```
////////////////////////////////////////
// The TPU task
////////////////////////////////////////
void TPU_Task() {
   mrapi_shmem_hndl_t sMem;   /* handle to shmem */
   mrapi_mutex_hndl_t sMem_mutex;
   char* sPtr;
   mrapi_key_t lock_key;
   size_t msgSize;
   mcapi_endpoint_t cntrl_endpt;
   mcapi_request_t r1;
   mcapi_status_t err;

   // init the system
   mcapi_initialize(TPU_NODE, &err);
   CHECK_STATUS(err);

   mrapi_initialize(AUTO_USE_CASE_DOMAIN_ID, TPU_NODE,
                    MRAPI_NULL, MRAPI_NULL,&mrapi_status);
   CHECK_STATUS(mrapi_status);

   cntrl_endpt =
        mcapi_create_endpoint(TPU_PORT_CNTRL, &err);
   CHECK_STATUS(err);

   // now get the shared mem ptr
   mcapi_msg_recv(cntrl_endpt, &sMem, sizeof(sMem),
                  &msgSize, &err);
   CHECK_STATUS(err);

   sPtr = (void*) mrapi_shmem_attach(sMem, &mrapi_status);
   CHECK_STATUS(mrapi_status);

   // ALL bootstrapping is finished, begin processing
   while (1) {

     // NOTE – get an MRAPI lock
     mrapi_mutex_lock(sMem_mutex, &lock_key, 0,
                      &mrapi_status);
     CHECK_STATUS(mrapi_status);

     // do something that updates shared mem
     sPtr[0] = 1;

    // NOTE – release the MRAPI lock
    void mrapi_mutex_unlock(sMem_mutex, &lock_key,
                            &mrapi_status);
   }
}
```