



---

## Developer Package for Release 3.0 Release Notes

---

### Release Notes for the Developer Package

This document lists the most important changes to the Software Development Kit (SDK) since the previous release. In addition, it lists the known open issues and limitations in this release.

For more information regarding the status and workarounds related to any of these issues, please contact ClearSpeed support quoting the relevant CTS number.

You should check the ClearSpeed customer support website (<http://support.clearspeed.com>) for updates to these release notes.

# 1 What's new in Release 3.0

Release 3.0 contains many improvements to the development tools which have been made since Release 2.51. Highlights include:

- Improvements to optimizing compiler. Many new optimizations have been added, particularly for poly code. This should lead to significant performance improvements for C programs.
- Improved use of on-chip memory:
  - Ability to place critical code in fast on-chip memory using the `hot` pragma
  - Dynamic allocation of stacks and runtime checking of stack usage

**Warning:** The way that stack sizes are specified has changed in this release. If you specify stack sizes in your code then you *must* change your code to use one of the new methods. Failure to do so may cause your program to fail in unpredictable ways.

*Note:* If you get an unexpected "Not enough memory" error from `csrun` or `CSAPI_load` then a possible cause is the use of the old-style stack size specification.

- New library functions, including:
  - Reduction functions for generating results from data on the processor array
  - New functions in the Vector Math Library: `cs_erfp`, `cs_efcp`, `cs_powp`, `cs_log10p` and `cs_signp`.
  - Improvements to the asynchronous `memcpy` functions
- A number of improvements to the programming model such as:
  - Operator overloading to simplify the use of vector data types
- The CSX600 cycle-accurate simulator, `casim`, is included for the first time. This can be used in conjunction with the ClearSpeed Visual Profiler (`csvprof`) to provide extended instruction-issue pipeline profile information for user applications.
- Improvements to the host interface library, CSAPI.
- A preview of the Eclipse integrated development environment (IDE) will be available with the 3.0 release. This will be the first release of the ClearSpeed Eclipse IDE and will integrate the SDK toolchain and provide a graphical user interface for the `cs gdb` debugger.

## 2 Issues fixed in Release 3.0

**CTS 2119:** Object files or executables created with a given release of the SDK are not guaranteed to be compatible with other versions of the runtime libraries. The runtime code now checks for a compatible CSX executable before attempting to load it.

**CTS 1257:** In previous releases, the size of both the mono and poly stack sizes were hard coded in the C<sup>#</sup> runtime.

### 2.1 Compiler

**CTS 4321:** The compiler could generate incorrect assembly code for floating-point expressions that can be statically evaluated to NaN or Inf.

**CTS 4404:** The compiler would fail with an internal error on the declaration of functions that return a struct and that have an empty parameter list.

**CTS 4407:** When initializing global declarations, certain address calculations in the initializer expression could cause internal compiler errors.

**CTS 4408:** The following piece of code caused an internal compiler error.

```
unsigned long x[4];
void foo(void) {
    ((void (*)())(x+2))();
}
```

**CTS 4409:** Certain function declarations for which the parameters are undefined would cause internal compiler errors.

**CTS 3774:** There was a known instability in the compiler running on the windows platform. In such situations the compiler would either perform an illegal memory access and fail with a pop-up box informing you of the bad memory access, or a message similar to the one was displayed:

```
Internal fatal error in engine setpsr (pid = 1864):
failed domcheck; dom=355(LIST_mirParameter), op=23073896(???)
Compiler quits
```

**CTS 4937:** In certain cases, when compiling code at optimization level `-O4`, it was possible that you would get an internal compiler error of the following form:

```
0x8c7e4b0 mirCondAssign(Rhs, Lhs) NO CODE SELECTED
...
Internal fatal error in engine match (pid = 754):
no code selected
Compiler quits.
```

**CTS 4955:** There was a problem compiling and/or assembling code that contained eight-byte integer constants on 64-bit platforms. The assembler's `.msbytes` and `.lsbytes` directives failed to process the constant correctly, resulting in incorrect assembly that would build without warnings, but would produce erroneous results at runtime.

**CTS 4974:** Certain functions caused a compilation error when an argument to the function was an expression containing a `const` qualified variable. For example, the following code:

```
const poly float one = 1.0;
... cs_logp(one + one) ....
```

could cause the error:

```
type-generic call to 'cs_logp_explicit_mem' could not find a matching
candidate
Warning: implicit declaration of function 'cs_logp_explicit_mem'
```

**CTS 5091:** There was a problem with the implementation of dynamic stack checking when you have made changes to the default stack model (either to add stacks for additional threads, or to remove or redefine the stacks that are defined by default).

**CTS 5477:** The `?:` conditional operator produced incorrect results with some operand types.

## 2.2 Debugger

**CTS 949:** A breakpoint set on the macro `cycles.get`, could cause the breakpoint to be hit again on continue when the cycle count register wraps.

**CTS 4417:** The debugger did not work correctly with dynamically linked executables.

**CTS 4482:** In some cases `csgdb` reported erroneous values for function parameters.

**CTS 4687:** When both `-g` and an optimization level of `-O1` or above were specified, the debugging information was incomplete. This meant that variables which were allocated to registers could not be printed in the debugger.

**CTS 4846:** When debugging code containing the `DVECTOR * type` `csgdb` would print the address with a size of four bytes.

## 2.3 Assembler

**CTS 568:** The assembler did not correctly handle parameters to the `.double` directive.

## 2.4 Profiler

**CTS 3745:** The documentation did not clearly state that the ClearSpeed Visual Profiler (`csvprof`) required Sun's Java Runtime Environment version 1.5 or later be installed as a prerequisite.

## 2.5 Libraries

**CTS 647:** Calling `dprint` frequently on the board caused the host to block other interrupts causing the system to slow down.

**CTS 2974:** For consistency with the latest implementation of the BLAS library in CSXL, the name of the FFT library has been changed:

```
from libcsdft.so to libcsxl_csdft.so (Linux)
from csdft.dll to csxl_csdft.dll (Windows)
```

**CTS 4243:** The documentation for the standard libraries contains sections on memory usage for each function. Some of the figures in these tables were incorrect.

**CTS 4322:** Calling `printfp` to display a poly floating point expression that evaluates to NaN could cause the code to hang at runtime.

**CTS 4373:** The `sprintf()` function did not append a null (`'\0'`) character to the end of the generated string.

## 2.6 Linker

**CTS 5090:** Overriding the stack sizes for thread 0 did not work as documented.

## 2.7 Instruction Set

**CTS 2460:** There was an error in the swazzle documentation. All of the following instructions could also take eight-byte operands, each taking four cycles:

- `swazzle.low.out.get`
- `swazzle.high.out.get`
- `swazzle.low.in.put`
- `swazzle.high.in.put`

There was also an error in some of the arithmetic instructions. Both `addc` and `subc` support the following versions, the cycle counts being as follows:

- `addc p2, p2, p2 2 cycles`
- `addc p4, p4, p4 4 cycles`
- `subc p2, p2, p2 2 cycles`
- `subc p4, p4, p4 4 cycles`

## 3 Known Issues in Release 3.0

The following issues are currently open.

### 3.1 Installation

**CTS 1285:** The Windows installer for the SDK creates an empty file called `Admin` in the directory it is run from. This file can be deleted or ignored.

**CTS 2113:** When the SDK is installed on a security enabled linux distribution (SELinux) a problem can occur. To ensure correct operation of the tools, the user should ensure that they either disable the security extensions, or run the following command to add all the shared libraries in the installation to the `texrel_shlib_t` context:

```
chcon -t texrel_shlib_t /opt/clearspeed/csx600_m512_le/lib/*.so
```

### 3.2 Compiler

**CTS 3797:** There is a known problem with the compiler that may cause unusually long compile times. If you find that the compile stage of building a program is very slow, you can try disabling peephole optimization by using the compiler, `cscn`, with the `--nopeephole` command line option. This may allow compilation to complete in a reasonable amount of time.

**CTS 4929:** When dynamic stack checking is enabled, the compiler will generate code to calculate the expected stack usage of the function. Currently, this assumes the worst case for memory use, which is that a function includes a `div` instruction requiring 28 extra bytes of poly stack and 4 bytes of mono stack. These values are always added to the stack usage of any function for the purposes of checking for overflow. This means that a stack overflow warning may be generated even though there is enough memory available.

**CTS 5343:** Location lists in the DWARF2 debug information generated by the compiler will contain absolute locations, rather than relative locations. In this respect, the debug information deviates from the DWARF2 standard.

**CTS 5670:** An incomplete array declaration such as `int arr[]` (if not made complete by a subsequent declaration) should become an array of one element, initialized with zero. Currently the compiler will produce a zero-initialized array with length derived from the element type. For example the array above will generate a 4 element array based on the 4-byte element type.

**CTS 5671:** A volatile expression, if used as the left-hand side of the `'` operator (as shown below) will not be evaluated. This does not respect the volatile attribute.

```
int fun(volatile int * p)
{
    return (*p, 0);
}
```

**CTS 5708:** There is a known bug in one of the optimization components of the **C<sup>n</sup>** compiler. This may cause incorrect results to be generated for code involving poly expressions.

Should this be the case, the following command-line option can be given to `cscn` to prevent the error:

```
-Wcn,--*.csPolyAssignAnalysis:skip_analysis
```

This only occurs at optimization levels of `-O3` and above, so a simple solution to test whether this is the problem is to run the compiler at `-O2`.

**CTS 5710:** There is a known bug in the optimization engine in the compiler that becomes enabled at `-O3`. The manifests itself with erroneous results at runtime. If you suspect that this may be causing a problem you can either reduce the optimization level to `-O2`, or disable this specific optimization using the following option to `cscn`:

```
-Wcn,--*.csNormaliseOffsetsLow:skip
```

This has the benefit of only disabling this optimization and not all optimizations at `-O3` or `-O4`. There may be some performance impact to disabling this optimization.

**CTS 5720:** There is a known bug which can occur whilst completing some comparisons of poly expressions. The bug is in the scheduler and can be avoided by using the `--nosched` option to `cscn`. Additionally the scheduler is only enabled at `-O3`, so lowering the optimization level to `-O2` or lower will also solve the issue.

### 3.3 Libraries

**CTS 5650:** The `async_memcpym2p_inline` suite of functions reuse the specified semaphore internally to check all load stores have completed. Therefore it is unsafe to issue two calls back to back using the same completion semaphore. This could cause problems copying data that has not yet been stored to memory. The workaround is to use different completion semaphores for calls you wish to chain together.

**CTS 5692:** Chapter 3.5 of the **C<sup>++</sup>** Standard Libraries Reference Manual describes the methodology for performing DMA between DRAM and on-chip memory (ESRAM). What is not mentioned is that the steps only support such DMA operations for chip 0. This means that the local address arguments to the `cs_initDCD` function `lAddressW1` and `lAddressW2` must correspond to a local address in DRAM on chip 0. The same method will not operate correctly for DMA on chip 1.

### 3.4 Debugger

**CTS 309:** It is not possible currently to cast a value in the `csgdb` command language to a poly type.

**CTS 393:** It is not possible to currently view poly structure members individually. The whole structure will be displayed. Trying to reference the individual members of poly structures will produce an error message.

**CTS 864:** If you read poly registers or poly memory while using `csgdb` to single step through the instructions which issue a PIO transfer, it is possible to corrupt the data in the PIO node.

**CTS 1875:** If you attempt to debug code executing on thread 1 when it is currently executing a `sem.wait` instruction, the debugger will lock up if an attempt is made to read poly state.

## 3.5 Simulators

**CTS 191:** When code executing on the simulator (`isim`) writes to address 0x0, the simulator will stop with an error indicating that it has tried to access out-of-range memory. This causes `csgdb` or `csr` to become unresponsive.

**CTS 4146:** The socket connection between `isim` and the runtime components (`csreset`, `csr`) may fail occasionally on SLES9 platforms. The failure is reported as “socket 0 service Socket Exception: receive length == 0”. The workaround is to restart `isim` and try again.

**CTS 4923:** The socket connection between the simulators (`casim` and `isim`) and the runtime components (`csreset`, `csr`) is very slow when run on SLES10. This may cause timeout errors. The workaround is to run, at least, the runtime components on a different (non SLES10) operating system. If it is necessary to run the simulator on a SLES10 system then this can be done by running the runtime components on another host and connecting to the simulator using the `--host` command line option.

**CTS 5849:** The information displayed by `casim --help` for the `-i` option is misleading. It should say: “Instance of simulator for host tools to connect to. (0-31)”

## 3.6 Instruction Set

**CTS 255:** The 4 byte poly shift instructions `lsl`, `lsr` and `asr` do not work when destination and source operands are overlapped. There is no warning from the assembler that this is the case.

**CTS 4325:** The mono and poly divide instructions are not IEEE compliant in their handling of divide by zero. The result is undefined. They also do not correctly perform rounding, which will lead to slight inaccuracies. This will affect both compiled and assembler code.

**CTS 6108:** The documentation states that you can use the `--use-add-ucode` option to gain access to an extended swizzle API as well as some complex multiply instructions. The necessary files for this have not been included with the release and so this will not work. There is no workaround for this issue.

## 3.7 CSDFT

**CTS 2483:** Using `printfp` in conjunction with the CSDFT library will fail at link time with the error message:

```
Definition for the symbol 'PRINT_AREA_CONTROL' already found in module
default.cso
```

**CTS 2666:** When the environment variable `CS_CSAPI_DEBUGGER` is set, the CSDFT host library assumes that the `.csx` file to be loaded has `_debug` appended to the file name. If this file does not exist on the `CSPATH`, the library will fail to find the `.csx` file.

**CTS 3308:** The function `CSDFT_get_csapi_handle_board` appears in `csapi_support.h` but is not documented in the [CSDFT Reference Manual](#) nor does it work correctly. Use of this function is not currently supported.

**ClearSpeed Technology, Inc.**

3031 Tisch Way, Suite 200  
San Jose, CA 95128  
United States of America

Tel: +1 408 557 2067  
Fax: +1 408 557 9054

Email: [info@clearspeed.com](mailto:info@clearspeed.com)

Web: <http://www.clearspeed.com>

Support: <http://support.clearspeed.com>

**ClearSpeed Technology plc**

3110 Great Western Court  
Hunts Ground Road  
Bristol BS34 8HP  
United Kingdom

Tel: +44 (0)117 317 2000  
Fax: +44 (0)117 317 2002

1. Information and data contained in this document, together with the information contained in any and all associated ClearSpeed documents including without limitation, data sheets, application notes and the like ('Information') is provided in connection with ClearSpeed products and is provided for information only. Quoted figures in the Information, which may be performance, size, cost, power and the like are estimates based upon analysis and simulations of current designs and are liable to change.
2. Such Information does not constitute an offer of, or an invitation by or on behalf of ClearSpeed, or any ClearSpeed affiliate to supply any product or provide any service to any party having access to this Information. Except as provided in ClearSpeed Terms and Conditions of Sale for ClearSpeed products, ClearSpeed assumes no liability whatsoever.
3. ClearSpeed products are not intended for use, whether directly or indirectly, in any medical, life saving and/ or life sustaining systems or applications.
4. The worldwide intellectual property rights in the Information and data contained therein is owned by ClearSpeed. No license whether express or implied either by estoppel or otherwise to any intellectual property rights is granted by this document or otherwise. You may not download, copy, adapt or distribute this Information except with the consent in writing of ClearSpeed.
5. The system vendor remains solely responsible for any and all design, functionality and terms of sale of any product which incorporates a ClearSpeed product including without limitation, product liability, intellectual property infringement, warranty including conformance to specification and or performance.
6. Any condition, warranty or other term which might but for this paragraph have effect between ClearSpeed and you or which would otherwise be implied into or incorporated into the Information (including without limitation, the implied terms of satisfactory quality, merchantability or fitness for purpose), whether by statute, common law or otherwise are hereby excluded.
7. ClearSpeed reserves the right to make changes to the Information or the data contained therein at any time without notice.

© Copyright ClearSpeed Technology plc 2007. All rights reserved.

Advance is a registered trademark of ClearSpeed Technology plc

ClearSpeed, ClearConnect, Advance and the ClearSpeed logo are trade marks or registered trade marks of ClearSpeed Technology plc. All other brands and names are the property of their respective owners.