

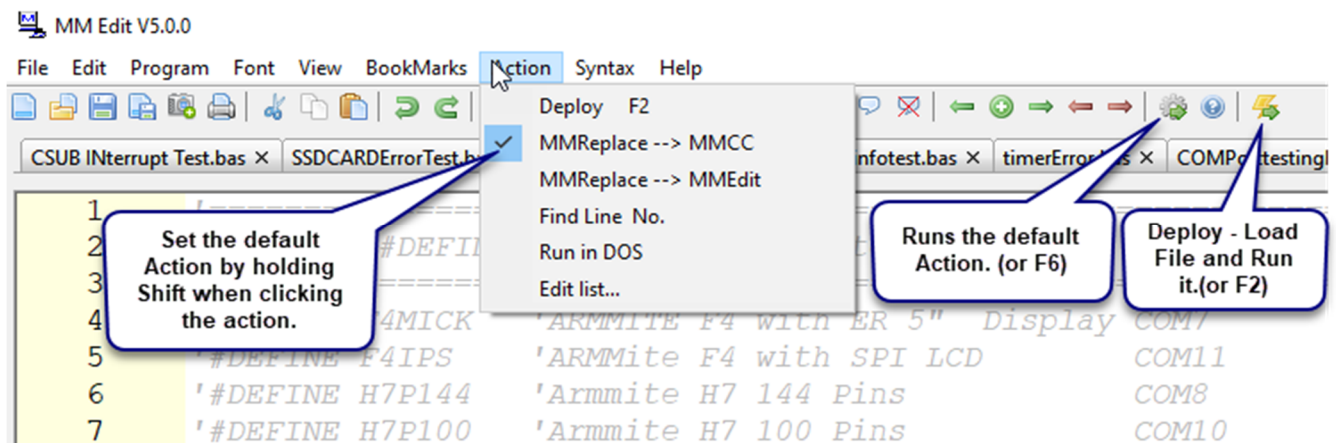
MMReplace - Addon for MMEdit Version 5

Version 5.00 - 01 Aug 2022

This document details the new MMReplace.exe that supports MMEdit Version 5. [MMEdit Version 5 post on TBS.](#)

MMReplace.exe is a utility written in PureBasic which supports the addition of several new Directives which can be used while developing code in MMEdit Version 5. It is integrated into MMEdit by calling it from within the MMEdit Action menu. Three options are added.

Action Menu	Function
MMReplace → MMCC	This calls MMREPLACE.EXE which processes any DIRECTIVES, deletes comments, blank lines and loads the program via MMCC (Maximite Control Centre)
MMReplace → MMEDIT	This calls MMREPLACE.EXE which processes any DIRECTIVES and sends the code and comments back to MMEdit as a new file.
Find Line No.	This calls FINDLINENO.EXE which prompts for an error line no to be input, and this line is then located in the original source file, taking into account comments and directives which were not sent as part of the loaded code. (This is done via sending keystrokes and does not expect SUBS/Functions to be folded)



The original MMReplace should be used for MMEdit versions 3 and 4. This version will only work with MMEdit Version 5. [MMReplace for MMEDIT 3 and 4 on TBS](#)

The filename remains as MMReplace.exe, it is up to you to keep them separate if you are using both versions.

Installation

The MMReplace files are installed alongside the MMEdit and MMCC executables. The integration is completed by adding options to the MMEdit Actions menu. This can be done via the MMEdit File/Preferences menu, but is probably easier done by directly adding them to the MMEdit.inf file located in the MMEdit's data directory while MMEdit is closed. The new directives can then be highlighted as keywords by adding them to the end of the various MMEdit syntax files(*.tkn files) also in the MMEdit data directory.

The installation consists of a number of files below provided in a zip file mmreplace.zip

./
MMReplace.exe
FindLineNo.exe
MMReplace5.pdf – This help file.

./source
MMReplaceV5.pb - The PureBasic source file for MMReplace.exe
Findline.pb - The PureBasic source file for findlineno.exe

MMEdit Portable and Nonportable installations

MMEdit has a portable and a traditional windows installation. The portable installation has the files placed in your chosen location and the data is located directly underneath this directory in a directory named DATA. The file portable.inf exists to identify a portable installation. e.g.

documents/MMEdit5/* .EXE, portable.inf

documents/MMEdit5/DATA/* .tkn

The tradition 64bit windows installation is at:

C:/Program Files/MMEdit5/* .exe, nonportable.inf (64 bit installation)

The data is located in the user's directory located at:

C:/Users/*username*/AppData/Local/MMEdit5

Add MMReplace EXE files

Place the two EXE files alongside the MMCC.EXE and MMEDIT5.EXE

Add Action Menu Options

Find the [External programs] section in the MMEdit.inf file located in the relevant MMEdit data location and replace it with the data below while MMEdit is shut down.

[External programs]

```
; %bas% The full path and filename to the current BAS file.
; %basshort% The filename to the current BAS file without any path.
; %baspath% The path to the current BAS file
; %extpath% The path to the external.exe (Default starting folder)
; %mmepath% The path To the MMEdit And MMCC program folder including trailing \
; %q% = quote Chr(34)
External hot link = 0
Ext0 = MMReplace --> MMCC F6|%q%%extpath%%q%|"MMReplace.exe" | %q%%bas%%q% "%mmepath%" "MMCC"
Ext1 = MMReplace --> MMEDIT|%q%%extpath%%q%|"MMReplace.exe" | %q%%bas%%q% "%mmepath%" "MMEDIT"
Ext2 = Find Line No.|%q%%extpath%%q%|"findlineno.exe" | "%mmepath%"
Ext3 = Run in DOS| %q%%baspath%%q%|"MMBasic.exe" | %q%%bas%%q%
```

Restart MMEdit and check for the new entries under the Action Menu.

Add Optional #ArrayGenerate Directives

The next step is only necessary if you want to use the #ARRAYGENERATE directives.

- Read the TBS forum post at http://www.thebackshed.com/forum/forum_posts.asp?TID=10253&PN=7
- Download the **simpleArrayFuncGen.exe** file. This should be placed in the same directory as **MMReplace.exe**.

Add Directives to MMEdit5 Syntax Files (Optional)

The new directives can be highlighted as keywords by adding them to the end of the various MMEdit syntax files(*.tkn files) in the relevant MMEdit data location.

Just copy and paste the codes below to the end of each required file.

```
#DEFINE #ELSE #ENDIF #IFDEF #IFNDEF #REPLACE #ENDCODE #KEEPCOMMENTS #DEBUG #LIBRARYSTART
#LIBRARYSTART1 #LIBRARYSTART2 #LIBRARYEND #LIBRARYLOAD #LIBRARYLOAD1 #LIBRARYLOAD2 #ARRAYSTART
#ARRAYEND #ARRAYINCLUDE #ARRAYGENERATE
```

Using the Directives

The MMReplace utility implements a number of additional directives that can be applied to code developed using the MMEdit code editing utility. Each directive is added into the code and all begin with a # and must be at the start of the line.

Replacing variable names at load time

#REPLACE *longname* *shortname*

At load time all occurrences of *longname* are replaced with *shortname*.

This can save program memory when the program is loaded/stored and can give small increase in execution speed when variables are resolved, but still allow the use of descriptive variable names in the code. e.g.

```
#REPLACE millivoltstep1 mvst1
```

```
#REPLACE millivoltstep2 mvst2
```

```
#REPLACE millivoltsperdivision mvpd
```

```
#REPLACE indexcalibrationx1probe ic1p
```

Another use is to decrease the number of discrete variables required by using an array to store similar variables, while still allowing your code to have meaningful names. e.g.

```
#REPLACE errortoobig errs(0)
```

```
#REPLACE errortoosmall errs(1)
```

```
#REPLACE errortoobad errs(2)
```

```
...
```

```
#REPLACE errorflagoutofrange errs(99)
```

Another use is to reduce the memory taken by discrete short strings by placing them in an array which can have the string length set to < 255, while also keeping you code meaningful. e.g.

```
#REPLACE string0$ strings$(0)
```

```
#REPLACE string1$ strings$(1)
```

```
...
```

```
#REPLACE string99$ strings$(99)
```

Condition Inclusion of Code

This set of directives allows code to be included in the load conditionally, by branching on the `#DEFINE` variables set.

This allows one source to be maintained and the respective `#DEFINE` directives uncommented to load the desired code. This can include including the appropriate `'target port\com7:115200` setting for MMCC to send to the matching device. Multiple nested `#DEFINE` statements can give good control over what is loaded. The one source can hold multiple test cases and these can be enabled as required. Alternate commands required by different devices can be conditionally selected e.g.

```
' Set the #DEFINE to get the corrected target for MMCC
#define F4MICK
#define F4IPS
#define H7P144
#define H7P100
#define PICO
#define WINBASIC
#ifdef F4MICK
    #define F4
    'target port\com7:115200 s\armite
#endif
#ifdef F4IPS
    #define F4
    'target port\com11:115200 s\armite
#endif
#ifdef PICO
    'target port\com14:115200 s\picomite settime\
#endif
#ifdef H7P144
    'target port\com8:115200 s\ARmite_H7
    #define H7
#endif
#ifdef H7P100
    'target port\com10:115200 s\ARmite_H7
    #define H7
#endif
#ifdef WINBASIC
    'target winbasic\ autorun
#endif
' No backlight on windows
#ifdef WINBASIC
    Backlight bright
#endif
```

'target winbasic\ autorun is a special case

The winbasic\ option for the MMCC 'target directive is a special case handled by MMReplace and is not part of MMCC. It will only work with windows. It will find an open MMBasic for Windows window and paste the code into. It will give an error message if the window cannot be found.

Once the MMBasic for Windows window is found:

- The code to be loaded is placed in windows clipboard
- A CNTRL C is sent to stop any running program
- CLS is sent via keystrokes to clear the screen
- AUTOSAVE *filename* is sent via keystrokes to the window
- CNTRL V is sent to paste the contents of the clipboard
- F1 (or F2 if autorun is part of the 'target) is sent to save and optionally run the program

This should be considered experimental. Works well on my machine, but there are some timings between the keystroke up/down events which may not suit all situations. The default is 50 microseconds. The directive below will allow some tuning if keystrokes are missed. **delay** is the new value to be used.

#WINDELAY *delay*

The filename used to save the code is the same as the filename in MMEdit

OneLoneCoder.com - Pixel Game Engine - MMBasic V5.07.03 - FPS: 62

```
> autosave "Lineinputtest"
'target winbasic\ autorun

PRINT "HELLO WORLD"
END
Saved 61 bytes
HELLO WORLD
>
—
```

Managing Library Code Micromite & MicromitePlus

#LIBRARYLOAD #LIBRARYSTART #LIBRARYEND

#LIBRARYLOAD1 #LIBRARYSTART1 #LIBRARYEND

#LIBRARYLOAD2 #LIBRARYSTART2 #LIBRARYEND

These directives were added to simplify the management of a project by allowing the library code to be kept with the main code. The #LIBRARYSTART and #LIBRARYEND directives are used to mark the beginning and end of the code to be kept in the library code. MMBasic (where LIBRARY is supported) allows incrementally adding to the library so provision is made to allow up to three library sections, which are loaded by the matching #LIBRARYLOAD directive. #LIBRARYEND will end the current library for any of the library sections.

When the #LIBRARYLOAD directive is present only the library code is loaded. If the directive is commented then the library is not loaded with the normal code but can reside in the one source file. e.g.

```
'--- Library Stuff -----  
'Uncomment the #LIBRARYLOAD directive below to load only the library code.  
'which is bounded by #LIBRARYSTART #LIBRARYEND directives  
'#LIBRARYLOAD 'Must start in column 1. Uncomment to enable.  
'---some handy library commands ---  
'LIBRARY SAVE  
'LIBRARY LIST  
'LIBRARY DELETE
```

Placing code that is stable into the library during development has the advantage of reducing the amount of code reloaded each time during the development cycle. This can be quite a saving if the program is large. The code transferred to the library is still visible in the MMEdit code window if you need to inspect it.

#ENDCODE

This directive also helps support a single source file for the project. This directive is placed at the end of the valid MMBasic code and marks the end of code to be loaded. This allows other relevant code/notes to be stored after the directive but in the same source file.

e.g. associated Annex code, Arduino libraries to reference during development, other notes etc. these do not affect the load time.

#DEBUG

MMReplace writes progress information to a console window, which it automatically closes at the end of processing. This directive will prevent the MMReplace console automatically closing after code is sent to MMCC. The information displayed in the console may help in diagnosing any MMReplace issues. The console window needs to be manually closed.

C Style Comment Blocks

```
/*
```

These C style comment blocks can be used in the code to mark a block as comments. The individual lines are made into comments by MMREPLACE. The MMEdit feature to comment and uncomment selected blocks is just as effective and also shows clearly as comments in MMEdit.

```
*/
```

Loading Arrays with simpleArrayFuncGen

The simpleArrayFuncGen.exe written by @Nathan from The Backshed Forum (TBS) allows the loading of static arrays into flash to be stored as either CFunctions or CSubs. The details and exe file are available in the following forum post. (Only relevant for the PIC Micromites.)

http://www.thebackshed.com/forum/forum_posts.asp?TID=10253&PN=7

[simpleArrayFuncGen](#)

The following directives allow the MMEdit code window to include the definitions for the simpleArrayFuncGen config files and the array data and for the CFunction/CSub to be generated at load time by calling simpleArrayGenerator at that time.

#ARRAYSTART [string|integer|float] [cfunction|csub] *functionorsubname*

Indicates the start of the array definition and the parameters detail, the type of array, where a CFunction or a CSub should be reduced and the *name* of the function or sub. The *name* is used as the base *name* for the *name.cfg* and *name.bas* file names passed to simpleArrayFuncGen.

#ARRAYEND

Indicates the end of the array definition.

#ARRAYINCLUDE *functionorsubname*

Place in source where the generated CFunction/CSub is to be included during Load and Run

Placing between #LIBRARYSTART and #LIBRARYEND directives ensures it only included when loading the library code.

#ARRAYGENERATE

This directive causes the array CFunction/CSub to be generated during Load and Run. If not present the simpleArrayFuncGen .exe is not called so only any previously generated array CFunction/CSub functions would be loaded. If the code is to be included in the library, then using both the #LOADLIBRARY and #ARRAYGENERATE would achieve this. They could then be both commented during development of the main code.

The array data is entered between the #ARRAYSTART and #ARRAYEND directives. The format of this is determined by simpleArrayFuncGen as it is passed straight through, but essentially each line containing data begins with a: followed by a space.

'Example of string array

```
#ARRAYSTART  
: this is string1  
: this is string2  
#ARRAYEND
```

'Example of string array with comment included so MMEdit syntax highlighting

'is not applied to the text. The comment is automatically stripped when to array config file is produced.

```
#ARRAYSTART  
' : this is string1  
' : this is string2  
#ARRAYEND
```

'Example of int array

```
#ARRAYSTART  
: 1 2 3 4 5  
: 10 20 30 40 50  
#ARRAYEND
```

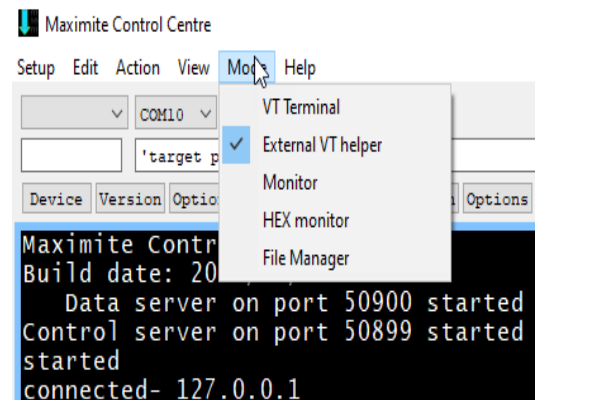
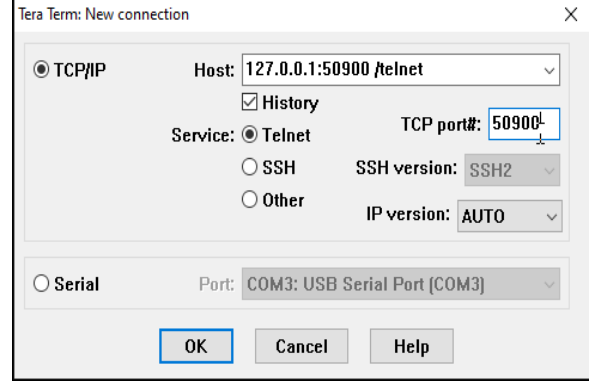
Summary of Directives Added by MMReplace

Directive	Usage
<code>#REPLACE <i>longname shortname</i></code>	Replaces all occurrences of <i>longname</i> with <i>shortname</i> when code is loaded. The original directive which gave this program its name.
<code>#LIBRARYSTART</code>	Marks beginning of code to be stored in the library
<code>#LIBRARYSTART1</code>	Marks beginning of code to be stored in the library 1
<code>#LIBRARYSTART2</code>	Marks beginning of code to be stored in the library 2
<code>#LIBRARYEND</code>	Marks end of code to be stored in library
<code>#LIBRARYLOAD</code>	Indicates only the library code should be loaded
<code>#LIBRARYLOAD1</code>	Indicates only the library1 code should be loaded
<code>#LIBRARYLOAD2</code>	Indicates only the library2 code should be loaded
<code>#ENDCODE</code>	All code beyond this directive is not loaded
<code>#DEFINE <i>variable</i></code> <code>#IFDEF <i>variable</i></code> <code>#IFNDEF <i>variable</i></code> <code>#ELSE</code> <code>#ENDIF</code>	This set of directives allows code to be included in the load conditionally, based on the <code>#DEFINE <i>variable</i></code> .
<code>#KEEPCOMMENTS</code>	Comments are by default removed when loading code via the MMReplace → MMCC action. This directive will ensure they are included. (Note: Comments are always kept for the MMReplace → MMEdit action.)

<pre>/* */</pre>	<p>Start comment block End comment block Allows a block to be treated as comments. This can easily be done in MMEdit itself, so of limited value.</p>
#DEBUG	<p>Add this to the code to keep the console window of MMReplace.exe open after a program is loaded. The information in here may help in diagnosing any MMReplace issues.</p>
#WINDELAY <i>delay</i>	<p>Changes the default 50 microseconds delay between keystrokes when sending keystrokes to MMBasic for Windows to the specified value. Allows some tuning if keystrokes are being missed.</p>
#ARRAYSTART [string integer float] [cfunction csub] <i>functionorsubname</i>	<p><i>Only applicable to Micromites. i.e. Microchip PIC32. No equivalent for ARM chips.</i> Indicates the start of the array definition and the parameters detail, the type of array, where a cfunction of a csub should be reduced and the <i>name</i> of the function or sub. The <i>name</i> is used as the base <i>name</i> for the <i>name.cfg</i> and <i>name.bas</i> file names passed to simpleArrayGenerator.</p>
#ARRAYEND	<p>Indicates the end of the array definition.</p>
#ARRAYINCLUDE <i>functionorsubname</i>	<p>Place in source where the generated cfunction/csub is to be included during Load and Run</p>
#ARRAYGENERATE	<p>This directive causes the array cfunction/csub to be generated during Load and Run. If not present the simpleArrayFuncGen.exe is not called so only any previously generated array cfunction/csub functions would be loaded.</p>

Connecting TeraTerm via MMCC

The MMEdit help file should be consulted for more details, but below are the basic requirements to connect Teraterm via MMCC.

<p>Start MMCC and select Mode → External VT Helper MMCC will now accept TCP connections on port 50900 This allows both MMEdit and TeraTerm to access the connected device.</p>	
<p>Start a TeraTerm TCP session that connects to 127.0.0.1:50900 or Localhost:50900 Once MMEdit finishes loading a program, control of the session is handed to TeraTerm.</p>	

Tip

If TeraTerm looks like it is in Block mode, i.e. doesn't transmit until the enter key is pressed check this entry in the teraterm.ini file and sure it is set to off.

```
; Line at a time mode  
EnableLineMode=off
```

Appendix 1 - MMEdit5, MMCC , TeraTerm and MMReplace Interworking

MMEdit5 , Maximite Control Centre , TeraTerm and MMReplace Interworking

