```
!=====================================================================!
!                                                                     !
!   Software Name : HEC-MW Library for PC-cluster                     !
!         Version : 2.1                                               !
!                                                                     !
!     Last Update : 2006/06/01                                        !
!        Category : Linear Solver                                     !
!                                                                     !
!            Written by Kengo Nakajima (Univ. of Tokyo)               !
!                                                                     !
!     Contact address :  IIS,The University of Tokyo RSS21 project   !
!                                                                     !
!     "Structural Analysis System for General-purpose Coupling        !
!      Simulations Using High End Computing Middleware (HEC-MW)"      !
!                                                                     !
!=====================================================================!

module m_hecmw_comm_f
contains

!C
!C***
!C*** hecmw_barrier
!C***
!C
      subroutine hecmw_barrier (hecMESH)
      use hecmw_util
      implicit none
      integer(kind=kint):: ierr
      type (hecmwST_local_mesh) :: hecMESH

      call MPI_BARRIER (hecMESH%MPI_COMM, ierr)
      end subroutine hecmw_barrier

      subroutine hecmw_scatterv_DP(sbuf, sc, disp, rbuf, rc, root, comm)
      use hecmw_util
```

```
37          integer(kind=kint) :: sc      !send counts
38          double precision   :: sbuf(sc) !send buffer
39          integer(kind=kint) :: disp    !displacement
40          integer(kind=kint) :: rc      !receive counts
41          double precision   :: rbuf(rc) !receive buffer
42          integer(kind=kint) :: root
43          integer(kind=kint) :: comm
44          integer(kind=kint) :: ierr
45
46          CALL MPI_scatterv( sbuf, sc, disp, MPI_DOUBLE_PRECISION, &
47                             rbuff, rc, MPI_DOUBLE_PRECISION, &
48                             root, comm, ierr )
49
50          end subroutine hecmw_scatterv_DP
51
52    !C
53    !C***
54    !C*** hecmw_allREDUCE
55    !C***
56    !C
57          subroutine hecmw_allreduce_DP(VAL,VALM,n,hec_op,comm )
58          use hecmw_util
59          implicit none
60          integer(kind=kint) :: n, hec_op,op, comm, ierr
61          double precision, dimension(n) :: VAL
62          double precision, dimension(n) :: VALM
63
64          select case( hec_op )
65          case ( hecmw_sum )
66              op = MPI_SUM
67          case ( hecmw_prod )
68              op = MPI_PROD
69          case ( hecmw_max )
70              op = MPI_MAX
71          case ( hecmw_min )
72              op = MPI_MIN
```

```
 73         end select
 74         call MPI_allREDUCE(VAL,VALM,n,MPI_DOUBLE_PRECISION,op,comm,ierr)
 75
 76         end subroutine hecmw_allREDUCE_DP
 77
 78         subroutine hecmw_allreduce_DP1(s1,s2,hec_op,comm )
 79         use hecmw_util
 80         implicit none
 81         integer(kind=kint) ::  hec_op, comm
 82         double precision :: s1, s2
 83         double precision, dimension(1) :: VAL
 84         double precision, dimension(1) :: VALM
 85         VAL(1) = s1
 86         VALM(1) = s2
 87         call hecmw_allreduce_DP(VAL,VALM,1,hec_op,comm )
 88         s1 = VAL(1)
 89         s2 = VALM(1)
 90         end subroutine hecmw_allreduce_DP1
 91   !C
 92   !C***
 93   !C*** hecmw_allREDUCE_R
 94   !C***
 95   !C
 96         subroutine hecmw_allreduce_R (hecMESH, VAL, n, ntag)
 97         use hecmw_util
 98         implicit none
 99         integer(kind=kint):: n, ntag, ierr
100         real(kind=kreal), dimension(n) :: VAL
101         real(kind=kreal), dimension(:), allocatable :: VALM
102         type (hecmwST_local_mesh) :: hecMESH
103
104         allocate (VALM(n))
105         VALM= 0.d0
106         if (ntag .eq. hecmw_sum) then
107           call MPI_allREDUCE                                          &
108        &       (VAL, VALM, n, MPI_DOUBLE_PRECISION, MPI_SUM,          &
```

```
109         &        hecMESH%MPI_COMM, ierr)
110          endif
111
112          if (ntag .eq. hecmw_max) then
113            call MPI_allREDUCE                                          &
114         &       (VAL, VALM, n, MPI_DOUBLE_PRECISION, MPI_MAX,          &
115         &        hecMESH%MPI_COMM, ierr)
116          endif
117
118          if (ntag .eq. hecmw_min) then
119            call MPI_allREDUCE                                          &
120         &       (VAL, VALM, n, MPI_DOUBLE_PRECISION, MPI_MIN,          &
121         &        hecMESH%MPI_COMM, ierr)
122          endif
123
124          VAL= VALM
125          deallocate (VALM)
126
127          end subroutine hecmw_allreduce_R
128
129          subroutine hecmw_allreduce_R1 (hecMESH, s, ntag)
130          use hecmw_util
131          implicit none
132          integer(kind=kint):: ntag
133          real(kind=kreal) :: s
134          real(kind=kreal), dimension(1) :: VAL
135          type (hecmwST_local_mesh) :: hecMESH
136          VAL(1) = s
137          call hecmw_allreduce_R(hecMESH, VAL, 1, ntag )
138          s = VAL(1)
139          end subroutine hecmw_allreduce_R1
140
141     !C
142     !C***
143     !C*** hecmw_allREDUCE_I
144     !C***
```

```
145   !C
146         subroutine hecmw_allreduce_I(hecMESH, VAL, n, ntag)
147         use hecmw_util
148         implicit none
149         integer(kind=kint):: n, ntag, ierr
150         integer(kind=kint), dimension(n) :: VAL
151         integer(kind=kint), dimension(:), allocatable :: VALM
152         type (hecmwST_local_mesh) :: hecMESH
153
154         allocate (VALM(n))
155         VALM= 0
156         if (ntag .eq. hecmw_sum) then
157           call MPI_allREDUCE                                          &
158        &       (VAL, VALM, n, MPI_INTEGER, MPI_SUM,                  &
159        &        hecMESH%MPI_COMM, ierr)
160         endif
161
162         if (ntag .eq. hecmw_max) then
163           call MPI_allREDUCE                                          &
164        &       (VAL, VALM, n, MPI_INTEGER, MPI_MAX,                  &
165        &        hecMESH%MPI_COMM, ierr)
166         endif
167
168         if (ntag .eq. hecmw_min) then
169           call MPI_allREDUCE                                          &
170        &       (VAL, VALM, n, MPI_INTEGER, MPI_MIN,                  &
171        &        hecMESH%MPI_COMM, ierr)
172         endif
173
174
175         VAL= VALM
176         deallocate (VALM)
177         end subroutine hecmw_allreduce_I
178
179         subroutine hecmw_allreduce_I1 (hecMESH, s, ntag)
180         use hecmw_util
```

```
181         implicit none
182         integer(kind=kint)::  ntag, s
183         integer(kind=kint), dimension(1) :: VAL
184         type (hecmwST_local_mesh) :: hecMESH
185
186         VAL(1) = s
187         call hecmw_allreduce_I(hecMESH, VAL, 1, ntag )
188         s = VAL(1)
189         end subroutine hecmw_allreduce_I1
190
191   !C
192   !C***
193   !C*** hecmw_bcast_R
194   !C***
195   !C
196         subroutine hecmw_bcast_R (hecMESH, VAL, n, nbase)
197         use hecmw_util
198         implicit none
199         integer(kind=kint):: n, nbase, ierr
200         real(kind=kreal), dimension(n) :: VAL
201         type (hecmwST_local_mesh) :: hecMESH
202         call MPI_BCAST (VAL, n, MPI_DOUBLE_PRECISION, nbase, hecMESH%MPI_COMM, ierr)
203         end subroutine hecmw_bcast_R
204
205         subroutine hecmw_bcast_R1 (hecMESH, s, nbase)
206         use hecmw_util
207         implicit none
208         integer(kind=kint):: nbase, ierr
209         real(kind=kreal) :: s
210         real(kind=kreal), dimension(1) :: VAL
211         type (hecmwST_local_mesh) :: hecMESH
212         VAL(1)=s
213         call MPI_BCAST (VAL, 1, MPI_DOUBLE_PRECISION, nbase, hecMESH%MPI_COMM, ierr)
214         s = VAL(1)
215         end subroutine hecmw_bcast_R1
216   !C
```

```
217   !C***
218   !C*** hecmw_bcast_I
219   !C***
220   !C
221         subroutine hecmw_bcast_I (hecMESH, VAL, n, nbase)
222         use hecmw_util
223         implicit none
224         integer(kind=kint):: n, nbase, ierr
225         integer(kind=kint), dimension(n) :: VAL
226         type (hecmwST_local_mesh) :: hecMESH
227         call MPI_BCAST (VAL, n, MPI_INTEGER, nbase, hecMESH%MPI_COMM, ierr)
228         end subroutine hecmw_bcast_I
229
230         subroutine hecmw_bcast_I1 (hecMESH, s, nbase)
231         use hecmw_util
232         implicit none
233         integer(kind=kint):: nbase, ierr,s
234         integer(kind=kint), dimension(1) :: VAL
235         type (hecmwST_local_mesh) :: hecMESH
236         VAL(1) = s
237         call MPI_BCAST (VAL, 1, MPI_INTEGER, nbase, hecMESH%MPI_COMM, ierr)
238         s = VAL(1)
239         end subroutine hecmw_bcast_I1
240   !C
241   !C***
242   !C*** hecmw_bcast_C
243   !C***
244   !C
245         subroutine hecmw_bcast_C (hecMESH, VAL, n, nn, nbase)
246         use hecmw_util
247         implicit none
248         integer(kind=kint):: n, nn, nbase, ierr
249         character(len=n) :: VAL(nn)
250         type (hecmwST_local_mesh) :: hecMESH
251
252         call MPI_BCAST (VAL, n*nn, MPI_CHARACTER, nbase, hecMESH%MPI_COMM,&
```

```
253        &                                                    ierr)
254         end subroutine hecmw_bcast_C
255
256   !C
257   !C***
258   !C*** hecmw_update_1_R
259   !C***
260   !C
261   !C    1-DOF, REAL
262   !C
263         subroutine hecmw_update_1_R (hecMESH, VAL, n)
264         use hecmw_util
265         use  hecmw_solver_SR_11
266
267         implicit none
268         integer(kind=kint):: n
269         real(kind=kreal), dimension(n) :: VAL
270         real(kind=kreal), dimension(:), allocatable :: WS, WR
271         type (hecmwST_local_mesh) :: hecMESH
272
273         n = hecMESH%n_node
274
275         allocate (WS(n), WR(n))
276         WS= 0.d0
277         WR= 0.d0
278         call hecmw_solve_SEND_RECV_11                                    &
279        &    ( n, hecMESH%n_neighbor_pe, hecMESH%neighbor_pe,             &
280        &      hecMESH%import_index, hecMESH%import_item,                 &
281        &      hecMESH%export_index, hecMESH%export_item,                 &
282        &      WS, WR, VAL , hecMESH%MPI_COMM, hecMESH%my_rank)
283         deallocate (WS, WR)
284
285         end subroutine hecmw_update_1_R
286
287   !C
288   !C***
```

```
289   !C*** hecmw_update_2_R
290   !C***
291   !C
292   !C    2-DOF, REAL
293   !C
294         subroutine hecmw_update_2_R (hecMESH, VAL, n)
295         use hecmw_util
296         use  hecmw_solver_SR_22
297
298         implicit none
299         integer(kind=kint):: n
300         real(kind=kreal), dimension(2*n) :: VAL
301         real(kind=kreal), dimension(:), allocatable :: WS, WR
302         type (hecmwST_local_mesh) :: hecMESH
303
304         n = hecMESH%n_node
305
306         allocate (WS(2*n), WR(2*n))
307         WS= 0.d0
308         WR= 0.d0
309         call hecmw_solve_SEND_RECV_22                                    &
310        &    ( n, hecMESH%n_neighbor_pe, hecMESH%neighbor_pe,             &
311        &      hecMESH%import_index, hecMESH%import_item,                 &
312        &      hecMESH%export_index, hecMESH%export_item,                 &
313        &      WS, WR, VAL , hecMESH%MPI_COMM, hecMESH%my_rank)
314         deallocate (WS, WR)
315
316         end subroutine hecmw_update_2_R
317
318   !C
319   !C***
320   !C*** hecmw_update_3_R
321   !C***
322   !C
323   !C    3-DOF, REAL
324   !C
```

```
325         subroutine hecmw_update_3_R (hecMESH, VAL, n)
326         use hecmw_util
327         use  hecmw_solver_SR_33
328
329         implicit none
330         integer(kind=kint):: n
331         real(kind=kreal), dimension(3*n) :: VAL
332         real(kind=kreal), dimension(:), allocatable :: WS, WR
333         type (hecmwST_local_mesh) :: hecMESH
334
335         n = hecMESH%n_node
336
337         allocate (WS(3*n), WR(3*n))
338         WS= 0.d0
339         WR= 0.d0
340         call hecmw_solve_SEND_RECV_33                                    &
341        &    ( n, hecMESH%n_neighbor_pe, hecMESH%neighbor_pe,             &
342        &      hecMESH%import_index, hecMESH%import_item,                 &
343        &      hecMESH%export_index, hecMESH%export_item,                 &
344        &      WS, WR, VAL , hecMESH%MPI_COMM, hecMESH%my_rank)
345         deallocate (WS, WR)
346
347         end subroutine hecmw_update_3_R
348
349   !C
350   !C***
351   !C*** hecmw_update_m_R
352   !C***
353   !C
354   !C    m-DOF, REAL
355   !C
356         subroutine hecmw_update_m_R (hecMESH, VAL, n, m)
357         use hecmw_util
358         use  hecmw_solver_SR_mm
359
360         implicit none
```

```
361         integer(kind=kint):: n, m
362         real(kind=kreal), dimension(m*n) :: VAL
363         real(kind=kreal), dimension(:), allocatable :: WS, WR
364         type (hecmwST_local_mesh) :: hecMESH
365
366         n = hecMESH%n_node
367
368         allocate (WS(m*n), WR(m*n))
369         WS= 0.d0
370         WR= 0.d0
371         call hecmw_solve_SEND_RECV_mm                                    &
372        &    ( n, m, hecMESH%n_neighbor_pe, hecMESH%neighbor_pe,          &
373        &      hecMESH%import_index, hecMESH%import_item,                 &
374        &      hecMESH%export_index, hecMESH%export_item,                 &
375        &      WS, WR, VAL , hecMESH%MPI_COMM, hecMESH%my_rank)
376         deallocate (WS, WR)
377
378         end subroutine hecmw_update_m_R
379
380   !C
381   !C***
382   !C*** hecmw_update_1_I
383   !C***
384   !C
385   !C    1-DOF, INTEGER
386   !C
387         subroutine hecmw_update_1_I (hecMESH, VAL, n)
388         use hecmw_util
389         use  hecmw_solver_SR_11i
390
391         implicit none
392         integer(kind=kint):: n
393         integer(kind=kint), dimension(n) :: VAL
394         integer(kind=kint), dimension(:), allocatable :: WS, WR
395         type (hecmwST_local_mesh) :: hecMESH
396
```

```
397         n = hecMESH%n_node
398
399         allocate (WS(n), WR(n))
400         WS= 0.d0
401         WR= 0.d0
402         call hecmw_solve_SEND_RECV_11i                                    &
403        &   ( n, hecMESH%n_neighbor_pe, hecMESH%neighbor_pe,               &
404        &     hecMESH%import_index, hecMESH%import_item,                    &
405        &     hecMESH%export_index, hecMESH%export_item,                    &
406        &     WS, WR, VAL , hecMESH%MPI_COMM, hecMESH%my_rank)
407         deallocate (WS, WR)
408
409         end subroutine hecmw_update_1_I
410
411   !C
412   !C***
413   !C*** hecmw_update_2_I
414   !C***
415   !C
416   !C    2-DOF, INTEGER
417   !C
418         subroutine hecmw_update_2_I (hecMESH, VAL, n)
419         use hecmw_util
420         use  hecmw_solver_SR_22i
421
422         implicit none
423         integer(kind=kint):: n
424         integer(kind=kint), dimension(2*n) :: VAL
425         integer(kind=kint), dimension(:), allocatable :: WS, WR
426         type (hecmwST_local_mesh) :: hecMESH
427
428         n = hecMESH%n_node
429
430         allocate (WS(2*n), WR(2*n))
431         WS= 0.d0
432         WR= 0.d0
```

```
433         call hecmw_solve_SEND_RECV_22i                                  &
434        &   ( n, hecMESH%n_neighbor_pe, hecMESH%neighbor_pe,             &
435        &     hecMESH%import_index, hecMESH%import_item,                 &
436        &     hecMESH%export_index, hecMESH%export_item,                 &
437        &     WS, WR, VAL , hecMESH%MPI_COMM, hecMESH%my_rank)
438         deallocate (WS, WR)
439
440         end subroutine hecmw_update_2_I
441
442   !C
443   !C***
444   !C*** hecmw_update_3_I
445   !C***
446   !C
447   !C    3-DOF, INTEGER
448   !C
449         subroutine hecmw_update_3_I (hecMESH, VAL, n)
450         use hecmw_util
451         use  hecmw_solver_SR_33i
452
453         implicit none
454         integer(kind=kint):: n
455         integer(kind=kint), dimension(3*n) :: VAL
456         integer(kind=kint), dimension(:), allocatable :: WS, WR
457         type (hecmwST_local_mesh) :: hecMESH
458
459         n = hecMESH%n_node
460
461         allocate (WS(3*n), WR(3*n))
462         WS= 0.d0
463         WR= 0.d0
464         call hecmw_solve_SEND_RECV_33i                                  &
465        &   ( n, hecMESH%n_neighbor_pe, hecMESH%neighbor_pe,             &
466        &     hecMESH%import_index, hecMESH%import_item,                 &
467        &     hecMESH%export_index, hecMESH%export_item,                 &
468        &     WS, WR, VAL , hecMESH%MPI_COMM, hecMESH%my_rank)
```

```
469         deallocate (WS, WR)
470
471         end subroutine hecmw_update_3_I
472
473   !C
474   !C***
475   !C*** hecmw_update_m_I
476   !C***
477   !C
478   !C    m-DOF, REAL
479   !C
480         subroutine hecmw_update_m_I (hecMESH, VAL, n, m)
481         use hecmw_util
482         use  hecmw_solver_SR_mmi
483
484         implicit none
485         integer(kind=kint):: n, m
486         integer(kind=kint), dimension(m*n) :: VAL
487         integer(kind=kint), dimension(:), allocatable :: WS, WR
488         type (hecmwST_local_mesh) :: hecMESH
489
490         n = hecMESH%n_node
491
492         allocate (WS(m*n), WR(m*n))
493         WS= 0.d0
494         WR= 0.d0
495         call hecmw_solve_SEND_RECV_mmi                                    &
496        &   ( n, m, hecMESH%n_neighbor_pe, hecMESH%neighbor_pe,            &
497        &     hecMESH%import_index, hecMESH%import_item,                   &
498        &     hecMESH%export_index, hecMESH%export_item,                   &
499        &     WS, WR, VAL , hecMESH%MPI_COMM, hecMESH%my_rank)
500         deallocate (WS, WR)
501
502         end subroutine hecmw_update_m_I
503
504   end module m_hecmw_comm_f
```

505
506