MATLAB Compiler Release Notes

The "MATLAB Compiler 3.0.1 Release Notes" on page 1-1 describe two important bug fixes added in the latest version of the MATLAB Compiler.

If you are upgrading from a release earlier than Release 13, you should also see:

- "MATLAB Compiler 3.0 Release Notes" on page 2-1
- "MATLAB Compiler 2.1 Release Notes" on page 3-1

Printing the Release Notes

If you would like to print the Release Notes, you can link to a PDF version.

Contents

1 MATLAB Compiler 3.0.1 Release Notes

2 MATLAB Compiler 3.0 Release Notes

3 MATLAB Compiler 2.1 Release Notes

1

MATLAB Compiler 3.0.1 Release Notes

Major Bug Fixes

The MATLAB Compiler 3.0.1 includes important bug fixes made since Version 3.0.

If you are viewing these Release Notes in PDF form, please refer to the HTML form of the Release Notes, using either the Help browser or the MathWorks Web site and use the link provided.

2

MATLAB Compiler 3.0 Release Notes

New Features

This section summarizes the new features and enhancements introduced in the MATLAB Compiler 3.0.

Note The MATLAB Compiler 2.3 was made available in Web-downloadable form after Release 12.1. These Release Notes include information about features added in Version 2.3, as well as Version 3.0 (i.e., since Release 12.1).

If you are upgrading from a release earlier than Release 12.1, then you should see "New Features" on page 3-2 in the MATLAB Compiler 2.1 Release Notes.

MATLAB Compiler 3.0

- The MATLAB Compiler includes the MATLAB C/C++ Math and Graphics Libraries. The libraries are no longer supported as separate products.
- A new optimization, `speculate`, is available. It improves the performance whenever the run-time types of variables match what is expected. It allows the Compiler to achieve similar performance to the new accelerator technology in MATLAB. See Performance Acceleration in the MATLAB documentation for more information. For more information about this optimization, see Optimizing Performance in the MATLAB Compiler documentation.
- The MATLAB Compiler allows you to create COM components from MATLAB M-files. For more information, see Building COM Objects in the MATLAB Compiler documentation. To create these components from the MATLAB Compiler, you must have the MATLAB COM Builder product installed on your system. To learn more about the MATLAB COM Builder, see its documentation.

MATLAB Compiler 2.3

MATLAB Compiler 2.3 allows you to create Microsoft Excel components from MATLAB M-files. This Windows-only feature translates a collection of M-files into a single Microsoft Excel add-in. Both C and C++ code generation are supported.

To support the creation of these components, the following Compiler options have been added or enhanced:

- The bundle option (`-B`) has been enhanced so that you can include replacement parameters for Compiler options that accept names and version numbers and they will be expanded properly.
- The new option, `-b`, causes the Compiler to generate a Visual Basic (`.bas`) file that contains the Microsoft Excel Formula Function interface to a Compiler-generated COM object.
- The new option, `-i`, causes the Compiler to include only the M-files that are specified on the command line as exported interfaces.

Note To create Microsoft Excel components with the MATLAB Compiler, you must have the MATLAB Excel Builder product installed on your system. To learn more about the MATLAB Excel Builder, see its documentation.

Upgrading from an Earlier Release

This section describes upgrade issues involved in moving from earlier versions of the MATLAB Compiler to Version 3.0.

MATLAB Compiler 3.0 is fully compatible with previous releases of the Compiler. If you have M-files that were compiled with a previous version of the Compiler and compile them with the new version, you will get the same results.

Upgrading from MATLAB Compiler 2.0/2.1/2.2/2.3

MATLAB Compiler 2.1 (and later versions) does not support the `-V1.2` option that was available in Compiler 2.0.

Upgrading from MATLAB Compiler 1.0/1.1

In many cases, M-code that was written and compiled in MATLAB 4.2 will work as is in the MATLAB 6 and the MATLAB 5 series. There are, however, certain changes that could impact your work, especially if you integrated Compiler-generated code into a larger application.

Changed Library Name

Beginning with MATLAB 5.0, the name of the shared library that contains compiled versions of most MATLAB M-file math routines, `libtbx`, has changed. The new library is now called `libmmfile`.

Changed Data Type Names

In C, beginning with MATLAB 5.0, the name of the basic MATLAB data type, `Matrix`, has changed. The new name for the data type is `mxArray`.

In C++, beginning with MATLAB 5.0, the name of the basic MATLAB data type, `mwMatrix`, has changed. The new name for the data type is `mwArray`.

3

MATLAB Compiler 2.1 Release Notes

New Features

This section introduces the new features added in the MATLAB Compiler 2.1 since the MATLAB Compiler 2.0.1 (Release 11.0).

Note The options for Compiler 2.1 are different from prior versions. Use

```
mcc -?
```

for help on the current set of options.

Optimizations

The MATLAB Compiler 2.1 supports several types of optimizations. The format of the optimization option is

```
-O <optimization>
```

There are three possibilities.

```
-O <optimization class>:[on|off]
```

```
-O <list>
```

```
-O <optimization level>
```

The first form

```
-O <optimization class>:[on|off]
```

turns the optimization class on or off. This table describes the possibilities.

Optimization Class	Description
`fold_scalar_mxarrays`	Fold scalar valued array constants
`fold_non_scalar_mxarrays`	Fold nonscalar valued array constants
`optimize_integer_for_loops`	Optimize `for`-loops with integer starts and increments

Optimization Class	Description
`optimize_conditionals`	Optimize conditional expressions when both operands are integers
`array_indexing`	Optimize simple one- and two-dimensional array index expressions

The second form of the optimization option

```
-O <list>
```

lists the available optimization classes.

The third form

```
-O <optimization level>
```

uses a bundle file called `opt_bundle_<level>` to determine which optimizations are on or off. For example,

```
-O all
```

looks for a bundle file called `opt_bundle_all` and uses the options in the bundle file. Bundles for `-O all` and `-O none` are provided by default.

Dynamically Linking in MEX-Files in the Stand-Alone Environment

The MATLAB Compiler 2.1 supports dynamically linking in MEX-files in the stand-alone environment. Specifying `-h` automatically compiles in any referenced MEX-files. Specifying MEX-files on the command line is supported and will work the same way as compiling an M-file.

The only restriction is that you cannot run a Compiler-generated MEX-file from stand-alone code. This restriction is minimal because you can include the M-file source directly, which is preferred by the Compiler.

MATLAB Add-In for Visual Studio®

The MATLAB add-in for the Visual Studio development environment integrates the MATLAB Compiler 2.1 into Visual C/C++ 5 or 6. Running

`mbuild -setup` or `mex -setup` automatically adds the Project Wizard into Microsoft Visual Studio so that you can use the Visual Studio environment to compile and run M-files. See the *MATLAB Compiler User's Guide* for more information on configuration.

mlib Files

The introduction of `mlib` files makes it possible to produce a shared library out of a toolbox and then compile M-files that make calls into that toolbox.

When compiling a collection of files (from a toolbox, for example) into a library, the Compiler creates a separate file with the M interface descriptions of the various M-functions that are available in the library. You can then use this file to compile other functions that make calls into the M-files in the library without recompiling the M-files. The extension of these library description files is `.mlib`.

Prior to this release, the Compiler would pass any file that it did not recognize on the command line as an M-file to `mbuild`. In this release, the Compiler uses files with the `.mlib` extension to allow compilation against a library of compiled M-files. See the *MATLAB Compiler User's Guide* for more information.

Additional Datatype Support

Integer Data Types. The MATLAB Compiler 2.1 supports the integer datatypes.

`int8`	`uint8`
`int16`	`uint16`
`int32`	`uint32`

Function Handles. A function handle is a new MATLAB data type that captures all the information about a function that MATLAB needs to execute, or evaluate, it. The MATLAB Compiler 2.1 supports function handles. For more information on function handles, see the `function_handle` reference page.

Improved Support for load and save

The `load` and `save` commands are supported when they do not list the variables to be loaded or saved.

Faster C/C++ Math Library Applications

The performance of the C/C++ Math Library has been improved considerably. Scalar accelerated versions of many library functions have been added, and improvements have been made to the overall performance of all library applications.

Printing from the C/C++ Graphics Library

The MATLAB C/C++ Graphics Library now supports printing.

Additional Language Support

pause and continue. The MATLAB Compiler 2.1 supports the MATLAB commands `pause` and `continue`.

eval and input. `eval` and `input` are supported for strings that do not contain workspace variables.

Note The MATLAB Compiler 2.1 does *not* support user-defined classes (MATLAB objects), scripts, or calls to the MATLAB Java interface.

Third Party Compilers

This version of the MATLAB Compiler supports the following PC compilers as they are shipped:

- MSVC C/C++ 5.0, and 6.0
- Borland C++ 5.0, 5.02
- Borland C++Builder 3, 4, 5
- Borland C++ 5.5 (free command line tools)
- LCC (C only compiler, bundled with MATLAB)

Note When installing MSVC 6.0, if you need to change where this compiler is installed, you must change the location of the `Common` directory (at the appropriate installer dialog.) If you change the location of the `VC98` directory from its default setting, the `MEX` and `MBUILD` scripts will not work properly.

Exception Handling

When using C++, the MATLAB Compiler relies on the availability of exception handling in the C++ language. Several of the supported compilers do not properly support C++ exception handling. Consequently, our support for exception handling is limited on those platforms.

GNU C++

GNU C++ 2.7.2 does not support C++ exception handling.

Borland

Borland C++ (all versions) has a restriction on support of `goto` statements within `try/catch` blocks. The MATLAB Compiler sometimes generates `goto` statements for complicated `if` conditional statements. The generated code for these will not compile in Borland C++; as a result you will be required to simplify the `if` condition.

Note The `-A debugline` option is implemented using `try/catch` statements. Therefore, use of this option is also restricted in the above C++ implementation.