Date:	2019-11-20
Author:	Chandra Sekar Venkataramani, IPG Solutions Engineering, Germany
Release No.:	CM-6.x to CM-8.x

Improving portability and code exchange of C-code extensions

This is a guide on integrating user's C-code extensions. The example focuses primarily on best practices for extending the CarMaker folder structure, the steps to improve portability across versions and ease code exchange by compiling the C-code extension as a library

Technical Background

CarMaker as an open-integration platform provides users the possibility to extend the source code, either in the *src* or *src_cm4sl* folder of a CarMaker project, to obtain a custom project-specific application.

Over the project lifecycle, it may be necessary to

- to update CarMaker to a newer version, after which it is typically necessary to repeat the user's C-code integration. Depending on the approach followed, this task of repeating the C-code integration could be tedious, time-consuming and error-prone.
- to share code extensions with project partners/customers without providing the source code.

This approach also allows easily distinguishing between development & end-use environments and to generate libraries for different target architectures (Linux, Windows).

Solution

The following steps provide an outline of a best practice that can address both these points on portability and code exchange. They are explained with a sample implementation that prints the local time at the instant of the function call.

Step 1: After creating a CarMaker project with Sources/Build Environment, manually create a sub-folder, *src_UserCode*. Unpack the contents of the .zip (i.e. MyTimer.c, MyTimer.h and Makefile) into this folder.

Step 2: Perform 'make' and 'make install' in <ProjDir>/src_UserCode folder

- make - compiles the user code into a library file
- make install – makes copies of user code's library and header files to the *lib* and *include* folders of CM project directory

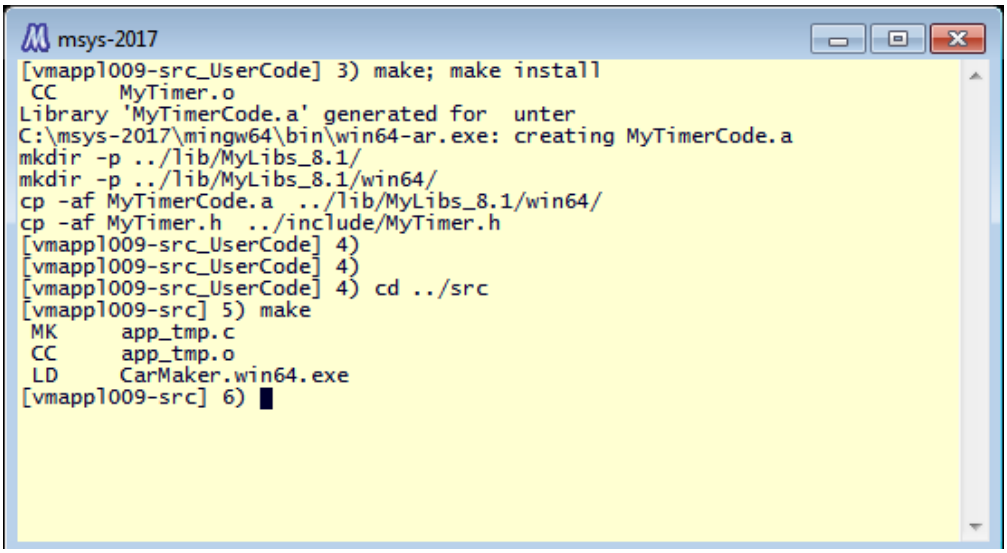

Figure 1: Compiling user-code as library and linking it to new CarMaker.exe

Step 3: Add code snippets to User.c and Makefile in *src* folder. This typically involves

- including header files in User.c

```
# include "../include/MyTimer.h"
```

- adding function calls to user code in User.c (e.g. Init, DeclQuants, Calc)

```
int
User_Init (void)
{
	MyTimer_Init();
	return 0:
}
```

```
void
User_DeclQuants (void)
{
	MyTimer_DeclQuants();
}
```

```
int
User_TestRun_Start_atEnd (void)
{
	MyTimer_TestRun_Start_atEnd();
	return 0;
}
```

```
int
User_Calc (double dt)
{
	if (SimCore.State! =SCState_Simulate)
	return 0;

	if (SimCore.CycleNo%10000==0) {
		Log("MyTimer_Calc() is called here\n");
		MyTimer_Calc();
	}

	return 0;
}
```

- adding the previously compiled lib to the list of libraries to be linked in Makefile

```
LD_LIBS += ../lib/MyLibs_8.1/$(ARCH)/MyTimerCode.a
```

The version number <ver> in sub-folder 'MyLibs_<ver>' is required, only if the code has a dependency to the CarMaker version. Else, it is also sufficient to have a version-independent sub-folder 'MyLibs'

Now, perform 'make' in <ProjDir>/src folder to generate a new CarMaker executable

To update to a newer CarMaker version, just update the CM version number in Makefile of src_UserCode folder and repeat Steps 2 & 3.

To share code, it is sufficient to share the final CarMaker project folder without the src_UserCode.