Fortran Interface

Initialization

```
CALL FENROLL(     PROC,       INUM )
CALL FINITIATE(   PROC, ARCH, INUM )
CALL FINITIATEM(  PROC, HOST, INUM )
CALL FWHOAMI(     PROC,       INUM, INFO )
```

Information

```
CALL FPSTATUS(    NCPUS, NTYPES, INFO )
CALL FSTATUS(     PROC,  INUM,   PSTATUS )
```

Sending

```
CALL FINITSEND( )
CALL FPUTBYTES(   CP,   CNT,         INFO )
CALL FPUTNCPLX(   XP,   CNT,         INFO )
CALL FPUTNDCPLX(  ZP,   CNT,         INFO )
CALL FPUTNDFLOAT( DP,   CNT,         INFO )
CALL FPUTNFLOAT(  FP,   CNT,         INFO )
CALL FPUTNINT(    NP,   CNT,         INFO )
CALL FPUTNLONG(   NP,   CNT,         INFO )
CALL FPUTNSHORT(  JP,   CNT,         INFO )
CALL FPUTSTRING(  CP,                INFO )
CALL FPUTSTRINGL( CP,   LEN,         INFO )
CALL FSND(        PROC, INUM, TYPE, INFO )
CALL FVSND(       PROC, INUM, TYPE, INFO )
```

Receiving

```
CALL FPROBE(              TYPE,  MT )
CALL FPROBEMULTI( NTYPES, TYPES, MT )
CALL FRCV(                TYPE,  MT )
CALL FRCVMULTI(   NTYPES, TYPES, MT )
CALL FVRCV(               TYPE,  MT )
CALL FVRCVMULTI(  NTYPES, TYPES, MT )
CALL FRCVINFO(    LEN, TYPE,  PROC, INUM, INFO )
CALL FGETBYTES(   CP, CNT, INFO )
CALL FGETNCPLX(   XP, CNT, INFO )
CALL FGETNDCPLX(  ZP, CNT, INFO )
CALL FGETNDFLOAT( DP, CNT, INFO )
CALL FGETNFLOAT(  FP, CNT, INFO )
CALL FGETNINT(    NP, CNT, INFO )
CALL FGETNLONG(   NP, CNT, INFO )
CALL FGETNSHORT(  JP, CNT, INFO )
CALL FGETSTRING(  CP,      INFO )
CALL FGETSTRINGL( CP, LEN, INFO )
```

Synchronization

```
CALL FBARRIER(   NAME, CNT, INFO )
CALL FREADY(     NAME,      INFO )
CALL FWAITUNTIL( NAME,      INFO )
```

Termination

```
CALL FLEAVE(  )
CALL FTERMINATE( PROC, INUM, INFO )
```

Error Handling

```
CALL FPVMPERROR( MSG )
```

ERROR CODE	MEANING
0	No Error
- 1	System Error
- 2	Bad Parameter
- 3	Barrier Count Mismatch
- 4	Name Too Long (32 char max)
- 5	Read Past End of Buffer
- 6	No Such Host
- 7	No Such Executable
- 8	No Such Process Component
- 9	No Message Available
-10	Can't Get Memory
-11	Process Already Enrolled
-12	Can't decode message

Declarations

```
INTEGER          CNT, INFO, INUM, LEN, MT
INTEGER          NCPUS, NP(CNT), NTYPES
INTEGER          PSTATUS, TYPE, TYPES(NTYPES)
INTEGER*2        JP(CNT)
CHARACTER        ARCH, CP, HOST, MSG, NAME, PROC
REAL             FP(CNT)
DOUBLE PRECISION DP(CNT)
COMPLEX          XP(CNT)
COMPLEX*16       ZP(CNT)
```


Parallel Virtual Machine

Quick Reference Guide

Release 2.4

May 21, 1992

University of Tennessee

Oak Ridge National Laboratory

Emory University

C Interface

Initialization

```
int inum = enroll(char *proc)
int inum = initiate(char *proc, char *arch)
int inum = initiateM(char *proc, char *host)
int cc = whoami(char *proc, int *inum)
```

Information

```
int cc = pstatus(int *ncpus, int *ntypes)
int stat = status(char *proc, int inum)
```

Sending

```
void initsend()
int cc = putbytes(char *cp, int cnt)
int cc = putncplx(float *xp, int cnt)
int cc = putndfloat(double *dp, int cnt)
int cc = putnfloat(float *fp, int cnt)
int cc = putnint(int *np, int cnt)
int cc = putnlong(long *np, int cnt)
int cc = putnshort(short *np, int cnt)
int cc = putstring(char *cp)
int cc = snd(char *proc, int inum, int type)
int cc = vsnd(char *proc, int inum, int type)
```

Receiving

```
int mt = probe(int type)
int mt = probemulti(int ntypes, int *types)
int mt = rcv(int type)
int mt = rcvmulti(int ntypes, int *types)
int mt = vrcv(int type)
int mt = vrcvmulti(int ntypes, int *types)
int cc = rcvinfo(int *len, int *type,
                 char *proc, int *inum)
int cc = getbytes(char *cp, int cnt)
int cc = getncplx(float *xp, int cnt)
int cc = getndcplx(double *zp, int cnt)
int cc = getndfloat(double *dp, int cnt)
int cc = getnfloat(float *fp, int cnt)
int cc = getnint(int *np, int cnt)
int cc = getnlong(long *np, int cnt)
int cc = getnshort(short *np, int cnt)
int cc = getstring(char *cp)
```

Synchronization

```
int cc = barrier(char *name, int cnt)
int cc = ready(char *name)
int cc = waituntil(char *name)
```

Termination

```
int cc = terminate(char *proc, int inum)
void leave()
```

Error Handling

```
#include "pvmuser.h"

extern int pvm_errno;
extern char *pvm_errlist[];
extern int pvm_nerr;
extern int pvm_aerr;

void pvm_perror(char *msg)
```

ERROR CODE	MEANING
PvmOk	No Error
PvmSysErr	System Error
PvmBadParam	Bad Parameter
PvmMismatch	Barrier Count Mismatch
PvmTooLong	Name Too Long (32 char max)
PvmNoData	Read Past End of Buffer
PvmNoHost	No Such Host
PvmNoFile	No Such Executable
PvmNoComp	No Such Process Component
PvmNoMsg	No Message Available
PvmNoMem	Can't Get Memory
PvmAlready	Process Already Enrolled
PvmBadMsg	Can't decode message

Obtaining PVM

To receive a copy of PVM send email to `netlib@ornl.gov` and in the message type `send pvm_shar from pvm` . Send comments and questions to `pvm@msr.epm.ornl.gov` .

PVM Daemon

Syntax

```
pvmd [ -i ] hostfile
pvmd hostfile &
```

Host File Syntax

```
hostname [ options ]
...
```

Hostname "*" used to set defaults.

Comment lines begin with "#".

Options:

OPTION	MEANING	DEFAULT
`lo=`	Unix Loginname	Name on Master
`pw`	Ask for Password	Don't ask
`dx=`	Pvmd Executable	/tmp/pvm/pvmd
`ep=`	Component Path	$HOME/pvm/%

Multiple path elements may be separated by colons (":"). "%" in path expands to architecture string.