VisualDSP++® 5.0 Update 6 Release Notes

Revision 1.3

August 24, 2009

Table of Contents

Nomenclature

In the past, VisualDSP++ updates were labeled by the month and year of their release. In order to improve clarity, updates are now numbered (e.g., Update 1, Update 2, etc.).

Release Notes

These release notes subsume the release notes for previous updates. Release notes for previous updates can be found at the end of this document.

Installation

This update can only be installed on a previous VisualDSP++ 5.0 installation. If VisualDSP++ 5.0 is not installed, please install it first. Installation on a previous update is permitted. If a newer update has already been installed, please do not install this update. This update is not intended to be installed on alpha or beta releases. For example, do not install this update on the ADSP-2146x Beta 1 Update.

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++.

Installing the Update

Follow the instructions below for installing this update. Please note that since VisualDSP++ supports having multiple instances installed on a single system. See the *Cloning VisualDSP++* section below for more information.

1. Use the Start Menu to navigate to VisualDSP++ “Maintain this installation”. By default, this is at Start Menu → All Programs → Analog Devices → VisualDSP++ 5.0.
2. Select “Go to the Analog Devices website” and click Next. This will open a window in your web browser.
3. Select the appropriate Processor Software Tools Upgrades to match your processor.
4. Select and download the desired update (VisualDSP++ 5.0_Update6.vdu) to your hard drive.
5. Again, use the Start Menu to navigate to VisualDSP++ “Maintain this installation”.
6. Select “Apply a downloaded Update” and click Next.
7. Browse for the downloaded Update file (VisualDSP++ 5.0_Update6.vdu) and click Next.
8. Follow the on-screen prompts to complete installation of this Update.

Cloning VisualDSP++

VisualDSP++ supports cloning of an existing installation. A clone of an installation creates a new instance of a product from an existing installation, rather than from a CD or web software distribution. The use of clones allows you to maintain multiple versions of VisualDSP++ on the same PC at different update levels, and provides a risk-free way to "test" new updates or patches.

To clone your existing installation of VisualDSP++:

1. Go to Start->Programs->Analog Devices->VisualDSP++ 5.0 (or equivalent)->Maintain this Installation
2. Select "Clone this Installation" and click Next.
3. Optionally click Advanced to set the Start menu path.
4. Enter the Clone install path and click Next.

Definitions

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

TAR – Tools Anomaly Reference Number

Tools Anomaly Reference Number, or TAR, is used for tracking confirmed defect reports in VisualDSP++.

New Hardware Support

VisualDSP++ updates often include support for new processors, new silicon revisions for existing processors and new EZ-KIT Lite® and EZ-Board® 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.

New Processors and Processor Revision Support

This section lists new processors and processor revisions available in this update. Refer to the data sheets and hardware reference manuals for information on system configuration, peripherals, registers, and operating modes.

Update 6 introduces a new processor series to the SHARC® processor family:

- ADSP-21462 silicon revision 0.0
- ADSP-21465 silicon revision 0.0
- ADSP-21467 silicon revision 0.0
- ADSP-21469 silicon revision 0.0

No new Blackfin® or TigerSHARC® processors are supported with Update 6.

Update 6 also provides support for the following silicon revisions to existing Blackfin® processors:

- ADSP-BF512 silicon revision 0.1
- ADSP-BF514 silicon revision 0.1
- ADSP-BF516 silicon revision 0.1
- ADSP-BF518 silicon revision 0.1

No new silicon revisions to existing SHARC® or TigerSHARC® processors are supported with Update 6.

Processor Revision Deprecation

Support for the following silicon revisions are deprecated in Update 6 as the revision was never released.

- ADSP-BF561 silicon revision 0.4

New Evaluation Board Support

Support has been added for the following new evaluation boards.

USB EZ-Extender®

Update 6 introduces initial support for the USB EZ-Extender which connects with both Blackfin and SHARC EZ-KIT Lite and EZ-Board evaluation systems. The Blackfin evaluation systems include the ADSP-BF518F, ADSP-BF526, ADSP-BF537, ADSP-BF538F and ADSP-BF548 evaluation systems as well as future EZ-Boards. The SHARC evaluation systems include soon to be released ADSP-21469 EZ-Board as well as other future EZ-Boards. Examples for the ADSP-BF518F EZ-Board are provided for bulk loopback, bulk redirect and mass storage:

```
Blackfin\Examples\USB EZ-EXTENDER\bulk_loopback_app
Blackfin\Examples\USB EZ-EXTENDER\bulk_redirect_io_app
Blackfin\Examples\USB EZ-EXTENDER\mass_storage_app
```

ADSP-BF518F EZ-Board Examples

VisualDSP++ 5.0 Update 6 includes additional examples for the ADSP-BF518F EZ-Board evaluation system:

```
Blackfin\Examples\ADSP-BF518F EZ-Board\LAN\DNS_Client
Blackfin\Examples\ADSP-BF518F EZ-Board\LAN\FileServerStdio
Blackfin\Examples\ADSP-BF518F EZ-Board\LAN\HTTP_Server
Blackfin\Examples\ADSP-BF518F EZ-Board\LAN\Multicast_Sender
Blackfin\Examples\ADSP-BF518F EZ-Board\LAN\PTP*
Blackfin\Examples\ADSP-BF518F EZ-Board\LAN\TCPIP_Trace
```

*This is a new example for Precise Time Protocol necessary for IEEE-1588 support.

ADSP-BF526 EZ-Board Examples

VisualDSP++ 5.0 Update 6 includes additional examples for the ADSP-BF526 EZ-Board. The example can be found in the following directories:

```
Blackfin\Examples\ADSP-BF526 EZ-KIT Lite\drivers\usb\bulk_loopback_app
Blackfin\Examples\ADSP-BF526 EZ-KIT Lite\drivers\usb\bulk_redirect_io_app
Blackfin\Examples\ADSP-BF526 EZ-KIT Lite\drivers\usb\mass_storage_app
Blackfin\Examples\ADSP-BF526 EZ-KIT Lite\drivers\usb\mass_storage_host_app
Blackfin\Examples\ADSP-BF526 EZ-KIT Lite\services\FileSystem\NAND
Blackfin\Examples\ADSP-BF526 EZ-KIT Lite\services\FileSystem\VDK
```

Landscape LCD EZ-EXTENDER Examples

VisualDSP++ 5.0 Update 6 includes a Landscape LCD EZ-Extender example for the ADSP-BF526 EZ-Board evaluation system. This example demonstrates the use of the LCD Driver, Capacitive Touch Controller Driver and Touch Screen Controller Driver.

```
Blackfin\Examples\Landscape LCD EZ-EXTENDER\SketchPad
```

New System Services and Device Drivers

The following are now supported by VisualDSP++ 5.0:

USB Audio Class Driver

The USB Audio Class Driver is new with Update 6. Examples are provided for the ADSP-BF526, ADSP-BF527 and ADSP-BF548 evaluation systems:

```
Blackfin\Examples\ADSP-BF526 EZ-KIT Lite\Drivers\usb\usb_audio_app
Blackfin\Examples\ADSP-BF527 EZ-KIT Lite\Drivers\usb\usb_audio_app
Blackfin\Examples\ADSP-BF548 EZ-KIT Lite\Drivers\usb\usb_audio_app
```

LCD Driver

The LCD Driver to support the LCD on the Landscape LCD Extender has been added for Update 6. The new Sketchpad example for the ADSP-BF526 EZ-Board demonstrates how to use this new driver.

Capacitive Touch Controller Driver

The Capacitive Touch Controller Driver to support the AD7147 on the Landscape LCD Extender has been added for Update 6. The new Sketchpad example for the ADSP-BF526 EZ-Board demonstrates how to use this new driver.

Touch Screen Controller Driver

The Touch Screen Controller Driver to support the AD7879 on the Landscape LCD Extender has been added for Update 6. The new Sketchpad example for the ADSP-BF526 EZ-Board demonstrates how to use this new driver.

ADSP-BF51x System Services

System Services and Device Driver support for the ADSP-BF51x processor series and ADSP-BF518F EZ-Board has been added for Update 6.

PWM System Service

The PWM peripheral is new with Update 6. It is supported for the ADSP-BF51x processor series. An example for the ADSP-BF518 EZ-Board has been provided:

```
Blackfin\Examples\ADSP-BF518F EZ-Board\Services\PWM\pwm_sine_wave
```

New Examples

This section specifies new examples that are not specific to evaluation boards. No new examples for this section are provided with Update 6.

Boot ROM Code

This section describes changes to the Boot ROM code.

New Boot ROM Code is available in Update 6 for the following processors:

- ADSP-BF51x rev 0.0
- ADSP-BF522/524/526 rev 0.1
- ADSP-BF54xM rev 0.3

Init Code

This section describes the changes to the Init Code.

PLL / Voltage Regulator

Changing the PLL and the Voltage Regulator is now turned off by default. You can activate this feature again in the corresponding initcode header file. For example, undefine the following in `ezkitBF527_initcode.h`:

```
#define __ACTIVATE_DPM__
```

This was done to be compatible with the emulator register resets in the processor XML files, which just set up the EBIU.

ADSP-2146x Processor Series

The ADSP-2146x is a new SHARC processor series that supports a short word instruction set for code size reduction. The ADSP-2146x is binary backwards compatible; that is, ADSP-2137x binary code will run on the ADSP-2146x without modification. Short word instructions and normal word instructions must be in separate sections, but otherwise can both be used in the same application for optimal compatibility. This section outlines the changes necessary to VisualDSP++ 5.0 provided in Update 6 to support this new processor series.

At the time of the Update 6 release, the short word instruction set had not been fully tested. It is advisable to use the normal word instruction set.

Feature Macros

The following macros are automatically predefined by the assembler, compiler and linker:

New macro	Description
`__2146x__` `__214xx__`	All ADSP-2146x processors
`__ADSP21462__`	ADSP-21462
`__ADSP21465__`	ADSP-21465
`__ADSP21467__`	ADSP-21467
`__ADSP21469__`	ADSP-21469

The following macros are automatically predefined by the assembler and compiler:

New macro	Description
`__NORMAL_WORD_CODE__`	ADSP-2146x processors when building in normal word mode.
`__SHORT_WORD_CODE__`	ADSP-2146x processors when building in short word mode.

New Compiler Switches

The following C/C++ compiler switches have been added:

`-normal-word-code`

The `-normal-word-code` switch is for ADSP-2146x processors only. It has the same effect as compiling with the `-nwc` switch. It directs the compiler to generate instructions of normal word size (48-bits).

`-nwc`

Same as "`-normal-word-code`".

`-short-word-code`

The `-short-word-code` switch is for ADSP-2146x processors only. It has the same effect as compiling with the `-swc` switch and directs the compiler to generate instructions of short word size (16/32/48-bits). This switch is the default setting when compiling for ADSP-2146x processors.

`-swc`
Same as "`-short-word-code`".

New Pragma Support

The C/C++ compiler support for `#pragma section` and `#pragma default_section` has been modified to accept new section qualifiers.

New Qualifier	Description
`SW`	code is short word (ADSP-2146x only)
`NW`	code is normal word (ADSP-2146x only)

Example usage:
```
#pragma section("foo", SW)                    // Code is short word
#pragma default_section(CODE,"foo2", NW)      // Code is normal word
```

New Automatic Attributes

When `-no-auto-attrs` is not specified, the compiler defines a new default attribute called "Encoding". The value of the attribute depends on the code produced during compilation.
- If only short word code is produced, the attribute has the value “`SW`”.
- If only normal word code is produced, the attribute has the value “`NW`”.
- If both short and normal word code is produced, the attribute has the value “`Mixed`”.

New Assembler Switches

The following assembler switches have been added:

`-normal-word-code`
Instructs the assembler not to treat input sections bearing the qualifier "`/PM`" as if they were "`/SW`".

`-nwc`
Same as "`-normal-word-code`".

`-short-word-code`
Instructs the assembler to treat input sections bearing the qualifier "`/PM`" as if they were "`/SW`".

`-swc`
Same as "`-short-word-code`".

`-swc-exclude name1[,name2...]`
Excludes the named section[s] from the effect of the “`-swc`“ switch.

New .SECTION Directive Qualifiers

There are two new qualifiers for the `.SECTION` directive:

`SW`

16-bit short word section. This specifies that the section contains instructions that are to be assembled for loading into a 16-bit short word memory segment. Instructions will be assembled as compressed 32- or 16-bit instructions, if possible.

`NW`
Normal word section. Instructions will be assembled as normal 48-bit instructions to be loaded into a 48-bit memory segment. Unlike the PM section qualifier, NW sections will always be 48-bits -- they are not affected by the `-swc` switch.

New Assembler Directives

Three new directives are available to control whether instructions will be compressed:

`.COMPRESS`
This directive indicates that all the following instructions in the source file should be compressed, if possible. It has no effect on sections that are not being assembled as short word. Its effect is canceled by a `.NOCOMPRESS` directive later in the source file.

`.NOCOMPRESS`
This directive indicates that all the following instructions in the source file should not be compressed. Its effect is canceled by a `.COMPRESS` directive later in the source file.

`.FORCECOMPRESS`
This directive causes the next instruction to be compressed, if possible. Has no effect on sections that are not being assembled as short word. Only the immediately following assembly instruction is affected by this directive. This directive overrides the effect of a previous `.NOCOMPRESS` directive, but only for this one instruction. It can also override certain conservative assumptions normally made by the assembler, such as when an immediate value is an expression containing a symbol; in this case, the assembler normally does not generate a compressed instruction because the ultimate value of the symbolic expression might not fit in the immediate field of the compressed instruction.

Note that `.COMPRESS` and `.FORCECOMPRESS` are advisory only:

- There is no guarantee that a particular instruction will be compressed, even if it is theoretically possible to do so.
- Instructions might be 'uncompressed' if they are near the end of a DO loop.
- Whether you get compression for a particular instruction might change due to assembler enhancements or fixes.
- There will be no warnings if instructions cannot be compressed.

Therefore it is probably not recommended to create code layouts (e.g. tables with fixed size entries) that depend on particular instructions being compressed.

New Assembler Informational ea2536

There is a difference in addressing while in short word mode –instructions are no longer one address unit apart as it is in normal word mode (and for all other SHARC parts). So, address arithmetic is dangerous. Unfortunately, the SHARC calling and returning sequence relies on address arithmetic based on each instruction being one instruction apart.

SHARC parts have always used a call sequence like:

```
                cjump _func (db); dm(i7,m7)=r2; dm(i7,m7)=pc;
```

The function that gets called will compute the return address by taking the saved PC and adding 1 to it. The problem occurs in short word mode where the return address is no longer one address further than the saved PC because all instructions are no longer guaranteed to be one address unit long.

In order to maintain compatibility with all existing SHARC calling conventions, we have adopted the following paradigm for invoking a function:

```
                cjump _func (db); dm(i7,m7)=r2; dm(i7,m7)=<Unique_label>-1;
<Unique_label>:
```

This calling sequence will work in short word mode and in normal word mode and with any parts in the SHARC family. All existing return code will work since it will take the saved address (no longer necessarily the PC of the last instruction in the delay slot) and incrementing it will get the address of the next instruction.

The assembler does this fix up automatically when necessary. Informational message ea2536 lets the user know that the assembler made this change. (If you were to look at the saved address in the debugger you might notice that it didn't save the PC and wonder why). You can modify your source code and it will still work for the 2136x as well as the 2146x. If you don't modify your source code, the assembler will continue to do the fix up to make it work.

New System Registers Header Files

The following header files to provide symbolic names of system registers and their bits have been added. These are included from `platform_include.h`.

```
214xx/include/def21462.h
214xx/include/def21465.h
214xx/include/def21467.h
214xx/include/def21469.h
214xx/include/Cdef21462.h
214xx/include/Cdef21465.h
214xx/include/Cdef21467.h
214xx/include/Cdef21469.h
```

Processor Header Files

The following processor-specific header files have been added. These are included from `processor_include.h`.

```
214xx/include/21462.h            ADSP-21462 DSP functions
214xx/include/21465.h            ADSP-21465 DSP functions
214xx/include/21467.h            ADSP-21467 DSP functions
214xx/include/21469.h            ADSP-21469 DSP functions
```

Run-Time Libraries

New C/C++ files and libraries have been included for ADSP-2146x processors.

Description	Library/File Name
C run-time library	`libc.dlb libcmt.dlb libc_nwc.dlb libcmt_nwc.dlb`
C++ run-time library	`libcpp.dlb libcppmt.dlb libcpp_nwc.dlb libcppmt_nwc.dlb`
C++ run-time library with exception handling support	`libcppeh.dlb libcppehmt.dlb libcppeh_nwc.dlb libcppehmt_nwc.dlb`
DSP run-time library	`libdsp.dlb libdsp_nwc.dlb`
I/O run-time library	`libio.dlb libiomt.dlb libio_nwc.dlb libiomt_nwc.dlb`
I/O run-time library with no support for alternative device drivers or `printf("%a")`	`libio_lite.dlb libio_litemt.dlb libio_lite_nwc.dlb libio_litemt_nwc.dlb`
VDK libraries	`TMK-2146X.dlb VDK-CORE-21469.dlb VDK-i-2146X.dlb VDK-e-2146X.dlb VDK-n-2146X.dlb TMK-2146X_nwc.dlb VDK-CORE-21469_nwc.dlb VDK-i-2146X_nwc.dlb VDK-e-2146X_nwc.dlb VDK-n-2146X_nwc.dlb`
C start-up file — calls set-up routines and `main()`	`21462_hdr.doj 21465_hdr.doj 21469_hdr.doj 21467_hdr.doj`
C++ start-up file — calls set-up routines and `main()`	`21462_cpp_hdr.doj 21462_cpp_hdr_mt.doj 21465_cpp_hdr.doj 21465_cpp_hdr_mt.doj 21467_cpp_hdr.doj 21467_cpp_hdr_mt.doj 21469_cpp_hdr.doj 21469_cpp_hdr_mt.doj`

The new libraries located in the `214xx\lib` in the install are built without any workarounds enabled (`-si-revision none`).

In addition, a library directory called `21469_rev_any` is supplied. Libraries in this directory will contain workarounds for all relevant anomalies on all revisions of ADSP-2146x processors.

Libraries with names that end in "`_nwc.dlb`" have been built for normal word mode (`-nwc`). Other libraries are built for short word mode (`-swc`) where possible.

LDF and Mapping Changes

A new SW memory type keyword has been introduced for the ADSP-2146x processor family. The SW memory type refers to short word memory (16 bits wide). This new memory type can be used within the MEMORY command, and as a section qualifier in the section mapping command. See the *VisualDSP++ 5.0 Linker and Utilities Manual* for more information.

A new output section memory type qualifier has been introduced. This qualifier forces all input objects within the output section to be mapped as the memory type specified. Not all combinations are valid. See the table below for a list of newly supported translation mappings.

Input Section Type	Output Section Qualifier	Memory Segment Mapped To
PM	PM	SW
DM	DM (32-bit)	SW
DATA64	DATA64	SW

Using the section qualifier mapping of NW/DATA64/DM input sections into SW memory segments is allowed for ADSP-2146x processor series. The linker will place the input sections into a SW memory segment and then translate the objects' addresses to the specified memory space. The resulting behavior is as if the memory segment was defined as the memory type specified in the output section qualifier.

Enhanced Disassembly Window Display of Pipeline Stages

Changes have been made to the Disassembly Window to provide better information about the instruction pipeline:

1. The Disassembly Window now indicates when the instruction in Address or Decode stage has been overridden or had a stall inserted. The gutter label of the instruction in that stage will change from upper case to lower case in such instances – that is, from 'A' to 'a' for Address stage, and from 'D' to 'd' for Decode stage.
2. The window will show that the pipeline continues to advance while stepping through a non-delayed branch in the ADSP-2136x, ADSP-2137x, and ADSP -2146x parts. Previously, the arrow indicating the execute stage had stayed on the branch instruction for the three steps following such branches. Now the arrow becomes gray and advances with each step following the branch until it becomes a green arrow at the branch target address.

Project Configuration

Additional project configurations are provided for the ADSP-2146x processor series to better support the new Variable Instruction Set Assembly (VISA):

The current Project Configurations set the Assembler, Compiler-Processor and Linker-Processor to default to VISA:

Debug
Release

New Project Configurations set the Assembler, Compiler-Processor and Linker-Processor to default to normal word:

DebugNWC
ReleaseNWC

Migrating Projects from the ADSP-2146x Beta 1 Update

It is not advisable to migrate projects directly from the Beta 1 Update as there has been a number of changes. Should this be necessary, please keep in mind the following changes:

- The default LDFs have changed
- The Debug and Release Project Configurations in Beta 1 built normal word projects. They will build VISA / short word projects in Update 6. In order to build a normal word project, make the following changes to the Project Options:

- On the Compiler-Processor property page, set the Variable Instruction Set Encoding (VISA) to ”Generate Normal Word code”
- On the Assembler property page, set the Variable Instruction Set Encoding (VISA) to ”Generate Normal Word code”
- On the Linker-Processor property page, set the Libraries to ”Use Normal Word code Run-Time Libraries.

Accelerator Simulation

This release has simulator support for the ADSP-2146x FIR, IIR and FFT hardware accelerators.

Peripheral Simulation

The simulator will provide support for the following ADSP-2146x peripherals in future releases:

- DDR2
- Link Ports
- SPORT DMA

The simulator does not provide support for the following ADSP-2146x peripherals:

- DDR2 controller
- DTCP
- Link Ports
- MediaLB
- Programmable Clock Generator
- Shared Memory
- S/PDIF
- SPI
- SPORTs
- SRC
- TWI
- UART

TAR 40528: .SEGMENT assembler directive may behave incorrectly

The .SEGMENT and .SECTION directives behave differently with the assembler. The .SEGMENT defaults to normal word code, whereas the .SECTION defaults to whichever mode is enabled. To avoid any issues, it is best to use the .SECTION assembler directive and not the .SEGMENT assembler directive.

USB Stack Enhancements

The ADI USB Stack has undergone a considerable overhaul and with Update 6 provision is made for a more robust and extensible solution to your USB needs.

Robustness

Several issues have been addressed that were the cause of lock ups and data corruption, particularly regarding the operation of the ADI-specific Bulk transfer class driver.

With ADSP-BF548 rev 0.1 and higher and ADSP-BF52x rev 0.2 and higher, the PHY calibration register is set upon reset to a factory-defined default, obtained from direct calibration of the specific part. This calibration value may vary from part to part. It is recommended that an application adds the following statements to the initialization code of their application:

```
#include <builtins.h>
#include <sys/platform.h>
/* ... */

       /* for rev 0.0 we need to calibrate the USB analog PHY
       */
       if ( 0x00 == ((*pDSPID) & 0xFF))
       {
              /* clear it to make sure USB will work after booting */
              *pUSB_APHY_CNTRL = 0;

              /* setup calibration register */
              *pUSB_APHY_CALIB = 0x5411;
              ssync();
       }
```

Similarly, for ADSP-BF52x:

```
       if (  (0x00 == ((*pDSPID) & 0xFF)) ||
              (0x01 == ((*pDSPID) & 0xFF)))
       {
              /* clear it to make sure USB will work after booting */
              *pUSB_APHY_CNTRL = 0;

              /* setup calibration register */
              *pUSB_APHY_CALIB = 0x6510;
              ssync();
       }
```

Another aspect of the improvements in robustness, the USB controller driver changed to use separate handlers for each of the four USB interrupts (ADI_INT_USB_INT0, ADI_INT_USB_INT1, ADI_INT_USB_INT2, ADI_INT_USB_DMAINT). By default, all the USB Interrupt handlers are chained at the default USB priority level (IVG 11 for ADSP-BF54x and IVG10 for ADSP-BF52x). It is important therefore that memory is made available to the interrupt manager for three additional secondary interrupt handlers. Otherwise you may see the following message displayed in the Output Window of VisualDSP++ 5.0:

```
              ERROR: Insufficient Memory in the Interrupt Manager
              USB requires memory for four secondary handlers
              [sizeof(ADI_INT_SECONDARY_MEMORY * 4)]
              Supply more memory using adi_int_Init()
```

The simplest way to resolve the issue is to increase the amount of memory passed to the adi_int_Init() routine by (**3 * ADI_INT_SECONDARY_MEMORY**). You could also experiment with changing the IVG levels of each interrupt to suit your application requirements. This will change the number of additional secondary handlers according to your existing interrupt priority assignments.

To change the priority level of an interrupt, add the following statement in your application code ahead of enabling the USB driver (using the ADI_USB_ENABLE_USB Command). For example to change the priority of the ADI_INT_USB_INT0 interrupt to IVG 10:

```
            adi_int_SICSetIVG(ADI_INT_USB_INT0, 10 );
```

To assist you in this, note that each of the above interrupts are assigned to handle the following events:

`ADI_INT_USB_INT0`	Data receive events (to Blackfin)
`ADI_INT_USB_INT1`	Data transmission events (from Blackfin)
`ADI_INT_USB_INT2`	Connection events
`ADI_INT_USB_DMAINT`	DMA buffer completion events

Better Device Detection

More USB memory devices can now be detected, including some card reader devices (single cards only – multiple cards are outside the scope of the hardware) and low-cost USB flash drives. Device detection can also be achieved at higher core clock frequencies (up to 600 MHz) on the ADSP-BF548 processor.

Hot plug support is greatly improved making the removal and insertion of USB memory devices quick and reliable. A consequence of this is that it may take slightly longer for the device mode applications to be recognized by the host PC.

Extensibility

Support has been added so that more complex device classes can be implemented, such as the USB Audio class for which Update 6 provides both a class driver and example applications. Support is included for:

- Large configurations
- Multiple alternate interfaces
- Arbitrary data transmitted and received on Endpoint Zero for control interfaces.

Net2272

As well as providing improved support for the built-in USB OTG controllers on ADSP-BF52x/54x, improvements have also been made to the net2272 controller driver to provide enhanced support for Blackfin parts without integrated USB. The Audio class driver can be implemented on the net2272 interface but no examples are released at this time. Please contact Support if you require these examples, stating the processor type for which you require them (ADSP-BF533, ADSP-BF561 only). In addition to the extensibility improvements above, the following issue is also addressed.

- Handling of multiple read/writes requests
- No longer required to copy device driver to project directory

In order to provide greater flexibility with interrupt priorities, the net2272 driver has been changed to use the default IVG level of the Memory DMA streams and Programmable Flag interrupts as set in the System Interrupt Controller (SIC_IARx) registers. Unfortunately, on ADSP-BF533/BF537 processors, the default priority of the Memory DMA interrupts are lower than that of the Programmable Flag interrupt, which will result in lock up in existing applications. All net2272 examples have been updated to set the priority of the Memory DMA interrupt to a higher priority than the Programmable Flag interrupt as follows:

BF533/BF537	IVG 11
BF561	IVG 9
BF518 (new)	IVG 10

The ADI_USB_CMD_SET_DMA_CHANNEL I/O command has been added to facilitate the setting of this priority level (without detailed knowledge of the Interrupt Manager peripheral ID value), e.g.:

```
adi_dev_Control(PeripheralDevHandle, ADI_USB_CMD_SET_DMA_IVG,
 (void *)11);
```

Please see the examples in the VisualDSP++ installation for further details. These examples for the ADSP-BF53x and ADSP-BF561 can be found under:

```
$(VDSP)\Blackfin\Examples\USB-LAN EZ-EXTENDER\USB
```

For the ADSP-BF518, the examples can be found under:

```
$(VDSP)\Blackfin\Examples\USB EZ-EXTENDER
```

Configuring the net2272 driver for DMA operation.

Users who previously took a private copy of the adi_usb_net2272.c driver source file in order to modify the definition of the USE_DMA pre-processor macro should note that this is no longer necessary, and its use will not produce the desired effect. The driver source now uses two macros, USE_RX_DMA and USE_TX_DMA, to enable or disable its use of memory DMA. These macros may be defined in a project's pre-processor options and used to selectively enable and disable DMA in either direction. USE_RX_DMA and USE_TX_DMA should be given the value 0 to disable the use of DMA in the respective direction or a non-zero value to enable it. The default values if none are supplied are USE_RX_DMA=1 and USE_TX_DMA=0. This combination is chosen to provide the best performance for the bulk loopback examples.

Critical Fixes/Changes

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

ADSP-BF51x any/none/0.0 Support

Revision "any" and "none" application builds failed to link in Update 5 as these revisions use the revision 0.1+ memory map in the default and generated LDFs. The Update 6 linker is modified to correctly check for the revision 0.1+ memory map so revisions "any" and "none" no longer fail.

This linker change means that applications built for revision 0.0 are built with the linker `-nomemcheck` switch. This is automatically applied by the compiler driver (ccblkfn) when building for ADSP-BF51x processors revision 0.0.

The linker will now issue the following warning when building for ADSP-BF518 parts and revision 0.0:

```
 [Warning li2280]  The ADSP-BF518 si-revision 0.0 has been deprecated, use -nomemcheck to
suppress error el2011
```

This warning is issued even when `-nomemcheck` is used. While this message says that revision 0.0 has been deprecated that is not currently the case. The warning is not issued for builds that target ADSP-BF512 or ADSP-BF516 but the use of `-nomemcheck` for these parts is also automatically included as they have the same memory map change as ADSP-BF518.

Using `-nomemcheck` disables linker memory checking so care should be taken to ensure that memory is defined correctly in customized LDFs in ADSP-BF51x revision 0.0 applications as any errors will not be caught by the linker.

One potential issue is when building a project using revision "automatic" without being connected and then subsequently connecting and loading that DXE to a revision 0.0 ADSP-BF51x target is that the will fail. This is because "automatic" causes the tools to build for the default revision which is 0.1 when not connected to target with rev 0.0. This issue can be avoided by explicitly building for revision "0.0" when the target is an ADSP-BF51x revision 0.0 part.

Please review silicon anomaly 05-00-0444 in the ADSP-BF51x parts Errata Sheets for details of the memory map change.

Active CPLBs (Blackfin)

All locked CPLBs will be loaded into the CPLB registers before any unlocked CPLBs, instead of just the first 16 as happened previously. Error labels `too_many_locked_data_cplbs` and `too_many_locked_instruction_cplbs` will indicate that there are at least 16 locked data or instruction cplbs, respectively, and additional cplbs will be locked out.

Compiler Changes for MISRA Exemplar Suite (Blackfin)

The compiler MISRA checking has been enhanced to improve compliance against the MISRA-C:2004 Exemplar Suite. Other significant changes have been made to many of the VisualDSP++ header files to further improve VisualDSP++ MISRA compliance.

New compiler error (cc2238) for pragma interrupt functions

A new error, cc2238, has been added to the C/C++ compilers. The error is issued if an interrupt handler function is defined to take parameters. An interrupt handler function is a function defined using any of the following compiler pragmas:

- `#pragma interrupt` (used in the definition of Blackfin sys/exception.h macros EX_INTERRUPT_HANDLER, EX_EXCEPTION_HANDLER and EX_NMI_HANDLER)
- `#pragma interrupt_complete` (cc21k only)
- `#pragma interrupt_complete_nesting` (cc21k only)
- `#pragma interrupt_reentrant` (ccts only)

Functions that are defined as interrupt handler functions cannot be passed parameters because they are not explicitly called and are only run in response to an interrupt event. Functions that cause this error to be issues should be modified to remove parameters.

New DCPLB_DATAx and ICPLB_DATAx bit position macros (Blackfin)

The `defblackfin.h` and `def_LPBlackfin.h` include files have had new bit positions macros added for the DCPLB_DATAx and ICPLB_DATAx registers.

The new macros added to `defblackfin.h` for use by the ADSP-BF535 part are:

- `#define CPLB_USER_WR_P 3 /* 0=no write access, 0=write access allowed (user mode) */`
- `#define CPLB_SUPV_WR_P 4 /* 0=no write access, 0=write access allowed (supervisor mode) */`
- `#define CPLB_L1SRAM_P 5 /* 0=SRAM mapped in L1, 0=SRAM not mapped to L1 */`
- `#define CPLB_DA0ACC_P 6 /* 0=access allowed from either DAG, 1=access from DAG0 only */`
- `#define CPLB_DIRTY_P 7 /* 1=dirty, 0=clean */`
- `#define CPLB_L1_CHBL_P 12 /* 0=non-cacheable in L1, 1=cacheable in L1 */`
- `#define CPLB_WT_P 14 /* 0=write-back, 1=write-through */`

The new macros added to `def_LPblackfin.h` for use by non ADSP-BF535 parts are:

- `#define CPLB_PORTPRIO_P 9 /* 0=low priority port, 1= high priority port */`
- `#define CPLB_LRUPRIO_P 8 /* 0=can be replaced by any line, 1=priority for non-replacement */`
- `#define CPLB_USER_WR_P 3 /* 0=no write access, 0=write access allowed (user mode) */`
- `#define CPLB_SUPV_WR_P 4 /* 0=no write access, 0=write access allowed (supervisor mode) */`
- `#define CPLB_DIRTY_P 7 /* 1=dirty, 0=clean */`
- `#define CPLB_L1_CHBL_P 12 /* 0=non-cacheable in L1, 1=cacheable in L1 */`
- `#define CPLB_WT_P 14 /* 0=write-back, 1=write-through */`
- `#define CPLB_L1_AOW_P 15 /* 0=do not allocate cache lines on write-through writes, 1= allocate cache lines on write-through writes. */`

Applications that contain their own local definitions of these macros may encounter compiler warning "`cc0047: {D} warning: incompatible redefinition of macro`" if the definition does not match the ones given above. Presuming that the application's local definition is for the same purpose as the new VisualDSP++ 5.0 ones it can be deleted. Otherwise the local definition and uses of the local definition will need to be renamed.

New VSTAT Macro in defBF52x_base.h (ADSP-BF51x/BF52x)

A new macro called VSTAT for the PLL_STAT voltage regulator status bit has been added to the def headers for ADSP-BF52x parts. If your application contains a macro called VSTAT and uses the ADSP-BF52x def include files, you will need to change the application's macro and uses to use a different name to avoid build errors or warnings.

Deprecated IWR Macro (ADSP-BF51x/BF52x)

The ADSP-BF51x and ADSP-BF52x Blackfin processor series have two wakeup registers called SIC_IWR0 and SIC_IWR1. The ADSP-BF54x processor series has three wakeup registers, but other single-core Blackfin processors designed prior to the ADSP-BF54x have only one wakeup register, called SIC_IWR.

If the register name SIC_IWR is used in an ADSP-BF51x or ADSP-BF52x application, the compiler does not issue an error or a warning. It uses the legacy definitions contained in the include files '`def_BF51xbase.h`' and '`defBF52xbase.h`', which define SIC_IWR as equivalent to SIC_IWR0. This can be misleading when porting an application from a previous Blackfin processor, which has only one wakeup register, to a Blackfin processor with two wakeup registers. The application may have intended to address all the wakeup bits, but by using the SIC_IWR register name, the compiled code will only address the wakeup bits in SIC_IWR0, while SIC_IWR1 is neglected. The application may need to be modified, if the intent was to address both the SIC_IWR0 and SIC_IWR1 registers.

If porting an application to an ADSP-BF51x or ADSP-BF52x, from an older Blackfin processor, any usage of the SIC_IWR register in the source code should be examined to determine whether it should be replaced by both the SIC_IWR0 and SIC_IWR1 registers.

New Messages Due to the SIMD SHARC Assembler Enforcing Loop Restrictions

The assembler will analyze hardware loop code, warning the user when the documented restrictions have been violated, and providing a reminder when its static analysis cannot verify the code adheres to the restrictions – specifically that a CALL in the last three instructions must return with the (LR) option to preserve integrity of the loop.

The new messages the assembler can produce during this validation are:
Ea2018 – error detected that two loops end on the same instruction.
Ea2019 – warns that the code violates restrictions on loop end instructions.
Ea2020 – warns of a possible violation of the loop end restrictions, as when a call is in two instruction loop and the assembler does not know the loop count. (a one iteration two instruction loop should not have a CALL in it)
Ea2021 - reminder that any call in the last three instructions of a hardware loop must return with the (LR) option.
Ea2022 – an arithmetic loop cannot end one instruction after the last instruction of a loop nested within the arithmetic loop
Ea2023 – In the 2136x and later processors, loops cannot contain operations on LPSTK and PCSTK.
Ea2024 – if the end instruction resides at a lower address than the do instruction, the assembler cannot validate the loop.

TAR 39444: RFRAME Instruction Only Used in Delay Slots (SHARC)

It has been reported by some customers that the rframe instruction is not, in fact, atomic, and it is possible for an interrupt to occur during the execution of the rframe instruction. If this happens, the function return sequence may execute incorrectly. To avoid this issue, the compiler will ensure that the rframe instruction is either used only in a delay slot, or it is not used. This change will affect code and projects that use the following compiler features:

- the "optimize for space (`-Os`)" command line switch
- the '`-no-db`' command-line switch
- '`#pragma no_db_return`'

TAR 40221: TRUNC Incorrect for Negative Underflow (SHARC)

There have been enhancements to the compiler support routines which mean that the TRUNC instruction is now used more often. Due to a simulator issue, some arithmetic operations (particularly division and modulus operations) may report incorrect answers when executed on the simulator.

TAR 40290: New Simulator Warning for __lib_prog_term (SHARC)

The CRT header source (`06x_hdr.asm for example`) defined function `___lib_prog_term`, executed after `_main` returns, used to be: `___lib_prog_term pm(__done_execution)=pc; idle; jump ___lib_prog_term;`

As of VisualDSP++ 5.0 Update 6, this code may cause the following simulator warning:

```
*** Write access to Read-only Address 0x900c1 in 32-bit space *** from instruction at PC
0x900be
```

The warning is valid and the write to `__done_execution` does not work. The warning indicates the issue on parts where the write is actually to ROM. The fix that has been done is to remove this store along with the associated `__done_execution` for all SHARC processors.

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.

Silicon Anomaly 05000412 (ADSP-BF561)

"TESTSET Instruction Causes Data Corruption with Writeback Data Cache Enabled"

The anomaly occurs when a TESTSET instruction is used to operate on L2 memory and there is data in external memory that is cached using write-back mode. The result is that data in L2 and/or external memory can become corrupted.

Runtime library workarounds for this anomaly were made in VisualDSP++ 5.0 Update 5. In Update 6 compiler and assembler support has been added.

The Update 6 compiler issues the required workaround sequence of code to avoid the errata for calls to the testset compiler built-in function when the workaround is enabled. This workaround is automatically enabled when building for parts and revisions impacted by the 05000412 anomaly or when "-workaround 05000412" is passed to the compiler. The macro __WORKAROUND_05000412 is defined when the workaround is enabled.

The assembler issues a detection warning ea5519 when a TESTSET instruction is not immediately preceded by an SSYNC (required by the 05000412 errata workaround). This detection warning is automatically enabled when building for parts and revisions impacted by the 05000412 anomaly or when "-anomaly-detect 05000412" is passed to the assembler. The macro __ASM_DETECT_05000412__ is defined when this anomaly detection is enabled.

Known limitations with this anomaly support:

- TAR40784 : "05000412 anomaly detection fails if TESTSET follows .align"

Silicon Anomaly 05000426 (ADSP-BF5xx)

"Speculative Instruction Fetches Can Cause Spurious Hardware Errors"

The anomaly occurs when there is an indirect jump or call through a pointer which may point to an invalid address on the opposite control flow of a conditional jump to the predicted taken path and ICPLBS are disabled. The result of this is potentially spurious hardware errors.

Blackfin C/C++ compiler, VDK and run-time libraries workarounds for anomaly 05000426 were added in VisualDSP++ 5.0 Update 5.

In Update 6 assembler anomaly detection support has been added. The assembler issues warning ea5518 if either of the two instructions following a not predicted taken conditional jump or the target of a predicted taken conditional jump is an indirect jump or call instruction. The anomaly detection is not automatically enabled as many users will have ICPLBS meaning that the errata is avoided. Users that do not use ICPLBS can enable the detection warning support by passing "-anomaly-detect 05000426" to the assembler. The macro __ASM_DETECT_05000426__ is defined when this anomaly detection support is enabled.

Known limitations with this anomaly support:

- TAR40786 : assembler-behavior information in BLACKFIN-EDN-anomaly.xml for 05000426 is not correct
- TAR40787 : -anomaly-detect 05000426 causes ea1222 "ID does not exist"

Silicon Anomaly 05000428 (ADSP-BF561)

"Lost Write to L2 Memory Following Speculative Read from L2 Memory"

The Blackfin C/C++ compiler, assembler, VDK and run-time libraries have been enhanced to include workarounds for anomaly 05000428.

The anomaly occurs when a write to L2 memory is followed by a speculative read from L2 memory in the shadow of a branch executed on core B. This results in the write being lost or corrupted.

The anomaly workaround was incomplete in Update 5. The following additional support is provided in Update 6:

Assembler

- Assembler detection of silicon anomaly 05000428 is enabled for silicon revision "any".

Compiler

- The compiler ensures that the targets of predicted jumps are safe against the anomaly.

Run-Time Libraries

- The source code for memcpy has been modified to work around this anomaly.
- The source code for zero_crossd has been modified to work around this anomaly.

VDK

- The code for the API PostMessage has been modified to work around this anomaly.

Silicon Anomaly 09000020 (ADSP-2137x)

"Wrong instruction address may be cached when PMDA instruction executing from external memory is interrupted"

The compiler will ensure a "FLUSH CACHE" instruction is inserted at the start of interrupt service routines - functions marked #pragma interrupt_complete and #pragma interrupt_complete_nesting (those marked simply #pragma interrupt contain the workaround in the appropriate interrupt dispatcher). This workaround is automatically enabled when building for parts and revisions impacted by the 09000020 anomaly or when "-workaround 09000020" is passed to the compiler. The macro __WORKAROUND_09000020 is defined when the workaround is enabled.

All runtime library interrupt dispatchers have been modified to workaround this anomaly using the "FLUSH CACHE" instruction. In VDK, all interrupt vector table entries now have "FLUSH CACHE" as their first instruction to workaround the anomaly.

Silicon Anomaly 09000023 (ADSP-2137x)

“Writes to LCNTR, CURLCNTR and LADDR from Internal Memory may fail if there is a DMA block conflict”

The anomaly is avoided by ensuring that writes to LCNTR, CURLCNTR occur in two stages, rather than loading them directly from memory. This workaround is used in the runtime libraries and VDK linked when building for parts and revisions affected by the 09000023 errata.

Known limitations with this anomaly support:

- TAR40789 : rtl-behavior description in SHARC-2137X-anomaly.xml confusingly references "DMA memory"

Anomaly Charts

Tools Anomalies Addressed

The following table is a list of tools anomalies addressed in VisualDSP++ 5.0 Update 6 for which details can be found on the public tools anomaly website. Other tools anomalies have also been fixed in the Update.

Details 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
All	39414	Compiler	MISRA issue with logical negative expression assigned to boolean
All	39921	Compiler	Bad dependencies for paths with drive letters and using '/'
Blackfin	39634	Compiler	asm() cannot load to A1
Blackfin	39783	Compiler	BF548 "Getting Started Examples" misaligned address violation
Blackfin	40179	Compiler	internal error (circbuf.c:258) building code with circindex
Blackfin	36774	Device Driver	SPI driver requires baud rate in slave mode
Blackfin	39698	Device Driver	Failure to create more than 507 files on FAT 32 volume
Blackfin	40541	Device Driver	adi_rawpid.h has no C++ linkage
Blackfin	36663	Emulator	SW Breakpoints Ignored Upon Return From Lockbox Authentication
Blackfin	39639	Emulator	BF533 POST failed at Ethernet test for EZ-USBLAN board
Blackfin	39939	Emulator	Windows - Driver Entry Point Not Found
Blackfin	39766	Examples	BF548 Driver Example UART AutoBaud readme.txt pb2 doesn't halt.
Blackfin	40469	Examples	BF518F example Flash programmer Internal SPI won't load dxe.
Blackfin	40592	Examples	BF526 POST example readme for USB Peripheral/ Host confusing.
Blackfin	40073	Linker	LDF SIZEOF macro reports incorrect value when using RESOLVE
Blackfin	40024	Loader	Compression: the later part of user application is lost in LDR
Blackfin	38085	Run Time Libraries	float16: negate_fl16 is incorrect
Blackfin	38090	Run Time Libraries	strtoull may give incorrect result
Blackfin	39572	Run Time Libraries	li2152 due to missing Instrumented Profiling support for BF51x
Blackfin	39623	Run Time Libraries	no prototype for adi_acquire_lock adi_try_lock with -no-builtin
Blackfin	39636	Run Time Libraries	memcpy() not safe against IC anomaly 05000428
Blackfin	39637	Run Time Libraries	zero_crossd() not safe against IC anomaly 05000428

Blackfin	39681	Run Time Libraries	libdsp FFT functions fail for sizes > 32K
Blackfin	39705	Run Time Libraries	Default 561 CPLB data table causes CPLB exception
Blackfin	39844	Run Time Libraries	meminit for BF53[467], BF51x, BF52x BF54x revision none fails
Blackfin	40186	Run Time Libraries	PLL_STAT macro VSTAT wrong in defBF561.h and defBF532.h
Blackfin	39516	Source Generator	Force Regen Project:Link:Processor:Use C++ exceptions libraries
Blackfin	39712	Source Generator	Custom clock settings in Project Option is not proper for BF561.
Blackfin	39701	System Services	FSS locks when random seeking is performed in a file
Blackfin	39896	System Services	Files created on removable media cannot be deleted in Windows
Blackfin	40245	System Services	directory name truncated during creation
Blackfin	40593	System Services	defect in adi_ebiu_SelfRefreshEnable
Blackfin	39625	USB Stack	USB detection issues on BF548 for CCLK greater than 400MHz
Blackfin	39641	VDK	VDK does not work around all instances of anomaly 05000428
SHARC	39444	Compiler	rframe instruction is not atomic, but compiler acts as if it is
SHARC	39664	Compiler	internal error at bitmatrix.c:81, -restrict-hardware-loops 0
SHARC	40089	Compiler	"#pragma interrupt_complete" functions corrupt MRF registers
SHARC	38048	Loader	Memory overlap when PM segment is too big
SHARC	38082	Run Time Libraries	scanf/strtold may not convert long double hexadecimal fp data
SHARC	39482	Run Time Libraries	BR glitch anomalies in assembler library sources
SHARC	40045	Run Time Libraries	matsaddf, matsmltf, and matssubf may return a wrong result
SHARC	40164	Run Time Libraries	cfftN, ifftN, rfftN in LIBDSP not safe against anomaly 07000010
SHARC	40290	Run Time Libraries	lib_prog_term causes simulator warning
SHARC	39857	Simulator	Does not zero unspecified upper bits of XML register resets
TigerSharc	39450	Run Time Libraries	heap_switch() reports error switching to last run-time heap.
TigerSharc	39610	Run Time Libraries	heap_realloc() doesn't work for first run time heap.

Known Tools Anomalies

Details can be found on the Tools Anomaly Web page. The URL is:

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

VisualDSP++® 5.0 Update 5 Release Notes

Revision 1.8

November 21, 2008

Table of Contents

Nomenclature

In the past, VisualDSP++ updates were labeled by the month and year of their release. In order to improve clarity, updates are now numbered (e.g., Update 1, Update 2, etc.).

Release Notes

These release notes subsume the release notes for previous updates. Release notes for previous updates can be found at the end of this document.

Installation

This update can only be installed on a previous VisualDSP++ 5.0 installation. If VisualDSP++ 5.0 is not installed, please install it first. Installation on a previous update is permitted. If a newer update has already been installed, please do not install this update. This update is not intended to be installed on alpha or beta releases.

Identifying Your VisualDSP++ Version

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

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

Installing the Update

Follow the instructions below for installing this update. Please note that since VisualDSP++ supports having multiple instances installed on a single system. See the *Cloning VisualDSP++* section below for more information.

9. Use the Start Menu to navigate to VisualDSP++ "Maintain this installation". By default, this is at Start Menu → All Programs → Analog Devices → VisualDSP++ 5.0.
10. Select "Go to the Analog Devices website" and click Next. This will open a window in your web browser.
11. Select the appropriate Processor Software Tools Upgrades to match your processor.
12. Select and download the desired update (VisualDSP++ 5.0_Update5.vdu) to your hard drive.
13. Again, use the Start Menu to navigate to VisualDSP++ "Maintain this installation".
14. Select "Apply a downloaded Update" and click Next.
15. Browse for the downloaded Update file (VisualDSP++ 5.0_Update5.vdu) and click Next.
16. Follow the on-screen prompts to complete installation of this Update.

Cloning VisualDSP++

VisualDSP++ supports cloning of an existing installation. A clone of an installation creates a new instance of a product from an existing installation, rather than from a CD or web software distribution. The use of clones allows you to maintain multiple versions of VisualDSP++ on the same PC at different update levels, and provides a risk-free way to "test" new updates or patches.

To clone your existing installation of VisualDSP++:

5. Go to Start->Programs->Analog Devices->VisualDSP++ 5.0 (or equivalent)->Maintain this Installation
6. Select "Clone this Installation" and click Next.
7. Optionally click Advanced to set the Start menu path.
8. Enter the Clone install path and click Next.

Definitions

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

TAR – Tools Anomaly Reference Number

Tools Anomaly Reference Number, or TAR, is used for tracking confirmed defect reports in VisualDSP++.

New Hardware Support

VisualDSP++ updates often include support for new processors, new silicon revisions for existing processors and new EZ-KIT Lite® 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.

New Processors and Processor Revision Support

This section lists new processors and processor revisions available in this update. Refer to the data sheets and hardware reference manuals for information on system configuration, peripherals, registers, and operating modes.

Update 5 introduces two new processors to the ADSP-BF51x Blackfin® processor family. In addition to the ADSP-BF512 and ADSP-BF514 processors, the following new processors are supported:

- ADSP-BF516* silicon revision 0.0
- ADSP-BF518* silicon revision 0.0

*Please note that the previous ADSP-BF516 processor was renamed shortly before release to be the ADSP-BF518. The ADSP-BF518 is newly supported in Update 5. A new ADSP-BF516 processor has been introduced and is supported in Update 5. Please refer to the processor datasheet for details on the new ADSP-BF516 processor.

Update 5 also introduces support for the Mobile DDR variant of the ADSP-BF54x Blackfin processor family. The Mobile DDR variants are provided separate processor names in VisualDSP++ for full support:

- ADSP-BF542M silicon revision 0.3
- ADSP-BF544M silicon revision 0.3
- ADSP-BF547M silicon revision 0.3
- ADSP-BF548M silicon revision 0.3
- ADSP-BF549M silicon revision 0.3

Update 5 provides support for the following silicon revisions to existing Blackfin® processors:

- ADSP-BF522 silicon revision 0.1
- ADSP-BF524 silicon revision 0.1
- ADSP-BF526 silicon revision 0.1

There are no new silicon revisions to existing SHARC® or TigerSHARC® processors with Update 5. Initial support for the ADSP-2146x SHARC processor family will be provided in a separate release.

Processor Revision Deprecation

Support for the following silicon revisions will be deprecated in Update 5.

- ADSP-TS201 silicon revision 0.1
- ADSP-TS202 silicon revision 0.1
- ADSP-TS203 silicon revision 0.1

New Evaluation Board Support

ADSP-BF518F EZ-KIT Lite®

Update 5 introduces initial support for the ADSP-BF518F EZ-KIT Lite. The Power On Self Test (POST) and Flash Programmer are provided with this release.

New System Services and Device Drivers

The following are now supported by VisualDSP++ 5.0:

NAND Flash Access

Support for the NAND flash access has been extended to support the ADSP-BF526 EZ-KIT Lite.

New Examples

ADSP-BF518F EZ-KIT Lite Examples

VisualDSP++ 5.0 Update 5 includes initial support for the ADSP-BF518F EZ-KIT Lite evaluation system. The following examples can be found in the following directories:

```
Blackfin\Examples\ADSP-BF518F EZ-Board\Flash Programmer\Parallel
Blackfin\Examples\ADSP-BF518F EZ-Board\Flash Programmer\Serial
Blackfin\Examples\ADSP-BF518F EZ-Board\Power_On_Self_Test
```

ADSP-BF526 EZ-KIT Lite Examples

VisualDSP++ 5.0 Update 5 provides a LockBox example for the ADSP-BF526 EZ-KIT Lite. The example can be found in the following directory:

```
Blackfin\Examples\ADSP-BF526 EZ-KIT Lite\lockbox
```

Landscape LCD EZ-EXTENDER Examples

VisualDSP++ 5.0 Update 5 provides an LCD example for the ADSP-BF526, ADSP-BF527 and ADSP-BF548 EZ-KIT Lites:

```
Blackfin\Examples\Landscape LCD EZ-EXTENDER\LCD_ColorBarDisplay
```

Critical Fixes/Changes

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

New Blackfin Compiler Switches

New Blackfin Compiler Switches: -icplbs & -dcplbs

-icplbs

The -icplbs (Instruction CPLBs are active) switch instructs the compiler to assume that all instruction memory accesses will be validated by the Blackfin processor's memory protection hardware. This allows the compiler to identify situations where the cacheability protection lookaside buffers (CPLBs) will avoid issues the compiler would otherwise workaround (e.g. anomaly 05-00-0426), improving code size and performance.

-dcplbs

The -dcplbs (Data CPLBs are active) switch instructs the compiler to assume that all data memory accesses will be validated by the Blackfin processor's memory protection hardware. This allows the compiler to identify situations where the cacheability protection lookaside buffers (CPLBs) will avoid issues the compiler would otherwise workaround (e.g. anomaly 05-00-0428), improving code size and performance.

If both ICPLBs and DCPLBs are active, the -cplbs switch should still be used

Feature Macros (Blackfin)

The <feature-macros> block in the `System\ArchDef\*-compiler.xml` files contain macros that the assembler and compiler automatically pre-define. New feature macros have been added for processor family names for standardization. To maintain backwards compatibility, no preexisting feature macros have been deleted. For more details, see below:

Family Macro (New at Update 5)	Group Macro	Processors
__ADSPBF518_FAMILY__	__ADSPBF51x__	ADSP-BF512, ADSP-BF514, ADSP-BF516 , ADSP-BF518
__ADSPBF526_FAMILY__	__ADSPBF52xLP__	ADSP-BF522, ADSP-BF524, ADSP-BF526
__ADSPBF527_FAMILY__	__ADSPBF52x__	ADSP-BF523, ADSP-BF525, ADSP-BF527
__ADSPBF533_FAMILY__	__ADSPBF53x__	ADSP-BF531, ADSP-BF532, ADSP-BF533
__ADSPBF535_FAMILY__	Not available	ADSP-BF535
__ADSPBF537_FAMILY__	__ADSPBF53x__	ADSP-BF534, ADSP-BF536, ADSP-BF537
__ADSPBF538_FAMILY__	__ADSPBF53x__	ADSP-BF538, ADSP-BF539
__ADSPBF548_FAMILY__	__ADSPBF54x__	ADSP-BF542, ADSP-BF544, ADSP-BF547, ADSP-BF548, ADSP-BF549
__ADSPBF548M_FAMILY__	__ADSPBF54x__	ADSP-BF542M, ADSP-BF544M, ADSP-BF547M, ADSP-BF548M, ADSP-BF549M
__ADSPBF561_FAMILY__	__ADSPBF56x__	ADSP-BF561

NAND Boot Release notes

NAND Boot Command-line Option

-b NAND

Appending 256 byte End Block as Ignore Block

The Blackfin loader appends an ignore block containing 256 bytes (all zeros) to avoid prefetch CRC error.

For the ADSP-BF54x, ADSP-BF52x and the ADSP-BF54xM the loader stream will be appended with 256 bytes of data. This is because the boot kernel uses a prefetch mechanism and while processing one 256 block of data it will fetch in the next 256 byte block of data. Padding the loader stream with an additional 256 bytes at the very end ensures that the 256 byte block of data following the final block of the loader stream is programmed and the error correction parity data is written. Appending this block prevents the boot from failing at the very end of the boot cycle as data will be fetched (although never actually required) and it will have valid ECC parity data resulting in the successful completion of the boot.

Coexistence of NAND Boot and File System Support

The ADI file system service (FSS) provides support for the NAND flash device (NFD) to be formatted as a file system and leverages a flash translation layer (FTL) to provide wear leveling and bad block management. To cater for both NAND boot and FSS support, the FTL is instructed to manage only a portion of the NFD beyond a specified reserved area at the beginning of the NAND array. This reserved area starting from block 0 is provided for NAND boot purposes. The size of the reserved area is determined upon format and the following examples are provided to format the NFD as a FAT 16 volume:

```
($ADI_DSP)\Blackfin\Examples\ADSP-BF548 EZ-KIT Lite\Services\File System\NAND\NandFormat
($ADI_DSP)\Blackfin\Examples\ADSP-BF527 EZ-KIT Lite\Services\File System\NAND\NandFormat
($ADI_DSP)\Blackfin\Examples\ADSP-BF526 EZ-KIT Lite\Services\File System\NAND\NandFormat
```

The default reserved area size is 10 blocks. For the EZ-KIT implementations this provides 10 blocks of 64 pages of 2112 bytes per page = 1.3MB (approx).

If this default is to be overridden it is important that the ADI_NAND_CMD_SET_RESERVED_SIZE command is passed to the FSS NAND Physical Interface Driver (PID) with the required reserved size upon format and every time it is required for file system access. For example if the NAND boot required twice as many blocks you would need to pass the following command pair to the NAND PID on format as well as every time file system access is required:

```
{ ADI_NAND_CMD_SET_RESERVED_SIZE, (void*)20 }
```

This command-value pair must be passed in the configuration stage of the NAND PID. Please refer to the NAND PID documentation for further details:

```
($ADI_DSP)\Blackfin\docs\drivers\pid\nand\adi_nand.pdf
```

For NAND and OTP boot, please refer to Table 2-3 in the “VisualDSP++ 5.0 Loader and Utilities Manual”.

Implicit Push STS Handler

Support for a new pragma, implicit_push_sts_handler has been added to the SHARC compiler so that the compiler does not generate an explicit PUSH and POP of STS for interrupt handlers. This only applies when compiling for Hammerhead parts (211xx, 212xx, 213xx, and 214xx). When compiling for non-Hammerhead parts, the pragma is silently ignored.

The pragma will only take effect when it is used in conjunction with one of the SHARC/Hammerhead interrupt pragmas e.g. interrupt_complete.

The compiler is not able to determine if the handler the pragma is applied to a handler for the IRQ, VIRPTL or TIMER interrupts. It will be up to the user to determine whether or not the pragma should be used.

Syscontrol() variation for ADSP-BF54xM rev 0.3

For the ADSP-BF54xM rev 0.3, the syscontrol() function in ROM increments the VLEV parameter before setting the VR_CTL register.

TAR 36697: MISRA Rule 19.4 Change

MISRA Rule 19.4 Change: Checks on #define directive and not macro expansion use

MISRA Rule 19.4 states that C macros shall only expand to a braced initializer, a constant, a string literal, a parenthesized expression, a type qualifier, a storage class specifier, or a do-while-zero construct.

Prior to VisualDSP++ 5.0 Update 5 the compiler would check this rule in macros that were used. To improve the MISRA compliance support the checks are now done when a macro is defined.

One effect of this is that macros that are defined but unused may now cause rule 19.4 errors when they did not using the tools prior to Update 5. Another impact is that macros that are valid when expanded on use may not be valid and cause Rule 19.4 errors when defined. For example:

```
 #define ONE 1
 #define ANOTHER_ONE ONE /* This violates rule 19.4 even though when expanded will result as
the constant 1. */
```

The example is corrected by parenthesizing ONE in the definition of ANOTHER_ONE.

TAR 38060: ADSP-BF52x Macros Removed

The NONGPIO_SLEW, PORTF_SLEW, PORTG_SLEW, and PORTH_SLEW macros previously defined by including the defBF52x.h and cdefBF52x.h headers have been removed in Update 5. They should not be used. Any application source that contains them will need to be changed.

TAR 37863: Run-Time Library uses dual-data move

"Some run-time library functions use dual-data move instructions (213xx)"

Page 3-78 of the ADSP-21368 SHARC Processor Hardware Reference manual (which includes the ADSP-21367, ADSP-21369, ADSP-21371, and ADSP-21375) documents some restrictions with accessing external memory. One of these restrictions is that in a dual-data move instruction, both accesses should not be to external memory. This restriction is not observed by some functions in the run-time library, in particular those functions that operate on arrays and vectors that are allocated in Program Memory (PM).

This issue has been resolved in Update 5 apart from the biquad, fir, fir_decima, fir_interp, and iir filter functions. Addressing the issue in the filter functions would have a severe impact on their performance and so rather than modifying the functions; the following restriction on their use of external data has been imposed:

When running on an ADSP-21367, ADSP-21368, ADSP-21369, ADSP-21371, or ADSP-21375 processor, both the filter coefficients and the delay line must not be allocated in external memory otherwise the function can generate an incorrect set of results. This is because in a dual-data move instruction, the hardware does not support both memory accesses being to external memory. Therefore, ensure that the filter coefficients or the delay line (or optionally both) are allocated in internal memory when running on one of the ADSP-213xx processors specified above.

Changes have been made to the VisualDSP++ 5.0 Run-Time Library Manual for SHARC Processors to document this restriction.

TAR 39783: Misaligned address violation

"BF548 "Getting Started Examples" misaligned address violation"

Using the "-section" C/C++ compiler switch or "#pragma default_section" to move "alldata" or "constdata" to a non-default memory section may cause a data access misaligned address violation run-time exception. The compiler generated assembly code is not always correctly aligning the non-default section which the data is placed into.

The use of these methods for placing "alldata" and "constdata" need to be avoided to workaround this problem. Alternate placement can be performed in the LDF.

The ADSP-BF548 "Blackfin\Examples\ADSP-BF548 EZ-KIT Lite\Getting Started Examples\Example_2" example exhibits this problem when built in release mode.

TAR 39756: Some examples cause hardware error

"BF548 NAND format example causes hardware error"

When building and running the ADSP-BF548 NAND format example "`Blackfin\Examples\ADSP-BF548 EZ-KIT Lite\Services\File System\NAND\NandFormat`" for the first time, a hardware error interrupt may occur. This may also occur with other examples. This is a configuration issue.

To resolve this problem, change the configuration from Debug to Release, rebuild the project and the example will run successfully. After the first successful run, the configuration may be returned to Debug and the example will continue to run successfully.

Limitations

This section highlights known significant limitations

ADSP-BF51x Silicon Revision Support

For the ADSP-BF51x, only use the silicon revision 0.0. Silicon revisions ”none” and ”any” are not supported in Update 5. Attempts to use “none” or “any” for ADSP-BF51x in Update 5 will result in link-time errors relating to MEM_L1_CODE.

OTP Boot

Preliminary work for OTP Boot is in Update 5. This feature will be available for use in a future Update.

Silicon Anomaly Workarounds

The file `System\ArchDef\BLACKFIN-EDN-anomaly.xml` has been modified to include anomaly workarounds specific to the system services and device driver libraries.

Anomaly workaround information is still available in the online help: Select Help → Contents → Graphical Environment → Silicon Anomaly Support → Silicon Anomalies Tools Support and then select the appropriate processor family.

Silicon Anomaly 05000412 (ADSP-BF561)

"TESTSET Instruction Causes Data Corruption with Writeback Data Cache Enabled"

The Blackfin runtime libraries have been enhanced to include workarounds for anomaly 05000412.

The anomaly occurs when a TESTSET instruction is used to operate on L2 memory and there is data in external memory that is cached using writeback mode. The result is that data in L2 and/or external memory can become corrupted.

The runtime libraries that are linked in when building for impacted parts and silicon revisions have been modified to avoid the anomaly. The workaround involves preceding TESTSET instructions with a dummy read and an SSYNC instruction.

Assembler detection and modifications to the compiler's testset built-in function will be provided in a future update.

Silicon Anomaly 05000426 (ADSP-BF5xx)

"Speculative Instruction Fetches Can Cause Spurious Hardware Errors"

The Blackfin C/C++ compiler, VDK and runtime libraries have been enhanced to include workarounds for anomaly 05000426.

The anomaly occurs when there is an indirect jump or call through a pointer which may point to an invalid address on the opposite control flow of a conditional jump to the predicted taken path and ICPLBS are disabled. The result of this is potentially spurious hardware errors.

The compiler works around this anomaly by not generating indirect call or jump instructions in the 2 instruction slots following a conditional jump for impacted parts, unless either the "-icplbs" or "-cplbs" switches are used.

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

Assembler detection for this anomaly will be provided in a future update.

Silicon Anomaly 05000428 (ADSP-BF561)

"Lost Write to L2 Memory Following Speculative Read from L2 Memory"

The Blackfin C/C++ compiler, assembler, VDK and runtime libraries have been enhanced to include workarounds for anomaly 05000428.

The anomaly occurs when a write to L2 memory is followed by a speculative read from L2 memory in the shadow of a branch executed on core B. This results in the write being lost or corrupted.

The compiler works around this anomaly by not generating potentially problematic reads in the 2 slots following a conditional jump for any impacted parts. The compiler will allow reads from MMR's or external memory, if they can be identified as such, to remain in the 3 slots following the conditional jump. The compiler does not currently avoid the placement of potentially problematic reads in the instruction following the target of a predicted not taken branch.

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

The support for this anomaly workaround is incomplete. The ADI components affected are:

Assembler

- Assembler detection of silicon anomaly 05000428 is not enabled for silicon revision "any". "any" is intended to enable all workarounds for the processor. The detection can be obtained by explicitly building with -si-revision 0.4 or -si-revision 0.5. In future updates, detection for this anomaly will be enabled by default for "any".

Compiler

- The compiler will not ensure that the targets of predicted jumps are safe against the anomaly.

Runtime Libraries

- The source code for memcpy does not work around the anomaly. To solve this you can edit the code provided in the VisualDSP install in the file Blackfin\lib\src\libc\memcpy.asm. For more information see tools anomaly 39636.
- The source code for zero_crossd does not work around the anomaly. To solve this you can edit the code provided in the VisualDSP install in the file Blackfin\lib\src\libdsp\zero_crossd.asm. For more information see tools anomaly 39637.

VDK

- The code for the API PostMessage does not work around the anomaly. To avoid hitting the anomaly, place the variable tmk in L1 memory. For more information see tools anomaly 39641.

Warning

The assembler detection warning (ea5517) for anomaly 05000428 will be triggered by code that contains the prescribed workaround for anomaly 05000283 (System MMR Write Is Stalled Indefinitely when Killed in a Particular Stage). This case of this warning can be safely ignored and the warning may be suppressed using the .MESSAGE directive as the code will not cause the 05000428 anomaly.

Anomaly Charts

Tools Anomalies Addressed

The following table is a list of tools anomalies addressed in VisualDSP++ 5.0 Update 5 for which details can be found on the public tools anomaly website. Other tools anomalies have also been fixed in the Update.

Details 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
All	36599	Compiler	Compiler saves, restores reserved registers if they are used in asm
All	36678	Compiler	MISRA Rule 16.7 incorrectly reported for pointer to function
All	36697	Compiler	MISRA Rule 19.4- Check #define directive,not expansion.
All	36910	Compiler	MISRA Rule 10.1(a) violation incorrectly reported for += operation
All	36535	Run Time Libraries	Call to sysreg_read results in MISRA Rule 19.7 warning
All	38001	VDK	Thread's msg queue should initialize fields to 0
Blackfin	36575	Assembler	Assembler allows illegal instruction Preg = RETS on 535
Blackfin	36507	Compiler	incorrect register allocation of multiply inlined asm statements
Blackfin	36562	Compiler	rvitmax builtin doesn't zero A0.x before storing history to it
Blackfin	36909	Compiler	writes to a pointer removed inconsistently
Blackfin	37970	Emulator	AD7879 chip is new and causes LCD Touch example to not work.
Blackfin	38065	Flash Programmer	BF527 & BF548 programmer fail if "Verify while programming"
Blackfin	38078	Flash Programmer	Serial FLASH Programmer (Driver) not functional
Blackfin	34201	IDDE	incorrect variable used in Automation API example
Blackfin	34263	IDDE	Enabling custom board support through automation doesn't work
Blackfin	36922	IDDE	PGO example failing
Blackfin	32741	Installation	installer is not cleaning up HPPCI driver INFs
Blackfin	26133	Loader	Specified addr from -p did not get into the .ldr file with -init
Blackfin	33934	Run Time Libraries	float16.h issues with negate_fl16 and add_fl16
Blackfin	36729	Run Time Libraries	LIBDSP twiddle table generators can overflow stack
Blackfin	36825	Run Time Libraries	ADSP-BF51x SD_CARD_DET_MASK of RSI_EMASK register Incorrect
Blackfin	36871	Run Time Libraries	l1_memcpy and memcpy_l1 use cli and sti incorrectly
Blackfin	37859	Run Time Libraries	Misra rule 8.8 error in bfrom.h: __aes_init prototype duplicated
Blackfin	38060	Run Time Libraries	Remove *_SLEW registers from BF52x def/cdef headers
Blackfin	36343	Simulator	Accumulator Signbits simulates incorrectly for input of 0, -1
Blackfin	37924	Simulator	Floating point 32 bit memory display gives incorrect value
Blackfin	37976	Simulator	VIT_MAX appears to give the wrong result in compiled sim

Blackfin	36872	Source Generator	quoted library search directories causes malformed input XML
Blackfin	37852	Source Generator	a multi-core project should always have C++ support enabled.
Blackfin	32230	System Services	Add command to sense GP timer period
Blackfin	36352	System Services	Kookaburra SPI System Service has a bad select ID
Blackfin	37986	TCPIP Stack	lwIP init_stack() clears all other flags when setup IGMP flag
Blackfin	37987	TCPIP Stack	lwIP nifce_driver_init() should set up more flags
Blackfin	37988	TCPIP Stack	lwIP header files under /include folder should be updated
Blackfin	37989	TCPIP Stack	BF526 LAN Software Readme file need be updated
Blackfin	36860	VDK	VDK headers should not include RTL headers in extern "C" blocks
SHARC	36427	ADspCommon XML Files	Cannot view the DMA addressing registers for SPIB on the 21364
SHARC	36536	Compiler	Call to sysreg_read results in MISRA Rule 19.4 error
SHARC	36540	Compiler	Compiler uses a POP STS instruction in a RTI(DB) delay slot
SHARC	36621	Compiler	bad optimization of conditional stores
SHARC	34165	Run Time Libraries	using saturate.h will result in errors
SHARC	36489	Run Time Libraries	strncat() incorrect if strings in different memory banks
SHARC	36598	Run Time Libraries	The strncat() implementation for DM to PM is flawed
SHARC	36600	Run Time Libraries	Strncat() of NULL string behaves incorrectly
SHARC	36709	Run Time Libraries	division support funcs incorrect when placed in external memory
SHARC	36770	Run Time Libraries	MISRA violation 8.8 with string.h and builtins.h
SHARC	36914	Run Time Libraries	Byte-address mode qsort() multi-thread can cause failure
SHARC	36919	Run Time Libraries	FFT functions read beyond the end of an array
SHARC	37382	Run Time Libraries	64-bit double addition with results close to zero (denorm) wrong
SHARC	37863	Run Time Libraries	Some run-time library functions use dual-data move instructions
SHARC	35592	Simulator	Interrupt during abort of an arithmetic loop never serviced

Known Tools Anomalies

Details can be found on the Tools Anomaly Web page. The URL is:

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

VisualDSP++® 5.0 Update 4 Release Notes

Revision 1.2

September 3, 2008

Table of Contents

Nomenclature

In the past, VisualDSP++ updates were labeled by the month and year of their release. In order to improve clarity, updates are now numbered (e.g., Update 1, Update 2, etc.).

Release Notes

These release notes subsumes the release notes for previous updates. Release notes for previous updates can be found at the end of this document.

Installation

This update should only be installed after installing the VisualDSP++ 5.0 base release. If VisualDSP++ 5.0 is not installed, please install it first. Installation on a previous update is permitted. If a newer update has already been installed, please do not install this update. This update is not intended to be installed on alpha or beta releases.

Identifying Your VisualDSP++ Version

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

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

Installing the Update

Please follow the instructions below for installing this update. Please note that since VisualDSP++ supports having multiple instances installed on a single system, you can install this update on top of one instance while maintaining the previous installation.

17. Use the Start Menu to navigate to VisualDSP++ “Maintain this installation”. By default, this is at Start Menu → All Programs → Analog Devices → VisualDSP++ 5.0.
18. Select “Go to the Analog Devices website” and click Next. This will open a window in your web browser.
19. Select the appropriate Processor Software Tools Upgrades to match your processor.
20. Select and download the desired update (VisualDSP++ 5.0_Update4.vdu) to your hard drive.
21. Again, use the Start Menu to navigate to VisualDSP++ “Maintain this installation”.
22. Select “Apply a downloaded Update” and click Next.
23. Browse for the downloaded Update file (VisualDSP++ 5.0_Update4.vdu) and click Next.
24. Follow the on-screen prompts to complete installation of this Update.

Cloning VisualDSP++

VisualDSP++ supports cloning of an existing installation. A clone of an installation creates a new instance of a product from an existing installation, rather than from a CD or web software distribution. The use of clones allows you to maintain multiple versions of VisualDSP++ on the same PC at different update levels, and provides a risk-free way to "test" new updates or patches.

To clone your existing installation of VisualDSP++:

9. Go to Start->Programs->Analog Devices->VisualDSP++ 5.0 (or equivalent)->Maintain this Installation
10. Select "Clone this Installation" and click Next.
11. Optionally click Advanced to set the Start menu path.
12. Enter the Clone install path and click Next.

Definitions

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

TAR – Tools Anomaly Reference Number

Tools Anomaly Reference Number, or TAR, is used for tracking confirmed defect reports in VisualDSP++.

New Hardware Support

VisualDSP++ updates often include support for new processors, new silicon revisions for existing processors and new EZ-KIT Lite® 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.

New Processors and Processor Revision Support

The New Product Release Bulletin contains the list of new processors available with VisualDSP++ 5.0. Refer to the processor's data sheet and hardware reference manuals for information on system configuration, peripherals, registers, and operating modes.

Update 4 introduces the ADSP-BF51x Blackfin® processor family. The following new processors are supported:

- ADSP-BF512 silicon revision 0.0
- ADSP-BF514 silicon revision 0.0
- ADSP-BF516* silicon revision 0.0

*Please note that the ADSP-BF516 processor was renamed shortly before release as the ADSP-BF518. Users developing for the ADSP-BF518 should select the ADSP-BF516 processor. This name change will be reflected in Update 5. Also in Update 5, a new ADSP-BF516 processor will be introduced. Please refer to the processor datasheet for details on the new ADSP-BF516 processor.

Update 4 provides support for the following silicon revisions to existing Blackfin® processors:

- ADSP-BF542 silicon revision 0.2
- ADSP-BF544 silicon revision 0.2
- ADSP-BF547 silicon revision 0.2
- ADSP-BF548 silicon revision 0.2
- ADSP-BF549 silicon revision 0.2

There are no new silicon revisions to existing SHARC® or TigerSHARC® processors with Update 4.

New Evaluation Board Support

ADSP-BF526 EZ-KIT Lite®

Update 4 introduces support for the ADSP-BF526 EZ-KIT Lite. The ADSP-BF526 EZ-KIT Lite is the first in the new standard of EZ-KITs. There are two expansion slots for easy stacking of extender boards. The board itself is now labelled EZ-Board™ instead of EZ-KIT Lite. EZ-KIT Lite refers to the package, not the board itself. Instead of having the debug agent designed into the board, the Standalone Debug Agent (SADA) is officially released, and is paired with the ADSP-BF526 EZ-Board. This new technology allows the reuse of the SADA board with multiple ADSP-BF526 EZ-Board and future EZ-Board offerings.

Landscape LCD EZ-Extender®

Update 4 also introduces initial support for the Landscape LCD EZ-Extender which connects to the ADSP-BF526, ADSP-BF537, ADSP-BF538 and ADSP-BF548 EZ-KIT Lites as well as other current and future EZ-KIT Lites. This extender board will allow customers to quickly interface a landscape LCD to their own custom board for evaluation. This extender board also includes a touch screen, capacitance touch push buttons and a scroll wheel. Visit the website for more information: http://www.analog.com/en/embedded-processing-dsp/blackfin/BF-EXTENDERLCD/processors/product.html.

New System Services and Device Drivers

The following are now supported by VisualDSP++ 5.0:

NAND Flash Access

A new physical interface driver (PID) is released for use with both the file system service and the USB mass storage device (MSD) class driver to enable NAND flash devices (NFD) to be accessible from both an embedded application and externally from a Host PC.

The NAND PID requires the use of a flash translation layer (FTL) to affect wear-leveling and bad block management. To this end an FTL has been licensed from HCC-Embedded (www.hcc-embedded.com) for evaluation use with EZ-KIT Lite platforms only. As such, the FTL is distributed as a binary library only. Usage beyond this requires a full license that can be obtained upon contacting HCC-Embedded. The installation of Update 4 contains a license dialog to this effect which you must accept before installation can be completed.

A new example is provided to prepare the NFD for use by the FTL and to format it as a FAT 16 volume. Please see the New Examples section for further details. Once formatted, the NFD can be mounted from the shell_browser application. This is done by default in the ADSP-BF527 EZ-KIT Lite example but needs to be selected as an alternative to the SD card for the equivalent ADSP-BF548 EZ-KIT Lite example; this is because the NAND flash controller shares a DMA channel with the SD host controller. To select the NAND, ensure that _USE_NAND_ macro is set in the Project Options.

Improved FAT driver

To provide better throughput performance on removable media such as USB flash drives and SD cards, the FAT driver has been upgraded to use memory caches for the File Allocation Table (FAT) and directory information. These caches are allocated from the cache heap assigned to the FAT driver. The default settings are to provide 256 (512-byte) sectors of FAT cache and 16 clusters (size depends on media) of directory information, and can be changed with ADI_FAT_CMD_SET_FAT_CACHE_SIZE and ADI_FAT_CMD_SET_DIR_CACHE_SIZE commands.

IMPORTANT: Please note that for optimal benefit these caches are only synchronized with the media upon closure of an open file; halting and restarting an application while a file is open for write access can result in data loss. The ADI_FAT_CMD_SET_CACHE_PERFORMANCE command can be used to change this default behavior to either synchronize immediately on cache changes for maximum data integrity or to synchronize only when media is unmounted. Extreme caution is advised for the latter option.

Power Management Change

The power management service for the ADSP-BF54x revisions 0.1 and 0.2 have been modified to call the Boot ROM function 'bfrom_syscontrol()'. This function safely programs the VR and PLL registers, incorporating trim offsets values that have been stored in the OTP.

New Examples

ADSP-BF526 EZ-KIT Lite Examples

VisualDSP++ 5.0 Update 4 includes full support for the ADSP-BF526 EZ-KIT Lite evaluation system. Examples can be found in the following directory:

```
Blackfin\Examples\ADSP-BF526 EZ-KIT Lite
```

Landscape LCD EZ-EXTENDER Examples

VisualDSP++ 5.0 Update 4 includes initial support of the Landscape LCD EZ-Extender. There is sample source code available for the LCD, touch screen and capacitance touch. Additional support will be available in future updates.

```
Blackfin\Examples\Landscape LCD EZ-EXTENDER\Power_On_Self_Test\LCD
Blackfin\Examples\Landscape LCD EZ-EXTENDER\Power_On_Self_Test\TouchScreen
Blackfin\Examples\Landscape LCD EZ-EXTENDER\Power_On_Self_Test\CapTouch
```

NAND Flash Access Examples

VisualDSP++ 5.0 Update 4 includes a new example to prepare and format the NAND flash device (NFD) on the ADSP-BF527 EZ-KIT Lite and ADSP-BF548 EZ-KIT Lite. The NFD is formatted as a single FAT 16 partition. They can be found in the following directories:

```
Blackfin\Examples\ADSP-BF527 EZ-KIT Lite\Services\File System\NAND\NandFormat
Blackfin\Examples\ADSP-BF548 EZ-KIT Lite\Services\File System\NAND\NandFormat
```

LwIP TCP/IP Example

LwIP TCP/IP stack has been upgraded to 1.3.0 which includes multicasting support. Multicasting is a mechanism to send messages to group of destination addresses among networked machines. An example, Multicast_Sender, is included in the ADSP-BF526 EZ-KIT Lite LAN examples to demonstrate how to make multicast connection. It can be found in the following directory:

```
Blackfin\Examples\ADSP-BF526 EZ-KIT Lite\LAN\Multicast_Sender
```

Critical Fixes/Changes

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

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). As LwIP 1.3 supports multicast, multicast has been added to the Ethernet drivers.

Boot ROM API Additions (ADSP-BF51x, ADSP-BF52x, ADSP-BF54x)

In certain Blackfin products, optimized software implementations of some cryptographic algorithms are stored in ROM. These algorithms are C-callable with an API that is documented in the respective Hardware Reference Manual (HRM) and defined in the Boot ROM header `Blackfin\include\bfrom.h`:

1. In the Blackfin ADSP-BF54x processors, the algorithms available are SHA-1, AES, and ARC4. In these devices, the cryptographic algorithms are stored in Instruction ROM (IROM), which operates in the CCLK domain.
2. In the Blackfin ADSP-BF52x and ADSP-BF51x processors, the only cryptographic algorithm available is SHA-1. In these processors, SHA-1 is stored in Boot ROM, which operates in the SCLK domain.

In order to build with the algorithm definitions, users must build for the correct silicon revision. The following table shows which revisions of the above processor families introduced the cryptographic support:

Processor Family	Silicon Revision	Cryptographic Algorithms
ADSP-BF51x	Rev 0.0+	SHA-1
ADSP-BF522/524/526	Rev 0.0+	SHA-1
ADSP-BF523/525/527	Rev 0.1+	SHA-1
ADSP-BF54x	Rev 0.1+	SHA-1, AES, ARC4

These cryptographic algorithms have not been FIPS-certified to ensure their correctness and cryptographic security. Therefore, while the intent of providing these algorithms in ROM is to enable their use by customers, users are strongly urged to verify the adequacy of the algorithms before using them in real applications.

Changes to Reserved DDR Bits (ADSP-BF54x)

Some incomplete support from mobile DDR has been removed from the BOOT ROM header `Blackfin\include\bfrom.h`. These changes are for the ADSP-BF542, ADSP-BF544, ADSP-BF547, ADSP-BF548 and ADSP-BF549 parts:

- Definitions of macros OTP_EBIU_MOBILE_DDR_P and OTP_EBIU_MOBILE_DDR_M have been removed
- The ebiu_mobile_ddr field has been deleted from the ADI_PBS_BITFIELDS struct type

- Bit 5, the former MDDRENABLE bit, is now a reserved bit field

Feature Macros (Blackfin)

The <feature-macros> block in the `System\ArchDef\*-compiler.xml` files contain macros that the assembler and compiler automatically pre-define. General macros for processor families (e.g. `__ADSPBF54x__`) are now consistently defined for the Blackfin processor families. For more details, see below:

Macro	Processors
`__ADSPBF51x__`	ADSP-BF512, ADSP-BF514, ADSP-BF516
`__ADSPBF52x__`	ADSP-BF522, ADSP-BF523, ADSP-BF524, ADSP-BF525, ADSP-BF526, ADSP-BF527
`__ADSPBF52xLP__`	ADSP-BF522, ADSP-BF524, ADSP-BF526
`__ADSPBF53x__`	ADSP-BF531, ADSP-BF532, ADSP-BF533, ADSP-BF534, ADSP-BF536, ADSP-BF537, ADSP-BF538, ADSP-BF539
`__ADSPBF54x__`	ADSP-BF542, ADSP-BF544, ADSP-BF547,ADSP-BF548, ADSP-BF549
`__ADSPBF56x__`	ADSP-BF561

Init Code Changes (Blackfin)

The Blackfin initialization code has been modified in the following ways:

- Added a UART Baud Rate Handler
- SysControl() available in the ADSP-BF52x and ADSP-BF54x Boot ROM
- Fixed EBIU_SDBCTL_VAL in `ldr\init_code\ezkitbf5xx_initcode_asm\adsp-bf561 ez-kit lite\ezkitBF561_initcode.h` to use EB0_CAW_10 instead of EB0_CAW_9.

TAR23499 - Misallocation of Local Static Variables (Common)

Local variables declared "static const" or local uninitialized variables declared "static" were previously placed in the "data1" section by the compiler. These will now be placed in "constdata" and "bsz" respectively. These changes are for all Blackfin, SHARC and TigerSHARC parts. This might cause some applications to fail to link if the LDF does not map "constdata" or "bsz" to memory that can accommodate the increased size of these sections. In this situation the memory output section that maps "data1" (reduced in size) can also be used to map "constdata" and "bsz" which should allow the link to succeed.

Limitations

This section highlights known significant limitations

USB Driver Host (ADSP-BF52x, ADSP-BF54x)

For reliable operation of the USB controller in host mode to access USB flash drives, it is necessary to ensure that the release-build libraries are used for the System Services, Device Drivers and USB libraries. This is achieved by ensuring that the “Use Debug System libraries” option is unchecked in the “Project: Link: Processor” page of the Project Options dialog for the appropriate project. The shell_browser examples distributed with the kit are already configured such.

ADSP-BF51x Headers

The definition for the Mask Card Detect in defBF514.h / defBF516.h contains an error that will be fixed in the next update:

```
#define SD_CARD_DET_MASK 0x40 /* Mask Card Detect */
```

should be:

```
#define SD_CARD_DET_MASK 0x10 /* Mask Card Detect */
```

Power Service May Cause Unhandled CPLB Miss Exceptions (ADSP-BF54x)

Users upgrading from VisualDSP++ 5.0 Base, Update 1 or Update 2 that use a customized CPLB table and System Services may experience a problem upgrading to Update 4. In the older releases, the generated CPLB tables did not provide page descriptors for Boot ROM address space. In Update 4, the Power Service in the System Service Layer calls SysControl() in the Boot ROM. If the project has a CPLB table generated prior to VisualDSP++ 5.0 Update 3, a CPLB Miss exception will be generated. In this case, applications may experience unhandled CPLB Miss exceptions.

To avoid this problem, add the following entries to the CPLB tables.

dcplbs_table

```
{0xEF000000, (PAGE_SIZE_4KB | CPLB_L1_CHBL | CPLB_VALID | CPLB_USER_RD)}, // 4kB Boot ROM
```

Icplbs_table

```
{0xEF000000, (PAGE_SIZE_4KB | CPLB_IDOCACHE)},   // 4kB Boot ROM
```

Silicon Anomaly Workarounds

The file `System\ArchDef\BLACKFIN-EDN-anomaly.xml` has been modified to include anomaly workarounds specific to the system service and device driver libraries.

Anomaly workaround information is still available in the online help: Select Help → Contents → Graphical Environment → Silicon Anomaly Support → Silicon Anomalies Tools Support and then select the appropriate processor family.

Silicon Anomaly 09000014 (ADSP-2137x)

"Incorrect Execution of Conditional External data accesses in a delayed branch (DB) slot"

The SHARC C/C++ compiler, assembler, VDK and runtime libraries have been enhanced to include workarounds for anomaly 09000014.

The anomaly occurs when a conditional external data access instruction is in the delayed slots of a branch instruction (such as JUMP/CALL/RTS/RTI). The result is that the access can be incorrectly executed because the evaluation of the condition maybe wrong. This applies to both internal as well as external memory execution.

The compiler workaround for this anomaly avoids having conditional data accesses in delayed branch slots. To enable this compiler workaround manually the "-workaround 09000014" switch can be used. When the workaround is enabled the macro __WORKAROUND_09000014 is defined at compile, assemble and link stages.

The SHARC assembler has been modified to issue a warning (ea2521) for code that may hit the anomaly and require a workaround to be inserted. An example of this new warning is:

```
[Warning ea2521] "wa_09000014.s":13 Potential Hardware Anomaly 09000014 due to conditional
memory access by one of the two instructions following a delayed branch.
```

The assembler detection warning is enabled manually using the "-anomaly-detect 09000014" switch. The assembler defines macro __ASM_DETECT_09000014__ when detection for this anomaly is enabled.

The compiler and assembler workarounds are enabled automatically when building for ADSP-21371 and ADSP-21375 revisions 0.0 and any.

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

Silicon Anomaly 09000015 (ADSP-2137x)

"Incorrect Popping of stacks possible when exiting IRQx/Timer Interrupts with DB modifiers"

A new SHARC compiler pragma, #pragma no_db_return has been added. This pragma is used immediately before a function definition and will cause the compiler to ensure that non-delayed-branch instructions are used to return from the function. The pragma may be applied to both interrupt and non-interrupt function definitions. Applying the pragma to an interrupt function can be used as a workaround for ADSP-2137x silicon anomaly 09000015 "Incorrect Popping of stacks possible when exiting IRQx/Timer Interrupts with DB modifiers".

Silicon Anomaly 09000018 (ADSP-2137x)

"Specific Multiplier operations must not be part of the same Instruction as an External Memory access"

The SHARC C/C++ compiler, assembler, VDK and runtime libraries have been enhanced to include workarounds for anomaly 09000018.

The anomaly occurs when specific multiplier operations where multiplier results registers (MRF/MRB) are used as a destination as part of the same instruction as an external memory access. The result is that the multiplier results registers (MRF/MRB) are not correctly updated.

The compiler workaround for this anomaly avoids parallel issue of data accesses and instructions with results in MR*F or MR*B. To enable this compiler workaround manually the "-workaround 09000018" switch can be used. When the workaround is enabled the macro __WORKAROUND_09000018 is defined at compile, assemble and link stages.

The SHARC assembler has been modified to issue a warning (ea2523) for code that may hit the anomaly and require a workaround to be inserted. An example of this new warning is:

```
[Warning ea2523] "wa_09000018.s":27 Potential Hardware Anomaly 09000018 due to combining a
multiply operation into multiplier result register with a data access operation.
```

The assembler detection warning is enabled manually using the "-anomaly-detect 09000018" switch. The assembler defines macro __ASM_DETECT_09000018__ when detection for this anomaly is enabled.

The compiler and assembler workarounds are enabled automatically when building for ADSP-21371 and ADSP-21375 revisions 0.0 and any.

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

Anomaly Charts

Tools Anomalies Addressed

The following table is a list of tools anomalies addressed in VisualDSP++ 5.0 Update 4 for which details can be found on the public tools anomaly website. Other tools anomalies have also been fixed in the Update.

Details 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
All	23499	Compiler	Misallocation of Local Static Variables.
All	29851	Compiler	-section does not apply qualifiers
All	30247	Compiler	section __attribute__ does not work as documented
All	36116	Compiler	.edt and .gdt generated when not requested on some PCs
All	36188	Compiler	Prelinker loops in MISRA mode
All	36247	Compiler	Incorrect violation of MISRA Rule 10.5
All	36292	Compiler	MISRA Rule 17.4 incorrectly reported at run-time
All	35664	Run Time Libraries	The %n conversion code may assign a wrong value
All	36096	VDK	VDK::Yield() does not reset timeslice
Blackfin	36057	ADspCommon XML Files	DMAx_CONFIG Register Couples DI_SEL and DI_EN In Display Window
Blackfin	36093	ADspCommon XML Files	compiler XML enables workaround for 05-00-0283 when unnecessary
Blackfin	36266	ADspCommon XML Files	workaround for 05-00-00371 required for ADSP-BF54[24789] rev 0.0, 0.1
Blackfin	36341	Compiler	MISRA Rule 21.1. run-time checking of global arrays not enabled
Blackfin	36424	Compiler	typedef'd bit fields can be incorrectly packed with #pragma pack
Blackfin	36453	Compiler	MISRA Rule 19.4 not always suppressed using #pragma diag
Blackfin	30369	Debug Agent	Debug agent scans too fast [can cause external memory issues]
Blackfin	36186	Device Driver	ADSP-BF527 LCD Example does not work with deferred callbacks
Blackfin	34944	Run Time Libraries	clock() is not thread-safe
Blackfin	35495	Run Time Libraries	05-00-0248 workaround in adi_acquire_lock() and adi_try_lock()
Blackfin	36110	Run Time Libraries	default multi-thread CRT objects may result in CPLB misses
Blackfin	36125	Run Time Libraries	interrupt macros fail with MISRA Rule 16.5(Req) errors
Blackfin	36304	Run Time Libraries	Edge case INT_MIN/INT_MAX returns incorrect result
Blackfin	36451	Run Time Libraries	cdefBF561.h pX macro uses result in MISRA Rule 19.4 errors

Blackfin	35571	Simulator	Incorrect display of data in HEX8 format in Blackfin Memory Window
Blackfin	36070	Simulator	binary 16-bit memory window format only displays 8 bits
Blackfin	29902	System Builder	Project fails to build when user sets heap space to less than 1k
Blackfin	34546	System Builder	Incorrect path for basiccrt in exported makefile
Blackfin	35610	System Builder	Handling of stack and heap sizes in Bytes should be reviewed.
Blackfin	36158	System Builder	Heap/Stack configured with KB will not select kB in Proj Options
Blackfin	36223	System Builder	CACHE_MEM_MODE is undefined
Blackfin	35045	System Services	error during fss_rename function
Blackfin	29736	TCPIP Stack	Multiple network interface issue
Blackfin	30157	TCPIP Stack	lwip send function returns bytes sent, but sends only 64K max
SHARC	36505	Assembler	asm warns about the part of anomaly 07000009 fixed in rev 0.5
SHARC	35661	Emulator	Unable to set core child register bits via register window
SHARC	36421	Emulator	0xCDCDCDCD in the SDRAM ranges higher than 0x80000 for 21065L
SHARC	36444	Loader	The kernel dxe does not correlate with the asm file in project
SHARC	36075	Run Time Libraries	PM version of setlocale(___setlocaleP) has code in wrong section
SHARC	36371	Run Time Libraries	powd() returns wrong result for powers >1e8 and <-1e8

Known Tools Anomalies

Details can be found on the Tools Anomaly Web page. The URL is:

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

VisualDSP++® 5.0 Update 3 Release Notes

Revision 1.3

June 16, 2008

Table of Contents

Nomenclature

In the past, VisualDSP++ updates were labeled by the month and year of their release. In order to improve clarity, updates are now numbered (e.g. Update 1, Update 2, etc).

Installation

This update should only be installed after installing the VisualDSP++ 5.0 base release. If VisualDSP++ 5.0 is not installed, please install it first. Installation on a previous update is fine. If a newer update has already been installed, please do not install this update. This update is not intended to be installed on alpha or beta releases.

Identifying Your VisualDSP++ Version

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

1. In the Control Panel, open the Add/Remove Programs applet.
2. In the VisualDSP++ Development Environment, select Help – About VisualDSP++.

Installing the Update

Please follow the instructions below for installing this update. Please note that since VisualDSP++ supports having multiple instances installed on a single system, you can install this update on top of one instance while keeping the previous installation.

1. Use the Start Menu to navigate to VisualDSP++ “Maintain this installation”. By default this is at Start Menu - select All Programs - Analog Devices - VisualDSP++ 5.0.
2. Select “Go to the Analog Devices website” and click Next. This will open a window in your web browser.
3. Select the appropriate Processor Software Tools Upgrades to match your processor.
4. Select and download the desired update (VisualDSP++ 5.0_Update3.vdu) to your hard drive.
5. Again, use the Start Menu to navigate to VisualDSP++ “Maintain this installation”.
6. Select “Apply a downloaded Update” and click Next.
7. Browse for the downloaded Update file (VisualDSP++ 5.0_Update3.vdu) and click Next.
8. Follow the on-screen prompts to complete installation of this Update.

Cloning VisualDSP++

VisualDSP++ supports cloning of an existing installation. A clone of an installation creates a new instance of a product from an existing installation, rather than from a CD or web software distribution. The use of clones allows you to maintain multiple versions of VisualDSP++ on the same PC at different update levels, and provides a risk-free way to "test" new updates or patches.

To clone your existing installation of VisualDSP++:

13. Go to Start->Programs->Analog Devices->VisualDSP++ 5.0 (or equivalent)->Maintain this Installation
14. Select "Clone this Installation" and click Next.
15. Optionally click Advanced to set the Start menu path.
16. Enter the Clone install path and click Next.

Definitions

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

TAR – Tools Anomaly Reference Number

Tools Anomaly Reference Number, or TAR, is used for tracking confirmed defect reports in VisualDSP++.

New Hardware Support

VisualDSP++ updates often include support for new processors, new silicon revisions for existing processors and new EZ-KIT Lite® 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.

New Processors and Revisions Support

The Product Bulletin contains the list of new processors available with VisualDSP++ 5.0. Refer to the processor's data sheet and hardware reference manuals for information on system configuration, peripherals, registers, and operating modes.

Update 3 provides support for the following silicon revisions to existing Blackfin® processors:

- ADSP-BF523 silicon revision 0.2
- ADSP-BF525 silicon revision 0.2
- ADSP-BF527 silicon revision 0.2

- 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 3.

New System Services and Device Drivers

The following are now supported by VisualDSP++ 5.0:

- Support for on-chip peripherals for the ADSP-BF522, BF524 and BF526 processors

New Examples

ADSP-BF527 EZ-KIT Lite Audio Loopback Examples

VisualDSP++ 5.0 Update 3 includes a new audio loopback example to demonstrate use of the audio codec supplied on the ADSP-BF527 EZ-KIT Lite® evaluation system. It can be found in the following directory:

```
Blackfin\Examples\ADSP-BF527 EZ-KIT Lite\drivers\AudioCodec\Audio_Loopback
```

ADSP-BF527 and ADSP-BF548 EZ-KIT Lite Autobaud Examples

VisualDSP++ 5.0 Update 3 includes new examples to demonstrate use of the UART device driver in Autobaud mode supplied on the ADSP-BF527 and ADSP-BF548 EZ-KIT Lite® evaluation systems. They can be found in the following directory:

```
Blackfin\Examples\ADSP-BF527 EZ-KIT Lite\drivers\UART\Autobaud
Blackfin\Examples\ADSP-BF548 EZ-KIT Lite\drivers\UART\Autobaud
```

Critical Fixes/Changes

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

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

The workaround for silicon anomaly 09000011 may generate 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 Description 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.

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.

SHARC Workaround Informational

A new informational message is generated for PC relative jump instructions:

```
ea1130 - Constant offset in JUMP is not recommended. JUMP to a label instead. The
assembler has done this for you.
```

The assembler automatically changes PC relative jumps in order to prevent problems with the insertion of anomaly workaround code. You can turn off this message by adding the following to assembly files which generate the warning:

```
.message/suppress 1130;
```

New Type Header Files for all Processors

The following header files have been added for all processors:

stdint.h

ANSI C99 standard conformant header file that defines various integer typedefs.

stdbool.h

ANSI C99 standard conformant header file that defines various boolean related macros.

services_types.h

Header file that defines various integer typedefs for use in system services code, some boot kernels and examples.

adi_types.h

Includes stdint.h, stdbool.h and defines other float and char typedefs. For use in generic code that requires a complete set of typedefs such as MISRA conformant applications.

Boot Code Sources Available for ADSP-BF52x and ADSP-BF54x

The Boot Code Sources are now available for the ADSP-BF52x and ADSP-BF54x and can be found in “Blackfin/ldr/Boot ROM”.

Changes to Blackfin ldr Source Tree

Prior to VisualDSP++ 5.0 Update 3, there was a single set of Blackfin Boot Kernel sources. With the introduction of the ADSP-BF52x and ADSPBF54x processors, a next generation Boot Kernel was written. VisualDSP++ 5.0 Update 3 contains the sources for both. Two new folders were created within Boot ROM\src:

bk_ad00 contains the "legacy" Boot Kernel sources for the ADSP-BF53x and ADSP-BF561.
bk_ad03 contains the "latest generation" Boot Kernel sources, versions 03 and 02

New Blackfin ROM Header API

To better support the on-chip Boot ROM and L1 ROM, the ROM Header API is now defined in bfrom.h.

Location: <install-dir>\Blackfin\include\bfrom.h

C-Versions of Initialization Code

Update 3 includes initialization code examples for the ADSP-BF52x and ADSP-BF54x processor written in C language. Unlike former Blackfin derivatives, such as ADSP-BF53x and ADSP-BF561 devices, the new ADSP-BF52x and ADSP-BF54x processors’ initialization code concept is compliant to C-language calling conventions. Therefore, the user has the choice to implement initialization codes in C or assembly language. The examples can be found at:

Blackfin\ldr\init_code\asm	The assembly are the legacy versions
Blackfin\ldr\init_code\c	New C versions

These examples are developed and tested for respective EZ-KIT Lite boards. Customized hardware makes need for modifications likely.

For dynamic power management the initcode examples make use of the bfrom_SysControl() function which is part of the ROM API featured by new processors.

Changes to Support MISRA Technical Corrigendum 1

The document for Technical Corrigendum 1 (TC1) was produced in July, 2007 to clarify and address issues with MISRA-C:2004. The TC1 document describes changes to the original MISRA-C:2004 document. The TC1 document can be downloaded from the MISRA site (http://misra.org.uk). The site requires you to register in order to download the document. Most changes are simple clarifications, although some changes affect the rule violations. What follows is a brief description of the rules, where the rule violations reported will significantly change.

Rule 4.1
The normative text now states that "All hexadecimal-escape-sequences are prohibited".

Rule 10.3
Headline rule changed to "The value of a complex expression of integer type shall only be cast to a type of the same signedness that is no wider than the underlying type of the expression."

This has the effect that casting the type of a complex expression to the same type as the complex expression will not report a rule violation.

Rule 10.4
Headline rule changed to "The value of a complex expression of floating type shall only be cast to a floating type which is narrower or of the same size."

This has the effect that casting the type of a complex expression to the same type as the complex expression will not report a rule violation.

Rule 10.5
The headline rule has not changed but the normative text has changed in respect to a cast.

- A cast is no longer required in all circumstances.

- Bitwise operations do not require a cast if:
  - immediately assigned to an object of the same underlying type
  - used as a function argument of the same underlying type as the operand

- used as a return expression of a function whose return type is the same underlying type as the operand.

Rule 12.6
Additional operators '=, ==, != and ?:' added to the list of operators.

Rule 19.4
C macros can also expand to a string literal.

Ability to Suppress all MISRA Rules Checking

In some code it is necessary to suppress all MISRA checks. New support for #pragma diag has been added to make that easy to do to. For example:

```
#pragma diag(suppress:misra_rules_all:"Misra rules all suppressed because ... ")
```

TAR 35448 - MCMEN Defined in ADSP-BF54x Definition Header Files

A macro for the SPORTx_MCMC2 Multi channel Frame Mode Enable bit has been added to defBF54x_base.h. This file is included by all the platform include files (defBF549.h etc). The new macro is called MCMEN. Any code that defines or uses a macro with the same name will need to be modified to avoid a conflict with the new definition.

The prior definition of a macro for this bit, MCMEM, is deprecated. It should not be used and will not be supported in future releases.

TAR 34699 - EBIU_AMGCTL Bit Macros in ADSP-BF54x Definition Header Files

Two new macros for EBIU_AMGCTL to enable all Async memory banks have been added to defBF54x_base.h.

The new macros are defined as follows:

```
#define AMBEN_B0_B1_B2_B3 0x0008 /* Enable Async Memory Banks 0, 1, 2 and 3 */
#define AMBEN_ALL 0x0008         /* Enable All Async Memory Banks */
```

Any code that defines or uses a macro with the same name as these will need to be modified to avoid a conflict with the new definitions.

TAR 34700 - HMDMAx_CONTROL Bit Macros in ADSP-BF54x Definition Header Files

Two new macros for HMDMAx_CONTROL bit for source not destination have been added to defBF54x_base.h.

The new macros are defined as follows:

```
#define                         SND  0x80       /* Source/Not Destination */
#define                        nSND  0x0
```

Any code that defines or uses a macro with the same name as these will need to be modified to avoid a conflict with the new definitions.

TAR 35154 - SIC_RVECT Removed from Definition Header Files

The definition of a macro SIC_RVECT has been removed from the various def header files. This macro was incorrectly defined and should not be used.

TAR 35481 - Fixed Signed CHAR definition

The definition of integer type s8 in services.h has been fixed. This could cause backwards compatibility issues if the user relied on it being unsigned.

The following definition:

```
typedef char s8;
```

has been modified to the following:

```
typedef signed char s8;
```

TAR 33557 – Blackfin 64-bit double modf Result Changed

The implementation of the Blackfin 64-bit double precision modf standard C function has been modified to return 0.0 rather than a NaN (Not-A-Number) when the second operand to modf is 0.0. The result in this situation is implementation-defined according to the ANSI C standard. The documentation for the function has always stated that it should return 0.0 for this input.

Limitations

This section highlights known significant limitations

VisualDSP++ 5.0 ADSP-BF54x Known Limitations

The following device drivers will be supported in a future update:

- NAND FLASH driver

VisualDSP++ 5.0 ADSP-BF52x Known Limitations

The following device drivers will be supported in a future update:

- NAND FLASH driver

Workarounds for Silicon Anomaly 05-00-00371 ADSP-BF54[24789] Rev 0.1

The compiler and assembler workarounds for anomaly 05-00-0371 "Possible RETS Register Corruption when Subroutine Is under 5 Cycles in Duration" are not automatically enabled when building for ADSP-BF54[24789] revision 0.1.

To avoid this issue change the project target to build for revision 0.0 rather than 0.1. Alternatively, build C source for ADSP-BF54[24789] revision 0.1 with switches:

```
-workaround avoid-quick-rts-371
```

and build assembly source with switches:

```
-anomaly-workaround 05000371 -anomaly-detect 05000371
```

TAR 35159 - 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.

TAR 35556 – System Services Caps CCLK for DDR reliability

Description: The ADSP-BF54X is the first Blackfin to use double data rate SDRAM (DDR). The first revision of the silicon was tested to 533 MHz @ VDD_INT = (1.25v - 5%). Some applications, with data buffers located in external L3 memory (DDR), had occasional, intermittent data corruption problems at certain combinations of core clock frequency (CCLK) and voltage level (VLEV). The temporary workaround has been to reduce the core clock frequency (CCLK) to 400 MHz, with VLEV set at 1.2 V. The System Services Power Management module caps CCLK and VLEV, as a precautionary measure.

The second revision of ADSP-BF54X silicon is still under evaluation, and the recommendations will soon be available for the CCLK vs. VLEV relationship. In the absence of exact characterization data for the silicon, the System Services Power Management module continues to limit CCLK and VLEV for reliable DDR performance.

If the application is not affected by the intermittent data corruption problem, higher core clock frequencies may be attained using the System Services Power Management command "ADI_PWR_CMD_SET_CCLK_TABLE", described in the Power Management API reference section of the Device Drivers and System Services User Manual in VisualDSP Help. This command overwrites the hard-coded CCLK vs. VLEV values, located in the Power Management source file:

```
        Blackfin\lib\src\services\pwr\adi_pwr.c
```

The command "ADI_PWR_CMD_SET_CCLK_TABLE" is passed to the Power Management initialization function, 'adi_pwr_init', to define one such CCLK vs. VLEV relationship. As a general guideline for defining the CCLK vs. VLEV relationship, see table 13 (Core Clock Requirements - 500 MHz, 533 MHz, and 600 MHz Models) in Revision E of the ADSP-BF531/2/3 datasheet, as a general guideline, but note that the ADSP-BF54X is NOT guaranteed for these same values.

Define an array of ADI_PWR_NUM_VLEVS elements (defined in the API header file 'adi_pwr.h') of type unsigned 32 bit integer (u32) which specifies the maximum core clock frequency for the associated voltage level, as shown below.

```
static u32 pwr_cclk_vlev_table [ADI_PWR_NUM_VLEVS] =
{
    /* ADI_PWR_VLEV_085 */   250,
    /* ADI_PWR_VLEV_090 */   334,
    /* ADI_PWR_VLEV_095 */   334,
    /* ADI_PWR_VLEV_100 */   400,
    /* ADI_PWR_VLEV_105 */   400,
    /* ADI_PWR_VLEV_110 */   444,
    /* ADI_PWR_VLEV_115 */   444,
    /* ADI_PWR_VLEV_120 */   500,
    /* ADI_PWR_VLEV_125 */   533,
    /* ADI_PWR_VLEV_130 */   533
};
```

Create the command pair table to include the command ADI_PWR_CMD_SET_CCLK_TABLE, followed by a pointer to the array, as shown below.

```
ADI_PWR_COMMAND_PAIR PowerInitTable[] = {
{
    ADI_PWR_CMD_SET_PROC_VARIANT,  (void*)ADI_PWR_PROC_BF549SBBC1533 },
    { ADI_PWR_CMD_SET_PACKAGE,          (void*)ADI_PWR_PACKAGE_MBGA },
    { ADI_PWR_CMD_SET_VDDEXT,             (void*)ADI_PWR_VDDEXT_330 },
    { ADI_PWR_CMD_SET_CLKIN,                 (void*)25 },
    { ADI_PWR_CMD_SET_CCLK_TABLE,    (void *) pwr_cclk_vlev_table },
    { ADI_PWR_CMD_END, 0 }
};
```

Pass the command pair table to 'adi_pwr_Init' as shown below.

```
Result = adi_pwr_Init( PowerInitTable );
```

The Power Management Service will then use the CCLK vs. VLEV relationship defined by the array "pwr_cclk_vlev_table", instead of the columns of the array defined in the source file 'adi_pwr.c'.

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 customized LDFs may require modification to map this section.

The assembler will provide informational messages for each instance of an applied workaround 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 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.
```

This workaround may generate 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*

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.

Anomaly Charts

Tools Anomalies Addressed

The following table is a list of tools anomalies addressed in VisualDSP++ 5.0 Update 3 for which details can be found on the public tools anomaly website. Other tools anomalies have also been fixed in the Update.

Details 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
All	34023	Compiler	Extern "C" after default_section(ALLDATA,"L1_data") gives error
All	34828	Compiler	default_section pragma doesn't work with concatenated strings
All	34928	Compiler	sizeof multi-dimensional array of variable & static length array
All	35158	IDDE	VDK Status window shows incorrect message channel
Blackfin	35512	ADspCommon XML Files	anomalies 05-00-0312 and 05-00-0283 don't apply to BF52x
Blackfin	35047	Compiler	MISRA rules 10.1.b & 10.2.b incorrectly reported
Blackfin	35122	Compiler	int div/mod causes internal compiler error (peephole.c:1472) -O
Blackfin	35102	Examples	Missing boot ROM sources for ADSP-BF52x / ADSP-BF54x processors
Blackfin	35488	Examples	ADSP-BF561 internal/external regulator readme note incorrect
Blackfin	35178	Flash Programmer	Compare does not work for BF527EzFlashDriver_M25P16
Blackfin	34397	IDDE	Trying to run VisualDSP++ 5.0 but nothing happens.
Blackfin	35039	IDDE	License Server's License Manager reads both license.dat files
Blackfin	29526	Loader	Request to support -p # exclusively for application HEX address.
Blackfin	28489	Run Time Libraries	64-bit fast float mult inaccurate when result close to denorm
Blackfin	32319	Run Time Libraries	crtn.doj can be removed from .LDF File without warning
Blackfin	33761	Run Time Libraries	Incorrect figures from instrumented profiling using compiled sim

Blackfin	34699	Run Time Libraries	AMBEN_ALL missing in defBF54x_base.h
Blackfin	35448	Run Time Libraries	Error in defBF54x_base.h which defines MCMEM, instead of MCMEN
Blackfin	34370	Simulator	memory dma fails for 54x meminit on compiled simulation
Blackfin	33007	TCPIP Stack	INETD example should not set the user_data_ptr in the header
Blackfin	35276	TCPIP Stack	ADSP-BF527 LAN examples fail because of MAC address in reverse order
Blackfin	34703	TCPIP Wizard	Unable to modify MAC address for Network0 in LwIP Project.
Blackfin	35060	VDK	VDK does not reset the contents in memory for mempools
Blackfin	35254	VDK	Context switch code may get split between memory regions
SHARC	35205	Compiler	#pragma interrupt_complete_nesting causes unsafe code
SHARC	35245	Compiler	internal error at bitmatrix.c:81, -restrict-hardware-loops 1
SHARC	35390	Compiler	memory access from 0 pre-modified with address –O
SHARC	35340	Run Time Libraries	Missing ")" in 212xx/include/def21266.h
TigerSHARC	34924	Compiler	terminate not called when exception thrown during handler (TS)
TigerSHARC	35194	VDK	VDK Thread stack incorrectly configured during thread creation

Known Tools Anomalies

Details can be found on the Tools Anomaly Web page. The URL is:

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

VisualDSP++® 5.0 Update 2 Release Notes

Revision 1.1

2008 February 26

Table of Contents

Nomenclature

In the past, VisualDSP++ updates were labeled by the month and year of their release. In order to improve clarity, updates will now be numbered (e.g. Update 1, Update 2, etc).

Installation

This update should only be installed after installing the VisualDSP++ 5.0 base release. If VisualDSP++ 5.0 is not installed, please install it first. Installation on a previous update is fine. If a newer update has already been installed, please do not install this update. This update is not intended to be installed on alpha or beta releases.

Identifying Your VisualDSP++ Version

The VisualDSP++ release and update level can be found in 2 locations:
In the Control Panel, open the Add/Remove Programs applet.
In the VisualDSP++ Development Environment, select Help – About VisualDSP++.

Installing the Update

Please follow the instructions below for installing this update. Please note that since VisualDSP++ supports having multiple instances installed on a single system, you can install this update on top of one instance while keeping the previous installation.
Use the Start Menu to navigate to VisualDSP++ “Maintain this installation”. By default this is at Start Menu - select All Programs - Analog Devices - VisualDSP++ 5.0.
Select “Go to the Analog Devices website” and click Next. This will open a window in your web browser.
Select the appropriate Processor Software Tools Upgrades to match your processor.
Select and download the desired update (VisualDSP++ 5.0_Update2.vdu) to your hard drive.
Again, use the Start Menu to navigate to VisualDSP++ “Maintain this installation”.
Select “Apply a downloaded Update” and click Next.
Browse for the downloaded Update file (VisualDSP++ 5.0_Update2.vdu) and click Next.
Follow the on-screen prompts to complete installation of this Update.

New Hardware Support

VisualDSP++ updates often include support for new processors, new silicon revisions for existing processors and new EZ-KIT Lite® 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.

New Processors and Revisions Support

The Product Bulletin contains the list of new processors available with VisualDSP++ 5.0. Refer to the processor's data sheet and hardware reference manuals for information on system configuration, peripherals, registers, and operating modes. The following are Blackfin® processors newly supported with Update 2:

- ADSP-BF522* silicon revision 0.0
- ADSP-BF524 silicon revision 0.0
- ADSP-BF526 silicon revision 0.0

*Please note that the ADSP-BF522 processor supported in VisualDSP++ 5.0 has been renamed as the ADSP-BF523.

The following are newly supported silicon revisions to existing Blackfin® processors with Update 2:

- ADSP-BF523 silicon revision 0.1
- ADSP-BF525 silicon revision 0.1
- ADSP-BF527 silicon revision 0.1

There are no new silicon revisions to existing SHARC® or TigerSHARC® processors with Update 2.

New Emulation Support

The following emulation features are now supported by VisualDSP++ 5.0:
Support for the ADSP-BF522*, BF524 and BF526 processors

*Please note that the ADSP-BF522 processor supported in VisualDSP++ 5.0 has been renamed as the ADSP-BF523.

New System Services and Device Drivers

The following are now supported by VisualDSP++ 5.0:
Initial System Services Library support for the ADSP-BF522*, BF524 and BF526 processors

*Please note that the ADSP-BF522 processor supported in VisualDSP++ 5.0 has been renamed as the ADSP-BF523.

New Examples

LCD

VisualDSP++ 5.0 Update 2 includes a new LCD example to demonstrate use of the LCD supplied on the ADSP-BF527 EZ-KIT Lite® evaluation system. It can be found in the following directory:

Blackfin\Examples\ADSP-BF527 EZ-KIT Lite\drivers\LCD

File System

VisualDSP++ 5.0 Update 2 now includes a File System example for the ADSP-BF527 EZ-KIT Lite® evaluation system similar to the ADSP-BF548 EZ-KIT Lite® evaluation system example. It can be found in the following directory:

Blackfin\Examples\ADSP-BF527 EZ-KIT Lite\services\File System

Critical Fixes/Changes

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

ADSP-BF522 processor name change

The ADSP-BF522 has been renamed as the ADSP-BF523. Support for this new name is available in Update 2 to VisualDSP++ 5.0. Those who already created projects for the BF522 and did not use automatically generated LDF's for the ADSP-BF522 may need to rewrite or modify their LDF's in the future. There is a new ADSP-BF522 processor. Please refer to the datasheet online for clarification: http://www.analog.com/en/epProd/0,,ADSP-BF527,00.html

Linker error li1040 and .meminit in LDFs – TAR 34071

The linker has a modification to resolve issues with meminit support (TAR34071) that can expose errors in existing LDFs. The linker issues error li1040 for these problems. This is an example of the linker output:

```
[Error li1040] "C:\Program Files\Analog Devices\VisualDSP5.0\TS\ldf\ADSP-
TS101.ldf":204 Out of memory in output section '.meminit' in processor 'p0'.
Total of 0x1 word(s) were not mapped.
```

The default ADSP-TS101 LDFs had the problem and has been fixed in Update 2 (TAR34273).
Older Blackfin default LDFs also had the problem so user customized LDF based on these older versions of the files may also encounter the error (for example see TAR35101).
The fix for the problem is to remove the .meminit command from the LDF file. This can be done by either deleting it or by guarding it with the __MEMINIT__ pre-processor macro (defined by the linker when meminit support is actually required). For example:

```
   #if defined(__MEMINIT__)
      .meminit { ALIGN(4) } >MEM_L1_DATA_A
   #endif
```

Limitations

This section highlights known significant limitations

VisualDSP++ 5.0 ADSP-BF54x Known Limitations

The following device drivers will be supported in a future update:

- NAND FLASH driver

VisualDSP++ 5.0 ADSP-BF52x Known Limitations

The following device drivers will be supported in a future update:

- NAND FLASH driver

Incomplete ADSP-BF523/BF525/BF527 silicon rev. 0.1 Support – TAR 35224

The VisualDSP++ 5.0 Project Target selections will not allow the option to build for the new 0.1 silicon revisions of ADSP-BF523, ADSP-BF525 or ADSP-BF527. Only "Automatic", "none", "0.0" or "any" can be selected. If "Automatic" is selected and you are connected through an emulator to a revision 0.1 part, you will get multiple cc3146 and ea1142 warnings when building your project. To avoid these warnings change the Project Target revision selection to "any".

Full support for these new revisions will be provided in Update 3 of VisualDSP++ 5.0.

No ADSP-BF523/BF524 Startup Wizard Support – TAR 35164

When creating a new project for either the ADSP-BF523 or ADSP-BF524 the startup code page in the project wizard does not appear.

To avoid this problem, create a project for the ADSP-BF527 instead of ADSP-BF523 or a project for the ADSP-BF526 instead of the ADSP-BF524. Full processor support will be available in Update 3.

Silicon Anomaly Workarounds

ADSP-BF5xx Silicon Anomaly 05-00-0323

"Erroneous GPIO Flag Pin Operations under Specific Sequences" anomaly workarounds support has been added.

Include file sys/05000323.h is now supplied with VisualDSP++ 5.0. 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++ 5.0 Install>\Blackfin\include\sys\05000323.h) for further details.

ADSP-BF5xx Silicon Anomaly 05-00-0371

"Possible RETS Register Corruption when Subroutine is under 5 Cycles in Duration" anomaly workarounds support has been added.

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 Blackfin assembler has been modified to issue a warning (ea5516) for code that may hit the anomaly and require a workaround to be inserted. An example of this new warning is:

```
 [Warning ea5516] "memchr.asm":39 RTS instruction use may trigger hardware
anomaly 05-00-0371. See appropriate Blackfin anomaly lists for more
information.
```

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

ADSP-BF52x Silicon Anomaly 05-00-0380

“Data Read from L3 Memory by USB DMA May be Corrupted”

To workaround this anomaly, the USB Physical Interface Driver employs an intermediate buffer in L1 memory. The larger this buffer, the better the performance. However, the driver that is released with VisualDSP++ 5.0 employs a medium sized L1 buffer of size 8KB, providing 10MB/s read throughput and 0.8MB/s write throughput. These figures represent a 30% decrease in performance compared to the driver implemented for the ADSP-BF548 processor.

Invalid SCLK Frequency for ADSP-BF548 at Power Up – TAR 35129

At power-up, SCLK frequency on ADSP-BF548 EZ-KIT Lite must be set to within 83MHz-133MHz if the stack or heap is located in DDR, or else 'adi_pwr_SetFreq' may fail.

The ADSP-BF548 EZ-KIT Lite is populated with double data rate SDRAM (DDR). There are two types of DDR: mobile and non-mobile. The EZ kit uses non-mobile DDR. The nominal system clock (SCLK) frequency range for non-mobile DDR is 83 MHz to 133 MHz.

The input clock (CLKIN) on the ADSP-BF548 is 25MHz. At reset, the multiplier select (MSEL) value in the PLL control register (PLL_CTL) is decimal 10, while the PLL divider ratio register (PLL_DIV) contains 5. Together these values produce a SCLK value of 50MHz (25 * 10 / 5), which is below the minimum for non-mobile DDR to work properly. Therefore, DDR should not be accessed until after the PLL registers have been set up to produce a system clock frequency in the range of 83 MHz to 133 MHz.

If the stack is located in DDR, then DDR will be accessed as soon as the application starts running, so the PLL must be set up prior to loading and executing the application.

The EZ-KIT Lite is delivered with an application in flash, which sets SCLK to 133MHz. If flash is erased and a new application is programmed into flash, and the new application uses DDR for stack, then prior to loading and executing the new application, the boot kernel should set system clock frequency (SCLK) to a valid frequency, using “Pre-boot” or “Init Code".

If SCLK is out of the valid range while DDR stack activity is taking place, the power management function ‘adi_pwr_SetFreq’ will fail. This function takes DDR into self refresh mode, to protect external memory while the system clock is adjusted. This is problematic with SCLK at 50 MHz, and stack/heap located in DDR. The function will “hang” under those conditions. If stack/heap is not located in DDR, and no other DDR access is taking place, then the ‘adi_pwr_SetFreq’ function will succeed in changing the clock frequencies so that subsequently, DDR can be used without problems.

Problem Charts

Problems Addressed

The following table is a list of problems addressed in VisualDSP++ 5.0 Update 2 for which details can be found on the public tools anomaly website. Other problems have also been fixed in the Update.

Details can be found on the Tools Anomaly Web page. The URL is:
http://www.analog.com/processors/tools/anomalies

Product Family	Problem Number	Tool	Description
All	33743	IDDE	Dumping empty 21160 core file fails
All	33758	IDDE	trouble opening file in IDDE after adding it to a project
All	34558	Run Time Libraries	snprintf and vsnprintf may write 1 too many chars to the output
All	34720	VDK	Scheduling is disabled after call to DestroyMutex
Blackfin	34809	ADspCommon XML Files	EBSZ Field in EBIU_SDBCTL is 3 bits in BF534/6/7/8/9 Processors
Blackfin	33643	Compiler	keywords such as section cause spurious errors in MISRA mode
Blackfin	32752	Debug Agent	IceTest fails on RoHS EZ-KITs using USB 2.0 HUB
Blackfin	34628	Device Driver	NEC LCD driver broken by a PPI driver change
Blackfin	34668	Emulator	Watchdog timer does not fully reset when reset through emulator
Blackfin	33862	Examples	CDemo Buffer description are incorrect
Blackfin	33942	Examples	BF561 Chained DMA example does not work.
Blackfin	34444	Examples	BF537_SAFP.js does not run to completion
Blackfin	34597	Examples	Problems with BF533 EZ-kit Example 'Video-In'
Blackfin	34368	Flash Programmer	When erasing sector 1 on the BF548 it also erases sector 0
Blackfin	33680	IDDE	Changing project options may overwrite working LDF
Blackfin	34621	IDDE	si-revision any in project options does not work
Blackfin	34259	LDFGen	Start symbol of second user heap in SDRAM has wrong value
Blackfin	34478	Loader	Loader Driver creates incomplete dependency
Blackfin	34317	Run Time Libraries	L2_shared memory, used to map locks etc, can be cached
Blackfin	34487	Run Time Libraries	SYSCR bits not yet updated in defBF52x_base.h
Blackfin	34488	Run Time Libraries	SYSCR bits not yet updated in defBF54x_base.h
Blackfin	34744	Run Time Libraries	meminit support zero init of arrays larger than 64k fails
Blackfin	34291	Simulator	MDMA needs to be supported in BF54x for meminit to work
Blackfin	34319	Simulator	Filling memory with Hex32 format file reads the wrong way
Blackfin	34434	Simulator	"Binary 16 Bit" does not work in memory window.
Blackfin	34742	Simulator	DMA MMRs incorrect in memory/locals/expr windows

Blackfin	33518	System Services	pwr mgmt to facilitate transition from SLEEP
Blackfin	29313	TCPIP Stack	ETHARP_ALWAYS_INSERT option is deprecated in lwIP
Blackfin	34680	VDK	VDK Status window does not display any threads
SHARC	32749	Compiler	slowdown of code using division when build -Os
SHARC	34819	Compiler	USTAT1 and USTAT2 used in compiler generated code
SHARC	29561	Emulator	VisualDSP+ +disconnects if Sport DMA Address reg window is open
SHARC	32810	Emulator	Incorrect display of instructions in external memory on Sharc
SHARC	35013	Emulator	Cannot load 16-bit external memory on 2126x
SHARC	34792	Flash Programmer	ADSP-21375 SPI flash will change from the Atmel to the STMicro
SHARC	33670	Run Time Libraries	SIG_MTM to be defined for 21362/3/4/5/6
SHARC	34727	Run Time Libraries	sinf may return poor results for inputs close to a 2*PI multiple

Known Problems

Details can be found on the Tools Anomaly Web page. The URL is:

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

VisualDSP++® 5.0 Update 1 Release Notes

Revision 1.1

2008 February 26

Table of Contents

Nomenclature

In the past, VisualDSP++ updates were labeled by the month and year of their release. In order to improve clarity, updates will now be numbered (e.g. Update 1, Update 2, etc).

Installation

This update should only be installed after installing the VisualDSP++ 5.0 base release. If VisualDSP++ 5.0 is not installed, please install it first. If a newer update has already been installed, please do not install this update. This update is not intended to be installed on alpha or beta releases.

Identifying Your VisualDSP++ Version

The VisualDSP++ release and update level can be found in 2 locations:
In the Control Panel, open the Add/Remove Programs applet.
In the VisualDSP++ Development Environment, select Help – About VisualDSP++.
In these locations, VisualDSP++ 5.0 should be visible without any update listed.

Installing the Update

Please follow the instructions below for installing this update. Please note that since VisualDSP++ supports having multiple instances installed on a single system, you can install this update on top of one instance while keeping the previous installation.
Use the Start Menu to navigate to VisualDSP++ "Maintain this installation". By default this is at Start Menu - select All Programs - Analog Devices - VisualDSP++ 5.0.
Select "Go to the Analog Devices website" and click Next. This will open a window in your web browser.
Select the appropriate Processor Software Tools Upgrades to match your processor.
Select and download the desired update (VisualDSP++ 5.0_Update2.vdu) to your hard drive.
Again, use the Start Menu to navigate to VisualDSP++ "Maintain this installation".
Select "Apply a downloaded Update" and click Next.
Browse for the downloaded Update file (VisualDSP++ 5.0_Update2.vdu) and click Next.
Follow the on-screen prompts to complete installation of this Update.

New Hardware Support

VisualDSP++ updates often include support for new processors, new silicon revisions for existing processors and new EZ-KIT Lite® 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.

New Processors and Revisions Support

The Product Release Bulletin contains the list of new processors available with VisualDSP++ 5.0. Refer to the processor's data sheet and hardware reference manuals for information on system configuration, peripherals, registers, and operating modes. The following are Blackfin® processors newly supported with Update 1:

- ADSP-BF547 silicon revision 0.1

The following are newly supported silicon revisions to existing Blackfin® processors with Update 1:

- ADSP-BF542 silicon revision 0.1
- ADSP-BF544 silicon revision 0.1
- ADSP-BF548 silicon revision 0.1
- ADSP-BF549 silicon revision 0.1

The following are newly supported silicon revisions to existing SHARC® processors with Update 1:

- ADSP-21367 silicon revision 0.2
- ADSP-21368 silicon revision 0.2
- ADSP-21369 silicon revision 0.2

New Emulation Support

The following emulation features are now supported by VisualDSP++ 5.0:
Support for the ADSP-BF52x processors
Support for the BF527 EZ-KIT Lite and onboard debug agent
Flash programming for the BF527 EZ-KIT Lite for STMicroelectronics M25P16 and STMicroelectronics M29W320

New System Services and Device Drivers

The following are now supported by VisualDSP++ 5.0:
Full System Services Library support for the ADSP-BF522, BF525 and BF527 processors
Device Drivers for the ADSP-BF54x processors

- USB Mass Storage OTG Host

Device Drivers and Middleware for the BF548 EZ-KIT Lite

- AD1980 AC-97 Codec Driver for the BF548 EZ-KIT Lite
- Added SD Write Capability

Device Drivers and Middleware for the ADSP-BF52x processors

- PPI
- SPI
- SPORT
- UART
- TWI
- Rotary Counter
- Integrated Stereo Audio Codec
- Background Telemetry
- USB Mass Storage Device
- USB Mass Storage OTG Host
- FAT File System
- Ethernet
- LwIP

Device Drivers for the BF52x EZ-KIT Lite

- LCD
- Touch Screen controller
- Keypad

Library and Examples to support the SHARC USB EZ-Extender with the ADSP-2137x processors

New Examples

Lockbox

The ADSP-BF52x and BF54x processors include the new Secure Lockbox Technology (http://www.analog.com/processors/blackfin/lockboxSecureTechnology.html) for Blackfin. Lockbox enables secure execution by providing a secure mode of operation in which only trusted code is allowed to execute. Two new examples have been added to demonstrate this technology. They can be found in the following examples:

Blackfin\Examples\ADSP-BF527 EZ-KIT Lite\lockbox
Blackfin\Examples\ADSP-BF548 EZ-KIT Lite\lockbox

Getting Started Guide

The Getting Started Guide for the BF548 EZ-KIT Lite has been added. This includes 8 easy to use and well documented examples. The examples can be found at the following location:

Blackfin\Examples\ADSP-BF548 EZ-KIT Lite\Getting Started Examples

The documentation can be found in the Hardware Tools Manual under EZ-KIT Lite Evaluation Systems.

Critical Fixes/Changes

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

ADSP-BF522 processor name change

The ADSP-BF522 has been renamed as the ADSP-BF523. Support for this new name will be available in a future update. Those who already created projects for the BF522 and did not use automatically generated LDFs for the ADSP-BF522 may need to rewrite or modify their LDF files in the future.

Two header files for *builtins_support.*h files – TAR 33949

The Blackfin/include/builtins_support.h include file was erroneously part of the VisualDSP++ 5.0 base release. It has been removed in Update 1. If you were including it explicitly in your application source you should <builtins.h> instead.

SSL, USB and DRV libs for ADSP-BF52x not in default LDFs – TAR 34050

The default and generated LDF files for the ADSP-BF52x parts now explicitly link against the system services (libssl527y.dlb), device drivers (libdrv527y.dlb) and USB (libusb527y.dlb) libraries. If you were using Alpha releases of these libraries you would have required project or LDF modifications to link with them. These changes will no longer be required when using default and generated LDFs and should be undone.

Default changed for EBIU_SDBCTL for ADSP-BF533 / LDF – TAR 33491

The default LDFs for ADSP-BF533 prior to Update 1 only populated 32MB of SDRAM (when enabled) unless macro EZKIT_SDRAM_64MB was defined in which case 64MB was used. This has changed in Update 1 to make use of the 64MB SDRAM that is on revisions 1.7 and above of the ADSP-BF533 EZ-KIT Lite. The LDFs now default to use 64MB of SDRAM and for revisions 1.6 of the EZ-KIT Lite and below macro EZKIT_SDRAM_32MB can be defined to revert to using 32MB.

Rename MISCPORT register macros in ADSP-BF52x def file – TAR 33835

Register name changes in the ADSP-BF52x Hardware Reference Manual have also resulted in macro name changes in the various ADSP-BF52x def and cdef headers in \Blackfin\include.

MISCPORT_DRIVE / pMISCPORT_DRIVE
MISCPORT_SLEW / pMISCPORT_SLEW
MISCPORT_HYSTERISIS / pMISCPORT_HYSTERISIS

are replaced with:

```
  NONGPIO_DRIVE / pNONGPIO_DRIVE
  NONGPIO_SLEW / pNONGPIO_SLEW
  NONGPIO_HYSTERESIS / pNONGPIO_HYSTERESIS
```

Sometimes unable to connect to multiprocessor boards – TAR 33968

If a multiprocessor board contains a processor with an unknown silicon revision, the target could not connect in a multi-processor session. This issue has been resolved in this update.

Limitations

This section highlights known significant limitations

VisualDSP++ 5.0 ADSP-BF54x Known Limitations

The following device drivers will be supported in a future update:

- NAND FLASH driver

VisualDSP++ 5.0 ADSP-BF52x Known Limitations

The following device drivers will be supported in a future update:

- NAND FLASH driver

Set memory option fails for NET2272 USB loopback – TAR 34450

The following examples fail after executing the set memory "hostapp –s" when any other option is run:

\Blackfin\Examples\USB-LAN EZ-EXTENDER\USB\bulk_loopback_app
\Blackfin\Examples\USB-LAN EZ-EXTENDER\USB\bulk_redirect_io_app

BF548 EZ-KIT Lite USB drives may need to be formatted – TAR 34633

Before using the USB drives, build and run the format utility found here:

Blackfin\Examples\ADSP-BF548 EZ-KIT Lite\Services\File System\HardDisk\HardDiskFormat

File System rename function does not work – TAR 34561

The adi_fss_FileRename function fails to rename files within the same partition.

LCD driver for the BF527 EZ-KIT Lite needs modification

The ADSP-BF527 EZ-KIT Lite LCD requires that there be at least a 2 PPI CLK delay between the enabling of the frame and horizontal sync signals. In the following 3 files, insert the following lines at the specified:

Blackfin\lib\src\drivers\adc\adi_ad7674.c - line 898

```
        ppi_fs_data.enable_delay = 0;
```

Blackfin\lib\src\drivers\lcd\adi_lcd.c - line 504

```
        FsTmrBuf.enable_delay = 0;
```

```
Blackfin\lib\src\drivers\lcd\nec\adi_nl6448bc33_54.c - line 477
        FsTmrBuf.enable_delay = 0;
```

LCD Example for the BF527 EZ-KIT Lite is missing

There is no example for the ADSP-BF527 EZ-KIT Lite LCD. LCD Drivers exist, but there is no example project on how to use it. This will be available in a future Update.

USB-LAN EZ-Extender examples may fail

When using BF561 EZ-KIT Lite rev 2.0 or 2.1 with ADSP-BF561 revision 0.5 silicon in conjunction with the USB-LAN EZ-Extender, USB and LAN examples may fail to run. To avoid this problem, reduce the SCLK to 100 MHz or lower. To avoid this problem, make the following changes:

On line 105 in Blackfin\Examples\USB-LAN EZ-EXTENDER\USB\bulk_loopback_app\usb_ezkit_utils.c:
#define SCLK 100000000

On line 105 in Blackfin\Examples\USB-LAN EZ-EXTENDER\USB\bulk_redirect_io_app\usb_ezkit_utils.c:
#define SCLK 100000000

On line 105 in Blackfin\Examples\USB-LAN EZ-EXTENDER\USB\mass_storage_app\usb_ezkit_utils.c:
#define SCLK 100000000

ADSP-BF5xx Silicon Anomaly 05-00-0245

The 05-00-0245 anomaly causes hardware errors on speculative loads. The tools workarounds for this anomaly is not enabled for all parts and revisions which are impacted by the anomaly. The missing parts and revisions are:

ADSP-BF54x - all revisions
ADSP-BF52x - revision 0.0
ADSP-BF561 - revision 0.5
ADSP-BF53[123] - revision 0.5

If you enable hardware errors in your application, and are building for one of these parts and revisions you can avoid the 05-00-0245 related hardware errors in the following ways:

Adding "–workaround speculative-loads" to the compiler additional options to enable the compiler workaround when building C and C++ source.
For ADSP-BF53[123] and ADSP-BF561 parts building for silicon revision 0.4 (rather than 0.5) will avoid the anomaly in the compiler generated code and system libraries.

Problem Charts

Problems Addressed

The following table is a list of problems addressed in VisualDSP++ 5.0 Update 1 for which details can be found on the public tools anomaly website. Other problems have also been fixed in the Update.

Details can be found on the Tools Anomaly Web page. The URL is:
http://www.analog.com/processors/tools/anomalies

Product Family	Problem Number	Tool	Description
Blackfin	33976	ADspCommon XML Files	Wrong breakout width for BF52x EBSZ bit field of EBIU_SDBCTL reg
Blackfin	28492	Compiler	csqu_fr16 should be using saturating fractional operations
Blackfin	33786	Compiler	Use of -overlay can result in compiler assert (bitset.c:67)
Blackfin	33963	Compiler	Compiler doesn't count inline asm length towards size of hw loop
Blackfin	33403	CRTGen	Generated cplbtab file unusable
Blackfin	33405	CRTGen	CPLB_D_PAGE_MGMT used indiscriminately in gen'd BF535 cplbtab
Blackfin	33808	Emulator	BF533 POST: Release mode sets Loader width to 8 bit
Blackfin	33803	Examples	ADSP-BF537\Drivers\UART\AutoBaud readme has no jumper settings
Blackfin	34185	Examples	The host side of the Inetd does not build for release build
Blackfin	34198	Examples	USB-LAN EZ-Extender board examples may fail to run correctly
Blackfin	34061	Flash Programmer	Error in ReadData for BF548 flash driver
Blackfin	33049	IDDE	Loading DWARF3 debugging information may crash VisualDSP++
Blackfin	31695	LDF	data1 is mapped before L1_bsz
Blackfin	34059	LDF	p1 and -p2 do not work with default LDFS (5.0)
Blackfin	33652	LDFGen	Stack in mem covered by cplb data table entry in WB mode problem
Blackfin	33722	LDFGen	Two output sections with the same name are generated
Blackfin	33178	Run Time Libraries	Remove NWIDTH in NFC_CTL bit definitions for ADSP-BF52x devices
Blackfin	33654	Run Time Libraries	DSP library function conv2d3x3_fr16() based on wrong algorithm
Blackfin	33733	Run Time Libraries	Disable_data_cache() does not work
Blackfin	33744	Run Time Libraries	Incorrect macro names for HOSTDP masks in BF52x/BF54x headers
Blackfin	33792	Run Time Libraries	Remove PORT_MUX from ADSP-BF52x def/cdef headers
Blackfin	33825	Run Time Libraries	PPI_STATUS missing bit masks for ADSP-BF52x
Blackfin	33835	Run Time Libraries	Rename HYSTERESIS / MISCPORT_* register macros in ADSP-BF52x hdr

Blackfin	33901	Run Time Libraries	Including before "vdk.h" will result in an error
Blackfin	34112	Run Time Libraries	BF561 - Memory Initializer will not initialize external SDRAM
Blackfin	33627	TCPIP Stack	Corrupted BF537 EZ-KIT proj LAN\Host\FILESERVER\FileServer.dsp
Blackfin	33843	TCPIP Stack	BF USB LAN Extender Examples and library do not work
SHARC	33968	Emulator	Unable to connect to Multi-processor boards
SHARC	33938	Hardware Board	21369 EZ-KIT Lite SPI Flash support changing
SHARC	33671	Run Time Libraries	MTM registers missing from cdef21364.h
SHARC	34118	Run Time Libraries	CYCLE_COUNT_* macros can give wrong results with optimization
SHARC	33887	Simulator	Reg modify then write to ext mem writes old reg value
TigerSharc	28363	Compiler	Functions with #pragma weak_entry can be inlined
TigerSharc	32429	Compiler	Internal error: diag_message: missing string substitution
TigerSharc	34022	VDK	VDK API level check can cause false positive Kernel Panic

Known Problems

Details can be found on the Tools Anomaly Web page. The URL is:

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

VisualDSP++® 5.0 Release Notes

2007 August 28

Table of Contents

VisualDSP++ 5.0 Documents

Release Notes

This document provides the Release Notes for the VisualDSP++® 5.0 release.

Product Release Bulletin

Your primary source of information for the VisualDSP++ 5.0 Release is the Product Release Bulletin manual in .pdf format that accompanies this release.

Documentation Set

The complete set of documentation in .pdf format is provided on the VisualDSP++ Installation CD. The manuals are available in .chm online Help format in the installation.

Additional information is available online in the Technical Library:

http://www.analog.com/processors/technicalSupport/technicalLibrary

Licensing Guide

The VisualDSP++ 5.0 Licensing Guide is a new document that describes how to manage your license for VisualDSP++ software. For users who purchase floating licenses, the guide describes the VisualDSP++ Floating License Server.

Note: The Licensing Guide does not describe versions of VisualDSP++ licensing prior to VisualDSP++ 5.0. For information about older versions, refer to:

Help –> Contents –> Assistance -> Software License Management.

The VisualDSP++ License Installation Procedure is also available on the Analog Devices Web site on the “Upgrades Archives” page, available at:

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

ADI ELF Documentation

If you have tools that consume the ELF object files produced by VisualDSP++, the following document will be of interest. Most VisualDSP++ 5.0 users need not be concerned with this level of detail.

VisualDSP5_0_ADI_ELF_Changes.pdf

The ADI ELF document covers the most recent changes in the ADI ELF since VisualDSP++ 4.5 was released. Updated versions of the complete ADI ELF ABI specification (general and processor-specific) are available from Customer Support by request.

Problem Reports

Charts summarizing the problems fixed in this release and the known open problems are included at the end of this document.

Project Upgrades

We recommend working with a copy of your existing applications when first upgrading to the VisualDSP++ 5.0 release. The upgrade will change existing *.dpj projects and in some instances, the Project Wizard will prompt for regeneration of the LDF and startup code. These upgrade changes are covered in more detail in the following two sub-sections.

VisualDSP++ 5.0 .dpj Projects Have New Format

The format of VisualDSP++ .dpj projects has changed from previous releases and the new VisualDSP++ 5.0 format is not backwardly compatible. At the time VisualDSP++ 5.0 reads an older generation project, the IDDE will provide a pop-up asking if it can convert the project to the new format. It will save the pre-existing version in 'MyProject.dpj.bak' and the VisualDSP++ 5.0 version becomes 'MyProject.dpj'.

If you would like to keep working with VisualDSP++ 4.5 without any changes to your application and/or projects, make a copy of your application for use with the VisualDSP++ 5.0 version.

Project Wizard Template Changes – Blackfin

If you have a project that was generated with the Project Wizard, loading the project after installing VisualDSP++ 5.0 may result in a pop-up requesting regeneration of the code/LDF.

Regeneration affects three files:

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

After regeneration, you will be current with the latest improvements in the templates.

If you would like to keep working with VisualDSP++ 4.5 without any changes to your application and/or projects, follow the recommendation in the previous section and make a copy of your application for use with VisualDSP++ 5.0.

Project Wizard template changes include:

- TAR 31346: Shared data, locks, etc. need to be non-cached
- TAR 31938: inputs sections for tables require FORCE_CONTIGUITY
- TAR 32725: Workaround comment incomplete in generated LDFs

TAR 31346: dual-core (ADSP-BF561) applications, in order for shared data and locks to be correctly accessed by each core, that data must not be allowed to be cached. It has been the case that LDFs and CPLB tables generated by the Project Wizard did not respect that requirement. That problem has been fixed.

TAR 31938: The linker will not guarantee contiguous placement of sections unless the FORCE_CONTIGUITY operator is used. If you have table inputs in your LDF that require contiguous placement, these should be mapped in a separate memory output section using FORCE_CONTIGUITY. In VisualDSP++ 5.0 the default LDFs have been modified to reflect this. More information on the FORCE_CONTIGUITY can be found in the "Linker and Utilities" manual.

TAR 32725: In LDFs generated by the Project Wizard, there is a particular section of code that works around two silicon anomalies: 05-00-0189 and 05-00-0310. However, the comment for that section of code only mentions 05-00-0189. If a user believes that 05-00-0189 does not apply, the user may remove that section of code, only to run into problems because 05-00-0310 does indeed apply. To avoid this possibility, the comment for that section of code has been corrected.

Processor-Specific Release Notes

New Blackfin Processor Support

The Product Bulletin contains the list of new processors available at VisualDSP++ 5.0. Refer to the processor's data sheet and hardware reference manuals for information on system configuration, peripherals, registers, and operating modes. The following are new Blackfin® processors:

- ADSP-BF542, ADSP-BF544, ADSP-BF548, ADSP-BF549
- ADSP-BF522, ADSP-BF525, ADSP-BF527

Ignore any mention of the ADSP-BF541. It does not exist, but is reserved for future use and references to it may appear in some places.

VisualDSP++ 5.0 ADSP-BF54x Known Limitations

The following device drivers are not yet available:

- NAND
- Mass Storage Host (USB)

VisualDSP++ 5.0 ADSP-BF52x Support

Emulator support and the EZ-KIT Lite® debug agent are provided for the ADSP-BF52x parts. VisualDSP++ 5.0 provides the tools required to build and debug ADSP-BF52x code.

The ADSP-BF522/ADSP-BF525/ADSP-BF527 Blackfin Embedded Processor Preliminary Data Sheet is located here:

http://www.analog.com/processors/blackfin/technicalLibrary/dataSheets.html

VisualDSP++ 5.0 ADSP-BF52x Known Limitations

These are the known limitations specific to the new Blackfin ADSP-BF52x processors:

- The System Service Libraries are not yet available
- The Device Driver Libraries are not yet available
- LwIP support is not yet available
- ADSP-BF527 EZ-KIT Lite example set is not yet available
- The Blackfin ADSP-BF52x Hardware Reference Manuals are not included in VisualDSP++ 5.0
- No online help for the ADSP-BF52x Hardware Reference Manuals

Compiler Release Notes

Compiler Assumes Strong Alignment of Global Arrays / TAR 33540

For performance reasons, the compiler explicitly aligns arrays at global scope, which allows the compiler to vectorize accesses to the array. For example:

```
char glob_array[BYTECOUNT] = { /* data */ }; // aligned on a 4-byte boundary.
```

The compiler assumes that externally-defined arrays will also be aligned in this manner:

```
extern char ext_array[];  // compiler assumes aligned on a 4-byte boundary.
```

If such arrays are defined in other C files, this will be the case. If, however, you define such arrays in assembly source, you must ensure that they are suitably aligned, otherwise run-time exceptions are possible.

For example:

```
    .GLOBAL _unsafe_array;
    .TYPE _unsafe_array,STT_OBJECT;
    .BYTE _unsafe_array[100];    // no alignment - misaligned access possible
    .ALIGN 4;
    .GLOBAL _safe_array;
    .TYPE _safe_array,STT_OBJECT;
    .BYTE _safe_array[100];      // 4-byte aligned - access is safe
```

Simulator Release Notes

Limitations -- Blackfin

The following is a list of supported peripherals in the Blackfin simulators:

Core Peripherals

All Blackfin Processors:

- Data Cache & SRAM Memory
- Instruction Cache & SRAM Memory
- Event/Interrupt Controller Registers
- Core Timer Registers
- Trace Buffer Registers
- Watchpoint Control Registers
- Performance Monitor Registers

System Peripherals

All Blackfin processors:

- PLL Registers
- CHIPID
- RTC Registers
- System Timers
- System Interrupt Controller (SIC)
- DMA
- MDMA
- UART
- SPORT

ADSP-BF535 also includes:

- SYSCR
- Watch Dog Timer
- PCI
- GPIO
- SPI

All Blackfin Processors NOT including the ADSP-BF535 also have:

- PPI
- EBIU - Full MMR support on MP Processors. Single Core only has SRAM support

Note: The ADSP-BF54x processors have a limited list of Core and System peripherals that are supported:

- Data Cache & SRAM Memory
- Instruction Cache & SRAM Memory
- Event/Interrupt Controller Registers
- Core Timer Registers

- Trace Buffer Registers
- Watchpoint Control Registers
- Performance Monitor Registers
- PLL Registers
- CHIPID
- RTC Registers

System Services Release Notes – Blackfin

Silicon Anomaly (05-00-0311)

The previous compiler workaround for this anomaly has been deemed unsafe and removed from this release. As such the Programmable Flag service no longer relies on the compiler to workaround this anomaly. Therefore in this VisualDSP++ 5.0 release, the Programmable Flag service, in conjunction with the Interrupt Manager service, collectively workaround this anomaly in a safe fashion. All versions of the System Service Libraries for Blackfin processors that could potentially be affected by this anomaly inherently work around the anomaly. Users of the System Services do not need to take any action other than simply linking with the appropriate System Services library as usual. Users of the System Services do not need to include the file "sys/05000311.h" nor do they need to use the FIO_ANOM_0311_XXX macros (unless they are accessing the flag MMRs directly).

See below: "Noteworthy VisualDSP++ 4.5 Update Changes: 05-00-0311" section for further details.

SDH Driver Corrupts Directory Structures for Write Operations / TAR 33464

The Secure Digital Host (SDH) driver is currently only cleared for read-only access to SD cards inserted into the SD slot on the ADSP-BF548 EZ-KIT Lite development board.

Note: This problem has been identified as a symptom of anomaly 05-00-0340 that is planned to be fixed in Rev 0.1 silicon.

adi_pwr_SetPowerMode() Does Not Help Transition from SLEEP / TAR 33518

The Blackfin System Services power management function *adi_pwr_SetPowerMode()* does not currently support a transition from SLEEP or DEEP SLEEP into any other mode. Upon wakeup from SLEEP or DEEP SLEEP, a call to *adi_pwr_SetPowerMode()* will fail. The function was not written to support either of these transitions is because upon wakeup, the processor transitions automatically from SLEEP or DEEP SLEEP into the FULL_ON or ACTIVE mode, depending on the status of the BYPASS bit, so it was assumed that this function call was not necessary. This assumption was correct with regard to the transition from DEEP SLEEP. But the problem is that when transitioning from SLEEP, the STOPCK bit is NOT automatically cleared, the same way it is cleared upon wakeup from DEEP SLEEP. The core clock is enabled, but the STOPCK bit does not reflect this. The application must explicitly clear the STOPCK bit upon wakeup to resume running, or else a subsequent read-modify-write of PLL_CTL followed by the IDLE sequence can put the processor back to sleep.

In VisualDSP++ 5.0 Update 1, the *adi_pwr_SetPowerMode()* function will be modified to facilitate the transition from SLEEP mode to ACTIVE or FULL-ON mode. The function will update the appropriate register values to complete the transition from SLEEP mode.

As a workaround, the following code can be used clear the STOPCK bit manually, upon wakeup from the SLEEP mode, enabling the application to resume successfully:

```
u16 PLLCtlVal = *pPLL_CTL;

PLLCtlVal &= 0xFFF7;

*pPLL_CTL = PLLCtlVal;
```

A subsequent call to *adi_pwr_GetPowerMode()* will then reflect the correct power mode.

File System Corruption When Number of Files Exceeds One Cluster / TAR 33677

A known issue with the ADI FAT File System Driver is that when more file entries are created in a directory than there is space available with one cluster directory, corruption may occur as subsequent clusters are not zeroed before use. For the hard disk attached to the ADSP-BF548 EZ-KIT Lite development board, formatted as a 32GB FAT 32 partition, this limitation equates to 512 short name (8.3) entries per cluster. Please note that deleting files does not alleviate the issue.

Additional System Service Library Documentation

In the VisualDSP++ 5.0 installation directory, is a subdirectory called "…/Blackfin/docs/services". This subdirectory contains updated documentation for the EBIU and Dynamic Power system services. In addition, this subdirectory contains new documentation for the File System Service and the Real-Time Clock service.

Device Driver Release Notes – Blackfin

Additional Device Driver Documentation

In the VisualDSP++ 5.0 installation directory, is a subdirectory called "…/Blackfin/docs/drivers". This subdirectory contains detailed documentation for each device driver. Within each subdirectory is detailed information describing each driver including the dataflow methods it supports, command IDs, return codes, configuration issues, etc.

Included in the USB documentation subdirectory is a porting guide document. This document describes the application changes necessary to migrate an application using the USB device driver provided in VisualDSP++ 4.5 to the newer USB driver provided with VisualDSP++ 5.0. It is very strongly recommended that all USB users refer to this document.

Emulator Release Notes

Customizing XML Register Reset Values

The **Use XML Reset Values** target option relies on the register reset definitions defined in the XML files in the *<install-dir-5.0>\System\ArchDef* directory. The list of register names and reset values are extracted from the XML block:

```
<register-reset-definitions>
        …
</register-reset-definitions>
```

that is located within the XML files for that processor's EZ-KIT Lite. For the TigerSHARC® processors, the register resets are located in the ADSP-TS*-resets.xml files. For the Blackfin® and SHARC® processors, the register resets are located within the *-proc.xml files.

In previous releases, the only method for overriding the XML reset values for custom boards was to edit the system XML files directly. If you had more than one custom board, you needed to rename the XML file to a known processor name prior to use.

At VisualDSP++ 5.0, you no longer need to make edits to the XML register resets in the shipped versions or manage multiple boards by renaming files. The new Custom Board Support includes a feature that enables you to specify register reset values for your custom boards in separate XML files, with names and locations of your choice. For details, refer to "Custom Board Support" within "Graphical Environment" in the VisualDSP++ 5.0 online Help.

Noteworthy VisualDSP++ 4.5 Update Changes

If you have kept current to the VisualDSP++ 4.5 2007 June update, skip this section.

Incorrect Memory Mapping for ADSP-21375 / TAR 31816

TAR 31816: Incorrect memory mapping for ADSP-21375

The memory map for the ADSP-21375 SHARC processor has been corrected throughout the tools, including the linker and the default LDFs. This was fixed in the VisualDSP++ 4.5 June update. 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 the VisualDSP++ 4.5 June 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 is 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 have 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).

Former Workaround for 05-00-0311 is Not Safe – Blackfin TAR 32344

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 was 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 the anomaly 05-00-0311 workaround, the macro __WORKAROUND_FLAGS_MMR_ANOM_311 is defined at compile, assemble, and link stages.

05-00-0311

Anomaly 05-00-0311 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;
      ...
}
```

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);
      ...
}
```

Note: System Service Libraries are anomaly safe for 05-00-0311. See above: “System Services Release Notes: Silicon Anomaly (05-00-0311)” section.

For more information on anomaly 05-00-0311, see the appropriate errata sheet, which can be downloaded from

<http://www.analog.com/processors/blackfin/support/ICanomalies.html>.

Problem Charts

Problems Addressed

The following table is a list of the problems addressed in the VisualDSP++ 5.0 release.

Details 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	24089	Compiler	Generates bad code for old style C args with -double-size-64
All	24929	Compiler	#pragma pack doesn't work as expected with bitfields
All	25649	Compiler	Compiler crashes if a 64-bit float variable is used in an asm
All	26325	Compiler	Speed/size ratio inlining warning gives wrong source line
All	28566	Compiler	Alternate pre-processing sequences cause error with -pedantic
All	28684	Compiler	Multiple PGO files confuses IPA
All	28814	Compiler	-MQ switch crashes driver
All	29484	Compiler	The "optimize" pragmas do not override -Og
All	29611	Compiler	Compiler switch -s does not work
All	29617	Compiler	Assertion (macdefs.c:2475) with extremely long variable names
All	29660	Compiler	Pre-compiled headers doesn't work with IPA and VDK
All	29860	Compiler	#pragma alignment_region modified by prior extern statements
All	29910	Compiler	#pragma always_inline in system headers can cause a warning
All	29952	Compiler	Compiler doesn't recognize -1,0 as fract literal.
All	30096	Compiler	Circular buffer loops containing fn pointers don't zero Lregs
All	24335	IDDE	Unnecessary silicon revision warning
All	27965	IDDE	Default for new projects should be std:: enabled
All	28592	IDDE	getTargetFileNameList returns bad filenames
All	31938	LDF	Inputs sections for tables require FORCE_CONTIGUITY
All	24204	Linker	Pragma align can lead to wasteful memory allocation
All	28389	Linker	No way to map anything after PLIT
All	28541	Run Time Libraries	Cycle counting macros fail to compile in conditional statements
All	28599	Run Time Libraries	printf ignores the 'h' length modifier with %o, %x, and %X
All	8136	Utilities	elfdump doesn't flag error when archive(object) doesn't exist
All	29569	VDK	RunLastTime in VDK Status is displaying the wrong figure
Blackfin	29189	ADspCommon XML Files	DMA register names have an extra number in the name
Blackfin	30604	ADspCommon XML Files	BF561 has RTC window / register defs. These should be taken out.
Blackfin	26076	Compiler	WDOG_DISABLE not defined in defBF53{2|4|8}.h and defBF561.h
Blackfin	28145	Compiler	label displayed at wrong address
Blackfin	28483	Compiler	#pragma no_alias is too strict
Blackfin	28898	Compiler	includes in UNC/shares not found
Blackfin	29099	Compiler	Debug info associated with wrong line of C++ source code.
Blackfin	30547	Compiler	BF shift-with-clipped-shift-distance builtins literal inconsistencies
Blackfin	30554	Compiler	Local variables totalling >64KB can result in internal error
Blackfin	30886	Compiler	Using "n" asm constraint results in compiler error

Blackfin	31849	Compiler	The complex fract function csqu_fr16 doesn't work
Blackfin	32823	Compiler	abs saturates even with -no-saturation
Blackfin	32858	Compiler	Callee function doesn't truncate parameter to expected type (K&R C)
Blackfin	32904	Compiler	internal compiler error in peephole.c:2387
Blackfin	32910	Compiler	"Buffer overrun detected" error message from linker
Blackfin	32939	Compiler	Short names for video functions being defined with -no-builtins
Blackfin	33668	Compiler	Fatal error in do_expr()
Blackfin	30796	Emulator	HPPCI-ICE does not work under OEM Windows Vista
Blackfin	22657	IDDE	Backslash causes problems for assembler property page
Blackfin	23285	IDDE	Cannot export from VDK State History pane top-bar
Blackfin	27938	IDDE	Random license failure when building using a floating license.
Blackfin	28150	IDDE	Cannot view defined static member variable
Blackfin	28197	IDDE	LDFGen ignores multicore settings
Blackfin	28404	IDDE	User-corrupted VDK history data/window can crash Idde
Blackfin	28891	IDDE	Startup code/LDF wizard doesn't warn when overwriting LDF file
Blackfin	28902	IDDE	Additional include dirs from 3.0 or earlier project settings
Blackfin	28922	IDDE	Show tabs now also show spaces
Blackfin	28975	IDDE	Creating a new TCPIP project pops up message about replacing srcs
Blackfin	29017	IDDE	$(VDSP) not expanded when just building one file in a project.
Blackfin	29087	IDDE	Thread Types missing from Threads in VDK Status Window
Blackfin	29384	IDDE	Go To in BTC Memory window causes Runtime - Abnormal termination
Blackfin	29605	IDDE	VDK Status window Event Bit display error
Blackfin	29846	IDDE	Doubles not fully displayed when -double-size-64
Blackfin	30494	IDDE	Whole word replacement does not work with undercores
Blackfin	31336	IDDE	Two elements allowed to be placed at the same location
Blackfin	32024	IDDE	Trace window does not display all of its entries
Blackfin	32038	IDDE	Expert Linker crashes when opening LDF file
Blackfin	32067	IDDE	ADspStreamList Add* methods don't work for BF561
Blackfin	32620	IDDE	C++ NMI interrupt handler does not end with an RTN
Blackfin	30349	Installation	msxml3.dll registration problems prevent install
Blackfin	31346	LDFGen	shared data, locks etc need to be non-cached
Blackfin	32725	LDFGen	Workaround comment incomplete in generated LDFs
Blackfin	30935	Linker	Cannot jump-call expand PLIT?
Blackfin	22930	Run Time Libraries	C++ exception handling may not work with spilled sections.
Blackfin	28518	Run Time Libraries	Interrupt dispatcher does not include 05-00-0071
Blackfin	28558	Run Time Libraries	32-bit signed division wrong for inputs near INT_MIN
Blackfin	28965	Run Time Libraries	min_fr1x16 and max_fr1x16 missing
Blackfin	29525	Run Time Libraries	adi_core_b_enable() unresolved in assembly
Blackfin	30752	Run Time Libraries	CPLB Manager can cause double exception
Blackfin	31869	Run Time Libraries	meminit support fails to for ZERO_INIT when stack in scratchpad
Blackfin	31881	Run Time Libraries	FLT_MAX not a float literal
Blackfin	32864	Run Time Libraries	DMA32 bit in PPI erroneously appears in single core def headers
Blackfin	32911	Run Time Libraries	mulfl64.asm in release not same as used to build library
Blackfin	28099	Simulator	BF535: crash running attached DXE in BF535 CAS
Blackfin	29583	Simulator	Self-Nesting Interrupts not supported in Blackfin BF533 CAS
Blackfin	30628	Simulator	32 bit registers in EBIU only accept 16 bit writes on a BF561.

Blackfin	31895	Simulator	Size information for all of the caches show up as "0" Kbytes
Blackfin	28946	TCPIP Stack	LwIP Project does not accept broadcast traffic in VDSP 4.5
Blackfin	30450	VDK	Contradictory information provided for popping regions
Blackfin	30991	VDK	VDK does not handle all the exceptions that the cplb_hdr does
Blackfin	32156	VDK	Enabling self-nested interrupts breaks VDK
SHARC	28087	ADspCommon XML Files	REVPID register displays PROCID and SIREV swapped
SHARC	20526	Compiler	Annotation information is incorrect for registers clobbered by an asm
SHARC	28993	Compiler	-pedantic should put out warnings
SHARC	29964	Compiler	C/C++ runtime not honored on SHARC.
SHARC	31767	Compiler	Compiler not working around 2136x anomaly (07-00-0009)
SHARC	32198	Compiler	Asm statements using circ buf regs don't work
SHARC	29246	Examples	21262 AsmDemo / CDemo BTC example README files need correction
SHARC	25305	IDDE	Additional options lost converting 3.0 dpj to 4.0
SHARC	31421	IDDE	Zooming in the plot window may cause the IDDE to crash
SHARC	29802	LDF	LDF can allow 1 too many words to be assigned to heap
SHARC	28074	Linker	No output sections issued when "empty" with run spaces
SHARC	30078	Run Time Libraries	delete operator doesn't work with heap_install
SHARC	31200	Run Time Libraries	Multi-threaded realloc() will not allocate correct amount of mem
SHARC	31850	Run Time Libraries	heap_malloc with nonexistant heap causes invalid data accesses
SHARC	33303	Run Time Libraries	Bit macro FAR changed to FARF for SDCTL register in def header
SHARC	29900	Utilities	Mem21k update generates "1" exit code, but seems to work anyway
SHARC	30460	Simulator	ADSP-21375 Primes example does not work in simulator
SHARC	32673	Simulator	Hang executing from ext mem consecutive reads from ext mem
TigerSHARC	32115	ADspCommon XML Files	-workaround all does not turn on all workarounds
TigerSHARC	22672	Assembler	Assembler accepts invalid register move
TigerSHARC	31832	Assembler	Symbols sizes for .INC/BINARY wrong
TigerSHARC	32041	Assembler	Invalid warning on 2nd .section directive
TigerSHARC	29096	Compiler	Confusing annotations for compiler-generated fp-divide code
TigerSHARC	32803	IDDE	Porting a VisualDSP++ 4.5 project causes different libraries to be linked in
TigerSHARC	29735	Run Time Libraries	strtol and strtoul error for garbage bases
TigerSHARC	29836	Run Time Libraries	libsim for TS101 rev 0.4 is incomplete

Known Problems

The following table is a list of known problems in VisualDSP++ 5.0.

Details 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	30713	Compiler	Compiler is not using BSS
All	32429	Compiler	Internal error: diag_message: missing string substitution
All	33665	Compiler	"internal compiler error / driver.c:1488" building VLA source

All	28272	Run Time Libraries	C++ library code linked in with -rtti is bigger than 4.0
All	32303	Run Time Libraries	cycle_t function return types causes compiler warning cc0815
All	32092	IDDE	int PrimIOCB; means no output, multiply defined sym or software exception
All	31923	Compiler	Compiler driver accepts illegal -flags-* options
All	32237	Compiler	Workaround switches don't match annotations
Blackfin	32004	Assembler	Inconsistent Assembler behavior with integer constants
Blackfin	28572	Compiler	BF535: float div returns small denorm result when zero expected
Blackfin	29394	Compiler	-Wremarks doesn't always warn about deprecated switches
Blackfin	29851	Compiler	-section does not apply qualifiers
Blackfin	29870	Compiler	ISR problems with SAVE_REGS functionality
Blackfin	29874	Compiler	-no-builtin switch causing failures
Blackfin	30247	Compiler	section __attribute__ does not work as documented
Blackfin	32466	Compiler	C++ template instantiations ignore #pragma uses
Blackfin	32299	Compiler	volatile store inputs also treated as volatile when unnecessary
Blackfin	32749	Compiler	Slowdown of code using division when build -Os
Blackfin	33643	Compiler	Keywords such as section cause spurious errors in MISRA mode
Blackfin	33721	Compiler	still possible to write bad context sensitive __builtin_aligned
Blackfin	33403	CRTGen	Generated cplbtab file unusable
Blackfin	33405	CRTGen	CPLB_D_PAGE_MGMT used indiscriminately in generated BF535 cplbtab
Blackfin	30369	Debug Agent	Debug agent scans too fast [can cause external memory issues]
Blackfin	32752	Debug Agent	IceTest fails on RoHS EZ-KIT Lite's using USB 2.0 HUB
Blackfin	24859	Device Driver	Autobauding fails at 38,400bps on the 561 only
Blackfin	26184	Device Driver	UART autobaud timer selection
Blackfin	27061	Device Driver	Memory size given to adi_dev_Init() must be larger than expected
Blackfin	29791	Device Driver	PPI Error Callbacks
Blackfin	31608	Device Driver	Error Interrupt Side Effects
Blackfin	30087	elf2flt	Bad relocations out of elf2flt when no code
Blackfin	33691	Examples	ADSP-BF561 POST does not stop at main on load with 1.3 rev EZ-KIT Lite
Blackfin	27445	IDDE	Step over/out doesn't work in flash
Blackfin	27685	IDDE	Deleted SW breakpoints keep re-appearing after load
Blackfin	28755	IDDE	After selecting text, print from source window prints entire file
Blackfin	29687	IDDE	Terminal font doesn't work in the source window
Blackfin	29727	IDDE	static member of class not resolved in expressions window
Blackfin	31238	IDDE	No warning on Memory Fill/Dump outside valid memory
Blackfin	31720	IDDE	Various tool switches reported as not enabled via automation
Blackfin	32578	IDDE	Should keep the paths in the additional include as absolute
Blackfin	32665	IDDE	File-specific compile options do not take defaults from existing
Blackfin	32312	IDDE	License not migrated when installing under Windows Vista
Blackfin	32957	IDDE	F2 does not rename VDK items
Blackfin	33049	IDDE	Loading DWARF3 debugging information may crash VisualDSP++
Blackfin	33680	IDDE	Changing project options may overwrite working LDF
Blackfin	31173	Installation	Install_CL does not handle VC2005 SP1 update
Blackfin	33057	Installation	Unknown publisher warnings during 5.0 Installation on Vista
Blackfin	28515	LDF	Issues with the 2-link approach for BF561 projects
Blackfin	31695	LDF	data1 is mapped before L1_bsz
Blackfin	29902	LDFGen	Project fails to build when user sets heap space to less than 1k
Blackfin	32747	LDFGen	LDFGen doesn't sufficiently support run-from-flash
Blackfin	33652	LDFGen	Stack in mem covered by cplb data table entry in WB mode problem
Blackfin	33722	LDFGen	Two output sections with the same name are generated
Blackfin	29565	Loader	Wrong assignment in the ADSP-BF537 Init file
Blackfin	29065	Run Time Libraries	Hyperbolic Functions do not return Inf when called with Inf arg
Blackfin	29221	Run Time Libraries	Multicore runtime libraries always link in I/O library

Blackfin	32179	Run Time Libraries	VDK and RTL link in different libs for -si-revision none -workaround
Blackfin	32319	Run Time Libraries	crtn.doj can be removed from .LDF File without warning
Blackfin	32867	Run Time Libraries	Make the header files MISRA compliant
Blackfin	33654	Run Time Libraries	DSP library function conv2d3x3_fr16() based on wrong algorithm
Blackfin	33733	Run Time Libraries	disable_data_cache() does not work
Blackfin	33744	Run Time Libraries	Incorrect macro names for HOSTDP masks in BF52x def header
Blackfin	28581	System Services	adi_pwr_SetFreq() locks up sometimes on ASDP-BF561
Blackfin	31568	System Services	Add command to sense the PERIOD register for GP timers
Blackfin	32230	System Services	Add command to sense GP timer period
Blackfin	33464	System Services	SDH driver corrupts directory structures for write operations Note: This problem has been identified as a symptom of anomaly 05-00-0340 that is planned to be fixed in Rev 0.1 silicon.
Blackfin	33518	System Services	pwr mgmt to facilitate transition from SLEEP
Blackfin	33677	System Services	File System Corruption when number of files exceeds 1 cluster
Blackfin	29313	TCPIP Stack	ETHARP_ALWAYS_INSERT option is deprecated in lwIP
Blackfin	29736	TCPIP Stack	Multiple network interface problem
Blackfin	30157	TCPIP Stack	lwip send function returns bytes sent, but sends only 64K max
Blackfin	32362	TCPIP Stack	getsockopt() with SO_ERROR does not return error
Blackfin	33007	TCPIP Stack	INETD example should not set the user_data_ptr in the header
Blackfin	33627	TCPIP Stack	Corrupted ADSP-BF537 EZ-KIT Lite in Blackfin\Examples (patch available) ADSP-BF537 EZ-KIT Lite\LAN\Host\FILESERVER\FileServer.dsp
Blackfin	32949	USB Stack	Intermittent USB connectivity on ADSP-BF548 EZ-KIT Lite
SHARC	32920	Compiler	PCH fails with cc0219 on Vista
SHARC	29561	Emulator	VisualDSP++ disconnect if Sport DMA Addressing debug window open
SHARC	32810	Emulator	Incorrect display of instructions in external memory on Sharc
SHARC	33574	IDDE	Value of float pointers displayed in unexpected format
SHARC	32706	Run Time Libraries	Increase in printf footprint
SHARC	32881	Run Time Libraries	Thread-safe time library ctime() problem
SHARC	33670	Run Time Libraries	SIG_MTM to be defined for ADSP-21362/3/4/5/6
SHARC	33671	Run Time Libraries	MTM registers missing from cdef21364.h
TigerSHARC	28363	Compiler	Functions with #pragma weak_entry can be inlined
TigerSHARC	32961	Compiler	Use of setjmp/longjmp incompatible with compiler optimizations
TigerSHARC	33655	Compiler	link error: __memzero could not be resolved
TigerSHARC	30749	Emulator	Halting single proc during MP run halts both processors
TigerSHARC	28195	Run Time Libraries	Compiler fails using some library functions prefixed with std::
TigerSHARC	29110	Run Time Libraries	-fp-div-lib doesn't work when compiling Inf/NaN with -ve value
TigerSHARC	32626	Run Time Libraries	ADSP-TS201 BTB not enabled by default in the boot process

TigerSHARC	32758	Run Time Libraries	namespace std does not contain builtins
TigerSHARC	32782	Run Time Libraries	DSP real vector functions can raise FP exceptions
TigerSHARC	27911	Simulator	ADSP-TS203 session displays the CLU registers