```
#include <std.h>
#include <mem.h>
#include <log.h>
#include <stdio.h>
#include "messaging.h"

extern unsigned int * MSG_pLocalPID;

#define MSG_LocalPID (* MSG_pLocalPID)

#pragma FUNC_EXT_CALLED (user_main);
void user_main(void){

   int * datain;
   int * datain2;
   int * datain3;
   int * dataout;
   int * databack1, * databack2, * databack3;
   int i,sum=0;
   int error;
   MSG_Thread h1,h2,h3,h4,h5,h6;

   LOG_printf(hMSG_DEBUG_LOG,"PID: %i", MSG_LocalPID);

   printf("PID: %i\n", MSG_LocalPID);

   /* allocate space for the int arrays*/
   /* User0 = Low ext SDRAM, not PCI writable */
   /* USer1 = High ext SDRAM, PCI writable */
   dataout = MEM_alloc(MSG_MEM_User0, 4000, 0);
   datain = MEM_alloc(MSG_MEM_User1, 4000, 0);
   datain2 = MEM_alloc(MSG_MEM_User1, 4000, 0);
   datain3 = MEM_alloc(MSG_MEM_User1, 4000, 0);
   databack1 = MEM_alloc(MSG_MEM_User1, 4000, 0);
   databack2 = MEM_alloc(MSG_MEM_User1, 4000, 0);
   databack3 = MEM_alloc(MSG_MEM_User1, 4000, 0);


   /* seed the random generator with the SDRAM timing register*/
   srand(* (unsigned int *) 0x180001c);

   MSG_Init();

   while(1) {

      for (i=0; i<1000; i++) {
         dataout[i]=rand();
         datain[i]=0;
         datain2[i]=0;
         datain3[i]=0;
         databack1[i]=0;
         databack2[i]=0;
         databack3[i]=0;
         }

      if (MSG_LocalPID == 0) {

         MSG_Send(2, dataout, 1000, & h4);
         MSG_Send(3, dataout, 1000, & h5);
         MSG_Send(5, dataout, 1000, & h6);
         MSG_Recv(2, datain, 1000, & h1);
         MSG_Recv(3, datain2, 1000, & h2);
         MSG_Recv(5, datain3, 1000, & h3);

      } else if (MSG_LocalPID == 2) {

         MSG_Send(0, dataout, 1000, & h4);
```

```
   MSG_Send(3, dataout, 1000, & h5);
   MSG_Send(5, dataout, 1000, & h6);
   MSG_Recv(0, datain, 1000, & h1);
   MSG_Recv(5, datain2, 1000, & h2);
   MSG_Recv(3, datain3, 1000, & h3);

} else if (MSG_LocalPID == 3) {

   MSG_Send(0, dataout, 1000, & h4);
   MSG_Send(2, dataout, 1000, & h5);
   MSG_Send(5, dataout, 1000, & h6);
   MSG_Recv(2, datain, 1000, & h1);
   MSG_Recv(0, datain2, 1000, & h2);
   MSG_Recv(5, datain3, 1000, & h3);

} else {

   MSG_Send(0, dataout, 1000, & h4);
   MSG_Send(2, dataout, 1000, & h5);
   MSG_Send(3, dataout, 1000, & h6);
   MSG_Recv(0, datain, 1000, & h1);
   MSG_Recv(2, datain2, 1000, & h2);
   MSG_Recv(3, datain3, 1000, & h3);

}


MSG_ThreadWait(& h1);
MSG_ThreadWait(& h2);
MSG_ThreadWait(& h3);

sum=0;
for (i=0; i<1000; i++)
   sum += dataout[i] - datain[i] + datain2[i] - datain3[i] ;

LOG_printf (hMSG_DEBUG_LOG,"\tResult:%i",sum);

printf ("\tResult:%i\n",sum);

MSG_ThreadWait(& h4);
MSG_ThreadWait(& h5);
MSG_ThreadWait(& h6);

MSG_ThreadFinalize(& h1);
MSG_ThreadFinalize(& h2);
MSG_ThreadFinalize(& h3);
MSG_ThreadFinalize(& h4);
MSG_ThreadFinalize(& h5);
MSG_ThreadFinalize(& h6);


if (MSG_LocalPID == 0) {

   MSG_Send(2, datain, 1000, & h4);
   MSG_Send(3, datain2, 1000, & h5);
   MSG_Send(5, datain3, 1000, & h6);
   MSG_Recv(2, databack1, 1000, & h1);
   MSG_Recv(3, databack2, 1000, & h2);
   MSG_Recv(5, databack3, 1000, & h3);

} else if (MSG_LocalPID == 2) {

   MSG_Send(0, datain, 1000, & h4);
   MSG_Send(3, datain3, 1000, & h5);
   MSG_Send(5, datain2, 1000, & h6);
   MSG_Recv(0, databack1, 1000, & h1);
   MSG_Recv(5, databack2, 1000, & h2);
```

```
   MSG_Recv(3, databack3, 1000, & h3);

} else if (MSG_LocalPID == 3) {

   MSG_Send(0, datain2, 1000, & h4);
   MSG_Send(2, datain, 1000, & h5);
   MSG_Send(5, datain3, 1000, & h6);
   MSG_Recv(2, databack1, 1000, & h1);
   MSG_Recv(0, databack2, 1000, & h2);
   MSG_Recv(5, databack3, 1000, & h3);

} else {

   MSG_Send(0, datain, 1000, & h4);
   MSG_Send(2, datain2, 1000, & h5);
   MSG_Send(3, datain3, 1000, & h6);
   MSG_Recv(0, databack1, 1000, & h1);
   MSG_Recv(2, databack2, 1000, & h2);
   MSG_Recv(3, databack3, 1000, & h3);

}

MSG_ThreadWait(& h1);
MSG_ThreadWait(& h2);
MSG_ThreadWait(& h3);
MSG_ThreadWait(& h4);
MSG_ThreadWait(& h5);
MSG_ThreadWait(& h6);

error=0;
for (i=0; i<1000; i++) {
   if (dataout[i] != databack1[i]) {
      LOG_printf (hMSG_DEBUG_LOG,"\t\tError in 1 @[%x]",i);
      error=1;
   }
   if (dataout[i] != databack2[i]){
      LOG_printf (hMSG_DEBUG_LOG,"\t\tError in 2 @[%x]",i);
      error=1;
   }
   if (dataout[i] != databack3[i]){
      LOG_printf (hMSG_DEBUG_LOG,"\t\tError in 3 @[%x]",i);
      error=1;
   }
}

if(error) break;


MSG_ThreadFinalize(& h1);
MSG_ThreadFinalize(& h2);
MSG_ThreadFinalize(& h3);
MSG_ThreadFinalize(& h4);
MSG_ThreadFinalize(& h5);
MSG_ThreadFinalize(& h6);


if (MSG_LocalPID == 0) {

   MSG_Send(2, datain, 1, & h4);
   MSG_Send(3, datain2, 1, & h5);
   MSG_Send(5, datain3, 1, & h6);
   MSG_Recv(2, databack1, 1, & h1);
   MSG_Recv(3, databack2, 1, & h2);
   MSG_Recv(5, databack3, 1, & h3);

} else if (MSG_LocalPID == 2) {
```

```
            MSG_Send(0, datain, 1, & h4);
            MSG_Send(3, datain3, 1, & h5);
            MSG_Send(5, datain2, 1, & h6);
            MSG_Recv(0, databack1, 1, & h1);
            MSG_Recv(5, databack2, 1, & h2);
            MSG_Recv(3, databack3, 1, & h3);

        } else if (MSG_LocalPID == 3) {

            MSG_Send(0, datain2, 1, & h4);
            MSG_Send(2, datain, 1, & h5);
            MSG_Send(5, datain3, 1, & h6);
            MSG_Recv(2, databack1, 1, & h1);
            MSG_Recv(0, databack2, 1, & h2);
            MSG_Recv(5, databack3, 1, & h3);

        } else {

            MSG_Send(0, datain, 1, & h4);
            MSG_Send(2, datain2, 1, & h5);
            MSG_Send(3, datain3, 1, & h6);
            MSG_Recv(0, databack1, 1, & h1);
            MSG_Recv(2, databack2, 1, & h2);
            MSG_Recv(3, databack3, 1, & h3);

        }

        MSG_ThreadWait(& h1);
        MSG_ThreadWait(& h2);
        MSG_ThreadWait(& h3);
        MSG_ThreadWait(& h4);
        MSG_ThreadWait(& h5);
        MSG_ThreadWait(& h6);
    }

}
```