From MATLAB to Embedded C

By Grant Martin and Houman Zarrinkoub

The flexibility to explore concepts is essential to innovation. Yet an idea that cannot be implemented in an actual system remains just that—an idea. In a typical development process, many engineers, scientists, and researchers rely on the flexibility of the MATLAB® language to explore designs, but for system implementation, they must translate their high-level MATLAB algorithms into the lower-level language of C.

Embedded MATLAB™ code enables a MATLAB to C workflow that brings exploration and implementation together. Embedded MATLAB is a subset of the MATLAB technical computing language that lets you generate C code directly from MATLAB algorithms. With this capability, teams can maintain a single design source and use one language in one development environment from concept to implementation.

Closing the Gap Between Concept and Implementation

To build an embedded system, engineers typically follow a familiar paradigm: develop in MATLAB, translate to C, and compile for the target hardware, verifying at each step. During the translation to C, typical modifications include:

- Optimizing to accommodate processor speed and memory limitations
- Introducing appropriate numerical representations (floating or fixed point)
- Incorporating real-time data management, such as buffering, streaming, and pipelining

Manually translating MATLAB code to C poses several challenges. First, the design and implementation teams use different tools and development environments, making it difficult to communicate complex algorithmic concepts. Second, the C code often diverges from the MATLAB concept code. This means that additional steps must be taken to verify that the concept code and the C code remain equivalent and that no manual coding errors were introduced during development.

In addition, it is often necessary to write C-code test harnesses to construct and download tables and other data structures on the processor. Writing and maintaining

The Embedded MATLAB subset includes more than 270 MATLAB operators and functions, as well as functions from Signal Processing Toolbox™, Aerospace Toolbox, and Fixed-Point Toolbox™. It includes MATLAB language features such as matrices and arrays, real and complex numbers, structures, flow control, and subscripting.
MATLAB code for a Kalman filter with three states (inset) and equivalent C code generated with the Embedded MATLAB subset. The 17 lines of MATLAB code translate to 144 lines of C code, including comments.
They are used in a wide range of applications, including control, signal and image processing, radar and sonar, and financial modeling.

The simple tracking algorithm is developed in MATLAB in a file named `kalmanf.m` (Figure 1).

This algorithm can be automatically translated to C using `emlc`, a command-line tool in Real-Time Workshop® that generates C code from Embedded MATLAB code.

Typical syntax for translation is:
```
> emlc -s cfg -eg { A, B, C, Q, R, u, t, yv }...
kalmanf.m
```

The configuration parameter (following the `-s` delimiter) is used to specify code generation properties and fine-tune the compiler. These options can be specified using the GUI shown in Figure 2.

The example option (following the `-eg` delimiter) sets the data types and dimensions of function variables by specifying an example at the function interface. The generated C code is created in a file named `kalmanf.c`.

In MATLAB, the algorithm is expressed in 17 compact lines of code based on matrix operations. The equivalent C code uses `for` loops for scalar computations of the matrix operations and comprises 144 lines.

Because data types for all variables of the algorithm were introduced at compile time, the designer has the flexibility to develop multiple versions of the algorithm destined for different processors. The generated C code is readable, and it contains the same comments as the original MATLAB code, inserted at corresponding lines of the algorithm. This means that you only need to maintain a single copy in MATLAB for further design iterations. The C code is generated automatically, reflecting any changes.

The plot in Figure 3 shows the results of the generated code. These results were verified as equivalent to the MATLAB simulation code.

To illustrate the connection between concept and implementation, let us look at a typical tracking algorithm implemented in MATLAB.

Generating C Code from Embedded MATLAB Code

Kalman filters are recursive filters that estimate the state of a linear dynamic system from a series of noisy measurements.

These custom applications is cumbersome, takes time, and consumes engineering resources. With Embedded MATLAB code, the embedded algorithms and data structures that you implemented in MATLAB can be automatically translated to C.

To illustrate the connection between concept and implementation, let us look at a typical tracking algorithm implemented in MATLAB.
Incorporating MATLAB Code into Simulink and Stateflow Models

For many teams, the job is complete once the generated C code has been verified against the elaborated MATLAB code and the compiled algorithms are deployed to the target. If the algorithm will be part of a larger, more complex system, you can extend the workflow by placing precompiled algorithmic function blocks developed using the Embedded MATLAB subset in your Simulink® models or Stateflow® charts (Figure 4). You can then perform system-level simulations and generate C or HDL code for implementation.

Embedded MATLAB serves as a common thread between MATLAB, Simulink, and Stateflow. In this way, it streamlines the workflow between algorithm developers and system modelers and brings exploration and implementation together.

Figure 3. Output from the MATLAB and C-code versions of the Kalman filter.

Figure 4. Simulink radar system model with the Embedded MATLAB function block referencing the Kalman tracking filter.

Resources

WEBINAR: Algorithm Design and Code Generation with Embedded MATLAB
www.mathworks.com/nn/wnbn30822