# CACC Software Installation Instructions

## Scope

This document is aimed at software developers with knowledge of and access to Simulink and dSPACE software packages. It also assumes that the developer has a functional development environment including Simulink, dSPACE, an appropriate C compiler for their operating system, and a cross-compiler for the MicroAutobox II architecture. This document aims to provide instructions to allow the developer to successfully compile and install the CACC software from a starting point of downloading the source code from the Git repository. For the installation of the log processing tool it is also assumed the developer has a basic understanding of Linux and Python.

## CARMA Platform & CACC Algorithm

### Compilation

The CARMA Platform interface and the CACC Algorithm software are distributed as Simulink model files grouped into a Simulink project. As both of these software packages are distributed in separate repositories the first step is to check out both into a local directory. Merge the “src” folder of the CACC-2015 repository into the “src” folder of the CarmaPlatform-MAB repository by copy and pasting it. Everything from here will refer to this directory that contains the contents of both repositories. Open the CarmaPlatform.prj file with Simulink to load the appropriate .m and .mat files into the workspace. Add the “for-\_all\_3\_veh” folder to the Simulink path. Several Simulink S-Functions must be compiled before the whole model can be compiled, this can be accomplished with the built-in Simulink MEX tool.

The follow S-Functions must be compiled:

* src/
  + platform/adapter/c\_src
    - ParseBSM.mexw64
      * “mex ParseBSM.c ParseBSM\_wrapper.c der\_decoder.c bsm\_parser.c”
    - EncodeBSM.mexw64
      * “mex EncodeBSM.c EncodeBSM\_wrapper.c der\_encoder.c bsm\_encoder.c”
  + project/
    - toggle\_sw.mexw64
      * “mex toggle\_sw.c”
    - switch\_RC.mexw64
      * “mex switch\_RC.c”
    - sw\_contr2.mexw64
      * “mex sw\_contr2.c”
    - sw\_contr.mexw64
      * “mex sw\_contr.c”
    - run\_dist.mexw64
      * “mex run\_dist.c”
    - gap\_change.mexw64
      * “mex gap\_change.c”
    - dist\_contr\_n.mexw64
      * “mex dist\_contr\_n.c”
    - dist\_contr.mexw64
      * “mex dist\_contr.c”
    - coordination.mexw64
      * “mex coordination.c”
    - cc\_v\_ref.mexw64
      * “mex cc\_v\_ref.c”

Once the S-Functions are compiled, open the “src/project/for-\_all\_3\_veh/CACC\_system\_struct.mdl” file and compile it using Simulink coder.

### Installation

To load the compiled artifacts onto a vehicle first you must connect to the MicroAutobox II with the “HostPC” ethernet port on the device. Create a new dSPACE project (or open an existing one) and create a new experiment under it. Add the MicroAutobox II hardware as a platform for the experiment. Load the .SDF file created by the Simulink compilation process under the “bld” directory as a Variable Description for the Experiment. To load the .SDF file to the flash memory on the MicroAutobox II right click the .SDF file and select “Install to Flash” option. The real-time-application should start automatically.

## Log Processing Tool

The log processing tool is distributed as a simple, standalone Python script file. This file is found in the CarmaPlatform-SecondaryProcessor repository under “/linux\_host/src/python/carma\_udp\_logger.py”. Copy this script to the secondary processor in the vehicle. Start the script with “python carma\_udp\_logger.py”. If connected remotely to the vehicle (via SSH or otherwise) the script can be left running after exit by using “nohup python carma\_udp\_logger.py &” instead.