Deploy Generated C/C++ Code

Step 4 of 4 in Code Generation Guide: Generate Deployable C/C++ Code

Once you have verified that the generated code behaves according to your requirements, you can deploy it. You can deploy the generated code as source code, static libraries, dynamic libraries, or executable applications.

To package the required source files to build your application on an external platform, use the `packNGo` function.

To build executable applications with the code generator, you must:

1. Generate the required source code or library (completed in previous steps).

2. Create a main function that calls the generated code.

3. Iterate through the code generation process with build type set to `Executable (.exe)`.

Edit Generated Main Function and Interfaces

To create an application, create or use a C/C++ main function to call the C/C++ entry-point functions generated from your MATLAB$^{®}$ functions. For example, see Generating Standalone C/C++ Executables from MATLAB Code.

Note

Do not modify the files `main.c` and `main.h` in the `examples` subfolder. Before using the example main function, copy the example main source and header files to a location outside of the build folder. Modify the files in the new location to meet the requirements of your application.

Use the generated example main as a starting point for creating a main function. The example main provides a clear example for how to pass input to and output from the generated code. For more information and examples, see Incorporate Generated Code Using an Example Main Function and Structure of Generated Example C/C++ Main Function.

Generated Function Interfaces

To write a main function, you must be familiar with the generated function interfaces. See Mapping MATLAB Types to Types in Generated Code.

C/C++ entry-point functions follow these conventions:

- Pass input arrays by reference.

- Return output arrays by reference.
- Pass input scalars by value.
- Return scalars by value for single-output functions.
- Return scalars by reference:
  - For functions with multiple outputs.
  - When you use the same variable as input and output.

If you use the same variable as input and output in your MATLAB code, the generated code passes the scalar by reference. See Avoid Data Copies of Function Inputs in Generated Code.

Array Definition

The code generator creates C/C++ array definitions that depend on the array element type and their memory allocation type. See Representation of Arrays in Generated Code.

To learn more about the methods associated with arrays in the generated code, see:

- Use C Arrays in the Generated Function Interfaces
- Use Dynamically Allocated C++ Arrays in Generated Function Interfaces

Initialize and Terminate Functions

Your C/C++ code must call an initialize function and a terminate function that are generated in addition to your C/C++ entry-point functions. By default, the generated C/C++ entry-point function calls the initialize function. The generated example main function calls the terminate function. As you create and edit your own main function, make sure to call both initialize and terminate functions.

For more information, see Use Generated Initialize and Terminate Functions.

Build Executable Applications by Using MATLAB Coder

After you create a main file (`main.c`) and main header file (`main.h`), follow these steps to build executable applications by using the app:

1. Open the **Generate Code** page in the app.
2. Set **Build type** to `Executable (.exe)`.
3. Click **More Settings**.
4. On the **Custom Code** tab, in **Additional source files**, enter `main.c`
5. On the **Custom Code** tab, in **Additional include directories**, enter the location of the modified `main.c` and `main.h` files. For example, `c:\myfiles`. Click **Close**.
6. To generate the executable, click **Generate**.

The app indicates that code generation succeeded.

7. Click **Next** to go to the **Finish Workflow** step.

8. Under **Generated Output**, you can see the location of the generated executable `filename.exe`.

After generating code and writing a main file that uses the generated code, you can generate executable applications with the code generator or other build tools. If you want to transfer the generated code to a target platform in an exportable zip file, use the `packNGo` function.

Target Specific Code Generation

To deploy your code to another platform, use the hardware support packages that support generating and building binary code for that platform.

In the MATLAB Coder™ app, during the **Generate Code** step, select a hardware support package from the **Hardware Board** drop-down list.

For a list of support packages provided for MATLAB Coder, see MATLAB Coder Supported Hardware. If you want to specify a custom toolchain for build that is not available from a hardware support package, you can register your own toolchain. See Custom Toolchain Registration.

Tips

Choose Hardware at the Command Line

At the command line, specify a hardware support package by using the `coder.hardware` function.

Run Executables in MATLAB

To run the executable in MATLAB on a Windows® platform:

```
system('filename.exe')
```

See Also

`coder.hardware` | `packNGo`

Related Topics

- Deploy Generated Code
- Generating Standalone C/C++ Executables from MATLAB Code
- Incorporate Generated Code Using an Example Main Function
- Structure of Generated Example C/C++ Main Function
- Mapping MATLAB Types to Types in Generated Code
- Avoid Data Copies of Function Inputs in Generated Code

- Representation of Arrays in Generated Code
- Use C Arrays in the Generated Function Interfaces
- Use Dynamically Allocated C++ Arrays in Generated Function Interfaces
- MATLAB Coder Supported Hardware
- Custom Toolchain Registration

Code Generation Guide: Generate Deployable C/C++ Code