Release Notes for VisualDSP++ 5.1.0

VisualDSP++® 5.1.0 Release Notes

Revision 1.0

September 2013

Nomenclature

VisualDSP++ is upgraded from 5.0 to 5.1.0 to reflect support for Windows 8, along with other important updates.

Release Notes

Cumulative release notes for previous updates to VisualDSP++ 5.0 can be found at www.analog.com/VDSPUpdate in the section ***Update 10.1 - August 2012***.

Identifying Your VisualDSP++ Version

The VisualDSP++ release and update number can be found in 2 locations:

1. In the Control Panel, open the Add/Remove Programs applet.
2. In the VisualDSP++ Integrated Development and Debug Environment (IDDE), select Help About VisualDSP++.

Definitions

This section provides definitions for terminology relating to VisualDSP++ and this document.

TAR – Tools Anomaly Report

Tools Anomaly Report, or TAR, has been the terminology used for a defect report for VisualDSP++. Each TAR is automatically assigned a unique number upon creation, aiding in the logging, tracking and closure of the anomaly.

Analog Devices have renamed the TAR designation, to more clearly identify the affected product family. Accordingly, TARs that affect VisualDSP++ are now renamed and renumbered with a VDSP- prefix. For example, TAR-xxxxx would now be VDSP-yyyyy.

New Silicon Support

VisualDSP++ updates often include support for new processors, new silicon revisions for existing processors and new EZ-KIT Lite®, EZ-Board® and EZ-Extender® evaluation systems. In order to support these, minor revisions are made to the tool chain and additional system services and device drivers need to be added. This section describes the new support available in this update.

No new processors are supported in this release.

New Silicon Revisions

There are no new revisions for generally available silicon supported by this revision.

Silicon Anomaly Workarounds

Anomaly workaround information is available in the online help: Select Help > Contents> Graphical Environment > Silicon Anomaly Support > Silicon Anomalies Tools Support and then click the appropriate processor series.

The following table lists new silicon anomaly workaround support.

Silicon Errata	Parts	Workaround Support Added	Further information
15000023: A three column data access over DM bus immediately following an indirect delayed branch (db) may not work as expected in VISA mode	All SHARC 214xx parts and revisions.	The compiler will work around the issue. The assembler will detect and warn about potential issues. Run-time libraries have been rebuilt with the workaround enabled	See System\ArchDef\SHARC- 2146X-anomaly.xml

New Evaluation Board Support

Support has been added for the following new evaluation boards and evaluation board revisions.

No new evaluation boards are supported in this release.

What's New in VisualDSP++ 5.1.0?

Several enhancements have been added to VisualDSP++ 5.1.0. This section provides an overview for the new features.

Windows 8 Support

The list of supported Windows 8 editions are:

- Windows 8 (32-bit and 64-bit)
- Windows 8 Professional (32-bit and 64-bit)
- Windows 8 Enterprise (32-bit and 64-bit)

New Features in the Loaders / Loader Collateral

ADSP-21371 Specific Loader Kernels

The ADSP-21371 is now supported by 371 specific kernels installed to:

- 213xx/ldr/371_prom.dxe
- 213xx/ldr/371_spi.dxe

with sources and projects available:

- 213xx/ldr/371_prom
- 213xx/ldr/371_spi

The Loader Property Page has been updated to present the 371 specific kernels as the default for the ADSP-21371.

Likewise the loader has been updated to default the ADSP-21371 to a 371 kernel DXE file if it encounters a command-line that requires a default kernel.

Note that this is a change in default for the ADSP-21371 from the VisualDSP++ 5.0 updates. If you have an existing build that relied on the previous default, you may need to update your project.

The ADSP-21375 boot kernel fails when used with a ADSP-21371 application that uses external memory. The ADSP-21371 supports 32-bit external port whereas the ADSP-21375 supports 16-bit external interface.

Byte Format for ADSP-214xx Non-Bootable Loader Files

An additional format is available in the SHARC loader when creating non-bootable loader files for the ADSP-214xx.

Byte format is provided for use with the *-splitter SectionName* switch. It can be set via Additional Options on the Loader Property Page.

The new switches are *-fBYTE* and *-u value*.

-fBYTE

Specify format BYTE (for -splitter only)

-u value

Specify a value for the content of the user flag field in a BYTE format header.
Value range is 0x0 - 0xFF. If no *-u* switch, user flag field is zero.
For use with -fBYTE and -splitter switches only.

-splitter SectionName

Extracts content of section named “SectionName” and generates a LDR file in raw format.

- The section name is a required argument for *–splitter*. It specifies what section the loader is to extract content from. All other sections are ignored.
- The -splitter switch results in a "raw format" LDR file containing the content of the section.

  Since this is LDR output that will not be processed by the kernel, the loader prevents inclusion of the kernel and zero block creation by setting the following:

  -NoKernel [0,0]

  -NoZeroBlock
- The switch is available for the ADSP-214xx processor family for short-word or normal-word sections.

BYTE File Format Example

The non-bootable file in BYTE format has these characteristics:

- A one-line header
- A block of one or more lines of section data from the .dxe file
- A zero header that signals the end of the file

The following table shows an example of a byte-format file created by the loader utility.

The Additional Options on the Loader Property Page for this example:

-splitter my_seg_swco -fBYTE -u 0xAB

Field	Purpose
200688AB0012435D00000768	Example header record (the first line of file)
20	Width of address and length fields (in bits) Addresses are 32-bit width in this example.

06	Reserved field in use by ADI for versioning. The SHARC loader is currently setting this to Version 6. Note: The elfspl21k utility is currently setting this to Version 5 for .stf fi les
88	Flags (88 = SW, 80 = PM, 00 = DM) This shows a build with *-splitter SectionName* that is a SW section.
AB	User-defined flags (loaded with *-u value* switch). This build shows the result of a build with -u 0xAB. If no -u switch is present, the user-defined flag field is 00.
0012435D	Start address of the data block
00000768	Number of bytes of data that follow
0f14 000b 2001 0fb4 0000 …	Lines of section data. The *-hostwidth [8|16|32]* switch determines the number of bytes per line. This example shows the content from a SW section for a build using -hostwidth 16.
000000000000000000000000	Example header record (footer) that signals end-of-file

Changed Functionality

2146x / 2147x / 2148x Loader Kernels

The ADSP-2146x / 2147x/ 2148x loader kernels were updated to workaround PLL anomaly 15000020.

Critical Fixes / Changes

This section highlights significant changes due to software anomaly fixes or functional changes.

"PEyRegs" is now a supported register set for "pragma regs_clobbered", the "-section" command line switch and inline "asm" statements

The register set "PEyRegs" (covering S0 -> S15 on SHARC SIMD processors) can now be used with "pragma regs_clobbered", the "-section" command line switch and in the clobber field of inline asm statements.Note that when "-reserve PEyRegs" is used, SIMD code generation will be disabled. Using the compiler switch "-no-simd" will not automatically reserve the PEy registers.

The Blackfin sqrtf function now returns (Not-A-Number) NaN for negative inputs

The three floating-point square root functions sqrtf, sqrtd, and sqrt return 0.0 whenever the input argument is not a positive value. This behavior is in accordance with their description in the "C/C++ Compiler and Library Manual" which states that "The sqrt functions return a zero if the input argument is negative." However the IEEE floating-point standard ANSI/IEEE Std 754-1985 defines the square root of a negative number as an invalid operation and the result therefore should be a NaN.

As the run-time libraries conform to the IEEE standard, the floating-point square root functions should return a NaN whenever the input argument is invalid.

The coherence functions implement an approximation of the correct formula which may lead to misleading results

The coherence functions either compute the coherence of an input signal with itself (auto-coherence) or they compute the coherence between two input signals (cross-coherence). The implementations of these functions are based on an approximation of the correct formula, which could lead to misleading or incorrect results.

Taking the cross-coherence functions as an example, their implementation is based on the formula:

```
coherence[k] = (1/n) * Sum(i=0 to n-k-1) (x[i]*y[i+k]) - (x * y)
```

where k = 0 to lags-1

n = samples

–

and x is the mean value of x

–

y is the mean value of y

The divisor is n and the computed means of the input signals x and y have n terms, but the summation only has n-k terms. This means that the formula used may lead to misleading or wrong answers if the number of lags is not significantly smaller than the number of samples.

The correct formula for cross-coherence is:

```
coherence[k] = ((1/(n-k)) * Sum(i=0 to n-k-1) (x[i] * y[i+k])) -
                          (((1/(n-k)) * Sum(i=0 to n-k-1) (x[i])) *
                           ((1/(n-k)) * Sum(i=k to n-1) (y[i])))
```

where k = 0 to lags-1
n = samples

There is a similar issue with the auto-coherence functions. They have been implemented using the formula:

```
coherence[k] = (1/n) * Sum(i=0 to n-k-1) (x[i]*x[i+k]) - (x * x)
```

where k = 0 to lags-1

n = samples

–

and x is the mean value of x

But they should be based on the following formula:

```
coherence[k] = ((1/(n-k)) * Sum(i=0 to n-k-1) (x[i] * x[i+k])) -
                          (((1/(n-k)) * Sum(i=0 to n-k-1) (x[i])) *
                           ((1/(n-k)) * Sum(i=k to n-1) (x[i])))
```

where k = 0 to lags-1

n = samples

The DSP run-time library provides the following coherence functions:

- autocohf
- autocoh
- autocohd
- autocoh_fr16 (**Blackfin only**)
- autocoh_fr32 (**Blackfin only**)
- crosscohf
- crosscoh
- crosscohd
- crosscoh_fr16 (**Blackfin only**)
- crosscoh_fr32 (**Blackfin only**)

DLLEN must be set to 0 in DDR2CTL3 register on SHARC 2146x processors

The DDR2CTL3 register controls the memory setting. To enable the memory's DLL (delay locked loop) bit 0 of the DDR2CTL3 register should be cleared. Previously, the macro definition to control the DLL enable/disable is as follows:

```
#define DDR2DLLEN (BIT_0) /* DLL enable */
```

where BIT_0 is defined as (0x00000001). This definition was confusing as it suggested that enabling DLL is achieved by setting bit 0 of the register, rather than clearing it.

New 2146x DDR2 macros

The following new bit position macros have been added to def21469.h:

Register	Macro	Description
DDR2PADCTL0	DATA_PWD	Data Pad Receiver Power Down
DDR2PADCTL0	DQS_PWD	DQS Pad Receiver Power Down
DDR2PADCTL0	DDR2CLK_PWD	Clock Pad Receiver Power Down
DDR2PADCTL1	ADDR_PWD	Address Pad Receiver Power Down
DDR2PADCTL1	CMD_PWD	Command Pad Receiver Power Down

Changes to the behavior of SHARC compiler builtin functions __builtin_lllleftz and __builtin_lllefto

In previous releases of VisualDSP++, the compiler builtin function "__builtin_llleftz" generated code that used the SHARC processor's "lefto" instruction, and "__builtin_lllefto" generated code that used the "leftz" instruction. This issue has been addressed, and any code that relies on the previous behavior should be changed.

Key	Family	Component/s	Summary
VDSP-10058	SHARC - ADSP-2136x	Assembler	Certain sequences incorrectly cause warnings for anomalies 07000009 (ea2501) and/or 07000010 (ea2504)
VDSP-24952	Blackfin - All	Compiler	ccompose_fr16 composes operands in the wrong order
VDSP-21735	SHARC - All	Compiler	Compiler does not honour "pragma no_vectorization" when compiling main()
VDSP-17089	SHARC - ADSP-2116x, SHARC - ADSP-2126x, SHARC - ADSP-2136x, SHARC - ADSP-2137x, SHARC - ADSP-2146x, SHARC - ADSP-2147x, SHARC - ADSP-2148x	Compiler	Compiler can use PEy registers when -no-simd is used
VDSP-20348	Blackfin - All	Compiler	Registers can be corrupted by the exceptions handling mechanism
VDSP-17341	SHARC - All	Compiler	-extra-precision switch should be deprecated
VDSP-21609	SHARC - All	Compiler	Compiler builtin functions __builtin_lllleftz and __builtin_lllefto generate wrong code
VDSP-24938	SHARC - ADSP-2136x, SHARC - ADSP-2137x, SHARC - ADSP-2146x, SHARC - ADSP-2147x, SHARC - ADSP-2148x	Emulator	External bus hang conditions are not detected by emulator
VDSP-24931	SHARC - ADSP-2146x, SHARC - ADSP-2147x, SHARC - ADSP-2148x	Emulator	214xx disassembly window does not display instructions correctly for external memory
VDSP-20424	Blackfin - All	IDDE	Long thread names may cause VDK History window crash
VDSP-21983	SHARC - ADSP-2137x, SHARC - ADSP-2146x, SHARC - ADSP-2147x, SHARC - ADSP-2148x	Loader	Newer SHARC parts have incorrect default kernel settings in absence of -l option
VDSP-21908	SHARC - ADSP-2146x, SHARC - ADSP-2147x, SHARC - ADSP-2148x	Loader	PLL anomaly workaround 15000020 needed in 2146x / 2147x / 2148x kernels
VDSP-25107	Blackfin - All	Run Time Libraries	tolower and toupper may return the wrong result
VDSP-22886	Blackfin - All	Run Time Libraries	modff does not handle a NaN correctly

VDSP-9183	SHARC - All	Run Time Libraries	CLONE - favgd is not safe against NaNs or Infs
VDSP-25546	SHARC - All	Run Time Libraries	favgd() can return a wrong result
VDSP-21406	SHARC - ADSP-2146x	Run Time Libraries	ADSP-21469 - DLLEN must be set to 0 in DDR2CTL3 register
VDSP-9469	Blackfin - All	Run Time Libraries	sqrtf, sqrtd, and sqrt return zero when their argument is negative
VDSP-10207	SHARC - All	Run Time Libraries	For some data the powf library function may return Infinity instead of 0.0
VDSP-14201	Blackfin - All	Run Time Libraries	The floating-point sqrt functions do not calculate the square root of Infinity as Infinity
VDSP-25076	SHARC - All	Run Time Libraries	CB7 isn't being set on startup so latched stack overflow interrupts won't get handled
VDSP-15997	SHARC - All	Run Time Libraries	The coherence functions implement an approximation of the correct formula which may lead to misleading results
VDSP-14558	Blackfin - All	Run Time Libraries	CLONE - The coherence functions implement an approximation of the correct formula which may lead to misleading results
VDSP-13203	TigerSHARC - All	Run Time Libraries	CLONE - The coherence functions implement an approximation of the correct formula which may lead to misleading results
VDSP-18898	SHARC - All, SHARC - ADSP-2146x, SHARC - ADSP-2147x, SHARC - ADSP-2148x	Run Time Libraries	Interrupt set-up functions can trigger hardware ADSP-214xx anomaly 15000024
VDSP-24980	SHARC - All	Run Time Libraries	Code compiled with Update 9 and containing calls to memcpy() may not work with Update 10
VDSP-25032	SHARC - ADSP-2106x	Run Time Libraries	heaps mistakenly identified as PM for non SIMD-SHARCs
VDSP-13798	Blackfin - ADSP-BF53x	Run Time Libraries	ADSP-BF535 USE_L2_STACK support is problematic with CPLBs enabled