Creating Standalone Matlab Applications (v. 1.2)

By Dr. C. S. Tritt
January 4, 2012

Reusing Code

- A major theme in programming over the past 20 years is the convenient reuse of code (effort).
- Mathwork provides several ways in which your custom Matlab functions and scripts (code) can be reused.
- One of the simplest is the conversion of the code into a standalone Windows executable (a.k.a. an app).

2

Related Products & Uses

- **Compiler (mcc) – Compile and package your Matlab programs as Royalty-free standalone applications and shared libraries.**
- Complier – Also allows you to create and distribute C & C++ applications that use Matlab functions.
- Coder (and Embedded Coder) – Generate human readable C and C++ code from Matlab code. It generates portable source code.

3

Related Products (cont)

- External Interfaces – Integrate custom C code with Matlab and call Matlab from C and Fortran programs.
- Builder JA – Create Java classes from your Matlab programs.
- Builder NE – Create .NET and COM components from your Matlab programs.
- Builder EX – Package your Matlab programs as Excel add-ins.
- Spreadsheet Link EX – Use Matlab from within Excel.

4

About Applications

- Modern technology users expect sophisticated "applications" rather than simple programs.
- They want to click on an icon and complete a task, rather than think about all the necessary "behind the scenes" details.
- Engineers and programmers generally still need to think about the details.

5

Application Anatomy

- Modern applications are typically made of code (often in more than one language), graphics and other software components.
- While some of this code is in the application file, much of it is typically in separate "library" files possibly shared by other applications (for example *lib* and *dll* files).

6

Machine Instructions

- Applications ultimately consist of a sequence of machine instructions being executed by a real or virtual machine.
- Software compiling and linking tools convert human readable code and related graphical files into a form that can be executed on a device.
- The Matlab Compiler is such a tool.

7

What the Compiler Does

- Determines the Matlab functions for the final application.
- Generates the C or C++ interface (wrapper) code to create the executable.
- Encrypts all the files into a single, compressed file.
- Compiles the C or C++ interface code into object code.
- Links the object files and libraries to create an encrypted executable (.exe) file.

8

…but there's a catch

- While the users of your application need not have Matlab installed on their computer, they do need a collection of libraries called the Matlab Compiler Runtime (MCR):
  - It contains all necessary functions, etc., needed by Matlab programs.
  - It can be packaged with your application for easy installation (the MCR only needs to be installed once on a given computer).

9

On My Computer

- It appears that the Matlab Compiler and the related *mbuild* tool and MCR library were all automatically installed and configured correctly on my laptop. Previously this was a major process.
- The following slides will determine if you were as lucky.
- If not, some help can be found in Matlab help at *Matlab Compiler > Getting Started > Before You Use…*, etc.

10

The Simple Approach

- Be sure the Matlab current folder is set to the one containing your program (script or function).
- Run the Matlab compiler (*mcc*) in the Matlab Command window.
- You will need to specify the main .m file to compile and typically also need to specify compiler options and may need to specify other files to be included in the "build."

11

An Example

- I've created a very simple sample program for testing and demonstrating the compiler:

```
% CompilerToy.m
%
% This script demonstrates the creation and use of standalone
% Matlab applications. It prompts the user to enter two
% numbers and displays their sum.
%
% Created by Dr. C. S. Tritt
% Last revised 1/2/12 (v. 1.0)
x = input('Enter a number: ');
y = input('Enter another number: ');
disp(['Their sum is ' num2str(x+y)]);
```

12

Your First Standalone

- Enter or download CompilerToy.m.
- Run it once normally to confirm the current folder is correct set and that the script works as expected.
- Then enter:

  mcc -mv compilerToy.m
- (I'll explain the *–mv* option later)
- Wait. Compilation and linking is a slow process even for very simple scripts.

13

Running Your Program

- In addition to spewing output to the command window, the Compiler should have also produced a *compilerToy.exe* file in the same folder as the source (.m) file.
- This program can be run by double clicking on it or creating a desktop shortcut that refers to it.
- The following slide shows it running…

14

Sample Command Output

- Note the "ugly" MS-DOS command window.
- The first run may be very slow.
- Window closes immediately when program completes.

15

Not Good Enough

- Most modern users would find a command window (or line) interface unacceptable.
- Matlab programs with fully graphical user interfaces can be compiled.
- Run *mcc* with the *–e* option in place of *–m* to create an application that doesn't automatically open a MS-DOS command window.

16

Another Example

- I've created a script, zip2grade2.m, that will package your code and documentation into a single zip file to be submitted for grading.
- It uses only graphic (dialog box) user I/O, so it doesn't need a command window.
- It is available at http://people.msoe.edu/~tritt/ge4200/examples.html).

17

zip2grade2 Setup

- Download the *zip2grade2* script and sample configuration file (z2g2.zip) and unzip them in your Matlab working directory (one level above your source code and documentation folders).
- Edit the configuration file to *z2gconfig.txt* include your name and documentation folder name.
- Open, study and run *zip2grade2.m*.

18

Eliminating the DOS Window

- Once you have *zip2grade2* working in the Matlab environment, compile it using:

  mcc -e zip2grade2.m
- Create a desktop shortcut to *zip2grade2.exe* (right click on the desktop and select *New* > *Shortcut* and complete the dialog).
- Double click on the shortcut to run the program.

19

mcc Options

- The *mcc* command/function has numerous options.
- Its full syntax: mcc [-options] mfile1 [mfile2 ... mfileN] [C/C++file1 ... C/C++fileN]
- Multiple options can be specified (as in –mv) and they are case sensitive.
- See *Help* > *Matlab Compiler* > *Functions* > *Command-line Tools* > *mcc* for more information.

20

Some mcc Options

- *-B* specifies a bundle file that can contain compiler options and file names. Also consider using *deploytool*.
- *-d* specifies a directory for output.
- *-e* suppress MS-DOS command window (as opposed to -m).
- *-g* include debugging information in generated executable file.
- *-l* generate function library (as opposed to a standalone application.

21

More mcc Options

- *-m* generate a standalone application (with MS-DOS command window for user I/O; alternative is -e).
- *-o* specify output file name.
- *-R* specify run-time options like log file name and startup message.
- *-v* generate verbose output.
- *-W* specify wrapper function type (main, C library, C++ library).
- *-win32* build a 32-bit Windows app.

22

deploytool

- *deploytool* is a Matlab script that simplifies the process of compiling and distributing Matlab scripts as standalone applications.
- It creates a project (.prj) file containing project configuration information (files to include, features to activate or disable, etc.).
- See *Help > Matlab Compiler > Comand-line Tools > deploytool* and *deploytool > Help* for more information.

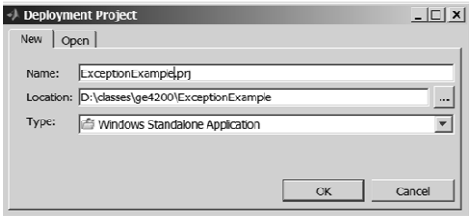
23

deploytool Example

- Have a program ready to deploy (compile, etc.). The *ExceptionExample.m* script works well.
- The process is demonstrated on the following screens.
- Note that by default the executable is placed in the ProjectName\distrib folder.

24

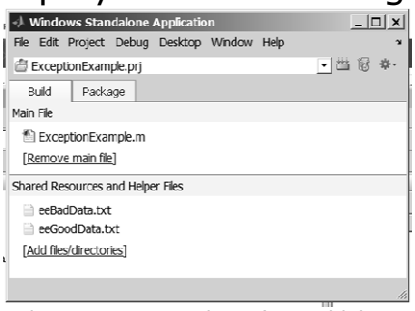
deploytool Startup Dialog



25

deploytool Main Dialog

Un-docked



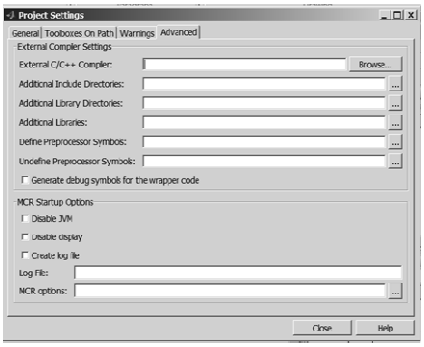
Select *Project* > *Package* (or *Build* then *Package*) to compile and package your application for distribution.

Note that only the main script or function and non-.m resource files (like data and configuration files) must be specified.

26

deploytool Project Settings

I've not investigated these options. My point is for you to know they exist.



27

deploytool Notes

- I've not fully investigated Matlab Compiler and *deploytool* behavior.
- *deploytool* creates a "ProjectName" folder containing *src* and *distrib* subfolders.
- *deploytool* appears to package data files and/or path information into the *_Install.bat* and/or *ProjectName.exe* files.

28

MCR Installer

- There is now an MCRInstalle.exe application that appears to be able to automatically install and configure the MCR files on a computer that needs them.
- I've not fully investigated its operation (it seems mostly automatic).
- I note that the use of the compiler and MCR would likely get complicated when compiling for different operating systems and multiple languages.

29