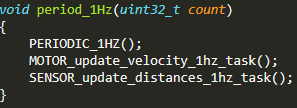
SCons Build Automation

TL;DR

Use SCons in the Git repository to automatically build, isolate modules in firmware, and generate a DBC header file.

Problem

Since we are using a single project for multiple ECUs with different functionality, we will encounter conflicts.



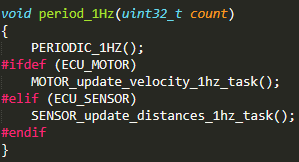
In foxp2/L5\_application/periodic\_scheduler/period\_callbacks.cpp

In the snippet above, the motor team may want to add a function call to the 1 Hz context to update motor velocity, but the sensor team may also want to update distances in the same 1 Hz context.

The problem is that an ECU responsible for motors does not have access to sensor peripherals; meanwhile, the sensor ECU does not have access to motors.

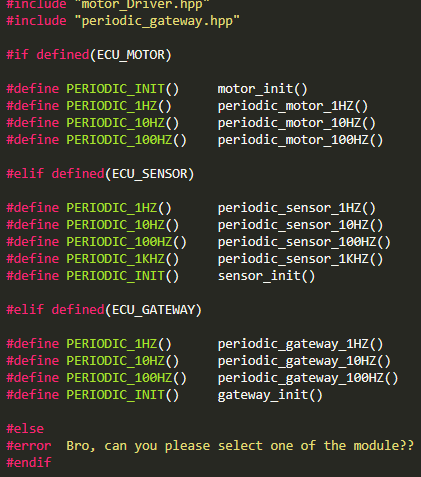
Approach

To resolve the problem, we should be using MACROS or feature flags to isolate and disable tasks primarily by removing function calls from the compiled firmware.

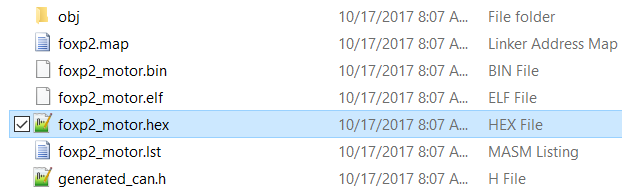


In foxp2/L5\_application/periodic\_scheduler/period\_callbacks.cpp

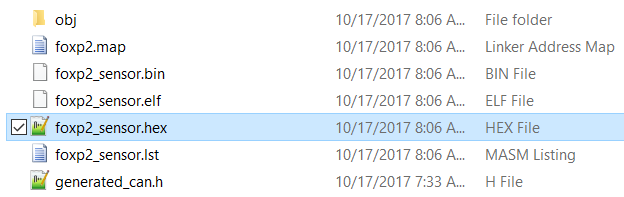
In addition, we should add one more layer of abstraction instead of the approach above.



In foxp2/L5\_application/FOXP2/Periodic\_Dispatcher.hpp



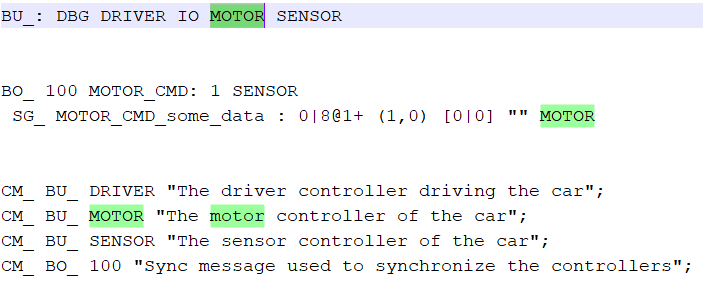
Compiled firmware in foxp2/build/motor/foxp2\_motor.hex



Compiled firmware in foxp2/build/sensor/foxp2\_sensor.hex

We also need to specify the name of the ECU for the DBC parser Python script to generate the DBC represented header file. This step needs to be done before compilation.

**> python dbc\_parse.py -i 243.dbc -s {NODE\_NAME} > generated\_dbc.h**



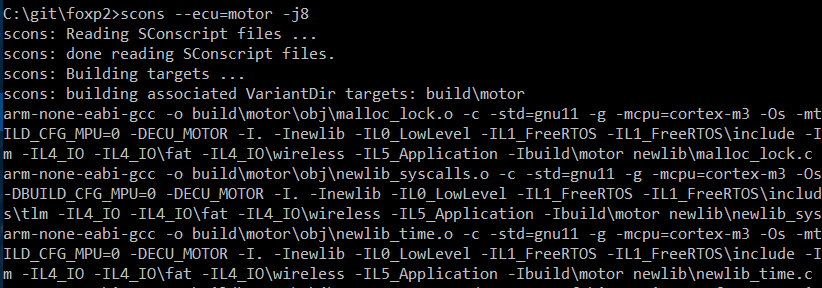
Proposed Solution

SCons, a C builder tool (like Make but more awesome), is now integrated to the foxp2 Git repository.

Foxp2: <https://gitlab.com/foxp2/foxp2>

To build firmware, navigate to the repository foxp2 directory and invoke SCons.





In the example, SCons will do the following:

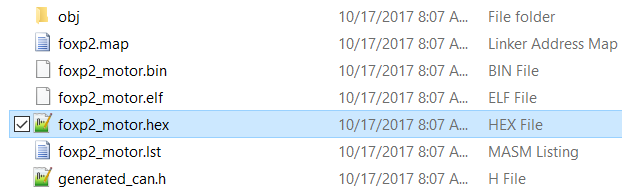
1. Build firmware and store it in foxp2/build/motor
2. Invoke the DBC parser and store the generated header in foxp2/build/motor

In the example, the flag --ecu=motor will do the following:

1. Define ECU\_MOTOR in firmware
2. Define DBC node as MOTOR

The flag -j<n> allows you to build firmware using multiple CPU cores.

A successful build will produce artifacts in a newly created build directory.



Compiled firmware in foxp2/build/motor/foxp2\_motor.hex

Setup – Linux

SCons



ARM toolchain



Setup - Windows

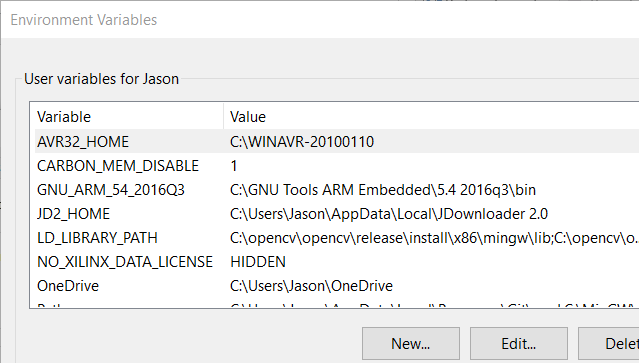
Download SCons

* <http://scons.org/>

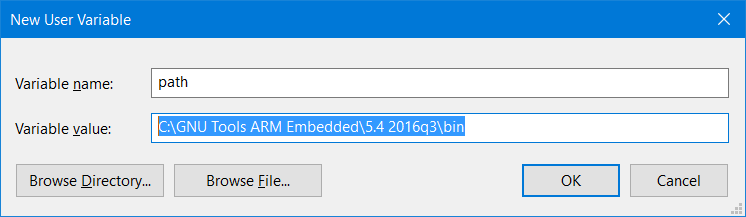
Download ARM GCC

* <https://developer.arm.com/open-source/gnu-toolchain/gnu-rm/downloads>

SCons also needs a path to the ARM toolchain



Environment variable for Windows



Setting up the environment variable **path** in Windows

You can download a new copy of the ARM toolchain

* C:\GNU Tools ARM Embedded\5.4 2016q3\bin

Or you can steal Preet’s copy of the ARM toolchain from the SJSU\_Dev package

* C:\SJSU\_Dev\toolchain\bin