C Interface

Initialization

```
void Cblacs_pinfo   ( int *mypnum, int *nprocs )
void Cblacs_setup   ( int *mypnum, int *nprocs )
void Cblacs_get     ( int icontxt, int what, int *val )
void Cblacs_set     ( int icontxt, int what, int *val )
void Cblacs_gridinit( int *icontxt, char *order,            int nprow, int npcol )
void Cblacs_gridmap ( int *icontxt, int *pmap, int ldpmap, int nprow, int npcol )
```

Destruction

```
void Cblacs_freebuff( int icontxt, int wait )
void Cblacs_gridexit( int icontxt )
void Cblacs_abort   ( int icontxt, int errornum )
void Cblacs_exit    ( int doneflag )
```

Informational and Miscellaneous

```
void Cblacs_gridinfo( int icontxt, int *nprow, int *npcol, int *myprow, int *mypcol )
int  Cblacs_pnum    ( int icontxt,           int  prow, int  pcol )
void Cblacs_pcoord  ( int icontxt, int pnum, int *prow, int *pcol )
void Cblacs_barrier( int icontxt, char *scope )
```

Non-standard

```
void   Csetpvmtids ( int ntasks, int *tids )
double Cdcputime00 ( )
double Cdwalltime00( )
int    Cksendid    ( int icontxt,                int rdest, int cdest )
int    Ckrecvid    ( int icontxt,                int rsrc,  int csrc  )
int    Ckbsid      ( int icontxt, char *scope                         )
int    Ckbrid      ( int icontxt, char *scope, int rsrc,  int csrc    )
```

Sending

```
void C□gesd2d( int icontxt,                                                int m, int n, TYPE *A, int lda, int rdest, int cdest )

void C□gebs2d( int icontxt, char *scope, char *top,                        int m, int n, TYPE *A, int lda )

void C□trsd2d( int icontxt, char *uplo, char *diag,                        int m, int n, TYPE *A, int lda, int rdest, int cdest )

void C□trbs2d( int icontxt, char *scope, char *top, char *uplo, char *diag, int m, int n, TYPE *A, int lda)
```

Receiving

```
void C□gerv2d( int icontxt,                                                int m, int n, TYPE *A, int lda, int rsrc, int csrc )

void C□gebr2d( int icontxt, char *scope, char *top,                        int m, int n, TYPE *A, int lda, int rsrc, int csrc )

void C□trrv2d( int icontxt, char *uplo, char *diag,                        int m, int n, TYPE *A, int lda, int rsrc, int csrc )

void C□trbr2d( int icontxt, char *scope, char *top, char *uplo, char *diag, int m, int n, TYPE *A, int lda, int rsrc, int csrc )
```

Combine Operations

```
void C□gamx2d( int icontxt, char *scope, char *top, int m, int n, TYPE *A, int lda, int *RA, int *CA, int RCflag, int rdest, int cdest )

void C□gamn2d( int icontxt, char *scope, char *top, int m, int n, TYPE *A, int lda, int *RA, int *CA, int RCflag, int rdest, int cdest )

void C□gsum2d( int icontxt, char *scope, char *top, int m, int n, TYPE *A, int lda,                              int rdest, int cdest )
```

Definition of □

□is	Data operated on is	TYPE is
s	single precision real	float
d	double precision real	double
c	single precision complex	float
z	double precision complex	double
i	integer	int

Options

```
UPLO  = "Upper triangular", "Lower triangular";
DIAG  = "Non-unit triangular", "Unit triangular";
SCOPE = "All", "row", "column";
TOP   = (SEE DESCRIPTION BELOW).
```

Broadcast Topologies

```
TOP   = " " : System dependent default topology;
      = "I" : increasing ring;
      = "D" : decreasing ring;
      = "H" : hypercube (minimum spanning tree);
      = "S" : split-ring;
      = "F" : fully connected;
      = "M" : nodes divided into I increasing
              rings, where I is set with call
              to Cblacs_set;
      = "T" : tree broadcast with NBRANCHES = I,
              where I is set with call to
              Cblacs_set;
      = "1" : tree broadcast with NBRANCHES = 1;
      = "2" : tree broadcast with NBRANCHES = 2;
         ⋮
      = "9" : tree broadcast with NBRANCHES = 9.
```

Global Topologies

```
TOP   = " " : System dependent default topology;
      = "1" : tree gather with NBRANCHES = 1;
      = "2" : tree gather with NBRANCHES = 2;
         ⋮
      = "9" : tree gather with NBRANCHES = 9;
      = "T" : tree gather with NBRANCHES = I,
              where I is set with call to
              Cblacs_set;
      = "F" : Fully connected;
      = "H" : if rdest = -1, a specialized
              "leave on all" hypercube topology
              called bidirectional exchange is used.
              Otherwise, TOP = "1" is substituted.
```

Notation

Underlined parameters are output arguments. If a routine is underlined it is a function that returns a value. The prefix **p** usually stands for process. Other standard notations are:

```
GE - GENERAL       TR - TRAPEZOIDAL
SD - SEND          BS - BROADCAST/SEND
RV - RECEIVE       BR - BROADCAST/RECEIVE
GAMX - General element-wise Absolute value MAXIMUM
GAMN - General element-wise Absolute value MINIMUM
GSUM - General element-wise SUMMATION
```

Key Ideas:

A BLACS context is created via a call to either `Cblacs_gridinit` or `Cblacs_gridmap`. No routine requiring a context may be used until one of these routines has been called. Multiple calls to `Cblacs_gridinit` or `Cblacs_gridmap` result in the creation of new contexts. To preserve resources, the user should free unused contexts by calling `Cblacs_gridexit`. When all BLACS operations are done, a call to `Cblacs_exit` frees any remaining contexts, and shuts down all BLACS operations.
Please note that `Cblacs_gridinit` and `Cblacs_gridmap` accept system contexts as input. A default system context encompassing all available processes may be obtained by a call to `Cblacs_get`.

`Cblacs_set` can only be used to set the BLACS' message ID range before the creation of the first context. Subsequent calls will be ignored.

Topology Hints

Topologies allow the user to optimize communication patterns for a particular operation. If the user does not have a communication pipe to maintain, the default TOP = " " is recommended. For more details, examine the mosaic page or the papers referenced below.

References

R. Clint Whaley, LAPACK, Working Note 73, *Basic Linear Algebra Communication Subprograms: Analysis and Implementation Across Multiple Parallel Architectures*, Computer Science Dept. Technical Report CS-94-234, University of Tennessee, Knoxville, May, 1994. To receive a postscript copy, send email to netlib@ornl.gov and in the mail message type: `send lawn73.ps from lapack/lawns`.

To get the user's guide to the BLACS, send email to netlib@ornl.gov and in the mail message type: `send blacs_ug.ps from blacs`. (Can also be downloaded via mosaic).

Reference, examples, troubleshooting, downloading options and installation instructions are available on mosaic. The URL is `http://www.cs.utk.edu/~rwhaley/Blacs.html`

Send comments and questions to `blacs@cs.utk.edu`.

Basic Linear Algebra Communication Subprograms

Quick Reference Guide,
C Interface

Release 1.0

February 14, 1995

University of Tennessee

Obtaining the BLACS

mosaic: `http://www.cs.utk.edu/~rwhaley/Blacs.html`

ftp: `netlib2.cs.utk.edu`, directory `blacs/`

email: `netlib@ornl.gov` with the message `send index from blacs`