

A

Generating List Files

This appendix describes the source listings the compiler can generate and explains how you can control the format of the output. It contains the following sections:

§A.1: *Generating a Source Listing*

§A.2: *Customizing the Listing with Pragma Page, Skip, and Title*

§A.3: *Listing Format*

A.1 Generating a Source Listing

Use option -Hlist To get a listing of your source code, specify option **-Hlist** on the command line (see Chapter 3: *Using Compiler Options*), or turn On toggle **List** (see *toggle List* Chapter 4: *Using Compiler Toggles*).

The source listing goes to standard output unless you specify a different destination as an argument to **-Hlist** on the command line.

A.2 Customizing the Listing with Pragma Page, Skip, and Title

You can use pragmas to specify a title for your listing, control the number of lines per page, and insert blank lines in the listing.

Pragma Page To cause *n* page ejects at some point in the listing, insert:
inserts page ejects

#pragma Page(*n*)

where *n* is the number of form feeds, or the number of pages to eject. Ordinarily you would set *n* to 1 to get a page break.

Pragma Skip To generate n blank lines at some point in the listing, insert:
inserts blank lines

```
#pragma Skip( $n$ )
```

where n is the number of blank lines.

Pragma Title To get a title at the top of each successive page, place the following pragma
puts a title on in the source:
each page

```
#pragma Title("Listing_title")
```

where *Listing_title* is the title to be printed.

Each successive `Title` pragma changes the title for subsequent pages. The only way to title the *first* page is to place the `Title` pragma in the profile and use `#pragma On(List)` at the end of the profile, or wherever the listing should start in the source file.

When you request a listing by specifying option `-Hlist` on the command line, the first page is not titled, because the listing starts before the compiler sees the `Title` pragma.

A.3 Listing Format

Ruler The first line on each page, after any header and title lines, is a “ruler” that defines three fields:

1. the level numbers
2. the line number
3. the line contents

The ruler appears as follows:

```
Levels LINE#|-----1-----2-----3-----4-----5
```

Level numbers Use level numbers to find a missing “}” or comment terminator when the compiler produces a message such as `Unexpected end-of-file`. All three level numbers are initially 0 (zero), but they print as blanks rather than as a 0 (zero).

Scope nesting level The first level number indicates the scope nesting level for **struct** or **union** declarations.

<i>Statement nesting level</i>	The second level number indicates the statement nesting level. It is incremented at the beginning of each “{” and decremented at the corresponding “}”.
<i>Structure initialization nesting level</i>	The third level number indicates the structure initialization nesting level. As with the second level number, it is incremented at the beginning of each “{” and decremented at the corresponding “}”.
<i>First-level include files</i>	<p>A first-level include file named <code>File_name</code> is shown as starting after a line with “+(<code>File_name</code>” in the line-number field, and ending just before a matching “+)<code>File_name</code>” line.</p> <p>The included lines have “+” in the leftmost column of the line-number field, and those lines are numbered independently of the main source file.</p>
<i>Nested include files</i>	An included file <i>inside</i> an include file has an extra “+” on each line for each level of inclusion, except that line numbers take precedence over “+” characters in the line-number field if the “+” characters would otherwise intrude into that field.
<i>Profile</i>	The profile, if any, is listed as an include file.

