

New Technical Notes

Macintosh



Developer Support

MPW Tool Q&As

Platforms & Tools

Revised by: Developer Support Center

June 1993

Written by: Developer Support Center

October 1990

This Technical Note contains a collection of Q&As relating to a specific topic—questions you've sent the Developer Support Center (DSC) along with answers from the DSC engineers. While DSC engineers have checked the Q&A content for accuracy, the Q&A Technical Notes don't have the editing and organization of other Technical Notes. The Q&A function is to get new technical information and updates to you quickly, saving the polish for when the information migrates into reference manuals.

Q&As are now included with Technical Notes to make access to technical updates easier for you. If you have comments or suggestions about Q&A content or distribution, please let us know by sending an AppleLink to DEVFEEDBACK. Apple Partners may send technical questions about Q&A content to DEVSUPPORT for resolution.

New Q&As in this Technical Note:

Linking a stand-alone code resource that uses `sprintf`

Linking a stand-alone code resource that uses `sprintf`

Date Written: 1/21/93

Last reviewed: 4/1/93

I want to use `sprintf` in a stand-alone code resource, but I'm having trouble linking my resource because `sprintf` apparently requires data-to-code references. I get the following error:

```
### While reading file "HD:MPW:Libraries:Libraries:Runtime.o"
### Link: Error: Data to Code reference not supported (no Jump Table). (Error 59)
### Link: Errors prevented normal completion.
```

What can I do to avoid this?

Unfortunately, a great deal of the standard C library uses globals, which necessitates your creating an A5 world as described in the Macintosh Technical Note "Stand-Alone Code, Ad Nauseum." An additional misfortune is that some of the global data used is initialized with function pointers; for example, if you had the following global variable defined:

```
ProcPtr gMyProcedure = (ProcPtr)MyFunc;
```

where MyFunc is a function of yours, you can see that the global variable would have to be initialized with a pointer to the function before your code started executing. MPW's loader supports only pointing variables such as these at A5-relative references—that is, into the jump table. It cannot initialize them to the PC-relative reference that a stand-alone code segment requires. This means that you can't link any code resource that requires a data-to-code reference, thus the error.

Fortunately, there's a workaround. Because the standard library is written very generally and uses a lot of indirection, the linker believes `sprintf` might use some of the standard output calls; however, it never does, and since it's these calls that necessitate the use of the data-to-code references, if we can fool the linker into accepting an innocuous substitute, we won't require the data-to-code references that are causing us such agony now. To cut to the chase, add the lines:

```
size_t fwrite (const void *, size_t, size_t, FILE *) { return 0; }
flsbuf() {}      // these calls won't actually be called by sprintf.

fcvt() {}        // these calls are used only for floating point %f and %e.
ecvt() {}
```

to your code someplace. The first two override `fwrite` and `flsbuf`; these calls will now no longer cause the linker to believe you require the data-to-code references; because they're never called by `sprintf`, replacing them isn't a difficulty. The second two are only conveniences; they stub out the standard library's code for converting floating point arguments; if you never use `%f` or `%e` in your `sprintf` calls, then these will reduce the size of your compiled code. To get these overriding functions to mask the versions in the `StdCLib.o` library, you should make sure that the object file that contains these stubs appears on the Link command line before the `StdCLib.o` library.

Decompressing compressed System 7 file resources

Date Written: 1/22/91

Last reviewed: 8/1/92

I tried to DeRez some of the resources in my MacintoshSystem 7.0 system file, but for some of them all I got was garbage. Why?

Some of the resources shipped by Apple with System 7.0 are compressed. DeRez doesn't use the Resource Manager (mostly for historical reasons), so it doesn't benefit from the Resource Manager's automatic decompression of these resources.

If you wish to examine the contents of one of these compressed resources, use ResEdit version 2.1. It will warn you that the resource you're about to examine is compressed and that the resource must be decompressed to edit it (after which you can use DeRez to examine it in all its textual glory).

InitGraf is only managers' init routine called by MPW Tools

Date Written: 1/28/91

Last reviewed: 8/1/92

I have compiled and linked a multi-segmented MPW tool using the following switches in a Build/MAKE file:

```
C -sym full -s Main -r -t XXXXXX.C ...(etc.)
  (first segment named "Main"; others are arbitrarily named)
Link -c 'MPS ' -t MPST -map -rn -srt -sym full -x XXXXxref ... (etc.)
  (removed segment names)
```

I believe the above set of compile and link switches will, among other things, allow me to generate multi-segmented code without having to resort to #pragma segment directives in the source. Please correct me if I am wrong.

I have no difficulty building a multi-segmented product using THINK C 4.0. Although the MPW tool seems to build properly, when I run the tool, I get a bus error deep down in ROM after a call in my source to SFGetFile. I have used SADE, MacsBug, THINK C Debugger, and MacNosy/The Debugger and I just about give up! The information I've supplied, generated using Jasik's Debugger (V2), will help to give you some clues.

—

The call to SFGetFile:

```
ADEA Pack3 AutoPop; (selector:INTEGER)
selector:      2
SFGetFile(where:Point; prompt:Str255; fileFilter:ProcPtr;
numTypes:INTEGER;
typeList:SFTYPEList; dlgHook:ProcPtr; VAR reply:SfReply)
where          : 0 0
prompt         : "
fileFilter     :  NIL
numTypes       :      1
typeList       :Array[1:4] of ASCII
[ 0] = 'AHED'          <-- This is the only one that matters
[ 1] = $00 $14 $00 $00  <-- Elements 1,2,3: Junk?
[ 2] = $00 $0A $00 '4'
[ 3] = $C8 'ûfl'
```

There certainly doesn't appear to be anything wrong with your segmentation approach; it's very unlikely that it's the source of your problem.

What is more likely is that you're initializing more than you need to in your tool. MPW Tools must call InitGraf in order to provide a valid global environment for themselves, but they cannot call any of the other Managers' initialization functions. In particular, calling InitWindows will result in the behavior you are encountering when you call SFGetFile.

MPW 3.2 'HEXA' resources

Date Written: 6/11/91

Last reviewed: 8/1/92

What are the four 'HEXA' resources in the MPW 3.2 shell, and will changing any of them give me more room for linking?

—

Each 'HEXA' resource ID with a corresponding description follows. Twenty-three words of caution, however, are that these resources may change in the future, so use this information at your own risk.

128 : initial stack size
129 : initial number of master pointers
130 : initial pointer zone size (MPW allocates a block for random pointers)
131 : pointer zone delta (when the zone fills, MPW allocates this much more room).

So, if you want to make more memory available to your MPW tools (like the linker and compiler), you can try modifying some of these values, starting with the 'HEXA' resource #128. This process is described further in the MPW C++ release notes.

Balloon Help with MPW Tools

Date Written: 8/13/91

Last reviewed: 8/1/92

Is it possible to add Balloon Help to MPW 3.2 Tools?

After extensive head scratching, the answer is yes/no/no. There are basically three scenarios that might require Balloon Help: The first is if your tool takes it upon itself to have its own human interface with dialogs and windows of its own. This is not supported under MPW, but several people have succeeded in getting it to work with much work and programming. If you have gotten this to work, then Balloon Help is easy. It works just like an application; simply add the 'hdlg' resources to your resource fork (assuming you are using a dialog). This was verified by adding a proper 'hdlg' resource to the Choose tool; worked like a champ.

As for MPW supplied dialogs and alerts that may come up while your tool is running, there is no way to override the resources being loaded in many cases, since the dialogs are not created from a DLOG. In the case of MPW-supplied dialogs, Balloon Help is non-modifiable for you.

As for the only other dialog associated with your tool, the Commando dialog, you cannot add balloons to this either, but that is no real problem since Commando has its own built-in help system that allows you to associate help with any item in the commando dialog.

“-model far” and “PC-relative edit” link error

Date Written: 10/4/91

Last reviewed: 8/1/92

When using “-model far,” I get the following error from the linker:

```
### While reading file "HD:MPW:Libraries:Libraries:Runtime.o"
```

```
### Link: Error: PC-relative edit, offset out of range. (Error 48) %__MAIN  
(260) Reference to: main in file: xxx.c.o
```

What does it mean?

—

In your Link command, you should put Runtime.o and the object file containing your main program close together, and as early in the list as possible. This is because `%__MAIN` (the part of Runtime.o that actually receives control when your application is launched) uses a PC-relative BSR instruction to transfer control to your “real” main. (OK, so it’s “32-bit-almost-everything”!)

“-model far” affects jump table & code segment, not code module

Date Written: 10/4/91

Last reviewed: 8/1/92

I can’t define a procedure larger than 32K, even if I use -model far.

—

-Model farcode relaxes the 32K limit on jump table size, and eliminates the 32K limit on code segment size. It does *not* affect the limit on the size of an individual module: that remains at 32K. This is why you had trouble with your strings. [We don’t expect the module-size limitation to be relaxed anytime soon].

MPW linker problem with PC-relative function address production

Date Written: 10/29/91

Last reviewed: 8/1/92

When I use “-model far” in MPW 3.2, all my intra-segment function addresses are calculated from the actual code address, rather than as a jump table address. What can I do to work around this?

—

Thanks for your report of a problem with the MPW linker regarding production of PC-relative function addresses. This is a verified bug. Until it’s fixed, you can either (1) place your functions in segments other than the segments where the address is taken, to force A5-relative addressing, or (2) place all functions whose addresses might be taken into a segment which is never unloaded, so that an absolute address is acceptable.

MPW ScriptCheck 3.2.1 and file or resource atom size field

Date Written: 12/19/91

Last reviewed: 8/1/92

Is there a way to keep ScriptCheck 3.2.1 from overwriting the size field in file and resource atoms? I’m implementing a decompression action atom and need the Installer to check that there is sufficient hard disk space to decompress the compressed files.

—

This is a limitation of the current ScriptCheck version. Engineering is looking into revising the tool so that ScriptCheck will leave this field alone if desired. As things stand, after using ScriptCheck, one would need to use ResEdit to enter the correct values into the 'inpk', 'infa', and 'inra' resources. Note that the size field for the 'inpk' resource is for informational use

only, in the display window at the Custom Install menu.

MPW Linker “Data initialization code not called” error

Date Written: 5/3/89

Last reviewed: 8/1/92

The MPW linker is reporting a “Data initialization code not called” error. What is wrong?

This usually happens when trying to link an INIT or XCMD/XFCN code resource. Code resources are not allowed to use global variables or variables that are initialized at declaration time. First, make sure that you aren’t using any global or static variables. In Pascal and C, a global variable is one that is declared outside of any function or procedure. Also note that in MPW C, string constants (e.g. “hello, world”) are actually compiled as global variables (MPW C 3.0 has the -b option to override this). If you are using MPW C, check for string constants in your code.

To work around the string constant problem, read the string from a resource file:

```
char    *myStr;
...
myStr = "Hello world";
```

becomes:

```
str255  myStr;
...
GetIndString(&myStr, 256, 1);
c2pstr(myStr);
```

Variables that are initialized at declaration time in the source code are actually initialized when the program starts up. The compiler generates the initialization code and puts it in a segment called %A5INIT. Because XCMDs, for example, must be a single segment, you can’t link the %A5INIT segment as well. You need to change the initialization to an explicit assignment at the beginning of the code:

```
short  myCounter = 1;
```

becomes:

```
short myCounter;
...
myCounter = 1;
```

MPW 3.0 Linker and Vaccine

Date Written: 5/3/89

Last reviewed: 8/1/92

The MPW 3.0 Linker hangs or prints odd error messages when I try to link my program. What’s wrong?

This is usually a problem associated with Vaccine, the anti-virus 'INIT'. You will either get the following error messages, or the Linker will apparently hang:


```
### Link: Resource not found (OS error -192)
### Link: Error: Resource I/O error, # (Error 6) setting resource attributes.
### Link: Errors prevented normal completion.
```

In the former case, either remove Vaccine, or select the option that allows MPW to work (in the current version of Vaccine). In the latter case, the MPW option has already been set, and Vaccine is merely waiting for you to press “y” to signify that the link process should continue.

MPW “unable to swap in Shell segment” error

Date Written: 5/3/89

Last reviewed: 8/1/92

What causes an MPW tool to abort with the message, “Unable to swap in Shell segment”?

This message almost always means memory allocation problems. It is most often encountered when compiling and linking large programs, especially with the -sym option. It often goes away in subsequent executions of the tool or when you exit and re-launch the shell. Another symptom that MPW is on the edge is the dreaded “bulldozer” cursor. You may think MPW is hung in this case, but thrashing itself to death is a better description.

Try increasing the MultiFinder memory partition of the Shell and see if the problem persists. A better option under MPW 3.2 is to use the -mf option to enable the use of MultiFinder temporary memory, if the tool supports it.

MPW “unable to swap in tool segment” error

Date Written: 5/3/89

Last reviewed: 8/1/92

When linking my application I get the error “unable to swap in tool segment.” What has gone wrong?

The MPW linker has run out of memory. The first thing you can try is to turn off the RAM cache (with the Control Panel). If that doesn’t work, then take out your debugger. You will also want to remove any memory-hungry cdevs and 'INIT's. Another way to solve your problem is to segment your program using the Segment Loader (that is, make your big app into many little apps). If you still have problems, you need to get more memory.

New calls available as MPW 3.2 glue for System 6

Date Written: 6/24/91

Last reviewed: 8/1/92

How do I find out what calls in *Inside Macintosh* Volume VI are accessible from System 6?

As it turns out, there isn't a comprehensive list of calls that are available as MPW glue. I used the search command in MPW 3.2 to search for “`#ifdef SystemSevenOrLater`” to locate the full list:

System 7 Calls Glue

```
FindFolder()
Gestalt()
HOpenResFile()    [this has been available for some time] TN#214
HCreateResFile()  [this has been available for some time] TN#214
```

And also, if you have a sense of history,

System 6 Calls Glue

```
StripAddress()
SysEnviron()
```

In other words, FindFolder() works back to when HFS was introduced, Gestalt works back to when SysEnviron was introduced, and if SysEnviron is not present Environ is called.

MPW Rez and AnimCursors frame field color cursor bit

Date Written: 6/14/91

Last reviewed: 8/1/92

DTS's AnimCursors sample code checked the high bit of the second word in an 'acur' resource to see if it should check a series of 'crsr's rather than 'CURS's. To quote from AnimCursor.c, "If the high bit of the frame field is set, AnimCurs is a list of color cursors." The bit's not represented in MPW 3.2 resources, however. Was the frame field "feature" forgotten in MPW Rez, or has it gone away?

It looks like the folks who put together the Rez template didn't know about the bit. To get around this, copy the 'acur' template from Types.r into your own include file, and change the first "fill word;" line to

```
unsigned integer useCURS = 0, usecrsr = 0x8000; /* use 'CURS' or 'crsr'?
*/
```

This should get you on your way (albeit with one more compilation warning). The alternative is to define the resource as raw data, the way the sample does (of course, it has the excuse that it existed before types.r knew about it).

MPW Backup command problem report

Date Written: 3/18/91

Last reviewed: 8/1/92

When I use the MPW Backup command to generate a list of Duplicate commands, everything works properly with the form:

```
Set FileList 'Tweety 640:All Files'
Export FileList
Set newerFiles "Tweety 640:Newer Files"
Export newerFiles

Backup 0
```

```
-from 'Tweety 640:Empty Folder:' ␣
-to 'Tweety 640:Taz:SCSI 90:' ␣
-r -check newer -a ␣
> "{newerFiles}"
```

But when I use the form:

```
Backup ␣
  -from 'Tweety 640:Empty Folder:' ␣
  -to 'Tweety 640:Taz:SCSI 90:' ␣
  -a -r -since '01/01/04' ␣
  > "{FileList}"
```

The “to” or destination folders are not correctly generated in the output file. An example is shown below:

```
Duplicate      'Tweety 640:Empty Folder:AppleLink:MachApp.Tech$' 'Tweety
640:Taz:SCSI 90:MachApp.Tech$'   ### New File ###
Duplicate      'Tweety 640:Empty Folder:AppleLink:More on Daystar' 'Tweety
640:Taz:SCSI 90:More on Daystar'  ### New File ###
```

Note that the Duplicate commands being generated are copying everything into the “to” folder instead of recursively generating sub-folder names as it does in the first form of “Backup” (without a -since option). What am I doing wrong?

—

I tried to duplicate everything that you’ve done, and I came up with exactly the opposite results: Using the MPW 3.2 Backup command with the -check newer parameter, subdirectories are NOT generated recursively, but with the -since '01/01/04' parameter, the Backup command does generate subdirectories recursively! As I understand it, this is exactly the opposite of what happened for you! In either case, it looks like you’ve found a bug with the Backup tool.

MPW Linker (-pg) option

Date Written: 10/31/90

Last reviewed: 8/1/92

Link option (-pg) allows you to set the page size. The use is

```
link -pg 2048..
```

or some integer greater than 1024, which is the default size.

Use “Duplicate” to split file into data & resource fork files

Date Written: 12/14/90

Last reviewed: 8/1/92

Is there any simple way under MPW to split a Macintosh file into two new files—one containing the data fork and the other containing the resource fork?

—

The “Duplicate” tool will do this quite easily. Either use the Commando dialog, or this from the command line:

```
duplicate A B -d # copies A's data fork ONLY to B
duplicate A C -r # copies A's resource fork ONLY to C
```

Debugging MPW Tools with SADE

Date Written: 5/1/91

Last reviewed: 8/1/92

How do I debug MPW Tools written in Pascal using SADE 1.0?

Keep in mind the following:

- SADE 1.0, 1.1, and 1.2 are not compatible with Macintosh System 7.
- If you're using System 7, you'll need to use MPW 3.2 and SADE 1.3.
- You must be using MultiFinder 6.1 to use SADE with System 6.0.x.
- You need to compile and link your tool with the symbolic debugging flags on.
- SADE 1.0 has been upgraded to SADE 1.1 (so you should probably upgrade).

To start working with SADE on an MPW tool (in Pascal), compile and link the ResEqual tool in the Pascal examples folder using MPW. Here are the commands I used from MPW to do this:

<First, set the MPW directory to the PExamples folder>

```
set SymOptions "-sym on"
rez ResEqual.r -o ResEqual -append
Pascal ResEqual.p -o ResEqual.p.o {SymOptions}
Asm FStubs.a -o FStubs.a.o
Link -w -c 'MPS ' -t MPST ResEqual.p.o FStubs.a.o 0
    -sn STDIO=Main 0
    -sn INTENV=Main 0
    -sn %A5Init=Init 0
    "{Libraries}"Interface.o 0
    "{Libraries}"Runtime.o 0
    "{Libraries}"ToolLibs.o 0
    "{PLibraries}"Paslib.o 0
    -o ResEqual {SymOptions}
```

Once the ResEqual tool is compiled, get into SADE and type in the target command with the appropriate information. Mine looked like this:

```
target "HD:MPW 3.2:MPW Shell 3.2b12" using "HD:MPW 3.2:Examples:PExamples:
ResEqual.SYM"
```

You shouldn't have to use the SourcePath add command because all of the source code should be in the same folder as the ResEqual.SYM file.

Next, execute the MPW Shell by double clicking it from the Finder or by launching it from SADE (the launch command). The target command may launch it, depending on the version of SADE that you are using. Once you're in MPW, execute the tool (type in ResEqual xyz xyz). When the tool is executing, you can press the command-option-period (the period must be the one on the numeric key pad) and it will break out of the tool back into SADE. Alternatively,

you can set breakpoints in SADE before you run the tool from MPW. This will pop you into SADE without requiring you to press the break key (command-option-period).

MPW Linker >65K global size error message (Error 34)

Date Written: 4/2/92

Last reviewed: 8/1/92

We have been using the -srt option to great effect for several years in order to get around the 32K data limit problem. Recently, we have broken the 64K data barrier by pushing large constant arrays into the “beyond 32K” space. The linker is now giving a warning `### Link: Warning: Size of global data area exceeds 65K. (Error 34)` for every such data item! We’ve been ignoring these warnings, but is this really a problem?

Yes, when you compile with MPW C’s -m option, you can then safely ignore the message:

```
### Link: Warning: Size of global data area exceeds 65K. (Error 34)
```

To entirely avoid getting this error you’d have to use -model farData or -model far in place of -m in your compile. However, you would then be prohibited from using -br on in your link. For those links that you want to use -br on with, you may want to consider using a custom script that filters out these error 34 messages.

Linking a driver with MPW 3.2

Date Written: 5/6/92

Last reviewed: 8/1/92

When I try to link my driver in MPW 3.2, it tells me

```
### Link: Error : Output must go to exactly one segment when using  
"-rt" (Error 98)  
### Link: Errors prevented normal completion.
```

In all my source files I have `#pragma segment Main {C}` and `SEG 'Main' {Asm}` directives. Why is it doing this? What factors determine how segments are assigned (besides the `#pragma` stuff)? How can I get it to work?

The problem is probably that you’re including modules from the libraries that are marked for another segment. Usually the culprit here is that some of the routines in StdCLib or some other library are marked for the StdIO segment. You can generally fix this by using the -sg option to merge segments, either explicitly by naming all the segments you want to merge, or implicitly by just putting everything into one segment. You probably want to do the latter, because you only want one segment anyway. Thus, what you should do is add the option “-sg Main” to your link line in place of the “-sn Main=segment” option. This will merge all segments into the Main segment, making it possible to link.

MPW Linker error 114

Date Written: 6/15/92

Last reviewed: 10/11/92

Could you tell me where I can get some explanation on the following linker error?

```
### MPW:MPW:Tools:Link: Error: 16-bit reference offset out of range. (Error 114)
```

MPW's default is to generate 16-bit references for your program. This means that your program's code segments, jump table and global data is limited 32K. In recent versions of MPW these limitations have been overcome. What you should do is compile and link your code with the "-modal far" option. MPW will now generate load-time relocatable 32-bit references. Remember to add -modal far on each compile step, and the link step in your makefile. See the MPW documentation for more information on using the -modal far option with the compilers and linker. There's no additional documentation on MPW Linker errors, other than the Q&A Technical Notes on the *Developer CD Series* disc.