```c
/*
 * MCAPI 2.000 Automotive Use Case
 * tpu_task.c
 *
 */

#include "mcapi.h"
#include "automotive.h"

////////////////////////////////////////
// The TPU task
////////////////////////////////////////
void TPU_Task() {
   char* sMem;
   size_t msgSize;
   size_t nSize;
   mcapi_endpoint_t cntrl_endpt, cntrl_remote_endpt;
   mcapi_sclchan_send_hndl_t cntrl_chan;
   mcapi_request_t r1;
   mcapi_status_t err;
   mcapi_timeout_t timeout = 500;
   mcapi_param_t mcapi_parameters;
   mcapi_info_t mcapi_info;

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

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

   mcapi_endpoint_get_i(ENGINE_DOMAIN, CNTRL_NODE, CNTRL_PORT_TPU,
&cntrl_remote_endpt, &r1, &err);
   CHECK_STATUS(err);

   // wait on the remote endpoint
   mcapi_wait(&r1, &nSize, &err, timeout);
   CHECK_STATUS(err);

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

   // NOTE – connection handled by control task
   // open the channel
   mcapi_sclchan_send_open_i(&cntrl_chan, cntrl_endpt, &r1, &err);
   CHECK_STATUS(err);

   // wait on the open
   mcapi_wait(&r1,NULL,&err,timeout);
   CHECK_STATUS(err);

   // ALL bootstrapping is finished, begin processing
   while (1) {
   // do something that updates shared mem
   sMem[0] = 1;

   // send a scalar flag to cntrl process
```