integer*4; input/output
           set ncase to 0 for the initial call
           incremented by 1 inside inp
           output:
           < 0 end-of-case card (period card) was detected
  ndata     flag for previous problem data, integer*4; input/output
           input:
           | 0 ignore previous table and list arrays
           > 0 use previous table and list arrays
           output:
           > 0 total storage used in x11 array
           < 0 no data cards were found for this case
  isw       return status, integer*4; output
           0 = normal return, no errors were detected
           1 = end-of-file mark found
           2 = card format errors
           3 = x11 array is too small

```

### 11.2.7 inp2 - Read One Dataset

The inp2 subroutine is used to move data from the card input buffer and perform certain checking functions. The input buffer array is loc1 and the array into which the data will be stored is loc2. The ics array is the specification array that controls the way inp2 operates on and checks the data in the input buffer. The subroutine inplnk is used to locate each card, and inpmod is used to check the alphanumeric-real-integer format. On return, ics(6) is set to -1 if any card requested contained a conversion error, or if any of the tests specified by the parameters in ics(2), ics(3), ics(4), or ics(7) failed. Otherwise, if ics(6) was

positive on entry, ics(6) is set to +nmove, the number of items moved and if ics(6) was negative on entry, ics(6) is set to -nmove if no errors were found. The ics(7...) array is used to specify the format of the data items on the card. It uses the same format as used by inpmod subroutine: -1 for alphanumeric, 0 for integer, and +1 for real. There is also a provision for condensing this part of the array by using  $\pm n$ ,  $n > 2$ , in the format string to indicate the next  $n$  format items are to be repeated. The inp2 subroutine statement is:

```

subroutine inp2 (loc1, loc2, ics)
loc1    input buffer, real*8, input
loc2    output array, real*8, output
ics     data specification and checking controls, integer*4, input/output
       ics(1) c1, first card number, input
       ics(2) tc2, last card number, input
              = 0 only one card is wanted
              > 0 card numbers are sequential
              < 0 card numbers are increasing, not sequential
       ics(3) minz, minimum number of items to be processed, input
              can be 0 if not sure if the card is present
              ics(6) will be 1 if inp2 found one or more cards
       ics(4) maxz, maximum number of items to be processed, input
              = 0 ignored
       ics(5) nj, skip factor in output array, loc2, input
              usually 0
       ics(6) tj, abs(j) = starting index in loc2, input/output
              input:
                     > =0 data is moved and all checking is done
                     <= 0 no data is moved but all checking is done
              output:
                     > 0 number of items moved and checked
                     < 0 number of items checked
       ics(7) array defining the data item format on the card, integer*4,
              input
              = -1   alphanumeric
              = 0    integer
              = +1   real
              =  $\pm n$  |n|  $\geq 2$  format repeats and n is the number of items
                     repeated
                     +n the cycle is reset at the beginning of each new
                     card
                     -n the cycle can overlap cards
                     Within a cycle, all elements must be -1, 0, or +1
                     (no allowance for nested repeat cycles).

```

### 11.2.8 inp4 - Read Multiple Datasets

The inp4 subroutine permits several sets of related data to be moved and checked by a single call. A do loop which includes a call to inp2 is executed ntimes inside inp4. The first time through the loop, the c1, c2, minz, maxz, nj, j, loc1, and loc2 are as described in inp2. The format array is loc5. For each subsequent time through the loop, an internal variable that is set to c1 is increased by c3. The same is true for abs(c2). Also, if j is positive, it is increased by j1. The value of j when inp4 returns is as described for inp2. The maximum size of array loc5 is 40. There are no calls to inp4 in RELAP5. The inp4 subroutine statement is:

```

subroutine inp4 (c1, c2, minz, maxz, nj, j, c3, ntimes, j1, loc1, loc2,
               loc5)
c1      first card number, integer*4, input
c2      ± last card number, integer*4, input
minz    minimum number of items to process, integer*4, input
maxz    maximum number of items, ignored if 0, integer*4, input
nj      number of spaces to skip between items in loc2, integer*4, input
j       starting point in loc2, integer*4, input/output
c3      increment for c1, integer*4, input
ntimes   number of times do loop is executed, > 0, integer*4, input
j1      increment for j, integer*4, input
loc1    input buffer, real*8, input
loc2    output array, real*8, output
loc5    format array, integer*4, input
        = -1    alphanumeric field
        = 0     integer field
        = +1    real field

```

### 11.2.9 inp5 - Read Vector Dataset

The inp5 subroutine is similar to inp4 in that it will process several sets of cards on a single entry. It is more powerful than inp4 in that it performs a sequential or overlay expansion of the input and stores the expanded data in either dense or scattered mode.

The basic unit input data is a vector of components. If n1 is positive, the data is of sequential form where means that is to be repeated times and represents expanded vectors with, and. If n1 is negative, the data is of the overlay form where the data is overlaid on an initial set of vectors beginning at the vector and extends through the vector, with, and. For either type, there results a sequence of expanded input of the form. For overlay data, a positive nmin requires that the lower limit be included while a negative nmin only specifies a lower bound. For sequential type, a negative nmin can be used for negative indexing. For both types, a positive nmax is used to require that the upper limit be included while a negative nmax only specifies an upper bound.

The vectors form a two dimensional array with elements. This array can be stored into xl2 in two different modes where one mode is the transpose of the other mode. If nstore is positive, it is stored in  $xl2(j+(k-1)+nstore*(i-1))$ . If nstore is negative, it is stored in  $xl2(j+(i-1)+|nstore|*(k-1))$  and proper size of nstore depends on nmin and nmax. This is equivalent to having a two-dimensional array defined as  $real*8 ss(nstore,n)$  and equivalence  $(xl2(j),ss(1,1))$ . If nstore is positive, the data is stored in  $ss(k,i)$ , and if nstore is negative, the data is stored in  $ss(i,k)$ .

The exact location in which a particular element of an input set is stored depends upon both the expansion type and the storage mode. Let  $j$  be the index of a particular element in an input set ( $1 \leq j \leq in1$ ), and let  $i$  be one of the index values with which this set is associated. Then the storage location for element  $j$  associated with index value  $i$  is one of the following:

sequential expansion - dense storage  
 $loc2(j + in4 * (i - 1 - nmin) + (j - 1))$

overlay expansion - dense storage

$\text{loc2}(j + \text{in4} * (i - 1) + (j - 1))$

sequential expansion - scattered storage

$\text{loc2}(j + (i - 1 - \text{in2}) + |\text{in4}| * (j - 1))$

overlay expansion - scattered storage

$\text{loc2}(j + (i - 1) + |\text{in4}| * (j - 1))$

The parameters c1, c2, c3, in5, in7, and loc5 are handled as in inp4. The in7 parameter contains not the expanded amount of data, but instead, the amount of data on the cards. In particular, c1, c2, c3, and ntimes specify the cards in a set and the number of sets to process, while in6 is added to in7 after each set to change the starting point in loc2.

The in1 input parameter is the input set length and is positive for sequential expansion and negative for overlay expansion. The in4 input parameter is the length of the store vector and is positive for dense storage and negative for scattered storage. The use of in2 and in3 depends upon the expansion type. For sequential expansion, in2 is the base index (one less than the initial index value) and in3 is an upper bound for the terminal indices. The in2 parameter is normally zero, but may have any positive or negative value required to define the initial index. The in3 parameter is positive if the upper limit is to be taken on, negative if it is merely an upper bound, and zero if it is to be ignored. For overlay expansion, in2 is the lower bound for the initial indices and in3 is the upper bound for the terminal indices. In either case, a positive value specifies that the limit must be taken on by some input set, and a negative value merely provides a bound. The in3 parameter may be zero to signify no upper bound.

The array loc6 has a length of in8 and is used for temporary storage inside inp5. It must be large enough to hold the un-expanded input data for one card set,  $c1 < c < c2$ . Additional error checks are made because of the form of the input, and error messages 15 through 20 can be printed. Subroutine inp6 is called for error processing. If the parameters, in1, in4, or in5 are less than or equal to zero, an abnormal termination occurs. The value of in7 must be checked on the return from inp5 since it is set to -1 if any errors have been detected. As in inp4, the maximum size of array loc5 is 40. The inp5 subroutine statement is:

```

subroutine inp5 (c1, c2, c3, in1, in2, in3, in4, in5, in6, in7, loc1, loc2,
                 loc5, loc6, in8)
c1      first card number, integer*4, input
c2      ± last card number, integer*4, input
c3      card number increment, integer*4, input
in1     ±n1, input set length, integer*4, input
        > 0 sequential expansion
        < 0 overlay expansion
in2     ±nmin, base index, integer*4, input
in3     ±nmax, upper bound for terminal indices, integer*4, input
in4     ±nstore, length of storage vector, integer*4, input
        > 0 for dense storage
        < 0 for scattered storage
in5     ntimes, number of times do loop is executed, integer*4, input

```

```

in6      newj, increment for j, integer*4, input
in7      j, starting index in loc2, integer*4, input
loc1     input card buffer array, real*8, input
loc2     output array, real*8, input
loc5     format array, integer*4, input
         = -1  alphanumeric field
         = 0   integer field
         = +1  real field
loc6     temporary storage array, real*8, input/output
in8      length of loc6 array, integer*4, input

```

### **11.2.10 inp6 - Error Processing for inp2**

The inp6 subroutine may be used when item n1 of a set obtained from cards c1 through c2 by inp2 is found to be in error. Upon return, card will be the card number and item the number of the field on that card which contained the data item in error. The inp6 subroutine statement is:

```

subroutine inp6 (c1, c2, n1, card, item, loc1)
c1      initial card number, integer*8, input
c2      final card number, integer*8, input
n1      item number desired, integer*8, input
card    number of the card that contains the error, integer*8, output
item    field number that is in error the card, integer*8, output
loc1    input card buffer array, real*8, input

```

### **11.2.11 inp7 - Error Message Printing**

The inp7 subroutine prints on the output file an error message that says items on card are in error.

The inp7 subroutine statement is:

```

subroutine inp7 (card, item)
card    the card number, integer*8, input
item    the item number on the card, integer*8, input

```

### **11.2.12 inp8 - Count Unreferenced Cards**

The integer\*4 inp8 function returns the number of unreferenced cards in the input card buffer table array for the current case. A card is unreferenced if it has not been called by inplnk. If list = 1, the card numbers of unreferenced cards are listed, and if list = 0, no output is generated. Asterisks will precede an unreferenced card number if the e-flag bit is set for this card in the table array. The e-flag bit is set when the card number is illegible and means that number printed is not a card number, but instead, it is an index of its location in the card input deck. The inp8 function statement is:

```

integer function inp8 (list, a)
list    output flag, integer*8, input
        = 0    no output
        = 1    listing
a      input card buffer (table-list) array, real*8, input

```

### 11.2.13 inp9 - Delete All Processed Cards

The integer\*4 inp9 function returns the new length of the table-list after it has deleted from the table array those cards which have been referenced at least once during the current case by inplnk, inp2, inp4, or inp5 and the associated data and mode words. A card is marked as referenced in the table when the u-flag bit is set. The one parameter to this function is the name of the array containing the table-list arrays. When all the cards and associated data have been deleted, the length is 1, and only the control word is left.

When a card is deleted, a hole in the table array and list array is created, and the remaining table and list array entries must be moved to recover the storage made available. In order to move the remaining table and list array entries only once and not use storage outside of the array xl1, the holes are marked with the bit pattern 7fffffffffffff as they are formed. After all cards are deleted, holes are located by testing for this special bit pattern. Thus, the bit pattern used for marking holes must not be allowed as a data item. The bit pattern selected is the largest positive floating point number on the IBM-360-370 and is unlikely to occur. As described for inp, if that bit pattern is entered as input data, the last significant bit is set to zero.

Any holes created by card and data deletion are squeezed out, the table indices are adjusted accordingly, and the control word is updated. The inp9 function statement is:

```
integer function inp9 (a)
a      input card buffer (table-list) array, real*8, input/output
```

### 11.2.14 inp10 - Delete Specified Range of Processed Cards

The integer\*4 inp10 function performs the same type of deletion as inp9, but for a specified range of cards rather than for all cards that have been processed. It returns the new length of the table-list array. Parameter 11 is the name of the array containing table and list; c1 and c2 specify that all cards,  $c$ ,  $c_1 \leq c \leq c_2$ , are to be deleted from the table-list. Any holes created by card deletion are squeezed out, the table indices are adjusted accordingly, and the control word is updated. The inp10 function statement is:

```
integer function inp10 (a, nc1, nc2)
a      input card buffer (table-list) array, real*8, input/output
nc1    first card number to delete, integer*8, input
nc2    last card number to delete, integer*8, input
```

### 11.2.15 inplnk - Card Number Search

The inplnk subroutine searches the table array part of array a for the card numbered card. It sets next equal to the next largest card number in the table array. If no such card exists, then next = -1. On return from inplnk, if many = 0, the card numbered card is not in the table. If many < 0, a format error was detected by inp on the card. If many > 0, many is set to the number data fields (excluding the card number) on the card, and the data are located in the list array beginning at a(where). If card is positive, the u-flag bit is set in the table entry for the card if it is found. The u-flag bit is the used bit and signifies that the card has been used and will be removed when inp9 or inp10 is called. If card is negative, the u-flag bit is cleared if the card is found. The inplnk subroutine issues no error messages to the output file. The inplnk subroutine statement is:

```

subroutine inplnk (card, next, where, many, a)
card   ± the card number, integer*4, input
       > 0 u-flag (used) bit is set if in table array
       < 0 u-flag (used) bit is cleared if in table array
next   the next largest card number, integer*4, output
       = -1 no such card exists
where  location in the a array where data from card is stored, integer*4,
       output
many   number of data fields on the card, integer*4, output
       < 0 format error on card
       = 0 no such card number in table array
       > 0 number of data fields on card excluding card number
a      array where the data is stored, real*8, input/output

```

### 11.2.16 inpmmod - Data Type Identification

The inpmmod subroutine checks mnum items of data stored sequentially beginning at loc1(j) for appropriate format (where n and mnum must be exactly equal to the values returned by an previous inplnk call). The format specification begins at loc3(7) with entries of -1 for alphanumeric, 0 for integer, and +1 for real. Cyclic repetition of two or more entries can be condensed by prefixing the repeated format by ±n, the number of items repeated, +n if the cycle is to be reset for each entry to inpmmod or -n if it is to pick up at the stopping point of the previous entry. To allow starting within the format specification, nc is the number of items on cards previously checked with the current format. On return from inpmmod, n = 0 if no errors were found; 0 < n < 10000 if item n should have been integer but was not; n < 0 if item -n should have been real but was not; and n > 10000 if item (n - 10000) should have been alphanumeric but was not. A decimal zero is considered either integer or floating point as required to satisfy these mode tests. The inpmmod subroutine issues error messages to the output file. The inpmmod subroutine calls inpupk to unpack the mode flags out of the list array. Note that the format specification contains an entry for each field in the case of real and integer but contains an entry for each word created from an alphanumeric field. The inpmmod subroutine statement is:

```

subroutine inpmmod (loc1, loc3, j, mnum, n, nc)
loc1   array for data items to be checked, real*8,
loc3   control words and format specification, integer*4, input
j     index in loc1 where data starts, integer*4, input
mnum  number of items to check, integer*4, input
n     cyclic repetition factor and output error flag, integer*4, input/
       output
       < 0, item |n| should be real, but is not
       = 0, mode is correct
       < 10000, item n should be integer. but is not
       > 10000, item n should be alphanumeric, but is not
nc    number of items on previous card checked with current format,
       integer*4, input

```

### 11.2.17 inppck - Pack Mode Words in List Array

The inppck subroutine is called from the inp subroutine to construct the packed mode words for insertion into the list array. The mode values came from the cvic subroutine call that read in the card deck. The inppck subroutine statement is:

```
subroutine inppck (l1, n, 12)
11      array (2) of two packed words for insertion into list array,
           integer*4, output
n       number of mode words to pack into l1, integer*4, input
12      array (2,*) of mode words in right 32 bits of a 64-bit word,
           integer*4, input
```

### 11.2.18 inpukp - Unpack Mode Words from List Array

The inpukp subroutine is called by inpmod to unpack the mode words from the list array so that they can more easily be used to check the format of the individual data items. The inpukp subroutine statement is:

```
subroutine inpukp (l1, n, 12)
11      array (2) of two packed words containing the modes from list array,
           integer*4, input
n       size of 12 array, number of mode words to unpack into 12,
           integer*4, input
12      array (n) of unpacked mode words in right 32 bits of a 64-bit word,
           integer*4, output
```

## 11.3 INP Summary

### 11.3.1 Error Message Summary

All error messages are preceded by eight asterisks to facilitate recognition of errors in the midst of the normal program output. The symbols used in the following are replaced by actual values when they occur in the output listing.

**Table 11.3-1** INP Package Error Messages

Number	Message
1	Insufficient storage allocation for previous data, processing terminated
2	Insufficient storage for data, processing terminated
3	^ points to card error at col. (a ^ sign is placed under the column in error)
4	End of tile encountered before end (.) card
5	Continuation card indicated, but no previous data card. Treated as a new card
6	Unrecognizable card number
7	Word "n5" on card "ic" should be alphanumeric format

**Table 11.3-1** INP Package Error Messages

Number	Message
8	Card “c+a+1” missing in sequence
9	Too few numbers on cards “ic1” through “ic2”
10	Too many numbers on cards “ic1” through “ic2”
11	Word “n5” on card “ic” should be in integer format
12	Word “n5” on card “ic” should be in floating point format
13	Cards “ic1” through “ic2” missing
14	Illegal format on card “ic”
15	“m” numbers on cards “ic1” through “ic2” are not a multiple on “n1”
16	Item “m” on card “ic” is less than minimum allowed of “nmin”
17	Item “m” on card “ic” exceeds maximum allowed for “nmax”
18	Error in limits of the set beginning at item “m” on card “ic”
19	Lower limit of “nmin” not included on cards “ic1” through “ic2”
20	Upper limit on “nmax” not included on cards “ic1” through “ic2”
21	The following cards were not used
22	Item “item” on card “card” in error

### 11.3.2 Word Structure Used in xl1

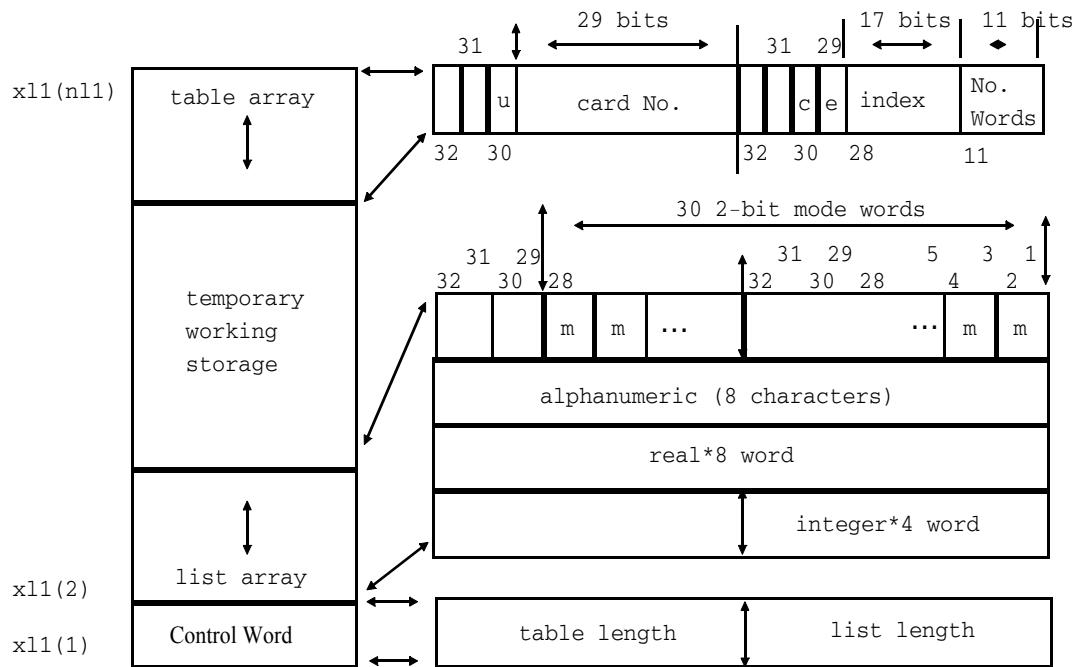
#### 11.3.2.1 Control Word Structure

The lx1 array contains a control word, converted data from the cards, mode indicators and the table entries. The control word is the first word of this array. This control word is divided into two 32-bit integer words. Word 1 contains the length of the table, and word 2 contains the length of the list containing the binary data and the mode indicators. **Figure 113-1** shows this structure and is repeated here as a reference for the following detailed description.

#### 11.3.2.2 Table Word Structure

The table consists of 64-bit words. The first word is the control word and is further broken into two 32-bit words that are further subdivided into a set of bit fields. In these 32-bit words, the bits are numbered from the right starting at 1 and going to 32 for the leftmost bit.

For the leftmost 32-bit word, word 1, bits 1-29 contain the card number or the sequence number. Bit 30 is the usage flag (u). This bit is 0 if the card has not been processed and 1 if it has been processed. The sequence number is generated by the inp subroutine if the card number is unintelligible.



**Figure 113-1** Storage Layout for the xl1 Array (64-bit Words) in the INP Package

Bit 30 is the card-sequence flag. It is 0 if bits 1-29 contain a card number and 1 if bits 1-29 contain a sequence number. Bit 31 is the error flag. It is 0 if no format errors occurred when parsing the card, and 1 if there were format errors.

For the second 32-bit word, word 2, bits 0-15 contain the index in the xl1 array for the first data word of the card associated with this table array. Bits 16-31 contain the number of words on the card excluding the card number. A summary of this information is shown in **Table 11.3-2**.

Card numbers are limited to  $536,870,911$  ( $2^{29} - 1$ ). Card numbers greater than this result in error message 6 being printed. The list length and word count are limited to 131,072 ( $2^{17}$ ). The number of words on a card is limited to 2048 ( $2^{11}$ ).

**Table 11.3-2** Table Word Structure

Word	Bit range	Contents
1	1-29	Card number (sequence number if card number is illegal)
	30	Used flag (u) = 0 not used = 1 used (processed by inp2, inp4, etc.)
	31-32	not used

**Table 11.3-2** Table Word Structure

Word	Bit range	Contents
2	1-11	Number of words on the card
	12-28	Index into ill array for card data
	29	Format error flag (e) from cvic call = 0 no errors = 1 one or more errors
	30	Card number in error flag (c) = 0 no error, card number in word 1, 1-29 = 1 illegal card number, sequence number in word 1, 1-29
	31-32	Not used

### **11.3.2.3 Mode Indicator Word Structure**

For each card (continuation cards are considered as part of the first card here), the mode indicator words are stored immediately following the last data word. Mode indicators consist of two bits, and these are packed, right-justified, 30 indicators per word. Because packing only 30 mode indicators per word leaves the left-most four bits zero, there is no possibility of a mode indicator word to be treated as the special bit pattern that is used for a deleted card. The mode indicators are as follows:

- 0 for an integer or floating point zero,
- 1 for a non-zero integer,
- 2 for a non-zero floating point quantity, and
- 3 for an alphanumeric quantity.

## **11.4 References**

- 11.4-1. Cadwell, W. R. Reference Manual - Bettis Programming Environment. WAPD-TM-1181 (Draft). Bettis Atomic Power Laboratory: Pittsburgh, Pennsylvania. March 1974.
- 11.4-2. NRTS Environmental Subroutine Manual. National Reactor Testing Station. Idaho Falls, Idaho. (unpublished) December 1972.
- 11.4-3. Pfeifer, C. CDC-6600 Fortran Programming - Bettis Environmental Report. WAPD-TM-668. Bettis Atomic Power Laboratory: Pittsburgh, Pennsylvania. January 1967.

## References

## 12 THERMODYNAMIC PROPERTY PACKAGE

The thermodynamic properties of water are obtained from subroutines that interpolate values in binary tables. The tables are constructed from the evaluation of functions in the 1967 ASME Steam Tables<sup>12.4-1</sup> using the ASTEM package<sup>12.4-2</sup>. The stgh2o program is used to generate the binary tables and write them out to a file. Initialization of the tables in memory in RELAP5 is done once in the icmpn1 subroutine by calling stread. The table interpolation subroutines, sth2x0, sth2x2, sth2x3, sth2x4, sth2x5, and sth2x6, return all the single phase and/or saturated thermodynamic properties for a given set of input parameters. Allowable combinations of input parameters are temperature and pressure, temperature and specific volume, pressure and enthalpy, or pressure and internal energy. Saturated properties can also be obtained for a given set of input parameters. Allowable combinations of input parameters for saturation properties are temperature, temperature and quality, or pressure and quality. The psatpd subroutine returns the partial derivative of pressure with respect to temperature along the saturation line and either the saturation pressure or saturation temperature when it is given either a temperature or pressure.

Other subroutines were developed to generate the heavy water binary tables and to interpolate in these tables. Since these subroutines are almost identical to the ones for light water, they are not described here. Over the course of the development of RELAP5, other thermodynamic property subroutines have been developed. Some of these subroutines were developed for liquid metals, like lithium, potassium, NaK, etc., and for cryogenic fluids, like hydrogen, helium, etc. The interpolation subroutines that were used to for these fluids were more general and an attempt was made to put the light water and heavy water tables into this set. The attempt was not judged to be completely successful when a comparison was made with the original subroutines, so the original light and heavy water subroutines were not replaced with this more general set. However, calls to the more general set can still be seen in the coding as one of the cases in the if-then-elseif-endif blocks that decide which set of tables to use.

All the subroutines require SI units for input and return SI units for output. The set of thermodynamic properties used in RELAP5 are shown in **Table 12.0-1**. When the symbols are used in the text or equations, saturated quantities are denoted with a superscript, s, saturated liquid quantities are denoted with a subscript, f, and saturated vapor quantities are denoted with a subscript, g.

**Table 12.0-1** Thermodynamic Quantities

Quantity	Symbol	SI Units
Temperature	T	K
Pressure	P	Pa or N/m <sup>2</sup>
Specific Volume	v = 1/ρ	m <sup>3</sup> /kg
Internal Energy	u	J/kg

**Table 12.0-1 Thermodynamic Quantities**

<b>Quantity</b>	<b>Symbol</b>	<b>SI Units</b>
Enthalpy	$h = u + Pv$	J/kg
Entropy	$s$	J/kg-K
Coefficient of Isobaric Thermal Expansion	$\beta = \frac{1}{v} \left( \frac{\partial v}{\partial T} \right)_P$	1/K
Coefficient of Isothermal Compressibility	$\kappa = \frac{-1}{v} \left( \frac{\partial v}{\partial P} \right)_T$	1/Pa
Quality	$x$	(kg vapor)/(kg total)
Specific Heat at Constant Pressure	$C_P$	J/kg-K

## 12.1 stgh2o - Generate Water Properties

The stgh2o program generates the binary tables that are used in RELAP5 to compute the thermodynamic properties of water. There are five tables and they are written to one binary file as three records. Two of the tables are essentially an echo of the temperature and pressure input values, and the other three tables are generated using the ASTEM package. The first table contains the temperature input values. The second table contains the pressure input values. The third table contains the saturation properties at each of the input pressure values. The fourth table contains the saturation properties at each of the input temperature values. The fifth table contains the single phase properties for each of the input temperature-pressure value combinations. The stgh2o program as well as the ASTEM package are in the envrl directory in the RELAP5 distribution.

### 12.1.1 Input Data

The input data for stgh2o is free-format. The input consists of five groups of records plus an optional sixth record. In the following description, an active data record means any non-blank record. Each record is limited to 80 characters. The active data records can be separated by blank records.

The first data record is ignored if first character is an asterisk,'\*'. In the RELAP5 distribution, the first data record is the \*deck record that names the file and is used by the usplit utility for a file name when the envrl.s file is split into individual files.

The next active record is the title record and contains the title that is given to the thermodynamic properties file.

The next active data record is the temperature count record and contains the number of temperatures, nt, followed by at least one blank, comma, or slash, followed by any character strings the user wishes.

The next active data record(s) are the temperature records and contain nt temperature values in ascending order, delimited by at least one blank, comma, or end-of-record. The temperatures must be in SI units, i.e., K. For proper operation of the interpolation subroutines, three temperatures must be included in this set. For water, these are the triple point temperature,  $T_{tp} = 273.16$  K, the critical point temperature,  $T_{cp} = 647.3$  K, and the ASME maximum temperature limit, 1073.15 K. The maximum temperature that can be entered is  $T_{max} = 5000$  K.

The next active data record is the pressure count record and contains the number of pressures, np, followed by at least one blank, comma, or slash, followed by any character strings the user wishes.

The next active data record(s) are the pressure records and contain np pressure values in ascending order, delimited by at least one blank, comma, or end-of-record. The pressures must be in SI units, i.e., Pa. For proper operation of the interpolation subroutines, two pressures must be included in the set. These are the triple point pressure,  $P_{tp} = 611.2445$ , and the critical point pressure,  $P_{cp} = 2.212E+7$ . At least one pressure must be larger than the critical point pressure.

The last active record(s) are the trailing records and contain any characters the user wishes. The trailing records are optional.

The limits on the input temperatures and pressures are shown in the following table along with the symbol that will be used in the following discussion.

**Table 12.1-1 Data Limits for Water**

Description	Temperature (K)		Pressure (Pa)		Specific Volume (m <sup>3</sup> /kg)	
Minimum	$T_{min}$	2.73160d+02	$P_{min}$	6.11240d+02		n/a
Maximum	$T_{max}$	5.00000d+03	$P_{max}$	1.00000d+08		n/a
Triple Point	$T_{tp}$	2.73160d+02	$P_{tp}$	6.11240d+02	$v_{tp}$	1.00020d-03
Critical Point	$T_{cp}$	6.47300d+02	$P_{cp}$	2.21200d+07	$v_{cp}$	3.17000d-03

## 12.1.2 Sample Input File

The following listing shows the input file that is currently used for generating the light water properties for RELAP5. This file contains one \*deck record, one title record, one temperature count record, 66 temperatures on seven records, one pressure count record, 38 pressures on seven records, and no trailing records.

```
*deck stgh2oi
```

```

tpfh2o version 1.1.1, tables of thermodynamic properties of light water
66 temperatures as follows:
273.16 277.5 280. 285. 290. 295. 300. 305. 310. 315.
320. 330. 340. 350. 360. 380. 400. 420. 440. 460.
480. 500. 510. 520. 530. 540. 550. 560. 570. 580.
590. 595. 600. 605. 610. 615. 620. 625. 630. 635.
640. 642. 644. 645. 646. 647. 647.3 648. 649. 650.
652. 654. 660. 675. 700. 800. 1073.15 1450.
1800. 2200. 2600. 3000. 3500. 4000. 4500. 5000.

38 pressures as follows:
611.2445 2.e3   5.e3   10.e3   20.e3 50.e3   100.e3
200.e3   400.e3 800.e3   1000.e3 1500.e3 2000.e3
2500.e3   3000.e3 3500.e3   4000.e3 5000.e3 6000.e3
7000.e3   8000.e3 10000.e3 12500.e3 15000.e3 17500.e3
20000.e3 21000.e3 22000.e3 22120.e3 23000.e3 24000.e3
25000.e3
30.e6    40.e6   50.e6   60.e6   80.e6   100.e6

```

### 12.1.3 Binary Table Format

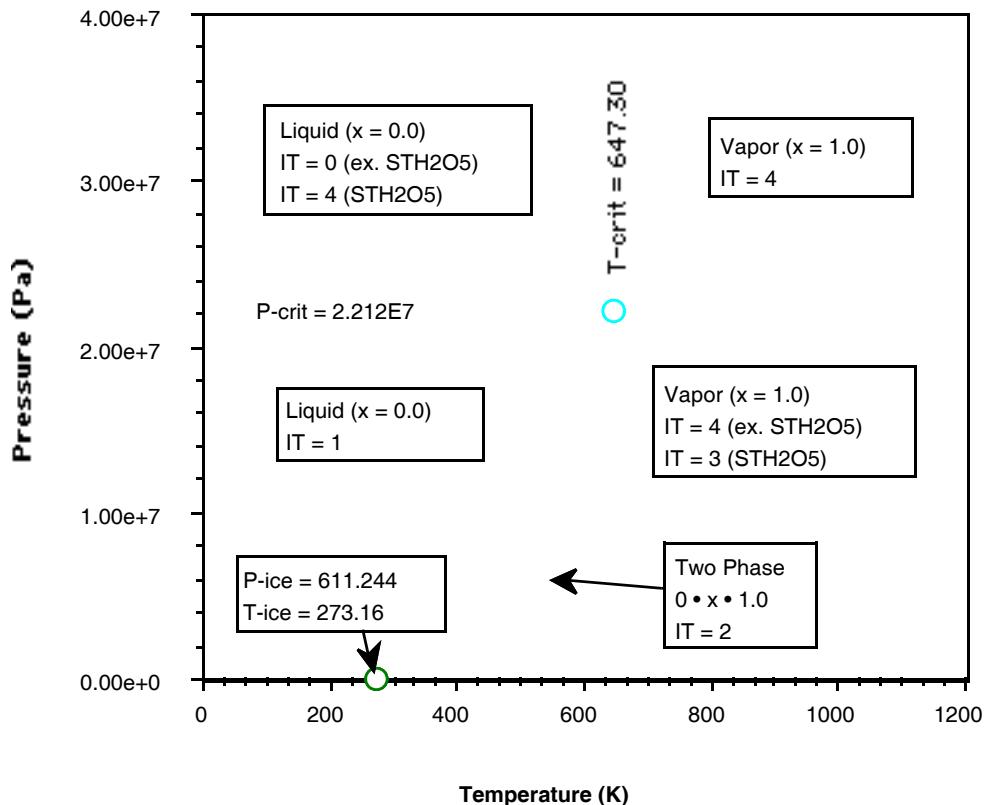
There are five binary tables and they are packed into a single one-dimensional array. In the following description, nst refers to the number of saturation temperatures, i.e., temperatures that are less than or equal to the critical point temperature,  $T_{cp}$ , and nsp is the number of saturation pressures, i.e., pressures that are less than or equal to the critical point pressure,  $P_{cp}$ . In the example above, nst = 47 and nsp = 29.

The first record contains data limits, table statistics, and pointers into the third record. The second record contains the total number of words in the third record. The quantities in the first record are shown in **Table 12.1-2**, and the one quantity in the second record is shown in **Table 12.1-3**. **Figure 121-1** shows the location of these points in the pressure-temperature plane.

**Table 12.1-2** Record 1 of the Steam Table Binary File, tpfh2o

Word No.	Variable Name in Coding	Symbol	Description	Value for Water
1	ttrip	$T_{tp}$	Triple Point Temperature	2.73160D+02
2	ptrip	$P_{tp}$	Triple Point Pressure	6.11240D+02
3	vtrip	$v_{tp}$	Triple Point Specific Volume	1.00020D-03
4	tcrit	$T_{cp}$	Critical Point Temperature	6.47300D+02
5	pcrit	$P_{cp}$	Critical Point Pressure	2.21200D+07
6	vcrit	$v_{cp}$	Critical Point Specific Volume	3.17000D-03

### Definitions for STH2X... Subroutines



**Figure 121-1** Pressure-Temperature Diagram Showing Regions and Critical and Triple Points

**Table 12.1-2** Record 1 of the Steam Table Binary File, tpfh2o

Word No.	Variable Name in Coding	Symbol	Description	Value for Water
7	tmin	T <sub>min</sub>	Minimum Temperature	2.73160D+02
8	pmin	P <sub>min</sub>	Minimum Pressure	6.11240D+02
9	tmax	T <sub>max</sub>	Maximum Temperature	5.00000D+03
10	pmax	P <sub>max</sub>	Maximum Pressure	1.00000D+08
11	nt		No. Temperature Points	
12	np		No. Pressure Points	
13	nst		No. Saturation Temperatures	
14	nsp		No. Saturation Pressures	

**Table 12.1-2** Record 1 of the Steam Table Binary File, tpfh2o

Word No.	Variable Name in Coding	Symbol	Description	Value for Water
15	it3bp		Base Pointer to Table 3	
16	it4bp		Base Pointer to Table 4	
17	it5bp		Base Pointer to Table 5	
18	it3p0		Pointer to start of Table 3	

**Table 12.1-3** Record 2 of the Steam Table Binary File, tpfh2o

Word	Variable Name in Coding	Description
1	ntot	Total No. Points in Record 3

Record 3 contains the values that are used in the interpolation subroutines. Logically, it is composed of five binary tables.

1. Temperature table,
2. Pressure table,
3. Saturation pressure table,
4. Saturation temperature table, and
5. Single phase table.

#### **12.1.3.1 Table 1, Temperatures**

The temperatures in the temperature table are the values from the input and are in increasing order. There are nt values, and they are the first nt words in record 3. For the example input deck shown above, there are 66 words in this table.

#### **12.1.3.2 Table 2, Pressures**

The pressures in the pressure table are the values from the input and are in increasing order. There are np values, and they are stored behind the temperature table. For the example input deck shown above, there are 38 words in this table.

#### **12.1.3.3 Table 3, Saturation Properties vs. Temperature**

The saturation properties as a function of temperature are stored immediately behind the pressure table. There are  $1 + 2 * \text{nprops} * \text{nst}$  values, where nprops is the number of saturation properties, and nst is the number of saturation temperatures. The parameter nprops is set to 6 in a parameter statement.

The liquid and vapor saturation properties are shown in **Table 12.1-4**. These values are added to record 3 in the order shown in the table: saturation pressure, saturation liquid specific volume, saturation vapor specific volume, saturation liquid internal energy, etc. There is one set for each temperature that is less than or equal to the critical point temperature. For the example input deck shown above, there are 565 words in this table.

**Table 12.1-4** Saturation Quantities as a Function of Temperature in Table 3

Quantity	Symbol		SI Units
Pressure	$P^s(T^s)$		Pa or N/m <sup>2</sup>
Specific Volume	$v_f^s(T^s)$	$v_g^s(T^s)$	m <sup>3</sup> /kg
Internal Energy	$u_f^s(T^s)$	$u_g^s(T^s)$	J/kg
Coefficient of Isobaric Thermal Expansion	$\beta_f^s(T^s)$	$\beta_g^s(T^s)$	1/K
Coefficient of Isothermal Compressibility	$\kappa_f^s(T^s)$	$\kappa_g^s(T^s)$	1/Pa
Specific Heat at Constant Pressure	$C_{pf}^s(T^s)$	$C_{pg}^s(T^s)$	J/kg-K
Entropy	$s_f^s(T^s)$	$s_g^s(T^s)$	J/kg-K

#### **12.1.3.4 Table 4, Saturation Properties vs. Pressure**

The saturation properties as a function of pressure are added to record 3 immediately behind the saturation properties as a function of temperature. There are  $1 + 2 * \text{nprops} * \text{nsp}$  values, where nsp is the number of saturation pressures.

The liquid and vapor saturation properties are shown in **Table 12.1.3.5**. The values are added to record 3 in the order shown in the table: saturation temperature, saturation liquid specific volume, saturation vapor specific volume, saturation liquid internal energy, etc. For the example input deck shown above, there are 349 words in this table.

**Table 12.1-5** Saturation Quantities as a Function of Pressure in Table 4

Quantity	Symbol		SI Units
Pressure	$P^s(P^s)$		Pa or N/m <sup>2</sup>

**Table 12.1-5** Saturation Quantities as a Function of Pressure in Table 4

Quantity	Symbol	SI Units
Specific Volume	$v_f^s(P^s)$	$v_g^s(P^s)$
Internal Energy	$u_f^s(P^s)$	$u_g^s(P^s)$
Coefficient of Isobaric Thermal Expansion	$\beta_f^s(P^s)$	$\beta_g^s(P^s)$
Coefficient of Isothermal Compressibility	$\kappa_f^s(P^s)$	$\kappa_g^s(P^s)$
Specific Heat at Constant Pressure	$C_{pf}^s(P^s)$	$C_{pg}^s(P^s)$
Entropy	$s_f^s(P^s)$	$s_g^s(P^s)$

**12.1.3.5 Table 5, Single Phase Properties**

The single phase properties as a function of temperature and pressure are added to record 3 immediately behind the saturation properties as a function of pressure. There are nprops\*nt\*np values.

The single phase properties are shown in **Table 12.1-6**. The values are added to record 3 in order shown in the table: liquid or vapor specific volume, liquid or vapor internal energy, liquid or vapor coefficient of isobaric thermal expansion, liquid or vapor coefficient of isothermal compressibility, liquid or vapor specific heat at constant pressure, and liquid or vapor entropy. In this table, the pressure index varies the most rapidly. The liquid properties are used when the pressure-temperature point is in the liquid region and vapor properties are written when the pressure-temperature point is in the vapor region. For the example input deck shown above, there are ntot = 16127 words in this table.

The binary output file is written as three records to FORTRAN unit 1. The file is named tpfh2o.

**Table 12.1-6** Single Phase Quantities as a Function of Temperature and Pressure in Table 5

Quantity	Symbol	SI Units
Specific Volume	$v(P, T)$ or $v_g(P, T)$	$m^3/kg$
Internal Energy	$u_f(P, T)$ or $u_g(P, T)$	$J/kg$
Coefficient of Isobaric Thermal Expansion	$\beta_f(P, T)$ or $\beta_g(P, T)$	$1/K$

**Table 12.1-6** Single Phase Quantities as a Function of Temperature and Pressure in Table 5

Quantity	Symbol	SI Units
Coefficient of Isothermal Compressibility	$\kappa_f(P, T)$ or $\kappa_g(P, T)$	1/Pa
Specific Heat at Constant Pressure	$C_{pf}(P, T)$ or $C_{pg}(P, T)$	J/kg-K
Entropy	$s_f(P, T)$ or $s_g(P, T)$	J/kg-K

## 12.2 Interpolation Subroutines

The table interpolation subroutines have a set of arguments that are quite similar. The following defines these arguments:

a	steam tables
err	true if the subroutine call was successful, false otherwise
it	region in the steam tables (see Figure D.3-1) 1 = liquid below the critical point pressure 2 = two phase 3 = vapor below the critical point pressure 4 = above the critical point pressure
itype	input flag for psatpd subroutine 1 = evaluate the derivative at the input temperature and return the saturation pressure 2 = evaluate the derivative at the input pressure and return the return saturation temperature
p	pressure (Pa)
pdt	partial derivative of pressure with respect to temperature along the saturation line (Pa/K)
s	array of 26 thermodynamic quantities returned by the subroutine (see <b>Table 12.2-1</b> )
t	temperature (K)

**Table 12.2-1** Thermodynamic Properties Returned in the s Array

No.	Property	No.	Property	No.	Property
1	T	10	$P^s$ or $T^s$	19	$\kappa_f^s$
2	P	11	$v_f^s$	20	$\kappa_g^s$
3	v	12	$v_g^s$	21	$C_{pf}^s$
4	u	13	$u_f^s$	22	$C_{pg}^s$
5	h	14	$u_g^s$	23	indices

**Table 12.2-1** Thermodynamic Properties Returned in the s Array

No.	Property	No.	Property	No.	Property
6	$\beta$	15	$h_f^s$	24	$s$
7	$\kappa$	16	$h_g^s$	25	$s_f^s$
8	$C_p$	17	$\beta_f^s$	26	$s_g^s$
9	$x$	18	$\beta_g^s$		

Slot 23, labeled indices in **Table 12.2-1** is a packed 64-bit word. In the top 32-bits part of the word, the subroutines store the last index into the temperature table, and in the bottom 32-bits part of the word, the subroutines store the last index into the pressure table. This is done to speed up the interpolation process.

### 12.2.1 stread - Read Water Properties

The stread subroutine reads the tpfh2o file the five binary tables into memory and initializes the tables. This subroutine is in the relap directory in the RELAP5 distribution. The subroutine statement is:

```
subroutine stread (n, nuse, record, a)
n      FORTRAN unit number from which thermodynamic properties file data
            is read (input)
nuse    Number of words available in array a for storage of binary tables
            (input)
            This value must be greater than or equal to ntot
            Number of words of array a actually needed for the binary tables
            (output)
            = -1 if error detected during binary table read or initialization
            (output)
record   Character array into which information about the binary tables and
            generating program is placed (output)
a       Array into which binary tables are read (output)
```

### 12.2.2 sth2x0 - Saturated Properties, F(T)

Subroutine sth2x0 returns the saturation pressure for a given temperature. This subroutine does not use the binary tables, and therefore, it can be called before the binary tables are read in using the stread subroutine. This subroutine only returns the saturation pressure in p. The err flag is set true if the input temperature is below the triple point temperature or above the critical point temperature. The subroutine statement is:

```
subroutine sth2x0 (t, p, err)
t      Temperature (K)
p      Pressure (Pa)
err    Error flag
      true if t < Ttp or t > Tcp
```

The formula for the saturation pressure as a function of temperature that is used is

$$P^s = P_{cp} \left[ \frac{\sum_{i=1}^5 k_i F_{r1}^i}{F_r(k_7 F_{r1}^2 + k_6 F_{r1} + 1)} - \frac{F_{r1}}{(k_g F_{r1}^2 + k_9)} \right] \quad (12.2-1)$$

where

$$F_r = \frac{T}{T_{cp}} \quad (12.2-2)$$

$$F_{r1} = 1 - F_r \quad (12.2-3)$$

The values of  $k_i$  are given in **Table 12.2-2**. The values for the critical point pressure,  $P_{cp}$ , critical point temperature,  $T_{cp}$ , are given in **Table 12.1-1**.

**Table 12.2-2** Parameters for Saturation Pressure vs. Temperature

Index i	$k_i$
1	-7.691234564d0
2	-26.08023696d0
3	-168.1706546d0
4	6.423285504d1
5	-1.189646225d2
6	4.167117320d0
7	2.097506760d1
8	1.d996.d0

### 12.2.3 sth2x1 - Saturated Properties, F(T,x)

Subroutine sth2x1 returns the saturation properties for a given temperature and quality. The temperature is entered as s(1) and the quality is entered as s(9). The err flag is set true if the input temperature is below the triple point temperature or above the critical point temperature. It is also set true if the input quality is less than 0.0 or greater than 1.0. The two-phase mixture quantities are returned in s(3) through s(5) and s(24). The saturated liquid and vapor values are returned in s(11) through s(22) and s(25)

and s(26). The variable s(10) is set to s(2). The sth2xb entry point in sth2x1 is for skipping over the evaluation of the saturation temperature when it is already known. It is not called by any subroutine in the current version of the RELAP5. The other entry point, sth2x2, is discussed next. The subroutine statement for sth2x1 is:

```
subroutine sth2x1(a, s, err)
a      Binary table array (input)
s      Array of 26 thermodynamic quantities returned by the subroutine
      (see Table 12.2-1)
err    Error flag
      true if t < Ttp or t > Tcp or x < 0.0 or x > 1.0
```

#### 12.2.4 sth2x2 - Saturated Properties, F(P,x)

Subroutine sth2x2 returns the saturation properties for a given pressure and quality. The err flag is set true if the input pressure is below the triple point or above the critical point pressure. It is also set true if the input quality is less than 0.0 or greater than 1.0. This subroutine is an entry point in sth2x1, and therefore, it is found in the sth2x1.ff file in the envrl directory of the RELAP5 distribution. The subroutine statement is:

```
subroutine sth2x2(a, s, err)
a      Binary table array (input)
s      Array of 26 thermodynamic quantities returned by the subroutine
      (see Table 12.2-1)
err    Error flag
      true if t < Ptp or t > Pcp or x < 0.0 or x > 1.0
```

The saturated temperature is obtained from the pressure from equations given in a SHARE program 1095<sup>12.4-3</sup>. The equations and constants were given with pressure in psia and temperature in F. The constants were converted to SI units, Pascals and K. There are two equations: one for low pressures and one for high pressures.

$$T^s = c_1 + \sum_{i=2}^9 c_i [\ln(p_1 P)]^{i-1} \quad p_x \leq P \leq p_y \quad (12.2-4)$$

$$T^s = b_1 + \sum_{i=2}^6 b_i [\ln(p_2 P)]^{i-1} \quad p_y < P < P_{cp} \quad (12.2-5)$$

where the limits on pressure are given in **Table 12.2-3**. The coefficients are given in **Table 12.2-4** and **Table 12.2-5**. The variable names are the same as those in sth2x1.

The equations in the SHARE program were augmented for some calculations with an ice condenser where the pressure is below the triple point. For this case, the following equation is used.

$$T^s = c_{c1}[\ln P]^2 + c_{c2}[\ln P] + c_{c3} \quad p_{tp} \leq P \leq p_x \quad (12.2-6)$$

The coefficients for this equation are given in **Table 12.2-3**.

Once the saturation temperature is obtained, it is adjusted using a single-step Newton iteration to be consistent with the formula for the saturation pressure in Equation (12.2-1).

**Table 12.2-3** Pressure Limits for Saturation Temperature vs. Pressure Formulas

Variable in code	Symbol	Value
pxxx1	$p_x$	378.951459d0
pxxy3	$p_y$	102640.782d0
pxx1	$p_1$	1.450377377d-3
pxx2	$p_2$	1.450377377d-4
crp	$P_{cp}$	22120000.d0
plow	$P_{tp}$	611.2444d0

**Table 12.2-4** Parameters for Saturation Temperature vs. Pressure Formula for Low Pressure Range

Variable in Code	Symbol	Value
c(1)	$c_1$	274.9043833d0
c(2)	$c_2$	13.66254889d0
c(3)	$c_3$	1.176781611d0
c(4)	$c_4$	-.189693d0
c(5)	$c_5$	8.74535666d-2
c(6)	$c_6$	-1.7405325d-2
c(7)	$c_7$	2.147682333d-3
c(8)	$c_8$	-1.383432444d-4
c(9)	$c_9$	3.800086611d-6.

**Table 12.2-5** Parameters for Saturation Temperature vs. Pressure Formula for High Pressure Range

Variable	Symbol	Value
b(1)	$b_1$	6669.352222d0
b(2)	$b_2$	-4658.899d0
b(3)	$b_3$	1376.536722d0
b(4)	$b_4$	-201.9126167d0
b(5)	$b_5$	14.82832111d0
b(6)	$b_6$	-.4337434056d0

**Table 12.2-6** Parameters for Saturation Temperature vs. Pressure Formula for Below Triple Point Pressure

Variable	Symbol	Value
cc(1)	$c_{c1}$	0.84488898d0
cc(2)	$c_{c2}$	2.9056480d0
cc(3)	$c_{c3}$	219.74589d0

### 12.2.5 sth2x3 - Single Phase Properties, F(T,P)

Subroutine sth2x3 returns the single phase properties for a given temperature and pressure. The err flag is set true if the input temperature is below the triple point or above the maximum temperature. It is also set true if the input pressure is less than 0.0 or greater than maximum pressure. The it variable is set to the appropriate value, (1, 3, or 4) depending upon the location of the temperature-pressure point (see **Figure 121-1**). The it variable can never be in the two-phase region, region 2, because temperature and pressure do not determine a unique state in the two-phase region. The subroutine statement is:

```

subroutine sth2x3(a, s, it, err)
a      Binary table array (input)
s      Array of 26 thermodynamic quantities returned by the subroutine
      (see Table 12.2-1)
it     Region in the table
      1      single phase liquid
      3      single phase vapor
      4      above the critical point pressure
err    Error flag
      true if t < Ttp or t > Tmax or p < Ptp or p > Pmax

```

### 12.2.6 sth2x4 - Water Properties, F(T,v)

Subroutine sth2x4 returns the single phase properties for a given temperature and specific volume. The err flag is set true if the input temperature is below the triple point or above the maximum temperature. It is not set true if the input specific volume is out of range. The it variable is set to the appropriate value, (1, 2, 3, or 4) depending upon the location of the temperature-specific volume point (see **Table 12.2-1**). The saturation properties are set to the appropriate values at the input temperature if the input temperature is less than the critical point temperature. The subroutine statement is:

```
subroutine sth2x4(a, s, it, err)
a      Binary table array (input)
s      Array of 26 thermodynamic quantities (see Table 12.2-1)
       t      temperature (input)
       v      specific volume (input)
it     Region in the table
       1      single phase liquid
       2      two-phase
       3      single phase vapor
       4      above the critical point pressure
err    Error flag
       true if t < Ttp or t > Tcp or v < 0.0 or v > 1.0
```

### 12.2.7 sth2x5 - Water Properties, F(P,h)

Subroutine sth2x5 returns the single phase properties for a given pressure and enthalpy. The err flag is set true if the input pressure is less than 0.0 or above the maximum pressure. It is not set true if the input enthalpy is out of range. The it variable is set to the appropriate value, (1, 2, 3, or 4) depending upon the location of the pressure-enthalpy point (see **Table 12.2-1**). The saturation properties are set to the appropriate values at the input pressure if the input pressure is less than the critical point pressure. The subroutine statement is:

```
subroutine sth2x5(a, s, it, err)
a      Binary table array (input)
s      Array of 26 thermodynamic quantities (see Table 12.2-1)
       p      pressure (input)
       h      enthalpy (input)
it     Region in the table
       1      single phase liquid
       2      two-phase
       3      single phase vapor
       4      above the critical point pressure
err    Error flag
       true if p < Ptp or p > Pcp or h is out of range
```

### 12.2.8 sth2x6 - Water Properties, F(P,u)

Subroutine sth2x6 returns the single phase properties for a given pressure and internal energy. The err flag is set true if the input pressure is less than 0.0 or above the maximum pressure. It is not set true if the input internal energy is out of range. The it variable is set to the appropriate value, (1, 2, 3, or 4) depending upon the location of the pressure-internal energy point (see **Table 12.2-1**). The saturation

properties are set to the appropriate values at the input pressure if the input pressure is less than the critical point pressure. The subroutine statement is:

```

subroutine sth2x6(a, s, it, err)
a      Binary table array (input)
s      Array of 26 thermodynamic quantities (see Table 12.2-1)
       p      pressure (input)
       u      internal energy (input)
it     Region in the table
       1      single phase liquid
       2      two-phase
       3      single phase vapor
       4      above the critical point pressure
err    Error flag
       true if p < Ptp or p > Pcp or u is out of range

```

### **12.2.9 psatpd - Saturation Pressure or Temperature and dP/dT**

Subroutine psatpd returns the partial derivative of pressure with respect to temperature along the saturation line and either the saturation pressure (itype = 1) or the saturation temperature (itype = 2). The err flag is set true if itype = 1 and the input temperature is less than the triple point temperature or above the critical point temperature. The err flag is set true if itype = 2 and the input pressure is above the critical point pressure. This subroutine is in the relap directory while the other interpolation subroutines are in the envrl directory in the RELAP5 distribution. The subroutine statement is:

```

subroutine psatpd (t, p, pdt, itype, err)
t      temperature (input if itype = 1, output if itype = 2)
p      pressure (output if itype = 1, input if itype = 2)
pdt   partial derivative of pressure with respect to temperature along
       the saturation line
itype  input flag for psatpd subroutine
       1      evaluate the derivative at the input temperature and return
              the saturation pressure
       2      evaluate the derivative at the input pressure and return
              the return saturation temperature
err    true if the subroutine call was successful, false otherwise

```

## **12.3 Interpolation Procedures**

All the interpolation subroutines use similar table search and interpolation procedures. Weighted derivative interpolation is used or the liquid specific volume interpolation and linear interpolation is used for all the other liquid properties. Linear or reciprocal interpolation is used or the vapor properties based on the perfect vapor relations.

$$P_v = RT \quad (12.3-1)$$

$$u = C_v T \quad (12.3-2)$$

$$h = C_p T \quad (12.3-3)$$

$$\beta = T^{-1} \quad (12.3-4)$$

$$\kappa = P^{-1} \quad (12.3-5)$$

As an example, since  $v = RTP^{-1}$ , linear interpolation is used for the temperature dependence and reciprocal interpolation is used for the pressure dependence. The linear interpolation formula is:

$$y = y_1 + \frac{(x - x_1)(y_2 - y_1)}{(x_2 - x_1)} \quad (12.3-6)$$

and the reciprocal interpolation formula is:

$$y = y_1 + \frac{(x - x_1)(y_2 - y_1)}{(x_2 - x_1)x} \quad (12.3-7)$$

The saturation pressure as a function of temperature is computed using the K function of the IFC formulation. The sth2x0 subroutine is just a coding of this formula. Enthalpy values are obtained from the formula

$$h = u + Pv \quad (12.3-8)$$

Inside the sth2x1 subroutine, the saturation pressure is obtained from the K function. Both the saturation tables are searched for the nearest bracketing values. These are used for the interpolation. The saturated liquid properties are obtained by linear interpolation on temperature. The saturated vapor specific volume is obtained from the following equations:

$$r = \frac{T - T_1}{T_2 - T_1} \quad (12.3-9)$$

$$v = \left[ (1 - r) \frac{P_1}{T_1} v_1 + r \frac{P_2}{T_2} v_2 \right] \frac{T}{P} \quad (12.3-10)$$

Linear interpolation on temperature is used for saturated vapor internal energy. The saturated vapor coefficient of isobaric thermal expansion is obtained from reciprocal interpolation on pressure. The saturated vapor specific heat at constant pressure is obtained from linear interpolation on temperature. The

## References

two-phase values of specific volume, internal energy, and enthalpy are obtained from the saturated liquid and vapor values and the quality, e.g,

$$v = (1 - x)v_f + xv_g \quad (12.3-11)$$

## 12.4 References

- 12.4-1. Meyer, C. A., R. B. McClintock, G. J. Silvestri, R. C. Spencer, Jr. 1967 ASME Steam Tables -- Thermodynamic and Transport Properties of Steam. New York: the American Society of mechanical Engineers (1967).
- 12.4-2. Moore, K. V. ASTEM - A Collection of FORTRAN Subroutines to Evaluate the 1967 ASME Equations of State for Water/Steam and Derivatives of these Equations. ANCR-1026. October 1971.
- 12.4-3. Martin, T. W. SHARE Program 1095. submitted by Westinghouse Electric Corporation.

## 13 RSTPLT FILE STRUCTURE

The restart-plot file, `rstplt`, is used by RELAP5 to restart and continue a problem. It is also the file used by plotting programs like XMGR5 to plot the results of a RELAP5 run and by the strip option to extract data from a RELAP5 run for use in other programs. This file is written in binary on the local disk. The default file name is `rstplt`, but can be set to any filename by use of the `-r` option on the command line.

Inside the code, the `rstplt` file name is stored in the character\*32 variable, `filsch(5)`, in the `/ufilef/` common block, which is in comdeck file `ufilef.hh`. It is initialized to `rstplt` in the main program, RELAP5 and modified to the file name entered on the command line after the `-r` or `-R` option if either of these options are used. In the following discussion, the restart file will be referred to as `rstplt` even though it may have been modified to the name entered on the command line. The unit variable number is stored in the variable `rstplt` in the `/ufiles/` common block, which is in comdeck file, `ufiles.hh`, and is initialized to 13 in RELAP5.

Whenever the restart option is turned on in RELAP5, the `rstplt` file is written. If the RELAP5 problem is a restart of a previous problem, then this file is read up to the restart point and overwritten from this point on. The file is written using buffer out and read using buffer in statements on the CRAY and by Fortran binary writes and reads on all other computers. The type of write/read that is used is under the control of the `bufr` define variable. If `bufr` is defined, then buffer out/in is used. All the binary write records are prefaced by a two-word record containing two integers. The first integer is the number of words in the following record, and the second word is the size in bytes of the words in the following record. The size is always 8 bytes for the restart records. Since this two-word record precedes all `rstplt` records, it is not included in the record descriptions below.

If the problem is a new problem, the `rrstd` subroutine opens the file as a new file. The name of the file is stored in `filsch(5)`. If this file exists, the code exits after it informs the user that this file exists. This behavior can be altered by using the `-R` option on the command line. Normally, the user would use the `-r` option on the command line to specify the restart file name, but if the `-R` option is used instead, the restart file can exist and will be overwritten. This is convenient during a sequence of debug runs in which the programmer wants to skip the process of removing the restart file between each debug run.

If the problem is a restart of a previous problem, the `rrestf` subroutine opens the file as an old file. If the problem is a strip problem, the `srestd` subroutine opens the file as an old file. The restart file must exist in both of these cases.

The file structure consists of six types of records:

1. `rstplt` - identifies the file as a restart-plot file
2. `plotinf` - length information about the following `plotalf` and `plotnum` records
3. `plotalf` - alphanumeric parts of the plot variable/number combination

4. plotnum - numeric parts of the plot variable/number combination
5. plotrec - actual values to be plotted for the plot variable/number combination
6. restart - record header for the restart records

In the description of the record structures, the letters A, I, R4, and R8 represent alphanumeric (Hollerith or character), integer, real\*4, and real\*8 type variables, respectively. The integer variables are two integer\*4 words on the 32-bit computers and one integer\*8 word on the 64-bit computers. Each alphanumeric character string is eight bytes long and padded on the right with blanks if necessary.

### 13.1 Rstplt Record Structure

The rstplt record is the first record of the restart-plot file and is written by the rrstd subroutine. If this record is not correct, the file is not a RELAP5 rstplt file. The record consists of seven 8-byte words. This

**Table 13.1-1** Rstplt Record Structure

Word	Description
1-3 (A)	Program Identification in the form 'Program_Name/Modification_No./Cycle_No.' The information is identical to the first 16 columns at the top of every printed page ' RELAP5/3.2pt '
4-6 (A)	These three words contain the character string: 'restart-plot file '
7-9 (A)	This word contains the date and time the file was written in the form, day-month-year followed by hour:minute:second '12-OCT-01 15:08:12 '
10 (I)	Problem type. This is the variable problemopt611. This flag identifies the type of the problem, transient or a steady-state type. This flag is used to discard the restart-plot file and start over if the problem type changes, e.g., if a steady-state problem is restarted as a transient problem, then the steady-state restart-plot file is discarded and a new rstplt file is started. 2

record is written by rrstd for a new problem and read in and checked by rrestf for a restart problem.

### 13.2 Plotinf Record Structure

The plotinf record is a three-word header record for the plotalf and plotnum records. Plotalf and plotnum records always follow a plotinf record. There is only one plotinf record per restart. For a new problem, there is only one plotinf record in the rstplt file. After a restart, another plotinf record is written to

the file at the location of the restart. This is done because the number of plot variables in the problem could have been changed by the restart deck. The record structure is.

**Table 13.2-1** Plotinf Record Structure

Word	Description
1 (A)	This word contains the character string: "plotinf "
2 (I)	This word contains the length of the plotalf records. Since the length of the plotnum records is the same as the length of the plotalf records, word 2 is also equal to the length of the plotnum records. Example: 379
3 (I)	This word contains the length of the plotrec records. If the values have been compressed using the sqoz subroutine, then word 3 will be equal to $(\text{word}2 + 1)/2$ . If the values have not been compressed, then word 3 will equal word 2. In the present version of the RELAP5 program, the plotrec values are always compressed. Example: 190

### 13.3 Plotalf Record Structure

The plotalf record is a multiple-word record whose length is given by word 2 of the plotalf record. The plotalf record contains all the alphanumeric parts of the plot variables. It is needed to identify the variable for the plots. The identification of the quantities is the same as the alphanumeric part of the variable request code used for the minor edits and plotting. The record consists of 8-byte words. The record structure is:

**Table 13.3-1** Plotalf Record Structure

Word	Description
1 (A)	This word contains the character string: 'plotalf '
2... (A)	This and subsequent 8-byte words contain the alphanumeric part of the label for the variable. Examples of the first one and a later one are: 'time       ' 'rhof       '

### 13.4 Plotnum Record Structure

The plotnum record is a multiple-word record whose length is given by word 2 of the plotalf record. The plotnum record contains all the numeric parts of the plot variables. It is needed to identify the variable

for the plots. The identification of the quantities is the same as the alphanumeric part of the variable request code used for the minor edits and plotting. The record consists of 8-byte words. The record structure is:

**Table 13.4-1 Plotnum Record Structure**

<b>Word</b>	<b>Description</b>
1 (A)	This word contains the character string: 'plotnum '
2... (A)	This and subsequent 8-byte integers contain the numeric part of the label for the variable. Examples of the first one and a later one are: '0 ' '203010000'

## 13.5 Plotrec Record Structure

The plotrec record is a multiple-word record whose length is given in word 3 of the plotinf record. The plotrec record contains the actual values of the variables to be plotted. The record consists of 8-byte records, where each 8-byte word is normally composed of two real\*4 floating point values packed into one 8-byte word. The plotrec records do not necessarily follow the plotinf, plotalf, and plotnum records. The plotrec records can be interspersed in the file with restart records.

**Table 13.5-1 Plotrec Record Structure**

<b>Word</b>	<b>Description</b>
1 (A)	This word contains the character string: "plotrec "
2... (2R4)	This and subsequent 8-byte words contain the numeric value for the variable to be plotted. The first one is time. Example: 634.94

## 13.6 Restart Record Structure

The restart record actually consists of several records. The first record is a three word header record. Subsequent records contain the title of the problem and values from the common blocks.

### 13.6.1 Restart Header Record

The structure of the restart header record is:

**Table 13.6-1** Restart Header Record Structure

Word	Description
1 (A)	<p>This word contains the character string:</p> <pre data-bbox="461 485 612 498">'restart '</pre>
2 (I)	<p>This word contains the "print" variable. the print variable is a packed word used in RELAP5 to store numerous bits of information about the problem.</p> <p>23</p>
3 (I)	<p>This word contains the variable ncount. This variable is the number of advancements, successful or otherwise, taken by the RELAP5 problem.</p> <p>0</p>

### **13.6.2 Restart Title Record**

The structure of the 20 8-byte restart title record is:

**Table 13.6-2** Restart Title Record Structure

Word	Description
1-8 (A)	<p>These eight 8-byte words contain the ptitle character string. This string contains the program version identification and title and is initialized in the comdeck, ptitle.hh.</p> <pre data-bbox="468 1176 1308 1216">' RELAP5/3.2pt          Reactor Loss Of Coolant Analysis       Program'</pre>
9-20 (A)	<p>These 12 8-byte words contain the ctitle character string. This string contains the title card of the case plus the time and date. It is read in the rcards subroutine.</p> <pre data-bbox="468 1256 1308 1298">' ROSA.3.2.Com.PBL.Rev2: ROSA/AP600; Mod 3.2; Common PBL       Configuration           12-OCT-01      15'</pre>

### 13.6.3 Restart Common Block Records

The common blocks used in RELAP5 are written out next as a series of records. First, the normal common blocks that are necessary for restarting the problem are written out. The starting positions and lengths of these normal common blocks are determined in the subroutine gninit. These values are measured in 8-byte words. The values are written as real\*4 or real\*8 words on their respective computers.

After the normal or named common blocks are written to the restart file, the dynamic common blocks that are part of the /fast/ common block are written out.

**Table 13.6-3** Restart Common Block Record Structure

<b>Record</b>	<b>Description</b>
1 (R8)	Named common block /comctl/ This record is currently 724 4-byte words long
2 (R8)	Named common block /contrl/ This record is currently 268 words long
3 (R8)	Named common block /statec/ This record is currently 140 words long
4 (R8)	Named common block /k3all/ This record is currently 114 words long
5 (R8)	Named common block /k3point/ This record is currently 190 words long
6 (R8)	Named common block /lvel/ This record is currently 10 words long
4 (R8)	Dynamic common blocks in the /fast/ named common block. The starting position and length of these dynamic common blocks were computed in the FTB subroutines when these dynamic common blocks were created. These values are measured in 8-byte words. The values are written as real*8 words. The length of this record is problem dependent. Example: voldat dynamic common block.

## 13.7 Summary

In summary, the restart-plot file always starts with a rstplt record. Whenever a plotinf record appears, it is immediately followed by a plotalf and a plotnum record. These records can be followed by one or more plotrec records. The number depends upon the minor edit frequency that was entered on the time step control card. Interspersed among the plotrec records can be one or more blocks of restart records. If the problem is restarted, then the rstplt file will contain another set of plotinf, plotalf, and plotnum records followed by one or more plotrec records and one or more blocks of restart records. This file can become quite large and is often many megabytes long.

## 14 FTB PACKAGE

The FTB package is used in RELAP5 for dynamic memory allocation. RELAP5 is written using FORTRAN 77, which does not contain any dynamic memory allocation statements like malloc in C. Fortran 90<sup>a</sup> does contain dynamic memory allocation statements, e.g., allocate and deallocate, so when RELAP5 is converted to Fortran 90, the FTB package can be removed. Currently, the FTB package is deeply embedded in the RELAP5 code, so its understanding is essential to programmers that make major changes to the RELAP5 code, especially in the input processing subroutines.

The FTB package originally came from the Bettis Atomic Power Laboratory (BAPL). It was part of the Bettis Programming Environment,<sup>14.3-1,14.3-3</sup> and in particular, the Bettis PDQ neutron diffusion program used FTB. The FTB description in the referenced Bettis manual is for CDC 6600 and 7600 computers. Prior to that it was on the Philco 2000 computer. The PDQ code, and consequently the FTB package, was ported to the IBM 360 computer by Dick Wagner when the PDQ code was imported from Bettis [NRTS 1970]. It was later ported by Dick Wagner to other computers including the CDC 176, Cray XMP, and most of the UNIX workstations.

In the 1970s and 1980s, computer memory was expensive and the charging algorithms included a factor for the amount of memory that was used by the program. Therefore, when RELAP5 was being developed, it paid to minimize the amount of memory that RELAP5 used. Thus, the FTB package was used by Wagner in the RELAP5 program for dynamic memory allocation in order to minimize the computer cost when RELAP5 was run.

The following description of the FTB package is paraphrased from the Bettis manual and describes the FTB package as it was used at Bettis. Comments are added in square brackets [comment] to indicate how the FTB package is used in RELAP5 on the UNIX workstations. Since the CRAY XMP computer at the INEEL is no longer in service, and the majority of usage of RELAP5 by the NRC is on UNIX workstations. The following discussion will be directed towards running on UNIX platforms and not the mainframe computers like the CRAY.

### 14.1 Overview

FTB is a collection of subroutines and functions which control the allocation and use of three levels of 7600 storage: (1) small core memory (SCM), (2) large core memory (LCM), and (3) disks. [RELAP5 on a UNIX workstation only uses SCM, but coding for the other items are still part of the package.] The objectives of FTB are to:

- provide storage allocation and storage management capabilities,
- permit file processing independent of the storage device involved,

---

a. FORTRAN 77 is spelled with all capital letters while Fortran 90 is spelled with only an initial capital letter F according to the Fortran 90 standard.

- provide very high disk streaming rates,
- permit the calling program to determine its storage environment by asking appropriate questions, and
- provide extensive software as well as hardware diagnostics.

The largest unit of data in FTB is a file. Each file has a unique integer identification which is used in all communication with FTB. [RELAP5 uses an 8-byte real for the identification number.] In RELAP5, the sign of the identification number is used for controlling which FTB files are written to the restart file. If the sign is positive, the FTB file is written to the restart file, and if it is negative, the FTB file is not written to the restart file. Two file types are permitted, (1) reserve files and (2) process files. [RELAP5 only uses reserve files.]

Reserve files may reside only in SCM or LCM and are generally used for temporary working storage. [On the CDC and Cray computers, FTB was extended to allow reserve files to exist as a word-addressable file on disk.] Space for these files is allocated through FTB subroutine calls, and the storage is accessed directly by the calling program.

Process files may reside in any of the three levels of storage, SCM, LCM, or disk. These files are subdivided into sets, with all the sets of a particular file required to be of the same length. Process files may not be accessed directly by the calling program. They must be accessed, a set at a time, through calls to subroutines in the FTB package. The sets of a process disk file must be processed sequentially, in either the forward or reverse direction. The sets of a SCM or LCM process file may be processed randomly.

Storage available to FTB includes: (1) an SCM array beginning at the origin of blank common, (2) an LCM array beginning at the origin of the direct access area, and (3) a set of tracks on each of up to six<sup>a</sup> logical disks. [RELAP5 uses a labeled common area called /fast/ instead of blank common. It contains a one-dimensional array called fa in RELAP5 and a in the FTB package subroutines. On the early mainframe computers, blank common was allocated at the end of the program, and any excess space could be released back to the operating system after the problem was loaded into memory and running. This allowed codes like RELAP5 to allocate lots of memory when processing the input, and then later, give the memory back to the operating system and run with the minimum space required for the problem. Thus, when the problem was consuming most of its CPU time, it was running with the minimum amount of memory required, which kept the cost down. With virtual memory on UNIX workstations, this space conserving feature is not necessary because the unused memory is paged out to disk.] The using program may declare an FTB file to reside in any of these devices, and storage is assigned accordingly by FTB. The file is then processed by the program independent of the storage level chosen. As required, working areas are allocated automatically in SCM when file processing is initiated and are released when the processing terminates. FTB also uses the SCM array for its own bookkeeping links.

Storage allocation is performed to assign space for process files, reserve files, bookkeeping links, disk buffers, driver queues, flaw tables, and extended descriptions. Whenever storage is required, a search

---

a. The current FTB package in RELAP5 has 6 logical disks and one drum unit.

is made for the first available area which is sufficiently large, and the allocation is made at that location. Thus, every SCM and LCM FTB file occupies contiguous memory locations, and every FTB disk file occupies contiguous record positions. This mode of allocation can lead to a fragmentation of the available storage when FTB files are removed, and the burden for preventing this situation is placed on the calling program. The FTB package includes subroutines to move and compact the storage so that fragmentation can be eliminated.

A special control option is provided for SCM allocation. The user may specify on initialization that the standard allocation is to be backward from the end of the SCM array rather than forward from the beginning. [The file is not stored backward; it starts at a location such that its end is at the end of the allocation area.] Use of this option permits a moving boundary between program text and data storage. This is achieved by placing a single reserve file at the beginning of the SCM array, loading all program overlays at this position (which is actually the origin of blank common), and adjusting the length of the reserve file to reflect the size of the overlay currently in memory. Any space not required for the current overlay is thus available for working storage, and variations in overlay length do not affect the files allocated at the other end of the SCM array. [RELAP5 does not use overlays anymore, but it still allocates SCM from the end of the SCM array for some files. In particular, the one FTB bookkeeping link in RELAP5 is allocated at the end of the SCM area. The procedure in RELAP5 is to allocate the space for a file at the end of the SCM area when the input for the file is being processed, and after all the input is processed and the size of the file is minimized, move the file to the beginning of the SCM area.]

The allocation procedure may be described in terms of the initial location and terminal location of each storage device. For disks, the initial location is the first record position available to FTB, and the terminal location is the last record position. If the standard SCM allocation is being used, the initial location is the first word of the blank common area [/fast/ fa(1) in the current FTB package used in RELAP5] and the terminal location is the last word of this area. However, if the special SCM control option is specified, the initial and terminal locations are reversed, i.e., the last word of the area is the initial location, and first word is the terminal location. [A better description would be to refer to a preferred end for the allocation. This preferred end is set when the FTB package is initialized with a call to ftbint(lowhi). When lowhi is 1, the preferred end is the beginning of the /fast/ common block, and when lowhi is 2, the preferred end is the upper end of the /fast/ common block. In RELAP5, ftbint is called with lowhi equal to 2, so the preferred end is the upper end of the /fast/ common block area. Each later allocation call to FTB has a parameter in the argument list, lohi, that indicates whether this allocation is to be made at the preferred end, lohi = 1, or the other end of the /fast/ common block, lohi = 2. So it is easy to allocate each FTB file at either end.]

Storage for a process file is allocated when the file is opened for the first time, and the search for storage always begins at the initial location of the unit. For each reserve file, the calling program must specify whether the search is to begin at the initial or terminal location of the unit. In addition, a reserve file may be shifted at any time from its present position toward either the initial or terminal location for that unit. The shift is equivalent to deleting the file, performing a new allocation of storage for the file, and then restoring all of the original data. [In the current FTB package, this shift can also include an increase or decrease in the size of the reserve file. This feature is used quite often in RELAP5. When a particular file is being worked on, it is initially allocated with all the available memory. When its final size is known, it is then shifted to either one end or the other of memory and reset to its final size.]

The first 200-word bookkeeping link is allocated on initialization and is always positioned at the SCM initial location. [Read initial location as preferred location in this last sentence. This is at the end of the /fast/ common block SCM storage area in RELAP5.] This first link can hold up to 50 file descriptions. When the last file description is entered in the link, storage is immediately allocated for another link, with the search for storage beginning at the initial [preferred] location. [This never happens in RELAP5 since there are less than 50 file descriptions.] Similarly, whenever a file description is deleted, the last description is moved into the vacant position, and the last link is released if it is no longer needed. [Thus, the descriptions are close-packed in the bookkeeping link, but not necessarily in the order they were initially allocated.] To minimize fragmentation of the SCM array, an attempt is made to shift all links toward the initial [preferred] location before each allocation of SCM storage. [Again, this will never happen in RELAP5 since only the initial link is used.]

An extended description area in SCM is required to support the processing of any [process] file. For a [process] file residing in SCM, the extended description consists of three [8-byte] words which are permanently allocated immediately preceding the [process] file itself. The extended description area for an LCM [process] file (four words) or for a disk [process] file (20 words) is allocated when the file is opened, with the search for storage beginning at the initial [preferred] location. For other than random processing, a buffer capable of holding a single set is appended to the extended description when the allocation is performed. [This extra allocation never happens in RELAP5 because RELAP5 does not use process files.]

The bookkeeping link is 200 words long and is divided into 50 slots, each containing four words. The slots that contain descriptions for each of the 50 files that can be described by a 200-word link. The words are eight bytes long, i.e., real\*8 words. The first slot in the first link describes the link itself. Successive slots describe one FTB file each. Whenever another bookkeeping link is needed, i.e., 48 files have been allocated, the last slot is used to describe the new link of 200 words and the first slot of the new bookkeeping link contains the description of this new FTB link. Thus, the slots can contain a description of a bookkeeping link, a description of a FTB reserve file, or description of a FTB process file. In RELAP5, there are only the initial bookkeeping link and FTB reserve file descriptions.

In RELAP5, the bookkeeping link is allocated from the end of the fa(lfsiz) array in the /fast/ common block. The four-word slots are further subdivided into eight half-words that can be accessed using the ia(2, lfsiz) array. The ia and fa arrays are equivalenced using an equivalence statement so that they start at the same location. This allows access of each half-word using the ia(1,\*) or ia(2,\*) name or each whole word using the fa(\*) name. In the current version of RELAP5, lfsiz is a parameter and is set to 2,200,000 in the fast.hh comdeck, which is in the relap directory of the RELAP5 distribution. The lfsiz parameter is also set to 500,000 in the ftbcom.hh comdeck in the envrl directory of the RELAP5 distribution, but the larger number rules when RELAP5 is loaded.

**Table 14.1-1** shows the structure of the slot for a link. **Table 14.1-2** shows the structure of the slot a reserve file. In the following tables, indx refers to the starting index for the file in the fa or ia array, i.e., for an FTB reserve file, fa(indx) is the value of the first word in that file.

Looking at the tables, we see that link descriptions are made unique by having a zero for the file id, while reserve files are made unique by having a zero for the number of sets (lower half word in slot i+1).

So if the file id is not zero, the description is for either a reserve or process file, and if the number of sets is not zero, then the description is for a process file. [Remember, RELAP5 does not use process files].

**Table 14.1-1** Four-Word File Description Slot for a Link

Address	ia(1,*)	ia(2,*)
fa(i) or ia(*,i)	0.0d0 Zero is reserved for indicating a link description as opposed to a reserve or process file description	
fa(i+1) or ia(*,i+1)	Index of last file in previous link 0 if first link	0
fa(i+2) or ia(*,i+2)	0	1
fa(i+3) or ia(*,i+3)	indx = Location of this link in SCM if first link index = Next link if not the first link, i.e., link starts at fa(indx)	200 = Storage words used by this link

**Table 14.1-2** Four-Word File Description Slot for a Reserve File

Address	ia(1,*)	ia(2,*)
fa(i) or ia(*,i)	id = File identification number (real*8). Can be positive or negative but not zero.	
fa(i+1) or ia(*,i+1)	filsiz = File size in words	0
fa(i+2) or ia(*,i+2)	0	unit = Unit number 1 = SCM 2 = LCM 3-6 = Disk 7 = Drum
fa(i+3) or ia(*,i+3)	indx = Location of the file, i.e., file starts at fa(indx) 0 means no storage is assigned for this file	Amount of storage used by the file in real*8 words

## 14.2 FTB Subroutines And Common Blocks

The FTB subroutines that are used in RELAP5 are described below. Some of these subroutines are functions and some are subroutines and that is indicated by the subroutine or functions statement. Unless indicated otherwise, all these subroutines and functions are in the envrl directory that is part of the RELAP5 distribution.

Information on the status of the FTB files is passed between that various subroutines and functions via the ftbcom.hh comdeck. This comdeck contains either two or three named common blocks, /fast/, /ftb/ and /ftblcm/. The /ftblcm/ named common is only used on computers that have LCM. None of the current

day UNIX or Windows computers have LCM. The first two named common blocks are described first before the subroutines are described because they contain information that is passed between the various subroutines. The FTB subroutines and functions are described in alphabetical order.

#### **14.2.1 /fast/ - FTB SCM Storage Pool**

The /fast/ named common block contains one large array that contains all the FTB files that are located in SCM. It is called the a array in the comdeck, but is also known as the real\*8 fa array as well as the integer ia array in RELAP5. The fa and ia arrays are equivalenced to each other so that both real\*8 and integer variables can be stored in the same storage pool. Since integers on the UNIX workstations are four bytes long, the ia array is doubly dimensioned, i.e., ia(2,i), and only the ia(2,\*) variables are stored in the array. This procedure puts the integers at the bottom half of the real\*8 word. [This has recently been changed for little Endian computers due to the fact that the bottom half of the real\*8 word contains the sign and exponent bits, whereas on a big Endian computer, the bottom half of a real\*8 word only contains the least significant parts of the mantissa. On Intel computers, storing negative integers on top of the sign and exponent bits results in a signaling NaN being generated, which was sometimes changed from to a quiet NaN by flipping the most significant mantissa bit from 0 to 1. This resulted in the integer being changed in memory by the operating system, which caused unusual results in RELAP5. Hence, the switch to store the integers in the top half of the real\*8 word on little Endian computers.] The size of the a array is set in a parameter statement in the ftbcom.hh comdeck at 500,000. It is later increased in the fast.hh comdeck to 2,200,000, which is in the relap directory of the RELAP5 distribution. The largest value is used by the loader when the RELAP5 executable is built, so in RELAP5, there are 2,200,000 words in the fa array.

#### **14.2.2 /ftb/ - FTB Variables**

The /ftb/ named common contains the variables that are used to pass information between the various FTB subroutines and functions. These variables are:

```

first    integer flag, indicates the FTB subroutines have been initialized
         initialized to 0 in ftbftb.F
         set to iftb after the call to ftbftb.F
         iftb is initialized to 12345 in a data statement in ftbint.F
size(7)  number of words available on each unit
maxz(7)  address of last available word in pool on this unit
         a(maxz(i)-1) = last location we can store information into
minz(7)  address of first available word on this unit
         a(minz(i)) = first location we can store information into
nolink   number of link tables
nofils   number of file descriptions in the current link table
link1    address of first link table (RELAP5 uses only one link table)
lasdes   address of last described file
nexdes   address of location to place next file description
ndsk2    number of disk units defined (set to 7 in ftbftb.F)
reclim   length of buffer for disk files
         initialized to 1024 in ftbftb.F
ireclt(5) buffer size on units 3-7 (disk units)
szz(8)   initial space on each unit
         on units 1, 2, and 8, szz is initialized to 0 in ftbftb.F
         on units 3-7, szz is initialized to 100,000,000 in ftbftb.F
hilo     logical flag indicating preferred end of SCM

```

```

true if preferred end is the high end (RELAP5 uses this end)
false if the preferred end is the low end
dlt logical flag to control link shifting
false implies that links are shifted as far as possible
dly logical flag that is set when a write is issued to force a check
before deleting the buffer
dpn(4) logical flag to check for open files on units 3-6
true if a file is open on the unit
ftbun(5) this is only added if Cray is defined
these are the disk unit numbers
they are initialized in a data statement to 21-25 in ftbftb.F

```

### **14.2.3 dmpfil - Print a FTB File**

The dmpfil subroutine is used to print to the output unit the reserve FTB file or a closed FTB process file having the given identification number, id. The format argument controls the format of the numbers that are printed, 0 for integer and 1 for floating point. The dmpfil subroutine is not called in RELAP5. The subroutine statement is:

```

subroutine dmpfil (id, format)
id      file identification number input
format  file format where 0 = integer, 1 = floating point; input

```

### **14.2.4 dmplst - Print Bookkeeping Information for FTB Files**

The dmplst subroutine is used to print information stored in the FTB bookkeeping links to the output unit. Once the ftbint subroutine has been called successfully, all errors in FTB usage call dmplst before calling the abnormal termination subroutine, fabend. The dmplst subroutine is a general subroutine and will print the bookkeeping information for any set of FTB files. The fldmp subroutine, which is described below, is a more useful for dumping out this same information for the FTB files used in RELAP5. The dmplst subroutine is called by trnset and wrplid which are in the relap directory and by ftberr which is in the envrl directory of the RELAP5 distribution. The subroutine statement is:

```
subroutine dmplst
```

The output from an earlier edhtrk.i input deck is shown in **Table 14.2-1**. This table is not written to the RELAP5 output file anymore. It has been replaced by the more useful information in **Table 14.2-2**.

In **Table 14.2-1**, the number in the first column corresponds the number that RELAP5 uses for the file and is the index for the file in the filid array. For example, filid(4) is the volume data block, voldat, and this is where RELAP5 stores its data for the hydrodynamic volumes. The second column is the filid number that was assigned in RELAP5. Positive filids are written to the restart file. The third column is the address of the file in the fa array and the last column is the size of the file. For example, the volume data

block, voldat, which is number 4, has a filid of 10.0, is written to the restart file, starts at fa(2952), and is 6595 words long.

**Table 14.2-1** Output from dmplst for the edhtrk.i Input Deck

<b>List of Dynamic Storage Information for Transient Calculation</b>			
<b>Number</b>	<b>filid</b>	<b>File Index</b>	<b>File Size</b>
1	27.0	32697	9274
2	3.0	2	28
3	9.0	9547	50
4	10.0	2952	6595
5	11.0	311	2641
6	-18.0	16097	16156
7	6.0	232	13
8	12.0	9597	3071
9	13.0	12668	250
10	19.0	13746	142
11	15.0	13174	48
12	-4.0	30	76
13	-24.0	32253	2
14	22.0	14438	102
16	23. 0	14540	1557
18	7.0	245	65
20	21.0	13889	549
21	14.0	12918	256
27	16.0	13222	476
28	8.0	310	1
30	17.0	13698	48
33	5.0	106	126
35	-25.0	32255	157
40	26.0	32412	285
43	20.0	13888	1

#### **14.2.5 ftbchk - Buffered Input Output Synchronization Check**

The ftbchk subroutine is used to synchronize the ftbin and ftbout calls. The ftbin and ftbout subroutines transfer information between SCM or LCM and disk. The current versions of ftbin and ftbout are synchronized so the ftbchk subroutine is a do-nothing subroutine. The ftbchk subroutine is in the ftbout.ff file in the envrl directory in the RELAP5 distribution. The subroutine statement for ftbchk is:

```
subroutine ftbchk (n)
n          dummy argument, not used input
```

#### **14.2.6 ftbcls - File Close**

The ftbcls subroutine is used to close an FTB file. It is similar to the original close subroutine in the BAPL FTB package. A file that has been opened in random mode may be closed at any time. A file that has been opened in read, write, or read-write mode may not be closed by the ftbcls subroutine until all sets have been processed. The ftbcls subroutine is called from dmpfil and ftbcpy which are in the envrl directory of the RELAP5 distribution. The subroutine statement is:

```
subroutine ftbcls (id)
id        file identification number; input
```

#### **14.2.7 ftbcpy - File Copy**

The ftbcpy subroutine is used to copy a "from" FTB process file id1 to a "to" FTB process file id2 and is similar to the cpyfil subroutine in the BAPL FTB package. The file identification of the second file must be described, the setsize and number of sets of the two files must be the same, and the file identification of the from file must not equal the file identification of the to file. Both files cannot be on the same disk or drum unit, and both files must be closed when ftbcpy is called. Since RELAP5 does not use process files, the ftbcpy subroutine is not called. The subroutine statement is:

```
subroutine ftbcpy (id1, id2)
id1      from file identification number; input
id2      to file identification number; input
```

#### **14.2.8 ftbdel - File Delete**

The ftbdel subroutine is used to delete all traces of a FTB file. The file description in the bookkeeping links and all storage is deleted. A process file must be closed before it can be deleted. This subroutine is similar to the delete subroutine in the BAPL FTB package. The ftbdel subroutine is called from cmpcom, fpinit, fspread, gninit, icmpn1, icompn, ihtcmp, imlp, invhts, invjt, irflht, levskt, rcards, rchng, rcompn, rconvr, rgntbl, rhelp, rhtcmp, rintry, rmadat, rmflds, rmiedt, rnewp, rr5pvmc, rrestf, rrkin, rssi, rstrip, rtrip, rtsc, rusrvr, trnfin, trnset, tsets, and wrplid which are in the relap directory and from dmpfil and nanvd which are in the envrl directory of the RELAP5 distribution. The subroutine statement is:

```
subroutine ftbdel (id)
id        file identification number; input
```

### 14.2.9 fildmp - Print Bookkeeping Information for RELAP5 FTB Files

The fildmp subroutine is used to print the FTB bookkeeping information stored in the links for the current RELAP5 FTB files to the output unit. The fildmp subroutine is in the relap directory on the RELAP5 distribution. It is called from trnset which is in the relap directory of the RELAP5 distribution. The subroutine statement is:

```
subroutine fildmp
```

The fildmp output from the edhtrk.i input deck is shown in **Table 14.2-2**. The entries in this table are sorted by storage location in the fa array. When a programmer encounters odd behavior, like memory values changing unexpectedly, knowing which information is stored where in the big fa storage pool can be quite useful for tracking down the source of the error.

The number in the first column corresponds the number that RELAP5 uses for the file and is the index for the file in the filid array. For example, filid(num = 4) is the volume data block, voldat, and is where RELAP5 stores its data for the hydrodynamic volumes. The second column is the filid number that was assigned in RELAP5. Positive filids are written to the restart file. The third column is the address of the file in the fa array. The fourth column is the name of the comdeck that is used to assign storage in the fa array and the last column is the size of the file. For example, the volume data block, voldat, which is number 4, has a filid of 9.0, is written to the restart file because the fileid is positive. It starts at fa(3544) and is described in the voldat.hh comdeck. It is 7666 words long. The minor edit data, which is the miedtc.hh common block, is not written to the restart file because it has a negative filid of -4.0. This means the user has to re-enter all the minor edit cards on a restart.

**Table 14.2-2** Output From fildmp for the edhtrk.i Input Deck

List of dynamic storage information for transient calculation				
Number	filid	File Index	Name	File Size
2	3.0	2	tstpct.hh	70
12	-4.0	72	miedtc.hh	76
7	5.0	148	intrac.hh	13
18	6.0	161	trpbblk.hh	101
28	7.0	262	sysdatc.hh	1
5	10.0	263	jundat.hh	3281
4	9.0	3544	voldat.hh	7666
3	8.0	11210	cmpdat.hh	98
8	11.0	11308	htsrcm.hh	3659
9	12.0	14967	matdat.hh	250
21	13.0	15217	rkinchhh	267

**Table 14.2-2** Output From fldmp for the edhtrk.i Input Deck

List of dynamic storage information for transient calculation				
Number	filid	File Index	Name	File Size
11	14.0	15484	gentblc.hh	63
27	15.0	15547	convarc.hh	710
33	16.0	16257	usrvar.hh	11
30	17.0	16268	lpdat.hh	68
10	19.0	16336	invtbl.hh	262
43	20.0	16598		1
20	21.0	16599	statc.hh	549
14	22.0	17148	htsttab.hh	102
16	23.0	17250	miedtc.hh	1557
6	-18.0	18807	stcom.hh	16156
13	-24.0	34963	tdpptr.hh	2
35	-25.0	34965	lvectr.hh	157
40	26.0	35122	Sparmat.hh	285
1	-27.0	35407	scrtch.hh	9274

#### 14.2.10 ftbdsb - Create a New Process File

The ftbdsb subroutine describes a FTB process file and is similar to the dscrib subroutine in the BAPL FTB package. A process file must be described before any other calls that operate on that process file can be made. The ftbdsb subroutine is called from ftbrsv which is in the relap directory in the RELAP5 distribution. Even though process files are not used in RELAP5, the ftbdsb subroutine is called because of the way ftbrsv initializes the bookkeeping links. (See the ftbrsv description). The subroutine statement is:

```
subroutine ftbdsb (id, setsiz, nosets, unit)
id      file identification number input
setsiz set size; input
nosets number of sets; input
unit    the unit number; input
        1 = SCM, 2 = LCM, 3-6 = disk, 7 = drum;
```

#### 14.2.11 ftberr - Write FTB File Errors

The ftberr subroutine prints out the error numbers from FTB package. The error numbers and their meaning are given in the **Table 14.2-3**. Some error messages can occur in the initialization phase before the ftberr subroutine can be called. These error messages are described as part of those initialization

subroutines, ftbint and ftbnic. After the error number is printed, this subroutine calls dmplst to print out the bookkeeping links, and then calls fabend to abort the program. The ftberr subroutine is called from dmpfil, ftbcls, ftbcpy, ftbdel, ftbdsb, ftbexp, ftbget, ftblct, ftbopn, ftbpr1, ftbrdc, ftbrsv, ftbsft, ftbtnc, inxget, isfdes, isfopn, issfrg, lavail, lcntgs, lcontg, lifopn, mxsets, nfsets, nfsiz, and nfunit which in the envrl directory of the RELAP5 distribution. The subroutine statement is:

```
subroutine ftberr (err)
err      error number in Table 14.2-3
```

**Table 14.2-3** Ftb Errors

No.	Description
1	Another file is already open on disk unit in ftbopn.
4	Reserve file cannot be closed or truncated in ftbcls.
5	Files in read-write or random modes cannot be truncated in ftbtnc.
6	File id wrong on block just read, probably disk or drum seek error or program error destroyed buffer in ftbpr1.
7	Block number wrong on block just read, probably disk or drum seek error or program error destroyed buffer in ftbpr1.
8	File is a reserve file and shouldn't be in ftbopn or lifopn.
9	File is already open in dmpfil, ftbdel, ftbopn, or lifopn.
11	File is not described in dmpfil, ftbcls, ftbcpy, ftbdel, ftbget, ftbopn, ftbpr1, ftbsft, ftbtnc, inxget, isfopn, lcntgs, lifopn, nfsets, nfsiz, or nfunit.
12	File is not open in ftbcls, ftbpr1, or ftbtnc.
13	File has not been written and has no storage assigned in dmpfil, ftbopn, or ftbtnc.
14	File is already described in ftbdsb.
15	Processing is not random and should be in ftbcpy.
16	File size is less than or equal to zero or is too large in ftbrsv or ftbsft.
17	Incorrect format for dmpfil.
19	File id is zero in dmpfil, ftbcls, ftbcpy, ftbdel, ftbdsb, ftbget, ftbopn, ftbpr1, ftbsft, ftbtnc, inxget, isfdes, isfopn, lifopn, nfsets, nfsiz, or nfunit.
20	Files have same id in ftbcpy.
22	Subroutine parameter for setting preferred end of SCM is incorrect in ftbrsb or ftbsft.
23	Processing mode is random and unit is disk or drum or proc has been called in ftbopn or ftbpr1.
24	Incorrect processing mode specified in subroutine argument in ftbopn.
25	Unit number not equal to 4 in ftbget.
27	No space available for buffers in SCM in ftbopn.

**Table 14.2-3** Ftb Errors

No.	Description
29	No space available for file in disk or drum data sets in ftbopn.
30	No space available for file in SCM or LCM in ftbopn.
31	Number of sets less than or equal to zero in ftbdsb.
32	Not all sets processed on call to ftbcls.
34	Not enough space in SCM for links in ftbdsb.
36	Set number out of range in call to ftbget.
37	Set size greater than that allowed by block size of disk or drum data set in ftbopn.
38	Set size less than or equal to zero in ftbdsb and mxsets.
39	Files do not have same number of sets in ftbcpy.
40	Files do not have same setsize in ftbcpy.
41	Calls to proc exceed number of sets in file in ftbpr1.
42	Unit number not between 1 and 7 in ftbdsb, ftbrsv, issfrg, lavail, lcngts, lcontg, lifopn, or mxsets.
43	File is not a reserve file and should be in ftbsft, or lcngts.
47	Enough space is available in SCM, LCM, disk, or drum, but it is not contiguous space in ftblct.
48	File size cannot be increased without moving file in ftbsft.
49	Shifting a disk or drum file is not allowed in ftbsft.
61	Size and maxz-minz for SCM are not equal in ftbrdc.
62	Size and maxz-minz for LCM are not equal in ftbrdc.
63	Link 1 is not at correct location in ftbrdc.
65	Incorrect ityp argument to ftbexp.
66	Not enough space to expand FTB in ftbexp.

#### 14.2.12 ftbexp - Expand FTB Memory

The ftbexp subroutine expands the FTB space in SCM and LCM. If ityp is 0, then i1 and i2 become the new field lengths. If ityp is 1, then SCM is lengthened by i1 and LCM is lengthened by i2. The ftbexp subroutine is called from rrestf, strip, and trnfin which are in the relap directory of the RELAP5 distribution. The subroutine statement is:

```
subroutine ftbexp (ityp, i1, i2)
```

```

ityp    flag for type of expansion input
i1      if ityp = 0, i1 = new SCM field length; output
        if ityp = 1, SCM field length is lengthened by i1; input
i2      if ityp = 0, i2 = new LCM field length; output
        if ityp = 1, LCM field length is lengthened by i2; input

```

#### **14.2.13 ftbftb - Initialize FTB Arrays**

The ftbftb subroutine initializes the FTB szz array. There is one szz element each for SCM, LCM, the five disk units, and the one drum unit. The five disk unit szz elements are initialized to 100,000,000 and the rest are initialized to 0. On the Cray, the disk units are 21 through 25 and those units are set in the ftbftb subroutine. The ftbftb subroutine is called from ftbint which is in the envrl directory of the RELAP5 distribution. The subroutine statement is:

```
subroutine ftbftb
```

#### **14.2.14 ftbget - Read a Set from a Process File**

The ftbget subroutine is used to obtain a set from a FTB process file that has been opened in random mode. The set is moved to location locx. Since ftbget operates on a process file, and RELAP5 uses no process files, this subroutine is not called. The ftbput subroutine is an entry point in the ftbget.ff file. The ftbget subroutine statement is:

```

subroutine ftbget (id, set, locx)
id      file identification number; input
set     the set number; input
locx   location to move the set to; input

```

#### **14.2.15 ftbin - Buffered Input from Disk**

The ftbin subroutine transfers information between the disk and SCM or LCM. The lun parameter is two less than the unit parameter that is used in the ftbrsv and ftbdsb subroutines. So lun = 1 corresponds to unit = 3. The buf parameter is the word address in SCM or LCM, the parameter nw is the number of words, and parameter pos is the index on disk where the information is located. The workstation version of ftbin calls the ftbmov subroutine to do the actual moving of data. In addition, the transfer is from a virtual disk, which is actually another location in memory, zzz in the /virtul/ named common block, rather than from a real hard disk. The ftbin subroutine is the ftbout.ff file in the envrl directory in the RELAP5 distribution. The subroutine statement for ftbin is:

```

subroutine ftbin (lun, buf, nw, pos)
lun    unit number minus 2; input
buf    SCM or LCM address; input
nw     number of words; input
pos    position on disk; input

```

#### **14.2.16 ftbint - Initialize FTB Package**

The ftbint subroutine initializes the FTB package and must be one of the first calls to the FTB package subroutines. The ftbint subroutine is similar to the initial subroutine in the BAPL FTB package. The lowhi argument sets the preferred end of SCM for FTB file storage. Only the call to ftbmem can

precede a call to ftbint. The ftbint subroutine can be called only once. The ftbint subroutine is called by gninit in the relap directory of the RELAP5 distribution. The ftbint subroutine is called with lowhi set to 2 so that the preferred end of SCM is the high end. This means that the bookkeeping link, which is just a reserve file, is also placed at the high end of SCM. The subroutine statement is:

```
subroutine ftbint (lowhi)
  lowhi  preferred location of SCM for the bookkeeping links; input
         1 = put the bookkeeping links at the beginning of SCM
         2 = put the bookkeeping links at the end of SCM (RELAP5 uses this
              end)
```

Two error messages can occur during the call to ftbint. Normally, error messages are handled by ftberr, but ftberr can only be called after the call to ftbint has returned. The two error messages from ftbint are:

```
***** not sufficient fast core space for ftb subroutines.
***** argument error to ftb initialization
```

The first error occurs if there is less than 200 locations allocated to the a array in SCM, and the second error occurs if the lowhi argument is not 1 or 2.

#### **14.2.17 ftblct - Find Contiguous Space on a Unit**

The ftblct subroutine is used to find a contiguous space of length siz on the given unit. The starting location is returned as the parameter, is. The ftblct subroutine is similar to the locate subroutine in the BAPL FTB package. The ftblct subroutine is called by ftbdsb, ftbopn, ftbrsv, ftbsft, ftbslk, and lcontg which are in the envrl directory of the RELAP5 distribution. The subroutine statement is:

```
subroutine ftblct (unit, siz, is)
  unit    the unit number; input
          1 = SCM, 2 = LCM, 3-6 = disk, 7 = drum;
  siz     file size requested; input.
  is      address of the space that was located; output.
```

#### **14.2.18 ftbmem - Measure and Modify FTB Memory Size**

The ftbmem subroutine measures and modifies the FTB memory size. It is called from gninit which is in the relap directory and ftbexp, ftbint, and ftbrdc which are in the envrl directory of the RELAP5 distribution. The ftbmem subroutine is in the relap directory in the RELAP5 distribution. The subroutine statement is:

```
subroutine ftbmem (lscm, llcm)
  lscm   one plus the last word address in a(i) array in /fast/ common block
         in SCM storage; inputoutput.
         if 0 on input, returns one plus the last word address in a array
  llcm   was used for a similar purpose for the LCM storage, not used in
         RELAP5
```

### 14.2.19 ftbmov - Move a Block of Words in Memory

The ftbmov subroutine moves data from array a to another array b. The move can be forward, i.e., element a(1), then element a(2), etc., or it can be backward, i.e., element a(n), then element a(n-1), etc. If the two arrays do not overlap, a forward move is the fastest, and that is obtained by using a positive value for n. If the two arrays overlap, then depending upon their relative starting locations in memory, either a forward move (location of a(1) > location of b(1)) or a backward move (location of a(1) < location of b(1)) is appropriate. A backward move is obtained by using a negative value for n. The subroutine statement is:

```
subroutine ftbmov (a, b, n)
a      from array; input
b      to array; input
n      number of words to move; input
      if n > 0, then a(1)->b(1), a(2)->b(2), ... a(n)->b(n)
      if n < 0, then a(n)->b(n), a(n-1)->b(n-1) ... a(1)->b(1)
```

### 14.2.20 ftbnid - Get Next Available FTB File Identifier

The ftbnid function returns the absolute value of the largest id currently described or reserved plus 1. This function can be used to guarantee that unique file ids are being used. The function statement is:

```
real*8 function ftbnid (dummy)
dummy   a dummy argument; input
```

An error message is printed if ftbint has not been called before the call to ftbnid. The error message is:

```
***** ftbint not first call to ftb subroutines
```

### 14.2.21 ftbopn - Open a Process File

The ftbopn subroutine is used to open an FTB process file. The mode parameter is 1 for read access, 2 for write access, 3 for read-write access, and 4 for random access. If mode is 1, 2, or 3, the files must be processed sequentially. If mode is 4, the unit specified in ftbdsb must be 1 or 2 and the sets are processed randomly by calls to ftbget and ftbput. Storage is allocated to this file if mode is 2 or 4, and the file has not been opened before. It is an error if the first call to ftbopn for a file is with mode equal to 1 or 3. Rewriting a file is not an error, and no additional storage is allocated. Random processing may be preceded or followed by sequential processing. When mode is 1 or 3, the call to ftbopn initiates the first read if the file is a disk or drum file. Since RELAP5 does not use process files, the ftbopn subroutine is not called. The subroutine statement is:

```
subroutine ftbopn (id, mode)
id      file identification number; input
mode    access mode; input
        mode = 1, read access
        mode = 2, write access
        mode = 3, read-write access
        mode = 4, random access
```

### 14.2.22 ftbout - Buffered Output to a Disk

The ftbout subroutine transfers information from SCM or LCM to disk. The lun parameter is two less than the unit parameter that is used in the ftbrsv and ftbdsb subroutines. So lun = 1 corresponds to unit = 3 and lun = 2 corresponds to unit = 4, etc. The parameter, buf, is the word address in SCM or LCM, the parameter, nw, is the number of words, and the parameter, pos, is the index on disk where the information is located. The workstation versions of ftbout call the ftbmov subroutine to do the actual moving of data. In addition, the transfer is to a virtual disk, which is actually another location in memory, zzz in the /virtul/ named common block, rather than to an actual hard disk. The subroutine statement for ftbout is:

```
subroutine ftbout (lun, buf, nw, pos)
lun      unit number minus 2; input
buf      SCM or LCM address; input
nw       number of words; input
pos      position on disk; input
```

### 14.2.23 ftbpr1 - Process One Process File

The ftbpr1 subroutine is used to process the next FTB process file that has been opened in read, write, or read-write access mode. Since RELAP5 does not use process files, ftbpr1 is not called. The subroutine statement is:

```
subroutine ftbpr1 (id, indx)
id       file identification number; input
indx    SCM or LCM address; input
```

### 14.2.24 ftbpr2 - Process Two Process Files

The ftbpr2 subroutine is used to process the next two FTB process files that have been opened in read, write, or read-write access mode. Since RELAP5 does not use process files, ftbpr2 is not called. The subroutine statement is:

```
subroutine ftbpr2 (id1, indx1, id2, indx2)
id1     file 1 identification number; input
indx1   SCM or LCM address 1; input
id2     file 2 identification number; input
indx2   SCM or LCM address 2; input
```

### 14.2.25 ftbpr3 - Process Three Process Files

The ftbpr3 subroutine is used to process the next three FTB process files that have been opened in read, write, or read-write access mode. Since RELAP5 does not use process files, ftbpr3 is not called. The subroutine statement is:

```
subroutine ftbpr3 (id1, indx1, id2, indx2, id3, indx3)
id1     file 1 identification number; input
indx1   SCM or LCM address 1; input
id2     file 2 identification number; input
indx2   SCM or LCM address 2; input
id3     file 3 identification number; input
indx3   SCM or LCM address 3; input
```

### **14.2.26 ftbpr4 - Process Four Process Files**

The ftbpr3 subroutine is used to process the next four FTB process files that have been opened in read, write, or read-write access mode. Since RELAP5 does not use process files, ftbpr4 is not called. The subroutine statement is:

```
subroutine ftbpr4 (id1, indx1, id2, indx2, id3, indx3, id4, indx4)
  id1      file 1 identification number; input
  indx1   SCM or LCM address 1; input
  id2      file 2 identification number; input
  indx2   SCM or LCM address 2; input
  id3      file 3 identification number; input
  indx3   SCM or LCM address 3; input
  id4      file 4 identification number; input
  indx4   SCM or LCM address 4; input
```

### **14.2.27 ftbput - Write a Set to a Process File**

The ftbput subroutine is used to store a set in a FTB process file that has been opened in random mode. The set must be in location locx when the call is issued. The ftbput subroutine is an entry point in the ftbget subroutine and is in the ftbget.ff file which is in the envrl directory of the RELAP5 distribution. Since the ftbput subroutine operates on an FTB process file, and RELAP5 uses no process files, this subroutine is not called. The ftbput subroutine statement is:

```
subroutine ftbput (id, set, locx)
  id      file identification number; input
  set      the set number; input
  locx    location to move the set from; input
```

### **14.2.28 ftbrdc - Release Excess Space**

The ftbrdc subroutine releases all excess SCM space between the last location needed by a file and the end of memory or the beginning of a link if the links are at the high end of SCM. The ftbrdc subroutine is similar to the reduce subroutine in the BAPL FTB package. If the links are at the end of memory, they are moved downward before releasing the space. The space can be regained by calling ftbexp. This subroutine is called by rplotf, rrestf, strip, and trnset which are in relap directory in the RELAP5 distribution. The subroutine statement is:

```
subroutine ftbrdc
```

### **14.2.29 ftbrsv - Create a New Reserve File**

The ftbrsv subroutine is used to define an FTB reserve file. The arguments in the call statement specify the file identification number, the size, and the unit number for the file. The memory address where the file is located is returned. This subroutine is similar to the original reserv subroutine in the BAPL FTB package. The sign of the iunit parameter is used to indicate at which end of the storage space the file is to be allocated. Positive signifies the preferred end, and negative signifies the other end. In RELAP5, the preferred end is the high memory end.

The coding for ftbrsv calls ftbdsb to allocate space for a process file with the number of sets = 1. It then proceeds to set the number of sets in the four-word file description block to 0, which converts the description from a process file to a reserve file. (Weird, but it works).

The ftbrsv subroutine is called by cmpcom, fpinit, fspread, gninit, icmpn1, icompn, ihtcmp, imiedt, imlp, invhts, invjt, irflht, levskt, rcards, rchng, rcompn, rconvr, rgntbl, rhelp, rhtcmp, rintrv, rmadat, rmflds, rmiedt, rr5pvmc, rradht, rrestf, rrkin, rssi, rstrip, rtrip, rtsc, rusrvr, strip, trnset, tsets, and wrplid which are in the relap directory and by dmpfil and nanvd which are in the envrl directory in the RELAP5 distribution. The subroutine statement is:

```
subroutine ftbrsv (id, filsiz, iunit, i2)
id      file identification number; input.
filsiz  file size; input.
iunit   unit number, input
        1 = SCM; 2 = LCM; 3-7 = disk or drum;
positive = allocate from preferred end (top end in RELAP5)
negative = allocate from other end
i2      index of first location in the fa array; output.
```

#### **14.2.30 ftbsft - Shift and Resize a Reserve File**

The ftbsft subroutine changes the size of an FTB reserve file. It also shifts the file id towards the end specified by the lohi parameter. If lohi = 1, the shift is towards the preferred end, if lohi = 2, the shift is towards the other end, and if lohi = 3, there is no shift, just a possible expansion or contraction of the file. The desired size of the shifted file is flsz. After the file has been shifted, the new index is returned in the i2 argument. Shifting a file does not destroy the file. Disk and drum files can not be shifted. They are always located at the beginning of the disk or drum file space. The ftbsft subroutine is similar to the BAPL FTB shift subroutine. The subroutine statement is:

```
subroutine ftbsft (id, flsz, lohi, i2)
id      file identification number; input.
flsz   file size; input.
lohi   flag for preferred location of shift, input
        1 = move file towards preferred end (top end in RELAP5)
        2 = move file towards other end
        3 = no move, just resize
i2      index of first location in the fa array; output.
```

#### **14.2.31 ftbslk - Shift Links Toward Preferred End**

The ftbslk subroutine shifts the FTB bookkeeping links towards the preferred end. The subroutine statement is:

```
subroutine ftbslk
```

#### **14.2.32 ftbtnc - Truncate and Close a Process File**

The ftbtnc subroutine truncates and closes the FTB process file id. It is called trncat in the BAPL FTB package. The subroutine statement is:

```
subroutine ftbtnc (id)
```

```
id      file identification number; input.
```

#### **14.2.33 idfind - Find Index for a File**

The idfind subroutine returns the index i1 for a FTB file having a file identification of id. If file does not exist, then i1 is set to 0.

```
subroutine idfind (id, i1)
id      file identification number; input.
i1      index for the file; output.
```

#### **14.2.34 inxget - Get Index for a File**

The inxget integer function returns the index for file id. The function statement is:

```
integer function inxget(id)
id      file identification number; input.
```

#### **14.2.35 isfdes - Is Process File Described**

The isfdes logical function returns true if the process file id has been described. It returns false if the process file id has not been described. Isfdes is called from fmlwr. The function statement is:

```
logical function isfdes (id)
id      file identification number; input.
```

#### **14.2.36 isfopn - Is Process File Open**

The isfopn integer function returns 1 if the process file id is open. It returns 0 if the process file id is closed. Since RELAP5 does not use process files, isfopn is not called. The function statement is:

```
integer function isfopn (id)
id      file identification number; input.
```

#### **14.2.37 issfrg - Is Space on Unit Fragmented**

The issfrg logical function returns true if the space is fragmented in the unit that contains file id. The function statement is:

```
logical function issfrg (id)
id      file identification number; input.
```

#### **14.2.38 lavail - Space Available in This Unit**

The lavail integer function returns the number of locations available in unit. The number of locations are in words if unit is SCM or LCM, and blocks if unit is a disk or drum. The function statement is:

```
integer function lavail (unit)
unit    the unit number, input
       1 = SCM; 2 = LCM; 3-7 = disk or drum
```

### **14.2.39 Icntgs - Largest Contiguous Space Left on Unit after a File is Loaded**

The lcntgs integer function returns the largest contiguous space on unit assuming the space occupied by file fid is available. This is useful for determining if there is enough space for ftbsft to increase the size of file fid. The function statement is:

```
integer function lcntgs (fid, unit)
fid      file identification number; input.
unit     the unit number, input
        1 = SCM; 2 = LCM; 3-7 = disk or drum
```

### **14.2.40 Icontg - Largest Contiguous Space on Unit**

The lcontg integer function returns the length of the largest contiguous space on unit. For disk and drum units, the length is the product of the largest number of contiguous blocks available and the maximum number of words that can be stored in a block. The function statement is:

```
integer function lcontg (unit)
unit     the unit number, input
        1 = SCM; 2 = LCM; 3-7 = disk or drum
```

### **14.2.41 Lifopn - Space Required to Open This Process File**

The lifopn integer function returns the number of words on unit that are required if process file id is opened. The parameter, unit, must be 1 or 2, and the file must not be open. The required space includes space for the file itself, its extended description, and its buffers. Since RELAP5 does not use process files, lifopn is not called.

```
integer function lifopn (id, unitb)
id      file identification number; input.
unitb   the unit number, input
        1 = SCM; 2 = LCM
```

### **14.2.42 Mxsets - Number of Sets of Given Size on This Unit**

The mxsets integer function returns the number of sets of size setsiz that can be stored on unit. Since RELAP5 does not use process files, mxsets is not called. The function statement is:

```
integer function mxsets (setsiz, unit)
setsiz  set size; input
unit    the unit number, input
        1 = SCM; 2 = LCM; 3-7 = disk or drum
```

### **14.2.43 Nfsets - Number of Sets in a Process File**

The nfsets integer function returns the number of sets in the process file id. Since RELAP5 does not use process files, nfsets is not called. The function statement is:

```
integer function nfsets (id)
id      file identification number; input.
```

#### **14.2.44 nysize - Process File Setsize or Reserve File Size**

The nysize integer function returns the setsize for a process file or the file size for a reserve file having a file identifier of id. It is used in RELAP5 in rcards to read in the previous case data when there are multiple cases. The function statement is:

```
integer function nysize (id)
id      file identification number; input.
```

#### **14.2.45 nfunit - Unit Number for This File**

The nfunit integer function returns the unit number for file id. The function statement is:

```
integer function nfunit (id)
id      file identification number; input.
```

### **14.3 References**

- 14.3-1. Cadwell, W. R. Reference Manual - Bettis Programming Environment. WAPD-TM-1181 (Draft). Bettis Atomic Power Laboratory. Pittsburgh, Pennsylvania. March 1974.
- 14.3-2. NRTS Environmental Subroutine Manual. National Reactor Testing Station. Idaho Falls, Idaho (unpublished) December 1972.
- 14.3-3. Pfeifer, C. J. CDC-6600 Fortran Programming - Bettis Environmental Report. WAPD-TM-668. Bettis Atomic Power Laboratory. Pittsburgh, Pennsylvania. January 1967.

## 15 UNIT TEST PACKAGE

The unit test package is used in RELAP5 for studying the constitutive packages. There are currently three packages. One to study the interphase drag, one to study the interphase heat transfer, and one to study the wall heat transfer.

All three of the unit test packages use the RELAP5 code to drive the unit test driver subroutines and can be run using any input deck. In their most basic form, these packages will give surface plots of the interphase drag coefficient, or interphase heat transfer coefficient, or wall heat transfer coefficient vs. any of the seven independent variables that are used in RELAP5. The independent variables are: pressure, void fraction, internal liquid energy, internal vapor energy, noncondensable quality, liquid velocity, and vapor velocity. In their more elaborate form, these packages can be used to generate surface plots of any internal variable vs. any other two variables. For example, the interphase drag package has been used to plot the drift flux coefficient  $C_D$  vs. liquid velocity and void fraction. These packages are very useful in determining the smoothness of the correlations that are used in the code. They have also been used to look at the smoothness of the transition between adjoining correlations.

All these packages work by having RELAP5 repeat a time step many times with slightly different initial conditions, one repeat for each point on the surface plot. The driver subroutine in RELAP5 writes out a data file for the surface plotting package. The surface plotting package reads the data file and writes a PostScript file that contains the commands for drawing the surface plot. The PostScript file can then be viewed on the CRT screen using ghostview and/or sent to the a PostScript printer. Above the surface plot, there are 7-9 lines of information that detail the conditions that prevailed when the plot was generated. These information lines contain the values of important parameters, e.g., the volume number, junction number, heat transfer structure number, mass flux, quality, etc.

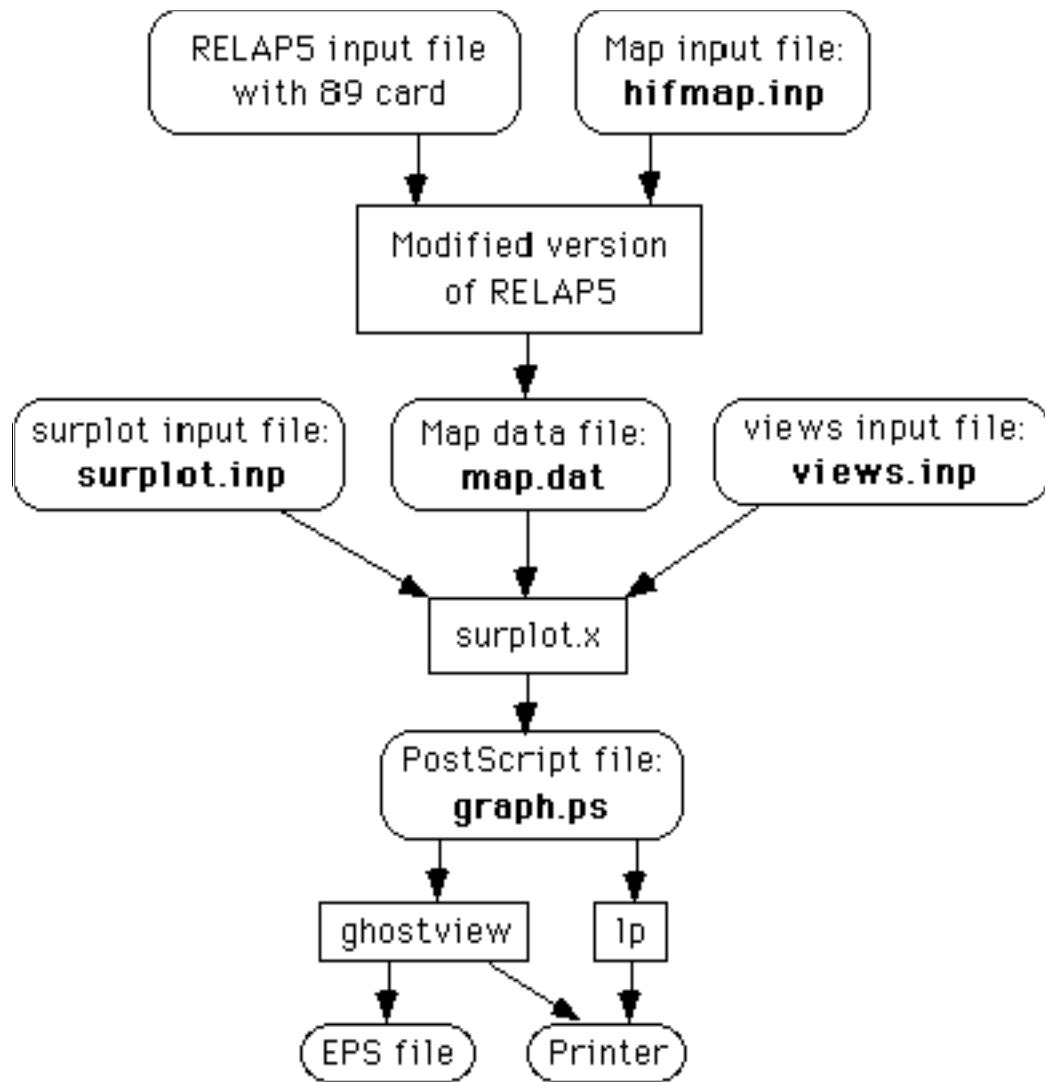
### 15.1 Overall View Of The Unit Test Packages

**Figure 151-1** shows a flow diagram of the flow through the unit test package. Initially, the user starts with a RELAP5 input deck that contains an 89 card. This card contains the name of the subroutine (phantj, phantv, or htcond) to be tested, the value of ncount at which RELAP5 will generate the plot, and the volume, junction, or heat structure number. The map input file is read by the RELAP5 driver subroutine to determine: (1) the number of points and the range for the two independent axis variables, (2) the dependent axis variable name, and (3) a flag that indicates whether to continue or stop the RELAP5 calculation after the map data file is generated. RELAP5 has to have been compiled with the makemap define option included in the define file. This is done by including the word map as the eighth parameter on the dinstls command line when the code is built. Otherwise, the 89 card is flagged as an invalid card.

The surface plotting program, surplot, reads the map data file as well as two other files: a surplot input file, surplot.inp, and a views input file, views.inp. The surplot input file contains the character string that is used as a label for the plot, the name of the map input file, the name of the views input file, the orientation of the plot (landscape or portrait) and the length of the x-, y-, and z-axes. The views input file contains the elevation and rotation angle for each of the views. As many surface plots are made as there are

## Overall View Of The Unit Test Packages

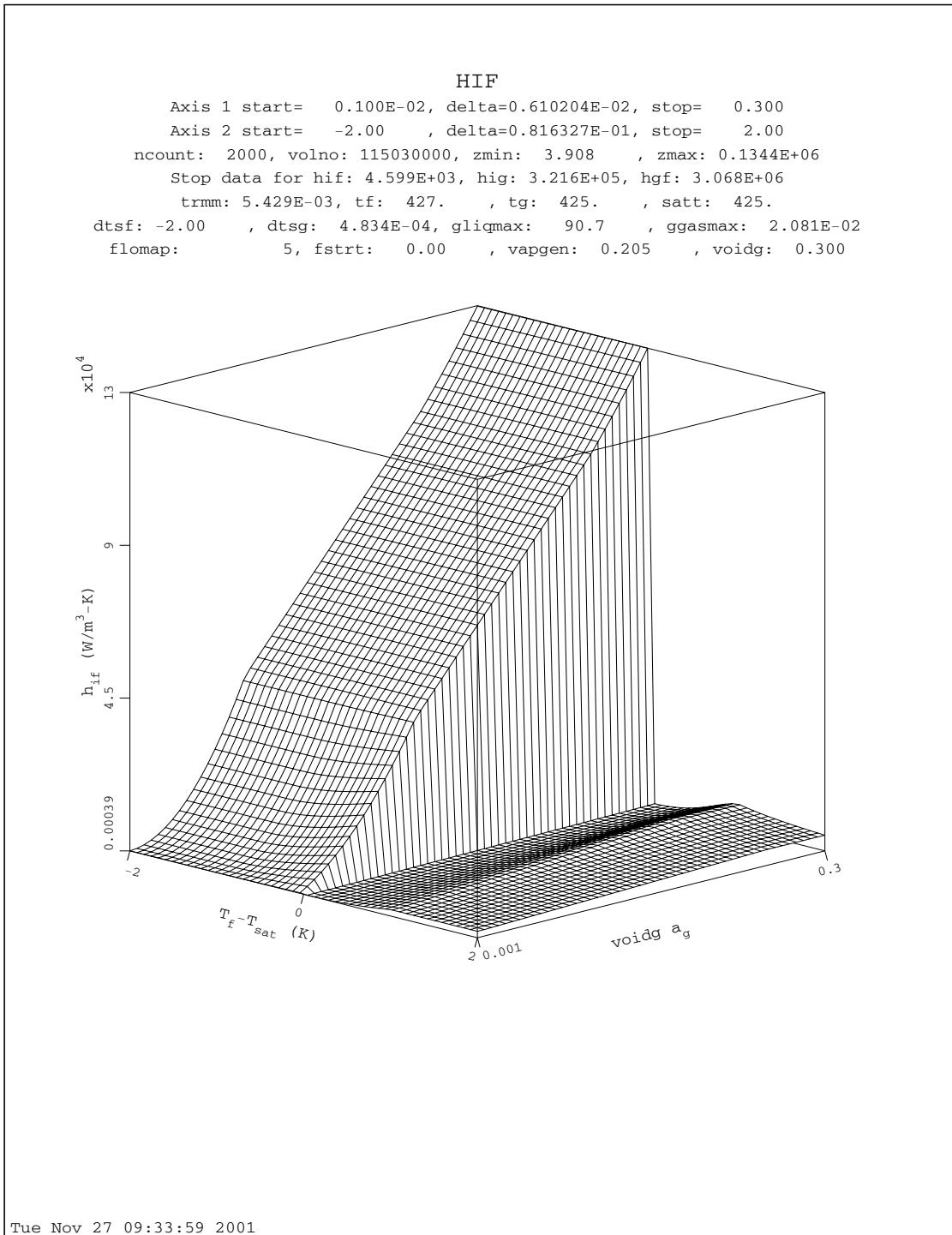
sets of view angles in the views input file. The surface plotting program produces a PostScript output file. This file can be viewed on the CRT screen using ghostview or sent directly to the PostScript printer using the line printer command, lp.



**Figure 151-1** Flow Diagram of the phantvd Unit Test Package for RELAP5

An example surface plot is shown in **Figure 151-2**. This is a plot of the interphase heat transfer coefficient to the liquid,  $h_{if}$ , vs. the void fraction,  $\alpha_g$ , and the liquid superheat,  $\Delta T_{fs} = T_f - T_{sat}$ . The void

fraction ranges from 0.001 to 0.3 and the liquid superheat ranges from -2 to 2. It was made using the phantv driver, phantvd.



**Figure 151-2** Surface Plot from the Unit Test Package for phantv

## **15.2 Interphase Drag Unit Test Package**

The interphase drag unit test package serves as a driver for the phantj subroutine. The driver subroutine is called phantjd.

## **15.3 Interphase Heat Transfer Unit Test Package**

The interphase heat transfer unit test package serves as a driver for the phantv subroutine. The driver subroutine is called phantvd.

## **15.4 Wall Heat Transfer Unit Test Package**

The wall heat transfer unit test package serves as a driver for the htcond subroutine. The driver subroutine is called htcondd.

## **15.5 Surface Plotting Package**

The surface plotting package generates the surface plot as a PostScript file. It can be sent to a PostScript printer or viewed on the screen using ghostview.

## 16 TO DO LIST

As of the end of February 2006, the following items are on the To Do list for Volume 8.

1. Expanded descriptions for some of the relap directory subroutines.
2. Descriptions for the all envrl directory subroutines.
3. The new steam table information from the Steam Table SDID.
4. The installation and maintenance section revisions to reflect the new improved installation method that uses the configure script and makefiles.
5. Descriptions of missing common block variables, e.g., new variables that were created when the bit-packed words were unpacked into new individual logical and integer variables.
6. A section on coding conventions in RELAP5. It can be added using information in the RELAP5 Coding Standards document.



## 17 INDEX OF VARIABLES IN COMDECKS

	Page		Page		
<b>A</b>					
a1	scrtch.hh.....	165	a43	scrtch.hh.....	166
a11	scrtch.hh.....	165	a44	scrtch.hh.....	166
a12	scrtch.hh.....	165	a45	scrtch.hh.....	166
a14	scrtch.hh.....	165	a5	scrtch.hh.....	165
a15	scrtch.hh.....	166	a51	scrtch.hh.....	166
a2	scrtch.hh.....	165	a51s	scrtch.hh.....	166
a21	scrtch.hh.....	166	a52	scrtch.hh.....	166
a22	scrtch.hh.....	166	a52s	scrtch.hh.....	166
a23	scrtch.hh.....	166	a53	scrtch.hh.....	166
a24	scrtch.hh.....	166	a53s	scrtch.hh.....	166
a25	scrtch.hh.....	166	a54	scrtch.hh.....	166
a3	scrtch.hh.....	165	a54s	scrtch.hh.....	166
a31	scrtch.hh.....	166	a55	scrtch.hh.....	166
a32	scrtch.hh.....	166	a55s	scrtch.hh.....	166
a33	scrtch.hh.....	166	abelan	cmpdtv.hh .....	95
a34	scrtch.hh.....	166	abelin	cmpdtv.hh .....	95
a35	scrtch.hh.....	166	acctrp	cmpdac.hh .....	90
a4	scrtch.hh.....	165	accum .....	42	
a41	scrtch.hh.....	166	cmpalf.hh .....	89	
a42	scrtch.hh.....	166	accumulator component.....	90	
			acpgtg	cmpdac.hh .....	90
			acpnit	cmpdac.hh .....	90
			acvgtg	.....	

## INDEX OF VARIABLES IN COMDECKS

adann	cmpdac.hh .....	90	angmm	scrtch.hh .....	167
	cmpdty.hh .....	95	annulus	cmpalf.hh .....	89
adisk	cmpdty.hh .....	95	ANS79 neutron absorption effect	.....	161
advancement statistics	.....	185	apfacn	htsrcm.hh.....	120
advn	statec .....	188	apfaco	htsrcm.hh.....	120
aflag	contrl.hh .....	101	arat	jundat.hh .....	128
aflapr	cmpdty.hh .....	95	arean	htsrcm.hh.....	120
afrr	scrtch.hh.....	166	areao	htsrcm.hh.....	120
afuf	scrtch.hh .....	166	areav	scrtch.hh.....	167
agrq	scrtch.hh .....	167	areaxx	cmpdty.hh .....	95
agug	scrtch.hh .....	167	arhof	scrtch.hh .....	167
agxa	scrtch.hh .....	167	arhog	scrtch.hh .....	167
ahfgtf	cmpdac.hh .....	90	arings	cmpdty.hh .....	95
ahfgtg	cmpdac.hh .....	90	atank	cmpdac.hh .....	91
ahftg	cmpdac.hh .....	90	aterm	scrtch.hh .....	167
ahgtf	cmpdac.hh .....	91	athrot	jundat.hh .....	128
aijr	htscr2.hh.....	118	atlast	cmpdty.hh .....	95
aijt	htscr2.hh.....	118	avelf	scrtch.hh .....	167
ajun	jundat.hh .....	128	avelfg	scrtch.hh .....	167
ajuno	jundat.hh .....	128	avfgtf	cmpdac.hh .....	91
aleak	cmpdty.hh .....	95	avgtg	cmpdac.hh .....	91
alpf10	htrcom.hh .....	113	aviscf	scrtch.hh .....	167
alpg10	htrcom.hh .....	113	aviscg	scrtch.hh .....	167
altlbflg9	rkinc.hh .....	152	aviscn		

cmpdac.hh	91
avisfs	
scrtch.hh	167
avisgs	
scrtch.hh	167
avkx	
scrtch.hh	167
avlx	
scrtch.hh	167
avol	
voldat.hh	198
avrfl	
scrtch.hh	167
avrfg	
scrtch.hh	167
avrg	
scrtch.hh	167
axpf	
htrecom.hh	113
axvelf	
scrtch.hh	167
<b>B</b>	
b1	
scrtch.hh	167
b2	
scrtch.hh	167
b3	
scrtch.hh	167
b4	
scrtch.hh	167
b5	
scrtch.hh	167
badfw	
scrtch.hh	167
bb5	
scrtch.hh	167
bdamp	
cmpdtv.hh	95
beta	
htrecom.hh	113
scrq.hh	163
scrtch.hh	167
betacc	
jundat.hh	129
betaf	
scrq.hh	163
scrtch.hh	168
betaff	
voldat.hh	198
betafs	
scrq.hh	163
scrtch.hh	168
betag	
scrq.hh	163
scrtch.hh	168
betagg	
voldat.hh	198
betags	
scrq.hh	163
scrtch.hh	168
betav	
cmpdac.hh	91
blhcmpopt	
cmpdat.hh	92
boron	
voldat.hh	198
borona	
scrtch.hh	168
boronb	
scrtch.hh	168
borono	
voldat.hh	199
branch	
cmpalf.hh	89
bterm	
scrtch.hh	168
bxvelg	
scrtch.hh	168
<b>C</b>	
c0j	
jundat.hh	129
c0jo	
jundat.hh	129
c0jos	
jundat.hh	129
celvec	
scrtch.hh	168
cflno	
scrtch.hh	168
chf	

## INDEX OF VARIABLES IN COMDECKS

htrcom.hh .....	114	cmptrp ssiblk.hh .....	185
chfmul htrcom.hh .....	114	cmptyp cmpdat.hh.....	92
chfmun htsrcm.hh.....	120	cmpvnm ssiblk.hh .....	185
chfmuo htsrcm.hh.....	120	cmpvno ssiblk.hh .....	185
chnglosscoeff12 htsrcm.hh.....	120	cnvalf cmpvarc.hh.....	106
chngno contrl.hh .....	101	cnavn convarc.hh.....	106
chokef jundat.hh .....	129	cnavro convarc.hh.....	106
cladex htsrcm.hh.....	120	cndct convarc.hh.....	106
claptf cmpdac.hh .....	91	cndft convarc.hh.....	106
clstrp cmpdtv.hh .....	95	cndla convarc.hh.....	106
cmpalf rcompc.hh .....	151	cndli convarc.hh.....	106
cmpalf.hh comdeck .....	89, 150	cnygen convarc.hh.....	107
cmpdac.hh comdeck .....	90	cnvint convarc.hh.....	107
cmpdacc.hh comdeck .....	90	cnvlen convarc.hh.....	107
cmpdat.hh comdeck .....	92	cnvmax convarc.hh.....	107
cmpdatec.hh comdeck .....	92	cnvmin convarc.hh.....	107
cmpdtv.hh comdeck .....	95	cnvnam convarc.hh.....	107
cmpdtvc.hh comdeck .....	95	cnvnop convarc.hh.....	107
cmpflg rcompc.hh .....	151	cvnnpa convarc.hh.....	107
cmpiii ssiblk.hh .....	185	cvnnum convarc.hh.....	107
cmplen cmpdat.hh.....	92	cvnxt cmpvarc.hh.....	107
cmpnam cmpdat.hh.....	92	cnvold convarc.hh.....	107
cmpnum cmpdat.hh.....	92	cvopt convarc.hh.....	107
cmpphi eccmxh.hh .....	109		
cmpsrc rcompc.hh .....	151		
cmptbl cmpdat.hh.....	92		
cmptno ssiblk.hh .....	185		

cnpck	
convarc.hh	107
cnpnmp	
convarc.hh	107
cnsan	
convarc.hh	107
cnscl	
convarc.hh	107
cnsdp	
convarc.hh	108
cnsct	
convarc.hh	108
cnsfr	
cmpvarc.hh	108
cnsin	
convarc.hh	108
cnstbl	
convarc.hh	108
cnspta	
cnvpta.hh	97
cnspta.hh comdeck	97
cnsptad.hh comdeck	97
cnsprt	
convarc.hh	108
cnsotyp	
convarc.hh	108
coefp	
scrtch.hh	168
coefv	
scrtch.hh	168
coefx	
scrtch.hh	168
comctl common block	100
comctl.hh comdeck	98
comctlc.hh comdeck	98
comdat	
comctl.hh	98
comdln	
comctl.hh	98
comlst common block	100
comlst.hh comdeck	100
compid	
ssiblk.hh	185
component variable names	150
components	92
cond	
scrtch.hh	168
connm	
scrtch.hh	168
comnmf	
scrtch.hh	168
comng	
scrtch.hh	168
cons common block	100
cons.hh comdeck	100
constc	
jundat.hh	129
constm	
jundat.hh	129
contid	
ssiblk.hh	185
contrl common block	100, 101
contrl.hh comdeck	101
control variables	106
contrx.hh comdeck	101
convarc.hh comdeck	106
convarx.hh comdeck	106
convf	
scrtch.hh	168
convfs	
scrtch.hh	168
convg	
scrtch.hh	168
convgs	
scrtch.hh	168
coordcodeoroffset1215	
htsrcm.hh	120
correlationnum15	
htsrcm.hh	120
costhe	
scrtch.hh	168
count	
contrl.hh	101
coupfl	
ufiles.hh	197
cp	
scrq.hh	163
scrtch.hh	168
cpf	
scrq.hh	163
scrtch.hh	168

## INDEX OF VARIABLES IN COMDECKS

cpfs	scrq.hh.....	163	scrtch.hh.....	169
	scrtch.hh.....	168	cvmrv	scrtch.hh.....
cpg	scrq.hh.....	163		169
	scrtch.hh.....	168	cvnit	cmpdac.hh.....
cpgs	scrq.hh.....	163		91
	scrtch.hh.....	168	cvrtno	ssiblk.hh.....
cps	htrcom.hh.....	114		185
	scrq.hh.....	163	cvtbl	cmpdty.hh.....
	scrtch.hh.....	168		95
cpurei	contrl.hh.....	101	cwfco	separ.hh.....
cpurem	contrl.hh.....	101		183
csupbf	voldat.hh.....	199	cwgcu	separ.hh.....
	scrtch.hh.....	169	cyl360degflg7	cmpdat.hh.....
csubpg	tstpct.hh.....	195		92
	curclm	195	cylmltisctr5	cmpdat.hh.....
ctermx	curcmlj	195		92
	curcmi	195	<b>D</b>	
	curcmj	195	dbodx1	scrtch.hh.....
	curers	195		169
	curretl	195	dbodx2	scrtch.hh.....
	curtmi	195		169
	curtmj	195	dbodxj	scrtch.hh.....
	curtrs	195		169
	cvaox	195	dbodxv	scrtch.hh.....
	cvelf	195		169
	cvelg	169	dconst	statec .....
				188
			devax	statec .....
				188
			deldim	separ.hh.....
				183
			delgrv	htrcom.hh.....
				114
			delv	scrtch.hh.....
				169
			delxa	scrtch.hh.....
				169
			dfrnto	voldat.hh.....
				199
			dfront	voldat.hh.....
				199
			dialn	cmpdac.hh.....
				91
			diamj	

jundat.hh .....	129	scrtch.hh.....	169
diamjo		dpxgxr	
jundat.hh .....	129	scrtch.hh.....	169
diamtk		dplev	
cmpdac.hh.....	91	voldat.hh .....	199
diamv		dpm	
voldat.hh .....	199	ftbcom.hh .....	111
difdpk		dpstf	
scrtch.hh.....	169	scrtch.hh.....	169
difdpl		drfdp	
scrtch.hh.....	169	voldat.hh .....	199
diff		drfduf	
scrtch.hh.....	169	voldat.hh .....	199
dfg		drgdp	
scrtch.hh.....	169	voldat.hh .....	199
difold		drgdug	
scrtch.hh.....	169	voldat.hh .....	199
difr		drgdxa	
scrtch.hh.....	169	voldat.hh .....	199
divvfx		drho	
scrtch.hh.....	169	scrtch.hh.....	169
divvgx		drings	
scrtch.hh.....	169	cmpdtv.hh .....	95
dl		drivev	
voldat.hh .....	199	scrtch.hh.....	169
dlev		drod	
voldat.hh .....	199	htrcom.hh .....	114
dlevo		dseat	
voldat.hh .....	199	cmpdtv.hh .....	95
dlt		dstar	
ftbcom.hh .....	110	scrtch.hh.....	169
dly		dstvec	
ftbcom.hh .....	111	scrtch.hh.....	169
dmgdt		dt	
cmpdac.hh.....	91	contrl.hh .....	101
dmvvec		dtcrnf	
scrtch.hh.....	169	scrtch.hh.....	169
done		dtcrng	
contrl.hh .....	101	scrtch.hh.....	169
dotm		dtdp	
voldat.hh .....	199	voldat.hh .....	199
dpd		dtdug	
cmpdac.hh.....	91	voldat.hh .....	199
dppdp		dtdxa	
cmpdac.hh.....	91	voldat.hh .....	199
dpdxfx		dtfdp	

## INDEX OF VARIABLES IN COMDECKS

voldat.hh .....	199	voldat.hh .....	200
dtfdfuf		dvelefj	
voldat.hh .....	199	scrtch.hh .....	170
dtgdgp		dvelgj	
voldat.hh .....	199	scrtch.hh .....	170
dtgdug		dvliq	
voldat.hh .....	200	cmpdac.hh .....	91
dtgdxa		dvoidc	
voldat.hh .....	200	lvel.hh .....	140
dtht		dvoidi	
contrl.hh .....	101	lvel.hh .....	140
dthy		dx	
contrl.hh .....	101	scrtch.hh .....	170
dtidp		dxk	
voldat.hh .....	200	scrtch.hh .....	170
dtiduf		dxkx	
voldat.hh .....	200	scrtch.hh .....	170
dtidug		dxl	
voldat.hh .....	200	scrtch.hh .....	170
dtidxn		dxlx	
voldat.hh .....	200	scrtch.hh .....	170
dtlev		dxx	
lvel.hh .....	140	scrtch.hh .....	170
dtmax		dztank	
tstpct.hh .....	195	cmpdac.hh .....	91
dtmin		<b>E</b>	
tstpct.hh .....	195		
dtn		e	
contrl.hh .....	102	cons.hh .....	100
dtsat		ECC mixer variable	109
htrcom.hh .....	114	eccmix	
dtsf		cmpalf.hh .....	89
scrtch.hh .....	170	eccmxcc.hh comdeck	109
dtsfm		eccmxcc.hh comdeck	109
scrtch.hh .....	170	emass	
dtsfsb		contrl.hh .....	102
scrtch.hh .....	170	emasso	
dtsfsp		contrl.hh .....	102
scrtch.hh .....	170	emis	
dtsg		radhtcc.hh .....	149
scrtch.hh .....	170	enliq	
dtsgm		htrcom.hh .....	114
scrtch.hh .....	170	entfs	
dtsgms		scrq.hh .....	163
scrtch.hh .....	170	scrtch.hh .....	170
dttdp		entgs	

scrq.hh	163	scrtch.hh	170
scrtch.hh	170	fast common block	109
enthn		fast.hh comdeck	109
voldat.hh	200	fastc.hh comdeck	109
enths		fberr	
voldat.hh	200	rrkinc.hh	162
entpy		ff	
scrq.hh	163	scrtch.hh	170
scrtch.hh	170	fgrw	
entpyf		scrtch.hh	170
scrq.hh	163	fidxup	
scrtch.hh	170	scrtch.hh	170
entpyg		fifj	
scrq.hh	163	scrtch.hh	171
scrtch.hh	170	figj	
eoин		scrtch.hh	171
ufiles.hh	197	fij	
epsal1		jundat.hh	129
lvel.hh	140	fijo	
erhi		jundat.hh	129
tsctlc.hh	194	fijos	
erro		jundat.hh	129
tsctlc.hh	194	filflg	
erm		comctl.hh	98
scrtch.hh	170	filid	
ermax		comctl.hh	99
contrl.hh	102	filndx	
errmf		comctl.hh	99
scrtch.hh	170	filsch	
extjnn		ufilef.hh	197
jundat.hh	129	filsiz	
extvnn		comctl.hh	99
voldat.hh	200	fines	
<b>F</b>		htrflb.hh	116
f2		first	
scrtch.hh	170	ftbcom.hh	111
fa		fjet	
fast.hh	109	scrtch.hh	171
faaj		fjuf	
jundat.hh	129	jundat.hh	129
fail		fjufb	
contrl.hh	102	jundat.hh	129
fal		fjufc	
scrtch.hh	170	jundat.hh	129
fanms		fjunft	
		jundat.hh	129

## INDEX OF VARIABLES IN COMDECKS

fjnr		
jundat.hh	.....	129
fjunrb		
jundat.hh	.....	130
fjunrc		
jundat.hh	.....	130
fjunrt		
jundat.hh	.....	130
flenlh		
jundat.hh	.....	130
flomap		
scrtch.hh	.....	171
flompj		
scrtch.hh	.....	171
flood common block		110
flood.hh comdeck		110
floreg		
voldat.hh	.....	201
florgj		
jundat.hh	.....	130
fluxm		
scrtch.hh	.....	171
flxtrn		
scrtch.hh	.....	171
fmurex		
voldat.hh	.....	201
formfj		
jundat.hh	.....	130
formmgj		
jundat.hh	.....	130
fp		
scrtch.hh	.....	171
fpress		
scrtch.hh	.....	171
fracag		
scrtch.hh	.....	171
fractal		
scrtch.hh	.....	171
frahaw		
invhtb.hh	.....	127
frica		
voldat.hh	.....	201
fricb		
voldat.hh	.....	201
fricc		
voldat.hh	.....	201
fricfj		
scrtch.hh	.....	171
fricfk		
scrtch.hh	.....	171
fricfl		
scrtch.hh	.....	171
fricgj		
scrtch.hh	.....	171
fricgk		
scrtch.hh	.....	171
fricgl		
scrtch.hh	.....	171
frlmf1		
scrtch.hh	.....	171
frlmf2		
scrtch.hh	.....	171
frlmg1		
scrtch.hh	.....	171
frlmg2		
scrtch.hh	.....	171
fromface1214		
jundat.hh	.....	130
frphpw		
invhtb.hh	.....	127
frtbfl		
scrtch.hh	.....	171
frtbf2		
scrtch.hh	.....	171
frtbg1		
scrtch.hh	.....	171
frtbg2		
scrtch.hh	.....	171
frtrf1		
scrtch.hh	.....	171
frtrf2		
scrtch.hh	.....	171
frtrg1		
scrtch.hh	.....	171
frtrg2		
scrtch.hh	.....	172
fshape		
voldat.hh	.....	201
fstrat		
htrcom.hh	.....	114
fstrt		
voldat.hh	.....	201

fsymb1  
     mxnfcdf.hh ..... 145  
 ftb common block ..... 110  
 ftbcom.hh comdeck ..... 110  
 ftbun  
     ftbcom.hh ..... 111  
 fuf  
     scrtch.hh ..... 172  
 fug  
     scrtch.hh ..... 172  
 fwalf  
     voldat.hh ..... 201  
 fwalfj  
     jundat.hh ..... 130  
 fwalg  
     voldat.hh ..... 201  
 fwalgj  
     jundat.hh ..... 130  
 fwf1  
     scrtch.hh ..... 172  
 fwf2  
     scrtch.hh ..... 172  
 fwfaf1  
     scrtch.hh ..... 172  
 fwfag1  
     scrtch.hh ..... 172  
 fwfxaf  
     scrtch.hh ..... 172  
 fwfxag  
     scrtch.hh ..... 172  
 fwg1  
     scrtch.hh ..... 172  
 fwg2  
     scrtch.hh ..... 172  
 fxa  
     scrtch.hh ..... 172  
 fxa  
     scrtch.hh ..... 172  
 fxj  
     jundat.hh ..... 130  
 fxjo  
     jundat.hh ..... 131  
 fxjos  
     jundat.hh ..... 131

**G**  
 g  
     htrcom.hh ..... 114  
 gabs  
     htrcom.hh ..... 114  
 gal  
     scrtch.hh ..... 172  
 gaman  
     voldat.hh ..... 201  
 gamas  
     voldat.hh ..... 202  
 gammac  
     voldat.hh ..... 202  
 gammaw  
     voldat.hh ..... 202  
 gammsc  
     scrtch.hh ..... 172  
 gammsw  
     scrtch.hh ..... 172  
 gamw  
     htrcom.hh ..... 114  
 gapvar common block ..... 112  
 gapvar.hh comdeck ..... 112  
 gapwd  
     htsrcm.hh ..... 120  
 gasln  
     cmpdac.hh ..... 91  
 gaslno  
     cmpdac.hh ..... 91  
 gcrosf  
     htrcom.hh ..... 114  
 gcross  
     htrcom.hh ..... 114  
 gesub ..... 42  
 gctpm ..... 42  
 gctpmoody ..... 43  
 geaas  
     separ.hh ..... 183  
 geads  
     separ.hh ..... 183  
 geai  
     separ.hh ..... 183  
 gean  
     separ.hh ..... 183  
 geang  
     separ.hh ..... 183

## INDEX OF VARIABLES IN COMDECKS

gebbs		
separ.hh	.....	183
gecks		
separ.hh	.....	183
gedds		
separ.hh	.....	183
gedry	.....	46
geefflds		
separ.hh	.....	183
gehds		
separ.hh	.....	183
gehksks		
separ.hh	.....	183
gendtr		
convarc.hh	.....	108
general table variables	.....	112
generatorflag4		
convarc.hh	.....	108
genfr		
convarc.hh	.....	108
genint		
convarc.hh	.....	108
genpow		
convarc.hh	.....	108
genrl common block	.....	112
genrl.hh comdeck	.....	112
genrlc common block	.....	112
gensvl		
convarc.hh	.....	108
gentblc.hh comdeck	.....	112
gentblx.hh comdeck	.....	112
gentrp		
convarc.hh	.....	108
gentrq		
convarc.hh	.....	108
genvel		
convarc.hh	.....	108
geometryflag4		
cmpdat.hh	.....	92
gerh		
separ.hh	.....	183
gerr		
trnhlp.hh	.....	192
gerr1		
separ.hh	.....	183
gerrs		
lpdat.hh	.....	137
trnhlp.hh	.....	192
gerrss		
separ.hh	.....	184
gerws		
separ.hh	.....	184
gfwabs		
scrtch.hh	.....	172
gg		
scrtch.hh	.....	172
ggas		
voldat.hh	.....	202
ggasa		
htrcom.hh	.....	114
gliq		
voldat.hh	.....	202
gliqa		
htrcom.hh	.....	114
gmaxfctrstarflg14		
rkinc.hh	.....	152
gp		
scrtch.hh	.....	172
gpintp		
htsrcm.hh	.....	120
gprinc		
htsrcm.hh	.....	121
gprouf		
htsrcm.hh	.....	121
gpudis		
htsrcm.hh	.....	121
gravcn		
contrl.hh	.....	102
gravv		
voldat.hh	.....	202
grdkfn		
htsrcm.hh	.....	121
grdkfo		
htsrcm.hh	.....	121
grdkrn		
htsrcm.hh	.....	121
grdkro		
htsrcm.hh	.....	121
grdzfn		
htsrcm.hh	.....	121
grdzfo		
htsrcm.hh	.....	121

grdzrn	
htsrcm.hh	121
grdzro	
htsrcm.hh	121
gridk	
htrcom.hh	114
gridz	
htrcom.hh	114
gsum	
scrtch.hh	172
gtarg	
gentblc.hh	112
gtbl	
gentblc.hh	112
gbptr	
gentblc.hh	113
gtinfo	
gentblc.hh	113
gtlen	
gentblc.hh	113
gtnum	
gentblc.hh	113
gttrp	
gentblc.hh	113
gttyp	
gentblc.hh	113
gtval	
gentblc.hh	113
guf	
scrtch.hh	172
gug	
scrtch.hh	172
gxa	
scrtch.hh	172
gxaa	
scrtch.hh	172
<b>H</b>	
h2gen	
htsrcm.hh	121
h2geno	
htsrcm.hh	121
haslatchingoccurred5	
cmpdat.hh	92
hbar	
scrq.hh	163
scrtch.hh	172
hd	
htrcom.hh	114
headntorqmltplrlflg8	
cmpdat.hh	93
headntorqmltplrsatflg9	
cmpdat.hh	93
heat structure variables	120
heat transfer scratch variables	119
heathydrocoupledimplct2	
tstpct.hh	195
heathydrotimestepequal1	
tstpct.hh	195
heatstrcttmpblkcomittd6	
tstpct.hh	195
help	
contrl.hh	102
hetrat	
htsrcm.hh	121
hfg	
htrcom.hh	114
scrtch.hh	172
hfpg	
htrcom.hh	114
hgf	
voldat.hh	202
hgfc	
scrtch.hh	172
hgfc1	
scrtch.hh	172
hgfo	
voldat.hh	202
hgfos	
voldat.hh	202
hggff	
scrtch.hh	173
hgrad	
voldat.hh	202
hif	
voldat.hh	202
hifc	
scrtch.hh	173
hifc1	
scrtch.hh	173
hiff	
scrtch.hh	173

## INDEX OF VARIABLES IN COMDECKS

**hifh**  
 scrtch.hh ..... 173  
**hifhdt**  
 scrtch.hh ..... 173  
**hifo**  
 voldat.hh ..... 202  
**hifos**  
 voldat.hh ..... 202  
**hig**  
 voldat.hh ..... 202  
**higc**  
 scrtch.hh ..... 173  
**higc1**  
 scrtch.hh ..... 173  
**higg**  
 scrtch.hh ..... 173  
**high**  
 scrtch.hh ..... 173  
**highdt**  
 scrtch.hh ..... 173  
**higo**  
 voldat.hh ..... 202  
**higos**  
 voldat.hh ..... 202  
**higsub**  
 scrtch.hh ..... 173  
**hilo**  
 ftbcom.hh ..... 111  
**hlossf**  
 scrtch.hh ..... 173  
**hlossg**  
 scrtch.hh ..... 173  
**hmac**  
 flood.hh ..... 110  
**hmic**  
 flood.hh ..... 110  
**hring1**  
 cmpdtv.hh ..... 95  
**hring2**  
 cmpdtv.hh ..... 96  
**hsgf**  
 scrtch.hh ..... 173  
**hsteam**  
 voldat.hh ..... 203  
**hsubf**  
 scrq.hh ..... 163

**hsubfs**  
 scrq.hh ..... 163  
 scrtch.hh ..... 173  
**hsubg**  
 scrq.hh ..... 163  
**hsubgs**  
 scrq.hh ..... 163  
 scrtch.hh ..... 173  
**htavwt**  
 htsrcm.hh ..... 121  
**htb**  
 htscr.hh ..... 119  
**htb2**  
 htscr1.hh ..... 118  
**htbc2**  
 htscr1.hh ..... 118  
**htbcan**  
 htsrcm.hh ..... 121  
**htbcao**  
 htsrcm.hh ..... 121  
**htbccn**  
 htsrcm.hh ..... 122  
**htbcco**  
 htsrcm.hh ..... 122  
**htbd2**  
 htscr1.hh ..... 118  
**htbnt**  
 htsrcm.hh ..... 122  
**htbnr**  
 htsrcm.hh ..... 122  
**htbnts**  
 htsrcm.hh ..... 122  
**htbvc**  
 htsrcm.hh ..... 122  
**htbvo**  
 htsrcm.hh ..... 122  
**htc2**  
 htscr1.hh ..... 118  
**htcap**  
 cmpdac.hh ..... 91  
**htcf**  
 htrcom.hh ..... 114  
**htcff**  
 scrtch.hh ..... 173  
**htcffn**  
 htsrcm.hh ..... 122

htcffo	
htsrcm.hh.....	122
htcfg	
scrtch.hh.....	173
htcfgn	
htsrcm.hh.....	122
htcfgo	
htsrcm.hh.....	122
htcfp	
scrtch.hh.....	173
htcft	
scrtch.hh.....	173
htcg	
htrcom.hh.....	114
htcfg	
scrtch.hh.....	173
htcgg	
scrtch.hh.....	173
htcgp	
scrtch.hh.....	173
htcgt	
scrtch.hh.....	173
htchfn	
htsrcm.hh.....	122
htchfo	
htsrcm.hh.....	122
htcmp	
htsrcm.hh.....	122
htcmpf	
htsrcm.hh.....	122
htcnon	
htrcom.hh.....	114
htcoef	
htrcom.hh.....	114
htcond	
htrcom.hh.....	114
htdiam	
htrcom.hh.....	114
htdt	
htsrcm.hh.....	122
htdtmn	
htsrcm.hh.....	122
htdtmo	
htsrcm.hh.....	122
htdz	
htscr1.hh.....	118
htdzlm	
htrflb.hh.....	116
hte	
htscr.hh.....	119
htee	
htscr.hh.....	119
htf	
htscr.hh.....	119
htf2	
htscr1.hh.....	118
htfcfr	
htsrcm.hh.....	122
htfftr	
htsrcm.hh.....	122
htfixa	
htsrcm.hh.....	122
htflag	
htscr.hh.....	119
htftrn	
htsrcm.hh.....	122
htftro	
htsrcm.hh.....	122
htgamf	
htrcom.hh.....	115
htgamg	
htrcom.hh.....	115
htgap	
htsrcm.hh.....	123
htgcfg	
scrtch.hh.....	173
htgcgg	
scrtch.hh.....	173
htgcgp	
scrtch.hh.....	173
htgctgt	
scrtch.hh.....	174
htgmr	
htsrcm.hh.....	123
htgom	
htsrcm.hh.....	123
htgskp	
htsrcm.hh.....	123
htgsmf	
htscr.hh.....	119
htgsmg	
htscr.hh.....	119

## INDEX OF VARIABLES IN COMDECKS

htgwff		
scrtch.hh.....	174	
htgwfg		
scrtch.hh.....	174	
htgwfp		
scrtch.hh.....	174	
htgwft		
scrtch.hh.....	174	
hthdmn		
htsrcm.hh.....	123	
hthdmo		
htsrcm.hh.....	123	
hthhf		
htscr.hh.....	119	
hthhg		
htscr.hh.....	119	
hthhp		
htscr.hh.....	119	
hthgt		
htscr.hh.....	119	
hthht		
htscr.hh.....	119	
htimeo		
htsrcm.hh.....	123	
htiscr		
htsrcm.hh.....	123	
htivfc		
htsrcm.hh.....	123	
htivfo		
htsrcm.hh.....	123	
htivrc		
htsrcm.hh.....	123	
htivro		
htsrcm.hh.....	123	
htlen		
htrcom.hh .....	115	
htlenc		
htrcom.hh .....	115	
htlenz		
htrlb.hh.....	116	
htlncf		
htsrcm.hh.....	123	
htlnfn		
htsrcm.hh.....	123	
htlnfo		
htsrcm.hh.....	124	
htlnrn		
htsrcm.hh.....	124	
htlnro		
htsrcm.hh.....	124	
htlvwt		
htsrcm.hh.....	124	
htmod		
htsrcm.hh.....	124	
htnaxl		
htsrcm.hh.....	124	
htnmpt		
htsrcm.hh.....	124	
htnusr		
htsrcm.hh.....	124	
htopt		
htsrcm.hh.....	124	
htopta		
htrcom.hh .....	115	
htpovd		
htsrcm.hh.....	124	
htpown		
htsrcm.hh.....	124	
htpowo		
htsrcm.hh.....	124	
htpws		
htscr2.hh.....	118	
htqof		
htrcom.hh .....	115	
htqog		
htrcom.hh .....	115	
htqosf		
htscr.hh.....	119	
htqosg		
htscr.hh.....	119	
htqost		
htscr.hh.....	119	
htqot		
htrcom.hh .....	115	
htradn		
htsrcm.hh.....	124	
htrado		
htsrcm.hh.....	124	
htrcom common block		113
htrcom.hh comdeck.....	113	
htrcomc common block		113
htrcomc.....	113	
htrlb.hh comdeck		116
htrlb.....	116	

htrlbc.hh comdeck .....	116
htrlfg	
htsrcm.hh.....	124
htrfnn	
htsrcm.hh.....	124
htrfno	
htsrcm.hh.....	124
htrfon	
htsrcm.hh.....	124
htrfoo	
htsrcm.hh.....	124
htrfpt	
htsrcm.hh.....	124
htrgnn	
htsrcm.hh.....	124
htrgno	
htsrcm.hh.....	124
htrgon	
htsrcm.hh.....	124
htrgoo	
htsrcm.hh.....	125
htrnrrn	
htsrcm.hh.....	125
htrnro	
htsrcm.hh.....	125
htrnsn	
htsrcm.hh.....	125
htrnso	
htsrcm.hh.....	125
htrvwt	
htsrcm.hh.....	125
htsa	
htrcm.hh .....	115
htscr.hh comdeck .....	119
htscr1.hh comdeck .....	117
htscr2.hh comdeck .....	118
htscr2c.hh comdeck .....	118
htscrp	
htscr.hh.....	119
htsens	
voldat.hh .....	203
htsrc	
htsrcm.hh.....	125
htsrcm.hh comdeck .....	120
htsrcmc.hh comdeck .....	120
htsrfn	
htsrcm.hh.....	125
htsrcfo	
htsrcm.hh.....	125
htsrt	
htsrcm.hh.....	125
htsrwt	
htsrcm.hh.....	125
htstno	
htsrcm.hh.....	125
htstyp	
htsrcm.hh.....	125
htsxrp	
htscr.hh.....	119
httcc	
htscr.hh.....	119
httcc2	
htscr2.hh.....	118
httmp	
htsrcm.hh.....	125
httots	
htsrcm.hh.....	125
htt2	
htscr1.hh.....	118
https2	
htscr1.hh.....	118
htv1	
htscr1.hh.....	118
htv2	
htscr1.hh.....	118
htvatp	
htsrcm.hh.....	125
htvhc	
htscr.hh.....	120
htscr2.hh.....	118
htxft	
htsrcm.hh.....	125
htxit	
htsrcm.hh.....	125
htxr	
cmpdac.hh .....	91
htzhff	
htrcm.hh .....	115
htzhft	
htrcm.hh .....	115
htzhgg	
htrcm.hh .....	115

## INDEX OF VARIABLES IN COMDECKS

htzhgp		macheff.hh.....	140
htrcom.hh .....	115	i4wxyz	
htzght		macheff.hh.....	140
htrcom.hh .....	115	i5wxyz	
htzht		machof.hh .....	142
htrcom.hh .....	115	i6wxyz	
hvmix		machof.hh .....	142
voldat.hh .....	203	ia	
hyanpr		fast.hh.....	109
voldat.hh .....	203	ialpof	
hyarf		radhtcc.hh.....	149
voldat.hh .....	203	iand	
hyarg		machaf.hh.....	140
voldat.hh .....	203	iareof	
hyaruf		radhtcc.hh.....	149
voldat.hh .....	203	ias	
hyarug		radhtcc.hh.....	149
voldat.hh .....	203	iawxyz	
hydltf		machlf.hh .....	141
scrtch.hh.....	174	ibundl	
hydltg		htrcom.hh .....	115
scrtch.hh.....	174	idin	
hydltp		rrkinc.hh.....	162
scrtch.hh.....	174	idrfl	
hydltt		rflhtc.hh.....	151
scrtch.hh.....	174	ieor	
hydrodynamic systems control	.....	macheff.hh.....	140
hydronearlyimplicit3		iextr2	
tstpct.hh.....	195	contrl.hh.....	102
hydrotmeintgrtionsel5		iextra	
tstpct.hh.....	195	contrl.hh.....	102
hydxc		iglrfl	
voldat.hh .....	203	htrflb.hh.....	116
hydyc		ihh1	
voldat.hh .....	204	scrtch.hh.....	174
hydzc		ihh2	
voldat.hh .....	204	scrtch.hh.....	174
hyposv		ihh3	
voldat.hh .....	204	scrtch.hh.....	174
<b>I</b>		ihld1	
i1wxyz		scrq.hh.....	163
machaf.hh.....	140	scrtch.hh.....	174
i2wxyz		ihld10	
machaf.hh.....	140	scrq.hh.....	164
i3wxyz		scrtch.hh.....	174

scrq.hh	163	jundat.hh	131
scrtch.hh	174	ijsk5	
ihld2a		jundat.hh	131
scrq.hh	164	ijsk6	
scrtch.hh	174	jundat.hh	131
ihld3		ijskp	
scrtch.hh	174	jundat.hh	131
ihld4		image	
scrtch.hh	174	plotpc.hh	146
ihld4a		imap	
scrq.hh	164	voldat.hh	204
scrtch.hh	174	imdctl	
ihld5		contrl.hh	102
scrtch.hh	174	imw	
ihld6		htsrcm.hh	125
scrtch.hh	174	incnd	
ihld7a		htrcom.hh	115
scrtch.hh	174	indtol	
ihld8		rflhtc.hh	151
scrtch.hh	174	indzhs	
ihld9		htscr1.hh	118
scrq.hh	164	inhcnt	
scrtch.hh	174	invhtb.hh	127
ihlppr		inhtno	
contrl.hh	102	invhtb.hh	127
ihtptr		initfailflg8	
htsrcm.hh	125	rkinc.hh	152
ij1		initflag0	
jundat.hh	131	convarc.hh	108
ij1nx		initflag6	
jundat.hh	131	rkinc.hh	152
ij1vn		inittrp0	
jundat.hh	131	trpbblk.hh	193
ij2		inittype1	
jundat.hh	131	convarc.hh	108
ij2nx		inpout	
jundat.hh	131	ufiles.hh	197
ij2vn		input	
jundat.hh	131	ufiles.hh	197
ijflg		inscr1	
jundat.hh	131	rflhtc.hh	151
ijsk2		inscr2	
jundat.hh	131	rflhtc.hh	151
ijsk3		insrft	
jundat.hh	131	invhtb.hh	127
ijsk4		integrationflag9	

## INDEX OF VARIABLES IN COMDECKS

contrl.hh .....	102	inxtmn rflhtc.hh.....	152
interactive variables .....	126	ior machof.hh .....	142
interactiveflag5		ip scrtch.hh.....	174
contrl.hh .....	102	ipmtbl pumpblk.hh .....	147
intrac.hh comdeck .....	126	ipmval pumpblk.hh .....	147
intracc.hh comdeck .....	126	ipmvnm pumpblk.hh .....	147
intrcv		ipmvpc pumpblk.hh .....	147
intrac.hh .....	126	ipmvtl pumpblk.hh .....	147
intrla		ipmvtr pumpblk.hh .....	147
intrac.hh .....	126	ipr scrtch.hh.....	174
intrni		ipu2di pumpblk.hh .....	147
intrac.hh .....	126	ipucctr pumpblk.hh .....	147
intrno		ipuhmi pumpblk.hh .....	147
intrac.hh .....	127	ipumtk pumpblk.hh .....	147
intrva		ipurvi pumpblk.hh .....	147
intrac.hh .....	127	ipuspi pumpblk.hh .....	147
invcnt		iputdi pumpblk.hh .....	148
invtbl.hh .....	128	iputmi pumpblk.hh .....	148
inverted heat structure table .....	127	iputrp pumpblk.hh .....	148
inverted junction table .....	127	ireclt ftbcom.hh .....	111
invfnd		ireg scrtch.hh.....	175
voldat.hh .....	204	iregj jundat.hh .....	131
invhkp		irhflx radhtcc.hh.....	149
invhtb.hh .....	127		
invhos			
invhtb.hh .....	127		
invhtb.hh comdeck .....	127		
invhtbc.hh comdeck .....	127		
invhtf			
voldat.hh .....	205		
invjun			
invtbl.hh .....	128		
invofs			
invtbl.hh .....	128		
invskp			
invtbl.hh .....	128		
invtbl.hh comdeck .....	127		
invtblc.hh comdeck .....	127		
invvnno			
invtbl.hh .....	128		
invvnx			
invtbl.hh .....	128		
inxlsr			
rflhtc.hh.....	152		
inxrf1			
htrflb.hh.....	117		

irhoff	
radhtcc.hh	149
irn	
scrtch.hh	175
irnr	
scrtch.hh	175
iroute	
contrl.hh	102
irwt	
htrcom.hh	115
is23	
scrq.hh	164
scrtch.hh	175
is2turbjnjflg29	
jundat.hh	131
is2vel1velflg9	
jundat.hh	131
is3dflg2	
cmpdat.hh	93
is3dvol17	
voldat.hh	205
isabrareachgflg8	
jundat.hh	131
isaccactvflg15	
jundat.hh	131
isaccum5	
voldat.hh	205
isairjunflg30	
jundat.hh	132
isairrpt1	
voldat.hh	205
isansinpflg27	
voldat.hh	205
isathenaopt10	
contrl.hh	102
isbundle16	
voldat.hh	205
isbundle30	
voldat.hh	205
isccflflg1	
jundat.hh	132
ischkvlvclosed4	
cmpdat.hh	93
ischokflg0	
jundat.hh	132
ischoktstflgaccjun6	
jundat.hh	132
iscnfnctcntrl01	
rkinc.hh	152
iscntrlsyschng6	
contrl.hh	102
iscompleterestart4	
contrl.hh	102
iscoordcode11	
htsrem.hh	125
iscoorddir14	
voldat.hh	205
iscurrsaveflg9	
voldat.hh	205
iscurrvollflg7	
voldat.hh	205
isdbgprntflg0	
voldat.hh	205
isdbgprntflg20	
jundat.hh	132
isdonprespvwrk15	
jundat.hh	132
isdrainflag2	
cmpdat.hh	93
isdum3	
cmpdat.hh	93
isdum6	
cmpdat.hh	93
iseccmixflg18	
jundat.hh	132
iseccmixflg19	
jundat.hh	132
iselevchk0	
jundat.hh	132
isepst	
separ.hh	184
isequilflg1	
voldat.hh	205
isexpfri12	
voldat.hh	205
isfissionprdctchn5	
contrl.hh	102
isflowregmnum1823	
voldat.hh	205
isfrmmonflxopt13	
jundat.hh	132
isgasappearance6	

## INDEX OF VARIABLES IN COMDECKS

voldat.hh .....	205	isjetmixflg20	
isgeneratorflag2		jundat.hh .....	132
convarc.hh .....	108	isjetmixflg21	
isgodunovflg22		jundat.hh .....	132
jundat.hh .....	132	isjunfloregnum38	
isgodunovflg23		jundat.hh .....	132
jundat.hh .....	132	islamfricfactor26	
isgodunovflg4		voldat.hh .....	205
voldat.hh .....	205	islevelcrossing	
isgodunovflg5		contrl.hh .....	102
voldat.hh .....	205	islevelmod27	
ish2opackervol30		jundat.hh .....	132
voldat.hh .....	205	islevelmodel0	
ish2opck7		contrl.hh .....	102
voldat.hh .....	205	isleveltrckinp28	
ish2opckstat3		voldat.hh .....	205
voldat.hh .....	205	isliqentrain30	
isheatcondchng1		jundat.hh .....	132
contrl.hh .....	102	isloopcomponent1	
ishft		cmpdat.hh .....	93
machss.hh .....	142	islosscoeffchng10	
ishorzstrat7		htsrcm.hh .....	125
voldat.hh .....	205	islvlflg6	
ishorzvertjunflg26		voldat.hh .....	205
jundat.hh .....	132	ismajoreditenabled1	
ishydrochng0		contrl.hh .....	102
contrl.hh .....	102	ismapinfo05	
ishysteresis2		voldat.hh .....	206
cmpdat.hh .....	93	ismaxcomponents1	
isimplicithttrnsfr6		convarc.hh .....	108
isinithsemod24		ismetalapprnce24	
jundat.hh .....	132	voldat.hh .....	206
isinithsemod25		ismetalapprnce25	
jundat.hh .....	132	voldat.hh .....	206
isinitllyopen4		isminoreditenabled2	
cmpdat.hh .....	93	contrl.hh .....	102
isinpflg7		ismomfluxoff3	
jundat.hh .....	132	invtbl.hh .....	128
isinput3		ismomfluxoffflg14	
voldat.hh .....	205	jundat.hh .....	132
isinpvertstrat9		ismulti3	
voldat.hh .....	205	cmpdat.hh .....	93
isjetjunflg25		ismwrinnersurface13	
jundat.hh .....	132	htsrcm.hh .....	125
isjetmixflg19		ismwrsetinput14	
jundat.hh .....	132	htsrcm.hh .....	125

isnegvelflg5		
cmpdat.hh	.....	93
isnegvoid12		
voldat.hh	.....	206
isnew2phaserr15		
voldat.hh	.....	206
isnewcnvgerr16		
voldat.hh	.....	206
isnewdrvqtyerr18		
voldat.hh	.....	206
isnewliqerr14		
voldat.hh	.....	206
isnewlrgstmaserr12		
voldat.hh	.....	206
isnewmasserr9		
voldat.hh	.....	206
isnewnegpres8		
voldat.hh	.....	206
isnewqualovrn11		
voldat.hh	.....	206
isnewsonvelerr17		
voldat.hh	.....	206
isnewvaperr13		
voldat.hh	.....	206
isnewxtraperr10		
voldat.hh	.....	206
isnochokflg4		
jundat.hh	.....	132
isnolosscoeffabrun29		
jundat.hh	.....	132
isnoncond11		
voldat.hh	.....	206
ispccflflg2		
jundat.hh	.....	132
isold2phaserr26		
voldat.hh	.....	206
isoldcnvgerr27		
voldat.hh	.....	206
isolddrvqtyerr29		
voldat.hh	.....	206
isoldliqerr25		
voldat.hh	.....	206
isoldlrgstmaserr23		
voldat.hh	.....	207
isoldmasserr20		
voldat.hh	.....	207
isoldnegpres19		
voldat.hh	.....	207
isoldqualovrn22		
voldat.hh	.....	207
isoldsonvelerr28		
voldat.hh	.....	207
isoldtimchokflg5		
jundat.hh	.....	132
isoldvaperr24		
voldat.hh	.....	207
isoldxtraperr21		
voldat.hh	.....	207
isopt1latch3		
cmpdat.hh	.....	93
isopt2latch6		
cmpdat.hh	.....	93
isoutletflag1		
invtbl.hh	.....	128
isplasticstrain11		
htsrcm.hh	.....	125
isplotenabled3		
contrl.hh	.....	102
isplotsqzflg12		
contrl.hh	.....	102
ispltrcrdchng3		
contrl.hh	.....	102
ispresschngrpt2		
voldat.hh	.....	207
ispresscnrl3		
cmpdat.hh	.....	93
isprevsaveflg10		
voldat.hh	.....	207
isprevvollvlflg8		
voldat.hh	.....	207
isprntaccum0		
contrl.hh	.....	103
isprntbrntrn1		
contrl.hh	.....	103
isprntccfl2		
contrl.hh	.....	103
isprntchfc13		
contrl.hh	.....	103
isprntconden4		
contrl.hh	.....	103
isprntcontrol21		
contrl.hh	.....	103

## INDEX OF VARIABLES IN COMDECKS

isprntdittus5		
contrl.hh .....	103	
isprntrqfinl6		
contrl.hh .....	103	
isprntfsnprdtr20		
contrl.hh .....	103	
isprntfwdrag7		
contrl.hh .....	103	
isprntheadstr17		
contrl.hh .....	103	
isprnht2tdp9		
contrl.hh .....	103	
isprnhtfinl12		
contrl.hh .....	103	
isprnhtrc113		
contrl.hh .....	103	
isprnthydro15		
contrl.hh .....	103	
isprntinput22		
contrl.hh .....	103	
isprntistate17		
contrl.hh .....	103	
isprntjchoke18		
contrl.hh .....	103	
isprntjprop19		
contrl.hh .....	103	
isprntjunction16		
contrl.hh .....	103	
isprntmiedit23		
contrl.hh .....	103	
isprntnoncnd20		
contrl.hh .....	103	
isprnphantj21		
contrl.hh .....	103	
isprnphantv22		
contrl.hh .....	103	
isprntpimplt23		
contrl.hh .....	103	
isprntpintfc24		
contrl.hh .....	103	
isprntpower14		
contrl.hh .....	103	
isprntprednb25		
contrl.hh .....	103	
isprnpreseq26		
contrl.hh .....	104	
isprntpstdnb27		
contrl.hh .....	104	
isprntqfmove28		
contrl.hh .....	104	
isprntradht18		
contrl.hh .....	104	
isprntreflood19		
contrl.hh .....	104	
isprntsimplt29		
contrl.hh .....	104	
isprntstacc0		
contrl.hh .....	104	
isprntstate1		
contrl.hh .....	104	
isprntstate2		
contrl.hh .....	104	
isprntsuboil3		
contrl.hh .....	104	
isprntsysisr4		
contrl.hh .....	104	
isprnttrip13		
contrl.hh .....	104	
isprnttstate6		
contrl.hh .....	104	
isprntvalve7		
contrl.hh .....	104	
isprntvexplt8		
contrl.hh .....	104	
isprntvfinl9		
contrl.hh .....	104	
isprntvimplt10		
contrl.hh .....	104	
isprntvvela11		
contrl.hh .....	104	
isprntvolume15		
contrl.hh .....	104	
isprntvvolvel12		
contrl.hh .....	104	
isptr		
scrtch.hh .....	175	
ispump6		
voldat.hh .....	207	
ispumpstopflg2		
cmpdat.hh .....	93	
ispumpstoppedflg3		
cmpdat.hh .....	93	

isradiationchng4	
contrl.hh	104
isreflood29	
voldat.hh	207
isreliefvlv4	
cmpdat.hh	93
isrestartenabled0	
contrl.hh	104
isreversecoordflg0	
invtbl.hh	128
isrevfrmvolconflg2	
jundat.hh	133
isrevtovolconflg3	
jundat.hh	133
isrkncnchg2	
contrl.hh	104
isrupture9	
htsrcm.hh	126
isscdapopt11	
contrl.hh	104
issepflg10	
jundat.hh	133
issepflg22	
jundat.hh	133
issepflg23	
jundat.hh	133
issepflg24	
jundat.hh	133
issshftconnectedflg4	
cmpdat.hh	93
issphericalaccum8	
cmpdat.hh	93
isstratflg16	
jundat.hh	133
isstratflwflg11	
jundat.hh	133
isstratinpdat17	
jundat.hh	133
isstratinpdat18	
jundat.hh	133
isstretch8	
voldat.hh	207
isstrtcjhunflg17	
jundat.hh	133
istdpjunflg1	
jundat.hh	133
istdpvol0	
voldat.hh	207
isthermfrnt2	
voldat.hh	207
istoflag2	
invtbl.hh	128
istomomflxoffopt12	
jundat.hh	133
istripvlv3	
cmpdat.hh	93
istripvlv4	
cmpdat.hh	93
istsppac10	
tstpct.hh	195
istsppac11	
tstpct.hh	195
isturbtypeone3	
cmpdat.hh	93
isturbtypetwo4	
cmpdat.hh	93
istwostepflag7	
contrl.hh	104
isupdwnjunflg27	
jundat.hh	133
isvapcontphasejun28	
jundat.hh	133
isvapdis4	
voldat.hh	207
isvertstrat10	
voldat.hh	207
isvertstrat11	
voldat.hh	207
isvlvcycled2	
cmpdat.hh	93
isvlvflg28	
jundat.hh	133
iswallfricinp13	
voldat.hh	207
iswatpackflg21	
jundat.hh	133
iswatpackjunflg16	
jundat.hh	133
isyskp	
sysdate.hh	191
it3bp	
stcom.hh	189

## INDEX OF VARIABLES IN COMDECKS

it3p0		ivsk9	
stcom.hh.....	189	voldat.hh .....	208
it4bp		ivskp	
stcom.hh.....	189	voldat.hh .....	208
it5bp		ix	
stcom.hh.....	189	rrkinc.hh.....	162
itemof		ixcofp	
radhtcc.hh.....	149	lpdat.hh .....	137
itg		trnhlp.hh.....	192
radhtcc.hh.....	149	ixip	
itrchf		lpdat.hh .....	137
flood.hh .....	110	trnhlp.hh.....	192
itt		ixipr	
scrq.hh.....	164	lpdat.hh .....	137
iv		trnhlp.hh.....	192
htrcom.hh .....	115	ixipx	
iv1		lpdat.hh .....	137
htrcom.hh .....	115	ixirn	
iv2		lpdat.hh .....	137
htrcom.hh .....	115	trnhlp.hh.....	192
invert		ixirnr	
scrtch.hh.....	175	lpdat.hh .....	137
ivewof		trnhlp.hh.....	192
radhtcc.hh.....	149	ixirnx	
ivindx		lpdat.hh .....	137
htrcom.hh .....	115	ixivrn	
ivrn		lpdat.hh .....	137
scrtch.hh.....	175	ixpv	
ivsk1		trnhlp.hh.....	192
voldat.hh .....	207	ixsopr	
ivsk10		lpdat.hh .....	137
voldat.hh .....	208	ixw	
ivsk2		lpdat.hh .....	137
voldat.hh .....	207	trnhlp.hh.....	192
ivsk3		<b>J</b>	
voldat.hh .....	207		
ivsk4		ja	
voldat.hh .....	207	scrtch.hh.....	175
ivsk5		jb	
voldat.hh .....	207	scrtch.hh.....	175
ivsk6		jbinfo	
voldat.hh .....	208	ufiles.hh.....	197
ivsk7		jc	
voldat.hh .....	208	jundat.hh .....	133
ivsk8		jcattn	
voldat.hh .....	208	jundat.hh .....	133

jcato  
     jundat.hh ..... 133  
 jcex  
     jundat.hh ..... 133  
 jcfnfd  
     jundat.hh ..... 133  
 jcnx1  
     jundat.hh ..... 133  
 jcnx2  
     jundat.hh ..... 133  
 jcnx3  
     jundat.hh ..... 134  
 jcnxd  
     jundat.hh ..... 134  
 jcnxs  
     jundat.hh ..... 134  
 jdissc  
     jundat.hh ..... 134  
 jdissn  
     jundat.hh ..... 134  
 jdistp  
     jundat.hh ..... 134  
 jetdf  
     scrtch.hh ..... 175  
 jetdg  
     scrtch.hh ..... 175  
 jetjdf  
     scrtch.hh ..... 175  
 jetjdg  
     scrtch.hh ..... 175  
 jetjsf  
     scrtch.hh ..... 175  
 jetjsg  
     scrtch.hh ..... 175  
 jetmixer  
     cmpalf.hh ..... 89  
 jetsf  
     scrtch.hh ..... 175  
 jetsg  
     scrtch.hh ..... 175  
 jhld1  
     scrtch.hh ..... 175  
 jhld2  
     scrtch.hh ..... 175  
 jhld3  
     scrtch.hh ..... 175

jhld4  
     scrtch.hh ..... 175  
 jhld5  
     scrtch.hh ..... 175  
 jhld6  
     scrtch.hh ..... 175  
 jhld7  
     scrtch.hh ..... 175  
 jjet  
     voldat.hh ..... 208  
 jlr  
     radhtcc.hh ..... 149  
 jprop ..... 43  
 jrh  
     radhtcc.hh ..... 149  
 jtcond  
     scrtch.hh ..... 175  
 jtcons  
     scrtch.hh ..... 175  
 jtdjdf  
     scrtch.hh ..... 175  
 jtdjdg  
     scrtch.hh ..... 176  
 jtdjsf  
     scrtch.hh ..... 176  
 jtdjsg  
     scrtch.hh ..... 176  
 junctions ..... 128  
 jundat.hh comdeck ..... 128  
 jundatc.hh comdeck ..... 128  
 junftl  
     jundat.hh ..... 134  
 junno  
     jundat.hh ..... 134  
**K**  
 k3all common block ..... 100  
 k3point common block ..... 100  
 kapa  
     scrq.hh ..... 164  
     scrtch.hh ..... 176  
 kapaf  
     scrq.hh ..... 164  
     scrtch.hh ..... 176  
 kapaff  
     scrq.hh ..... 164

## INDEX OF VARIABLES IN COMDECKS

scrtch.hh .....	176
kapafs	
scrq.hh .....	164
scrtch.hh .....	176
kapag	
scrq.hh .....	164
scrtch.hh .....	176
kapagg	
scrq.hh .....	164
scrtch.hh .....	176
kapags	
scrq.hh .....	164
scrtch.hh .....	176
 <b>L</b>	
l24skp	
sscntr.hh .....	184
lasdes	
ftbcom.hh .....	111
latchcode1	
trpbblk.hh .....	193
lcntrl.hh comdeck .....	135
lcntrlc.hh comdeck .....	135
ld	
rrkinc.hh .....	162
lde	
rrkinc.hh .....	162
level model .....	136, 139
levska	
levtbl.hh .....	136
levskb	
levtbl.hh .....	136
levskd	
levtbl.hh .....	136
levskh	
levtbl.hh .....	136
levskl	
levtbl.hh .....	136
levskn	
levtbl.hh .....	136
levsko	
levtbl.hh .....	136
levstk	
lpdat.hh .....	137
levtbl.hh comdeck .....	136
levtblc.hh comdeck .....	136

lflag	
scrq.hh .....	164
scrtch.hh .....	176
lflag2	
scrq.hh .....	164
scrtch.hh .....	176
lfsiz	
fast.hh .....	109
lgg1	
scrtch.hh .....	176
lhtrfl	
htrflb.hh .....	117
lhtsol	
trnhlp.hh .....	192
lic	
lpdat.hh .....	137
licn	
lpdat.hh .....	137
lij	
lpdat.hh .....	137
lijn	
lpdat.hh .....	137
lim	
rrkinc.hh .....	162
link1	
ftbcom.hh .....	111
list vector pointers .....	138
liv	
lpdat.hh .....	137
livn	
lpdat.hh .....	137
livnn	
lpdat.hh .....	137
llvect	
lpdat.hh .....	137
lnelv	
cmpdac.hh .....	91
lngylv	
cmpdtv.hh .....	96
lnlen	
cmpdac.hh .....	91
lnoncn	
lpdat.hh .....	137
lnonnmf	
lpdat.hh .....	137
locf	

machlf.hh .....	140
locf4	
machlf.hh .....	140
locfi	
machlf.hh .....	140
locfi4	
machlf.hh .....	140
loop control array.....	135
lpackr	
lpdat.hh .....	137
lpdat.hh comdeck .....	136, 138, 139
lpdatec.hh comdeck .....	136
lpskp	
lpdat.hh .....	137
lrhof	
scrtch.hh.....	176
lrhog	
scrtch.hh.....	176
lstcom	
stcom.hh.....	190
lsuces	
lpdat.hh .....	138
lsysnm	
lpdat.hh .....	138
ltest	
scrtch.hh.....	176
ltest2	
scrtch.hh.....	176
ltest3	
scrtch.hh.....	176
ltestt	
scrtch.hh.....	176
lv3d	
lvectr.hh .....	139
lvabrp	
lvectr.hh .....	139
lvaccm	
lvectr.hh .....	139
lvajun	
lvectr.hh .....	139
lvavol	
lvectr.hh .....	139
lvectrc.hh comdeck .....	138
lvel common block.....	100
lvhzfl	
lvectr.hh .....	139
lvjtmx	
lvectr.hh .....	139
lvjusr	
lvectr.hh .....	139
lvnofr	
lvectr.hh .....	139
lvprz	
lvectr.hh .....	139
lpvptr	
lvectr.hh .....	139
lpump	
lvectr.hh .....	139
lrvvol	
lvectr.hh .....	139
lvscr	
lvectr.hh .....	139
lvsepr	
lvectr.hh .....	139
lvturb	
lvectr.hh .....	139
lvtvol	
lvectr.hh .....	139
lvvalv	
lvectr.hh .....	139
lvwifr	
lvectr.hh .....	139
lx	
rrkinc.hh.....	162
lxa	
rrkinc.hh.....	162
lxid	
rrkinc.hh.....	162
lxl	
rrkinc.hh.....	162
lxp	
rrkinc.hh.....	162
<b>M</b>	
m1a	
rmamacac.hh.....	161
m1af	
rmamacac.hh.....	161
m1b	
rmamacac.hh.....	161
m1c	
rmamacac.hh.....	161

## INDEX OF VARIABLES IN COMDECKS

m2a	tstpct.hh.....	195
rmacac.hh.....		161
m2af		
rmacac.hh.....		161
m2b		
rmacac.hh.....		161
m2c		
rmacac.hh.....		161
m3a		
rmacac.hh.....		161
m3af		
rmacac.hh.....		161
m3b		
rmacac.hh.....		161
m3c		
rmacac.hh.....		161
m4a		
rmacac.hh.....		162
m4af		
rmacac.hh.....		162
m4b		
rmacac.hh.....		162
m4c		
rmacac.hh.....		162
m5a		
rmacac.hh.....		162
m5af		
rmacac.hh.....		162
m5b		
rmacac.hh.....		162
m5c		
rmacac.hh.....		162
machaf.hh comdeck		140
machas.hh comdeck		140
machef.hh comdeck		140
maches.hh comdeck		140
machlf.hh comdeck		141
machls.hh comdeck		140
machof.hh comdeck		142
machos.hh comdeck		142
machss.hh comdeck		142
majeditevrytmestp12		
tstpct.hh.....		195
makmap common block		142
makmap.hh comdeck		142
masserrtimepcntr0		
material data for heat structures		161
maximumflag3		
convarc.hh.....		108
maxlcm		
maxmem.hh.....		143
maxmem common block		143
maxmem.hh comdeck		143
maxscm		
maxmem.hh.....		143
maxtht		
cmpdty.hh .....		96
maxz		
ftbcom.hh .....		111
mcme17		
cons.hh .....		101
mesh4		
rflhtc.hh.....		152
meshptnumber08		
htsrem.hh.....		126
meshy		
rflhtc.hh.....		152
mflowj		
jundat.hh .....		134
mflwjo		
jundat.hh .....		134
micode		
miedtc.hh.....		143
miconv		
miedtc.hh.....		143
miedtc.hh comdeck		143
mietab		
miedtc.hh.....		143
mietaf		
miedtc.hh.....		143
mihold		
miedtc.hh.....		143
milabl		
miedtc.hh.....		143
minor edits		143
mintht		
cmpdty.hh .....		96
minz		
ftbcom.hh .....		111
mipck		
miedtc.hh.....		144

mmeshz	
rflhtc.hh	152
mnreditevrytmestp13	
tstpct.hh	195
mode	
htrcom.hh	115
molecular weights	144
moment	
cmpdtv.hh	96
motortorqreferalflg12	
cmpdat.hh	93
mtbl	
mtbls.hh	144
mtblc.hh comdeck	144
mtblen	
mtbls.hh	144
mtblr	
mtbls.hh	144
mtbls.hh comdeck	144
mtbnum	
mtbls.hh	144
mtbptr	
mtbls.hh	144
mpljun	
cmpalf.hh	89
mtype	
trnhlp.hh	192
multid	
cmpalf.hh	89
mxnfcd.hh comdeck	144
mxnfld	
mxnfcd.hh	145
<b>N</b>	
nany	
contrl.hh	104
ncase	
contrl.hh	104
ncmps	
cmpdat.hh	93
ncomp	
rcompc.hh	151
ncoms	
comctl.hh	100
ncount	
contrl.hh	105
ncrosk	
scrtch.hh	176
ncrosl	
scrtch.hh	176
ntalf	
cmpdat.hh	93
ntble	
cmpdat.hh	94
ntblt	
cmpdat.hh	94
ntbtn	
cmpdat.hh	94
ntbttx	
cmpdat.hh	94
ntdpv	
cmpdat.hh	94
ntpc	
cmpdat.hh	94
nttrp	
cmpdat.hh	94
nttrrx	
cmpdat.hh	94
ntctype	
statec	188
ncvtbl	
cmpdtv.hh	96
ndsk2	
ftbcom.hh	111
ndxstd	
stcblk.hh	189
newrst	
comctl.hh	100
newtimenooutflowflg2	
cmpdat.hh	94
newtimevolactveflg4	
cmpdat.hh	94
nexdes	
ftbcom.hh	111
nfeeds	
ssiblk.hh	185
nfiles	
comctl.hh	100
nflsch	
ufilef.hh	197
nfluid	
stcblk.hh	189

## INDEX OF VARIABLES IN COMDECKS

ngtbls	statec .....	188																																																												
gentblc.hh.....																																																														
nhtga	noncn																																																													
htrflb.hh.....	statec .....	188																																																												
nhtma	nondum																																																													
htrflb.hh.....	statec .....	188																																																												
nhtstr	nonhe																																																													
htsrcm.hh.....	statec .....	188																																																												
nindx	nonhy																																																													
usrvar.hh .....	statec .....	188																																																												
nix	nonkr																																																													
scrtch.hh.....	statec .....	188																																																												
njc	nonni																																																													
cmpdat.hh.....	statec .....	188																																																												
njcn	nonxe																																																													
cmpdat.hh.....	statec .....	188																																																												
njco	nonzeroinnerradiusflag6																																																													
cmpdat.hh.....	cmpdat.hh.....	94																																																												
njuns	np																																																													
jundat.hh .....	stcom.hh.....	190																																																												
nloops	npaccm common block .....	145																																																												
lpdat.hh .....	npacom common block .....	145																																																												
nmapp	npacom.hh comdeck .....	145																																																												
scrtch.hh.....	npamsg																																																													
nmbrtblcoord0204	npacom.hh.....	145																																																												
rkinc.hh .....	nprpnt																																																													
nmechk	stcom.hh.....	190	contrl.hh .....	npumps		nmiet	ssiblk.hh.....	185	miedtc.hh.....	nqfmov		nmtbls	rflhtc.hh.....	152	mtbls.hh.....	nrepet		nmzht	contrl.hh .....	105	htrflb.hh.....	nrfht		nnz	htrflb.hh.....	117	lpdat.hh .....	nrh		trnhlp.hh.....	radhtcc.hh.....	149	nnz2	nrhskp		lpdat.hh .....	radhtcc.hh.....	150	nofils	nrset		ftbcom.hh .....	radhtcc.hh.....	150	nolink	nrsskp		ftbcom.hh .....	radhtcc.hh.....	150	nonair	nscra1		statec .....	htrflb.hh.....	117	nonar	nscra2			htrflb.hh.....	117
stcom.hh.....	190																																																													
contrl.hh .....	npumps																																																													
nmiet	ssiblk.hh.....	185	miedtc.hh.....	nqfmov		nmtbls	rflhtc.hh.....	152	mtbls.hh.....	nrepet		nmzht	contrl.hh .....	105	htrflb.hh.....	nrfht		nnz	htrflb.hh.....	117	lpdat.hh .....	nrh		trnhlp.hh.....	radhtcc.hh.....	149	nnz2	nrhskp		lpdat.hh .....	radhtcc.hh.....	150	nofils	nrset		ftbcom.hh .....	radhtcc.hh.....	150	nolink	nrsskp		ftbcom.hh .....	radhtcc.hh.....	150	nonair	nscra1		statec .....	htrflb.hh.....	117	nonar	nscra2			htrflb.hh.....	117						
ssiblk.hh.....	185																																																													
miedtc.hh.....	nqfmov																																																													
nmtbls	rflhtc.hh.....	152	mtbls.hh.....	nrepet		nmzht	contrl.hh .....	105	htrflb.hh.....	nrfht		nnz	htrflb.hh.....	117	lpdat.hh .....	nrh		trnhlp.hh.....	radhtcc.hh.....	149	nnz2	nrhskp		lpdat.hh .....	radhtcc.hh.....	150	nofils	nrset		ftbcom.hh .....	radhtcc.hh.....	150	nolink	nrsskp		ftbcom.hh .....	radhtcc.hh.....	150	nonair	nscra1		statec .....	htrflb.hh.....	117	nonar	nscra2			htrflb.hh.....	117												
rflhtc.hh.....	152																																																													
mtbls.hh.....	nrepet																																																													
nmzht	contrl.hh .....	105	htrflb.hh.....	nrfht		nnz	htrflb.hh.....	117	lpdat.hh .....	nrh		trnhlp.hh.....	radhtcc.hh.....	149	nnz2	nrhskp		lpdat.hh .....	radhtcc.hh.....	150	nofils	nrset		ftbcom.hh .....	radhtcc.hh.....	150	nolink	nrsskp		ftbcom.hh .....	radhtcc.hh.....	150	nonair	nscra1		statec .....	htrflb.hh.....	117	nonar	nscra2			htrflb.hh.....	117																		
contrl.hh .....	105																																																													
htrflb.hh.....	nrfht																																																													
nnz	htrflb.hh.....	117	lpdat.hh .....	nrh		trnhlp.hh.....	radhtcc.hh.....	149	nnz2	nrhskp		lpdat.hh .....	radhtcc.hh.....	150	nofils	nrset		ftbcom.hh .....	radhtcc.hh.....	150	nolink	nrsskp		ftbcom.hh .....	radhtcc.hh.....	150	nonair	nscra1		statec .....	htrflb.hh.....	117	nonar	nscra2			htrflb.hh.....	117																								
htrflb.hh.....	117																																																													
lpdat.hh .....	nrh																																																													
trnhlp.hh.....	radhtcc.hh.....	149	nnz2	nrhskp		lpdat.hh .....	radhtcc.hh.....	150	nofils	nrset		ftbcom.hh .....	radhtcc.hh.....	150	nolink	nrsskp		ftbcom.hh .....	radhtcc.hh.....	150	nonair	nscra1		statec .....	htrflb.hh.....	117	nonar	nscra2			htrflb.hh.....	117																														
radhtcc.hh.....	149																																																													
nnz2	nrhskp																																																													
lpdat.hh .....	radhtcc.hh.....	150	nofils	nrset		ftbcom.hh .....	radhtcc.hh.....	150	nolink	nrsskp		ftbcom.hh .....	radhtcc.hh.....	150	nonair	nscra1		statec .....	htrflb.hh.....	117	nonar	nscra2			htrflb.hh.....	117																																				
radhtcc.hh.....	150																																																													
nofils	nrset																																																													
ftbcom.hh .....	radhtcc.hh.....	150	nolink	nrsskp		ftbcom.hh .....	radhtcc.hh.....	150	nonair	nscra1		statec .....	htrflb.hh.....	117	nonar	nscra2			htrflb.hh.....	117																																										
radhtcc.hh.....	150																																																													
nolink	nrsskp																																																													
ftbcom.hh .....	radhtcc.hh.....	150	nonair	nscra1		statec .....	htrflb.hh.....	117	nonar	nscra2			htrflb.hh.....	117																																																
radhtcc.hh.....	150																																																													
nonair	nscra1																																																													
statec .....	htrflb.hh.....	117	nonar	nscra2			htrflb.hh.....	117																																																						
htrflb.hh.....	117																																																													
nonar	nscra2																																																													
	htrflb.hh.....	117																																																												
htrflb.hh.....	117																																																													

nsep	
separ.hh	..... 184
nsp	
stcom.hh	..... 190
nst	
stcom.hh	..... 190
nstctl	
ssiblk.hh	..... 185
nstsp	
contrl.hh	..... 105
nt	
stcom.hh	..... 190
ntabl	
usrvar.hh	..... 198
ntabla	
usrvar.hh	..... 198
ntdpvs	
tdpptr.hh	..... 191
ntlskp	
trpblk.hh	..... 193
ntrcv1	
trpblk.hh	..... 193
ntrcv2	
trpblk.hh	..... 193
ntrnv1	
trpblk.hh	..... 193
ntrnv2	
trpblk.hh	..... 193
ntrpc1	
trpblk.hh	..... 193
ntrpc2	
trpblk.hh	..... 193
ntrpff	
trpblk.hh	..... 193
ntrpn1	
trpblk.hh	..... 193
ntrpno	
trpblk.hh	..... 193
ntrpnv	
trpblk.hh	..... 193
ntrpof	
trpblk.hh	..... 193
ntrpop	
trpblk.hh	..... 193
ntrps1	
trpblk.hh	..... 193
ntrps2	
trpblk.hh	..... 193
ntrtr1	
trpblk.hh	..... 193
ntrtr2	
trpblk.hh	..... 193
ntvskp	
trpblk.hh	..... 194
nusvar	
usrvar.hh	..... 198
nusys	
sysdatc.hh	..... 191
nvalhi	
scrtch.hh	..... 176
nvalhx	
scrtch.hh	..... 176
nvc	
cmpdat.hh	..... 94
nvcn	
cmpdat.hh	..... 94
nvco	
cmpdat.hh	..... 94
nvols	
voldat.hh	..... 208
nvr	
lpdat.hh	..... 138
trnhlp.hh	..... 192
nvrp	
lpdat.hh	..... 138
nwq	
npacom.hh	..... 145
<b>O</b>	
octantnumber2431	
cmpdat.hh	..... 94
oldtimenooutflowflg5	
cmpdat.hh	..... 94
oldtimevolactveflg7	
cmpdat.hh	..... 94
omega	
cmpdtv.hh	..... 96
omegao	
cmpdtv.hh	..... 96
onrefl	
flood.hh	..... 110
opcode2431	

## INDEX OF VARIABLES IN COMDECKS

trblk.hh	194	pgcall	
opntrp		htsrcm.hh	126
cmpdtv.hh	96	pgflag	
optnumortblnum06		htrcom.hh	115
htsrcm.hh	126	pgopta	
output		htrcom.hh	115
ufiles.hh	197	pgopti	
oxti		htsrcm.hh	126
htsrcm.hh	126	pi	
oxtio		cons.hh	100
htsrcm.hh	126	piovr2	
oxto		cons.hh	100
htsrcm.hh	126	piovr4	
oxtoo		cons.hh	100
htsrcm.hh	126	pipe	
<b>P</b>		cmpalf.hh	90
p		pithier	
voldat.hh	208	htrcom.hh	116
paa		pk	
scrtch.hh	176	scrtch.hh	177
pack		pl	
scrtch.hh	176	scrtch.hh	177
packer	43, 44	plotfl	
pageno		ufiles.hh	197
contrl.hh	105	plotpc common block	146
pbb		plotpc.hh comdeck	146
scrtch.hh	176	pltrcdevrytmestp14	
pbellw		tstpct.hh	195
cmpdtv.hh	96	pmax	
pcrit		stcom.hh	190
stcom.hh	190	pmhig	
pcv		scrtch.hh	177
cmpdtv.hh	96	pmin	
pecln		stcom.hh	190
htsrcm.hh	126	pmpbsp	
peclo		pumpblk.hh	148
htsrcm.hh	126	pmpfsp	
pecltv		pumpblk.hh	148
voldat.hh	208	pmpint	
pfinrg		pumpblk.hh	148
scrtch.hh	177	pmpmt	
pfluid		pumpblk.hh	148
gapvar.hh	112	pmpnr	
pgas		pumpblk.hh	148
gapvar.hh	112	pmpold	
		pumpblk.hh	148

pmpph		
	scrtch.hh	177
pmp rfl		
	pumpblk.hh	148
pmp rhd		
	pumpblk.hh	148
pmp rho		
	pumpblk.hh	148
pmp rmt		
	pumpblk.hh	148
pmp rsp		
	pumpblk.hh	148
pmp rtk		
	pumpblk.hh	148
pmp spr		
	pumpblk.hh	148
pmp stm		
	pumpblk.hh	148
pmp tbl		
	pumpblk.hh	148
pmp tf1		
	pumpblk.hh	148
pmp tf2		
	pumpblk.hh	148
pmp tf3		
	pumpblk.hh	148
pmp tfy		
	pumpblk.hh	148
pmp thd		
	pumpblk.hh	149
pmp trp		
	pumpblk.hh	149
pmp ttk		
	pumpblk.hh	149
pmp vtl		
	pumpblk.hh	149
po		
	voldat.hh	208
pps		
	voldat.hh	208
ppso		
	voldat.hh	208
pr		
	htrcom.hh	116
pres		
	scrq.hh	164
	scrtch.hh	177
	preseq	43
	pressurizer component	146
	print	
	contrl.hh	105
	prizer	
	cmpalf.hh	90
	problemopt61	
	contrl.hh	105
	problemtype05	
	contrl.hh	105
	prop	
	statec	188
	tmsrcm.hh	191
	przdat.hh comdeck	146
	przdatc.hh comdeck	146
	prjnx	
	przdat.hh	146
	przlln	
	przdat.hh	146
	przlvl	
	przdat.hh	146
	prznds	
	przdat.hh	146
	przvnx	
	przdat.hh	146
ps		
	scrq.hh	164
	scrtch.hh	177
	psat	
	scrq.hh	164
	scrtch.hh	177
	pshig	
	scrtch.hh	177
	psld	
	scrtch.hh	177
	pslope	
	scrtch.hh	177
	psmf	
	scrtch.hh	177
	psmg	
	scrtch.hh	177
	psumf	
	scrtch.hh	177
	psumg	
	scrtch.hh	177

## INDEX OF VARIABLES IN COMDECKS

ptans  
     voldat.hh ..... 208  
 ptitle  
     genrl.hh ..... 112  
 ptrip  
     stcom.hh ..... 190  
 pump ..... 41  
     cmpalf.hh ..... 90  
 pump component ..... 146  
 pump2 ..... 41  
 pumpblk.hh comdeck ..... 146  
 pumpblk.hh comdeck ..... 146  
 pumpspeedtblreferralflg14  
     cmpdat.hh ..... 94  
 pumpspeedtblreferralsatflg15  
     cmpdat.hh ..... 94  
 pumpv  
     scrtch.hh ..... 177  
 pvblk  
     htrcom.hh ..... 116  
 pzhtcf  
     przdat.hh ..... 146  
 pzhtcg  
     przdat.hh ..... 146  
 pzpres  
     ssiblk.hh ..... 185

**Q**

q  
     voldat.hh ..... 208  
 qffo  
     htrcom.hh ..... 116  
 qfgo  
     htrcom.hh ..... 116  
 qfox  
     htrcom.hh ..... 116  
 qfluxo  
     htrcom.hh ..... 116  
 qlrad  
     radhtcc.hh ..... 150  
 qn  
     statec ..... 188  
 qrad  
     radhtcc.hh ..... 150  
 qradlr  
     htscr.hh ..... 120

qrrad  
     radhtcc.hh ..... 150  
 qsater  
     scrtch.hh ..... 177  
 qtank  
     cmpdac.hh ..... 91  
 qtanko  
     cmpdac.hh ..... 91  
 qter  
     scrq.hh ..... 164  
     scrtch.hh ..... 177  
 qual  
     scrq.hh ..... 164  
     scrtch.hh ..... 177  
 quala  
     voldat.hh ..... 208  
 qualaj  
     jundat.hh ..... 134  
 qualan  
     voldat.hh ..... 208  
 qualao  
     voldat.hh ..... 208  
 quale  
     voldat.hh ..... 208  
 qualem  
     scrtch.hh ..... 177  
 qualep  
     htrcom.hh ..... 116  
 qualnj  
     jundat.hh ..... 134  
 qualno  
     voldat.hh ..... 208  
 quals  
     voldat.hh ..... 208  
 qwf  
     voldat.hh ..... 208  
 qwg  
     voldat.hh ..... 208

**R**

radhtc.hh comdeck ..... 149  
 radhtcc.hh comdeck ..... 149  
 radiation heat transfer ..... 149  
 ratdpf  
     scrtch.hh ..... 177  
 ratio

scrtch.hh	177	scrtch.hh	178
ratiof		reynl1	
scrtch.hh	177	scrtch.hh	178
ratiog		reyn2	
scrtch.hh	177	scrtch.hh	178
ravrf		rflhtc common block	151
scrtch.hh	177	rflhtc.hh comdeck	151
ravrg		rvfj	
scrtch.hh	177	scrtch.hh	178
rax		rvfrc	
statec	188	scrtch.hh	178
rcompa common block	150	rgvgj	
rcompc common block	150	scrtch.hh	178
rcompc.hh comdeck	150	rgvgrc	
rdin		scrtch.hh	178
rrkinc.hh	162	rhfg	
rdsylv		scrtch.hh	178
cmpdtv.hh	96	rho	
reactor kinetics	152	voldat.hh	209
reactor kinetics data	162	rhocpf	
recipv		scrtch.hh	178
voldat.hh	208	rhof	
reclim		voldat.hh	209
ftbcom.hh	111	rhofa	
recrit		scrtch.hh	178
voldat.hh	209	rhofj	
refbun		jundat.hh	134
flood.hh	110	rhofs	
reflood heat transfer	151	scrq.hh	164
reflood scratch variables	117, 118	scrtch.hh	178
reflood variables	116	rhog	
refset		voldat.hh	209
radhtcc.hh	150	rhoga	
reftempunits10		scrtch.hh	178
rkinc.hh	152	rhogj	
reftempunits11		jundat.hh	134
rkinc.hh	152	rhogo	
resorm		voldat.hh	209
scrtch.hh	177	rhogs	
resoru		scrq.hh	164
scrtch.hh	177	scrtch.hh	178
rey		rhom	
htrcom.hh	116	voldat.hh	209
reynfl		rhon	
scrtch.hh	178	cmpdac.hh	91
reynf2		rhono	

## INDEX OF VARIABLES IN COMDECKS

cmpdac.hh	91	rkinc.hh	153
rhoo		rkdopi	
voldat.hh	209	rkinc.hh	154
rhos		rkdopr	
htrcom.hh	116	rkinc.hh	154
rhot		rkdppt	
cmpdac.hh	91	rkinc.hh	154
rhov		rkdt	
scrtch.hh	178	rkinc.hh	154
rkbol		rkfcd	
rkinc.hh	152	rkinc.hh	154
rkcapt		rkfi	
rkinc.hh	152	rkinc.hh	154
rkcfl		rkfta	
rkinc.hh	153	rkinc.hh	154
rkcfl2		rkfu38	
rkinc.hh	153	rkinc.hh	154
rkcfl3		rkfwf	
rkinc.hh	153	rkinc.hh	154
rkcflh1		rkhtno	
rkinc.hh	153	rkinc.hh	154
rkcflh2		rkil	
rkinc.hh	153	rknatb.hh	161
rkcflh3		rkinc.hh comdeck	152
rkinc.hh	153	rkinc.cc comdeck	152
rkcoef		rkincfeedbackpt5	
rkinc.hh	153	rkinc.hh	154
rkcfl1		rkincpoint1dflg7	
rkinc.hh	153	rkinc.hh	154
rkcfl2		rklmda	
rkinc.hh	153	rkinc.hh	154
rkcfl3		rknatb common block	161
rkinc.hh	153	rknatb.hh comdeck	161
rkcflh1		rknatbc.hh comdeck	161
rkinc.hh	153	rknden	
rkcflh2		rkinc.hh	154
rkinc.hh	153	rkns	
rkcflh3		rkinc.hh	154
rkinc.hh	153	rknsr	
rkdeni		rkinc.hh	154
rkinc.hh	153	rknsfb	
rkdentr		rkinc.hh	155
rkinc.hh	153	rkntbl	
rkdepv		rknatb.hh	161
rkinc.hh	153	rkntbx	
rkdnp		rknatb.hh	161

rknum	
rkinc.hh	..... 155
rknumd	
rkinc.hh	..... 155
rknumi	
rrkinc.hh	..... 162
rknvfb	
rkinc.hh	..... 155
rko	
rkinc.hh	..... 155
rkoacf	
rkinc.hh	..... 155
rkoalp	
rkinc.hh	..... 155
rkoazf	
rkinc.hh	..... 155
rkob	
rkinc.hh	..... 155
rkooby	
rkinc.hh	..... 155
rkocfg	
rkinc.hh	..... 155
rkoofi	
rkinc.hh	..... 155
rkocid	
rkinc.hh	..... 155
rkoden	
rkinc.hh	..... 155
rkoegv	
rkinc.hh	..... 155
rkoepk	
rkinc.hh	..... 156
rkofbs	
rkinc.hh	..... 156
rkofbv	
rkinc.hh	..... 156
rkoffa	
rkinc.hh	..... 156
rkoffd	
rkinc.hh	..... 156
rkofi	
rkinc.hh	..... 156
rkoih0	
rkinc.hh	..... 156
rkoil2	
rkinc.hh	..... 156
rkoiot	
rkinc.hh	..... 156
rkoips	
rkinc.hh	..... 156
rkoizd	
rkinc.hh	..... 156
rkoizu	
rkinc.hh	..... 156
rkokit	
rkinc.hh	..... 156
rkolmd	
rkinc.hh	..... 156
rkomer	
rkinc.hh	..... 156
rkomeg	
rkinc.hh	..... 156
rkomnd	
rkinc.hh	..... 156
rkonc	
rkinc.hh	..... 157
rkoncf	
rkinc.hh	..... 157
rkongt	
rkinc.hh	..... 157
rkonhr	
rkinc.hh	..... 157
rkonmg	
rkinc.hh	..... 157
rkonnn	
rkinc.hh	..... 157
rkonng	
rkinc.hh	..... 157
rkonnm	
rkinc.hh	..... 157
rkonnx	
rkinc.hh	..... 157
rkonny	
rkinc.hh	..... 157
rkonzz	
rkinc.hh	..... 157
rkonr	
rkinc.hh	..... 157
rkonsr	
rkinc.hh	..... 158
rkonth	
rkinc.hh	..... 158
rkonvr	

## INDEX OF VARIABLES IN COMDECKS

rkinc.hh .....	158	rkinc.hh .....	159
rkony rkinc.hh .....	158	rkozfi rkinc.hh .....	159
rkonz rkinc.hh .....	158	rkozfp rkinc.hh .....	159
rkonzf rkinc.hh .....	158	rkozgp rkinc.hh .....	159
rkool2 rkinc.hh .....	158	rkozid rkinc.hh .....	159
rkooli rkinc.hh .....	158	rkozkp rkinc.hh .....	159
rkopt rkinc.hh .....	158	rkoztp rkinc.hh .....	159
rkoral rkinc.hh .....	158	rkpow rkinc.hh .....	159
rkorcfc rkinc.hh .....	158	rkpowa rkinc.hh .....	160
rkorc1 rkinc.hh .....	158	rpkpowf rkinc.hh .....	160
rkorf rkinc.hh .....	158	rpkpowg rkinc.hh .....	160
rkorid rkinc.hh .....	158	rpkpowk rkinc.hh .....	160
rkorif rkinc.hh .....	158	rkpsi rkinc.hh .....	160
rkorin rkinc.hh .....	158	rkqval rkinc.hh .....	160
rkosn rkinc.hh .....	159	rkrn rkinc.hh .....	160
rkoswf rkinc.hh .....	159	rkro rkinc.hh .....	160
rkosym rkinc.hh .....	159	rkslob rkinc.hh .....	160
rkotf rkinc.hh .....	159	rksptr rkinc.hh .....	160
rkotm rkinc.hh .....	159	rksum rkinc.hh .....	160
rkovn rkinc.hh .....	159	rktabl rkinc.hh .....	160
rkovwf rkinc.hh .....	159	rkvoln rkinc.hh .....	160
rkozap rkinc.hh .....	159	rkvta rkinc.hh .....	160
rkozdv rkinc.hh .....	159	rkvwf rkinc.hh .....	160
rkozfg		rmadac common block	161

rmadac.hh comdeck .....	161	scrtch.hh comdeck .....	165
roughv		scrtchc.hh comdeck .....	165
voldat.hh .....	209	scrch	
rrkinc common block .....	162	scrtch.hh .....	178
rrkinc.hh comdeck .....	162	scv1	
rstblknumber1531		scrtch.hh .....	178
contrl.hh .....	105	scv10	
rstplt		scrtch.hh .....	179
ufiles.hh.....	197	scv2	
rverit		scrtch.hh .....	178
scrtch.hh .....	178	scv2n	
rwtrstpltcmprssflag13		scrtch.hh .....	178
contrl.hh .....	105	scv3	
<b>S</b>		scrtch.hh .....	178
s		scv4	
statec .....	188	scrtch.hh .....	178
tmsrcm.hh .....	192	scv5	
safe1		scrtch.hh .....	178
comctl.hh.....	100	scv6	
safe2		scrtch.hh .....	179
contrl.hh .....	105	scv7	
safe7		scrtch.hh .....	179
lvel.hh .....	140	scv8	
sathf		scrtch.hh .....	179
voldat.hh .....	209	scv9	
sathfp		scrtch.hh .....	179
htrcom.hh .....	116	scvj1	
sathfx		scrtch.hh .....	179
scrtch.hh .....	178	scvj11	
sathg		scrtch.hh .....	179
voldat.hh .....	209	scvj12	
sathgx		scrtch.hh .....	179
scrtch.hh .....	178	scvj2	
satt		scrtch.hh .....	179
voldat.hh .....	209	scvj22	
sbmlhf		scrtch.hh .....	179
scrtch.hh .....	178	scvj3	
sbmllf		scrtch.hh .....	179
scrtch.hh .....	178	scvj33	
scndjunblkommittd7		scrtch.hh .....	179
tstpct.hh .....	196	scvj4	
scratch variables.....	165	scrtch.hh .....	179
scrchh		scvj44	
scrtch.hh .....	178	scrtch.hh .....	179
scrq.hh comdeck .....	163	scvj5	
		scrtch.hh .....	179

## INDEX OF VARIABLES IN COMDECKS

scvj6	
scrtch.hh .....	179
scvj7	
scrtch.hh .....	179
scvj8	
scrtch.hh .....	179
scvjck	
scrtch.hh .....	179
scvjn	
scrtch.hh .....	179
scvtur	
scrtch.hh .....	179
separ.hh comdeck	183
separator component	183
separatr	
cmpalf.hh .....	90
separc.hh comdeck	183
setno	
radhtcc.hh .....	150
sflag	
lpdat.hh .....	138
trnhlp.hh .....	192
sigma	
voldat.hh .....	209
signbitlefttripnum4	
trpbblk.hh .....	194
signbitrighttripnum5	
trpbblk.hh .....	194
sinb	
voldat.hh .....	209
sinbt	
scrtch.hh .....	179
size	
ftbcom.hh .....	111
skipt	
contrl.hh .....	105
smndrh	
sscbtr.hh .....	184
smnrho	
sscntr.hh .....	184
smxdrh	
sscntr.hh .....	184
smxrho	
sscntr.hh .....	184
sngljun	
cmpalf.hh .....	90
snglphasreferalsatflg7	
cmpdat.hh .....	94
snglphasreferralflg6	
cmpdat.hh .....	94
snglvlol	
cmpalf.hh .....	90
snk	
scrtch.hh .....	179
snl	
scrtch.hh .....	179
soncjo	
jundat.hh .....	134
sonicj	
jundat.hh .....	134
sorp	
scrtch.hh .....	179
sounde	
voldat.hh .....	209
soura	
scrtch.hh .....	179
sourca	
scrtch.hh .....	179
sourcef	
scrtch.hh .....	179
sourceg	
scrtch.hh .....	180
sourci	
scrtch.hh .....	180
sourcm	
scrtch.hh .....	180
sourcn	
scrtch.hh .....	180
sourcep	
scrtch.hh .....	180
sourceq	
scrtch.hh .....	180
sourcev	
scrtch.hh .....	180
sourff	
scrtch.hh .....	180
sourgg	
scrtch.hh .....	180
sourn	
scrtch.hh .....	180
span	
voldat.hh .....	209

spckincdensvar12		
rkinc.hh	.....	160
spckincpwrflg13		
rkinc.hh	.....	160
sprcon		
cmpdtv.hh	.....	96
sqr2		
cons.hh	.....	100
sqr2p		
cons.hh	.....	101
sqrtpi		
cons.hh	.....	100
srcamn		
voldat.hh	.....	209
srgjun		
przdat.hh	.....	146
srgvol		
przdat.hh	.....	146
sscntr.hh comdeck		184
sscntrc.hh comdeck		184
sscnvgnottested4		
tstpct.hh	.....	196
ssiblk.hh comdeck		185
ssiblkc.hh comdeck		185
ssiskp		
ssiblk.hh	.....	185
stantn		
htsrcm.hh	.....	126
stanto		
htsrcm.hh	.....	126
statc.hh comdeck		185
statcc.hh comdeck		185
statec common block		100
statec.hh comdeck		187
statecc.hh comdeck		187
stateq		163
statsblkommittd9		
tstpct.hh	.....	196
stcblk.hh comdeck		189
stcblkc.hh comdeck		189
stccf1		
statc.hh	.....	186
stccf2		
statc.hh	.....	186
stcom.hh comdeck		189
stcomc.hh comdeck		189
stdfncttype2431		109
convarc.hh	.....	109
stdry		46
stdsp		46
stdtrn		
contrl.hh	.....	105
steady state self initialization		185
steady state variables		184
sth2xt		
ufiles.hh	.....	197
sth2xv		
voldat.hh	.....	209
stjck1		
statc.hh	.....	186
stjck2		
statc.hh	.....	186
stjpk1		
statc.hh	.....	186
stjpk2		
statc.hh	.....	186
stjskp		
statc.hh	.....	186
stlte1		
statc.hh	.....	186
stlte2		
statc.hh	.....	186
strap1		
statc.hh	.....	186
strap2		
statc.hh	.....	186
strcl1		
statc.hh	.....	186
strcl2		
statc.hh	.....	186
strdc1		
statc.hh	.....	186
strdc2		
statc.hh	.....	186
strdp1		
statc.hh	.....	186
strdp2		
statc.hh	.....	186
stret		
scrtch.hh	.....	180
strexp1		
statc.hh	.....	186

## INDEX OF VARIABLES IN COMDECKS

stre2		
statc.hh .....	186	
strgeo		
htrlb.hh.....	117	
stripf		
ufiles.hh.....	198	
strnpl		
htsrcm.hh.....	126	
strpe1		
statc.hh .....	186	
strpe2		
statc.hh .....	187	
strte1		
statc.hh .....	187	
strte2		
statc.hh .....	187	
strxl1		
statc.hh .....	187	
strxl2		
statc.hh .....	187	
stsatp		
statc.hh .....	187	
stsc11		
statc.hh .....	187	
stsc12		
statc.hh .....	187	
stscpu		
statc.hh .....	187	
stsdata		
statc.hh .....	187	
stsdtm		
statc.hh .....	187	
stsdtx		
statc.hh .....	187	
stsjpt		
statc.hh .....	187	
stslen		
statc.hh .....	187	
stsreq		
statc.hh .....	187	
stsskp		
statc.hh .....	187	
stvpk1		
statc.hh .....	187	
stvpk2		
statc.hh .....	187	
subrtnum0710		
htsrcm.hh.....	126	
succes		
contrl.hh .....	105	
sumdpk		
scrtch.hh.....	180	
sumdpl		
scrtch.hh.....	180	
sumf		
scrtch.hh.....	180	
sumg		
scrtch.hh.....	180	
sumold		
scrtch.hh.....	180	
sumvfx		
scrtch.hh.....	180	
sumvgx		
scrtch.hh.....	180	
sysdate.hh comdeck		190
sysdatm.hh comdeck		190
sysdtc		
lpdat.hh .....	138	
sysebt		
lpdat.hh .....	138	
sysel		
sysdate.hh.....	191	
sysmaf		
sysdate.hh.....	191	
sysmat		
sysdate.hh.....	191	
sysmer		
lpdat.hh .....	138	
sysnam		
sysdate.hh.....	191	
sysopt		
sysdate.hh.....	191	
syssol		44
system		190
systemc		
lpdat.hh .....	138	
systemo		
lpdat.hh .....	138	
systems		
lpdat.hh .....	138	
sysvol		
sysdate.hh.....	191	

**szz**  
 ftbcom.hh ..... 111  
  
**T**  
**t1**  
 vreqd.hh ..... 212  
**t10a**  
 vreqd.hh ..... 220  
**t10b**  
 vreqd.hh ..... 220  
**t11**  
 vreqd.hh ..... 220  
**t2**  
 vreqd.hh ..... 213, 214  
**t3**  
 vreqd.hh ..... 215  
**t4**  
 vreqd.hh ..... 215  
**t5**  
 vreqd.hh ..... 216  
**t6**  
 vreqd.hh ..... 217  
**t7**  
 vreqd.hh ..... 217  
**t8**  
 vreqd.hh ..... 217  
**t9a**  
 vreqd.hh ..... 218  
**t9b**  
 vreqd.hh ..... 218  
**t9c**  
 vreqd.hh ..... 219  
**t9d**  
 vreqd.hh ..... 219  
**t9e**  
 vreqd.hh ..... 219  
**taa**  
 scrtch.hh ..... 180  
**tao**  
 statec ..... 188  
**targcn**  
 makmap.hh ..... 142  
**target**  
 makmap.hh ..... 142  
**targmp**  
 makmap.hh ..... 143

**tbb**  
 scrtch.hh ..... 180  
**tblnum**  
 cmpdtv.hh ..... 96  
**tchfjf**  
 htrflb.hh ..... 117  
**tcouri**  
 scrtch.hh ..... 180  
**tcrit**  
 stcom.hh ..... 190  
**tdpptr.hh comdeck**  
 ..... 191  
**temp**  
 voldat.hh ..... 209  
**tempf**  
 voldat.hh ..... 209  
**tempg**  
 voldat.hh ..... 209  
**testda**  
 contrl.hh ..... 105  
**tf**  
 htrcom.hh ..... 116  
 scrq.hh ..... 164  
 scrtch.hh ..... 180  
**thca**  
 statec ..... 188  
**thcb**  
 statec ..... 188  
**thcnd**  
 cmpdac.hh ..... 91  
**thconf**  
 voldat.hh ..... 210  
**thcong**  
 voldat.hh ..... 210  
**thcons**  
 htrcom.hh ..... 116  
**thermal properties**  
 ..... 144  
**thermodynamic property file names** ..... 144  
**theta**  
 cmpdtv.hh ..... 96  
**thetao**  
 cmpdtv.hh ..... 96  
**thick**  
 cmpdac.hh ..... 91  
**thrdfrthvolblkommittd8**  
 tstpct.hh ..... 196  
**tiengv**

## INDEX OF VARIABLES IN COMDECKS

voldat.hh .....	210	tmpsfn htrflb.hh.....	117
time dependent volume-junction pointers.....		tmpsfo htrflb.hh.....	117
191		tmsrcm.hh comdeck .....	191
time step control.....	194	tnmins scrtch.hh.....	180
timeht		tnplus scrtch.hh.....	181
contrl.hh .....	106	toface911 jundat.hh .....	134
timehy		tpdpdx scrtch.hh.....	181
contrl.hh .....	106	tpfnam mxnfcd.hh .....	145
timeoflgleftvar2		tpfncl mxnfcd.hh .....	145
trpbblk.hh.....	194	tpfnha sysdate.hh.....	191
timeoflgrightvar3		tpfnin mxnfcd.hh .....	145
trpbblk.hh.....	194	tquala scrtch.hh.....	181
timinv		transrotatflag14 contrl.hh .....	106
scrtch.hh.....	180	tref statec .....	188
timrof		trewet htrflb.hh.....	117
radhtcc.hh.....	150	trip components.....	192
timron		tripcomplement6 convarc.hh.....	109
radhtcc.hh.....	150	trmin radhtcc.hh.....	150
tintf		trmm scrtch.hh.....	181
voldat.hh .....	210	trmm1 scrtch.hh.....	181
tklen		trnhlp.hh comdeck .....	192
cmpdac.hh .....	92	trnhlpc.hh comdeck .....	192, 194
tloc		trpbblk.hh comdeck .....	192
scrtch.hh.....	180	trpbblkc.hh comdeck .....	192
tloc2		trpcon trpbblk.hh.....	194
scrtch.hh.....	180	trptim trpbblk.hh.....	194
tmass			
contrl.hh .....	106		
tmasso			
contrl.hh .....	106		
tmassv			
voldat.hh .....	210		
tmax			
stcom.hh.....	190		
tmddpjn			
cmpalf.hh .....	90		
tmddpvol			
cmpalf.hh .....	90		
tmin			
stcom.hh.....	190		
tmprfn			
htrflb.hh.....	117		
tmprfo			
htrflb.hh.....	117		
tmpscr			
htscr2.hh.....	119		

trptm	
trblk.hh	194
tsat	
scrq.hh	164
scrtch.hh	181
tsater	
scrq.hh	164
scrtch.hh	181
tsatt	
voldat.hh	210
tscclc.hh comdeck	194
tpctr	
tstpct.hh	196
tspend	
tstpct.hh	196
tsppac	
tstpct.hh	196
tsppmi	
tstpct.hh	196
tsppmj	
tstpct.hh	196
tspprs	
tstpct.hh	196
tspskp	
tstpct.hh	196
tstpct.hh comdeck	194
tstpctc.hh comdeck	194
tt	
scrq.hh	164
scrtch.hh	181
ttank	
cmpdac.hh	92
ttanko	
cmpdac.hh	92
ttempi	
voldat.hh	210
tter	
scrq.hh	164
scrtch.hh	181
ttg	
scrq.hh	164
scrtch.hh	181
ttrip	
stcom.hh	190
tty	
ufiles.hh	198
tuf	
scrtch.hh	181
tug	
scrtch.hh	181
turbin.hh comdeck	196
turbinc.hh comdeck	196
turbine	
cmpalf.hh	90
turbine component	196
turst	42
turctr	
turbin.hh	196
turdef	
turbin.hh	196
tureff	
turbin.hh	196
turfr	
turbin.hh	196
turint	
turbin.hh	196
turpow	
turbin.hh	197
turrrds	
turbin.hh	197
tursem	
turbin.hh	197
turtrq	
turbin.hh	197
turupj	
turbin.hh	197
turvel	
turbin.hh	197
turx	
turbin.hh	197
tvapo	
cmpdac.hh	92
tw	
htrcom.hh	116
twchf	
flood.hh	110
twnvg	
flood.hh	110
twophasereferralsflg10	
cmpdat.hh	94
twophasereferralsatflg11	
cmpdat.hh	95

## INDEX OF VARIABLES IN COMDECKS

twopi	convarc.hh.....	109
cons.hh .....		
twqf	units5	
flood.hh .....	convarc.hh.....	109
<b>U</b>		
uaox	uo	
statec .....	ssctrn.hh.....	184
ubar	us	
scrq.hh.....	scrtch.hh.....	181
scrtch.hh.....	user variables .....	198
uf	usrvar.hh comdeck .....	198
voldat.hh .....	ustm	
ufao	voldat.hh .....	210
voldat.hh .....	ustmo	
ufbo	voldat.hh .....	210
voldat.hh .....	usubf	
ufilef.hh comdeck .....	scrq.hh.....	165
ufilef/ common block.....	scrtch.hh.....	181
ufiles common block.....	<b>usubfs</b>	
ufiles.hh comdeck .....	scrq.hh.....	165
ufilesc.hh comdeck .....	scrtch.hh.....	181
ufj	<b>usubg</b>	
jundat.hh .....	scrq.hh.....	165
ufla	scrtch.hh.....	181
voldat.hh .....	<b>usubgs</b>	
uflb	scrq.hh.....	165
voldat.hh .....	scrtch.hh.....	181
ufnc	<b>V</b>	
scrtch.hh.....	v	
ufo	voldat.hh .....	210
voldat.hh .....	<b>vagrg</b>	
ug	scrtch.hh.....	181
voldat.hh .....	valve	
ugj	cmpalf.hh .....	90
jundat.hh .....	valve components .....	95
ugnc	vapgen	
scrtch.hh.....	voldat.hh .....	210
ugo	vapgno	
voldat.hh .....	voldat.hh .....	210
unit test.....	vbar	
contrl.hh .....	scrq.hh.....	165
uniti	scrtch.hh.....	181
contrl.hh .....	<b>vcnfnd</b>	
unito	voldat.hh .....	210
contrl.hh .....	vercrit	
units0	stcom.hh.....	190
	vctrl	

voldat.hh .....	211	cmpdat.hh.....	95
vctrld		velvcf	
voldat.hh .....	211	scrtch.hh.....	181
vctrln		velvcg	
voldat.hh .....	211	scrtch.hh.....	181
vctrls		velvl	
voldat.hh .....	211	cmpdtv.hh .....	96
vctrlx		velvlo	
voldat.hh .....	211	cmpdtv.hh .....	96
vdfjoo		vfa	
jundat.hh .....	134	scrtch.hh.....	181
vdgjoo		vfdpk	
jundat.hh .....	135	scrtch.hh.....	181
vdm		vfdpl	
cmpdac.hh .....	92	scrtch.hh.....	182
vdmo		vfi	
cmpdac.hh .....	92	radhtcc.hh.....	150
vdryl		vfront	
separ.hh .....	184	voldat.hh .....	211
vdryu		vga	
separ.hh .....	184	scrtch.hh.....	182
velf		vgdpk	
voldat.hh .....	211	scrtch.hh.....	182
velfj		vgdpl	
jundat.hh .....	135	scrtch.hh.....	182
velfjo		vglj	
jundat.hh .....	135	jundat.hh .....	135
velfjs		visao	
scrtch.hh.....	181	statec .....	188
velfo		viscf	
voldat.hh .....	211	voldat.hh .....	211
velfoo		viscg	
voldat.hh .....	211	voldat.hh .....	211
velg		viscos1	
voldat.hh .....	211	.....	41
velgj		viscs	
jundat.hh .....	135	htrcom.hh .....	116
velgio		vlev	
jundat.hh .....	135	voldat.hh .....	211
velgjs		vlevo	
scrtch.hh.....	181	voldat.hh .....	211
velgo		vlfjos	
voldat.hh .....	211	jundat.hh .....	135
velgoo		vlfmas	
voldat.hh .....	211	cmpdtv.hh .....	96
velmassflwswtch0		vlgjos	
		jundat.hh .....	135

## INDEX OF VARIABLES IN COMDECKS

vliq		voldat.hh .....	211
cmpdac.hh .....	92		
vliqo		voidfa	
cmpdac.hh .....	92	scrtch.hh .....	182
vlpndx		voidfj	
lcntrl.hh .....	136	jundat.hh .....	135
vlsetp		voidg	
cmpdtv.hh .....	96	voldat.hh .....	211
vlstm		voidga	
cmpdtv.hh .....	96	scrtch.hh .....	182
vlstmo		voidgd	
cmpdtv.hh .....	96	scrtch.hh .....	182
vlstmx		voidgj	
cmpdtv.hh .....	97	jundat.hh .....	135
vlvcon		voidgk	
cmpdtv.hh .....	97	scrtch.hh .....	182
vlvnm		voidgl	
cmpdtv.hh .....	97	scrtch.hh .....	182
vlvslp		voidgo	
cmpdtv.hh .....	97	voldat.hh .....	211
vlvtrp		voidgu	
cmpdtv.hh .....	97	scrtch.hh .....	182
vmgnx		voidij	
scrtch.hh .....	182	jundat.hh .....	135
vngen		voidla	
scrtch.hh .....	182	voldat.hh .....	211
vo		voidlb	
voldat.hh .....	211	voldat.hh .....	212
vodfjo		voidlt	
jundat.hh .....	135	lvel.hh .....	140
vodfjr		voidmn	
jundat.hh .....	135	radhtcc.hh .....	150
vodgjo		voiddat.hh comdeck .....	198
jundat.hh .....	135	voiddatc.hh comdeck .....	198
vodgjr		vollev	
jundat.hh .....	135	voldat.hh .....	212
vodgoo		volmat	
voldat.hh .....	211	voldat.hh .....	212
void fraction		volno	
.....	221	voldat.hh .....	212
voidaa		volume variables .....	198
scrtch.hh .....	182	vover	
voidao		.....	46
voldat.hh .....	211	separ.hh .....	184
voidbo		vpgen	
voldat.hh .....	211	scrtch.hh .....	182
voidf		vgpx	
		scrtch.hh .....	182

vreqs.hh comdeck .....	212
vrhof	
scrtch.hh .....	182
vrhog	
scrtch.hh .....	182
vruf	
scrtch.hh .....	182
vrug	
scrtch.hh .....	182
vxrg	
scrtch.hh .....	182
vxgn	
scrtch.hh .....	182
vs	
scrq.hh .....	165
scrtch.hh .....	182
vsubf	
scrq.hh .....	165
scrtch.hh .....	182
vsubfs	
scrq.hh .....	165
scrtch.hh .....	182
vsubg	
scrq.hh .....	165
scrtch.hh .....	182
vsubgs	
scrq.hh .....	165
scrtch.hh .....	182
vtank	
cmpdac.hh .....	92
vtrip	
stcom.hh .....	190
vunder .....	46
separ.hh .....	184
vvfx	
scrtch.hh .....	182
vvgx	
scrtch.hh .....	182

**W**

wetbot	
flood.hh .....	110
wettop	
flood.hh .....	110
wmolea	
statec .....	188

wmoles	
mxnfcfd.hh .....	145
wrplid subroutine .....	212

**X**

xawxyz	
machlf.hh .....	141
xco	
separ.hh .....	184
xcu	
separ.hh .....	184
xej	
jundat.hh .....	135
xim	
separ.hh .....	184
xliqh	
scrtch.hh .....	183
XMGR5 .....	212
xncn	
scrtch.hh .....	183
xvaph	
scrtch.hh .....	183

**Z**

zalfag.hh comdeck .....	221
zbun	
flood.hh .....	110
zbunht	
htrflb.hh .....	117
zqbot	
htrflb.hh .....	117
zqf	
flood.hh .....	110
zqftop	
flood.hh .....	110
zqtop	
htrflb.hh .....	117

## INDEX OF VARIABLES IN COMDECKS

**18 INDEX OF CARD NUMBERS**

Page	Page
<b>Numerics</b>	
1 .....	14, 23
100-101 .....	24
1001-1999 .....	24
102 .....	24
103 .....	13, 24
104 .....	13, 24
105 .....	24, 28
110 .....	15, 24
115 .....	15, 24
120-129 .....	16, 24
140-147 .....	23, 24
1CCCG000-1CCCG999 .....	26
1CCCGxxx.....	18
200 .....	14, 19
200-299 .....	24
201-199 .....	29
201-299 .....	14
201MMM00-201MMM99 .....	26
201MMNx .....	18
202TTT00-201TTT99 .....	26
202TTT00-99 .....	19
20500000 .....	19
20500000-205CCC98 .....	26
20500000-205CCCC8 .....	26
205CCC00 .....	19
205CCCC0 .....	19
20600000 .....	15
20600000-20620000 .....	24
20600010-206010000 .....	15
20610010-29620000 .....	15
2080001-20809999 .....	24
2080xxx .....	23
2080xxxx .....	19, 23
20900000-20903000 .....	26
2-5 .....	23
30000000 .....	19
30000000-30000499 .....	26
30000001 .....	19
30000002 .....	19
30000011-20 .....	19
30000011-30000020 .....	26
30000101-199 .....	19
30000201-299 .....	19
30000301 .....	19
30000401-499 .....	19
30000501-30002999 .....	26
30000501-599 .....	19
30000601-699 .....	19
30000701-799, .....	19
30000801-899 .....	19
30001701-1799 .....	19
30001801-1899 .....	19
300019C1-19C8 .....	19
30002001-2999 .....	19
301-399 .....	15, 22, 24
400 .....	15
400-799 .....	24
401-599 .....	15
600 .....	15
60000000 .....	18
60000000-6SSNN199 .....	26
601-799 .....	15
6SSNNxxx .....	18
801-899 .....	15, 24
<b>A</b>	
ACCUM .....	26
ANNULUS .....	25
<b>B</b>	
BRANCH .....	25
<b>C</b>	
CANDU .....	25
CCC0000 .....	16, 24
CCC0001-CCC0400 .....	25
CCC0001-CCC0600 .....	25
CCC0001-CCC3NNM .....	26
CCC0001-CCCN201 .....	25
CCC0101-CCC0200 .....	24
CCC0101-CCC0299 .....	25
CCC0101-CCC0499 .....	25

## INDEX OF CARD NUMBERS

CCC0101-CCC2200 ..... 26  
CCC0101-CCC3199 ..... 25  
CCC0101-CCC6199 ..... 25

### E

ECCMIX ..... 25

### J

JETMIXER ..... 25

### M

MTPLJUN ..... 26

### P

PIPE ..... 25  
PRIZER ..... 25  
PUMP ..... 25

### S

SEPARATR ..... 25  
SNGLJUN ..... 25  
SNGVOL ..... 24

### T

TMDPJUN ..... 25  
TMDPVOL ..... 24  
TURBINE ..... 25

### V

VALVE ..... 25

**19 INDEX OF COMMON BLOCKS**

	Page		Page
<b>C</b>			
cmpalf .....	89, 150	htrcomc .....	113
cmpdac .....	90	htrfbl .....	27, 116
cmpdacc .....	90	htrflbc .....	116
cmpdat .....	16, 24, 25, 26, 92	htscr .....	119
cmpdate .....	92	htscr1 .....	117, 118
cmpdtv .....	95	htscr2 .....	118
cmpdtvc .....	95	htscr2c .....	118
cnvtpa .....	97	htsrcm .....	18, 26, 120
cnvtpad .....	97	htsrcmc .....	120
comctl .....	24, 59, 61, 98	<b>I</b>	
cometlc .....	98	intrac .....	15, 24, 126
comlst .....	100	intracc .....	126
cons .....	100	invhtb .....	26, 127
contrl .....	23, 57, 63, 101	invhtbc .....	127
contrx .....	101	invtbl .....	26, 127, 128
convarc .....	19, 26, 106	invtblc .....	127
convarx .....	106	<b>J</b>	
<b>E</b>		jundat .....	21, 25, 26, 65, 66, 128
eccmxcc .....	109	jundatc .....	128
eccmxcc .....	109	<b>L</b>	
efilesd .....	228	lcntrl .....	135, 136
efiless .....	228	lcntrlc .....	135
<b>F</b>		levtbl .....	27, 136
fast .....	59, 61, 62, 63, 109	levtblc .....	136
fastc .....	109	lpdat .....	26, 27, 136, 137
flood .....	110	lpdatc .....	136
ftbcom .....	110	lvectr .....	27, 138, 139
<b>G</b>		lvectrc .....	138
gapvar .....	112	lvel .....	139, 140
genrl .....	112	<b>M</b>	
genrlc .....	112	machaf .....	140
gentblc .....	26, 112	machas .....	140
gentblx .....	112	macheff .....	140
<b>H</b>		maches .....	140
htrcom .....	113	machlf .....	140
		machls .....	140
		machof .....	142

## INDEX OF COMMON BLOCKS

machos ..... 142  
 machss ..... 142  
 makmap ..... 142  
 matdat ..... 26  
 maxmem ..... 143  
 miedtc ..... 24, 143  
 miedtcl ..... 143  
 mtblc ..... 144  
 mtbls ..... 18, 26, 144  
 mxnfcd ..... 144, 145

### N

npacom ..... 145

### P

plotpc ..... 146  
 przdat ..... 146  
 przdatac ..... 146  
 pumpblk ..... 146, 147  
 pumpblx ..... 146

### R

r5pvmc ..... 26  
 radhtc ..... 26, 149  
 radhtcc ..... 149  
 rcomp ..... 24  
 rcompc ..... 150, 151  
 rflhtc ..... 27, 151  
 rkinc ..... 19, 26, 152  
 rkinc ..... 152  
 rknatb ..... 19, 26, 161  
 rknatbc ..... 161  
 rmadac ..... 18, 161  
 rrkinc ..... 19, 26, 162  
 rstsum ..... 57

### S

seddat ..... 27  
 scrq ..... 163  
 scrpch ..... 41, 54, 67, 165  
 scrchc ..... 67, 165  
 separ ..... 183  
 separec ..... 183  
 ssctr ..... 26, 184  
 ssctr ..... 184

ssiblk ..... 24, 185  
 ssiblkc ..... 185  
 statec ..... 26, 185, 186  
 statcc ..... 185  
 statec ..... 15, 24, 187, 188  
 statecc ..... 187  
 stcblk ..... 24, 189  
 stcblk ..... 189  
 stcom ..... 24, 33, 189  
 stcomc ..... 189  
 steam tables ..... 24  
 sysdatc ..... 16, 24, 190  
 sysdatm ..... 190

### T

tdpptr ..... 26, 28, 191  
 tmsrcm ..... 191  
 trnhlp ..... 192  
 trnhlpc ..... 192  
 trpblk ..... 15, 24, 192, 193  
 trpblk ..... 192  
 tsctlc ..... 29, 194  
 tstpc ..... 24, 194, 195  
 tstpc ..... 194  
 turbin ..... 196  
 turbinc ..... 196

### U

ufilef ..... 24, 197, 261  
 ufiles ..... 197, 261  
 ufilesc ..... 197  
 usrvr ..... 19, 24, 198

### V

voldat ..... 24, 25, 26, 64, 198  
 voldatc ..... 198  
 vreqd ..... 212  
 vreqs ..... 212

### Z

zalfag ..... 221

## 20 INDEX OF SUBROUTINES

	Page		Page	
<b>A</b>				
acclev .....	31, 34	dispwethif.....	35, 37	
accum .....	41	dittus.....	31	
adjustthif .....	38, 40	drydrag.....	38, 40	
airprops .....	48	dryer.....	46	
amistdrag.....	38	dryhif.....	35, 37	
amisthif .....	35, 37	dtstep.....	28, 29	
<b>B</b>				
blkdta .....	29	eccmxj.....	38, 39	
brntrn.....	32, 47	eccmxv.....	35, 36	
bubbhif .....	35, 37	eqfinl .....	32, 41, 44, 45	
bugoutdrag .....	38, 40			
bugouthif.....	35, 38			
<b>C</b>				
ccfl .....	32, 41, 43	fidis2 .....	38, 40	
chfcal.....	31	fidisj .....	38	
chfd2o .....	31	fidisv .....	36	
chfkup .....	31	fldmp.....	27	
chforn .....	31	filterdrag.....	38, 40	
chfpfgf .....	31	filterhif.....	35, 38	
chfpgg .....	31	finaldrag.....	38, 40	
chfpgp .....	31	flostj .....	20	
chfsrl .....	31	flostv .....	21	
chftab .....	31	fpinit.....	27	
chklev .....	28, 30	fwdrag.....	31, 40	
chng18drag.....	38, 40			
classifyvols.....	47			
cmpcom.....	24			
conden .....	31			
condn2 .....	31			
convar.....	28, 32, 52			
courn1 .....	29			
cournt .....	30			
cvic.....	11, 81, 85, 87, 229			
cvirc .....	11, 81, 85, 87			
<b>D</b>				
dispdrag.....	38	helphd.....	35	
dispdryhif.....	35, 37, 40	hifbub .....	36	
<b>E</b>				
dispwethif.....	35, 37	hloss .....	31, 41	
dittus.....	31	horizdragreg.....	38, 39	
drydrag.....	38, 40	horizhif.....	35, 37	
dryer.....	46	horizhifreg.....	35, 36	
dryhif.....	35, 37			
dtstep.....	28, 29			
<b>F</b>				
eccmxj.....	38, 39	fidis2 .....	38, 40	
eccmxv.....	35, 36	fidisj .....	38	
eqfinl .....	32, 41, 44, 45	fidisv .....	36	
<b>G</b>				
gapcon.....	31	fldmp.....	27	
gasthc .....	31	filterdrag.....	38, 40	
gbsub .....	42	filterhif.....	35, 38	
gctpm .....	42	finaldrag.....	38, 40	
gesep .....	46	flostj .....	20	
gninit .....	9	flostv .....	21	
gtpmoody .....	42	fpinit.....	27	
<b>H</b>				
helphd.....	35	fwdrag.....	31, 40	
hifbub .....	36			
hloss .....	31, 41			
horizdragreg.....	38, 39			
horizhif.....	35, 37			
horizhifreg.....	35, 36			

## INDEX OF SUBROUTINES

horizstartdrag ..... 38  
 hseflw ..... 46  
 htlinp ..... 18  
 htlsst ..... 21  
 htltdp ..... 31, 49, 50  
 htadv ..... 28, 31, 49  
 htcond ..... 31, 50, 51  
 htfilm ..... 31, 36  
 htfinl ..... 32, 49  
 htlev ..... 35  
 htrc1 ..... 31  
 hydro ..... 28, 31, 33  
 hzflow ..... 46

**I**

icmpn1 ..... 10, 12, 20, 24  
 icompn ..... 10, 12, 20  
 icompn1 ..... 21  
 iconvr ..... 11, 12, 22, 52  
 idealgas ..... 48  
 igntbl ..... 10, 12, 20  
 ihtcmp ..... 10, 12, 21  
 imiedt ..... 11, 12, 15, 22, 24  
 imlp ..... 20, 26  
 inp ..... 27, 225, 228, 230  
 inp10 ..... 226, 230, 236  
 inp2 ..... 225, 226, 231, 235, 236  
 inp4 ..... 226, 232, 234, 236  
 inp5 ..... 226, 233, 234, 236  
 inp6 ..... 226, 235  
 inp7 ..... 226, 235  
 inp8 ..... 226, 235  
 inp9 ..... 226, 230, 236  
 inplnk ..... 225, 226, 236, 237  
 inpmod ..... 225, 226, 237  
 inppck ..... 238  
 inpupk ..... 238  
 inputd ..... 24  
 invanndrag ..... 38, 40  
 invannhif ..... 35, 37  
 invhts ..... 11, 12, 22, 26  
 invjt ..... 21, 26  
 invslugdrag ..... 38, 40  
 invslughif ..... 35, 37  
 ipipe ..... 20  
 iradht ..... 10, 12, 18, 21

irflht ..... 11, 12, 22, 27  
 irkin ..... 11, 12, 22, 51  
 isngj ..... 20  
 issi ..... 11, 12, 23  
 istate ..... 20  
 itrip ..... 10, 12, 20  
 itstck ..... 10, 12, 19  
 iusrvr ..... 11, 12, 23  
 ivelst ..... 20

## **J**

jchoke ..... 31, 41, 42  
 jetjhif ..... 35, 38  
 jprop ..... 32, 45, 46, 49

## **K**

kloss ..... 31, 50

## **L**

level ..... 32, 49  
 levelhif ..... 35, 37  
 levskt ..... 21, 27  
 locf ..... 61

## **M**

madata ..... 31, 50  
 masserror ..... 29, 47  
 mover ..... 28

## **N**

ncfilm ..... 35  
 ncwall ..... 31  
 newstread ..... 20  
 noncnd ..... 31  
 nth2x ..... 47  
 nwithair ..... 47, 48

## **P**

phantj ..... 31, 38, 44, 49  
 phantv ..... 31, 35, 45  
 pimplt ..... 32, 45  
 pltwr ..... 23  
 prebun ..... 31  
 prednb ..... 31

preseq ..... 43, 45  
 psatpd ..... 258  
 pstdnb ..... 31  
 pump ..... 41  
 pump2 ..... 41  
 pumpdragreg ..... 38, 39  
 pumphifreg ..... 35, 36  
 pzrlev ..... 28, 31

**Q**

qfmove ..... 31, 49, 50  
 qmwr ..... 31, 50

**R**

raccum ..... 26  
 radht ..... 22, 31, 49, 50  
 rbrnch ..... 25  
 rcards ..... 9  
 rchng ..... 9, 12, 14, 23  
 rcompn ..... 10, 12, 16, 24  
 rconvr ..... 10, 12, 19, 26, 52  
 relap5 ..... 27  
 rgntbl ..... 10, 12, 19, 26  
 rhelp ..... 23  
 rhtcmp ..... 10, 12, 18, 26  
 rintrv ..... 10, 12, 15, 24  
 rkin ..... 28, 32, 51  
 rmadat ..... 10, 12, 18, 26  
 rmflds ..... 10, 12, 16, 20, 24  
 rmiedt ..... 10, 12, 15, 22, 24  
 rmtplj ..... 26  
 rnewp ..... 9, 11, 12, 24  
 rmoncn ..... 10, 12, 15, 24  
 rpipe ..... 25  
 rpump ..... 25  
 rr5pvmc ..... 26  
 rradht ..... 10, 12, 26  
 rrestf ..... 9, 12, 13, 24  
 rrkin ..... 10, 12, 19, 26, 51  
 rrkinh ..... 19  
 rrkinp ..... 19, 26  
 rrstd ..... 9, 12, 13, 24, 262  
 rsngj ..... 25  
 rsngv ..... 24  
 rssi ..... 9, 24

rstrip ..... 24  
 rtmdj ..... 25  
 rtmdv ..... 24  
 rtrip ..... 10, 12, 15, 24  
 rtsc ..... 9, 12, 14, 19, 24  
 rturb ..... 25  
 rusrvr ..... 10, 12, 19, 24  
 rvalve ..... 25

**S**

simplt ..... 32, 45  
 simul ..... 18  
 slugdrag ..... 38  
 slughif ..... 35, 37  
 srestf ..... 24  
 stacc ..... 47  
 state ..... 29, 32, 47  
 statep ..... 47  
 statefinal ..... 47  
 stcset ..... 24, 31, 33  
 std2x ..... 47  
 stdsp ..... 46  
 stgh2o ..... 243, 244  
 sth2x ..... 47  
 sth2x0 ..... 243, 252, 259  
 sth2x1 ..... 253, 259  
 sth2x2 ..... 31, 243, 254  
 sth2x3 ..... 243, 256  
 sth2x4 ..... 243, 257  
 sth2x5 ..... 243, 257  
 sth2x6 ..... 243, 257  
 stread ..... 20, 243, 252  
 suboil ..... 31  
 supertosub ..... 47, 48  
 syssol ..... 43, 45

**T**

tcvi ..... 81, 87  
 tevid ..... 81  
 tfront ..... 31, 35  
 thcond1 ..... 48  
 tran ..... 11, 27, 28, 51, 52  
 trnctl ..... 11, 27  
 trnfin ..... 11, 27, 32  
 trnset ..... 11, 26, 27

## INDEX OF SUBROUTINES

tsetsl ..... 27, 34  
tstate ..... 28  
turbst ..... 41

### V

valve ..... 31, 34  
vertdragreg ..... 38, 39  
verthifreg ..... 35, 37  
vexplt ..... 31, 37, 38, 41, 44  
vfinl ..... 32, 41, 43  
vimplt ..... 32, 45  
viscos1 ..... 48  
vlvela ..... 32, 49  
volvel ..... 31, 34, 49  
vstrathif ..... 35, 37

### W

wetdrag ..... 38, 39  
wethif ..... 35, 37  
withair ..... 47, 48  
withoutair ..... 47, 48  
wrplid ..... 11, 12, 23

## 21 INDEX OF VARIABLES

	Page		Page
<b>A</b>			
a	235, 236, 237, 251, 252, 254, 256, 257, 258		
a3	228		
acctrp	20		
ajun	66, 67		
arat	66, 67		
athrot	66, 67		
avol	64, 65		
<b>B</b>			
b	256		
bin	82, 86, 87		
blhimap	65		
blhjc	66		
btflag	64		
bufr	261		
<b>C</b>			
c	255		
c0j	38		
c1	233, 234, 235		
c2	233, 234, 235		
c3	233, 234		
card	235, 237		
cc	256		
celvec	35		
chkvlv	34		
chngno	14		
cmpalf	89		
comdat	61		
comdln	61		
count	57		
cpurei	28		
crp	255		
<b>D</b>			
dfront	35		
diamj	66, 67		
diamv	64, 65		
dl	64, 65		
done	28		
<b>E</b>			
emass	29, 57		
err	251, 252, 254, 256, 257, 258		
errhi	29		
errlo	29		
<b>F</b>			
fa	59, 61, 62		
faaj	41		
fal	41		
ff	41		
fidxup	35		
fij	38, 41		
filid	18, 59		
filid(11)	19		
filid(12)	15		
filid(16)	22		
filid(18)	15		
filid(2)	15		
filid(21)	19		
filid(27)	19		
filid(28)	16		
filid(3)	16		
filid(32)	49		
filid(33)	19		
filid(38)	12, 18, 49		
filid(6)	27		
filid(7)	15		
filid(8)	12, 18		
filid(9)	18		
filndx	59		
filndx(1)	67		
filndx(11)	19		
filndx(12)	15		

## INDEX OF VARIABLES

filndx(16) .....	22
filndx(18) .....	15
filndx(2) .....	15
filndx(21) .....	19
filndx(27) .....	19
filndx(28) .....	16
filndx(3) .....	16
filndx(30) .....	54
filndx(33) .....	19
filndx(38) .....	18
filndx(4) .....	65
filndx(5) .....	66
filndx(6) .....	20
filndx(7) .....	15
filndx(8) .....	18
filndx(9) .....	18
filsch.....	261
filsiz .....	59
filsiz(11).....	19
filsiz(12).....	15
filsiz(16).....	22
filsiz(18).....	15
filsiz(2).....	15
filsiz(21).....	19
filsiz(27).....	19
filsiz(28).....	16
filsiz(3).....	16
filsiz(33).....	19
filsiz(38).....	18
filsiz(7).....	15
filsiz(8).....	18
filsiz(9).....	18
fjunft.....	41
fjunrt.....	41
florege.....	35
florgj.....	38
formfj.....	41
formgj.....	41
fwalf.....	35
fwalfj.....	41
fwalg.....	35
fwalgj.....	41
fwfxaf.....	35
fwfxag.....	35
fxj .....	38

## G

gal.....	41
gammac.....	44
gammaw.....	44
gg .....	41
gravcn.....	20
gravv .....	20

## H

headntorqmltplrlflg8 .....	20
hed.....	231
hgf.....	35, 45
hif.....	35, 45
hig .....	35, 45

## I

ia.....	62, 63
ibin .....	82, 87
ic.....	86, 87
icheck .....	49
ics .....	232
ihld3 .....	41
ij1 .....	66
ij1vn .....	66
ij2 .....	66
ij2vn .....	66
ijskp .....	67
imap .....	64
in1 .....	234
in2 .....	234
in3 .....	234
in4 .....	234
in5 .....	234
in6 .....	234, 235
in7 .....	234, 235
in8 .....	234, 235
inp9 .....	236
inrvlv .....	34
invcnt .....	21
invvno .....	21
invvnx .....	21
irstno .....	57
is4 .....	61
iscoorddirc14 .....	20
isdwnjunflg27 .....	21

isgasapperance6 ..... 48  
 ishorzvertjunflg26 ..... 21  
 islevelmodel0 ..... 28  
 ismapinfo05 ..... 20, 21  
 ismomfluxoff3 ..... 21  
 isoutletflag1 ..... 21  
 isreversecoordflg0 ..... 21  
 issys ..... 54  
 istdpjunflg1 ..... 20  
 istdpvol0 ..... 20  
 istoflag2 ..... 21  
 isupdwnjunflag27 ..... 20  
 isvertjunflg26 ..... 21  
 isw ..... 231  
 it ..... 251, 256, 257, 258  
 it3bp ..... 248  
 it3p0 ..... 248  
 it4bp ..... 248  
 it5bp ..... 248  
 item ..... 235  
 itype ..... 251, 258  
 ivskp ..... 53, 54, 65  
 ixjff ..... 54  
 ixvff ..... 54

**J**

j ..... 233, 237  
 j1 ..... 233  
 jc ..... 66  
 jcnxs ..... 54  
 junftl ..... 66

**L**

l1 ..... 228, 238  
 l2 ..... 228, 238  
 l3 ..... 228  
 list ..... 226, 228, 235  
 liv ..... 53, 54  
 livn ..... 53, 54  
 loc1 ..... 232, 233, 235, 237  
 loc2 ..... 232, 233, 234, 235  
 loc3 ..... 237  
 loc5 ..... 233, 234, 235  
 loc6 ..... 234, 235  
 lx1 ..... 239

**M**

many ..... 236, 237  
 maxz ..... 233  
 minz ..... 233  
 mnum ..... 237  
 motortorqreferalflag12 ..... 20  
 mtrvlv ..... 34

**N**

n ..... 237, 238, 252  
 n1 ..... 235  
 nc ..... 237  
 nc1 ..... 236  
 nc2 ..... 236  
 ncase ..... 228, 231  
 ncoms ..... 61  
 ndata ..... 230, 231  
 nerr ..... 86  
 newrst ..... 12  
 next ..... 236, 237  
 nfiles ..... 59  
 nj ..... 233  
 njco ..... 20  
 njun ..... 66, 67  
 njuns ..... 66  
 nl1 ..... 228, 231  
 np ..... 247, 248, 250  
 nprops ..... 249, 250  
 nrepeat ..... 30  
 nsp ..... 247, 249  
 nst ..... 247, 249  
 nt ..... 247, 248, 250  
 ntimes ..... 233, 234  
 ntot ..... 248  
 num ..... 86  
 numairvol ..... 47  
 numnoairvol ..... 47  
 numnormvol ..... 47  
 nuse ..... 252  
 nvco ..... 20  
 nvols ..... 64, 65

**P**

p ..... 44, 251, 252, 258  
 pcrit ..... 246

## INDEX OF VARIABLES

pdt ..... 251, 258  
 pfinrg ..... 35  
 pk ..... 41  
 pl ..... 41  
 plotalf ..... 23, 261, 262, 263  
 plotinf ..... 23, 261, 262, 263  
 plotnum ..... 23, 262  
 plotrec ..... 262  
 plow ..... 255  
 pmax ..... 247  
 pmhig ..... 41  
 pmin ..... 247  
 pmpph ..... 41  
 power ..... 41  
 pps ..... 48  
 problemopt611 ..... 13  
 pshig ..... 41  
 ptrip ..... 246  
 pxx1 ..... 255  
 pxx2 ..... 255  
 pxxx1 ..... 255  
 pxxxy3 ..... 255

### **Q**

quala ..... 44, 47, 48  
 qualao ..... 44

### **R**

recipv ..... 65  
 record ..... 252  
 restart ..... 262  
 rhom ..... 44  
 rlfvlv ..... 34  
 rradht ..... 18  
 rstplt ..... 13, 261, 262  
 rtime ..... 57

### **S**

s ..... 251, 254, 256, 257, 258  
 safel ..... 61  
 satfhfx ..... 41  
 sathgx ..... 41  
 savebtflag ..... 65  
 scskp ..... 67  
 scvjck ..... 41

scvtur ..... 41  
 singlphasreferralflag6 ..... 20  
 sourca ..... 41  
 sourcf ..... 41  
 sourcg ..... 41  
 sourcm ..... 41  
 sourcn ..... 41  
 srvvlv ..... 34  
 succes ..... 28  
 sysmer ..... 29

### **T**

t ..... 251, 252, 258  
 table ..... 226, 228  
 tcrit ..... 246  
 timehy ..... 30, 57, 63  
 tmass ..... 57  
 tmax ..... 247  
 tmin ..... 247  
 torque ..... 41  
 tpfh2o ..... 250  
 trpvlv ..... 34  
 ttrip ..... 246  
 twophasereferralflag10 ..... 20

### **U**

uf ..... 44  
 ustmr ..... 48

### **V**

v ..... 65  
 vapgen ..... 44  
 vcrit ..... 246  
 vctrls ..... 54  
 velf ..... 49  
 velfj ..... 30, 41, 66, 67  
 velfjo ..... 66, 67  
 velg ..... 49  
 velgj ..... 30, 41, 66, 67  
 vfdpk ..... 41  
 vfdpl ..... 41  
 vfront ..... 35  
 vgdpk ..... 41  
 vlvnm ..... 34  
 vname ..... 34

voidg ..... 44  
volmat ..... 64, 65  
volno ..... 64, 65  
vtrip ..... 246

**W**

where ..... 237  
writecommonblk2 ..... 61  
writedynamicfile0 ..... 61

**X**

xl1 ..... 228, 230, 231, 236, 239

## INDEX OF VARIABLES