

FLOATING POINT FORMATS NOT LINKED : ABNORMAL PROGRAM TERMINATION

This document explains why you might be getting the error FLOATING POINT FORMATS NOT LINKED : ABNORMAL PROGRAM TERMINATION and tells you how to resolve it.

What are floating point formats?

Floating point formats are a collection of formatting information used to manipulate floating point numbers in certain runtime library functions such as `scanf()` and `atof()`.

Some compilers for small machines, including Turbo C (and Ritchie's original PDP-11 compiler), leave out certain floating point support if it looks like it will not be needed. The intent was to avoid linking the floating point formats (about 1K of overhead) when they are not required. In particular, the non-floating-point versions of `printf` and `scanf` saved space by not including code to handle `%e`, `%f`, and `%g`.

The tradeoff of this feature is that the programmer must explicitly request that the floating point formats to be linked in for some programs which manipulate floats in a limited and specific fashion.

It happens that Borland's heuristics for determining whether the program uses floating point are insufficient, and the programmer must sometimes insert a dummy call to a floating-point library function (such as `sqrt`; any will do) to force loading of floating-point support.

The problems and solutions below apply to ALL versions of Turbo C, Turbo C++, and Borland C++, except where noted.

"Floating point formats not linked" is a Borland run-time error (Borland C or C++, Turbo C or C++). Borland's compilers try to be smart and not link in the floating- point (f-p) library unless you need it. Alas, they all get the decision wrong. One common case is where you don't call any floating point functions, but you have `%f` or other floating point formats in `scanf()` or `printf()` calls. The cure is to call an floating point function, or at least force one to be present in the link.

To do that, define this function somewhere in a source file but don't call it:

```
static void forcefloat(float *p)
{
    float f = *p;
    forcefloat(&f);
}
```

or

```
static void force_fpf(void)
{
    float x, *y; /* Just declares two variables */
    y = &x; /* Forces linkage of FP formats */
    x = *y; /* Suppress warning message about x */
}
```

It doesn't have to be in the module with the main program, as long as it's in a module that will be

included in the link.

The Borland C++ 3.0, README file documents a slightly less ugly work-around. Insert these statements in your program:

```
extern unsigned _floatconvert;  
#pragma extref _floatconvert
```

also the following works:

```
extern void _floatconvert();  
#pragma extref _floatconvert
```

Both the above options using the pragma preprocessor command should be placed at the top of your program after the required library include calls.

You should also check the following possible reasons for this type of error.

1. The floating point option set to None when it should be set to either Emulation or 80x87.

FIX: Set Floating Point to Emulation or <80x87>. In the Integrated Development Environment (IDE), this is either

Options | Compiler | Advanced Code Generation (TurboC++)

or

Options | Compiler | Code Generation | More, (Borland C++)

2. Forgetting to put the address operator & on the scanf variable expression. For example,

```
float foo;  
scanf("%f", foo);
```

FIX: Change the code so that the & operator is used where it is needed. For example, the above code should be

```
float foo;  
scanf("%f", &foo);
```

3. A bug exists in Turbo C 2.0 when using scanf()

FIX: Obtain and apply the patches in TC2PAT.ARC. This file can be downloaded from the Languages / C++ / Patches section on DLBBS (408-439-9096).

4. A bug exists in Turbo C 2.01 when using atof() or strtod()

FIX: Obtain and apply the patches in TC21PT.ARC. This file can be downloaded from the Languages / C++ / Patches section on DLBBS (408-439-9096).