Control with AMDC Using Simulink Autogen

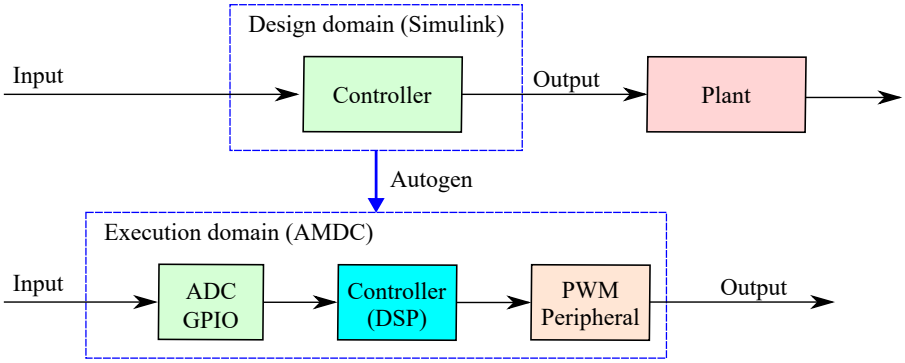
Background

Simulink provides a MATLAB-based graphical environment for modeling and simulating control systems. It is extensively used to model, simulate, and analyze complex dynamical systems, including motor drives. The GUI and block diagram environment in Simulink make it user-friendly and easy to validate system performance and controller performance.

To apply the controller built in Simulink to the AMDC platform, we need Automatic Code Generation (Autogen) that can convert the Simulink block diagrams into an equivalent C code for an embedded system. It enables developers to simulate and validate control strategies before deploying them to hardware, which reduces implementation errors and provides a more intuitive framework for control system design.

Control Approach with Simulink and AMDC

The figure below shows the Simulink + AMDC workflow. The Simulink model represents the control logic, while the AMDC is responsible for executing this logic at a fixed time interval using real sensor data.



The Simulink model is typically structured into three subsystems:

1. **Input/Output (I/O):** Used for simulation and visualization only
2. **Plant:** Represents the physical system (used for simulation)
3. **Controller:** Contains the control logic to be deployed

Only the **controller subsystem** is converted into embedded C code.

Then, the Simulink + AMDC workflow separates control development into two domains:

- **Design domain (Simulink):**
  The control algorithm is developed and validated using a graphical model.

Skip to content **›main (AMDC):**
d C code is executed in real time on the embedded controller.

Recommended Workflow

The recommended workflow for developing control code is:

1. Develop and validate the control algorithm in Simulink.
2. Convert the controller to an atomic subsystem and then to a referenced model.
3. Generate C code using Simulink Autogen.
4. Integrate the generated code into the AMDC project.
5. Execute and validate on hardware.

Important Considerations for Simulink Models

For successful development and integration of control code, the following considerations must be observed:

1. **Discrete-Time Implementation**: All blocks within the controller must be discrete-time, since the AMDC executes control logic at fixed sampling intervals.
2. **Fixed-Step Solver**: The Simulink model must use a fixed-step solver to ensure compatibility with real-time execution.
3. **Consistent Sample Time**: The entire controller subsystem should operate at a single, well-defined sample time before converting to an atomic subsystem and creating a referenced model.
4. **Code Generation Settings**: The code generation target should be set to Embedded Coder (`ert.tlc`). The build configuration should enable “Generate Code Only”.
5. **Referenced Model Usage**: The controller subsystem should be converted to an atomic subsystem, then converted to a referenced model. Any updates to model settings should be performed after opening the referenced model as the top model.
6. **AMDC Integration Details**
   - All generated source (`*.c`) and header (`*.h`) files must be included in the AMDC project.
   - The autogenerated folder must be added to the compiler include paths.
   - Do not delete any generated files, as some auxiliary files may be required during compilation.
7. **File and Path Constraints**
   - File paths must not contain whitespace.
   - The path to the folder containing the MATLAB script and the Simulink model, as well as any parent directory, must not include whitespace. Otherwise, it will result in a build error.

This organization ensures that the autogenerated controller code is correctly compiled and integrated into the AMDC firmware.

Example Model

An example tutorial, Tutorial: Autogen, is provided to demonstrate the Simulink Autogen workflow for control implementation and its integration with the AMDC.

Skip to content

Conclusion

The Simulink Autogen workflow provides a structured and efficient approach for implementing control algorithms on the AMDC. By separating control design from embedded implementation, this approach enables:

- Rapid development and iteration
- Improved reliability through simulation
- Clear mapping between design and execution

This methodology is recommended for developing advanced control systems on the AMDC platform.