

Standard Products

# UT69R000 C Cross-Compiler

User Manual (XGCC Version 4.0)

May 2005



## Conventions

The following conventions are used in this document:

TABLE 1. Conventions Used in This Document

Convention	Definition
<i>italics</i>	Titles of documents referred to in this manual appear in italics.
<variables>	Information that can vary in a command, such as a file name, appears in lowercase letters or words enclosed in angle brackets.
<KEYS>	Keys, such as the <C> or <ENTER> key on the keyboard, are indicated by uppercase letters or words enclosed in angle brackets. Press the indicated key, do <i>not</i> type the letters or words.
system information	system-related information, including prompts, file names, and error messages, is indicated by a <code>courier</code> font.
%	The Unix system prompt, as used in this document. Do not type this prompt.
<b>input</b>	Information that you must enter exactly as it appears in this document is indicated by a <b>bold</b> font.

## Introduction

This document describes a C cross-compiler for the Aeroflex Colorado Springs UT69R000 microcontroller. Cross-compiler features include:

- Extensive code optimization routines
- Detailed warning and error diagnostics
- A C pre-processor
- C++ compiler “front end”
- A floating point math library

The source code for GNU Version 2.7.2 of the C compiler is provided with this release. The source code for the compiler is provided by the Free Software Foundation and . This software is distributed under a special license agreement called Copyleft, which states:

- Companies may not distribute the software without its source code.
- Although companies are allowed to sell free software, they are not allowed to prevent their customers from making copies and giving it away.

The cross-compiler has been successfully built for the following Unix host systems:

- sparc-sun-sunos4.1.3
- sparc-sun-sunos5.3

To build the compiler, you must know the following:

- Processor type (Sparc)
- Vendor (Sun)
- Operating System Version (Sun OS4.1.3)
- Tape Format (8 mm, 4 mm, etc.)

## Supported Platforms

- The C cross-compiler software is available in the following tape formats:
  - 1/4-inch cartridge
  - 4 mm cassette
  - 8 mm cassette
- The C cross-compiler runs on the following hosts: *1750a, a29000, alpha, arm, clipper, convex, dsp16xx, elxsi, fx80, gmicro, h8300, i370, i386, i860, i960, m68000, m88000, mips, ns32000, pa, pdp-11, pyramid, romp, rs6000, sparc, spur, vax, we32000*. In each case, the host creates target code for the UT69R000.
- The cross-assembler (RASM) and the cross-linker (RLNK) have also been ported to the Unix operating system (SunOS4.X and SunOS5.X). When properly installed, the compiler driver programs (gcc, g++) automatically invoke the cross-assembler and cross-linker.
- The distribution tape contains all the GNU source files and the following files:

**TABLE 2. Required Aeroflex Files**

The File...	is...	which...
<code>local.h</code>	a C header file	contains constants and macros describing the architectural information of the UT69R000 microcontroller.
<code>ut69r.c</code>	a C program file	contains C functions which format assembler output.
<code>ut69r.md</code>	an Emacs-LISP machine description	contains rules for pattern-matching intermediate code (Emacs-LISP), and generates UT69R000 assembler code.

The files shown in the table are copied to the `gcc-2.7.2/config/ut69r` subdirectory when all the source files are extracted from the tape.

## Installation

The installation process involves:

- Verifying that you have access to Unix operating system utilities
- Making sure you have write permission to certain directories
- Unloading tape files
- Installing the software

## Required Operating System Utilities

The installation procedure makes extensive use of several Unix operating system utilities, including:

- The *make* utility, which provides inter-file dependencies and commands for compiling and linking the compiler, assembler, and linkage-editor.

If the *make* utility is not provided with your Unix operating system, GNUmake is an acceptable substitute.

## Pre-Installation Requirements

To correctly install the finished compiler, be sure that the following directories exist, and that you have write permission to them:

- `/usr/local/bin`

- /usr/local/lib
- /usr/local/man

Make sure that the directory /usr/local/bin is in your search path. If you want to specify a different target location than the default of /usr/local, see step three of the section “Installing the Compiler”.

## Installing the Compiler

The following installation instructions assume that you are using the C-shell on a Unix operating system.

- Determine the device-file name for the tape device for your system. In the example below, /dev/rst8 is the file name.
- To extract the tar file from the distribution tape, type:

```
% tar xvf /dev/rst8
```

Table 3, “Subdirectories and What They Contain” shows the contents of the distribution media.

**TABLE 3. Subdirectories and What They Contain**

The Subdirectory...	Contains...
RASM	the UT69R000 assembler source code
RLNK	the UT69R000 linkage-editor source code
gcc-2.7.2	the source code for the C compiler
UTIL	additional assembler and C libraries
IRSIM	the UT69R000 simulator software

Sun Solaris users must have a native C compiler in addition to the C cross-compiler.

1. Before building the compiler, change to the gcc-2.7.2 subdirectory.
2. Remove files from previous compilations by typing:

```
% make distclean
```

3. Follow these steps to compile the cross-compiler:

- Configure the makefile for your host system, providing the processor, vendor, and operating system for the host. For example, to configure the makefile for a Sun SPARC running SunOS4.1.3, type:

```
% configure --host=sparc-sun-sunos4.1.3 --target=ut69r-local
```

This installs the software into /usr/local. If you want to install the compiler in a location other than /usr/local, type:

```
% configure --host=sparc-sun-sunos4.1.3 --target=ut69r-local \
--prefix=<different directory>
```

- i. To compile a C, C++, and objective C compilers, type:

```
% make
```

- ii. To compile *only* a C compiler, type:

```
% make LANGUAGES=c
```

Ignore this error message:

```
You must find a way to make libgcc1.a
***Error Code 1
```

4. Follow these steps to install the compiler. Perform all work in the `gcc-2.7.2` subdirectory.
  - a. To install *all* compilers, type:
    - i. `% make install`  
at the system prompt.
    - ii. To install the C compiler only, type these commands in the order shown:  
`% make install-common LANGUAGES=c`  
`% make install-driver LANGUAGES=c`
  - c. To install the compiler manual pages, type:  
`% make install-man`
  - d. To view the manual pages, type:  
`% man gcc`  
at the command line.

Note that the directory `/usr/local/man` must be added to your `MANPATH` environment variable.

5. Install the cross-assembler and cross-linker by typing:

```
make; make install
```

in the `RASM` and `RLNK` subdirectories.

If you use the `--prefix` option when compiling the cross-compiler, make sure that the `$PREFIX` directory matches `$PREFIX` in the `rasm` and `rlnk` makefiles.

6. Build the library utilities by typing **make** in the `-util` subdirectory.
7. Install the library utilities by typing **make install** in the `-util` subdirectory.
8. It is recommended that you create an UNIX alias for the compiler:

```
% alias xgcc '<prefix path>/bin/ut69r-local-gcc -fno-builtin'
```

This directs the cross-compiler to search the supplied utilities library for functions which are normally “builtin”

## Characteristics of the Run-time Environment

### Run-time Stack Characteristics

- A run-time stack is used for subroutine linkage.
- The run-time stack will start at address `FFFF(hex)` of the data space and grow downward.
- The user must ensure that the application program does not *underflow* the stack area.
- The run-time stack is organized into activation records or frames.
  - Data items within the stack (function arguments and local variables) are referenced by a *frame pointer* (`r14`).

- A *stack pointer* (r15) is used to push and pop function arguments.
- The frame pointer is an “anchor” within a single activation record. Frames or activation records have the following organization:
  - Function arguments are at negative offsets from the “caller” frame pointer.
  - Function arguments are at positive offsets from the “callee” frame pointer.
  - The links to the previous activation records are at positive offsets from the “callee” frame pointer.
  - Local variables are at negative offsets from the frame pointer.

## Float Utilities

Variables of the type *float* and *double* are represented as a 64 bit quantity consisting of a 48 bit 2’s complement mantissa and a 16 bit 2’s complement exponent. Floating point numbers are represented as a fractional mantissa times 2 raised to the power of the exponent.

The following utilities are provided for argument types *float* or *double*. The results are normalized unless otherwise indicated.

<code>fsqrt (x)</code>	floating point square root of x
<code>flog2 (x)</code>	log base 2 of x
<code>floor (x)</code>	floating point floor function of x
<code>fpow (x,y)</code>	real base x to real power y
<code>acos (x)</code>	arc-cosine of x (radians)
<code>asin (x)</code>	arc-sine of x (radians)
<code>atan (x)</code>	arc-tangent of x (radians)
<code>cos (x)</code>	cosine of x (radians)
<code>exp (x)</code>	raise e to the x power
<code>log (x)</code>	base e logarithm of x
<code>log10 (x)</code>	base 10 logarithm of x
<code>pow (x,y)</code>	raise x to the y power
<code>sin (x)</code>	sine of x (radians)
<code>sqrt (x)</code>	square root of x
<code>tan (x)</code>	tangent of x (radians)

## Known Errors

The following bugs have been identified at this time:

- If you are using the SunOS4.1.X `make` utility, include:

```
COMPILE.c= $(CC) $(CFLAGS) $(CPPFLAGS) -c
```

```
LINK.c= $(CC) $(CFLAGS) $(CPPFLAGS)
```

in your makefile.

- Use the compiler option `-fno-builtin` to avoid using built-in gcc utilities.

The built-in gcc utilities are:

- `_exit`
- `abort`

- abs
- alloca
- cos
- exit
- tabs
- labs
- memcmp
- memcpy
- sin
- sort
- strcmp
- strcpy
- strlen

## Known Limitations

The following list and examples show current limitations of the compiler.

1. Do not use the `long long` integer constant.

C code example:

```
int long long value;
```

2. Do not optimize code with the `-O2` option of the compiler.

Optimization can corrupt the linking process, which may cause an `unresolved extrn symbol error`. If the compiler fails during the linking phase of the compile operation, and you are compiling with optimization, you may need to reduce the level of optimization. For more information, see the manual page provided with the GCC software.

3. Do not create `switch` statements with more than 24 `case` labels.
4. Do not initialize global strings. The following example shows code that will *not* assemble as written.

*Non-working C program example:*

```
char *s = "test str.";
main()
{
printf("%s\n", s);
}
```

XGCC failure messages:

```
RISC Assembler v4.0 (c1988) ,
All rights reserved.
pass 1 . . .
LINE 9 :          dw          LC0
          ^
ERROR : invalid syntax for data statement
ERROR : symbol s in module text_0 refers to empty label
Number of ERRORS   : 2
```

You can rewrite this code in two ways:

- a. Rewrite the `char` statement like this: `char s[] = "test str.";`
  - b. Change the `char` statement to: `char *s;` in the body of code, and initialize the string later. Pass the string as a variable into functions and/or procedures: `strcpy ( s, "test str." );`
5. Do not take an address of a label. The following example shows code that will *not* assemble as written.

*Non-working C program example:*

```
f()
{
    static void t[]={&&x};
x:;
}
```

XGCC failure messages:

```
RISC Assembler v4.0 (c1988) ,
All rights reserved.
pass 1 . . .
LINE 4 :          dw          L2
                ^
ERROR : invalid syntax for data statement
Number of ERRORS   : 1
```

## Frequently Asked Questions

1. Question: When I build the cross-compiler, it concludes with an error message: "You must find a way to make `libgcc.a.`" What does this mean?  
Answer: Ignore this message. GCC expects an archive library of functions for type conversions, for instance, `float` to `int.` 's cross-linker cannot currently read archive libraries. Aeroflex substitutes `libgcc1.0` instead.
2. Question: How do I build the cross-compiler if I am a Solaris user?  
Answer: You must have already installed a *native mode* C compiler before you can build the Aeroflex cross-compiler.
3. Question: Some of the ANSI-C functions are unresolved after

**COLORADO**

Toll Free: 800-645-8862  
 Fax: 719-594-8468

**INTERNATIONAL**

Tel: 805-778-9229  
 Fax: 805-778-1980

**NORTHEAST**

Tel: 603-888-3975  
 Fax: 603-888-4585

**SE AND MID-ATLANTIC**

Tel: 321-951-4164  
 Fax: 321-951-4254

**WEST COAST**

Tel: 949-362-2260  
 Fax: 949-362-2266

**CENTRAL**

Tel: 719-594-8017  
 Fax: 719-594-8468

[www.aeroflex.com](http://www.aeroflex.com)    [info-ams@aeroflex.com](mailto:info-ams@aeroflex.com)

Aeroflex Colorado Springs, Inc., reserves the right to make changes to any products and services herein at any time without notice. Consult Aeroflex or an authorized sales representative to verify that the information in this data sheet is current before using this product. Aeroflex does not assume any responsibility or liability arising out of the application or use of any product or service described herein, except as expressly agreed to in writing by Aeroflex; nor does the purchase, lease, or use of a product or service from Aeroflex convey a license under any patent rights, copyrights, trademark rights, or any other of the intellectual rights of Aeroflex or of third parties.



Our passion for performance is defined by three attributes represented by these three icons: solution-minded, performance-driven and customer-focused