# Table of Contents

## 1 Using Burner

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introduction</td>
<td>7</td>
</tr>
<tr>
<td>Highlights</td>
<td>7</td>
</tr>
<tr>
<td>Structure of this Document</td>
<td>8</td>
</tr>
<tr>
<td>Interactive Burner</td>
<td>8</td>
</tr>
<tr>
<td>User Interface</td>
<td>8</td>
</tr>
<tr>
<td>Batch Burner</td>
<td>18</td>
</tr>
<tr>
<td>User Interface</td>
<td>18</td>
</tr>
<tr>
<td>Syntax of Burner Command Files</td>
<td>19</td>
</tr>
<tr>
<td>Batch Burner with Makefile</td>
<td>20</td>
</tr>
<tr>
<td>Parameters of the Command File</td>
<td>23</td>
</tr>
<tr>
<td>baudRate</td>
<td>24</td>
</tr>
<tr>
<td>busWidth</td>
<td>25</td>
</tr>
<tr>
<td>CLOSE</td>
<td>26</td>
</tr>
<tr>
<td>dataBit</td>
<td>26</td>
</tr>
<tr>
<td>destination</td>
<td>27</td>
</tr>
<tr>
<td>DO</td>
<td>28</td>
</tr>
<tr>
<td>ECHO</td>
<td>29</td>
</tr>
<tr>
<td>ELSE</td>
<td>29</td>
</tr>
<tr>
<td>END</td>
<td>30</td>
</tr>
<tr>
<td>FOR</td>
<td>31</td>
</tr>
<tr>
<td>format</td>
<td>32</td>
</tr>
<tr>
<td>header</td>
<td>33</td>
</tr>
<tr>
<td>IF</td>
<td>34</td>
</tr>
<tr>
<td>len</td>
<td>35</td>
</tr>
<tr>
<td>OPENCOM</td>
<td>36</td>
</tr>
<tr>
<td>OPENFILE</td>
<td>36</td>
</tr>
<tr>
<td>origin</td>
<td>37</td>
</tr>
<tr>
<td>parity</td>
<td>38</td>
</tr>
<tr>
<td>SENDBYTE</td>
<td>39</td>
</tr>
<tr>
<td>SENDWORD</td>
<td>40</td>
</tr>
<tr>
<td>Command</td>
<td>Page</td>
</tr>
<tr>
<td>-----------------</td>
<td>------</td>
</tr>
<tr>
<td>SLINELEN</td>
<td>41</td>
</tr>
<tr>
<td>SRECORD</td>
<td>42</td>
</tr>
<tr>
<td>swapByte</td>
<td>43</td>
</tr>
<tr>
<td>THEN</td>
<td>44</td>
</tr>
<tr>
<td>TO</td>
<td>45</td>
</tr>
<tr>
<td>undefByte</td>
<td>45</td>
</tr>
<tr>
<td>PAUSE</td>
<td>46</td>
</tr>
<tr>
<td>Burner Options</td>
<td>47</td>
</tr>
<tr>
<td>Option Details</td>
<td>48</td>
</tr>
<tr>
<td>-D</td>
<td>49</td>
</tr>
<tr>
<td>-Env</td>
<td>50</td>
</tr>
<tr>
<td>-F</td>
<td>51</td>
</tr>
<tr>
<td>-H</td>
<td>52</td>
</tr>
<tr>
<td>-Lic</td>
<td>53</td>
</tr>
<tr>
<td>-LicA</td>
<td>54</td>
</tr>
<tr>
<td>-N</td>
<td>55</td>
</tr>
<tr>
<td>-NoBeep</td>
<td>56</td>
</tr>
<tr>
<td>-NoEnv</td>
<td>57</td>
</tr>
<tr>
<td>-Ns</td>
<td>58</td>
</tr>
<tr>
<td>-Prod</td>
<td>59</td>
</tr>
<tr>
<td>-V</td>
<td>60</td>
</tr>
<tr>
<td>-View</td>
<td>61</td>
</tr>
<tr>
<td>-W</td>
<td>62</td>
</tr>
<tr>
<td>-Wmsg8x3</td>
<td>63</td>
</tr>
<tr>
<td>-WErrFile</td>
<td>64</td>
</tr>
<tr>
<td>-WmsgCE</td>
<td>65</td>
</tr>
<tr>
<td>-WmsgCF</td>
<td>66</td>
</tr>
<tr>
<td>-WmsgCI</td>
<td>67</td>
</tr>
<tr>
<td>-WmsgCU</td>
<td>68</td>
</tr>
<tr>
<td>-WmsgCW</td>
<td>69</td>
</tr>
<tr>
<td>-WmsgFb (-WmsgFbi, -WmsgFbm)</td>
<td>70</td>
</tr>
<tr>
<td>-WmsgFi (-WmsgFiv, -WmsgFim)</td>
<td>71</td>
</tr>
<tr>
<td>-WmsgFob</td>
<td>72</td>
</tr>
<tr>
<td>Topic</td>
<td>Page</td>
</tr>
<tr>
<td>----------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>-WmsgFoi.</td>
<td>74</td>
</tr>
<tr>
<td>-WmsgFonf</td>
<td>75</td>
</tr>
<tr>
<td>-WmsgFonp</td>
<td>76</td>
</tr>
<tr>
<td>-WmsgNe.</td>
<td>78</td>
</tr>
<tr>
<td>-WmsgNi</td>
<td>78</td>
</tr>
<tr>
<td>-WmsgNu.</td>
<td>79</td>
</tr>
<tr>
<td>-WmsgNw</td>
<td>80</td>
</tr>
<tr>
<td>-WmsgSd</td>
<td>81</td>
</tr>
<tr>
<td>-WmsgSe</td>
<td>82</td>
</tr>
<tr>
<td>-WmsgSi</td>
<td>83</td>
</tr>
<tr>
<td>-WmsgSw.</td>
<td>83</td>
</tr>
<tr>
<td>-WOutFile.</td>
<td>84</td>
</tr>
<tr>
<td>-WStout.</td>
<td>85</td>
</tr>
<tr>
<td>-W1</td>
<td>86</td>
</tr>
<tr>
<td>-W2</td>
<td>87</td>
</tr>
<tr>
<td>Environment</td>
<td>88</td>
</tr>
<tr>
<td>The Current Directory</td>
<td>89</td>
</tr>
<tr>
<td>Global Initialization File (MCUTOOLS.INI) (PC only)</td>
<td>89</td>
</tr>
<tr>
<td>Local Configuration File (usually project.ini)</td>
<td>94</td>
</tr>
<tr>
<td>Paths</td>
<td>108</td>
</tr>
<tr>
<td>Line Continuation.</td>
<td>109</td>
</tr>
<tr>
<td>Environment Variable Details</td>
<td>109</td>
</tr>
<tr>
<td>DEFAULTDIR</td>
<td>110</td>
</tr>
<tr>
<td>ENVIRONMENT.</td>
<td>111</td>
</tr>
<tr>
<td>ERRORFILE</td>
<td>112</td>
</tr>
<tr>
<td>GENPATH</td>
<td>114</td>
</tr>
<tr>
<td>TMP</td>
<td>115</td>
</tr>
<tr>
<td>Messages</td>
<td>116</td>
</tr>
<tr>
<td>Message Kinds</td>
<td>116</td>
</tr>
<tr>
<td>Message Details</td>
<td>117</td>
</tr>
<tr>
<td>Message List</td>
<td>117</td>
</tr>
</tbody>
</table>

**Index** 125
Using Burner

Introduction

The burner utility converts a .ABS file into a file that can be handled by an EPROM burner. The Burner is available as:

- An interactive burner with a graphical user interface (GUI).
- A batch burner that either accepts commands from a command line or in a command file. It can then be invoked by the Make Utility.

Highlights

- Powerful User Interface
- On-line Help
- Flexible Message Management
- 32bit Application
- Generation of Motorola S-Record files, Binary or Intel Hex files
- Splitting up application into different EEPROMS (1, 2 or 4 bytes bus width)
- Both interactive (GUI) and batch language interface (Batch Burner)
- Batch Burner Language with baudRate, busWidth, CLOSE, dataBit, destination, DO, ECHO, ELSE, END, FOR, format, header, IF, len, OPENCOM, OPENFILE, origin, parity, PAUSE, SENDBYTE, SENDWORD, SLINELEN, SRECORD, swapByte, THEN, TO, undefByte.
- Supports HIWARE and ELF/Dwarf Object File Format, Motorola S-Records and Intel Hex Files as input
- Supports a serial programmer attached to serial port with various configuration settings
- Powerful batch burner language with various commands (fillByte, origin, destination, range, baudRate, header, ...)
Interactive Burner

Instead of writing a batch burner language file, you can use the burner user interface to burn your EEPROM. You can set all parameters and receive the output needed for a batch burner language file.

User Interface

When the Burner is started, it opens the following window.

To open the burner dialog box, click in the tool bar or select

in the tool bar or select
from the menu.

The dialog box can also be accessed by the following command line option:

`burner.exe -D`

The dialog box consists of three tabs:

- Input/Output
- Content
- Command File

The dialog is initialized with the values of the last burn session. Values are written to the `project.ini` file in the `[BURNER]` section.

**Input/Output**

In the Input/Output tab, specify which file the burner uses for input and where to write the output. Click the Execute button to start the operation.

Output from the burn process usually goes to a PROM burner connected to the serial port. Output also may be redirected to a file written in either Intel Hex format, as Motorola S-Records or as plain binary.
Specify the input file in the Input File: text field. The browse button on the right side is used to browse for a file. Following file types are supported:

- Absolute files produced by linker. The absolute file format may be either HIWARE or ELF/Dwarf
- Motorola S Record File
- Intel Hex File

The corresponding Batch Burner command is **SENDBYTE** or **SENDWORD**.
To specify the input file, you can use the following macro `%ABS_FILE%` where ABS_FILE is passed by an environment variable. See the description of environment variables.

For example:

```
-ENV" ABS_FILE=file_name"
```

**Output**

Output is written to a serial port (COM1, COM2, COM3 or COM4) or a file.

**File**

Select the File radio button to write output to a file. In the corresponding text box, you can enter the output file name or browse for an existing file.

The corresponding Batch Burner command is **OPENFILE**.

If you use the macro `%ABS_FILE%` for the input file, you can add an extension to automatically generate the output file.

Example:

```
%ABS_FILE%.s19
```

**Com1, Com2, Com3, Com4**

To write the output to a serial port, select an available port and define the communication settings.

The corresponding Batch Burner command is **OPENCOM**.
Using Burner

Interactive Burner

Baud Rate
Supported Baud Rates: 300, 600, 1200, 2400, 4800, 9600, 19200 and 38400
The corresponding Batch Burner command is `baudRate`.

Parity
Communication parity can be set to none, even or odd.
The corresponding Batch Burner command is `parity`.

Data Bits
Number of data bits transferred can be set to 7 or 8 bits.
The corresponding Batch Burner command is `dataBit`.

Header File
You can specify an initialization file for the PROM burner. This file is sent to the PROM burner byte by byte (binary) without modification before anything else is sent.
The corresponding Batch Burner command is `header`.

Execute
From the dropdown list select which byte or word to be written and click Execute:
• 1st Byte (msb)
• 2nd Byte
• 3rd Byte
Using Burner

Interactive Burner

- 4th Byte
- 1st Word
- 2nd Word

Depending on the data width chosen, you may have to send the result to more than one output file.

Example: Format is Motorola S Record and data bus is 2 Bytes

Two output files are generated. Data for the 1st Byte (msb) is sent to a file named fibo_1.s19 and data for the 2nd byte is sent to fibo_2.s19.

Select 1st Byte (msb) and click Execute to transfer the code bytes, if you select a data bus width of 1 byte.

If your data bus is 2 bytes wide, the code is split into two parts. Selecting the 1st Byte (msb) and clicking Execute will transfer the even part of the data (corresponding to D8 to D15). Selecting 2nd Byte will transfer the odd part, which corresponds to LSB or D0 to D7.

If the data bus is 4 bytes wide, 1st Byte (msb) transfers D24 to D31, while 4th Byte sends the LSB (D0 to D7).

If using 16bit EPROMs, select one of the Word formats. If necessary, you can exchange the high and low byte. Check Swap Bytes in the Content tab of the Burner dialog box.

The corresponding Batch Burner commands are SENDBYTE and SENDWORD.

**Content**

In the content tab, the data format and range to be written is specified:
Using Burner

Interactive Burner

The Burner supports three output formats:

- Motorola S Records
- Intel Hex Files
- Binary Files

The corresponding Batch Burner command is `format`.

---

<table>
<thead>
<tr>
<th>Input/Output</th>
<th>Content</th>
<th>Command File</th>
</tr>
</thead>
<tbody>
<tr>
<td>Format:</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Motorola S</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Intel Hex</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Binary</td>
<td></td>
</tr>
<tr>
<td>Data Bus:</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1 Byte</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2 Bytes</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4 Bytes</td>
<td></td>
</tr>
<tr>
<td>Swap Bytes:</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>checked</td>
<td></td>
</tr>
<tr>
<td>Undef Byte (hex):</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Range to Copy:</td>
<td></td>
<td>Extracts all bytes from 0x1000 to 0x4fff (including 0x4fff) and writes the address of the byte 0x1000 as address 0x100</td>
</tr>
<tr>
<td>Origin:</td>
<td>1000</td>
<td></td>
</tr>
<tr>
<td>Length:</td>
<td>4000</td>
<td></td>
</tr>
<tr>
<td>Destination Offset:</td>
<td>100</td>
<td></td>
</tr>
</tbody>
</table>
### SRecord Configuration

For SRecords, the type can be set to automatic, S1, S2 or S3.
The corresponding Batch Burner command is **SRECORD**.
The number of bytes per SRecord line can be configured. This is useful when using tools with restricted capacity. The Batch Burner command is **SLINELEN**.

### Data Bus

Data Bus/Data Width may be either 1, 2 or 4 bytes.
The corresponding Batch Burner command is **busWidth**.

### Swap Bytes

Swapping bytes may be enabled, if the data bus is 2 or 4 bytes.
The corresponding Batch Burner command is **swapByte**.

### Undef Byte

The Batch Burner command is **undefined**.
For a binary output file, normally all undefined bytes in the output are written as 0xFF. If desired, another pattern can be specified.

The corresponding Batch Burner command is `undefByte`.

**Range**

In the Range to Copy group, the origin (start), length, and offset is specified. The text box on the right explains the result.

<table>
<thead>
<tr>
<th>Range to Copy:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Origin:</td>
<td>0x000</td>
</tr>
<tr>
<td>Length:</td>
<td>0x4000</td>
</tr>
<tr>
<td>Destination Offset:</td>
<td>0x1000</td>
</tr>
</tbody>
</table>

Example: If your application is linked at address $3000 to $4000 and the EPROM is at address $2000 (Origin) and Length is $2000, the code will start at address $1000 relative to the EPROM. If the EPROM is at address $3000 (Origin) and Length is $1000, it is filled from the start.

**Origin**

This has to be set to the EEPROM start address in your system.

The corresponding Batch Burner command is `origin`.

**Length**

This is the range of program code that is actually copied.

The corresponding Batch Burner command is `len`.

**Destination Offset**

This is an additional offset which will be added to the resulting Motorola S Record or Intel Hex File. For example, if the Origin is set to 0x3000 and the Destination offset is 0x1000, then the written address will be 0x4000.

The corresponding Batch Burner command is `destination`.
Command File

In the Command File tab of the Burner dialog box, a summary of your settings are displayed as Batch Burner commands. You can select and copy the commands for use in make files or Batch Burner Language (.bbl) files.

If you use the selection in a make file, either place everything on a single line or use the line continuation character (\) as shown.

```plaintext
burn:
  $(BURN) \n    OPENFILE "fibo.s19" \n    format = motorola \n    origin = 0xE000 \n    len = 0x2000 \n    busWidth = 1
```
Using Burner
Batch Burner

Batch Burner

User Interface

Starting the Burner displays the following window:

To use the Batch Burner, you can type in your batch burner commands on the command line, specify a file using the -F option on the command line, or as a startup option:

-F fibob.bb

or

OPENFILE "fibo.s19" origin=0xE000 len=0x2000 SENDBYTE 1 "fibo.abs"

You can also specify options and burner commands with the burner program.

burner.exe -F fibo.bb

You can also use the Burner directly from a make file:

burn:

$(BURN) \n    OPENFILE "fibo.s19" \n    format = motorola \n    origin = 0xE000 \n    len = 0x2000 \n    busWidth = 1 \n    SENDBYTE 1 "fibo.abs"
Syntax of Burner Command Files

Syntax of burner commands:

```
StatementList = Statement {Separator Statement}.
Statement = [IfSat | ForStat | Open | Send | Close | Pause
    | Echo | Format | SFormat | Origin | Len
    | BusWidth | Parity | SwapByte | Header
    | BaudRate | DataBit | UndefByte
    | Destination | AssignExpr | SLineLen].
IfStat = "IF" RelExpr "THEN" StatementList
    ["ELSE" StatementList] "END".
Assign = ("=" | ":=").
ForStat = "FOR" Ident Assign SimpleExpr "TO" SimpleExpr
    "DO" StatementList "END".
Send = ("SENDBYTE" | "SENDWORD") SimpleExpr String.
Close = "CLOSE".
Pause = "PAUSE" [String].
Echo = "ECHO" [String].
Format = "format" Assign ("motorola" | "intel" | "binary").
SFormat = "SRECORD" Assign ("Sx" | "S1" | "S2" | "S3").
Origin = "origin" Assign SimpleExpr.
Len = "len" Assign SimpleExpr.
BusWidth = "busWidth" Assign ("1" | "2" | "4").
Parity = "parity" Assign ("none" | "even" | "odd").
SwapByte = "swapByte" Assign ("yes" | "no").
Header = "header" Assign string.
BaudRate = "baudRate" Assign ("300" | "600" | "1200"
    | "2400" | "4800" | "9600"
    | "19200" | "38400").
DataBit = "dataBit" Assign ("7" | "8").
UndefByte = "undefByte" Assign SimpleExpr.
Destination = "destination" Assign SimpleExpr.
SLineLen = "SLINELEN" Assign SimpleExpr.
AssignExpr = Ident Assign SimpleExpr.
RelExpr = SimpleExpr {RelOp SimpleExpr}.
RelOp = ":=" | ":=" | ":#" | ":<>" | ":!=" | ":<"
    | ":<=" | ":>" | ":>=".
SimpleExpr = ["+" | "-" | "*" | "/"] Term {AddOp Term}.
AddOp = "+" | ":-".
Term = Number | String | Ident.
Number = 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0 | {Number}
Ident = "i".
```
String = '"' {char} '"'.

NOTE The identifier used in a FOR statement must be called "i".

Command files accept both ANSI-C style or Modula-2 style comments.

Example
/* This is a C like comment */
/* This is a Modula-2 like comment */

Assignments can be specified using ANSI-C or Modula-2 syntax:

Example
dataBit := 2 (* Modula-2 like *)
dataBit = 2 /* C like */

Constant format can be specified using either ANSI-C or Modula-2 syntax:

Example
origin = 0x1000 /* C like */
origin := 1000H (* Modula-2 like *)

Batch Burner with Makefile

In a makefile, the burner can be used in two different ways. The first way is to specify a command file:

BURNER.EXE -f "<CmdFile>"

The second way is to directly specify commands on the command line:

BURNER.EXE SENDBYTE 1 "InFile.abs"

If the commands are long, you can use line continuation characters in your make file as below:

burn:
$(BURN) \ 
OPENFILE "fibo.s19" \ 
format = motorola \ 
origin = 0xE000 \ 
len = 0x2000 \ 
busWidth = 1
If the second method is used, parameter initialization may be included in the file DEFAULT.ENV located in the working directory. This will reduce the length of the command line parameters, which are limited to 4096 bytes. Variables that can be specified using environment variables are listed below:

```bash
header=
format=motorola
busWidth=1
origin=0
len=0x10000
parity=none
undefByte=0xff
baudRate=9600
dataBit=8
swapByte=no
```

The example above shows the default values but any legal value can be assigned (see section Parameters of the Command File). For further details, see the example in the following section.

## Examples

The examples below show how to write a command file.

### Example 1

Example for conditional and repetitive statements. If the symbol # appears in a string it is replaced by the value of i.

```bash
ECHO
ECHO " I can count... and I can take decisions"
FOR i = 0 TO 8 DO
  IF i == 7 THEN
    ECHO "This is the number seven"
  ELSE
    ECHO "#"
  END
END
IF i == 3 THEN
  ECHO "This was the number three"
END
END
```

### Example 2
Example for redirecting the output to a file. To redirect output, use the command OPENFILE.

```vhdl
ECHO
ECHO "Programming 2 EPROMs with 3 files"
ECHO "the first byte of the word goes into the first EPROM"
ECHO "the second byte of the word goes into the second EPROM"
PAUSE "Hit any key to continue"
    format = motorola
    busWidth = 2
    origin = 0
    len = 0x3000
FOR i = 1 TO 2 DO
    PAUSE "Insert EPROM n# and press <return>"
    OPENFILE "prom#.bin"
        origin = 0X
        SENDBYTE i "demo1.abs"
        origin = origin + 0x500
        SENDBYTE i "demo2.abs"
        origin = origin + 0x500
        SENDBYTE i "demo3.abs"
    CLOSE
END
```

**Example 3**

Example for redirecting the output to a serial port. To redirect output to serial port, use the command OPENCOM.

```vhdl
ECHO
ECHO "I can also program 16-bit EPROMs with header"
PAUSE "Hit any key to continue"
header = "init.prm"
format = intel
busWidth = 2
origin = 0x0
len = 0x1000
OPENCOM 1 /* here com1, com2, com3 or com4 could be used*/
SENDWORD 1 "fbin1.map"
CLOSE
```

**Example 4**
Example for calling the burner from a makefile. This example shows how the burner can be called from a makefile. After compiling and linking the application, the generated code is prepared to be burned into two EPROMs, one containing the odd bytes (fibo_odd.s1) and the other the even bytes (fibo_eve.s1).

```
makeall:
  $(COMP) $(FLAGS) fibo.c
  $(LINK) fibo.prm
  burner.exe OPENFILE "fibo_odd.s1" \ 
    busWidth=2 SENDBYTE 1 "fibo.abs"
  burner.exe OPENFILE "fibo_eve.s1" \ 
    busWidth=2 SENDBYTE 2 "fibo.abs"
```

Note that for all parameters that are not specified in the parameter list, default values or the values specified by environment variables will be used.

**Parameters of the Command File**

This section describes valid parameter values that can be used in commands. For more details about commands, refer to the file FIBO.BBL, which shows how to write a script.

Following commands are available:

- `baudRate`
- `busWidth`
- `CLOSE`
- `dataBit`
- `destination`
- `DO`
- `ECHO`
- `ELSE`
- `END`
- `FOR`
- `format`
- `header`
- `IF`
- `len`
• OPENCOM
• OPENFILE
• origin
• parity
• PAUSE
• SENDBYTE
• SENDWORD
• SLINELEN
• SRECORD
• swapByte
• THEN
• TO
• undefByte

baudRate

baudRate: Baudrate for Serial Communication

Syntax:
"baudRate" assign <baud>.

Arguments:

<baud>: valid baudrate.

Default:

baudrate = 9600

Description:

Sets the transmission speed. This parameter must not be used when the burner output is redirected to a file. Valid identifier values are 300, 600, 1200, 2400, 4800, 9600, 19200 or 38400 (default is 9600).

This command is only used if output is sent to a communication port.
Example:

`baudRate = 19200`

See also:

- `dataBit`
- `parity`
- `header`
- `OPENCOM`

### busWidth

**busWidth: Data Bus Width**

**Syntax:**

```
"busWidth" assign ("1" | "2" | "4").
```

**Arguments:**

A bus width of 1, 2 or 4

**Default:**

`busWidth = 1`

**Description:**

Most EPROMs are 1 byte wide. In order to burn an application into EPROMs, 1, 2 or 4 EPROMs are needed depending on the width of the data bus of the target system used. The Burner program allows you to select the data bus width using the identifier `busWidth`. Only 1, 2 and 4 are valid values for the parameter `busWidth` (the default is 1).

**Example:**

`busWidth = 4`

**See also:**

none.
CLOSE

CLOSE: Close Open File or Communication Port

Syntax:
"CLOSE".

Arguments:
none.

Default:
none.

Description:
To close a file opened by OPENFILE or COM port opened with OPENCOM.

Example:
CLOSE

See also:
- OPENFILE
- OPENCOM

dataBit

dataBit: Number of Data Bits

Syntax:
"dataBit" assign ("7" | "8").

Arguments:
7 or 8 data bits.

Default:
dataBit = 8

**Description:**

Sets the number of data bits. This parameter must not be used when the burner output is redirected to a file. Valid identifier values are 7 or 8 (default is 8).

This command is only used if the output is sent to a communication port.

**Example:**

dataBit = 7

**See also:**
- baudRate
- parity
- header
- OPENCOM

**destination**

**destination: Destination Offset**

**Syntax:**

"destination" assign <offset>.

**Arguments:**

<offset>: offset to be added

**Default:**

destination = 0

**Description:**

With this command an additional offset may be added to the address field of an S-Record or a Intel Hex Record.

**Example:**
destination = 0x2000

See also:
- len
- origin

DO

DO: For Loop Statement List

Syntax:
"FOR" Ident Assign SimpleExpr
"TO" SimpleExpr "DO" StatementList "END".

Arguments:
none.

Default:
none.

Description:
This command starts the FOR statement list. As ident only ‘i’ may be used, and each occurrence of # in the loop is replaced with the actual value of ‘i’.

Example:
FOR i=0 TO 10 DO
  ECHO "#"
END

See also:
- FOR
- TO
- END
ECHO

ECHO: Echo String onto Output Window

Syntax:
"ECHO" [<string>].

Arguments:
<string>: a string written to the output window

Default:
none.

Description:
With this command, a string can be written to the output window. If no string is specified, an empty line is written.

Example:
ECHO
ECHO "hello world!"

See also:
none.

ELSE

ELSE: Else Part of If Condition

Syntax:
"IF" RelExpr "THEN" StatementList
["ELSE" StatementList] "END".

Arguments:
none.
**Default:**

none.

**Description:**

This command starts the optional ELSE part of an IF conditional section.

**Example:**

```plaintext
FOR i=0 TO 10 DO
  IF i==7 THEN
    ECHO "i is 7"
  ELSE
    ECHO "#"
  END
END
```

**See also:**

- **END**
- **IF**
- **THEN**

## END

**END: For Loop End or If End**

**Syntax:**

"FOR" Ident Assign SimpleExpr
"TO" SimpleExpr "DO" StatementList "END".

or

"IF" RelExpr "THEN" StatementList
"ELSE" StatementList "END".

**Arguments:**

none.

**Default:**
none.

**Description:**
This command ends either a FOR loop or IF condition.

**Example:**

```plaintext
FOR i=0 TO 10 DO
  IF i==7 THEN
    ECHO "i is 7"
  END
  ECHO "#"
END
```

**See also:**
- **IF**
- **THEN**
- **ELSE**
- **TO**
- **DO**
- **FOR**

**FOR**

**For: For Loop**

**Syntax:**

"FOR" Ident Assign SimpleExpr
"TO" SimpleExpr "DO" StatementList "END".

**Arguments:**
none.

**Default:**
none.
**Description:**

This command starts a FOR loop.

**Example:**

```plaintext
FOR i=0 TO 10 DO
    IF i==7 THEN
        ECHO "i is 7"
    END
    ECHO "#"
END
```

**See also:**

- TO
- DO
- END

---

### format

**format: Output Format**

**Syntax:**

```
"format" assign ("motorola" | "intel" | "binary").
```

**Arguments:**

Format, either Motorola S, Intel Hex or Binary.

**Default:**

format = motorola

**Description:**

The Burner supports three different data transfer formats: Motorola S-Records, Intel Hex-Format and binary format. With the binary format the output destination must be a file. Valid identifiers are: motorola, intel, binary (the default is motorola)

**Example:**
format = binary

See also:
none.

header

header: Header File for PROM Burner

Syntax:
"header" assign <fileName>.

Arguments:
<fileName>: header file to be sent to serial port

Default:
header =

Description:
Specifies an initialization file for the PROM burner. This parameter must not be used when the burner output is redirected to a file. This file is sent byte by byte (binary) without modification to the PROM burner before anything else is sent.

This command is only used if the output is sent to a communication port.

Example:
header = "myheader.txt"

See also:
- baudRate
- parity
- header
- dataBit
- OPENCOM
**IF**

**IF: If Condition**

**Syntax:**
"IF" RelExpr "THEN" StatementList
["ELSE" StatementList] "END".

**Arguments:**
none.

**Default:**
none.

**Description:**
This command starts an IF conditional section.

**Example:**

```plaintext
FOR i=0 TO 10 DO
  IF i==7 THEN
    ECHO "i is 7"
  END
  ECHO "#"
END
```

**See also:**
- **END**
- **THEN**
- **ELSE**
len

len: Length to be Copied

Syntax:
"len" assign <number>.

Arguments:
<number>: length to be copied.

Default:
len = 0x10000

Description:
Range of program code to be copied. Length can also be specified using the ANSI-C or Modula-2 notation for hexadecimal constants (default is 0x10000).

Example:
If an application is linked between address $3000 and $4000 and the EPROM start address is $2000 (origin), then len must be set to $2000. The code will be stored at address $1000 relative to the EPROM start address.

If the EPROM start address is $3000 (origin) then len must be set to $1000. The code will be stored at the beginning of the EPROM.

Example:
len = 0x2000

See also:
- destination
- origin
OPENCOM

OPENCOM: Open Output Communication Port

Syntax:
"OPENCOM" <port>.

Arguments:

<port>: valid COM port number (1, 2, 3, 4).

Default:
none.

Description:
With this command, the Burner will send the output to the specified communication port. To close the port opened, CLOSE has to be used.

Example:
OPENCOM 2

See also:
- baudRate
- parity
- header
- dataBit
- OPENFILE
- CLOSE

OPENFILE

OPENFILE: Open Output File

Syntax:
"OPENFILE" <file>.

**Arguments:**

<file>: valid file name.

**Default:**

none.

**Description:**

With this command, the Burner will send the output to the specified file. To close the file, use `CLOSE` command.

**Example:**

OPENFILE "myFile.s19"

**See also:**

- OPENCOM
- CLOSE

---

**origin**

**origin: EEPROM Start Address**

**Syntax:**

"origin" assign <address>.

**Arguments:**

<address>: start address.

**Default:**

origin = 0

**Description:**

Initialized with the EPROM start address in the target system. The start address can be specified using ANSI C or Modula-2 notation for hexadecimal constants (default is 0).
Using Burner
Batch Burner

Example:
origin = 0xC000

See also:
- len
- destination

parity

parity: Set Communication Parity

Syntax:
"parity" assign ("none" | "even" | "odd").

Arguments:
parity none, even or odd.

Default:
parity = none

Description:
Sets the parity used for transfer. This parameter must not be used when the burner output is redirected to a file. Valid identifier values are none, odd, and even (default is none).

This command is only used if the output is sent to a communication port.

Example:
parity = even

See also:
- baudRate
- dataBit
- header
- OPENCOM
SENDBYTE

SENDBYTE: Transfer Bytes

Syntax:
"SENDBYTE" <number> <file>.

Arguments:

<number>: valid byte number (1, 2, 3, 4)
<file>: valid source file name.

Default:
none.

Description:
This command starts the transfer.
If the data format is binary, destination must be a file. The size of the file is the size specified by len divided by the busWidth. All undefined bytes are initialized with $FF or with the value specified by undefByte.
If a data bus width of 1 byte is selected, the following command must be used to transfer the code:
SENDBYTE 1 "InFile.abs"
If the data bus is 2 bytes wide, the code is split into two parts; the command SENDBYTE 1 "InFile.abs" transfers the even part of the code (corresponding to D8 to D15) while the command SENDBYTE 2 "InFile.abs" transfers the odd part, which corresponds to the LSB (D0 to D7).
If the data bus is 4 bytes wide, the command SENDBYTE 1 "InFile.abs" transfers the MSB (D24 to D31), while the command SENDBYTE 4 "InFile.abs" sends the LSB (D0 to D7).
Using 16-bit EPROMs the commands SENDWORD 1 "InFile.abs" and SENDWORD 2 "InFile.abs" must be used. If necessary, high and low byte can be swapped by initializing swapBytes with yes.

Example:
SENDBYTE 1 "myApp.abs"

See also:
- busWidth
- SENDWORD

SENDWORD

SENDWORD: Transfer Words

Syntax:
"SENDWORD" <number> <file>.

Arguments:
<number>: valid word number (1, 2)
<file>: valid source file name.

Default:
none.

Description:
This command starts the transfer.

If the data format is binary, the destination must be a file. The size of the file is the size specified by len divided by the busWidth. All undefined bytes are initialized with $FF or value specified by undefByte.

If a data bus width of 1 byte is selected, the following command must be used to transfer the code:
SENDBYTE 1 "InFile.abs"

If the data bus is 2 bytes wide, the code is split into two parts; the command SENDBYTE 1 "InFile.abs" transfers the even part of the code (corresponding to D8 to D15) while the command SENDBYTE 2 "InFile.abs" transfers the odd part, which corresponds to the LSB (D0 to D7).
If the data bus is 4 bytes wide, the command `SENDBYTE 1 "InFile.abs"` transfers the MSB (D24 to D31), while the command `SENDBYTE 4 "InFile.abs"` sends the LSB (D0 to D7).

Using 16-bit EPROMs, the commands `SENDWORD 1 "InFile.abs"` and `SENDWORD 2 "InFile.abs"` must be used. If necessary, the high and low byte can be swapped by initializing swapBytes with yes.

**Example:**

`SENDWORD 1 "myApp.abs"`

**See also:**
- `SENDBYTE`
- `busWidth`

## SLINELEN

### SLINELEN: SRecord Line Length

**Syntax:**

"SLINELEN" assign <number>.

**Arguments:**

<number>: valid line length (1, 2, ...)

**Default:**

<number> == 32.

**Description:**

This command configures how many bytes written are on a single SRECORD line. This command only effects SRECORD file generation.

**Example:**

With SLINELEN 16, the burner generates:
S113200000000000001010000000000000000000000CA
S1132010000880020820800000000001020408106B

With SLINELEN 8, the burner generates:
S10B2000000000000000000010100000D2
S10B2000000000000000000000000000CC
S10B2010000880020820800092
S10B201800000001020408109D

See also:
- format

SRECORD

SRECORD: S-Record Type

Syntax:
"SRECORD= " ("Sx" | "S1" | "S2" | "S3").

Arguments:
"Sx": Automatic choose between S1, S2 or S3 records
"S1": use S1 records
"S2": use S2 records
"S3": use S3 records

Default:
SRECORD=Sx.

Description:
This command is for Motorola S-Record output format.

Normally the Burner chooses the matching S Record type depending on the addresses used. However, with this option a certain type may be forced because the PROM burner only supports one type.

If Sx is active, the burner is in automatic mode:
if the highest address is \( \geq 0x1000000 \), then S3 records are used,
if the highest address is \( \geq 0x10000 \), then S2 records are used,
otherwise S1 records are used.

**Example:**
SRECORD=S2

**See also:**
- format

**swapByte**

**swapByte: Swap Bytes**

**Syntax:**
"swapByte" assign ("on" | "off").

**Arguments:**
"on": enables byte swapping
"off": disables byte swapping

**Default:**
swapByte = off

**Description:**
If necessary, the high and low byte can be exchanged when 16-bit or 32-bit EPROMs are used.

**Example:**
swapByte = on

**See also:**
- busWidth
THEN

THEN: Statementlist for If Condition

Syntax:
"IF" RelExpr "THEN" StatementList
["ELSE" StatementList] "END".

Arguments:
none.

Default:
none.

Description:
This command starts an IF conditional section.

Example:

FOR i=0 TO 10 DO
   IF i==7 THEN
      ECHO "i is 7"
   END
   ECHO "#"
END

See also:
- END
- IF
- ELSE
TO

TO: For Loop End Condition

Syntax:
"FOR" Ident Assign SimpleExpr
"TO" SimpleExpr "DO" StatementList "END".

Arguments:
none

Default:
none

Description:
Specifies the FOR loop end condition. As ident, only ‘i’ may be used, and each occurrence of # in the loop is replaced with the actual value of ‘i’.

Example:
FOR i=0 TO 10 DO
  ECHO "#"
END

See also:
- FOR
- DO
- END

undefByte

undefByte: Fill Byte for Binary Files

Syntax:
"undefByte" assign <number>.

**Arguments:**

<number>: 8bit number

**Default:**

undefByte = 0xFF

**Description:**

This command assigns the default fill byte to undefined bytes in binary output files. This command is only used for binary files.

**Example:**

undefByte = 0x33

**See also:**

- format

---

**PAUSE**

**PAUSE: Wait until Key Pressed**

**Syntax:**

"PAUSE" [<string>]

**Arguments:**

<string>: a string written to output window

**Default:**

none

**Description:**

This command causes the batch burner language program to wait until a key is pressed. An optional message text may be specified. For Windows 95/98/NT, a dialog box will appear:
Burner Options

The Burner offers a number of options that you can use to control the application. Options are composed of a minus/dash (‘-’) followed by one or more letters or digits.

Command line options are not case sensitive, e.g. "–F" is the same as "–f".

The burner command line can contain the name of a file to be built with the option -f, or a list of commands.

Options before the first command on the command line are recognized. Then, all remaining text is taken as arguments to the command, including options.

For example:

'OPENFILE "fibo.out" format=motorola len=0x1000 SENDBYTE 1 "fibo.abs.abs" CLOSE'

Command is executed.

-f=fibo.bbl

Command file fibo.bbl is executed.

-f fibo.bbl

This is an alternate form of the recommended "-f=fibo.bbl". This form is allowed for compatibility only.

The following is not allowed:
fibo.bbl -f

fibo.bbl is interpreted as a command with argument -f. This generates an error, since no such command exists.

Option Details

Option Groups

Options are grouped by:

HOST, INPUT, MESSAGE and VARIOUS

<table>
<thead>
<tr>
<th>Group</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST</td>
<td>Lists options related to host.</td>
</tr>
<tr>
<td>INPUT</td>
<td>Lists options related to input file.</td>
</tr>
<tr>
<td>MESSAGES</td>
<td>Lists options that generate error messages.</td>
</tr>
<tr>
<td>VARIOUS</td>
<td>Lists various options.</td>
</tr>
</tbody>
</table>

The group corresponds to the property sheets of the graphical option settings.

**NOTE** Not all command line options are accessible through the property sheets, for example, -H or -Lic.

Option Detail Description

The remainder of this section describes options available for the application. Options are listed in alphabetical order and divided into several sections.

<table>
<thead>
<tr>
<th>Topic</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Group</td>
<td>HOST, INPUT, MESSAGE or VARIOUS.</td>
</tr>
<tr>
<td>Syntax</td>
<td>Specifies syntax of option in EBNF format.</td>
</tr>
<tr>
<td>Arguments</td>
<td>Describes and lists optional and required arguments for the option.</td>
</tr>
<tr>
<td>Default</td>
<td>Shows default setting for option.</td>
</tr>
</tbody>
</table>
**Using Burner**

**Burner Options**

<table>
<thead>
<tr>
<th>Topic</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td>Provides a detailed description of the option and how to use it.</td>
</tr>
<tr>
<td><strong>Example</strong></td>
<td>Gives an example of usage, and effects of the option where possible.</td>
</tr>
<tr>
<td><strong>See also</strong></td>
<td>Names related topics.</td>
</tr>
</tbody>
</table>

**-D**

**-D: Display Dialog Box**

**Group:**

VARIOUS

**Syntax:**

"-D".

**Arguments:**

none.

**Default:**

none.

**Description:**

This option displays the Burner dialog box. This allows you to launch the burner from a make file and await user input.
Example:
burner.exe -D

See also:
none.

-Env

-Env: Set Environment Variable

Group:
HOST
Using Burner
Burner Options

**Syntax:**
"-Env" <Environment Variable> "=" <Variable Setting>.

**Arguments:**
- `<Environment Variable>`: Environment variable to be set
- `<Variable Setting>`: Value to be assigned

**Default:**
none.

**Description:**
This option sets an environment variable. This environment variable may be used in the maker or used to overwrite system environment variables.

**Example:**
-EnvOBJPATH=\sources\obj
This is the same as
OBJPATH=\sources\obj
in the default.env file

**See also:**
none.

**-F**

**-F: Execute Command File**

**Group:**
INPUT

**Syntax:**
"-F=" <fileName>.

**Arguments:**
<fileName>: Batch Burner command file to be executed.

Default:
none.

Description:
This option causes the Burner to execute a Batch Burner command file (usual extension is .bbl).

Example:
-F=fibo.bbl

See also:
none.

-H

-H: Short Help

Group:
VARIOUS

Syntax:
"-H".

Arguments:
none.

Default:
none.

Description:
The -H option displays a short list (i.e. help list) of available options within main window.
No other option or source file should be specified with the \(-H\) option.

**Example:**

\(-H\) may produce following list:

... 

VARIOUS:

\(-H\) Prints this list of options

\(-V\) Prints the Compiler version

...

**See also:**

none.

**-Lic**

**-Lic: License Information**

**Group:**

VARIOUS

**Syntax:**

"-Lic".

**Arguments:**

none.

**Default:**

none.

**Description:**

The \(-Lic\) option prints the current license information (e.g. if it is a demo version or a full version). This information is also displayed in the about box.

**Example:**
-Lic

See also:

• Option -LicA

-LicA

-LicA: License Information about every Feature in Directory

Group:
VARIOUS

Syntax:
"-LicA".

Arguments:
none.

Default:
none.

Description:
The -LicA option prints the license information of every tool or .dll in the directory containing the executable (e.g. if tool or feature is demo or full version). Because the option analyzes every file in the directory, this may take a long time.

Example:
-LicA

See also:

• Option -Lic
-N

-N: Display Notify Box

Group:
MESSAGE

Syntax:
"-N".

Arguments:
none.

Default:
none.

Description:
Causes the application to display an alert box if there was an error during the process. This is useful when running a make file (refer to Make Utility) since the tool waits for the user to acknowledge the message, thus suspending make file processing. (The 'N' stands for “Notify”.)

This feature is useful for halting and aborting a build using the Make Utility.

NOTE
This option is only present in PC versions.

Example:
-Fnofile -N

If an error occurs during processing, a dialog box similar to the following one will appear:
Using Burner
Burner Options

See also:
none.

-NoBeep

-NoBeep: No Beep in Case of an Error

Group:
MESSAGE

Syntax:
"-NoBeep".

Arguments:
none.

Default:
none.

Description:
Normally there is a ‘beep’ at the end of processing, if an error occurs. Use this option to turn off the beep.

Example:
-NoBeep

See also:
none.

-NoEnv

-NoEnv: Do not use Environment

Group:
Startup. (This option can not be specified interactively)

Syntax:
"-NoEnv".

Arguments:
none.

Default:
none.

Description:
This option can only be specified on the command line, while starting the application. It can not be specified in an environment file, such as the default.env file.

When this option is used, the application does not use an environment file, such as default.env, project.ini or tips file.

Example:
burner.exe -NoEnv

See also:
- Section Configuration File
-Ns

-Ns: No S-Records

Group:
OUTPUT

Syntax:
"-Ns" [ "=" { "p" | "0" | "7" | "8" | "9" } ].

Arguments:
"p": no path in S0 record
"0": no S0 record
"7": no S7 record
"8": no S8 record
"9": no S9 record

Default:
none.

Description:
Usually a S-Record file contains a S0-Record at the beginning that contains the name of the file and a S7, S8 or S9 record at the end, depending on the address size. For the S3 format, a S7 record is written at the end. For S2 format, a S8 record is written at the end. For the S1 format, a S9 record is written at the end.

This feature is useful for disabling some S-Record generation in case a non-standard S-Record file reader cannot read S0, S7, S8 or S9 records.

In case the option is specified without suboptions (only -Ns), no start (S0) and no end records (S7, S8 or S9) are generated.

The option -Ns=p will remove the path (if present) from the file name in the S0 record.

Example:
-Ns=0

See also:
- SRECORD.

-Prod

-Prod: specify project file at startup

Group:
none. (this option cannot be specified interactively)

Syntax:
"-Prod=" <file>.

Arguments:
<file>: name of project or project directory

Default:
none.

Description:
This option can only be specified on the command line while starting the application. It can not be specified in any other situation, including the default.env file.

When this option is used, the application opens the specified file as a configuration file. If a directory is specified, the default name project.ini is appended. If loading fails, a message box appears.

Example:
burner.exe -prod=project.ini

See also:
- Section Configuration File
-V

-V: Prints Version Information

Group:
VARIOUS

Syntax:
"-V".

Arguments:
none.

Default:
none.

Description:
Prints the application version and versions of modules internal to the application, and the current directory.

NOTE
This option is useful to determine the current directory of the application. This option is NOT present in the dialog box.

Example:
-V produces the following list:

Directory: \software\sources
Common Module V-5.0.7, Date Apr 12 1999
User Interface Module, V-5.0.18, Date Apr 8 1999

See also:
none.
-View

-View: Application Standard Occurrence

Group:
HOST

Syntax:
"-View" <kind>.

Arguments:
<kinds> is one of:
  "Window": Default window size
  "Min": Application window is minimized
  "Max": Application window is maximized
  "Hidden": Application window is not visible (only if arguments)

Default:
Application started with arguments: Minimized.
Application started without arguments: Window.

Description:
The application (e.g. linker, compiler, ...) is started with default window, if no
arguments are given. If the application is started with arguments (e.g. from the maker
to compile/link a file) then the application is minimized to allow batch processing.
However, with this option the window mode may be specified.

Use -ViewWindow to display application in its normal window. Use -ViewMin to
start application as an icon in the task bar.

Use -ViewMax to start application window maximized.

Use -ViewHidden for the application to process arguments (e.g. files to be compiled/
linked) in the back ground (no window/icon visible). If you use the -N option, a dialog
box is still possible.
NOTE  This option is only present on the IBM PC version.

Example:

c:\Metrowerks\prog\burner.exe -ViewHidden fibo.bbl

See also:

none.

-W

-W: Display Window

Group:

VARIOUS

Syntax:

"-W".

Arguments:

none.

Default:

none.

Description:

In the V2.7 Burner, this option was used to show the Batch Burner Window. This option is ignored with the V5.x versions or later. This option will be removed in a future release.

NOTE  This option is only provided for compatibility reasons. This option is NOT present in the dialog box.

Example:
burner.exe -W

See also:
none.

-Wmsg8x3

-Wmsg8x3: Cut file names in Microsoft format to 8.3

Group:
MESSAGE

Syntax:
"-Wmsg8x3".

Arguments:
none.

Default:
none.

Description:
Some editors (e.g. early versions of WinEdit) are expecting the file name in the Microsoft message format in a strict 8.3 format, which means the file name can have a maximum of 8 characters with a maximum 3 character extension. Longer file names are possible with Win95 or WinNT.

This option causes the file name in the Microsoft message to be truncated to the 8.3 format.

Example:
x:\mysourcefile.bbl(3): INFORMATION B1000: message text

With the option -Wmsg8x3 set, the above message will be
x:\mysource.bbl(3): INFORMATION C2901: message text

See also:
Using Burner
Burner Options

- Option -WmsgFi
- Option -WmsgFb

-WErrFile

-WErrFile: Create "err.log" Error File

Group:
MESSAGE

Syntax:
"-WErrFile" ("On" | "Off").

Arguments:
none.

Default:
err.log is created/deleted.

Description:
A return code provides error feedback from the application to called tools. In a 16 bit windows environment, this was not possible, so a file "err.log" was used to store errors. To state no error, the file "err.log" was deleted. With UNIX or WIN32 applications, a return code is available, so the error file is no longer needed. To use a 16 bit maker with this tool, the error file must be created in order to signal an error.

Example:
-WErrFileOn
err.log is created/deleted when the application is finished.
-WErrFileOff
existing err.log is not modified.

See also:
- Option -WStdout
-WmsgCE

-WmsgCE: RGB color for error messages

**Group:**
MESSAGE

**Scope:**
Function

**Syntax:**
"-WmsgCE" <RGB>.

**Arguments:**
<RGB>: 24bit RGB (red green blue) value.

**Default:**
-WmsgCE16711680 (rFF g00 b00, red)

**Defines:**
none.

**Description:**
This option is used to change the error message color. The value is specified as a RGB (Red-Green-Blue) value in decimal format.

**Example:**
-WmsgCE255 changes error messages to blue.

**See also:**
none.
-WmsgCF

-WmsgCF: RGB color for fatal messages

**Group:**
MESSAGE

**Scope:**
Function

**Syntax:**
"-WmsgCF" <RGB>.

**Arguments:**

<RGB>: 24bit RGB (red green blue) value.

**Default:**
-WmsgCF8388608 (r80 g00 b00, dark red)

**Defines:**
none.

**Description:**
This option specifies the fatal message color. The value is specified as a RGB (Red-Green-Blue) value in decimal format.

**Example:**
-WmsgCF255 changes fatal messages to blue.

**See also:**
none.
-WmsgCI

-WmsgCI: RGB color for information messages

Group:
MESSAGE

Scope:
Function

Syntax:
"-WmsgCI" <RGB>.

Arguments:
<RGB>: 24bit RGB (red green blue) value.

Default:
-WmsgCI32768 (r00 g80 b00, green)

Defines:
none.

Description:
This option specifies the information message color. The value is specified as a RGB (Red-Green-Blue) value in decimal format.

Example:
-WmsgCI255 changes information messages to blue.

See also:
none.
-WmsgCU

-WmsgCU: RGB color for user messages

**Group:**
MESSAGE

**Scope:**
Function

**Syntax:**
"-WmsgCU" \(<RGB>\).

**Arguments:**
\(<RGB>\): 24bit RGB (red green blue) value.

**Default:**
-WmsgCU0 (r00 g00 b00, black)

**Defines:**
none.

**Description:**
This option specifies the user message color. The value is specified as a RGB (Red-Green-Blue) value in decimal format.

**Example:**
-WmsgCU255 changes user messages to blue.

**See also:**
none.
-WmsgCW

-WmsgCW: RGB color for warning messages

Group:
MESSAGE

Scope:
Function

Syntax:
"-WmsgCW" <RGB>.

Arguments:
<RGB>: 24bit RGB (red green blue) value.

Default:
-WmsgCW255 (r00 g00 bFF, blue)

Defines:
none.

Description:
This option specifies the warning message color. The value is specified as a RGB (Red-Green-Blue) value in decimal format.

Example:
-WmsgCW0 changes warning messages to black.

See also:
none.
-WmsgFb (-WmsgFbi, -WmsgFbm)

-WmsgFb: Set message file format for batch mode

Group:
MESSAGE

Syntax:
"-WmsgFb" ["v" | "m"].

Arguments:
"v": Verbose format.
"m": Microsoft format.

Default:
-WmsgFbm

Description:
The Application can be started with additional arguments (e.g. files to be processed together with options). If the Application has been started with arguments (e.g. from the Make Tool or with the ‘%f’ argument from WinEdit), the application processes the files in a batch mode. No application window is visible and the application terminates after job completion.

If the application is in batch mode, messages are written to a file instead of the screen. This file only contains the application messages (see examples below).

By default, the application uses a Microsoft message format to write the messages (errors, warnings, information messages) if the application is in batch mode.

With this option, the default format may be changed from the Microsoft format (only line information) to a more verbose error format with line, column and source information.

NOTE

Using the verbose message format may slow down processing, because the tool has to write more information into the message file.

Example:
By default, the application may produce following file if it is running in batch mode (e.g. started from the Make tool):

X:\C.bbl(3): INFORMATION B2901: Message

Setting the format to verbose stores more information in the file:

- WmsgFbv

>> in "X:\C.bbl", line 3, col 2, pos 33
  some text
^  
INFORMATION B2901: Message

See also:
- Environment variable ERRORFILE
- Option -WmsgFi

-WmsgFi (-WmsgFiv, -WmsgFim)

-WmsgFi: Set message format for interactive mode

Group:  
MESSAGE

Syntax:  
"-WmsgFi" ["v" | "m"].

Arguments:  
"v": Verbose format.
"m": Microsoft format.

Default:  
-WmsgFiv

Description:  
If the application is started without additional arguments (e.g. files to be processed and options), the application is in interactive mode (window is visible).
By default, the application uses the verbose error file format to write messages (errors, warnings, information messages).

With this option, the default format may be changed from the verbose format (with source, line and column information) to the Microsoft format (only line information).

NOTE  Using the Microsoft format may speed up processing, because the application has to write less information to the screen.

Example:
By default, the application may produce the following error output in the application window if it is running in interactive mode:
Top: X:\C.bbl

>> in "X:\C.bbl", line 3, col 2, pos 33
   some text
   ^
INFORMATION B2901: Message

Setting the format to Microsoft, less information is displayed:
-WmsgFim
Top: X:\C.bbl
X:\C.bbl(3): INFORMATION B2901: Message

See also:
- Environment variable ERRORFILE
- Option -WmsgFb

-WmsgFob

-WmsgFob: Message format for Batch Mode

Group:
MESSAGE

Syntax:
"-WmsgFob"<string>.

**Arguments:**

<string>: format string (see below).

**Default:**

-WmsgFob"%f%e(%l): %K %d: %m
"

**Description:**

With this option it is possible to modify the default message format in batch mode. Following formats are supported (example source file x:\Metrowerks\mysourcefile.cpph)

<table>
<thead>
<tr>
<th>Format</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>%s</td>
<td>Source Extract</td>
<td>x:\Metrowerks\</td>
</tr>
<tr>
<td>%p</td>
<td>Path</td>
<td>x:\Metrowerks\mysourcefile</td>
</tr>
<tr>
<td>%f</td>
<td>Path and name</td>
<td>x:\Metrowerks\mysourcefile</td>
</tr>
<tr>
<td>%n</td>
<td>File name</td>
<td>mysourcefile</td>
</tr>
<tr>
<td>%e</td>
<td>Extension</td>
<td>.cpph</td>
</tr>
<tr>
<td>%N</td>
<td>File (8 chars)</td>
<td>mysource</td>
</tr>
<tr>
<td>%E</td>
<td>Extension (3 chars)</td>
<td>.cpp</td>
</tr>
<tr>
<td>%l</td>
<td>Line</td>
<td>3</td>
</tr>
<tr>
<td>%c</td>
<td>Column</td>
<td>47</td>
</tr>
<tr>
<td>%o</td>
<td>Pos</td>
<td>1234</td>
</tr>
<tr>
<td>%K</td>
<td>Uppercase kind</td>
<td>ERROR</td>
</tr>
<tr>
<td>%k</td>
<td>Lowercase kind</td>
<td>error</td>
</tr>
<tr>
<td>%d</td>
<td>Number</td>
<td>B1815</td>
</tr>
<tr>
<td>%m</td>
<td>Message</td>
<td>text</td>
</tr>
<tr>
<td>%</td>
<td>Percent</td>
<td>%</td>
</tr>
<tr>
<td>\n</td>
<td>New line</td>
<td></td>
</tr>
</tbody>
</table>

**Example:**

-WmsgFob"%f%e(%l): %K %d: %m\n"

produces a message in following format:

X:\C.C(3): information B2901: Message

**See also:**

- [Environment variable ERRORFILE](#)
-WmsgFoi

-WmsgFoi: Message Format for Interactive Mode

**Group:**
MESSAGE

**Syntax:**
"-WmsgFoi"<string>.

**Arguments:**
<string>: format string (see below).

**Default:**
-WmsgFoi"\n>> in "%f%e", line %l, col %c, pos %o\n%s\n%K %d:
%m\n"

**Description:**
With this option it is possible to modify the default message format in interactive mode. Following formats are supported (example source file x:\Metrowerks\mysourcefile.cpph):

<table>
<thead>
<tr>
<th>Format</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>%s</td>
<td>Source Extract</td>
<td>x:\Metrowerks\</td>
</tr>
<tr>
<td>%p</td>
<td>Path</td>
<td>x:\Metrowerks\mysourcefile</td>
</tr>
<tr>
<td>%f</td>
<td>Path and name</td>
<td>x:\Metrowerks\mysourcefile</td>
</tr>
<tr>
<td>%n</td>
<td>File name</td>
<td>mysourcefile</td>
</tr>
<tr>
<td>%e</td>
<td>Extension</td>
<td>.cpph</td>
</tr>
<tr>
<td>%N</td>
<td>File (8 chars)</td>
<td>mysource</td>
</tr>
<tr>
<td>%E</td>
<td>Extension (3 chars)</td>
<td>.cpp</td>
</tr>
<tr>
<td>%l</td>
<td>Line</td>
<td>3</td>
</tr>
<tr>
<td>%c</td>
<td>Column</td>
<td>47</td>
</tr>
</tbody>
</table>
Using Burner
Burner Options

%o Pos 1234
%K Uppercase kind ERROR
%k Lowercase kind error
%d Number B1815
%m Message text
% Percent
\n New line

Example:
-WmsgFoi"%f%e(\l): %k %d: %m\n"
produces a message in following format:
X:\C.C(3): information B2901: Message

See also:
- Environment variable ERRORFILE
- Option -WmsgFb
- Option -WmsgFi
- Option -WmsgFonp
- Option -WmsgFob

-WmsgFonf

-WmsgFonf: Message Format for no File Information

Group:
MESSAGE

Syntax:
"-WmsgFonf"<string>.

Arguments:
<string>: format string (see below).

Default:
Using Burner

Burner Options

-\texttt{WmsgFonf}"\%K \%d: \%m\\n"

**Description:**

Sometimes there is no file information available for a message (e.g. if a message is not related to a specific file). Then this message format string is used. Following formats are supported:

<table>
<thead>
<tr>
<th>Format</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>%K</td>
<td>Uppercase kind</td>
<td>ERROR</td>
</tr>
<tr>
<td>%k</td>
<td>Lowercase kind</td>
<td>error</td>
</tr>
<tr>
<td>%d</td>
<td>Number</td>
<td>B1815</td>
</tr>
<tr>
<td>%m</td>
<td>Message</td>
<td>text</td>
</tr>
<tr>
<td>%</td>
<td>Percent</td>
<td>%</td>
</tr>
<tr>
<td>\n</td>
<td>New line</td>
<td></td>
</tr>
</tbody>
</table>

**Example:**

-\texttt{WmsgFonf}"\%k \%d: \%m\\n"

produces a message in following format:

information B1034: Message

See also:

- [Environment variable ERRORFILE](#)
- [Option -WmsgFb](#)
- [Option -WmsgFi](#)
- [Option -WmsgFonp](#)
- [Option -WmsgFoi](#)

-\texttt{WmsgFonp}

-\texttt{WmsgFonp}: Message Format for no Position Information

**Group:**

MESSAGE
Using Burner
Burner Options

Syntax:
"-WmsgFonp"<string>.

Arguments:
<string>: format string (see below).

Default:
-WmsgFonp"%f%e: %K %d: %m\n"

Description:
Sometimes there is no position information available for a message (e.g. if a message is not related to a certain position). Then this message format string is used. Following formats are supported:

<table>
<thead>
<tr>
<th>Format</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>%K</td>
<td>Uppercase kind</td>
<td>ERROR</td>
</tr>
<tr>
<td>%k</td>
<td>Lowercase kind</td>
<td>error</td>
</tr>
<tr>
<td>%d</td>
<td>Number</td>
<td>B1815</td>
</tr>
<tr>
<td>%m</td>
<td>Message</td>
<td>text</td>
</tr>
<tr>
<td>%</td>
<td>Percent</td>
<td>%</td>
</tr>
<tr>
<td>\n</td>
<td>New line</td>
<td></td>
</tr>
</tbody>
</table>

Example:
-WmsgFonf"%k %d: %m\n"

produces a message in following format:
information B1324: Message

See also:
- Environment variable ERRORFILE
- Option -WmsgFb
- Option -WmsgFi
- Option -WmsgFonp
- Option -WmsgFoi
-WmsgNe

-WmsgNe: Number of Error Messages

Group:
MESSAGE

Syntax:
"-WmsgNe" <number>.

Arguments:
<number>: Maximum number of error messages.

Default:
50

Description:
Use this option to set the number of errors allowed until the application stops processing.

Example:
-WmsgNe2
The application stops after two error messages.

See also:
- Option -WmsgNi
- Option -WmsgNw

-WmsgNi

-WmsgNi: Number of Information Messages

Group:
MESSAGE
Using Burner

Burner Options

Syntax:
"-WmsgNi" <number>

Arguments:

<number>: Maximum number of information messages.

Default:
50

Description:
Use this option to set the number of information messages.

Example:
-WmsgNi10

Only ten information messages are logged.

See also:

- Option -WmsgNe
- Option -WmsgNw

-WmsgNu

-WmsgNu: Disable User Messages

Group:
MESSAGE

Syntax:

"-WmsgNu" ["=" {"a" | "b" | "c" | "d"}].

Arguments:

"a": Disable messages that relate to include files

"b": Disable messages that relate to files being read (input files)

"c": Disable messages that relate to generated files
“d”: Disable messages that relate to processing statistics (code size, RAM/ROM usage, etc.)

“e”: Disable informal messages

**Default:**

none.

**Description:**

Some messages produced by the application are not in the normal message categories (WARNING, INFORMATION, ERROR, FATAL). With this option such messages can be disabled. This option reduces the number of messages and simplifies error parsing of other tools.

**NOTE**

Depending on the application, not all suboptions may make sense. In this case they are just ignored for compatibility.

**Example:**

-WmsgNu=c

**See also:**

none.

**-WmsgNw**

- **WmsgNw: Number of Warning Messages**

**Group:**

MESSAGE

**Syntax:**

"-WmsgNw" <number>.

**Arguments:**

<number>: Maximum number of warning messages.
**Burner Options**

**Default:**

50

**Description:**

Sets the number of warning messages.

**Example:**

`-WmsgNw15`

Only 15 warning messages are logged.

**See also:**

- Option `-WmsgNe`
- Option `-WmsgNi`

**-WmsgSd**

**-WmsgSd: Setting a Message to Disable**

**Group:**

MESSAGE

**Syntax:**

"-WmsgSd" <number>.

**Arguments:**

<number>: Message number to be disabled, e.g. 1801

**Default:**

none.

**Description:**

With this option a message can be disabled, so it does not appear in the error output.

**Example:**
-WmsgSd1801

disables the message for implicit parameter declaration.

See also:

- Option -WmsgSi
- Option -WmsgSw
- Option -WmsgSe

-WmsgSe

-WmsgSe: Setting a Message to Error

Group:
MESSAGE

Syntax:
"-WmsgSe" <number>.

Arguments:
<number>: Message number to be set as an error message, e.g. 1853

Default:
none.

Description:
Changes message category for a message to error message category.

Example:
-WmsgSe1853

See also:

- Option -WmsgSd
- Option -WmsgSi
- Option -WmsgSw
-WmsgSi

-WmsgSi: Setting a Message to Information

Group:
MESSAGE

Syntax:
"-WmsgSi" <number>.

Arguments:
<number>: Message number to be set as an information message, e.g. 1853

Default:
none.

Description:
Changes category for message to information message category.

Example:
-WmsgSi1853

See also:
• Option -WmsgSd
• Option -WmsgSw
• Option -WmsgSe

-WmsgSw

-WmsgSw: Setting a Message to Warning

Group:
MESSAGE
Using Burner
Burner Options

Syntax:
"-WmsgSw" <number>.

Arguments:
<number>: Message number to be set as a warning, e.g. 2901

Default:
none.

Description:
Sets message category for a message to warning message.

Example:
-WmsgSw2901

See also:
- Option -WmsgSd
- Option -WmsgSi
- Option -WmsgSe

-WOutFile

-WOutFile: Create Error Listing File

Group:
MESSAGE

Syntax:
"-WOutFile" ("On" | "Off").

Arguments:
none.

Default:
Error list file is created.

**Description:**
This option controls creation of an error log file. The file contains a list of all messages and errors encountered during processing. Since text error feedback can also be handled with pipes to the calling application, it is possible to obtain this feedback without an explicit file. The name of the list file is controlled by the environment variable `ERRORFILE`.

**Example:**
- `-WOutFileOn`
  The error file specified by `ERRORFILE` is created.
- `-WOutFileOff`
  No error file is created.

**See also:**
- Option `-WErrFile`
- Option `-WStdout`

**-WStdout**

**-WStdout: Write to standard output**

**Group:**
MESSAGE

**Syntax:**
```
"-WStdout" ("On" | "Off").
```

**Arguments:**
none.

**Default:**
output is written to stdout.
**Description:**

With Windows applications, the usual standard streams are available. But text written into them do not appear anywhere unless explicitly requested by the calling application. With this option text written to an error file can also be written to stdout.

**Example:**

-WStdoutOn

All messages are written to stdout.

-WErrFileOff

Nothing is written to stdout.

**See also:**

- Option -WErrFile
- Option -WOutFile

-W1

-W1: No Information Messages

**Group:**

MESSAGE

**Syntax:**

"-W1".

**Arguments:**

none.

**Default:**

none.

**Description:**

Excludes INFORMATION messages, only WARNING and ERROR messages are generated.
Example:

-\(W1\)

See also:

- Option -\(WmsgNi\)

\(-W2\)

\(-W2\): No Information and Warning Messages

Group:

MESSAGE

Syntax:

"-\(W2\)"

Arguments:

none.

Default:

none.

Description:

Suppresses all INFORMATION and WARNING messages, only ERRORs are generated.

Example:

-\(W2\)

See also:

- Option -\(WmsgNi\)
- Option -\(WmsgNw\)
Environment

This section describes environment variables used by the application. Some environment variables are also used by other tools (e.g. Linker), for related information refer to their respective manual.

Various parameters may be set in an environment using environment variables. The syntax is:

Parameter = KeyName "=" ParamDef.

NOTE Normally no blanks are allowed in the definition of an environment variable.

Example:

GENPATH=C:\INSTALL\LIB;D:\PROJECTS\TESTS;/usr/local/lib;/home/me/my_project

Parameters may be defined in several ways:

- Using system environment variables supported by your operating system.
  Put definitions in a file called DEFAULT.ENV (.hidefaults for UNIX) in the default directory.

  NOTE The maximum length of environment variable entries in the DEFAULT.ENV or .hidefaults is 4096 characters.

- Put definitions in a file specified by the system environment variable ENVIRONMENT.

  NOTE The default directory mentioned above can be set by the system environment variable DEFAULTDIR.

When looking for an environment variable, all programs first search the system environment, then the DEFAULT.ENV (.hidefaults for UNIX) file and finally the global environment file given by ENVIRONMENT. If no definition can be found, a default value is assumed.

NOTE The environment may also be changed using the -Env option.
NOTE
Ensure that no spaces are at the end of environment variables.

The Current Directory
The most important environment for all tools is the current directory. The current directory is the base search directory where the tool starts to search for files (e.g. DEFAULT.ENV or .hidefaults).

Normally, the current directory of an executed tool is determined by the operating system or program that launches another one, for example WinEdit.

For the UNIX operating system, the current directory of an executable started is also the current directory from where the binary file has been started.

For MS Windows operating systems, the current directory definition is quite complex:

- If the tool is launched using a File Manager/Explorer, the current directory is the location of the executable launched.
- If the tool is launched using an Icon on the Desktop, the current directory is the one specified and associated with the Icon.
- If the tool is launched by dragging a file on the icon of the executable under Windows 95 or Windows NT 4.0, the desktop is the current directory.
- If the tool is launched by another launching tool with its own current directory specification (e.g. WinEdit), the current directory is the one specified by the launching tool (e.g. current directory definition in WinEdit).
- The current directory is where the local project file is located. Changing the current project file also changes the current directory, if the other project file is in a different directory. Note that browsing for a C file does not change the current directory.

To overwrite this behavior, the environment variable DEFAULTDIR may be used.

The current directory is displayed along with other information in the about box or with the option “-v”.

Global Initialization File (MCUTOOLS.INI) (PC only)
All tools may store global data in MCUTOOLS.INI. A tool searches for this file in the directory of the tool itself (path of the executable). If there is no MCUTOOLS.INI file in this directory, the tool looks for a MCUTOOLS.INI file located in the MS Windows installation directory (e.g. C:\WINDOWS).
Example:
C:\WINDOWS\MCUTOOLS.INI
D:\INSTALL\PROG\MCUTOOLS.INI

If a tool is started in D:\INSTALL\PROG directory, the project file in the same directory as the tool is used (D:\INSTALL\PROG\MCUTOOLS.INI).

However, if the tool is started outside the D:\INSTALL\PROG directory, the project file in the Windows directory is used (C:\WINDOWS\MCUTOOLS.INI).

The following section provides a short description of entries in the MCUTOOLS.INI file:

[Installation] Section

Path

Arguments:
Last installation path.

Description:
When a tool is installed, the installation script stores the installation destination directory in this variable.

Example:
Path=c:\install

Group

Arguments:
Last installation program group.

Description:
When a tool is installed, this variable stores the installation program group created.

Example:
Group=Burner

[Options] Section

DefaultDir

Arguments:
Default Directory to be used.

Description:
Specifies the current directory for all tools on a global level (see also, environment variable DEFAULTDIR).

Example:
DefaultDir=c:\install\project

[Burner] Section

SaveOnExit

Arguments:
1/0

Description:
1 if the configuration should be stored when the tool is closed, 0 if it should not be stored. The tool does not ask to store a configuration, in either case.

SaveAppearance

Arguments:
1/0

Description:
1 if the visible topics should be stored when writing a project file, 0 if not. The command line, its history, the windows position and other topics belong to this entry.
SaveEditor

Arguments:
1/0

Description:
1 if the visible topics should be stored when writing a project file, 0 if not. The editor setting contains all information from the editor configuration dialog.

SaveOptions

Arguments:
1/0

Description:
1 if the options should be contained when writing a project file, 0 if not. The options also contain the message settings.

RecentProject0, RecentProject1, ...

Arguments:
names of last and prior project files

Description:
This list is updated when a project is loaded or saved. Its current content is shown in the file menu.

Example:

SaveOnExit=1
SaveAppearance=1
SaveEditor=1
SaveOptions=1
RecentProject0=C:\myprj\project.ini
RecentProject1=C:\otherprj\project.ini
[Editor] Section

Editor_Name

Arguments:
The name of the global editor

Description:
Specifies the name displayed for the global editor. This entry provides only a description. Its content is not used to start the editor.
This entry cannot be modified with the tool.

Editor_Exe

Arguments:
The name of executable file for global editor

Description:
Specifies file called when the global editor setting is active. In the editor configuration dialog, the global editor selection is active when this entry is present and not empty.

Editor_Opts

Arguments:
Options to use the global editor

Description:
Specifies options that should be used with the global editor. If this entry is not present or empty, “%f” is used. The command line to launch the editor is built by taking the Editor_Exe content, then appending a space followed by this entry.

Example:
[Editor]
editor_name=WinEdit
Using Burner
Environment

editor_exe=C:\Winedit\WinEdit.exe
editor_opts=%f

Example

The following example shows a typical layout of the MCUTOOLS.INI:

[Installation]
Path=c:\Metrowerks
Group=Burner

[Editor]
editor_name=WinEdit
editor_exe=C:\Winedit\WinEdit.exe
editor_opts=%f

[Options]
DefaultDir=c:\myprj

[Burner]
SaveOnExit=1
SaveAppearance=1
SaveEditor=1
SaveOptions=1
RecentProject0=c:\myprj\project.ini
RecentProject1=c:\otherprj\project.ini

Local Configuration File (usually project.ini)

The tools does not change the default.env file. Its content is only read. All configuration properties are stored in the configuration file. The same configuration file can be used by different applications.

The shell uses the configuration file with the name "project.ini" in the current directory only, that is why this name is also suggested to be used with the tool. Only when the shell uses the same file as the tool, the editor configuration written and maintained by the shell can be used by the tool. Apart from this, the tool can use any file name for the project file.

The configuration file has the same format as windows .ini files. The application stores its own entries with the same section name as in the global mcutools.ini file. The application backend is encoded into the section name, so that different application
backends can use the same file without overlapping. Different versions of the same tool use the same entries. This plays a role when options only available in one version should be stored in the configuration file. In such situations, two files must be maintained for different tool versions. If no incompatible options are enabled when the file is last saved, the same file may be used for both tool versions.

The current directory is always the directory containing the configuration. If a configuration file in a different directory is loaded, then the current directory also changes. When the current directory changes, the default.env file is reloaded.

At startup there are two ways to load a configuration:

- use the command line option -Prod
- from the project.ini file in the current directory

If the option -Prod is used, then the current directory is the directory containing the project file. If the option -Prod is used with a directory, the file project.ini in this directory is loaded.

[Editor] Section

Editor_Name

Arguments:
Name of the local editor

Description:
Specifies the name displayed for the local editor. Its content is not used to start the editor.

Saved:
This entry has the same format as global editor configuration in the mcutools.ini file.

Editor_Exe

Arguments:
Name of executable file for local editor

Description:
Specifies file when the local editor setting is active. In the editor configuration dialog, the local editor selection is only active when this entry is present and not empty.

**Saved:**
This entry has the same format as the global editor configuration in the mcutools.ini file.

**Editor_Opts**

**Arguments:**
Options to use with the local editor

**Description:**
Specifies options to use with local editor. If this entry is not present or empty, "%f" is used. The command line to launch the editor is built by taking the Editor_Exe content, then appending a space followed by this entry.

**Saved:**
This entry has the same format as the global editor configuration in the mcutools.ini file.

**Example:**

```ini
[Editor]
editor_name=WinEdit
editor_exe=C:\Winedit\WinEdit.exe
editor_opts=%f
```

**[Burner] Section**

**RecentCommandLineX, X= integer**

**Arguments:**
String with a command line history entry, e.g. “fibo.bbl”

**Description:**
This list of entries contains the content of the command line history.
Saved:
Only with Appearance set in the File->Configuration Save Configuration dialog.

CurrentCommandLine

Arguments:
String with the command line, e.g. "-ffibo.bbl -wl"

Description:
The currently visible command line content.

Saved:
Only with Appearance set in the File->Configuration Save Configuration dialog.

StatusBarEnabled

Arguments:
1/0

Special:
This entry is only considered at startup.

Description:
1: the statusbar is visible
0: the statusbar is hidden

Saved:
Only with Appearance set in the File->Configuration Save Configuration dialog.

ToolBarEnabled

Arguments:
1/0

Special:
Using Burner

Environment

This entry is only considered at startup.

Description:
1: the toolbar is visible
0: the toolbar is hidden

Saved:
Only with Appearance set in the File->Configuration Save Configuration dialog.

WindowPos

Arguments:
10 integers, e.g. “0,1,-1,-1,-1,-1,390,107,1103,643”

Special:
This entry is only considered at startup.
Entry changes do not show “*” in the title.

Description:
These numbers contain the position and state of the window (maximized, etc.) and other flags.

Saved:
Only with Appearance set in the File->Configuration Save Configuration dialog.

WindowFont

Arguments:
size: == 0 -> generic size, < 0 -> font character height, > 0 font cell height
weight: 400 = normal, 700 = bold (valid values are 0..1000)
italic: 0 == no, 1 == yes
font name: max 32 characters.

Description:
Font attributes.

**Saved:**
Only with Appearance set in the **File->Configuration Save Configuration** dialog.

**Example:**
WindowFont=-16,500,0,Courier

**TipFilePos**

**Arguments:**
any integer, e.g. 236

**Description:**
Actual position of tip of the day file.

**Saved:**
Always when saving a configuration file.

**ShowTipOfDay**

**Arguments:**
0/1

**Description:**
Display Tip of the Day dialog at startup.
1: show
0: hide (can be displayed from the help menu)

**Saved:**
Always when saving a configuration file.

**Options**

**Arguments:**
-W2

**Description:**
The currently active option string. This entry can be long, since messages are also contained here.

**Saved:**
Only with Options set in the **File->Configuration Save Configuration** dialog.

**EditorType**

**Arguments:**
0/1/2/3

**Description:**
This entry specifies which editor configuration is active.
0: global editor configuration (in `mcutools.ini` file)
1: local editor configuration
2: command line editor configuration, entry EditorCommandLine
3: DDE editor configuration, entries beginning with EditorDDE

**Saved:**
Only with Editor Configuration set in the **File->Configuration Save Configuration** dialog.

**EditorCommandLine**

**Arguments:**
command line, for WinEdit: “C:\Winapps\WinEdit.exe %f /#:%l”

**Description:**
Command line to open a file.

**Saved:**
Only with Editor Configuration set in the File->Configuration Save Configuration dialog.

**EditorDDEClientName**

**Arguments:**
client commend, e.g. "[open(\%f)]"

**Description:**
Name of client for DDE editor configuration.

**Saved:**
Only with Editor Configuration set in the File->Configuration Save Configuration dialog.

**EditorDDETopicName**

**Arguments:**
topic name, e.g. "system"

**Description:**
Name of topic for DDE editor configuration.

**Saved:**
Only with Editor Configuration set in the File->Configuration Save Configuration dialog.

**EditorDDEServiceName**

**Arguments:**
service name, e.g. "system"

**Description:**
Name of service for DDE editor configuration.

**Saved:**
Only with Editor Configuration set in the **File->Configuration Save Configuration** dialog.

**Burner Dialog entries in [BURNER]**

**BurnerUndefByte**

**Arguments:**
integral value of undefined bytes. Default is 0xff.

**Description:**
Value of the Undef Byte entry on the Content page in the Burner dialog.

**Saved:**
Only with Appearance set in the **File->Configuration Save Configuration** dialog.

**BurnerSwapByte**

**Arguments:**
0: do not swap
1: swap

**Description:**
Value of the Swap Bytes check box on the Content page in the **Burner** dialog.

**Saved:**
Only with Appearance set in the **File->Configuration Save Configuration** dialog.

**BurnerOrigin**

**Arguments:**
integral value (0,1,2...)

**Description:**
Value of the Origin field on the Content page in the **Burner** dialog.
Saved:
Only with Appearance set in the File->Configuration Save Configuration dialog.

BurnerDestination

Arguments:
integral value (0,1,2...)

Description:
Value of the Destination Offset field on the Content page in the Burner dialog.

Saved:
Only with Appearance set in the File->Configuration Save Configuration dialog.

BurnerLength

Arguments:
integral value (0,1,2...)

Description:
Value of the Length field on the Content page in the Burner dialog.

Saved:
Only with Appearance set in the File->Configuration Save Configuration dialog.

BurnerFormat

Arguments:
0: Motorola S
1: Intel Hex
2: Binary

Description:
Format type specified on the Content page in the Burner dialog.
Saved:
Only with Appearance set in the File->Configuration Save Configuration dialog.

BurnerDataBus

Arguments:
0: “1 Byte”
1: “2 Bytes”
2: “4 Bytes”
Not the size in bytes.

Description:
Setting in the Data Bus field on the Content page in the Burner dialog.

Saved:
Only with Appearance set in the File->Configuration Save Configuration dialog.

BurnerOutputType

Arguments:

Description:
Setting in the Output field on the Input/Output page in the Burner dialog.

Saved:
Only with Appearance set in the File->Configuration Save Configuration dialog.

BurnerDataBits

Arguments:
0: “7 Bits”
1: “8 Bits”
**Description:**
Setting in the Data Bits field on the Input/Output page in the Burner dialog.

**Saved:**
Only with Appearance set in the File->Configuration Save Configuration dialog.

**BurnerParity**

**Arguments:**
0: “None”
1: “Odd”
2: “Even”

**Description:**
Setting in the Parity field on the Input/Output page in the Burner dialog.

**Saved:**
Only with Appearance set in the File->Configuration Save Configuration dialog.

**BurnerByteCommands**

**Arguments:**
0: “1st Byte (msb)”
1: “2nd Byte”
2: “3rd Byte”
3: “4th Byte”
4: “1st Word”
5: “2nd Word”

**Description:**
Setting in the command box on the Input/Output page in the Burner dialog.

**Saved:**
Using Burner
Environment

Only with Appearance set in the File->Configuration Save Configuration dialog.

**BurnerBaudRate**

**Arguments:**
300, 600, 1200, 2400, 4800, 9600, 19200, 38400

**Description:**
Setting in the Baud Rate box on the Input/Output page in the Burner dialog.

**Saved:**
Only with Appearance set in the File->Configuration Save Configuration dialog.

**BurnerOutputFile**

**Arguments:**
File Name. E.g. “file.s19”.

**Description:**
Content of the Name box on the Input/Output page in the Burner dialog.

**Saved:**
Only with Appearance set in the File->Configuration Save Configuration dialog.

**BurnerHeaderFile**

**Arguments:**
File Name. E.g. “headerfile”.

**Description:**

**Saved:**
Only with Appearance set in the File->Configuration Save Configuration dialog.
BurnerInputFile

Arguments:
File Name. E.g. “file.abs”.

Description:

Saved:
Only with Appearance set in the File->Configuration Save Configuration dialog.

Example
The following example shows a typical layout of the configuration file (usually project.ini):

[Editor]
Editor_Name=WinEdit
Editor_Exe=C:\WinEdit\WinEdit.exe %f /#:%l
Editor_Opts=%f

[Burner]
StatusBarEnabled=1
ToolbarEnabled=1
WindowPos=0,1,-1,-1,-1,-1,390,107,1103,643
WindowFont=-16,500,0,Courier
TipFilePos=0
ShowTipOfDay=1
Options=-w1
EditTextype=3
RecentCommandLine=-ffibo.bbl -w1
CurrentCommandLine=-ffibo.bbl -w2
EditorDDEClientName=[open(%f)]
EditorDDETopicName=system
EditorDDEServiceName=msdev
EditorCommandLine=C:\WinEdit\WinEdit.exe %f /#:%l
BurnerUndefByte=255
BurnerSwapByte=0
BurnerOrigin=0
BurnerDestination=0
BurnerLength=65536
BurnerFormat=0

Burner
Using Burner

**Environment**

BurnerDataBus=0
BurnerOutputType=4
BurnerDataBits=1
BurnerParity=0
BurnerByteCommands=0
BurnerBaudRate=9600
BurnerOutputFile=outputfile.s19
BurnerHeaderFile=headerfile
BurnerInputFile=InputFile.abs

---

**Paths**

Most environment variables contain path lists indicating where to look for files. A path list is a list of directory names separated by semicolons.

\[
\text{PathList} = \text{DirSpec} \ (\text{"}" \ \text{DirSpec}).
\]

DirSpec = ["**"] DirectoryName.

**Example:**

\[
\text{GENPATH} = \text{C:"INSTALL\LIB};\text{D:"PROJECTS\TESTS};/\text{usr/local/}
\text{Metrowerks/lib};/\text{home/me/my_project}
\]

- If a directory name is preceded by an asterisk ("*") the programs recursively search the directory tree for a file, not just the given directory. Directories are searched in the order they appear in the path list.

**Example:**

\[
\text{LIBPATH} = \text{*C:"INSTALL\LIB}
\]

---

**NOTE**

Some DOS/UNIX environment variables (like GENPATH, LIBPATH, etc.) are used.

We recommend working with WinEdit and setting the environment by means of a `DEFAULT.ENV` (`.hidefaults` for UNIX) file in your project directory. This 'project directory' can be set in WinEdit's 'Project Configure...' menu command. This way you can have different projects in different directories, each with its own environment.

---

**NOTE**

When using WinEdit, do not set the system environment variable `DEFAULTDIR`. If you do and this variable does not contain the
project directory given in WinEdit’s project configuration, files might not be put where you expect them.

Line Continuation

It is possible to specify an environment variable in an environment file (default.env/.hidefaults) over multiple lines using the line continuation character ‘\’:

Example:

```
OPTIONS=\
-W2  \\
-Wpd
```

This is the same as

```
OPTIONS=-W2  -Wpd
```

Be careful using this feature with paths, e.g.

```
GENPATH=.\\
TEXTFILE=.\txt
```

will result in

```
GENPATH=.TEXTFILE=.\txt
```

To avoid such problems, we recommend using a semicolon ‘;’ at the end of a path, if there is a ‘\’ at the end:

```
GENPATH=.\;
TEXTFILE=.\txt
```

Environment Variable Details

The remainder of this section describes each of the environment variables available for a tool. Options are listed in alphabetical order and each is divided into several sections.
<table>
<thead>
<tr>
<th>Topic</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tools</td>
<td>Lists tools that use variable</td>
</tr>
<tr>
<td>Synonym</td>
<td>For some environment variables a synonym also exists. The synonyms may be</td>
</tr>
<tr>
<td></td>
<td>used for older releases of the tool and will be removed in the future. A</td>
</tr>
<tr>
<td></td>
<td>synonym has lower precedence than the environment variable.</td>
</tr>
<tr>
<td>Syntax</td>
<td>Specifies syntax of option in EBNF format.</td>
</tr>
<tr>
<td>Arguments</td>
<td>Describes and lists optional and required arguments for the variable.</td>
</tr>
<tr>
<td>Default</td>
<td>Shows default setting for the variable or none.</td>
</tr>
<tr>
<td>Description</td>
<td>Provides a detailed description of the option and how to use it.</td>
</tr>
<tr>
<td>Example</td>
<td>Gives an example of usage, and effects of the variable where possible.</td>
</tr>
<tr>
<td></td>
<td>Examples show an entry in <code>default.env</code> for PC or in the</td>
</tr>
<tr>
<td></td>
<td><code>.hidefaults</code> file for UNIX.</td>
</tr>
<tr>
<td>See also</td>
<td>Related sections.</td>
</tr>
</tbody>
</table>

**DEFAULTDIR**

**DEFAULTDIR: Default Current Directory**

**Tools:**
Compiler, Assembler, Linker, Decoder, Debugger, Librarian, Maker, Burner

**Synonym:**
none.

**Syntax:**
```
"DEFAULTDIR=" <directory>.
```

**Arguments:**
`<directory>`: The default current directory.

**Default:**
none.
**Description:**
This environment variable specifies the default directory for all tools. All tools indicated above will use the directory specified as their current directory instead of the one defined by the operating system or launching tool (e.g. editor).

**NOTE**
This is a system level (global) environment variable. It CANNOT be specified in a default environment file (DEFAULT.ENV/.hidedefaults).

**Example:**
```
DEFAULTDIR=C:\INSTALL\PROJECT
```

**See also:**
- Section ‘The Current Directory’
- Section MCUTOOLS.INI File

---

**ENVIRONMENT**

**ENVIRONMENT: Environment File Specification**

**Tools:**
Compiler, Linker, Decoder, Debugger, Librarian, Maker, Burner

**Synonym:**
HIENVIRONMENT

**Syntax:**
```
"ENVIRONMENT=" <file>.
```

**Arguments:**
- `<file>`: file name with path specification, without spaces

**Default:**
none.
**Using Burner**

**Environment**

**Description:**

This variable is specified at the system level. Normally the application looks in the current directory for an environment file named default.env (.hidefaults on UNIX). Using ENVIRONMENT (e.g. set in the autoexec.bat (DOS) or .cshrc (UNIX)), a different file name may be specified.

**NOTE**

This is a system level (global) environment variable. It CANNOT be specified in a default environment file (DEFAULT.ENV/.hidefaults).

**Example:**

```
ENVIRONMENT=\Metrowerks\prog\global.env
```

**See also:**

none.

**ERRORFILE**

**ERRORFILE: Error File Name Specification**

**Tools:**

Compiler, Assembler, Linker, Burner

**Synonym:**

none.

**Syntax:**

```
"ERRORFILE=\ <file name>\.
```

**Arguments:**

- `<file name>`: File name with possible format specifiers.

**Description:**

This environment variable specifies the name of the error file.
Possible format specifiers are:

'\%n': Substitutes with file name without path.

'\%p': Path of the source file.

'\%f': Full file name, including path (same as '\%p%\n')

In case of an illegal error file name, a notification box is shown.

**Example:**

```plaintext
ERRORFILE=MyErrors.err
```

lists all errors into the file MyErrors.err in current directory.

```plaintext
ERRORFILE=\tmp\errors
```

lists all errors into the file errors in the directory \tmp.

```plaintext
ERRORFILE=\%f.err
```

lists all errors into a file with the same name as the source file, but with extension .err, and placed in the same directory as the source file. For example, if we process a file \sources\test.c, an error list file \sources\test.err will be generated.

```plaintext
ERRORFILE=\dir1\%n.err
```

For a source file test.c, an error list file \dir1\test.err will be generated.

```plaintext
ERRORFILE=\%p\errors.txt
```

For source file \dir1\dir2\test.c, an error file \dir1\dir2\errors.txt will be generated.

If the environment variable ERRORFILE is not set, errors are written to the file EDOUT in the current directory.

**Example:**

Another example shows usage of this variable to support correct error feedback with the WinEdit Editor, which looks for an error file called EDOUT:

---

Installation directory: E:\INSTALL\PROG
Project sources: D:\MEPHISTO
Common Sources for projects: E:\CLIB

Entry in default.env (D:\MEPHISTO\DEFAULT.ENV):
ERRORFILE=E:\INSTALL\PROG\EDOUT

Entry in WINEDIT.INI (in Windows directory):
See also:
none.

GENPATH

GENPATH: #include “File” Path

Tools:
Compiler, Linker, Decoder, Debugger, Burner

Synonym:
HIPATH

Syntax:
"GENPATH=" {<path>}

Arguments:
<path>: Paths separated by semicolons, without spaces.

Default:
Current directory

Description:
This path specification is used by the burner to search for input files.

NOTE
If a directory specification in this environment variable starts with an asterisk (“*”), the complete directory tree is searched recursively. All subdirectories are searched. Within one level in the directory tree, search order of the subdirectories is random undeterminable.

Example:
Using Burner
Environment

GENPATH=\sources\include;..\..\headers;\usr\local\lib

See also:
none.

TMP

TMP: Temporary directory

Tools:
Compiler, Assembler, Linker, Debugger, Librarian, Burner

Synonym:
none.

Syntax:
"TMP= " <directory>.

Arguments:
<directory>: Directory used for temporary files.

Default:
none.

Description:
If a temporary file has to be created, normally the ANSI function tmpnam() is used. This library function stores the temporary files created in the directory specified by this environment variable. If the variable is empty or does not exist, the current directory is used. Check this variable if you get an error message "Cannot create temporary file".

NOTE This is a system level (global) environment variable. It CANNOT be specified in a default environment file (DEFAULT.ENV/.hidefaults).
Using Burner

Messages

Example:

TMP=C:\TEMP

See also:

• Section ‘The Current Directory’

Messages

This section describes messages produced by the Application. Because of the number
of messages produced, some may not have been documented at the time of this
release.

Message Kinds

There are five kinds of messages generated:

INFORMATION

A message will be printed and compilation will continue. Information messages are
used to indicate actions taken by the application.

WARNING

A message will be printed and processing will continue. Warning messages are used to
indicate possible possible programming errors.

ERROR

A message will be printed and processing is stopped. Error messages are used to
indicate illegal use of the language.

FATAL

A message will be printed and processing is aborted. A fatal message indicates a
severe error that will stop processing.
**DISABLE**

No message will be issued and processing will continue. The application ignores this type of message.

**Message Details**

If the application prints a message, the message contains a message code and a four to five digit number. This number may be used to search for the indicated message. Following message codes are supported:

- “A” for Assemblers
- “B” for Burner
- “C” for Compilers
- “L” for Linker
- “LM” for Libmaker
- “M” for Maker

All messages generated by the application are documented in increasing number order for quick retrieval.

Each message also has a description and if available a short example with a possible solution or tips to fix a problem.

For each message, the type of message is also noted, e.g. [ERROR] indicates that the message is an error message.

[DISABLE, INFORMATION, WARNING, ERROR] indicates that the message is a warning message by default, but the user might change the message to either DISABLE, INFORMATION or ERROR.

After the message type, there may be an additional entry indicating the related language:

- C++: Message is generated for C++
- M2: Message is generated for Modula-2

**Message List**

The following pages describe all messages.
B1: Unknown message occurred

[FATAL]

Description:
The application tried to emit a message which was not defined. This is an internal error that should not occur. Please report any occurrences to your distributor.

B2: Message overflow, skipping <kind> messages

[DISABLE, INFORMATION, WARNING, ERROR]

Description:
Application displayed the number of messages as controlled by the options \texttt{-WmsgNi}, \texttt{-WmsgNw} and \texttt{-WmsgNe}. Further options of this kind are not displayed.

TIP
Use the options \texttt{-WmsgNi}, \texttt{-WmsgNw} and \texttt{-WmsgNe} to change the number of messages.

B50: Input file ‘<file>’ not found

[FATAL]

Description
The Application was not able to find a file needed for processing.

TIP
Check if the file really exits. Check if you are using a file name containing spaces (in this case you have to place quotes around filename).

B51: Cannot open statistic log file <file>

[DISABLE, INFORMATION, WARNING, ERROR]

Description
It was not possible to open a statistic output file, therefore no statistics are generated.
NOTE

Not all tools support statistic log files. Even if a tool does not support it, the message still exists, but never issued.

---

**B52: Error in command line '<cmd>'**

[FATAL]

**Description**

In case there is an error while processing the command line, this message is issued.

---

**B64: Line Continuation occurred in <FileName>**

[DISABLE, INFORMATION, WARNING, ERROR]

**Description:**

In an environment file, the character '\' at the end of a line is interpreted as line continuation. This line and the next one are interpreted as one line. Because the path separation character of MS-DOS is also '\', paths are often incorrectly written that end with '\'. Instead use a '.' after the last '\' in a path.

**Example:**

Current Default.env:

```plaintext
... LIBPATH=c:\Metrowerks\lib
OBJPATH=c:\Metrowerks\work
... Is interpreted as
... LIBPATH=c:\Metrowerks\lib
OBJPATH=c:\Metrowerks\work
...```

**TIP**

To fix it, append a '.' at the end of '\'

```plaintext
... LIBPATH=c:\Metrowerks\lib.
OBJPATH=c:\Metrowerks\work
...```
NOTE Because this information occurs during the initialization phase of the application, the message prefix might not occur in the error message. So it might occur as "64: Line Continuation occurred in <FileName>".

B65: Environment macro expansion error '
<description>' for <variablename>

[DISABLE, INFORMATION, WARNING, ERROR]

Description:
During an environment variable macro substitution a problem occurred. Possible causes could be that the named macro did not exist or some length limitation was reached. Recursive macros may also cause this message.

Example:
Current variables:

... LIBPATH=${LIBPATH}
...  

TIP Check the definition of the environment variable.

B66: Search path <Name> does not exist

[DISABLE, INFORMATION, WARNING, ERROR]

Description
The tool searched for a file that was not found. During the failed search, a non existing path was encountered.

TIP
- Check the spelling of your paths.
- Update the paths when moving a project.
- Use relative paths in your environment variables.
- Check if network drives are available.
**B1000: Could not open '<FileType>' '<File>'**

[ERROR]

**Description:**
The specified file could not be open.

This message is used for input and output file.

**TIP** For files to be generated, check if they are modifiable and sufficient space exists on the disk. Ensure that the file is not locked by another application and that the path exists.

**B1001: Error in input file format**

[ERROR]

**Description:**
An error occurred while reading the input file.

**TIP**
- Try to generate the input file again.
- Check if you have enough free disk space.

**B1002: Selected communication port is busy**

[ERROR]

**Description:**
The selected communication port can not be accessed.

**TIP**
1. Check if another application has locked the serial port.
2. Check if the correct serial port is specified.
B1003: Timeout or failure for the selected communication

[ERROR]

Description:
There was a timeout or general failure on the selected communication port.

TIP Check if another application has locked the serial port.

B1004: Error in macro ‘<macro>’ at position <pos>: ‘<msg>’

[ERROR]

Description:
While resolving a macro, the Burner was not able to resolve it. A macro is surrounded by ‘%’ characters (e.g. %ABS_FILE%).

TIP Check if you have a definition of your macro in the environment. Check if the macro is passed on the command line using the -Env option.

B1005: Error in command line at position <pos>: ‘<msg>’

[ERROR]

Description:
If the command line scanner has detected an illegal command line, this message is produced.

TIP Check the syntax of your command line.
B1006: ‘<msg>’

[ERROR]

**Description:**

This message is used to indicate generic errors.
Index

Symbols
.ABS 7
.bbl 52
.hidefaults 88, 89, 111, 112, 115

B
baudRate 24
BURNER 9, 102
Burner Dialog 102
BurnerBaudRate 106
BurnerByteCommands 105
BurnerDataBits 104
BurnerDataBus 104
BurnerDestination 103
BurnerFormat 103
BurnerHeaderFile 106
BurnerInputFile 107
BurnerLength 103
BurnerOrigin 102
BurnerOutputFile 106
BurnerOutputType 104
BurnerParity 105
BurnerSwapByte 102
BurnerUndefByte 102
busWidth 25

C
CLOSE 26
color 65, 66, 67, 68, 69
Compiler
   Error messages 116
Current Directory 89, 110
CurrentCommandLine 97

D
-D 49
dataBit 26
Default Directory 91
DEFAULT.ENV 88, 89, 111, 112, 115
DEFAULTDIR 89, 91, 110
DefaultDir 91
destination 27

DO 28

E
ECHO 29
Editor 95
Editor_Exe 93, 95
Editor_Name 93, 95
Editor_Opts 93, 96
EditorCommandLine 100
EditorDDEClientName 101
EditorDDESeroniceName 101
EditorDDETopicName 101
EditorType 100
ELSE 29
END 30
-Env 50, 88
ENVIRONMENT 111
Environment
   DEFAULTDIR 89, 91, 110
   ENVIRONMENT 88, 111
   ERRORFILE 112
   File 88
   GENPATH 114
   HIENVIRONMENT 111
   HIPATH 114
   TMP 115
   Variable 88, 109
err.log 64
Error Format
   Microsoft 71, 72
   Verbose 71
Error messages 116
ERRORFILE 112
Explorer 89

F
-F 51
File
   Environment 88
File Manager 89
FOR 31
format 32
G
GENPATH 114
Group 90

H
-H 52
header 33
HIENVIRONMENT 111
HIPATH 114
HOST 48

I
IF 34
INPUT 48

L
len 35
-Lic 53
-LicA 54
Line Continuation 109

M
MCUTOOLS.INI 89, 111
MESSAGE 48
MESSAGES 48
Microsoft 71, 72

N
-N 55
-NoBeep 56
-NoEnv 57
-Ns 58

O
OPENCOM 36
OPENFILE 36
Option
HOST 48
INPUT 48
MESSAGE 48
MESSAGES 48
VARIOUS 48
Options 91, 99
origin 37
P
parity 38
Path 90
path in S0 record 58
Path List 108
PAUSE 46
-Prod 59, 95
project.ini 9, 95

R
RecentCommandLine 96
RGB 65, 66, 67, 68, 69

S
S0 58
S1 58
S2 58
S3 58
S7 58
S8 58
S9 58
SaveAppearance 91
SaveEditor 92
SaveOnExit 91
SaveOptions 92
SENDBYTE 39
SENDWORD 40
ShowTipOfDay 99
SLINELEN 41
SRECORD 42
startup 95
StatusbarEnabled 97
stdout 86
swapByte 43

T
THEN 44
TipFilePos 99
TMP 115
TO 45
ToolbarEnabled 97

U
undefByte 45
UNIX 89

V
-V 60,62
Variable
  Environment 88
VARIOUS 48
-View 61

W
-W1 86
-W2 87
-WErrFile 64
WindowFont 98
WindowPos 98
Windows 89
WinEdit 89,113
-Wmsg8x3 63
-WmsgCE 65
-WmsgCF 66
-WmsgCI 67
-WmsgCU 68
-WmsgCW 69
-WmsgPb 64,70,72,74,75,76,77
-WmsgFbi 70
-WmsgFbm 70
-WmsgFi 64,71,74,75,76,77
-WmsgFim 71
-WmsgFiv 71
-WmsgFob 72,75
-WmsgFoi 74,76,77
-WmsgFonf 75
-WmsgFonp 74,75,76,77
-WmsgNe 78
-WmsgNi 78
-WmsgNu 79
-WmsgNw 80
-WmsgSd 81
-WmsgSe 82
-WmsgSi 83
-WmsgSw 83
-WOutFile 84
-WStdout 85