Decoder

Revised 07/30/2003

metrowerks
Software Starts Here

How to Contact Metrowerks

Corporate Headquarters	Metrowerks Corporation 7700 West Parmer Lane Austin, TX 78729 U.S.A.
World Wide Web	`http://www.metrowerks.com`
Sales	Voice: 800-377-5416 Fax: 512-996-4910 Email: sales@metrowerks.com
Technical Support	Voice: 800-377-5416 Email: support@metrowerks.com

Table of Contents

1

Introduction

This document describes the ELF/HIWARE Decoder. It disassembles object files, absolute files and libraries in the HIWARE object file format or ELF/DWARF format and Motorola S-Record files. Various output formats are available.

The decoder has:

- Graphical User Interface (GUI)
- On-line Help
- Message Management
- 32-bit Functionality
- Decodes HIWARE object file format
- Decodes ELF/Dwarf 1.1 and 2.0 object file format
- Decoder Motorola S-Record files

Click any of the following links to jump to the corresponding section of this chapter:

- User Interface
- Read the Release Notes
- Technical Support
- Contacting the Documentation Team

User Interface

The decoder provides a command line interface and an interactive interface. If no arguments are given on the command line, a window is opened that prompts for arguments.

The Decoder accepts object or absolute files, libraries, and Motorola S-Record files as input to generate the listing file. The name of the source files are encoded in the object or absolute file or library. For Motorola S-Record files, the processor must be specified with the option -Env: Set Environment Variable.

The generated listing file has the same name as the input file but with extension `.LST`. It contains source and assembly statements. The corresponding C/C++ source statements can be displayed within the generated assembly instructions.

Read the Release Notes

Before you use a tool such as the Decoder, read the release notes. They contain important last-minute information about new features and technical issues or incompatibilities that may not be included in the documentation.

Technical Support

If you are having problems installing or using any Metrowerks product, contact Technical Support as shown in Table 1.1.

Table 1.1 Ways to Contact Technical Support

E-mail	`support@metrowerks.com` (USA) `support_europe@metrowerks.com` (Europe)
World Wide Web	`http://www.metrowerks.com/`
Compuserve	`Go: Metrowerks`
`CW_Release_Notes\email_Report_Forms\email_Tech_Question_Form.txt`	

Contacting the Documentation Team

Please report errors or omissions to the Metrowerks Documentation Department: `wordwarrior@metrowerks.com`.

2

Decoder Controls

This chapter describes Decoder controls; pull-down menus and the Graphical User Interface (GUI).

Click any of the following links to jump to the corresponding section of this chapter:

- Pull-down Menus
- Graphical User Interface
- Specifying the Input File
- Message and Error Feedback

Pull-down Menus

The Decoder pull-down menus are on the menu bar of the Decoder main window (Figure 2.5).

Table 2.1 lists and describes the Decoder's top-level pull-down menus.

Table 2.1 Decoder Pull-down Menus

Menu Name	Contains
File	Options for managing configuration files
Decoder	Commands for setting options
View	Options for customizing window output
Help	Standard Windows Help menu

File Menu

With the File Menu (Figure 2.1), you can save or load configuration files. Configuration files contain:

- Configuration dialog option settings (Figure 2.8).
- Message settings that specify which messages to display and which to treat as errors.

- List of last commands executed and current command line
- Window position
- Tip of the Day settings, including if enabled at startup and current entry

Figure 2.1 File Menu

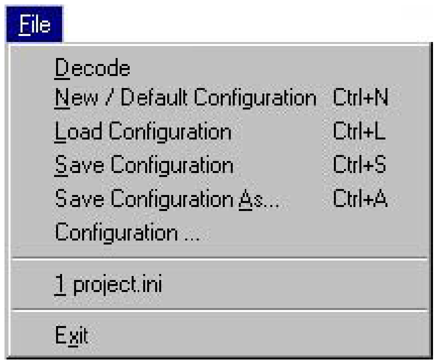

File
Decode
New / Default Configuration Ctrl+N
Load Configuration Ctrl+L
Save Configuration Ctrl+S
Save Configuration As... Ctrl+A
Configuration ...
1 project.ini
Exit

Table 2.2 lists and describes **File** menu selections:

Table 2.2 File Menu Selections

Menu Selection	Description
Decode	Opens a standard **Open File** dialog. The selected file will be processed as soon as the open File box is closed using *OK*.
New/Default Configuration	Resets the option settings to the default value. The options which are activated per default are specified in section ***Command Line Options*** from this document.
Load Configuration	Opens the standard open file dialog. Configuration data stored in the selected file is loaded and will be used by a further session.
Save Configuration	Saves the current settings.
Save Configuration as...	Opens a standard **Save As** dialog. The current settings are saved in a configuration file which has the specified name.
Configuration...	Opens the ***Configuration*** dialog to specify the editor used for error feedback and which parts to save with a configuration.
1. .... project.ini 2.	Recent project list. This list can be accessed to open a recently opened project again.
Exit	Closes the Decoder.

Decoder Menu

With the **Decoder** Menu (Figure 2.2), you can customize the Decoder, graphically set or reset options, and access message settings.

Figure 2.2 Decoder Menu

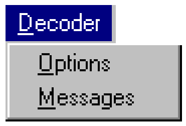


Table 2.3 lists and describes **Decoder** menu selections.

Table 2.3 Decoder Menu Selections

Menu entry	Description
Options	Displays the Option Settings dialog (Figure 2.12), where you can define options for processing an input file.
Messages	Opens the Message Settings dialog (Figure 2.13), where the different error, warning or information messages can be mapped to another message class.

View Menu

With the **View** Menu (Figure 2.3), you can customize the main window (Figure 2.5). You can specify whether the status bar and the tool bar are displayed or hidden, choose the font used in the window, and clear the window.

Figure 2.3 View Menu

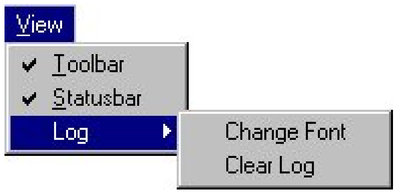


Table 2.4 lists and describes **View** menu selections.

Table 2.4 View Menu Selections

Menu entry	Description
Tool Bar	Displays tool bar in the main window.
Status Bar	Displays status bar in the main window.
Log...	Lets you customize the output in the main window content area.
Change Font	Opens a standard font-selection dialog. Your selections appear in the main window content area.
Clear Log	Lets you clear the main window content area.

Help Menu

From the **Help** Menu (Figure 2.4), you can customize the **Tip of the Day** dialog and display help, as well as Decoder version and license information.

Figure 2.4 Help Menu

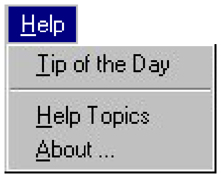


Table 2.5 lists and describes **Help** menu selections.

Table 2.5 Help Menu Selections

Menu entry	Description
Tip of the Day	Switches on or off the Tip of the Day display during startup.
Help Topics	Displays standard Help.
About ...	Displays an About box with version and license information.

Graphical User Interface

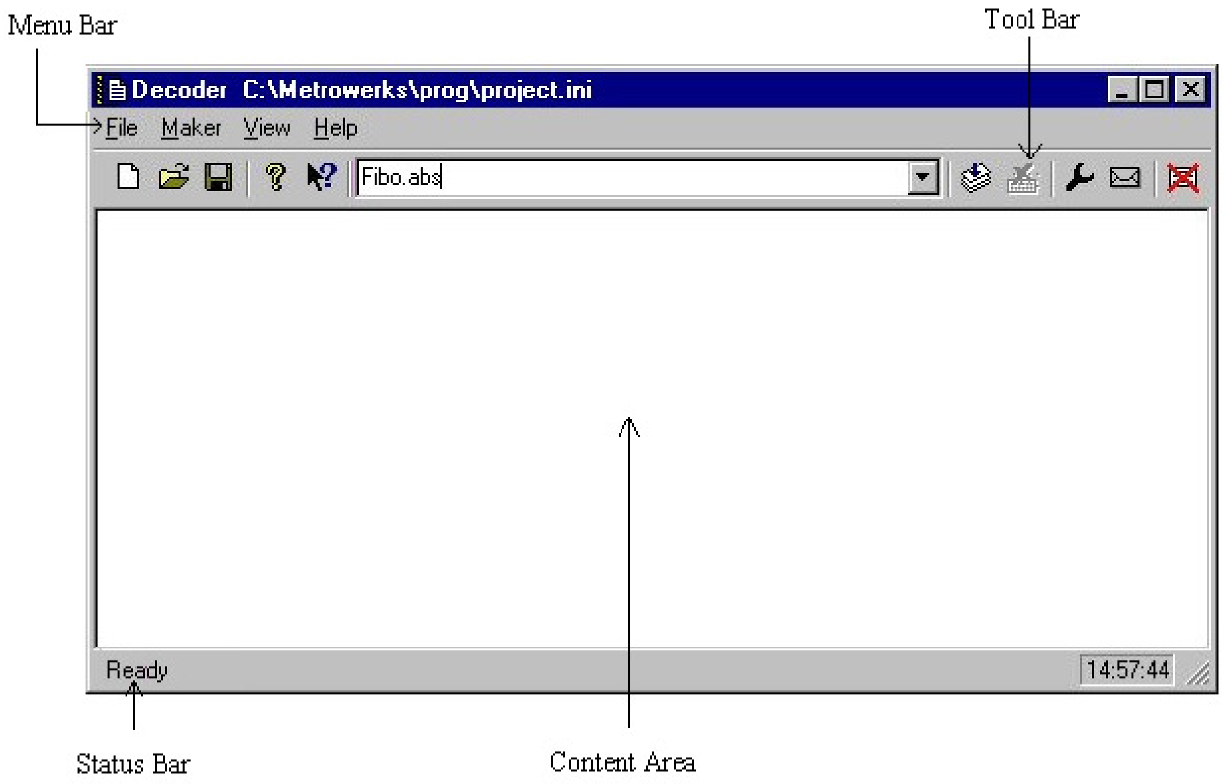
This section describes important aspects of the Decoder graphical user interface (GUI). Windows and dialogs covered here are:

- Decoder Main Window
- Configuration Dialog

Decoder Main Window

The Decoder main window (Figure 2.5) displays if you do not specify a file name on the command line. If you start a tool using the Decoder, the Decoder main window does not appear.

Figure 2.5 Decoder Main Window



Main Window Components

The Decoder main window has these components:

- Window title
- Tool bar

- Content area
- Status bar
- Menu bar

Window Title

The window title displays the tool name and project name. If no project is loaded, **Default Configuration** displays in the title area. An asterisk after the configuration name indicates you have made an unsaved change.

Tool Bar

Figure 2.6 indicates main window tool bar buttons.

Figure 2.6 Decoder Main Window Tool Bar Buttons

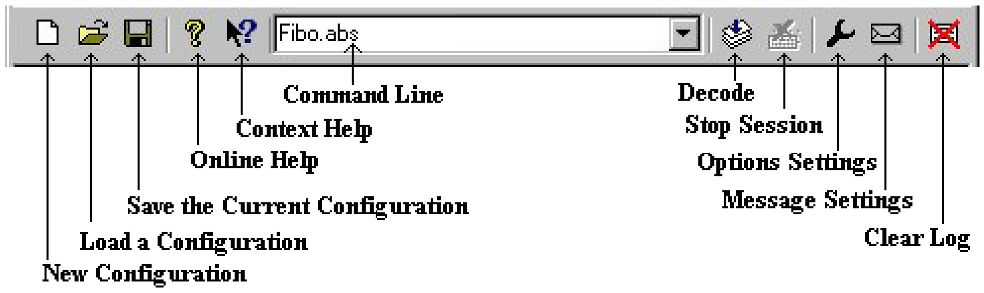


Table 2.6 lists the Decoder main window tool bar buttons and describes their functions:

Table 2.6 Main Window Tool Bar Buttons

Button Name	Function
New Configuration	Same as the **File > New Configuration** pull-down menu selection
Load a Configuration	Same as the **File > Load Configuration** pull-down menu selection
Save the Current Configuration	Same as the **File > Save Configuration** pull-down menu selection
Online Help	Displays Decoder online help
Context Help	Changes cursor to question mark. When you hover your cursor over a Decoder screen area and click the left mouse button, context-sensitive help appears for the area you selected.

Table 2.6 Main Window Tool Bar Buttons (*continued*)

Button Name	Function
Command Line	Displays a context menu associated with the command line.
Decode	Starts execution of a desired command.
Stop Session	Stops the current session
Option settings	Displays the **Option Settings** dialog
Message settings	Displays the **Message Settings** dialog
Clear log	Clears main window content

Status Bar

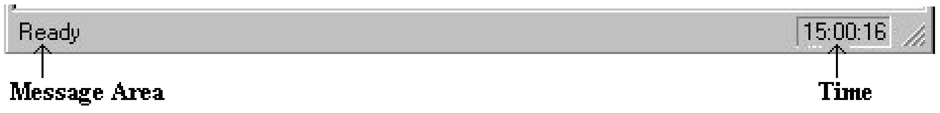
The status bar (Figure 2.7) has two dynamic areas:

- Messages
- Time

When you point to a button in the tool bar or a menu entry, the message area displays the function of the button or menu entry.

The time field shows the start time of the current session (if one is active) or current system time.

Figure 2.7 Main Window Status Bar



Configuration Dialog

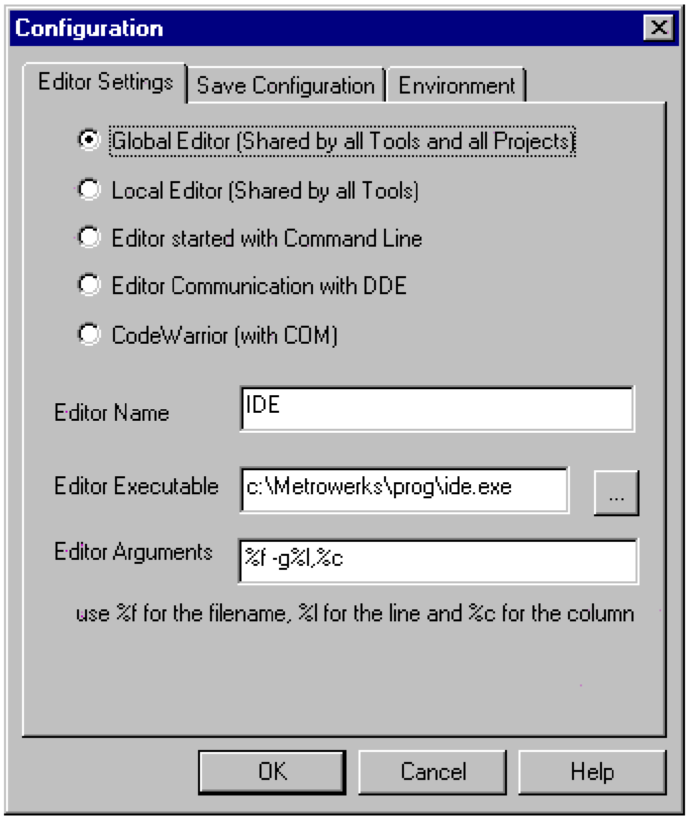
When you choose **File > Configuration** from the Decoder pull-down menus, the **Configuration** dialog appears. The **Configuration** dialog has these three tabs:

- Editor Settings
- Save Configuration
- Environment

Configuration Dialog Editor Settings Tab

Figure 2.8 shows the **Configuration** dialog with the **Editor Settings** tab selected.

Figure 2.8 Configuration Dialog - Editor Settings Tab



[● Global Editor (Shared by all Tools and all Projects)]
○ Local Editor (Shared by all Tools)
○ Editor started with Command Line
○ Editor Communication with DDE
○ CodeWarrior (with COM)

Editor Name: IDE

Editor Executable: c:\Metrowerks\prog\ide.exe

Editor Arguments: %f -g%l,%c

use %f for the filename, %l for the line and %c for the column

Editor Settings Tab

Table 2.7 lists and describes **Editor Settings** tab controls.

Table 2.7 Editor Settings Tab Controls

Control	Function
Global Editor	Shared among all tools and projects on one computer. It is stored in the `MCUTOOLS.INI` global initialization file.
Local Editor	Shared among all tools using the same project file

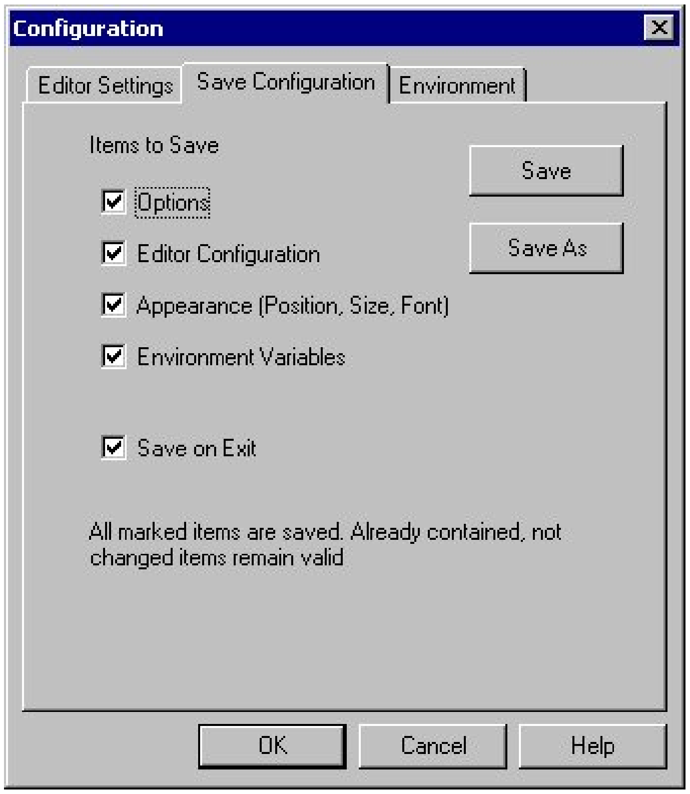
Table 2.7 Editor Settings Tab Controls

Control	Function
Editor started with Command Line	Enable command-line editor. For the Winedit 32-bit version use the `winedit.exe` file `C:\WinEdit32\WinEdit.exe%f /#:%l`
Editor started with DDE	Enter the service, topic and client name to be used for a DDE connection to the editor. All entries can have modifiers for file name and line number.
CodeWarrior (with COM)	If selected, the CodeWarrior registered in the Windows Registry launches.
Editor Name	Type a name for the desired editor in the text-entry field
Editor Executable	Specify the editor's path and executable name. Use the browse button (...) to locate the executable.
Editor Arguments	Type in the command-line arguments for the editor in the text-entry field. Use `%f` for the filename, `%l` for the line number, and `%c` for the column number.

Save Configuration Tab

Figure 2.9 shows the **Configuration** dialog with the **Save Configuration** tab selected.

Figure 2.9 Configuration Dialog - Save Configuration Tab



All marked items are saved. Already contained, not changed items remain valid

Table 2.8 lists and describes **Save Configuration** tab controls.

Table 2.8 Save Configuration Tab Controls

Control	**Function**
Options	When checked, the current option and message settings are contained when a configuration is written. When unchecked, the last saved content remains valid.
Editor Configuration	When checked, the current editor setting is contained when a configuration is written. By disabling this option, the last saved content remains valid.
Appearance	When checked, window position, command line content, and history settings are contained when a configuration is written.

Table 2.8 Save Configuration Tab Controls (*continued*)

Control	Function
Environment Variables	When checked, the environment variable settings in the Environment Tab are written to the configuration.
Save on Exit	When checked, Decoder writes configuration on exit. No confirmation message appears. When unchecked, Decoder does not write configuration on exit, even if options or another part of the configuration has changed. No confirmation message appears when closing the Decoder.

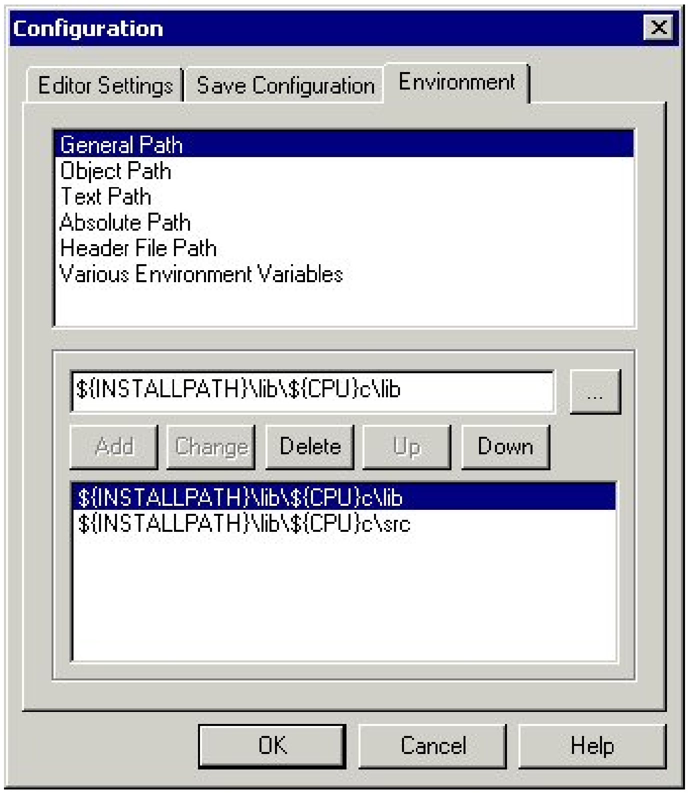
NOTE Settings are stored in the configuration file. Exceptions are recently used configuration list and settings in this dialog.

NOTE Configurations can coexist in the same file as the shell project configuration. When the shell configures an editor, the Decoder can read the content from the project file. The shell project configuration filename is `project.ini`.

Environment Tab

Figure 2.10 shows the **Configuration** dialog with the **Environment** tab selected.

Figure 2.10 Configuration Dialog - Environment Tab



General Path
Object Path
Text Path
Absolute Path
Header File Path
Various Environment Variables

```
${INSTALLPATH}\lib\${CPU}c\lib
```

```
${INSTALLPATH}\lib\${CPU}c\lib
${INSTALLPATH}\lib\${CPU}c\src
```

Use this dialog to configure the environment. The content of the dialog is read from the project file in the [Environment Variables] section. You can choose from the following environment variables:

- General Path: GENPATH
- Object Path: OBJPATH
- Text Path: TEXTPATH
- Absolute Path: ABSPATH
- Header File Path: LIBPATH
- Various Environment Variables: other variables not covered in this list

Table 2.9 lists and describes **Environment** tab controls.

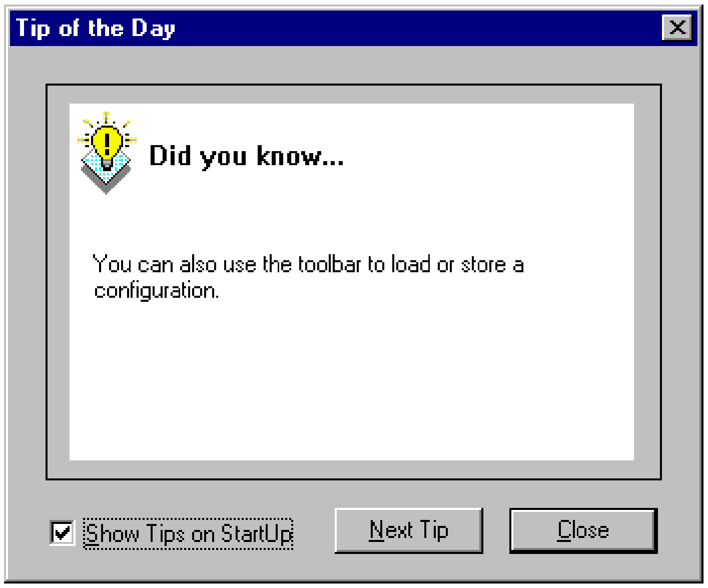
Table 2.9 Environment Tab Buttons

Button	Function
Add	Adds a new line/entry
Change	Changes a new line/entry
Delete	Deletes a new line/entry
Up	Moves up a line/entry
Down	Moves down a line/entry

Tip of the Day Window

When you start the tool, a **Tip of the Day** window (Figure 2.11) displays a randomly chosen user tip.

Figure 2.11 Tip of the Day Window



You can also use the toolbar to load or store a configuration.

The **Next Tip** button lets you read the next hint. If you don't want the Tip of the Day window to open the program starts, uncheck the **Show Tips on StartUp** box. Click **Close** to close the **Tip of the Day** window.

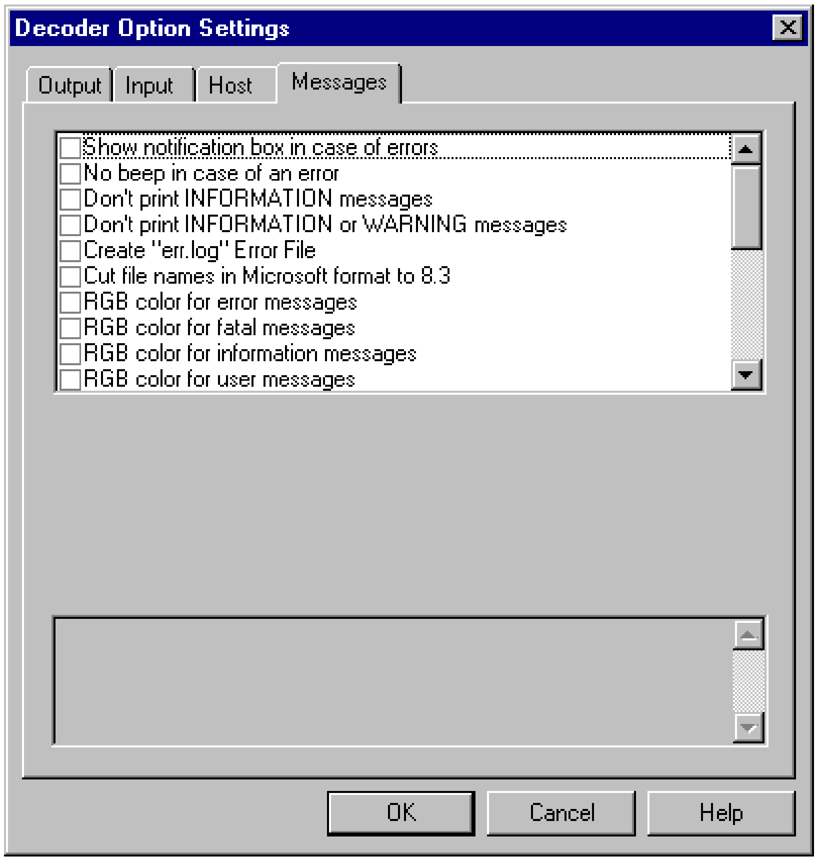
NOTE User configurations are in the local project file.

Options Settings Dialog

The Options Settings dialog (Figure 2.12) displays when you select **Decoder > Options** from the pull-down menus. Click on the text in the list box to select an option. For help, select an option and press **F1**. The command-line option in the lower part of the dialog corresponds with your selection in the list box.

NOTE When options requiring additional parameters are selected, a dialog box or subwindow may appear.

Figure 2.12 Options Settings Dialog



- [ ] Show notification box in case of errors
- [ ] No beep in case of an error
- [ ] Don't print INFORMATION messages
- [ ] Don't print INFORMATION or WARNING messages
- [ ] Create "err.log" Error File
- [ ] Cut file names in Microsoft format to 8.3
- [ ] RGB color for error messages
- [ ] RGB color for fatal messages
- [ ] RGB color for information messages
- [ ] RGB color for user messages

Table 2.10 lists and describes the tabs in the **Option Settings** dialog.

Table 2.10 Option Settings Dialog Tabs

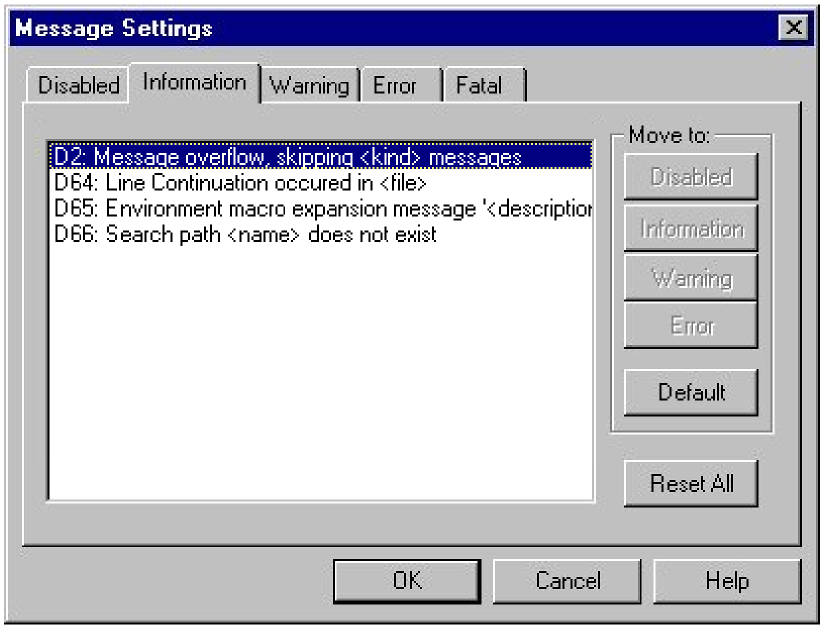
Tab	Description
Output	Command-line execution and print output settings
Input	Macro settings
Host	Lists options related to the host operating system
Messages	Message-handler settings - format, kind, and number of printed messages

Message Settings Dialog

The **Message Settings** dialog (Figure 2.13) displays when you select **Decoder > Messages** from the pull-down menus. This dialog lets you map messages to different message classes.

Each message has its own id (a character followed by a 4- or 5-digit number). This number allows you to search for the message in the manual and online help.

Figure 2.13 Message Settings Dialog



D2: Message overflow, skipping <kind> messages
D64: Line Continuation occured in <file>
D65: Environment macro expansion message '<descriptior
D66: Search path <name> does not exist

Table 2.11 lists and describes the tabs in the **Message Settings** dialog.

Table 2.11 Message Settings Dialog Tabs

Message Group	Description
Disabled	Lists disabled messages. Messages displayed in the list box are not written to the output stream.
Information	Lists information messages. Information messages inform you of actions taken.
Warning	Lists warning messages. When a warning message is generated, processing of the input file continues.
Error	Lists error messages. When an error message is generated, processing of the input file stops.
Fatal	Lists fatal error messages. These messages report system consistency errors. Fatal error messages cannot be ignored or moved.

Changing a Message Class

You can configure your own mapping of messages in different classes using one of the buttons at the right of the dialog. Each button refers to a message class. To change the class associated with a message, select the message in the list box and then click the button corresponding with the desired message class.

Example:

To define message `D51 could not open statistic log file` (warning message) as an error message:

1. Click the **Warning** tab

   A list of warning messages displays in the list box.

2. Click the string `D51 could not open statistic log file` in the list box.
3. Click the **Error** button to define the message as an error message.

NOTE You cannot move messages from or to the **fatal** error class.

NOTE	The **move to** buttons are active only when you select messages that can be moved. When you try to move a message that can not be moved to a group, the corresponding move to button is greyed out.

If you want to validate the change you made in the error message mapping, click **OK** to close the **Message Settings** dialog. If you click **Cancel** button, the previous message mapping remains valid.

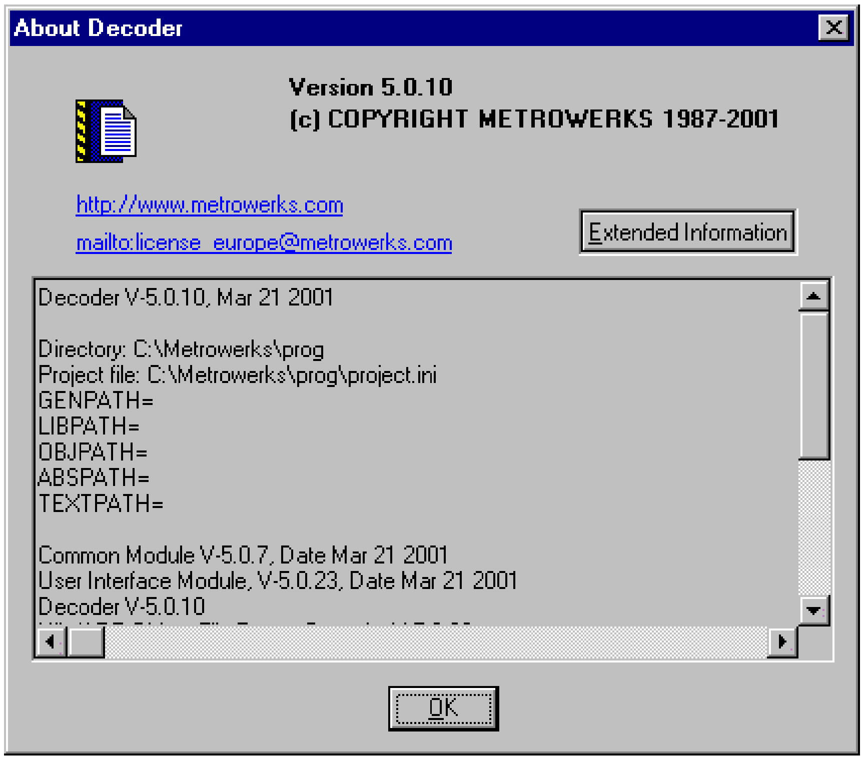
Retrieving Information about an Error Message

You can access information about each message in the list box. Select the message in the list box, then click ***Help***. An information box opens, which contains a more detailed description of the error message as well as a small example of code that could produce the error. If you select several messages, help for the first message displays. If you select no message, pressing **F1** shows help for the dialog.

About Dialog

The **About** dialog (Figure 2.14) displays when you select **Help > About** from the pull-down menus. This dialog shows the current directory and the versions of Decoder components, with the version displayed at the top of the dialog. Click **OK** to close the dialog.

Figure 2.14 About Dialog



Version 5.0.10
(c) COPYRIGHT METROWERKS 1987-2001

http://www.metrowerks.com
mailto:license_europe@metrowerks.com

```
Decoder V-5.0.10, Mar 21 2001

Directory: C:\Metrowerks\prog
Project file: C:\Metrowerks\prog\project.ini
GENPATH=
LIBPATH=
OBJPATH=
ABSPATH=
TEXTPATH=

Common Module V-5.0.7, Date Mar 21 2001
User Interface Module, V-5.0.23, Date Mar 21 2001
Decoder V-5.0.10
```

Specifying the Input File

There are different ways to specify the decode file to be processed. During processing, the software sets options according to configurations that you specified in Decoder dialogs.

Before starting the decoding process of a file, use your editor to specify a working directory.

Use the Command Line in the Tool Bar to Decode

This section explains how to use the command line to process files. The command line lets you enter a new file name and additional Decoder options.

Processing a File Already Run

You can display the previously executed command using the arrow at the right of the command line. Select a command by clicking it, which puts it on the command line. The software processes the file you choose after you click the **Decode** button in the tool bar or press the **Enter** key.

File - Decode...

When you select **File > Decode...**, a standard open file dialog displays. Browse to the file you want to process. The software processes the file you choose after you click the **Decode** button in the tool bar or press the **Enter** key.

Drag and Drop

You can drag a file from other programs (such as the File Manager or Explorer) and drop it into the Decoder main window. The software processes the dropped file after you release the mouse button.

If the dragged file has a `.ini` extension, it is loaded and treated as a configuration file, not as a file to be decoded.

Message and Error Feedback

After making, there are several ways to check for different errors or warnings. The format of an error message looks like this:

```
<msgType> <msgCode>: <Message>
```

Examples:

```
Could not open the file 'Fibo.abs'

FATAL D50: Input file 'Fibo.abs' not found

*** command line: 'Fibo.abs' ***

Decoder: *** Error occurred while processing! ***
```

The second example shows that messages from called applications are also displayed, but only if an error occurs. They are extracted from the error file if the application called has reported an error.

Using Information from the Main Window

Once a file has been processed, the Decoder window content area displays the list of detected errors or warnings. Use your editor of choice to open the source file and correct the errors.

Using a User-defined Editor

You must first configure the editor you want to use for message or error feedback in the **Configuration** dialog. Once a file has been processed, you can double-click on an error message. Your selected editor opens automatically and points to the line containing the error.

3

Environment

This chapter describes the environment variables used by the Decoder. Some of them may be used by other tools (for example, Macro Assembler, Compiler, ...).

Click any of the following links to jump to the corresponding section of this chapter:

- Settings
- Paths
- Line Continuation
- Environment Variables

Settings

Various settings for the Decoder may be set in an environment using environment variables. The syntax is always the same and given below:

```
Parameter = KeyName = ParamDef.
```

NOTE *No* blank spaces are allowed in the definition of an environment variable.

Example:

```
GENPATH=C:\INSTALL\LIB;D:\PROJECTS\TESTS;\usr\local\lib;
```

These parameters may be defined in several ways:

- Using system environment variables supported by your operating system.
- Putting the definition into a project file (usually `project.ini`)
- Putting the definitions in a file called `DEFAULT.ENV` (.hidefaults for UNIX) in the project directory.
- Putting definitions in a file given by the value of the system environment variable `ENVIRONMENT`.

NOTE The maximum length of environment variable entries in the `DEFAULT.ENV`/`.hidefaults` is 1024 characters.

When looking for an environment variable, all programs first search the system environment, then the current project file (usually `project.ini`), then the `DEFAULT.ENV` file. If no definition can be found, a default value is assumed.

NOTE You can set the project directory via the `DEFAULTDIR` system environment variable.

NOTE The environment may also be changed using the `-Env: Set Environment Variable` Decoder option.

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 following the syntax below:

```
PathList = DirSpec {";" DirSpec}.

DirSpec  = ["*"] DirectoryName.
```

Example:

```
GENPATH=C:\INSTALL\LIB;D:\PROJECT\TEST;\usr\loc\hiware\lib;\home\me
```

If a directory name is preceded with an asterisk, programs recursively search in the given directory and subdirectories. Directories are parsed in the order they appear in the path list.

Example:

```
LIBPATH=*C:\INSTALL\LIB
```

NOTE Some DOS/UNIX environment variables (like `GENPATH`, `LIBPATH`, etc.) are used. For more detail refer to “Environment” section.

Environment variables are stored usually in the project file (for example, project.pjt) and can be modified in the IDF. Adapt the project properties where the environment can be modified.

NOTE When using an external editor (WinEdit, Codewright), do *not* set the system environment variable `DEFAULTDIR`. If you do so and this variable does not contain the project directory given in the editor's project configuration, files might not be located where you expect them to be saved.

Line Continuation

It is possible to specify an environment variable in an environment file (`project.pjt` or `.hidefaults`) over multiple lines with the line continuation character '\':

Example:

```
COMPOPTIONS=\
-W2 \
-Wpd
```

This is the same as

```
COMPOPTIONS=-W2 -Wpd
```

But this feature is interpreted differently when used with paths, for example:

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

will be interpreted as

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

To avoid such problems, use a semicolon';' at the end of a path if there is a '\' at the end, as shown below:

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

Environment Variables

This section describes each environment variable available for the Decoder. Variables are listed in alphabetical order. Topics describing each variable are in Table 3.1.

Table 3.1 Environment Variables

Topic	Description
Tools	For some environment variables, a synonym also exists. Those synonyms may be used for earlier releases of the Decoder and will be removed in the future. A synonym has lower precedence than the environment variable.
Synonym	Specifies the syntax of the option in EBNF format.
Syntax	Describes and lists optional and required arguments for the variable.
Arguments	Shows the default setting for the variable or none.
Description	Provides a detailed description of the option and how to use it.
Example	Gives an example of usage, and effects of the variable when possible. Shows an entry in the `default.env` for PC or in the `.hidefaults` for UNIX.

DEFAULTDIR: Current Directory

Tools:	Compiler, Assembler, Linker, Decoder, Debugger, Librarian, Maker
Synonym:	none
Syntax:	`DEFAULTDIR= <directory>.`
Arguments:	<directory>: Default current directory
Default:	none
Description:	With this environment variable the default directory for all tools may be specified. All tools indicated above will take 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 an environment variable at system level (global environment variable). It cannot be specified in a default environment file (`DEFAULT.ENV`/`.hidefaults`).

Example:

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

ENVIRONMENT: Environment File Specification

Tools:	Compiler, Linker, Decoder, Debugger, Librarian, Maker
Synonym:	`HIENVIRONMENT`
Syntax:	`ENVIRONMENT= <file>.`
Arguments:	<file>: file name with path specification
Default:	`DEFAULT.ENV` on PC, .hidefaults on UNIX
Description:	You must specify this variable at the system level. Usually the Decoder looks in the current directory for an environment file named `DEFAULT.ENV` (`.hidefaults` on UNIX). Using `ENVIRONMENT` (set in `AUTOEXEC.BAT` (DOS) or `.cshrc` (UNIX)), a different file name may be specified.

NOTE This is an environment variable at system level (global environment variable). It cannot be specified in a default environment file (`DEFAULT.ENV/.hidefaults`).

Example: `ENVIRONMENT=\METROWERKS\prog\global.env`

GENPATH: Defines Paths to Search for Input Files

Tools:	Compiler, Assembler, Linker, Decoder, Debugger
Synonym:	`HIPATH`
Syntax:	`GENPATH= {<path>}.`
Arguments:	<path>: Paths separated by semicolons
Description:	The Decoder will look for the required input files (binary input files and source files) first in the project directory, then in the directories listed in the environment variable `GENPATH`.

NOTE If a directory specification in this environment variable starts with an asterisk, the whole directory tree is searched recursively. All subdirectories and their subdirectories are searched. Within one level in the tree, search order of the subdirectories is indeterminate.

Example: `GENPATH=\obj;..\..\lib;`

TEXTPATH: Text Path

Tools:	Compiler, Assembler, Linker, Decoder
Synonym:	None
Syntax:	`TEXTPATH= {<path>}.`
Arguments:	<path>: Paths separated by semicolons
Description:	When you define this environment variable, the Decoder stores the list file it produces in the first directory specified. If `TEXTPATH` is not set, the generated `.LST` file is stored in the directory in which the binary input file was found.
Example:	`TEXTPATH=\sources ..\..\headers;\usr\local\txt`

4

Input and Output Files

This chapter describes Decoder input and output files.

Click any of the following links to jump to the corresponding section of this chapter:

- Input Files
- Output Files

Input Files

Input files include the following file types:

- Absolute files
- Object files
- Motorola S-Record files
- Intel Hex files

Absolute Files

The decoder takes any file as input, it does not require the file name to have a special extension. However, we suggest that all your absolute file names have extension "`.ABS`". Absolute files will be searched first in the project directory and then in the directories listed in `GENPATH`. The absolute file must be a valid ELF/DWARF V1.1, ELF/DWARF V2.0 or HIWARE absolute file.

NOTE For HIWARE absolute files, no source information is decoded because the absolute file does not contain source information.

Object File

The decoder takes any file as input, it does not require the file name to have a special extension. However, we suggest that all your relocatable file names have extension

"`.o`". Object files will be searched first in the project directory and then in the directories listed in `GENPATH`. The object file must be a valid ELF/DWARF V1.1, ELF/DWARF V2.0, or HIWARE relocatable file.

Motorola S-Record Files

For Motorola S-Record files, the processor must be specified with the option -Proc: Set Processor. Otherwise only the structure of the S-Record file is printed, but the code is not disassembled.

Intel Hex Files

For Intel Hex files the same applies as for Motorola S-Record Files. To disassemble the code, the processor must be specified with the option -Proc: Set Processor.

Output Files

After a successful decoding session, the Decoder generates a listing file containing the disassembled instructions generated by each source statement. This file is written to the directory given in the environment variable `TEXTPATH`. If that variable contains more than one path, the listing file is written in the first directory given. If this variable is not set, the listing file is written in the directory containing the binary input file. Listing files always get the extension "`.LST`".

In a standard listing file, the code depends on the target. A sample listing is as follows:

```
DISASSEMBLY OF: '.text' FROM 331 TO 416 SIZE 85 (0X55)
Source file: 'Y:\DEMO\WAVE12C\fibo.c'
    8: unsigned int Fibonacci(unsigned int n)
Fibonacci:
00000867 1B98         LEAS  -8,SP
00000869 3B           PSHD
   13:   fib1 = 0;
0000086A C7           CLRB
0000086B 87           CLRA
0000086C 6C88         STD   8,SP
   14:   fib2 = 1;
0000086E 52           INCB
0000086F 6C84         STD   4,SP
   15:   fibo = n;
00000871 EE80         LDX   0,SP
00000873 6E86         STX   6,SP
   16:   i = 2;
00000875 58           ASLB
00000876 6C82         STD   2,SP
   17:   while (i <= n) {
00000878 2011         BRA   *+19   ;abs = 088B
   18:     fibo = fib1 + fib2;
0000087A EC88         LDD   8,SP
0000087C E384         ADDD  4,SP
0000087E 6C86         STD   6,SP
   19:     fib1 = fib2;
00000880 EE84         LDX   4,SP
00000882 6E88         STX   8,SP
   20:     fib2 = fibo;
00000884 6C84         STD   4,SP
   21:     i++;
00000886 EE82         LDX   2,SP
00000888 08           INX
00000889 6E82         STX   2,SP
   17:   while (i <= n) {
0000088B EC82         LDD   2,SP
0000088D AC80         CPD   0,SP
0000088F 23E9         BLS   *-21   ;abs = 087A
   23:   return(fibo);
00000891 EC86         LDD   6,SP
   24: }
00000893 1B8A         LEAS  10,SP
00000895 3D           RTS
```

5

Decoder Options

The Decoder offers options that you can use to control its operation. Options are composed of a dash (or minus sign) followed by one or more letters or digits. You can specify decoder options on the command line. Command line options are not case sensitive; for example, –`otest.lst` is the same as –`OTEST.LST`.

NOTE Arguments for an option must not exceed 128 characters.

Anything not starting with a dash is the name of a parameter file to be linked. Options for the HIWARE object file format may differ from the options for decoding ELF/DWARF binaries.

Click the following link to jump to the corresponding section of this chapter:

- Using Options

Using Options

This section lists and describes each Decoder option.

Option Topics

Table 5.1 describes topics in each option listing.

Table 5.1 Decoder Option Topics

Topic	Description
Group	Specifies one of four option groups: - OUTPUT: Format and content of the listing file. - INPUT: Control and specification of input files - HOST: Host and Operation System-dependent options - MESSAGE: Control of error and message output
Syntax	Specifies the syntax of the option in EBNF format.

Table 5.1 Decoder Option Topics (*continued*)

Topic	Description
Arguments	Describes and lists optional and required arguments for the option.
Default	Shows the option's default setting
Description	Provides a detailed description of the option and how to use it

Special Modifiers

You can use special modifiers with some options, although some modifiers may not make sense for all options. Table 5.2 lists and describes these modifiers.

Table 5.2 Supported Modifiers

Modifier	Description
%p	path including file separator
%N	file name in strict 8.3 format
%n	file name without extension
%E	extension in strict 8.3 format
%e	extension
%f	path + file name without extension
%”	a double quote (“) if the file name, path or extension contains a space
%’	a single quote (‘) if the file name, path or extension contains a space
%(ENV)	replaces it with contents of an environment variable
%%	generates a single ‘%’

Examples:

For these examples we assume that our actual file name (base file name for the modifiers) is:

```
c:\Metrowerks\my demo\TheWholeThing.myExt
```

%p gives the path only with a file separator:

```
c:\Metrowerks\my demo\
```

%N results in the file name in 8.3 format, that is the name with only 8 characters:

```
TheWhole
```

%n returns just the file name without extension:

```
TheWholeThing
```

%E gives the extension in 8.3 format, that is the extension with only 3 characters:

```
myE
```

%e is used for the whole extension:

```
myExt
```

%f gives the path plus the file name:

```
c:\Metrowerks\my demo\TheWholeThing
```

Because the path contains a space, using %" or %' is recommended: Thus %"%f%" gives:

```
"c:\Metrowerks\my demo\TheWholeThing"
```

where %'%f%' gives:

```
'c:\Metrowerks\my demo\TheWholeThing'
```

Using %(envVariable) an environment variable may be used too. A file separator after %(envVariable) is ignored if the environment variable is empty or does not exist. For example, $(TEXTPATH)\myfile.txt is replaced with

```
c:\Metrowerks\txt\myfile.txt
```

if TEXTPATH is set to

```
TEXTPATH=c:\Metrowerks\txt
```

But is set to

```
myfile.txt
```

if TEXTPATH does not exist or empty.

%% may be used to print a percent sign. %e%% gives:

```
myExt%
```

-A: Print Full Listing

Group:	OUTPUT
Syntax:	`"-A".`
Arguments:	none
Default:	none

File Format:	only HIWARE. ELF Object files are not affected by this option.
Description:	Print a listing with the header information of the object file.
Example:	Listing with command line `fibo.o -A`:

```
*** Header information ***
Program Version           2700
Format Version            2
File Id                   129
flags                     0
processor family          11
processor type            1
Unitname                  fibo.abs
Username                  PFR
Program time string       Feb 25 1998
Creation time string      Wed Feb 25 11:43:22 1998
CopyRight
*** Directory information for Absfile***
Is romlib?   0
Init start:end   32774:32774
Code beg:end     32768:32939
Data beg:end     384:4096
Total number of objects   7

At address: 8000 code size: 40
00008000 1410         ORCC  #16
.........
```

-C: Write Disassembly Listing With Source Code

Group:	OUTPUT.
Syntax:	`"-C"`.
Arguments:	none
Default:	none
File Format:	only HIWARE. ELF Object files are not affected by this option.
Description:	This option setting is default for the HIWARE object files as input. When this option is specified, the Decoder decoding HIWARE object files writes the source code within the disassembly listing.
Example:	Listing with command line `fibo.o -C` (code depends on target):

```
    8:  unsigned int Fibonacci(unsigned int n)
    9:  {
   10:    unsigned fib1, fib2, fibo;
   11:    int i;
   12:
00000000 3B           PSHD
00000001 3B           PSHD
   13:    fib1 = 0;
00000002 C7           CLRB
00000003 87           CLRA
   14:    fib2 = 1;
00000004 52           INCB
00000005 6C82         STD   2,SP
   15:    fibo = n;
00000007 EE80         LDX   0,SP
   16:    i = 2;
.................
```

-D: Decode DWARF Sections

Group:	OUTPUT
Syntax:	`"-D"`
Arguments:	none
Default:	Disabled
File Format:	only ELF. HIWARE Object files are not affected by this option.
Description:	When you specify this option, DWARF section information is also written to the listing file. Decoding from the DWARF section inserts this information in the listing file:

Information about source/code references:

```
.debug_line
  0x4 Version 2
  0x6 PrologLen 122l
  0xa MinInstrLen 1c
  0xb DefIsStmt 0c
  0xc LineBase 0c
  0xd LineRange 4c
  0xe DW2L_OpcodeBase 9c
  0xf Opcodelengths : 0c 1c 1c 1c 1c 0c 0c 0c 1c
Includedir :
 0x19 File  1: Y:\DEMO\WAVE12C\fibo.c, 0, 0, 0
 0x33 File  2: y:\LIB\ELF12C\hidef.h, 0, 0, 0
 0x4c File  3: y:\LIB\ELF12C\default.sgm, 0, 0, 0
 0x69 File  4: y:\LIB\ELF12C\stddef.h, 0, 0, 0
 0x84 Set Addr  867(2151):   ADDR  FILE LINE COL STMT BASIC
 0x8b set column         :   867    1     1   14   0     0
 0x8d advance line       :   867    1     8   14   0     0
 0x8f negate stmt        :   867    1     8   14   1     0
 0x90 negate stmt        :   867    1     8 14   0     0
...
```

Information about argument location for local variables:

```
.debug_loc
    0 Start 867, End 869  (2)DW_OP_breg15 0(0)
  0xc Start 869, End 86a  (2)DW_OP_breg15 8(8)
 0x18 Start 86a, End 895  (2)DW_OP_breg15 10(a)
 0x24 Start 895, End 896  (2)DW_OP_breg15 0(0)
 0x30 0, 0 : end of location-list
```

Symbol Debug information:

```
DWARF: .debug_info (1053) [0x734]
Compi.Unit Header: size 304, version 2, abbrev 0, addrsize 4
  0xb Abbrevation 128 ,compile_unit
  0xd name         string      fibo.c
 0x14 producer     string      HIWARE
 0x1b comp_dir     string      Y:\DEMO\WAVE12C
 0x2b language     udata       DW_LANG_C89
 0x2c stmt_list    data4       0(0)
```

Frame Debug Information:

```
.debug_frame
    0 CIE Information  0x8 Version 1
  0x9 Augmentor Hiware CFA 1.0
 0x18 CodeAlign: 1, DataAlign: 1, ReturnAddr-Column: 18
 0x1b instruction                    PC   FP(Reg) R[ 0] R[
1] R[ 2] R[ 3] R[ 4] R[ 5] R[ 6] R[ 7] R[ 8] R[ 9] R[10]
R[11] R[12] R[13] R[14] R[15] R[16] R[17] R[18] R[19] R[20]
R[21] R[22] R[23] R[24] R[25] R[26] R[27] R[28] R[29] R[30]
```

```
R[31]
0x1bstart-values   84d: 0(15)
 0x1b Def CFA Register reg: 15,
 0x1d Def CFA Offset ofs: 0
 0x1f Offset: reg 18, Ofs: 0
 0x21 Undefined reg: 0
 0x23 Undefined reg: 1
```

NOTE We recommend that you specify the `-E` option when the `-D` option is activated.

-E: Decode ELF sections

Group:	OUTPUT
Syntax:	`"-E".`
Arguments:	none
Default:	none
File Format:	only ELF. HIWARE Object files are not affected by this option.
Description:	When you specify this option, ELF section information is also written to the listing file. Decoding from the ELF section inserts the following information in the listing file:

ELF Header Information:

```
File: Y:\DEMO\WAVE12C\fibo.abs

Ident: ELF with 32-bit objects, MSB encoding, Version 1

Type:  Executable file,   Machine:  Motorola HC12,   Vers:  1

Entry point: 83D

Elf flags: 0

ElfHSiz: 34

ProgHOff:      34,   ProgHSi:     20,   ProgHNu:      6

SectHOff:     E3A,   SectHSi:     28,   SectHNu:      19,   SectHSI:       18
```

ELF Program header Table Information (when available). Usually this section is available only for absolute files.

```
PROGRAM HEADER TABLE - 6 Items

Starts at:       34, Size of an entry:     20, Ends at:       F4

  NO   TYPE        OFFSET     SIZE VIRTADDR  PHYADDR  MEMSIZE   FLAGS ALIGNMNT

  0 - PT_PHDR         34       C0

  1 - PT_LOAD         F4        0        0      800        4       6        0

  2 - PT_LOAD         F4       AE        0      810       AE       1        0
```

ELF Section Header Table Information:

```
SECTION HEADER TABLE - 19 Items

Starts at:      E3A, Size of an entry:     28, Ends at:     1132

String table is in section: 12

 NO NAME                 TYPE     FLAGS OFFSET   SIZE   ADDR ALI RECS LINK INFO

  0-                     NULL                0      0      0   0    0    0    0

  1-.common              NOBITS    WA       F4      4    800   0    0    0    0

  2-.init                PROGBITS  AX       F4     3D    810   0    0    0    0

  3-.startData           PROGBITS  AX      131     1A    84D   0    0    0    0

  4-.text                PROGBITS  AX      14B     55    867   0    0    0    0

  5-.copy                PROGBITS  AX      1A0      2    8BC   0    0    0    0

  6-.stack               NOBITS    WA      1A2    100    B00   0    0    0    0

  7-.vectSeg0_vect       PROGBITS  AX      1A2      2   FFFE   0    0    0    0
```

Symbol Table Information:

```
SYMBOL TABLE: .symtab - 13 Items

Starts at:      1A4, Size of an entry:     10, Ends at:      274

String table is in section: 9

First global symbol is in entry no.: 8

 NO NAME                                     VALUE  SIZE BIND   TYPE     SECT

  0-                                             0     0 LOCAL  NOTYPE

  1-                                             0     0 LOCAL  SECTION     1
  2-                                                      0      0 LOCAL   SECTION     2
  3-Init                                                810     2D LOCAL   FUNC        2
  4-                                                      0      0 LOCAL   SECTION     3
  5-                                                      0      0 LOCAL   SECTION     4
```

Relocation Section Information:

```
RELOCATION TABLE RELA: .rela.init - 1 Items
Starts at:      2AA, Size of an entry:      C, Ends at:      2B6
Symbol table is in section: 8
Binary code/data is in section: 2
   NO     OFFSET            SYMNDX TYP   ADDEND SYMNAME
    0 -     2163      873        3   3     4107 Init
```

Hexadecimal dump from all sections defined in the binary file:

```
HEXDUMP OF: .init FROM 244 TO 305 SIZE 61 (0X3D)
OFFSET  +0 +1 +2 +3 +4 +5 +6 +7 : +8 +9 +A +B +C +D +E +F  ASCII DATA
000000  FE 08 55 FD 08 53 27 10 : 35 ED 31 EC 31 69 70 83  ..U..S'.5.1.1ip.
000010  00 01 26 F9 31 03 26 F0 : FE 08 57 EC 31 27 0D ED   .&.1.&...W.1'..
000020  31 18 0A 30 70 83 00 01 : 26 F7 20 EF 3D FC 08 4D  1..0p. .&. .=..M
000030  26 03 FF 08 51 07 C9 15 : FB 00 04 20 F0           &...Q.... . .
```

-Ed: Dump ELF sections in LST File

Group:	OUTPUT
Syntax:	`"-Ed".`
Arguments:	none
Default:	none
File Format:	only ELF. HIWARE object files are not affected by this option.
Description:	This option generates a HEX dump of all ELF sections.

NOTE The related option “-E” shows the information contained in ELF sections in a more readable form.

-Env: Set Environment Variable

Group:	HOST
Syntax:	`"-Env" <Environment Variable> "=" <Variable Setting>.`
Arguments:	<Environment Variable>: Environment variable to be set <Variable Setting>: Environment variable value
Default:	none
Description:	This option sets an environment variable.

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

-F: Object File Format

Group:	INPUT
Syntax:	`"-F" (“A” | “E” | “I” | “H” | “S”).`
Arguments:	none
Default:	`“-FA”.`
Description:	The decoder is able to decode different object file formats. This option defines which object file format should be decoded. With “-FA”, the decoder determines the object file format automatically. With “-FE”, this can be overridden and only ELF files are correctly decoded. With “-FH” only HIWARE files are decoded. With “-FS“ only S-Record files can be decoded. With “-FI“ Intel Hex files can be decoded.

NOTE This option defines the Object File Format, which also defines the format of absolute files and libraries. It does not only affect object files.

NOTE	Many other options only effect a specific object file format. See the corresponding option for details.

NOTE	To decode a S-Record or Intel Hex file, use the option -Proc: Set Processor to specify the processor.

-H: Prints the List of All Available Options

Group:	OUTPUT
Syntax:	`"-H"`.
Arguments:	none
Default:	none
Description:	Prints the list of all Decoder options. The options are sorted by Group. Options in the same group are sorted alphabetically.

-L: Produce inline assembly file

Group:	OUTPUT.
Syntax:	`"-L"`
Arguments:	none
Default:	none
File Format:	only HIWARE. ELF Object files are not affected by this option.
Description:	The output listing is an inline assembly file without additional information, but in C comments.
Example:	Part of Listing with command line `fibo.o -L` (code depends on target):

```
unsigned int Fibonacci(unsigned int n)
{
  unsigned fib1, fib2, fibo;
  int i;
  asm{
                 CLRB
                 CLRA
                 INCB
                 STD   2,SP
                 LDX   0,SP
                 LDY   #2
                 SEX   A,D
                 BRA   LBL25
        LBL16:   ADDD  2,SP
                 ...
                 RTS

    }

}
```

-Lic: Print license information

Group:	Various
Syntax:	`"-Lic"`
Arguments:	none
Default:	none
Description:	This option shows the current state of the license information. When no full license is available, the Decoder runs in demo mode. Currently, there are no restrictions in the demo mode, but future versions might limit the size of the listing or the size of the input files.
Example:	none

-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. For example, if tool or feature is a demo version or full version. Because the option analyzes every file in the directory, this takes a long time.
Example:	none

-N: Display Notify Box

Group:	Various
Syntax:	`"-N"`
Arguments:	none
Default:	none
Description:	Causes the Decoder to display an alert box if there was an error during decoding. This is useful when running a make file (refer to Make Utility), since the Decoder 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 on the PC version of the Decoder. The UNIX version does not accept “-n” as an option string.

Example:	none

-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 there was an error. This ‘beep’ may be switched off with this option.
Example:	none

-NoEnv: Do not use Environment

Group:	Startup. This option cannot be specified interactively.
Syntax:	`"-NoEnv"`
Arguments:	none
Default:	none
Description:	This option can only be specified at the command line while starting the application. It can not be specified in any other circumstance, including the default.env file or command line. When this option is given, the application does not use any environment (default.env, project.ini or tips file).
Example:	`decoder.exe -NoEnv`

-NoSym: No Symbols in Disassembled Listing

Group:	OUTPUT
Syntax:	`"-NoSym"`
Arguments:	none
Default:	none
Description:	No Symbols are printed in the disassembled listing.

NOTE In previous versions of the Decoder, this option was called “-N”. It was renamed because of a conflict with the common option “-N”, which was not present in previous versions. The option -NoSym has no effect when decoding abs file. As the abs file do not contain any relocation information, it is not possible to display symbol names in the disassembly listing.

Example: Part of Listing with command line `“fibo.o -NoSym”`

```
DISASSEMBLY OF: '.text' FROM 531 TO 664 SIZE 133 (0X85)
Source file: 'fibo.c'
   19: unsigned int Fibonacci(unsigned int n)
Fibonacci:
00000000 1B98         LEAS  -8,SP
00000002 3B           PSHD
   24:   fib1 = x[0] + f + g;
00000003 FC0000       LDD   $0000
00000006 F30000       ADDD  $0000
00000009 F30000       ADDD  $0000
0000000C 6C88         STD   8,SP
   25:   fib2 = x[1];
0000000E FC0002       LDD   $0002
00000011 6C84         STD   4,SP
```

-O: Defines Listing File Name

Group: OUTPUT

Syntax: `"-O" <FileName>`

Arguments: <fileName>: Name of listing file that must be generated by the decoding session.

Default: none

Description: This option defines the name of the output file to be generated.

Example: `-O=TEST.LST`

The decoder will generate a file named `TEST.LST`.

-Proc: Set Processor

Group: INPUT

Syntax: `"-Proc="<ProcessorName>[“:” <Derivative>].`

Arguments: <ProcessorName>: Name of a supported processor.
<DerivativeName>: Name of supported derivative.

Default: none

Description: This option specifies which processor should be decoded. For object files, libraries and applications, the processor is usually detected automatically. For Motorola S-Record and Intel Hex files, however, the decoder cannot determine which CPU the code is for, and therefore the processor must be specified with this option to get a disassembly output. Without this option, only the structure of a S-Record file is decoded.

The following values are supported:

HC05, HC08, HC08:HCS08, HC11, HC12, HC16, M68k, MCORE, PPC, 8500, 8300, 8051, ST7 and XA

Example: `decoder.exe fibo.s19 -proc=HC12`

-T: Shows the Cycle Count for each Instruction

Group:	OUTPUT
Syntax:	`"-T"`
Arguments:	none
Default:	none
Description:	If you specify this option, each instruction line contains the count of cycles in '[',']' braces. The cycle count is written before the mnemonics of the instruction. Note that the cycle count display is not supported for all architectures.
Example:	Part of Listing (HC12, ELF) with command line `fibo.o -T`

```
DISASSEMBLY OF: '.text' FROM 531 TO 664 SIZE 133 (0X85)
Source file: 'X:\CHC12E\DEMO\ELF12C\fibo.c'
   19: unsigned int Fibonacci(unsigned int n)
Fibonacci:
00000000 1B98         [2]    LEAS  -8,SP
00000002 3B           [2]    PSHD
   24:   fib1 = x[0] + f + g;
00000003 FC0000       [3]    LDD   x
00000006 F30000       [3]    ADDD  f
00000009 F30000       [3]    ADDD  g
0000000C 6C88         [2]    STD   8,SP
   25:   fib2 = x[1];
0000000E FC0002       [3]    LDD   x
00000011 6C84         [2]    STD   4,SP
   26:   fibo = 0;
00000013 C7           [1]    CLRB
00000014 87           [1]    CLRA
00000015 6C86         [2]    STD   6,SP
   27:   i = 2;
00000017 C602         [1]    LDAB  #2
00000019 6C82         [2]    STD   2,SP
```

-V: Prints the Decoder Version

Group: Various

Syntax: `"-V"`

Arguments: none

Default: none

Description: Prints the Decoder current directory and version information. Version of the complete Decoder is the main version. Additionally, version numbers for the HIWARE and ELF Object File Format decoding parts are printed separately.

-View: Application Standard Occurrence (PC)

Group: HOST

Syntax: `"-View" <kind>`

Arguments: <kind> is one of:
"Window": Application window has 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: Normally the application (linker, compiler, ...) is started as a normal window if no arguments are given. If the application is started with arguments (for example, from the maker to compile/link a file) then the application is running minimized to allow for batch processing. However, with this option the behavior may be specified.
Using -ViewWindow the application is visible with its normal window.
Using -ViewMin the application is visible iconified (in the task bar).
Using -ViewMax the application is visible maximized (filling the screen).
Using -ViewHidden the application processes arguments (files to be compiled/linked) in the background (no window/icon in the taskbar visible). However, if you are using the “-N: Display Notify Box” option a dialog box is still possible.

Example: `-ViewMin fibo.o`

-WErrFile: Create "err.log" Error File

Group: MESSAGE

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

Arguments: none

Default: err.log is created/deleted.

Description: Error feedback from the compiler to called tools is now done with a return code. In 16 bit windows environments, this was not possible. An err.log file with the number of errors written was used to signal an error. To state no error, the err.log file was deleted. Using UNIX or WIN32, there is now a return code available, so this file is no longer needed when UNIX / WIN32 applications are involved. 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.

-Wmsg8x3: Cut file names in Microsoft format to 8.3

Group:	MESSAGE
Syntax:	`"-Wmsg8x3"`
Arguments:	none
Default:	none
Description:	Some editors (early versions of WinEdit) expect the file name in the Microsoft message format in strict 8.3 format. That means the file name can have at most 8 characters with not more than a 3 character extension. With Win95 or WinNT, longer file names are possible. With this option, the file name in the Microsoft message is truncated to the 8.3 format.

NOTE This option is only present in PC versions of the Decoder. UNIX decoders do not accept `"-Wmsg8x3"`.

Example:	none

-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)
Description:	With this option it is possible to change the error message color. The value to be specified has to be a RGB (Red-Green-Blue) value, and specified in decimal.
Example:	`-WmsgCE255` changes error messages to blue

-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)
Description:	With this option it is possible to change the fatal message color. The value to be specified has to be a RGB (Red-Green-Blue) value, and specified in decimal.
Example:	`-WmsgCF255` changes the fatal messages to blue

-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)
Description:	With this option it is possible to change the information message color. The value to be specified has to be a RGB (Red-Green-Blue) value, and specified in decimal.
Example:	`-WmsgCI255` changes information messages to blue

-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)
Description:	With this option it is possible to change the user message color. The value to be specified has to be a RGB (Red-Green-Blue) value, and specified in decimal.
Example:	`-WmsgCU255` changes user messages to blue

-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)
Description:	With this option it is possible to change the warning message color. The value to be specified has to be a RGB (Red-Green-Blue) value, and specified in decimal.
Example:	`-WmsgCW0` changes warning messages to black

-WmsgFb: Set message file format for batch mode

Group:	MESSAGE
Syntax:	`"-WmsgFb" ["v" | "m"]`
Arguments:	"v": Verbose format. "m": Microsoft format.
Default:	`“-WmsgFbm”`
Description:	If the Decoder has been started with arguments, the Decoder operates in batch mode. There is no window visible and the Decoder terminates after job completion. In batch mode, messages are written to a file instead of the screen. This file only contains the messages. By default, the Decoder uses a Microsoft message format to write messages (errors, warnings, information messages). 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 decoding, because the decoder has to write more information into the message file.

Example:	none

-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 Decoder is started without additional arguments (for example, files to be decoded), the Decoder is in interactive mode (that is, a window is visible). By default, the Decoder uses the verbose error file format to write the 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 compiler has to write less information to the screen.

Example:	none

-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\n".`
Description:	With this option it is possible to modify the default message format in batch mode. Following formats are supported (assume that the input file is x:\hiware\mysourcefile.object).

```
Format Description        Example
-----------------------------------
%s     Source Extract
%p     Path                 x:\hiware\
%f     Path and name        x:\hiware\source
%n     File name            mysourcefile
%e     Extension            .object
%N     File (8 chars)       mysource
%E     Extension (3 chars)  .obj
%l     Line                 3
%c     Column               47
%o     Pos                  1234
%K     Uppercase kind       ERROR
%k     Lowercase kind       error
%d     Number               M1815
%m     Message              text
%%     Percent              %
\n     New line
```

Example: none

-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, you can modify the default message format in interactive mode. Following formats are supported (assume that the source file is x:\hiware\mysourcefile.object):

```
Format Description        Example
------------------------------------
%s     Source Extract
%p     Path                 x:\hiware\
%f     Path and name        x:\hiware\source
%n     File name            mysourcefile
%e     Extension            .object
%N     File (8 chars)       mysource
%E     Extension (3 chars)  .obj
%l     Line                 3
%c     Column               47
%o     Pos                  1234
%K     Uppercase kind       ERROR
%k     Lowercase kind       error
%d     Number               M1815
%m     Message              text
%%     Percent              %
\n     New line
```

Example: none

-WmsgFonf: Message Format for no File Information

Group: MESSAGE

Syntax: `"-WmsgFonf"<string>`

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

Default: `-WmsgFonf"%K %d: %m\n".`

Description: Sometimes there is no file information available for a message, for example, if a message is not related to a specific file. Then this message format string is used. Following formats are supported:

```
Format Description        Example
------------------------------------
%K     Uppercase kind       ERROR
%k     Lowercase kind       error
%d     Number               C1815
%m     Message              text
%%     Percent              %
\n     New line
```

Example: none

-WmsgFonp: Message Format for no Position Information

Group: MESSAGE

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. For example, if a message is not related to a certain position then this message format string is used. Following formats are supported. Assume that the source file is:

`x:\hiware\mysourcefile.object`

```
Format Description        Example
------------------------------------
%p     Path                 x:\hiware\
%f     Path and name        x:\hiware\source
%n     File name            mysourcefile
%e     Extension            .object
%N     File (8 chars)       mysource
%E     Extension (3 chars)  .obj
%K     Uppercase kind       ERROR
%k     Lowercase kind       error
%d     Number               D1815
%m     Message              text
%%     Percent              %
\n     New line
```

Example: none

-WmsgNe: Number of Error Messages

Group: MESSAGE

Syntax: `"-WmsgNe" <number>`

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

Default: 50

Description: With this option, you can set the number of error messages that occur before the Decoder stops.

Example: `“-WmsgNe2”.`

The Decoder stops processing after two error messages.

-WmsgNi: Number of Information Messages

Group:	MESSAGE
Syntax:	`"-WmsgNi" <number>`
Arguments:	<number>: Maximum number of information messages.
Default:	50
Description:	With this option the number of information messages can be set.
Example:	`“-WmsgNi10”.`

Only ten information messages are logged.

-WmsgNu: Disable User Messages

Group:	MESSAGE
Syntax:	`"-WmsgNu" ["=" {"a" | "b" | "c" | "d"}].`
Arguments:	“a”: Disable messages about include files “b”: Disable messages about reading files “c”: Disable messages about generated files “d”: Disable messages about processing statistics “e”: Disable informal messages
Default:	none
Description:	The application produces some messages that are not in the normal message categories (WARNING, INFORMATION, ERROR, FATAL). With this option these messages can be disabled. The idea of this option is to reduce the amount of messages and simplify error parsing of other tools. “a”: This option causes the application to disable messages regarding included files. “b”: This option causes the application to disable messages regarding reading files, for example, files used as input. “c”: Disables messages regarding generated files. “d”: Disables messages regarding statistics, for example, code size and RAM/ROM usage. “e”: Disables informal messages, for example, memory model and floating point format messages.

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

Example:	`“-WmsgNu=c”`

-WmsgNw: Number of Warning Messages

Group:	MESSAGE
Syntax:	`"-WmsgNw" <number>.`
Arguments:	<number>: Maximum number of warning messages.
Default:	50
Description:	Use this option to set the number of warning messages.
Example:	`“-WmsgNw15”`
	Only 15 warning messages are logged.

-WmsgSd: Setting a Message to Disable

Group:	MESSAGE
Syntax:	`"-WmsgSd" <number>`
Arguments:	<number>: Message number to be disabled, for example 1801
Default:	none.
Description:	Use this option to disable a message, so it does not appear in error output.
Example:	none

-WmsgSe: Setting a Message to Error

Group:	MESSAGE
Syntax:	`"-WmsgSe" <number>`
Arguments:	<number>: Message number specified to be an error message, for example 1853
Default:	none.
Description:	Changes a message to an error message.

Example:	none

-WmsgSi: Setting a Message to Information

Group:	MESSAGE
Syntax:	`"-WmsgSi" <number>`
Arguments:	<number>: Message number specified to be an information message, for example 1853
Default:	none
Description:	Changes a message to an information message.
Example:	none

-WmsgSw: Setting a Message to Warning

Group:	MESSAGE
Syntax:	`"-WmsgSw" <number>`
Arguments:	<number>: Error number specified to be a warning message, for example 2901
Default:	none
Description:	Specifies a message to be a warning message.
Example:	none

-WOutFile: Create Error Listing File

Group:	MESSAGE
Syntax:	`"-WOutFile" ("On" | "Off").`
Arguments:	none
Default:	Error listing file is created
Description:	This option creates an error listing file. The error listing file contains a list of all messages and errors created during a compilation. Since the 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 listing file is controlled by the environment variable ERRORFILE.
Example:	`-WOutFileOn`

The error file is created.
`“-WOutFileOff”`

No error file is created.

-WStdout: Write to standard output

Group:	MESSAGE
Syntax:	`"-WStdout" ("On" | "Off")`
Arguments:	none
Default:	output written to stdout
Description:	With Windows applications, the 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 is also written to stdout.
Example:	`“-WStdoutOn”`

All messages are written to stdout.
`-WErrFileOff`

Nothing is written to stdout.

-W1: No Information Messages

Group:	MESSAGE
Syntax:	`"-W1"`
Arguments:	none
Default:	none
Description:	Disables INFORMATION messages, only WARNING and ERROR messages are output.

-W2: No Information and Warning Messages

Group:	MESSAGE
Syntax:	`"-W2"`
Arguments:	none

Default:	none
Description:	Suppresses INFORMATION and WARNING messages, only ERRORs are printed.

-X: Write disassembled listing only

Syntax:	`"-X"`
Arguments:	none
Default:	none
Description:	Writes the pure disassembly listing without any source or comments within the listing.

-Y: Write disassembled listing with source and all comments

Syntax:	`"-Y"`
Arguments:	none
Default:	none
File Format:	only HIWARE. ELF Object files are not affected by this option.
Description:	Writes the origin source and its comments within the disassembly listing.
Example:	none

6

Messages

This chapter describes messages produced by the application. Because of the huge amount of different messages produced, not all of them may be in this document.

Click any of the following links to jump to the corresponding section of this chapter:

- Types of Generated Messages
- Message Details
- List of Messages

Types of Generated Messages

Table 6.1 lists and describes the five types of generated messages.

Table 6.1 Types of Generated Messages

Message	Behavior
Information	A message prints and compilation continues.
Warning	A message prints and processing continues. These messages indicate possible programming errors.
Error	A message prints and processing stops. These messages indicate illegal language usage.
Fatal	A message prints and processing aborts. These messages indicate a severe error, which causes processing to stop.
Disable	The message is disabled. No message is issued and processing continues. The application ignores the Disabled message.

Message Details

If an application generates a message, that message contains a message code as well as a four- to five-digit number. You can use this number to search for a message. The following message codes are supported:

- A = Assembler
- B = Burner
- C = Compiler
- D = Decoder
- L = Linker
- LM = Libmaker
- M = Maker

Messages that the application generates are sorted in ascending order for quick retrieval.

Each message contains a description and usually a short example with a possible solution or tips to fix a problem.

For each message, the type of message is indicated. For example, `[ERROR]` indicates that the message is an error message.

```
[DISABLE, INFORMATION, WARNING, ERROR]
```

indicates that the message is a warning message by default, although you can change that message to DISABLE, INFORMATION or ERROR.

After the message type, there may be an additional entry indicating the language for which the message is generated:

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

List of Messages

This section lists all messages. Message numbers less than 10000 are common to all tools. Not every compiler can issue all messages. For example, many compilers do not support any type of `struct` return. Those compilers will never issue the message `C2500: Expected: No support of class/struct return type`.

D1: Unknown message occurred

```
[FATAL]
```

Description:

The application tried to issue a message that was not defined. This is an internal error that should not occur. Please report this message to your distributor.

Tip:

None

D2: Message overflow, skipping <kind> messages

`[DISABLE, INFORMATION, WARNING, ERROR]`

Description:

The application showed the number of message types specified with the options "-WmsgNi: Number of Information Messages", "-WmsgNw: Number of Warning Messages" and "-WmsgNe: Number of Error Messages". Additional messages of this type are not displayed.

Tip:

Use the options "-WmsgNi: Number of Information Messages", "-WmsgNw: Number of Warning Messages" and "-WmsgNe: Number of Error Messages" to change the number of messages to display.

D50: Input file '<file>' not found

`[FATAL]`

Description:

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

Tips:

Check if the file really exists. Check if you are using a file name containing spaces (in this case you have to quote it).

D51: 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 is not issued in this case.

D52: Error in command line <cmd>

`[FATAL]`

Description

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

D64: Line Continuation occurred in <FileName>

`[DISABLE, INFORMATION, WARNING, ERROR]`

Description:

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

Example:

`Current Default.env:`

...

LIBPATH=c:\hiware\lib\

OBJPATH=c:\hiware\work

...

Is identical to

...

LIBPATH=c:\hiware\libOBJPATH=c:\hiware\work

...

Tips:

To fix it, append a '.' after the '\'

...

LIBPATH=c:\hiware\lib\.

OBJPATH=c:\hiware\work

...

D65: Environment macro expansion message '<description>' for <variablename>

`[DISABLE, INFORMATION, WARNING, ERROR]`

Description:

During an environment variable macro substitution a problem occurred. Possible causes are that the named macro did not exist or a length limitation was reached. Also, recursive macros may cause this message.

Example:

Current variables:

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

Tips:

Check the definition of the environment variable.

D66: Search path <Name> does not exist

`[DISABLE, INFORMATION, WARNING, ERROR]`

Description:

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

Tips:

Check the spelling of your paths. Update the paths when moving a project. Use relative paths.

D1000: Bad hex input file <Description>

`[DISABLE, INFORMATION, WARNING, ERROR]`

Description:

While decoding a Motorola S-Record or an Intel Hex file, the decoder detected incorrect entries in the file. The <Description> gives more detail.

Tips:

Check the descriptive text and if the file passed to the decoder is correct.

D1001: Because current processor is unknown, no disassembly is generated. Use -proc.

`[DISABLE, INFORMATION, WARNING, ERROR]`

Description:

While decoding a Motorola S-Record or an Intel Hex file, the decoder needs to know about the processor used to decode the file with disassembly information. This is needed because these formats do not contain information about the processor.

Tips:

Use the Option -Proc: Set Processor to specify the processor.

Index