Date:	2019-11-25
Author:	Lukas Schmoll (IPG Solutions Engineering)
Release No.:	CM-8.0

How can the CarMaker executable be run in debug mode?

When writing c-code extensions you may find yourself in the situation of being able to successfully compile your code but still running into errors while running the simulation. In this case it can be very helpful to start CarMaker in debugging mode.
This article shows an easy to use approach on how to do so using the GDB debugger in CarMaker Office and on the XPack4.

Step by step instruction

Step 1: Uncomment debug flag in the Makefile

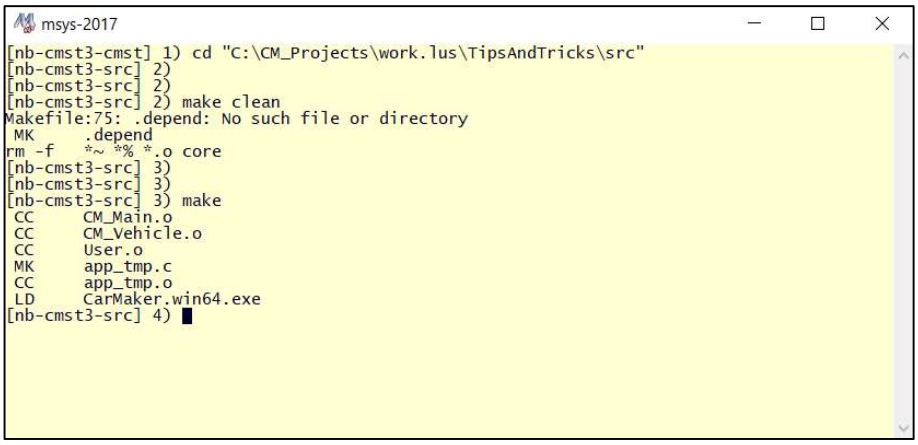
Navigate to the src folder of your project directory and open the Makefile with an editor. Enable the debug mode by uncommenting the following line:

```
21  OPT_CFLAGS =          -g -O1
```

Step 2: Recompiling the CarMaker executable

After the changes in the Makefile the CarMaker executable has to be rebuilt. The quickest way is to do this in a console. For windows users the MSYS console is delivered with the CarMaker installation. It is available under START > IPG > MSYS-2017

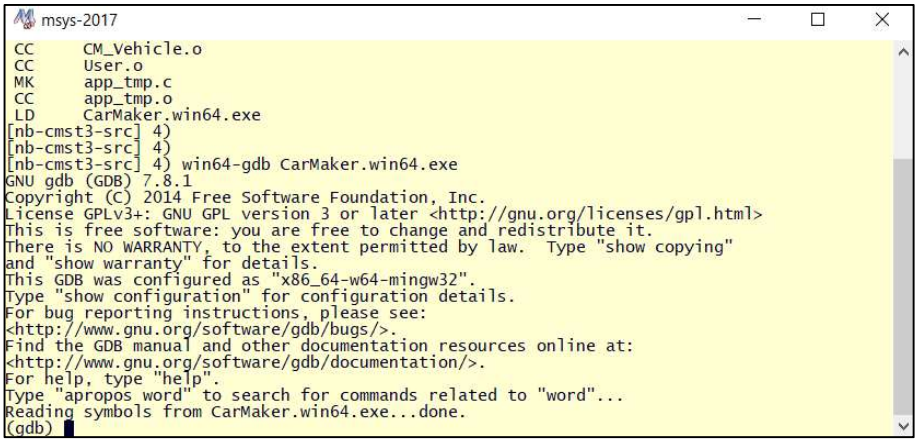
Navigate in MSYS to the src folder of your project directory. Enter the command “Make clean” first to make sure the old libraries are fully removed. Then enter the command “make” to start building the new executable.



Step 3: Start GDB debugger

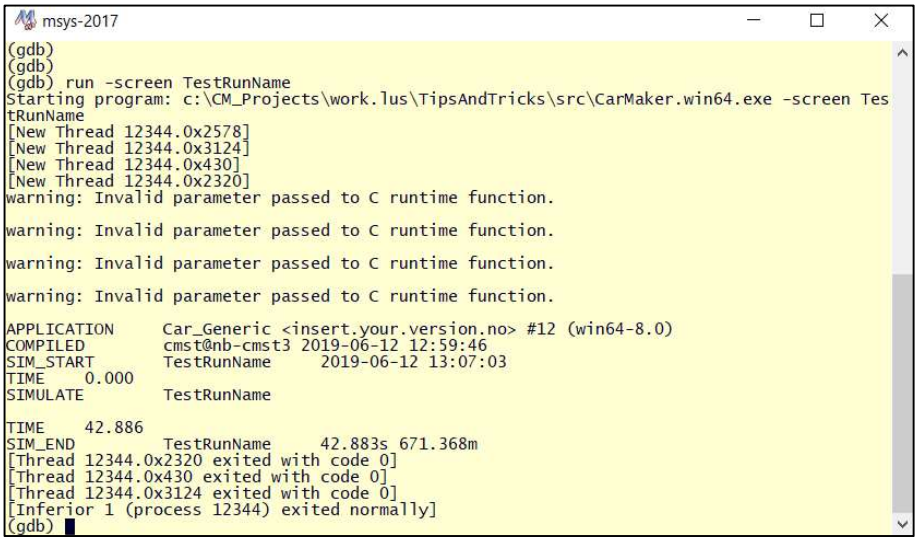
Start the previously built executable in debug mode using the command “win64-gdb CarMaker.win64.exe”.

The corresponding command for Linux is “gdb CarMaker.linux64”.



Step 4: Start the simulation

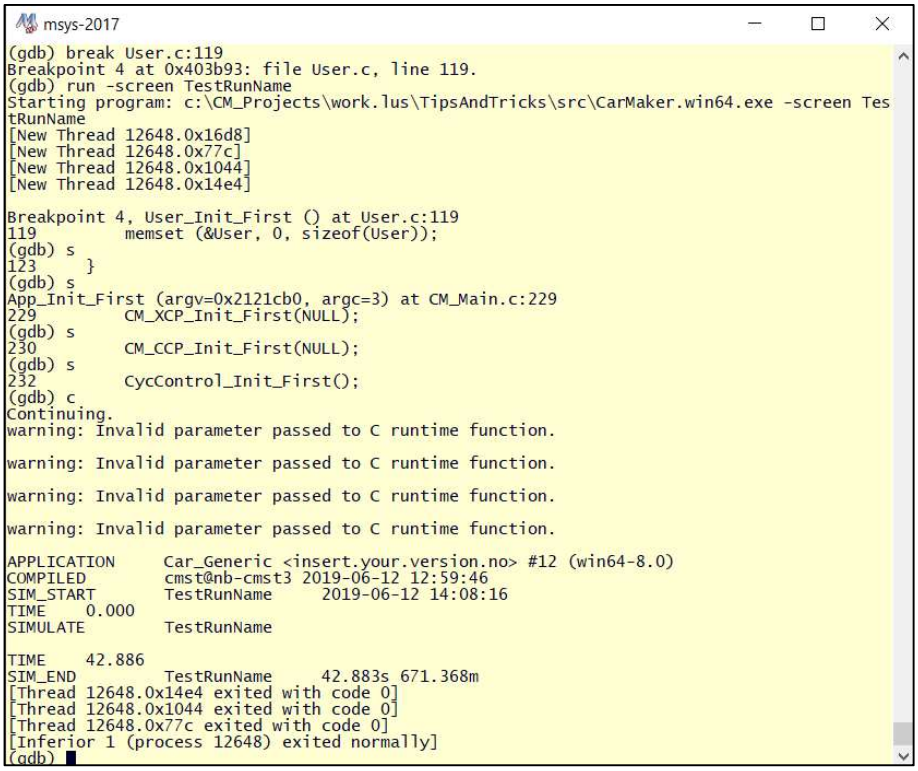
Start the simulation using the command “run -screen <Name of your TestRun>”. The option -screen enables a log output in the console, which is very useful.



Useful commands and options

In order to determine where exactly your code crashes, you can set a breakpoint in any source code. This has to be done after step 3 using the command “break <filename.c>:<line>”, e.g. “break User.c:119”. After starting the simulation with the command explained in step 4, the program will progress until the given line in the given source code. From now on you can progress line by line using the command “s” or continue the simulation until the next breakpoint using the command “c”.

A full list of available commands can be found using the “help” command or in the internet, e.g. https://darkdust.net/files/GDB%20Cheat%20Sheet.pdf



Using the GDB debugger on XPack4

In case you are using CarMaker/HIL in combination with XPack4 hardware, you can start the gdb debugger directly on the real-time system.

In this case you have to access the real-time system after rebuilding the executable using the telnet command in your console:

telnet <name of your realtime system>
login: hil
pw: CarMaker

After logging into the real-time system and navigating to the src folder of your project directory, start the executable in debug mode using the command “gdb CarMaker.xeno” corresponding to step 3.

For step 4 two important additional options have to be used. The first is to set the option “-constdt”. This is necessary to prevent a warning or an error message coming up, which would terminate the simulation. This would of course be the case as the cycle time would exceed 1 ms, when the executable stops at a breakpoint. However it does not totally disable cycle time measurements in the background. All TCPU quantities for the user to measure the cycle times for the different modules are still available.

Also all IO flags activated in the source code have to be specified as an option while running the simulation. IO demoapp serves as an example. The command will then be: “run -constdt -screen -io demoapp <Name of your TestRun>”.

In case no IO flag is used, you still have to use the options “-constdt -io none”.