Prepare MATLAB Code for Code Generation

Step 1 of 4 in Code Generation Guide: Generate Deployable C/C++ Code

To generate C/C++ code, the code generator converts dynamically typed MATLAB® code to statically typed C/C++ code. In dynamically typed languages, the class, size, and complexity of a given variable can change at run time. In contrast, statically typed languages must determine variable types at compile-time.

Before generating code, identify which function to generate code for. This function is called the entry-point function or primary function. To prepare your code for code generation:

1. Initialize variables for code generation.
2. Screen your code for unsupported functions and language features.

Initialize Variables for Code Generation

Because the generated code is statically typed, initialize all variables in your code before use to allow the code generator to identify and allocate the variables properly in the generated code. To identify some of these issues, include this line in your code:

```
%#codegen
```

This table lists some common errors that might occur when you initialize variables in code intended for code generation.

Original Code	Issue	Modified Code
`y = zeros(1,10);` `y(3) = 1 + 2i;`	y is defined as double but assigned complex double value.	`y = complex(zeros(1,10));` `y(3) = 1 + 2i;`

`for i = 1:N`
`  y(i,i) = i;`
`end`
	The array y is extended dynamically without being defined.	`y = zeros(N,N);`
		`for i = 1:N`
		`  y(i,i) = i;`
		`end`

For information about data definition for code generation of specific data types, see Data Definition Considerations for Code Generation and Best Practices for Defining Variables for C/C++ Code Generation.

Screen Code for Unsupported Functions and Language Features

The code generator supports most language features and functions. See Functions and Objects Supported for C/C++ Code Generation. To check for unsupported functions and language features in your code:

1. Start the MATLAB Coder™ App from the **Apps** tab. Alternatively, enter this at the command line:

```
>> coder
```

2. Enter the entry-point function name in the app. Do not add subfunctions in this step. The code generator automatically includes the required subfunctions.
3. To see the unsupported functions or language features in your code, open the Code Generation Readiness Tool Report by clicking **Next**.

Alternatively, call the `coder.screener` function on your entry-point function. At the command line, enter:

```
coder.screener('filename');
```

This function parses your code and highlights unsupported MATLAB functions and some unsupported language features. See `coder.screener`.

If your code includes unsupported functions, consider these workarounds:

- Check for replacement functions and System objects that support code generation.
- Write custom code for the unsupported functions.
- Use `coder.ceval` to call custom C functions that replace the unsupported functions.
- Use `coder.extrinsic` to call the unsupported functions.

For more details of these workarounds, see Resolve Error: Function Is Not Supported for Code Generation.

Tips

Set Advanced Code Generation Options at Command Line

Use the `codegen` function with the configuration object `coder.config`. Depending on the type of build, you can also use `coder.CodeConfig`, `coder.EmbeddedCodeConfig`, and `coder.MexCodeConfig`.

Research Code Generation Considerations for Specific Functions

The reference pages of functions supported for code generation contain a section titled *Extended Capabilities*. This section lists special considerations when generating code for those functions. For example, see *Extended Capabilities* in `interp2`.

Call `coder.extrinsic`

Calls to `coder.extrinsic` declare a function as an extrinsic function. The code generator does not produce code for the body of the extrinsic function and instead uses the MATLAB engine to execute the call.

See Also

`coder.target` | `coder.screener` | `coder.ceval` | `coder.extrinsic` | `codegen` | `coder.config` | `coder.CodeConfig` | `coder.EmbeddedCodeConfig` | `coder.MexCodeConfig`

Related Topics

- Data Definition Considerations for Code Generation
- Best Practices for Defining Variables for C/C++ Code Generation
- MATLAB Language Features Supported for C/C++ Code Generation
- Use MATLAB Engine to Execute a Function Call in Generated Code

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