



Developer Package for Release 3.1 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 3.1

Release 3.1 contains improvements to the development tools which have been made since Release 3.0. Highlights include:

1.1 New documentation availability

The latest and most accurate documentation is now only available via the ClearSpeed support website at <http://support.clearspeed.com/documentation/>. Documentation is no longer provided with the software.

1.2 Support for new processors

The tools now support both the CSX600 and the CSX700 processors. The same code is generated for both processors. The only exception is that code written for the CSX600 using non-ECC memory instructions (with the compiler option `--no-poly-ecc` or the assembler pragma `non_ecc_st on`) will *not* run on the CSX700 unless ECC is disabled.

As a consequence, the predefined Cn macro `__CSX600__` is no longer defined.

The names of the installation packages have changed. The string `CSX600` has been removed from the package names and `clearspeed` has been added as a common prefix.

The simulators `isim` and `casim` have a new command line option to choose the processor type to be simulated. The `--processor-type` option takes a parameter of either `CSX600` or `CSX700`. The default (if the option is not used) is `CSX700`.

The installation path for the software has changed to remove the target specific directory. The software now installs directly under the `clearspeed` directory, that is:

- For Linux: `/opt/clearspeed`
- For Microsoft Windows: `C:\Program Files\clearspeed`

You may need to update any environment variables, makefiles, or build scripts which specify the search path for include files, libraries, and so on.

1.3 Dynamic linking

The linker now produces object files for dynamic loading by default. There is a new command line option to select static linking if required. The option `--static` option takes a parameter of either `CSX600_C0/CSX600_C1/CSX700_C0/CSX700_C1`⁽¹⁾ (where the `_Cx` postfix is to indicate which chip to statically link for). This will lay out memory for the appropriate processor using a predefined linker script which should be present in the path specified by the `CSPATH` environment variable.

The option `--use-script filename` allows you to specify your own linker script file to control static linking⁽¹⁾.

1. The options are accepted by the SDK driver not the linker.

The linker script format is similar to that used by the Linux `ld` command. They have the following form:

```
MEMORY
{
  monodram : ORIGIN = 0x80000000, LENGTH = 512M
  monosram : ORIGIN = 0x02000000, LENGTH = 128K
  polyram : ORIGIN = 0x0, LENGTH = 6K
  noload : ORIGIN = 0x0, LENGTH = 512M
}
```

1.4 Dual channel DMA

The driver and CSAPI now support the dual channel DMA feature of the Advance e710 and Advance e720 accelerators under Linux platforms only.

Dual channel DMA provides the ability to read and write at the same time across the PCI Express bus, making use of the increased combined bandwidth. It also provides the ability to interleave large DMA transfers with smaller DMA transfers in the same direction. The total bandwidth remains the same, but the smaller transfers can start and finish before the larger transfer has completed.

When the `CSAPI_read_mono_memory` and `CSAPI_write_mono_memory` functions are called from separate threads, they will each block until their transfer has started and completed. Each call will take the next available DMA channel and start the transfer as soon as possible. There is no defined order in which multiple threads are allocated a DMA engine. Optimum use is achieved by performing one large read and one large write operation at the same time.

The `CSAPI_feature` function can be used to determine the number of DMA channels supported by the currently connected hardware.

1.5 CSPX host interface library

This release includes the first release of a new API called CSPX for communicating between the host and software running on one or more accelerators.

CSPX provides a high-level API that can easily be used by an application running on the host and some number of accelerators. CSPX can be used by a host application to call functions running on the accelerators and pass data to and from those accelerated functions. CSPX complements and extends the features provided by CSAPI. This allows you to concentrate on decomposing the application problem for acceleration and on implementing the algorithms on the host and accelerator, rather than the interface between them.

This is a preview release of CSPX intended to provide early access to developers and get feedback on the supported features. It is downloadable as a separate package from the ClearSpeed support site (<http://support.clearspeed.com>).

See the *CSPX User Guide* for more information.

CSPX supports the following features for both C and C++ host applications:

- Transparent Remote Procedure Call (RPC) allowing the host to call functions on an accelerator as if they were running on the host.
- A simple “object migration” model for data movement between host and accelerators. This separates data transfer and synchronization and makes it easy to overlap communication and compute.
- Extensions for managing groups of accelerators, double buffering, broadcast and pipes.

2 Issues fixed in Release 3.1

2.1 Installation

CTS 6241: The Linux `bashrc` setup script in `/opt/clearspeed/bin/` did not correctly configure the `CSHOSTINC` environment variable for the developer package.

2.2 Compiler

CTS 5670: The compiler did not correctly handle an incomplete array declaration such as `int arr[]`.

CTS 5671: A volatile expression used as the left-hand side of the `' , '` operator was not evaluated.

CTS 5720: There was a problem with comparisons of poly expressions at optimization levels O3 and above.

CTS 5748: The compiler could fail with the error 'Internal error in engine match'. This occurred on code that took the address of a global poly object and assigned it to a mono variable.

CTS 5848: There was a known bug in an optimization phase of the compiler which incorrectly performed type conversions implied in an operation or explicitly by casting.

2.3 Debugger

CTS 6052: The `csgdb` documentation stated that you can automatically attach to a running application's CSX code by using the `CS_CSAPI_DEBUGGER` environment variable. This did not work properly.

2.4 Standard libraries

CTS 6005: In the documentation for the Random Number Generator library (`libcs_rng`), the values for the amount of poly memory used by each of the generators were incorrect.

2.5 Simulators

CTS 191: When code executing on the simulator (`isim`) wrote to address 0x0, the simulator would stop with an error indicating that it had tried to access out-of-range memory.

2.6 CSDFT

CTS 5897: The functions for 3D FFTs did not work in release 3.1 Alpha.

2.7 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 5905: There was an issue with the `lsl/r p1, p1, immed`, `lsl/r p2, p2, immed`, `lsl/r p3, p3, immed` and `lsl/r p4, p4, immed` instructions where the immediate is one of 1, 2 or 4. The status was left unchanged by these instruction and any subsequent conditionals based on the status after this operation would have unpredictable results.

CTS 6002: The instruction set document included entries for unsigned compare instructions (`cmp`, `cmp.xx`, `cmpc`). In fact, all integer comparisons treat the operands as if they are signed.

CTS 6108: The documentation stated 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 were not been included with the previous release.

2.8 CSPX

CTS 6069: This is the first public release of this API and, while care has been taken to ensure it is correct and usable, changes may need to be made in future releases. In the preview (Beta) releases, some functions did not use the standard way of handling errors.

CTS 6070: The function `CSPX_process_group_get_return_code` was declared in the header file `cspcx_process_group.h` but not defined.

CTS 6120: The CSPX example `cpp_dbl` will terminate with an exception before execution completes. This is because the host code calls the function `cspcx_terminate_function` on the accelerator with no parameters. This is a valid thing to do but causes an error in the preview release.

3 Known Issues in Release 3.1

The following issues are currently open.

3.1 Installation

CTS 1285: The Microsoft 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/lib/*.so
```

3.2 Assembler

CTS 3369: The assembler does not check all instructions that take constant operands for the correct size of operand. This may result in run-time errors if the operand given for such an instruction exceeds the size of the field for that operand within the instruction definition.

3.3 Compiler

CTS 3774: There is a known instability in the compiler running on the Microsoft Windows platform. In such situations the compiler will 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 will be displayed:

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

Should this occur, set the environment variable `ECP_MEMINIT=1` and recompile your code.

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 driver, `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.

3.4 Standard 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 *Cn Standard Libraries Reference Manual* describes the methodology for performing DMA between DRAM and on-chip memory (ESRAM). This does not state that these functions only support 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 work for DMA on chip 1.

CTS 5700: In the mono and poly standard libraries, variants of the following functions are known to be inaccurate on certain inputs:

```
gamma
erf
erfc
```

Where possible we would advise the use of equivalent functions in the vector math library, as these are known to be better in terms of accuracy and far better in terms of performance.

CTS 5845: The `get_cycles` and `get_cycles_ila` functions in the Cn extension library may be adversely affected by compiler optimizations. This is due to the fact that the functions are not barriers to code motion in the compiler. This means that it is possible for expressions or statements that a user may expect to be timed by calls to `get_cycles/get_cycles_ila` to be moved across the calls and hence not be timed. This is likely to be more impacted by higher levels of optimization at `-O3` and `-O4`.

3.5 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.

3.6 Simulators

CTS 4912/4923: The socket connection between the simulators (`casim` and `isim`) and the runtime components (`csreset`, `csrunc`) 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.

This can be fixed at boot time by adding the following lines to the end of the file `/etc/sysctl.conf`:

```
# disable tcp abc
net.ipv4.tcp_abc = 0
```

3.7 Instruction Set

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.

3.8 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.

3.9 CSPX

CTS 5771: The compiler emits an error when using the CSPX `rpc_export` pragma if the return type of the function is `void`. The filename and line number information for the error are incorrect and garbled.

The solution to this problem is always to make `rpc_export` functions return a base type (such as `int`).

ClearSpeed Technology, Inc.
800 West El Camino Real, Suite 180
Mountain View, CA 94040
United States of America

Tel: +1 650 943 2329
Fax: +1 650 962 1188

Email: 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 2008. All rights reserved.

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.