VisualDSP++ 4.5 (Update 7) Release Notes

The following release note concerns Update 7 to the VisualDSP++ 4.5 release. This release is inclusive of previous Updates. The release notes for past Updates are appended to the end of this release note.

Identifying Which Update Is Currently Installed on Your System

The Update level is identified in three places:

1. The Add/Remove Programs Control Panel entry for VisualDSP++ 4.5.
2. The VisualDSP++ GUI's About box, located at "Help" > "About VisualDSP++".
3. In the file …\System\VisualDSP.ini, in the ProductName key.

Installing an Update

The procedure for installing an Update to VisualDSP++ is described below. Note that with VisualDSP++'s support for installing multiple instances of itself, it is possible to "trial" an Update in a new directory before switching over your "golden" tools installation to the Update.

1. Use the Start Menu to navigate to VisualDSP++'s "Maintain this Installation" item.
2. Select "Go to the Analog Devices website" and click "Next". This will launch your web browser and navigate it to the proper URL to download Updates from.
3. Download the VisualDSP++ Update file (.VDU) of interest to your hard disk. Note that these files have a .VDU file extension and cannot be executed directly.
4. Navigate to "Maintain this Installation" again. If you have multiple installations of VisualDSP++ on your computer, be doubly sure you are navigating to the installation you wish to Update.
5. Select "Apply a downloaded Update" and click "Next". Click the "…" browser button and navigate to the .VDU file that you downloaded in step 3. Click "OK", then "Next".
6. Follow the on-screen prompts to complete the installation of the Update.

New Hardware Support

VisualDSP++ updates often include support for new silicon revisions for existing processors and new EZ-KIT Lite® evaluation systems. This section describes the new support available in this update.

New Processors and Revisions Support

The Product Bulletin contains the list of new processors available with VisualDSP++ 4.5. Refer to the processor's data sheet and hardware reference manuals for information on system configuration, peripherals, registers, and operating modes.

Support will be provided for the following silicon revisions to existing Blackfin® processors with Update 7:

- ADSP-BF531 silicon revision 0.6
- ADSP-BF532 silicon revision 0.6
- ADSP-BF533 silicon revision 0.6

- ADSP-BF538 silicon revision 0.5
- ADSP-BF539 silicon revision 0.5

There are no new silicon revisions to existing SHARC® or TigerSHARC® processors with Update 7.

New Features

VisualDSP++ updates occasionally include new features. This section describes the new support available in this update.

Upgrade to lwIP 1.3

The lwIP Ethernet stack has been upgraded from revision 1.2 to 1.3. Please refer to lwIP homepage for more information (http://savannah.nongnu.org/projects/lwip/) and the associated Wiki (http://lwip.scribblewiki.com/LwIP_Main_Page).

Add Multicast Support to Ethernet Drivers

Multicast support has been added to the Ethernet drivers.

Critical Fixes/Changes

When addressing issues, we attempt to make any changes backward compatible with existing projects. However, depending on the nature of the issue, compatibility cannot always be maintained. This section highlights any changes in the Update that may require the modification of "working" projects or otherwise influence existing behavior.

Deprecate Support for ADSP-BF52x and ADSP-BF54x Processor Families

The initial support on VisualDSP++ 4.5 for the new ADSP-BF52x and ADSP-BF54x processor families has long since been superseded by the VisualDSP++ 5.0 support for these processor families. Support for these processor families on VisualDSP++ 4.5 will be discontinued. All projects using any of the ADSP-BF52x or ADSP-BF54x should be migrated to the latest update of VisualDSP++ 5.0 as soon as possible.

Use Linker Elimination Options for ADSP-21371/ADSP-21375

The workaround for silicon anomaly 09000011 generates unused assembly code. To avoid linking this unused assembly code, turn on linker elimination:

1. Select *Project – Project Options* from the VisualDSP++ menu.
2. Select *Link – Elimination*
3. Check the box *Eliminate unused objects*
4. Click *OK*

Customized Linker Definition Files (LDFs) Change for ADSP-21371/ADSP-21375

Customers using the ADSP-21371 and ADSP-21375 that have non-default customized LDFs may need to make a modification to their LDFs. If the workaround for silicon anomaly 09000011 is required, trampoline code (see definition in *Silicon Anomaly Workarounds*) is placed in section *seg_int_code*. If not already done, the section *seg_int_code* should be mapped to internal memory code.

VDK Internal Memory Code Size Increase for ADSP-21371/ADSP-21375

As part of the workaround for silicon anomaly 09000011, the time-critical part of VDK has been mapped to *seg_int_code* instead of *seg_pmco*. Customers using VDK with the ADSP-21371 and ADSP-21375 will see an increase in the size of the code that is required to be in internal memory.

VDK LDF Change for ADSP-21371/ADSP-21375

Customers that use VDK with the ADSP-21371 and ADSP-21375 must change their LDFs to link TMK-2137x.dlb instead of TMK-213xx.dlb.

TAR 35198 - VDK Thread Stack Space Reduced on TigerSHARC

An anomaly has been identified in VisualDSP++ for TigerSHARC where the VDK thread stack pointers are not configured correctly during thread creation (TAR 35194). Thread stack space is allocated and the stack pointers are configured so that they point to the end of the stack allocation spaces, as the stacks grow from high to low memory. The issue is that the stack pointers are placed too close to the end of each stack allocation, resulting in up to 8 words of data being corrupted before the start of each thread stack space (higher memory). The fix correctly configures the stack pointers so that they are further into the thread stack allocation space on creation, 8 words further-in for the J stack and 4 for the K stack. This effectively means that the stacks for each thread will reach their maximum limit slightly sooner than with previous releases.

Silicon Anomaly Workarounds

Silicon Anomaly 09000011 (ADSP-2137x)

"Indirect Branches from External to Internal Memory may corrupt the Instruction Cache."

Workarounds for this anomaly have been implemented in the assembler; the default behavior is to apply a workaround. The compiler relies upon the default behavior of the assembler to apply the workarounds. The runtime libraries and VDK have been rebuilt to avoid the anomaly or apply the workarounds, except for code that must be mapped to internal memory. One of the workarounds used by the assembler generates new code in section "seg_int_code" that must be mapped to internal memory. The default Linker Description File (LDF) provided in VisualDSP++ does this already; projects with individual LDFs may require modification to map this section.

The assembler will provide informational messages for each instance of an applied workaround and to notify the user about code generated to seg_int_code. When some condition prevents the assembler from applying the workaround, the assembler will produce a descriptive error message instead. The compiler driver will suppress the informational messages generated by the assembler about the workarounds. The assembler will not apply workarounds to code defined in a section named "seg_int_code".

If a user prefers to adjust their code to avoid the anomaly, specifying "-anomaly-detect 09000011" will cause the assembler to instead produce a warning for each instance of a problematic branch instruction. Specifying "-no-anomaly-workaround 09000011" will suppress all assembler activity for this anomaly.

The assembler will apply one of two identified workarounds depending upon the specific instruction containing an indirect branch. One form of workaround avoids the anomaly by inserting a PC-relative branch around the potentially improperly cached location and inserting a NOP instruction at that location, thus preventing execution of an instruction at the location that could be improperly cached due to the anomaly, at the cost of two words of memory and a branch execution. Each instance of the workaround will produce a message ea2517:

```
[Informational ea2517] ".\BranchAroundCache.asm":24 Applied Workaround
for Hardware Anomaly 09000011
Inserted "JUMP(PC,2); nop;" after the instruction following the indirect
branch.
```

The second workaround replaces the problematic indirect branch with an indirect branch to a "trampoline" (see definition below) JUMP instruction which will use the same index and modify register as the replaced branch to jump to the original

destination of that replaced instruction. To avoid the anomaly, the trampoline JUMP must execute from internal memory. For the simplest type 9 instructions, this workaround avoids the cache corruption at the cost in execution of an additional branch and a maximum of one word of memory per index and modify register pair used in branch instructions. Each instance of the trampoline workaround will produce a message ea2518:

```
[Informational ea2518] ".\myFile.asm":43 Applied Workaround for Hardware
Anomaly 09000011
converted the indirect branch to a direct branch to trampoline at label
__JUMP_m08i08__
```

The assembler will add the trampoline instructions to the section “seg_int_code’; it will generate that section if necessary. The assembler will emit message ea2519 identifying the trampolines generated. For the message below, the source code contained indirect branch instructions using only I8 and m8:

```
[Informational ea2519]  Trampolines generated for Hardware Anomaly
09000011
section name: seg_int_code;
trampolines:
__JUMP_m08i08__
```

Each object file in which trampoline workarounds have been applied will contain a section seg_int_code providing the trampolines for the code in that object. Where different objects each contain the same trampoline, the linker will resolve all references to a single instance of the trampoline.

When the assembler fails to apply the workaround, it will produce message 2516. The following series of instructions illustrates one case that will produce this message:

```
CALL (M14,I12) (DB);
i14 = DM(i6,m7);
m7 = PM(i12, m14); // postmodify.
```

When the file is assembled with the workaround enabled, the assembler will produce the following message specifying why neither workaround could be applied:

```
[Error ea2516] ".\trampolineDBerrors.asm":69 Workaround for Hardware
Anomaly 09000011 not applied:
Trampoline cannot be used because a delay slot instruction modifies a DAG
register used in the branch instruction.
Branch around improperly cached location cannot be used because delayed
branch call: cannot insert jump around third location after the call.
```

For more information about this silicon anomaly, please refer to the latest ADSP-21371/ADSP-21375 Silicon Anomaly List.

Trampoline

A trampoline solution is replacing a problematic branch instruction with a direct branch to a location in internal memory containing a branch that uses the index and modify registers of the original, replaced branch instruction.

Tools Anomalies Addressed

The following table is a list of the tools anomalies addressed in this Update. Details on any particular anomaly can be found on the Tools Anomaly web page. The URL is:

http://www.analog.com/processors/tools/anomalies

Processor Family	Tools Anomaly Report #	Tool	Description
Blackfin	36094	ADspCommon XML Files	compiler XML enables workaround for 05-00-0283 when unnecessary
Blackfin	34208	Compiler	Compiler cangenerate bad debug information for virtual functions
Blackfin	35464	Compiler	shl() instr being replaced with non-saturated folded value -O
Blackfin	34799	Run Time Libraries	LIBDSP functions rsqrtf, rsqrtd incorrect for input power of 4
Blackfin	34851	Run Time Libraries	disable_data_cache() does not work
Blackfin	36111	Run Time Libraries	default multi-thread CRT objects may result in CPLB misses
Blackfin	35001	System Services	adi_pwr_Init hangs when data cache is enabled
Blackfin	36095	VDK	VDK::Yield() does not reset timeslice
SHARC	35278	Compiler	#pragma interrupt_complete_nesting causes unsafe code
SHARC	34918	Emulator	VisualDSP++ disconnects if Sport DMA Addressing register window opened
SHARC	35012	Emulator	Cannot load 16-bit external memory on 2126x
SHARC	34697	Run Time Libraries	sinf may return poor results for inputs close to a 2*PI multiple
TigerSHARC	34842	Compiler	Compiler incorrectly omits jump over loop

VisualDSP++ 4.5 (Update 6) Release Notes

The following release note concerns Update 6 to the VisualDSP++ 4.5 release. This release is inclusive of previous Updates. The release notes for past Updates are appended to the end of this release note.

Update Name Change

Consistent with the VisualDSP++ 5.0 release, updates are now specified by the number of the update. This change helps users and our support team easily identify the installed update. As the June 2007 Update was the 5th update to VisualDSP++ 4.5, this is the 6th Update or Update 6.

Identifying Which Update Is Currently Installed on Your System

The Update level is identified in three places:

4. The Add/Remove Programs Control Panel entry for VisualDSP++ 4.5.
5. The VisualDSP++ GUI's About box, located at "Help" > "About VisualDSP++".
6. In the file …`\System\VisualDSP.ini`, in the `ProductName` key.

Installing an Update

The procedure for installing an Update to VisualDSP++ is described below. Note that with VisualDSP++'s support for installing multiple instances of itself, it is possible to "trial" an Update in a new directory before switching over your "golden" tools installation to the Update.

7. Use the Start Menu to navigate to VisualDSP++'s "Maintain this Installation" item.
8. Select "Go to the Analog Devices website" and click "Next". This will launch your web browser and navigate it to the proper URL to download Updates from.
9. Download the VisualDSP++ Update file (.VDU) of interest to your hard disk. Note that these files have a .VDU file extension and cannot be executed directly.
10. Navigate to "Maintain this Installation" again. If you have multiple installations of VisualDSP++ on your computer, be doubly sure you are navigating to the installation you wish to Update.
11. Select "Apply a downloaded Update" and click "Next". Click the "…" browser button and navigate to the .VDU file that you downloaded in step 3. Click "OK", then "Next".
12. Follow the on-screen prompts to complete the installation of the Update.

Significant Additions

The primary purpose of VisualDSP++ Updates is to address problems and stabilize the release. Significant new functionality is not expected to be introduced in an Update. However, incremental support (i.e., emulation, example programs, header files, default LDF, errata accommodations, EZ-KIT Lite software, etc.) for new semiconductor products will be added as these products become available and gain support within the VisualDSP++ tools.

In this release:

1. Blackfin ADSP-21367/8/9
   Support for silicon revision 0.2 has been added.

2. Addition of __VISUALDSPVERSION__ Predefined Macro

   The __VISUALDSPVERSION__ macro enables code to be configured for multiple versions of VisualDSP++. It was introduced in VisualDSP++ 5.0 and is available in VisualDSP++ 4.5 beginning at Update 6.

   The compilers and assemblers predefine the VisualDSP++ version as:

   ```
   -D__VISUALDSPVERSION__=0xMMmmUUxx
   ```

   Where

   ```
   MM    VersionMajor
   mm    VersionMinor
   UU    VersionPatch (Update)
   xx    Reserved for Future Use (always 00 initially)
   ```

   Examples:

`0x04050600`	VisualDSP++ 4.5 Update 6
`0x05000100`	VisualDSP++ 5.0 Update 1
`0xffffffff`	Unexpected problem, unable to determine version
`!defined(__VISUALDSPVERSION__)`	VisualDSP++ 4.5 Update 5 or earlier (not available)

   Here is a C language example:

   ```
   #if __VISUALDSPVERSION__ >= 0x04050600
   /* Code relying on feature in VisualDSP++ 4.5 Update 6 or later */
   #elif
   /* Legacy code */
   #endif
   ```

   Here is an Assembly example:

   ```
   #if __VISUALDSPVERSION__ == 0x04050600
   .VAR VersionBuildString[] = 'VisualDSP++ 4.5 Update 6 Build';
   #endif
   ```

Changes to Existing Behaviors, Projects, and Source Code

When addressing problems, we attempt to make any changes backward compatible with existing projects. However, depending on the nature of a problem, compatibility issues are sometimes unavoidable. This section highlights any changes in the Update that may require the modification of “working” projects or otherwise influence existing behavior.

In this release:

1. TAR 33304: ADSP-21367/8/9, 21371, 21375 macro SDCTL definition changed

   The definition of the macro defined for bit 20 in the SDCTL register has been changed in the following files from “FAR” to “FARF”

2. Silicon anomaly 05-00-0371 workaround: Possible RETS Register Corruption when Subroutine Is under 5 Cycles in Duration

The Blackfin C/C++ compiler has been enhanced to include workarounds for anomaly 05-00-0371 "Possible RETS Register Corruption when Subroutine Is under 5 Cycles in Duration". The anomaly happens (very rarely) when calling functions with an RTS within 5 instructions from the start of the function. The C/C++ compiler workaround is to avoid generating such functions in the assembly it produces, these would typically result from stub function code. The workaround involves inserting NOP instructions or an unconditional JUMP instruction before the RTS. The JUMP workaround variant is used when optimizing for code-size (-Os) and there would be more than two NOPs otherwise required.

To enable this compiler workaround manually the -workaround avoid-quick-rts-371 switch can be used. When the workaround is enabled the macro __WORKAROUND_AVOID_QUICK_RTS_371 is defined at compile, assemble and link stages.

The runtime libraries and VDK support linked when building for impacted parts and silicon revisions have been modified to avoid the anomaly.

3. Silicon anomaly 05-00-0323 workaround: Erroneous GPIO Flag Pin Operations under Specific Sequences

   Include file sys/05000323.h is now supplied with VisualDSP++ 4.5. It contains a group of macros for reading and writing MMRs applicable to this anomaly; if the anomaly applies for the current value of the silicon revision of your target, these macros will ensure that the read or write is safe against anomaly 05-00-0323. When building for parts and silicon revisions that require the anomaly 05-00-0323 workaround, the macro __WORKAROUND_FLAGS_MMR_ANOM_323 is defined at compile, assemble, and link stages. To enable the workaround manually you can define use the -D__WORKAROUND_FLAGS_MMR_ANOM_323 switch. See comments in the new file (<VisualDSP++ 4.5 Install>\Blackfin\include\sys\05000323.h) for further details.

Problems Addressed

The following table is a list of the problems addressed in this Update. Details on any particular problem can be found on the Tools Anomaly web page. The URL is:

http://www.analog.com/processors/tools/anomalies

Processor Family	Problem Number	Tool	Description
Blackfin	32910	Compiler	Buffer overrun detected error message from linker
Blackfin	34040	LDF	-p1 and -p2 do not work with default LDFs
Blackfin	32911	Run Time Libraries	mulfl64.asm in release not same as used to build library
Blackfin	34139	Run Time Libraries	BF561 - Memory initializer will not initialize ext SDRAM
SHARC	33621	Simulator	reg modify then write to ext mem writes old reg value
SHARC	33780	Run Time Libraries	External memory functions can fail when optimization is used
SHARC	33781	Simulator	read from initialized data in 16 or 8 bit memory reads wrong value
SHARC	33788	Simulator	Write to an address in 16 or 8 bit external memory cannot be read.
SHARC	33304	Run Time Libraries	def files macro FAR changing to FARF

VisualDSP++ 4.5 (Updated June 2007) Release Note

The following release note concerns the June 2007 Update to the VisualDSP++ 4.5 release. This release is inclusive of previous Updates. The release notes for past Updates are appended to the end of this release note.

Identifying Which Update Is Currently Installed on Your System

The Update level is identified in three places:

7. The Add/Remove Programs Control Panel entry for VisualDSP++ 4.5.
8. The VisualDSP++ GUI's About box, located at "Help" > "About VisualDSP++".
9. In the file …`\System\VisualDSP.ini`, in the `ProductName` key.

Installing an Update

The procedure for installing an Update to VisualDSP++ is described below. Note that with VisualDSP++'s support for installing multiple instances of itself, it is possible to "trial" an Update in a new directory before switching over your "golden" tools installation to the Update.

13. Use the Start Menu to navigate to VisualDSP++'s "Maintain this Installation" item.
14. Select "Go to the Analog Devices website" and click "Next". This will launch your web browser and navigate it to the proper URL to download Updates from.
15. Download the VisualDSP++ Update file (.VDU) of interest to your hard disk. Note that these files have a .VDU file extension and cannot be executed directly.
16. Navigate to "Maintain this Installation" again. If you have multiple installations of VisualDSP++ on your computer, be doubly sure you are navigating to the installation you wish to Update.
17. Select "Apply a downloaded Update" and click "Next". Click the "…" browser button and navigate to the .VDU file that you downloaded in step 3. Click "OK", then "Next".
18. Follow the on-screen prompts to complete the installation of the Update.

Significant Additions

The primary purpose of VisualDSP++ Updates is to address problems and stabilize the release. Significant new functionality is not expected to be introduced in an Update. However, incremental support (i.e., emulation, example programs, header files, default LDF, errata accommodations, EZ-KIT Lite software, etc.) for new semiconductor products will be added as these products become available and gain support within the VisualDSP++ tools.

In this release:

Blackfin ADSP-BF54x:

As in the previous VisualDSP++ 4.5 Update, the Blackfin processor ADSP-BF54x family has emulation, compiler, assembler, linker and IDDE / debugging support, with additional bug fixes available as of this Update. The loader and System Services are not available in this Update. The ADSP-BF54x processors, including loader and System Services, are fully supported in the upcoming major release.

Changes to Existing Behaviors, Projects, and Source Code

When addressing problems, we attempt to make any changes backward compatible with existing projects. However, depending on the nature of a problem, compatibility issues are sometimes unavoidable. This section highlights any changes in the Update that may require the modification of "working" projects or otherwise influence existing behavior.

In this release:

1. TAR 31816: Incorrect memory mapping for ADSP-21375

   The memory map for the ADSP-21375 has been corrected throughout the tools, including the linker and the default LDFs. There are three consequences to these changes:

   1 -- Any LDF that is heavily derived from a default LDF of a version of VisualDSP++ prior to this Update may result in linker error el2011 "Invalid memory range and/or width for memory" when linking. In this situation, the LDF must be corrected to reflect the actual memory map of the ADSP-21375 target.

   2 -- Any application that uses the default LDF and more memory than available on the ADSP-21375 part memory map will cause linker errors li1040 "Out of memory in output section". In previous Updates the link of such applications may have succeeded. In this situation it will be necessary to reduce memory usage or build for a part with more memory available.

   3 – Out of the box, the VDK-21375.ldf will get a linker error li1040 for "Out of memory in output section 'seg_pmco' in processor". VDK is too large for the ADSP-21375 to fit in internal memory. To use VDK in an ADSP-21375 processor, external memory must be used.

   The data sheets for these parts has corrected memory map information and can be downloaded from www.analog.com by doing a search for the required part number (e.g. ADSP-21375).

2. The SPI flash on the ADSP-21262 and ADSP-21364 EZ-KIT Lite has been changed from the Atmel AT25F2048 to the STMicroelectronics M25P20. The flash programmer and associated device drivers have been updated accordingly. The flash programmer will automatically determine which driver to use, no special intervention on the part of the user should be required. However, any user application that is heavily derived from an older version of the flash programmer and/or underlying driver may need to be updated to ensure correct operation on newer EZ-KIT Lite boards.

   The following examples are no longer applicable to the newer boards:
   <install-dir>\212xx\Examples\ADSP-21262 EZ-KIT Lite\Atmel SPI Flash Programmer (ASM)
   <install-dir>\213xx\Examples\ADSP-21364 EZ-KIT Lite\Atmel SPI Flash Programmer (ASM)

   The following new example is included.
   <install-dir>\212xx\Examples\ADSP-21262 EZ-KIT Lite\STMicro SPI Flash Programmer (ASM)

   There is no ADSP-21364 STMicro SPI Flash Programmer (ASM) example available at this time.

3. Because of difficulties found with the reschedule interrupt in SHARC processors, VDK now reserves the SFT2I and SFT3I interrupts for the reschedule interrupt. These interrupts cannot be used in any other manner.

4. TAR 32344 : Former workaround for 05-00-0311 is not safe

New information regarding anomaly 05-00-0311 has moved the scope of this anomaly beyond the realm of a VisualDSP++ Blackfin compiler workaround and into the region of application-specific behavior.

In the VisualDSP++ 4.5 February 2007 Update the Blackfin compiler, runtime, VDK and SSL libraries automatically included a new workaround for hardware anomaly 05-00-0311. The VisualDSP++ 4.5 February 2007 Update C/C++ compiler also automatically enabled this workaround when building for parts and silicon revisions that require it.

New information about anomaly 05-00-0311 reveals that it is necessary to temporarily disable interrupts during MMR accesses, which is a decision the compiler should not be making as it could be disabling interrupts for far too long or during a critical moment when the code relies on receiving one. For this reason the implementation of the workaround has been changed for the VisualDSP++ 4.5 June 2007 Update.

In the VisualDSP++ 4.5 June 2007 Update the Blackfin compiler, runtime, VDK and SSL libraries no longer workaround hardware anomaly 05-00-0311. Instead, an include file called sys/05000311.h is supplied and contains a group of macros for reading and writing the MMRs; if the anomaly applies for the current value of the silicon revision of your target, the macro will ensure that the read or write is safe against anomaly 05-00-0311.

When building for parts and silicon revisions that require anomaly 05-00-0311 workaround the macro __WORKAROUND_FLAGS_MMR_ANOM_311 is defined at compile, assemble and link stages.

05-00-0311 –

The anomaly is seen when an access of a System MMR Flag register is followed by an access of a specific MMR. The result of the anomaly can be that flag pins configured as outputs that are "set" can erroneously transition to "clear". The anomaly impacts all revisions of ADSP-BF53[123] and ADSP-BF561 parts.

"Given some sample application code, such as:"

```
int accessMMR()
{
      unsigned short w, x, y, z;
      x = *pFIO_FLAG_D;
      y = *pFIO_MASKA_D;
      z = x & y;
      *pFIO_FLAG_C = z;
      w = *pFIO_EDGE;
      *pFIO_DIR = 0;
      ...
}
```

then the anomaly-safe code would be:

```
#include <sys/05000311.h>
...
int accessMMR()
{
      unsigned short w, x, y, z;
      FIO_ANOM_0311_FLAG_R(x, pFIO_FLAG_D);
      FIO_ANOM_0311_MASKA_R(y, pFIO_MASKA_D);
      z = x & y;
```

```
        FIO_ANOM_0311_FLAG_W(z, pFIO_FLAG_C);
        FIO_ANOM_0311_EDGE_R(w);
        FIO_ANOM_0311_DIR_W(0);
        ...
}
```

For more information on anomaly 05-00-0311 please see the appropriate errata sheet which can be downloaded from <http://www.analog.com/processors/blackfin/support/ICanomalies.html>.

5. System Builder Template Changes

If you have a project that was generated with System Builder, loading the project after installing this Update will result in a popup requesting regeneration of the code/LDF. Regenerating is recommended to keep current with the latest template changes. This affects three files:

1. LDF
2. basiccrt.s
3. heaptab.c

After regenerating, you are current with the latest improvements. No further action is needed.

If you didn't use System Builder, refer to:

Tar 31774: C++ exceptions do not work with generated multicore ldfs
Tar 31555: Generated cplbtab file uses undefined macro CACHE_MEM_MODE

6. SDRAM Differences Between ADSP-533 EZ-KIT Board Versions
(Note for Tar 32480: adi_pwr_SetFreq causes VDK exception on BF533)

The SDRAM configuration setup when loading an ADSP-BF533 EZ-KIT from VisualDSP++ with the emulator target option **Use XML Reset Values** selected sets the EBIU_SDBCTL register for 32MB of SDRAM. This is the correct setting for EZ-KIT Revisions 1.6 and lower. If using an EZ-KIT Revision 1.7 or above there is actually 64MB of SDRAM available. However the EBIU_SDBCTL will remain the same as for the earlier revisions when loading through VisualDSP++. The effect of this is that attempts to use the memory over 32MB will result in runtime errors or hardware exceptions.

To enable the full 64MB, edit *<install-dir>\System\ArchDef\ADSP-BF533-proc.xml* and modify the EBIU_SDBCTL register reset value located within the <register-reset-definitions> block near the end of the file. Change:

```
<!-- For BF533 EZ-KIT Lite's rev 1.7 and above use 0x25 -->
<!-- register name="EBIU_SDBCTL" reset-value="0x25" core="Common" / -->

<!-- For BF533 EZ-KIT Lite's rev 1.6 and below use 0x13 -->
<register name="EBIU_SDBCTL" reset-value="0x13" core="Common" />
```

to:

```
<!-- For BF533 EZ-KIT Lite's rev 1.7 and above use 0x25 -->
<register name="EBIU_SDBCTL" reset-value="0x25" core="Common" />

<!-- For BF533 EZ-KIT Lite's rev 1.6 and below use 0x13 -->
<!--register name="EBIU_SDBCTL" reset-value="0x13" core="Common" / -->
```

Then disconnect and reconnect to the target at which point the changes will take effect.

7. Loader is Packing External Memory PM for SHARC LX3/LX4 Processors

The loader was updated to pack the external memory PM data in the ADSP-2136x and ADSP-2137x processors without requiring the PACKING() command in the project .ldf file. This was in response to Tar 30900: Elfloader drops 16 bits when creating 48-bit image file.

Prior to this update, you were required to use the PACKING() command in the project .ldf file to get the external data packed in the .dxe file, before the loader generated a loader file from the .dxe file. The loader was also updated to transfer the logical addresses to the physical addresses for the external packed data. The following is a simple example of the packing scheme taken by the loader:

In .dxe file:
External logical address instruction

```
0x200000 0x112233445566
0x200001 0x778899aabbcc
```

In loader file:

```
0x300000 0x33445566
0x300001 0xbbcc1122
0x300002 0x778899aa
```

8. Loader Switches

In response to customer problems, the following loader switches were introduced:

1. The –NoZeroBlock switch directs the SHARC loader not to compress zero data (VisualDSP++ 4.5 June Update)

2. The -NoSecondStageKernel and -NoFinalTag switches were introduced for the ADSP-BF561. They can be used when the boot ROM is expected to be used to boot two cores without a second stage kernel involved. (VisualDSP++ 4.5 November Update)

Problems Addressed

The following table is a list of the problems addressed in this Update. Details on any particular problem can be found on the Tools Anomaly web page. The URL is:

http://www.analog.com/processors/tools/anomalies

Processor Family	Problem Number	Tool	Description
All	30957	Compiler	Assertion failed: v->storage_class != sc_auto
All	32181	Compiler	Internal error (brilops.c:536) building with -force-circbuf
All	32270	Run Time Libraries	A value that is just less than 0.1 may print as 0.0 with %f
Blackfin	31407	Compiler	__builtin_cmplx_mul fault when the optimization is enabled
Blackfin	32344	Compiler	Former workaround for 05-00-0311 not safe
Blackfin	31555	LDFGen	generated cplbtab file uses undefined macro CACHE_MEM_MODE
Blackfin	31774	LDFGen	C++ exceptions do not work with generated multicore ldfs
Blackfin	29507	Run Time Libraries	libdsp functions cabs_fr16, cartesian_fr16 fail for input 0x8000
Blackfin	30779	Run Time Libraries	CPLB function declarations not guarded with extern "C"
Blackfin	31806	Run Time Libraries	math_const.h defines long doubles that can result in larger code
Blackfin	30854	Simulator	DMA in 2D mode and AUTO fails with interrupt
Blackfin	30909	Simulator	Array Mode in a DMA transfer does not advance to the next desc
Blackfin	30955	Simulator	Second DMA does not occur correctly
Blackfin	31859	Utilities	Using -meminit results in error about missing map file
SHARC	29877	ADspCommon XML Files	USTAT3 and USTAT4 do not appear in the ADSP-21160 USTAT register window
SHARC	29948	ADspCommon XML Files	ADSP-21160 does not allow 4 column window for external memory
SHARC	31777	Assembler	Anomaly warnings incorrectly enabled for 08-00-0014
SHARC	23536	Assembler	Floating point cannot appear in "discarded" conditional asm block
SHARC	22335	Assembler	Assembler accepts multiply with m reg destination
SHARC	28090	Assembler	Assembling without specifying processor now produces error
SHARC	30677	Emulator	Some loads to external 16-bit memory fail
SHARC	31498	Emulator	Inactive PEy Register set does not display correctly
SHARC	31393	Emulator	EMUCTL_NOBOOT is incorrectly set to '0' for all processors
SHARC	32170	Flash Programmer	ADSP-21262 and ADSP-21364 EZ-KIT Serial flash will change
SHARC	30877	LDF	LDF needs to include packing() for DMAONLY sections
SHARC	31816	LDF	Incorrect memory mapping for ADSP-21375
SHARC	30878	Loader	Loader needs to process packed data from DMAONLY sections
SHARC	30900	Loader	Elfloader drops 16 bits when creating 48-bit image file
SHARC	30513	Run Time Libraries	SIMD DSP functions ifft() and cfft() not interrupt safe
SHARC	31633	Run Time Libraries	logd and log10d not safe for negative input

SHARC	31058	Run Time Libraries	pragma interrupt dispatcher should return with RTS(LR)
SHARC	29816	Simulator	External memory data not loaded correctly when WIDTH(8)
SHARC	32607	Simulator	DM words mapped to external 16-bit segment truncated
SHARC	32608	Simulator	External data symbols not visible in Data(DM)/Two Column window
SHARC	31333	VDK	Link warning with 2137X VDK projects using si revision 'any'
SHARC	30862	VDK	Hardware clears VDK kernel IRPTL bit when at kernel level

The following table is a list of problems that were addressed in the February update but had not been included in the February Release notes.

SHARC	13790	Assembler	Float point multiply with fixed-point result register becomes fixed-point multiply
SHARC	15704	Assembler	Instruction sequence error in included file reported against the wrong file
SHARC	29914	Assembler	Hex initializers with an odd number of digits > 8 right shifted
SHARC	30414	Assembler	Incorrect report of missing stall after MODE1 write for ADSP-21367
SHARC	29364	Simulator	ADSP-2137x simulator does not execute from external memory

VisualDSP++ 4.5 (Updated February 2007) Release Note

The following release note concerns the February 2007 Update to the VisualDSP++ 4.5 release. This release is inclusive of previous Updates. The contents of future Updates will be inclusive of all previous Updates. The release notes for past Updates are appended to the end of this release note.

Identifying Which Update Is Currently Installed on Your System

The Update level is identified in three places:

1. The Add/Remove Programs Control Panel entry for VisualDSP++ 4.5.
2. The VisualDSP++ GUI's About box, located at "Help" > "About VisualDSP++".
3. In the file …`\System\VisualDSP.ini`, in the `ProductName` key.

Installing an Update

The procedure for installing an Update to VisualDSP++ is described below. Note that with VisualDSP++'s support for installing multiple instances of itself, it is possible to "trial" an Update in a new directory before switching over your "golden" tools installation to the Update.

1. Use the Start Menu to navigate to VisualDSP++'s "Maintain this Installation" item.
2. Select "Go to the Analog Devices website" and click "Next". This will launch your web browser and navigate it to the proper URL to download Updates from.
3. Download the VisualDSP++ Update file (.VDU) of interest to your hard disk. Note that these files have a .VDU file extension and cannot be executed directly.
4. Navigate to "Maintain this Installation" again. If you have multiple installations of VisualDSP++ on your computer, be doubly sure you are navigating to the installation you wish to Update.
5. Select "Apply a downloaded Update" and click "Next". Click the "…" browser button and navigate to the .VDU file that you downloaded in step 3. Click "OK", then "Next".
6. Follow the on-screen prompts to complete the installation of the Update.

Significant Additions

The primary purpose of VisualDSP++ Updates is to address problems and stabilize the release. Significant new functionality is not expected to be introduced in an Update. However, incremental support (i.e., emulation, example programs, header files, default LDF, errata accommodations, EZ-KIT Lite software, etc.) for new semiconductor products will be added as these products become available and gain support within the VisualDSP++ tools.

In this release:

1. Emulation support for the ADSP-BF54x family has been added.

2. Support for ADSP-BF538|9 revision 0.4 has been added.

3. The Blackfin compiler, runtime, VDK, and SSL libraries include new workarounds for hardware anomalies 05-00-0311 and 05-00-0312. The C/C++ compiler will automatically enable these workarounds when building for parts and silicon revisions that require them. Alternatively they can be enabled using the `-workaround` switch. The Blackfin assembler has a new warning to indicate potential instances of anomaly 05-00-0312.

   05-00-0311 –

   The anomaly is seen when an access of a System MMR Flag register is followed by an access of a specific MMR. The result of the anomaly can be that flag pins configured as outputs that are "set"

can erroneously transition to "clear". The anomaly impacts all revisions of ADSP-BF53[123] and ADSP-BF561 parts.

The compiler works around the anomaly:

1. By adding a load of the `CHIPID` MMR at the start of code generated for C defined event handlers. System Services' handlers and VDK's interrupt templates make similar accommodations. The related workaround for 05-00-0283 will be used if it is enabled at the same time as this workaround.

2. By identifying accesses, or potential accesses, of any of the various System MMR Flag registers associated with the errata and generating a load of CHIPID after each such access. The compiler relies on use of literal addressing of MMRs to identify these accesses, such as using the various `p<MMR>` macros defined in the `cdef<PART>.h` include files. Any loads and stores that do not use literal addresses and the type of the access is defined qualified as `volatile`, will be assumed to be flag MMR accesses, unless the `-no-assume-vols-are-mmrs` switch is used. If the `-no-assume-vols-are-mmrs` compiler switch is used the compiler will apply the workaround for suitable literal address accesses only and will make no assumptions for non-literal loads and stores even if `volatile`.

To enable this compiler workaround manually the `-workaround flags-mmr-anom-311` switch can be used. When the workaround is enabled the macro `__WORKAROUND_FLAGS_MMR_ANOM_311` is defined at compile, assemble and link stages.

05-00-0312 -

The anomaly is seen when SSYNC, CSYNC instructions or loads of registers `LCx`, `LTx`, and `LBx` are interrupted. The anomaly impacts all Blackfin parts and revisions except ADSP-BF535.

The compiler workarounds are as follows:

1. When the workaround is enabled the compiler builtin functions `__builtin_ssync()` and `__builtin_csync()` have been modified to ensure that interrupts are disabled before the `sync`/`csync` instruction and enabled after.

2. New `ssync` and `csync` builtins have been provided that do not disable interrupts. These can be used in place of the existing builtins for code that has been manually verified as safe against the anomaly. These new builtins are called `__builtin_ssync_int()` and `__builtin_csync_int()`.

3. When the workaround is enabled the compiler will ensure that any loads of the `LBx`, `LTx` and `LCx` registers are executed with interrupts disabled. It does this by inserting a `CLI` instruction before such loads and an `STI` instruction after. The workaround is not required for hardware loop `LSETUP` instructions.

4. The compiler will arrange to save and restore the loop registers (`LBx`, `LTx` and `LCx`) while interrupts are disabled for C/C++ nested interrupt handlers defined using the `sys/exception.h` `EX_REENTRANT_HANDLER` macro.

The assembler has been modified to include a warning which when enabled will identify potential causes of the anomaly. This warning can be enabled using the `-anomaly-warn 05-00-0312` switch. The warning can be suppressed in assembly code ranges which have manually been determined to be safe against the anomaly using the assembler's `.MESSAGE` directive. The warning id to use when this is required is `5515`.

To enable this compiler workaround manually the `-workaround sync-loop-anom-312` switch can be used. When the workaround is enabled the macro `__WORKAROUND_SYNC_LOOP_ANOM_312` is defined at compile, assemble and link stages.

For more information on these anomaly please see the appropriate errata sheets which can be downloaded from http://www.analog.com/processors/blackfin/support/ICanomalies.html.

Changes to Existing Behaviors, Projects, and Source Code

When addressing problems, we attempt to make any changes backward compatible with existing projects. However, depending on the nature of a problem, compatibility issues are sometimes unavoidable. This section highlights any changes in the Update that may require the modification of “working” projects or otherwise influence existing behavior.

In this release, no changes have been identified.

Problems Addressed

The following table is a list of the problems addressed in this Update. Details on any particular problem can be found on the Tools Anomaly web page. The URL is:

http://www.analog.com/processors/tools/anomalies

Processor Family	Problem Number	Tool	Description
All	17307	Utilities	MEMINIT implementation makes part of .map file useless
All	27177	Compiler	Adding start up code to project breaks build
All	29624	Compiler	-pch -eh crashes compiler - unoptimised and optimised
All	29838	VDK	Destroying a periodic semaphore can damage the time queue
All	30048	VDK	UserHistoryLog called incorrectly during context switch
All	30181	Compiler	Register locked too many times
All	30351	Compiler	assoc_source_line_modif: bad address
Blackfin	29221	Run Time Libraries	multicore runtime libraries always link in I/O library
Blackfin	29552	Emulator	Remove setting PF12 on BF533 USB example
Blackfin	29717	Run Time Libraries	rms_fr16 can return a negative result
Blackfin	29833	Loader	Page needs to support SPI Slave for 561 Rev 0.5
Blackfin	29834	Compiler	Internal error "Cannot locate Reg to be Released" at call
Blackfin	29841	CRTGen	BF52x proj with start up code & Ldf do not build
Blackfin	29917	VDK	VDK exception handler could corrupt P1
Blackfin	29955	Run Time Libraries	Link error with multi-theaded C library and setlocale
Blackfin	30049	Run Time Libraries	C++ streams broken for multicore applications
Blackfin	30112	LDFGen	I/O from both cores of multi-core app fails with cache enabled
Blackfin	30118	Emulator	cache not flushed when emulator performs a write of 1
Blackfin	30164	Emulator	External memory read error at address x200FFFFF - x20100000
Blackfin	30220	Linker	Linker miscalculates program size for EZ-KIT restriction
SHARC	29236	IDDE	Server License expires 'n' days after validation when compiling
SHARC	29726	VDK	Sharc thread level context switches not interrupt safe
SHARC	29742	Emulator	Usage of Parallel port interrupt restricted
SHARC	29761	Simulator	2136[7,8,9] sysctl incorrect
SHARC	29819	LDF	LDFs contain unnecessary macros to deal with EZ-kit monitors
SHARC	29828	Run Time Libraries	ADSP-2136x CRT need support for new LX3 interrupts
SHARC	29829	Compiler	circindex returns wrong index when constant 0 used fails –Og
SHARC	29856	Run Time Libraries	213xx\lib\2136x_any\libcppeh37x.dlb should be libcpp37xeh.dlb
SHARC	29898	Run Time Libraries	The ifftf() function can generate wrong results
SHARC	29916	Assembler	Hex init < 15 digits for LW space is left-justfied with no warn
SHARC	30123	Loader	Loader does not convert PROM non-zero address from Hex to S3
SHARC	30126	Loader	Loader fails to add 0x00000000 to an odd SPI 48-bit block
SHARC	30410	Assembler	Check for anomaly 08000002 absent for 21368/9
SHARC	30411	Assembler	Check for anomaly 07000010 absent for 21362, 21366
SHARC	30415	Assembler	Assembler not checking for anomaly 08000001 on 21367/8/9
TigerSHARC	29792	Run Time	strtod may convert floating-point hex constants incorrectly

		Libraries	
TigerSHARC	29801	Run Time Libraries	Link Buffer Registers macros defined incorrectly in defts201.h
TigerSHARC	29830	Run Time Libraries	Floating point comparison routine clobbers an unexpected reg
TigerSHARC	30029	Run Time Libraries	The conv2d function can generate wrong results

VisualDSP++ 4.5 (Updated November 2006) Release Note

The following release note concerns the November 2006 Update to the VisualDSP++ 4.5 release. This release is inclusive of previous Updates. The contents of future Updates will be inclusive of all previous Updates. The release notes for past Updates are appended to the end of this release note.

Identifying Which Update Is Currently Installed on Your System

The Update level is identified in three places:

1. The Add/Remove Programs Control Panel entry for VisualDSP++ 4.5.
2. The VisualDSP++ GUI's About box, located at "Help" > "About VisualDSP++".
3. In the file …`\System\VisualDSP.ini`, in the `ProductName` key.

Installing an Update

The procedure for installing an Update to VisualDSP++ is described below. Note that with VisualDSP++'s support for installing multiple instances of itself, it is possible to "trial" an Update in a new directory before switching over your "golden" tools installation to the Update.

1. Use the Start Menu to navigate to VisualDSP++'s "Maintain this Installation" item.
2. Select "Go to the Analog Devices website" and click "Next". This will launch your web browser and navigate it to the proper URL to download Updates from.
3. Download the VisualDSP++ Update file (.VDU) of interest to your hard disk. Note that these files have a .VDU file extension and cannot be executed directly.
4. Navigate to "Maintain this Installation" again. If you have multiple installations of VisualDSP++ on your computer, be doubly sure you are navigating to the installation you wish to Update.
5. Select "Apply a downloaded Update" and click "Next". Click the "…" browser button and navigate to the .VDU file that you downloaded in step 3. Click "OK", then "Next".
6. Follow the on-screen prompts to complete the installation of the Update.

Significant Additions

The primary purpose of VisualDSP++ Updates is to address problems and stabilize the release. Significant new functionality is not expected to be introduced in an Update. However, incremental support (i.e., emulation, example programs, header files, default LDF, errata accommodations, EZ-KIT Lite software, etc.) for new semiconductor products will be added as these products become available and gain support within the VisualDSP++ tools.

In this release:

1. Compiler, assembler, and linker support for a new Blackfin processor family, the ADSP-BF54x, has been added. The loader and System Services are not yet available and will be added in a future release.

2. This Update has been broadly tested under Windows Vista Beta 2 (5308) and is believed to be fully operational. While the base release of VisualDSP++ 4.5 installs successfully, changes were required to avoid run-time operational problems. Versions of VisualDSP++ prior to this Update will generally not be operational under Vista and are not supported. It is possible that the final "gold" release of Window Vista will change and require further (minor) modification of VisualDSP++. Future Update release notes will discuss this topic as required.

   Note that, at this time, the emulator and EZ-KIT Lite Windows device drivers are not digitally signed. Windows will pop up a warning message for each driver installed. Confirm the installation of each driver during base and/or Update installation.

Changes to Existing Behaviors, Projects, and Source Code

When addressing problems, we attempt to make any changes backward compatible with existing projects. However, depending on the nature of a problem, compatibility issues are sometimes unavoidable. This section highlights any changes in the Update that may require the modification of "working" projects or otherwise influence existing behavior.

In this release:

1. The VDK libraries included in this update now flag an additional value used as a timeout in Pend APIs (`PendSemaphore`, `PendMessage`, `PendEvent`, `PendDeviceFlag`) as being invalid. Passing the value `(0 | kNoTimeoutError)` as the timeout will now result in a `kInvalidTimeout` error. In previous versions of VDK the value was silently accepted. However, specifying that no error should be dispatched in the event of a timeout, but also that the Pend call should never timeout, has never been a useful thing to do. It was also ambiguous as to what the result would be. The issuing of an error in this case will now draw attention to this fact.

2. For Blackfin processors, the assembler keyword `LOOP` was incorrectly also defined as a macro in Blackfin header files (for UART loopback enablement). This has been corrected in this Update. Any code that uses `LOOP` in its macro form may now fail to build and must be updated to use `LOOP_ENA` instead.

1. For Blackfin processors, the user interface management of the `-pFlag` parameter to the loader has been revised. It is now dynamically managed and varies with processor, silicon revision, boot mode, and width. The loader now generates new warnings for illegal combinations.

   After application of this Update, any Blackfin project using `-pFlag` should be verified to ensure that the correct setting is being used. This setting should also be verified whenever the processor, silicon revision, boot mode, or width is changed in VisualDSP++.

   The following tables show the allowed values for `–pFlag`:

   ADSP-BF531/2/3

Revision	0.0-0.2		0.3-0.5	
Width	8	16	8	16
Flash	NONE	NONE	NONE	NONE
SPI	NONE		NONE	
SPI slave			1-15 PF1-15	

   ADSP-BF538/9

Revision	0.0-0.3	
Width	8	16
Flash	NONE	NONE
SPI	NONE	
SPI slave	1-15 PF1-15	

ADSP-BF534/6/7

Revision	0.0		0.1-0.2		0.3	
Width	8	16	8	16	8	16
Flash	NONE	NONE	NONE PF0-15 PG0-15 PH0-15	NONE PF0-15 PG0-15 PH0-15	NONE PF0-15 PG0-15 PH0-15	NONE PF0-15 PG0-15 PH0-15
SPI	NONE		NONE PF0-9 PF15 PG0-15 PH0-15		NONE PF0-9 PF15 PG0-15 PH0-15	
SPI slave	1-15 PF1-15		NONE PF0-10 PF15 PG0-15 PH0-15		NONE PF0-10 PF15 PG0-15 PH0-15	
TWI	NONE		NONE PF0-15 PG0-15 PH0-15		NONE PF0-15 PG0-15 PH0-15	
TWI slave	NONE		NONE PF0-15 PG0-15 PH0-15		NONE PF0-15 PG0-15 PH0-15	
UART	2-15 PF2-15		NONE PF2-15 PG0-15 PH0-15		NONE PF2-15 PG0-15 PH0-15	
FIFO						NONE PF0 PF2-15 PG0-15 PH0-15

Notes:

1. Blank fields indicate not supported boot modes.
2. BF533/4/6/7/8/9 always has the RESVECT bit (bit #2 in block header flag word) set.
3. BF531/2 have always RESVECT bit (bit #2 in block header flag word) cleared.
4. VisualDSP++ property page provides a "NONE" option in the pflag pull-down menu. When chosen, no –pflag switch goes to the command line and the PPORT and PFLAG fields in the block header flag word are zero.

Problems Addressed

The following table is a list of the problems addressed in this Update. Details on any particular problem can be found on the Tools Anomaly web page. The URL is:

http://www.analog.com/processors/tools/anomalies

Processor Family	Problem Number	Tool	Description
All	23803	IDDE	Floating License expires warning with permanent license string
All	29233	Run Time Libraries	Wrong file positions may be returned for a text stream
All	29234	Run Time Libraries	scanf does not input data of the form .d{dddd}edd correctly
All	29576	Compiler	circindex returns wrong index when constant 0 used
Blackfin	28080	Linker	False -si-rev incompatibilities when build from command line
Blackfin	29070	Simulator	DMA Descriptor Array uses NXT_DESC not CURR_DESC
Blackfin	29112	Debug Agent	cache flush functions fail on blackfin ez-kits
Blackfin	29163	Config	freqency selection in Icetest doesn't work with HPUSB
Blackfin	29164	Config	Icetest scan test fails on MP targets using HPUSB/USB
Blackfin	29235	Run Time Libraries	signal() will never return SIG_IGN
Blackfin	29258	Compiler	P0 clobbered in -ieee-fp compiler support function
Blackfin	29308	LDFGen	Custom SDRAM setting can make the LDF unusable
Blackfin	29320	System Services	printf before and after adi_pwr_Init causes lockup in Rel on 538
Blackfin	29352	Run Time Libraries	si-revision 'any' builds result in linker warning li2152
Blackfin	29363	Run Time Libraries	defBFXXX.h files #define LOOP, keyword in Blackfin assembly
Blackfin	29434	Flash Program	0 sectors in a flash programmer drivers crashes IDDE
Blackfin	29513	System Services	adi_pwr_SetFreq() disables CLKBUFOE bit in VR_CTL
Blackfin	29524	Assembler	Assembler rejects source file name without .suffix
Blackfin	29648	CRTGen	Customising clock settings can cause the CRT not to compile
Blackfin	29652	Compiler	circptr builtin may fail especially without -O
Blackfin	29662	Run Time Libraries	__var_wa_06000047 undefined when linking for BF561 rev 0.5
Blackfin	29672	Simulator	Signed/Unsigned Integer 8-bit format not properly handled
SHARC	29059	Loader	change in PLL setup in example codes
SHARC	29236	IDDE	Server License expires 'n' days after validation when compiling
SHARC	29267	Compiler	two more cases that can lead to anomaly #4 for ADSP-2126x DSPs
SHARC	29451	Run Time Libraries	The FFTs from filter.h may loop forever
SHARC	29518	Run Time Libraries	Interrupt handler does not restore multiplier regs properly
SHARC	29602	Run Time Libraries	Some interrupt handlers are called with an incorrect parameter
SHARC	29706	Compiler	2nd switch statement not reached
TigerSHARC	29382	VDK	“Run Total Time (cycles)” calculation is incorrect

VisualDSP++ 4.5 (Updated September 2006) Release Note

The following release note concerns the September 2006 Update to the VisualDSP++ 4.5 release. This release is inclusive of previous Updates. The contents of future Updates will be inclusive of all previous Updates. The release notes for past Updates are appended to the end of this release note.

Identifying Which Update Is Currently Installed on Your System

The Update level is identified in three places:

1. The Add/Remove Programs Control Panel entry for VisualDSP++ 4.5.
2. The VisualDSP++ GUI's About box, located at "Help" > "About VisualDSP++".
3. In the file …`\System\VisualDSP.ini`, in the `ProductName` key.

Installing an Update

The procedure for installing an Update to VisualDSP++ is described below. Note that with VisualDSP++'s support for installing multiple instances of itself, it is possible to "trial" an Update in a new directory before switching over your "golden" tools installation to the Update.

1. Use the Start Menu to navigate to VisualDSP++'s "Maintain this Installation" item.
2. Select "Go to the Analog Devices website" and click "Next". This will launch your web browser and navigate it to the proper URL to download Updates from.
3. Download the VisualDSP++ Update file (.VDU) of interest to your hard disk. Note that these files have a .VDU file extension and cannot be executed directly.
4. Navigate to "Maintain this Installation" again. If you have multiple installations of VisualDSP++ on your computer, be doubly sure you are navigating to the installation you wish to Update.
5. Select "Apply a downloaded Update" and click "Next". Click the "…" browser button and navigate to the .VDU file that you downloaded in step 3. Click "OK", then "Next".
6. Follow the on-screen prompts to complete the installation of the Update.

Significant Additions

The primary purpose of VisualDSP++ Updates is to address problems and stabilize the release. Significant new functionality is not expected to be introduced in an Update. However, incremental support (i.e., emulation, example programs, header files, default LDF, errata accommodations, EZ-KIT Lite software, etc.) for new semiconductor products will be added as these products become available and gain support within the VisualDSP++ tools.

In this release:

1. Software support for the ADSP-BF538F EZ-KIT Lite has been added. This includes debug connectivity via the on-board USB debug agent, the flash programmer (both GUI support and underlying drivers), and LwIP (Ethernet) drivers. Flash drivers are provided for both the on-chip flash found on the ADSP-BF538F and the off-chip flash device on the EZ-KIT Lite. The on-line help system has been updated to include this product.

2. The System Service Library (SSL) has been enhanced to support ADSP-BF538 Blackfin processor. Included in this Update is support for the EBIU, Dynamic Power Management, DMA, Interrupt, Deferred Callback, Timer, Flag and Port Control system services for the ADSP-BF538 processor. The default LDFs have been updated to link against SSL.

3. Blackfin device drivers have been updated. The adi_ad1836a_ii and adi_ad1938_ii codec drivers now support automatic SPORT configuration. PPI, UART, SPI, TWI and SPORT device drivers for the ADSP-BF538 processor have been introduced.

4. Software support for the ADSP-21375 EZ-KIT Lite has been added. This includes debug connectivity via the on-board USB debug agent, and well as flash programmer GUI support and underlying drivers. Execution from external memory is now supported in simulator, emulator, and EZ-KIT Lite debugging sessions. The on-line help system has been updated to include this product.

Changes to Existing Behaviors, Projects, and Source Code

When addressing problems, we attempt to make any changes backward compatible with existing projects. However, depending on the nature of a problem, compatibility issues are sometimes unavoidable. This section highlights any changes in the Update that may require the modification of "working" projects or otherwise influence existing behavior.

In this release:

1. Within the Blackfin DMA Manager's include file, `adi_dma.h`, an additional field named `CallbackFlag` has been added to the data structures that describe large and small model descriptors, `ADI_DMA_DESCRIPTOR_LARGE` and `ADI_DMA_DESCRIPTOR_SMALL` respectively. This field should be set to `TRUE`, if a callback is requested after the descriptor has been processed or `FALSE` if no callback is requested after the descriptor has been processed. Previously, the `DI_EN` bit within the configuration register of the descriptor was used to trigger a callback.

   This change affects only user code that explicitly calls the `adi_dma_Queue()` function.

2. Source code files that make calls into the System Services (code that includes the file "`services.h`") should be rebuilt after installation of this update.

Problems Addressed

The following table is a list of the problems addressed in this Update. Details on any particular problem can be found on the Tools Anomaly web page. Note that after the Issues headings in the top half of the Tools Anomaly web page, problems are detailed in numeric order. The URL is:

http://www.analog.com/processors/tools/anomalies

Processor Family	Problem Number	Tool	Description
All	28661	Install	FYI Norton Internet Security prevents installation
All	28799	Run Time Libraries	fread may signal EOF prematurely
All	28849	Compiler	Compiler driver+LDF requires dummy.c file
All	28895	Run Time Libraries	C++ runtime support for alternative heaps fails to link (li1021)
All	28948	Compiler	loop pragmas don't work on C++ loops with embedded declarations
All	29012	VDK	VDK kThreadSwitched history events don't call UserHistoryLog
All	29090	IDDE	-g is added when convert project from 4.0 to 4.5 with -g>0 in th
All	29378	VDK	Incorrect behaviour following VDK timequeue wraparound
Blackfin	25362	Emulator	Emulator returns unknown family when targeting BF534 rev 0.2
Blackfin	26646	CRTGen	Rev 1.7 BF533 EZ-KIT not properly supported in generated CRT
Blackfin	28335	Simulator	C++ exceptions cannot be used on single core BF561 simulator
Blackfin	28339	Simulator	Simulator not updating registers correctly
Blackfin	28595	IDDE	Additional include directories not always added to command line
Blackfin	28764	System Services	1836A_ii codec device driver fails in TDM mode
Blackfin	28819	IDDE	Adding file to project with user defined config cause error
Blackfin	28823	Compiler	3-byte structs returned from functions incorrectly when -O used
Blackfin	28839	IDDE	impossible sclk value in Project wizard
Blackfin	28875	Run Time Libraries	Typo in cdefBF532.h - voild instead of void
Blackfin	28967	Configurator	configurator doesn't handle more than 11 devices properly
Blackfin	29025	VDK	The placement of TMK and VDK libraries can cause link errors
Blackfin	29072	IDDE	Blackfin processors not listed in EL Global Properties dialog
Blackfin	29177	Run Time Libraries	SIC_IMASK set with bad bit before DMA transfer (meminit)
Blackfin	29231	Loader	BF533 rev 0.5 loader files add a zero byte for each data byte
SHARC	28857	Emulator	primes for 21061 does not run
SHARC	28882	Run Time Libraries	SIMD libdsp function vecdotf() might fail
TigerSHARC	28720	Compiler	TigerSHARC wrapper generation/regs_clobbered not saving K conds
TigerSHARC	28736	Splitter	The splitter counts one more byte per word in .stk format
TigerSHARC	28752	Run Time Libraries	TigerSHARC fread can incorrectly return 0 in byte-address mode
TigerSHARC	28907	VDK	TIMER0H register not set when Timer interrupt is set to None

TigerSHARC	29032	IDDE	Can't load Annotations
TigerSHARC	29043	Assembler	invalid warning about mult instruction option
TigerSHARC	29227	VDK	VDK API level check can cause false positive Kernel Panics

VisualDSP++ 4.5 (Updated July 2006) Release Note

The following release note concerns the July 2006 Update to the VisualDSP++ 4.5 release. This is the first in what is anticipated to be a series of Updates. The contents of future product Updates will be inclusive of all previous Updates. At that time, the release notes for past Updates will be appended to the end of the current release note.

Identifying Which Update Is Currently Installed on Your System

The Update level is identified in three places:

1. The Add/Remove Programs Control Panel entry for VisualDSP++ 4.5.
2. The VisualDSP++ GUI's About box, located at "Help" > "About VisualDSP++".
3. In the file …`\System\VisualDSP.ini`, in the `ProductName` key.

Installing an Update

The procedure for installing an Update to VisualDSP++ is described below. Note that with VisualDSP++'s support for installing multiple instances of itself, it is possible to "trial" an Update in a new directory before switching over your "golden" tools installation to the Update.

1. Use the Start Menu to navigate to VisualDSP++'s "Maintain this Installation" item.
2. Select "Go to the Analog Devices website" and click "Next". This will launch your web browser and navigate it to the proper URL to download Updates from.
3. Download the VisualDSP++ Update file (.VDU) of interest to your hard disk. Note that these files have a .VDU file extension and cannot be executed directly.
4. Navigate to "Maintain this Installation" again. If you have multiple installations of VisualDSP++ on your computer, be doubly sure you are navigating to the installation you wish to Update.
5. Select "Apply a downloaded Update" and click "Next". Click the "…" browser button and navigate to the .VDU file that you downloaded in step 3. Click "OK", then "Next".
6. Follow the on-screen prompts to complete the installation of the Update.

Significant Additions

The primary purpose of VisualDSP++ Updates is to address problems and stabilize the release. Significant new functionality is not expected to be introduced in an Update. However, incremental support (i.e., emulation, example programs, header files, default LDF, errata accommodations, EZ-KIT Lite software, etc.) for new semiconductor products will be added as these products become available and gain support within the VisualDSP++ tools.

In this release:

1. The Blackfin compiler, runtime, VDK, and SSL libraries include new workarounds for hardware anomalies 05-00-0189 and 05-00-0283. The compiler will automatically enable these workaround when building for parts and silicon revisions that require them. Alternatively they can be enabled using the `-workaround` switch.

   05-00-0283 –

   > One part of the workaround is to include a code sequence in all event handlers. The sequence makes a mispredicted jump over a dummy MMR read. This must be done before any `SSYNC`s in the handler. This sequence is generated by the compiler for C/C++ based event handlers that use `#pragma interrupt` or `sys/exception.h` defined macros such as `EX_INTERRUPT_HANDLER`. The two handlers affected in the

runtime libraries are `_cplb_hdr` and the interrupt dispatcher `_despint` which have been modified to include the workaround.

The second part of the workaround is to avoid system MMR writes in the two instructions after a not-predicted conditional jump. The compiler will insert `nop` instructions to avoid this when it identifies the problem sequence.

These workarounds can be enabled using the `-workaround stalled-mmr-write-283` switch. When the workaround is enabled the macro `__WORKAROUND_STALLED_MMR_WRITE_283` is defined at compile, assemble and link stages

05-00-0198 –

A workaround for this anomaly was already available in the compiler. However the conditions which cause anomaly have changed to include a new code sequence. The compiler has been modified to identify this new sequence. The anomaly may occur where MMR reads or writes occur immediately after a stalled memory read. The compiler will avoid such code being generated for C/C++ compiled code. The runtime libraries are safe against this anomaly.

This workaround can be enabled using the `-workaround sdram-mmr-read` switch. When the workaround is enabled the macro `__WORKAROUND_SDRAM_MMR_READ` is defined at compile, assemble and link stages.

2. There has been a change of compiler behavior relating to MMR (Memory Mapped Register) accesses and volatile variables. The new switch `-no-assume-vols-are-mmrs` has been added.

   There are various MMR related hardware errata that the compiler supports workarounds for; 05-00-0122, 05-00-0157, 05-00-0198, 05-00-0283. Previously the compiler would only implement these workaround for accesses that it could absolutely determine were to MMRs. This in practice meant that only literal MMR addresses accesses could be determined accurately. More complex accesses, for example using addresses stored in variables, might not be identifiable as MMR accesses and could therefore result in the various anomalies being hit.

   The compiler has been modified to try and avoid missing these more complex MMR accesses. If there is an access to a variable that is defined as `volatile`, and the compiler cannot determine that the access is not to an MMR, the compiler will now assume it is an access to an MMR unless the new switch `-no-assume-vols-are-mmrs` is used.

Changes to Existing Behaviors, Projects, and Source Code

When addressing problems, we attempt to make any changes backward compatible with existing projects. However, depending on the nature of a problem, compatibility issues are sometimes unavoidable. This section highlights any changes in the Update that may require the modification of “working” projects or otherwise influence existing behavior.

In this release no changes have been identified.

Problems Addressed

The following table is a list of the problems addressed in this Update. Details on any particular problem can be found on the Tools Anomaly web page. The URL is:

http://www.analog.com/processors/tools/anomalies

Processor Family	Reference Number	Tool	Description
All	28180	Compiler	modena test c0527101 fails byte addressing when compiled –eh
All	28225	Compiler	C++ exceptions thrown from inline virtual functions may fail
All	28244	VDK	Issue with dynamically created VDK components at startup
All	28271	Run Time Libraries	Increase in code size for printf
All	28341	Run Time Libraries	attributes missing in libx dojs
All	28399	Compiler	static C++ classes can cause bad debug
All	28876	Assembler	Cannot perform source-level debug of assembly source files
All	28929	Emulator	USB-ICE inoperable when updating to 4.5 while connected
All	28935	VDK	User's timer interrupt settings can be overwritten
All	29140	Emulator	RoHS USB-ICE does not work with base 4.5 install
Blackfin	28043	Loader	Loader supports different default Rev #s from what it should.
Blackfin	28229	IDDE	Annotations left in source pane after they are turned off
Blackfin	28287	Loader	Zero padding to booting stream
Blackfin	28297	Compiler	Compiler internal error (macdefs.c:1162) with –O
Blackfin	28305	Run Time Libraries	ftell() with -full-io in text mode can return incorrect position
Blackfin	28309	Compiler	Non-interrupt safe prologue code generated for BF535
Blackfin	28338	Compiler	INTERNAL COMPILER ERROR: No switch note found
Blackfin	28383	Assembler	.inc/binary produces corrupted doj
Blackfin	28410	Run Time Libraries	Cache flushing on BF535 and wireless parts doesn't work
Blackfin	28445	IDDE	If I add cplbtable and then disable cache project does not link
Blackfin	28450	IDDE	configurator screen not coming to front
Blackfin	28467	IDDE	errors and other issues removing configurations
Blackfin	28472	Run Time Libraries	Possibility of erroneous result computed by fir_decima_fr16()
Blackfin	28487	Run Time Libraries	Wrong comment in the source of the radix2 FFT library functions
Blackfin	28497	Run Time Libraries	Incorrect macro in defBF534.h and defBF538.h
Blackfin	28517	LDFGen	Possible link error with generated BF561 LDF and mem init
Blackfin	28521	LDF	OTHERCORE not implemented correctly in default multi-core LDFs
Blackfin	28588	Compiler	bad compare of unsigned short and unsigned literals –O
Blackfin	28600	Loader	Loader does not work with Rev 0.3 for 539.
Blackfin	28679	Loader	Remove the ignore block from loader files.
Blackfin	28686	LDFGen	Single core generated LDF uses $OBJECTS before definition
Blackfin	28688	Run Time Libraries	Instance of speed path anomaly 05-00-0209 in cache flush func
Blackfin	28689	LDFGen	LDFGen does not always use the correct CPLB table
Blackfin	28710	Loader	Loader need to support Rev. 0.5 for 531/2/3
Blackfin	28765	IDDE	project not restored after starting connection-less IDDE
Blackfin	28779	Run Time	defbf534.h has incorrect PFDE_UART macro definitions

		Libraries	
Blackfin	28994	VDK	Potential for excessive stack usage on Blackfin processors
SHARC	28283	LDF	The section "seg_int_code" has grown unnecessarily
SHARC	28569	VDK	Sharc li2152 link warnings when using earlier Si Revision
SHARC	28692	Run Time Libraries	0.0 2126x libraries built with an inappropriate silicon revision
SHARC	28761	Run Time Libraries	No SRU header files
TigerSHARC	28263	Compiler	long long to double conversion fails in byte-addressing mode
TigerSHARC	28267	Compiler	Assertion failure: bril/zp/macdefs.c:2747 with -O -never-inline
TigerSHARC	28295	VDK	Cannot view system stack usage in the expert linker
TigerSHARC	28490	VDK	CCNTx register is read in the wrong order
TigerSHARC	28880	VDK	TS20x Idle thread prevents scheduling of user threads

2 VISUALDSP++ 4.5 NEW FEATURES AND ENHANCEMENTS

VisualDSP++ 4.5 has new features and enhancements designed to increase productivity and shorten application development cycles. This chapter describes the features and enhancements introduced in VisualDSP++ 4.5. The information is presented as follows.