```
   // indicating sMem has been updated
   mcapi_sclchan_send_uint8(cntrl_chan, (mcapi_uint8_t) 1, &err);
   CHECK_STATUS(err);
   }
}


/*
 * MCAPI 2.000 Automotive Use Case
 * sig_task.c
 *
 */

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

//////////////////////////////////////////
// The SIG Processing Task
//////////////////////////////////////////
void SIG_task() {
   mcapi_endpoint_t cntrl_endpt, cntrl_remote_endpt;
   mcapi_pktchan_send_hndl_t cntrl_chan;
   mcapi_request_t r1;
   mcapi_status_t err;
   size_t size;
   mcapi_param_t mcapi_parameters;
   mcapi_info_t mcapi_info;
   mcapi_timeout_t timeout = 500;

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

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

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

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

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

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


   // All boostrap is finished, now begin processing
   while (1) {
        // Read sensor & process signal
        SIG_DATA sDat;  // populate this with results

      // send the data to the control process
```