Date:	2017-12-15
Author:	Valerie Bensch, CarMaker Service Team Germany
Release No.:	CarMaker 6.0.x

Introduction to CarMaker's C-code interface

We often say that CarMaker is an "open integration platform". That means, the user can connect CarMaker to other tools, replace component models with his/her own models and add his/her own algorithms for additional modules that doesn't exist in CarMaker yet. Today I want to focus to the last use case and introduce you to the usage of CarMaker's C-code interface. Dealing with this topic for the first time, you might feel a little insecure about where to place your code. So let me tell you how I usually proceed.

The structure of CarMaker's C-code interface

All files that are part of CarMaker's C-code interface can be found in the folder "src" of your CarMaker project. If this folder doesn't exist, you can add it by updating the project folder and selecting the option "Source / Build environment". If you want to use the C-code interface in combination with CarMaker for Simulink, your working environment is the folder "src_cm4sl" instead of "src". This folder contains several C-code files. The most important file of the C-code interface is *User.c*. Please don't change anything in *CM_Main.c* or *CM_Vehicle.c* without consulting with CarMaker Service Team in prior.

User.c contains a lot of functions where you can add your own code. Each function is called at a specific point of the CarMaker simulation:

- Once during the initialization of the virtual vehicle environment (after clicking "Application > Start & Connect" or when starting the first simulation of a session)
- Before each TestRun
- At each step of the CarMaker main cycle (compare to Programmer's Guide section "The main cycle explained")
- After each TestRun
- Once during the shutdown of the virtual vehicle environment (when closing CarMaker)
- At special events (e.g. when receiving or sending an APO message)

Please refer to the comments inside *User.c* to learn to which group each function belongs.

Modify global variables during the simulation

As an example, I want to add wind gusts acting on the vehicle.

In Reference Manual section "User Accessible Quantities", I can see that there is a global CarMaker variable for the external wind velocities that I can modify:

Name UAQ	Name C-Code	Name CM4SL	Unit	Info
...				
Env.WindVel_ext.x Env.WindVel_ext.y Env.WindVel_ext.z	Env.WindVel_ext[0] Env.WindVel_ext[1] Env.WindVel_ext[2]	-	m/s	Additional user defined wind in inertial frame

Alternatively, the global CarMaker variable names can be found in the header files that are available in the folder “include” of the CarMaker installation (e.g. C:\IPG\hil\win32-6.0.4\include\Environment.h). Let’s take the positive part of a sinus wave as wind gust velocity in direction of the global frame y-axis:

```
1: Env.WindVel_ext[1] = 20 * sin(SimCore.Time);
2: Env.WindVel_ext[1] = M_MAX(Env.WindVel_ext[1], 0);
```

But where should I place these lines of code? The wind is meant to change during the simulation. So I should place it somewhere in CarMaker’s main cycle where the code is executed once in each simulation cycle (each millisecond). In addition, I want to see the effect of the wind gust on the vehicle immediately – that means, in the same simulation cycle. Therefore I need to choose a function that is called before the vehicle model and even before the environment module that sums up all wind velocities to a total wind velocity. So, *User_in()* is the best choice. *User_in()* and other functions that are called in each simulation step, include the following lines

```
1: if (SimCore.State != SCState_Simulate)
2:   return;
```

If the code is added after these lines, it is executed only during the simulation, whereas if the code is placed above of them, it is executed during the simulation, in the preparation phase, during post-processing and in between two simulations. My wind gusts are supposed to apply during the simulation only (otherwise it could be difficult to find the vehicle’s equilibrium state in the preparation phase), so I place it after the lines mentioned above.

Now I can rebuilt a new CarMaker virtual vehicle environment including my calculations as described in Programmer’s Guide section “Rebuilding the CarMaker Simulation Program”.

Initialize variables with values read from the Infofile

Now, I want to extend my example in a way that the amplitude of my wind gusts is not fixed to 20m/s, but can be modified via the Additional Parameters of a vehicle data set. So I change my code inside *User_In()* to

```
1: Env.WindVel_ext[1] = Wind_amplitude * sin(SimCore.Time);
2: Env.WindVel_ext[1] = M_MAX(Env.WindVel_ext[1], 0);
```

Programmer’s Guide section “C-Code Command Reference” describes some useful, CarMaker specific C-code functions, e.g. *iGetXXX()* for reading values from an infofile. That is exactly what I’m looking for. Once again, the question is: Where should I place this line of code? Of course, I could add it to *User_In()*, too, but reading information from a file is relatively time consuming and there is no benefit in doing it every millisecond because I already know that it won’t change during the TestRun (remember: it is a parameter, not an User Accessible Quantity). That is why I’m looking for a function that is called once at TestRun start and after all other Infofile parameters have been read (e.g. the currently selected vehicle, so that the vehicle Infofile handle *SimCore.Vhcl.Inf* already is up to date). *User_TestRun_Start_atEnd()* meets these requirements.
Finally I declare my C-code variable *Wind_amplitude* at the top of *User.c*, e.g.

```
1: Wind_amplitude = iGetDblOpt(SimCore.Vhcl.Inf,"Env.Wind.amplitude", 0);
```

and rebuilt the executable to test my extension.

```
1: tUser User; //already existing code
2: double Wind_amplitude;
```

Create a new User Accessible Quantity connected to a C-code variable

It would be nice, if I could trigger the wind gusts via an User Accessible Quantity (UAQ) e.g. inside a Realtime Expression. So, I declare an additional variable *Wind_trigger* at the top of User.c

```
1: double Wind amplitude;
2: int Wind_trigger;
```

and I modify my code in *User_In()* so that *End.WindVel_ext[1]* follows half of a sinus wave as soon as *Wind_trigger* is set to one and remains zero afterwards.

To be able to access *Wind_trigger* from outside the C-code interface (e.g. from Realtime Expressions), it needs to be connected to an UAQ. In Programmer's Guide chapter "C-Code Command Reference", I find a suitable CarMaker specific function for this task, too:

```
1: DDefInt (NULL, "Wind.trigger", "", &Wind_trigger, DVA_IO_In);
```

UAQ need to be declared only once during initialization. My example always is active. It doesn't belong to a model that I select or deselect for a TestRun. So, it is sufficient to declare the UAQ at the initialization of CarMaker's virtual vehicle environment. Actually, there is a special function *User_DeclQuants()* for this

```
1: static int t0=0;
2: …
3: if (Env.WindVel ext[1]<=0.0001) {
4:     if (Wind trigger<0.5) {
5:         t0 = SimCore.Time; // set offset for sinus
6:     } else {
7:         Wind trigger = 0; // reset trigger after half of the sinus wave
8:     }
9: }
10:
11: Env.WindVel ext[1] = Wind amplitude * sin(SimCore.Time-t0);
12: Env.WindVel_ext[1] = M_MAX(Env.WindVel_ext[1], 0);
```

purpose in User.c.
Now, the only thing left to do is setting the initial value of *Wind_trigger*.

```
1: Wind_trigger = 0;
```

Actually, there are two possible positions for this initialization – at TestRun start e.g. in *User_TestRun_Start_atEnd()* or at the start of the virtual vehicle environment e.g. in *User_Init()*. They lead to a slightly different behavior in case the UAQ *Wind.trigger* is set to 1 **before** simulation start.

- If I only initialize *Wind_trigger* in *User_Init()*, I can see the wind at simulation start.
- If I initialize *Wind_trigger* in *User_TestRun_Start_atEnd(),* my TestRun always starts without wind gust.

You can find a version of *User.c* for CarMaker 6.0.4 inlcuding this example in the download package.