Debugging of OEM protocols on Linux/RaspberryPI platforms

Procedure for debugging of OEM protocols on Linux/RaspberryPI platforms

- Disable the autostart of the KOM process (in the process configuration using CNF) and turn off the KOM process (using Sysconsole).
- Modify the *Makefile* for the OEM protocol and add the "-g" flag to it so that the compiler includes debugging information. Example: change from
  CFLAGS= -I ../h
  to
  *CFLAGS=**-g** -I ../h*
- Recompile the OEM protocol and copy the resulting library to the *protdll* directory:
  cd /home/d2000/oem
  rm template.o template.so
  make
  cp template.so /opt/d2000/protdll/oem_prot11.so
- In the directory where the source codes of the OEM protocol are, start the debugger and set the breakpoint to a specific line/lines in the OEM protocol (e.g. within the *Init* or *ReadAllPoints* procedure)
  Confirm "y" that you want to activate the breakpoint when loading future shared libraries.
  cd /home/d2000/oem
  gdb /opt/d2000/bin/kom
  (gdb) ***break template.c:73***
  No symbol table is loaded. Use the "file" command.
  Make breakpoint pending on future shared library load? (y or [n]) ***y***
- Start the KOM process (the */F0* parameter is used to turn off the watchdog)
  run /F0
- The debugger stops on the specified line of the OEM protocol source line:
  [Switching to Thread 0x72f161a0 (LWP 3369)]
  Thread 14 "S204b" hit Breakpoint 1, ReadAllPoints (St=0x75847580)
  at template.c:73
  73 template.c: Permission denied.
  (gdb) ***bt***
  #0 ReadAllPoints (St=0x75847580) at template.c:73
  #1 0x00eb00b0 in ?? ()
  #2 0x006ae364 in ?? ()
  Backtrace stopped: previous frame identical to this frame (corrupt stack?)
- Continue debugging with the standard *gdb* debugger commands.

Related pages:

D2000 KomAPI
D2000 KomAPI - interface description
D2000 KomAPI - interface structures
D2000 KomAPI - interface functions
D2000 KomAPI - interface call-back functions