THINK pre-v7 and Other Compilers

This chapter provides instructions on how to build the required support libraries if you are using an earlier version of THINK. It also includes some general guidance for users of other compilers, but because I don't own any other compilers, I can only provide general guidance.

Use of Mac F2C code with the C++ compiler is strongly not recommended with version 6 of the THINK/Symantec C++ compiler. There is no C++ compiler prior to version 6. For these reasons, the C++ libraries and C++ model projects are omitted from the below discussions.

If you have THINK version 6, you can upgrade to version 7 and beyond (minus Visual Architect and TCL version 2.0) for free. Look in your favorite Macintosh archive site for the free updater.

Building the support libraries with THINK pre-v7

There are five support libraries required to run programs translated by Mac F2C. These libraries are:

(1) ANSI F2C
(2) unix F2C
(3) libI77a
(4) libI77b
(5) libF77

To make ANSI F2C and unix F2C: Duplicate the ANSI and unix libraries in the Standard Libraries folder (the ones provided by Symantec). Rename the duplicates ANSI F2C and unix F2C. Open each one and do the following:

• Remove objects

• In the Options dialog select:
  - 4-byte integers
  - 8-byte doubles
  - 68020 code generation (recommended, not required)

• In the Set Project Type dialog select:

- Far code
- Far data

• Bring Up To Date

To make libI77a, libI77b, and libF77: In the Mac F2C Libraries folder you will find these three libraries and two folders of source code (libI77 Sources and libF77 Sources). The process is the same for all three libraries. I will illustrate it with libI77a:

• Delete libI77a supplied with Mac F2C

• Create a new empty project called libI77a

• In the Options dialog select:
- 4-byte integers
- 8-byte doubles
- 68020 code generation (recommended, not required)
- remove the prefix #include <MacHeaders>
- add the prefix #include TPM_I.h
  (for libI77a and libI77b only)
- add the prefix #include TPM_F.h
  (for libF77 only)

• In the Set Project Type dialog select:
- Far code
- Far data

• Add source files from the folder libI77 Sources as required from the corresponding list below.

• Bring the project up to date

Repeat this process with libI77b and libF77. libF77 contains over one hundred files. You may find it easier to include all the source files in the folder libF77 Sources initially and then remove the handful that don't belong in libF77. Then copy the three built project files into the folder containing the THINK application (you can put them anywhere within the THINK tree).

Contents of libI77a

libI77a
1 segments
15 files

Segment 2
Name: libI77 Part I
Preload: false, Protected: true, Locked: true
Purgeable: true, SystemHeap: false
15 files:

access.c	backspace.c	close.c	dfe.c
dolio.c	due.c	endfile.c	err.c
fmt.c	fmtlib.c	iio.c	ilnw.c
inquire.c	lread.c	lwrite.c	DoMultiTask.c

Contents of libI77b

libI77b
1 segments
18 files

Segment 3
Name: libI77 Part II
Preload: false, Protected: true, Locked: true
Purgeable: true, SystemHeap: false
18 files:

open.c	rdfmt.c	rewind.c	rsfe.c
rsli.c	rsne.c	sfe.c	sue.c
typesize.c	uio.c	util.c	Version.c
wref.c	wrtfmt.c	wsfe.c	wsle.c
wsne.c	xwsne.c		

Contents of libF77

libF77
1 segments
115 files

Segment 2
Name: libF77
Preload: false, Protected: true, Locked: true
Purgeable: true, SystemHeap: false
115 files:

abort_.c	cabs.c	c_abs.c	c_cos.c
c_div.c	c_exp.c	c_log.c	c_sin.c
c_sqrt.c	derfc_.c	derf_.c	d_abs.c
d_acos.c	d_asin.c	d_atan.c	d_atn2.c
d_cnjg.c	d_cos.c	d_cosh.c	d_dim.c
d_exp.c	d_imag.c	d_int.c	d_lg10.c
d_log.c	d_mod.c	d_nint.c	d_prod.c
d_sign.c	d_sin.c	d_sinh.c	d_sqrt.c

d_tan.c	d_tanh.c	ef1asc_.c	ef1cmc_.c
erfc_.c	erf_.c	erf_fctns.c	getarg_.c
getenv_.c	getpid.c	hl_ge.c	hl_gt.c
hl_le.c	hl_lt.c	h_abs.c	h_dim.c
h_dnnt.c	h_indx.c	h_len.c	h_mod.c
h_nint.c	h_sign.c	iargc_.c	i_abs.c
i_dim.c	i_dnnt.c	i_indx.c	i_len.c
i_mod.c	i_nint.c	i_sign.c	l_ge.c
l_gt.c	l_le.c	l_lt.c	pause.c
pow_ci.c	pow_dd.c	pow_di.c	pow_hh.c
pow_ii.c	pow_ri.c	pow_zi.c	pow_zz.c
r_abs.c	r_acos.c	r_asin.c	r_atan.c
r_atn2.c	r_cnjg.c	r_cos.c	r_cosh.c
r_dim.c	r_exp.c	r_imag.c	r_int.c
r_lg10.c	r_log.c	r_mod.c	r_nint.c
r_sign.c	r_sin.c	r_sinh.c	r_sqrt.c
r_tan.c	r_tanh.c	signal_.c	sig_die.c
system_.c	s_cat.c	s_cmp.c	s_copy.c
s_paus.c	s_rnge.c	s_stop.c	Version.c
z_abs.c	z_cos.c	z_div.c	z_exp.c
z_log.c	z_sin.c	z_sqrt.c	

Converting the Test Project to THINK pre-v7

The folder Test Project ƒ contains a project file called Test.π. To create a version compatible with your version of THINK C, do the following:

• Delete Test.π

• Create a new empty project called Test.π

• In the Options dialog select:
  - 4-byte integers
  - 8-byte doubles
  - 68020 code generation (recommended, not required)
  - remove the prefix #include <MacHeaders>

• In the Set Project Type dialog select:
  - Far code
  - Far data

• Add sources and libraries as required to match the list that appears below (you will not be able to add test.c until you create it by translating test.f with Mac F2C). Segment the project as indicated in the list below (in some versions of THINK C you cannot name the segments; this is not a problem, just ignore the

segment names).

Contents of Test.π

Test.π
6 segments
7 files

Segment 2
Name: Your Code Here
Preload: false, Protected: true, Locked: true
Purgeable: true, SystemHeap: false
2 files:
F2Cmain.c test.c (you cannot add test.c until you create it)

Segment 3
Name: libF77
Preload: false, Protected: false, Locked: false
Purgeable: false, SystemHeap: false
1 files:
Mac F2C Libraries:libF77

Segment 4
Name: libI77 Part A
Preload: false, Protected: true, Locked: true
Purgeable: true, SystemHeap: false
1 files:
Mac F2C Libraries:libI77a

Segment 5
Name: libI77 Part B
Preload: false, Protected: true, Locked: true
Purgeable: true, SystemHeap: false
1 files:
Mac F2C Libraries:libI77b

Segment 6
Name: ANSI for F2C
Preload: false, Protected: true, Locked: true
Purgeable: true, SystemHeap: false
1 files:
Standard Libraries:ANSI F2C

Segment 7
Name: UNIX for F2C
Preload: false, Protected: true, Locked: true
Purgeable: true, SystemHeap: false
1 files:
Standard Libraries:unix F2C

Using the Project Model with THINK pre-v7

The project model is the folder Mac F2C Project which is located in the folder For '(Project Models)'. The folder Mac F2C Project contains the following files:

• @1.π -- a THINK C version 7.0 project file.

• main.c -- a driver program required to run programs compiled by Mac F2C.

• f2c.h -- a header file required by code compiled by Mac F2C.

This what you need to compile, link, and run files produced by Mac F2C. In THINK C version 7.0 or later, the THINK project manager automatically makes a copy of the entire model project (i.e., the entire Mac F2C Project folder) and renames the folder and project file when you create a new project using this model. To use the project model with versions of THINK C prior to 7.0, you will need to:

• replace @1.π which a project compatible with your version of THINK.

• every time you want to create a new project for Mac F2C code, duplicate the project model folder by hand and rename the folder and project file by hand.

To create a version of @1.π compatible with your version of THINK C, do the following:

• Delete @1.π

• Create a new empty project called @1.π

• In the Options dialog select:
    - 4-byte integers
    - 8-byte doubles
    - 68020 code generation (recommended, not required)
    - remove the prefix #include <MacHeaders>

• In the Set Project Type dialog select:
    - Far code
    - Far data

• Add sources and libraries as required to match the list that appears below.

Segment the project as indicated (in some versions of THINK C you cannot name the segments; this is not a problem).

Contents of @1.π

@1.π
7 segments
9 files

Segment 2
Name: Your Code Here
Preload: false, Protected: true, Locked: true
Purgeable: true, SystemHeap: false
1 files:
F2Cmain.c <your code goes here or in new segments>

Segment 3
Name: libF77
Preload: false, Protected: false, Locked: false
Purgeable: false, SystemHeap: false
1 files:
Mac F2C Libraries:libF77

Segment 4
Name: libI77 Part A
Preload: false, Protected: true, Locked: true
Purgeable: true, SystemHeap: false
1 files:
Mac F2C Libraries:libI77a

Segment 5
Name: libI77 Part B
Preload: false, Protected: true, Locked: true
Purgeable: true, SystemHeap: false
1 files:
Mac F2C Libraries:libI77b

Segment 6
Name: ANSI for F2C
Preload: false, Protected: true, Locked: true
Purgeable: true, SystemHeap: false
1 files:
Standard Libraries:ANSI F2C

Segment 7
Name: UNIX for F2C
Preload: false, Protected: true, Locked: true
Purgeable: true, SystemHeap: false
1 files:
Standard Libraries:unix F2C

Segment 8
Name: Mac Stuff
Preload: false, Protected: true, Locked: true
Purgeable: true, SystemHeap: false
2 files:
MacTraps MacTraps2

Using Mac F2C with Other Compilers

If you use a compiler other than THINK C/C++, Symantec C/C++, or CodeWarrior C/C++ you will have to make libraries on your own. You will almost certainly have to make some minor modifications to some of the source files in the support libraries.

You can use the contents listing that appear above to figure out what to put in each library. If your compiler/linker allows libraries larger than 32K, you can combine libI77a and libI77b into a single library. In that case you can also combine all three (libI77a, libI77b, and libF77) into a single library if you so desire.

Make sure to compile the libraries so that integers are 4-bytes, doubles are 8-bytes, and you can have more than 32K of static/global data and jump tables larger than 32K. See the sections on "Using Code Generated by Mac F2C" in the THINK v7 and CodeWarrior chapters for more information on these requirements and when they can be relaxed.

You must also #define the following things when compiling these files:

For libF77
```
<None>
```

For libI77a and libI77b
```
#define NON_UNIX_STDIO        // force the use of ANSI standard I/O
#define _POSIX_SOURCE         // force the use of mktemp() functions
```

Most of the files in the Mac F2C support libraries compile without trouble on Macintoshes. Nevertheless, I had to modify a few files, and in doing so I sometimes had to rely on features specific to the THINK and CodeWarrior compilers. The files in question are:

In libI77:

• access.c This function returns 0 if file called fileName exists, 1 otherwise. I wrote this from scratch using a Macintosh Toolbox call, so it should work with any

Macintosh compiler.

In libF77:

• erf_fctns.c THINK C does not provide error functions in their library. I wrote my own. They have decent numeric properties and will compile with any ANSI C compiler. You may wish to replace this file with (perhaps faster) vendor provided error functions.

• getenv_.c I had to write my own. I wrote one that returns a blank environment string. Modify as appropriate for your compiler.

• getpid.c I had to write my own. I exploited a global variable that is defined and maintained by the THINK C unix library. CodeWarrior doesn't define these, so in this case I just defined by own. You will have to modify this file to work with your system. If all else fails, return 0.

Code produced by Mac F2C also relies on the THINK C ANSI and unix libraries. The THINK C ANSI library is exactly what it says it is. Your compiler should include such a library. The THINK C unix library provides a large collection of functions commonly available on unix systems. Your compiler may or may not include such a library. If it doesn't, you will get link errors when compiling programs produced by Mac F2C. You will have to provide the missing functions yourself. It's actually not hard to do, especially because in many cases you only need a place holder function (e.g., see getenv_.c or getpid.c). When a placeholder isn't enough, you can figure what the function needs to do by checking a unix manual or using the man command on a unix system.

In any case make sure your versions of the ANSI and unix libraries are compiled with the proper options (4-byte integers, 8-byte doubles, jump tables larger than 32K, global data larger than 32K, and any C++ compatibility options if you plan to ever produce C++ code as Mac F2C output).

Finally, to run a program translated by Mac F2C, you need to link the following things together (this is automatically set up for THINK C users by the model project):

- your translated code
- main.c
- libI77a
- libI77b

- libF77
- ANSI libraries
- unix libraries

The file main.c is a driver that sets up a bunch of things and then calls the translated version of the FORTRAN main program. I modified main.c to use the THINK C console interface. In particular, main.c calls the function ccommand() to get command-line arguments from the user. You will need to modify this accordingly for your compiler.

Good luck. If you have any questions, feel free to email me at igormt@alumni.caltech.edu. If you make a stable, general purpose port of the libraries and/or F2Cmain.c to another Macintosh compiler, send me a copy and I will include it in future distributions of Mac F2C (with full credit and much gratitude).