Use Generated Initialize and Terminate Functions:

When generating C/C++ code from MATLAB® code, the code generator automatically produces two housekeeping functions, initialize and terminate. The initialize function initializes the state on which the generated C/C++ entry-point functions operate. It must be called before you call the entry-point functions for the first time. The terminate function frees allocated memory and performs other cleanup operations. It must be called after you call the entry-point functions for the last time.

Initialize Function:

The name of the generated initialize function is primary function name initialize, where primary function name is the name of the first MATLAB entry-point function that you specify while generating code. The initialize function initializes the state on which the generated entry-point functions operate. The initialize function can include:

1. Calls to supporting code for nonfinite data (Inf and NaN). These calls are generated if your MATLAB code contains operations that can generate nonfinite values.
2. Code that initializes global or persistent variables.
3. Custom code for creating an initial state that you specify. To include custom code in the initialize function, do one of the following:

- In a code configuration object, set CustomInitializer to a character vector that contains the custom code.

- In the MATLAB Coder™ app, on the Custom Code tab, specify custom code for the initialize function.

Calling Initialize Functions:

If you generate a MEX function, the generated code automatically includes a call to the initialize function. If you generate standalone code, there are two possible situations:

1. By default, if the initialize function is nonempty, the code generator includes a call to the initialize function at the beginning of the generated C/C++ entry-point functions. The generated code also includes checks to make sure that the initialize function is called automatically only once, even if there are multiple entry-point functions. In this situation, you do not need to manually call the initialize function.
   If the initialize function is empty, the generated C/C++ entry-point functions do not include a call to the initialize function.
2. You can choose to not include a call to the initialize function in the generated entry-point functions. Do one of the following:

- In a coder. CodeConfig or coder.EmbeddedCodeConfig object, set RunInitializeFcn to false.
- In the MATLAB Coder app, on the All Settings tab, set Automatically run the initialize function to No.

If you make this choice, you must manually call the initialize function before you call a generated entry-point function for the first time. Not calling the initialize function causes the generated entry-point functions to operate on an invalid state.

If you generate C++ code with a class interface, then the code generator produces a class constructor and destructor that perform initialization and termination operations. You do not need to manually call the initialize and terminate functions. See Generate C++ Code with Class Interface.

Examples of Generated Initialize Functions:

Examples of MATLAB code patterns and the corresponding generated initialize functions:

Your MATLAB code uses global or persistent variables. For example, define this MATLAB function:

```
function y = bar
global g
y = g;
end
```

Generate a static library for bar. Specify the initial value of g as 1.

```
codegen -config:lib -globals {'g',1} bar
```

The code generator produces the file `bar_initialize.c` in *work*`\codegen\lib\bar`, where *work* is the folder that contains `bar.m`. The function `bar_initialize` initializes the global variable `g`.

```
void bar_initialize(void)
{
  g = 1.0;
  isInitialized_bar = true;
}
```

The generated C function `bar` includes a call to `bar_initialized`. It uses the boolean `isInitialized_bar` to make sure that the initialize function is called automatically only once.

```
double bar(void)
{
  if (!isInitialized_bar) {
    bar_initialize();
  }

  return g;
}
```

Your MATLAB code contains an operation that can generate nonfinite values (Inf or NaN). For example, define a MATLAB function foo that calls factorial. The factorial function grows quickly and returns Inf for inputs greater than a certain threshold. For an input of type double, the threshold is 170. Executing factorial(171) in MATLAB returns Inf.

```
function y = foo(a)
y = factorial(a);
end
```

Generate a static library for foo.

```
codegen -config:lib foo -args {1}
```

The code generator produces the file foo_initialize.c in work\codegen\lib\foo, where work is the folder that contains foo.m. The function foo_initialize calls supporting code for nonfinite data, rt_InitInfAndNaN, which is defined in another generated file rt_nonfinite.c.

```
void foo_initialize(void)
{
  rt_InitInfAndNaN();
  isInitialized_foo = true;
}
```