Product Release Notes

CTOOLS Release 5.1

These product release notes are divided into the following sections:

- Product Checklist
- Product Enhancements and New Features
- Finding Updates to the Release Notes
- i960 RD Processor Support
- Compatibility Note
- Summary of Changes and Known Problems for Each Component (in the order shown on page 5)
- Manual Update

Product Checklist

Item	Description
1.	Maintenance and Support Information
2.	Product Release Notes
3.	*Getting Started with the i960® Processor Development Tools*
4.	*i960® Processor Compiler User's Guide*
5.	*i960® Processor Assembler User's Guide*
6.	*i960® Processor Software Utilities User's Guide*
7.	*i960® Processor Library Supplement*
8.	*gdb960 User's Manual*
9.	*i960® Processor Tools License Guide*
10.	Tape or CD-ROM containing CTOOLS

Product Enhancements and New Features

Release 5.1 of the development tools modifies the support for the i960 RP processor, and improves the program development process in several ways.

- Code generation for the i960 RP processor has been modified for compatibility with proposed future variations on the architecture. New libraries have been added specifically for the i960 RP processor.
- A new graphical user interface to the debugger allows source-level debugging with point and click ease.
- Windows* NT* is officially supported. The same copies of the tools run under Windows NT 3.51, Windows NT 4.0, and Windows 95. Though the Windows versions of the tools continue to run as batch mode programs, operation under plain MS-DOS* is no longer supported.

Finding Updates to the Release Notes

Updates to this document can be found on Intel's World-Wide Web site at:

```
http://developer.intel.com/design/i960/devtools/relnotes/
```

Use an HTML browser such as Microsoft* Internet Explorer* or Netscape* Navigator* to view the documents in this area.

i960 RD Processor Support

CTOOLS now supports the new i960 RD processor. To generate code for this new i960 processor family member, use the existing `-ARP` switch. Future releases of the tools will include a separate `-ARD` switch.

Compatibility Note

Code generated by Release 5.1 is fully compatible with Release 5.0. Source programs compiled with Release 5.0 are accepted by Release 5.1 without change. Almost all environment variables and invocation options are unchanged. Object modules generated with Release 5.0 can be linked with objects created with Release 5.1. However, object modules compiled with Release 5.0 for the i960 RP processor should be recompiled with Release 5.1 in order to generate objects that are forward compatible with future i960 RP processors.

Summary of Changes and Known Problems for Each Component

Installation Changes from Release 5.0

Install Renamed winstall

The CTOOLS installation program formerly named `install` has been renamed `winstall`.

Install Deselect Option Doesn't Work

In certain windows during the installation process, you see the option to deselect items. In this release of CTOOLS, the deselect option does not work.

Compatible-Mode Library Names

GNU/960 users upgrading from versions older than 5.0 will notice that the runtime libraries and linker directive files in this release have been renamed using a standard scheme described in the library manual.

Both Windows 95 and Windows NT Products Included

Release 5.1 contains the CTOOLS products for Windows 95 and Windows NT hosts.

Changes to Tools from Release 5.0

Archiver (arc960 / gar960)

The archiver no longer strips the symbol table information from an archive file when an element is deleted.

Assembler (asm960 / gas960 / gas960c / gas960e)

Changes from Release 5.0

i960 RP Architecture Specification

The implementation of the `-ARP` architecture option has been redefined in CTOOLS to represent a subset of the 80960 Jx instruction set chosen for performance and future compatibility reasons. These restrictions are enforced by the assembler and other tools when the `-ARP` switch is used or when the i960 RP architecture is specified using the `I960ARCH` or `G960ARCH` environment variables.

The following i960 Jx processor instructions are not supported with the i960 RP architecture:

addi	halt	remo
addi<cc>	intctl	shli
atadd	ldt	shrdi
atmod	mark	spanbit
cmpdeci	modac	stib
cmpdeco	modi	stis
cmpinci	modify	stt
cmpinco	modtc	subi
concmpi	movl	subi<cc>
concmpo	movq	sysctl
eshro	movt	test<cc>
extract	notor	xnor
fault<cc>	remi	

In addition, the following addressing mode restrictions exist for MEM format instructions when specifying an i960 RP processor-based target:

- Indexed addressing modes are not available.
- IP-relative addressing is not available.

- Two-word MEM-format is not available for the following instructions:

  `ldl`

  `stl`

  `ldq`

  `stq`

  `bx`

  `callx`

- The `balx` instruction may only use register-indirect addressing (no offsets or displacements allowed).

Other consequences of using the 80960RP output architecture are:

- The `calls` instruction may use register g13 or a literal as its target only.
- For the `modpc` instruction, the mask cannot specify the same register as the src/dst register.
- The `calljx` pseudo-instruction requires a second argument, a temporary register into which the address of the first argument can be loaded. See “Linker (gld960 / lnk960)” for information on the use of `calljx` with the i960 RP architecture.

No Big-Endian Support for i960 RP Architecture

Big endian byte order is not supported when code is being generated for the i960 RP processor.

No b.out OMF Support for i960 RP Architecture

The b.out object module format is not supported when code is being generated for the i960 RP processor.

New CORE Architecture Options

The assembler now supports new architecture settings to allow the generation of code that is compatible with multiple i960 processor types. These settings are referred to as *core* architectures. The table below shows the types of i960 processors that are supported by each core architecture.

-A Switch Used	Compatible Architectures
CORE0	Jx, Hx, RP
CORE1	Kx, Sx, Cx, Jx, Hx
CORE2	Jx, Hx
CORE3	Cx, Jx, Hx

Environment Variables

The assembler now supports all I960 and G960 environment variables, preferring those that match the invocation style. For example, when you invoke the assembler as asm960, the assembler looks first for I960 environment variables, and for those settings not found, looks for G960 environment variables. The environment variables used by the assembler are listed in the table below.

gnu Tools Name	CTOOLS Name	Purpose
G960ARCH	I960ARCH	Specifies target architecture.
G960IDENT	I960IDENT	Allows use of the COFF .ident directive.
G960INC	I960INC	Specifies include directory path.
G960BASE	I960BASE	Specifies base environment directory.

Branch Prediction Bits Ignored for i960 RP Architecture

The assembler no longer lets you set branch-prediction bits on the following instructions:

```
b
call
ret
bal
```

Decimal Instructions

The assembler no longer accepts decimal instructions when it is assembling for a KA or SA target, since decimal instructions are not supported by those processors. The instructions that are no longer supported are:

```
daddc

dsubc

dmovt
```

Using the modpc Instruction with the i960 RP Architecture

The syntax for using the `modpc` instruction with any i960 architecture other than RP is:

```
modpc src, mask, src/dst
```

When using a `modpc` instruction with the i960 RP architecture, the first and third arguments must be the same.

Compiler (cc1) and Drivers (gcc960, ic960)

This section describes changes and known problems for the compiler drivers gcc960 and ic960, and for the cc1 compiler.

Changes from Release 5.0

-ARP Switch Generates Compatible Code

The `-ARP` architecture option causes the generated code to be compatible with current and proposed future variations on the i960 RP architecture.

Common Architecture Code Generation Switches Added

The options `-mcore0`, `-mcore1`, `-mcore2`, and `-mcore3` for gcc960 and `-Gcore0`, `-Gcore1`, `-Gcore2`, and `-Gcore3` for ic960 let you generate code that compatible with multiple i960 processor types. Additionally, when you use `-mcoreX` or `-GcoreX`, you can include another `-A` switch to generate code that is optimized for a particular architecture, but still compatible with a group of architectures. The table below lists the architectures that are supported by the `-mcoreX` or `-GcoreX switches` and the `-A` options that you can use with them.

Option Name	Compatible Architectures	Can Be Used with:
`Mcore0, Gcore0`	Jx, Hx, RP	`-AJA`, `-AJD`, `-AJF`, `-AHA`, `-AHD`, `-AHT` or `-ARP`
`Mcore1, Gcore1`	Kx, Sx, Cx, Jx, Hx	Any architecture option except `-ARP`
`Mcore2, Gcore2`	Jx, Hx	`-AJA`, `-AJD`, `-AJF`, `-AHA`, `-AHD`, or `-AHT`
`Mcore3, Gcore3`	Cx, Jx, Hx	`-ACA`, `-ACF`, `-AJA`, `-AJD`, `-AJF`, `-AHA`, `-AHD`, or `-AHT`

No b.out OMF Support for i960 RP Architecture

The b.out object module format is not supported when code is being generated for the i960 RP processor.

Cache and Timer Control Header Files Not Usable with i960 RP Architecture

The include files `icache.h`, `dcache.h`, and `timer.h`, used for on-chip cache and timer control are not supported with the `-ARP` switch.

Known Problems

Renaming Global Variables in Object Modules

If you use the gcc960 `.asm` extension to name a global variable as in:

```
int var1 asm ("my_var1");
```

and use the b.out object format, the debug information for `var1` is not correct, and you will not be able to examine `var1` using the gdb960 debugger.

Invalid Command-Line Options Not Diagnosed

The gcc960 compiler driver does not check the command line options for validity. Invalid options are ignored without producing a warning message.

Assertion failure When Using Pragma Cave

The compiler sometimes produces an assertion failure when inlining a function that has not been declared as compressible (via pragma cave) into a function that been declared as compressible. The workaround is to turn off inlining or use an optimization level of 2 or less (1 or less for ic960) for all those modules that are compressible with CAVE. This problem occurs only when the `-ARP` switch is used.

Converter for b.out, COFF and ELF (cof960 / objcopy)

No changes from Release 5.0.

Converter to IEEE 695 (cvt960)

No changes from Release 5.0.

Coverage Analyzer (gcov960)

Known Problems

Execution Counts for a Function that is Inlined

The reports produced by gcov960 may give misleading information about functions that are inlined. The reports may indicate that the code of the inlined function has never been executed, or may show execution counts that are unexpectedly low. This happens because the inlined code fragments are treated as part of the function they are inlined into and not as part of the original function.

Debugger (gdb960)

Known Problems (Command Line Version)

Long Double Type Not Fully Supported

The C type `long double` is implemented internally as `double`. Variables of `long double` type are stored in IEEE-extended format on the target, but when examining their values in gdb960, or when setting a new value manually, gdb960 stores them internally as type `double`. This is of concern only if a variable value is too large, too small, or has too much precision to be represented as a `double`. See the following note.

Floating-Point Infinities Not Reported

From the gdb960 command line, if you assign a value to a variable with `float`, `double`, or `long double` type, and the value is too large, too small, or has too much precision to represent the variable's type, the variable appears to hold a legal value and no error is reported. The variable's value is meaningless.

CTRL-C Disabled During Target Connect

While gdb960 is attempting to connect to a remote MON960 target it is not possible to break with CTRL-C. If the debugger cannot connect with the target for some reason (*e.g.*, cable not physically connected) the operation times out after about 20 seconds and the gdb960 prompt returns.

i960 Jx, Hx, and RP Memory-Mapped Registers Must Be Read as Words

The memory-mapped registers provided by the i960 Jx, Hx, and RP processors must be read and written in 4-byte quantities. Larger or smaller accesses are flagged as errors by gdb960 and the access is not attempted.

Some i960 Jx, Hx, and RP Memory-Mapped Registers Cannot Be Read or Written

A few of the i960 Jx, Hx, and RP processor memory-mapped registers cannot be written by gdb960. These are IPB0, IPB1, DAB0, DAB1, and BPCON. Attempts to modify them directly causes an HDIL (MON960) error. These registers are modified by MON960 when a hardware breakpoint or hardware watchpoint command is used. Additionally, gdb960 cannot read IPB0 and IPB1 directly. The debugger always reports their current value as 0.

Absolute symbols are Relocated Under -pd Option

When an ELF or DWARF file contains position-independent data, and gdb960 is invoked with the `-pd` option, absolute symbols are erroneously relocated along with data symbols. This condition does not affect the runtime behavior of the program being debugged, but it makes it difficult to print or set the values of the absolute symbols.

Known Problems (GUI Debugger)

Display Anomaly on UNIX*

On UNIX hosts, a clear rectangular box or a Delay dialog box may appear in the center of the source pane after downloading code. To remove the clear rectangular box, refresh the gdb960 window by minimizing it to an icon, then maximize it. The Delay dialog box can be sent to the rear display by right clicking on the title bar.

PCI I/O Interrupt Problem

On Windows NT 3.51 hosts using PCI to communicate with the target board, interrupting the target during file I/O (via HDIL) intermittently generates file write errors. There is no known workaround. Do not interrupt the target while doing file I/O on PCI.

Progress Information

In the current release, some operations lack progress indicators. Three examples are opening binaries, downloading binaries, and disassembling files.

Tcl Error

On UNIX hosts, the debugger may generate a spurious error such as:

```
A Tcl evaluation resulted in an error: invalid command name: "A\"
```

If this message appears, disregard it and close the error message box.

No Symbolic Debug of Optimized Code Support

Though the underlying gdb960 debugger supports Symbolic Debugging of Optimized Code (SDOC), the GUI does not support this functionality.

Dumper/Disassembler (dmp960 / gdmp960)

Changes from Release 5.0

Archive support

gdmp960 now supports dumping of archive files and archive file members. Previous versions of the dumper only worked with object files. Archive support allows you to dump:

- all members of an archive
- one or more object files within an archive
- information on the structure of an archive (e.g., the archive symbol list)

The table below lists the options that allow archive support:

Option	Description
`-e*`	applies all options on the command line (e.g., `-r`, `-f`) to each member of an archive.
`-m`	displays a map of the archive contents. See the first example later in this section.
`-O` *`filename`*`*`	applies all command line options to the named archive member file only.
`-p`	suppresses headers.
`-q`	queries the archive file and displays its object module format and host byte order.
`-t`	displays the archive symbol list.

* Indicates a new dumper option.

The examples that follow show an archive file `lib.a`, which contains object files `a.o`, `b.o`, and `c.o`, in that order.

Displaying Archive Structure Information

This first set of examples shows how the dumper can display information on the structure of an archive file using the `-q`, `-m`, and `-t` options.

This example demonstrates the behavior of the dumper when querying an archive file for its type. The command:

```
gdmp960 -q lib.a
```

produces the output:

```
File:              lib.a
OMF:               elf archive
Host Byte Order:   big
Target Byte Order: unknown
```

In this example, the dumper maps the internal structure of an archive file. The command:

```
gdmp960 -m lib.a
```

produces the output:

```
                          HEX     DEC     OCT
                          ---     ---     ---
+-----------------------+ 0       0       0
|     Magic String      |
|       0x8 (8)         |
+-----------------------+ 8       8       10
|   Symbol List HDR     |
|      0x3c (60)        |
+-----------------------+ 44      68      104
|     Symbol List       |
|      0x64 (100)       |
+-----------------------+ a8      168     250
|       a.o HDR         |
|      0x3c (60)        |
+-----------------------+ e4      228     344
|         a.o           |
|    0x2934 (10548)     |
+-----------------------+ 2a18    10776   25030
|       b.o HDR         |
|      0x3c (60)        |
+-----------------------+ 2a54    10836   25124
|         b.o           |
|    0x253c (9532)      |
+-----------------------+ 4f90    20368   47620
|       c.o HDR         |
|      0x3c (60)        |
+-----------------------+ 4fcc    20428   47714
|         c.o           |
|    0x7b74 (31604)     |
+-----------------------+ cb40    52032   145500
|     END OF FILE       |
|       0x0 (0)         |
+-----------------------+ cb40    52032   145500
```

The `-t` option of the dumper permits dumping of the archive symbol list information. For example, the command:

```
gdmp960 -t lib.a
```

produces the output:

```
Name                          Offset              Filename
_dwarf_init                   168                 a.o
_dwarf_tag                    10776               b.o
__dw_build_a_die              20368               c.o
__dw_build_tree               20368               c.o
__dw_build_cu_list            20368               c.o
```

Dumping the Contents of Archive Members

The dumper also now allows you to disassemble or display information about a file within an archive by using the `-e` and `-O` options in combination with gdmp's other command line switches. In the example below, the `-e` option applies all command line options to each member of an archive:

```
gdmp960 -q -e lib.a
```

```
a.o:
File:              a.o
OMF:               elf
Host Byte Order:   big
Target Byte Order: little

b.o:
File:              b.o
OMF:               elf
Host Byte Order:   big
Target Byte Order: little

c.o:
File:              c.o
OMF:               elf
Host Byte Order:   big
Target Byte Order: little
```

The example below shows how the `-O` option lets you apply all command line options to the named object file only. The command:

```
gdmp960 -q -Oa.o -Oc.o lib.a
```

produces the following output:

```
a.o:
File:              a.o
OMF:               elf
Host Byte Order:   big
Target Byte Order: little

c.o:
File:              c.o
OMF:               elf
Host Byte Order:   big
Target Byte Order: little
```

Dumping Absolute Symbols

The `-S` option instructs the disassembler to output symbol labels rather than their values for any symbols for which you have specified the absolute address. This option works in conjunction with the `-s` (lowercase) option, which instructs the disassembler to perform symbolic disassembly. For example, with an object file created with the following instructions:

```
.globl    proc1
.set      proc1,0xc
callx     proc1
callx     0xc
addi      proc1,r5,r6
```

If you use the following gdmp960 command line:

```
gdmp960 t2.o -s
```

you would see the output:

```
Section '.text':
0:  8600000c              callx     0xc
4:  8600000c              callx     0xc
8:  5931488c              addi     12,r5,r6
```

Notice that in the second line, `proc1` from the source code is converted to `0xc`, the user-specified address for `proc1`.

Adding the `-S` option to the command line instructs the disassembler to display the symbol name instead of its address. For example, this command line:

```
gdmp960 t2.o -s -S
```

produces the following output:

```
Section '.text':
0:  8600000c             callx     proc1
4:  8600000c             callx     proc1
8:  5931488c             addi     12,r5,r6
```

Notice that in the both callx statements, `proc1` now appears instead of `0xc`. Using the `-S` option causes the disassembler to display the symbol name for all calls to that address.

NOTE. *This option was supported in the rev. 5.0 disassembler as the undocumented `-A` switch. This option has been renamed `-S`.*

Libraries

This release of CTOOLS adds libraries and linker directive files the i960 RP processor that are distinct from the i960 Jx processor libraries and directive files. A consequence of adding the new libraries has been a change in the names of several existing libraries as noted in the section that follows.

Library File Names Changed

The following libraries have been renamed:

Old Name	New Name	Description
libfp.a	Unchanged	US Software floating-point library
libfp_e.a	libfpe.a	Position-independent data version
libfp_b.a	libfpb.a	Big-endian version
libfp_p.a	libfpp.a	Big-endian, position-independent data version
libhis.a	libhs.a	Profiling library for ghist960
libhis_p.a	libhs_p.a	Position-independent data version
libhis_b.a	libhs_b.a	Big-endian version
libhis_e.a	libhse.a	Big-endian, position-independent data version
libll.a	Unchanged	Low-level library
libll_p.a	libllp.a	Position-independent data version
libll_b.a	libllb.a	Big-endian version
libll_e.a	liblle.a	Big-endian, position-independent data version

continued ☛

Old Name	New Name	Description
libmon.a	libmn.a	Interface to board-specific functions in MON960
libmon_p.a	libmnp.a	Position-independent data version
libmon_b.a	libmnb.a	Big-endian version
libmon_e.a	libmne.a	Big-endian, position-independent data version
libmstb.a	libmst.a	Stub math library
libmstbp.a	libstp.a	Position-independent data version
libmstbb.a	libmstb.a	Big-endian version
libmstbe.a	libste.a	Big-endian, PID data version
libq.a	Unchanged	Profiling library when no file system is available
libq_p.a	libqp.a	Position-independent data version
libq_b.a	libqb.a	Big-endian version
libq_e.a	libqe.a	Big-endian, position-independent data version
libqf.a	Unchanged	Profiling library when a file system is available
libqf_p.a	libqfp.a	Position-independent data version
libqf_b.a	libqfb.a	Big-endian version
libqf_e.a	libqfe.a	Big-endian, position-independent data version
librom.a	librm.a	Library for supporting serially-reusable programs
librom_p.a	librmp.a	Position-independent data version
librom_b.a	librmb.a	Big-endian version
librom_e.a	librme.a	Big-endian, position-independent data version

i960 RP Processor-Specific Libraries and Linker Directive Files Added

The libraries and Linker Directive Files for the i960 RP processor have now been separated from the i960 Jx processor files. The new names are as follows:

Library Name	Description
libhrp.a	US Software floating-point library
libhrp_p.a	Position-independent data version
libhrp_b.a	Big-endian version
libhrp_e.a	Big-endian, position-independent data version
libcrp.a	High-level C library
libcrp_p.a	Position-independent data version
libcrp_b.a	Big-endian version
libcrp_e.a	Big-endian, position-independent data version
libfprp.a	US Software floating-point library
libfprpe.a	Position-independent data version
libfprpb.a	Big-endian version
libfprpp.a	Big-endian, position-independent data version
libhsrp.a	Profiling library for ghist960
libhsrpp.a	Position-independent data version
libhsrpb.a	Big-endian version
libhsrpe.a	Big-endian, position-independent data version

continued ☛

Library Name	Description
libllrp.a	Low-level library
libllrpp.a	Position-independent data version
libllrpb.a	Big-endian version
libllrpe.a	Big-endian, position-independent data version
crtrp.o	Initialization (start-up) library
crtrp_p.o	Position-independent data version
crtrp_b.o	Big-endian version
crtrp_e.o	Big-endian, position-independent data version
libmrp.a	Math library for i960 Cx and Hx processors.
libmrp_p.a	Position-independent data version
libmrp_b.a	Big-endian version
libmrp_e.a	Big-endian, position-independent data version
libmnrp.a	Interface to board-specific functions in MON960.
libmnrpp.a	Position-independent data version
libmnrpb.a	Big-endian version
libmnrpe.a	Big-endian, position-independent data version
libstrp.a	Stub math library.
libstrpp.a	Position-independent data version
libstrpb.a	Big-endian version
libstrpe.a	Big-endian, position-independent data version

continued ☛

Library Name	Description
libqrp.a	Profiling library when no file system is available
libqrpp.a	Position-independent data version
libqrpb.a	Big-endian version
libqrpe.a	Big-endian, position-independent data version
libqfrp.a	Profiling library when a file system is available
libqfrpp.a	Position-independent data version
libqfrpb.a	Big-endian version
libqfrpe.a	Big-endian, position-independent data version
librmrp.a	Library for supporting serially-reusable programs
librmrpp.a	Position-independent data version
librmrpb.a	Big-endian version
librmrpe.a	Big-endian, position-independent data version
cyrx.ld	Linker command file for Cyclone RP board
cyrxp.ld	Linker command file for Cyclone RP board, position-independent data
cyrxfls.ld	Linker command file for Cyclone RP flash
cyrxpfls.ld	Linker command file for Cyclone RP flash, position-independent data
mcyrx.gld	Compiler directive file for Cyclone RP board
mcyrxfls.gld	Compiler directive file for Cyclone RP flash

Linker (gld960 / lnk960)

Changes from Release 5.0

-q switch has been retired

This switch allowed the user to adjust the search paths used by the linker in order to support release 4.6 style library installations. Although the linker will still check the old paths during a library search, users will no longer be able to direct the linker to use these paths explicitly.

New Architecture Options

The linker now accepts `-ARP`, `-ACORE0`, `-ACORE1`, `-ACORE2`, and `-ACORE3` architecture switches or environment variable settings. See "Assembler (asm960 / gld960)" for more information on these architecture options. The following table shows the input/output compatibilities of all architectures that are supported by the toolset.

		Output									
		SA/KA	**SB/KB**	**Cx**	**Jx**	**Hx**	**RP**	**CORE0**	**CORE1**	**CORE2**	**CORE3**
	SA/KA	C	C	NA	NA	NA	NA	NA	NA	NA	NA
	SB/KB	NA	C	NA	NA	NA	NA	NA	NA	NA	NA
I	**Cx**	NA	NA	C	NA	NA	NA	NA	NA	NA	NA
n	**Jx**	NA	NA	NA	C	C	NA	NA	NA	C	NA
p	**Hx**	NA	NA	NA	NA	C	NA	NA	NA	NA	NA
u	**RP**	NA	NA	NA	C	C	C	C	NA	C	NA
t	**CORE0**	NA	NA	NA	C	C	C	C	NA	C	NA
	CORE1	C	C	C	C	C	NA	NA	C	C	C
	CORE2	NA	NA	NA	C	C	NA	NA	NA	C	NA
	CORE3	NA	NA	C	C	C	NA	NA	NA	C	C

C = compatible.

NA = incompatible. Warning issued.

Changes to the calljx Pseudo Instruction When -ARP Is Selected

The `calljx` pseudo instruction lets you assemble a call instruction, allowing the linker to perform call optimization, when possible. For example, inserting a `calljx` instruction while using the `-AJD` setting might produce the following linker output depending upon whether the target is a default call, leaf procedure, or system call:

Default Call	Leaf Procedure	System Call
`callx _target`	`balx _target,g14`	`lda _sysprocIndex,g13` `calls (g13)`

When used with the new `-ARP` option, `calljx` uses the syntax:

```
calljx _target, tmpreg
```

where *`tmpreg`* is a local or global register. This change results in the following sequences in the linker:

Default Call	Leaf Procedure	System Call
`lda _target,tmpreg` `callx (tmpreg)`	`lda _target,tmpreg` `balx (tmpreg),g14`	`lda _sysprocIndex,g13` `calls (g13)`

Notice that with the i960 RP processor `calljx` format all three call types result in a three-word instruction sequence, whereas the previous `calljx` format requires only two words.

Library Search Order When i960 RP Architecture Is Selected

When an non-i960 RP architecture is specified, the linker searches first for architecture-neutral libraries, then for architecture-specific libraries. For example, when the linker looks for the i960 KA processor `libc` library, it first tries to find `libc.a` and if the library is not found, the linker looks for `libcka.a`. Because files targeted for the i960 RP processor require target-specific libraries, the linker looks first for architecture-specific libraries (e.g., `libcrp.a`), and if those libraries are not found, the linker looks for architecture-neutral libraries (e.g., `libc.a`).

New PRE_HLL() Directive

The new linker directive PRE_HLL() allows the user to specify libraries that are processed immediately before the high-level language libraries specified with the HLL() directive. The syntax for the new directive is:

```
PRE_HLL(libraries)
```

libraries	is one or more high-level support libraries to be linked prior to those specified with an HLL() directive.

The linker now loads the object files and libraries in the following order:

1. The file name specified with STARTUP.
2. All the object files and libraries listed individually in the invocation, in the order listed.
3. All the object files and libraries listed individually in the directive files, in the order listed.
4. All the libraries specified with PRE_HLL.
5. All the libraries specified with HLL or default libraries in response to HLL().
6. All the libraries specified with SYSLIB.

Two-Pass Compilation Requires a .text Section

The linker does not properly handle a file with cc_info (two-pass profiling information generated by the compiler) without the presence of a .text section.

Group() Directive Section Ordering

The linker directive GROUP() allows users to block a group of output sections into a single unit. Although the GROUP() directive should keep the sections in the order specified, empty sections that contain no external symbols and are not user-defined (e.g., .data, .text, .bss) do not always appear in the specified order. This behavior can be avoided by declaring an external symbol in the empty section.

Known Problems

Inappropriate Error Messages When Sections Overlap

The linker emits inappropriate error messages when sections overlap. For example, in the linker directive file:

```
SECTIONS {
    .text 0 : {}
    .data 0x10 : {}
}
```

if the sum of the `.text` sections is greater than `0x10`, but the `.data` section size is less than `0x10`, the linker emits the following error message:

```
Section .data (start 16: size: 0) won't fit into defined memory.
```

This is misleading as it is the sum of the `.text` sections causing the overlap, not the size of the `.data` section.

Macro Processor (mpp960)

Known Problems

Built-in Macro, sysval()

The `sysval()` built-in macro does not work on Windows 95 or Windows NT.

Munger (gmung960)

No changes or problems are known at this time.

Name Lister (gnm960 / nam960)

No changes or problems are known at this time.

Profile Merger (gmpf960)

No changes or problems are known at this time.

Profile Decision Maker (gcdm960)

No changes or problems are known at this time.

Rommers (grom960, rom960)

No changes or problems are known at this time.

Section Size Printer (gsize960 / siz960)

Changes from Release 5.0

New -n Option

The `-n` option includes ELF `.debug` sections in the size calculation. Note that using `siz960/gsize960` with the `-n` option produces output that is identical to that produced by version 5.0 of the sizer.

Statistical Profiler (ghist960)

No changes or problems are known at this time.

Stripper (gstrip960 / str960)

No changes or problems are known at this time.

Version Printer (gver960)

No changes or problems are known at this time.

Hypertext

Known Problems

Decision Maker Syntax Error

The syntax for the gcdm Substitution Controls should read:

```
subst=arg[,arg]... and nosubst=module-set
```

Manual Update

Correction

Page 12-3, Table 12-1 of the *i960® Processor Software Utilities User's Guide* erroneously lists the `mkfill` option as `mkfile`.

Updated Legal Section

The following text replaces the legal section found in the CTOOLS 5.1 manuals:

> Copyright © 1988, 1989, 1990, 1991, 1992, 1993, 1994, 1995 Free Software Foundation, Inc.
>
> Permission is granted to make and distribute verbatim copies of this manual provided the copyright notice and this permission notice are preserved on all copies.
>
> Permission is granted to copy and distribute modified versions of this manual under the conditions for verbatim copying, provided also that the entire resulting derived work is distributed under the terms of a permission notice identical to this one.
>
> Permission is granted to copy and distribute translations of this manual into another language, under the above conditions for modified versions.

Note also that the updated legal section for the *gdb960 User's Manual* includes the following text:

> Copyright © 1984 - 1996 Wind River Systems, Inc.

* Other brands and names are the property of their respective owners.